Agilent Technologies

Advanced Design System 2011.01

Feburary 2011
Examples

Application Examples

Detailed application examples of how ADS can be used to solve real-life problems.

- *A-to-D D-to-A Applications Guide* (examples)
- *Budget Analysis Application Guide* (examples)
- *Load-Pull Simulations* (examples)
- *Radar Applications Guide* (examples)
- *Signal Integrity Applications* (examples)
- *VPI ADS Link* (examples)
- *Wireline Applications* (examples)

A-to-D D-to-A Applications Guide

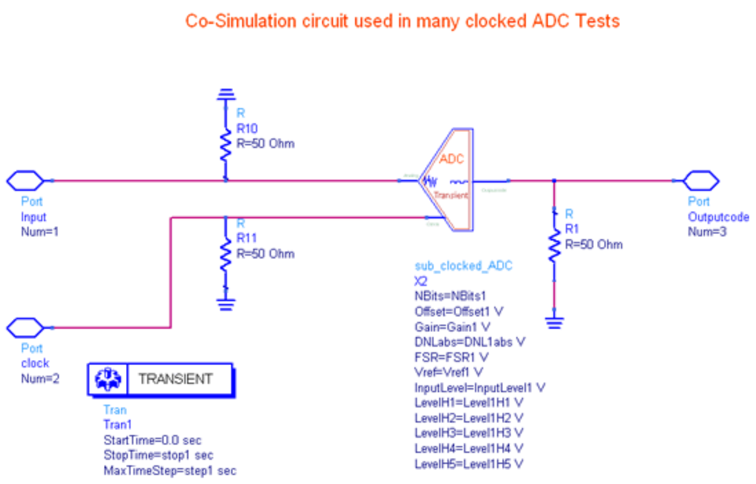
The Analog-to-Digital Converters Application Guide is accessible from the schematic window under the **DesignGuide** menu.

Objective

The objective of the A-to-D D-to-A Applications guide is to demonstrate the capability of Advanced Design System to design Analog to Digital and Digital to Analog converters.

The DUT blocks used in this application guide can be replaced with circuit blocks for simulation after making some simulation setup and parameter adjustment to support the replacing DUT.

Figure 1: Clocked ADC Schematic

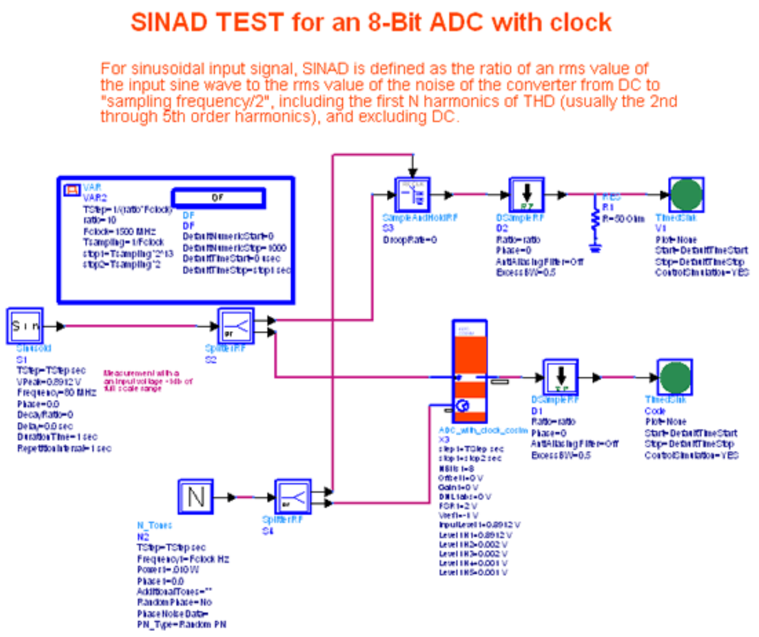


ADC Tests

In these series of test templates, ADCs can be fully characterized. Two types of ADC models are used: with clock and without clock. Those without a clock use the simulator time step to sample the input analog. ADCs can be fully characterized with the following tests.

- "Test_ADC_with_clock_DNL" and "Test_ADC_without_clock_DNL" testDifferential Non Linearity (DNL)
- "Test_ADC_with_clock_INL" and "Test_ADC_without_clock_INL" testIntegral Non linearity (INL)
- "Test_ADC_with_clock_SNR" and "Test_ADC_without_clock_SNR" test Signalto Noise Ratio (SNR)
- "Test_ADC_with_clock_SINAD" and "Test_ADC_without_clock_SINAD" testSignal Noise and Distortion (SINAD)
- "Test_ADC_without_clock_OffsetError" tests Offset Error
- "Test_ADC_without_clock_Gain Error" tests Gain Error
- "Test_ADC_without_clock_THD" tests Total Harmonic Distortion (THD)
- "Test_ADC_without_clock_SFDR" tests Spurious-free Dynamic Range(SFDR)
- "Test_ADC_without_clock_IMD" tests Intermodulation Distortion (IMD)

Figure 2: SINAD Test for 8-Bit ADC with Clock

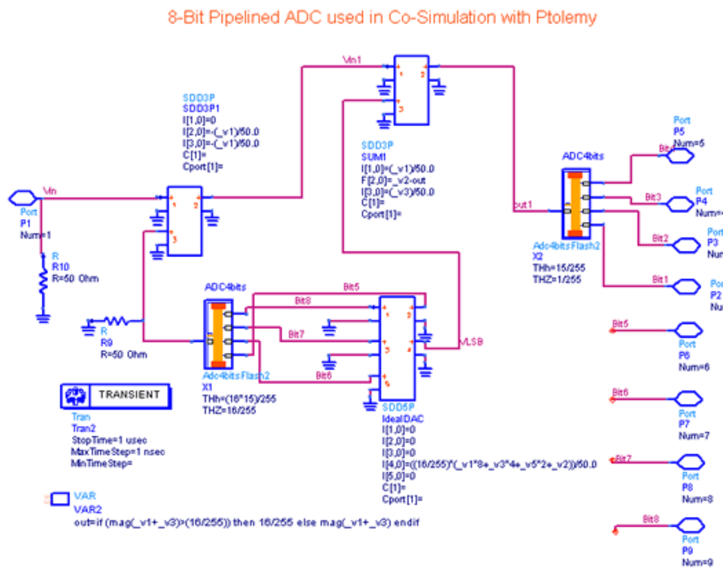

SINAD TEST for an 8-Bit ADC with clock
For sinusoidal input signal, SINAD is defined as the ratio of an rms value of the input sine wave to the rms value of the noise of the converter from DC to "sampling frequency/2", including the first N harmonics of THD (usually the 2nd through 5th order harmonics), and excluding DC.
Measurement with a an input voltage -1db of full scale range

ADC Examples

The following ADC examples are included:

- "ADC_with_clock_Demo" is an example using a clock. It demonstrates a sinusoidal signal input to an 8-bit A-D. The output of the A-D is reconstructed with an ideal Ptolemy D-A and Lowpass filter, and can be observed with a TK plot. The data display also shows the output of the ADC using the TimedSink data collector.
- "ADC_without_clock_Demo" is an example without a clock. It demonstrates a sinusoidal signal input to an 8-bit A-D. The output of the A-D is branched into two paths. The first path goes through an ideal Ptolemy D-A to reconstruct the digitized signal back to analog and observe it with TK plot.The second path is used to analyze the output bits. The data display shows each bit and their combinations that make up numerical "words".
- "Test_ADC_4bit_Flash" is an example of a 4-bit ADC.
- "Test_ADC_pipeline" is an example of an 8-bit pipelined MOS ADC.
- "Test_8Bit_Pipelined_ADC_DNL" is an example of a DNL test for the 8-bit pipelined MOS ADC.
- "Test_DNL_in_Edge" is an application example using EDGE modulation.

Figure 3: 8-Bit Pipelined MOS ADC.

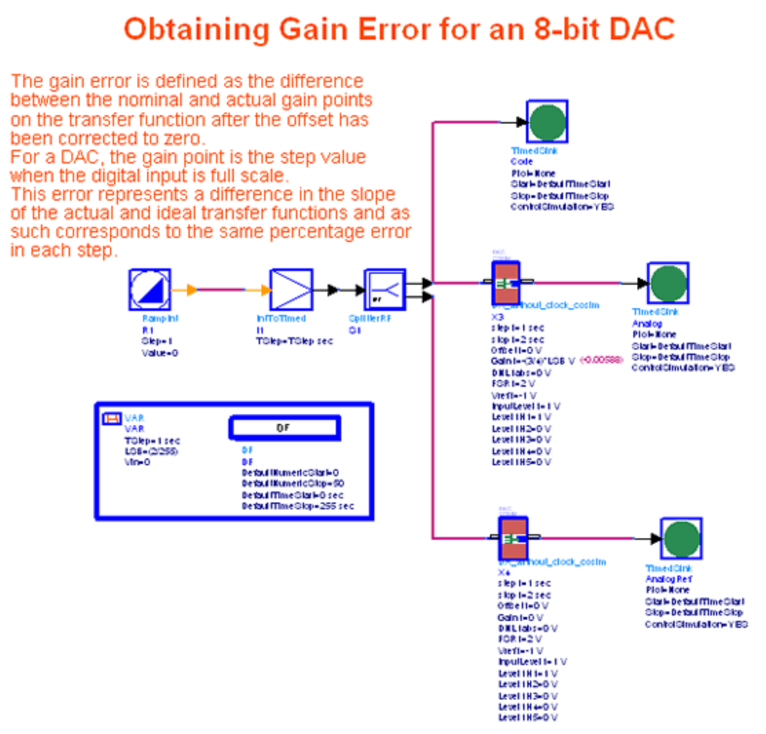

8-Bit Pipelined ADC used in Co-Simulation with Ptolemy
TRANSIENT
out=if (mag(_v1+_v3)>(16/255)) then 16/255 else mag(_v1+_v3) endif

DAC Tests

In these series of test templates, DACs can be fully characterized with the following tests.

- "Test_DAC_without_clock_DNL" tests Differential Non Linearity (DNL)
- "Test_DAC_without_clock_SNR" tests Signal to Noise Ratio (SNR)
- "Test_DAC_without_clock_SINAD" tests Signal Noise and Distortion(SINAD)
- "Test_DAC_without_clock_OffsetError" tests Offset Error
- "Test_DAC_without_clock_GainError" tests Gain Error
- "Test_DAC_without_clock_THD" tests Total Harmonic Distortion (THD)
- "Test_DAC_without_clock_SFDR" tests Spurious-free Dynamic Range(SFDR)

Figure 4: Gain Error for 8-Bit DAC



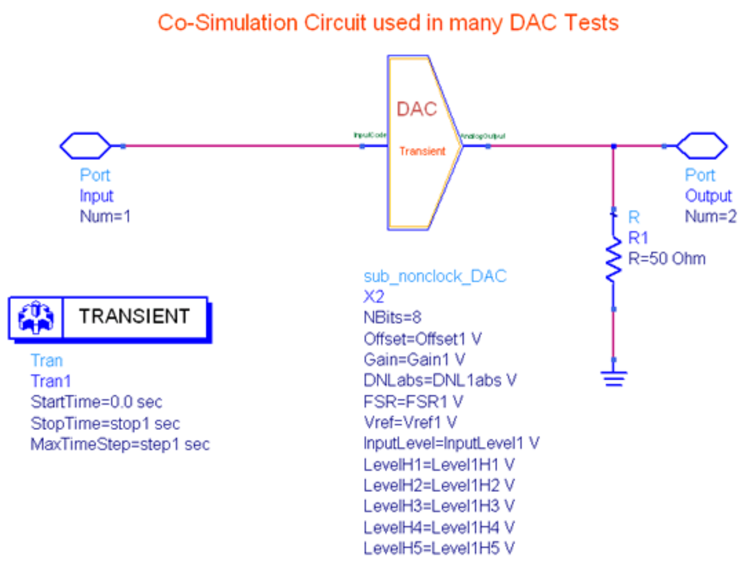
Obtaining Gain Error for an 8-bit DAC

The gain error is defined as the difference between the nominal and actual gain points on the transfer function after the offset has been corrected to zero.
For a DAC, the gain point is the step value when the digital input is full scale.
This error represents a difference in the slope of the actual and ideal transfer functions and as such corresponds to the same percentage error in each step.

DAC Examples

One general purpose DAC example ("DA_without_clock_cosim") is used to obtain the input/output characteristics.

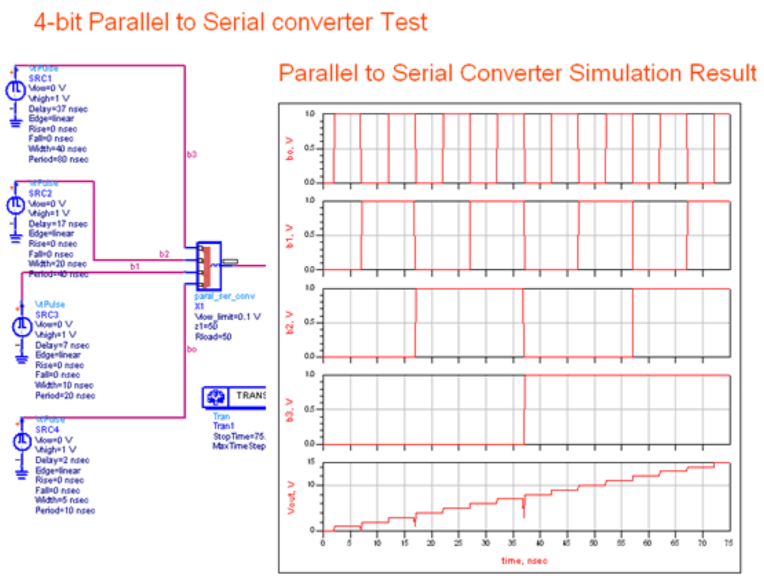
Figure 5: DAC Cosimulation Schematic



Libraries

This guide includes 13 system level logic components that can be used in system level ADC/DAC designs, and 7 MOSFET logic devices for demonstration. These components also have equivalent test circuits and data display setups.

Figure 6: 4-Bit Parallel to Serial Converter



Notes

- "sub_nonclock_ADC" uses TSTEP, the simulation time step, instead of a clockfor sampling. 1/TSTEP is the sampling rate. It also defines the Bandwidth afterthe FFT process.
- "sub_clocked_ADC" uses an external clock for sampling. If the aperture erroris not critical, use the non clock ADC for faster simulation.
  sub_nonclock_ADC" and "sub_clocked_ADC"Parameters
  NBITS Number of Bits (up upped1).
  Offset Offset error of the ADC (in Volts).
  Gain Gain error of the ADC (in Volts).
  DNLabs Absolute differential non linearity error of the ADC (in Volts or LSB range from 0 to 0.7, for example 0.3*LSB). It modifies the output step width to1LSB +/- DNLabs. If DNLabs exceeds 0.7LSB, the model can become nonmonotonic, and can miss codes.
  FSR Full Scale Range of the ADC (in Volts).
  Vref Minimum value of the Analog input for the ADC.
  InputLevel Magnitude of the fundamental of the analog input spectrum. It normalizes the coefficients in the polynomial that characterizes the output vs. the input.
  LevelH1 Magnitude of the fundamental at the output with InputLevel as stimulus (in Volts).
  LevelH2(H3, H4, and H5) Magnitude of the second, third, fourth, and fifth Harmonic at the output with InputLevel as stimulus (in Volts).

Budget Analysis Application Guide

Introduction

This application guide provides convenient access to examples of various budget analysis calculations for an RF system, chain or line-up.

Using the pull down menu, all examples with their corresponding data-displays can be opened and viewed. The components used in these designs can be replaced with user's own circuit/behavioral block for simulation. Some simulation setup and parameter adjustment may be required to support the replacing component.

Here is a list of the examples, and how the application guide is organized:

2-Port Cascaded Networks (Using Budget Controller)
In this series of examples, budget analysis calculations are performed on 2-port 2-pin

cascaded networks. 13 examples are included in this section of the application guide to illustrate budget analysis. These examples are used to obtain Noise, Gain, Power and non-linearities including third-order intercept and P1-dB compression points. There are also examples that illustrate the selection between alternate paths and exporting of results to an excel spreadsheet.

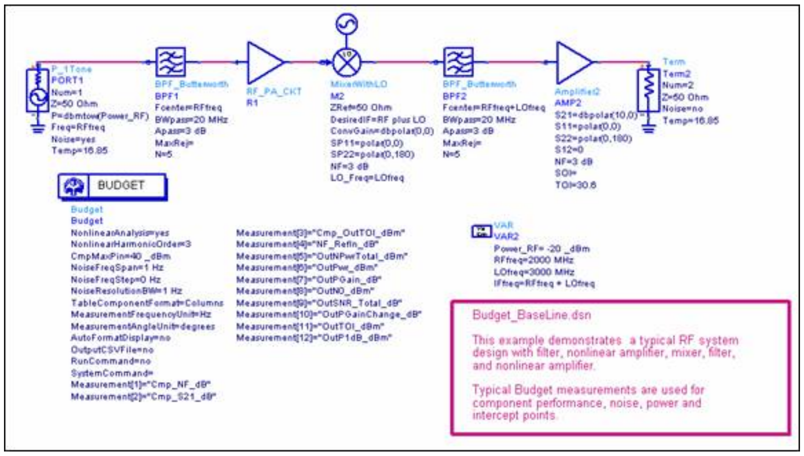
2-Port Cascaded Networks (Using Budget Expressions)
In this area are found two examples focused on gain and power measurements using either the AC analysis controller or the Harmonic balance controller. Although the Budget Controller is recommended for most budget simulations, in some cases the flexibility of options afforded by using AC or HB simulation may be desirable.

Multi-Port Topology Networks (Using Budget Expressions)
In this example, budget analysis calculations are performed on networks having an arbitrary topology. Gain, Power and VSWR measurements are obtained. If the user replaces the components with their own circuits, some changes may be required in the simulation controller to achieve convergence in highly nonlinear cases.

Mixer Spurious Response and Spur Tracking
In this example, the mixer is simulated to see the spurs generated when there is no filtering. The RF frequency is swept and the spectrum at the IF frequency is computed. The MixerIMT2 component is used to model the mixer. The spurious characteristics are provided by a data file in the ".imt" format.

2-Port Cascaded Networks (Using Budget Controller)

"Noise, Power and Intercept Points" demonstrates a typical RF system chain with nonlinear amplifiers, mixer and filters for computing component performance for noise, power and intercept points.



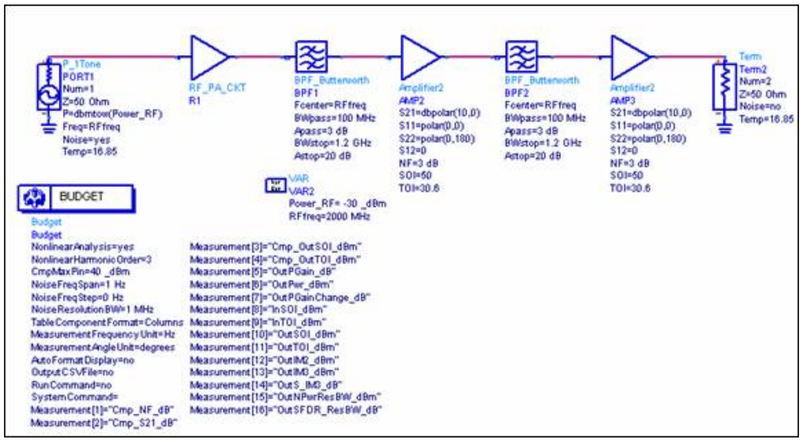
```
P_1Tone
PORT1
Num=1
Z=50 Ohm
P=dbmtow(Power_RF)
Freq=RFfreq
Noise=yes
Temp=16.85

BPF_Butterworth
BPF1
Fcenter=RFfreq
BWpass=20 MHz
Apass=3 dB
MaxRej=
N=5

RF_PA_CKT
R1

MixerWithLO
M2
ZRef=50 Ohm
DesiredIF=RF plus LO
ConvGain=dbpolar(0,0)
SP11=polar(0,0)
SP22=polar(0,180)
NF=3 dB
LO_Freq=LOfreq

BPF_Butterworth
BPF2
Fcenter=RFfreq+LOfreq
BWpass=20 MHz
Apass=3 dB
MaxRej=
N=5

Amplifier2
AMP2
S21=dbpolar(10,0)
S11=polar(0,0)
S22=polar(0,180)
S12=0
NF=3 dB
SOI=
TOI=30.6

Term
Term2
Num=2
Z=50 Ohm
Noise=no
Temp=16.85

BUDGET
Budget
Budget
NonlinearAnalysis=yes
NonlinearHarmonicOrder=3
CmpMaxPin=40 _dBm
NoiseFreqSpan=1 Hz
NoiseFreqStep=0 Hz
NoiseResolutionBW=1 Hz
TableComponentFormat=Columns
MeasurementFrequencyUnit=Hz
MeasurementAngleUnit=degrees
AutoFormatDisplay=no
OutputCSVFile=no
RunCommand=no
SystemCommand=
Measurement[1]="Cmp_NF_dB"
Measurement[2]="Cmp_S21_dB"
Measurement[3]="Cmp_OutTOI_dBm"
Measurement[4]="NF_RefIn_dB"
Measurement[5]="OutNPwrTotal_dBm"
Measurement[6]="OutPwr_dBm"
Measurement[7]="OutPGain_dB"
Measurement[8]="OutNO_dBm"
Measurement[9]="OutSNR_Total_dB"
Measurement[10]="OutPGainChange_dB"
Measurement[11]="OutTOI_dBm"
Measurement[12]="OutP1dB_dBm"

VAR
VAR2
Power_RF= -20 _dBm
RFfreq=2000 MHz
LOfreq=3000 MHz
IFfreq=RFfreq + LOfreq
```

Budget_BaseLine.dsn

This example demonstrates a typical RF system design with filter, nonlinear amplifier, mixer, filter, and nonlinear amplifier.

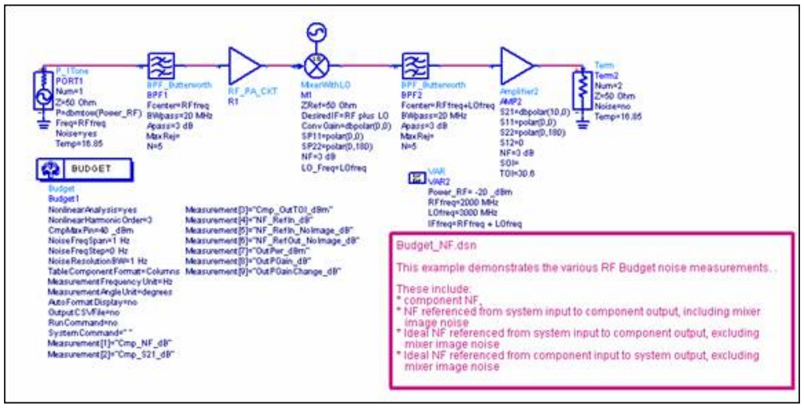
Typical Budget measurements are used for component performance, noise, power and intercept points.

"SOI, TOI points, IM levels and SFDR" demonstrates the various RF budget SOI and TOI measurements.



```
P_1Tone
PORT1
Num=1
Z=50 Ohm
P=dbmtow(Power_RF)
Freq=RFfreq
Noise=yes
Temp=16.85

RF_PA_CKT
R1

BPF_Butterworth
BPF1
Fcenter=RFfreq
BWpass=100 MHz
Apass=3 dB
BWstop=1.2 GHz
Astop=20 dB

Amplifier2
AMP2
S21=dbpolar(10,0)
S11=polar(0,0)
S22=polar(0,180)
S12=0
NF=3 dB
SOI=50
TOI=30.6

BPF_Butterworth
BPF2
Fcenter=RFfreq
BWpass=100 MHz
Apass=3 dB
BWstop=1.2 GHz
Astop=20 dB

Amplifier2
AMP3
S21=dbpolar(10,0)
S11=polar(0,0)
S22=polar(0,180)
S12=0
NF=3 dB
SOI=50
TOI=30.6

Term
Term2
Num=2
Z=50 Ohm
Noise=no
Temp=16.85

VAR
VAR2
Power_RF= -30 _dBm
RFfreq=2000 MHz

BUDGET
Budget
Budget
NonlinearAnalysis=yes
NonlinearHarmonicOrder=3
CmpMaxPin=40 _dBm
NoiseFreqSpan=1 Hz
NoiseFreqStep=0 Hz
NoiseResolutionBW=1 MHz
TableComponentFormat=Columns
MeasurementFrequencyUnit=Hz
MeasurementAngleUnit=degrees
AutoFormatDisplay=no
OutputCSVFile=no
RunCommand=no
SystemCommand=
Measurement[1]="Cmp_NF_dB"
Measurement[2]="Cmp_S21_dB"
Measurement[3]="Cmp_OutSOI_dBm"
Measurement[4]="Cmp_OutTOI_dBm"
Measurement[5]="OutPGain_dB"
Measurement[6]="OutPwr_dBm"
Measurement[7]="OutPGainChange_dB"
Measurement[8]="InSOI_dBm"
Measurement[9]="InTOI_dBm"
Measurement[10]="OutSOI_dBm"
Measurement[11]="OutTOI_dBm"
Measurement[12]="OutIM2_dBm"
Measurement[13]="OutIM3_dBm"
Measurement[14]="OutS_IM3_dB"
Measurement[15]="OutNPwrResBW_dBm"
Measurement[16]="OutSFDR_ResBW_dB"
```

"Budget Noise Figure Measurements" demonstrates the various RF budget noise measurements.

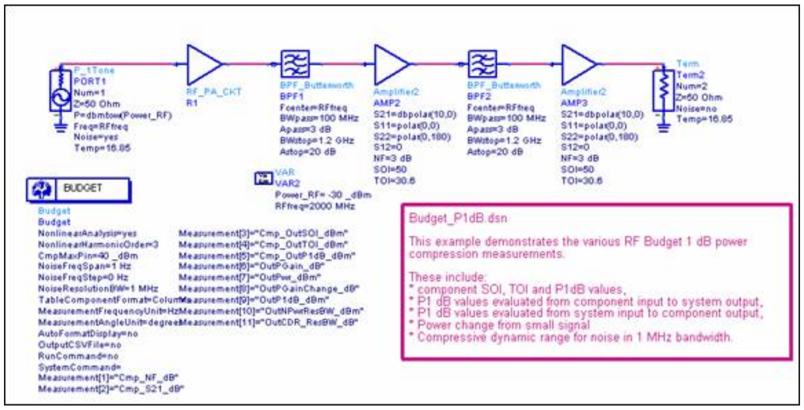


Budget_NF.dsn

This example demonstrates the various RF Budget noise measurements. .

These include:
* component NF,
* NF referenced from system input to component output, including mixer image noise
* Ideal NF referenced from system input to component output, excluding mixer image noise
* Ideal NF referenced from component input to system output, excluding mixer image noise

"1-dB Power Compression Measurements" demonstrates the RF budget P1-dB compression measurements.

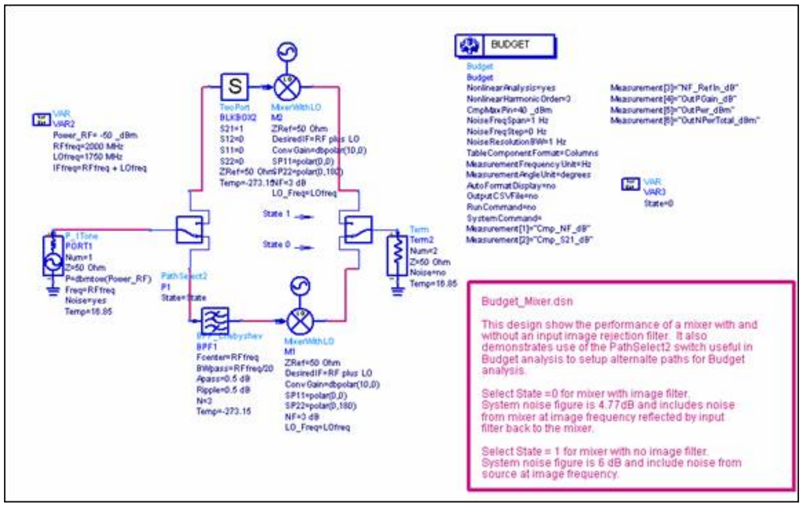


Budget_P1dB.dsn

This example demonstrates the various RF Budget 1 dB power compression measurements.

These include:
* component SOI, TOI and P1dB values,
* P1 dB values evaluated from component input to system output,
* P1 dB values evaluated from system input to component output,
* Power change from small signal
* Compressive dynamic range for noise in 1 MHz bandwidth.

"Mixer Performance with Alternate Paths" demonstrates the mixer performance and the use of "pathselect2" component to setup alternate paths.



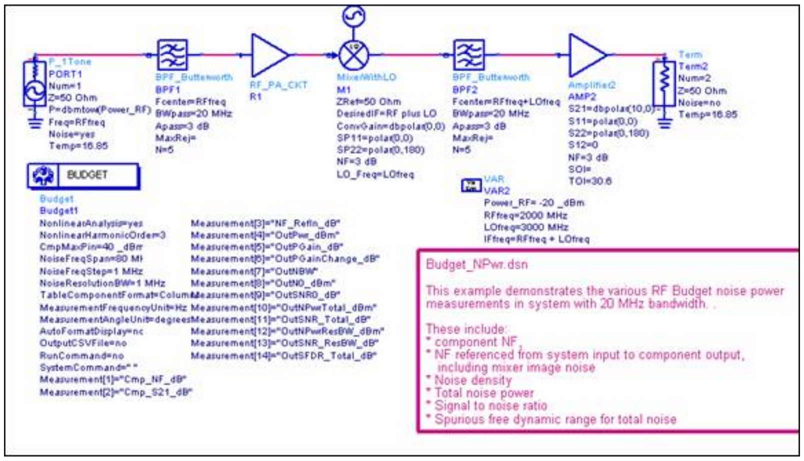
Budget_Mixer.dsn

This design show the performance of a mixer with and without an input image rejection filter. It also demonstrates use of the PathSelect2 switch useful in Budget analysis to setup alternalte paths for Budget analysis.

Select State =0 for mixer with image filter.
System noise figure is 4.77dB and includes noise from mixer at image frequency reflected by input filter back to the mixer.

Select State = 1 for mixer with no image filter.
System noise figure is 6 dB and include noise from source at image frequency.

"Noise Power Measurements" demonstrates the RF budget noise measurements in a system with 20MHz bandwidth.



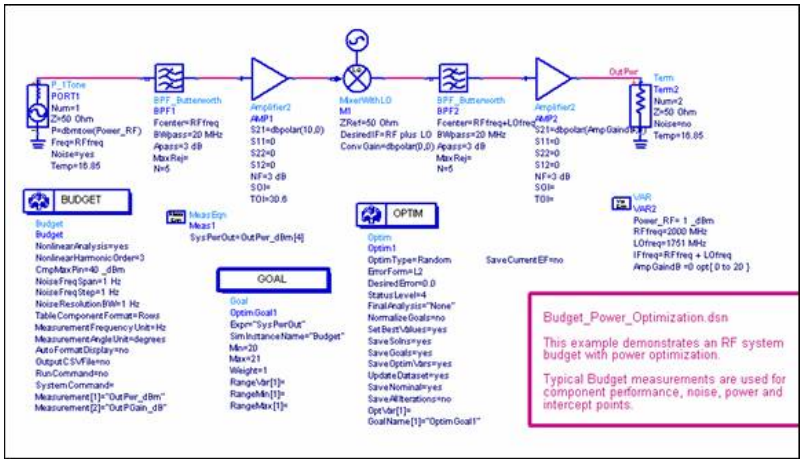
```
P_1Tone
PORT1
Num=1
Z=50 Ohm
P=dbmtow(Power_RF)
Freq=RFfreq
Noise=yes
Temp=16.85

BPF_Butterworth
BPF1
Fcenter=RFfreq
BWpass=20 MHz
Apass=3 dB
MaxRej=
N=5

RF_PA_CKT
R1

MixerWithLO
M1
ZRef=50 Ohm
DesiredIF=RF plus LO
ConvGain=dbpolar(0,0)
SP11=polar(0,0)
SP22=polar(0,180)
NF=3 dB
LO_Freq=LOfreq

BPF_Butterworth
BPF2
Fcenter=RFfreq+LOfreq
BWpass=20 MHz
Apass=3 dB
MaxRej=
N=5

Amplifier2
AMP2
S21=dbpolar(10,0)
S11=polar(0,0)
S22=polar(0,180)
S12=0
NF=3 dB
SOI=
TOI=30.6

Term
Term2
Num=2
Z=50 Ohm
Noise=no
Temp=16.85

BUDGET
Budget
Budget1
NonlinearAnalysis=yes
NonlinearHarmonicOrder=3
CmpMaxPin=40 _dBm
NoiseFreqSpan=80 MH
NoiseFreqStep=1 MHz
NoiseResolutionBW=1 MHz
TableComponentFormat=Colum
MeasurementFrequencyUnit=Hz
MeasurementAngleUnit=degrees
AutoFormatDisplay=nc
OutputCSVFile=no
RunCommand=no
SystemCommand=" "
Measurement[1]="Cmp_NF_dB"
Measurement[2]="Cmp_S21_dB"
Measurement[3]="NF_RefIn_dB"
Measurement[4]="OutPwr_dBm"
Measurement[5]="OutPGain_dB"
Measurement[6]="OutPGainChange_dB"
Measurement[7]="OutNBW"
Measurement[8]="OutN0_dBm"
Measurement[9]="OutSNR0_dB"
Measurement[10]="OutNPwrTotal_dBm"
Measurement[11]="OutSNR_Total_dB"
Measurement[12]="OutNPwrResBW_dBm"
Measurement[13]="OutSNR_ResBW_dB"
Measurement[14]="OutSFDR_Total_dB"

VAR
VAR2
Power_RF= -20 _dBm
RFfreq=2000 MHz
LOfreq=3000 MHz
IFfreq=RFfreq + LOfreq
```

Budget_NPwr.dsn

This example demonstrates the various RF Budget noise power measurements in system with 20 MHz bandwidth. .

These include:
* component NF,
* NF referenced from system input to component output,
  including mixer image noise
* Noise density
* Total noise power
* Signal to noise ratio
* Spurious free dynamic range for total noise

"Noise, Power and intercept Points with Power Optimization" demonstrates an RF system budget with power optimization.



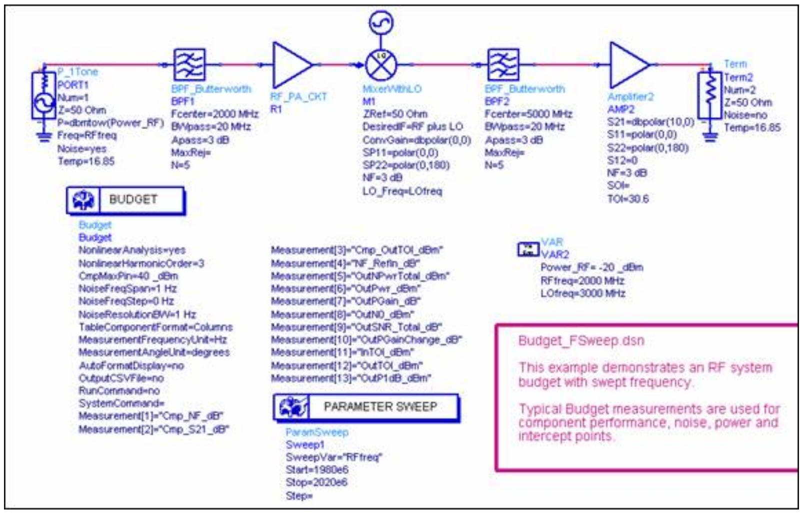
```
P_1Tone
PORT1
Num=1
Z=50 Ohm
P=dbmtow(Power_RF)
Freq=RFfreq
Noise=yes
Temp=16.85

BPF_Butterworth
BPF1
Fcenter=RFfreq
BWpass=20 MHz
Apass=3 dB
MaxRej=
N=5

Amplifier2
AMP1
S21=dbpolar(10,0)
S11=0
S22=0
S12=0
NF=3 dB
SOI=
TOI=30.6

MixerWithLO
M1
ZRef=50 Ohm
DesiredIF=RF plus LO
ConvGain=dbpolar(0,0)

BPF_Butterworth
BPF2
Fcenter=RFfreq+LOfreq
BWpass=20 MHz
Apass=3 dB
MaxRej=
N=5

Amplifier2
AMP2
S21=dbpolar(AmpGaindB,0)
S11=0
S22=0
S12=0
NF=3 dB
SOI=
TOI=

OutPwr

Term
Term2
Num=2
Z=50 Ohm
Noise=no
Temp=16.85

BUDGET
Budget
Budget
NonlinearAnalysis=yes
NonlinearHarmonicOrder=3
CmpMaxPin=40 _dBm
NoiseFreqSpan=1 Hz
NoiseFreqStep=1 Hz
NoiseResolutionBW=1 Hz
TableComponentFormat=Rows
MeasurementFrequencyUnit=Hz
MeasurementAngleUnit=degrees
AutoFormatDisplay=no
OutputCSVFile=no
RunCommand=no
SystemCommand=
Measurement[1]="OutPwr_dBm"
Measurement[2]="OutPGain_dB"

MeasEqn
Meas1
SysPwrOut=OutPwr_dBm[4]

GOAL
Goal
OptimGoal1
Expr="SysPwrOut"
SimInstanceName="Budget"
Min=20
Max=21
Weight=1
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=

OPTIM
Optim
Optim1
OptimType=Random
ErrorForm=L2
DesiredError=0.0
StatusLevel=4
FinalAnalysis="None"
NormalizeGoals=no
SetBestValues=yes
SaveSolns=yes
SaveGoals=yes
SaveOptimVars=yes
UpdateDataset=yes
SaveNominal=yes
SaveAllIterations=no
OptVar[1]=
GoalName[1]="OptimGoal1"
SaveCurrentEF=no

VAR
VAR2
Power_RF= 1 _dBm
RFfreq=2000 MHz
LOfreq=1751 MHz
IFfreq=RFfreq + LOfreq
AmpGaindB =0 opt[ 0 to 20 ]
```

Budget_Power_Optimization.dsn

This example demonstrates an RF system budget with power optimization.

Typical Budget measurements are used for component performance, noise, power and intercept points.

"Noise, Power and Intercept Point with Swept Frequency" demonstrates an RF system budget with frequency sweep.



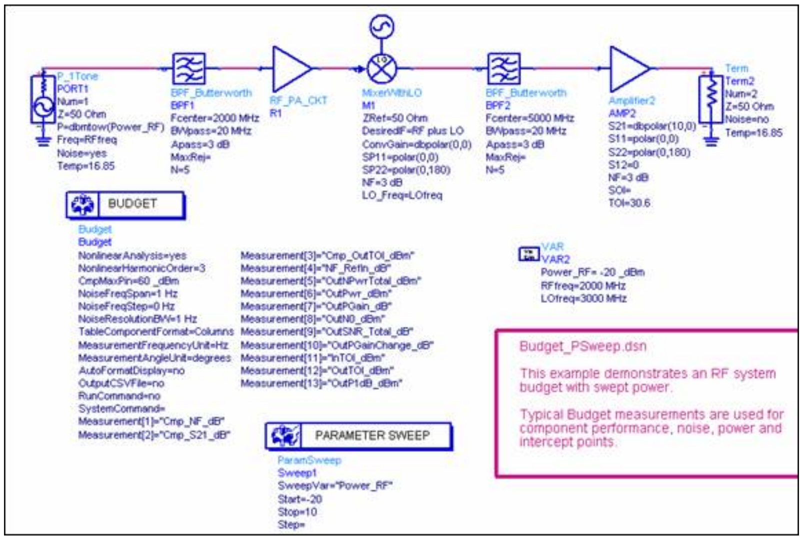
```
P_1Tone
PORT1
Num=1
Z=50 Ohm
P=dbmtow(Power_RF)
Freq=RFfreq
Noise=yes
Temp=16.85

BPF_Butterworth
BPF1
Fcenter=2000 MHz
BWpass=20 MHz
Apass=3 dB
MaxRej=
N=5

RF_PA_CKT
R1

MixerWithLO
M1
ZRef=50 Ohm
DesiredIF=RF plus LO
ConvGain=dbpolar(0,0)
SP11=polar(0,0)
SP22=polar(0,180)
NF=3 dB
LO_Freq=LOfreq

BPF_Butterworth
BPF2
Fcenter=5000 MHz
BWpass=20 MHz
Apass=3 dB
MaxRej=
N=5

Amplifier2
AMP2
S21=dbpolar(10,0)
S11=polar(0,0)
S22=polar(0,180)
S12=0
NF=3 dB
SOI=
TOI=30.6

Term
Term2
Num=2
Z=50 Ohm
Noise=no
Temp=16.85

BUDGET
Budget
Budget
NonlinearAnalysis=yes
NonlinearHarmonicOrder=3
CmpMaxPin=40 _dBm
NoiseFreqSpan=1 Hz
NoiseFreqStep=0 Hz
NoiseResolutionBW=1 Hz
TableComponentFormat=Columns
MeasurementFrequencyUnit=Hz
MeasurementAngleUnit=degrees
AutoFormatDisplay=no
OutputCSVFile=no
RunCommand=no
SystemCommand=
Measurement[1]="Cmp_NF_dB"
Measurement[2]="Cmp_S21_dB"
Measurement[3]="Cmp_OutTOI_dBm"
Measurement[4]="NF_RefIn_dB"
Measurement[5]="OutNPwrTotal_dBm"
Measurement[6]="OutPwr_dBm"
Measurement[7]="OutPGain_dB"
Measurement[8]="OutN0_dBm"
Measurement[9]="OutSNR_Total_dB"
Measurement[10]="OutPGainChange_dB"
Measurement[11]="InTOI_dBm"
Measurement[12]="OutTOI_dBm"
Measurement[13]="OutP1dB_dBm"

PARAMETER SWEEP
ParamSweep
Sweep1
SweepVar="RFfreq"
Start=1980e6
Stop=2020e6
Step=

VAR
VAR2
Power_RF= -20 _dBm
RFfreq=2000 MHz
LOfreq=3000 MHz
```

Budget_FSweep.dsn

This example demonstrates an RF system budget with swept frequency.

Typical Budget measurements are used for component performance, noise, power and intercept points.

"Noise, Power and Intercept Point with Swept Power" demonstrates an RF system budget with power sweep.



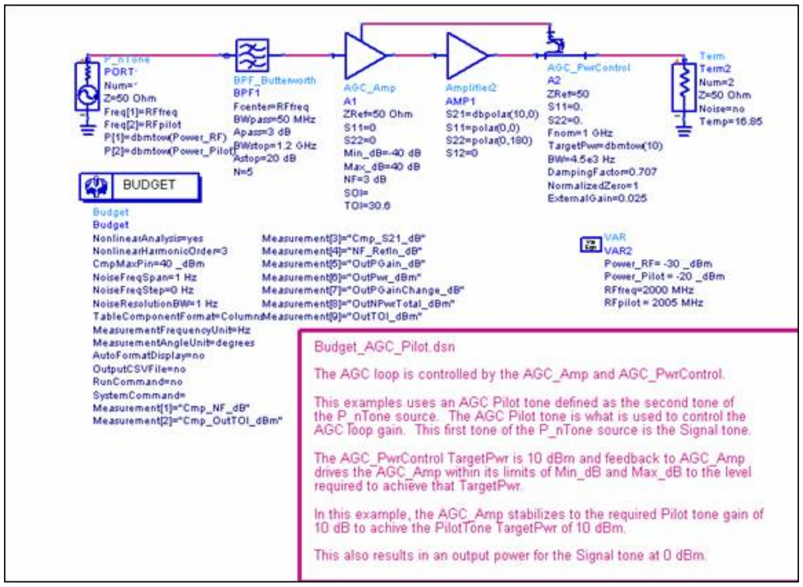
```
P_1Tone
PORT1
Num=1
Z=50 Ohm
P=dbmtow(Power_RF)
Freq=RFfreq
Noise=yes
Temp=16.85

BPF_Butterworth
BPF1
Fcenter=2000 MHz
BWpass=20 MHz
Apass=3 dB
MaxRej=
N=5

RF_PA_CKT
R1

MixerWithLO
M1
ZRef=50 Ohm
DesiredIF=RF plus LO
ConvGain=dbpolar(0,0)
SP11=polar(0,0)
SP22=polar(0,180)
NF=3 dB
LO_Freq=LOfreq

BPF_Butterworth
BPF2
Fcenter=5000 MHz
BWpass=20 MHz
Apass=3 dB
MaxRej=
N=5

Amplifier2
AMP2
S21=dbpolar(10,0)
S11=polar(0,0)
S22=polar(0,180)
S12=0
NF=3 dB
SOI=
TOI=30.6

Term
Term2
Num=2
Z=50 Ohm
Noise=no
Temp=16.85

BUDGET
Budget
Budget
NonlinearAnalysis=yes
NonlinearHarmonicOrder=3
CmpMaxPin=60 _dBm
NoiseFreqSpan=1 Hz
NoiseFreqStep=0 Hz
NoiseResolutionBW=1 Hz
TableComponentFormat=Columns
MeasurementFrequencyUnit=Hz
MeasurementAngleUnit=degrees
AutoFormatDisplay=no
OutputCSVFile=no
RunCommand=no
SystemCommand=
Measurement[1]="Cmp_NF_dB"
Measurement[2]="Cmp_S21_dB"
Measurement[3]="Cmp_OutTOI_dBm"
Measurement[4]="NF_RefIn_dB"
Measurement[5]="OutNPwrTotal_dBm"
Measurement[6]="OutPwr_dBm"
Measurement[7]="OutPGain_dB"
Measurement[8]="OutN0_dBm"
Measurement[9]="OutSNR_Total_dB"
Measurement[10]="OutPGainChange_dB"
Measurement[11]="InTOI_dBm"
Measurement[12]="OutTOI_dBm"
Measurement[13]="OutP1dB_dBm"

VAR
VAR2
Power_RF= -20 _dBm
RFfreq=2000 MHz
LOfreq=3000 MHz

PARAMETER SWEEP
ParamSweep
Sweep1
SweepVar="Power_RF"
Start=-20
Stop=10
Step=
```

Budget_PSweep.dsn

This example demonstrates an RF system budget with swept power.

Typical Budget measurements are used for component performance, noise, power and intercept points.

"Power, Noise and TOI for AGC loops with/without Pilot Tone" demonstrates 2 designs where budget measurements are computed for AGC loops, one without a pilot tone and the other with a pilot tone.



```
P_nTone
PORT1
Num=1
Z=50 Ohm
Freq[1]=RFfreq
Freq[2]=RFpilot
P[1]=dbmtow(Power_RF)
P[2]=dbmtow(Power_Pilot)

BPF_Butterworth
BPF1
Fcenter=RFfreq
BWpass=50 MHz
Apass=3 dB
BWstop=1.2 GHz
Astop=20 dB
N=5

AGC_Amp
A1
ZRef=50 Ohm
S11=0
S22=0
Min_dB=-40 dB
Max_dB=40 dB
NF=3 dB
SOI=
TOI=30.6

Amplifier2
AMP1
S21=dbpolar(10,0)
S11=polar(0,0)
S22=polar(0,180)
S12=0

AGC_PwrControl
A2
ZRef=50
S11=0
S22=0
Fnom=1 GHz
TargetPwr=dbmtow(10)
BW=4.5e3 Hz
DampingFactor=0.707
NormalizedZero=1
ExternalGain=0.025

Term
Term2
Num=2
Z=50 Ohm
Noise=no
Temp=16.85

BUDGET
Budget
Budget
NonlinearAnalysis=yes
NonlinearHarmonicOrder=3
CmpMaxPin=40 _dBm
NoiseFreqSpan=1 Hz
NoiseFreqStep=0 Hz
NoiseResolutionBW=1 Hz
TableComponentFormat=Columns
MeasurementFrequencyUnit=Hz
MeasurementAngleUnit=degrees
AutoFormatDisplay=no
OutputCSVFile=no
RunCommand=no
SystemCommand=
Measurement[1]="Cmp_NF_dB"
Measurement[2]="Cmp_OutTOI_dBm"
Measurement[3]="Cmp_S21_dB"
Measurement[4]="NF_RefIn_dB"
Measurement[5]="OutPGain_dB"
Measurement[6]="OutPwr_dBm"
Measurement[7]="OutPGainChange_dB"
Measurement[8]="OutNPwrTotal_dBm"
Measurement[9]="OutTOI_dBm"

VAR
VAR2
Power_RF= -30 _dBm
Power_Pilot = -20 _dBm
RFfreq=2000 MHz
RFpilot = 2005 MHz
```

Budget_AGC_Pilot.dsn

The AGC loop is controlled by the AGC_Amp and AGC_PwrControl.

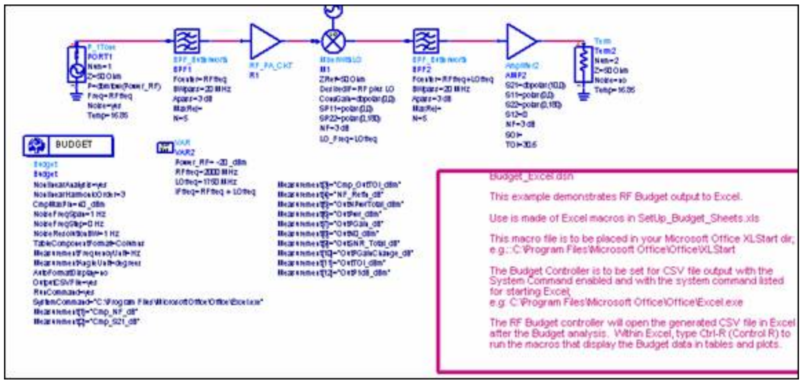
This examples uses an AGC Pilot tone defined as the second tone of the P_nTone source. The AGC Pilot tone is what is used to control the AGC Loop gain. This first tone of the P_nTone source is the Signal tone.

The AGC_PwrControl TargetPwr is 10 dBm and feedback to AGC_Amp drives the AGC_Amp within its limits of Min_dB and Max_dB to the level required to achieve that TargetPwr.

In this example, the AGC_Amp stabilizes to the required Pilot tone gain of 10 dB to achive the PilotTone TargetPwr of 10 dBm.

This also results in an output power for the Signal tone at 0 dBm.

"Exporting RF Budget Results to Excel" demonstrates how RF budget results can be exported to a text file, in the Comma Separated Values (CSV) format. ADS includes an example of a user-defined Excel spreadsheet. It contains a macro that you can use to post-process an exported CSV file. This Excel macro will process the CSV file into formatted tables and plots for each measurement. The spreadsheet is named "SetUp_Budget_Sheets.xls" and it is located in $HPEESOF_DIR/examples/Tutorial/RF_Budget_Examples_wrk.



```
VAR
VAR2
Power_RF= -20 _dBm
RFfreq=2000 MHz
LOfreq=1150 MHz
IFfreq=RFfreq + LOfreq

BUDGET
Budget
Budget
NonlinearAnalysis=yes
NonlinearHarmonicOrder=3
OutputCSVFile=yes
RunCommand=yes
SystemCommand="C:\Program Files\Microsoft Office\Office\Excel.exe"
Measurement[1]="Cmp_NF_dB"
Measurement[2]="Cmp_S21_dB"
Measurement[3]="Cmp_OutTOI_dBm"
Measurement[4]="NF_RefIn_dB"
Measurement[5]="OutNPwrTotal_dBm"
Measurement[6]="OutPwr_dBm"
Measurement[7]="OutPGain_dB"
Measurement[8]="OutN0_dBm"
Measurement[9]="OutSNR_Total_dB"
Measurement[10]="OutPGainChange_dB"
Measurement[11]="OutTOI_dBm"
Measurement[12]="OutP1dB_dBm"
```

Budget_Excel.dsn

This example demonstrates RF Budget output to Excel.

Use is made of Excel macros in SetUp_Budget_Sheets.xls

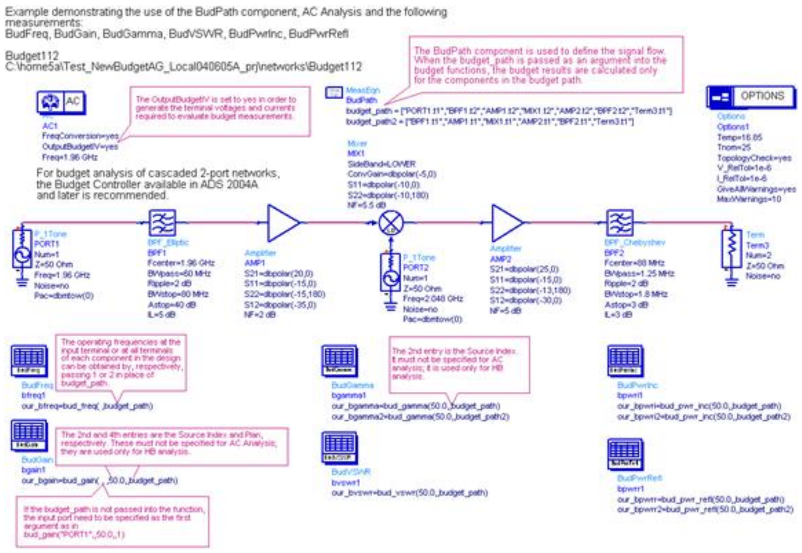
This macro file is to be placed in your Microsoft Office XLStart dir, e.g.: C:\Program Files\Microsoft Office\Office\XLStart

The Budget Controller is to be set for CSV file output with the System Command enabled and with the system command listed for starting Excel
e.g. C:\Program Files\Microsoft Office\Office\Excel.exe

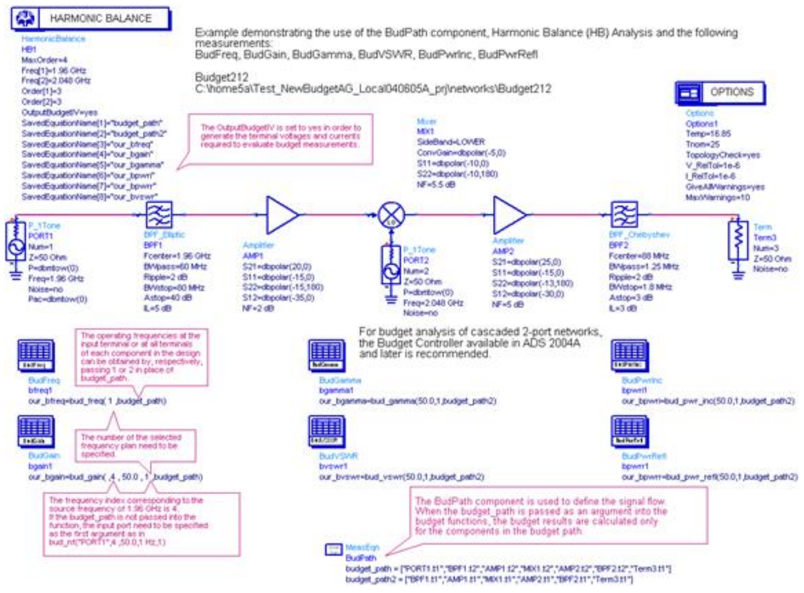
The RF Budget controller will open the generated CSV file in Excel after the Budget analysis. Within Excel, type Ctrl-R (Control R) to run the macros that display the Budget data in tables and plots.

2-Port Cascaded Networks (Using Budget Expressions)

"Gain and Incident Power Measurements (AC Controller)" demonstrates the budget gain and incident power calculation for a linear cascaded network. The BudPath component is used to define the signal flow.

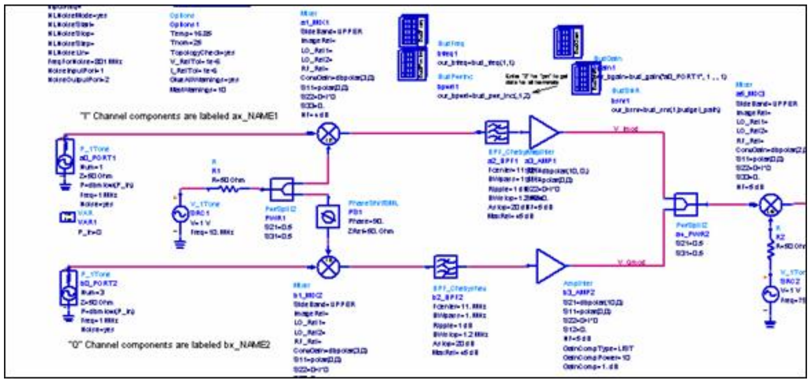

Example demonstrating the use of the BudPath component, AC Analysis and the following measurements:
BudFreq, BudGain, BudGamma, BudVSWR, BudPwrInc, BudPwrRefl
Budget112
C:\home5a\Test_NewBudgetAG_Local040605A_prj\networks\Budget112
AC
AC1
FreqConversion=yes
OutputBudgetIV=yes
Freq=1.96 GHz
The OutputBudgetIV is set to yes in order to generate the terminal voltages and currents required to evaluate budget measurements.
For budget analysis of cascaded 2-port networks, the Budget Controller available in ADS 2004A and later is recommended.
The BudPath component is used to define the signal flow. When the budget_path is passed as an argument into the budget functions, the budget results are calculated only for the components in the budget path.
MeasEqn
BudPath
budget_path = ["PORT1.t1","BPF1.t2","AMP1.t2","MIX1.t2","AMP2.t2","BPF2.t2","Term3.t1"]
budget_path2 = ["BPF1.t1","AMP1.t1","MIX1.t1","AMP2.t1","BPF2.t1","Term3.t1"]
OPTIONS
Options
Options1
Temp=16.85
Tnom=25
TopologyCheck=yes
V_RelTol=1e-6
I_RelTol=1e-6
GiveAllWarnings=yes
MaxWarnings=10
Mixer
MIX1
SideBand=LOWER
ConvGain=dbpolar(-5,0)
S11=dbpolar(-10,0)
S22=dbpolar(-10,180)
NF=5.5 dB
P_1Tone
PORT1
Num=1
Z=50 Ohm
Freq=1.96 GHz
Noise=no
Pac=dbmtow(0)
BPF_Elliptic
BPF1
Fcenter=1.96 GHz
BWpass=60 MHz
Ripple=2 dB
BWstop=80 MHz
Astop=40 dB
IL=5 dB
Amplifier
AMP1
S21=dbpolar(20,0)
S11=dbpolar(-15,0)
S22=dbpolar(-15,180)
S12=dbpolar(-35,0)
NF=2 dB
P_1Tone
PORT2
Num=1
Z=50 Ohm
Freq=2.048 GHz
Noise=no
Pac=dbmtow(0)
Amplifier
AMP2
S21=dbpolar(25,0)
S11=dbpolar(-15,0)
S22=dbpolar(-13,180)
S12=dbpolar(-30,0)
NF=5 dB
BPF_Chebyshev
BPF2
Fcenter=88 MHz
BWpass=1.25 MHz
Ripple=2 dB
BWstop=1.8 MHz
Astop=3 dB
IL=3 dB
Term
Term3
Num=2
Z=50 Ohm
Noise=no
The operating frequencies at the input terminal or at all terminals of each component in the design can be obtained by, respectively, passing 1 or 2 in place of budget_path.
BudFreq
bfreq1
our_bfreq=bud_freq(,budget_path)
The 2nd entry is the Source Index. It must not be specified for AC analysis; it is used only for HB analysis.
BudGamma
bgamma1
our_bgamma=bud_gamma(50.0,,budget_path)
our_bgamma2=bud_gamma(50.0,,budget_path2)
BudPwrInc
bpwri1
our_bpwri=bud_pwr_inc(50.0,,budget_path)
our_bpwri2=bud_pwr_inc(50.0,,budget_path2)
The 2nd and 4th entries are the Source Index and Plan, respectively. These must not be specified for AC Analysis; they are used only for HB analysis.
BudGain
bgain1
our_bgain=bud_gain(, 50.0,,budget_path)
If the budget_path is not passed into the function, the input port need to be specified as the first argument as in bud_gain("PORT1",,50.0,,1)
BudVSWR
bvswr1
our_bvswr=bud_vswr(50.0,,budget_path)
BudPwrRefl
bpwrr1
our_bpwrr=bud_pwr_refl(50.0,,budget_path)
our_bpwrr2=bud_pwr_refl(50.0,,budget_path2)

"Gain, Incident and Reflected Power, VSWR Measurements (HB Controller)" demonstrates various power measurements using the Harmonic Balance controller and measurement expressions. The BudPath component is used to define the signal flow.


HARMONIC BALANCE
HarmonicBalance
HB1
MaxOrder=4
Freq[1]=1.96 GHz
Freq[2]=2.048 GHz
Order[1]=3
Order[2]=3
OutputBudgetIV=yes
SavedEquationName[1]="budget_path"
SavedEquationName[2]="budget_path2"
SavedEquationName[3]="our_bfreq"
SavedEquationName[4]="our_bgain"
SavedEquationName[5]="our_bgamma"
SavedEquationName[6]="our_bpwri"
SavedEquationName[7]="our_bpwrr"
SavedEquationName[8]="our_bvswr"
Example demonstrating the use of the BudPath component, Harmonic Balance (HB) Analysis and the following measurements:
BudFreq, BudGain, BudGamma, BudVSWR, BudPwrInc, BudPwrRefl
Budget212
C:\home5a\Test_NewBudgetAG_Local040605A_prj\networks\Budget212
The OutputBudgetIV is set to yes in order to generate the terminal voltages and currents required to evaluate budget measurements.
Mixer
MIX1
SideBand=LOWER
ConvGain=dbpolar(-5,0)
S11=dbpolar(-10,0)
S22=dbpolar(-10,180)
NF=5.5 dB
OPTIONS
Options
Options1
Temp=16.85
Tnom=25
TopologyCheck=yes
V_RelTol=1e-6
I_RelTol=1e-6
GiveAllWarnings=yes
MaxWarnings=10
P_1Tone
PORT1
Num=1
Z=50 Ohm
P=dbmtow(0)
Freq=1.96 GHz
Noise=no
Pac=dbmtow(0)
BPF_Elliptic
BPF1
Fcenter=1.96 GHz
BWpass=60 MHz
Ripple=2 dB
BWstop=80 MHz
Astop=40 dB
IL=5 dB
Amplifier
AMP1
S21=dbpolar(20,0)
S11=dbpolar(-15,0)
S22=dbpolar(-15,180)
S12=dbpolar(-35,0)
NF=2 dB
P_1Tone
PORT2
Num=2
Z=50 Ohm
P=dbmtow(0)
Freq=2.048 GHz
Noise=no
Amplifier
AMP2
S21=dbpolar(25,0)
S11=dbpolar(-15,0)
S22=dbpolar(-13,180)
S12=dbpolar(-30,0)
NF=5 dB
BPF_Chebyshev
BPF2
Fcenter=88 MHz
BWpass=1.25 MHz
Ripple=2 dB
BWstop=1.8 MHz
Astop=3 dB
IL=3 dB
Term
Term3
Num=3
Z=50 Ohm
Noise=no
The operating frequencies at the input terminal or at all terminals of each component in the design can be obtained by, respectively, passing 1 or 2 in place of budget_path.
BudFreq
bfreq1
our_bfreq=bud_freq(1 ,budget_path)
For budget analysis of cascaded 2-port networks, the Budget Controller available in ADS 2004A and later is recommended.
BudGamma
bgamma1
our_bgamma=bud_gamma(50.0,1,budget_path2)
BudPwrInc
bpwri1
our_bpwri=bud_pwr_inc(50.0,1,budget_path2)
The number of the selected frequency plan need to be specified.
BudGain
bgain1
our_bgain=bud_gain(4 , 50.0 , 1 ,budget_path)
The frequency index corresponding to the source frequency of 1.96 GHz is 4. If the budget_path is not passed into the function, the input port need to be specified as the first argument as in bud_nf("PORT1",4 ,50.0,1 Hz,1)
BudVSWR
bvswr1
our_bvswr=bud_vswr(50.0,1,budget_path2)
BudPwrRefl
bpwrr1
our_bpwrr=bud_pwr_refl(50.0,1,budget_path2)
The BudPath component is used to define the signal flow. When the budget_path is passed as an argument into the budget functions, the budget results are calculated only for the components in the budget path.
MeasEqn
BudPath
budget_path = ["PORT1.t1","BPF1.t2","AMP1.t2","MIX1.t2","AMP2.t2","BPF2.t2","Term3.t1"]
budget_path2 = ["BPF1.t1","AMP1.t1","MIX1.t1","AMP2.t1","BPF2.t1","Term3.t1"]

Multi-Port Topology Networks (Using Budget Expressions)

"Gain, SNR and Incident Power Measurements" demonstrates RF system budget for multi-port, multi-channel network. To better understand the results, note that there are two non-linear amplifiers in the design. The incident power is different at the two input terminals of the combiner. This is due to the fact that one of the amplifiers looking into the combiner is linear and the other one is non-linear. By changing the "GainCompressionPower" in the non-linear amplifier to around 100 dB (where the amplifier becomes linear) and setting the ripple voltage in the filters to a minimum value of 0.01 dB, you can notice the incident power and gain values match at both the input terminals of the combiner. This shows the advantage of having multiple channels and analyzing the non-linearities in ADS.

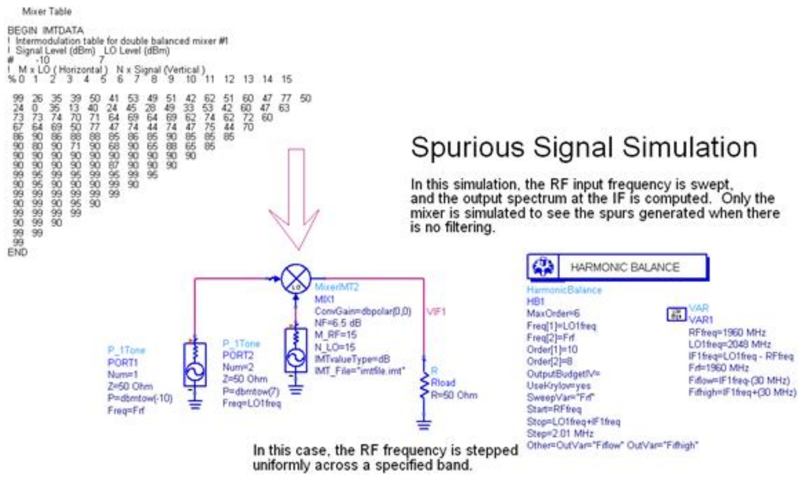


"I" Channel components are labeled ax_NAME1

"Q" Channel components are labeled bx_NAME2

Mixer Spurious Response and Spur Tracking

"Mixer Spurs" demonstrates the spurs generated in a mixer when there is no filtering.



Mixer Table

```
BEGIN  IMTDATA
!  Intermodulation table for double balanced mixer #1
!  Signal Level (dBm)   LO Level (dBm)
#       -10                7
!   M x LO ( Horizontal )   N x Signal (Vertical )
%0  1   2   3   4   5   6   7   8   9   10  11  12  13  14  15

 99 26  35  39  50  41  53  49  51  42  62  51  60  47  77  50
 24 0   35  13  40  24  45  28  49  33  53  42  60  47  63
 73 73  74  70  71  64  69  64  69  62  74  62  72  60
 67 64  69  50  77  47  74  44  74  47  75  44  70
 86 90  86  88  88  85  86  85  90  85  85  85
 90 80  90  71  90  68  90  65  88  65  85
 90 90  90  90  90  90  90  90  90  90
 90 90  90  90  90  87  90  90  90
 99 95  99  95  99  95  99  95
 90 95  90  90  90  99  90
 99 99  99  99  99  99
 90 99  90  95  90
 99 99  99  99
 90 99  90
 99 99
 99
END
```

Spurious Signal Simulation

In this simulation, the RF input frequency is swept, and the output spectrum at the IF is computed. Only the mixer is simulated to see the spurs generated when there is no filtering.

In this case, the RF frequency is stepped uniformly across a specified band.

Load-Pull Simulations

The Load Pull Simulation Guide is available in the schematic window under the **DesignGuide** menu.

Objective

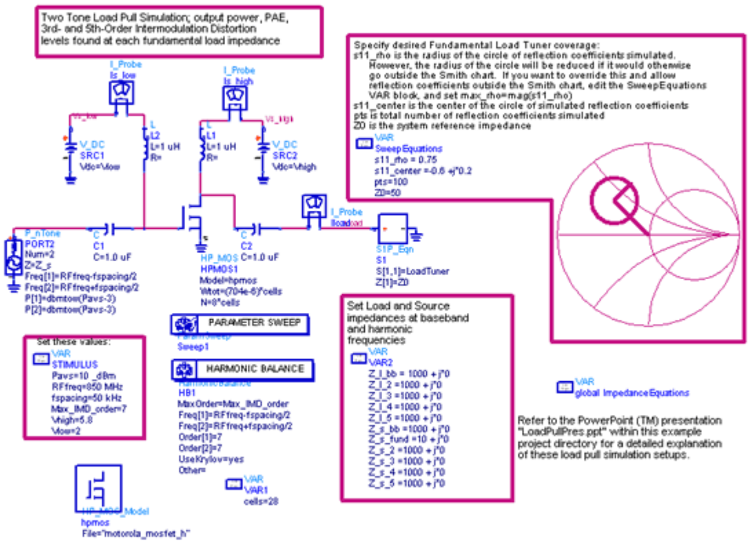
Shows how to do load-pull simulations to generate contours that indicate load impedances. These contours cause a certain power to be delivered to the load whenthey are presented to the output of a device along with the specified source impedances and available source power.

Setup

1. "HB1Tone_LoadPull" is a simulation set-up that generates actual contour lines for output power and power-added efficiency.
2. "HB2Tone_LoadPull" is identical to "HB1Tone_LoadPull" except that it shows all the equations, and includes some explanatory text.
3. "HB1Tone_LoadPull.dds" displays the output power and power-added efficiency contours and includes information on the equation syntaxs.
4. "ReflectionCoefUtility.dds" shows how the variables s11rho and s11centerdetermine the circular region of the Smith Chart within which load reflection coefficients are generated.
5. "HB1Tone_LoadPullMagPh" is identical to "HB1Tone_LoadPull", except that it uses a much simpler simulation setup. Instead of generating loads in a circular region of the Smith chart, the magnitude and phase of the load reflection coefficient are swept independently.
6. "HB1Tone_LoadPull_ConstPdel" uses an optimization to vary parameters on the schematic until a desired power is delivered to each load impedance. Contours of constant power-added efficiency and constant bias current are plotted on the data display.
7. "LoadPullMagPh_ConstPdel" is similar to "HB1Tone_LoadPull_ConstPdel",except that it varies the magnitude and phase of the load reflection coefficient rather than sweeping out a circular region of the Smith chart. Also the data display has contours

of constant operating and transducer power gain.

8. "HB2Tone_LoadPull" is a simulation set-up that generates actual contour lines for output power, power-added efficiency, third-order intermodulation distortion, and fifth-order intermodulation distortion.
9. "HB2Tone_LoadPull.dds" has the contour plots on one page, and the equations used to calculate output power, power-added efficiency, third-order intermodulation distortion, and fifth-order intermodulation distortion on another.
10. "HB2Tone_LoadPullMagPh" is identical to "HB2Tone_LoadPull", except that it uses a much simpler simulation setup. Instead of generating loads in a circular region of the Smith chart, the magnitude and phase of the load reflection coefficient are swept independently.
11. "contours" shows the old method of generating load pull contours, that was used in ADS before release 1.3. It generates load-pull contours on a Smith chart.These contours indicate load impedances that, when presented to the output of advice (along with the specified source impedances and available source power),would cause a certain power delivered to the load. "contours.dds"is the corresponding data display.


Two Tone Load Pull Simulation; output power, PAE, 3rd- and 5th-Order Intermodulation Distortion levels found at each fundamental load impedance
Specify desired Fundamental Load Tuner coverage:
s11_rho is the radius of the circle of reflection coefficients simulated.
However, the radius of the circle will be reduced if it would otherwise go outside the Smith chart. If you want to override this and allow reflection coefficients outside the Smith chart, edit the SweepEquations VAR block, and set max_rho=mag(s11_rho)
s11_center is the center of the circle of simulated reflection coefficients
pts is total number of reflection coefficients simulated
Z0 is the system reference impedance
SweepEquations
s11_rho = 0.75
s11_center =-0.6 +j*0.2
pts=100
Z0=50
PARAMETER SWEEP
HARMONIC BALANCE
Set these values:
STIMULUS
Pavs=10 _dBm
RFfreq=850 MHz
fspacing=50 kHz
Max_IMD_order=7
Vhigh=5.8
Vlow=2
Set Load and Source impedances at baseband and harmonic frequencies
Z_l_bb = 1000 + j*0
Z_l_2 =1000 + j*0
Z_l_3 =1000 + j*0
Z_l_4 =1000 + j*0
Z_l_5 =1000 + j*0
Z_s_bb =1000 + j*0
Z_s_fund =10 + j*0
Z_s_2 =1000 + j*0
Z_s_3 =1000 + j*0
Z_s_4 =1000 + j*0
Z_s_5 =1000 + j*0
global Impedance Equations
Refer to the PowerPoint (TM) presentation "LoadPullPres.ppt" within this example project directory for a detailed explanation of these load pull simulation setups.
HB1
MaxOrder=Max_IMD_order
Freq[1]=RFfreq-fspacing/2
Freq[2]=RFfreq+fspacing/2
Order[1]=7
Order[2]=7
UseKrylov=yes
VAR1
cells=28
hpmos
File="motorola_mosfet_h"
S1P_Eqn
S1
S[1,1]=LoadTuner
Z[1]=Z0
HP_MOS
HPMOS1
Model=hpmos
Wtot=(704e-6)*cells
N=8*cells
PORT2
Num=2
Z=Z_s
Freq[1]=RFfreq-fspacing/2
Freq[2]=RFfreq+fspacing/2
P[1]=dbmtow(Pavs-3)
P[2]=dbmtow(Pavs-3)
C1
C=1.0 uF
C2
C=1.0 uF
L1
L=1 uH
L2
L=1 uH
V_DC
SRC1
Vdc=Vlow
SRC2
Vdc=Vhigh
I_Probe
Is_low
Is_high
Iload

Analysis

Figure 1: Output power and PAE set-up

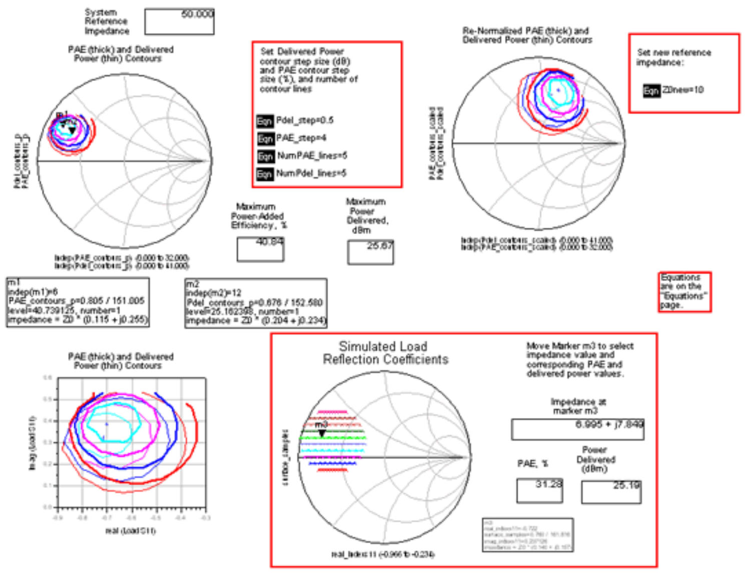
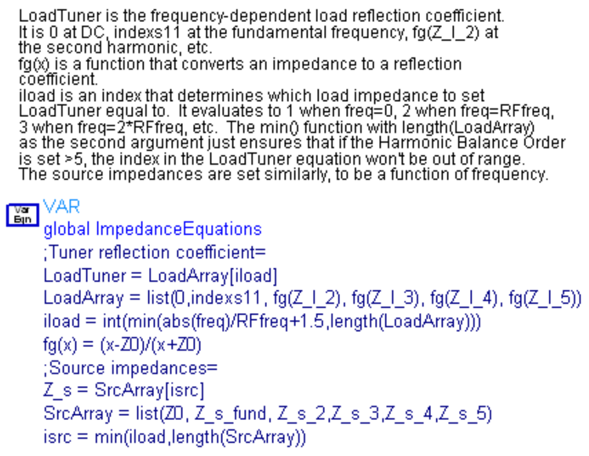


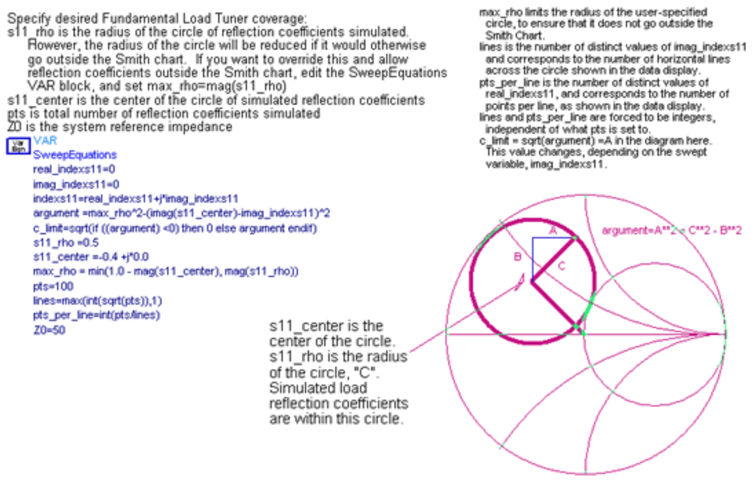
Figure 2: Load Tuner equation



LoadTuner is the frequency-dependent load reflection coefficient.
It is 0 at DC, indexs11 at the fundamental frequency, fg(Z_l_2) at
the second harmonic, etc.
fg(x) is a function that converts an impedance to a reflection
coefficient.
iload is an index that determines which load impedance to set
LoadTuner equal to. It evaluates to 1 when freq=0, 2 when freq=RFfreq,
3 when freq=2*RFfreq, etc. The min() function with length(LoadArray)
as the second argument just ensures that if the Harmonic Balance Order
is set >5, the index in the LoadTuner equation won't be out of range.
The source impedances are set similarly, to be a function of frequency.

```
VAR
global ImpedanceEquations
;Tuner reflection coefficient=
LoadTuner = LoadArray[iload]
LoadArray = list(0,indexs11, fg(Z_l_2), fg(Z_l_3), fg(Z_l_4), fg(Z_l_5))
iload = int(min(abs(freq)/RFfreq+1.5,length(LoadArray)))
fg(x) = (x-Z0)/(x+Z0)
;Source impedances=
Z_s = SrcArray[isrc]
SrcArray = list(Z0, Z_s_fund, Z_s_2,Z_s_3,Z_s_4,Z_s_5)
isrc = min(iload,length(SrcArray))
```

Figure 3: Fundamental Load Tuner coverage and sweep equation



Specify desired Fundamental Load Tuner coverage:
s11_rho is the radius of the circle of reflection coefficients simulated.
However, the radius of the circle will be reduced if it would otherwise
go outside the Smith chart. If you want to override this and allow
reflection coefficients outside the Smith chart, edit the SweepEquations
VAR block, and set max_rho=mag(s11_rho)
s11_center is the center of the circle of simulated reflection coefficients
pts is total number of reflection coefficients simulated
Z0 is the system reference impedance

```
VAR
SweepEquations
real_indexs11=0
imag_indexs11=0
indexs11=real_indexs11+j*imag_indexs11
argument =max_rho^2-(imag(s11_center)-imag_indexs11)^2
c_limit=sqrt(if ((argument) <0) then 0 else argument endif)
s11_rho =0.5
s11_center =-0.4 +j*0.0
max_rho = min(1.0 - mag(s11_center), mag(s11_rho))
pts=100
lines=max(int(sqrt(pts)),1)
pts_per_line=int(pts/lines)
Z0=50
```

max_rho limits the radius of the user-specified
circle, to ensure that it does not go outside the
Smith Chart.
lines is the number of distinct values of imag_indexs11
and corresponds to the number of horizontal lines
across the circle shown in the data display.
pts_per_line is the number of distinct values of
real_indexs11, and corresponds to the number of
points per line, as shown in the data display.
lines and pts_per_line are forced to be integers,
independent of what pts is set to.
c_limit = sqrt(argument) =A in the diagram here.
This value changes, depending on the swept
variable, imag_indexs11.

s11_center is the
center of the circle.
s11_rho is the radius
of the circle, "C".
Simulated load
reflection coefficients
are within this circle.

Notes

- Simulation controller used: Harmonic Balance.
- Only the impedance at the fundamental frequency is varied. Impedances at harmonic frequencies are set independently but they are not varied in this example. Detailed information on setting up the load and source impedances for the load-pull can be found in the example descriptions.
- See also the example workspace *Loadpull Simulations in ADS* (examples)

Radar Applications Guide

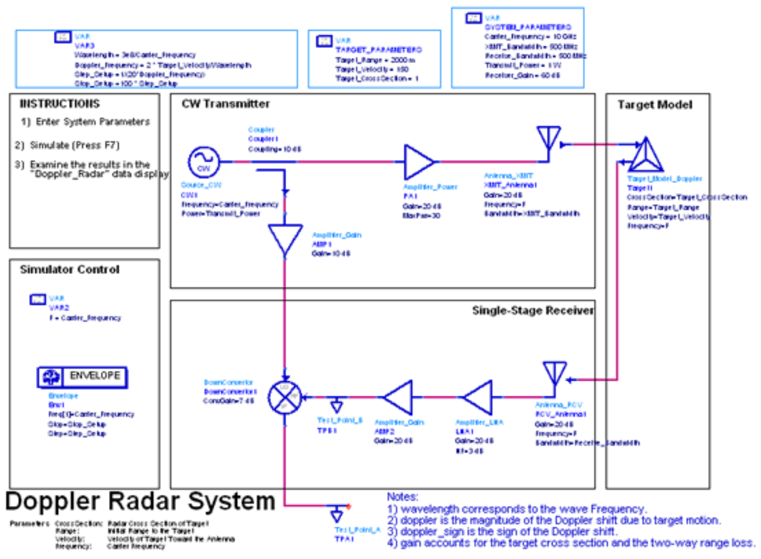
The Radar Applications Guide is available from the schematic window under the **DesignGuide** menu.

Objective

The objective of the Radar Applications guide is to demonstrate the capability of Advanced Design System to simulate Pulse compression simulation in a radar system and to demonstrate IFM receiver simulation for EW application.

- "FM-CW Radar Simulation" demonstrates a simple Doppler radar simulation using envelope simulator. A user defined target model is implemented. The echo signal is a function of target range, velocity, and cross-section. The Doppler shift in frequency is plotted.

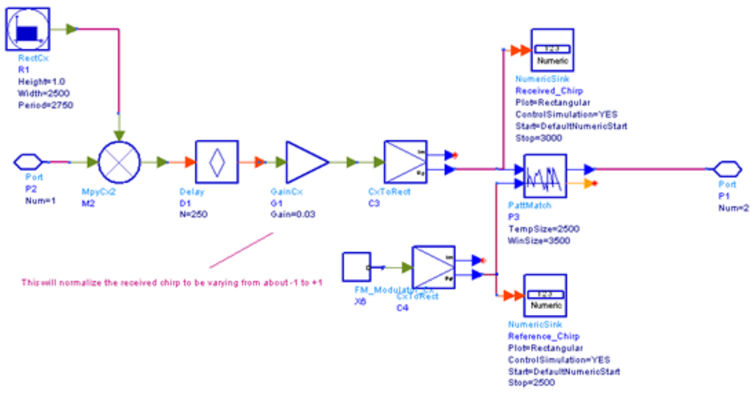
Figure 1: Doppler Radar System



Notes:
1) wavelength corresponds to the wave Frequency.
2) doppler is the magnitude of the Doppler shift due to target motion.
3) doppler_sign is the sign of the Doppler shift.
4) gain accounts for the target cross section and the two-way range loss.

LFM Radar Component and Simulation Setup

Figure 2: LFM Radar Component and Simulation



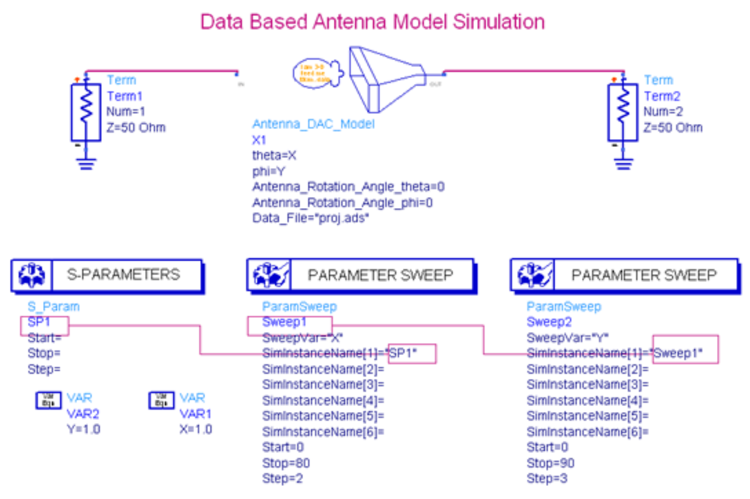
Setup

1. "LFM Waveform Generation" demonstrates generation of linear frequency modulated signal using numerical signal processing components. The generation ofchirp signal for different compression ratio is also demonstrated.
2. "LFM Using Direct Digital Synthesis" demonstrates direct digital synthesis of a Linear Frequency modulated signal. Numeric synthesizable DSP components areused for direct digital synthesis, and output waveform is plotted for 4, 8, and 32bit resolution.
3. "LFM Chirped Transmitter and Receive Simulation" demonstrates LFM pulse compression implementation a using pattern match component at the receiver. Theplot shows the transmitted waveform and compressed pulse signal at the receiver.
4. "LFM Two Target Modeling and Detection" demonstrates the detection of two targets using a single pulse.
5. "LFM with RF Transmitter" demonstrates co-simulation of DSP and RF Components. The upconverter is designed using RF system components and co-simulation between Agilent Ptolemy and Envelope simulator is demonstrated. The RF signal is down converted and the pulse is compressed using pattern match component.
6. "LFM with RF Transmitter and Receiver" demonstrates cosimulation of RF with DSP. This system includes an RF transmitter and receiver. Sensitivity time control, automatic gain control, and channel modeling are included in the simulation. The simulation is performed only for a single pulse period.
7. "LFM Radar Receiver Simulation" demonstrates LFM pulse compression using an alternative implementation of Chirp signal generation and Compression Filter. A parametric model of LFM Signal Generator and Channel model is implemented with this simulation.

Antenna Components and Simulations

"MOM2ADS Antenna Radiation Pattern Translation Utility" can be used to translate the 3-D radiation pattern of an antenna designed using Momentum forsimulation in ADS. Once the planar antenna is analyzed using Momentum, the 3-D radiation pattern can be calculated using Momentum Visualization. The normalized electric far-field components for the complete hemisphere will be saved in ASCII format in the file "proj.fff" in the <workspace_dir>/mom_dsn/<design_name>directory. Select proj.fff using the file browser in the MOM2ADS Antenna radiation Pattern Translation Utility. The translation utility will calculate the total electric field as a function of theta and phi and change file format so that it can be accessed using the DAC component in ADS. This new file "proj.ads" will be added to the same directory as the original "proj.fff" file.

Figure 3: Data Based Antenna Model Simulation

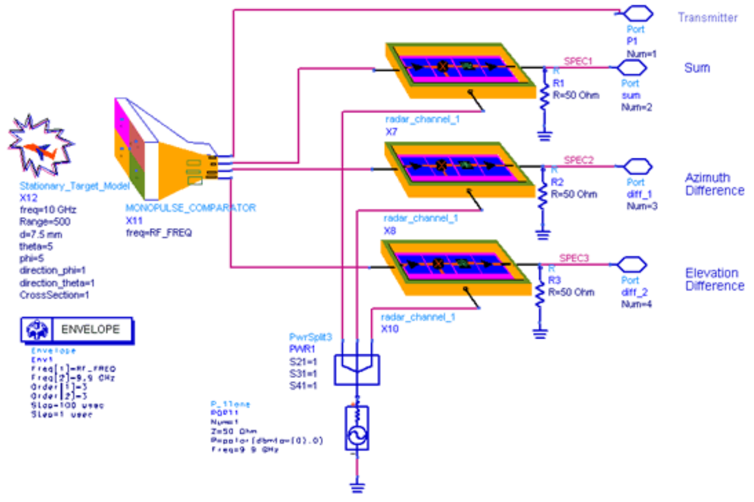


Setup

1. "Data Based Antenna Model Simulation" demonstrates the simulation of a 3-D data based antenna model. The simulation accepts the 3-D radiation pattern file"proj.ads" generated by the MOM2ADS File Translation Utility. The radiation pattern can be simulated as a function of theta, phi, and Antenna Rotation Angle.
2. "2-D Antenna Radiation Pattern Implementation Using Data Display"demonstrates the Sin(X)/X implementation of a 2-D Antenna radiation pattern. The marker can be scrolled to change the antenna direction. The data display is used for antenna radiation pattern verification before an equation can be implemented as a behavioral model using the S2P equation based component.
3. "2-D Antenna Radiation Pattern Simulation" demonstrates the simulation of a2-D user-defined Antenna behavioral model for various Antenna Rotation Angle.Sin(X)/X radiation pattern equations are implemented using the S2P_Equation based model to generate radiation pattern. The model provides ideal isolation in the reverse direction (S[1,2]=0), and can be modified as desired.

Monopulse Radar Components and Simulations

Figure 4: Monopulse RF System Simulation

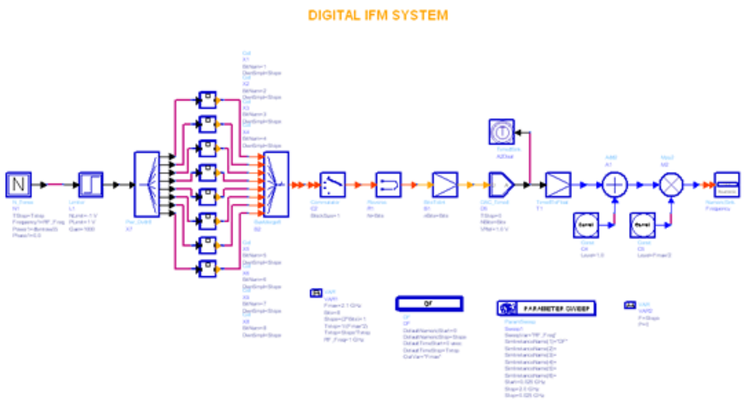


Setup

1. "Monopulse Antenna Feed Simulation for Sum and Difference Channel"demonstrates the simulation of a Target and Monopulse Radar feed model integrated with a 2-D radiation pattern. +25 degrees Antenna Rotation Angle for the pairs of vertical antenna are defined. The simulation is performed as a function of azimuth angle theta keeping the target position fixed. The Sum and Difference channel output is plotted.
2. "RF Front End Simulation for Pulse Modulated Signal" demonstrates envelope simulation for an RF subsystem. The data display shows the pulse parameter derived from measurement. The subsystem shows SPST switch modeling.
3. "Monopulse RF System Simulation" demonstrates the RF simulation of a three channel Monopulse system with double down conversion. The sum channel, azimuth,and elevation channel response is plotted. Target model can be modified to include Doppler frequency effects.
4. "Target Azimuth Angle Tracking" demonstrates a three channel Monopulse Radar Receiver integrated with a 2-D Antenna Radiation Pattern behavioral model to calculate target Azimuth Angle. The simulation is performed as a function of target azimuth position and the target azimuth angle is calculated and plotted as a function of target position.
5. "Pulse Compression Using Barker Code" demonstrates the pulse compression simulation using Barker Sequence. The data display shows the transmitted phase modulated signal and the received pulse compressed waveform.
6. "Three Channel Monopulse Radar Simulation" demonstrates a Barker sequence pulse compression implementation in a three channel Monopulse radar receiver. The compressed pulse is shown for sum, azimuth, and elevation channel.

IFM Receiver Components and Simulations

Figure 5: Eight Channel Digital IFM Simulation



Setup

1. "IFM Correlator Simulation" demonstrates cosimulation of a single channel IFM receiver. Output angle is plotted and instantaneous frequency is calculated.

2. "Single Channel simulation of IFM" demonstrates envelope simulation for an RF subsystem. The data display shows the pulse parameter derived from measurement.The subsystem shows SPST switch modeling.
3. "Two Tone Simulation of IFM" demonstrates an IFM single channel cosimulationin the presence of two input independent frequencies. The power level of one signal tone is kept fixed while the power level of the other signal tone is varied from minimum value to maximum value. The data display shows IFM always respond to the highest power level signal, and when two-power levels are nearly equal, the output of IFM shows uncertainty in angle measurement.
4. "Four Channel Simulation of IFM" demonstrates a four-channel simulation of an IFM system. A DSP algorithm can be implemented to resolve ambiguity infrequency measurement over a wide frequency bandwidth.
5. "Eight Channel Digital IFM System Simulation" demonstrates the Agilent Ptolemy implementation of an eight channel IFM system. The instantaneous output frequency as a function of input RF frequency is plotted.

Note
The information on this page is obsolete, but is kept here for archival purposes. Please go to *Signal Integrity Examples* (examples) for an up-to-date introduction and examples for signal integrity. *Click here* (sigint) for documentation of the *Signal Integrity DesignGuide* (sigint)

Signal Integrity Simulations

Setup

1. "Eye Closure Measurement " shows three new functions for eye parameter calculation. This is used to demonstrate the customization capabilities of ADS.The AEL function shown on the data display page can be copied to $HPEESOF_DIR/expressions/ael/ user_defined_fun.ael to calculate eye amplitude, eye height, and eye closure.
2. "IBIS Simulation of cross talk" demonstrates a partial IBIS model implementation in ADS. Pullup/pulldown, power and ground clamps are simulated using SDD and DataAccess Components. The data display shows crosstalk. Slew rate is not implemented in this example.
3. "TDR Simulation" shows time domain simulation of Multilayer Interconnect transmission line models using TDR simulation instrument. The simulated and measurement data is compared.
4. "Differential and Common mode S Parameter Basic" shows the data display convert four port single ended S-parameter to Differential and common mode S-parameter.
5. "Differential Impedance Simulation" shows how ADS can be used to calculate differential impedance.
6. "Common Impedance Simulation" demonstrates how ADS can be used to calculate common mode impedance.

Figure 1: Laser Driver circuit simulation with and without jitter source

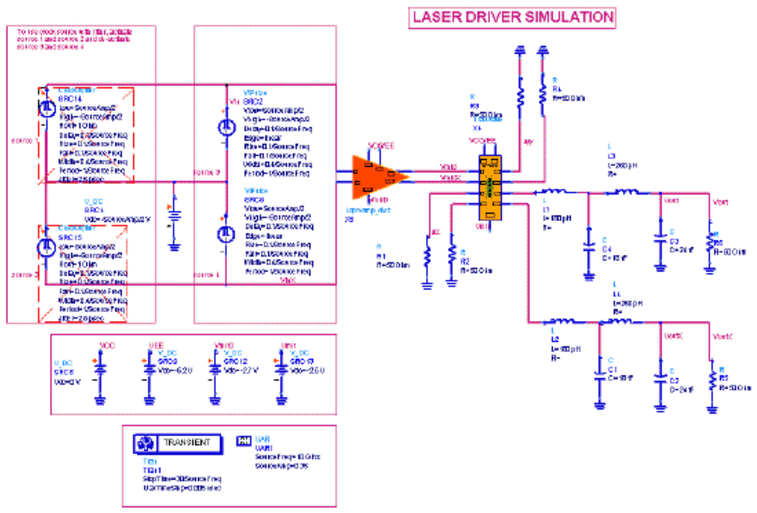


Figure 2: Laser Driver Simulation Response

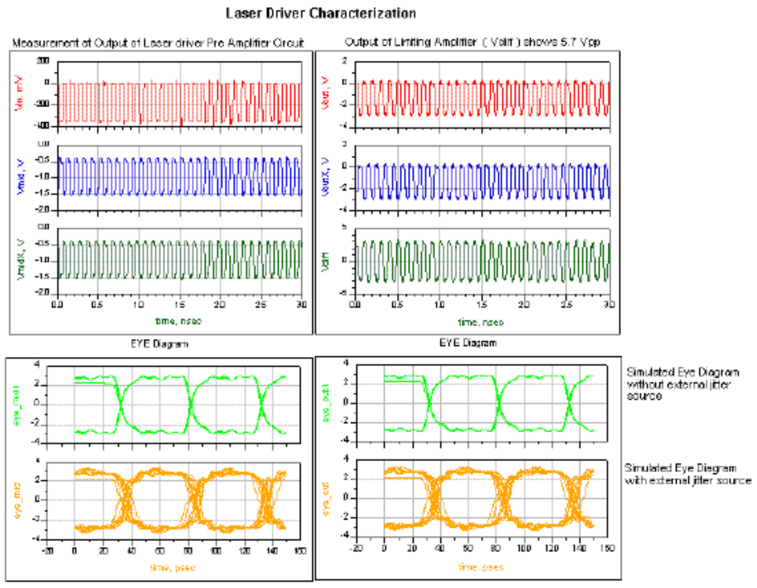
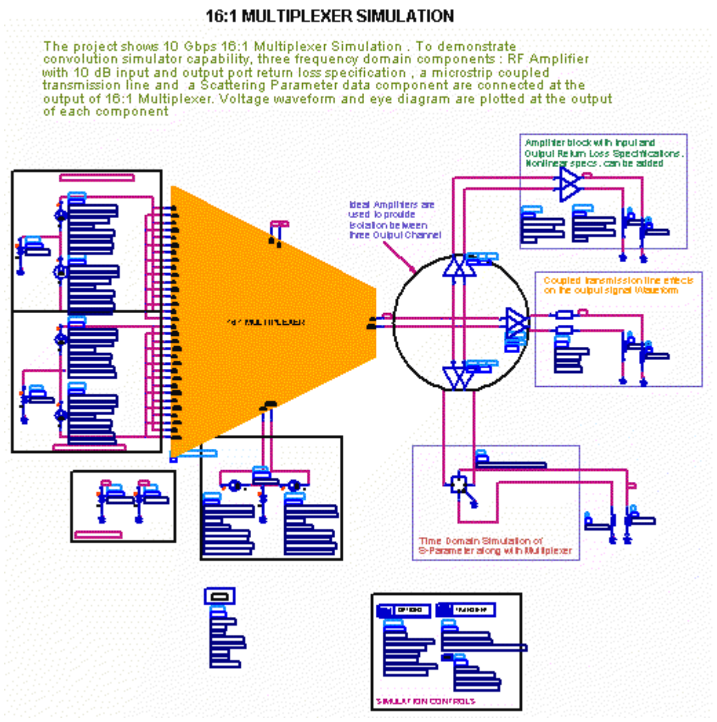


Figure 3: 16:1 Multiplexer with Clock Distribution Circuit


16:1 MULTIPLEXER SIMULATION

The project shows 10 Gbps 16:1 Multiplexer Simulation . To demonstrate convolution simulator capability, three frequency domain components : RF Amplifier with 10 dB input and output port return loss specification , a microstrip coupled transmission line and a Scattering Parameter data component are connected at the output of 16:1 Multiplexer. Voltage waveform and eye diagram are plotted at the output of each component

Amplifier block with Input and Output Return Loss Specifications. Nonlinear specs. can be added
Ideal Amplifiers are used to provide isolation between three Output Channel
16:1 MULTIPLEXER
Coupled transmission line effects on the output signal Waveform
Time Domain Simulation of S-Parameter along with Multiplexer
SIMULATION CONTROLS

Figure 4: 16:1 Multiplexer Simulation Result

The Input signal pattern has been selected to provide 101010.. bit at the output

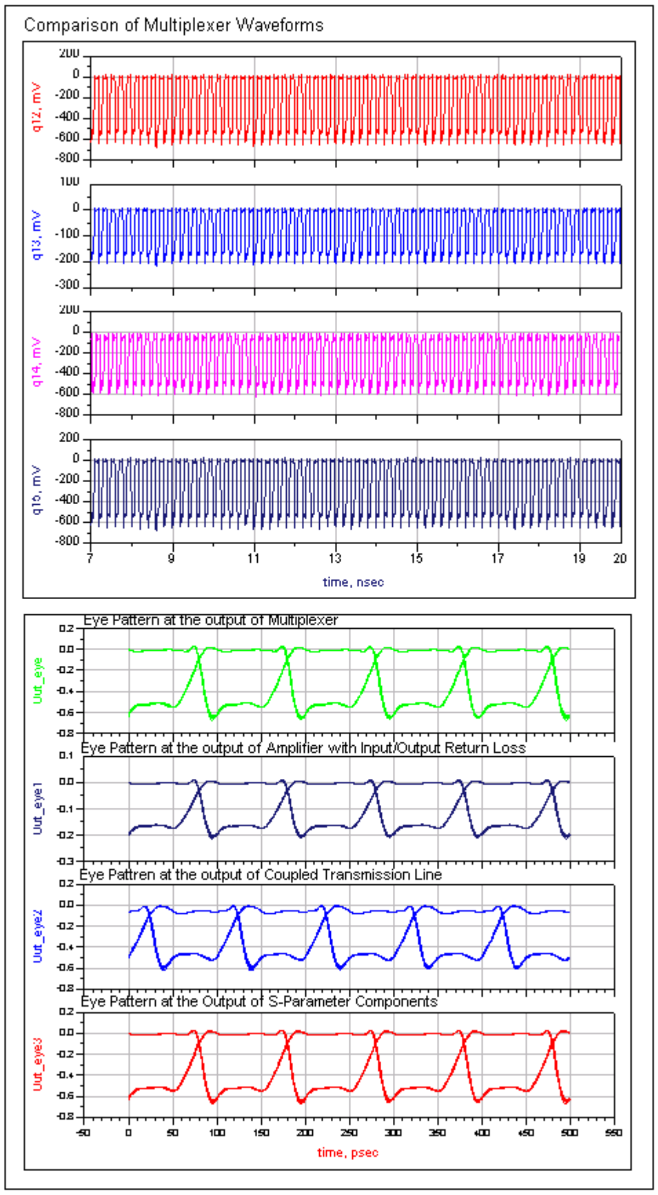
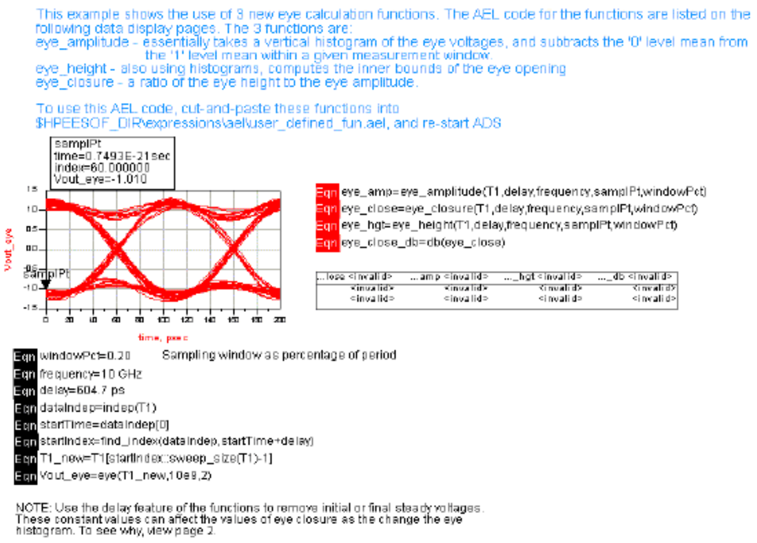


Figure 5: Eye Parameter Calculation



This example shows the use of 3 new eye calculation functions. The AEL code for the functions are listed on the following data display pages. The 3 functions are:
eye_amplitude - essentially takes a vertical histogram of the eye voltages, and subtracts the '0' level mean from the '1' level mean within a given measurement window.
eye_height - also using histograms, computes the inner bounds of the eye opening
eye_closure - a ratio of the eye height to the eye amplitude.

To use this AEL code, cut-and-paste these functions into $HPEESOF_DIR\expressions\ael\user_defined_fun.ael, and re-start ADS

```
samplPt
time=0.7493E-21sec
index=60.000000
Vout_eye=-1.010
```

```
Eqn eye_amp=eye_amplitude(T1,delay,frequency,samplPt,windowPct)
Eqn eye_close=eye_closure(T1,delay,frequency,samplPt,windowPct)
Eqn eye_hgt=eye_height(T1,delay,frequency,samplPt,windowPct)
Eqn eye_close_db=db(eye_close)
```

...lose <invalid>	...amp <invalid>	..._hgt <invalid>	..._db <invalid>
<invalid>	<invalid>	<invalid>	<invalid>
<invalid>	<invalid>	<invalid>	<invalid>

```
Eqn windowPct=0.20        Sampling window as percentage of period
Eqn frequency=10 GHz
Eqn delay=604.7 ps
Eqn dataIndep=indep(T1)
Eqn startTime=dataIndep[0]
Eqn startIndex=find_index(dataIndep, startTime+delay)
Eqn T1_new=T1[startIndex::sweep_size(T1)-1]
Eqn Vout_eye=eye(T1_new,10e9,2)
```

NOTE: Use the delay feature of the functions to remove initial or final steady voltages. These constant values can affect the values of eye closure as the change the eye histogram. To see why, view page 2.

VPI ADS Link

Waveform transfer between VPItransmissionMaker and Advanced Design System

The increased data rate in optical fiber communication poses many design challenges to high-speed designers. Accurate modeling and simulation of the complete communication signal path is essential to overcome many complex design issues and to reduces the product design cycle.

VPI software provides the designer an excellent optical signal simulation environment and an exhaustive photonic model library for optical system design. The Advanced Design System from Agilent EEsof provides some unique and powerful simulation capabilities to design high speed Analog circuits.

This application note is written for the designer who may want to use the best of both the optical and electrical world and wants to design electrical circuits for optical systems. There is no EDA tool existing today, which is capable of providing true co-simulation capability between optical system models and electrical circuits to predict performance of the complete communication signal path.

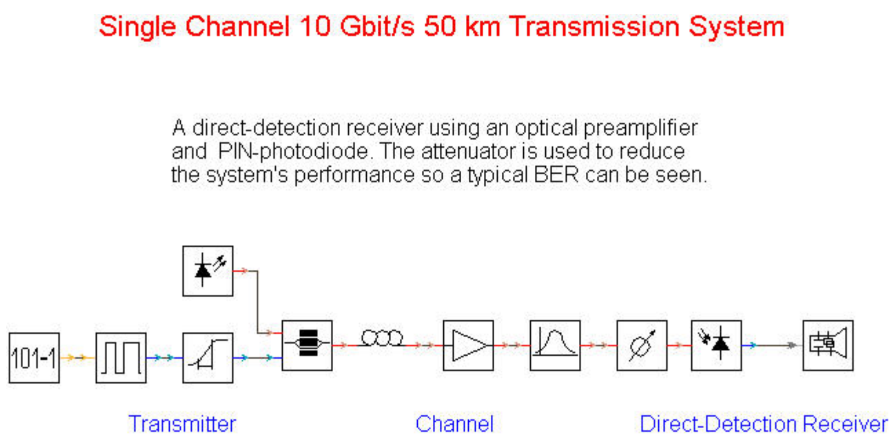
In an effort to integrate the optical and electrical design environments, this application note highlights a simple process to translate the output signal of VPItransmissionMaker software used for photonic system design to the Advanced Design System for electrical circuit design and vice versa using VPI Link utility. The VPI Link utility will enable the user to simulate the complete communication signal path of any high speed electro-optic design. The VPI Link utility can be downloaded from Agilent EEsof web site for free and can be installed over the Advanced Design System. The VPI Link utility will help you to translate the file format between the two software and demonstrate the use of simulation results to define data based time domain sources to analyze electrical circuits in Advanced Design System or Optical system in VPItransmissionMaker software. An example of a 10 Gbps direct detection receiver using an optical preamplifier and a PIN photodiode is used here to demonstrate data transfer between VPI and ADS software. In ADS this data is used for a 10 Gbps transimpedance amplifier simulation.

File translation from VPItransmissionMaker software to ADS software uses a four-step procedure:

1. Generate the time domain electrical signal waveform at the output of optical receiver in VPItransmissionMaker software.
2. Output the time domain electrical waveform signal to an ASCII file.
3. Convert the file format from VPItransmissionMaker to Advanced Design System.
4. Use the time domain data file to define time dependent voltage and current sources in Advanced Design System.

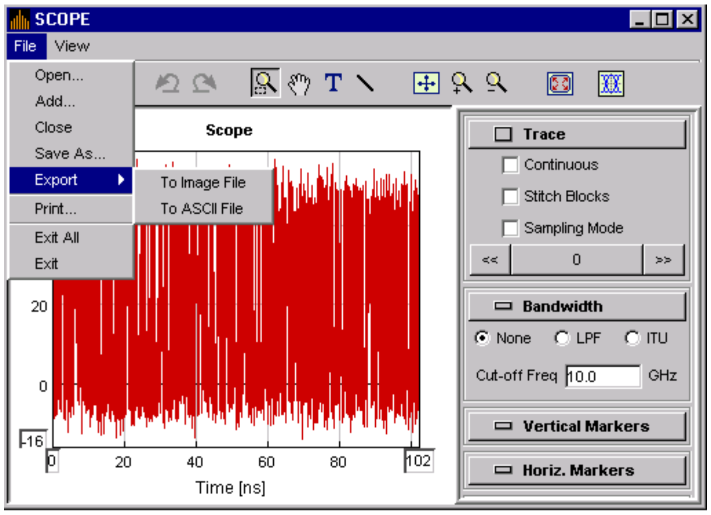
The figure 1 shows a single channel 10 Gbps transmission system designed using VPItransmissionMaker software. This transmission system uses a CW laser diode, an external modulator, fiber channel, optical amplifier and a PIN photodiode detector at the output. The system uses a 10 Gbps data bit stream source at the input. The VPItransmissionMaker software generates an electrical signal at the output of a PIN-photodiode detector and needed to be translated into the Advanced Design System to simulate high-speed analog circuits.

Figure1: Optical Transmission System design using VPItransmissionMaker software


Single Channel 10 Gbit/s 50 km Transmission System
A direct-detection receiver using an optical preamplifier and PIN-photodiode. The attenuator is used to reduce the system's performance so a typical BER can be seen.
101-1
Transmitter
Channel
Direct-Detection Receiver

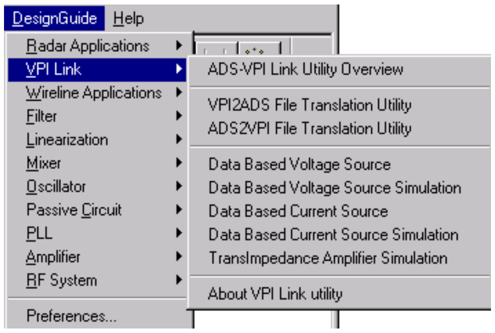
Viscope/Visualizer in VPItransmissionMaker software captures the time domain output waveform, which is written as an ASCII data file by using the "File>Export>To ASCII File".

Figure 2: Electrical waveform at the output of PIN Photodiode



VPI2ADS file translation from the VPI Link utility is used to convert the VPI file format to the time domain file format required by the Advanced Design System.

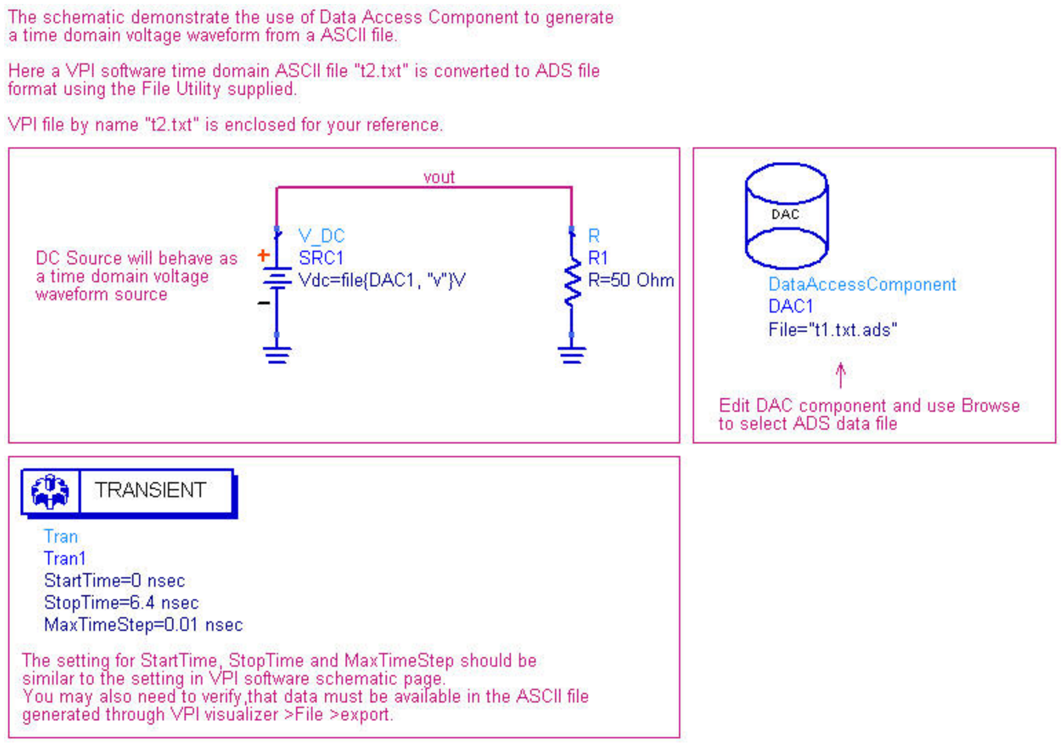
Figure 3: VPI Link Menu Structure in Advanced Design System and data based time domain voltage and current source


DesignGuide
Help
Radar Applications
VPI Link
Wireline Applications
Filter
Linearization
Mixer
Oscillator
Passive Circuit
PLL
Amplifier
RF System
Preferences...
ADS-VPI Link Utility Overview
VPI2ADS File Translation Utility
ADS2VPI File Translation Utility
Data Based Voltage Source
Data Based Voltage Source Simulation
Data Based Current Source
Data Based Current Source Simulation
TransImpedance Amplifier Simulation
About VPI Link utility

The file generated by VPI software will have time and amplitude information and can be used to define a time domain current source to simulate circuits like transimpedance amplifiers or as a time domain voltage waveform to simulate circuits like traveling wave amplifier. For convenience both the voltage and current source simulation setup are provided with VPI Link utility. You can pull these sources directly from Advanced Design System VPI Link menu and use it for circuit simulation.

Please note that the file name has to be enclosed in quotes (" ") if you are using any of above source.

Figure 4: Data based time domain voltage sources simulation. [DesignGuide > VPI link > Data based Voltage Source Simulation]



The schematic demonstrate the use of Data Access Component to generate a time domain voltage waveform from a ASCII file.

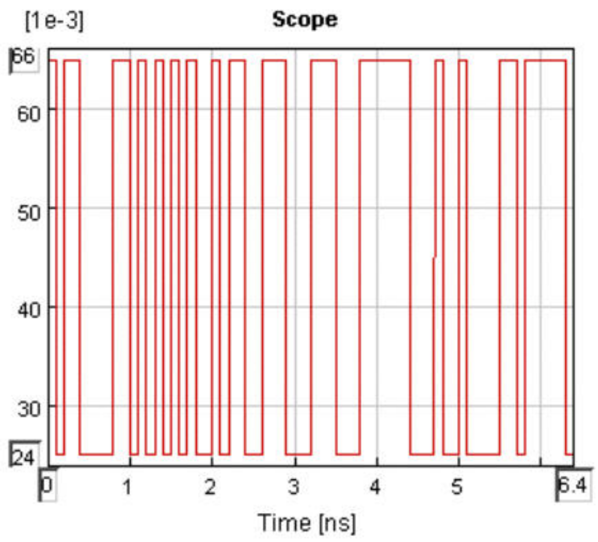
Here a VPI software time domain ASCII file "t2.txt" is converted to ADS file format using the File Utility supplied.

VPI file by name "t2.txt" is enclosed for your reference.

The setting for StartTime, StopTime and MaxTimeStep should be similar to the setting in VPI software schematic page.
You may also need to verify,that data must be available in the ASCII file generated through VPI visualizer >File >export.

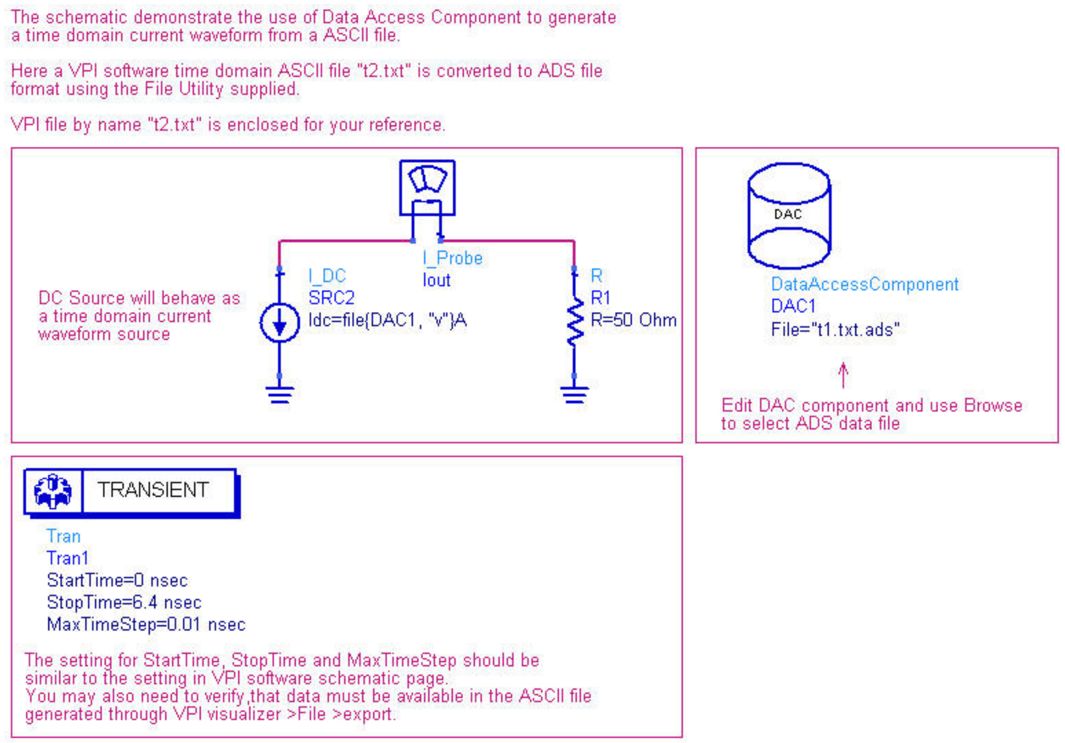
The data based voltage source uses a DC voltage source which is used in conjunction with Data Access Component (DAC) component to create time domain voltage waveform. The DAC component reads an ASCII file containing time domain waveform data and supply to DC voltage source.

Figure 5: Comparison of Advanced Design System and VPItransmissionMaker simulated waveform



The above figure shows the simulated result of VPI and ADS software to be identical.

Figure 6: Data based time domain current source simulation. DesignGuide > VPI Link > Data Based Current Source Simulation



The schematic demonstrate the use of Data Access Component to generate a time domain current waveform from a ASCII file.

Here a VPI software time domain ASCII file "t2.txt" is converted to ADS file format using the File Utility supplied.

VPI file by name "t2.txt" is enclosed for your reference.

DC Source will behave as a time domain current waveform source

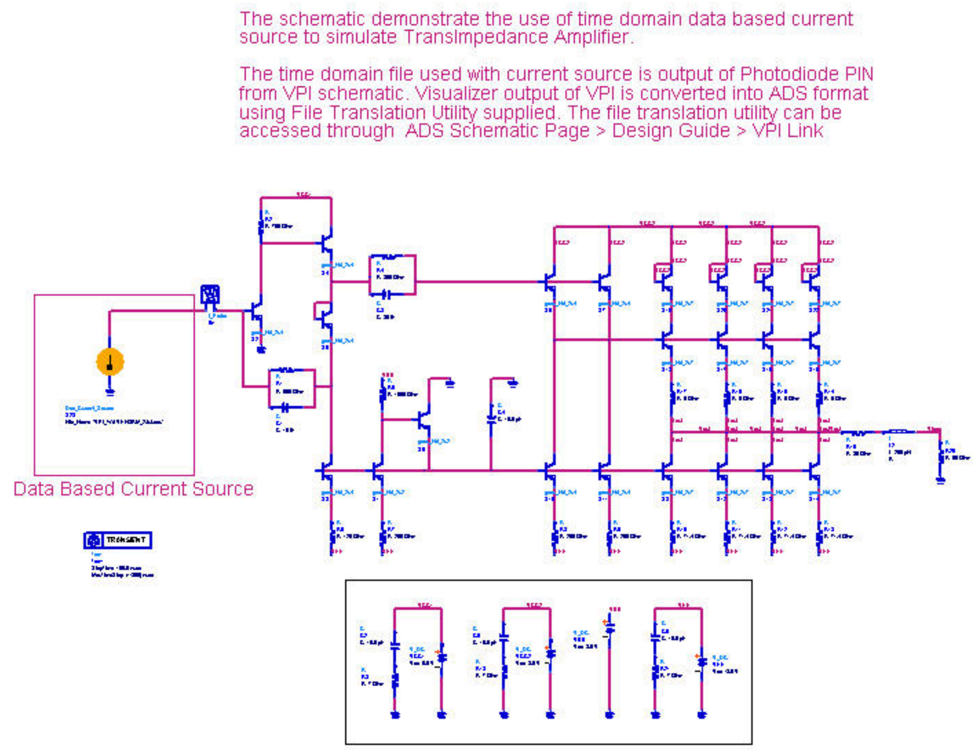
Edit DAC component and use Browse to select ADS data file

The setting for StartTime, StopTime and MaxTimeStep should be similar to the setting in VPI software schematic page.
You may also need to verify,that data must be available in the ASCII file generated through VPI visualizer >File >export.

The above figure shows the simulation setup required for a data based current source.

To demonstrate the use of VPI Link utility for electrical circuit simulation, the output of PIN photodiode in 10 Gbps direct detection transmission system is translated from VPItransmissionMaker to Advanced Design System to simulate the performance of a transimpedance amplifier. The time waveform is converted to ADS format using VPI2ADS File translation utility and used with the data based current source available in the VPI Link utility.

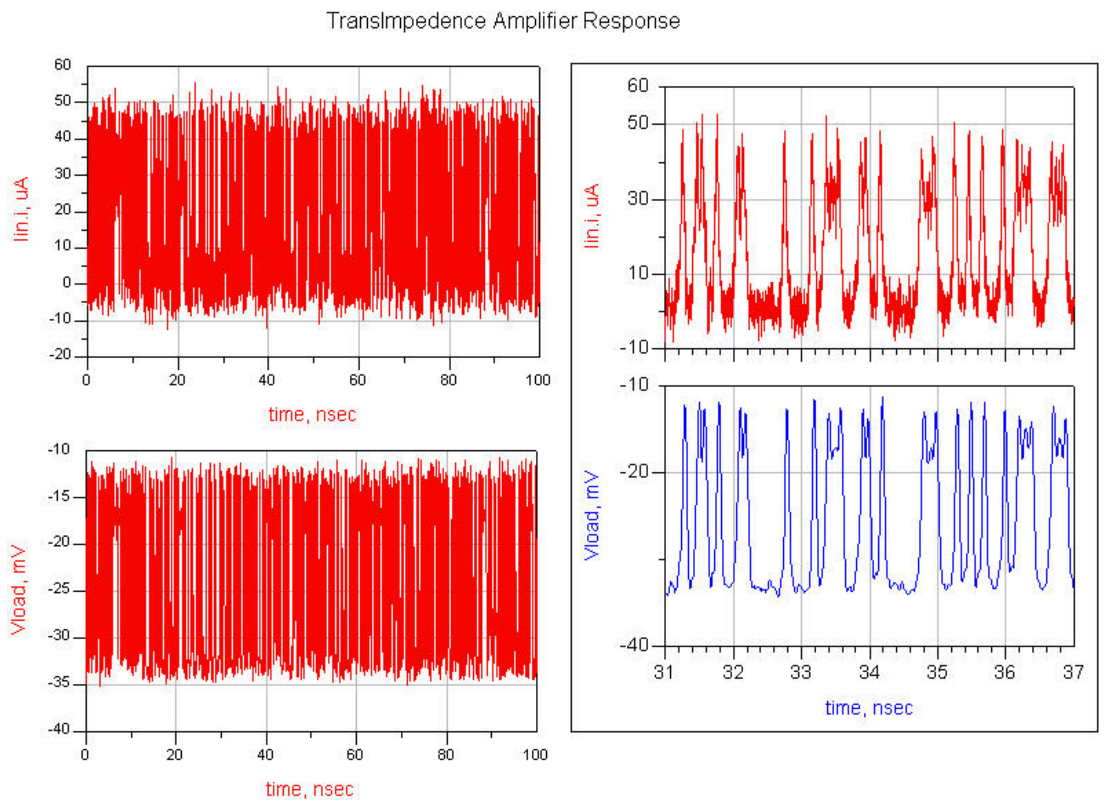
Figure 7: Transimpedance amplifier simulation using data based time domain source. [DesignGuide > VPI Link > TransImpedance Amplifier Simulation]



The schematic demonstrate the use of time domain data based current source to simulate TransImpedance Amplifier.

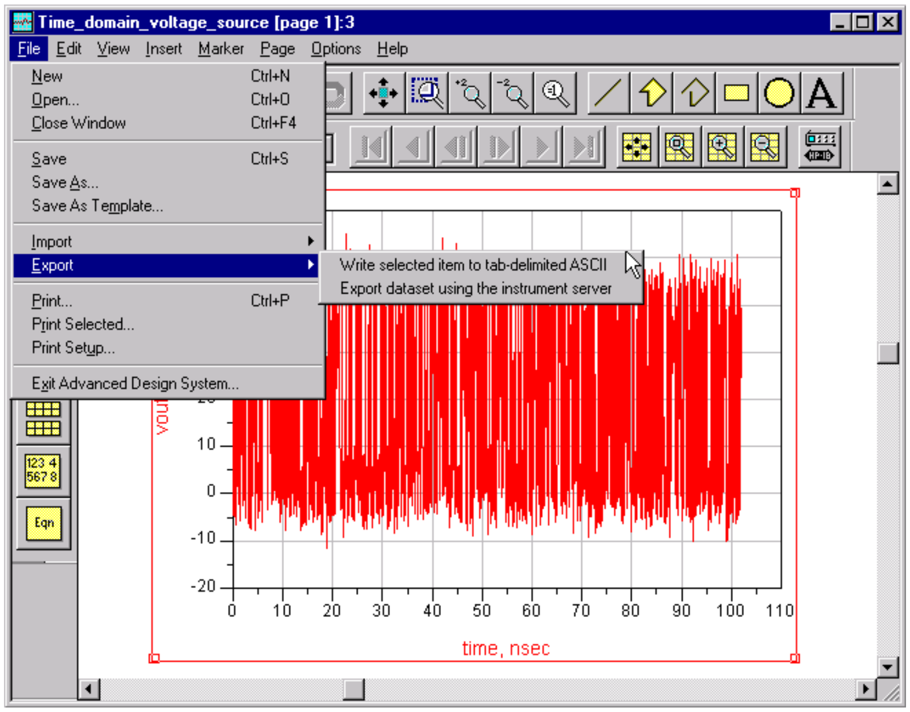
The time domain file used with current source is output of Photodiode PIN from VPI schematic. Visualizer output of VPI is converted into ADS format using File Translation Utility supplied. The file translation utility can be accessed through ADS Schematic Page > Design Guide > VPI Link

Figure 8: Simulated input current and output voltage waveform



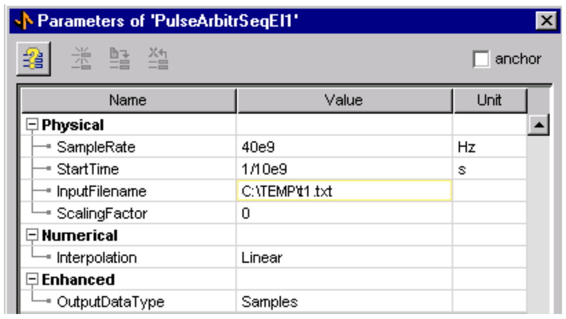
ADS2VPI file translation utility enables you to design high-speed electrical circuit and translates simulated response to VPItransmissionMaker software for optical system design and simulation.

Figure 9: Generation of ASCII file in Advanced Design System


Time_domain_voltage_source [page 1]:3
File Edit View Insert Marker Page Options Help
New Ctrl+N
Open... Ctrl+O
Close Window Ctrl+F4
Save Ctrl+S
Save As...
Save As Template...
Import
Export
Print... Ctrl+P
Print Selected...
Print Setup...
Exit Advanced Design System...
Write selected item to tab-delimited ASCII
Export dataset using the instrument server
time, nsec

To transfer files form ADS to VPI software, an ADS time domain waveform can be exported as a tab-delimited ASCII file using File > Export > Write selected items to tab-delimited ASCII. The file generated using this procedure can be translated into VPItransmissionMaker software using ADS2VPI file translation utility. This translated file can be used with PulseArbitrEl component available in VPItransmissionMaker software to provide input signal source for optical system simulation. As on March 2002 PulseArbitrEl component in VPItransmissionMaker ver 4.0 requires a software patch to read the data file correctly. The VPI software patch can be obtained from vpisupport@vpisystems.com.

Figure 10: PulseArbitrEl component in VPItransmissionMaker software generates a time domain signal from file to simulate optical network


Parameters of 'PulseArbitrSeqEl1'

anchor

Name	Value	Unit
Physical		
SampleRate	40e9	Hz
StartTime	1/10e9	s
InputFilename	C:\TEMP\t1.txt	
ScalingFactor	0	
Numerical		
Interpolation	Linear	
Enhanced		
OutputDataType	Samples	

Note
The ASCII files generated on PC may require conversion to Unix format before it can be used on HP or SUN platform

Wireline Applications

The Wireline Applications Guide is available from the schematic window under the **DesignGuide** menu.

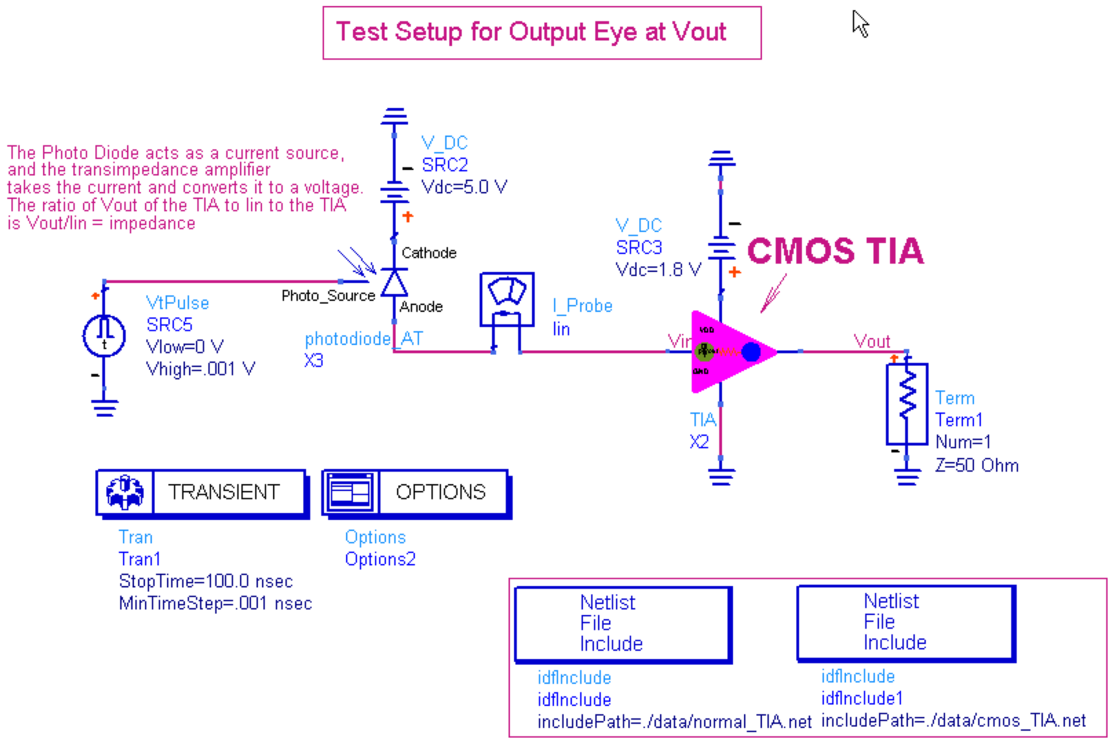
Objective

The objective of Wireline Applications is to demonstrate the capability of Advanced Design System to simulate high datarate circuits. The Wireline Applications are categorized into Analog Component and Test, Digital Component and Test, Noise Jitter, and Signal Integrity Simulations.

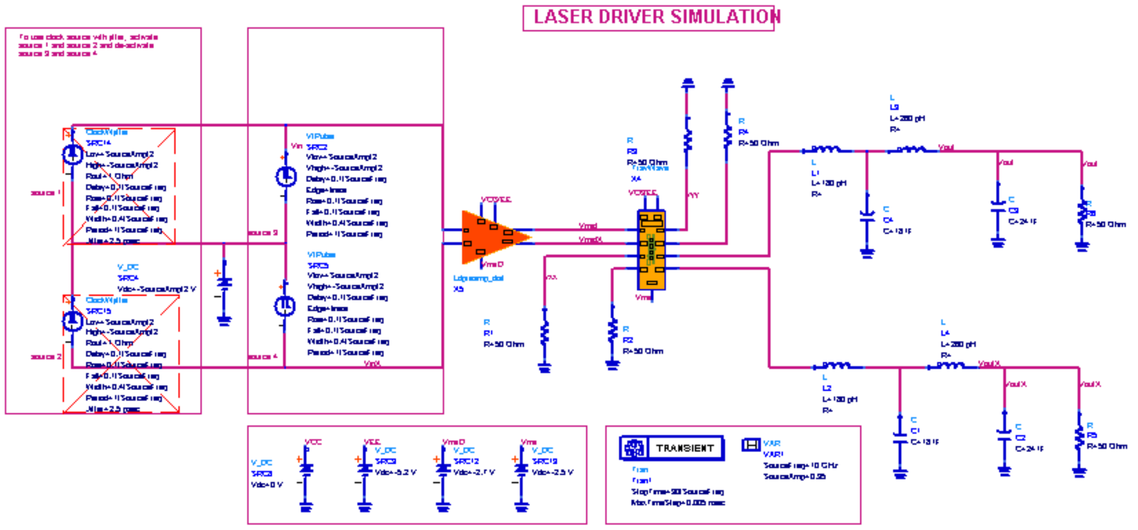
Analog Component and Test

Setup

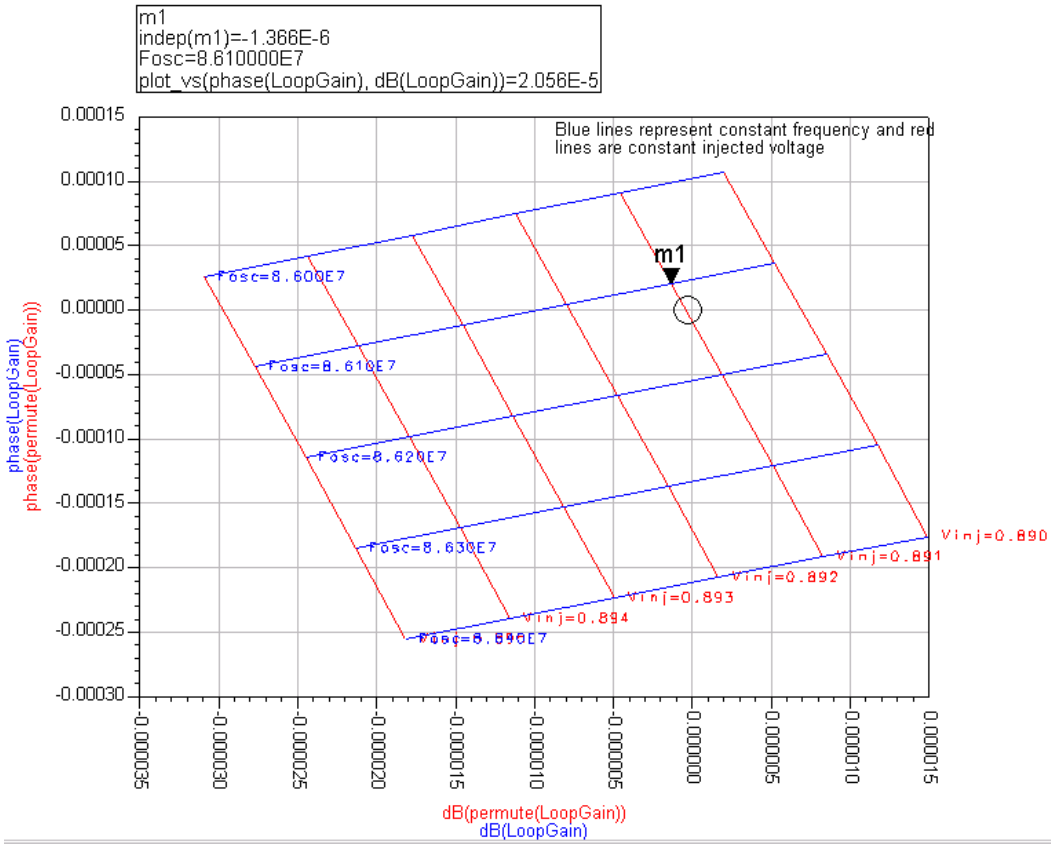
1. "Photo diode equivalent circuit "acts as a current source. The equivalent circuit uses a Voltage controlled current source to generate an output current waveform. This model can be used to find the performance of a Transimpedance Amplifier.
2. "1.25 Gbps CMOS Transimpedance Amplifier Linear Simulation" demonstrates a CMOS Transimpedance amplifier simulation in 50 ohms environment. Transimpedance is plotted using a Vout/Iinput calculation.



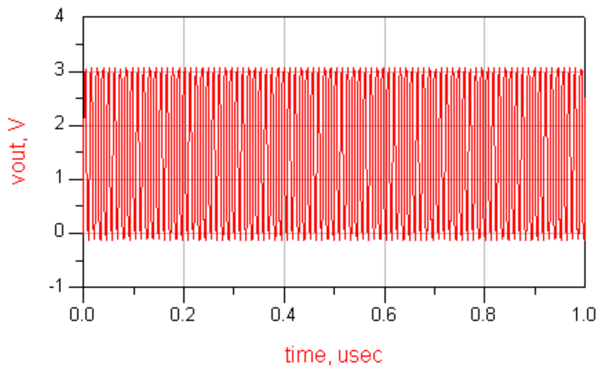
3. "1.25 Gbps CMOS Transimpedance Amplifier Transient Simulation" shows time domain simulation of CMOS Transimpedance Amplifier with input current waveform generated through photo diode model. Input current, output voltage waveform and eyediagram at the output is plotted.
4. "10 Gbps HBT Transimpedance Amplifier Linear Simulation" shows AC simulation of Trans- impedance Amplifier. The input signal is a current waveform and the output is a voltage waveform. The Transimpedance performance of this amplifier is plotted.

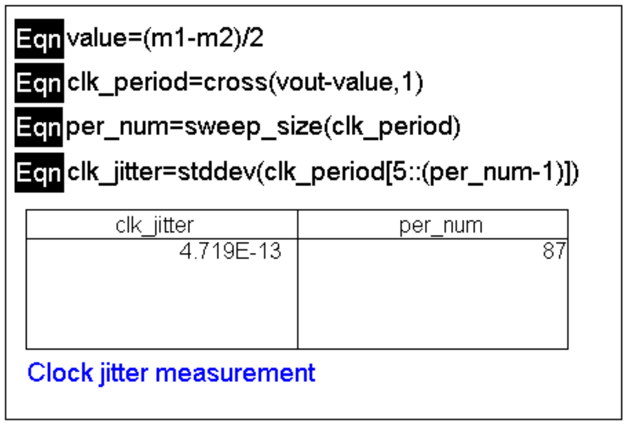


5. "10 Gbps Laser Driver Simulation" shows time domain response of a laser driver circuit. The Laser driver is designed using two stages consisting of Laser driver preamplifier and Travelling wave amplifier to achieve 6 Vpp output. The simulation demonstrates flexibility to define differential source with or without jitter. For comparison, the output eye diagram is shown for both cases.
6. "1 Gbps CMOS VCSEL Laser Driver Circuit Simulation" shows the transient simulation of VCSEL driver with simplified VCSEL model. Current waveform and eye diagram is plotted at the output of VCSEL equivalent Circuit.
7. "Large Signal Loop Gain of Ring Oscillator" demonstrates loop gain measurement using S4P equation component. Harmonic Balance Simulation is performed to calculate Loop Gain and phase response of ring oscillator.



8. "Ring Oscillator Loop Optimization" shows the set up to Optimize Oscillator loop gain and phase response at the required frequency.
9. "Ring Oscillator Initial Guess Simulation" demonstrates the creation of initial guess file using transient simulation.This file can be used as initial guess solution to perform Time Assisted Harmonic Balance Simulation.
10. "Ring Oscillator Time Assisted Harmonic Balance Simulation" shows time assisted harmonic balance simulation of Ring Oscillator. The initial Guess file generated in previous step is used to reduce simulation time and to avoid convergence problems.





Eqn value=(m1-m2)/2

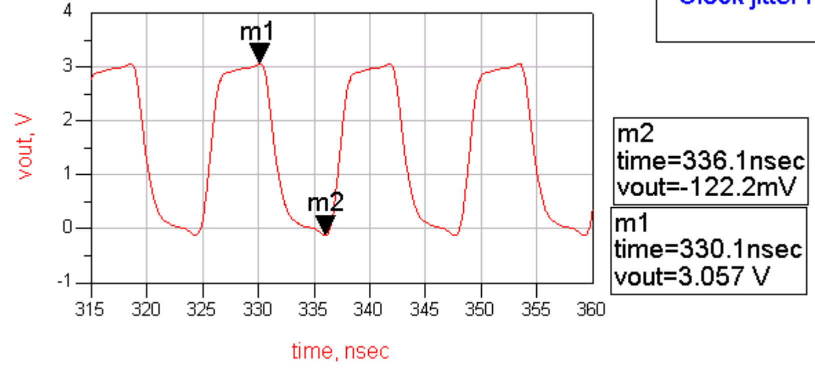
Eqn clk_period=cross(vout-value,1)

Eqn per_num=sweep_size(clk_period)

Eqn clk_jitter=stddev(clk_period[5::(per_num-1)])

clk_jitter	per_num
4.719E-13	87

Clock jitter measurement



Digital Component and Test

Setup

1. "Data buffer", "10 Gbps HBT Clock Buffer", "10 Gbps HBT Divide by 2", "10Gbps HBT Divider Buffer" , "10 Gbps HBT Output Amplifier" , "10 Gbps HBT Latch"and "10 Gbps 2:1 Selector" provides some samples of sub components used in designing Multiplexer and De-Multiplexer circuits.
2. "Test_Latch" demonstrates time domain simulation of High Speed Latch using differential signal source.
3. "2:1 Selector Simulation" shows basic Multiplexer building block simulation.Two differential data sources with different delay parameters are used as input signal. The output waveform demonstrates multiplexing action on two input waveforms.
4. "Clock Distribution Simulation" provides time domain simulation of Clock distribution circuit for 16:1 Multiplexer. The Clock distribution circuit divides a 10 GHz input frequency to generate 5 GHz, 2.5 GHz, 1.25 GHz and 625 MHz clock signal. The data display shows the input and divided output waveforms.
5. "16:1 Multiplexer with Clock Distribution Circuit " demonstrates time domain simulation of 16:1 Multiplexer for 10 GB application. Frequency domain models of Microstrip line, behavioral model for RF System Amplifier and S-Parameter block are connected at the output of Multiplexer to demonstrate time domain simulation capability of frequency domain models.
6. "1:16 De-Multiplexer Simulation With Clock Distribution Circuit" shows time domain simulation of 1:16 De-Multiplexer for 10 Gbps input signal. TheDe-Multiplexer is designed using a modular approach, consisting of 1:2De-Multipexer and clock distribution circuits.

Behavioral Model Examples

Examples of how reduced-order models of amplifiers, mixers, modulators, and demodulators used as building blocks in receivers and transmitters enable fast and accurate system level simulations that cannot be completed at the circuit level.

- *AmplifierP2D_Setup and AmplifierP2D* (examples)
- *AmplifierS2D_Setup and AmplifierS2D* (examples)
- *Data Based Amplifier* (examples)
- *Data Based IQ Demodulator* (examples)
- *Data Based IQ Modulator* (examples)
- *Data Based Load Pull Amplifier* (examples)
- *Data Based Mixer* (examples)
- *Data Based Models for Differentially Fed Components* (examples)
- *Extraction and use of IMT Based System Level Mixers* (examples)
- *VCA_Setup and VCA_Data* (examples)

AmplifierP2D_Setup and AmplifierP2D

Location: $HPEESOF_DIR/examples/BehavioralModels/AmpP2D_wrk

Objective

This example illustrates the use VME amplifier tools for the extraction and use of an MDIF P2D profile defining a narrow-band power amplifier. P2D files contain full 2 port S-parameters of a device under small and large signal conditions. A 2-port noise section is optional. For details on the P2D file format, see *Working with Data Files* (cktsim).

P2D files may be generated from a circuit level design using the AmplifierP2D_Setup component or obtained from a 2-port network analyzer measuring the S-parameters of a hardware prototype. Regardless of the source, the VME data model AmplifierP2D is capable of using a P2D profile for simulating the non-linear behavior of the fundamental or nominal frequency component at the output. No harmonics or intermodulation products are produced by this model.

AmplifierP2D is recommended for the design verification of applications involving narrow-band, arbitrarily nonlinear RF subsystems where full 2-port analysis is required but only the response at the nominal or carrier frequency is of interest.

Details on the VME amplifier pair consisting of AmplifierP2D_Setup and AmplifierP2D are described in Components > Circuit Components > System Models > *System Data Models* (ccsys).

Setup

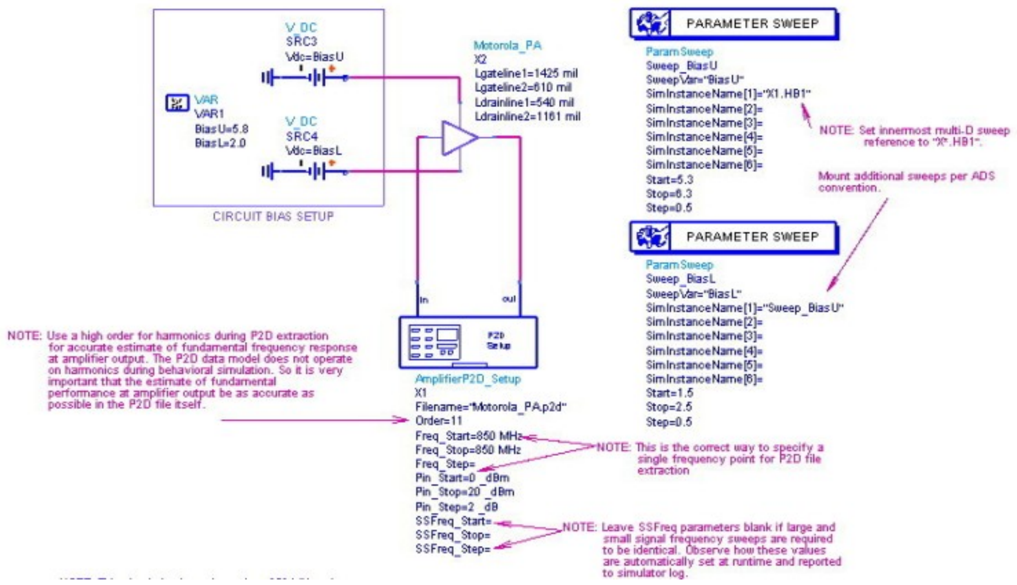
Circuit Level Models
The main circuit level standard used in this example is **Motorola_PA**. This amplifier is ideally biased at 5.8 V at the upper bias port and 2.0 V at the lower bias port. It is effectively a 4-port model. Note that the AmplifierP2D data model is a 2-port model. In the following experiments bias information extracted to the P2D file is used only for data indexing into the appropriate P2D profile by AmplifierP2D. No sophisticated bias and RF power dependent parameters such as DC power drawn from source is modeled by this amplifier. The VCA_Setup and VCA_Data VME pair is recommended for applications that require true bias modeling for PAE compuations.

A pair of system level composite amplifiers are used to imitate filtering effects of a realistic circuit level amplifier for envelope-band distortion tests. **Amp_BPF_Atten** and **BPF_Amp_Atten** are used in **BEH_P2D_CE_filter** and **BEH_P2D_CE_sample** for in-situ extraction and modeling of post- and pre-filtered conditions respectively.

P2D Extraction Process
The AmplifierP2D_Setup extractor component shown in Figure 1 is capable of driving RF input through a circuit level amplifier model and registering the P2D profile in the user-specified filename. The principal simulator within the extractor (see subcircuit) is the P2D controller instance " HB1 ". It is possible to place arbitrary user-defined parametric sweeps outside the AmplifierP2D_Setup component to generate multi-dimensional P2D files. The only requirement creation of a valid multi-dimensional P2D file is that the lowest of the external sweeps point to "Xi.HB1" where "Xi" is the ID of the AmplifierP2D_Setup controller component instance in use. See the design **CKT_P2D_extraction** for further details.

No data display schematic is associated with the extraction process. The P2D file desposited in the data subdirectory is human readable. The curious user is urged to use the Instrument Server to translate the P2D file into a P2D dataset for viewing on a data display schematic. This is a useful exercise in checking the contents of a P2D file.

Figure 1: Performing multi-dimensional P2D extraction


V_DC
SRC3
Vdc=BiasU
VAR
VAR1
BiasU=5.8
BiasL=2.0
V_DC
SRC4
Vdc=BiasL
CIRCUIT BIAS SETUP
Motorola_PA
X2
Lgateline1=1425 mil
Lgateline2=610 mil
Ldrainline1=540 mil
Ldrainline2=1161 mil
PARAMETER SWEEP
ParamSweep
Sweep_BiasU
SweepVar="BiasU"
SimInstanceName[1]="X1.HB1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=5.3
Stop=6.3
Step=0.5
NOTE: Set innermost multi-D sweep reference to "X".HB1".
Mount additional sweeps per ADS convention.
PARAMETER SWEEP
ParamSweep
Sweep_BiasL
SweepVar="BiasL"
SimInstanceName[1]="Sweep_BiasU"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1.5
Stop=2.5
Step=0.5
In
out
P2D Setup
NOTE: Use a high order for harmonics during P2D extraction for accurate estimate of fundamental frequency response at amplifier output. The P2D data model does not operate on harmonics during behavioral simulation. So it is very important that the estimate of fundamental performance at amplifier output be as accurate as possible in the P2D file itself.
AmplifierP2D_Setup
X1
Filename="Motorola_PA.p2d"
Order=11
Freq_Start=850 MHz
Freq_Stop=850 MHz
Freq_Step=
Pin_Start=0 _dBm
Pin_Stop=20 _dBm
Pin_Step=2 _dB
SSFreq_Start=
SSFreq_Stop=
SSFreq_Step=
NOTE: This is the correct way to specify a single frequency point for P2D file extraction
NOTE: Leave SSFreq parameters blank if large and small signal frequency sweeps are required to be identical. Observe how these values are automatically set at runtime and reported to simulator log.

Single Tone Harmonic Balance Analysis

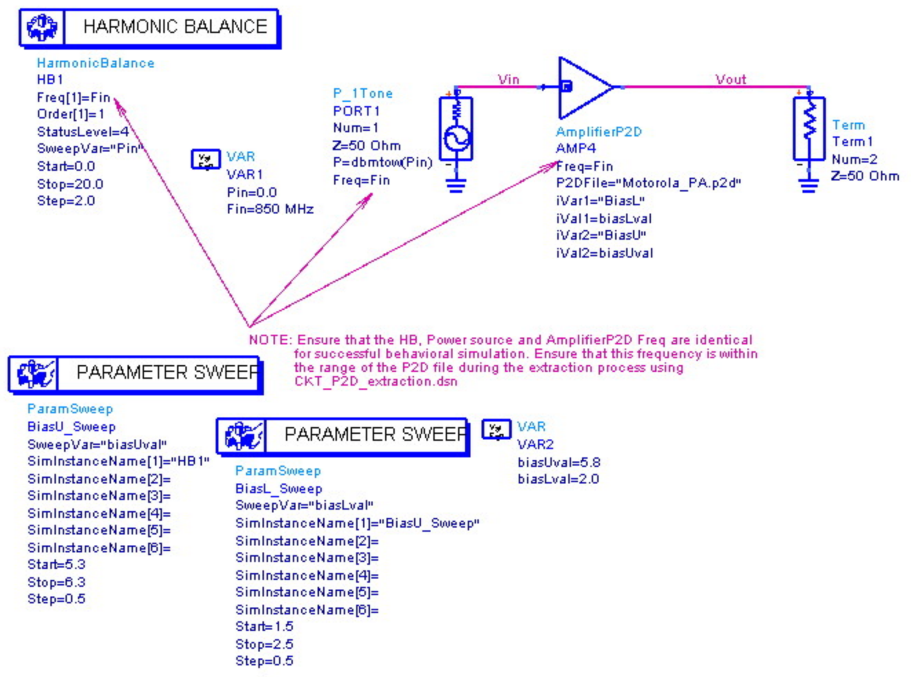
The example design **BEH_P2D_HB_1tone** imitates the simulation environment of the circuit level standard **CKT_HB_1tone** , substituting Motorola_PA with its AmplifierP2D counterpart which reads in the extracted P2D file and uses bias values to index into the appropriate S-parameters.

Note that the *Freq* parameter must be set to the nominal harmonic balance tone.

Note that the multi-dimensional parameters " BiasU " and " BiasL " may be arranged in any order using the parameters *iVar1* and *iVar2* and the corresponding values in *iVal1* and *iVal2* . All elements of the multi-dimensional VAR statements of a P2D file except for " temp " and " freq " must be explicitly mentioned in some *iVarN* variable. Omission of one or more essential multi-dimensional parameters will lead to termination of simulation on the grounds of data ambiguity. Superfluous variable names, if not present in the P2D file but present in a parameter *iVarN* are ignored during simulation.

Note that the AmplifierP2D data model has only 2 ports and therefore cannot model bias dependent DC power and amplifier efficiency properties. The results of the two simulations are presented in **Comparison_P2D_HB_1tone.dds** and shown in Figure 4 below.

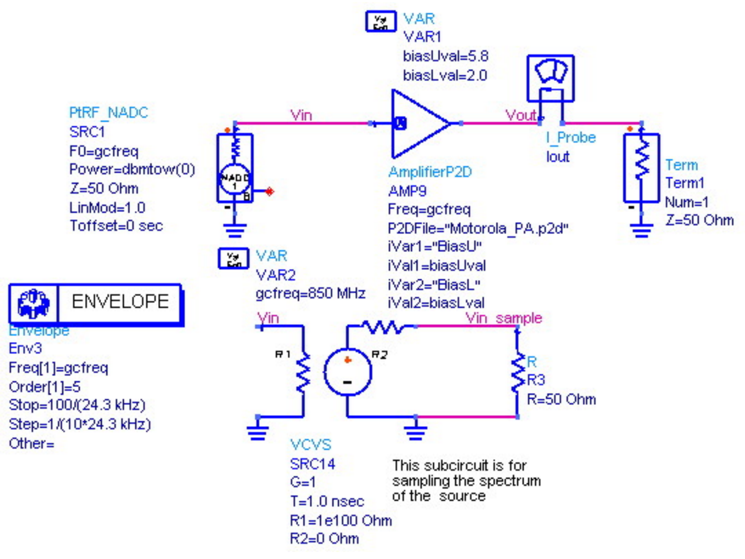
Figure 2: Using AmplifierP2D for single tone HB analysis



Simple Circuit Envelope Analysis
The example design **BEH_P2D_CE_basic** imitates the simulation environment of the circuit level standard **CKT_CE_basic**, substituting Motorola_PA with its AmplifierP2D counterpart which reads in the extracted P2D file and uses bias values to index into the appropriate S-parameters. Note that the *FilteringOption* parameter of the AmplifierP2D instance is left at the default setting of NoFilter indicating that no envelope-band distortion effects will be modeled in this simulation.

The results of the two simulations are presented in **Comparison_P2D_CE_basic.dds** and shown in Figure 5 below.

Figure 3: Using AmplifierP2D for simple CE analysis



Modeling Envelope-band Distortion Effects
AmplifierP2D is capable of estimating the distortion of the fundamental tone due to envelope-band frequencies to imitate a realistic design where a nonlinear amplifier may have prefiltering of a band pass signal prior to nonlinearity or conversely it may have the filtering effects after the nonlinear processing. The P2D data model has settings of PreFilter and PostFilter for the *FilteringOption* parameter to direct the nature of precedence.

Two composite designs are used to illusriate the distortion modeling feature. A composite design consists of separate netweoks that must be simulated in a mutually exclusive fashion.

BEH_P2D_CE_filter allows the user to compare the behavioral responses under the two filtering settings against the circuit level responses. It also contains (deactivated by default) the extraction setups for generating the necessary P2D files. Note that the same P2D file may be used to generate both types of filtered responses.

When either of the active filtering modes are in effect, AmplifierP2D relies on the impluse sample spacing parameter *CEFreqSpacing* to determine the accuracy of the distortion modeling. The trade-off of simulation speed versus accuracy is shown in the example design **BEH_P2D_CE_sample**.

Please note that the data display schematics need to be manually invoked after each of the behavioral simulations. Please follow the sequence of tutorial guidelines in each composite design to suppress network activity and to invoke the data display schematics after the appropriate simulation.

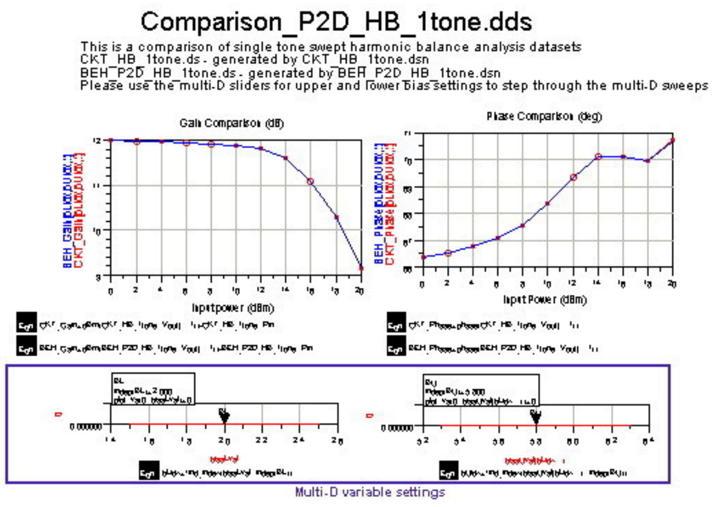
Figures 6 and 7 below show screen shots of the two display schematics.

Analysis

Single Tone Harmonic Balance Analysis

Figure 4 shows the comparison of the circuit and behavioral level fundamental frequency reponses from **Comparison_P2D_HB_1tone.dds**. The sliders allow an interactive assessment of the performance of AmplifierP2D against the circuit level model for various bias conditions. Note that all behavioral responses were generated from a single P2D file.

Figure 4: Comparison of circuit and behavioral model responses to single tone HB analysis



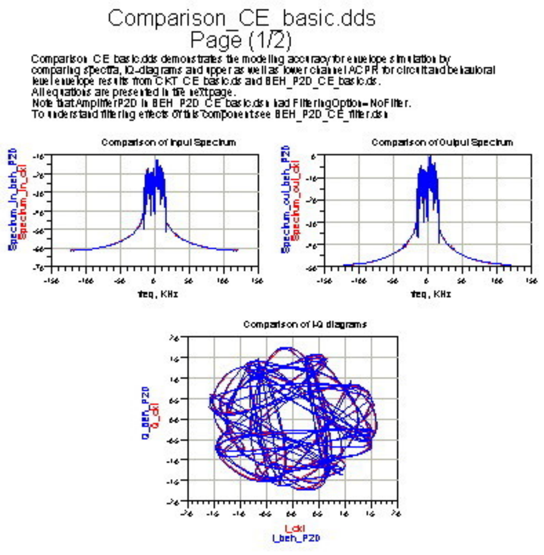
Comparison_P2D_HB_1tone.dds

This is a comparison of single tone swept harmonic balance analysis datasets
CKT_HB_1tone.ds - generated by CKT_HB_1tone.dsn
BEH_P2D_HB_1tone.ds - generated by BEH_P2D_HB_1tone.dsn
Please use the multi-D sliders for upper and lower bias settings to step through the multi-D sweeps

Simple Circuit Envelope Analysis

Figure 5 shows the comparison of the circuit and behavioral level spectral and quad-trajectory reponses from **Comparison_P2D_CE_basic.dds**. The sliders allow an interactive assessment of the performance of AmplifierP2D against the circuit level model for various bias conditions. The input and output spectra and the output I-Q trajectory match up agreeably.

Figure 5: Comparison of circuit and behavioral model responses to simple CE analysis



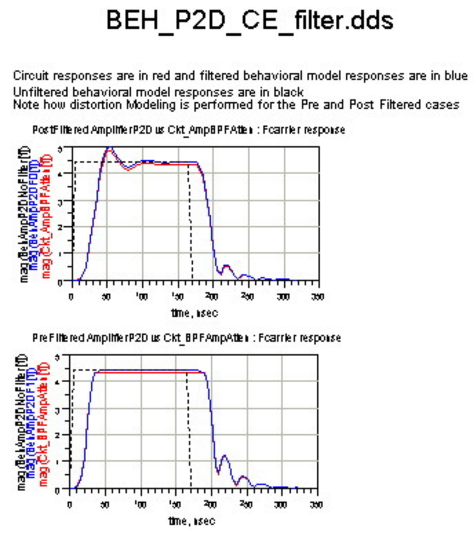
Comparison_CE_basic.dds
Page (1/2)

Comparison_CE_basic.dds demonstrates the modeling accuracy for envelope simulation by comparing spectra, IQ-diagrams and upper as well as lower channel ACPR for circuit and behavioral level envelope results from CKT_CE_basic.ds and BEH_P2D_CE_basic.ds.
All equations are presented in the next page.
Note that AmplifierP2D in BEH_P2D_CE_basic.dsn had FilteringOption=NoFilter.
To understand filtering effects of this component see BEH_P2D_CE_filter.dsn

Envelope Band Distortion Analysis

Figure 6 is a screen shot of **BEH_P2D_CE_filter.dds** which shows how AmplifierP2D conforms to various levels of envelope-band distortion effects on the fundamental tone at the amplifier output. The performance of circuit level system Amp_BPF_Atten is compared with that of AmplifierP2D with *FilteringOption* set to PostFilter whereas that of BPF_Amp_Atten is compared with another instance of AmplifierP2D set to PreFilter . Both plots show time response magnitude of fundamental tone to the pulse input. The response of a third AmplifierP2D, with its *FilteringOption* set to NoFilter is overlaid on both plots to show the relative accuracy achieved by the distortion modeling feature.

Figure 6: Comparison of circuit and behavioral modeling of envelope-band distortion



BEH_P2D_CE_filter.dds

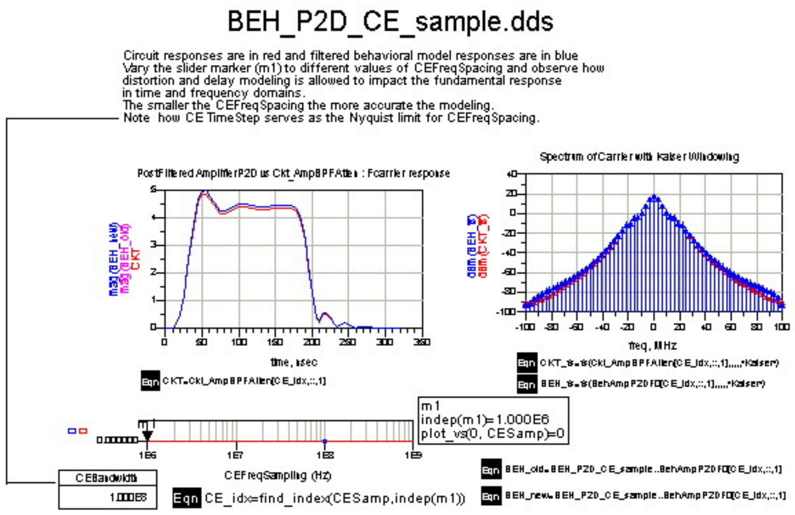
Circuit responses are in red and filtered behavioral model responses are in blue
Unfiltered behavioral model responses are in black
Note how distortion Modeling is performed for the Pre and Post Filtered cases

PostFiltered AmplifierP2D vs Ckt_AmpBPFAtten : Fcarrier response
mag(BehAmpP2DNoFilter[1])
mag(BehAmpP2DFO[1])
mag(Ckt_AmpBPFAtten[1])
time, usec
PreFiltered AmplifierP2D vs Ckt_BPFAmpAtten : Fcarrier response
mag(BehAmpP2DNoFilter[1])
mag(BehAmpP2DF[1])
mag(Ckt_BPFAmpAtten[1])
time, usec

Figure 7 is a screen shot of **BEH_P2D_CE_sample.dds** which shows how varying the impulse sampliging with __ on AmplifierP2D helps reconstruct various levels of accuracy in envelope-band distortion effects. The performance of circuit level system Amp_BPF_Atten is compared with that of AmplifierP2D with *FilteringOption* set to PostFilter . The plot shows time response magnitude of fundamental tone to the pulse input.

Please use the CEsamp slider to study how increasing sample spacing gradually removes all attempts at distortion modeling in the vicinity of the Nyquist rate of *CEFreqSpacing* =0.5* *Freq* . Good distortion modeling occurs when there are more than 1000 samples within the envelope band.

Figure 7: Regulating the amount of envelope-band distortion


BEH_P2D_CE_sample.dds

Circuit responses are in red and filtered behavioral model responses are in blue
Vary the slider marker (m1) to different values of CEFreqSpacing and observe how distortion and delay modeling is allowed to impact the fundamental response in time and frequency domains.
The smaller the CEFreqSpacing the more accurate the modeling.
Note how CE TimeStep serves as the Nyquist limit for CEFreqSpacing.

Eqn CKT=Ckt_AmpBPFAtten[CE_idx,::,1]

Eqn CKT_s=s(Ckt_AmpBPFAtten[CE_idx,::,1],,,,,"Kaiser")

Eqn BEH_s=s(BehAmpP2DFD[CE_idx,::,1],,,,,"Kaiser")

m1
indep(m1)=1.000E6
plot_vs(0, CESamp)=0

Eqn BEH_old=BEH_P2D_CE_sample..BehAmpP2DFD[CE_idx,::,1]

Eqn BEH_new=BEH_P2D_CE_sample..BehAmpP2DFD[CE_idx,::,1]

CEBandwidth
1.000E8

Eqn CE_idx=find_index(CESamp,indep(m1))

PostFiltered AmplifierP2D vs Ckt_AmpBPFAtten : Fcarrier response
mag(BEH_new)
mag(BEH_old)
CKT
time, nsec
Spectrum of Carrier with Kaiser Windowing
dBm(BEH_s)
dBm(CKT_s)
freq, MHz
CEFreqSampling (Hz)

AmplifierS2D_Setup and AmplifierS2D

Location: $HPEESOF_DIR/examples/BehavioralModels/AmpS2D_wrk

Objective

This example illustrates the use VME amplifier tools for the extraction and use of an MDIF S2D profile defining a narrow-band power amplifier. S2D files contain full 2 port S-parameters of a device under small signal conditions and forward transmission or gain compression information under large signal conditions. A 2-port noise section is optional. For details on the S2D file format, see *Working with Data Files* (cktsim).

S2D files may be generated from a circuit level design using the AmplifierS2D_Setup component.

The VME data model AmplifierS2D is capable of using a S2D profile for simulating the non-linear behavior of the fundamental frequency and upto 9th order odd harmonics and intermodulation products at the output.

AmplifierS2D is recommended for the design verification of applications involving general purpose, arbitrarily nonlinear RF subsystems where only forward transmission modeling is important and where even-order harmonics and intermods are not of interest.

Details on the VME amplifier pair consisting of AmplifierS2D_Setup and AmplifierS2D are described in Components > Circuit Components > System Models > System Data Models.

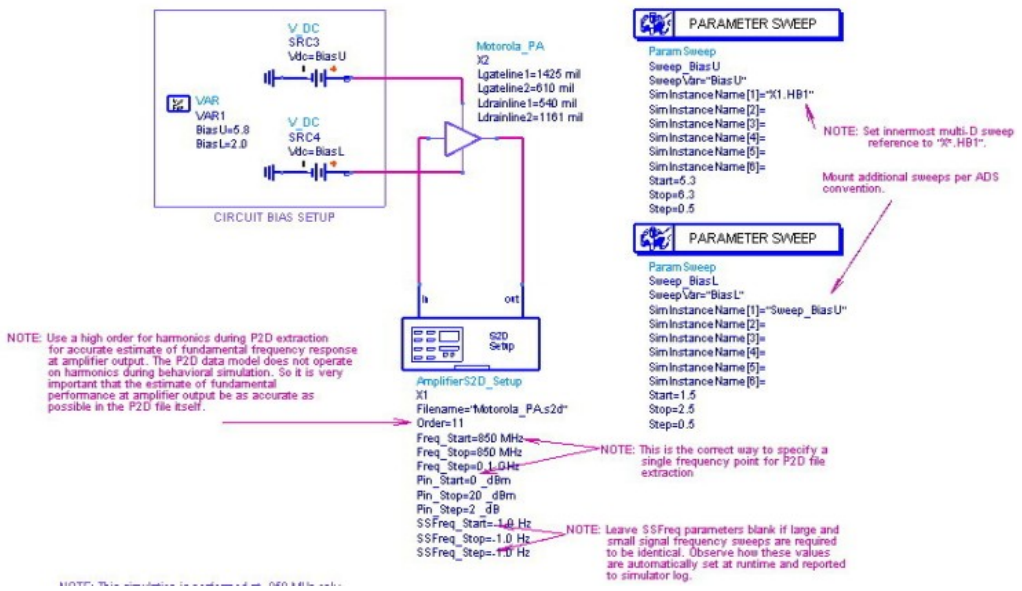
Setup

Circuit Level Model
The main circuit level standard used in this example is **Motorola_PA**. This amplifier is ideally biased at 5.8 V at the upper bias port and 2.0 V at the lower bias port. It is effectively a 4-port model. Note that the AmplifierS2D data model is a 2-port model. In the following experiments bias information extracted to the S2D file is used only for data indexing into the appropriate S2D profile by AmplifierS2D. No sophisticated bias and RF power dependent parameters such as DC power drawn from source is modeled by this amplifier. The VCA_Setup and VCA_Data VME pair is recommended for applications that require true bias modeling for PAE compuations.

S2D Extraction Process
The AmplifierS2D_Setup extractor component shown in Figure 1 is capable of driving RF input through a circuit level amplifier model and registering the S2D profile in the user-specified filename. The principal simulator within the extractor (see subcircuit) is the P2D controller instance " HB1 ". It is possible to place arbitrary user-defined parametric sweeps outside the AmplifierS2D_Setup component to generate multi-dimensional P2D files. The only requirement creation of a valid multi-dimensional S2D file is that the lowest of the external sweeps point to " Xi.HB1 " where " Xi " is the ID of the AmplifierS2D_Setup controller component instance in use. See the design **CKT_S2D_extraction** for further details.

No data display schematic is associated with the extraction process. The S2D file despositied in the data subdirectory is human readable. The curious user is urged to use the Instrument Server to translate the S2D file into a S2D dataset for viewing on a data display schematic. This is a useful exercise in checking the contents of a S2D file.

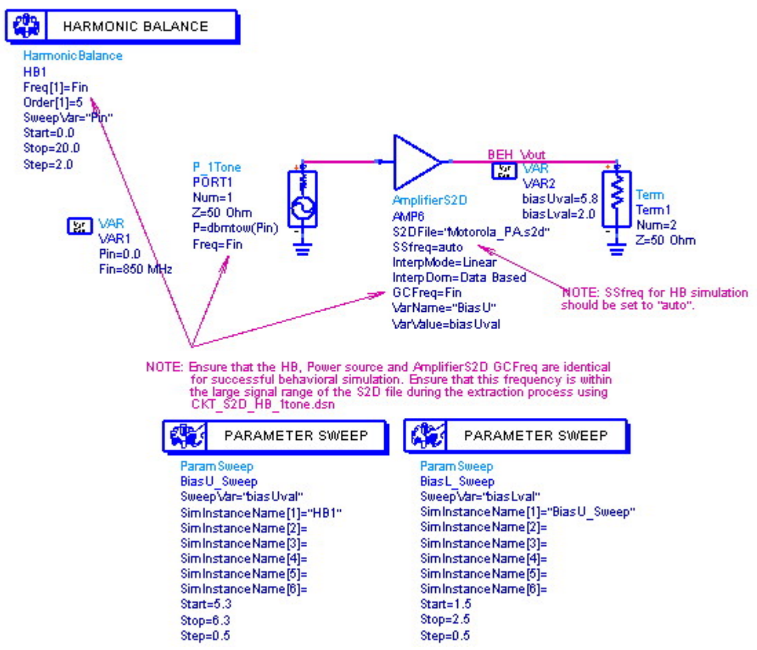
Figure 1: Performing multi-dimensional S2D extraction


V_DC
SRC3
Vdc=BiasU
VAR
VAR1
BiasU=5.8
BiasL=2.0
V_DC
SRC4
Vdc=BiasL
CIRCUIT BIAS SETUP
Motorola_PA
X2
Lgateline1=1425 mil
Lgateline2=610 mil
Ldrainline1=540 mil
Ldrainline2=1161 mil
PARAMETER SWEEP
ParamSweep
Sweep_BiasU
SweepVar="BiasU"
SimInstanceName[1]="X1.HB1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=5.3
Stop=6.3
Step=0.5
NOTE: Set innermost multi-D sweep reference to "X*.HB1".
Mount additional sweeps per ADS convention.
PARAMETER SWEEP
ParamSweep
Sweep_BiasL
SweepVar="BiasL"
SimInstanceName[1]="Sweep_BiasU"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1.5
Stop=2.5
Step=0.5
S2D Setup
NOTE: Use a high order for harmonics during P2D extraction for accurate estimate of fundamental frequency response at amplifier output. The P2D data model does not operate on harmonics during behavioral simulation. So it is very important that the estimate of fundamental performance at amplifier output be as accurate as possible in the P2D file itself.
AmplifierS2D_Setup
X1
Filename="Motorola_PA.s2d"
Order=11
Freq_Start=850 MHz
Freq_Stop=850 MHz
Freq_Step=0.1 GHz
Pin_Start=0 dBm
Pin_Stop=20 dBm
Pin_Step=2 dB
SSFreq_Start=-1.0 Hz
SSFreq_Stop=-1.0 Hz
SSFreq_Step=-1.0 Hz
NOTE: This is the correct way to specify a single frequency point for P2D file extraction
NOTE: Leave SSFreq parameters blank if large and small signal frequency sweeps are required to be identical. Observe how these values are automatically set at runtime and reported to simulator log.

Single Tone Harmonic Balance Analysis

The example design **BEH_S2D_HB_1tone** uses the same single tone HB analysis environment to compare the performance of Motorola_PA and AmplifierS2D.
Note that the *GCFreq* parameter on the behavioral model must be set to the nominal harmonic balance tone to read the correct set of compression data from the S2D file.
Note that the multi-dimensional parameters " BiasU " and " BiasL " may be arranged in any order using the multi-dimensional parameters *iVar1* and *iVar2* with corresponding values in *iVal1* and *iVal2* . All elements of the multi-dimensional VAR statements of an S2D file except for " temp " and " freq " must be explicitly mentioned using an *iVarN* parameter. Omission of one or more essential multi-dimensional parameters will lead to termination of simulation on the grounds of data ambiguity. Superfluous variable names, if not present in the S2D file but present in the *iVarN* list, are ignored during simulation.
Note that the AmplifierS2D data model has only 2 ports and therefore cannot model bias dependent DC power and amplifier efficiency properties. The result of the joint simulation is presented in **BEH_S2D_HB_1tone.dds** and shown in Figure 3 below.

Figure 2: Using AmplifierS2D for single tone HB analysis



Dual Tone Harmonic Balance Analysis
The example design **BEH_S2D_HB_2tone** uses the same dual tone HB analysis environment to compare the performance of Motorola_PA and AmplifierS2D. The objective is to study how well AmplifierS2D reconstructs odd-order harmonics and intermodulation products compared to a circuit level model.
Note that the *GCFreq* parameter on the behavioral model must be set to the one of the two harmonic balance tones to read the correct set of compression data from the S2D file.

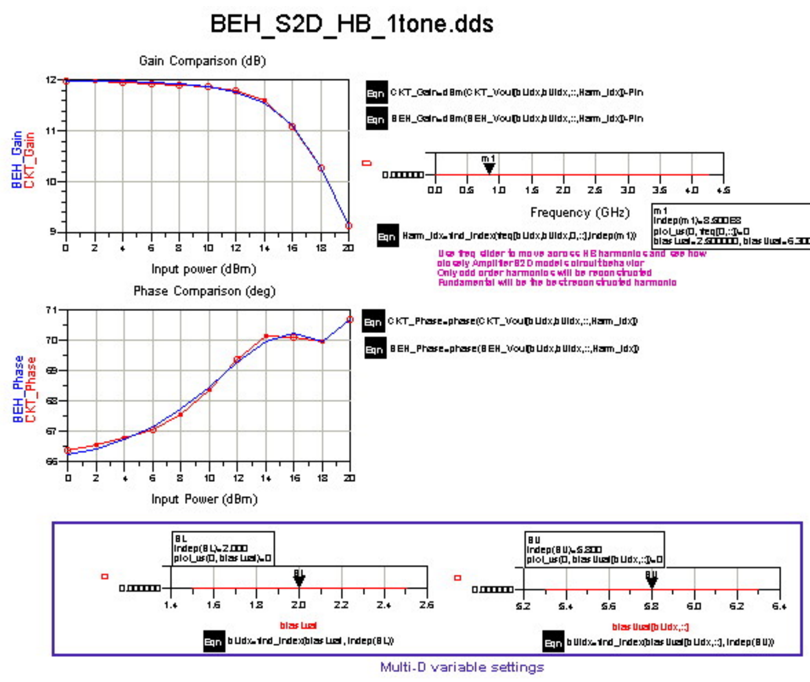
The result of the joint simulation is presented in **BEH_S2D_HB_2tone.dds** and shown in Figure 4 below.

Simple Circuit Envelope Analysis
The example design **BEH_S2D_CE_basic** uses the same circuit envelope analysis environment to compare the performance of Motorola_PA and AmplifierS2D.
The result of the joint simulation is presented in **Comparison_S2D_CE_basic.dds** and shown in Figure 5 below.

Analysis

Single Tone Harmonic Balance Analysis
Figure 3 shows the comparison of the circuit and behavioral level fundamental frequency reponses from **BEH_S2D_HB_1tone.dds**. The sliders allow an interactive assessment of the performance of AmplifierS2D against the circuit level model for various bias conditions. Note that all behavioral responses were generated from a single S2D file.

Figure 3: Comparison of circuit and behavioral model responses to single tone HB analysis



Eqn CKT_Gain=dBm(CKT_Vout[bLIdx,bUIdx,::,Harm_Idx])-Pin
Eqn BEH_Gain=dBm(BEH_Vout[bLIdx,bUIdx,::,Harm_Idx])-Pin
Eqn Harm_Idx=find_index(freq[bLIdx,bUIdx,0,::],indep(m1))

m1
indep(m1)=8.500E8
plot_vs(0, freq[0,::])=0
bias Lval=2.500000, bias Uval=6.300000

Use freq slider to move across HB harmonics and see how closely AmplifierS2D models circuit behavior
Only odd order harmonics will be reconstructed
Fundamental will be the best reconstructed harmonic

Eqn CKT_Phase=phase(CKT_Vout[bLIdx,bUIdx,::,Harm_Idx])
Eqn BEH_Phase=phase(BEH_Vout[bLIdx,bUIdx,::,Harm_Idx])

BL
indep(BL)=2.000
plot_vs(0, bias Lval)=0

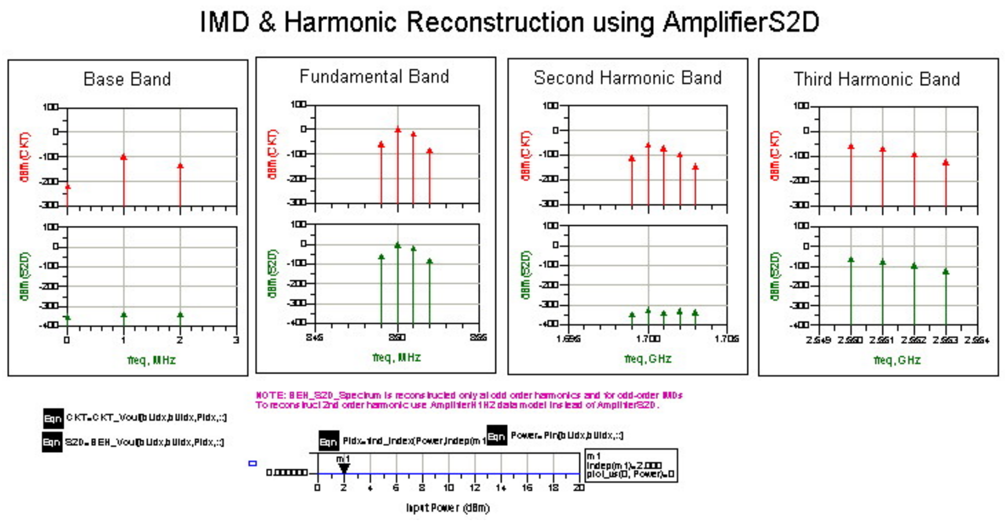
BU
indep(BU)=5.800
plot_vs(0, bias Uval[bLIdx,::])=0

Eqn bLIdx=find_index(bias Lval, indep(BL))
Eqn bUIdx=find_index(bias Uval[bLIdx,::], indep(BU))

Dual Tone Harmonic Balance Analysis
Figures 4a and 4b show the comparison of the circuit and behavioral level fundamental frequency reponses from pages of **BEH_S2D_HB_2tone.dds**. The sliders allow an interactive assessment of the performance of AmplifierS2D against the circuit level model for various bias conditions. Note that all behavioral responses were generated from a single S2D file.

Figure4A shows solid reconstructions of the spectral clusters around the first and third harmonic tones. Within each cluster the third and fifth order intermodulation products are well reconstructed by the behavioral model. In contrast the second order band and baseband signals are poorly reconstructed. Please use the VME pair AmpH1H2_Setup and AmplifierH1H2 for applications dependent on estimation of second order harmonics.

Figure 4a: Spectral responses to dual tone HB analysis



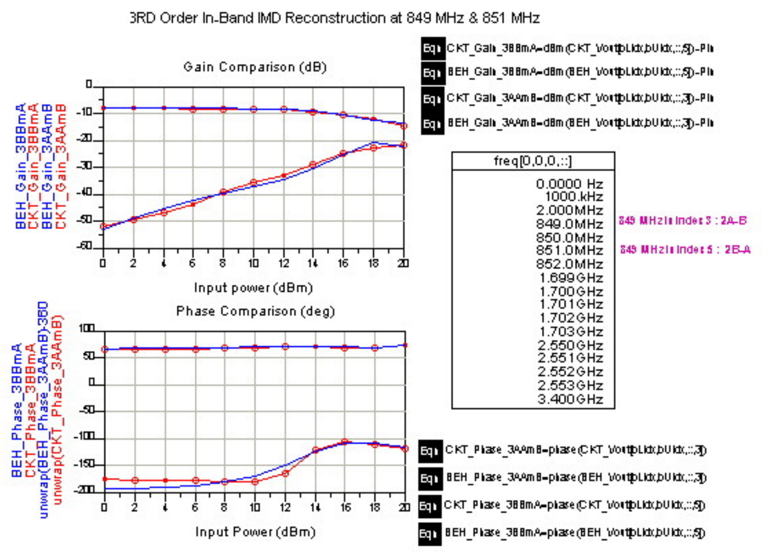
NOTE: BEH_S2D_Spectrum is reconstructed only at odd order harmonics and for odd-order IMDs
To reconstruct 2nd order harmonic use AmplifierH1H2 data model instead of AmplifierS2D.

Eqn CKT=CKT_Vout[bUIdx,bUIdx,PIdx,::]
Eqn S2D=BEH_Vout[bUIdx,bUIdx,PIdx,::]
Eqn PIdx=find_index(Power,indep(m1
Eqn Power=Pin[bUIdx,bUIdx,::]

m1
indep(m1)=2.000
plot_vs(0, Power)=0

Figure 4b shows the reconstruction of the third order intermodulation products ($2*F_1-F_2$) and ($2*F_2-F_1$) within the envelope of the fundamental band against input power. Both dB and phase plots show reasonable reconstruction of these two tones at the output. Please note that these tones are generated using a polynomial approximation within the AmplifierS2D model, hence some ringing effects are observed near saturation.

Figure 4b: Reconstruction of 3rd order IMDs during dual tone HB analysis



Eqn CKT_Gain_3BBmA=dBm(CKT_Vout[bUIdx,bUIdx,::,5])-Pin
Eqn BEH_Gain_3BBmA=dBm(BEH_Vout[bUIdx,bUIdx,::,5])-Pin
Eqn CKT_Gain_3AAmB=dBm(CKT_Vout[bUIdx,bUIdx,::,3])-Pin
Eqn BEH_Gain_3AAmB=dBm(BEH_Vout[bUIdx,bUIdx,::,3])-Pin

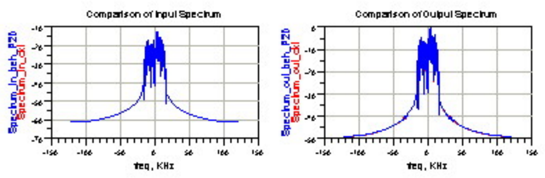
freq[0,0,0,::]
0.0000 Hz
1000.kHz
2.000MHz
849.0MHz
850.0MHz
851.0MHz
852.0MHz
1.699GHz
1.700GHz
1.701GHz
1.702GHz
1.703GHz
2.550GHz
2.551GHz
2.552GHz
2.553GHz
3.400GHz

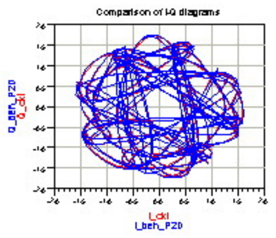
849 MHz is index 3 : 2A-B
849 MHz is index 5 : 2B-A

Eqn CKT_Phase_3AAmB=phase(CKT_Vout[bUIdx,bUIdx,::,3])
Eqn BEH_Phase_3AAmB=phase(BEH_Vout[bUIdx,bUIdx,::,3])
Eqn CKT_Phase_3BBmA=phase(CKT_Vout[bUIdx,bUIdx,::,5])
Eqn BEH_Phase_3BBmA=phase(BEH_Vout[bUIdx,bUIdx,::,5])

Simple Circuit Envelope Analysis
Figure 5 shows the comparison of the circuit and behavioral level spectral and quad-trajectory reponses from **BEH_S2D_CE_basic.dds**. The sliders allow an interactive assessment of the performance of AmplifierS2D against the circuit level model for various bias conditions. The input and output spectra and the output I-Q trajectory match up agreeably.

Figure 5: Comparison of circuit and behavioral model responses to simple CE analysis





Notes

Other system level amplifiers that can use the gain compression information of a basic (non-multidimensional) S2D files are the SML models Amplifier2 and AmpSingleCarrier. Note that neither of these models can read small signal parameters or noise parameters from an S2D file. They are not compatible with multi-dimensional S2D files either. AmplifierS2D is the only S2D reliant model that can support multi-dimensional modeling without the need for external parameters.

Data Based Amplifier

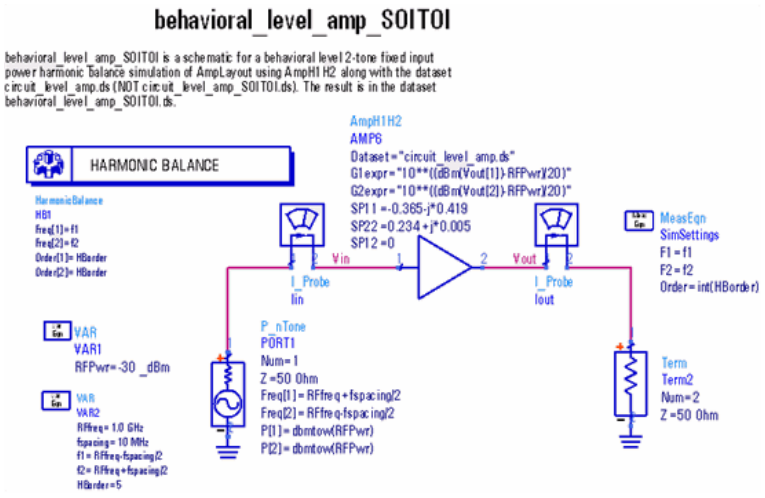
Location: $HPEESOF_DIR/examples/BehavioralModels/AmpH1H2_wrk

Objective

Behavioral models are reduced-order models of circuit level devices such as amplifiers, mixers, modulators, and demodulators. After initial extraction of data for the circuit level devices and subsequent generation of their behavioral models, such models can be used as building blocks for receivers and transmitters and allow a fast but accurate system level simulation that cannot be completed at the circuit level.The AmpH1H2 workspace illustrates the use of the AmpH1H2 component in ADS. This component is part of the ADS behavioral model suite found under the System - Data Models palette.

Setup

1. "AmpLayout" is a schematic as well as a layout for the amplifier of interest. For details about this amplifier, please see *Design of a 1GHz Low Noise Amplifier* (examples).
2. "circuit_level_amp" is a schematic for a circuit level 1-tone swept input power harmonic balance simulation of "AmpLayout" using AmpH1H2_Setup. The result is in the dataset "circuit_level_amp.ds."
3. "behavioral_level_amp" is a schematic for a behavioral level 1-tone swept input power harmonic balance simulation of "AmpLayout" using AmpH1H2 along with the dataset "circuit_level_amp.ds." The result is in the dataset "behavioral_level_amp.ds."
4. "circuit_level_amp_SOITOI" is a schematic for a circuit level 2-tone fixed input power harmonic balance simulation of "AmpLayout." The result is in the dataset "circuit_level_amp_SOITOI.ds."
5. "behavioral_level_amp_SOITOI" is a schematic for a behavioral level 2-tone fixed input power harmonic balance simulation of "AmpLayout" using AmpH1H2 along with the dataset "circuit_level_amp.ds" (NOT "circuit_level_amp_SOITOI.ds"). The result is in the dataset "behavioral_level_amp_SOITOI.ds."



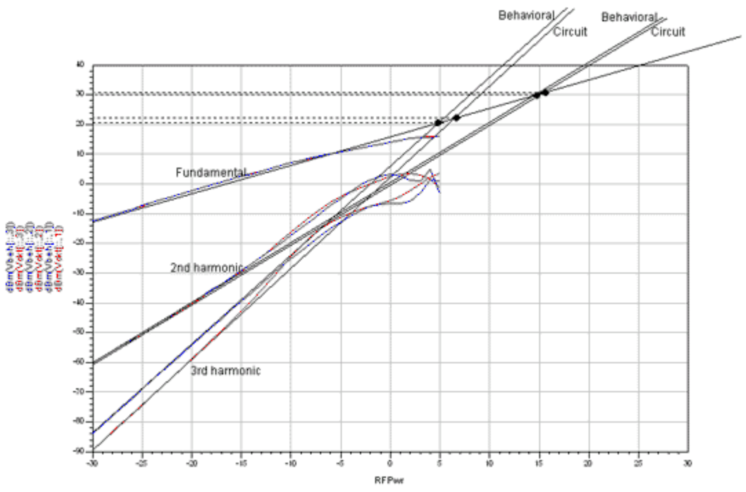
behavioral_level_amp_SOITOI

behavioral_level_amp_SOITOI is a schematic for a behavioral level 2-tone fixed input power harmonic balance simulation of AmpLayout using AmpH1H2 along with the dataset circuit_level_amp.ds (NOT circuit_level_amp_SOITOI.ds). The result is in the dataset behavioral_level_amp_SOITOI.ds.

```
HARMONIC BALANCE
HarmonicBalance
HB1
Freq[1]=f1
Freq[2]=f2
Order[1]= HBorder
Order[2]= HBorder

VAR
VAR1
RFPwr=-30 _dBm

VAR
VAR2
RFfreq= 1.0 GHz
fspacing= 10 MHz
f1= RFfreq-fspacing/2
f2= RFfreq+fspacing/2
HBorder=5

AmpH1H2
AMP6
Dataset="circuit_level_amp.ds"
G1expr="10**((dBm(Vout[1])-RFPwr)/20)"
G2expr="10**((dBm(Vout[2])-RFPwr)/20)"
SP11 =-0.365-j*0.419
SP22 =0.234+j*0.005
SP12 =0

I_Probe
Iin

I_Probe
Iout

P_nTone
PORT1
Num=1
Z =50 Ohm
Freq[1]=RFfreq+fspacing/2
Freq[2]=RFfreq-fspacing/2
P[1]=dbmtow(RFPwr)
P[2]=dbmtow(RFPwr)

MeasEqn
SimSettings
F1=f1
F2=f2
Order=int(HBorder)

Term
Term2
Num=2
Z =50 Ohm
```

Analysis

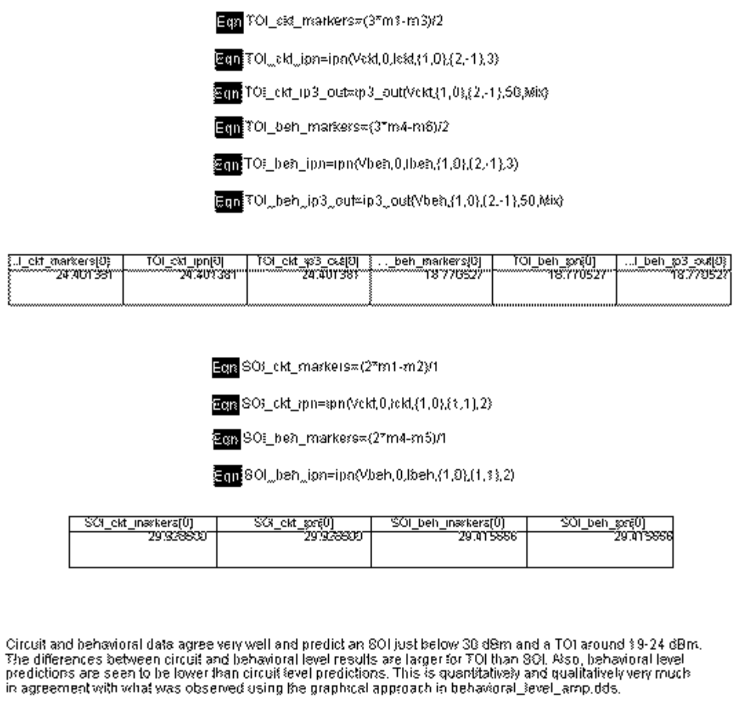
Figure 1: Graphical SOI and TOI determination



Tangents for 2nd and 3rd harmonics of circuit and behavioral data intersect the tangent for the fundamental (1st harmonic) at points where circuit and behavioral SOI and TOI values can be determined as the output power.

Circuit and behavioral data agree very well and predict an SOI around 30 dBm and a TOI of just over 20 dBm. The differences between circuit and behavioral level results are larger for TOI than SOI. Also, behavioral level predictions are seen to be lower than circuit level predictions.

Figure 2: Numerical SOI and TOI determination



```
Eqn TOI_ckt_markers=(3*m1-m3)/2
Eqn TOI_ckt_ipn=ipn(Vckt,0,Ickt,{1,0},{2,-1},3)
Eqn TOI_ckt_ip3_out=ip3_out(Vckt,{1,0},{2,-1},50,Mix)
Eqn TOI_beh_markers=(3*m4-m6)/2
Eqn TOI_beh_ipn=ipn(Vbeh,0,Ibeh,{1,0},{2,-1},3)
Eqn TOI_beh_ip3_out=ip3_out(Vbeh,{1,0},{2,-1},50,Mix)
```

...I_ckt_markers[0]	TOI_ckt_ipn[0]	TOI_ckt_ip3_out[0]	..._beh_markers[0]	TOI_beh_ipn[0]	...I_beh_ip3_out[0]
24.401381	24.401381	24.401381	18.770527	18.770527	18.770527

```
Eqn SOI_ckt_markers=(2*m1-m2)/1
Eqn SOI_ckt_ipn=ipn(Vckt,0,Ickt,{1,0},{1,1},2)
Eqn SOI_beh_markers=(2*m4-m5)/1
Eqn SOI_beh_ipn=ipn(Vbeh,0,Ibeh,{1,0},{1,1},2)
```

SOI_ckt_markers[0]	SOI_ckt_ipn[0]	SOI_beh_markers[0]	SOI_beh_ipn[0]
29.928830	29.928830	29.415556	29.415556

Circuit and behavioral data agree very well and predict an SOI just below 30 dBm and a TOI around 19-24 dBm. The differences between circuit and behavioral level results are larger for TOI than SOI. Also, behavioral level predictions are seen to be lower than circuit level predictions. This is quantitatively and qualitatively very much in agreement with what was observed using the graphical approach in behavioral_level_amp.dds.

Data Based IQ Demodulator

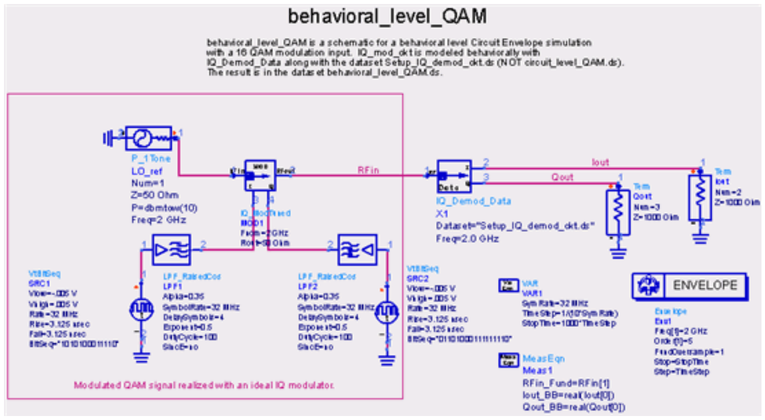
Location: $HPEESOF_DIR/examples/BehavioralModels/IQ_Demod_wrk

Objective

Behavioral models are reduced-order models of circuit level devices such as amplifiers, mixers, modulators, and demodulators. After initial extraction of data for the circuit level devices and subsequent generation of their behavioral models, such models can be used as building blocks for receivers and transmitters and allow a fast but accurate system level simulation that cannot be completed at the circuit level.The IQ_Demod workspace illustrates the use of the IQ_Demod_Setup and IQ_Demod_Data components in ADS. These components are part of the ADS behavioral model suite found under the System - Data Models palette.

Setup

1. "IQ_demod_ckt" is a schematic for the demodulator of interest. This demodulator follows the topology of the system-level demodulator constructed in "examples/RFIC/Cosim_lab_wrk". The modulator is constructed of the sub-circuits "Mixer_GilCel", "phase_shift", and "Wilkinson".
2. "Setup_IQ_demod_ckt" is a for a behavioral model extraction of "IQ_demod_ckt" using IQ_Demod_Setup. The result is in the dataset "Setup_IQ_demod_ckt.ds".
3. "circuit_level_ramp" is a schematic for a circuit level Circuit Envelope simulation of "IQ_demod_ckt" with I_Q waveforms that have time-varying amplitude. The result is in the dataset "circuit_level_ramp.ds".
4. "behavioral_level_ramp" is a schematic for a behavioral level Circuit Envelope simulation with I_Q waveforms that have time-varying amplitude. "IQ_demod_ckt" is modeled behaviorally with IQ_Demod_Data along with the dataset "Setup_IQ_demod_ckt.ds". The result is in the dataset "behavioral_level_ramp.ds".
5. "circuit_level_QAM" is a schematic for a circuit level Circuit Envelope simulation of "IQ_demod_ckt" with I_Q waveforms for a 16 QAM modulation. The result is in the dataset "circuit_level_QAM.ds".
6. "behavioral_level_QAM" is a schematic for a behavioral level Circuit Envelope simulation of with I_Q waveforms for a 16 QAM modulation. "IQ_demod_ckt" is modeled behaviorally with IQ_Demod_Data along with the dataset "Setup_IQ_demod_ckt.ds". The result is in the dataset "behavioral_level_QAM.ds".

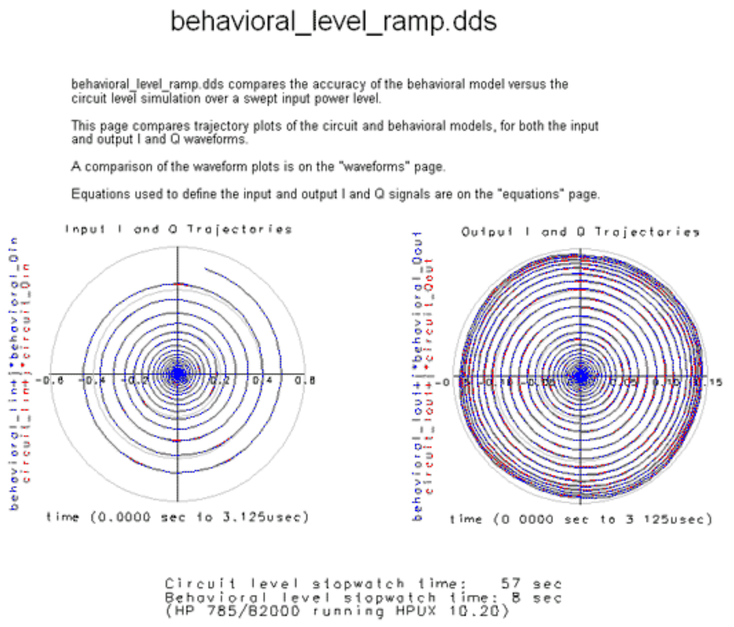


behavioral_level_QAM

behavioral_level_QAM is a schematic for a behavioral level Circuit Envelope simulation with a 16 QAM modulation input. IQ_mod_ckt is modeled behaviorally with IQ_Demod_Data along with the dataset Setup_IQ_demod_ckt.ds (NOT circuit_level_QAM.ds). The result is in the dataset behavioral_level_QAM.ds.

Analysis

Figure 1: Circuit and behavioral level results for I_Q waveforms with time varying amplitude



behavioral_level_ramp.dds

behavioral_level_ramp.dds compares the accuracy of the behavioral model versus the circuit level simulation over a swept input power level.

This page compares trajectory plots of the circuit and behavioral models, for both the input and output I and Q waveforms.

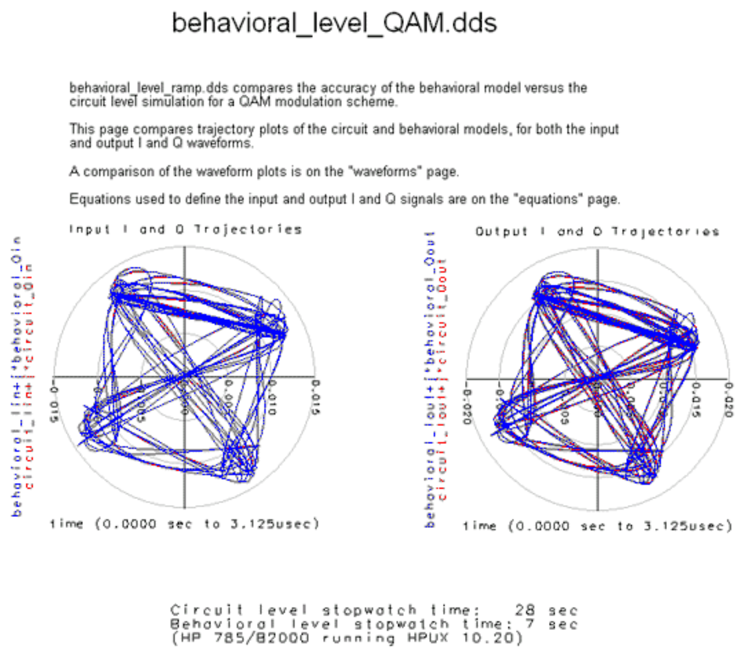
A comparison of the waveform plots is on the "waveforms" page.

Equations used to define the input and output I and Q signals are on the "equations" page.

Input I and Q Trajectories
behavioral_Iin+j*behavioral_Qin
circuit_Iin+j*circuit_Qin
-0.6
-0.4
-0.2
0.2
0.4
0.8
time (0.0000 sec to 3.125usec)
Output I and Q Trajectories
behavioral_Iout+j*behavioral_Qout
circuit_Iout+j*circuit_Qout
-0.15
0.15
time (0.0000 sec to 3.125usec)

Circuit level stopwatch time: 57 sec
Behavioral level stopwatch time: 8 sec
(HP 785/B2000 running HPUX 10.20)

Figure 2: Circuit and behavioral level results for 16 QAM modulation



behavioral_level_QAM.dds

behavioral_level_ramp.dds compares the accuracy of the behavioral model versus the circuit level simulation for a QAM modulation scheme.

This page compares trajectory plots of the circuit and behavioral models, for both the input and output I and Q waveforms.

A comparison of the waveform plots is on the "waveforms" page.

Equations used to define the input and output I and Q signals are on the "equations" page.

Input I and Q Trajectories
behavioral_Iin+j*behavioral_Qin
circuit_Iin+j*circuit_Qin
-0.015
-0.005
0.000
0.005
0.010
0.015
time (0.0000 sec to 3.125usec)
Output I and Q Trajectories
behavioral_Iout+j*behavioral_Qout
circuit_Iout+j*circuit_Qout
-0.020
-0.010
-0.005
0.000
0.005
0.010
0.015
0.020
time (0.0000 sec to 3.125usec)

```
Circuit level stopwatch time:    28 sec
Behavioral level stopwatch time: 7 sec
(HP 785/B2000 running HPUX 10.20)
```

Notes

- In Figure 1, a comparison of circuit and behavioral level results is given for Circuit Envelope simulation. Both input and output waveforms are modeled accurately.
- In Figure 2, a comparison of circuit and behavioral level results is given for Circuit Envelope simulation. Strong agreement is shown for both the input and output IQ-trajectories.

Data Based IQ Modulator

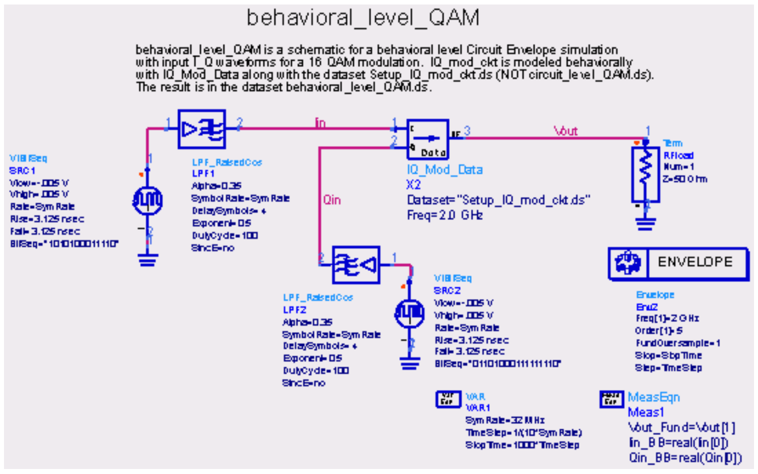
Location: $HPEESOF_DIR/examples/BehavioralModels/IQ_Mod_wrk

Objective

Behavioral models are reduced-order models of circuit level devices such as amplifiers, mixers, modulators, and demodulators. After initial extraction of data for the circuit level devices and subsequent generation of their behavioral models, such models can be used as building blocks for receivers and transmitters and allow a fast but accurate system level simulation that cannot be completed at the circuit level.The IQ_Mod workspace illustrates the use of the IQ_Mod_Setup and IQ_Mod_Data components in ADS. These components are part of the ADS behavioral model suite found under the System - Data Models palette.

Setup

1. "IQ_mod_ckt" is a schematic for the modulator of interest. This modulator is a modification of the circuit-level modulator constructed in "examples/RFIC/Cosim_lab_wrk." The biggest difference is the removal of the amplifier ["PA_2stg"]. The modulator is constructed of the sub-circuits "Mixer_GilCel", "phase_shift", and "Wilkinson".
2. "Setup_IQ_mod_ckt" is a for a behavioral model extraction of "IQ_mod_ckt" using IQ_Mod_Setup. The result is in the dataset "Setup_IQ_mod_ckt.ds".
3. "circuit_level_ramp" is a schematic for a circuit level Circuit Envelope simulation of "IQ_mod_ckt" with I_Q waveforms that have time-varying amplitude. The result is in the dataset "circuit_level_ramp.ds".
4. "behavioral_level_ramp" is a schematic for a behavioral level Circuit Envelope simulation with I_Q waveforms that have time-varying amplitude. "IQ_mod_ckt" is modeled behaviorally with IQ_Mod_Data along with the dataset "Setup_IQ_mod_ckt.ds". The result is in the dataset "behavioral_level_ramp.ds".
5. "circuit_level_QAM" is a schematic for a circuit level Circuit Envelope simulation of "IQ_mod_ckt" with I_Q waveforms for a 16 QAM modulation. The result is in the dataset "circuit_level_QAM.ds".
6. "behavioral_level_QAM" is a schematic for a behavioral level Circuit Envelope simulation of with I_Q waveforms for a 16 QAM modulation. "IQ_mod_ckt" is modeled behaviorally with IQ_Mod_Data along with the dataset "Setup_IQ_mod_ckt.ds". The result is in the dataset "behavioral_level_QAM.ds".

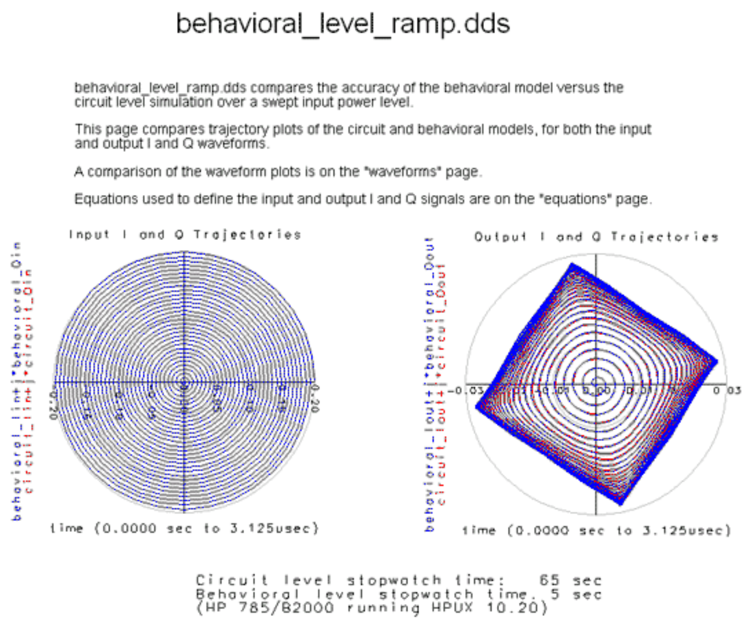


behavioral_level_QAM

behavioral_level_QAM is a schematic for a behavioral level Circuit Envelope simulation with input I_Q waveforms for a 16 QAM modulation. IQ_mod_ckt is modeled behaviorally with IQ_Mod_Data along with the dataset Setup_IQ_mod_ckt.ds (NOT circuit_level_QAM.ds). The result is in the dataset behavioral_level_QAM.ds.

Analysis

Figure 1:Circuit and behavioral level results for I_Q waveforms with time varying amplitude



behavioral_level_ramp.dds

behavioral_level_ramp.dds compares the accuracy of the behavioral model versus the circuit level simulation over a swept input power level.

This page compares trajectory plots of the circuit and behavioral models, for both the input and output I and Q waveforms.

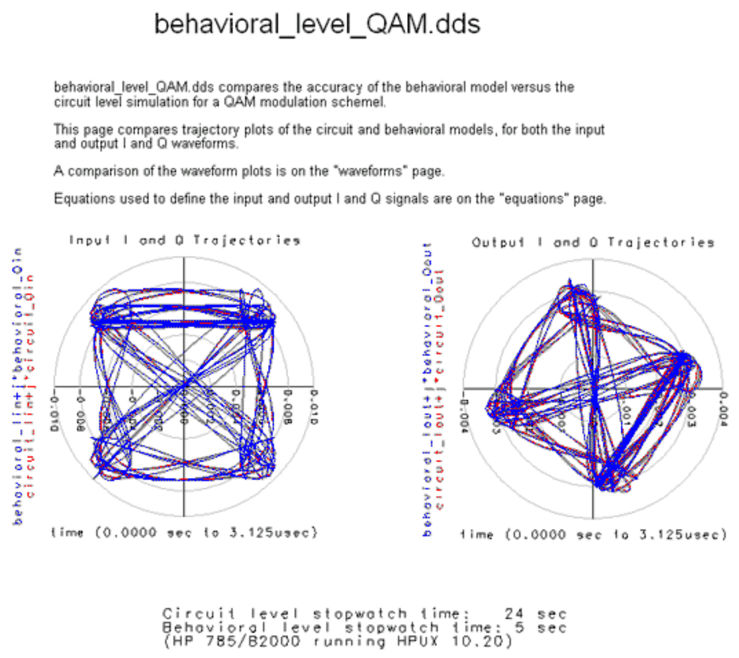
A comparison of the waveform plots is on the "waveforms" page.

Equations used to define the input and output I and Q signals are on the "equations" page.

Input I and Q Trajectories
behavioral_Iin+j*behavioral_Qin
circuit_Iin+j*circuit_Qin
time (0.0000 sec to 3.125usec)
Output I and Q Trajectories
behavioral_Iout+j*behavioral_Qout
circuit_Iout+j*circuit_Qout
time (0.0000 sec to 3.125usec)

Circuit level stopwatch time: 65 sec
Behavioral level stopwatch time: 5 sec
(HP 785/B2000 running HPUX 10.20)

Figure 2: Circuit and behavioral level results for 16 QAM modulation



behavioral_level_QAM.dds

behavioral_level_QAM.dds compares the accuracy of the behavioral model versus the circuit level simulation for a QAM modulation schemel.

This page compares trajectory plots of the circuit and behavioral models, for both the input and output I and Q waveforms.

A comparison of the waveform plots is on the "waveforms" page.

Equations used to define the input and output I and Q signals are on the "equations" page.

Circuit level stopwatch time: 24 sec
Behavioral level stopwatch time: 5 sec
(HP 785/B2000 running HPUX 10.20)

Input I and Q Trajectories
behavioral_Iin+j*behavioral_Qin
circuit_Iin+j*circuit_Qin
-0.010
-0.008
0.008
0.010
time (0.0000 sec to 3.125usec)
Output I and Q Trajectories
behavioral_Iout+j*behavioral_Qout
circuit_Iout+j*circuit_Qout
-0.004
0.001
0.003
0.004
time (0.0000 sec to 3.125usec)

Notes

- In Figure 1, a comparison of circuit and behavioral level results is given for Circuit Envelope simulation. Both input and output waveforms are modeled accurately.
- In Figure 2, a comparison of circuit and behavioral level results is given for Circuit Envelope simulation. Strong agreement is shown for both the input and output IQ-trajectories.

Data Based Load Pull Amplifier

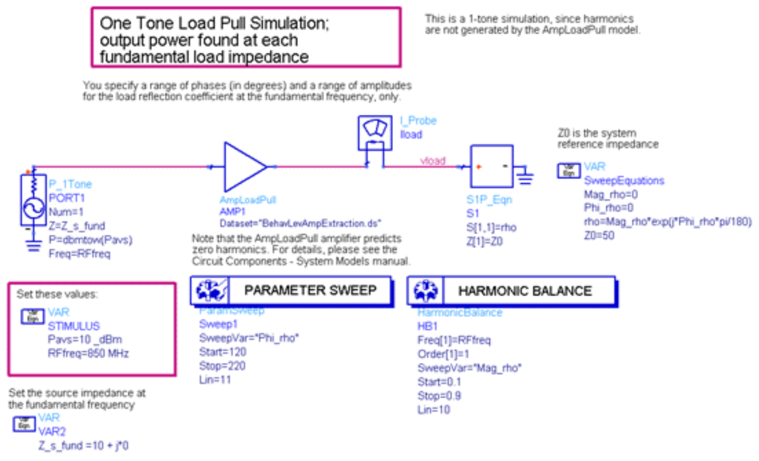
Location: $HPEESOF_DIR/examples/BehavioralModels/AmpLoadPull_wrk

Objective

This example demonstrates the use of the AmpLoadPull and LoadPullSetup components in ADS. Load-pull simulations generate contours that indicate load impedances that, when presented to the output of a device (along with the specified source impedances and available source power), would cause a certain power to be delivered to the load.

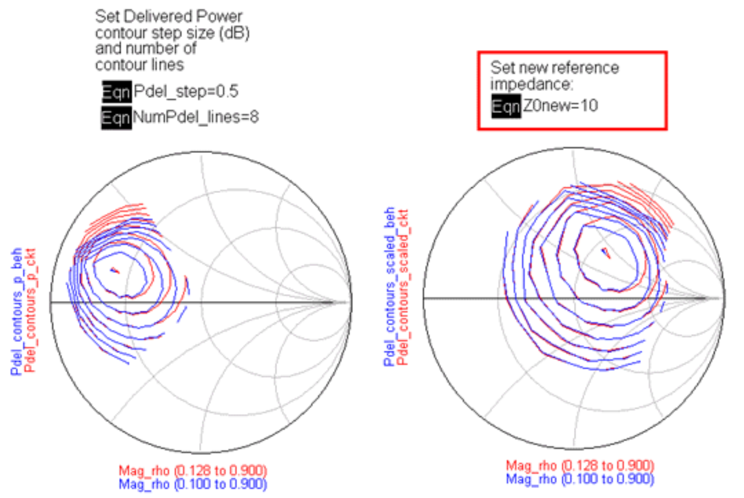
Setup

1. "CktLevelAmp1Tone" is a schematic for a circuit level 1-tone fixed input power swept reflection coefficient harmonic balance simulation of the amplifier. The result is in the dataset "CktLevelAmp1Tone.ds."
2. "BehavLevelAmpExtraction" is a schematic using LoadPullSetup for the generation of the behavioral model for the amplifier. The result is in the dataset "BehavLevelAmpExtraction.ds."
3. "BehavLevelAmp1Tone" is a schematic for a behavioral level 1-tone fixed input power swept reflection coefficient harmonic balance simulation of the amplifier of interest using AmpLoadPull along with the dataset "BehavLevelAmpExtraction.ds." The result is in the dataset "BehavLevelAmp1Tone.ds."


One Tone Load Pull Simulation; output power found at each fundamental load impedance
This is a 1-tone simulation, since harmonics are not generated by the AmpLoadPull model.
You specify a range of phases (in degrees) and a range of amplitudes for the load reflection coefficient at the fundamental frequency, only.
I_Probe
Iload
Z0 is the system reference impedance
vload
P_1Tone
PORT1
Num=1
Z=Z_s_fund
P=dbmtow(Pavs)
Freq=RFfreq
AmpLoadPull
AMP1
Dataset="BehavLevAmpExtraction.ds"
Note that the AmpLoadPull amplifier predicts zero harmonics. For details, please see the Circuit Components - System Models manual.
S1P_Eqn
S1
S[1,1]=rho
Z[1]=Z0
VAR
SweepEquations
Mag_rho=0
Phi_rho=0
rho=Mag_rho*exp(j*Phi_rho*pi/180)
Z0=50
Set these values:
VAR
STIMULUS
Pavs=10 _dBm
RFfreq=850 MHz
PARAMETER SWEEP
ParamSweep
Sweep1
SweepVar="Phi_rho"
Start=120
Stop=220
Lin=11
HARMONIC BALANCE
HarmonicBalance
HB1
Freq[1]=RFfreq
Order[1]=1
SweepVar="Mag_rho"
Start=0.1
Stop=0.9
Lin=10
Set the source impedance at the fundamental frequency
VAR
VAR2
Z_s_fund =10 + j*0

Analysis

Figure 1: Circuit and behavioral level results of load pull analysis



Notes

- In Figure 1, a comparison of circuit and behavioral level results is given for a Harmonic Balance loadpull analysis. Specifically, contours of constant power delivered to the load are shown on 50 Ohm and 10 Ohm reference impedance Smith charts. The agreement is generally good. You may set the source impedance of the LoadPullSetup by pushing into the subcircuit.
- This workspace is an extension of examples/RF_Board/LoadPull_wrk.

Data Based Mixer

Location: $HPEESOF_DIR/examples/BehavioralModels/MixerHBdata_wrk

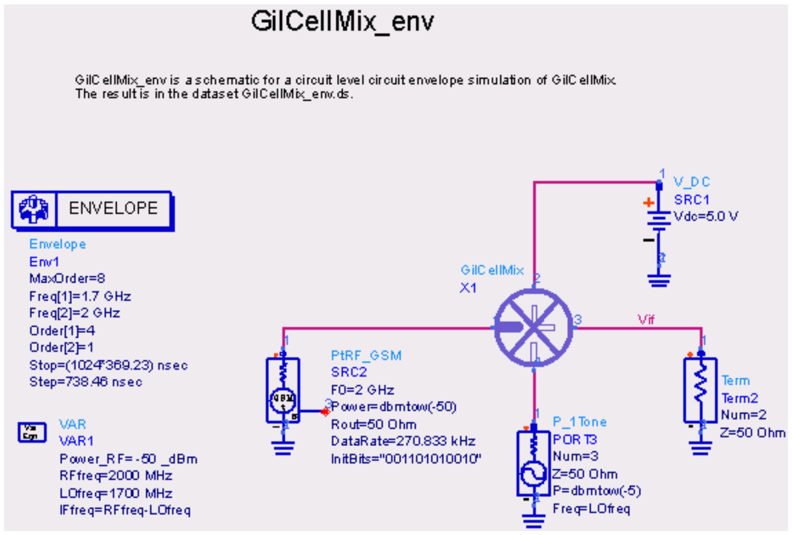
Objective

Behavioral models are reduced-order models of circuit level devices such as amplifiers, mixers, modulators, and demodulators. After initial extraction of data for the circuit level devices and subsequent generation of their behavioral models, such models can be used as building blocks for receivers and transmitters and allow a fast but accurate system level simulation that cannot be completed at the circuit level.The MixerHBdata workspace illustrates the use of the MixerHBdata and MixerHBsetup components in ADS. These components are part of the ADS behavioral model suite found under the System - Data Models palette.

Setup

1. "GilCellMix_env" is a schematic for a circuit level circuit envelope simulation of "GilCellMix". The result is in the dataset "GilCellMix_env.ds".
   "GilCellMix_HB" is a schematic for a circuit level 1-tone fixed input power harmonic

2. balance simulation of "GilCellMix". The result is in the dataset "GilCellMix_HB.ds".
3. "MixerHBsetup_GilCellMix" is a schematic for generating the behavioral model for "GilCellMix". The result is in the dataset "MixerHBsetup_GilCellMix.ds".
4. "MixerHBdata_downconvert" is a schematic for a behavioral level 1-tone fixed input power harmonic balance simulation of "GilCellMix" using MixerHBdata along with the dataset "MixerHBsetup_GilCellMix.ds". The result is valid for down-conversion only and is in the dataset "MixerHBdata_downconvert.ds".
5. "MixerHBdata_upconvert" is a schematic for a behavioral level 1-tone fixed input power harmonic balance simulation of "GilCellMix" using MixerHBdata along with the dataset "MixerHBsetup_GilCellMix.ds". The result is valid for up-conversion only and is in the dataset "MixerHBdata_upconvert.ds".
6. "GilCellMix_HB_powersweep" is a schematic for a circuit level 1-tone swept input power harmonic balance simulation of "GilCellMix". The result is in the dataset "GilCellMix_HB_powersweep.ds".
7. "MixerHBdata_downconvert_powersweep" is a schematic for a behavioral level 1-tone swept input power harmonic balance simulation of "GilCellMix" using MixerHBdata along with the "dataset MixerHBsetup_GilCellMix.ds". The result is valid for down-conversion only and is in the dataset "MixerHBdata_downconvert_powersweep.ds".
8. "MixerHBdata_upconvert_powersweep" is a schematic for a behavioral level 1-tone swept input power harmonic balance simulation of "GilCellMix" using MixerHBdata along with the dataset "MixerHBsetup_GilCellMix.ds". The result is valid for up-conversion only and is in the dataset "MixerHBdata_upconvert_powersweep.ds".
9. "GilCellMix_env" is a schematic for a circuit level circuit envelope simulation of "GilCellMix". The result is in the dataset "GilCellMix_env.ds".
10. "MixerHBdata_downconvert_env" is a schematic for a behavioral level circuit envelope simulation of "GilCellMix" using MixerHBdata along with the dataset "MixerHBsetup_GilCellMix.ds". The result is valid for down-conversion only and is in the dataset "MixerHBdata_downconvert_env.ds".

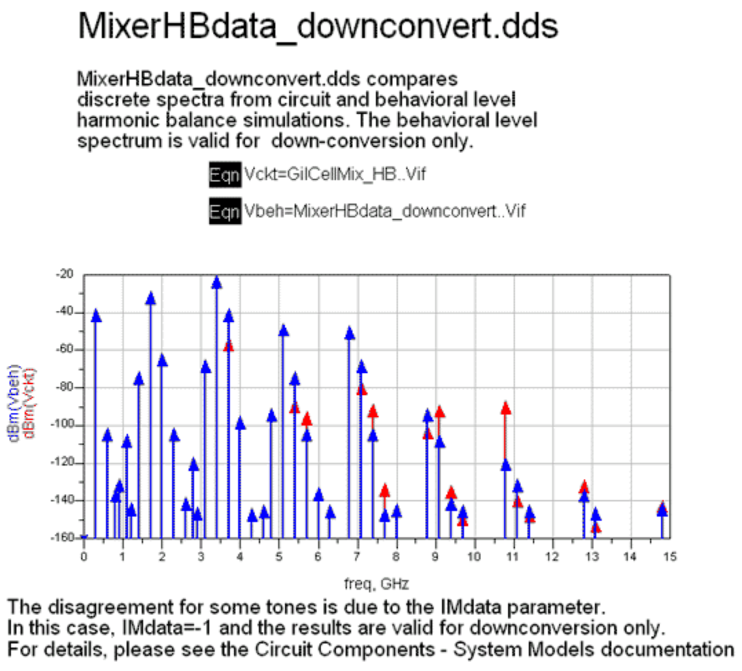

GilCellMix_env

GilCellMix_env is a schematic for a circuit level circuit envelope simulation of GilCellMix.
The result is in the dataset GilCellMix_env.ds.

```
ENVELOPE
Envelope
Env1
MaxOrder=8
Freq[1]=1.7 GHz
Freq[2]=2 GHz
Order[1]=4
Order[2]=1
Stop=(1024*369.23) nsec
Step=738.46 nsec

VAR
VAR1
Power_RF=-50 _dBm
RFfreq=2000 MHz
LOfreq=1700 MHz
IFfreq=RFfreq-LOfreq

PtRF_GSM
SRC2
F0=2 GHz
Power=dbmtow(-50)
Rout=50 Ohm
DataRate=270.833 kHz
InitBits="001101010010"

GilCellMix
X1

V_DC
SRC1
Vdc=5.0 V

P_1Tone
PORT3
Num=3
Z=50 Ohm
P=dbmtow(-5)
Freq=LOfreq

Term
Term2
Num=2
Z=50 Ohm
```

Analysis

Figure 1: Circuit and behavioral level results for Harmonic Balance simulation



MixerHBdata_downconvert.dds

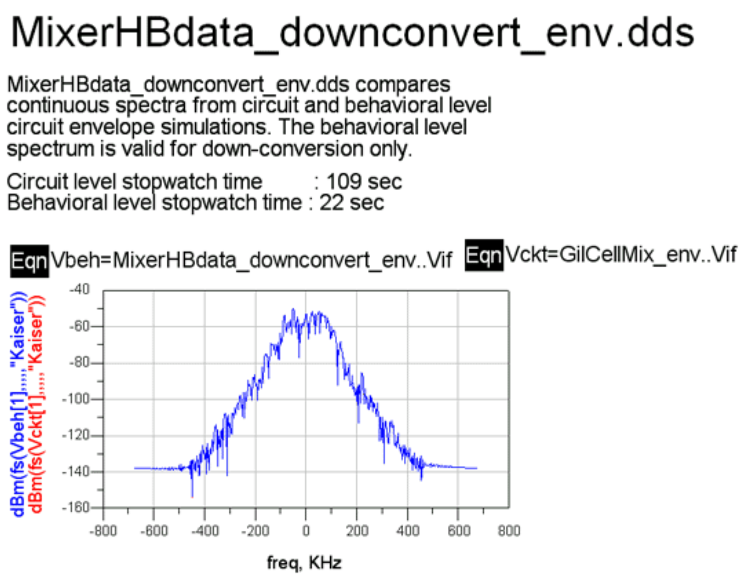
MixerHBdata_downconvert.dds compares discrete spectra from circuit and behavioral level harmonic balance simulations. The behavioral level spectrum is valid for down-conversion only.

Eqn Vckt=GilCellMix_HB..Vif

Eqn Vbeh=MixerHBdata_downconvert..Vif

The disagreement for some tones is due to the IMdata parameter.
In this case, IMdata=-1 and the results are valid for downconversion only.
For details, please see the Circuit Components - System Models documentation

Figure 2: Circuit and behavioral level down-conversion results for Circuit Envelope simulation



MixerHBdata_downconvert_env.dds

MixerHBdata_downconvert_env.dds compares continuous spectra from circuit and behavioral level circuit envelope simulations. The behavioral level spectrum is valid for down-conversion only.

Circuit level stopwatch time : 109 sec
Behavioral level stopwatch time : 22 sec

Eqn Vbeh=MixerHBdata_downconvert_env..Vif

Eqn Vckt=GilCellMix_env..Vif

Notes

1. In Figure 1, a comparison of circuit and behavioral level results is given for Harmonic Balance simulation. The behavioral level spectrum is valid for down-conversion only. The up-conversion tones are not accurately modeled. For details, please see the description of the IMdata parameter in the *System Models* (ccsys).
2. In Figure 2, a comparison of circuit and behavioral level results is given for Circuit Envelope simulation. Strong agreement is shown for the spectra around the first difference carrier frequency.

Data Based Models for Differentially Fed Components

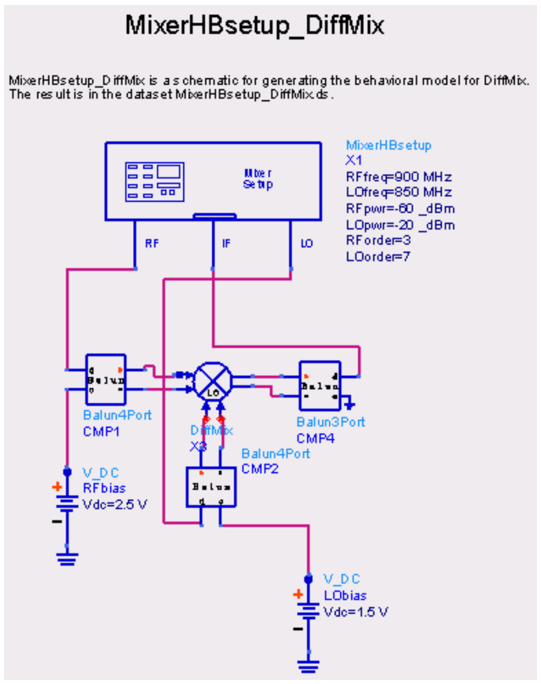
Location: $HPEESOF_DIR/examples/BehavioralModels/DifferentialModels_wrk

Objective

Behavioral models are reduced-order models of circuit level devices such as amplifiers, mixers, modulators, and demodulators. After initial extraction of data for the circuit level devices and subsequent generation of their behavioral models, such models can be used as building blocks for receivers and transmitters and allow a fast but accurate system level simulation that cannot be completed at the circuit level.The Differential Models workspace illustrates the use of the Balun3Port and Balun4Port components in ADS. These components are part of the ADS behavioral model suite found under the System - Data Models palette. They are also available from the System - Passive palette. For details on creating data based models for single ended mixers, please see *Data Based Mixer* (examples).

Setup

1. "DiffMix_HB" is a schematic for a circuit level 1-tone fixed input power harmonic balance simulation of "DiffMix". The result is in the dataset "DiffMix_HB.ds".
2. "MixerHBsetup_DiffMix" is a schematic for generating the behavioral model for "DiffMix". The result is in the dataset "MixerHBsetup_DiffMix.ds".
3. "MixerHBdata_downconvert" is a schematic for a behavioral level 1-tone fixed input power harmonic balance simulation of "DiffMix" using MixerHBdata along with the dataset "MixerHBsetup_DiffMix.ds". The result is valid for down-conversion only and is in the dataset "MixerHBdata_downconvert.ds".
4. "DiffMix_env" is a schematic for a circuit level circuit envelope simulation of "DiffMix". The result is in the dataset "DiffMix_env.ds".
5. "MixerHBdata_downconvert_env" is a schematic for a behavioral level circuit envelope simulation of "DiffMix" using MixerHBdata along with the dataset "MixerHBsetup_DiffMix.ds". The result is valid for down-conversion only and is in the dataset "MixerHBdata_downconvert_env.ds".


MixerHBsetup_DiffMix

MixerHBsetup_DiffMix is a schematic for generating the behavioral model for DiffMix.
The result is in the dataset MixerHBsetup_DiffMix.ds.

MixerHBsetup
X1
RFfreq=900 MHz
LOfreq=850 MHz
RFpwr=-60 _dBm
LOpwr=-20 _dBm
RForder=3
LOorder=7

V_DC
RFbias
Vdc=2.5 V

V_DC
LObias
Vdc=1.5 V

Analysis

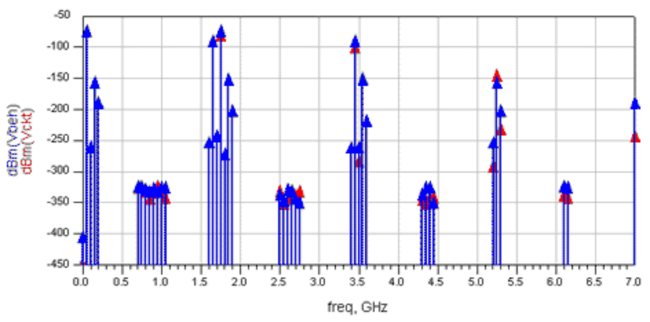
Figure 1: Circuit and behavioral level results for Harmonic Balance simulation

MixerHBdata_downconvert.dds

MixerHBdata_downconvert.dds compares discrete spectra from circuit and behavioral level harmonic balance simulations. The behavioral level spectrum is valid for down-conversion only.

Eqn Vckt=DiffMix_HB..Vif

Eqn Vbeh=MixerHBdata_downconvert..Vif



The disagreement for some tones is due to the IMdata parameter.
In this case, IMdata=-1 and the results are valid for downconversion only.
For details, please see the Circuit Components - System Models documentation

Notes

1. In Figure 1, a comparison of circuit and behavioral level results is given for Harmonic Balance simulation. The behavioral level spectrum is valid for down-conversion only. The up-conversion tones are not accurately modeled. For details, please see the description of the IMdata parameter of the *MixerHBdata (2-Tone HB Mixer)* (ccsys) component in the *System Models* (ccsys) documentation.

Extraction and use of IMT Based System Level Mixers

Location: $HPEESOF_DIR/examples/BehavioralModels/MixIMT_wrk

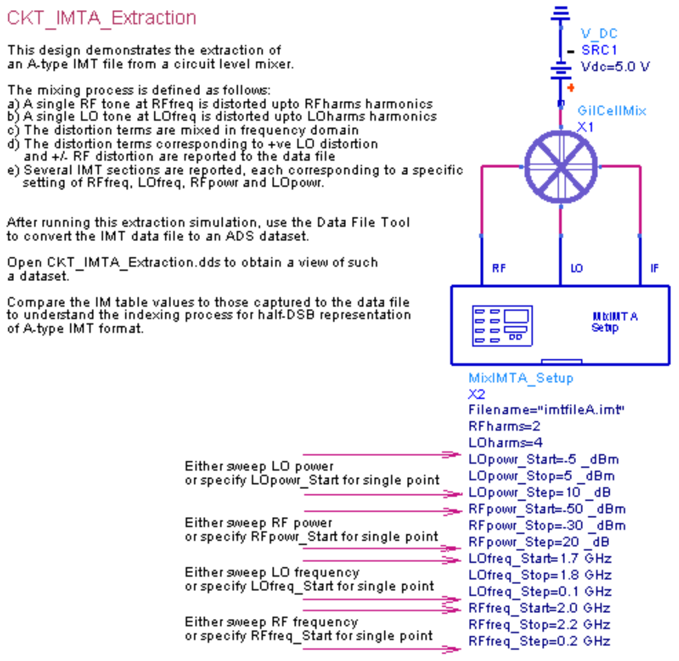
Objective

To provide the Use Model of the MixIMT Family
The MixIMT family provides a behavioral modeling solution for mixers. The family consists of two IMT data extractor components MixIMTA_Setup and MixIMTB_Setup and a data model component MixIMT_Data. All three components are available from the *System-Data Models* library.This example workspace shows how the two extractors can be set up to generate IMT files from a circuit level mixer model in ADS environment and how the data model can be used to interpret such IMT files to regenerate mixer behavior.This example also shows how to use original IMT files obtained from manufacturer specification sheets or legacy projects with the MixIMT_Data model.This example does not explicitly contain instances of MixerIMT or MixerIMT2. However, descriptive and graphical references are made to these two legacy components to illustrate the capabilities of MixIMT_Data.

Setup

Extraction of IMT Data
Mixer behavior has been defined using intermodulation table (IMT) data or spur charts since the 1980s. Such charts used to be sufficient only for single RF signals and contain only single side banded IF strengths without phase data. Application of such incomplete IM tables to multi-tone RF systems was insupportable in theory and frustrating in practice. As of ADS2006A it is possible to perform automatic extraction of single or multi-RF spur charts using the MixIMT*_Setup components as shown below. Both extractors generate complex double side banded spur information with some capability of sweeping signal power and frequency such that the same IMT file can capture.MixIMTA_Setup is designed to extract IMT data for a mixer response to the classic single RF, single LO. The degree of distortion expected to contribute to the mixing matrix can be specified at RF and LO ports via relevant parameters. Both LO and RF frequencies and powers can be swept with an IMT being captured at each combination of the 4-tuple {RFfreq, LOfreq, RFpowr, LOpowr} as shown in *CKT_IMTA_extraction*.

Figure 1: Extraction of A-type IMT data


CKT_IMTA_Extraction

This design demonstrates the extraction of
an A-type IMT file from a circuit level mixer.

The mixing process is defined as follows:
a) A single RF tone at RFfreq is distorted upto RFharms harmonics
b) A single LO tone at LOfreq is distorted upto LOharms harmonics
c) The distortion terms are mixed in frequency domain
d) The distortion terms corresponding to +ve LO distortion
and +/- RF distortion are reported to the data file
e) Several IMT sections are reported, each corresponding to a specific
setting of RFfreq, LOfreq, RFpowr and LOpowr.

After running this extraction simulation, use the Data File Tool
to convert the IMT data file to an ADS dataset.

Open CKT_IMTA_Extraction.dds to obtain a view of such
a dataset.

Compare the IM table values to those captured to the data file
to understand the indexing process for half-DSB representation
of A-type IMT format.

MixIMTA_Setup
X2
Filename="imtfileA.imt"
RFharms=2
LOharms=4
LOpowr_Start=-5 _dBm
LOpowr_Stop=5 _dBm
LOpowr_Step=10 _dB
RFpowr_Start=-50 _dBm
RFpowr_Stop=-30 _dBm
RFpowr_Step=20 _dB
LOfreq_Start=1.7 GHz
LOfreq_Stop=1.8 GHz
LOfreq_Step=0.1 GHz
RFfreq_Start=2.0 GHz
RFfreq_Stop=2.2 GHz
RFfreq_Step=0.2 GHz

Either sweep LO power
or specify LOpowr_Start for single point

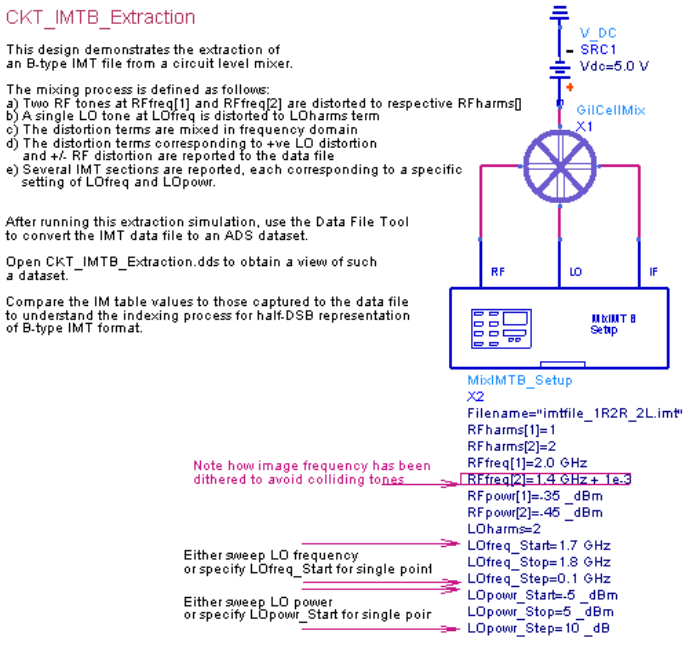
Either sweep RF power
or specify RFpowr_Start for single point

Either sweep LO frequency
or specify LOfreq_Start for single point

Either sweep RF frequency
or specify RFfreq_Start for single point

MixIMTB_Setup is designed to extract IMT data for a mixer response to multiple RF tones and single LO. The degree of distortion expected to contribute to the mixing matrix can be specified at RF and LO ports via relevant parameters. Both LO frequency and power can be swept with an IMT being captured at each combination of the {LOfreq, LOpowr} as shown in *CKT_IMTB_extraction*. RF frequencies and powers should remain constant throughout the extraction process. Note how RFfreq[2], the image frequency is offset by a minor amount to capture individual mixing contributions to colliding tones.

Figure 2: Extraction of B-type IMT data


CKT_IMTB_Extraction

This design demonstrates the extraction of
an B-type IMT file from a circuit level mixer.

The mixing process is defined as follows:
a) Two RF tones at RFfreq[1] and RFfreq[2] are distorted to respective RFharms[]
b) A single LO tone at LOfreq is distorted to LOharms term
c) The distortion terms are mixed in frequency domain
d) The distortion terms corresponding to +ve LO distortion
and +/- RF distortion are reported to the data file
e) Several IMT sections are reported, each corresponding to a specific
setting of LOfreq and LOpowr.

After running this extraction simulation, use the Data File Tool
to convert the IMT data file to an ADS dataset.

Open CKT_IMTB_Extraction.dds to obtain a view of such
a dataset.

Compare the IM table values to those captured to the data file
to understand the indexing process for half-DSB representation
of B-type IMT format.

MixIMTB_Setup
X2
Filename="imtfile_1R2R_2L.imt"
RFharms[1]=1
RFharms[2]=2
RFfreq[1]=2.0 GHz
RFfreq[2]=1.4 GHz + 1e-3
RFpowr[1]=-35 _dBm
RFpowr[2]=-45 _dBm
LOharms=2
LOfreq_Start=1.7 GHz
LOfreq_Stop=1.8 GHz
LOfreq_Step=0.1 GHz
LOpowr_Start=-5 _dBm
LOpowr_Stop=5 _dBm
LOpowr_Step=10 _dB

Note how image frequency has been
dithered to avoid colliding tones

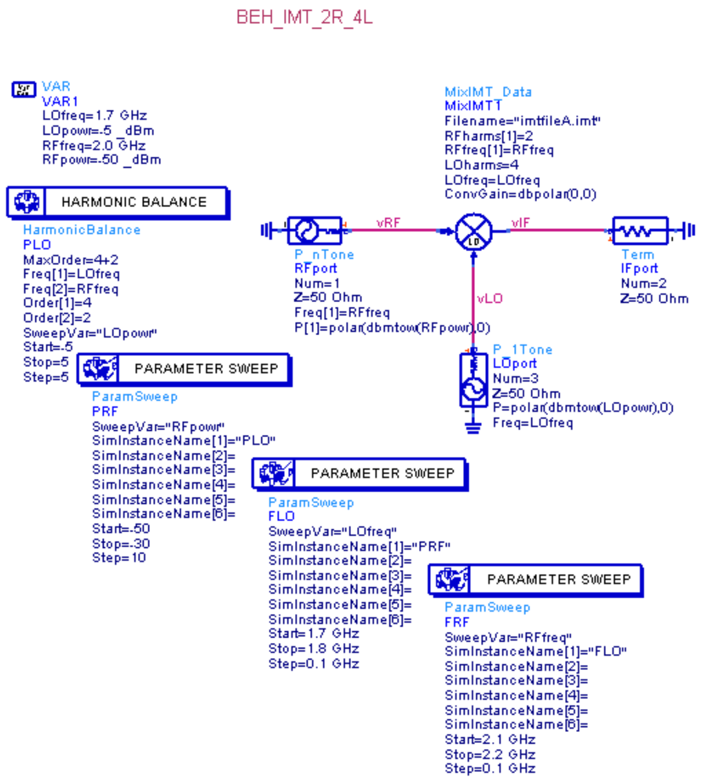
Either sweep LO frequency
or specify LOfreq_Start for single point

Either sweep LO power
or specify LOpowr_Start for single poir

Use of IMT data with MixIMT_Data model
MixIMT_Data can accept both original single side banded (O-type) and double side banded (A-type and B-type) IMT data. It can be assigned sensitivity to RF and LO frequencies using relevant parameters. The degree of distortion desired from any particular tone can be set independently. This data model can perform some amount of frequency and power domain interpolation of IMT data. Frequency domain interpolation is linear along RF (only for A-type data) and LO (for both A and B types) with constant extrapolation enforced beyond the highest and lowest frequencies on file. Power domain interpolation and extrapolation involves identification of the nearest neighboring IMT data followed by up or

down scaling IF vector values to adjust various mixing products.The design set CKT_IMT_2R_4L and *BEH_IMT_2R_4L* show the frequency and power domain interpolation performance of the data model for the A-type IM table captured in CKT_IMTA_extraction.

Figure 3: Testing MixIMT_Data for frequency and power domain sensitivity



```
VAR
VAR1
LOfreq=1.7 GHz
LOpowr=-5 _dBm
RFfreq=2.0 GHz
RFpowr=-50 _dBm

HarmonicBalance
PLO
MaxOrder=4+2
Freq[1]=LOfreq
Freq[2]=RFfreq
Order[1]=4
Order[2]=2
SweepVar="LOpowr"
Start=-5
Stop=5
Step=5

MixIMT_Data
MixIMTT
Filename="imtfileA.imt"
RFharms[1]=2
RFfreq[1]=RFfreq
LOharms=4
LOfreq=LOfreq
ConvGain=dbpolar(0,0)

P_nTone
RFport
Num=1
Z=50 Ohm
Freq[1]=RFfreq
P[1]=polar(dbmtow(RFpowr),0)

Term
IFport
Num=2
Z=50 Ohm

P_1Tone
LOport
Num=3
Z=50 Ohm
P=polar(dbmtow(LOpowr),0)
Freq=LOfreq

ParamSweep
PRF
SweepVar="RFpowr"
SimInstanceName[1]="PLO"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=-50
Stop=-30
Step=10

ParamSweep
FLO
SweepVar="LOfreq"
SimInstanceName[1]="PRF"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1.7 GHz
Stop=1.8 GHz
Step=0.1 GHz

ParamSweep
FRF
SweepVar="RFfreq"
SimInstanceName[1]="FLO"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=2.1 GHz
Stop=2.2 GHz
Step=0.1 GHz
```

Test designs CKT_IMT_1R2R_2L, BEH_IMT_1R2R_2L and BEH_IMT_1R1R_2L are used to demonstrate the multi-RF modeling capability of MixIMT_Data. The B-type IMT file extracted by CKT_IMTB_extraction are used with the data model as follows:

1. BEH_IMT_1R2R_2L - Full mixing matrix is utilized.
2. BEH_IMT_1R1R_2L - The distortion effects of the 2nd RF tone is restricted to 1st degree.

Figure 4: Reduced simulation

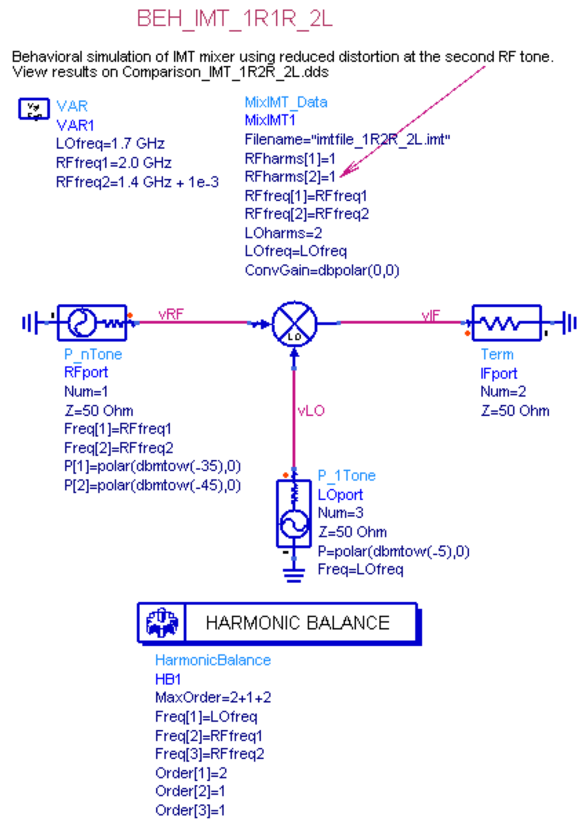
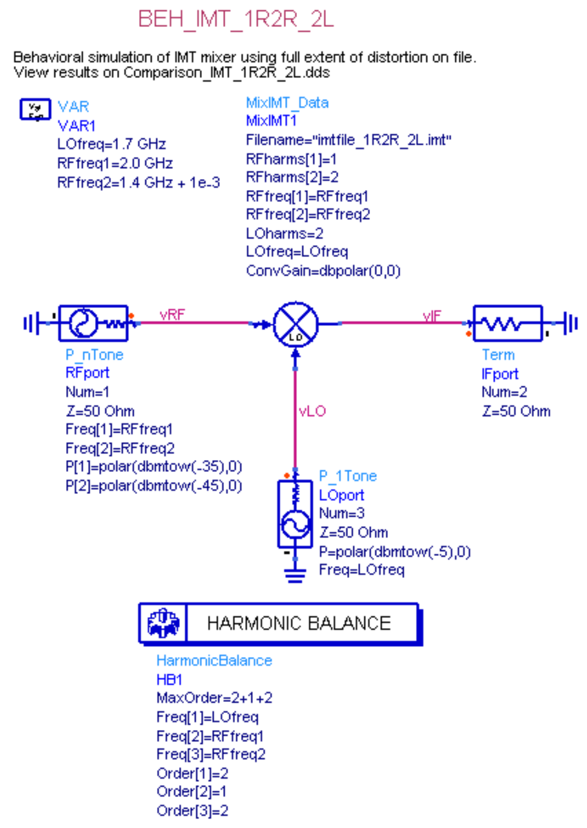


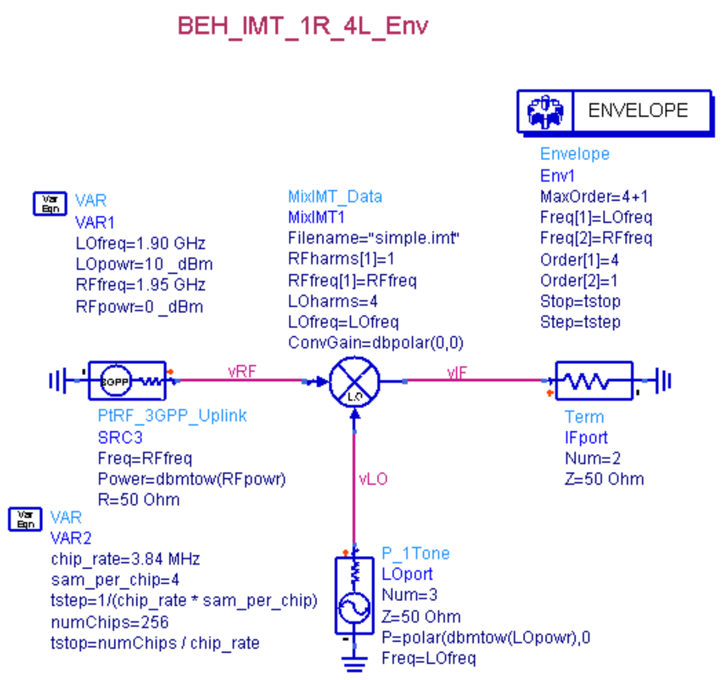
Figure 5: Full simulation



The results are compared against the full distortion performance of the circuit level model.The performance and memory consumption for the second case is less than that of full distortion behavioral simulation. This demonstrates the dynamic nature of the data model by which it is able respond to user request for reduced resources without compromizing data accuracy.The response of the data model to ConvGain and leakage

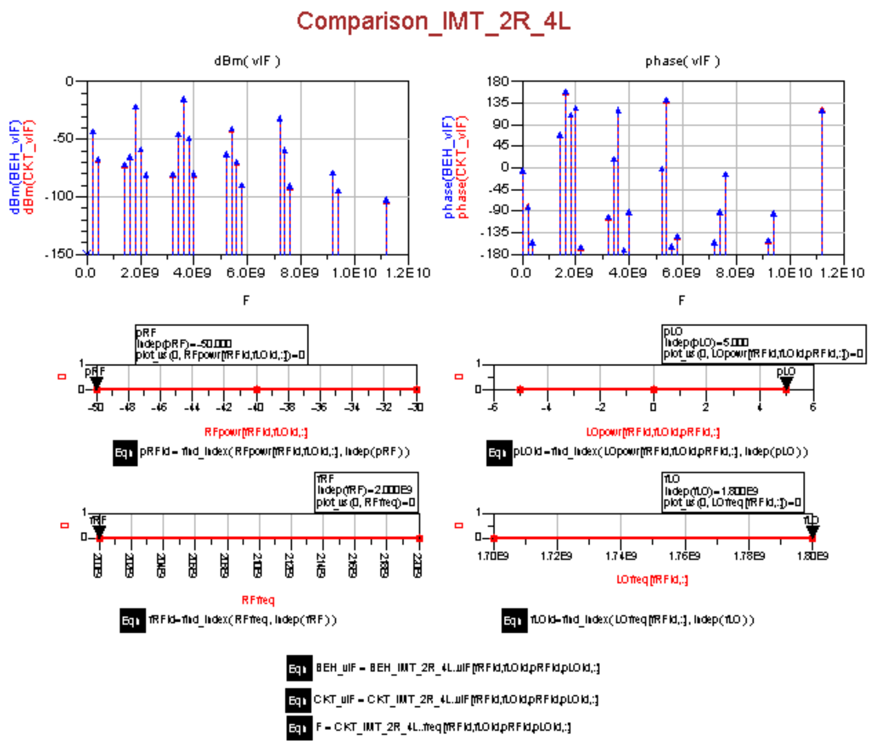
and reflection parameters are shown in BEH_IMT_2R_4L_ConvGain and BEH_IMT_2R_4L_LeakageReflection respectively. See analysis section for results.
The MixIMT_Data model is capable of generating fast and accurate envelope responses to complex modulated waveforms e.g. a 3GPP Uplink signal.

Figure 6: Using MixIMT_Data in an envelope simulation for a 3GPP signal



Analysis

Verifying frequency and power domain accuracy of MixIMT_Data
Comparison_IMT_2R_4L.dds shows the accuracy of the behavioral model in modeling A-type IMT data for swept RF and LO powers and frequencies. The extremal points of powers and frequencies along each of the four independent axes are on file, with the middle points being interpolated estimates.



Eqn pRFid= find_index(RFpowr[fRFid,fLOid,::], indep(pRF))

Eqn pLOid=find_index(LOpowr[fRFid,fLOid,pRFid,::], indep(pLO))

Eqn fRFid=find_index(RFfreq, indep(fRF))

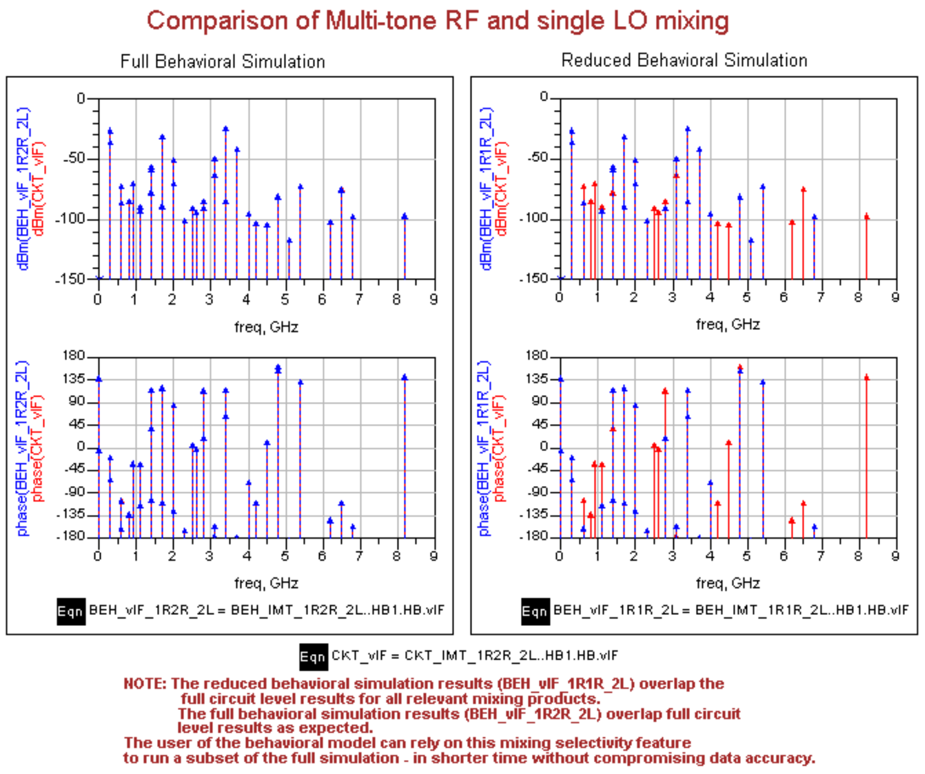
Eqn fLOid=find_index(LOfreq[fRFid,::], indep(fLO))

Eqn BEH_vIF = BEH_IMT_2R_4L.vIF[fRFid,fLOid,pRFid,pLOid,::]

Eqn CKT_vIF = CKT_IMT_2R_4L.vIF[fRFid,fLOid,pRFid,pLOid,::]

Eqn F = CKT_IMT_2R_4L..freq[fRFid,fLOid,pRFid,pLOid,::]

Customized distortion effects from MixIMT_Data
Comparison_IMT_1R2R_2L.dds shows that comparable levels of accuracy are acheieved for mixing products common to full scale IMT modeling in BEH_IMT_1R2R_2L and reduced scale IMT modeling in BEH_IMT_1R1R_2L.



Eqn BEH_vIF_1R2R_2L = BEH_IMT_1R2R_2L..HB1.HB.vIF

Eqn BEH_vIF_1R1R_2L = BEH_IMT_1R1R_2L..HB1.HB.vIF

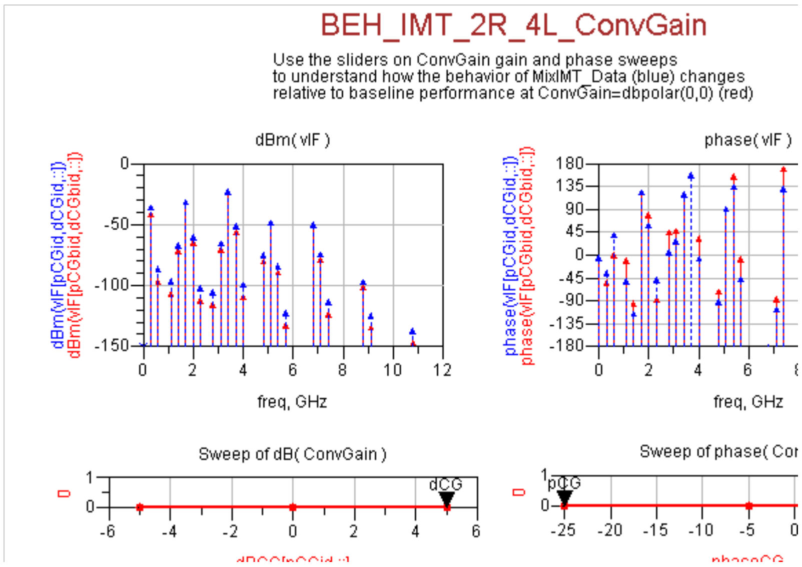
Eqn CKT_vIF = CKT_IMT_1R2R_2L..HB1.HB.vIF

NOTE: The reduced behavioral simulation results (BEH_vIF_1R1R_2L) overlap the full circuit level results for all relevant mixing products.
The full behavioral simulation results (BEH_vIF_1R2R_2L) overlap full circuit level results as expected.
The user of the behavioral model can rely on this mixing selectivity feature to run a subset of the full simulation - in shorter time without compromising data accuracy.

Conversion Gain
Conversion Gain is a voltage valued parameter defined on the mixer model that allows the IF side output to be impacted by a fixed vector. The test design BEH_IMT_2R_4L_ConvGain demonstrates how the dB and phase values of this parameter may be swept to achieve relevant IF offsets from the baseline behavior defined in the data profile. Note that ConvGain affects behavior only at IF port with no impact at the RF or LO

inputs.

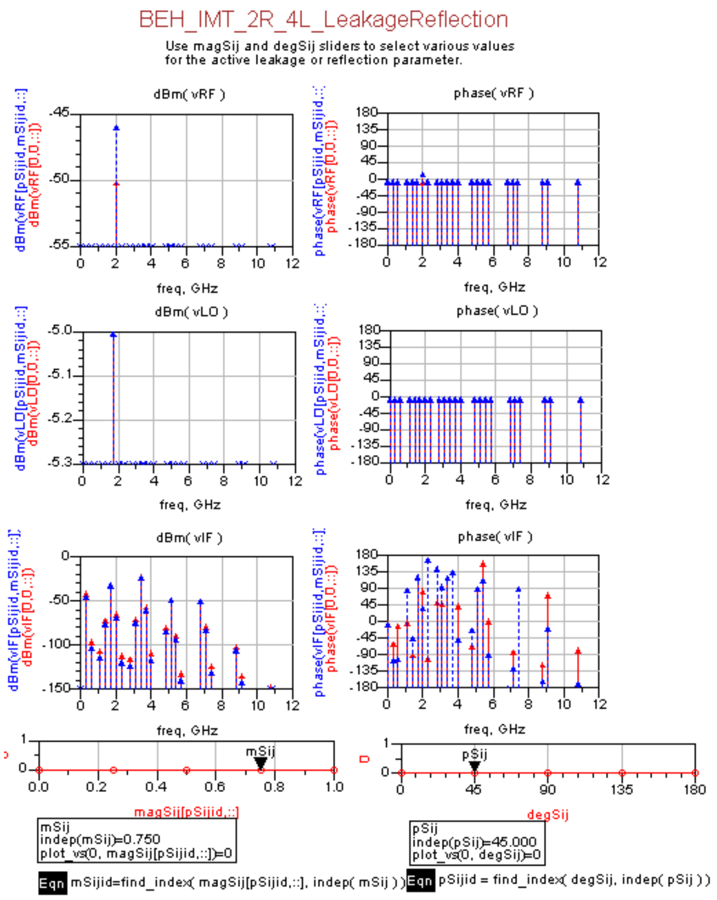
Figure 7: Effect of conversion gain { 5 dB, -25 deg} raises power while adding phase lag at IF port



Leakage and Reflection

Leakge and reflection at the three mixer ports can be defined using nine S-parameters on the MixIMT_Data model. Note that these are not the FSP parameters captured to IMT files in CKT_IMTA_extraction.dds. These S-parameters are not frequency-translated. They indicate alternate routes for flow of power between the ports other than through the nonlinear mixing element. In the example design BEH_IMT_2R_4L_LeakageReflection a single complex number is associated with one of the nine S-parameters and its magnitude and phase swept to generate the mixer response to user-imposed perturbations. The impacted complex parameter can be changed to view the effects of various port to port leakages and reflections. The data display allows interactive viewing of the changes to RF, LO and IF voltage vectors relative to the unperturbed state defined in the IM table.

Figure 8: Effect of setting RFreflection = polar(0.75, 45)


BEH_IMT_2R_4L_LeakageReflection
Use magSij and degSij sliders to select various values for the active leakage or reflection parameter.
dBm(vRF)
phase(vRF)
dBm(vLO)
phase(vLO)
dBm(vIF)
phase(vIF)
freq, GHz
dBm(vRF[pSijid,mSijid,::])
dBm(vRF[0,0,::])
phase(vRF[pSijid,mSijid,::])
phase(vRF[0,0,::])
dBm(vLO[pSijid,mSijid,::])
dBm(vLO[0,0,::])
phase(vLO[pSijid,mSijid,::])
phase(vLO[0,0,::])
dBm(vIF[pSijid,mSijid,::])
dBm(vIF[0,0,::])
phase(vIF[pSijid,mSijid,::])
phase(vIF[0,0,::])
mSij
pSij
magSij[pSijid,::]
degSij

mSij
indep(mSij)=0.750
plot_vs(0, magSij[pSijid,::])=0

pSij
indep(pSij)=45.000
plot_vs(0, degSij)=0

Eqn mSijid=find_index(magSij[pSijid,::], indep(mSij))

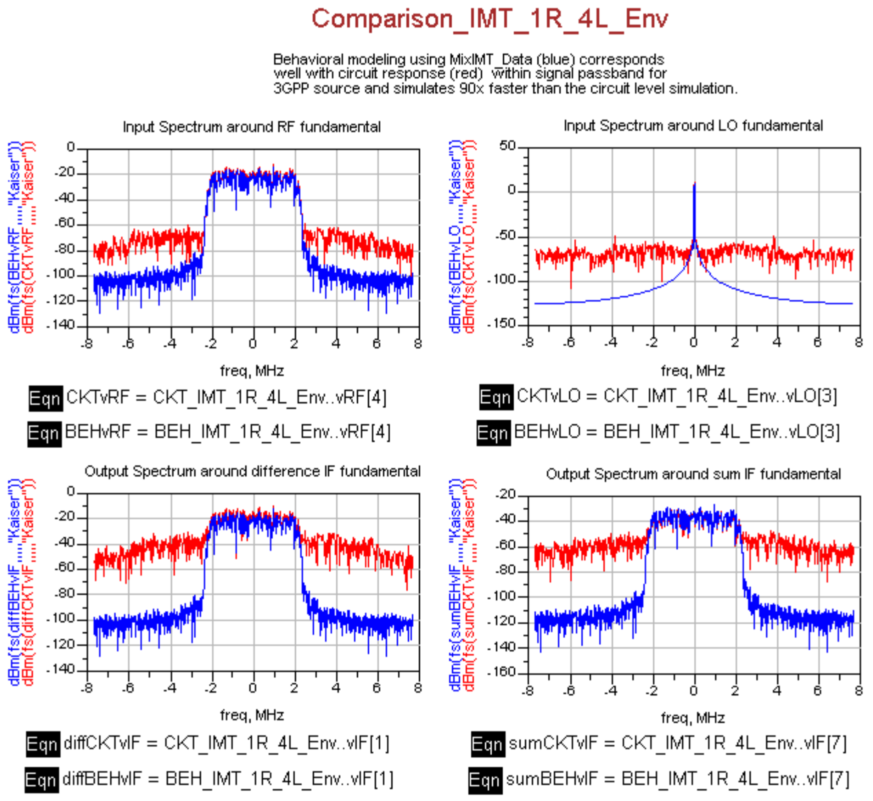
Eqn pSijid = find_index(degSij, indep(pSij))

Note impact on vRF and vIF spectra.

3GPP Envelope Simulation

As shown in the figure below, the behavioral response (blue) matches the circuit level response of the input and output side envelopes within respective passbands. Simulation time for the behavioral model was found to be less than 10 sec compared to 1 hour 10 minutes for the circuit level model.

Figure 9: Comparison of behavioral and circuit level spectra for mixing of a 3GPP signal at 1.95 GHz with a local oscillator at 1.9 GHz


Comparison_IMT_1R_4L_Env

Behavioral modeling using MixIMT_Data (blue) corresponds well with circuit response (red) within signal passband for 3GPP source and simulates 90x faster than the circuit level simulation.

Eqn CKTvRF = CKT_IMT_1R_4L_Env..vRF[4]

Eqn BEHvRF = BEH_IMT_1R_4L_Env..vRF[4]

Eqn CKTvLO = CKT_IMT_1R_4L_Env..vLO[3]

Eqn BEHvLO = BEH_IMT_1R_4L_Env..vLO[3]

Eqn diffCKTvIF = CKT_IMT_1R_4L_Env..vIF[1]

Eqn diffBEHvIF = BEH_IMT_1R_4L_Env..vIF[1]

Eqn sumCKTvIF = CKT_IMT_1R_4L_Env..vIF[7]

Eqn sumBEHvIF = BEH_IMT_1R_4L_Env..vIF[7]
Input Spectrum around RF fundamental
dBm(fs(BEHvRF,...,"Kaiser"))
dBm(fs(CKTvRF,...,"Kaiser"))
freq, MHz
Input Spectrum around LO fundamental
dBm(fs(BEHvLO,...,"Kaiser"))
dBm(fs(CKTvLO,...,"Kaiser"))
freq, MHz
Output Spectrum around difference IF fundamental
dBm(fs(diffBEHvIF,...,"Kaiser"))
dBm(fs(diffCKTvIF,...,"Kaiser"))
freq, MHz
Output Spectrum around sum IF fundamental
dBm(fs(sumBEHvIF,...,"Kaiser"))
dBm(fs(sumCKTvIF,...,"Kaiser"))
freq, MHz

Notes

Conversion Gain
Conversion Gain is an optional voltage gain parameter defined at the RF input of the IMT based mixer model. It applies independently to each incoming RF tone. The baseline behavior of the mixer is defined in the IMT file. If the user wishes to fine tune this behavior at IF output without impacting RF or LO signals themselves they can define ConvGain=dbpolar(x,y). This causes the RF signal to internally undergo the gain and phase change prior to nonlinear distortion and mixing.The baseline IF vector is impacted as follows:a. Un-mixed LO fundamental and LO harmonics are unchanged.b. RF-dependent tones increase in strength by m*x dB with phase gain of m*y degrees where m is a linear sum of RF mixing at the given tone. For instance if there are two RF tones, then the IF vector at (m1*f $_{RF1}$ - m2*f $_{RF2}$ + n*f $_{LO}$) will get impacted by a strength increase of (m1+m2)*x dB and a phase shift of (m1-m2)*y degrees.

VCA_Setup and VCA_Data

Location: $HPEESOF_DIR/examples/BehavioralModels/VCA_wrk

Objective

This example illustrates the use VME amplifier tools for the extraction and use of a Voltage Controlled Amplifier (VCA) profile defining a narrow-band power amplifier. VCA profiles are custom-designed ADS datasets extracted at a signle implicit frequency. They contain information regarding forward and reverse characteristics of an amplifier with respect to input RF power and DC bias. VCA datasets do not contain noise information. VCA datasets may be generated from a circuit level design using the VCA_Setup component. The VME data model VCA_Data is capable of using a VCA profile for simulating the non-linear behavior of the fundamental or nominal frequency component at the output. Harmonic or intermodulation product modeling is not recommended when using VCA_Data.VCA_Data is recommended for the design verification of applications operating at a fixed RF fundamental frequncy where estimation of efficiency under varying bias and input power conditions is of interest. This model is also capable of performing well under various loading conditions based on a single dataset is extracted at a nominal load of 50 Ohms. Details on the VME amplifier pair consisting of VCA_Setup and VCA_Data are described in Components > Circuit Components > System Models > System Data Models.

Setup

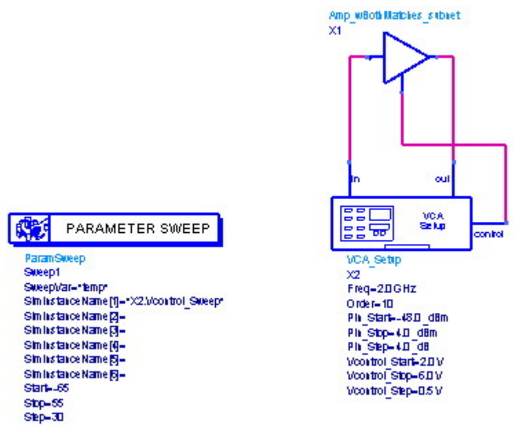
Circuit Level Model

The main circuit level standard used in this example is **Amp_wBothMatches_subnet**. This amplifier is ideally biased at 5.0 V. It is effectively a 3-port model. Note that the VCA_Data model is also a 3-port model. For details about this amplifier, please see *2GHz BJT Low Noise Amplifier* (examples).

Extraction Process

The VCA_Setup component is used to extract a basic building block of a VCA dataset in **Amp_wBothMatches_setup**. The setup component provides both RF and DC bias inputs to the circuit level model. If a circuit level amplifier were to have multiple bias pins, VCA_Setup should be connected to bias the pin experiencing maximum bias sensitivity. The other bias pins of the circuit level model should be fed off separate source components connected to the common DC ground. It is possible to mount additional custom sweeps above the VCA_Setup component in a single chain so long as the lowest sweep of the hierarchy is referenced to "Xy.Vcontrol_Sweep" where "Xy" is the instance ID of the extractor component. The resulting VCA dataset is extracted only at one frequency set by *Freq* on the extractor and must be used to emulate amplifier behavior only at this frequency of interest.

Figure 1: Performing multi-dimensional VCA extraction



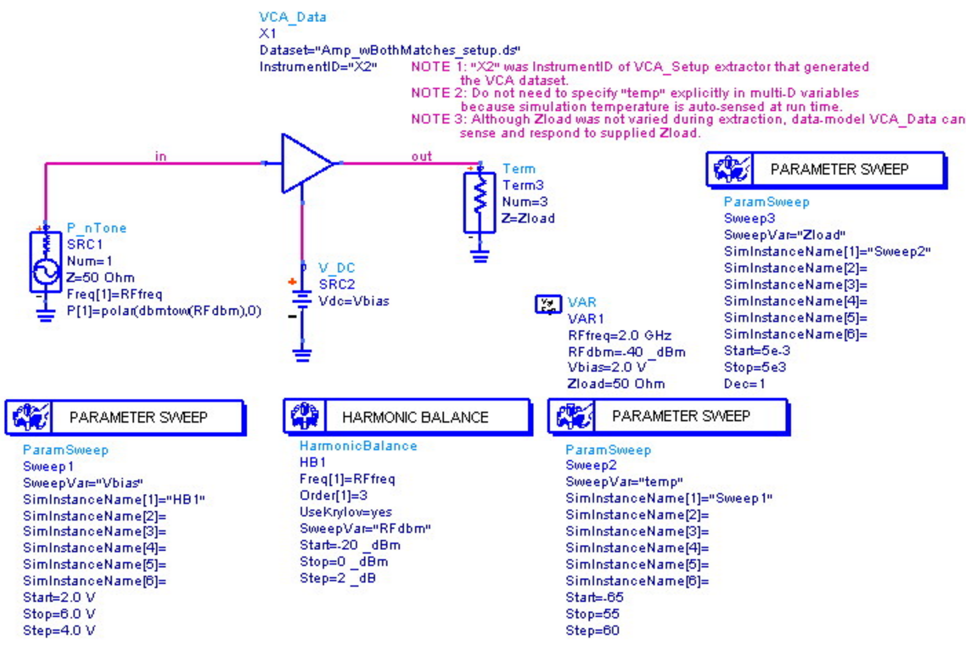
ParamSweep
Sweep1
SweepVar="temp"
SimInstanceName[1]="X2.Vcontrol_Sweep"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=-65
Stop=55
Step=30

VCA_Setup
X2
Freq=2.0 GHz
Order=10
Pin_Start=-48.0 _dBm
Pin_Stop=4.0 _dBm
Pin_Step=4.0 _dB
Vcontrol_Start=2.0 V
Vcontrol_Stop=6.0 V
Vcontrol_Step=0.5 V

One-Tone Harmonic Balance Analysis

Behavioral simulation using the multi-dimensional VCA dataset is shown in **Amp_wBothMatches_behavioral_1tone**. Note that although the dataset contains a sweep of temperature, it is not necessary to explicitly list it in the multi-dimensional parameters *iVar1* of the VCA_Data model because temperature is sensed internally. In this example a sweep of output impedance Zload is applied although the VCA dataset was extracted using a load impedance of 50 Ohm. A corresponding circuit level simulation is done in **Amp_wBothMatches_circuit_1tone**. The results are compared in **Amp_wBothMatches_behavioral_1tone.dds** as shown in the Analysis section below.

Figure 2: Using VCA_Data for single tone HB analysis



```
VCA_Data
X1
Dataset="Amp_wBothMatches_setup.ds"
InstrumentID="X2"

NOTE 1: "X2" was InstrumentID of VCA_Setup extractor that generated
        the VCA dataset.
NOTE 2: Do not need to specify "temp" explicitly in multi-D variables
        because simulation temperature is auto-sensed at run time.
NOTE 3: Although Zload was not varied during extraction, data-model VCA_Data can
        sense and respond to supplied Zload.

Term
Term3
Num=3
Z=Zload

PARAMETER SWEEP
ParamSweep
Sweep3
SweepVar="Zload"
SimInstanceName[1]="Sweep2"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=5e-3
Stop=5e3
Dec=1

P_nTone
SRC1
Num=1
Z=50 Ohm
Freq[1]=RFfreq
P[1]=polar(dbmtow(RFdbm),0)

V_DC
SRC2
Vdc=Vbias

VAR
VAR1
RFfreq=2.0 GHz
RFdbm=-40 _dBm
Vbias=2.0 V
Zload=50 Ohm

PARAMETER SWEEP
ParamSweep
Sweep1
SweepVar="Vbias"
SimInstanceName[1]="HB1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=2.0 V
Stop=6.0 V
Step=4.0 V

HARMONIC BALANCE
HarmonicBalance
HB1
Freq[1]=RFfreq
Order[1]=3
UseKrylov=yes
SweepVar="RFdbm"
Start=-20 _dBm
Stop=0 _dBm
Step=2 _dB

PARAMETER SWEEP
ParamSweep
Sweep2
SweepVar="temp"
SimInstanceName[1]="Sweep1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=-65
Stop=55
Step=60
```

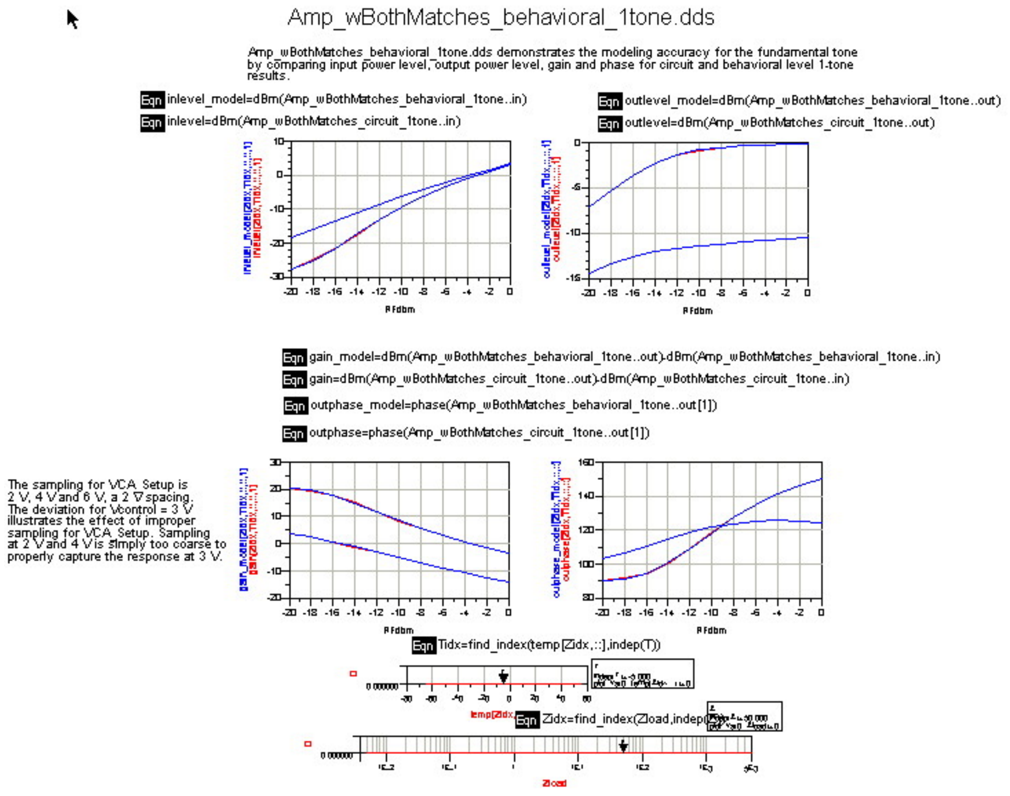
Note analysis frequency should be same as extraction frequency.

Circuit Envelope Simulation
Cicuit envelope simulation is done using the VCA dataset in **Amp_wBothMatches_behavioral_env**. Note that although the dataset contains a sweep of temperature, it is not necessary to explicitly list it in the multi-dimensional parameters *iVar1* of the VCA_Data model because temperature is sensed internally. A corresponding circuit level simulation is done in **Amp_wBothMatches_circuit_env**. The results are compared in **Amp_wBothMatches_behavioral_env.dds** as shown in the Analysis section below.

Analysis

One-Tone Harmonic Balance Analysis
Comparison of the VCA behavioral model against the circuit level model for single tone Harmonic Balance Simulation is provided in **Amp_wBothMatches_behavioral_1tone.dds**. The circuit responses in red and behavioral ones in blue. Using the Zload slider it is possible to interactively validate how the behavioral model tracks the circuit level model even at load impedances that differ significantly from the extraction load of 50 Ohms.

Figure 3: Comparison of circuit and behavioral model responses to single tone HB analysis



Amp_wBothMatches_behavioral_1tone.dds

Amp_wBothMatches_behavioral_1tone.dds demonstrates the modeling accuracy for the fundamental tone by comparing input power level, output power level, gain and phase for circuit and behavioral level 1-tone results.

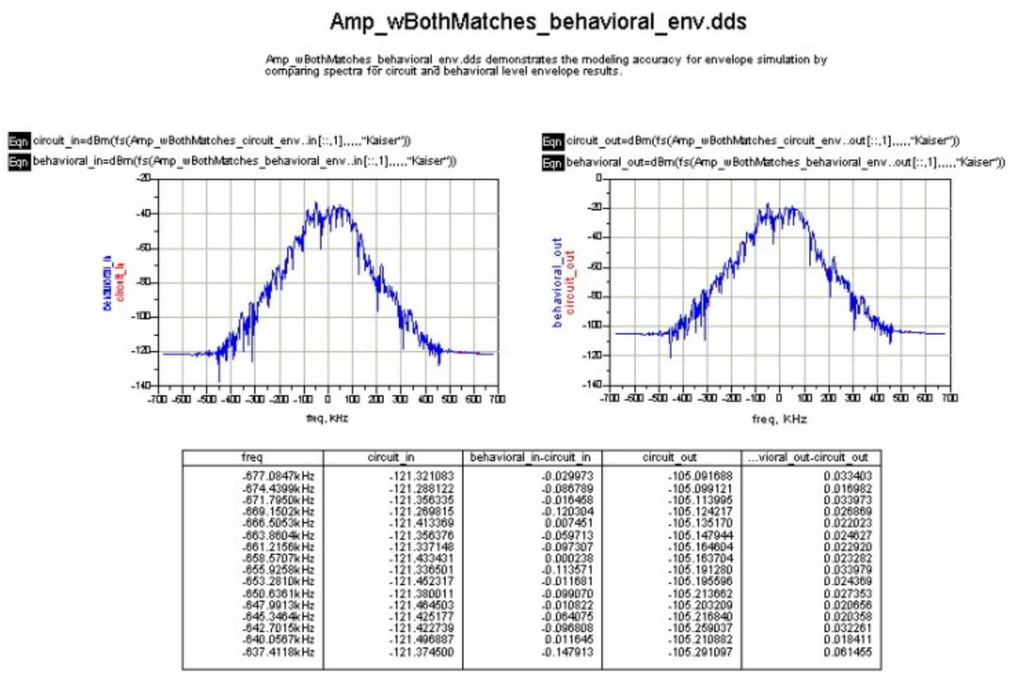
```
Eqn inlevel_model=dBm(Amp_wBothMatches_behavioral_1tone..in)
Eqn inlevel=dBm(Amp_wBothMatches_circuit_1tone..in)
Eqn outlevel_model=dBm(Amp_wBothMatches_behavioral_1tone..out)
Eqn outlevel=dBm(Amp_wBothMatches_circuit_1tone..out)
Eqn gain_model=dBm(Amp_wBothMatches_behavioral_1tone..out)-dBm(Amp_wBothMatches_behavioral_1tone..in)
Eqn gain=dBm(Amp_wBothMatches_circuit_1tone..out)-dBm(Amp_wBothMatches_circuit_1tone..in)
Eqn outphase_model=phase(Amp_wBothMatches_behavioral_1tone..out[1])
Eqn outphase=phase(Amp_wBothMatches_circuit_1tone..out[1])
Eqn Tidx=find_index(temp[Zidx,::],indep(T))
Eqn Zidx=find_index(Zload,indep(Z))
```

The sampling for VCA_Setup is 2 V, 4 V and 6 V, a 2 V spacing. The deviation for Vcontrol = 3 V illustrates the effect of improper sampling for VCA_Setup. Sampling at 2 V and 4 V is simply too coarse to properly capture the response at 3 V.

inlevel_model[Zidx,Tidx,::,1]
inlevel[Zidx,Tidx,::,1]
outlevel_model[Zidx,Tidx,::,1]
outlevel[Zidx,Tidx,::,1]
gain_model[Zidx,Tidx,::,1]
gain[Zidx,Tidx,::,1]
outphase_model[Zidx,Tidx,::]
outphase[Zidx,Tidx,::]
RFdbm
temp[Zidx,
Zload

Circuit Envelope Analysis
Comparison of the VCA behavioral model against the circuit level model for single tone Harmonic Balance Simulation is provided in **Amp_wBothMatches_behavioral_env.dds**. The circuit responses in red and behavioral ones in blue.

Figure 4: Comparison of circuit and behavioral model responses to CE analysis



Amp_wBothMatches_behavioral_env.dds

Amp_wBothMatches_behavioral_env.dds demonstrates the modeling accuracy for envelope simulation by comparing spectra for circuit and behavioral level envelope results.

```
Eqn circuit_in=dBm(fs(Amp_wBothMatches_circuit_env..in[::,1],....,"Kaiser"))
Eqn behavioral_in=dBm(fs(Amp_wBothMatches_behavioral_env..in[::,1],....,"Kaiser"))
Eqn circuit_out=dBm(fs(Amp_wBothMatches_circuit_env..out[::,1],....,"Kaiser"))
Eqn behavioral_out=dBm(fs(Amp_wBothMatches_behavioral_env..out[::,1],....,"Kaiser"))
```

freq	circuit_in	behavioral_in-circuit_in	circuit_out	...vioral_out-circuit_out
-677.0847kHz	-121.321083	-0.029973	-105.091688	0.033403
-674.4399kHz	-121.288122	-0.086789	-105.099121	0.016982
-671.7950kHz	-121.356335	-0.016458	-105.113995	0.033973
-669.1502kHz	-121.269815	-0.120304	-105.124217	0.026869
-666.5053kHz	-121.413369	0.007451	-105.135170	0.022023
-663.8604kHz	-121.356376	-0.059713	-105.147944	0.024627
-661.2156kHz	-121.337148	-0.097307	-105.164604	0.022920
-658.5707kHz	-121.433431	0.000238	-105.163704	0.023282
-655.9258kHz	-121.336501	-0.113571	-105.191280	0.033979
-653.2810kHz	-121.452317	-0.011681	-105.195596	0.024369
-650.6361kHz	-121.380011	-0.099070	-105.213662	0.027353
-647.9913kHz	-121.464503	-0.010822	-105.203209	0.020656
-645.3464kHz	-121.425177	-0.064075	-105.216840	0.020358
-642.7015kHz	-121.422739	-0.096808	-105.259037	0.032261
-640.0567kHz	-121.496887	0.011646	-105.210882	0.018411
-637.4118kHz	-121.374500	-0.147913	-105.291097	0.061455

Notes

- The VCA_Data component may be used to model narrow band low-noise or power amplifiers. The data profile is extracted only at one frequency with the understanding that this frequency of use is known implicitly prior to behavioral simulation and used in that process.

- It is possible to include sweeps of additional user-specified parameters above the basic control voltage and power sweeps necessary in VCA dataset.
- VCA_Data accurately models DC power drawn from the bias source based on bias voltage and supplied RF power at amplifier input. This model can therefore be used for PAE estimation of RF amplifiers.
- VCA_Data is able to provide accurate output voltage and current for a wide range of varying load conditions.

Communication Systems Examples

Examples of how cosimulation between DSP and RF/Analog tools can be used to create real circuits and electromagnetic propagation models to enhance system-level performance.

- *Adaptive Equalizer with Training Sequence* (examples)
- *Bit-Error-Rate Estimation* (examples)
- *BlueTooth Example System* (examples)
- *CATV Example System* (examples)
- *Convolution Coder - Viterbi Decoder BER example* (examples)
- *Cosimulation of a Rectifier Circuit* (examples)
- *Cosimulation of Baseband Sine Wave and Amplifier Circuit* (examples)
- *Cosimulation of I-Q Signals with RF Subnetwork* (examples)
- *Cosimulation of QPSK and RF Downconverting Mixer* (examples)
- *Cosimulation of QPSK and RF Upconverting Mixer* (examples)
- *Cosimulation of QPSK Modulation and Behavioral Model Amplifier* (examples)
- *Delta Sigma Converter* (examples)
- *GSM System - Basic Simulation* (examples)
- *IS-95 CDMA Filter* (examples)
- *IS-95 CDMA Forward-Channel Test Modulator* (examples)
- *Linear Budget Analysis* (examples)
- *LSF (Load Sharing Facility) usage with BER simulation* (examples)
- *Multi-Channel Nonlinear Budget Analysis* (examples)
- *OFDM Modulation* (examples)
- *Prototype 8-VSB Modulator* (examples)
- *QAM System Co-Simulation Including LNA Noise* (examples)
- *RFID Reader and Tag Basic Mode 1 Link* (examples)
- *RFID Transponder and Antenna Design using Advanced Model Composer* (examples)
- *Simulation of Spurious Signals* (examples)
- *SINAD Measurements* (examples)
- *Subband Speech Codec* (examples)
- *Test Benches for Evaluating PDC-TDMA Receivers* (examples)
- *Various Examples on RF System-Level Simulations* (examples)
- *Wideband CDMA Test Modulator* (examples)
- *Wireless MAN 802.16d Transmitter Test Project* (examples)

Adaptive Equalizer with Training Sequence

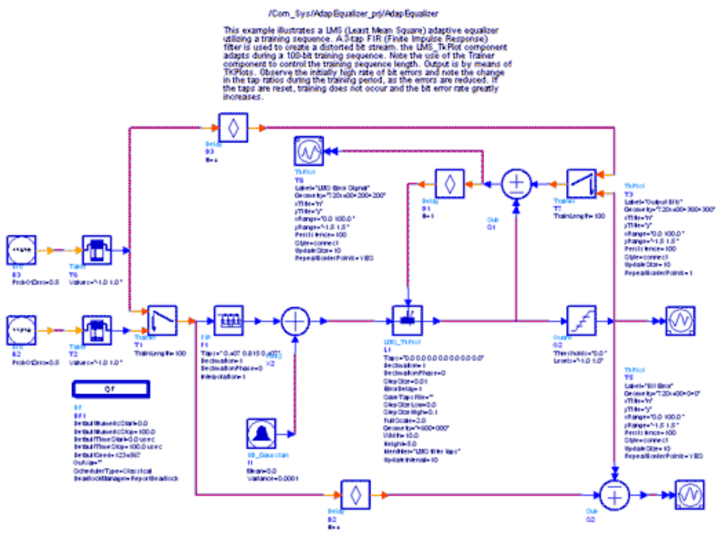
Location: $HPEESOF_DIR/examples/Com_Sys/AdapEqualizer_wrk

Objective

This example illustrates a Least Mean Square (LMS) adaptive equalizer utilizing a training sequence.

Setup

The design applies a linear adaptive equalizer to a band-limited Additive White Gaussian Noise (AWGN) channel. A 4-tap Finite Impulse Response (FIR) filter is used to create a distorted bit stream. During the short training period, a tap adjustment is performed, the LMS_TkPlot component adapts during a 100-bit training sequence. A decision-directed mode of operation is employed for the continuous adjustment of taps. Note the use of the Trainer component to control the training sequence length. Output is by means of TKPlots.


/Com_Sys/AdapEqualizer_prj/AdapEqualizer

This example illustrates a LMS (Least Mean Square) adaptive equalizer utilizing a training sequence. A 3-tap FIR (Finite Impulse Response) filter is used to create a distorted bit stream. the LMS_TkPlot component adapts during a 100-bit training sequence. Note the use of the Trainer component to control the training sequence length. Output is by means of TkPlots. Observe the initially high rate of bit errors and note the change in the tap ratios during the training period, as the errors are reduced. If the taps are reset, training does not occur and the bit error rate greatly increases.

Analysis

Figure 1: LMS error signal before training

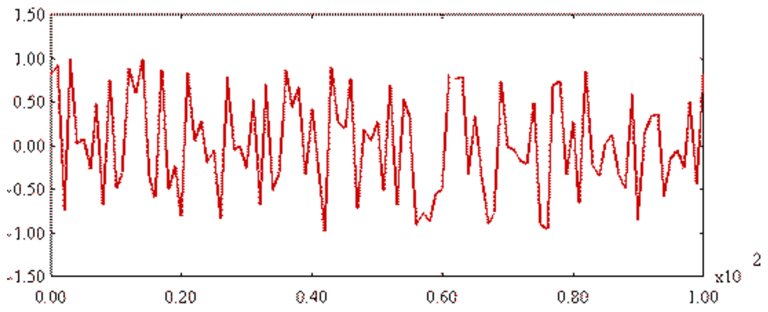


Figure 2: Bit errors before training

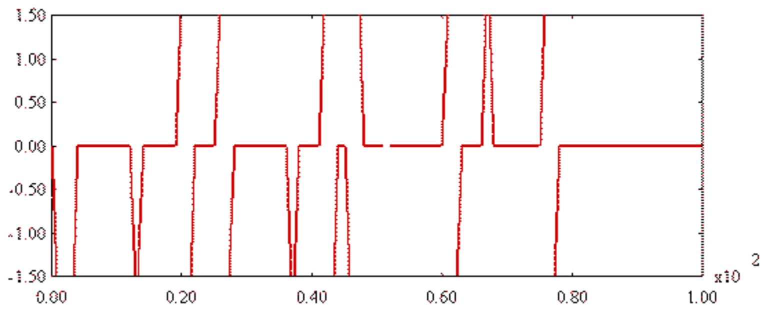


Figure 3: LMS taps before training

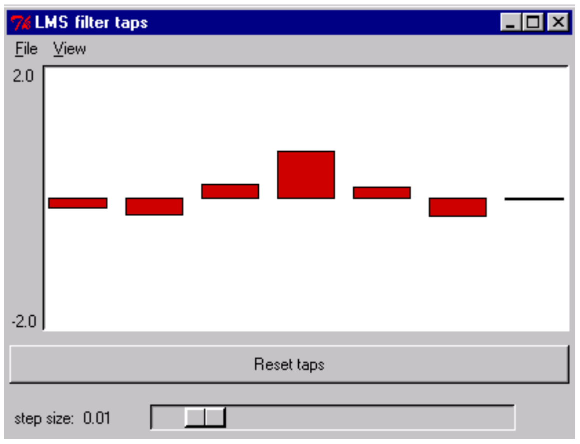
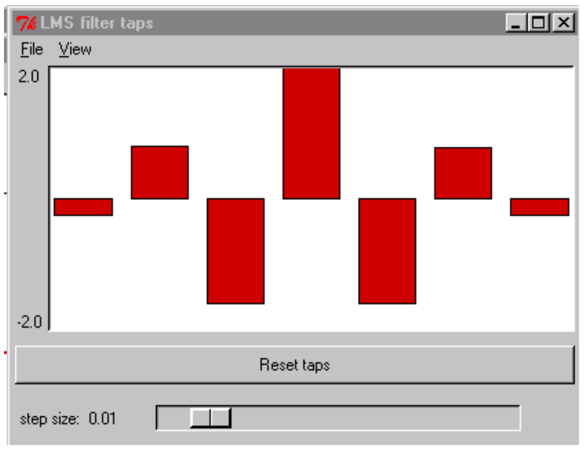


Figure 4: LMS taps after training



Notes

- Observe the initially high rate of bit errors and note the change in the tap ratios during the training period, as the errors are reduced. If the taps are reset, training does not occur and the bit error rate greatly increases.

Bit-Error-Rate Estimation

Location: $HPEESOF_DIR/examples/Com_Sys/BER_wrk

Objective

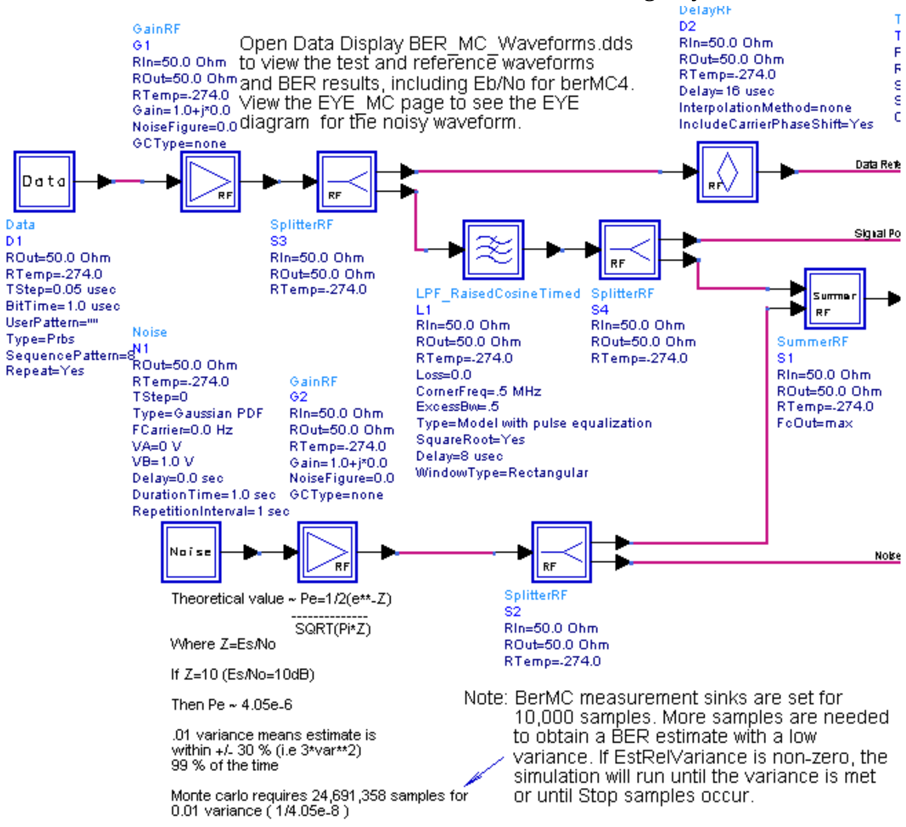
This example illustrates two basic Bit Error Rate (BER) measurements, one uses importance sampling to reduce the number of samples and the other uses a Monte Carlo BER measurement. Simple baseband signaling with Nyquist filtering is used, and theoretical calculations are included for comparison with simulation results.

Setup

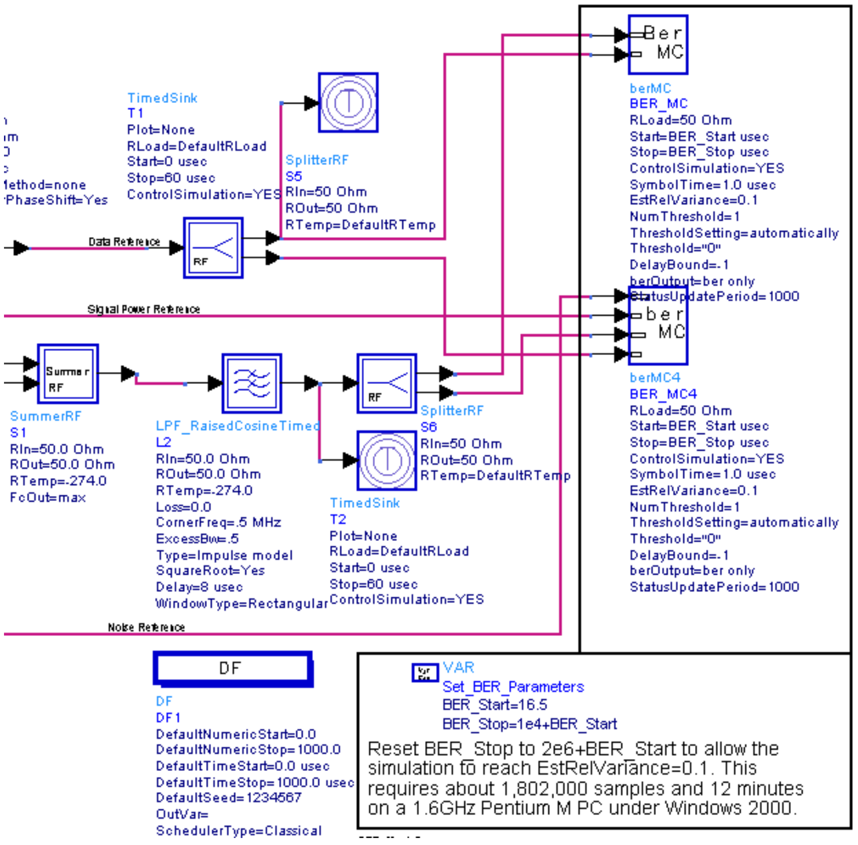
1. "BER_IIS" uses importance sampling to greatly reduce the number of samples required to estimate BER, even for very low BER systems. "BER_IS_Sweep" adds swept Noise value to create a "waterfall" curve.
2. "BER_MonteC" is a Monte Carlo BER measurement. A larger number of samples may be required. This measurement is more general and is independent of the modulation or signaling method.

left half of schematic

BER_MonteC: Monte Carlo BER Estimation



right half of schematic



Analysis

Figure 1: Estimated BER using Monte Carlo, Relative Variance = 0.1

EsOverNo	BER_MC4	BER_MC[0]
10.008	5.548E-6	5.548E-6

Figure 2: Output signal versus time

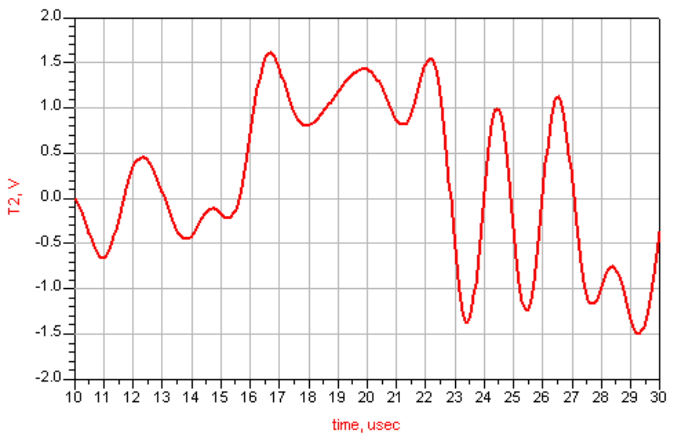
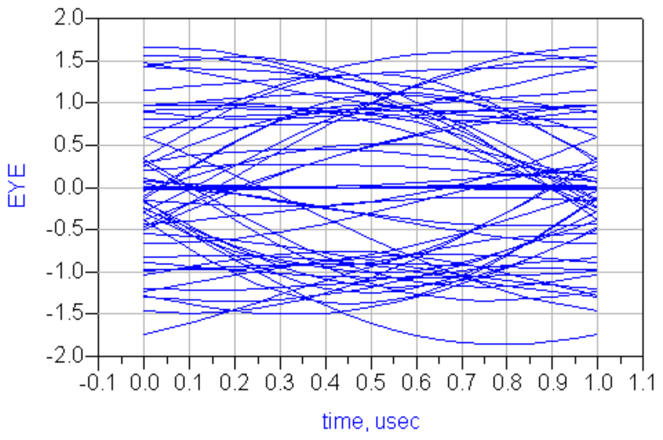


Figure 3: Eye diagram of the output signal



Notes

- Importance Sampling relies on certain assumptions to be made concerning the system under test. Please see *Sinks* (sinks) for more information.

- For BER_MonteC, the measurement sinks are set for 10,000 samples for fast simulation. More samples are needed to obtain a low variance BER estimate using BER_MC4. If EstRelVariance is non-zero, e.g. set to 0.1, the simulation will run until the variance requirement is met or until Stop samples occur. For EstRelVariance=0.1 and BER_MC4 Stop=(2e6 + 16.5) usec, the simulation will terminate at 1,802,469 symbols and a variance of 0.1 . This simulation requires about 12 minutes to complete (1.6 GHz Pentium M under Windows 2000). A datset created with these settings, BER_MonteC_VAR0pt1.ds, is available in the examples /data directory.

BlueTooth Example System

Location: $HPEESOF_DIR/examples/Com_Sys/BlueTooth_wrk

Objective

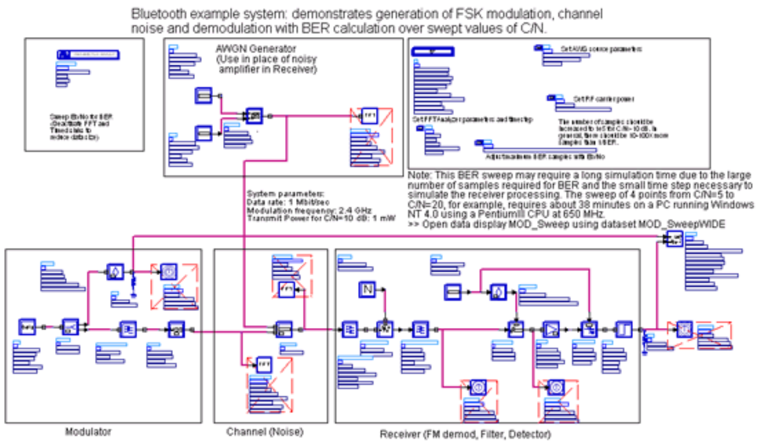
This example demonstrates a simple BlueTooth system. The system includes a Gaussian Frequency-Shift-Keyed (FSK) transmitter, an Additive White Gaussian Noise (AWGN) source, and a candidate receiver.

Setup

1. "MOD" is the basic air interface model, including spectrum and waveform output (use MOD_NoSweep to display data).
2. "MOD_BERSweep" is the same model as "MOD", but the Eb/No is swept and BER is

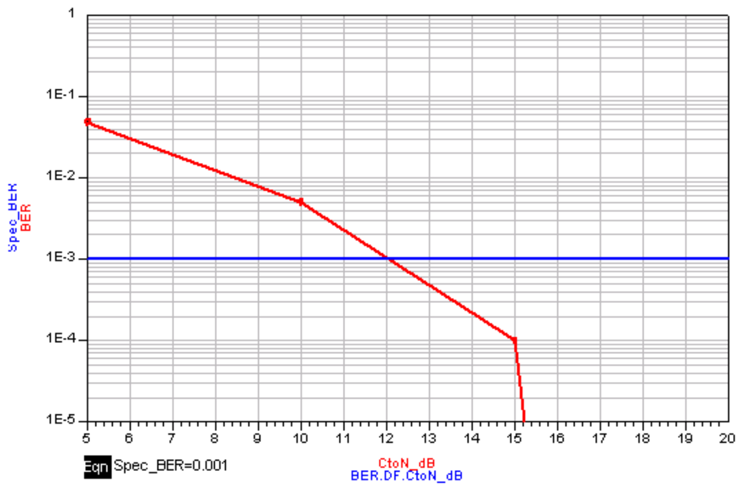
measured (use MOD_Sweep to display saved data; this simulation takes several hours to run).

3. "MOD_EYE.dds" shows the EYE diagram for GFSK modulation. "bpf_impulse.dds" shows the impulse response of noise filter. "lpf_impulse.dds" shows the impulse response of receiver filter.


Bluetooth example system: demonstrates generation of FSK modulation, channel noise and demodulation with BER calculation over swept values of C/N.
AWGN Generator (Use in place of noisy amplifier in Receiver)
System parameters: Data rate: 1 Mbit/sec Modulation frequency: 2.4 GHz Transmit Power for C/N=10 dB: 1 mW
Note: This BER sweep may require a long simulation time due to the large number of samples required for BER and the small time step necessary to simulate the receiver processing. The sweep of 4 points from C/N=5 to C/N=20, for example, requires about 38 minutes on a PC running Windows NT 4.0 using a PentiumIII CPU at 650 MHz.
>> Open data display MOD_Sweep using dataset MOD_SweepWIDE
Modulator
Channel (Noise)
Receiver (FM demod, Filter, Detector)

Analysis

Analysis plots



CtoN_dB	BER[0]
5.0000	0.0480
10.0000	0.0050
15.0000	0.0001
20.0000	9.9990E-25

CATV Example System

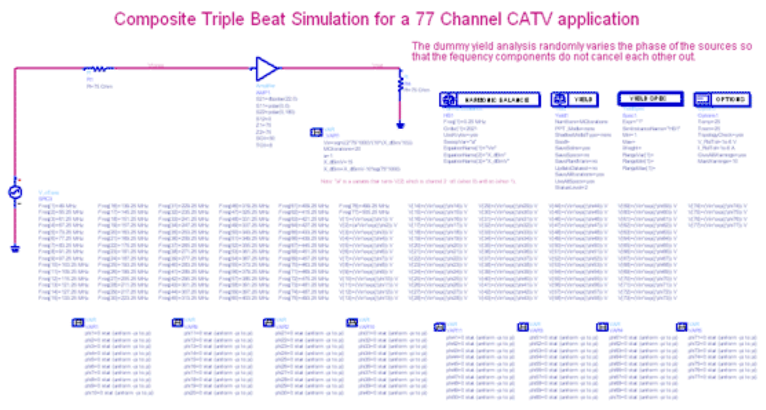
Location: $HPEESOF_DIR/examples/Com_Sys/CATV_wrk

Objective

This workspace demonstrates how ADS can be used to measure the impact of a broadband amplifier in a CATV system with examples of how to compute the Composite Triple Beat and how to predict the Composite Second Order of a broadband amplifier. It includes NTSC and PAL system video source designs to simulate video circuits and transmission.

Setup

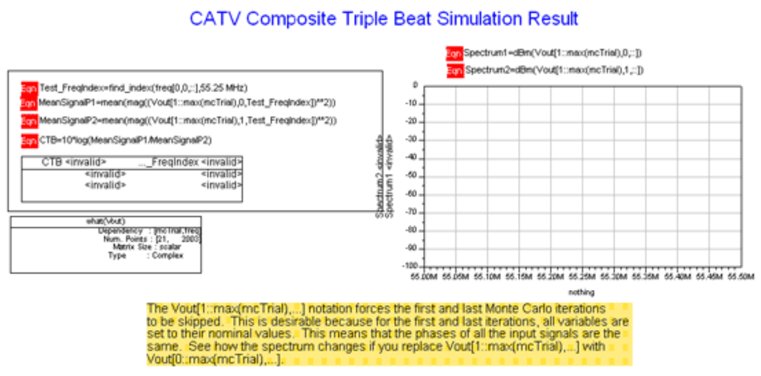
1. "CTB_Test" computes the Composite Triple Beat produced by a broadband amplifier used in a 77 channel CATV system.
2. "CSO_Test" predicts the Composite Second Order caused by a broadband amplifier in a CATV system.
3. "Video_Test" is an amplifier test circuit.
4. NTSC Sources:"NTSC_Composite", "NTSC_HorizSynch", "NTSC_ModRamp", and "NTSC_MultiBurst" create the NTSC source signals for a composite test, horizontal sync interval, modulated ramp, and multiburst test, respectively.
5. PAL Sources:"PAL_Composite", "PAL_HorizSynch", "PAL_ModRamp", and "PAL_MultiBurst" create the PAL source signals for a composite test, horizontal sync interval, modulated ramp, and multiburst test, respectively.
6. "Video_Test.dds" plots the video waveform at the input and output of an amplifier.
7. "CSO_Test.dds" plots the Composite Second Order.
8. "CTB_Test.dds" plots the Composite Triple Beat.


Composite Triple Beat Simulation for a 77 Channel CATV application

The dummy yield analysis randomly varies the phase of the sources so that the fequency components do not cancel each other out.

Analysis

Composite Triple Beat plot


CATV Composite Triple Beat Simulation Result

```
Eqn Spectrum1=dBm(Vout[1::max(mcTrial),0,::])
Eqn Spectrum2=dBm(Vout[1::max(mcTrial),1,::])

Eqn Test_FreqIndex=find_index(freq[0,0,::],55.25 MHz)
Eqn MeanSignalP1=mean(mag((Vout[1::max(mcTrial),0,Test_FreqIndex])**2))
Eqn MeanSignalP2=mean(mag((Vout[1::max(mcTrial),1,Test_FreqIndex])**2))
Eqn CTB=10*log(MeanSignalP1/MeanSignalP2)
```

CTB <invalid>	..._FreqIndex <invalid>
<invalid>	<invalid>
<invalid>	<invalid>

```
what(Vout)
Dependency  : [mcTrial,freq]
Num. Points : [21,     2003]
Matrix Size : scalar
Type        : Complex
```

The Vout[1::max(mcTrial),...] notation forces the first and last Monte Carlo iterations to be skipped. This is desirable because for the first and last iterations, all variables are set to their nominal values. This means that the phases of all the input signals are the same. See how the spectrum changes if you replace Vout[1::max(mcTrial),...] with Vout[0::max(mcTrial),...].

Convolution Coder-Viterbi Decoder BER example

Location: $HPEESOF_DIR/examples/Com_Sys/Viterbi_wrk

Objective

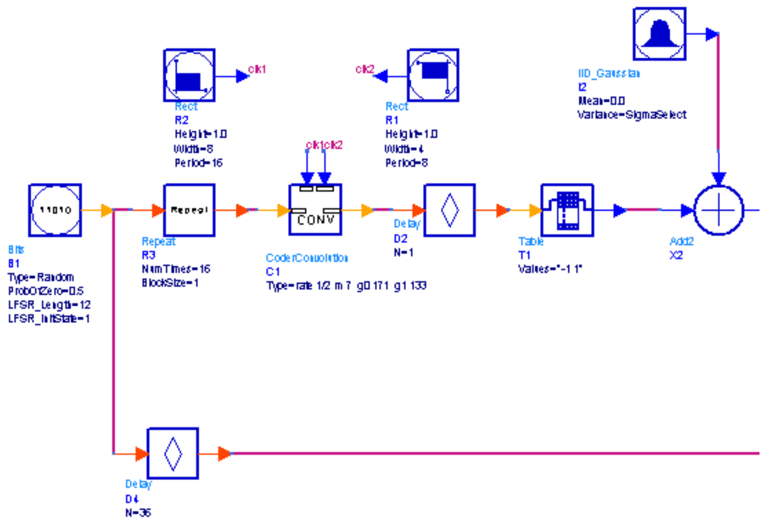
This example provides a simulation example for the use of Convolutional Coder and Viterbi decoder. Swept noise is added to the channel to vary Eb\No.

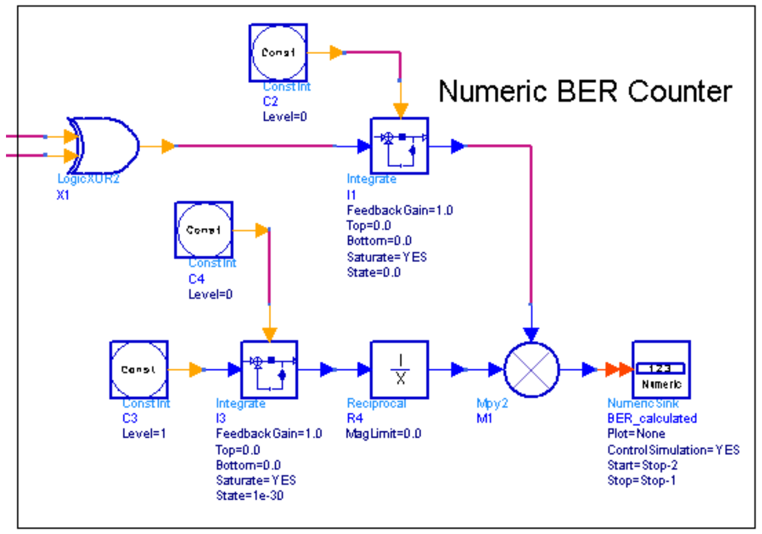
Setup

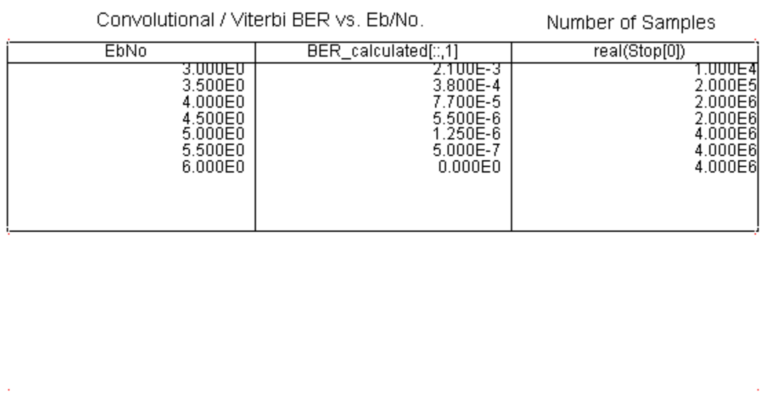
Convolutional Coder-Viterbi Decoder "Soft" BER Example

Uses rate=1/2 code. Swept noise is added to the channel to vary Eb/No. Higher values of Eb/No require significantly more samples for an accurate estimate of the BER; the value of "Stop" is set by a "PWL" or piecewise-linear equation.

Analysis


clk1
clk2
Rect
R2
Height=1.0
Width=8
Period=16
Rect
R1
Height=1.0
Width=4
Period=8
IID_Gaussian
I2
Mean=0.0
Variance=SigmaSelect
clk1clk2
11010
Repeat
CONV
Bits
B1
Type=Random
ProbOfZero=0.5
LFSR_Length=12
LFSR_InitState=1
Repeat
R3
NumTimes=16
BlockSize=1
CoderConvolution
C1
Type=rate 1/2 m 7 g0 171 g1 133
Delay
D2
N=1
Table
T1
Values="-1 1"
Add2
X2
Delay
D4
N=36


Numeric BER Counter
Const
ConstInt
C2
Level=0
LogicXOR2
X1
Integrate
I1
FeedbackGain=1.0
Top=0.0
Bottom=0.0
Saturate=YES
State=0.0
Const
ConstInt
C4
Level=0
Const
ConstInt
C3
Level=1
Integrate
I3
FeedbackGain=1.0
Top=0.0
Bottom=0.0
Saturate=YES
State=1e-30
Reciprocal
R4
MagLimit=0.0
Mpy2
M1
123
Numeric
NumericSink
BER_calculated
Plot=None
ControlSimulation=YES
Start=Stop-2
Stop=Stop-1



Convolutional / Viterbi BER vs. Eb/No.

Number of Samples

EbNo	BER_calculated[::,1]	real(Stop[0])
3.000E0	2.100E-3	1.000E4
3.500E0	3.800E-4	2.000E5
4.000E0	7.700E-5	2.000E6
4.500E0	5.500E-6	2.000E6
5.000E0	1.250E-6	4.000E6
5.500E0	5.000E-7	4.000E6
6.000E0	0.000E0	4.000E6

Cosimulation of a Rectifier Circuit

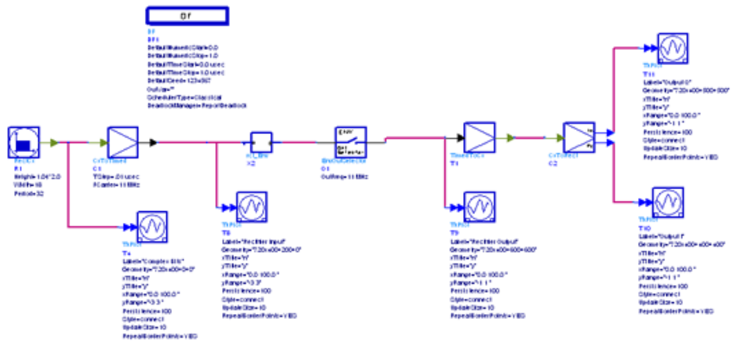
Location: $HPEESOF_DIR/examples/Com_Sys/RectifierCosim_wrk

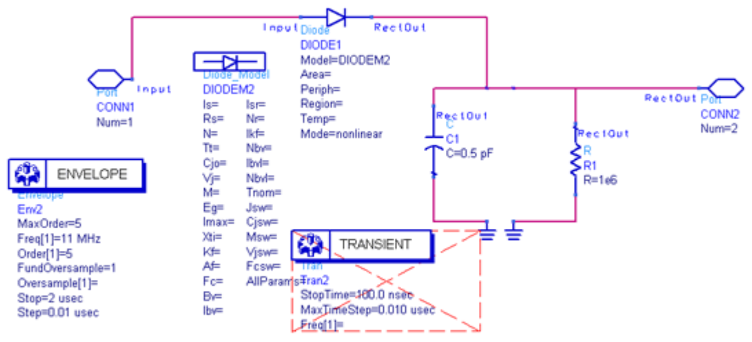
Objective

This example illustrates a simple cosimulation of a modulated RF signal with a rectifier circuit using circuit envelope.

Setup

A rectangular pulse train (18usec/32usec) is created (using Agilent Ptolemy) as a complex signal, placed onto a RF carrier (11MHz), and directed into a simple diode rectifier circuit. The circuit envelope cosimulation output may be displayed as the real or imaginary carrier voltage versus time.





Analysis

Figure 1: Complex pulsed input signal

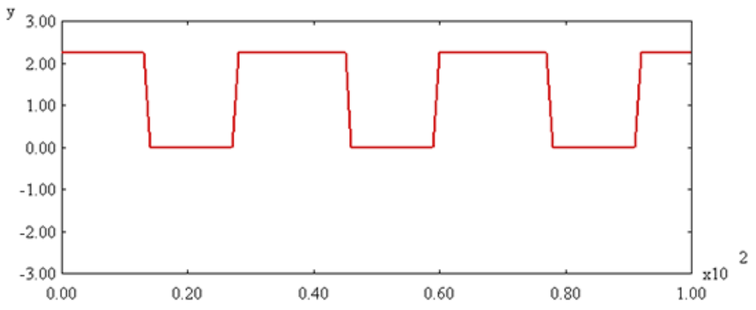


Figure 2: Rectifier input envelope

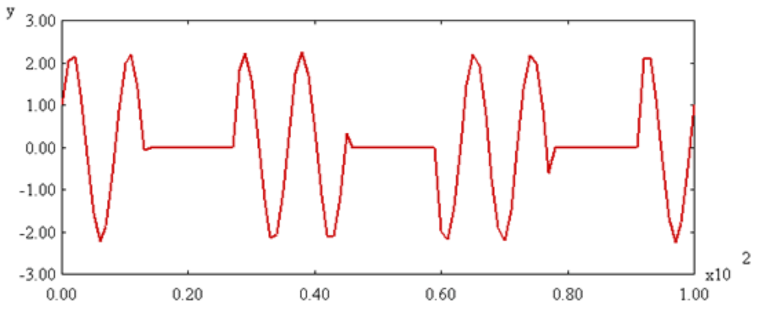


Figure 3: Output I data

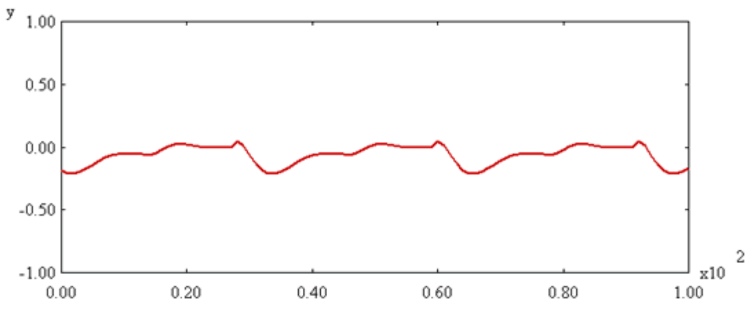
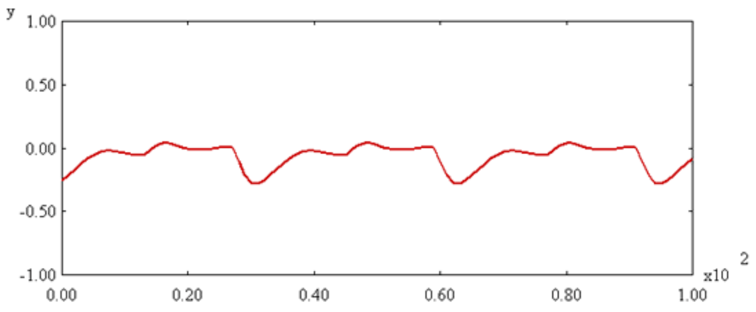


Figure 4: Output Q data



Notes

- In "EnvOutSelector" component, the parameter OutFreq is set to type Bandpass and to the carrier frequency of 11 MHz; this means that the Circuit Envelope output is an RF signal having only the modulation envelope but with an associated carrier.
- If set to All(pass), the output is the composite time-varying signal; that is, it contains both modulation and carrier. This requires that the sample rate be high enough to properly sample the carrier.
- Setting OutFreq=Lowpass (or 0 Hz) will produce only time-varying DC.
- Changing the RC values in the circuit subnetwork rct1 would change the time constant and hence affect the fall-off of the pulses.
- The Transient controller may be used instead of the Envelope controller; the EnvOutSelector may remain as it has no effect when the Transient simulator is used.

In the Transient case, the output is similar to the Allpass setting used with Circuit Envelope.

Cosimulation of Baseband Sine Wave and Amplifier Circuit

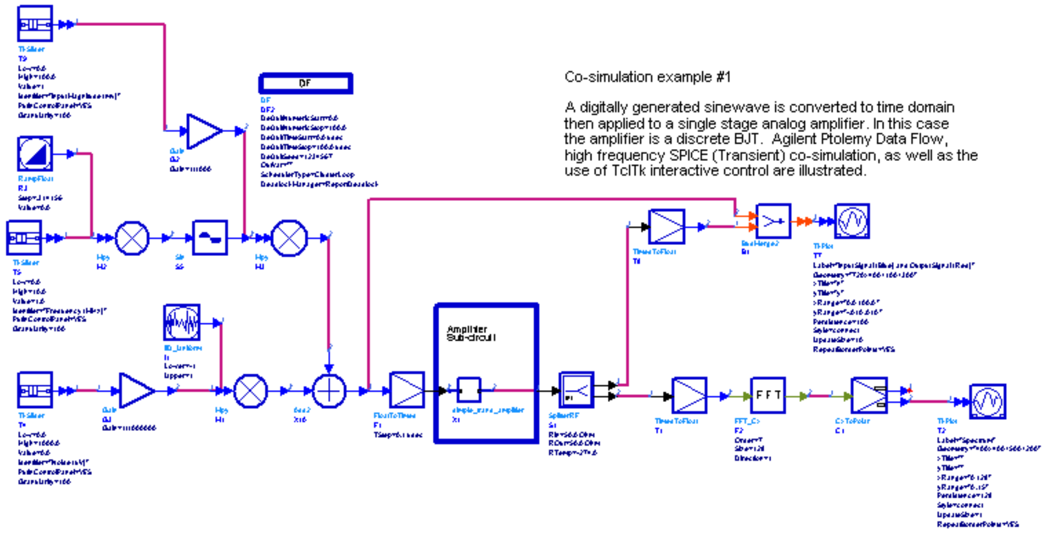
Location: $HPEESOF_DIR/examples/Com_Sys/Co_Sim_wrk/Co_sim_1

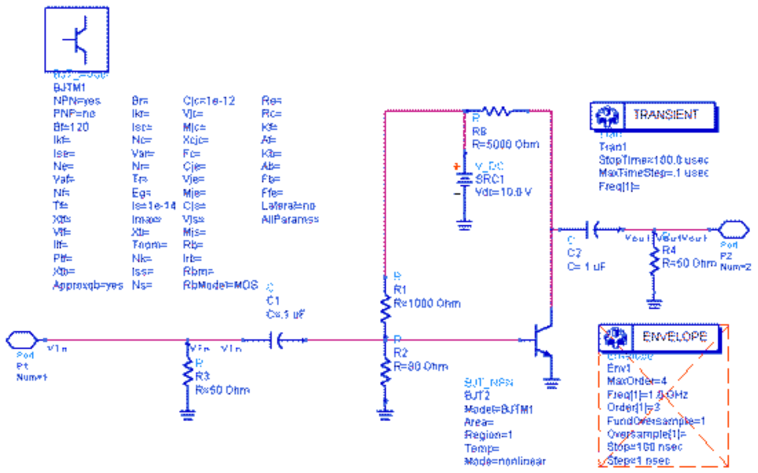
Objective

This example illustrates the basic applications of cosimulation between Agilent Ptolemy at the system level and Transient simulation at the circuit level.

Setup

"Co_sim_1" Sine wave input with BJT amplifier sub-circuit under Transient simulation. A digitally generated sinewave is converted to time domain then applied to a single stage analog amplifier. In this case the amplifier is a discrete BJT. The use of TclTk interactive control is also illustrated.


Co-simulation example #1
A digitally generated sinewave is converted to time domain then applied to a single stage analog amplifier. In this case the amplifier is a discrete BJT. Agilent Ptolemy Data Flow, high frequency SPICE (Transient) co-simulation, as well as the use of TclTk interactive control are illustrated.
DF
Amplifier Sub-circuit



Analysis

Figure 1: Input signal

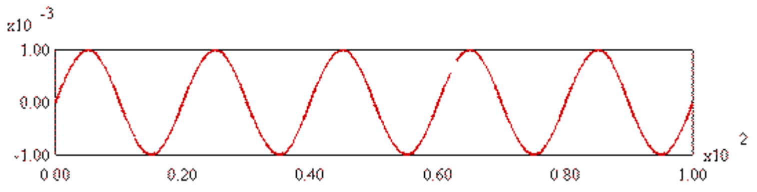


Figure 2: Output signal

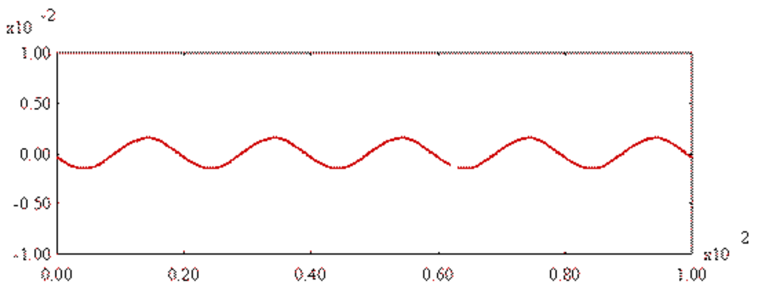


Figure 3: Output spectrum

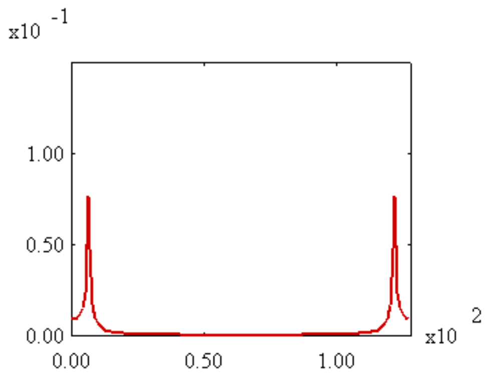
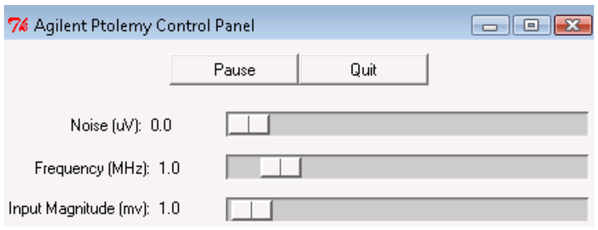


Figure 4: Real-time slider bar for noise control



Noise (uV): 0.0
Frequency (MHz): 1.0
Input Magnitude (mv): 1.0

Cosimulation of I/Q Signals with RF Subnetwork

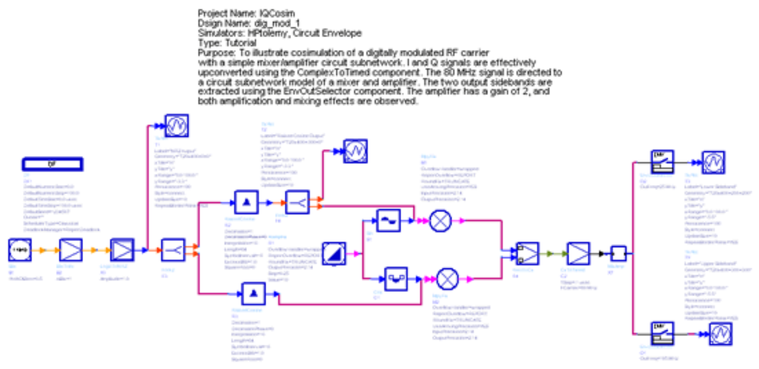
Location: $HPESSOF_DIR/examples/Com_Sys/IQCosim_wrk/dig_mod_1

Objective

This Example illustrates the cosimulation of a digitally modulated RF carrier with a simple mixer/amplifier circuit subnetwork. The ability of ADS to easily accommodate modulated signals in a combined DSP and circuit simulation environment is demonstrated.

Setup

I and Q signals are effectively upconverted using the ComplexToTimed component. The 80 MHz signal is directed to a circuit subnetwork model of a mixer and amplifier. The LO signal is 105MHz. The two output sidebands are extracted using the EnvOutSelector component. The amplifier has a gain of 2, and both amplification and mixing effects are observed.



Project Name: IQCosim
Dsign Name: dig_mod_1
Simulators: HPtolemy, Circuit Envelope
Type: Tutorial
Purpose: To illustrate cosimulation of a digitally modulated RF carrier
with a simple mixer/amplifier circuit subnetwork. I and Q signals are effectively
upconverted using the ComplexToTimed component. The 80 MHz signal is directed to
a circuit subnetwork model of a mixer and amplifier. The two output sidebands are
extracted using the EnvOutSelector component. The amplifier has a gain of 2, and
both amplification and mixing effects are observed.

Analysis

Figure 1: NRZ output

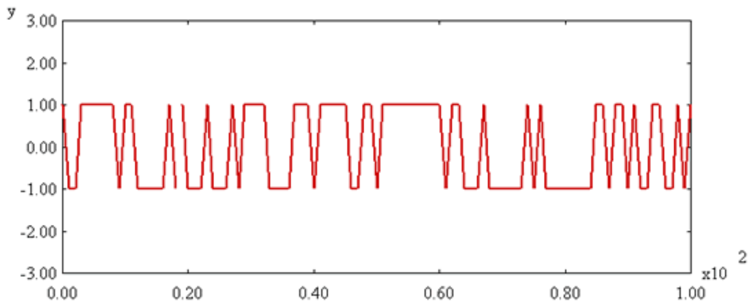


Figure 2: Raised-Cosine output

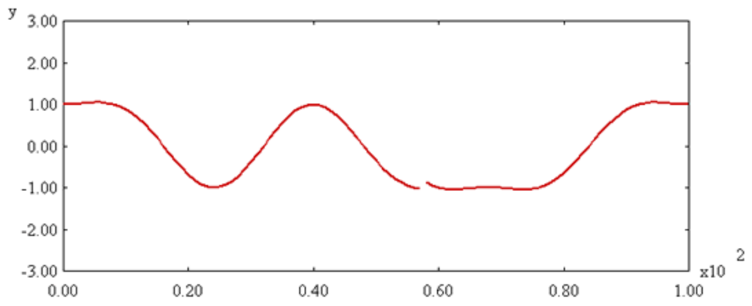


Figure 3: Lower sideband

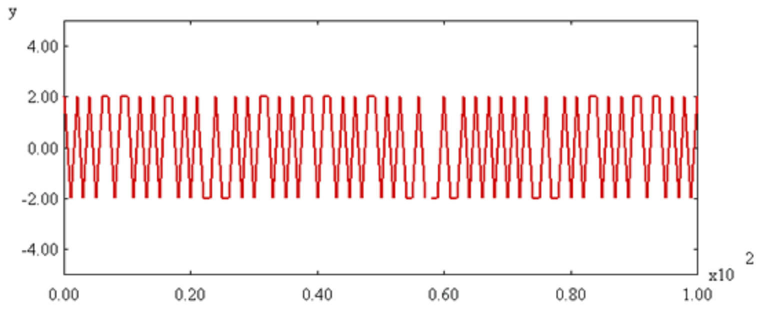
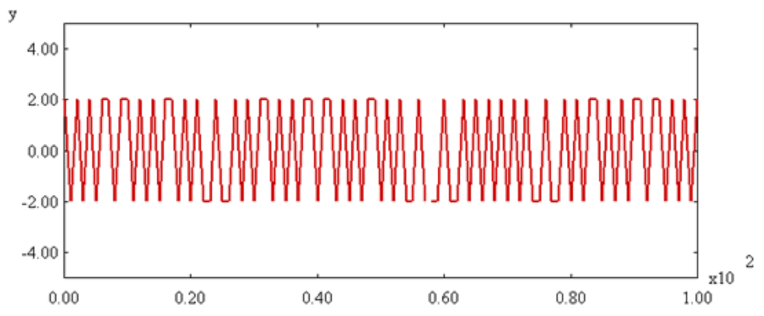


Figure 4: Upper sideband



Cosimulation of QPSK and RF Downconverting Mixer

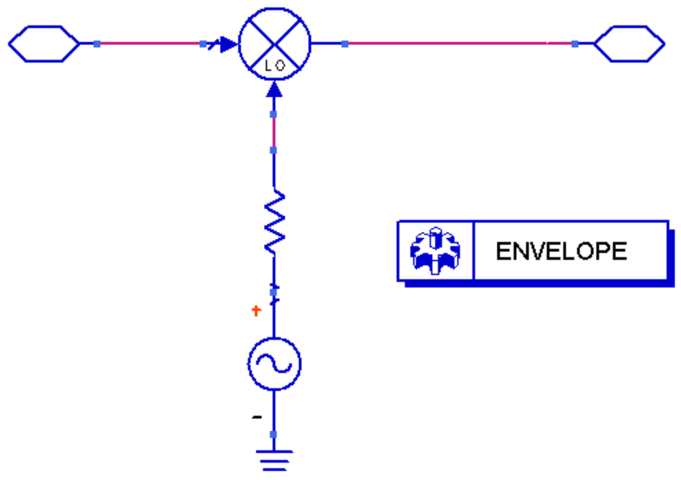
Location: $HPEESOF_DIR/examples/Com_Sys/Co_Sim_wrk/Co_sim_4

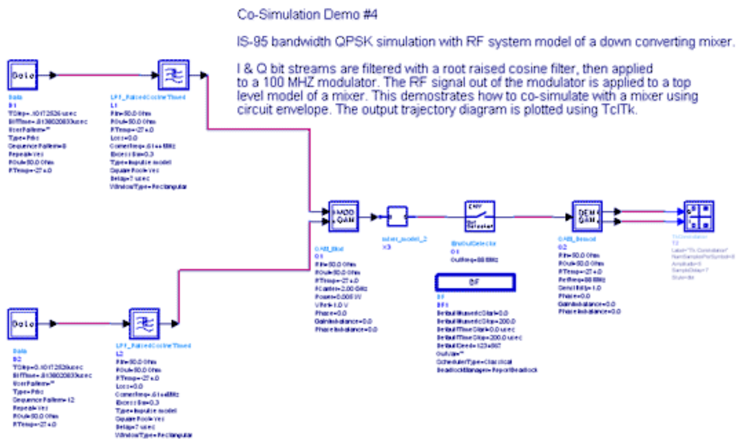
Objective

This example illustrates the basic applications of cosimulation between Agilent Ptolemy at the system level and Circuit Envelope simulation at the circuit level. It shows a IS-95 (1.25 MHz CDMA) bandwidth Quadrature Phase-Shift Keyed (QPSK) simulation with an RF system model of a down-converting mixer.

Setup

I & Q bit streams are filtered with a root raised cosine filter, then applied to a 100 MHZ modulator, symbol rate is 50Ksps. The RF signal out of the modulator is applied to a top level model of a mixer. This demonstrates how to co-simulate with a mixer using Circuit Envelope. The output trajectory diagram is plotted using TclTk.

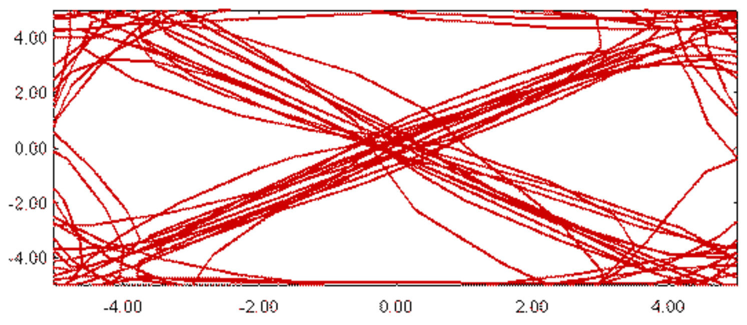



Co-Simulation Demo #4
IS-95 bandwidth QPSK simulation with RF system model of a down converting mixer.
I & Q bit streams are filtered with a root raised cosine filter, then applied to a 100 MHZ modulator. The RF signal out of the modulator is applied to a top level model of a mixer. This demostrates how to co-simulate with a mixer using circuit envelope. The output trajectory diagram is plotted using TclTk.
Data
Data
RF

Analysis

Figure 1: Output trajectory

Figure 1: Output trajectory



Cosimulation of QPSK and RF Upconverting Mixer

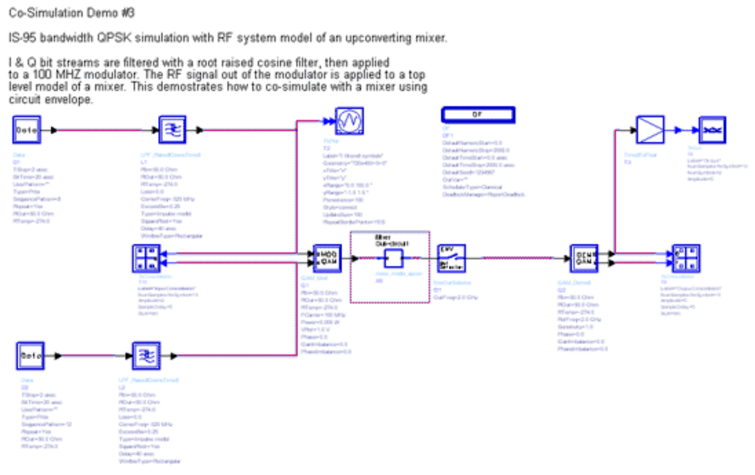
Location: $HPEESOF_DIR/examples/Com_Sys/Co_Sim_wrk/Co_sim_3

Objective

This example illustrates the basic applications of cosimulation between Agilent Ptolemy at the system level and Circuit Envelope simulation at the circuit level. It shows a Quadrature Phase-Shift Keyed (QPSK) modulation signal with a behavioral mixer model set for upconversion.

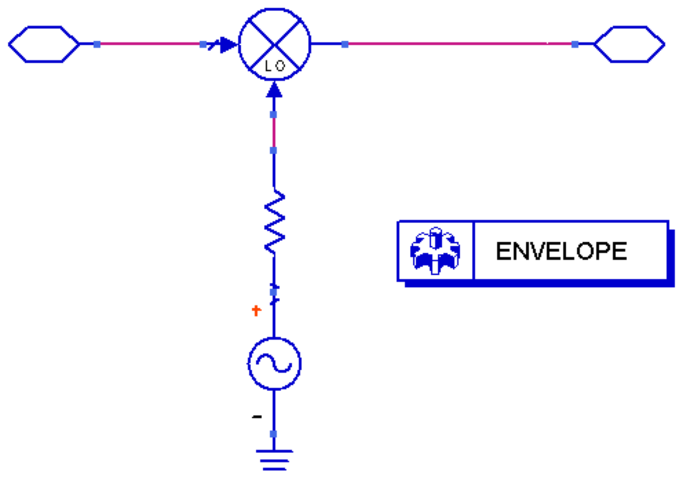
Setup

I & Q bit streams are filtered with a root raised cosine filter, then applied to a 100 MHZ modulator, the symbol rate is 50Ksps. The RF signal out of the modulator is applied to a top level model of a mixer. This demonstrates how to co-simulate with a mixer using Circuit Envelope.


Co-Simulation Demo #3

IS-95 bandwidth QPSK simulation with RF system model of an upconverting mixer.

I & Q bit streams are filtered with a root raised cosine filter, then applied to a 100 MHZ modulator. The RF signal out of the modulator is applied to a top level model of a mixer. This demostrates how to co-simulate with a mixer using circuit envelope.



Analysis

Figure 1: Input trajectory

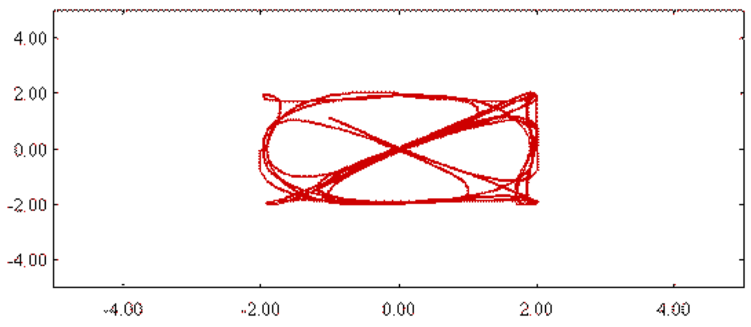


Figure 2: Output trajectory

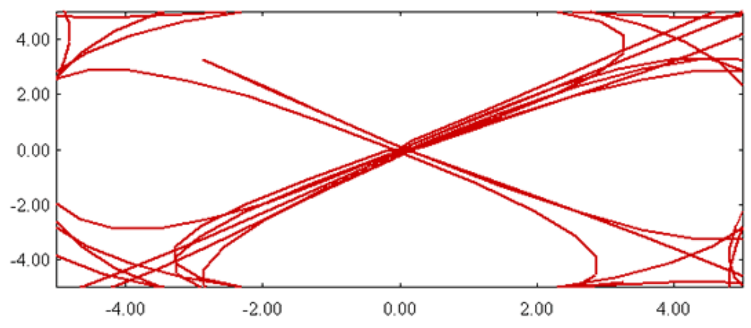


Figure 3: Filtered input I symbols

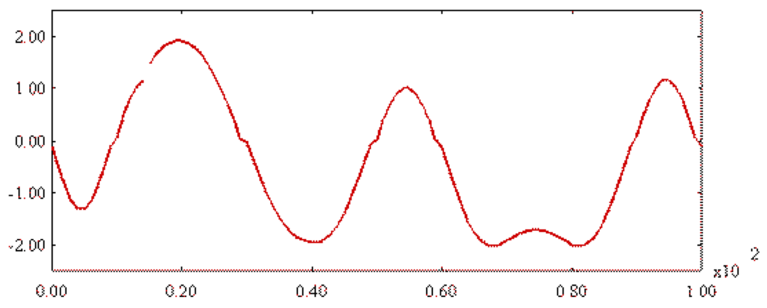
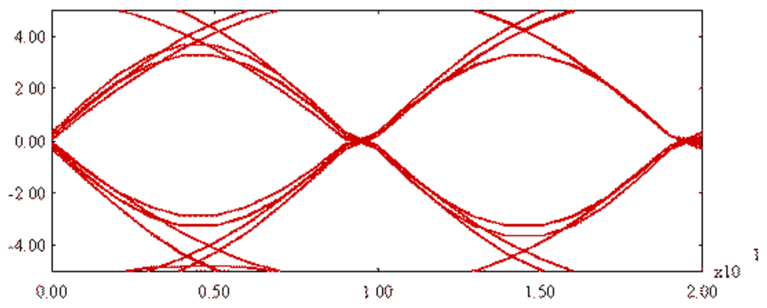


Figure 4: Eye diagram of output I symbols



Cosimulation of QPSK Modulation and Behavioral Model Amplifier

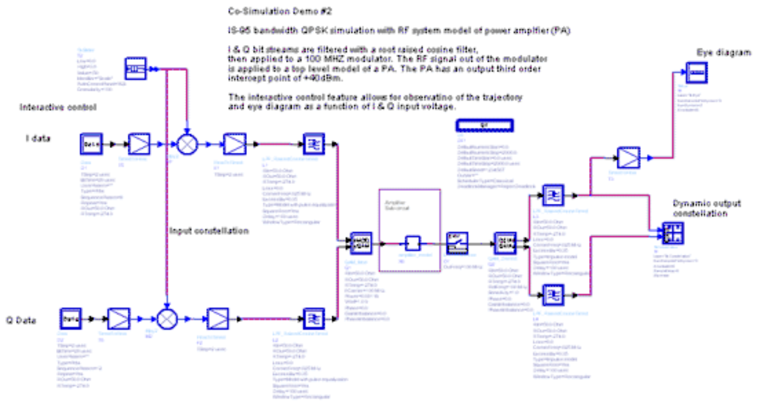
Location: $HPEESOF_DIR/examples/Com_Sys/Co_Sim_wrk/Co_sim_2

Objective

This example illustrates the basic applications of cosimulation between Agilent Ptolemy at the system level and Transient simulation at the circuit level. It uses Quadrature Phase-Shift Keyed (QPSK) modulation with Circuit Envelope simulation of a behavioral model power amplifier.

Setup

I & Q bit streams are filtered with a root raised cosine filter, then applied to a 100 MHZ modulator. The RF signal out of the modulator is applied to a top level model of a PA. The PA has an output third order intercept point of +40dBm. The interactive control feature allows for observation of the trajectory and eye diagram as a function of I & Q input voltage.


Co-Simulation Demo #2
IS-95 bandwidth QPSK simulation with RF system model of power amplifier (PA)
I & Q bit streams are filtered with a root raised cosine filter, then applied to a 100 MHZ modulator. The RF signal out of the modulator is applied to a top level model of a PA. The PA has an output third order intercept point of +40dBm.
The interactive control feature allows for observation of the trajectory and eye diagram as a function of I & Q input voltage.
Interactive control
I data
Q Data
Input constellation
Eye diagram
Dynamic output constellation

Analysis

Figure 1: Output trajectory

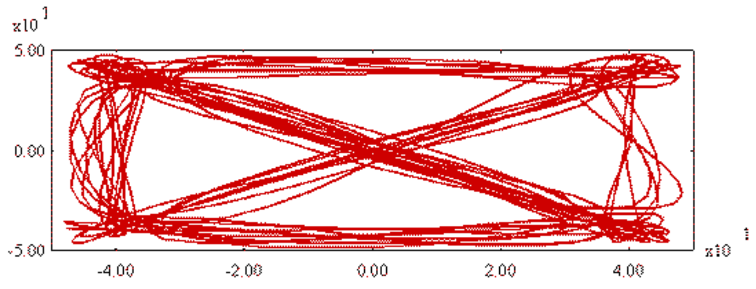
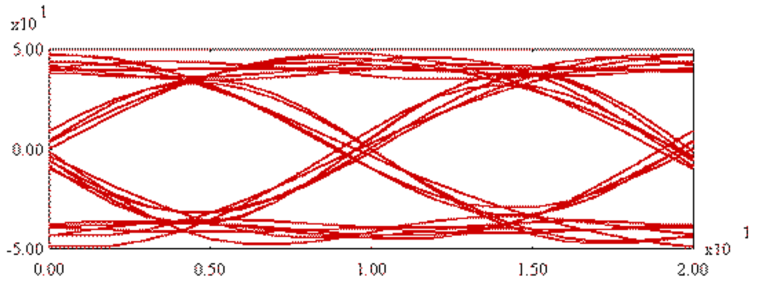


Figure 2: Eye diagram



Delta Sigma Converter

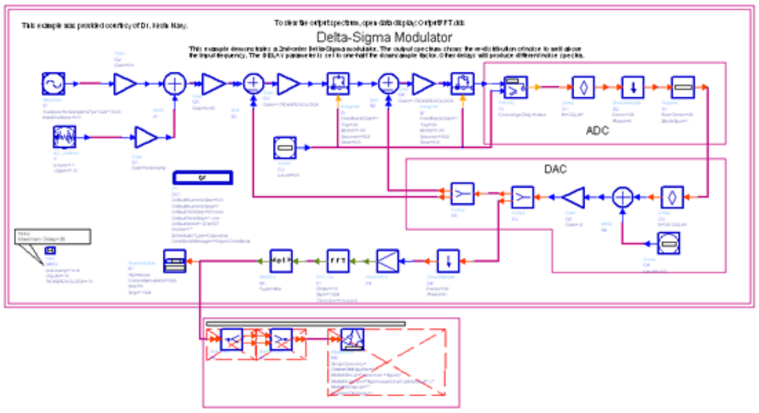
Location: $HPEESOF_DIR/examples/Com_Sys/DeltaSigma_wrk

Objective

The Delta_Sigma modulator is useful for the accurate digitization of radio signals in the implementation of a digital receiver. This example demonstrates a 2nd-order Delta-Sigma modulator.

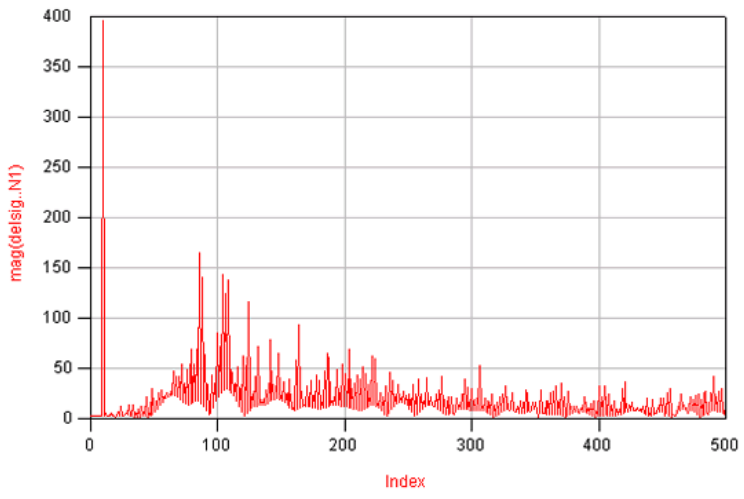
Setup

The use of oversampling and feedback results in less stringent requirements on circuit component tolerances compared to conventional converters using analog comparators. 1024-point FFT provides the output spectrum. The output spectrum shows the re-distribution of noise to well above the input frequency. The DELAY parameter is set to one-half the downsample factor. Other delays will produce different noise spectra.


This example was provided courtesy of Dr. Kevin Nary.
To view the output spectrum, open data display OutputFFT.dds
Delta-Sigma Modulator
ADC
DAC

Analysis

Figure 1: Sample spectrum



Notes

This example was provided courtesy of Dr. Kevin Nary.

GSM System - Basic Simulation

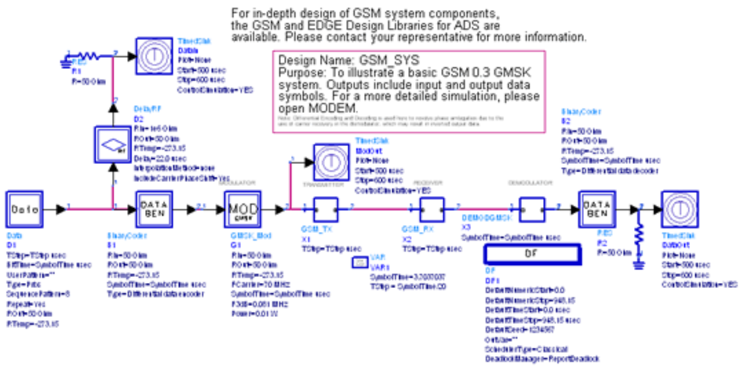
Location: $HPEESOF_DIR/examples/Com_Sys/gsm_wrk

Objective

This example illustrates a basic GSM 0.3 Gaussian Minimum-Shift Keyed (GMSK) system built using standard ADS components. It does not include any baseband coding. For detailed simulation and modeling of GSM systems, the GSM Design Library is available for use with ADS.

Setup

Random data are differentially encoded (to allow for carrier recovery) and Gaussian filtered at BT=0.3. The modulated signal is passed through a transmitter, receiver and demodulator cascade and differentially decoded for output display.



For in-depth design of GSM system components, the GSM and EDGE Design Libraries for ADS are available. Please contact your representative for more information.

Design Name: GSM_SYS
Purpose: To illustrate a basic GSM 0.3 GMSK system. Outputs include input and output data symbols. For a more detailed simulation, please open MODEM.

Analysis

Figure 1: Input and output data

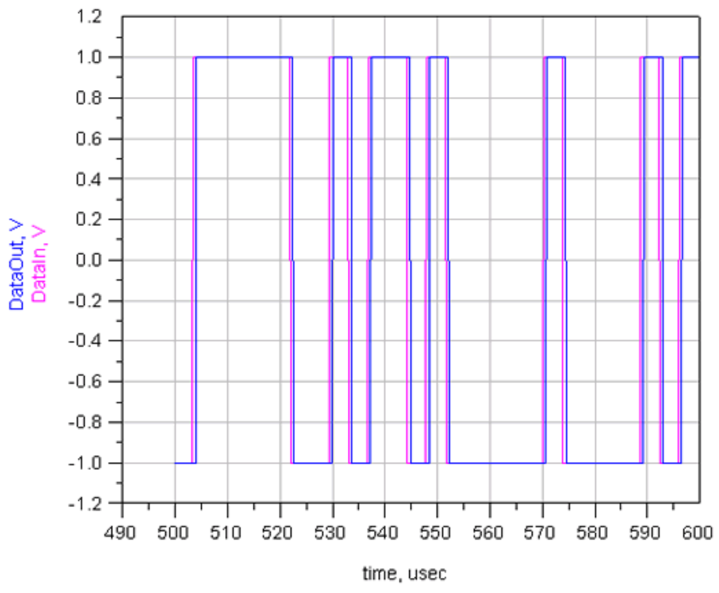


Figure 2: GMSK constellation

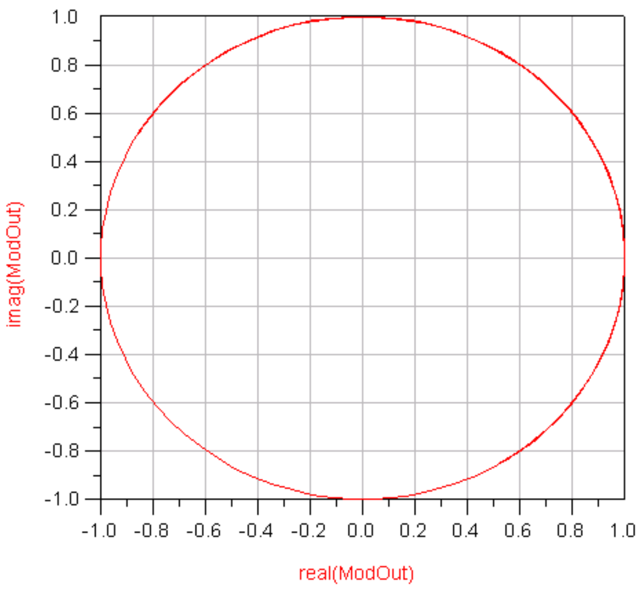


Figure 3: Recovered clock

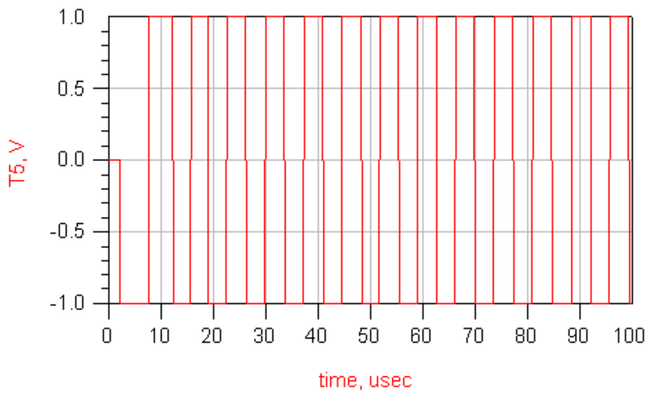
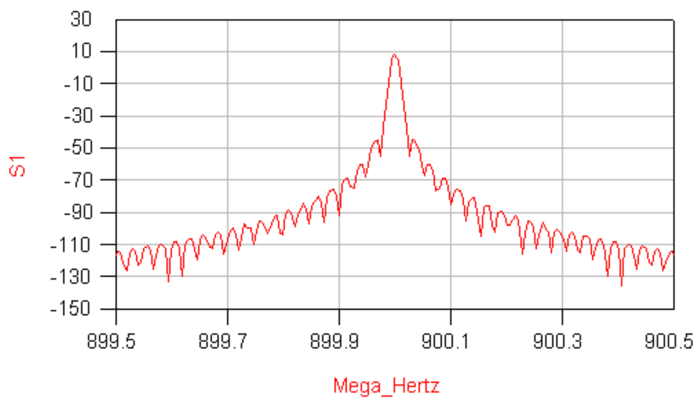


Figure 4: Recovered carrier



IS-95 CDMA Filter

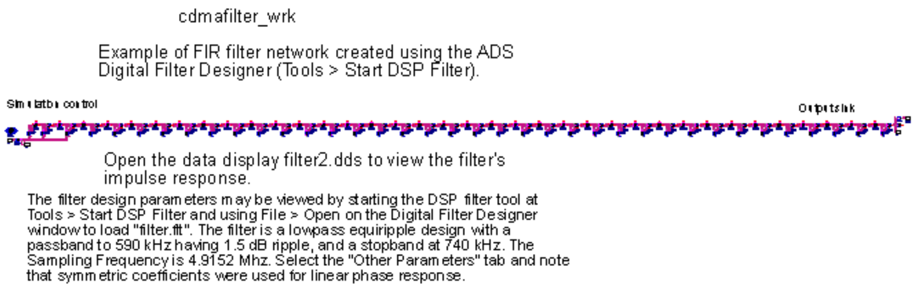
Location: $HPEESOF_DIR/examples/Com_Sys/cdmafilter_wrk

Objective

This example shows a Finite Impulse Response (FIR) filter network created using the ADS Digital Filter Designer (Tools > Digital Filter > Start Digital Filter). The filter is a lowpass equiripple design with a passband to 590 kHz having 1.5 dB ripple, and a stopband at 740 kHz. The Sampling Frequency is 4.9152 Mhz.

Setup

1. "filter1" is an FIR filter created by the ADS Digital Filter Designer using the synthesizable Library. The filter design parameters may be viewed by starting the DSP filter tool at Tools > Digital Filter > Start Digital Filter and using File > Open on the Digital Filter Designer window to load "filter.flt". Select the "Other Parameters" tab and note that symmetric coefficients were used for linear phase response.
2. "filter2" includes the necessary simulation for fixed point performance in ADS.

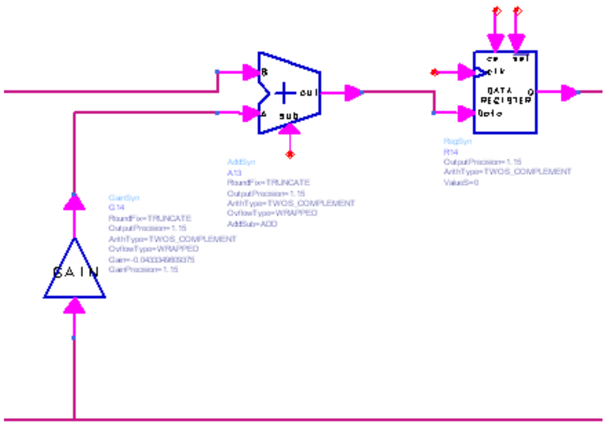


cdmafilter_wrk

Example of FIR filter network created using the ADS Digital Filter Designer (Tools > Start DSP Filter).

Open the data display filter2.dds to view the filter's impulse response.

The filter design parameters may be viewed by starting the DSP filter tool at Tools > Start DSP Filter and using File > Open on the Digital Filter Designer window to load "filter.flt". The filter is a lowpass equiripple design with a passband to 590 kHz having 1.5 dB ripple, and a stopband at 740 kHz. The Sampling Frequency is 4.9152 Mhz. Select the "Other Parameters" tab and note that symmetric coefficients were used for linear phase response.



Analysis

Figure 1: Frequency response created by Filter Design Tool

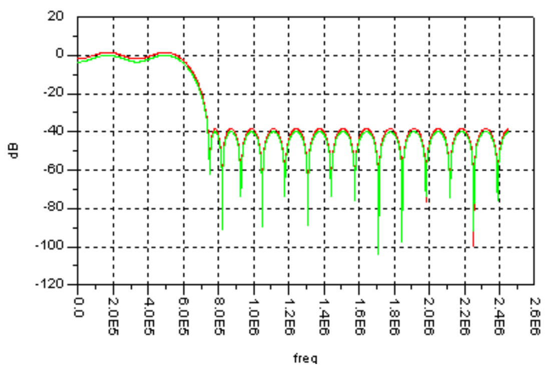
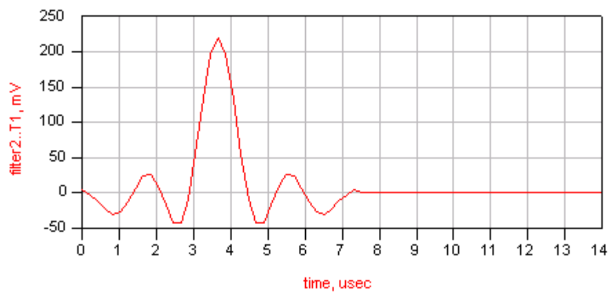


Figure 2: Filter step response from ADS simulation



IS-95 CDMA Forward-Channel Test Modulator

Location: $HPEESOF_DIR/examples/Com_Sys/IS95_wrk

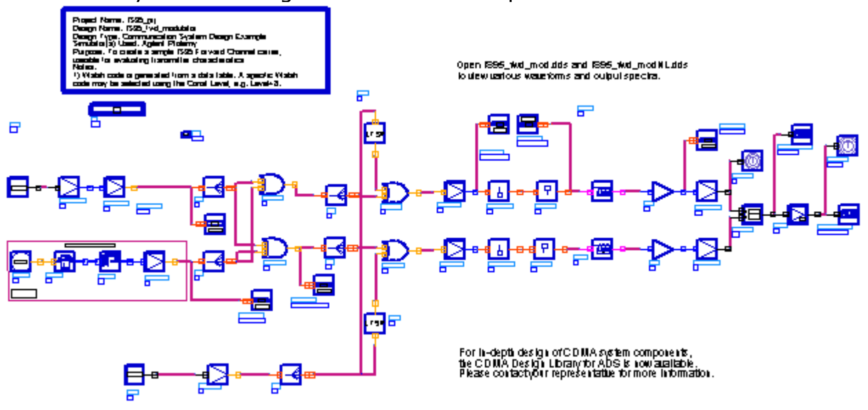
Objective

This example illustrates the use of Agilent Ptolemy to create a basic IS95 Forward-Channel signal for the evaluation of transmitter characteristics. A nine-channel modulator is also shown as an example. It does not included any baseband coding or framing. For detailed simulation of CDMA systems, the CDMA Design Library for ADS is available.

Setup

1. "IS95_fwd_modulator" is a basic IS95 modulator for one channel, determined by the selected Walsh function and modulator power. First pseudo-random I and Q bit sequences are generated and combined with a Walsh code read from a time-domain data file. The sequences are then combined with PN short codes generated by linear feedback shift registers. After decimation and zero insertion to change rectangular pulses to impulses, pulse shaping is performed. This is followed by quadrature modulation onto a carrier. A spectrum analyzer provides a normalized and video-averaged output spectrum display.

2. "IS97_9Channel_TestGen" is a 9-channel modulator similar to that used for the evaluation of base station power amplifier performance. The peak-to-average characteristic of the modulated signal is highly dependent upon the particular traffic channels selected. An alternate traffic channel setting is suggested for a higher peak-to-average signal. This design provides a pilot, synch and paging channel, plus 6 selectable traffic channels. A suitable power amplifier model may be included, and non-linear performance may be evaluated. At present, a spectral display is available. The PA model could be a behavioral model as shown, or it could be a circuit model simulated by cosimulating with Circuit Envelope.


Project Name: IS95_prj
Design Name: IS95_fwd_modulator
Design Type: Communication System Design Example
Simulator(s) Used: Agilent Ptolemy
Purpose: To create a simple IS95 Forward Channel source, useable for evaluating transmitter characteristics
Notes:
1) Walsh code is generated from a data table. A specific Walsh code may be selected using the Const Level, e.g. Level=3.

Open IS95_fwd_mod.dds and IS95_fwd_modNL.dds to view various waveforms and output spectra.

For in-depth design of CDMA system components, the CDMA Design Library for ADS is now available. Please contact your representative for more information.

Analysis

Figure 1: Input I data, PN spread I data, and filtered I data

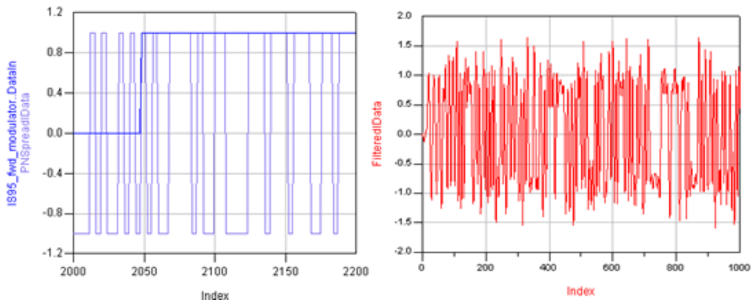
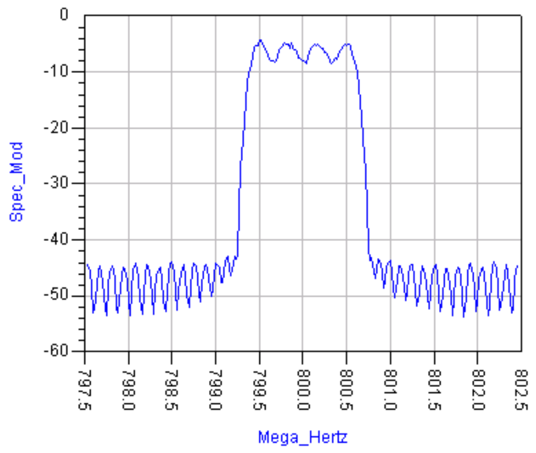


Figure 2: Spectrum of modulated signal



Notes

- Design "IS97_9Channel_TestGen" may take a long time to simulate. You can read in a saved data file by opening the Data Display format file IS97_9Channel.dds (ADS Main window - Window > Open Data Display).
- Modulator ACPR performance is dependent upon the FIR filter used. This example uses the filter designed in the example cdmafilter_wrk using the ADS DSP Filter tool, which provides about 40 dB of sidelobe rejection. If better performance is needed, it is suggested that a different filter be used.

- Data Flow Controllers (DF) in subnetworks must be deactivated when the network is simulated. There should be only one DF item and it must be at the top level network.

Linear Budget Analysis

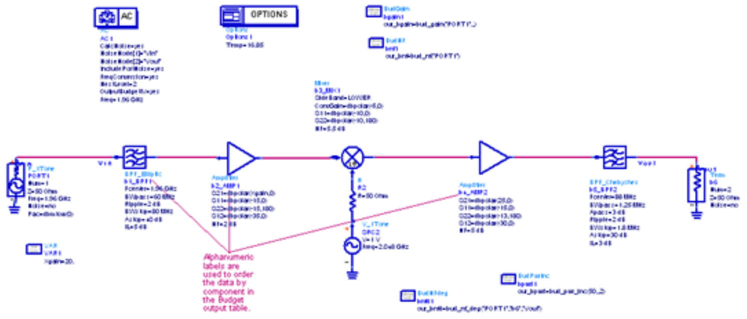
Location: $HPEESOF_DIR/examples/Com_Sys/Linear_Budget_wrk

Objective

This workspace contains an example of Budget simulation using expressions. For cascaded two-port networks, such as this, the RF Budget Simulation controller (see Tutorial/RF_Budget_Examples_wrk) is recommended and preferred.

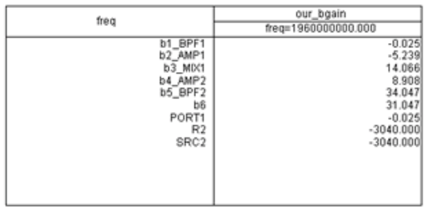
Setup

1. "Linear_Budget" is a simple cascade of two-port components, including a mixer for frequency conversion. AC analysis is used for simulation. Open "Linear_Budget.dds" to view a completed data display showing Gain, Incident Power, Noise Figure, and Noise Figure Degradation (due to each component).
2. The measurements may be obtained by two methods. The user can place Budget measurements from the Simulation-AC Palette or library and use Data Display "Schematic Measurements", or place Budget measurement equations onto the Data Display window, see Data Display "DataDisplayMeasurements".


AC
OPTIONS
Alphanumeric labels are used to order the data by component in the Budget output table.

Analysis

Figure 1. Gain budget



freq	our_bgain freq=1960000000.000
b1_BPF1	-0.025
b2_AMP1	-5.239
b3_MIX1	14.066
b4_AMP2	8.908
b5_BPF2	34.047
b6	31.047
PORT1	-0.025
R2	-3040.000
SRC2	-3040.000

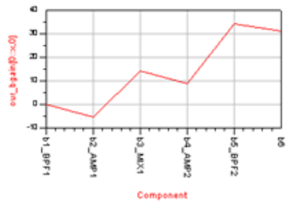
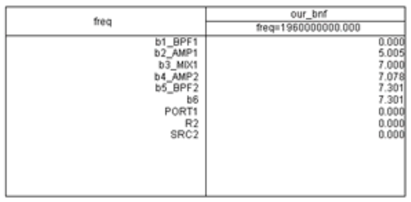
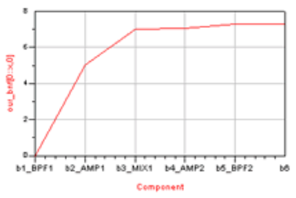


Figure 2: Noise figure budget



freq	our_bnf freq=1960000000.000
b1_BPF1	0.000
b2_AMP1	5.005
b3_MIX1	7.000
b4_AMP2	7.078
b5_BPF2	7.301
b6	7.301
PORT1	0.000
R2	0.000
SRC2	0.000



Notes

- For cascaded two-port networks such as Linear_Budget, the RF Budget controller is recommended and preferred. Please see Tutorial/RF_Budget_Examples_wrk.

LSF (Load Sharing Facility) usage with BER simulation

Location: $HPEESOF_DIR/examples/Com_Sys/ParallelBER_wrk

Objective

In this example, the usage of LSF (Load Sharing Facility) and the simulation performance improvement are illustrated. The workspace contains two example designs that demonstrate the use, accuracy and performance improvement of the parallel BER option.

Setup

This workspace contains 2 example designs that demonstrate the use, accuracy and performance improvement of the Parallel BER option (Simulate menu > Simulation Setup window > Parallel tab). To use this option one must have the LSF client installed on his/her machine and the machine must be connected to an LSF cluster.

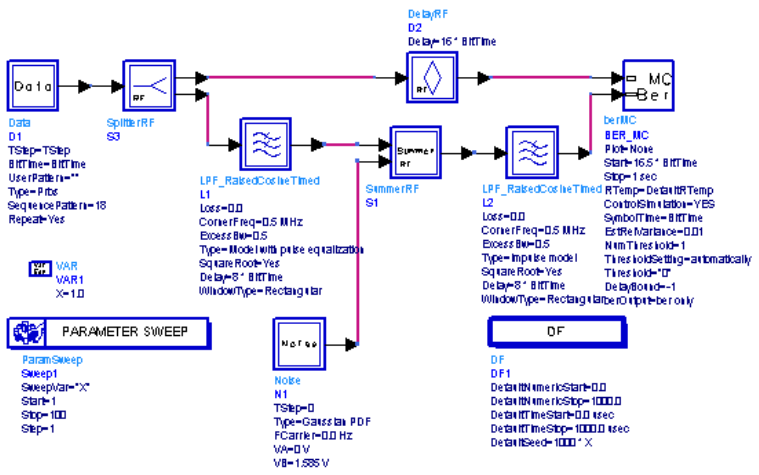
For more information on installing LSF and running remote simulations refer to *Circuit Remote Simulation* (cktsim).

For more information on the Parallel BER option usage refer to *berMC* (sinks).

The first example design, ParallelBER_Example1, is used to demonstrate the statistical equivalence between the BER results obtained from a single simulation and the BER results obtained using the Parallel BER option. Performance improvement results are also shown. The design is a simple 2-level PAM system. The Eb/No of the system is set to 6 dB. The DefaultSeed parameter of the DF controller is swept in order to get 100 statistically independent BER measurements. The statistics of these measurements from the single simulation as well as from the parallel simulations are shown and compared in the ParallelBER_Example1.dds file.

The second example design, ParallelBER_Example2, uses the same 2-level PAM system as the first example. However, the DefaultSeed is set to its default value of 1234567 and the Eb/No of the system is swept from 4 dB to 9 dB. The results from the single simulation and the parallel simulations are compared in the ParallelBER_Example2.dds file.

Analysis



Data
D1
TStep=TStep
BitTime=BitTime
UserPattern=""
Type=Prbs
SequencePattern=18
Repeat=Yes

SplitterRF
S3

DelayRF
D2
Delay=16 * BitTime

LPF_RaisedCosineTimed
L1
Loss=0.0
CornerFreq=0.5 MHz
ExcessBw=0.5
Type=Model with pulse equalization
SquareRoot=Yes
Delay=8 * BitTime
WindowType=Rectangular

SummerRF
S1

LPF_RaisedCosineTimed
L2
Loss=0.0
CornerFreq=0.5 MHz
ExcessBw=0.5
Type=Impulse model
SquareRoot=Yes
Delay=8 * BitTime
WindowType=Rectangular

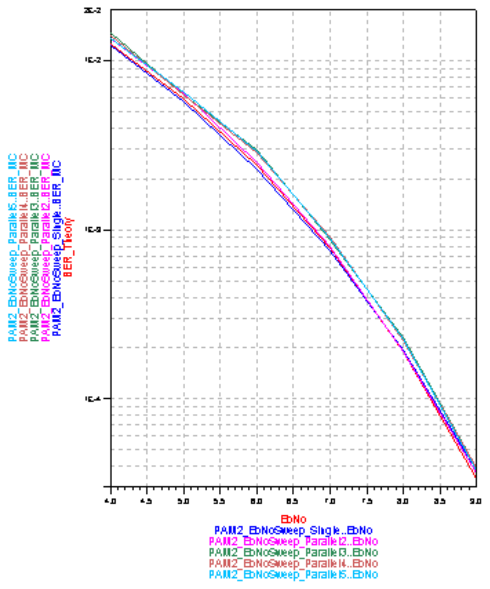
berMC
BER_MC
Plot=None
Start=16.5 * BitTime
Stop=1 sec
RTemp=DefaultRTemp
ControlSimulation=YES
SymbolTime=BitTime
EstReVariance=0.01
NumThreshold=1
ThresholdSetting=automatically
Threshold="0"
DelayBound=-1
berOutput=ber only

VAR
VAR1
X=1.0

PARAMETER SWEEP
ParamSweep
Sweep1
SweepVar="X"
Start=1
Stop=100
Step=1

Noise
N1
TStep=0
Type=Gaussian PDF
FCarrier=0.0 Hz
VA=0 V
VB=1.585 V

DF
DF
DF1
DefaultNumericStart=0.0
DefaultNumericStop=1000.0
DefaultTimeStart=0.0 usec
DefaultTimeStop=1000.0 usec
DefaultSeed=1000 * X



Multi-Channel Nonlinear Budget Analysis

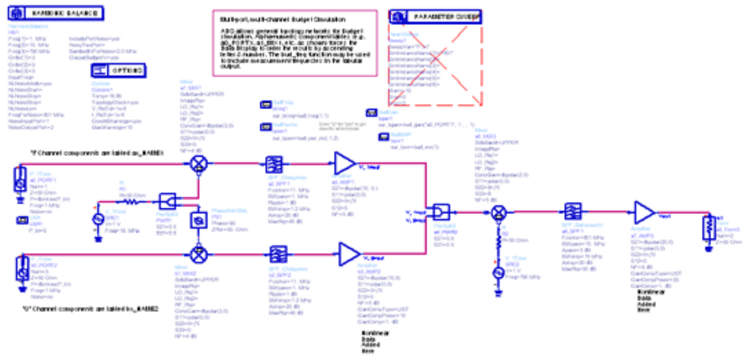
Location: $HPEESOF_DIR/examples/Com_Sys/MultiChan_NL_Budget_wrk

Objective

This example workspace contains an example network having a multi-channel, multi-port topology with nonlinear components for Budget simulation using Harmonic Balance analysis. This example illustrates several important measurements as well as highlighting the capability to obtain Budget measurements on multi-channel networks incorporating multi-port components or networks.

Setup

1. "IQ_mod_bud": A multi-port network is simulated using Harmonic Balance analysis. This example also includes frequency conversion, nonlinear data and multi-port components. An alphanumeric labeling method is used to make the data display order the components in a table. Measurements include Gain and Incident Power.
2. Open data display "IQ_Budget.dds" to see measurements using Data Display equations. Open "IQ_BudgetSchematic.dds" to see the same measurements implemented using the schematic page measurement items found under the Simulation-HB Palette or Library.


HARMONIC BALANCE
OPTIONS
PARAMETER SWEEP
Multi-port, multi-channel Budget Simulation
"I" Channel components are labled a_NAME
"Q" Channel components are labled b_NAME

Analysis

Figure 1: Gain from port1 to the input of each component in the "a" path

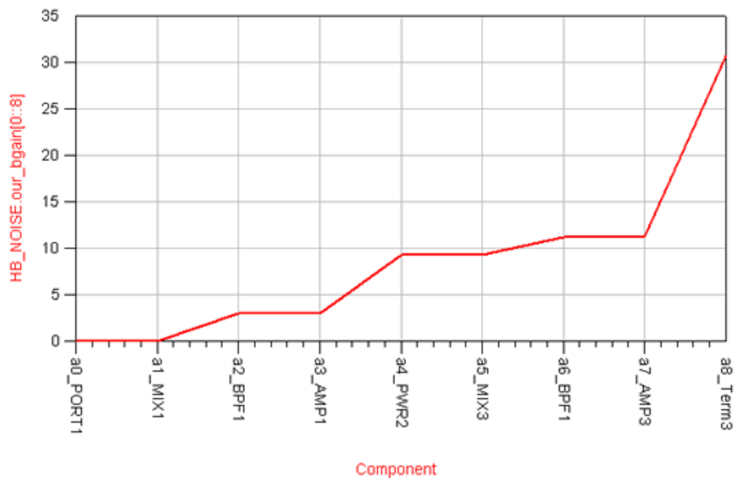
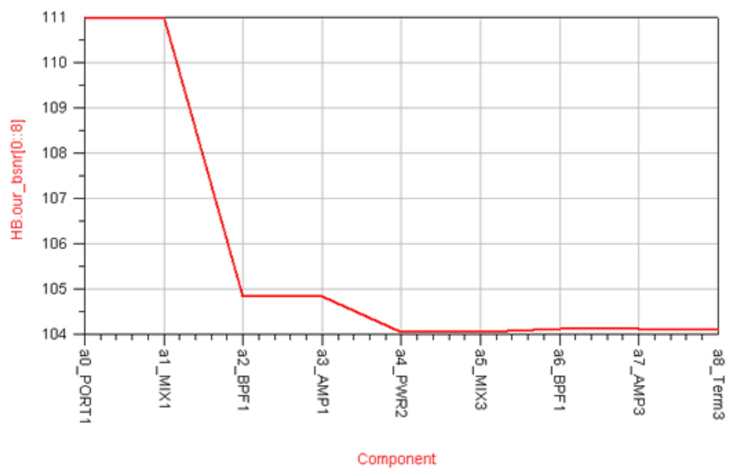


Figure 2: SNR from port1 to the input of each component in the "a" path



Notes

- Alphanumeric component labels (e.g. a0_PORT1, a1_MIX1, etc. as shown) force the Data Display to order the results by ascending letter & number. In ADS 1.5, you can use Generate Budget Path under the Simulate menu to define the endpoints of a signal path through a muti-channel network. The bud_freq function may be used to include measurement frequencies in the tabular output.

OFDM Modulation

Location: $HPEESOF_DIR/examples/Com_Sys/OFDM_WLAN_wrk

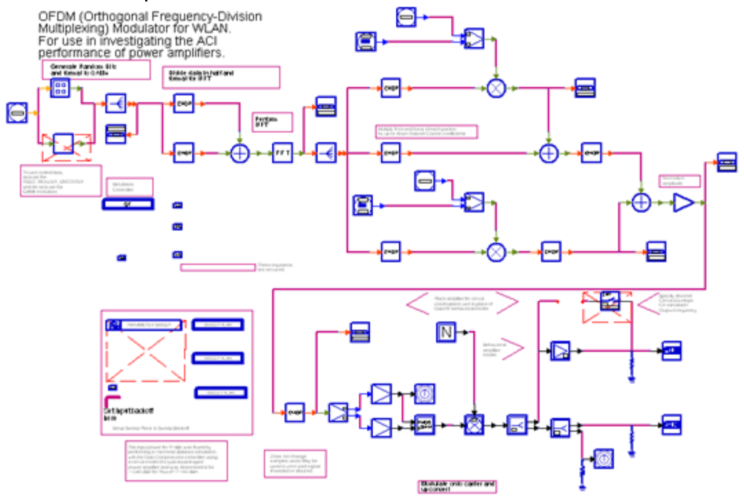
Objective

This example illustrates a method for investigating Adjacent Channel Power (ACP) Ratio or Interference (ACPR/ACI) performance of a power amplifier using Orthogonal Frequency Division Multiplexing (OFDM) modulation. This type of signal has a large (> 12 dB) peak-to-average ratio. In this case, a prototype OFDM system is used, based upon that proposed for use at 5.2 GHz and having a data rate of about 20 Mb/sec.

Setup

1. Random bits are first mapped to a QAM4 (QPSK) constellation. Then, they are divided into two packets and zero padded in between; this is required for the Inverse Fast Fourier Transform (IFFT) that follows. To minimize the bandwidth, the carrier packets are shaped using Raised-Cosine coefficients, then overlapped. Finally, the packets are zero-padded to 2^N to minimized FFT processing noise in the spectrum measurements.
2. Modulation onto a RF carrier is accomplished with the MODQAM component. The signal is up-converted and then applied to a nonlinear amplifier model. The model shown, GainRF, may be used in two ways: 1) Nonlinearity modeled by just the 1dB

compression point (no phase shift vs. input power is considered), or 2) Nonlinearity modeled by measured Gain Compression vs. Input Power. This includes the phase shift vs. input power, or "AM-to-PM" effect. Alternatively, a circuit-level model of the PA may be used via the ability of ADS / Agilent Ptolemy to co-simulate with the Circuit Envelope simulator.


OFDM (Orthogonal Frequency-Division Multiplexing) Modulator for WLAN.
For use in investigating the ACI performance of power amplifiers.
N

Analysis

Figure 1: PA output spectrum

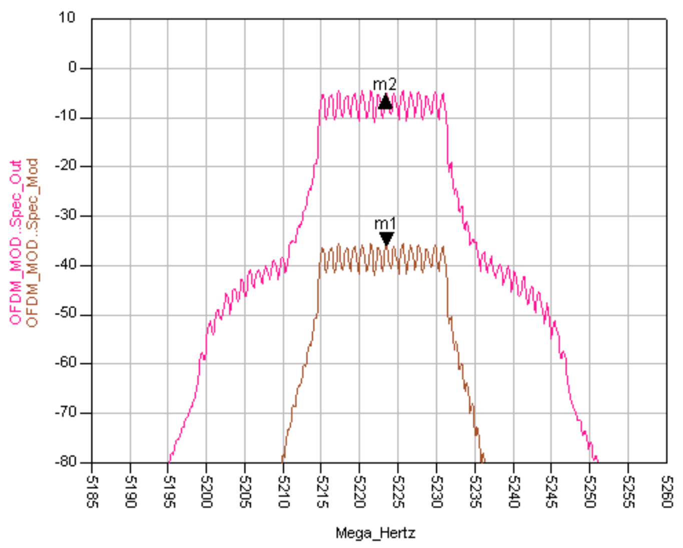
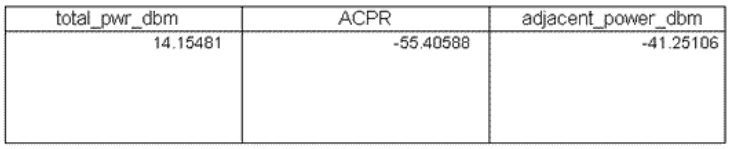


Figure 2: ACPR



total_pwr_dbm	ACPR	adjacent_power_dbm
14.15481	-55.40588	-41.25106

Prototype 8-VSB Modulator

Location: $HPEESOF_DIR/examples/Com_Sys/VSB_wrk

Objective

This example illustrates a simple prototype 8-VSB modulator. The 8-VSB modulation format is part of the US Advanced Television Standards Committee (ATSC) standard for terrestrial digital television broadcasting.

Setup

In the primary design "VSB_Mod2", a pseudo-random sequence (as specified) is generated and mapped to 8-level symbols. An FIR (Finite Impulse Response) filter provides the overall spectral containment of the signal. This filter was designed using the ADS Digital Filter design program (Tools > Digital Filter > Start Digital Filter). The filtered signal is split and directed to a quadrature modulator with 50MHz carrier. However, the "Q" signal is passed through a Hilbert transformer. This adds a 90-degree phase shift to the "Q" signal. When modulated in quadrature, this results in cancellation of the lower sideband.

Analysis

Figure 1: Unfiltered and output spectrum

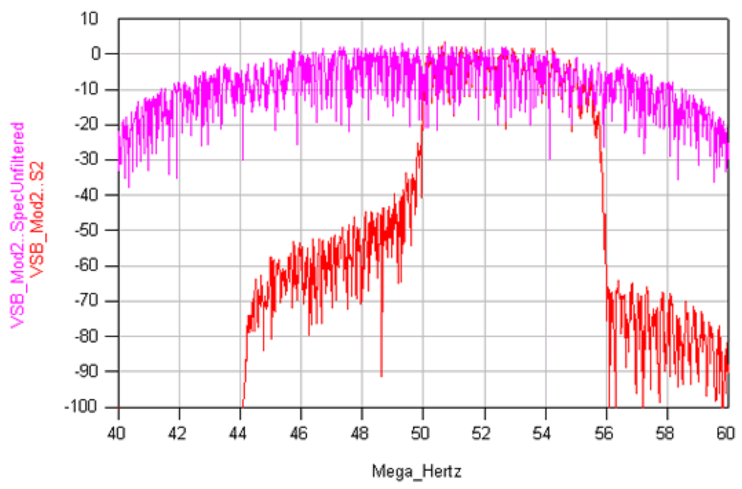
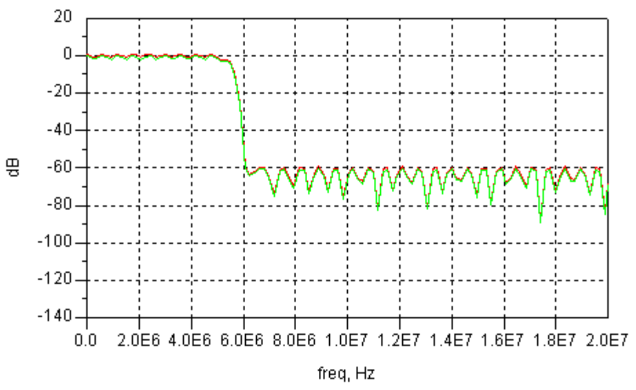


Figure 2: Filter response



Notes

- The LSB attenuation will depend upon the delay of the HilbertSplit component. However, an increased delay will also increase the simulation time. Doubling the delay to 4096 or to 8192 will significantly improve the performance.
- The Root-Raised-Cosine FIR filter parameters are: Sample Rate = 387.4392 MHz (12 samples per bit), Alpha (Rolloff Factor) = 0.1152, Ripple = 0.01 dB (design goal), Sinc compensation = unity (not used),Number of taps as designed = 1418.
- "PN511_code.dds": 511-bit PN code generated by Linear Feedback Shift Register.
- "Symbols.dds": code symbols and 8-level-mapped symbols.
- "TimedOut.dds": real and imaginary parts of modulated envelope.
- "FilterResponse.dds": data display automatically generated by the DSP Filter Tool.

QAM System Co-Simulation Including LNA Noise

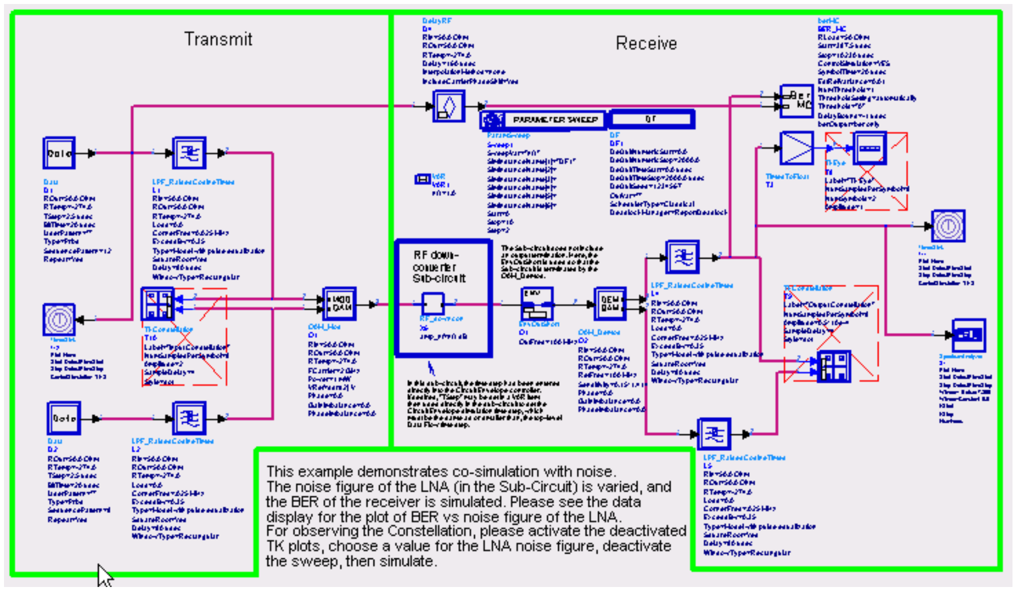
Location: $HPEESOF_DIR/examples/Com_Sys/Co_Sim_noise_wrk

Objective

This example demonstrates co-simulation of a QAM system, with noise included in the subcircuit. The noise figure of the receiver's LNA is varied and the BER is simulated versus this noise figure.

Setup

The Co_sim_noise schematic shows the QAM transmit and receive sections as well as simulation controllers and BER measurement sinks. The RF downconverter is the subcircuit inside the blue box

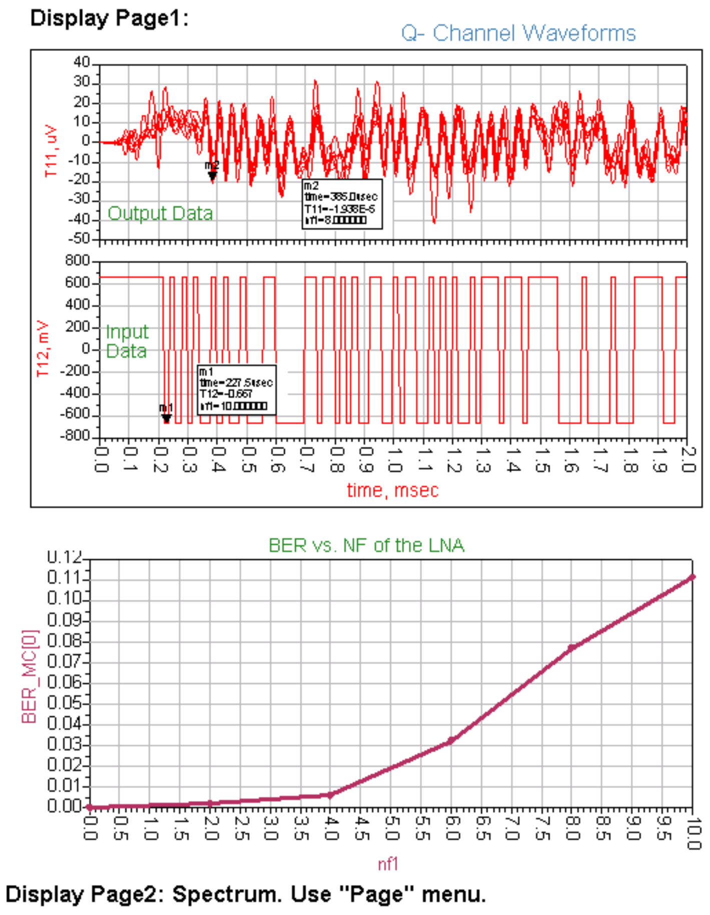


This example demonstrates co-simulation with noise.
The noise figure of the LNA (in the Sub-Circuit) is varied, and the BER of the receiver is simulated. Please see the data display for the plot of BER vs noise figure of the LNA.
For observing the Constellation, please activate the deactivated TK plots, choose a value for the LNA noise figure, deactivate the sweep, then simulate.

QAM transmitter/receiver simulation setup

Analysis

The data display shows the BER versus amplifier noise figure as well as the Q-channel input and output waveforms. There is a different output waveform for each value of the LNA's noise figure.



Q-channel input and output waveforms and BER versus amplifier noise figure; display page 2 (not shown) has the Q-channel output spectrum

RFID Reader and Tag Basic Mode 1 Link

Location: $HPEESOF_DIR/examples/Com_Sys/RFID_System_wrk

Objective

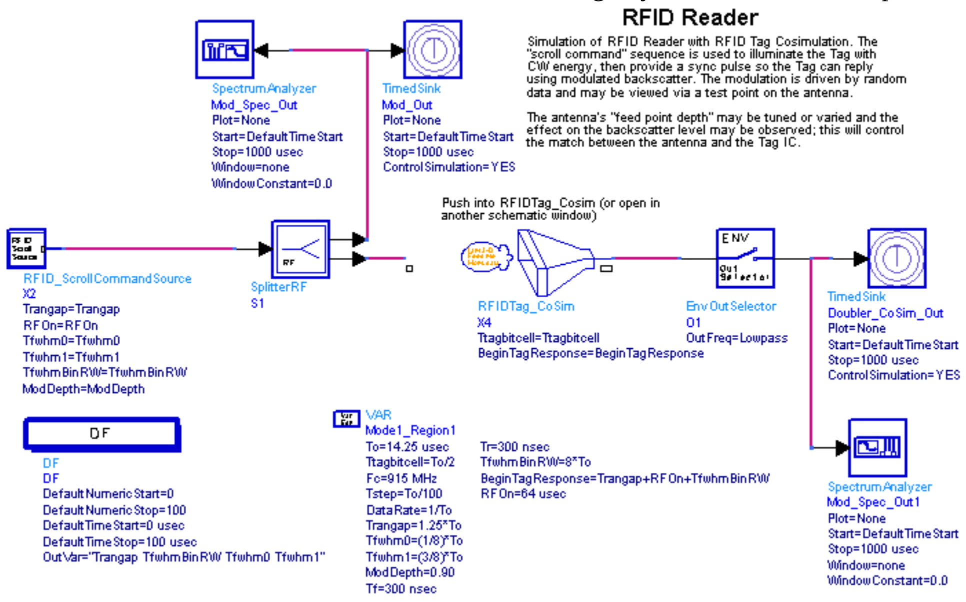
This workspace is a companion to RFID_TagDesign_wrk. The patch antenna and Schottky diode tag rectifier designed in that workspace are used here in a system-level cosimulation.A design (ANTENNA_MODEL_SIMULATION) for sweeping the antenna pattern vs. azimuth is also included.

Setup

Four windows will open when this workspace is selected:

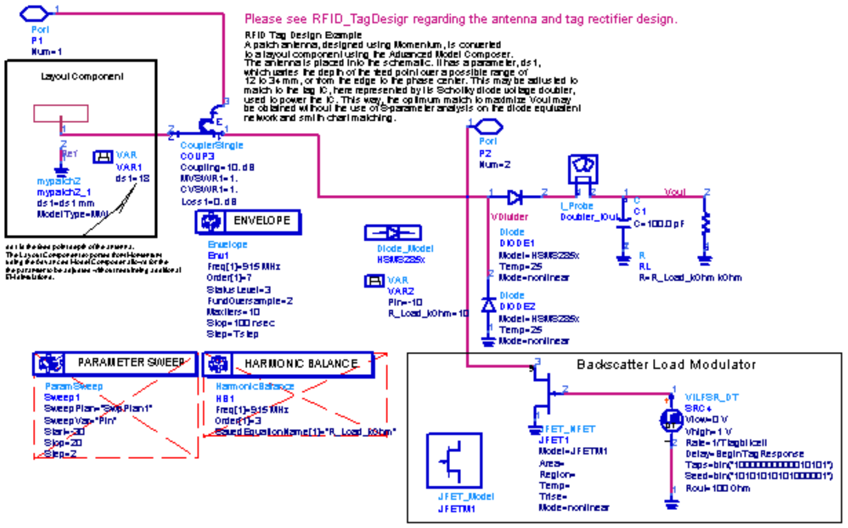
- A_README
- Reader

Figure: RFID Reader-Tag Link



- RFIDTag_CoSim

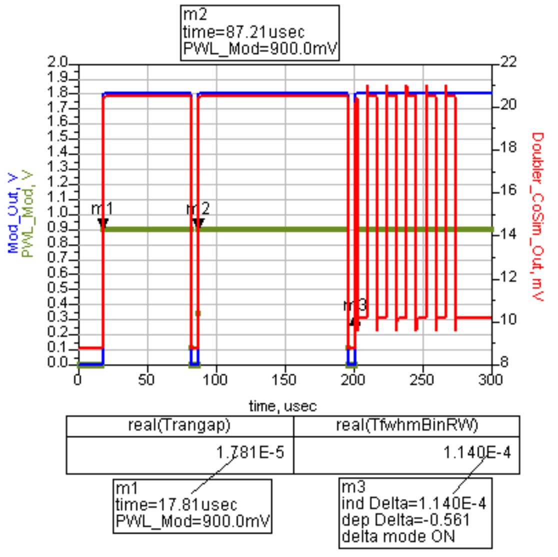
Figure: Subnetwork for Co-Simulation of Tag


Please see RFID_TagDesign regarding the antenna and tag rectifier design.
RFID Tag Design Example
Layout Component
Port P1 Num=1
CouplerSingle COUP3 Coupling=10. dB
ENVELOPE
PARAMETER SWEEP
HARMONIC BALANCE
Port P2 Num=2
VDivider
Diode DIODE1 Model=HSMS285x
Diode DIODE2 Model=HSMS285x
Diode_Model HSMS285x
VAR VAR2 Pin=-10 R_Load_kOhm=10
C1 C=100.0 pF
R RL R=R_Load_kOhm kOhm
Vout
Backscatter Load Modulator
JFET_Model JFETM1
JFET_NFET JFET1 Model=JFETM1
VtLFSR_DT SRC4 Vlow=0 V Vhigh=1 V

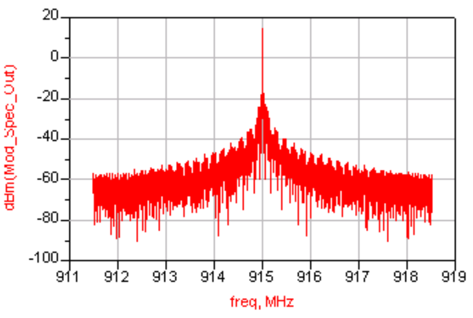
- Display: Reader

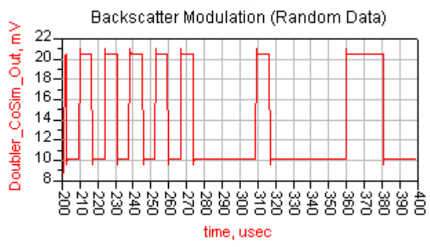
Analysis

The reader sends a "scroll command" sequence similar to the one described in the Mode 1 specification. After a CW signal illuminates the tag for 64 usec, a sync pulse is sent and then the tag "replies" by modulating the backscatter with a pseudorandom sequence.



real(Trangap)	real(TfwhmBinRW)
1.781E-5	1.140E-4





RFID Transponder and Antenna Design using Advanced Model Composer

Location: $HPEESOF_DIR/examples/Com_Sys/RFID_TagDesign_wrk

Objective

This example addresses the issue of matching the tag antenna to the tag IC to maximize rectification and modulation depth. Please also see RFID_System_wrk. There are 3 parts to this example:

1) Matching using the Smith Chart Utility
Using the Smith Chart Utility that is included with several DesignGuides and a Schottky diode equivalent circuit to find a conjugate match.

- See designs and associated displays:SingleDiodeMatch, DoublerDiodeMatch,RectifierModel, Doubler

2) Design of a Patch Antenna using Momentum
This design is converted to a layout component using the Advanced Model Composer. It has a parameter "ds1" which allows the feed point depth of the antenna to be adjusted.

- See designs and displays:Layout: mypatch2, Schematic/Display: AntennaSweep

3) Design of a Voltage Doubler Circuit
Using the Shottky diode non-linear model, with the patch antenna integrated into the schematic asa layout component. The antenna feed point depth parameter ds1 is adjusted to maximize the voltage output.

- See design and associated display: AntennaDoubler

Setup

This example opens several windows when selected:

- README
- Layout mypatch_2
- AntennaDoubler
- Displays: AntennaSweep and AntennaDoubler

Figure 1: RFID Tag Patch Antenna

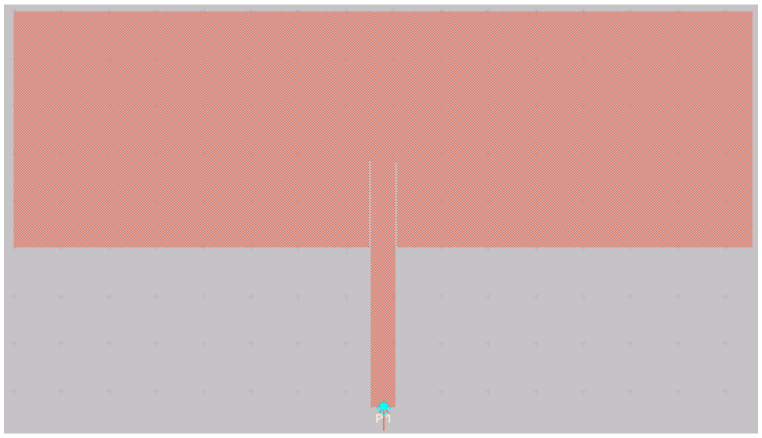
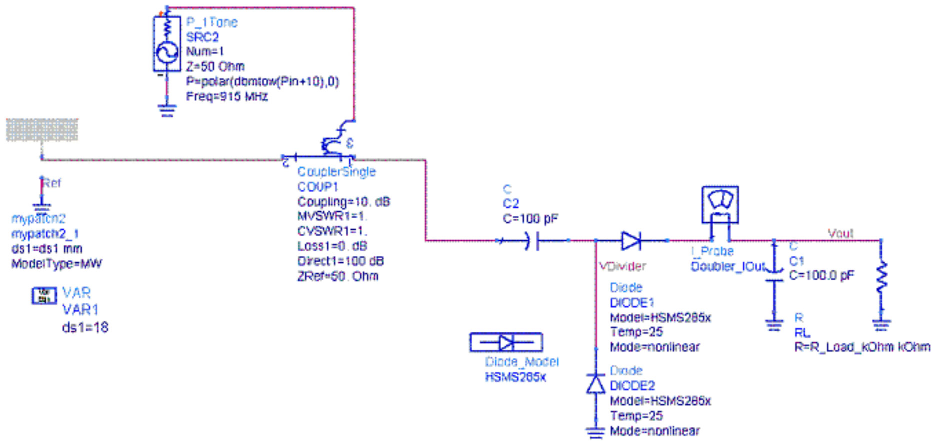


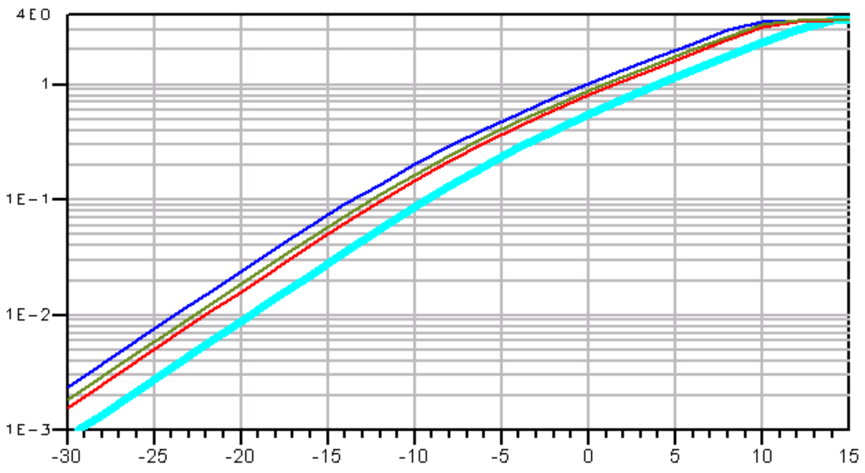
Figure 2: Antenna and Doubler Simulation



Analysis

Tune the antenna's feed point depth parameter ds1 and observe the output voltage at 12, 18 and 22 mm.

Figure 3: Doubler Vout vs. Pin Tuning Antenna Feed Point



Simulation of Spurious Signals

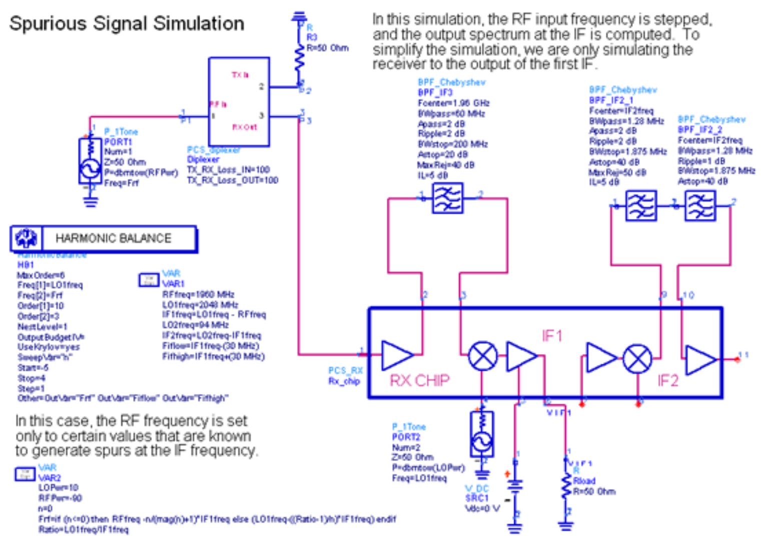
Location: $HPEESOF_DIR/examples/Com_Sys/Spur_Track_wrk

Objective

This example examines how a RF receiver generates spurious signals as the RF input signal frequency is swept.

Setup

1. "Rfsweep" simulates the amplitude of spurious signals that appear at the IF when the RF input signal is stepped in a range of values.
2. "MixerSpurs" is the same as RFsweep except that just the mixer is simulated, without any filtering.
3. "Rfnstep" is the same as "Rfsweep" except that a parameter n is stepped, and RF input frequencies are calculated from n such that a spurious intermodulation product always appears at the IF.
4. "PCS_RX" models the parts of the receiver that are on one IC, including many amplifiers and two mixers, although these are all behavioral models. Note that if all the deactivated components are activated, and the appropriate wires are added, this design can be simulated.
5. "PCS_RF_rcvr" is the same thing as "PCS_RX", except without all the deactivated components. It is not meant to be simulated by itself, but used as a subnetwork.
6. "PCS_diplexer" is a simple "diplexer" that passes signals from the transmitter to the RF_In terminal and passes received signals from the RF_In terminal to the receiver chain. It is not meant to be simulated by itself, but used as a subnetwork.



Spurious Signal Simulation

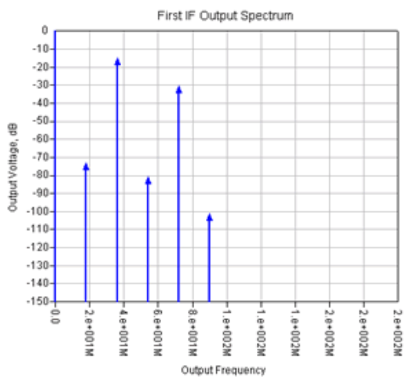
In this simulation, the RF input frequency is stepped, and the output spectrum at the IF is computed. To simplify the simulation, we are only simulating the receiver to the output of the first IF.

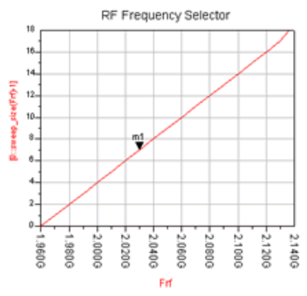
In this case, the RF frequency is set only to certain values that are known to generate spurs at the IF frequency.

```
R
R3
R=50 Ohm

P_1Tone
PORT1
Num=1
Z=50 Ohm
P=dbmtow(RFPwr)
Freq=Frf

PCS_diplexer
Diplexer
TX_RX_Loss_IN=100
TX_RX_Loss_OUT=100

HARMONIC BALANCE
HarmonicBalance
HB1
MaxOrder=6
Freq[1]=LO1freq
Freq[2]=Frf
Order[1]=10
Order[2]=3
NestLevel=1
OutputBudgetIV=
UseKrylov=yes
SweepVar="n"
Start=-6
Stop=4
Step=1
Other= OutVar="Frf" OutVar="Fiflow" OutVar="Fifhigh"

VAR
VAR1
RFfreq=1960 MHz
LO1freq=2048 MHz
IF1freq=LO1freq - RFfreq
LO2freq=94 MHz
IF2freq=LO2freq-IF1freq
Fiflow=IF1freq-(30 MHz)
Fifhigh=IF1freq+(30 MHz)

VAR
VAR2
LOPwr=10
RFPwr=-90
n=0
Frf=if (n<=0) then RFfreq -n/(mag(n)+1)*IF1freq else (LO1freq-((Ratio-1)/n)*IF1freq) endif
Ratio=LO1freq/IF1freq

BPF_Chebyshev
BPF_IF3
Fcenter=1.96 GHz
BWpass=60 MHz
Apass=2 dB
Ripple=2 dB
BWstop=200 MHz
Astop=20 dB
MaxRej=40 dB
IL=5 dB

BPF_Chebyshev
BPF_IF2_1
Fcenter=IF2freq
BWpass=1.28 MHz
Apass=2 dB
Ripple=2 dB
BWstop=1.875 MHz
Astop=40 dB
MaxRej=50 dB
IL=5 dB

BPF_Chebyshev
BPF_IF2_2
Fcenter=IF2freq
BWpass=1.28 MHz
Ripple=1 dB
BWstop=1.875 MHz
Astop=40 dB

PCS_RX
Rx_chip

P_1Tone
PORT2
Num=2
Z=50 Ohm
P=dbmtow(LOPwr)
Freq=LO1freq

V_DC
SRC1
Vdc=0 V

R
Rload
R=50 Ohm
```

Analysis

Figure: Output spectrum at first IF





Notes

- "RFsweep.dds" shows a single IF output spectrum at a time. The user can move a marker to a particular RF input frequency and the corresponding IF output spectrum will be displayed.

- "RFsweep2MHz.dds" is identical to "RFsweep.dds" except that the RF input frequency is changed in 2 MHz steps.
- "RFswpSpurTrack.dds" shows the results when the step size is set to 10 MHz. It opens slowly because it uses a complex function to extract the powers of all spurious signals that appear within a user-specified IF band. It also lists all of the spur frequencies (provided they fall within band) and their corresponding power levels.
- "RFswpSpurTrack2MHz.dds" shows the results when the step size is set to 2.01 MHz (not 2 MHz to prevent multiple intermodulation terms from landing on the same frequencies). It opens very quickly, because it uses a simple function (spur_track()) that only outputs the power of the highest spur within the user-specified frequency band.

SINAD Measurements

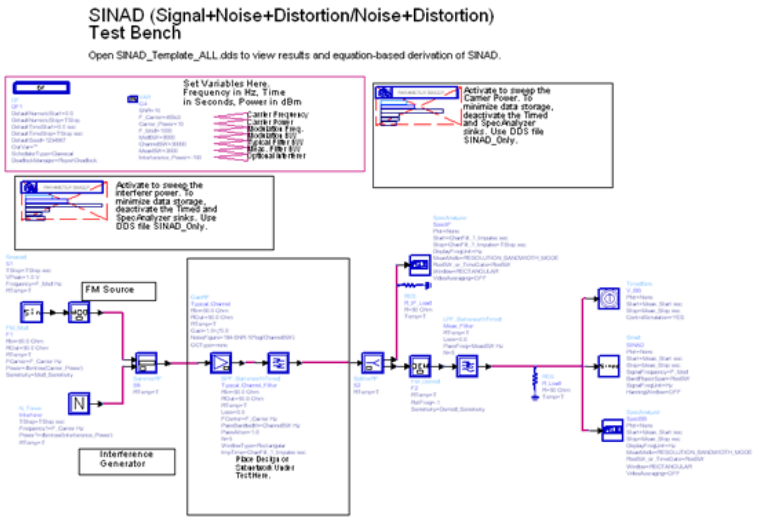
Location: $HPEESOF_DIR/examples/Com_Sys/SINAD_wrk

Objective

This example illustrates a test setup for Signal, Noise and Distortion (SINAD) measurements.

Setup

1. "SINAD_setup" contains a test setup for SINAD measurements. The device under test is a behavioral amplifier and channel filter, but a circuit-level design could be used instead. The setup includes two Parameter Sweep controllers. You can activate these to sweep the power levels of the carrier and interfering signals.
2. Agilent Ptolemy has a built-in SINAD measurement sink (output displayed in "SINAD_only.dds").SINAD_setup also has spectrum analyzers at the IF and BaseBand outputs, which are used to calculateSINAD directly. The user can view the calculations and compare to the built-in measurement in"SINAD_template_All.dds".


SINAD (Signal+Noise+Distortion/Noise+Distortion)
Test Bench
Open SINAD_Template_ALL.dds to view results and equation-based derivation of SINAD.
Set Variables Here. Frequency in Hz, Time in Seconds, Power in dBm
Carrier Frequency
Carrier Power
Modulation Freq.
Modulation BW
Typical Filter BW
Mod. Filter BW
Optional Interferer
Activate to sweep the Carrier Power. To minimize data storage, deactivate the Timed and SpecAnalyzer sinks. Use DDS file SINAD_Only.
Activate to sweep the interferer power. To minimize data storage, deactivate the Timed and SpecAnalyzer sinks. Use DDS file SINAD_Only.
FM Source
Interference Generator
Place Design or Cicuit Under Test Here.

Analysis

Figure 1: Input and output spectrum

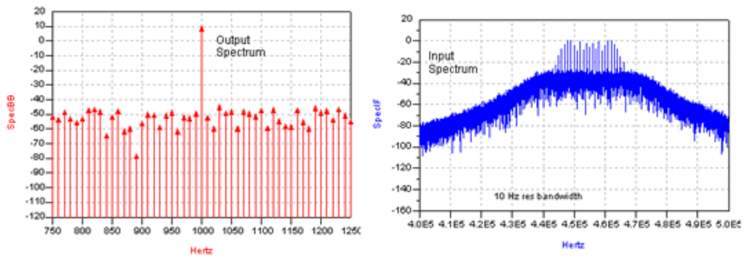
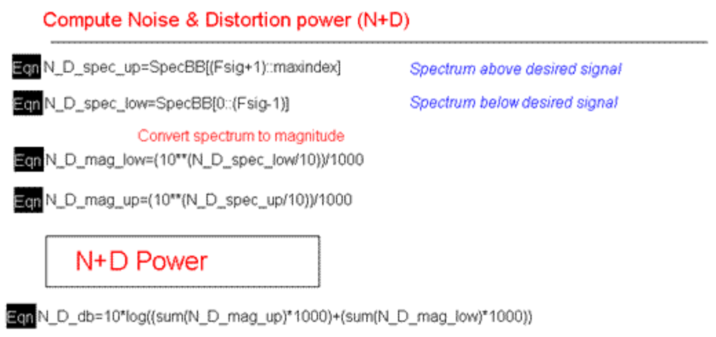


Figure 2: Calculation of SINAD using two methods



Compute Noise & Distortion power (N+D)

Eqn N_D_spec_up=SpecBB[(Fsig+1)::maxindex] Spectrum above desired signal

Eqn N_D_spec_low=SpecBB[0::(Fsig-1)] Spectrum below desired signal

Convert spectrum to magnitude

Eqn N_D_mag_low=(10**(N_D_spec_low/10))/1000

Eqn N_D_mag_up=(10**(N_D_spec_up/10))/1000

N+D Power

Eqn N_D_db=10*log((sum(N_D_mag_up)*1000)+(sum(N_D_mag_low)*1000))

Subband Speech Codec

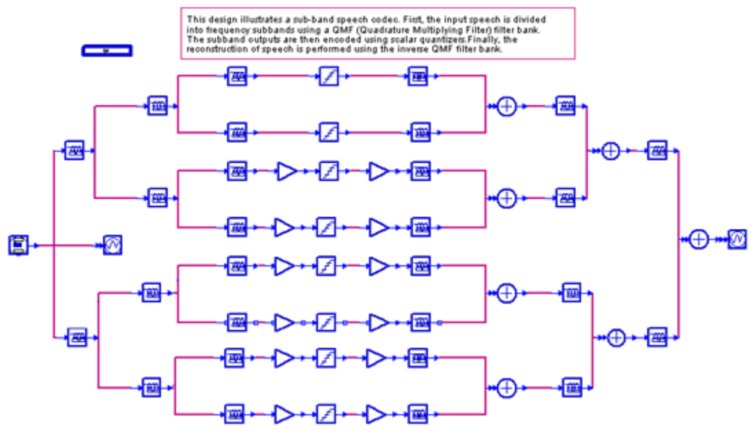
Location: $HPEESOF_DIR/examples/Com_Sys/SubbandCodec_wrk

Objective

This design illustrates a sub-band speech codec implemented using standard Agilent Ptolemy components.

Setup

In the codec, the input speech (8 bits/sample) is first divided into frequency subbands using a Quadrature Multiplying Filter (QMF) bank. The subband outputs are then encoded using scalar quantizers. Finally, the reconstruction of speech is performed using the inverse QMF filter bank. The output speech signal is 4 bits/sample.


This design illustrates a sub-band speech codec. First, the input speech is divided into frequency subbands using a QMF (Quadrature Multiplying Filter) filter bank. The subband outputs are then encoded using scalar quantizers.Finally, the reconstruction of speech is performed using the inverse QMF filter bank.

Analysis

Figure 1: Portion of input speech

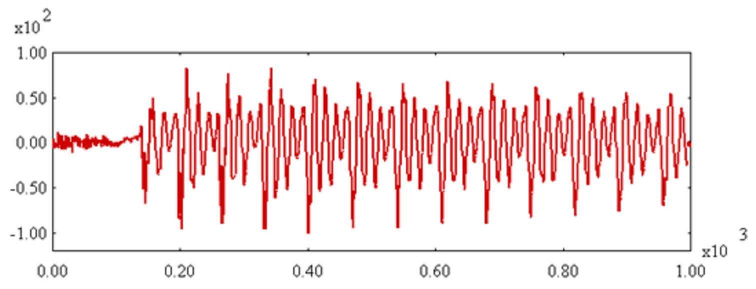
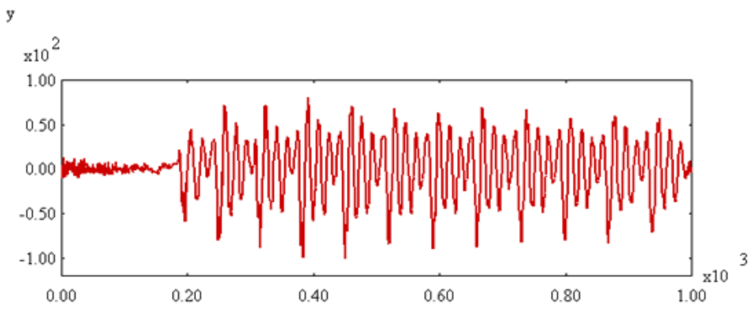


Figure 2: Portion of output speech



Test Benches for Evaluating PDC/TDMA Receivers

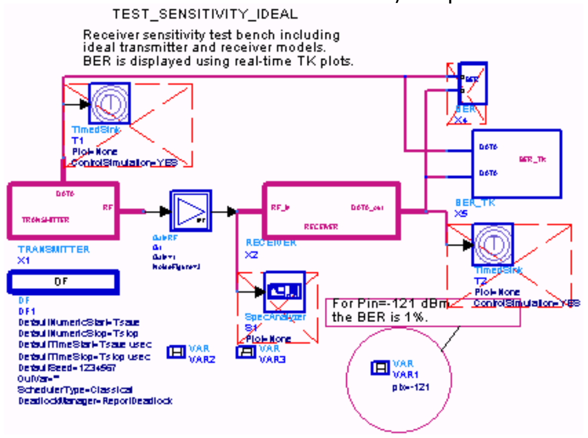
Location: $HPEESOF_DIR/examples/Com_Sys/PDC_wrk

Objective

This workspace contains a collection of test benches for evaluating PDC/TDMA receiver performance expressed as BER for various interference, noise and intermodulation distortion conditions. Some designs also include the effect of reciprocal mixing of out-of-band interfering signals with the receiver's LO phase noise.

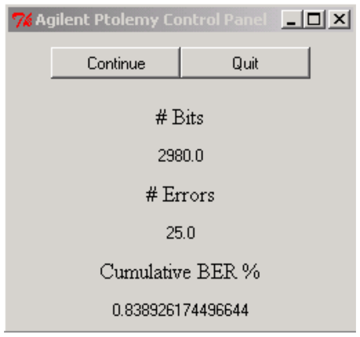
Setup

1. "TEST_SENSITIVITY" and "TEST_SENSITIVITY_IDEAL" are receiver sensitivity test benches including non-ideal and ideal transmitter and receiver models, respectively. BER is displayed using real-time TK plots.
2. "TEST_ADJCH" and "TEST_ADJCH_RM" are adjacent-channel rejection test benches including non-ideal transmitter and receiver models, the latter includes reciprocal mixing of out-of-band interfering signals with receiver LO phase noise.
3. "TEST_INTERMOD_RM", "TEST_INTERMOD_Rmpoly" and "TEST_INTERMOD_INDIRECT" are 3rd-order Intermodulation test benches including non-ideal transmitter and receiver models, and reciprocal mixing of out-of-band interfering signals with receiver LO phase noise. The "Indirect Method" injects the in band IM component. The "Direct Method" uses a 2-tone interference signal.
4. "TEST_CCR" is a co-channel interference test bench including non-ideal transmitter and receiver models. The co-channel interferer is derived from the transmitted signal by time delay and attenuation, ensuring that the interferer is not correlated with the wanted signal.
5. "TEST_SPUR" is a receiver spurious response test bench. Receiver spurs produce an in-band modulated interferer due to a NXM-order non-linearity. The non-linearity mixes the out-of-band interferers to in-band, which can change the modulation bandwidth of the component that falls in-band. The simulation preserves the bandwidth since the exact receiver architecture and frequency plan is unknown. The modulation BW doubles for the half/IF spur.



Analysis

Figure 1: Tk Output Display of simulation bits, error count and % BER



Bits

2980.0

Errors

25.0

Cumulative BER %

0.838926174496644

Figure 2: Two-tone output spectrum following the LNA

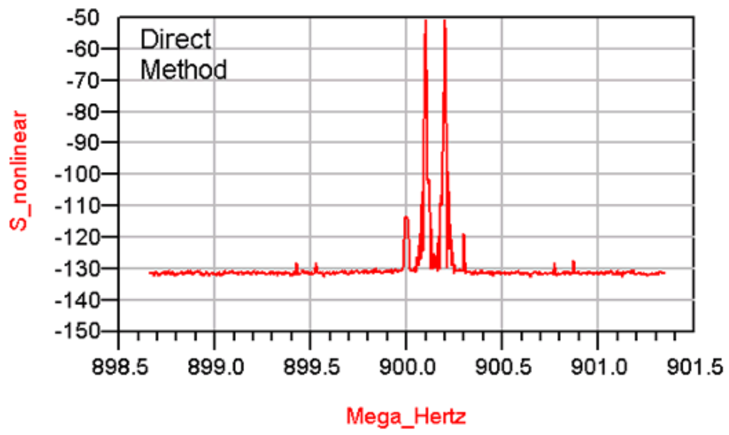
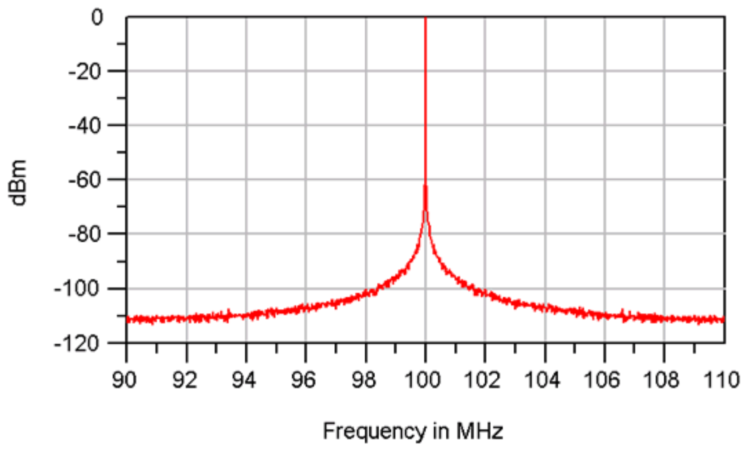


Figure 3: Oscillator phase noise spectrum



Various Examples on RF System-Level Simulations

Location: $HPEESOF_DIR/examples/Com_Sys/RFSystem_wrk

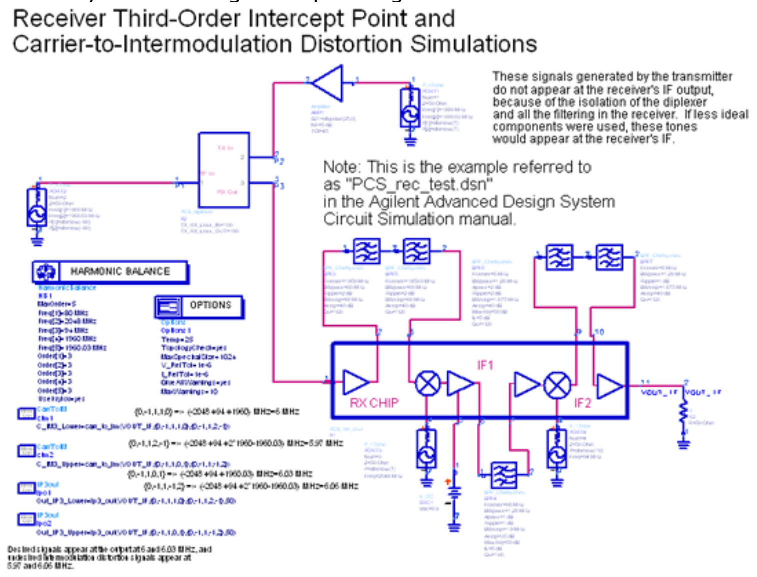
Objective

This example contains many different RF system-level simulations.

Setup

1. "Amp_TOI" shows how to simulate the output-referred third-order intercept point of an amplifier.
2. "Mixer_TOI" shows how to simulate the output-referred third-order intercept point of a mixer. Mixer_Noise1 shows a simple receiver noise figure simulation with both the LO and RF included as large-signal tones. "Mixer_Noise2" shows an alternative method of simulating noise figure, that does not require a large-signal tone at the RF input. The result is the same.
3. "LS_Mix_Sweep" simulates the conversion gain of a receiver as a function of RF input frequency.

4. "ACPR_IS_136" is a set-up to simulate an amplifier with a pi/4 DQPSK-modulated signal, corresponding to the IS-136 standard. The user may set the source power, modulation data bit sequence, source and load impedances, and other parameters. The set-up includes a pair of filters for generating an undistorted signal for error vector magnitude calculation, as well as a set of filter banks so the effects of receive-side filtering in the main and adjacent channels can be included. There are several different data displays for displaying the results from this simulation.
5. "Linear_Budget" simulates the tranducer power gain from PORT1 to pin 1 of each component, and the noise figure from PORT1 to the input of each component, in a simple receiver. For more detailed information, refer to the "Linear_Budget_wrk". "Linear_Sweep" simulates the "linear" conversion gain, phase shift and group delay of the same receiver.
6. "PCS_Rx_Budget" simulates the power and gain budget of a PCS receiver. The simulation results are shown in "PCS_Rx_Bud_Gain.dds" and "PCS_Rx_Bud_Power.dds".
7. "PCS_RX" models the parts of the receiver that are on one IC, including many amplifiers and two mixers, although these are all behavioral models. Note that if all the deactivated components are activated, and the appropriate wires are added, this design can be simulated. "PCS_RF_rcvr" is the same thing as "PCS_RX", except without all the deactivated components. It is not meant to be simulated by itself.
8. "PCS_diplexer" is a simple diplexer that passes signals from the transmitter to the RF In terminal and passes received signals from the RF In terminal to the receiver chain. It is not meant to be simulated by itself.
9. "Diplexer_test" simulates the S-parameters from the RF In terminal to the receiver out terminal.
10. "PCS_Tx" is a transmitter chain, without filters. It is not meant to be simulated by itself, and is not used in any other design.
11. "PCS_Rx_TOI_test" simulates the third-order intercept point of the receiver, with two tones at the input. Two tones are included from the transmitter, and if the isolation of the diplexer is reduced, they make the intermodulation distortion of the receiver worse by over-driving the input stages.


Receiver Third-Order Intercept Point and
Carrier-to-Intermodulation Distortion Simulations
These signals generated by the transmitter do not appear at the receiver's IF output, because of the isolation of the diplexer and all the filtering in the receiver. If less ideal components were used, these tones would appear at the receiver's IF.
Note: This is the example referred to as "PCS_rec_test.dsn" in the Agilent Advanced Design System Circuit Simulation manual.
HARMONIC BALANCE
OPTIONS
IF1
IF2
RX CHIP
Vout_IF
Desired signals appear at the output at 6 and 6.03 MHz, and undesired intermodulation distortion signals appear at 5.97 and 6.06 MHz.

Analysis

Figure 1: Upper and lower adjacent-channel power ratios when the receive-side filtering present in the IS-136 system is included

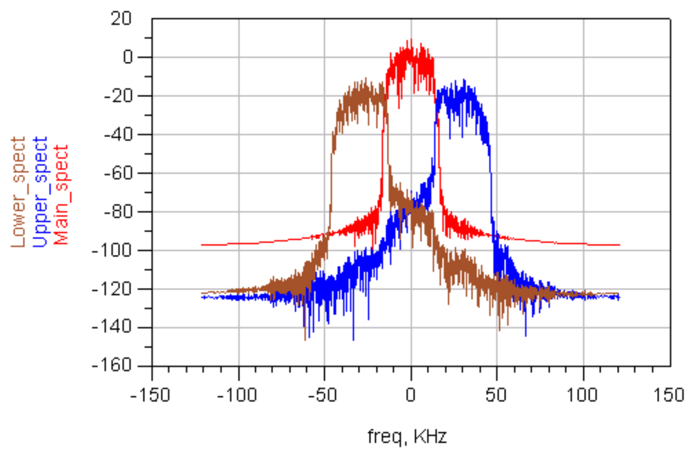
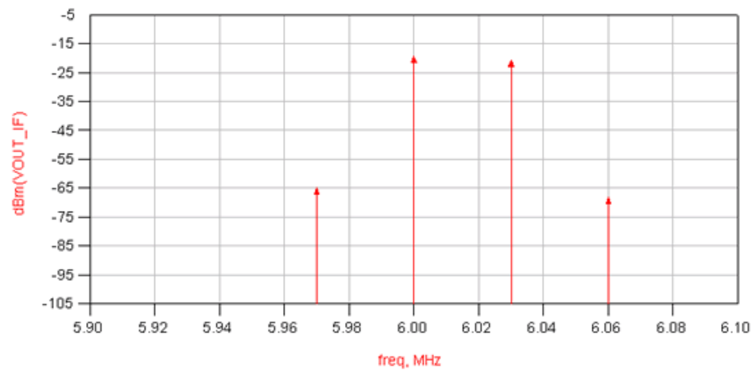


Figure 2: Spectrum at output in design "PCS_Rx_TOI_test"



Wideband CDMA Test Modulator

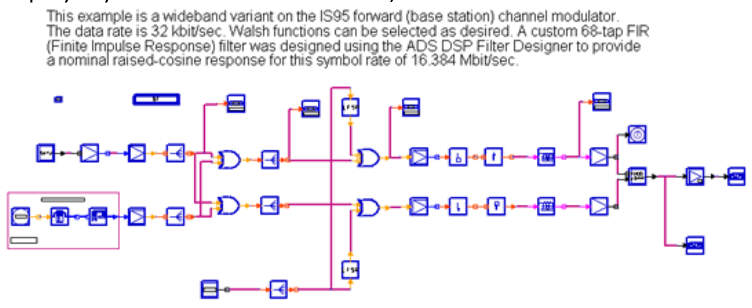
Location: /examples/Com_Sys/widebandCDMA_wrk

Objective

This example illustrates a simple variation on an IS-95 Forward Channel modulator. The chip rate is 16.384 Mcps and the input data rate is 32 ksps. A Finite Impulse Response (FIR) filter was designed using the ADS Digital Filter Designer. Note it does not include or reference recent WCDMA or "3GPP" specifications. For conforming applications, please refer to the WCDMA and WCDMA3G Design Libraries for ADS.

Setup

This example is a wideband variant on the IS95 forward (base station) channel modulator. Pseudo-random I and Q bit sequences are generated and combined with a user-selectable Walsh code. The sequences are then combined with PN short codes generated by linear-feedback shift registers. After decimation and zero insertion to change rectangular pulses to impulses, FIR pulse shaping is performed, followed by quadrature modulation onto a carrier. A spectrum analyzer provides a normalized and video-averaged output spectrum display. symbol rate of 16.384 Mbit/sec.



This example is a wideband variant on the IS95 forward (base station) channel modulator. The data rate is 32 kbit/sec. Walsh functions can be selected as desired. A custom 68-tap FIR (Finite Impulse Response) filter was designed using the ADS DSP Filter Designer to provide a nominal raised-cosine response for this symbol rate of 16.384 Mbit/sec.

Analysis

Figure 1: Filtered spread data

Figure 1: Filtered spread data

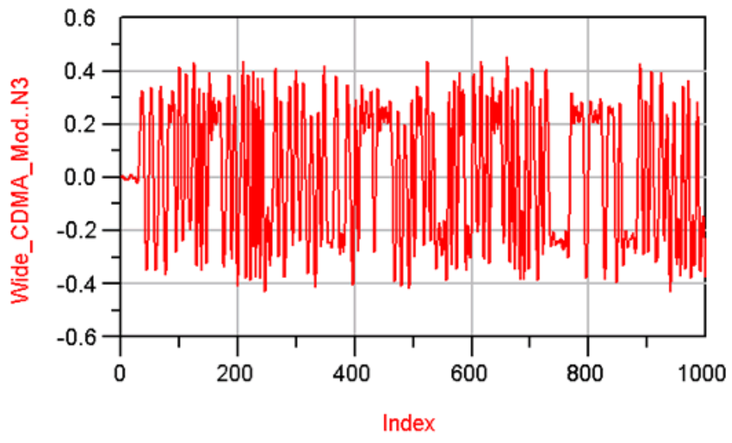


Figure 2: Modulator Q envelope

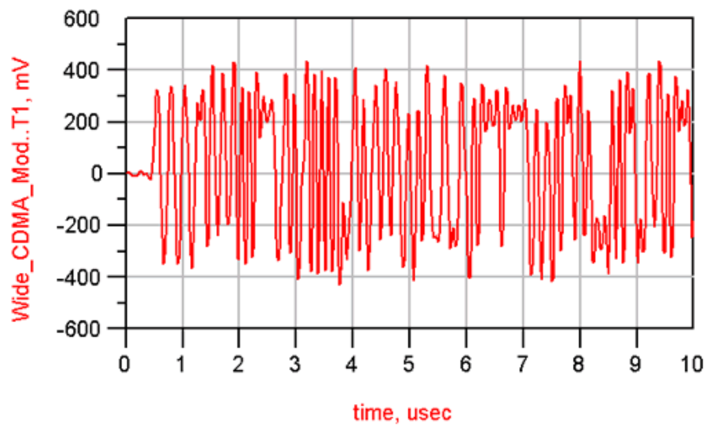
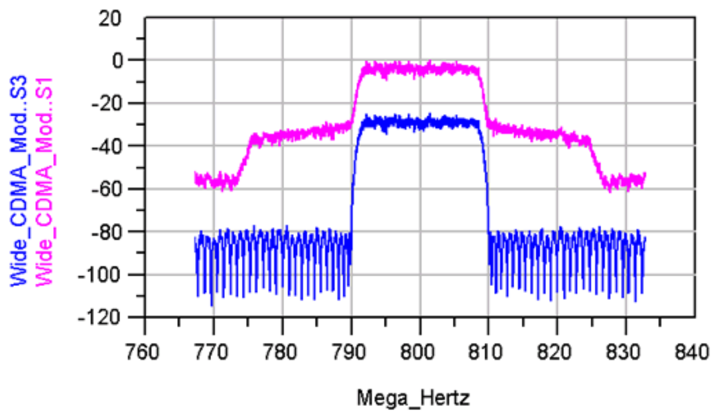


Figure 3: Output spectrum



Notes

- This example does not include or reference recent WCDMA or "3GPP" specifications. For conforming applications, please refer to the WCDMA and WCDMA3G Design Libraries for ADS.

Wireless MAN 802.16d Transmitter Test Project

Location: $HPEESOF_DIR/examples/Com_Sys/WMAN_802_16d_TX_wrk

Objective

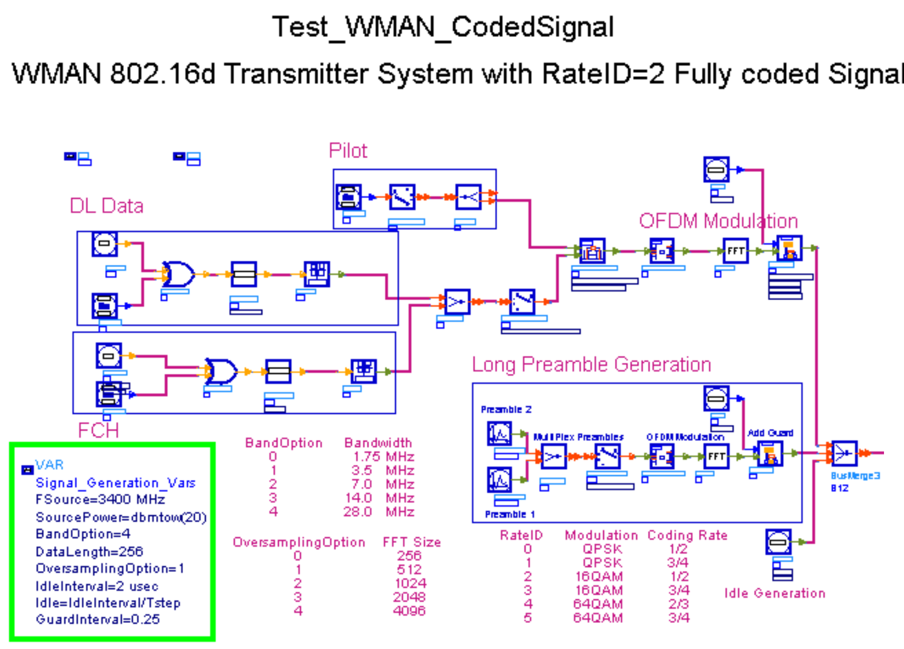
As a typical application of the Numeric Advanced Comm library, Wireless Metropolitan

Access Networks (WMAN) designs have been created for Wireless MAN market based on the IEEE 802.16d standard. These designs use the new Numeric Advanced Comm components, basic ADS components, as well as imported Matlab components. These application designs are focused on the physical layer of WMAN systems and are intended to be a baseline system for designers to get an idea of what nominal or ideal system performance would be. Evaluations can be made regarding degraded system performance due to system impairments that may include non-ideal component performance.

Setup

1. Fully-coded signal generation: Test_WMAN_CodedSignals shows how to build OFDM frame structure for the WMAN frequency division duplex downlink (FDD DL) system in ADS. Main components in the subsystem level include long preamble generation, frame control header (FCH) and FDD DL data generation, OFDM modulation, multiplexing, RF modulation, and measurements. Signals are fully coded by RS-CC encode based on the 16d standard.
2. WMAN RF Test: Test_WMAN_RFSource provides a test environment for a WMAN FDD DL transmitter system. Users can place their own DUT in the design and measure waveforms, spectrum, power CCDF, and constellation.
3. Downloading WMAN Signals to ESG: Test_WMAN_ESG is used for sending WMAN framed data to an ESG for testing devices

left side of the schematic



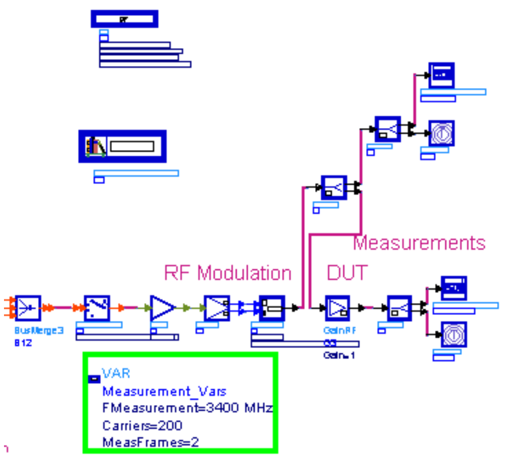
VAR
Signal_Generation_Vars
FSource=3400 MHz
SourcePower=dbmtow(20)
BandOption=4
DataLength=256
OversamplingOption=1
IdleInterval=2 usec
Idle=IdleInterval/Tstep
GuardInterval=0.25

BandOption	Bandwidth
0	1.75 MHz
1	3.5 MHz
2	7.0 MHz
3	14.0 MHz
4	28.0 MHz

OversamplingOption	FFT Size
0	256
1	512
2	1024
3	2048
4	4096

RateID	Modulation	Coding Rate
0	QPSK	1/2
1	QPSK	3/4
2	16QAM	1/2
3	16QAM	3/4
4	64QAM	2/3
5	64QAM	3/4

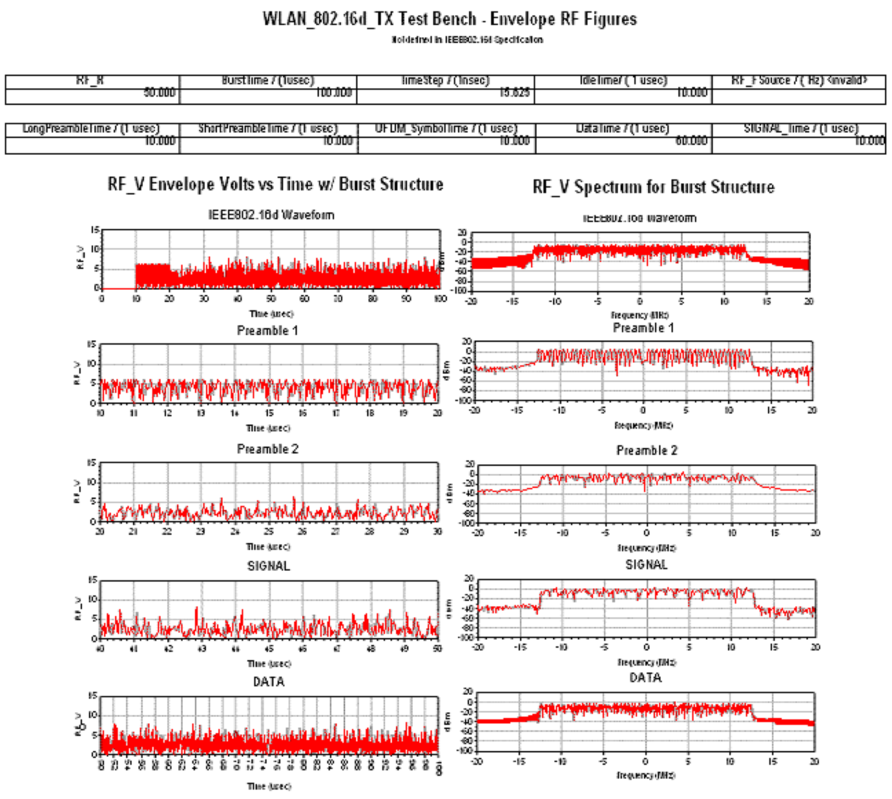
right side of the schematic



VAR
Measurement_Vars
FMeasurement=3400 MHz
Carriers=200
MeasFrames=2

Analysis

Figure 1. WMAN 16d OFDM Signal Waveforms and Spectrum



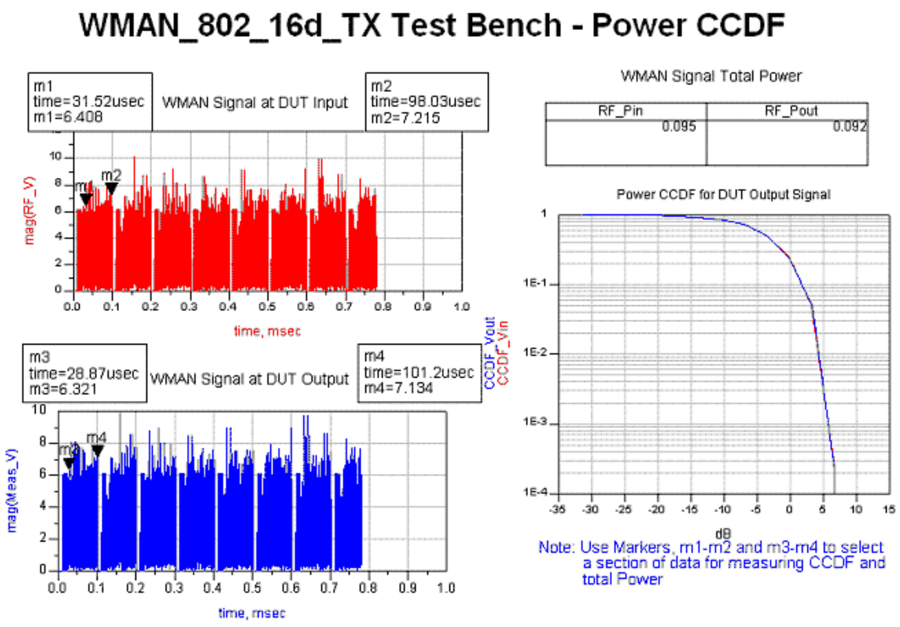
WLAN_802.16d_TX Test Bench - Envelope RF Figures

Not defined in IEEE802.16d Specification

RF_R	BurstTime / (1usec)	TimeStep / (1nsec)	IdleTime/ (1 usec)	RF_FSource / (Hz) <invalid>
50.000	100.000	15.625	10.000	

LongPreambleTime / (1 usec)	ShortPreambleTime / (1 usec)	OFDM_SymbolTime / (1 usec)	DataTime / (1 usec)	SIGNAL_Time / (1 usec)
10.000	10.000	10.000	60.000	10.000

Figure 2. WMAN 16d OFDM Signal Power and CCDF



WMAN_802_16d_TX Test Bench - Power CCDF

m1
time=31.52usec
m1=6.408

m2
time=98.03usec
m2=7.215

m3
time=28.87usec
m3=6.321

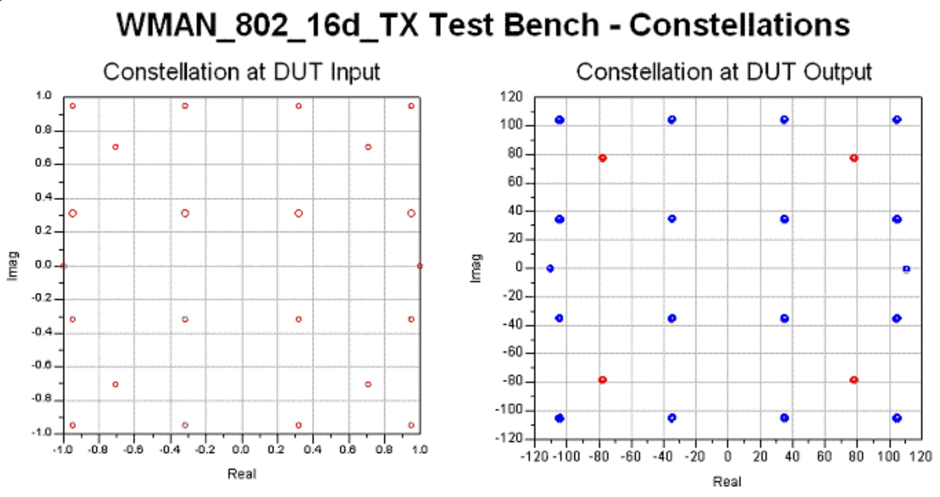
m4
time=101.2usec
m4=7.134

WMAN Signal Total Power

RF_Pin	RF_Pout
0.095	0.092

Note: Use Markers, m1-m2 and m3-m4 to select a section of data for measuring CCDF and total Power

Figure 3. WMAN 16d OFDM Signal Constellations



Notes

- Designs are based on Draft IEEE Standard for Metropolitan Area Networks IEEE P802.16-REVd/D2-2003, Dec, 2003.
- Designs are for WMAN OFDM Transmitter Simulation and Test only.

Connected Solutions Examples

Examples of the basic functions of the Sequencer controller and the concept of 3GPP and WLAN Connected Solution BER measurements.

- *3GPP Uplink BER Receiver Characteristics Test* (examples)
- *WLAN 802.11a Receiver Minimum Input Level Sensitivity Test* (examples)

3GPP Uplink BER Receiver Characteristics Test

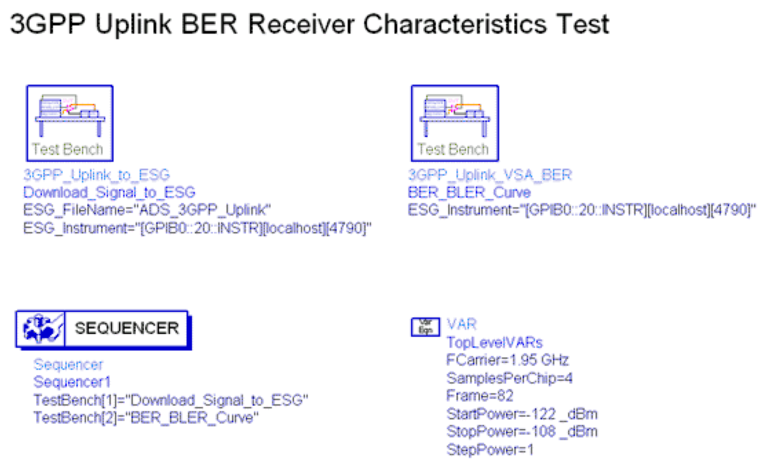
Location: $HPEESOF_DIR/examples/Connected_Solutions/3GPP_BER_wrk

Objective

This example demonstrates the basic functions of the *Sequencer controller* (examples) and the concept of 3GPP Connected Solution BER measurement. It consists of two test benches. The first test bench generates and initiates playback of the 3GPP signal. The second test bench sweeps the RF power of the 3GPP signal and uses a VSA to capture the IQ stream, which is then processed by the simulator to compute the BER of the receiver. An E4438C ESG (or PSG) and a 89600 Series VSA (e.g. 89641 VSA) instrument are needed for simulation. Currently this example works only on the PC platform.

Setup

1. "3GPP" Top-level design with two sub-networks - "3GPP_Uplink_to_ESG" and "3GPP_Uplink_VSA_BER". This design also contains a *Sequencer Controller* (examples) to control the simulation flow and simulate the "3GPP_Uplink_to_ESG" first (downloading signal to ESG), and then "3GPP_Uplink_VSA_BER" (generating BER curve).
2. "3GPP_Uplink_to_ESG" This design downloads the 3GPP Uplink signal to the ESG and stores reference signal/data in text files for BER measurement made by "3GPP_Uplink_VSA_BER". This design can be simulated on its own using the VAR "Activate_to_simulate_standalone" option.
3. "3GPP_Uplink_VSA_BER" This design obtains a BER waterfall curve by iteratively acquiring data from a VSA, making BER measurements, setting the ESG output power for next point, and then acquiring data from VSA again. This design can be simulated on its own using the VAR "Activate_to_simulate_standalone" option.
4. "3GPP.dds" Plots the DTCH BER (after error correction) and DPDCH BER. Both are displayed in red. At a specification defined threshold of 0.1% the DTCH BER is plotted in blue.
   Behavioral Model Examples



3GPP_Uplink_to_ESG
Download_Signal_to_ESG
ESG_FileName="ADS_3GPP_Uplink"
ESG_Instrument="[GPIB0::20::INSTR][localhost][4790]"

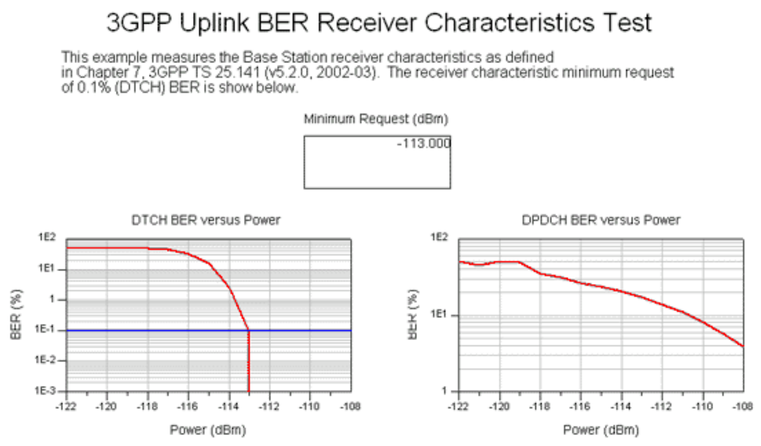
3GPP_Uplink_VSA_BER
BER_BLER_Curve
ESG_Instrument="[GPIB0::20::INSTR][localhost][4790]"

Sequencer
Sequencer1
TestBench[1]="Download_Signal_to_ESG"
TestBench[2]="BER_BLER_Curve"

TopLevelVARs
FCarrier=1.95 GHz
SamplesPerChip=4
Frame=82
StartPower=-122 _dBm
StopPower=-108 _dBm
StepPower=1

Analysis

Figure 1: DTCH BER and DPDCH BER Plots



3GPP Uplink BER Receiver Characteristics Test

This example measures the Base Station receiver characteristics as defined in Chapter 7, 3GPP TS 25.141 (v5.2.0, 2002-03). The receiver characteristic minimum request of 0.1% (DTCH) BER is show below.

Notes

- Settings for the VSA_89600_Source in "3GPP_Uplink_VSA_BER"
- OutputType=Timed
- Pause=NO
- VSATrace (depends on the setup file, usually uses "B")
- RepeatData=Reacquire
- TStep (correct Ptolemy time step)
- SetupFile (always specify setup files)
- RecordingFile (always leave it blank)
- SetupUse=Always
- AutoCapture=YES
- Settings for the 3GPPFDD_RF_Uplink_Receiver in "3GPP_Uplink_VSA_BER"
- ChannelType=Fading
- ChannelInfo=Estimated
- Single VSA setup file (Reacquire_1950MHz_12MSpan_-45dBm_1000ms.set) provided with this example.
- Sets the VSA to 1950 MHz Center Frequency, 12 MHz Span, -45 dBm Range.
- VSA will acquire about 1 second of data, enough to make 3GPP BER measurements over up to 100 (radio) frames. Change the recording length in the setup file to acquire more or less data.
- The acquired data can support TStep equal to or larger than 1/3.84/4 usec (up to 4 samples per chip). To support smaller TStep (higher oversampling rate), please increase "Span" in the setup file.
- For information on sequencing simulations, please refer to the Sequencer Controller documentation.
- For information on ESG connectivity, please refer to the ESG Sink documentation.
- For VSA connectivity, please refer to the VSA_89600_Source documentation.
- For Connection Manager details, please refer to the Connection Manager documentation.
- For 3GPP Wireless Design Library details, please refer to the 3GPP documentation.

WLAN 802.11a Receiver Minimum Input Level Sensitivity Test

Location: $HPEESOF_DIR/examples/Connected_Solutions/WLAN_BER_wrk

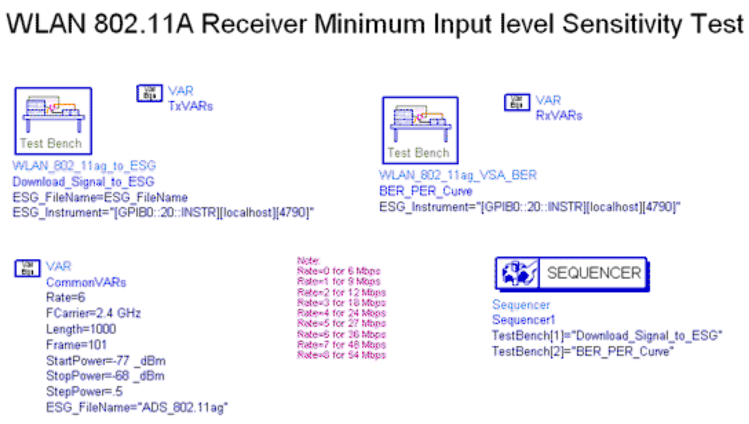
Objective

This example demonstrates the basic functions of the *Sequencer Controller* (examples) and the concept of WLAN Connected Solution BER measurement. It consists of two test benches. The first test bench generates and initiates playback of the WLAN signal. The second test bench sweeps the RF power of the WLAN signal and uses a VSA to capture the IQ stream, which is then processed by the simulator to compute the BER and PER of the receiver. An E4438C ESG (or PSG) and a 89600 Series VSA (e.g. 89641 VSA) instrument are needed for simulation. Currently this example works only on the PC platform.

Setup

1. "WLAN" Top-level design with two sub-networks - "WLAN_802_11ag_to_ESG" and "WLAN_802_11ag_VSA_BER". This design also contains a Sequencer controller to control the simulation flow and simulate the "WLAN_802_11ag_to_ESG" first (downloading signal to ESG), and then "WLAN_802_11ag_VSA_BER" (generating BER curve).

2. "WLAN_802_11ag_to_ESG" This design downloads the WLAN 802.11a (or 802.11g) signal to the ESG. This design can be simulated on its own using the VAR "Activate_to_simulate_standalone" option.
3. "WLAN_802_11ag_VSA_BER" This design obtains a PER vs. Power curve by iteratively acquiring data from a VSA, making PER measurements, setting the ESG output power for next point, and then acquiring data from VSA again. This design can be simulated on its own using the VAR "Activate_to_simulate_standalone" option.
4. "WLAN.dds" Plots the BER and PER. Both are displayed in red. At a specification defined threshold of 10% the PER is plotted in blue.
   Behavioral Model Examples

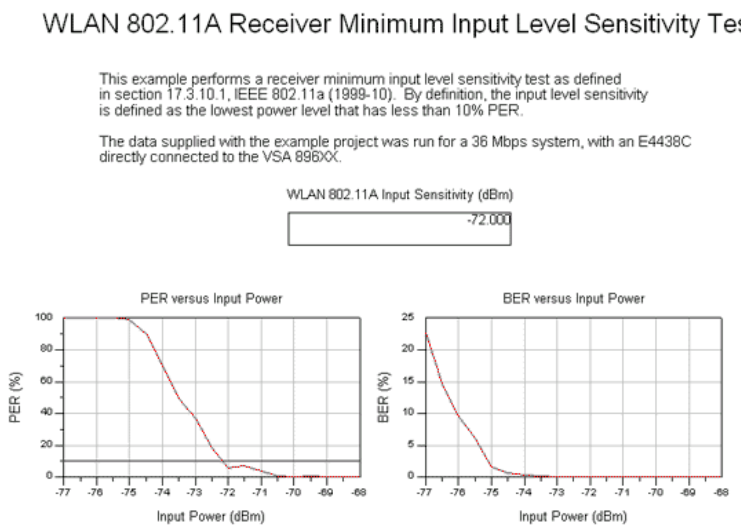


WLAN 802.11A Receiver Minimum Input level Sensitivity Test

```
Test Bench
WLAN_802_11ag_to_ESG
Download_Signal_to_ESG
ESG_FileName=ESG_FileName
ESG_Instrument="[GPIB0::20::INSTR][localhost][4790]"

VAR
TxVARs

Test Bench
WLAN_802_11ag_VSA_BER
BER_PER_Curve
ESG_Instrument="[GPIB0::20::INSTR][localhost][4790]"

VAR
RxVARs

VAR
CommonVARs
Rate=6
FCarrier=2.4 GHz
Length=1000
Frame=101
StartPower=-77 _dBm
StopPower=-68 _dBm
StepPower=.5
ESG_FileName="ADS_802.11ag"

Note:
Rate=0 for 6 Mbps
Rate=1 for 9 Mbps
Rate=2 for 12 Mbps
Rate=3 for 18 Mbps
Rate=4 for 24 Mbps
Rate=5 for 27 Mbps
Rate=6 for 36 Mbps
Rate=7 for 48 Mbps
Rate=8 for 54 Mbps

SEQUENCER
Sequencer
Sequencer1
TestBench[1]="Download_Signal_to_ESG"
TestBench[2]="BER_PER_Curve"
```

Analysis

Figure 1: BER and PER Plots



WLAN 802.11A Receiver Minimum Input Level Sensitivity Test

This example performs a receiver minimum input level sensitivity test as defined in section 17.3.10.1, IEEE 802.11a (1999-10). By definition, the input level sensitivity is defined as the lowest power level that has less than 10% PER.

The data supplied with the example project was run for a 36 Mbps system, with an E4438C directly connected to the VSA 896XX.

WLAN 802.11A Input Sensitivity (dBm)

-72.000

Notes

- Settings for the VSA_89600_Source in WLAN_802_11ag_VSA_BER
- OutputType=Timed
- Pause=NO
- VSATrace (depends on the setup file, usually uses "B")
- RepeatData=Reacquire
- TStep (correct Ptolemy time step)
- SetupFile (always specify setup files)
- RecordingFile (always leave it blank)
- SetupUse=Always
- AutoCapture=YES
- Single VSA setup file (Reacquire_-45dBm_26ms.set) provided with this example.
- Sets the VSA to 2.4 GHz Center Frequency, 36 MHz Span, -45 dBm Range.
- VSA will acquire about 26 milliseconds data, enough to make WLAN BER/PER measurements over up to 100 packets (frames). Change the recording length in the setup file to acquire more or less data.
- The acquired data can support TStep equal to or larger than 22 nsec.

- For information on sequencing simulations, please refer to the Sequencer Controller documentation.
- For information on ESG connectivity, please refer to the ESG Sink documentation.
- For VSA connectivity, please refer to the VSA_89600_Source Doc documentation.
- For Connection Manager details, please refer to the Connection Manager documentation.
- For WLAN Wireless Design Library details, please refer to the WLAN documentation.

Design Kit Examples

Examples of the use of a design kit within ADS, including a tutorial demonstration of a design kit architecture.

- *Demonstration PDK Used by Other ADS Examples* (examples)
- *Using the Graphical Cell Compiler to Create a FET* (examples)

Demonstration PDK Used by Other ADS Examples

Location: $HPEESOF_DIR/examples/DesignKit/DemoKit

Objective

This is a demonstration PDK that is used by other ADS example workspaces, including these examples:

- examples/MW_Ckts/MMIC_Amp_wrk
- examples/MW_Ckts/MMIC_Osc_wrkIt is not an ADS example.

Using the Graphical Cell Compiler to Create a FET

Location: $HPEESOF_DIR/examples/DesignKit/GCC_FET_wrk

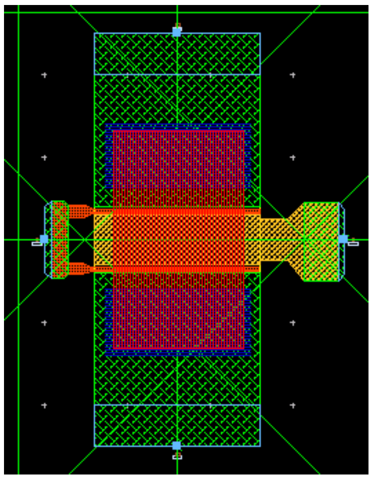
Objective

Show a FET layout created using the Graphical Cell Compiler.

Notes

This example has one layout, of a FET that is in the DemoKit included with ADS. You cannot have the DemoKit design kit enabled when using this workspace.

Figure: FET Layout



Digital Signal Processing Examples

Examples of how DSP tools and the extensive element and design libraries can be used to address complex digital signal processing tasks.

- *16-Point FFT in Synthesizable Logic* (examples)
- *16-QAM Modem* (examples)
- *Adaptive Differential Pulse Code Modulation (ADPCM) Codec* (examples)
- *A Low Pass Filter* (examples)
- *DSP Cosimulation with Transient Circuit Simulation* (examples)
- *Equalized 16-QAM with Multipath and Phase Noise* (examples)
- *Example of using A-D-D-A Models* (examples)
- *Eye Diagram with Variable Noise Generator* (examples)
- *MATLAB and Agilent Ptolemy Co-simulation* (examples)
- *Matlab Cosimulation* (examples)
- *PLL Demo 1 in DSP* (examples)
- *PLL Demo 2 in DSP* (examples)
- *Sine and Cosine Wave Generator* (examples)
- *Timed QAM Modem* (examples)

16-Point FFT in Synthesizable Logic

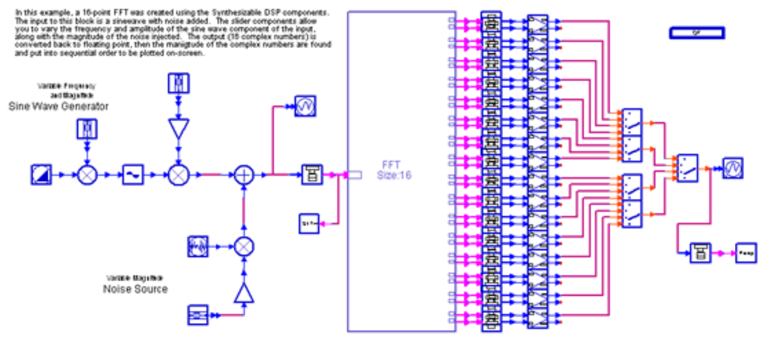
Location: $HPEESOF_DIR/examples/DSP/dsp_demos_wrk/FFT_SYNTH

Objective

This example is to illustrate the implementation of a simple 16-point fast Fourier transform (FFT) in synthesizable logic.

Setup

The input to this block is a sinewave with noise added. The slider components allow the user to vary the frequency and amplitude of the sine wave component of the input, along with the magnitude of the injected noise. The output (16 complex numbers) is converted back to floating point. Then, the magnitudes of the complex numbers are found and put into sequential order to be plotted on-screen.



In this example, a 16-point FFT was created using the Synthesizable DSP components. The input to this block is a sinewave with noise added. The slider components allow you to vary the frequency and amplitude of the sine wave component of the input, along with the magnitude of the noise injected. The output (16 complex numbers) is converted back to floating point, then the manigtude of the complex numbers are found and put into sequential order to be plotted on-screen.

Analysis

Figure 1: Input waveform, noise added

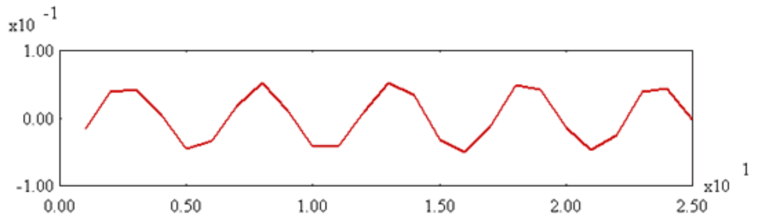


Figure 2: 16-Pt FFT spectrum output

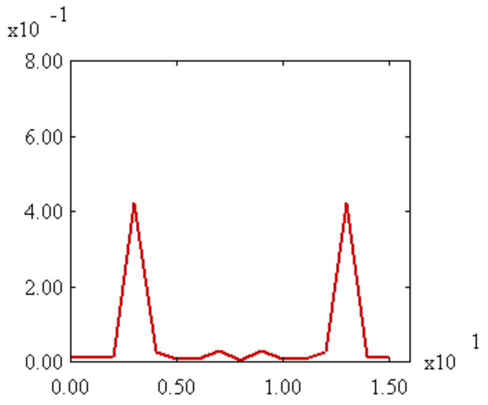
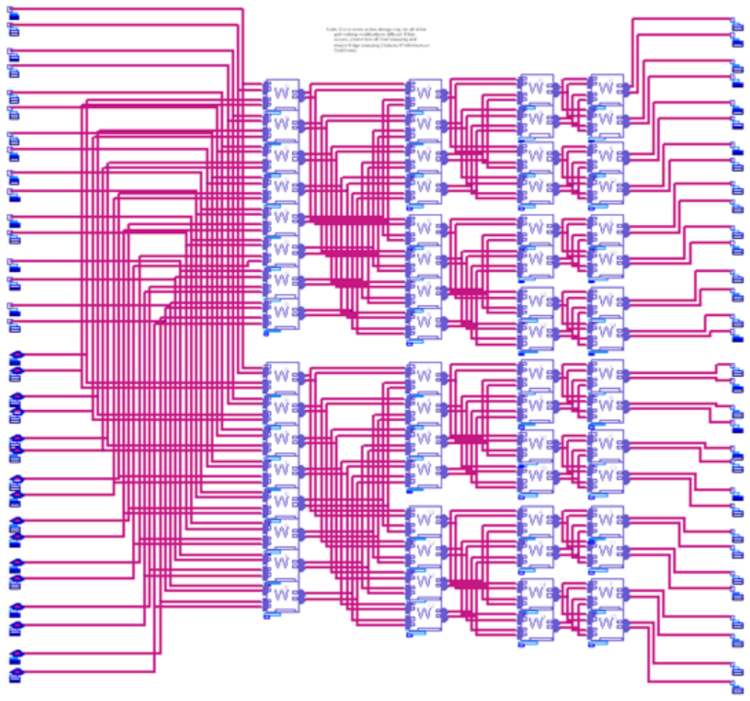


Figure 3: Butterfly array details



16-QAM Modem

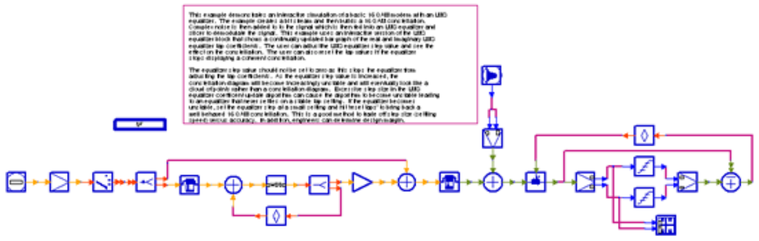
Location: $HPEESOF_DIR/examples/DSP/dsp_demos_wrk/MODEM

Objective

This example demonstrates an interactive simulation of a basic 16 Quadrature Amplitude Modulation (QAM) modem with a Least Mean Squares (LMS) equalizer. This example uses an interactive version of the LMS equalizer block that shows a continually updated bar graph of the real and imaginary LMS equalizer tap coefficients. The user can adjust the LMS equalizer step value and see the effect on the constellation. The user can also reset the tap values if the equalizer stops displaying a coherent constellation.

Setup

The example creates a bit stream and then builds a 16 QAM constellation. Complex noise is then added to the signal which is then fed into an LMS equalizer and slicer to demodulate the signal.
Behavioral Model Examples



Analysis

Figure 1: Output constellation

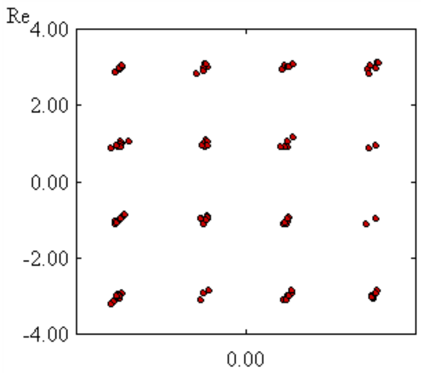
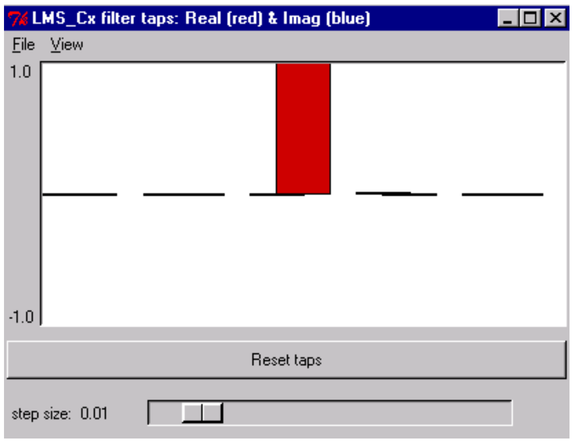


Figure 2: Equalizer taps



Notes

- The equalizer step value should not be set to zero as this stops the equalizer from adjusting the tap coefficients. As the equalizer step value is increased, the constellation diagram will become increasingly unstable and will eventually look like a cloud of points rather than a constellation diagram. Excessive step size in the LMS equalizer coefficient update algorithm can cause the algorithm to become unstable leading to an equalizer that never settles on a stable tap setting. If the equalizer becomes unstable, set the equalizer step at a small setting and hit "reset taps" to bring back a well behaved 16 QAM constellation. This is a good method to trade off step size (settling speed) versus accuracy.

Adaptive Differential Pulse Code Modulation (ADPCM) Codec

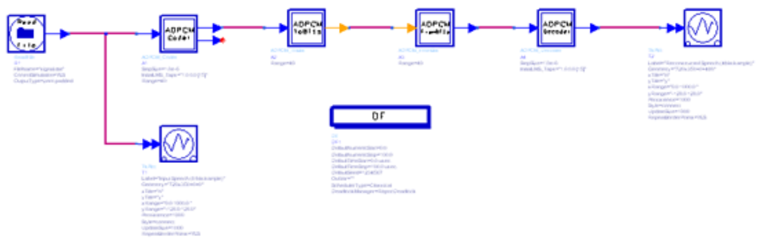
Location: $HPEESOF_DIR/examples/DSP/ADPCMCodec_wrk

Objective

This design demonstrates an ADPCM codec using Agilent Ptolemy components. The input speech has a bit rate of 64kbit/sec and the compressed speech is 32kbit/sec. Input sample rate is 8kHz, word length 8 bit. Output sample rate is 8kHz, word length 4 bit.

Setup

1. A digitized speech segment, at 8bits/sample, is read from the data file signal.dat and is displayed using Tcl/Tk plots for reference.
2. After ADPCM encoding, the signal is converted to a bitstream and then recovered using ADPCMToBits and ADPCMFromBits.
3. The reconstructed speech at 4 bits/sample after decoding is displayed for comparison.
   Behavioral Model Examples



Analysis

Figure 1: Portion of input waveform, 8 bits/sample

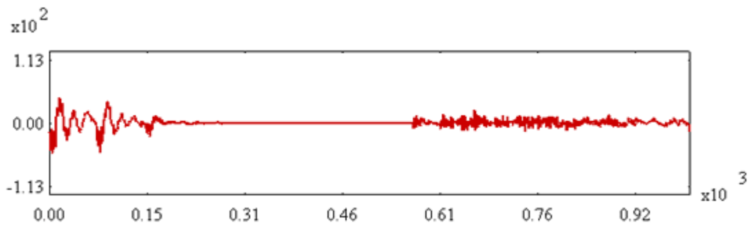
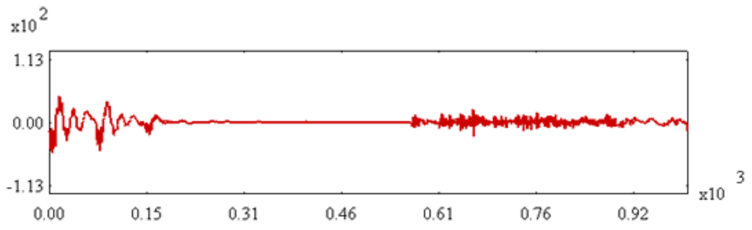


Figure 2: Portion of output waveform, 4 bits/sample



Notes

- Simulation used: Data Flow. TKPlot is used for display.

A Low Pass Filter

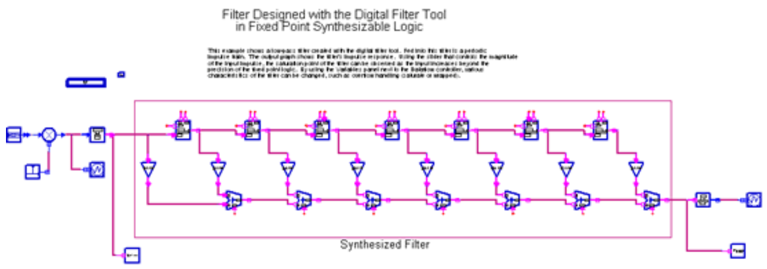
Location: $HPEESOF_DIR/examples/DSP/dsp_demos/SYN_FILT

Objective

This example shows the evaluation of a low-pass filter. By using the VAR(iable) component, various characteristics of the filter can be changed, such as overflow handling (saturate or wrapped).

Setup

A periodic impulse train is fed into the low-pass filter. The output graph shows the filter's impulse response. Using the slider that controls the magnitude of the input impulse, the saturation point of the filter can be observed as the input increases beyond the precision of the fixed point logic.


Filter Designed with the Digital Filter Tool
in Fixed Point Synthesizable Logic
Synthesized Filter

Analysis

Figure 1: Input impulse train

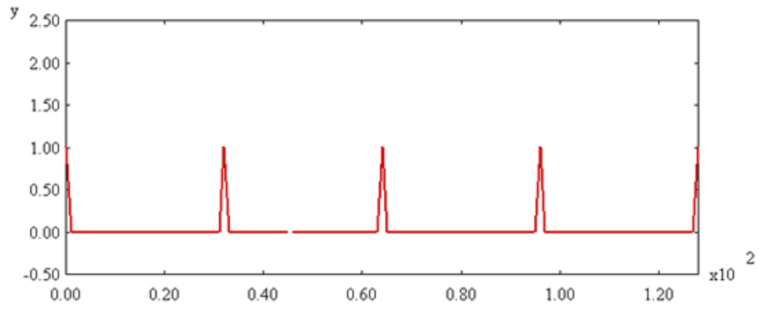
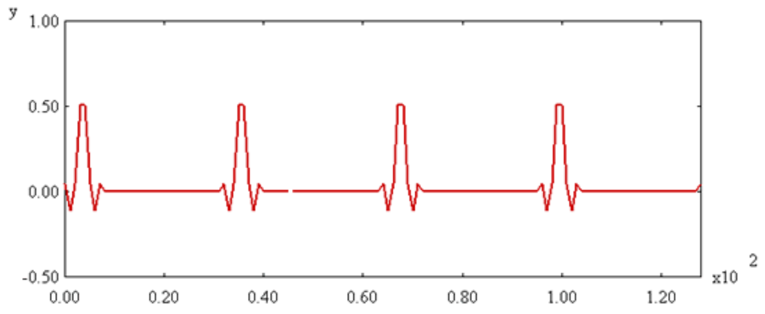


Figure 2: Finite Impulse Response (FIR)



DSP Cosimulation with Transient Circuit Simulation

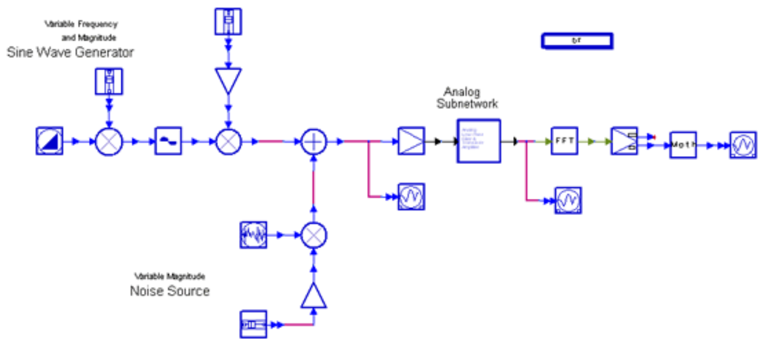
Location: $HPEESOF_DIR/examples/DSP/dsp_demos_wrk/SPICE_COSIM

Objective

This example shows Agilent Ptolemy's ability to cosimulate with Transient (SPICE) simulators.

Setup

In Agilent Ptolemy, a variable magnitude and frequency waveform is generated. A variable magnitude noise is then added and the resulting waveform is sent into the analog simulator. In the analog sub-network, the signal passes through a Butterworth Lowpass filter to eliminate the added noise and then the signal passes through a single transistor amplifier stage. The resulting waveform is then sent back to Agilent Ptolemy where a fast Fourier transform (FFT) is performed.



Analysis

Figure 1: Analog subcircuit for Transient cosimulation

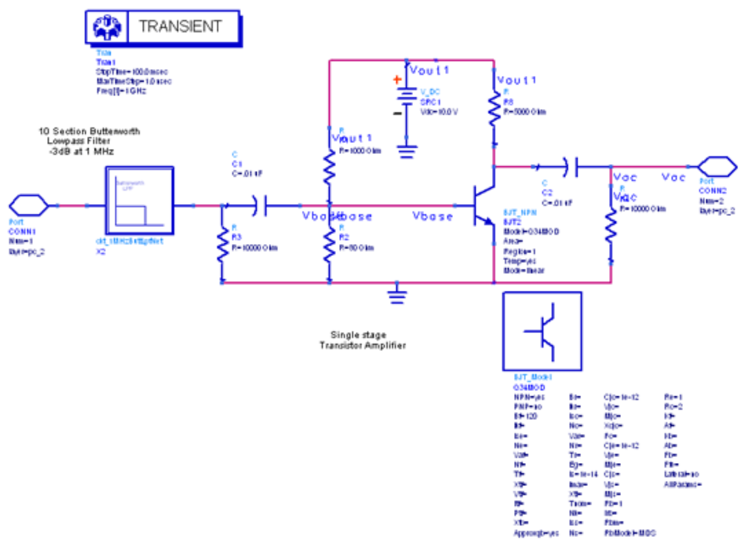

TRANSIENT
10 Section Butterworth Lowpass Filter -3dB at 1 MHz
Vout1
Vbase
Voc
Single stage Transistor Amplifier

Figure 2: Input Signal

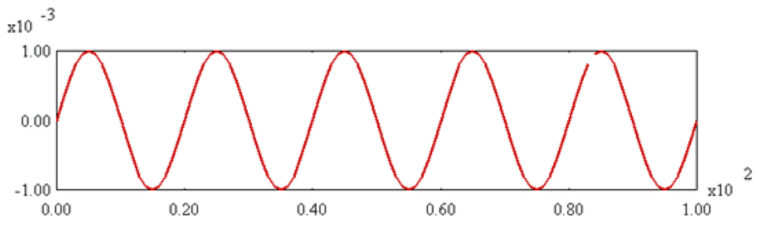


Figure 3: Output Signal

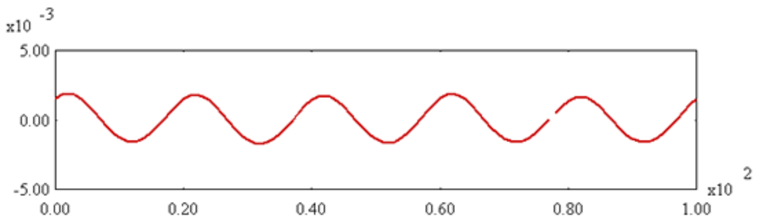
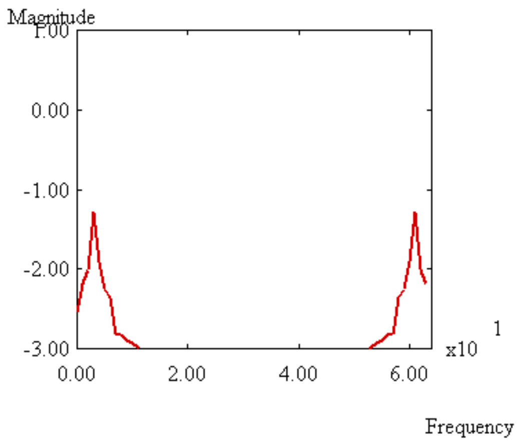


Figure 4: Output Spectrum



Equalized 16-QAM with Multipath and Phase Noise

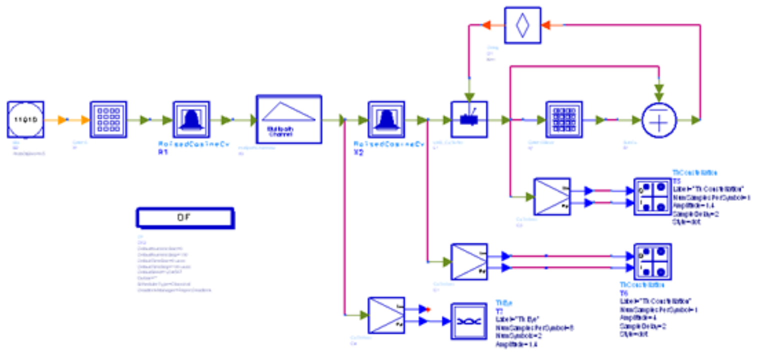
Location: $HPEESOF_DIR/examples/DSP/dsp_demos_wrk/EQ_16QAM

Objective

In this example, an equalized 16-QAM system is simulated. It features interactive slider bars for real-time analysis of multipath distortion effects.

Setup

Through the use of the Tcl/Tk run-time control components, the multipath distortion (Echo Amplitude vs. Line-of-Sight (LOS) Amplitude) can be adjusted along with the phase shift between transmitter and receiver. The received signal out of the channel is then fed into an equalizer. The tap cofficents of the equalizer are displayed in a bar graph and can be reset interactively. The output of the equalizer is displayed in a constellation diagram and an eye diagram.



Analysis

Figure 1: Constellation before equalization

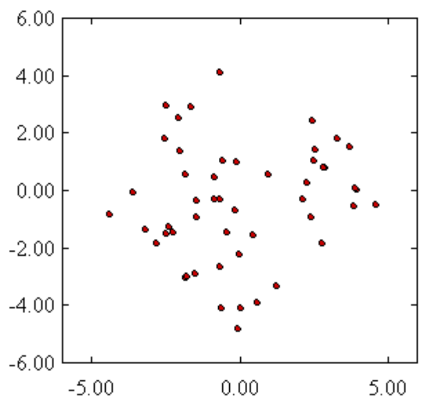


Figure 2: Constellation after equalization

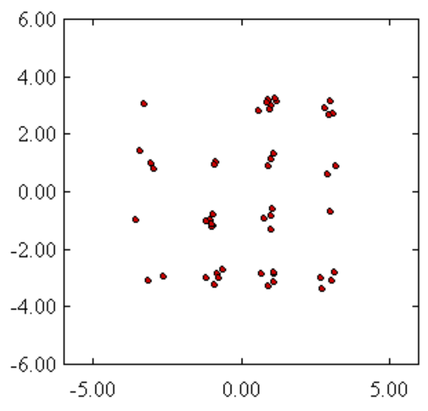


Figure 3: Eye diagram

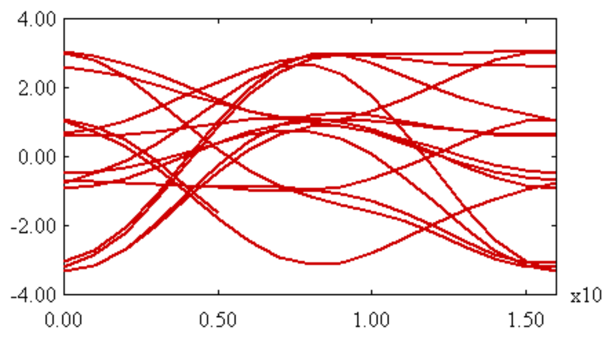
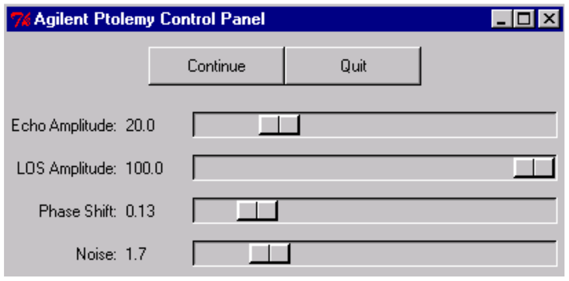


Figure 4: Slider bars to adjust Echo, LOS, phase shift and noise in real time



Echo Amplitude: 20.0

LOS Amplitude: 100.0

Phase Shift: 0.13

Noise: 1.7

Example of using A/D-D/A Models

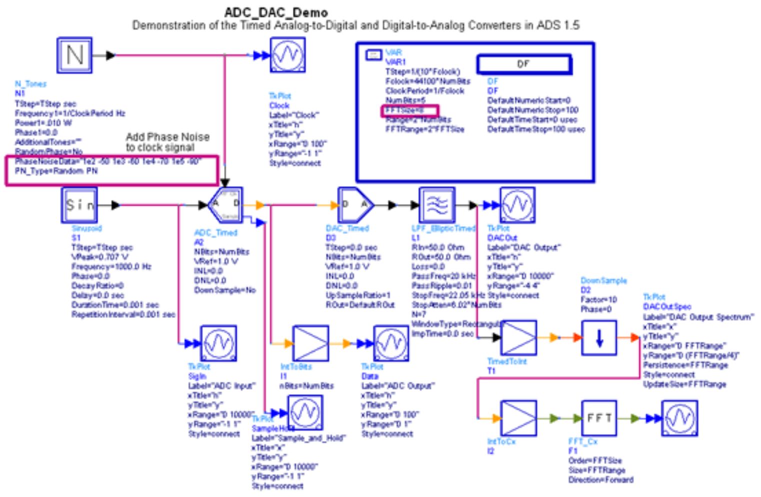
Location: $HPEESOF_DIR/examples/DSP/AtoDDtoA_wrk

Objective

This example demonstrates the Timed ADC and DAC models in Advanced Design System through a basic digital audio codec application. These models are suitable for the modeling of ideal or non-ideal conversion in signal processing applications.

Setup

The design "ADC_DAC_Demo" shows a basic digital audio codec application having a 44.1 kHz sampling rate. The bit resolution may be adjusted using the variable "Numbits" found in the VAR block adjacent to the DataFlow controller. Tk Plots are used to display the input, clock, sampled input, ADC output, filtered DAC output and a 256-point FFT of the DAC output. Phase noise defined by pairs of {frequency offset, dBc} may be introduced into the "N_Tones" clock source.


ADC_DAC_Demo
Demonstration of the Timed Analog-to-Digital and Digital-to-Analog Converters in ADS 1.5
Add Phase Noise to clock signal
N
Sin
A D
D A
DF
FFT

N_Tones
N1
TStep=TStep sec
Frequency1=1/ClockPeriod Hz
Power1=.010 W
Phase1=0.0
AdditionalTones=""
RandomPhase=No
PhaseNoiseData="1e2 -50 1e3 -60 1e4 -70 1e5 -90"
PN_Type=Random PN

TkPlot
Clock
Label="Clock"
xTitle="n"
yTitle="y"
xRange="0 100"
yRange="-1 1"
Style=connect

VAR
VAR1
TStep=1/(10*Fclock)
Fclock=44100*NumBits
ClockPeriod=1/Fclock
NumBits=5
FFTSize=8
Range=2^NumBits
FFTRange=2^FFTSize

DF
DF
DefaultNumericStart=0
DefaultNumericStop=100
DefaultTimeStart=0 usec
DefaultTimeStop=100 usec

Sinusoid
S1
TStep=TStep sec
VPeak=0.707 V
Frequency=1000.0 Hz
Phase=0.0
DecayRatio=0
Delay=0.0 sec
DurationTime=0.001 sec
RepetitionInterval=0.001 sec

ADC_Timed
A2
NBits=NumBits
VRef=1.0 V
INL=0.0
DNL=0.0
DownSample=No

DAC_Timed
D3
TStep=0.0 sec
NBits=NumBits
VRef=1.0 V
INL=0.0
DNL=0.0
UpSampleRatio=1
ROut=DefaultROut

LPF_EllipticTimed
L1
RIn=50.0 Ohm
ROut=50.0 Ohm
Loss=0.0
PassFreq=20 kHz
PassRipple=0.01
StopFreq=22.05 kHz
StopAtten=6.02*NumBits
N=7
WindowType=Rectangular
ImpTime=0.0 sec

TkPlot
DACOut
Label="DAC Output"
xTitle="n"
yTitle="y"
xRange="0 10000"
yRange="-4 4"
Style=connect

DownSample
D2
Factor=10
Phase=0

TkPlot
DACOutSpec
Label="DAC Output Spectrum"
xTitle="x"
yTitle="y"
xRange="0 FFTRange"
yRange="0 (FFTRange/4)"
Persistence=FFTRange
Style=connect
UpdateSize=FFTRange

TkPlot
SigIn
Label="ADC Input"
xTitle="n"
yTitle="y"
xRange="0 10000"
yRange="-1 1"
Style=connect

IntToBits
I1
nBits=NumBits

TkPlot
Data
Label="ADC Output"
xTitle="n"
yTitle="y"
xRange="0 100"
yRange="0 1"
Style=connect

TkPlot
SampleHold
Label="Sample_and_Hold"
xTitle="x"
yTitle="y"
xRange="0 10000"
yRange="-1 1"
Style=connect

TimedToInt
T1

IntToCx
I2

FFT_Cx
F1
Order=FFTSize
Size=FFTRange
Direction=Forward

Analysis

Figure 1: Output of Sample and Hold, Fin = 10 kHz, Numbits = 5, no Phase Noise

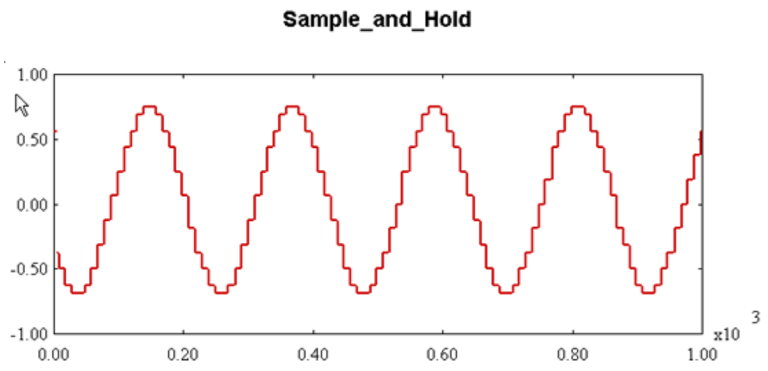


Figure 2: Output of Sample and Hold, Fin = 10 kHz, Numbits = 5, with Phase Noise

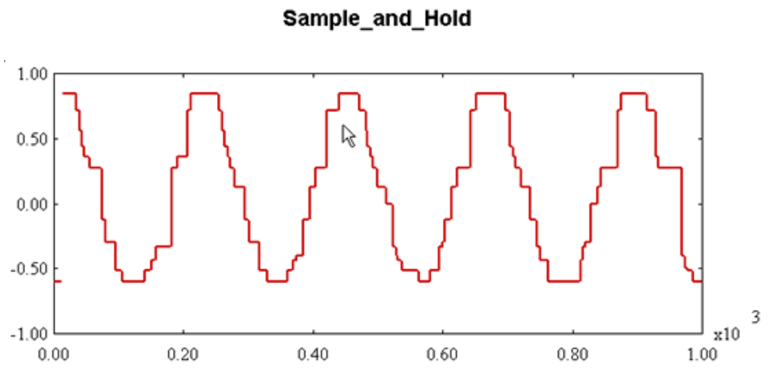
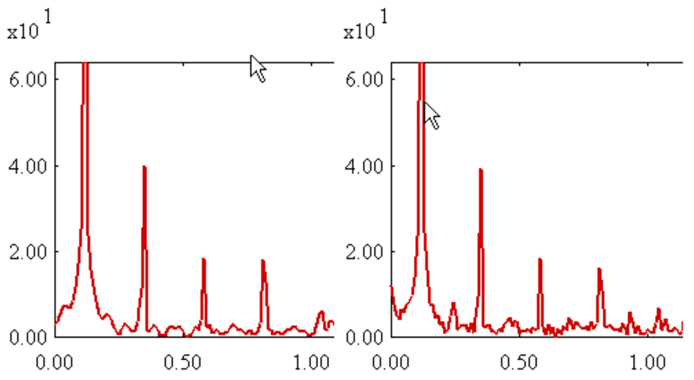


Figure 3: Output of DAC filter, Numbits = 5, without (left) and with (right) Phase Noise and 1 LSB differential nonlinearity



Notes

- ADC_Timed has an external clock input that uses a "clock" signal from an RF source which may include phase noise.
- Both ADC_TImed and DAC_Timed have the parameters INL and DNL for integral and differential non-linearity with respect to the Least Significant Bit (LSB).

Eye Diagram with Variable Noise Generator

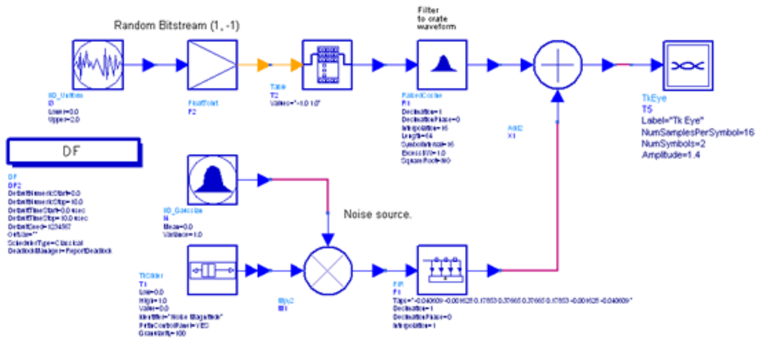
Location: $HPEESOF_DIR/examples/DSP/dsp_demos_wrk/EYE

Objective

This example demonstrates basic DSP simulation capability with an interactive user interface. It features the use of Tcl/Tk interactive display and control, adjustable noise voltage during simulation, Raised-Cosine filter and Finite Impulse Response (FIR) noise filter.

Setup

A random bitstream is raised-cosine filtered and summed with an adjustable band-limited Gaussian noise signal. The resulting antipodal signal is displayed on an eye diagram by means of Tcl/Tk. An interactive slider allows for adjustment of the noise voltage with real-time update to the eye display.



Analysis

Figure 1: Eye diagram for a binary antipodal signal without noise added

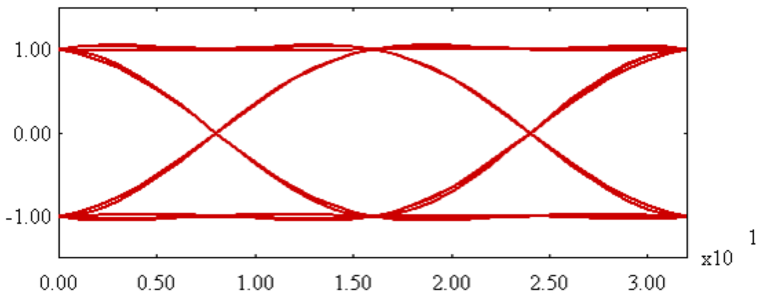


Figure 2: Eye diagram after noise added

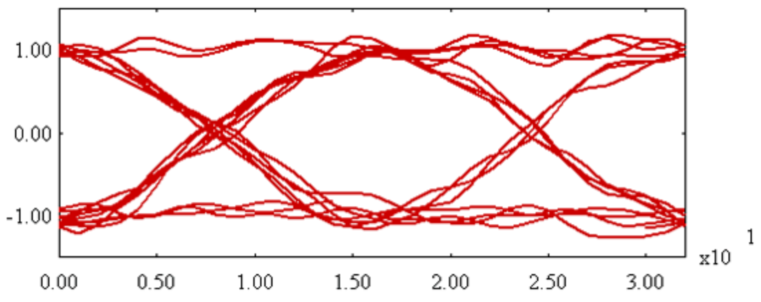
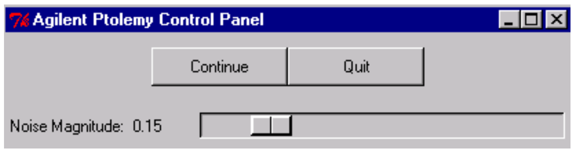


Figure 3: Slide bar to control noise magnitude



Noise Magnitude: 0.15

MATLAB and Agilent Ptolemy Co-simulation

Location: $HPEESOF_DIR/examples/DSP/MATLABlink_wrk

Objective

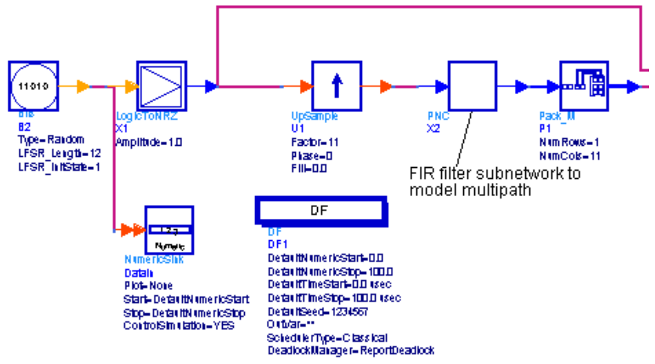
Demonstrate the link between MATLAB and Agilent Ptolemy

Setup

This workspace contains a simple example, called Channel_Estimate, to demonstrate the link between MATLAB and Agilent Ptolemy. A random bitstream is fed through a basic multipath channel modeled using an FIR filter (subnetwork PNC). This output is then fed into the MATLAB link which calls a basic LMS (Least Mean Squares) equalizer routine (see examples/DSP/MATLABlink_wrk/data/hplms.m). The recovered signal along with the error signal are displayed by MATLAB and also saved to datasets.

> **Note**
> The MATLAB figures may plot at the same screen location or be minimized. Please restore if needed and move them so both are visible.

left side of the schematic



Bits
B2
Type=Random
LFSR_Length=12
LFSR_InitState=1

LogicToNRZ
X1
Amplitude=1.0

UpSample
U1
Factor=11
Phase=0
Fill=0.0

PNC
X2

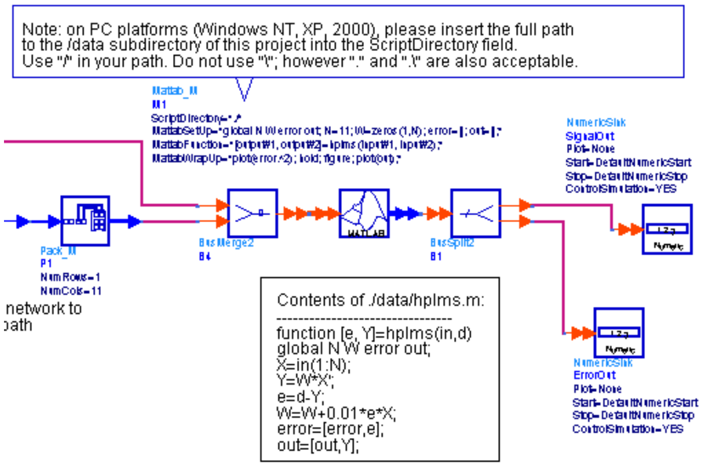
Pack_M
P1
NumRows=1
NumCols=11

FIR filter subnetwork to model multipath

NumericSink
DataIn
Plot=None
Start=DefaultNumericStart
Stop=DefaultNumericStop
ControlSimulation=YES

DF
DF1
DefaultNumericStart=0.0
DefaultNumericStop=100.0
DefaultTimeStart=0.0 usec
DefaultTimeStop=100.0 usec
DefaultSeed=1234567
OutVar=""
SchedulerType=Classical
DeadlockManager=ReportDeadlock

right side of the schematic



Note: on PC platforms (Windows NT, XP, 2000), please insert the full path to the /data subdirectory of this project into the ScriptDirectory field. Use "/" in your path. Do not use "\"; however "." and ".\" are also acceptable.

Matlab_M
M1
ScriptDirectory="./"
MatlabSetUp="global N W error out; N=11;W=zeros(1,N);error=[];out=[];"
MatlabFunction="[output#1,output#2]=hplms(input#1, input#2);"
MatlabWrapUp="plot(error.^2); hold; figure; plot(out);"

NumericSink
SignalOut
Plot=None
Start=DefaultNumericStart
Stop=DefaultNumericStop
ControlSimulation=YES

Pack_M
P1
NumRows=1
NumCols=11

network to
path

BusMerge2
B4

BusSplit2
B1

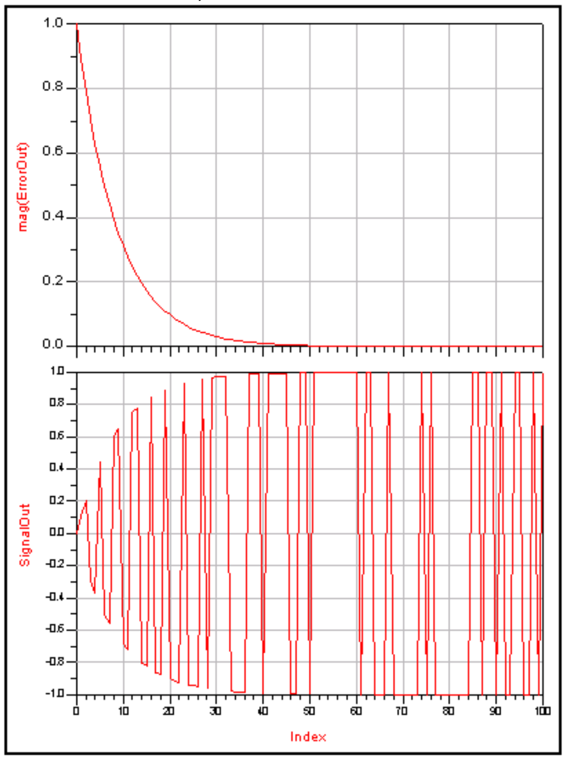
```
Contents of ./data/hplms.m:
--------------------------------
function [e, Y]=hplms(in,d)
global N W error out;
X=in(1:N);
Y=W*X';
e=d-Y;
W=W+0.01*e*X;
error=[error,e];
out=[out,Y];
```

NumericSink
ErrorOut
Plot=None
Start=DefaultNumericStart
Stop=DefaultNumericStop
ControlSimulation=YES

Co-simulation setup for modeling multipath channel errors and correcting them

Analysis

Simulation results, from MATLAB



Notes

- This example requires that MATLAB 6.5 or greater be installed and available. Please see the ADS Ptolemy Simulation documentation, Introduction to MATLAB, "Setting up MATLAB" for further instructions. In particular, the MATLAB variable in the hpads.cfg file found in your ADS installation directory ($HPEESOF_DIR\..., e.g. C:\ADS2005A\config\hpads.cfg) should be set to the path leading to where the MATLAB /bin and /extern directories are located, e.g. C:\MATLAB6p5\bin .
- This workspace also contains an identical example, called Channel_Estimate2 which uses the MATLAB LIBLink component. It requires the MATLAB Compiler to be installed which will compile .m script files into binary format for speed improvement.
- Also note that on the PC, the full path to the /data subdirectory of this workspace must be inserted into the ScriptDirectory field of the Matlab_M component, using forward slashes "/".

Matlab Cosimulation

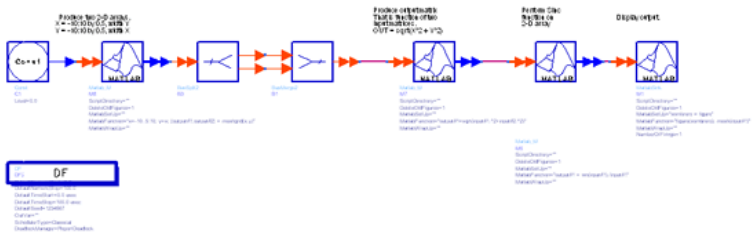
Location: $HPEESOF_DIR/examples/DSP/dsp_demos_wrk/SOMBRERO

Objective

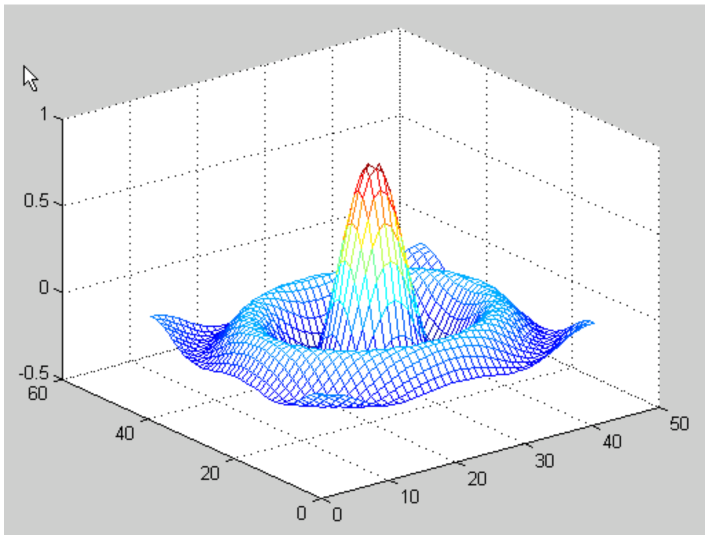
This example demonstrates the basic Matlab cosimulation using Matlab graphic display.

Setup

The data is generated in Matlab and passed into Agilent Ptolemy where it is passed between separate Matlab components for processing into a SINC function. The output is then displayed in Matlab.



Analysis



Notes

- A "divide by zero" warning message from Matlab is normal for this example.

PLL Demo 1 in DSP

Location: $HPEESOF_DIR/examples/DSP/dsp_demos_wrk/PLL1

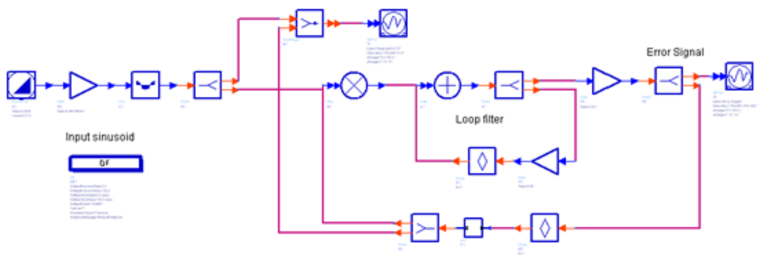
Objective

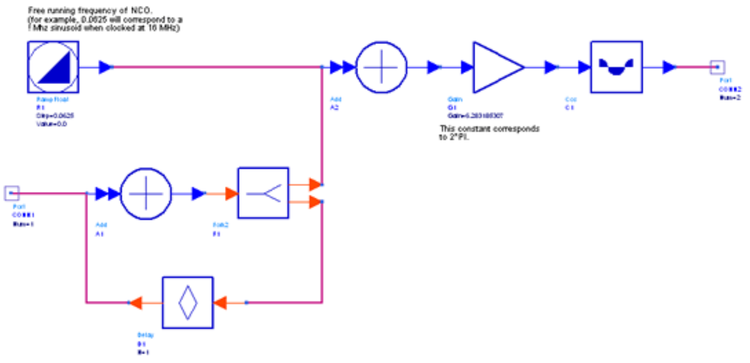
This example is used to demo a basic simulation of a phase-locked loop (PLL) using Agilent Ptolemy.

Setup

In this example the frequency of the input sinusoid matches the free running frequency of the numerically-controlled oscillator (NCO), but the phase is offset. Hence the PLL generates an initial error signal to shift the phase of the NCO. After the phase locks, the error signal decreases to zero.

Figure 1: Numerically controlled oscillator subcircuit




Free running frequency of NCO.
(for example, 0.0625 will correspond to a 1 Mhz sinusoid when clocked at 16 MHz)
RampFloat
R1
Step=0.0625
Value=0.0
Add
A2
Gain
G1
Gain=6.283185307
This constant corresponds to 2*PI.
Cos
C1
Port
COSN2
Num=2
Port
COSN1
Num=1
Add
A1
Fork2
F1
Delay
D1
N=1

Analysis

Figure 2: Input and NCO output

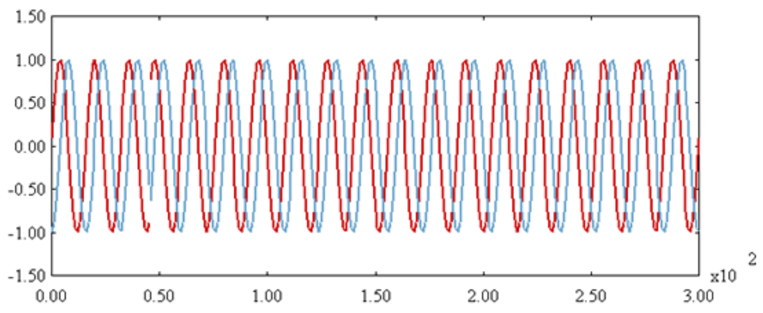
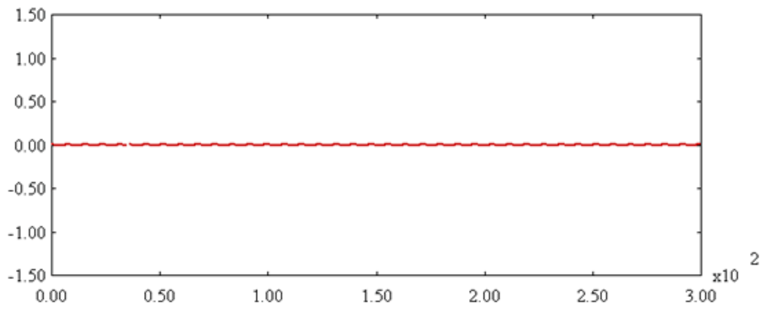


Figure 3: Corresponding error signal



Notes

- For more detailed PLL examples please see: examples/RF_Board/PLL_Examples.
- For help with the design of PLLs, the PLL DesignGuide is available.

PLL Demo 2 in DSP

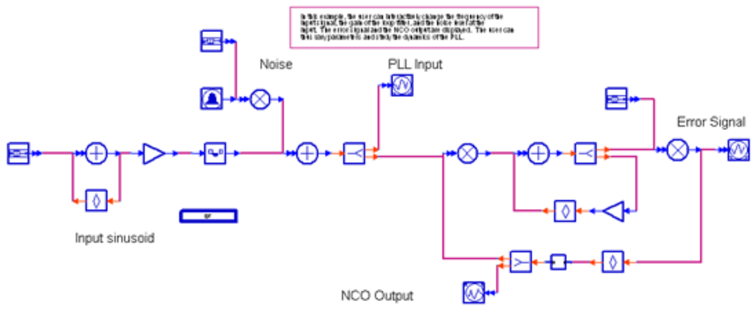
Location: $HPEESOF_DIR/examples/DSP/dsp_demos_wrk/PLL2

Objective

This example is used to demonstrate a basic simulation of a phase-locked loop (PLL) while allowing the user to interactively change parameters to study the dynamics of the PLL.

Setup

In the example, the user can interactively change the frequency of the input signal, the gain of the loop filter, and the noise level at the input. The error signal and the numerically-controlled oscillator (NCO) output are displayed.



Analysis

Figure 1: Input signal

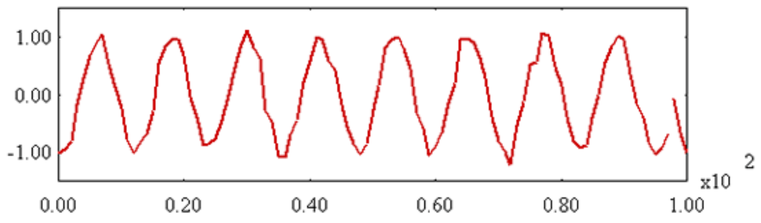


Figure 2: NCO signal

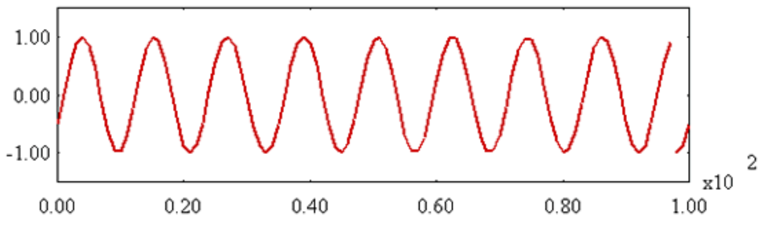


Figure 3: Corresponding error signal

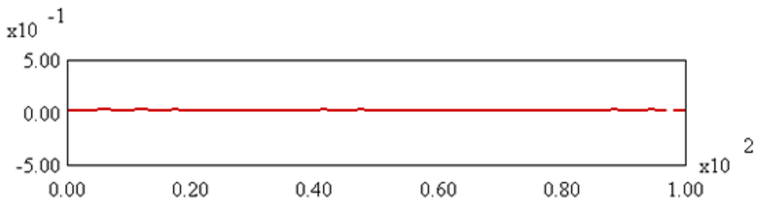
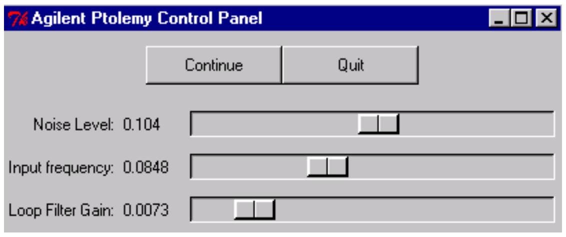


Figure 4: Slider bars for real-time system-level adjustments



Noise Level: 0.104
Input frequency: 0.0848
Loop Filter Gain: 0.0073

Notes

- For more detailed PLL examples please see: examples/RF_Board/PLL_Examples.
- For help with the design of PLLs, the PLL DesignGuide is available.

Sine and Cosine Wave Generator

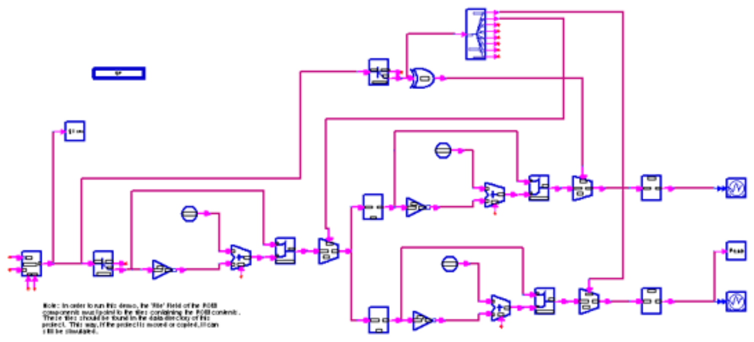
Location: $HPEESOF_DIR/examples/DSP/dsp_demos_wrk/SINE

Objective

This example illustrates a Sine and Cosine wave generator implemented in synthesizable.

Setup

A Sine and Cosine wave generator was created using synthesizable components. The first quadrant of each waveform is stored in a Synthesizable Read Only Memory (ROM) component. The other three quadrants are derived from this ROM by manipulating the input value to the ROM and/or by multiplying the output value by -1.



Analysis

Figure 1: Generated Sine wave

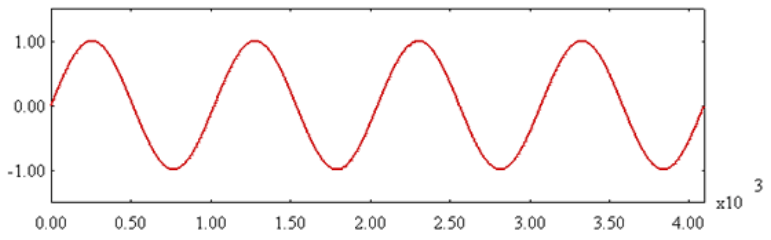
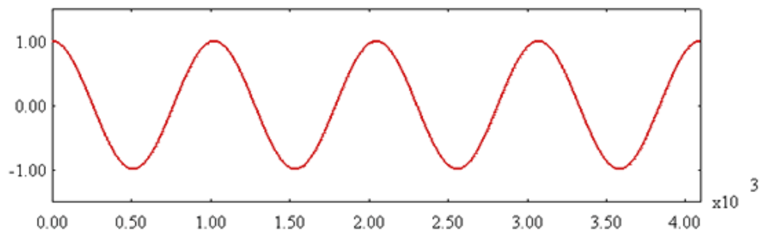


Figure 2: Generated Cosine wave



Notes

- In order to run this demo, the 'File' Field of the ROM components must point to the files containing the ROM contents. These files should be found in the data directory of this workspace. This way, if the workspace is moved or copied, it can still be simulated.

Timed QAM Modem

Location: $HPEESOF_DIR/examples/DSP/ModemTimed_wrk

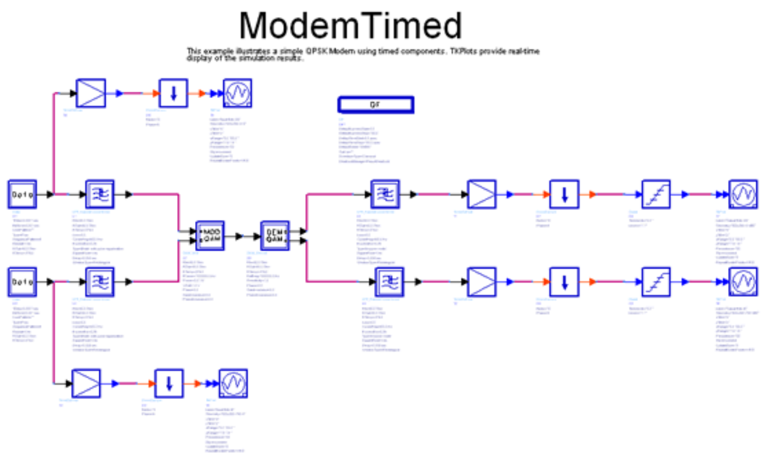
Objective

This example illustrates a Quadrature Amplitude Modulation (QAM) modulator and demodulator simulated using timed components. It features the use of Tcl/Tk plots for run-time output display and uses basic components such as raised-cosine filters, QAM modulator and demodulator, etc.

Setup

1. The I-Q data inputs and outputs are displayed at run-time by means of Tcl/Tk plots.

2. Downsampling is used after the conversion from Timed to Floating-point to reduce the number of output samples for more efficient display.


ModemTimed
This example illustrates a simple QPSK Modem using timed components. TKPlots provide real-time display of the simulation results.

Analysis

Figure 1: Portion of input I data

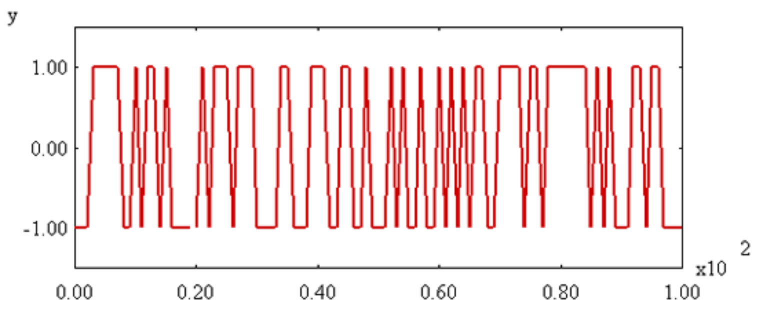
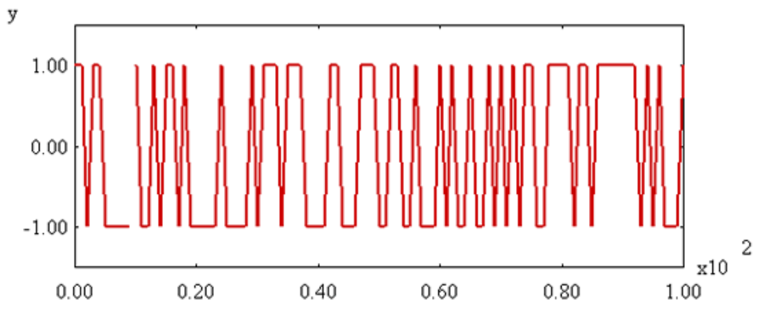


Figure 2: Portion of output I data



FEM Simulator Examples

FEM Simulator was developed with the designer of high-frequency/high-speed circuits in mind, FEM Simulator offers a powerful finite-element EM simulator that solves a wide array of applications with impressive accuracy and speed. These examples illustrate some of the capabilities found in FEM Simulator enabling you to to achieve optimal structures that meet circuit and device performance goals.

- *Directional Coupler* (examples)
- *Low Pass Filter* (examples)
- *LTCC Balun* (examples)
- *Panel Antenna With Radome* (examples)
- *QFN Package* (examples)

Directional Coupler

Location: $HPEESOF_DIR/examples/FEM/DirectionalCoupler20dB_wrk

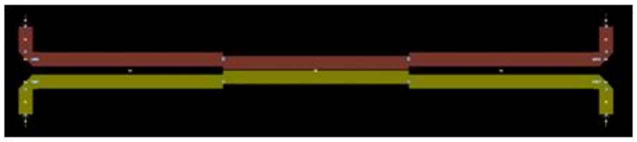
Objective

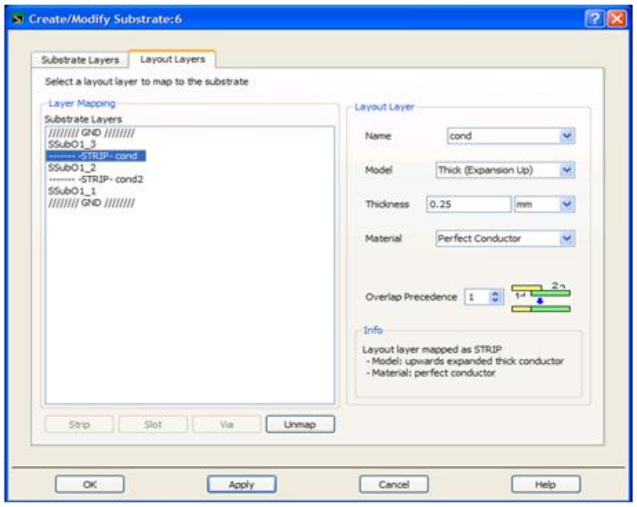
This example shows the application of Dielectric Via FEM.

Setup

- **ideal_cplr**: The design begins with selection of even Ze and odd Zo impedances for -20 dB and coupling with 50 Ohm input. The reference paper is mentioned in the design. The values of impedances obtained from tables is implemented with Ideal coupled line CLIN and schematic simulation is carried out to check whether the choice of Ze and Zo is giving the required performance.
- **cplr_strp**: The Line Calculator is used to find the width of the line W and gap Wo for Offset Stripline SOCLIN based on calculated Ze and Zo in previous step. The schematic circuit is developed and circuit simulation is carried out to check required coupling of -20dB.
- **cplr_strp_1**: From Momentum simulation, it is found that coupling is -19.78 db instead of the required -20dB. Therefore, the circuit is actually over coupling than the required level. The optimization for gap Wo (which is responsible for coupling) is performed at the circuit level. After the optimization, the value of Wo is found to be more than used in cplr_strp. The momentum simulation shows coupling of -20 dB now.
- **cplr_strp_1_input**: The input section is added to the circuit to make it compatible with 50 Ohm input connector. Again circuit simulation, Momentum Simulation and Optimization is carried out at this level.
- **cplr_strp_1_input_spacers**: The Teflon Spacers are added to the circuit. FEM simulation is carried out. The circuit cplr_strp_1_input is used and teflon spacers are added. The coupling is found to be increased because of the presence of Teflon Spacers(see cplr_strp_1_input_spacers_emds.dds). The optimization is again carried out for Wo to bring back the coupling at -20 dB level.
- **Comparison_emds_mom.dds**: Comparison of FEM and Momentum result when the gap Wo is not optimized in the presence of Teflon spacer.
- **Comparison_emds_opt_mom.dds**: Comparison of FEM and Momentum result when gap Wo is optimized in the presence of Teflon.

Figure 1: Directional Coupler and Substrate Stackup





Create/Modify Substrate:6

Substrate Layers | Layout Layers

Select a layout layer to map to the substrate

Layer Mapping

Substrate Layers

```
//////// GND ////////
SSubO1_3
-------- -STRIP- cond
SSubO1_2
-------- -STRIP- cond2
SSubO1_1
//////// GND ////////
```

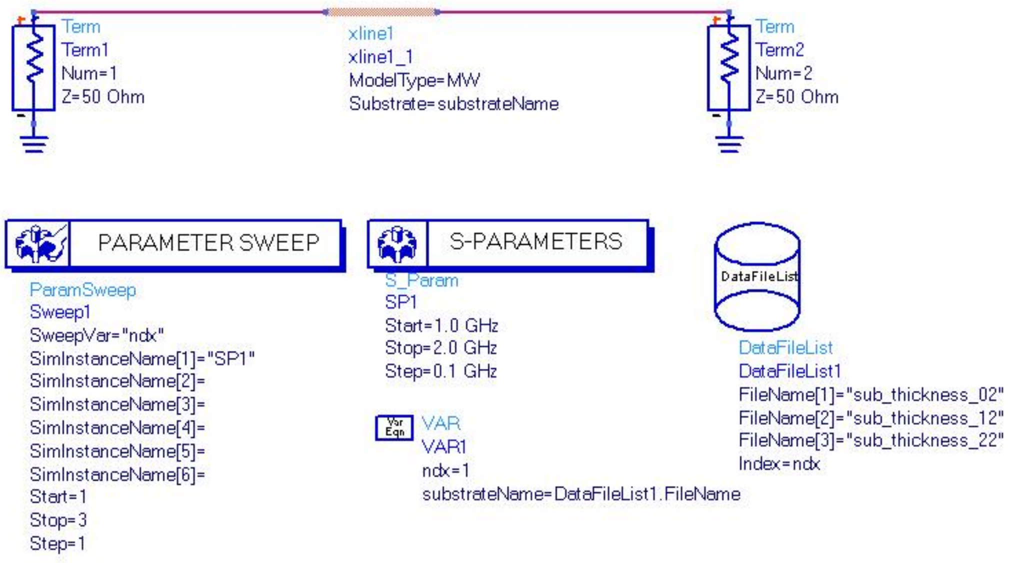
Layout Layer

Name	cond
Model	Thick (Expansion Up)
Thickness	0.25 mm
Material	Perfect Conductor
Overlap Precedence	1

Info

Layout layer mapped as STRIP
- Model: upwards expanded thick conductor
- Material: perfect conductor

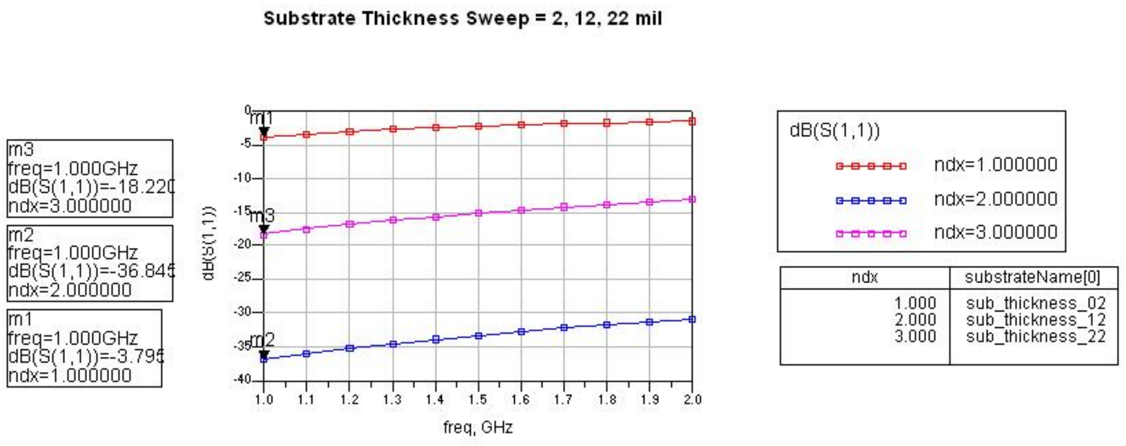
Figure 2: Directional Coupler with Teflon Spacers and Associated Substrate Stackup



```
Term
Term1
Num=1
Z=50 Ohm

xline1
xline1_1
ModelType=MW
Substrate=substrateName

Term
Term2
Num=2
Z=50 Ohm

PARAMETER SWEEP
ParamSweep
Sweep1
SweepVar="ndx"
SimInstanceName[1]="SP1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1
Stop=3
Step=1

S-PARAMETERS
S_Param
SP1
Start=1.0 GHz
Stop=2.0 GHz
Step=0.1 GHz

VAR
VAR1
ndx=1
substrateName=DataFileList1.FileName

DataFileList
DataFileList
DataFileList1
FileName[1]="sub_thickness_02"
FileName[2]="sub_thickness_12"
FileName[3]="sub_thickness_22"
Index=ndx
```

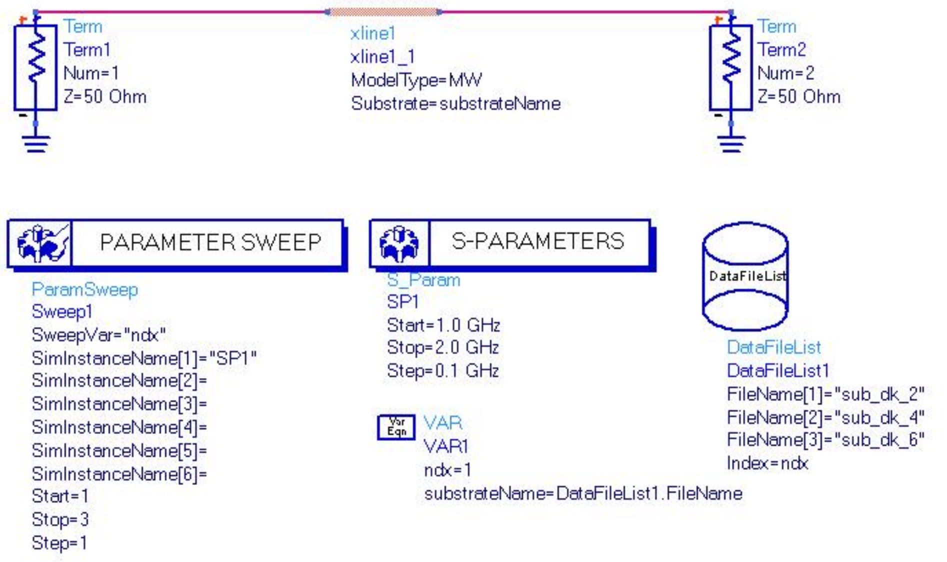
Analysis

Figure 3: FEM simulation Results with Teflon Spacers before Optimization



ndx	substrateName[0]
1.000	sub_thickness_02
2.000	sub_thickness_12
3.000	sub_thickness_22

Figure 4: FEM simulation Results with Teflon Spacers after Optimization and Comparison with Momentum Result



Low Pass Filter

Location: $HPEESOF_DIR/examples/FEM/LOWPASSFILTER_wrk

Objective

This example shows the application of *Symmetric Plane* of FEM Simulator

Setup

1. "LOWPASSFILTER" shows both Momentum and FEM Simulation. Results from both match closely with Measured result.
2. "LOWPASSFILTER_SYMMX" shows the Symmetric plane feature of FEM implemented in original circuit. The simulation time and memory consumption for FEM is reduced because of the reduced size of the structure, but results are same as of original circuit.
3. "lowpassfilter.dds" compares the simulation results from Momentum and FEM (with and without Symmetric plane) with the measured result.

Figure 1: Low Pass Filter

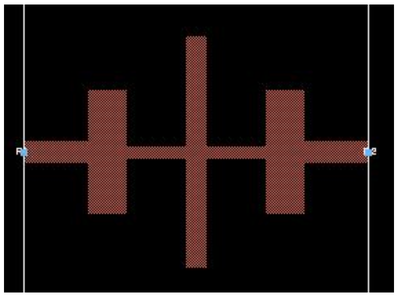
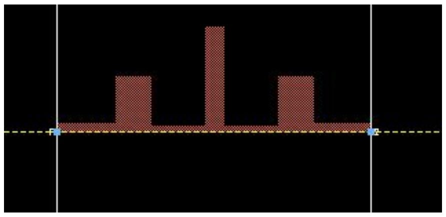


Figure 2: Low PassFilter With Symmetric plane



Analysis

Figure 3: dB(S2,1)

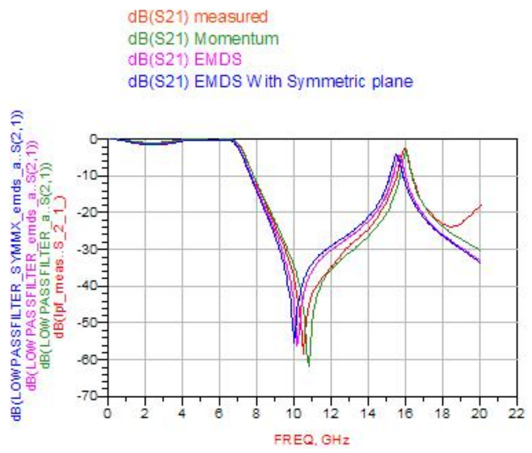
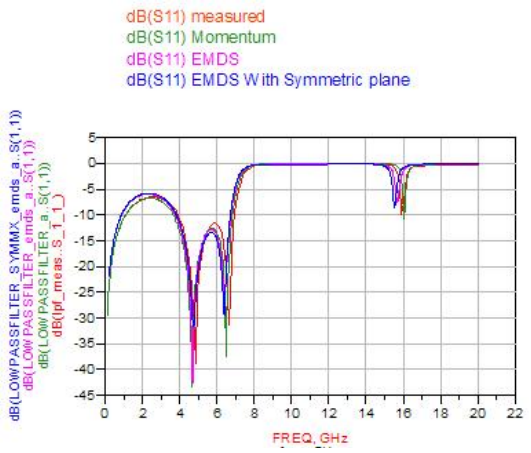


Figure 4: Return Loss



Notes

The Symmetric plane facility is provided in both the X and Y axis. In this example the X axis Symmetric plane is implemented.

LTCC Balun

Location: $HPEESOF_DIR/examples/FEM/LTCC_BALUN

Objective

This example explains the use of dielectic via in FEM which is the exact scenario of LTCC balun design. This gives an edge to FEM over Momentum.

Setup

- **LTCC_balun_31_07_08_1**: It is the layout design of LTCC Balun use for momentum simulation.
- **LTCC_balun_22_10_08__emds_3**: It is the layout design of LTCC balun for FEM simulation using dielectric via property.

Figure 1: Layout Design used for Momentum Simulation

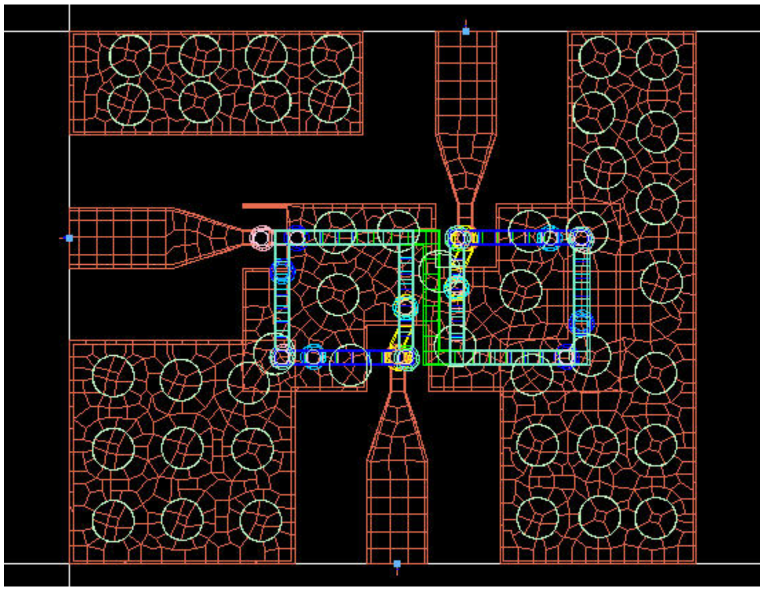
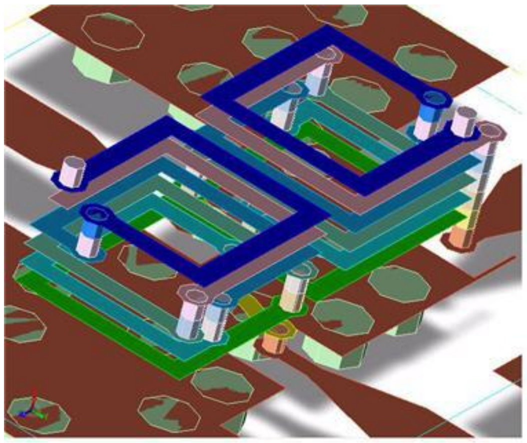


Figure 2: 3D Preview for the FEM Simulation



Analysis

Figure 3: Momentum Simulation Result for LTCC Merchand Balun

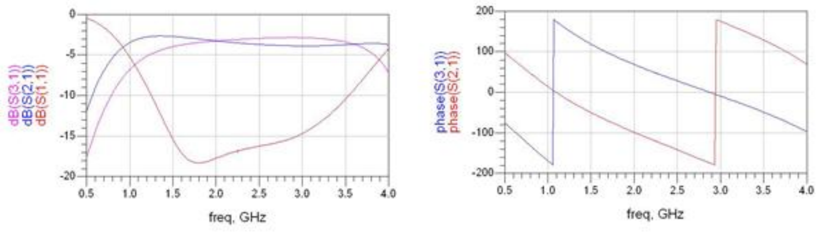
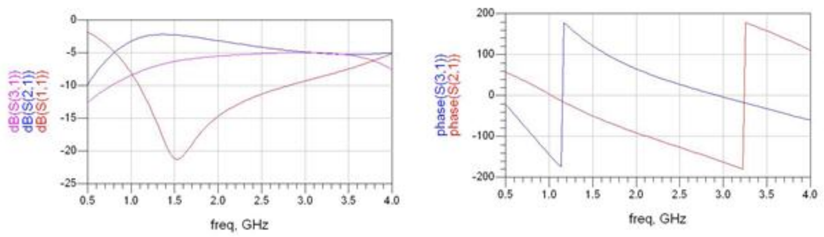


Figure 4: FEM Simulation Result for LTCC Merchand Balun



Panel Antenna With Radome

Location: $HPEESOF_DIR/examples/FEM/Antenna_with_radome_wrk

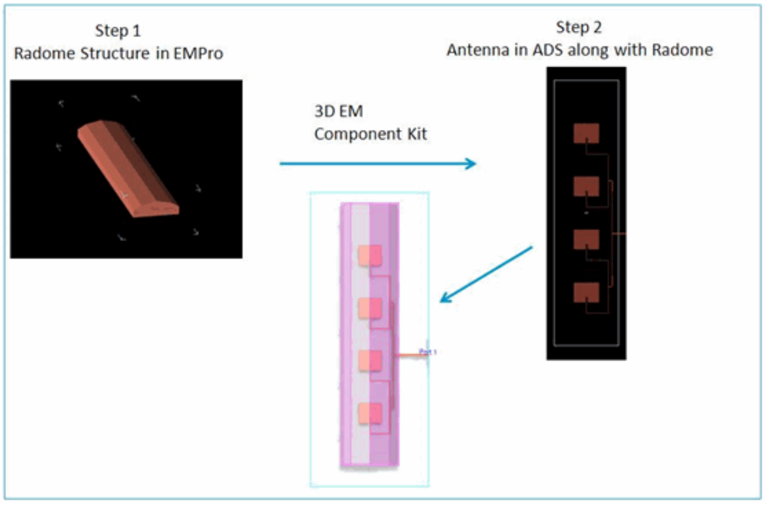
Objective

This example illustrates the application of EMPro-ADS link through 3D EM Component Design Kit generated in EMPro. The radome structure was designed using EMPro. The material used in Radome structure is ABS Plastic. The 3D EM Component Design kit for Radome is generated in EMPro and installed in ADS. The radome structure is placed over a Microstrip Panel antenna. The complete structure is subjected to FEM simulation to analyze the affect of Radome on Antenna performance.

Setup

Design Flow is shown in Figure 1. The radome structure is designed in EMPro and brought as a Design Kit into ADS. The analysis of complete structure is carried out in ADS using FEM simulator.

Figure 1: Design Flow



Analysis

Figure 2 shows S parameter of Panel antenna with and without Radome.

Figure 2: S Parameter of Panel Antenna with and without Radome

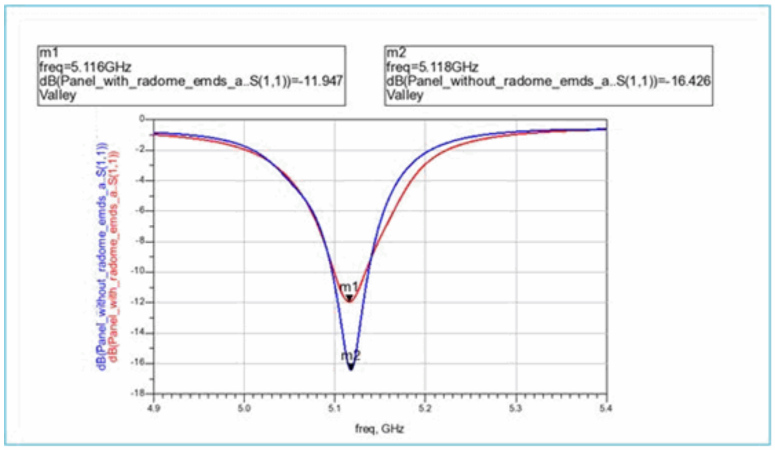
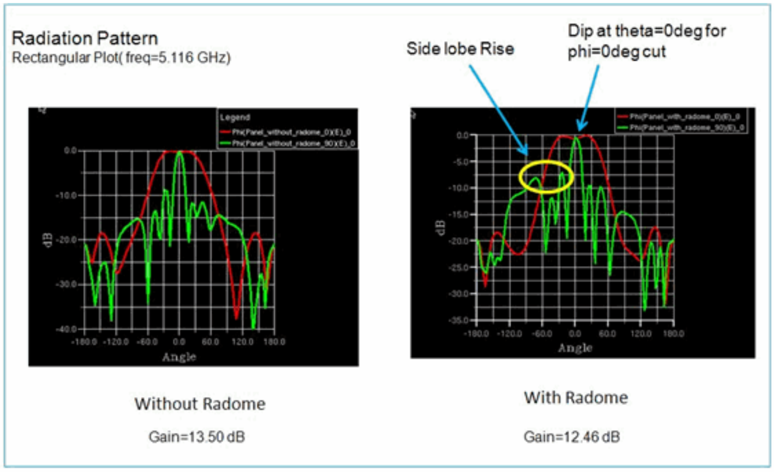


Figure 3 shows 2D Radiation Pattern in Phi=0 deg plane with and without radome.

Figure 3: Radiation Pattern of Panel Antenna with and without Radome



Notes

Install Design Kit **"EMProRadome_DesignKit.zip"** placed inside the workspace directory to carry out antenna analysis with radome.

QFN Package

Location: $HPEESOF_DIR/examples/FEM/QFN_Designer_wrk

Objective

This example illustrates FEM simulation capabilities for QFN package that contains an IC, and includes traces on the PC board the package is mounted to. The example shows a comparison between:

1. FEM simulation of the package and IC (really just a through line) together and
2. FEM simulation of the package and IC (with just bondpads) combined with a Momentum simulation of the IC (the same through line).

Setup

- **Chip_Thru_Line**: It is the layout design of a simple microstrip line on package dia.
- **example_board_3x3_QFN_chip_Bondw**: It is the layout design of QFN package with 3 pins on each side of package. A microstrip line placed on dia is connected with QFN packages pins with two JEDEC bondwire. This shows just a single bondwire from the input to the chip, a simple rectangular trace on the chip, and a single bondwire at the output of the chip.
- **example_board_3x3_QFN_open_chip_Bondw**: It is the layout design of QFN package with 3 pins on each side of package with microstrip line on dia is connected.
- **Re_using_EM_Models**: It is the schematic design of QFN package for em co-

simulation.

Figure 1: Layout Design used for FEM Simulation

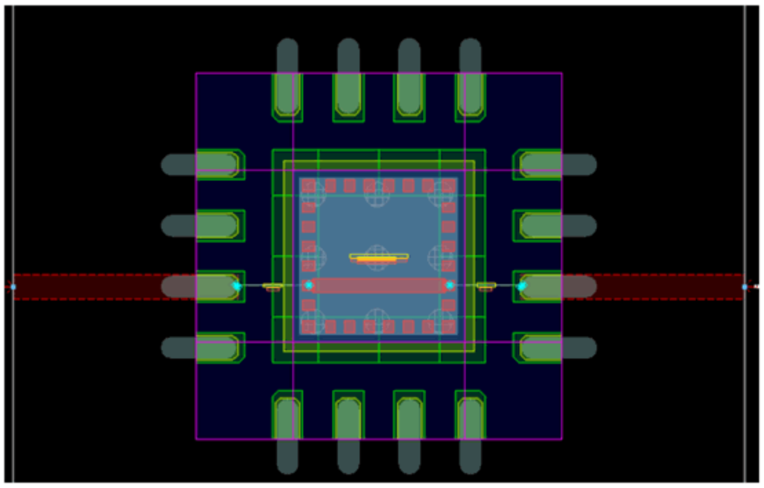
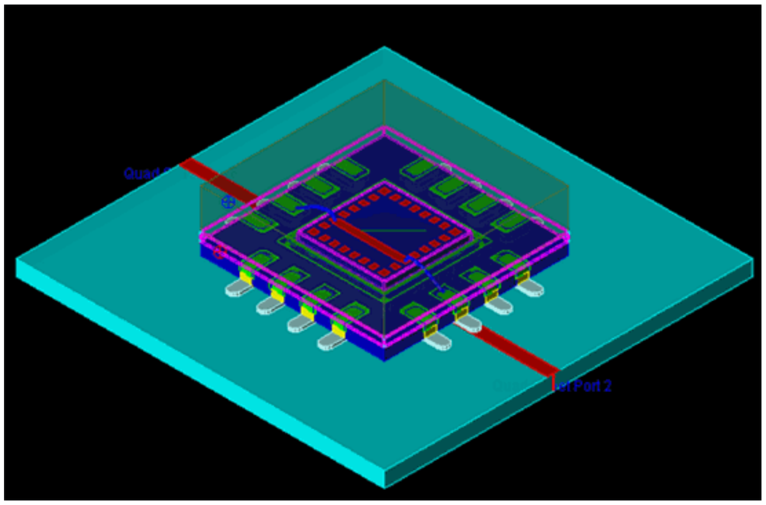
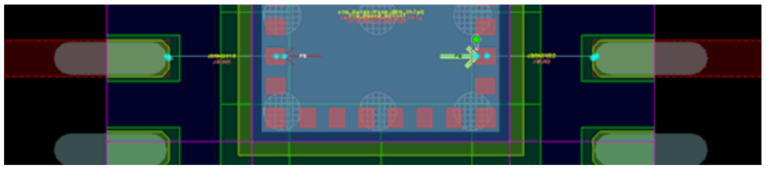


Figure 2: 3D Preview for the FEM Simulation



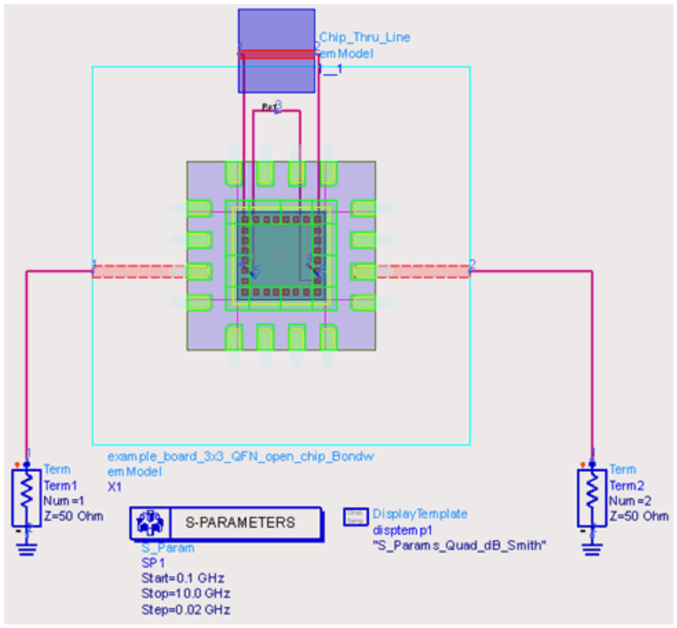
Ideally, we would like to be able to have whatever patterns and designs on the IC that we want and be able to simulate everything together. This works fine, provided the patterns on the IC are simple and don't have any non-linear devices. However, for complex ICs, we are assuming the chip will have to be simulated separately from the package. To see how to do this in a simple case, we simulate the above with just bond pads on the IC and with ports at the bond pads on the IC. Separately we will simulate just the through line by itself using Momentum. Then we will combine the results together and see if we get the same response, as when running FEM on everything including a through line on the chip.

Figure 3: Layout preview of "example_board_3x3_QFN_open_chip_Bondw"



From the EM Setup view, Model/Symbol section, we created a Layout look-alike symbol for em co-simulations. The EM Setup view has two ports defined, and we specify that the Layout look-alike symbol have a reference pin added so it can be connected correctly to the Layout look-alike symbol from the QFN package simulation where the "IC" just has bond pads. This shows an S-parameter simulation setup where we are re-using the FEM simulation results.

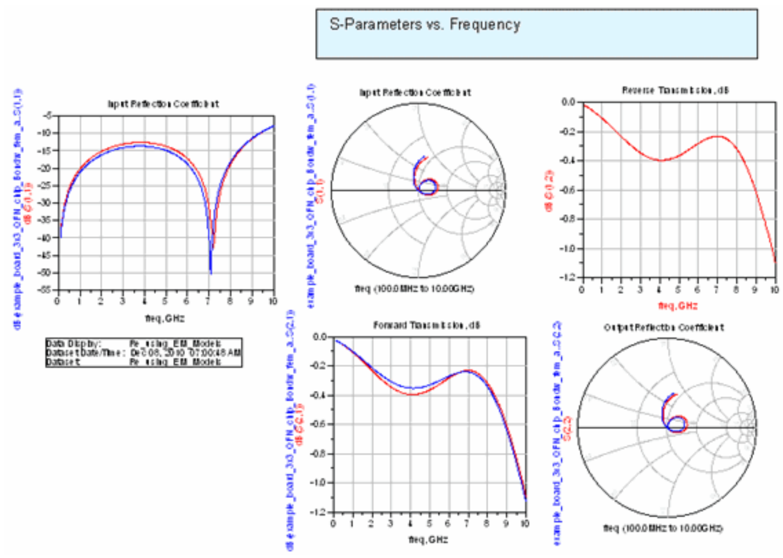
Figure 4: QFN Package setup in schematic for EM co-simulation



Analysis

These plots show good agreement between the original FEM simulation (in blue) of the whole package with IC with a through line and the re-use of the FEM and Momentum results (in red).

Figure 5: Simulation Result



Instrument Examples

Example of how to use an instrument component to acquire data.

- *CM_Infiniium_548xx_Source2 to acquire digitized data from an oscilloscope* (examples)

CM_Infiniium_548xx_Source2 to acquire digitized data from an oscilloscope

Location: $HPEESOF_DIR/examples/Instruments/CMInfiniium_wrk

Objective

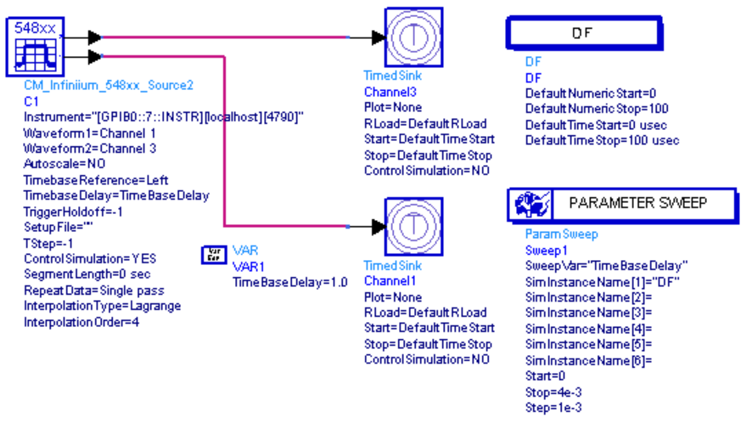
This example illustrates the use of the CM_Infiniium_548xx_Source2 component to acquire two channels of time-synchronized, digitized data. Acquiring data in this way would be useful when trying to process down-converted data representing the I and Q channels of a modulated signal.

This example also includes a sweep, illustrating the capability to position the acquisition time within the signal.

Setup

This example assumes that there is an instance of the Connection Manager Server running on the same PC that will run the simulation. If this is not the case, change the "Instrument" parameter of the C1 component to point at the host that runs the server.

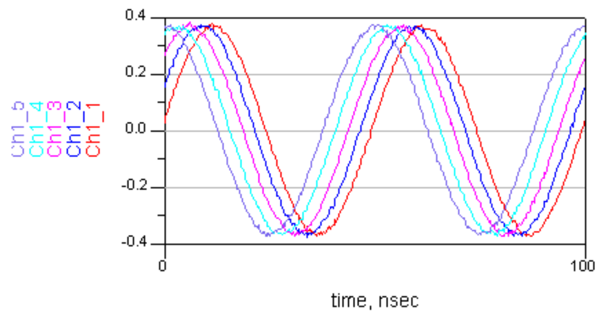
Figure: Schematic setup for Infiniium scope

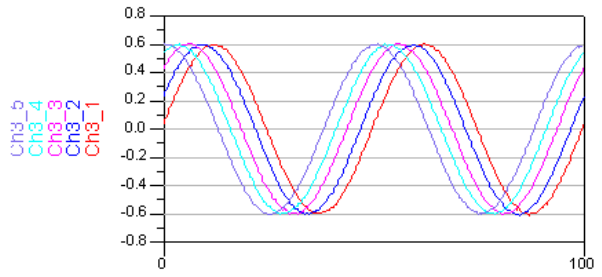


```
CM_Infiniium_548xx_Source2
C1
Instrument="[GPIB0::7::INSTR][localhost][4790]"
Waveform1=Channel 1
Waveform2=Channel 3
Autoscale=NO
TimebaseReference=Left
TimebaseDelay=TimeBaseDelay
TriggerHoldoff=-1
SetupFile=""
TStep=-1
ControlSimulation=YES
SegmentLength=0 sec
RepeatData=Single pass
InterpolationType=Lagrange
InterpolationOrder=4

VAR
VAR1
TimeBaseDelay=1.0

TimedSink
Channel3
Plot=None
RLoad=DefaultRLoad
Start=DefaultTimeStart
Stop=DefaultTimeStop
ControlSimulation=NO

TimedSink
Channel1
Plot=None
RLoad=DefaultRLoad
Start=DefaultTimeStart
Stop=DefaultTimeStop
ControlSimulation=NO

DF
DF
DefaultNumericStart=0
DefaultNumericStop=100
DefaultTimeStart=0 usec
DefaultTimeStop=100 usec

ParamSweep
Sweep1
SweepVar="TimeBaseDelay"
SimInstanceName[1]="DF"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=0
Stop=4e-3
Step=1e-3
```

Analysis

Traces form Scope in ADS Data Display





Notes

The included data display includes a page titled "Equations" containing expressions that unpack the swept data.

Knowledge Center Examples

- *8DPSK Modulator* (examples)
- *Another Method of Drawing Limit Lines* (examples)
- *Calculating NF Along an RF Path* (examples)
- *Calculating Q on a Resonator* (examples)
- *Circuit Optimization for Diff and Comm Mode* (examples)
- *Class C Amp Design Using Loadpull* (examples)
- *Constant Mismatch Analysis of Power RF Transistors* (examples)
- *Constant VSWR Circles* (examples)
- *Coplanar Differential Lines - Finite GND* (examples)
- *Deriving Differential Impedance* (examples)
- *Design Name - Time Stamp Example* (examples)
- *Differential Impedance - Coupled Lines* (examples)
- *Eye Diagram Optimization* (examples)
- *find index spec* (examples)
- *Find Z0* (examples)
- *Fixture Deembed Example* (examples)
- *Frequency Dependent Lumped Components* (examples)
- *Frequency Divider Simulations* (examples)
- *Group Delay Using DDS* (examples)
- *Injection Locking Oscillator Simulation* (examples)
- *Limit Lines in the Data Display* (examples)
- *Loadpull Contours of Oscillators* (examples)
- *Loadpull Simulation with a Modulated Source* (examples)
- *Macro Model S-Parameter Optimization* (examples)
- *Measuring Transient Settling Time* (examples)
- *Mixed Mode S-Parameter Basics DDS Template* (examples)
- *Optimization or Yield Analysis Goal (Sloped or Curved Line)* (examples)
- *Optimizing a Nonlinear Capacitor Model* (examples)
- *PCI Express Examples Workshop* (examples)
- *PLL Example with an Active Filter* (examples)
- *Save Simulation Data to ASCII* (examples)
- *Simple Sourcepull and Loadpull Explained* (examples)
- *SI Primer (Convolutions)* (examples)
- *SI Primer (TDR and TDT)* (examples)
- *S-Parameter Simulations Using HB* (examples)
- *S-Parameters Versus Bias* (examples)
- *Swept Optimization and Simulation (AEL Script)* (examples)
- *Synthesizing Geometries of Inductors* (examples)
- *Time-Domain Optimization* (examples)
- *TOI and SOI Example* (examples)
- *Two-Tone Loadpull Simulation* (examples)
- *VCO Behavioral Model from Simulated Data* (examples)
- *Waveprobe Model to Measure Forward and Reverse Power* (examples)

Disclaimer

Unsupported Examples
USER ACKNOWLEDGES THAT THE KNOWLEDGE CENTER EXAMPLES ARE EXPERIMENTAL, ARE PROVIDED "AS-IS", AND HAVE NOT COMPLETED AGILENT'S FULL QUALITY ASSURANCE PROGRAM AND MAY HAVE ERRORS OR DEFECTS. AGILENT MAKES NO EXPRESS OR IMPLIED WARRANTY OF ANY KIND WITH RESPECT TO THESE EXAMPLES, AND SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Agilent shall not be responsible for any loss or damage to User, its customers, or any other third parties caused by the examples. Agilent shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort or other legal theory, arising out of the use of these examples. Agilent shall have no obligation to maintain or support the examples.

Note
Example prior to ADS2011 can be located at 8DPSK Modulator.

8DPSK Modulator

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/8DPSK_wrk

Description

This workspace demonstrates one way to implement an 8DPSK modulator using Agilent Ptolemy. A combination of numeric and timed components are used. Other approaches exist for such an implementation.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Another Method of Drawing Limit Lines.

Another Method of Drawing Limit Lines

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/SPEC_LINE2_wrk

Description

Limit lines (specification lines) can be drawn on a rectangular chart for a quick assessment of circuit performance. However, they have their own independent variables because of the way the vectors are created and make the plot look busy. Especially, if the x-axis is frequency. This example shows how to create spec lines and still retain the clean frequency format from ADS.

After unarchiving the workspace, the user can open the data display immediately to learn about this example. No need to rerun the simulation.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Calculating NF Along an RF Path.

Calculating NF Along an RF Path

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/INTER_NF_wrk

Description

This example workspace contains schematic set-ups and data display calculations to measure noise figure along an RF signal path using the IEEE definition. This is useful for noise figure budget analysis.

This example has been updated for ADS 2008 and utilizes the P_Probe power probe component.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Calculating Q on a Resonator.

Calculating Q on a Resonator

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/Q_CalculationR2_wrk

Description

This example shows how to compute the Q of a resonator using markers and equations in the data display.

Updated September 2008 to show simpler method of computing Q.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Circuit Optimization for Diff and Comm Mode.

Circuit Optimization for Differential and Common Mode Impedances for Coupled Lines

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/ZDIFF_CM_wrk

Description

This example shows how to calculate even and odd mode impedance for coupled transmission line, which are directly related to differential and common mode impedance.

⚠ Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Class C Amp Design Using Loadpull.

Class C Amplifier Design Using Load and Source Pull Simulation

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/ClassC_AmpDesR2_wrk

Description

This example shows a sequence of steps for designing a Class C amplifier.

A load pull was run on an active device, and a load impedance was chosen that was a compromise between PAE and power delivered. Impedance values at the 2nd and 3rd harmonic frequencies were chosen experimentally. Then a matching network was designed to set the desired impedances at the fundamental and harmonic frequencies. A source pull was run with the load matching network in place, and then a source matching network was designed. Finally the amplifier including input and output matching networks was simulated as a function of input power level, giving about 80% PAE and almost 37 dBm power delivered (using ideal, lumped-element components in the matching networks, however.)

See Also

Loadpull DesignGuide for ADS 2009 Update 1 with new content (This is the most advanced Load Pull DesignGuide for use with ADS 2009 Update 1, but it does not have the enhancements of the ADS 2011 version.)

⚠ Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Constant Mismatch Analysis of Power RF Transistors.

Constant Mismatch Analysis of Power RF Transistors

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/LoadMismatchAnalysisR1_wrk

Description

Problem: You're designing a power amplifier and know the optimal load impedance to present to the device you are using. What happens (how much degradation in output power, power dissipated within the device, efficiency, etc.) if the load impedance is not at this optimal value?

This example shows a method of simulating how the output power, dissipated power, drain efficiency, and input return loss of a device or amplifier vary as the VSWR of the load reflection coefficient (relative to the optimum value) is varied. The magnitude of the VSWR may be swept in an arbitrary range and the phase of the load impedance used to generate the VSWR may be swept over an arbitrary range of values.

This technique was created by John Pritiskutch and Craig Rotay of STMicroelectronics. See their article discussing this technique: "A Constant Mismatch Analysis of Power RF Transistors Using EDA Tools," High Frequency Electronics, June 2008.

⚠ Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Constant VSWR Circles.

Constant VSWR Circles

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/vswr_circle_wrk

Description

In ADS there is no direct way to plot constant VSWR circles, which are frequently used for LNA design. In this example it is shown how to accomplish this task using a few equations on the data display based on a textbook example.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Coplanar Differential Lines - Finite GND.

Coplanar Differential Lines - Finite GND

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/CPW_Diff_Lines_Finite_wrk

Description

This workspace shows how to simulate a coplanar differential line with a finite ground plane. Both the conductors and the groundplanes are drawn on a strip layer and the ground planes are connected to ground with via holes.

The Momentum simulation is using Single ports only, also for the ground planes. We then recombine the S-parameters in a schematic simulation using the correct ground references for each port.

Simulating this structure as a slot layer is much more efficient. This method should only be used when your groundplanes are small and you are concerned about end effects etc.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Deriving Differential Impedance.

Deriving Differential Impedance

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/differential_lines_wrk

Description

This example shows how to find the differential impedance from a pair of microstrip lines. The S-parameters of a coupled pair of lines are simulated with Momentum using 4 ports. The differential input impedance is found by a S-Parameter circuit analysis where source and load impedances are swept. Using the data display, the differential impedance is the sweep value of impedance that results in the smallest reflection.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Design Name - Time Stamp Example.

Design Name - Time Stamp examples

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/name_timeR1_wrk

Description

Example of how to create an AEL function to display design workspace information (design name, time and environment variables) in the data set and data display.

This workspace shows various ways of displaying information such as **design name**, **time and environmental variables**.

In the workspace directory you will find a custom AEL Expression, **design_name.ael**. Once you have installed this function you can use it in a MeasEqn on a schematic to send the current design name to the dataset. Please see A_README in the workspace for more information.

Notes from A_README

This workspace illustrates various methods of displaying information like design name, time and environmental variables. Open the design name_time and simulate it. The corresponding data display shows the values of the various variables.

Note
A custom AEL Expression, **design_name.ael**, needs to be installed if you wish to get the design name into the dataset.

1. Copy the file design_name.ael to the folder $HOME/hpeesof/expressions/ael.
2. $HOME is your ADS home directory (typically c:\users\default on PC).
3. The expressions folder does not exist by default; you may need to create it.
4. Create a file, user_defined_fun.ael, in the same directory (if it doesn't exist already).
5. Open user_defined_fun.ael in a text editor and add the following line:
   `load("design_name");`
6. Restart ADS.

See AEL Expressions Installation Instructions for more information about installing custom AEL Expressions.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Differential Impedance - Coupled Lines.

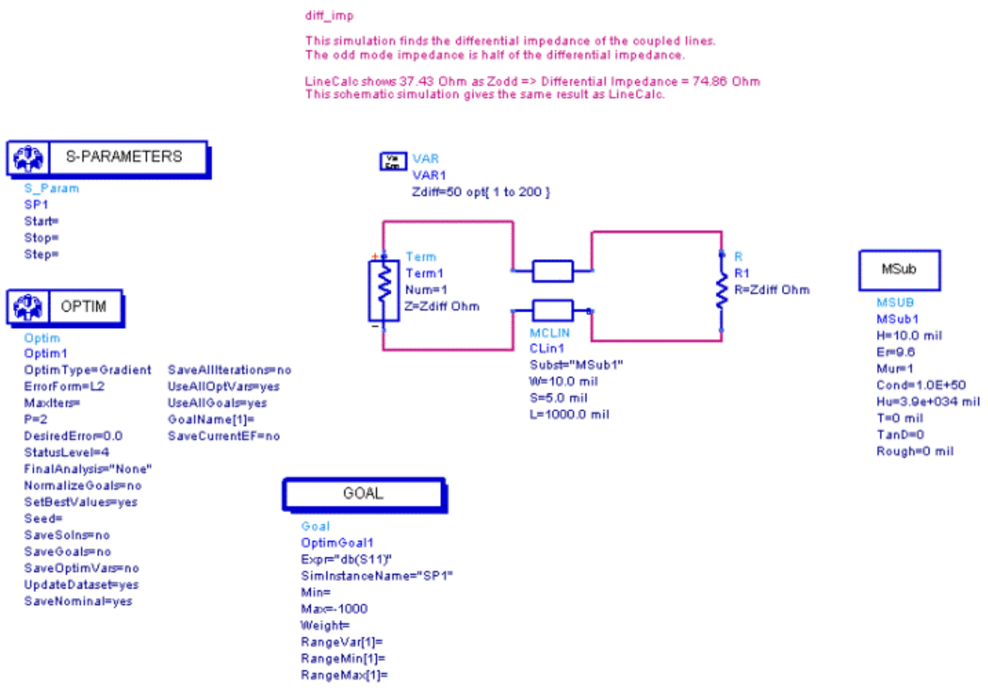
Two Methods to Compute Differential and Common Mode Impedances for Coupled Lines

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/diff_ex_wrk

Description

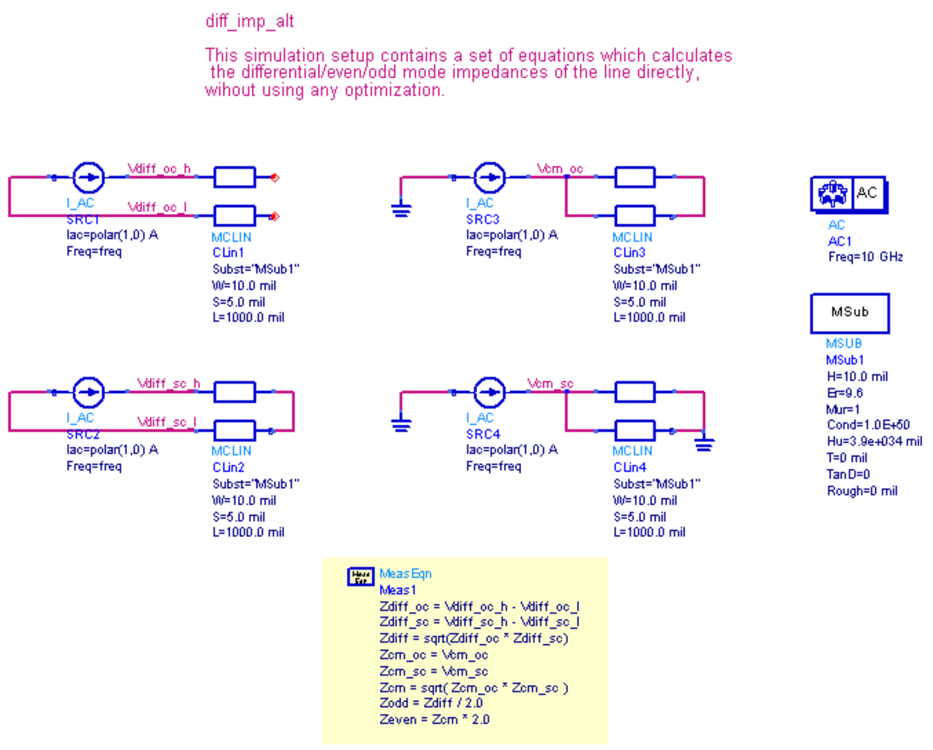
This example contains two examples of how to calculate the differential impedance of a line:

diff_imp



Using an S-parameter optimization to find the differential- and odd-mode impedances of the line.

diff_imp_alt


diff_imp_alt

This simulation setup contains a set of equations which calculates the differential/even/odd mode impedances of the line directly, wihout using any optimization.

```
MeasEqn
Meas1
Zdiff_oc = Vdiff_oc_h - Vdiff_oc_l
Zdiff_sc = Vdiff_sc_h - Vdiff_sc_l
Zdiff = sqrt(Zdiff_oc * Zdiff_sc)
Zcm_oc = Vcm_oc
Zcm_sc = Vcm_sc
Zcm = sqrt( Zcm_oc * Zcm_sc )
Zodd = Zdiff / 2.0
Zeven = Zcm * 2.0
```

Using an AC simulation and a set of equations to calculate the differential-, even- and odd-mode impedances directly.

Click on the thumbnail images above to see full size schematics.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Eye Diagram Optimization.

Eye Diagram Optimization

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/Optimizing_Eye_KC_wrk

Description

This Example shows how you can optimize your Eye Diagram. You can use measurements such as Eye Height, Eye Width, Eye Opening Factor, Eye SNR, Rise/Fall Time and Jitter, and others.

Open the schematic Test_Circuit_2. Push into the Control_2 sub-circuit to access the Measurements and Optimization Setup.

Meta Description

- Article from Printed Circuit Design & Manufacture magazine on Thursday, 31 August 2006... entitled "Optimizing Interconnect Performance with Eye Diagrams" by Agilent's By Sanjeev Gupta and Gunnar Boe: http://pcdandm.com/cms/cms/content/view/2876/95/

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at find index spec.

find_index_spec

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/find_index_spec_wrk

Description

An alternative to the find_index() function. This function works better in some cases where the data is non-monotonic.

This function does find the index of the independent variable which is closest to the lookup value. It supports multi-dimensional data up to 3 swept parameters.

The workspace contains the AEL function, find_index_spec.ael, as well as two test cases.

See AEL Expressions Installation Instructions for more information about installing custom AEL Expressions. It is also explained in Introduction to Measurement Expressions in ADS manuals.

⚠ Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Find Z0.

Find Z0

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/find_Zo_wrk

Description

Momentum calculates Z0 when you are using Single Ports, but not when you are using Internal Ports. Here is one example of how to find Z0 by optimization.

⚠ Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Fixture Deembed Example.

Fixture deembed example

Location (in ADS 2011+):

- $HPEESOF_DIR/examples/KC_Examples/Deembed/deembed_2port/Fixture_deembed_wrk
- $HPEESOF_DIR/examples/KC_Examples/Deembed/deembed_2port_6A/Fixture_deembed_wrk
- $HPEESOF_DIR/examples/KC_Examples/Deembed/deembed_2port_8U1/Fixture_deembed_wrk

Description

This workspace shows how deembed data for a fixture can be generated from calibration measurements. The fixture is assumed to have perfect symmetry, so that the output half of the fixture is the mirror image of the input half of the fixture. The half fixture is not assumed to be symmetrical.

The calibration measurements include thru measurements and S11 of a known load.

Here are the steps for deembedding S-parameters for a DUT from measurements taken of the DUT in a fixture. You need three calibration devices that can be placed in the fixture: a transmission line of length L, a transmission line of length 2*L, and a known load. You need a data file with S11 of the known load. Let's assume you've put the data in a dataset called **Gamma.ds**.

1. **SET UP EXAMPLE WORKSPACE WITH MEASURED OR MODELED KNOWN LOAD S11 DATA:**
   - Copy the workspace **Fixture_deembed_wrk** to a directory with write permission. Remove all of the data files from the data directory. **Store Gamma.ds** in the data directory. (In the example, this step is artificially simulated by **gamma.dsn**.)
2. **NWA CALIBRATION:**
   - Perform an appropriate network analyzer calibration at the input planes of the fixture.
3. **MEASURE S-PARAMETERS OF SHORT LENGTH "L" LINE IN FIXTURE/BRING DATA INTO ADS:**
   - Place the short transmission line in the fixture. Measure the s-parameters of the short transmission line. Put the data in a dataset called **L1.ds**. Store it in the data directory. (In the example, this step is artificially simulated by **L1.dsn**.)
4. **MEASURE S-PARAMETERS OF LONG LENGTH "2*L" LINE IN FIXTURE/BRING DATA INTO ADS:**
   - Place the long transmission line in the fixture. Measure the s-parameters of the long transmission line. Put the data in a dataset called L2.ds. Store it in the data

directory. (In the example, this step is artificially simulated by **L2.dsn**.)

5. **MEASURE KNOWN LOAD S11 DATA IN FIXTURE/BRING DATA INTO ADS:**
   - Place the known load in the fixture. Measure S11 of the known load. Put the data in a dataset called Load_meas.ds. Store it in the data directory. (In the example, this step is artificially simulated by **Load_meas.dsn**.)
6. **MEASURE DUT IN FIXTURE/BRING DATA INTO ADS:**
   - Place your device under test (DUT) in the fixture. Measure the s-parameters of the DUT. Put the data in a dataset called DUT_meas.ds. Store it in the data directory.. (In the example, this step is artificially simulated by **DUT_meas.dsn** .)
7. **RUN ADS DESIGN EQUATIONS TO REMOVE LONG AND SHORT THROUGH, EXTRACT ONLY INPUT/OUTPUT HALF FIXTURE:**
   - In ADS, open Thru.dsn and run the simulation. This will derive thru data from L1.ds and L2.ds. Results will be stored in the data directory in a file called **Thru.ds**.
8. **RUN ADS DESIGN EQUATIONS TO CALCULATE HALF-FIXTURE S-PARAMENTERS:**
   - In ADS, open Half_fixture_model.dsn and run the simulation. This will calculate the s-parameters of the half fixture from Thru.ds, Load_meas.ds, and Gamma.ds. Results will be stored in the data directory in a file called Half_fixture_model.ds.
9. **FINALLY RUN ADS DESIGN W/ Deembed2 COMPONENT TO DEEMBED DUT FROM THE FIXTURE:**
   - In ADS, open **use_Deembed2.dsn** and run the simulation. This will deembed the DUT from the fixture. The default dataset name will be use_Deembed2.ds. If desired, change the dataset name before running the simulation.

Note that steps 3, 4, 5, 7, and 8 only need to be performed once. Once you have Half_fixture_model.ds, you can reuse the model data to deembed future DUT measurements.

UPDATE NEW N-PORT DE-EMBEDDING COMPONENTS AVAILABLE STARTING IN ADS2008 Update 1 – De_EmbedN and De_EmbedSnPN (where N=2,4,6,8,10 or 12).

The attached archived file includes examples based on the Fixture dembeding discussion above that are ADS version specific. Please read the ReadME_1st.txt file included in the archive for further details.

Contents of ReadMe_1st.txt

This archived file (.zip) includes three ADS archived workspaces. They are examples on how to use the deembed components. Prior to ADS 2006A, only the "Deembed1" (1-Port) and "Deembed2" (2-Port) components were available to use in ADS RF/Analog simulations. Starting in ADS 2006A, new multi-port de-embedding components are available in the form of "DeembedN" where N=1,2,4,6,8,12 port de-embedding and are supported up to the ADS2008 release inclusive.

The proper use of Deembed2 was not well documented in the ADS manual in prior releases. In older releases (prior to ADS2006A), customers can get correct answers with Deembed2 even if they don't use it as designed.

In ADS2006A, a change was deliberately made to the Deembed2 component to bring it in line with theoretical definition and to implement the newly introduced DeembedN components. As a result, the statement in the ADS manual that for cancellation behavior the de-embed component should be set back-to-back against the SnP component, must now be strictly adhered to.

Staring in ADS2008 Update 1 two new sets of deembed components were introduced De_EmbedN and De_EmbedSnPN. These latest deembed components offer much more flexibility and ease of use when compared to the old DeembedN components. For more details about the new components, see your ADS2008 Update 1 or greater documentation.

Description of the files included:

1. deembed_2port.zap - 2 port deembed example that has been verified with ADS2004A and ADS2005A. This will **NOT** work properly in ADS2006A and later releases.
2. deembed_2port_6A.zap - 2 port deembed example (similar to above) that properly

connects the DUT and Deembed2 components back to back. This example will work properly in ADS2006A or later releases only.

3. deembed_2port_8U1.zap - 2 port deembed example (similar to above) that uses the newly introduced De_Embed2 component.

Note
Starting with ADS2008 Update 1 release the DeemdbedN components are no longer available on the compomponent palettes but will still work.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Frequency Dependent Lumped Components for ADS.

Frequency Dependent Lumped Components for ADS

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/FreqDependentRLCs_wrk

Description

Simulating frequency-dependent lumped elements in ADS is challenging, because ADS does not normally provide for frequency-related variation of lumped elements. This workspace shows both file-based and equation-based methods of making frequency-dependent lumped components.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Frequency Divider Simulations.

Frequency divider simulations, up to divide-by-128, ADS 2005A version

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/FreqDivHigherOrder_wFETsR4_wrk

Description

This example has simulations of frequency dividers. Included are simulations of dividers up to divide-by-128. There are also simulations that include a VCO at the input, and some phase noise simulations.

- This example was updated on October 28, 2005, showing a method of getting accurate phase noise simulation results when using Krylov.

- This example was updated on November 1, 2005, and now includes a phase noise simulation of the VCO with divide-by-64, that must be run on a machine with 2 GBytes of RAM.

- This example was updated on January 13, 2005. The convergence tolerances, I_RelTol and V_RelTol, that affect the transient simulations to generate an initial guess, where loosened. **This dramatically speeds up the transient part of the simulations.**

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Group Delay Using DDS.

Group Delay

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/numDerivsR1_wrk

Description

This example has been updated to use the diff() and unwrap() functions, which greatly simplify the calculation.

This workspace shows how to calculate the group delay on the data display. If you do an S-parameter simulation, you don't need to use this function, since you can just enable Group Delay in the S-parameter analysis controller. But if you are doing any other simulation, you can use this method to calculate the group delay from the phase response.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Injection Locking Oscillator Simulation.

Injection locking oscillator simulation

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/InjectionLockingOscR1_wrk

Description

This example shows some injection locking experiments. The simulations use a MOS VCO, that uses MOSFETs as the capacitive tuning elements. One interpretation of "injection locking" is the coupling of some stable oscillating signal into a free-running oscillator in such a way that the free-running oscillator's frequency becomes the same as the more stable oscillator's. The Envelope simulator may be used for these simulations. You have to know the free-running oscillator's oscillation frequency and the stable oscillator's frequency must be within the same envelope bandwidth (=1/timestep) as the free-running oscillator's.

VCO_wFETcaps uses harmonic balance and shows the large-signal steady-state solutions versus tuning voltage.

VCO_HB_Test uses harmonic balance to simulate the VCO subcircuit used in the other simulations. It is just used to verify the performance of the subcircuit.

VCO_wFETcapsEnv simulates the VCO using Envelope. It does not run an oscillator analysis, but still detects an oscillating signal within the envelope bandwidth centered on the analysis frequency.

InjectionLockingTest simulates two identical free-running oscillators with their outputs coupled together via a resistive power combiner. The two oscillators are tuned to slightly different frequencies but end up oscillating at the same frequency in this simulation. **This shows that ADS is able to simulate multiple oscillators in the same schematic.**

InjectionLocking_wIdealSig_FSwp simulates the free-running oscillator coupled (via a resistive power combiner) to an ideal, behavioral VCO, which acts as the stable oscillator. The frequency of the ideal VCO is varied by applying a sinusoid to its control input. The output indicates approximately the range of frequencies over which the free-running oscillator is locked to the ideal VCO.

InjectionLocking_wIdealSig_PSwp simulates how the amplitude of the ideal stable signal affects the ability to achieve lock. The power of this signal is stepped in a piecewise-linear fashion. When it reaches -5 dBm, the free-running oscillator suddenly locks to it.

VCO_EnvTest_wFM applies frequency modulation to the open-loop VCO.

See Also

Starting Examples For Setting Up Injection Locking Oscillator Simulation

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Limit Lines in the Data Display.

Creating Limit Lines in the Data Display with Limitlines ADS DesignGuide...

Issue

Q: How do you add limit lines to the data display?

How to get the ADS Limitlined ADS DesignGuide...

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/Limitlines

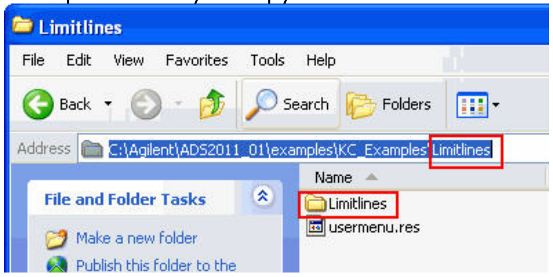
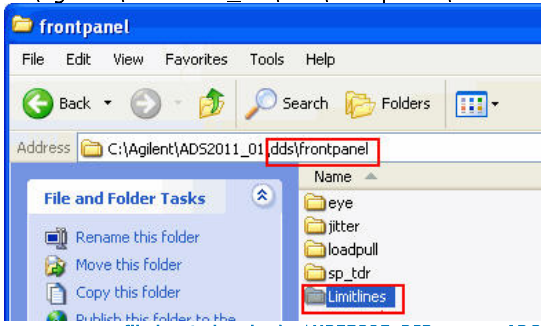
Note
As the contents of the directory shown above are not ADS workspaces, they are not visible in the Open Example window; use a file browser to access the files.

Solution

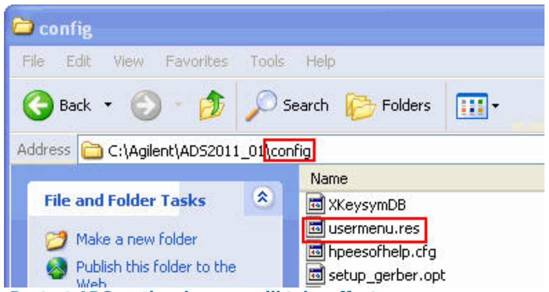
The limit lines provide a way to define user defined masks on a rectangular plot in the data display window. One can easily create a static, non intelligent mask using Limit Line utility.

Note
The Limit Line utility is a contributed functionality, and is provided as-is. Future versions of ADS may include limit line functionality that differs from this approach.

Limit Line utility can be installed using the following steps:

1. **Copy and Place the "Limitlines" directory folder located under \<$HPEESOF_DIR or your ADS 2011 Installation Directory\>/examples/KC_Examples/Limitlines to \<$HPEESOF_DIR or your ADS 2011 Installation Directory\>/dds/frontpanel directory.**
   - For example, suppose ADS 2011 is installed in $HPEESOF_DIR C:\agilent\ADS2011_01\ then use file explorer to find the following "Limitlines" example directory to copy from:
   
   Limitlines
File Edit View Favorites Tools Help
Back Search Folders
Address C:\Agilent\ADS2011_01\examples\KC_Examples\Limitlines
File and Folder Tasks
Make a new folder
Publish this folder to the
Name
Limitlines
usermenu.res
   - Then, paste the example "Limitlines" example directory to $HPEESOF_DIR C:\agilent\ADS2011_01\dds\frontpanel\ directory:
   

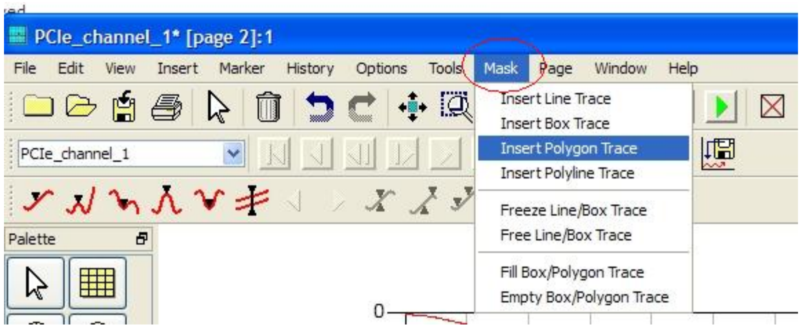
2. **Copy usermenu.res file located under \<$HPEESOF_DIR or your ADS 2011 Installation Directory\>/examples/KC_Examples/Limitlines... and paste it in <$HPEESOF_DIR>/config directory:**

3. **Restart ADS so the changes will take effect...**

Once installed the "Mask" menu will appear in the data display window:

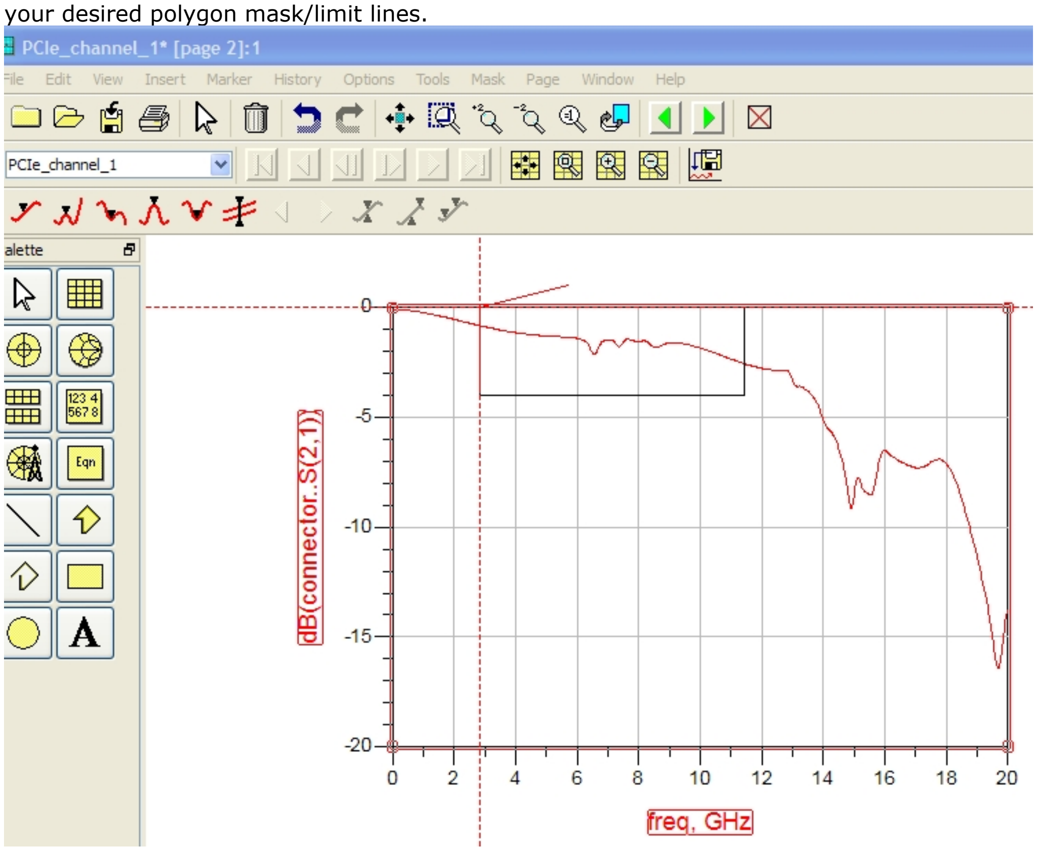
- Here is a glimpse of what the newly added Limitlines "Mask" pull down window will look like in your ADS DataDisplay window:



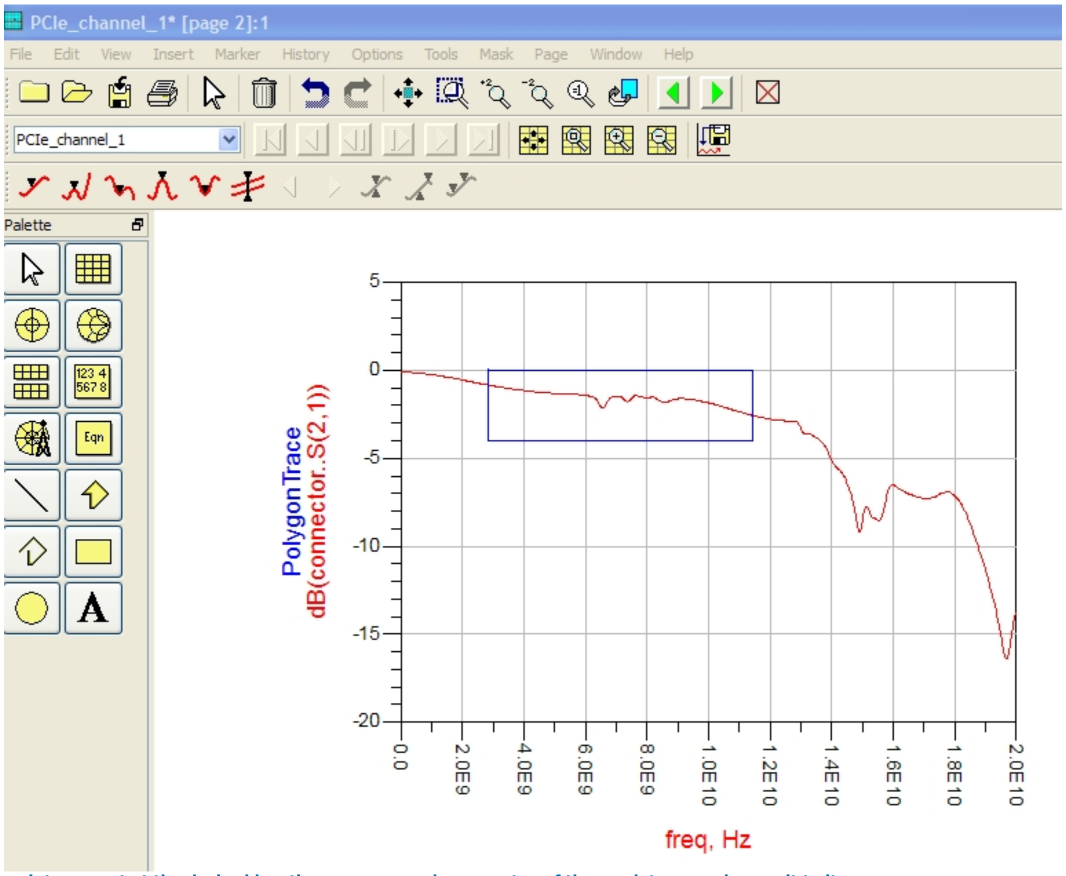
To create a mask:

1. **Select "Insert Polygon Trace" or "Insert Box Trace" from the Mask menu.**
2. **In the rectangular grid plot, create the polygon or the box using mouse clicks...**

- First use a series of left mouse clicks to place the ghost image cardinal points of your desired polygon mask/limit lines.

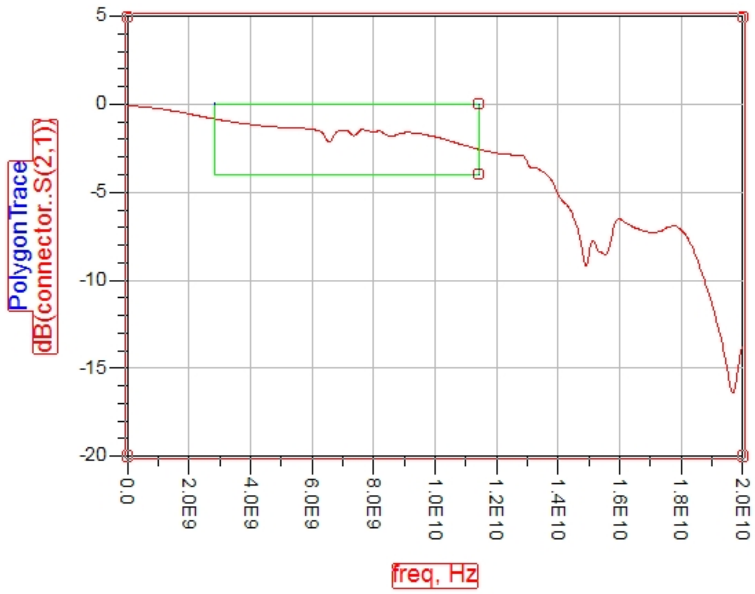

PCIe_channel_1* [page 2]:1
File Edit View Insert Marker History Options Tools Mask Page Window Help
PCIe_channel_1
alette
dB(connector..S(2,1))
0
-5
-10
-15
-20
0 2 4 6 8 10 12 14 16 18 20
freq, GHz

- To finish a desired polygone mask/limit line, do one additional final left mouse click (left double mouse click). The final shape will look like this blue polygone:

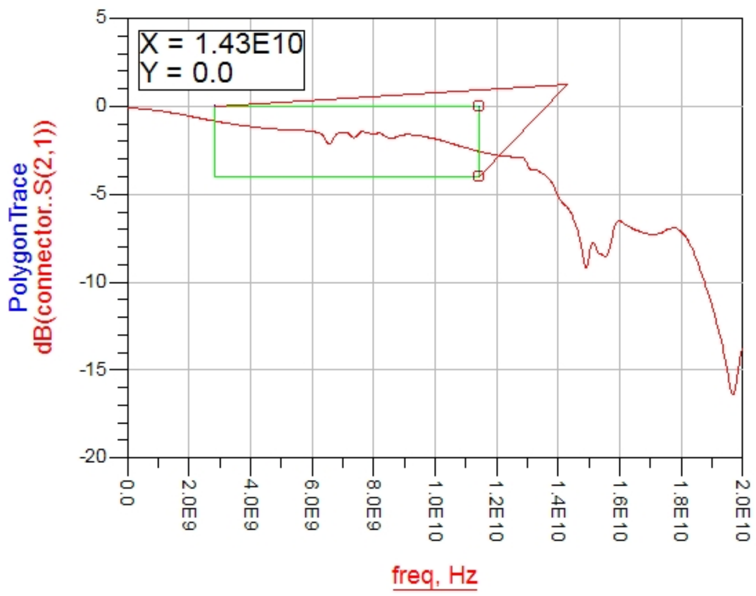


3. **If the points are not at the desired location, you can grab any vertex of the mask trace and move it to its correct location. The coordinates will be displayed when a vertex location is moved...**

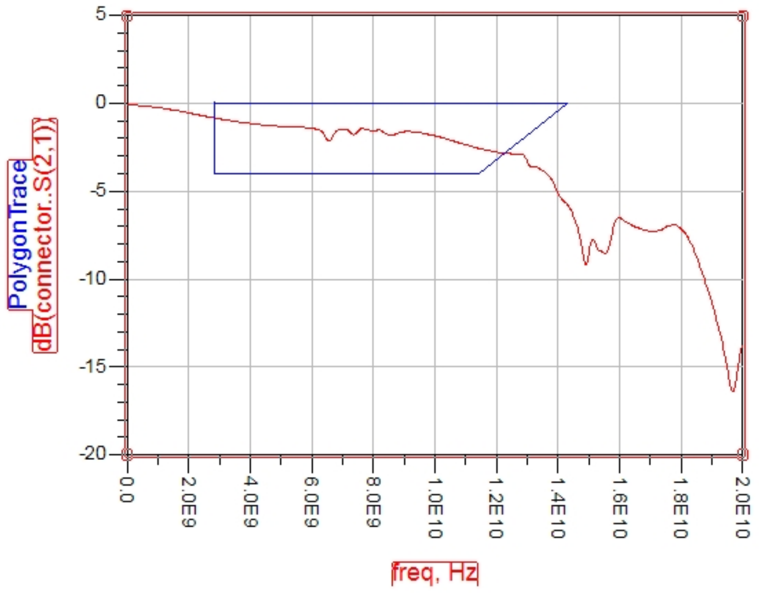
- Now if you left-click your mouse on the blue established polygone mask shape to select it, it will turn green color and vertexes will be highlighted with circles:



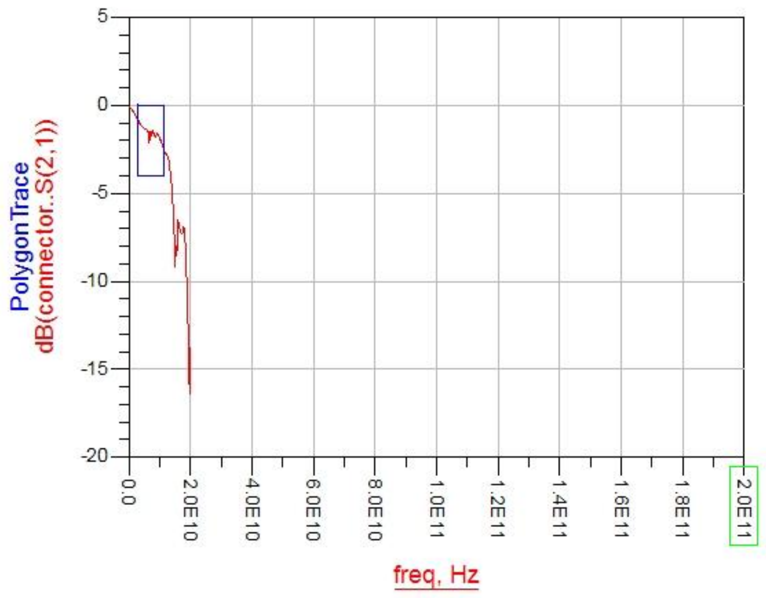
- With the green highlighted polygone mask selected via left click, hold/drag the vertex to another location... The xy coordinates will be shown on the screen:



- With one final double left mouse click, you can set a modified mask/polygone shape (back to blue color):

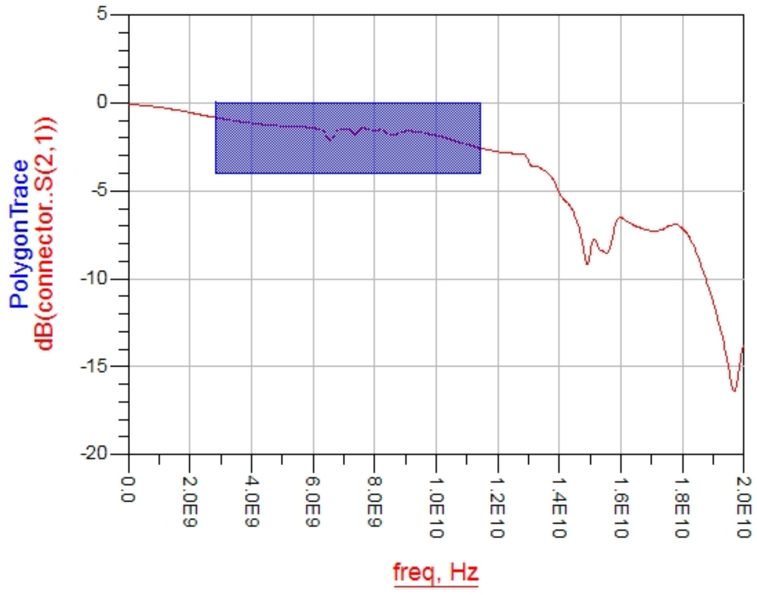


- Now, if you say modify the entire plot's scale (Ex: change max frequency point from 2.0E10 to 2.0E11, notice how the mask/polygone limit shape retains the proper scale...



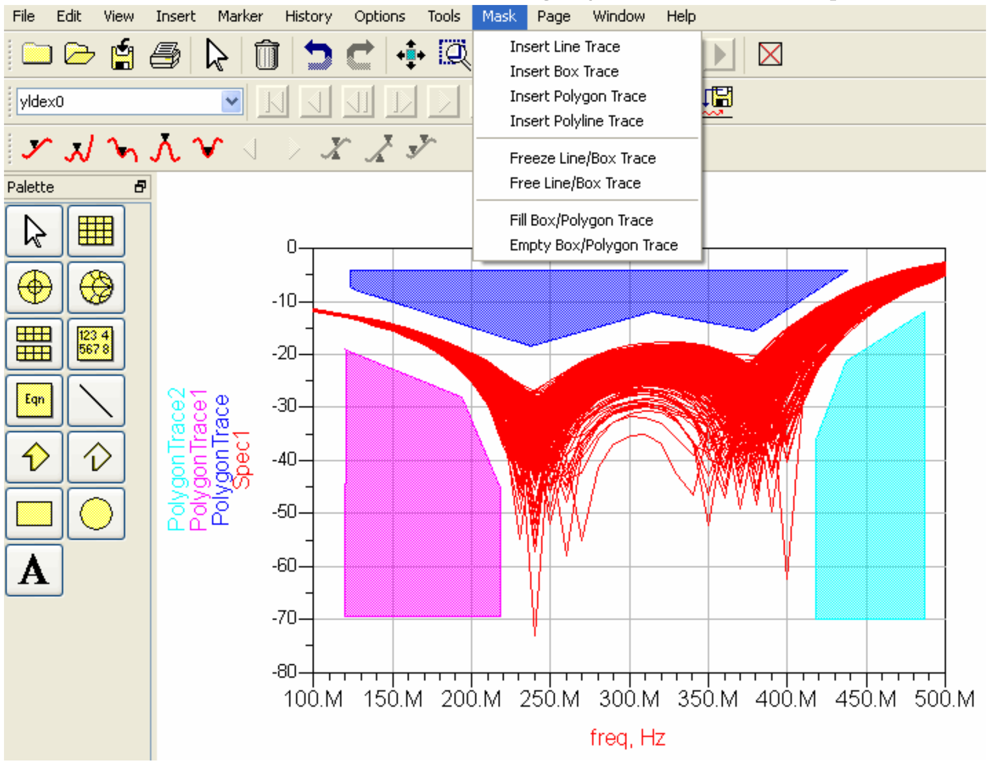
4. **To move the entire mask on XY axis, select the mask using a mouse click at the center and drag it to its correct location.**
5. **To fix the position of the mask to its position.... select the mask and than from Mask menu select... "Freeze Line/Box Trace". It will freeze the mask to... its current location and can not be changed.**
6. **To free a mask, select mask using mouse click and using Mask menu /Free Line/Box Trace.**
7. **To fill a Polygon mask, select mask using Mouse click from using Mask menu /Fill Box/Polygon Traces.**

- Here is an example of a filled-in polygone mask/limit shape:

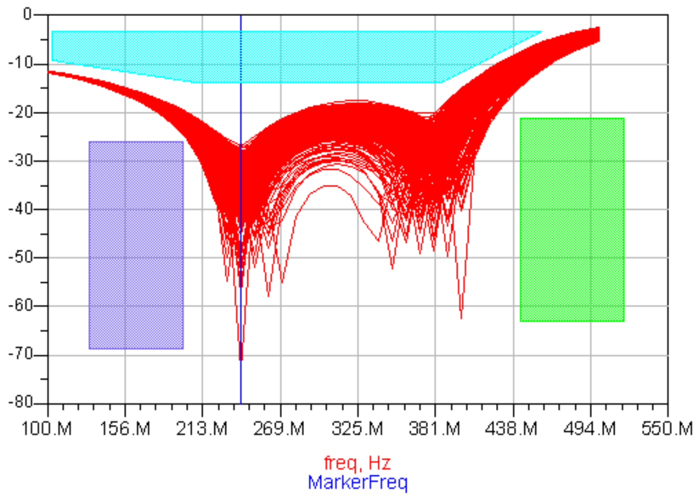


Here are some other practical examples of how the Limitlines "Mask" pulldown ADS Data Display features can be used:

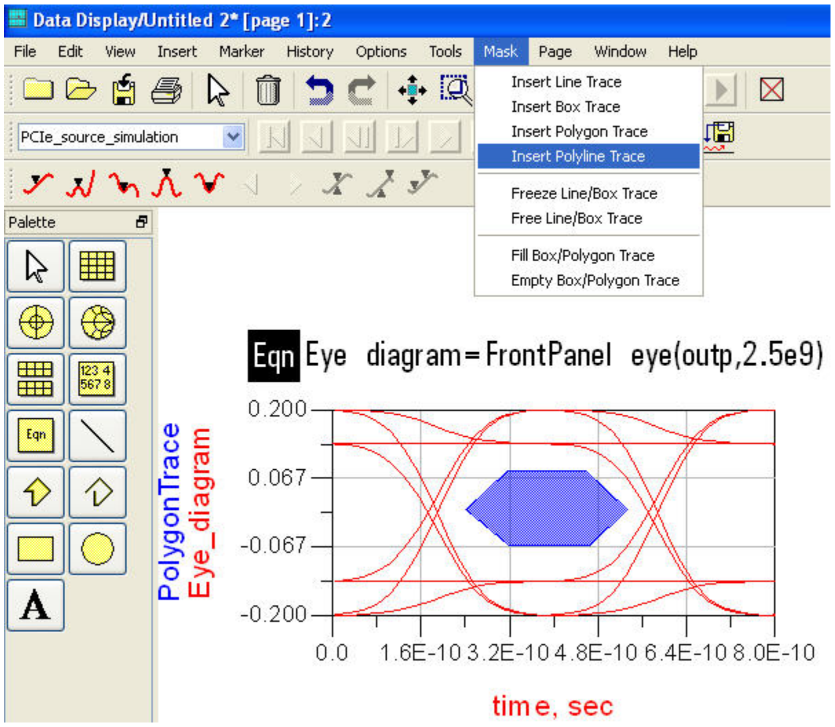
-
    - Selecting "Mask" pulldown menu:



- Final spectral mask/limit polygone shapes applied:



- Eye mask:



See Also

- **Adding a spec limit, or mask, to the DDS**
- **Adding spec lines in a rectangular plot**

Meta Description

DocID:
DocOwner:SGupta
DocCustViewable:YES
Keywords:ADS2009

Note
Example prior to ADS2011 can be located at Loadpull Contours of Oscillators

Loadpull Contours of Oscillators

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/OscPhNoiseVsLoad_wrk

Description

This example simulates the phase noise, output power and variation in fundamental frequency versus load reflection coefficient. It is based on the MW_Ckts/MMIC_Osc_wrk ADS example workspace.

This workspace uses an EEsof DemoKit, that is in the $HPEESOF_DIR/examples/DesignKit directory.

Note
Example prior to ADS2011 can be located at Loadpull Simulation with a Modulated Source.

Loadpull Simulation with a Modulated Souce - ACPR, Pdel, PAE Contours

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/LoadPull/LoadPullACPR_R2_wrk

Description

This example shows how to do loadpull simulations while using a CDMA 2000 signal source. (It could be modified to use other modulated signal sources.) These simulations generate contours that indicate load impedances that, when presented to the output of a device (along with the specified source impedances and available source power), would cause a certain power to be delivered to the load. Contours for ACPR, power delivered, and power-added efficiency (PAE) are all generated.

The PAE (power-added efficiency) calculation has been updated.

See Also

If you have ADS, from any schematic, select **DesignGuide > Loadpull > WCDMA Loadpull > Constant Power Delivered...** for better examples.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

See also: *Simple Sourcepull and Loadpull Explained* (examples).

Note
Example prior to ADS2011 can be located at Macro Model S-Parameter Optimization.

Macro Model S-parameter Optimization from Multiple S-Parameter Files

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/optim_multi_sparam_files_wrk

Description

This example is designed to solve the following problem: you have multiple S-parameter files (which may come from various sources, such as measurement, parametric simulation, and so on). You want to optimize your macro model so that you will get the element values of the model for each sparameter file.

The sparameter files in this example are named:
spar_1.s2p, spar_2.s2p to spar_5.s2p.

To use this example you only need the S-parameter files and your macro-model. The workspace has one design called optim_sp.

Under the data folder you will find the touchstone .s2p files.

Optim_sp consists of: a var block that reads by means of a DAC the touchstone .s2p files, a special variable that sweeps the file name as a function of sweep index, a measurement equation to create the optimization goals, a sweep parameter, a simulation SP, and the optimizer.

The optimization method we use here is: for each sweep index, we optimize the model and keep the last optimization value for display.

In this example the value of the PI capacitor is being optimized and its value will be 1 to 5. This is the same as optimization index and it is like this just for simplicity.

In data display you will see OPTSOLNVALS.c1, which is a function of index and has the final optimized capacitor C1 values.

Meta Description

EX: Create Multiple S-parameter files from a Sweep S-parameter Simulation

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Measuring Transient Settling Time.

Measuring the Settling Time of a Transient Signal

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/T_SettleR2_wrk

Description

This example shows two methods of measuring the settling time of a transient signal.

Final_Known shows how to measure the settling time when you know the final value.

Final_Unknown shows how to measure the settling time when you do not know the final value.

In either case, you have to enter a value (Veps), which defines a window around the final value and will be the criterion for determining whether the signal has settled or not.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Mixed Mode S-Parameter Basics DDS Template.

ADS Mixed-Mode S-Parameter Basics Data Display Template

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/mixed_mode_wrk

Description

The attached ADS2011 workspace contains a design file with MCLIN coupled microstrip line (2 port single-ended in/2 port single-ended out), a corresponding Data Display file (.dds), and a set of sample data sets (.ds files, comparing linecalc, circuit simulator model and momentum results), for illustrating conversion of measured single ended S-parameters (4-port) into mixed mode/differential (2-port) equivalent. Save the Data Display as a template, in order to reuse the equations and formatted display of smith and rectangular graphs.

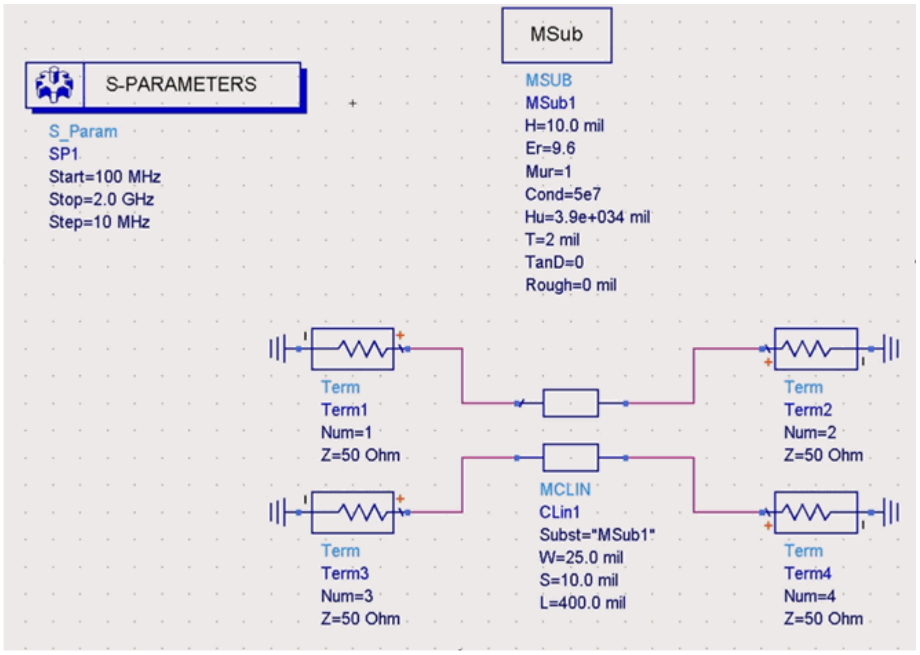
The use of balanced (differential) devices in both handsets and base stations continues to increase due to their ability to provide higher system performance, lower noise, and lower power consumption than their more common single-ended counterparts.

Like other RF devices, the evaluation of balanced devices is necessary to ensure optimal circuit and system performance, but balanced device measurements present many challenges such as interfacing a balanced DUT to single-ended test equipment, coping with non-50 ohm reference impedances, and having a traceable calibration technique. In addition, new parameters such as the degree of balance and conversion between differential and common modes must be understood to ensure the results that we expect.

Summary of Example Workspace Contents:

Here is a glimpse of what can be found in the example that ships with ADS 2011 workspace attached below:

1.
    1. **ADS "mixed_mode" example circuit:**



S_Param
SP1
Start=100 MHz
Stop=2.0 GHz
Step=10 MHz

MSUB
MSub1
H=10.0 mil
Er=9.6
Mur=1
Cond=5e7
Hu=3.9e+034 mil
T=2 mil
TanD=0
Rough=0 mil

Term1
Num=1
Z=50 Ohm

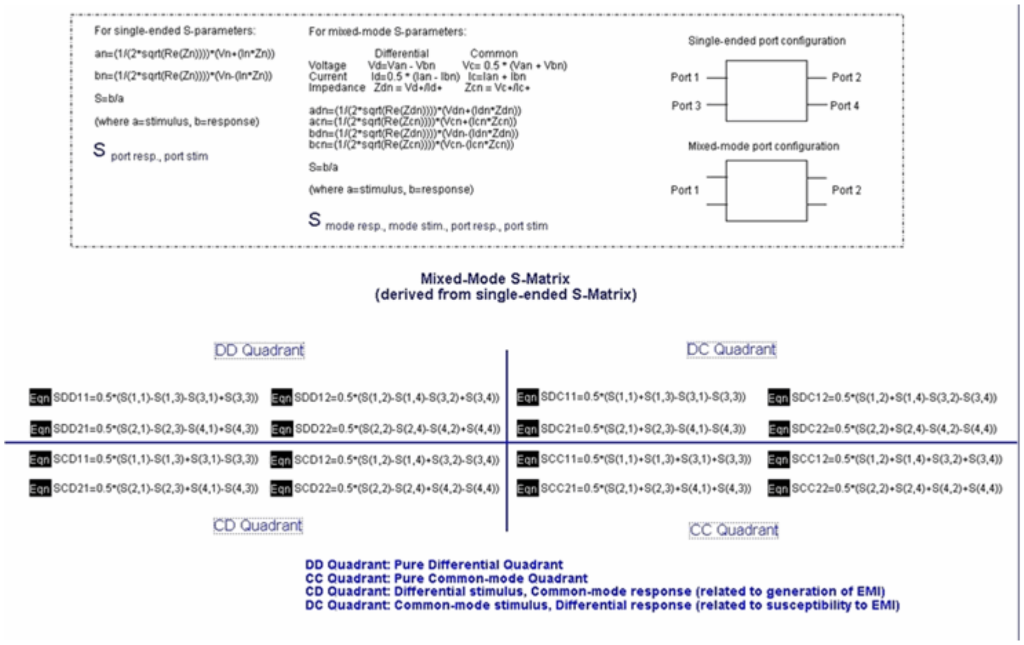
Term2
Num=2
Z=50 Ohm

CLin1
Subst="MSub1"
W=25.0 mil
S=10.0 mil
L=400.0 mil

Term3
Num=3
Z=50 Ohm

Term4
Num=4
Z=50 Ohm

2. **"mixed_mode" DDS [Mixed-mode equations] page:**


Single-ended port configuration
Port 1
Port 2
Port 3
Port 4
Mixed-mode port configuration
Port 1
Port 2

For single-ended S-parameters:

an=(1/(2*sqrt(Re(Zn))))*(Vn+(In*Zn))

bn=(1/(2*sqrt(Re(Zn))))*(Vn-(In*Zn))

S=b/a

(where a=stimulus, b=response)

S port resp., port stim

For mixed-mode S-parameters:

	Differential	Common
Voltage	Vd=Van - Vbn	Vc= 0.5 * (Van + Vbn)
Current	Id=0.5 * (Ian - Ibn)	Ic=Ian + Ibn
Impedance	Zdn = Vd+/Id+	Zcn = Vc+/Ic+

adn=(1/(2*sqrt(Re(Zdn))))*(Vdn+(Idn*Zdn))
acn=(1/(2*sqrt(Re(Zcn))))*(Vcn+(Icn*Zcn))
bdn=(1/(2*sqrt(Re(Zdn))))*(Vdn-(Idn*Zdn))
bcn=(1/(2*sqrt(Re(Zcn))))*(Vcn-(Icn*Zcn))

S=b/a

(where a=stimulus, b=response)

S mode resp., mode stim., port resp., port stim

Mixed-Mode S-Matrix
(derived from single-ended S-Matrix)

DD Quadrant

Eqn SDD11=0.5*(S(1,1)-S(1,3)-S(3,1)+S(3,3))
Eqn SDD12=0.5*(S(1,2)-S(1,4)-S(3,2)+S(3,4))
Eqn SDD21=0.5*(S(2,1)-S(2,3)-S(4,1)+S(4,3))
Eqn SDD22=0.5*(S(2,2)-S(2,4)-S(4,2)+S(4,4))

DC Quadrant

Eqn SDC11=0.5*(S(1,1)+S(1,3)-S(3,1)-S(3,3))
Eqn SDC12=0.5*(S(1,2)+S(1,4)-S(3,2)-S(3,4))
Eqn SDC21=0.5*(S(2,1)+S(2,3)-S(4,1)-S(4,3))
Eqn SDC22=0.5*(S(2,2)+S(2,4)-S(4,2)-S(4,4))

CD Quadrant

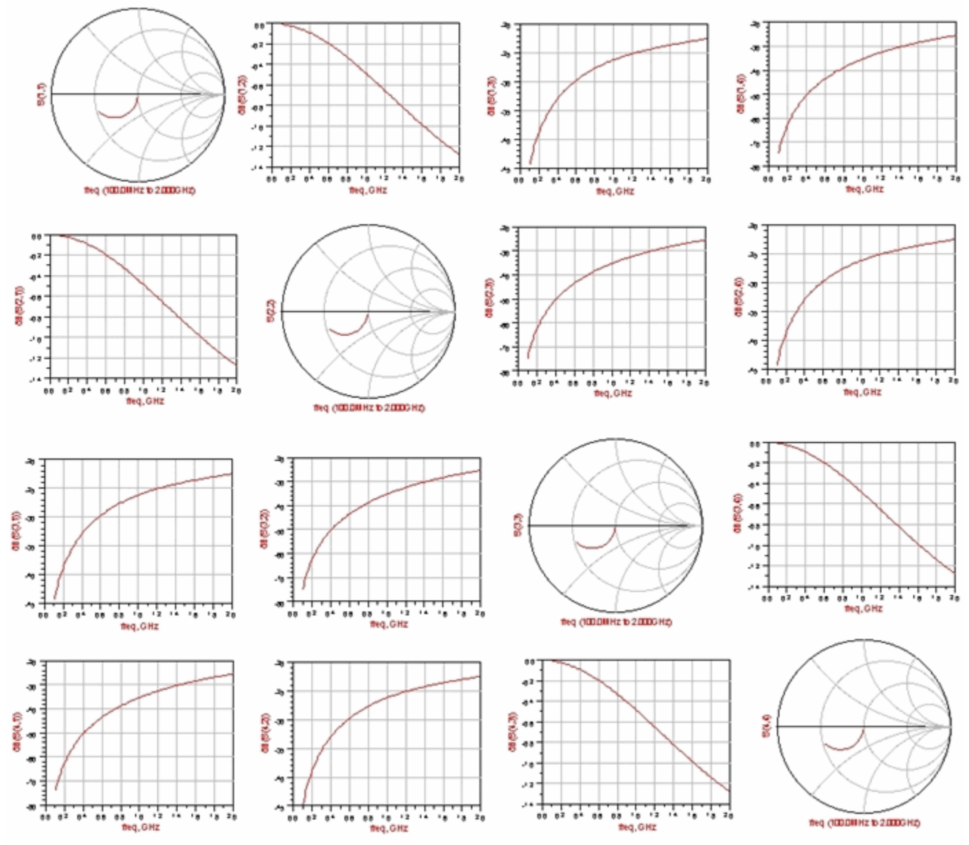
Eqn SCD11=0.5*(S(1,1)-S(1,3)+S(3,1)-S(3,3))
Eqn SCD12=0.5*(S(1,2)-S(1,4)+S(3,2)-S(3,4))
Eqn SCD21=0.5*(S(2,1)-S(2,3)+S(4,1)-S(4,3))
Eqn SCD22=0.5*(S(2,2)-S(2,4)+S(4,2)-S(4,4))

CC Quadrant

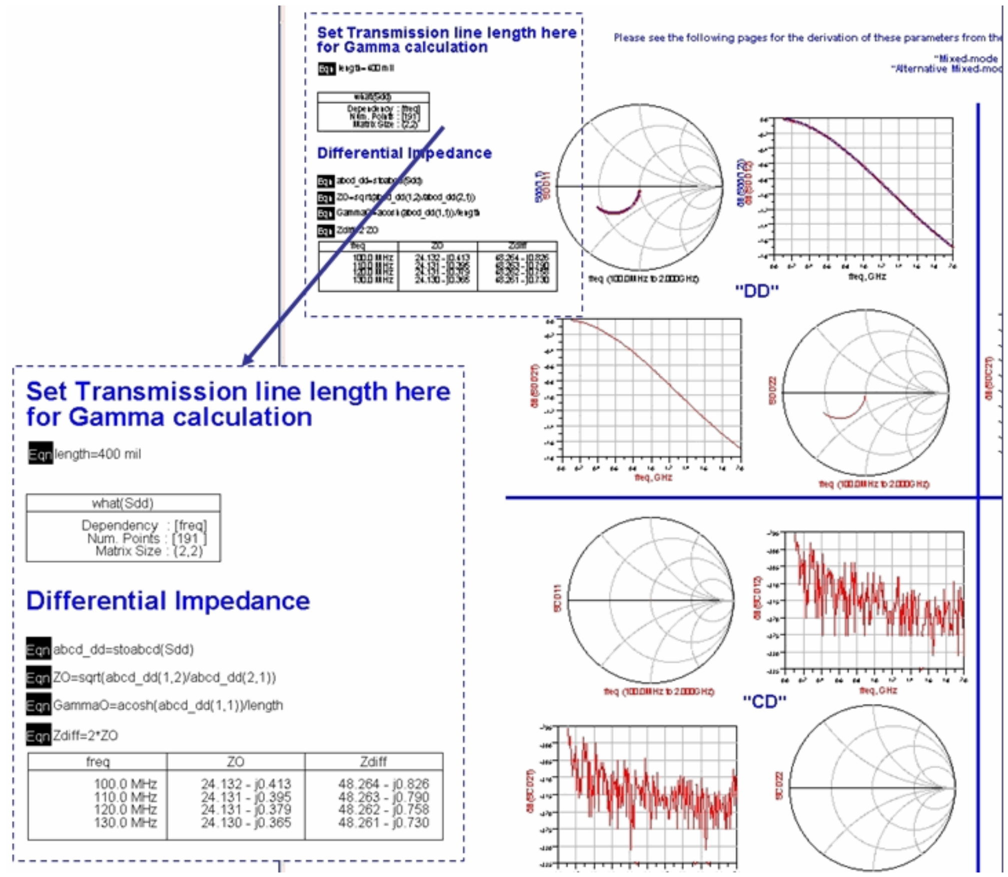
Eqn SCC11=0.5*(S(1,1)+S(1,3)+S(3,1)+S(3,3))
Eqn SCC12=0.5*(S(1,2)+S(1,4)+S(3,2)+S(3,4))
Eqn SCC21=0.5*(S(2,1)+S(2,3)+S(4,1)+S(4,3))
Eqn SCC22=0.5*(S(2,2)+S(2,4)+S(4,2)+S(4,4))

DD Quadrant: Pure Differential Quadrant
CC Quadrant: Pure Common-mode Quadrant
CD Quadrant: Differential stimulus, Common-mode response (related to generation of EMI)
DC Quadrant: Common-mode stimulus, Differential response (related to susceptibility to EMI)

3. **"mixed_mode" DDS [Single-ended S-parameters] page:**



4. **"mixed_mode" DDS [Mixed-mode S-parameters] page left-half:**


Please see the following pages for the derivation of these parameters from the
"Mixed-mode
"Alternative Mixed-mod
"DD"
"CD"
freq (100.0MHz to 2.000GHz)
freq, GHz

Set Transmission line length here for Gamma calculation

Eqn length=400 mil

what(Sdd)
Dependency : [freq] Num. Points : [191] Matrix Size : (2,2)

Differential Impedance

Eqn abcd_dd=stoabcd(Sdd)

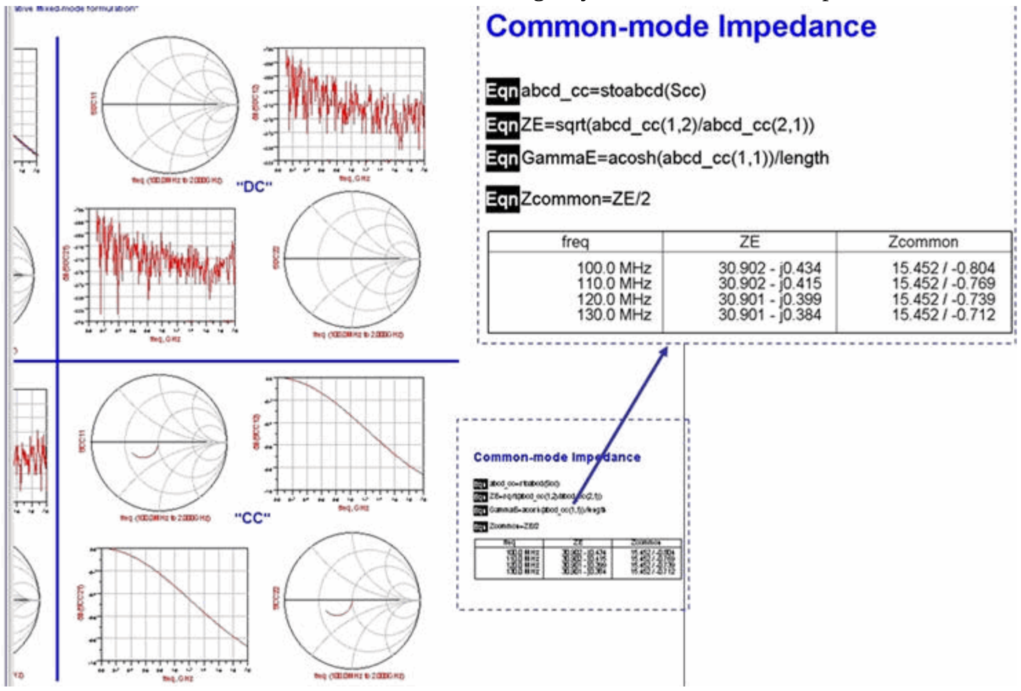
Eqn ZO=sqrt(abcd_dd(1,2)/abcd_dd(2,1))

Eqn GammaO=acosh(abcd_dd(1,1))/length

Eqn Zdiff=2*ZO

freq	ZO	Zdiff
100.0 MHz	24.132 - j0.413	48.264 - j0.826
110.0 MHz	24.131 - j0.395	48.263 - j0.790
120.0 MHz	24.131 - j0.379	48.262 - j0.758
130.0 MHz	24.130 - j0.365	48.261 - j0.730

5. **"mixed_mode" DDS [Mixed-mode S-parameters] page right-half:**



Common-mode Impedance

Eqn abcd_cc=stoabcd(Scc)

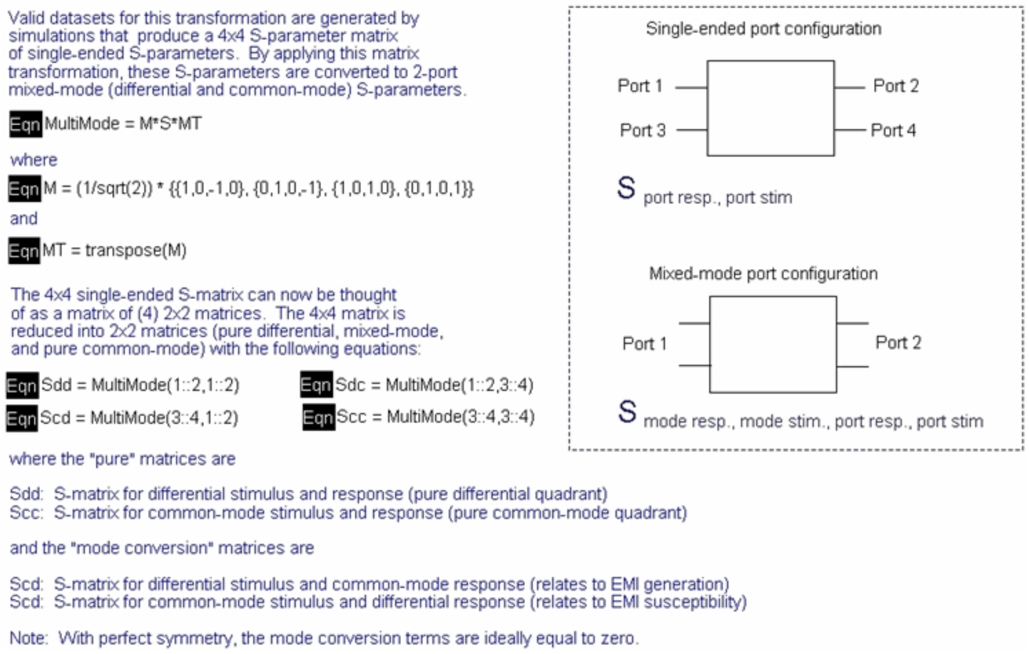
Eqn ZE=sqrt(abcd_cc(1,2)/abcd_cc(2,1))

Eqn GammaE=acosh(abcd_cc(1,1))/length

Eqn Zcommon=ZE/2

freq	ZE	Zcommon
100.0 MHz	30.902 - j0.434	15.452 / -0.804
110.0 MHz	30.902 - j0.415	15.452 / -0.769
120.0 MHz	30.901 - j0.399	15.452 / -0.739
130.0 MHz	30.901 - j0.384	15.452 / -0.712

6. **"mixed_mode" DDS [Alternative mixed-mode formulation] page:**


Single-ended port configuration
Port 1
Port 2
Port 3
Port 4
S port resp., port stim
Mixed-mode port configuration
Port 1
Port 2
S mode resp., mode stim., port resp., port stim

Valid datasets for this transformation are generated by simulations that produce a 4x4 S-parameter matrix of single-ended S-parameters. By applying this matrix transformation, these S-parameters are converted to 2-port mixed-mode (differential and common-mode) S-parameters.

Eqn MultiMode = M*S*MT

where

Eqn M = (1/sqrt(2)) * {{1,0,-1,0}, {0,1,0,-1}, {1,0,1,0}, {0,1,0,1}}

and

Eqn MT = transpose(M)

The 4x4 single-ended S-matrix can now be thought of as a matrix of (4) 2x2 matrices. The 4x4 matrix is reduced into 2x2 matrices (pure differential, mixed-mode, and pure common-mode) with the following equations:

Eqn Sdd = MultiMode(1::2,1::2)

Eqn Sdc = MultiMode(1::2,3::4)

Eqn Scd = MultiMode(3::4,1::2)

Eqn Scc = MultiMode(3::4,3::4)

where the "pure" matrices are

Sdd: S-matrix for differential stimulus and response (pure differential quadrant)
Scc: S-matrix for common-mode stimulus and response (pure common-mode quadrant)

and the "mode conversion" matrices are

Scd: S-matrix for differential stimulus and common-mode response (relates to EMI generation)
Scd: S-matrix for common-mode stimulus and differential response (relates to EMI susceptibility)

Note: With perfect symmetry, the mode conversion terms are ideally equal to zero.

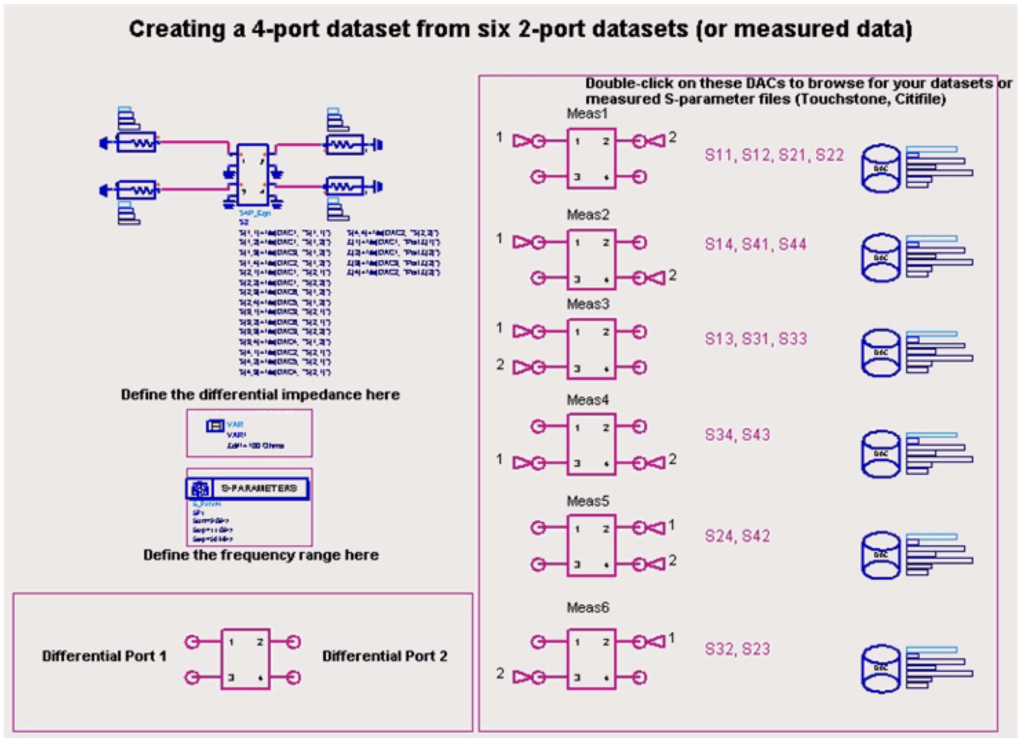
Alternative Reference: Single-Ended To Mixed-Mode S-parameter Conversion Equations also can be found in ADS "Signal Integrity Applications" Design Guide.

From an ADS 2005A design schematic, select /DesignGuide/Signal Integrity Applications/... and one of the following to find the same ADS data display template which demonstrates converting single-ended s-parameters to mixed-mode s-parameters:

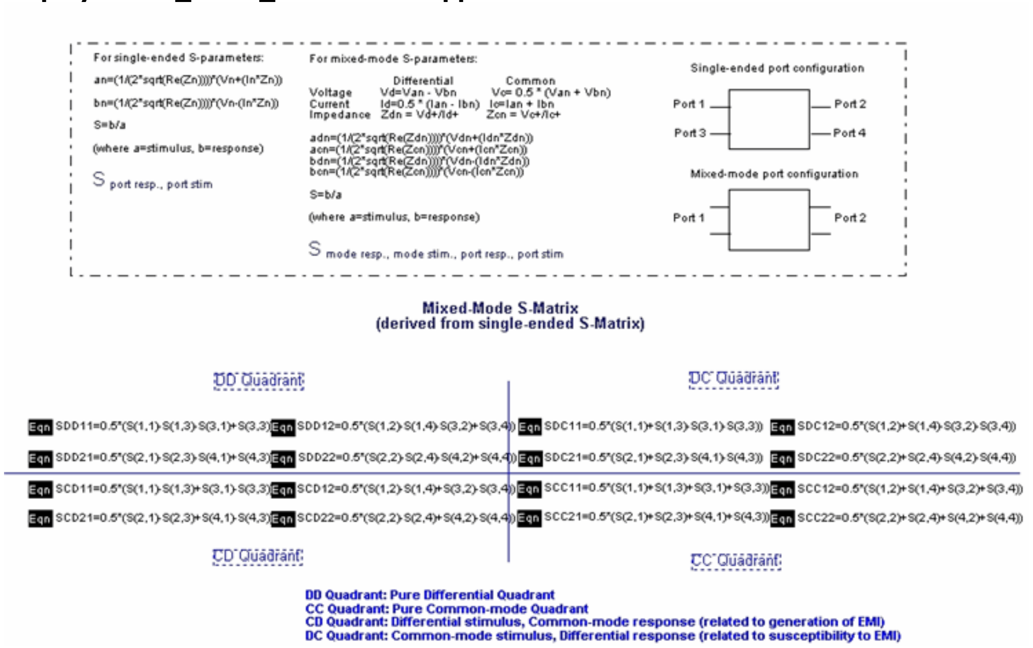
- **Mixed Mode S-Parameter Basics**
- **Mixed Mode Simulation Using Measured Data**
  - **Four Port Simulation [Measured Data]**

Here is a glimpse of some of the good reference examples available:

1.
   1. Example ADS simulation design showing how to create 4-port/2-port differential dataset from six 2-port single-ended datasets (Found in Agilent Signal Integrity Applicaitons Design Guide, under "Mixed Mode Simulation Using Measured Data"--Not included in downloadable ADS dataset + data display below)
      - **Click on the following picture icon to make it larger: Create_4port**


Creating a 4-port dataset from six 2-port datasets (or measured data)
Double-click on these DACs to browse for your datasets or measured S-parameter files (Touchstone, Citifile)
Meas1
S11, S12, S21, S22
Meas2
S14, S41, S44
Meas3
S13, S31, S33
Meas4
S34, S43
Meas5
S24, S42
Meas6
S32, S23
Define the differential impedance here
S-PARAMETERS
Define the frequency range here
Differential Port 1
Differential Port 2

2. **Approach #1: Mixed-mode ADS Data Display Equations**
   - **Click on the following picture icon to make it larger: ADS data display mixed_mode_basics.dds--Approach #1...**



For single-ended S-parameters:

an=(1/(2*sqrt(Re(Zn))))*(Vn+(In*Zn))

bn=(1/(2*sqrt(Re(Zn))))*(Vn-(In*Zn))

S=b/a

(where a=stimulus, b=response)

S port resp., port stim

For mixed-mode S-parameters:

	Differential	Common
Voltage	Vd=Van - Vbn	Vc= 0.5 * (Van + Vbn)
Current	Id=0.5 * (Ian - Ibn)	Ic=Ian + Ibn
Impedance	Zdn = Vd+/Id+	Zcn = Vc+/Ic+

adn=(1/(2*sqrt(Re(Zdn))))*(Vdn+(Idn*Zdn))
acn=(1/(2*sqrt(Re(Zcn))))*(Vcn+(Icn*Zcn))
bdn=(1/(2*sqrt(Re(Zdn))))*(Vdn-(Idn*Zdn))
bcn=(1/(2*sqrt(Re(Zcn))))*(Vcn-(Icn*Zcn))

S=b/a

(where a=stimulus, b=response)

S mode resp., mode stim., port resp., port stim

Single-ended port configuration
Port 1
Port 2
Port 3
Port 4
Mixed-mode port configuration
Port 1
Port 2

Mixed-Mode S-Matrix
(derived from single-ended S-Matrix)

DD Quadrant

DC Quadrant

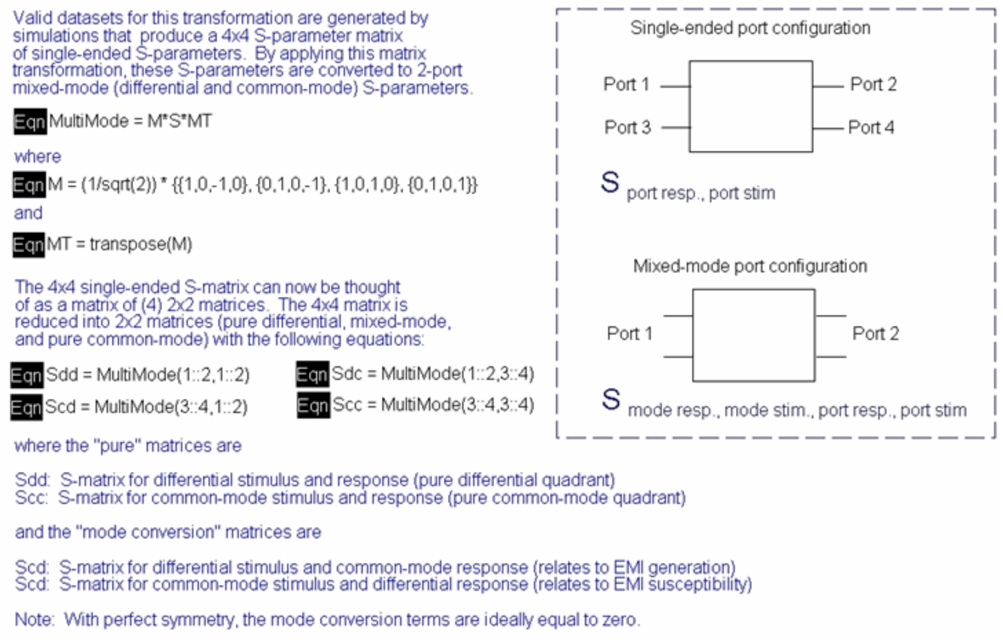
```
Eqn SDD11=0.5*(S(1,1)-S(1,3)-S(3,1)+S(3,3))  Eqn SDD12=0.5*(S(1,2)-S(1,4)-S(3,2)+S(3,4))  Eqn SDC11=0.5*(S(1,1)+S(1,3)-S(3,1)-S(3,3))  Eqn SDC12=0.5*(S(1,2)+S(1,4)-S(3,2)-S(3,4))
Eqn SDD21=0.5*(S(2,1)-S(2,3)-S(4,1)+S(4,3))  Eqn SDD22=0.5*(S(2,2)-S(2,4)-S(4,2)+S(4,4))  Eqn SDC21=0.5*(S(2,1)+S(2,3)-S(4,1)-S(4,3))  Eqn SDC22=0.5*(S(2,2)+S(2,4)-S(4,2)-S(4,4))
Eqn SCD11=0.5*(S(1,1)-S(1,3)+S(3,1)-S(3,3))  Eqn SCD12=0.5*(S(1,2)-S(1,4)+S(3,2)-S(3,4))  Eqn SCC11=0.5*(S(1,1)+S(1,3)+S(3,1)+S(3,3))  Eqn SCC12=0.5*(S(1,2)+S(1,4)+S(3,2)+S(3,4))
Eqn SCD21=0.5*(S(2,1)-S(2,3)+S(4,1)-S(4,3))  Eqn SCD22=0.5*(S(2,2)-S(2,4)+S(4,2)-S(4,4))  Eqn SCC21=0.5*(S(2,1)+S(2,3)+S(4,1)+S(4,3))  Eqn SCC22=0.5*(S(2,2)+S(2,4)+S(4,2)+S(4,4))
```

CD Quadrant

CC Quadrant

DD Quadrant: Pure Differential Quadrant
CC Quadrant: Pure Common-mode Quadrant
CD Quadrant: Differential stimulus, Common-mode response (related to generation of EMI)
DC Quadrant: Common-mode stimulus, Differential response (related to susceptibility to EMI)

3. **Approach #2: Alternative mixed-mode formulation:**
   - **Click on the following picture icon to make it larger: ADS data display mixed_mode_basics.dds--Approach #2...**


Single-ended port configuration
Port 1
Port 2
Port 3
Port 4
S port resp., port stim
Mixed-mode port configuration
Port 1
Port 2
S mode resp., mode stim., port resp., port stim

Valid datasets for this transformation are generated by simulations that produce a 4x4 S-parameter matrix of single-ended S-parameters. By applying this matrix transformation, these S-parameters are converted to 2-port mixed-mode (differential and common-mode) S-parameters.

Eqn MultiMode = M*S*MT

where

Eqn M = (1/sqrt(2)) * {{1,0,-1,0}, {0,1,0,-1}, {1,0,1,0}, {0,1,0,1}}

and

Eqn MT = transpose(M)

The 4x4 single-ended S-matrix can now be thought of as a matrix of (4) 2x2 matrices. The 4x4 matrix is reduced into 2x2 matrices (pure differential, mixed-mode, and pure common-mode) with the following equations:

Eqn Sdd = MultiMode(1::2,1::2)
Eqn Sdc = MultiMode(1::2,3::4)
Eqn Scd = MultiMode(3::4,1::2)
Eqn Scc = MultiMode(3::4,3::4)

where the "pure" matrices are

Sdd: S-matrix for differential stimulus and response (pure differential quadrant)
Scc: S-matrix for common-mode stimulus and response (pure common-mode quadrant)

and the "mode conversion" matrices are

Scd: S-matrix for differential stimulus and common-mode response (relates to EMI generation)
Scd: S-matrix for common-mode stimulus and differential response (relates to EMI susceptibility)

Note: With perfect symmetry, the mode conversion terms are ideally equal to zero.

Meta Description

DocID:168856
DocOwner:TOSU
DocCustViewable:Yes
Keywords:
DocVersion:ADS-All
DocPlatform:All

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Optimization or Yield Analysis Goal (Sloped or Curved Line).

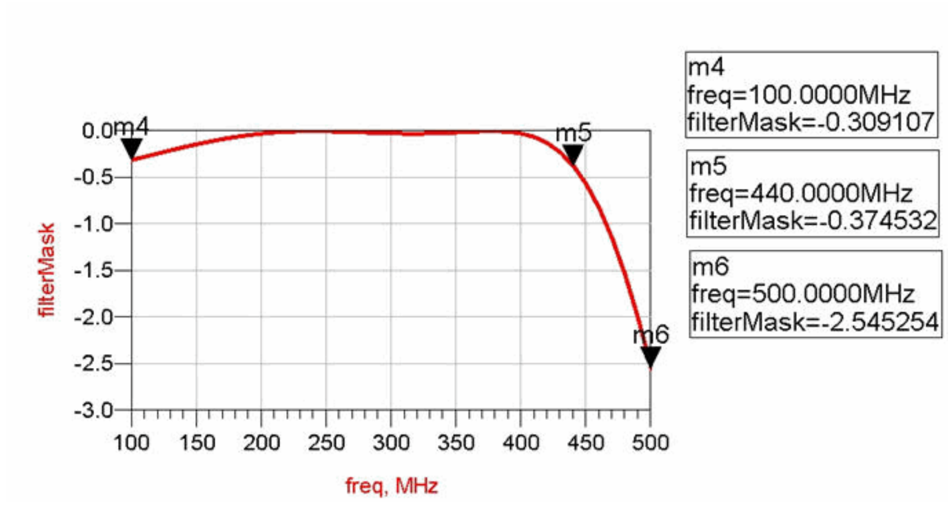
ADS Optimization/Yield: How to write optimization or yield analysis goal statements to specify a sloped or curved line (Ex: filter mask)?

Description

Example Problem/Question:

Q: How to write optimization or yield analysis goal statements to specify a sloped or curved line (Ex: a filter mask)?

- For example: you want to write ADS Optmization (or Yield) goal/spec statements to match the filter mask illustrated below (note that the final fitler curve was the result of ADS examples/ Tutorial/ optex1_wrk/optex2 schematic):

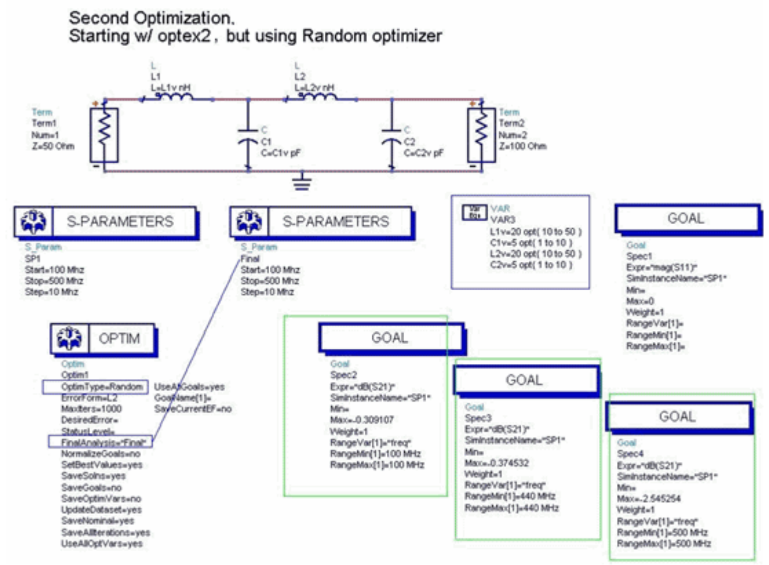


Solution Approach to Consider:

You could try to create a set of sophisticated measurement equations to map your desired filter mask target for your goal expression. However, we have found that a more intuitive and efficient method is to do the following:

- **Instead of trying to use one single target expression, place multiple ADS Goal (Optimization) or Yield Specifications (Yield) statements, each at single strategic points or at narrow frequency bands with in your target curve.**
    - Using the above filter curve as an example, my first optimization goal could be S21 = -0.309107 at initial frequency of 100 MHz in the curve of interest; the second optimization goal could be S21 = -0.374532 at mid-point frequency 440 MHz in the target curve; the third optimization goal could be S21 = 2.545254 dB within the highest frequency band of the target curve.
- **Here is a brief summary of how the "optex2_filterS21" was set up to optimize our simple filter design against the above filter curve:**

1. Based upon the strategic points in our target filter curve (beginning, middle, end), I added the following 3 target S21 goals as shown below to the original ADS examples/ Tutorital/ optex1_wrk/ optex2 schematic. This design will optimize the "L1v", C1v", "L2v", "C2v" variables in the design against the single target frequency goals that we set based upon the original filter mask from above.
2. For best results with this circuit, I chose the "Random" Optim "OptimType" parameter. You may need to modify the optimization algorithm to best fit your application requirement.
3. To save the best optimized result for comparison, I used the Optim "FinalAnalysis" feature to have ADS run one more S-parameter analysis called "Final" after optimization is complete.

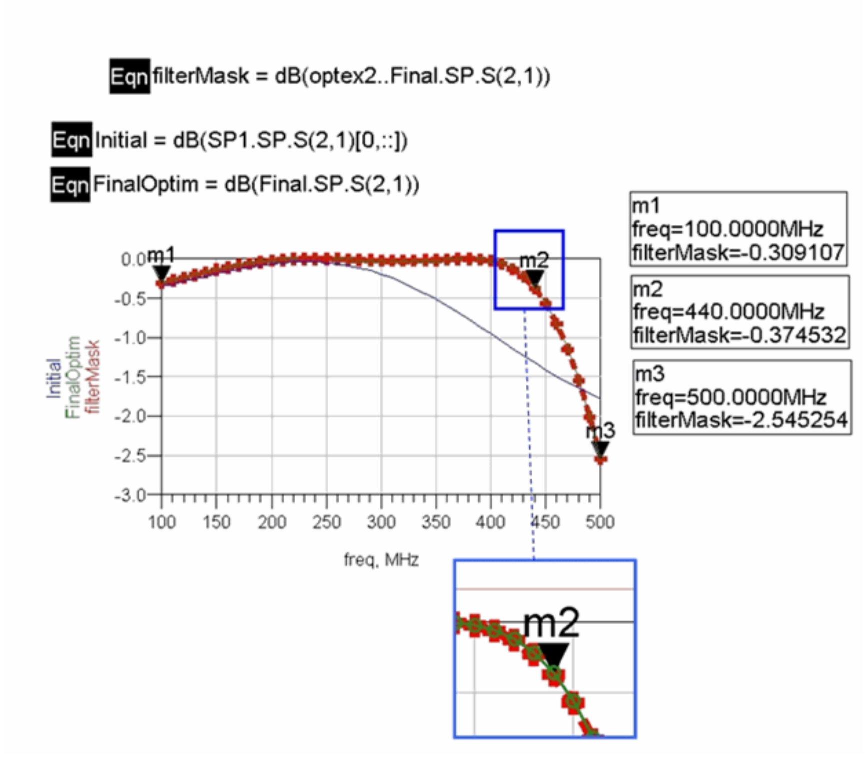

Second Optimization.
Starting w/ optex2 , but using Random optimizer

```
S_Param
SP1
Start=100 Mhz
Stop=500 Mhz
Step=10 Mhz

S_Param
Final
Start=100 Mhz
Stop=500 Mhz
Step=10 Mhz

VAR
VAR3
L1v=20 opt( 10 to 50 )
C1v=5 opt( 1 to 10 )
L2v=20 opt( 10 to 50 )
C2v=5 opt( 1 to 10 )

Goal
Spec1
Expr="mag(S11)"
SimInstanceName="SP1"
Min=
Max=0
Weight=1
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=

Optim
Optim1
OptimType=Random        UseAllGoals=yes
ErrorForm=L2            GoalName[1]=
MaxIters=1000           SaveCurrentEF=no
DesiredError=
StatusLevel=
FinalAnalysis="Final"
NormalizeGoals=no
SetBestValues=yes
SaveSolns=yes
SaveGoals=no
SaveOptimVars=no
UpdateDataset=yes
SaveNominal=yes
SaveAllIterations=yes
UseAllOptVars=yes

Goal
Spec2
Expr="dB(S21)"
SimInstanceName="SP1"
Min=
Max=-0.309107
Weight=1
RangeVar[1]="freq"
RangeMin[1]=100 MHz
RangeMax[1]=100 MHz

Goal
Spec3
Expr="dB(S21)"
SimInstanceName="SP1"
Min=
Max=-0.374532
Weight=1
RangeVar[1]="freq"
RangeMin[1]=440 MHz
RangeMax[1]=440 MHz

Goal
Spec4
Expr="dB(S21)"
SimInstanceName="SP1"
Min=
Max=-2.545254
Weight=1
RangeVar[1]="freq"
RangeMin[1]=500 MHz
RangeMax[1]=500 MHz
```

4. If we look at our final results in the corresponding ADS data display, we see the

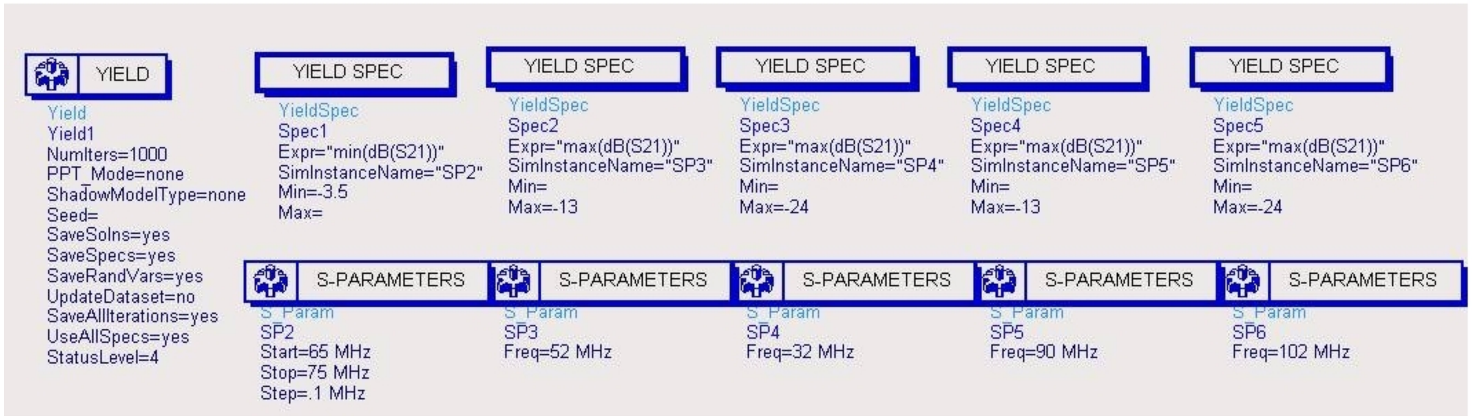
following:

- The **blue "Initial" trace** is the resulting simulated filter trace using the intial values of the "L1v", C1v", "L2v", "C2v" variables as shown in design above (20,5,20,5, respectively).
- The **red cross line "filterMask" trace** is our target filter curve from above.
- The **green dotted line "FinalOptim" trace** is our final filter curve model after applying above ADS Optimization scheme.



Alternative Approaches to Consider:

This same approach can also be applied to Yield/Sensitivity Analysis when you set up your appropriate set of "Yield Spec" goal target expressions. Here is an example of how to set up an ADS Yield Sensitivity simulation against a target filter curve. Notice that instead of using the RangeVar / RangeMin/ RangeMax approach within a target "freq" frequency band setup that we used in the above Optimization/single S-parameter controller example, we could also use multiple ADS S-parameter simulation controllers (SP2, SP3, SP4, etc...) with a specific target frequency or narrow band of frequencies that we want a certain yield specification goal to apply towards. Using multiple S-parameter simulation controllers in this manner is advantageous if we want to apply some post-processing data templates (Ex: component yield/histogram sensitivity analysis) which only takes data at a single frequency of interest.


YIELD
Yield
Yield1
NumIters=1000
PPT_Mode=none
ShadowModelType=none
Seed=
SaveSolns=yes
SaveSpecs=yes
SaveRandVars=yes
UpdateDataset=no
SaveAllIterations=yes
UseAllSpecs=yes
StatusLevel=4

YIELD SPEC
YieldSpec
Spec1
Expr="min(dB(S21))"
SimInstanceName="SP2"
Min=-3.5
Max=

YIELD SPEC
YieldSpec
Spec2
Expr="max(dB(S21))"
SimInstanceName="SP3"
Min=
Max=-13

YIELD SPEC
YieldSpec
Spec3
Expr="max(dB(S21))"
SimInstanceName="SP4"
Min=
Max=-24

YIELD SPEC
YieldSpec
Spec4
Expr="max(dB(S21))"
SimInstanceName="SP5"
Min=
Max=-13

YIELD SPEC
YieldSpec
Spec5
Expr="max(dB(S21))"
SimInstanceName="SP6"
Min=
Max=-24

S-PARAMETERS
S_Param
SP2
Start=65 MHz
Stop=75 MHz
Step=.1 MHz

S-PARAMETERS
S_Param
SP3
Freq=52 MHz

S-PARAMETERS
S_Param
SP4
Freq=32 MHz

S-PARAMETERS
S_Param
SP5
Freq=90 MHz

S-PARAMETERS
S_Param
SP6
Freq=102 MHz

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Optimizing a Nonlinear Capacitor Model.

Optimizing a Nonlinear Model of a Capacitor to Match Bias-Dependent S-Parameter Data

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/NonlinearC_L_Modeling_wrk

Description

This example shows how to optimize a nonlinear model of a capacitor to match bias-dependent S-parameter data. The same technique could be used to generate some other nonlinear model, such as a nonlinear inductor.

A simple nonlinear inductor model, implemented with an SDD (Symbolically-Defined Device) is included.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at PCI Express Examples Workshop

PCI Express Examples/Workshop

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/PCI_Express/PCIE.DEB

Description

The PCI Express workshop provides the PCI express related simulation examples for ADS2011 in a design guide form. The .deb file can be installed using ADS main window>DesignGuide> Add DesignGuide menu

The installation allows both global and personal installation.

Once installed it will appear under DesignGuide window of your RF/Analog Schematic page.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at PLL Example with an Active Filter.

PLL Example with an Active Filter

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/pll_testR1_wrk

Description

This example contains one design: PLL_tranAH shows the transient response simulation of a simple, behavioral PLL with an active filter. After a brief delay at the start, the divide ratio is stepped and the transient response of the loop is shown.

The reference and divided VCO signals (phases) are in phase at the input to the phase/frequency detector, and vtune is stable, at 0 V at time=0, so there is no turn-on transient.

There are other PLLs with active filters in the PLL DesignGuide.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Save Simulation Data to ASCII.

Save Simulation Data to ASCII File Directly (writepara)

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/writepara_wrk

Description

These AEL expressions export the passed data to a text file.

You can write a Measurement Equation on a schematic window and pass the parameters that you wish to export. For example:

```
X = writepara2d("c:/temp/data.txt", "W", "My Simulation Data", ",", freq,
db(S21))
```

This was developed as a way to export data to a text file in ADS 1.1. ADS 1.3 solved that problem by providing a way to export listings in the Data Display. However, there are other types of applications for this script. In swept optimizations, for example, you can use this script to write the best values out to a text file after each parameter iteration. You can also use it to export the simulation data to a third party tool and bypass the ADS Data Display entirely.

The workspace writepara.7zap contains several examples on how to use the writepara() functions.

Note
In ADS 2003C and later you can use the built-in function *write_snp()* to write Touchstone files from S-parameter data. In ADS 2004A and later there is a built-in function called *write_var()* which is similar to writepara2d().

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Simple Sourcepull and Loadpull Explained

Simple Sourcepull and Loadpull Explained

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/Loadpull/LOADNSOURCEPULL_wrk

Description

This example workspace provides an alternative to the source pull and load pull examples in ADS. The set up is straightforward and allows users to familiarize themselves with this concept in ADS.

Updated 08-20-2003

See Also

Loadpull DesignGuide for ADS 2009 Update 1 with new content (This is the most advanced Load Pull DesignGuide for use with ADS 2009 Update 1, but it does not have the enhancements of the ADS 2011 version.)

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

See also: *Loadpull Simulation with a Modulated Source* (examples).

Note
Example prior to ADS2011 can be located at SI Primer part 1 of 2 Convolution.

ADS [Signal Integrity Primer 1/2]: Basic Principles of Convolution

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/ConvolutionBasics_2006U1_wrk

Description

Q: Where can I find some basic ADS examples for analyzing convolved signals?

For the most comprehensive walk-through, hands-on labs we recommend you consider taking one of the following Signal Integrity training courses offered by the Agilent EEsof Customer Education team:

- N3215A: Designing for Signal Integrity with Advanced Design System
- N3216A: Designing for Signal Integrity with Advanced Design System 2

The following excerpts provide an introduction to some of the deeper analysis tools that you can master quicker by taking an EEsof training course such as N3215A.

What Is Convolution?

- Time-domain simulation – requires time-domain component characterization
- Many components – frequency-domain characterization
- Convolution engine performs the following tasks:
    - Detects components defined in the frequency-domain
    - Calculates impulse response from frequency response
    - Convolves input signal with impulse response (time-domain process)
- All these – as part of transient simulation, transparent to the user

Time-to/from-Frequency Conversions

Time-to-frequency (time-waveform to spectrum)*

- the fs() function
    - implements a Fourier Transform (like a spectrum analyzer)
        - Eqn Vspectrum=fs(Vtime)

Frequency-to-time (spectrum to time-waveform)*

- the ts() function
    - Implements an Inverse Fourier Transform
        - Eqn Vtime=ts(Vspectrum)

Time-Step versus Span

- Time-step small enough to capture all high-frequency variations
- Span large enough to capture all frequency-domain energy
- Maximum time-step (Nyquist): MaxTimeStep = 1/(2*Span)
- Recommended: Timestep <= 1/(5*Span)

Time-Length versus RBW (Resolution Bandwidth)

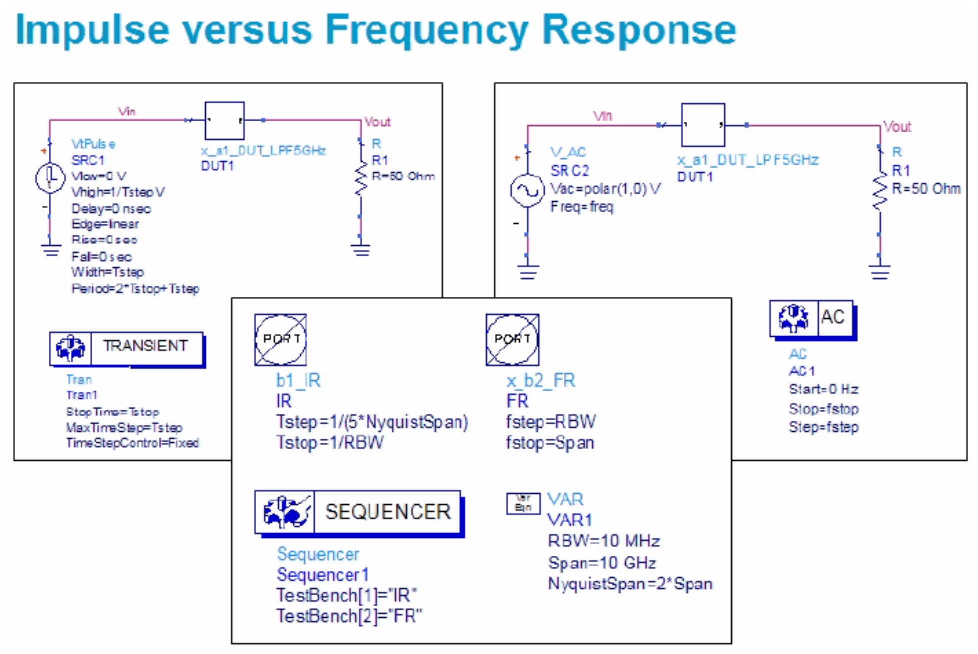
- Time-length long enough to ensure IR goes to 0
- RBW small enough to capture all relevant spectral components
- TimeLength = 1/RBW

Creating an S-Parameter Model for Convolution

- From simulations or from measurement (network-analyzer)
- Set span large enough to capture highest freq component
- Set RBW small enough to capture variations of freq response
- Save the S-parameters in a file
- Add/review the DC point in the S-parameter file

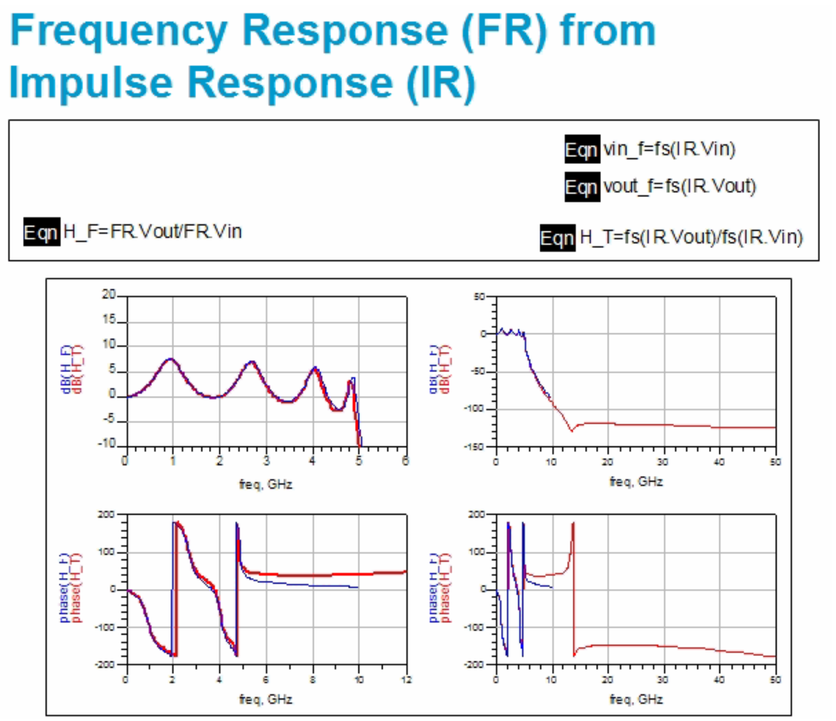
Impulse versus Frequency Response

The hierarchical design shown on this slide is used to compare the impulse response and the frequency response of the filter when AC simulation is used.



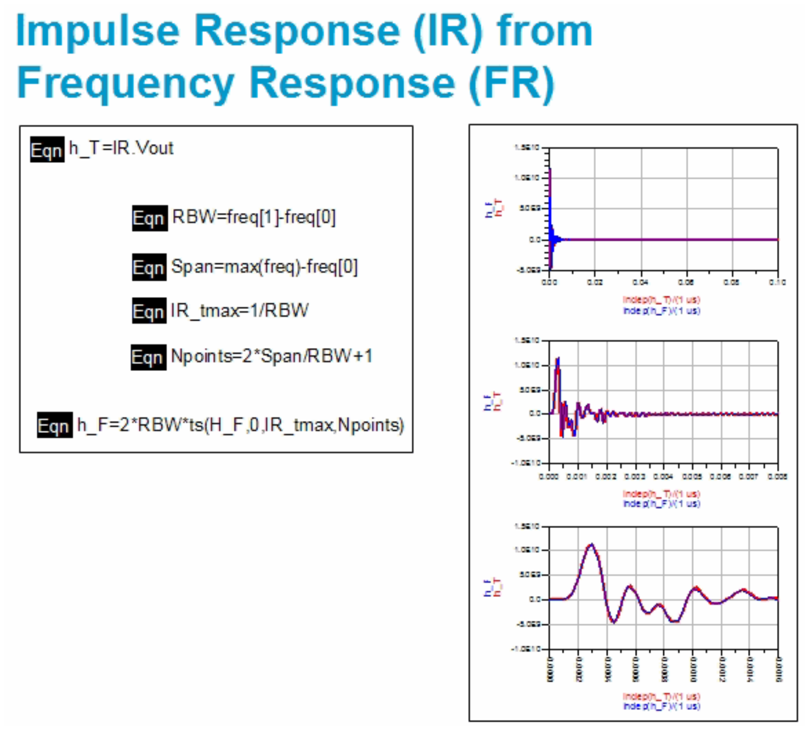
Frequency Response (FR) from Impulse Response (IR)

The frequency domain transfer function is calculated from the impulse response and then compared to the transfer function determined by the AC simulation.



Impulse Response (IR) from Frequency Response (FR)

Impulse response is calculated from the frequency domain transfer function and then compared to the impulse response determined by the transient simulation.



Impulse Response (IR) from Frequency Response (FR)

Eqn h_T=IR.Vout

Eqn RBW=freq[1]-freq[0]

Eqn Span=max(freq)-freq[0]

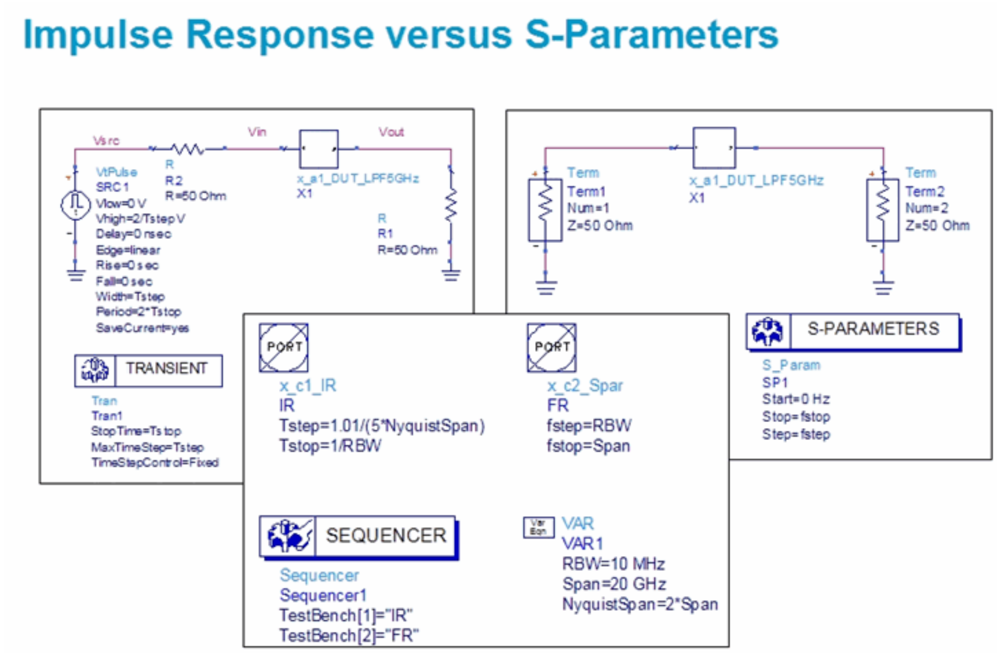
Eqn IR_tmax=1/RBW

Eqn Npoints=2*Span/RBW+1

Eqn h_F=2*RBW*ts(H_F,0,IR_tmax,Npoints)

Impulse Response versus S-Parameters:

- S21 from Transmission-IR
- Transmission-IR from S21
- S11 from Reflection-IR
- Reflection-IR from S11



Impulse Response versus S-Parameters

VtPulse
SRC1
Vlow=0 V
Vhigh=2/Tstep V
Delay=0 nsec
Edge=linear
Rise=0 sec
Fall=0 sec
Width=Tstep
Period=2*Tstop
SaveCurrent=yes

R
R2
R=50 Ohm

x_a1_DUT_LPF5GHz
X1

R
R1
R=50 Ohm

Tran
Tran1
StopTime=Tstop
MaxTimeStep=Tstep
TimeStepControl=Fixed

Term
Term1
Num=1
Z=50 Ohm

x_a1_DUT_LPF5GHz
X1

Term
Term2
Num=2
Z=50 Ohm

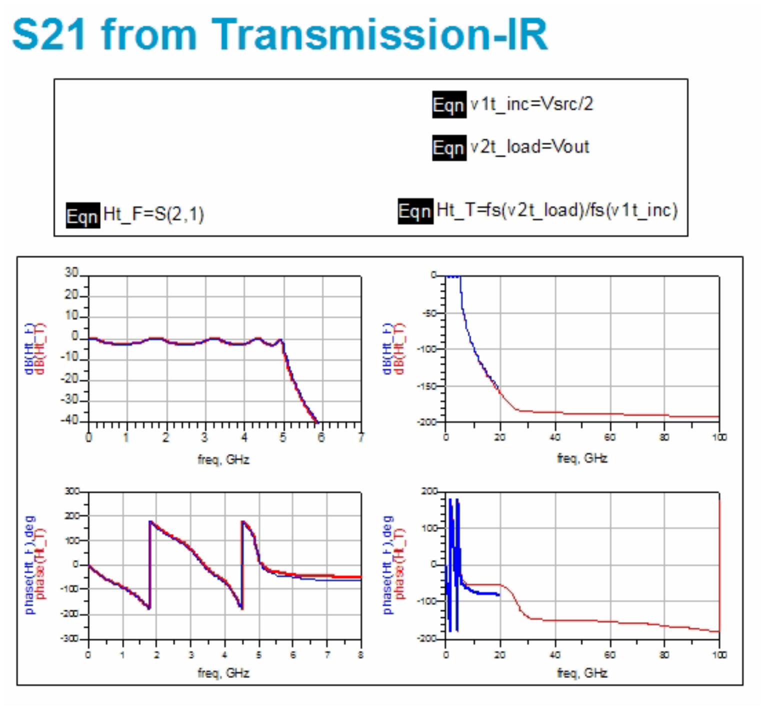
S_Param
SP1
Start=0 Hz
Stop=fstop
Step=fstep

x_c1_IR
IR
Tstep=1.01/(5*NyquistSpan)
Tstop=1/RBW

x_c2_Spar
FR
fstep=RBW
fstop=Span

Sequencer
Sequencer1
TestBench[1]="IR"
TestBench[2]="FR"

VAR
VAR1
RBW=10 MHz
Span=20 GHz
NyquistSpan=2*Span

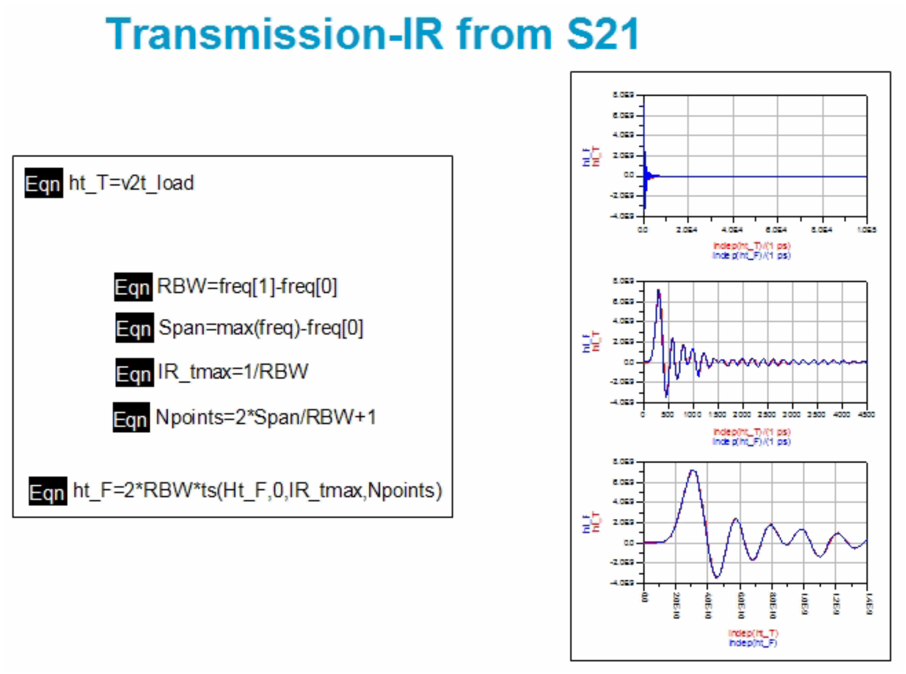

S21 from Transmission-IR

Eqn v1t_inc=Vsrc/2

Eqn v2t_load=Vout

Eqn Ht_F=S(2,1)

Eqn Ht_T=fs(v2t_load)/fs(v1t_inc)


Transmission-IR from S21

Eqn ht_T=v2t_load

Eqn RBW=freq[1]-freq[0]

Eqn Span=max(freq)-freq[0]

Eqn IR_tmax=1/RBW

Eqn Npoints=2*Span/RBW+1

Eqn ht_F=2*RBW*ts(Ht_F,0,IR_tmax,Npoints)

See Also

- SI Primer 2 of 2 TDR and TDT
- connection manager ts frequencytime conversion working with importing timedomain or tdr measured data from network analyzer
- Comparing measured TDR vs measured s-parameter in time domain Set-up assumptions tdr_sp_gamma tdr_sp_imped tdr_step_imped
- How to get PLTS plots from ADS
- transforming sparameter data into tdr time domain representation
- what simplest way get impedance results from sparm sim im re
- display bandlimited filtered or rise time limited tdr response using sparameters
- Import TDR measurements with PNA
- how convolution computes an impulse response harmonic balance transient or envelope simulations
- Differential TDR Comparison using SP TDR FrontPanel and Transient

Meta Description

DocID:316598
DocOwner:MKNUTSON
DocCustViewable:Yes
Keywords:Basic Principles of Convolution Signal Integrity SI
DocVersion:ADS2008,ADS-All
DocPlatform:All <#comment>

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at SI Primer 2 of 2 TDR and TDT.

ADS [Signal Integrity Primer 2 of 2]: TDR/TDT Simulations and Measurements

Location (in ADS2011+): $HPEESOF_DIR/examples/KC_Examples/TDR_2006U1_wrk

Description

Q: Where can I find some basic ADS examples for generating and analyzing TDR(Time-Domain Reflectometry)/TDT(Time-Domain Transmission) stimulus, response?

For the most comprehensive walk-through, hands-on labs we recommend you consider taking one of the following Signal Integrity training courses offered by the Agilent EEsof Customer Education team:

- N3215A: Designing for Signal Integrity with Advanced Design System
- N3216A: Designing for Signal Integrity with Advanced Design System 2

The following excerpts provide a glimpse of some of the deeper analysis tools that you can master faster by taking an EEsof training course such as N3215A.

Delayed Reflections

Both the input and output voltage-waveforms show a staircase shape. This is caused by the reflections on various discontinuities in the circuit, combined with the time necessary for these reflections to propagate through the system.

- The observation of such effects at the input of the system is usually referred to as TDR, or Time-Domain Reflectometer.
- The observation of such effects at the output of the system is usually referred to as TDT, or Time-Domain Transmission.

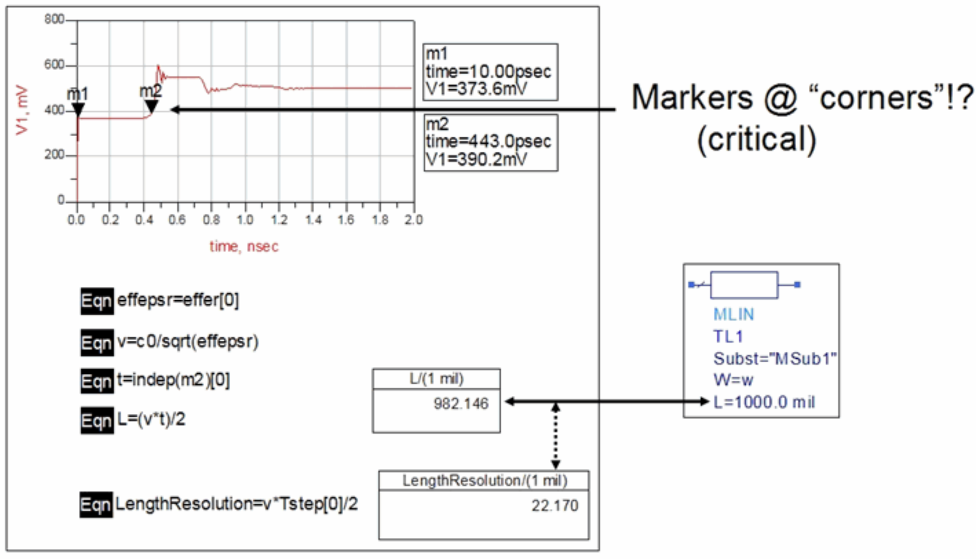
TDR Simulations and Measurements

- Detect discontinuities
- Locate discontinuities = distance from input
- Quantify discontinuities = calculate Z0 after discontinuity, when Z0 before discontinuity is known

TDR - Locate/Quantify Discontinuities

- Measure time-delay
- Known propagation-speed (?)
- Propagation speed:
- Conventional structures; known formulas for prop.-speed (theoretical calculations)
- Un-conventional structures; calibration (simulations / measurements)

TDR - Locate Discontinuity - Results



Eqn effepsr=effer[0]

Eqn v=c0/sqrt(effepsr)

Eqn t=indep(m2)[0]

Eqn L=(v*t)/2

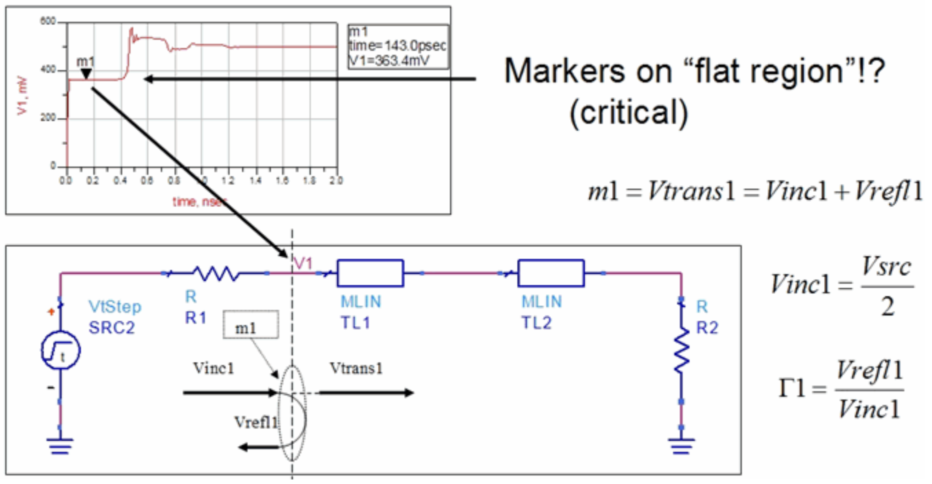
Eqn LengthResolution=v*Tstep[0]/2

L/(1 mil)
982.146

LengthResolution/(1 mil)
22.170

TDR – Quantify 1st Discontinuity

Calculate Γ_1

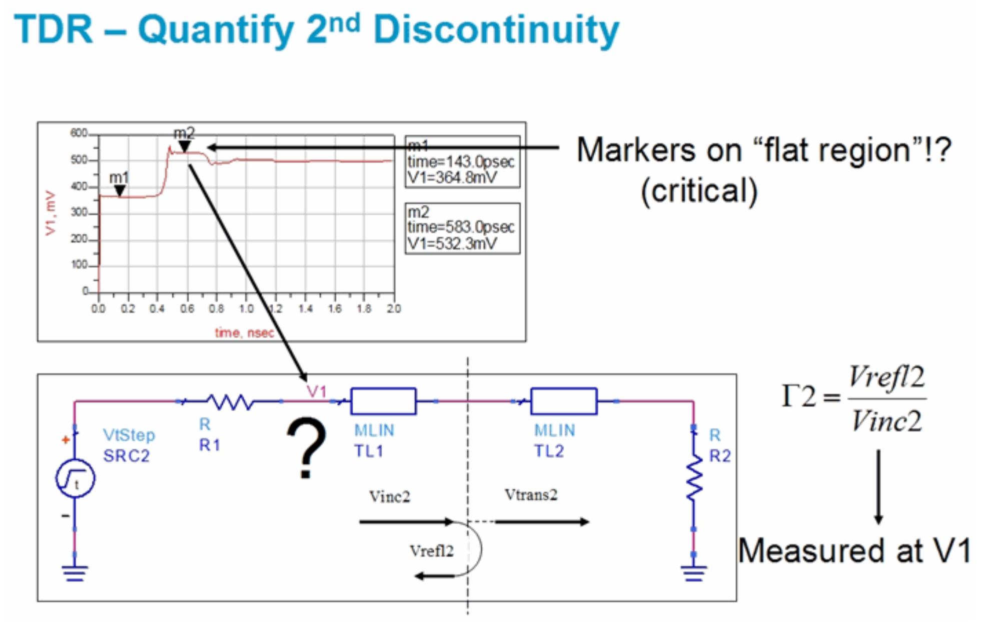


$$m1 = Vtrans1 = Vinc1 + Vrefl1$$

$$Vinc1 = \frac{Vsrc}{2}$$

$$\Gamma 1 = \frac{Vrefl1}{Vinc1}$$

- vinc1 = Vsrc[0]/2
- vrefl1 = m1[0]-vinc1
- Gamma1 = vrefl1/vinc1
- So, if Gamma = (Z01 - Zsrc)/(Z01 + Zsrc), Z01 = Zsrc*(1+Gamma1)/(1-Gamma1)...
- Z01 = Zref[0]*(1+Gamma1)/(1-Gamma1)

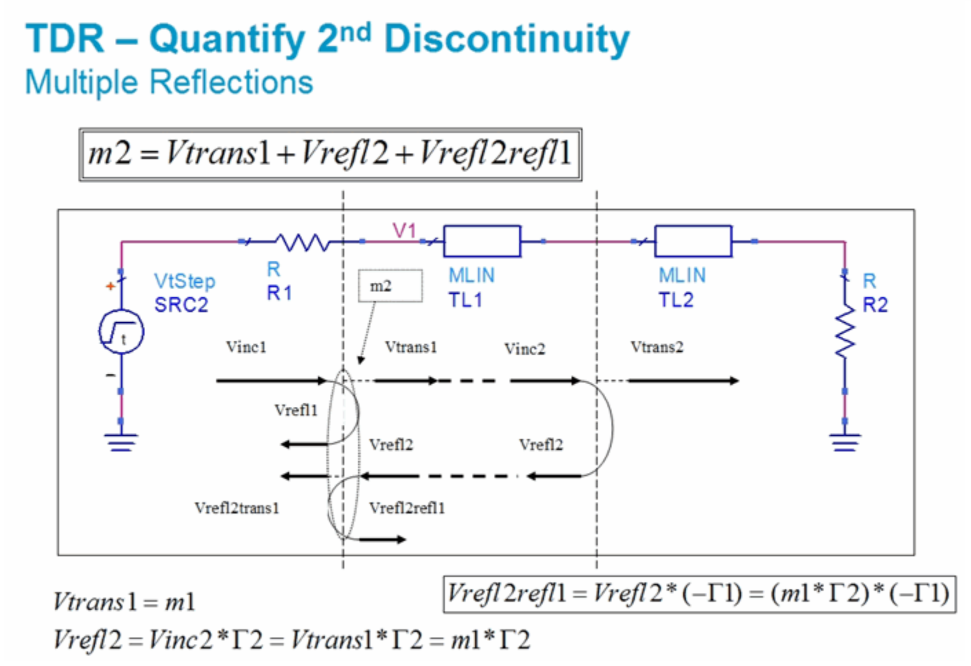


TDR – Quantify 2nd Discontinuity

Markers on "flat region"!? (critical)

$$\Gamma 2 = \frac{Vrefl2}{Vinc2}$$

Measured at V1



TDR – Quantify 2nd Discontinuity

Multiple Reflections

$$\boxed{m2 = Vtrans1 + Vrefl2 + Vrefl2refl1}$$

$$Vtrans1 = m1$$

$$Vrefl2 = Vinc2 * \Gamma 2 = Vtrans1 * \Gamma 2 = m1 * \Gamma 2$$

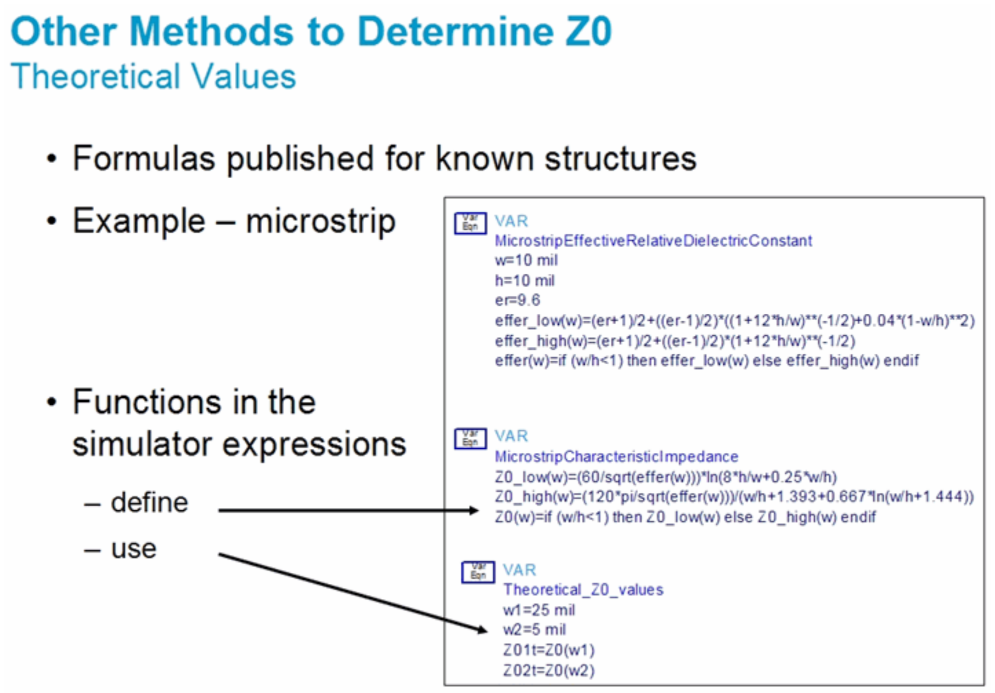
$$\boxed{Vrefl2refl1 = Vrefl2 * (-\Gamma 1) = (m1 * \Gamma 2) * (-\Gamma 1)}$$

- m2 = Vtrans1+Vrefl2+Vrefl2refl1
- Vtrans1=m1
- Vrefl2 = Vinc2*Gamma2 = Vtrans1*Gamma2 = m1*Gamma2
- Vrefl2refl1 = Vrefl2*(-Gamma1) = (m1*Gamma2)*(-Gamma1)
- Gamma2 = (m2-m1)/(m1*(1-Gamma1))
- Gamma2 = (m2[0]-m1[0])/(m1[0]*(1-Gamma1))

Gamma2 - Between the two transmission lines

- Gamma2 = (Z02-Z01)/(Z02+Z01), then Z02 = Z01*(1+Gamma2)/(1-Gamma2)
- Z02 = Z01*(1+Gamma2)/(1-Gamma2)

Other Methods to Determine Z0 Theoretical Values

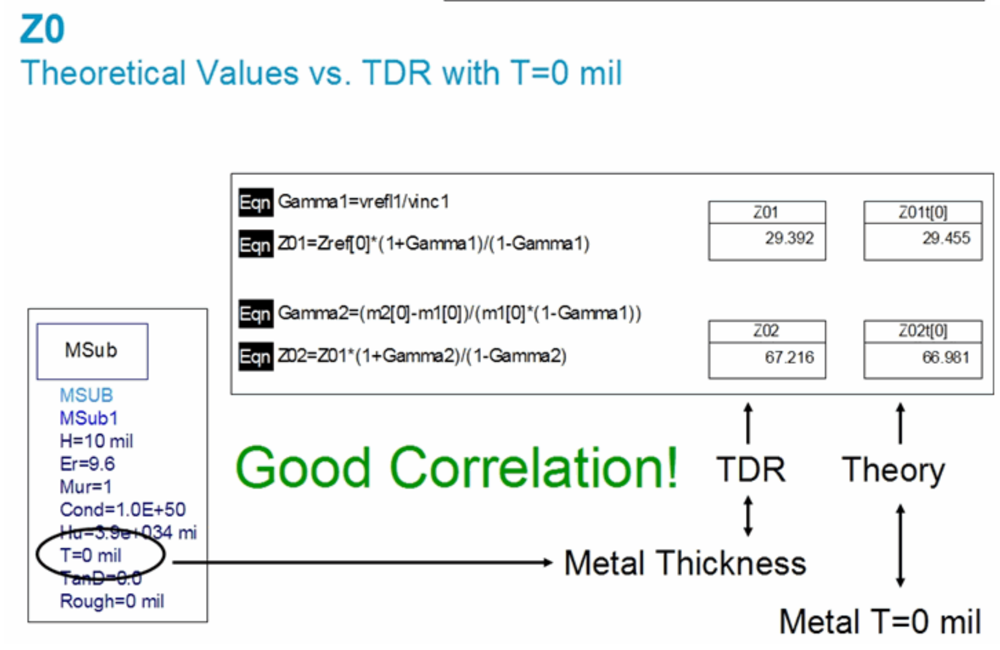


Other Methods to Determine Z0

Theoretical Values

- Formulas published for known structures
- Example – microstrip
- Functions in the simulator expressions
  - define
  - use

```
VAR
MicrostripEffectiveRelativeDielectricConstant
w=10 mil
h=10 mil
er=9.6
effer_low(w)=(er+1)/2+((er-1)/2)*((1+12*h/w)**(-1/2)+0.04*(1-w/h)**2)
effer_high(w)=(er+1)/2+((er-1)/2)*(1+12*h/w)**(-1/2)
effer(w)=if (w/h<1) then effer_low(w) else effer_high(w) endif

VAR
MicrostripCharacteristicImpedance
Z0_low(w)=(60/sqrt(effer(w)))*ln(8*h/w+0.25*w/h)
Z0_high(w)=(120*pi/sqrt(effer(w)))/(w/h+1.393+0.667*ln(w/h+1.444))
Z0(w)=if (w/h<1) then Z0_low(w) else Z0_high(w) endif

VAR
Theoretical_Z0_values
w1=25 mil
w2=5 mil
Z01t=Z0(w1)
Z02t=Z0(w2)
```



Z0

Theoretical Values vs. TDR with T=0 mil

```
Eqn Gamma1=vrefl1/vinc1
Eqn Z01=Zref[0]*(1+Gamma1)/(1-Gamma1)
Eqn Gamma2=(m2[0]-m1[0])/(m1[0]*(1-Gamma1))
Eqn Z02=Z01*(1+Gamma2)/(1-Gamma2)
```

Z01	Z01t[0]
29.392	29.455

Z02	Z02t[0]
67.216	66.981

```
MSub
MSUB
MSub1
H=10 mil
Er=9.6
Mur=1
Cond=1.0E+50
Hu=3.9e+034 mi
T=0 mil
TanD=0.0
Rough=0 mil
```

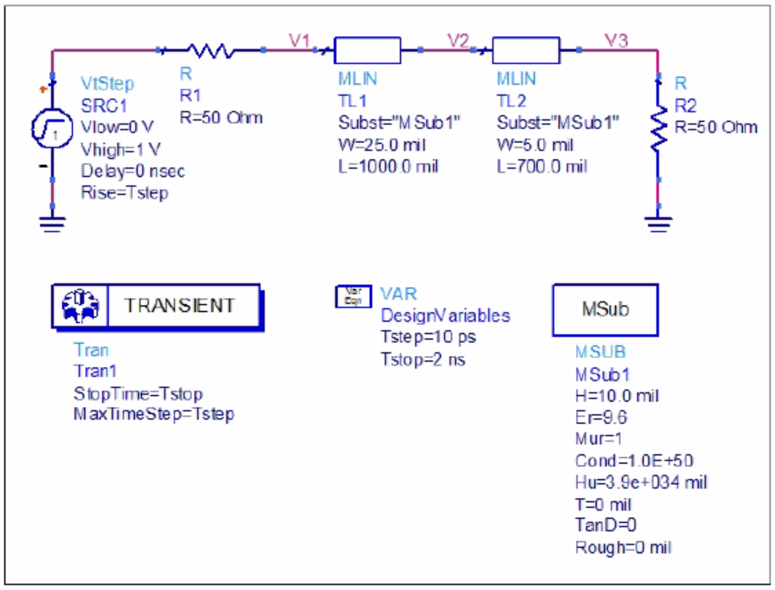
Also, try LineCalc ...

TDT – Time Domain Transmission

- Various transmission measurements(impulse response, step response)
- Also used for measurements on unconventional structures:
  - Measure time-delay
  - Known length – determine wave propagation speed
- Wave speed – locate discontinuities using TDR (on similar structures of unknown lengths)

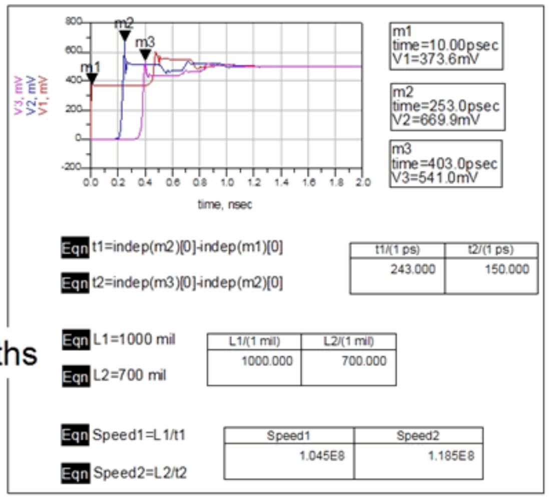
TDT

Measuring Time-Delay and Wave Speed



TDT – Results

Measuring Time-Delay and Wave Speed



Delays (Measured)

Eqn t1=indep(m2)[0]-indep(m1)[0]

Eqn t2=indep(m3)[0]-indep(m2)[0]

t1/(1 ps)	t2/(1 ps)
243.000	150.000

Known Lengths

Eqn L1=1000 mil

Eqn L2=700 mil

L1/(1 mil)	L2/(1 mil)
1000.000	700.000

Wave-Speed (Calculated)

Eqn Speed1=L1/t1

Eqn Speed2=L2/t2

Speed1	Speed2
1.045E8	1.185E8

See Also

- Reference: Agilent Time Domain Analysis Using a Network Analyzer (AN 1287-12)
- Comparing measured TDR vs measured s-parameter in time domain Set-up assumptions tdr_sp_gamma tdr_sp_imped tdr_step_imped
- SI Primer part 1 of 2 Convolution
- How to get PLTS plots from ADS
- connection manager ts frequencytime conversion working with importing timedomain or tdr measured data from network analyzer
- transforming sparameter data into tdr time domain representation
- what simplest way get impedance results from sparm sim im re
- display bandlimited filtered or rise time limited tdr response using sparameters
- how convolution computes an impulse response harmonic balance transient or envelope simulations
- Import TDR measurements with PNA
- Differential TDR Comparison using SP TDR FrontPanel and Transient

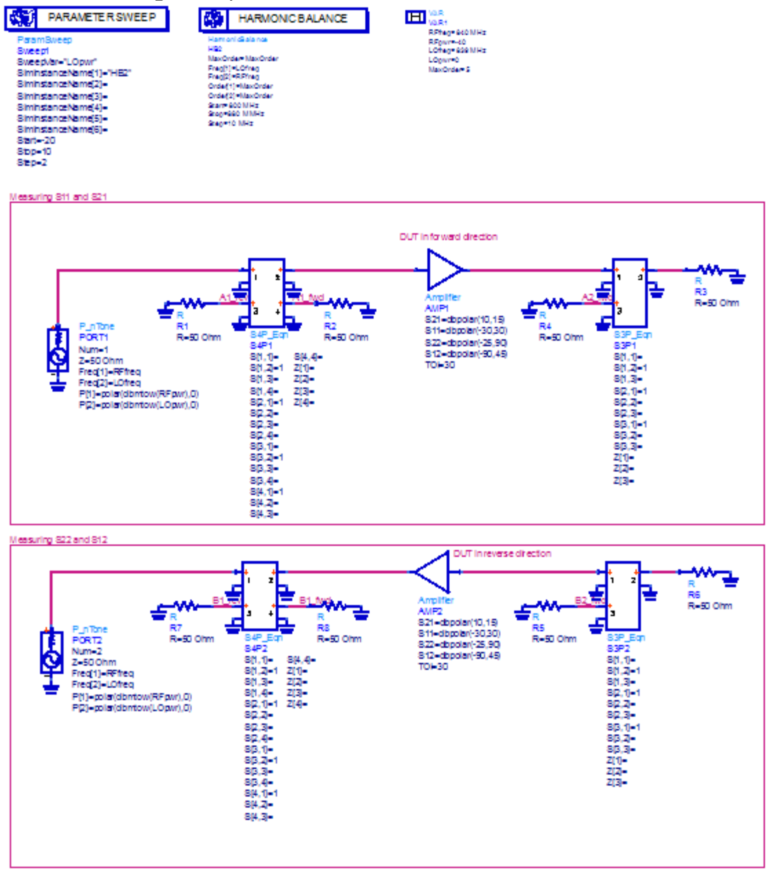
Note
Example prior to ADS2011 can be located at S-Parameter Simulations Using HB.

S-Parameter Simulations Using HB

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/HB_SP_Stab_wrk

Description

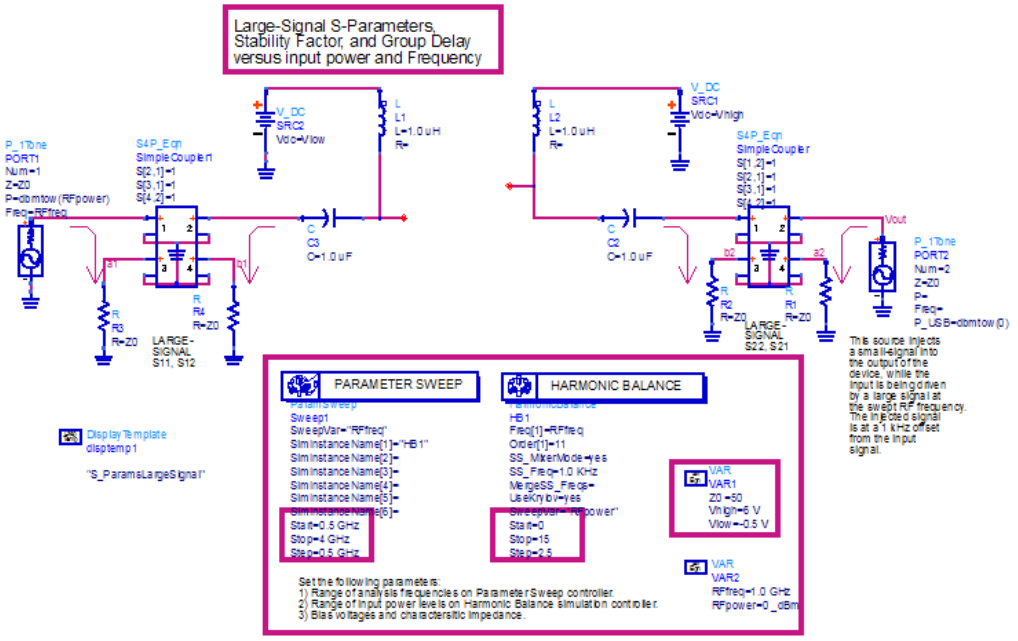
- **I. Contents of the example workspace:**
  - This workspace shows how HB can be used to calculate S-parameters in various ways. ** The following test cases are included in the workspace:
    1. **HPSparam: One large signal test carrier (Single freq)**
    2. **HPSparam_2: One large signal test carrier, with frequency sweep**
    3. **HPSparam_3: One large signal carrier plus one small signal test carrier. The small signal test carrier is swept over frequency.**
    4. **HPSparam_4: Same as the case above but adding a power sweep of the large signal carrier.**
- **II. Options available in ADS to simulate S-parameters from large signal harmonic balance simulations.**
  - Consider the following examples to find the one that works best for your application:
    1. **Example below uses Harmonic Balance with SNP_Eqn Component(Couplers) with DUT in forward and reverse topology.**
       - This example uses SNP_Eqn Components as virtual couplers in Harmonic Balance Simulation to give large signal s-parameters. This approach has high reverse-isolation device under test DUT (Ex: amplifer) in the schematic twice to allow for the computation of both the forward and reverse direction s-parameters:
         - Here is picture of the basic HPSparam set-up that uses two S4P_Eqn components as virtual couplers in a Harmonic Balance simulation to give s-parameters:



    2. **Use ADS LSSP Controller Instead of Harmonic Balance.**
       - Another alternative would be to use the ADS "LSSP" controller instead of Harmonic Balance to automatically generate large-signal s-parameter results...

3. **Modify ADS "S_ParamsLargeSignal" Design and/or Data Display Template – Uses Harmonic Balance with SNP_Eqn Component(Couplers), Injects Small-Signal Frequency Into Load, Simulates All a1, a2, b1, and b2 S-parameters:**

    - Also, a thrid alternative would be to consider the following template in ADS RF/Analog schematic window: **Insert/Template.../S_ParamsLargeSignal...** This template uses two S4P_Eqn components as couplers to pick off a1, a2, b1, and b2 s-parameter component signals--the device under test can be placed only one time into the template. Note that the P_1Tone source at "Vout" node injects a small-signal into the output of the device, while the input is being driven by a large signal at the swept RF frequency.
        - Here is a picture of the S_ParamsLargeSignal ADS RF/Analog template. (Note: It includes parameter sweeps for RF Power and RF Frequency as well as device biasing. If you did not need these features, you could easily remove them from the template.)


Large-Signal S-Parameters, Stability Factor, and Group Delay versus input power and Frequency

P_1Tone PORT1 Num=1 Z=Z0 P=dbmtow(RFpower) Freq=RFfreq

S4P_Eqn SimpleCoupler1 S[2,1]=1 S[3,1]=1 S[4,2]=1

V_DC SRC2 Vdc=Vlow

L L1 L=1.0 uH R=

C C3 C=1.0 uF

R R3 R=Z0

R R4 R=Z0

LARGE-SIGNAL S11, S12

L L2 L=1.0 uH R=

V_DC SRC1 Vdc=Vhigh

S4P_Eqn SimpleCoupler S[1,2]=1 S[2,1]=1 S[3,1]=1 S[4,2]=1

C C2 C=1.0 uF

R R2 R=Z0

R R1 R=Z0

LARGE-SIGNAL S22, S21

P_1Tone PORT2 Num=2 Z=Z0 P= Freq= P_USB=dbmtow(0)

This source injects a small-signal into the output of the device, while the input is being driven by a large signal at the swept RF frequency. The injected signal is at a 1 kHz offset from the input signal.

DisplayTemplate disptemp1 "S_ParamsLargeSignal"

ParamSweep Sweep1 SweepVar="RFfreq" SimInstanceName[1]="HB1" SimInstanceName[2]= SimInstanceName[3]= SimInstanceName[4]= SimInstanceName[5]= SimInstanceName[6]= Start=0.5 GHz Stop=4 GHz Step=0.5 GHz

HarmonicBalance HB1 Freq[1]=RFfreq Order[1]=11 SS_MixerMode=yes SS_Freq=1.0 KHz MergeSS_Freqs= UseKrylov=yes SweepVar="RFpower" Start=0 Stop=15 Step=2.5

VAR VAR1 Z0 =50 Vhigh=6 V Vlow=-0.5 V

VAR VAR2 RFfreq=1.0 GHz RFpower=0 _dBm

Set the following parameters:
1) Range of analysis frequencies on Parameter Sweep controller.
2) Range of input power levels on Harmonic Balance simulation controller.
3) Bias voltages and characterstic impedance.

Note
Example prior to ADS2011 can be located at S-Parameters Versus Bias.

S-Parameters Versus Bias

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/sp_vs_bias_wrk

Description

This example shows how to bundle a series of S- (and N-) parameter measurements at different bias conditions into one MDIF file and use that inside ADS. The example is a FET transistor that is measured at different VDS and IDS conditions.

One benefit with this technique is that you can specify bias-points that lies in between the measured points. ADS will then interpolate the S-parameters from the nearby measured bias/frequency points.

Note
Example prior to ADS2011 can be located at Swept Optimization and Simulation (AEL Script).

Swept Optimization/Simulation

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/swept_optimization_wrk

Description

Problem: You have a set of different files of measured S-parameters and you need to create a matching network to match them to 50 Ohms.

This example shows several different methods of doing this. sweptopt2 optimization generates a different matching network for each of the different measured S-parameter files. sweptopt3 generates a single matching network that is optimized to work with all of the measured S-parameter files.

sweptopt_2_orig uses an AEL file to run the swept optimization (this was necessary up to ADS 1.5 releases). It generates one generic MDIF file with all optimum values. There is no reason to use this unless you want to create an MDIF file.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Synthesizing Geometries of Inductors.

Synthesizing Geometries of Inductors Based on Desired L and Q

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/Inductor_Synthesis_wrk

Description

The design problem of interest to most users of spiral inductors, whether single (microstrip) or multilayer (LTCC, Silicon, RF Board), is to determine the geometry of such a spiral based on a desired value of inductance, L, and quality factor, Q. Until recently, modeling of inductors was a very complex task with no real practical method to synthesize geometries based on a specified L and Q. Due to the availability of Advanced Model Composer, this synthesis task is now a practical possibility. This example file does not focus on the generation of the required data, only on the synthesis aspect of the device using contour functions in the data display window.

The example file contains a large dataset that is generated by doing a multi-dimensional sweep of an AMC generated spiral model. The assumption is that one has a spiral with a fixed spacing and fixed number of turns/layers in this data file, and that the AMC model was generated over a continuous width, W, and diameter, D. Then the circuit simulator is run over the entire range of W, D, and freq to cover the complete design space.

A user goes to page "Synthesis" of the self_extract.dds file. First, the slider bars are used to choose the frequency. Then one reads the possible values of L (min/max). Next, using the equation "self=4", the user types in a desired value of inductance. Finally, one slides the Q-value for this inductance. If unrealizable values are selected, then the contour plots will display traces as "Invalid". If they are possible, then the user will find the red L_iso_synthesis trace and the blue Q_iso_synthesis trace. These are contours centered around a narrow range of the desired values. The magenta iso_L_extract1 and the cyan iso_Q_extract1 contours display the entire design space. If the synthesized L and Q traces intersect, then a device with the specified properties is realizable by reading off the diameter on the X-axis and the width on the Y-axis. Note that the Y-axis is not labeled as "W" but instead shows the names of the contour traces.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Time-Domain Optimization.

Time-Domain Optimization; Improving the Rise Time of a Signal

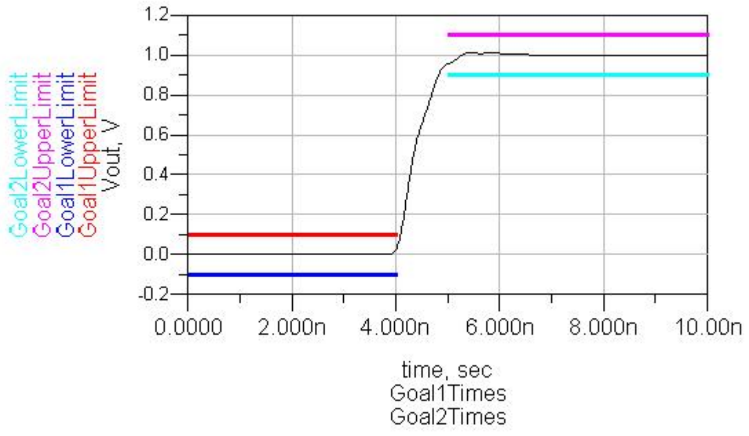
Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/FilterRiseTimeOpt_wrk

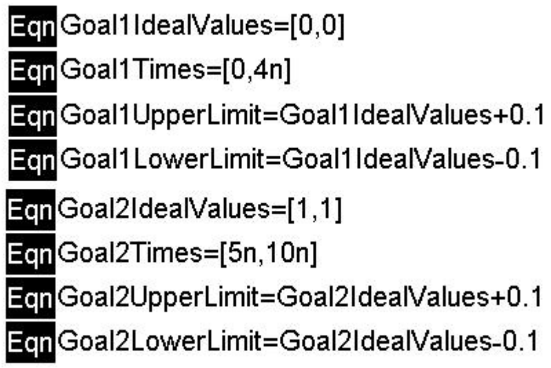
Description

This example shows three different ways of optimizing the rise time of a step signal passing through a simple low-pass filter.

- Low_Pass_Filter_Opt1 utilizes two TimeStamp components to measure when the output signal passes through the 10% and 90% thresholds. The filter 3-dB bandwidth, passband attenuation, and stopband attenuation are also included in the optimization.

- Low_Pass_Filter_Opt2 optimizes the rise time by specifying two different windows in the time domain inside which the output signal waveform must reside.
- Low_Pass_Filter_Opt3 optimizes the output voltage versus time to match an ideal piecewise linear function that you specify.





Eqn Goal1IdealValues=[0,0]

Eqn Goal1Times=[0,4n]

Eqn Goal1UpperLimit=Goal1IdealValues+0.1

Eqn Goal1LowerLimit=Goal1IdealValues-0.1

Eqn Goal2IdealValues=[1,1]

Eqn Goal2Times=[5n,10n]

Eqn Goal2UpperLimit=Goal2IdealValues+0.1

Eqn Goal2LowerLimit=Goal2IdealValues-0.1

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at TOI and SOI Example.

TOI & SOI example

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/TOI_MEAS_wrk

Description

This example contains a simple design to show how to set up ADS to measure the third and second-order intercept point of a non-linear device.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Two-Tone Loadpull Simulation.

Two-Tone Loadpull Simulation using Envelope Simulator; Potentially Better Convergence

Location (in ADS 2011+):
$HPEESOF_DIR/examples/KC_Examples/Loadpull/LoadPullACPR_R2_wrk

Description

June 24, 2008 Update: This example is now identical to the LoadPull_prj example that is included in the ADS 2008 Update 2 release. The PAE (power-added efficiency) calculations have been updated.

This example shows how to run a two-tone load pull simulation using the Envelope simulator. This technique uses a single large signal analysis tone that is effectively amplitude modulated by the Envelope simulator to generate two tones. It uses less memory and may allow convergence at a higher input power level than when a two-tone harmonic balance load pull is run. This example includes comparisons between the harmonic balance and Envelope techniques, which show nearly identical results.

See Also

Loadpull DesignGuide for ADS 2009 Update 1 with new content (This is the most advanced Load Pull DesignGuide for use with ADS 2009 Update 1, but it does not have the enhancements of the ADS 2011 version.)

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at VCO Behavioral Model from Simulated Data.

VCO Behavioral Model from Simulated Data

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/VCO_BehModwPN_wrk

Description

This example shows how to reuse data (frequency, fundamental signal amplitude, and phase noise) from a transistor-level HB simulation of an oscillator, in a behavioral model. This type of model would be useful for simulating phase-locked loops with ADS.

- VCO_HB simulates the transistor oscillator subcircuit (VCO_5_2GHz), versus tuning voltage. The dataset includes frequency, amplitude and phase noise data.
- VCO_5_2GHz is a 5.2 GHz VCO that uses the eesofDemoKit PDK.
- VCOsubckt comes from the MOS_VCO_prj example in examples/RFIC, and is an alternative VCO that could be used.
- VCO_Data is a behavioral model of the 5.2 GHz oscillator that re-uses the harmonic balance simulation results but does not include phase noise.
- test_VCO_Data simulates the behavioral model.
- VCOwPN_Data is the same as VCO_Data, except that it includes phase noise data in the model.
- test_VCOwPN_Data simulates the phase noise generated by the behavioral model.

Knowledge Center examples are experimental workspaces; please see *Disclaimer* (examples).

Note
Example prior to ADS2011 can be located at Waveprobe Model to Measure Forward and Reverse Power.

Waveprobe Model to Measure the Forward and Reverse Wave and also the Power Delivered

Location (in ADS 2011+): $HPEESOF_DIR/examples/KC_Examples/WaveProbe_wrk

Description

The attached workspace shows the approach that uses a WaveProbe. It's inserted just like a current probe and has a single parameter Z0 to
define the characteristic wave impedance. It then computes and stores to the dataset the forward wave, the reverse wave, and the power delivered (in dBm).

In the dataset presentations, you can then define whatever ratio of these you would like: to show reflection/impedance at intermediate points, gain to intermediate points, delivered power, distortion, etc.

It should work for all analyses except for noise, unfortunately. For that, you probably still need the coupler or some other approach.

If you add a ParamSweep to switch the source from the input to the output, you could then compute and display forward and reverse parameters.

Changes were recently done to the original example and a method to do "impedance probing" has now been added.

There are 4 designs:

- **WaveProbe :** Actual subcircuit which contains the equations to measure the forward and reverse waves.
- **TestWaveProbe :** A Harmonic Balance simulation which shows the WaveProbe component being used
- **Network1 :** Shows how to convert the reflection coefficient into an input impedance.
- **Network2_SP :** Shows how to use the SProbe element but this gives the small-signal input impedance.

Microwave Circuit Examples

Examples of how to create and analyze microwave, hybrid, and MMIC design solutions and applications such as amplifiers, filters, mixers, oscillators, and subsystems.

- *2GHz BJT Low Noise Amplifier* (examples)
- *12GHz Two Section Microstrip Filter* (examples)
- *Computing ACPR, Modulated Output Power, and EVM from a 1-tone, Swept Harmonic Balance Simulation* (examples)
- *Design for Manufacturing Example Using Yield Sensitivity Histograms, DOE, and Sensitivity Analysis* (examples)
- *Design of a 1GHz Low Noise Amplifier* (examples)
- *EM Simulation of MMIC passive components and circuits* (examples)
- *Large Signal Amplifier Simulations* (examples)
- *MMIC Amplifier* (examples)
- *MMIC Oscillator* (examples)
- *Optimizing A Linear FET Model to Match Measured S-Parameters* (examples)
- *Statistical Design of an X-Band LNA* (examples)
- *Test Lab for Two Stage Amplifier Design* (examples)
- *Using the Design Rule Checker (DRC)* (examples)
- *Using SP_Probes to design a simple, two-stage LNA* (examples)
- *Yield Sensitivity Histogram Design Templates* (examples)
- *Using Momentum to simulate an entire amplifier layout* (examples)

2GHz BJT Low Noise Amplifier

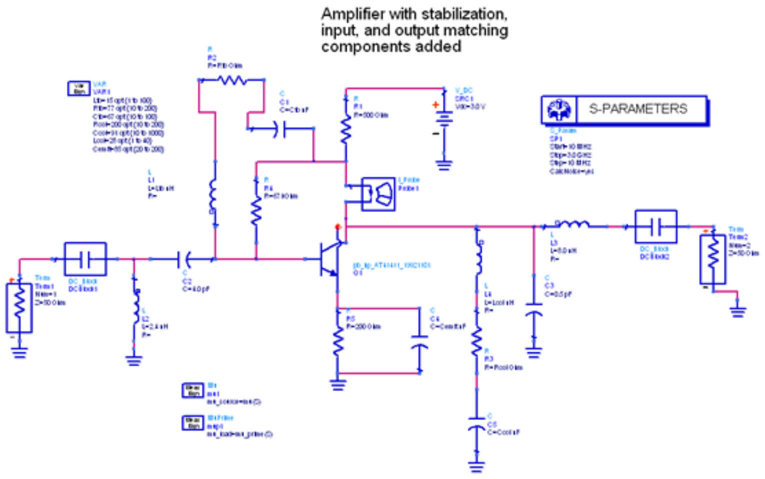
Location: $HPEESOF_DIR/examples/MW_Ckts/LNA_wrk

Objective

This example shows a simple procedure for designing a low-noise amplifier. The specifications are: 3 V supply, 2 mA collector current, lowest noise at 2 GHz, gain > 10 dB at 2 GHz, input and output reflection coefficients better than -10 dB at 2 GHz and unconditionally stable.

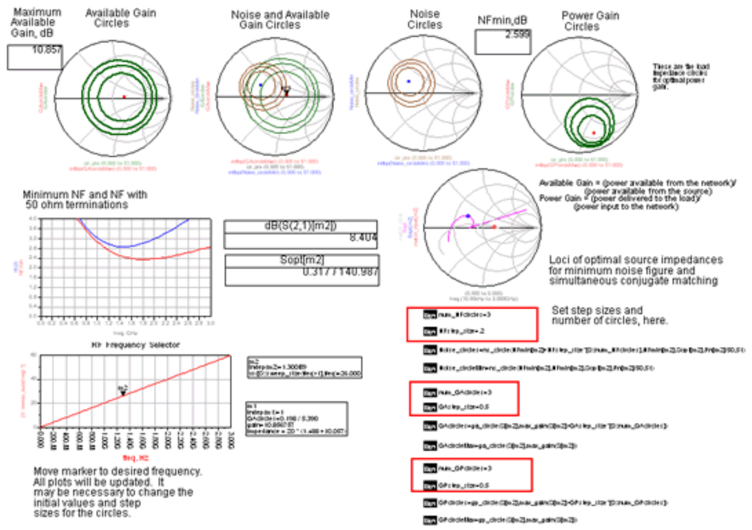
Setup

1. "Curve_Tracer" simulates the collector current versus collector voltage with the base current swept as a parameter.
2. "DC_and_Sparams" shows a bias network design, and includes a simulation of the S-parameters and a calculation of the stability of the biased device.
3. "Gain_and_Stab_opt" shows the addition of an optimized stabilizing networks added to achieve unconditional stability, simulated from 0 to 3 GHz.
4. "Circles" simulates the S-parameters, noise parameters and stability parameters before the addition of input and output matching networks.
5. "Input_match" designs the input matching network to generate a source impedance that is a tradeoff between gain and noise figure.
6. "Amp_wInputMatch" shows a simulation of the amplifier with both stability networks and the input matching network added.
7. "Output_match" designs the output matching network for simultaneous conjugate match.
8. "Amp_wBothMatches" simulates the amplifier with stability networks and both matching networks added. The results are not as good as expected, so an optimization is run next.
9. "Amp_wMatchOpt" is an optimization of the input and output matching component values to achieve a better gain, input match, and output match at 2 GHz, while not degrading the noise figure.
10. "DC_OP_POINT" simulates the colelctor current versus base voltage, with the collector voltage set to 3 V.


Amplifier with stabilization, input, and output matching components added
S-PARAMETERS

Analysis

Figure 1: Gain and Noise Circles



12GHz Two Section Microstrip Filter

Location: $HPEESOF_DIR/examples/MW_Ckts/mw_filter_wrk

Objective

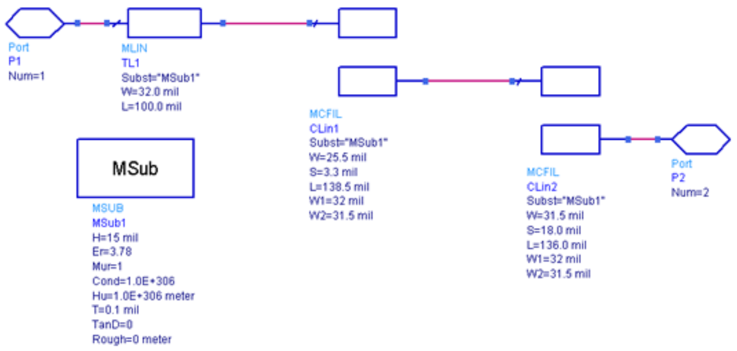
Demonstrates a simple bandpass filter composed of two concatenated microstrip subnetworks to achieve the desired performance at 12 GHz. It illustrates the Advanced Design System (ADS) feature of a microstrip circuit simulation and auto creating the layout from the schematic.

Setup

1. "halffil" is composed of two section of coupled microstrip line, which is going to be used as a subnetwork in the following design.
2. "cmstpfil" uses subnetworks from "halffil" to design a simple bandpass filter. Layout can be generated directly from the schematic.
3. "spar_sim" simulates the filter response and shows a way to measure the bandwidth.

Analysis

Figure 1: Filter layout



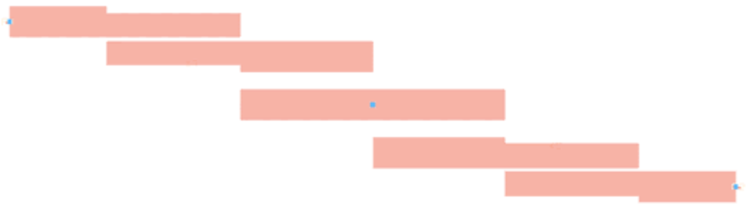
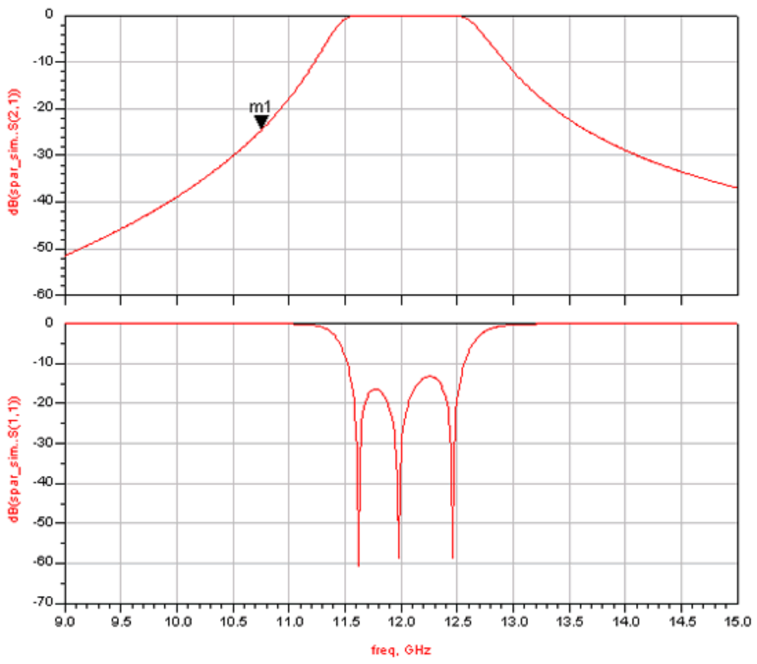


Figure 2: Filter response



Notes

- The layout for this circuit was created automatically and can be regenerated by the user. To do this, first deactivate the S-parameter simulation components in the schematic. Second, initiate the autolayout function from the schematic window via Layout > Generate/Update Layout. In the dialog box, set the Starting Component = X1. The resulting layout will be displayed.
- Simulation used: S-Parameter.

Computing ACPR, Modulated Output Power, and EVM from a 1-tone, Swept Harmonic Balance Simulation

Location: $HPEESOF_DIR/examples/MW_Ckts/ACPR_From_1_Tone_Swp_wrk

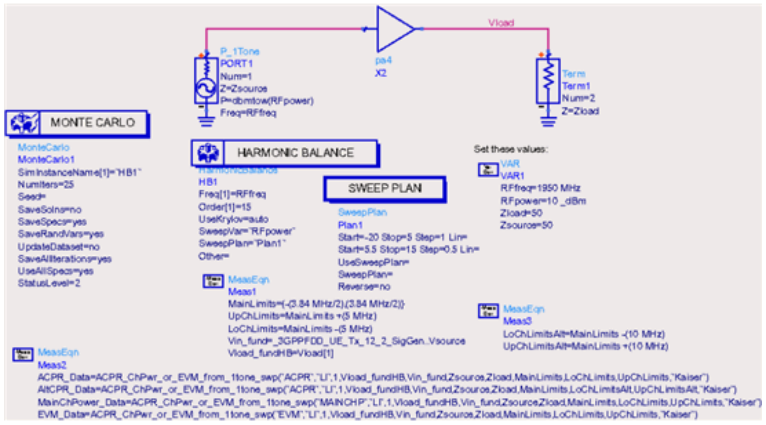
This describes a fast way of computing ACPR, modulated output power, and/or EVM of an amplifier versus input power from a 1-tone, swept harmonic balance simulation. This is an outline of the steps (many of these are carried out by an AEL function):

1. Run a swept input power harmonic balance simulation of the amplifier.

2. Determine the amplitude-out-versus-amplitude-in (some would call this AM-to-AM) and phase-out-versus-phase-in (AM-to-PM) transfer functions of the amplifier from the harmonic balance simulation.
3. Obtain (from a separate simulation or from some file) the envelope-versus-time waveform of the modulated input signal. This should be the magnitude and phase of the signal.
4. Apply this envelope-versus-time signal to the transfer functions from step 2 above to compute the envelope-versus-time of the output signal.
5. Use the fs() function to compute the spectrum of the modulated output signal.
6. From this spectrum, compute the powers in the main, adjacent, and alternate channels. Compute the adjacent- and alternate-channel power ratios from these values.
7. Compute EVM as follows. First compute the mean phase and RMS amplitude differences between the input and output modulated signals. At each time point, the ideal modulated output signal magnitude and phase should be the input modulated signal shifted by the mean phase difference and scaled by the RMS amplitude difference. The magnitude of the distance between the modulated output signal and this ideal modulated output signal is the EVM at each time point. The computed EVM is the RMS value of all the EVM values versus time.

These various calculations can be done while including an extra sweep, for example, of a parameter value or with a Monte Carlo analysis. The calculations are carried out using special measurement expressions that have been added to ADS 2011 that you may place on the schematic or in the data display.

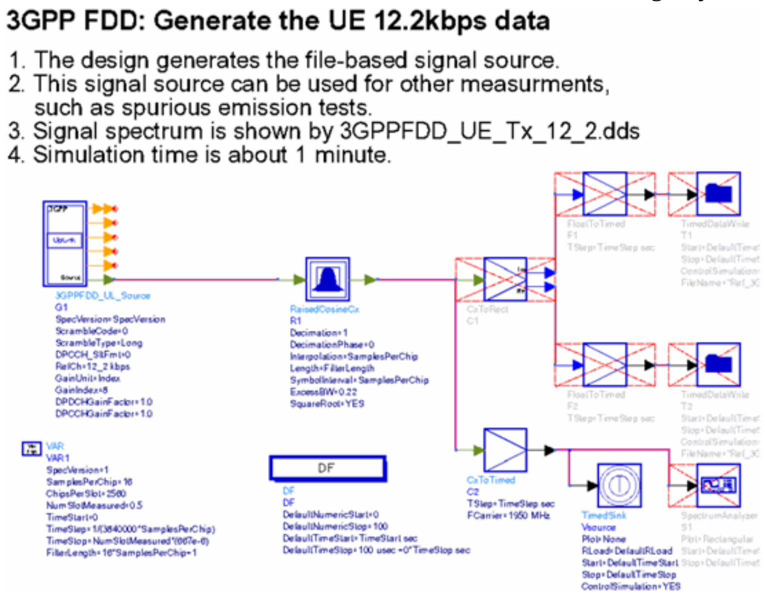
Here is an example applying the functions to compute ACPR, main channel power, and EVM:



```
P_1Tone
PORT1
Num=1
Z=Zsource
P=dbmtow(RFpower)
Freq=RFfreq

Term
Term1
Num=2
Z=Zload

MonteCarlo
MonteCarlo1
SimInstanceName[1]="HB1"
NumIters=25
Seed=
SaveSolns=no
SaveSpecs=yes
SaveRandVars=yes
UpdateDataset=no
SaveAllIterations=yes
UseAllSpecs=yes
StatusLevel=2

HarmonicBalance
HB1
Freq[1]=RFfreq
Order[1]=15
UseKrylov=auto
SweepVar="RFpower"
SweepPlan="Plan1"
Other=

SweepPlan
Plan1
Start=-20 Stop=5 Step=1 Lin=
Start=5.5 Stop=15 Step=0.5 Lin=
UseSweepPlan=
SweepPlan=
Reverse=no

VAR
VAR1
RFfreq=1950 MHz
RFpower=10 _dBm
Zload=50
Zsource=50

MeasEqn
Meas1
MainLimits={-(3.84 MHz/2),(3.84 MHz/2)}
UpChLimits=MainLimits +(5 MHz)
LoChLimits=MainLimits -(5 MHz)
Vin_fund=_3GPPFDD_UE_Tx_12_2_SigGen..Vsource
Vload_fundHB=Vload[1]

MeasEqn
Meas3
LoChLimitsAlt=MainLimits -(10 MHz)
UpChLimitsAlt=MainLimits +(10 MHz)

MeasEqn
Meas2
ACPR_Data=ACPR_ChPwr_or_EVM_from_1tone_swp("ACPR","LI",1,Vload_fundHB,Vin_fund,Zsource,Zload,MainLimits,LoChLimits,UpChLimits,"Kaiser")
AltCPR_Data=ACPR_ChPwr_or_EVM_from_1tone_swp("ACPR","LI",1,Vload_fundHB,Vin_fund,Zsource,Zload,MainLimits,LoChLimitsAlt,UpChLimitsAlt,"Kaiser")
MainChPower_Data=ACPR_ChPwr_or_EVM_from_1tone_swp("MAINCHP","LI",1,Vload_fundHB,Vin_fund,Zsource,Zload,MainLimits,LoChLimits,UpChLimits,"Kaiser")
EVM_Data=ACPR_ChPwr_or_EVM_from_1tone_swp("EVM","LI",1,Vload_fundHB,Vin_fund,Zsource,Zload,MainLimits,LoChLimits,UpChLimits,"Kaiser")
```

The "pa4" subcircuit is a power amplifier. A Monte Carlo simulation is run, and for each trial a swept-power harmonic balance simulation is run. For higher accuracy, we use a finer step size as the input power is increased driving the amplifier into compression. For the ACPR and main channel power calculations, the MainLimits equation sets the main channel bandwidth at +/-1.92 MHz. The UpChLimits equation sets the upper adjacent channel frequency limits to the same bandwidth as the main channel but offset by +5 MHz. The LoChLimits equation is similar. The LoChLimitsAlt and UpChLimitsAlt equations define the lower and upper alternate channel limits, respectively.

The Vin_fund equation reads in the envelope of the modulated signal from a dataset that was generated from an ADS example, examples/WCDMA3G/WCDMA3G_SignalSource_wrk/3GPPFDD_UE_Tx_12_2_SigGen schematic, with modifications as shown:



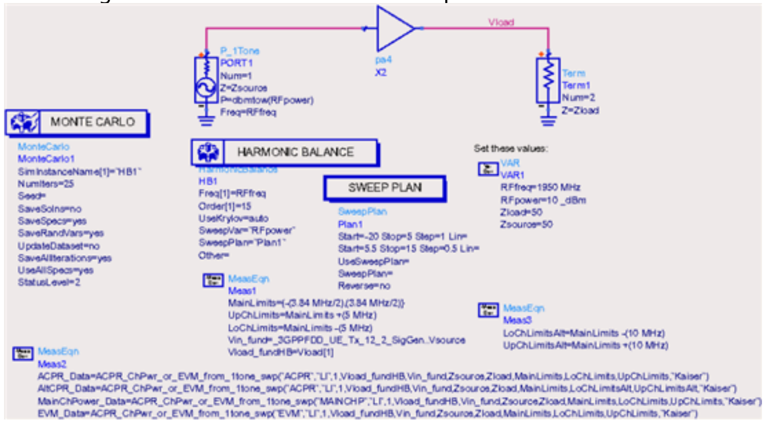
3GPP FDD: Generate the UE 12.2kbps data

1. The design generates the file-based signal source.
2. This signal source can be used for other measurments, such as spurious emission tests.
3. Signal spectrum is shown by 3GPPFDD_UE_Tx_12_2.dds
4. Simulation time is about 1 minute.

```
3GPPFDD_UL_Source
G1
SpecVersion=SpecVersion
ScrambleCode=0
ScrambleType=Long
DPCCH_SltFmt=0
RefCh=12_2 kbps
GainUnit=Index
GainIndex=8
DPDCHGainFactor=1.0
DPCCHGainFactor=1.0

RaisedCosineCx
R1
Decimation=1
DecimationPhase=0
Interpolation=SamplesPerChip
Length=FilterLength
SymbolInterval=SamplesPerChip
ExcessBW=0.22
SquareRoot=YES

CxToRect
C1

FloatToTimed
F1
TStep=TimeStep sec

TimedDataWrite
T1
Start=DefaultTime
Stop=DefaultTime
ControlSimulation
FileName="Ref_X

FloatToTimed
F2
TStep=TimeStep sec

TimedDataWrite
T2
Start=DefaultTime
Stop=DefaultTime
ControlSimulation
FileName="Ref_X

VAR
VAR1
SpecVersion=1
SamplesPerChip=16
ChipsPerSlot=2560
NumSlotMeasured=0.5
TimeStart=0
TimeStep=1/(3840000*SamplesPerChip)
TimeStop=NumSlotMeasured*(667e-6)
FilterLength=16*SamplesPerChip+1

DF
DF
DefaultNumericStart=0
DefaultNumericStop=100
DefaultTimeStart=TimeStart sec
DefaultTimeStop=100 usec +0*TimeStop sec

CxToTimed
C2
TStep=TimeStep sec
FCarrier=1950 MHz

TimedSink
Vsource
Plot=None
RLoad=DefaultRLoad
Start=DefaultTimeStart
Stop=DefaultTimeStop
ControlSimulation=YES

SpectrumAnalyzer
S1
Plot=Rectangular
Start=DefaultTime
Stop=DefaultTime
```

The stop time was reduced from 100 to 50 usec., which gives 3073 time points. This is a sufficiently long time record to give reasonable results. The greater the number of time points in the modulated data file, the longer the time required to compute ACPR, etc. However, it is possible to use a subset of the data. (In the ACPR_ChPwr_or_EVM_from_1tone_swp() function, use Vin_fund[0::1000] instead of Vin_fund to use just the first 1001 time points, for example.)

Returning to the harmonic balance setup:



```
P_1Tone
PORT1
Num=1
Z=Zsource
P=dbmtow(RFpower)
Freq=RFfreq

pa4
X2

Term
Term1
Num=2
Z=Zload

MonteCarlo
MonteCarlo1
SimInstanceName[1]="HB1"
NumIters=25
Seed=
SaveSolns=no
SaveSpecs=yes
SaveRandVars=yes
UpdateDataset=no
SaveAllIterations=yes
UseAllSpecs=yes
StatusLevel=2

HarmonicBalance
HB1
Freq[1]=RFfreq
Order[1]=15
UseKrylov=auto
SweepVar="RFpower"
SweepPlan="Plan1"
Other=

SweepPlan
Plan1
Start=-20 Stop=5 Step=1 Lin=
Start=5.5 Stop=15 Step=0.5 Lin=
UseSweepPlan=
SweepPlan=
Reverse=no

Set these values:
VAR
VAR1
RFfreq=1950 MHz
RFpower=10 _dBm
Zload=50
Zsource=50

MeasEqn
Meas1
MainLimits={-(3.84 MHz/2),(3.84 MHz/2)}
UpChLimits=MainLimits +(5 MHz)
LoChLimits=MainLimits -(5 MHz)
Vin_fund=_3GPPFDD_UE_Tx_12_2_SigGen..Vsource
Vload_fundHB=Vload[1]

MeasEqn
Meas3
LoChLimitsAlt=MainLimits -(10 MHz)
UpChLimitsAlt=MainLimits +(10 MHz)

MeasEqn
Meas2
ACPR_Data=ACPR_ChPwr_or_EVM_from_1tone_swp("ACPR","LI",1,Vload_fundHB,Vin_fund,Zsource,Zload,MainLimits,LoChLimits,UpChLimits,"Kaiser")
AltCPR_Data=ACPR_ChPwr_or_EVM_from_1tone_swp("ACPR","LI",1,Vload_fundHB,Vin_fund,Zsource,Zload,MainLimits,LoChLimitsAlt,UpChLimitsAlt,"Kaiser")
MainChPower_Data=ACPR_ChPwr_or_EVM_from_1tone_swp("MAINCHP","LI",1,Vload_fundHB,Vin_fund,Zsource,Zload,MainLimits,LoChLimits,UpChLimits,"Kaiser")
EVM_Data=ACPR_ChPwr_or_EVM_from_1tone_swp("EVM","LI",1,Vload_fundHB,Vin_fund,Zsource,Zload,MainLimits,LoChLimits,UpChLimits,"Kaiser")
```

the Vload_fundHB=Vload[1] equation specifies the complex fundamental output voltage which will be used in the ACPR_ChPwr_or_EVM_from_1tone_swp() function for the computation of the magnitude and phase transfer functions.

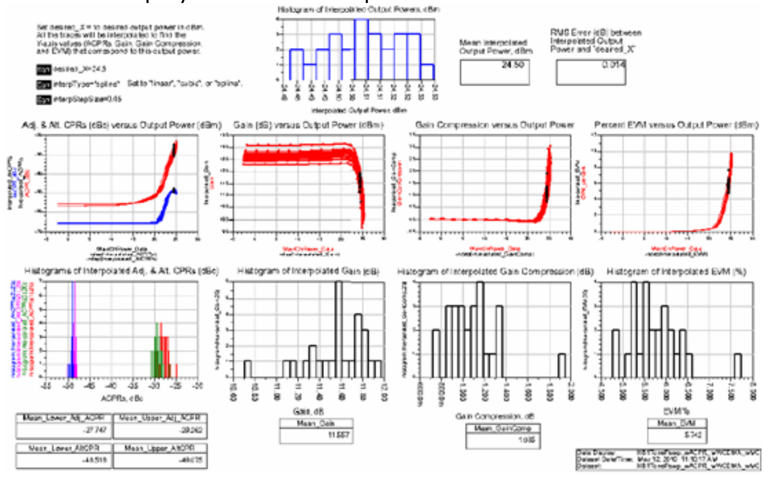
What the ACPR_ChPwr_or_EVM_from_1tone_swp() functions do
The ACPR_ChPwr_or_EVM_from_1tone_swp(returnVal, algorithm, allowextrap, charVoltage, inputSig, sourceZ, loadZ, mainCh, lowerAdjCh, upperAdjCh, winType, winConst) function computes the ACPR values (upper and lower), the main channel power, or the EVM. These are the passed parameters:

Parameter	Description
returnVal	flag for defining what to return. “ACPR” returns the ACPR, “MAINCHP” returns the main channel power, and “EVM” returns the error vector magnitude.
algorithm	flag for setting the algorithm for approximating the vout/vin transfer function. Set to “CF” for curve fit or “LI” for linear interpolation. Our experience has been that linear interpolation produces better results, although it takes longer.
allowextrap	flag for allowing or disallowing extrapolation. If set to 1, scale factors will be used to vary the power of the modulated input signal between the maximum input power from the harmonic balance sweep and this maximum power – 30 dB. If this flag is set to 0 then any scale factor that, when applied to the modulated input signal, would cause its peak value to exceed the maximum input power of the harmonic balance sweep, will not be allowed.
charVoltage	this is the characterization voltage (the fundamental output voltage from the harmonic balance sweep.) Example: Vload_fund, where Vload_fund=Vload[1].
inputSig	this is the input modulated signal (the envelope.) This signal should just be a function of time. See the above figure for an example.
sourceZ	this is the source impedance.
loadZ	this is the load impedance.
mainCh	these are the main channel frequency limits, as an offset from the carrier frequency. Example: {(-3.84 MHz/2),(3.84 MHz/2)}
lowerAdjCh	these are the lower adjacent (or alternate) channel frequency limits as an offset from the carrier frequency. Example: MainLimits – (5 MHz).
upperAdjCh	these are the upper adjacent (or alternate) channel frequency limits as an offset from the carrier frequency.
winType	window type (for example, “Kaiser”). Possible windows are exactly the same as those used in the fs() function.
winConst	window constant. Possible constant values are associated with each window type and are exactly the same as the values used in the fs() function. Leave blank to use the default value.

Here is the sequence of steps carried out by the ACPR_ChPwr_or_EVM_from_1tone_swp() function:

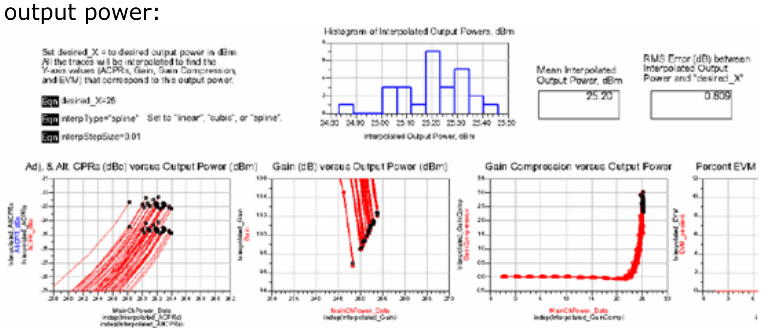
1. Compute the scale factors based on the maximum input power from the swept harmonic balance simulation (call this MaxInPwrHB.) These scale factors are hard-coded such that the average power of the modulated input signal will be scaled from MaxInPwrHB -30 dB to MaxInPwrHB -7.5 dB in 2.5 dB steps and from MaxInPwrHB -6 dB to MaxInPwrHB in 1 dB steps. This assumes extrapolation is allowed. If extrapolation is not allowed, then some of these higher scale factors may not be used.
2. Compute the Vout/Vin transfer function using either a curve fit or linear interpolation (preferred.)
3. Scale the input modulated signal using the scale factors from step 1 above.
4. Apply the scaled input signal (now a function of the scale factor and time) to the transfer function to get the output voltage versus both scale factor and time.
5. If returnVal is “ACPR” then return the ACPR versus the modulated input signal power.
6. If returnVal is “MAINCHP” then return the main channel power versus the modulated input signal power.
7. If returnVal is “EVM” then return the EVM (this is the raw EVM, not specification-compliant) versus the modulated input power.

This data display shows the computed results:


Histogram of Interpolated Output Powers, dBm
Mean interpolated Output Power, dBm
24.50
RMS Error (dB) between Interpolated Output Power and 'desired_X'
0.014
desired_X=24.5
interpType="spline" Set to "linear", "cubic", or "spline".
interpStepSize=0.05
Adj. & Alt. CPRs (dBc) versus Output Power (dBm)
Gain (dB) versus Output Power (dBm)
Gain Compression versus Output Power
Percent EVM versus Output Power (dBm)
Histograms of Interpolated Adj. & Alt. CPRs (dBc)
Histogram of Interpolated Gain (dB)
Histogram of Interpolated Gain Compression (dB)
Histogram of Interpolated EVM (%)
ACPRs, dBc
Gain, dB
Gain Compression, dB
EVM%
Mean_Lower_Adj_ACPR
-27.747
Mean_Upper_Adj_ACPR
-28.262
Mean_Lower_AltCPR
-44.518
Mean_Upper_AltCPR
Mean_Gain
11.557
Mean_GainComp
1.085
Mean_EVM
5.742

This shows Adjacent and Alternate channel power ratios versus output power, gain versus output power, gain compression versus output power, and EVM versus output power. You enter the desired output power value, desired_X, interpolation is carried out, and the histograms show the distribution of each response while the amplifier is delivering output power equal (approximately) to desired_X.

If you specify a desired_X that is beyond the x-axis range of the data, then just the maximum x-axis values of the traces will be used. This may result in a significant difference between the desired output power (desired_X) and the mean interpolated output power:



Set desired_X = to desired output power in dBm. All the traces will be interpolated to find the Y-axis values (ACPRs, Gain, Gain Compression, and EVM) that correspond to this output power.

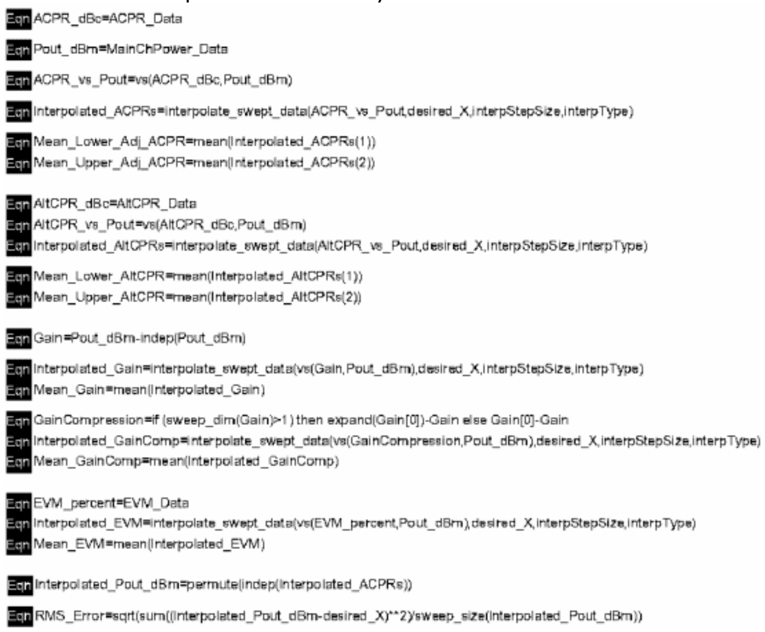
```
Eqn desired_X=26
Eqn interpType="spline"   Set to "linear", "cubic", or "spline".
Eqn interpStepSize=0.01
```

Mean Interpolated Output Power, dBm: 25.20

RMS Error (dB) between Interpolated Output Power and "desired_X": 0.809

Histogram of Interpolated Output Powers, dBm
Interpolated Output Power, dBm
Adj. & Alt. CPRs (dBc) versus Output Power (dBm)
Gain (dB) versus Output Power (dBm)
Gain Compression versus Output Power
Percent EVM

These are the equations that carry out these calculations:



```
Eqn ACPR_dBc=ACPR_Data
Eqn Pout_dBm=MainChPower_Data
Eqn ACPR_vs_Pout=vs(ACPR_dBc,Pout_dBm)
Eqn Interpolated_ACPRs=interpolate_swept_data(ACPR_vs_Pout,desired_X,interpStepSize,interpType)
Eqn Mean_Lower_Adj_ACPR=mean(Interpolated_ACPRs(1))
Eqn Mean_Upper_Adj_ACPR=mean(Interpolated_ACPRs(2))

Eqn AltCPR_dBc=AltCPR_Data
Eqn AltCPR_vs_Pout=vs(AltCPR_dBc,Pout_dBm)
Eqn Interpolated_AltCPRs=interpolate_swept_data(AltCPR_vs_Pout,desired_X,interpStepSize,interpType)
Eqn Mean_Lower_AltCPR=mean(Interpolated_AltCPRs(1))
Eqn Mean_Upper_AltCPR=mean(Interpolated_AltCPRs(2))

Eqn Gain=Pout_dBm-indep(Pout_dBm)
Eqn Interpolated_Gain=interpolate_swept_data(vs(Gain,Pout_dBm),desired_X,interpStepSize,interpType)
Eqn Mean_Gain=mean(Interpolated_Gain)
Eqn GainCompression=if (sweep_dim(Gain)>1) then expand(Gain[0])-Gain else Gain[0]-Gain
Eqn Interpolated_GainComp=interpolate_swept_data(vs(GainCompression,Pout_dBm),desired_X,interpStepSize,interpType)
Eqn Mean_GainComp=mean(Interpolated_GainComp)

Eqn EVM_percent=EVM_Data
Eqn Interpolated_EVM=interpolate_swept_data(vs(EVM_percent,Pout_dBm),desired_X,interpStepSize,interpType)
Eqn Mean_EVM=mean(Interpolated_EVM)

Eqn Interpolated_Pout_dBm=permute(indep(Interpolated_ACPRs))

Eqn RMS_Error=sqrt(sum((Interpolated_Pout_dBm-desired_X)**2)/sweep_size(Interpolated_Pout_dBm))
```

In this case, ACPR_Data, MainChPower_Data, AltCPR_Data, and EVM_Data were all calculated on the schematic using the ACPR_ChPwr_or_EVM_from_1tone_swp() function.

The interpolate_swept_data(original_data_Y_vs_X, desired_X, interpStepSize, interpType) function is another new function that has been added to ADS 2011. It returns the interpolated Y value that corresponds to (and is versus) the interpolated X value, which will be closest to desired_X. This result is returned as a single value or an array, depending on the dimensionality of the original_data_Y_vs_X argument. These are the passed parameters:

Parameter	**Description**
original_data_Y_vs_X	any 1-, 2-, or 3-dimensional set of curves of Y versus X data.
desired_X	desired X value, to be found by interpolation.
interpStepSize	the smaller this number is, the closer the interpolation will get to the desired x value.
interpType	"linear", "cubic" or "spline". This specifies the type of interpolation.

This function does not do any extrapolation. If the requested desired_X is above or below the maximum or minimum X value, then it just returns the Y value corresponding to the maximum or minimum X value, respectively.

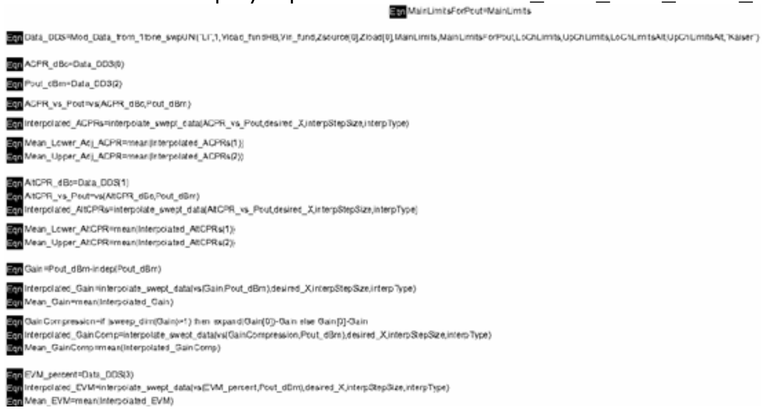
Using equations on the data display to compute the ACPRs, main channel power, or EVM

There is another custom function, Mod_Data_from_1tone_swpUNI() that returns the adjacent and alternate channel powers, main channel power, and EVM. Because this function returns multiple results, it cannot be used in a measurement expression on the schematic. It can only be used in the data display. The advantage of using this function instead of ACPR_ChPwr_or_EVM_from_1tone_swp() is that it is more efficient for obtaining all these results. When using ACPR_ChPwr_or_EVM_from_1tone_swp(), you have to call it once to get the adjacent channel power ratios, once again to get the alternate channel power ratios, once again to get the main channel power, and once again to get the EVM. The big disadvantage of using the Mod_Data_from_1tone_swpUNI() is that this function will get executed each time you open a data display that contains it. While it is slower to use the ACPR_ChPwr_or_EVM_from_1tone_swp() function on the schematic, the advantage is that the results are written into the dataset, and data displays that show these results open instantly.

The Mod_Data_from_1tone_swpUNI(algorithm, allowextrap, charVoltage, inputSig, sourceZ, loadZ, mainCh, mainChForPout, lowerAdjCh, upperAdjCh, lowerAltCh, upperAltCh, winType, winConst) function computes the ACPR values (upper and lower), Alternate CPR values (upper and lower), the main channel power, and the EVM. These are the passed parameters:

Parameter	**Description**
algorithm	flag for setting the algorithm for approximating the vout/vin transfer function. Set to “CF” for curve fit or “LI” for linear interpolation. Our experience has been that linear interpolation produces better results, although it takes longer.
allowextrap	flag for allowing or disallowing extrapolation. If set to 1, scale factors will be used to vary the power of the modulated input signal between the maximum input power from the harmonic balance sweep and this maximum power – 30 dB. If this flag is set to 0 then any scale factor that, when applied to the modulated input signal, would cause its peak value to exceed the maximum input power of the harmonic balance sweep, will not be allowed.
charVoltage	this is the characterization voltage (the fundamental output voltage from the harmonic balance sweep.) Example: Vload_fund, where Vload_fund=Vload[1].
inputSig	this is the input modulated signal (the envelope.) This signal should just be a function of time. See the figure above for an example.
sourceZ	this is the source impedance.
loadZ	this is the load impedance.
mainCh	these are the main channel frequency limits, as an offset from the carrier frequency. Example: {(-3.84 MHz/2),(3.84 MHz/2)}
mainChForPout	these are the frequency limits used for computing the modulated output power. Normally this would be the same as the mainCh argument, but this allows you to specify a different bandwidth for computing the modulated output power.
lowerAdjCh	these are the lower adjacent channel frequency limits as an offset from the carrier frequency. Example: MainLimits – (5 MHz)
upperAdjCh	these are the upper adjacent channel frequency limits as an offset from the carrier frequency.
lowerAltCh	these are the lower alternate channel frequency limits as an offset from the carrier frequency. Example: MainLimits – (10 MHz)
upperAltCh	these are the upper alternate channel frequency limits as an offset from the carrier frequency. Example: MainLimits + (10 MHz)
winType	window type (for example, “Kaiser”). Possible windows are exactly the same as those used in the fs() function.
winConst	window constant. Possible constant values are associated with each window type and are exactly the same as the values used in the fs() function. Leave blank to use the default value.

This set of data display equations uses the Mod_Data_from_1tone_swpUNI() function.



```
Eqn MainLimitsForPout=MainLimits
Eqn Data_DDS=Mod_Data_from_1tone_swpUNI("LI",1,Vload_fundHB,Vin_fund,Zsource[0],Zload[0],MainLimits,MainLimitsForPout,LoChLimits,UpChLimits,LoChLimitsAlt,UpChLimitsAlt,"Kaiser")
Eqn ACPR_dBc=Data_DDS(0)
Eqn Pout_dBm=Data_DDS(2)
Eqn ACPR_vs_Pout=vs(ACPR_dBc,Pout_dBm)
Eqn Interpolated_ACPRs=interpolate_swept_data(ACPR_vs_Pout,desired_X,interpStepSize,interpType)
Eqn Mean_Lower_Adj_ACPR=mean(Interpolated_ACPRs(1))
Eqn Mean_Upper_Adj_ACPR=mean(Interpolated_ACPRs(2))
Eqn AltCPR_dBc=Data_DDS(1)
Eqn AltCPR_vs_Pout=vs(AltCPR_dBc,Pout_dBm)
Eqn Interpolated_AltCPRs=interpolate_swept_data(AltCPR_vs_Pout,desired_X,interpStepSize,interpType)
Eqn Mean_Lower_AltCPR=mean(Interpolated_AltCPRs(1))
Eqn Mean_Upper_AltCPR=mean(Interpolated_AltCPRs(2))
Eqn Gain=Pout_dBm-indep(Pout_dBm)
Eqn Interpolated_Gain=interpolate_swept_data(vs(Gain,Pout_dBm),desired_X,interpStepSize,interpType)
Eqn Mean_Gain=mean(Interpolated_Gain)
Eqn GainCompression=if (sweep_dim(Gain)=1) then expand(Gain[0])-Gain else Gain[0]-Gain
Eqn Interpolated_GainComp=interpolate_swept_data(vs(GainCompression,Pout_dBm),desired_X,interpStepSize,interpType)
Eqn Mean_GainComp=mean(Interpolated_GainComp)
Eqn EVM_percent=Data_DDS(3)
Eqn Interpolated_EVM=interpolate_swept_data(vs(EVM_percent,Pout_dBm),desired_X,interpStepSize,interpType)
Eqn Mean_EVM=mean(Interpolated_EVM)
```

The only differences in these equations relative to those shown earlier are:
Data_DDS=Mod_Data_from_1tone_swpUNI(...)
ACPR_dBc=Data_DDS(0)
Pout_dBm=Data_DDS(2)
AltCPR_dBc=Data_DDS(1)
EVM_percent=Data_DDS(3)

The plots are all exactly the same.

Benchmark test results

A simulation of a much more complex amplifier with these custom equations just on the data display
took about 4 minutes for 10 MC trials (+1 for the nominal case) and about 1 minute and 31 seconds to open the data display. This was using a modulated input signal with 3073 time points.

The same simulation with 4 equations, each using the ACPR_ChPwr_or_EVM_from_1tone_swp() function to compute the adjacent channel power ratios, alternate channel power ratios, main channel power, and EVM took about 3 minutes 54 seconds to run the simulation and another 4 minutes and 45 seconds to evaluate the equations.

Technical discussion comparing this 1-tone swept power harmonic balance simulation technique for computing ACPR and EVM and the Ptolemy co-simulation approach.

For EVM, the single-tone method is not specification-compliant. It just measures the “raw” EVM, computed at each time point. The EVM is computed after correcting for the average phase difference and RMS amplitude difference between the output and input modulated signals. If the modulated signal at the output of the amplifier has only a constant phase shift and a constant gain (meaning that neither vary with the amplitude of the input modulated signal), then the EVM will be zero. With this method, the EVM is computed at each time point, not at just the symbol times. There is no demodulation or decoding of the signal, so you can’t calculate the EVM of each sub-carrier, say for an LTE signal.

With the Ptolemy method, the EVM calculation uses an EVM sink that is specification-compliant.

The only advantage of the single-tone method is that it may be faster. Clearly it is not as accurate, but might still be useful as a proxy for the specification-compliant EVM.

For ACPR, the single-tone method does not include any receive-side filtering. It just generates the spectrum at the output of the amplifier, integrates the power in the main, adjacent, and alternate channels, then computes the ratios.
The single-tone method of computing EVM (and ACPR) will tend to become less accurate as the bandwidth of the signal gets larger. This is because this method assumes the response of the amplifier is constant across the modulation bandwidth (we’re modeling the nonlinearity by injecting a single tone at the carrier frequency, after all.)

With Ptolemy fast co-simulation, the behavioral model that is created can include frequency variation across the modulation bandwidth.

Design for Manufacturing Example Using Yield Sensitivity Histograms, DOE, and Sensitivity Analysis

Location: $HPEESOF_DIR/examples/MW_Ckts/Ku_Band_LNA_DFM_wrk

Objective

Demonstrates all of the statistical design tools in ADS used to build robust designs that are insensitive to process variations, temperature variations, changes in the supply voltages, and package and bond wire effects.

Setup

To work with this example, very little setup is needed. All of the circuit designs are set to begin using immediately:

1. Install the design kit *DemoKit_V3* which is included in this workspace.
2. Open any of the circuit designs.
3. Simulate the design. The results will appear when the simulation completes.

The following table describes the designs and templates that are available in the example workspace. This workspace is organized into the following sub-folders which are viewable on the ADS Main window's Folder View tab:

Folder	Description
A_Sensitivity_Analysis	Illustrates the Sensitivity Analysis tool to find the sensitive elements to the specs in the design.
B_Yield_Analysis	Illustrates Monte Carlo Yield Analysis.
C_Yield_YSH_Templates	Illustrates how the post processes yield information is used in *Yield Sensitivity Histograms Templates* to extract very useful information on the design and help turn it into a highly manufacturable design with first-pass success and high yield. For details about using these templates, see *Yield Sensitivity Histogram Design Templates* (examples).
D_DOE_on_Matching_Networks	Illustrates the use of Design of Experiments (DOE) to find out which matching network(s) are sensitive and cause problems in the yield. Designers are then able to come up with other less-sensitive matching networks to make the design less sensitive to process variations.
E_DOE_on_elements	Illustrates the use of Design of Experiments (DOE) to find out which component is sensitive in the design and cause problems in yield.
F_Fix_Design	Based on the analysis done above, this is the fixed design that results with high yield and low sensitivity to process variation, temperature and voltage variation.
G_Network_to_test_YSH_templates	This is a dummy circuit that is included here in order to illustrate how to use of the Yield Sensitivity Histogram templates outside of this example. Please also see *Yield Sensitivity Histogram Design Templates* (examples) for more information on the use of these templates. These displays are available as templates which can be inserted using the *Insert* > *Template* command in the Data Display window. On your own designs, select the required number of specifications (1-5) and set the name of each Yield Spec to *Spec1*, *Spec2*, etc.

Notes

- To use this workspace, you must first install the design kit *DemoKit_V3* which is included in this workspace.

Design of a 1GHz Low Noise Amplifier

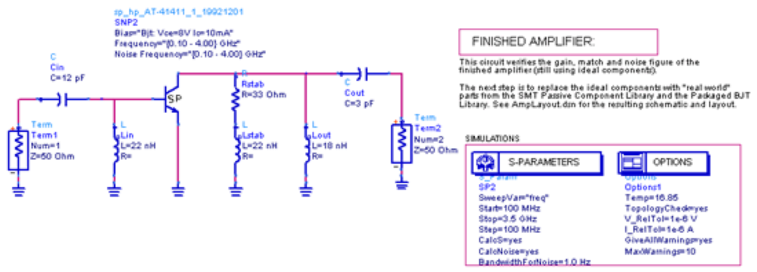
Location: $HPEESOF_DIR/examples/MW_Ckts/LNA_1GHz_wrk

Objective

This example shows how to examine stability, noise and gain, and how to design the input and output matching networks for a 1GHz LNA. It also compares the simulation data with those from layout.

Setup

1. "ModelVerif" shows how to compare two sets of data: the device model and measured S-parameters for the device are simulated to validate the model.
2. "BiasSetup" determines the required VBE so that the operating point of the model matches that of the S-parameter-based component.
3. "Bias_Network" performs an optimization to calculate the required bias network resistors for the device.
4. "SparamsNoise" shows how to set up the S-parameter simulation control to calculate noise parameters for the device and how to generate constant noise figure circles.
5. "Stability" calculates and displays input and output stability circles for the device and also shows the effect of adding a stabilizing networks on the output.
6. "Match1" and "Match2" calculate the required input match network for minimum noise figure at 1GHz.
7. "Match3" and "Match4" calculate the required output matching network for the device with the optimum input noise match network in place.
8. "FinishedAmp" calculates gain, input and output match, and noise figure for the completed amplifier.
9. "AmpLayout" is a subcircuit containing the layout and schematic of the finished amplifier, replacing ideal components with models from the SMT Passive Component Library.
10. "SimFromLayout" contains the symbol for AmpLayout and simulation controllers. Results are compared to the ideal results from FinishedAmp.



FINISHED AMPLIFIER:

This circuit verifies the gain, match and noise figure of the finished amplifier (still using ideal components).

The next step is to replace the ideal components with "real world" parts from the SMT Passive Component Library and the Packaged BJT Library. See AmpLayout.dsn for the resulting schematic and layout.

```
S_Param                          Options
SP2                              Options1
SweepVar="freq"                  Temp=16.85
Start=100 MHz                    TopologyCheck=yes
Stop=3.5 GHz                     V_RelTol=1e-6 V
Step=100 MHz                     I_RelTol=1e-6 A
CalcS=yes                        GiveAllWarnings=yes
CalcNoise=yes                    MaxWarnings=10
BandwidthForNoise=1.0 Hz
```

Analysis

Figure 1:S22 and S11: ideal versus simulation from layout.

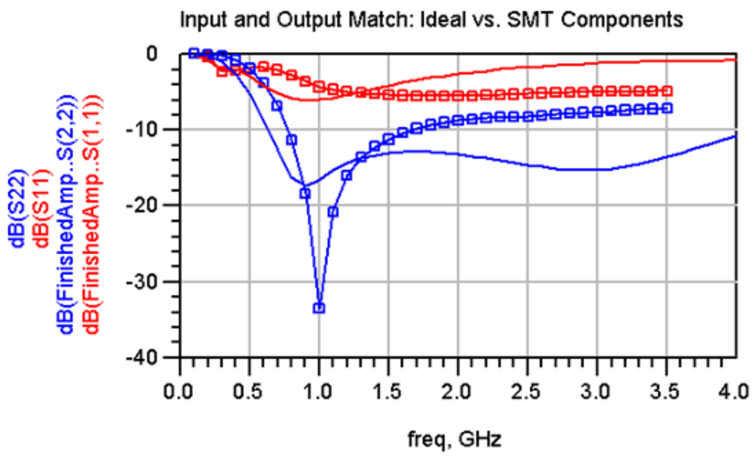


Figure 2:Amplifier gain comparison between ideal simulation and that from layout.

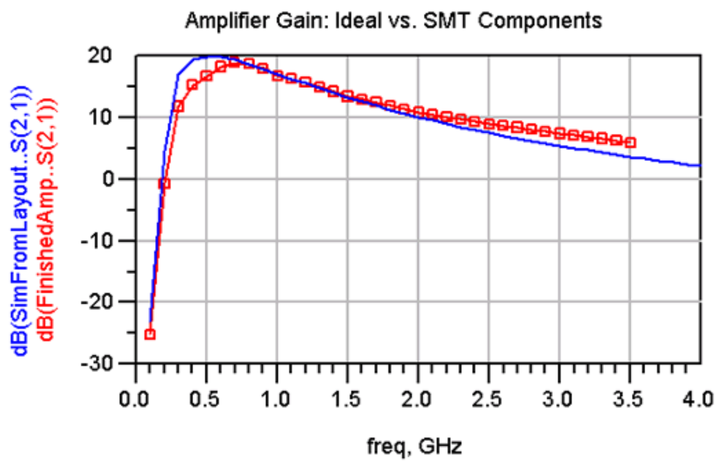


Figure 3:Noise figure comparison.

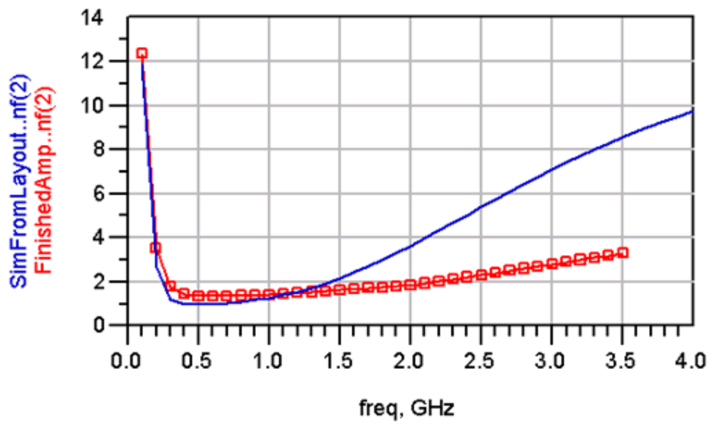
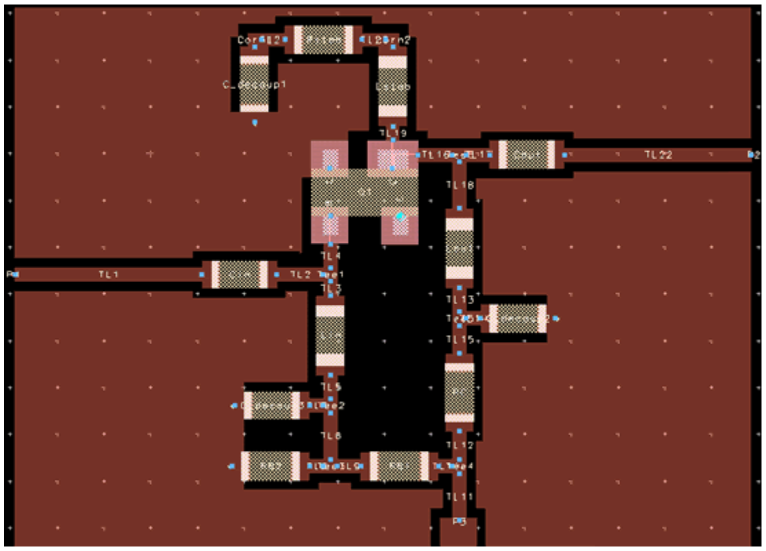


Figure 4:Amplifier layout.



Notes

- Some design files in this example use a transistor from the "Packaged BJT Library". "AmpLayout" also uses SMT passive components. You can edit these files and view results but in order to simulate, you must have a license for the Packaged BJT Library and the SMT Passive Components Library.

EM Simulation of MMIC passive components and circuits

Location: $HPEESOF_DIR/examples/MW_Ckts/MMIC_AmpEM_Sims_wrk

Objective

This example shows additional Electro-Magnetic simulations of inductors and the lumped-element branch-line coupler that is used in the MMIC_Amp_wrk example workspace. In this example, the equation based design kit models are compared with EM simulated results.

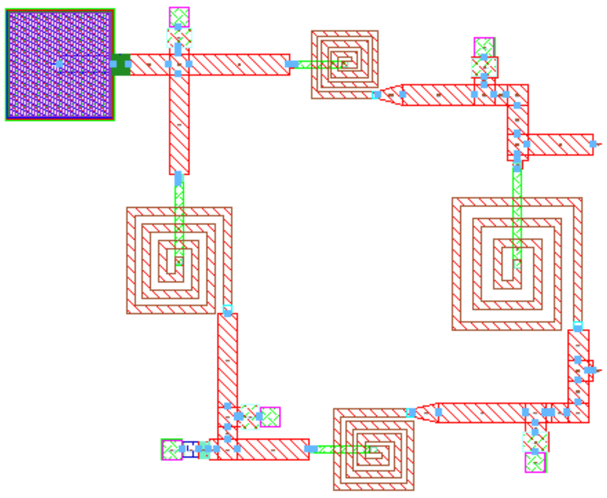
Setup

This example shows some electromagnetic simulations of inductors and the lumped-element branch-line coupler that is used in the MMIC_Amp_wrk example workspace. [NOTE: This example requires the DemoKit Design Kit be added. This is done from the Main Window of ADS, select the menu for Design Kit, Manage Libraries, and browse to the $HPEESOF_DIR/examples/DesignKit/DemoKit folder.]

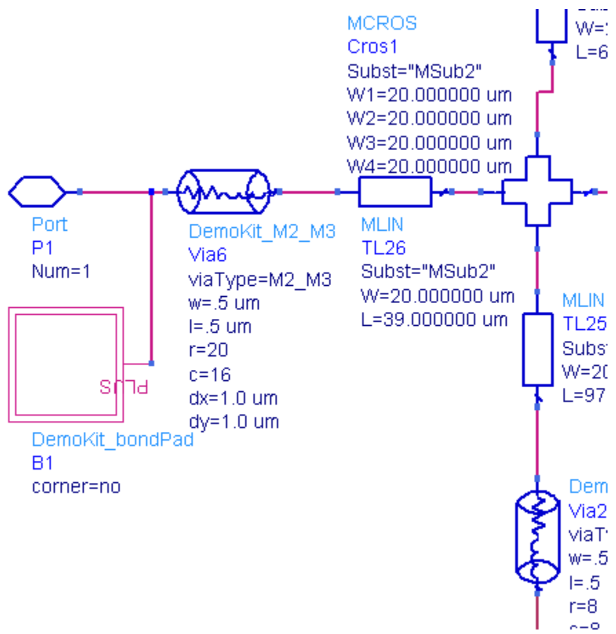
1. "L_0_356n_mom" is the layout of a 0.356 nH inductor from the DemoKit design kit, after several small modifications to make it easier to simulate using Momentum.
2. "L_0_356n_DK_SP" simulates the S-parameters of the same inductor, using the design kit model, which is equation based.
3. "L_0_356n_mom_a data display compares the S-parameters from the two simulations. The
4. "L_0_522n_mom" is the layout of a 0.522 nH inductor from the DemoKit design kit.
5. "L_0_522n_DK_SP" simulates the S-parameters of the same inductor, using the design kit model.
6. "The L_0_522n_mom_a" data display compares the S-parameters from the two simulations.
7. "BLC_Lumped_wTLsEM" shows the layout of the branch-line coupler that was first simulated
8. "BLC_LumpedEM_TB" shows the simulation of the coupler with the capacitors included. The
9. "BLC_LumpedR2_wTLs" is the branch-line coupler layout after moving the inductors away
10. "BLC_LumpedR2_wTLs_TB" shows the performance of this re-optimized coupler. The
11. "BLC_LumpedR2_wTLsEM" is the layout from which the Momentum simulation was run.
12. "BLC_LumpedR2EM_TB" shows the simulation of the revised coupler with the capacitors
13. In the "BLC_LumpedR2_wTLs" design, if one of the shunt inductor values is increased slightly,

Analysis

Layout of divider circuit



Partial schematic showing the Demo Kit components



Large Signal Amplifier Simulations

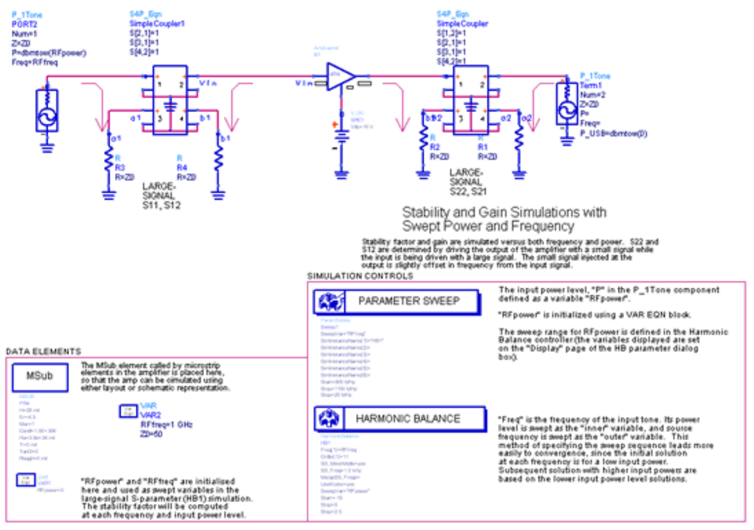
Location: $HPEESOF_DIR/examples/MW_Ckts/LargeSigAmp_wrk

Objective

This example provides details on how to use Advanced Design System (ADS) to characterize large-signal 1GHz RF amplifiers. It shows how to determine gain compression, third-order intercept (TOI) level, saturation output power, the output voltage spectrum, power dissipation and dynamic load line.

Setup

1. "AmpLayout" is the amplifier circuit, including layout. Steps for designing this circuit are shown in "LNA_1GHz_wrk". This design is to be used as a subnetwork by the other designs.
2. "HB1Tone" calculates gain, output voltage spectrum and voltage waveform.
3. "SweptPower" uses a single frequency, swept power level input signal to determine gain, compression, TOI (using single tone) and saturated output power.
4. "TOI_2Tone" shows the more rigorous two-tone method for calculating TOI.
5. "AmpPAE" calculates power dissipation and power-added efficiency (PAE) of the amplifier.
6. "DLL1" and "DLL2" generate the amplifiers dynamic load line graph.
7. "Stab_vs_freq_pwr" simulates the amplifiers stability factor and S-parameters versus input power level and frequency.



Stability and Gain Simulations with Swept Power and Frequency

Stability factor and gain are simulated versus both frequency and power. S22 and S12 are determined by driving the output of the amplifier with a small signal while the input is being driven with a large signal. The small signal injected at the output is slightly offset in frequency from the input signal.

The MSub element called by microstrip elements in the amplifier is placed here, so that the amp can be simulated using either layout or schematic representation.

"RFpower" and "RFfreq" are initialised here and used as swept variables in the large-signal S-parameter (HB1) simulation. The stability factor will be computed at each frequency and input power level.

The input power level, "P" in the P_1Tone component defined as a variable "RFpower".

"RFpower" is initialized using a VAR EQN block.

The sweep range for RFpower is defined in the Harmonic Balance controller (the variables displayed are set on the "Display" page of the HB parameter dialog box).

"Freq" is the frequency of the input tone. Its power level is swept as the "inner" variable, and source frequency is swept as the "outer" variable. This method of specifying the sweep sequence leads more easily to convergence, since the initial solution at each frequency is for a low input power. Subsequent solution with higher input powers are based on the lower input power level solutions.

Analysis

Figure 1: S21 versus RF power and frequency

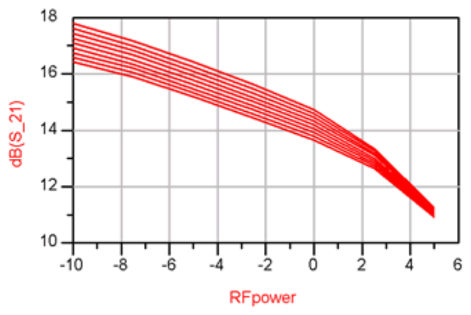
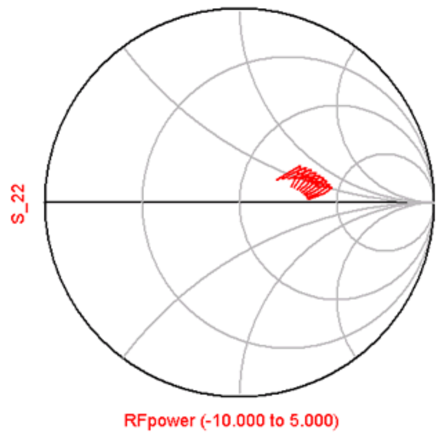


Figure 2: S22 versus RF power and frequency



Notes

- The user can use either a built-in IP3 function or write an equation in the two-tone test to calculate IP3.
- Simulation used: Harmonic Balance, Gain Compression, DC.

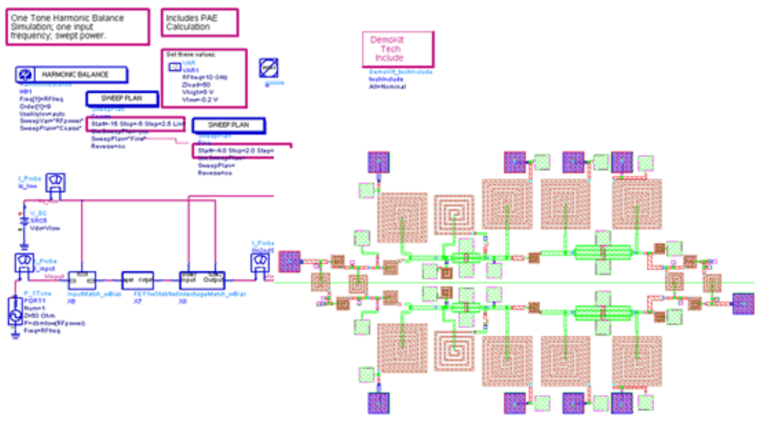
MMIC Amplifier

Location: $HPEESOF_DIR/examples/MW_Ckts/MMIC_Amp_wrk

Objective

This example shows the design of a 10 GHz, 0.5-Watt balanced amplifier. The finished design consists of two, two-stage amplifiers in parallel. The input signal to the pair of amplifiers is split via a branch line coupler, which is implemented with lumped elements to save space. The outputs from the two amplifiers are combined together via a branch line coupler identical to the one at the input.This example includes an evaluation of the HEMT devices used in the amplifier, including determination of device transconductance Gm versus bias and load pull analysis.This example uses elements from a "generic" design kit that was created to show some relatively simple examples of design kit elements. Design kits for real manufacturing processes would have many more elements, and the elements would have more detail, especially the transistors.

Complete Amplifier.



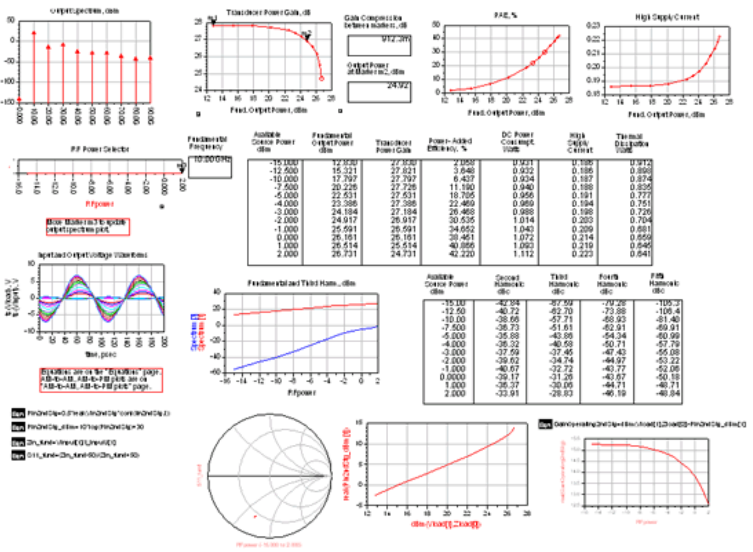
Setup

1. "FET_Gm_Calcs" simulates the transconductance Gm of a single device, versus bias. This enables you to pick a bias point where the Gm is a maximum to maximize gain.
2. "FET_SP_NF_Match_Circ" determines the optimal source impedance for minimum noise figure and the corresponding load impedance for maximum gain, both as a function of bias voltage. These impedances are used in the first stage amplifier design where gain is important.
3. Because the first stage device is quite unstable by itself, stabilization networks are added. The R, L, and C values were optimized initially using ideal lumped elements that are shown in "Gain_and_Stab_opt".
4. When these ideal elements are replaced with the equivalent R, L, and C components from the design kit, stability is greatly degraded. So it is necessary to run another optimization using the design kit elements, "Gain_and_Stab_DiscOpt". A discrete value optimization is run to keep the width and spacing of the inductor spiral lines multiples of 1 um and the number of turns an integer.
5. With the stabilization network added, the optimal source and load impedances to present to the stabilized device are determined using "SP_NF_GainMatchK".
6. "HB1Tone_LoadPullMagPh" simulates the load pull of the second-stage FET. The power-added efficiency varies strongly with the bias point.
7. "HB1Tone_SourcePull" simulates the source pull of the second-stage FET and indicates that the power delivered to the load does not depend much on the source impedance.
8. "InputMatch1" determines the ideal lumped-element values for a simple shunt-C, series-L matching network to generate the desired source impedance for the input stage. It uses a Passive Circuit DesignGuide lumped-element matching network component.
9. "InputMatch_wBias" includes an L-C bias network and design kit elements and a layout with interconnects modeled as traces rather than transmission lines.
10. "InputMatch_wBias_wTLs" is the result of converting all interconnect traces to transmission lines, and will give the most accurate modeling results, other than using EM simulation. The inductor value has been reduced to compensate for the parasitic inductance of the interconnect transmission lines.
11. "InterstageMatch1" has a simple shunt-C, series-L matching network generated from the Matching DesignGuide. This transforms the impedance seen looking into the second stage to the desired impedance to present to the first stage.
12. "InterstageMatch_wBias" includes an L-C bias network and design kit elements, and a layout with the interconnects modeled as traces rather than transmission lines.
13. "InterstageMatch_wBias_wTLs" is the result of converting most interconnect traces to transmission lines, and will give the most accurate modeling results, other than using EM simulation. The inductor value has been reduced to compensate for the parasitic inductances of the interconnect transmission lines.
14. "OutputMatch1" is used to determine component values for ideal series-C, shunt-L or series-L, shunt-C networks for generating the desired load impedance for the second stage.
15. "OutputMatch_wBias" includes an L-C bias network and design kit elements, and a layout with interconnects modeled as traces rather than transmission lines.
16. "OutputMatch_wBias_wTLs" is the result of converting most interconnect traces to transmission lines, and will give the most accurate modeling results, other than using EM simulation. The inductor value has been reduced to compensate for the parasitic inductances of the interconnect transmission lines.
17. "TwoStgAmpInZ_TB" uses an S-Probe pair to calculate the reflection coefficients looking both directions at both the input and output planes of the first stage device. The reflection coefficients generated by the input matching network and by the

interstage matching network are close to the desired values, and the conditions for oscillation are not satisfied anywhere in the simulated frequency range.

18. "TwoStgAmpInZ_TB" uses an S-Probe pair to calculate the reflection coefficients looking both directions at both the input and output planes of the first stage device. The reflection coefficients generated by the input matching network and by the interstage matching network are close to the desired values, and the conditions for oscillation are not satisfied anywhere in the simulated frequency range.
19. "TwoStgAmpOutZ_TB" is the same as "TwoStgAmpInZ_TB", except that the reflection coefficients at the input and output planes of the second stage device are simulated. The desired impedances are generated, and the conditions for oscillation are not satisfied within the simulated frequency range.
20. "TwoStgAmp_wTLsInZ_TB" is the same as "TwoStgAmpInZ_TB", except that the interconnects are modeled as transmission lines.
21. "Similarly, "TwoStgAmp_wTLsOutZ_TB" is like "TwoStgAmpOutZ_TB".
22. "TwoStgAmp_TB" simulates the gain, gain compression, PAE, and other nonlinear characteristics of the two-stage amplifier. It shows a 1-dB gain compression point of about 25 dBm, and a maximum output power of about 26.7 dBm.
23. "TwoStgAmp_wTLs_TB" is an identical simulation setup, except that all the subcircuits include transmission line effects. It shows a maximum output power of about 26.7 dBm and a 1-dB gain compression point of about 25.2 dBm, but this was only achieved after modifying component values to compensate for the interconnects modeled as transmission lines.
24. "BLC_LumpedIdeal" is an ideal branch-line coupler, implemented with lumped elements.
25. "BLC_LumpedIdeal_TB" simulates the "BLC_LumpedIdeal" S-parameters.
26. "BLC_Lumped" is a branch-line coupler implemented with lumped elements from the design kit.
27. "BLC_Lumped_TB" simulates the "BLC_Lumped" S-parameters.
28. "BranchLineCoupDiscOpt" is a discrete-value optimization of the branch-line couplers inductor parameter values, with transmission line effects included, to minimize insertion loss and optimize the phase difference between the two arms. With transmission line effects included, the optimizer finds that unequal inductor values in the series and shunt arms leads to more optimal performance.
29. "BLC_Lumped_wTLs" is the result of the optimization.
30. "BLC_LumpedBk_to_Bk_TB" simulates two optimized branch-line couplers, back-to-back. Ideally the insertion loss should be 0 dB, but the simulated insertion loss is about 1.4 dB. Increasing the drive power overcomes the loss of the input coupler but not the loss of the output coupler.
31. "BalancedLumpedAmp" is the amplifier without transmission line effects included.
32. "BalancedLumpedAmp_TB" simulates this amplifier. The maximum output power is about 29 dBm, and the output power at the 1-dB gain compression point is about 26.3 dBm.
33. "BalancedLumpedAmp_wTLs" is the final amplifier with transmission line effects included.
34. "BalLumpedAmp_wTLs_TB" simulates this amplifier. The maximum output power is 29.3 dBm and the output power at the 1-dB gain compression point is about 27 dBm.
35. "BalancedLumpedAmp_wTLs_SP_NF" shows the small-signal S-parameters and noise figure versus frequency. The amplifier has 27 dB of gain, from 9.6 GHz to 10.2 GHz, and a minimum noise figure of about 3 dB.

Simulation Results.


Output Spectrum, dBm
Transducer Power Gain, dB
Gain Compression between markers, dB
912.3m
Output Power at Marker m2, dBm
24.92
PAE, %
High Supply Current
Fund. Output Power, dBm
RF Power Selector
RFpower
Fundamental Frequency
10.00 GHz
Input and Output Voltage Waveforms
Fundamental and Third Harm., dBm

Available Source Power dBm	Fundamental Output Power dBm	Transducer Power Gain	Power-Added Efficiency, %	DC Power Consumpt. Watts	High Supply Current	Thermal Dissipation Watts
-15.000	12.830	27.830	2.058	0.931	0.186	0.912
-12.500	15.321	27.821	3.848	0.932	0.186	0.898
-10.000	17.797	27.797	6.437	0.934	0.187	0.874
-7.500	20.226	27.726	11.190	0.940	0.188	0.835
-5.000	22.531	27.531	18.705	0.956	0.191	0.777
-4.000	23.386	27.386	22.469	0.969	0.194	0.751
-3.000	24.184	27.184	26.468	0.988	0.198	0.726
-2.000	24.917	26.917	30.535	1.014	0.203	0.704
-1.000	25.591	26.591	34.652	1.040	0.209	0.681
0.000	26.161	26.161	38.461	1.072	0.214	0.659
1.000	26.514	25.514	40.856	1.093	0.219	0.646
2.000	26.731	24.731	42.220	1.112	0.223	0.641

Available Source Power dBm	Second Harmonic dBc	Third Harmonic dBc	Fourth Harmonic dBc	Fifth Harmonic dBc
-15.00	-42.84	-67.59	-79.28	-105.3
-12.50	-40.72	-62.70	-73.88	-106.4
-10.00	-38.66	-57.71	-68.93	-81.40
-7.500	-36.73	-51.61	-62.91	-69.91
-5.000	-35.88	-43.86	-54.34	-60.99
-4.000	-36.32	-40.58	-50.71	-57.79
-3.000	-37.59	-37.46	-47.43	-55.08
-2.000	-39.62	-34.74	-44.97	-53.22
-1.000	-40.67	-32.72	-43.77	-52.06
0.0000	-39.17	-31.26	-43.67	-50.18
1.000	-36.37	-30.06	-44.71	-48.71
2.000	-33.91	-28.83	-46.19	-48.84

Notes

- If you run the Design Rule Checker on the BalancedLumpedAmp design, the only errors you should get are that the widths of the MIM layers must be >= 4 um. This rule is violated by a small margin, by several small capacitors in the layout.

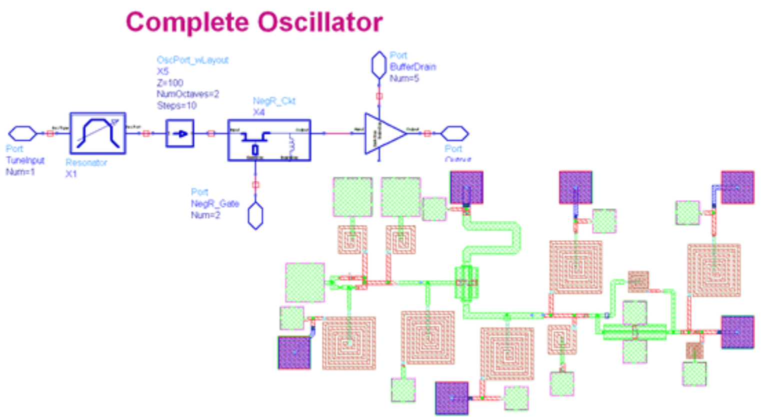
MMIC Oscillator

Location: $HPEESOF_DIR/examples/MW_Ckts/MMIC_Osc_wrk

Objective

This example shows the design of a 20 GHz, varactor-tuned oscillator. It consists of a resonant circuit, a grounded-gate, a negative-resistance generating circuit, and a buffer amplifier.The buffer amplifier presents a suitable reflection coefficient to the drain of the HEMT device to help generate a negative resistance at the source, and it amplifies the oscillating signal while providing a reasonable match to 50 Ohms at the output.This example includes an evaluation of the HEMT devices used in the oscillator and it uses elements from a "generic" design kit that was created to show some relatively simple examples of design kit elements. Design kits for real manufacturing processes would have many more elements, and the elements would have more detail, especially the transistors.

Figure 1: Complete Oscillator

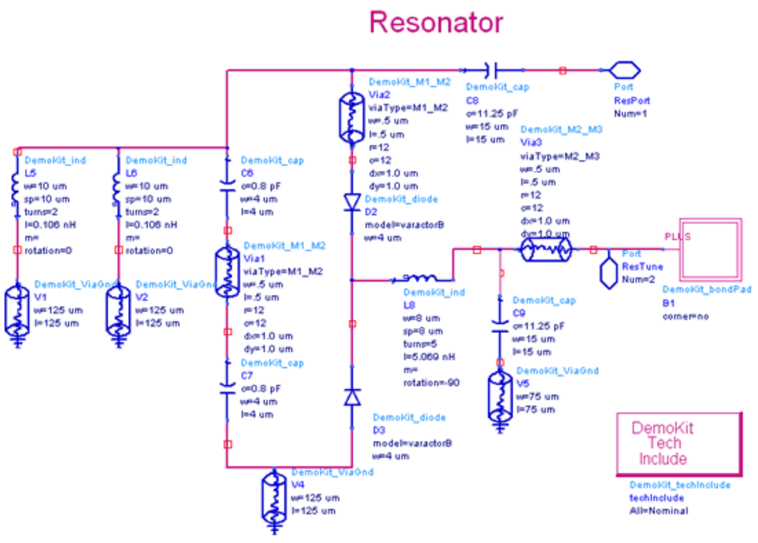


Setup

The bias point for the device is based on results from the FET_Gm_Calcs simulation in the examples/MW_Ckts/MMIC_Amp_wrk. It simulates the transconductance Gm of a single device, versus bias. This allows you to pick a bias point where the Gm is a maximum to maximize gain.

1. "NegR_TestSimple" simulates the reflection coefficient of the common-gate, negative-resistance generating circuit, as a function of frequency and the source inductance connected between the gate and ground. This circuit uses ideal components (except for the FET), which are useful for determining over what frequency range the circuit is capable of generating oscillations.
2. "NegR_vsLoadTest" simulates the reflection coefficient of the common-gate, negative-resistance generating circuit, as a function of the reflection coefficient presented to the drain of the FET. The reflection coefficient is generated by the buffer amp. This simulation uses design kit elements and a microstrip transmission line to replace the source inductor.
3. "NegR_vsAmplitude" simulates the impedance that the common-gate, negative-resistance generating circuit presents to the resonator, as a function of the signal amplitude and width of the gate transmission line.
4. "TransLineLequiv" compares the S-parameters of a microstrip transmission line with those of an ideal inductor.
5. "SP_NF_GainMatchK" is a design from the Amplifier DesignGuide, and is used to determine the optimal source and load impedances to present to the device in the buffer stage.
6. "BufferStageOpt" is an optimization of the input matching network of the buffer stage, such that the desired reflection coefficient looking into the buffer stage is generated. This reflection coefficient is determined from the NegR_vsLoadTest simulation.
7. "BufferStageSP_TB" is a simulation of the finished buffer amplifiers S-parameters. There is not much difference between these simulation results and the BufferStageOpt results, which included fewer parasitic elements.
8. "NegR_vsAmpFinal" simulates the impedance that the final version of the negative-resistance-generating circuit and buffer amp produce. The results are plotted along with the impedance of the final version of the resonator, indicating that the circuit should oscillate.
9. "ResTestSimple" is a simulation of a very simple resonator, consisting of ideal, lumped elements and two varactor diodes.
10. "Resonator" is the finished resonator.
11. "Resonator_TB" is a simulation of the finished resonator, versus varactor bias voltage. This simulation indicates that this resonator is significantly lossier than the more ideal one simulated in ResTestSimple.
12. "DesignKitC_TB" compares a DemoKit capacitor with an ideal capacitor to determine the effects of parasitics.
13. "DesignKitL_TB" compares a DemoKit inductor with an ideal inductor.
14. "Varactor_Test" simulates the varactor diode capacitance versus bias voltage.
15. "Via_TB" determines the equivalent inductance of a via through the substrate.
16. "CompleteOsc_TB" simulates the complete (finished) oscillator, including tuning characteristic, output power, and phase noise.
17. "CompleteOsc" is the complete (finished) oscillator.

Figure 2: Resonator


Resonator
DemoKit_ind L5 w=10 um sp=10 um turns=2 l=0.106 nH m= rotation=0
DemoKit_ind L6 w=10 um sp=10 um turns=2 l=0.106 nH m= rotation=0
DemoKit_ViaGnd V1 w=125 um l=125 um
DemoKit_ViaGnd V2 w=125 um l=125 um
DemoKit_cap C6 c=0.8 pF w=4 um l=4 um
DemoKit_M1_M2 Via1 viaType=M1_M2 w=.5 um l=.5 um r=12 c=12 dx=1.0 um dy=1.0 um
DemoKit_cap C7 c=0.8 pF w=4 um l=4 um
DemoKit_ViaGnd V4 w=125 um l=125 um
DemoKit_M1_M2 Via2 viaType=M1_M2 w=.5 um l=.5 um r=12 c=12 dx=1.0 um dy=1.0 um
DemoKit_diode D2 model=varactorB w=4 um
DemoKit_diode D3 model=varactorB w=4 um
DemoKit_cap C8 c=11.25 pF w=15 um l=15 um
Port ResPort Num=1
DemoKit_M2_M3 Via3 viaType=M2_M3 w=.5 um l=.5 um r=12 c=12 dx=1.0 um dy=1.0 um
DemoKit_ind L8 w=8 um sp=8 um turns=5 l=5.069 nH m= rotation=-90
DemoKit_cap C9 c=11.25 pF w=15 um l=15 um
DemoKit_ViaGnd V5 w=75 um l=75 um
Port ResTune Num=2
PLUS
DemoKit_bondPad B1 corner=no
DemoKit Tech Include
DemoKit_techInclude techInclude All=Nominal

Notes

- If you run the Design Rule Checker on the CompleteOsc design, the only errors you should get are that the widths of the MIM layers must be >= 4 um. This rule is violated by a small margin, by several small capacitors in the layout.

Optimizing A Linear FET Model to Match Measured S-Parameters

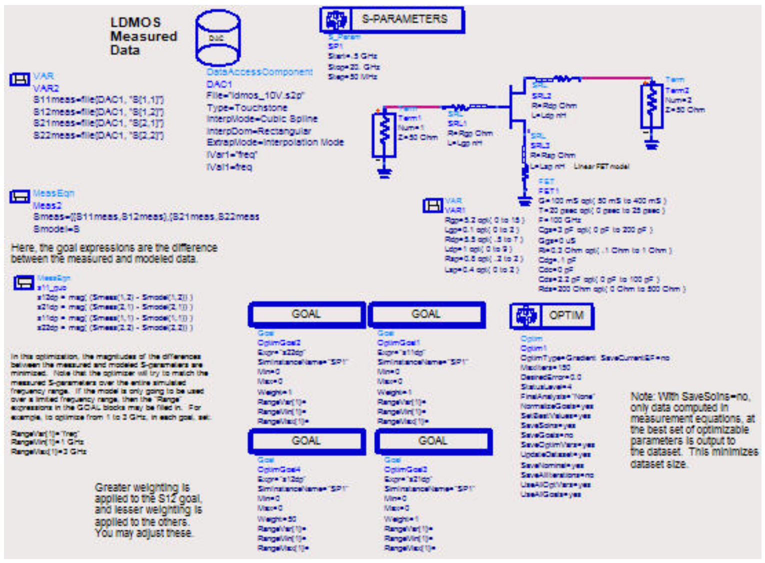
Location: $HPEESOF_DIR/examples/MW_Ckts/AmodelB_wrk

Objective

This example shows how to optimize the model parameters of a linear FET to match measured S-parameter data. A similar example can be found at TestLab_HOWTO_wrk under the "Tutorial" example directory.

Setup

1. Design file "BeforeOpt" shows the poor match between measured and modeled S-parameters before optimization.
2. Design file "AfterOpt" use "Gradient optimization" and "S Parameters" simulation to optimize the match. In this example, the magnitude and phase differences between the corresponding S-parameters are set as optimization goals.
3. The design can be further optimized to get a better match, however it is left for the users.


LDMOS Measured Data
DAC
S-PARAMETERS
S_Param
SP1
Start=.5 GHz
Stop=20. GHz
Step=50 MHz
VAR
VAR2
S11meas=file{DAC1, "S[1,1]"}
S12meas=file{DAC1, "S[1,2]"}
S21meas=file{DAC1, "S[2,1]"}
S22meas=file{DAC1, "S[2,2]"}
DataAccessComponent
DAC1
File="ldmos_10V.s2p"
Type=Touchstone
InterpMode=Cubic Spline
InterpDom=Rectangular
ExtrapMode=Interpolation Mode
iVar1="freq"
iVal1=freq
Term
Term1
Num=1
Z=50 Ohm
SRL
SRL1
R=Rgp Ohm
L=Lgp nH
SRL
SRL2
R=Rdp Ohm
L=Ldp nH
SRL
SRL3
R=Rsp Ohm
L=Lsp nH
Linear FET model
Term
Term2
Num=2
Z=50 Ohm
MeasEqn
Meas2
Smeas={{S11meas,S12meas},{S21meas,S22meas
Smodel=S
Here, the goal expressions are the difference between the measured and modeled data.
MeasEqn
s11_pub
s12dp = mag((Smeas(1,2) - Smodel(1,2)))
s21dp = mag((Smeas(2,1) - Smodel(2,1)))
s11dp = mag((Smeas(1,1) - Smodel(1,1)))
s22dp = mag((Smeas(2,2) - Smodel(2,2)))
VAR
VAR1
Rgp=5.2 opt{ 0 to 15 }
Lgp=0.1 opt{ 0 to 2 }
Rdp=5.5 opt{ .5 to 7 }
Ldp=1 opt{ 0 to 9 }
Rsp=0.5 opt{ .2 to 2 }
Lsp=0.4 opt{ 0 to 2 }
FET
FET1
G=100 mS opt{ 50 mS to 400 mS }
T=20 psec opt{ 0 psec to 25 psec }
F=100 GHz
Cgs=3 pF opt{ 0 pF to 200 pF }
Ggs=0 uS
Ri=0.2 Ohm opt{ .1 Ohm to 1 Ohm }
Cdg=.1 pF
Cdc=0 pF
Cds=2.2 pF opt{ 0 pF to 100 pF }
Rds=200 Ohm opt{ 0 Ohm to 500 Ohm }
In this optimization, the magnitudes of the differences between the measured and modeled S-parameters are minimized. Note that the optimizer will try to match the measured S-parameters over the entire simulated frequency range. If the model is only going to be used over a limited frequency range, then the "Range" expressions in the GOAL blocks may be filled in. For example, to optimize from 1 to 3 GHz, in each goal, set:
RangeVar[1]="freq"
RangeMin[1]=1 GHz
RangeMax[1]=3 GHz
GOAL
Goal
OptimGoal2
Expr="s22dp"
SimInstanceName="SP1"
Min=0
Max=0
Weight=1
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=
GOAL
Goal
OptimGoal1
Expr="s11dp"
SimInstanceName="SP1"
Min=0
Max=0
Weight=1
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=
GOAL
Goal
OptimGoal4
Expr="s12dp"
SimInstanceName="SP1"
Min=0
Max=0
Weight=50
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=
GOAL
Goal
OptimGoal3
Expr="s21dp"
SimInstanceName="SP1"
Min=0
Max=0
Weight=1
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=
Greater weighting is applied to the S12 goal, and lesser weighting is applied to the others. You may adjust these.
OPTIM
Optim
Optim1
OptimType=Gradient SaveCurrentEF=no
MaxIters=150
DesiredError=0.0
StatusLevel=4
FinalAnalysis="None"
NormalizeGoals=yes
SetBestValues=yes
SaveSolns=yes
SaveGoals=no
SaveOptimVars=yes
UpdateDataset=yes
SaveNominal=yes
SaveAllIterations=no
UseAllOptVars=yes
UseAllGoals=yes
Note: With SaveSolns=no, only data computed in measurement equations, at the best set of optimizable parameters is output to the dataset. This minimizes dataset size.

Analysis

Figure 1: S-parameter comparison before optimization

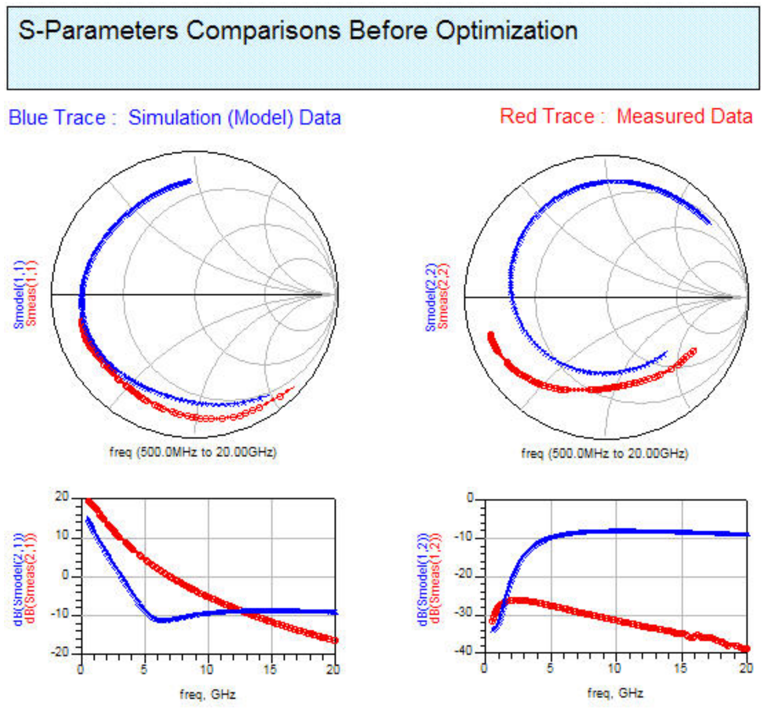
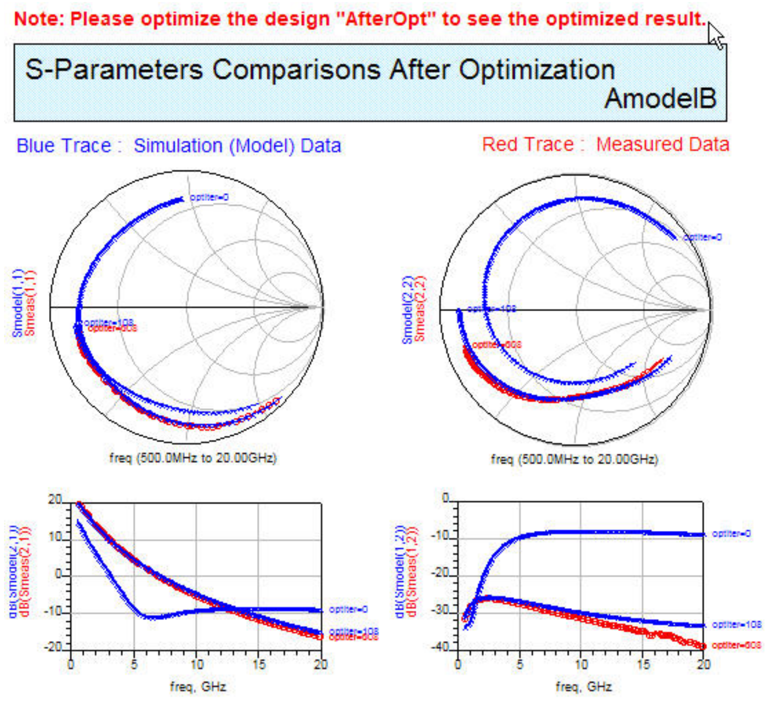


Figure 2: S-parameter comparison after optimization



Notes

- In the "Nominal Optimization" controller, set "SaveSolns" equal to "no" will output only the best set of parameters to the dataset, this minimizes the dataset size.
- If all goals are active, there is no need to specify any GoalNames in the "Nominal Optimization" controller.
- The "Range" expressions in the "GOAL" blocks can be used to specify the desired frequency range of the optimized model.

Statistical Design of an X-Band LNA

Location: $HPEESOF_DIR/examples/MW_Ckts/Design_Manufacturing_MMIC_wrk

Objective

Process variations could turn a seemingly great MMIC design into a big failure. This example demonstrates a new design process for microwave circuits using advanced statistical methods that result in robust designs with high overall manufacturing yield.
A thorough step-by-step design process of an X-band MMIC Low Noise Amplifier is used here to illustrate this new design methodology. Some of the techniques described include programmable optimization, sensitivity analysis, design centering or yield optimization, and design of experiments. Final simulations predict that the design should be insensitive to process variation and will have a high yield.

Setup

Design file descriptions:

1. A1_X-band_LNAInitial design of LNA with initial simulation results.
2. A2_X-band_LNA_sensitivityPerforming initial sensitivity analysis of the specs versus component values. Be aware that this analysis is localized and changes one component at a time.
3. A3_X-band_LNA_prog_optimizedThis design file illustrates the Programmable Optimization feature in ADS. First we optimize the Noise Figure using only the Input Matching Network components, followed by a second Optimization for Gain and Return loss using the rest of the components. These optimization and analysis steps are done automatically in one run.
4. A4_X-band_LNA_initial_Yield_analysisHere we run an initial Yield analysis after having optimized the design and met its nominal specs. Initial Yield is shown to be around 40% (Not good). Next, we will use Statistical tools in ADS in order to pinpoint the problem in the design that is causing this low yield. Then we fix this problem and bring the design to be robust with close to 100% yield.
5. A5_X-band_LNA_DOEDesign of Experiments (DOE) analysis of the LNA with three variables track the sensitivity and interactions of the three matching networks (Input, Interstage, and Output matching networks) shows that the yield problem is due to a sensitive Output Matching Network. Therefore the Output Matching Network needs to be redesigned.
6. A6_X-band_LNA_Output_redesignSensitivity analysis, Yield analysis, Yield Sensitivity Histograms analysis*, and DOE analysis all pointed out that the Output Matching Network is very sensitive. This design now includes a new redesigned (less sensitive) output matching network.
7. A6_X-band_LNA_Output_redesign_YieldRepeating the Yield analysis with the new redesigned output matching network shows higher Yield (about 80%), but needs to be design centered for higher optimized yield. Yield Optimization (or Design Centering) is used next.
8. A7_X-band_LNA_Design_centering_finalThis file uses the Design Centering or Yield Optimization technique to adjust the component values in order to meet all specs and maximize the overall yield of the design.
9. A8_X-band_LNA_DOE2This second DOE analysis on the redesigned LNA is included here in order to show that the output matching network is no longer a contributor to the Yield problem. In fact, the analysis now shows that the interstage matching network is the next main contributor to the S22 variation.
   It is up to the designer to decide whether or not to spend time to create a less sensitive Interstage network or not. Of course it is highly recommended in order to produce a highly Robust circuit with a first pass success.
10. A9_X-band_LNA_Final_designFinal design shows that the Yield has been improved to over 97% and the design is robust and insensitive to any component or process variation. The reason the yield is not 100% is due to the few weak or bad FETs that are included in the set of 42 FETs used.

Notes

From the Data Display, the file "Sens_histogram" can be used with any Yield Simulation run to plot the Yield sensitivity of the LNA with respect to any chosen component with its tolerance changes.
Open the file and you can visualize the yield sensitivity with respect to each element in the design.
This is a very useful tool that helps designers understand their designs and pinpoint the sensitive elements that could be causing the Yield problem in the design.

Test Lab for Two Stage Amplifier Design

Location: $HPEESOF_DIR/examples/MW_Ckts/TestLabForTwoStageAmp_wrk

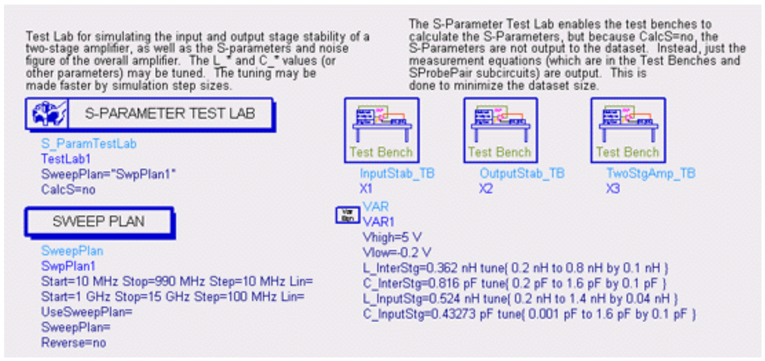
Objective

This example shows the use of a TestLab, TestBenches, and SProbes while tuning component values in a two-stage amplifier design. The objective is to maintain stability, by checking the stability factors at both the first and second stage devices, while maximizing the overall gain and minimizing the noise figure of the amplifier.

Setup

1. "TwoStgAmp_TL" is the top-level schematic, that has an S-Parameter Test Lab. This schematic has three test benches as well as several component variables for tuning.
2. "InputStab_TB" is a test bench with an SProbePair for simulating the stability of the first stage. The SProbePair measures the reflection coefficients (and impedances) presentedto the input and output of the first stage FET, as well as the reflection coefficients (andimpedances) looking into the input of the FET and looking into the output. Based on these reflection coefficients, a stability factor is computed.
3. "OutputStab_TB" is identical to the InputStab_TB, except that it is set up to measure the stability factors and reflection coefficients associated with the second stage FET.
4. "TwoStgAmp_TB" is the same as the other two test benches, except that it is used for measuring the S-parameters and noise figure of the overall amplifier. It does not have any SProbes.

Top level test lab simulation setup.

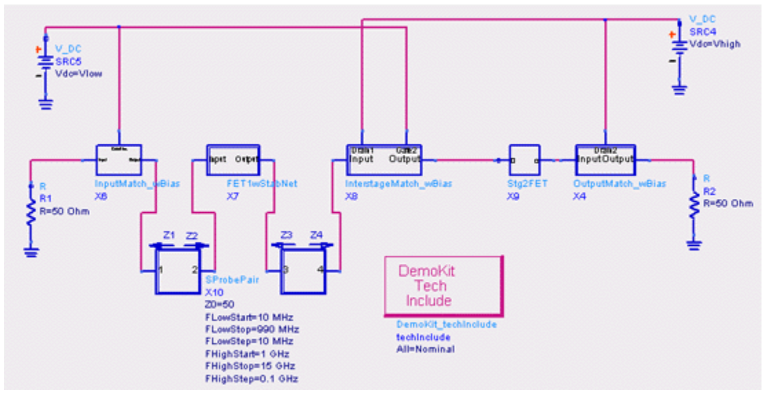


Test Lab for simulating the input and output stage stability of a two-stage amplifier, as well as the S-parameters and noise figure of the overall amplifier. The L_* and C_* values (or other parameters) may be tuned. The tuning may be made faster by simulation step sizes.

The S-Parameter Test Lab enables the test benches to calculate the S-Parameters, but because CalcS=no, the S-Parameters are not output to the dataset. Instead, just the measurement equations (which are in the Test Benches and SProbePair subcircuits) are output. This is done to minimize the dataset size.

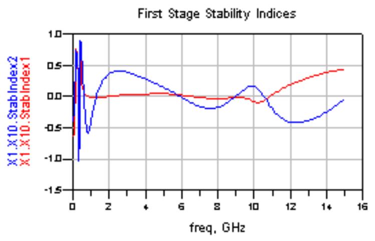
```
S-PARAMETER TEST LAB
S_ParamTestLab
TestLab1
SweepPlan="SwpPlan1"
CalcS=no

SWEEP PLAN
SweepPlan
SwpPlan1
Start=10 MHz Stop=990 MHz Step=10 MHz Lin=
Start=1 GHz Stop=15 GHz Step=100 MHz Lin=
UseSweepPlan=
SweepPlan=
Reverse=no

Test Bench        Test Bench         Test Bench
InputStab_TB      OutputStab_TB      TwoStgAmp_TB
X1                X2                 X3

VAR
VAR1
Vhigh=5 V
Vlow=-0.2 V
L_InterStg=0.362 nH tune{ 0.2 nH to 0.8 nH by 0.1 nH }
C_InterStg=0.816 pF tune{ 0.2 pF to 1.6 pF by 0.1 pF }
L_InputStg=0.524 nH tune{ 0.2 nH to 1.4 nH by 0.04 nH }
C_InputStg=0.43273 pF tune{ 0.001 pF to 1.6 pF by 0.1 pF }
```

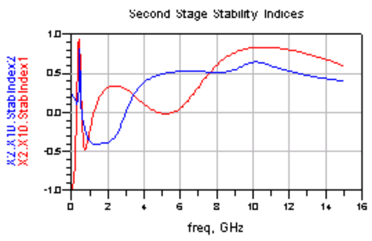
Setup for simulating the stability of the first stage device.

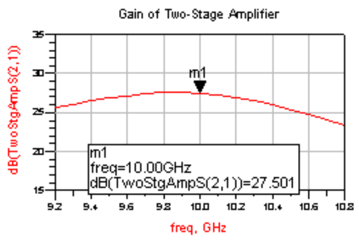


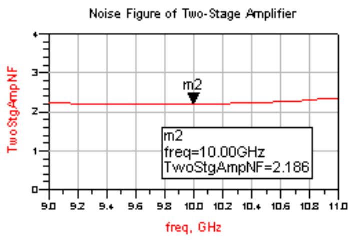
Results from the test lab simulation

StabIndex1 =real(Gamma1*Gamma2) and
StabIndex2 =real(Gamma3*Gamma4). If these two
are always <1, then the conditions for oscillation are
never satisfied.

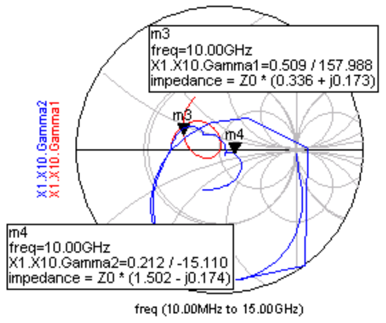




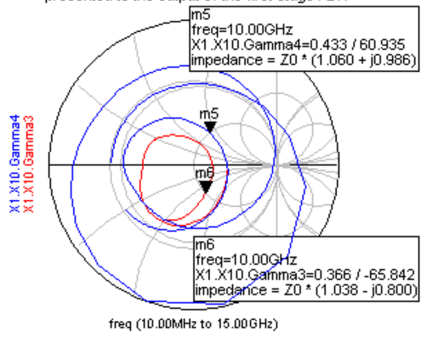


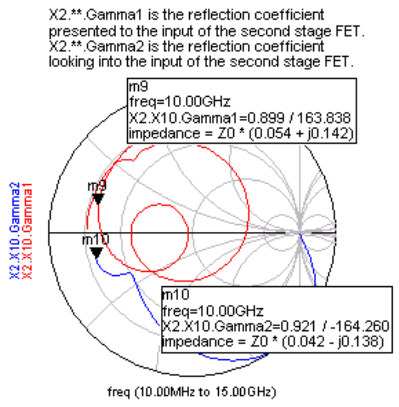


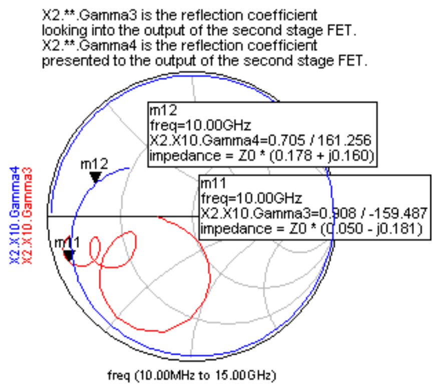
X1.**.Gamma1 is the reflection coefficient
presented to the input of the first stage FET.
X1.**.Gamma2 is the reflection coefficient
looking into the input of the first stage FET.



X1.**.Gamma3 is the reflection coefficient
looking into the output of the first stage FET.
X1.**.Gamma4 is the reflection coefficient
presented to the output of the first stage FET.







Using the Design Rule Checker (DRC)

Location: $HPEESOF_DIR/examples/MW_Ckts/drc_via_wrk

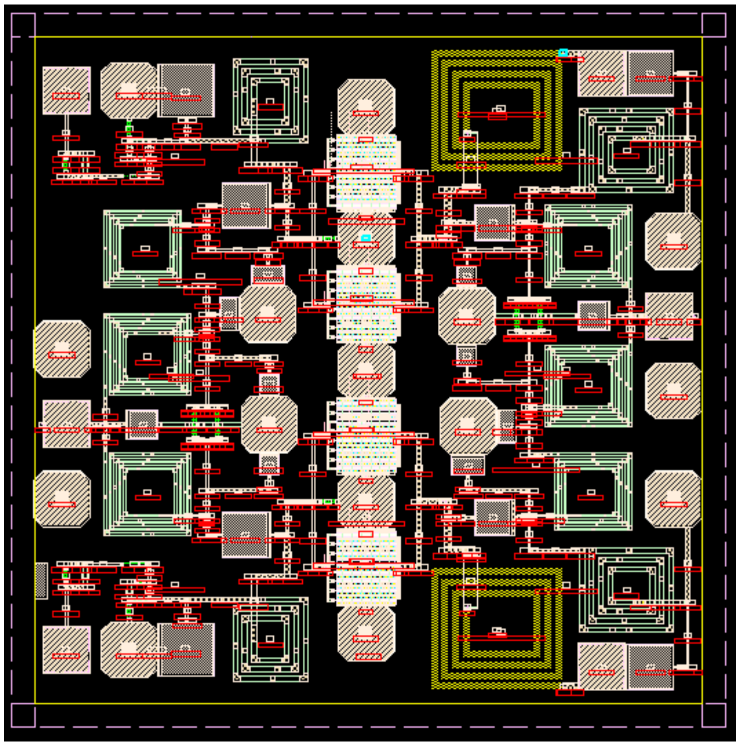
Objective

This example illustrates the use of the Design Rule Checker (DRC) in a power amplifier layout.

Setup

1. The "subvia" layout shows two substrate vias that are closely spaced and violate the 150um spacing rule. This can be checked with the Design Rule "subvia.ael", or it can be checked with the DRC dialog by checking the minimum spacing of the BacksideVia process layer.
2. The second example is the "pwramp". The substrate vias on it have a design violation. Run the Design Rule "subgate.ael" to view the error. If you examine the Application Extension Language (AEL) file, it illustrates the use of derived layers, meaning that you do not check the process layer directly, but first "derive" a logical subset of structures to check with the DRC.

Power Amplifier layout



Using SP_Probes to design a simple, two-stage LNA

Location: $HPEESOF_DIR/examples/MW_Ckts/Simple_LNA_wSP_Probes_wrk

This example shows an 11.5-12.5 GHz LNA designed using SP_Probes. These probes allow you to see impedances, reflection coefficients, and noise parameters looking to their left and right without loading the network. This allows you to see whether the various impedance matching networks are generating the desired impedances at the input and output of each device. This design example is meant to be simple and easy to understand, but is not complete, since the various impedance matching and stabilization networks use ideal elements.

The design procedure used was as follows:

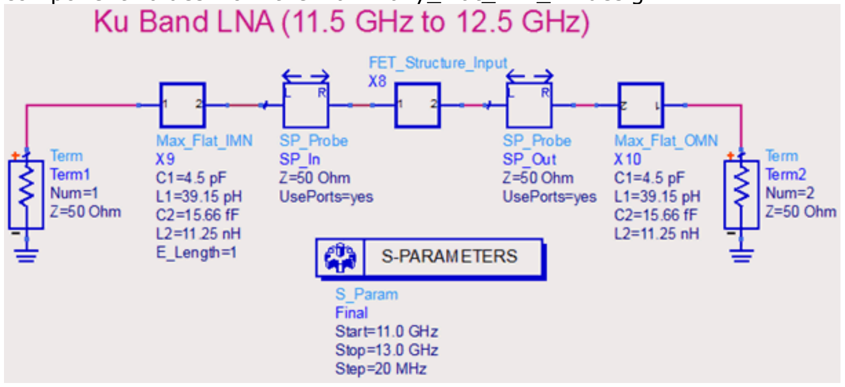
1. Evaluate the performance of the FET versus DC bias point.
2. Design/optimize feedback network element values to attain/improve stability. This may mean degradation in noise figure and gain performance.
3. Design prototype filters to be used as initial starting points for the impedance matching networks.
4. Design the first stage. This consists of optimizing the input matching network to improve noise figure (presenting Sopt to the active device) and then optimizing the output matching network to improve the gain.
5. Design the second stage using the same procedure to design the first stage.
6. Both the output matching network of the first stage and the input matching network of the second stage have been designed for 50-Ohm output and input terminations, respectively. Therefore, these output and input matching networks may be connected directly to each other to create the interstage matching network.
7. Assuming the performance of the combined amplifier could be improved further by adjusting the various matching networks, run a final optimization as follows:
   - Optimize the input matching network to provide a noise match (Sopt) to the first stage to improve the noise figure.
   - Optimize the interstage matching network to maximize gain of the first stage and minimize the noise figure of the overall amplifier (effectively providing a noise match to the second stage.)
   - Optimize the output matching network of the second stage to maximize the gain of the overall amplifier.

The Device Evaluation folder contains a FET_SP_NF_Match_Circ schematic and corresponding data display for evaluating the performance (S-Parameters, noise figure, stability, maximum available gain, etc.) of a FET as a function of its DC bias point. This is from the **Amplifier DesignGuide**. This could be useful for determining a good DC bias point and determining whether a device might be able to produce the gain and noise figure you need.

The FET_SP_NF_Match_Circ schematic indicated that the device is potentially unstable. The Device Stabilization folder contains two schematics and corresponding data displays for optimizing feedback network component values to attain stability. Gain_and_Stab_opt_First_Stage shows the optimization of the gain, noise figure, and stability of the first stage device. The corresponding data display shows reasonable gain, NFmin (minimum noise figure), and stability. Gain_and_Stab_opt_Second_Stage shows an optimization of the gain and stability (but not noise) of the second stage device, which is twice as large as the first stage device. For the second stage, the gain goal is emphasized at the expense of the noise figure.

The Filter Simulations folder contains schematics and corresponding data displays of several lumped-element, low-pass and band-pass filters. The Maximally_Flat_BPF_n2 is the starting point for the impedance-matching networks in the two-stage amplifier design. The simplest impedance-matching network has just an L and C, but a simple band-pass filter structure should enable you to achieve better performance. These filters are from tables in Matthaei, Young, and Jones, and could be adjusted easily to operate in different frequency bands or have different frequency response characteristics.

The LNA 1st Stage Design folder has the design and optimization of the first stage. This is the First_Stage_wSP_Probes schematic. Its input and output matching networks use component values from the Maximally_Flat_BPF_n2 design.


Ku Band LNA (11.5 GHz to 12.5 GHz)

Term
Term1
Num=1
Z=50 Ohm

Max_Flat_IMN
X9
C1=4.5 pF
L1=39.15 pH
C2=15.66 fF
L2=11.25 nH
E_Length=1

SP_Probe
SP_In
Z=50 Ohm
UsePorts=yes

FET_Structure_Input
X8

SP_Probe
SP_Out
Z=50 Ohm
UsePorts=yes

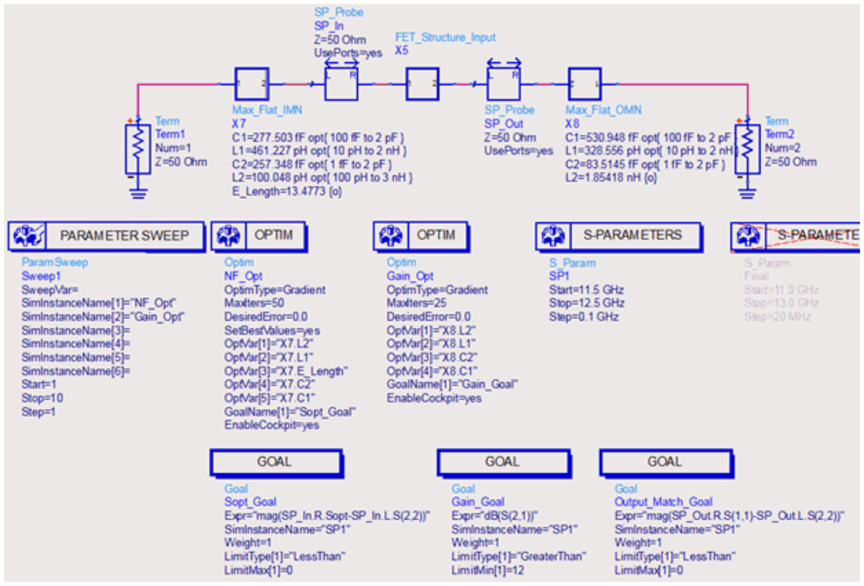
Max_Flat_OMN
X10
C1=4.5 pF
L1=39.15 pH
C2=15.66 fF
L2=11.25 nH

Term
Term2
Num=2
Z=50 Ohm

S-PARAMETERS
S_Param
Final
Start=11.0 GHz
Stop=13.0 GHz
Step=20 MHz

This shows the First_Stage_wSP_Probes_Opt schematic for optimizing the input and output impedance matching networks. This is a sequential optimization, with the sequence determined by the Parameter Sweep:

1. The parameters in the input matching network are optimized such that Sopt is presented to the input of FET_Structure_Input.
2. The parameters in the output matching network are optimized such that either:
    - a complex conjugate match occurs at the output of FET_Structure_Input or
    - the gain of the overall amplifier is maximized. Optimizing for the gain of the overall amplifier appears to give better results.


SP_Probe
SP_In
Z=50 Ohm
UsePorts=yes

FET_Structure_Input
X5

Term
Term1
Num=1
Z=50 Ohm

Max_Flat_IMN
X7
C1=277.503 fF opt{ 100 fF to 2 pF }
L1=461.227 pH opt{ 10 pH to 2 nH }
C2=257.348 fF opt{ 1 fF to 2 pF }
L2=100.048 pH opt{ 100 pH to 3 nH }
E_Length=13.4773 {o}

SP_Probe
SP_Out
Z=50 Ohm
UsePorts=yes

Max_Flat_OMN
X8
C1=530.948 fF opt{ 100 fF to 2 pF }
L1=328.556 pH opt{ 10 pH to 2 nH }
C2=83.5145 fF opt{ 1 fF to 2 pF }
L2=1.85418 nH {o}

Term
Term2
Num=2
Z=50 Ohm

PARAMETER SWEEP
ParamSweep
Sweep1
SweepVar=
SimInstanceName[1]="NF_Opt"
SimInstanceName[2]="Gain_Opt"
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1
Stop=10
Step=1

OPTIM
Optim
NF_Opt
OptimType=Gradient
MaxIters=50
DesiredError=0.0
SetBestValues=yes
OptVar[1]="X7.L2"
OptVar[2]="X7.L1"
OptVar[3]="X7.E_Length"
OptVar[4]="X7.C2"
OptVar[5]="X7.C1"
GoalName[1]="Sopt_Goal"
EnableCockpit=yes

OPTIM
Optim
Gain_Opt
OptimType=Gradient
MaxIters=25
DesiredError=0.0
OptVar[1]="X8.L2"
OptVar[2]="X8.L1"
OptVar[3]="X8.C2"
OptVar[4]="X8.C1"
GoalName[1]="Gain_Goal"
EnableCockpit=yes

S-PARAMETERS
S_Param
SP1
Start=11.5 GHz
Stop=12.5 GHz
Step=0.1 GHz

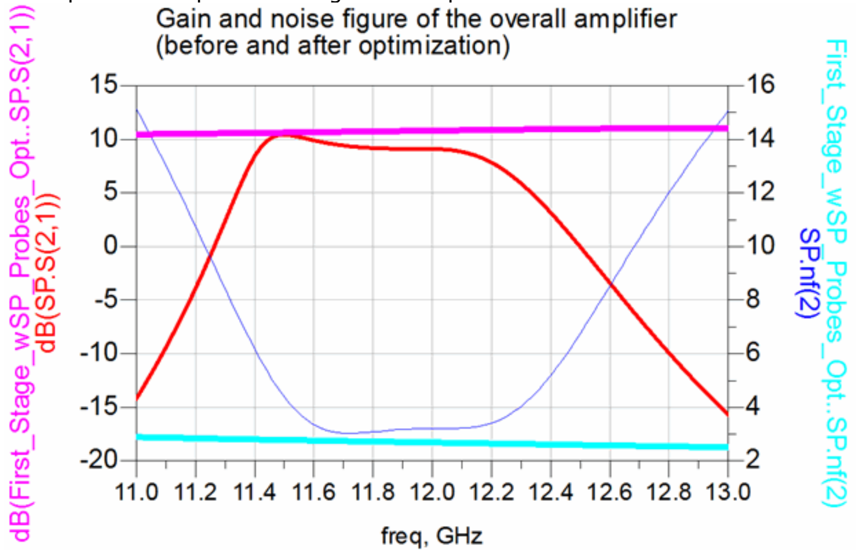
S-PARAMETE
S_Param
Final
Start=11.0 GHz
Stop=13.0 GHz
Step=20 MHz

GOAL
Goal
Sopt_Goal
Expr="mag(SP_In.R.Sopt-SP_In.L.S(2,2))"
SimInstanceName="SP1"
Weight=1
LimitType[1]="LessThan"
LimitMax[1]=0

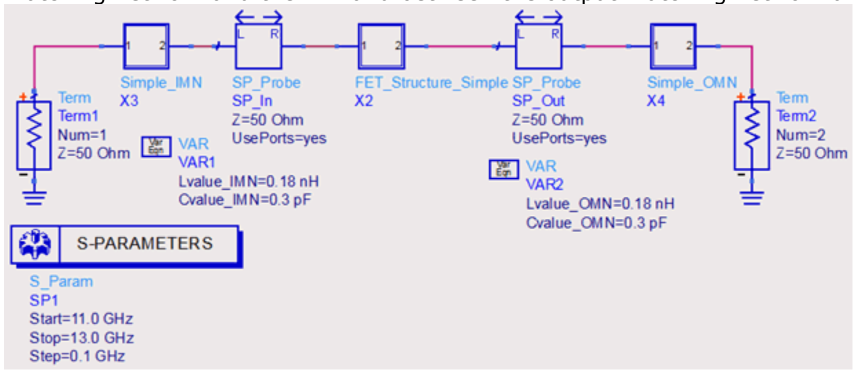
GOAL
Goal
Gain_Goal
Expr="dB(S(2,1))"
SimInstanceName="SP1"
Weight=1
LimitType[1]="GreaterThan"
LimitMin[1]=12

GOAL
Goal
Output_Match_Goal
Expr="mag(SP_Out.R.S(1,1)-SP_Out.L.S(2,2))"
SimInstanceName="SP1"
Weight=1
LimitType[1]="LessThan"
LimitMax[1]=0

This shows the gain and noise figure of the overall amplifier before and after optimizing the input and output matching network parameter values.



The Potentially Unstable 1-Stage LNA folder contains an amplifier design that is potentially unstable. The Simple_LNA_wSP_Probes schematic has an input matching network (Simple_IMN), a FET with bias and feedback (FET_Structure_Simple) to improve stability and an output matching network (Simple_OMN). There are SP_Probes between the input matching network and the FET and between the output matching network and the FET.

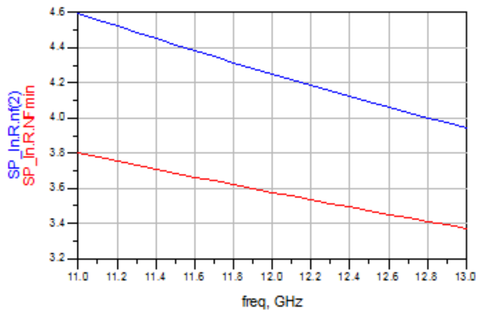


We use the SP_In probe to see what Sopt is for the network looking to its right, and adjust the L and C values in Simple_IMN such that its S22 (looking to the left from SP_In) is approximately the same as Sopt. This should produce the best noise figure of the overall amplifier.

After designing the input matching network, ideally, we should design the output matching network such that it produces the complex conjugate of the reflection coefficient seen looking into the output of FET_Structure_Simple. SP_Out is used to see whether this complex conjugate impedance matching is achieved or not. It also can be used to produce the load stability circles of the network looking to its left (Simple_IMN and FET_Structure_Simple) so we can avoid generating a load impedance that is in the unstable region.

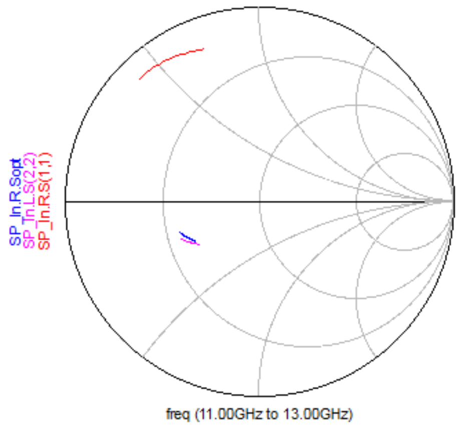
The next plot shows that NFmin of the network looking to the right of SP_In is better than nf(2), which is the noise figure of the network in a 50-Ohm system. This means that (as expected) you can improve the noise figure by using an impedance matching network to generate Sopt.

This shows the noise figure and NFmin of the network between SP_In and Term2. The difference here indicates that the Input Matching Network should present an impedance other than 50 Ohms to the FET to attain a noise figure closer to NFmin.



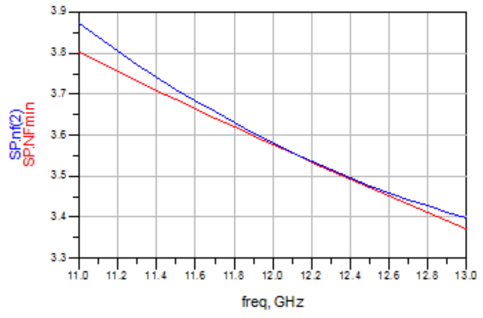
This Smith Chart shows that the input matching network generates Sopt, so the noise figure of the overall amplifier will be minimized.

For best noise figure, SP_In.R.Sopt and SP_In.L.S(2,2) should be the same. For best small-signal gain, you want SP_In.R.S(1,1) = conj(SP_In.L.S(2,2)). However, the impedance for minimum noise figure and the impedance for conjugate matching are different. If you change the Cvalue_IMN and Lvalue_IMN, the noise figure will get worse.

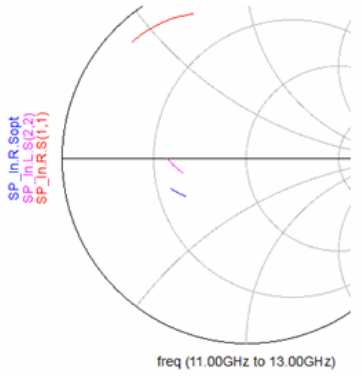


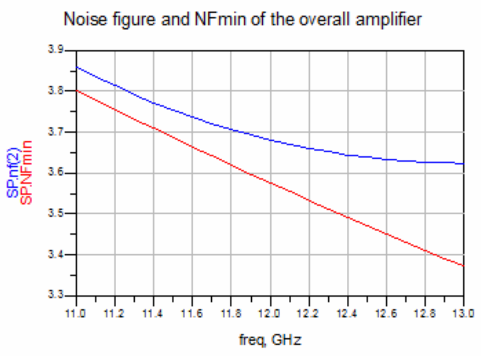
This plot shows that the noise figure of the overall amplifier is very close to NFmin.

Noise figure and NFmin of the overall amplifier

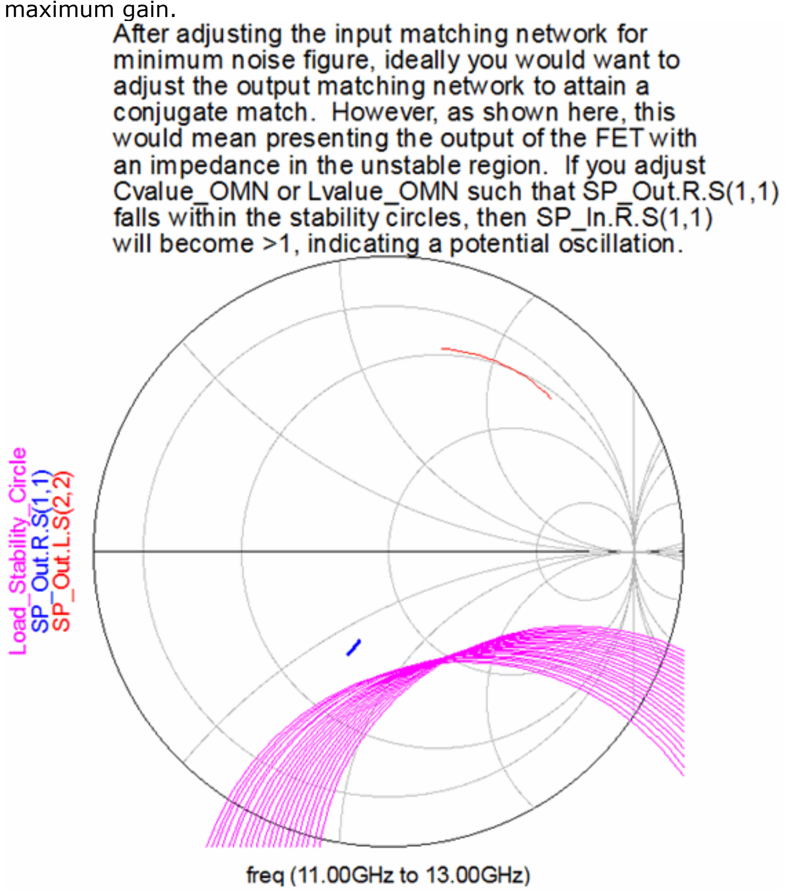


If we change the L value in the Simple_IMN from 0.18 nH to 0.3 nH, the Sopt reflection coefficient is no longer generated and the noise figure gets worse.

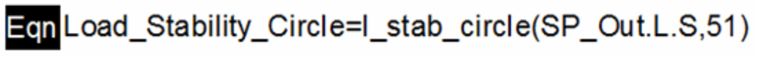


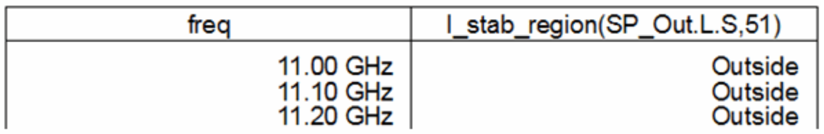


With the input matched for minimum noise figure, we now want to match the output for maximum gain.


After adjusting the input matching network for minimum noise figure, ideally you would want to adjust the output matching network to attain a conjugate match. However, as shown here, this would mean presenting the output of the FET with an impedance in the unstable region. If you adjust Cvalue_OMN or Lvalue_OMN such that SP_Out.R.S(1,1) falls within the stability circles, then SP_In.R.S(1,1) will become >1, indicating a potential oscillation.

The load stability circles and the stable region are computed from these expressions.


Eqn Load_Stability_Circle=l_stab_circle(SP_Out.L.S,51)



freq	l_stab_region(SP_Out.L.S,51)
11.00 GHz	Outside
11.10 GHz	Outside
11.20 GHz	Outside

Yield Sensitivity Histogram Design Templates

Location: $HPEESOF_DIR/examples/MW_Ckts/Ku_Band_LNA_DFM_wrk

Objective

This example explains how to use the Yield Sensitivity Histogram (YSH) templates provided in ADS to support Design for Manufacturing (DFM). DFM techniques yield higher performance and more consistent designs. A design that uses DFM techniques is not as sensitive to process variation and results in first-pass success with much higher yield.

The YSH templates have been prepared so you can automatically apply them to any of your designs with a minimum amount of set-up to quickly validate your designs for manufacturing with high yield.

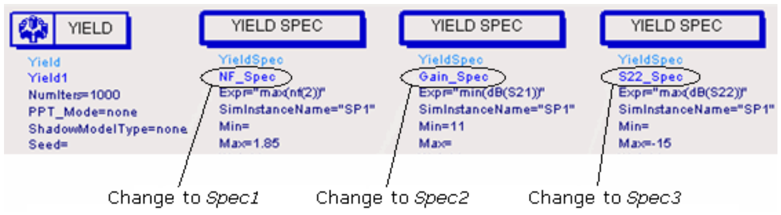
The example workspace *Ku_Band_LNA_DFM_wrk* demonstrates how to use these templates with real designs. Though the example workspace also covers Design of Experiments (DOE) and Sensitivity Analysis, these topics are not discussed here. For details about the workspace, see the example documentation *Design for Manufacturing Example Using Yield Sensitivity Histograms, DOE, and Sensitivity Analysis* (examples).

Setup

To use this design tool efficiently, use the following procedure when investigating the yield in your design:

1. Run Monte Carlo yield analysis on your design. On the schematic page, label each *Yield Spec* component as *Spec1*, *Spec2*, and *Spec3*. In other words, if one of your design's Yield Specs is *Noise Figure*, label it as *Spec1* instead of *NF*. This is required since the YSH template described here is generalized for use with any design with different specs by using the labels *Spec1*, *Spec2*, and *Spec3*.

   For example, in a typical yield analysis setup in ADS shown in the following figure, the *Yield Spec* components are labeled as, *NF_Spec*, *Gain_Spec*, and *S22_Spec*. To use the YSH template, these names must be changed to *Spec1*, *Spec2*, and *Spec3*.



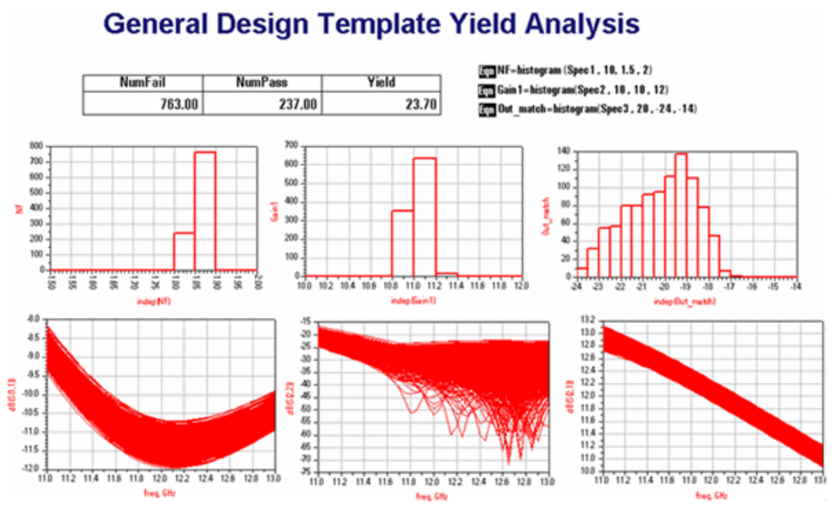
```
YIELD                 YIELD SPEC                 YIELD SPEC                   YIELD SPEC
Yield                 YieldSpec                  YieldSpec                    YieldSpec
Yield1                NF_Spec                    Gain_Spec                    S22_Spec
NumIters=1000         Expr="max(nf(2))"          Expr="min(dB(S21))"          Expr="max(dB(S22))"
PPT_Mode=none         SimInstanceName="SP1"      SimInstanceName="SP1"        SimInstanceName="SP1"
ShadowModelType=none  Min=                       Min=11                       Min=
Seed=                 Max=1.85                   Max=                         Max=-15
```

   Next, revise the value for *Expr=* (on the third line). It is important to write yield expressions using the *min* and *max* functions so the expressions use only one data point in every iteration. In the previous figure, the expression for the *S22_Spec* is *Expr="max(dB(S22))"* and the *Max* parameter is set to -15. Therefore, anything greater than -15 dB results in a *Fail* and anything less than or equal to -15 dB will be a *Pass*.

   At this point, run the yield analysis.

   Note
   You are free to use any expressions you prefer when setting up and running a yield analysis; however, it is recommended that you to use *min* and *max* for compatibility with the YSH templates.

2. Use the *General Design Template for Yield* provided with the YSH templates to display the yield results. You can customize it to your type of design and Specs by changing the Specs names on the template. The following figure shows a sample template for NF, Gain, and S22. If your design *Spec1* happens to be *Pout*, then change the *NF* in the first equation to *Pout*.



General Design Template Yield Analysis

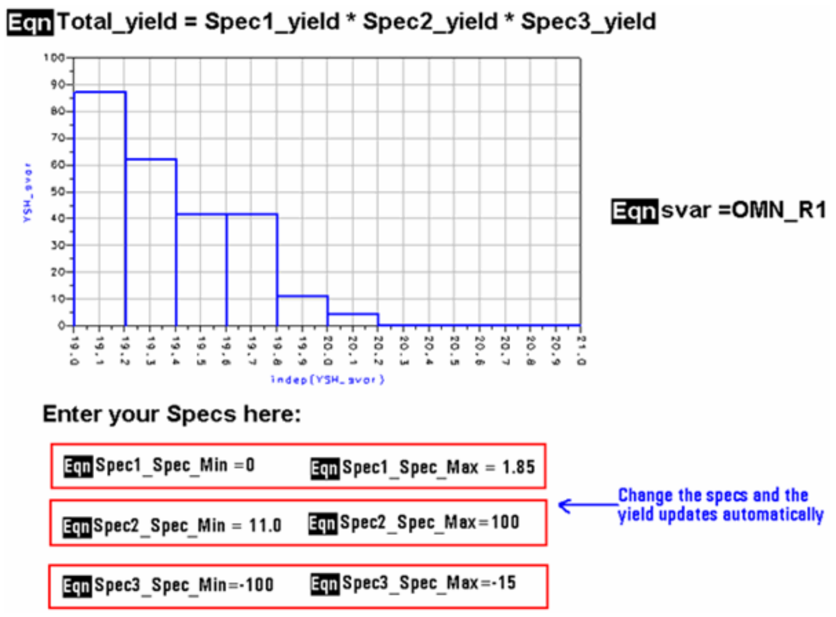
NumFail	NumPass	Yield
763.00	237.00	23.70

Eqn NF=histogram(Spec1, 10, 1.5, 2)
Eqn Gain1=histogram(Spec2, 10, 10, 12)
Eqn Out_match=histogram(Spec3, 20, -24, -14)

Analysis

After running the analysis, all yield analysis data (stored in ADS data set) can now be post processed in many ways in the ADS Data Display to obtain valuable information and insights on the design. You can extract useful information to help you find exactly what element(s) in the design are causing any low yield.

Open the data display template called *General Design Template Sensitivity Histogram*. The following figure shows how it looks when it is applied to the LNA design provided with the template:



Eqn Total_yield = Spec1_yield * Spec2_yield * Spec3_yield

Eqn svar =OMN_R1

Enter your Specs here:

Eqn Spec1_Spec_Min =0   Eqn Spec1_Spec_Max = 1.85

Eqn Spec2_Spec_Min = 11.0   Eqn Spec2_Spec_Max=100

Eqn Spec3_Spec_Min=-100   Eqn Spec3_Spec_Max=-15

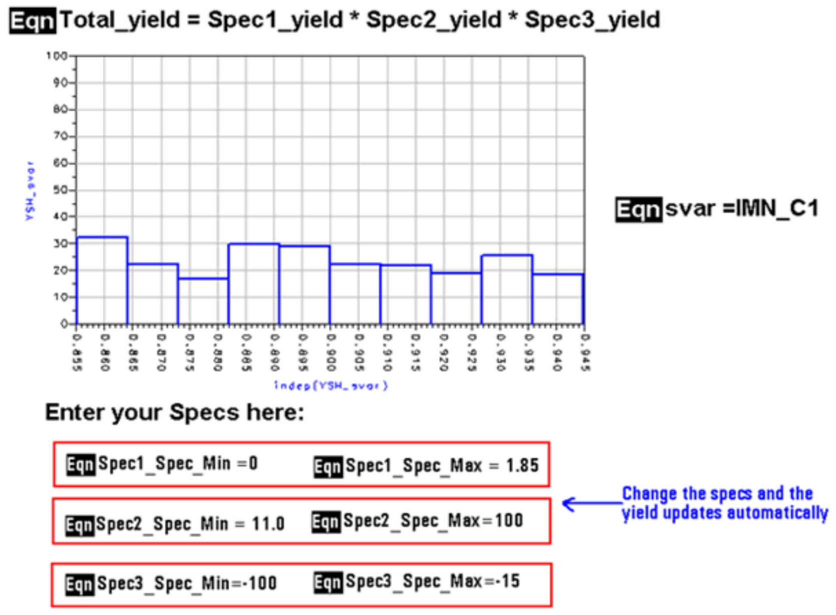
On this template, designers enter their design Specs (in the red boxes) and select any component in the design (from the equation on the right hand side of the display) in order to investigate.

The sample YSH output shown in the previous figure shows that *OMN_R1* (Output Matching Network, Resistor R1) has a nominal value of 20Ω with ±5% tolerance. The plot clearly shows that that the element *OMN_R1* is contributing significantly to the low yield. Notice that when *OMN_R1* is slightly higher than 20Ω, the yield drops to zero.

To investigate *OMN_R1* further, in the design's schematic set *OMN_R1* to exactly 20.5Ω (with no variation), and keep the rest of the components varying. The circuit's yield changes to *zero*. Then, if you set *OMN_R1* to exactly 19.1Ω (with no variation), and keep the rest of the components varying, the yield of the circuit changes to 87%. All of this information is extracted from the display plot in the previous figure. This design template clearly indicates that *OMN_R1* requires modification to improve yield results.

Let us now select another component and investigate its contribution to the overall yield. In the following figure, the display is set to investigate *IMN_C1* (capacitor C1 in the input matching network). Notice that this capacitor is not a sensitive element and the yield stays constant as it varies around its nominal point. This indicates that *IMN_C1* is not an element to worry about and is a good choice in our circuit.

To complete this analysis, continue selecting all other components in the design to study their effects on the yield.



Total_yield = Spec1_yield * Spec2_yield * Spec3_yield

svar =IMN_C1

Enter your Specs here:

Spec1_Spec_Min =0 Spec1_Spec_Max = 1.85

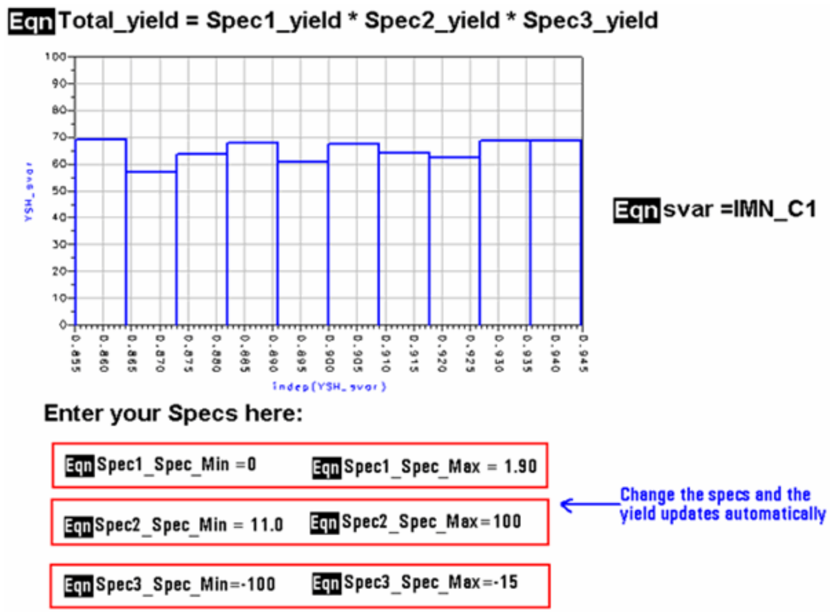
Spec2_Spec_Min = 11.0 Spec2_Spec_Max=100

Spec3_Spec_Min=-100 Spec3_Spec_Max=-15

Change the specs and the yield updates automatically

There are other ways you can use this template:

- Change the specs in the red-boxed equations and automatically see how the yield changes with respect to the chosen Specs. This helps you determine which spec(s) are causing the low yield. If you look in the example template in the following figure, the Noise figure spec (*Spec1_Spec_Max*) is relaxed from 1.85 to 1.9 improving the yield to 70%.



Total_yield = Spec1_yield * Spec2_yield * Spec3_yield

svar =IMN_C1

Enter your Specs here:

Spec1_Spec_Min =0 Spec1_Spec_Max = 1.90

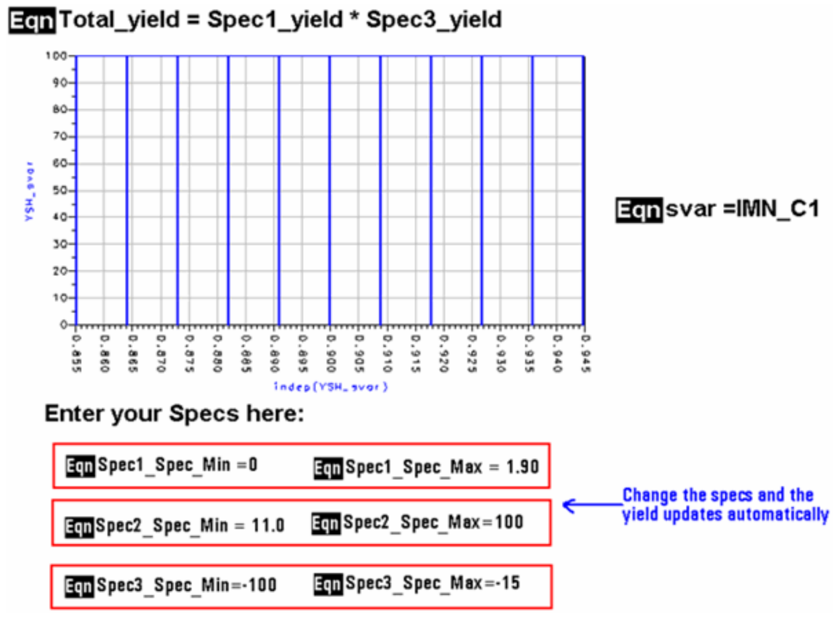
Spec2_Spec_Min = 11.0 Spec2_Spec_Max=100

Spec3_Spec_Min=-100 Spec3_Spec_Max=-15

Change the specs and the yield updates automatically

- Manipulate the *Total_yield* equation at the top of its display page. The template provides the *Total_yield* equation consisting of the product of the three specified yields (*Spec1_yield*, *Spec2_yield*, and *Spec3_yield*). Simply delete one or two of the *SpecN_yield* terms from the *Total_yield* equation and check how the design yield changes.

  In the following figure, notice that the *Spec2_yield* (the Gain spec) is removed from the *Total_yield* equation on the top of the display. The resulting output shows the yield improving to 100% after eliminating the Gain spec (*Spec2_yield*).

- 
Eqn Total_yield = Spec1_yield * Spec3_yield
YSH_svar
indep(YSH_svar)
Eqn svar =IMN_C1
Change the specs and the yield updates automatically

Enter your Specs here:

```
Eqn Spec1_Spec_Min =0        Eqn Spec1_Spec_Max = 1.90
Eqn Spec2_Spec_Min = 11.0    Eqn Spec2_Spec_Max=100
Eqn Spec3_Spec_Min=-100      Eqn Spec3_Spec_Max=-15
```

Using Momentum to simulate an entire amplifier layout

Location: $HPEESOF_DIR/examples/MW_Ckts/MMIC_LNA_wrk

Momentum Examples

Examples on how to evaluate and design modern communications systems for RF and microwave antennas and circuits. Also includes examples on optimization of geometry parameters to achieve optimal structures that meet circuit and device performance goals.

1x4 E-Plane Linear Patch Antenna Array

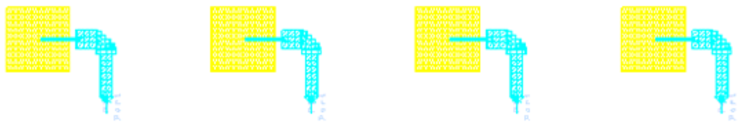
Location: /examples/Momentum/Antenna/E_plane_wrk

Objective

This example shows a Momentum simulation of a four element linear patch antenna array.

Setup

1. "E_plane" shows the layout of the four element patch array with microstip line feeding from underneath.
2. "E_plane.dds" shows the return loss as well as the mutual coupling between the elements. Comparison between Momentum simulation and measurements is also shown.
3. "E_plane_FF_10_GHz_phi90_uniform.dds" shows the array pattern, efficiency as well as gain when all elements are fed in phase.
   "E_plane_FF_10_GHz_phi90_phaseShift.dds" is for the case when there is a phase taper in the excitation.



Analysis

Figure 1: Return loss

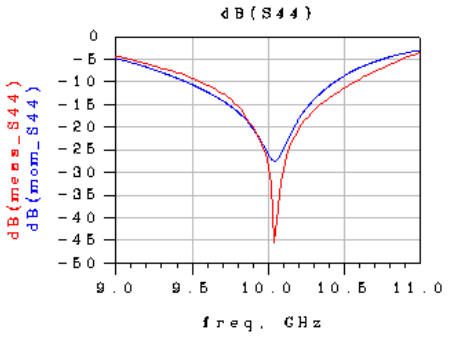


Figure 2: Mutual coupling level

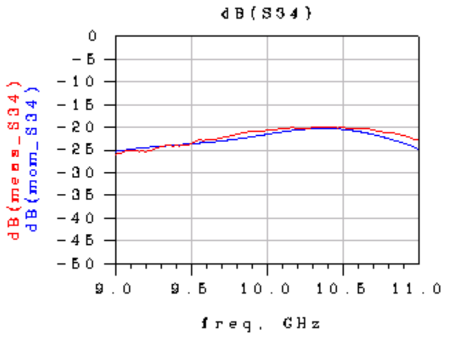


Figure 3: Array pattern

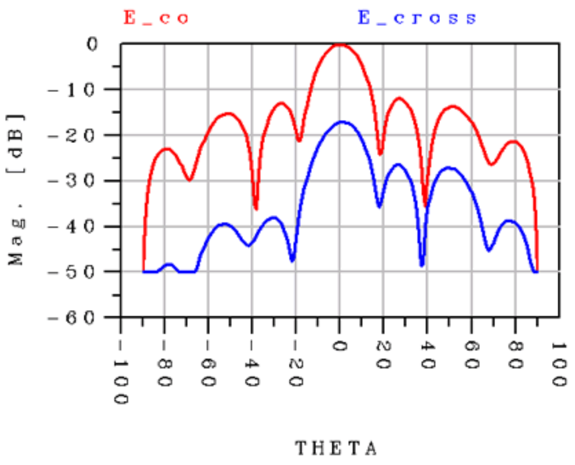
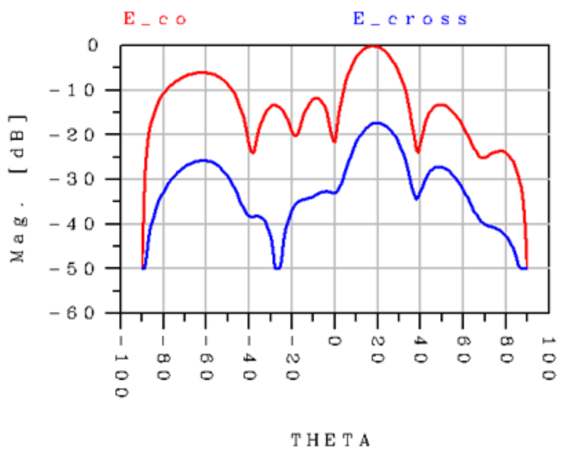


Figure 4: Steered main beam due to phase tapering



Notes

- Applying a phase taper over the elements of 90 deg/element steer the main beam by 18 deg. The directivity decreases to 13.3 dB, mainly due to the rise of the grating lobe, the efficiency drops to 70%.

Antenna with Circular Polarization

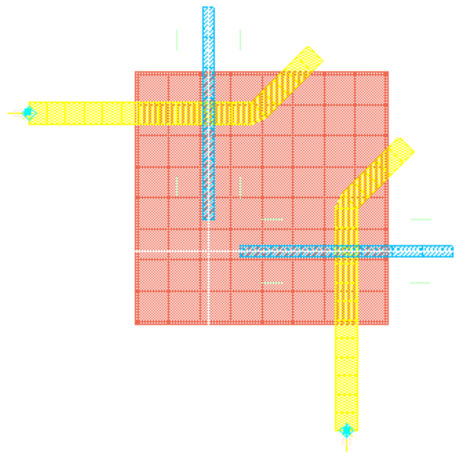
Location: /examples/Momentum/Antenna/Circular_Polarization_wrk

Objective

Use Momentum to analyze a patch antenna with circular polarizaiton. The antenna is excited by two microstrip line through two orthogonally placed slots in the ground plane.

Setup

1. The design is shown in "Circular_Polarization_wrk". By exciting the patch using two orthogonally placed slots with 90 phase difference, one can obtain a circularly polarized antenna.
2. The radiation impedance is plotted in "circular_polarization.dds". The resonant frequency is 10 GHz. The antenna is not matched, which is part of the feeding network design.
3. A planar cut of the far fields is stored in "circular_polarization_FF_left_phi0.dds".



Analysis

Figure 1: Radiation Impedance

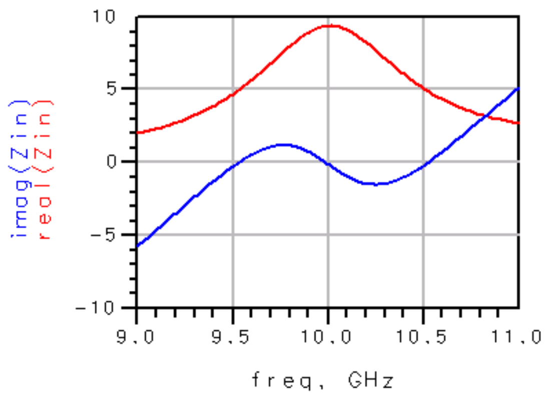


Figure 2: Right-hand and left-hand fields

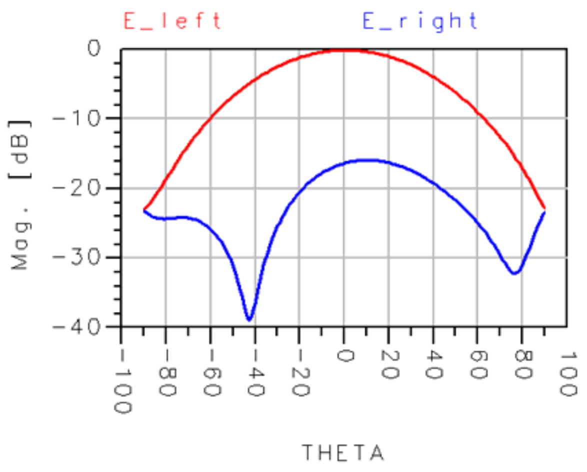
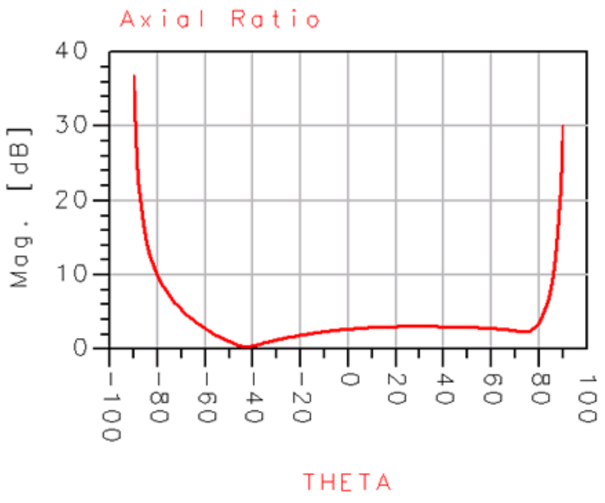


Figure 3: Axial ratio



Notes

- In this example, the feed network to achieve this condition is not shown. Concentration is on the far field characteristics.
- Via strips are used to short-circuit the parallel plate mode that propagates in the strip line region. Removing the vias results in an important drop in the radiation efficiency. The energy is launched in the parallel plate mode rather than being radiated.

Box Example

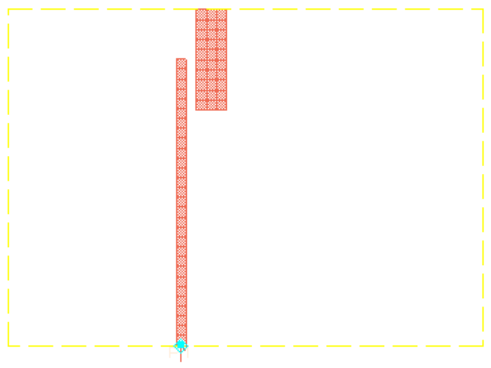
Location: $HPEESOF_DIR/examples/Momentum/Microwave/BoxExample_wrk

Objective

This example illustrates the capability of Momentum to analyze passive planar circuits in a highly-resonant rectangular metallic enclosure.

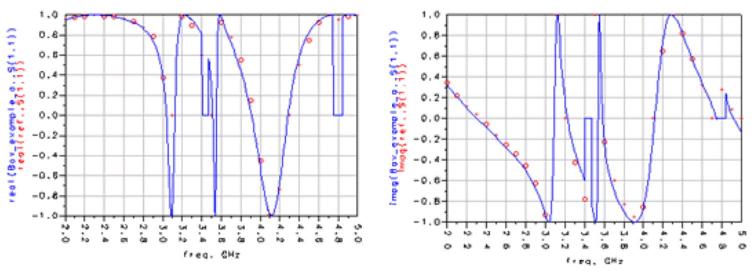
Setup

The microstrip structure is printed on a 0.635cm thick substrate with a dielectric constant of 2.2. The structure is meshed at 5GHz with 30 cells per wavelength. The circuit is enclosed in a rectangular box with an uncalibrated port on a box wall.



Analysis

Figure 1: S11 Comparison between simulation and measurement



Notes

- Layout and measured results were obtained from: "On the summation of double infinite series field computations inside rectangular cavities", S. Hashemi-Yeganeh, IEEE-MTT, vol. 43, no. 3, March 1995, pp. 641-646.

Broadband Planar Antenna for GPS, DCS-1800, IMT-2000, and WLAN Applications

Location: $HPEESOF_DIR/examples/Momentum/Antenna/Multi_Band_wrk

Objective

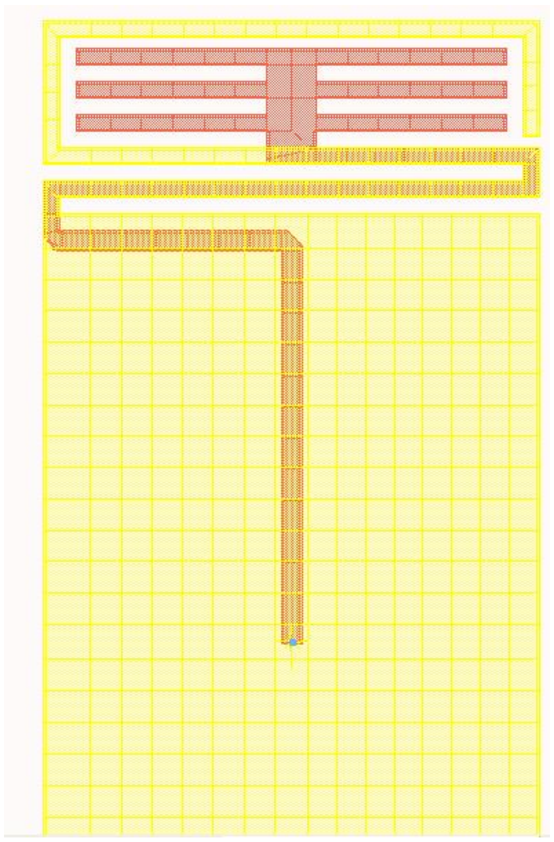
This example shows Momentum simulations of a Broadband WLAN Antenna with a limited ground plane. This Multi-band antenna can be use for GPS, DCS-1800, IMT-2000 and WLAN-IEEE 802.11b applications.

Setup

1. "MultiBand_WLAN_Antenna" shows the layout of the WLAN antenna. The antenna consists of an S-strip and a T-strip, separately printed on two sides of a thin substrate.
2. "MultiBand_WLAN_Antenna.dds" shows the return loss of antenna. The S-parameter curve shows three resonances at 1.64, 2.03 and 2.52 GHz. The first resonance band is for GPS applications, the second resonance band is for DCS and IMT bands, and the third resonance is for WLAN Applications. A comparison between Momentum simulation and measurements is also shown.
3. "MultiBand_WLAN_Antenna_FF_2_03GHz_Phi0_f.dds" shows the antenna far field pattern at 2.03 GHz.
4. "MultiBand_WLAN_Antenna_FF_2_52GHz_Phi0_f.dds" shows the antenna far field

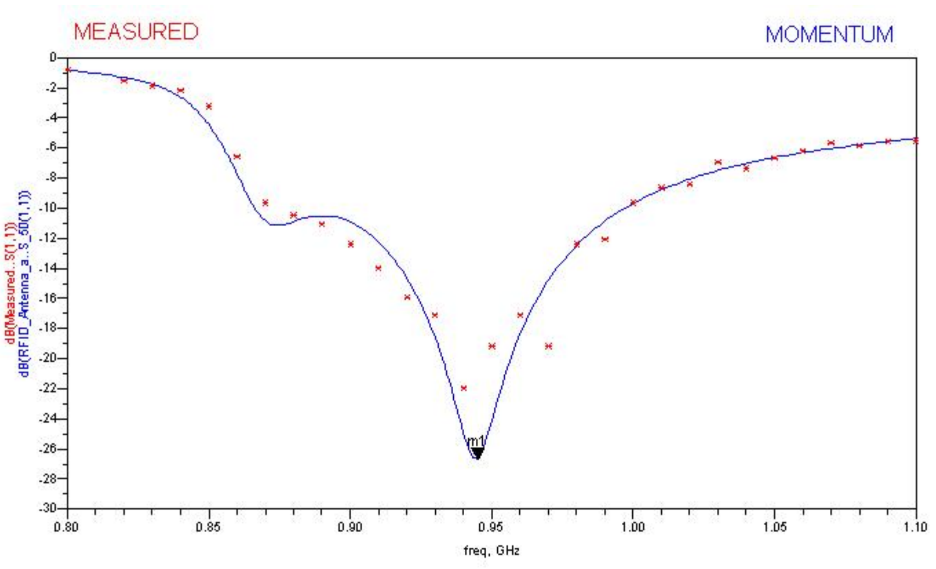
pattern at 2.52 GHz.

Figure 1: Antenna Layout



Analysis

Figure 2: Return Loss Plot



Eqn S11_Momentum=dB(RFID_Antenna_a..S_50(1,1))

Eqn S11_Measured=dB(RFID_Antenna_a..S(1,1))

Figure 3: Radiation Pattern at 2.03 GHz

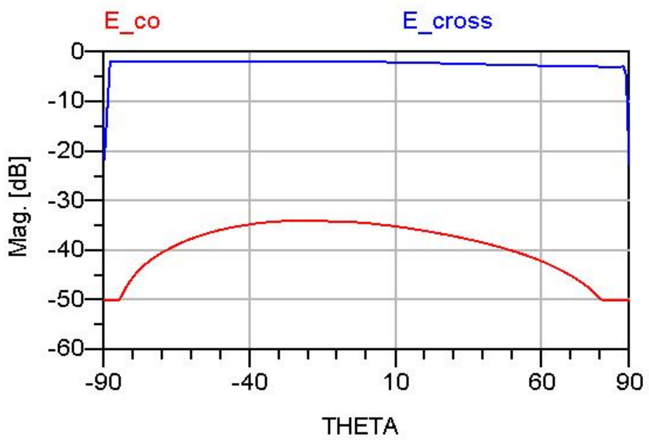
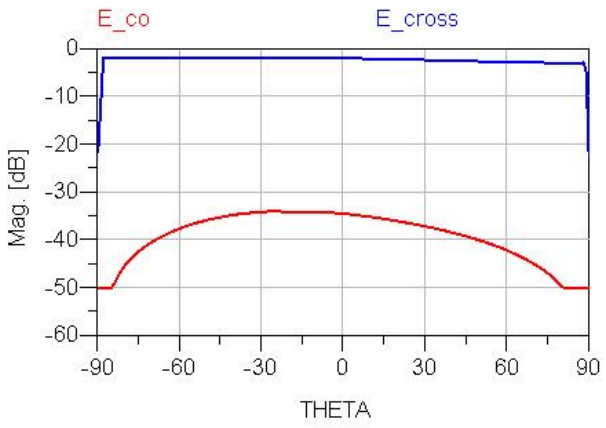


Figure 4: Radiation Pattern at 2.52 GHz



References

1. "A Compact Broadband Planar Antenna for GPS, DCS-1800, IMT-2000 and WLAN Applications", RongLin Le, Bo Pan, Joy Lasker and Monos M. Tentzeris, IEEE transaction on Antenna and Wireless Propagation Letters, Vol 6, 2007, pp 25-27.

Coplanar Waveguide Bend

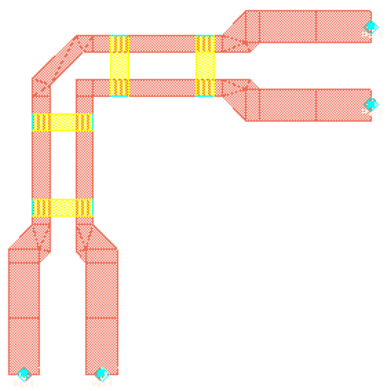
Location: $HPEESOF_DIR/examples/Momentum/Microwave/CPW_bend_wrk

Objective

This example illustrates the simulation of a coplanar waveguide (CPW) bend with air bridges.

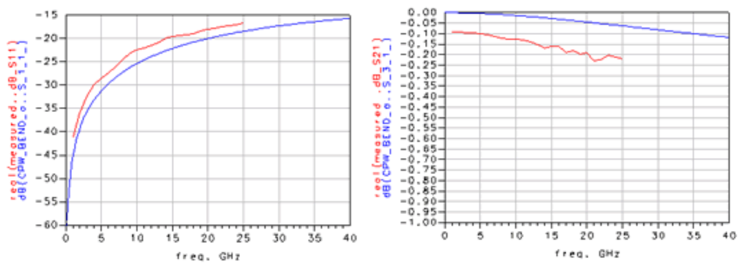
Setup

The CPW is printed on a 90um GaAs substrate. In CPW, both even and odd modes are excited. To compensate for this at discontinuities, air bridges are commonly used to short-out the odd mode. The air bridge is 2um above the substrate surface. The structure is meshed with 30 cells per wavelength at 40GHz. It uses 7 frequency points to cover the frequency from DC to 40 GHz.



Analysis

Figure 1: Comparison between simulation and measured S-parameters



Coplanar Waveguide Line with Finite Metal Thickness and Loss

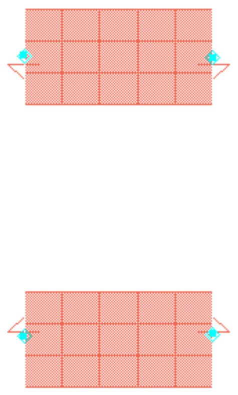
Location: $HPEESOF_DIR/examples/Momentum/Microwave/CPW_line_wrk

Objective

This examples illustrates the simulation of a coplanar waveguide (CPW) line using Momentum. In the Momentum analysis, the conductor loss and finite thickness are also considered. The analyzed CPW line is also a multilayer structure.

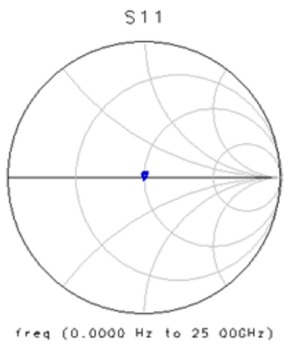
Setup

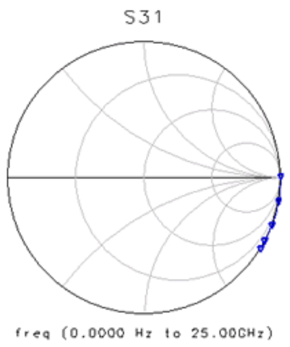
The CPW is printed on a 635um substrate with permittivity of 9.9. The substrate is put on another thick Teflon substrate. In the Momentum simulation, the CPW is implemented as a pair of coupled slots with associated coplanar ports. The structure is meshed with 50 cells per wavelength at 25GHz.



Analysis

Figure 1: Simulated S-parameters





Notes

- CPW can be regarded as two coupled slot lines, instead of a single transmission line structure. In actual circuit applications, the signal excitations for CPWs are such that the slot modes cannot propagate independently from each other. The central strip is the signal line of a given polarity and the E-field points either symmetrically away from or towards this signal line from the outer ground plane sections. This is known as the CPW mode. The "slot mode" corresponds to the case where the E-field points from one ground to the signal line to the other ground.

Coplanar Waveguide Notch

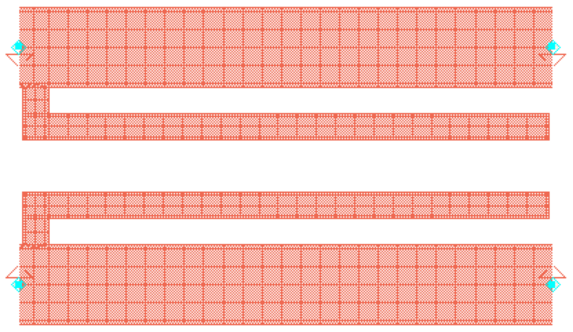
Location: $HPEESOF_DIR/examples/Momentum/Microwave/CPW_notch_wrk

Objective

This example illustrates the simulation of a coplanar waveguide (CPW) notch filter using Momentum.

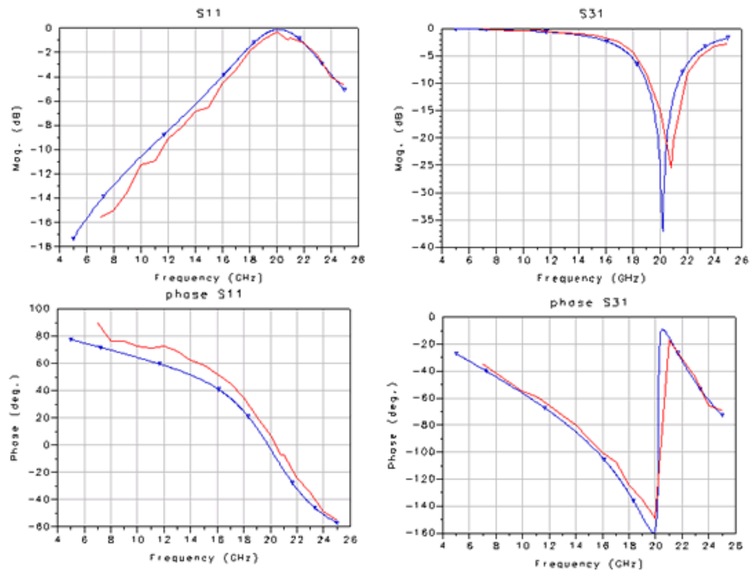
Setup

The filter is realized using resonant notches in the center conductor of a coplanar waveguide structure. The substrate is a 635um Alumina. Finite conductor thickness and meal loss is also included. In Momentum simulation, the region without metal (slot area) is meshed just as in the other CPW case. Each Coplanar port consists of two slot ports. The structure is meshed with 100 cells per wavelength at 25GHz with edge mesh on.



Analysis

Figure 1: S-parameter comparison between simulation and measurements



Notes

- The measured results can be found in a paper by Nihab I. Dib, Nathi, Ponchak, Simons, "Theoretical and Experimental Characterization of Coplanar Waveguide Discontinuities for Filter Applications", IEEE Transations on MTT, Vol. 39, No. 5, May 1991, pp. 873-882.

Coplanar Waveguide Open

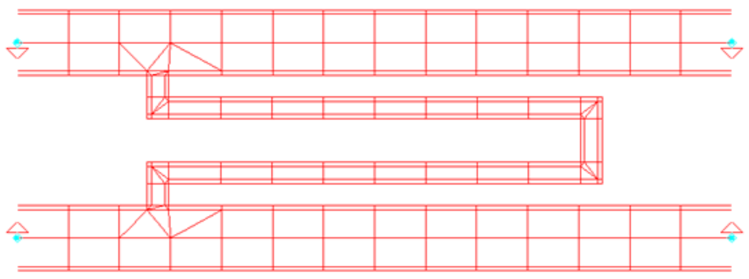
Location: $HPEESOF_DIR/examples/Momentum/Microwave/CPW_open_wrk

Objective

This example illustrates the simulation of a coplanar waveguide (CPW) open using Momentum.

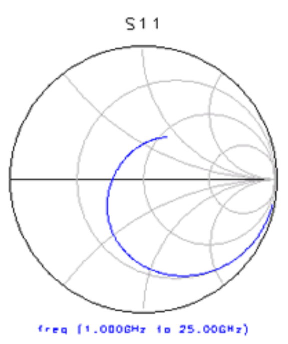
Setup

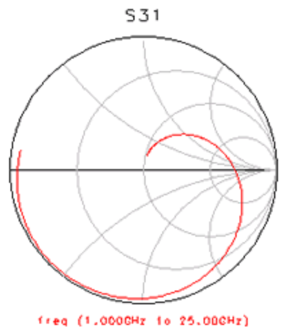
The CPW open is a simple variation of the CPW_notch circuit, except that the notches are connected together to form an open for the CPW. The structure is meshed with 50 cells per wavelength at 25GHz with edge mesh on.



Analysis

Figure 1: S-parameter from simulation





Notes

- The measured results can be found in a paper by Nihad I. Dib, Nathi, Ponchak, Simons, "Theoretical and Experimental Characterization of Coplanar Waveguide Discontinuities for Filter Applications", IEEE Transactions on MTT, Vol. 39, No. 5, May 1991, pp. 873 -882.

Coupled Stripline Filter

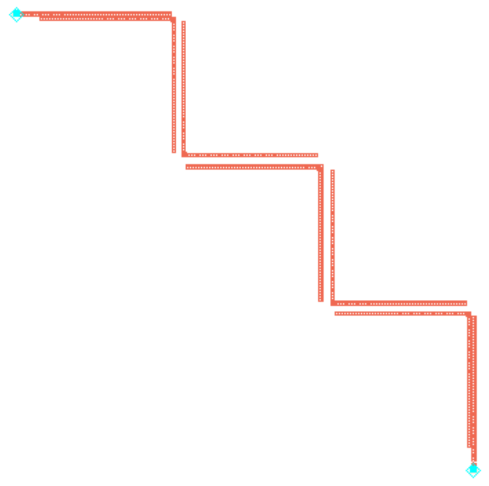
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Six_pole_filter_wrk

Objective

This example illustrates a six-section side-coupled stripline filter. Finite metal thickness and metal loss are also considered in the Momentum simulation.

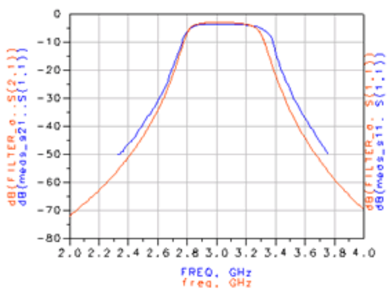
Setup

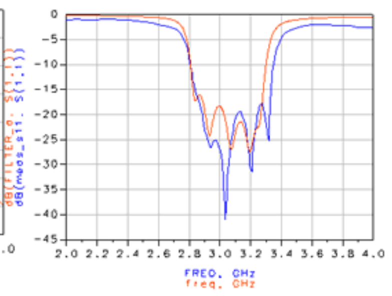
The filter is between two 25mil substrate with dielectric constant of 10.5 with loss tangent of 0.003. The mesh is at 30 cells per wavelength at 5GHz with edge mesh on.



Analysis

Figure 1: Simulated versus measured S-parameters





Notes

- This filter was constructed on a stripline substrate, it was measured with additional microstrip feedlines which themselves were connected to a type of transmission line called a "waffle line". Therefore, the measurements indicate a greater insertion loss than can be accounted for with material losses alone.

Coupled Stubs

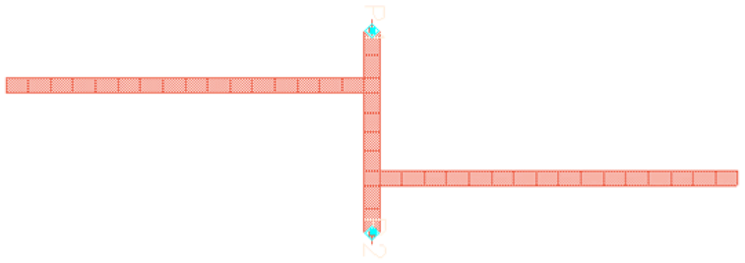
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Coupled_Stubs_wrk

Objective

This example shows the Momentum simulation of the mutual coupling between two tuning stubs.

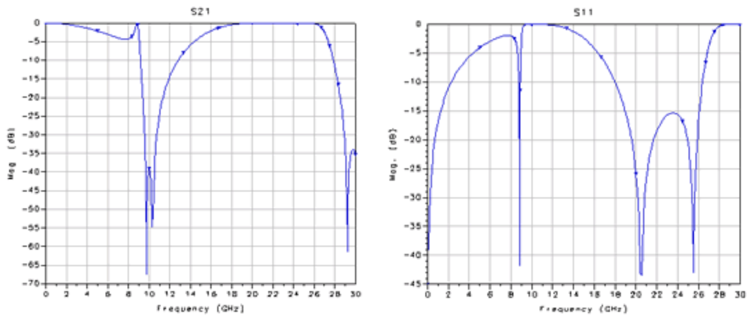
Setup

The stubs are printed on a 5 mil substrate with permittivity 9.9. The structure is meshed with 20 cells per wavelength at 30GHz with arc facet angle of 45 degrees. It uses 15 frequencies to cover the frequency range between DC and 30GHz.



Analysis

Figure 1: Simulated S-parameters



Notes

- Because of the resonant conditions that are set up, the stubs couple via their mutual inductance causing the single resonance to split into two resonances and therefore reduce the Q of the circuit. Sometimes these effects may prove to be insignificant in the final circuit application, but quantitative information need to be obtained through an accurate electromagnetic analysis.

Elliptic Filter Simulation with Model Composer models

Location: /examples/Momentum/ModelComposer/EllipticFilter_wrk

Objective

This example shows the use of Model Composer models in an elliptic filter simulation and also demonstrates its advantages against ADS built-in models and full-wave EM simulations.

Setup

ELLIPTIC FILTER

The data display "elliptic_filter.dds" compares

1. simulations with Model Composer models (green)
2. global Momentum simulations (red), and
3. simulations based on standard ADS analytical models (blue).
   The ADS analytical models were used outside their validity range.

The results based on the Model Composer models correspond well to the global full-wave Momentum simulations, and yet the simulations based on the Model Composer models only took a fraction of the time required for global full-wave analysis (about 1 second compared to 37 minutes).

The Model Composer models provide EM accuracy
and generality at traditional circuit simulation
speed.

MODEL COMPOSER LIBRARY - INSTALLATION INFO
The Model Composer libraries are stored as ADS Design Kits.

ADS Design Kits can easily be installed by selecting:
ADS Main window > DesignKits > Manage Favorite Design Kits

Make sure that the ADS Design Kit(Library definition file) is enabled.

Check also the ADS Model Composer documentation and the ADS Design Kit documentation for more installation info.

Analysis

Figure 1:Schematic of Elliptic Filter with components from Model Composer

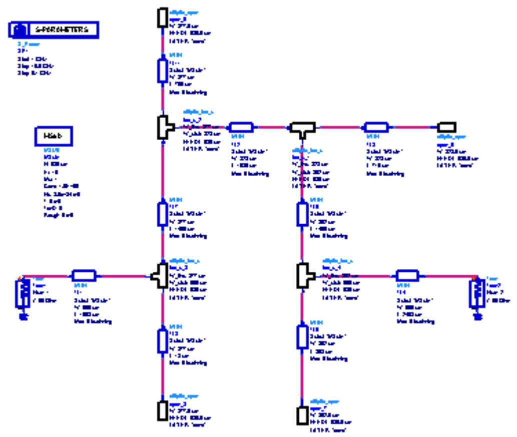


Figure 2:Layout of Elliptic Filter

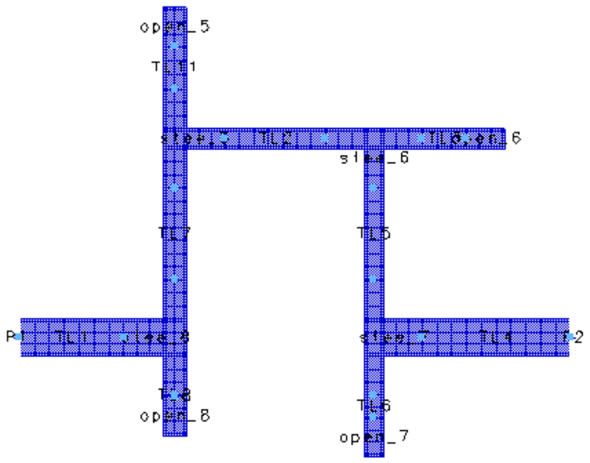
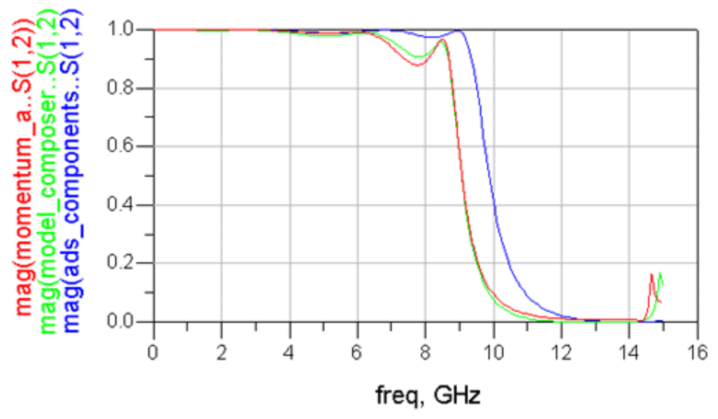


Figure 3:Comparison of simulation results



Notes

REFERENCE : F. Giannini, M. Salerno and R. Sorrentino, "Design of Low-Pass Elliptic Filters by Means of Cascaded Microstrip Rectangular Elements", IEEE Transactions on Microwave Theory and Techniques, vol. 30, no. 9, 1982.

EM-Circuit Cosimulation with 1GHz LNA

Location: /examples/Momentum/emcktcosim/LNAEmCktCosim_wrk

Objective

This example provides details on how to use the Momentum EM-Circuit cosimulation feature. 1GHz LNA was used for the example and the two simulation results, the one with ADS built-in microstrip models and the other with Momentum RF EM simulation, are compared.

Setup

In this workspace,the use of the Momentum EM-Circuit cosimulation feature is illustrated for the LNA design from the examples\MW_Ckts\LNA_1GHz_wrk workspace.

We will compare the simulation results for this LNA design using two ways to characterize the board interconnections: using the ADS Microstrip component models and using Momentum RF.

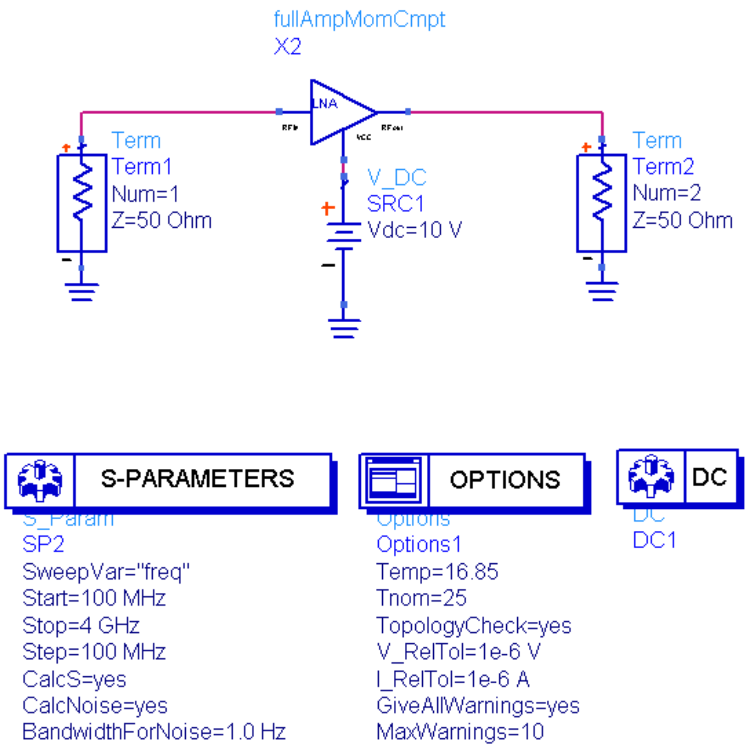
The simulation setup using the ADS Microstrip components is in the design called: SimFullAmp, which includes an instance of the LNA design called fullAmp.

The simulation setup that makes use of the EM-Circuit cosimulation feature is called SimFullAmpMomCmpt and includes an instance of SimFullAmpMomCmpt. Push into this component to see how the EM-Circuit cosimulation was set up for this simulation. The board layout is a Momentum component, called "BoardLayout". This component is created from the associated layout design called BoardLayout. Note that the boardLayout Momentum component is connected directly to other components in the fullAmpMomCmpt design. The S-parameter simulation in the SimFullAmpMomCmpt will automatically do the Momentum board characterization as part of the circuit simulation run, and if possible will reuse previous data stored in the database.

The data display comparing the simulation results for the SimFullAmp and SimFullAmpMomCmpt designs is called SimResults

Analysis

Figure 1:Top-level schematic



S_Param
SP2
SweepVar="freq"
Start=100 MHz
Stop=4 GHz
Step=100 MHz
CalcS=yes
CalcNoise=yes
BandwidthForNoise=1.0 Hz

Options
Options1
Temp=16.85
Tnom=25
TopologyCheck=yes
V_RelTol=1e-6 V
I_RelTol=1e-6 A
GiveAllWarnings=yes
MaxWarnings=10

DC
DC1

Figure 2:Amplifier with Momentum component

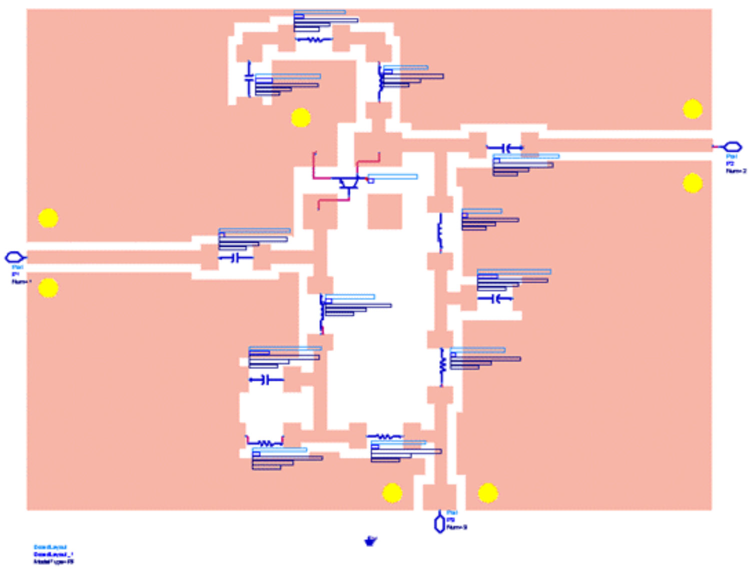
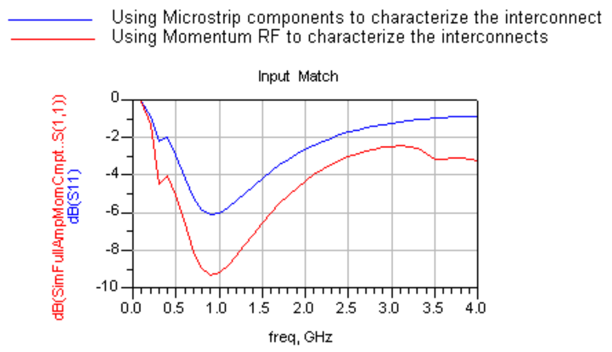
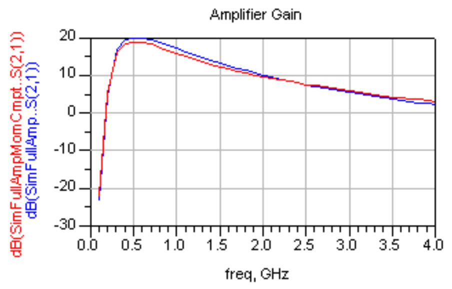
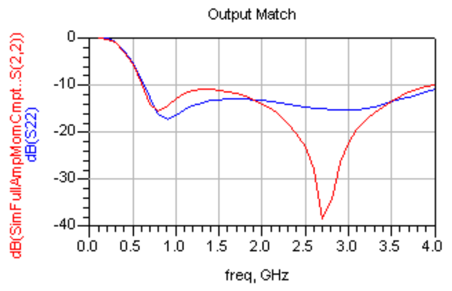
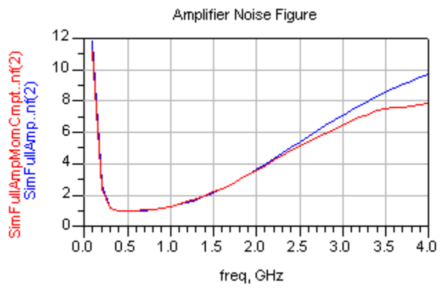


Figure 3:Simulation results comparing EM Simulation result with built-in models









EM-Circuit Co-Simulation with a LTCC low pass filter

Location: /examples/Momentum/emcktcosim/LTCC_wrk

Objective

This workspace illustrates the usage of the Momentum EM\ckt co-simulation feature as a way to create Layout Components that can be used to design a larger structure. In the example three layout components are used to build a low pass filter: an interconnect, a spiral and a capacitor. For the capacitor component, two designs will be used where one has the surface of the capacitor plates as a layout parameters.

Setup

This workspace illustrates the usage of the Momentum EM/ckt cosimulation feature as a way to create Layout Components that can be used to design a larger structure. In the example we will use three layout components: an interconnect, a spiral and a capacitor to build a low pass filter. For the capacitor component, two designs will be used where one has the surface of the capacitor plates as a layout parameters.

The technology used in this example is LTCC. The basic components have been designed, a spiral inductor, two capacitor components (one is parameterized) and a T interconnect component. The layouts for these componens are called 'spiral', 'cap', 'cap_p' and 'connect'. From these layouts, schematic components have been created using the Momentum> Component>Create/Update dialog box. Note that for the 'cap_p' design, a layout parameter 'deltalen' has been defined that controls the size of the capacitor plates. To view different instances of the 'cap_p' design with different values of the 'deltalen' parameter, you can inspect the 'cap_p_instances' layout design.

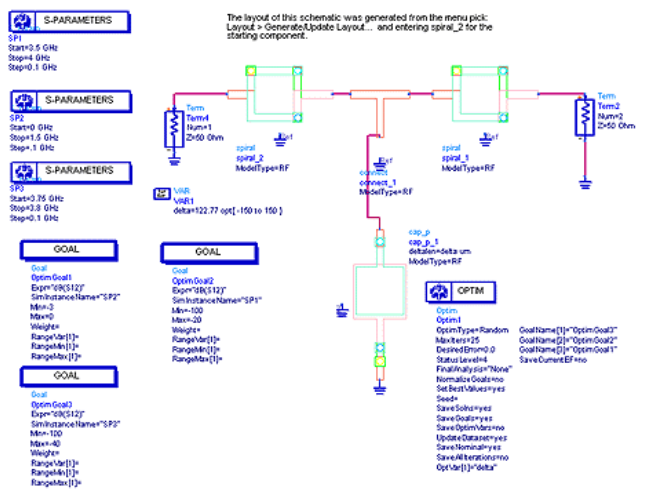
The spiral, cap and connect components are used in a schematic design to build a bigger circuit. This has been done in the schematic design called "filterdesign". From this design a simple circuit simulation run generates the S-parameter result for the complete design, where if needed the Momentum simulations are run in automatically as part of this process. The S-parameter results of the simulation on the 'filterdesign' schematic are shown in filterdesign.dds.

It is seen from the results of this simulation that the S11 parameter at 2 GHz is - 8.5db. We will optimize the design (more particularly the size of the capacitor) so that S11 has a value inbetween -15 and -16 dB at 2 GHz. This is done using the design called 'filterdesign_optimize'. This optimization process results in a optimized value for 'deltalen' equal to -31.25. The resulting S-paramaters and the comparison with the original S-parameters is given in the data display called 'filterdesign_optimize'.

Performing a Layout>Generate/Update Layout creates the complete layout of the filter design.

Analysis

Filter with Momentum components



Low-Pass Filter

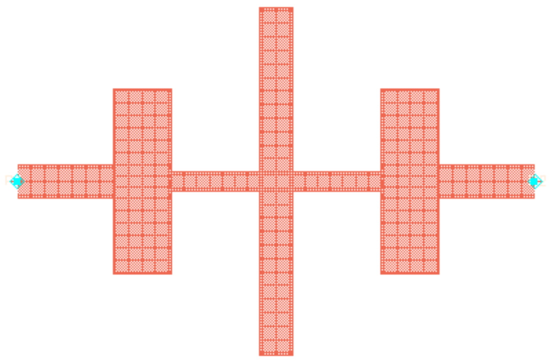
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Low_pass_filter_wrk

Objective

This example illustrates how to use basic library elements (microstrip lines) to creat a low-pass filter layout.

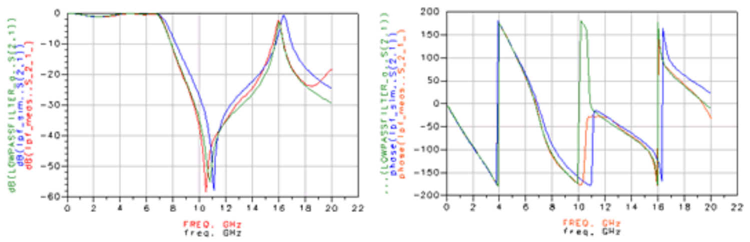
Setup

The low pass filter can be built by using library microstrip line element directly in the layout. The structure has a thickness of 25 mil with a dielectric constant of 9.9. It is meshed with 20 cells per wavelength at 20GHz with edge mesh on.



Analysis

Figure 1: Comparison of measured, part-library-based simulation and Momentum-based simulation for the magnitude and phase of S21



Low-Pass Filter with Hairpin Bend

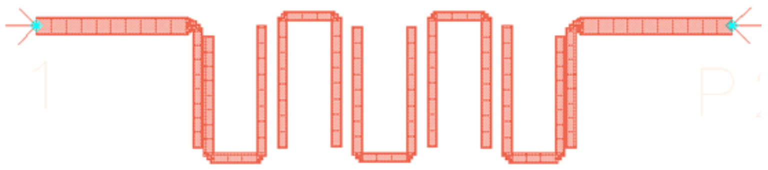
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Hairpin_filter_wrk

Objective

This example illustrates the simulation of a hairpin low-pass filter.

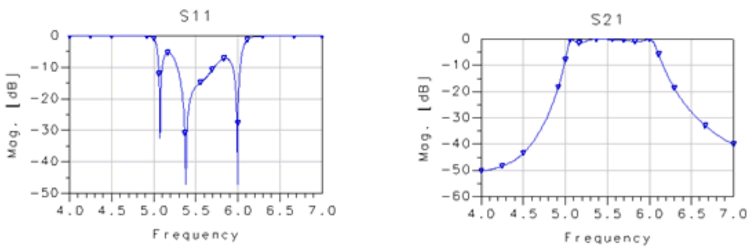
Setup

The hairpin filter is interesting because of the very tightly coupled transmission line sections that are used. In this example, the filter is built on a 25mil Alumina substrate, a metal plane which is 4.7mm above the filter covers the structure. The mesh is at 20 cells per wavelength at 10GHz with edge mesh on.



Analysis

Figure 1: Filter response



Low-Pass Filter with High Out-of-Band Rejection

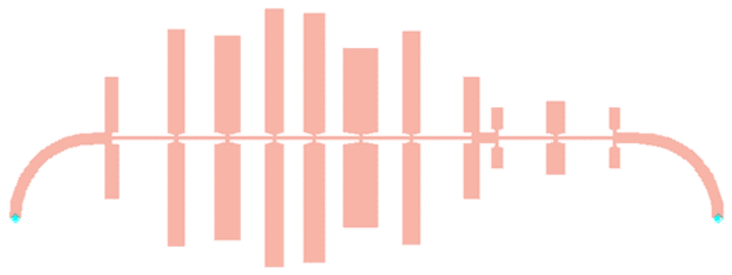
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Lp4_8GHz_wrk

Objective

This multi-pole filter example shows the Momentum ability to simulate a complex structure. The filter has a greater than 100-dB out-of-band rejection and shows Momentum's numerical noise floor occurs around 80-90dB.

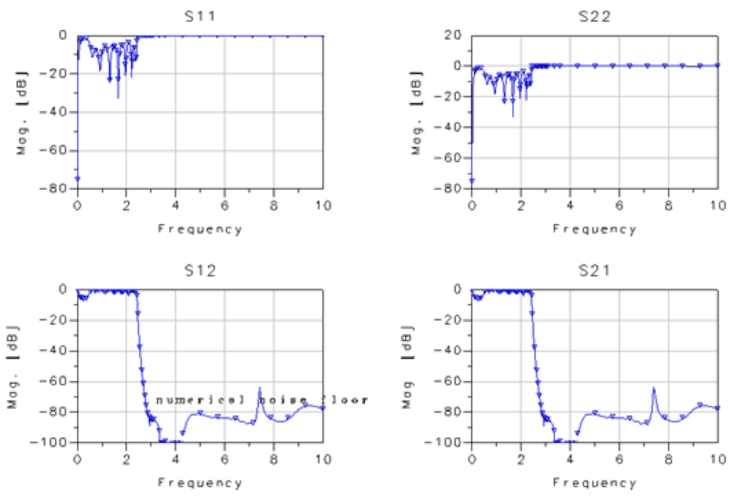
Setup

The low pass filter is built on a 10mil Alumina substate, it is designed for maximum rejection in the 4-8 GHz band. It is meshed with 15 cells per wavelength at 10GHz with edge mesh on.



Analysis

Figure 1: Simulated S-Parameters



Low-Pass Stripline Filter with/without Side Walls

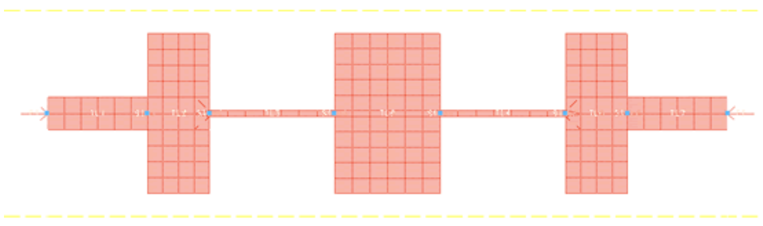
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Waveguide_wrk

Objective

This example investigates the effect of adding side walls to a stripline low-pass filter. It is similar to example Stripline_LPF_wrk.

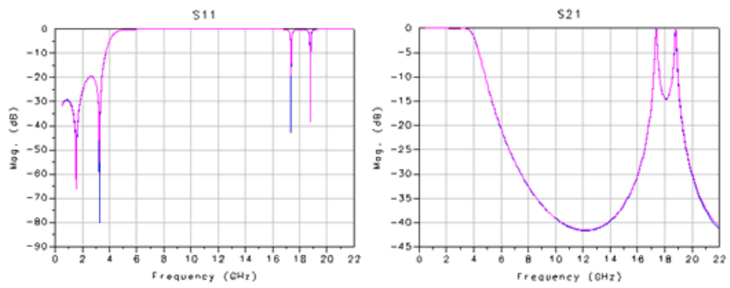
Setup

The low-pass filter layout can be built through standard library components. The filter is built on Duroid board with dielectric constant of 2.2. It is meshed on 20 cells per wavelength at 15GHz.



Analysis

Figure 1:S-parameter comparison for the case with and without waveguide walls



Microstrip Line with Via Stubs

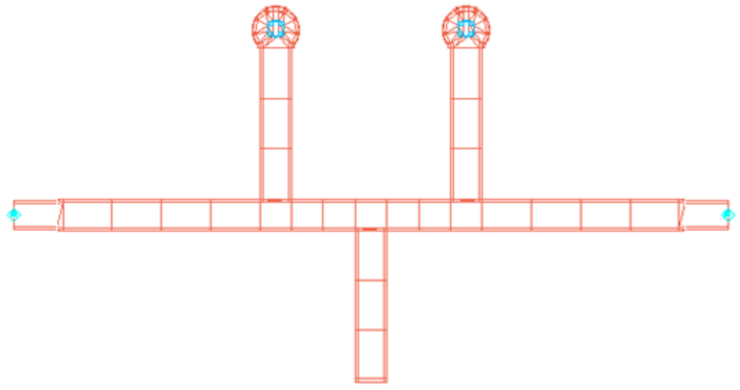
Location: $HPEESOF_DIR/examples/Momentum/Microwave/ViaStub_wrk

Objective

This example illustrates the use of arc facet cell enabling efficient mesh in structures with round via holes.

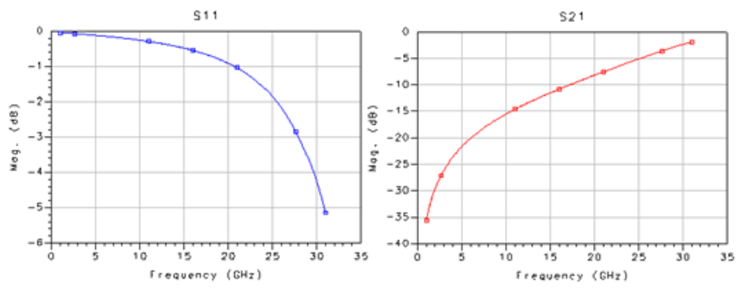
Setup

The original layout was drawn with a 5-degree-arc resolution, which results in a very dense mesh in the region around the vias. The Arc recognition feature provides an accurate mesh with minumum cell density. The mesh is with 30 cells per wavelength at 3GHz with Arc facet angle 45 degrees and edge mesh on.



Analysis

Figure 1: Simulated S-parameters



Microstrip Meander Line

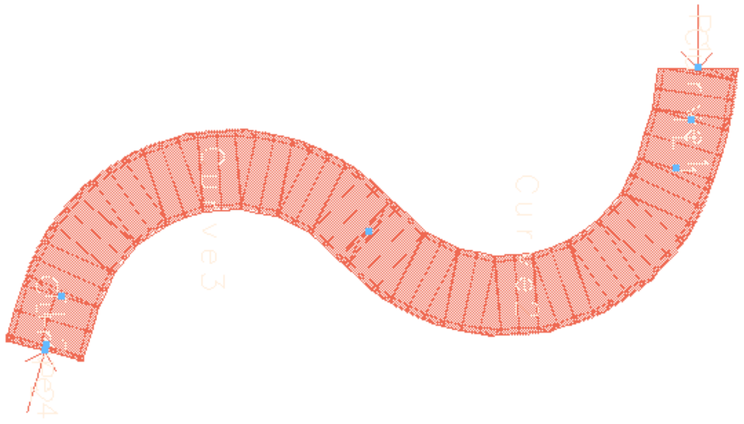
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Meander_wrk

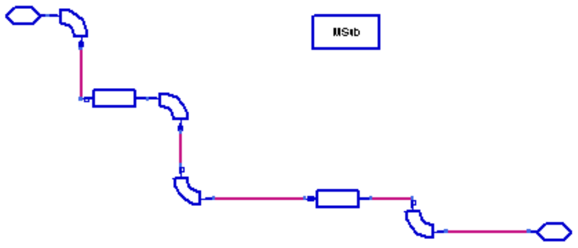
Objective

This example illustrates the Momentum simulation of a curved microstrip meander line.

Setup

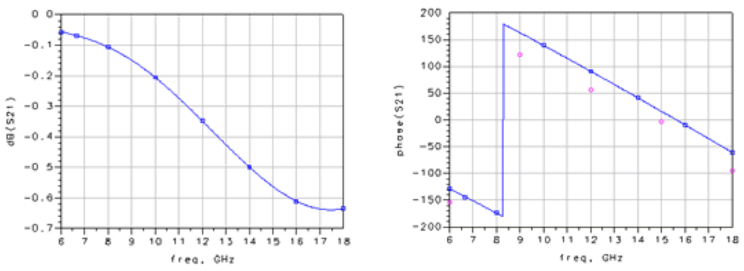
The meander line is built on a 25mil Alumina substrate; it is meshed with 30 cells per wavelength at 20GHz with edge mesh on. Finite conductor thickness and metal loss are also considered. The layout is translated from the schematic automatically.





Analysis

Figure 1: Comparison of simulation and measurement data on S21



Optimization of A Microstrip Line

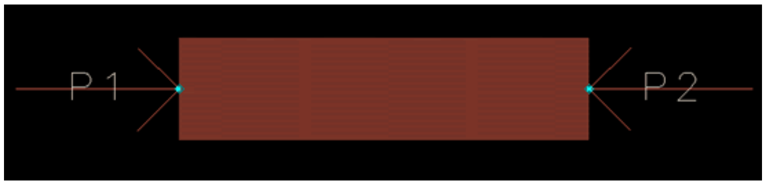
Location: $HPEESOF_DIR/examples/Momentum/Optimization/Microstrip_line_wrk

Objective

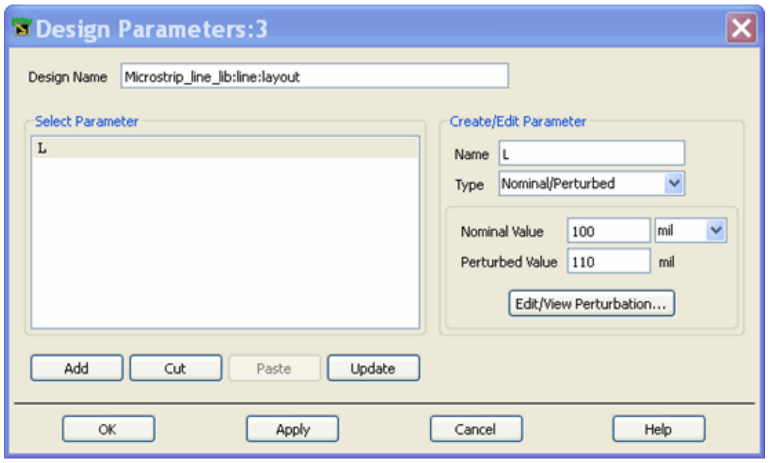
The objective of this example is to optimize a microstrip line length to achieve a phase lag of Phase(S21) = -120 degrees at 10GHz using a layout component.

Setup

1. This example is a 50-ohm microstrip line on a 25-mil substrate. The line width is 25 mils. The substrate dielectric constant is 9.8. The length **L** of the line is optimized to satisfy the specification:
   Phase S21 = -120 degrees at f=10 GHz.
2. Open " *line* ".



3. In the layout a nominal/perturbed parameter **L** is defined by selecting **EM** > **Component** > **Parameters**. This is shown in the following illustration.

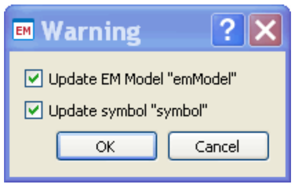


Design Name: Microstrip_line_lib:line:layout

Select Parameter: L

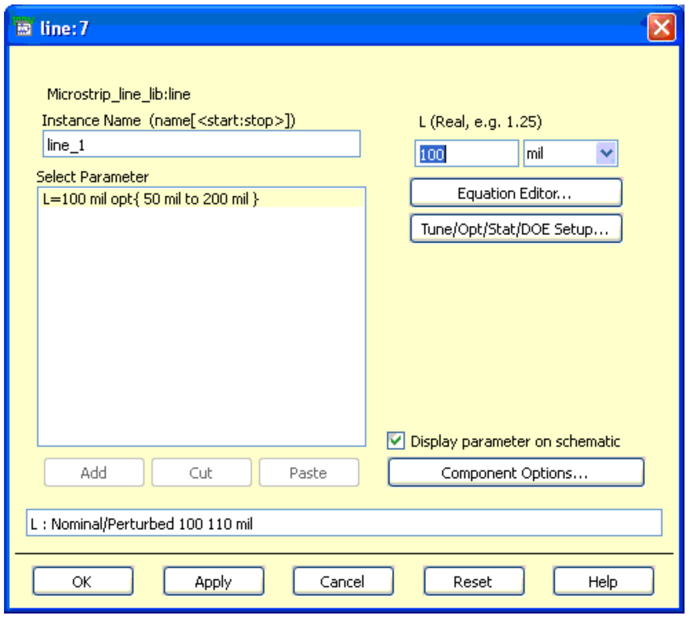
Create/Edit Parameter
Name: L
Type: Nominal/Perturbed
Nominal Value: 100 mil
Perturbed Value: 110 mil

4. Next, the EM Model settings can be done using Symbol/model tab in EMsetup. Select **EM** > **Component** > **Create EM Model and Symobol** to open this dialog.



☑ Update EM Model "emModel"
☑ Update symbol "symbol"

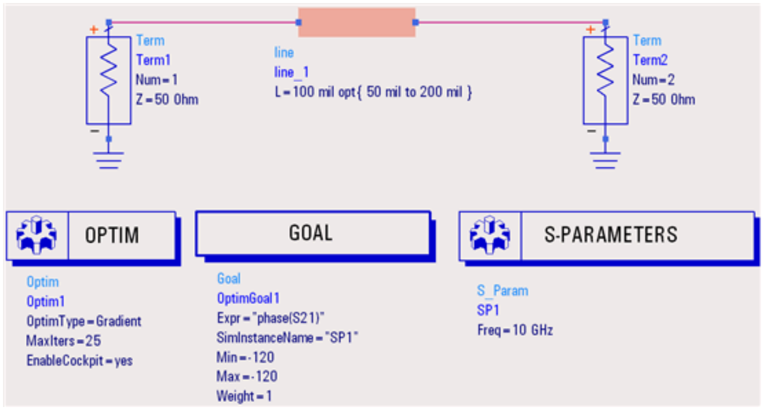
5. The EM Model is then inserted in a Schematic window. This is shown in "*line_optimization*".
6. Double click the EM Model to open the Component dialog. This is where parameter **L** is set for use in the optimization.



Microstrip_line_lib:line
Instance Name (name[<start:stop>]): line_1
Select Parameter: L=100 mil opt{ 50 mil to 200 mil }
L (Real, e.g. 1.25): 100 mil
☑ Display parameter on schematic
L : Nominal/Perturbed 100 110 mil

7. An optimization *Setup* and *Goal* are added from the *Optim/Stat/Yield/DOE* palette. In this case: Goal: Phase S21 = -120 Optimization type used: Random

Note
S-parameter terminations and a simulation setup were added from the "*Simulation-S_Param*" palette.

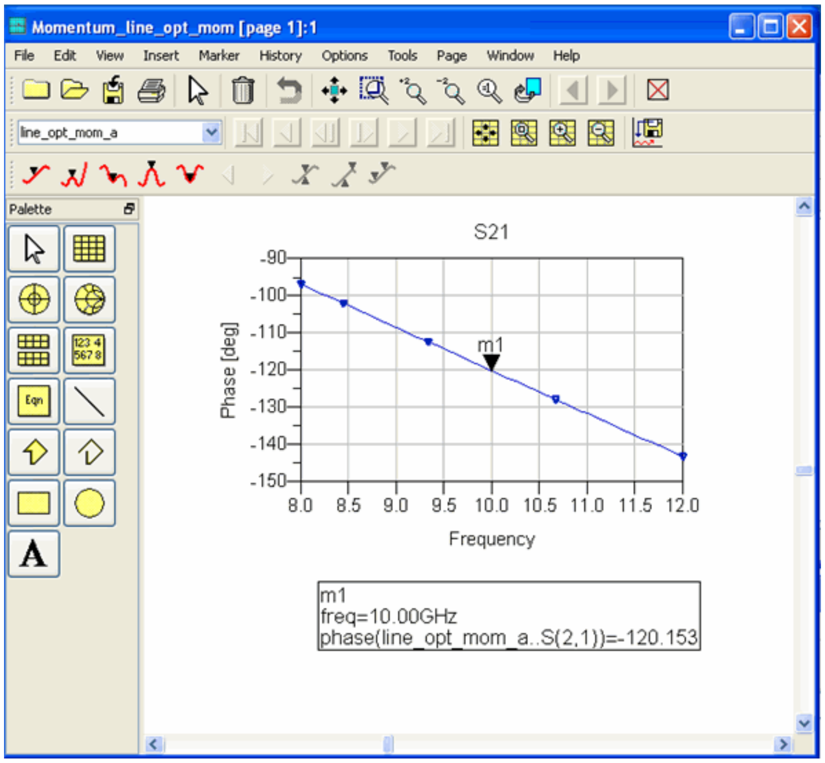


8. You can now run the optimization. When finished select **Simulation** > **Update Optimization Values** to update the parameter value.

Analysis

The results are displayed in "*Momentum_line_opt_mom.dds*".

Figure 1: Optimization results in Data Display.


Momentum_line_opt_mom [page 1]:1
File Edit View Insert Marker History Options Tools Page Window Help
line_opt_mom_a
Palette
S21
Phase [deg]
Frequency
m1

m1
freq=10.00GHz
phase(line_opt_mom_a..S(2,1))=-120.153

Remove the optimization setup and goal from the schematic. You now have the optimized value for **L**.

Optimization of A Microstrip Resonator

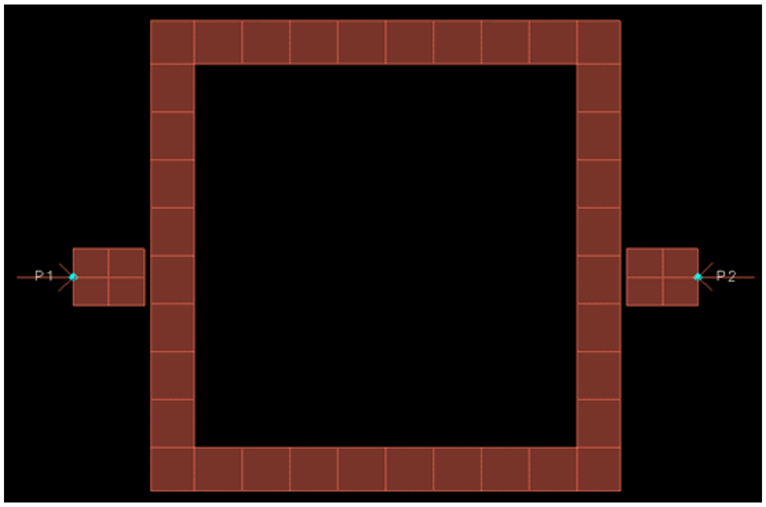
Location: $HPEESOF_DIR/examples/Momentum/Optimization/Microstrip_resonator_wrk

Objective

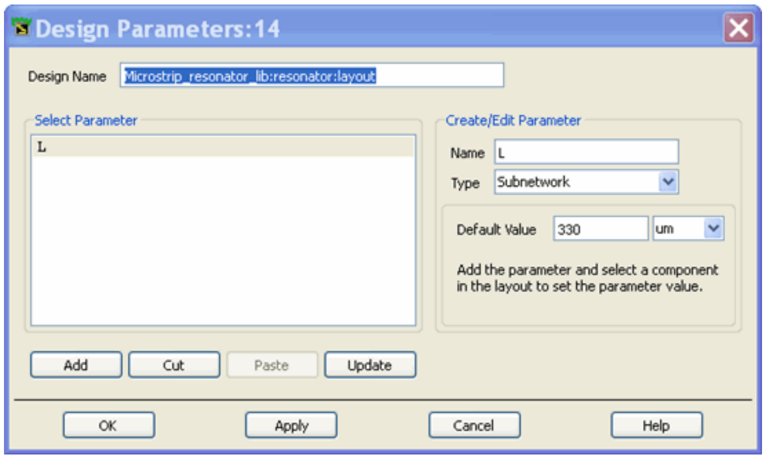
This example illustrates the optimization of a microstrip resonator to meet the specified resonance frequency. The objective is to optimize the length of the resonator for a center frequency of 65 GHz.

Setup

1. The resonator is built on a 100um thick GaAs susbtrate. The center frequency of the nominal design (L= 330 um) is 88 GHz. The goal of the optimization is to adjust the resonator length **L** so that the center frequency is 65 GHz. The optimal value is L = 531.77 um.
2. Open *Resonator*



3. In the layout a nominal/perturbed parameter **L** is defined by selecting **EM** > **Component** > **Parameters**. This is shown in the following illustration.



Design Parameters:14

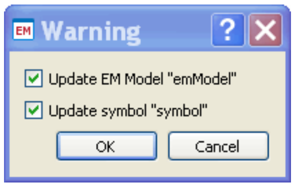
Design Name: Microstrip_resonator_lib:resonator:layout

Select Parameter: L

Create/Edit Parameter
Name: L
Type: Subnetwork
Default Value: 330 um

Add the parameter and select a component in the layout to set the parameter value.

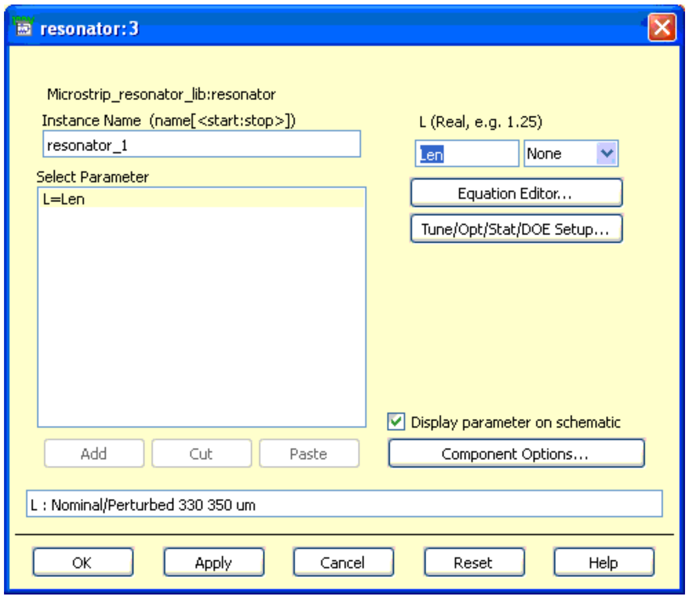
4. Next, the EM Model is created from the EM menu or from the EMsetup. Select **EM** > **Component** > **Create EM Model and Symobol** to open this dialog.



Warning

✓ Update EM Model "emModel"
✓ Update symbol "symbol"

5. The Em Model is then inserted in a Schematic window (right click on EM model and press place component). This is shown in "*resonator_opt*".
6. Double click the Em Model to open the below dialog. This is where parameter **L** is set for use in the optimization.


resonator:3

Microstrip_resonator_lib:resonator

Instance Name (name[<start:stop>])

resonator_1

Select Parameter

L=Len

L (Real, e.g. 1.25)

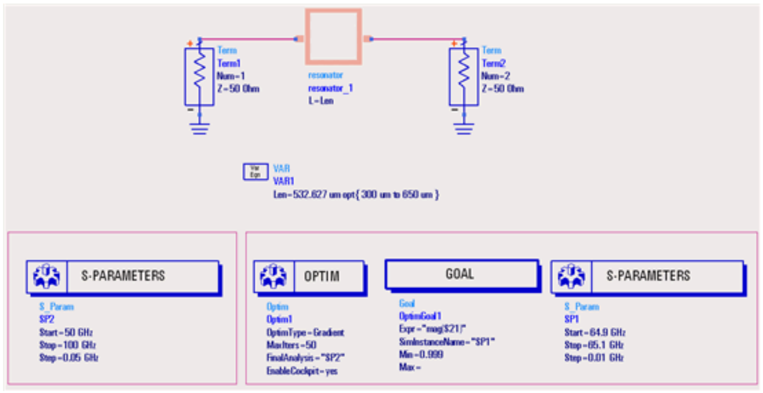
Len | None

☑ Display parameter on schematic

L : Nominal/Perturbed 330 350 um

7. An optimization *Setup* and *Goal* are added from the *Optim/Stat/Yield/DOE* palette. In this case: Goal: For this optimization different goals were used for different frequency ranges. This can be seen in "*resonator_optimization*", shown below.
   Optimization type used: gradient

> **Note**
> S-parameter terminations and a simulation setup were added from the "*Simulation-S_Param*" palette.



Len=532.627 um opt{ 300 um to 650 um }

S_Param
SP2
Start=50 GHz
Stop=100 GHz
Step=0.05 GHz

Optim
Optim1
OptimType=Gradient
MaxIters=50
FinalAnalysis="SP2"
EnableCockpit=yes

Goal
OptimGoal1
Expr="mag(S21)"
SimInstanceName="SP1"
Min=0.999
Max=

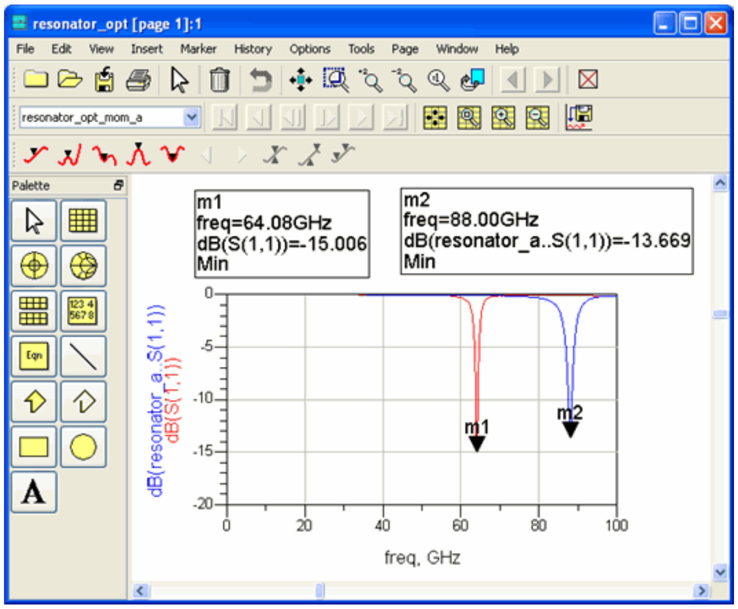
S_Param
SP1
Start=64.9 GHz
Stop=65.1 GHz
Step=0.01 GHz

8. You can now run the optimization. When finished select **Simulation** > **Update Optimization Values** to update the parameter value.

Analysis

The results are displayed in "*resonator_opt.dds*".

Figure 1: Initial and optimized resonance frequencies



m1
freq=64.08GHz
dB(S(1,1))=-15.006
Min

m2
freq=88.00GHz
dB(resonator_a..S(1,1))=-13.669
Min

Remove the optimization setups and goals from the schematic. You now have the optimized value for **L**.

Printed Dipole Antenna for Ultra High Frequency RFID Handheld Reader

Location: $HPEESOF_DIR/examples/Momentum/Antenna/RFID_Antenna_wrk

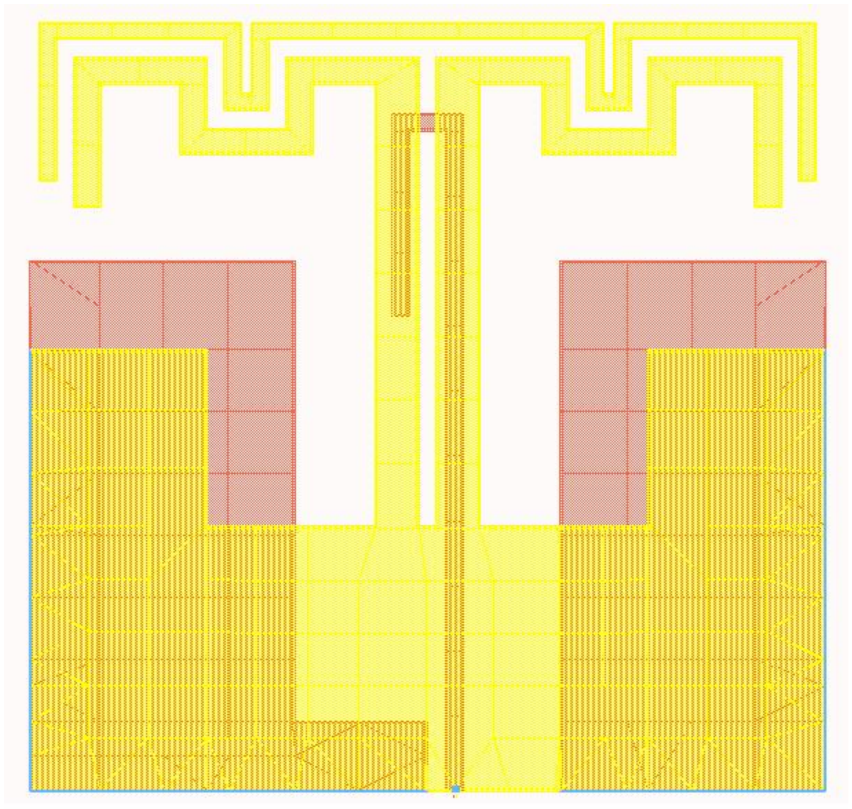
Objective

This example shows the Momentum simulation of a RFID (Radio Frequency Identification) handheld reader antenna in the UHF band. This example demonstrates modeling of a 3D planar antenna in Momentum. The connecting walls are modeled using conductive vias.

Setup

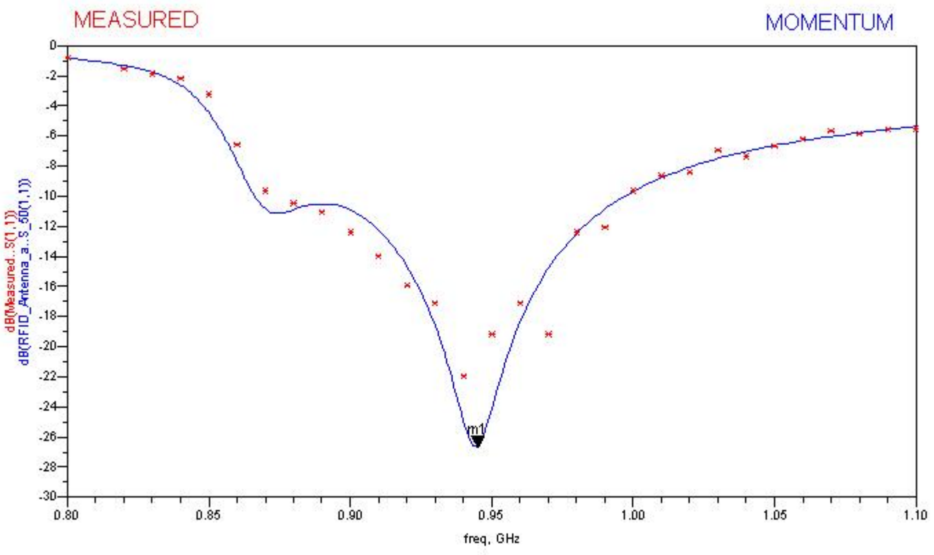
1. "RFID_antenna" shows the layout of the RFID antenna. The antenna consists of a microstrip-to-stripline transition, a meandered driven dipole, a closely coupled parasitic element and a folded finite size ground plan.
2. "RFID_Antenna.dds" shows the return loss. The S-parameter curve shows resonance at 945 MHz. Comparison between Momentum simulation and measurements is also shown.
3. "RFID_Antenna_FF_945MHz_phi0" shows the antenna far field pattern.

Figure 1: RFID Antenna Layout in Momentum



Analysis

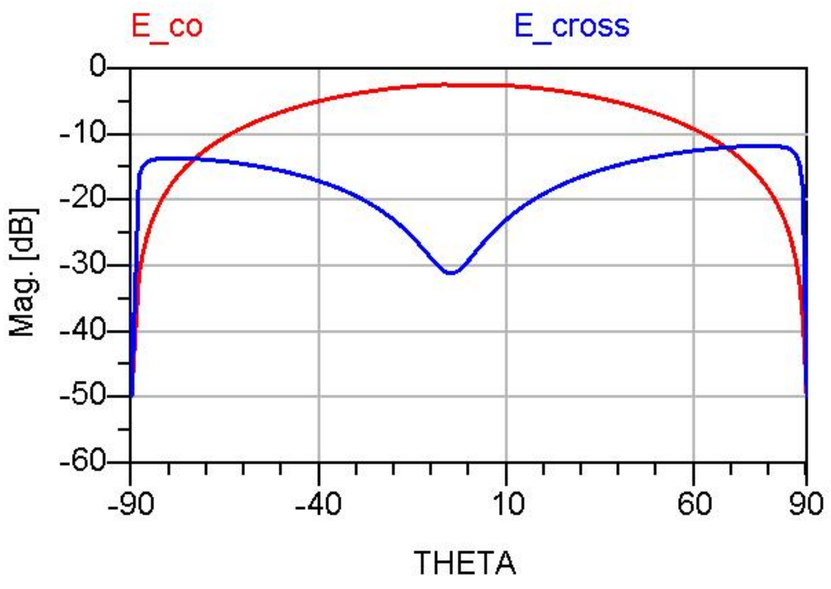
Figure 2: Simulated and Measured Return Loss



Eqn S11_Momentum=dB(RFID_Antenna_a..S_50(1,1))

Eqn S11_Measured=dB(RFID_Antenna_a..S(1,1))

Figure 3: Antenna Radiation Patter (Linearly Polarized)



References

1. "A Printed Dipole Antenna for Ultra High Frequency (UHF) Radio Frequency Identification (RFID) Handheld Reader", Ren-Ching Hua and Tzyh-Ghung, IEEE Transaction on Antenna and Propagation, Vol 55, No 12 , December 2007, pp 3742-3745.

Proximity Coupled Two Semi-Circular Patch Radiators

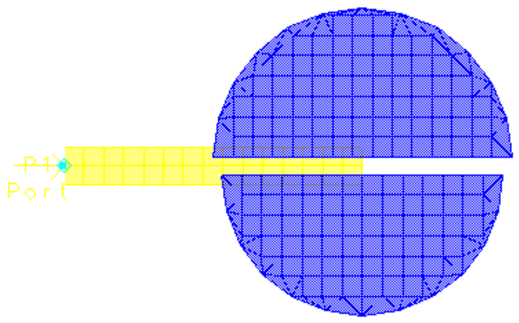
Location: /examples/Momentum/Antenna/Double_Patch_wrk

Objective

This dual-radiator example is used to show the generality of the Momentum mesh maker. Momentum makes meshes that conform to the drawn geometry within the resolutions specified by the user.

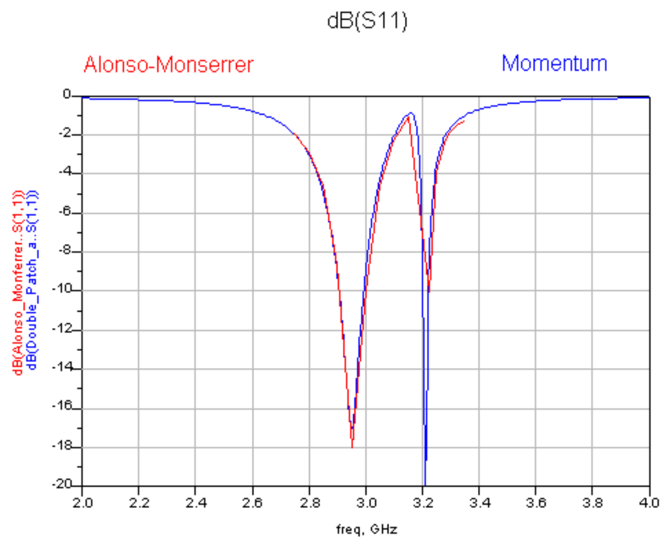
Setup

1. In "Double_Patch", two semi-circular patches of different diameters are excited by a microstrip line through EM coupling.
2. "Double_Patch.dds" shows the S-parameter curve. "Double_Patch_FF_2_943Ghz_phi0.dds" and "Double_Patch_FF_3_203Ghz_phi0.dds" shows the far field in the XZ-plane at the two resonance frequencies, respectively.



Analysis

Figure 1: Return loss



At the first resonance, the current is predominantly excited at the larger semi-circular patch. The second resonance corresponds with the resonance of the smaller semi-circular patch.

Notes

- Reference: "Green's Functions Analysis of Planar Circuits in a Two_Layer Grounded Medium", F. Alonso-Monferrer, A. Kishk, A. Glisson, IEEE Trans. on APS, vol. 40, no. 6, pp. 690-696, June 1992.

RF Board Simulation Comparison between Momentum RF and Momentum Microwave

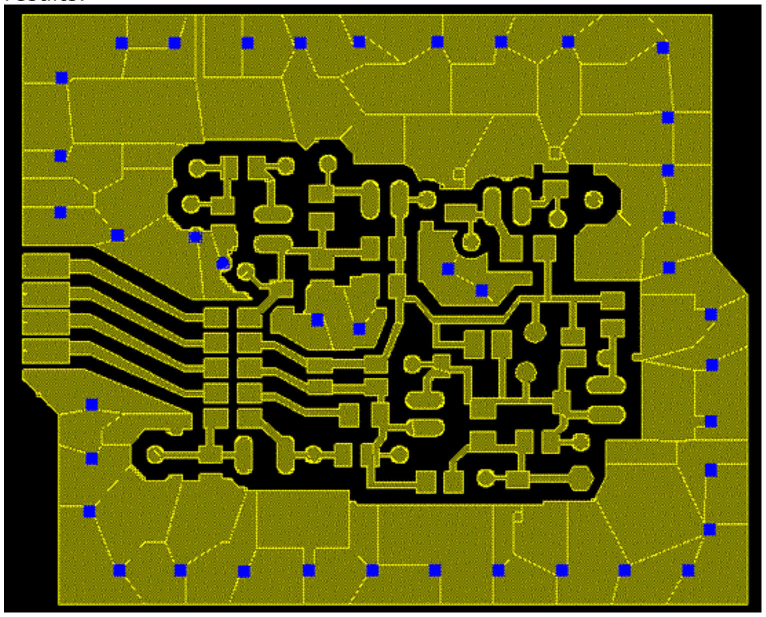
Location: $HPEESOF_DIR/examples/Momentum/RF/RFBoard_wrk

Objective

Through the comparison between the Momentum RF and Momentum Microwave simulation of a complicated RF Board layout, this example verifies the accuracy of Momentum RF for typical board level applications.

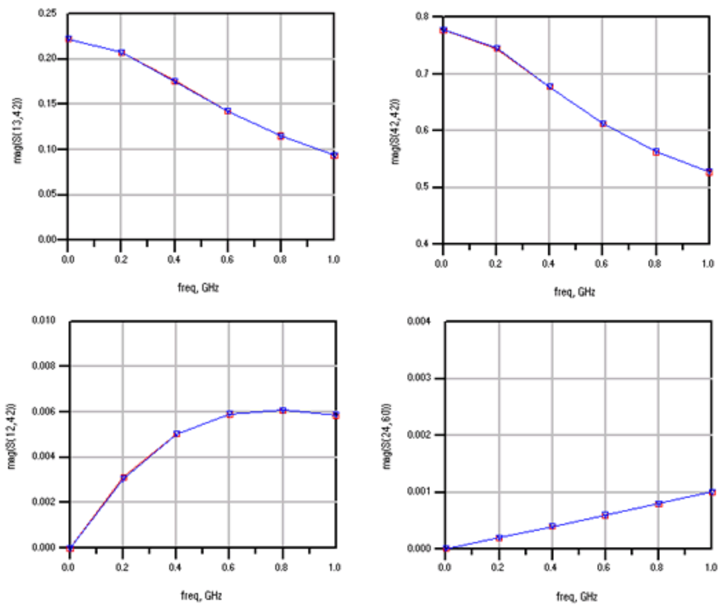
Setup

"board_momRF" shows the mesh used by Momentum RF, "board_momMW" shows the mesh used by Momentum Microwave, "RFBoard.dds" compares the RF and Microwave results.



Analysis

Figure 1: Comparison between Momentum RF and Momentum Microwave simulation



RF Board with Holes in the Ground

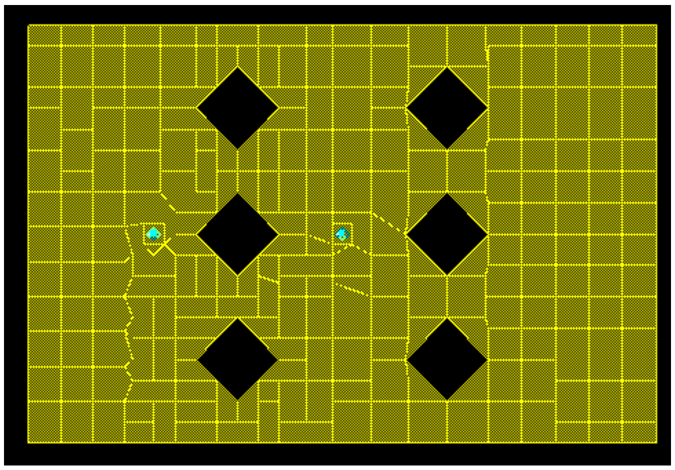
Location: $HPEESOF_DIR/examples/Momentum/RF/PowerGround_wrk

Objective

This example shows the Momentum RF simulation for RF power ground plane and compares the results with measurement data.

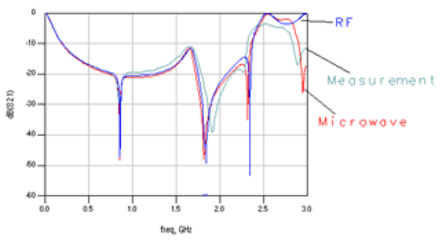
Setup

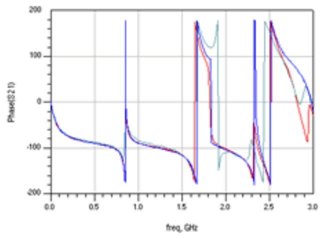
Two RF test boards (slot1 and diamond1) are simulated with Momentum in RF mode, one is with rectangular slots and the other with diamond slots in the ground. The results are compared (in the data displays slot1.dds and diamond1.dds) with Momentum Microwave and measurement data.



Analysis

Figure 1: Comparison on S21 between simulation and measurements





Simple Microstrip Patch Antenna

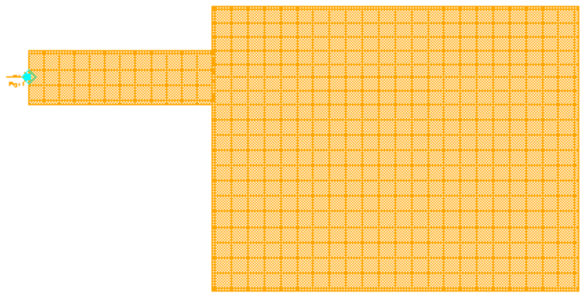
Location: /examples/Momentum/Antenna/Single_patch_wrk

Objective

This example shows a Momentum simulation of a simple patch antenna with direct microstrip line feeding.

Setup

1. "Single_patch" shows the layout of the microstrip line fed rectangular patch antenna.
2. "Single_patch.dds" shows the return loss comparison with measurement. "Single_patch_FF_JX30_phi0.dds" shows the far field pattern for the Jx(3,0) mode. "Single_patch_FF_JY01_phi0.dds" shows that for JY(0,1) mode.



Analysis

Figure 1: Measured and simulated return loss

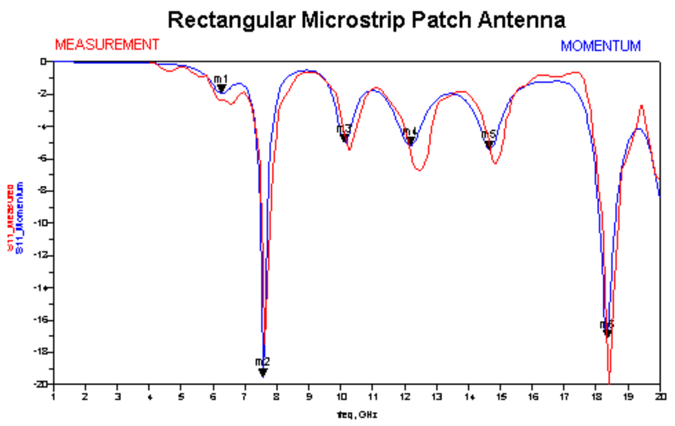
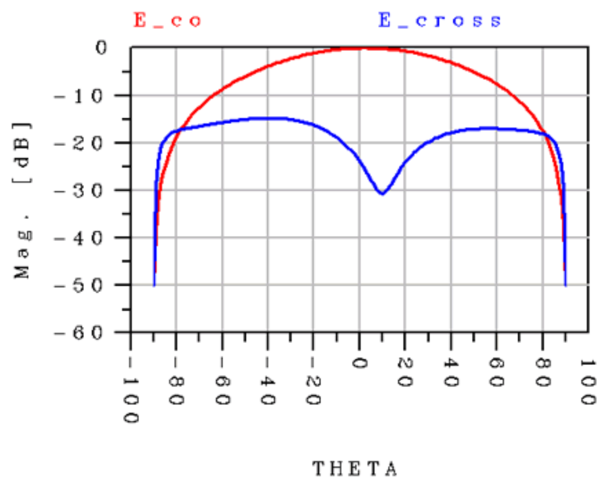


Figure 2: Radiation pattern at 7.685GHz due to Jy(0,1) mode



Notes

- Reference: "Feeding Structure Contributions to Radiation by Patch Antennas with Rectangular Boundaries", Shih-Chang Wu, N. Alexopoulos, O. Fordham, IEEE Trans. on APS, vol. 40, no. 10, pp. 1245-49, October 1992.
- The currents on the patch are in general a superposition of an X-directed and a Y-directed mode. Each mode is characterized by its variation along the X and Y direction. For example, the Jx (2,0) mode means that the amplitude of the X-directed current follows a complete period of a sine/cosine along X and is constant along the Y-direction.
- The S-parameters don't reveal the information about the eigenmode that is predominantly excited on the patch. Each resonance corresponds with a certain eigenmode, for example, a certain surface current distribution. By visualizing the current, one can easily verify which mode is excited and make the link with the far field that will be produced.

Simulation of A Ball Grid Array with 96 Solder Balls

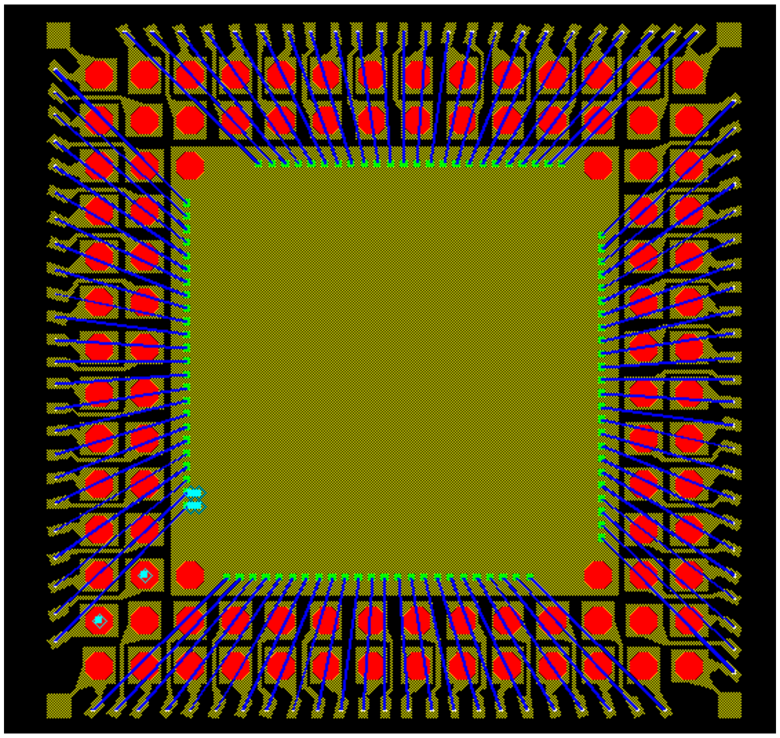
Location: $HPEESOF_DIR/examples/Momentum/RF/BGA96_wrk

Objective

This workspace illustrates the Momentum RF simulation of a Ball Grid Array with 96 solder balls. It shows the Momentum RF capability of dealing with large size, multilayer complicated structure.

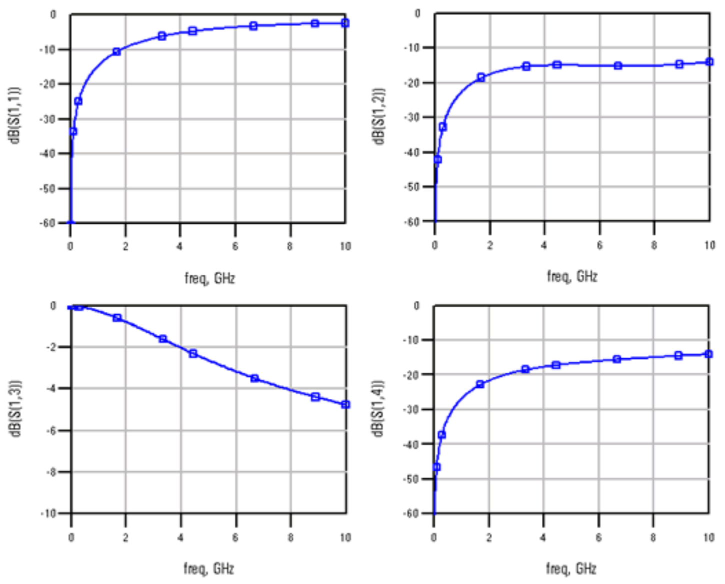
Setup

This Ball Grid Array package with 96 balls is simulated as a whole structure with Momentum RF. Ports are defined at the board level and at the chip level. Ground port references are needed at the chip level to include the ground impedance distribution in the S-parameters. Simulation results for two neighboring ball-bondwire traces are in bga96.dds.



Analysis

Figure 1: Simulated S-parameters



Results include the port reflection at port 1 (S11), trace loss(S13), and isolation between two traces (S12 & S14).

Simulation of a Balun

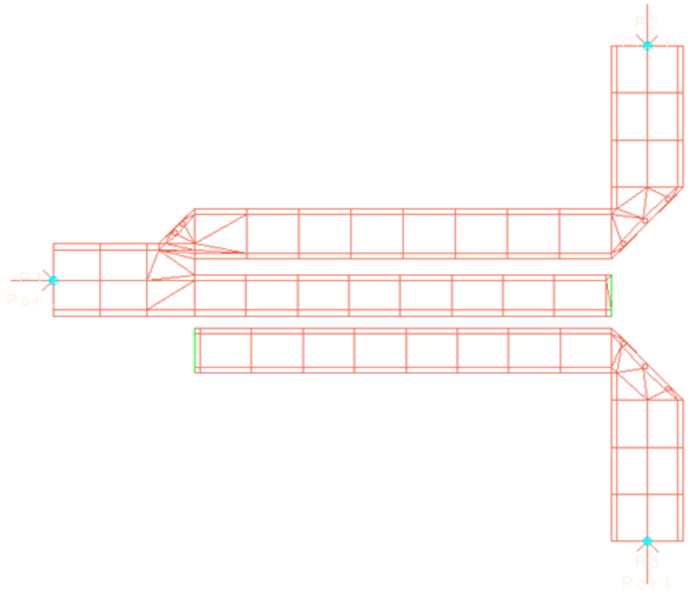
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Balun_wrk

Objective

This example illustrates the simulation of a coupled line balun.

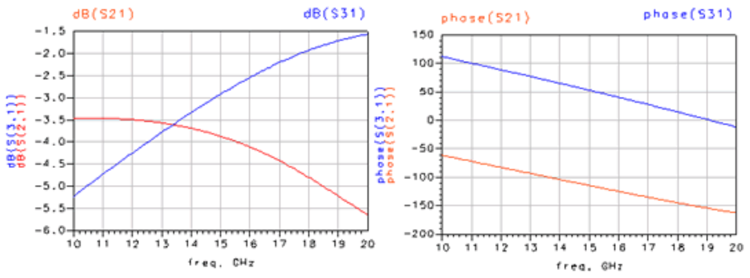
Setup

The Balun is built on a 20 mils thick substrate with dielectric constant of 10. Two polyline sheet vias are used. The structure is meshed at 20 GHz with 30 cells per wavelength and edge mesh on.



Analysis

Figure 1: Transmission properties of the Balun



Simulation of Coupled Lines on Printed Circuit Board

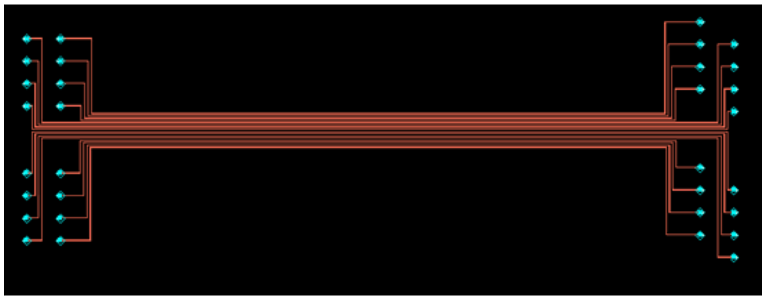
Location: $HPEESOF_DIR/examples/Momentum/RF/PCBlines_wrk

Objective

This example illustrates the Momentum RF simulation of a 16 coupled lines on a printed circuit board up to 2 GHz.

Setup

The PCB is a FR4 board. The transmission and cross-talk S-parameter results are shown in PCBlines16.dds



Analysis

Figure 1: Transmission properties

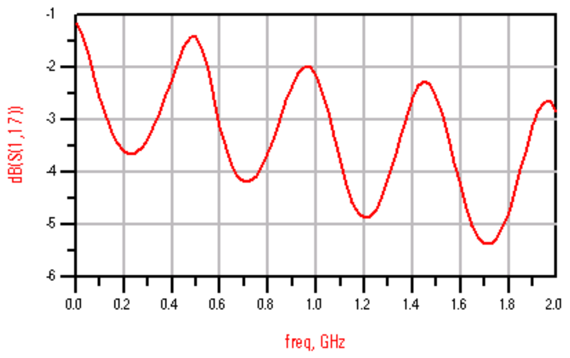
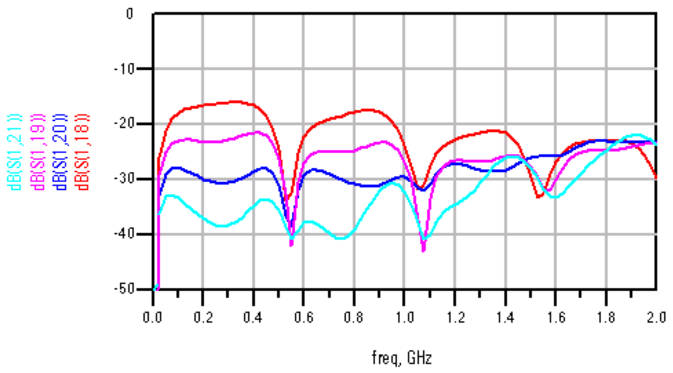


Figure 2: Cross talk between lines



Slanted Coupled Line Filter

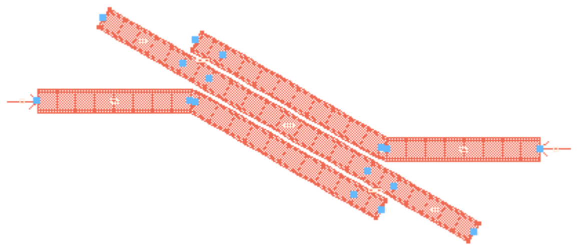
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Coupled_line_filter_wrk

Objective

Design of a classic coupled line filter as a validation example for Momentum.

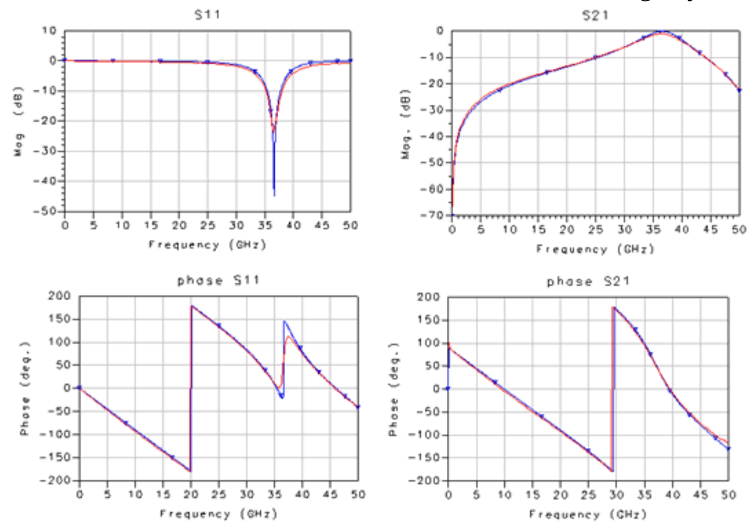
Setup

The filter is built on a 100um thick GaAs substrate. The layout can be generated from the schematic. The filter is meshed with 30 cells per wavelength at 30GHz with arc facet angle = 45 degrees and transmission Line Mesh ON. The Adaptive Frequency Sampling (AFS) feature uses 15 frequencies to cover frequency range from DC to 50 GHz.



Analysis

Figure 1: Comparison between Momentum simulation and measurements



Slot Dipole Antenna with CPW Feeding

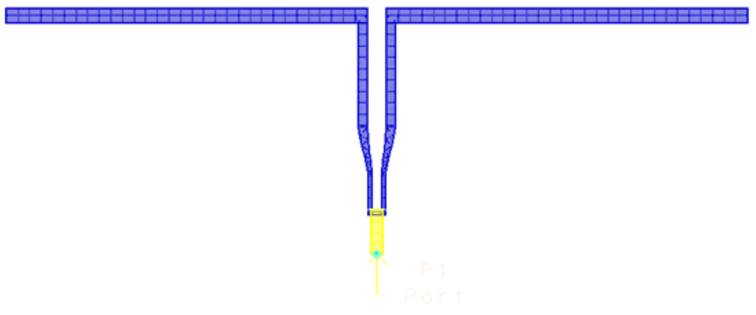
Location: /examples/Momentum/Antenna/Slot_dipole_wrk

Objective

This example shows a Momentum simulation of a slot dipole antenna with coplanar waveguide (CPW) feeding. The effect of the mesh on the solution accuracy is studied.

Setup

1. "Dipole" shows the layout of the CPW fed slot dipole.
2. "dipole_us" shows the same dipole with a microstrip line feeding, subdivision is 20/wavelength. "dipole_us_denseMesh" uses a dense mesh with edge mesh on, subdivision is 40/wavelength.



Analysis

Figure 1: Effect of feeding mechanism on return loss

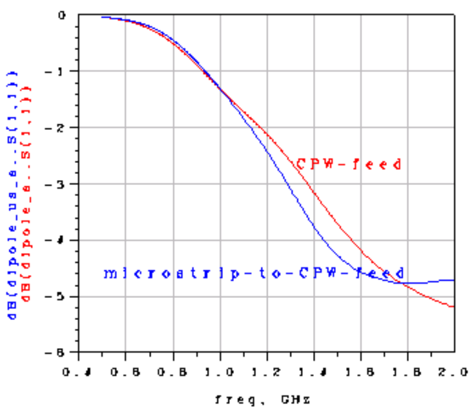
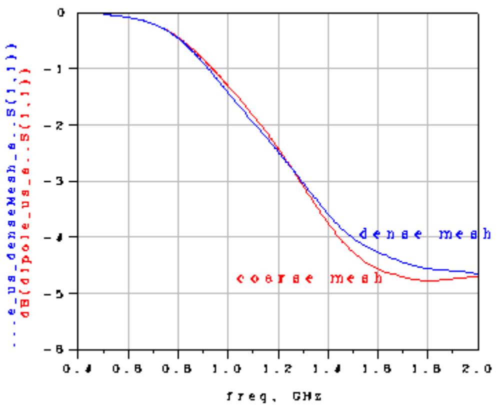


Figure 2: Effect of mesh on return loss



Notes

- In the example, the CPW itself can be fed through a microstrip line. It is found that both feeding mechanisms are quite similar, as long as the microstrip line impedance is close to that of CPW.
- The coarse mesh uses 20 cells/wavelength, the fine mesh uses 40 cells/wavelength with an edge mesh.

SMD and Delta Gap Port Calibration

Location: $HPEESOF_DIR/examples/Momentum/emcktcosim/SMD_wrk

Objective

This workspace illustrates the usage and meaning of the DeltaGap, SMD port types in Momentum. The test structure is a simple coupled line structure (see cell 'coupledLines'), consisting of two parallel lines:

- One line is setup to use a component port (DeltaGap, SMD)
- The other line will serve as a reference line.

In this example, the component port will be short-circuited. Then we will compare the total inductance of the line with the shorted component port and the inductance of the reference line.

Setup

Different EM setups and EM models are generated for the coupled line structure.

Momentum simulations:
emSetup_MoM_DeltaGap: uses DeltaGap port for pins 5 and 6.
emSetup_MoM_SMD: uses SMD port for pins 5 and 6.
emSetup_MoM_NoCalib: no calibration is used for the pins 5 and 6.

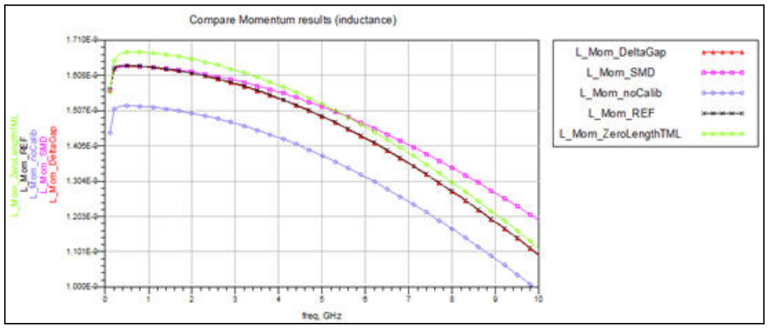
FEM simulations:
emSetup_FEM_DeltaGap: uses DeltaGap port for pins 5 and 6.
emSetup_FEM_SMD: uses SMD port for pins 5 and 6.
emSetup_FEM_NoCalib: no calibration is used for the pins 5 and 6.

This workspace contains the following designs:

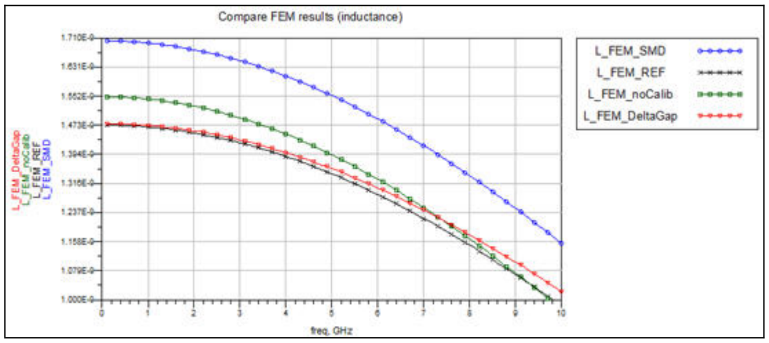
- uses_coupledLines_MoM_SMD: uses the SMD port setup and a line model to fill the gap between the SMD pins. The SMD port model in Momentum by construction does not include the physical effects of the gap, so to compare the results with the parallel line, the line model has to be added.
- uses_coupledLines_MoM_DeltaGap: uses the DeltaGap port setup and an ideal connect to short circuit the gap. The DeltaGap port model in Momentum by construction includes the physical effects of the gap, so to compare the results with the parallel line.
- uses_coupledLines_MoM_NoCalib: does not use the SMD or DeltaGap setup.

Analysis

Momentum simulation results are compared in below graph. Observation when comparing the inductance plots is that both the SMD and DeltaGap sources, give good agreement with the reference line results.



Similar comparison is done for the FEM simulation results. Here the conclusion is that, in the current ADS version (also see the documentation), the SMD port setup does not eliminate the gap filling effect and therefore is giving an incorrect result, whereas the DeltaGap source is producing good agreement.



Spiral Inductor on Silicon Substrate

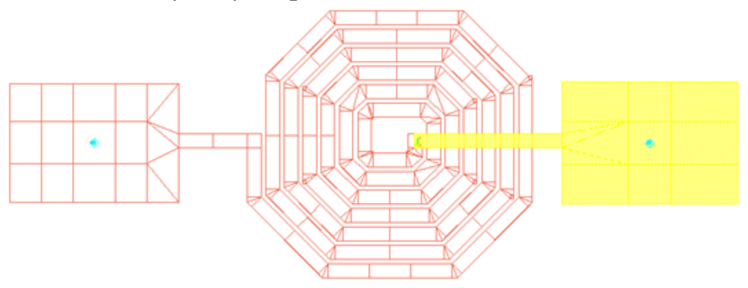
Location: $HPEESOF_DIR/examples/Momentum/Microwave/SPIRAL_wrk

Objective

This example illustrates a spiral inductor on a conductive Silicon substrate. Metal loss is also included.

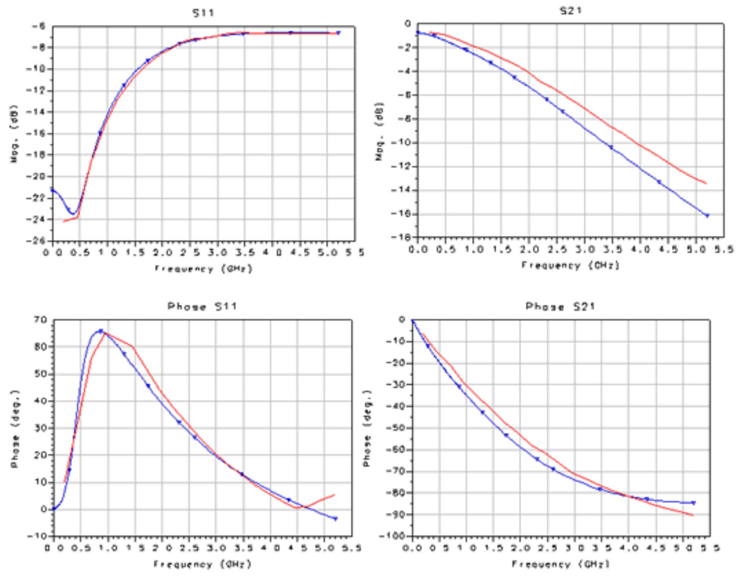
Setup

The spiral is meshed with 30 cells per wavelength at 5.2GHz. 13 frequency points are used to cover the frequency range from 0 to 6.0GHz.



Analysis

Figure 1: Comparison between simulated and measured S-parameters



Spiral Inductor with Hole in The Ground Plane

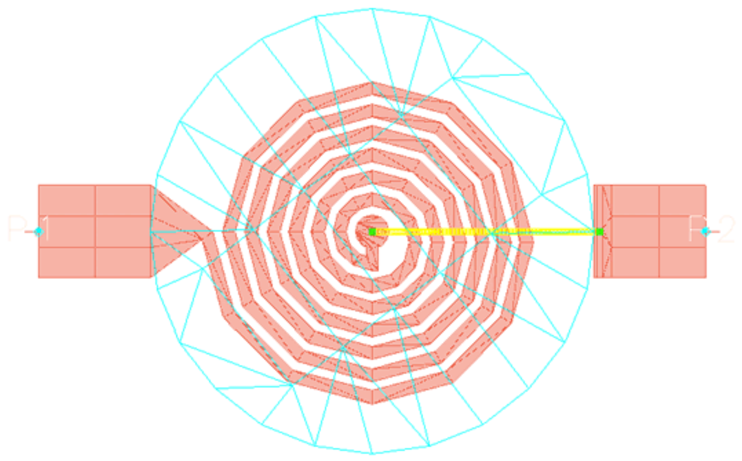
Location:
$HPEESOF_DIR/examples/Momentum/Microwave/Spiral_with_hole_in_ground_wrk

Objective

This example illustrates a spiral inductor with holes in the ground plane in order to reduce the capacitance and increase the resonance frequency of the inductor.

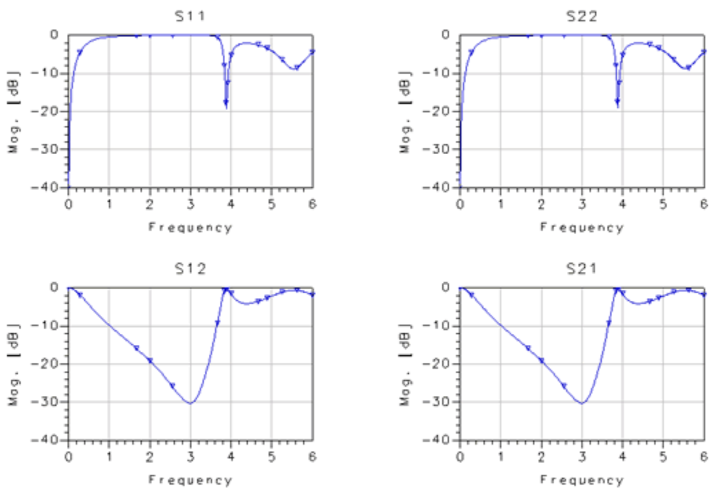
Setup

The spiral is built on a 15mil sapphire substrate, the air bridge is 10mil above the spiral layer. In the layout, hole in the ground plane is produced with a slot layer and it is meshed together with the spiral metal. The mesh is at 30 cells per wavelength at 6GHz.



Analysis

Figure 1: Simulated S-paramters. First resonance is around 3GHz



Spiral Splitter on GaAs Substrate

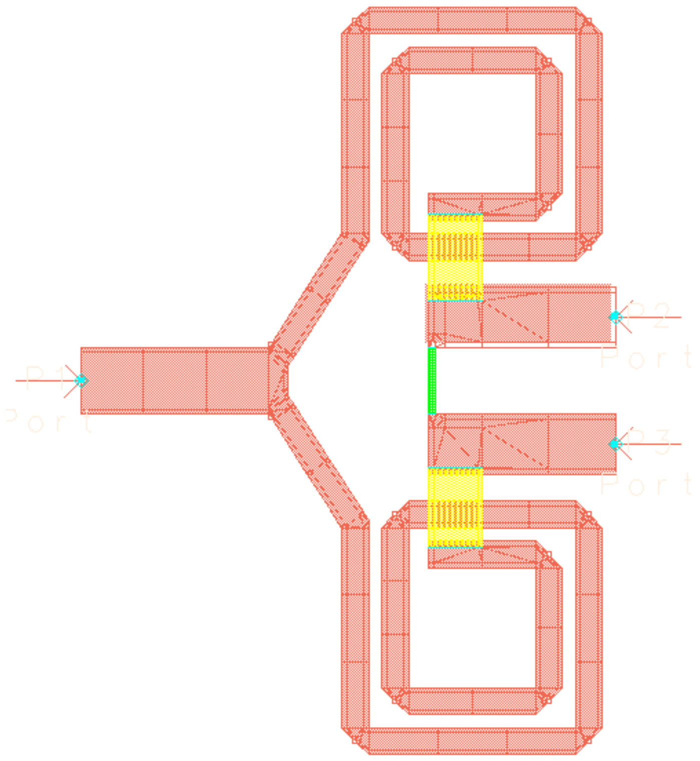
Location: $HPEESOF_DIR/examples/Momentum/Microwave/SPIRAL_Splitter_wrk

Objective

This example illustrates a pair of spiral inductors used as a splitter in a Wilkinson-type structure on GaAs substrate.

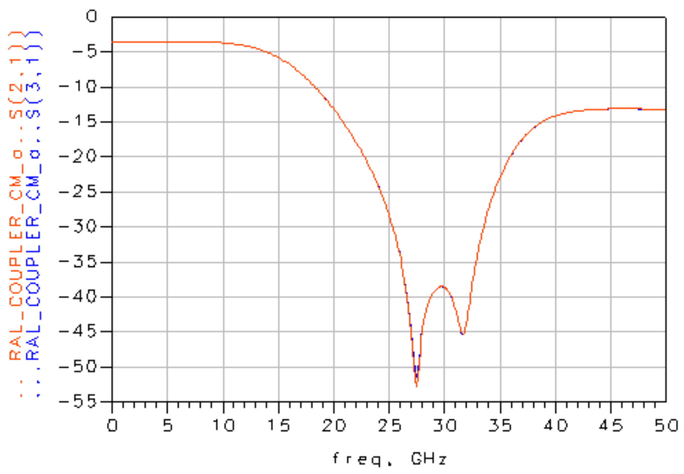
Setup

The spiral splitter is built on a GaAs substrate, a second conductor and vias are used to make an air bridge. A resistor film is also used. The structure is meshed with 20 cells per wavelength at 50GHz.



Analysis

Figure 1: Simulated S21 and S31



Stripline Low-Pass Filter

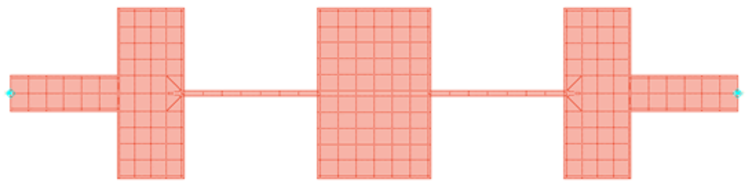
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Stripline_LPF_wrk

Objective

This example illustrates the simulation of a stripline low-pass filter with and without box enclosure.

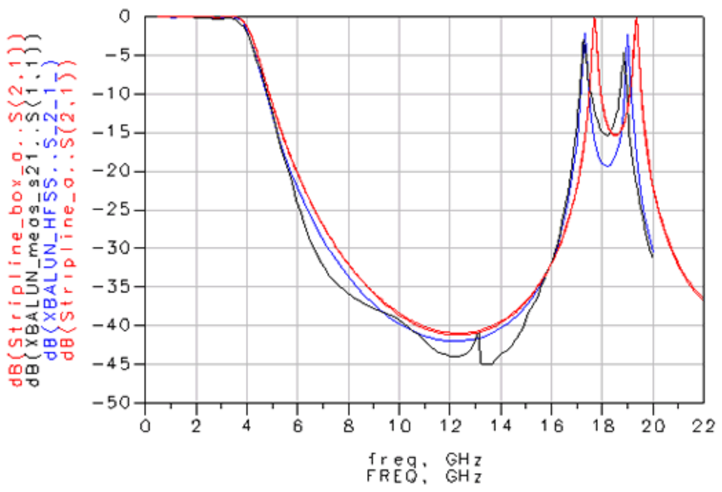
Setup

The filter is meshed with 20 cells per wavelength at 15GHz with edge mesh on. The filter is built on Duroid substrate with dielectric constant of 2.2 and loss tangent of 0.0001.



Analysis

Figure 1: Transmission properties of the low-pass filter



Curves are from Momentum, HFSS and measurements. The enclosure effect is significant at high frequency range.

Strip Lines with Different Via Structures

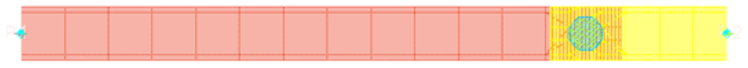
Location: $HPEESOF_DIR/examples/Momentum/Microwave/via_structure_wrk

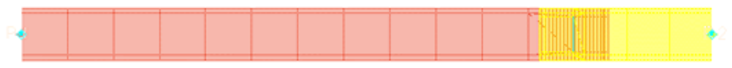
Objective

This example investigates the difference between using round via and simplified sheet via in Momentum simulation of via stubs.

Setup

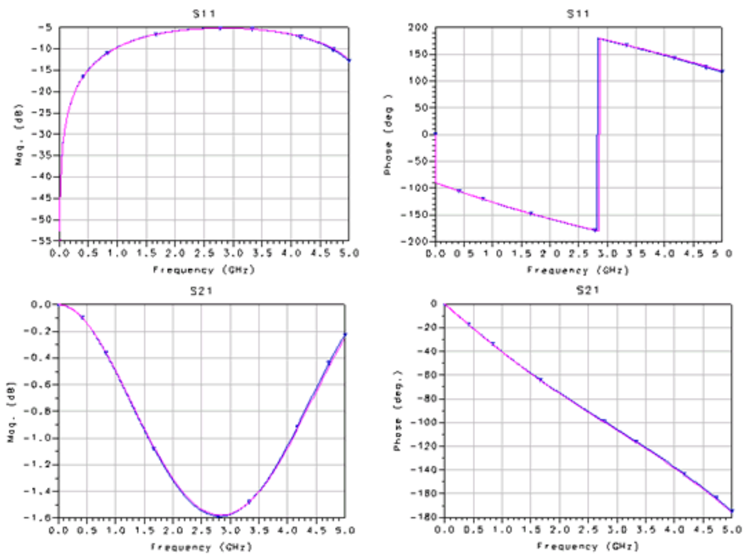
This example illustrates the use of vias. When via size is relatively small in wavelength, there is not much difference in accuracy using a sheet via (using a polyline) as compared to a circular via, but there is large saving in simulation time due to the simple mesh. The test strip structure is built on a two layer structure. The first layer is 254um thick with a dielectric constant of 9.6. The second layer is 304.8 thick with dielectric constant of 9.8. The mesh is with 30 cells per wavelength at 5GHz with edge mesh on.





Analysis

Figure 1: Comparison of S-parameters using sheet via and round via



Sweep Substrate Parameters using DataFileList

Location:$HPEESOF_DIR/examples/Momentum/emcktcosim/Sweep_Substrate_Parameters_wrk

Objective

By using the **DataFileList** Component from the Simulation-Batch Palette, you can sweep the substrate Parameters like thickness of substrate/Dielectric constant of the substrate.

Setup

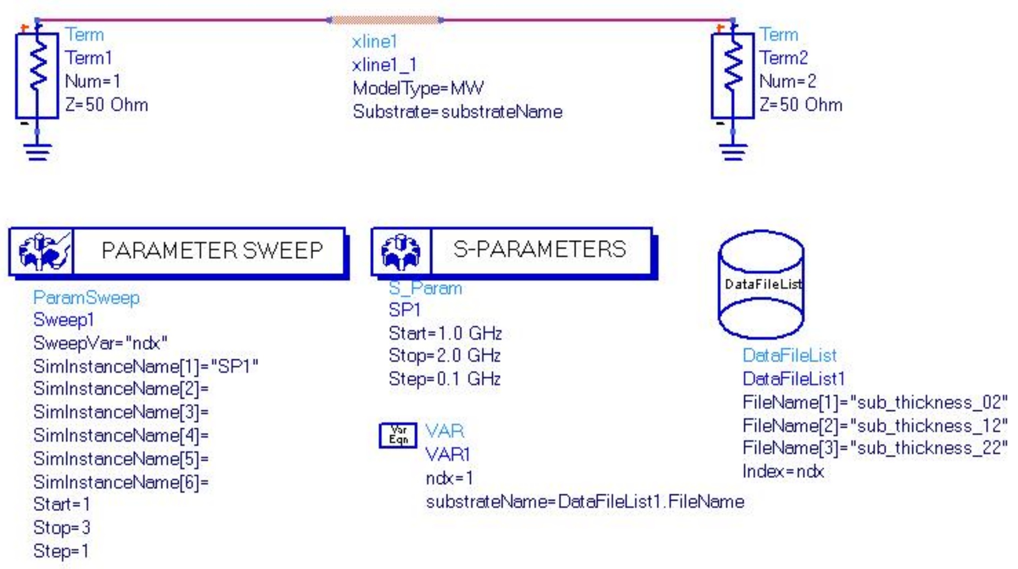
In this workspace,the use of the Momentum EM-Circuit co-simulation feature is utilized to sweep the substrate parameters of a simple microstrip line. The simulation setup uses the following:

- Layout component of a Microstrip line and a substrate name variable for sweeping the substrate files
- The substrate files with different substrate parameters are saved separately from the Layout
- The saved substrate files are inturn swept in schematic using 'DataFileList' component

- Two separate designs are illustrated to sweep the substrate thickness(Sweep_Substrate_Thickness) and Dielectic constant(Sweep_Substrate_Dk

Analysis

Figure 1: Schematic for Sweeping Substrate Thickness



Term
Term1
Num=1
Z=50 Ohm

xline1
xline1_1
ModelType=MW
Substrate=substrateName

Term
Term2
Num=2
Z=50 Ohm

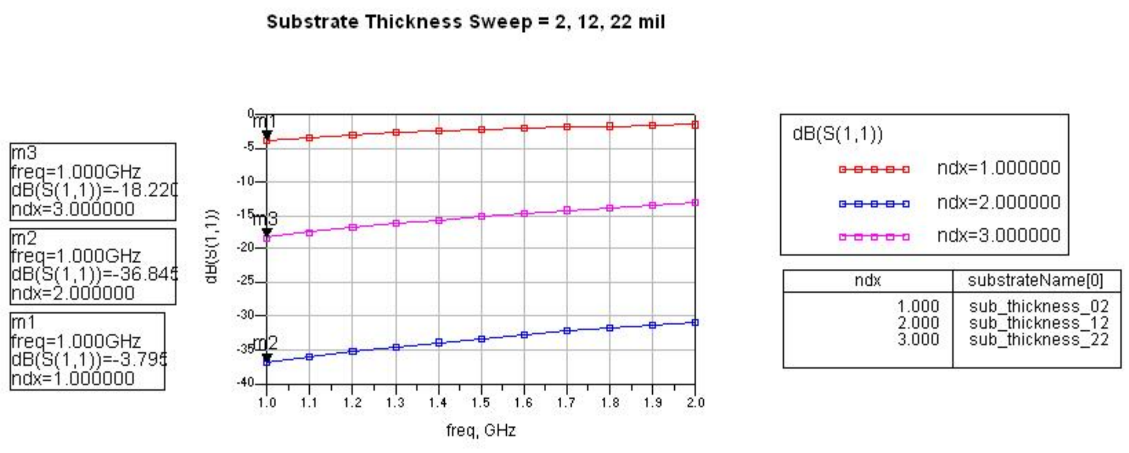
PARAMETER SWEEP
ParamSweep
Sweep1
SweepVar="ndx"
SimInstanceName[1]="SP1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1
Stop=3
Step=1

S-PARAMETERS
S_Param
SP1
Start=1.0 GHz
Stop=2.0 GHz
Step=0.1 GHz

VAR
VAR1
ndx=1
substrateName=DataFileList1.FileName

DataFileList
DataFileList
DataFileList1
FileName[1]="sub_thickness_02"
FileName[2]="sub_thickness_12"
FileName[3]="sub_thickness_22"
Index=ndx

Figure 2: Results for Sweeping Substrate Thickness



m3
freq=1.000GHz
dB(S(1,1))=-18.22
ndx=3.000000

m2
freq=1.000GHz
dB(S(1,1))=-36.845
ndx=2.000000

m1
freq=1.000GHz
dB(S(1,1))=-3.795
ndx=1.000000

ndx	substrateName[0]
1.000	sub_thickness_02
2.000	sub_thickness_12
3.000	sub_thickness_22

Figure 3: Schematic for Sweeping Dielectric Thickness

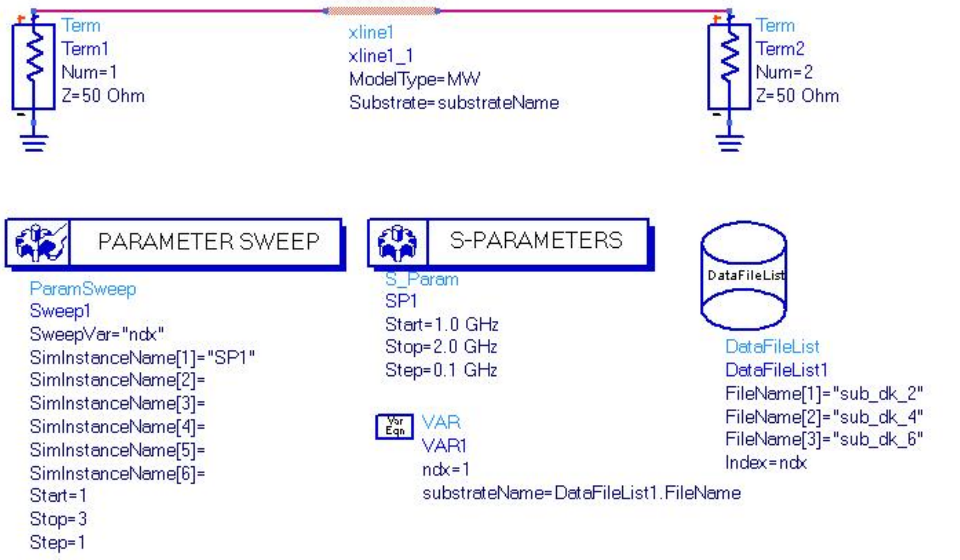
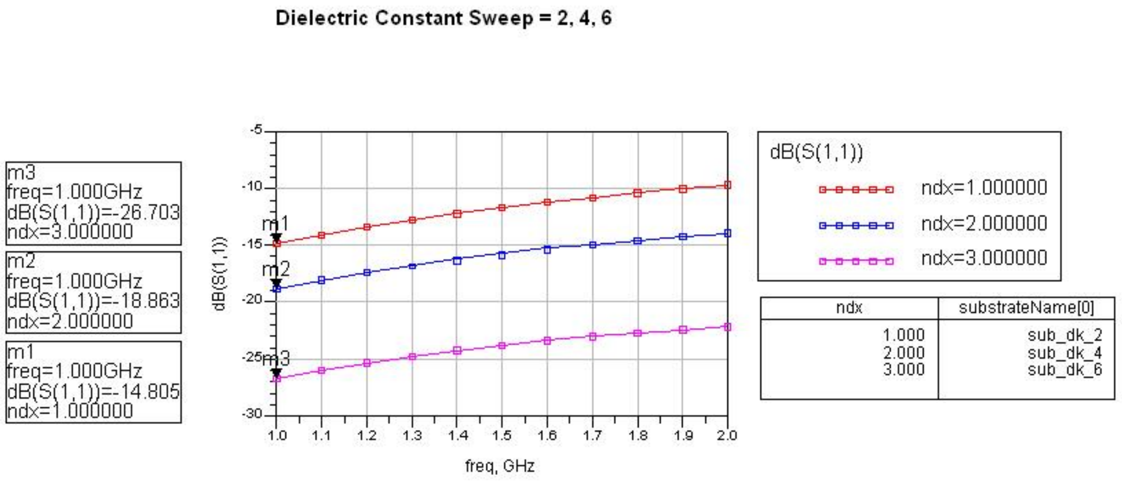


Figure 4: Results for Sweeping Dilectric Thickness



ndx	substrateName[0]
1.000	sub_dk_2
2.000	sub_dk_4
3.000	sub_dk_6

Thick Conductor Spiral

Location: $HPEESOF_DIR/examples/Momentum/Microwave/ThickCond_Spiral_wrk

Objective

This example illustrates the modelling of currents in thick conductors. The difference in results are displayed in spiral.dds

Setup

Two metallization layers are simulated as thick conductor.

layer "cond" : thick conductor (up)
layer "cond2" : thick conductor (down)

The simulation is started from schematic, using a Momentum component.
The resistance and inductance are shown in spiral.dds

Figure 1: double_spiral - layout of spiral inductor with thick metal

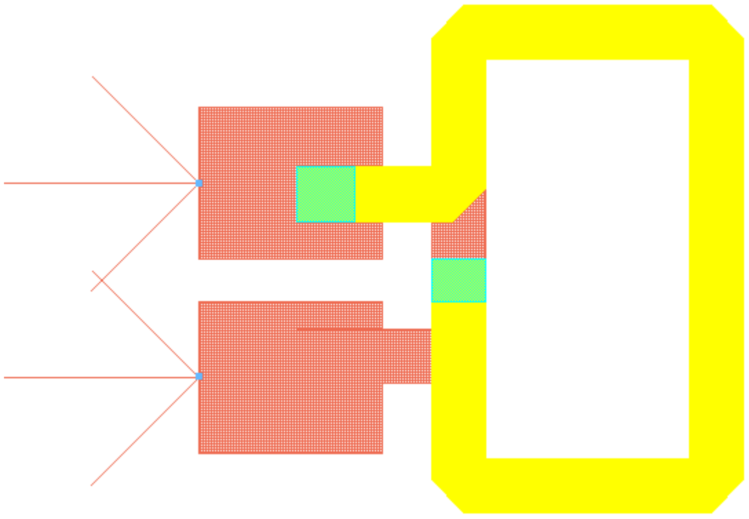
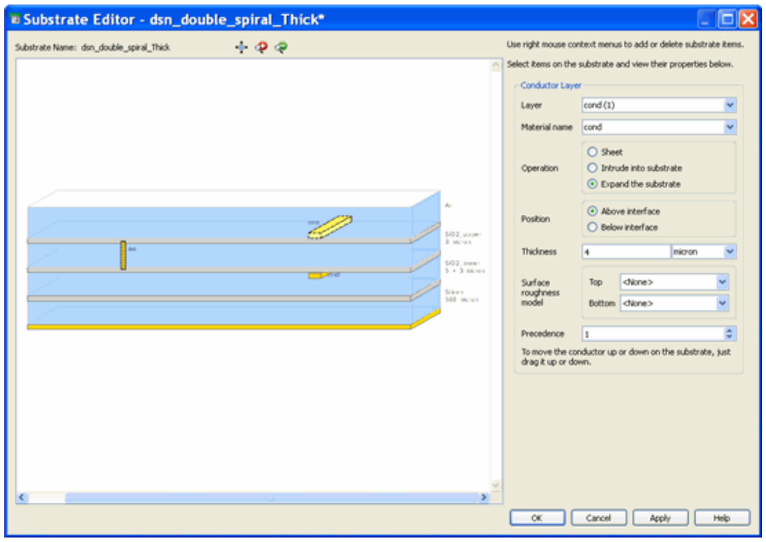


Figure 2: Substrate Definition - Metallization Layers with thick metal



Substrate Editor - dsn_double_spiral_Thick*

Substrate Name: dsn_double_spiral_Thick

Use right mouse context menus to add or delete substrate items.

Select items on the substrate and view their properties below.

Conductor Layer

Layer: cond (1)

Material name: cond

Operation: ( ) Sheet ( ) Intrude into substrate [✓ Expand the substrate]

Position: [✓ Above interface] ( ) Below interface

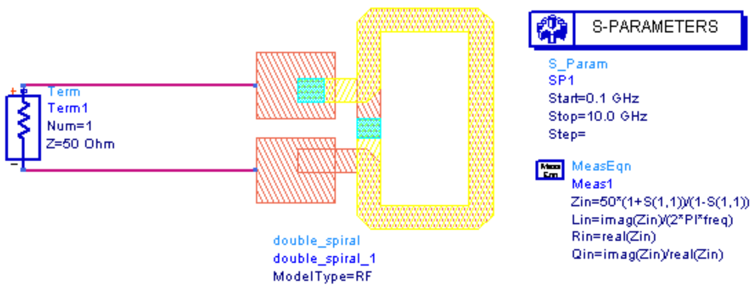
Thickness: 4 micron

Surface roughness model: Top <None> Bottom <None>

Precedence: 1

To move the conductor up or down on the substrate, just drag it up or down.

Figure 3: Double_Spiral_SP - Momentum component used in a simulation



Analysis

Figure 4: spiral.dds - resistance vs. freq of spiral inductor

Figure 4: spiral.dds - resistance vs. freq of spiral inductor

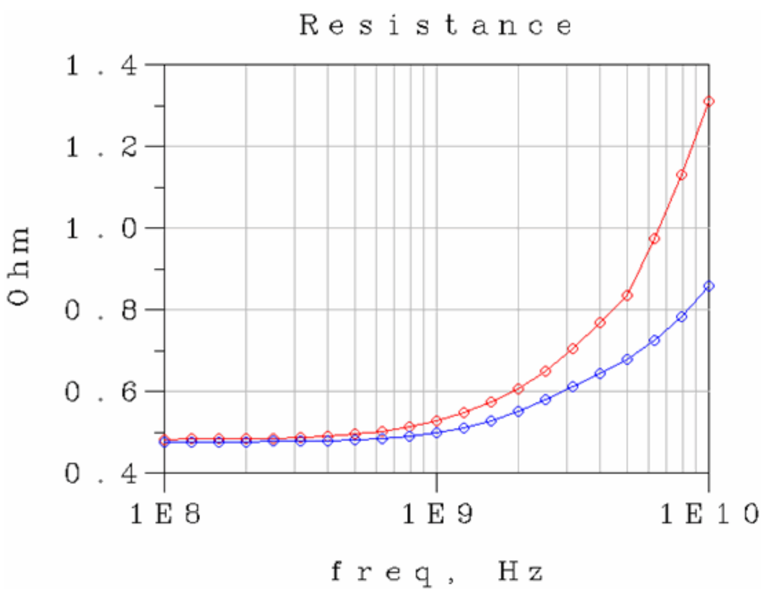
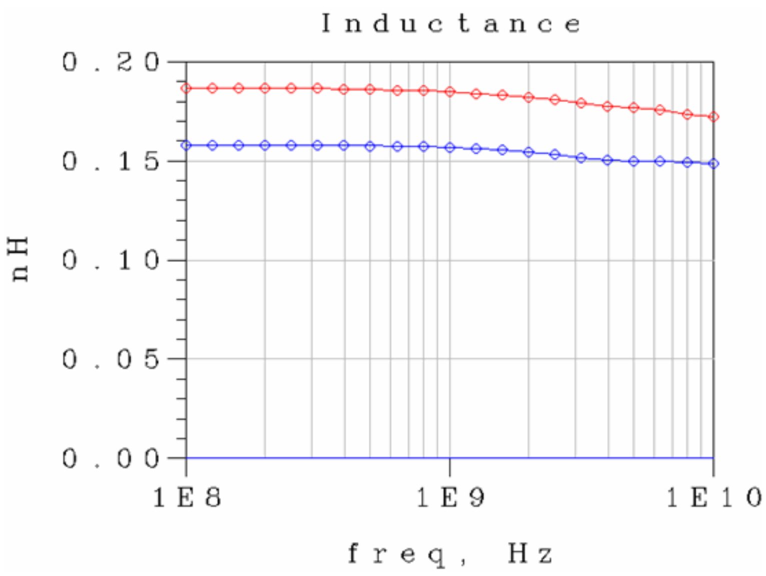


Figure 5: spiral.dds - inductance vs. freq of spiral inductor



Tuning Using Layout Components and Momentum

Location: $HPEESOF_DIR/examples/Momentum/emcktcosim/Coupled_Stubs_tune_wrk

Objective

This example illustrates using a EM Model with layout parameters to examine the behavior of a coupled line filter as a function of the length of the stub arms. The example is setup with a precalculated database of Momentum simulation results for a range of values of the stub length. This database will allow you to examine the effect of changing the length of the stubs on the S-parameters using the tuning feature.

Setup

The EM Model was generated from the layout called "CoupledStubs" in the workspace. The CoupledStubs design is setup for Momentum simulations and has the definition of the parameters L1 and L2 (the lengths of the two stubs.) This can be seen using the EM > Component > Parameters dialog. A Layout Component was created from this layout using the EM > Component > Create EM Model and Symbol dialog box.

The EM Model can be inserted into any schematic as a regular component. In this example, the database of Momentum models for this component was generated from the schematic design, "createDB." For the purpose of this example, the two stub lengths L1 and L2 will be varied simultaneously using the variable L. The design is setup with a parameter sweep to vary L from 100 to 130 mil in steps of 1.5 mil (note that the step size in the database is chosen to be smaller than the default interpolation step which can be viewed under EM Model Interpolation tab.) Simulating the "createDB" schematic generated the Layout Component database. The contents of the database can be inspected from the EM Model under database tab.

The next step is to examine the behavior of the filter while changing the stub length layout parameter using tuning. This is illustrated in the design called "tuneL." In this design, an S-parameter simulation has been setup with a corresponding data display window. To tune the stub length parameter (L) and view the results, open the "tuneL" design and open the "tuneL" data display. You can then tune the parameter between 100 and 130 mil while examining the effect on the S-parameters.

Figure 1: Coupled stub filter layout

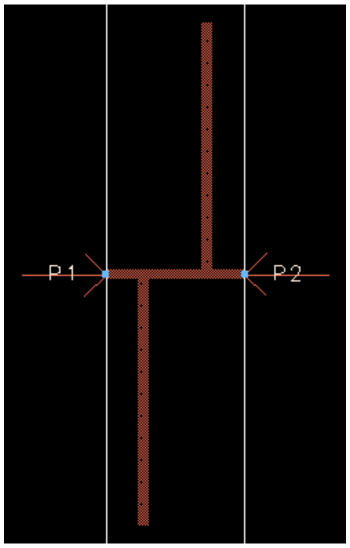
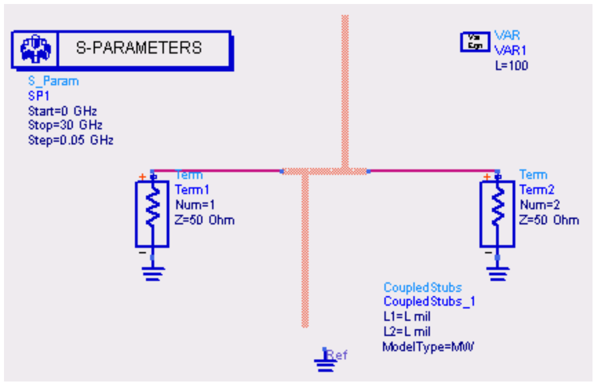
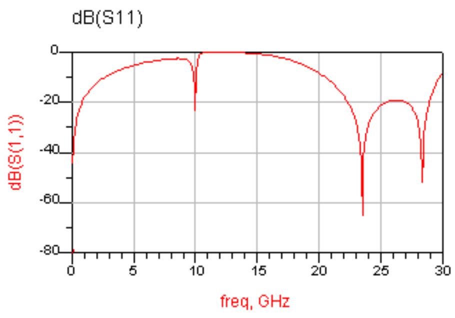


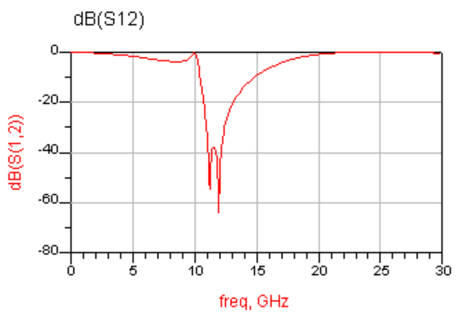
Figure 2: Layout Component in a schematic



Analysis

Figure 3: Filter frequency response for one stub length





X-Band Balun

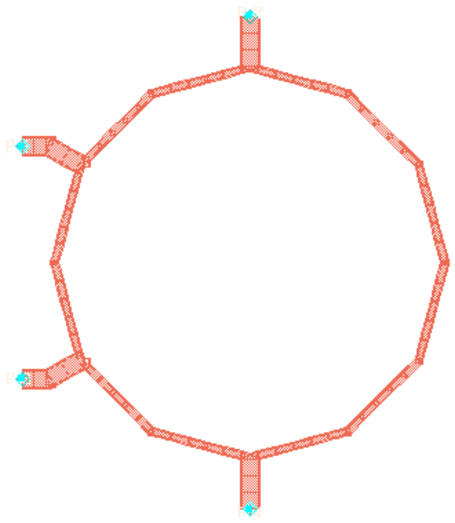
Location: $HPEESOF_DIR/examples/Momentum/Microwave/Rat_Race_wrk

Objective

This example illustrates a rat race balun designed for use in the X-band.

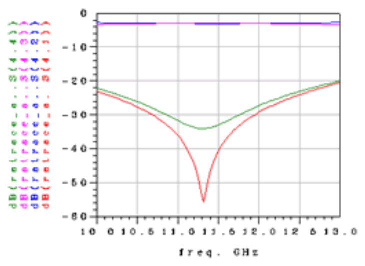
Setup

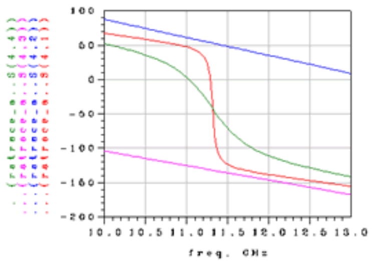
The balun is built on a sapphire substrate. The center frequency is 11.31GHz. If port 1 is excited, the power is equally split over port 2 and 3 in phase, port 4 is isolated from port 1. If port 4 is excited, the power is split equally to port 2 and 3, which are about 180 degrees out of phase. The structure is meshed with 20 cells per wavelength at 20GHz.



Analysis

Figure 1: S-Paramaters from simulation





MultiTech_Module Examples

This section lists the Multi-technology examples.

- *Front-end Transceiver Circuit Design using Multi-Technology* (examples)
- *MMIC Chip on a QFN package* (examples)

Front-end Transceiver Circuit Design using Multi-Technology

Location: $HPEESOF_DIR/examples/MultiTech_Module/Antenna_SPDT_LNA_PA_wrk

Objective

This example illustrates the Multi Technology Flow for Front-End Transceiver Circuit Design. This flow demonstrates the usage of two different technologies of PDKs defined in separate libraries along with ADS standard libraries to realize the complete design of transceiver circuit.

The transceiver circuit that has been designed consists of mainly four major components:

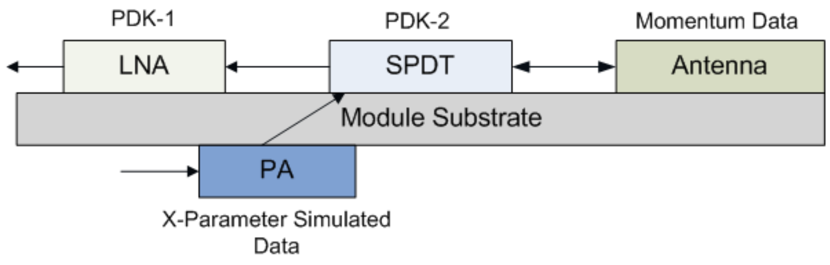
- Antenna
- SPDT
- LNA
- Power Amplifier

The LNA and SPDT are designed using two different PDKs having different technologies defined in two separate libraries whereas as Antenna and PA are designed using ADS standard libraries. The complete transceiver circuit is then created with and without using the Module substrate. The design is then simulated for S-Parameter.

The workspace includes four major designs based on the following technologies:

1. Antenna Design using Momentum simulated data
2. SPDT Design the model of MMIC Chip from the DemoKit_V3
3. LNA Design using the model of MMIC Chip from DemoKit_Non_Linear
4. Power Amplifier using X-Parameter simulated data

Setup



1. Antenna_Data: Antenna design using momentum simulated data.
2. SPDT_Design2: SPDT circuit design using DemoKit_V3 PDK. SPDT_SP: simulation on SPDT circuit
3. LNA_Design: LNA circuit design DemoKit_Non_Linear PDK. LNA_SP: simulation performed on LNA circuit
4. Power_Amplifier: Power Amplifier design using X-Parameter simulated data.
5. LNA_SPDT_Patch_PA1: Integrated Transceiver circuit design without using Module Substrate
6. LNA_SPDT_Patch_PA3: Integrated Transceiver circuit design using Module Substrate

Analysis

The below graphs illustrate the SPDT S-Parameter simulated data:

Figure 1: Insertion and Isolation Data from SPDT circuit simulation

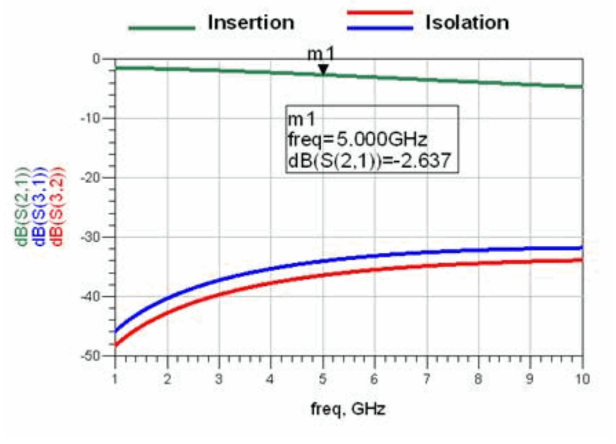


Figure 2: Gain, Loss, Stability and NF measurement on LNA Circuit

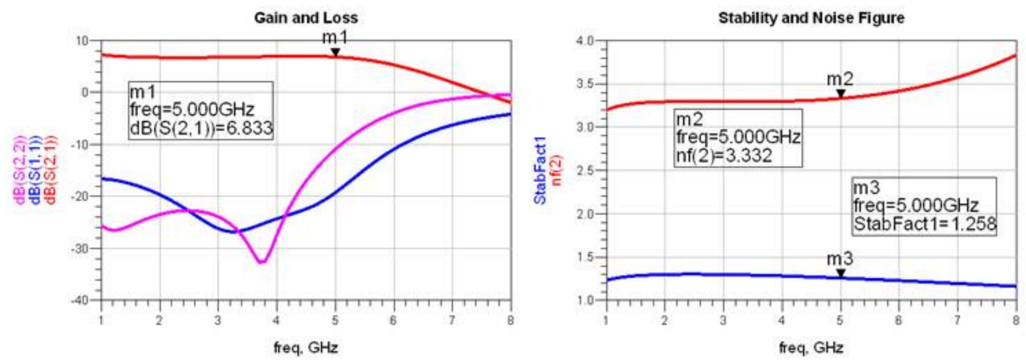


Figure 3: Complete Transceiver Circuit simulation without using Module Substrate

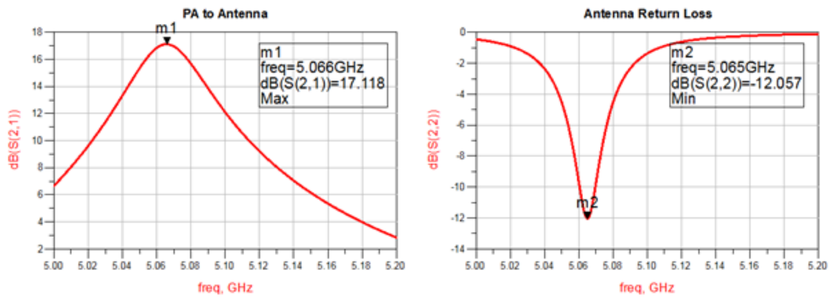
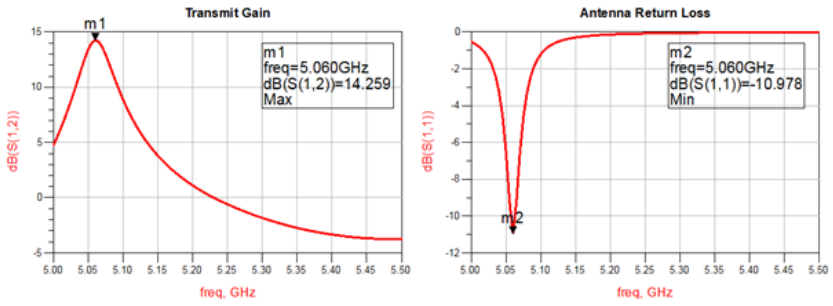


Figure 4: Complete Transceiver Circuit simulation using Module Substrate



Notes

For more details, see Multi-Technology for Circuit Simulation.

MMIC Chip on a QFN package

Location:
$HPEESOF_DIR/examples/MultiTech_Module/MTM_Flow_MMIC_Chip_QFN_Package_wrk

Objective

This examples illustrates the Multi Technology flow using a MMIC Low Noise Amplifier on a QFN package in the layout for EM simulation. The flow begins with the realization of a MMIC LNA implemented with ideal and lumped elements. Later these elements are replaced with PDK models in a layout. This MMIC LNA is simulated and compared with the ideal design. A QFN package is realized in layout using standard QFN packaging technology and simulation is carried out using FEM. However, to see the overall performance including parasitic coupling between the input and output along with the package, entire package and LNA is modeled using Multi-Technology, and FEM simulation is carried out. In this example, the LNA substrate is used as a nested substrate on the package substrate and the LNA is placed as a library component on the package in the layout design. The design is then simulated using the FEM solver.

The workspace includes two designs based on the following technologies:

1. Amplifier Design using the model of MMIC Chip from the Demo Kit
2. QFN Package Design in Layout

Setup

1. LNA_Chip_Design: Low Noise Amplifier design using Non-Linear Demo Kit
2. LNA_Nested_FEM: LNA design modified for use as a nested design in a QFN package for FEM simulation
3. QFN_Package: QFN package design in a layout
4. QFN_Nested_LNA_FEM: complete design of QFN package with LNA as nested technology on the package for FEM
5. QFN_Nested_LNA_wFET: complete design of QFN package with LNA as nested technology on the package for FEM Co-simulation along with FET

Analysis

The below graphs illustrate the FEM simulated data:

Figure 1: Surface Field plot over two planes

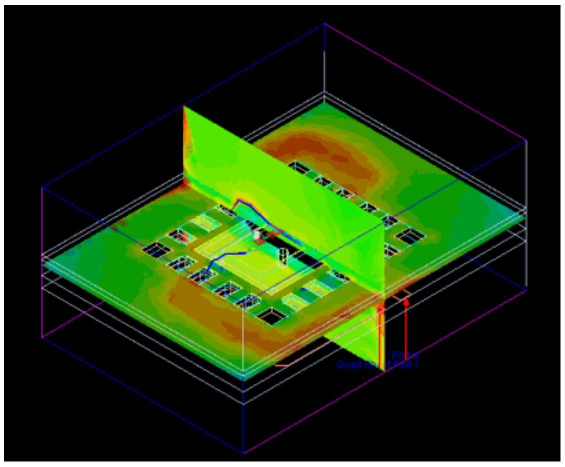


Figure 2: Boundaries and surface mesh, Volume Mesh

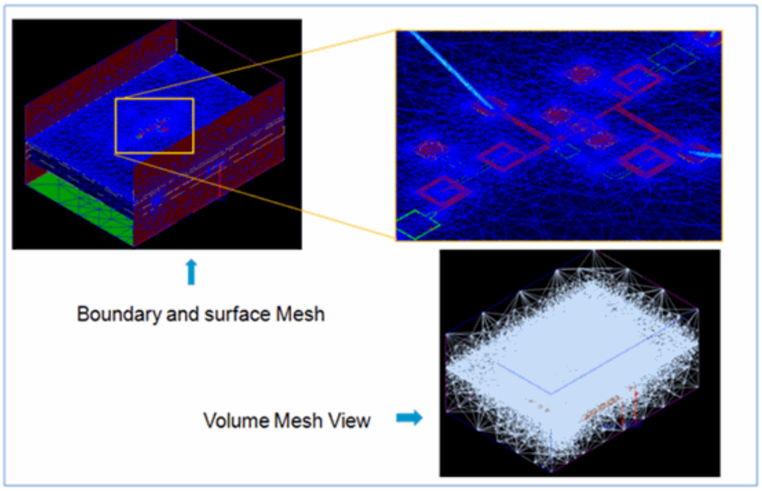


Figure 3: Simulation Data of LNA

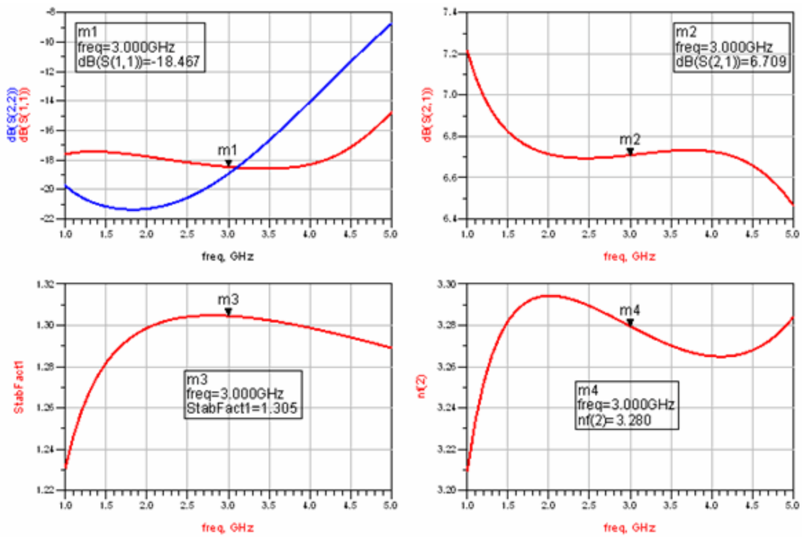
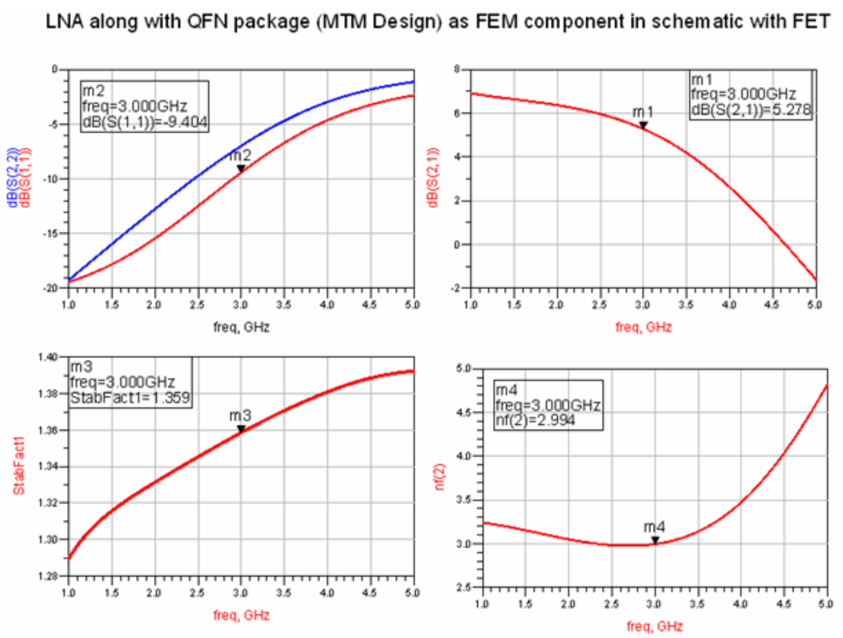


Figure 4: S-parameter simulation data



Notes

For more details, see Multi-Technology for EM Simulation.

Ptolemy Doc Examples

Examples of how the theoretical BER performance in an Additive White Gaussian Noise environment can be achieved for various modulation types.

- *BER Validation Guide* (examples)
- *Ptolemy Cosimulation with Simple SystemC Sink* (examples)
- *Ptolemy Cosimulation with SystemC FIR (RTL Implementation)* (examples)
- *Ptolemy Cosimulation with SystemC Sine Wave Generator* (examples)
- *Ptolemy CoSimulation with SystemC UpSample* (examples)
- *Ptolemy-SystemC-Transient Cosimulation* (examples)

BER Validation Guide

Location: /examples/PtolemyDocExamples/BER_Validation_wrk

Objective

This workspace demonstrates how the theoretical BER performance in an AWGN (Additive White Gaussian Noise) environment can be achieved for various modulation types.

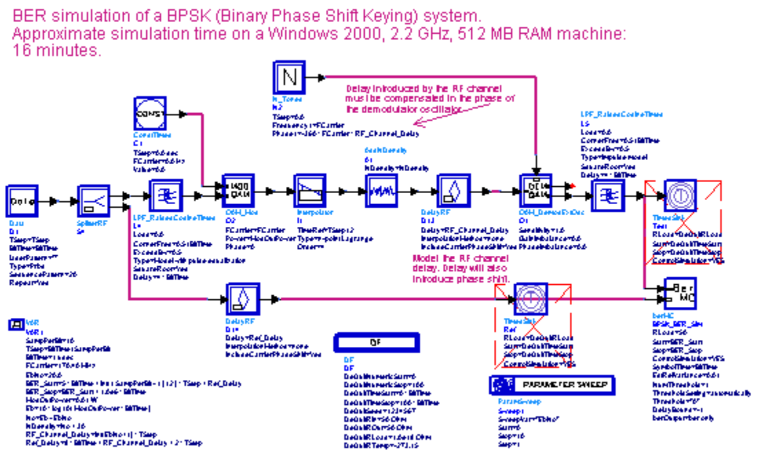
The approximate simulation times on a Windows 2000, 2.2 GHz, 512 MB RAM machine are as follows.

- BPSK 16 minutes
- DBPSK 12 minutes
- QPSK 21 minutes
- DQPSK 30 minutes
- Pi4DQPSK 30 minutes
- 16QAM 25 minutes
- 64QAM 20 minutes

Setup

BER-BPSK

Figure 1: BER simulation of a BPSK (Binary Phase Shift Keying) system.

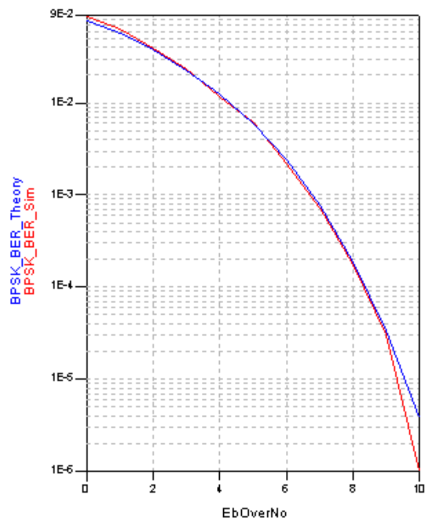

BER simulation of a BPSK (Binary Phase Shift Keying) system.
Approximate simulation time on a Windows 2000, 2.2 GHz, 512 MB RAM machine: 16 minutes.
Delay introduced by the RF channel must be compensated in the phase of the demodulator oscillator.
Model the RF channel delay. Delay will also introduce phase shift.
PARAMETER SWEEP

1. The input signal is a 1 Mbit/sec NRZ bit sequence generated by the Data source component. The bit sequence is a pseudo random binary sequence (Type parameter of Data source is set to Prbs) with a period of 220 bits (SequencePattern parameter of Data source is set to 20).
2. The input bit sequence is filtered with a root raised cosine filter. The bandwidth of the filter is set to half of the input signal bit rate (500 kHz). The filter model is using pulse equalization in order to compensate for the ISI (InterSymbol Interference) due to the finite pulse width of the input data.
3. To generate the BPSK signal the filtered bit sequence is sent to the I (in-phase) input of a QAM modulator (QAM_Mod component). The Q (quadrature phase) input of the modulator is connected to a constant output source with 0 Volts output. The Interpolator component is connected to the ouput of the QAM modulator in order to correct the signal sampling instant. For more details, see the Limitations of Root Raised Cosine Filter paragraph in the Notes section.
4. The modulated signal is sent through an RF channel modeled by the AddNDensity

and DelayRF components. The RF channel noise power is swept so that the Eb/No of the system is swept from 0 to 10 dB. For more details, see the Setting the Noise Power and the RF Channel Delay paragraphs in the Notes section.

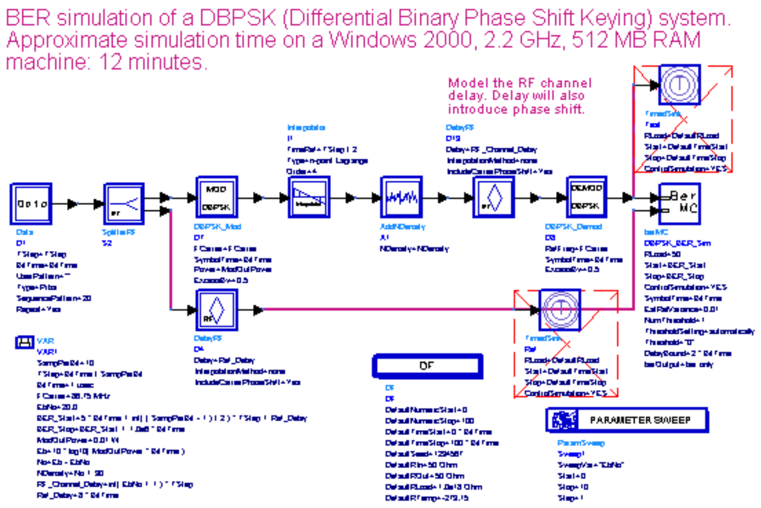
5. BPSK modulation requires coherent (or synchronous) demodulation. A QAM demodulator (QAM_DemodExtOsc component) is used with a local oscillator signal generated by an N_Tones source as explained in the Coherent and Non-coherent Demodulation paragraph in the Notes section. Only the I signal at the output of the QAM demodulator is used.
6. The I signal at the output of the QAM demodulator is filtered using a root raised cosine filter. This filter has the same characteristics as the root raised cosine filter on the transmitter side except that it does not include pulse equalization.
7. The signal at the output of the receiver root raised cosine filter is connected to the test input of the berMC sink. The input bit sequence generated by the Data source is the reference signal. This design synchronizes the test and reference signals manually by first determining the delay between them. The delay between the test and reference signals is 8 x SymbolTime (due to the root raised cosine filters in the transmitter and receiver) + RF_Channel_Delay (due to the RF channel) + 2 x TStep (due to the Interpolator). Therefore, the output of the Data source is delayed by this amount and connected to the reference input of the berMC sink.
8. The berMC sink is set up as explained in the BER Sink Setup paragraph of the Notes section. With this setup, the estimation relative variance achieved for Eb/No between 0 dB and 8 dB is the desired 0.01. For Eb/No=9 dB the estimation relative variance is approximately 0.037. Although this is not too close to the desired 0.01, it is still quite low and so the BER result is expected to be accurate. For Eb/No=10 dB the estimation relative variance is approximately 0.33. This variance is quite high and so the BER result for Eb/No=10 dB is expected to be less accurate. To achieve an estimation relative variance of 0.01 at Eb/No of 9 dB and 10 dB, the Stop parameter of the berMC sink needs to be set to a higher value, e.g. BER_Start + 1.0e7 * BitTime. Simulation time will increase accordingly.
9. The simulation results are shown below. As can be seen, the simulation results agree very well with theory except for Eb/No=10 dB. At Eb/No=10 dB, the deviation betweeen the simulation and the theoretical results is larger compared to lower Eb/No values. This is because of the higher estimation relative variance achieved at Eb/No=10 dB.

Figure 2: BPSK BER simulation results.



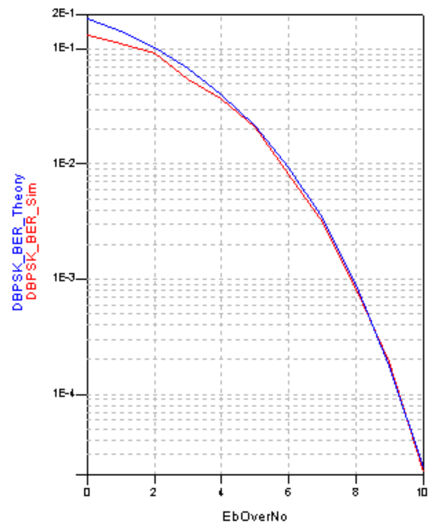
BER - DBPSK

Figure 3: BER simulation of a DBPSK (Differential Binary Phase Shift Keying) system.


BER simulation of a DBPSK (Differential Binary Phase Shift Keying) system.
Approximate simulation time on a Windows 2000, 2.2 GHz, 512 MB RAM machine: 12 minutes.
Model the RF channel delay. Delay will also introduce phase shift.
MOD DBPSK
DEMOD DBPSK
Ber MC
DF
PARAMETER SWEEP
VAR

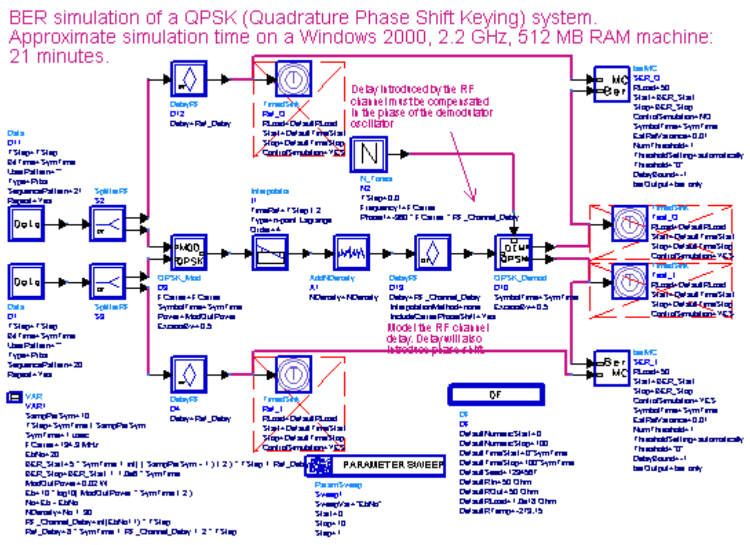
1. The input signal is a 1 Mbit/sec NRZ bit sequence generated by the Data source component. The bit sequence is a pseudo random binary sequence (Type parameter of Data Source is set to Prbs) with a period of 220 bits (SequencePattern parameter of Data source is set to 20).
2. The input bit sequence is sent to a DBPSK modulator (DBPSK_Mod component), where it is first differentially encoded, then filtered with a root raised cosine filter, and finally sent to the I (in-phase) input of a QAM modulator. The root raised cosine filter model is using pulse equalization in order to compensate for the ISI (InterSymbol Interference) due to the finite pulse width of the input data. The Interpolator component is connected to the ouput of the DBPSK modulator in order to correct the signal sampling instant. For more details, see the Limitations of Root Raised Cosine Filter paragraph in the Notes section.
3. The modulated signal is sent through an RF channel modeled by the AddNDensity and DelayRF components. The RF channel noise power is swept so that the Eb/No of the system is swept from 0 to 10 dB. For more details, see the Setting the Noise Power and the RF Channel Delay paragraphs in the Notes section.
4. DBPSK modulation does not require coherent demodulation. The DBPSK_Demod component implements a non-coherent DBPSK demodulator. DBPSK_Demod includes filtering with a root raised cosine filter and differential decoding.
5. The signal at the output of the demodulator is connected to the test input of the berMC sink. The input bit sequence generated by the Data source is the reference signal. This design uses the auto synchronization feature of the berMC sink and assumes the delay introduced by the RF channel is not known but has an upper bound of 2 x BitTime. The Interpolator component introduces a delay of two simulation time steps, which will also be considered as part of the unknown RF channel delay. The delay introduced by the transmitter and receiver filters is known to be 8 x BitTime. The output of the Data source is delayed by the known amount of delay (8 x BitTime) and connected to the reference input of the berMC sink. The DelayBound parameter of the berMC sink is set to the upper bound of the unknown part of the delay (2 x BitTime).
6. The berMC sink is set up as explained in the BER Sink Setup paragraph of the Notes section. With this setup, the estimation relative variance achieved for Eb/No between 0 dB and 9 dB is the desired 0.01. For Eb/No=10 dB the estimation relative variance is approximately 0.047. To achieve an estimation relative variance of 0.01 at Eb/No of 9 dB, the Stop parameter of the berMC sink needs to be set to a higher value, e.g. BER_Start + 1.0e7 * BitTime. Simulation time will increase accordingly.
7. The simulation results are shown below. As can be seen, the simulation results agree very well with theory. Although the desired estimation relative variance of 0.01 was not achieved at Eb/No=10 dB, 0.047 is still a pretty low variance and so the simulation and the theoretical results agree very well even for Eb/No=10 dB.

Figure 4: DBPSK BER simulation results.



BER - QPSK

Figure 5: BER simulation of a QPSK (Quadrature Phase Shift Keying) system.

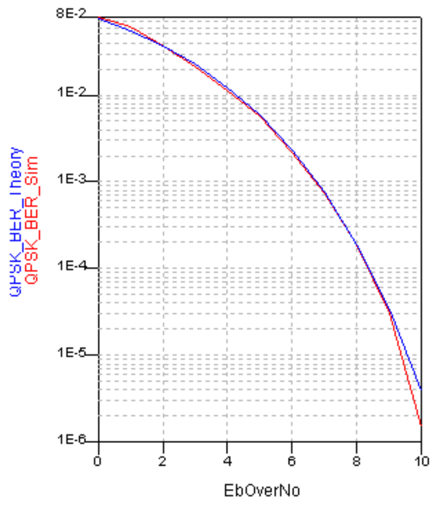

BER simulation of a QPSK (Quadrature Phase Shift Keying) system.
Approximate simulation time on a Windows 2000, 2.2 GHz, 512 MB RAM machine:
21 minutes.
Delay introduced by the RF channel must be compensated in the phase of the demodulator oscillator
Model the RF channel delay. Delay will also introduce phase shift.
PARAMETER SWEEP
DF

1. A QPSK system needs two input bit sequences: one for the I (in-phase) channel and one for the Q (quadrature phase) channel. The two 1 Mbit/sec input bit sequences are generated using Data source components. The bit sequences are pseudo random binary sequences (Type parameter of Data sources is set to Prbs). To avoid generating the same bit sequence for both the I and Q channels the SequencePattern parameter of the two Data sources is set to different values (20 for the I channel and 21 for the Q channel).
2. The two input bit sequences are connected to the I and Q inputs of a QPSK modulator (QPSK_Mod component), where they are first filtered with root raised cosine filters and then used to modulate the in-phase and quadrature phase carriers of a QAM modulator. The root raised cosine filter models are using pulse equalization in order to compensate for the ISI (InterSymbol Interference) due to the finite pulse width of the input data. The Interpolator component is connected to the ouput of the QPSK modulator in order to correct the signal sampling instant. For more details, see the Limitations of Root Raised Cosine Filter paragraph in the Notes section.
3. The modulated signal is sent through an RF channel modeled by the AddNDensity and DelayRF components. The RF channel noise power is swept so that the Eb/No of the system is swept from 0 to 10 dB. For more details, see the Setting the Noise Power and the RF Channel Delay paragraphs in the Notes section.
4. QPSK modulation requires coherent (or synchronous) demodulation. A QPSK demodulator (QPSK_Demod component) is used with a local oscillator signal generated by an N_Tones source as explained in the Coherent and Non-coherent Demodulation paragraph in the Notes section.
5. The QPSK_Demod component demodulates the received signal and then filters the I

and Q channel signals with root raised cosine filters. These root raised cosine filters have the same characteristics as the root raised cosine filters on the transmitter side except that they do not include pulse equalization.

6. The filtered I and Q signals at the output of QPSK_Demod are connected to the test inputs of two berMC sinks. The two input bit sequences generated by the Data sources are the two reference signals. This design synchronizes the test and reference signals manually by first determining the delay between them. The delay between the test and reference signals is 8 x SymbolTime (due to the root raised cosine filters in the transmitter and receiver) + RF_Channel_Delay (due to the RF channel) + 2 x TStep (due to the Interpolator). Therefore, the outputs of the Data sources are delayed by this amount and connected to the reference inputs of the two berMC sinks.
7. The berMC sinks are set up as explained in the BER Sink Setup paragraph of the Notes section. With this setup, the estimation relative variance achieved by SER_Q (sink that does not control the simulation) for Eb/No between 0 dB and 8 dB is very close to the desired 0.01. SER_I, the sink that controls the simulation, achieves an estimation relative variance of 0.01 for these Eb/No values. For Eb/No=9 dB, SER_I and SER_Q achieve an estimation relative variance of 0.033 and 0.032 respectively. Although these variances are not close to the desired variance of 0.01, they are are still low and so the BER result is expected to be quite accurate. For Eb/No=10 dB, SER_I and SER_Q achieve an estimation relative variance of 0.50 and 1.0 respectively. These variances are quite high and the BER result at Eb/No=10 dB is expected to be less accurate. To achieve a smaller estimation relative variance at Eb/No of 9 dB and 10 dB, the Stop parameter of the berMC sinks needs to be set to a higher value, e.g. BER_Start + 1.0e7 * SymbolTime. Simulation time will increase accordingly.
8. The simulation results are shown below. As can be seen, the simulation results agree very well with theory except for Eb/No=10 dB. At Eb/No=10 dB, the deviation betweeen the simulation and the theoretical results is larger compared to lower Eb/No values. This is because of the higher estimation relative variance achieved at Eb/No=10 dB.

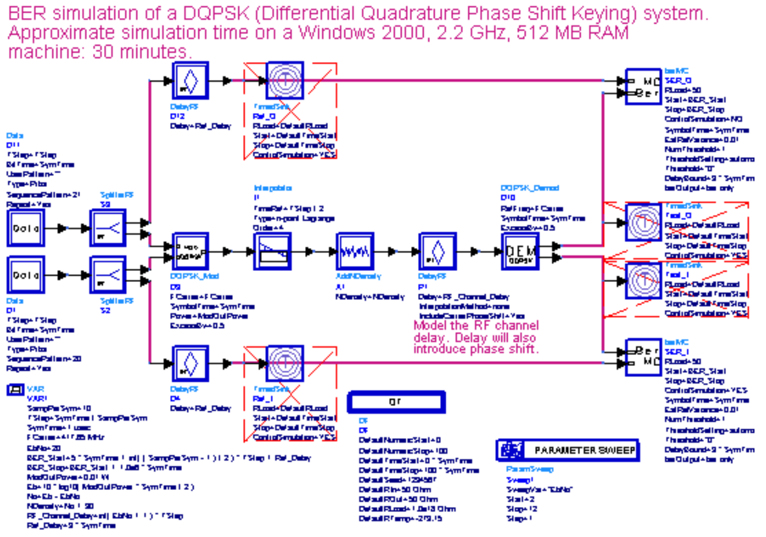
Figure 6: QPSK BER simulation results.



BER_DQPSK and BER_Pi4DQPSK

The BER_DQPSK design demonstrates the theoretical BER performance of a DQPSK (Differential Quadrature Phase Shift Keying). A nearly identical design demonstrates the theoretical BER performance using a pi/4-DQPSK (pi/4 Differential Quadrature Phase Shift Keying) system. The two setups are the same except that the BER_Pi4DQPSK design uses pi/4 DQPSK (DQPSK_Pi4Mod and DQPSK_Pi4Demod) components instead of the DQPSK (DQPSK_Mod and DQPSK_Demod) components.

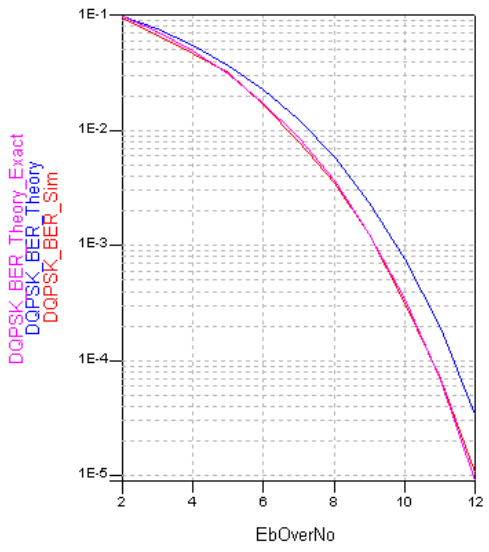
Figure 7: BER simulation of a DQPSK (Differential Quadrature Phase Shift Keying) system.



1. A DQPSK system needs two input bit sequences: one for the I (in-phase) channel and one for the Q (quadrature phase) channel. The two 1 Mbit/sec input bit sequences are generated using Data source components. The bit sequences are pseudo random binary sequences (Type parameter of Data sources is set to Prbs). To avoid generating the same bit sequence for both the I and Q channels the SequencePattern parameter of the two Data sources is set to different values (20 for the I channel and 21 for the Q channel).
2. The two input bit sequences are connected to the I and Q inputs of a DQPSK modulator (DQPSK_Mod component), where they are first differentially encoded, then filtered with root raised cosine filters and finally used to modulate the in-phase and quadrature phase carriers of a QAM modulator. The root raised cosine filter models are using pulse equalization in order to compensate for the ISI (InterSymbol Interference) due to the finite pulse width of the input data. The Interpolator component is connected to the ouput of the DQPSK modulator in order to correct the signal sampling instant. For more details, see the Limitations of Root Raised Cosine Filter paragraph in the Notes section.
3. The modulated signal is sent through an RF channel modeled by the AddNDensity and DelayRF components. The RF channel noise power is swept so that the Eb/No of the system is swept from 2 to 12 dB. For more details, see the Setting the Noise Power and the RF Channel Delay paragraphs in the Notes section.
4. DQPSK modulation does not require coherent demodulation. The DQPSK_Demod component implements a non-coherent DQPSK demodulator. DQPSK_Demod includes filtering with a root raised cosine filter and differential decoding.
5. The I and Q signals at the output of DQPSK_Demod are connected to the test inputs of two berMC sinks. The two input bit sequences generated by the Data sources are the two reference signals. This design uses the auto synchronization feature of the berMC sink and assumes the delay introduced by the RF channel is not known but has an upper bound of 3 x SymbolTime. The Interpolator component introduces a delay of two simulation time steps, which will also be considered as part of the unknown RF channel delay. The delay introduced by the transmitter and receiver filters is known to be 8 x SymbolTime. Differential encoding also introduces a delay of SymbolTime so the Data source outputs are delayed by 9 x SymbolTime and connected to the reference inputs of the berMC sinks. The DelayBound parameter of the berMC sinks is set to the upper bound of the unknown part of the delay (3 x SymbolTime).
6. The berMC sinks are set up as explained in the BER Sink Setup paragraph of the Notes section. With this setup, the estimation relative variance achieved by SER_Q (sink that does not control the simulation) for Eb/No between 2 dB and 10 dB is very close to the desired 0.01. SER_I, the sink that controls the simulation, achieves an estimation relative variance of 0.01 for these Eb/No values. For Eb/No=11 dB, SER_I and SER_Q achieve an estimation relative variance of 0.0147 and 0.0142 respectively. These variances are very close to 0.01 and so the BER result is expected to be quite accurate. For Eb/No=12 dB, SER_I and SER_Q achieve an estimation relative variance of 0.083 and 0.11 respectively. These variances are not close to the desired variance of 0.01 and so the BER result at Eb/No=12 dB may not be as accurate as for lower Eb/No values. To achieve a smaller estimation relative variance at Eb/No of 11 dB and 12 dB, the Stop parameter of the berMC sinks needs to be set to a higher value, e.g. BER_Start + 1.0e7 * SymbolTime. Simulation time will increase accordingly.
7. The simulation results are shown below. As can be seen, the simulation results agree

very well with theory even for Eb/No=12 dB (where a higher estimation relative variance was achieved; see the BER Sink Setup paragraph in the Notes section for why this can happen). The results in the plot below show two theoretical BER curves: an exact one (DQPSK_BER_Theory_Exact) and an approximation one (DQPSK_BER_Theory). Read the notes in the corresponding data display for more details on how these two theoretical BER curves were generated.

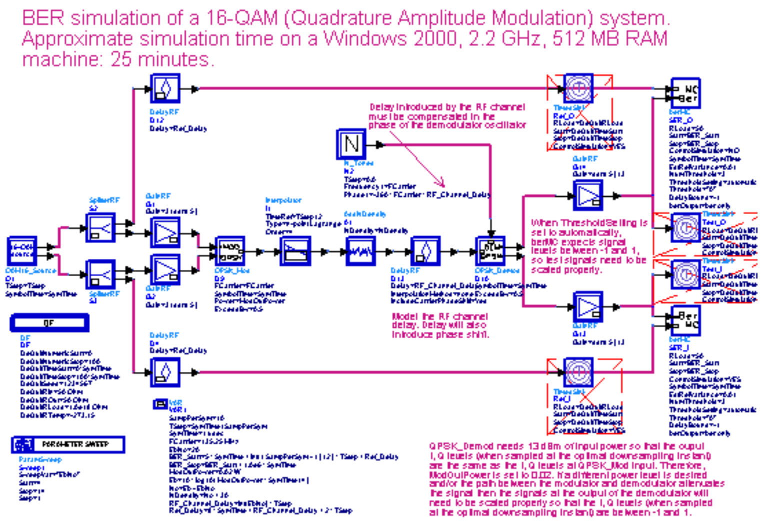
Figure 8: DQPSK BER simulation results.



BER_16QAM and BER_64QAM

The BER_16QAM design demonstrates the theoretical BER performance of a 16-QAM (Quadrature Amplitude Modulation) system. A nearly identical design demonstrates the theoretical BER performance using a 64-QAM system. The two setups are the same except that the BER_16QAM design uses the QAM16_Source component while the BER_64QAM design uses the QAM64_Source component. Scaling factors applied to the signals at different points in the design are also different between the 16-QAM and 64-QAM setups. These differences are discussed at the end of this section.

Figure 9: BER simulation of a 16-QAM (Quadrature Amplitude Modulation) system.


BER simulation of a 16-QAM (Quadrature Amplitude Modulation) system.
Approximate simulation time on a Windows 2000, 2.2 GHz, 512 MB RAM machine: 25 minutes.
Delay introduced by the RF channel must be compensated in the phase of the demodulator oscillator
When ThresholdSetting is set to automatically, berMC expects signal levels between -1 and 1, so test signals need to be scaled properly.
Model the RF channel delay. Delay will also introduce phase shift.
QPSK_Demod needs 13 dBm of input power so that the output I, Q levels (when sampled at the optimal downsampling instant) are the same as the I, Q levels at QPSK_Mod input. Therefore, ModOutPower is set to 0.02. If a different power level is desired and/or the path between the modulator and demodulator attenuates the signal then the signals at the output of the demodulator will need to be scaled properly so that the I, Q levels (when sampled at the optimal downsampling instant) are between -1 and 1.
PARAMETER SWEEP
DF

1. 16-QAM modulation uses two 4-level baseband signals to modulate the in-phase and quadrature phase carriers of a QAM modulator. The two 4-level baseband signals are generated by the QAM16_Source component. This source is a hierarchical design located in the example workspace. It uses two Bits sources to generate I and Q bit sequences. The bits in each of the I and Q bit sequences are grouped in pairs and mapped to four different signal levels (-1, -1/3, 1/3, 1) using a lookup table. The lookup table uses Gray encoding. The signal levels in the I and Q channels are then

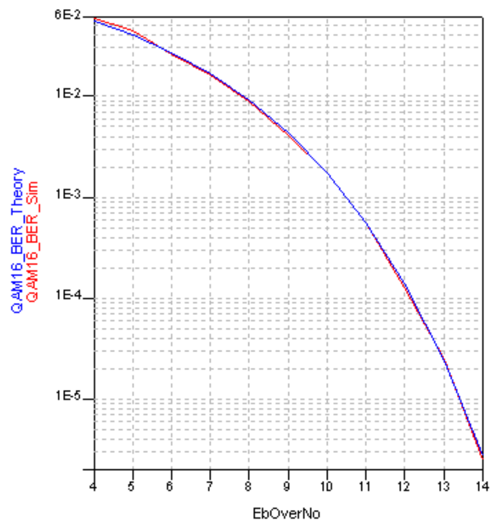
sampled and held (repeated) so that they remain constant during the symbol period. The RMS value of the signal at the output of the source (considered as a complex signal with the I channel being the real part and the Q channel being the imaginary part) is equal to sqrt(10)/3.

2. To generate the 16-QAM signal, the I and Q signals at the output of the QAM16_Source need to be filtered and applied to the I and Q inputs of a QAM modulator. The QPSK_Mod component, which includes root raised cosine filters and a QAM modulator, can be used to generate other multi-level QAM signals, as long as the IQ signal at its input has an RMS value of sqrt(2) (otherwise the power at its output will not be correct). To achieve this, the I and Q signals at the output of the QAM16_Source are scaled by 3/sqrt(5). The root raised cosine filter models are using pulse equalization in order to compensate for the ISI (InterSymbol Interference) due to the finite pulse width of the input data. The Interpolator component is connected to the ouput of the QPSK modulator in order to correct the signal sampling instant. For more details, see the Limitations of Root Raised Cosine Filter paragraph in the Notes section.
3. The modulated signal is sent through an RF channel modeled by the AddNDensity and DelayRF components. The RF channel noise power is swept so that the Eb/No of the system is swept from 4 to 14 dB. For more details, see the Setting the Noise Power and the RF Channel Delay paragraphs in the Notes section.
4. 16-QAM modulation requires coherent (or synchronous) demodulation. A QPSK demodulator (QPSK_Demod component) is used with a local oscillator signal generated by an N_Tones source as explained in the Coherent and Non-coherent Demodulation paragraph in the Notes section. The QPSK_Demod component demodulates the received signal and then filters the I and Q channel signals with root raised cosine filters. These root raised cosine filters have the same characteristics as the root raised cosine filters on the transmitter side except that they do not include pulse equalization.
5. To measure BER of signals with more than two levels the BER sink uses more than one error detection threshold and so the exact level of the test and reference signals is important (see the Scaling of Test and Reference Signals sub-section in the Setting up BER Simulations section in the Sinks > PE Estimator Usage documentation). The reference I and Q signals in this example are the signals at the ouput of the QAM16_Source. These signals were generated so that they can be used with the threshold settings the BER sink uses when the thresholds are set automatically (ThresholdSetting parameter of the BER sink set to automatically). The level of the I and Q signals at the ouput of the QPSK_Demod component can be determined in two ways:
    1. By plotting the eye diagrams one can determine the signal levels at the optimal sampling instant. In this case, it is recommended that noise and other impairments in the system are turned off, so that the eye diagram is as clean as possible and the signal levels are determined more accurately.
    2. By tracking the signal amplitude changes through the system.
6. In this design, it is very easy to track the signal amplitude changes through the system and come up with the exact scale factor needed to scale the signals at the ouput of the QPSK_Demod component. Starting at the ouput of the QAM16_Source, the signal is first scaled by 3/sqrt(5). The QPSK_Mod component also affects the signal amplitude depending on the value of the Power parameter. The Power parameter of QPSK_Mod is set to 0.02 W = 13 dBm. This is based on the fact that when the signal power at the input of QPSK_Demod is 13 dBm, its output I and Q levels (when sampled at the optimal sampling instant) are the same as the I and Q levels at the input of QPSK_Mod. Since the path between the modulator and demodulator does not attenuate/amplify the signal, the power at the input of QPSK_Demod is the same as the power at the output of QPSK_Mod, which is equal to 13 dBm. Therefore, the QPSK_Demod output I and Q levels (when sampled at the optimal sampling instant) are the same as the I and Q levels at the input of QPSK_Mod and so in order to make the test signals have the same level as the reference signals, the I and Q signals at the ouput of QPSK_Demod need to be scaled by sqrt(5)/3. The demodulator output signals will need to be scaled by a different scale factor if the power at the input of QPSK_Demod is not 13 dBm.
7. The scaled demodulator output I and Q signals are connected to the test inputs of two berMC sinks. The I and Q signals at the output of the QAM16_Source are the two reference signals. This design synchronizes the test and reference signals manually by first determining the delay between them. The delay between the test and reference signals is 8 x SymbolTime (due to the root raised cosine filters in the transmitter and receiver) + RF_Channel_Delay (due to the RF channel) + 2 x TStep (due to the Interpolator). Therefore, the I and Q outputs of the QAM16_Source are delayed by this amount and connected to the reference inputs of the two berMC sinks.
8. The berMC sinks are set up as explained in the BER Sink Setup paragraph of the Notes section. With this setup, the estimation relative variance achieved by SER_Q (sink that does not control the simulation) for Eb/No between 4 dB and 12 dB is very

close to the desired 0.01. SER_I, the sink that controls the simulation, achieves an estimation relative variance of 0.01 for these Eb/No values. For Eb/No=13 dB, SER_I and SER_Q achieve an estimation relative variance of 0.018 and 0.022 respectively. Although these variances are not close to the desired variance of 0.01, they are are still low and so the BER result is expected to be quite accurate. For Eb/No=14 dB, SER_I and SER_Q achieve an estimation relative variance of 0.14 and 0.33 respectively. These variances are quite high and the BER result at Eb/No=14 dB is expected to be less accurate. To achieve a smaller estimation relative variance at Eb/No of 13 dB and 14 dB, the Stop parameter of the berMC sinks needs to be set to a higher value, e.g. BER_Start + 1.0e7 * SymbolTime. Simulation time will increase accordingly.

9. The simulation results are shown below. As can be seen, the simulation results agree very well with theory even for Eb/No=14 dB (where a higher estimation relative variance was achieved; see the BER Sink Setup paragraph in the Notes section for why this can happen).

Figure 10: 16-QAM BER simulation results.



The differences between the 16-QAM and 64-QAM setups are discussed below.

- The QAM64_Source (used in the BER_64QAM design) is very similar to the QAM16_Source (used in the BER_16QAM design) but it generates two 8-level signals at its outputs. The 8 signal levels are: -1, -5/7, -3/7, -1/7, 1/7, 3/7, 5/7, and 1. The RMS value of the signal at the output of the source (considered as a complex signal with the I channel being the real part and the Q channel being the imaginary part) is equal to sqr(42)/7. Therefore, the correct scaling factors before the QPSK_Mod and after the QPSK_Demod components are 7/sqrt(21) and sqrt(21)/7 respectively.

- For the BER_64QAM design the Eb/No of the system is swept from 6 to 18 dB.

- The NumThreshold parameter of the berMC sinks is set to 7 (test and reference signals have 8 levels).

Notes/Analysis

Limitations of Root Raised Cosine Filter

The typical input signal for the designs in this workspace is a multi-level waveform, whose level is kept constant during a symbol (or bit) period. When a signal like this has an even number of samples per symbol (or bit) and is filtered using a root raised cosine filter with pulse equalization, the signal at the output of the filter (and eventually at the output of the matched root raised cosine filter used in the receiver) does not have a sample at the optimal sampling instant. This is corrected by using an Interpolator component at the ouput of the filter or modulator.
For details, see the Choosing the Optimal Sampling Instant sub-section in the Setting up BER Simulations section in the Sinks > PE Estimator Usage documentation.

Setting the Noise Power

White Gaussian noise is generated and added to the signal using the AddNDensity component. The power spectral density No of the noise added by AddNDensity is NDensity dBm/Hz, where NDensity is the value of the NDensity parameter. Therefore, in order to set Eb/No to a desired value NDensity must be set to 10*log10(P*BitTime)-(Eb/No)+30, where P is the modulator output power in watts and (Eb/No) is in dB.

RF Channel Delay

The Delay RF component is used to model the RF channel delay between the modulator and demodulator. The delay introduced varies with the swept value of Eb/No to demonstrate that the provided setup can work for arbitrary delays. In addition to delaying the signal, the RF channel delay will also result in a carrier phase shift. For differential modulation types (DBPSK, DQPSK) this helps demonstrate the immunity of differential modulation to carrier phase shifts.

Coherent and Non-coherent Demodulation

Some modulation types (such as BPSK, QPSK, 16-QAM, and 64-QAM) require coherent or synchronous demodulation, which means that the frequency and phase of the local oscillator in the receiver need to be synchronized with the frequency and phase of the carrier at the demodulator input. The local oscillator signal can be generated using an N_Tones source. The frequency of N_Tones (Frequency1 parameter) is set to the same value as the FCarrier parameter of the modulator. To synchronize the phase of the signal at the ouput of N_Tones to the phase of the carrier at the demodulator input, the Phase1 parameter of N_Tones must be set to (-360 x FCarrier x Delay), where Delay is the delay between the modulator and demodulator. For the example designs in this workspace, the delay between the modulator and demodulator is only due to the RF channel and is equal to RF_Channel_Delay.

The Interpolator component also introduces a delay of two simulation time steps. However, this delay is only applied to the RF signal envelope and does not result in a carrier phase shift. For this reason, it must not be included in the delay between the modulator and demodulator for the purpose of adjusting the receiver local oscillator phase.

Systems that use differential modulation (such as DBPSK, DQPSK) do not require coherent demodulation because the information they carry is encoded in the carrier phase changes. For these systems a local oscillator is not needed in the receiver to demodulate the signal.

BER Sink Setup

- For systems with two channels (I and Q) two BER sinks are needed to measure the BER performance of the system. In addition, the SER (Symbol Error Rate) performance is also of interest. The BER and SER performance of the overall system can be derived by combining the results recorded in the two sinks. To combine the results of the two sinks in an unbiased way, it is important that the sinks process the same amount of data. This can be done by setting one of the sinks to control the simulation (ControlSimulation parameter set to YES) and the other one to not control the simulation. When using two BER sinks in this configuration, it is not guaranteed that the sink that is not controlling the simulation will achieve the desired estimation relative variance before the sink that controls the simulation forces the simulation to finish. However, the estimation relative variance achieved by both sinks should be very similar.
- The BER measurement is set to achieve an estimation relative variance of 0.01 (EstRelVariance parameter of berMC sinks). However, the upper limit on the number of symbols (or bits) to be simulated is set to 1 million (Stop parameter is set to BER_Start + 1.0e6 * SymbolTime, where BER_Start is the value of the Start parameter).
- A smaller estimation relative variance means higher probability (higher confidence) that the result is accurate. Not all the examples in this workspace achieve the 0.01 variance for all values of Eb/No. However, in some cases, even though the variance is much higher, the simulation results still seem to agree pretty well with the theoretical ones. This can happen because the noise signal is generated using random number generation algorithms. If the simulation random number generator seed (DefaultSeed parameter in the DF controller) is changed, a different noise signal will be generated and the design that achieved a variance of 0.1 will give BER results that vary a lot more compared to a design that achieved a variance of 0.01. So higher variance means less repeatable results (with different DefaultSeed values), which in turn means lower confidence that the result of a specific simulation run is accurate.
- The values for some of the other parameters of the berMC sink are as follows:
    - **Start:** based upon the recommendations in the Choosing the Optimal Sampling Instant sub-section of the Setting up BER Simulations section in the Sinks > PE Estimator Usage documentation, Start is set to

5 * BitTime + int((SampPerBit - 1) / 2) * TStep + Ref_Delay
==or==
5 * SymbolTime + int((SampPerSym-1)/2) * Tstep + Ref_Delay.

- **NumThreshold:** is set to N-1, where N is the number of levels of the input signal.
- **DelayBound:** is set to -1 if the test and reference signals were synchronized manually.

Ptolemy Cosimulation with Simple SystemC Sink

Location: $HPEESOF_DIR /examples /SystemC_Cosim /SystemC_Cosim_wrk /SystemC_Cosim_lib /SimpleSink

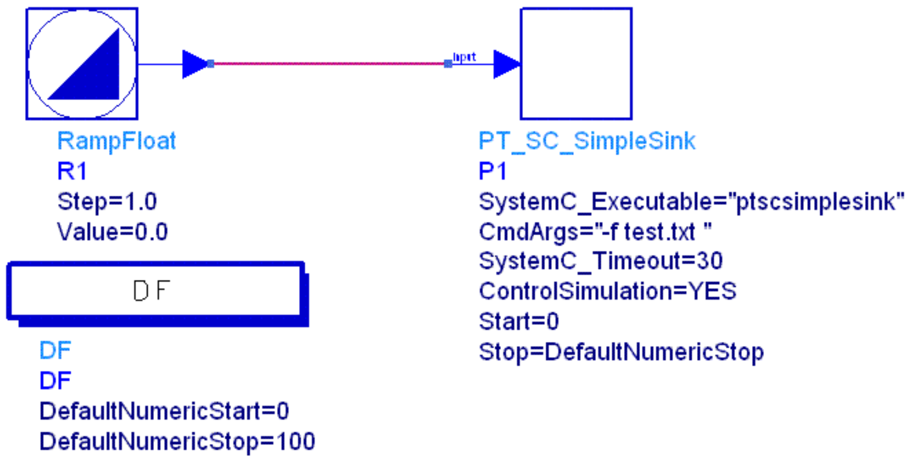
Objective

This example shows a SystemC sink model with Ptolemy and SystemC cosimulation ability to pass command line arguments.

Setup

The data from the input ramp signal is written to a file by the SystemC sink model. SystemC sink model accepts the command line argument "-f filename" to write the input data to a file under the data sub-directory of the workspace. The command line argument is specified in the ADS schematic and passed to SystemC sink models automatically.

See the source_code/systemc_files/PT_SC_SimpleSink sub-directory of this workspace to examine the SystemC code for this example and source_code/pl_files/SDFPT_SC_SimpleSink.pl or the corresponding Ptolemy model.



Analysis

The SystemC sink will write the data to

/examples/SystemC_Cosim/SystemC_Cosim_wrk/data/test.txt

Ptolemy Cosimulation with SystemC FIR (RTL Implementation)

Location: $HPEESOF_DIR /examples /SystemC_Cosim /SystemC_Cosim_wrk /SystemC_Cosim_lib /SystemC_FIR

Objective

This example:

1. Illustrates the ability of Ptolemy-SystemC cosimulation to pass array parameters.
2. Illustrates that ports other than sc_fifo can be used in internal design as long as the interface with Ptolemy is using sc_fifo and the resulting design does not create a deadlock in reading/writing values from/to the Ptolemy interface fifo.

Setup

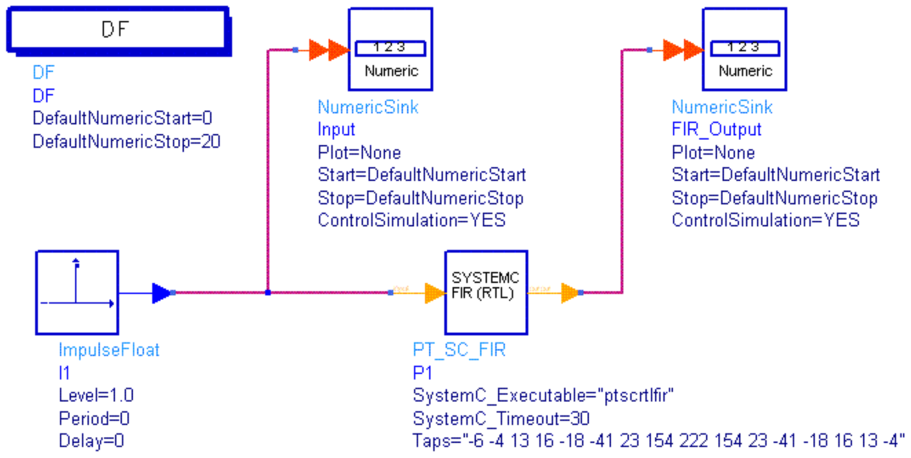
The SystemC FIR (RTL implementation) example that is shipped with OSCI SystemC 2.1, is modified in this example to cosimulate with ptolemy. The modified code is available in the following subdirectory of this workspace:

source_code/systemc_files/PT_SC_FIR

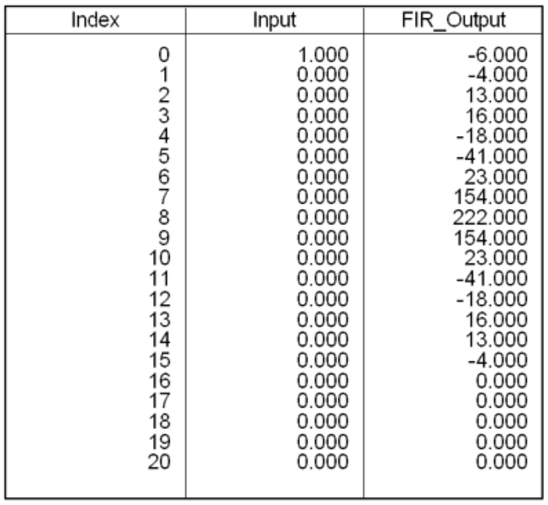
The stimulus and display SystemC modules are modified to read and write values to Ptolemy instead of generating ramp as stimuli and writing outputs to standard output. The *fir_top* and *fir_data* modules are also modified to read Taps values from *SystemcPtolemyInterface_PT_SC_FIR* instead of reading from a header file. The corresponding Ptolemy model can be found at:

source_code/pl_files/SDFPT_SC_FIR.pl

This example generates the impulse response of the FIR filter. The same Taps are used as in the standalone SystemC example which is shipped with OSCI SystemC 2.1.



Analysis



Index	Input	FIR_Output
0	1.000	-6.000
1	0.000	-4.000
2	0.000	13.000
3	0.000	16.000
4	0.000	-18.000
5	0.000	-41.000
6	0.000	23.000
7	0.000	154.000
8	0.000	222.000
9	0.000	154.000
10	0.000	23.000
11	0.000	-41.000
12	0.000	-18.000
13	0.000	16.000
14	0.000	13.000
15	0.000	-4.000
16	0.000	0.000
17	0.000	0.000
18	0.000	0.000
19	0.000	0.000
20	0.000	0.000

Ptolemy Cosimulation with SystemC Sine Wave Generator

Location: $HPEESOF_DIR /examples /SystemC_Cosim /SystemC_Cosim_wrk /SystemC_Cosim_lib /SineSource

Objective

This example shows SystemC cosimulation ability to write to a parameterized SystemC Source model. Where parameter values are specified in the ADS schematic and used in the SystemC model.

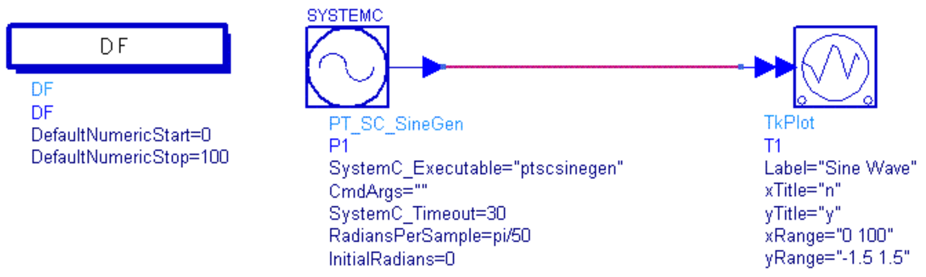
Setup

In this example, a Sine wave generator written in SystemC, is cosimulating with Ptolemy. User parameters can be changed in the schematic and read in SystemC code to generate the corresponding sine wave. SystemC code for this example is available in the following sub-directory of this workspace:

source_code/systemc_files/PT_SC_SineGen

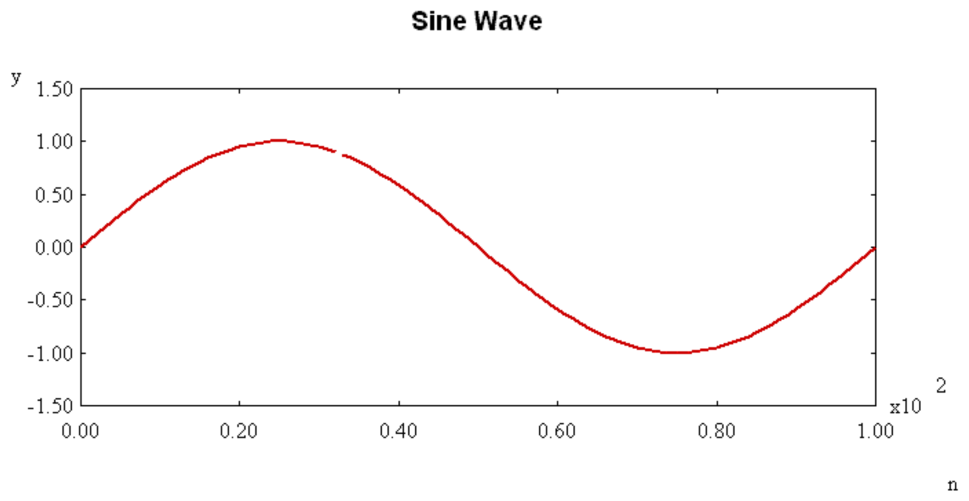
The corresponding Ptolemy model can be found at:

source_code/pl_files/SDFPT_SC_SineGen.pl



Analysis

Figure 1: Output of SystemC Sine wave generator



Ptolemy CoSimulation with SystemC UpSample

Location: $HPEESOF_DIR /examples /SystemC_Cosim /SystemC_Cosim_wrk /SystemC_Cosim_lib /UpSampler

Objective

This example shows how to cosimulate Ptolemy with a multi-rate SystemC design.

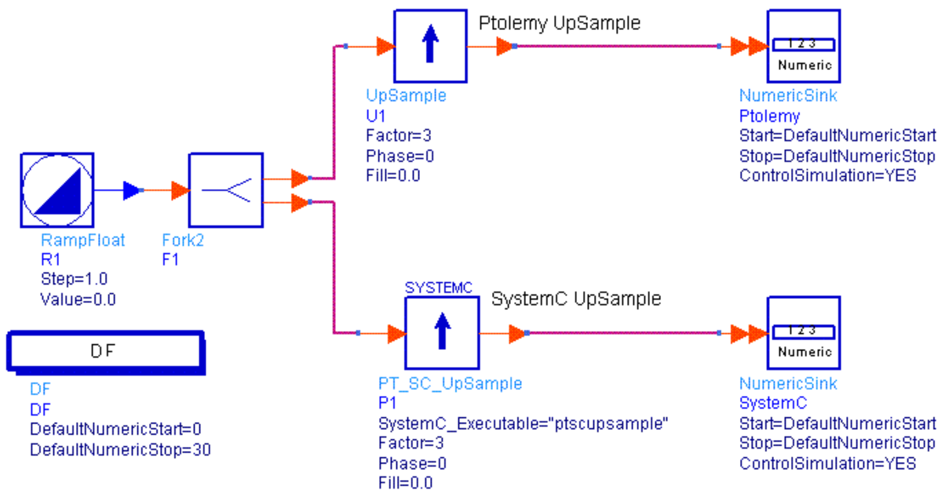
Setup

In this example, Ptolemy Cosimulates with an UpSampler written in SystemC. This design shows how to use multi-rate properties in SystemC Cosim. The multi-rate parameter (Factor) can be set in the Schematic and read in SystemC. For comparison purposes a native Ptolemy UpSample is also used in this design. The parameters Factor, Phase, and Fill can be changed in the ADS schematic to control these parameters in SystemC. SystemC code for this design is available in the following sub-directory of this workspace:

source_code/systemc_files/PT_SC_UpSample

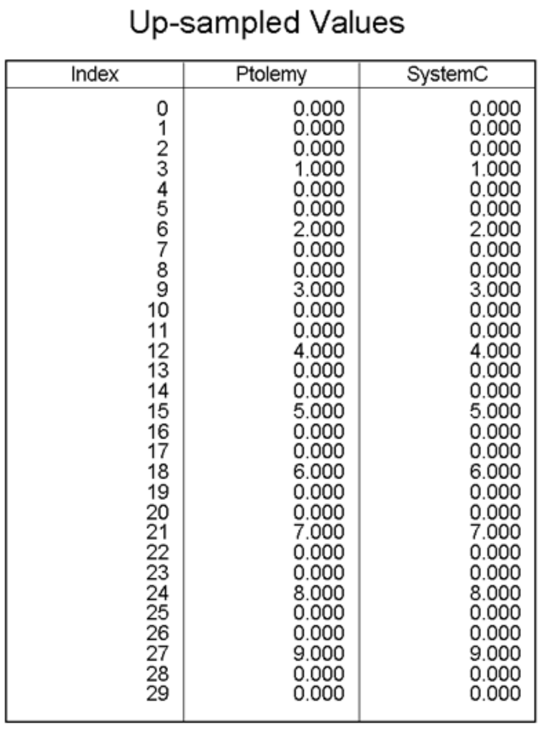
The corresponding Ptolemy model is located at:

source_code/pl_files/SDFPT_SC_UpSample.pl



Analysis

Figure 1: Upsampled values



Up-sampled Values

Index	Ptolemy	SystemC
0	0.000	0.000
1	0.000	0.000
2	0.000	0.000
3	1.000	1.000
4	0.000	0.000
5	0.000	0.000
6	2.000	2.000
7	0.000	0.000
8	0.000	0.000
9	3.000	3.000
10	0.000	0.000
11	0.000	0.000
12	4.000	4.000
13	0.000	0.000
14	0.000	0.000
15	5.000	5.000
16	0.000	0.000
17	0.000	0.000
18	6.000	6.000
19	0.000	0.000
20	0.000	0.000
21	7.000	7.000
22	0.000	0.000
23	0.000	0.000
24	8.000	8.000
25	0.000	0.000
26	0.000	0.000
27	9.000	9.000
28	0.000	0.000
29	0.000	0.000

Ptolemy-SystemC-Transient Cosimulation

Location: $HPEESOF_DIR /examples /SystemC_Cosim /SystemC_Cosim_wrk /SystemC_Cosim_lib /TransientCosim

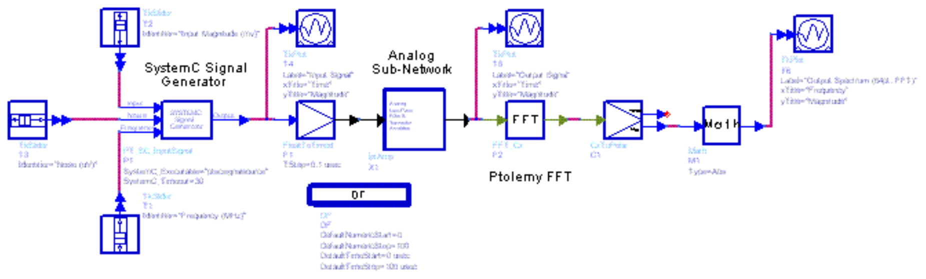
Objective

This example shows ADS Ptolemy's ability to simultanously cosimulate with SystemC and Transient Simulator.

Setup

In this example, Ptolemy's interactive TkSlider is used to select the input signal, noise amplitude and input frequency during simulation. These three items are sent to the SystemC signal generator. The signal generator creates the signal with input signal magnitude/frequency and adds variable noise. The output of the SystemC signal generator is sent to the analog simulator. In the analog sub-network, the signal passes through a 1 Mhz Butterworth Lowpass filter to eliminate the added noise, and then through a single transister amplifier stage. The resulting waveform is sent back to ADS Ptolemy, where an FFT is performed.

SystemC code for this example is available in the following sub-directory of this workspace:
source_code/systemc_files/PT_SC_InputSignal
The corresponding Ptolemy models are available at:
source_code/pl_files/SDFPT_SC_InputSignal.pl



Analysis

Figure 1: Analog subcircuit for Transient cosimulation

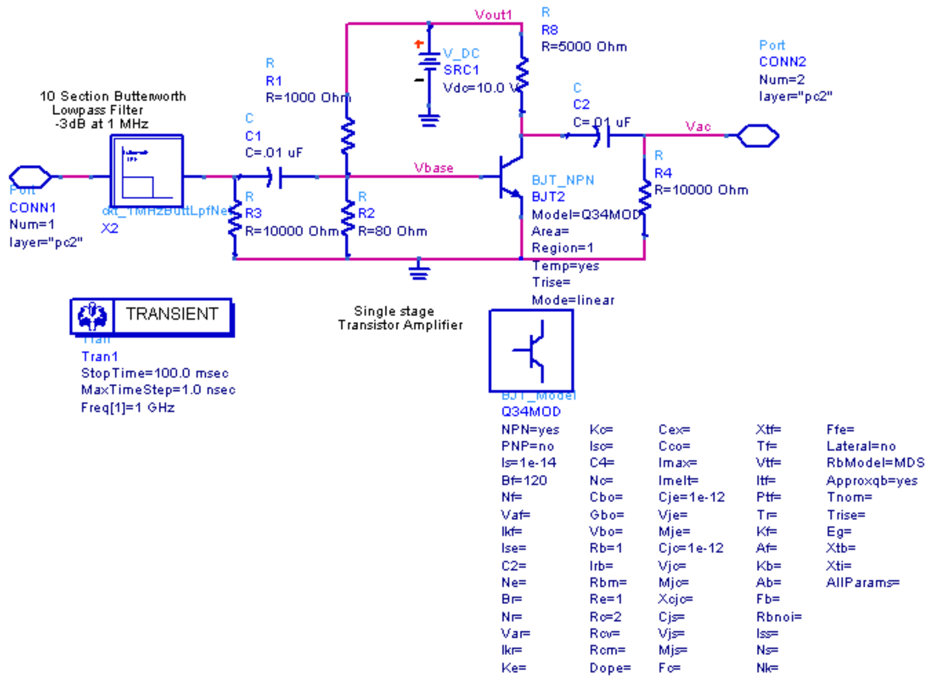


Figure 2: Input signal generated by SystemC model

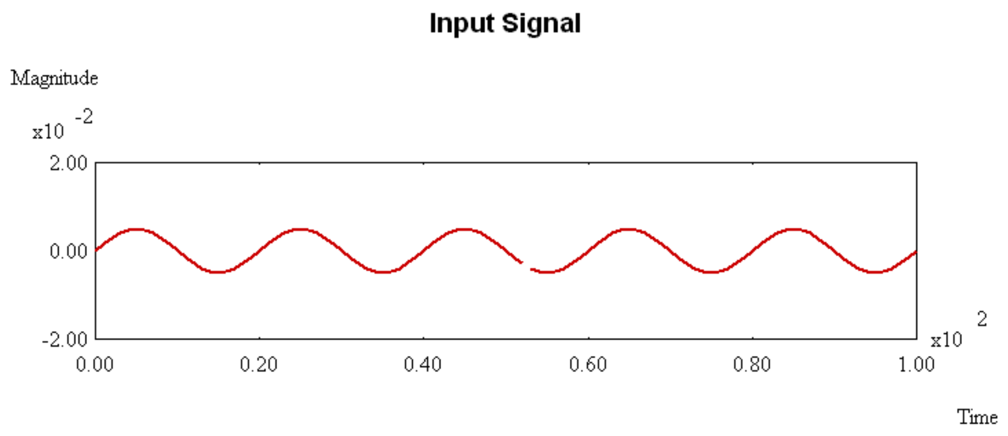


Figure 3: Output signal

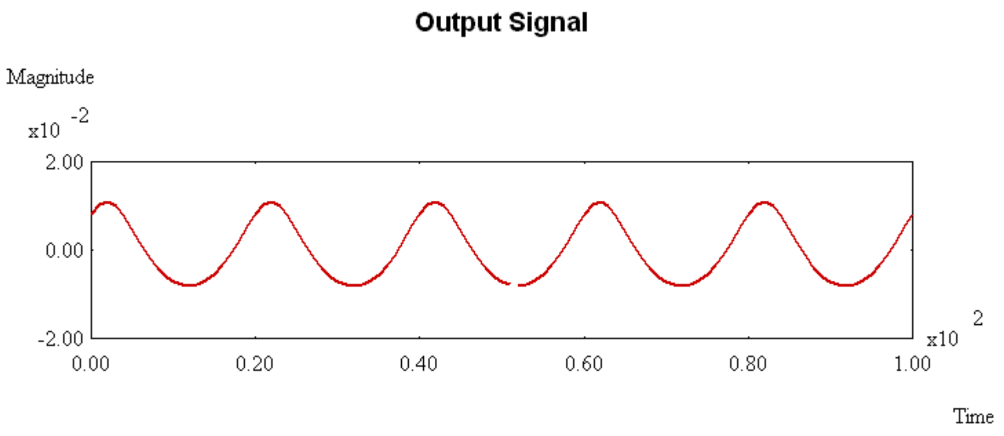


Figure 4: Output Spectrum

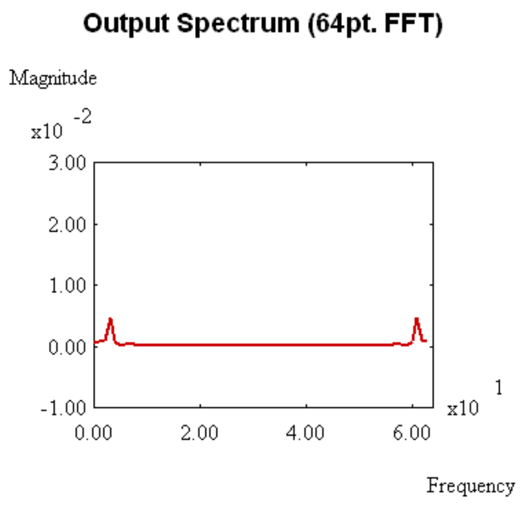
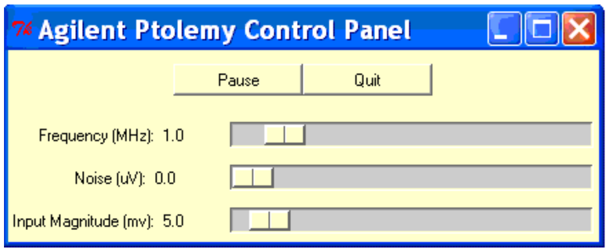


Figure 5: Slider bars for real-time input signal adjustments


Agilent Ptolemy Control Panel

Frequency (MHz): 1.0

Noise (uV): 0.0

Input Magnitude (mv): 5.0

RF Board Examples

Examples of how to design and translate into physical reality complex circuits such as phase-locked loops (PLLs), AGCs, and power amplifiers.

- *A 28-32GHz 3-dB Lange Coupler Simulation* (examples)
- *Comparing SPICE Model and ADS Distributed Model for Coupled Transmission Line Simulation* (examples)
- *Fifth-Order Phase-Locked Loop* (examples)
- *Fraction-N PLL* (examples)
- *Frequency and Time Domain Simulation of Multi-Coupled Microstrip Lines* (examples)
- *Graphical Cell Compiler Examples* (examples)
- *Loadpull Simulations in ADS* (examples)
- *Low Power Mixer for Pager Applications* (examples)
- *Multi-Layer Printed Circuit Board Filter* (examples)
- *NADC Power Amplifier* (examples)
- *Open and Closed Loop Simulation of PLL* (examples)
- *Optimization of A Multi-Harmonic Source and Load* (examples)
- *PCS Cellular Power Amplifier Design and Analysis* (examples)
- *Phase Noise Simulation* (examples)
- *Phase Noise Simulations using Small Signal Model Loop Components* (examples)
- *PLL Simulation of DECT Radio System* (examples)
- *Power Amplifier Layout and Design Rule Checker* (examples)
- *Power Amplifier using Cartesian Feedback* (examples)
- *Power Amplifier with an IS-95 CDMA Source Input* (examples)
- *Simulation of An Automatic Gain Control Loop* (examples)
- *TDR Analysis of Board Level Crosstalk* (examples)
- *TDR and S-parameter Simulations of Microstrip Step Discontinuities* (examples)

A 28-32GHz 3-dB Lange Coupler Simulation

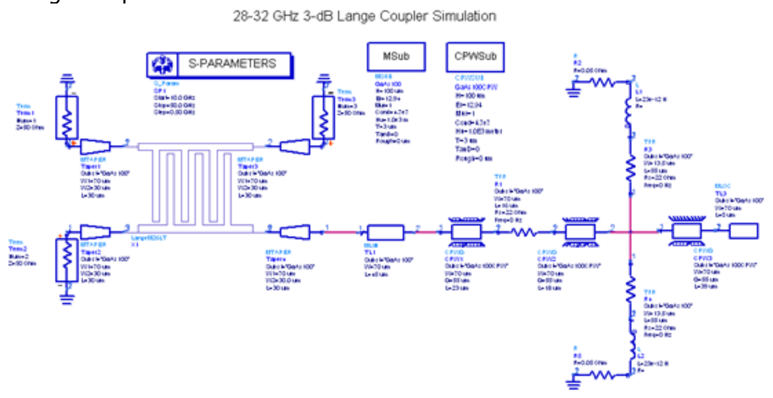
Location: $HPEESOF_DIR/examples/RF_Board/LangeCoupler_wrk

Objective

This example simulates a 3-dB hybrid Lange coupler in the 28 to 32GHz range and compares simulations with measurements.

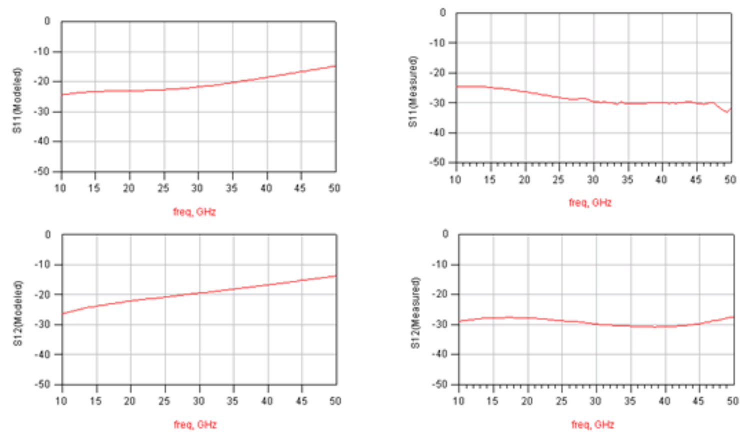
Setup

1. "LangeM26LT"is the Lange coupler design. It is a 6-wire Lange coupler with the lines on the second metal, and a right-angle 50-ohm termination with an added via connection.
2. "lngcp" contains the simulation controllers and passive terminations for testing the Lange coupler.



Analysis

Figure 1: Good agreement between simulation and measurement



Notes

- Good modeling of the coupling effect is attributed to the good performance of the "multilayer coupled line" components.
- Obtaining results similar to the measured data is also attributed to carefully modelling a right-angle, 50 ohm termination using microstrip, coplanar waveguide, and thin film resistor segments.
- All data show close agreements except for the quadrature phase.

Comparing SPICE Model and ADS Distributed Model for Coupled Transmission Line Simulation

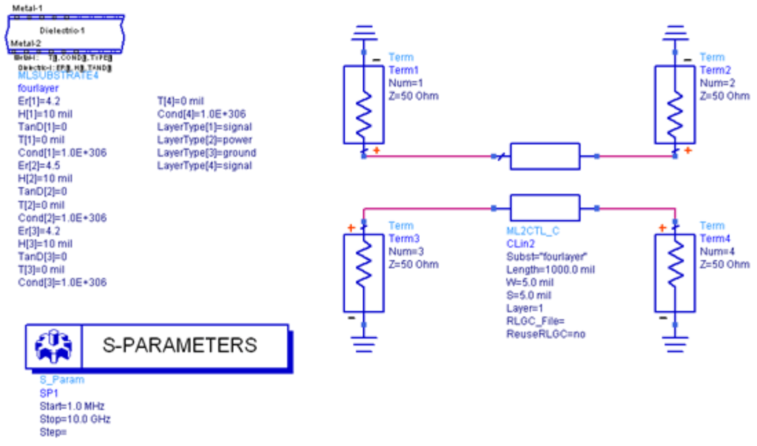
Location: $HPEESOF_DIR/examples/RF_Board/SpiceModel_wrk

Objective

This example compares the S-parameters from an edge-coupled transmission line simulation and a SPICE model equivalent circuit simulation.

Setup

1. "coupledLine" simulates the S-parameter using a distributed edge-coupled transmission line model.
2. "smgCoupledLine" uses a SPICE model in the S-parameter simulation.



MLSUBSTRATE4
fourlayer
Er[1]=4.2
H[1]=10 mil
TanD[1]=0
T[1]=0 mil
Cond[1]=1.0E+306
Er[2]=4.5
H[2]=10 mil
TanD[2]=0
T[2]=0 mil
Cond[2]=1.0E+306
Er[3]=4.2
H[3]=10 mil
TanD[3]=0
T[3]=0 mil
Cond[3]=1.0E+306
T[4]=0 mil
Cond[4]=1.0E+306
LayerType[1]=signal
LayerType[2]=power
LayerType[3]=ground
LayerType[4]=signal

Term
Term1
Num=1
Z=50 Ohm

Term
Term2
Num=2
Z=50 Ohm

Term
Term3
Num=3
Z=50 Ohm

ML2CTL_C
CLin2
Subst="fourlayer"
Length=1000.0 mil
W=5.0 mil
S=5.0 mil
Layer=1
RLGC_File=
ReuseRLGC=no

Term
Term4
Num=4
Z=50 Ohm

S_Param
SP1
Start=1.0 MHz
Stop=10.0 GHz
Step=

Analysis

Figure 1: Agreement for the coupled path

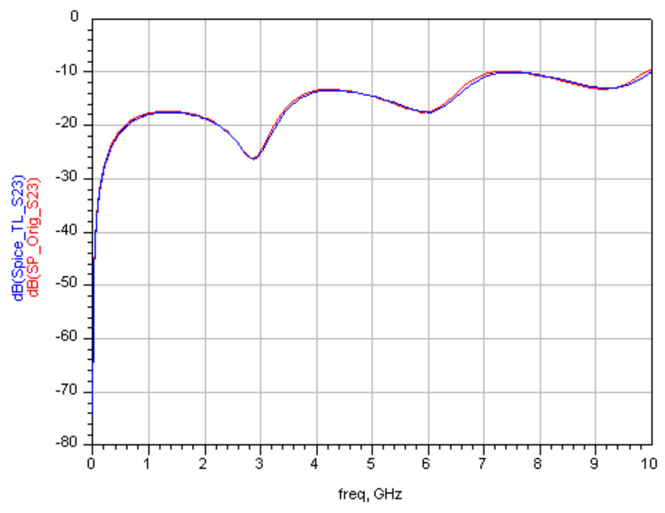
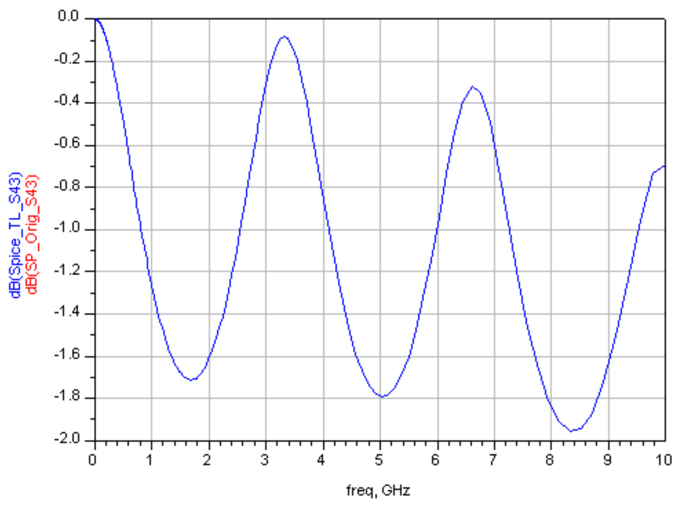


Figure 2: Agreement for the through path. (/examples/RF_Board/SpiceModel_wrk/comparison.dds)



Notes

- Simulation controller used: S-Parameters.

Fifth-Order Phase-Locked Loop

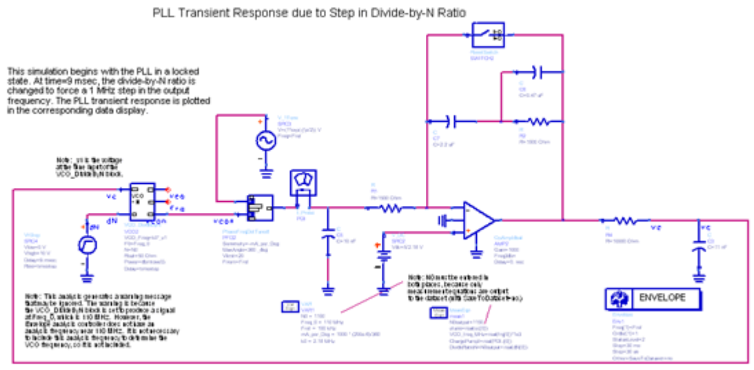
Location: $HPEESOF_DIR/examples/RF_Board/PLL_Examples/PLL_5th_Order_wrk

Objective

This example shows Envelope simulations of several block level phase-locked loop (PLL) transient responses, in which the voltage-controlled oscillator (VCO) frequency changes in response to a step in divide ratio.

Setup

1. "PLL_1_MHz_step" shows the transient response when the VCO frequency changes by 1 MHz.
2. "PLL_1pt5_MHz_step" shows the transient response when the VCO frequency changes by 1.5 MHz.
3. "PLL_4_MHz_step" shows the transient response when the VCO frequency changes by 4 MHz.
4. "PLL_1_MHz_step_BBPFD" shows the transient response when a charge-pump phase/frequency detector is used.


PLL Transient Response due to Step in Divide-by-N Ratio
This simulation begins with the PLL in a locked state. At time=9 msec, the divide-by-N ratio is changed to force a 1 MHz step in the output frequency. The PLL transient response is plotted in the corresponding data display.
ENVELOPE

Analysis

Figure 1: VCO tune voltage and Divide ratio versus time

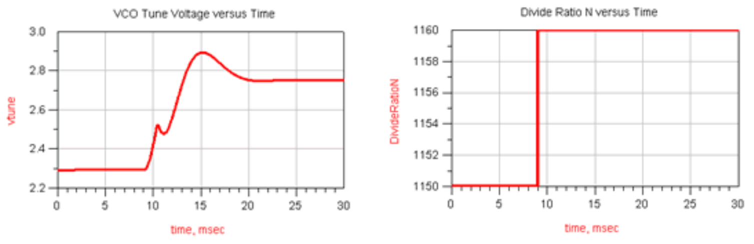
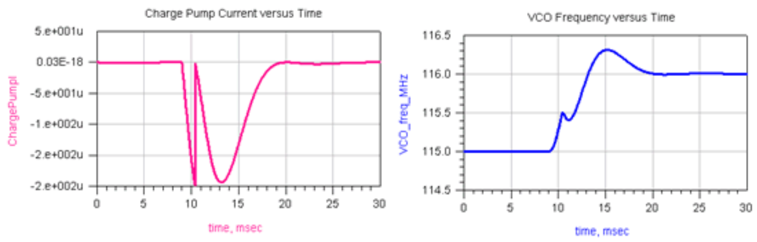


Figure 2: Charge pump current and VCO frequency versus time



Fraction-N PLL

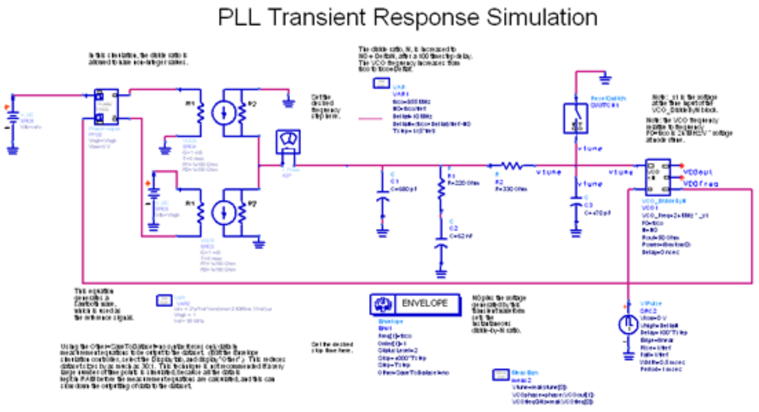
Location: $HPEESOF_DIR/examples/RF_Board/PLL_Examples/PLL_FracN_wrk

Objective

This example shows the transient response simulation of a phase-locked loop (PLL) using several different approaches.

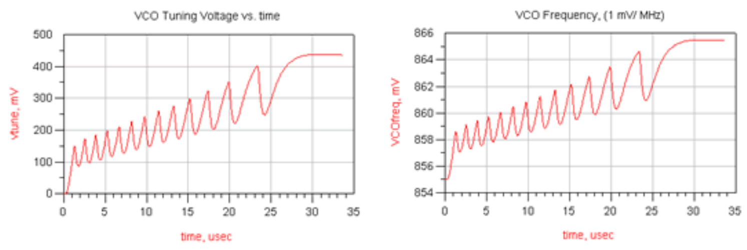
Setup

1. "PFDchpmpTran" uses a charge-pump behavioral model of a phase-frequency detector, with external current sources, and uses the transient simulator (similar to SPICE). Frequency versus time, tuning voltage versus time, and a plot of the charge pump current pulses versus time are the outputs.
2. "PFDchpmpEnv" is the same as "PFDchpmpTran", except that output includes voltage-controlled oscillator (VCO) relative phase versus time. Also, it takes much longer to run, and generates a much larger dataset.
3. "PFDtuned" is a similar simulation, except with a more ideal "tuned" phase-frequency detector.
4. "FracNsim" simulates a fractional-N synthesizer PLL. This PLL is from /examples/RF_Board/PLL_Examples/DECT_LO_Synth_wrk. In a fractional-N synthesizer, the divide ratio is toggled between N and N+1, which allows the VCO to be tuned in smaller steps than the reference frequency.
5. "FracNsim3" is similar to "FracNsim", except that the loop design is quite different, and it uses a much smaller divide ratio. This simulation behaves as expected, with the VCO settling out to the correct frequency.



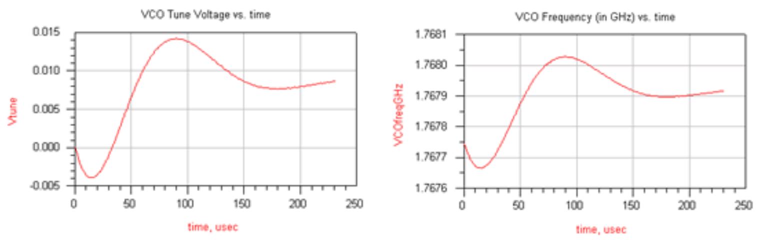
Analysis

Figure 1: VCO tuning voltage and VCO frequency versus time from transient analysis



With divide ratio N stepped causing a 10 MHz increase in the VCO frequency from 855 MHz to 865 MHz.

Figure 2: VCO tuning voltage and VCO frequency versus time from the Fraction-N PLL



Frequency and Time Domain Simulation of Multi-Coupled Microstrip Lines

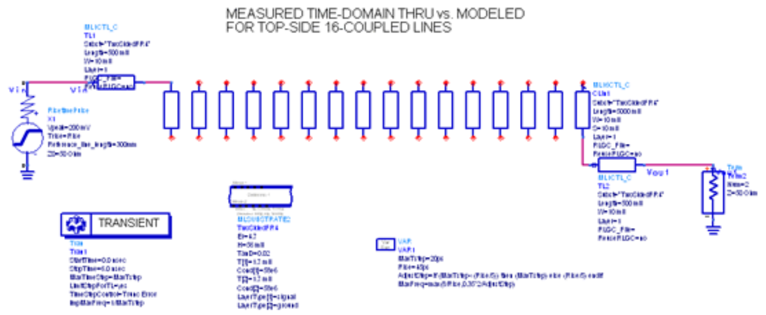
Location: $HPEESOF_DIR/examples/RF_Board/MultilayerMeas_wrk

Objective

This design compares measured and simulated S-parameters and time domain reflectometry (TDR) response data for several coupled microstrip lines. Some designs also compare different methods of simulating transmission lines.

Setup

1. "SP_16line" and "TDR_16line" are simulation setups for determining cross-talk/coupling in 16 coupled lines in frequency- and time-domains, respectively.
2. "SP_oneline", "SP_2line" and "SP_4line" show several ways of modeling the frequency response for a single line and for 2 and 4 coupled lines. Results are compared to measured data in the data display pages.
3. "TDR_2line" also shows a time-domain (TDR) simulation for the case of 2 coupled lines.


MEASURED TIME-DOMAIN THRU vs. MODELED
FOR TOP-SIDE 16-COUPLED LINES
TRANSIENT
Vin
Vout

Analysis

Figure 1: Comparison of measured and simulated time domain response of a 16 coupled line

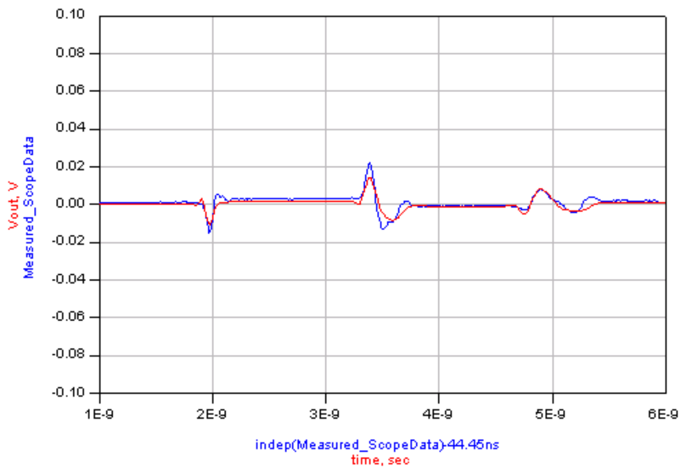
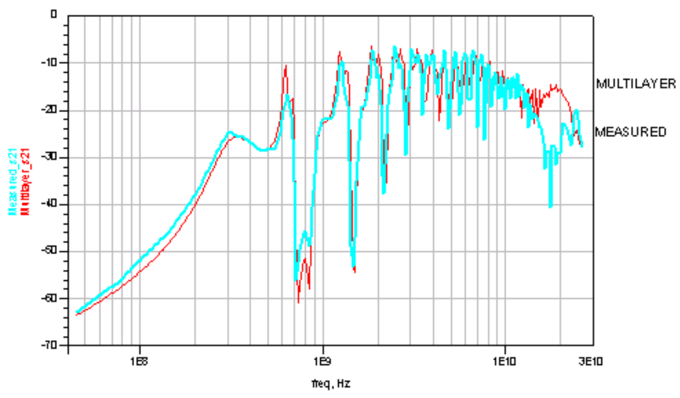


Figure 2: Comparison of measured and simulated S21 of a 16 coupled line



Notes

1. Simulation controllers used: S Parameters, Transient.

Graphical Cell Compiler Examples

Location: $HPEESOF_DIR/examples/RF_Board/gcc_examples_wrk

Objective

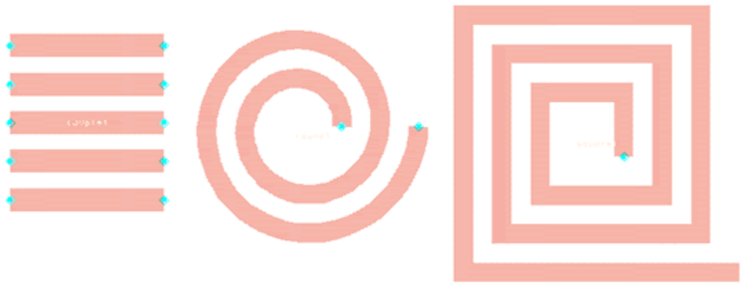
The Graphical Cell Compiler (GCC) makes the job of adding parameterized artwork to a layout convenient. This workspace contains four designs that are the implementation of examples used in *Graphical Cell Compiler* (gcc). They are not set-up to be simulated here.

Setup

1. "couple" implements a simple multi-coupled line model.

2. "round" implements a simple rounded spiral model.
3. "square" implements a more complex square spiral example.
4. "demo" has a single instance of each of the three models inserted for easy viewing and editing.

Figure 1: Model Layouts



Notes

- For another example of a FET refer to /examples/RFIC/GCC_FET_wrk.

Loadpull Simulations in ADS

Location: $HPEESOF_DIR/examples/RF_Board/LoadPull_wrk

Objective

This example shows how to do load-pull simulations. These simulations generate contours that indicate load impedances that when presented to the output of a device (along with the specified source impedance and available source power) would cause a certain power to be delivered to the load.

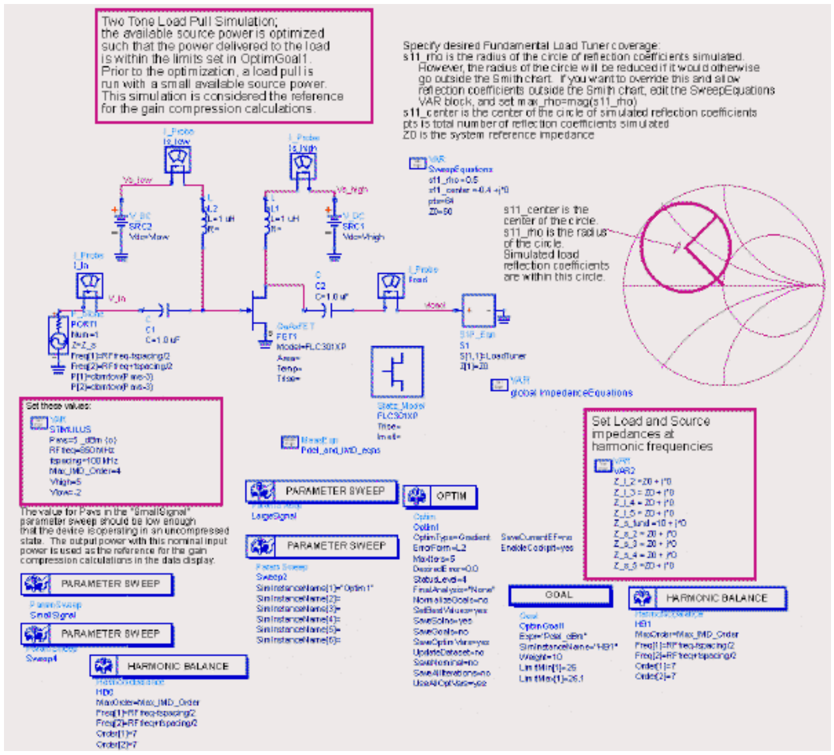
Setup

These simulations are discussed in detail in two PDF files in this example workspace directory, **LoadPullPres.pdf** and **LoadPullConstPdel.pdf**. You may view them with Adobe Reader. This example shows how to do load-pull simulations. These simulations generate contours that indicate load impedances that, when presented to the output of a device (along with the specified source impedances and available source power), would cause a certain power to be delivered to the load. Note that only the impedance at the fundamental frequency is varied (although these setups could be modified easily to allow varying an impedance at an arbitrary harmonic frequency.) Impedances at the harmonic frequencies may be set independently.

1. **HB1Tone_LoadPull** is a simulation set-up that generates actual contour lines for output power and power-added efficiency.
2. The schematic **HB1Tone_LoadPull_eqns** is identical to **HB1Tone_LoadPull**, except that it shows all the equations, and includes some explanatory text.
3. **HB1Tone_LoadPull.dds**, displays the output power and PAE contours and has an equations page with some documentation of their syntaxes.
4. **ReflectionCoefUtility.dds** shows how the variables s11rho and s11center determine the circular region of the Smith Chart within which load reflection coefficients are generated.
5. **HB1Tone_LoadPullMagPh** is identical to **HB1Tone_LoadPull**, except that it uses a much simpler simulation setup. Instead of generating loads in a circular region of the Smith Chart, the magnitude and phase of the load reflection coefficient are swept independently.
6. **HB1Tone_LoadPullPSweep** includes a power sweep at each load, so you may see the gain compression characteristics of the device.
7. **HB1Tone_LoadPull_vsFreq** repeats the fixed available source power simulation as a function of frequency.
8. **HB1Tone_LoadPull_ConstPdel** uses an optimization to vary the available source power on the schematic until a desired power is delivered to each load impedance. Contours of constant power-added efficiency, constant bias current, constant gain, and constant gain compression are plotted on the data display.
9. **LoadPullMagPh_ConstPdel** is similar to **HB1Tone_LoadPull_ConstPdel**, except that it varies the magnitude and phase of the load reflection coefficient rather than sweeping out a circular region of the Smith Chart. Also the data display has contours of constant operating and transducer power gain.

10. **HB2Tone_LoadPull** is a simulation set-up that generates actual contour lines for output power, power-added efficiency, third-order intermodulation distortion, and fifth-order intermodulation distortion.
11. **HB2Tone_LoadPull.dds** has the contour plots on one page, and the equations used to calculate output power, PAE, 3rd-order IMD, and 5th-order IMD on another.
12. **HB2Tone_LoadPull_ConstPdel** uses an optimization to vary the available source power on the schematic until a desired power is delivered to each load impedance. This is run with two input tones so includes contours of constant intermodulation distortion.
13. **HB2Tone_LoadPullMagPh** is identical to **HB2Tone_LoadPull**, except that it uses a much simpler simulation setup. Instead of generating loads in a circular region of the Smith Chart, the magnitude and phase of the load reflection coefficient are swept independently.
14. **TwoTone_LoadPullEnv** shows a method of running a two-tone load pull simulation by amplitude modulating a one-tone signal. It uses the Envelope simulator. In some extreme cases, this might be worthwhile because it uses less memory than a 2-tone harmonic balance simulation.
15. **TwoTone_LoadPullEnvMagPh** is similar to **TwoTone_LoadPullEnv**, except that it sweeps out a pie-shaped region of the Smith Chart instead of a circular region.

Analysis



Two Tone Load Pull Simulation;
the available source power is optimized such that the power delivered to the load is within the limits set in OptimGoal1.
Prior to the optimization, a load pull is run with a small available source power.
This simulation is considered the reference for the gain compression calculations.

Specify desired Fundamental Load Tuner coverage:
s11_rho is the radius of the circle of reflection coefficients simulated.
However, the radius of the circle will be reduced if it would otherwise go outside the Smith chart. If you want to override this and allow reflection coefficients outside the Smith chart, edit the SweepEquations VAR block, and set max_rho=mag(s11_rho)
s11_center is the center of the circle of simulated reflection coefficients
pts is total number of reflection coefficients simulated
Z0 is the system reference impedance

VAR
SweepEquations
s11_rho=0.5
s11_center=-0.4+j*0
pts=64
Z0=50

s11_center is the center of the circle.
s11_rho is the radius of the circle.
Simulated load reflection coefficients are within this circle.

Set these values:
VAR
STIMULUS
Pavs=5 _dBm {o}
RFfreq=850 MHz
fspacing=100 kHz
Max_IMD_Order=4
Vhigh=5
Vlow=-2

The value for Pavs in the "SmallSignal" parameter sweep should be low enough that the device is operating in an uncompressed state. The output power with this nominal input power is used as the reference for the gain compression calculations in the data display.

PARAMETER SWEEP
ParamSweep
SmallSignal

PARAMETER SWEEP
ParamSweep
Sweep1

HARMONIC BALANCE
HarmonicBalance
HB0
MaxOrder=Max_IMD_Order
Freq[1]=RFfreq-fspacing/2
Freq[2]=RFfreq+fspacing/2
Order[1]=7
Order[2]=7

PARAMETER SWEEP
ParamSweep
LargeSignal

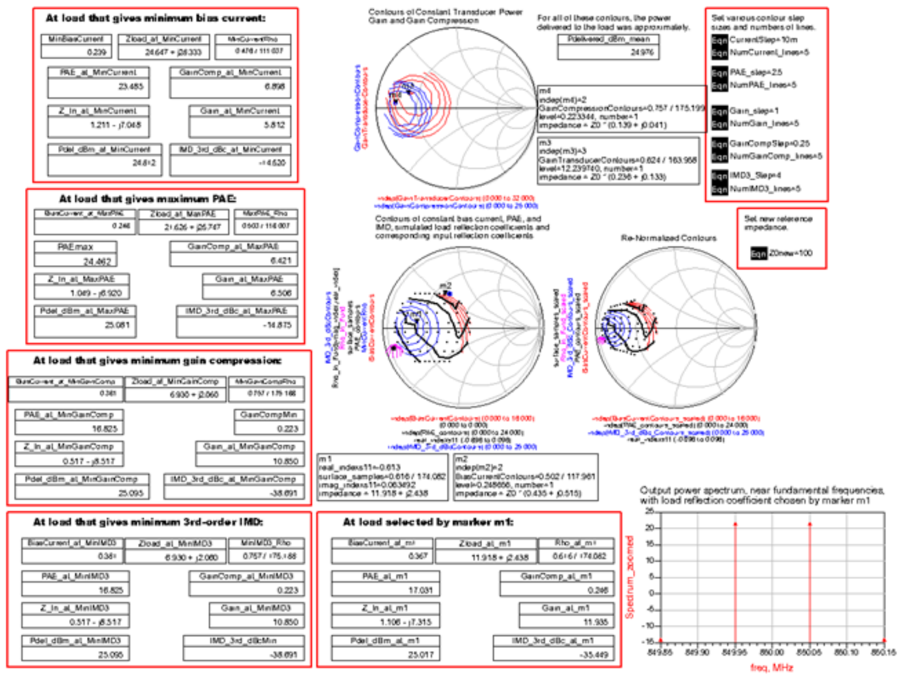
PARAMETER SWEEP
ParamSweep
Sweep2
SimInstanceName[1]="Optim1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=

OPTIM
Optim
Optim1
OptimType=Gradient
ErrorForm=L2
MaxIters=5
DesiredError=0.0
StatusLevel=4
FinalAnalysis="None"
NormalizeGoals=no
SetBestValues=yes
SaveSolns=yes
SaveGoals=no
SaveOptimVars=yes
UpdateDataset=no
SaveNominal=no
SaveAllIterations=no
UseAllOptVars=yes
SaveCurrentEF=no
EnableCockpit=yes

GOAL
Goal
OptimGoal1
Expr="Pdel_dBm"
SimInstanceName="HB1"
Weight=10
LimitMin[1]=25
LimitMax[1]=26.1

Set Load and Source impedances at harmonic frequencies
VAR
VAR2
Z_l_2 = Z0 + j*0
Z_l_3 = Z0 + j*0
Z_l_4 = Z0 + j*0
Z_l_5 = Z0 + j*0
Z_s_fund = 10 + j*0
Z_s_2 = Z0 + j*0
Z_s_3 = Z0 + j*0
Z_s_4 = Z0 + j*0
Z_s_5 = Z0 + j*0

HARMONIC BALANCE
HarmonicBalance
HB1
MaxOrder=Max_IMD_Order
Freq[1]=RFfreq-fspacing/2
Freq[2]=RFfreq+fspacing/2
Order[1]=7
Order[2]=7



Low Power Mixer for Pager Applications

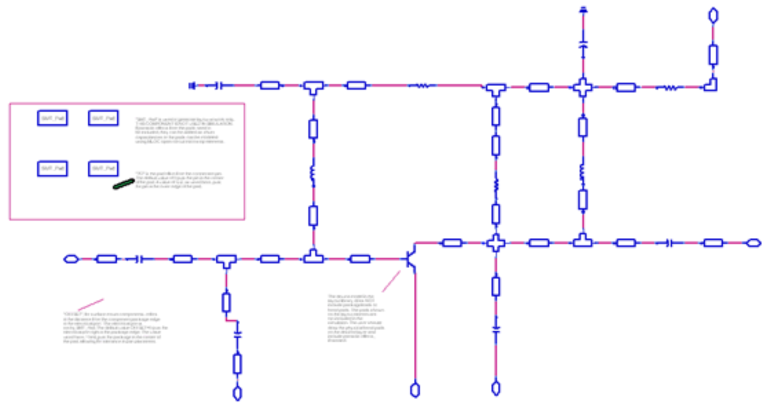
Location: $HPEESOF_DIR/examples/RF_Board/MixerPager_wrk

Objective

This examples shows a complete design of a low power mixer using Advanced Design System (ADS) for pager applications. Vdd is 1V, RF is at 900MHz, LO is 855MHz. It also shows a comparison between the ideal simulation and that from the layout.

Setup

1. DC operation point set up is achieved through three designs. "DC_curves" obtains the I-V curve of the device. "BiasPoint" establishes the operating point and bias resistor values. "BiasNet" checks the S-parameter of the biased device.
2. "Compression" shows two methods for calculating the compression point of the device.
3. "RFIFmatch1" determines input matching at RF and output matching at IF.
4. "LOdrive" calculates the mixer response as Lo drive sweeps.
5. "MixCompr" shows that the mixer compression point can be obtained by either sweeping the RF power or using the built in "Gain Compression" simulator.
6. "MixerLayout" contains the layout and schematic of the mixer. Simulating this design is done from "SimFromLayout", which contains the symbol for MixerLayout, together with the components needed to simulate it. Results are compared to the ideal simulation, showing the effects of surface-mount component parasitics and losses.



Analysis

Figure 1: Layout of the mixer

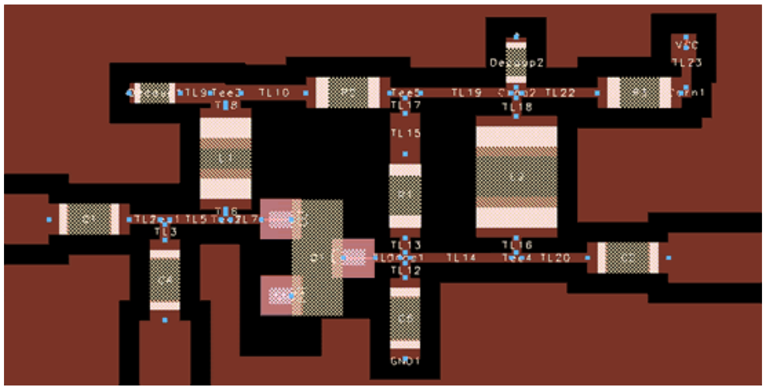


Figure 2: Mixer 1dB gain compression point using marker readout

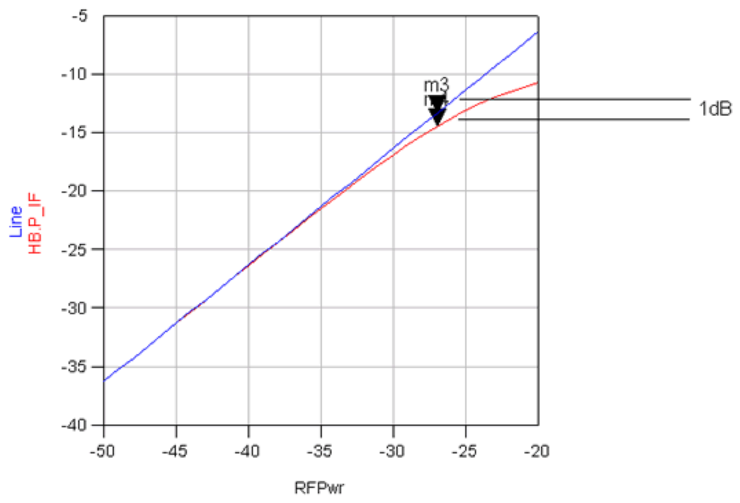
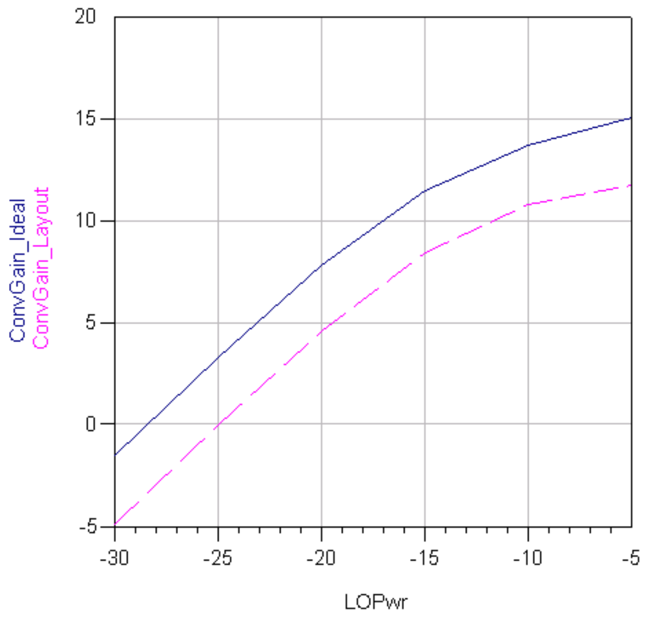


Figure 3: Conversion gain: ideal case and simulation from layout



Calculated conversion gain using surface mount components and microstrip elements is 3dB lower than ideal case. This is mostly attributable to losses in the inductors. Note that the load resistor is increased up to 3.3kOhm to help compensate for reduced gain.

Notes

- Simulation controllers used: Harmonic Balance, Gain Compression, DC, S Parameters
- A conventional way of determining 1dB compression point is to plot power gain and

finding the input power for which gain is 1dB lower than the small-signal value. An equivalent method is to plot Pout vs. Pin and find the point at which Pout falls off 1dB from the ideal linear response.

Multi-Layer Printed Circuit Board Filter

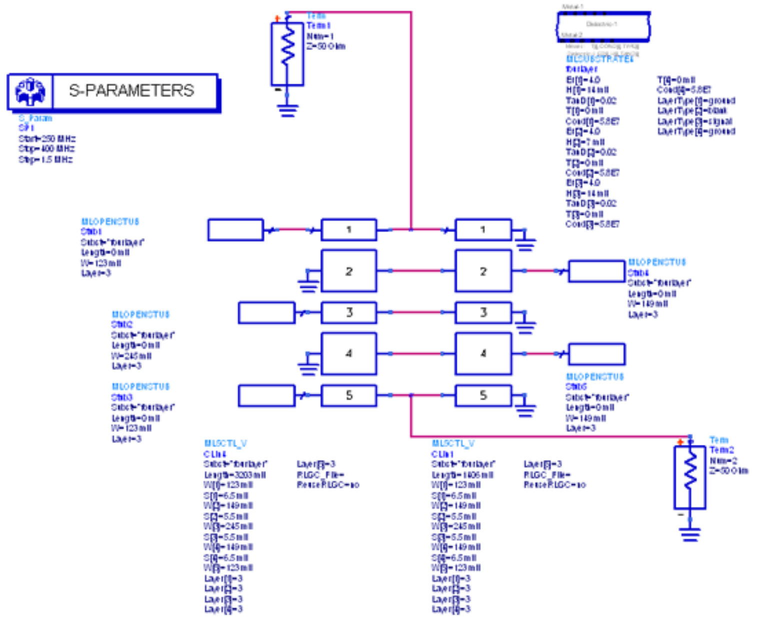
Location: $HPEESOF_DIR/examples/RF_Board/MLfilter_wrk

Objective

This design shows an edge coupled filter at 321.4MHz, realized using an Advanced Design System (ADS) five coupled line model on a four layer substrate.

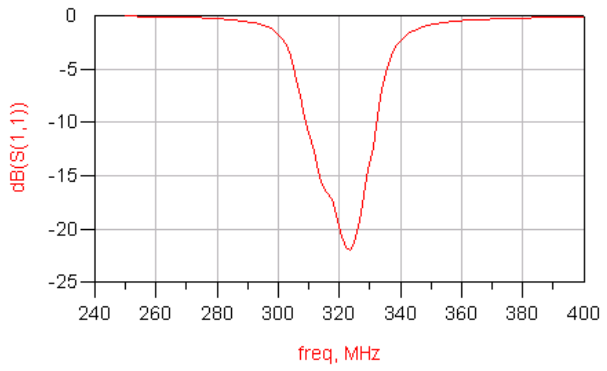
Setup

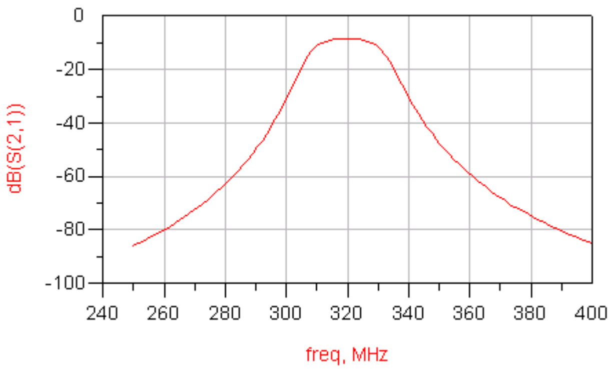
1. "Mlfilter" contains both the schematic and layout of the filter. A four layer substrate is used, where the outer layers are ground planes, shielding the rest of the circuit from the RF. The filter lines are placed on one of the inner layers, resulting in an unbalanced stripline configuration.



Analysis

Figure 1: Frequency response of the filter





Notes

- Simulation controller used: S Parameters.
- Printed circuit boards offer low cost alternatives to implementing passive RF filters. The multilayer interconnect models can help evaluate the performance of different filter types and materials to optimize a design.

NADC Power Amplifier

Location: $HPEESOF_DIR/examples/RF_Board/NADC_PA_wrk

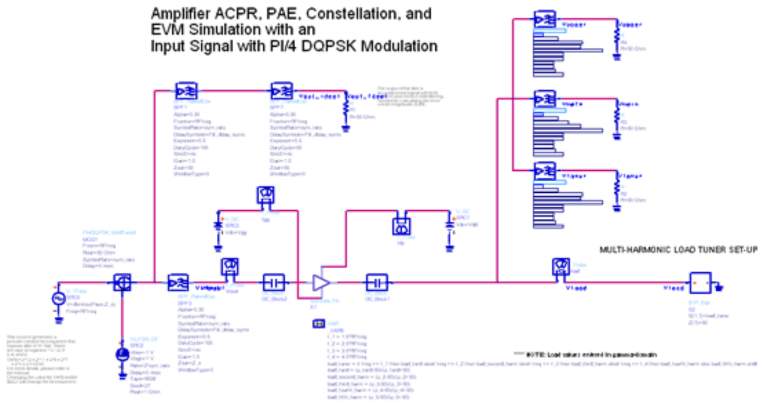
Objective

This example shows several simulations and layout of a 850MHz power FET and an amplifier using it. Through simulations such as the small signal S-parameter, one tone and two-tone Harmonic Balance and Circuit Envelope simulations with a pi/4 DQPSK-modulated source, PA properties such as the load-pull, intermodulation, Power-Added Efficiency (PAE), gain, Adjacent Channel Power Ratios (ACPR), constellation, trajectory and error vector magnitude (EVM), etc, can be characterized.

Setup

1. "DCTest" is a curve tracer simulation that generates the I-V curves.
2. "Load_pull" allows the user to sweep the load impedance at the fundamental frequency and set the impedances at the harmonic frequencies separately to obtain load-pull contours.
3. "Motorola_Mosfet_Model" is an HP_MOS model with parasitics. "Motorla_PA" shows a power amplifier with input and output impedance matching networks. "Motorola_Palayout" is the same design with an added layout.
4. "PA_Sparam" shows the small signal S-parameter simulation. "PA_HB1tone" is a large-signal, single input tone harmonic balance simulation. "PA_HB2tone" shows a two-tone large-signal harmonic balance simulation.
5. "NADC_PA_Test" simulates the amplifier with a pi/4 DQPSK-modulated signal, corresponding to the NADC standard. The user may set the source power, modulation data bit sequence, source and load impedances versus harmonic frequency, and other parameters. The set-up includes a pair of filters for generating an undistorted signal for error vector magnitude, as well as a set of filter banks so the effects of receive-side filtering in the main and adjacent channels can be included.
6. "NADC_FET_Test" is a set-up identical to the NADC_PA_Test, except that instead of the amplifier the FET without any matching network is simulated. "NADC_Src_Test" is a set-up for testing the source by itself. There is no circuit to cause distortion, so the EVM should be very close to zero. Also, the ACPRs should be as close to ideal as is

allowed by the filtering used. (In the NADC system, because of a small channel overlap, even without a circuit to cause distortion, the ACPR will be non-zero.) The results of this simulation are displayed in "ConstEVMSrcTest.dds", "NADC_Src_ACPRtransmitted.dds", and "NADC_Src_ACPRreceived.dds. "SmampHBtest" is a simple, one tone, swept input power harmonic balance analysis of a system model (behavioral) amplifier.


Amplifier ACPR, PAE, Constellation, and
EVM Simulation with an
Input Signal with PI/4 DQPSK Modulation
MULTI-HARMONIC LOAD TUNER SET-UP

Analysis

Figure 1: Transmitted spectrum and lower and upper channel ACPR values

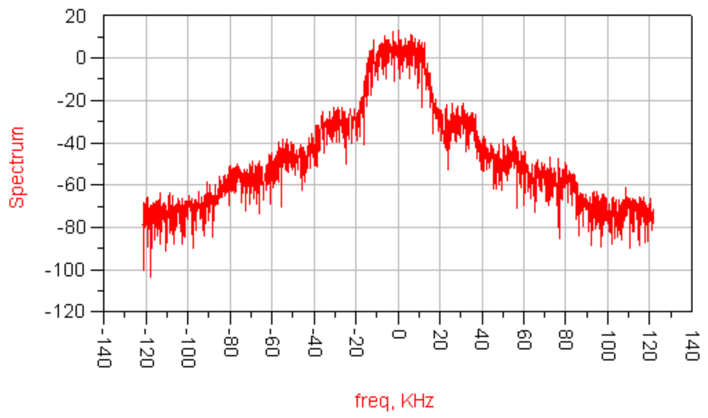


Figure 2: Trajectory diagram of transmitted signal

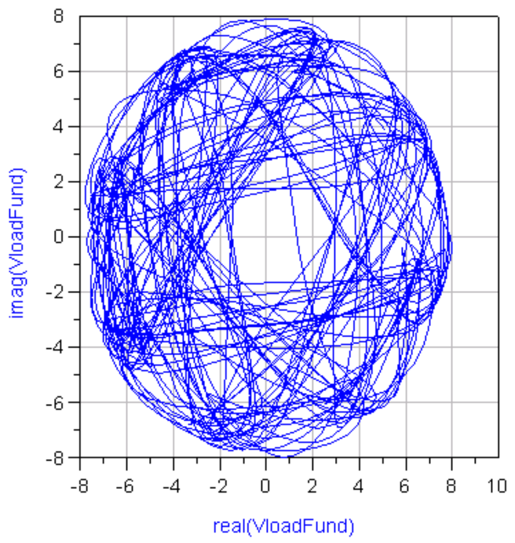


Figure 3: Main channel and adjacent channel spectra after receive-side filtering

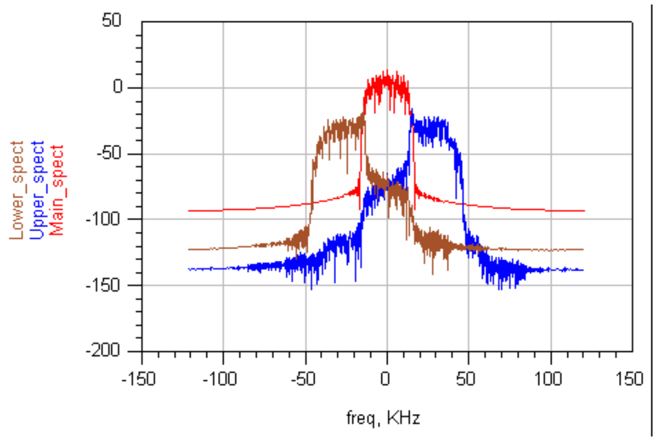


Figure 4: Ideal and distorted signal trajectory and constellation

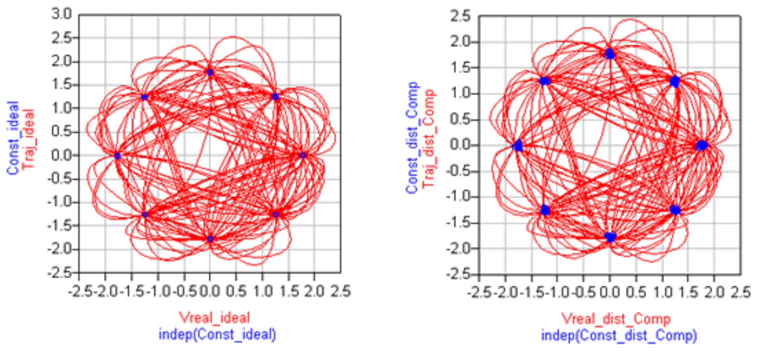
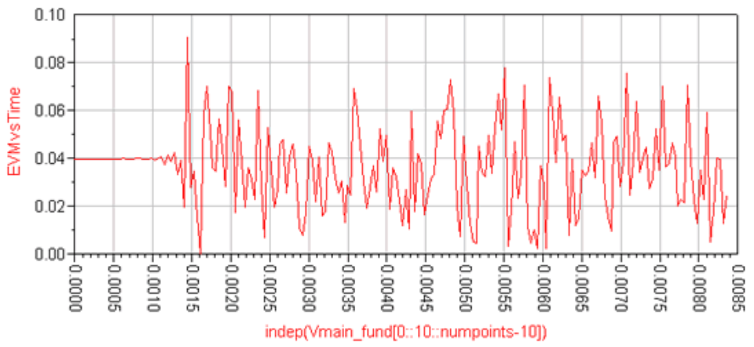


Figure 5: Error vector magnitude versus time



Notes

- Simulation controllers used: S Parameters, Harmonic Balance, Circuit Envelope.
- "Motorola_PAlayout" can be used to demonstrate the Intermediate File Format (IFF) file transfer to Mentor Graphics. To do this use the File > Export command from the layout window. Use the MGC/PCB option for file type. This creates a new directory within the workspace directory called "to_mgc" which contains the schematic and layout .iff files.

Open and Closed Loop Simulation of PLL

Location: $HPEESOF_DIR/examples/RF_Board/PLL_Examples/PLL_Freq_Resp_wrk

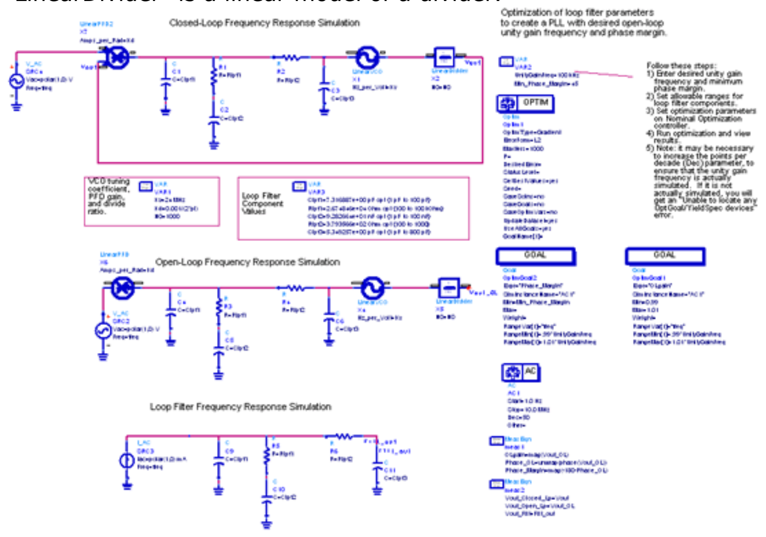
Objective

This example shows various simulations of a phase-locked loop (PLL), including open- and closed-loop frequency responses, and an optimization of open-loop unity gain frequency and phase margin.

Setup

1. "PLL_Freq_Resp" simulates the frequency response of a PLL, with linear models for

the phase-frequency detector, voltage-controlled oscillator (VCO) and divider. "PLL_Freq_Resp.dds" shows the open- and closed-loop amplitude responses, the open- and closed-loop phase responses, the unity-gain frequency and the phase margin at the unity-gain frequency. "PLL_LPF_Freq_Resp.dds" shows the frequency and phase responses of the low-pass filter by itself, as well as the unity gain frequency and the phase at the unity-gain frequency.
2. "PLL_Freq_Resp_opt" optimizes the filter component values to create a PLL with a user-specified unity-gain frequency and phase margin.
3. "LinearVCO" is a linear model of a VCO.
4. "LinearPFD" is a linear model of a phase/frequency detector.
5. "LinearPFD2" is a 2-input linear model of a phase/frequency detector.
6. "LinearDivider" is a linear model of a divider.


Closed-Loop Frequency Response Simulation
Optimization of loop filter parameters to create a PLL with desired open-loop unity gain frequency and phase margin.
Follow these steps:
1) Enter desired unity gain frequency and minimum phase margin.
2) Set allowable ranges for loop filter components.
3) Set optimization parameters on Nominal Optimization controller.
4) Run optimization and view results.
5) Note: it may be necessary to increase the points per decade (Dec) parameter, to ensure that the unity gain frequency is actually simulated. If it is not actually simulated, you will get an "Unable to locate any OptGoal/YieldSpec devices" error.
VCO tuning coefficient, PFD gain, and divide ratio.
Loop Filter Component Values
OPTIM
GOAL
GOAL
Open-Loop Frequency Response Simulation
Loop Filter Frequency Response Simulation
AC

Analysis

Figure 1: Open and closed loop amplitude response and unity gain frequency

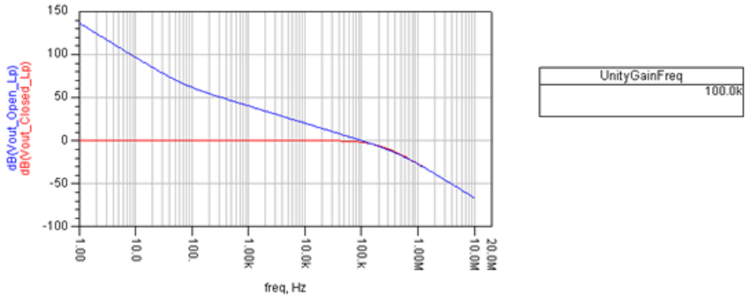
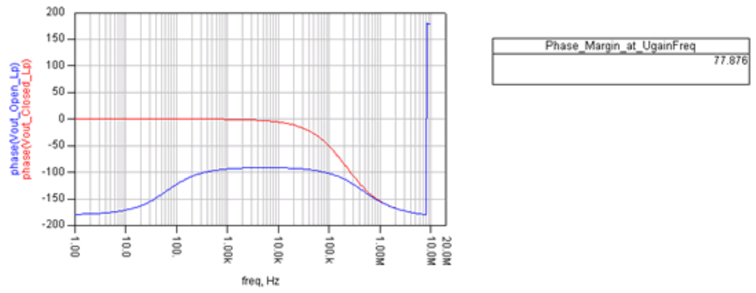


Figure 2: Open and closed loop phase response and phase margin



Optimization of A Multi-Harmonic Source and Load

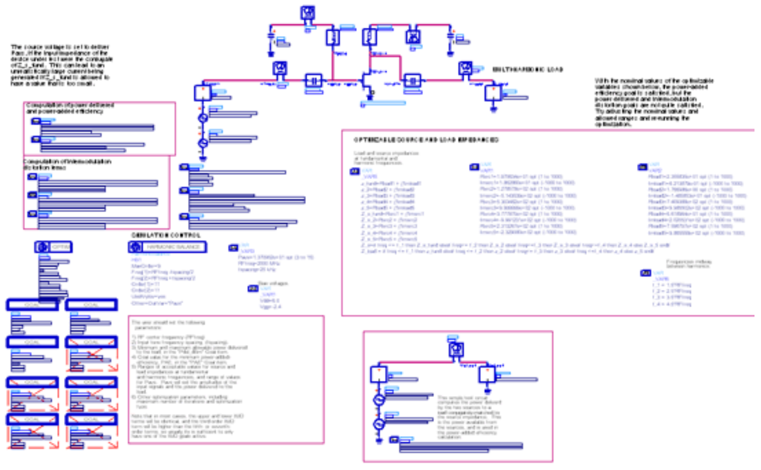
Location: $HPEESOF_DIR/examples/RF_Board/HarmonicZopt_wrk

Objective

This example shows the optimization of a MOSFET source and load impedances at fundamental and harmonics up to the 5th, to deliver a user-specified power to the load, maximize power-added efficiency, and minimize 3rd-, 5th-, and 7th-order intermodulation distortion terms.

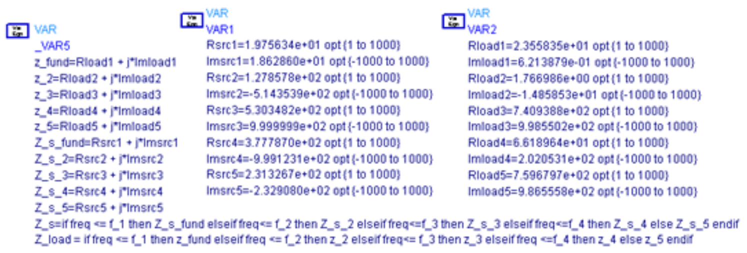
Setup

1. "FET_Ivtest" generates the I-V curves of a MOSFET.
2. "HB1toneSwp" simulates the transducer power gain, harmonic distortion, output power, and input reflection coefficient and impedance versus input power, for 50 ohm source and load impedances.
3. "HarmZopt1tone" optimizes the device source and load impedances at the fundamental and harmonics, as well as the available source power, to deliver a user-specified output power to the load, and maximize power-added efficiency.
4. "HarmZopt2tone" optimizes the device source and load impedances at the fundamental and harmonics, as well as the available source power, to deliver a user-specified output power to the load, and maximize power-added efficiency, and intermodulation distortion terms.



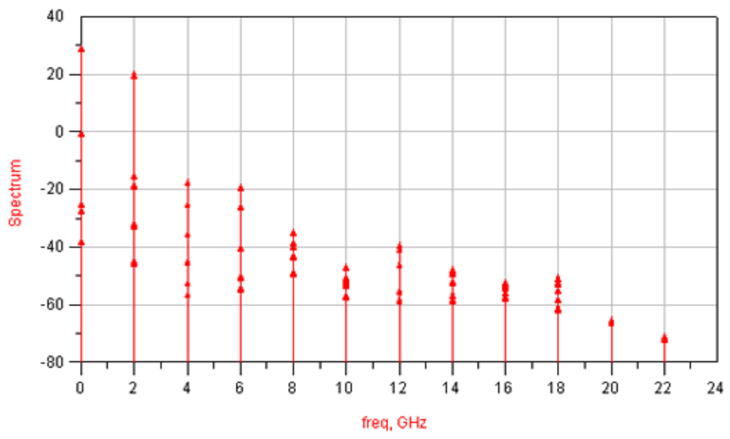
Analysis

Figure 1: Definition of source and load impedances



```
VAR                            VAR                                         VAR
_VAR5                          VAR1                                        VAR2
z_fund=Rload1 + j*Imload1      Rsrc1=1.975634e+01 opt{1 to 1000}           Rload1=2.355835e+01 opt{1 to 1000}
z_2=Rload2 + j*Imload2         Imsrc1=1.862860e+01 opt{-1000 to 1000}      Imload1=6.213879e-01 opt{-1000 to 1000}
z_3=Rload3 + j*Imload3         Rsrc2=1.278578e+02 opt{1 to 1000}           Rload2=1.766986e+00 opt{1 to 1000}
z_4=Rload4 + j*Imload4         Imsrc2=-5.143539e+02 opt{-1000 to 1000}     Imload2=-1.485853e+01 opt{-1000 to 1000}
z_5=Rload5 + j*Imload5         Rsrc3=5.303482e+02 opt{1 to 1000}           Rload3=7.409388e+02 opt{1 to 1000}
Z_s_fund=Rsrc1 + j*Imsrc1      Imsrc3=9.999999e+02 opt{-1000 to 1000}      Imload3=9.985502e+02 opt{-1000 to 1000}
Z_s_2=Rsrc2 + j*Imsrc2         Rsrc4=3.777870e+02 opt{1 to 1000}           Rload4=6.618964e+01 opt{1 to 1000}
Z_s_3=Rsrc3 + j*Imsrc3         Imsrc4=-9.991231e+02 opt{-1000 to 1000}     Imload4=2.020531e+02 opt{-1000 to 1000}
Z_s_4=Rsrc4 + j*Imsrc4         Rsrc5=2.313267e+02 opt{1 to 1000}           Rload5=7.596797e+02 opt{1 to 1000}
Z_s_5=Rsrc5 + j*Imsrc5         Imsrc5=-2.329080e+02 opt{-1000 to 1000}     Imload5=9.865558e+02 opt{-1000 to 1000}
Z_s=if freq <= f_1 then Z_s_fund elseif freq<= f_2 then Z_s_2 elseif freq<=f_3 then Z_s_3 elseif freq<=f_4 then Z_s_4 else Z_s_5 endif
Z_load = if freq <= f_1 then z_fund elseif freq <= f_2 then z_2 elseif freq<= f_3 then z_3 elseif freq <=f_4 then z_4 else z_5 endif
```

Figure 2: Full output spectrum



Notes

- Z1P component can be used effectively to realize optimizable source and load impedances.

PCS Cellular Power Amplifier Design and Analysis

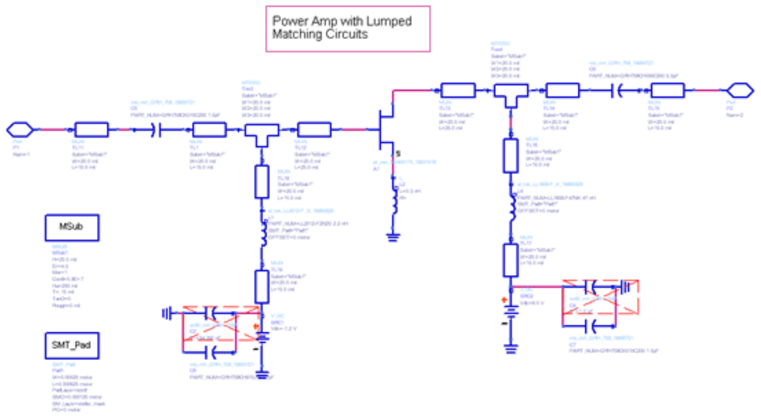
Location: $HPEESOF_DIR/examples/RF_Board/PCS_pamp_wrk

Objective

This example demonstrates a 1.9GHz PCS power amplifier design using lumped and distributed matching circuits. It features matching circuit design, optimization, and autolayout.

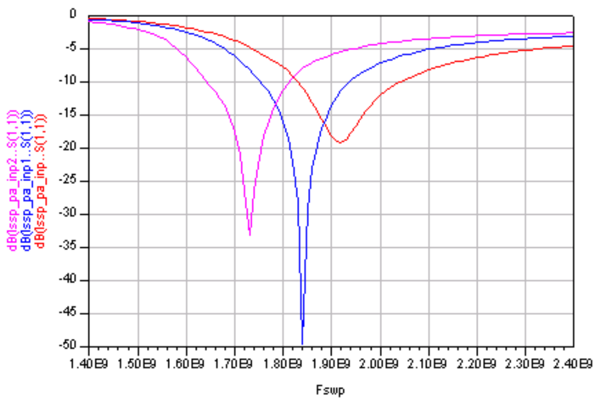
Setup

1. "lssp_sim" performs a large signal S-parameter of all of the variations in the power amplifier design sequence. The evolution of the input matching circuit is contained in "pa_inp" (basicparts), "pa_inp1" (adding SMT parts) and "pa_inp2" (adding SMT pads).
2. "sys_sim" is a system level analysis using subckt "sys_block".
3. "match_sim" compares lumped vs. distributed input matching circuits using designs "match_lump" and "match_distr".
4. "perf_opt" optimizes the input and output matching circuits using "pa_opt".
5. The final designs are "pa_final", "pa_distr", and "pa_tee".
6. "CDMA_sim" tests the PA using an IS-95 reverse link modulated signal and calculates the Adjacent Channel Power Ratios (ACPR), spectrum and plots the trajectory diagram.



Analysis

Figure 1: Input matching simulation results



Highest resonance is with ideal components, middle resonance is with parts from a surface mount library, lowest resonance is with surface mount library and models for their pads included.

Figure 2: Intermediate results of gain and input match optimization

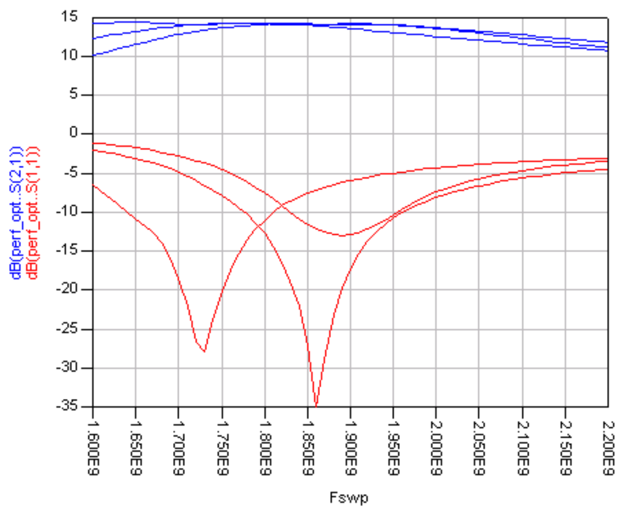


Figure 3: Amplifier layout using distributed matching network elements

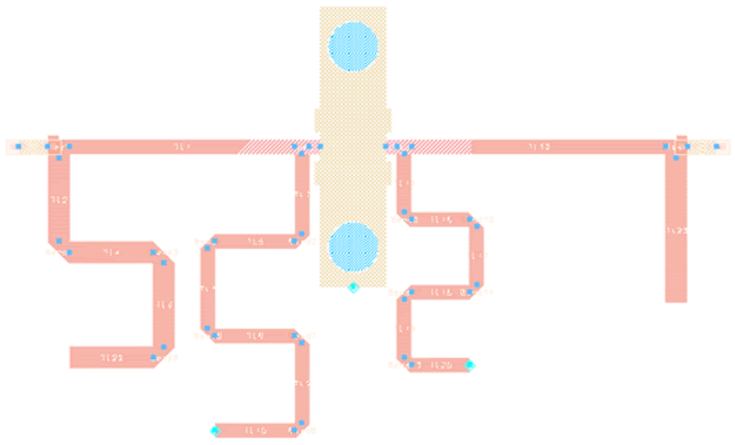
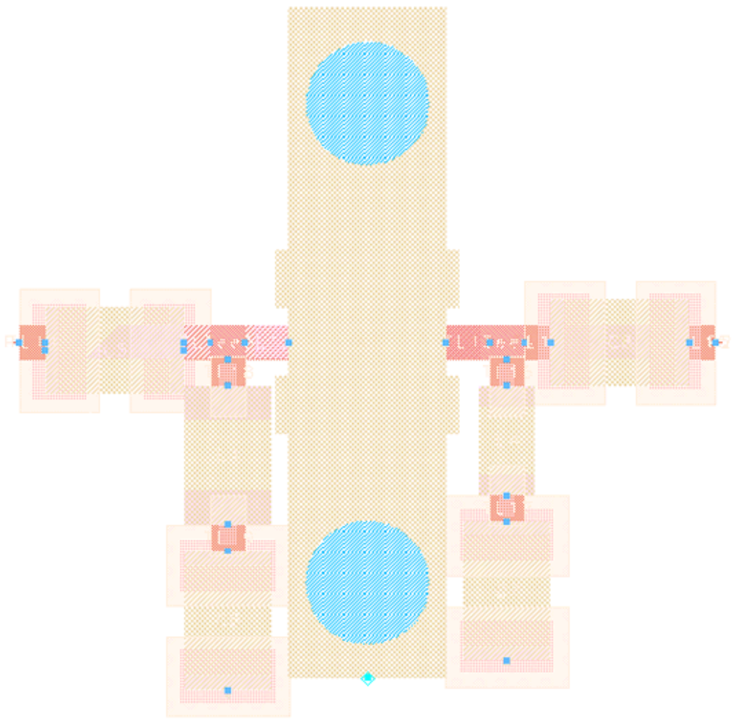


Figure 4: Amplifier layout using lumped (surface-mount) matching network elements



Notes

- The layout for each circuit was created automatically and can be regenerated by the user. Initiate the autolayout function from the schematic window via Layout > Generate/Update Layout. In the dialog box, set the Starting Component = P1. The resulting layout will be displayed.
- To view the subcircuits, activate the component in the schematic, then click on the Down Arrow icon. To return to the high level design, click on the Up Arrow icon.
- Simulation controllers used: LSSP and optimization, S-parameter, Harmonic Balance, Circuit Envelope.

Phase Noise Simulation

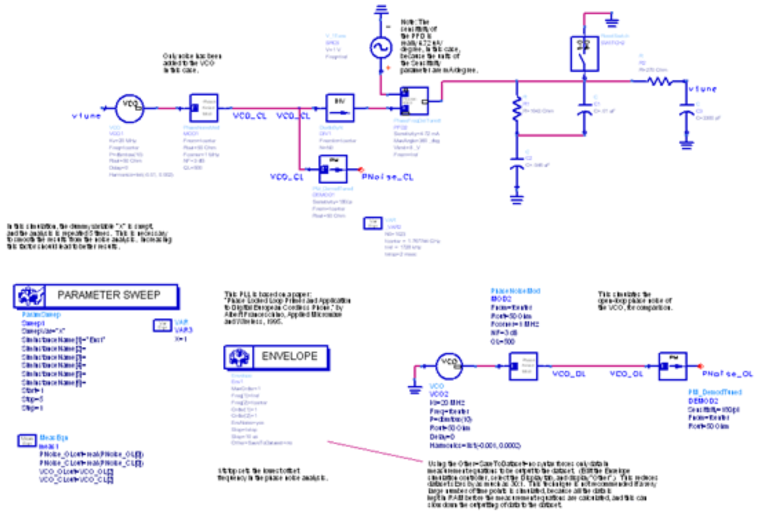
Location: $HPEESOF_DIR/examples/RF_Board/PLL_Examples/PLL_PhaseNoise1_wrk

Objective

This example shows ways of simulating phase-locked loop (PLL) phase noise, modeling the noise as a time-domain or frequency-domain signal. The time-domain method would be useful when simulating a PLL with modulation, where the noise effects on the modulation must be simulated in the time domain.

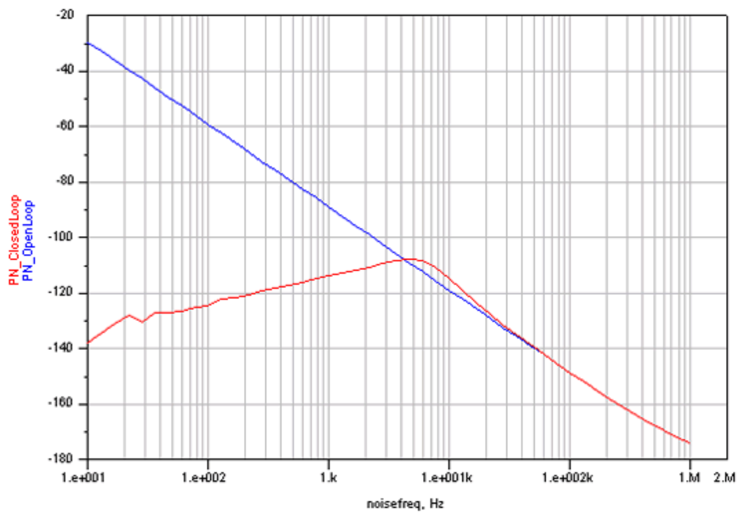
Setup

1. "PNSrc_wPMdemENV" simulates adding phase noise to a source using the PhaseNoiseMod component. A phase modulation (PM) demodulator component is used to demodulate the phase of the signal after noise has been added. The noise is simulated in the frequency domain.
2. "PhaseNoiseCalcs.dds" is a "calculator" that shows how the various parameter settings on the PhaseNoiseMod component affect the single-sideband phase noise that is generated. This approximately predicts the noise that will be generated by a Circuit Envelope simulation.
3. "PNSrc_wPMdemod" simulates adding phase noise to a source using the "PhaseNoiseMod" component, using harmonic balance. This method cannot be used when simulating PLLs.
4. "LockTest" is a simple test of a PLL to confirm that it is locked and that all transient responses have died out. Before starting PLL noise simulations, it is good to verify that the loop is in a quiescent state.
5. "PLL_PhNoise" simulates the phase noise of a PLL (from the /examples/RF_Board/DECT_LO_Synth_wrk), with noise added to the VCO. The noise is simulated as a time-domain signal.
6. "PLL_PhNoiseFreqDom" is similar to "PLL_PhNoise", except that noise is simulated in the frequency domain. This simulation is much faster.
7. "PLL_ImpulseResp" simulates the impulse response of the PLL, from which the closed loop frequency response is derived.



Analysis

Figure 1: Open and closed loop VCO phase noise



Notes

- Note that noise in PLLs can be simulated most quickly using small-signal, linear models of the PLL components (see also the example file: PLL_PhaseNoise2_wrk).
- Reference paper: "Phase Locked Loop Primer and Application to Digital European Cordless Phone," by Albert Franceschino, Applied Microwave and Wireless, 1995.

Phase Noise Simulations using Small Signal Model Loop Components

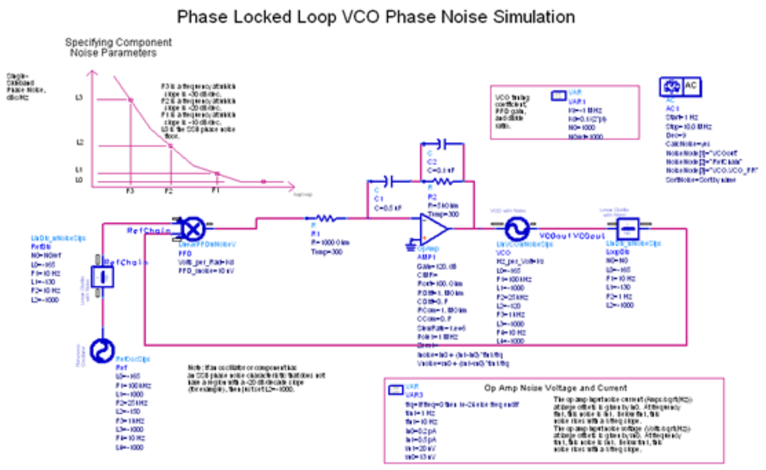
Location: $HPEESOF_DIR/examples/RF_Board/PLL_Examples/PLL_PhaseNoise2_wrk

Objective

This example shows phase-locked loop (PLL) phase noise simulation. It uses linear, small-signal models for each of the loop components. The contribution of each noise source to the total voltage-controlled oscillator (VCO) phase noise is computed and plotted. This way of simulating PLL phase noise is very fast, requiring only several seconds.

Setup

1. "PLL_Noise_Contrib" shows a simple PLL with small-signal models for each of the components. Phase noise is modeled as voltage or current noise sources within each component. The user must specify offset frequency-single-sideband phase noise pairs for the dividers and reference and VCO sources. Straight-line interpolation on a dB versus log of frequency plot is used.
2. "PLL_Noise_Contrib2" shows the same PLL with noise added to each component slightly differently. In this case, noise slopes for each element are added together to generate the composite noise characteristic.
3. "PLL_Noise_Contrib3" is similar to "PLL_Noise_Contrib2", except that it uses a loop filter with an operational amplifier. The noise of the operational amplifier is also modeled.
4. "RefOscTest" compares the phase noise of the two reference oscillator models. They are nearly identical.
5. "LinDiv_wNoise" is a linear divider model with phase noise modeled as a noise voltage source at the output. The noise is generated from pairs of frequency points and phase noise data.
6. "LinDiv_wNoiseSlps" is the same as "LinDiv_wNoise" except that the phase noise data is entered as points on the phase noise versus offset frequency curve, where the slopes are at a particular level.
7. "LinVCOwNoise", "LinVCOwNoiseSlps", "RefOsc", and "RefOscSlps" are all similar, with the two different ways of defining oscillator phase noise.
8. "LinearPFDwNoise" is a linear phase-frequency detector model, with a current source output.
9. "LinearPFDwNoiseV" is a linear phase-frequency detector model, with a voltage source output.


Phase Locked Loop VCO Phase Noise Simulation
Specifying Component Noise Parameters
Op Amp Noise Voltage and Current

Analysis

Figure 1: Contributions to VCO noise from different components

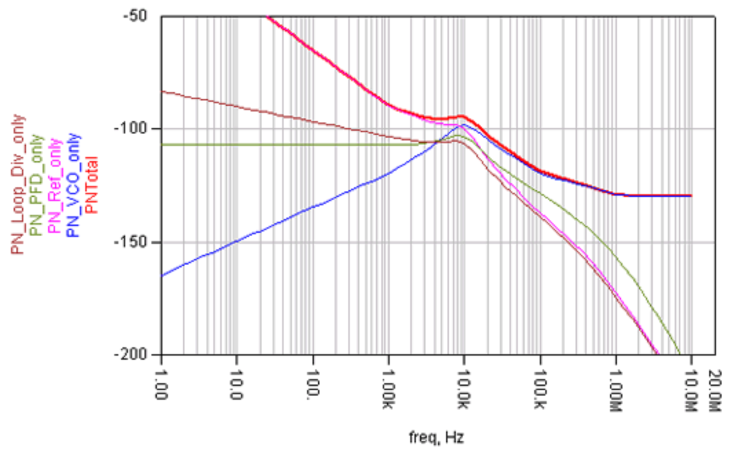
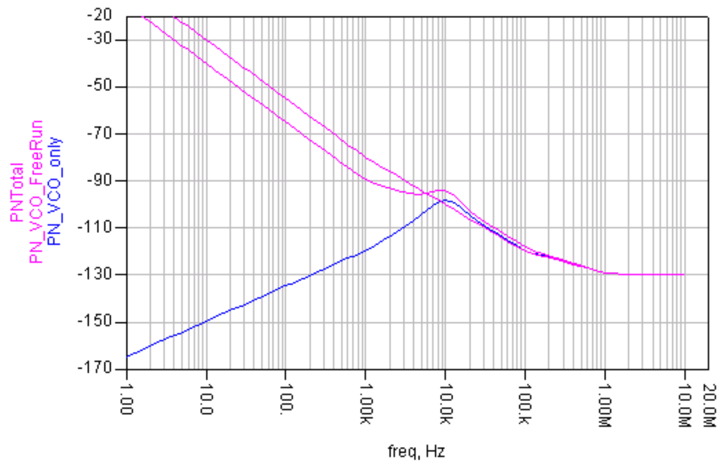


Figure 2: Free-run phase noise and closed loop phase noise and VCO contribution to noise



PLL Simulation of DECT Radio System

Location: $HPEESOF_DIR/examples/RF_Board/PLL_Examples/DECT_LO_Synth_wrk

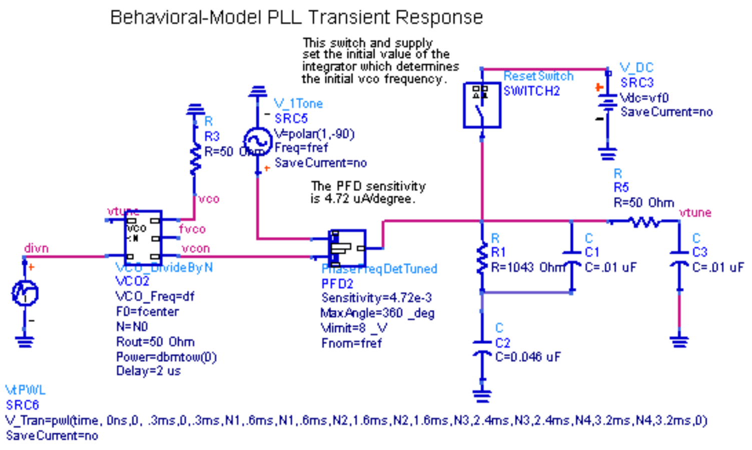
Objective

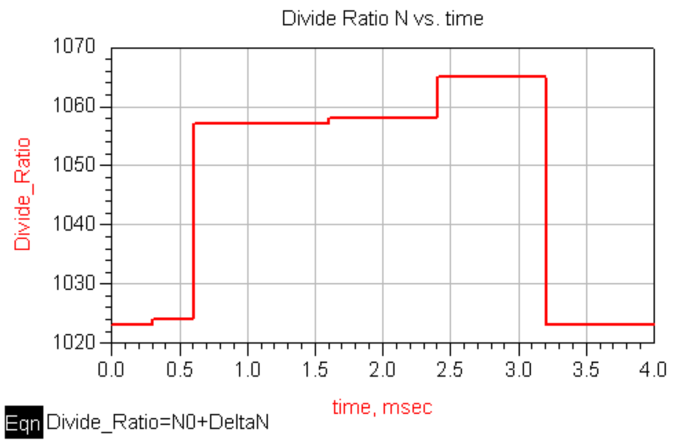
This example provides three simulations on DECT radio PLL that is originally described in a paper, "Phase Locked Loop Primer and Application to a Digital European Cordless Phone" by Albert Franceschino, Applied Microwave and Wireless, Fall, 1994.

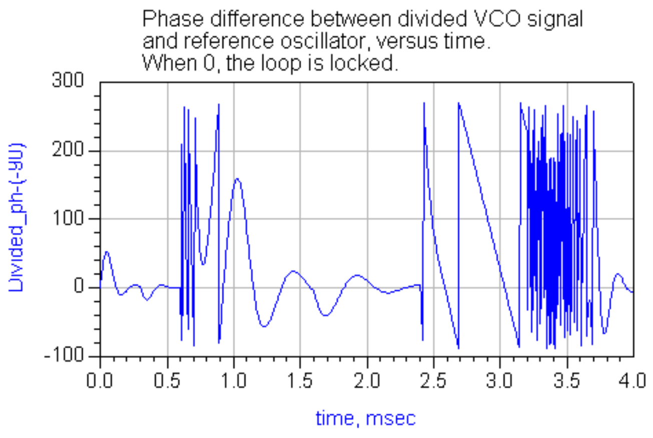
Setup

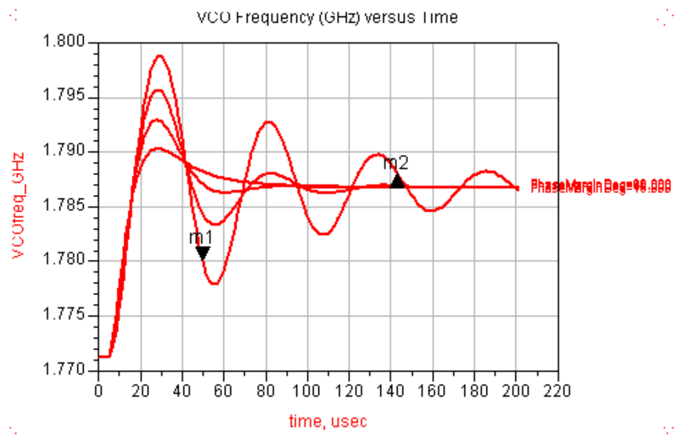
1. VCO_Switching_divN simulates the transient response of the PLL, as the divide ratio is varied as a function of time.
2. PLL_Tran_SweptPhMargin simulates the transient response of the PLL with a modified unity-gain frequency (20 kHz), as a function of desired phase margin. It uses the tuned phase/frequency detector, and the loop filter component values are defined as a function of the the unity gain frequency and phase margin. The simulation shows that the step response has increasing "ringing" as the phase margin is reduced.
3. PLL_Tran_wBBPFD simulates the same PLL, except using a base-band phase/frequency detector. To save a little disk space, the simulation results from this one are not saved.

Analysis









Power Amplifier Layout and Design Rule Checker

Location: $HPEESOF_DIR/examples/RF_Board/cellular_pamp_wrk

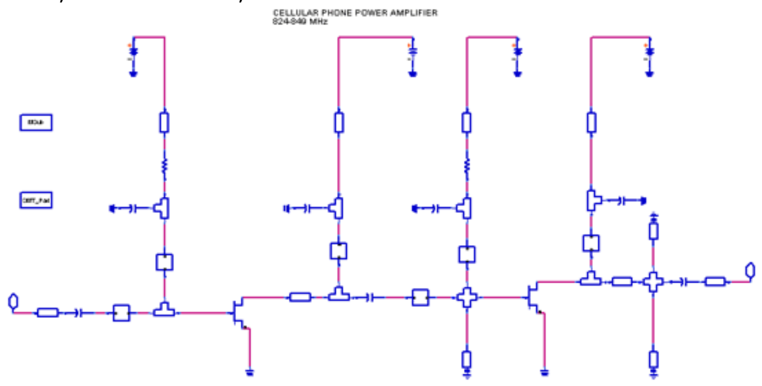
Objective

This example demonstrates an 800 MHz cellular power amplifier using distributed matching circuits. The design is not simulatable but instead is used for demonstrating layout tools such as autolayout and the Design Rule Checker (DRC).

Setup

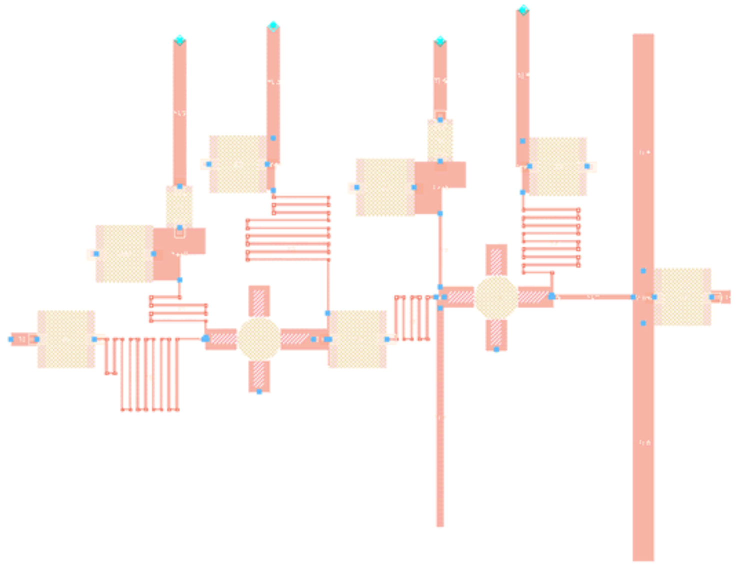
1. The layout for this circuit was created automatically and can be regenerated by the user. Initiate the autolayout function from the schematic window via Layout > Generate/Update Layout. In the dialog box, set the Starting Component = P1. The resulting layout will be displayed.
2. The DRC can be demonstrated on the final layout to inspect for process variations such as minimum line width. Several example process rule files are included (PC5, PC6, PC8, PC10, and PC12) to demonstrate the custom DRC capability. Access the custom DRC from the layout window via Verify > Custom DRC. To select from one of the process rule sets, select workspace, the 5 process rules should be displayed. Select one of them, then Apply. Sequence through Select Rules, Run DRC, Load Result, and View Errors. When viewing errors, enable the Auto Select and Auto Zoom

features, then select First, then Next to scroll through each DRC rule failure. When done, select Clear All, then Cancel.



Analysis

Figure 1: PA Layout generated automatically from the schematic



Notes

- In the Layout Window, choose File > Generate Artwork before running DRC to flatten the design. Wait for the status window to show that the DRC has completed for one rule before proceeding to next step. Select Clear All before moving to the next rule.
- It is also possible to use the Interactive DRC to check the minimum width. Access the interactive DRC from the layout window via Verify > DRC. Disable Minimum Spacing and Minimum Angle, enter a minimum trace width, select the cond layer, then Apply. Sequence through Select Rules, Run DRC, Load Result, and View Errors. When viewing errors, enable the Auto Select and Auto Zoom features, then select First, then Next to scroll through each DRC rule failure. When done, select Clear All, then Cancel.
- The Ground Plane pouring (or trace clearance) feature can be viewed by accessing the flat_art design file. Since the pamp design has subnetworks, the final layout artwork must be flattened first before adding the ground plane pour. This can be done by choosing File > Generate Artwork.

Power Amplifier using Cartesian Feedback

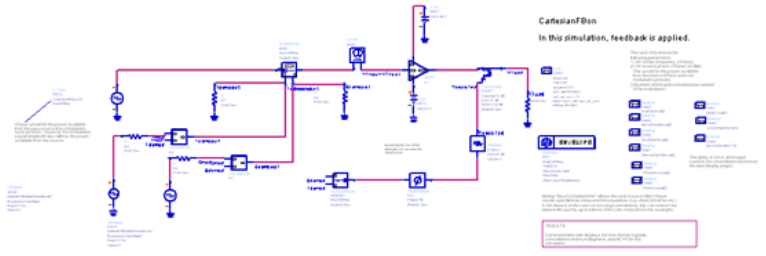
Location: $HPEESOF_DIR/examples/RF_Board/CartesianFB_wrk

Objective

This example shows the use of Cartesian feedback in a power amplifier design. Simulation with a NADC pi/4 DQPSK input signal shows improvement in the Adjacent-Channel Power Ratios (ACPR) by using the feedback.

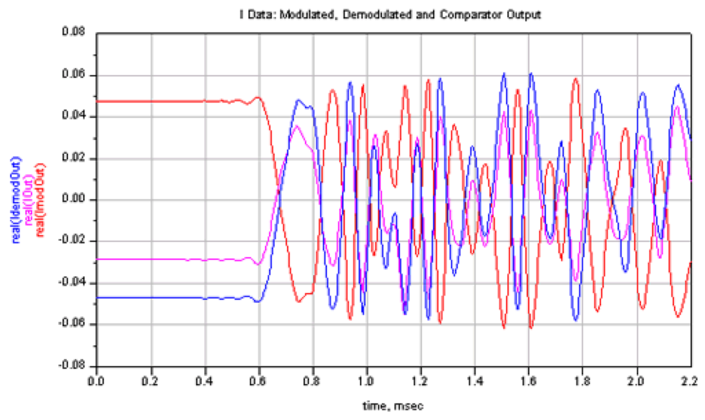
Setup

1. "BbdataGeneration" generates the NADC pi/4 DQPSK signal.
2. "ComparatorCkt" contains the comparator/filter circuit. "OpAmpCkt" contains the ideal opamp and limiter that are used in "ComparatorCkt". "ComparatorTest" is a simple setup to verify that the comparator/filter is working properly.
3. "TestPA" is the power amplifier circuit. "Motorola_MOSFET_model" is the device model used in the PA.
4. "CartesianFBoff" and "CartesianFBon" simulate the output power and ACPR of the PA with/without the feedback applied, respectively.
5. "sweptCartesianFBon" and "sweptCartesianFBoff" analyze the performance of the same circuit with and without feedback, with the swept RF power level.



Analysis

Figure 1: I Data: modulated, demodulated and comparator output. With feedback (loop closed), the demodulated I and Q signals are equal in magnitude, but opposite in phase to the modulated I and Q signals.



Notes

- Cartesian feedback amplifiers are used to generate high output power signals with good ACPR. This is accomplished by coupling off part of the demodulated signal to pre-distort the input baseband I and Q signals via a comparator/filter circuit.
- There is about a 1 dB improvement in ACPR for the same output level. Circuit parameters can be adjusted to achieve better performance.
- To get loop stability in the Cartesian Feedback implementations, the comparator/filter circuit uses a low-pass filter to limit loop bandwidth. The cutoff frequency must be sufficiently wider than the bandwidth spread due to the amplifier nonlinearity. This limits the upper modulation bandwidth, but is sufficient for mobile radio applications.

Power Amplifier with an IS-95 CDMA Source Input

Location: $HPEESOF_DIR/examples/RF_Board/ACPRopt_wrk

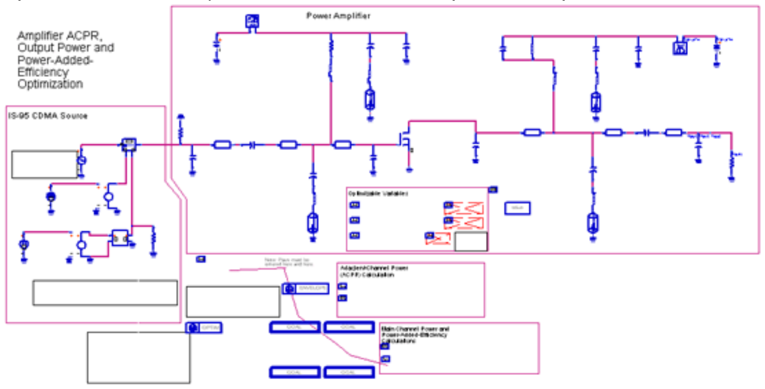
Objective

This example shows how to optimize a power amplifier's adjacent-channel power ratio (ACPR), power-added efficiency (PAE) and output power with an IS-95 Code Division Multiple Access (CDMA) source input signal.

Setup

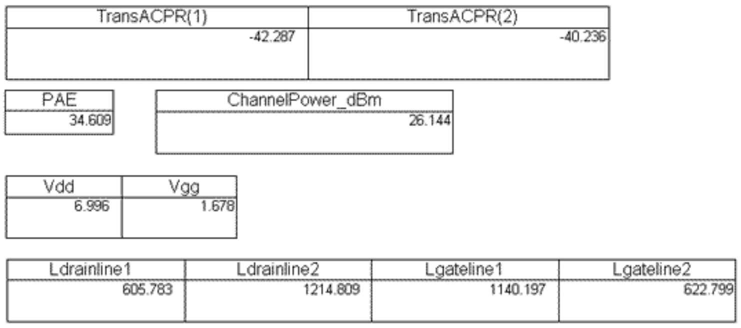
1. "Motorola_Mosfet_Model" incorporates the effect of bond-wire inductance and bonding-pad capacitance in the Motorola MOSFET Root model.
2. "PseudoRandImpulse" create a pseudo-random impulse source, which is required because the CDMA filter response is defined as an impulse response.

3. "CDMA_ACPR_opt" uses a Circuit Envelope Simulator and Nominal Optimization to optimize the ACPR, PAE and Pout of the power amplifier.


Amplifier ACPR, Output Power and Power-Added-Efficiency Optimization
Power Amplifier
IS-95 CDMA Source
Optimizable Variables
Adjacent Channel Power (ACPR) Calculation
Main Channel Power and Power-Added-Efficiency Calculations

Analysis

Figure 1: Optimized Parameters



TransACPR(1)	TransACPR(2)
-42.287	-40.236

PAE
34.609

ChannelPower_dBm
26.144

Vdd	Vgg
6.996	1.678

Ldrainline1	Ldrainline2	Lgateline1	Lgateline2
605.783	1214.809	1140.197	622.799

Notes

- Note that because the baseband filter finite impule response (FIR) are given as an impulse response, the signal driving the filter must be a series of pseudorandom impulses, or the filter's frequency response will be incorrect.
- The VCVS_Z sampled-data voltage-controlled voltage sources implement the FIR filter in the IS-95 CDMA specification. The filter coefficients can be found in Table 7.1.3.1.10.1-1 of that specification. The list of coefficients can be seen here by editing the VCVS_Z. Note that the ACPR of the signal generated here may be too high for amplifier testing purposes. A better signal can be generated by using a different set of FIR filter coefficients.
- It is better not to save the solutions of each iteration, to save memory. Instead, at the end of the simulation, select Simulate/Update Optimization Values, deactivate the Nominal Optimization controller and run a simulation to see the results with the best parameter values.
- The samples per bit of the modulation signal should be kept at 4, otherwise the baseband filter response will be incorrect.

Simulation of An Automatic Gain Control Loop

Location: $HPEESOF_DIR/examples/RF_Board/AGC_wDownConv_wrk

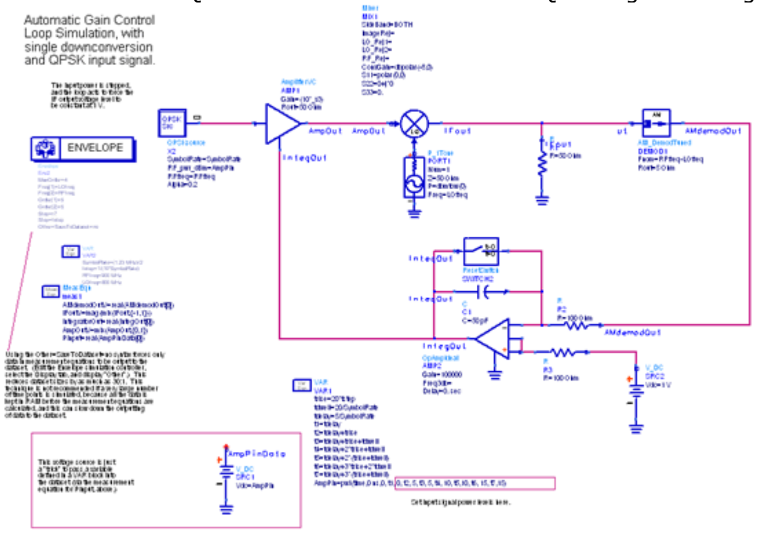
Objective

This example shows simulations of a basic automatic gain control (AGC) loop. The AGC loop is based on an ideal voltage controlled amplifier, either an AM demodulator or a simple diode can be used at the amplifier output, and its output is compared to a reference voltage. Idealized op-amp is used as the integrator and to produce the control voltage. AGC with a Quadrature Phase-Shift Keyed (QPSK) source is also tested.

Setup

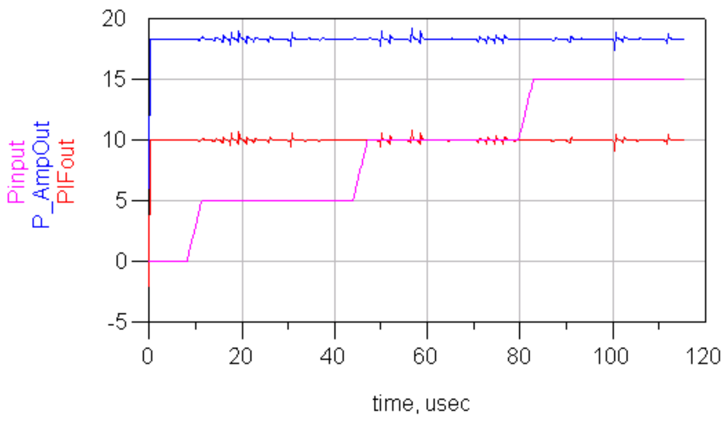
1. "B_VCAtest" shows the simulation of a voltage-controlled amplifier, which is the basic element for the following AGC loop.
2. "C_AGCtest" shows the transient response of a simple AGC loop where the reference voltage at the integrator input is stepped. "D_PinChange" is another similar test with the input signal power stepped instead.

3. "F_wDetector" simulates the AGC loop with the detector replacing the AM demodulator, with stepped reference voltage. "Detector" shows the simple diode. "E_DetectorTest" simulates its output voltage versus input power.
4. "G_wDownConvOL" and "H_wDownConv" add a stage of downconversion (an ideal mixer) between the amplifier and detector for open and closed loop respectively. "I_wDownConvOL" and "J_wDownConvOpAmp0" use an AM instead of the diode detector.
5. "L_wQPSKsource" uses an QPSK source. " M_wQPSKsrc_wDet" is similar and uses the diode detector. "QPSKsourceTest" simulates the QPSK signal driving a resistor.


Automatic Gain Control Loop Simulation, with single downconversion and QPSK input signal.
ENVELOPE
AmpOut
IFout
IntegOut
AMdemodOut

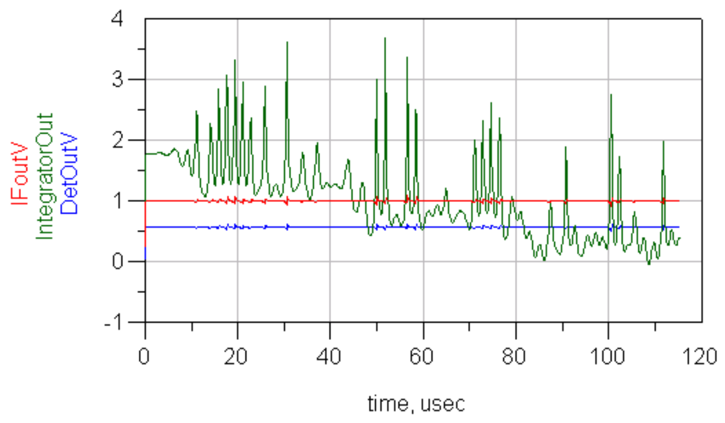
Analysis

Figure 1: Amplifier, IF output power, and stepped input power



The amplifier and IF output powers change in response to variations in input power due to the QPSK modulation, but the loop forces them back to their target levels.

Figure 2: IF, detector, and integrator output voltage



The integrator output voltage steps down in response to increases in the input power, to force the IF output voltage to be constant.

Notes

- Note that in "K_FreqResp" the switch in the integrator feedback path has been replaced by a large resistor. Otherwise the feedback will be shorted out (for harmonic balance simulations.)

TDR Analysis of Board Level Crosstalk

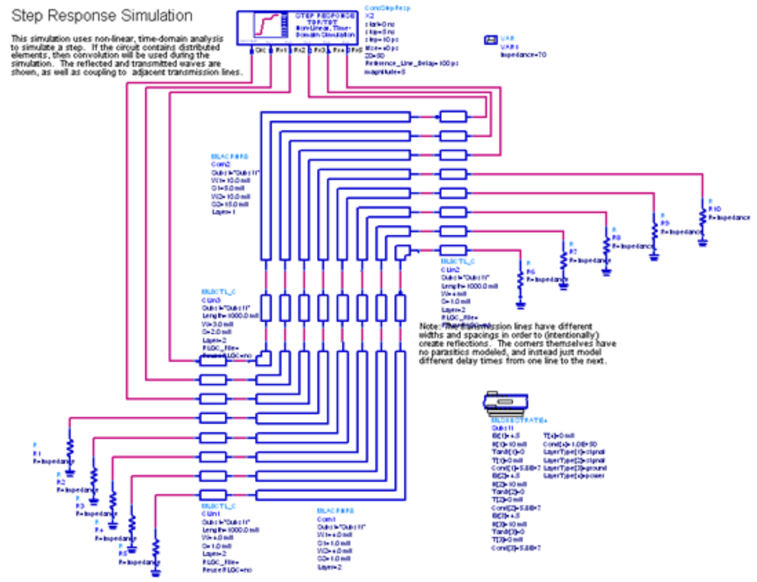
Location: $HPEESOF_DIR/examples/RF_Board/TDRcrosstalk_wrk

Objective

This workspace shows several time-domain step and pulse response simulations. These simulations illustrate how signals are coupled from one transmission line to another, as well as how much a pulse waveform is degraded due to impedance mismatches. These set-ups also model time-domain reflectometry (TDR).

Setup

1. "LinearStepResp_Simple" is just a simulation of two transmission lines of different impedances, which cause reflections and distortion in the transmitted waveform.
2. "LinearStepResp_test" does a swept-frequency simulation to model the step response of an 8-coupled transmission line structure with two bends and different impedances to generate mismatches. The frequency-domain data is converted to the time-domain in the data display, using the ts() function.
3. "LinearPulseResp_test" uses the LinearPulseResp instrument, which does a swept-frequency simulation to model the pulse response of the same transmission line structure.
4. "ConvStepResp_test" does the same simulation as LinearStep_Resp_test, except that it uses the Transient simulator, which will use convolution if distributed elements are included on the schematic.
5. "ConvPulseResp_test" does the same simulation as LinearPulseResp_test, except that it uses the Transient simulator, which will use convolution if distributed elements are included on the schematic.



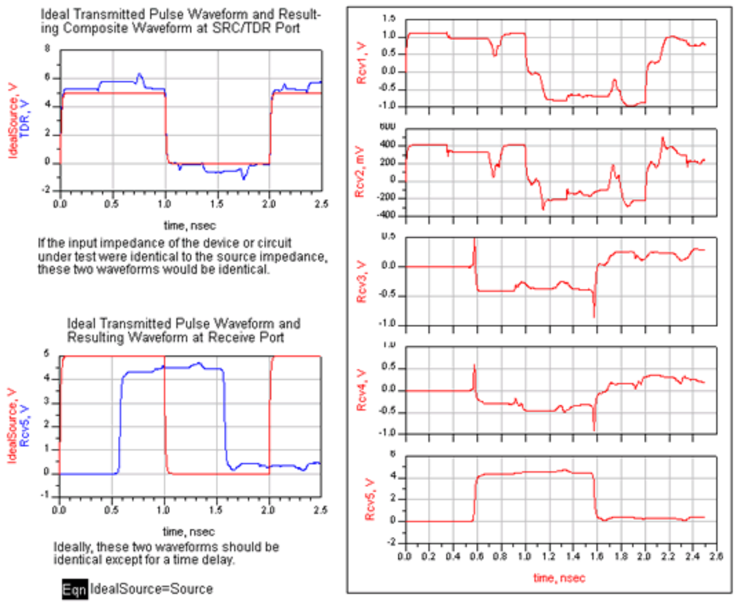
Step Response Simulation

This simulation uses non-linear, time-domain analysis to simulate a step. If the circuit contains distributed elements, then convolution will be used during the simulation. The reflected and transmitted waves are shown, as well as coupling to adjacent transmission lines.

Note: The transmission lines have different widths and spacings in order to (intentionally) create reflections. The corners themselves have no parasitics modeled, and instead just model different delay times from one line to the next.

Analysis

Figure 1: Result from a transient analysis of an 8-coupled transmission line structure



If the input impedance of the device or circuit under test were identical to the source impedance, these two waveforms would be identical.

Ideally, these two waveforms should be identical except for a time delay.

Eqn IdealSource=Source

TDR and S-parameter Simulations of Microstrip Step Discontinuities

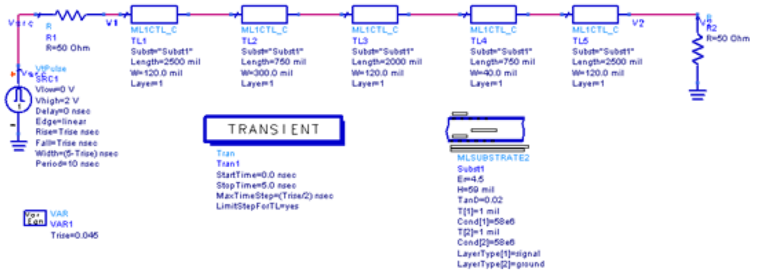
Location: $HPEESOF_DIR/examples/RF_Board/TDRmeas_vs_model_wrk

Objective

This workspace compares measured and simulated results for a microstrip step discontinuities. Both time domain reflectometry (TDR) and S-parameter simulations are shown.

Setup

1. "TDR_transient" is used to simulate the TDR response. The TDR measurements were made using the HP54120 and normalizing the step source to a 45 picosecond rise time.
2. SP_freqsweep is used to calculate the frequency response. The response is also transformed into time domain.



R
R1
R=50 Ohm

ML1CTL_C
TL1
Subst="Subst1"
Length=2500 mil
W=120.0 mil
Layer=1

ML1CTL_C
TL2
Subst="Subst1"
Length=750 mil
W=300.0 mil
Layer=1

ML1CTL_C
TL3
Subst="Subst1"
Length=2000 mil
W=120.0 mil
Layer=1

ML1CTL_C
TL4
Subst="Subst1"
Length=750 mil
W=40.0 mil
Layer=1

ML1CTL_C
TL5
Subst="Subst1"
Length=2500 mil
W=120.0 mil
Layer=1

R
R2
R=50 Ohm

VtPulse
SRC1
Vlow=0 V
Vhigh=2 V
Delay=0 nsec
Edge=linear
Rise=Trise nsec
Fall=Trise nsec
Width=(5-Trise) nsec
Period=10 nsec

Tran
Tran1
StartTime=0.0 nsec
StopTime=5.0 nsec
MaxTimeStep=(Trise/2) nsec
LimitStepForTL=yes

MLSUBSTRATE2
Subst1
Er=4.5
H=59 mil
TanD=0.02
T[1]=1 mil
Cond[1]=58e6
T[2]=1 mil
Cond[2]=58e6
LayerType[1]=signal
LayerType[2]=ground

VAR
VAR1
Trise=0.045

Analysis

Figure 1: Comparison of TDR responses

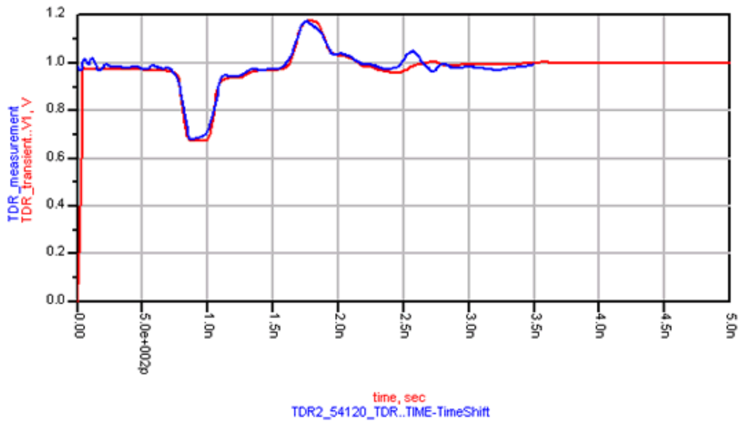
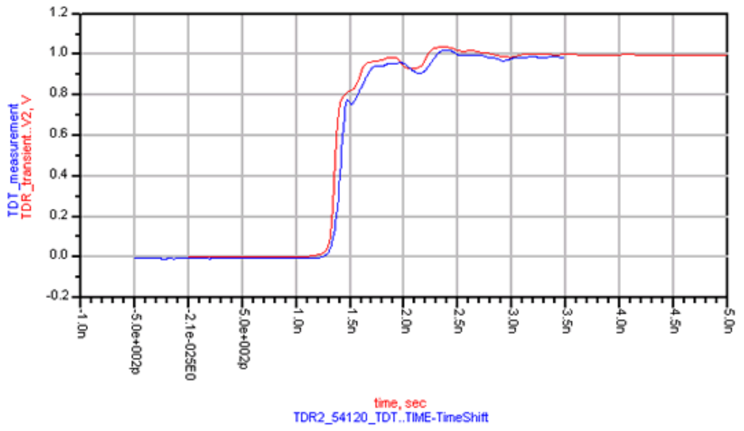


Figure 2: comparison of TDT responses



Notes

- Simulation controllers used: S-Parameters, Transient.
- The test microstrip is fabricated on 59mil FR-4 material.

RFIC Examples

Examples of how to solve problems in the time, frequency, and modulation domains to design high-performance, low-cost RF integrated circuits.

- *Analog-to-Digital Converter, Track-and-Hold* (examples)
- *CMOS VCO Examples* (examples)
- *IQ Modulator Cosimulation* (examples)
- *RFIC Oscillator Simulations* (examples)
- *Simulation of a Differential-Mode Mixer* (examples)
- *Simulation of A Downconverter with FSK Input Signal* (examples)
- *Using Graphic Cell Compiler to Create A Parameterized FET Layout* (examples)
- *Various Simulations of a Gilbert Cell Mixer* (examples)
- *Various Simulations of A Power Amplifier* (examples)

Analog-to-Digital Converter, Track-and-Hold

Location: $HPEESOF_DIR/examples/RFIC/TrackAndHold_wrk

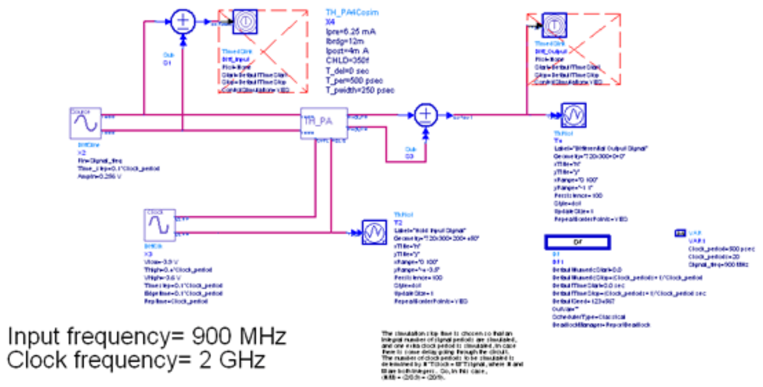
Objective

This example contains many designs and set-ups used to simulate a track-and-hold circuit, which is the front end of an analog-to-digital converter. The circuit uses a Complementary Heterojunction Bipolar Transistor (CHBT) process.

Setup

The highest level designs are "MS_TH" (Master-Slave Track and Hold) and "THCoSim". Below are descriptions of some of the simulations.

1. "MS_TH" is a cosimulation (using the Agilent Ptolemy and transient simulation engines) of a master track-and-hold circuit ("TH_PA4Cosim") driving a slave track-and-hold circuit (Slave_TH4Cosim). The input signal is a sine wave at 500 MHz, and the clock is at 2 GHz.
2. "THCoSim" is a cosimulation of the master track-and-hold circuit by itself, with an input sine wave at 900 MHz and a 2 GHz clock frequency. "THCoSim2" is similar to "THCoSim", except that it is set up to export data to Matlab for post-processing.
3. "TH_PA_SineTest" is a transient simulation of the master track-and-hold circuit only. The input sine wave is at 900 MHz and the clock frequency is at 3 GHz.
4. "TH_PA4DCTest" is an AC frequency response and a DC transfer function simulation of the master track-and-hold circuit. "TrackAndHold" is an AC frequency response and a DC transfer function simulation of a master track-and-hold circuit, with a different pre-amplifier and a different post-amplifier.
5. "TH_SFDR" is a spurious-free dynamic range simulation of the master track-and-hold circuit simulated in "TrackAndHold". The input signal is a sine wave at 15.625 MHz, and a 2 GHz clock is used. "TH_SFDR2" is the same as TH_SFDR, except that the input signal is a sine wave at 100 MHz. "TH_SFDR3" is the same as "TH_SFDR", except that the input signal is a sine wave at 15 MHz.



Analysis

Figure 1: Output at master track-and-hold ckt compared with input

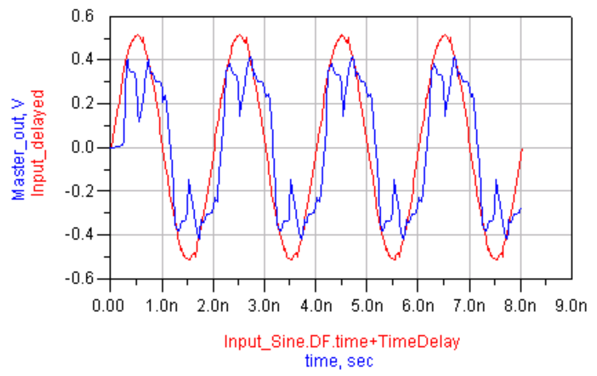
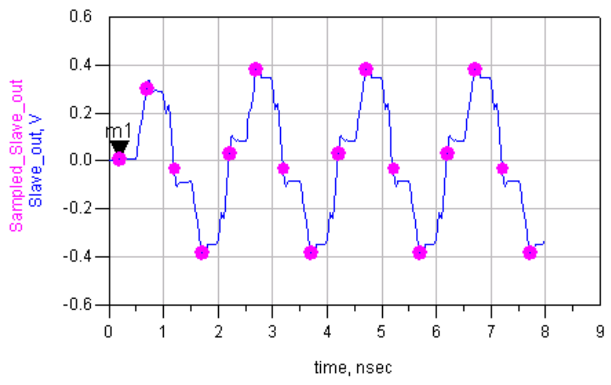


Figure 2: Slave output and sampling points



Notes

- References: Kevin Nary, "An Integrated Design Methodology for High Performance Analog to Digital Converters", 1998 DesignCon.

CMOS VCO Examples

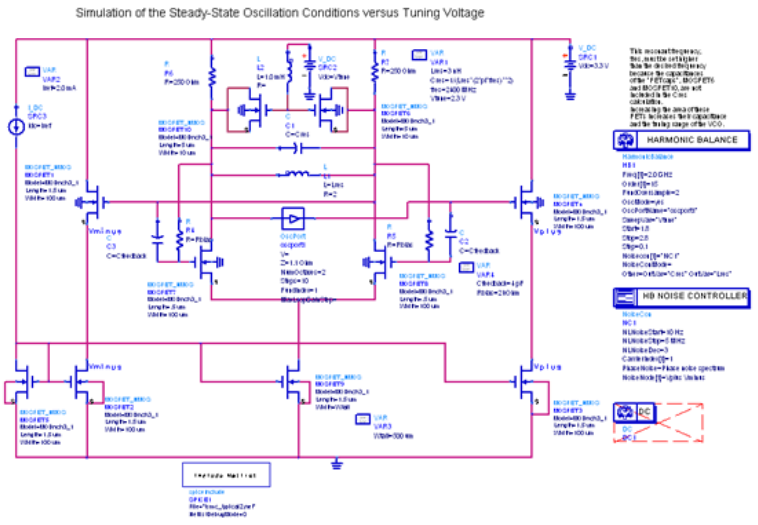
Location: $HPEESOF_DIR/examples/RFIC/MOS_VCO_wrk

Objective

This is a MOS VCO simulation and includes several intermediate steps.

Setup

1. "FET_curve_tracer" simulates the I-V curves of one of the MOSFETs. The transconductance versus bias is shown as well.
2. "diff_mode_gain" simulates the differential-mode gain of a source-coupled pair. Various parameters on the schematic can be swept and their effect on the gain can be seen.
3. "Biased_FET_Cap_Test" shows the capacitance of a FET configured as a voltage-variable capacitor versus bias.
4. "Varactor_Test" shows the capacitance of a varactor as a function of bias. "varactor" is the varactor diode model. Varactor_test shows the frequency response of the resonator circuit, including the varactors, as a function of bias voltage.
5. "Resonator_test" shows the frequency response of the resonator circuit, including varactor diodes, as a function of bias voltage.
6. "VCO_OscTest_wFETcaps" simulates the small-signal oscillation conditions versus tuning voltage (or any other parameter) when FETs are used as the tuning capacitors.
7. "VCO_wFETcaps" shows the large-signal steady-state solutions versus tuning voltage.
8. "VCO_OscTest_wVaractors" simulates the small-signal oscillation conditions versus tuning voltage (or any other parameter) when varactors are used as the tuning capacitors.
9. "VCO_wVaractors" shows the large-signal steady-state solutions versus tuning voltage.
10. "VCO_EnvTest_wFM" applies frequency modulation to the open-loop VCO.


Simulation of the Steady-State Oscillation Conditions versus Tuning Voltage
HARMONIC BALANCE
HB NOISE CONTROLLER
DC

Analysis

Figure 1: VCO output spectrum

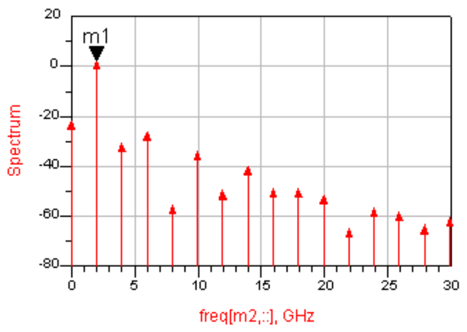
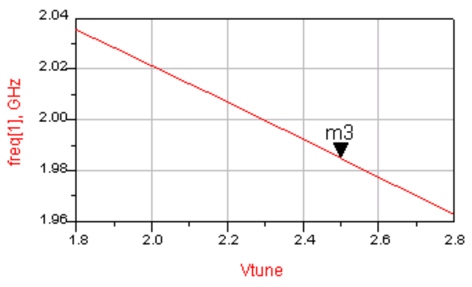


Figure 2: VCO fundamental frequency versus tuning range



IQ Modulator Cosimulation

Location: $HPEESOF_DIR/examples/RFIC/Cosim_lab_wrk

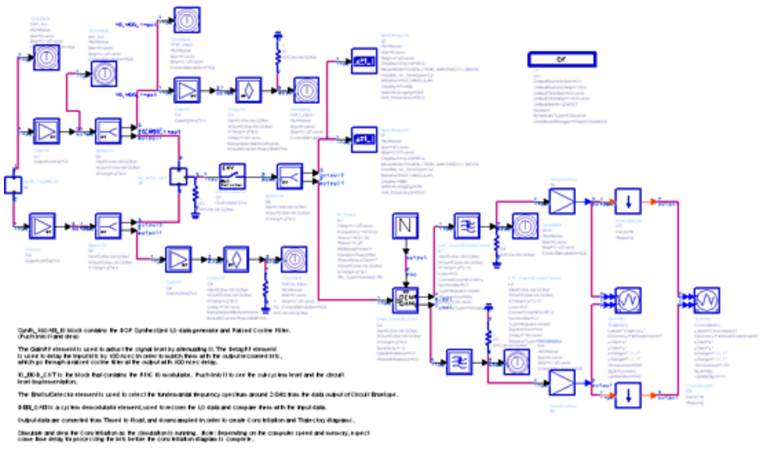
Objective

This workspace contains a thorough DSP, Analog-Mixed-Signal, and RFIC Co-Simulation example. It presents the power of the Advanced Design System (ADS) Co-Simulation feature that enables the designer to design, analyze, and verify a whole system under one environment.

Setup

1. The Agilent Ptolemy Time-Synchronous Data Flow engine generates the baseband I-Q waveforms required for 16 Quadrature Amplitude Modulation (QAM). The modulated Signals are applied to a transistor-level I-Q modulator circuit that has two Gilbert Cell mixers, a power amplifier, a phase shifter, and a power divider. The output of the modulated signal is then transmitted and received by a Sub-System level I-Q demodulator that recovers the original I-Q data.

2. "FULL_COSIM_16QAM_constel" generates a Constellation and Trajectory diagrams as the data are being processed and calculated. "FULL_COSIM_16QAM_interactive" allows one to interactively control the signal input level and see the effects of distortion on the constellation diagram. "FULL_COSIM_16QAM" enables the user to see the Modulated Spectrum and the Input and Output data waveforms.
3. "FULL_SOC1" and "FULL_SOC_constel" contain a subsystem level demodulator.
4. "IQmod_system" and "IQmod_subsystem" are system and subsystem level implementations of the modulator using components from the library prior to realizing the design blocks at the circuit level.
5. Other design files are used to analyze various parts of the system.



Analysis

Figure 1: Synthesized 16-QAM baseband I and Q data generation

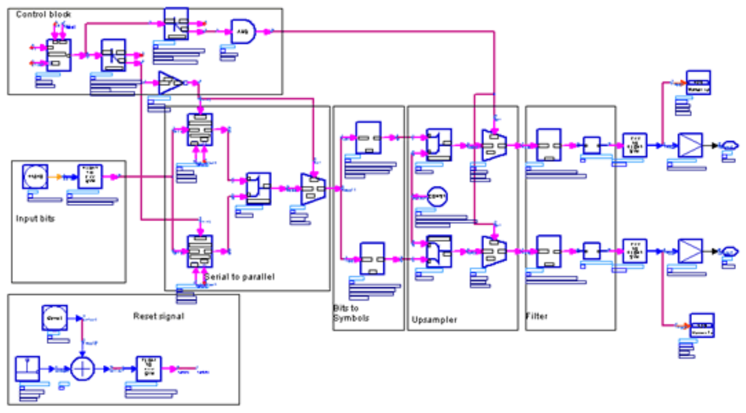


Figure 2: I-Q modulator block diagram

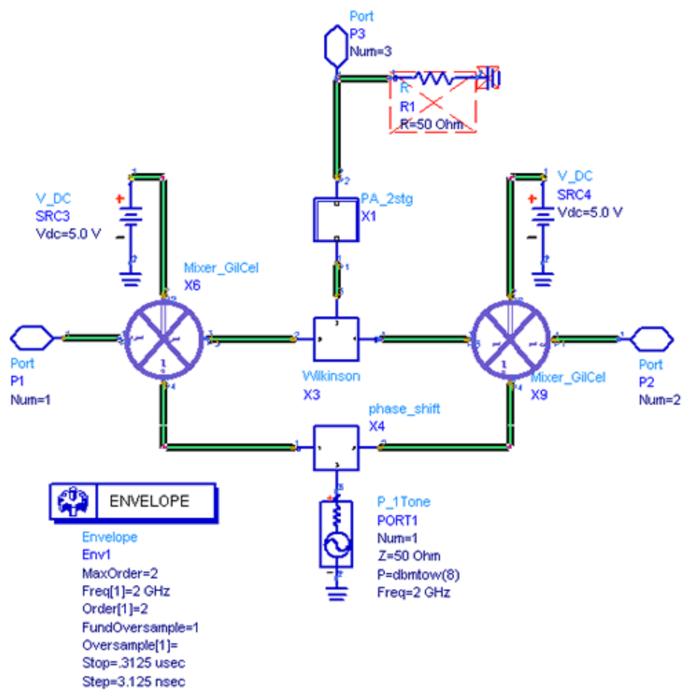


Figure 3: Gilbert cell mixer

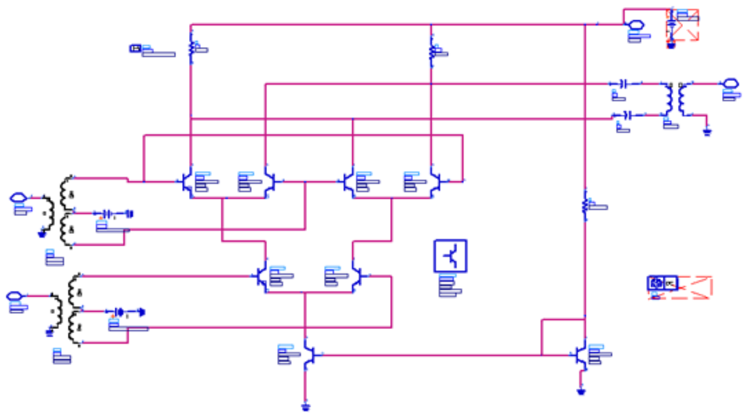


Figure 4: Input and recovered I signals

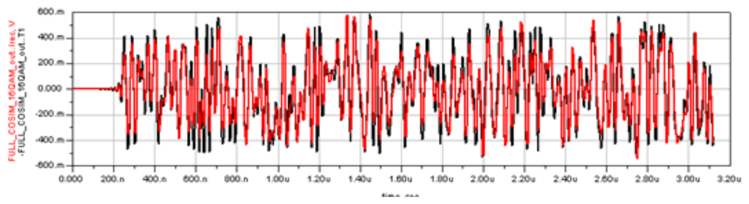
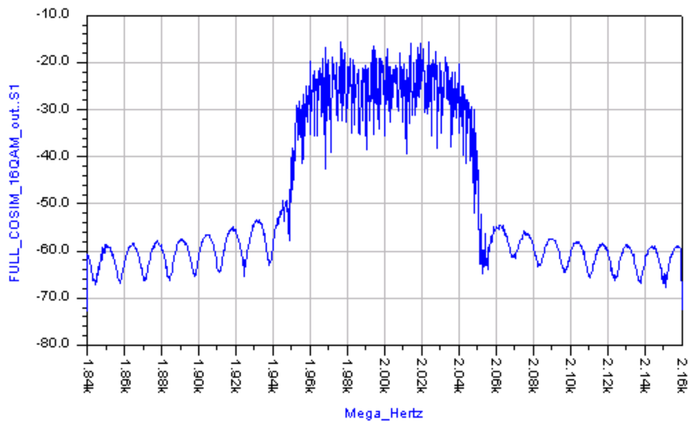


Figure 5: Output spectrum



Notes

- The GainRF element is used to adjust the signal level by attenuation. The DelayRF element is used to delay the input bits by 100 nsec in order to match them with the output recovered bits, which go through a raised cosine filter at the output with 100 nsec delay.
- The EnvOutSelector element is used to select the fundamental frequency spectrum around 2 GHz from the data output of Circuit Envelope.

RFIC Oscillator Simulations

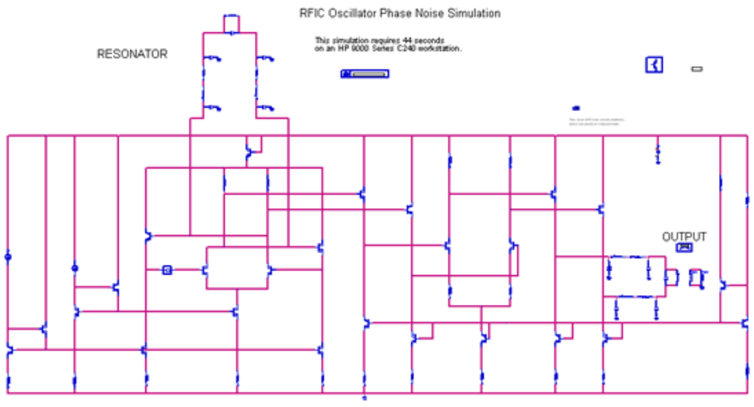
Location: $HPEESOF_DIR/examples/RFIC/RFICoscillator_wrk

Objective

Shows both the transient and Harmonic Balance (HB) simulations for an RFIC oscillator.

Setup

1. "Transient" performs a transient analysis on the oscillator, time domain response can be observed from the simulation.
2. "HB_PhaseNoise" uses an oscport in the HB simulation for the calculation of phase noise and harmonic spectrum.


RFIC Oscillator Phase Noise Simulation
This simulation requires 44 seconds on an HP 9000 Series C240 workstation.
RESONATOR
OUTPUT

Analysis

Figure 1: Time domain response of the oscillator

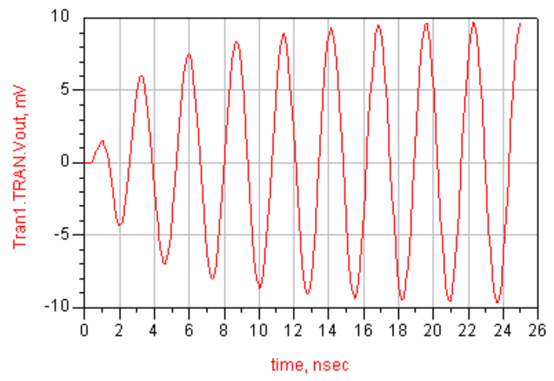
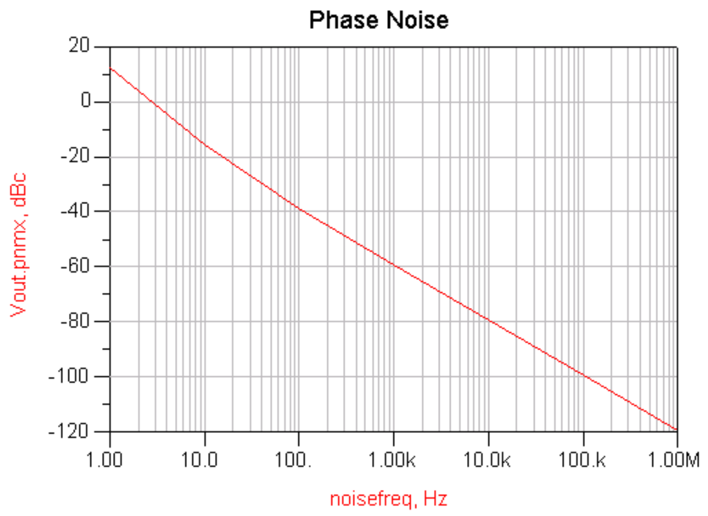


Figure 2: Phase noise due to noise mixing



Notes

1. Phase noise in an oscillator is analyzed by small-signal mixing of noise. The small signal mixing of noise comes from nonlinear behavior of the oscillator, where noise mixes with the oscillator signal and harmonics to mix to sideband frequencies.

Simulation of a Differential-Mode Mixer

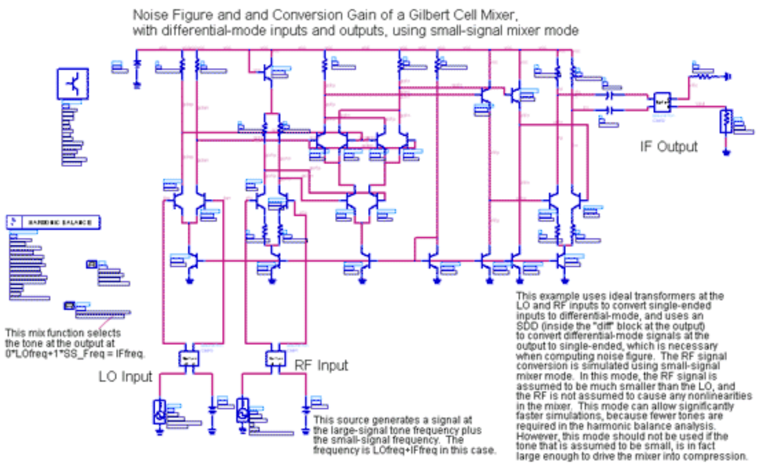
Location: $HPEESOF_DIR/examples/RFIC/MixerDiffMode_wrk

Objective

This example shows how to simulate a differential-mode Gilbert cell mixer. RF frequency is 900MHz, LO frequency is 850MHz. Small-signal mixer mode is used for a quick calculation of conversion gain assuming the RF signal is small enough to not drive the mixer into compression. A noise figure simulation is also included.

Setup

Design file "mixer" uses Harmonic Balance (HB) simulation to simulate the conversion gain and noise figure of a Gilbert Cell Mixer.


Noise Figure and and Conversion Gain of a Gilbert Cell Mixer,
with differential-mode inputs and outputs, using small-signal mixer mode
HARMONIC BALANCE
IF Output
LO Input
RF Input

This mix function selects the tone at the output at 0*LOfreq+1*SS_Freq = IFfreq.

This source generates a signal at the large-signal tone frequency plus the small-signal frequency. The frequency is LOfreq+IFfreq in this case.

This example uses ideal transformers at the LO and RF inputs to convert single-ended inputs to differential-mode, and uses an SDD (inside the "diff" block at the output) to convert differential-mode signals at the output to single-ended, which is necessary when computing noise figure. The RF signal conversion is simulated using small-signal mixer mode. In this mode, the RF signal is assumed to be much smaller than the LO, and the RF is not assumed to cause any nonlinearities in the mixer. This mode can allow significantly faster simulations, because fewer tones are required in the harmonic balance analysis. However, this mode should not be used if the tone that is assumed to be small, is in fact large enough to drive the mixer into compression.

Analysis

Figure 1: Noise figure versus LO power

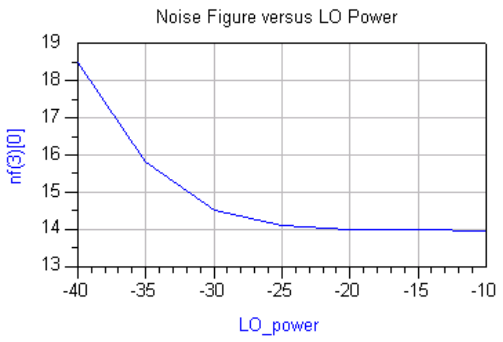
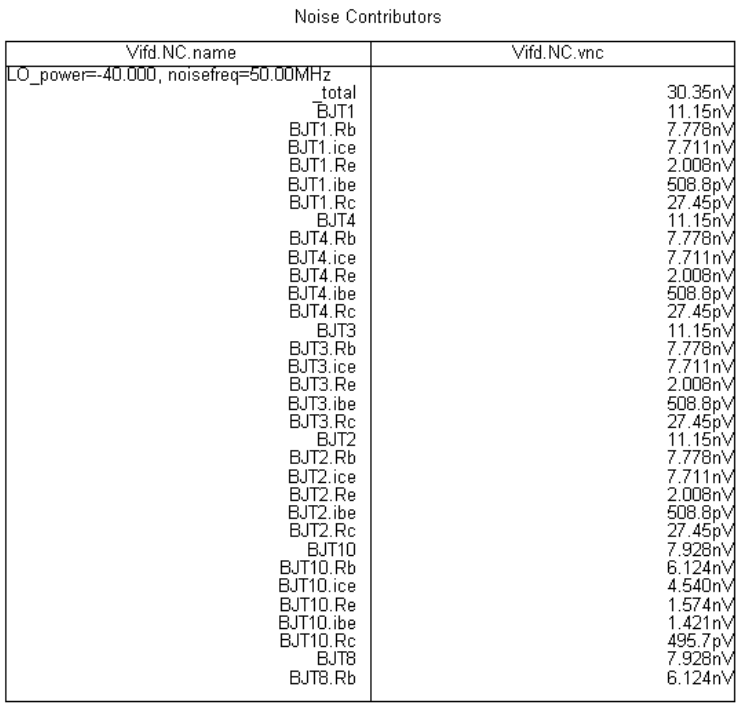


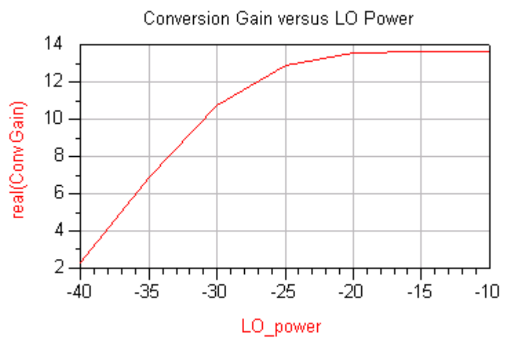
Figure 2: Noise contribution from each noise source



Noise Contributors

Vifd.NC.name	Vifd.NC.vnc
LO_power=-40.000, noisefreq=50.00MHz	
_total	30.35nV
BJT1	11.15nV
BJT1.Rb	7.778nV
BJT1.ice	7.711nV
BJT1.Re	2.008nV
BJT1.ibe	508.8pV
BJT1.Rc	27.45pV
BJT4	11.15nV
BJT4.Rb	7.778nV
BJT4.ice	7.711nV
BJT4.Re	2.008nV
BJT4.ibe	508.8pV
BJT4.Rc	27.45pV
BJT3	11.15nV
BJT3.Rb	7.778nV
BJT3.ice	7.711nV
BJT3.Re	2.008nV
BJT3.ibe	508.8pV
BJT3.Rc	27.45pV
BJT2	11.15nV
BJT2.Rb	7.778nV
BJT2.ice	7.711nV
BJT2.Re	2.008nV
BJT2.ibe	508.8pV
BJT2.Rc	27.45pV
BJT10	7.928nV
BJT10.Rb	6.124nV
BJT10.ice	4.540nV
BJT10.Re	1.574nV
BJT10.ibe	1.421nV
BJT10.Rc	495.7pV
BJT8	7.928nV
BJT8.Rb	6.124nV

Figure 3: Conversion gain versus LO power



Notes

- Small signal mixer mode is used in the HB simulation to achieve fast simulation speed. However, this mode should not be used if the RF tone is large enough to drive the mixer into compression.

Simulation of A Downconverter with FSK Input Signal

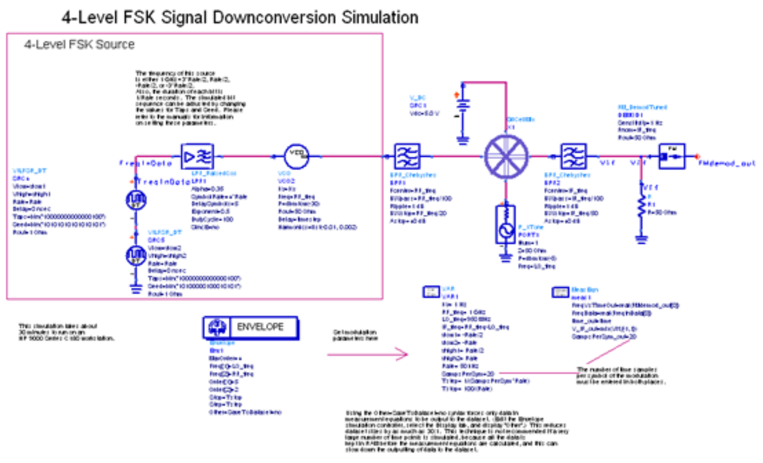
Location: $HPEESOF_DIR/examples/RFIC/FSK_receiver_wrk

Objective

Shows how to use circuit envelope in the simulation of a down-conversion mixer with a 4-level frequency-shift keying (FSK) input signal.

Setup

1. In design "FSK_receiver_wrk", a 4-level FSK source is generated using a VCO component. An FM demodulator is used at the output to obtain the baseband signal.
2. The downconversion mixer is from workspace .../examples/RFIC/Mixers_wrk.
3. The data display compares the ideal input waveform and the distorted, demodulated output waveform. The rms frequency error is also computed, and the user may adjust where the sampling instances are in the demodulated output waveform. The spectrum centered on the IF output frequency is also plotted.



Analysis

Figure 1: Comparison between the ideal input waveform and the demodulated output waveform after compensating for time delay

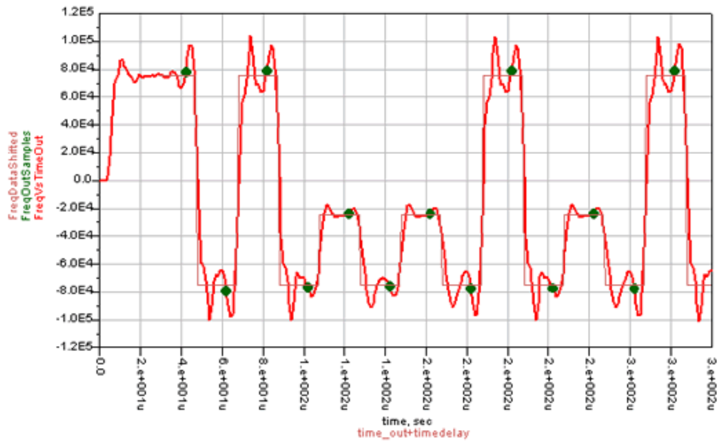
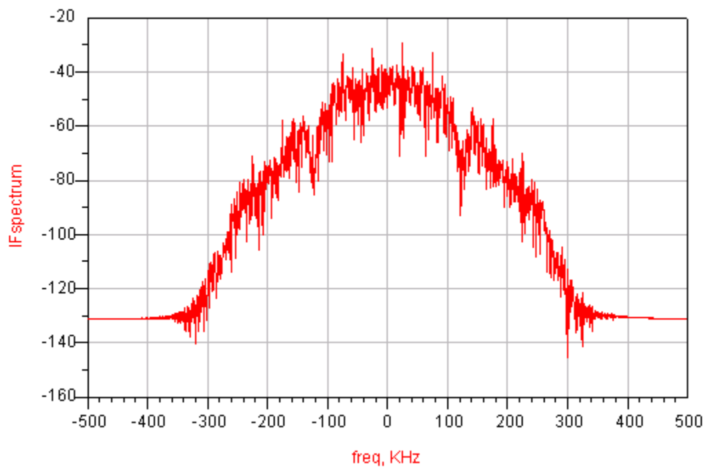


Figure 2: Output spectrum centered on IF frequency



Using Graphic Cell Compiler to Create A Parameterized FET Layout

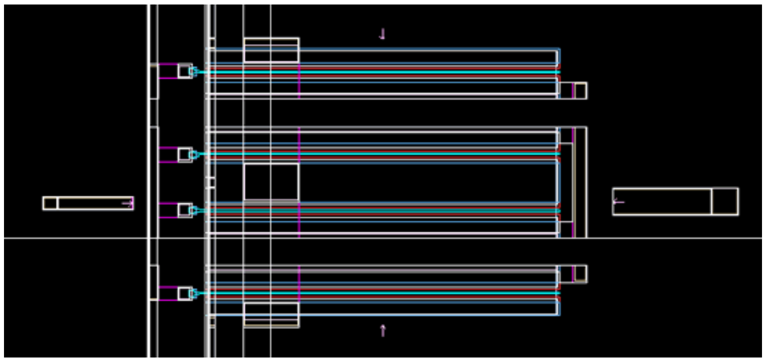
Location: $HPEESOF_DIR/examples/RFIC/GCC_FET_wrk

Objective

This example illustrates the use of Graphical Cell Compiler (GCC) to create a parameterized FET layout.

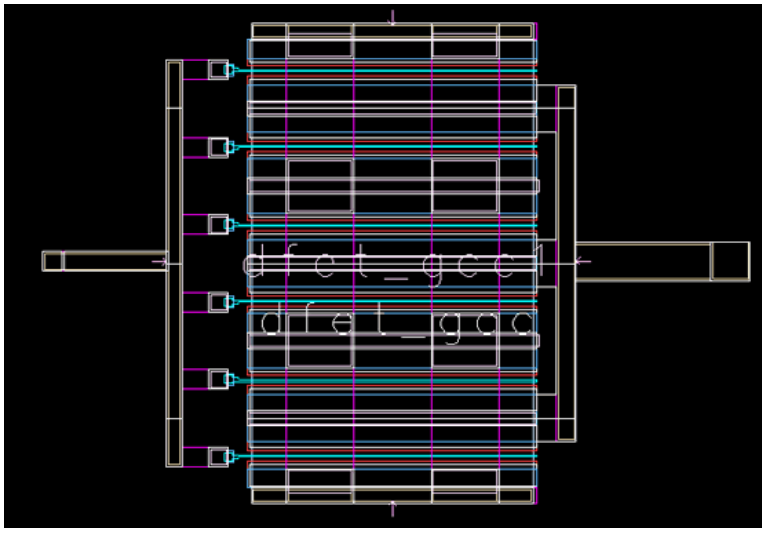
Setup

A complete non-parameterized FET geometry was imported using the GDS-II translator. The Boolean logic operator was used to separate the complete structure into the five unit cells used to create the FET layout (center cell, top cell, bottom cell and the input and output conductor). The GCC re-assembles the five unit cells, replicating the center cell as many times as required. The resulting macro is parameterized as a function of finger width and number of fingers.



Analysis

Figure 1: Parameterized FET Layout



Notes

The creation of this layout is described in detail in *Design Kit Development* (dkarch).

Various Simulations of a Gilbert Cell Mixer

Location: $HPEESOF_DIR/examples/RFIC/mixers_wrk

Objective

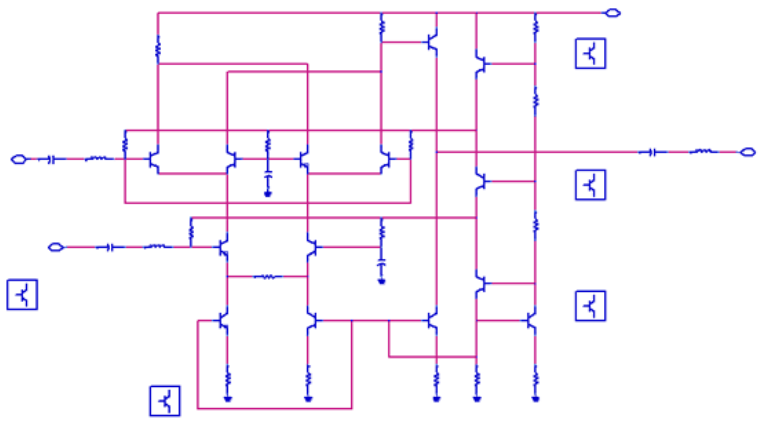
Provides various simulations of a Gilbert Cell mixer, which includes the conversion gain, third order IM, 1dB compression, noise floor and SFDR, RF-to-IF leakage, LO-to-RF leakage, and LO-to-IF leakage. It also provides simulations on a double conversion mixer. Improved noise figure simulation of ADS1.5 is also included.

Setup

1. "DCTests" shows the DC simulation of the mixer.
2. "ConvGain" simulates the conversion gain and noise figure at a single frequency. "ConvGain_wFilt" is the same set-up as ConvGain, except that an image-rejection filter has been added at the input, and a bandpass filter at the IF has been added.
3. "MixerTOI" sets up a two tone test to measure the third-order intermodulation.
4. "RFIFcompression" simulates the 1 dB gain compression point using Gain Compression simulator.
5. "SweptRF" simulates the conversion gain of the mixer as the RF frequency is swept from 200 MHz to 7 GHz. "SweptRF_NF" simulates the noise figure as a function of RF frequency, with the IF fixed at 70 MHz.
6. "NoiseFloor" simulates the total noise power at the output of the mixer. This result is combined with the "MixerTOI" simulation result to calculate the spurious-free dynamic range.
7. "IMDRFSwpHB" and "IMDLOSwpHB" simulates the mixer's intermodulation distortion, conversion gain, RF-to-IF leakage, LO-to-RF leakage, and LO-to-IF leakage versus RF

and LO input power, using harmonic balance.

8. "IMDRFSwpEnv" simulates the mixer's intermodulation distortion versus RF input power, using the Envelope simulator.
9. "DoubleConvHB" simulates a double downconversion receiver's third-order intercept point and conversion gain. "DblConvImag" simulates the image rejection and conversion gain of the mixer.
10. "RFBandFiltTest" simulates the RF filter's frequency response. "FirstIFFiltTest" simulates the first IF filter's frequency response.
11. "HotColdNF" is a simulation of the mixer noise figure using the Y-factor method. In this method, a noise source (a resistor) at two different temperatures is connected to the mixer's input. The contribution to the total noise power at the output due to the mixer itself is unchanged. But the contribution due to the noise source at the input is different because of its two different temperatures. From this, the noise figure of the mixer can be determined. This simulation models the behavior of the Agilent 8970B Noise Figure Meter.
12. "HotColdNF_wFilt" is the same as "HotColdNF", except that a filter is included at the input so that a single-sideband noise figure is obtained. The conversion gain is also calculated.
13. "HotColdNF_wFiltvsIF" is the same simulation versus IF frequency with the LO fixed.
14. "HotColdNF_wFiltvsLOpwr" is the same simulation versus LO power and IF frequency.



Analysis

Figure 1: IF and Third Order Intermodulation Tones vs RF Power

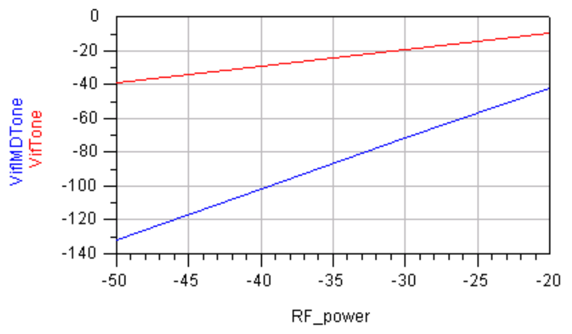
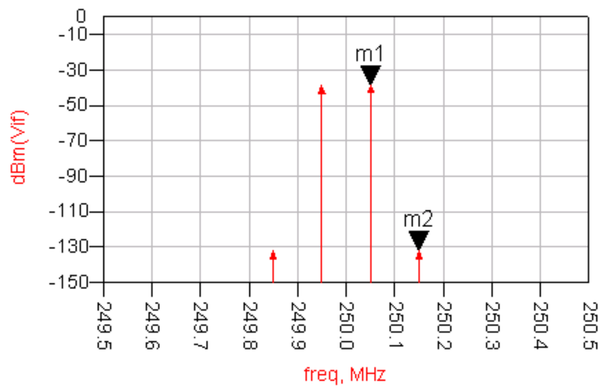


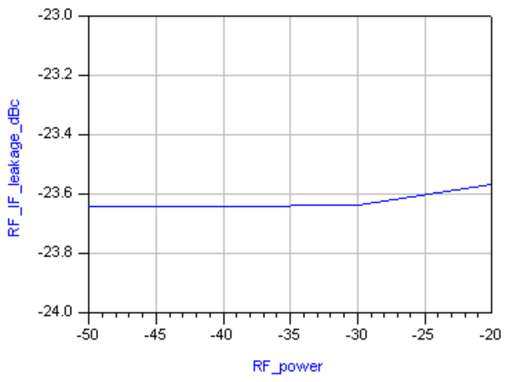
Figure 2: Calculation of IP3 in a two tone test



Eqn TOIoutput=1.5*m1-0.5*m2

TOIoutput
8.106

Figure 3: RF-IF leakage



Notes

- "GilCellMix" shows the Gilbert cell mixer. It is the IAM-81018 mixer originally designed at Avantek. This part is in the HP Communications Components catalog. The designers documented it in a 1990 Applied Microwave article.
- Details on the Y-Factor method of calculating noise figure are covered in the Agilent Technologies Application Note, Fundamentals of RF and Microwave Noise Figure Measurement: AN 57-1
- ADS 1.3 computes the single-sideband noise figure as: NF=(Na +kTo*G1)/(kTo*G1), where Na is the noise added by the mixer, and kTo*G1 is the noise power at the output due to the single side-band downconversion from the RF input to the IF output.
- ADS 1.5 computes the single-sideband noise figure as: NF=(Na kTo*(G1+G2 ...+Gn))/(kTo*G1), where Na and kTo*G1 are as defined above. G2,...,Gn are the conversion gains from the image frequency and all input frequencies that could be converted to the IF.
- Note that the Y-factor noise figure simulation technique does not make any assumptions about the mixer being single-sideband. You make it a single-sideband mixer by adding an image-reject filter at the input. The ADS 1.5 technique will give you the single-sideband noise figure, whether you have an image-reject filter at the input or not.

Various Simulations of A Power Amplifier

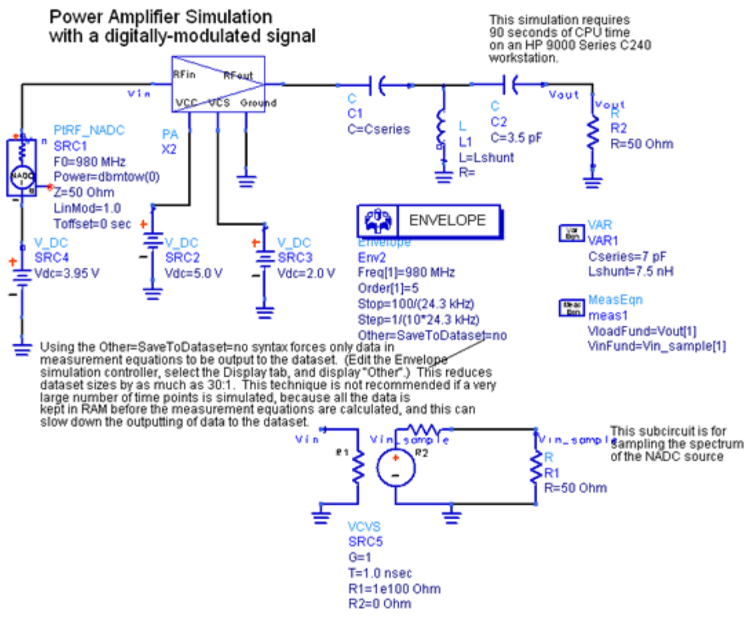
Location: $HPEESOF_DIR/examples/RFIC/amplifier_wrk

Objective

Shows how to set up time domain and frequency domain, as well as a circuit envelope analysis for a 960MHz power amplifier. Simulation includes harmonic balance, transient analysis and an Adjacent Channel Power Ratios (ACPR) test using circuit envelope.

Setup

1. "PA" is the power amplifier subcircuit. The PA is realized using BJT model, it is the last stage of an IQ modulator.
2. "HBtest" uses a HB simulation to calculate the spectrum of various harmonics.
3. "Trantest" does a transient analysis on the PA.
4. "ACPRtest" simulates the PA with a pi/4 DQPSK modulated signal. The output spectrum and trajectory diagram are plotted, and the ACPR is computed.


Power Amplifier Simulation with a digitally-modulated signal

This simulation requires 90 seconds of CPU time on an HP 9000 Series C240 workstation.

Using the Other=SaveToDataset=no syntax forces only data in measurement equations to be output to the dataset. (Edit the Envelope simulation controller, select the Display tab, and display "Other".) This reduces dataset sizes by as much as 30:1. This technique is not recommended if a very large number of time points is simulated, because all the data is kept in RAM before the measurement equations are calculated, and this can slow down the outputting of data to the dataset.

This subcircuit is for sampling the spectrum of the NADC source

Analysis

Figure 1: Spectrum of harmonics from Harmonic Balance simulation

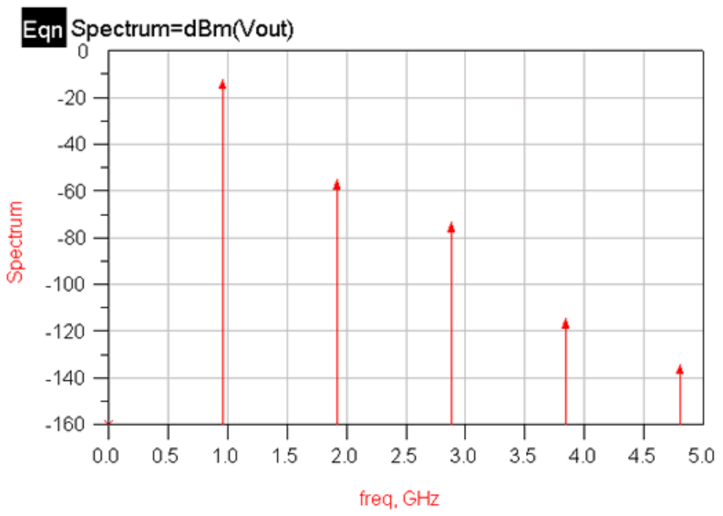


Figure 2: Time domain output waveform for a sinusoidal input signal

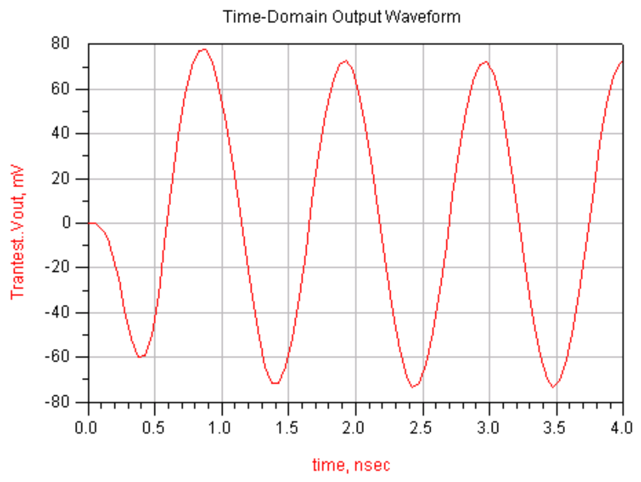


Figure 3: Input and output spectrum

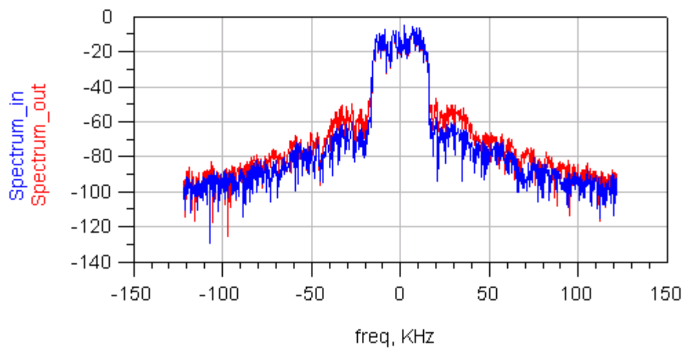


Figure 4: Trajectory diagram at output

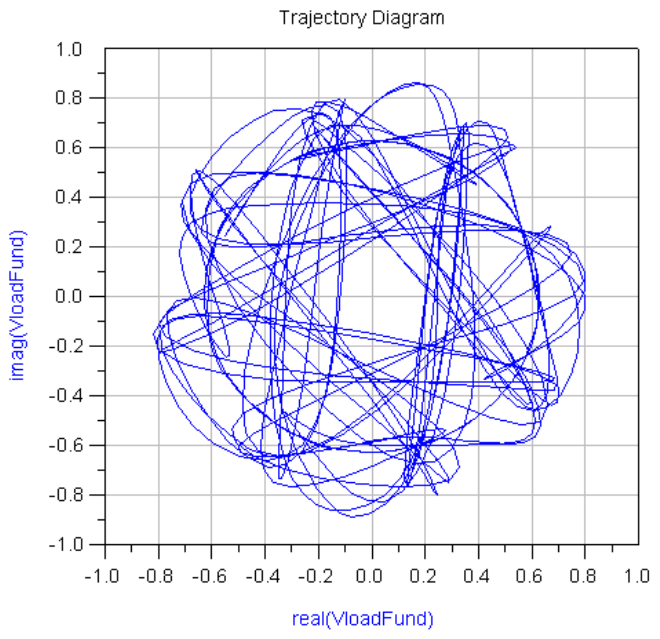
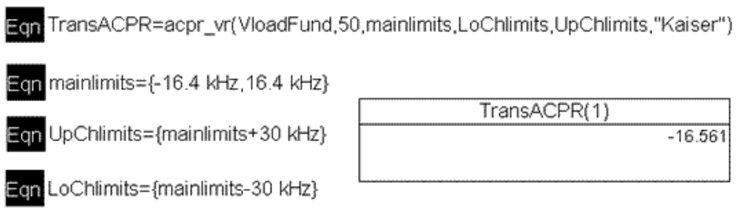


Figure 5: ACPR calculation



```
Eqn TransACPR=acpr_vr(VloadFund,50,mainlimits,LoChlimits,UpChlimits,"Kaiser")

Eqn mainlimits={-16.4 kHz,16.4 kHz}

Eqn UpChlimits={mainlimits+30 kHz}

Eqn LoChlimits={mainlimits-30 kHz}
```

TransACPR(1)
-16.561

Notes

- The acpr_vr() function uses built-in Application Extension Language (AEL) expressions to compute the adjacent channel power ratios. These are the ratios of the power in the frequency bands specified by UpChlimits and mainlimits and LoChlimits and mainlimits. The acpr_vr() function can be used when the terminating impedance is just a resistor. There is also an acpr_vi() function for use when the terminating impedance is complex, in which case the power delivered must be computed in terms of voltage and current. In this case, a current probe must be used to sample the current in the terminating impedance.
- The channel_power_vr() and channel_power_vi() functions are similar to the acpr_vr() and acpr_vi() functions except that they compute the power in Watts in a particular frequency band. They operate on Envelope data.

RF System in Package (SiP) Examples

Examples of how to create RF System-in-Package (or Module) designs in ADS.

- *An LTCC modeling and design example* (examples)
- *RF System-in-Package BPF design example* (examples)

An LTCC modeling and design example

Location: $HPEESOF_DIR/examples/RF_SiP/GT943_LTCC_Modeling_wrk

Objective

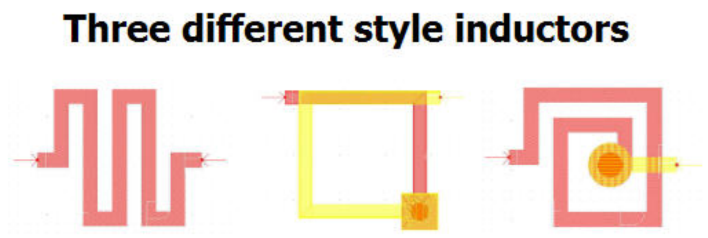
In this example, an LTCC modeling and design example is illustrated. The LTCC material used in this design is GreenTape 943 Material System from Dupont. The layer stack-up of this example is 8 metalization layers and 23mils thick. The dielectric constant of this material is 7.4 and the loss tangent is 0.002.

- LTCC inductor modeling and characterization
- Advanced Model Composer(AMC) based parameterized EM model development
- LTCC Low Pass Filter Design Example with Momentum Component

Setup

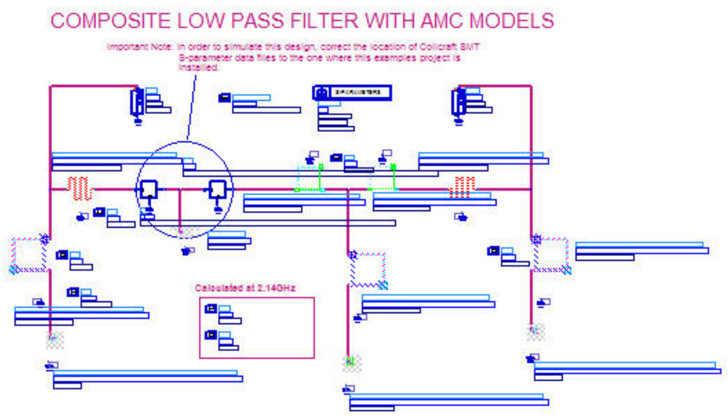
In order to simulate the AMC model based designs in this example, no further installation steps are required. The design kit file, AMC_DK_GT943_7LAYERS_converted, can be found within the workspace directory.

LTCC Modeling and Characterization
Three different style inductors,a meander line, rectangular spiral and helical inductor, are compared.



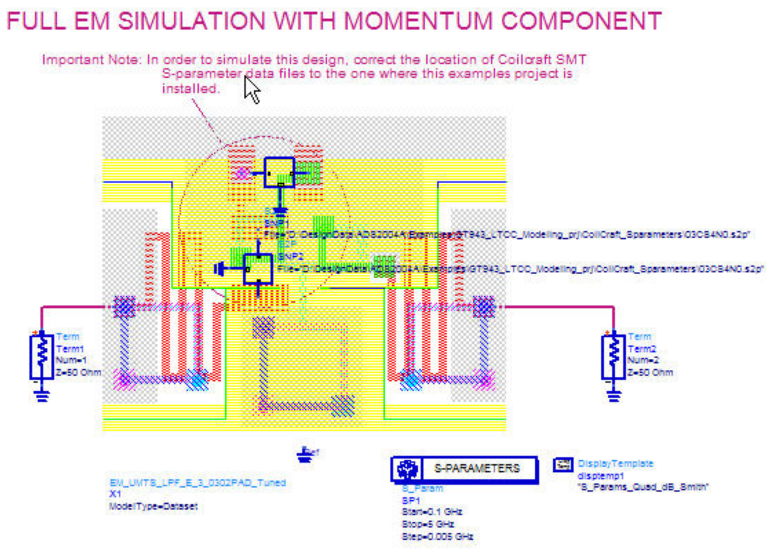
LTCC Low Pass Filter with AMC models
The LTCC composite low pass filter is simulated with accurate AMC models.

Figure 1: Low pass filter with AMC models (SCH_AMC_LTCC_LPF)


COMPOSITE LOW PASS FILTER WITH AMC MODELS
Important Note: In order to simulate this design, correct the location of Coilcraft SMT S-parameter data files to the one where this examples project is installed.
Calculated at 2.14GHz

LTCC Low Pass Filter with Momentum Component
The final LTCC low pass filter is simulated as a Momentum Component.

Figure 2: LTCC Low Pass Filter with Momentum Component (SCH_MOM_COMP_LTCC_LPF_TUNED)


FULL EM SIMULATION WITH MOMENTUM COMPONENT
Important Note: In order to simulate this design, correct the location of Coilcraft SMT S-parameter data files to the one where this examples project is installed.
SNP1
File="D:\DesignData\ADS2004A\Examples\GT943_LTCC_Modeling_prj\CoilCraft_Sparameters\03CS4N0.s2p"
S2P
SNP2
File="D:\DesignData\ADS2004A\Examples\GT943_LTCC_Modeling_prj\CoilCraft_Sparameters\03CS4N0.s2p"
Term
Term1
Num=1
Z=50 Ohm
Term
Term2
Num=2
Z=50 Ohm
EM_UMTS_LPF_E_3_0302PAD_Tuned
X1
ModelType=Dataset
S-PARAMETERS
S_Param
SP1
Start=0.1 GHz
Stop=5 GHz
Step=0.005 GHz
DisplayTemplate
disptemp1
"S_Params_Quad_dB_Smith"

Analysis

Figure 3: Comparison of 3 different style inductors (LTCC_Modeling_Results.dds)

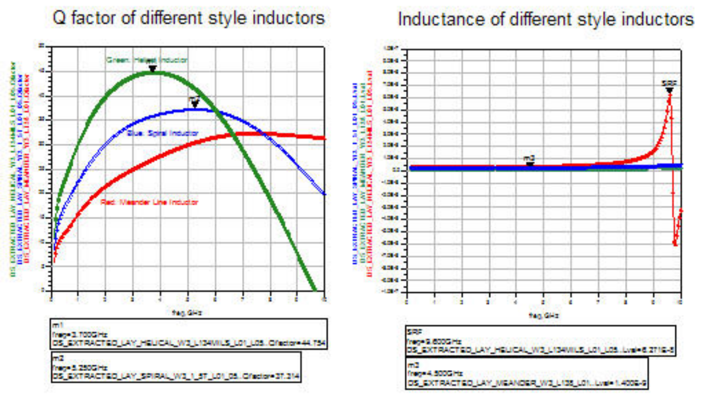
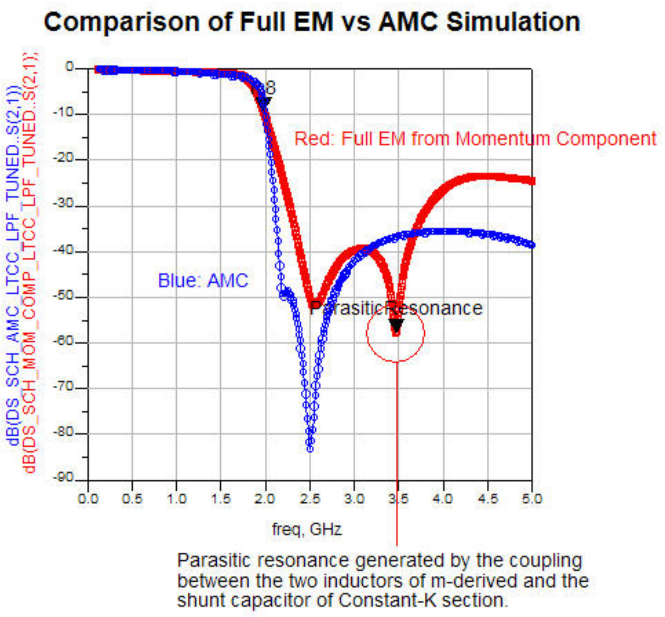


Figure 4: Final Simulation Results of LTCC Low Pass Filter (SimulationResults.dds)



ParasiticResonance
freq=3.470GHz
dB(DS_SCH_MOM_COMP_LTCC_LPF_TUNED..S(2,1))=-57.971

m8
freq=1.980GHz
dB(DS_SCH_MOM_COMP_LTCC_LPF_TUNED..S(2,1))=-9.468

RF System-in-Package BPF design example

Location: $HPEESOF_DIR/examples/RF_SiP/RF_SiP_BPF_wrk

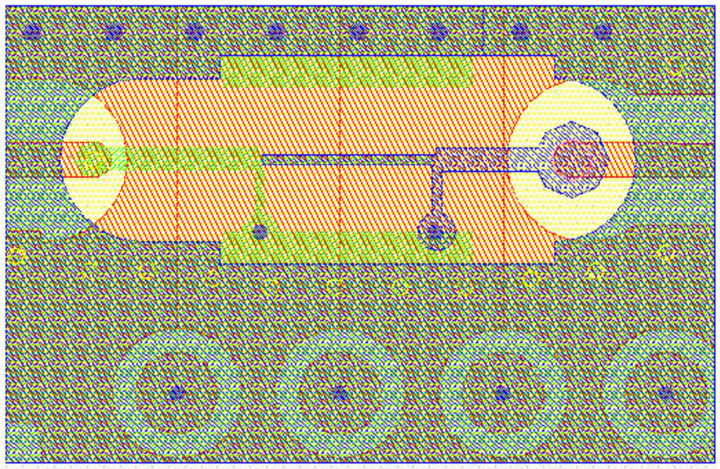
Objective

This example illustrates a RF System-in-Package (or Module) design in ADS. The design is a broadside coupler bandpass filter that is on 6 layer laminate. In this example, a step-by-step RF SiP design flow is demonstrated.

1. An electrical design started with ideal(lumped passive) or distributed components linear simulations which were fast and efficient for tuning and optimization.
2. EM simulations for individual blocks. The filter was sectioned into 3 blocks and each block was EM simulated.
3. A full EM simulation was performed.

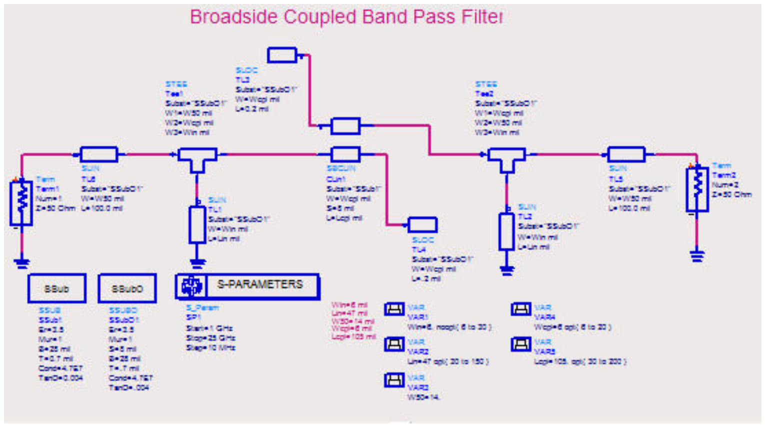
Setup

Original BPF Layout
The drawing shown below is the original layout of the broadside coupler band pass filter.



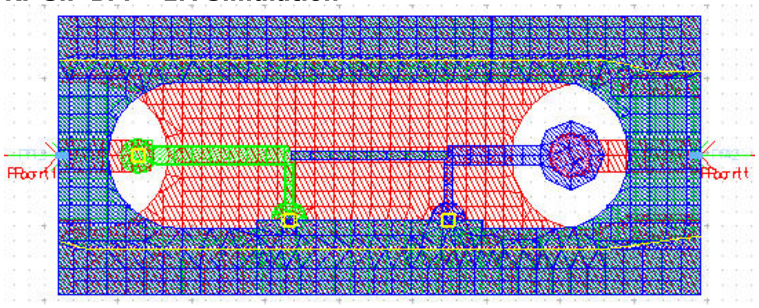
Original BPF drawing (LAY_BPF_ORIGINAL)

RF SiP BPF with distributed ADS models



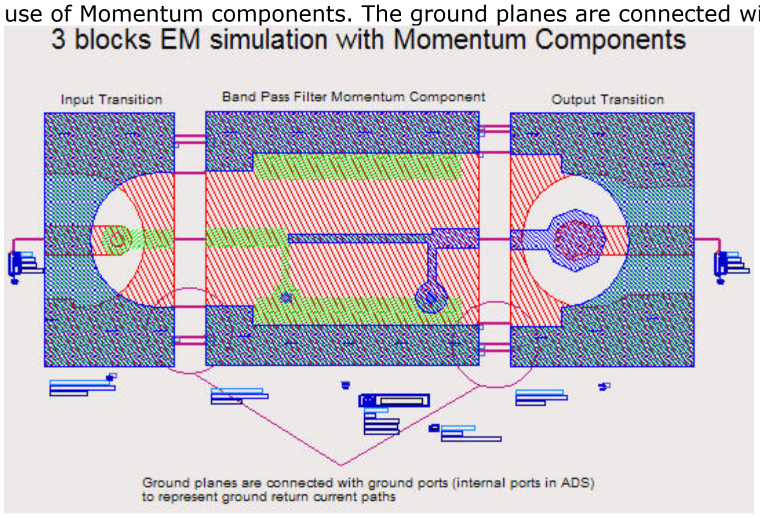
Broadside Coupler Band Pass Filter with ADS distributed models (BPF_Distributed)

RF SiP BPF - EM Simulation



Broadside Coupler Band Pass Filter for EM Simulation (EM_BPF_IMPROVED)

RF SiP BPF - Cascaded 3 networks simulation with GND ports
EM_3NETWORKS_GND_PORTS shows the effect of the ground current return path with the use of Momentum components. The ground planes are connected with ports in this design.


3 blocks EM simulation with Momentum Components
Input Transition
Band Pass Filter Momentum Component
Output Transition
Ground planes are connected with ground ports (internal ports in ADS) to represent ground return current paths

Cascaded 3 networks with ports for ground connections (EM_3NETWORKS_GND_PORTS)

Analysis

Figure 1: Final BPF EM Simulation Result (Results.dds)

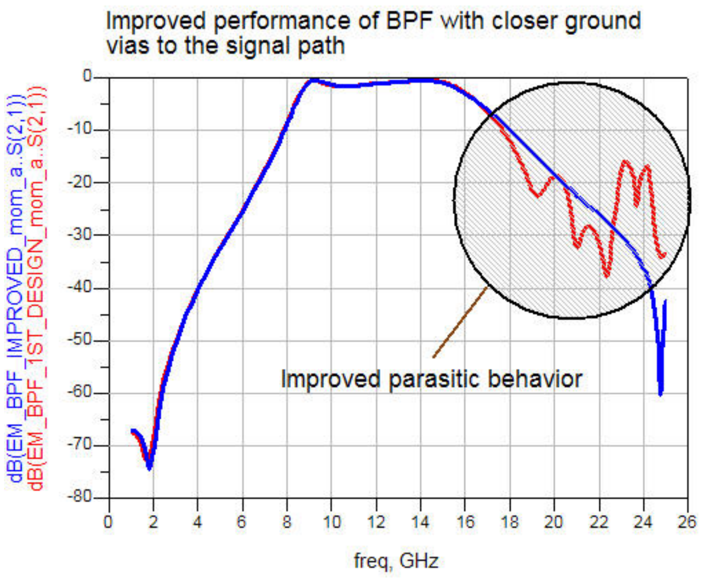
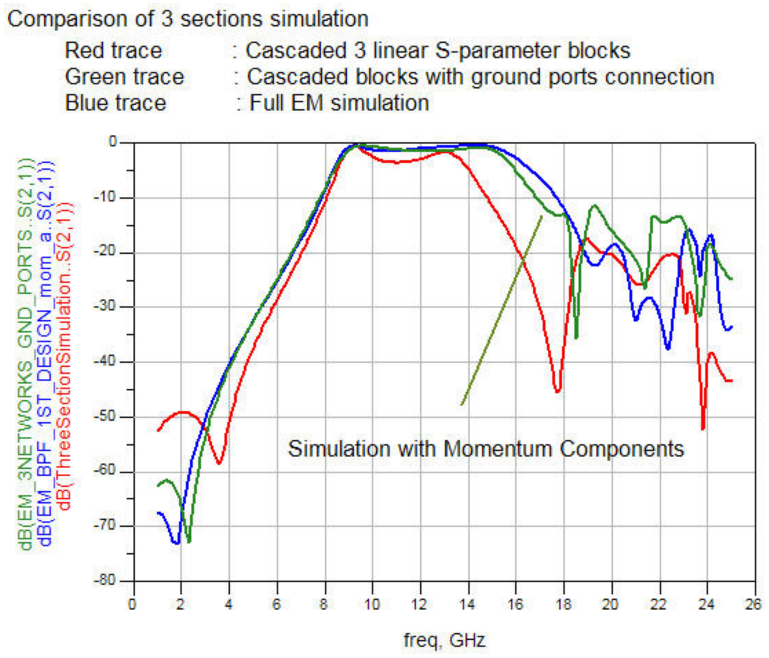


Figure 2: Comparison of 3 sections (networks) simulation (Results.dds)



SDF HDL Cosimulation Examples

Examples of how to run a Synchronous Data Flow and HDL co-simulation using Agilent Ptolemy and the ModelSim SE simulator.

- *HDL Cosimulation Example* (examples)

HDL Cosimulation Example

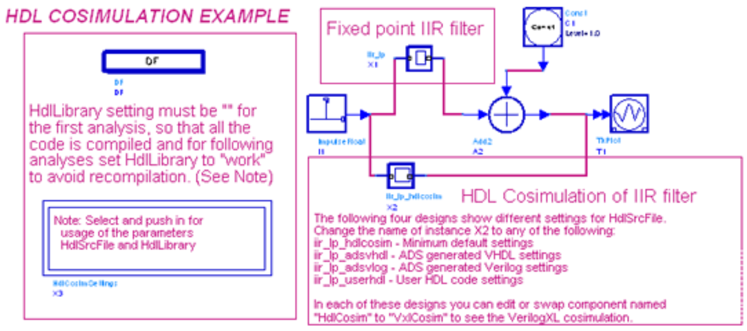
Location: $HPEESOF_DIR/examples/SDFHdlCosim/iir_filter_wrk

Objective

This IIR filter example demonstrates HDL cosimulation using the ModelSim SE simulator.

Setup

1. The impulse data generated in ADS is passed to a fixed point IIR filter design and HDL Cosimulation component, which simulates the equivalent HDL implementation of an IIR filter in the HDL simulator, ModelSim SE from ModelTech.
2. The lowpass filter was designed using the Digital Filter tool in ADS. The HDL code for it was generated using the HDL generation tool in ADS.



HdlLibrary setting must be "" for the first analysis, so that all the code is compiled and for following analyses set HdlLibrary to "work" to avoid recompilation. (See Note)

Note: Select and push in for usage of the parameters HdlSrcFile and HdlLibrary

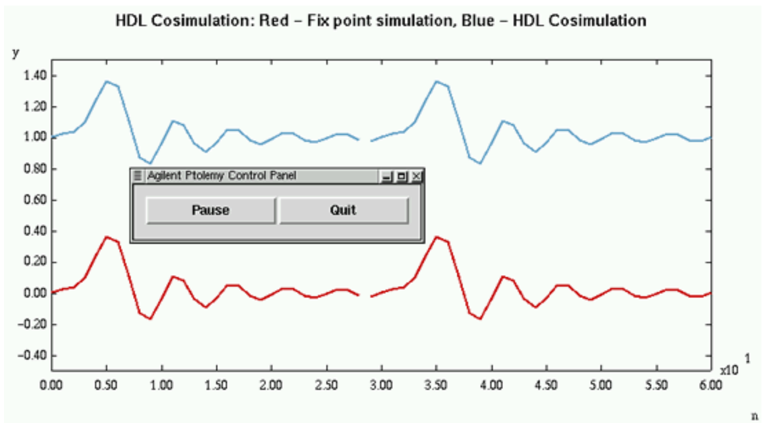
HDL Cosimulation of IIR filter

The following four designs show different settings for HdlSrcFile.
Change the name of instance X2 to any of the following:
iir_lp_hdlcosim - Minimum default settings
iir_lp_adsvhdl - ADS generated VHDL settings
iir_lp_adsvlog - ADS generated Verilog settings
iir_lp_userhdl - User HDL code settings

In each of these designs you can edit or swap component named "HdlCosim" to "VxlCosim" to see the VerilogXL cosimulation.

Analysis

Figure 1: Impulse response



Notes

1. To succesfully run this example you must have ModelSim SE/EE 5.2e or higher installed prior to running the simulation.
2. The output is an impulse response displayed in a TclTk window. Fixed point and HDL simulations can be run side-by-side for comparison using the Const source to produce a visual distinction and avoid overlapping of the signals.
3. To change the cosimulation to use the VerilogXL (3.2 or higher) simulator, push into the HDL design and change the name of the component "HdlCosim" to "VxlCosim".
4. HDL code is compiled the first time you run this example. To avoid recompilation of the code subsequently, set the HdlLibrary to "work".

Signal Integrity Examples

Note
This page has links to the example projects for signal integrity. For related pages please see:

- Signal Integrity Product Support page which has information for signal integrity engineers who are new to ADS
- Signal Integrity Task Groups page which has a mapping of signal integrity tasks to ADS elements. It includes a set of element overviews and links to their documentation.

Examples

Examples of various analyses used for signal integrity applications.

- *Channel Simulation for PCIe* (examples)
- *Channel Simulator Demonstration* (examples)
- *Convert Spectre Transient Data for use with ADS* (examples)
- *Impulse Writer Demonstration* (examples)
- *Jitter Analysis using Ptolemy DF data* (examples)
- *PRBS Jitter using Transient Analysis* (examples)
- *Statistical Eye Diagram Demonstration* (examples)
- *VtPRBS Features and Use Model* (examples)

Channel Simulation for PCIe

Location: $HPEESOF_DIR/examples/SignalIntegrity/Channel_SimulatorPCIe2_wrk

Objective

This workspace demonstrates a channel simulation of a PCIe Gen 2 system.

Setup

- *PCIe_channel_1* is a PCIe channel analysis, with a single TX_Diff component and no crosstalk effects.
- *PCIe_channel_2* demonstrates the effects of crosstalk.
- *PCIe_channel_3* includes the effects of crosstalk and 8B10B encoding.
- *PCIe_channel_4* shows the effects of crosstalk, 8B10B encoding, and receiver jitter.

Channel Simulator Demonstration

Location: $HPEESOF_DIR/examples/SignalIntegrity/ChannelSimulatorTutorial_wrk

Objective

This workspace demonstrate various applications of the ADS Channel Simulator.

Setup

- *DocExample* includes the example described in section *Using Channel Simulation* (cktsimchan).
- *FileSweep* provides an example of sweeping S-parameter files through a channel simulation using the *DataFileList* (cktsimbatch) component.
- *Tuning* demonstrates the ADS tuning features in conjunction with the ChannelSim controller.
- *BatchEQ* demonstrates the use of Batch Simulation to sweep through several sets of equalizer coefficients and study eye diagram response. See *Batch Simulation* (cktsimbatch) for more information.

Convert Spectre Transient Data for use with ADS

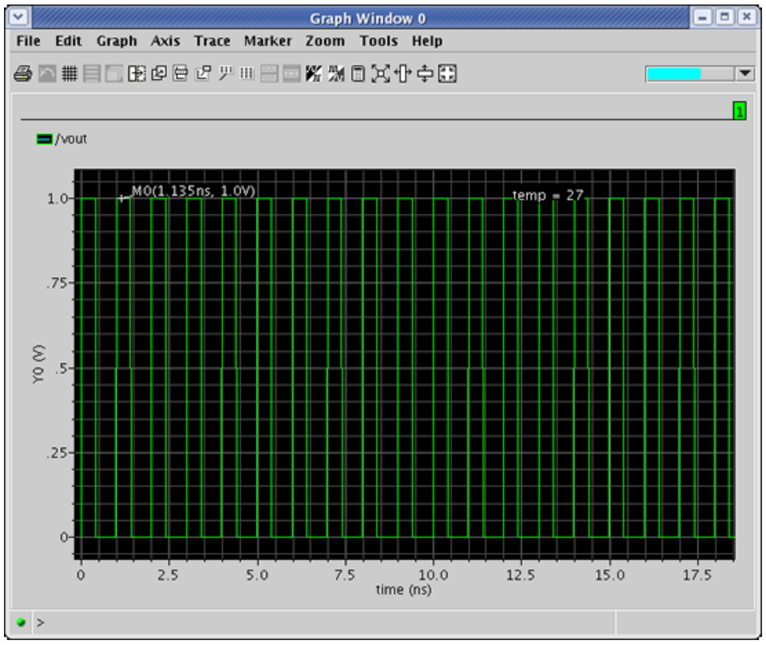
Location: Application Note only

Objective

This Application Note describes how to manually convert Spectre's transient data into a TIM format for use with ADS.

Setup

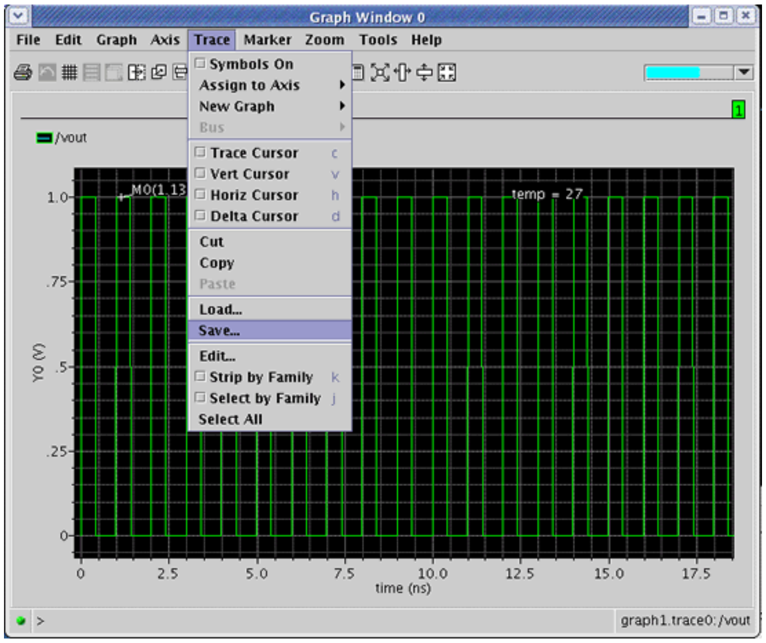
Open the Wavescan and select the trace.

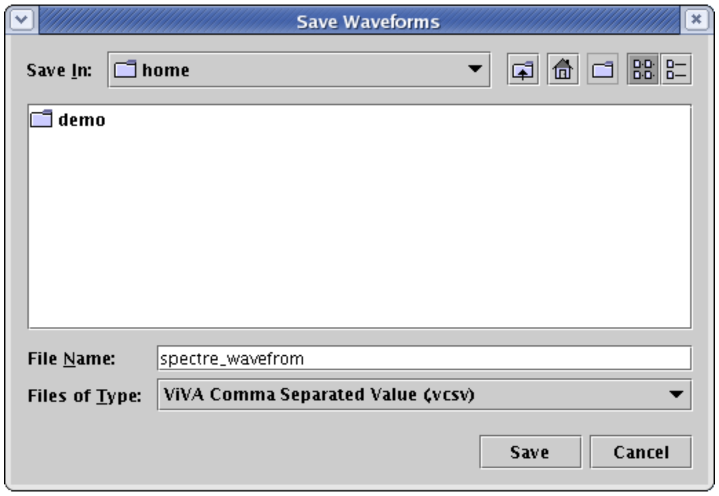


Step 1 - select the trace

Select Trace > Save

This will save the file with the name that you provide, and it will add the extension ".vcsv"





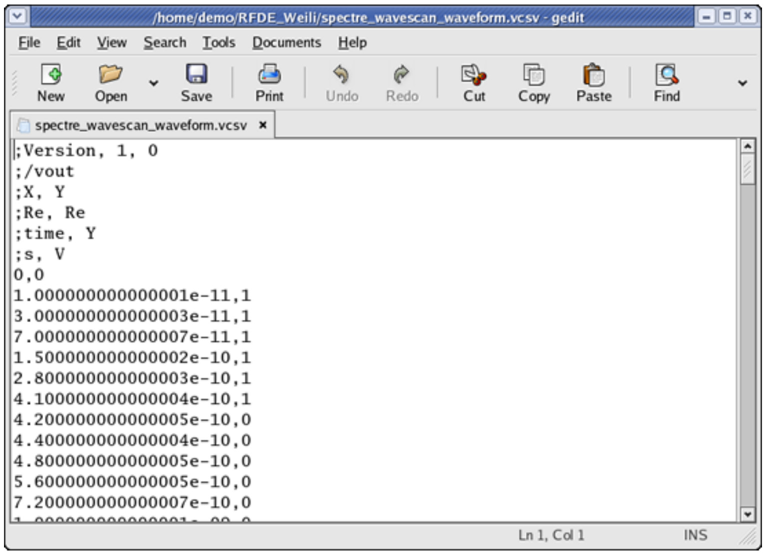
File Name: spectre_wavefrom

Files of Type: ViVA Comma Separated Value (vcsv)

Step 2 - save the trace

Edit the trace file - modify the header

Use a text editor to locate and edit the saved trace file. The header and footer information will have to be changed in order to convert it into a TIM file.


/home/demo/RFDE_Weili/spectre_wavescan_waveform.vcsv - gedit
File
Edit
View
Search
Tools
Documents
Help
New
Open
Save
Print
Undo
Redo
Cut
Copy
Paste
Find
spectre_wavescan_waveform.vcsv
Ln 1, Col 1
INS

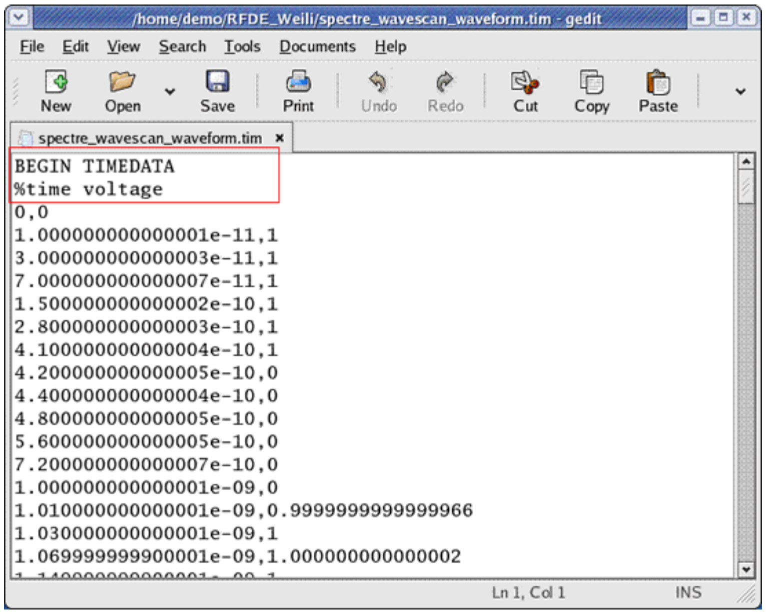
```
;Version, 1, 0
;/vout
;X, Y
;Re, Re
;time, Y
;s, V
0,0
1.000000000000001e-11,1
3.000000000000003e-11,1
7.000000000000007e-11,1
1.500000000000002e-10,1
2.800000000000003e-10,1
4.100000000000004e-10,1
4.200000000000005e-10,0
4.400000000000004e-10,0
4.800000000000005e-10,0
5.600000000000005e-10,0
7.200000000000007e-10,0
```

The original *.vcsv file

Remove all of the comment lines from the trace file and replace them with the following:

```
BEGIN TIMEDATA
% time voltage
```

"time" will be the independent variable, and "voltage" will be the dependent variable. The name "voltage" is arbitrary, and can be given any descriptive name. This will be the name of the data when it is used in ADS.

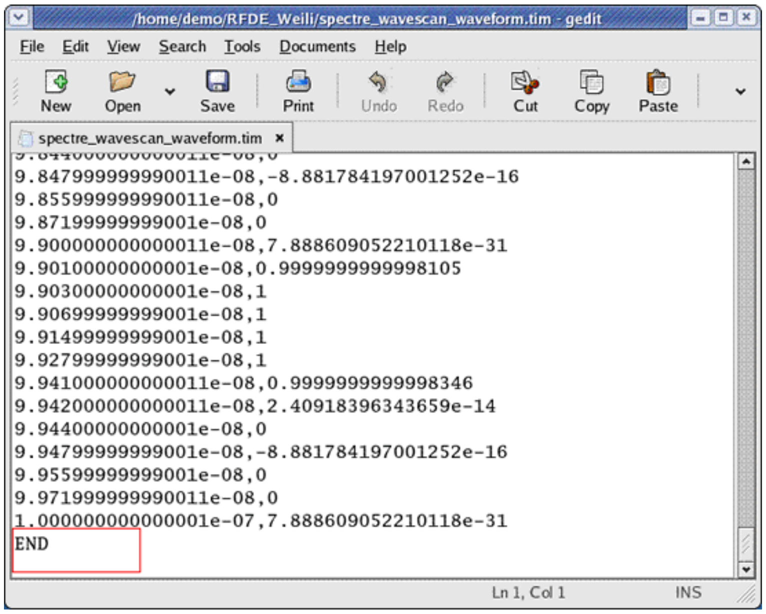

/home/demo/RFDE_Weili/spectre_wavescan_waveform.tim - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste
spectre_wavescan_waveform.tim
Ln 1, Col 1 INS

```
BEGIN TIMEDATA
%time voltage
0,0
1.000000000000001e-11,1
3.000000000000003e-11,1
7.000000000000007e-11,1
1.500000000000002e-10,1
2.800000000000003e-10,1
4.100000000000004e-10,1
4.200000000000005e-10,0
4.400000000000004e-10,0
4.800000000000005e-10,0
5.600000000000005e-10,0
7.200000000000007e-10,0
1.000000000000001e-09,0
1.010000000000001e-09,0.9999999999999966
1.030000000000001e-09,1
1.069999999900001e-09,1.000000000000002
```

Step 3 - Add the header information to define it as a TIM format.

Edit the trace file - modify the footer

The last line of the TIM file will be the word "END". Save the file and change the extension to .tim.


/home/demo/RFDE_Weili/spectre_wavescan_waveform.tim - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste
spectre_wavescan_waveform.tim
Ln 1, Col 1 INS

```
9.847999999990011e-08,-8.881784197001252e-16
9.855999999990011e-08,0
9.871999999999001e-08,0
9.900000000000011e-08,7.888609052210118e-31
9.901000000000001e-08,0.9999999999998105
9.903000000000001e-08,1
9.906999999999001e-08,1
9.914999999999001e-08,1
9.927999999999001e-08,1
9.941000000000011e-08,0.9999999999998346
9.942000000000011e-08,2.40918396343659e-14
9.944000000000001e-08,0
9.947999999999001e-08,-8.881784197001252e-16
9.955999999999001e-08,0
9.971999999990011e-08,0
1.000000000000001e-07,7.888609052210118e-31
END
```

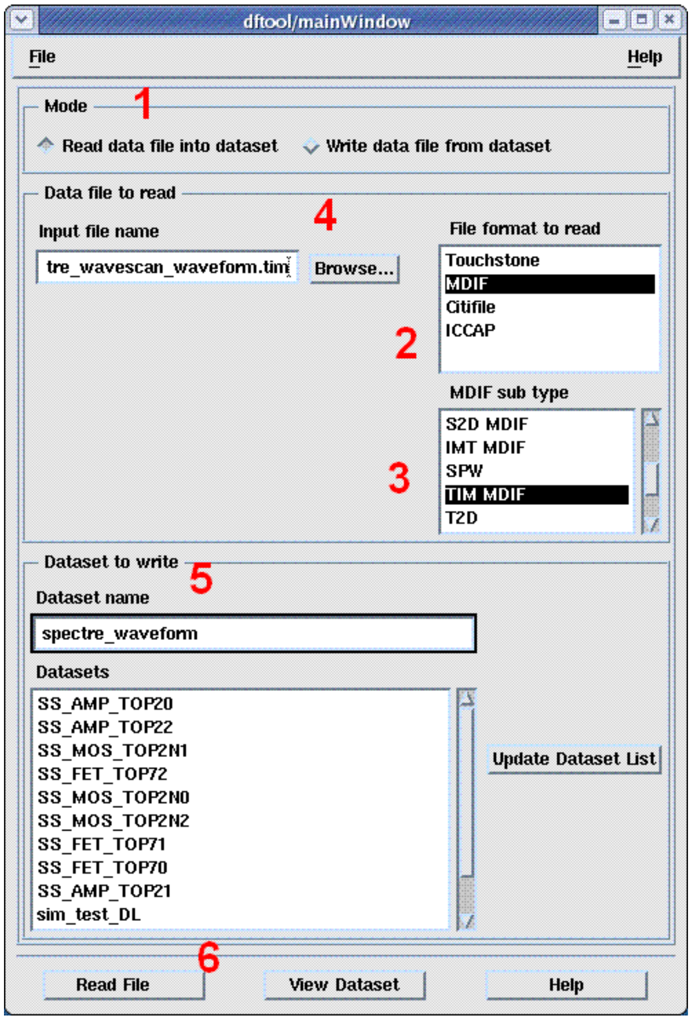
Step 4 - Add the footer information to end the file.

Open the Data File Tool in ADS and Import the data

From an ADS Data Display, select the Data File tool (note: the Data File tool is also available from a schematic page)

The Data File tool will provide a way to select the *.tim file and to save it as a dataset for use in ADS.

1. set the "Mode" to "Read data file into dataset".
2. Set the "File format to read" to "MDIF"
3. Set the "MDIF sub type" to "TIM MDIF"
4. In the "Data File to Read" area, use "Browse" to locate the *.tim file.
5. Enter the dataset name to be whatever name you want for the dataset.
6. Select the "Read File" button to start the translation, which will produce an ADS dataset.



Mode

Read data file into dataset [✓] Write data file from dataset

Data file to read

Input file name: tre_wavescan_waveform.tim

File format to read: Touchstone, [✓ MDIF], Citifile, ICCAP

MDIF sub type: S2D MDIF, IMT MDIF, SPW, [✓ TIM MDIF], T2D

Dataset to write

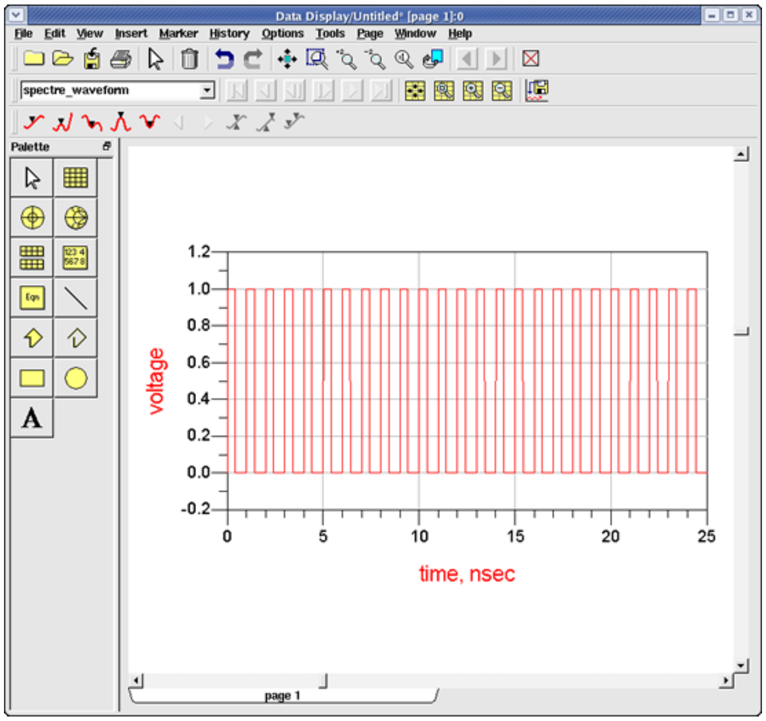
Dataset name: spectre_waveform

Datasets:
SS_AMP_TOP20
SS_AMP_TOP22
SS_MOS_TOP2N1
SS_FET_TOP72
SS_MOS_TOP2N0
SS_MOS_TOP2N2
SS_FET_TOP71
SS_FET_TOP70
SS_AMP_TOP21
sim_test_DL

Step 6 - Import the data into a dataset.

Display the results in the ADS Data Display

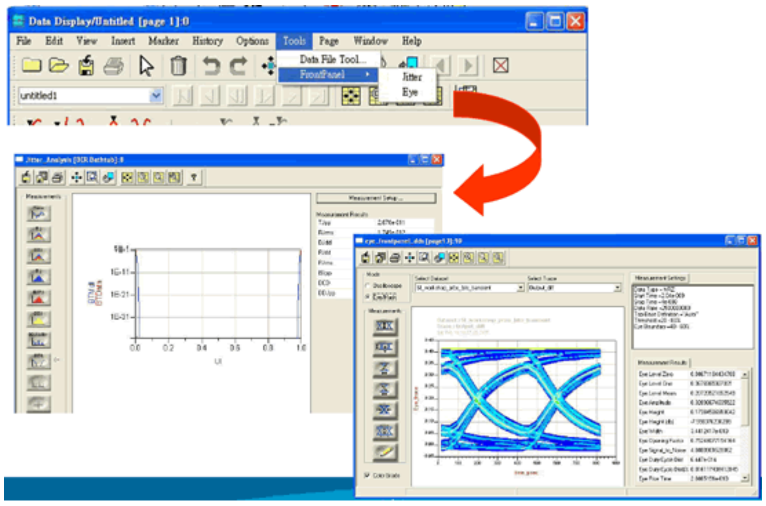
Now that the data is available in a dataset, it can be plotted directly, or used in a simulation as a source (not illustrated)


Data Display/Untitled* [page 1]:0
File Edit View Insert Marker History Options Tools Page Window Help
spectre_waveform
Palette
voltage
time, nsec
page 1

Step 7 - Plot the data in the data display.

The time trace can be used in the Eye Diagram FrontPanel

One purpose for importing the data, especially for digital waveforms, is to use the results in the Eye Diagram FrontPanel.


Data Display/Untitled [page 1]:0
File Edit View Insert Marker History Options Tools Page Window Help
Data File Tool...
FrontPanel
Jitter
Eye
untitled1

Step 8 - Plot the data in the data display.

Impulse Writer Demonstration

Location: $HPEESOF_DIR/examples/SignalIntegrity/ImpulseWriter_wrk

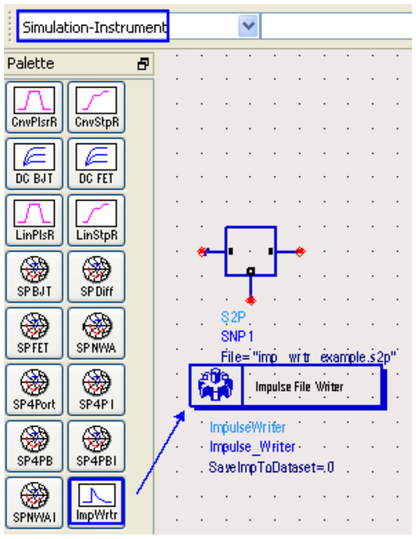
Objective

Import a 2-port Touchstone S-parameter file and use the ImpulseWriter to convert it into causal time domain impulse responses then export an impulse data file.

Setup

The 2-port S-parameter file *imp_wrtr_example.s2p* is imported by the S2P block SNP1. The Impulse Writer icon, named *ImpWrtr*, is located in the Simulation-Instrument palette.

Impulse_Writer



Analysis

The Impulse Writer exports the file *SNP1.CMP1_S2P.imp*.

The causal impulse responses that represent the original 2-port S-parameters are written into this file in the workspace data directory. The content of the file is shown below.

```
[Version] 1.0
# S R 5.000000e+01
[Number of Ports] 2
[Reference] 5.000000e+01 5.000000e+01
[OriginalFrequencyRange] 0.000000e+00 1.000000e+10 Hz
[Time Step] 2.500000e-11 sec
[Base Delay] 0.000000e+00 1.731859e-09 1.731859e-09 0.000000e+00
Number of time points: 1025
3.288703e-02 -8.949016e-03 -7.373915e-04 ...
Number of time points: 1025
3.409677e-01 2.650540e-01 1.488737e-01 ...
Number of time points: 1025
3.409677e-01 2.650540e-01 1.488737e-01 ...
Number of time points: 1025
3.288703e-02 -8.949016e-03 -7.373915e-04 ...
```

Jitter Analysis using Ptolemy DF data

Location: $HPEESOF_DIR/examples/SignalIntegrity/JitterAnalysis_wrk

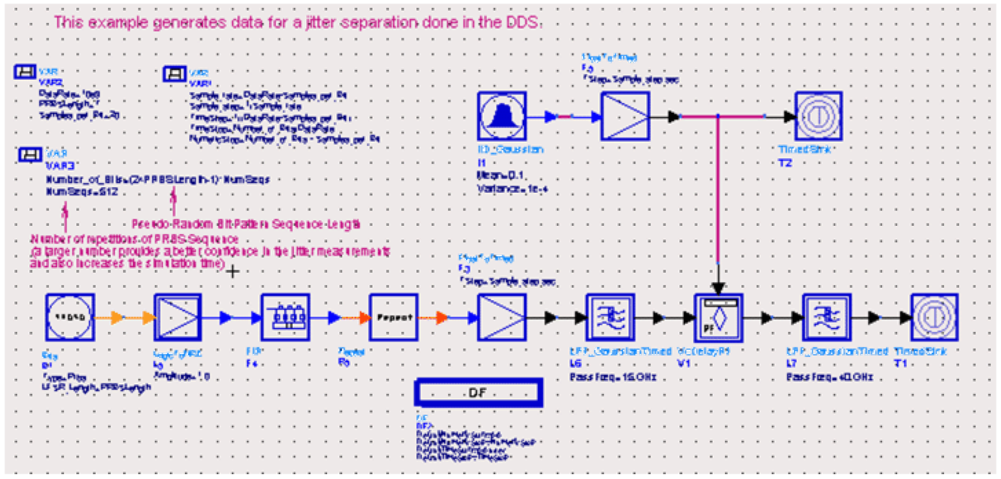
PRBS7_with_RJ_Preemphasis_JitterAnalysis

Objective

Simulate PRBS7 source using Ptolemy DF analysis. The simulated results create jittered data T1. The DDS file is used to do a jitter separation using the jitter_separation() file and the results displayed.

Setup

The jittered data T1 is written to the dataset on doing a simulation. There are 4 pages in the DDS file. The equations in page Jitter Analysis perform a jitter separation and extract the results from the returned values JitRes.

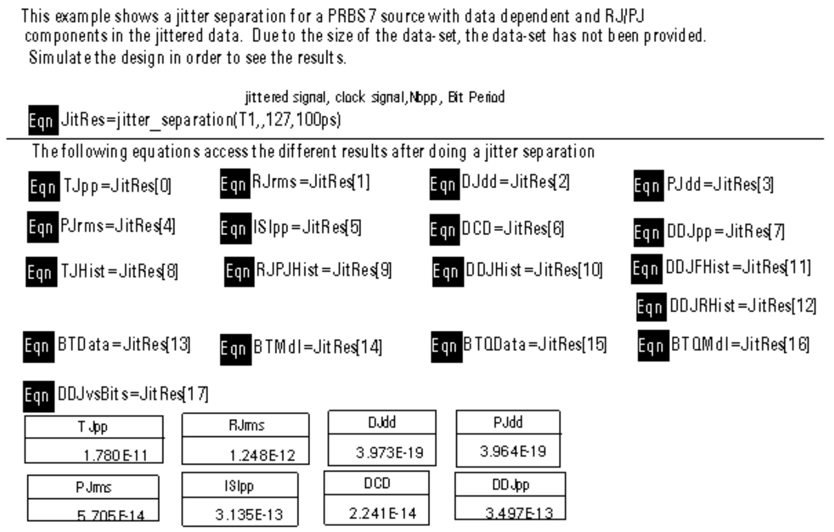

This example generates data for a jitter separation done in the DDS.
VAR2
VAR1
VAR3
Number_of_Bits=(2^PRBSLength-1)*NumSeqs
NumSeqs=512
Pseudo-Random Bit-Pattern Sequence-Length
Number of repetitions of PRBS Sequence
(a larger number provides a better confidence in the jitter measurements and also increases the simulation time)
ID_Gaussian
Mean=0.1
Variance=1e-4
TimedSink
T2
LPF_GaussianTimed
L6
Pass Freq=15 GHz
VcDelayRF
V1
LPF_GaussianTimed
L7
Pass Freq=40 GHz
TimedSink
T1
DF

PRBS7_with_RJ_Preemphasis_JitterAnalysis

Analysis

PRBS7_ with_RJ_Preemphasis_JitterAnalysis.dds

1. Page Jitter Analysis: TJpp, RJrms, DJdd, PJdd, PJrms, ISIpp, DCD and DDJpp display the values of the jitter components.



This example shows a jitter separation for a PRBS7 source with data dependent and RJ/PJ components in the jittered data. Due to the size of the data-set, the data-set has not been provided. Simulate the design in order to see the results.

jittered signal, clock signal, Nbpp, Bit Period
Eqn JitRes=jitter_separation(T1,,127,100ps)

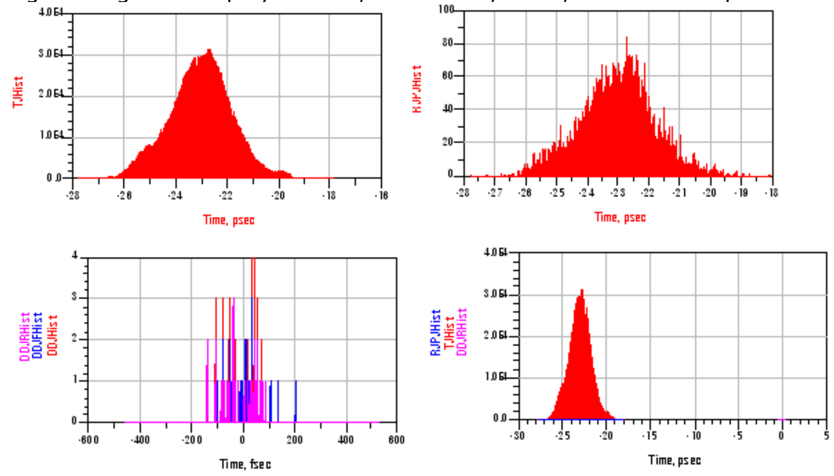
The following equations access the different results after doing a jitter separation

Eqn TJpp=JitRes[0]
Eqn RJrms=JitRes[1]
Eqn DJdd=JitRes[2]
Eqn PJdd=JitRes[3]
Eqn PJrms=JitRes[4]
Eqn ISIpp=JitRes[5]
Eqn DCD=JitRes[6]
Eqn DDJpp=JitRes[7]
Eqn TJHist=JitRes[8]
Eqn RJPJHist=JitRes[9]
Eqn DDJHist=JitRes[10]
Eqn DDJFHist=JitRes[11]
Eqn DDJRHist=JitRes[12]
Eqn BTData=JitRes[13]
Eqn BTMdl=JitRes[14]
Eqn BTQData=JitRes[15]
Eqn BTQMdl=JitRes[16]
Eqn DDJvsBits=JitRes[17]

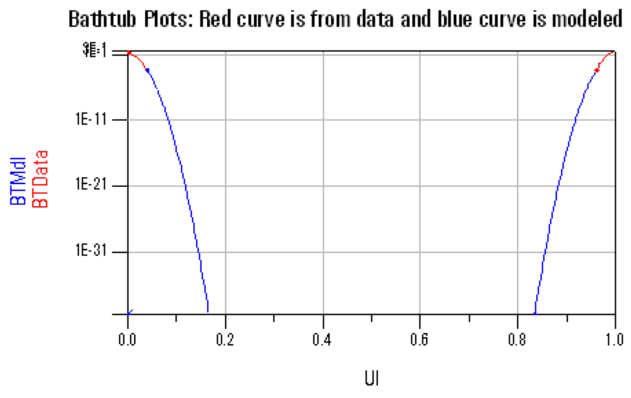
TJpp	RJrms	DJdd	PJdd
1.780E-11	1.248E-12	3.973E-19	3.964E-19

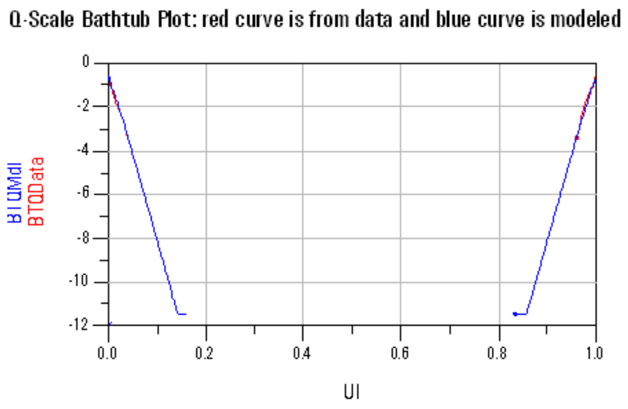
PJrms	ISIpp	DCD	DDJpp
5.705E-14	3.135E-13	2.241E-14	3.497E-13

2. Page Histogram: Displays the TJ, RJPJ DDJR, DDJF, DDJ and composite histograms.

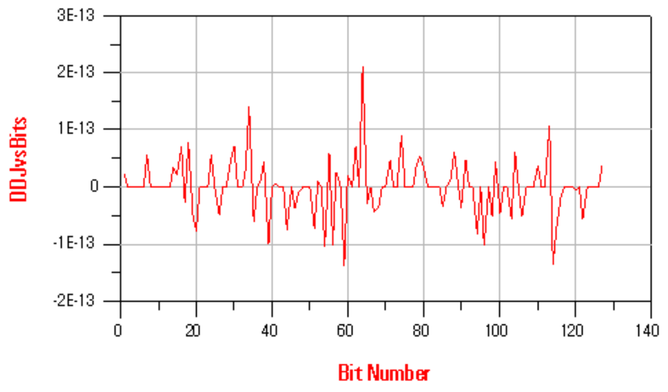


3. Page Bathtub: The bathtub BER versus UI displays the data based and modeled BER values on the Y-axis versus UI on the x-axis. The Q of BER plot displays the Q value of BER versus UI.





4. Page DDJ versus Bits: The plot DDJ versus Bits displays DDJ versus bits.



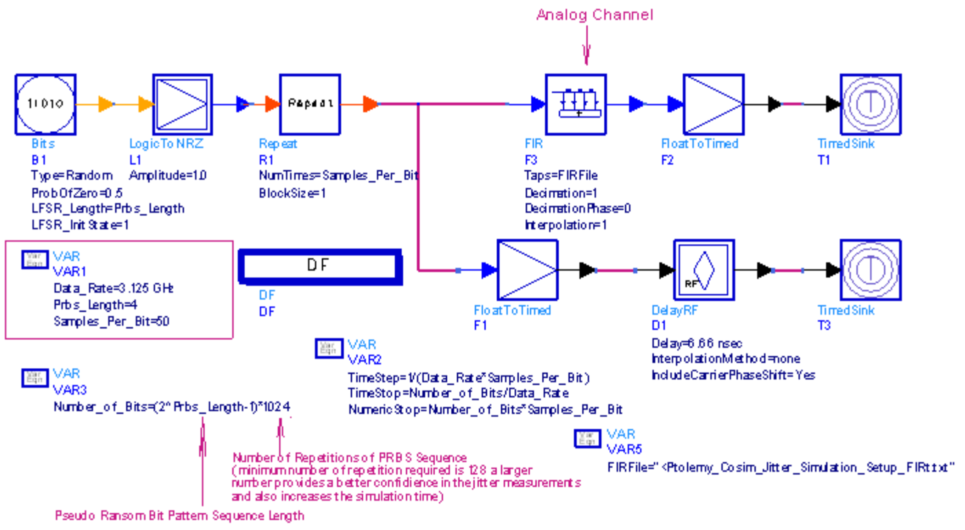
Ptolemy_Cosim_Jitter_Simulation_Setup_FIR

Objective

Simulate an analog channel (saved as taps to an FIR filter using impulse response) with a PRBS4 source using Ptolemy. The simulated results are analyzed in the DDS file Ptolemy_Cosim_Jitter_Simulation_Setup_FIR.dds. This example also shows the method to use time gated and reference signal.

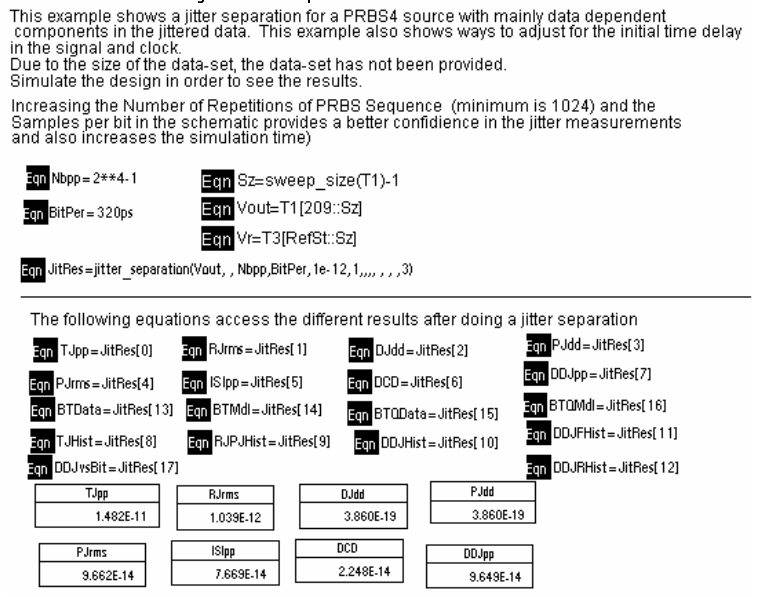
Setup

The jittered signal T1 and the reference signal T3 are written to the dataset on doing a simulation. There are 4 pages in the DDS files. The equations in page "Jitter Analysis" perform a jitter separation and extract the results from the returned values JitRes.


Analog Channel
11010
Bits
B1
Type=Random
ProbOfZero=0.5
LFSR_Length=Prbs_Length
LFSR_InitState=1
LogicToNRZ
L1
Amplitude=1.0
Repeat
R1
NumTimes=Samples_Per_Bit
BlockSize=1
FIR
F3
Taps=FIRFile
Decimation=1
DecimationPhase=0
Interpolation=1
FloatToTimed
F2
TimedSink
T1
VAR
VAR1
Data_Rate=3.125 GHz
Prbs_Length=4
Samples_Per_Bit=50
DF
DF
DF
FloatToTimed
F1
RF
DelayRF
D1
Delay=6.66 nsec
InterpolationMethod=none
IncludeCarrierPhaseShift=Yes
TimedSink
T3
VAR
VAR2
TimeStep=1/(Data_Rate*Samples_Per_Bit)
TimeStop=Number_of_Bits/Data_Rate
NumericStop=Number_of_Bits*Samples_Per_Bit
VAR
VAR3
Number_of_Bits=(2^Prbs_Length-1)*1024
Number of Repetitions of PRBS Sequence
(minimum number of repetition required is 128 a larger
number provides a better confidience in the jitter measurements
and also increases the simulation time)
Pseudo Ransom Bit Pattern Sequence Length
VAR
VAR5
FIRFile="<Ptolemy_Cosim_Jitter_Simulation_Setup_FIR.txt"

Analysis

1. Page Jitter Analysis: TJpp, RJrms, DJdd, PJdd, PJrms, ISIpp, DCD and DDJpp display the values of the jitter components.



This example shows a jitter separation for a PRBS4 source with mainly data dependent components in the jittered data. This example also shows ways to adjust for the initial time delay in the signal and clock.
Due to the size of the data-set, the data-set has not been provided.
Simulate the design in order to see the results.

Increasing the Number of Repetitions of PRBS Sequence (minimum is 1024) and the Samples per bit in the schematic provides a better confidience in the jitter measurements and also increases the simulation time)

Eqn Nbpp=2**4-1
Eqn BitPer=320ps
Eqn Sz=sweep_size(T1)-1
Eqn Vout=T1[209::Sz]
Eqn Vr=T3[RefSt::Sz]
Eqn JitRes=jitter_separation(Vout, , Nbpp,BitPer,1e-12,1,,,,, , , ,3)

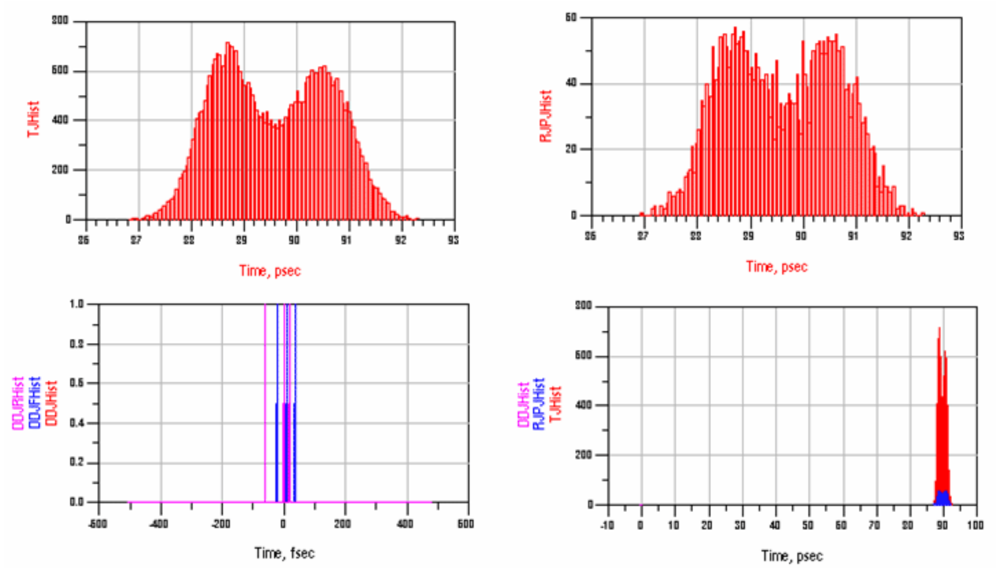
The following equations access the different results after doing a jitter separation

Eqn TJpp=JitRes[0]
Eqn RJrms=JitRes[1]
Eqn DJdd=JitRes[2]
Eqn PJdd=JitRes[3]
Eqn PJrms=JitRes[4]
Eqn ISIpp=JitRes[5]
Eqn DCD=JitRes[6]
Eqn DDJpp=JitRes[7]
Eqn BTData=JitRes[13]
Eqn BTMdl=JitRes[14]
Eqn BTQData=JitRes[15]
Eqn BTQMdl=JitRes[16]
Eqn TJHist=JitRes[8]
Eqn RJPJHist=JitRes[9]
Eqn DDJHist=JitRes[10]
Eqn DDJFHist=JitRes[11]
Eqn DDJvsBit=JitRes[17]
Eqn DDJRHist=JitRes[12]

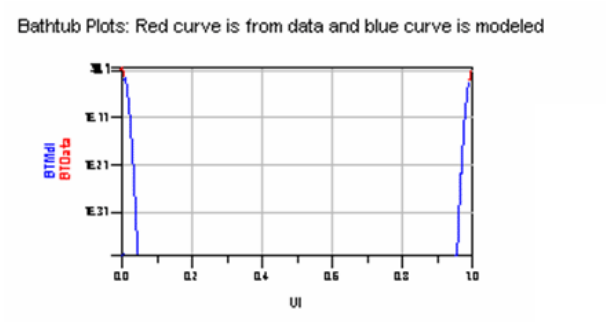
TJpp	RJrms	DJdd	PJdd
1.482E-11	1.039E-12	3.860E-19	3.860E-19

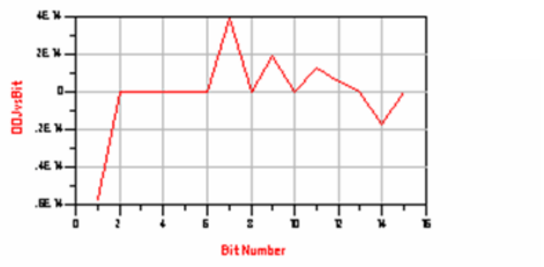
PJrms	ISIpp	DCD	DDJpp
9.662E-14	7.669E-14	2.248E-14	9.649E-14

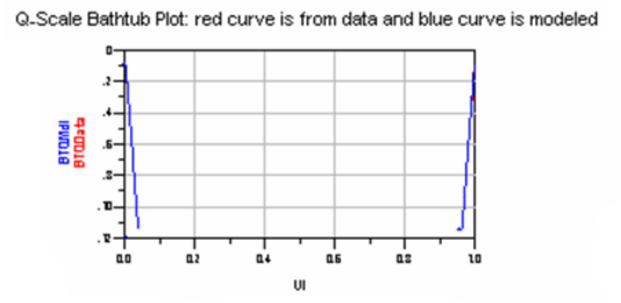
2. Page Histogram: Displays the TJ, RJPJ DDJR, DDJF and DDJ and composite histograms.



3. Page Bathtub and DDJ versus Bits: The bathtub BER versus UI displays the data based and modeled BER values on the Y-axis versus UI on the x-axis. The Q of BER plot displays the Q value of BER versus UI.







PRBS Jitter using Transient Analysis

Location: $HPEESOF_DIR/examples/SignalIntegrity/JitterAnalysis_wrk

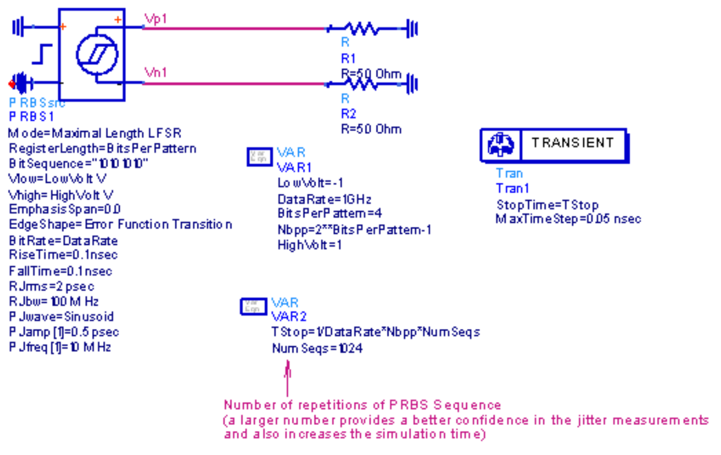
Objective

Simulate the PRBS source with register length of 4 using transient analysis. The simulated results create jittered data. The DDS file PRBS4_jitter.dds is used to do a jitter separation using the jitter_separation() file and the results are displayed.

Setup

The node voltages Vp1 and Vm1 are written to the dataset on doing a simulation. There are 4 pages in the DDS files. The equations in page “Jitter Analysis” perform a jitter separation and extract the results from the returned values JitRes.

This example uses a PRBS4 source to generate data for a jitter separation to be done in DDS..



PRBSsrc
PRBS1
Mode=Maximal Length LFSR
RegisterLength=BitsPerPattern
BitSequence="1010 1010"
Vlow=LowVolt V
Vhigh=HighVolt V
EmphasisSpan=0.0
EdgeShape=Error Function Transition
BitRate=DataRate
RiseTime=0.1nsec
FallTime=0.1nsec
RJrms=2 psec
RJbw=100 MHz
PJwave=Sinusoid
PJamp[1]=0.5 psec
PJfreq[1]=10 MHz

R
R1
R=50 Ohm

R
R2
R=50 Ohm

VAR
VAR1
LowVolt=-1
DataRate=1GHz
BitsPerPattern=4
Nbpp=2**BitsPerPattern-1
HighVolt=1

TRANSIENT
Tran
Tran1
StopTime=TStop
MaxTimeStep=0.05 nsec

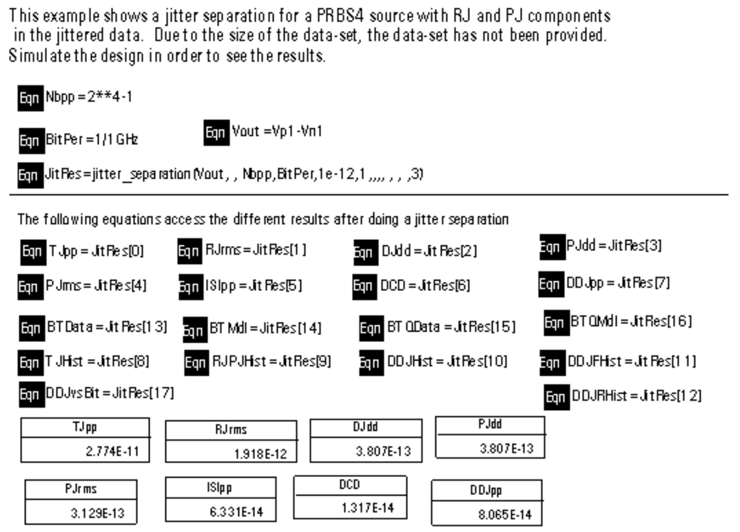
VAR
VAR2
TStop=1/DataRate*Nbpp*NumSeqs
NumSeqs=1024

Number of repetitions of PRBS Sequence
(a larger number provides a better confidence in the jitter measurements
and also increases the simulation time)

Analysis

PRBS4_Jitter.dds

1. Page Jitter Analysis: TJpp, RJrms, DJdd, PJdd, PJrms, ISIpp, DCD and DDJpp display the values of the jitter components.



This example shows a jitter separation for a PRBS4 source with RJ and PJ components in the jittered data. Due to the size of the data-set, the data-set has not been provided. Simulate the design in order to see the results.

```
Eqn Nbpp=2**4-1

Eqn BitPer=1/1 GHz          Eqn Vout=Vp1-Vn1

Eqn JitRes=jitter_separation(Vout, , Nbpp,BitPer,1e-12,1,,,,, , , ,3)
```

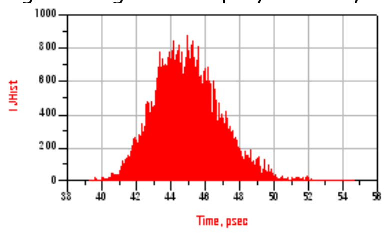
The following equations access the different results after doing a jitter separation

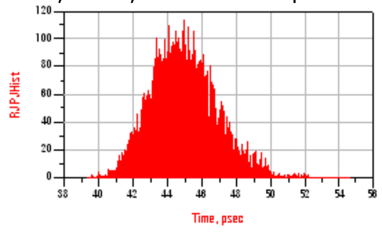
```
Eqn TJpp=JitRes[0]        Eqn RJrms=JitRes[1]       Eqn DJdd=JitRes[2]        Eqn PJdd=JitRes[3]
Eqn PJrms=JitRes[4]       Eqn ISIpp=JitRes[5]       Eqn DCD=JitRes[6]         Eqn DDJpp=JitRes[7]
Eqn BTData=JitRes[13]     Eqn BTMdl=JitRes[14]      Eqn BTQData=JitRes[15]    Eqn BTQMdl=JitRes[16]
Eqn TJHist=JitRes[8]      Eqn RJPJHist=JitRes[9]    Eqn DDJHist=JitRes[10]    Eqn DDJFHist=JitRes[11]
Eqn DDJvsBit=JitRes[17]                                                       Eqn DDJRHist=JitRes[12]
```

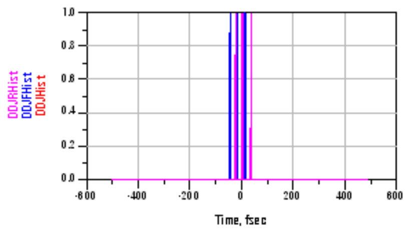
TJpp	RJrms	DJdd	PJdd
2.774E-11	1.918E-12	3.807E-13	3.807E-13

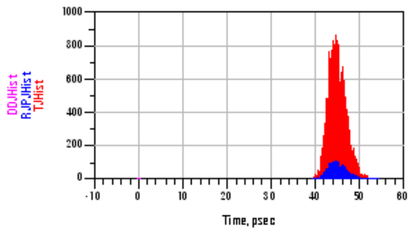
PJrms	ISIpp	DCD	DDJpp
3.129E-13	6.331E-14	1.317E-14	8.065E-14

2. Page Histogram: Displays the TJ, RJPJ DDJR, DDJF, DDJ and composite histograms.

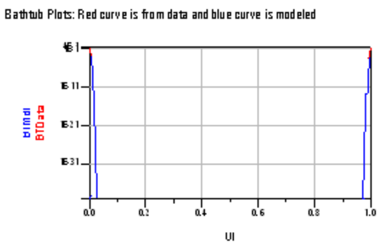


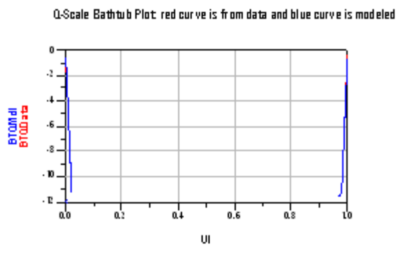


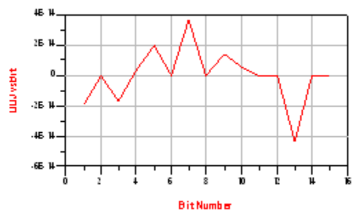




3. Page Bathtub: The bathtub BER versus UI displays the data based and modeled BER values on the Y-axis versus UI on the x-axis. The Q of BER plot displays the Q value of BER versus UI. The DDJ versus Bits plot displays DDJ versus bits.







PRBS7_2500MHz_Scope-2Mpts-NoInterp.dds

Objective

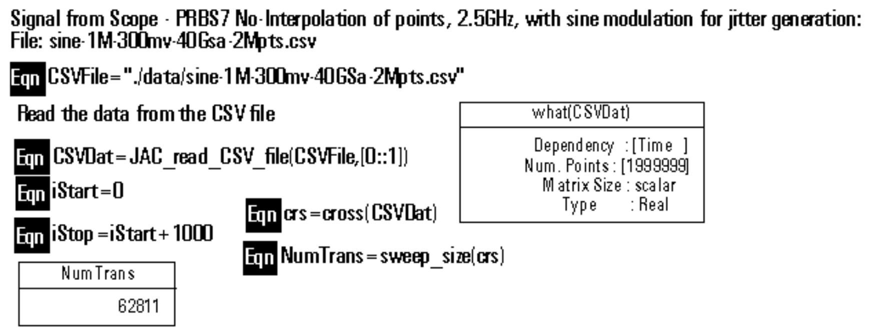
Read the jittered data for a PRBS7 source stored in a CSV file measured from a real-time scope. The data is used to do a jitter separation using the jitter_separation() file and the results displayed.

Setup

There are 5 pages in the DDS file. The equations in page Jitter Analysis perform a jitter separation and extract the results from the returned values JitRes.

Analysis

1. Page Scope Signal – PRBS7 Sine 2.5GHz 300mV: Read the data from the CSV file.



Signal from Scope - PRBS7 No-Interpolation of points, 2.5GHz, with sine modulation for jitter generation:
File: sine-1M-300mv-40Gsa-2Mpts.csv

```
Eqn CSVFile="./data/sine-1M-300mv-40GSa-2Mpts.csv"

Read the data from the CSV file

Eqn CSVDat=JAC_read_CSV_file(CSVFile,[0::1])
Eqn iStart=0
Eqn iStop=iStart+1000

Eqn crs=cross(CSVDat)
Eqn NumTrans=sweep_size(crs)
```

what(CSVDat)
Dependency : [Time] Num. Points : [1999999] Matrix Size : scalar Type : Real

NumTrans
62811

2. Jitter Analysis: TJpp, RJrms, DJdd, PJdd, PJrms, ISIpp, DCD and DDJpp display the values of the jitter components.

Signal from Scope - PRBS7 No-Interpolation of points, 2.5GHz, with sine modulation for jitter generation:
File: sine-1M-300mv-40Gsa-2Mpts.csv
Use the same bit period as Scope - ScopePer to get comparative results

Eqn Nbpp=2**7-1 Eqn ScopePer=1/2.4999341GHz Eqn BitPer=ScopePer

BitPer	Nbpp	ScopePer
4.000105E-10	127.000000	4.000105E-10

Rising Edges.

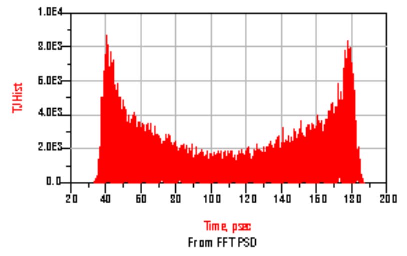
Eqn JitRes=jitter_separation(CSVDat, BitPer, Nbpp,BitPer,,,,,,, 1)

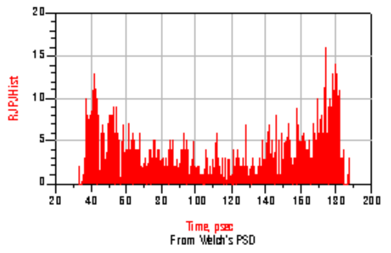
Eqn TJpp=JitRes[0] Eqn RJrms=JitRes[1] Eqn DJdd=JitRes[2] Eqn PJdd=JitRes[3]
Eqn PJrms=JitRes[4] Eqn ISIpp=JitRes[5] Eqn DCD=JitRes[6] Eqn DDJpp=JitRes[7]
Eqn BTData=JitRes[13] Eqn BTMdl=JitRes[14] Eqn BTQData=JitRes[15] Eqn BTQMdl=JitRes[16]
Eqn TJHist=JitRes[8] Eqn RJPJHist=JitRes[9] Eqn DDJHist=JitRes[10] Eqn DDJFHist=JitRes[11]
Eqn DDJvsBit=JitRes[17] Eqn DDJRHist=JitRes[12]

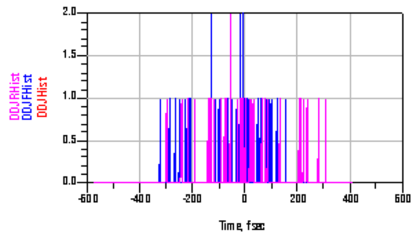
TJpp	RJrms	DJdd	PJdd
1.639E-10	1.573E-12	1.415E-10	1.416E-10

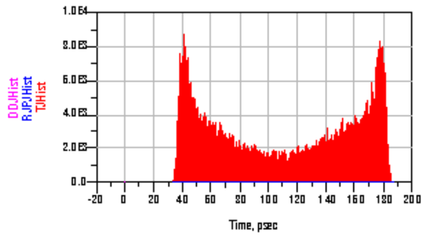
PJrms	ISIpp	DCD	DDJpp
5.097E-11	4.814E-13	6.638E-14	4.814E-13

3. Page Histogram: Displays the TJ, RJPJ DDJR, DDJF, DDJ and composite histograms.









4. Page Bathtub: The bathtub BER versus UI displays the data based and modeled BER values on the Y-axis versus UI on the x-axis. The Q of BER plot displays the Q value of BER versus UI.

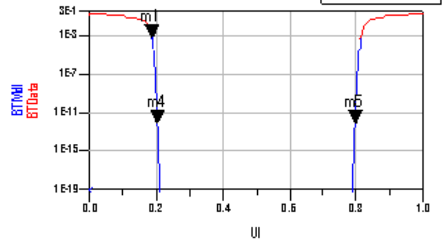
UI	Eye Opening UI	Eye Opening UI*ScopePer
<invalid>	0.593	2.372E-10

Eqn EyeOpeningUI=indep(m5)-indep(m4)

m4
UI=0.204
BTMdl=6.714E-13

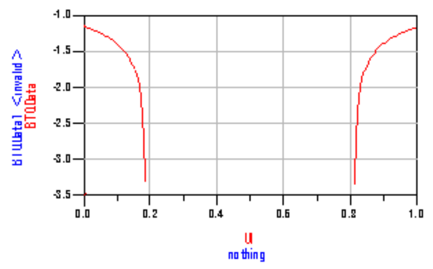
m1
UI=0.186
BTData=4.188E-

m5
UI=0.796
BTMdl=6.714E-13

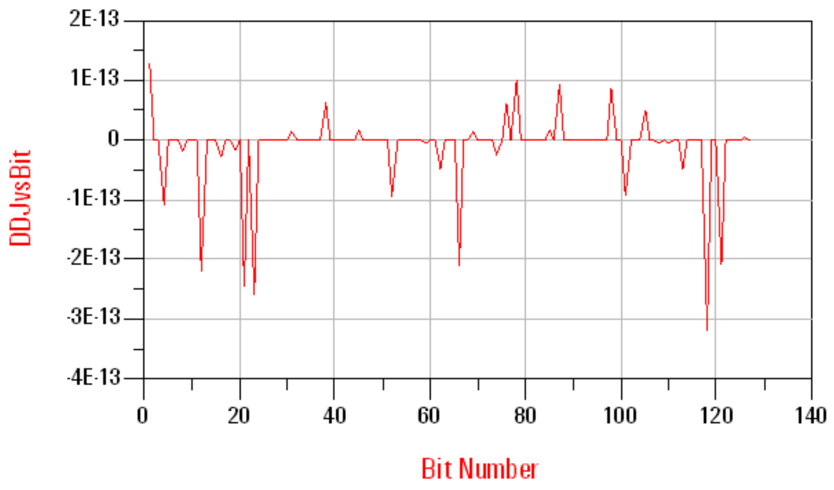


Eqn QBER=7.05
TJ at BER=1e-12
Eqn TJBER12=ScopePer*(1-EyeOpeningUI)
Eqn TJBER12_1=2*QBER*RJrms+DJdd

UI	TJBER12	TJBER12_1
<invalid>	1.629E-10	1.636E-10



5. DDJ versus Bits: The plot DDJ versus Bits displays DDJ versus bits.



StatEye Demonstration

Location: $HPEESOF_DIR/examples/DSP/serdes_wrk

Objective

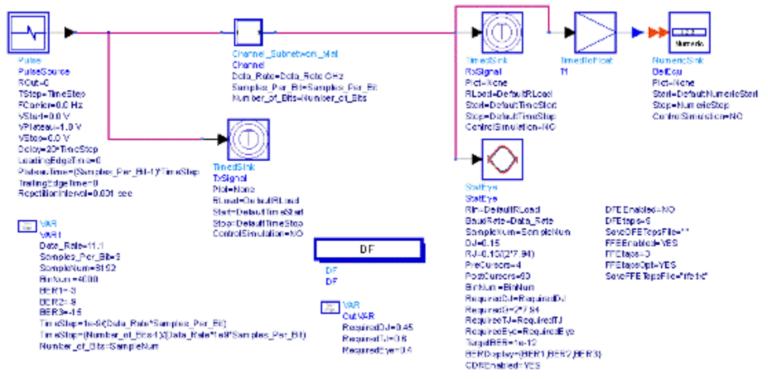
This example demonstrates the StatEye simulation capability with 2-port and 4-port channels. The Eye contour and BER bathtub curves are shown.

Setup

This example consits of an ideal pulse source, an end-to-end channel, and a receiver sink model. The 4-port channel is co-simulated with transient simulation. Different channels can be set by using different .s2p input files, or pushing into the channel model and using different .s4p input files. The BaudRate, DJ and RJ values are set to see how the signals are influenced by the system. The CDR, DFE, and FFE functions can be selected. For the ideal pulse source, the pulse width should be in range [1, 60], which is determined by Samples_Per_Bit. That means a single bit pulse source is oversampled Samples_Per_Bit times. It's not relevant to data rate, while the time step is influenced.

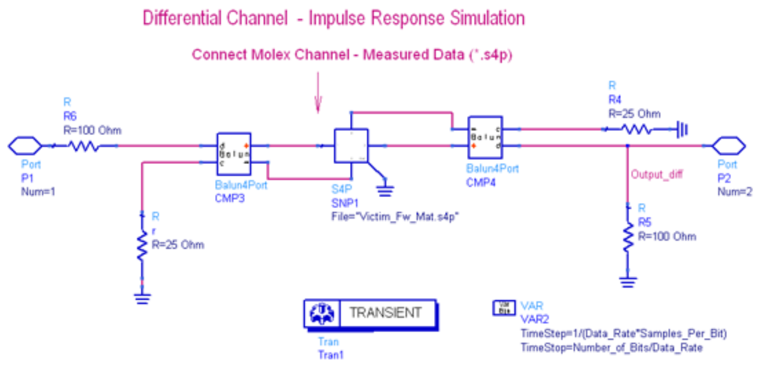
4-Port Setup

Figure: StatEye_Demo_Mat



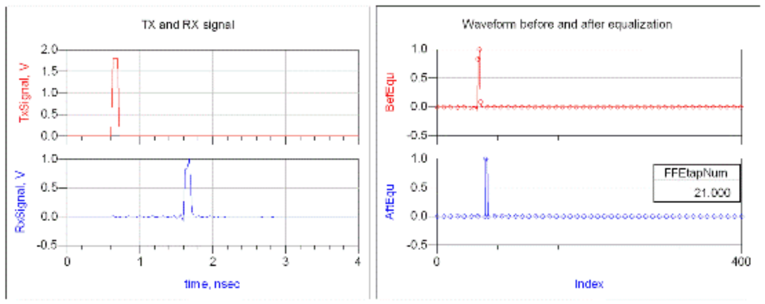
```
Pulse
PulseSource
ROut=0
TStep=TimeStep
FCarrier=0.0 Hz
VStart=0.0 V
VPlateau=1.0 V
VStop=0.0 V
Delay=20*TimeStep
LeadingEdgeTime=0
PlateauTime=(Samples_Per_Bit-1)*TimeStep
TrailingEdgeTime=0
RepetitionInterval=0.001 sec

VAR
VAR1
Data_Rate=11.1
Samples_Per_Bit=3
SampleNum=8192
BinNum=4000
BER1=-3
BER2=-9
BER3=-15
TimeStep=1e-9/(Data_Rate*Samples_Per_Bit)
TimeStop=(Number_of_Bits+1)/(Data_Rate*1e9*Samples_Per_Bit)
Number_of_Bits=SampleNum

Channel_Subnetwork_Mat
Channel
Data_Rate=Data_Rate GHz
Samples_Per_Bit=Samples_Per_Bit
Number_of_Bits=Number_of_Bits

TimedSink
TxSignal
Plot=None
RLoad=DefaultRLoad
Start=DefaultTimeStart
Stop=DefaultTimeStop
ControlSimulation=NO

DF
DF

VAR
OutVAR
RequiredDJ=0.45
RequiredTJ=0.6
RequiredEye=0.4

TimedSink
RxSignal
Plot=None
RLoad=DefaultRLoad
Start=DefaultTimeStart
Stop=DefaultTimeStop
ControlSimulation=NO

TimedToFloat
T1

NumericSink
DefEqu
Plot=None
Start=DefaultNumericStart
Stop=NumericStop
ControlSimulation=NO

StatEye
StatEye
Rin=DefaultRLoad
BaudRate=Data_Rate
SampleNum=SampleNum
DJ=0.15
RJ=0.15/(2*7.94)
PreCursors=4
PostCursors=60
BinNum=BinNum
RequiredDJ=RequiredDJ
RequiredRJ=2*7.94
RequiredTJ=RequiredTJ
RequiredEye=RequiredEye
TargetBER=1e-12
BERDisplay={BER1,BER2,BER3}
CDREnabled=YES

DFEEnabled=NO
DFEtaps=5
SaveDFETapsFile=""
FFEEnabled=YES
FFEtaps=3
FFEtapsOpt=YES
SaveFFETapsFile="ffe.txt"
```

Figure: Channel_Subnetwork_Mat

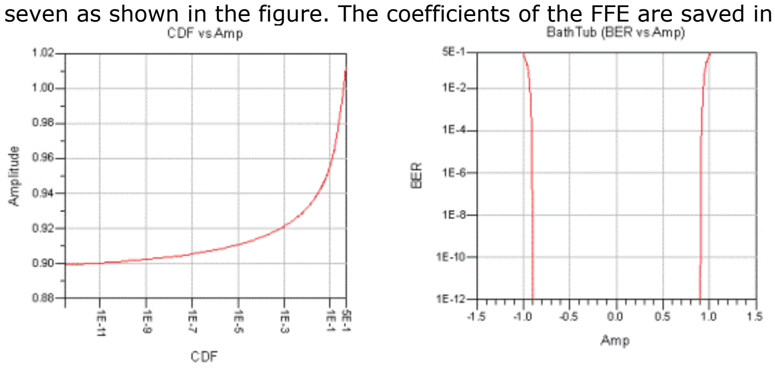

Differential Channel - Impulse Response Simulation
Connect Molex Channel - Measured Data (*.s4p)
Port P1 Num=1
R R6 R=100 Ohm
Balun4Port CMP3
R r R=25 Ohm
S4P SNP1 File="Victim_Fw_Mat.s4p"
Balun4Port CMP4
R R4 R=25 Ohm
Output_diff
R R5 R=100 Ohm
Port P2 Num=2
TRANSIENT
Tran Tran1
VAR VAR2
TimeStep=1/(Data_Rate*Samples_Per_Bit)
TimeStop=Number_of_Bits/Data_Rate

4-Port Analysis

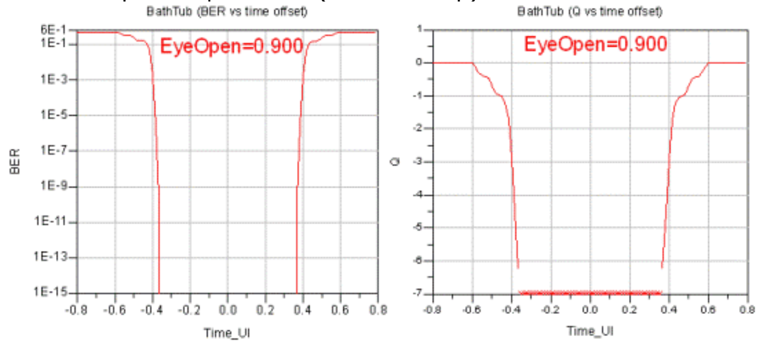
Figure: TX and RX Signal | Signal before and after equalization



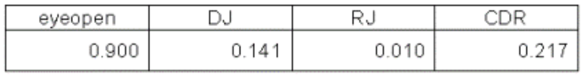
Automatically optimized functions are selected for FFE and FFE taps in this demo. As seen in the preceding image, the pulse signals before equalization are distorted to some extent. After equalization the amplitude of the pulse signal is about one and the ISI is mostly removed. The FFE taps number, which is automatically searched by the equalizer, is fifty-seven as shown in the figure. The coefficients of the FFE are saved in file ffe.txt.



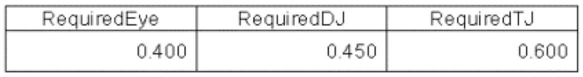
CDF vs Amplitude | Bathtub (BER vs. Amp)



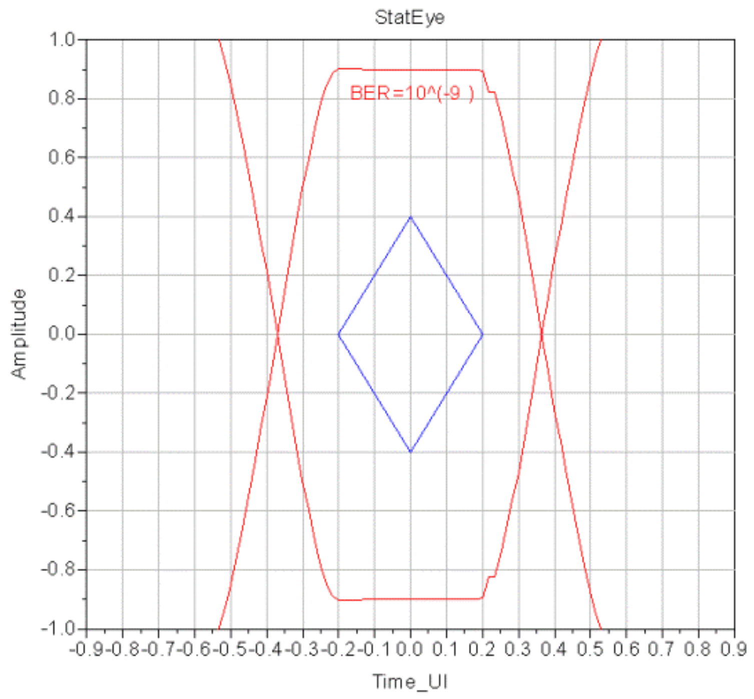
BathTub (BER and Q vs Time Offset)



eyeopen	DJ	RJ	CDR
0.900	0.141	0.010	0.217



RequiredEye	RequiredDJ	RequiredTJ
0.400	0.450	0.600



Eye Contour

Eqn Num1=Contour[0]

Eqn Data1=Contour[1::int(Num1)]

Eqn Num2=Contour[int(Num1)+1]

Eqn Data2=Contour[int(Num1)+2::int(Num1)+1+int(Num2)]

Eqn Num3=Contour[int(Num1)+1+int(Num2)+1]

Eqn Data3=Contour[int(Num1)+int(Num2)+3::int(Num1)+1+int(Num2)+1+int(Num3)]

Eqn BER1_int=int(BER1)

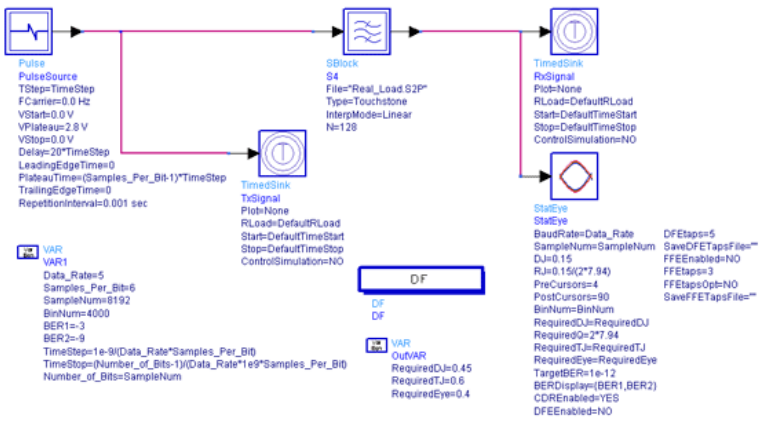
Eqn BER2_int=int(BER2)

Eqn BER3_int=int(BER3)

The Equations contain the data of contours for different BER. In this demo, contour for BER=10^(-9) is displayed. To show the contour for BER=10^(-3) or BER=10^(-15), change Data2 in the Eye Contour to Data1 or Data3.

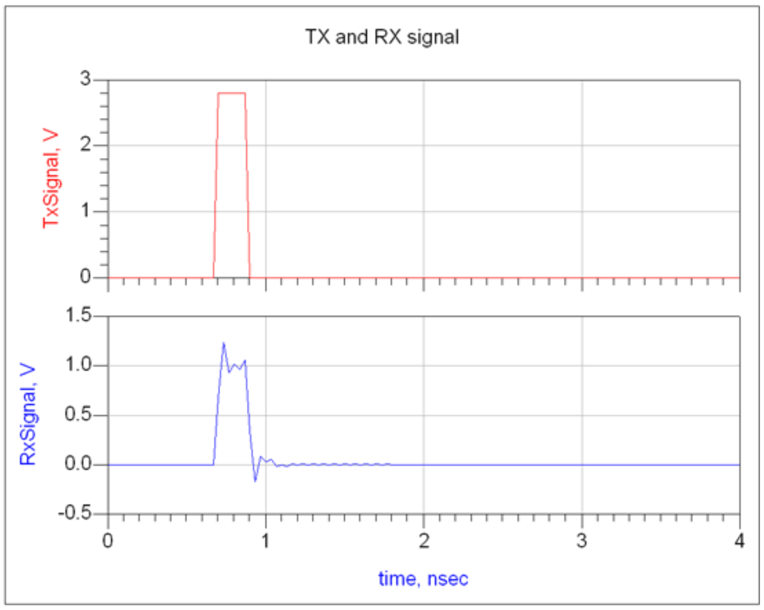
2-Port Setup

Figure: StatEye_Demo_2port

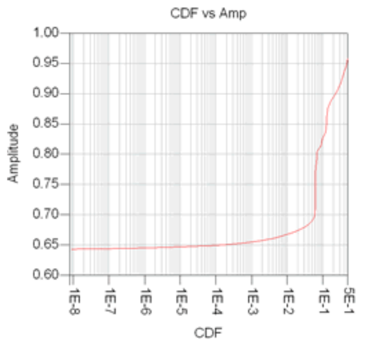


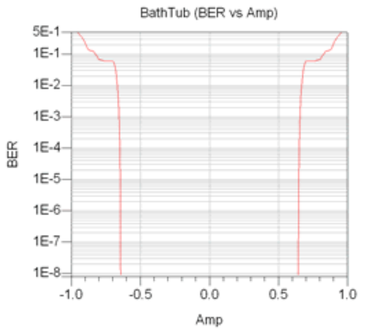
```
Pulse
PulseSource
TStep=TimeStep
FCarrier=0.0 Hz
VStart=0.0 V
VPlateau=2.8 V
VStop=0.0 V
Delay=20*TimeStep
LeadingEdgeTime=0
PlateauTime=(Samples_Per_Bit-1)*TimeStep
TrailingEdgeTime=0
RepetitionInterval=0.001 sec

SBlock
S4
File="Real_Load.S2P"
Type=Touchstone
InterpMode=Linear
N=128

TimedSink
TxSignal
Plot=None
RLoad=DefaultRLoad
Start=DefaultTimeStart
Stop=DefaultTimeStop
ControlSimulation=NO

TimedSink
RxSignal
Plot=None
RLoad=DefaultRLoad
Start=DefaultTimeStart
Stop=DefaultTimeStop
ControlSimulation=NO

VAR
VAR1
Data_Rate=5
Samples_Per_Bit=6
SampleNum=8192
BinNum=4000
BER1=-3
BER2=-9
TimeStep=1e-9/(Data_Rate*Samples_Per_Bit)
TimeStop=(Number_of_Bits-1)/(Data_Rate*1e9*Samples_Per_Bit)
Number_of_Bits=SampleNum

DF
DF

VAR
OutVAR
RequiredDJ=0.45
RequiredTJ=0.6
RequiredEye=0.4

StatEye
StatEye
BaudRate=Data_Rate              DFEtaps=5
SampleNum=SampleNum             SaveDFETapsFile=""
DJ=0.15                         FFEEnabled=NO
RJ=0.15/(2*7.94)                FFEtaps=3
PreCursors=4                    FFEtapsOpt=NO
PostCursors=90                  SaveFFETapsFile=""
BinNum=BinNum
RequiredDJ=RequiredDJ
RequiredQ=2*7.94
RequiredTJ=RequiredTJ
RequiredEye=RequiredEye
TargetBER=1e-12
BERDisplay=(BER1,BER2)
CDREnabled=YES
DFEEnabled=NO
```

2-Port Analysis

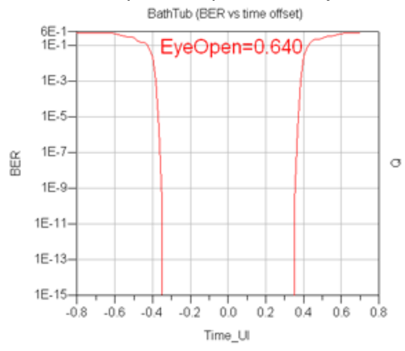


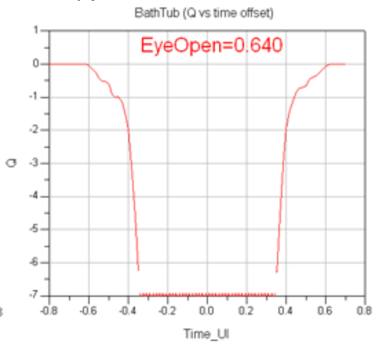
TX and RX Signal





CDF vs Amplitude | Bathtub (BER vs. Amp)

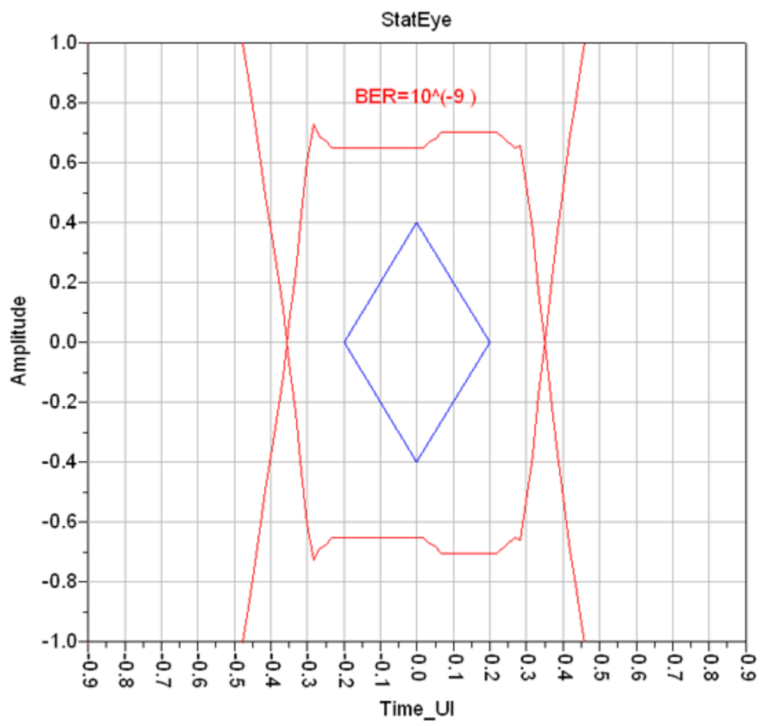




BathTub (BER and Q vs Time Offset)

eyeopen	DJ	RJ	CDR
0.640	0.170	0.010	0.300

RequiredEye	RequiredDJ	RequiredTJ
0.400	0.450	0.600



Eye Contour

EqnNum1=Contour[0]

EqnData1=Contour[1::int(Num1)]

EqnNum2=Contour[int(Num1)+1]

EqnData2=Contour[int(Num1)+2::int(Num1)+1+int(Num2)]

EqnBER1_int=int(BER1)

EqnBER2_int=int(BER2)

The Equations contain the data of contours for different BER. In this demo, contour for BER=10^(-9) is displayed. To show the contour for BER=10^(-3), change Data2 in the Eye Contour to Data1.

Notes

The diamond-shaped mask is defined by RequiredTJ and RequiredEye. Read the Data Displays to see it is inside the eye contour.

VtPRBS Features and Use Model

Location: $HPEESOF_DIR/examples/SignalIntegrity/VtPRBS_wrk

Objective

To demonstrate features and use model of VtPRBS, a multi-functional time domain voltage source for Signal Integrity applications.

Contents

The designs included in this example are best accessed in the following order:

Experiment	Description
PRBS_Mode	Demonstrates the various modes of operation for the source.
PRBS_MaximalSweep	Demonstrates maximal length PRBS sequence capability in comparison with that of ClockLFSR component.
PRBS_BitFileSweep	Demonstrates how multiple ASCII-file based bit sequences can be read and output as voltage waveforms by the source.
PRBS_Timing	Demonstrates how *BitRate*, *RiseTime*, *FallTime*, *Delay* and *TransitReference* parameters operate on defining waveform.
PRBS_EdgeShape	Demonstrates how the EdgeShape parameter can be used to define the waveform over level transitions.
PRBS_DeEmphasis	Demonstrates how the DeEmphasisMode and DeEmphasis parameters can be used to modify the waveform within prescribed limits. Also illustrates how waveform changes when *Vlow* is greater than or equal to *Vhigh*.
PRBS_EmphasisSpan	Demonstrates how the EmphasisSpan parameter can be used to modify the waveform.
PRBS_MultiTapDeEmphasis	Demonstrates how to set up the source for multiple level de-emphasis in the *Trigger*=[Internal] state.
PRBS_PAMlevels	Demonstrates how the source can be set up to provide pulse amplitude modulation (PAM) waveforms in various bit-stream encoding formats.
PRBS_Jitter	This very comprehensive design demonstrates how combinations of various parameters contributing to random and periodic jitter can alter the signal when viewed in eye diagrams or histograms at zero-crossing.
PRBS_Simulations	Demonstrates how the source behaves in DC, AC and HarmonicBalance simulations.

Each design and data display schematic contains comments and instructions necessary for understanding the basic operation of VtPRBS.

Note
For details on the functionality of each parameter of the VtPRBS component refer to *VtPRBS (Time-domain Pseudo-Random Bit Sequence Voltage Source)* (ccsrc).

PRBS_Mode

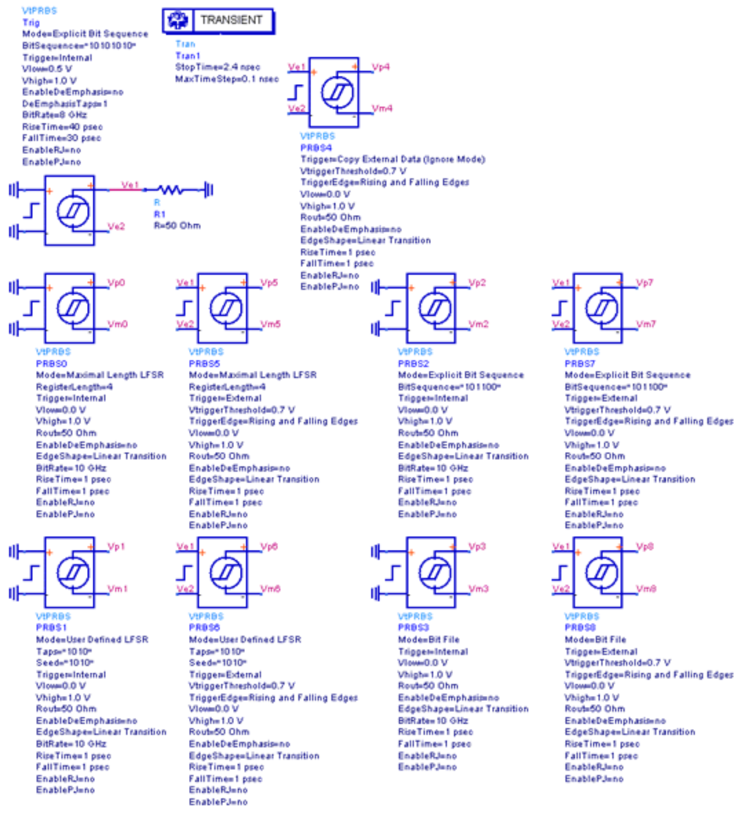
There are ten operational VtPRBS instances, of which nine are for demonstrating each of the nine modes of operation and one, named **Trig** is used to serve as an external trigger for some of the externally sensitive instances. It is set up to issue a clock signal between 0.5 V and 1.0 V at bit rate of 8 GHz with realistic rise and fall times of 40 psec and 30 psec respectively. Five instances, **PRBS4** through **PRBS8** operate based on this clock. Observe that the *VtriggerThreshold* parameter on each of these five instances is set to 0.7 V for clock detection.

Of the nine test instances, PRBS4, located at the top of the schematic, operates in the *Trigger*=[Copy External Data (Ignore Mode)]. This instance imitates the logic transitions o f the signal from **Trig**, and by definition, operates with *TriggerEdge*=[Rising and Falling Edges] to preserve data integrity. One application of this mode could be to serve as a repeater to clean up a severely distorted clock signal.

The remaining eight instances are paired for each of the four _Mode_s of operation, one set to internal trigger at 10 GHz and the other to external trigger at the 8 GHz clock from **Trig**:

Mode	Internal	External
[Maximal Length LFSR]	**PRBS0**	**PRBS5**
[User Defined LFSR]	**PRBS1**	**PRBS6**
[Explicit Bit Sequence]	**PRBS2**	**PRBS7**
[Bit File]	**PRBS3**	**PRBS8**

The explicit and file based bit sequences fed to **PRBS2**, **PRBS3**, **PRBS7**, and **PRBS8** are identical in data content. Their waveforms can be used to understand the various use modes for generating the same behavior.



```
VtPRBS
Trig
Mode=Explicit Bit Sequence
BitSequence="10101010"
Trigger=Internal
Vlow=0.5 V
Vhigh=1.0 V
EnableDeEmphasis=no
DeEmphasisTaps=1
BitRate=8 GHz
RiseTime=40 psec
FallTime=30 psec
EnableRJ=no
EnablePJ=no

Tran
Tran1
StopTime=2.4 nsec
MaxTimeStep=0.1 nsec

R
R1
R=50 Ohm

VtPRBS
PRBS4
Trigger=Copy External Data (Ignore Mode)
VtriggerThreshold=0.7 V
TriggerEdge=Rising and Falling Edges
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
EnableDeEmphasis=no
EdgeShape=Linear Transition
RiseTime=1 psec
FallTime=1 psec
EnableRJ=no
EnablePJ=no

VtPRBS
PRBS0
Mode=Maximal Length LFSR
RegisterLength=4
Trigger=Internal
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
EnableDeEmphasis=no
EdgeShape=Linear Transition
BitRate=10 GHz
RiseTime=1 psec
FallTime=1 psec
EnableRJ=no
EnablePJ=no

VtPRBS
PRBS5
Mode=Maximal Length LFSR
RegisterLength=4
Trigger=External
VtriggerThreshold=0.7 V
TriggerEdge=Rising and Falling Edges
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
EnableDeEmphasis=no
EdgeShape=Linear Transition
RiseTime=1 psec
FallTime=1 psec
EnableRJ=no
EnablePJ=no

VtPRBS
PRBS2
Mode=Explicit Bit Sequence
BitSequence="101100"
Trigger=Internal
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
EnableDeEmphasis=no
EdgeShape=Linear Transition
BitRate=10 GHz
RiseTime=1 psec
FallTime=1 psec
EnableRJ=no
EnablePJ=no

VtPRBS
PRBS7
Mode=Explicit Bit Sequence
BitSequence="101100"
Trigger=External
VtriggerThreshold=0.7 V
TriggerEdge=Rising and Falling Edges
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
EnableDeEmphasis=no
EdgeShape=Linear Transition
RiseTime=1 psec
FallTime=1 psec
EnableRJ=no
EnablePJ=no

VtPRBS
PRBS1
Mode=User Defined LFSR
Taps="1010"
Seed="1010"
Trigger=Internal
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
EnableDeEmphasis=no
EdgeShape=Linear Transition
BitRate=10 GHz
RiseTime=1 psec
FallTime=1 psec
EnableRJ=no
EnablePJ=no

VtPRBS
PRBS6
Mode=User Defined LFSR
Taps="1010"
Seed="1010"
Trigger=External
VtriggerThreshold=0.7 V
TriggerEdge=Rising and Falling Edges
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
EnableDeEmphasis=no
EdgeShape=Linear Transition
RiseTime=1 psec
FallTime=1 psec
EnableRJ=no
EnablePJ=no

VtPRBS
PRBS3
Mode=Bit File
Trigger=Internal
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
EnableDeEmphasis=no
EdgeShape=Linear Transition
BitRate=10 GHz
RiseTime=1 psec
FallTime=1 psec
EnableRJ=no
EnablePJ=no

VtPRBS
PRBS8
Mode=Bit File
Trigger=External
VtriggerThreshold=0.7 V
TriggerEdge=Rising and Falling Edges
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
EnableDeEmphasis=no
EdgeShape=Linear Transition
RiseTime=1 psec
FallTime=1 psec
EnableRJ=no
EnablePJ=no
```

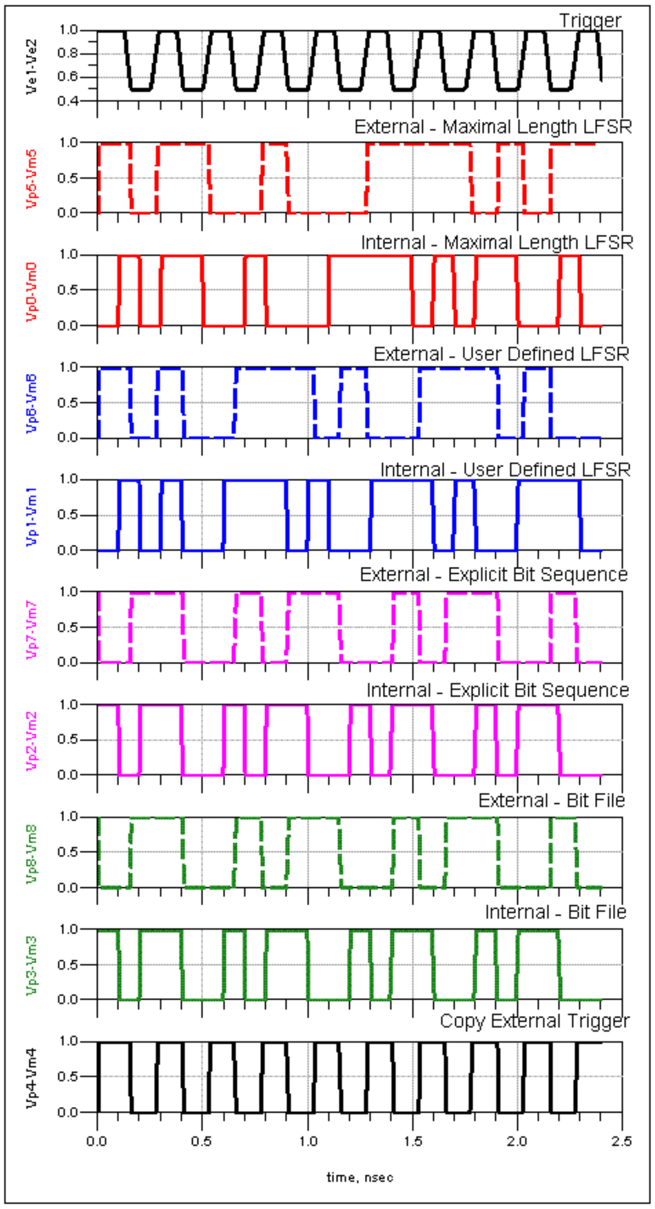
PRBS_Mode

Transient simulation of these sources results in the waveforms displayed on PRBS_Mode.dds.

The output of **PRBS4** is placed at the bottom of the column. The data content of these two signals is identical. Compare it to the clock output of **Trig** at the top of the column, by placing markers on each waveform and observe that **PRBS4** switches states exactly when **Trig** reaches 0.7 V threshold during level transitions.

The remaining eight waveforms are displayed adjacent to each other, with the dashed trace representing an externally triggered output and the solid trace representing the internally triggered version of the same data.

Note
The clock rates were intentionally staggered to be 8 GHz and 10 GHz respectively to demonstrate identical data content irrespective of bit timing. For instance, **PRBS5** issues the exact same 4-bit maximal length data stream of "010110010001111.." as **PRBS0**. The former runs with a slower external bit rate at 8 GHz whereas the latter runs at a faster internal bit rate of 10 GHz. Also note that the data content of the explicit and file based bit sequences are also identical, when clock timing is taken into account.

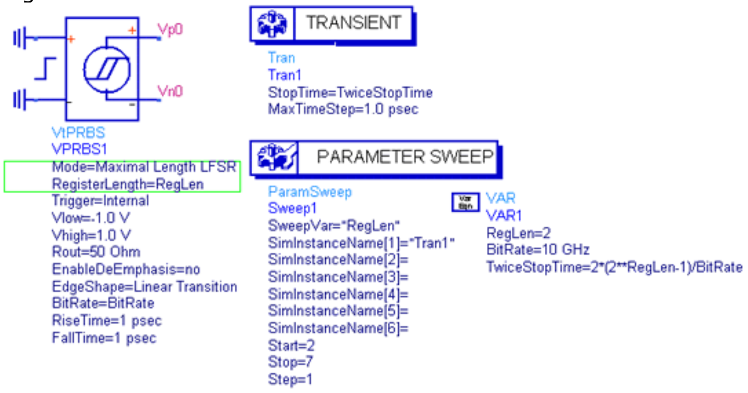


PRBS_Mode.dds

Return to Contents

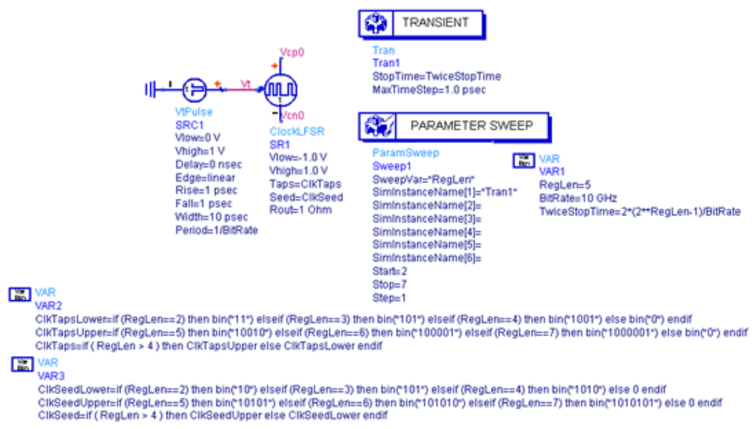
PRBS_MaximalSweep

This experiment compares the basic waveform generated by VtPRBS in maximal length LFSR mode to that generated by the *ClockLFSR* (ccsys) component for various register lengths.



```
VtPRBS
VPRBS1
Mode=Maximal Length LFSR
RegisterLength=RegLen
Trigger=Internal
Vlow=-1.0 V
Vhigh=1.0 V
Rout=50 Ohm
EnableDeEmphasis=no
EdgeShape=Linear Transition
BitRate=BitRate
RiseTime=1 psec
FallTime=1 psec

Tran
Tran1
StopTime=TwiceStopTime
MaxTimeStep=1.0 psec

ParamSweep
Sweep1
SweepVar="RegLen"
SimInstanceName[1]="Tran1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=2
Stop=7
Step=1

VAR
VAR1
RegLen=2
BitRate=10 GHz
TwiceStopTime=2*(2**RegLen-1)/BitRate
```

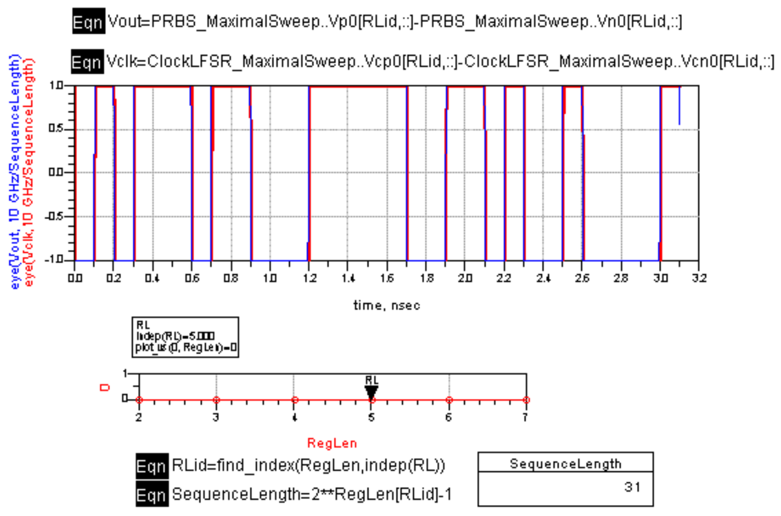
PRBS_MaximalSweep



```
VtPulse
SRC1
Vlow=0 V
Vhigh=1 V
Delay=0 nsec
Edge=linear
Rise=1 psec
Fall=1 psec
Width=10 psec
Period=1/BitRate

ClockLFSR
SR1
Vlow=-1.0 V
Vhigh=1.0 V
Taps=ClkTaps
Seed=ClkSeed
Rout=1 Ohm

Tran
Tran1
StopTime=TwiceStopTime
MaxTimeStep=1.0 psec

ParamSweep
Sweep1
SweepVar="RegLen"
SimInstanceName[1]="Tran1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=2
Stop=7
Step=1

VAR
VAR1
RegLen=5
BitRate=10 GHz
TwiceStopTime=2*(2**RegLen-1)/BitRate

VAR
VAR2
ClkTapsLower=if (RegLen==2) then bin("11") elseif (RegLen==3) then bin("101") elseif (RegLen==4) then bin("1001") else bin("0") endif
ClkTapsUpper=if (RegLen==5) then bin("10010") elseif (RegLen==6) then bin("100001") elseif (RegLen==7) then bin("1000001") else bin("0") endif
ClkTaps=if ( RegLen > 4 ) then ClkTapsUpper else ClkTapsLower endif

VAR
VAR3
ClkSeedLower=if (RegLen==2) then bin("10") elseif (RegLen==3) then bin("101") elseif (RegLen==4) then bin("1010") else 0 endif
ClkSeedUpper=if (RegLen==5) then bin("10101") elseif (RegLen==6) then bin("101010") elseif (RegLen==7) then bin("1010101") else 0 endif
ClkSeed=if ( RegLen > 4 ) then ClkSeedUpper else ClkSeedLower endif
```

ClockLFSR_MaximalSweep

For each value of register length, the transient controller is run to cover two sequence periods for the corresponding maximal length sequence. This enables the use of an eye-diagram in the data display output to overlay the two sweeps and verify that a repetable pseudo-random sequence of 2^RegLen-1 is being produced. The results of this design is compared against that of the ClockLFSR component in ClockLFSR_MaximalSweep. Observe how simpler the setting is for the VtPRBS case.



```
Eqn Vout=PRBS_MaximalSweep..Vp0[RLid,::]-PRBS_MaximalSweep..Vn0[RLid,::]
Eqn Vclk=ClockLFSR_MaximalSweep..Vcp0[RLid,::]-ClockLFSR_MaximalSweep..Vcn0[RLid,::]
Eqn RLid=find_index(RegLen,indep(RL))
Eqn SequenceLength=2**RegLen[RLid]-1
```

PRBS_MaximalSweep.dds

Using the slider to set register length in the display schematic observe how the ClockLFSR and VtPRBS data sequences contain the same data. Minor differences such as edge shape and exact transition times exist because the start of rising and falling edges is computed differently for the two source models.

Return to Contents

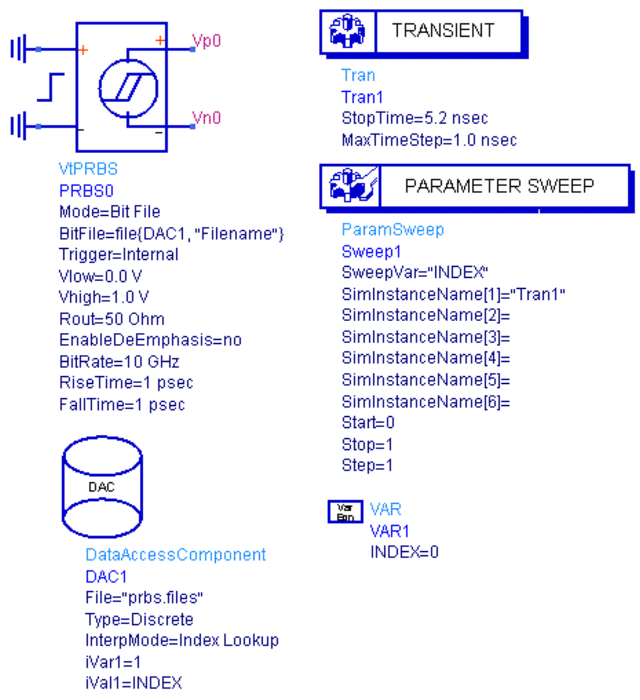
PRBS_BitFileSweep

This experiment demonstrates how to sweep through a set of ASCII bit files to generate PRBS behavior. A *DataAccessComponent* (ccsim) (DAC) is used to read a table of files named "prbs.files". The string containing the name of the accessed file is passed on to the *BitFile* parameter of VtPRBS, which then performs the reading and data importing. Note that the DAC is used with *Type*=[Discrete] and *Interp*=[Index Lookup] mode. The *iVar1* variable should always be set to 1 to indicate that the INDEX variable is the key to the 1st data column of the lookup table. The INDEX variable itself is swept externally starting with value 0.

The contents of "prbs.files" is as follows:

```
begin dscrdata
```

```
% Filenumber Filename
1 prbs.txt
2 Txt.prbs
end dscrdata
```

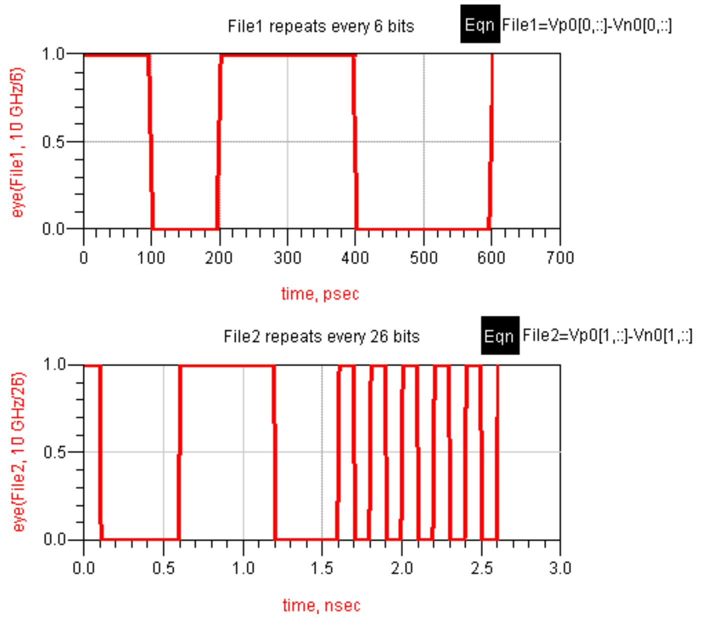


PRBS_BitFileSweep

Verify the contents of the two bit files prior to simulation:

Filename	Bit Contents	Sequence Length
prbs.txt	101100	6 bits
Txt.prbs	10000011111100001010101010	26 bits

The data display shows eye diagrams captured over the sequence length of each file. Each eye diagram shows the repeated nature of the bit sequence for the file of interest.



PRBS_BitFileSweep.dds

Return to Contents

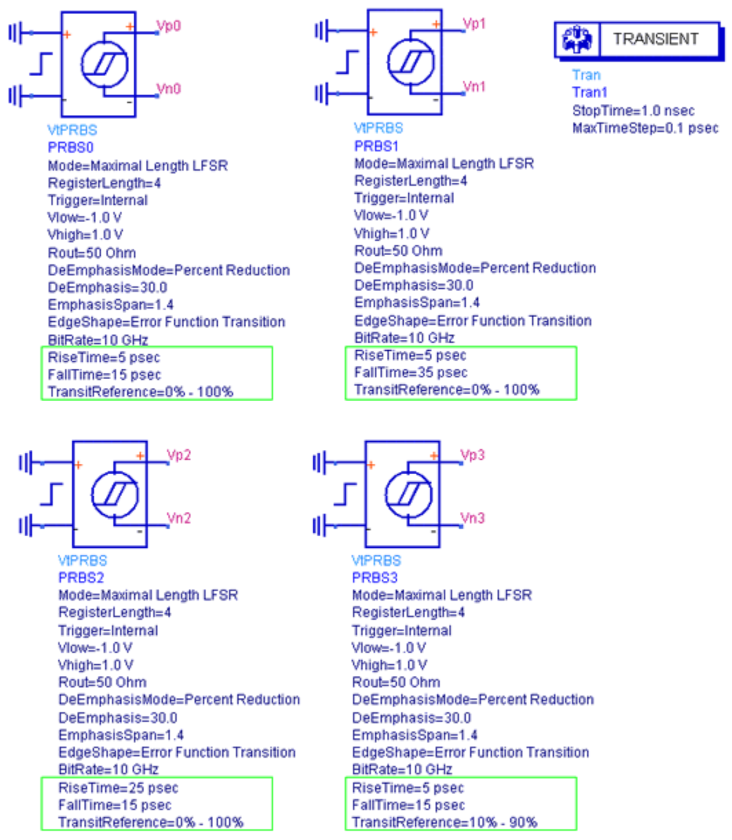
PRBS_Timing

The four instances of VtPRBS in this experiment all operate in internal trigger mode and generate the same maximal length sequence and use the same *EdgeShape* for conformity. Their un-jittered timing parameters are set differently to showcase the functionality of *RiseTime*, *FallTime* and *TransitReference* parameters.

PRBS0 is used as a control case where rise and fall times are set to explicit values of 5 psec and 15 psec respectively. Note that *TransitReference* is set to [0% - 100%] to indicate that effective transition times are to be identical to specified transition times.

Relative to **PRBS0**, the instances **PRBS1** and **PRBS2** have different settings for fall and rise times respectively, while *TransitReference* remains the same.

In **PRBS3**, rise and fall times are maintained as that of the control instance **PRBS0**, but *TransitReference* is set to the [10% - 90%] option indicating that effective rise and fall times should be internally computed based on the *EdgeShape* specification.



VtPRBS
PRBS0
Mode=Maximal Length LFSR
RegisterLength=4
Trigger=Internal
Vlow=-1.0 V
Vhigh=1.0 V
Rout=50 Ohm
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
EmphasisSpan=1.4
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=5 psec
FallTime=15 psec
TransitReference=0% - 100%

VtPRBS
PRBS1
Mode=Maximal Length LFSR
RegisterLength=4
Trigger=Internal
Vlow=-1.0 V
Vhigh=1.0 V
Rout=50 Ohm
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
EmphasisSpan=1.4
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=5 psec
FallTime=35 psec
TransitReference=0% - 100%

Tran
Tran1
StopTime=1.0 nsec
MaxTimeStep=0.1 psec

VtPRBS
PRBS2
Mode=Maximal Length LFSR
RegisterLength=4
Trigger=Internal
Vlow=-1.0 V
Vhigh=1.0 V
Rout=50 Ohm
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
EmphasisSpan=1.4
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=25 psec
FallTime=15 psec
TransitReference=0% - 100%

VtPRBS
PRBS3
Mode=Maximal Length LFSR
RegisterLength=4
Trigger=Internal
Vlow=-1.0 V
Vhigh=1.0 V
Rout=50 Ohm
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
EmphasisSpan=1.4
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=5 psec
FallTime=15 psec
TransitReference=10% - 90%

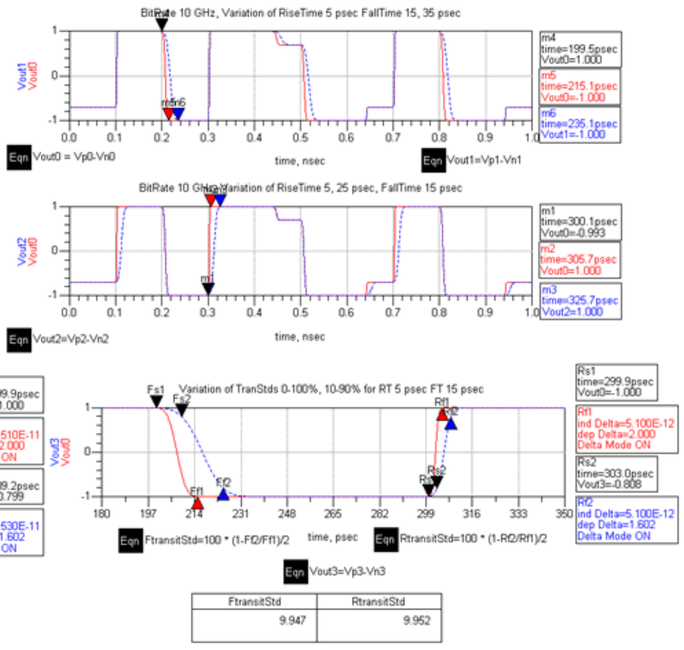
PRBS_Timing

In the data display, the first two plots show comparisons of the control waveform from **PRBS0** with those generated by **PRBS1** and **PRBS2**. In each case, note that the transition interval starts at the exact same point as the control waveform. This indicates the bit boundary for the first bit that initiates the level change. The duration of transition within this bit interval is determined by the rise or fall time specified on the component. Although simplistic, this bit boundary definition highlights a key feature of VtPRBS, namely, that it is a causal source. No *a priori* knowledge of the start of a level transition is used to initial roll-offs earlier than the start of the bit interval. The zero-crossing between levels occurs when the *EdgeShape* and timing parameters permit. The causal nature of the VtPRBS component allows it to optionally follow external trigger signals as closely as it would follow its own internal clock for all types of data input modes. It also enables jitter to be applied in a definitive manner to the unambiguously defined bit boundary.

In the following display note that *Vout0* and *Vout1* both start the 1 to 0 transition between the 2nd and 3rd bit at 200 psec for a 10 GHz signal. The longer fall time for *Vout* causes its zero-crossing to lag by half the difference in fall time and the transition ends at precisely 20 psec later than that of *Vout0*. Likewise note the delay in *Vout2* reaching level 1 from 0 although both traces start at the bit boundary of 300 psec.

The third and final plot shows a comparison of *Vout0* and *Vout3*. The latter waveform has both effective rise and fall times set to be larger than the former because of the non-trivial

TransitReference setting of **PRBS3**. This plot uses marhers and equations to recover the effective value of transition reference exhibited by VtPRBS. Note that 9.947% and 9.952% fall and rise time transition references are recovered when the original specification was 10% from each end along the time axis.



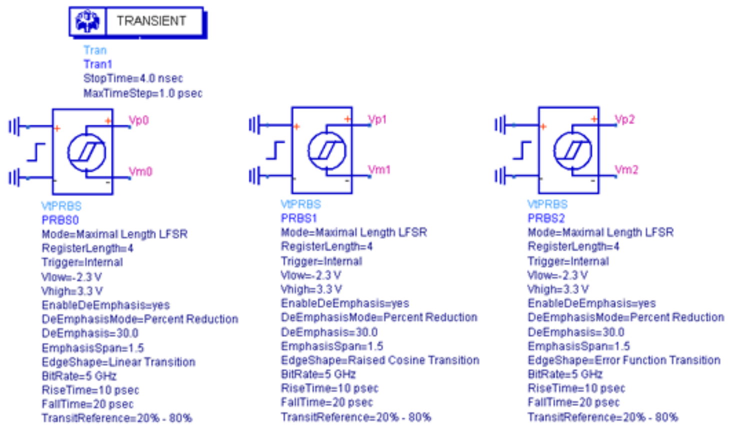
FtransitStd	RtransitStd
9.947	9.952

PRBS_Timing.dds

Return to Contents

PRBS_EdgeShape

VtPRBS supports three types of analytical edge shapes that are commonly used in Signal Integrity applocations - linear, raised cosine and error function. These can be selected from the *EdgeShape* parameter. All three instances have *TransitReference* set to [20% to 80%] and have identical values of non-trivial rise and fall times.



Tran
Tran1
StopTime=4.0 nsec
MaxTimeStep=1.0 psec

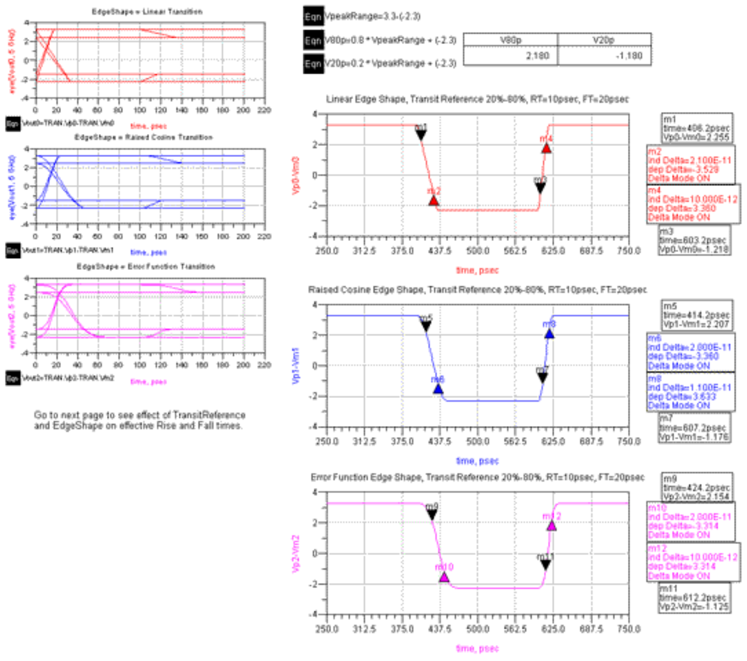
VtPRBS
PRBS0
Mode=Maximal Length LFSR
RegisterLength=4
Trigger=Internal
Vlow=-2.3 V
Vhigh=3.3 V
EnableDeEmphasis=yes
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
EmphasisSpan=1.5
EdgeShape=Linear Transition
BitRate=5 GHz
RiseTime=10 psec
FallTime=20 psec
TransitReference=20% - 80%

VtPRBS
PRBS1
Mode=Maximal Length LFSR
RegisterLength=4
Trigger=Internal
Vlow=-2.3 V
Vhigh=3.3 V
EnableDeEmphasis=yes
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
EmphasisSpan=1.5
EdgeShape=Raised Cosine Transition
BitRate=5 GHz
RiseTime=10 psec
FallTime=20 psec
TransitReference=20% - 80%

VtPRBS
PRBS2
Mode=Maximal Length LFSR
RegisterLength=4
Trigger=Internal
Vlow=-2.3 V
Vhigh=3.3 V
EnableDeEmphasis=yes
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
EmphasisSpan=1.5
EdgeShape=Error Function Transition
BitRate=5 GHz
RiseTime=10 psec
FallTime=20 psec
TransitReference=20% - 80%

PRBS_EdgeShape

The data display is spread over two pages. The eye-diagrams on the first page provide a visual assessment of the different transition shapes that are possible with this component. The second display page shows how the value provided by *TransitReference* influences the effective rise and fall times based on the choice of *EdgeShape*.


Eqn VpeakRange=3.3-(-2.3)
Eqn V80p=0.8 * VpeakRange + (-2.3)
Eqn V20p=0.2 * VpeakRange + (-2.3)

V80p	V20p
2.180	-1.180

Eqn Vout0=TRAN.Vp0-TRAN.Vm0
Eqn Vout1=TRAN.Vp1-TRAN.Vm1
Eqn Vout2=TRAN.Vp2-TRAN.Vm2

Go to next page to see effect of TransitReference and EdgeShape on effective Rise and Fall times.

```
m1
time=406.2psec
Vp0-Vm0=2.255

m2
ind Delta=2.100E-11
dep Delta=-3.529
Delta Mode ON

m4
ind Delta=10.000E-12
dep Delta=3.360
Delta Mode ON

m3
time=603.2psec
Vp0-Vm0=-1.218

m5
time=414.2psec
Vp1-Vm1=2.207

m6
ind Delta=2.000E-11
dep Delta=-3.360
Delta Mode ON

m8
ind Delta=1.100E-11
dep Delta=3.633
Delta Mode ON

m7
time=607.2psec
Vp1-Vm1=-1.176

m9
time=424.2psec
Vp2-Vm2=2.154

m10
ind Delta=2.000E-11
dep Delta=-3.314
Delta Mode ON

m12
ind Delta=10.000E-12
dep Delta=3.314
Delta Mode ON

m11
time=612.2psec
Vp2-Vm2=-1.125
```

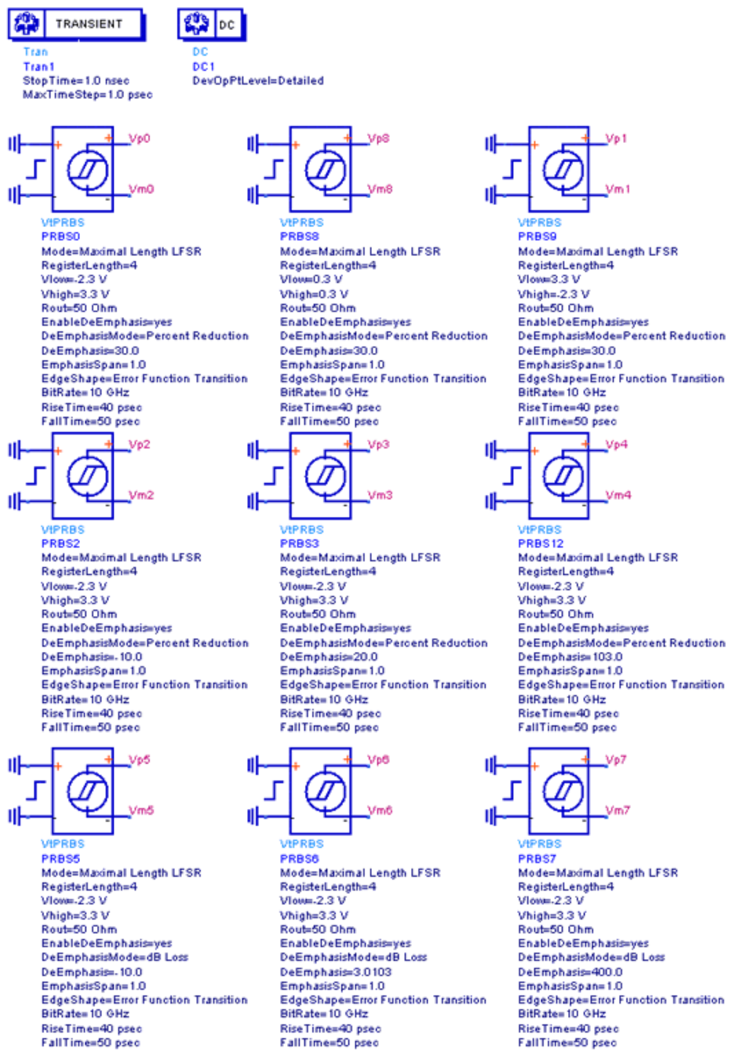
EdgeShape = Linear Transition
eye(Vout0, 5 GHz)
time, psec
EdgeShape = Raised Cosine Transition
eye(Vout1, 5 GHz)
EdgeShape = Error Function Transition
eye(Vout2, 5 GHz)
Linear Edge Shape, Transit Reference 20%-80%, RT=10psec, FT=20psec
Vp0-Vm0
Raised Cosine Edge Shape, Transit Reference 20%-80%, RT=10psec, FT=20psec
Vp1-Vm1
Error Function Edge Shape, Transit Reference 20%-80%, RT=10psec, FT=20psec
Vp2-Vm2
m1
m2
m3
m4
m5
m6
m7
m8
m9
m10
m11
m12

PRBS_EdgeShape.dds - Combined display of "Eye Diagrams" and "Transit Reference" pages.

Return to Contents

PRBS_DeEmphasis

This is the first of three experiments that highlight the de-emphasis features of VtPRBS. There are nine instances of the component on the schematic.



Tran
Tran1
StopTime=1.0 nsec
MaxTimeStep=1.0 psec

DC
DC1
DevOpPtLevel=Detailed

VtPRBS
PRBS0
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-2.3 V
Vhigh=3.3 V
Rout=50 Ohm
EnableDeEmphasis=yes
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
EmphasisSpan=1.0
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=40 psec
FallTime=50 psec

VtPRBS
PRBS8
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=0.3 V
Vhigh=0.3 V
Rout=50 Ohm
EnableDeEmphasis=yes
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
EmphasisSpan=1.0
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=40 psec
FallTime=50 psec

VtPRBS
PRBS9
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=3.3 V
Vhigh=-2.3 V
Rout=50 Ohm
EnableDeEmphasis=yes
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
EmphasisSpan=1.0
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=40 psec
FallTime=50 psec

VtPRBS
PRBS2
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-2.3 V
Vhigh=3.3 V
Rout=50 Ohm
EnableDeEmphasis=yes
DeEmphasisMode=Percent Reduction
DeEmphasis=-10.0
EmphasisSpan=1.0
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=40 psec
FallTime=50 psec

VtPRBS
PRBS3
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-2.3 V
Vhigh=3.3 V
Rout=50 Ohm
EnableDeEmphasis=yes
DeEmphasisMode=Percent Reduction
DeEmphasis=20.0
EmphasisSpan=1.0
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=40 psec
FallTime=50 psec

VtPRBS
PRBS12
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-2.3 V
Vhigh=3.3 V
Rout=50 Ohm
EnableDeEmphasis=yes
DeEmphasisMode=Percent Reduction
DeEmphasis=103.0
EmphasisSpan=1.0
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=40 psec
FallTime=50 psec

VtPRBS
PRBS5
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-2.3 V
Vhigh=3.3 V
Rout=50 Ohm
EnableDeEmphasis=yes
DeEmphasisMode=dB Loss
DeEmphasis=-10.0
EmphasisSpan=1.0
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=40 psec
FallTime=50 psec

VtPRBS
PRBS6
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-2.3 V
Vhigh=3.3 V
Rout=50 Ohm
EnableDeEmphasis=yes
DeEmphasisMode=dB Loss
DeEmphasis=3.0103
EmphasisSpan=1.0
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=40 psec
FallTime=50 psec

VtPRBS
PRBS7
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-2.3 V
Vhigh=3.3 V
Rout=50 Ohm
EnableDeEmphasis=yes
DeEmphasisMode=dB Loss
DeEmphasis=400.0
EmphasisSpan=1.0
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=40 psec
FallTime=50 psec

PRBS_DeEmphasis

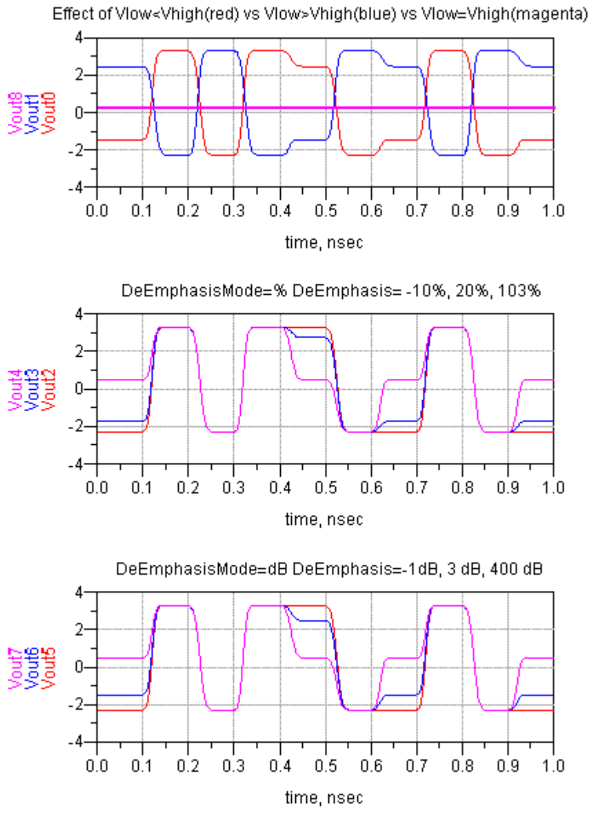
The first row consists of three instances which establish the behavior of peak voltage levels for logic 0 and logic 1. The parameter *Vlow* defines logic 0 and *Vhigh* defines logic 1. Using **PRBS0** as a control case, **PRBS8** and **PRBS9** help establish the fact that if *Vlow* is set to be a non-intuitive value such as equal to or greater than *Vhigh*, then the waveform flatlines or inverts.

The second and third rows takes a closer look at the settings of parameters *DeEmphasisMode* and *DeEmphasis* when this feature is enabled and *EmphasisSpan* is set to a non-zero value. In each case, verify that the undisplayed parameter of *DeEmphasisTaps* is set to 1. Setting it to a higher value will result in a more complicated waveform than is being discussed here.

1. The three instances **PRBS2**, **PRBS3** and **PRBS4** all have *DeEmphasisMode*, the definition of de-emphasis, set to [Percent Reduction]. In these cases, with zero-crossing remaining at the arithmetic average : 0.5 * (*Vhigh* + *Vlow*), the waveform is shifted from either peak towards the zero-crossing after an interval of *EmphasisSpan* * / BitRate has expired. The shift itself depends on the numeric value entered at the *DeEmphasis* parameter as follows:
   1. **PRBS2** - With *DeEmphasis* = -10, which is below the expected range of [0,100]%, the
      behavior adopted is that of the lower limit of 0% de-emphasis. Effectively the waveform is not de-emphasized at all.
   2. **PRBS3** - *DeEmphasis* = 20 implies that the dynamic range of the de-emphasized waveform i.e. the voltage difference between shelf values of logic 0 and logic 1 will be 10% reduced relative to the dynamic range established by the peak voltage values *Vlow* and *Vhigh*.
   3. **PRBS4** - With *DeEmphasis = 103, which is above the expected range of [0,100]%, the behavior adopted is that of the upper limit of 100% emphasis. In*

reality, the limit is set to 99.9999% for practical considerations of preserving waveform continuity. Full de-emphasization means that the waveform touches the zero-crossing line after the _EmphasisSpan duration has expired.

2. The three instances **PRBS5**, **PRBS6** and **PRBS7** all have *DeEmphasisMode*, the definition of de-emphasis, set to [dB Loss]. In each case, the voltage level of the shelf is computed to be lowered or raised with respect to the peak logic 1 or 0 by as many dB as are specified at the *DeEmphasis* parameter as follows:
   1. **PRBS5** - With *DeEmphasis* = -10, which is below the expected range of [0,∞) dB, the
      behavior adopted is that of the lower limit of 0 dB de-emphasis. Effectively the waveform is not de-emphasized at all.
   2. **PRBS6** - *DeEmphasis* = 3.0103 dB implies that the logic 1 shelf value is half the voltage level of the logic 1 peak value, both values being relative to the zero-crossing. The logic 0 peak and shelf values are set symmetrically with respect to the logic 1 values.
   3. **PRBS7** - While *DeEmphasis* = 400 is within the numeric range of [0,∞) dB, for all practical purposes it indicates that the shelf values of logic 1 and 0 approach the zero-crossing level.

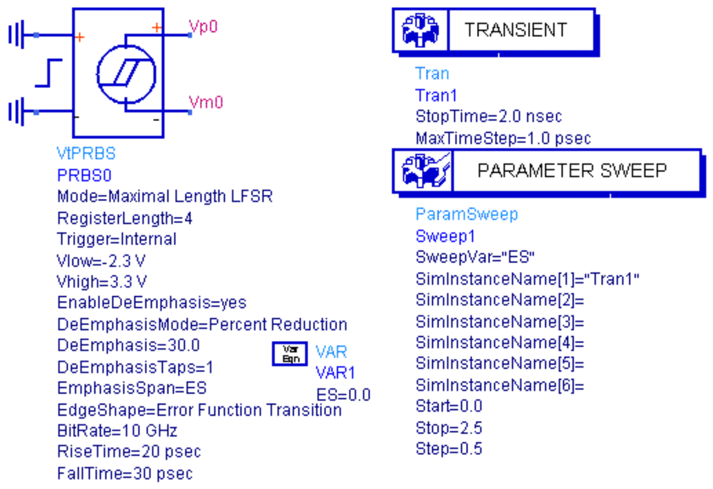


PRBS_DeEmphasis.dds

Return to Contents

PRBS_EmphasisSpan

Use of the parameter *EmphasisSpan* is demonstrated in this experiment. The effective time duration of maintaining signal level when de-emphasis is enabled and active, is computed as *EmphasisSpan* / *BitRate*. The peak (or intermediate shelf for *DeEmphasisTaps* > 1) value of a waveform is allowed to persist for this duration before a voltage level change occurs without any change to the logic level. When *EmphasisSpan* is set less than or equal to or is in excess of 32 bit intervals, shelf value is not enforced. The reason for 32-bit upper limit on *EmphasisSpan* is that VtPRBS is enabled primarily for a maximum register length of 32-bits where the largest contiguos sequence of 1's will be 32.



VtPRBS
PRBS0
Mode=Maximal Length LFSR
RegisterLength=4
Trigger=Internal
Vlow=-2.3 V
Vhigh=3.3 V
EnableDeEmphasis=yes
DeEmphasisMode=Percent Reduction
DeEmphasis=30.0
DeEmphasisTaps=1
EmphasisSpan=ES
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=20 psec
FallTime=30 psec

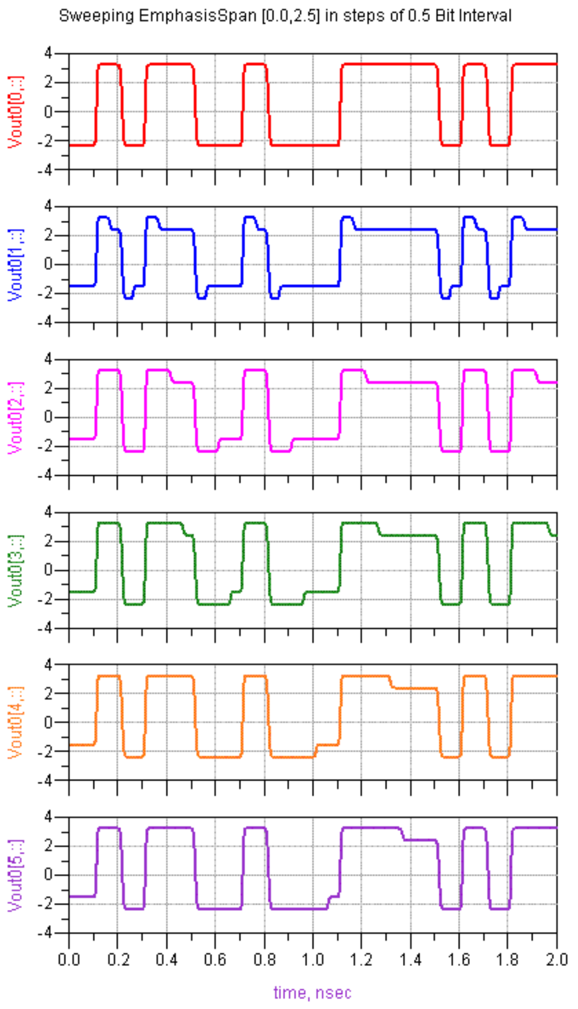
VAR
VAR1
ES=0.0

Tran
Tran1
StopTime=2.0 nsec
MaxTimeStep=1.0 psec

ParamSweep
Sweep1
SweepVar="ES"
SimInstanceName[1]="Tran1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=0.0
Stop=2.5
Step=0.5

PRBS_EmphasisSpan

EmphasisSpan is swept in intervals of 0.5 bit intervals from 0 though 2.5 UI resulting in the following plots.

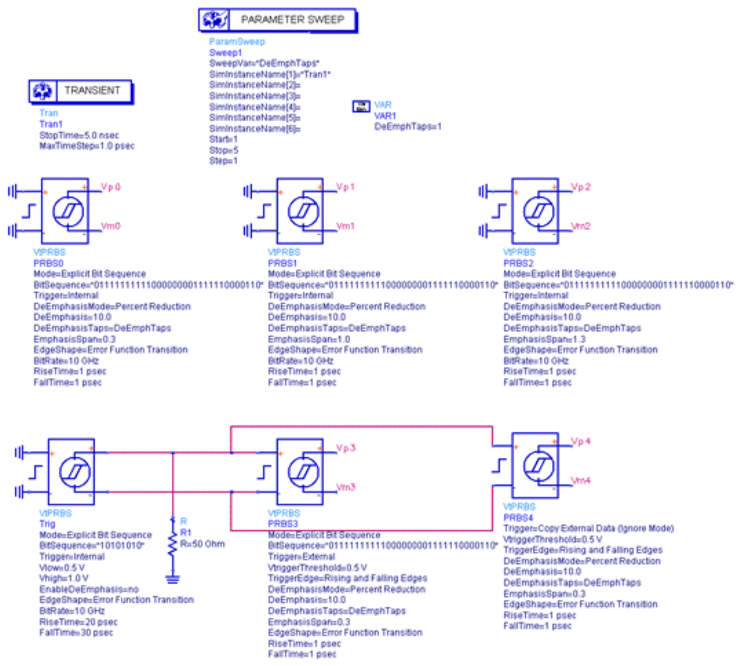


PRBS_EmphasisSpan.dds

Return to Contents

PRBS_MultiTapDeEmphasis

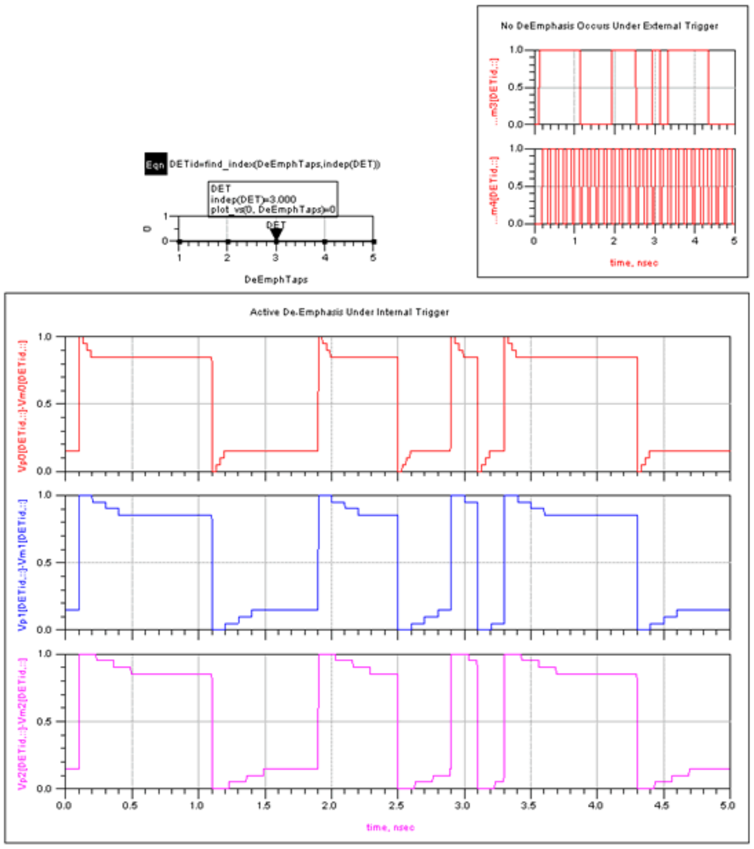
VtPRBS allows the setting of *DeEmphasisTaps* to enable progessive de-emphasis of waveform while the binary logic level is sustained. In this experiment, with *EmphasisSpan* set to three different values, in **PRBS0** through **PRBS2**, the number of levels are swept from 1 through 5. The waveforms generated by these three internally triggered sources show how for each specification of de-emphasis taps, the waveform is stepped towards the zero-crossing after a period of *EmphasisSpan* has been spent at each level.



```
Tran
Tran1
StopTime=5.0 nsec
MaxTimeStep=1.0 psec

ParamSweep
Sweep1
SweepVar="DeEmphTaps"
SimInstanceName[1]="Tran1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1
Stop=5
Step=1

VAR
VAR1
DeEmphTaps=1

VtPRBS
PRBS0
Mode=Explicit Bit Sequence
BitSequence="011111111110000000001111110000110"
Trigger=Internal
DeEmphasisMode=Percent Reduction
DeEmphasis=10.0
DeEmphasisTaps=DeEmphTaps
EmphasisSpan=0.3
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=1 psec
FallTime=1 psec

VtPRBS
PRBS1
Mode=Explicit Bit Sequence
BitSequence="011111111110000000001111110000110"
Trigger=Internal
DeEmphasisMode=Percent Reduction
DeEmphasis=10.0
DeEmphasisTaps=DeEmphTaps
EmphasisSpan=1.0
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=1 psec
FallTime=1 psec

VtPRBS
PRBS2
Mode=Explicit Bit Sequence
BitSequence="011111111110000000001111110000110"
Trigger=Internal
DeEmphasisMode=Percent Reduction
DeEmphasis=10.0
DeEmphasisTaps=DeEmphTaps
EmphasisSpan=1.3
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=1 psec
FallTime=1 psec

VtPRBS
Trig
Mode=Explicit Bit Sequence
BitSequence="10101010"
Trigger=Internal
Vlow=0.5 V
Vhigh=1.0 V
EnableDeEmphasis=no
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=20 psec
FallTime=30 psec

R
R1
R=50 Ohm

VtPRBS
PRBS3
Mode=Explicit Bit Sequence
BitSequence="011111111110000000001111110000110"
Trigger=External
VtriggerThreshold=0.5 V
TriggerEdge=Rising and Falling Edges
DeEmphasisMode=Percent Reduction
DeEmphasis=10.0
DeEmphasisTaps=DeEmphTaps
EmphasisSpan=0.3
EdgeShape=Error Function Transition
RiseTime=1 psec
FallTime=1 psec

VtPRBS
PRBS4
Trigger=Copy External Data (Ignore Mode)
VtriggerThreshold=0.5 V
TriggerEdge=Rising and Falling Edges
DeEmphasisMode=Percent Reduction
DeEmphasis=10.0
DeEmphasisTaps=DeEmphTaps
EmphasisSpan=0.3
EdgeShape=Error Function Transition
RiseTime=1 psec
FallTime=1 psec
```

PRBS_MultiTapDeEmphasis

The **PRBS3** and **PRBS4** instances are used to demonstrate that de-emphasis, and therefore, multi-tap de-emphasis is disabled when external trigger is applied. The reason for disabling de-emphasis for external trigger is because of *EmphasisSpan*. This value is expressed in terms of bit intervals. For an external control, the value of this interval is not known to the follower VtPRBS instance. Hence, the duration of any level cannot be pre-determined and multiple levels cannot be supported.

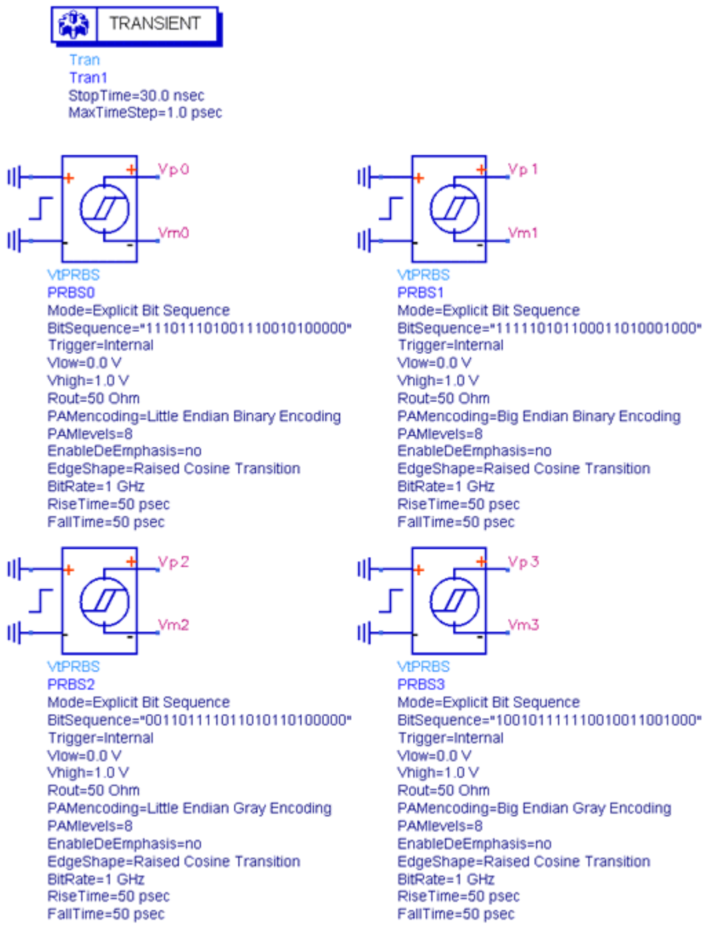


PRBS_MultiTapDeEmphasis.dds

Return to Contents

PRBS_PAMlevels

When VtPRBS operates in the Pulse Amplitude Modulation (PAM) mode, it suspends de-emphasis behavior and follows values entered in the *PAMencoding* and *PAMlevels* parameters. The former allows a choice of four conversions between binary logic and multi-level logic. PAM levels are distributed evenly between *Vlow* and *Vhigh*.



Tran
Tran1
StopTime=30.0 nsec
MaxTimeStep=1.0 psec

VtPRBS
PRBS0
Mode=Explicit Bit Sequence
BitSequence="111011101001110010100000"
Trigger=Internal
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
PAMencoding=Little Endian Binary Encoding
PAMlevels=8
EnableDeEmphasis=no
EdgeShape=Raised Cosine Transition
BitRate=1 GHz
RiseTime=50 psec
FallTime=50 psec

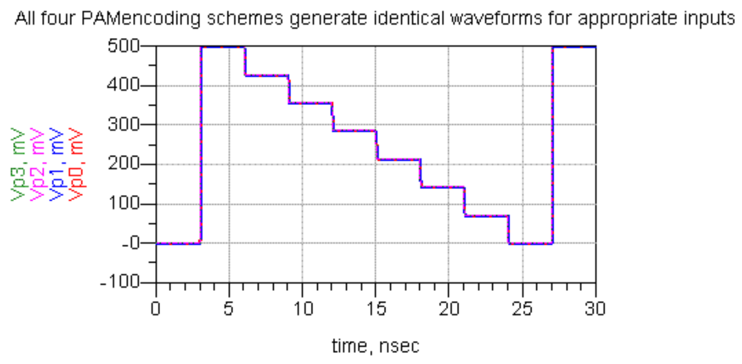
VtPRBS
PRBS1
Mode=Explicit Bit Sequence
BitSequence="111110101100011010001000"
Trigger=Internal
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
PAMencoding=Big Endian Binary Encoding
PAMlevels=8
EnableDeEmphasis=no
EdgeShape=Raised Cosine Transition
BitRate=1 GHz
RiseTime=50 psec
FallTime=50 psec

VtPRBS
PRBS2
Mode=Explicit Bit Sequence
BitSequence="001101111011010110100000"
Trigger=Internal
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
PAMencoding=Little Endian Gray Encoding
PAMlevels=8
EnableDeEmphasis=no
EdgeShape=Raised Cosine Transition
BitRate=1 GHz
RiseTime=50 psec
FallTime=50 psec

VtPRBS
PRBS3
Mode=Explicit Bit Sequence
BitSequence="100101111110010011001000"
Trigger=Internal
Vlow=0.0 V
Vhigh=1.0 V
Rout=50 Ohm
PAMencoding=Big Endian Gray Encoding
PAMlevels=8
EnableDeEmphasis=no
EdgeShape=Raised Cosine Transition
BitRate=1 GHz
RiseTime=50 psec
FallTime=50 psec

PRBS_PAMlevels

In this experiment 8-ary PAM is set up for each of the four encoding types. Note that in each case, the explicit bit sequence is different. They are designed so that the behavior of each VtPRBS instance for the specified encoding scheme generates identical waveforms at the output.



PRBS_PAMlevels.dds

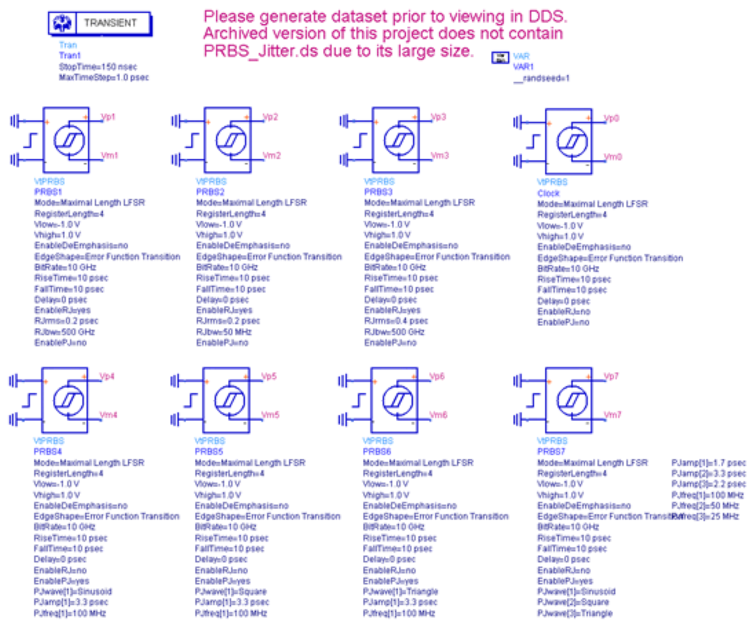
[Return to Contents](#)

PRBS_Jitter

VtPRBS is by far the most versatile single source model in ADS that generates time-variation of zero-crossings of a digital waveform, also known as jitter. For details on the formulation of random and periodic jitter supported by this component, refer to *VtPRBS (Time-domain Pseudo-Random Bit Sequence Voltage Source)* (ccsrc).

In this experiment, eight instances of VtPRBS are active.

1. **Clock** is an unjittered control source that operates at 10 GHz with 10 psec rise and fall times. The waveform range is [-1.0, 1.0]V. De-emphasis is disabled for complete focus on jitter behavior.
2. **PRBS1** is expected to exhibit random jitter (gaussian distribution around bit boundary) with a standard deviation of 0.2 psec and measuement bandwidth of 500 MHz.
3. **PRBS2** keeps the same standard deviation as **PRBS1** but decreases measurement bandwidth to 50 MHz.
4. **PRBS3** keeps the same measurement bandwidth as **PRBS1** but increases standard deviation to 0.4 psec.
5. **PRBS4**, **PRBS5** and **PRBS6** each exhibit, single mode, pure periodic jitter of sinusoidal, square and triangular types respectively. Note how the *PJwave* variable is set in each case. The *PJamp* and *PJfreq* values are set to 3.3 psec and 100 MHz in each case.
6. **PRBS8** demonstrates the multi-modal periodic jitter capabilities of VtPRBS. Each of the three modes on this instance are set to a differnt type of periodic waveform, simulating conditions of three different types of interferences that affect the waveform timing.

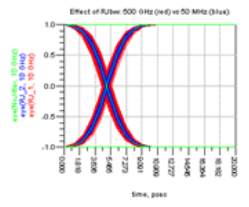


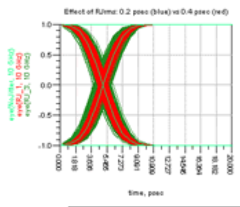
Please generate dataset prior to viewing in DDS.
Archived version of this project does not contain
PRBS_Jitter.ds due to its large size.

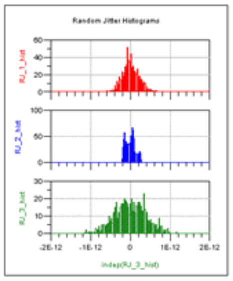
```
Tran
Tran1
StopTime=150 nsec
MaxTimeStep=1.0 psec

VAR
VAR1
__randseed=1

VtPRBS
PRBS1
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-1.0 V
Vhigh=1.0 V
EnableDeEmphasis=no
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=10 psec
FallTime=10 psec
Delay=0 psec
EnableRJ=yes
RJrms=0.2 psec
RJbw=500 GHz
EnablePJ=no

VtPRBS
PRBS2
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-1.0 V
Vhigh=1.0 V
EnableDeEmphasis=no
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=10 psec
FallTime=10 psec
Delay=0 psec
EnableRJ=yes
RJrms=0.2 psec
RJbw=50 MHz
EnablePJ=no

VtPRBS
PRBS3
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-1.0 V
Vhigh=1.0 V
EnableDeEmphasis=no
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=10 psec
FallTime=10 psec
Delay=0 psec
EnableRJ=yes
RJrms=0.4 psec
RJbw=500 GHz
EnablePJ=no

VtPRBS
Clock
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-1.0 V
Vhigh=1.0 V
EnableDeEmphasis=no
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=10 psec
FallTime=10 psec
Delay=0 psec
EnableRJ=no
EnablePJ=no

VtPRBS
PRBS4
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-1.0 V
Vhigh=1.0 V
EnableDeEmphasis=no
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=10 psec
FallTime=10 psec
Delay=0 psec
EnableRJ=no
EnablePJ=yes
PJwave[1]=Sinusoid
PJamp[1]=3.3 psec
PJfreq[1]=100 MHz

VtPRBS
PRBS5
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-1.0 V
Vhigh=1.0 V
EnableDeEmphasis=no
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=10 psec
FallTime=10 psec
Delay=0 psec
EnableRJ=no
EnablePJ=yes
PJwave[1]=Square
PJamp[1]=3.3 psec
PJfreq[1]=100 MHz

VtPRBS
PRBS6
Mode=Maximal Length LFSR
RegisterLength=4
Vlow=-1.0 V
Vhigh=1.0 V
EnableDeEmphasis=no
EdgeShape=Error Function Transition
BitRate=10 GHz
RiseTime=10 psec
FallTime=10 psec
Delay=0 psec
EnableRJ=no
EnablePJ=yes
PJwave[1]=Triangle
PJamp[1]=3.3 psec
PJfreq[1]=100 MHz

VtPRBS
PRBS7
Mode=Maximal Length LFSR              PJamp[1]=1.7 psec
RegisterLength=4                      PJamp[2]=3.3 psec
Vlow=-1.0 V                           PJamp[3]=2.2 psec
Vhigh=1.0 V                           PJfreq[1]=100 MHz
EnableDeEmphasis=no                   PJfreq[2]=50 MHz
EdgeShape=Error Function Transition   PJfreq[3]=25 MHz
BitRate=10 GHz
RiseTime=10 psec
FallTime=10 psec
Delay=0 psec
EnableRJ=no
EnablePJ=yes
PJwave[1]=Sinusoid
PJwave[2]=Square
PJwave[3]=Triangle
```

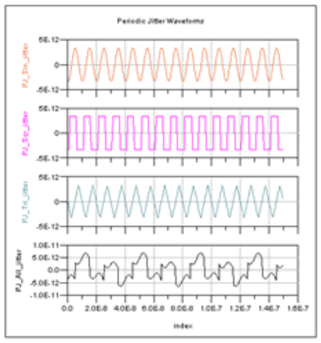
PRBS_Jitter

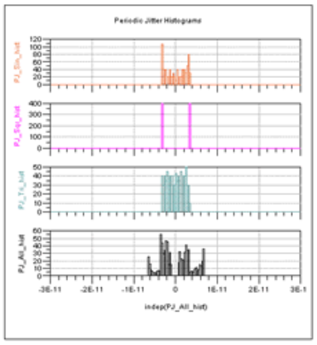
> **Note**
> This design is not shipped with a dataset because of space limitations. Please simulate this design prior to viewing the data display on PRBS_Jitter.dds. The data display is spread over two pages, one for graphs and one for equations. The graphs page should appear as shown below. The equations page contains information about how to use measurement expressions to generate histograms from the eye diagrams and how to recover the jitter waveforms from the voltage waveforms.











PRBS_Jitter.dds

Jitter is measured in terms of eye-closure along the time axis and in terms of the shape of the histogram which reflects the density of overlapped waveforms at a specific voltage level (typically at a zero-crossing).
The top row of plots show eye diagrams and histograms for random jitter. Compared to the **Clock** waveform, which is clear and unjittered, increasing jitter standard deviation and increasing measurement bandwidth both result in thickening of the transition band in the eye diagram. All three histograms across the zero-crossing of the jittered waveforms show gaussian distribution.

The bottom row consists of two composite plots. Each has four waveforms in it from the four periodically jittered sources. The left plot shows the jittering waveform which was recovered from the voltage waveform output by the sources. See the equations page for the exact derivation process. The right plot shows corresponding histograms at the zero-crossing.

Return to Contents

PRBS_Simulations

VtPRBS is intended to be used as a Time Domain source, i.e. in Transient simulation. This experiment serves to answer any questions regarding its behavior during the other Analog-RF simulations.

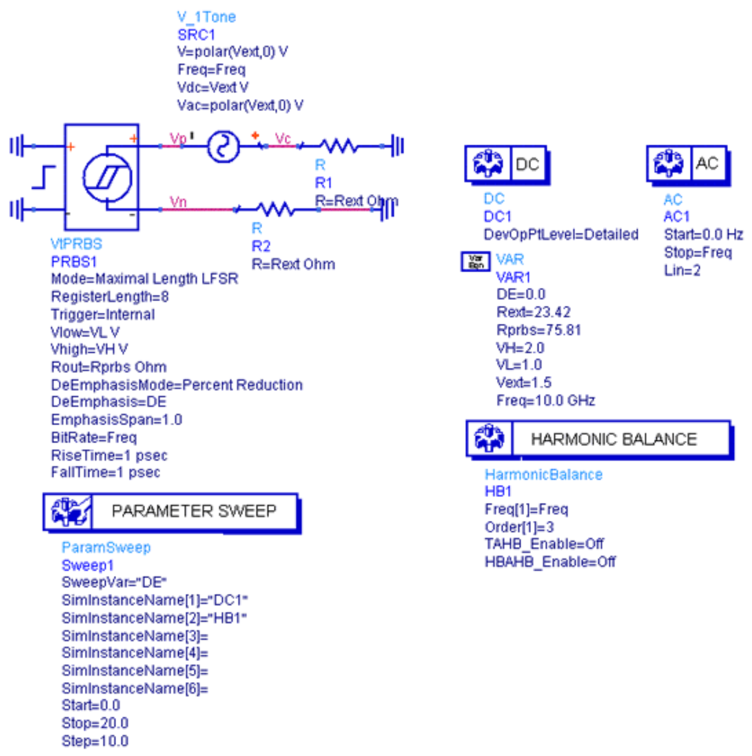
A single instance of VtPRBS is placed in series with a single tone voltage source that can operate both in time and frequency domains. External resistors are placed at the ends of this series combination terminating at ground. The reason for placing the extra voltage source is to derive a meaningful AC signal when VtPRBS itself is expected to act as a passive resistor and not a signal generator. Both the large signal and AC voltage of this source is set to the same value.

DC, AC and Harmonic Balance simulation controllers are set up as follows:

- DC - Detailed device operating point is requested. The current, voltage and power report from VtPRBS will be examined in the dataset.
- AC - This simulation is done at two frequencies, namely at DC and at the bit rate of the VtPRBS source which is also the fundamental tone of the series voltage source.
- HB - A third order simulation is performed for the fundamental tone equal to the bit rate of VtPRBS. The reason for choosing a higher order is to ensure numerical accuracy at convergence.

The peak voltage of the external voltage source is set to be at the zero-crossing value of the VtPRBS source for symmetry.

The amount of de-emphasis applied to VtPRBS is swept through 0%, 10% and 20% to illustrate voltage values at time=0 which forms the basis for evaluating HB and DC solutions.



V_1Tone
SRC1
V=polar(Vext,0) V
Freq=Freq
Vdc=Vext V
Vac=polar(Vext,0) V

R
R1
R=Rext Ohm

R
R2
R=Rext Ohm

VtPRBS
PRBS1
Mode=Maximal Length LFSR
RegisterLength=8
Trigger=Internal
Vlow=VL V
Vhigh=VH V
Rout=Rprbs Ohm
DeEmphasisMode=Percent Reduction
DeEmphasis=DE
EmphasisSpan=1.0
BitRate=Freq
RiseTime=1 psec
FallTime=1 psec

DC
DC1
DevOpPtLevel=Detailed

AC
AC1
Start=0.0 Hz
Stop=Freq
Lin=2

VAR
VAR1
DE=0.0
Rext=23.42
Rprbs=75.81
VH=2.0
VL=1.0
Vext=1.5
Freq=10.0 GHz

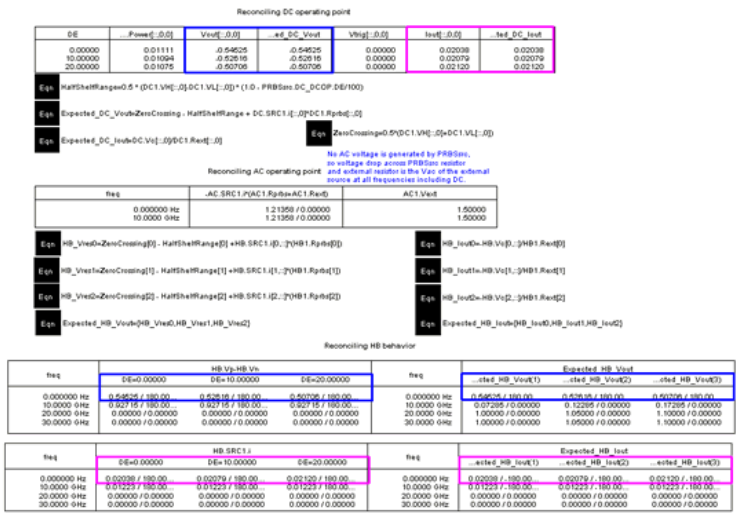
HarmonicBalance
HB1
Freq[1]=Freq
Order[1]=3
TAHB_Enable=Off
HBAHB_Enable=Off

ParamSweep
Sweep1
SweepVar="DE"
SimInstanceName[1]="DC1"
SimInstanceName[2]="HB1"
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=0.0
Stop=20.0
Step=10.0

PRBS_Simulations

Analytical equations are used to provide a set of expected results on the data display. Simulated data is compared against the corresponding analytical values as a test for accuracy.

- Reconciling DC operating point - Expected values of DC voltage and current are compared against dataset values across all three de-emphasis settings. In each case the values match up accurately. Note that for active de-emphasis, the time=0 voltage output by VtPRBS is its shelf value of logic 0 or logic 1 depending on whether the first data bit is a "0" or a "1". That is why voltage and current shelf values for a start of logic "0" (Maximal length LFSR, 8-bit register) are elevated with increase in de-emphasis amount, because the shelf values are migrating towards the zero crossing line.
- Reconciling AC operating point - Note that the VtPRBS source appears to have fallen silent at 0 Hz and 10 GHz (and every frequency in between) for AC simulations. Simulated current of 16.008 mA and known Rout value of 75.81 determine that net voltage drop of 1.21358 V occurs across VtPRBS. Simulated current through total external series resistance of 46.84 Ohm, shows the external voltage drop to be 0.74982 V. These two voltage values add up to exactly 1.5 V which is known to also come from the single tone source. The conclusion is that this source is the only active signal generator in AC simulations and that VtPRBS is silent.
- Reconciling HB behavior - Only the behavior at DC is of importance for understanding VtPRBS operation. The DC currents and voltages for each value of de-emphasis match with the DC-operating point values obtained during DC simulation. The behavior at the fundamental and harmonics are not affected by de-emphasis because it is left undefined for the VtPRBS. VtPRBS is a jitter-enabled source. With random variations permitted at the bit boundary, it is difficult to define the spectral response as a closed form solution. Hence there is no meaningful spectral behavior defined for this source other than at DC.



Reconciling DC operating point

DE	...Power[::,0,0]	Vout[::,0,0]	...ed_DC_Vout	Vtrig[::,0,0]	Iout[::,0,0]	...ted_DC_Iout
0.00000	0.01111	-0.54625	-0.54625	0.00000	0.02038	0.02038
10.00000	0.01094	-0.52616	-0.52616	0.00000	0.02079	0.02079
20.00000	0.01075	-0.50706	-0.50706	0.00000	0.02120	0.02120

```
Eqn HalfShelfRange=0.5 * (DC1.VH[::,0]-DC1.VL[::,0]) * (1.0 - PRBSsrc.DC_DCOP.DE/100)
Eqn Expected_DC_Vout=ZeroCrossing - HalfShelfRange + DC.SRC1.i[::,0]*DC1.Rprbs[::,0]
Eqn Expected_DC_Iout=DC.Vo[::,0]/DC1.Rext[::,0]
Eqn ZeroCrossing=0.5*(DC1.VH[::,0]+DC1.VL[::,0])
```

No AC voltage is generated by PRBSsrc,
so voltage drop across PRBSsrc resistor
and external resistor is the Vac of the external
source at all frequencies including DC.

Reconciling AC operating point

freq	-AC.SRC1.i*(AC1.Rprbs+AC1.Rext)	AC1.Vext
0.000000 Hz	1.21358 / 0.00000	1.50000
10.0000 GHz	1.21358 / 0.00000	1.50000

```
Eqn HB_Vres0=ZeroCrossing[0] - HalfShelfRange[0] +HB.SRC1.i[0,::]*(HB1.Rprbs[0])
Eqn HB_Vres1=ZeroCrossing[1] - HalfShelfRange[1] +HB.SRC1.i[1,::]*(HB1.Rprbs[1])
Eqn HB_Vres2=ZeroCrossing[2] - HalfShelfRange[2] +HB.SRC1.i[2,::]*(HB1.Rprbs[2])
Eqn Expected_HB_Vout={HB_Vres0,HB_Vres1,HB_Vres2}
Eqn HB_Iout0=-HB.Vo[0,::]/HB1.Rext[0]
Eqn HB_Iout1=-HB.Vo[1,::]/HB1.Rext[1]
Eqn HB_Iout2=-HB.Vo[2,::]/HB1.Rext[2]
Eqn Expected_HB_Iout={HB_Iout0,HB_Iout1,HB_Iout2}
```

Reconciling HB behavior

freq	HB.Vp-HB.Vn			freq	Expected_HB_Vout		
	DE=0.00000	DE=10.00000	DE=20.00000		...cted_HB_Vout(1)	...cted_HB_Vout(2)	...cted_HB_Vout(3)
0.000000 Hz	0.54625 / 180.00...	0.52616 / 180.00...	0.50706 / 180.00...	0.000000 Hz	0.54625 / 180.00	0.52616 / 180.00	0.50706 / 180.00
10.0000 GHz	0.92715 / 180.00...	0.92715 / 180.00...	0.92715 / 180.00...	10.0000 GHz	0.07285 / 0.00000	0.12285 / 0.00000	0.17285 / 0.00000
20.0000 GHz	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	20.0000 GHz	1.00000 / 0.00000	1.05000 / 0.00000	1.10000 / 0.00000
30.0000 GHz	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	30.0000 GHz	1.00000 / 0.00000	1.05000 / 0.00000	1.10000 / 0.00000

freq	HB.SRC1.i			freq	Expected_HB_Iout		
	DE=0.00000	DE=10.00000	DE=20.00000		...ected_HB_Iout(1)	...ected_HB_Iout(2)	...ected_HB_Iout(3)
0.000000 Hz	0.02038 / 180.00...	0.02079 / 180.00...	0.02120 / 180.00...	0.000000 Hz	0.02038 / -180.00...	0.02079 / -180.00...	0.02120 / -180.00...
10.0000 GHz	0.01223 / 180.00...	0.01223 / 180.00...	0.01223 / 180.00...	10.0000 GHz	0.01223 / -180.00...	0.01223 / -180.00...	0.01223 / -180.00...
20.0000 GHz	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	20.0000 GHz	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000
30.0000 GHz	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000	30.0000 GHz	0.00000 / 0.00000	0.00000 / 0.00000	0.00000 / 0.00000

PRBS_Simulations.dds

Return to Contents

Timed Examples

Examples that can be used as templates for simulating an RF AGC loop and an RF receiver for recovering 3GPP CPICH symbols and carrier frequency using a multi-rate PLL.

- *A 3GPP Receiver with RF PLL for Recovery of Carrier and CPICH Symbols* (examples)
- *RF AGC Loop Simulation* (examples)

A 3GPP Receiver with RF PLL for Recovery of Carrier and CPICH Symbols

Location: $HPEESOF_DIR/examples/Timed/RF_PLL_Examples_wrk

Objective

Existing receiver designs in the 3GPP library assumes carrier frequency is known and does not include techniques for recovering carrier frequency and CPICH data. In the workspace, an RF Receiver for recovering 3GPP CPICH symbols and carrier frequency using a multi-rate PLL is introduced. The test signal is generated by using a 3GPP Signal Source. An RF device under test (sub_RF_DUT) is used for the test. Before fully testing this receiver, RF_Receiver_ CPICP_PLL_check is set up to check the PLL to see if CPICH data and carrier frequency are fully recovered. For an RF channel with delay and no phase error, test bench measurements are shown in RF_PLL_Constellation_Delay.dds. As can be seen, the PLL successfully locks the carrier and the demodulated signal constellation is matched with the transmission constellation. In RF_Receiver_ CPICP_PLL_test, the RF channel with delay and phase error is considered. Measurements shown in RF_PLL_Constellation_PhaseError.dds demonstrate the PLL locks the carrier and the demodulated signal constellation is reasonably matched with the transmission constellation.

Setup

As seen from Figure 1, a 3GPP FDD RF Downlink source, G1, is employed to generate 3GPP downlink data. The 3GPP downlink signal is sent to an RF channel R1 with delay, phase interference and nonlinear distortion. For an accurate sampling process, a delay estimator, D1, is connected to RF channel input and output through R3, R4, R5 and R6 to estimating the RF channel delay. To demodulate the 3GPP data with carrier recovery capability, a PLL is designed in which an RF Demodulator with External Oscillator and RC filter, R2, as a phase detector, a subnet model, sub_CPICH_Recovery (X1) for recover CPICH data including magnitude and phase, a Circuit low pass filter, C1, as loop filter, and a FM modulator as VCO are used. In addition, an Up sampler, U1, is used in the PLL for the data rate matching. In the loop, the CPICH phase difference is sent from X1 to the loop filter. The loop filter can be used to extract the average phase difference. The loop will drive the phase difference toward to close to zero and finally the loop lock carrier frequency.

Figure 1: left half of schematic

3GPP CPICH Receiver with PLL
Demo recovering CPICH data and Carrier
RF_Receiver_CPICH_PLL_check

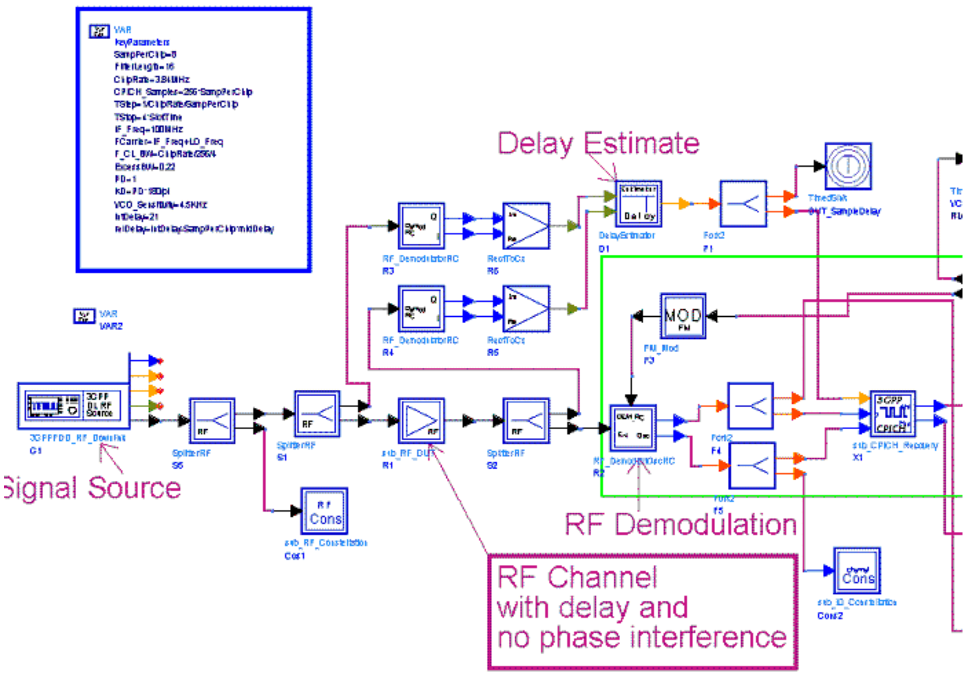
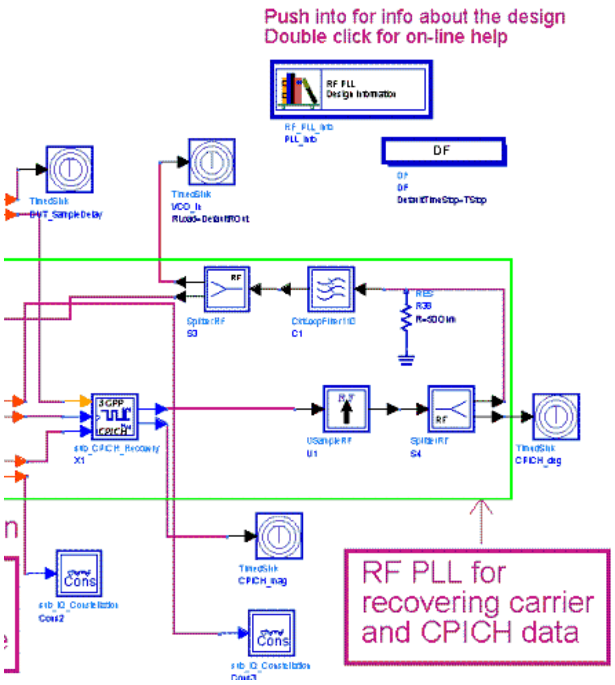
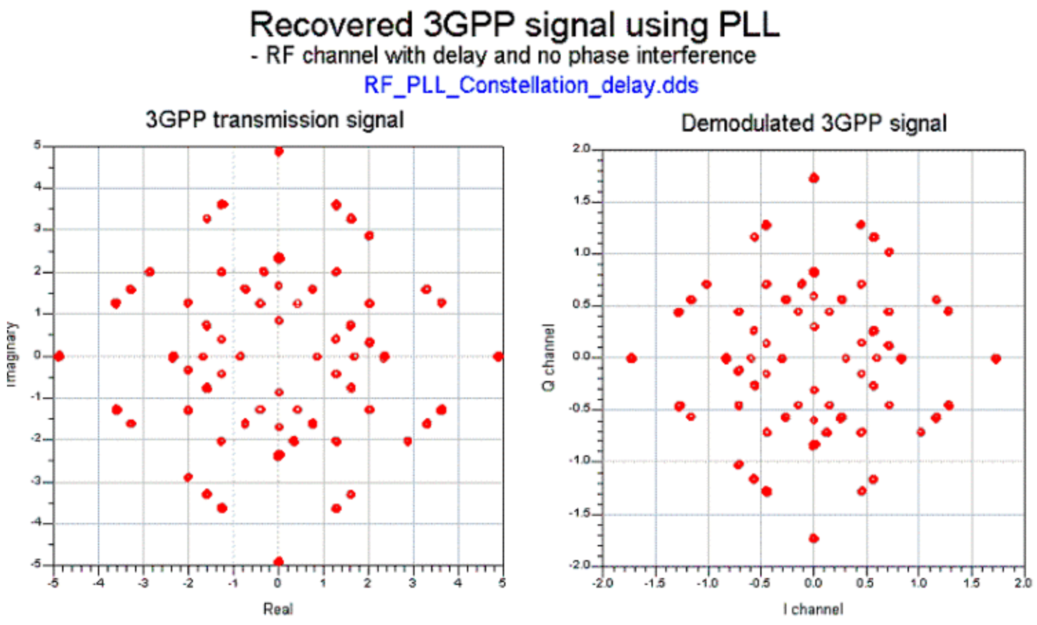


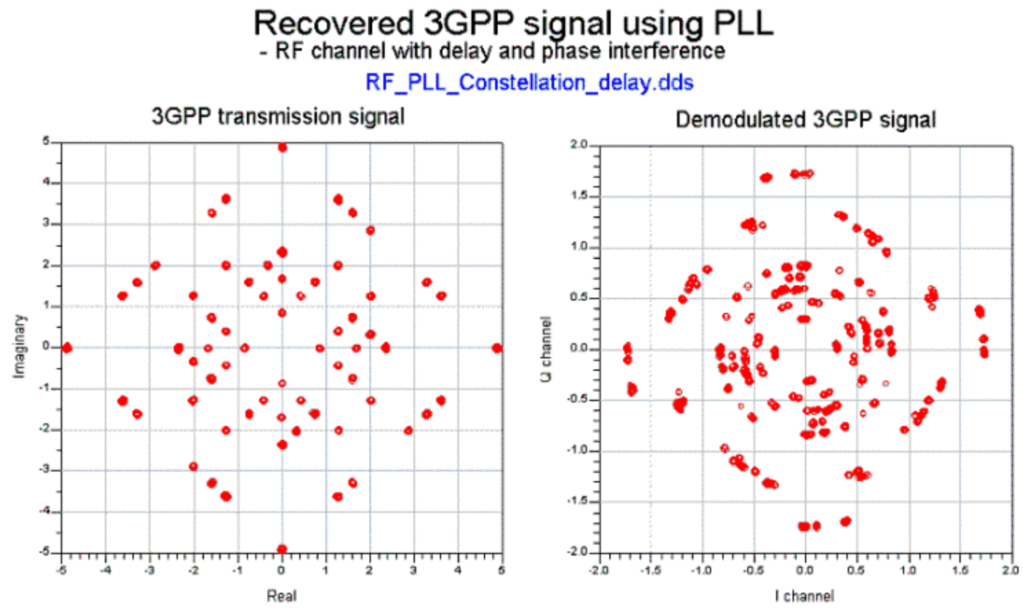
Figure 2: right half of schematic


Push into for info about the design
Double click for on-line help
RF PLL Design Information
DF
RF PLL for recovering carrier and CPICH data

Test design for 3GPP CPICH Receiver with PLL

Analysis





Notes

1. In this workspace, two designs are provided. These designs as templates can be used for simulation of a 3GPP receiver in which there is a PLL for recovering CPICH symbols as well as carrier frequency. Along with these designs, the data display files and reference datasets are available for user's reference.
2. In each design, there is an RF_PLL_Info model at the right upper corner. Push into this model you will see a brief description for the design. Double click this model a model parameter window will be popped up, then click the help button at the right lower corner; the on-line help for the workspace description will be available.

RF AGC Loop Simulation

Location: $HPEESOF_DIR/examples/Timed/RF_AGC_Loop_wrk

Objective

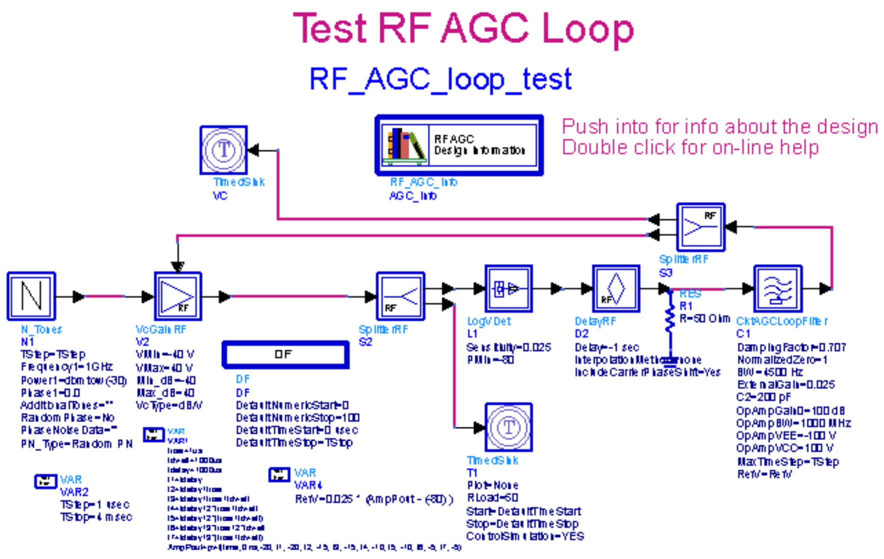
Automatic Gain Control (AGC) loops are important for communications systems where wide amplitude variations in the output signal lead to a performance degradation. These signals need a good control to maintain a constant signal level at the output. In the example an AGC loop for an amplifier with a RF input signal is shown. Test results are shown in RF_AGC_loop_test.dds. As can be seen, the loop output is independent of the input power level and can be controlled in desired power level.

Setup

The structure of an AGC system is determined by the requirements of the communications system. ADS provides top-down RF design that can be used to analyze the steady-state and transient response of the control loop. As seen from Figure 1, the control voltage feedback is to the voltage controlled gain block, VcGainRF, with gain dB/V. A delay block, DelayRF, is in the loop and is required to resolve DataFlow deadlock. 50 ohm resistor at DelayRF output is required since the CktAGCLoopFilter has infinite input resistance. LogVDet sensitivity is in terms of 0.025 volts/dB. The gain outside the CktAGCLoopFilter is 0.025. The power into the loop from the N_Tones source is fixed at -30 dBm. The pwl() expression sets the desired OpAmp reference voltage, RefV, as a function of time. This defines the power level desired for the AGC output that can be measured by the sink T1. The AGC output power level is independent of input power level sent by the signal source N1. This means the design has desired AGC characteristic. Figure 2 shows the simulation results.

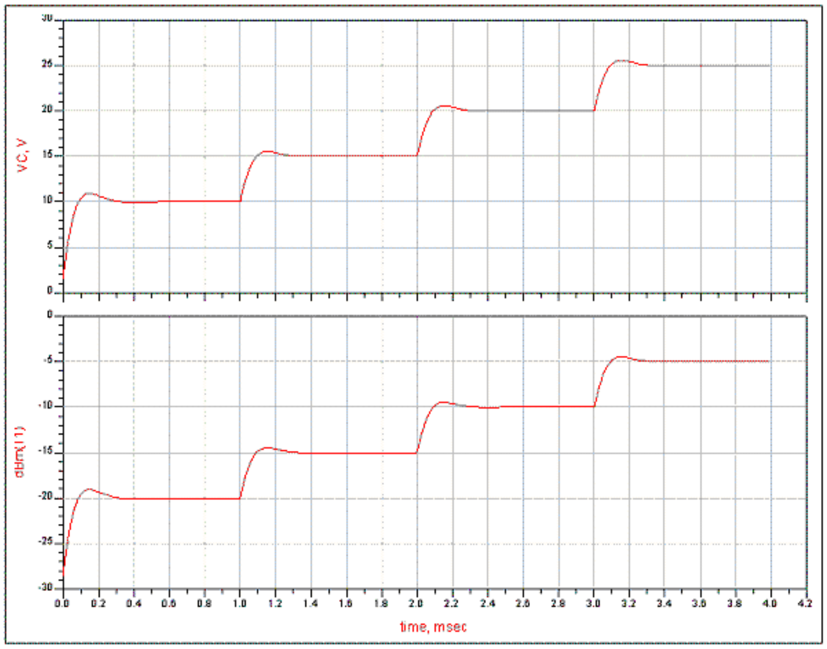
Test Design for AGC Loop

Figure 1: Test Design for AGC Loop



Analysis

Figure 2: Test result for the AGC Loop



Notes

- In this workspace, an AGC loop design is provided. As a template this design can be used for simulation of RF AGC loop. Along with the design, the data display file and reference dataset are provided for user's reference.
- In the design, there is an RF_AGC_Info model at the right upper corner. Push into this model you will see a brief description for the design. Double click this model a model parameter window will be popped up, then click the help button at the right lower corner; the on-line help for the workspace description will be available.

Tutorial Examples

Examples of basic usage and functionality concepts in the tutorials included with the documentation.

- *A Model B Test Lab Optimization* (examples)
- *A Simple Integrator* (examples)
- *Behavioral Amplifier model - Amplifier2* (examples)
- *Bit-Error-Rate Measurement Tutorial* (examples)
- *Bus Wire Basic Example* (examples)
- *Construct Three-Port S-Parameter File from Two-Port Measurement Datasets* (examples)
- *Example Test Lab for Two Stage Amplifier Design* (examples)
- *FIR Digital Filter* (examples)
- *Generate Various Modulated Sources* (examples)
- *Learn Tuning an Elliptic Filter, a Dynamic Load Line, and a Microstrip Bandpass Filter* (examples)
- *LSSP (Large-Signal S-Parameters), Basic Example* (examples)
- *Mixer2 Model Validation Example File* (examples)
- *Noise Power Ratio Simulation* (examples)
- *Noise Simulation in Envelope Analysis* (examples)
- *Optimization and Parameter Sweeps Using DSP Schematic* (examples)
- *Optimization Final Analysis Demonstration* (examples)
- *Optimization of a Low Pass Filter* (examples)
- *Optimization of An Impedance Transformation Network* (examples)
- *Oscillator Simulations using Transient, Harmonic Balance and Envelope Simulators* (examples)
- *Physical Layout Connectivity Check* (examples)
- *Power Amplifier Behavior Model* (examples)
- *Ptolemy DSP Sink Export to GoldenGate* (examples)
- *Ptolemy DSP Source Export to GoldenGate* (examples)
- *Quick Tour of Communication System Design in Ptolemy* (examples)
- *RF System Budget measurements for 2port cascaded networks* (examples)
- *Sensitivity Analysis, Basic Example* (examples)
- *Simple Example Showing Application of DOE* (examples)
- *Simple Ptolemy Sequencer Example* (examples)
- *S-Parameter Simulation Example* (examples)
- *S-Parameters of 2-Port Terminated with Other Networks* (examples)
- *Statistical Correlation in ADS* (examples)
- *Statistical Design Example for Oscillator Yield Analysis* (examples)
- *Swept Optimization, Basic Example* (examples)
- *Tutorial on Budget Analysis (Archive)* (examples)
- *User-Compiled Model Examples* (examples)
- *Using DataAccessComponent (DAC) and S2PMDIF Component* (examples)
- *Using Expressions in the Data Display Window* (examples)
- *Using Frequency-Domain Defined Devices (FDD)* (examples)
- *Using HB Noise Controller* (examples)
- *Using N-State Modulator Component to Generate Modulated Signals* (examples)
- *Using Measured Load Pull Data to Design and Optimize Impedance-Matching Networks* (examples)
- *Using OscPort2 in Oscillator Simulation* (examples)
- *Using SP_Probe in ADS* (examples)
- *Using Symbolically Defined Devices (SDD)* (examples)
- *Various Examples on using ADS Simulation Controllers* (examples)
- *VCO Simulations* (examples)
- *X-Parameters, Generating a Model and Comparing it with a Transistor-Level Simulation* (examples)
- *X-Parameters, Various Simulations Using Models Generated from Measurements* (examples)
- *Yield Analysis of An Impedance Transformer* (examples)
- *Yield Optimization of An Impedance Transformer* (examples)
- *Yield Sensitivity Analysis of A Low Pass Filter* (examples)
- *Low Pass Filter Demo* (examples)
- *W-element Extraction Example* (examples)

A Model B Test Lab Optimization

Location: $HPEESOF_DIR/examples/Tutorial/TestLab_HOWTO_wrk

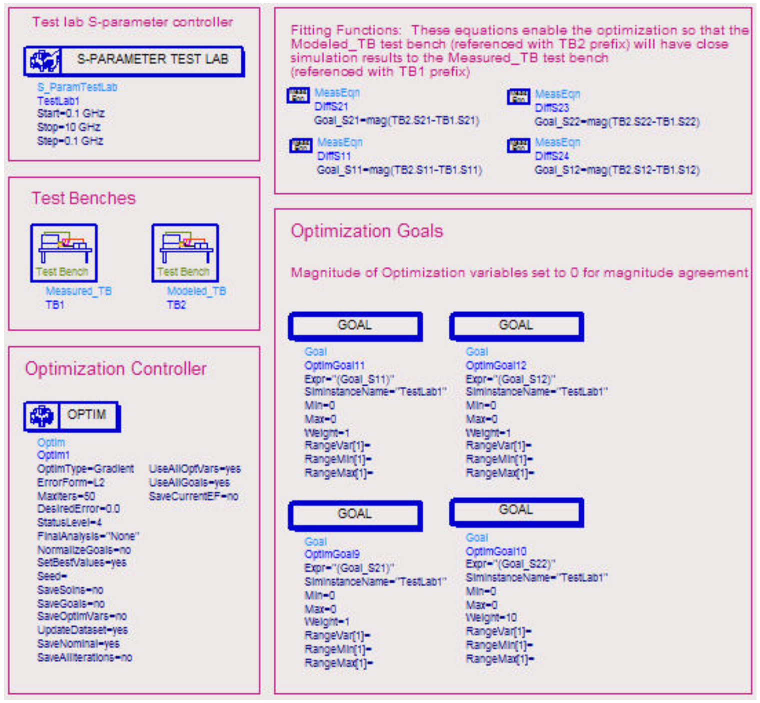
Objective

This example uses a Series IV style of Test Lab Optimization to demonstrate an alternative

A-Model-B example.

Setup

1. "TestLab_AmodelB" Uses four optimization variables to optimize the modeled data and fit S-Parameter characteristics to those of the measured data.
2. "Measured_tb" reads in measured S-Parameter data for a device at a particular device operating point.
3. "Modeled_tb" small signal model of an equivalent circuit for the transistor being measured.
4. "device_model" small signal equivalent circuit model used as a subcircuit in the Modeled_tb design.
5. "TestLab_AmodelB.dds" plots the measured and modeled data.



Test lab S-parameter controller

```
S_ParamTestLab
TestLab1
Start=0.1 GHz
Stop=10 GHz
Step=0.1 GHz
```

Test Benches

```
Measured_TB      Modeled_TB
TB1              TB2
```

Optimization Controller

```
Optim
Optim1
OptimType=Gradient        UseAllOptVars=yes
ErrorForm=L2              UseAllGoals=yes
MaxIters=50               SaveCurrentEF=no
DesiredError=0.0
StatusLevel=4
FinalAnalysis="None"
NormalizeGoals=no
SetBestValues=yes
Seed=
SaveSolns=no
SaveGoals=no
SaveOptimVars=no
UpdateDataset=yes
SaveNominal=yes
SaveAllIterations=no
```

Fitting Functions: These equations enable the optimization so that the Modeled_TB test bench (referenced with TB2 prefix) will have close simulation results to the Measured_TB test bench (referenced with TB1 prefix)

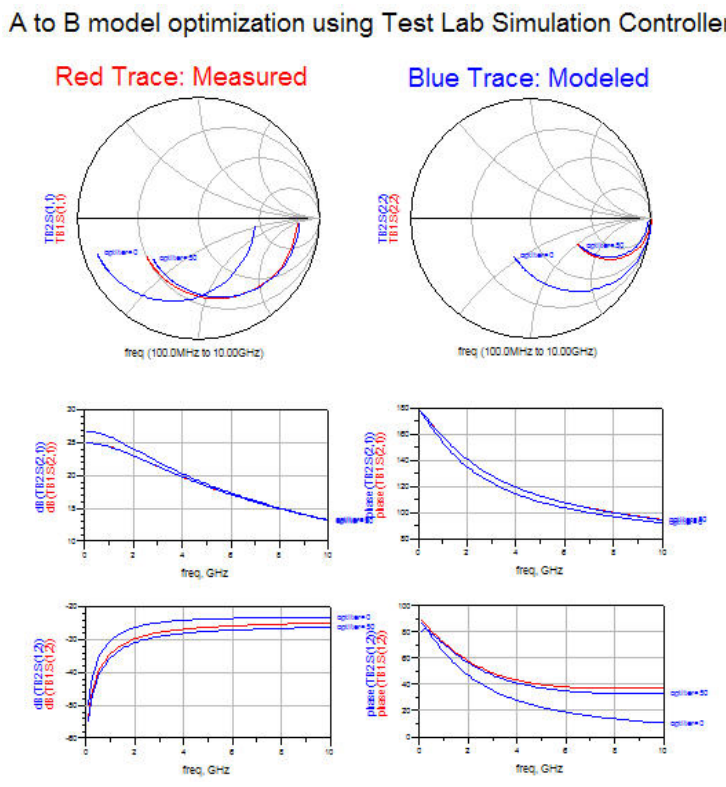
```
MeasEqn                                MeasEqn
DiffS21                                DiffS23
Goal_S21=mag(TB2.S21-TB1.S21)          Goal_S22=mag(TB2.S22-TB1.S22)

MeasEqn                                MeasEqn
DiffS11                                DiffS24
Goal_S11=mag(TB2.S11-TB1.S11)          Goal_S12=mag(TB2.S12-TB1.S12)
```

Optimization Goals

Magnitude of Optimization variables set to 0 for magnitude agreement

```
Goal                              Goal
OptimGoal11                       OptimGoal12
Expr="(Goal_S11)"                 Expr="(Goal_S12)"
SimInstanceName="TestLab1"        SimInstanceName="TestLab1"
Min=0                             Min=0
Max=0                             Max=0
Weight=1                          Weight=1
RangeVar[1]=                      RangeVar[1]=
RangeMin[1]=                      RangeMin[1]=
RangeMax[1]=                      RangeMax[1]=

Goal                              Goal
OptimGoal9                        OptimGoal10
Expr="(Goal_S21)"                 Expr="(Goal_S22)"
SimInstanceName="TestLab1"        SimInstanceName="TestLab1"
Min=0                             Min=0
Max=0                             Max=0
Weight=1                          Weight=10
RangeVar[1]=                      RangeVar[1]=
RangeMin[1]=                      RangeMin[1]=
RangeMax[1]=                      RangeMax[1]=
```

Analysis



Notes

- The Optimization goals in "TestLab_AmodelB" are used to force the magnitude of the difference between measured and modeled S-parameters to be zero.
- You can change the weighting factors in "TestLab_AmodelB" to emphasize the matching of one S-Parameter more than another.

A Simple Integrator

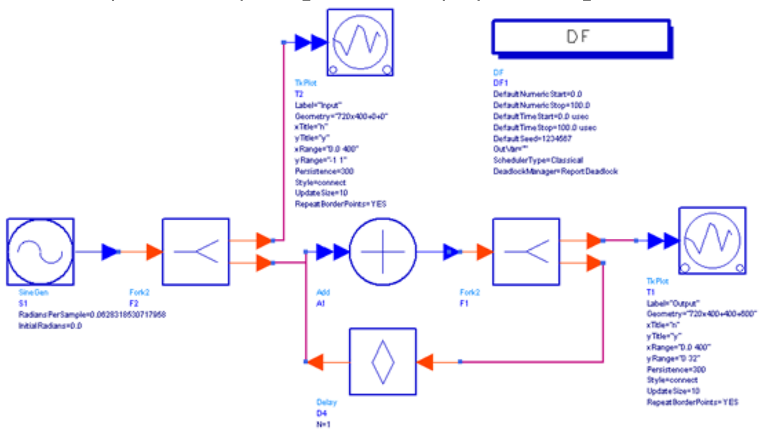
Location: $HPEESOF_DIR/examples/Tutorial/integrator_wrk

Objective

This example illustrates a simple integrator realized using Ptolemy building blocks and the basic Data Flow (DF) simulation and TkPlot display of results.

Setup

A sine wave is generated and then integrated through the use of add and delay functional blocks. Input and output signals are displayed through the TKPlot.



TkPlot
T2
Label="Input"
Geometry="720x400+0+0"
xTitle="n"
yTitle="y"
xRange="0.0 400"
yRange="-1 1"
Persistence=300
Style=connect
UpdateSize=10
RepeatBorderPoints=YES

DF
DF1
DefaultNumericStart=0.0
DefaultNumericStop=100.0
DefaultTimeStart=0.0 usec
DefaultTimeStop=100.0 usec
DefaultSeed=1234567
OutVar=""
SchedulerType=Classical
DeadlockManager=ReportDeadlock

SineGen
S1
RadiansPerSample=0.0628318530717958
InitialRadians=0.0

Fork2
F2

Add
A1

Fork2
F1

TkPlot
T1
Label="Output"
Geometry="720x400+400+500"
xTitle="n"
yTitle="y"
xRange="0.0 400"
yRange="0 32"
Persistence=300
Style=connect
UpdateSize=10
RepeatBorderPoints=YES

Delay
D4
N=1

Analysis

Figure 1: Input signal

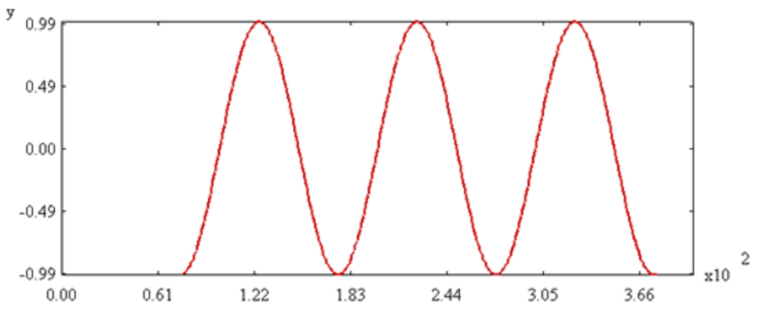
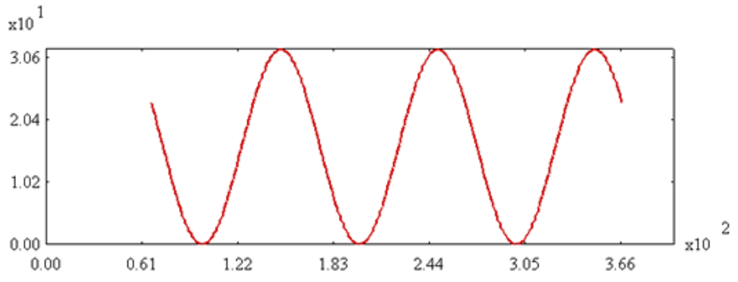


Figure 2: Output signal



Notes

- Setting TKPlot Geometry coordinates to negative values may cause the plot to be off screen on some PC displays.

Behavioral Amplifier model - Amplifier2

Location: $HPEESOF_DIR/examples/Tutorial/Amplifier2_Example_wrk

Objective

The purpose of this workspace is to demonstrate the use of the Amplifier2 system model. The Amplifier2 model behaves the same as the Amplifier model, but with improvements with noise calculations, AM-2-PM conversion, and the ability to handle complex S21.

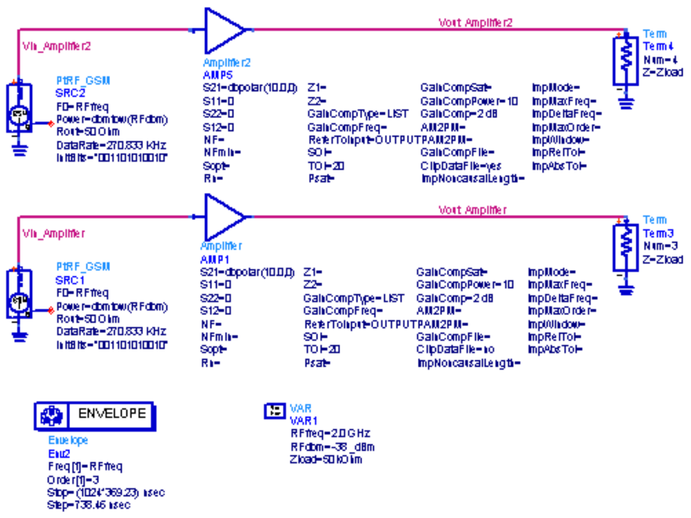
Setup

The following designs, datasets (**.ds**) and data displays (**.dds**) are included in this workspace:

1. **Amplifier2_DC** : Demonstrates the use of Amplifier2 for DC analysis.
2. **Amplifier2_AC** : Demonstrates the use of Amplifier2 for AC analysis.
3. **Amplifier2_SP** : Demonstrates the use of Amplifier2 for SP analysis.
4. **Amplifier2_Tran** : Demonstrates the use of Amplifier2 for Tran analysis.
5. **Amplifier2_HB** : Demonstrates the use of Amplifier2 for HB analysis.
6. **Amplifier2_CE** : Demonstrates the use of Amplifier2 for CE analysis.
7. **Amplifier2_Linear_AM2PM** : Demonstrates the use of Amplifier2 with AM2PM specified. AM2PM is demonstrated for a linear amplifier but is supported for all compression options.
8. **Amplifier2_TOI_GainCompPower_GainComp_HB1tone** : Demonstrates the use of Amplifier2 with compression parameters specified. Validates GainCompPower and GainComp.
9. **Amplifier2_TOI_GainCompPower_GainComp_HB2tone** : Demonstrates the use of Amplifier2 with compression parameters specified. Validates TOI.
10. **Amplifier2_GCOMP7** : Demonstrates the use of Amplifier2 with a GCOMP7 block specifying the compression.
11. **Amplifier2_GCOMP7_extrapolation** : Demonstrates the use of Amplifier2 with a GCOMP7 block specifying the compression and where extrapolation is necessary.
12. **Amplifier2_LinearNoise_NF** : Demonstrates the use of Amplifier2 for linear noise with NF set.
13. **Amplifier2_LinearNoise_NFminSoptRn** : Demonstrates the use of Amplifier2 for linear noise with (NFmin,Sopt,Rn) set.
14. **Amplifier2_NonlinearNoise_NF** : Demonstrates the use of Amplifier2 for nonlinear noise with NF set.
15. **Amplifier2_NonlinearNoise_NF_*** : These are auxiliary designs used by

Amplifier2_NonlinearNoise_*.

16. **Amplifier2_NonlinearNoise_NFminSoptRn** : Demonstrates the use of Amplifier2 for nonlinear noise with (NFmin,Sopt,Rn) set.

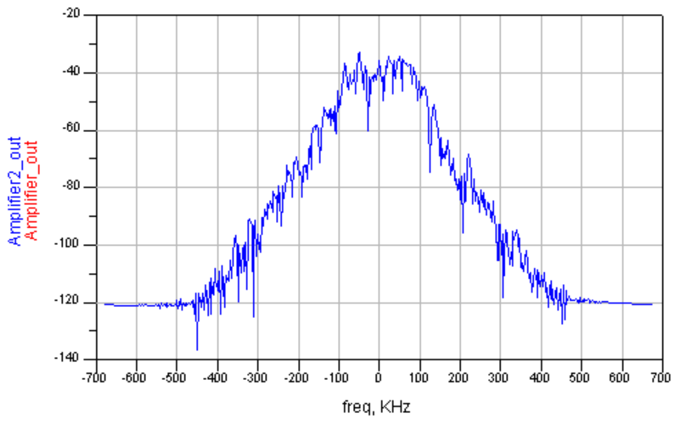


```
PtRF_GSM
SRC2
FD=RFfreq
Power=dbmtow(RFdbm)
Rout=50 Ohm
DataRate=270.833 KHz
InitBits="00110101001O"

Amplifier2
AMP5
S21=dbpolar(10,0,0)   Z1=                        GainCompSat=               ImpMode=
S11=0                 Z2=                        GainCompPower=10           ImpMaxFreq=
S22=0                 GainCompType=LIST          GainComp=2 dB              ImpDeltaFreq=
S12=0                 GainCompFreq=              AM2PM=                     ImpMaxOrder=
NF=                   ReferToInput=OUTPUTPAM2PM=                            ImpWindow=
NFmin=                SOI=                       GainCompFile=              ImpRelTol=
Sopt=                 TOI=20                     ClipDataFile=yes           ImpAbsTol=
Rn=                   Psat=                      ImpNoncausalLength=

Term
Term4
Num=4
Z=Zload

PtRF_GSM
SRC1
FD=RFfreq
Power=dbmtow(RFdbm)
Rout=50 Ohm
DataRate=270.833 KHz
InitBits="00110101001O"

Amplifier
AMP1
S21=dbpolar(10,0,0)   Z1=                        GainCompSat=               ImpMode=
S11=0                 Z2=                        GainCompPower=10           ImpMaxFreq=
S22=0                 GainCompType=LIST          GainComp=2 dB              ImpDeltaFreq=
S12=0                 GainCompFreq=              AM2PM=                     ImpMaxOrder=
NF=                   ReferToInput=OUTPUTPAM2PM=                            ImpWindow=
NFmin=                SOI=                       GainCompFile=              ImpRelTol=
Sopt=                 TOI=20                     ClipDataFile=no            ImpAbsTol=
Rn=                   Psat=                      ImpNoncausalLength=

Term
Term3
Num=3
Z=Zload

ENVELOPE
Envelope
Env2
Freq[1]=RFfreq
Order[1]=3
Stop=(1024*369.23) usec
Step=738.46 usec

VAR
VAR1
RFfreq=2.0 GHz
RFdbm=-38 _dBm
Zload=50 kOhm
```

Comparison simulation with Amplifier and Amplifer2 (Amplifier2_CE)

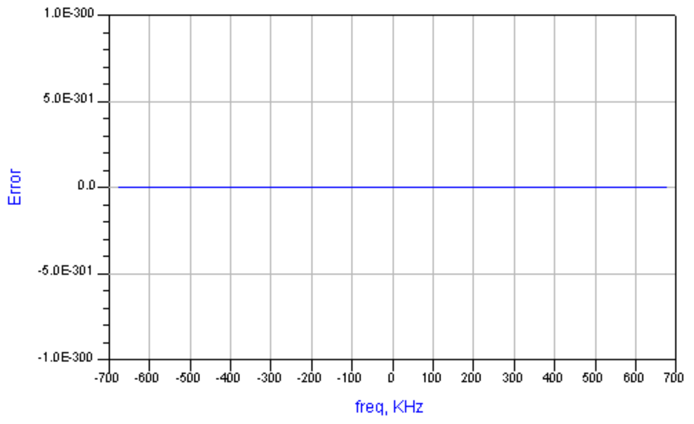
Eqn Amplifier2_out=dBm(fs(Vout_Amplifier2[1],,,,,"Kaiser"))

Eqn Amplifier_out=dBm(fs(Vout_Amplifier[1],,,,,"Kaiser"))



Comparison of Amplifier and Amplifier 2 with modulated signal (Amplifier2_CE.dds)

Eqn Error=dBm(fs(Vout_Amplifier[1],,,,,"Kaiser"))-dBm(fs(Vout_Amplifier[1],,,,,"Kaiser"))



Difference error between Amplifier and Amplifier 2 (Amplifier2_CE.dds)

Bit-Error-Rate Measurement Tutorial

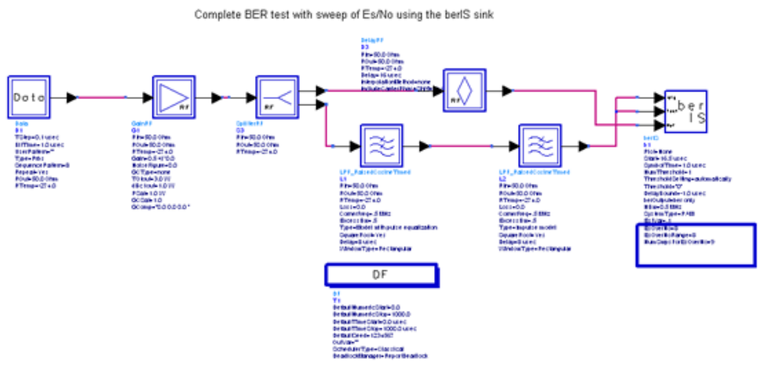
Location: $HPEESOF_DIR/examples/Tutorial/Learn_BER_wrk

Objective

This example illustrates basic Bit-Error-Rate (BER) measurements using importance sampling via the BER_IS measurement component.

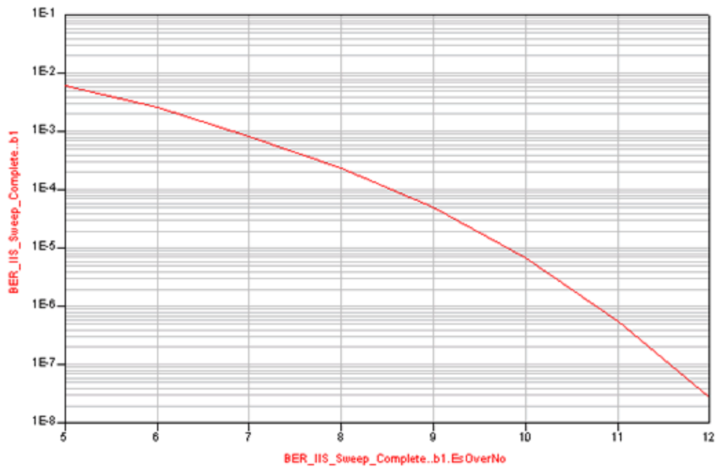
Setup

1. "BER_IIS" uses importance sampling to greatly reduce the number of samples required to estimate BER, even for very low BER systems.
2. "BER_IS_Sweep_Complete" adds swept Noise value to create a "waterfall" curve.



Analysis

Figure 1: BER versus swept Es/No



Notes

- Simulation controller used: DF.
- For more information about BER measurements, refer to "Sinks" in the Signal Processing Components documentation.

Bus Wire Basic Example

Location: $HPEESOF_DIR/examples/Tutorial/wire_bus_wrk

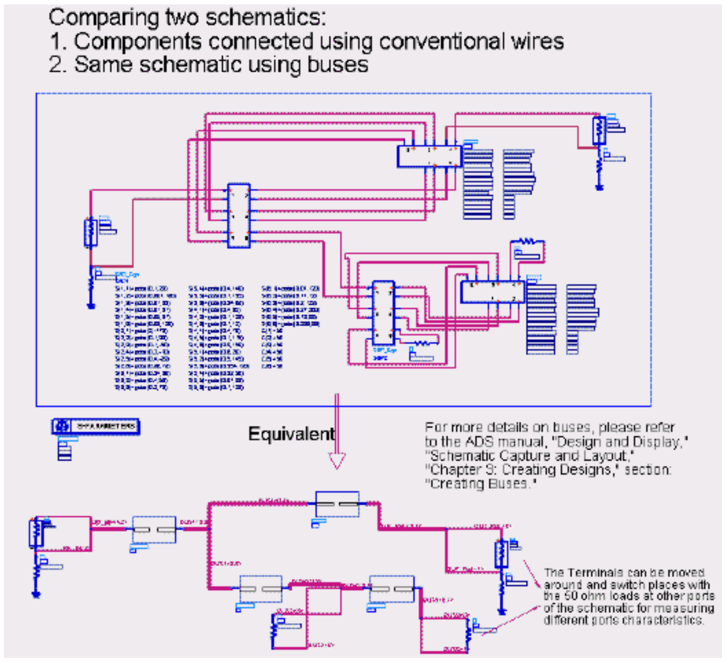
Objective

This workspace has two examples that show how to use and create bus wires.

Setup

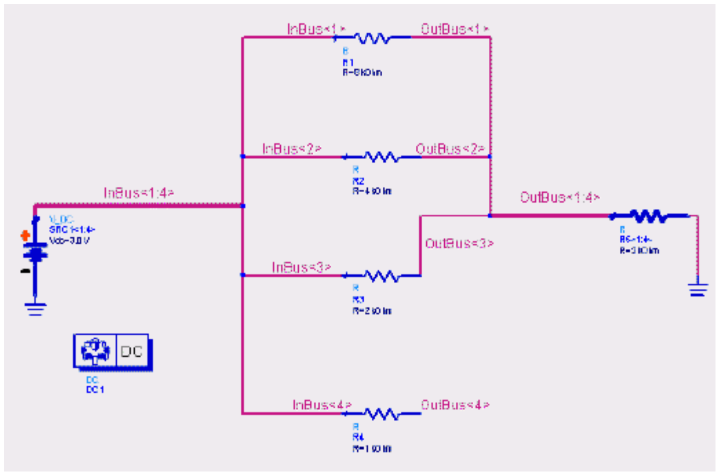
On design page RF_Taps, two equivalent schematics are shown. One uses conventional wires, and the other utilizes the new bus wire feature in ADS.

Figure 1: Two equivalent schematics


Comparing two schematics:
1. Components connected using conventional wires
2. Same schematic using buses
Equivalent
For more details on buses, please refer to the ADS manual, "Design and Display," "Schematic Capture and Layout," "Chapter 3: Creating Designs," section: "Creating Buses."
The Terminals can be moved around and switch places with the 50 ohm loads at other ports of the schematic for measuring different ports characteristics.

The simulation result is shown on RF_Taps.dds. The results are identical.simple_demo is another simple circuit showing a few illustrations on how to iterate instances, create or edit bus wires, and use labels with correct syntax to identify a bus and everything connected to it.

Figure 2: Basic bus wires



S_Param_BUS demonstrates how bus wires can be used in multi-port S-Parameter simulations.

Notes

- Please note that the application circuits used in this example are only for illustration purposes, and do not necessarily perform any specific functions.
- For more details on buses, please refer to *Creating Buses* (usrguide).

Use Data Access Component to Construct A Three-Port S-Parameter File from Two-Port Measurement Datasets

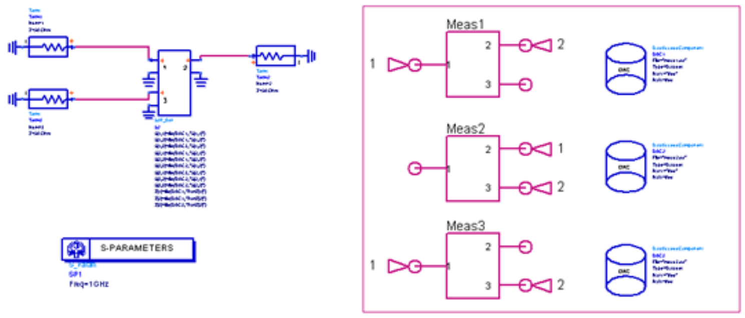
Location: $HPEESOF_DIR/examples/Tutorial/Data_comp_wrk

Objective

This example shows how to use the Data Access Component (DAC) to construct a 3-port S-parameter file from 3 different two-port measurements of a Device Under Test (DUT) using either dataset files or touchstone format files.

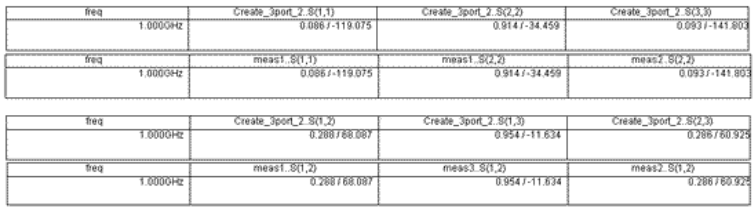
Setup

1. "DUT" is a simple circuit that is used to generate the test data.
2. "meas1", "meas2" and "meas3" each call DUT and create two-port S-parameter datasets. These datasets (meas1.ds, meas2.ds and meas3.ds) were used to generate Touchstone format files (*.s2p files) using the "New File/Instrument Server" function found under the "Window" menu in the schematic window.
3. "Create_3port" combines the 3 two-port datasets into one 3-port dataset.
4. "Create_3port_2" combines the 3 two-port touchstone data files (*.s2p files) into one 3-port dataset.



Analysis

Figure 1: Results from "Create_3port2" compared to original data from meas1.ds, meas2.ds and meas3.ds



freq	Create_3port_2..S(1,1)	Create_3port_2..S(2,2)	Create_3port_2..S(3,3)
1.000GHz	0.086 / -119.075	0.914 / -34.459	0.093 / -141.803

freq	meas1..S(1,1)	meas1..S(2,2)	meas2..S(2,2)
1.000GHz	0.086 / -119.075	0.914 / -34.459	0.093 / -141.803

freq	Create_3port_2..S(1,2)	Create_3port_2..S(1,3)	Create_3port_2..S(2,3)
1.000GHz	0.288 / 68.087	0.954 / -11.634	0.286 / 60.925

freq	meas1..S(1,2)	meas3..S(1,2)	meas2..S(1,2)
1.000GHz	0.288 / 68.087	0.954 / -11.634	0.286 / 60.925

Test Lab for Two Stage Amplifier Design

Location: $HPEESOF_DIR/examples/Tutorial/TestLabForTwoStgAmp_wrk

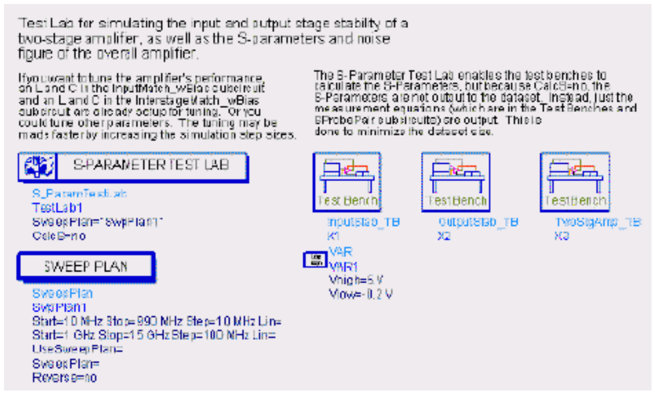
Objective

This example shows the use of a TestLab, TestBenches, and SProbes while tuning component values in a two-stage amplifier design. The objective is to maintain stability, by checking the stability factors at both the first and second stage devices, while maximizing the overall gain and minimizing the noise figure of the amplifier.

Setup

TwoStgAmp_TL is the top-level schematic, that has an S-Parameter Test Lab. This schematic has three test benches as well as several component variables for tuning.

Figure: Test Lab simulation setup



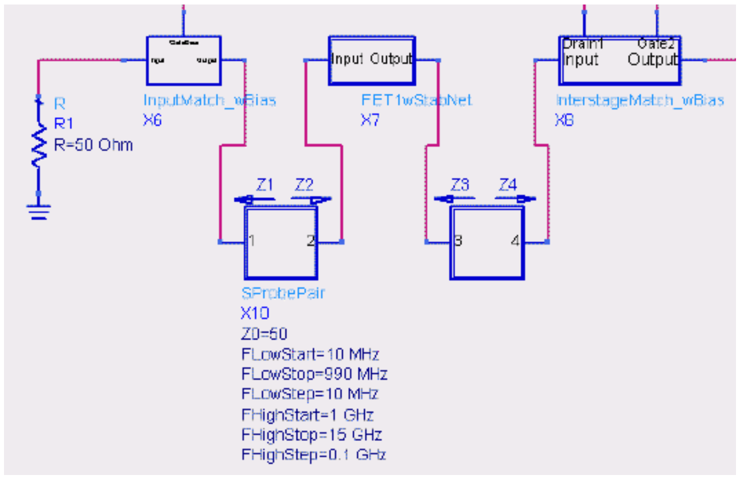
Test Lab for simulating the input and output stage stability of a two-stage amplifier, as well as the S-parameters and noise figure of the overall amplifier.

If you want to tune the amplifier's performance, an L and C in the InputMatch_wBias subcircuit and an L and C in the InterstageMatch_wBias subcircuit are already setup for tuning. Or you could tune other parameters. The tuning may be made faster by increasing the simulation step sizes.

The S-Parameter Test Lab enables the test benches to calculate the S-Parameters, but because CalcS=no, the S-Parameters are not output to the dataset. Instead, just the measurement equations (which are in the Test Benches and SProbePair subcircuits) are output. This is done to minimize the dataset size.

```
S_ParamTestLab
TestLab1
SweepPlan="SwpPlan1"
CalcS=no

SweepPlan
SwpPlan1
Start=10 MHz Stop=990 MHz Step=10 MHz Lin=
Start=1 GHz Stop=15 GHz Step=100 MHz Lin=
UseSweepPlan=
SweepPlan=
Reverse=no

InputStab_TB
X1

OutputStab_TB
X2

TwoStgAmp_TB
X3

VAR
VAR1
Vhigh=5 V
Vlow=-0.2 V
```

InputStab_TB is a test bench with an SProbePair for simulating the stability of the first stage. The SProbePair measures the reflection coefficients (and impedances) presented to the input and output of the first stage FET, as well as the reflection coefficients (and impedances) looking into the input of the FET and looking into the output. Based on these reflection coefficients, a stability factor is computed.

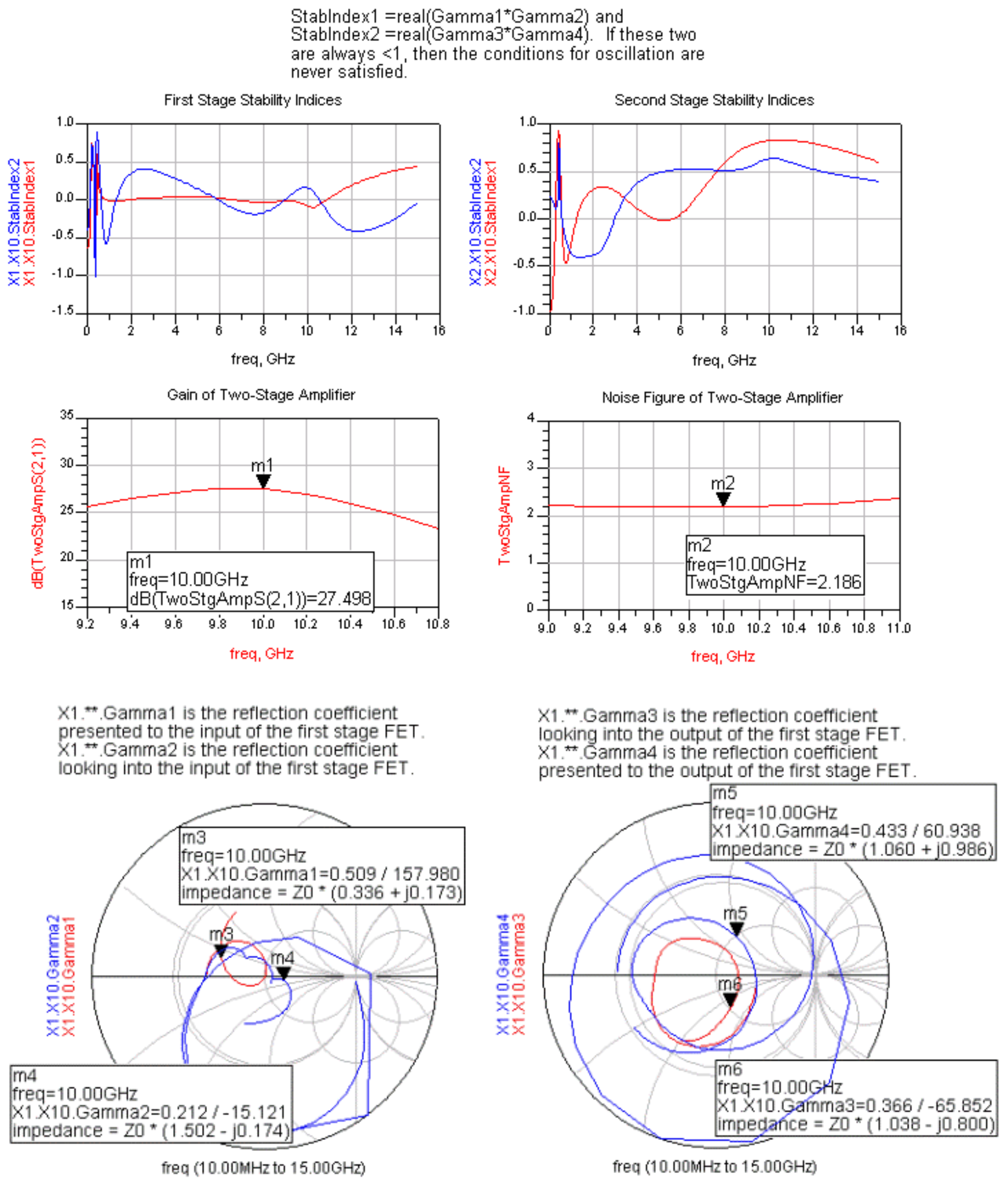


```
R
R1
R=50 Ohm

InputMatch_wBias
X6

FET1wStabNet
X7

InterstageMatch_wBias
X8

SProbePair
X10
Z0=50
FLowStart=10 MHz
FLowStop=990 MHz
FLowStep=10 MHz
FHighStart=1 GHz
FHighStop=15 GHz
FHighStep=0.1 GHz
```

Test Bench subcircuit, with S-probe inserted to measure impedances looking both directions at input and output of first stage FET

The frequency range of the S-parameter simulation is set on the Test Lab schematic, so the frequency range values in this Test Bench are ignored.
OutputStab_TB is identical to the InputStab_TB, except that it is set up to measure the stability factors and reflection coefficients associated with the second stage FET.
TwoStgAmp_TB is the same as the other two test benches, except that it is used for measuring the S-parameters and noise figure of the overall amplifier. It does not have any SProbes.

Analysis


StabIndex1 =real(Gamma1*Gamma2) and
StabIndex2 =real(Gamma3*Gamma4). If these two
are always <1, then the conditions for oscillation are
never satisfied.

X1.**.Gamma1 is the reflection coefficient
presented to the input of the first stage FET.
X1.**.Gamma2 is the reflection coefficient
looking into the input of the first stage FET.

X1.**.Gamma3 is the reflection coefficient
looking into the output of the first stage FET.
X1.**.Gamma4 is the reflection coefficient
presented to the output of the first stage FET.

m1
freq=10.00GHz
dB(TwoStgAmpS(2,1))=27.498

m2
freq=10.00GHz
TwoStgAmpNF=2.186

m3
freq=10.00GHz
X1.X10.Gamma1=0.509 / 157.980
impedance = Z0 * (0.336 + j0.173)

m4
freq=10.00GHz
X1.X10.Gamma2=0.212 / -15.121
impedance = Z0 * (1.502 - j0.174)

m5
freq=10.00GHz
X1.X10.Gamma4=0.433 / 60.938
impedance = Z0 * (1.060 + j0.986)

m6
freq=10.00GHz
X1.X10.Gamma3=0.366 / -65.852
impedance = Z0 * (1.038 - j0.800)

Even though the reflection coefficients are >1 over some of the frequency range, the conditions for oscillation are not satisfied.

Notes

This amplifier design is from the $HPEESOF_DIR/examples/MW_Ckts/MMIC_Amp_wrk example.

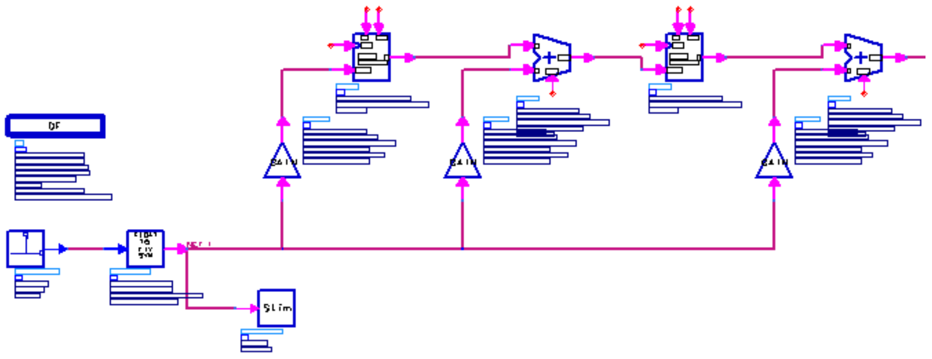
FIR Digital Filter

Location: $HPEESOF_DIR/examples/Tutorial/cdmafilter_wrk

Objective

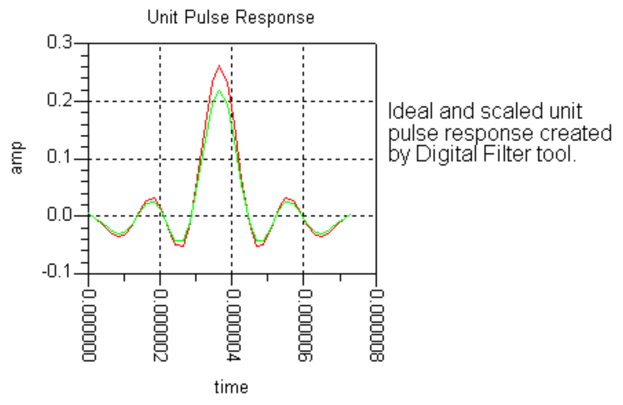
This example shows the simulation of an FIR digital filter.

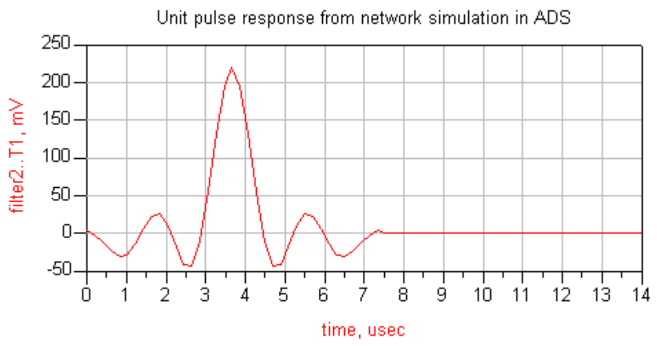
Setup



First few sections of digital filter and simulation controller

Analysis





Generate Various Modulated Sources

Location: $HPEESOF_DIR/examples/Tutorial/ModSources_wrk

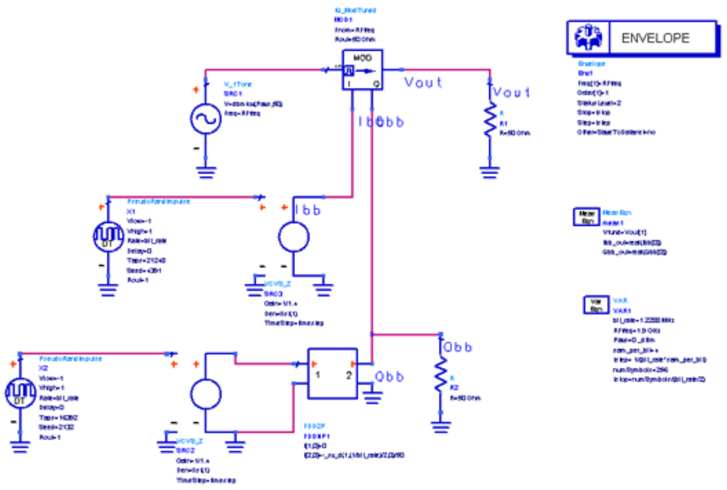
Objective

This example shows how to generate various modulated sources, including frequency-shift keying (FSK), quadrature amplitude modulation (QAM), quadrature phase-shift keying (QPSK) etc. Circuit Envelope simulation is used to check the generated modulated sources.

Setup

1. "FSK_2level" generates two two-level FSK sources, one with raised cosine filtering and one without. The corresponding data display shows the output spectra of the modulated signals.
2. "FSK_4level" generates two four-level FSK sources, one with raised cosine filtering and one without. The corresponding data display shows the output spectra of the modulated signals.
3. "QAM_16" generates a QAM signal with 16 symbol states. This example could be extended to generate higher-order QAM signals. The corresponding data displays show the modulated output spectrum, a trajectory and constellation diagram, the average signal power, a plot of peak power versus time, and the adjacent-channel

power ratios (ACPR).
4. "Pi4DQPSK" generates a signal with differential QPSK. In this example, the modulation and filtering correspond to the NADC specification. The corresponding data displays show the output spectrum, constellation diagram, and adjacent-channel power calculations, as well as the average signal power and a plot of peak power versus time.
5. "IS95RevLinkSrc" generates a signal with offset quadrature phase-shift keying (OQPSK) and baseband filtering that corresponds to the IS95 (CDMA) specification.
6. "IS95RevLinkSrc2" generates a signal with OQPSK also. The difference is that it uses baseband I and Q data that were generated using finite impulse response (FIR) filters with more taps, and consequently it has much lower residual ACPR, and should be more useful for testing amplifiers and other components.
7. "IS95FwdLinkSrc" is identical to "IS95RevLinkSrc2", except that the modulation is QPSK, which is used in the forward link, from base station to mobile.
8. "IS95FilterTest" simulates the impulse response of the baseband filter defined by the IS95 specification.



Analysis

Figure 1: Spectrum of generated signal

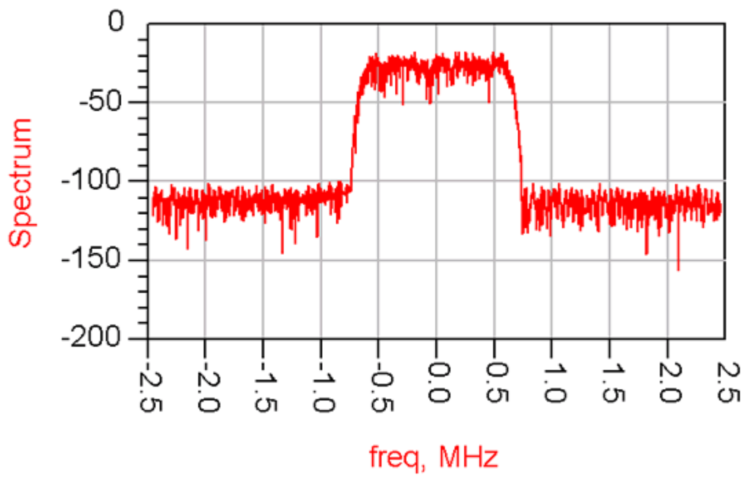


Figure 2: Signal trajectory

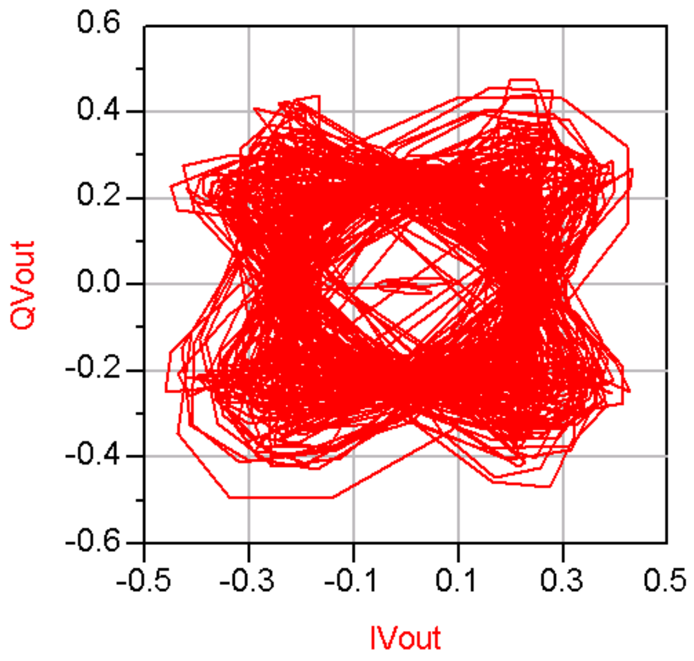
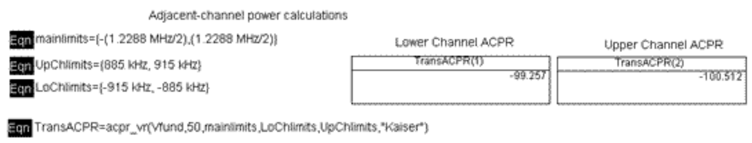


Figure 3: ACPR calculations



Adjacent-channel power calculations

```
Eqn mainlimits={-(1.2288 MHz/2),(1.2288 MHz/2)}
Eqn UpChlimits={885 kHz, 915 kHz}
Eqn LoChlimits={-915 kHz, -885 kHz}

Eqn TransACPR=acpr_vr(Vfund,50,mainlimits,LoChlimits,UpChlimits,"Kaiser")
```

Lower Channel ACPR	Upper Channel ACPR
TransACPR(1)	TransACPR(2)
-99.257	-100.512

Notes

- Simulation controller used: Envelope.
- Note that it is possible to generate a signal with arbitrary modulation by using an IQ modulator and VtDataset sources. (IS95RevLinkSrc2 and IS95FwdLinkSrc are both examples of this.) These sources allow the user to use time-domain signals generated in a different simulation (even using the Agilent Ptolemy simulator) in an Envelope simulation. Cosimulation can be used instead of these sources.
- See also the *NStateModulators* (examples) example workspace in the Tutorial example directory for additional ways of generating modulated signals.

Learn Tuning an Elliptic Filter, a Dynamic Load Line, and a Microstrip Bandpass Filter

Location: $HPEESOF_DIR/examples/Tutorial/Learn_Tune_wrk

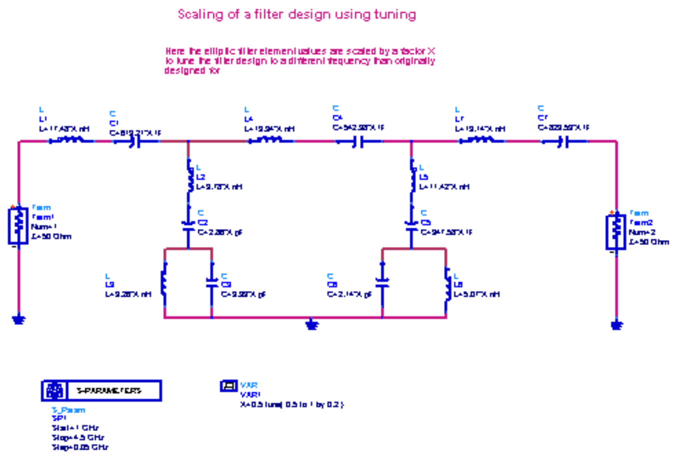
Objective

This example shows how to use the Advanced Design System (ADS) tuning feature to optimize your designs.

Setup

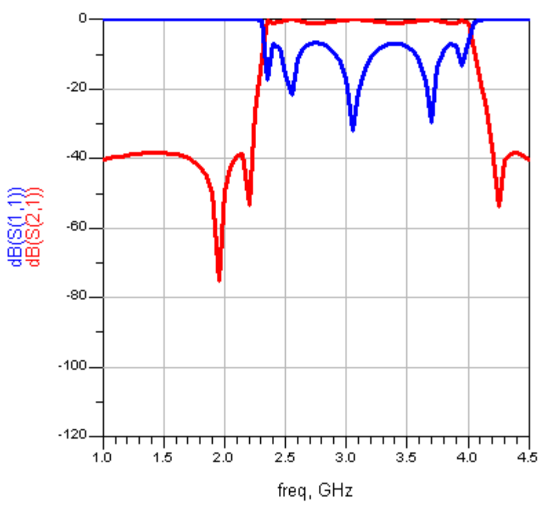
Elliptic Filter Scaling
"Elliptic_Filter_Scaling" demonstrate the tuning of an elliptic filter. The scaling coefficient "X" can be tuned to optimize filter performance at a different frequency.

Figure 1: Elliptic Filter Schematic


Scaling of a filter design using tuning
Here the elliptic filter element values are scaled by a factor X to tune the filter design to a different frequency than originally designed for
L L1 L=17.43*X nH
C C1 C=819.21*X fF
L L4 L=19.94*X nH
C C4 C=542.93*X fF
L L7 L=19.14*X nH
C C7 C=323.59*X fF
L L2 L=9.73*X nH
C C2 C=2.28*X pF
L L5 L=11.42*X nH
C C5 C=947.53*X fF
Term Term1 Num=1 Z=50 Ohm
Term Term2 Num=2 Z=50 Ohm
L L3 L=9.28*X nH
C C3 C=9.28*X pF
C C6 C=2.14*X pF
L L6 L=5.01*X nH
S-PARAMETERS
S_Param SP1 Start=1 GHz Stop=4.5 GHz Step=0.05 GHz
VAR VAR1 X=0.5 tune{ 0.5 to 1 by 0.2 }

"Elliptic_Filter_Scaling.dds" plots the elliptic filter response before tuning.

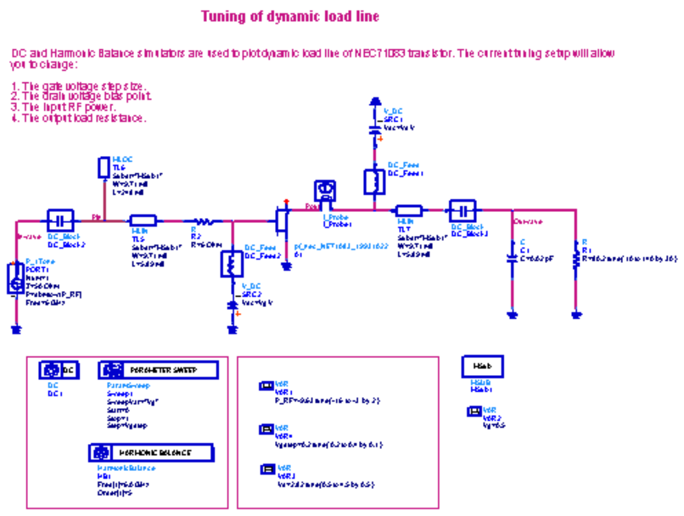
Figure 2: Elliptic Filter Response



Dynamic Load Line Simulation

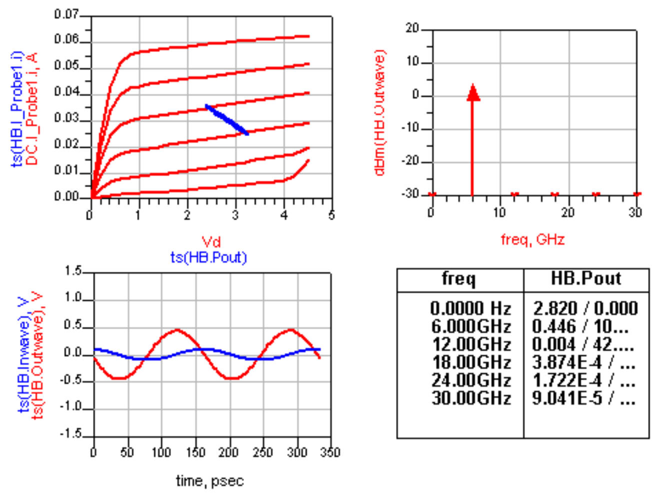
"Dynamic_Load_Line" demonstrate the dynamic load line simulation of a non linear circuit. Parameters such as drain voltage, input power, output resistance, and gate voltage step size can be tuned to optimize dynamic load line performance.

Figure 3: Dynamic Load Line Simulation


Tuning of dynamic load line
DC and Harmonic Balance simulators are used to plot dynamic load line of NEC71083 transistor. The current tuning setup will allow you to change:
1. The gate voltage step size.
2. The drain voltage bias point.
3. The input RF power.
4. The output load resistance.
V_DC
DC_Feed
I_Probe
DC_Block
PARAMETER SWEEP
HARMONIC BALANCE
MSub

"Dynamic_Load_Line.dds" shows dynamic load line plots.

Figure 4: Dynamic Load Line Plots



freq	HB.Pout
0.0000 Hz	2.820 / 0.000
6.000GHz	0.446 / 10...
12.00GHz	0.004 / 42....
18.00GHz	3.874E-4 / ...
24.00GHz	1.722E-4 / ...
30.00GHz	9.041E-5 / ...

Tuning of a Microstrip Filter

"tune_example" shows the schematic of a side coupled microstrip bandpass filter. Filter response before tune is shown in "before_tune.dds", after tune is plotted in "spar_sim.dds".

Figure 5: Microstrip Filter Schematic

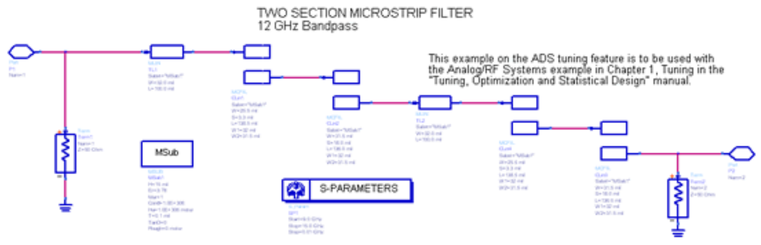

TWO SECTION MICROSTRIP FILTER
12 GHz Bandpass
This example on the ADS tuning feature is to be used with the Analog/RF Systems example in Chapter 1, Tuning in the "Tuning, Optimization and Statistical Design" manual.
MSub
S-PARAMETERS

Figure 6: Filter response before tuning

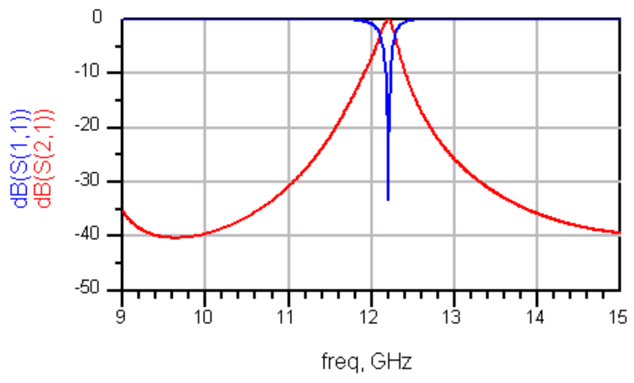
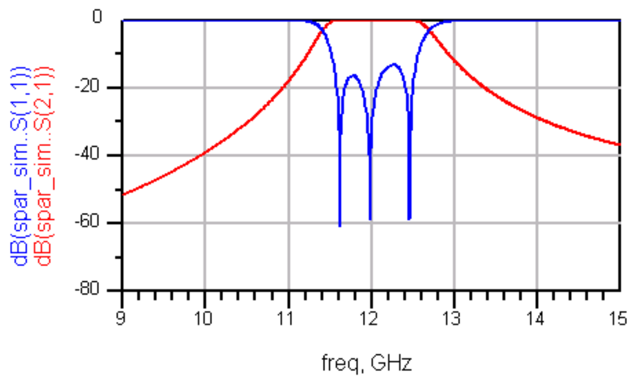


Figure 7: Filter response after tuning



Analysis

Notes

- This example on the ADS tuning feature is to be used with the Analog/RF Systems example in Tuning in the "Tuning, Optimization and Statistical Design" documentation.

LSSP (Large-Signal S-Parameters), Basic Example

Location: $HPEESOF_DIR/examples/Tutorial/LSSP_test_wrk

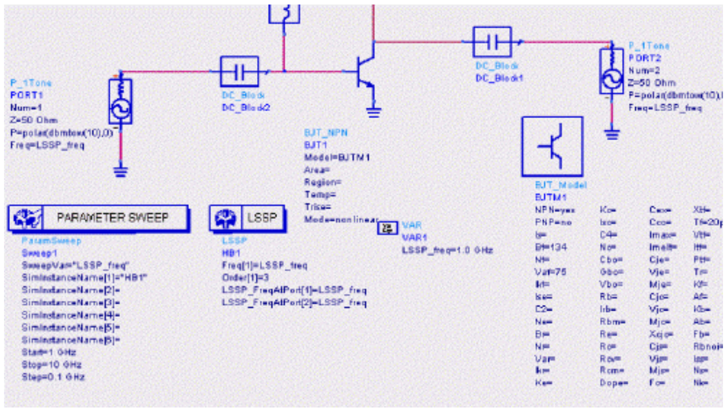
Objective

This tutorial example workspace demonstrates the use of Large Signal S-Parameter simulation.

Setup

LSSP_test shows a LSSP simulation set up, and the results are displayed in LSSP_test.dds.

Figure 1: Swept-frequency Large-Signal S-Parameter simulation setup*

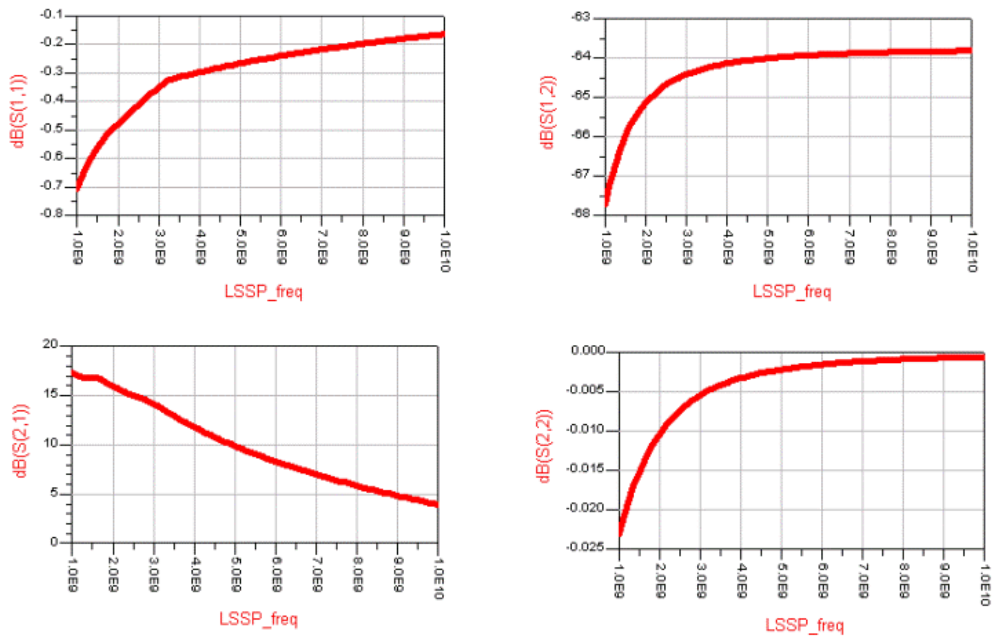


```
P_1Tone
PORT1
Num=1
Z=50 Ohm
P=polar(dbmtow(10),0)
Freq=LSSP_freq

DC_Block
DC_Block2

BJT_NPN
BJT1
Model=BJTM1
Area=
Region=
Temp=
Trise=
Mode=nonlinear

DC_Block
DC_Block1

P_1Tone
PORT2
Num=2
Z=50 Ohm
P=polar(dbmtow(10),
Freq=LSSP_freq

ParamSweep
Sweep1
SweepVar="LSSP_freq"
SimInstanceName[1]="HB1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1 GHz
Stop=10 GHz
Step=0.1 GHz

LSSP
HB1
Freq[1]=LSSP_freq
Order[1]=3
LSSP_FreqAtPort[1]=LSSP_freq
LSSP_FreqAtPort[2]=LSSP_freq

VAR
VAR1
LSSP_freq=1.0 GHz

BJT_Model
BJTM1
NPN=yes    Kc=      Cex=     Xtf=
PNP=no     Isc=     Cco=     Tf=20p
Is=        C4=      Imax=    Vtf=
Bf=134     Nc=      Imelt=   Itf=
Nf=        Cbo=     Cje=     Ptf=
Vaf=75     Gbo=     Vje=     Tr=
Ikf=       Vbo=     Mje=     Kf=
Ise=       Rb=      Cjc=     Af=
C2=        Irb=     Vjc=     Kb=
Ne=        Rbm=     Mjc=     Ab=
Br=        Re=      Xcjc=    Fb=
Nr=        Rc=      Cjs=     Rbnoi
Var=       Rcv=     Vjs=     Iss=
Ikr=       Rcm=     Mjs=     Ns=
Ke=        Dope=    Fc=      Nk=
```

SP_test is simply a small-signal S-Parameter simulation of the same circuit.
The simulation results are displayed in SP_test.dds.

Analysis

Figure 2: Large-Signal S-Parameter simulation results



Notes

Large-Signal S-Parameter simulation may give erroneous results for the output impedance of an amplifier. This is because amplifiers are normally driven with a signal at the input, not a large signal at the output with nothing at the input. To determine the output impedance of an amplifier while the input is being driven with a large signal, see examples/MW_Ckts/LargeSigAmp_wrk/Stab_vs_freq_pwr.

Mixer2 Model Validation Example File

Location: $HPEESOF_DIR/examples/Tutorial/Mixer2_Example_wrk

Objective

The purpose of this workspace is to validate the Mixer2 system model. This model was introduced as a replacement for the Mixer system model for everything but FCAC/Budget analysis. In many cases, Mixer and Mixer2 give the same results. In other cases such as Tran analysis, multiple RF tones and complex ConvGain values, Mixer2 gives better results than Mixer. In other cases such as reverse conversion, certain feedthru terms, finite sideband suppression, image rejection, AM-to-PM and LO limiting, Mixer2 supports behaviors that Mixer does not. This makes Mixer2 the recommended system model for mixer modeling for anything but FCAC/Budget analysis.

Setup

The following designs (**), datasets (**.ds) and data displays (*.dds) are included in this workspace:

1. Mixer2_DC: Demonstrates the use of Mixer2 for DC analysis. Mixer and Mixer2 agree.
2. Mixer2_Tran: Demonstrates the use of Mixer2 for Tran analysis. Mixer does not provide sideband suppression for Tran analysis.
3. Mixer2_HB: Demonstrates the use of Mixer2 for HB analysis. Mixer and Mixer2 agree.
4. Mixer2_CE: Demonstrates the use of Mixer2 for CE analysis. Mixer and Mixer2 agree.
5. Mixer2_ConvGain_S11: Demonstrates how Mixer2 works when ConvGain and S11 are specified. Mixer and Mixer2 agree.
6. Mixer2_ConvGain_S21_S31: Demonstrates how Mixer2 works when S21 and S31 are specified. Mixer and Mixer2 agree for S21, but Mixer does not support S31.
7. Mixer2_RevConvGain_S12: Demonstrates how Mixer2 works when RevConvGain and S12 are specified. Mixer does not support RevConvGain.
8. Mixer2_RF1_plus_minus_RF2: Demonstrates how Mixer2 passes sums/differences of two RF tones provided SP21 and SOI are set properly. Sums/differences of two RF tones are not passed for Mixer.
9. Mixer2_Image_SideBand_Amount: Demonstrates the amount of image rejection and sideband suppression provided by the various image/sideband options for Mixer2. Most of these cases are not supported for Mixer.
10. Mixer2_Image_SideBand_ConvGain: Demonstrates how the phase of ConvGain affects the phase of the image rejection and sideband suppression provided by the various image/sideband options for Mixer2. The phase of ConvGain is handled differently for Mixer, leading to confusing results for Mixer.
11. Mixer2_AM2PM: Demonstrates the use of Mixer2 with AM2PM specified. AM2PM is demonstrated for a mixer with no compression but is supported for all compression options. AM-to-PM is not supported for Mixer.
12. Mixer2_TOI_GainCompPower_GainComp_HB1tone: Demonstrates the use of Mixer2 with compression parameters specifed. Validates GainCompPower and GainComp. Mixer and Mixer2 agree.
13. Mixer2_TOI_GainCompPower_GainComp_HB2tone: Demonstrates the use of Mixer2 with compression parameters specifed. Validates TOI. Mixer and Mixer2 agree.
14. Mixer2_PminLO: Demonstrates how PminLO affects the IF output of Mixer2. Mixer and Mixer2 agree.
15. Mixer2_Nonharmonically_Related_LO_Frequencies: Demonstrates that large DetBW values (original Mixer2 LO limiting) should be used for LO limiting when the LO port has two nonharmonically related tones. Also demonstrates that Mixer2 can do the same LO limiting that Mixer does, making Mixer2 a superset of Mixer as far as LO limiting is concerned.
16. Mixer2_Harmonically_Related_LO_Frequencies: Demonstrates that small DetBW values (Mixer LO limiting) should be used for LO limiting when the LO port has two harmonically related tones. Also demonstrates that Mixer2 can do the same LO limiting that Mixer does, making Mixer2 a superset of Mixer as far as LO limiting is concerned.
17. Mixer2_GainCompFile_GCOMP1: Demonstrates the use of Mixer2 with a GCOMP1 block specifying the compression. Mixer and Mixer2 agree.
18. Mixer2_GainCompFile_GCOMP7: Demonstrates the use of Mixer2 with a GCOMP7 block specifying the compression. Since the polynomial fitting for Mixer and Mixer2 is not the same, Mixer and Mixer2 disagree.
19. Mixer2_NonlinearNoise_NF_BOTH: Demonstrates the use of Mixer2 for nonlinear noise with NF set and SideBand=BOTH. Mixer and Mixer2 agree.
20. Mixer2_NonlinearNoise_NF_LOWER: Demonstrates the use of Mixer2 for nonlinear noise with NF set and SideBand=LOWER. Mixer and Mixer2 agree.
21. Mixer2_NonlinearNoise_NF_LOWER_IMAGE_REJECTION: Demonstrates the use of Mixer2 for nonlinear noise with NF set and SideBand=LOWER IMAGE REJECTION. Image rejection is not supported for Mixer.
22. Mixer2_NonlinearNoise_NF_UPPER: Demonstrates the use of Mixer2 for nonlinear noise with NF set and SideBand=UPPER. Mixer and Mixer2 agree.
23. Mixer2_NonlinearNoise_NF_UPPER_IMAGE_REJECTION: Demonstrates the use of Mixer2 for nonlinear noise with NF set and SideBand=UPPER IMAGE REJECTION. Image rejection is not supported for Mixer.
24. Mixer2_NonlinearNoise_NF_UPPER_IMAGE_REJECTION_PowerSweep: Demonstrates how Mixer2 noise properties degrade in compression for a mixer with NF set and SideBand=UPPER IMAGE REJECTION. Image rejection is not supported for Mixer.
25. Mixer2_NonlinearNoise_NFminSoptRn_BOTH: Demonstrates the use of Mixer2 for nonlinear noise with (NFmin,Sopt,Rn) set and SideBand=BOTH.Mixer and Mixer2 agree.
26. Mixer2_NonlinearNoise_NFminSoptRn_LOWER: Demonstrates the use of Mixer2 for nonlinear noise with (NFmin,Sopt,Rn) set and SideBand=LOWER.Mixer and Mixer2 agree.

27. Mixer2_NonlinearNoise_NFminSoptRn_LOWER_IMAGE_REJECTION: Demonstrates the use of Mixer2 for nonlinear noise with (NFmin,Sopt,Rn) set andSideBand=LOWER IMAGE REJECTION. Image rejection is not supported for Mixer.
28. Mixer2_NonlinearNoise_NFminSoptRn_UPPER: Demonstrates the use of Mixer2 for nonlinear noise with (NFmin,Sopt,Rn) set and SideBand=UPPER.Mixer and Mixer2 agree.
29. Mixer2_NonlinearNoise_NFminSoptRn_UPPER_IMAGE_REJECTION: Demonstrates the use of Mixer2 for nonlinear noise with (NFmin,Sopt,Rn) set andSideBand=UPPER IMAGE REJECTION. Image rejection is not supported for Mixer.
30. Mixer2_NonlinearNoise_NFminSoptRn_UPPER_IMAGE_REJECTION_PowerSweep: Demonstrates how Mixer2 noise properties degrade in compression for a mixer with (NFmin,Sopt,Rn) set and SideBand=UPPER IMAGE REJECTION. Image rejection is not supported for Mixer.

Notes

For more information about Mixer2, see *Mixer2 (RF System Mixer)* (ccsys).

Noise Power Ratio Simulation

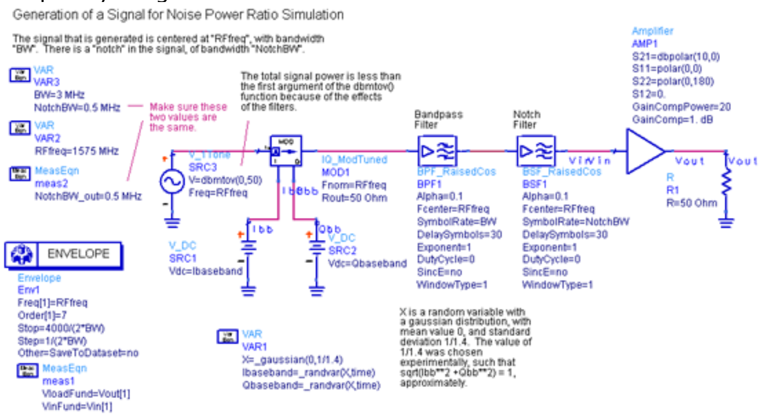
Location: $HPEESOF_DIR/examples/Tutorial/NoisePowerRatio_wrk

Objective

This example shows two methods of simulating noise power ratio and a method of characterizing the distortion generated by an amplifier.

Setup

1. NarrowBandNPR simulates noise power ratio by generating a narrow band (relative to the carrier frequency) noise signal via a bandpass filter and a bandstop filter. The Envelope simulator is used, and the noise signal is generated via two baseband noise sources and an I-Q modulator.
2. BPF_wNotchSparams simulates the S-Parameters of the filters used to band-limit the noise and generate the notch filter.
3. MultiTone_Test shows how to set-up and simulate an amplifier with multiple input tones. This sort of simulation would be of interest to a CATV system or amplifier designer. It uses harmonic balance and a Monte Carlo simulation to model multiple tones, and is appropriate when you want to simulate a number of tones over a broad frequency range.



The signal that is generated is centered at "RFfreq", with bandwidth "BW". There is a "notch" in the signal, of bandwidth "NotchBW".

VAR
VAR3
BW=3 MHz
NotchBW=0.5 MHz

Make sure these two values are the same.

VAR
VAR2
RFfreq=1575 MHz

MeasEqn
meas2
NotchBW_out=0.5 MHz

The total signal power is less than the first argument of the dbmtov() function because of the effects of the filters.

V_1Tone
SRC3
V=dbmtov(0,50)
Freq=RFfreq

IQ_ModTuned
MOD1
Fnom=RFfreq
Rout=50 Ohm

BPF_RaisedCos
BPF1
Alpha=0.1
Fcenter=RFfreq
SymbolRate=BW
DelaySymbols=30
Exponent=1
DutyCycle=0
SincE=no
WindowType=1

BSF_RaisedCos
BSF1
Alpha=0.1
Fcenter=RFfreq
SymbolRate=NotchBW
DelaySymbols=30
Exponent=1
DutyCycle=0
SincE=no
WindowType=1

Amplifier
AMP1
S21=dbpolar(10,0)
S11=polar(0,0)
S22=polar(0,180)
S12=0.
GainCompPower=20
GainComp=1. dB

R
R1
R=50 Ohm

V_DC
SRC1
Vdc=Ibaseband

V_DC
SRC2
Vdc=Qbaseband

Envelope
Env1
Freq[1]=RFfreq
Order[1]=7
Stop=4000/(2*BW)
Step=1/(2*BW)
Other=SaveToDataset=no

MeasEqn
meas1
VloadFund=Vout[1]
VinFund=Vin[1]

VAR
VAR1
X=_gaussian(0,1/1.4)
Ibaseband=_randvar(X,time)
Qbaseband=_randvar(X,time)

X is a random variable with a gaussian distribution, with mean value 0, and standard deviation 1/1.4. The value of 1/1.4 was chosen experimentally, such that sqrt(Ibb**2 +Qbb**2) = 1, approximately.

Analysis

Figure 1: Spectra at the Input and Output of the Amplifier in NarrowBandNPR design

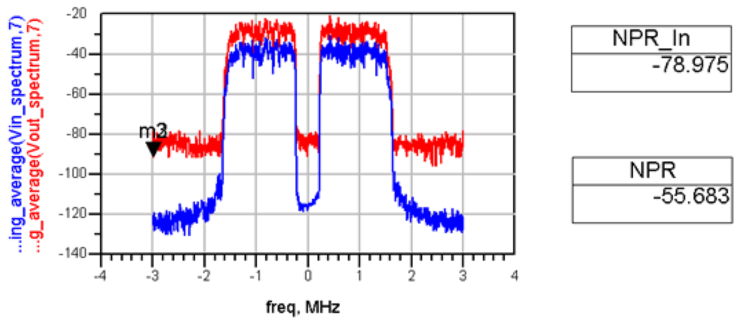


Figure 2: Transmission properties of the notch filter used in the NarrowBandNPR design

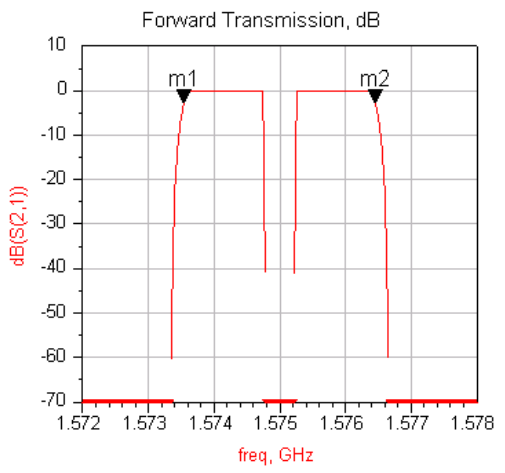
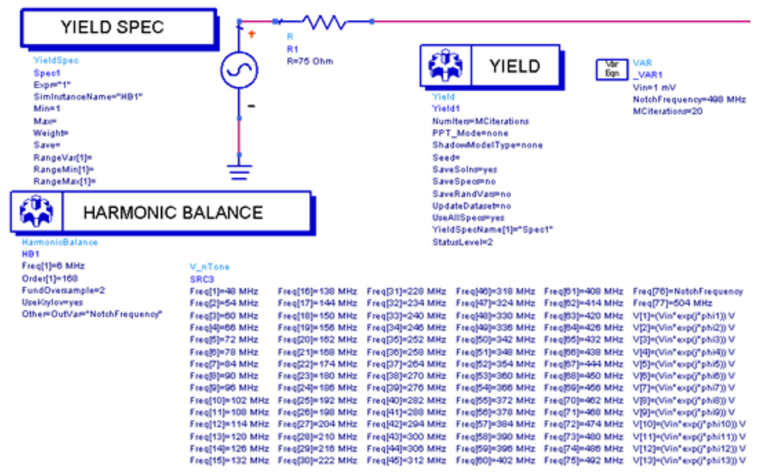
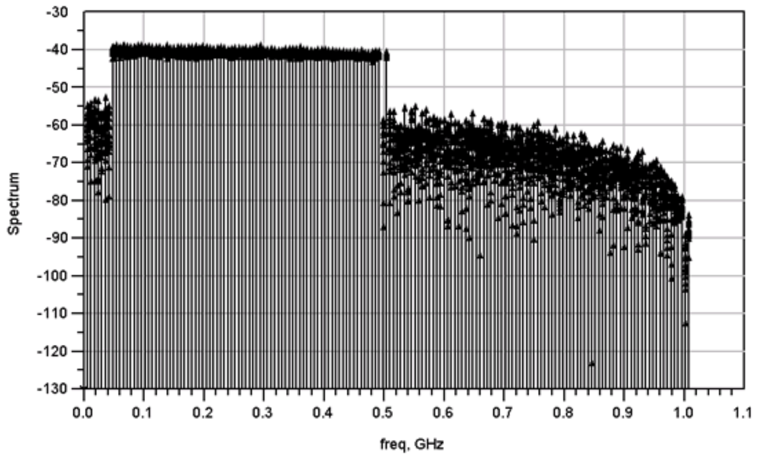


Figure 3: Amplifier distortion simulation using multiple tones



```
YieldSpec
Spec1
Expr="1"
SimInstanceName="HB1"
Min=1
Max=
Weight=
Save=
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=

R
R1
R=75 Ohm

Yield
Yield1
NumIters=MCiterations
PPT_Mode=none
ShadowModelType=none
Seed=
SaveSolns=yes
SaveSpecs=no
SaveRandVars=no
UpdateDataset=no
UseAllSpecs=yes
YieldSpecName[1]="Spec1"
StatusLevel=2

VAR
_VAR1
Vin=1 mV
NotchFrequency=498 MHz
MCiterations=20

HarmonicBalance
HB1
Freq[1]=6 MHz
Order[1]=168
FundOversample=2
UseKrylov=yes
Other=OutVar="NotchFrequency"

V_nTone
SRC3
Freq[1]=48 MHz    Freq[16]=138 MHz  Freq[31]=228 MHz  Freq[46]=318 MHz  Freq[61]=408 MHz  Freq[76]=NotchFrequency
Freq[2]=54 MHz    Freq[17]=144 MHz  Freq[32]=234 MHz  Freq[47]=324 MHz  Freq[62]=414 MHz  Freq[77]=504 MHz
Freq[3]=60 MHz    Freq[18]=150 MHz  Freq[33]=240 MHz  Freq[48]=330 MHz  Freq[63]=420 MHz  V[1]=(Vin*exp(j*phi1)) V
Freq[4]=66 MHz    Freq[19]=156 MHz  Freq[34]=246 MHz  Freq[49]=336 MHz  Freq[64]=426 MHz  V[2]=(Vin*exp(j*phi2)) V
Freq[5]=72 MHz    Freq[20]=162 MHz  Freq[35]=252 MHz  Freq[50]=342 MHz  Freq[65]=432 MHz  V[3]=(Vin*exp(j*phi3)) V
Freq[6]=78 MHz    Freq[21]=168 MHz  Freq[36]=258 MHz  Freq[51]=348 MHz  Freq[66]=438 MHz  V[4]=(Vin*exp(j*phi4)) V
Freq[7]=84 MHz    Freq[22]=174 MHz  Freq[37]=264 MHz  Freq[52]=354 MHz  Freq[67]=444 MHz  V[5]=(Vin*exp(j*phi5)) V
Freq[8]=90 MHz    Freq[23]=180 MHz  Freq[38]=270 MHz  Freq[53]=360 MHz  Freq[68]=450 MHz  V[6]=(Vin*exp(j*phi6)) V
Freq[9]=96 MHz    Freq[24]=186 MHz  Freq[39]=276 MHz  Freq[54]=366 MHz  Freq[69]=456 MHz  V[7]=(Vin*exp(j*phi7)) V
Freq[10]=102 MHz  Freq[25]=192 MHz  Freq[40]=282 MHz  Freq[55]=372 MHz  Freq[70]=462 MHz  V[8]=(Vin*exp(j*phi8)) V
Freq[11]=108 MHz  Freq[26]=198 MHz  Freq[41]=288 MHz  Freq[56]=378 MHz  Freq[71]=468 MHz  V[9]=(Vin*exp(j*phi9)) V
Freq[12]=114 MHz  Freq[27]=204 MHz  Freq[42]=294 MHz  Freq[57]=384 MHz  Freq[72]=474 MHz  V[10]=(Vin*exp(j*phi10)) V
Freq[13]=120 MHz  Freq[28]=210 MHz  Freq[43]=300 MHz  Freq[58]=390 MHz  Freq[73]=480 MHz  V[11]=(Vin*exp(j*phi11)) V
Freq[14]=126 MHz  Freq[29]=216 MHz  Freq[44]=306 MHz  Freq[59]=396 MHz  Freq[74]=486 MHz  V[12]=(Vin*exp(j*phi12)) V
Freq[15]=132 MHz  Freq[30]=222 MHz  Freq[45]=312 MHz  Freq[60]=402 MHz  Freq[75]=492 MHz  V[13]=(Vin*exp(j*phi13)) V
```

Figure 4: Output spectrum after Monte Carlo simulation



There is an empty channel at NotchFrequency (498MHz). If there were no distortion in the circuit, the notch depth (db of mean notch depth divided by signal depth) would be infinite. The smaller of this notch depth, the greater the distortion of the circuit.

Noise Simulation in Envelope Analysis

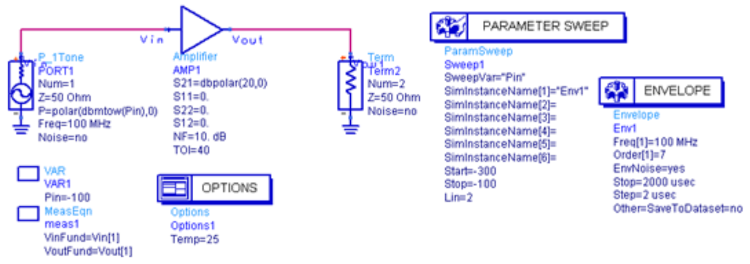
Location: $HPEESOF_DIR/examples/Tutorial/Amp_wNoiseEnv_wrk

Objective

This example illustrates how to do a noise analysis using an Envelope simulator.

Setup

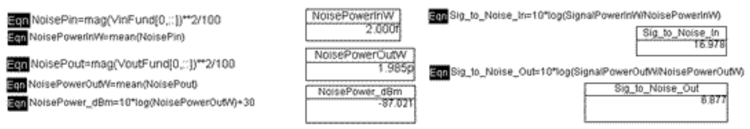
1. In "AmpTest", the Envelope simulator is used to test noise degradation through an amplifier. The noise is limited to the Envelope bandwidth = 1/(simulation time step). Two simulations are run. One with the signal power off (-300 dBm), to model the noise by itself, and one with the signal on (-100 dBm.) With this Envelope simulation, noise is modeled as a time-domain signal, and there are no small-signal assumptions.
2. "NoiseTest" calculates the available noise power from the noise source using an Envelope analysis.



```
P_1Tone
PORT1
Num=1
Z=50 Ohm
P=polar(dbmtow(Pin),0)
Freq=100 MHz
Noise=no

Amplifier
AMP1
S21=dbpolar(20,0)
S11=0.
S22=0.
S12=0.
NF=10. dB
TOI=40

Term
Term2
Num=2
Z=50 Ohm
Noise=no

ParamSweep
Sweep1
SweepVar="Pin"
SimInstanceName[1]="Env1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=-300
Stop=-100
Lin=2

Envelope
Env1
Freq[1]=100 MHz
Order[1]=7
EnvNoise=yes
Stop=2000 usec
Step=2 usec
Other=SaveToDataset=no

VAR
VAR1
Pin=-100

MeasEqn
meas1
VinFund=Vin[1]
VoutFund=Vout[1]

Options
Options1
Temp=25
```

Analysis

Figure 1: Noise analysis result



```
Eqn NoisePin=mag(VinFund[0,::])**2/100
Eqn NoisePowerInW=mean(NoisePin)
Eqn NoisePout=mag(VoutFund[0,::])**2/100
Eqn NoisePowerOutW=mean(NoisePout)
Eqn NoisePower_dBm=10*log(NoisePowerOutW)+30

NoisePowerInW
2.000f
NoisePowerOutW
1.985p
NoisePower_dBm
-87.021

Eqn Sig_to_Noise_In=10*log(SignalPowerInW/NoisePowerInW)
Sig_to_Noise_In
16.978
Eqn Sig_to_Noise_Out=10*log(SignalPowerOutW/NoisePowerOutW)
Sig_to_Noise_Out
6.877
```

The signal-to-noise ratio degrades by about 10 dB, which is expected since the noise figure is 10 dB.

Notes

- The available noise power in Watts per Hz is k*T, where T is the simulation temperature in degrees Kelvin. For a simulation temperature of 25 C, the available noise power is k*(25+273.15), or -173.8 dBm/Hz. This is verified in the "NoiseTest".

Optimization and Parameter Sweeps Using DSP Schematic

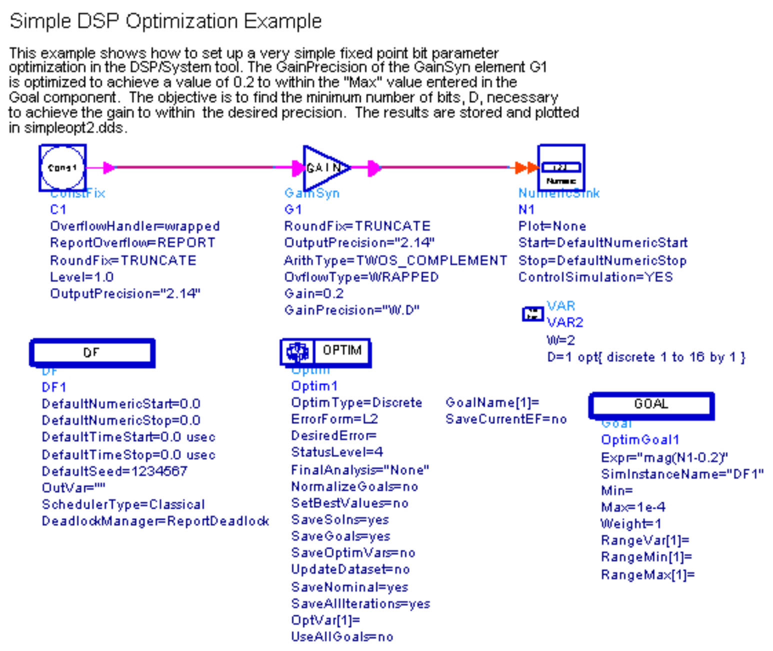
Location: $HPEESOF_DIR/examples/Tutorial/dspopt_wrk

Objective

This example shows how to setup two simple optimizations and two simple parameter sweeps within a Digital Signal Processing (Ptolemy) schematic window. The optimization setup is discussed in *Using Nominal Optimization* (ptolemy). However, this example has been improved and simplified, so it may not match the documentation exactly.

Setup

Figure 1:Bitwidth optimization setup



Simple DSP Optimization Example

This example shows how to set up a very simple fixed point bit parameter optimization in the DSP/System tool. The GainPrecision of the GainSyn element G1 is optimized to achieve a value of 0.2 to within the "Max" value entered in the Goal component. The objective is to find the minimum number of bits, D, necessary to achieve the gain to within the desired precision. The results are stored and plotted in simpleopt2.dds.

```
ConstFix
C1
OverflowHandler=wrapped
ReportOverflow=REPORT
RoundFix=TRUNCATE
Level=1.0
OutputPrecision="2.14"

GainSyn
G1
RoundFix=TRUNCATE
OutputPrecision="2.14"
ArithType=TWOS_COMPLEMENT
OvflowType=WRAPPED
Gain=0.2
GainPrecision="W.D"

NumericSink
N1
Plot=None
Start=DefaultNumericStart
Stop=DefaultNumericStop
ControlSimulation=YES

VAR
VAR2
W=2
D=1 opt{ discrete 1 to 16 by 1 }

DF
DF1
DefaultNumericStart=0.0
DefaultNumericStop=0.0
DefaultTimeStart=0.0 usec
DefaultTimeStop=0.0 usec
DefaultSeed=1234567
OutVar=""
SchedulerType=Classical
DeadlockManager=ReportDeadlock

Optim
Optim1
OptimType=Discrete
ErrorForm=L2
DesiredError=
StatusLevel=4
FinalAnalysis="None"
NormalizeGoals=no
SetBestValues=no
SaveSolns=yes
SaveGoals=yes
SaveOptimVars=no
UpdateDataset=no
SaveNominal=yes
SaveAllIterations=yes
OptVar[1]=
UseAllGoals=no
GoalName[1]=
SaveCurrentEF=no

Goal
OptimGoal1
Expr="mag(N1-0.2)"
SimInstanceName="DF1"
Min=
Max=1e-4
Weight=1
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=
```

Analysis

Figure 2:Optimization results

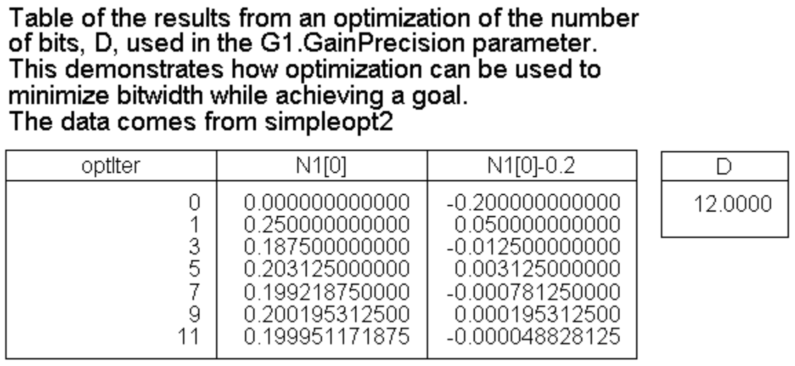


Table of the results from an optimization of the number of bits, D, used in the G1.GainPrecision parameter. This demonstrates how optimization can be used to minimize bitwidth while achieving a goal. The data comes from simpleopt2

optIter	N1[0]	N1[0]-0.2
0	0.000000000000	-0.200000000000
1	0.250000000000	0.050000000000
3	0.187500000000	-0.012500000000
5	0.203125000000	0.003125000000
7	0.199218750000	-0.000781250000
9	0.200195312500	0.000195312500
11	0.199951171875	-0.000048828125

D
12.0000

Optimization Final Analysis Demonstration

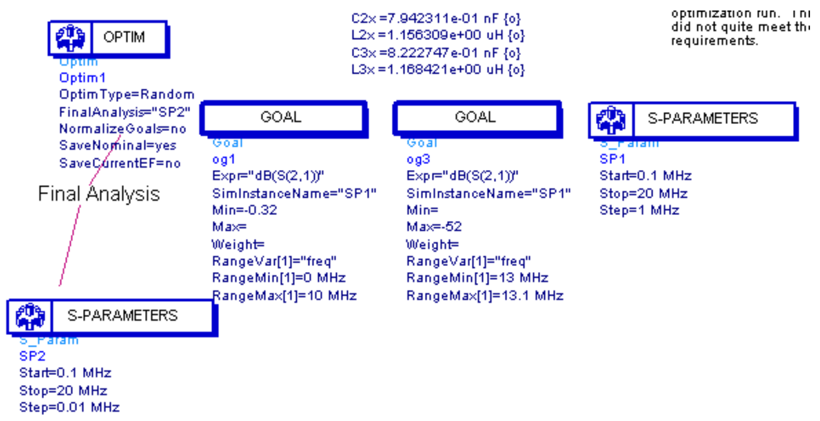
Location: $HPEESOF_DIR/examples/Tutorial/FinalAnalysis_wrk

Objective

In this example a feature called "final Analysis" in Optimization is demonstrated. Final analysis is useful when the analysis executed by the optimizer uses a different sweep grid than the one you want for your output. For example, if a coarse grid is required for optimization, but a finer grid, or a different range, is desired for output, then the analysis setup to generate this finer grid may be run after the optimization is completed, using the Final Analysis feature. This analysis can be of any analysis controller component that does not introduce circularity in analysis execution.

Setup

In LPF10MHz_final, a second controller, SP2, is specified as the final analysis in the parameter
tab of the optimization controller. The result is displayed in LPF10MHz_final.dds.

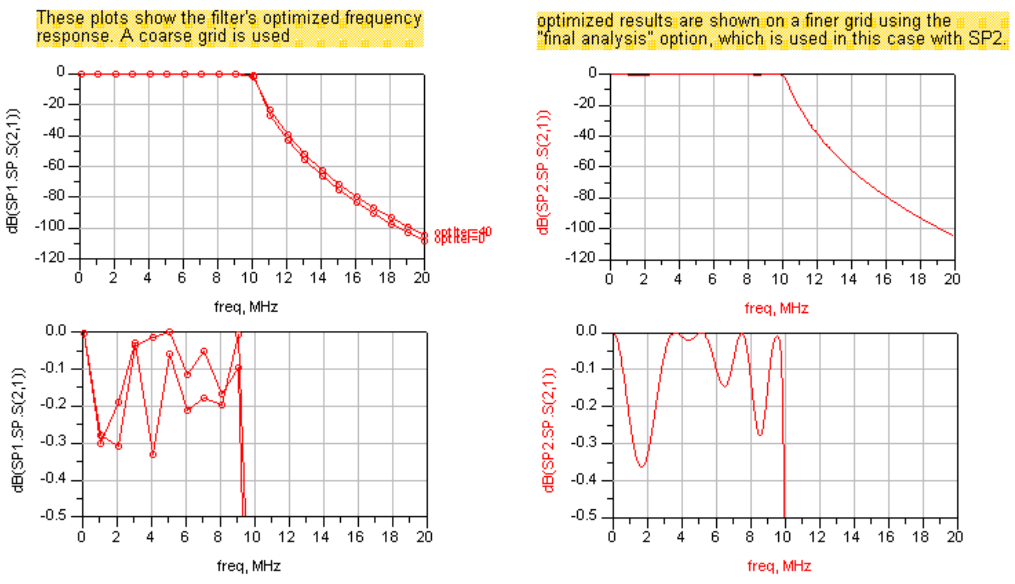


```
OPTIM
Optim
Optim1
OptimType=Random
FinalAnalysis="SP2"
NormalizeGoals=no
SaveNominal=yes
SaveCurrentEF=no

C2x =7.942311e-01 nF {o}
L2x =1.156309e+00 uH {o}
C3x =8.222747e-01 nF {o}
L3x =1.168421e+00 uH {o}

optimization run. Thi
did not quite meet th
requirements.

GOAL
Goal
og1
Expr="dB(S(2,1))"
SimInstanceName="SP1"
Min=-0.32
Max=
Weight=
RangeVar[1]="freq"
RangeMin[1]=0 MHz
RangeMax[1]=10 MHz

GOAL
Goal
og3
Expr="dB(S(2,1))"
SimInstanceName="SP1"
Min=
Max=-52
Weight=
RangeVar[1]="freq"
RangeMin[1]=13 MHz
RangeMax[1]=13.1 MHz

S-PARAMETERS
S_Param
SP1
Start=0.1 MHz
Stop=20 MHz
Step=1 MHz

Final Analysis

S-PARAMETERS
S_Param
SP2
Start=0.1 MHz
Stop=20 MHz
Step=0.01 MHz
```

The final analysis S-parameter setup has a finer sweep resolution than the one used during the optimization

In LPF10MHz_multiple_final_analysis, this feature is used for multiple controllers via a parameter sweep which drives two sweeps in two different simulation controller (SP2, and Tran1).
The plots are shown in LPF10MHz_multiple_final_analysis.dds.

Analysis



Optimization results and data from final analysis

Optimization of a Low Pass Filter

Location: $HPEESOF_DIR/examples/Tutorial/LPFoptim_wrk

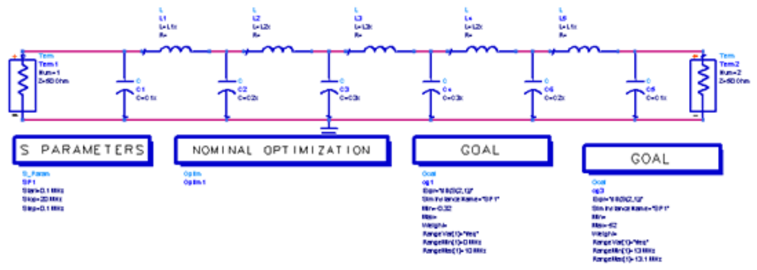
Objective

This example shows how to set-up and run a basic optimization for realizing a low pass filter.

Setup

1. "LPF1Hz" shows an ideal low-pass filter, with a 1 Hz cutoff frequency. The nominal parameter values can be changed and the optimizer can be tested to see if it can find the known best values.

2. "LPF10MHz" shows the same filter, scaled to have a 10 MHz cutoff, and to operate in a 50 ohm system.



Analysis

Figure 1: Transmission properties of the filter for various iterations

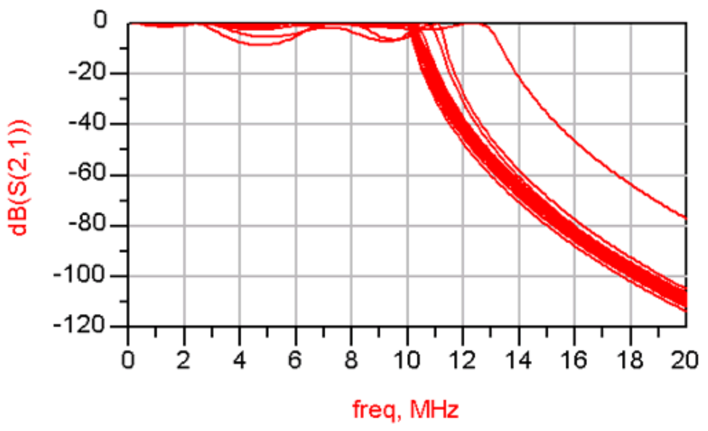
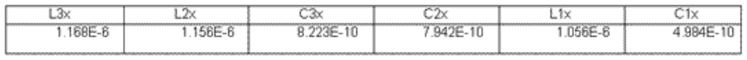


Figure 2: Optimal component values



L3x	L2x	C3x	C2x	L1x	C1x
1.168E-6	1.156E-6	8.223E-10	7.942E-10	1.056E-6	4.984E-10

Notes

- Simulation controllers used: S-Parameters, Optimization.

Optimization of An Impedance Transformation Network

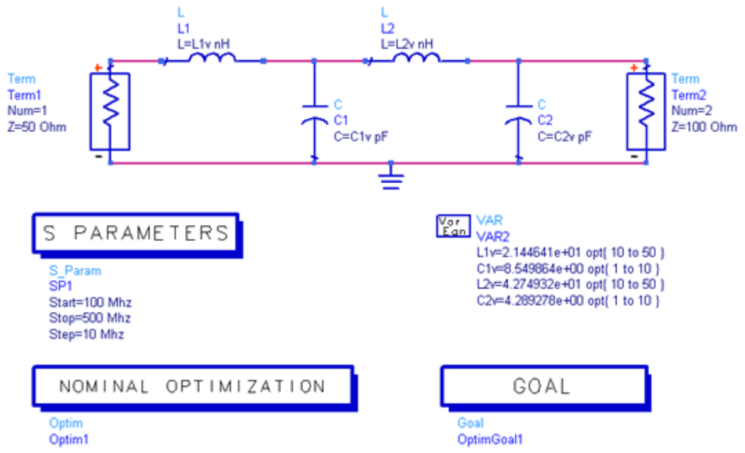
Location: $HPEESOF_DIR/examples/Tutorial/optex1_wrk

Objective

This example shows a basic optimization of an impedance transformation network.

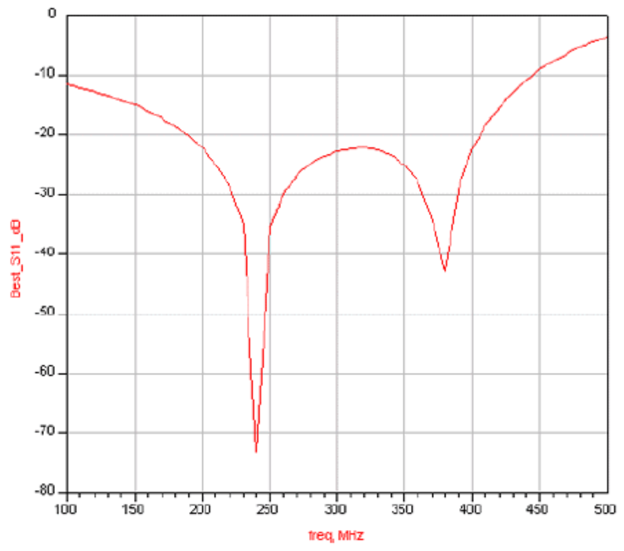
Setup

1. "optex0" simulates the reflection coefficient of the network prior to optimization.
2. "optex1" uses the gradient optimizer to improve the reflection coefficient.
3. "optex2" uses the Minimax optimizer to further improve the reflection coefficient.



Analysis

Figure 1: Optimized return loss using Minimax optimization



Notes

- Simulation controller used: S-Parameters.

Oscillator Simulations using Transient, Harmonic Balance and Envelope Simulators

Location: $HPEESOF_DIR/examples/Tutorial/Osc_Tran_HB_Env_wrk

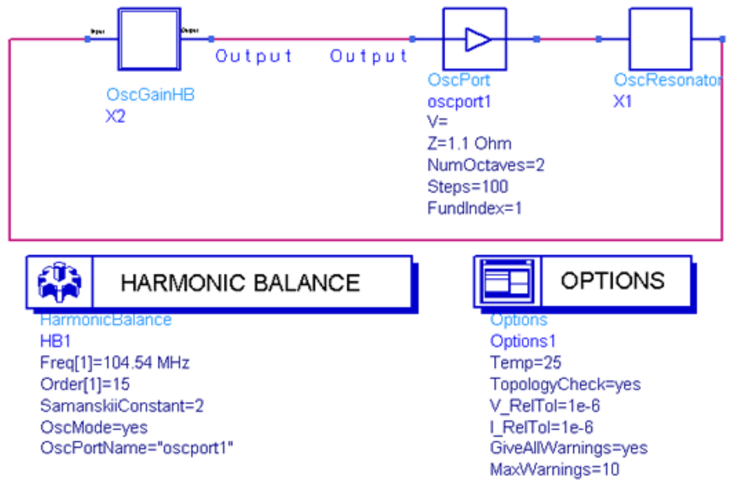
Objective

This example shows the simulation of a simple oscillator using the transient, harmonic balance and Circuit Envelope simulators.

Setup

1. "OSC_tran" shows the simulation set-up with transient. The data display shows the time-domain waveform and the oscillation building up and reaching steady state. The spectrum, computed after the waveform has reached steady state, is also shown, and compared with the harmonic balance results. Note that in the "OscGain" block, the negative supply is turned on via a voltage step. A transient signal like this is necessary to get the oscillation started, and how rapidly the oscillator turns on depends on the shape of the transient signal as well as where it is applied to the circuit. Note that it is not necessary to have an "OscPort" element in the circuit.
2. "OSC_HB" shows the harmonic balance simulation of the oscillator. This just calculates the steady-state output spectrum and frequency of oscillation. Note that the "OscGain" block has been replaced by one without the transient voltage source.

3. "OSC_Env" shows the same oscillator simulation, except using Circuit Envelope. With "ResetOsc =no", an "OscPort" element in the circuit, and no transient signal in the circuit, the simulation results are identical to the harmonic balance results.
4. "OSC_Env_wStartup" shows the Circuit Envelope simulation results when the start-up transient is simulated also. Noise generated by the resistors is enough to start the oscillation.



Analysis

Figure 1: Oscillation waveform

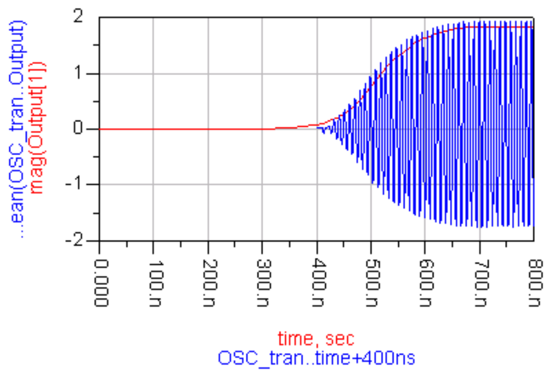
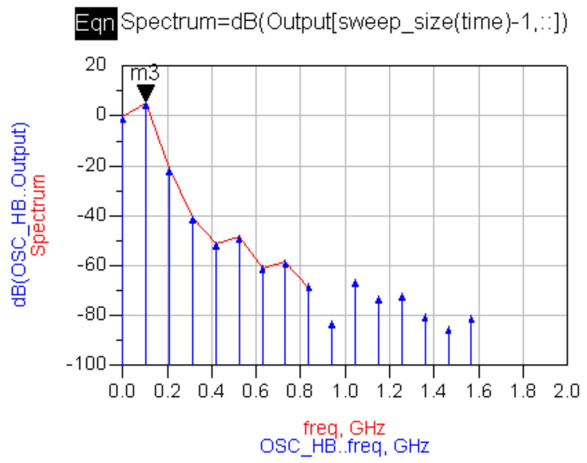


Figure 2: Spectrum



Physical Layout Connectivity Check

Location: $HPEESOF_DIR/examples/Tutorial/Learn_LayConn_wrk

Objective

This examples shows the Layconn feature that allows designers to check the physical connectivity of layouts. The physical connectivity check can be enabled by Layer Binding.

Setup

Figure 1: A sample Layer Binding

The graphic below shows an sample entry of Layer Binding.

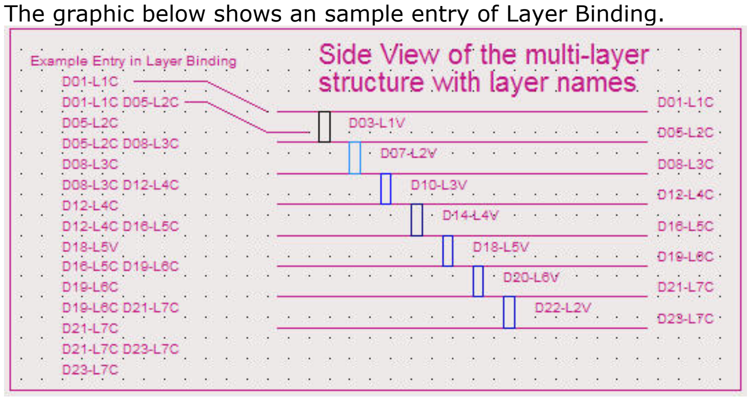


Figure 2: LayConn - Digital Board Example (Layconn_DigBoard)

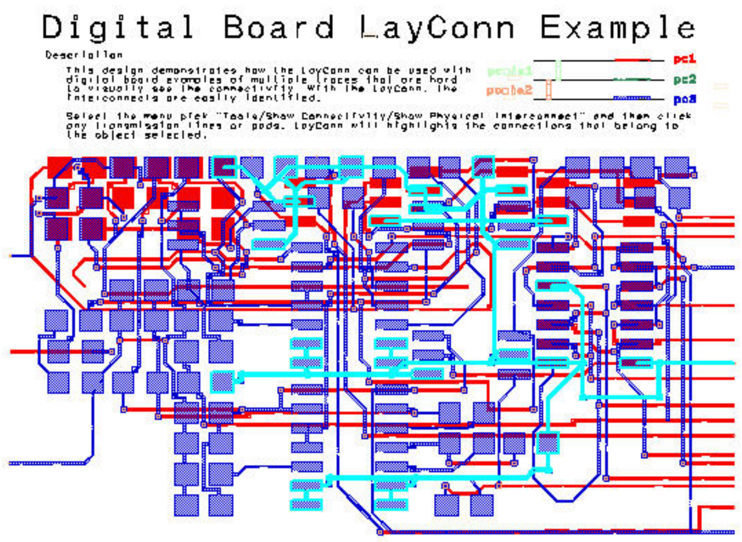
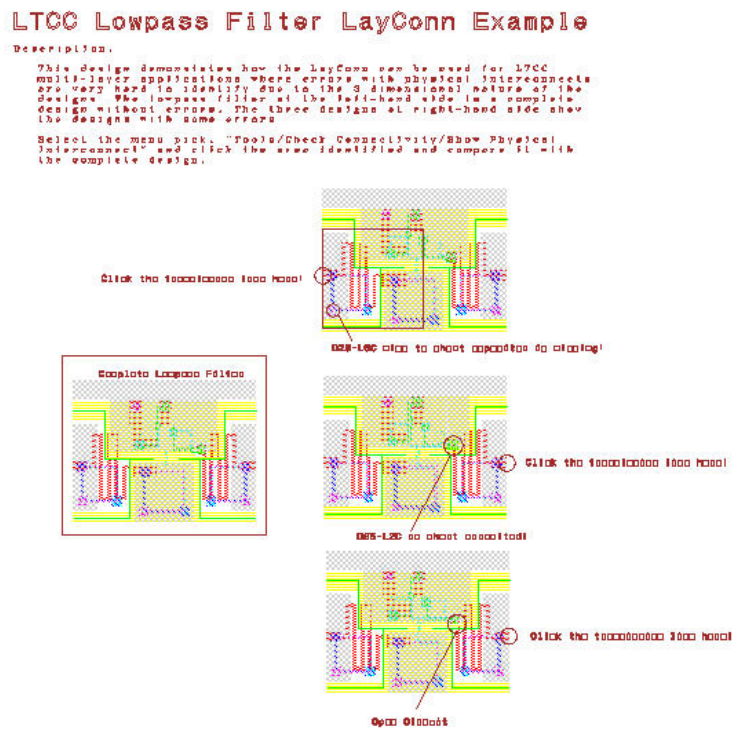

Digital Board LayConn Example
Description
This design demonstrates how the LayConn can be used with digital board examples of multiple traces that are hard to visually see the connectivity. With the LayConn, the interconnects are easily identified.
Select the menu pick "Tools/Show Connectivity/Show Physical Interconnect" and then click any transmission lines or pads. LayConn will highlights the connections that belong to the object selected.
pc1
pc2
pc3
pcola1
pcola2

Figure 3: LayConn - LTCC Examples (Layconn_LTCC_LPF)



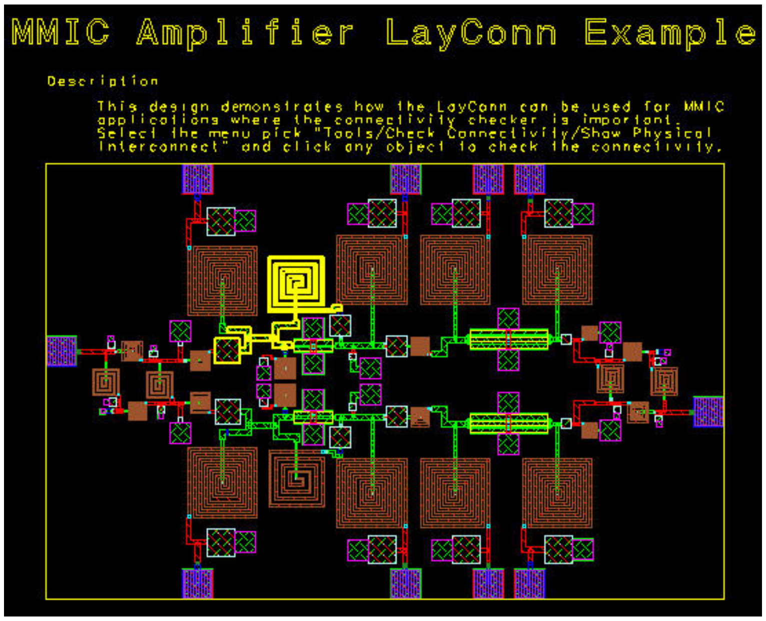
LTCC Lowpass Filter LayConn Example

Description.

This design demonstrates how the LayConn can be used for LTCC multi-layer applications where errors with physical interconnects are very hard to identify due to the 3 dimensional nature of the designs. The lowpass filter at the left-hand side is a complete design without errors. The three designs at right-hand side show the designs with some errors

Select the menu pick. "Tools/Check Connectivity/Show Physical Interconnect" and click the area identified and compare it with the complete design.

Figure 4: LayConn - MMIC Example (Layconn_MMIC_AMP)



MMIC Amplifier LayConn Example

Description

This design demonstrates how the LayConn can be used for MMIC applications where the connectivity checker is important. Select the menu pick "Tools/Check Connectivity/Show Physical Interconnect" and click any object to check the connectivity.

Power Amplifier Behavior Model

Location: $HPEESOF_DIR/examples/Tutorial/GComp7Test_wrk

Objective

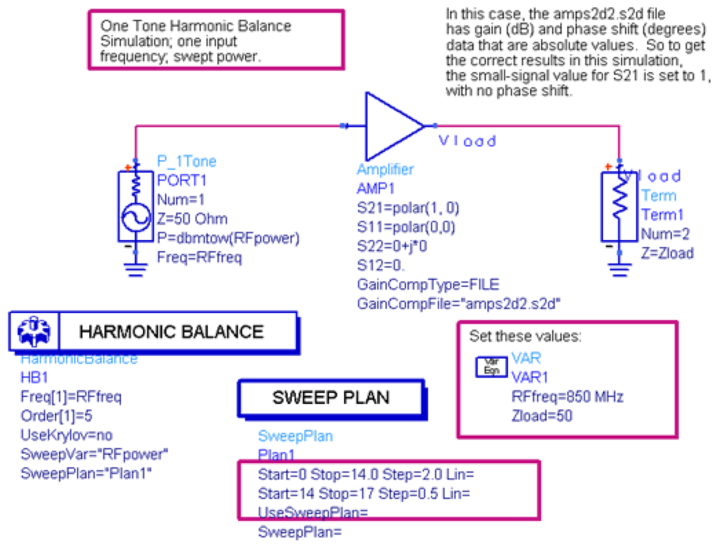
This example shows how to simulate an amplifier's gain and phase versus input power and use this data in a behavioral model of the amplifier, via the GCOMP7 model.

Setup

1. "HB1TonePswp" is a 1-tone, swept-power simulation of a transistor-level power amplifier. Note that a finer step size is used when the amplifier is driven into compression. Data display "HB1TonePswpAMtoPM.dds" shows the AM-to-PM and AM-to-AM simulation results and gives the phase and gain versus power level.
2. "GComp7test1" simulates a behavioral model amplifier which uses the gain compression and phase deviation data from the "HB1TonePswp" simulation, in a .s2d

file. In this case the deviation from the gain and phase relative to the gain and phase at the lowest input power level are used, so the behavioral model amplifier must have the small-signal S21 gain and phase defined.

3. "GComp7test2" is the same as GComp7test1, except that the .s2d file has the absolute values of the gain and phase shift through the amplifier versus input power level. So in this case, the small-signal S21 gain is set to 1 and the phase is set to 0. The simulation results are the same.


One Tone Harmonic Balance Simulation; one input frequency; swept power.

In this case, the amps2d2.s2d file has gain (dB) and phase shift (degrees) data that are absolute values. So to get the correct results in this simulation, the small-signal value for S21 is set to 1, with no phase shift.

P_1Tone
PORT1
Num=1
Z=50 Ohm
P=dbmtow(RFpower)
Freq=RFfreq

Amplifier
AMP1
S21=polar(1, 0)
S11=polar(0,0)
S22=0+j*0
S12=0.
GainCompType=FILE
GainCompFile="amps2d2.s2d"

Term
Term1
Num=2
Z=Zload

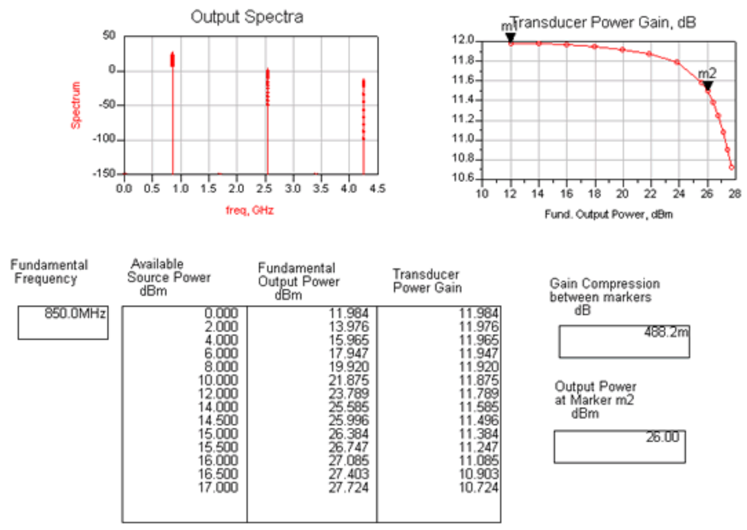
HARMONIC BALANCE
HarmonicBalance
HB1
Freq[1]=RFfreq
Order[1]=5
UseKrylov=no
SweepVar="RFpower"
SweepPlan="Plan1"

SWEEP PLAN
SweepPlan
Plan1
Start=0 Stop=14.0 Step=2.0 Lin=
Start=14 Stop=17 Step=0.5 Lin=
UseSweepPlan=
SweepPlan=

Set these values:
VAR
VAR1
RFfreq=850 MHz
Zload=50

Analysis

Figure 1: Simulation result from the behavior model



Fundamental Frequency
850.0MHz

Available Source Power dBm	Fundamental Output Power dBm	Transducer Power Gain
0.000	11.984	11.984
2.000	13.976	11.976
4.000	15.965	11.965
6.000	17.947	11.947
8.000	19.920	11.920
10.000	21.875	11.875
12.000	23.789	11.789
14.000	25.585	11.585
14.500	25.996	11.496
15.000	26.384	11.384
15.500	26.747	11.247
16.000	27.085	11.085
16.500	27.403	10.903
17.000	27.724	10.724

Gain Compression between markers dB
488.2m

Output Power at Marker m2 dBm
26.00

Notes

- Note that many simulation set-ups and data displays like those in "HB1TonePswp" are included in the *Amplifier DesignGuide* (dgpa).

Ptolemy DSP Sink Export to GoldenGate

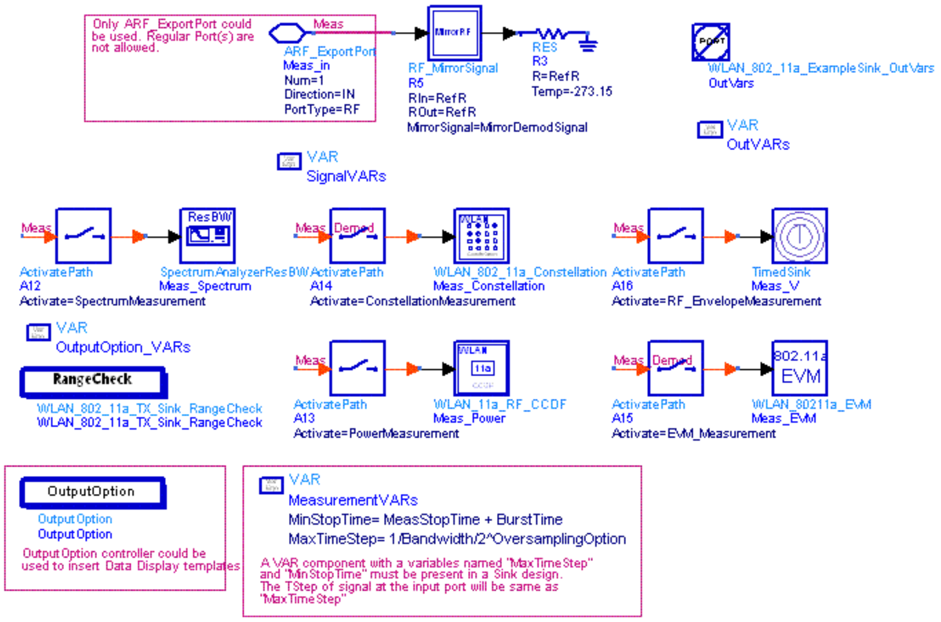
Location:
$HPEESOF_DIR/examples/Tutorial/WLAN_ExportToGoldenGate_wrk/WLAN_802_11a_TX_ExampleSink

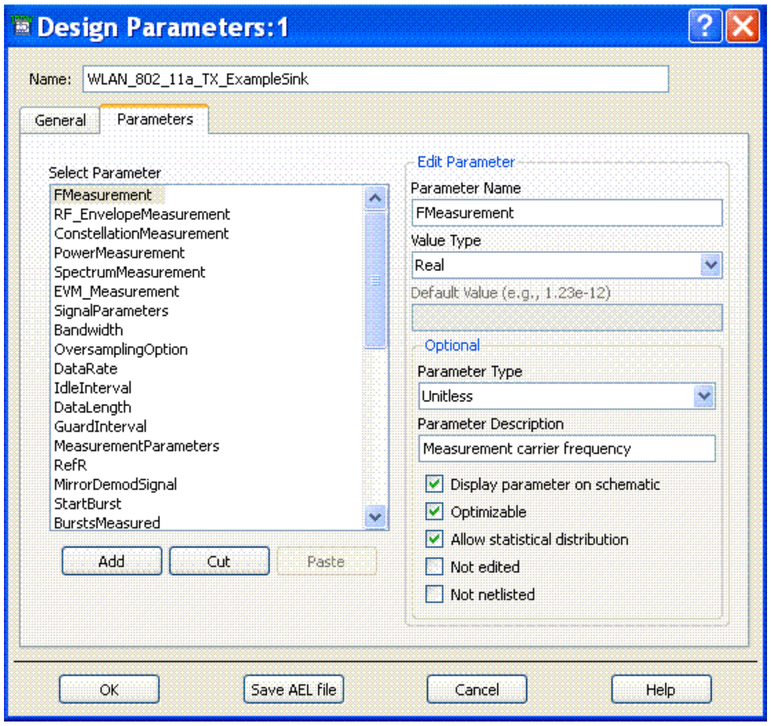
Objective

This example illustrates creation of a Ptolemy sink design to be exported to GoldenGate as Sink.

Setup

A WLAN 802.11a Sink is prepared for export to GoldenGate.


Only ARF_ExportPort could be used. Regular Port(s) are not allowed.
Meas
ARF_ExportPort
Meas_in
Num=1
Direction=IN
PortType=RF
MirrorRF
RF_MirrorSignal
R5
RIn=RefR
ROut=RefR
MirrorSignal=MirrorDemodSignal
RES
R3
R=RefR
Temp=-273.15
PORT
WLAN_802_11a_ExampleSink_OutVars
OutVars
VAR
OutVARs
VAR
SignalVARs
Meas
ActivatePath
A12
Activate=SpectrumMeasurement
ResBW
SpectrumAnalyzerResBW
Meas_Spectrum
Meas
Demod
ActivatePath
A14
Activate=ConstellationMeasurement
WLAN
WLAN_802_11a_Constellation
Meas_Constellation
Meas
ActivatePath
A16
Activate=RF_EnvelopeMeasurement
TimedSink
Meas_V
VAR
OutputOption_VARs
RangeCheck
WLAN_802_11a_TX_Sink_RangeCheck
WLAN_802_11a_TX_Sink_RangeCheck
Meas
ActivatePath
A13
Activate=PowerMeasurement
WLAN
11a
WLAN_11a_RF_CCDF
Meas_Power
Meas
Demod
ActivatePath
A15
Activate=EVM_Measurement
802.11a
EVM
WLAN_80211a_EVM
Meas_EVM
OutputOption
Output Option
Output Option
OutputOption controller could be used to insert Data Display templates
VAR
MeasurementVARs
MinStopTime= MeasStopTime + BurstTime
MaxTimeStep= 1/Bandwidth/2^OversamplingOption
A VAR component with a variables named "MaxTimeStep" and "MinStopTime" must be present in a Sink design.
The TStep of signal at the input port will be same as "MaxTimeStep"



Name: WLAN_802_11a_TX_ExampleSink

Select Parameter
FMeasurement
RF_EnvelopeMeasurement
ConstellationMeasurement
PowerMeasurement
SpectrumMeasurement
EVM_Measurement
SignalParameters
Bandwidth
OversamplingOption
DataRate
IdleInterval
DataLength
GuardInterval
MeasurementParameters
RefR
MirrorDemodSignal
StartBurst
BurstsMeasured

Edit Parameter
Parameter Name: FMeasurement
Value Type: Real
Default Value (e.g., 1.23e-12)

Optional
Parameter Type: Unitless
Parameter Description: Measurement carrier frequency

[✓] Display parameter on schematic
[✓] Optimizable
[✓] Allow statistical distribution
[ ] Not edited
[ ] Not netlisted

Notes

- This design should only be used on supported Linux and/or Unix platforms.
- ARF_ExportPort is used instead of normal Port.
- A VAR component with variables MaxTimeStep and MinStopTime is added.
- An OutputOption controller is added to insert Data Display Templates.
- Note the design parameter "FMeasurement" needed for an RF type ARF_ExportPort.
- Use "Tools -> Export ADS Ptolemy Design -> As GoldenGate Model" to export this sink to GoldenGate.

Ptolemy DSP Source Export to GoldenGate

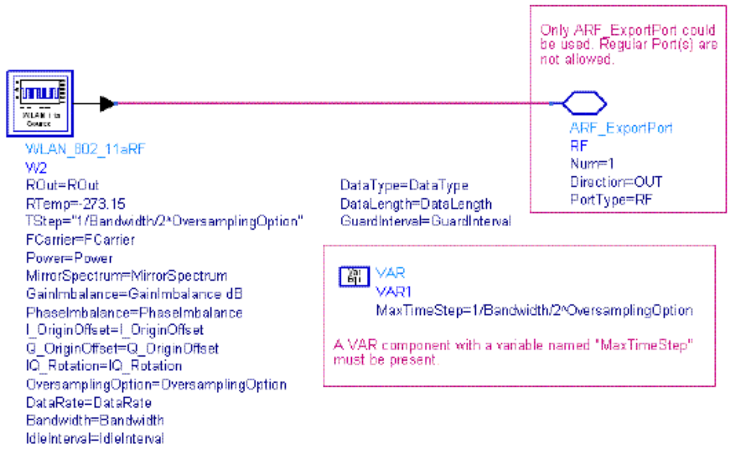
Location: $HPEESOF_DIR /examples /Tutorial /WLAN_ExportToGoldenGate_wrk /WLAN_802_11a_ExampleSource

Objective

This example illustrates creation of a Ptolemy source design to be exported to GoldenGate as Source.

Setup

A WLAN 802.11a Source is prepared for export to GoldenGate.



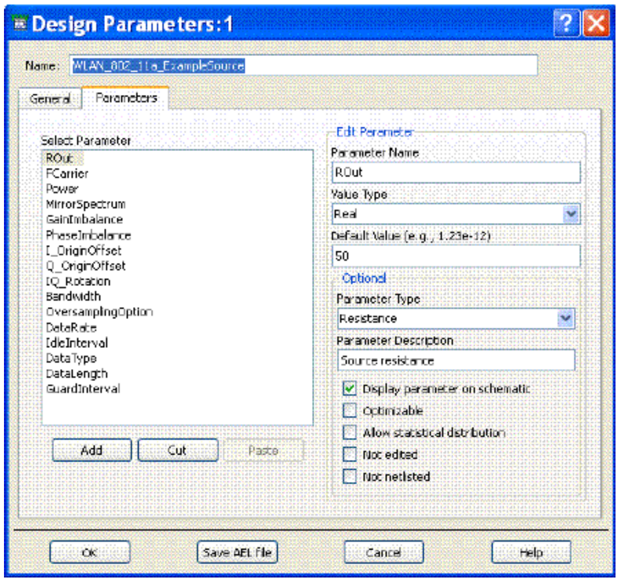
Only ARF_ExportPort could be used. Regular Port(s) are not allowed.

ARF_ExportPort
RF
Num=1
Direction=OUT
PortType=RF

WLAN_802_11aRF
W2
ROut=ROut
RTemp=-273.15
TStep="1/Bandwidth/2^OversamplingOption"
FCarrier=FCarrier
Power=Power
MirrorSpectrum=MirrorSpectrum
GainImbalance=GainImbalance dB
PhaseImbalance=PhaseImbalance
I_OriginOffset=I_OriginOffset
Q_OriginOffset=Q_OriginOffset
IQ_Rotation=IQ_Rotation
OversamplingOption=OversamplingOption
DataRate=DataRate
Bandwidth=Bandwidth
IdleInterval=IdleInterval
DataType=DataType
DataLength=DataLength
GuardInterval=GuardInterval

VAR
VAR1
MaxTimeStep=1/Bandwidth/2^OversamplingOption

A VAR component with a variable named "MaxTimeStep" must be present.



Name: WLAN_802_11a_ExampleSource

Select Parameter
ROut
FCarrier
Power
MirrorSpectrum
GainImbalance
PhaseImbalance
I_OriginOffset
Q_OriginOffset
IQ_Rotation
Bandwidth
OversamplingOption
DataRate
IdleInterval
DataType
DataLength
GuardInterval

Edit Parameter
Parameter Name: ROut
Value Type: Real
Default Value (e.g., 1.23e-12): 50
Optional
Parameter Type: Resistance
Parameter Description: Source resistance
[✓] Display parameter on schematic
[ ] Optimizable
[ ] Allow statistical distribution
[ ] Not edited
[ ] Not netlisted

Notes

- This design should only be used on supported Linux and/or Unix platforms.
- ARF_ExportPort is used instead of normal Port.
- A VAR component with variable MaxTimeStep is added.
- Note the design parameters "ROut" and "FCarrier" needed for an RF type ARF_ExportPort.
- Use "Tools -> Export ADS Ptolemy Design -> As GoldenGate Model" to export this source to GoldenGate.

Quick Tour of Communication System Design in Ptolemy

Location: $HPEESOF_DIR/examples/Tutorial/COMSYS_get_started_wrk

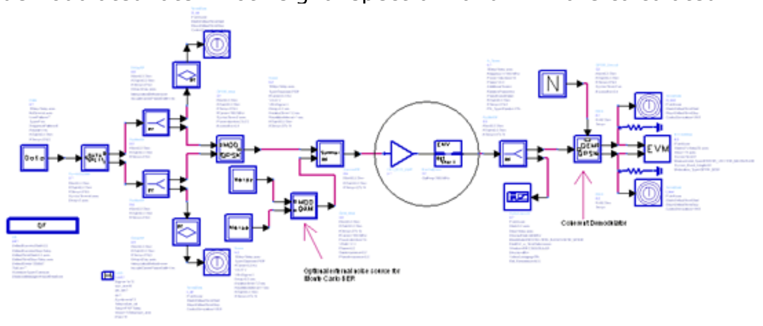
Objective

This workspace illustrates the basic Ptomely simulation of a simple communication system using quadrature phase shift keying (QPSK) modulation and demodulation building blocks.

Setup

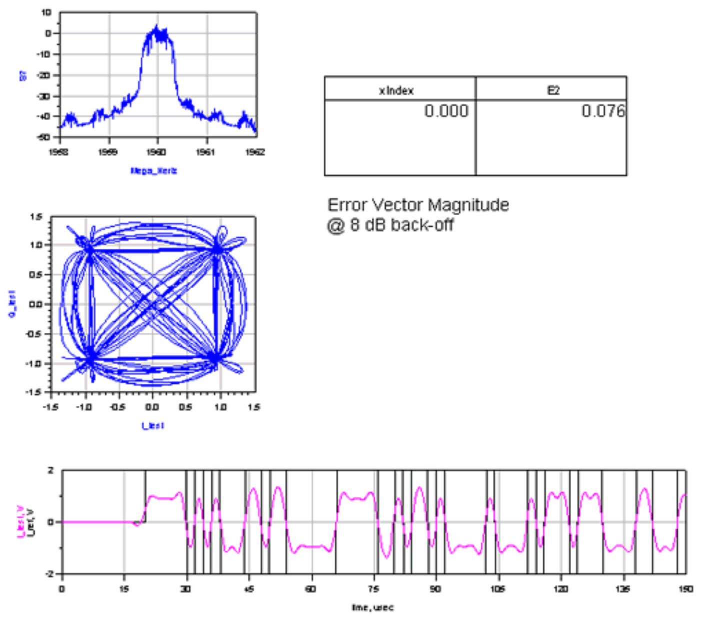
1. "QPSK_VIEW" shows a basic system using QPSK modulation and demodulation blocks. The carrier frequency is 1.96GHz. TkXYplot is used to display the constellation.

2. In "QPSK_EVM_test1", noise is added to the IQ channel of the QPSK signal, spectrum and Error Vector Magnitude (EVM) are simulated.
3. In "QPSK_EVM_test2", a cosimulation between Ptolemy and Circuit Envelope is illustrated, where the QPSK signal goes through a symbol RF amplifier block and is demodulated later. Both signal spectrum and EVM are calculated.



Analysis

Figure 1: QPSK output spectrum, trajectory diagram, error vector magnitude, input and output I-data



xIndex	E2
0.000	0.076

RF System Budget Measurements for 2-port Cascaded Networks

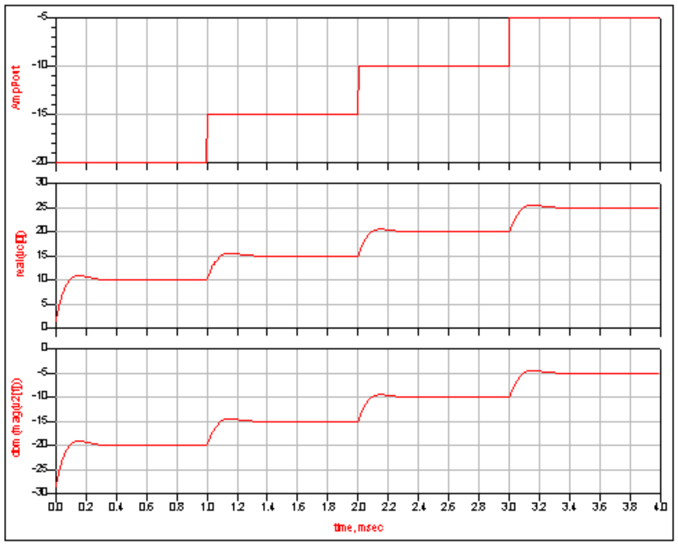
Location: $HPEESOF_DIR/examples/Tutorial/RF_Budget_Examples_wrk

Objective

This workspace contains designs demonstrating the RF Budget Controller, analyses and measurements. Results are displayed in same named Data Displays

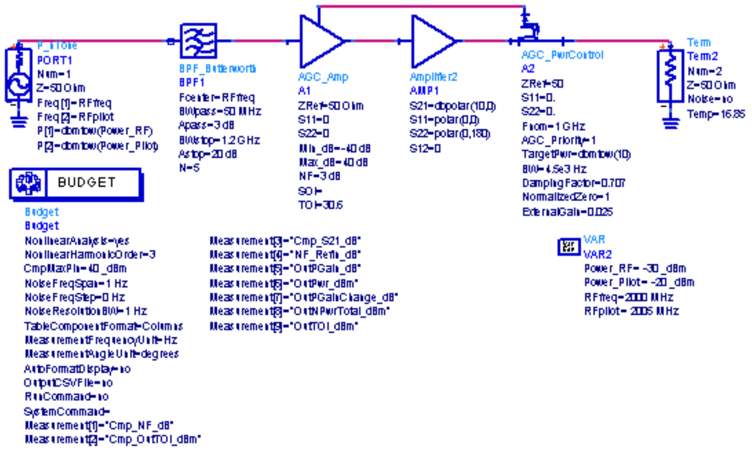
Setup

AGC_loop_CE_test - This design demonstrates use of the AGC loop components AGC_Amp and AGC_PwrControl.



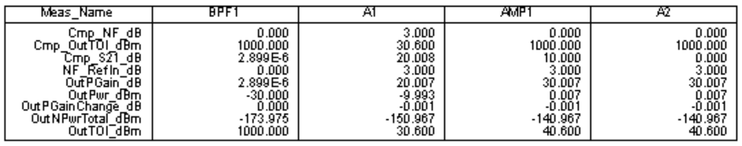
AGC_loop_CE_test data display of budget parameter vs. time

Budget_AGC - The AGC loop is controlled by the AGC_Amp and AGC_PwrControl.
Budget_AGC_Pilot - The AGC loop is controlled by the AGC_Amp and AGC_PwrControl.



```
P_nTone
PORT1
Num=1
Z=50 Ohm
Freq[1]=RFfreq
Freq[2]=RFpilot
P[1]=dbmtow(Power_RF)
P[2]=dbmtow(Power_Pilot)

BPF_Butterworth
BPF1
Fcenter=RFfreq
BWpass=50 MHz
Apass=3 dB
BWstop=1.2 GHz
Astop=20 dB
N=5

AGC_Amp
A1
ZRef=50 Ohm
S11=0
S22=0
Min_dB=-40 dB
Max_dB=40 dB
NF=3 dB
SOI=
TOI=30.6

Amplifier2
AMP1
S21=dbpolar(10,0)
S11=polar(0,0)
S22=polar(0,180)
S12=0

AGC_PwrControl
A2
ZRef=50
S11=0.
S22=0.
Fnom=1 GHz
AGC_Priority=1
TargetPwr=dbmtow(10)
BW=4.5e3 Hz
DampingFactor=0.707
NormalizedZero=1
ExternalGain=0.025

Term
Term2
Num=2
Z=50 Ohm
Noise=no
Temp=16.85

BUDGET
Budget
Budget
NonlinearAnalysis=yes
NonlinearHarmonicOrder=3
CmpMaxPin=40 _dBm
NoiseFreqSpan=1 Hz
NoiseFreqStep=0 Hz
NoiseResolutionBW=1 Hz
TableComponentFormat=Columns
MeasurementFrequencyUnit=Hz
MeasurementAngleUnit=degrees
AutoFormatDisplay=no
OutputCSVFile=no
RunCommand=no
SystemCommand=
Measurement[1]="Cmp_NF_dB"
Measurement[2]="Cmp_OutTOI_dBm"
Measurement[3]="Cmp_S21_dB"
Measurement[4]="NF_RefIn_dB"
Measurement[5]="OutPGain_dB"
Measurement[6]="OutPwr_dBm"
Measurement[7]="OutPGainChange_dB"
Measurement[8]="OutNPwrTotal_dBm"
Measurement[9]="OutTOI_dBm"

VAR
VAR2
Power_RF= -30 _dBm
Power_Pilot= -20 _dBm
RFfreq=2000 MHz
RFpilot= 2005 MHz
```

Budget_AGC_Pilot schematic



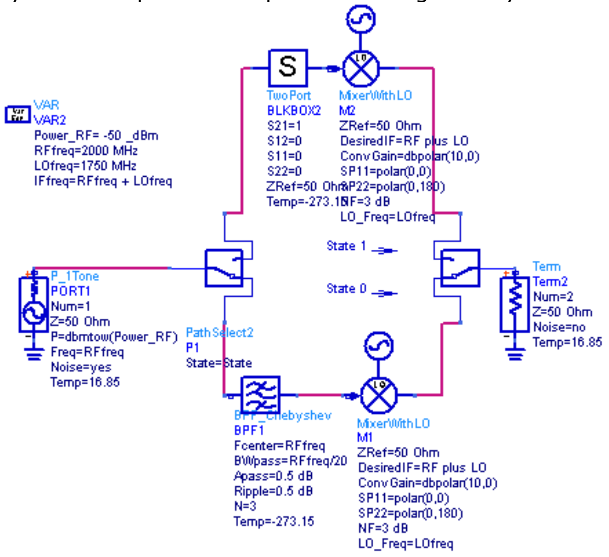
Meas_Name	BPF1	A1	AMP1	A2
Cmp_NF_dB	0.000	3.000	0.000	0.000
Cmp_OutTOI_dBm	1000.000	30.600	1000.000	1000.000
Cmp_S21_dB	2.899E-6	20.008	10.000	0.000
NF_RefIn_dB	0.000	3.000	3.000	3.000
OutPGain_dB	2.899E-6	20.007	30.007	30.007
OutPwr_dBm	-30.000	-9.993	0.007	0.007
OutPGainChange_dB	0.000	-0.001	-0.001	-0.001
OutNPwrTotal_dBm	-173.975	-150.967	-140.967	-140.967
OutTOI_dBm	1000.000	30.600	40.600	40.600

Budget_AGC_Pilot.dds data display showing budget parameter versus component

Budget_BaseLine - This example demonstrates a typical RF system design with filter, nonlinear amplifier, mixer, filter, and nonlinear amplifier.
Budget_Excel - This example demonstrates RF Budget output to Excel.
Budget_FSweep - This example demonstrates an RF system budget with swept frequency.
Budget_MixMax_Reflection - This example demonstrates a typical RF system design

with filter, nonlinear amplifier, mixer, filter, and nonlinear amplifier.

Budget_Mixer - This design show the performance of a mixer with and with out an input image rejection filter. It also demonstrates use of the PathSelect2 switch useful in Budget analysis to setup alternate paths for Budget analysis.



Budget Mixer with PathSelect2 switches

Budget_NF - This example demonstrates the various RF Budget noise measurements.These include:

1. component NF,
2. NF referenced from system input to component system output, including mixer image noise
3. Ideal NF referenced from system input to component output, excluding mixer image noise
4. Ideal NF referenced from component input to system output, excluding mixer image noise

Budget_NPwr - This example demonstrates the various RF Budget noise power measurements in system with 20 MHz bandwidth.These include:

1. component NF,
2. NF referenced from system input to component output, including mixer image noise
3. Noise density
4. Total noise power
5. Signal to noise ratio
6. Spurious free dynamic range for total noise

Budget_P1dB - This example demonstrates the various RF Budget 1 dB power compression measurements.These include:

1. component SOI, TOI and P1dB values,
2. P1 dB values evaluated from component input to system output,
3. P1 dB values evaluated from system input to component output,
4. Power change from small signal
5. Compressive dynamic range for

Budget_PSweep - This example demonstrates an RF system budget with swept power.
Budget_TOI_SOI - This example demonstrates the various RF Budget SOI and TOI measurements.These include:

1. component SOI and TOI values,
2. SOI/TOI values evaluated from component input to system output,
3. SOI/TOI values evaluated from system input to component output,
4. IM2 and IM3 levels at component outputs,
5. S/IM3 levels at component outputs
6. Spurious free dynamic range for noise in 1 MHz bandwidth.

Budget_Transformer - This example demonstrates use of a 50/75 ohm transformer after the 50ohm source, and a 75/50 ohm transformer before the system termination with use of 75 ohm system components.

Sensitivity Analysis, Basic Example

Location: $HPEESOF_DIR/examples/Tutorial/sensitivity_ex1_wrk

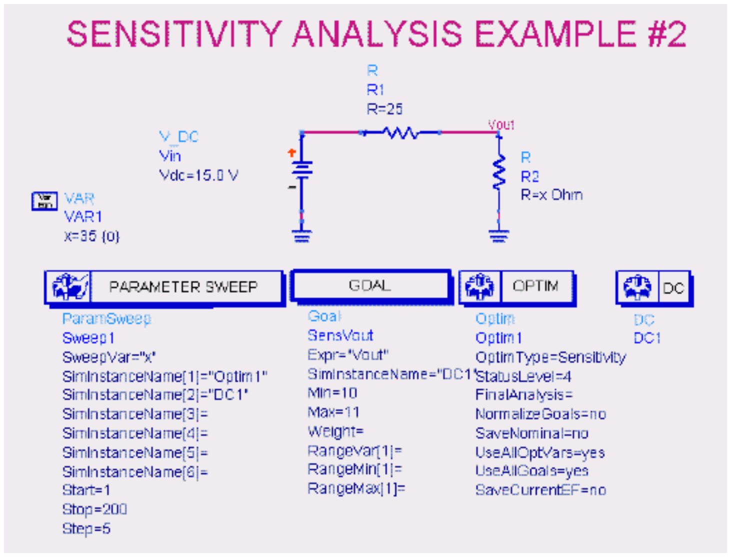
Objective

This workspace contains two examples that simulate the sensitivity (normalized and unnormalized) of a trivial circuit (a voltage divider.) The circuit is trivial so it is easy to calculate the sensitivity analytically.

Setup

The first,simulation, sens1 schematic and data display, shows a simple, single point sensitivity analysis compared against analytical results.The second simulation, sens2 schematic and data display, shows a swept sensitivity analysis.

Figure 1: Swept sensitivity simulation setup



SENSITIVITY ANALYSIS EXAMPLE #2

R
R1
R=25

V_DC
Vin
Vdc=15.0 V

Vout

R
R2
R=x Ohm

VAR
VAR1
x=35 {o}

PARAMETER SWEEP
ParamSweep
Sweep1
SweepVar="x"
SimInstanceName[1]="Optim1"
SimInstanceName[2]="DC1"
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1
Stop=200
Step=5

GOAL
Goal
SensVout
Expr="Vout"
SimInstanceName="DC1"
Min=10
Max=11
Weight=
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=

OPTIM
Optim
Optim1
OptimType=Sensitivity
StatusLevel=4
FinalAnalysis=
NormalizeGoals=no
SaveNominal=no
UseAllOptVars=yes
UseAllGoals=yes
SaveCurrentEF=no

DC
DC
DC1

Analysis

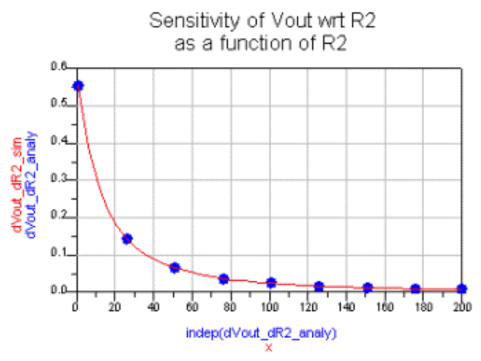
Expressions on the data display are used to calculate the analytical results for comparisonwith the calculated results.

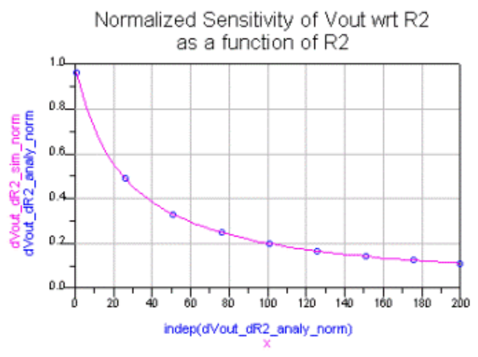
○ Analytic Sensitivity Calculation

Eqn R2=vs([1::25::200,200],[1::25::200,200])
Eqn R1=25
Eqn Vs=15
Eqn Vout_analy=R2/(R1+R2)*Vs
Eqn dVout_dR2_analy=R1/(R1+R2)**2*Vs
Eqn dVout_dR2_analy_norm=dVout_dR2_analy*R2/Vout_analy

— Results From Sensitivity Analysis

Eqn Vout_sim=Vout[0]
Eqn dVout_dR2_sim=SensVout[0]
Eqn dVout_dR2_sim_norm=norm_SensVout[0]





The swept sensitivity simulation results and results from analytic calculations agree

Notes

- Sensitivities are named in the dataset after the Goal instance name.
- Normalized sensitivities have "norm_" pre-pended to the goal name.
- Sensitivity analysis is discussed briefly in "Tuning, Optimization, and Statistical Design," under "Performing Nominal Optimization."

Simple Example Showing Application of DOE

Location: $HPEESOF_DIR/examples/Tutorial/doe2_wrk

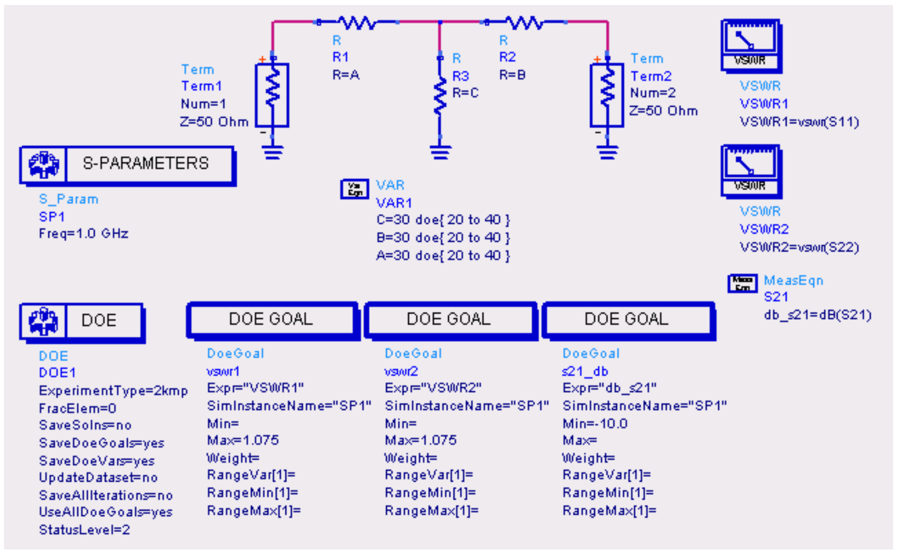
Objective

This example shows the use of Design of Experiments (DOE) to determine how changing resistor values affect the insertion loss and VSWR of a simple T attenuator.

Setup

The simulation setups and plots in this example are described in detail in *DOE Basic Example* (optstat).

Figure 1:DOE simulation setup



Term
Term1
Num=1
Z=50 Ohm

R
R1
R=A

R
R3
R=C

R
R2
R=B

Term
Term2
Num=2
Z=50 Ohm

VSWR
VSWR1
VSWR1=vswr(S11)

S-PARAMETERS
S_Param
SP1
Freq=1.0 GHz

VAR
VAR1
C=30 doe{ 20 to 40 }
B=30 doe{ 20 to 40 }
A=30 doe{ 20 to 40 }

VSWR
VSWR2
VSWR2=vswr(S22)

MeasEqn
S21
db_s21=dB(S21)

DOE
DOE1
ExperimentType=2kmp
FracElem=0
SaveSolns=no
SaveDoeGoals=yes
SaveDoeVars=yes
UpdateDataset=no
SaveAllIterations=no
UseAllDoeGoals=yes
StatusLevel=2

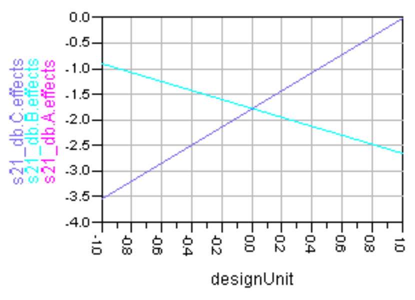
DOE GOAL
DoeGoal
vswr1
Expr="VSWR1"
SimInstanceName="SP1"
Min=
Max=1.075
Weight=
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=

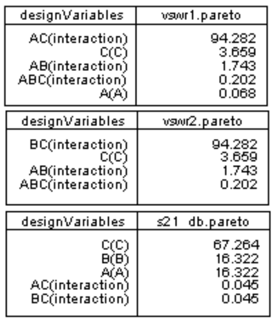
DOE GOAL
DoeGoal
vswr2
Expr="VSWR2"
SimInstanceName="SP1"
Min=
Max=1.075
Weight=
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=

DOE GOAL
DoeGoal
s21_db
Expr="db_s21"
SimInstanceName="SP1"
Min=-10.0
Max=
Weight=
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=

Analysis

Figure 2:DOE simulation results

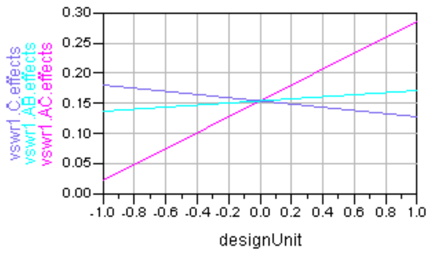


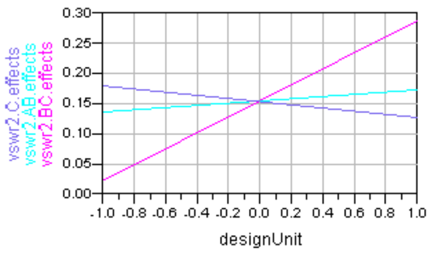


designVariables	vswr1.pareto
AC(interaction)	94.282
C(C)	3.659
AB(interaction)	1.743
ABC(interaction)	0.202
A(A)	0.068

designVariables	vswr2.pareto
BC(interaction)	94.282
C(C)	3.659
AB(interaction)	1.743
ABC(interaction)	0.202

designVariables	s21_db.pareto
C(C)	67.264
B(B)	16.322
A(A)	16.322
AC(interaction)	0.045
BC(interaction)	0.045





Simple Ptolemy Sequencer Example

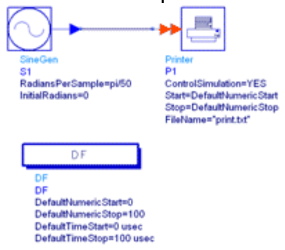
Location: $HPEESOF_DIR/examples/Tutorial/Sequencer_wrk

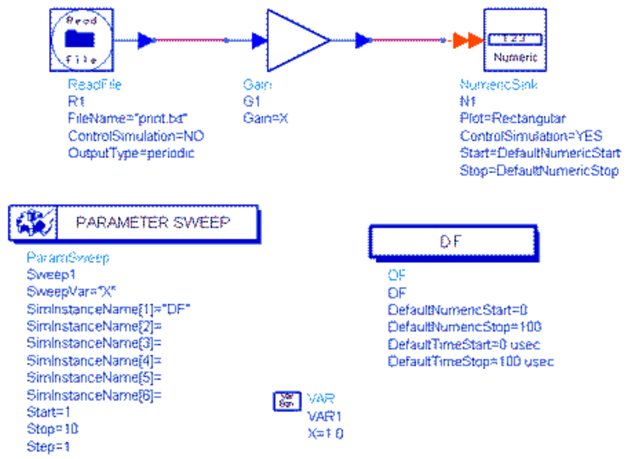
Objective

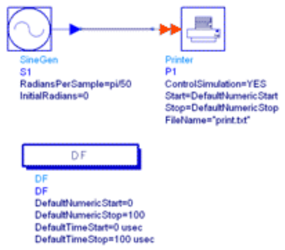
This example demonstrates the basic functionality of the Sequencer controller and how to reuse data generated from one test bench in a subsequence test bench.

Setup

1. "SineWaveSequence" Top-level design for the sequence. This design has two test benches that are sequenced - "SineWaveGeneration" and "SignalSweep". The Sequencer controller references both of these test benches.
2. "SineWaveGeneration" Generates one period of a sine wave and saves the signal to a text file using the Printer component. Note that the Printer component saves the data to the file when a test bench is completed and is thus available for use by the SignalSweep test bench. A NumericSink component could not been used in this example because the contents of a dataset are unavailable until the end of a simulation.
3. "SignalSweep" Reads the file generated from the SineWaveGeneration test bench using the ReadFile component and then multiplies the amplitude by the swept variable X. In this test bench, X is swept from 1-10.
4. "SineWaveSequence.dds" Plots the 10 result sine waves.

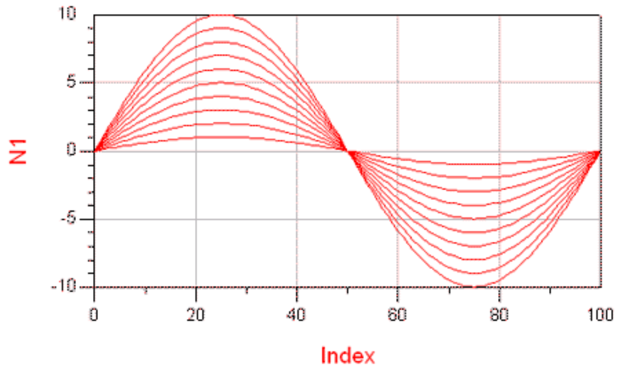






Analysis

Figure 1: Sine Wave Output



Notes

- For information on running multiple simulations in sequence, please refer to the *Sequencer Controller* (cktsim) documentation.
- For more complex examples of using the Sequencer Controller, please refer to the *3GPP BER* (examples) and *WLAN BER* (examples) examples.

S-Parameter Simulation Example

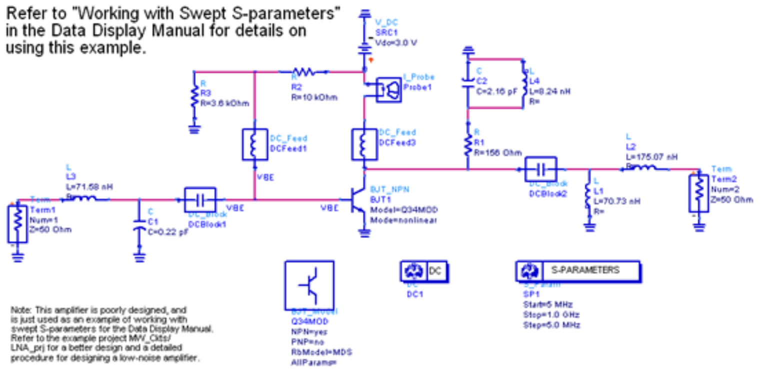
Location: $HPEESOF_DIR/examples/Tutorial/SweptSparams_wrk

Objective

This example shows a frequency swept S-Parameter analysis.

Setup

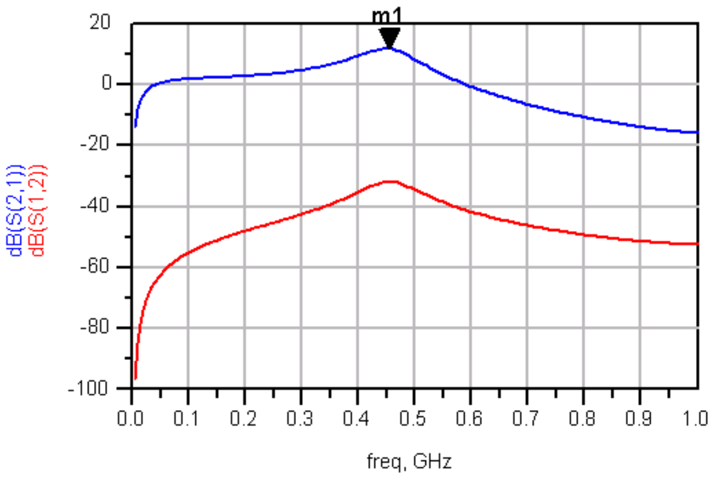
"amplifier" shows an S-parameter analysis of an amplifier with matching networks. Frequency is swept in the analysis.


Refer to "Working with Swept S-parameters" in the Data Display Manual for details on using this example.

Note: This amplifier is poorly designed, and is just used as an example of working with swept S-parameters for the Data Display Manual. Refer to the example project MW_Ckts/ LNA_prj for a better design and a detailed procedure for designing a low-noise amplifier.

Analysis

Figure 1: Gain and isolation versus frequency



Notes

- Simulation controller used: S-Parameters.

S-Parameters of 2-Port Terminated with Other Networks

Location: $HPEESOF_DIR/examples/Tutorial/refnet_wrk

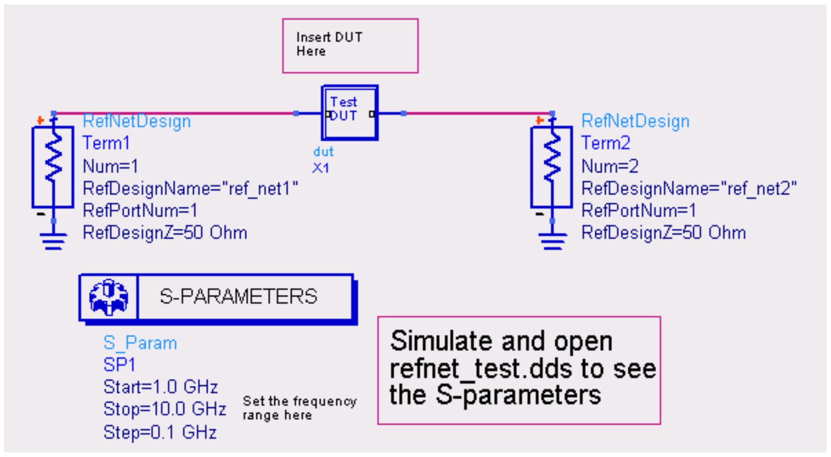
Objective

This example shows the use of RefNetDesign components which enable you to use arbitrary networks to terminate a DUT (Device Under Test) and measure its S-parameters when terminated with the arbitrary networks.

Setup

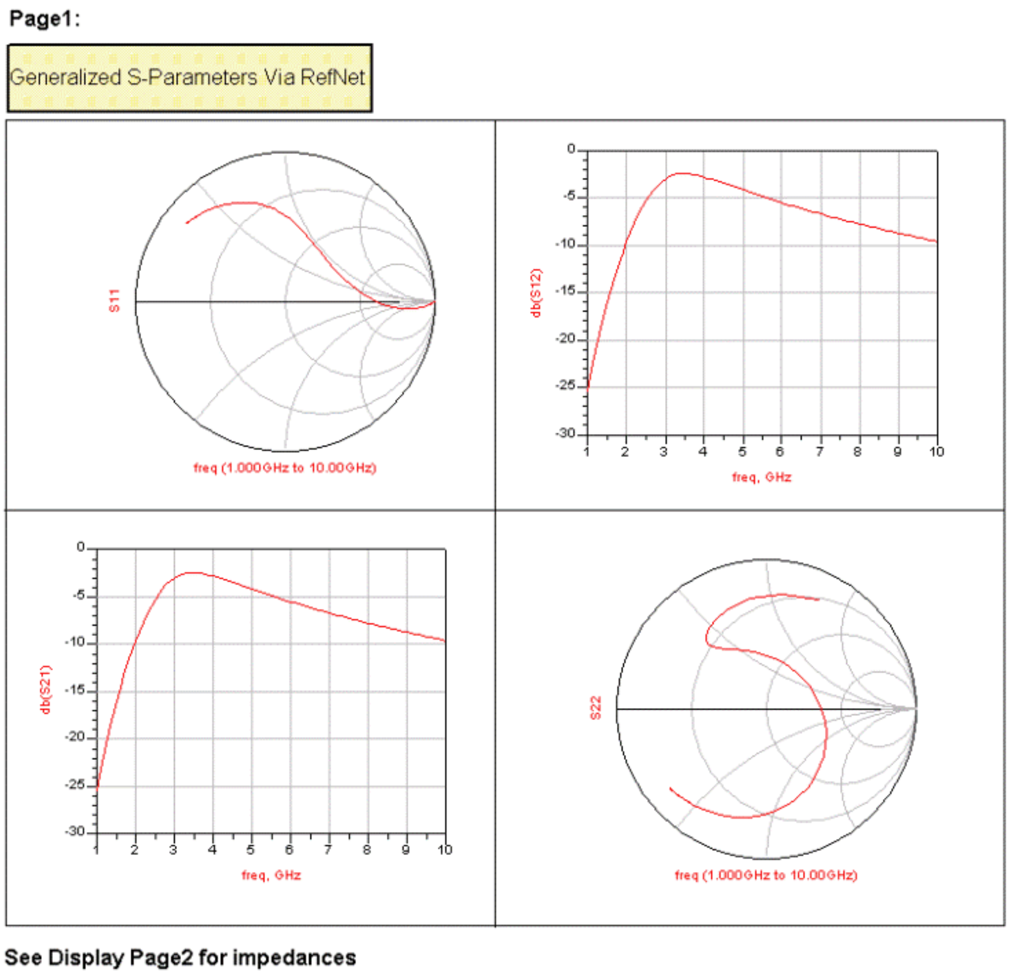
The primary design is "refnet_test" which uses subnetworks ref_net1, dut and ref_net2. The RefNetDesign components allow arbitrary subnetworks to terminate the "DUT" while measuring its S-parameters.

Figure 1: Schematic showing use of RefNetDesign components as input and output terminations



Analysis

Figure 2: S-parameters



Statistical Correlation in ADS

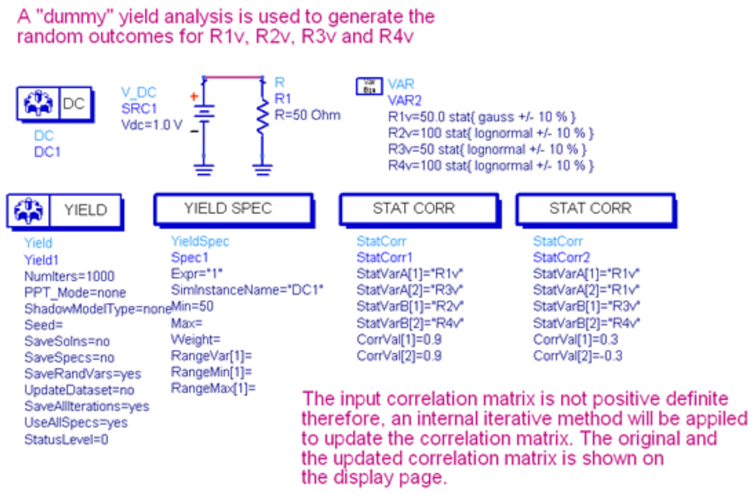
Location: $HPEESOF_DIR/examples/Tutorial/simpleCorr_wrk

Objective

This workspace illustrates the use of variables for statistical correlation and analysis in Advanced Design System.The Statistical variables used for statistical analysis (Monte Carlo, Yield, Yieldopt analysis) can be independently or statistically correlated. When they are statistically correlated, they are input via the "STAT CORR" component, which defines the input statistical correlation matrix (such as STAT CORR (R1 , R2) =0.9, STATCORR (R1,R3)=-9, etc).If the input correlation matrix is not Positive Definite, then there are problems producing samples. In such a situation, ADS provides a warning and then uses an internal iterative method to update the statistical correlation matrix until it is Positive Definite.

Setup

1. "simpleCorr" uses a "dummy" yield analysis to generate the random outcomes for R1v, R2v, R3v and R4v. An internal iterative method updates the correlation matrix because it is not positive definite. This schematic displays the original and the updated correlation matrix.



A "dummy" yield analysis is used to generate the random outcomes for R1v, R2v, R3v and R4v

DC
DC1

V_DC
SRC1
Vdc=1.0 V

R
R1
R=50 Ohm

VAR
VAR2
R1v=50.0 stat{ gauss +/- 10 % }
R2v=100 stat{ lognormal +/- 10 % }
R3v=50 stat{ lognormal +/- 10 % }
R4v=100 stat{ lognormal +/- 10 % }

YIELD
Yield
Yield1
NumIters=1000
PPT_Mode=none
ShadowModelType=none
Seed=
SaveSolns=no
SaveSpecs=no
SaveRandVars=yes
UpdateDataset=no
SaveAllIterations=yes
UseAllSpecs=yes
StatusLevel=0

YIELD SPEC
YieldSpec
Spec1
Expr="1"
SimInstanceName="DC1"
Min=50
Max=
Weight=
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=

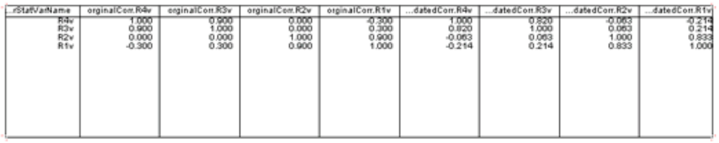
STAT CORR
StatCorr
StatCorr1
StatVarA[1]="R1v"
StatVarA[2]="R3v"
StatVarB[1]="R2v"
StatVarB[2]="R4v"
CorrVal[1]=0.9
CorrVal[2]=0.9

STAT CORR
StatCorr
StatCorr2
StatVarA[1]="R1v"
StatVarA[2]="R1v"
StatVarB[1]="R3v"
StatVarB[2]="R4v"
CorrVal[1]=0.3
CorrVal[2]=-0.3

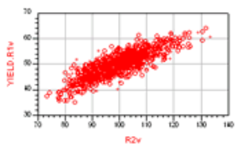
The input correlation matrix is not positive definite therefore, an internal iterative method will be appiled to update the correlation matrix. The original and the updated correlation matrix is shown on the display page.

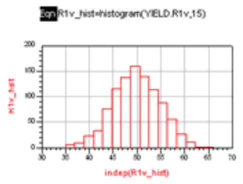
Analysis

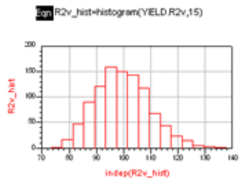
Figure 1: Correlation Results



...rStatVarName	orginalCorr.R4v	orginalCorr.R3v	orginalCorr.R2v	orginalCorr.R1v	...datedCorr.R4v	...datedCorr.R3v	...datedCorr.R2v	...datedCorr.R1v
R4v	1.000	0.900	0.000	-0.300	1.000	0.820	-0.063	-0.214
R3v	0.900	1.000	0.000	0.300	0.820	1.000	0.063	0.214
R2v	0.000	0.000	1.000	0.900	-0.063	0.063	1.000	0.833
R1v	-0.300	0.300	0.900	1.000	-0.214	0.214	0.833	1.000







Statistical Design Example for Oscillator Yield Analysis

Location: $HPEESOF_DIR/examples/Tutorial/StatisticalDesign_wrk

Objective

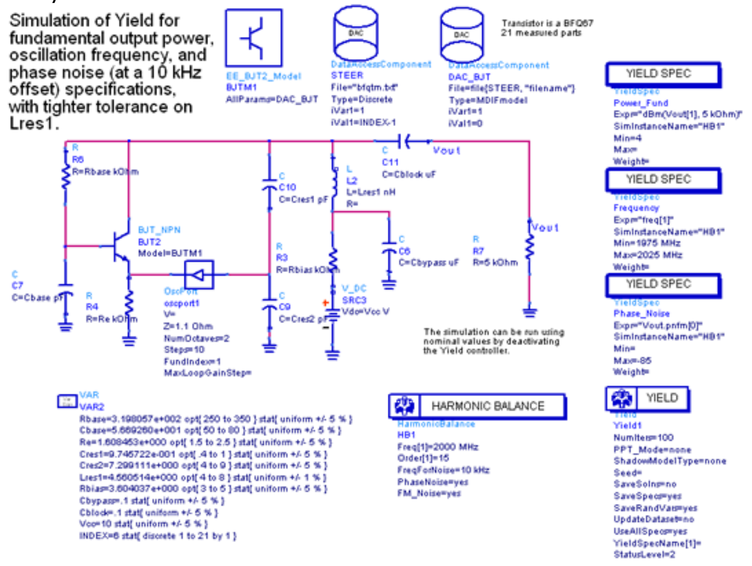
This workspace demonstrates ADS statistical design capability, which includes Optimization, Yield analysis, Yield Optimization (or Design Centering) and Yield versus component sensitivity histograms. The design objective is an oscillator with the following specs: Oscillation frequency = 2000 MHz +/- 25 MHz, Output Power > 4 dBm, Phase Noise at a 10 kHz offset frequency < -85 dBc.

Setup

1. "osc_1" and "osc_freq" designs include the initial calculated values of L's and C's for 2GHz oscillation, ignoring the transistor parasitics. The corresponding data displays clearly show the oscillation frequency is lower than designed, due to the transistor parasitics which were not accounted for.
2. "osc_freq_opt" design uses random optimization to modify the design to meet the specifications. "osc_tran" is the time domain transient analysis of the optimized circuit. The corresponding data displays show that the design is optimized and meets specs.
3. "yield_after_opt" design finds the yield with all the components varying +/- 5% and with measured data from 21 different transistors. The results show that the yield estimate is 36% after 100 trials. (Running more trials would produce a more accurate yield estimate. The number of trials was limited here to minimize dataset size.)
4. "osc_design_cntr" file is set up to optimize for yield. The design centering algorithm optimizes the design component values to maximize the yield according to their tolerances. The "osc_design_cntr" data display shows the improvement in the yield (from about 37% to 50%) and the optimal component values. In this case, the yield optimizer is able to improve the yield only slightly, and with unrealistically small changes in the nominal component values. What we discover below is that the poor

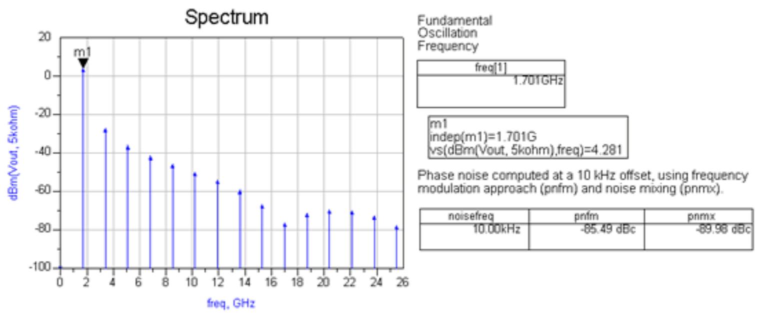
yield is due to a failure to meet the frequency specification and that tightening the tolerance on one component value (the inductor) will dramatically improve the yield.

5. "yield_after_design_cntr" is just a yield analysis using the nominal component values found from the "osc_design_cntr" schematic. The "yield_after_design_cntr" data display shows a yield of only 45% (more iterations would produce a more accurate estimate of the yield.)
6. An output Sensitivity Histogram data display, "Sens_hist_all_parts" was created to inspect and find out more about our design and what is causing it to be sensitive. With this file, the user can view the yield with respect to each component's variation. Also yield with respect to each individual spec can be seen, and the yield with respect to a change in specs can be updated instantantly. You have to change the corresponding equations on the data display page. You will notice that the inductor Lres1 is causing most of the yield problems.
7. "final_design" incorporates the recommended changes from the sensitivity histogram plots, and the yield is improved up to 75 % (shown in corresponding data display). You can also experiment with relaxing various specifications to see the impact on the final yield.



Analysis

Figure 1: Fundamental oscillation frequency, power and phase noise from initial design



Fundamental Oscillation Frequency

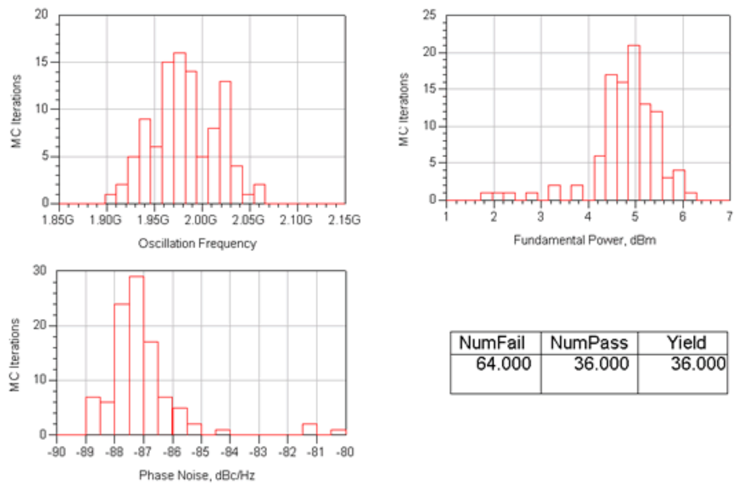
freq[1]
1.701GHz

m1
indep(m1)=1.701G
vs(dBm(Vout, 5kohm),freq)=4.281

Phase noise computed at a 10 kHz offset, using frequency modulation approach (pnfm) and noise mixing (pnmx).

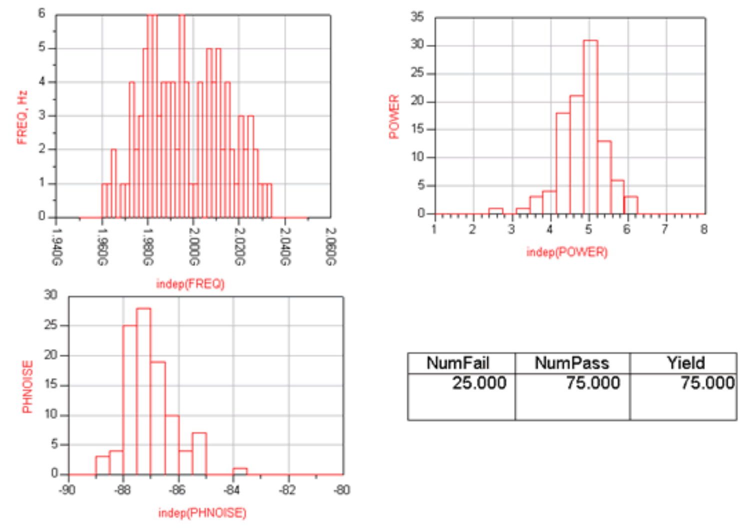
noisefreq	pnfm	pnmx
10.00kHz	-85.49 dBc	-89.98 dBc

Figure 2: Yield analysis after optimization



NumFail	NumPass	Yield
64.000	36.000	36.000

Figure 3: Yield for the final design after design centering and recommended changes from the sensitivity histogram plots



NumFail	NumPass	Yield
25.000	75.000	75.000

Swept Optimization, Basic Example

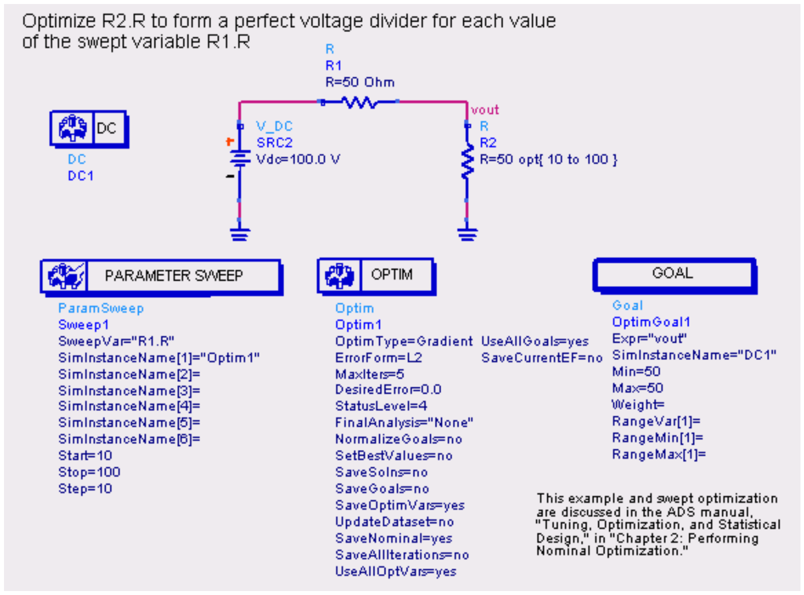
Location: $HPEESOF_DIR/examples/Tutorial/sweptOptTest_wrk

Objective

This example shows swept optimization applied to a trivial circuit, a voltage divider.

Setup

Figure 1: Swept optimization simulation setup



Optimize R2.R to form a perfect voltage divider for each value of the swept variable R1.R

```
ParamSweep
Sweep1
SweepVar="R1.R"
SimInstanceName[1]="Optim1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=10
Stop=100
Step=10
```

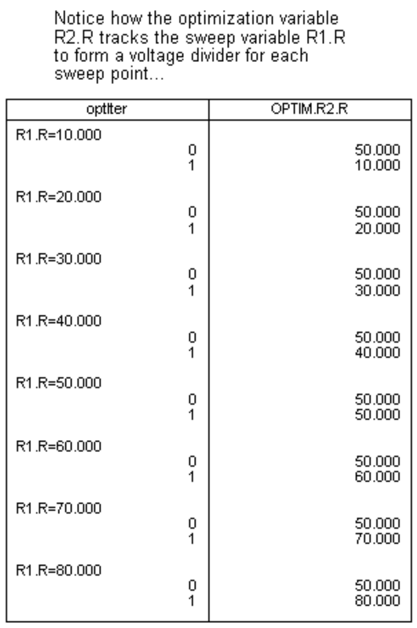
```
Optim
Optim1
OptimType=Gradient  UseAllGoals=yes
ErrorForm=L2        SaveCurrentEF=no
MaxIters=5
DesiredError=0.0
StatusLevel=4
FinalAnalysis="None"
NormalizeGoals=no
SetBestValues=no
SaveSolns=no
SaveGoals=no
SaveOptimVars=yes
UpdateDataset=no
SaveNominal=yes
SaveAllIterations=no
UseAllOptVars=yes
```

```
Goal
OptimGoal1
Expr="vout"
SimInstanceName="DC1"
Min=50
Max=50
Weight=
RangeVar[1]=
RangeMin[1]=
RangeMax[1]=
```

This example and swept optimization are discussed in the ADS manual, "Tuning, Optimization, and Statistical Design," in "Chapter 2: Performing Nominal Optimization."

Analysis

Figure 2: Swept optimization simulation results



Notice how the optimization variable R2.R tracks the sweep variable R1.R to form a voltage divider for each sweep point...

optIter		OPTIM.R2.R
R1.R=10.000		
	0	50.000
	1	10.000
R1.R=20.000		
	0	50.000
	1	20.000
R1.R=30.000		
	0	50.000
	1	30.000
R1.R=40.000		
	0	50.000
	1	40.000
R1.R=50.000		
	0	50.000
	1	50.000
R1.R=60.000		
	0	50.000
	1	60.000
R1.R=70.000		
	0	50.000
	1	70.000
R1.R=80.000		
	0	50.000
	1	80.000

Notes

This example and swept optimization are discussed in *Advanced Optimization Methodology* (optstat) in *Nominal Optimization* (optstat).

Tutorial on Budget Analysis (Archive)

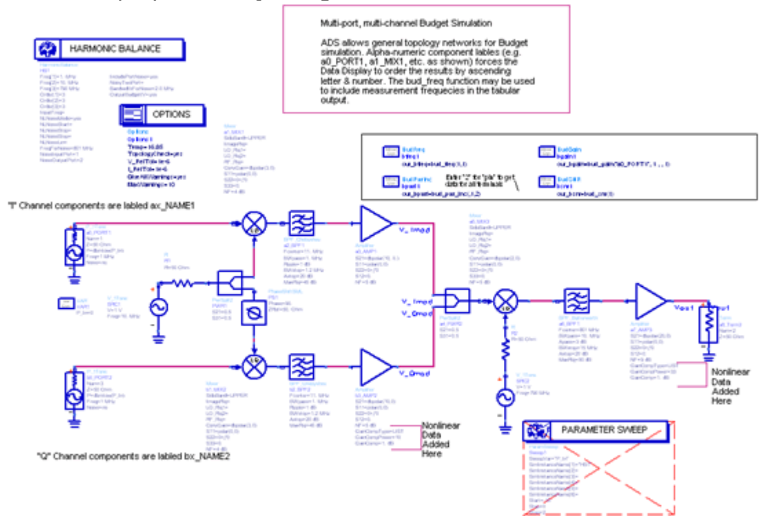
Location: $HPEESOF_DIR/examples/Tutorial/Archive_Learn_Budget_wrk

Objective

This example workspace contains two example networks for Budget simulations using expressions. For cascaded two-port networks, such as the one used in Linear_Budget, the RF Budget simulation controller (see Tutorial/RF_Budget_Examples_wrk) is recommended and preferred. The IQ_mod_bud example illustrate several important measurements as well as highlighting the capability to obtain Budget measurements on multi-channel networks incorporating multi-port components.

Setup

1. "Linear_Budget" is a simple cascade of two-port components, including a mixer for frequency conversion. AC analysis is used for simulation. Open "Linear_Budget.dds" to view a completed data display showing Gain, Incident Power, Noise Figure, and Noise Figure Degradation (due to each component).
2. "IQ_mod_bud" is a multi-port network is simulated using Harmonic Balance analysis. This example also includes frequency conversion, nonlinear data and multi-port components. An alphanumeric labeling method is used to make the data display order the components in a table. Measurements include Gain and Incident Power. The data is displayed in "IQ_Budget.dds".


Multi-port, multi-channel Budget Simulation
ADS allows general topology networks for Budget simulation. Alpha-numeric component lables (e.g. a0_PORT1, a1_MIX1, etc. as shown) forces the Data Display to order the results by ascending letter & number. The bud_freq function may be used to include measurement frequecies in the tabular output.
HARMONIC BALANCE
OPTIONS
"I" Channel components are labled ax_NAME1
"Q" Channel components are labled bx_NAME2
Nonlinear Data Added Here
Nonlinear Data Added Here
PARAMETER SWEEP
V_Imod
V_Qmod
Vout

Analysis

Figure 1: Gain from Port 1 to the Input of each component in the "a" path

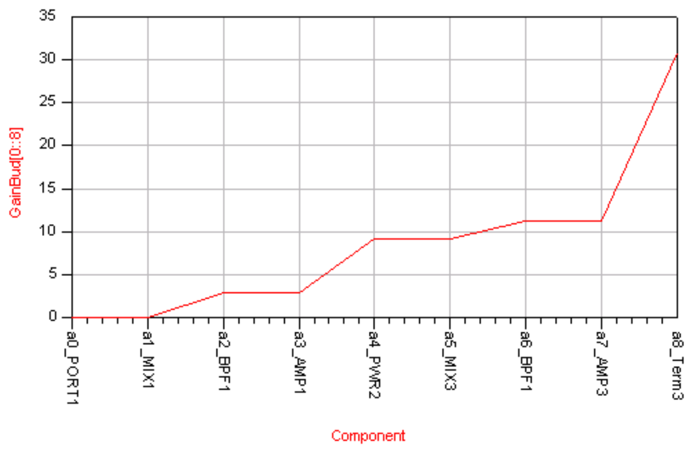
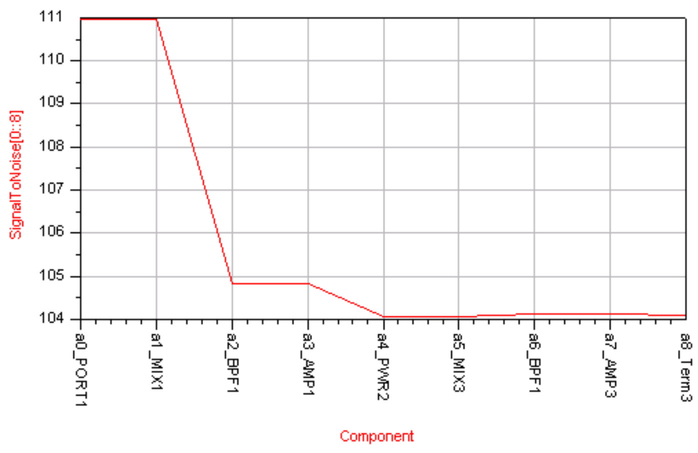


Figure 2: Signal-to-Noise Ratio from Port 1 to the Input of each component in the "a" path



Notes

- For cascaded two-port networks such as Linear_Budget, the RF Budget controller is recommended and preferred. Please see Tutorial/RF_Budget_Examples_wrk.
- The AC Analysis Controller is not included with design "Linear_Budget". Please see *Using Circuit Simulators for RF System Analysis* (cktsim).

User-Compiled Model Examples

Location: $HPEESOF_DIR/examples/Tutorial/UserCompiledModel_wrk

Objective

This workspace has several examples of User-Compiled Models.

Setup

MutInd model shows two coupled inductors. MutInd_test shows that the simulation results with user-compiled mutual inductor are consistent with simulation results with built-in MUC2 component.

PNDIODE model is a nonlinear, user-compiled model of a PN diode.
PNDIODE_test is a test design of this model in a circuit.

U2PA model is a model of three resistors in a "PI" connection.The equivalent circuit of the U2PA model is given in U2PA_test.

U2PB model is a coaxial line component. U2PB_test is a test of the model.

U2PC model is a lossy transmission line. U2PC_test is a test of the model.

RepeatParam_R is a simple resistor model. It demonstrates how to access the value of a repeated parameter.

VectorValue_R is a simple resistor model. It demonstrates how to access a parameter which has vector values.

Res is a simple resistor model. It demonstrates how to define parameters through another UCM model.

ucm_SnP is a UserCompiledModel version SnP model. It demontrates how to use a UCM model which supports variable number of external nodes.
ucm_SnP_test is a test of the 2-port ucm_SnP model and the 3-port ucm_SnP model.

UCM_Param_Test is the test of RepeatParam_R, VectorValue_R and VariableNodes_R.

Notes

Please note that this workspace only contains model source code. You must compile and link the models in order to run a simulation. To do this for the PNDIODE model, open the PNDIODE symbol view. Select **Tools** > **User-Compiled Model** > **Open User-Compiled Model**. In the User-Compiled Circuit Model dialog that appears, the No. of Internal Nodes should be 1 below the Model Code and, Transient Function should be checked below Code Options. Click Compile in the User-Compiled Circuit Model dialog. When you see the "Compile and link complete" message, click OK. Now you can run a simulation.

User-compiled models are discussed in "User-Defined Models," in "Building User-Compiled Analog Models."

Using DataAccessComponent (DAC) and S2PMDIF Component

Location: $HPEESOF_DIR/examples/Tutorial/DataAccess_wrk

Objective

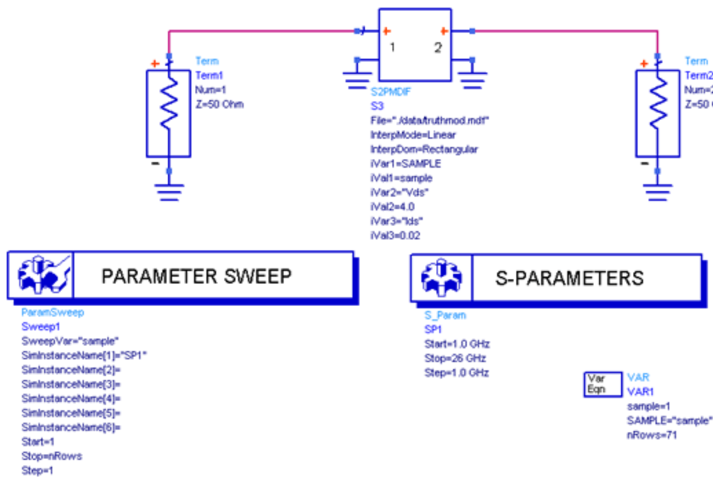
This example shows several ways of accessing data. Most of the designs use the DataAccessComponent (DAC). Others show using S2PMDIF component to access measured S-Parameter data.

Schematic list

1. "rfile" shows the design of a resistor with values stored in a .mdf data file. "rmall" shows a similar case where the resistance value is calculated from an R_Model, and several model parameters are stored in a .mdf file.
2. "amp1_test" shows an S-parameter simulation of an amplifier that has an EEBJT2 nonlinear device model. 21 different model files are simulated in a sweep. The DAC setup is included in "amp1".
3. "Truth_s2pmdif" shows the S-parameter simulation of an S2PMDIF component. The measured S-parameters are stored in file "truthmo.mdf".
4. "Truth_MonteCarlo" is similar to Truth_s2pmdif, except that a Monte Carlo simulation is run, in which the set of measured S-parameters used in the simulation is chosen randomly.
5. "truth_sweep" shows an alternative method of accessing S-parameter data.
6. "read_tim" shows how to read a TIM file.

Snapshots of some examples

Figure 1: "Truth_s2pmdif" circuit



Term
Term1
Num=1
Z=50 Ohm

S2PMDIF
S3
File="./data/truthmod.mdf"
InterpMode=Linear
InterpDom=Rectangular
iVar1=SAMPLE
iVal1=sample
iVar2="Vds"
iVal2=4.0
iVar3="Ids"
iVal3=0.02

Term
Term2
Num=2
Z=50 Ohm

ParamSweep
Sweep1
SweepVar="sample"
SimInstanceName[1]="SP1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1
Stop=nRows
Step=1

S_Param
SP1
Start=1.0 GHz
Stop=26 GHz
Step=1.0 GHz

VAR
VAR1
sample=1
SAMPLE="sample"
nRows=71

Figure 2: "Truth_s2pmdif" results — Measured S21 for 71 samples with vgs varying

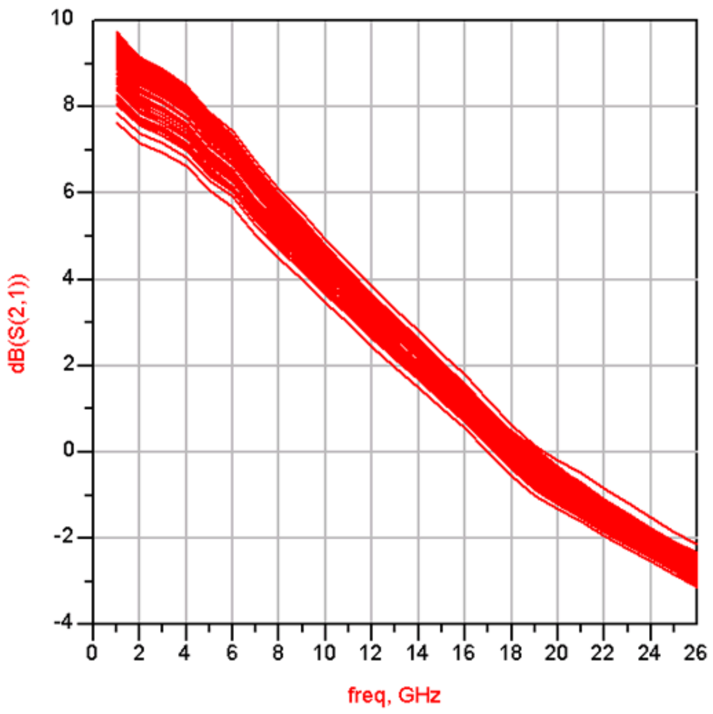


Figure 3: "amp1" in "amp1_test" shows swept models using two data access components

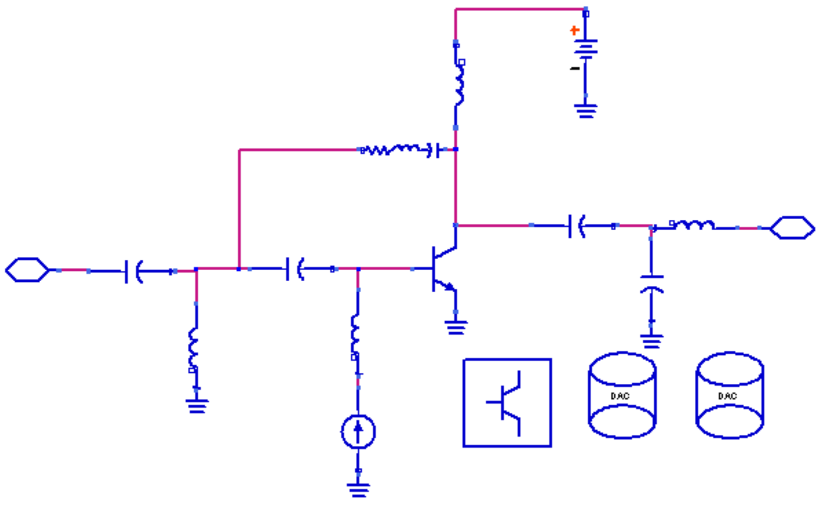
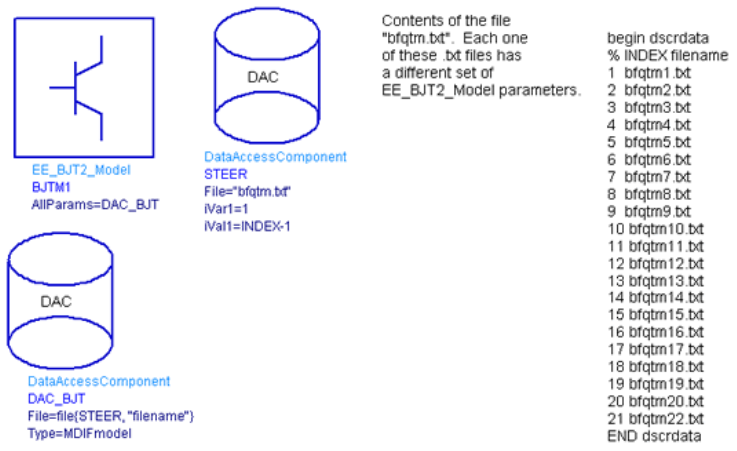


Figure 4: Details of the DACs in Figure 3



EE_BJT2_Model
BJTM1
AllParams=DAC_BJT

DataAccessComponent
STEER
File="bfqtm.txt"
iVar1=1
iVal1=INDEX-1

DataAccessComponent
DAC_BJT
File=file(STEER, "filename")
Type=MDIFmodel

Contents of the file "bfqtm.txt". Each one of these .txt files has a different set of EE_BJT2_Model parameters.

```
begin dscrdata
% INDEX filename
1  bfqtm1.txt
2  bfqtm2.txt
3  bfqtm3.txt
4  bfqtm4.txt
5  bfqtm5.txt
6  bfqtm6.txt
7  bfqtm7.txt
8  bfqtm8.txt
9  bfqtm9.txt
10 bfqtm10.txt
11 bfqtm11.txt
12 bfqtm12.txt
13 bfqtm13.txt
14 bfqtm14.txt
15 bfqtm15.txt
16 bfqtm16.txt
17 bfqtm17.txt
18 bfqtm18.txt
19 bfqtm19.txt
20 bfqtm20.txt
21 bfqtm22.txt
END dscrdata
```

Using Expressions in the Data Display Window

Location: $HPEESOF_DIR/examples/Tutorial/express_meas_wrk

Objective

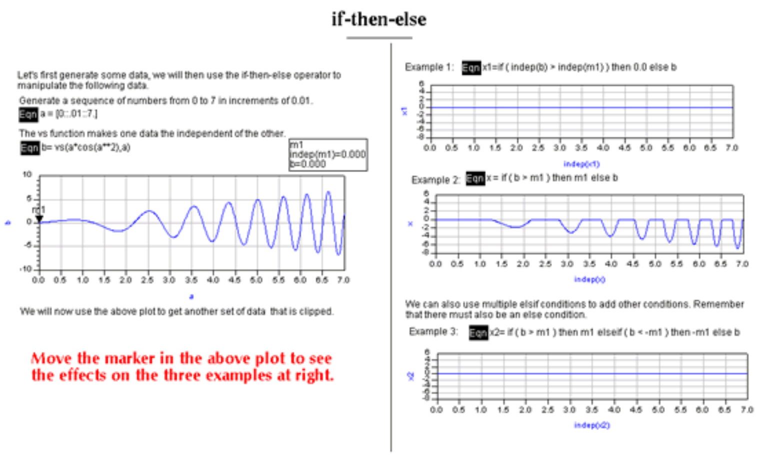
This example shows how to manipulate data using expressions in the data display window. Open the saved data display windows to see various functions and expressions applied.

Setup

1. "variable.dds" covers complete name hierarchy, short name and the double-dot operator, path substitution, names with special characters and the var() function.
2. "matrix.dds" covers matrix generation, indexing and matrix operators and functions.
3. "sweep.dds" covers the generation of a parameter sweep, indexing in the data and sweep functions.
4. "if_then_else.dds" shows examples of using if-then-else expression.
5. "analysis.dds" covers data manipulation from various simulations such as S-parameter, Transient, Harmonic, Envelope, it also has an eye() function tutorial.

Analysis

Figure 1: Use of if-then-else in the expression



if-then-else

Let's first generate some data, we will then use the if-then-else operator to manipulate the following data.

Generate a sequence of numbers from 0 to 7 in increments of 0.01.

Eqn a = [0::.01::7.]

The vs function makes one data the independent of the other.

Eqn b= vs(a*cos(a**2),a)

m1
indep(m1)=0.000
b=0.000

We will now use the above plot to get another set of data that is clipped.

Move the marker in the above plot to see the effects on the three examples at right.

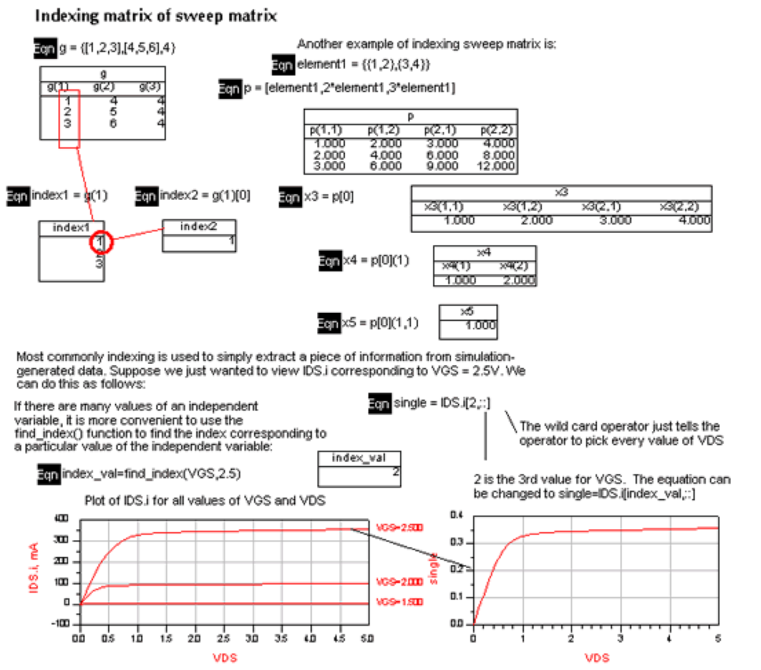
Example 1: Eqn x1=if (indep(b) > indep(m1)) then 0.0 else b

Example 2: Eqn x= if (b > m1) then m1 else b

We can also use multiple elsif conditions to add other conditions. Remember that there must also be an else condition.

Example 3: Eqn x2= if (b > m1) then m1 elseif (b < -m1) then -m1 else b

Figure 2: Indexing matrix in a sweep analysis



Indexing matrix of sweep matrix

Eqn g = {[1,2,3],[4,5,6],4}

g		
g(1)	g(2)	g(3)
1	4	4
2	5	4
3	6	4

Eqn index1 = g(1)

index1
1
2
3

Eqn index2 = g(1)[0]

index2
1

Another example of indexing sweep matrix is:

Eqn element1 = {(1,2),(3,4)}

Eqn p = [element1,2*element1,3*element1]

p			
p(1,1)	p(1,2)	p(2,1)	p(2,2)
1.000	2.000	3.000	4.000
2.000	4.000	6.000	8.000
3.000	6.000	9.000	12.000

Eqn x3 = p[0]

x3			
x3(1,1)	x3(1,2)	x3(2,1)	x3(2,2)
1.000	2.000	3.000	4.000

Eqn x4 = p[0](1)

x4	
x4(1)	x4(2)
1.000	2.000

Eqn x5 = p[0](1,1)

x5
1.000

Most commonly indexing is used to simply extract a piece of information from simulation-generated data. Suppose we just wanted to view IDS.i corresponding to VGS = 2.5V. We can do this as follows:

Eqn single = IDS.i[2,::]

The wild card operator just tells the operator to pick every value of VDS

If there are many values of an independent variable, it is more convenient to use the find_index() function to find the index corresponding to a particular value of the independent variable:

Eqn index_val=find_index(VGS,2.5)

index_val
2

2 is the 3rd value for VGS. The equation can be changed to single=IDS.i[index_val,::]

Plot of IDS.i for all values of VGS and VDS

Notes

- For further help on using expressions, use the on-line "Functions Help" by placing an equation on a data display page. When the Equation edit dialog box opens, click the "Functions Help" button for more information.

Using Frequency-Domain Defined Devices (FDD)

Location: $HPEESOF_DIR/examples/Tutorial/FDD_Examples_wrk

Objective

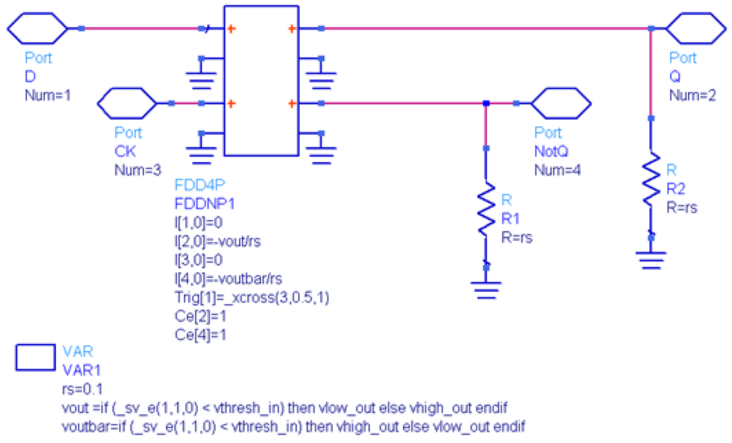
This file contains a number of Frequency-Domain Defined Device (FDD) examples. These devices allow users to quickly, easily and reliably add custom behavioral, nonlinear system models to the circuit and RF system simulator in the Advanced Design System, by describing the output spectral component(s) in terms of arbitrary functions of the input spectral components.

Setup

1. "IQ_modulator" implements an ideal I-Q modulator using FDD. "IQmodTest" simulates the I-Q modulator with baseband sine and cosine signals.
2. "FDDmixer" builds an ideal mixer using FDD. It takes into account of the upconversion, downconversion, LO leakage, RF leakage, and conversion gain

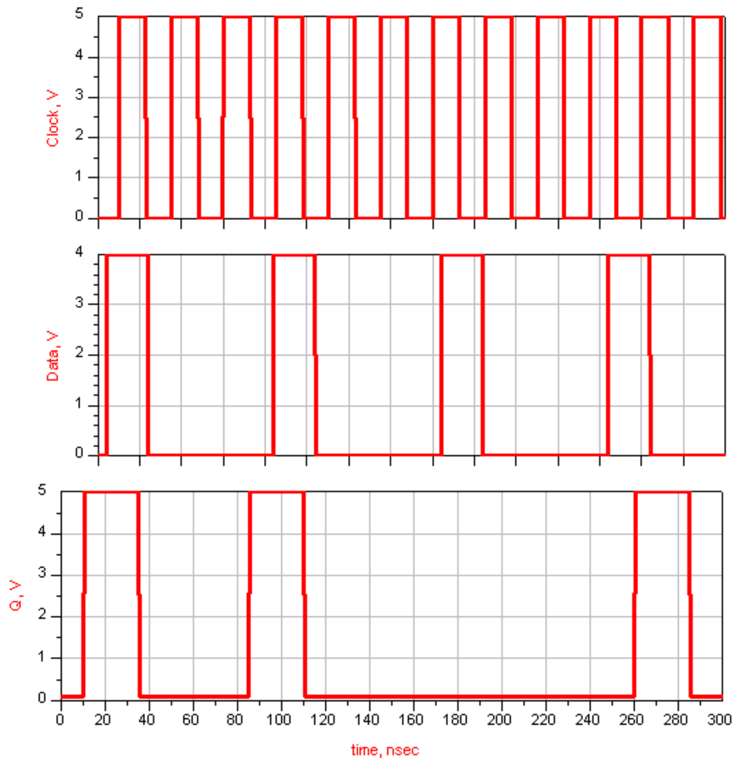
compression. "FDDmixerTest" is a harmonic balance simulation of the mixer. "FDDmixerTestEnv" simulates the mixer with Circuit Envelope, where the input signal is a carrier with upper and lower sidebands, and the mixer is configured to act as a frequency doubler.

3. "HarmSrc" generates a pulsed signal that is the sum of harmonics, using harmonic indexing (_harm).
4. "DflipFlop" builds a behavioral model DFF using FDD. It is simulated in "DflipFlopSim".
5. "SampleHold" simulates a sample and hold block implemented with an FDD.
6. "DynamicTherm_Amp" simulates an amplifier with gain reduced as a function of the heat that is dissipated within it. "testRC" simulates the RC filter that models the ability of the heat sink.



Analysis

Figure 1: Output clocked data of the DFF



Notes

- Simulation controllers used: HB, Envelope, Transient.

Using HB Noise Controller

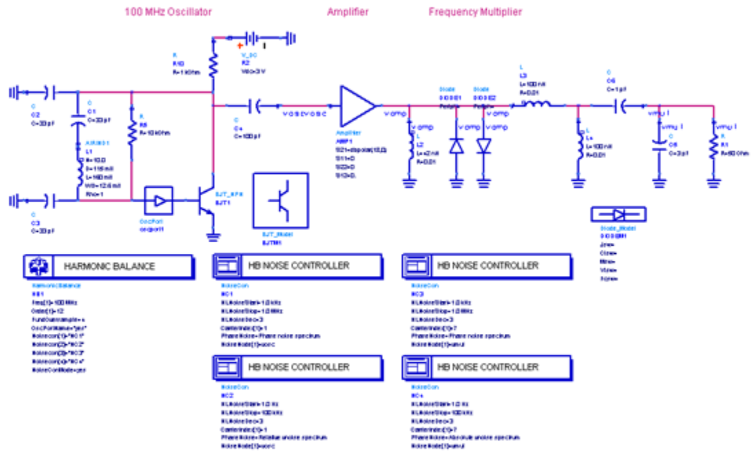
Location: $HPEESOF_DIR/examples/Tutorial/Noisecon_wrk

Objective

This example shows how to use HB Noise controller in Harmonic Balance (HB) simulation for nonlinear noise analysis including phase noise.

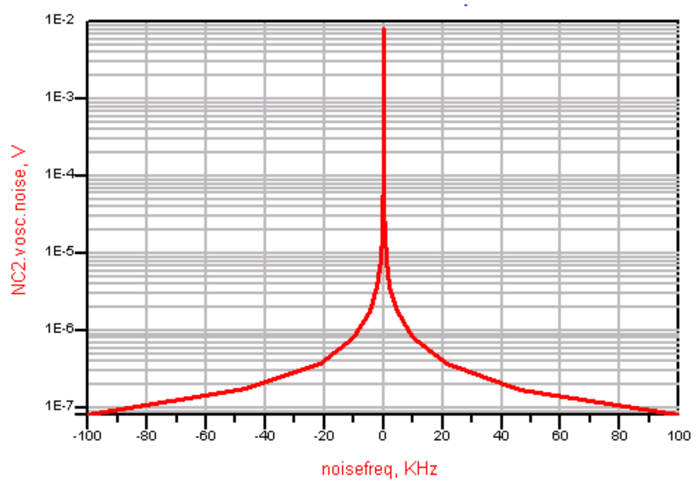
Setup

1. "mixer" uses HB Noise Controller to obtain noise figure for the lower and upper side bands for a heterodyne mixer and the noise voltage at specified nodes.
2. "mixer_pn" calculates both the spot noise and noise over a frequency band. The LO phase noise is also taken into account.
3. "oscmul" uses noise controller to show various noise spectrum calculation of an oscillator.



Analysis

Figure 1: Noise spectrum at node vosc



Notes

- Simulation controllers used: HB, HB Noise Controller.

Using N-State Modulator Component to Generate Modulated Signals

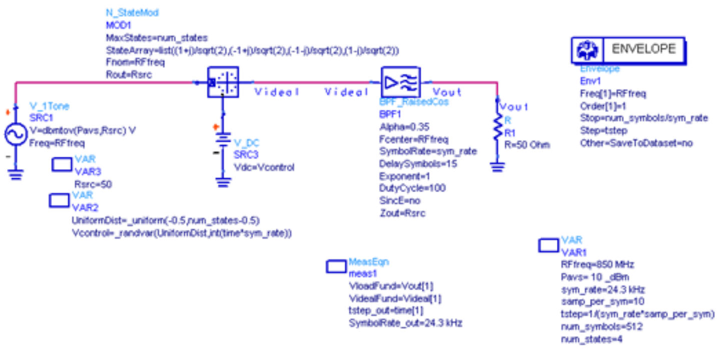
Location: $HPEESOF_DIR/examples/Tutorial/NstateModulators_wrk

Objective

This example shows how to generate modulated signals with arbitrary constellation diagrams, by using the N-state modulator component.

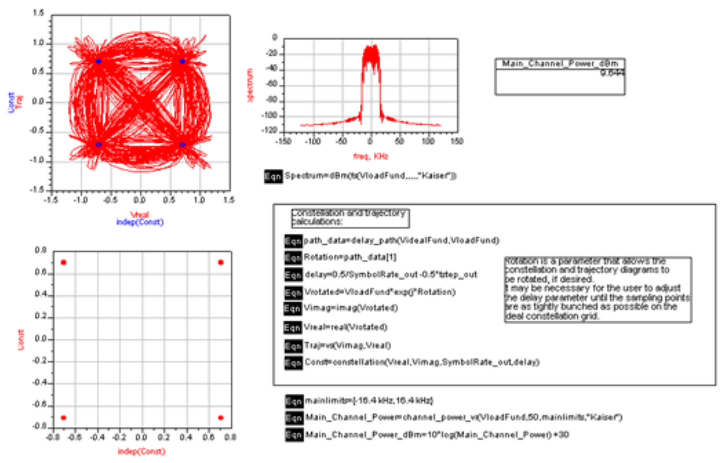
Setup

1. "QPSKsrc" shows how to generate a QPSK signal. The data display shows the trajectory and constellation diagrams, as well as the spectrum and the main channel power calculation.
2. "PSK8src" shows how to generate a signal with 8-phase shift keying modulation.
3. "QAM16src" shows how to generate a signal with 16-quadrature amplitude modulation. The QAM16_scale_fact data display shows how to calculate a scale factor such that the QAM modulation does not increase or decrease the average power of the unmodulated signal.
4. "RandSrcTest" is a test showing how to generate a voltage that is a random function of time.



Analysis

Figure 1: Properties of generated QPSK source



```
Eqn Spectrum=dBm(fs(VloadFund,,,,,"Kaiser"))

Constellation and trajectory
calculations:
Eqn path_data=delay_path(VidealFund,VloadFund)
Eqn Rotation=path_data[1]
Eqn delay=0.5/SymbolRate_out-0.5*tstep_out
Eqn Vrotated=VloadFund*exp(j*Rotation)
Eqn Vimag=imag(Vrotated)
Eqn Vreal=real(Vrotated)
Eqn Traj=vs(Vimag,Vreal)
Eqn Const=constellation(Vreal,Vimag,SymbolRate_out,delay)

Rotation is a parameter that allows the
constellation and trajectory diagrams to
be rotated, if desired.
It may be necessary for the user to adjust
the delay parameter until the sampling points
are as tightly bunched as possible on the
ideal constellation grid.

Eqn mainlimits=[-16.4 kHz,16.4 kHz]
Eqn Main_Channel_Power=channel_power_vi(VloadFund,50,mainlimits,"Kaiser")
Eqn Main_Channel_Power_dBm=10*log(Main_Channel_Power)+30
```

Notes

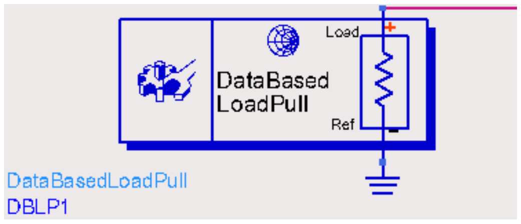
- See the *Modulated Sources* (examples) example (examples/Tutorial/ModSources_wrk) for more examples on how to generate modulated signals.

Using Measured Load Pull Data to Design and Optimize Impedance-Matching Networks

Location: $HPEESOF_DIR/examples/Tutorial/Using_Meas_Load_Pull_Data_wrk

The DataBasedLoadPull component makes it much easier to read in measured load pull data (from Maury systems) and use it directly to design output matching networks.

Figure: Data Based Load Pull Controller



Previously reading in such data was a non-obvious process from within the Load Pull DesignGuide. Now, you just insert a DataBasedLoadPull component from the new Simulation – Load Pull palette. You edit the component and select the Maury load pull file you want to use.
Ideally, it would be possible to see easily the contours of the data in the Maury load pull data file. You do this by using a Load_Tuner_Circular or Load_Tuner_Mag_Phase component to generate a load reflection coefficient within either a circular region or a pie-shaped region of the Smith Chart.

Figure: Load Tuner Circular

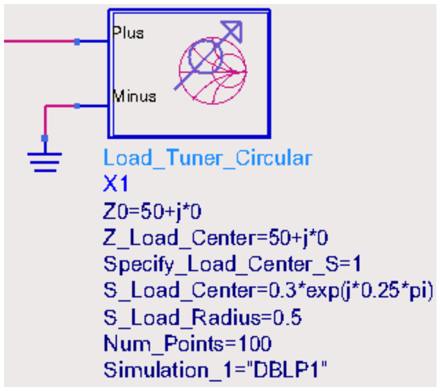
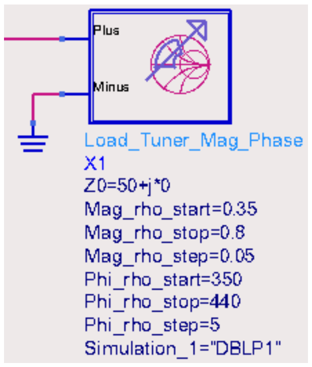
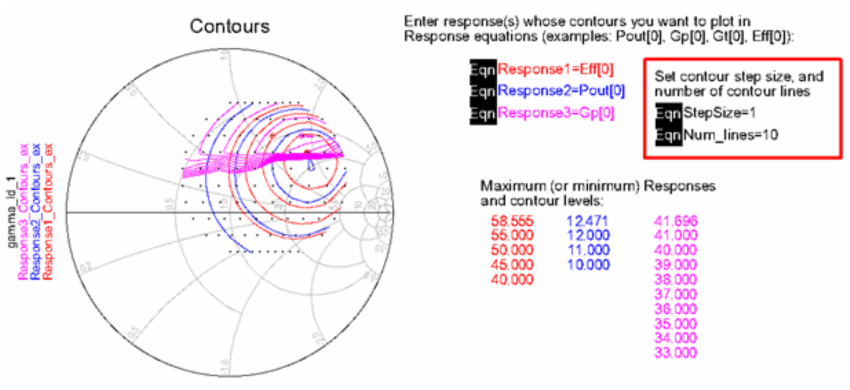


Figure: Load Tuner Mag Phase



These components are included in this example. The corresponding data display files in the example use a new function, contour_ex(), which makes it easier to compare load pull contours from different simulations and also sets the contour levels to "round number" values.

Figure: Contours



Enter response(s) whose contours you want to plot in
Response equations (examples: Pout[0], Gp[0], Gt[0], Eff[0]):

Eqn Response1=Eff[0]
Eqn Response2=Pout[0]
Eqn Response3=Gp[0]

Set contour step size, and
number of contour lines
Eqn StepSize=1
Eqn Num_lines=10

Maximum (or minimum) Responses
and contour levels:

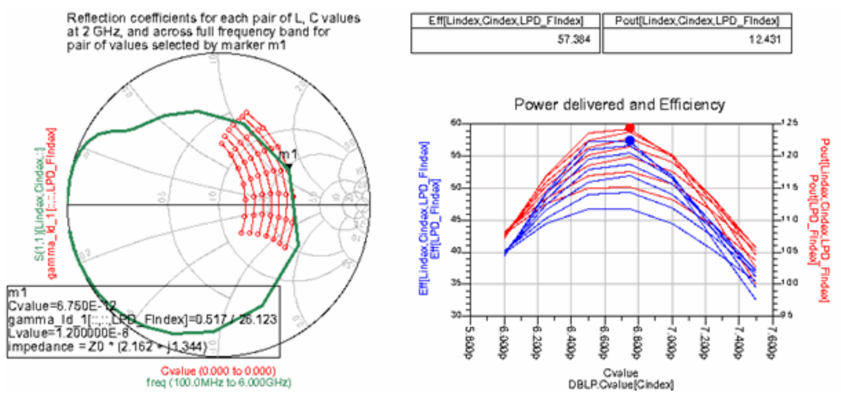
58.555	12.471	41.696
55.000	12.000	41.000
50.000	11.000	40.000
45.000	10.000	39.000
40.000		38.000
		37.000
		36.000
		35.000
		34.000
		33.000

This DataBasedLoadPull component has two main functions:

1. It runs an S-parameter simulation to see the reflection coefficient of whatever network is connected to it (usually this will be some output matching network you are designing), and
2. It uses this reflection coefficient to index into the data file to extract all the performance parameters that correspond to this particular reflection coefficient.

Under the Output Matching Network Design and Optimization folder, the Test_Meas_LP_Data_wOMN shows a sweep of L and C values in a simple output matching network. The corresponding data display shows how the reflection coefficient generated by the output matching network varies as well as how the power delivered and efficiency vary with the swept capacitance and inductance values.

Figure: Reflection Coefficient Generated by the Output Matching Network



m1
Cvalue=6.750E-12
gamma_ld_1[::,::,LPD_Findex]=0.517 / 26.123
Lvalue=1.200000E-8
impedance = Z0 * (2.162 + j1.344)

Eff[Lindex,Cindex,LPD_Findex]	Pout[Lindex,Cindex,LPD_Findex]
57.384	12.431

In the same folder, the Test_Meas_LP_Data_wOMN_Opt schematic shows an optimization of the output matching network parameter values to maximize efficiency, output power, and dB(S21) (in a 50-Ohm system) of the output matching network. You can use any performance parameters in the optimization goals.

Using OscPort2 in Oscillator Simulation

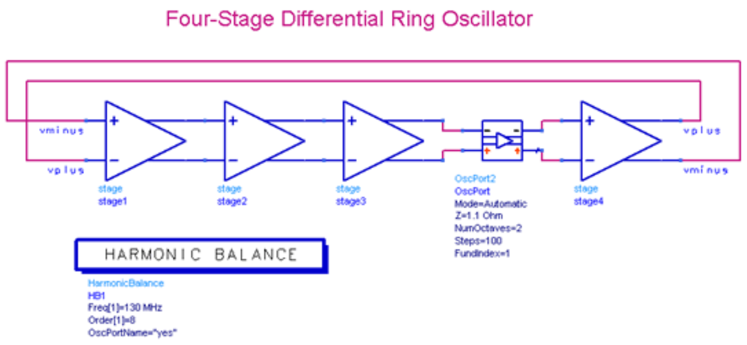
Location: $HPEESOF_DIR/examples/Tutorial/OscPort2_wrk

Objective

This example shows how to use Oscport2 in oscillator simulator to determine the large signal or small signal loop gain and steady state spectrum.

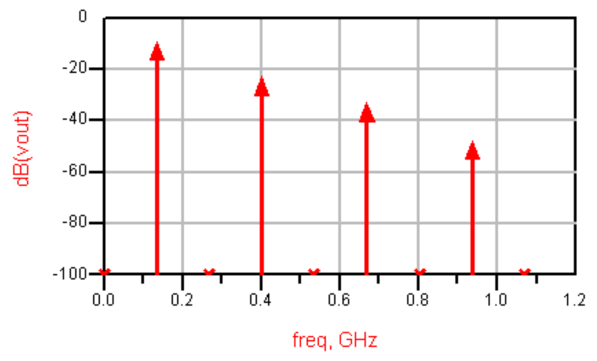
Setup

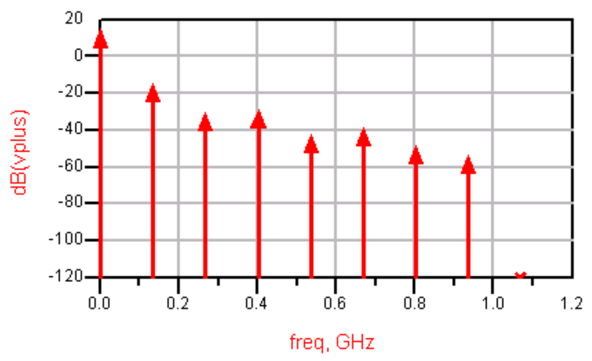
1. "OscPort2" shows a Harmonic Balance (HB) simulation of a four stage ring oscillator using OscPort2 element.
2. "loopgain_large" determines the large signal loop gain of the ring oscillator with swept injected signal voltage and frequency.
3. "loopgain_small" determines the small signal loop gain of an oscillator.



Analysis

Figure 1: Differential and single ended output spectrum





Notes

- Simulation controller used: HB.

Using SP_Probe in ADS

Location: $HPEESOF_DIR/examples/Tutorial/SP_Probe_how_to_wrk

Objective

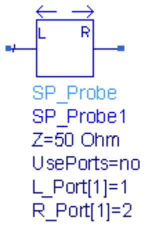
This example shows how to use the SP_Probe in s-parameter simulations. The user can follow the discussion by opening this example workspace. Notice that the figures correspond to the designs in the example workspace and are clearly marked.

Setup

The S-Parameter Probe (SP_Probe) available in ADS2008 is a dual mode component to be used in s-parameter simulations. In the first mode, it can measure one port network parameters (S, Z or Y) looking either to the right or left of the node to which it is connected. This mode is called the *gamma* mode, even though the Z11 and Y11 parameters are also available in addition to S11 in this mode.

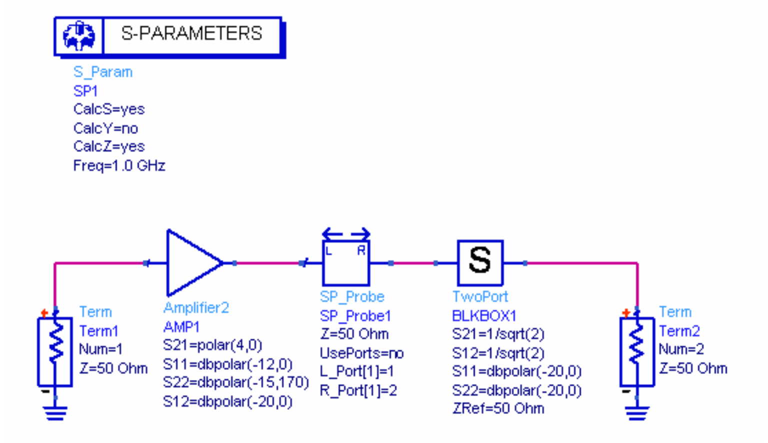
The second mode will calculate the full set of network parameters looking at the left and right of the node to which the SP_Probe is connected. This mode is called the *Use Ports* mode for reasons that will become apparent later.

The different modes are selected via the *UsePorts* parameter on the SP_Probe.


SP_Probe
SP_Probe1
Z=50 Ohm
UsePorts=no
L_Port[1]=1
R_Port[1]=2

where,
UsePorts=no gamma mode
UsePorts=yes use port mode

Gamma Mode:
The use of the SP_Probe in gamma mode is discussed in the following example. The following schematic represents the setup used:


S-PARAMETERS
S_Param
SP1
CalcS=yes
CalcY=no
CalcZ=yes
Freq=1.0 GHz

Term
Term1
Num=1
Z=50 Ohm

Amplifier2
AMP1
S21=polar(4,0)
S11=dbpolar(-12,0)
S22=dbpolar(-15,170)
S12=dbpolar(-20,0)

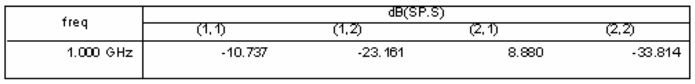
SP_Probe
SP_Probe1
Z=50 Ohm
UsePorts=no
L_Port[1]=1
R_Port[1]=2

TwoPort
BLKBOX1
S21=1/sqrt(2)
S12=1/sqrt(2)
S11=dbpolar(-20,0)
S22=dbpolar(-20,0)
ZRef=50 Ohm

Term
Term2
Num=2
Z=50 Ohm

sp_probe_gamma_test

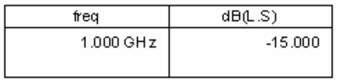
Note that the S and Z parameters have been requested from the s-parameter simulation. When you run this simulation the following s-parameters results will be displayed:



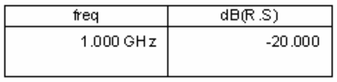
freq	dB(SP.S)			
	(1,1)	(1,2)	(2,1)	(2,2)
1.000 GHz	-10.737	-23.161	8.880	-33.814

Note that a SP prefix has been added to the normal s-parameter data produce by ADS. The reason for this is, there are now several sets of data and you must have a way to keeping these separated in the dataset. It is also worth mentioning that during other simulations (DC, Harmonic Balance, etc...) the SP_Probe is treated as a "short" and therefore does not have any impact on the results for those simulations.

The data for the SP_Probe is shown in the following table:

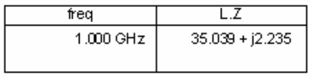


freq	dB(L.S)
1.000 GHz	-15.000

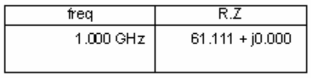


freq	dB(R.S)
1.000 GHz	-20.000

Notice that the prefix L and R (left and right respectively) have been added to the results. Since the SP_Probe is being used in gamma mode (UsePorts=no) there is only one network parameter in this case (e.g., L.S(1,1)). Next the impedance parameter data for the SP_Probe is shown:

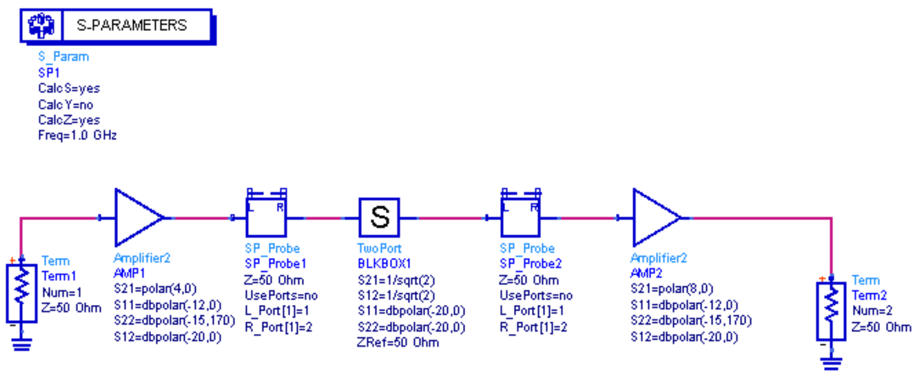


freq	L.Z
1.000 GHz	35.039 + j2.235



freq	R.Z
1.000 GHz	61.111 + j0.000

The following schematic illustrates the use of multiple gamma probes. First remember that when a SP_Probe is being analyzed the rest of the probes are shorted.



sp_probe_gamma_test2

freq	dB(SP_Probe1.L.S)
1.000 GHz	-15.000

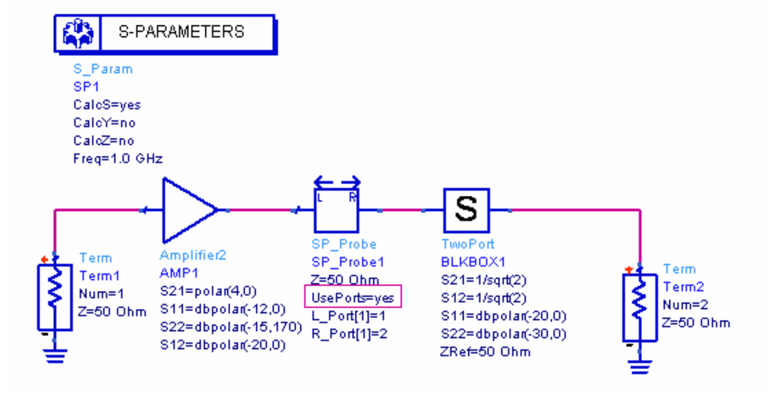
freq	dB(SP_Probe2.L.S)
1.000 GHz	-33.814

freq	dB(SP_Probe1.R.S)
1.000 GHz	-12.810

freq	dB(SP_Probe2.R.S)
1.000 GHz	-12.000

The most notable change is that the actual SP_Probe instance name (e.g., SP_Probe1) in addition to L or R has been added to the variable name containing the results. This is done to keep the data from the different SP_Probes separated in the dataset.

Use Port Mode:
The following schematic is used to discuss SP_Probe when **UsePorts=yes**.



sp_probe_port

In this mode the L_Port and R_Port parameters are used whereas in the Gamma mode they are not. You must realize that when the SP_Probe is in this mode and looking to the left side, the circuit is terminated at that node and anything to the right is open circuited. The reverse is true for the right side of the SP_Probe. Note that L_Port[1]=1, means that for the L.S s-parameters Term1 (Num=1) will be port one which by default will make the SP_Probe1.L become the second termination (Num=2). In this basic case, the left of the probe only sees the amplifier, so the s-parameters from the left side should be equal to those of the amplifier, as shown in the following:

freq	dB(L.S)			
	(1,1)	(1,2)	(2,1)	(2,2)
1.000 GHz	-12.000	-20.000	12.041	-15.000

The right side is therefore equal to the TwoPort component BLKBOX1:

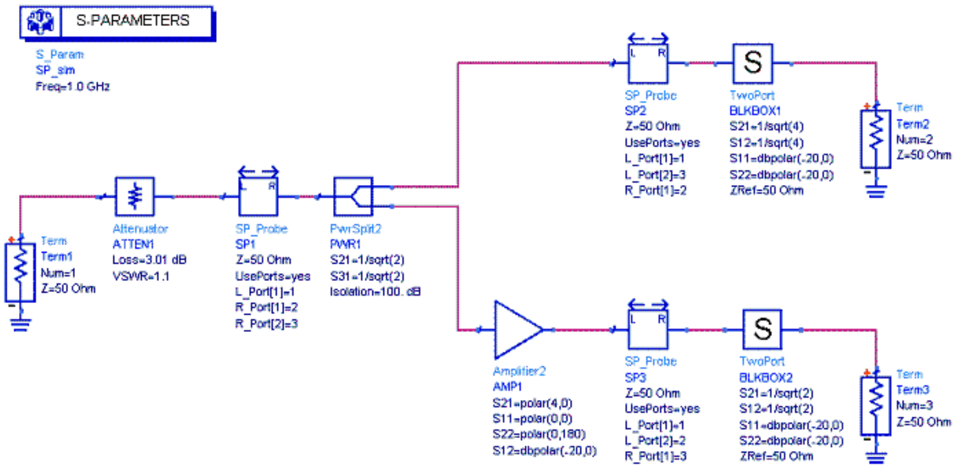
freq	dB(R.S)			
	(1,1)	(1,2)	(2,1)	(2,2)
1.000 GHz	-20.000	-3.010	-3.010	-30.000

The overall s-parameters between Term1 and Term2 are:

freq	dB(SP.S)			
	(1,1)	(1,2)	(2,1)	(2,2)
1.000 GHz	-10.737	-23.161	8.880	-24.962

Note the SP_Probe right side has R_Port[1]=2. This nomenclature tells ADS that R.S will use Term2 (Num=2). However, the right side is always set to be port one for R.S, so in this case *Term2* is still port 2 in the results. The bottom line is, the right side will always become port 1 of the R.S results and left side will always be the last port of the L.S results.

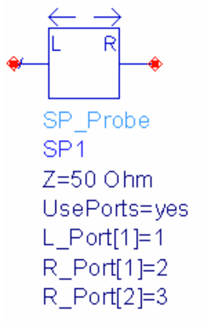
The next example shows the use of the SP_Probe with more than two ports and why careful attention to port assignments is important. The following 3 port schematic is used:



sp_probe_test_3port

First, note that probe SP1 to the left is equivalent to ATTEN1, SP2 to the right is equivalent to BLKBOX1 and SP3 to the right is equivalent to BLKBOX2. These are simple component results. The rest of this discussion will focus on three test cases SP1 right, SP2 left and SP3 left.

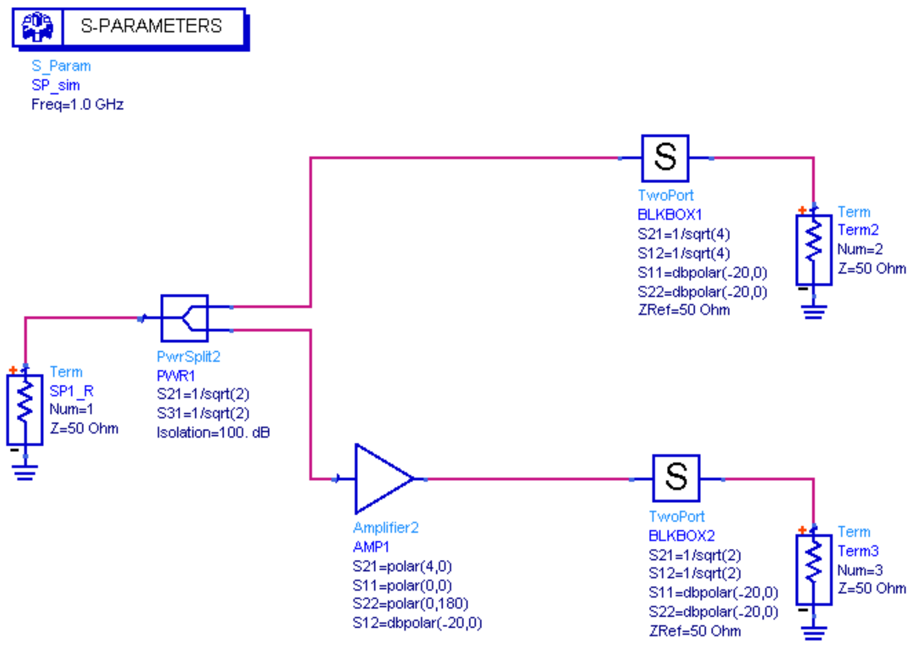
Test case SP1 right side:



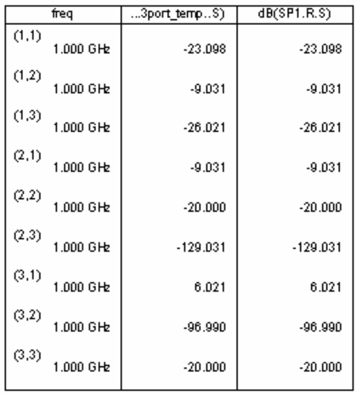
Note that in this setup R_Port[1]=2 (Term2) and R_Port[2]=3 (Term3), since the right port is always port one we have:

SP1.R Port 1 = the probe SP1 right side
SP1.R Port 2 = Term2
SP1.R Port 3 = Term3

Therefore, when analyzing SP_Probe SP1 to the right side, the equivalent circuit is:

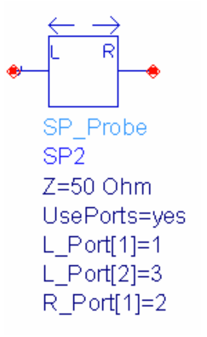


sp_probe_test_3port_temp



	freq	...3port_temp..S)	dB(SP1.R.S)
(1,1)	1.000 GHz	-23.098	-23.098
(1,2)	1.000 GHz	-9.031	-9.031
(1,3)	1.000 GHz	-26.021	-26.021
(2,1)	1.000 GHz	-9.031	-9.031
(2,2)	1.000 GHz	-20.000	-20.000
(2,3)	1.000 GHz	-129.031	-129.031
(3,1)	1.000 GHz	6.021	6.021
(3,2)	1.000 GHz	-96.990	-96.990
(3,3)	1.000 GHz	-20.000	-20.000

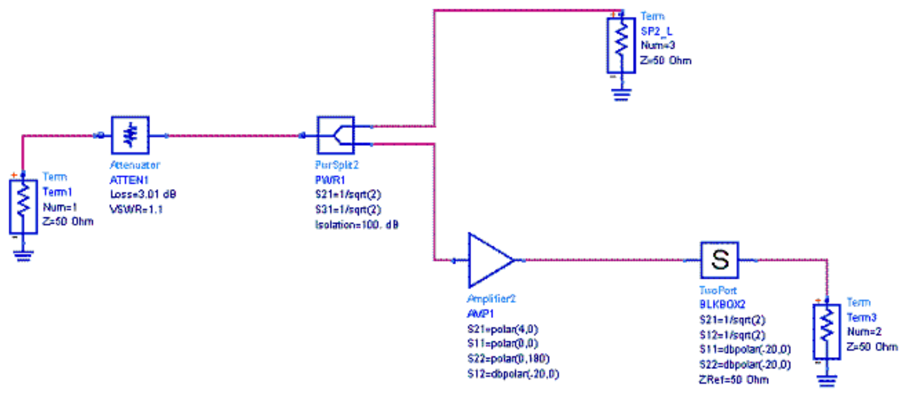
Test case SP2 left side:



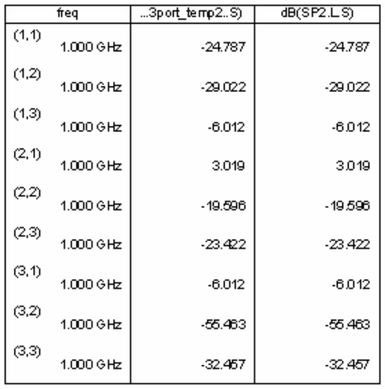
Note that in this setup L_Port[1]=1 (Term1) and L_Port[2]=3 (Term3), since the left port is always the last port we have:

SP2.L Port 1 = Term1
SP2.L Port 2 = Term3
SP2.L Port 3 = the probe SP2 left side

Therefore when analyzing SP_Probe SP2 to the left side, the equivalent circuit is:

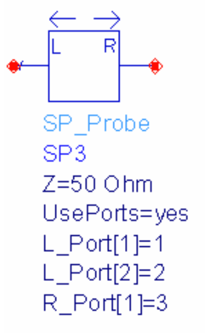


sp_probe_test_3port_temp2



	freq	...3port_temp2..S)	dB(SP2.L.S)
(1,1)	1.000 GHz	-24.787	-24.787
(1,2)	1.000 GHz	-29.022	-29.022
(1,3)	1.000 GHz	-6.012	-6.012
(2,1)	1.000 GHz	3.019	3.019
(2,2)	1.000 GHz	-19.596	-19.596
(2,3)	1.000 GHz	-23.422	-23.422
(3,1)	1.000 GHz	-6.012	-6.012
(3,2)	1.000 GHz	-55.463	-55.463
(3,3)	1.000 GHz	-32.457	-32.457

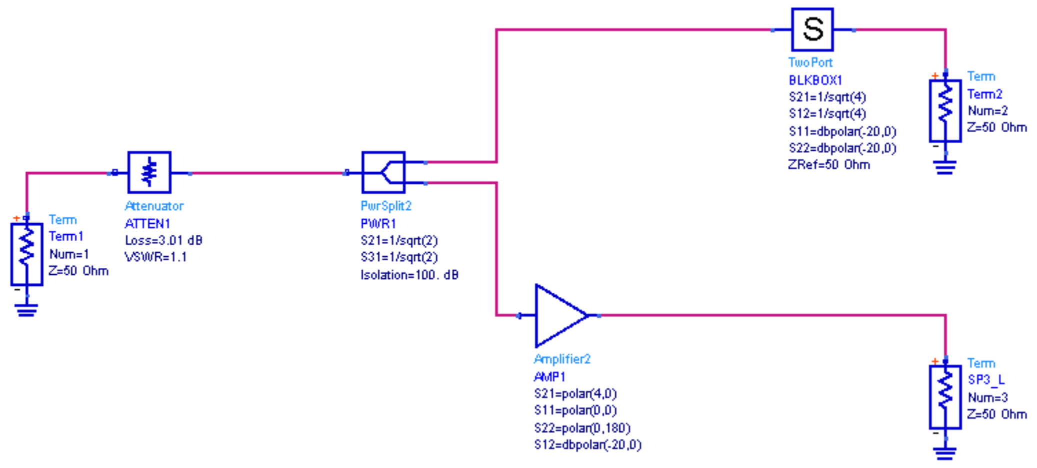
Test case SP3 left side:



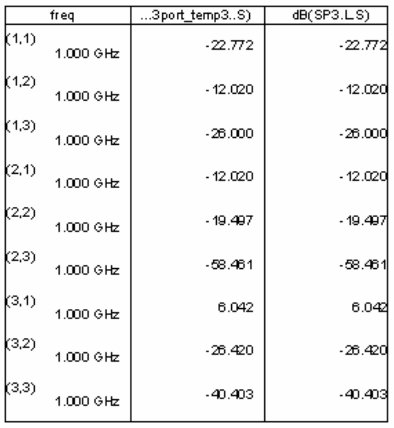
Note that in this setup L_Port[1]=1 (Term1) and L_Port[2]=2 (Term2), since the left port is always the last port we have:

SP3.L Port 1 = Term1
SP3.L Port 2 = Term2
SP3.L Port 3 = the probe SP2 left side

Therefore when analyzing SP_Probe SP3 to the left side, the equivalent circuit is:



sp_probe_test_3port_temp3



	freq	...3port_temp3..S)	dB(SP3.L.S)
(1,1)	1.000 GHz	-22.772	-22.772
(1,2)	1.000 GHz	-12.020	-12.020
(1,3)	1.000 GHz	-26.000	-26.000
(2,1)	1.000 GHz	-12.020	-12.020
(2,2)	1.000 GHz	-19.497	-19.497
(2,3)	1.000 GHz	-58.461	-58.461
(3,1)	1.000 GHz	6.042	6.042
(3,2)	1.000 GHz	-26.420	-26.420
(3,3)	1.000 GHz	-40.403	-40.403

Using Symbolically Defined Devices (SDD)

Location: $HPEESOF_DIR/examples/Tutorial/SDD_Examples_wrk

Objective

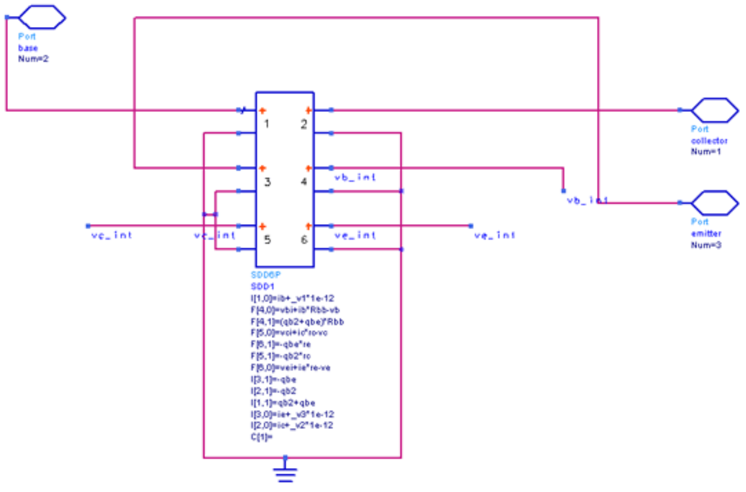
This example provides a number of Symbolically Defined Device (SDD) examples. These devices allow users to add custom nonlinear models in their design. Examples include nonlinear resistor, diode, nonlinear amplifier, mixer, voltage-controlled oscillator (VCO) and Gummel-Poon BJT model. It also shows how to use the weighting functions and control current function in SDD.

Setup

1. "Cubic" implements a third-order nonlinear "resistor" using SDD.
2. "SDD_Diode" builds a diode model, which can also be used as a varactor. "TestDiode" simulates the bias-dependent capacitance of the SDD_Diode.
3. "NonlinearAmp" realizes a nonlinear amplifier which is tested in "NonlinearAmp". The user can set input and output resistances, small-signal gain, and saturated output voltage.
4. "IdealMixer" realizes an ideal mixer (really a voltage multiplier) using a three port SDD.
5. "RemCC" shows how to use the control current function in SDD. The voltage at the SDD is generated to be equal to the instantaneous power dissipated in a resistor.
6. "Sine" shows an SDD implementation of an ideal VCO. A sine wave is applied at the input as the control voltage.
7. "VCObd" realizes a VCO using voltage controlled sources.
8. "WeightLPF" shows how to use weighting functions in SDD to define equations with derivatives and more complicated expressions.
9. "RecConv" shows a pulse driving a low-pass filter, implemented via a polynomial.
10. "GumPoon" implements a Gummel-Poon BJT model via an SDD. "GumPoon_Dctest" simulates the DC I-V curves of the device.
11. "GilCellMix" is a Gilbert cell mixer implemented with the SDD Gummel-Poon model (The biasing resistors have not been optimized so the mixer does not have good

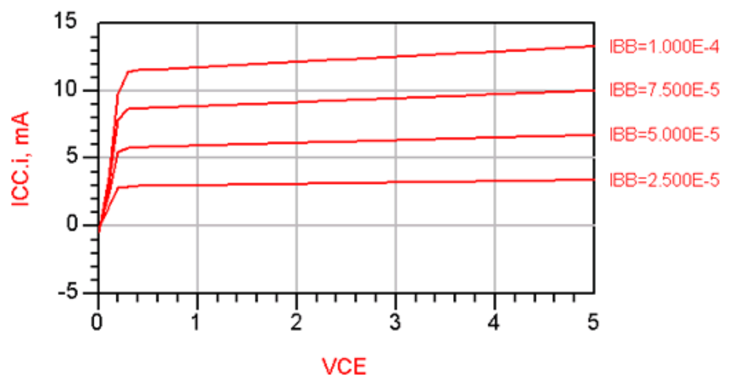
conversion gain.) "MixHBTest" performs a harmonic balance simulation of the mixer.

12. "SDD_cap" is a simple, voltage-dependent capacitor model. "SDD_cap2" is a similar voltage-dependent capacitor model, that behaves identically. "Test_SDD_cap" simulates the capacitance versus bias voltage.



Analysis

Figure 1: I-V curves of the Gummel-Poon BJT model



Notes

- Simulation controllers used: DC, HB, Transient, AC.

Various Examples on using ADS Simulation Controllers

Location: $HPEESOF_DIR/examples/Tutorial/SimModels_wrk

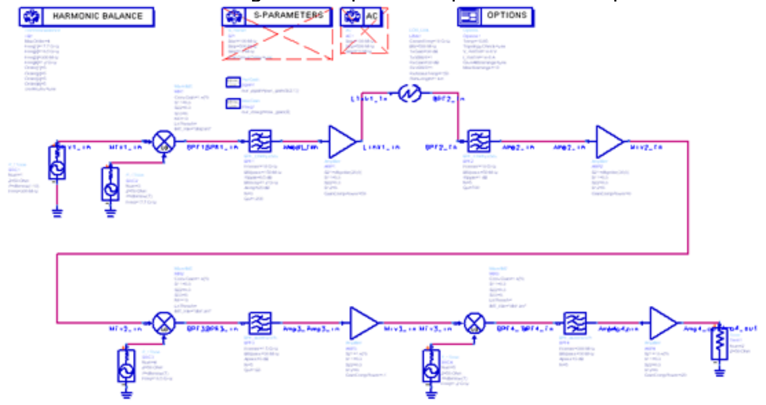
Objective

This workspace contains simple examples that illustrate how to use the various simulation controllers in typical setups.

Setup

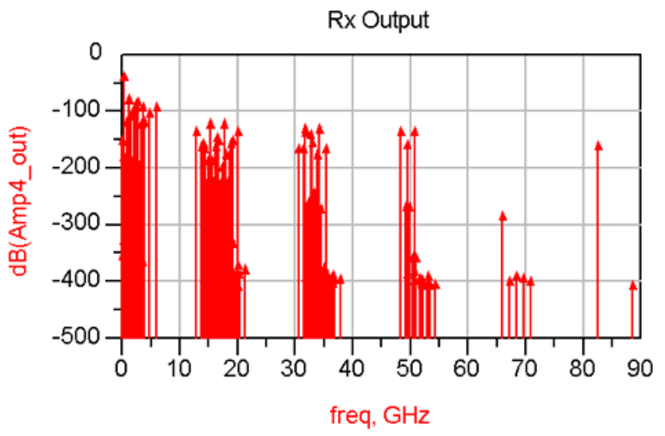
1. "AC1" shows an AC simulation.
2. "HB1" shows a harmonic balance (HB) setup.
3. "DC1" and "DC2" show DC simulations. "HB1" and "HB2" show the HB simulation of a one and two stage amplifier.
4. "LSSP1" and "LSSP2" show the large signal S-parameter simulation of an amplifier and a mizer.
5. "Mix1" show a HB simulation of a mixer. "Mix1_noise" shows the noise figure simulation. "Mix1_ssmix" shows a small signal mixer simulation. "Mix2_convgain" and "Mix2_TOI" uses a HB to calculate the conversion gain and third order intermodulation.
6. "OSC1" uses a OscTest component to check the loop gain. "OSC2" used a OscPort to

check the oscillation waveform. "OSC3" simulates the phase noise.
7. "RF_SYS1" simulates a whole RF system using HB.
8. "SP1" shows a simple S-parameter simulation of an amplifier.
9. "TRAN1" shows a transient simulation of a mixer.
10. "XDB1" simulates the gain compression point of an amplifier.



Analysis

Figure 1: Output spectrum



Notes

- Simulation controllers used: DC,AC, HB, XDB, S-Paramters, Transient, LSSP.

VCO Simulations

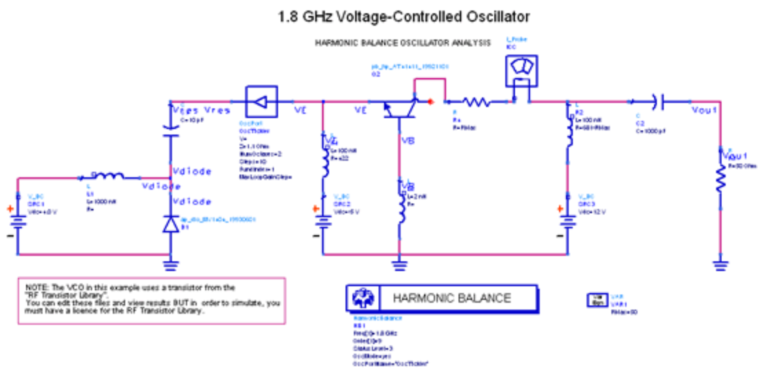
Location: $HPEESOF_DIR/examples/Tutorial/LearnOsc_wrk

Objective

Show several simulations in designing a voltage-controlled oscillator (VCO).

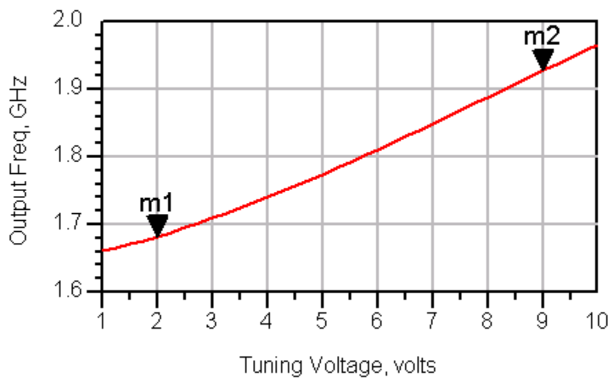
Setup

- "Osctest_VCO" uses the small-signal "probe" component "OscTest" to simulate the small-signal loop gain of the oscillator.
- "HBloopgain_VCO" simulates the large signal loop gain as the amplitude and frequency of the injected signal varies. "OsctestLS" is the probe used to determine the large-signal loop gain.
- "HB_VCO" uses "OscPort" to determine the large-signal steady-state oscillation conditions. The data display shows the steady-state spectrum and time-domain waveform, as well as the oscillation frequency.
- "HB_VCOswp" sweeps the bias voltage on the varactor diode and simulates the VCO tuning characteristic. The data display shows the output power and power flatness versus oscillation frequency, VCO tuning characteristic, tuning linearity deviation, and KVCO in MHz/V.


1.8 GHz Voltage-Controlled Oscillator
HARMONIC BALANCE OSCILLATOR ANALYSIS
Vres
Vdiode
VE
VB
Vout
HARMONIC BALANCE
NOTE: The VCO in this example uses a transistor from the "RF Transistor Library". You can edit these files and view results BUT in order to simulate, you must have a licence for the RF Transistor Library.

Analysis

Figure 1: VCO tuning characteristic



Notes

- Simulation controller used: HB, Transient.

X-Parameters: Generating a Model and Comparing it with a Transistor-Level Simulation

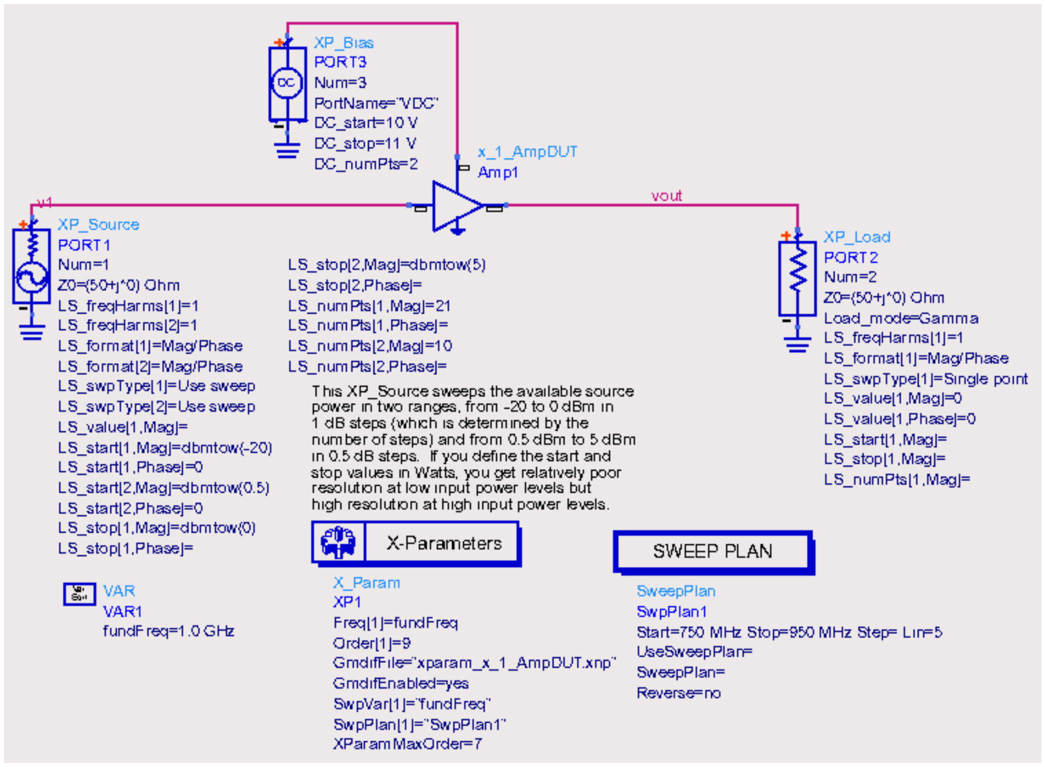
Location: $HPEESOF_DIR/examples/Tutorial/X_parameters_Generation_wrk

Objective

This workspace shows how to generate an X-Parameter model of transistor-level amplifier, then compares a simulation of the model with a simulation of the original schematic.

Setup

"XParamGeneration" generates the X-Parameters of a transistor-level amplifier.



XP_Bias
PORT3
Num=3
PortName="VDC"
DC_start=10 V
DC_stop=11 V
DC_numPts=2

x_1_AmpDUT
Amp1

XP_Source
PORT1
Num=1
Z0=(50+j*0) Ohm
LS_freqHarms[1]=1
LS_freqHarms[2]=1
LS_format[1]=Mag/Phase
LS_format[2]=Mag/Phase
LS_swpType[1]=Use sweep
LS_swpType[2]=Use sweep
LS_value[1,Mag]=
LS_start[1,Mag]=dbmtow(-20)
LS_start[1,Phase]=0
LS_start[2,Mag]=dbmtow(0.5)
LS_start[2,Phase]=0
LS_stop[1,Mag]=dbmtow(0)
LS_stop[1,Phase]=
LS_stop[2,Mag]=dbmtow(5)
LS_stop[2,Phase]=
LS_numPts[1,Mag]=21
LS_numPts[1,Phase]=
LS_numPts[2,Mag]=10
LS_numPts[2,Phase]=

This XP_Source sweeps the available source power in two ranges, from -20 to 0 dBm in 1 dB steps (which is determined by the number of steps) and from 0.5 dBm to 5 dBm in 0.5 dB steps. If you define the start and stop values in Watts, you get relatively poor resolution at low input power levels but high resolution at high input power levels.

XP_Load
PORT2
Num=2
Z0=(50+j*0) Ohm
Load_mode=Gamma
LS_freqHarms[1]=1
LS_format[1]=Mag/Phase
LS_swpType[1]=Single point
LS_value[1,Mag]=0
LS_value[1,Phase]=0
LS_start[1,Mag]=
LS_stop[1,Mag]=
LS_numPts[1,Mag]=

VAR
VAR1
fundFreq=1.0 GHz

X-Parameters
X_Param
XP1
Freq[1]=fundFreq
Order[1]=9
GmdifFile="xparam_x_1_AmpDUT.xnp"
GmdifEnabled=yes
SwpVar[1]="fundFreq"
SwpPlan[1]="SwpPlan1"
XParamMaxOrder=7

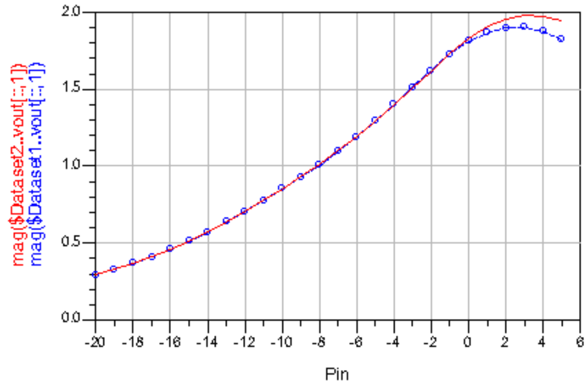
SWEEP PLAN
SweepPlan
SwpPlan1
Start=750 MHz Stop=950 MHz Step= Lin=5
UseSweepPlan=
SweepPlan=
Reverse=no

"HB1Tone_LoadPullPSweep_wXP simulates a load pull of the X-Parameter model. "HB1Tone_LoadPullPSweep simulates a load pull of the transistor-level circuit. It shows nearly identical results for the contours, gain, output power, input reflection coefficient, etc.

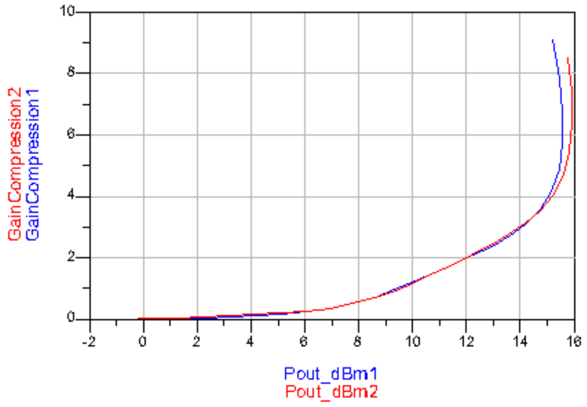
Analysis

"XParamModelTest and CircuitTest" generate datasets that are compared in the ModelVsCircuit data display.

Eqn Dataset1="CircuitTest"
Eqn Dataset2="XParamModelTest"



Gain compression versus output power agreement is very good until the amplifier is driven beyond 4 dB into compression.



Pin	$Dataset2..vout[::,1]	$Dataset1..vout[::,1]
-20.0000	0.2920 / 130.8852	0.2919 / 130.9003
-19.0000	0.3266 / 131.0807	0.3271 / 130.9989
-18.0000	0.3652 / 131.3013	0.3663 / 131.1265
-17.0000	0.4082 / 131.5505	0.4100 / 131.2928
-16.0000	0.4560 / 131.8321	0.4585 / 131.5116
-15.0000	0.5091 / 132.1506	0.5122 / 131.8018
-14.0000	0.5680 / 132.5113	0.5712 / 132.1909
-13.0000	0.6333 / 132.9201	0.6353 / 132.7161
-12.0000	0.7035 / 133.4961	0.7042 / 133.4244
-11.0000	0.7745 / 134.5136	0.7764 / 134.3624
-10.0000	0.8504 / 135.6981	0.8506 / 135.5529
-9.0000	0.9294 / 136.9975	0.9268 / 136.9747

Eqn Pout_dBm1=dBm($Dataset1..vout[1])
Eqn PowerGain1=Pout_dBm1-$Dataset1..Pin
Eqn GainCompression1=PowerGain1[0]-PowerGain1

Eqn Pout_dBm2=dBm($Dataset2..vout[1])
Eqn PowerGain2=Pout_dBm2-$Dataset2..Pin
Eqn GainCompression2=PowerGain2[0]-PowerGain2

Notes

- The datasets generated by the load pull simulations have been removed from the example, so you will have to re-run the load pull simulations to see these results.

X-Parameters: Various Simulations of X-Parameter Models Generated from Measurements

Location: $HPEESOF_DIR/examples/Tutorial/X_parameters_Demo_wrk

Objective

This workspace shows various simulations of X-parameter models generated from measurements of amplifiers. It includes one-, two-tone, and modulated input signals as well as input power and frequency sweeps.

Setup

The xnp files containing X-parameters for the measured amplifiers in this workspace are retrieved from the data subdirectory. These files were obtained from physical DUT measurements on Agilent's Nonlinear Vector Network Analyzer (NVNA.)

The following X-parameter xnp data files are used:

a) ZFL_11AD_X2P..xnp - X-parameter measurements of first individual DUT
b) Z60_2522M_X2P.xnp - X-parameter measurements of second Individual DUT
c) ZFL_11AD_ZX60_2522M_Cascaded_X2P.xnp - X-parameter measurements of above DUTs in cascade
d) ZX60_2252M_DC_X3P.xnp - X-parameter measurements of second Individual DUT with measured DC bias current
e) FP2189_LoadPull_X2P.xnp - Load-dependent X-parameter measurements taken on a packaged FET using NVNA together with a Maury tuner.

The six simulatable designs are best studied in the following order:

a) X2P_SingleAmp_HB_1Tone - This Harmonic Balance simulation provides an introductory view of the
harmonic behavior represented by the measured X-parameters at the input and at the output of an amplifier.
Sweeps of the frequency and power of the single input fundamental tone provide a basis for observing
variations in amplifier behaviors such as in Gain Compression.

b) X2P_CascadedAmps_HB_1Tone - This Harmonic Balance simulation compares the cascading of two
amplifiers represented by X2P components with measured X-parameters to a single X2P component with
the measured X-parameters of the physically cascaded amplifiers, thereby demonstrating fidelity under
cascaded conditions.

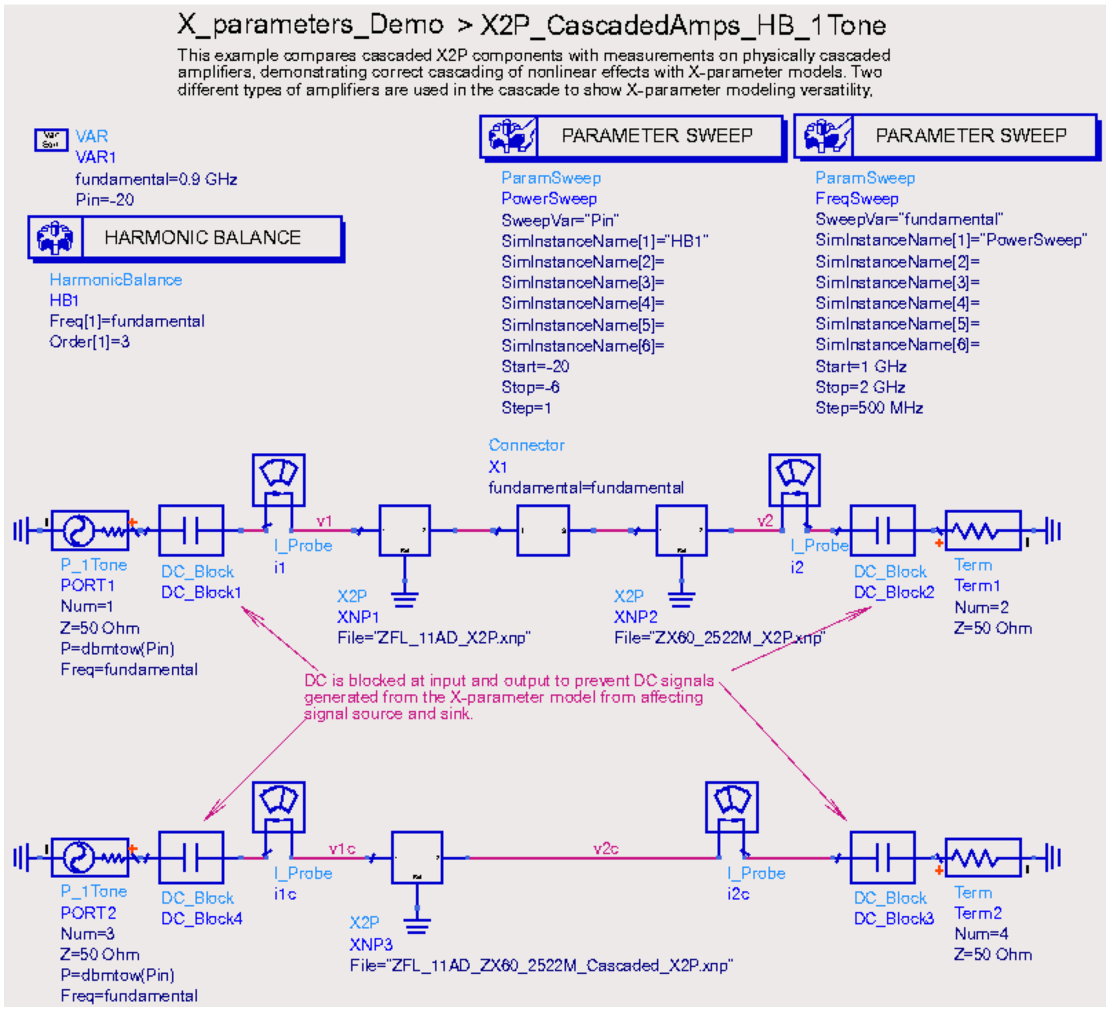
c) X3P_SingleAmp_LoadPull_HB_1Tone - This Harmonic Balance simulation demonstrates how X-parameters can be
used to track load-pull based variations. The X3P component used in this simulation includes measured DC bias current
information, and is therefore able to predict PAE in addition to power delivered as a function of load.

d) X2P_PackagedFET_LoadPull_HB_1Tone - This Harmonic Balance simulation demonstrates the use of load-dependent
X-parameters measured with the NVNA and a Maury tuner to enable accurate simulation of device behavior over a wide
range of load conditions.

e) X2P_SingleAmp_CE_2Tone - This Circuit Envelope simulation demonstrates the use of an X2P component to estimate
IMD3 response to a two-tone input. The simulation is very accurate for narrowly spaced tones, and provides a
reasonable estimate over wider bandwidths using the static gain and phase compression characteristics of the DUT.

f) X3P_SingleAmp_CE_CDMA - This Circuit Envelope simulation demonstrates use of an X3P component in a
communications system analysis. Again, the simulation is very accurate for narrow bandwidths, and provides a reasonable
estimate over wider bandwidths.

"X2P_CascadedAmps_HB_1Tone" compares a simulation of cascaded X2P components, each measured separately, with a simulation of an X2P component from a measurement of the cascaded amplifiers.


X_parameters_Demo > X2P_CascadedAmps_HB_1Tone

This example compares cascaded X2P components with measurements on physically cascaded amplifiers, demonstrating correct cascading of nonlinear effects with X-parameter models. Two different types of amplifiers are used in the cascade to show X-parameter modeling versatility.

VAR
VAR1
fundamental=0.9 GHz
Pin=-20

HARMONIC BALANCE
HarmonicBalance
HB1
Freq[1]=fundamental
Order[1]=3

PARAMETER SWEEP
ParamSweep
PowerSweep
SweepVar="Pin"
SimInstanceName[1]="HB1"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=-20
Stop=-6
Step=1

PARAMETER SWEEP
ParamSweep
FreqSweep
SweepVar="fundamental"
SimInstanceName[1]="PowerSweep"
SimInstanceName[2]=
SimInstanceName[3]=
SimInstanceName[4]=
SimInstanceName[5]=
SimInstanceName[6]=
Start=1 GHz
Stop=2 GHz
Step=500 MHz

Connector
X1
fundamental=fundamental

P_1Tone
PORT1
Num=1
Z=50 Ohm
P=dbmtow(Pin)
Freq=fundamental

DC_Block
DC_Block1

I_Probe
i1

v1

X2P
XNP1
File="ZFL_11AD_X2P.xnp"

X2P
XNP2
File="ZX60_2522M_X2P.xnp"

v2

I_Probe
i2

DC_Block
DC_Block2

Term
Term1
Num=2
Z=50 Ohm

DC is blocked at input and output to prevent DC signals generated from the X-parameter model from affecting signal source and sink.

P_1Tone
PORT2
Num=3
Z=50 Ohm
P=dbmtow(Pin)
Freq=fundamental

DC_Block
DC_Block4

I_Probe
i1c

v1c

X2P
XNP3
File="ZFL_11AD_ZX60_2522M_Cascaded_X2P.xnp"

v2c

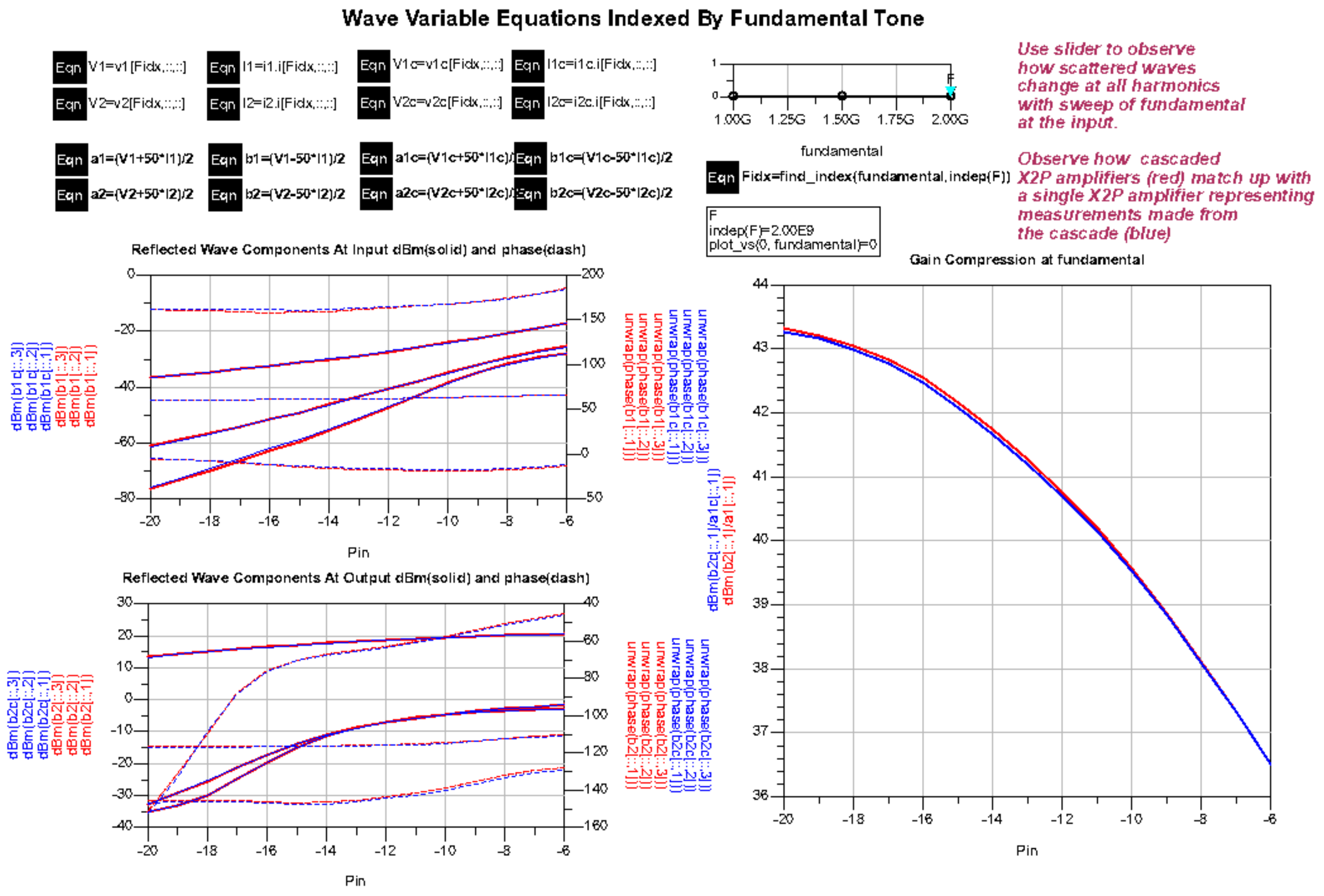
I_Probe
i2c

DC_Block
DC_Block3

Term
Term2
Num=4
Z=50 Ohm

Analysis

The plots below show that a simulation of the cascaded X2P components agrees with a simulation of an X2P component generated from a measurement of the cascaded components.



Notes

- The largest datasets have been removed from this example, so you will have to re-run some simulations to see results.

Yield Analysis of An Impedance Transformer

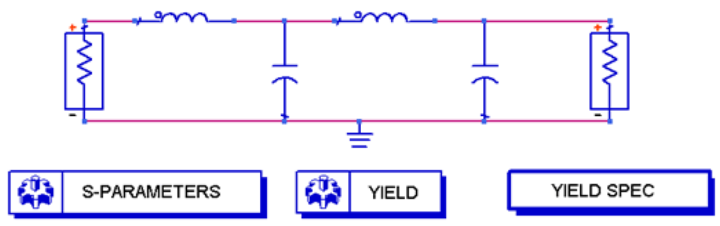
Location: $HPEESOF_DIR/examples/Tutorial/yldex1_wrk

Objective

This example shows the yield analysis of an impedance transformation network.

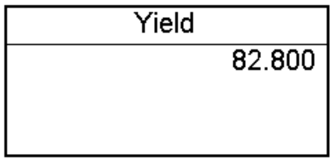
Setup

1. "yldex0" is a yield analysis of an impedance transformation network.
2. The data display shows the yield results, as well as the distribution of return loss for the given range of component values.



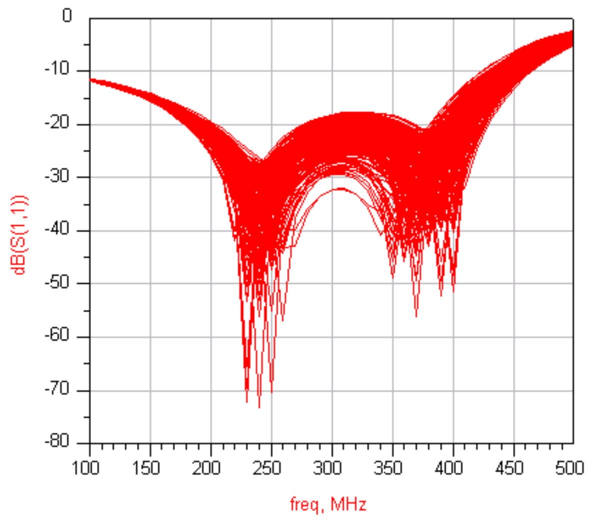
Analysis

Table 1: Yield result



Yield
82.800

Figure 1: Return loss distribution



Notes

- Simulation controllers used: S-Paramters, Yield.

Yield Optimization of An Impedance Transformer

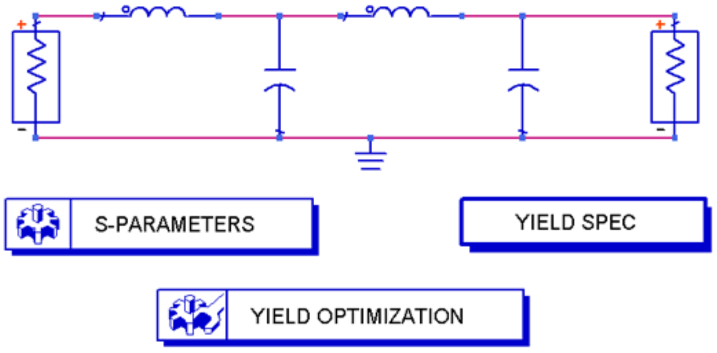
Location: $HPEESOF_DIR/examples/Tutorial/yldoptex1_wrk

Objective

This example shows the yield optimization of an impedance transformation network.

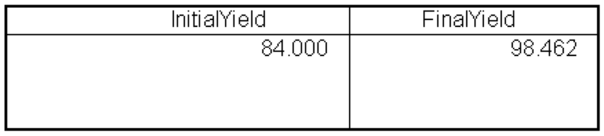
Setup

1. "yldoptex0" is a yield optimization of the impedance transformation network that was obtained from the optimization example in the Tutorial/optex1_wrk workspace. The initial yield estimate is less than 85%, but the yield optimizer finds nominal parameter values that give a yield estimate of >95%.
2. The data display shows the initial and final yield estimates, as well as the best nominal component values and reflection coefficient of the network with the best nominal component values.

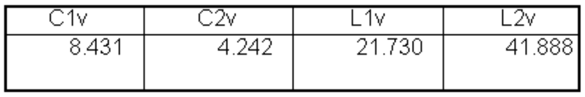


Analysis

Figure 1: Yield before and after the optimization as well as the optimized nominal component values

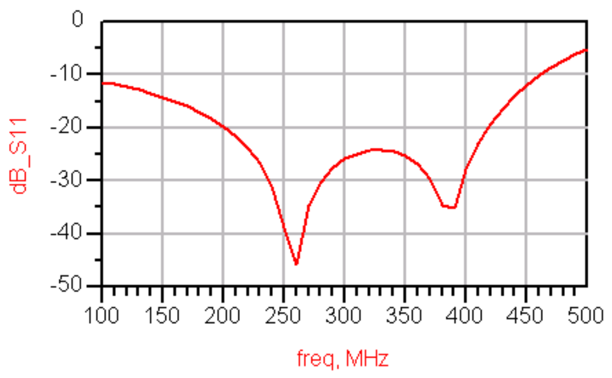


InitialYield	FinalYield
84.000	98.462



C1v	C2v	L1v	L2v
8.431	4.242	21.730	41.888

Figure 2: Optimized return loss



Notes

- Simulation controllers used: S-Parameters, Yield Optimization.

Yield Sensitivity Analysis of A Low Pass Filter

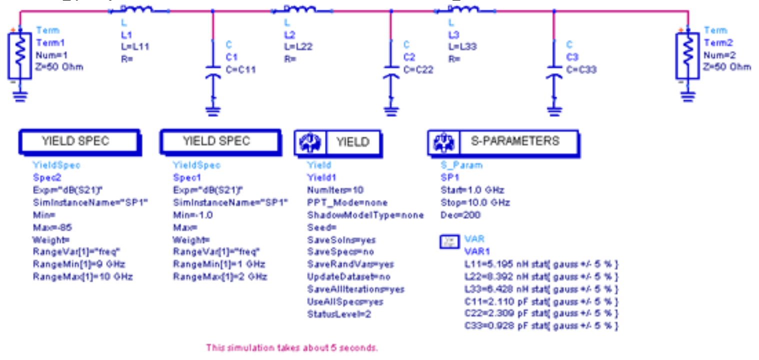
Location: $HPEESOF_DIR/examples/Tutorial/yield_sensitivity_wrk

Objective

This example shows the yield analysis using a low pass filter as an example. The data display contains some equations useful in the yield analysis.

Setup

1. Values of components in the low pass filter are varied using a Gaussian probability distributions, and simulation results are shown in the data display page.
2. For more details on statistical analysis please see "Using Statistical Design" of Tuning, Optimization and Statistical Design.



```
YieldSpec
Spec2
Expr="dB(S21)"
SimInstanceName="SP1"
Min=
Max=-85
Weight=
RangeVar[1]="freq"
RangeMin[1]=9 GHz
RangeMax[1]=10 GHz
```

```
YieldSpec
Spec1
Expr="dB(S21)"
SimInstanceName="SP1"
Min=-1.0
Max=
Weight=
RangeVar[1]="freq"
RangeMin[1]=1 GHz
RangeMax[1]=2 GHz
```

```
Yield
Yield1
NumIters=10
PPT_Mode=none
ShadowModelType=none
Seed=
SaveSolns=yes
SaveSpecs=no
SaveRandVars=yes
UpdateDataset=no
SaveAllIterations=yes
UseAllSpecs=yes
StatusLevel=2
```

```
S_Param
SP1
Start=1.0 GHz
Stop=10.0 GHz
Dec=200
```

```
VAR
VAR1
L11=5.195 nH stat{ gauss +/- 5 % }
L22=8.392 nH stat{ gauss +/- 5 % }
L33=6.428 nH stat{ gauss +/- 5 % }
C11=2.110 pF stat{ gauss +/- 5 % }
C22=2.309 pF stat{ gauss +/- 5 % }
C33=0.928 pF stat{ gauss +/- 5 % }
```

This simulation takes about 5 seconds.

Analysis

Figure 1: Yield Sensitivity Histogram with equations

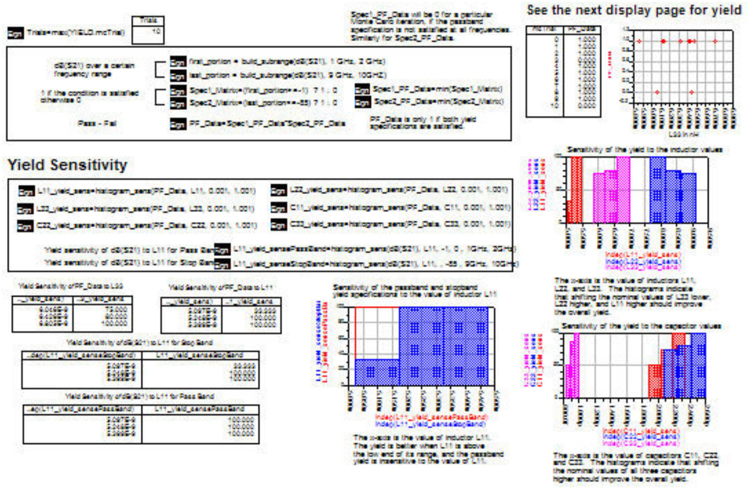
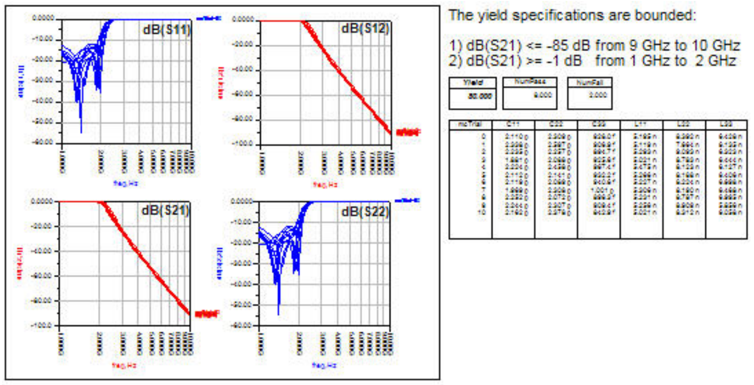

Trials=max(YIELD.mcTrial)
Trials
10
Spec1_PF_Data will be 0 for a particular Monte Carlo iteration, if the passband specification is not satisfied at all frequencies. Similarly for Spec2_PF_Data.
See the next display page for yield
dB(S21) over a certain frequency range
first_portion = build_subrange(dB(S21), 1 GHz, 2 GHz)
last_portion = build_subrange(dB(S21), 9 GHz, 10GHZ)
1 if the condition is satisfied otherwise 0
Spec1_Matrix=(first_portion>=-1) ? 1 : 0
Spec2_Matrix=(last_portion<=-85) ? 1 : 0
Spec1_PF_Data=min(Spec1_Matrix)
Spec2_PF_Data=min(Spec2_Matrix)
Pass - Fail
PF_Data=Spec1_PF_Data*Spec2_PF_Data
PF_Data is only 1 if both yield specifications are satisfied.
Yield Sensitivity
L11_yield_sens=histogram_sens(PF_Data, L11, 0.001, 1.001)
L22_yield_sens=histogram_sens(PF_Data, L22, 0.001, 1.001)
L33_yield_sens=histogram_sens(PF_Data, L33, 0.001, 1.001)
C11_yield_sens=histogram_sens(PF_Data, C11, 0.001, 1.001)
C22_yield_sens=histogram_sens(PF_Data, C22, 0.001, 1.001)
C33_yield_sens=histogram_sens(PF_Data, C33, 0.001, 1.001)
Yield sensitivity of dB(S21) to L11 for Pass Band
L11_yield_sensePassBand=histogram_sens(dB(S21), L11, -1, 0 , 1GHz, 2GHz)
Yield sensitivity of dB(S21) to L11 for Stop Band
L11_yield_senseStopBand=histogram_sens(dB(S21), L11, , -85 , 9GHz, 10GHz)
Yield Sensitivity of PF_Data to L33
Yield Sensitivity of PF_Data to L11
Yield Sensitivity of dB(S21) to L11 for Stop Band
Yield Sensitivity of dB(S21) to L11 for Pass Band
Sensitivity of the passband and stopband yield specifications to the value of inductor L11
The x-axis is the value of inductor L11. The yield is better when L11 is above the low end of its range, and the passband yield is insensitive to the value of L11.
Sensitivity of the yield to the inductor values
The x-axis is the value of inductors L11, L22, and L33. The histograms indicate that shifting the nominal values of L22 lower, L33 higher, and L11 higher should improve the overall yield.
Sensitivity of the yield to the capacitor values
The x-axis is the value of capacitors C11, C22, and C33. The histograms indicate that shifting the nominal values of all three capacitors higher should improve the overall yield.
L33 in nH

Figure 2: Yield Result



The yield specifications are bounded:

1) dB(S21) <= -85 dB from 9 GHz to 10 GHz
2) dB(S21) >= -1 dB from 1 GHz to 2 GHz

Yield	NumPass	NumFail
80.000	8.000	2.000

Notes

1. Simulation controllers used: S-Parameters, Yield.

Low Pass Filter Demo

Location:$HPEESOF_DIR/examples/Tutorial/LPF_Design_Demo_wrk

Objective

This example shows the use of ADS 2011 in the design of a simple low pass filter.

Setup

It begins with a simulation of the filter implemented with ideal, lumped elements and includes component value sweeps. Then the ideal, lumped elements are replaced with PDK models in a layout. This more realistic filter is simulated and compared with the ideal design. Then we run Momentum (using the substrate stack up in the PDK) on the layout and add these results to the comparison. Finally, we show how to modify the substrate stack up, replacing the ideal, sheet conductors with ones with finite thickness and conductivity, and re-run the Momentum simulation for comparison.

Notes

For more details, see Designing a Simple Low Pass Filter.

W-element Extraction Example

Location: $HPEESOF_DIR/examples/Tutorial/WExt_example

Objective

W-element is a general frequency-dependent transmission line model in terms of RLGC matrices. For more details, see *W_Element (Multi-Conductor Transmission Lines)* (ccsim).

This example shows how to use W-element Extraction controller to generate W-element files in various formats for multilayer T-Line components. This feature allows you to obtain W-element models from the accurate and efficient ADS multilayer T-Line models, which are advanced in many aspects such as skin effect modeling, conductor surface roughness modeling, dielectric loss modeling etc. Note that only the tabular format can preserve the full accuracy while static format cannot by definition. For more information on W-element Extraction, see *W-element Extraction* (ccdist).

In this example, the W-element files (RLGC) are generated for a multilayer T-Line component in both static and tabular formats. S-Parameter results calculated from both are compared to that from the original multilayer component. It is illustrated that the tabular format preserves the accuracy of the multilayer model while the static format loses accuracy for higher frequencies.

Setup

- **Multilayer**: Generates the W-element file and does the S-Parameter simulation using the original multilayer T-Line component. Use parameter
  - W_FileFormat in CLine1 to change the output W-element file format.
  - W_File to change the output file name and also the HSPICE subckt name. HSPICE subckt ckt_case1 uses the static file and ckt_case2 uses the tabular file.
- **Static**: Does the S-Parameter simulation using the static W-element file.
- **Tabular**: Does the S-Parameter simulation using the tabular W-element file.

Notes

The S-Parameters controller and the terminations are not necessary for outputting W-element files. Removing them will not affect the W-element output. If the S-Parameter controller is removed from the design Multilayer, a warning message "no output was generated" will occur during simulations because the W-element Extraction controller does not writes any data to the dataset file. If the terminations and connections to them are removed, there might be warning messages regarding unconnected nodes.