NOTICE: This document contains references to Agilent Technologies. Agilent’s former Test and Measurement business has become Keysight Technologies. For more information, go to www.keysight.com.
Acknowledgments

Mentor Graphics is a trademark of Mentor Graphics Corporation in the U.S. and other countries. Microsoft®, Windows®, MS Windows®, Windows NT®, and MS-DOS® are U.S. registered trademarks of Microsoft Corporation. Pentium® is a U.S. registered trademark of Intel Corporation. PostScript® and Acrobat® are trademarks of Adobe Systems Incorporated. UNIX® is a registered trademark of the Open Group. Java™ is a U.S. trademark of Sun Microsystems, Inc. SystemC® is a registered trademark of Open SystemC Initiative, Inc. in the United States and other countries and is used with permission. MATLAB® is a U.S. registered trademark of The Math Works, Inc.. HiSIM2 source code, and all copyrights, trade secrets or other intellectual property rights in and to the source code in its entirety, is owned by Hiroshima University and STARC.

Errata

The SystemVue product may contain references to "HP" or "HPEESOF" such as in file names and directory names. The business entity formerly known as "HP EEsoph" is now part of Agilent Technologies and is known as "Agilent EEsoph". To avoid broken functionality and to maintain backward compatibility for our customers, we did not change all the names and labels that contain "HP" or "HPEESOF" references.

Warranty

The material contained in this document is provided "as is", and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Portions of this product is derivative work based on the University of California Ptolemy Software System.

In no event shall the University of California be liable to any party for direct, indirect, special, incidental, or consequential damages arising out of the use of this software and its documentation, even if the University of California has been advised of the possibility of such damage.

The University of California specifically disclaims any warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The software provided hereunder is on an "as is" basis and the University of California has no obligation to provide maintenance, support, updates, enhancements, or modifications.

Portions of this product include code developed at the University of Maryland, for these portions the following notice applies.

In no event shall the University of Maryland be liable to any party for direct, indirect, special, incidental, or consequential damages arising out of the use of this software and its documentation, even if the University of Maryland has been advised of the possibility of such damage.

The University of Maryland specifically disclaims any warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. the software provided hereunder is on an "as is" basis, and the University of Maryland has no
Portions of this product include the SystemC software licensed under Open Source terms, which are available for download at http://systemc.org/. This software is redistributed by Agilent. The Contributors of the SystemC software provide this software "as is" and offer no warranty of any kind, express or implied, including without limitation warranties or conditions or title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose. Contributors shall not be liable for any damages of any kind including without limitation direct, indirect, special, incidental and consequential damages, such as lost profits. Any provisions that differ from this disclaimer are offered by Agilent only.

With respect to the portion of the Licensed Materials that describes the software and provides instructions concerning its operation and related matters, "use" includes the right to download and print such materials solely for the purpose described above.

**Restricted Rights Legend** If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.
**About Radar Baseband Verification Library**

**Introduction**

The Radar Model Library provides over 35 highly-parameterized primitive blocks and higher-level reference designs to create a working radar system, especially targeting pulsed-doppler (PD) radar architectures. The blockset and its example workspaces serve as algorithmic and architectural reference designs to verify radar performance under different signal conditions, target and RCS scenarios, clutter conditions, jammers and environmental interferers, and different receiver algorithms.

**Radar Library Applications**

The library is targeted for Radar system designers in military, regulatory, commercial, avionics, medical, automotive, research, academic, and consulting applications.

- Accurate radar system architecture and scenario analysis
  - Include realistic RF effects, clutter, fading, and directly-measured waveforms
  - Leverage the existing math, HDL, C++ algorithms
  - Continue into hardware test using the same SystemVue environment and IP
  - Ideal for proposals, hardware design, or verification.
- Algorithmic reference & test vector generation for baseband DSP hardware design
- Precisely-degraded BB/RF signal generation for receiver testing
  - Save time by verifying algorithms, prior to targeted hardware implementation
  - Reduce NRE and scripting with regression suites of simulated scenarios
  - Re-use Agilent equipment assets with SystemVue for functional verification
  - Reduce the need for expensive chambers, hardware emulators, faders, and field testing in the early phases of design

**Radar Library Block List**
**Signal Sources**

- **LFM** – Linear FM wave generator
- **NLFM** – Nonlinear FM wave generator
- **BarkerCode** – Poly phase code wave generator
- **FrankCode** – Frank code wave generator
- **ZCCode** – Zadoff–Chu code wave generator
- **MatchedSrc** – generate the matched source signal for pulse compression

**Signal Processing**

- **Detector** – Video Signal Detector
- **PC** – Pulse Compression processing
- **PD** – Pulse Doppler Processing
- **MTI** – Moving Target Indication
- **MTD** – Moving Target Detection
- **CFAR** – Constant false alarm rate process

**Transmitter**

- **CICInterp** – Interpolation with cascaded CIC filters
- **DUC** – Digital Up Converter, BB to IF
- **Tx** – Transmitter Front End

**Receiver**

- **CICDecimate** – decimation with cascaded CIC filters
- **DDC** – Digital Down Converter, IF to BB
- **Rx** – Receiver Front End

**Environment**

- **RCS** – Cross-Section modeling
- **Target** – Target modeling, including RCS, Doppler effect, Delay, Attenuation
- **Clutter** – Clutter modeling

**Measurement**

- **Pd Measurement** – Detection Probability Estimation
- **Pf Measurement** – False Alarm Rate Estimation

---

**Radar Examples and Test Benches**

**Transmitter Measurement**

- **PDRADAR_Tx_Waveform** - This example measures the radar Transmitter RF signal and IF signal waveforms and spectrums

1. The radar signal is a linear frequency modulation pulse in this example.
2. The simulation is carried out with digital IF and analog RF front end, where, both DUC/DDC and front end circuit effects are considered.
3. Results are shown in associated graphs.
4. Simulation time is about 12.95 seconds on an Intel Core 2 E6850/3.0G 4GB PC powered by MS Windows XP and SystemVue2010.07.
## Receiver Measurement

### PDRADAR_Rx_Waveform - This example measures the waveform of a PD RADAR receiver with clutter and noise

1. The waveform includes the components from target echo, clutter and noise.
2. Both the RF waveform and the RF spectrum of the received signal are measured.
3. On an Intel Core 2 E6850/3.0G 4GB PC powered by MS Windows XP and SystemVue2010.07, simulation time is about 1 minute.

### PDRADAR_Clutter - This example measures the clutter signal of the radar environment. The clutter refers to radio frequency (RF) echoes returned from targets which are uninteresting to the radar operators.

1. A statistical model is set up to simulate the real world clutter.
2. The clutter has certain magnitude probability density functions and power spectrum densities.
3. Results are shown in associated graphs.
4. Simulation time is about 11.93 seconds on an Intel Core 2 E6850/3.0G 4GB PC powered by MS Windows XP and SystemVue2010.07.

### PDRADAR_DynamicRange - This example measures dynamic range of the radar receiver. The dynamic range is the input signal power range to be amplified without distortion.

1. Measured the 1 dB gain compression power of the low noise amplifier in RF.
2. Measured the output IF signal power.
3. Measured the baseband signal power after ADC and digital down converter.
4. Results are shown in associated graphs.
5. Simulation time is about 10.26 seconds on an Intel Core 2 E6850/3.0G 4GB PC powered by MS Windows XP and SystemVue2010.07.

### PDRADAR_Selectivity - This example measures the adjacent band selectivity of the radar receiver. Radar frequency sensitivity is a very important characteristic as it determines radar’s interference immunity.

1. A filtered noise is constructed to simulates the interference in the adjacent band of the radar signal.
2. The simulation is carried out with digital IF and analog RF front end, where, both DUC/DDC and front end circuit effects are considered.
3. Results are shown in associated graphs.
4. Simulation time is about 19585.87 seconds on an Intel Xeon CPU 5130 2.00GHz, 2 processors, 3G RAM PC powered by MS Windows XP and SystemVue2010.07.

### PDRADAR_Sensitivity - This example measures sensitivity of the radar receiver. Radar sensitivity is determined by the ability to reliably detect weak signals in the presence of noise.

1. A noise model is the simulated thermal noise (The noise spectrum density is -173.975 dBm/Hz).
2. The simulation is carried out with digital IF and analog RF front end, where, both DUC/DDC and front end circuit effects are considered.
3. Results are shown in associated graphs.
4. Simulation time is about 21975.57 seconds on an Intel Core 2 E6850/3.0G 4GB PC powered by MS Windows XP and SystemVue2010.07.
**Performance Measurement**

**PDRADAR_DetectionProbability_AWGN** - This example measures a PD RADAR's detection probability under AWGN.

1. The PD RADAR is detecting a target in a distance of 60km and a velocity of 60m/s
2. The received Signal to Noise Ratio (SNR) is swept from -18dB to -10dB
3. The PD RADAR using a Hamming window based pulse compressor with a BT product of 50
4. The Pulse Doppler (PD) processing utilize a moving target detection (MTD) with a CPI(Coherent Processing Interval) of 32 pulses
5. The CFAR (Constant False Alarm Rate) algorithm is Cell Average (CA)
6. On an Intel Core 2 E6850/3.0G 4GB PC powered by MS Windows XP and SystemVue2010.07, Simulation time is about 45 minutes
7. Users can set up the system parameters or replace a component according to their particular requirements.

**PDRADAR_DetectionProbability** - This example measures a PD RADAR’s detection probability with Clutter

1. The PD RADAR is detecting a target in a distance of 60km and a velocity of 60m/s
2. The received Signal to Noise Ratio (SNR) is swept from -14dB to -6dB
3. The clutter is with a Rayleigh distribution in amplitude and a Gaussian distribution in spectrum and stored in data files for simulation speed up purpose.
4. The Clutter to Noise Ratio (CNR) is set to 10dB
5. The PD RADAR using a Hamming window based pulse compressor with a BT product of 50
6. The Pulse Doppler (PD) processing utilize a moving target detection (MTD) with a CPI(Coherent Processing Interval) of 32 pulses
7. The CFAR (Constant False Alarm Rate) algorithm is Cell Average (CA)
8. On an Intel Core 2 E6850/3.0G 4GB PC powered by MS Windows XP and SystemVue2010.07, Simulation time is about 90 minutes
9. Users can set up the system parameters or replace a component according to their particular requirements.

**PDRADAR_FalseAlarmRate_AWGN** - This example measures a PD RADAR's false alarm rate under AWGN

1. The input to the PD radar receiver is Noise only.
2. The PD RADAR using a Hamming window based pulse compressor with a BT product of 50
3. The Pulse Doppler (PD) processing utilize a moving target detection (MTD) with a CPI(Coherent Processing Interval) of 32 pulses
4. The CFAR (Constant False Alarm Rate) algorithm is Cell Average (CA)
5. On an Intel Core 2 E6850/3.0G 4GB PC powered by MS Windows XP and SystemVue2010.07, Simulation time is about 90 minutes.
6. Users can set up the system parameters or replace a component according to their particular requirements.

**PDRADAR_FalseAlarmRate** - This example measures a PD RADAR’s false alarm rate with Clutter

1. The inputs to the PD radar receiver are Clutter and Noise.
2. The clutter is with a Rayleigh distribution in amplitude and a Gaussian distribution in spectrum and stored in data files for simulation speed up purpose.
3. The Clutter to Noise Ratio (CNR) is set to 10dB
4. The PD RADAR using a Hamming window based pulse compressor with a BT product of 50
5. The Pulse Doppler (PD) processing utilize a moving target detection (MTD) with a CPI(Coherent Processing Interval) of 32 pulses
6. The CFAR (Constant False Alarm Rate) algorithm is Cell Average (CA)
7. On an Intel Core 2 E6850/3.0G 4GB PC powered by MS Windows XP and SystemVue2010.07, Simulation time is about 180 minutes.
8. Users can set up the system parameters or replace a component according to their particular requirements.

**PDRADAR_Measurement** - This example measures a target’s range and velocity in clutter and noise

1. The PD RADAR is detecting a target in a distance of 100km and a velocity of 60m/s
2. The received Signal to Noise Ratio (SNR) is -10dB
3. The clutter is with a Rayleigh distribution in amplitude and a Gaussian distribution in spectrum and stored in data files for simulation speed up purpose.
4. The Clutter to Noise Ratio (CNR) is set to 10dB
5. The PD RADAR using a Hamming window based pulse compressor with a BT product of 50
6. The Pulse Doppler (PD) processing utilize a moving target detection (MTD) with a CPI(Coherent Processing Interval) of 32 pulses
7. The CFAR (Constant False Alarm Rate) algorithm is Cell Average (CA)
8. On an Intel Core 2 E6850/3.0G 4GB PC powered by MS Windows XP and SystemVue2010.07, Simulation time is about 90 minutes.
9. Users can set up the system parameters or replace a component according to their particular requirements.

**Reference**
Radar Environment

Contents

- RADAR Clutter Part (radardesign)
- RADAR RCS Part (radardesign)
- RADAR Target Part (radardesign)
RADAR_Clutter Part

Categories: Environments (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_Clutter (radardesign)</td>
<td>Radar clutter simulation</td>
</tr>
</tbody>
</table>

**RADAR_Clutter**

Description: Radar clutter simulation

Domain: Timed

C++ Code Generation Support: NO

Associated Parts: RADAR Clutter Part (radardesign)

**Model Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF_Freq</td>
<td>RF carrier frequency</td>
<td>1e9</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>PDF</td>
<td>Clutter amplitude probability density: Rayleigh PDF, LogNormal PDF, Weibull PDF</td>
<td>Rayleigh PDF</td>
<td></td>
<td>Enumeration</td>
<td>NO</td>
</tr>
<tr>
<td>VA</td>
<td>Voltage value dependent on PDF</td>
<td>1.0</td>
<td>Float</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>VB</td>
<td>Voltage value dependent on PDF</td>
<td>1.0</td>
<td>Float</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>PSD</td>
<td>Clutter power spectrum density: Gaussian PSD, Cauchy PSD, AllPole PSD</td>
<td>Gaussian PSD</td>
<td></td>
<td>Enumeration</td>
<td>NO</td>
</tr>
<tr>
<td>PA</td>
<td>Parameter value dependent on PSD</td>
<td>1.0</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>PB</td>
<td>Parameter value dependent on PSD</td>
<td>1.0</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>TStep</td>
<td>simulation time step; TStep=0 results in use of externally set TStep</td>
<td>0.0001</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>FilterLen</td>
<td>The PSD filter length, also the fft size</td>
<td>256</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Seed</td>
<td>The initial seed for random sequence generator</td>
<td>1234567</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

**Output Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>output</td>
<td>output signal</td>
<td>envelope</td>
<td>NO</td>
</tr>
<tr>
<td>2</td>
<td>Coeff</td>
<td>Filter Coeff</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Notes/Equations**

1. This model is a timed model used to generate the RADAR clutter with specified amplitude probability and specified power spectrum density.
2. Each firing, 1 token is produced at the output port, FilterLen tokens are produced at the Coeff port.
3. Parameter details,
   - RF_Freq specifies the carrier frequency of the output RF signal.
   - PDF specifies the amplitude distribution of RADAR clutter. Three types are supported as Rayleigh, LogNormal and Weibull.
   - VA specifies the voltage value dependent on PDF.
   - VB specifies the voltage value dependent on PDF.
   - PSD specifies the power spectrum density of RADAR clutter.
   - PA specifies the voltage value dependent on PSD.
   - PB specifies the voltage value dependent on PSD.
   - TStep specifies the time step for the output signal. When TStep = 0, the time...
4. Clutter is used to describe any object that may generate unwanted radar echoes. Clutter is statistically described by a probability density function (PDF) the type of which depends on the nature of clutter itself, the radar operating frequency, and the grazing angle.

- Rayleigh distribution is used to describe the clutter which is composed of many small scatterers which is independent of each other. It is given by
  \[ f(x) = \frac{2x}{V_A} \exp \left( -\frac{x^2}{V_A} \right); x \geq 0 \]
  where \( V_A \) is the mean-squared value of \( x \).

- The log-normal distribution is used to describe the land clutter at low grazing angles. It is given by
  \[ f(x) = \frac{1}{V_B \sqrt{2\pi} x} \exp \left( -\frac{(\ln x - \ln V_A)^2}{2V_B^2} \right); x > 0 \]
  where \( V_A \) is the median of the random variable, \( V_B \) is the standard deviation of the random variable \( \ln(x) \).

- The Weibull distribution is used to describe clutter at low grazing angles (less than five degrees) for frequencies between 1 and 10GHz. It is given by
  \[ f(x) = \frac{V_A}{V_B} \left( \frac{x}{V_B} \right)^{V_A-1} \exp \left( -\frac{x^{V_A}}{V_B} \right); x \geq 0 \]
  where \( V_A \) is the shape parameter.

5. The clutter power spectrum can be written as

\[ S_c(f) = \frac{P_c}{T \sigma_f^2 / 2\pi} \sum_{k=-\infty}^{\infty} \exp \left( \frac{(f-kT)^2}{2\sigma_f^2} \right) \]

where \( T \) is the PRI (Pulse Repetition Interval), \( P_c \) is the clutter power or clutter mean square value, \( \sigma_f \) is the clutter spectral spreading parameter as defined in the above equation.

The clutter PSD is periodic with period equal to \( f_r \) as shown in the following figure.

Three types of clutter spectrum density are supported in this model and defined as

- Gaussian distribution
  The power density of gaussian distribution is
  \[ S(f) = \exp \left( \frac{-f^2}{2 \times P_A^2} \right) \]

- Cauchy distribution
  The power density of cauchy distribution is
  \[ S(f) = \frac{P_A}{P_A + f^2} \]

- AllPole distribution
  The power density of all pole distribution is
  \[ S(f) = \frac{P_A}{P_A + f^2} \]

6. ZMNL (Zero Memory Nonlinearity) method is applied to modeling the coherent random process of clutter. The ZMNL method is shows in the following figure,
A white Gaussian process $V$ is passed through the spectrum shaping filter $H$ which demonstrates the PSD of RADAR clutter, then the coherent Gaussian process $W$ is fed into a zero memory nonlinear transformation to get the clutter process $Z$ with specified PDF. For non-Gaussian PDF, as the ZMNL process will change the signal's PSD, precompensation is applied to the spectrum shaping filter. The filter taps of the spectrum shaping filter $H$ is output at the coeff port. The clutter process $Z$ is output at the output port.

Reference

RADAR_RCS Part

Categories: Environments (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_RCS</td>
<td>Radar target RCS</td>
</tr>
</tbody>
</table>

RADAR_RCS

Description: Radar target RCS
Domain: Timed
C++ Code Generation Support: NO
Associated Parts: RADAR RCS Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Noise probability density or Cumulative distribution cdf function: Const Value, Uniform PDF, Gaussian PDF, Rayleigh PDF, LogNormal PDF, Exponential PDF, Weibull PDF, ChiSquared PDF, Gamma PDF, Beta PDF, F PDF, Binomial CDF, Poisson CDF</td>
<td>Gaussian PDF</td>
<td>Enumeration</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>VA</td>
<td>Voltage value dependent on Type</td>
<td>1.0</td>
<td>Float</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>VB</td>
<td>Voltage value dependent on Type</td>
<td>1.0</td>
<td>Float</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>TStep</td>
<td>simulation time step; TStep=0 results in use of externally set TStep</td>
<td>0.0001</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>DurationTime</td>
<td>Time duration for a certain constant RCS</td>
<td>1</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>Seed</td>
<td>initial seed for random sequence generator</td>
<td>1234567</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Es</td>
<td>output the sqrt of target RCS</td>
<td>envelope</td>
<td>NO</td>
</tr>
<tr>
<td>2</td>
<td>RCS</td>
<td>output the target RCS</td>
<td>real</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This model is a timed model used to generate the Radar Cross Section with user-settable probability density function(pdf). The output values in one duration time do not change.
2. Each firing, 1 token is produced at the Es port, 1 token is produced at the RCS port.
3. Parameter details,
   • Type specifies the probability density or Cumulative distribution cdf function.
   • VA specifies the voltage value dependent on Type as shown in the table pdf function.
   • VB specifies the voltage value dependent on Type as shown in the table pdf function.
   • TStep specifies the output time step. If TStep = 0, the output time step comes from the output Es port.
   • DurationTime specifies the time duration for a certain constant RCS.
   • Seed specifies the initial seed for random sequence generator.

RCS Types
4. A radar detects or tracks a target, and sometimes can classify it, only because there is an echo signal. The radar cross section or RCS is used to describe the echo of the target. The formal definition of radar cross section is

\[ \sigma = \lim_{R \to \infty} 4\pi R^2 \frac{|E_r|^2}{|E_o|^2} \]

where

- \( E_o \) is electric-field strength of the incident wave impinging on the target.
- \( E_r \) is the electric-field strength of the scattered wave at the radar.

R is the range from the radar to the target.

Radar cross section fluctuates as a function of radar grazing angle, polarization and frequency. The RCS of simple bodies can be computed exactly by a solution of the wave equation in a coordinate system. Several statistical distribution is supplied in

<table>
<thead>
<tr>
<th>RCS Type</th>
<th>PDF ( f(v) ) or CDF ( F(v) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian PDF</td>
<td>( f(v) = \frac{1}{\sqrt{2\pi B^2}} e^{-\frac{(v-A)^2}{2B^2}} )</td>
</tr>
<tr>
<td>Uniform PDF</td>
<td>( f(v) = \begin{cases} \frac{1}{B-A} &amp; \text{if } A \leq v \leq B \ 0 &amp; \text{otherwise} \end{cases} )</td>
</tr>
<tr>
<td>Rayleigh PDF</td>
<td>( f(v) = 2\ln(2) \frac{v}{A^2} e^{-\ln(2)(\frac{v}{A})^2} U(v) )</td>
</tr>
<tr>
<td>Log-normal PDF</td>
<td>( f(v) = \begin{cases} \frac{1}{\sqrt{2\pi B^2 v^2}} e^{-\frac{1}{2} \left( \frac{\ln(v)-\frac{A}{B} \ln(2) \sqrt{\ln(2)}}{B} \right)^2} &amp; \text{if } v &gt; 0 \ 0 &amp; \text{otherwise} \end{cases} )</td>
</tr>
<tr>
<td>Exponential PDF</td>
<td>( f(v) = \frac{1}{A} e^{-\frac{\ln(v)}{A}} U(v) )</td>
</tr>
<tr>
<td>Weibull PDF</td>
<td>( f(v) = \frac{2 B \ln(2) v}{A^2} e^{-\ln(2)(\frac{v}{A})^2} U(v) )</td>
</tr>
<tr>
<td>Chi-squared PDF</td>
<td>( f(v) = \frac{v^{A-1} e^{-v/2}}{2^A \Gamma(A)} U(v) )</td>
</tr>
<tr>
<td>Gamma PDF</td>
<td>( f(v) = \frac{A^B}{\Gamma(B)} e^{-Av} v^{B-1} U(v) )</td>
</tr>
<tr>
<td>Beta PDF</td>
<td>( f(v) = \begin{cases} \frac{\Gamma(A+B)}{\Gamma(A)\Gamma(B)} v^{A-1} (1-v)^{B-1} &amp; \text{for } 0 &lt; v &lt; 1 \ 0 &amp; \text{otherwise} \end{cases} )</td>
</tr>
<tr>
<td>F PDF</td>
<td>( f(v) = A^{k/2} B^{B/2} \frac{\Gamma(A+B)}{\Gamma(A/2)\Gamma(B/2)} \frac{v^{A/2-1}}{(A+B)^{B/(2)}} U(v) )</td>
</tr>
<tr>
<td>Binomial CDF</td>
<td>( F(v) = \sum_{k=0}^{v} \frac{A^k}{k^k} (B^{1-k}) )</td>
</tr>
<tr>
<td>Poisson CDF</td>
<td>( F(v) = \sum_{k=0}^{v} \frac{A^k}{k!} e^{-A} )</td>
</tr>
</tbody>
</table>
this model to model the RCS and output in port RCS. The port Es outputs the square root of the distribution.

Reference

1. Bassem R. Mahafza, RADAR SIGNAL ANALYSIS AND PROCESSING USING MATLAB, Chapter 1.
RADAR_Target Part

Radar target simulation, including RCS, Doppler effect, Delay and Attenuation

Categories: Environments (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

Model
RADAR_Target (radardesign)

RADAR_Target

Description: Radar target simulation, including RCS, Doppler effect, Delay and Attenuation
Associated Parts: RADAR Target Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RF_Freq</td>
<td>RF carrier frequency ([0:inf])</td>
<td>10000000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>Distance</td>
<td>The distance between radar antenna and target ([0:inf])</td>
<td>100000</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>Attenuation</td>
<td>The atmosphere and rain attenuation ([0:inf])</td>
<td>1</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>Vt</td>
<td>The target Radial Velocity, &gt;0 represents approach radar, &lt;0 represents keep away from radar ([0:inf])</td>
<td>0</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>RCS_Type</td>
<td>Statistic model for target RCS: Swerling 0, Swerling I, Swerling II, Swerling III, Swerling IV</td>
<td>Swerling 0</td>
<td>none</td>
<td>Enumeration</td>
<td>NO</td>
</tr>
<tr>
<td>RCS_VA</td>
<td>Parameter dependent on RCS_Type</td>
<td>1</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>RCS_VB</td>
<td>Parameter value dependent on RCS_Type</td>
<td>1</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>TStep</td>
<td>simulation time step; TStep=0 results in use of externally set TStep ([0:inf])</td>
<td>0</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>DurationTime</td>
<td>Time duration for a certain constant RCS ([TStep:inf])</td>
<td>1</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>IncludeProp</td>
<td>Include the propagation effect or not: NO, YES</td>
<td>NO</td>
<td>none</td>
<td>Enumeration</td>
<td>NO</td>
</tr>
</tbody>
</table>

Input Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>input</td>
<td>Terminal: Standard Data Port</td>
<td>envelope</td>
<td>NO</td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>output</td>
<td>Terminal: Standard Data Port</td>
<td>envelope</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This subnetwork implements the target modeling for RADAR system with RCS, transmission loss, transmission delay and doppler shift. Five Swerling models are supported for target RCS fluctuation.
2. The RADAR_Target schematic is shown below,
RADAR_Target Schematic

3. Parameter details,
   - RF_FREQ specifies the carrier frequency of the input RF envelop signal.
   - Distance specifies the distance between RADAR antenna and the target.
   - Vt specifies the velocity of the target, when Vt > 0, the target is moving toward the RADAR antenna, when Vt < 0, the target is moving far away from RADAR antenna, when Vt = 0, the target is stationary.
   - RCS_Type specifies the type of statistic mode for target RCS.
   - RCS_VA specifies the parameter dependent on RCS_Type.
   - RCS_VB specifies the parameter dependent on RCS_Type.
   - TStep specifies the time step of input signal. If TStep = 0, the time step of input signal comes from the input port.
   - Duration specifies the time duration for a certain constant RCS.
   - IncludeProp specifies whether the propagation effect is included or not.

4. The target acts as a virtual antenna to reflect the electromagnetic waves sending from RADAR antenna to RADAR antenna. The echo in RADAR antenna is used to analyze the target. In this model, the input is the transmit signal in RADAR transmit antenna, the output is the echo signal from target in RADAR receive antenna. The baseband signal in RADAR transmit antenna is defined as
   \[ u(t) = a(t)e^{j\phi(t)} \]
   The RF signal in RADAR transmit antenna is defined as
   \[ s(t) = u(t)e^{j\omega t} \]
   The echo in RADAR antenna in R(t) distance is defined as
   \[ s_r(t) = s(t - \frac{2R(t)}{c}) \times k \times \sigma \]
   where k is the propagation effect and \( \sigma \) is RADAR RCS fluctuation.

   - propagation effect
     According to RADAR equation, the power density of the echo in RADAR antenna at a distance R away from the RADAR using a directive antenna of gain is then given by
     \[ P_r = \frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 R^4} \]
     where P_t is the power density of the transmit signal, \( \lambda \) is the wave length, the \( \sigma \) is RADAR RCS and G is transmitter gain which is set to 1 in this model.
     The propagation effect regardless of RCS is calculated as
     \[ k = \frac{P_r}{P_t \times \sigma} = \frac{G^2 \times \lambda^2}{(4\pi)^3 R^4} = \frac{G \times \lambda}{8\pi \times \sqrt{\pi} \times R^2} \]

   - target RCS
     Swerling models are used for target RCS.
     Swerling 0 also known as Swerling V is assumed as a constant RCS (nonfluctuating target).
     Swerling I and Swerling II can be described with a chi-square probability density function with two degrees of freedom as
     \[ f(x) = \frac{1}{\alpha^2} \exp\left(\frac{-x}{\alpha^2}\right) \quad x \geq 0 \]
     The amplitude of Swerling I target varies independently from scan to scan while
the amplitude of Swerling II targets varies independently from pulse to pulse. Swerling III and Swerling IV can be described with a chi-square probability density function with four degrees of freedom as
\[ f_d(x) = \frac{4x^{(n-1)/2}e^{-x}}{(2^{n/2}Γ(n/2))^2} \quad x \geq 0 \]

The amplitude of Swerling III target varies from scan to scan while the Swerling IV varies from pulse to pulse.

\* dopper shift and transmission delay

Let
\[ s(t) = s(t) = x(t) = y(t) = \frac{2 R(t)}{c} \]

where \( R(t) = R(t) = \frac{2 R(t)}{c} \)

\[ \text{doppler shift and transmission delay} \]

Let \( \omega \) is the distance between RADAR antenna and the target when \( t=0 \), \( v \) is the velocity of the target, then
\[ R(t) = R(t) = \frac{2 R(t)}{c} \]

then
\[ s(t) = s(t) = \frac{2 (R(t) - vt)}{c} e^{j\omega (t - \frac{2 (R(t) - vt)}{c})} \]

\[ = \frac{2 (R(t) - vt)}{c} e^{j\omega (t - \frac{2 (R(t) - vt)}{c})} \]

\[ = \frac{2 (R(t) - vt)}{c} e^{j\omega (t - \frac{2 (R(t) - vt)}{c})} \]

\[ = \frac{2 (R(t) - vt)}{c} e^{j\omega (t - \frac{2 (R(t) - vt)}{c})} \]

\[ = \frac{2 (R(t) - vt)}{c} e^{j\omega (t - \frac{2 (R(t) - vt)}{c})} \]

where \( \omega R \) is the dopper shift and \( 2R/c \) is transmission delay.

5. The input signal is firstly passed through an RF power amplifier to add attenuation, then multiplied with the target RCS after transforming from RF signal to baseband signal, then applied with propagation effect with Gain model. Then, the doppler shift is added by CxToEnv model. Lastly, transmission delay is added by DelayEnv model.

Reference

Radar Measurement

Contents

- RADAR Pd Measurement Part (radardesign)
- RADAR Pf Measurement Part (radardesign)
RADAR_Pd_Measurement Part

Categories: Measurement (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_Pd_Measurement</td>
<td>Detection Probability Estimation</td>
</tr>
</tbody>
</table>

RADAR_Pd_Measurement

Description: Detection Probability Estimation

Domain: Untimed

C++ Code Generation Support: NO

Associated Parts: RADAR Pd Measurement Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Data collection start index</td>
<td>0</td>
<td>Integer</td>
<td>NO</td>
<td></td>
<td>[0:∞)</td>
</tr>
<tr>
<td>PRI_NUM</td>
<td>number of samples in PRI</td>
<td>10000</td>
<td>Integer</td>
<td>NO</td>
<td></td>
<td>(0:∞)</td>
</tr>
<tr>
<td>FFT_Size</td>
<td>Number of Pulse for coherent detection</td>
<td>16</td>
<td>Integer</td>
<td>NO</td>
<td></td>
<td>(0:∞)</td>
</tr>
<tr>
<td>DetectionNum</td>
<td>number of detections</td>
<td>1</td>
<td>Integer</td>
<td>NO</td>
<td></td>
<td>(0:∞)</td>
</tr>
<tr>
<td>TargetsInPRI</td>
<td>number of targets in pri</td>
<td>1</td>
<td>Integer</td>
<td>NO</td>
<td></td>
<td>(0:∞)</td>
</tr>
<tr>
<td>TargetThreshold</td>
<td>TargetThreshold</td>
<td>1e-8</td>
<td>Float</td>
<td>NO</td>
<td></td>
<td>(0:∞)</td>
</tr>
<tr>
<td>ControlSimulation</td>
<td>let sink control how long the simulation will run?: NO, YES</td>
<td>YES</td>
<td>Enumeration</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Input Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>test</td>
<td>test stream</td>
<td>real</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This model is used to estimate the detection probability using the radar’s video output signal.
2. Each firing 1 token is consumed in the input.
3. Parameters detail:
   - Start is data collection start index
   - PRI_NUM is the number of samples in a PRI (Pulse Repetition Interval)
   - FFT_Size is the number of pulse in a CPI (Coherent Processing Interval)
   - DetectionNum is the number of detections
   - TargetsInPRI is the number of targets in a PRI return
   - TargetThreshold is the target detection threshold
4. The total simulation sample length is PRI_NUM*FFT_Size*DetectionNum.
5. The detection probability Pd is estimated based on Monte Carlo estimation as,

   \[
   \hat{p}_{pd,me} = \frac{1}{N} \sum_{i=1}^{N} D(t)
   \]

   and the indication function D(t) is defined on the detection event t as,

   \[
   D(t) = \begin{cases} 
   1 & \text{target detected with target input} \\
   0 & \text{no target detected with target input}
   \end{cases}
   \]

Reference
**RADAR_Pf_Measurement Part**

**Categories:** Measurement (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_Pf_Measurement</td>
<td>False Alarm Rate Estimation</td>
</tr>
</tbody>
</table>

**Description:** False Alarm Rate Estimation  
**Domain:** Untimed  
**C++ Code Generation Support:** NO  
**Associated Parts:** RADAR Pf Measurement Part (radardesign)

**Model Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Data collection start index</td>
<td>0</td>
<td>Integer</td>
<td>NO</td>
<td></td>
<td>(0:∞)</td>
</tr>
<tr>
<td>Stop</td>
<td>Data collection stop index when EstRelVariance is not met</td>
<td>1000</td>
<td>Integer</td>
<td>NO</td>
<td></td>
<td>(Start:∞)</td>
</tr>
<tr>
<td>ControlSimulation</td>
<td>Let sink control how long the simulation will run?: NO, YES</td>
<td>YES</td>
<td>Enumeration</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>test</td>
<td>test stream</td>
<td>real</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Notes/Equations**

1. This model is used to estimate the false alarm rate using radar’s video output signal.
2. Each firing, 1 token is consumed in the input pin.
3. Parameters detail
   - Start is data collection start index
   - Stop is data collection stop index
   - ControlSimulation is used to specify if use this model to control the DF simulation
4. The false alarm rate is estimated using Monte Carlo method as,

\[ \hat{\rho}_{fa, mc} = \frac{1}{N} \sum_{i=1}^{N} D(t) \]

and \( D(t) \) is the indication function defined on detection event \( t \) as,

\[ D(t) = \begin{cases} 1 & \text{target detected with no target input} \\ 0 & \text{no target detected with no target input} \end{cases} \]

**Reference**

**Radar Receiver**

**Contents**

- RADAR CICDecimate Part (radardesign)
- RADAR DDC Part (radardesign)
- RADAR Rx Part (radardesign)
RADAR_CICDecimate Part

Categories: Receiver (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_CICDecimate (radardesign)</td>
<td>RADAR CIC Decimation</td>
</tr>
</tbody>
</table>

**RADAR_CICDecimate**

Description: RADAR CIC Decimation  
Domain: Untimed  
C++ Code Generation Support: NO  
Associated Parts: RADAR_CICDecimate Part (radardesign)

**Model Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>the concatenation order of the CIC filter</td>
<td>5</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Ratio</td>
<td>the Decimation ratio applied to the input signal</td>
<td>2</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>DiffDelay</td>
<td>Differential delay. Changes both the shape and number of nulls in the filter response. Also affects the null locations. In practice, it is usually held to one or two</td>
<td>1</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Phase</td>
<td>downsampling phase</td>
<td>0</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

**Input Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>input</td>
<td>complex signal to be decimated</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Output Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>output</td>
<td>decimated complex signal</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Notes/Equations**

1. This model implements a multiple order cascaded integrator comb filter with decimation.
2. Each firing,
   - Ratio tokens are consumed at the input port. It is equal to 2 by default.
   - 1 token is produced at the output port.
3. Parameter details:
   - Order specifies the concatenation order of CIC filter.
   - Ratio specifies the decimation ratio applied to the input signal.
   - DiffDelay specifies the differential delay for basic comb part as specified in DiffDelay (radardesign).
   - The Phase parameter specifies which one sample (out of the Ratio input samples read) to output: if Phase = 0, the first input sample is output; if Phase = Factor - 1, the latest input sample is output. The equation describing the behavior of this model is \( y[n] = x[\text{Ratio} \times n + \text{Phase}] \), where \( n \) is the output sample number, \( y \) is the output, and \( x \) is the input.
4. The CIC filter is a flexible, multiplier-free filter suitable for hardware implementation, that can also handle arbitrary and large rate changes as specified in CICFilterBasic (radardesign).
5. The N stage CIC decimation is implemented as N cascaded integrator stages running at \( fs \), followed by a specified decimator, followed by N cascaded comb stages running
at fs/R. The three stage decimation CIC Filter is shown in the following figure.

three stage decimation CIC Filter schematic

References

RADAR_DDC Part

RADAR Digital Down Converter

Categories: Receiver (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

Model
RADAR_DDC
(radardesign)

RADAR_DDC

Description: RADAR Digital Down Converter
Associated Parts: RADAR DDC Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF_Freq</td>
<td>IF carrier frequency ((0:inf))</td>
<td>25000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>IF_SamplingRate</td>
<td>Digital IF signal sampling rate ((0:inf))</td>
<td>100000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>ADC_NBits</td>
<td>number of bits for ADC ((2:inf))</td>
<td>8</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
<tr>
<td>PhaseImbalance</td>
<td>phase imbalance in degrees, Q channel relative to I channel ((-inf:inf))</td>
<td>0</td>
<td>deg</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BB_DownSamplingRatio</td>
<td>the downsampling ratio from digital IF to baseband ((1:inf))</td>
<td>20</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
<tr>
<td>RC_ExcessBW</td>
<td>Excess bandwidth of raised cosine filter ((0.0:1.0))</td>
<td>0.22</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>Out_CenterFreq</td>
<td>The center frequency of the output signal ((0:inf))</td>
<td>0</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
</tbody>
</table>

Input Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IF_Signal</td>
<td>envelope</td>
<td>NO</td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>BB_Signal</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This subnetwork implements digital downconverter.
2. The RADAR_DDC schematic is shown below:
RADAR_DDC schematic

3. Parameter details:
- IF_Freq specifies the IF modulation frequency.
- IF_SamplingRate is the sampling rate of digital IF which is also the input sampling rate.
- ADC_NBits specifies the number of bits for analog to digital converter which converts the analog signal to digital signal.
- PhaseImbalance specifies phase imbalance in degree which is used to add certain impairments to the ideal output signal. The IF is given by
  \[ V_{IF}(t) = V_I(t) \times \cos(\omega_c t) - V_Q(t) \times \sin(\omega_c t + \frac{\phi \pi}{180}) \]
  \( \phi \) is the phase imbalance.
- BB_DownSamplingRatio specifies the downsampling ratio from IF to baseband signal.
- RC_ExcessBW specifies excess bandwidth of raised cosine filter which performs filtering after baseband signal downsampling.
- Out_CenterFreq specifies the center frequency of output signal.

4. The following figure provides a block diagram of a DDC that translates an IF signal down to complex baseband signal.

Digital downconversion architecture

In digital downconversion, the analog IF signal is first sampled by an ADC, then multiplied by the sine and cosine signals from the NCO. Lowpass filter is followed the multiplications to prevent aliasing. Decimation is applied to produce complex output samples.

- The Nyquist rate is a minimum but not sufficient sampling frequency for a bandlimited bandpass signal. The IF sampling rate should be chosen carefully to ensure that aliasing does not occur.
- The sampling rate of output baseband signal should be no smaller than Nyquist rate to avoid aliasing.

See: RADAR_DUC (radardesign)

References
**RADAR_Rx Part**

**RADAR Receiver Front End**

**Categories:** Receiver (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

### Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>TStep</td>
<td>simulation time step; TStep=0 results in use of externally set TStep ([0:inf])</td>
<td>0</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>RF_Freq</td>
<td>RF carrier frequency ([0:inf])</td>
<td>1000000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>RF_Gain</td>
<td>Complex voltage gain; with form re+j*im; to specify gain in dB use dbpolar(DB, degree) ([inf:inf])</td>
<td>1</td>
<td>none</td>
<td>Complex number</td>
<td>NO</td>
</tr>
<tr>
<td>IF_Freq</td>
<td>IF carrier frequency ([0:inf])</td>
<td>25000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>IF_Gain</td>
<td>Complex voltage gain; with form re+j*im; to specify gain in dB use dbpolar(DB, degree) ([inf:inf])</td>
<td>1</td>
<td>none</td>
<td>Complex number</td>
<td>NO</td>
</tr>
<tr>
<td>IF_SamplingRate</td>
<td>Digital IF signal sampling rate ([0:inf])</td>
<td>100000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BandWidth</td>
<td>passband bandwidth of IF filter ([0:inf])</td>
<td>5000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>ADC_NBits</td>
<td>number of bits for DAC ([2:inf])</td>
<td>8</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
<tr>
<td>PhaseImbalance</td>
<td>phase imbalance in degrees, Q channel relative to I channel ([inf:inf])</td>
<td>0</td>
<td>deg</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BB_DownSamplingRatio</td>
<td>the downsampling ratio from digital IF to baseband ([1:inf])</td>
<td>20</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
<tr>
<td>RC_ExcessBW</td>
<td>Excess bandwidth of raised cosine filter ([0.0:1.0])</td>
<td>0.22</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>Out_CenterFreq</td>
<td>The center frequency of the output signal ([0:inf])</td>
<td>0</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>NoiseFigure_RFGain</td>
<td>input noise figure for gain in RF, in dB ([0:inf])</td>
<td>0</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>NoiseFigure_IFGain</td>
<td>input noise figure for gain in IF, in dB ([0:inf])</td>
<td>0</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>NoiseFigure_Mixer</td>
<td>input noise figure for MixerRF, in dB ([0:inf])</td>
<td>0</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>GCTYPE_RFGain</td>
<td>Gain compression type: none, TOI, dBc1, TOI+dBc1, Psat+GCSat+TOI, Psat+GCSat+TOI</td>
<td>none</td>
<td>Enumeration</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>
**SystemVue - RADAR Baseband Verification Library**

<table>
<thead>
<tr>
<th>Parameter Details</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOIout_RFGain</td>
<td>Third order intercept power $((-\infty: \infty))$</td>
</tr>
<tr>
<td>dBc1out_RFGain</td>
<td>1 dB gain compression power $((-\infty: \infty))$</td>
</tr>
<tr>
<td>PSat_RFGain</td>
<td>Saturation power $((-\infty: \infty))$</td>
</tr>
<tr>
<td>GCSat_RFGain</td>
<td>Gain compression at saturation; dB $([3:7])$</td>
</tr>
<tr>
<td>GComp_RFGain</td>
<td>Array of triple values for large signal gain change vs signal power. Input Power in dBm, Gain change from small signal in dB, and Phase change from small signal in degree $[0,0,0]$</td>
</tr>
<tr>
<td>GCType_IFGain</td>
<td>Gain compression type: none, TOI, dBC1, TOI+dBC1, PSat+GCSat+TOI, PSat+GCSat+dBC1, PSat+GCSat+TOI+dBC1, RappNonlinearity, Gain compression vs input power, AM/AM and AM/PM vs input power</td>
</tr>
<tr>
<td>TOIout_IFGain</td>
<td>Third order intercept power $((-\infty: \infty))$</td>
</tr>
<tr>
<td>dBc1out_IFGain</td>
<td>1 dB gain compression power $((-\infty: \infty))$</td>
</tr>
<tr>
<td>PSat_IFGain</td>
<td>Saturation power $((-\infty: \infty))$</td>
</tr>
<tr>
<td>GCSat_IFGain</td>
<td>Gain compression at saturation; dB $([3:7])$</td>
</tr>
<tr>
<td>GComp_IFGain</td>
<td>Array of triple values for large signal gain change vs signal power. Input Power in dBm, Gain change from small signal in dB, and Phase change from small signal in degree $[0,0,0]$</td>
</tr>
</tbody>
</table>

**Input Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RF_Signal</td>
<td>envelope</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Output Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>BB_Signal</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Notes/Equations**

1. This subnetwork implements receiver front end for RADAR system.
2. The RADAR_Rx schematic is shown below:

![RADAR_Rx schematic](image)

3. Parameter details:
- TStep specifies the simulation time step which is the reciprocal of IF_SamplingRate.
- RF_Freq specifies the RF carrier frequency in Hz.
- RF_Gain specifies the gain of RF power amplifier in dbpolar.
- IF_Freq specifies the digital downconverter carrier frequency in Hz.
- IF_Gain specifies the gain of IF power amplifier in dbpolar.
- IF_SamplingRate specifies the sampling rate of digital downconverter which is also the sampling rate of input signal.
- Bandwidth specifies the bandwidth of IF bandpass filter which should be no smaller than the bandwidth of baseband signal.
- ADC_NBits specifies the number of bits for ADC which converts the analogy signal to digital signal.
- PhaseImbalance specifies phase imbalance in degree which is used to add certain impairments to the ideal IF signal as PhaseImbalance (radardesign).
- BB_DownSamplingRatio specifies the downsampling ratio from IF to baseband signal.
- RC_ExcessBW specifies excess bandwidth of raised cosine filter which performs filtering after baseband signal downsampling.
- Out_CenterFreq specifies the center frequency of output signal.
- NoiseFigure_RFGain specifies the input noise figure for power amplifier in RF in dB.
- NoiseFigure_IFGain specifies the input noise figure for power amplifier in IF in dB.
- NoiseFigure_Mixer specifies the input noise figure for RF Mixer in dB.
- GCTYPE_RF_Gain specifies the compression type for power amplifier in RF.
- TOIout_RFGain specifies the third order intercept power for power amplifier in RF.
- dBc1out_RFGain specifies the 1 dB gain compression power for power amplifier in RF.
- PSat_RFGain specifies the saturation power for power amplifier in RF.
- GCSat_RFGain specifies the compression at saturation for power amplifier in RF in dB.
- GComp_RFGain is an array parameter which specifies the triple values for large signal gain change vs signal power in RF.
- GCTYPE_IF_Gain specifies the compression type for power amplifier in IF.
- TOIout_IFGain specifies the third order intercept power for power amplifier in IF.
- dBc1out_IFGain specifies the 1 dB gain compression power for power amplifier in IF.
- PSat_IFGain specifies the saturation power for power amplifier in IF.
- GCSat_IFGain specifies the compression at saturation for power amplifier in IF in dB.
- GComp_IFGain is an array parameter which specifies the triple values for large signal gain change vs signal power in IF.

4. The following figure provides a block diagram of a digital receiver front end that translates an RF signal down to a complex baseband signal.

![Digital receiver front end architecture](image)

**Digital receiver front end architecture**

An analogy envelop RF waveform is first passed through an RF power amplifier then mixed with an RF (complex envelope) to get the IF signal, followed by a passband filter then a power amplifier, and passed through a digital downconverter to generate the baseband signal.

- The Nyquist rate is a minimum but not sufficient sampling frequency for a bandlimited bandpass signal. The IF sampling rate should be chosen carefully to ensure that aliasing does not occur.
- The sampling rate of output baseband signal should be no smaller than Nyquist rate to avoid aliasing.
SystemVue - RADAR Baseband Verification Library

See:
RADAR_DDC (radardesign)
RADAR_Tx (radardesign)

Reference

Radar Signal Processing

Contents

- RADAR CFAR Part (radardesign)
- RADAR Detector Part (radardesign)
- RADAR MTD Part (radardesign)
- RADAR MTI Part (radardesign)
- RADAR PC Part (radardesign)
- RADAR PD Part (radardesign)
RADAR_C FAR Part

Categories: Signal Processing (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_CFAR (radardesign)</td>
<td>Constant False Alarm Rate</td>
</tr>
</tbody>
</table>

RADAR_C FAR

Description: Constant False Alarm Rate
Domain: Untimed
C++ Code Generation Support: NO
Associated Parts: RADAR CFAR Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFAR_Type</td>
<td>the type of CFAR: CA, SOCA, GOCA, Clutter Map</td>
<td>CA</td>
<td>Enumeration</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>CellSize</td>
<td>Cell Size</td>
<td>100</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>ReferenceCell</td>
<td>Reference Cell Size</td>
<td>32</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>GuardCell</td>
<td>guard cell number</td>
<td>4</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Detector_Type</td>
<td>type of the detector: Evelop, Square, LogSquare</td>
<td>Square</td>
<td>Enumeration</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Threshold</td>
<td>the type of CFAR: Pf, Alpha</td>
<td>Pf</td>
<td>Enumeration</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Pf</td>
<td>Expected False Alarm Rate</td>
<td>1e-4</td>
<td>Float</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Alpha</td>
<td>Threshold Accumulation Factor</td>
<td>1.0</td>
<td>Float</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Beta</td>
<td>adaption coefficient for clutter map</td>
<td>1.0</td>
<td>Float</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

Input Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>input</td>
<td>real</td>
<td>NO</td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>output</td>
<td>real</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This model is used for Constant False Alarm Rate(CFAR) detection.
2. Each firing CellSize number of inputs are consumed and the same amount of outputs are generated.
3. Parameter detail,
   - CFAR_Type is used to specify which type of CFAR detector is used.
   - CellSize is used to specify the number of the cells
   - ReferenceCell is used to specify the number of reference cells in each side
   - GuardCell is used to specify the number of guard cells in each side
   - Detector_Type is used to specify the video detector's type associated with CFAR detector, refer to RADAR_Detector (radardesign)
   - Threshold is used to specify the CFAR detection threshold using false alarm rate derived one or direct given one. Notice that for log square detector, direct given threshold is used.
   - Pf is the expected false alarm rate when Threshold is chosen as Pf
SystemVue - RADAR Baseband Verification Library

- Alpha is the threshold value when Threshold is chosen as Alpha
- Beta is the adaptation coefficient for clutter map

4. The model can support both Cell Average (CA) based and clutter map based CFAR operation.

5. For CA CFAR, the cell's structure is as below:

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- is the test cell for target detection
- is the guard cell not to be included for threshold estimation
- is the reference cell used for threshold estimation

With this structure, the CA algorithm can be described as,

\[
x_0 \geq \frac{T}{z/M}
\]

where \(x_0\) is the value of test cell, \(z\) is the summation of the value of all reference cells, and \(M\) is the size of the reference cell. \(T\) is the detection threshold.

Let \(z_l\) is the sum of the left reference cell and \(z_r\) is the sum of right reference cell.

- For basic CA, \(z = z_l + z_r\)
- For GOCA (Greater Of Cell Average), \(z = \max(z_l, z_r)\)
- For SOCA (Smaller Of Cell Average), \(z = \min(z_l, z_r)\)

6. Clutter mapping is a technique for detecting large targets in the zero Doppler bin which would otherwise be discarded.

“Map” of recent zero-Doppler returns for each resolution cell is kept and used to compute threshold for zero bin only.

Map is updated continuously to allow for weather,

\[
c_n(m) = \beta x_n(m) + (1 - \beta)c_n(m - 1)
\]

- \(c_n(m)\) is the estimated clutter in nth cell for the mth return
- \(x_n(m)\) is the measured clutter in nth cell from the mth return
- \(\beta\) is the adaptation factor

Reference

RADAR_Detector Part

**Categories:** Signal Processing (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR Detector (radardesign)</td>
<td>RADAR Detector</td>
</tr>
</tbody>
</table>

**RADAR_Detector**

**Description:** RADAR Detector  
**Domain:** Untimed  
**C++ Code Generation Support:** NO  
**Associated Parts:** RADAR Detector Part (radardesign)

**Model Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>type of the detector: Envelop, Square, LogSquare</td>
<td>Square</td>
<td></td>
<td>Enumeration</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Input Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>input</td>
<td>input signal</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Output Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>output</td>
<td>output signal</td>
<td>real</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Notes/Equations**

1. This model is used for video signal detection.  
2. Each firing one token is consumed in input and one token is consumed in output.  
3. The detection algorithm supported as,

\[
y(n) = \begin{cases} 
|x(n)|, & \text{mean law} \\
|x(n)|^2, & \text{square law} \\
\ln|x(n)|^2, & \text{log square law} 
\end{cases}
\]

where \(x\) is the nth input and \(y\) is the nth output.

**Reference**

**RADAR_MTD Part**

**Categories:** Signal Processing (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_MTD</td>
<td>Moving Target Detection</td>
</tr>
</tbody>
</table>

**Description:** Moving Target Detection

**Domain:** Untimed

**C++ Code Generation Support:** NO

**Associated Parts:** RADAR MTD Part (radardesign)

**Model Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRI</td>
<td>Pulse Repeat Interval</td>
<td>1e-4</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>NumOfPulse</td>
<td>Number of pulses</td>
<td>8</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Freq_Weight</td>
<td>frequency domain weight</td>
<td>[1,1,1,1,1,1,1,1]</td>
<td>Floating point</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>WindowType</td>
<td>the type of window: Rectangle, Bartlett, Hanning, Hamming, Blackman, SteepBlackman, Kaiser</td>
<td>Rectangle</td>
<td>Enumeration</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>WindowParameters</td>
<td>array of values for the window</td>
<td>0</td>
<td>Floating point</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>BB_SamplingRate</td>
<td>Waveform Baseband Sampling Rate</td>
<td>10e6</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Input Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>input</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Output Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>output</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Notes/Equations**

1. This model is used for moving target detection (MTD).
2. Each firing, NumOfPulse*PRI*BB_SamplingRate number of tokens are consumed in input and the same amount of tokens are generated in output.
3. Parameters detail
   - PRI is used to specify the cycle of the pulse
   - NumOfPulse is used to specify the number of pulses in a CPI (Coherent Processing Interval)
   - Freq_Weight is an array parameter with the size as the NumOfPulse used to specify the weighting for MTD filter bank
   - WindowType is used to specify the type of window used in time domain
   - WindowParameters is an array parameter for time domain windowing
   - BB_SamplingRate is used to specify the baseband sampling rate
4. MTD is used a 2D data matrix for Doppler frequency domain analysis as below,
4. The implementation of MTD is based on FFT over same range gate data of different PRI. Frequency domain weighting is provided for user to define their own Doppler domain shaping and also user can specify different time domain windowing functions to support time windowing to the data.

Reference

RADAR_MTI Part

Categories: Signal Processing (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_MTI</td>
<td>Moving Target Indication</td>
</tr>
</tbody>
</table>

RADAR_MTI

Description: Moving Target Indication

Domain: Untimed

C++ Code Generation Support: NO

Associated Parts: RADAR MTI Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRI</td>
<td>Pulse Repeat Interval</td>
<td>1e-4</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>NumOfPulse</td>
<td>Number of pulses in MTI</td>
<td>32</td>
<td></td>
<td>Integer</td>
<td>NO</td>
</tr>
<tr>
<td>MTI_Type</td>
<td>The Type of Moving Target Indicator: Two Pulse Canceller, Three Pulse Canceller</td>
<td>Two Pulse Canceller</td>
<td>Enumeration</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>BB_SamplingRate</td>
<td>Waveform Baseband Sampling Rate</td>
<td>10e6</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
</tbody>
</table>

Input Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>input</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>output</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This model is used to perform pulse canceler based Moving Target Indicator (MTI).
2. Each firing, NumOfPulses*PRI*BB_SamplingRate inputs are consumed and (NumOfPulses-n)*PRI*BB_SamplingRate outputs are generated, where n is the order of pulse canceler used for MTI.
3. Parameters details,
   - PRI, specifies the pulse repetition interval,
   - NumOfPulses, specifies the number of pulses in a CPI (Coherent Processing Interval),
   - Type, specifies 2-pulses canceler or 3-pulses canceler is used,
   - BB_SamplingRate, specifies the baseband sampling rate.
4. This model can support both 2-pulses canceler and 3-pulses canceler as,
   - 2-pulse canceler,
     \[ H(z) = 1 - z^{-1} \]
   - 3-pulse canceler,
     \[ H(z) = 1 - 2z^{-1} + z^{-2} \]
   where the delay of the filter is in PRI interval.

Reference
RADAR_PC Part

Pulse Compression

Categories: Signal Processing (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRI</td>
<td>pulse repeat interval ((0:inf))</td>
<td>0</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BandWidth</td>
<td>signal bandwidth ((0:inf))</td>
<td>5000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>FM_Offset</td>
<td>Frequency Modulation Offset ((0:inf))</td>
<td>-1</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BB_SamplingRate</td>
<td>Waveform Baseband Sampling Rate ((0:inf))</td>
<td>1000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>WindowType</td>
<td>the type of window: Rectangle, Bartlett, Hanning, Hamming, Blackman, SteepBlackman, Kaiser</td>
<td>Hanning</td>
<td>none</td>
<td>Enumeration</td>
<td>NO</td>
</tr>
</tbody>
</table>

Input Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SigIn</td>
<td>data in</td>
<td>complex</td>
<td>NO</td>
</tr>
<tr>
<td>2</td>
<td>RefIn</td>
<td>reference in</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>SigOut</td>
<td>data out</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This model is used for pulse compression(PC) in pulse radar.
2. Each firing PRI*BB_SamplingRate number of inputs are consumed and the same amount of outputs are generated.
3. Parameters detail
   - PRI, is used to specify the cycle of pulse
   - Bandwidth, is used to specify pulse bandwidth
   - FM_Offset, is used to specify frequency domain offset
   - BB_SamplingFreq, is the baseband sampling rate for waveform generation.
   - WindowType is used to specify the type of window to control compressed pulse sidelobe.
4. The model is used a FFT based implementation as below,
4. Application example,
The input and output of PC can be seen as below,

Reference

RADAR_PD Part

Pulse Doppler

Categories: Signal Processing (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

Model

RADAR_PD
(radardesign)

RADAR_PD

Description: Pulse Doppler

Associated Parts: RADAR PD Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRI</td>
<td>Pulse Repeat Interval ((0:inf))</td>
<td>0</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>NumOfPulse</td>
<td>Number of pulses ([4:inf])</td>
<td>8</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
<tr>
<td>MTI_Type</td>
<td>The Type of Moving Target Indicator: Bypass MTI, Two Pulse Canceller, Three Pulse Canceller</td>
<td>Two Pulse Canceller</td>
<td>none</td>
<td>Enumeration</td>
<td>NO</td>
</tr>
<tr>
<td>MTD_Freq_Weight</td>
<td>frequency domain weight</td>
<td>[1,1,1,1,1,1,1,1]</td>
<td>none</td>
<td>Floating point</td>
<td>NO</td>
</tr>
<tr>
<td>MTD_WindowType</td>
<td>the type of window: Rectangle, Bartlett, Hanning, Hamming, Blackman, SteepBlackman, Kaiser</td>
<td>Rectangle</td>
<td>none</td>
<td>Enumeration</td>
<td>NO</td>
</tr>
<tr>
<td>MTD_WindowParameters</td>
<td>array of values for the window</td>
<td>0</td>
<td>none</td>
<td>Floating point</td>
<td>NO</td>
</tr>
<tr>
<td>BB_SamplingRate</td>
<td>Waveform Baseband Sampling Rate ((0:inf))</td>
<td>10000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
</tbody>
</table>

Input Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>input</td>
<td>data before FEC</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>mtd_output</td>
<td>data after FEC</td>
<td>complex</td>
<td>NO</td>
</tr>
<tr>
<td>3</td>
<td>mti_output</td>
<td>data after FEC</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This model is used for pulse-doppler(PD) processing
2. Parameters detail see
   RADAR_MTI (radardesign)
   RADAR_MTD (radardesign)
3. If MTI is enabled, each firing (N+M-1)*PRI*BB_SamplingRate number of tokens are consumed in the input and N*PRIi*BB_SamplingRate number of tokens will be produced in the output, where N is the number of pulses used for MTD and M is the the order of pulse canceler. If MTI is disabled, each firing, N*PRI*BB_SamplingRate number of tokens will be consumed in the input and the same amount of tokens will

45
be produced in the output.
4. The PD's structure is shown below,

Also see
RADAR_MTI (radardesign)
RADAR_MTD (radardesign)

Reference
Radar Signal Source

Contents

- RADAR BarkerCode Part (radardesign)
- RADAR FrankCode Part (radardesign)
- RADAR LFM Part (radardesign)
- RADAR MatchedSrc Part (radardesign)
- RADAR NLFM Part (radardesign)
- RADAR ZCCode Part (radardesign)
**RADAR_NLFM Part**

**Categories:** Signal Source (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_NLFM</td>
<td>Non-Linear Frequency Modulation Waveform Generator</td>
</tr>
</tbody>
</table>

**RADAR_NLFM**

**Description:** Non-Linear Frequency Modulation Waveform Generator

**Domain:** Untimed

**C++ Code Generation Support:** NO

**Associated Parts:** RADAR NLFM Part (radardesign)

**Model Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulsewidth</td>
<td>Pulse Width</td>
<td>1e-5</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>PRI</td>
<td>Pulse Repeat Interval</td>
<td>1e-4</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>Waveform Bandwidth</td>
<td>5e6</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BB_SamplingRate</td>
<td>Waveform Baseband Sampling Rate</td>
<td>10e6</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>NLF_Type</td>
<td>Nonlinear Function Type: Hamming, Cos4, Gauss, Polynomial</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polynomial_Coef</td>
<td>Polynomial Coeficient to generate nonlinear frequency modulation</td>
<td>[0.426]</td>
<td>Floating point array</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

**Output Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>output</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Notes/Equations**

1. This model is used to generate nonlinear frequency modulated signal.
2. Each firing, 1 token is generated in the output.
3. Parameters detail
   - Pulsewidth, is the width of the pulse.
   - PRI, is the cycles for pulses.
   - Bandwidth, is the pulse bandwidth.
   - BB_SamplingFreq, is the baseband sampling rate for waveform generation.
   - NLFM_Type, is the type of nonlinear frequency modulation signal
   - Polynomial_Coef, is the polynomial coefficient used to generate nonlinear frequency modulation when NLFM_Type is specified as polynomial.
4. The nonlinear frequency modulated signal is generated by interpolation of the linear frequency modulation signal using various kernel function such as Gauss function.
5. Application example
   A non-linear FM waveform with PRI=30us, Pulsewidth=10us, Bandwidth=5MHz, BB_SamplingRate=10MHz using Gaussian interpolation as below,
Baseband I and Q waveform

Bandpass Spectrum

Reference
RADAR_ZCCode Part

Categories: Signal Source (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_ZCCode</td>
<td>Zadoff-Chu Coded Waveform Generator</td>
</tr>
</tbody>
</table>

**RADAR_ZCCode**

Description: Zadoff-Chu Coded Waveform Generator
Domain: Untimed
C++ Code Generation Support: NO
Associated Parts: RADAR_ZCCode Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRI</td>
<td>Pulse Repetition Interval</td>
<td>1e-5</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>SubPulseWidth</td>
<td>SubPulse Width</td>
<td>1e-6</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>SubPulseNum</td>
<td>SubPulse Number</td>
<td>4</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>u</td>
<td>class of the GCL sequence</td>
<td>1</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>BB_SamplingRate</td>
<td>Waveform Baseband Sampling Rate</td>
<td>10e6</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>output</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This model is used to generate pulsed radar waveform with Zadoff-Chu code, a kind of General Chirp Like (GCL) sequence. In one pulse repetition interval, a specified number of subpulse will be generated then followed with zeros.
2. Each firing, PRIxBB_SamplingRate tokens are produced at the output port. It is equal to 100 by default.
3. Parameter details:
   - PRI specifies the pulse repetition interval.
   - SubPulseWidth specifies the cycle of one subpulse.
   - SubPulseNum specifies the number of subpulses in one pulse cycle.
   - u specifies the Zadoff-Chu root sequence index.
   - BB_SamplingRate specifies the baseband sampling rate for waveform generation.
4. The Zadoff-Chu sequence is generated by:
   \[ x_u(n) = e^{-j \pi n (n+1)/N_{zc}} \]
   where
   \[ 0 \leq n \leq N_{zc} - 1, \]
   \[ N_{zc} \text{ is the length of sequence} \]
   \[ x_u(n) \text{ is the initial phase for the n-th sub pulse in one PRI. The length of Zadoff-Chu sequence is prime. The waveform for each PRI are the same.} \]
5. Application example,
   The following example shows the waveform in 2 PRI with parameters as,
   PRI=1e-5, SubPulseWidth=1e-6, SubPulseNum=4, u=1, BB_SamplingRate=20e6.
Reference

RADAR_LFM Part

Categories: Signal Source (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_LFM</td>
<td>Linear Frequency Modulation Waveform Generator</td>
</tr>
</tbody>
</table>

RADAR_LFM

Description: Linear Frequency Modulation Waveform Generator
Domain: Untimed
C++ Code Generation Support: NO
Associated Parts: RADAR LFM Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulsewidth</td>
<td>Pulse Width</td>
<td>[1e-5] s</td>
<td>s</td>
<td>Floating point array</td>
<td>NO</td>
</tr>
<tr>
<td>PRI</td>
<td>Pulse Repeat Interval</td>
<td>[1e-4] s</td>
<td>s</td>
<td>Floating point array</td>
<td>NO</td>
</tr>
<tr>
<td>PRI_Combination</td>
<td>PRI Combination reprents by each PRI repeat number</td>
<td>[1]</td>
<td>Integer array</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Bandwidth</td>
<td>Waveform Bandwidth</td>
<td>[5e6] Hz</td>
<td>Hz</td>
<td>Floating point array</td>
<td>NO</td>
</tr>
<tr>
<td>FM_Offset</td>
<td>Frequency Modulation Offset</td>
<td>[0]</td>
<td>Hz</td>
<td>Floating point array</td>
<td>NO</td>
</tr>
<tr>
<td>BB_SamplingRate</td>
<td>Waveform Baseband Sampling Rate</td>
<td>10e6 Hz</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>output</td>
<td>output signal</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This model is used to generate Linear Frequency Modulation waveform.
2. Each firing, PRI*BB_SamplingRate tokens are generated in the output.
3. Parameters details,
   - Pulsewidth, is an array parameter to specify may-different widths of the pulse in different staggering PRI group.
   - PRI, is an array parameter to specify the cycles for multiple pulses to support PRI staggering.
   - PRI_Combination, is an array parameter to specify pulse numbers for each staggering PRI group.
   - Bandwidth, is an array parameter to specify different pulse bandwidth for each staggering PRI group.
   - FM_Offset, is an array parameter to specify different frequency domain offset for each staggering PRI group.
   - BB_SamplingFreq, is the baseband sampling rate for waveform generation.
4. LFM waveform is generated by following equation,
   \[ s(t) = \begin{cases} 
   e^{j2\pi \beta(t-\tau)^2/\tau} & 0 \leq t \leq \tau \\
   0 & \tau \leq t \leq PRI 
   \end{cases} \]
   where, \( \beta \) is the pulse bandwidth, \( \tau \) is the pulse width/duration, PRI is the pulse repetition interval.
5. Application example,
The first example shows a simple FM pulse train without PRI staggering and with parameters as,
Pulsewidth=10us, PRI=30us, PRI_Combination=1, Bandwidth=5MHz, FM_Offset=0, BB_SamplingRate=10MHz.
The waveform and the spectrum is as below,

I and Q Waveform

Baseband Spectrum
The second example shows a staggering PRI FM pulse with parameters as,
Pulsewidth=10us, 15us, 20us, PRI=30us, 40us, 50us, PRI_Combination=1, 1, 1, Bandwidth=5MHz, 4MHz, 3MHz, FM_Offset=0, BB_SamplingRate=10MHz.
Staggering PRI Waveform

Staggering PRI Baseband Spectrum

References

**RADAR_BarkerCode Part**

**Categories:** Signal Source (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_BarkerCode (radardesign)</td>
<td>Barker Coded Waveform Generator</td>
</tr>
</tbody>
</table>

**RADAR_BarkerCode**

**Description:** Barker Coded Waveform Generator

**Domain:** Untimed

**C++ Code Generation Support:** NO

**Associated Parts:** RADAR BarkerCode Part (radardesign)

**Model Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRI</td>
<td>Pulse Repetition Interval</td>
<td>1e-5</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>SubPulseWidth</td>
<td>SubPulse Width</td>
<td>1e-6</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>CodeLength</td>
<td>Number of Phases: Length_2_a, Length_2_b, Length_3, Length_4_a, Length_4_b, Length_5, Length_7, Length_11, Length_13</td>
<td>Length_2_a</td>
<td>Enumeration</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>BB_SamplingRate</td>
<td>Waveform Baseband Sampling Rate</td>
<td>10e6</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Output Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>output</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Notes/Equations**

1. This model is used to generate binary phase coded waveform by Barker code.
2. Each firing, PRI*BB_SamplingRate number of tokens are generated in output.
3. Parameters detail
   - PRI is used to specify the pulse repetition interval.
   - SubPulseWidth is used to specify the width of each sub-pulse, where we use sine wave as the sub-pulse.
   - CodeLength is used to specify the length of Barker code and the final Barker code is constructed by CodeLength number of sub-pulses with a pulse width of CodeLength*SubPulseWidth.
   - BB_SamplingRate is used to specify the baseband sampling rate.
4. The Barker code is defined,

<table>
<thead>
<tr>
<th>Code Length</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>2a</td>
<td>11</td>
</tr>
<tr>
<td>2b</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>110</td>
</tr>
<tr>
<td>4a</td>
<td>1110</td>
</tr>
<tr>
<td>4b</td>
<td>1101</td>
</tr>
<tr>
<td>5</td>
<td>11101</td>
</tr>
<tr>
<td>7</td>
<td>1110010</td>
</tr>
<tr>
<td>11</td>
<td>1110010010</td>
</tr>
<tr>
<td>13</td>
<td>1111100110101</td>
</tr>
</tbody>
</table>
5. **Application example**

The example shows a Barker pulse train with parameters as, SubPulsewidth=1us, PRI=30us, CodeLength=5, BB_SamplingRate=10MHz. The waveform is as below,

![Baseband I and Q waveform](image)

**References**

**RADAR_FrankCode Part**

**Categories:** Signal Source (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_FrankCode</td>
<td>Frank Coded Waveform Generator</td>
</tr>
</tbody>
</table>

**RADAR_FrankCode**

**Description:** Frank Coded Waveform Generator

**Domain:** Untimed

**C++ Code Generation Support:** NO

**Associated Parts:** RADAR FrankCode Part (radardesign)

**Model Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Frank and P code type: Frank, P1, P3, P4, Px</td>
<td>Frank</td>
<td></td>
<td>Enumeration</td>
<td>NO</td>
</tr>
<tr>
<td>PRI</td>
<td>Pulse Repetition Interval</td>
<td>1e-5</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>SubPulseWidth</td>
<td>SubPulse Width</td>
<td>1e-6</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>L</td>
<td></td>
<td>2</td>
<td></td>
<td>Integer</td>
<td>NO</td>
</tr>
<tr>
<td>BandWidth</td>
<td>Pulse BandWidth</td>
<td>5e6</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BB_SamplingRate</td>
<td>Waveform Baseband Sampling Rate</td>
<td>10e6</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Output Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>output</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Notes/Equations**

1. This model is used to generate Frank code and P code based phase coded waveform.
2. Each firing, PRI*BB_SamplingRate of tokens are generated in output.
3. Parameters detail,
   - Type is used to specify which type of code will be used as Frank, P1, P3, P4 or Px
   - PRI is used to specify the pulse repetition interval.
   - SubPulseWidth is used to specify the width of each sub-pulse, where we use sine wave as the sub-pulse.
   - CodeLength is used to specify the length of Barker code and the final Barker code is constructed by CodeLength number of sub-pulses with a pulse width of CodeLength*SubPulseWidth.
   - BB_SamplingRate is used to specify the baseband sampling rate.
4. The Frank code, P1 code and Px code is defined by a square matrix of length of M=L^2 as
   \[ s_{n+k} = \exp(j\phi_{nk}) \]
   where 0<n<=L and 0<k<=L
   Frank code,
   \[ \phi_{nk} = 2\pi(n-1)(k-1)/L \]
   P1 code,
   \[ \phi_{nk} = 2\pi((L+1)/2-n)((n-1)L + (k-1))/L \]
   Px code,
Unlike Frank code is defined for a length of square matrix, P3 and P4 code are defined for any length M,

**P3 code,**

\[ \phi_{nk} = \frac{2\pi((L + 1)/2 - k)/(L + 1)}{L}, \quad L \text{ is even} \]
\[ \frac{2\pi((L + 1)/2 - k)/(L/2)}{L}, \quad L \text{ is odd} \]

**P4 code,**

\[ \phi_{m} = \begin{cases} \frac{\pi(m - 1)^2}{M} & M \text{ is odd} \\ \pi m(m - 1)/M & M \text{ is even} \end{cases} \]

where \(0 < m \leq M\)

5. **Application example,**

A Frank code is generated with PRI=50us, SubPulsewidth=1us, L=4 and BB_SamplingRate=10MHz with following I and Q waveform,

**Baseband I and Q waveform**

**Reference**

**RADAR_MatchedSrc Part**

*generate the matched source signal for pulse compression*

**Categories:** Signal Source (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

### Model

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRI_NUM</td>
<td>number of samples in PRI ((0:inf))</td>
<td>10000</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
</tbody>
</table>

### Input Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SRC_In</td>
<td>Terminal: Standard Data Port</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

### Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>SRC_Out</td>
<td>Terminal: Standard Data Port</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

### Notes/Equations

1. This model is used to generate the frequency domain matched output for a given waveform source.
2. Each firing PRI*BB_SamplingRate number of tokens are consumed in the input pin and the same amount of tokens are generated in the output pin.
3. Parameters detail,
   - PRI_NUM is used to specify the number of samples in a PRI.
4. The processing structure of the matched source is based on reverse of the input sequence with FFT as below,

![Diagram]

### Reference

Radar Transmitter

Contents

- RADAR CICInterp Part (radardesign)
- RADAR DUC Part (radardesign)
- RADAR Tx Part (radardesign)
RADAR_CICInterp Part

Categories: Transmitter (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADAR_CICInterp</td>
<td>RADAR CIC Interpolation</td>
</tr>
</tbody>
</table>

RADAR_CICInterp

Description: RADAR CIC Interpolation
Domain: Untimed
C++ Code Generation Support: NO
Associated Parts: RADAR CICInterp Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>the concatenation order of the CIC filter</td>
<td>5</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Ratio</td>
<td>the Interpolation ratio applied to the input signal</td>
<td>2</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>DiffDelay</td>
<td>Differential delay. Changes both the shape and number of</td>
<td>1</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nulls in the filter response. Also affects the null locations.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>In practice, it is usually held to one or two</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phase</td>
<td>where to put the input in the output block</td>
<td>0</td>
<td>Integer</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>Fill</td>
<td>value to fill the output block</td>
<td>0</td>
<td>Complex number</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

Input Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>input</td>
<td>complex signal to be interpolated</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>output</td>
<td>interpolated complex signal</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This model implements a multiple order cascaded integrator comb filter with interpolation.
2. Each firing,
   - 1 tokens is consumed at the input port
   - Ratio tokens are produced at the output port. It is equal to 2 by default.
3. Parameter details:
   - Order specifies the concatenation order of CIC filter.
   - Ratio specifies the interpolation ratio applied to the input signal.
   - DiffDelay specifies the differential delay for basic comb part shown in graph [Basic comb](#) as M. DiffDelay can be any positive integer, but it is usually limited to 1 or 2.
Basic comb Schematic
- Phase specifies the position of input bit after interpolation. For example, I denotes the input bit and F denotes the filling bit when interpolation. The input bit sequences are IIII and are applied with ratio 3. If phase=0, the interpolated bit sequences are IFFIFFIFF, if phase=1, the interpolated bit sequences are FIFFIFFIF, if phase=2, the interpolated bit sequences are FFIFFIFFI.
- Fill specifies the value to be filled when interpolation.

4. The CIC filter is a flexible, multiplier-free filter suitable for hardware implementation, that can also handle arbitrary and large rate changes. The two basic building blocks of a CIC filter are an integrator and a comb. The CIC filter is a cascade of digital integrators followed by a cascade of combs (digital differentiators) in equal number. Interpolation is also applied in this model.
- An integrator is simply a single-pole IIR filter with a unity feedback coefficient:
  \[ y[n] = y[n-1] + x[n] \]
  An integrator is known as an accumulator, the transfer function for an integrator on the z-plane is
  \[ H_I(z) = \frac{1}{1 - z^{-1}}. \]
  The schematic is shown as

Basic Integrator Schematic
The frequency response of an integrator is:
\[ \mathcal{H}_I(\omega) = \frac{1}{1 - e^{-j\omega}} \]
\[ = \frac{e^{j\omega/2} - e^{-j\omega/2}}{2} \]
\[ = \frac{e^{j\omega/2}}{2} \left( \sin \frac{\omega}{2} \right)^{-1} \]
- A comb filter is a FIR filter described by
  \[ y[n] = x[n] - x[n-M] \]
  In this equation, M is a design parameter and is called the differential delay. M can be any positive integer, but it is usually limited to 1 or 2. The corresponding transfer function on the z-plane is
  \[ H_c(z) = 1 - z^{-M} \]
  The schematic is shown in Basic comb.
  The frequency response of a comb is:
The amplitude response of a comb is:

$$H_2(e^{j\omega}) = 2e^{-j\omega M} \cdot \sin(\omega M/2)$$

The figure below shows the frequency response of one comb for M=7.

- The one stage CIC is implemented as one integrator followed by one comb as shown in the following graph:

![One Stage CIC](image)

The frequency response for one stage CIC is:

$$|H(e^{j\omega})| = |H_1(e^{j\omega}) \cdot H_2(e^{j\omega})|$$

$$= \frac{\sin(\omega M/2)}{\sin(\omega/2)}$$

The figure below shows the frequency response of one stage CIC for M=7.

- The N stage CIC interpolator is implemented as N cascaded comb stages running at $fs/R$, followed by a specified-stuffer, followed by N cascaded integrator stages running at $fs$. The following figure shows three stage interpolating CIC Filter schematic.

![Three stage CIC interpolator](image)

References

RADAR_DUC Part

RADAR Digital Up Converter

Categories: Transmitter (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

Model
RADAR_DUC (radardesign)

RADAR_DUC

Description: RADAR Digital Up Converter
Associated Parts: RADAR DUC Part (radardesign)

Model Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF_Freq</td>
<td>IF carrier frequency ((0:inf))</td>
<td>25000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>IF_SamplingRate</td>
<td>Digital IF signal sampling rate ((0:inf))</td>
<td>100000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BandWidth</td>
<td>passband bandwidth of IF filter ((0:inf))</td>
<td>5000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>In_CenterFreq</td>
<td>The center frequency of the input signal ((0:inf))</td>
<td>0</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BB_UpSamplingRatio</td>
<td>the upsampling ratio from baseband to digital IF ((1:inf))</td>
<td>20</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
<tr>
<td>RC_ExcessBW</td>
<td>Excess bandwidth of raised cosine filter ((0.0:1.0))</td>
<td>0.22</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>PhaseImbalance</td>
<td>phase imbalance in degrees, Q channel relative to I channel ((-inf:inf))</td>
<td>0</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>DAC_NBits</td>
<td>number of bits for DAC ((2:inf))</td>
<td>8</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
</tbody>
</table>

Input Ports

<table>
<thead>
<tr>
<th>Port Number</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Port_1</td>
<td>Terminal: Standard Data Port Terminal</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

Output Ports

<table>
<thead>
<tr>
<th>Port Number</th>
<th>Name</th>
<th>Description</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Port_3</td>
<td>Terminal: Standard Data Port Terminal</td>
<td>envelope</td>
<td>NO</td>
</tr>
</tbody>
</table>

Notes/Equations

1. This subnetwork implements digital upconverter which is also referred to as arbitrary waveform generation.
2. The RADAR_DUC schematic is shown below:
3. Parameter details:
   - IF_Freq specifies the IF modulation frequency.
   - IF_SamplingRate is the sampling rate of digital IF which is also the output sampling rate.
   - Bandwidth specifies the bandwidth of IF bandpass filter.
   - In_CenterFreq specifies the center frequency of input signal.
   - BB_UpsamplingRatio specifies the upsampling ratio from baseband to digital IF signal.
   - RC_ExcessBW specifies excess bandwidth of raised cosine filter which performs interpolation after baseband signal upsampling.
   - PhaseImbalance specifies phase imbalance in degree which is used to add certain impairments to the ideal output signal. The IF is given by
     \[ v_F(t) = v_I(t) \cos(\omega_c t) - v_Q(t) \sin(\omega_c t + \frac{\varphi \pi}{180}) \]
     \( \varphi \) is the phase imbalance.
   - DAC_NBits specifies the number of bits for DAC which converts the analogy signal to digital signal.

4. The following figure provides a block diagram of a DUC that translates a complex baseband signal up to an IF signal.

**Digital upconverter (DUC) schematic**

The baseband I and Q signals enter the DUC at baseband sampling rate (identified by BB_SamplingRate) and are first up-sampled by a factor of R. These signals are then passed through a digital lowpass filter that performs the interpolation. These signals are then multiplied by sine and cosine waveforms which can be generated by SineGen model for the modulation carrier frequency. Then summed with the Add model and converted to analog signal with DtoA model. Because the carrier frequency of the output signal of DtoA model is zeros, FcChange model is used to convert the signal to complex envelope signal. Finally the signal is passed through a bandpass filter to produce the IF signal.

- The data rate of upsampled baseband signal should be no smaller than 2xIF_Freq to avoid aliasing.

See:
**RADAR_DDC** (radardesign)

**Reference**

RADAR_Tx Part

RADAR Transmitter Front End

Categories: Transmitter (radardesign)

The models associated with this part are listed below. To view detailed information on a model (description, parameters, equations, notes, etc.), please click the appropriate link.

<table>
<thead>
<tr>
<th>Model</th>
<th>RADAR_Tx (radardesign)</th>
</tr>
</thead>
</table>

**RADAR_Tx**

**Description:** RADAR Transmitter Front End

**Associated Parts:** RADAR Tx Part (radardesign)

**Model Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Default</th>
<th>Units</th>
<th>Type</th>
<th>Runtime Tunable</th>
</tr>
</thead>
<tbody>
<tr>
<td>TStep</td>
<td>simulation time step; TStep=0 results in use of externally set TStep ([0:inf])</td>
<td>0</td>
<td>s</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>RF_Freq</td>
<td>RF carrier frequency ([0:inf])</td>
<td>10000000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>RF_Gain</td>
<td>Complex voltage gain; with form re+j*im; to specify gain in dB use dbpolar( dB, degree) ([-inf:inf])</td>
<td>1</td>
<td>none</td>
<td>Complex number</td>
<td>NO</td>
</tr>
<tr>
<td>IF_Freq</td>
<td>IF carrier frequency ([0:inf])</td>
<td>25000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>IF_Gain</td>
<td>Complex voltage gain; with form re+j*im; to specify gain in dB use dbpolar( dB, degree) ([-inf:inf])</td>
<td>1</td>
<td>none</td>
<td>Complex number</td>
<td>NO</td>
</tr>
<tr>
<td>IF_SamplingRate</td>
<td>Digital IF signal sampling rate ([0:inf])</td>
<td>10000000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BandWidth</td>
<td>passband bandwidth of IF filter ([0:inf])</td>
<td>5000000</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>In_CenterFreq</td>
<td>The center frequency of the input signal ([0:inf])</td>
<td>0</td>
<td>Hz</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>BB_UpSamplingRatio</td>
<td>the upsampling ratio from baseband to digital IF ([1:inf])</td>
<td>20</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
<tr>
<td>RC_ExcessBW</td>
<td>Excess bandwidth of raised cosine filter ([0:0:1.0])</td>
<td>0.22</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>PhaseImbalance</td>
<td>phase imbalance in degrees, Q channel relative to I channel ([&lt;inf:inf])</td>
<td>0</td>
<td>none</td>
<td>Float</td>
<td>NO</td>
</tr>
<tr>
<td>DAC_NBits</td>
<td>number of bits for DAC ([2:inf])</td>
<td>8</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
<tr>
<td>DAC_UpSamplingRatio</td>
<td>the upsampling ratio of DAC from digital IF to analog IF ([1:inf])</td>
<td>1</td>
<td>none</td>
<td>Integer</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Input Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BB_Signal</td>
<td>complex</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Output Ports**

<table>
<thead>
<tr>
<th>Port</th>
<th>Name</th>
<th>Signal Type</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>RF_Signal</td>
<td>envelope</td>
<td>NO</td>
</tr>
</tbody>
</table>

**Notes/Equations**

1. This subnetwork implements transmitter front end for RADAR system.
2. The RADAR_Tx schematic is shown below:
RADAR_Tx schematic

3. Parameter details:
   - TStep specifies the simulation time step which is the reciprocal of IF_SamplingRate.
   - RF_Freq specifies the RF carrier frequency in Hz.
   - RF_Gain specifies the gain of RF power amplifier in dbpolar.
   - IF_Freq specifies the digital upconverter carrier frequency in Hz.
   - IF_Gain specifies the gain of IF power amplifier in dbpolar.
   - IF_SamplingRate specifies the sampling rate of digital upconverter which is also the sampling rate of output signal.
   - Bandwidth specifies the bandwidth of IF bandpass filter which should be no smaller than the bandwidth of baseband signal.
   - In_CenterFreq specifies the center frequency of input signal.
   - BB_UpsamplingRatio specifies the upsampling ratio from baseband to digital IF signal.
   - RC_ExcessBW specifies excess bandwidth of raised cosine filter which performs interpolation after baseband signal upsampling.
   - PhaseImbalance specifies phase imbalance in degree which is used to add certain impairments to the ideal IF output signal as PhaseImbalance (radardesign).
   - DAC_NBits specifies the number of bits for DAC which converts the digital signal to analogy signal.

4. The following figure provides a block diagram of a digital front end that translates a complex baseband signal up to an RF signal.

Digital front end architecture

A digital complex baseband waveform is first passed through a digital upconverter to produce an IF signal, then mixed with an RF (complex envelope) to generate the RF signal. Then filtered with a bandpass filter to decrease the out-band noise. Finally, a power amplifier is applied before output the RF signal.

- The data rate of upsampled baseband signal should be no smaller than 2xIF_Freq to avoid aliasing.

See:
- RADAR_DUC (radardesign)
- RADAR_Rx (radardesign)

Reference