Agilent Technologies

MBP

Implementing Verilog-A Models in MBP

Application Note

Copyright Notice and Proprietary Information

Right to Copy Documentation

Disclaimer

AGILENT TECHNOLOGIES, INC. AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Implementing Verilog-A Models in MBP

Application Note

This application note describes how to implement Verilog-A models in Model Builder Program (MBP).
Note: This document was originally released for MBP V2009.1.0 in July 2011.

Introduction

Verilog-A is an industry standard modeling language for analog circuits. MBP initiated support of Verilog-A models with MBP v2009.1.0.

This application note describes how to implement Verilog-A models in MBP. For more information go to www.agilent.com/find/eesof or contact your local Agilent office. The complete list is available at: www.agilent.com/find/contactus.

Preparation

To implement Verilog-A models, the user must first ensure that MBP v2009.1.0 or a later version has been properly installed on the computer. Also, the Verilog-A license feature is needed.

Windows users must add "*$MBP_HOME\win32\<username>\bin*" to the environment variable "*Path.*" Here, *$MBP_HOME* stands for the directory where MBP is installed, for example *C:\Agilent\modelbuilder*. Then, reboot the computer.

For Linux users, run "which gcc" and "which g++" in the command line to make sure both gcc and g++ have been installed properly on the machine. Otherwise, contact your IT administrator.

Sample Models

There must be a subcircuit model to define which Verilog-A model is to be called and which parameters are to be tweaked. MBP allows the user to load this subcircuit model and tweak the parameters in the same way as any other model parameters in MBP.

Sample models are listed below for the HSPICE and SPECTRE simulators.

HSPICE

The following model, ekv.l, is an example of a model that is simulated by HSPICE:

```
.hdl ekv.va // Define Verilog-A model to use: ekv.va.
.model verilog1 ekv // Define new model named verilog1.
Use erilog-AV odel ekmv from ekv.va.
+VTO=0.5 // Define model parameters to be tweaked in MBP.
+GAMMA=1
+PHI=0.5
.subckt rf_nch d g s b W=10E-6 L=10E-6
x3 d g s b verilog1 L=L W=W // The user could use the new model
named verilog1,
x2 d g s b ekv L=5E-6 W=10E-6 // or use the original model named ekv.
.ends
```

SPECTRE

The following model, *ekv.l*, is an example of a model that is simulated by SPECTRE:

```
simulator lang=spectre
ahdl_include "ekv.va" // Define Verilog-A model to use: ekv.va.
model verilog1 ekv // Define new model named verilog1. Use
Verilog-A model ekv from ekv.va.
+VTO=0.5 // Define model parameters to be tweaked
in BP. M
+GAMMA=1
+PHI=0.5
subckt rf_nch (d g s b)
parameters W=10E-6 L=10E-6
x3 (d g s b) verilog1 l=l w=w // The user could use the new model
named verilog1,
x2 (d g s b) ekv l=5e-6 w=10e-6 // or use the original model named ekv.
ends rf_nch
```

Note that the element using Verilog-A must start with '*x,*' even in SPECTRE. Only the parameters declared in *verilog1*, such as *VTO, GAMMA and PHI*, can be tweaked in MBP. The original Verilog-A model (e.g., *ekv*) can be called and simulated in the subcircuit. However, the parameters in *ekv* not declared in *verilog1* cannot be tweaked in MBP.

MBP Supported Functions and Keywords

MBP supports most of the common functions and keywords defined in Verilog-A, including:

- Basic operation: supports most of the basic operation in Verilog-A
- Syntax: supports if/else, for loop, while loop, etc… Does not support repeat
- Simulation system function**:** supports $stop, $temperature, $vt, $vt(temp), and strobe("express")
- Function: supports user-defined function

For additional details, refer to Table 1.

Table 1. Support Table

Table 1 Support Table

Category	Type	Item	Example	Support Status	Description

Basic Operator	Mathematic	/		Supported	Attention:
					res = 1 / 5;
					// The result of this integer division is zero, res = 0.

		+		Supported	
		-		Supported	
		*		Supported	
		sqrt	sqrt(x)	Supported	
		ln	ln(x)	Supported	
		log	log(x)	Supported	
		abs	abs(x)	Supported	

Category	Type	Item	Example	Support Status	Description
		pow	pow(x,y)	Supported	
		min	min(x,y)	Supported	
		max	max(x,y)	Supported	
	Relational Operators	<	a>b	Supported	
		>	a<b	Supported	
		<=	a<=b	Supported	
		>=	a>=b	Supported	
	Logical Operators	!=		Supported	
		==		Supported	
		&&		Supported	
		||		Supported	
	Conditional Operator	? :	(a<b)?a:b	Supported	

	access	I( )	I(branch)
			I(node1,node2)
			I(node1)
				Supported	Attention :
					I(node1) means the current from the node1 to the ground

		V( )	V(node1,node2)
			V(node1)
				Supported	

	contribution	I(a,b)<+ V(c,d)		Supported	

		V(c,d)<+ variable or constant	V(in,mid)<+0.5;
			V(in,mid)<+x;
				Supported	

		I(c,d)<+variable or constant		Supported	
		V(c,d)<+ I(a,b)		Supported	
		I(a,b)<+I (c,d)		Not supported	
	ddx	Y=ddx(z, x);		Supported	
		Y=ddx(z, x);		Supported	Attention : Y= k*ddx(z,x); is

Category	Type	Item	Example	Support Status	Description
		Y=k* Y;			unacceptable
		Y= ddx(func(x),x);	Fun(x) is the function of x And fun is maybe	Supported	

		Y=ddx(Y ,x);	Y is a var	Not supported	Attention :
					But z=ddx(Y ,x);
					Y=z;
					that is right

		ddx(Userdefined function ,x)		Supported	
		I()<+ddx(y,x)		Not supported	Attention : But ,z=ddx(y,x) I()<+z Is right
		More than 2th derivate	ddx(ddx(a,b),b)	Not supported	
	idt			Not supported	
	assignment	y=V(p,n);		Supported	
		z=I(p,n);		Not supported	
Syntax		If else			
		Forloop		Supported	
		case		Supported	
		whileloo		Supported	

Category	Type	Item	Example	Support Status	Description
		p			
		repeat		Not supported	
	Defining Macros	`define		Supported	
	Conditional Compilation	`ifdef `else `endif		Supported	
	including	`include	`include "disciplines.vams"	Supported	
Simulation System Function	Simulation control	$stop		Supported	
		$finish			
	Environment Parameter Functions	$realtime	Current simulation time in seconds.	Not supported	
		$temperature	Ambient temperature in kelvin.	Supported	
		$vt	Thermal voltage ( ).	Supported	
		$vt(temp)	Thermal voltage at given temperature.	Supported	
		$abstime	Returns the simulation time, in seconds.	Not supported	
	Input/output	$fopen		Not supported	
		$fclose		Not supported	
		$fwrite		Not supported	
		$strobe(“express”)		Supported	
Function	Userdefined Function			Supported	
Analog	Initial Step	@(init_s		Not supported	

Category	Type	Item	Example	Support Status	Description
Events		tep( ))			
	Final Step	@(final_ step( ))		Not supported	
	Cross	cross()		Not supported	
	Timer	timer()		Not supported	
Hierarchical	Basic_ hierarchical			Not supported	
	Port_connect			Not supported	