SIMPLIS-2-PSPICE Translator Information

MVHC Apps

Exported on 03/26/2025

Table of Contents

Link to SIMPLIS-2-PSPICE Translator Release History[1]

1 https://confluence.itg.ti.com/display/LT30VAPPS/SIMPLIS-2-PSPICE+Translator+Release+History

1 Installation

Directions for installing necessary software tools.

1.1 Installing Python

Download Python Version 3.7 from ESD/

Anaconda is a python environment that is great to use when working on the python script.

Anaconda Installer - Anaconda (1).EXE[2]

1.2 Installing PSPICE

Approx time: 1-2 hours

1. Go to Flames/ → EDA Software Downloads
2. Scroll down to "Cadence Products" and click on Silicon Package Board
3. On this page is a link for installing Cadence Allegro, go to the link. https://flamessw.design.ti.com/sam/cadence/spb/17.4/
4. Download the 2 zip files called "Base_SPB17..." (wint_1of2 and wint_2of2). Extract both zip folders when they finish downloading.
5. Go to the "wint_1of2" folder and run the "setup" application.
6. Click Install for "OrCAD and Allegro Products Installation". This will start a long installation
7. Half-way through the installation, it will prompt you to select Disk2 folder. Browse to the wint_2of2 folder, open it and select the Disk2 folder. The installation will finish up (this takes a while).

```
If prompted for the license file path during installation, enter the license path setting according to your location;

US user groups SPB/PCB license path setting:

  5280@lelvflames20.itg.ti.com
```

1.3 Installing SIMPLIS

1. Flames/ → EDA Software Downloads → Simplis/Simetrix

2 https://confluence.itg.ti.com/download/attachments/300391857/Anaconda%20%281%29.EXE?api=v2&modificationDate=1597068313000&version=1

1.4 Downloading the SIMPLIS-2-PSPICE Translator

The latest release for the SIMPLIS-2-PSPICE Translator can be found on the following confluence page.

SIMPLIS-2-PSPICE Translator Release History[3]

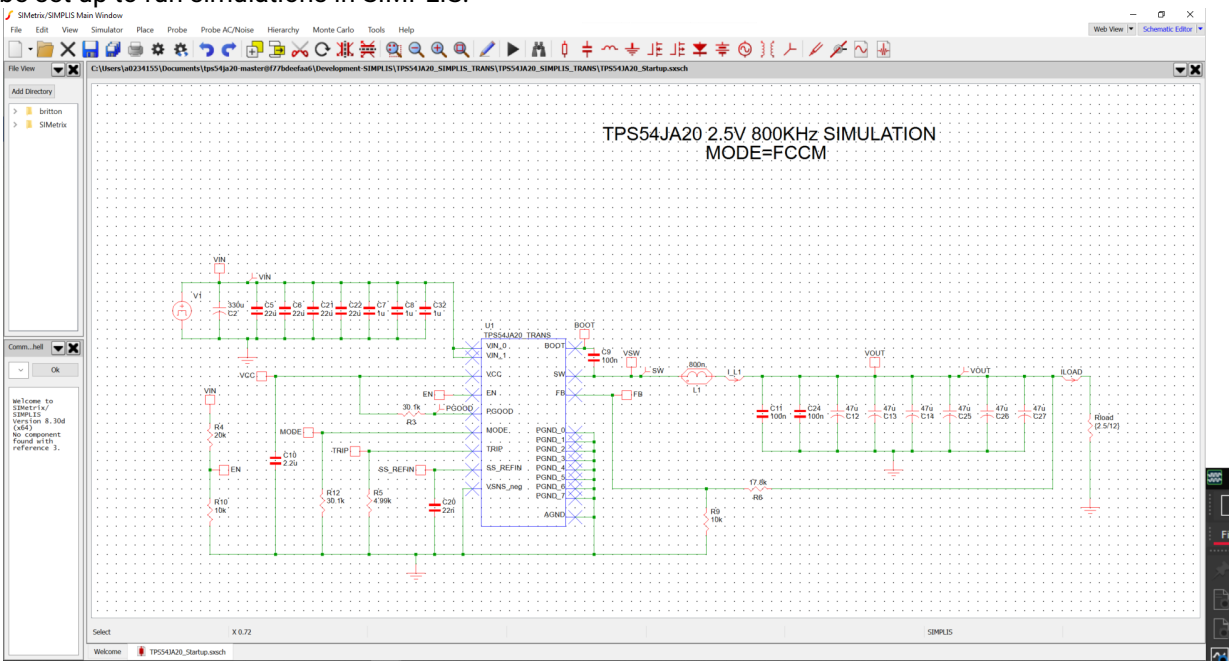
3 https://confluence.itg.ti.com/display/LT30VAPPS/SIMPLIS-2-PSPICE+Translator+Release+History

2 How to use the Translator

Directions for using the python script to translate a SIMPLIS netlist into a PSPICE netlist.

2.1 Getting the SIMPLIS Netlist

1. In SIMPLIS, open the top level SIMPLIS schematic for the part that you want to translate. This should be set up to run simulations in SIMPLIS.



Screenshot: Top level view of SIMPLIS schematic for Justin Jr.

2. Simulator > Run Schematic. This will run a SIMPLIS simulation

3. Simulator > Edit Netlist (before preprocess)

4. Save this .net file (For convenient use, save it to the folder where the SIMPLIS_2_PSPICE_Translator.py is located)

2.2 Running the Translator in Command Line

1. In command line, use the command [cd <destination>] to navigate to the folder where the Translator and SIMPLIS netlist are stored.

2. Enter the following command, replacing SIMPLIS_NETLIST with the name of your netlist.

```
[ python SIMPLIS_2_PSPICE_Translator.py SIMPLIS_NETLIST.net
<PSPICE_NETLIST.lib> ]
```

a. The PSPICE_NETLIST argument is optional if you want to choose the name of the PSPICE netlist. Without the optional argument, the PSPICE netlist will be the same name as the SIMPLIS netlist, only with a different extension.

```
Command Prompt
Microsoft Windows [Version 10.0.18363.693]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\a0234155>cd Desktop

C:\Users\a0234155\Desktop>python SIMPLIS_2_PSPICE_Translator.py TPS54JA20_Startu
File path containing translator   Translator   Input file path c
```

Screenshot: Command used to run the SIMPLIS_2_PSPICE Translator on the SIMPLIS Netlist for Justin Jr. (TPS54JA20_Startup.net)

3. When the program completes (takes about 1 second typically), locate the produced PSPICE netlist.

   a. The program will print out status checks to display which part of the netlist is being translated.

   b. The output file of the PSPICE netlist is printed in the commnad window when the program completes.

```
C:\Users\a0234155\Desktop>python SIMPLIS_2_PSPICE_Translator.py TPS54JA20_Startup.net
File read in complete...
Global variables complete...
Node mapping for subckt TPS54JA20_TRANS complete...
Translating subckt TPS54JA20_TRANS...
Subckt TPS54JA20_TRANS complete...
Node mapping for subckt PGOOD_OV_UV complete...
Translating subckt PGOOD_OV_UV...
Subckt PGOOD_OV_UV complete...
Node mapping for subckt MODE_SEL complete...
Translating subckt MODE_SEL...
Subckt MODE_SEL complete...
Node mapping for subckt Soft_Start complete...
Translating subckt Soft_Start...
Subckt Soft_Start complete...
Node mapping for subckt AMUX complete...
Translating subckt AMUX...
Subckt AMUX complete...
Node mapping for subckt HouseKeeping complete...
Translating subckt HouseKeeping...
Subckt HouseKeeping complete...
Appending subcircuit definitions...
TPS54JA20_Startup.lib written successfully...
Program complete.

C:\Users\a0234155\Desktop>
```

Status checks

Output file path of PSPICE netlist

Screenshot: Output window after running the command. Pictured here for the Justin Jr, netlist.

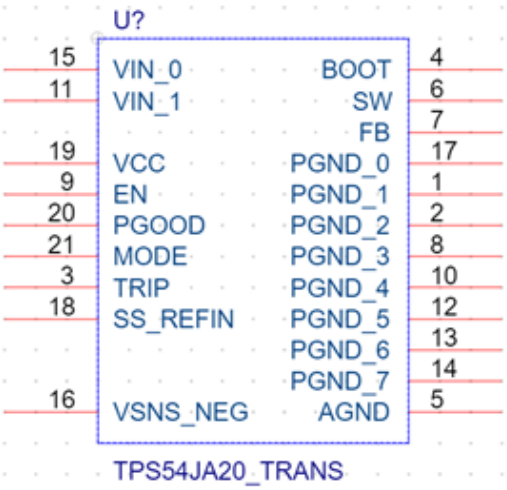
2.3 Verifying the PSPICE Model in Cadence

1. Open the PSPICE netlist (.lib file) in PSPICE Model Editor

2. Click File > Export to Capture Part Library (This creates the PSPICE symbol automatically (.olb file) for the PSPICE netlist)

3. Open the PSPICE symbol (.olb file) in Capture CIS

   Place the automatically generated symbol pins where you want them to go.



Screenshot: PSPICE symbol generated for the Justin Jr. PSPICE netlist

4. Create a top level PSPICE schematic and place your symbol inside to test it. This is the only manpower that needs to go into building circuitry by hand in PSPICE

5. Click New Simulation Profile and setup the simulation you want to run.

   a. Go to Configuration Files > Library and add your PSPICE netlist to the design from this window. This ensures that your netlist will be associated with your simulation.

   b. Go to Data Collection and ensure that “All” is selected for Voltages and Currents. This allows you to probe inside the netlist.

   c. Go to Probe Window and check “Last Plot” and “Display Probe Window: During Simulation”. This will allow you to debug the PSPICE simulation a lot quicker.

   d. Click Apply then Ok to leave the Edit Simulation Profile window

6. Click PSPICE > Run to start the simulation

7. Debug the PSPICE netlist as needed using the results of the simulation. You may please visit following pages for help in debugging:

   a. SIMPLIS-2-PSPICE Projects Learnings[4]

   b. SIMPLIS-2-PSPICE Debugging Tips[5]

2.4 What if something didn't get translated?

In the case that the Translator was not able to find a translation for some primitive in the SIMPLIS netlist, a blank dummy subckt is created in the PSPICE netlist. Scroll to the bottom of any translated PSPICE netlist to see if any blank subcircuits were created. You can use the blank subcircuit to reference which primitive didn’t get translated, then you can add your own PSPICE definition to the blank subckt. This is so that you can get your simulations up and running quickly in the event that a translation doesn’t exist.

4 https://confluence.itg.ti.com/display/LT30VAPPS/SIMPLIS-2-PSPICE+Projects+Learnings
5 https://confluence.itg.ti.com/display/LT30VAPPS/SIMPLIS-2-PSPICE+Debugging+Tips

If you are handy with python, please add the PSPICE translation logic to the script for any new SIMPLIS primitives, so that the script has that capability for future uses. If you are not comfortable with python, please reach out to Gerold Dhanabalan with information about the new primitive so that he can add it to a queue of primitives that will go on future releases of the SIMPLIS-2-PSPICE Translator.

3 About the Python Script

3.1 Script Structure

1. Read and store the SIMPLIS netlist (.net file) in an array
2. Concatenate all "+" lines in the netlist to the line they are associated with. Sometimes parameters for primitives can be written out on following lines. This is denoted by a "+" in the netlist. The script will concatenate all "+" lines onto the line they belong.
3. Translate all .globalvars and write them out as .PARAMS at the top level of the PSPICE netlist. If they are written abovethe first .subckt then they can be referenced by all following subckts.
4. Iterate through the array storing the SIMPLIS netlist and make translations based off of keywords. Create a blank subckt definition for any primitives that do not have translations.
    a. Make general translations (i.e. the starts and ends of subcircuits)
        i. Node mapping is done at the beginning of a user defined subckt
    b. Translate primitives of SIMPLIS netlist lines that start with an "X"
    c. Translate primitives of SIMPLIS netlist lines that start with an "!"
    d. Append primities of SIMPLIS netlist lines that don't need a translation
5. Limit all ROFF and RON values in the array storing the PSPICE netlist to 1T, so that ROFF and RON are not greater than 1/GMIN (the default GMIN value in PSPICE is 1E-12, hence the 1T limit).
6. Append subcircuit definitions
7. Append blank subcircuits defintions for missing primitives.
8. Write out the translated PSPICE netlist (.lib file)

3.2 Typical Logic Flow for Translations

1. Split up the elements of the SIMPLIS line to be translated into array2
2. Iterate through array2 to gather parameters from the SIMPLIS line
3. Build the PSPICE translation in array3 using information from array2
4. Write out array3 into one line, ensuring that the PSPICE maximum character limit is not breached
5. Add needed subckt definitions for the primitive, if they have not already been added

3.3 Functions

There are a number of functions in the script that will be extremely useful to anyone who is adding logic for new primitives.

For more information about these functions, see the comments for each function in the python file (SIMPLIS_2_PSPICE_Translator.py).

4 Translations

4.1 General Translations

Type	Search Word	SIMPLIS	PSPICE	Parameters in Use	Notes
☑ Type of primitive Check the box if this translation has been added to the script.	The keyword in SIMPLIS that the script looks for to make the translation.	SIMPLIS netlist example and any following lines that are used in translation	PSPICE equivalent netlist and associated subcircuit definitions.	List of parameters that are being used in the translation	Any other notes

Type	Search Word	SIMPLIS	PSPICE	Parameters in Use	Notes
	The search word(s) can also be used to search for this logic block in the python script. Copy and paste the search word into the Find tool in a text editor to quickly jump to this spot in the code. Include the quotation marks for greater precision.				
Start of User Created Subcircuit	".subckt"	.subckt NAME node1 node2 node3	.subckt NAME node1 node2 node3		

Type	Search Word	SIMPLIS	PSPICE	Parameters in Use	Notes
End of User Created Subcircuit	".ends " + subckt_name	.ends NAME	.ends NAME		
Global Variables	“.globalvar”	.globalvar NAME VALUE	.PARAM NAME VALUE		▪ Global variables are handled prior to the first subckt translation, so that the .PARAM will be referencable by all subcircuits below.
Local Variables	".var"	.var NAME VALUE	.PARAM NAME VALUE		▪ Local variables are written within .subckt translations and are local to that .sucbkt
Voltage Source	"V"	V2 V+ V- Voltage	V2 V+ V- Voltage		
Current Source	"I"	I1 v+ v- Current	1 v+ v- Current		

Type	Search Word	SIMPLIS	PSPICE	Parameters in Use	Notes
Resistor	"R"	R_R24 n1 n2 VALUE	R_R24 n1 n2 VALUE		▪ Can not have a value of 0 PSPICE but it could in SIMPLIS. If VALUE is 0, then it is changed to 1E-6 in PSPICE.
Capacitor	"C"	C_F24 n1 n2 VALUE	C_F24 n1 n2 VALUE		• Can not have a value of 0 PSPICE but it could in SIMPLIS. If VALUE is 0, then it is changed to 1E-6 in PSPICE.
Inductor	"L"	L_124 n1 n2 VALUE	L_124 n1 n2 VALUE		• Can not have a value of 0 PSPICE but it could in SIMPLIS. If VALUE is 0, then it is changed to 1E-6 in PSPICE.
Current Controlled Current Source	"F"	F2 node1 node2 VF2$TP_CCCS 1 VF2$TP_CCCS cc+ cc- 0	F2 node1 node2 VF2$TP_CCCS 1 VF2$TP_CCCS cc+ cc- 0		

Type	Search Word	SIMPLIS	PSPICE	Parameters in Use	Notes
Voltage Controlled Current Source	"G"	G1 Vout+ Vout- Vc+ Vc- gain	G1 Vout+ Vout- Vc+ Vc- gain		
Current Controlled Voltage Source	"H"	H_U1_H1 3 4 VH_U1_H1 1 VH_U1_H1 1 2 0V	H_U1_H1 3 4 VH_U1_H1 1 VH_U1_H1 1 2 0V		
Voltage Controlled Voltage Source	"E"	E1 Vout+ Vout- Vc+ Vc- gain	E1 Vout+ Vout- Vc+ Vc- gain		

4.2 Primitive translations that start with "X"

Summer	"SIMPLIS_SUMMER2_BB"	X$U1 VOUT+ VOUT- VIN1 VIN2 SIMPLIS_SUMMER2_BB vars: K1=1 K2=250m OFFSET=0 + USE_RIN=0 RIN=1G	E_ABM_U1 VOUT+ VOUT- VALUE {(K1*V(VIN1))+(K2*V(VIN2))}	• K1: Gain of VIN1 • K2: Gain of VIN2	

☑ Comparator with complimented output node and reference node	"SIMPLIS_DIGI1_COMP_Y" or "SIMPLIS_LOGIC_BB_COMP"	X$U13 out inv_out ref v+ v- SIMPLIS_DIGI1_COMP_Y vars: IC=0 RIN=50G ROUT=10 + HYSTWD=2m VOL=0 VOH=3 DELAY=2p	X_U13_comp v+ v- out ref COMPHYS2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=<((VDD-VSS)/2)+VSS> VHYS=HYSTWD ROUT=ROUT X_U13_compNot out inv_out INV_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=<((VDD-VSS)/2)+VSS> ROUT=ROUT .subckt COMPHYS2_BASIC_GEN INP INM OUT REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 VHYS=1u ROUT=1 EIN INP1 INM1 INP INM 1 EHYS INM2 INM1 VALUE { IF(V(1) > {VTHRESH}, -{VHYS}/2, {VHYS}/2) } EOUT OUT REF VALUE { IF(V(INP1)>V(INM2), {VDD}, {VSS}) } R1 OUT 1 {ROUT} C1 1 REF 1E-9 RINP1 INP1 REF 10K RINM2 INM2 REF 10K .ENDS COMPHYS2_BASIC_GEN	• VOH: Logic high voltage • VOL: Logic low voltage • HYSTWD: Hysteresis voltage used with the calculated parameter VTHRESH to compare input voltages • ROUT: Output resistance	

			.SUBCKT INV_BASIC_GEN A Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 + DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A)>{VTHRESH} , {VSS}, {VDD})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS INV_BASIC_GEN		

☑ Comparator with complimented output node and no reference node	"SIMPLIS_DIGI1_COMP_N"	X$U13 out inv_out v+ v- SIMPLIS_DIGI1_COMP_N vars: IC=0 RIN=50G ROUT=10 + HYSTWD=2m VOL=0 VOH=3 DELAY=2p	X_U13_comp v+ v- out ref COMPHYS2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=<((VDD-VSS)/2)+VSS> VHYS=HYSTWD ROUT=ROUT X_U13_compNot out inv_out INV_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=<((VDD-VSS)/2)+VSS> ROUT=ROUT .subckt COMPHYS2_BASIC_GEN INP INM OUT REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 VHYS=1u ROUT=1 EIN INP1 INM1 INP INM 1 EHYS INM2 INM1 VALUE { IF(V(1) > {VTHRESH}, -{VHYS}/2, {VHYS}/2) } EOUT OUT REF VALUE { IF(V(INP1)>V(INM2), {VDD}, {VSS}) } R1 OUT 1 {ROUT} C1 1 REF 1E-9 RINP1 INP1 REF 10K RINM2 INM2 REF 10K .ENDS COMPHYS2_BASIC_GEN	• VOH: Logic high voltage • VOL: Logic low voltage • HYSTWD: Hysteresis voltage used with the calculated parameter VTHRESH to compare input voltages • ROUT: Output resistance	

			.SUBCKT INV_BASIC_GEN A Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A)>{VTHRESH} , {VSS}, {VDD})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS INV_BASIC_GEN		

Comparator with delay	"SIMPLIS_COMP"	X$U1 out1 out2 in1 in2 SIMPLIS_COMP vars: IC=0 RIN=10Meg ROUT=1 HYSTWD=1u + VOL=0 VOH=3 DELAY=30n	X_U1 in1 in2 comp_out out2 COMPHYS2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=<[VOH-VOL]/2 + VOL> VHYS=HYSTWD ROUT=ROUT X_U1_delay comp_out out1 out2 BUF_DELAY_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=<[VOH-VOL]/2 + VOL> DELAY=DELAY ROUT=ROUT .subckt COMPHYS2_BASIC_GEN INP INM COMP_OUT REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 VHYS=1u + ROUT=1 EIN INP1 INM1 INP INM 1 EHYS INM2 INM1 VALUE { IF(V(1) > {VTHRESH}, -{VHYS}/2, {VHYS}/2) } EOUT COMP_OUT REF VALUE { IF(V(INP1)>V(INM2), {VDD}, {VSS}) } R1 COMP_OUT 1 {ROUT} C1 1 REF 1E-9 RINP1 INP1 REF 10K RINM2 INM2 REF 10K	• DELAY: Delay of the buffer. If DELAY=0 then no buffer is used. • VOH: Logic high voltage • VOL: Logic low voltage • HYSTWD: Hysteresis voltage used with the calculated parameter VTHRESH to compare input voltages • ROUT: Output resistance	

			.ENDS COMPHYS2_BASIC_GEN .subckt BUF_DELAY_BASIC_GEN A Y REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 DELAY=10n ROUT=1 E_ABMGATE1 YINT1 0 VALUE { {IF((V(A)-V(REF)) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT1 YINT2 1 CINT YINT2 REF {DELAY*1.3} E_ABMGATE2 YINT3 REF VALUE { {IF(V(YINT2) > {VTHRESH}, {VDD}, {VSS})} } RINT2 YINT3 Y {ROUT} CINT2 Y REF 1n .ENDS BUF_DELAY_BASIC_GEN		

Digital Comparator	"SIMPLIS_DIGI1_D_D_COMPARATOR"	X$U6 a<b a<=b a=b a>=b a>b a0 a1 a2 a3 b0 b1 b2 b3 rtn + SIMPLIS_DIGI1_D_D_COMPARATOR_YX$U6 vars: RIN=10Meg ROUT=10 TH=1 + HYSTWD=100m VOL=0 VOH=5 OUT_DELAY=1n CODE='UNSIGNED' NUMBITS_A=4 + NUMBITS_B=4 IC=0	a<b a<=b a=b a>=b a>b a0 a1 a2 a3 b0 b1 b2 b3 rtn DIGI_COMP_4IN_4IN PARAMS: + THRESH=TH VOH=VOH VOL=VOL .subckt DIGI_COMP_4IN_4IN out1 out2 out3 out4 out5 a0 a1 a2 a3 b0 b1 b2 b3 ref + PARAMS: THRESH=0.5 VOH=1 VOL=0 E_ABM_a0 1 0 VALUE { IF(V(a0)>{THRESH}, 1, 0) } E_ABM_a1 2 0 VALUE { IF(V(a1)>{THRESH}, 2, 0) } E_ABM_a2 3 0 VALUE { IF(V(a2)>{THRESH}, 4, 0) } E_ABM_a3 4 0 VALUE { IF(V(a3)>{THRESH}, 8, 0) } E_ABM_A A 0 VALUE { V(1)+V(2)+V(3)+V(4) } E_ABM_b0 6 0 VALUE { IF(V(b0)>{THRESH}, 1, 0) } E_ABM_b1 7 0 VALUE { IF(V(b1)>{THRESH}, 2, 0) } E_ABM_b2 8 0 VALUE { IF(V(b2)>{THRESH}, 4, 0) }	• Available cases: A=4bits/ B=4bits only • TH: Threshold voltage • VOH: Logic high voltage • VOL: Logic low voltage	

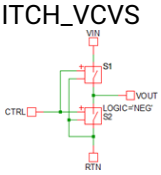
			E_ABM_b3 9 0 VALUE { IF(V(b3)>{THRESH}, 8, 0) } E_ABM_B B 0 VALUE { V(6)+V(7)+V(8)+V(9) } E_ABM_OUT1 out1 ref VALUE { IF(V(A)<V(B), VOH, VOL) } E_ABM_OUT2 out2 ref VALUE { IF(V(A)<=V(B), VOH, VOL) } E_ABM_OUT3 out3 ref VALUE { IF(V(A)==V(B), VOH, VOL) } E_ABM_OUT4 out4 ref VALUE { IF(V(A)>=V(B), VOH, VOL) } E_ABM_OUT5 out5 ref VALUE { IF(V(A)>V(B), VOH, VOL) } .ends DIGI_COMP_4IN_4IN		

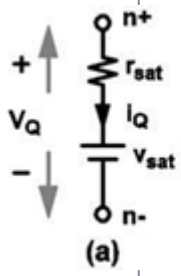
☑ Comparator Model	"TP_GATE" Note: The script searches for“COMP” in the .MODEL line after finding “TP_GATE” to ensure that this is a comparator model.	X$U1 Y A U1$TP_GATE .SUBCKT U1$TP_GATE 1 2 !D1 1 0 2 3 NAME IC=0 V1 3 0 VCOMP .MODEL NAME COMP RIN=10meg ROUT=41.67 HYSTWD=0.1 VOL=0 VOH=5 + DELAY=1e-008 .ENDS	X_U1 A Y COMP_MODEL_LOGIC PARAMS: VOH=VOH VOL=VOL VCOMP=VCOMP + VTHRESH=<[VOH-VOL]/2 + VOL> .subckt COMP_MODEL_LOGIC A Y PARAMS: VOH=1 VOL=0 VCOMP=0.5 VTHRESH=0.5 V1 3 0 {VCOMP} X_COMP A 3 Y 0 COMPHYS2_BASIC_GEN PARAMS: VDD={VOH} VSS={VOL} VTHRESH={VTHRESH} .ends COMP_MODEL_LOGIC .subckt COMPHYS2_BASIC_GEN INP INM COMP_OUT REF PARAMS: VDD=1 VSS=0 + VTHRESH=0.5 VHYS=1u ROUT=1 EIN INP1 INM1 INP INM 1 EHYS INM2 INM1 VALUE { IF(V(1) > {VTHRESH}, -{VHYS}/2, {VHYS}/2) } EOUT COMP_OUT REF VALUE { IF(V(INP1)>V(INM2), {VDD}, {VSS}) }	• VCOMP: Comparing the voltage of node A to this voltage • VOH: Logic high voltage • VOL: Logic low voltage	

			R1 COMP_OUT 1 {ROUT} C1 1 REF 1E-9 RINP1 INP1 REF 10K RINM2 INM2 REF 10K .ENDS COMPHYS2_BASIC_GEN		
☑ Diodes	"X$D"	X$D1 node1 node2 ... *Diodes come in a lot of different forms depending on the diode model chosen in SIMPLIS. For all cases, one of 3 diodes will be selected in PSPICE. • If diode line contains "SMBD2835/SIE", then choose PSPICE diode D_D1. • If diode line contains "DIODE_SPICE_V2", then choose PSPICE diode D_D1. • If diode line contains "D1n4148", then choose PSPICE diode D_D2. • Else, choose the default PSPICE diode D_IDEAL.	X_D1 node1 node2 DIODE *DIODE: D_D1 .subckt D_D1 1 2 d1 1 2 dd1 .model dd1 d is=1e-15 rs=0.05 n=0.5 .ends D_D1 *DIODE: D_D2 .subckt D_D2 1 2 d1 1 2 dd2 .model dd2 d is=1e-15 tt=1e-11 rs=0.001 n=0.001 .ends D_D2 *DIODE: D_IDEAL .subckt D_IDEAL 1 2 d1 1 2 dd1 .model dd1 d is=1e-15 rs=0.005 n=1 .ends D_IDEAL		• Diode models need to be parameterized in order to have the exact behavior of the SIMPLIS diode. • These diode options should get you close to where you want to go, but feel free to alter the diode definitions in the produced PSPICE netlist to reflect your specific diode cases.

☑ Digital Constant	"SIMPLIS_DIGI1_D_CONSTANT"	X$U17 node SIMPLIS_DIGI1_D_CONSTANTX$U17 vars: VALUE=0	V_U17 node 0 VOLTAGE	• Utilizes the find_H_L_THRESH() function to pull parameters VOH and VOL from another line in the netlist. • VALUE=0: VOLTAGE is set to VOL • VALUE=1: VOLTAGE is set to VOH	

☑ Voltage Controlled Switch	"VC_SWITCH"	X$S1 v+ v- ctrl+ ctrl- 4 SIMPLIS_VC_SWITCH vars: ROFF=1G RON=1m + THRESHOLD=500m HYSTWD=100m IC='OPEN' LOGIC='POS'	S_S5 1 2 3 4 SIMPLIS_VC_SWITCH_S1 .MODEL SIMPLIS_VC_SWITCH_S1 VSWITCH ROFF=1G RON=1m VOFF=<THRESHOLD-HYSTWD/2> + VON=<THRESHOLD+HYSTWD/2>	• RON: On resistance • ROFF: Off resistance • THRESHOLD: Threshold voltage. Used with HYSTERESIS to calculate VOFF and VON • HYSTWD: Hysteresis voltage. Used with THRESHOLD to calculate VOFF and VON • LOGIC: If LOGIC='NEG', then switch VOFF and VON.	

☑ Switch VCVS	"SIMPLIS_SWITCH_VCVS_BB"	X$U3 VOUT RTN VIN CTRL SIMPLIS_SWITCH_VCVS_BB vars: THRESHOLD=1 + HYSTWD=100m IC=0 LOGIC='POS'	X_U3 VOUT RTN VIN CTRL SIMPLIS_SWITCH_VCVS PARAMS: VON=<THRESHOLD +HYSTWD/2> VOFF=<THRESH-HYSTWD/2> .subckt SIMPLIS_SWITCH_VCVS vout rtn vin ctrl PARAMS: VON=0.6 VOFF=0.4 S1 VIN VOUT CTRL RTN SIMPLIS_VC_SWITCH1 .MODEL SIMPLIS_VC_SWITCH1 VSWITCH ROFF=1G RON=1m VOFF={VOFF} VON={VON} S2 VOUT RTN CTRL RTN SIMPLIS_VC_SWITCH2 .MODEL SIMPLIS_VC_SWITCH2 VSWITCH ROFF=1G RON=1m VOFF={VON} VON={VOFF} .ends SIMPLIS_SWITCH_VCVS	• THRESHOLD: Threshold voltage. Used with HYSTERESIS to calculate VOFF and VON • HYSTWD: Hysteresis voltage. Used with THRESHOLD to calculate VOFF and VON • LOGIC: If LOGIC='NEG', then switch VOFF and VON.	• Schematic for a SIMPLIS_SWITCH_VCVS 

Voltage Controlled Transistor Switch	"SIMPLIS_VC_QSWITCH"	X$S5 v+ v- vc+ vc- SIMPLIS_VC_QSWITCH vars: ROFF=1Meg RSAT=50 TH=200m + HYSTWD=1m VSAT=0 GAIN=10 IC='OPEN' LOGIC='POS' IFLOW='Bi-directional' + POLARITY='Positive' LEVEL=2	X_S5 v+ v- vc+ vc- SIMPLIS_VC_QSWITCH PARAMS: ROFF=ROFF RSAT=RSAT + VOFF=<TH-HYSTWD/2> VON=<TH+HYSTWD/2> VSAT=VSAT .subckt SIMPLIS_VC_QSWITCH vp vn vcp vcn PARAMS: ROFF=1Meg RSAT=50 VOFF=0.6 + VON=0.5 VSAT=0 S_on vp 2 vcp vcn SIMPLIS_VC_SWITCH1 .MODEL SIMPLIS_VC_SWITCH1 VSWITCH ROFF=1G RON=1m VOFF={VOFF} VON={VON} R_sat 2 3 {RSAT} V_sat 3 vn {VSAT} S_off vp 1 vcp vcn SIMPLIS_VC_SWITCH2 .MODEL SIMPLIS_VC_SWITCH2 VSWITCH ROFF=1G RON=1m VOFF={VON} VON={VOFF} R_off 1 vn {ROFF} .ends SIMPLIS_VC_QSWITCH	• RSAT: Saturation resistance • ROFF: Off resistance • TH: Threshold voltage. Used with HYSTERESIS to calculate VOFF and VON • HYSTWD: Hysteresis voltage. Used with THRESHOLD to calculate VOFF and VON • LOGIC: If LOGIC='NEG', then switch VOFF and VON. • VSAT: Saturation voltage that is produced when device is in saturation.	• Layout showing the saturation path (a) and off path (b) of a transistor switch 

Current Controlled Transistor Switch	"SIMPLIS_CC_QSWITCH"	X$S2 v+ v- cc+ cc- SIMPLIS_CC_QSWITCH vars: ROFF=1G RSAT=100k TH=0 HYSTWD=1p + VSAT=100m GAIN=1 IC='OPEN' LOGIC='POS' IFLOW='Uni-directional' + POLARITY='Positive' LEVEL=2	X_S2 v+ v- cc+ cc- SIMPLIS_CC_QSWITCH PARAMS: ROFF=ROFF RSAT=RSAT + IOFF=<TH-HYSTWD/2> ION=<TH+HYSTWD/2> VSAT=VSAT .subckt SIMPLIS_CC_QSWITCH vp vn ccp ccn PARAMS: ROFF=1Meg RSAT=50 IOFF=0 + ION=-1E-3 VSAT=0 V_W_S ccp ccn 0V W_S2 vp 2 V_W_S ISW_S2 .MODEL ISW_S2 ISWITCH ROFF=1G RON=1m IOFF={IOFF} ION={ION} R_sat 2 3 {RSAT} V_sat 3 vn {VSAT} W_S1 vp 1 V_W_S ISW_S1 .MODEL ISW_S1 ISWITCH ROFF=1G RON=1m IOFF={ION} ION={IOFF} R_off 1 vn {ROFF} .ends SIMPLIS_CC_QSWITCH	• RSAT: Saturation resistance • ROFF: Off resistance • TH: Threshold current. Used with HYSTERESIS to calculate IOFF and ION • HYSTWD: Hysteresis current. Used with THRESHOLD to calculate IOFF and ION • LOGIC: If LOGIC='NEG', then switch IOFF and ION. • VSAT: Saturation voltage that is produced when device is in saturation.	

XOR / XNOR with reference node	"SIMPLIS_DIGI1_XOR_Y" or "SIMPLIS_LOGIC_BB_XOR"	X$U25 xor xnor ref A B SIMPLIS_DIGI1_XOR_Y vars: NumInv=0 IC=0 RIN=10Meg + ROUT=10 HYSTWD=500m VOL=0 VOH=5 DELAY=120p TH=1.5	X_U25_xor A B xor XOR_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH ROUT=ROUT DELAY=DELAY X_U25_xnor A B xnor XNOR_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH ROUT=ROUT DELAY=DELAY .SUBCKT XOR_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF((V(A) > {VTHRESH} & V(B) < {VTHRESH}) | + (V(A) < {VTHRESH} & V(B) {VTHRESH}), {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS XOR_BASIC_GEN .SUBCKT XNOR_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF((V(A) > {VTHRESH} & V(B) < {VTHRESH}) | + (V(A) < {VTHRESH} & V(B) {VTHRESH}), {VSS}, {VDD})} }	• ROUT: Output resistance • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal • DELAY: Delay (in seconds) can be represented in Farads at the output capacitor by multiplying it by 1.3.	

			RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS XNOR_BASIC_GEN		

XOR / XNOR without reference node	"SIMPLIS_DIGI1_XOR_N"	X$U25 xor xnor A B SIMPLIS_DIGI1_XOR_Y vars: NumInv=0 IC=0 RIN=10Meg + ROUT=10 HYSTWD=500m VOL=0 VOH=5 DELAY=120p TH=1.5	X_U25_xor A B xor XOR_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH ROUT=ROUT DELAY=DELAY X_U25_xnor A B xnor XNOR_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH ROUT=ROUT + DELAY=DELAY .SUBCKT XOR_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF((V(A) > {VTHRESH} & V(B) < {VTHRESH}) | + (V(A) < {VTHRESH} & V(B) {VTHRESH}), {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS XOR_BASIC_GEN .SUBCKT XNOR_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF((V(A) > {VTHRESH} & V(B) < {VTHRESH}) |	• NumInv: Number of inverted inputs. The inputs are inverted in a backwards order, so the last input is inverted when NumInv=1 and the first input isn't inverted unless all the inputs are inverted. • ROUT: Output resistance • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal	

			+ (V(A) < {VTHRESH} & V(B) {VTHRESH}), {VSS}, {VDD})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS XNOR_BASIC_GEN	• DELAY: Delay (in seconds) can be represented in Farads at the output capacitor by multiplying it by 1.3.	

☑ AND / NAND with reference node	"SIMPLIS_DIGI1_AND2_Y" or "SIMPLIS_LOGIC_BB_AND2"	X$U13 and_out nand_out ref in1 in2 SIMPLIS_DIGI1_AND2_Y vars: NumInv=0 IC=1 + RIN=10Meg ROUT=10 HYSTWD=1 VOL=0 VOH=3 DELAY=2p TH=1.5	X_U13_and in1 in2 and_out AND2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=TH ROUT=ROUT DELAY=DELAY X_U13_nand in1 in2 nand_out NAND2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=TH ROUT=ROUT DELAY=DELAY .SUBCKT AND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH},{VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS AND2_BASIC_GEN .SUBCKT NAND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH},{VSS}, {VDD})} } RINT YINT Y {ROUT}	• Available cases: 2 inputs, 3, 4, 5, 6, 7, 8 • NumInv: Number of inverted inputs. The inputs are inverted in a backwards order, so the last input is inverted when NumInv=1 and the first input isn't inverted unless all the inputs are inverted. • ROUT: Output resistance • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal	

			CINT Y 0 {DELAY*1.3} .ENDS NAND2_BASIC_GEN	• DELAY: Delay (in seconds) can be represented in Farads at the output capacitor by multiplying it by 1.3.	

☑ AND / NAND without reference ndoe	"SIMPLIS_DIGI1_AND2_N"	X$U13 and_out nand_out in1 in2 SIMPLIS_DIGI1_AND2_N vars: NumInv=0 IC=1 + RIN=10Meg ROUT=10 HYSTWD=1 VOL=0 VOH=3 DELAY=2p TH=1.5	X_U13_and in1 in2 and_out AND2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=TH ROUT=ROUT DELAY=DELAY X_U13_nand in1 in2 nand_out NAND2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=TH ROUT=ROUT DELAY=DELAY .SUBCKT AND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH},{VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS AND2_BASIC_GEN .SUBCKT NAND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH},{VSS}, {VDD})} } RINT YINT Y {ROUT}	• Available cases: 2 inputs, 3, 4, 9 • NumInv: Number of inverted inputs. The inputs are inverted in a backwards order, so the last input is inverted when NumInv=1 and the first input isn't inverted unless all the inputs are inverted. • ROUT: Output resistance • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal	

			CINT Y 0 {DELAY*1.3} .ENDS NAND2_BASIC_GEN	• DELAY: Delay (in seconds) can be represented in Farads at the output capacitor by multiplying it by 1.3.	
☑ AND gate	"SIMPLIS_AND2"	X$U25 out ref in1 in2 SIMPLIS_AND2 vars: IC=1 RIN=100Meg ROUT=1 TH=1 + HYSTWD=10m VOL=0 VOH=3 DELAY=1f	X_U25 in1 in2 out AND2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH ROUT =ROUT DELAY=DELAY .SUBCKT AND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH},{VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS AND2_BASIC_GEN	• Available cases: 2 inputs • ROUT: Output resistance • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal • DELAY: Delay (in seconds) can be represented in Farads at the output capacitor by multiplying it by 1.3.	

☑ OR/NOR with reference node	"SIMPLIS_DIGI1_OR2_Y" or "SIMPLIS_LOGIC_BB_OR2"	X$U13 or_out nor_out ref in1 in2 SIMPLIS_DIGI1_OR2_Y vars: NumInv=0 IC=1 + RIN=10Meg ROUT=10 HYSTWD=1 VOL=0 VOH=3 DELAY=2p TH=1.5	X_U13_or in1 in2 or_out OR2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=TH ROUT=ROUT DELAY=DELAY X_U13_nor in1 in2 nor_out NOR2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=TH ROUT=ROUT DELAY=DELAY .SUBCKT OR2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 + DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} | V(B) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS OR2_BASIC_GEN .SUBCKT NOR2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 + DELAY=1n	• Available cases: 2inputs, 3, 4, 5, 6, 7, 8 • NumInv: Number of inverted inputs. The inputs are inverted in a backwards order, so the last input is inverted when NumInv=1 and the first input isn't inverted unless all the inputs are inverted. • ROUT: Output resistance • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal	

			E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} | V(B) > {VTHRESH}, {VSS}, {VDD})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS NOR2_BASIC_GEN	• DELAY: Delay (in seconds) can be represented in Farads at the output capacitor by multiplying it by 1.3.	

☑ OR/NOR without reference node	"SIMPLIS_DIGI1_OR2_N"	X$U13 or_out nor_out in1 in2 SIMPLIS_DIGI1_OR2_Y vars: NumInv=0 IC=1 + RIN=10Meg ROUT=10 HYSTWD=1 VOL=0 VOH=3 DELAY=2p TH=1.5	X_U13_or in1 in2 or_out OR2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=TH ROUT=ROUT DELAY=DELAY X_U13_nor in1 in2 nor_out NOR2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL + VTHRESH=TH ROUT=ROUT DELAY=DELAY .SUBCKT OR2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 + DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} | V(B) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS OR2_BASIC_GEN .SUBCKT NOR2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} | V(B) > {VTHRESH}, {VSS}, {VDD})} }	• Available cases: 2inputs, 3, 4 • NumInv: Number of inverted inputs. The inputs are inverted in a backwards order, so the last input is inverted when NumInv=1 and the first input isn't inverted unless all the inputs are inverted. • ROUT: Output resistance • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal	

			RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS NOR2_BASIC_GEN	• DELAY: Delay (in seconds) can be represented in Farads at the output capacitor by multiplying it by 1.3.	
OR gate	"SIMPLIS_OR2"	X$U25 out ref in1 in2 SIMPLIS_OR2 vars: IC=1 RIN=100Meg ROUT=1 TH=1 HYSTWD=10m VOL=0 VOH=3 DELAY=1f	X_U25 in1 in2 out OR2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH + ROUT=ROUT DELAY=DELAY .SUBCKT OR2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} | V(B) > {VTHRESH},{VDD},{VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS OR2_BASIC_GEN	• Avaialable cases: 2inputs • ROUT: Output resistance • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal • DELAY: Delay (in seconds) can be represented in Farads at the output capacitor by multiplying it by 1.3.	

☑ BUF/INV with reference node	"SIMPLIS_DIGI1_BUF_Y" or "SIMPLIS_LOGIC_BB_BUF"	X$U1 buf_out inv_out ref in SIMPLIS_DIGI1_BUF_Y vars: IC=1 RIN=10Meg ROUT=10 + HYSTWD=1 VOL=0 VOH=3 DELAY=2p TH=1.5	X_U1_buf in buf_out BUF_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH X_U1_inv in inv_out INV_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .SUBCKT BUF_BASIC_GEN A Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A)>{VTHRESH} , {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS BUF_BASIC_GEN .SUBCKT INV_BASIC_GEN A Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A)>{VTHRESH} , {VSS}, {VDD})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS INV_BASIC_GEN	• VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal • ROUT: Output resistance • DELAY: Effects output capacitance	

☑ BUF/INV without reference node	"SIMPLIS_DIGI1_BUF_N"	X$U1 buf_out inv_out in SIMPLIS_DIGI1_BUF_Y vars: IC=1 RIN=10Meg ROUT=10 + HYSTWD=1 VOL=0 VOH=3 DELAY=2p TH=1.5	X_U1_buf in buf_out BUF_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH X_U1_inv in inv_out INV_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .SUBCKT BUF_BASIC_GEN A Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A)>{VTHRESH} , {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS BUF_BASIC_GEN .SUBCKT INV_BASIC_GEN A Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A)>{VTHRESH} , {VSS}, {VDD})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS INV_BASIC_GEN	• VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal • ROUT: Output Resistance • DELAY: Effects output capacitance	

BUF with delay	"SIMPLIS_BUFFER"	X$U24 out ref in SIMPLIS_BUFFER vars: IC=0 RIN=100Meg ROUT=1 TH=1.5 + HYSTWD=10m VOL=0 VOH=3 DELAY=12n	X_U24 in out ref BUF_DELAY_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH + DELAY=DELAY ROUT=ROUT .subckt BUF_DELAY_BASIC_GEN A Y REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 + DELAY=10n ROUT=1 E_ABMGATE1 YINT1 0 VALUE { {IF((V(A)-V(REF)) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT1 YINT2 1 CINT YINT2 REF {DELAY*1.3} E_ABMGATE2 YINT3 REF VALUE { {IF(V(YINT2) > {VTHRESH}, {VDD}, {VSS})} } RINT2 YINT3 Y {ROUT} CINT2 Y REF 1n .ENDS BUF_DELAY_BASIC_GEN	• VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal • DELAY: Delay (in seconds) can be represented in Farads at the output capacitor by multiplying it by 1.3. • ROUT: Output resistance	

INV gate	"SIMPLIS_INV"	X$U6 A 0 Y SIMPLIS_INV vars: IC=0 RIN=10Meg ROUT=10 TH=2.5 HYSTWD=100m + VOL=0 VOH=5 DELAY=0	X_U6 A Y INV_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .SUBCKT INV_BASIC_GEN A Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A)>{VTHRESH} , {VSS}, {VDD})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS INV_BASIC_GEN	• VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage. Used to determine high or low signal	

☑ One Shot	"SIMPLIS_1SHOT_BB"	X$U1 in nodex out nodey nodez SIMPLIS_1SHOT_BB vars: T_DUR=200n	X_U1 in out ONE_SHOT PARAMS: T=T_DUR VDD=VOH VSS=VOL VTHRESH=VTHRESH .subckt ONE_SHOT in out params: T=100n VDD=1 VSS=0 VTHRESH=0.5 s_s1 meas 0 reset2 0 s1 e_abm1 ch 0 value { if(v(in)>{VTHRESH} | v(out)>{VTHRESH}, {VDD}, {VSS}) } r_r2 reset2 reset 0.1 e_abm3 out 0 value { if(v(meas)<{VTHRESH} & v(ch)>{VTHRESH}, {VDD}, {VSS}) } r_r1 meas ch {(t/(1E-9))} c_c2 0 reset2 1.4427n c_c1 0 meas 1.4427n e_abm2 reset 0 value { if(v(ch)<{VTHRESH}, {VDD}, {VSS}) } .model s1 vswitch roff=1e+09 ron=1 voff={VTHRESH-0.01} von={VTHRESH+0.01} .ends ONE_SHOT	• Utilizes the find_H_L_THRESH() function to pull parameters VOH, VOL, and VTHRESH from another line in the netlist. • VOH: Logic high voltage • VOL: Logic low voltage • VTHRESH: Threshold voltage. Used to determine high or low signal • T_DUR: One shot pulse width	

VCVS with limiter	"SIMPLIS_VCVS_WITH_LIMITER"	X$U33 VOUT+ VOUT- VIN1 VIN2 SIMPLIS_VCVS_WITH_LIMITERX$U14 vars: GAIN=111m + MIN_OUTPUT=-100m MAX_OUTPUT=100m	E_ABM_33 VOUT+ VOUT- VALUE {LIMIT((GAIN*(V(VIN1)-V(VIN2))), MIN_OUTPUT, MAX_OUTPUT)}	• GAIN: Gain of the VCVS • MIN_OUTPUT: Minimum output limit. Anything under this value will be set to this value. • MAX_OUTPUT: Maximum output limit. Anything over this value will be set to this value.	
VCCS with limiter	"SIMPLIS_VCCS_WITH_LIMITER"	X$U10 vout+ vout- in1 in2 SIMPLIS_VCCS_WITH_LIMITERX$U10 vars: GAIN=760u + MIN_OUTPUT=-12u MAX_OUTPUT=36u	G_ABMI_U10 vout+ vout- VALUE { LIMIT(GAIN*(V(in1)-V(in2)), MIN_OUTPUT, MAX_OUTPUT) }	• GAIN: Gain of the VCCS • MIN_OUTPUT: Minimum output limit. Anything under this value will be set to this value. • MAX_OUTPUT: Maximum output limit. Anything over this value will be set to this value.	

☑ CCVS with limiter	"SIMPLIS_CCVS_WITH_LIMITER"	X$U21 vout+ vout- in1 in2 SIMPLIS_CCVS_WITH_LIMITERX$U21 vars: + GAIN={Rtrip_Gain/0.3} MIN_OUTPUT=4 MAX_OUTPUT=17.6	H_U21 h_out 0 VH_U21 GAIN VH_U21 in1 in2 0V E_ABM_U21 abm_out 0 VALUE { LIMIT(V(h_out), MAX_OUTPUT, MIN_OUTPUT) } E_U21 vout+ vout- abm_out 0 1	• GAIN: Gain of the CCVS • MIN_OUTPUT: Minimum output limit. Anything under this value will be set to this value. • MAX_OUTPUT: Maximum output limit. Anything over this value will be set to this value.	

☑ SR Latch with reference node	"SIMPLIS_DIGI1_SRLATCH_Y" or "SIMPLIS_LOGIC_BB_SRFF" or "SIMPLIS_SRFF"	X$U2 q qbar ref s r SIMPLIS_DIGI1_SRLATCH_Y vars: IC=0 RIN=10Meg ROUT=10 + TH=1.5 HYSTWD=1 VOL=0 VOH=3 OUT_DELAY=20p SET_RESET_LEVEL=1 + GNDREF='Y' DOM='S'	*S dominant X_U2 s r q qbar srlatchshp_basic_gen params: vdd=VOH vss=VOL vthresh=TH *R dominant X_U2 s r q qbar srlatchrhp_basic_gen params: vdd=VOH vss=VOL vthresh=TH *S dominant .subckt srlatchshp_basic_gen s r q qb params: vdd=1 vss=0 vthresh=0.5 gq 0 qint value = { if(v(s) > {vthresh}, 5, if(v(r)>{vthresh}, -5, 0)) } cqint qint 0 1n rqint qint 0 1000meg d_d10 qint my5 d_d1 v1 my5 0 {vdd} d_d11 myvss qint d_d1 v2 myvss 0 {vss} eq qqq 0 qint 0 1 x3 qqq qqqd1 BUF_BASIC_GEN PARAMS: VDD={vdd} VSS={vss} VTHRESH={vthresh} rqq qqqd1 q 1 eqb qbr 0 value = {if(v(q) > {vthresh}, {vss},{vdd})} rqb qbr qb 1 cdummy1 q 0 1n cdummy2 qb 0 1n	• DOM: Determines S or R dominance. S is default. Can also be presented as DomType or not given at all. This will determine which subcircuit definition is used, the one for S dominant or the one for R dominant. • VOH: Logic high value • VOL: Logic low value • TH: Threshold voltage	

			.ic v(qint) {vss} .model d_d1 d is=1e-15 tt=1e-11 rs=0.05 n=0.01 .ends srlatchshp_basic_gen *R dominant .subckt srlatchrhp_basic_gen s r q qb params: vdd=1 vss=0 vthresh=0.5 gq 0 qint value = { if(v(r) > {vthresh}, -5, if(v(s)>{vthresh}, 5, 0)) } cqint qint 0 1n rqint qint 0 1000meg d_d10 qint my5 d_d1 v1 my5 0 {vdd} d_d11 myvss qint d_d1 v2 myvss 0 {vss} eq qqq 0 qint 0 1 x3 qqq qqqd1 BUF_BASIC_GEN PARAMS: VDD={vdd} VSS={vss} VTHRESH={vthresh} rqq qqqd1 q 1 eqb qbr 0 value = { if(v(q) > {vthresh}, {vss}, {vdd}) } rqb qbr qb 1 cdummy1 q 0 1n cdummy2 qb 0 1n .ic v(qint) {vss} .model d_d1 d is=1e-15 tt=1e-11 rs=0.05 n=0.01		

			.ends srlatchrhp_basic_gen *This subckt was called in the subcircuits above .SUBCKT BUF_BASIC_GEN A Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A)>{VTHRESH} , {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS BUF_BASIC_GEN		

SR Latch without reference node	"SIMPLIS_DIGI1_SRLATCH_N"	X$U2 q qbar s r SIMPLIS_DIGI1_SRLATCH_Y vars: IC=0 RIN=10Meg ROUT=10 + TH=1.5 HYSTWD=1 VOL=0 VOH=3 OUT_DELAY=20p SET_RESET_LEVEL=1 + GNDREF='Y' DOM='S'	*S dominant X_U2 s r q qbar srlatchshp_basic_gen params: vdd=VOH vss=VOL vthresh=TH *R dominant X_U2 s r q qbar srlatchrhp_basic_gen params: vdd=VOH vss=VOL vthresh=TH *S dominant .subckt srlatchshp_basic_gen s r q qb params: vdd=1 vss=0 vthresh=0.5 gq 0 qint value = { if(v(s) > {vthresh}, 5, if(v(r)>{vthresh}, -5, 0)) } cqint qint 0 1n rqint qint 0 1000meg d_d10 qint my5 d_d1 v1 my5 0 {vdd} d_d11 myvss qint d_d1 v2 myvss 0 {vss} eq qqq 0 qint 0 1 x3 qqq qqqd1 BUF_BASIC_GEN PARAMS: VDD={vdd} VSS={vss} VTHRESH={vthresh} rqq qqqd1 q 1 eqb qbr 0 value = {if(v(q) > {vthresh}, {vss},{vdd})} rqb qbr qb 1 cdummy1 q 0 1n cdummy2 qb 0 1n	• DOM: Determines S or R dominance. S is default. This will determine which subcircuit definition is used, the one for S dominant or the one for R dominant. • VOH: Logic high value • VOL: Logic low value • TH: Threshold voltage	

			.ic v(qint) {vss} .model d_d1 d is=1e-15 tt=1e-11 rs=0.05 n=0.01 .ends srlatchshp_basic_gen *R dominant .subckt srlatchrhp_basic_gen s r q qb params: vdd=1 vss=0 vthresh=0.5 gq 0 qint value = { if(v(r) > {vthresh}, -5, if(v(s)>{vthresh}, 5, 0)) } cqint qint 0 1n rqint qint 0 1000meg d_d10 qint my5 d_d1 v1 my5 0 {vdd} d_d11 myvss qint d_d1 v2 myvss 0 {vss} eq qqq 0 qint 0 1 x3 qqq qqqd1 BUF_BASIC_GEN PARAMS: VDD={vdd} VSS={vss} VTHRESH={vthresh} rqq qqqd1 q 1 eqb qbr 0 value = { if(v(q) > {vthresh}, {vss}, {vdd}) } rqb qbr qb 1 cdummy1 q 0 1n cdummy2 qb 0 1n .ic v(qint) {vss} .model d_d1 d is=1e-15 tt=1e-11 rs=0.05 n=0.01		

			.ends srlatchrhp_basic_gen *This subckt was called in the subcircuits above .SUBCKT BUF_BASIC_GEN A Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 + DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A)>{VTHRESH} , {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS BUF_BASIC_GEN		

D Flip Flop with reference node	"SIMPLIS_DIGI1_DFF_Y" or "SIMPLIS_DFF"	X$U29 q qb ref d clk SIMPLIS_DIGI1_DFF_Y vars: IC=0 RIN=10Meg ROUT=10 TH=1.5 + HYSTWD=1 VOL=0 VOH=3 MIN_CLK=10p TRIG_COND='0_TO_1' + CLK_TO_OUT_DELAY=20p SETUP_TIME=10p HOLD_TIME=1p GNDREF='Y'	X_U29_Buf d d_delayed 0 BUF_DELAY_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH X_U29 q qb clk d_delayed 0 0 DFF1SR_RHPBASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .subckt dff1sr_rhpbasic_gen q qb clk d r s params: vdd=1 vss=0 vthresh=0.5 x1 clk clkdel INV_DELAY_BASIC_GEN params: vdd={vdd} vss={vss} vthresh={vthresh} delay=10n x2 clk clkdel clkint AND2_BASIC_GEN params: vdd={vdd} vss={vss} vthresh={vthresh} gq 0 qint value = {if(v(r) > {vthresh},-5,if(v(s) > {vthresh}, 5, if(v(clkint)>{vthresh}, if(v(d)> {vthresh}, 5, -5), 0))))} cqint qint 0 1n rqint qint 0 1000meg d_d10 qint my5 d_d1 v1 my5 0 {vdd} d_d11 myvss qint d_d1 v2 myvss 0 {vss} eq qqq 0 qint 0 1	• VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage	

			v3 qqq qqqd1 rqq qqqd1 q 1 eqb qbr 0 value = {if(v(q) > {vthresh}, {vss},{vdd})} rqb qbr qb 1 cdummy1 q 0 1nf cdummy2 qb 0 1nf .ic v(qint) {vss} .model d_d1 d is=1e-15 tt=1e-11 rs=0.05 n=0.01 .ends dff1sr_rhpbasic_gen .subckt BUF_DELAY_BASIC_GEN A Y REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 DELAY=10n ROUT=1 E_ABMGATE1 YINT1 0 VALUE { {IF((V(A)-V(REF)) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT1 YINT2 1 CINT YINT2 REF {DELAY*1.3} E_ABMGATE2 YINT3 REF VALUE { {IF(V(YINT2) > {VTHRESH}, {VDD}, {VSS})} } RINT2 YINT3 Y {ROUT} CINT2 Y REF 1n .ENDS BUF_DELAY_BASIC_GEN		

			.subckt INV_DELAY_BASIC_GEN A Y REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 DELAY=10n ROUT=1 E_ABMGATE1 YINT1 0 VALUE { {IF((V(A)-V(REF)) > {VTHRESH}, {VSS}, {VDD})} } RINT YINT1 YINT2 1 CINT YINT2 REF {DELAY*1.3} E_ABMGATE2 YINT3 REF VALUE { {IF(V(YINT2) > {VTHRESH}, {VDD}, {VSS})} } RINT2 YINT3 Y {ROUT} CINT2 Y REF 1n .ENDS INV_DELAY_BASIC_GEN .SUBCKT AND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS AND2_BASIC_GEN		

D Flip Flop without reference node	"SIMPLIS_DIGI1_DFF_N"	X$U29 q qb d clk SIMPLIS_DIGI1_DFF_Y vars: IC=0 RIN=10Meg ROUT=10 TH=1.5 + HYSTWD=1 VOL=0 VOH=3 MIN_CLK=10p TRIG_COND='0_TO_1' + CLK_TO_OUT_DELAY=20p SETUP_TIME=10p HOLD_TIME=1p GNDREF='Y'	X_U29_Buf d d_delayed 0 BUF_DELAY_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH X_U29 q qb clk d_delayed 0 0 DFF1SR_RHPBASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .subckt dff1sr_rhpbasic_gen q qb clk d r s params: vdd=1 vss=0 vthresh=0.5 x1 clk clkdel INV_DELAY_BASIC_GEN params: vdd={vdd} vss={vss} vthresh={vthresh} delay=10n x2 clk clkdel clkint AND2_BASIC_GEN params: vdd={vdd} vss={vss} vthresh={vthresh} gq 0 qint value = {if(v(r) > {vthresh},-5,if(v(s) > {vthresh}, 5, if(v(clkint)>{vthresh}, if(v(d)>{vthresh}, 5, -5), 0))))} cqint qint 0 1n rqint qint 0 1000meg d_d10 qint my5 d_d1 v1 my5 0 {vdd} d_d11 myvss qint d_d1 v2 myvss 0 {vss} eq qqq 0 qint 0 1	• VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage	

			v3 qqq qqqd1 rqq qqqd1 q 1 eqb qbr 0 value = {if(v(q) > {vthresh}, {vss},{vdd})} rqb qbr qb 1 cdummy1 q 0 1nf cdummy2 qb 0 1nf .ic v(qint) {vss} .model d_d1 d is=1e-15 tt=1e-11 rs=0.05 n=0.01 .ends dff1sr_rhpbasic_gen .subckt BUF_DELAY_BASIC_GEN A Y REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 DELAY=10n ROUT=1 E_ABMGATE1 YINT1 0 VALUE { {IF((V(A)-V(REF)) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT1 YINT2 1 CINT YINT2 REF {DELAY*1.3} E_ABMGATE2 YINT3 REF VALUE { {IF(V(YINT2) > {VTHRESH}, {VDD}, {VSS})} } RINT2 YINT3 Y {ROUT} CINT2 Y REF 1n .ENDS BUF_DELAY_BASIC_GEN		

			.subckt INV_DELAY_BASIC_GEN A Y REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 DELAY=10n ROUT=1 E_ABMGATE1 YINT1 0 VALUE { {IF((V(A)-V(REF)) > {VTHRESH}, {VSS}, {VDD})} } RINT YINT1 YINT2 1 CINT YINT2 REF {DELAY*1.3} E_ABMGATE2 YINT3 REF VALUE { {IF(V(YINT2) > {VTHRESH}, {VDD}, {VSS})} } RINT2 YINT3 Y {ROUT} CINT2 Y REF 1n .ENDS INV_DELAY_BASIC_GEN .SUBCKT AND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS AND2_BASIC_GEN		

☑ DFF with S, R, and reference nodes	"SIMPLIS_DIGI1_DFF_SR_Y"	X$U4 q qb ref d clk s r SIMPLIS_DIGI1_DFF_SR_Y vars: IC=0 RIN=10Meg ROUT=10 + TH=2.5 HYSTWD=1 VOL=0 VOH=5 MIN_CLK=200p TRIG_COND='0_TO_1' + CLK_TO_OUT_DELAY=500p SETUP_TIE=10p HOLD_TIME=1p SET_RESET_DELAY=200p + SET_RESET_TYPE='ASYNC' SET_RESET_LEVEL=0 GNDREF='Y'	X_U29_Buf d d_delayed 0 BUF_DELAY_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH X_U29 q qb clk d_delayed r s DFF1SR_RHPBASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .subckt dff1sr_rhpbasic_gen q qb clk d r s params: vdd=1 vss=0 vthresh=0.5 x1 clk clkdel INV_DELAY_BASIC_GEN params: vdd={vdd} vss={vss} vthresh={vthresh} delay=10n x2 clk clkdel clkint AND2_BASIC_GEN params: vdd={vdd} vss={vss} vthresh={vthresh} gq 0 qint value = {if(v(r) > {vthresh},-5,if(v(s) > {vthresh}, 5, if(v(clkint)>{vthresh}, if(v(d)>{vthresh}, 5, -5), 0)))} cqint qint 0 1n rqint qint 0 1000meg d_d10 qint my5 d_d1 v1 my5 0 {vdd} d_d11 myvss qint d_d1 v2 myvss 0 {vss} eq qqq 0 qint 0 1 v3 qqq qqqd1	• VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage	

			rqq qqqd1 q 1 eqb qbr 0 value = {if(v(q) > {vthresh}, {vss},{vdd})} rqb qbr qb 1 cdummy1 q 0 1nf cdummy2 qb 0 1nf .ic v(qint) {vss} .model d_d1 d is=1e-15 tt=1e-11 rs=0.05 n=0.01 .ends dff1sr_rhpbasic_gen .subckt BUF_DELAY_BASIC_GEN A Y REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 DELAY=10n ROUT=1 E_ABMGATE1 YINT1 0 VALUE { {IF((V(A)-V(REF)) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT1 YINT2 1 CINT YINT2 REF {DELAY*1.3} E_ABMGATE2 YINT3 REF VALUE { {IF(V(YINT2) > {VTHRESH}, {VDD}, {VSS})} } RINT2 YINT3 Y {ROUT} CINT2 Y REF 1n .ENDS BUF_DELAY_BASIC_GEN		

			.subckt INV_DELAY_BASIC_GEN A Y REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 DELAY=10n ROUT=1 E_ABMGATE1 YINT1 0 VALUE { {IF((V(A)-V(REF)) > {VTHRESH}, {VSS}, {VDD})} } RINT YINT1 YINT2 1 CINT YINT2 REF {DELAY*1.3} E_ABMGATE2 YINT3 REF VALUE { {IF(V(YINT2) > {VTHRESH}, {VDD}, {VSS})} } RINT2 YINT3 Y {ROUT} CINT2 Y REF 1n .ENDS INV_DELAY_BASIC_GEN .SUBCKT AND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS AND2_BASIC_GEN		

☑ DFF with S, R nodes and no reference node	"SIMPLIS_DIGI1_DFF_SR_N"	X$U4 q qb d clk s r SIMPLIS_DIGI1_DFF_SR_N vars: IC=0 RIN=10Meg ROUT=10 + TH=2.5 HYSTWD=1 VOL=0 VOH=5 MIN_CLK=200p TRIG_COND='0_TO_1' + CLK_TO_OUT_DELAY=500p SETUP_TIE=10p HOLD_TIME=1p SET_RESET_DELAY=200p + SET_RESET_TYPE='ASYNC' SET_RESET_LEVEL=0 GNDREF='Y'	X_U29_Buf d d_delayed 0 BUF_DELAY_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH X_U29 q qb clk d_delayed r s DFF1SR_RHPBASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .subckt dff1sr_rhpbasic_gen q qb clk d r s params: vdd=1 vss=0 vthresh=0.5 x1 clk clkdel INV_DELAY_BASIC_GEN params: vdd={vdd} vss={vss} vthresh={vthresh} delay=10n x2 clk clkdel clkint AND2_BASIC_GEN params: vdd={vdd} vss={vss} vthresh={vthresh} gq 0 qint value = {if(v(r) > {vthresh},-5,if(v(s) > {vthresh}, 5, if(v(clkint)>{vthresh}, if(v(d)>{vthresh}, 5, -5), 0)))} cqint qint 0 1n rqint qint 0 1000meg d_d10 qint my5 d_d1 v1 my5 0 {vdd} d_d11 myvss qint d_d1 v2 myvss 0 {vss} eq qqq 0 qint 0 1 v3 qqq qqqd1	• VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage	

			rqq qqqd1 q 1 eqb qbr 0 value = {if(v(q) > {vthresh}, {vss},{vdd})} rqb qbr qb 1 cdummy1 q 0 1nf cdummy2 qb 0 1nf .ic v(qint) {vss} .model d_d1 d is=1e-15 tt=1e-11 rs=0.05 n=0.01 .ends dff1sr_rhpbasic_gen .subckt BUF_DELAY_BASIC_GEN A Y REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 DELAY=10n ROUT=1 E_ABMGATE1 YINT1 0 VALUE { {IF((V(A)-V(REF)) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT1 YINT2 1 CINT YINT2 REF {DELAY*1.3} E_ABMGATE2 YINT3 REF VALUE { {IF(V(YINT2) > {VTHRESH}, {VDD}, {VSS})} } RINT2 YINT3 Y {ROUT} CINT2 Y REF 1n .ENDS BUF_DELAY_BASIC_GEN			

			.subckt INV_DELAY_BASIC_GEN A Y REF PARAMS: VDD=1 VSS=0 VTHRESH=0.5 DELAY=10n ROUT=1 E_ABMGATE1 YINT1 0 VALUE { {IF((V(A)-V(REF)) > {VTHRESH}, {VSS}, {VDD})} } RINT YINT1 YINT2 1 CINT YINT2 REF {DELAY*1.3} E_ABMGATE2 YINT3 REF VALUE { {IF(V(YINT2) > {VTHRESH}, {VDD}, {VSS})} } RINT2 YINT3 Y {ROUT} CINT2 Y REF 1n .ENDS INV_DELAY_BASIC_GEN .SUBCKT AND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS AND2_BASIC_GEN		

Up Counter	"SIMPLIS_DIGI1_D_UP_COUNTER"	X$U26 d0 d1 d2 d3 clk en reset ref SIMPLIS_DIGI1_D_UP_COUNTER_YX$U26 vars: + RIN=10Meg TH=1.5 HYSTWD=1 ROUT=10 VOL=0 VOH=3 MIN_CLK=10p + TRIG_COND='0_TO_1' CLK_TO_OUT_DELAY=20p SETUP_TIME=10p HOLD_TIME=1p + ENABLE_DELAY=10p RESET_DELAY=15p RESET_TYPE='ASYNC' RESET_LEVEL=1 + RESET_TO=-1 IC=0	X_U26 clk count_fe counter_out gnd reset rst_int COUNTER PARAMS: MIN_PW=10N COUNT_RST=16 E_ABM_U26 d4 0 VALUE { if(V(counter_out)>15.5,VOH,VOL) } .SUBCKT Counter CLK COUNT_FE COUNT_RE GND RESET RST_INT PARAMS: MIN_PW=10n COUNT_RST=128 R_R4 N16838390 N16838391 {MIN_PW/2} R_R5 N16842332 CLKRE 1 C_C5 GND CLKRE 1n R_R3 RST_INT N16832011 1 E_ABM4 N168132710 VALUE { IF(V(RESET)>0.5 | V(RST_INT)>0.5, 0, IF(V(CLKRE)>0.55, + V(COUNT_FE)+1, V(COUNT_RE))) } R_R6 N168433471 CLKI 1 E_ABM6 N168320110 VALUE { if(v(COUNT_RE) < 10m,0,if(v(COUNT_FE)> + {COUNT_RST}-0.1, 1, V(RST_INT))) } C_C1 GND COUNT_RE_INT 1n IC=0 C_C6 GND CLKI 1n	• Available cases: Only in the case of 5bit output and when the fifth bit is what is used. • VOH: Logic high voltage used to set logic high output • VOL: Logic low voltage used to set logic low output • Can be given with or without reference node. The ref node does not affect other nodes indexes so the logic can handle both cases.	• The script only places an ABM to do analog to digital conversion on the 5th bit right now. This was what was used by Justin Jr. • The counter can likely be used for applications beyond 5 bits but an ADC would need to be set up for all bits at output to be fully functional.

			C_C3 GND RST_INT 1n IC=0 E_ABM10 N168433471 0 VALUE { IF(V(CLK)<0.5,1,0) } R_R1 COUNT_FE_INT N33733 1 E_ABM8 N168407060 VALUE { IF(V(RST_INT)<0.5, 1,0) } C_C4 GND N16838391 1.443n E_E4 COUNT_RE GND COUNT_RE_INT GND 1 R_R2 COUNT_RE_INT N16813271 1 E_ABM2 N33733 0 VALUE { IF(V(RESET)>0.5 | V(RST_INT)>0.5, 0,IF(V(CLKI)>0.55, + V(COUNT_RE_INT), V(COUNT_FE))) } E_E3 COUNT_FE GND COUNT_FE_INT GND 1 C_C2 GND COUNT_FE_INT 1n IC=0 E_ABM7 N168383900 VALUE { IF(V(CLK)<0.5,1,0) } E_ABM9 N168423320 VALUE { IF(V(N16838391)> 0.5,0,IF(V(CLK)>0.5 & + V(N16840706)>0.5, 1, 0)) } .ENDS Counter		

☑ Up/Down Counter	“SIMPLIS_DIGI1_D_UPDOWN_COUNTER”	X$U16 d0 d1 d2 clk en set reset cnt_up <ref> + SIMPLIS_DIGI1_D_UPDOWN_COUNTER_YX$U16 vars: RIN=10Meg TH=1 + HYSTWD=100m ROUT=10 VOL=0 VOH=5 MIN_CLK=10p TRIG_COND='0_TO_1' + CLK_TO_OUT_DELAY=20p SETUP_TIME=10p HOLD_TIME=1p ENABLE_DELAY=10p + SET_RESET_DELAY=15p SET_RESET_TYPE='ASYNC' SET_RESET_LEVEL=1 SET_TO=-1 + RESET_TO=-1 IC=0	X_U16 clk count_fe_U16 counter_out_U16 ref reset rst_int_U16 Counter PARAMS: COUNT_RST=8 X_U16_ADC d0 d1 d2 counter_out_U160 en ADC_3B_LOGIC PARAMS: VDD=VOH VSS=VOL THRESH=TH count_fe and rst_int are unused nodes .SUBCKT Counter CLK COUNT_FE COUNT_RE GND RESET RST_INT PARAMS: MIN_PW=10n COUNT_RST=128 R_R4 N16838390 N16838391 {MIN_PW/2} R_R5 N16842332 CLKRE 1 C_C5 GND CLKRE 1n R_R3 RST_INT N16832011 1 E_ABM4 N168132710 VALUE { IF(V(RESET)>0.5 | V(RST_INT)>0.5, 0, IF(V(CLKRE)>0.55, V(COUNT_FE)+1, V(COUNT_RE))) } R_R6 N168433471 CLKI 1 E_ABM6 N168320110 VALUE { if(v(COUNT_RE) < 10m,0,if(v(COUNT_FE)> {COUNT_RST}-0.1, 1, V(RST_INT))) }	• Available cases: Only set up to work with 3bits right now. • The script can calculate the number of bits for the counter. COUNT_RST value is set by 2^NUM_BITS. • VOH: Logic high voltage used to set logic high output • VOL: Logic low voltage used to set logic low output • Can be given with or without reference node. The ref node does not affect other nodes indexes so the logic can handle both cases.	• The logic can handle 3 bits right now, where it attaches a 3 bit ADC to the output of the counter to produce the 3 bits. • This method could be applied to other bit numbers.

			C_C1 GND COUNT_RE_INT 1n IC=0 C_C6 GND CLKI 1n C_C3 GND RST_INT 1n IC=0 E_ABM10 N168433471 0 VALUE { IF(V(CLK)<0.5,1,0) } R_R1 COUNT_FE_INT N33733 1 E_ABM8 N168407060 VALUE { IF(V(RST_INT)<0.5, 1,0) } C_C4 GND N16838391 1.443n E_E4 COUNT_RE GND COUNT_RE_INT GND 1 R_R2 COUNT_RE_INT N16813271 1 E_ABM2 N33733 0 VALUE { IF(V(RESET)>0.5 | V(RST_INT)>0.5, 0,IF(V(CLKI)>0.55, V(COUNT_RE_INT), V(COUNT_FE))) } E_E3 COUNT_FE GND COUNT_FE_INT GND 1 C_C2 GND COUNT_FE_INT 1n IC=0 E_ABM7 N168383900 VALUE { IF(V(CLK)<0.5,1,0) }		

			E_ABM9 N168423320 VALUE { IF(V(N16838391)>0.5,0,IF(V(CLK)>0.5 & V(N16840706)>0.5, 1, 0)) } .ENDS Counter .SUBCKT ADC_3B_LOGIC BIT0 BIT1 BIT2 VINP VINN EN PARAMS: REFP=8 VDD=1 VSS=0 THRESH=0.5 E_BIT0 BIT0temp 0 VALUE = {IF((V(VINP)-V(VINN))>=({REFP}/2-0.5) & V(EN)>{THRESH}, {VDD}, {VSS})} R1 BIT0temp BIT0 1 C1 BIT0 0 1p E_BIT1 BIT1temp VSS VALUE = {IF((V(VINP)-V(VINN)-V(BIT0temp)*{REFP}/2)>=(V(REFP)/4-0.5) & V(EN)>{THRESH}, {VDD}, {VSS})} R2 BIT1temp BIT1 1 C2 BIT1 0 1p E_BIT2 BIT2temp VSS VALUE = {IF((V(VINP)-V(VINN)-V(BIT0temp)*{REFP}/2-V(BIT1temp)*{REFP}/4)>=({REFP}/8-0.5) & V(EN)>{THRESH}, {VDD}, {VSS})} R3 BIT2temp BIT2 1 C3 BIT2 0 1p		

			.ends ADC_3B_LOGIC		

☑ Asymmetric Delay with reference node	"SIMPLIS_DIGI1_D_ASYMMETRIC_DELAY_Y"	X$U1 out ref in SIMPLIS_DIGI1_D_ASYMMETRIC_DELAY_Y vars: IC=0 RIN=1T ROUT=1m + TH=1.5 HYSTWD=1m VOL=0 VOH=3 RISE_DELAY=68u FALL_DELAY=1u GNDREF='Y'	E_U1_in del_in 0 in ref 1 X_U1 del_in del_out ASYMMETRIC_DELAY PARAMS: RISING_EDGE_DELAY=RISE_DELAY + VTHRESH=TH FALLING_EDGE_DELAY=FALL_DELAY VDD=VOH VSS=VOL E_U1_out out ref del_out 0 1 .subckt ASYMMETRIC_DELAY inp out PARAMS: RISING_EDGE_DELAY=1 VTHRESH=0.5 + FALLING_EDGE_DELAY=1 VDD=1 VSS=0 e_abm3 inp1 0 value { if(v(inp) > {vthresh}, {vdd} , {vss}) } e_abm1 yin4 0 value { if(v(yin3) > {vthresh}, {vdd} , {vss}) } e_abm2 yin2 0 value { if(v(yin1) > {vthresh}, {vdd} , {vss}) } r_rint inp1 yin1 1 c_cint yin1 0 {1.443*rising_edge_delay} d_d10 yin1 inp1 d_d1 r_r1 yin4 out 1 r_rout yin2 yin3 1 c_cout yin3 0 {1.443*falling_edge_delay} c_c1 0 out 1n	• RISE_DELAY: Rise time • FALL_DELAY: Fall time • TH: Threshold voltage • VOL: Logic low voltage • VOH: Logic high voltage	

			d_d11 yin2 yin3 d_d1 .model d_d1 d is=1e-15 tt=1e-11 rs=0.005 n=0.1 .ends asymmetric_delay		

☑ Asymmetric Delay without reference node	"SIMPLIS_DIGI1_D_ASYMMETRIC_DELAY_N"	X$U1 out in SIMPLIS_DIGI1_D_ASYMMETRIC_DELAY_N vars: IC=0 RIN=1T ROUT=1m + TH=1.5 HYSTWD=1m VOL=0 VOH=3 RISE_DELAY=68u FALL_DELAY=1u GNDREF='Y'	E_U1_in del_in 0 in ref 1 X_U1 del_in del_out ASYMMETRIC_DELAY PARAMS: RISING_EDGE_DELAY=RISE_DELAY + VTHRESH=TH FALLING_EDGE_DELAY=FALL_DELAY VDD=VOH VSS=VOL E_U1_out out ref del_out 0 1 .subckt ASYMMETRIC_DELAY inp out PARAMS: RISING_EDGE_DELAY=1 VTHRESH=0.5 + FALLING_EDGE_DELAY=1 VDD=1 VSS=0 e_abm3 inp1 0 value { if(v(inp) > {vthresh}, {vdd} , {vss}) } e_abm1 yin4 0 value { if(v(yin3) > {vthresh}, {vdd} , {vss}) } e_abm2 yin2 0 value { if(v(yin1) > {vthresh}, {vdd} , {vss}) } r_rint inp1 yin1 1 c_cint yin1 0 {1.443*rising_edge_delay} d_d10 yin1 inp1 d_d1 r_r1 yin4 out 1 r_rout yin2 yin3 1 c_cout yin3 0 {1.443*falling_edge_delay} c_c1 0 out 1n	• RISE_DELAY: Rise time • FALL_DELAY: Fall time • TH: Threshold voltage • VOL: Logic low voltage • VOH: Logic high voltage	

			d_d11 yin2 yin3 d_d1 .model d_d1 d is=1e-15 tt=1e-11 rs=0.005 n=0.1 .ends asymmetric_delay		

Digital Multiplexer	"SIMPLIS_DIGI1_D_MUX"	X$U18 out <ref> s0 s1 a0 b0 c0 d0 SIMPLIS_DIGI1_D_MUX_NX$U18 vars: + OUT_DELAY=2p NUM_INPUTS=4 NUMBITS_OUT=1 RIN=10Meg ROUT=10 TH=2.5 + HYSTWD=1 VOL=0 VOH=5 INVERSION='N' IC=0	X_U18 out ref s0 s1 a0 b0 c0 d0 D_MUX_4IN_1B PARAMS: THRESH=TH VOH=VOH VOL=VOL .subckt D_MUX_4IN_1B out ref s0 s1 a0 b0 c0 d0 PARAMS: THRESH=0.5 VOH=1 VOL=0 E_ABM1 1 0 VALUE { IF(V(s0)>{THRESH} & V(s1)>{THRESH}, V(d0), IF(V(s0)<{THRESH} & + V(s1)>{THRESH}, V(c0), IF(V(s0)>{THRESH} & V(s1)<{THRESH}, V(b0), V(a0)))) } E_ABM2 out ref VALUE { IF(V(1)>({THRESH}, {VOH}, {VOL}) } .ends D_MUX_4IN_1B	• Available cases: NUM_INPUTS=2/NUMBITS=1, NUM_INPUTS=4/NUMBITS=1 • This can be given with or without a reference node, it doesn't affect translation. • NUM_INPUTS: Number of inputs • NUMBITS_OUT: Number of bits of output • INVERSION: If INVERSION='Y' then the output will be inverted • TH: Threshold voltage • VOH: Logic high voltage • VOL: Logic low voltage	

☑ Digital Demultiplexer	"SIMPLIS_DIGI1_D_DEMUX"	X$U4 a0 b0 c0 d0 e0 f0 g0 h0 s0 s1 s2 i0 rtn SIMPLIS_DIGI1_D_DEMUX_YX$U4 vars: + OUT_DELAY=2p NUM_OUTPUTS=8 NUMBITS_IN=1 RIN=10Meg ROUT=10 TH=1 + HYSTWD=100m VOL=0 VOH=5 INVERSION='N' INACTIVE_LEVEL=0 IC=1	X_U4 a0 b0 c0 d0 e0 f0 g0 h0 s0 s1 s2 i0 rtn DIGI_DEMUX_8OUT_1B PARAMS: + THRESH=TH VOH=VOH VOL=VOL .subckt DIGI_DEMUX_8OUT_1B a0 b0 c0 d0 e0 f0 g0 h0 s0 s1 s2 i0 ref PARAMS: + THRESH=0.5 VOH=1 VOL=0 E_ABM1 1 0 VALUE { IF(V(s2)<{THRESH} & V(s1)<{THRESH} & V(s0)<{THRESH}, V(i0), 0) } E_ABM2 2 0 VALUE { IF(V(s2)<{THRESH} & V(s1)<{THRESH} & V(s0)>{THRESH}, V(i0), 0) } E_ABM_B b0 ref VALUE { IF(V(2)>{THRESH}, {VOH}, {VOL}) } E_ABM3 3 0 VALUE { IF(V(s2)<{THRESH} & V(s1)>{THRESH} & V(s0)<{THRESH}, V(i0), 0) } E_ABM_C c0 ref VALUE { IF(V(3)>{THRESH}, {VOH}, {VOL}) } E_ABM4 4 0 VALUE { IF(V(s2)<{THRESH} & V(s1)>{THRESH} & V(s0)>{THRESH}, V(i0), 0) } E_ABM_D d0 ref VALUE { IF(V(4)>{THRESH}, {VOH}, {VOL}) }	• Available cases: NUM_OUTPUTS=8 / NUMBITS=1 • This can be given with or without a reference node, it doesn't affect translation. • NUM_OUTPUTS: Number of outputs • NUMBITS_IN: Number of bits of input • INVERSION: If INVERSION='Y' then the input will be inverted • TH: Threshold voltage • VOH: Logic high voltage • VOL: Logic low voltage	

			E_ABM5 5 0 VALUE { IF(V(s2)>{THRESH} & V(s1)<{THRESH} & V(s0)<{THRESH}, V(i0), 0) } E_ABM_E e0 ref VALUE { IF(V(5)>{THRESH}, {VOH}, {VOL}) } E_ABM6 6 0 VALUE { IF(V(s2)>{THRESH} & V(s1)<{THRESH} & V(s0)>{THRESH}, V(i0), 0) } E_ABM_F f0 ref VALUE { IF(V(6)>{THRESH}, {VOH}, {VOL}) } E_ABM7 7 0 VALUE { IF(V(s2)>{THRESH} & V(s1)>{THRESH} & V(s0)<{THRESH}, V(i0), 0) } E_ABM_G g0 ref VALUE { IF(V(7)>{THRESH}, {VOH}, {VOL}) } E_ABM8 8 0 VALUE { IF(V(s2)>{THRESH} & V(s1)>{THRESH} & V(s0)>{THRESH}, V(i0), 0) } E_ABM_H h0 ref VALUE { IF(V(8)>{THRESH}, {VOH}, {VOL}) } .ends DIGI_DEMUX_8OUT_1B		

☑ Square Source/ Step Source	"SQU_SOURCE" or "STEP_SOURCE"	X$V2 out ref SQU_SOURCE vars: _V1=0 _V2=5 _FREQ=1Meg _DELAY=0 + _DRATIOx100=50 _T_RISE=0 _T_FALL=0 _DAMP_COEF=0 _PWIDTH=500n + _OFF_UNTIL_DELAY=0 _IDLE_IN_POP=0 _USEPHASE=0 _PHASE=0	V_U20 out ref PULSE _V1 _V2 _OFF_UNTIL_DELAY _T_RISE _T_FALL _PWIDTH <1/ _FREQ>	• _V1: Voltage 1 • _V2: Voltage 2 • _OFF_UNTIL_DELAY: The pulse will not rise to Voltage 2 until this time has passed. • _T_RISE: Rise time • _T_FALL: Fall time • _PWIDTH: Duration of Voltage 2 pulse • _FREQ: Frequency is used to calculate period in PSPICE. Period=<1/_FREQ>	

☑ Digital Pulse	"SIMPLIS_DIGI1_D_PULSE"	X$U20 out <out_not> <ref> SIMPLIS_DIGI1_D_PULSE_N_Y vars: PERIOD={1/4Meg} + WIDTH={1/(2*4Meg)} DELAY=0 ROUT=10 VOL=0 VOH=5 COMP='N' GNDREF='Y'	V_U20 out ref PULSE VOL VOH DELAY 0 0 WIDTH PERIOD	• VOL: Voltage 1 • VOH: Voltage 2 • COMP: Complement output node given or not. If COMP='Y', an INV is added to output. • GNDREF: Reference node given or not. Can handle both cases. • DELAY: The pulse will not rise to Voltage 2 until this time has passed. • WIDTH: Duration of Voltage 2 pulse • PERIOD: Duration of full Voltage 1 and 2 cycle	

Analog to Digital Converter	"SIMPLIS_DIGI1_D_A2D_CONVERTER"	X$U11 d0 d1 d2 d3 d4 d5 d6 d7 POFL NOFL DR IN CLK EN RTN + SIMPLIS_DIGI1_D_A2D_CONVERTER_YX $U11 vars: RIN=10Meg ROUT=10 TH=1 + HYSTWD=100m VOL=0 VOH=5 RANGE=128m OFFSET=64m CODE='UNSIGNED' + MIN_CLK=10p TRIG_COND='0_TO_1' SAMPLE_DELAY=1p ENABLE_DELAY=15p + CONVERT_TIME=50p DATA_READY_DELAY=10p IC=5 IC_OFL=0 IC_DATA_READY=1	X_U11 d0 d1 d2 d3 d4 d5 d6 d7 IN RTN EN ADC_8B_LOGIC PARAMS: VDD=VOH VSS=VOL THRESH=TH .SUBCKT ADC_8B_LOGIC BIT0 BIT1 BIT2 BIT3 BIT4 BIT5 BIT6 BIT7 VINP VINN EN + PARAMS: REFP=256 VDD=1 VSS=0 THRESH=0.5 E_BIT0 BIT0temp 0 VALUE = {IF((V(VINP)-V(VINN))>=({REFP}/2-0.5) & + V(EN)>{THRESH}, {VDD}, {VSS})} R1 BIT0temp BIT0 1 C1 BIT0 0 1p E_BIT1 BIT1temp VSS VALUE = {IF((V(VINP)-V(VINN)- + V(BIT0temp)*{REFP}/2)>=(V(REFP)/4-0.5) & V(EN)>{THRESH}, {VDD}, {VSS})} R2 BIT1temp BIT1 1 C2 BIT1 0 1p E_BIT2 BIT2temp VSS VALUE = {IF((V(VINP)-V(VINN)-V(BIT0temp)*{REFP}/2-	• Available cases: 8bit output • VOL: Logic low voltage • VOH: Logic high voltage • TH: Threshold voltage	• Currently the script only handles 8bit ADC. The logic for 3bit ADC is provided in the Up/Down Counter but has not been added to this logic yet. • ADC should be scaled for other numbers of bits.

			+ V(BIT1temp)*{REFP}/4)>=({REFP}/8-0.5) & V(EN)>{THRESH}, {VDD}, {VSS})} R3 BIT2temp BIT2 1 C3 BIT2 0 1p E_BIT3 BIT3temp VSS VALUE = {IF((V(VINP)-V(VINN)-V(BIT0temp)*V{REFP}/2- + V(BIT1temp)*{REFP}/4 - V(BIT2temp)*{REFP}/8) + >=({REFP}/16-0.5) & V(EN)>{THRESH}, {VDD}, {VSS})} R4 BIT3temp BIT3 1 C4 BIT3 0 1p E_BIT4 BIT4temp VSS VALUE = {IF((V(VINP)-V(VINN)-V(BIT0temp)*{REFP}/2- + V(BIT1temp)*{REFP}/4 - V(BIT2temp)*{REFP}/8 - + V(BIT3temp)*{REFP}/16)>=({REFP}/32-0.5) & V(EN)>{THRESH}, {VDD}, {VSS})} R5 BIT4temp BIT4 1 C5 BIT4 0 1p		

			E_BIT5 BIT5temp VSS VALUE = {IF((V(VINP)-V(VINN)-V(BIT0temp)*{REFP}/2-+ + V(BIT1temp)*{REFP}/4 - V(BIT2temp)*{REFP}/8) + - V(BIT3temp)*{REFP}/16 - V(BIT4temp)*{REFP}/32>={REFP}/64-0.5) & + V(EN)>{THRESH}, {VDD}, {VSS})} R6 BIT5temp BIT5 1 C6 BIT5 0 1p E_BIT6 BIT6temp VSS VALUE = {IF((V(VINP)-V(VINN)-V(BIT0temp)*{REFP}/2- + V(BIT1temp)*{REFP}/4 - V(BIT2temp)*{REFP}/8) + - V(BIT3temp)*{REFP}/16 - V(BIT4temp)*{REFP}/32 – + V(BIT5temp)*{REFP}/64>=({REFP}/128-0.5) & V(EN)>{THRESH}, {VDD}, {VSS})} R7 BIT6temp BIT6 1 C7 BIT6 0 1p		

			E_BIT7 BIT7temp VSS VALUE = {IF((V(VINP)-V(VINN)-V(BIT0temp)*{REFP}/2- + V(BIT1temp)*{REFP}/4 - V(BIT2temp)*{REFP}/8) + - V(BIT3temp)*{REFP}/16 - V(BIT4temp)*{REFP}/32 - V(BIT5temp)*{REFP}/64 – + V(BIT6temp)*{REFP}/128>=({REFP}/256-0.5) & V(EN)>{THRESH}, {VDD}, {VSS})} R8 BIT7temp BIT7 1 C8 BIT7 0 1p .ENDS ADC_8B_LOGIC		

☑ Digital to Analog Converter	"SIMPLIS_DIGI1_D_D2A_CONVERTER"	X$U1 out d0 d1 d2 d3 d4 d5 d6 d7 d8 <ref> SIMPLIS_DIGI1_D_D2A_CONVERTER_YX$U1	X_U1 out d0 d1 d2 d3 d4 d5 d6 d7 d8 ref D2A_CONVERTER_9B PARAMS: + THRESH=THRESH .subckt D2A_CONVERTER_9B out d0 d1 d2 d3 d4 d5 d6 d7 d8 ref PARAMS: + THRESH=0.5 E_ABM0 1 0 VALUE { IF(V(d0)>{THRESH}, 1, 0) } E_ABM1 2 0 VALUE { IF(V(d1)>{THRESH}, 2, 0) } E_ABM2 3 0 VALUE { IF(V(d2)>{THRESH}, 4, 0) } E_ABM3 4 0 VALUE { IF(V(d3)>{THRESH}, 8, 0) } E_ABM4 5 0 VALUE { IF(V(d4)>{THRESH}, 16, 0) } E_ABM5 6 0 VALUE { IF(V(d5)>{THRESH}, 32, 0) } E_ABM6 7 0 VALUE { IF(V(d6)>{THRESH}, 64, 0) } E_ABM7 8 0 VALUE { IF(V(d7)>{THRESH}, 128, 0) } E_ABM8 9 0 VALUE { IF(V(d8)>{THRESH}, 256, 0) } E_ABMOUT out ref VALUE { (V(1)+V(2)+V(3)+V(4)+V(5)+V(6)+V(7)+V(8)+V(9)) }	• Available cases: 9bit input • Use function find_H_L_THRESH() to pull the threshold voltage from another SIMPLIS primitive for use in this translation • A reference node can be supplied or not supplied, the logic can handle it • NUM_BITS: Number of bits is calculated depending on whether a ref node was given or not.	

			.ends D2A_CONVERTER_9B		

OPAMP	"PARAM_OPAMP"	X3 vin+ vin- VDD VSS out PARAM_OPAMP vars: LEVEL=2 VOS=0 IB=100n IBOS=1n + A0=100k GBW=500k SR_POS=10Meg SR_NEG=10Meg CMRR=100k PSRR=100k + RIN=10G IO_SRC_MAX=250u IO_SNK_MAX=250u ROUT=4k RO_AC=390 IQ=1m + VDIFF_POS=100m VDIFF_NEG=100m USE_BACK_ANNO=0 DEF_IC_R_FF2=1 + DEF_IC_R_FF3=1 BA_IC_R_FF2=2 BA_IC_R_FF3=2	X_3 vin+ vin- out VDD VSS Error_Amplifier .SUBCKT Error_Amplifier INM INP OUT VDD VSS PARAMS: isource=5m headroom=0 + bw=1meg isink=5m + co=1n rout=100m vslewn=1e6 gain=100000 vslewp=1e6 IC=0 .PARAM ro={1/(2*3.1428*pole*co)} gm={gain/ro} pole={bw/gain} islewn={- + vslewn*co} islewp={vslewp*co} G_ABM2I1 VDD N16716809 VALUE { LIMIT(((V(INP)-V(INM))*{gm}), {Islewn}, {Islewp}) } D_D5 N16742688 N16742726 D_EA R_R1 VSS N16716809 {Ro} C_C1 N16716809 VSS {Co} IC={IC} D_D3 N16716809 N167167730 D_EA R_R4 N16742680 OUT {Rout} D_D4 N167167951 N16716809 D_EA	• no parameters are currently in use. It only implements this default error_amplifier	

			V_V1 N167167951 VSS {{Headroom}-90m} V_V2 N167167730 VDD {{Headroom}-70m} I_I2 N16742680 N16742688 DC {Isink} D_D6 N16742688 N16742680 D_EA E_E1 N16742726 VSS N16716809 VSS 1 I_I1 N16742726 N16742688 DC {Isource} .model D_EA d is=1e-015 tt=1e-011 n=0.1 .ENDS Error_Amplifier		

4.3 Primitive translations that start with "!"

VPWL Resistor	"PWLR"	!R$R22 node+ node- R22$TP_SSPWLR IC=1 .MODEL R22$TP_SSPWLR VPWLR NSEG=6 X0=0 Y0=0.1 X1=1 Y1=0.162 X2=2 Y2=0.237 + X3=3 Y3=0.332 X4=4 Y4=0.442 X5=5 Y5=0.619 X6=6 Y6=0.825 … XN=10 YN=1 *Piece wise data points can extend as far as the user designs it	G_VPWL_R22 node+ node- TABLE {V(node+, node-)} ((X0,Y0) (X1,Y1) + (X2,Y2) (X3,Y3) ... (XN-1,YN-1) (XN,YN) (100k,Ycalculated)) R_VPWL_R22 node+ node- 1T	• X and Y values: The piecewise data points will be iterated over in full and copied over to the PSPICE netlist. The final 2 data points are used to calculate the slope and then that slope is used to calculate a distant data point at X=100k. • Ycalculated = m*(100k-XN) +YN	

INV model	"TP_GATE" Then search the .MODEL line to determine if: GATE == "INV"	!D$U15 Y 0 A U15$TP_GATE IC=0 .MODEL U15$TP_GATE INV RIN=10meg ROUT=41.67 TH=2.15 HYSTWD=0.1 VOL=0 + VOH=5 DELAY=1e-010	X_U15 A Y INV_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .SUBCKT INV_BASIC_GEN A Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A)>{VTHRESH} , {VSS}, {VDD})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS INV_BASIC_GEN	• VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage	
AND model	"TP_GATE" Then search the .MODEL line to determine if: GATE == "AND2"	!D$U12 Y 0 A B U12$TP_GATE IC=0 .MODEL U12$TP_GATE AND2 RIN=10meg ROUT=41.67 TH=2.15 HYSTWD=0.1 VOL=0 + VOH=5 DELAY=2e-010 LOGIC=POS	X_U12 A B Y AND2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .SUBCKT AND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS AND2_BASIC_GEN	• Available # of inputs: 2, 3, 4 • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage	

NAND model	"TP_GATE" Then search the .MODEL line to determine if: GATE == "NAND2"	!D$U12 Y 0 A B U12$TP_GATE IC=0 .MODEL U12$TP_GATE NAND2 RIN=10meg ROUT=41.67 TH=2.15 HYSTWD=0.1 VOL=0 + VOH=5 DELAY=2e-010 LOGIC=POS	X_U12 A B Y NAND2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .SUBCKT NAND2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} & V(B) > {VTHRESH}, {VSS}, {VDD})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS NAND2_BASIC_GEN	• Available # of inputs: 2 • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage	
OR model	"TP_GATE" Then search the .MODEL line to determine if: GATE == "OR2"	!D$U12 Y 0 A B U12$TP_GATE IC=0 .MODEL U12$TP_GATE OR2 RIN=10meg ROUT=41.67 TH=2.15 HYSTWD=0.1 VOL=0 + VOH=5 DELAY=2e-010 LOGIC=POS	X_U12 A B Y OR2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .SUBCKT OR2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} | V(B) > {VTHRESH}, {VDD}, {VSS})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS OR2_BASIC_GEN	• Available # of inputs: 2, 3, 4 • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage	

☑ NOR model	"TP_GATE" Then search the .MODEL line to determine if: GATE == "NOR2"	!D$U12 Y 0 A B U12$TP_GATE IC=0 .MODEL U12$TP_GATE NOR2 RIN=10meg ROUT=41.67 TH=2.15 HYSTWD=0.1 VOL=0 + VOH=5 DELAY=2e-010 LOGIC=POS	X_U12 A B Y NOR2_BASIC_GEN PARAMS: VDD=VOH VSS=VOL VTHRESH=TH .SUBCKT NOR2_BASIC_GEN A B Y PARAMS: VDD=1 VSS=0 VTHRESH=0.5 ROUT=1 DELAY=1n E_ABMGATE YINT 0 VALUE { {IF(V(A) > {VTHRESH} | V(B) > {VTHRESH}, {VSS}, {VDD})} } RINT YINT Y {ROUT} CINT Y 0 {DELAY*1.3} .ENDS NOR2_BASIC_GEN	• Available # of inputs: 2 • VOH: Logic high voltage • VOL: Logic low voltage • TH: Threshold voltage	

5 Script Support Contact

Developed by: Britton Jones (britton@ti.com), BSR-MV Applications Engineer

6 Additional Files

6.1 Step-by-Step Guide

SIMPLIS_to_PSPICE_Step_by_Step_Guide_rF.docx[6]

6.2 Intern Presentation - Translator Verification

Intern_Presentation_John_Donaghey_8_6_21_r4.ppt[7]

6 https://confluence.itg.ti.com/download/attachments/300391857/SIMPLIS_to_PSPICE_Step_by_Step_Guide_rF.docx?api=v2&modificationDate=1646288586000&version=1

7 https://confluence.itg.ti.com/download/attachments/300391857/Intern_Presentation_John_Donaghey_8_6_21_r4.ppt?api=v2&modificationDate=1646323893000&version=1