Computer-Aided VLSI System Design

Verilog Lab 2: (1) DesignWare, SDF Annotation, and Verdi (2) Synchronous RAM Control

Objectives:

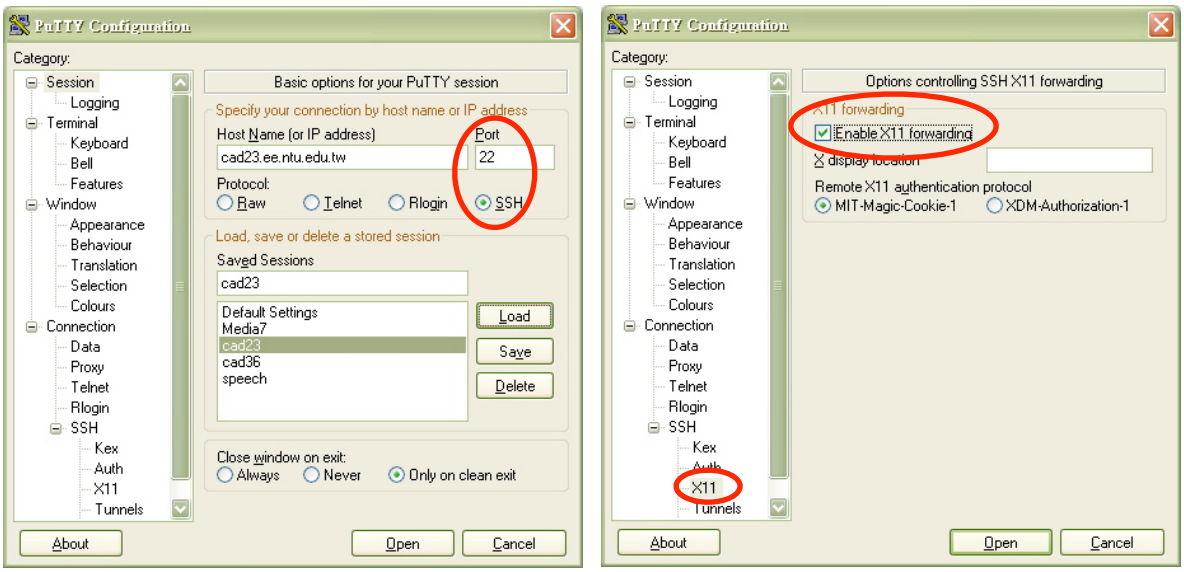
In this lab, you will learn:

1. How to use DesignWare
2. How to perform gate-level simulation with SDF annotation
3. How to generate fsdb files and use Verdi waveform debugger

Environment Setup:

1. start cygwin or X-win32
2. setup X-windows display environment variable by:

   `X11 Forwarding and SSH v2 from PuTTY`



3. source the default *.cshrc* file:

   source /home/raid1_1/.cshrc

4. add command alias for gate-level simulation: (one command line)

   alias verd ncverilog +access+r +lic_ncv -v /home/raid1_1/cic/CBDK018_UMC_Artisan/CIC/Verilog/umc18.v

*If you did not use alias command, you could copy umc18.v to your directory. After that, you have to add umc18.v in gate level simulation. For example:

verd –f Lab2_alu_run_s.f

ncverilog +access+r –f Lab2_alu_run_s.f umc18.v

Copy Files from cvsd Directory

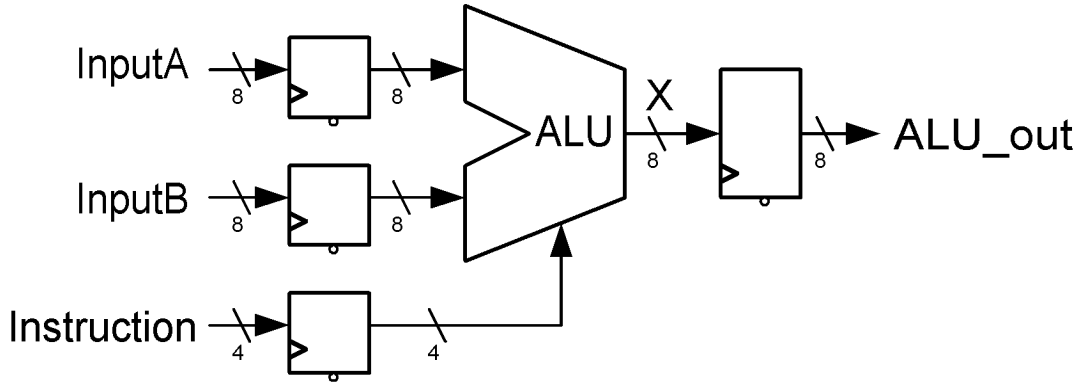
1. copy all the files into your work directory,

```
cp -R ~cvsd/CUR/Verilog/Lab2 .
```

2. check if you have these files

Filename	Description
DW01_addsub.v	DesignWare: adder and subtractor
Lab2_alu.v	RTL code of 8-bit ALU
Lab2_test_alu.v	test bench for 8-bit ALU
Lab2_alu_run.f	Command-line file for RTL code
Lab2_alu_s.v	Gate-level code of 8-bit ALU
Lab2_alu_run_s.f	Command-line file for Gate-level
Lab2_alu_s.sdf	sdf file for gate-level code
Lab2_test_ram.v	test bench for RAM control

Lab2-1 DesignWare, SDF Annotation, and Verdi



DesignWare

1. The ALU function of *Lab2_alu.v* is exactly the same as that of HW1, except the additional registers for all input signals. Thus, the test bench *Lab2_test_alu.v* should be modified. Find out how it is modified. Moreover, *Lab2_alu.v* replaces "+" by DesignWare module "*DW01_addsub*". Run the simulation by:

   ncverilog +access+r –f Lab2_alu_run.f
2. There are many errors about unmatched port size because the default port size of *DW01_addsub* is 4, but that of our ALU module should be 8. Please modify line 13 of *Lab2_alu.v* like

   DW01_addsub **#(8)** addsub(…);
3. Run simulation again. The RTL code is function correct now.

Gate-level Simulation with SDF Annotation

1. After synthesizing *Lab2_alu.v*, we can derive the gate-level netlist *Lab2_alu_s.v* . For gate-level simulation, the corresponding SDF file *Lab2_alu_s.sdf* should be annotated. Please unmark line 16 of *Lab2_test_alu.v*. Then gate-level simulation can be performed by:

   verd –f Lab2_alu_run_s.f
2. There might be some "Negative timing check limit" SDFA warning while the gate-level simulation was running, and it was okay. Gate-level simulation would take a longer time than RTL simulation. Finally, the simulation is successful. It means this gate-level netlist works well with clock cycle time 10ns. What will happen if you run the simulation by

   ncverilog +access+r –f Lab2_alu_run_s.f

Create fsdb Waveform Files

1. Please unmark line 17 and 18 of *Lab2_test_alu.v*. Then run simulation to generate the waveform:

   verd –f Lab2_alu_run_s.f

2. Check if you derive the waveform file *Lab2_alu_s.fsdb* . This waveform will record the signal transition of all internal nodes inside the gate-level netlist during the simulation period.
3. Please create the waveform file *Lab2_alu.fsdb* for the RTL netlist. You have to modify the testbench(Lab2_test_alu.v) to generate Lab2_alu.fsdb by rewriting the fsdb filename in line 17 as **Lab2_alu.fsdb.** After that, type the command:

   ncverilog +access+r -f Lab2_alu_run.f

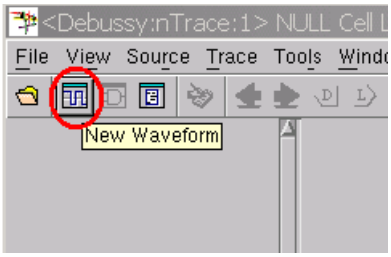
   You can check that Lab2_alu.fsdb is generated or not.

Verdi Waveform Debugger

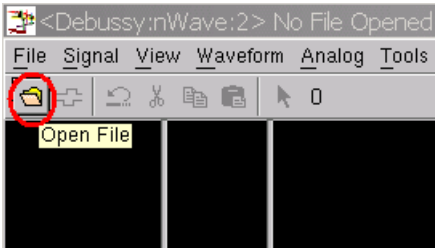
1. Open the GUI of Debussy: (Be sure X-windows is ready)

   `verdi&`

2. Create a waveform window by clicking:

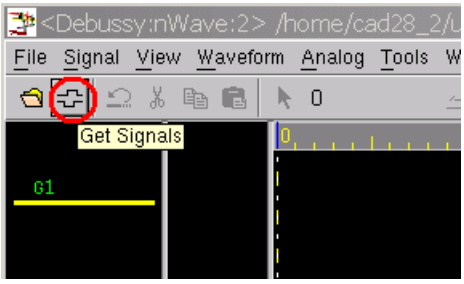


3. Open fsdb waveform file from the waveform window by clicking



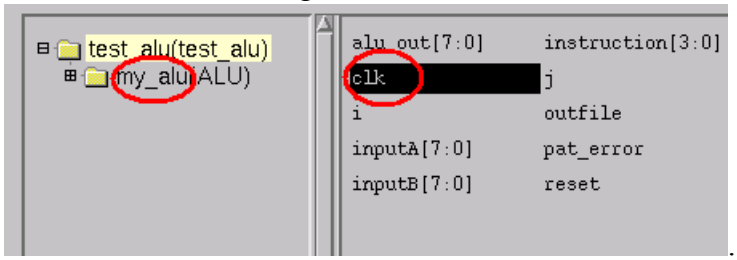
   and select *Lab2_alu_s.fsdb*.

4. Now you can select waveform of internal signals to debug. Click

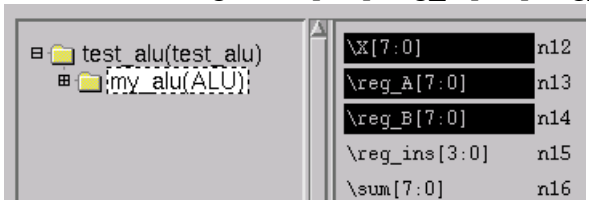


   There is **test_alu(test_alu)** in the signal list, select signal **clk**, and click **Apply**. Now you can see the waveform of **clk** from the waveform window.

5. You can further select signals inside ALU module. Select the **my_alu(ALU)**

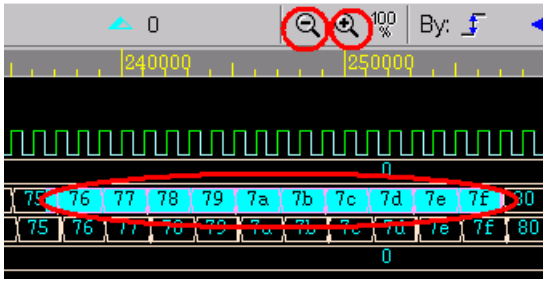


Choose these three signals X[7:0], reg_A[7:0], reg_B[7:0]

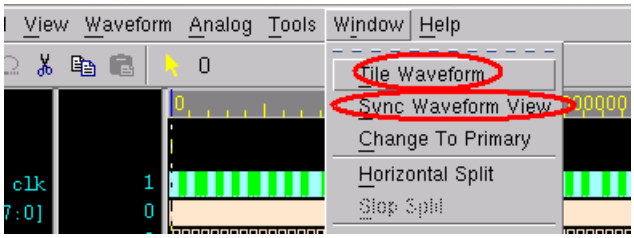


and then click **Apply**. Now you can examine if these signals work as the instruction set define. (Include signal **reg_ins** by yourself.)

6. Try to zoom in and zoom out the waveform by clicking icons or selecting directly:



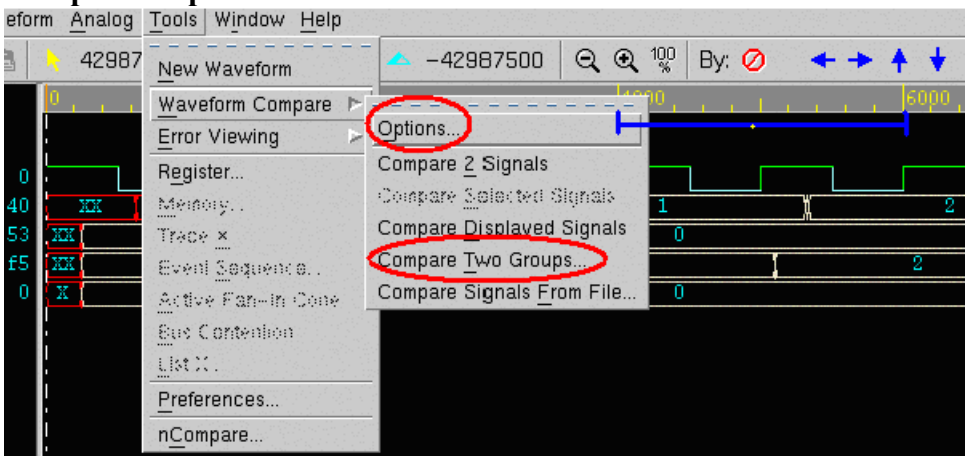
7. Open another waveform for *Lab2_alu.fsdb* and select the same signals as the above steps.
8. We can examine these two waveforms at the same time since the input signals and cycle time are all the same. Synchronize the waveform view by clicking **Sync. Waveform View** for both waveform windows.



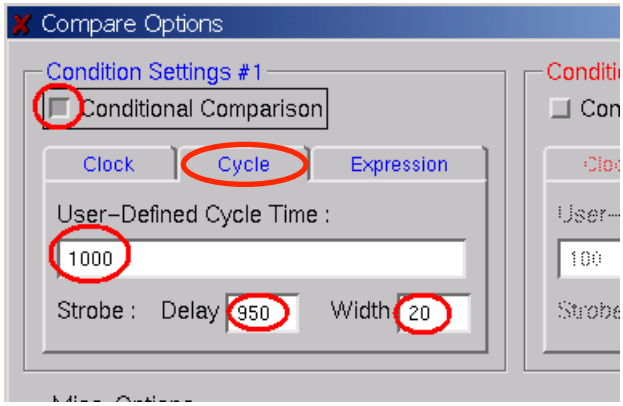
Then click **Tile Waveform** on one of the two waveform windows.

9. Please examine these two waveforms carefully. You will find the gate-level simulation requires more delay for signal transition than RTL simulation. This is because RTL simulation only checks logical functions while gate-level simulation contains timing information.

10. Sometimes we need to compare two waveforms. Click **Tools→Waveform Compare→Options..**

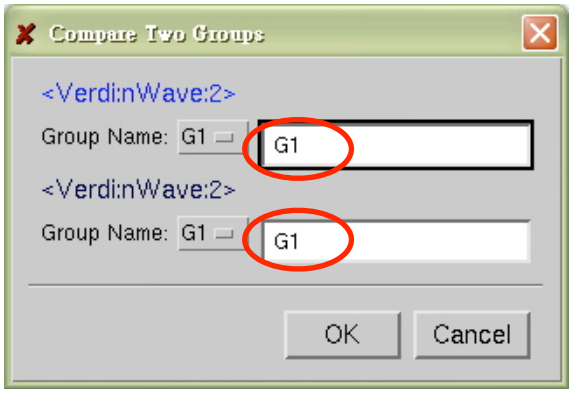


Then set the parameters as

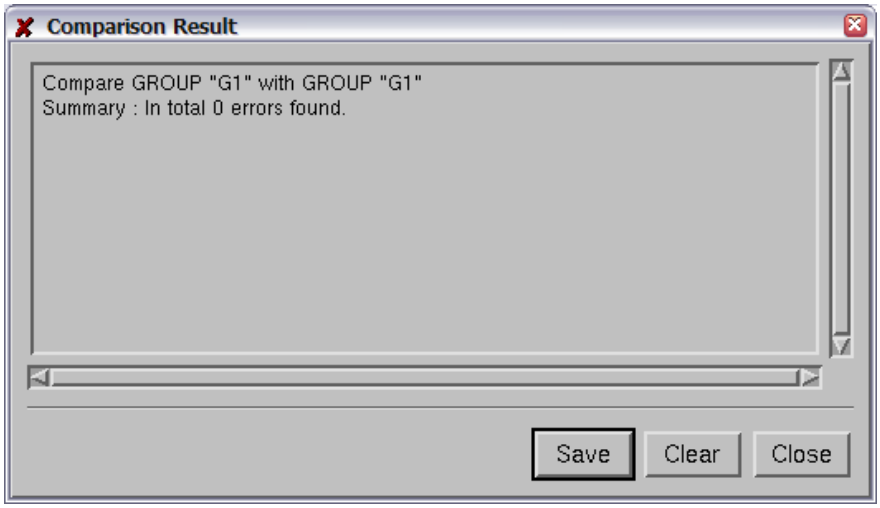


The cycle time is set as 1000 because the clock cycle time of this waveform window is 1000 according to the blue line. (The timing unit is the resolution set by `timescale.) The strobe setting represents the comparison is only performed at time 950~970 for every cycle. Click **OK** to close the **Compare Options** window.

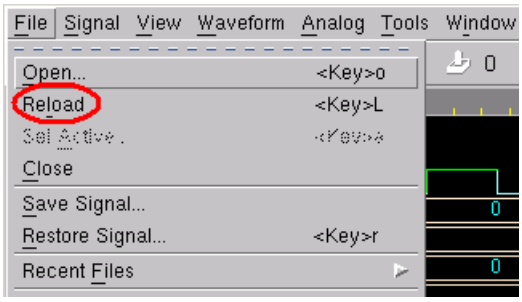
11. Now we can compare a group of signals between two waveforms. Be sure these two waveforms are synchronized and the appearing signals are correspondent. Click **Tools→Waveform Compare→Compare Two Groups** and set as



12. Finally, the comparison result appears.



13. Please find out how to compare two signals.
14. If the simulation is rerun and the fsdb waveform file is rebuilt, you can reload the waveform directly, instead of opening the waveform again.



Lab2-2 Synchronous RAM Control

Objectives:

In this lab, you will learn:

1. How to use Memory Compiler
2. How to control the Synchornous RAM

Using Memory Compiler

1. Design your memory name and specification

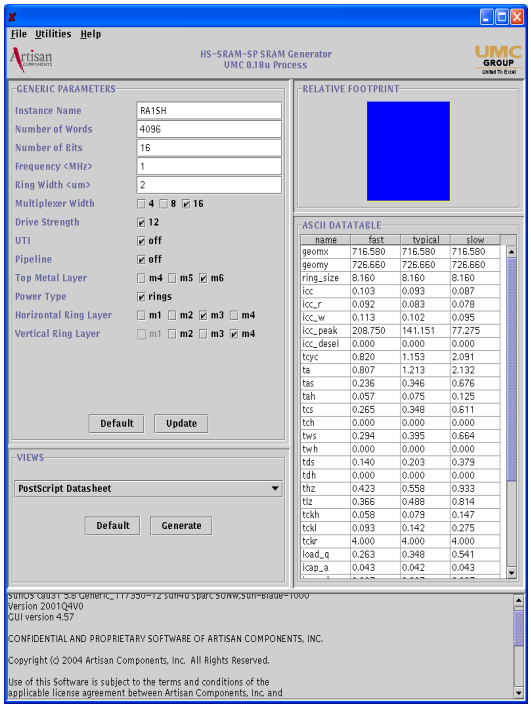
   The name of your memory should be conformed to some kind of rule if you want to maintain them easily in the future. Here we need a high speed, single port memory with 64 words, each word is 8bit. Therefore we name our memory as HSs18n_64x8, which means High Speed single port 0.18μm normal RAM with 64words/8bit. Remember the name "HSs18n_64x8", it's the name of your memory.

2. Run the memory compiler

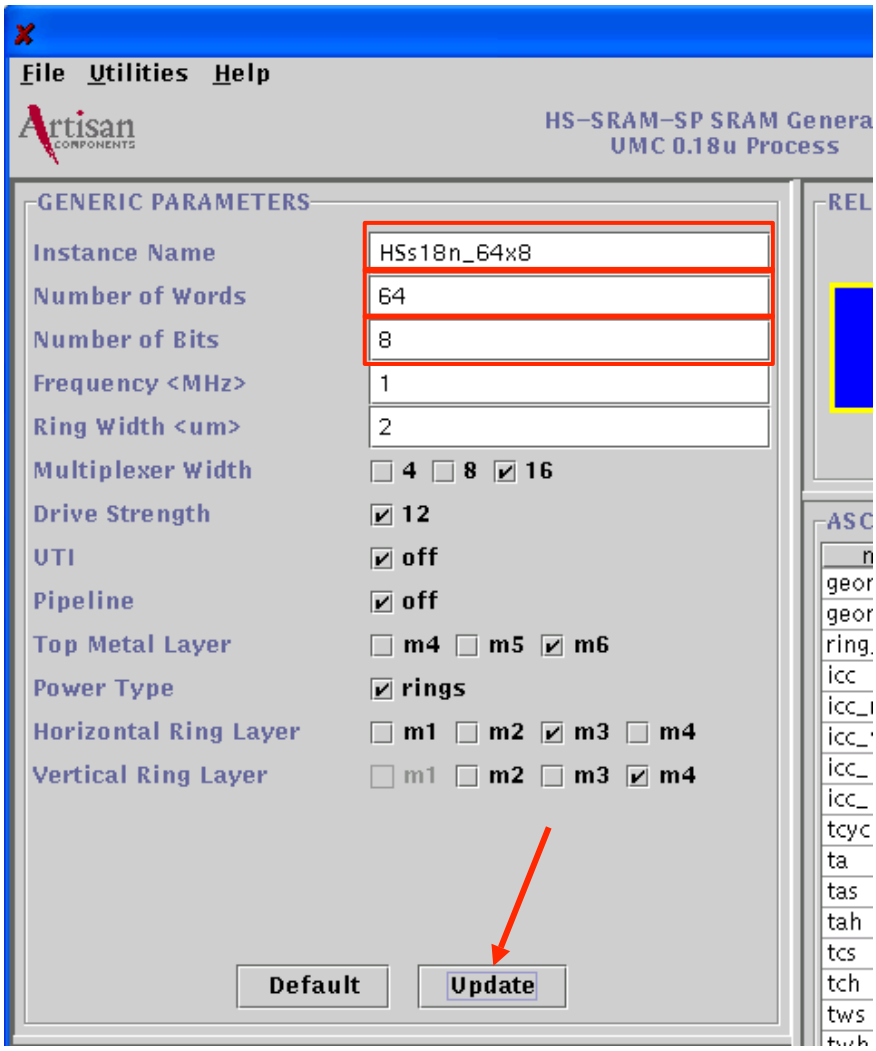
   There are three kinds of memory compiler in UMC 0.18μm cell library: ra1sh_1, ra1sh_2, and ra2sh. "ra1sh_1" is used for high-speed single-port SRAM, "ra1sh_2" is used for larger size SRAM, and "ra2sh" is for high-speed dual port SRAM designs. "ra1sh_1" is used in this Lab, you can type following command to start:

Generator	UMC	TSMC
High-Speed Single-Port SRAM	ra1sh_1	ra1sh
High-Speed Dual-Port SRAM	ra2sh	ra2sh
Large-Size Single-Port SRAM	ra1sh_2	
High-Density Single-Port SRAM		ra1shd

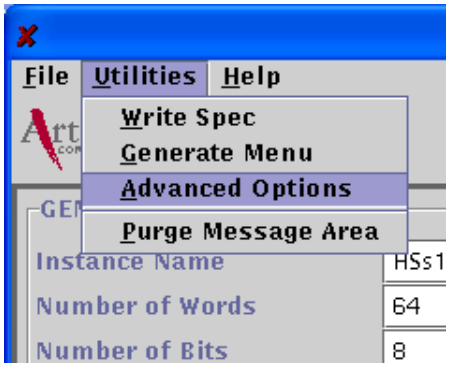
```
/home/raid1_1/cic/CBDK018_UMC_Artisan/CIC/Memory/
ra1sh_1/bin/ra1sh &
```

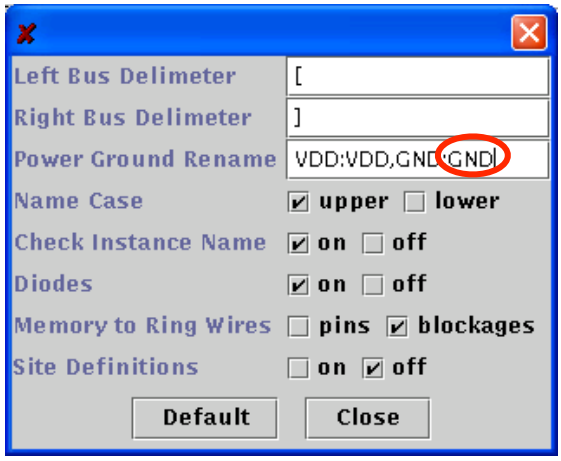


3. Click in the specification

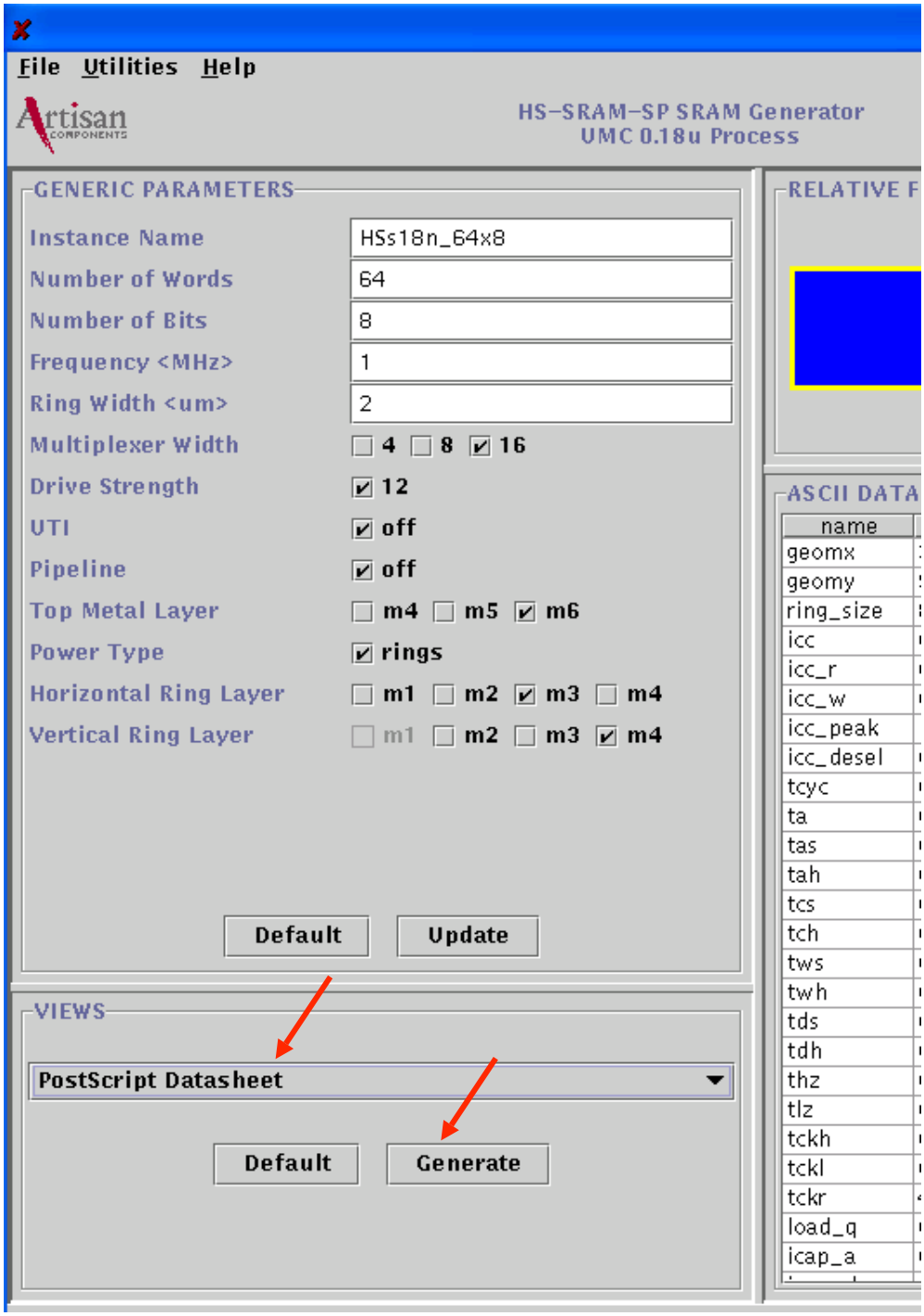


4. Change the “Ground” name, click on the “Utilities” then “Advanced Options” Then change the GND:VSS in the “Power Ground Rename” to GND:GND

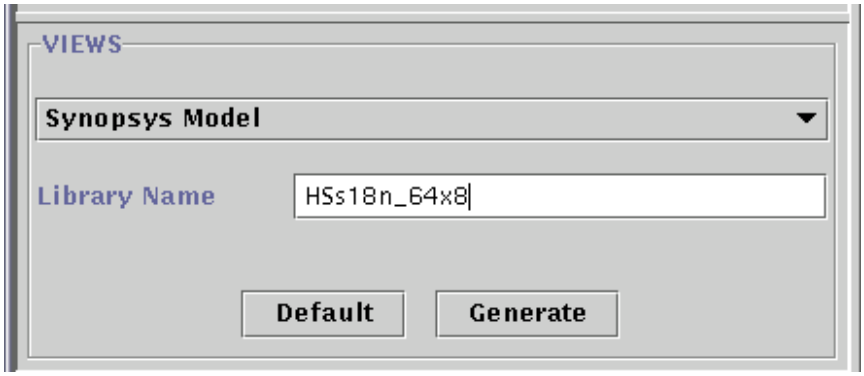




5. Generate the Data Sheet and Verilog Behavior Model.
   Click on the List box and choose "**`PostScript Datasheet`**", then click the "**`Generate`**" button. You'll get a HSs18n_64x8.ps which is the data sheet of the SRAM. And then click on the list box and choose "**`Verilog Model`**", and click the "**`Generate`**" button. You'll get a file named HSs18n_64x8.v.



6. Generate the library for using in Synthesis
   Choose "**`Synopsys Model`**" in the list box and type in your library name. Here we type "**`HSs18n_64x8`**" into it.



   Click on "Generate" then you can generate three ".lib" for use in Synopsys Synthesis tool. The three ".lib" files are HSs18n_64x8_fast_syn.lib, HSs18n_64x8_typical_syn.lib, HSs18n_64x8_slow_syn.lib.
7. After all, Click on the "Utilities -> Write Spec" to write down the spec of your SRAM. CIC will model your RAM according to the specification file you wrote.

Read Timing Diagram of Synchronous RAM

1. Derive the data sheet of synchronous RAM:
   Download the HSs18n_64x8.ps and use GhostView or Acrobat to read it.
2. Find out the function and timing diagram for every port of *HSs18n_64x8* module in *HSs18n_64x8.v* according to the data sheet. How many words can be stored? How many bits for a word?

Run Simulation of RAM Control

1. Run Verilog-XL simulator:
   ncverilog +access+r Lab2_test_ram.v HSs18n_64x8.v
2. Open the waveform *Lab2_ram.fsdb* via Debussy. Compare the waveform for every port of *HSs18n64x8* module with the timing diagram of data sheet. Furthermore, you can find that the output port *O[7:0]* has some timing delay because the timing information of *HSs18n_64x8s* module is described in *HSs18n_64x8.v* .
3. Please examine how *Lab2_test_ram.v* to control this synchronous RAM.

END of LAB

Creator:
1st Edition: Chao-Tsung Huang, 2002
2nd Edition: Yu-Lin Chang, 2004
3rd Edition: Yu-Lin Chang, 2006
4th Edition: Jui-Hsin Lai (Larry), 2008