RTL Logic Synthesis Tutorial

This tutorial is adapted from the tutorial created by *Mircea R. Stan* from the University of Virginia and has been modified for the ECE484 class in fall 2010.

The following Cadence CAD tools will be used in this tutorial: **RTL Compiler Ultra** for logic synthesis

Please revisit the Simulation Tutorial before doing this new tutorial.

Running the Cadence logic synthesis tools

First you need to log on to arda and set your project to ece484. Issue the following commands

- `fix` (and then enter your password)
- `cds`
- `p` (to check that the project ece484 is set ... if not use the "sp ece484" command)

Before lunching the RTL Compiler you need to write a tcl script, so type

```
gedit
```

Use gedit to create the following file:

```
## Define the search path
set_attribute lib_search_path $env(STDCELL_LIBS)
## This defines the library to use
set_attribute library {osu025_stdcells.lib}
## Read in verilog code for 8-bit accumulator
read_hdl -v2001 ../verilog.src/accu.v
## This creates a technology-independent schematic
elaborate
## Put in timing constraints
## We will use the SDC constraint format
## Our SDC commands must be placed into separate file.
read_sdc ./accu.sdc
## This is where you can put in non-timing related constraints
##set_attribute avoid true FAX1
## Create a techonology-dependent schematic
synthesize -to_mapped
## Write out synthesized verilog netlist
write_hdl -mapped > ../encounter/accu_synth.v
write_hdl -mapped > ../verilog.src/accu_synth.v
##  Write out the SDC file we will take into the place n route tool
write_sdc > ../encounter/accu_synth.sdc
## Write out area and timing reports
report area > ./accu_area_report
report timing > ./accu_timing_report
```

Save as "accu.tcl" in your working directory: ~/cds/ece484/synth

Open a new file using gedit once again and type in the following code

```
#
# Set the time unit
#
set_time_unit -nanoseconds
#
# Create a clock and use it to drive the clk pin
#
create_clock -name {clk} -period 20.0 -waveform {0.0 10.0} [get_ports {clk}]
#
# Don't optimize the reset
#
set_false_path -from [get_ports {reset}]
```

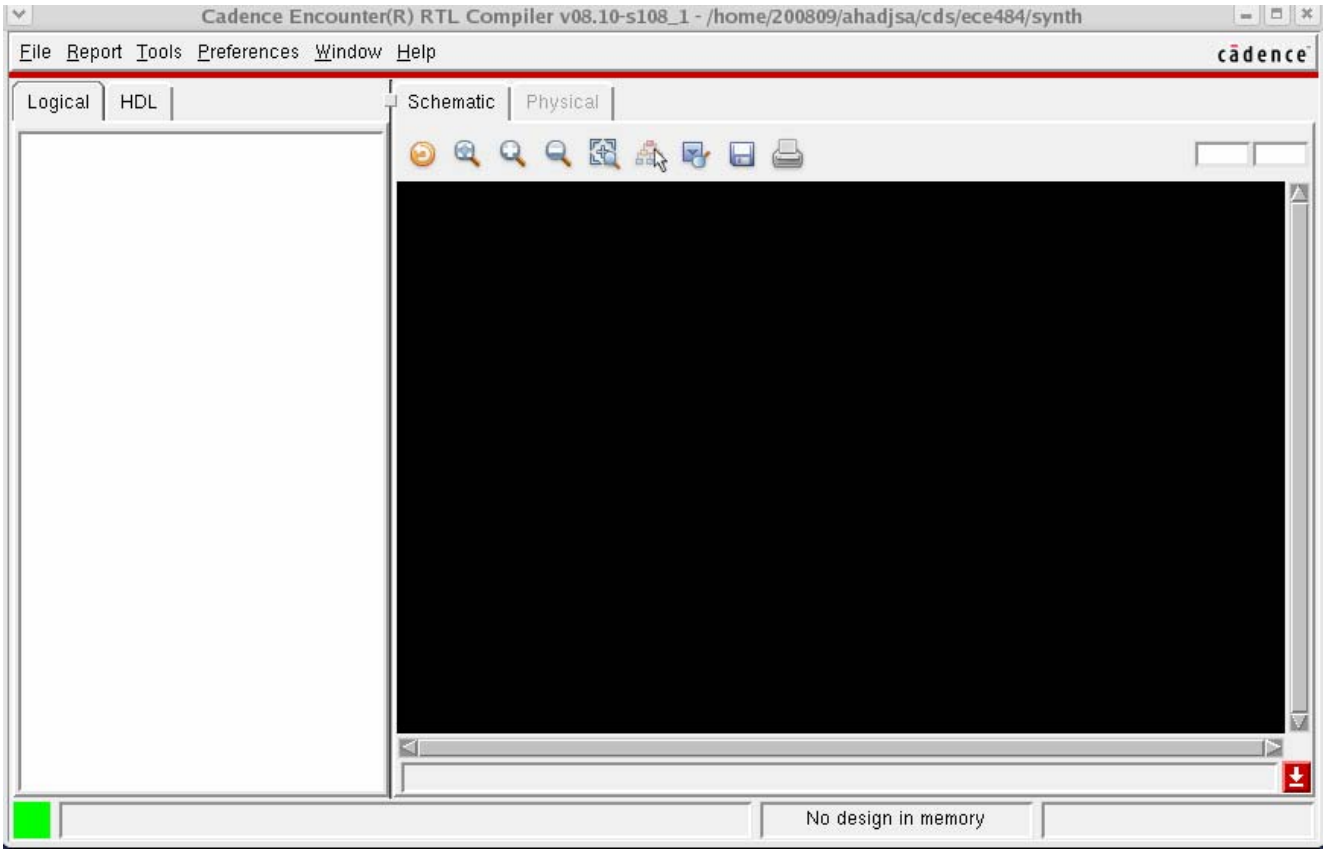
Save as “accu.sdc” in your working directory: ~/cds/ece484/synth

Close the files.

To lunch the **RTL Compiler** simulator, type the command:

`Synth`

The command `synth` starts RTL Compiler and you should get the rc startup window:



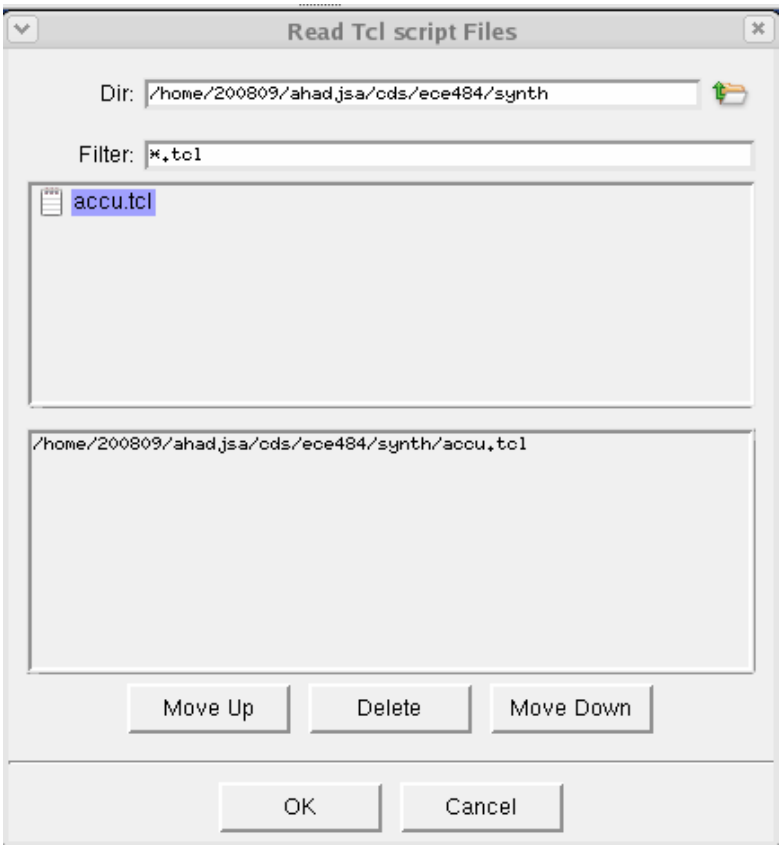
The window has three areas:

Menu Bar (top)
File Browser (left)
Schematic Browser (right)

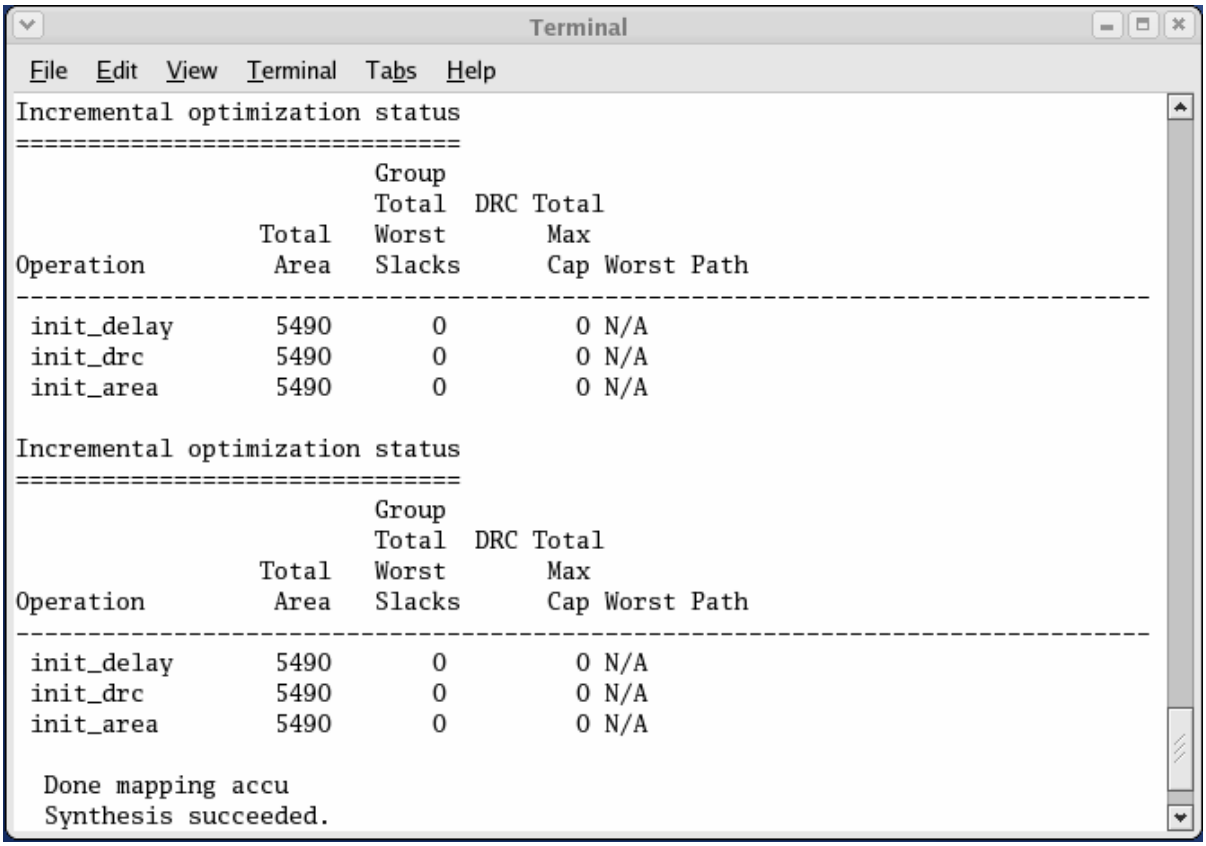
Unlike other GUI interfaces, the console this time is the initial window from which you launched rc (that's why it had to be launched in the foreground). Please try to familiarize yourself with the main window, click on the menus, etc. For more information on the various Cadence tools I encourage you to read the corresponding user manuals.

Now we need to run the script that you wrote so go to **File -> Source Script** from the File menu of the Menu Bar.

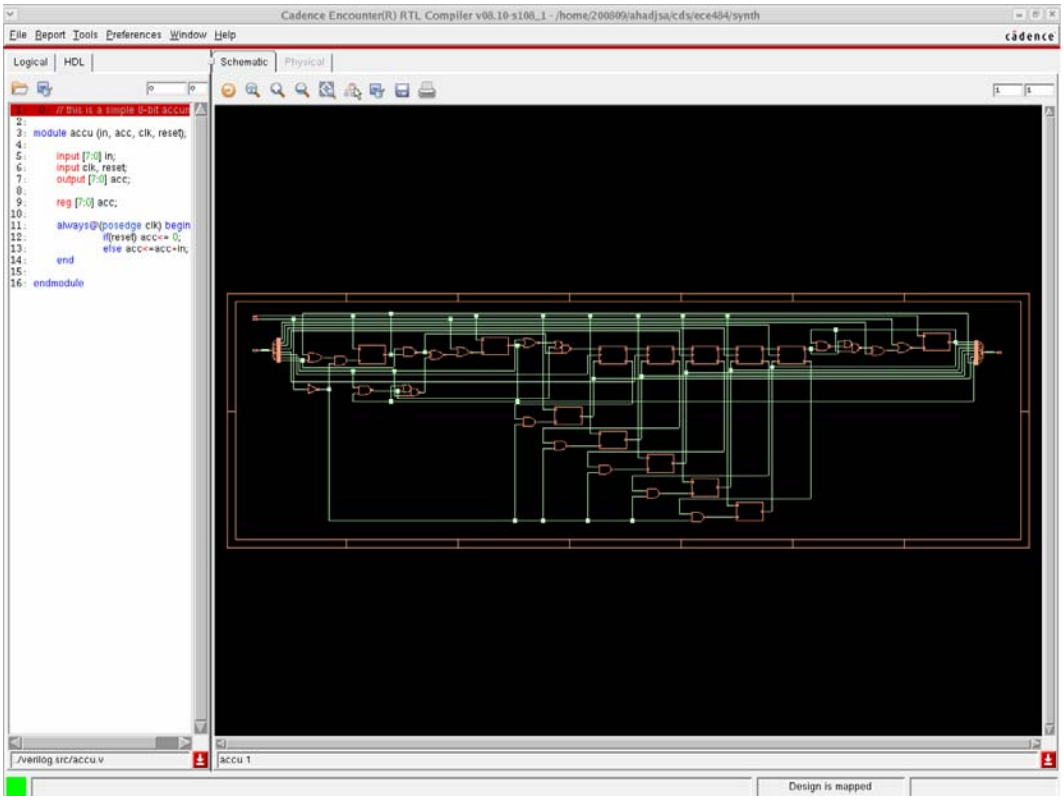
Click on accu.tcl, then OK.



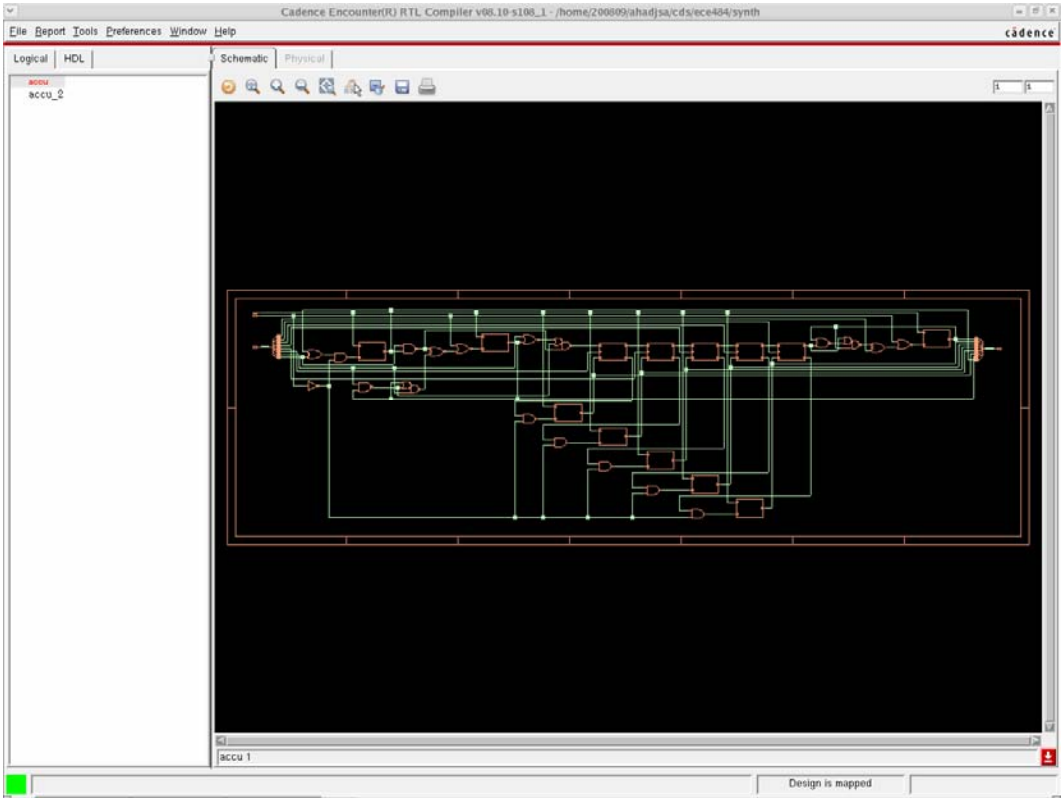
Your console should run for a brief time and you should get a success message:



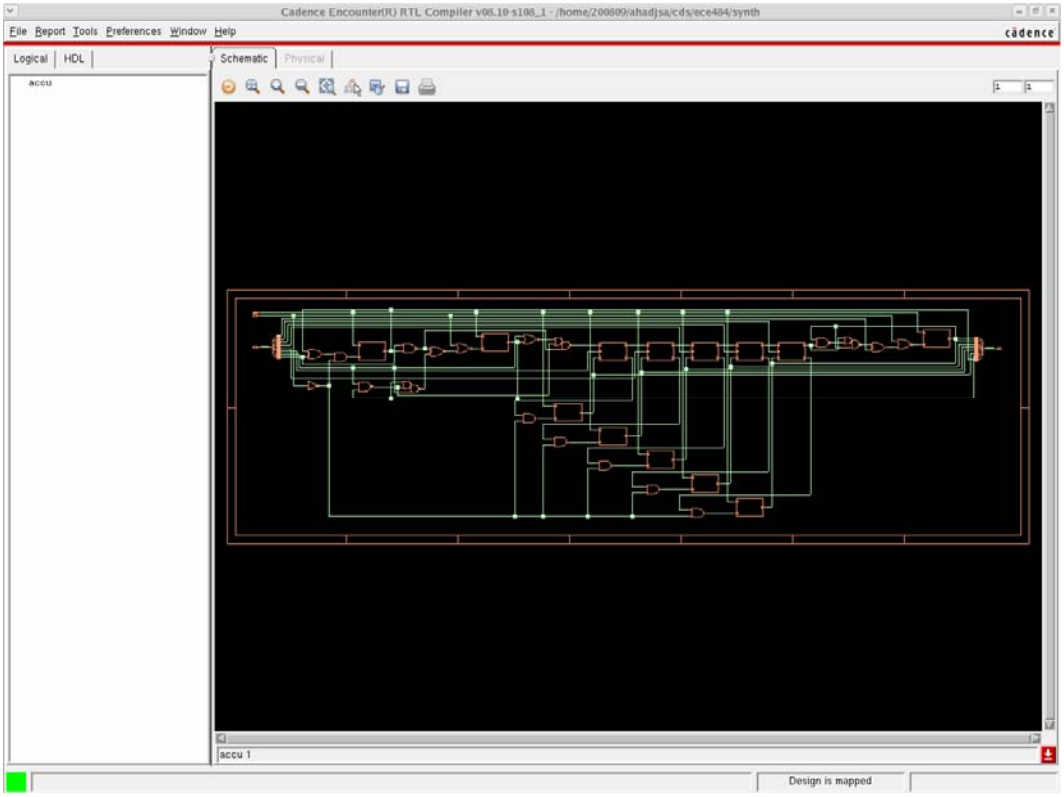
Now your GUI window should show a netlist on the right side, and if you click on HDL, the source code on the left side.



Notice that if you run the script again, another file with appear under the original file name (accu_2).



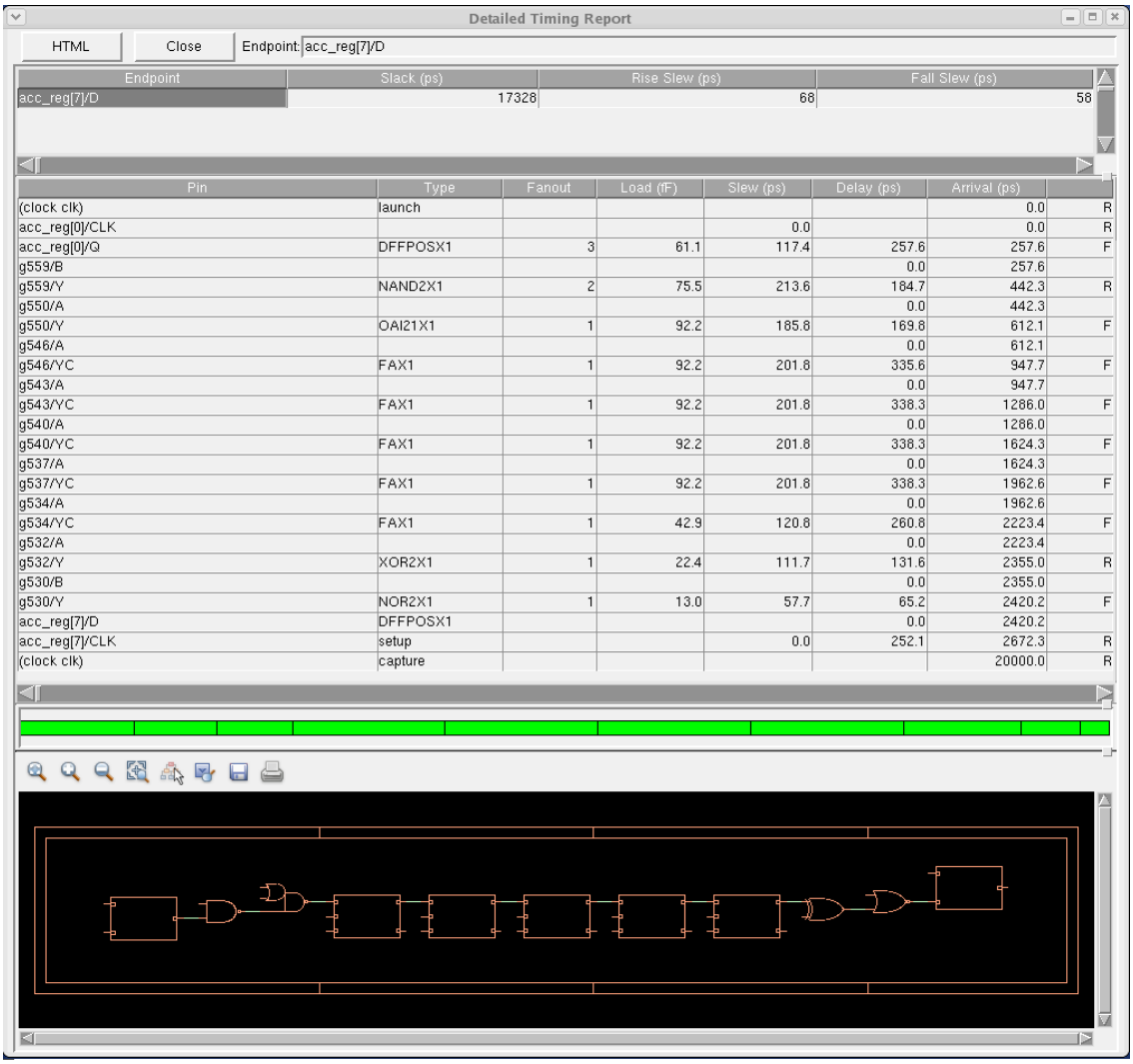
To remove that file, type “rm accu_2” in the GUI window (the terminal window) without closing the rc startup window.



Notice that you are driving the accumulator with a clock running at 50MHz (see the accu.sdc file).

“`create_clock -name {clk} -period` **`20.0`** `-waveform {0.0` **`10.0`**`} [get_ports {clk}])`

You can check the timing report by going to **Report -> Timing -> Worst Path**:

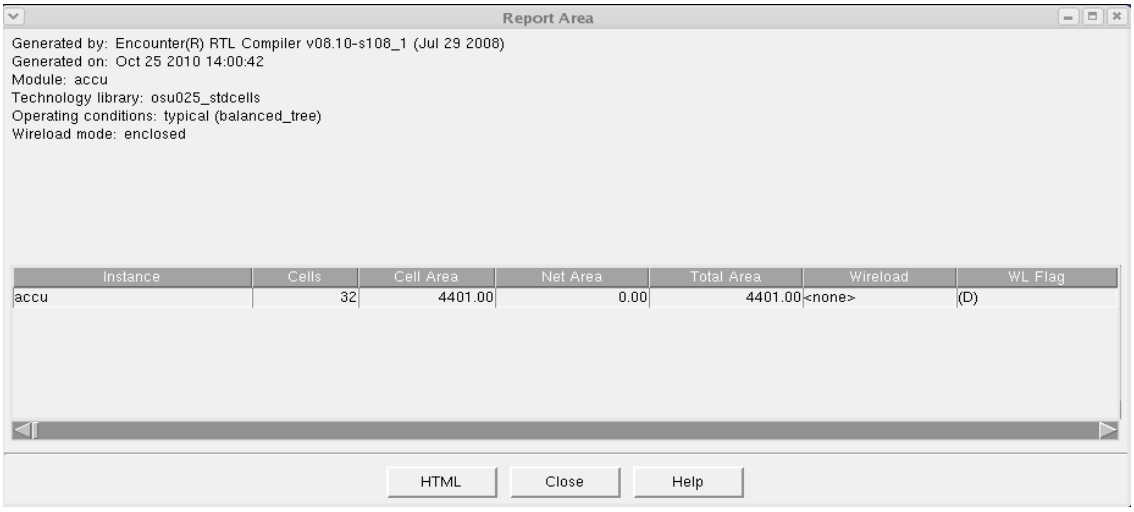


Endpoint	Slack (ps)	Rise Slew (ps)	Fall Slew (ps)
acc_reg[7]/D	17328	68	58

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
(clock clk)	launch					0.0	R
acc_reg[0]/CLK				0.0		0.0	R
acc_reg[0]/Q	DFFPOSX1	3	61.1	117.4	257.6	257.6	F
g559/B					0.0	257.6	
g559/Y	NAND2X1	2	75.5	213.6	184.7	442.3	R
g550/A					0.0	442.3	
g550/Y	OAI21X1	1	92.2	185.8	169.8	612.1	F
g546/A					0.0	612.1	
g546/YC	FAX1	1	92.2	201.8	335.6	947.7	F
g543/A					0.0	947.7	
g543/YC	FAX1	1	92.2	201.8	338.3	1286.0	F
g540/A					0.0	1286.0	
g540/YC	FAX1	1	92.2	201.8	338.3	1624.3	F
g537/A					0.0	1624.3	
g537/YC	FAX1	1	92.2	201.8	338.3	1962.6	F
g534/A					0.0	1962.6	
g534/YC	FAX1	1	42.9	120.8	260.8	2223.4	F
g532/A					0.0	2223.4	
g532/Y	XOR2X1	1	22.4	111.7	131.6	2355.0	R
g530/B					0.0	2355.0	
g530/Y	NOR2X1	1	13.0	57.7	65.2	2420.2	F
acc_reg[7]/D	DFFPOSX1				0.0	2420.2	
acc_reg[7]/CLK	setup			0.0	252.1	2672.3	R
(clock clk)	capture					20000.0	R

Notice that the “Slack time” is 17328 ps. Since it is positive, the tool is telling you that the signal arrives at the register on the right much earlier than is needed. This means that the circuit can work with a much higher clock frequency.

For the area report, go to **Report -> Netlist -> Area**:



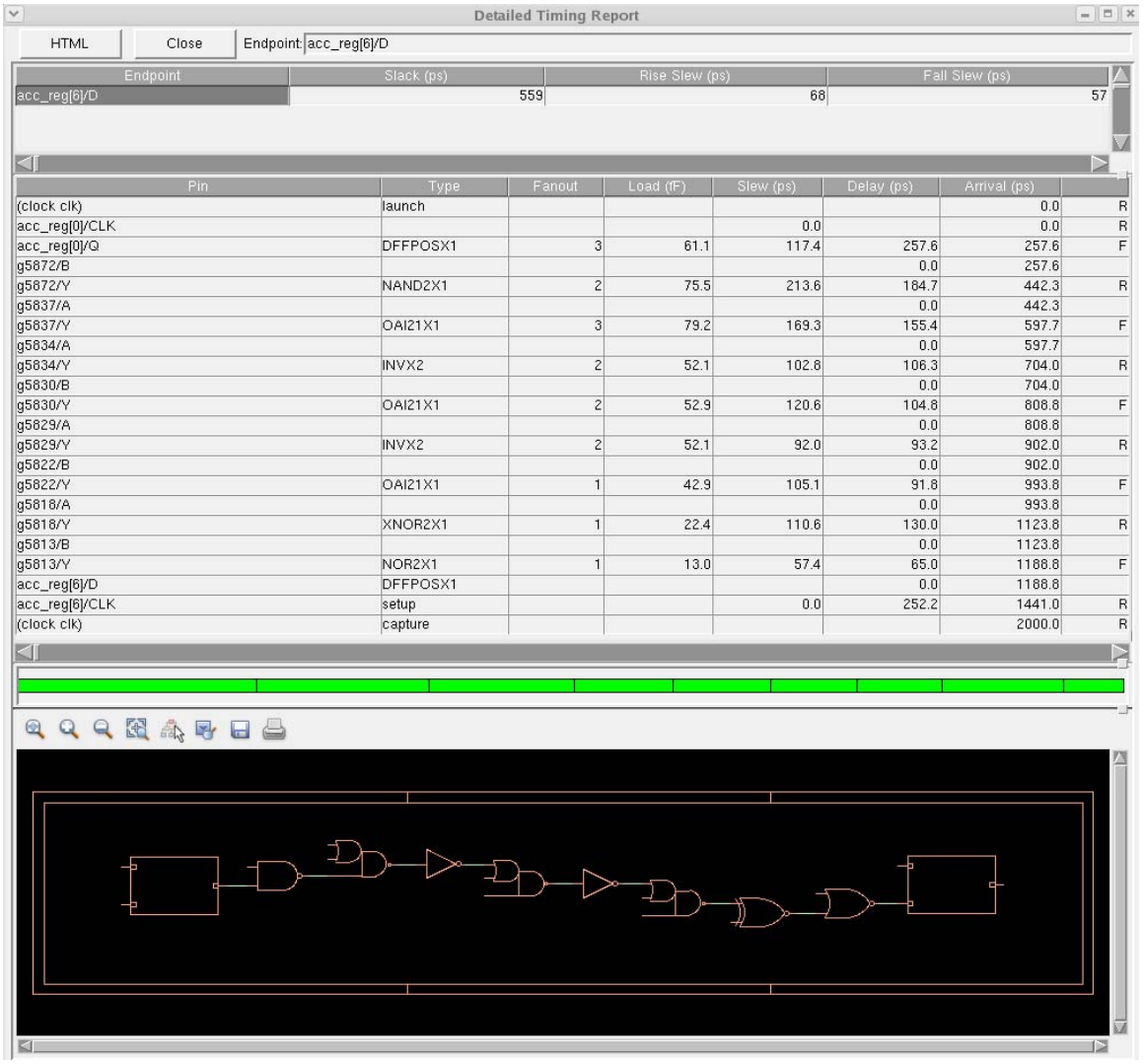
Instance	Cells	Cell Area	Net Area	Total Area	Wireload	WL Flag
accu	32	4401.00	0.00	4401.00	<none>	(D)

Check the total area. As we ask the synthesis tool to produce a faster circuit, this value is likely to increase.

Now run the accumulator with a 500MHz clock by editing the accu.sdc file:

```
create_clock -name {clk} -period 2.0 -waveform {0.0 1.0} [get_ports {clk}])
```

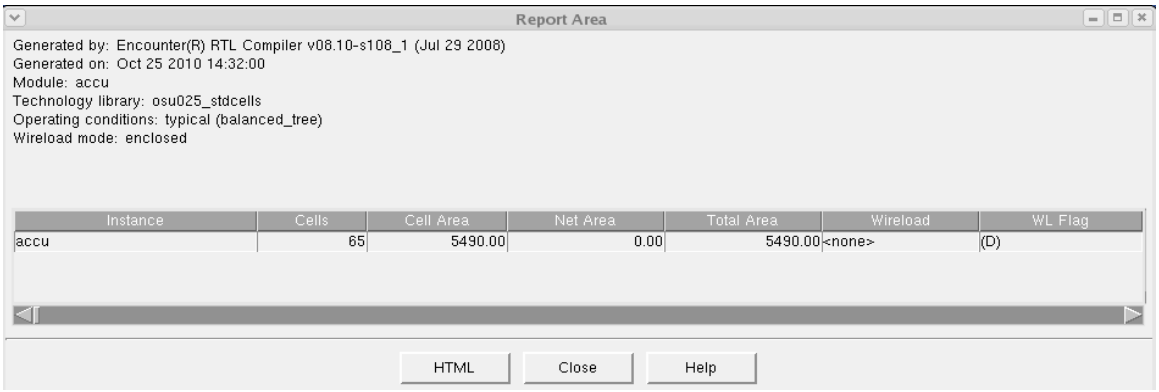
check the timing report by going to **Report -> Timing -> Worst Path**:



Endpoint	Slack (ps)	Rise Slew (ps)	Fall Slew (ps)
acc_reg[6]/D	559	68	57

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
(clock clk)	launch					0.0	R
acc_reg[0]/CLK				0.0		0.0	R
acc_reg[0]/Q	DFFPOSX1	3	61.1	117.4	257.6	257.6	F
g5872/B					0.0	257.6	
g5872/Y	NAND2X1	2	75.5	213.6	184.7	442.3	R
g5837/A					0.0	442.3	
g5837/Y	OAI21X1	3	79.2	169.3	155.4	597.7	F
g5834/A					0.0	597.7	
g5834/Y	INVX2	2	52.1	102.8	106.3	704.0	R
g5830/B					0.0	704.0	
g5830/Y	OAI21X1	2	52.9	120.6	104.8	808.8	F
g5829/A					0.0	808.8	
g5829/Y	INVX2	2	52.1	92.0	93.2	902.0	R
g5822/B					0.0	902.0	
g5822/Y	OAI21X1	1	42.9	105.1	91.8	993.8	F
g5818/A					0.0	993.8	
g5818/Y	XNOR2X1	1	22.4	110.6	130.0	1123.8	R
g5813/B					0.0	1123.8	
g5813/Y	NOR2X1	1	13.0	57.4	65.0	1188.8	F
acc_reg[6]/D	DFFPOSX1				0.0	1188.8	
acc_reg[6]/CLK	setup			0.0	252.2	1441.0	R
(clock clk)	capture					2000.0	R

and the area report by going to **Report -> Netlist -> Area**:



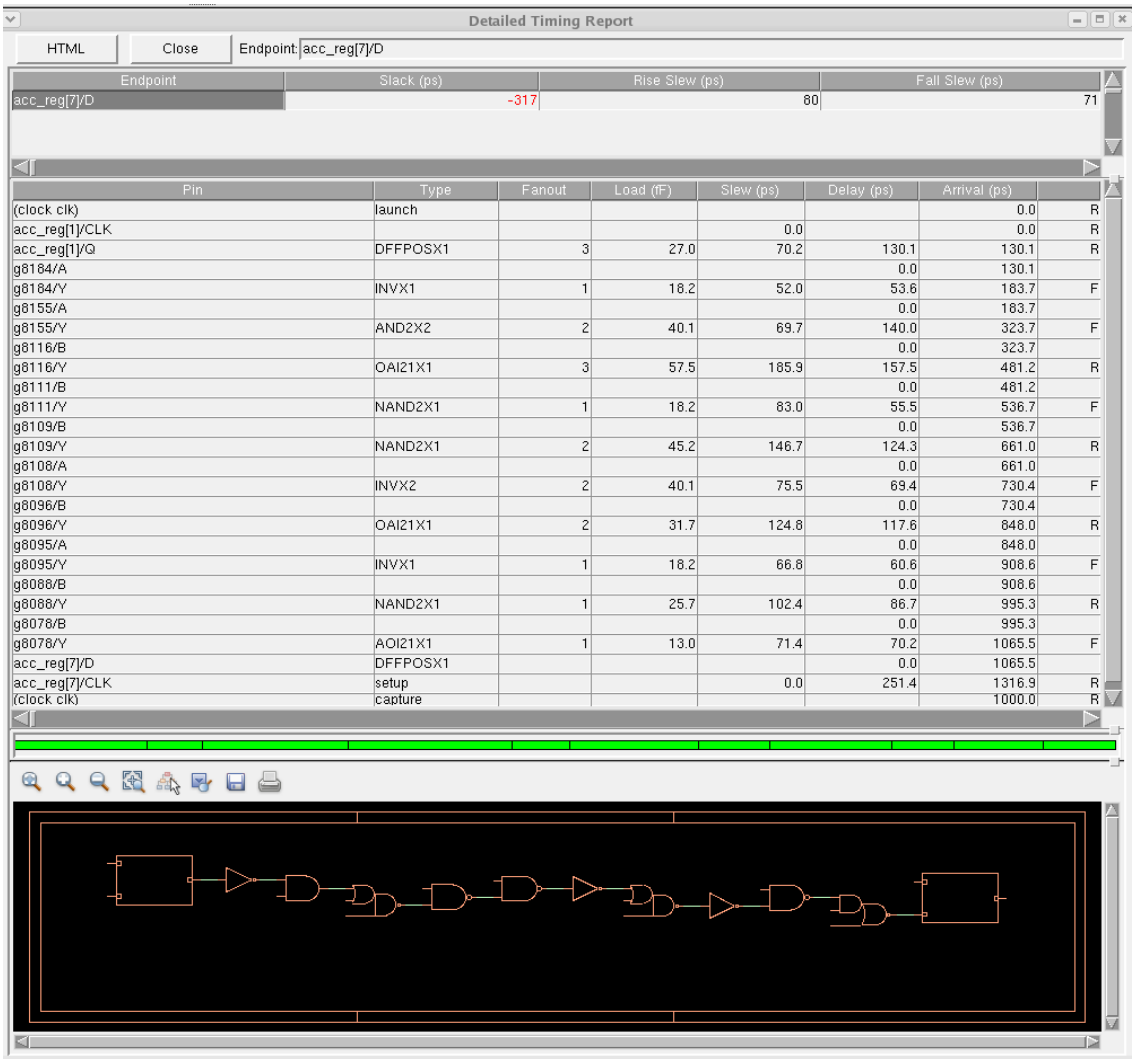
Notice that the slack time has decreased from 17328ps to 559ps but the accumulator is running 10 times faster. Also, notice that the number of cells has increased from 32 cells to 65 cells thus an increase in the area occupied (from 4401 to 5490).

Conclusion: The accumulator is running faster but occupies more area….. That's the price we pay for speed!!!

Now try to run the accumulator even faster: use a 1GHz clock. Edit the accu.sdc file:

```
create_clock -name {clk} -period 1.0 -waveform {0.0 0.5} [get_ports {clk}])
```

Check the timing report.



Endpoint	Slack (ps)	Rise Slew (ps)	Fall Slew (ps)
acc_reg[7]/D	-317	80	71

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
(clock clk)	launch					0.0	R
acc_reg[1]/CLK				0.0		0.0	R
acc_reg[1]/Q	DFFPOSX1	3	27.0	70.2	130.1	130.1	R
g8184/A					0.0	130.1	
g8184/Y	INVX1	1	18.2	52.0	53.6	183.7	F
g8155/A					0.0	183.7	
g8155/Y	AND2X2	2	40.1	69.7	140.0	323.7	F
g8116/B					0.0	323.7	
g8116/Y	OAI21X1	3	57.5	185.9	157.5	481.2	R
g8111/B					0.0	481.2	
g8111/Y	NAND2X1	1	18.2	83.0	55.5	536.7	F
g8109/B					0.0	536.7	
g8109/Y	NAND2X1	2	45.2	146.7	124.3	661.0	R
g8108/A					0.0	661.0	
g8108/Y	INVX2	2	40.1	75.5	69.4	730.4	F
g8096/B					0.0	730.4	
g8096/Y	OAI21X1	2	31.7	124.8	117.6	848.0	R
g8095/A					0.0	848.0	
g8095/Y	INVX1	1	18.2	66.8	60.6	908.6	F
g8088/B					0.0	908.6	
g8088/Y	NAND2X1	1	25.7	102.4	86.7	995.3	R
g8078/B					0.0	995.3	
g8078/Y	AOI21X1	1	13.0	71.4	70.2	1065.5	F
acc_reg[7]/D	DFFPOSX1				0.0	1065.5	
acc_reg[7]/CLK	setup			0.0	251.4	1316.9	R
(clock clk)	capture					1000.0	R

and the area report:

Instance	Cells	Cell Area	Net Area	Total Area	Wireload	WL Flag
accu	88	6435.00	0.00	6435.00	<none>	(D)

You will notice that the slack time becomes negative and the circuit does not meet timing. The synthesis Is telling you that the circuit will not run at 1 GHz. (It actually may run at 1 GHz but the tool is pessimistic.)

Try to find the maximum clock frequency that can run the accumulator (it's between 500MHz and 1GHz).

Finally, you can close the GUI by going to **File -> Exit** and you can now analyze the result of the synthesis in the file accu_synth.v that you can use for simulating the netlist and for subsequent place and route using Encounter:

File Edit View Search Tools Documents Help

New Open Save Print Undo Redo Cut Copy Paste Find

accu_synth.v

```
// Generated by Cadence Encounter(R) RTL Compiler v08.10-s108_1

module accu(in, acc, clk, reset);
  input [7:0] in;
  input clk, reset;
  output [7:0] acc;
  wire [7:0] in;
  wire clk, reset;
  wire [7:0] acc;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11, n_12, n_13, n_14, n_15;
  wire n_16, n_17, n_18, n_19, n_20, n_21, n_22, n_23;
  wire n_24, n_25, n_26, n_27, n_28, n_29, n_30, n_31;
  wire n_32, n_33, n_34, n_35, n_36, n_37, n_38, n_39;
  wire n_40, n_41, n_42, n_43, n_44, n_45, n_46, n_47;
  wire n_48, n_49, n_50, n_51, n_52, n_53, n_54, n_55;
  wire n_56;
  DFFPOSX1 \acc_reg[6] (.CLK (clk), .D (n_56), .Q (acc[6]));
  NOR2X1 g5813(.A (reset), .B (n_55), .Y (n_56));
  DFFPOSX1 \acc_reg[5] (.CLK (clk), .D (n_54), .Q (acc[5]));
  DFFPOSX1 \acc_reg[7] (.CLK (clk), .D (n_53), .Q (acc[7]));
  DFFPOSX1 \acc_reg[3] (.CLK (clk), .D (n_52), .Q (acc[3]));
  XNOR2X1 g5818(.A (n_47), .B (n_7), .Y (n_55));
  NOR2X1 g5811(.A (reset), .B (n_50), .Y (n_54));
  NOR2X1 g5812(.A (reset), .B (n_51), .Y (n_53));
  NOR2X1 g5819(.A (reset), .B (n_49), .Y (n_52));
  DFFPOSX1 \acc_reg[4] (.CLK (clk), .D (n_48), .Q (acc[4]));
  XNOR2X1 g5817(.A (n_45), .B (n_5), .Y (n_51));
  XOR2X1 g5816(.A (n_41), .B (n_22), .Y (n_50));
  MUX2X1 g5825(.A (n_40), .B (n_46), .S (n_12), .Y (n_49));
  NOR2X1 g5820(.A (reset), .B (n_44), .Y (n_48));
  OAI21X1 g5822(.A (n_28), .B (n_46), .C (n_36), .Y (n_47));
  DFFPOSX1 \acc_reg[2] (.CLK (clk), .D (n_42), .Q (acc[2]));
  DFFPOSX1 \acc_reg[1] (.CLK (clk), .D (n_39), .Q (acc[1]));
  OAI21X1 g5824(.A (n_31), .B (n_43), .C (n_34), .Y (n_45));
  XNOR2X1 g5826(.A (n_43), .B (n_9), .Y (n_44));
  NOR2X1 g5827(.A (reset), .B (n_38), .Y (n_42));
  OAI21X1 g5823(.A (n_15), .B (n_43), .C (n_24), .Y (n_41));
  INVX2 g5829(.A (n_40), .Y (n_46));
```

Ln 1, Col 1 INS

Adding constraints:

Open the **accu.tcl** file for editing, add this constraint line:

set_attribute avoid true FAX1

This constraint is telling the tool to not use the “Full Adder” cells during the synthesis but instead use gates to build the full adder.

Save and Close

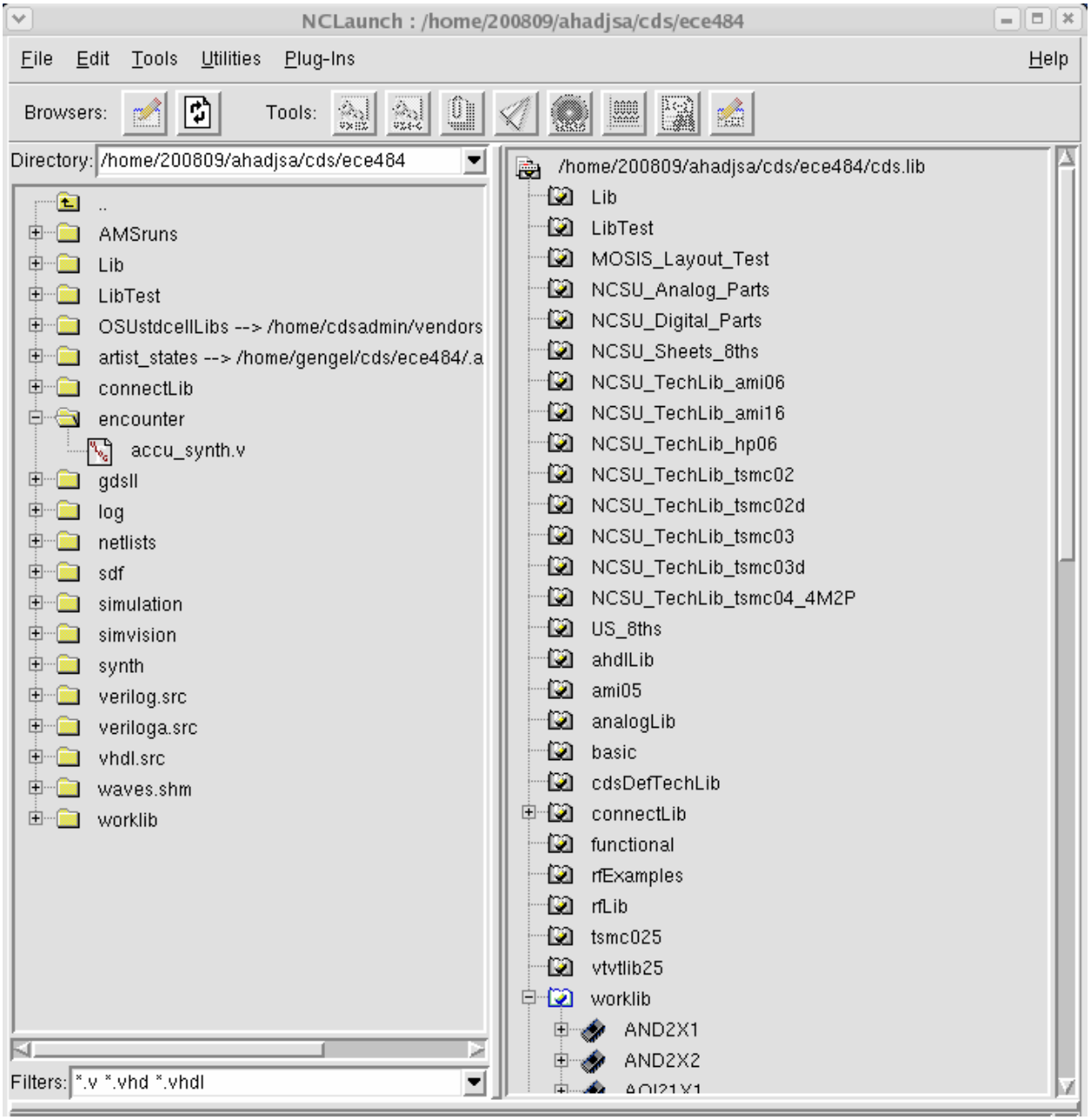
After the synthesis, you will notice that the new schematic does not contain the full adder cell and the number of cells has increased.

Assignment: Post-Synthesis Simulation

Now that you are done with the synthesis, it’s time to simulate the accumulator using the design as implemented using cells from the 0.25 micron standard cell library (use **accu_synth.v**).

Launch NCLaunch simulator by typing the command

sim



Click on the + sign in front of the **verilog.src** directory to expand its contents, then select **accu_synth.v,** right click and open it for edit

In the edit window, add the following line at the beginning of the verilog code:

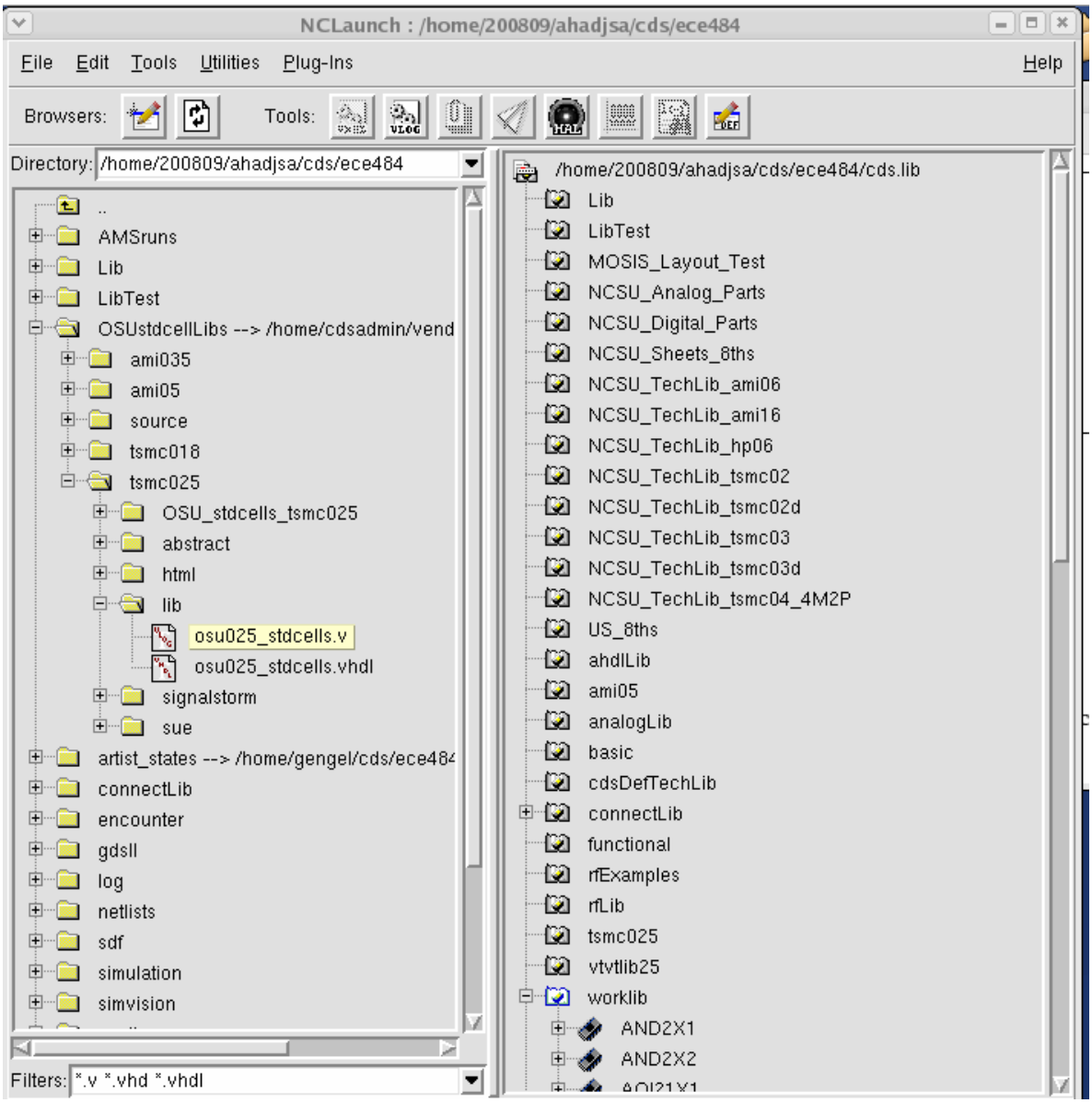
`timescale 1ns/10ps

Save and close the file, then click on the compile button

Now you need to elaborate, Click on the + sign in front of the **worklib,** select **accu_tb** and then click on the **Elaborate** button in the Menu (immediately to the right of the VLOG button).

You will notice that the **Console Window** (bottom) is showing some errors which you need to fix!!!!!

To do so, go the **OSUstdcellLibs/tsmc025/lib** library, select the **osu025_stdcells.v** file, and then click on the compile button. This will compile the Verilog descriptions for all of the standard cells. The descriptions include propagation delay information for each of the cells.

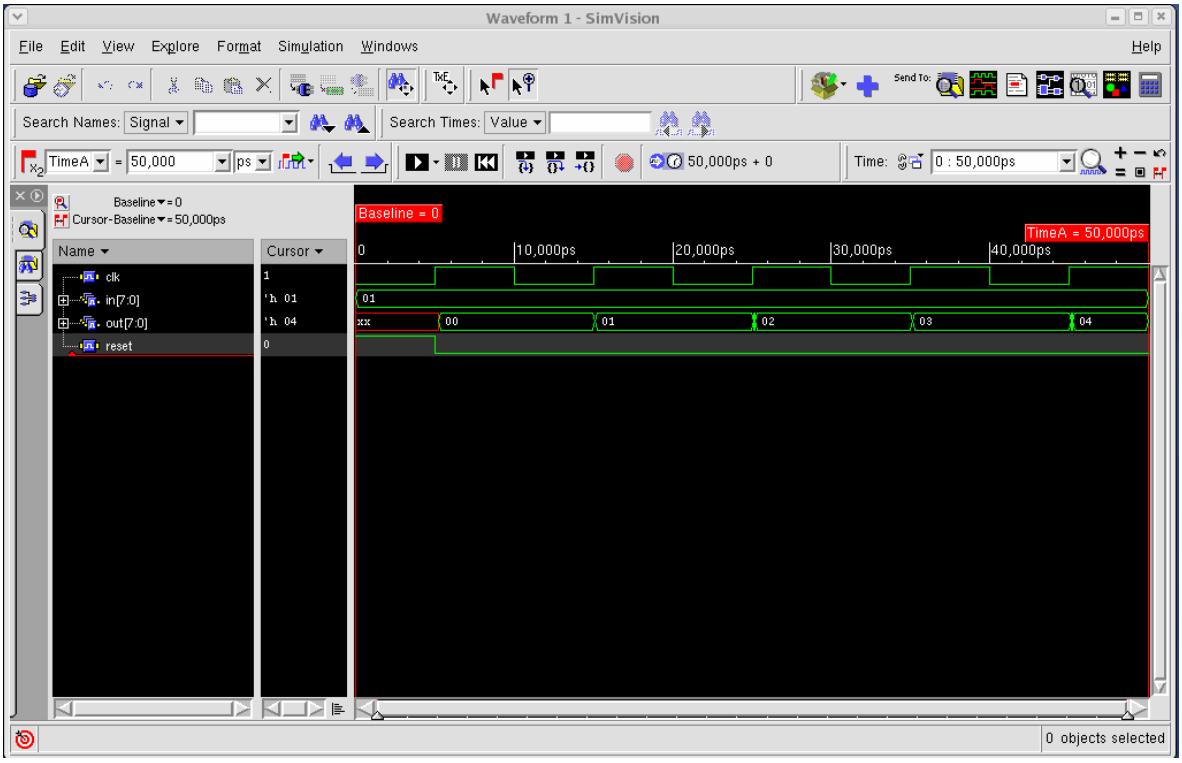


Elaborate **accu_tb.**

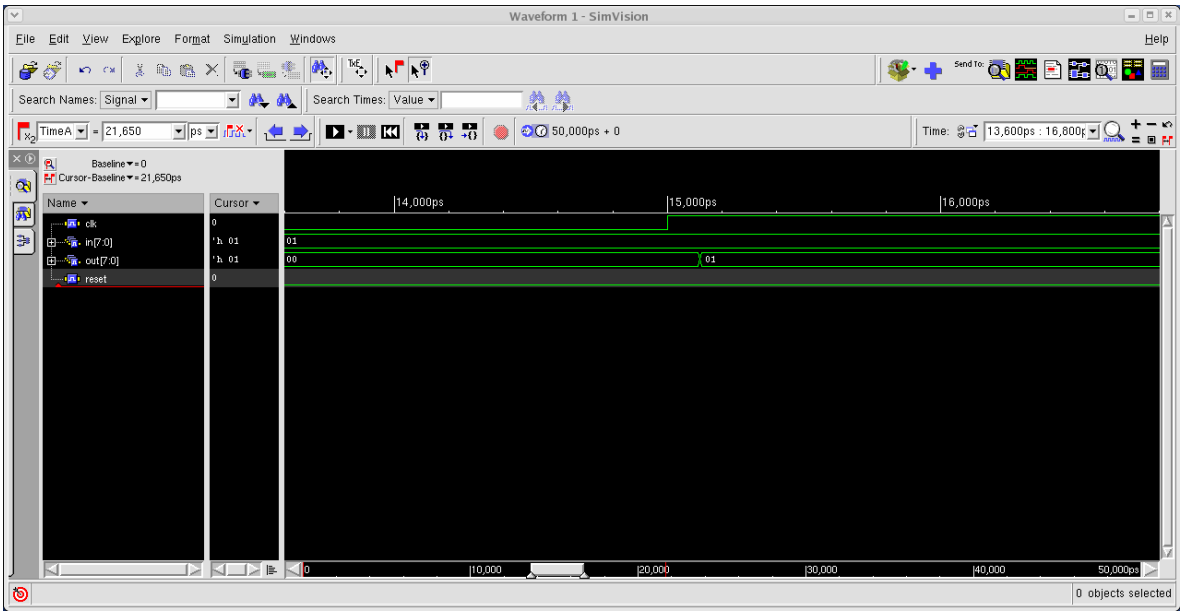
Check the **Console Window** for errors.

To simulate, click on the + sign in front of the **Snapshots** library to expand its contents, then select **worklib.accu_tb:module** and click on Simulate (next to the right of elaborate).

The signal waveforms display should look like this:



Notice that if you zoom in the waveforms, you will see that the transition of the output from one value to another doesn't occur at the positive edge of the clock, this delay in introduced by the cells.



Congratulations, this is the end of the Logic Synthesis Tutorial.