

EXPERIMENT NO	:	Lab 5 (Duration : 2 hours)
EXPERIMENT TITLE	:	RTL-to-Gates Level Synthesis of Counter using Cadence RTL Compiler
OBJECTIVE	:	<ol style="list-style-type: none">1. Code a Counter in Verilog and simulate using Cadence Incisive Simulator2. Synthesize the RTL codes to gates using Cadence RTL Compiler3. Perform Post-synthesis simulation of design

Exercise 1 : To simulate a **4-Bit Counter** in RTL codes.

- (i) Use editor, **gedit**, to view the codes
- (ii) Use Cadence Incisive Simulator to simulate the design

1. Open a new terminal and type: **pwd**
This linux command will show the present working directory.
cd term2
Change directory to “term2”
source cshrc
Run a script file that will setup a proper environment so that various software can be launched.
2. Type **nclaunch&** to start the GUI as shown in Figure 1. Select **counter.v** and **right-click→Edit**.

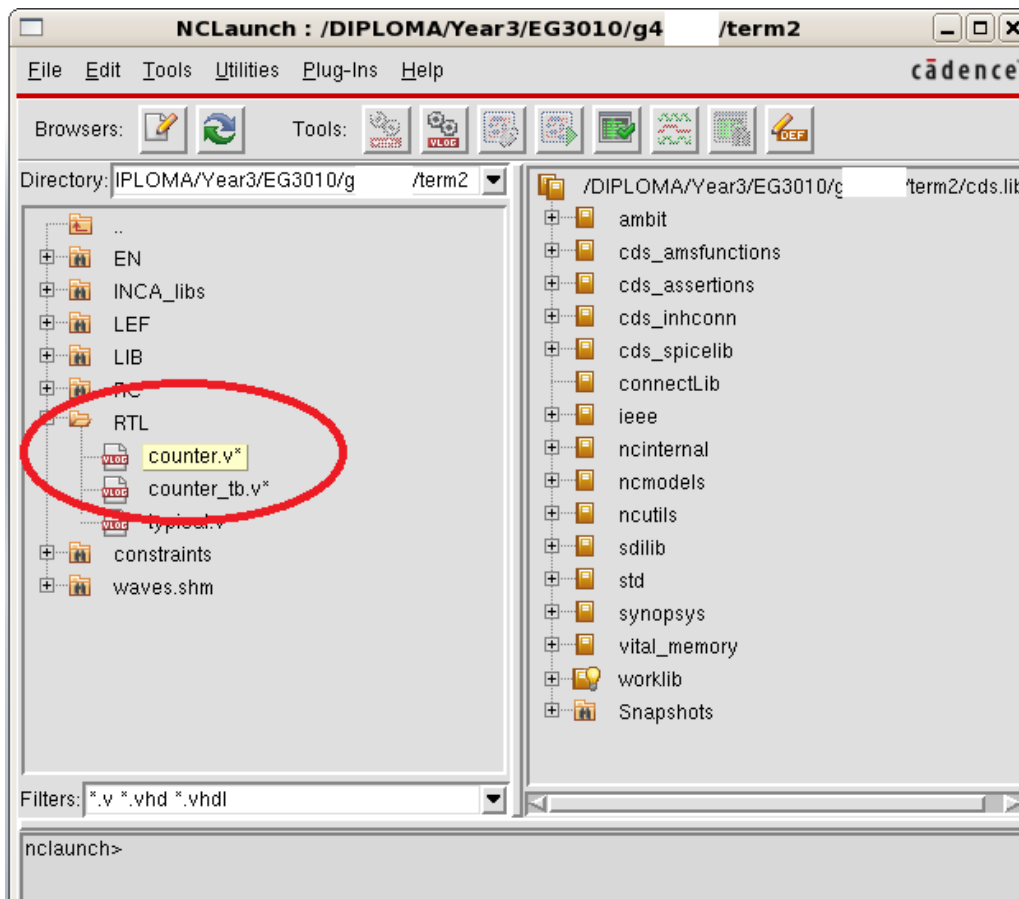
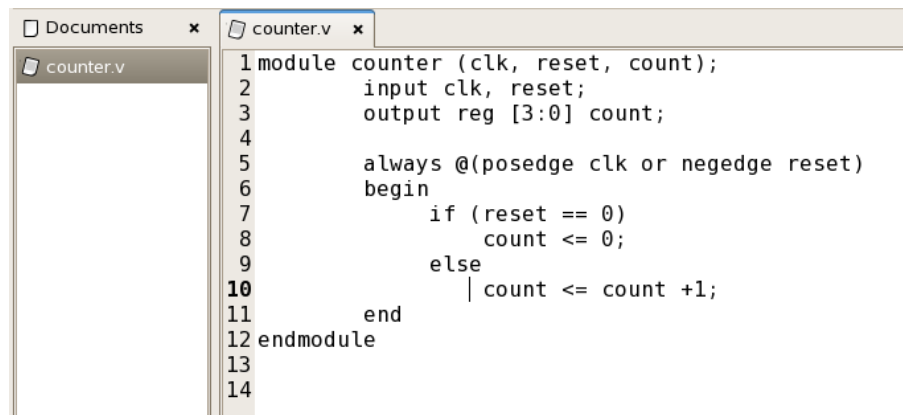


Figure 1 NCLaunch Window

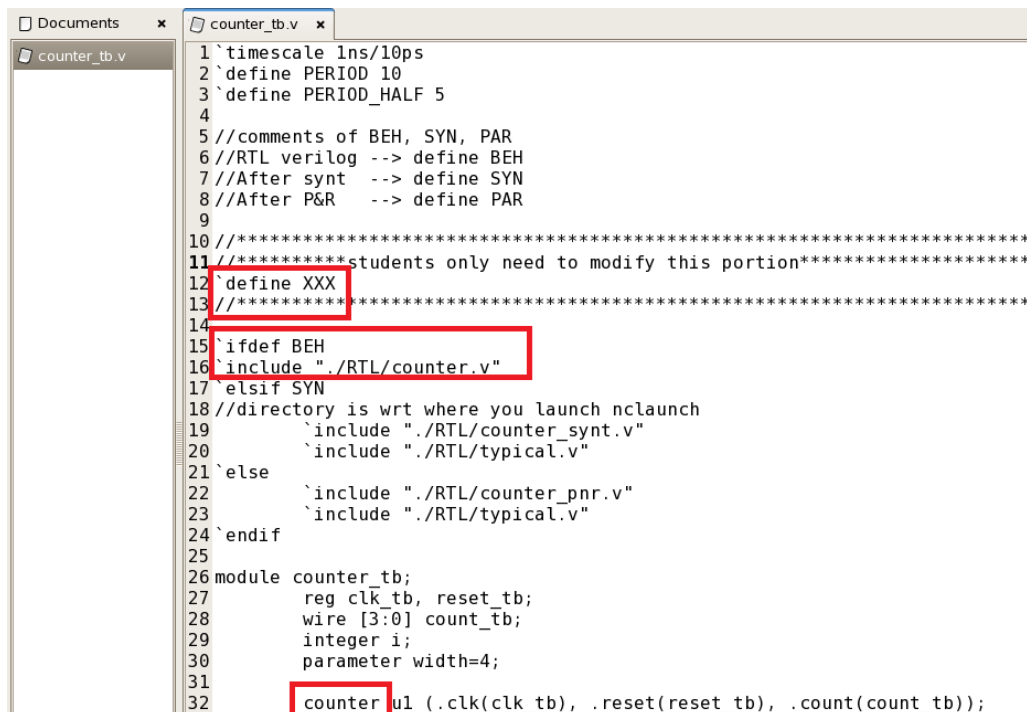
3. Figure 2 shows the verilog code of the counter. It has 4 bits output named as **count**. Read the code to gain some understanding about the counter's behavior. As you can see, **count** will only change either at positive-edge of **clock** or negative-edge of **reset**. Close the file.



```
1 module counter (clk, reset, count);
2     input clk, reset;
3     output reg [3:0] count;
4
5     always @(posedge clk or negedge reset)
6     begin
7         if (reset == 0)
8             count <= 0;
9         else
10            | count <= count +1;
11    end
12 endmodule
13
14
```

Figure 2 Verilog Code for 4-bit Counter

4. Select **counter_tb.v** and **right-click**→**Edit**. Figure 3 shows the test bench. Read the code to understand the test bench. Replace “**XXX**” with “**BEH**” for this exercise. When the variable **BEH** is defined, the file “./RTL/counter.v” will be read/included by the simulator as it contains the module **counter** to be tested. Save and close the file.



```
1 `timescale 1ns/10ps
2 `define PERIOD 10
3 `define PERIOD_HALF 5
4
5 //comments of BEH, SYN, PAR
6 //RTL verilog --> define BEH
7 //After synt --> define SYN
8 //After P&R --> define PAR
9
10 //*****students only need to modify this portion*****
11 //*****
12 `define XXX
13 //*****
14
15 `ifdef BEH
16 `include "./RTL/counter.v"
17 `elsif SYN
18 //directory is wrt where you launch nlaunch
19     `include "./RTL/counter_synt.v"
20     `include "./RTL/typical.v"
21 `else
22     `include "./RTL/counter_pnr.v"
23     `include "./RTL/typical.v"
24 `endif
25
26 module counter_tb;
27     reg clk_tb, reset_tb;
28     wire [3:0] count_tb;
29     integer i;
30     parameter width=4;
31     counter u1 (.clk(clk_tb), .reset(reset_tb), .count(count_tb));
32
```

Figure 3 Verilog Code for Test bench

5. Select **counter.v** and **counter_tb.v** (to select, press **Ctrl & click** on the file, as shown in Figure 4). **Right-click**→**NCVlog**. Then click **OK**. This will compile the verilog files to C code for faster simulation.

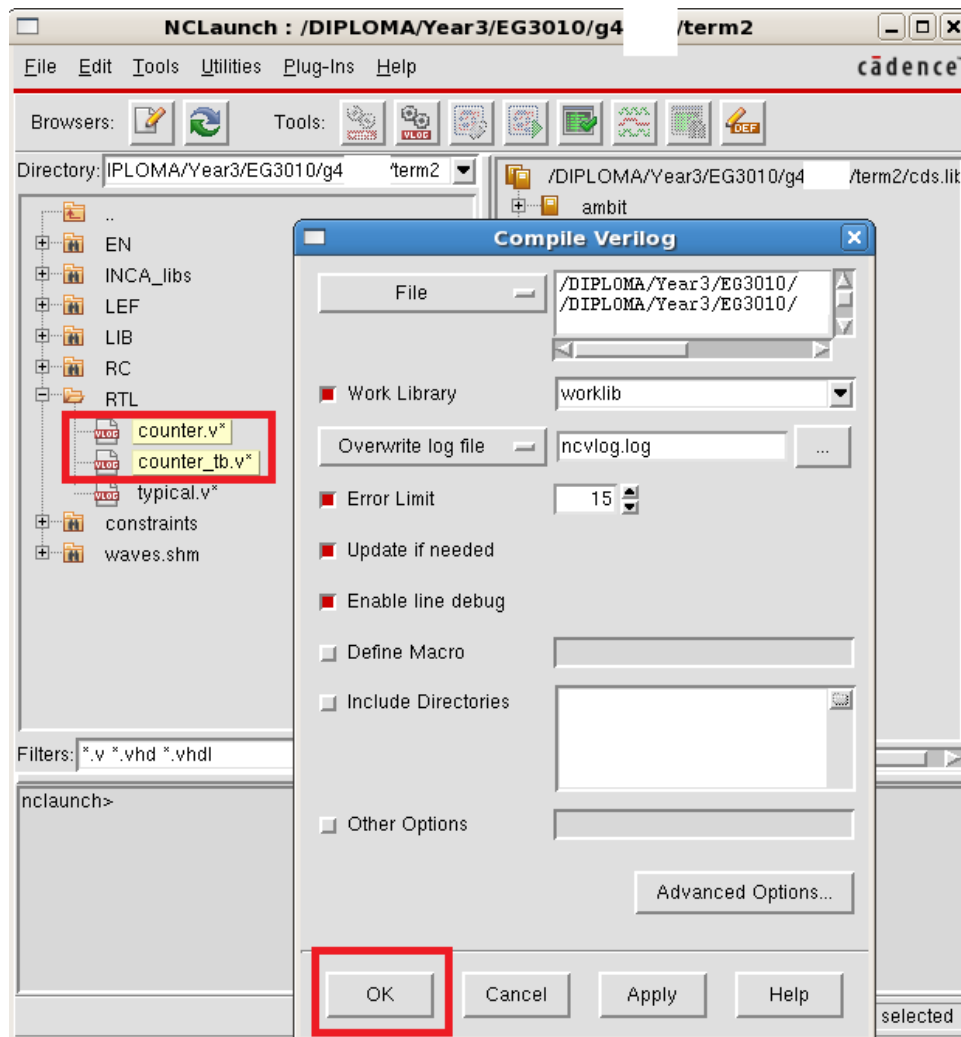


Figure 4 NCLaunch Window

6. Select **worklib/counter_tb** (as shown in Figure 5). **Right-click**→**NCElab**. Then click **OK**. This will link up all the related compiled modules for simulation.

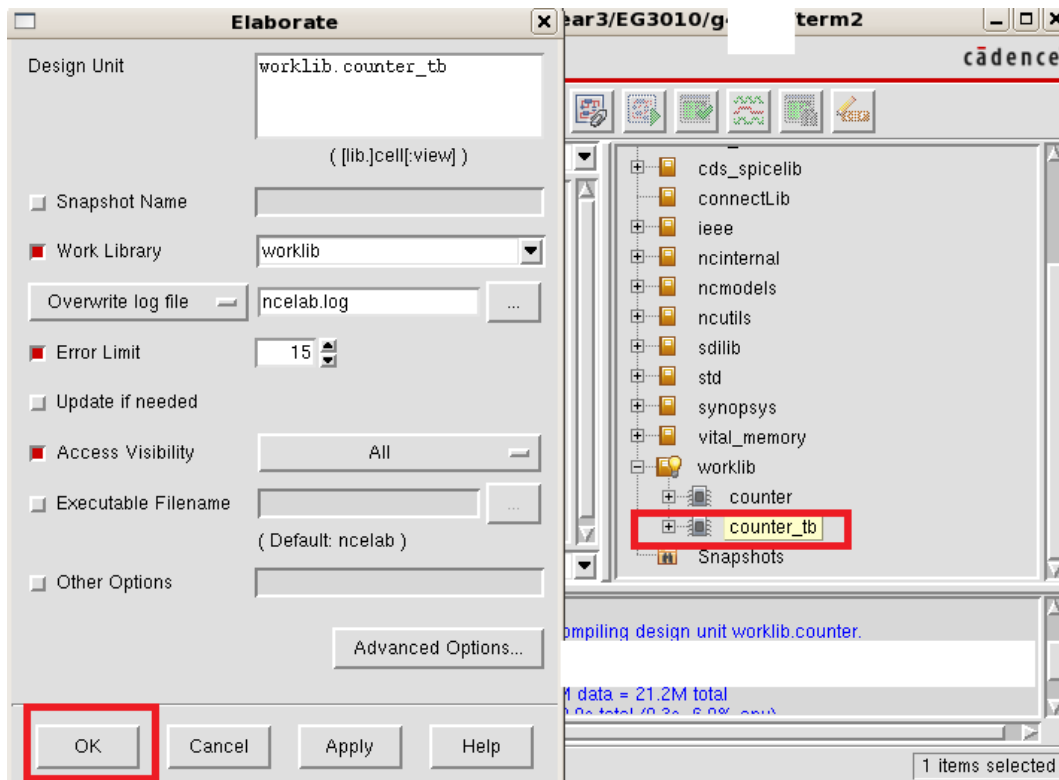


Figure 5 NCElab Window

7. Select **Snapshots/worklib.counter_tb:module** (as shown in Figure 6). **Right-click**→**NCSim**. Then click **OK**. This will allow you to setup the environment for simulation.

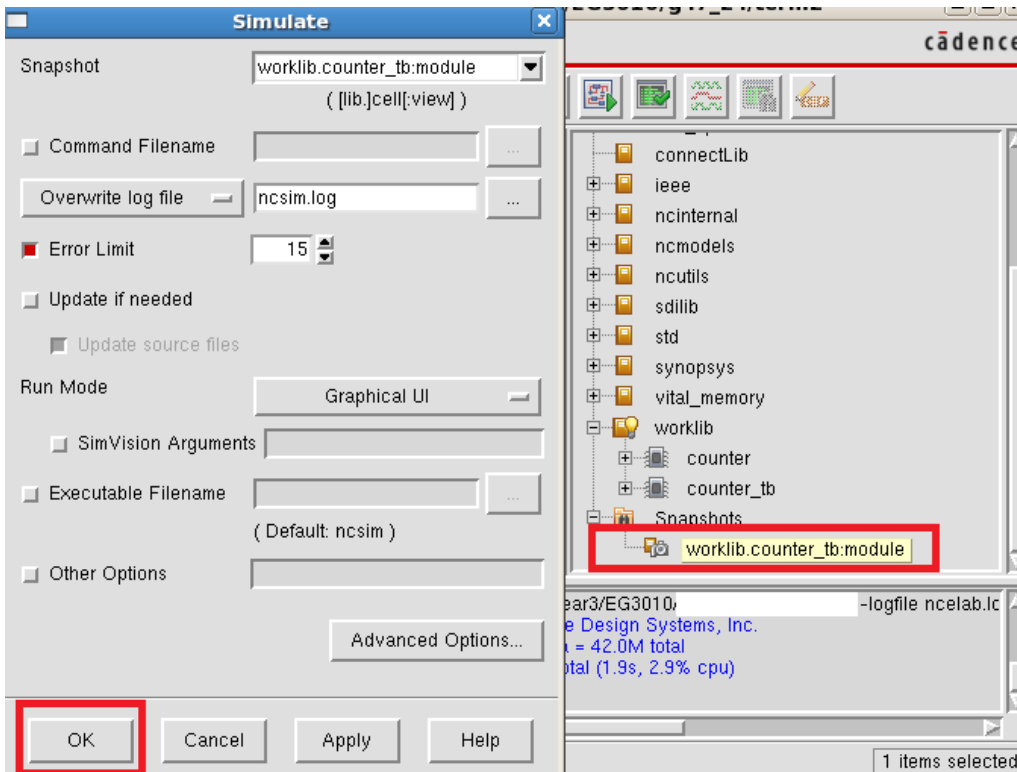


Figure 6 NCSim Window

8. Select **counter_tb** (as shown in Figure 7). **Right-click**→**Send To New**→**Waveform Window**. The related signals of counter_tb will be probed for display.

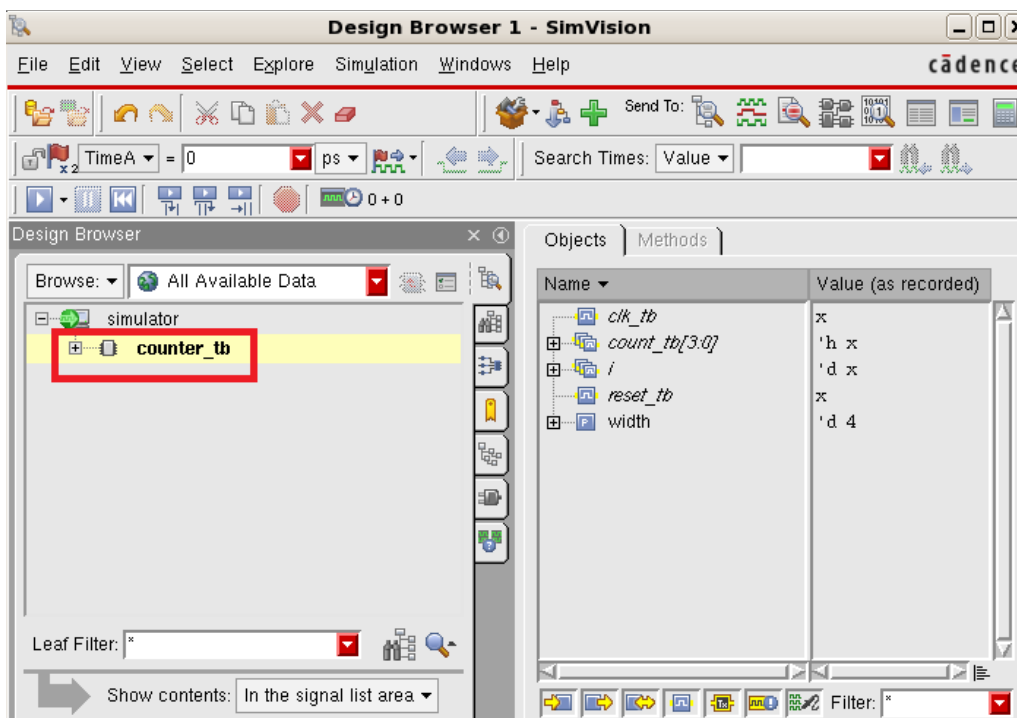


Figure 7 SimVision/Design Browser Window

9. Select **count_tb[3:0]** (the output of counter as shown in Figure 8). **Simulation**→**Run**. Then **View**→**Zoom**→**Full X**. Use the **Sliding Bar** and **View**→**Zoom**→**In X** (or **Alt-i**) to zoom in to **count_tb[3:0]=F** as shown. Click on the waveform screen to bring TimerA to clock edge as shown. There is no delay between the positive-edge of **clk_tb** and output of counter since the counter is modeled by verilog behavioral code.

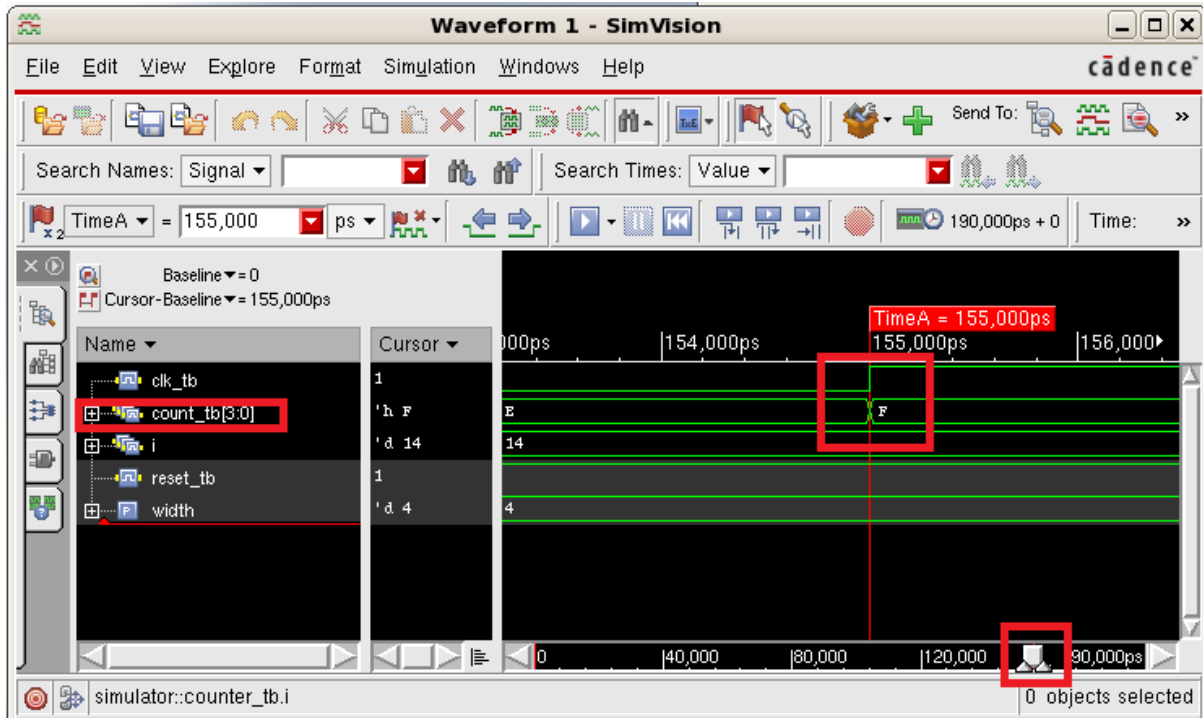


Figure 8 SimVision/Waveform Window

10. Exit all the software.

Exercise 2 : To synthesize the 4-Bit Counter, of Exercise 1, using Cadence Software – RTL Compiler.

1. Open a new terminal and type:

pwd

If you're not in your home directory, do a ***cd ~*** to change to your home directory
...g4x_xx.

cd term2

source cshrc

cd RC

2. Type ***rc*** to launch the RTL Compiler. You should see the RC-shell prompt as shown in Figure 9.

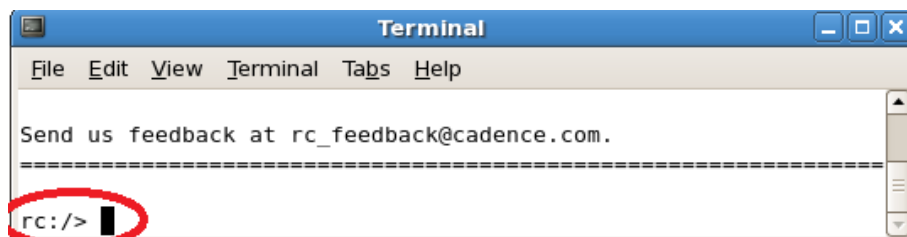


Figure 9 RC-shell Prompt

3. Type:

source counter.tcl

counter.tcl contains the instructions for the setup of various libraries, constraints and etc. for the software to produce the netlist from your behavioral verilog code.

- Type:
gui_show
This will show you the schematic of the synthesized circuit. 4 registers, 2 gates and 1 inverter are used to realise the function of the 4-bit counter.

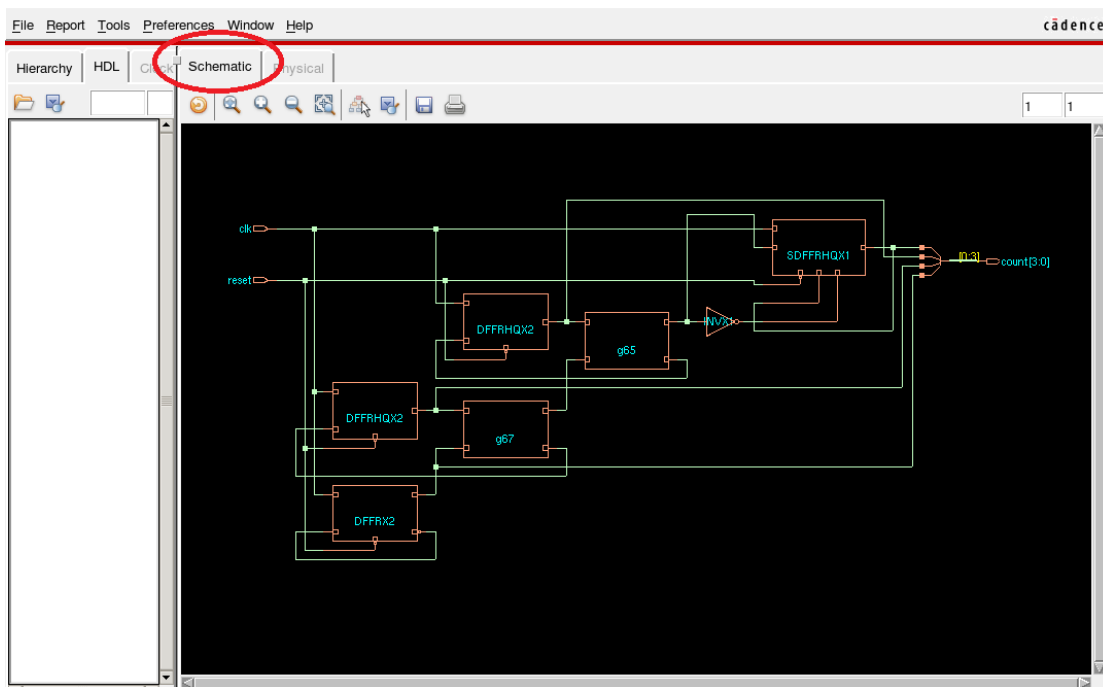


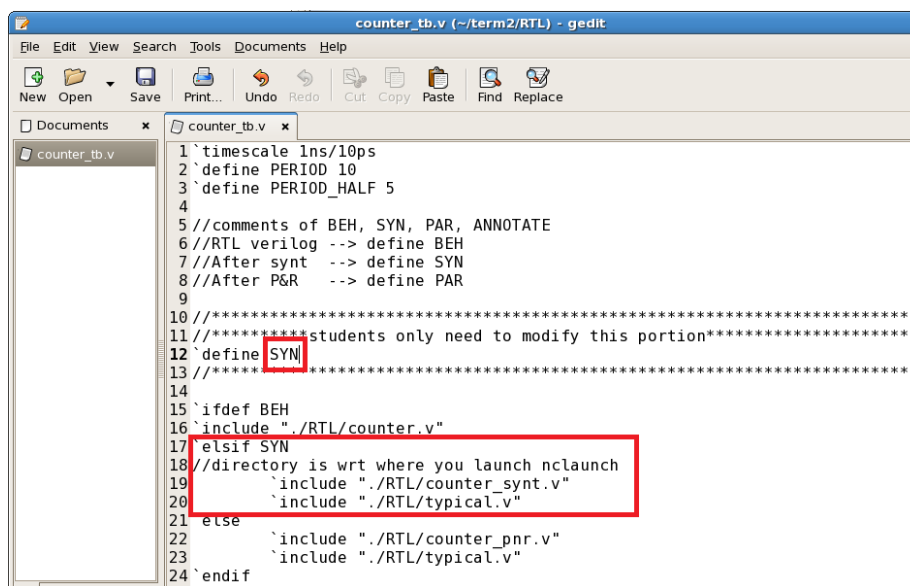
Figure 10 Schematic

- Close the window. You can type **gui_hide** to close the window.
Type **exit** to exit RTL Compiler.
- Type:
pwd to know which directory you are in.
You can **cd ..** to go up to the parent directory or **cd ~** to your home directory if you get lost.
Make sure you are in **.../term2/RC**, type **more counter.tcl** to view the setup of the synthesis run you have just performed. A lot of knowledge and expertise are required to do a synthesis for complex digital circuit. Use **space bar key** to scroll through the file.
- The synthesized netlist file is in **.../term2/RTL**.
Go to that directory and do a **more counter_synt.v** to view the netlist. You can see the 7 gates used for the 4-bit counter.

This exercise is optional:

Exercise 3 : Post-synthesize simulation using Cadence Incisive Simulator

1. Type:
cd .. to go up to .../term2.
nclaunch&
2. **Right-click→Edit** on counter_tb.v
Change the variable to **SYN** as shown in Figure 11. Save and close the file.
typical.v is the standard cell library that contains the modules of the gates that we're going to use.



```
1 `timescale 1ns/10ps
2 `define PERIOD 10
3 `define PERIOD_HALF 5
4
5 //comments of BEH, SYN, PAR, ANNOTATE
6 //RTL verilog --> define BEH
7 //After synt --> define SYN
8 //After P&R --> define PAR
9
10 //*****students only need to modify this portion*****
11 //*****students only need to modify this portion*****
12 `define SYN
13 //*****
14
15 `ifndef BEH
16 `include "./RTL/counter.v"
17 `elsif SYN
18 //directory is wrt where you launch nclaunch
19 `include "./RTL/counter_synt.v"
20 `include "./RTL/typical.v"
21 `else
22 `include "./RTL/counter_pnr.v"
23 `include "./RTL/typical.v"
24 `endif
```

Figure 11 Test Bench

3. We are going to simulate the test bench like what we did in Exercise 1. Refer to it if you not sure.
Select **counter_synt.v**, **counter_tb.v** and **typical.v** and **right-click**→**NCVlog** (Figure 12).

*Note : This may take a while for the **typical.v** file to be compiled.*

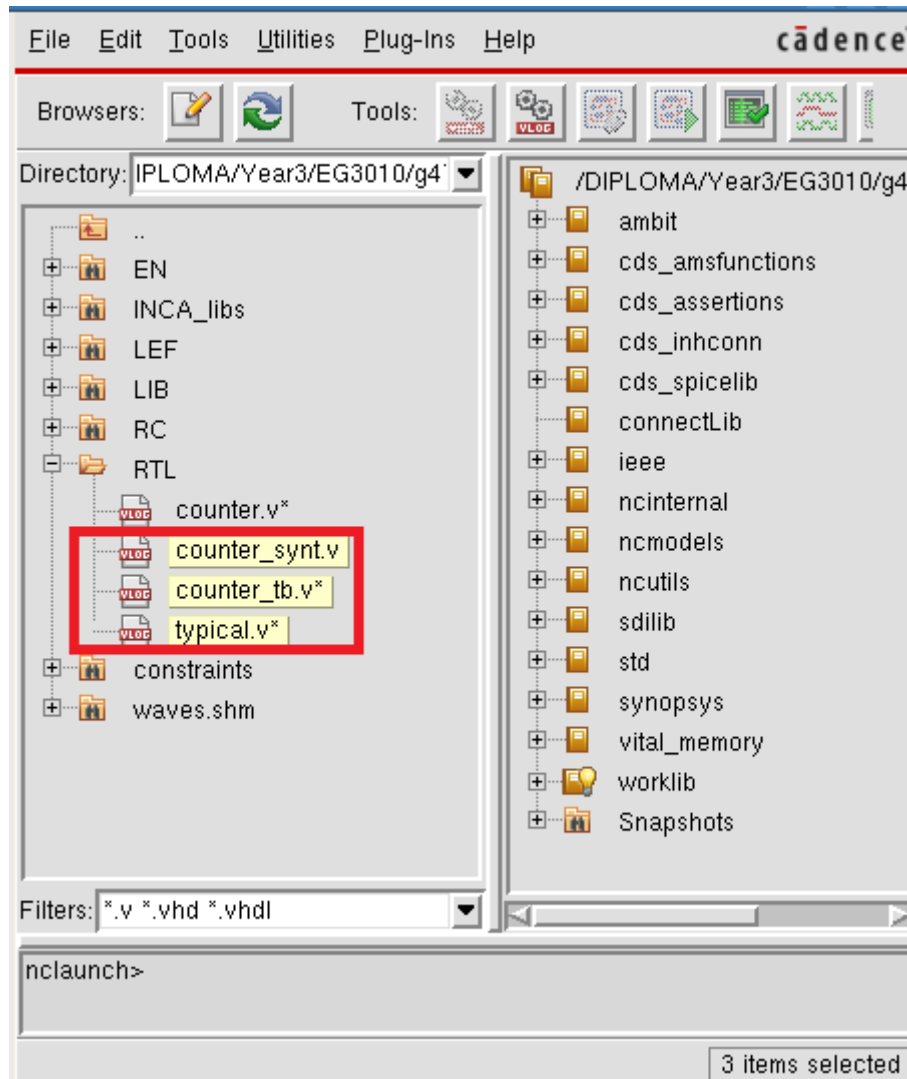


Figure 12 NCLaunch Window

4. Select **worklib/counter_tb** and **right-click**→**NCElab**.
 Select **Snapshot/worklib.counter_tb:module** and **right-click**→**NCSim**.
 Select **counter_tb** and **right-click**→**Send To New**→**Waveform Window**.
 Select **count_tb[3:0]**. **Simulation**→**Run**.
 Zoom in to **count_tb[3:0]=F** as shown in Figure 13.
 Click on waveform to bring **TimerA** to the desired location.
Right-click→**Create a marker** to put additional marker.

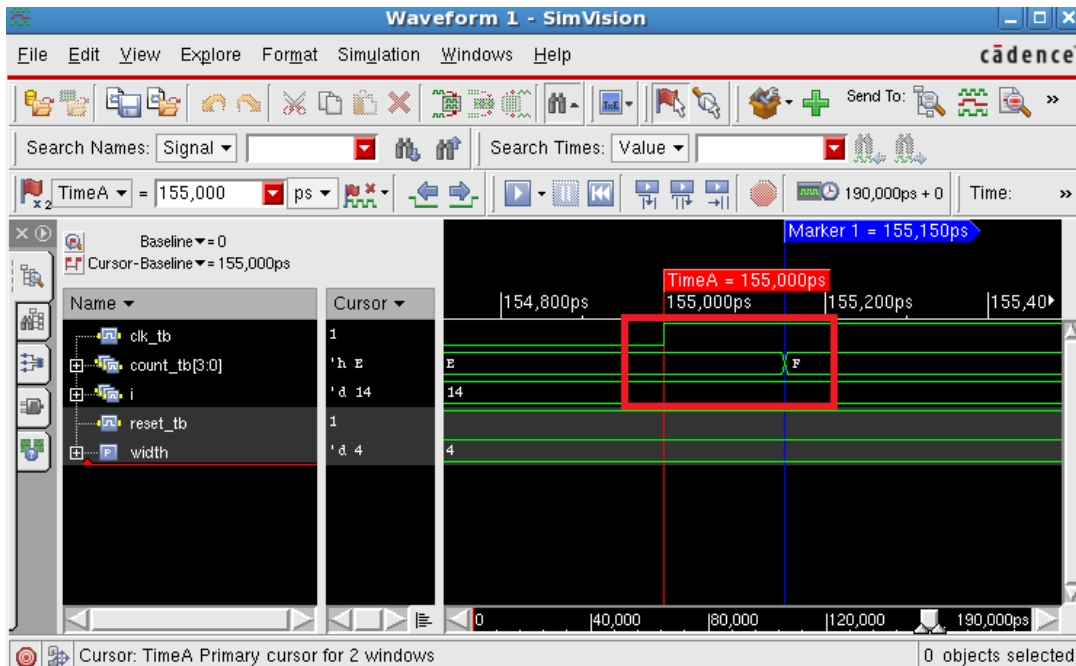


Figure 13 SimVision/Waveform Window

5. As you can see, there is delay of **150ps** between positive-edge of **clk_tb** and counter's output. This is due to the gate delay since now we are simulating a real circuit.

EXPERIMENT NO	:	Lab 6 (Duration : 2 hours)
EXPERIMENT TITLE	:	Place and Route of Counter using Cadence Encounter
OBJECTIVE	:	To perform place and route of Counter

Exercise 1

To do place and route of the counter synthesized in previous lab using Cadence software - Encounter.

1. Open a new Terminal. Type :
pwd (show present working directory)
Cd (change directory to home directory)
cd term2 (change directory to term2 directory)
source cshrc (execute a setup file called cshrc)
2. Change directory. Type :
cd EN
encounter to launch the software. You should see Figure 1.

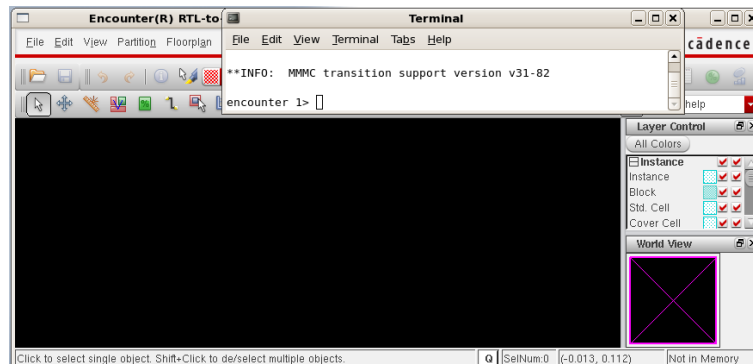


Figure 1 Encounter-shell Prompt

3. In Encounter GUI, **File**→**Import Design**.
As shown in Figure 2, click **Load**, select **counter.globals** and click **Open**. Click **OK** to import the design.

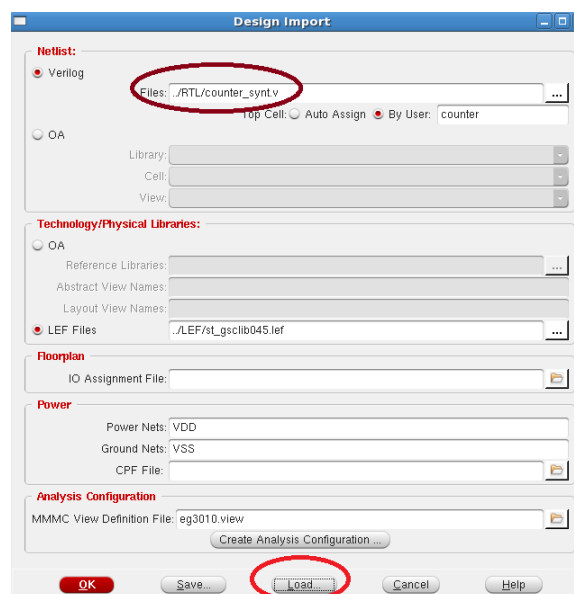


Figure 2 Import Design

4. Hit **f** to see the full view. Click on the icon (red circle) as shown in Figure 3.

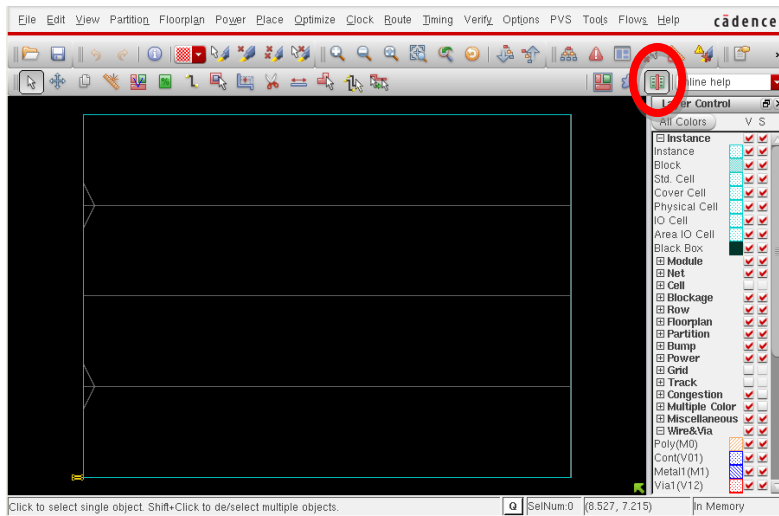


Figure 3 Ready for Floorplanning

5. Floorplanning:
Floor plan→**Specify Floorplan**. Key in the parameters as shown in Figure 4 and click **OK**.

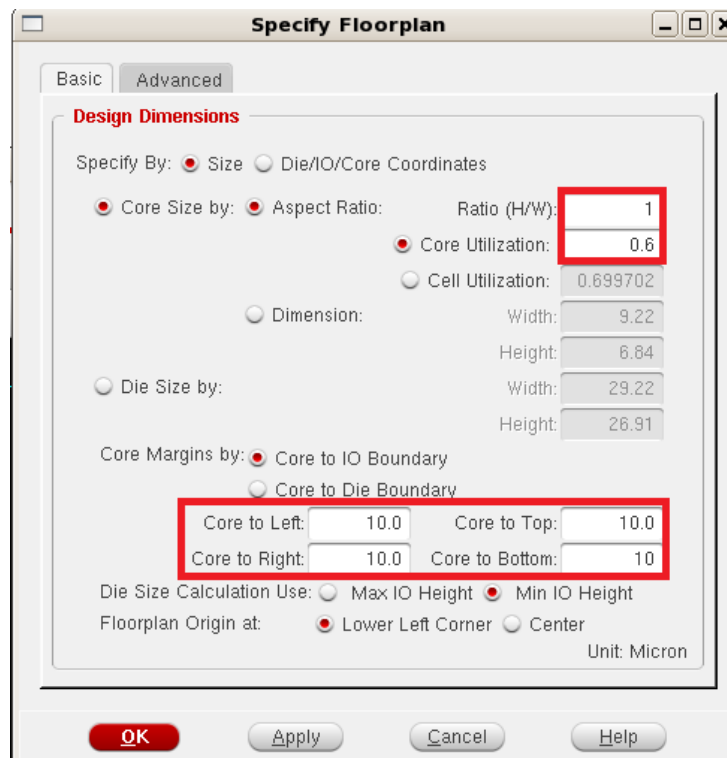


Figure 4 Floorplanning

6. We are going to add 2 supply rings for VDD and VSS.
Power→**Power Planning**→**Add Ring**. Key in the parameters as shown in Figure 5 and click **OK**.

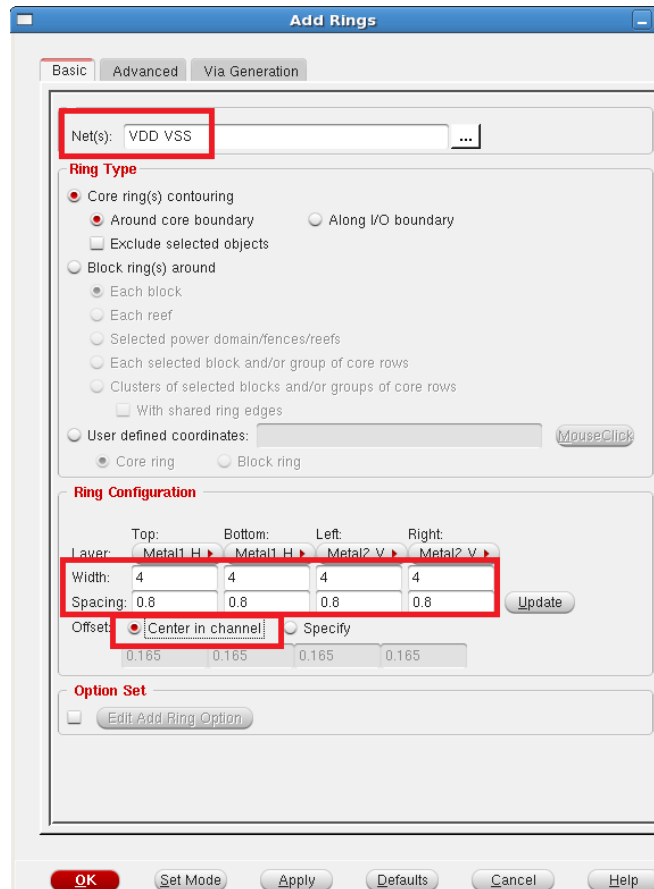


Figure 5 Add Power Ring

7. The layout should be the same as shown in Figure 6. It is time to save the work thus far. **File**→**SaveDesign**, check **Encounter** and call it **step1.enc**
Click **OK**.

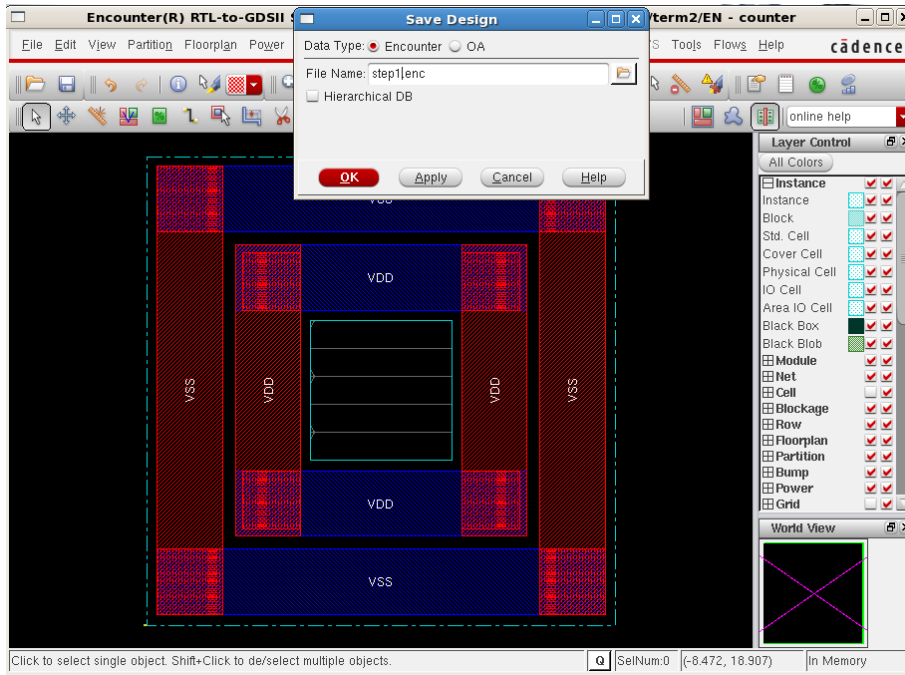


Figure 6 Save Design

8. Before we proceed further, we need to instruct Encounter to connect signals VDD or VSS, and signals that connect to logic high or logic low, to supply ring VDD and VSS respectively. This can be done via the **Encounter-shell** as shown in Figure 7.

After typing and finish the first line of command, hit **ENTER**. To save effort in typing, hit **UP Arrow Key** to bring up the previous command for modification. Key in the 4 commands as shown in the figure.

```

0.000M) ***
encounter 1> globalNetConnect VDD -type pppin -pin VDD -all
encounter 2> globalNetConnect VSS -type pppin -pin VSS -all
encounter 3> globalNetConnect VDD -type tiehi
encounter 4> globalNetConnect VSS -type tielo
encounter 5> █

```

Figure 7 Power Pins Connection

9. Gates are placed in the layout row-by-row. We need to provide supply rails for them. **Route→Special Route**. Key in parameters as shown in Figure 8.

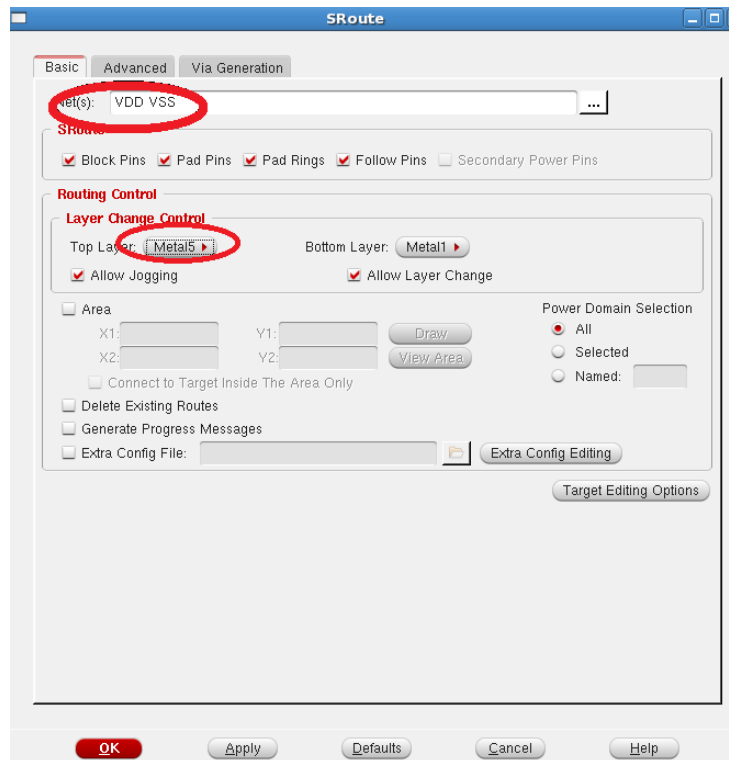


Figure 8 Supply Rails for Cells

10. Perform placement:
Place→**Place Standard Cell**. The cells should be placed as shown in Figure 9. If not,
View→**Redraw**.

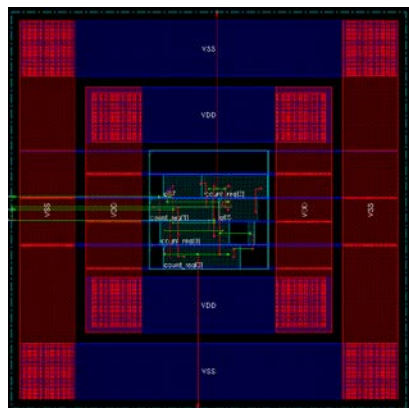


Figure 9 Placement

11. Routing
Route→**Nanoroute**→**Route** to complete the routing.
Place→**Physical Cell**→**Add Filler** and select **FILL1**, **FILL1A**, **FILL2**, **FILL4**, **FILL8** as shown in Figure 11, by using **Ctrl + Click** and **Add**. This will fill up the empty space.



Figure 10 Add Filler

12. **File**→**Save Netlist**.

Navigate to ../RTL and provide file name of **counter_pnr.v**
The netlist will be saved as ../RTL/counter_pnr.v

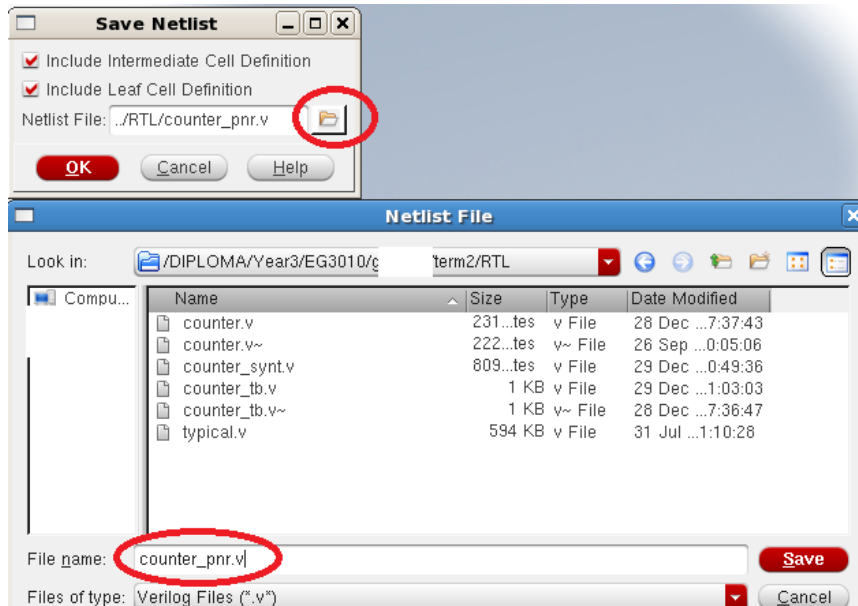


Figure 11 Save Netlist

13. **Timing**→**Extract RC** (as shown in Figure 12) to extract R and C of all the wires.

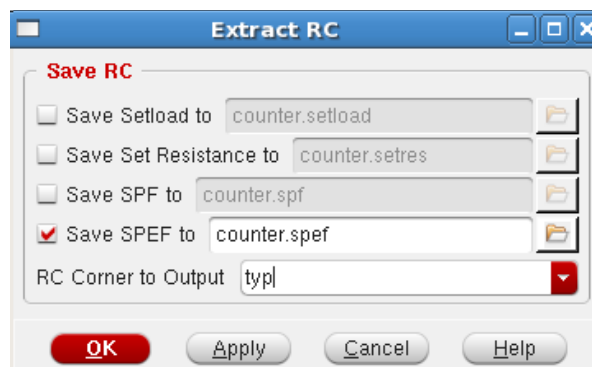


Figure 12 RC Extraction

14. From the Extracted RC file, a file that describes delay of all the wires can be produced.

Timing→**Write SDF.**

As shown in Figure 13, change to directory **RTL** and provide a file name of **counter.sdf**

Click **Save** and then **OK.**

We are ready to simulate the circuit again with the delay of wires included. You can exit all software.

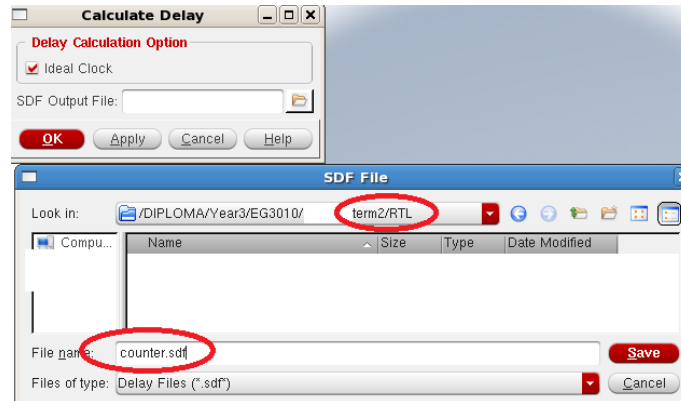


Figure 13 Delay Calculation

14. This is the complete layout produce by the software.
File→**SaveDesign** to **counter.enc**

Exercise 2 : Post-layout simulation

1. We are going to simulate the circuit again but this time with all the delay of wiring (post-layout) taken care of. Refer to Lab5 Exercise 1 if you are not sure about the commands of the simulator.
2. Type :
cd
cd term2
source cshrc
nclaunch&
3. Select **RTL/counter_tb.v** and **right-click→Edit**.
For the statement “define XXX”, replace it with “define **PAR**”. Save and close the file.
Select **counter_pnr.v** and **counter_tb.v** and **right-click→NCVlog**.
4. Select **worklib/counter_tb** and **right-click→NCElab**.
Select **Snapshot/worklib.counter_tb:module** and **right-click→NCSim**.
Select **counter_tb** and **right-click→Send To New→Waveform Window**.
Select **count_tb[3:0]**. **Simulation→Run**.
Zoom in to **count_tb[3:0]=F** as shown in Figure 14.
Click on waveform to bring **TimerA** to the desired location.
Right-click→Create a marker to put additional marker.

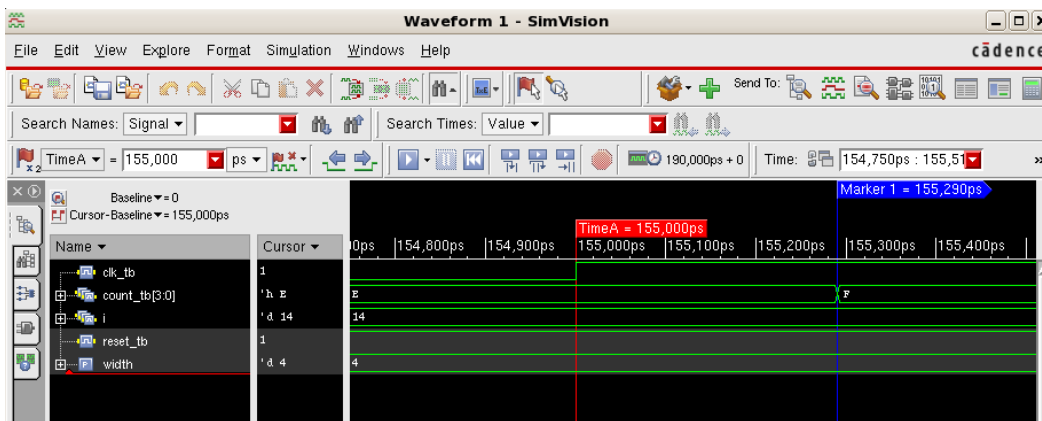


Figure 14 Result

5. As you can see, there is delay of **290ps** between positive-edge of **clk_tb** and counter's output. This is due to the gate delay and wire delay, since now we are simulating a post-layout netlist.

EXPERIMENT NO	:	Lab 7 (Duration : 2 hours)
EXPERIMENT TITLE	:	Place and Route of Shifter using Cadence Encounter
OBJECTIVE	:	To perform place and route of Shifter

Exercise 1

To do place and route of the Shifter synthesized in Lab9 using Cadence Encounter

1. Open a new Terminal
Type :
cd
cd term2
source cshrc
2. Change directory
Type :
cd EN
encounter to launch the software.
3. In Encounter GUI, **File→Import Design**.
Click **Load**, select **shifter_top.globals** and click **Open. OK** to load the design.
4. Floorplanning:
Floor plan→Specify Floorplan.
Ratio=1, Core Utilization=0.6, Core to Left/Right/Top/Bottom=10.
click **OK**.
5. We are going to add 2 supply rings for VDD and VSS.
Power→Power Planning→Add Ring.
Nets: VDD VSS
Width=4, Spacing=0.8, Center in Channel.
Click **OK**.
7. Before we proceed further, we need to instruct Encounter to connect signals VDD or VSS, and signals that connect to logic high or logic low, to supply ring VDD and VSS respectively.

This can be done via the **Encounter-shell** as shown in Figure 1. Hit **ENTER** if there is no encounter prompt.

```
0.000M) ***
encounter 1> globalNetConnect VDD -type pgpin -pin VDD -all
encounter 2> globalNetConnect VSS -type pgpin -pin VSS -all
encounter 3> globalNetConnect VDD -type tiehi
encounter 4> globalNetConnect VSS -type tielo
encounter 5> █
```

Figure 1 Power Pins Connection

8. Gates are placed in the layout row-by-row. We need to provide supply rails for them.
Route→Special Route.
Nets: VDD VSS
Top Layer: Metal5
9. Perform placement:
Place→Place Standard Cell.
10. Routing:
Route→Nanoroute→Route to complete the routing.
Place→Physical Cell→Add Filler and select **FILL1, FILL1A, FILL2, FILL4, FILL8**
The complete layout produce by the software is shown in Figure 2.
File→SaveDesign to **shifter_layout.enc**
Maximize the layout window. Hit **Print_scrn** to save the screen/result as **shifter_layout.png** in the desktop.

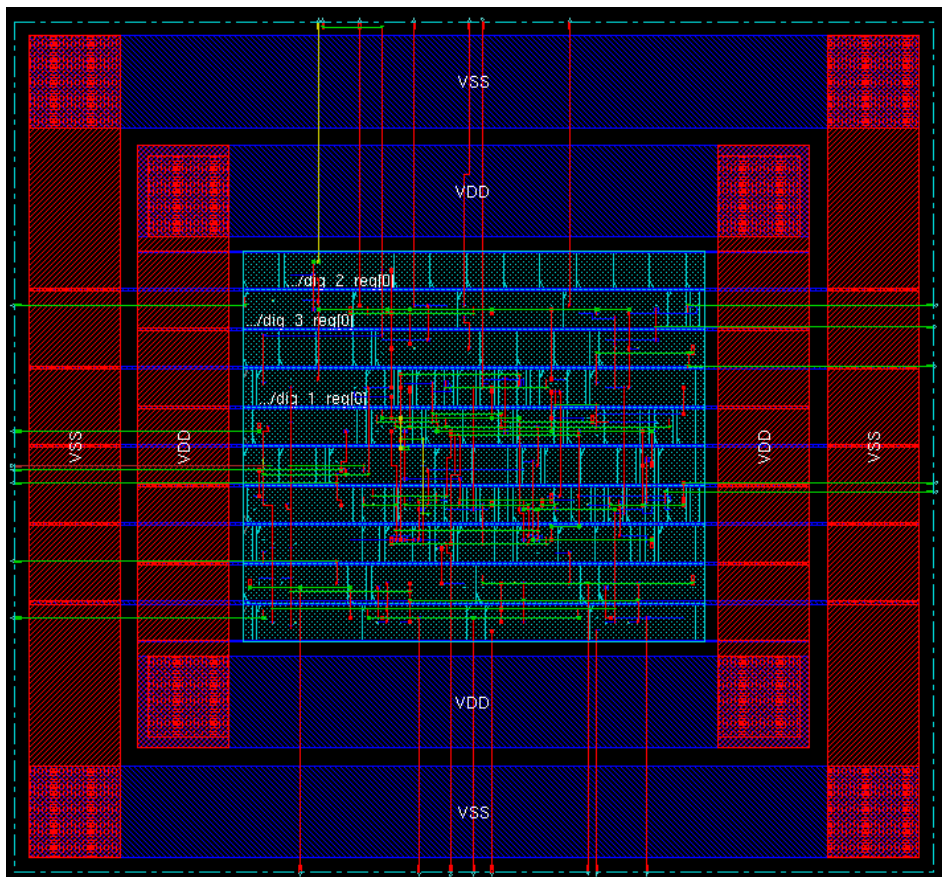


Figure 2 Complete Layout

11. **File→Save Netlist** to save the netlist as **../RTL/shifter_top_pnr.v**
Timing→Extract RC. Remember to check 'save SPEF ...'.
Timing→Write SDF. Name the file as **../RTL/shifter_top_pnr.sdf** (in **/RTL** directory).
We are ready to simulate the circuit again with the delay of wires included.
You can exit all the software.

Exercise 2 : Post-layout simulation

1. Type:
cd .. to go up to .../term2.
nclaunch&
2. **Right-click**→**Edit** on RTL/shifter_top_tb.v
Change the variable to **PAR**.
Save and **close** the file.
3. Select **shifter_top_pnr.v**, **shifter_top_tb.v** and **typical.v** and **right-click**→**NCVlog**
*Note : This may take a while for the **typical.v** file to be compiled.*
4. Select **worklib/shifter_top_tb** and **right-click**→**NCElab**.
Select **Snapshot/worklib.shifter_top_tb:module** and **right-click**→**NCSim**.
Select **shifter_top_tb** and **right-click**→**Send To New**→**Waveform Window**.
Simulation→**Run**.
Click on waveform to bring **TimerA** to the desired location.
Right-click→**Create a marker** to put additional marker.

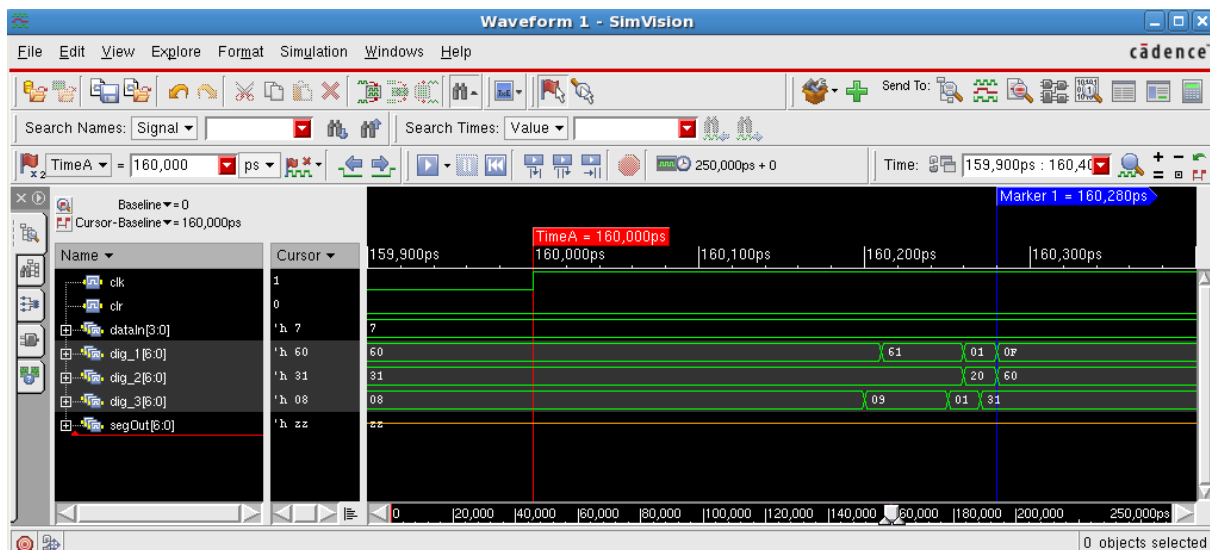


Figure 3 Result

5. As you can see, there is a delay of **280ps** between positive-edge of **clk** and shifter's output (when signal **dig_1[6:0]** finally settled). This is due to the gate delay and wiring since now we are simulating a post-layout circuit.
6. Maximize the waveform window. Hit **Print_scrn** to save the screen/result as **shifter_waveform.png** in the desktop.