



School of Engineering

Diploma in Electronics Computer & Communications Engineering
(EGDF01)

EXPERIMENT NO	:	Lab 07 (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 Lab7-1. Select **counter.v** and **right-click→Edit**.

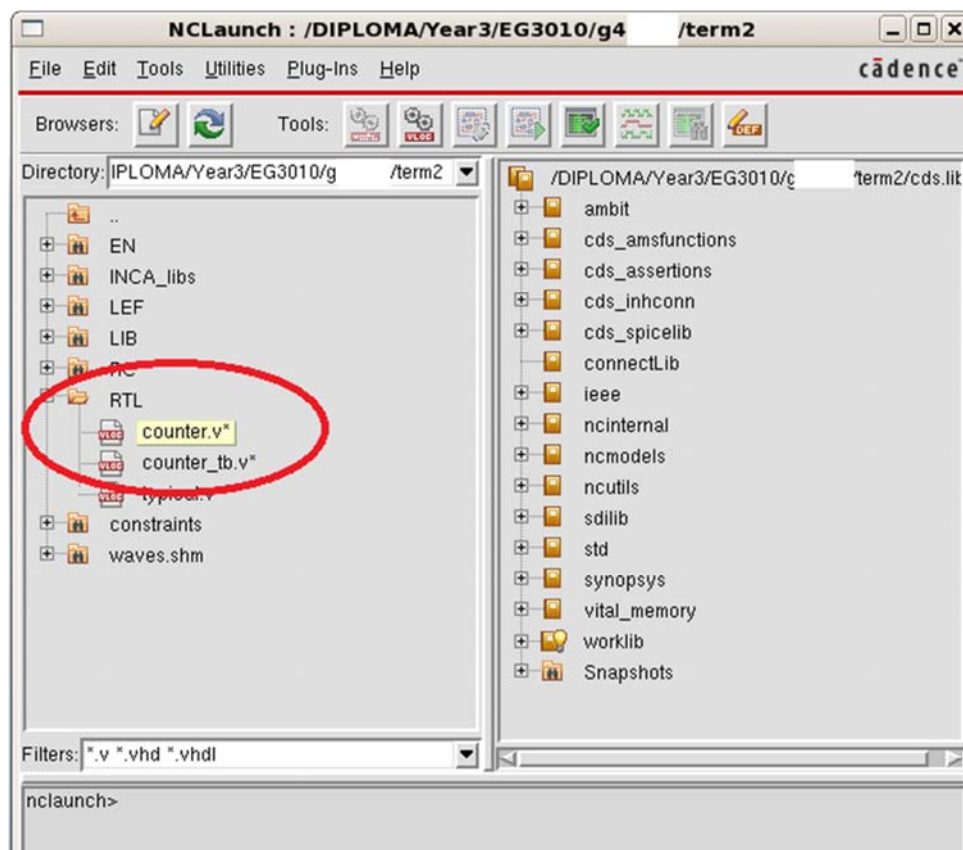
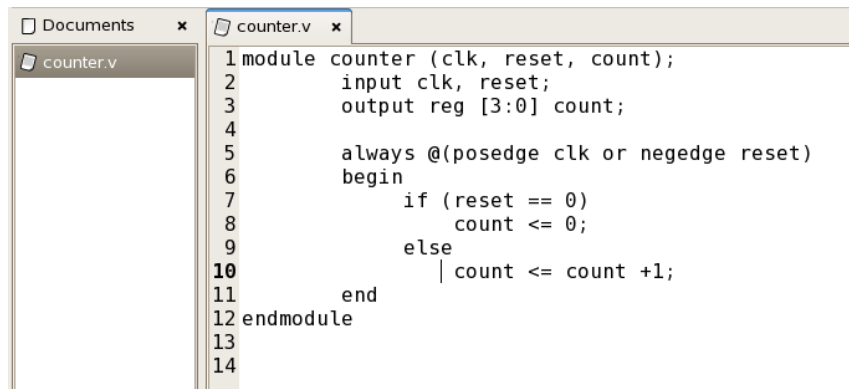


Figure Lab7-1 NCLaunch Window

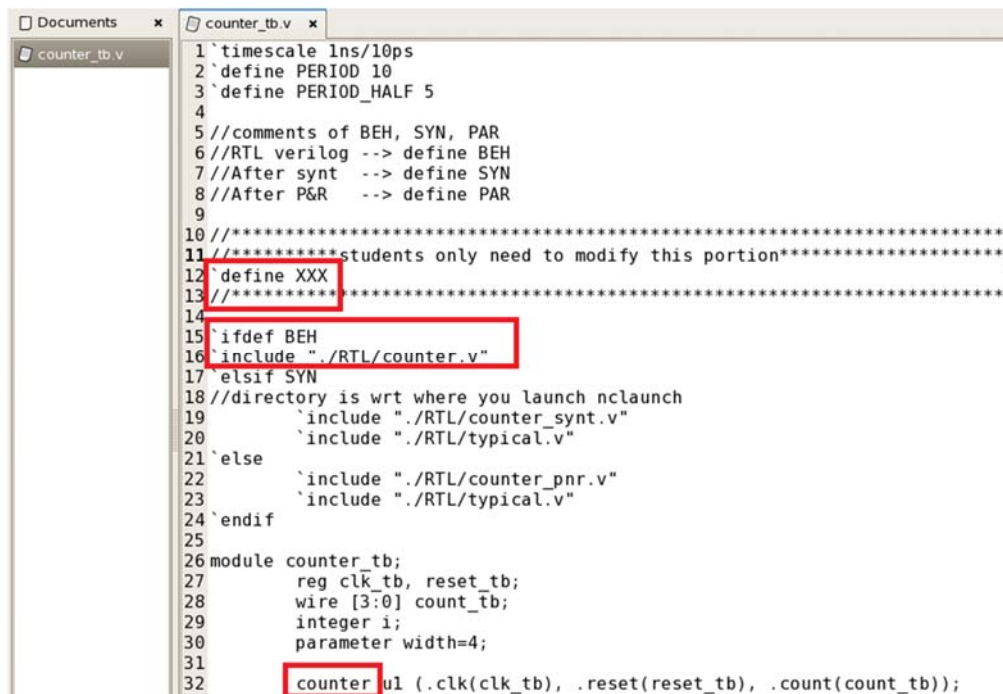
3. Figure Lab7-2 shows the verilog code of the counter. It has 4 bits output named as **count**. Read the code to gain some understanding about the counter's behavior. As you can see, **count** will only change either at positive-edge of **clock** or negative-edge of **reset**. Close the file.



```
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 Lab7-2 Verilog Code for 4-bit Counter

4. Select **counter_tb.v** and **right-click→Edit**. Figure Lab7-3 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 //*****
11 //*****students only need to modify this portion*****|
12 `define XXX
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
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
32     counter ul (.clk(clk_tb), .reset(reset_tb), .count(count_tb));
33
```

Figure Lab7-3 Verilog Code for Test bench

5. Select **counter.v** and **counter_tb.v** (as shown in Figure Lab7-4). **Right-click**→**NCVlog**. Then click **OK**. This will compile the verilog files to C code for faster simulation.

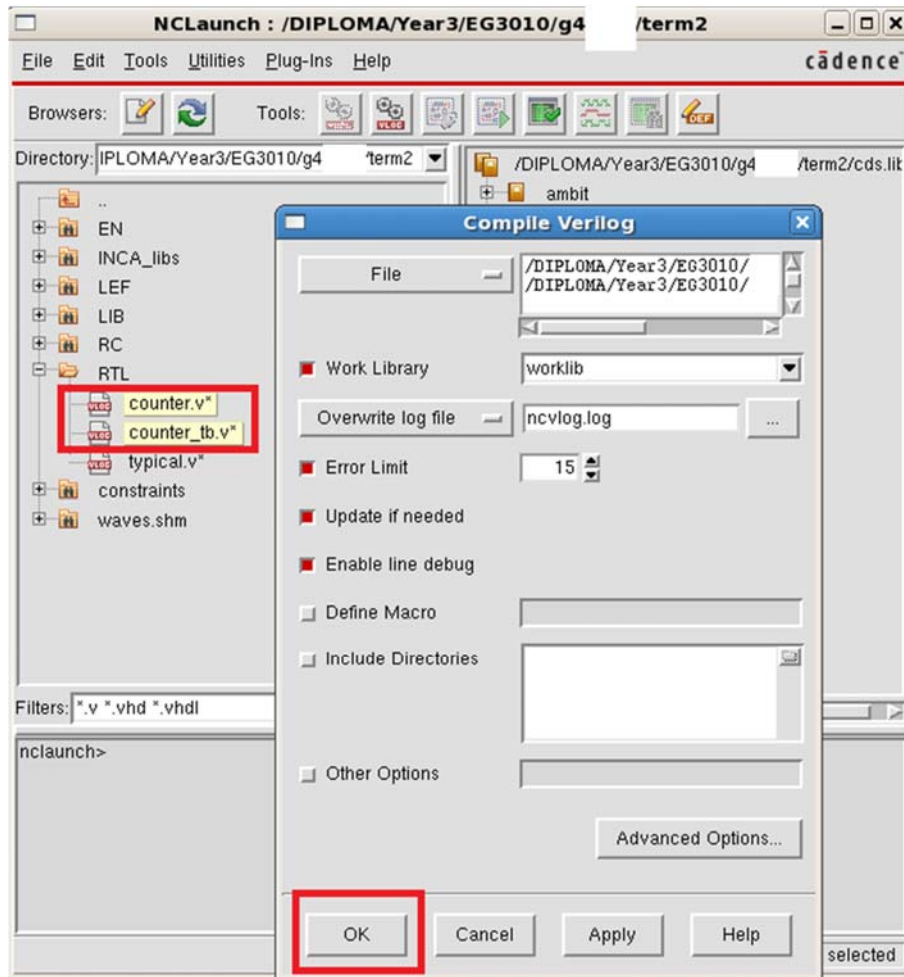


Figure Lab7-4 NCLaunch Window

6. Select **worklib/counter_tb** (as shown in Figure Lab7-5). **Right-click**→**NCElab**. Then click **OK**. This will link up all the related compiled modules for simulation.

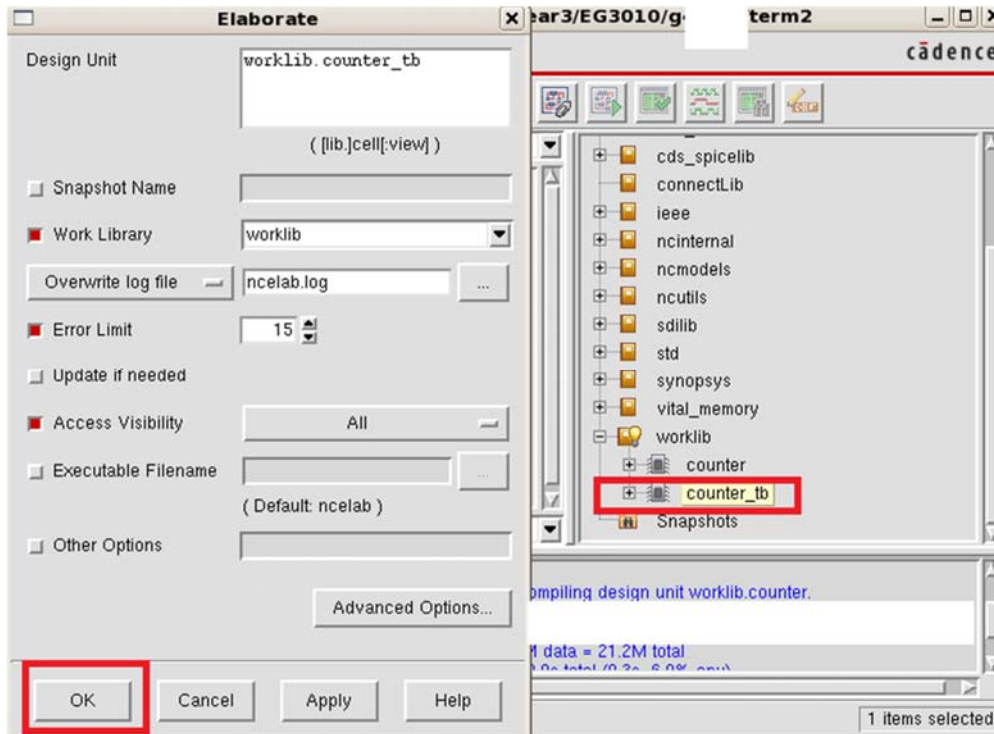


Figure Lab7-5 NCElab Window

7. Select **Snapshots/worklib.counter_tb:module** (as shown in Figure Lab7-6). **Right-click**→**NCSim**. Then click **OK**. This will allow you to setup the environment for simulation.

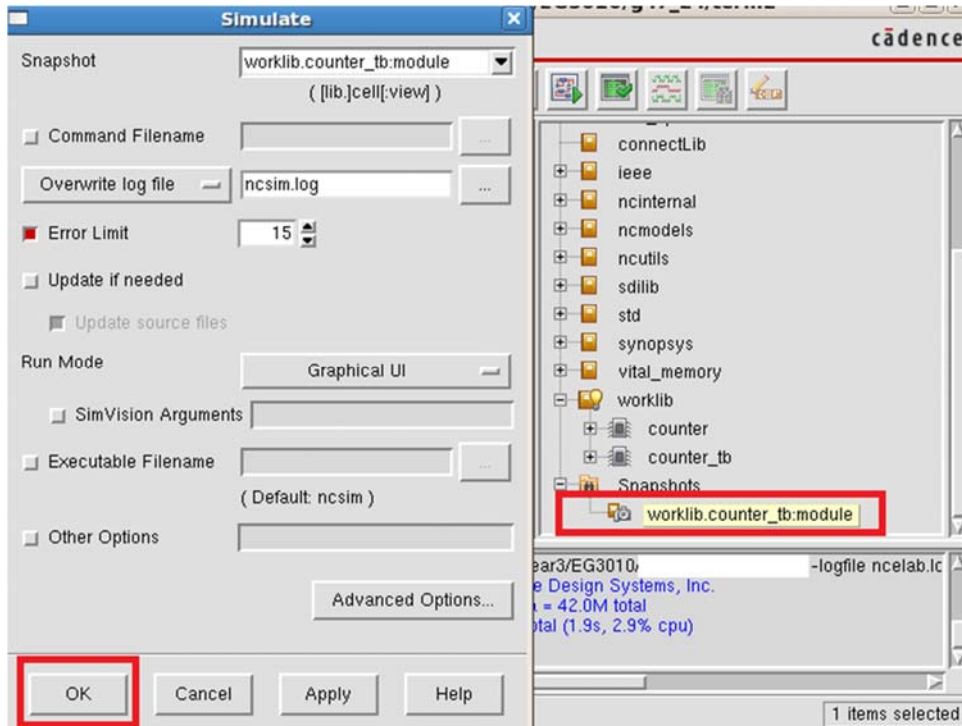


Figure Lab7-6 NCSim Window

8. Select **counter_tb** (as shown in Figure Lab7-7). **Right-click**→**Send To New**→**Waveform Window**. The related signals of counter_tb will be probed for display.

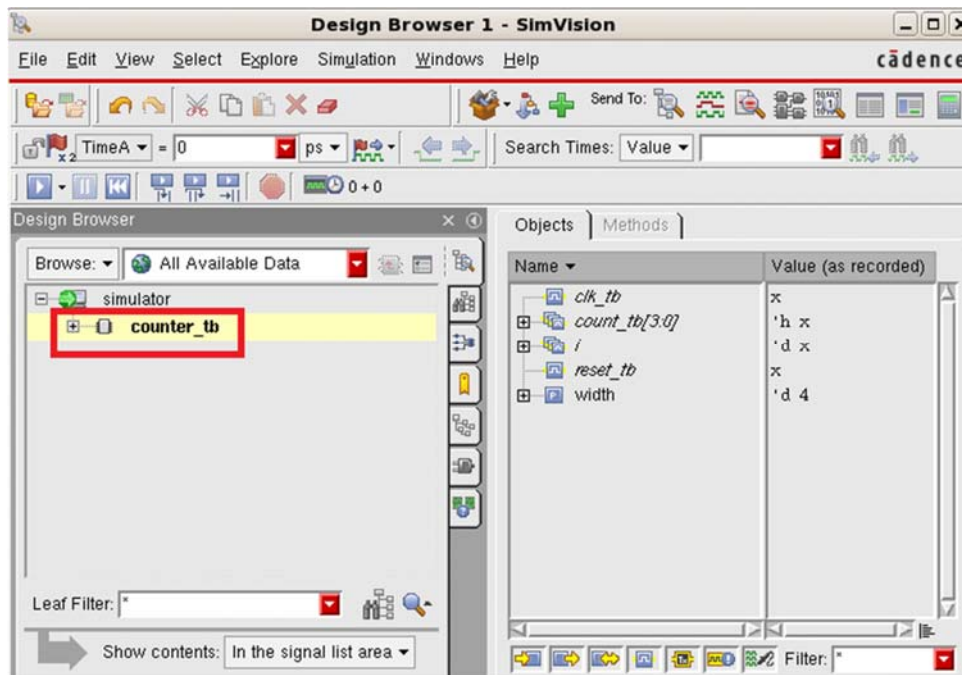


Figure Lab7-7 SimVision/Design Browser Window

9. Select **count_tb[3:0]** (the output of counter as shown in Figure Lab7-8). **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.

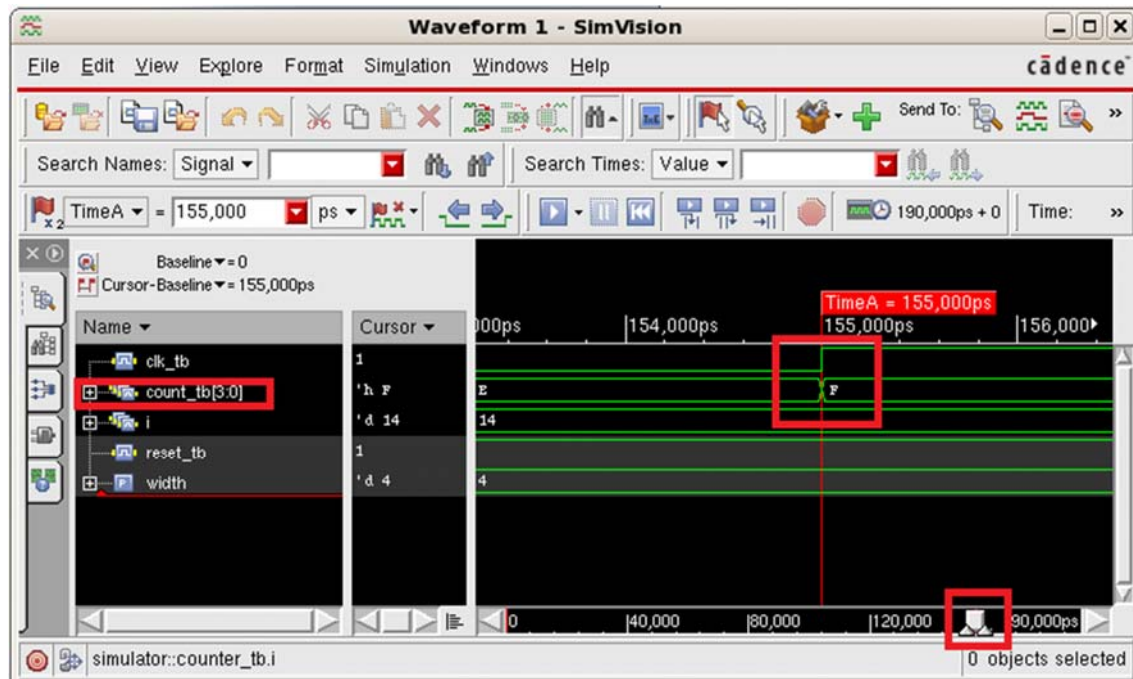


Figure Lab7-8 SimVision/Waveform Window

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 Lab7-9.

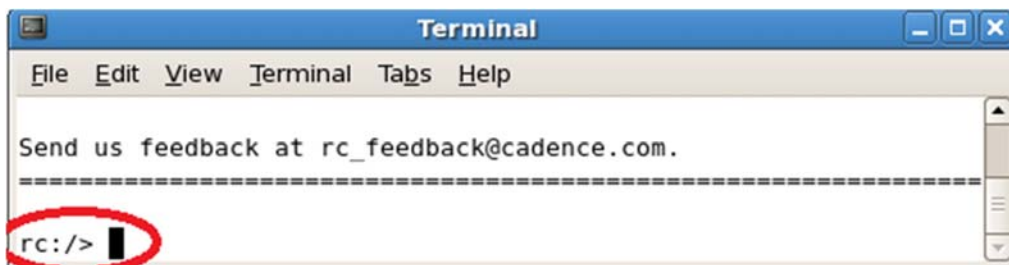


Figure Lab7-9 RC-shell Prompt

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.

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

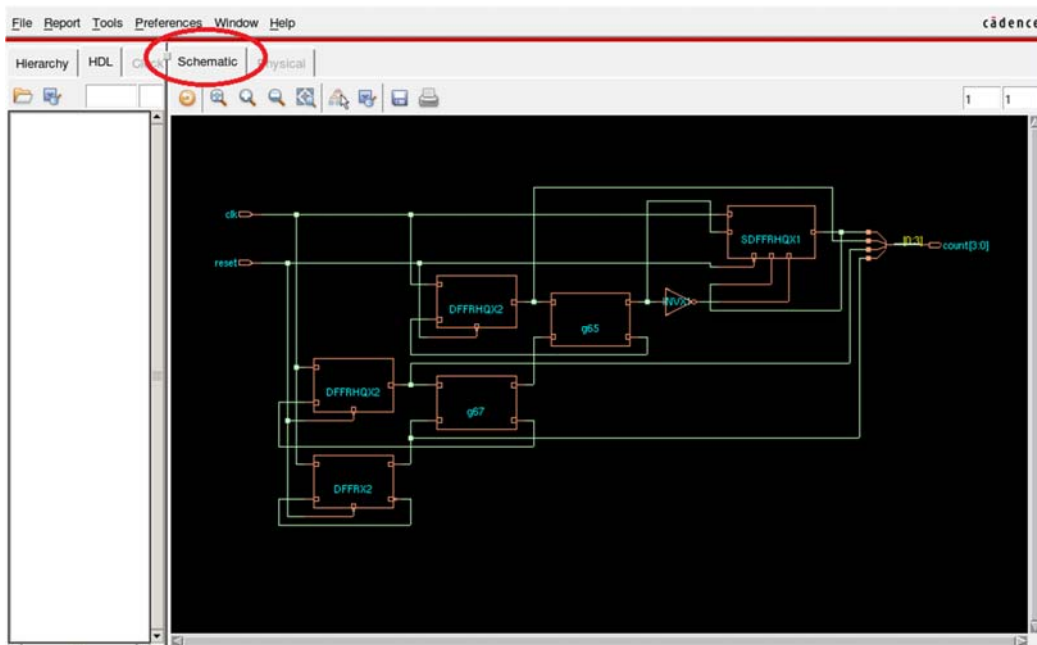
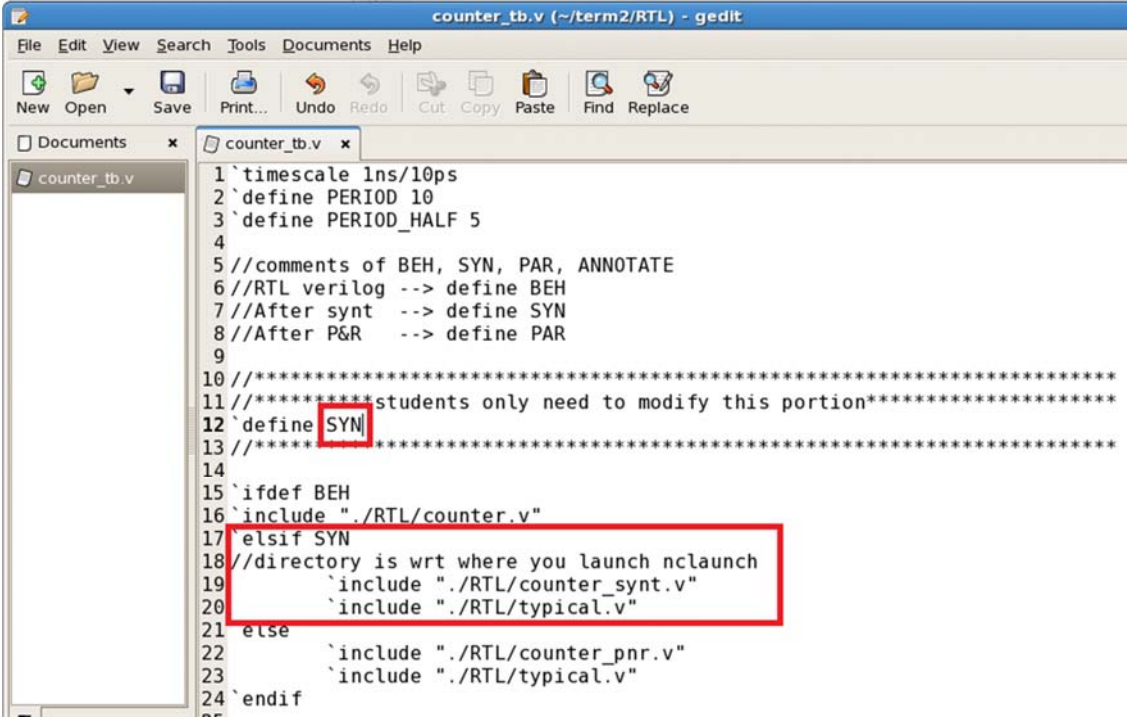


Figure Lab7-10 Schematic

5. Type:
gui_hide to close the window.
exit to exit RTL Compiler.
6. 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.
7. 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.

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 Lab7-11. 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 //*****
11 //*****students only need to modify this portion*****
12 `define SYN
13 //*****
14
15 `ifdef 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 Lab7-11 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 Lab7-12).

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

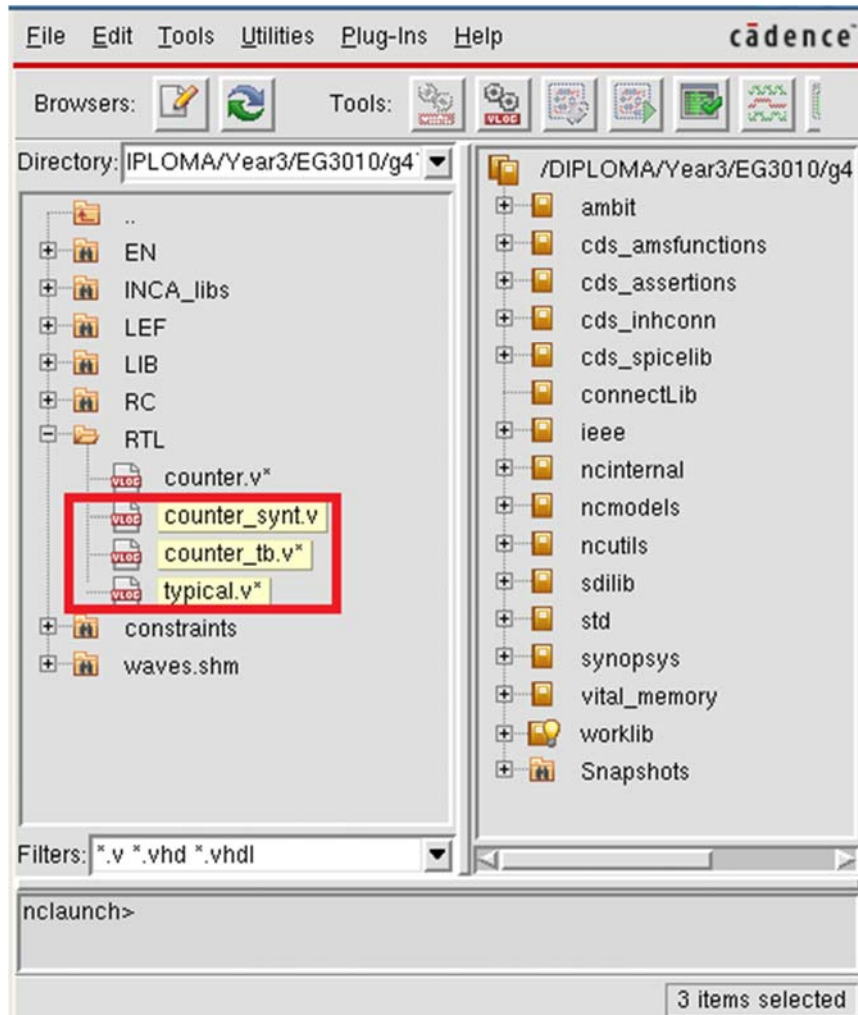


Figure Lab7-12 NCLaunch Window

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 Lab7-13.
 Click on waveform to bring **TimerA** to the desired location.
Right-click→Create a marker to put additional marker.

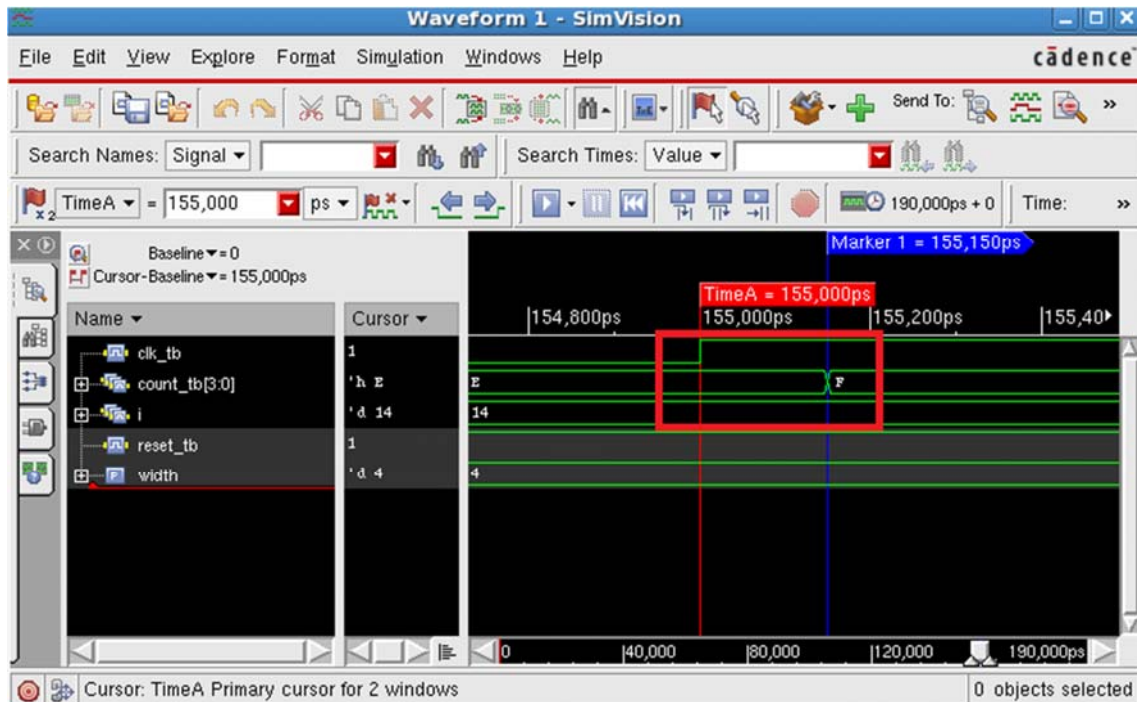


Figure Lab7-13 SimVision/Waveform Window

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.