Nanyang Polytechnic

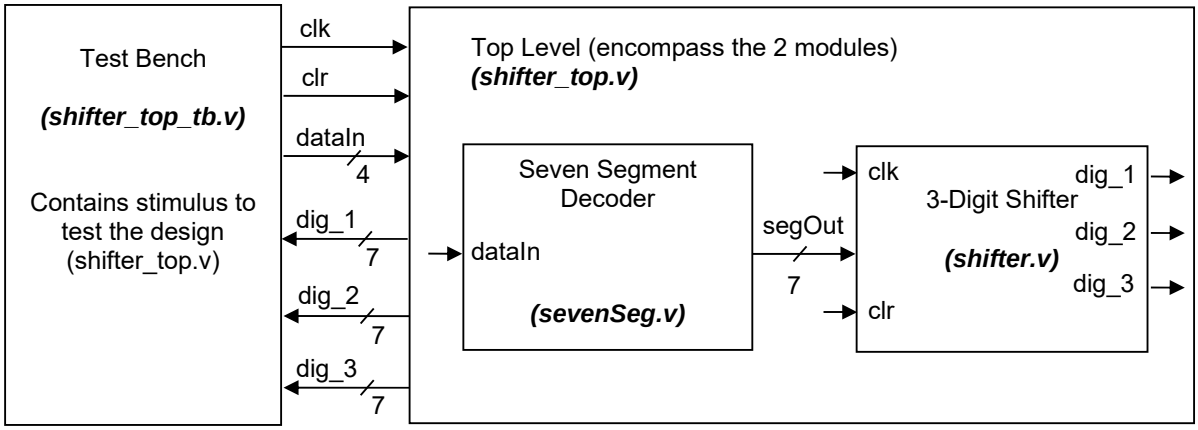
School of Engineering

Diploma in Electronics Computer & Communications Engineering (EGDF01)

EXPERIMENT NO	:	Lab 09 (Duration : 2 hours)
EXPERIMENT TITLE	:	RTL-to-Gates Level Synthesis of Shift Register using Cadence RTL Compiler
OBJECTIVE	:	1. Code a Shift Register in Verilog and simulate using Cadence Incisive Simulator 2. Synthesize the RTL codes to gates using Cadence RTL Compiler 3. Perform Post-synthesis simulation of design

---

Exercise 1 : To built a Shift Register from RTL codes using hierarchical approach.



1. Open a new Terminal
   Type : ***cd***
   cd term2
   source cshrc
   cd RTL

2. Verify the codes
   (a) ***gedit sevenSeg.v &***

```
module sevenSeg (dataIn, segOut);
input [3:0] dataIn;
output reg [6:0] segOut;

always @(dataIn)
case (dataIn)
        4'b0000: segOut = 7'b0000001;
        4'b0001: segOut = 7'b1001111;
        4'b0010: segOut = 7'b0010010;
        4'b0011: segOut = 7'b0000110;
        4'b0100: segOut = 7'b1001100;
        4'b0101: segOut = 7'b0100100;
        4'b0110: segOut = 7'b0100000;
        4'b0111: segOut = 7'b0001111;
        4'b1000: segOut = 7'b0000000;
        4'b1001: segOut = 7'b0001100;
        4'b1010: segOut = 7'b0001000;
        4'b1011: segOut = 7'b1100000;
        4'b1100: segOut = 7'b0110001;
        4'b1101: segOut = 7'b1000010;
        4'b1110: segOut = 7'b0110000;
        4'b1111: segOut = 7'b0111000;
endcase
endmodule
```

Figure 1 RTL codes of a 7-segment decoder

(b) ***gedit shifter.v***

```
module shifter (clr, clk, digitIn, dig_1, dig_2, dig_3);

input clr, clk;
input [6:0] digitIn;
output reg [6:0] dig_1, dig_2, dig_3;

always @(posedge clr or posedge clk)
   if (clr)
      begin
         dig_1 <= 7'b0000001;
         dig_2 <= 7'b0000001;
         dig_3 <= 7'b0000001;
      end
   else
   begin
         dig_3 <= dig_2;
         dig_2 <= dig_1;
         dig_1 <= digitIn;
   end
endmodule
```

Figure 2 RTL codes of a shifter

(c) ***gedit shifter_top.v***

```
module shifter_top (clr, clk, dataIn, dig_1, dig_2, dig_3);

input clr, clk;
input [3:0] dataIn;
output [6:0] dig_1, dig_2, dig_3;
wire [6:0] segOut_tb;

sevenSeg u1 (.dataIn(dataIn), .segOut(segOut));
shifter u2 (.clr(clr), .clk(clk), .digitIn(segOut), .dig_1(dig_1), .dig_2(dig_2), .dig_3(dig_3));

endmodule
```

Figure 3 Top-level of Shifter – Integrates the shifter and seven-seg modules

(d) ***gedit*** *shifter_top_tb.v*
Read and understand the code.
Close all the files.

3. We are going to repeat the steps of Lab 7 Exercise 1 to 3 and Lab 8 Exercise 1 to 2. Refer to those Lab if you're not sure about the details of the commands.
cd ..
nclaunch&

4. As shown in Figure 4, select **shifter_top_tb.v** and **right-click→Edit**. Again, modify to - 'ifdef **BEH** - for this exercise. When the variable **BEH** is defined, the related files contain the relevant modules will be read/included by the simulator.
Save and **close** the file.

5. Select **shifter_top_tb.v, shifter_top.v, shifter.v** and **sevenSeg.v** as shown in Figure 4. **Right-click→NCVlog**. Then click **OK**.
This will compile the verilog files to C code for faster simulation.

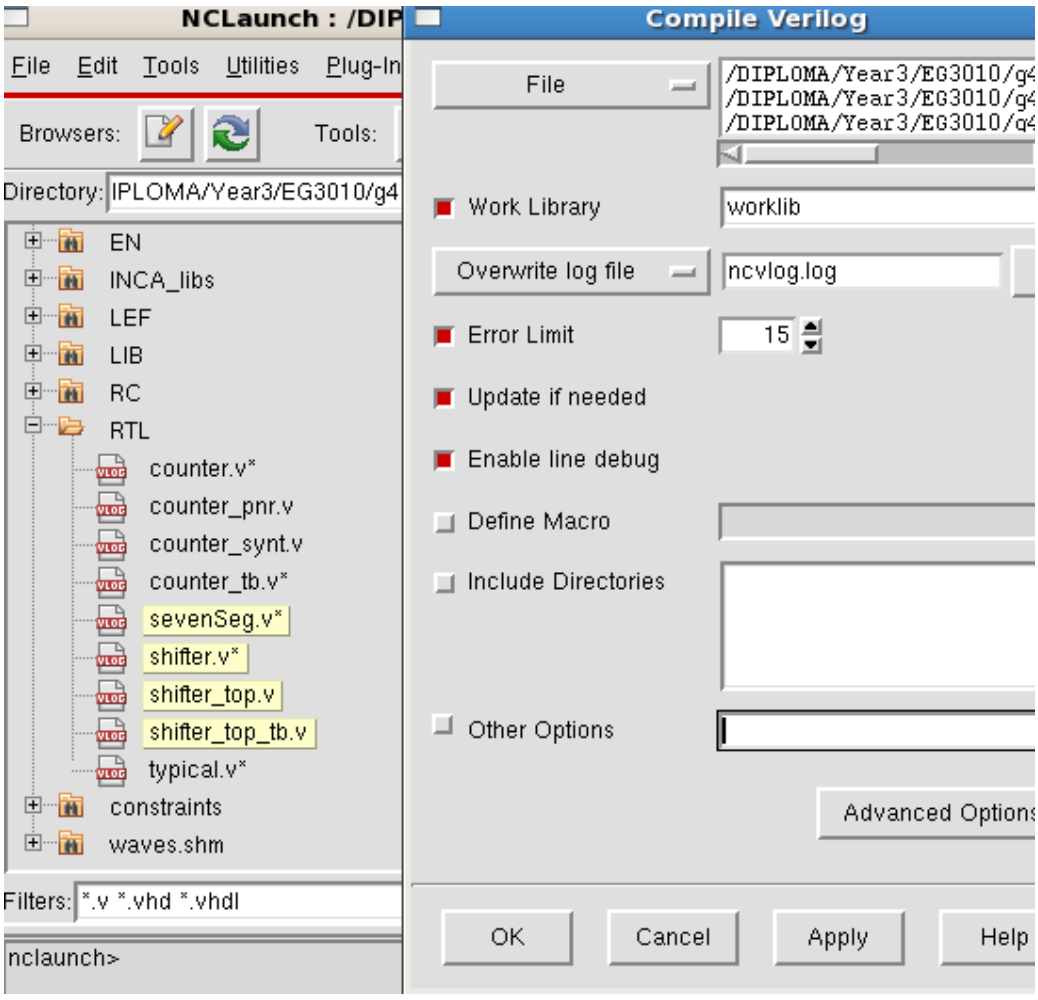


Figure 4 NCLaunch Window

6. Select **worklib/shifter_top_tb.**
Right-click→NCElab. Then click **OK**.
This will link up all the related compiled modules for simulation.

7. Select **Snapshots/worklib.shifter_top_tb:module.**
Right-click→NCSim. Then click **OK**.
This will allow you to setup the environment for simulation.

8. Select **shifter_top_tb**. **Right-click→Send To New→Waveform Window**. The related signals of counter_tb will be probed for display.

9. **Simulation→Run**. Then **View→Zoom→Full X**. Use the **Sliding Bar** and **View→Zoom→In X (or Alt-i)** to zoom in to region as shown in Figure 5. Click on the waveform screen to bring TimerA to the clock edge as shown. There is no delay between the positive-edge of **clk** and output of shifter (dig_1, dig_2 and dig_3) since the shifter is modeled by verilog behavioral code.

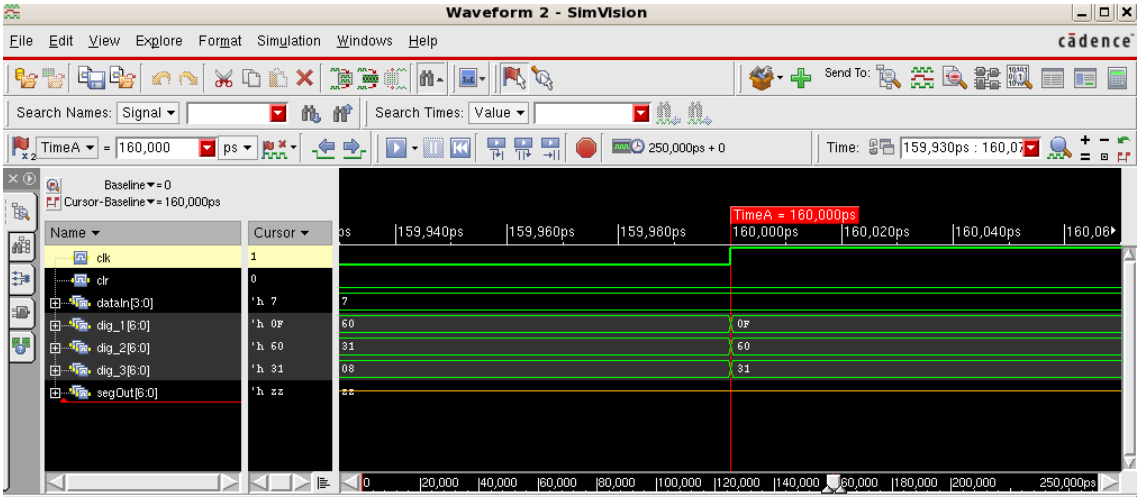


Figure 5 SimVision/Waveform Window

10. Exit all the software.

Exercise 2 : To synthesize the Shifter, of Exercise 1, using Cadence 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.

3. Type:
   source shifter_top.tcl
   shifter_top.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. Double-click on Hierarchy to see the sub-circuit.
   Report→Netlist→Statistics to see the number and type of gates used to build the shifter.

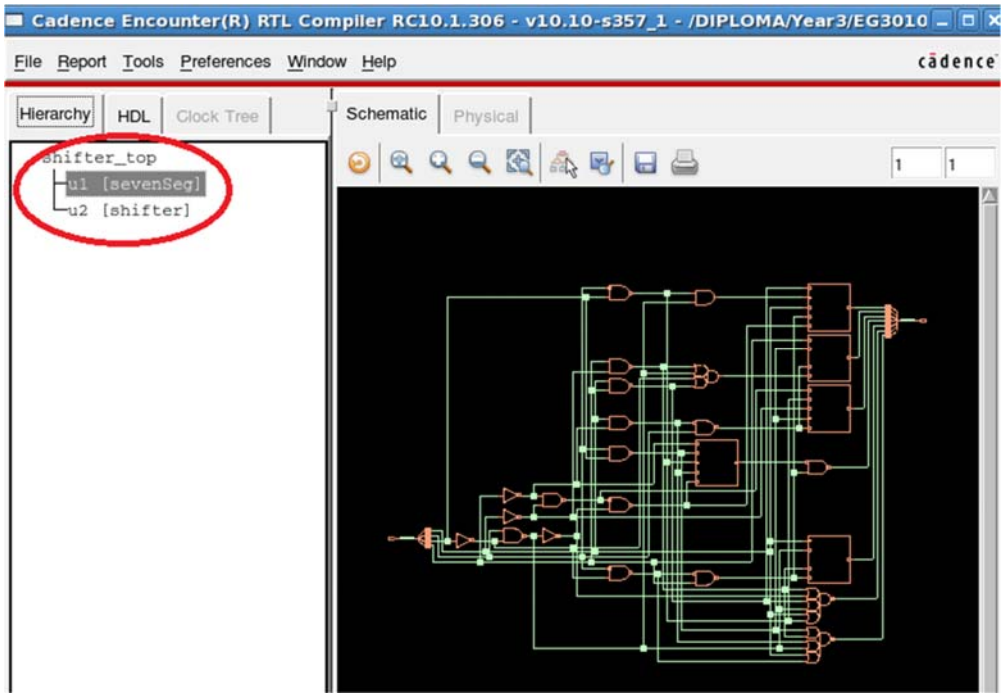


Figure Lab9-6 Schematic

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

6. The synthesized netlist file is in **…/term2/RTL**.
   Go to that directory by **cd ../RTL** and do a ***more shifter_top_synt.v*** to view the netlist.

Exercise 3 : Post-synthesize simulation

1. Type:
   cd .. to go up to …/term2.
   nclaunch&

2. **Right-click→Edit** on RTL/shifter_top_tb.v
   Change the variable to **SYN**.
   Close the file.

3. Select **shifter_top_synt.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 (around 160ns).
   Right-click→Create a marker to put additional marker.

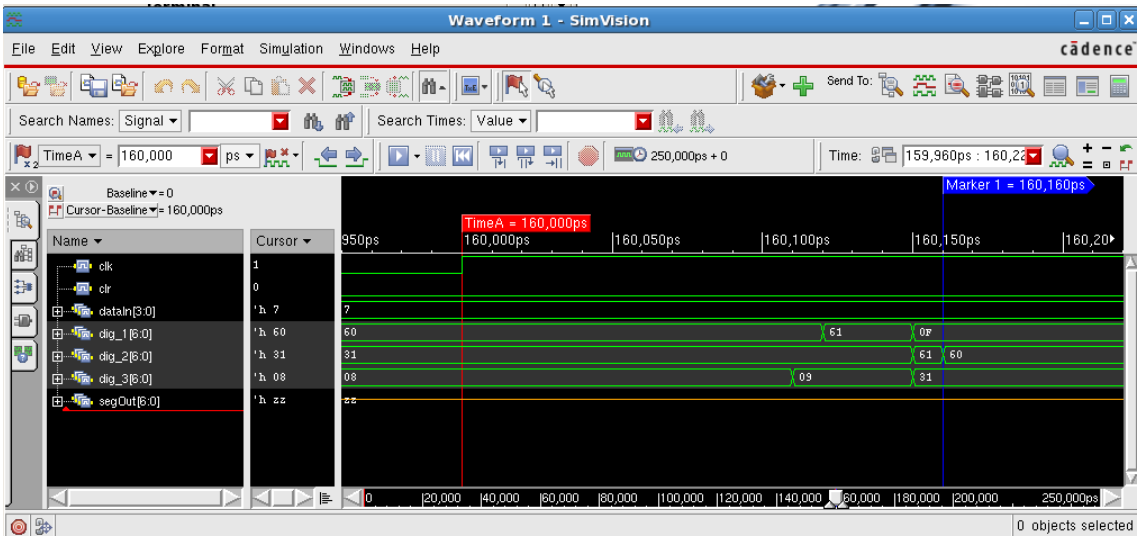


Figure 7 Result

5. As you can see, there is delay of **110ps** between postive-edge of **clk** and shifter's output. The shifter's output – **dig_1, dig_2** and **dig_3** are valid 110ps after the positive-edge of **clk**. This is due to the gate delay since now we are simulating a real circuit.