

# National Cheng Kung University

## Department of Electrical Engineering

### *Introduction to VLSI CAD (Spring 2024)*

#### Lab Session 4

#### Register Files, Manhattan Distance and LFSR

Name	Student ID	
游宗謀	E94106151	
Practical Sections	Points	Marks
Prob A	30	
Prob B	30	
Prob C	20	
Report	15	
File hierarchy, naming...etc.	5	
Notes:		

**Due Date: 15:00, March 27, 2024 @ moodle**

## Deliverables

- 1) All Verilog codes including testbenches for each problem should be uploaded.  
NOTE: Please **DO NOT** include source code in the paper report!
- 2) All homework requirements should be uploaded in this file hierarchy or you will not get the full credit.  
NOTE: Please **DO NOT** upload waveforms!
- 3) **Important! TA will use the command in Appendix A to check your design under SoC Lab environment, if your code can not be recompiled by TA successfully using the commands, you will not get the full credit.**
- 4) If you upload a dead body which we can't even compile you will get **NO** credit!
- 5) All Verilog file should get at least **90%** superLint Coverage.
- 6) **File hierarchy should not be changed; it may cause your code can not be recompiled by TA successfully using the autograding commands**

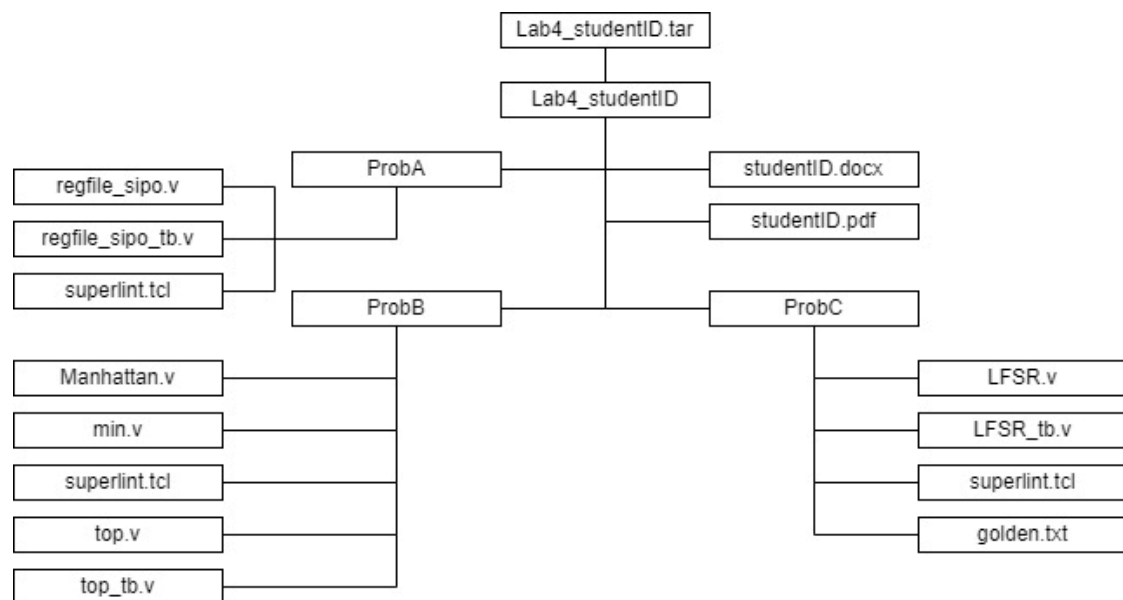
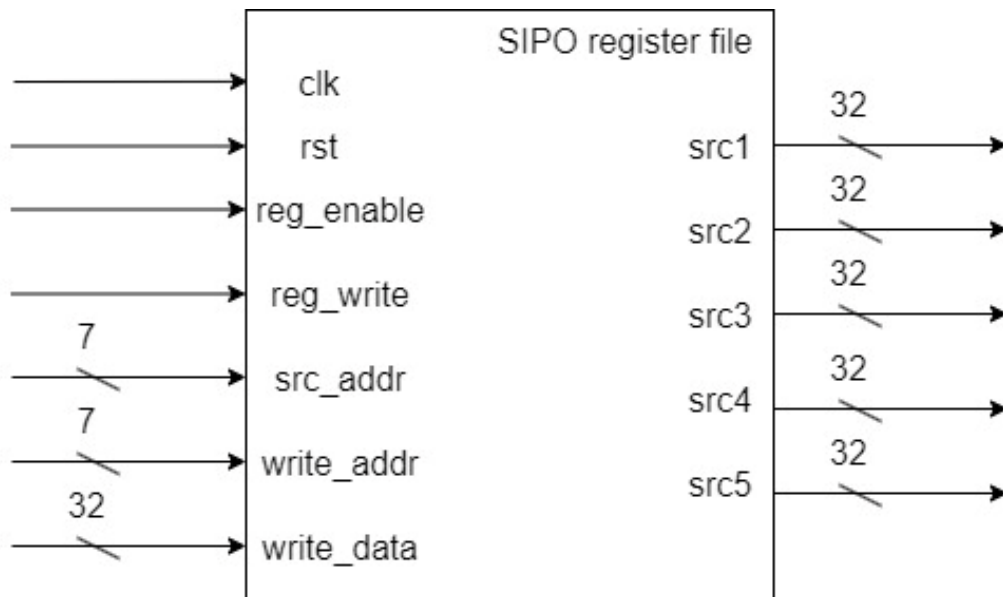


Fig.1 File hierarchy for Homework submission

Prob A: SIPO Register File



- Based on the SIPO register file structure in LabA, please design a **128 x 32** SIPO register file with **5** output ports.
- Port list

Signal	Type	Bits	Description
clk	input	1	clock
rst	input	1	reset
reg_enable	input	1	register file enable
reg_write	input	1	0 → read 1 → write
src_addr	input	7	source address
write_addr	input	7	write address
write_data	input	32	write data
src1	output	32	read data source1
src2	output	32	read data source2
src3	output	32	read data source3
src4	output	32	read data source4
src5	output	32	read data source5

3. Show the simulation result on the terminal.

```

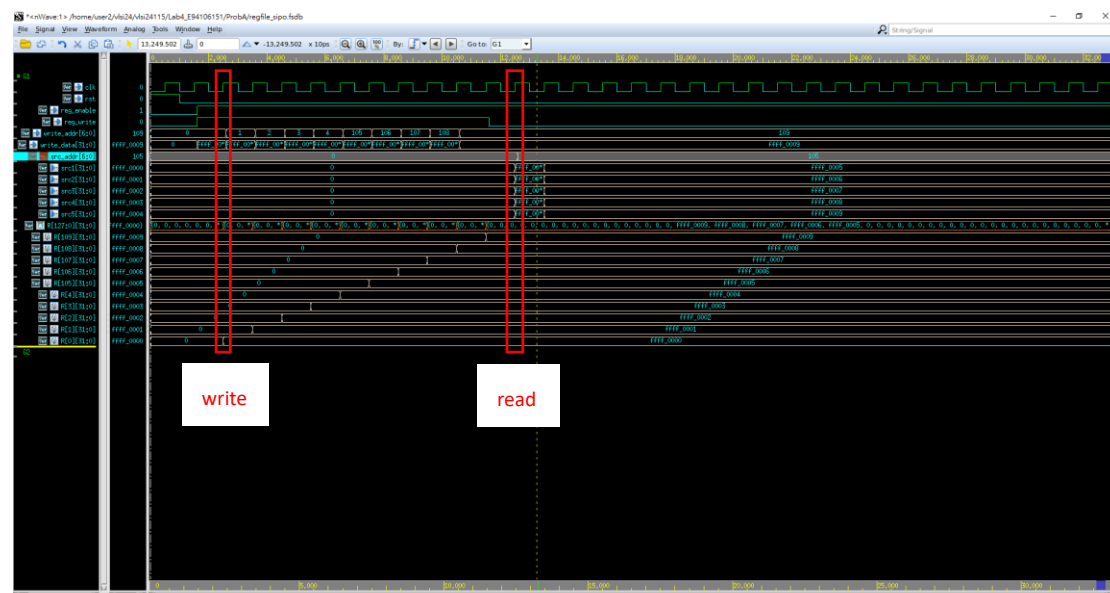
140.116.156.10 - PuTTY
[1] 841 = 00000000
[1] 871 = 00000000
[1] 881 = 00000000
[1] 891 = 00000000
[1] 901 = 00000000
[1] 911 = 00000000
[1] 921 = 00000000
[1] 931 = 00000000
[1] 941 = 00000000
[1] 951 = 00000000
[1] 961 = 00000000
[1] 971 = 00000000
[1] 981 = 00000000
[1] 991 = 00000000
[1] 1001 = 00000000
[1] 1011 = 00000000
[1] 1021 = 00000000
[1] 1031 = 00000000
[1] 1041 = 00000000
[1] 1051 = fffff004
[1] 1061 = fffff004
[1] 1071 = fffff007
[1] 1081 = fffff009
[1] 1091 = fffff009
[1] 1101 = 00000000
[1] 1111 = 00000000
[1] 1121 = 00000000
[1] 1131 = 00000000
[1] 1141 = 00000000
[1] 1151 = 00000000
[1] 1161 = 00000000
[1] 1171 = 00000000
[1] 1181 = 00000000
[1] 1191 = 00000000
[1] 1201 = 00000000
[1] 1211 = 00000000
[1] 1221 = 00000000
[1] 1231 = 00000000
[1] 1241 = 00000000
[1] 1251 = 00000000
[1] 1261 = 00000000
[1] 1271 = 00000000

*****
**                               / \
** Computations !! **          / O.G \
**                               |
**                               |
** Simulation PASS!! **        /---\
**                               | w |
**                               |
*****                               \___/

Finish called from file "regfile_nipo_th.v", line 160.
Finish at simulation time 32400
***** VCS 3 Simulation Report *****
Time: 324000 ps
CPU Time: 0.490 seconds/ Data structure size: 0.0MB
Thu Mar 21 16:12:05 2024
CPU time: 186 seconds to compile + 760 seconds to simulate + 198 seconds to link + 742 seconds in simulation
c:\csdcs\prowse2\vs15\vs1516115\lab5\2391016115\prowse2\

```

4. Show waveforms to explain that your register work correctly when **read** and **write**.

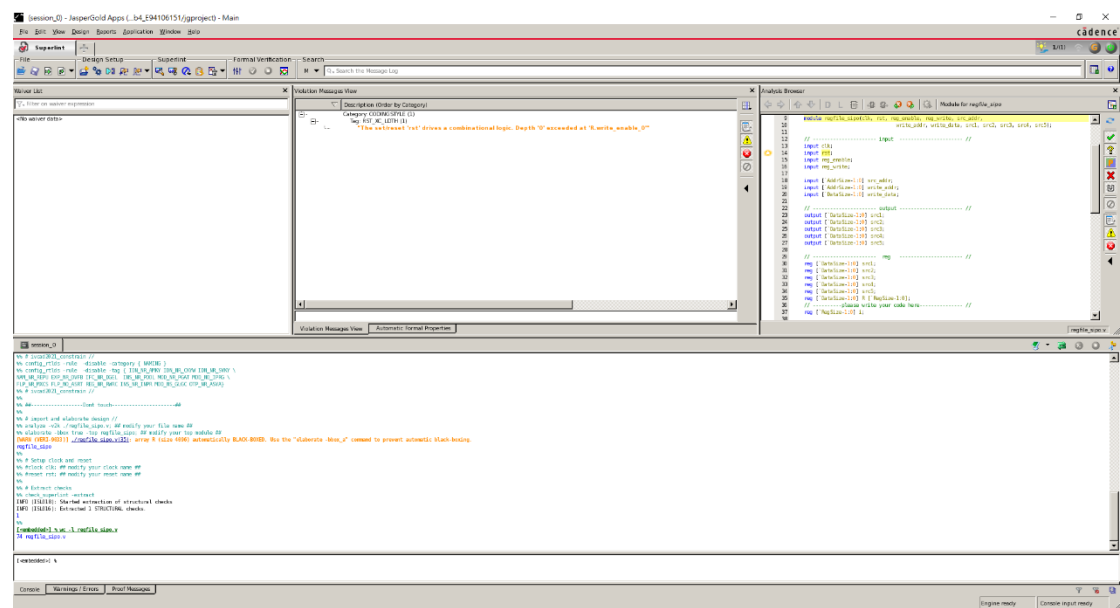


**Read:** 從標示出的位置可以看到要讀取的位址(src\_addr)為 0 到 0+4=4，根據 register(R)可以看出 R[0] = 32'hffff0000, R[1] = 32'hffff0001, R[2] = 32'hffff0002, R[3] = 32'hffff0003, R[4] = 32'hffff0004，對應到輸出 src1 = R[0] = 32'hffff0000, src2 = R[1] = 32'hffff0001, src3 = R[2] = 32'hffff0002, src4 = R[3] = 32'hffff0003, src5 = R[4] = 32'hffff0004。

**Write:** 從標示出的位置可以看到要寫入的位址(write\_addr)為 0、要寫入的值(write\_data)為 32'hffff0000，根據 register(R)可以看出輸入後 R[0] = 32'hffff0000。

由此可知 register 有正常運作。

## 5. Show SuperLint coverage

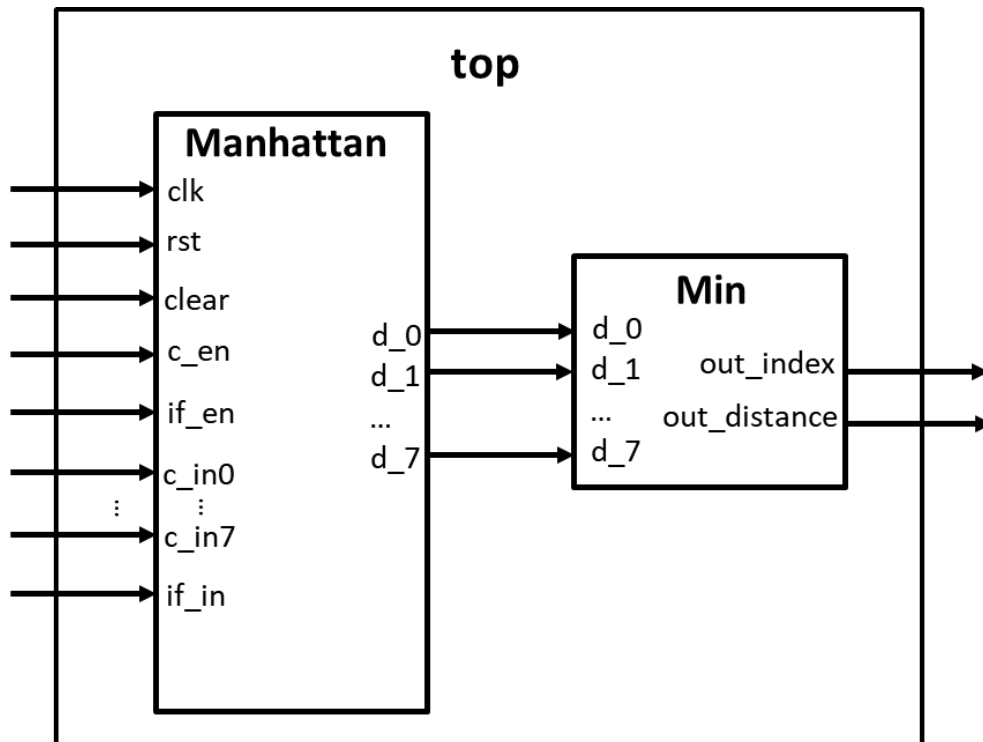


$$\text{Coverage} = (1 - (1/74)) * 100\% = 98.65\%$$

---

*Prob B: Finding Smallest Distance*

---



1. Please design a circuit that will find the smallest distance between the input feature and input colors, based on the structure given in the LAB4 slide.

2. Port list

Manhattan:

Signal	Type	Bits	Description
<code>clk</code>	input	1	Clock pin.
<code>rst</code>	input	1	Reset pin.
<code>clear</code>	input	1	Set all registers to 0.
<code>c_en</code>	input	1	Write compared colors enable. When <code>c_en</code> is high, then <code>c_in0~7</code> is available.
<code>if_en</code>	input	1	Write input pixel enable. When <code>if_en</code> is high, then <code>if_in</code> is available.
<code>c_in0~7</code>	input	24 each	Input color data.
<code>if_in</code>	input	24	Input input feature data.
<code>d_0~7</code>	output	10 each	Output distance data.

Min:

Signal	Type	Bits	Description
d_0~7	input	10 each	Input data.
out_index	output	3	Output index.
out_distance	output	10	Output minimum distance.

3. Show the simulation result on the terminal.

```

MS161556.6.PUTTY
...
Warning: License for product VCS-SAKE-SOFTWARE will expire within 11 days, on: 31-mar-2024.

If you would like to temporarily disable this message, set
the VCS_SAKE_DISABLE_WARNING environment variable to the number of days
before expiration that you want this message to start (the minimum is 0).

*Verdi* Loading libsimosa_vcs202303.a.o
FSDM compiler: gcc vcs, Release Verco_D-2023.03-SP2, Linux x86_64/64bit, 03/28/2023
(C) 1996 - 2023 by Synopsys, Inc.
*Verdi* FSDM Runtime: The FSDM file already exists. Overwriting the FSDM file may crash the programs that are using this file.
*Verdi* : Create FSDM file 'top.fsdm'
*Verdi* : Begin traversing the scope (top.tb-1, layer 0).
*Verdi* : Enable version and mode dumping.
*Verdi* : End of traversing.

-----
time      35 output is correct
time      45 output is correct
time      55 output is correct
time      65 output is correct
time      75 output is correct
time      85 output is correct

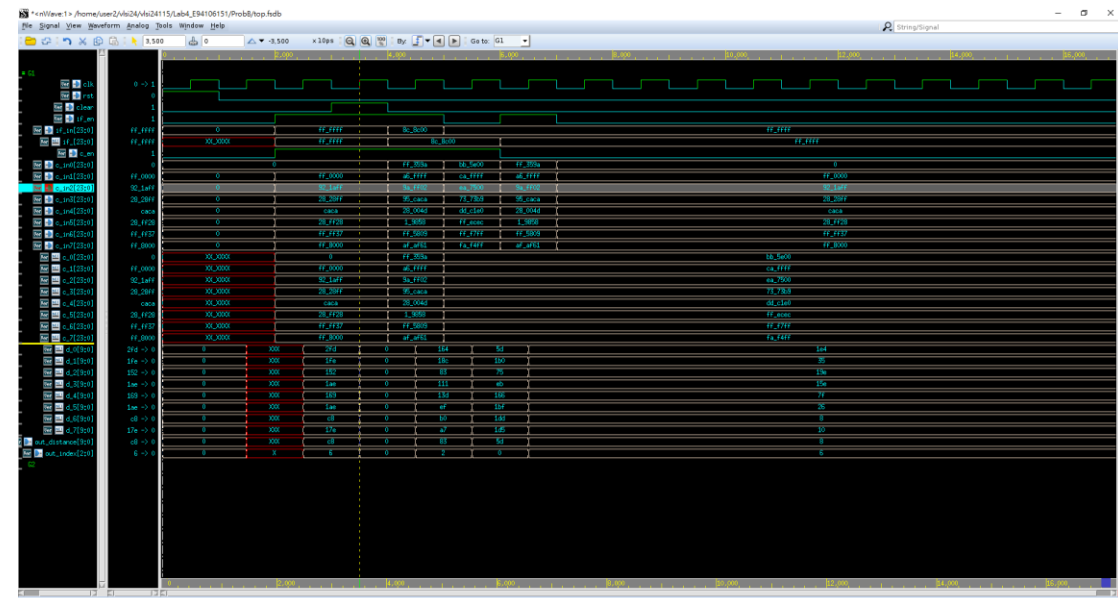
-----

*****
**      Congratulations !!      **      / 0.0 /
**      Simulation PASSED!!      **      / 0.0 /
*****

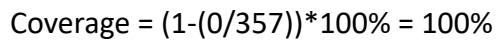
$finish called from file "top.tb.v", line 145.
$finish at simulation time 170000
VCS Simulation Report
Time: 170000 ps
CPU Time: 0.530 seconds; Data structure size: 0.00B
Thu Mar 21 21:04:05 2024
CPU time: 479 seconds to compile + 479 seconds to elab + 334 seconds to link + .560 seconds in simulation
vlsi06d6/home/user2/vlsi24/vlsi2415/Lab4_E9406151/RunB $

```

4. Show waveforms to explain that your design works correctly.



根據所標示的位置，Input feature data 為 if\_en = 24'hffffff(255, 255, 255)，Input color data 6 為 c\_in6 = 24'hffff37(255, 255, 55)，相差 distance data 為 d\_6 = 10'h0c8(200)是相比其他輸入差最少的，所以 output minimum distance 為 out\_distance = 10'h0c8，output index 為 out\_index = 3'h6。





---

### Prob C: LFSR

---

1. Please design an 8-bit-LFSR, with the given feedback function in the LAB4 slide.
2. Port list

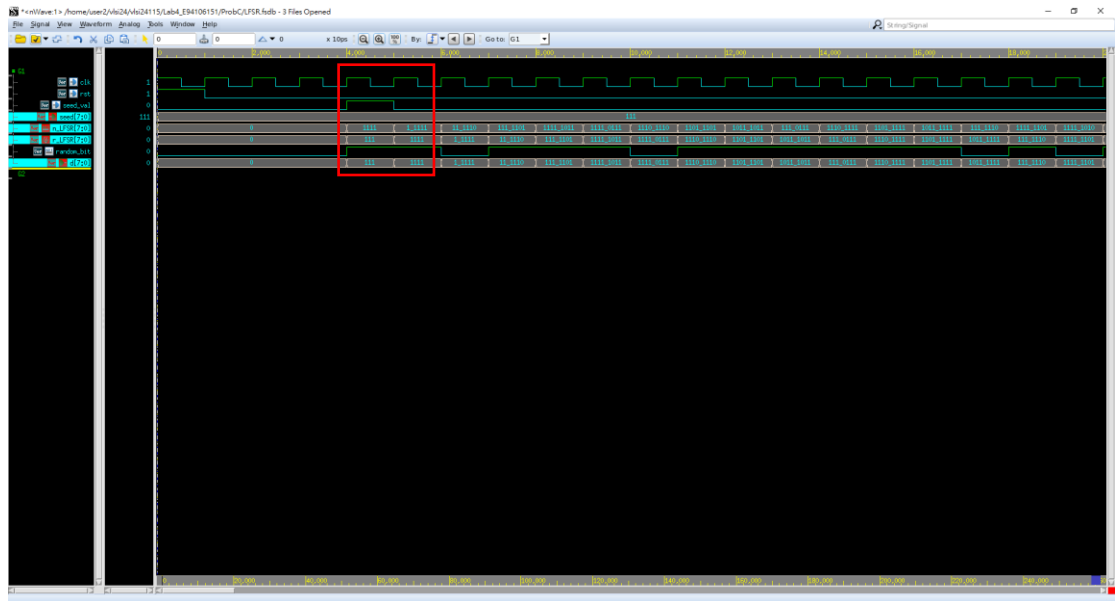
Signal	Type	Bits	Description
clk	input	1	Clock pin.
rst	input	1	Reset pin. Reset all of the flip flops to zeros.
seed_val	input	1	1: the flip flops take seed as the initial state. 0: the flip flops works as linear feedback shift register.
seed	input	8	Initial state value of LFSR.
d	output	8	Output value of LFSR

3. Feedback function

$$d[0] = ( d[7] \wedge d[5] ) \wedge ( d[4] \wedge d[2] )$$

4. Show waveforms to explain that your LFSR module works correctly.





從標示的位置可以看到在 `seed_val` 為 1 時，`seed = 8'b00000111` 有確實輸入進電路當中(`d = 8'b00000111`)，而在下一個 `clk` posedge 時，根據 Feedback function 對應  $d[0] = (d[7] \wedge d[5]) \wedge (d[4] \wedge d[2]) = (0 \wedge 0) \wedge (0 \wedge 1) = 0 \wedge 1 = 1$ ，可得 `d = 8'b00001111`。

5. Show the simulation result on the terminal.

```

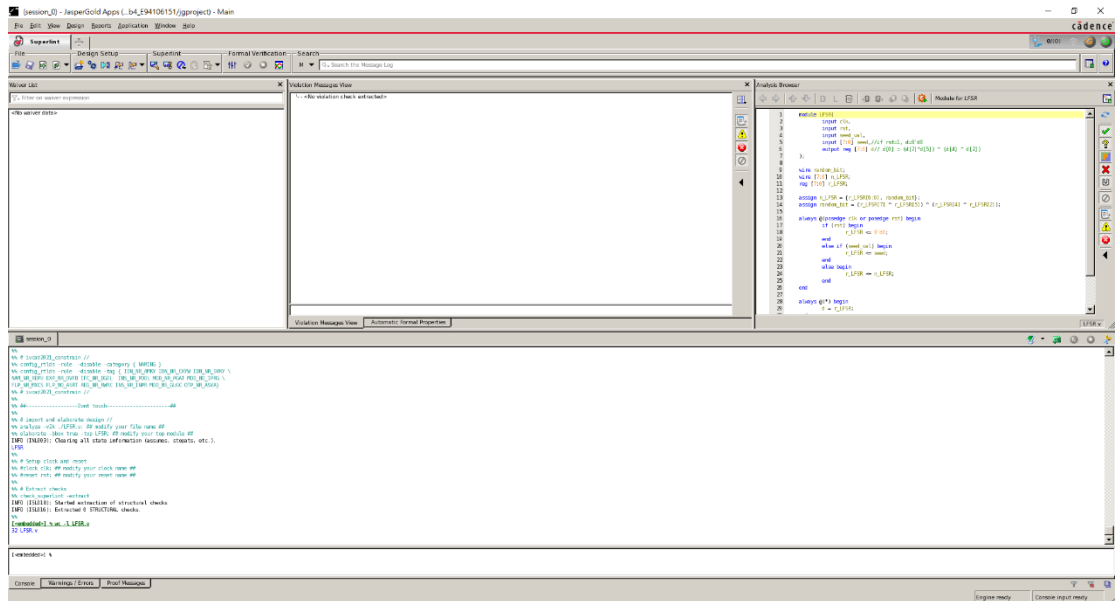
*AD11615610 - PUTTY
2170 output = 79, expect = 79
2180 output = 159, expect = 159
2190 output = 63, expect = 63
2200 output = 127, expect = 127
2210 output = 255, expect = 255
2220 output = 191, expect = 191
2230 output = 383, expect = 383
2240 output = 246, expect = 246
2250 output = 241, expect = 241
2260 output = 227, expect = 227
2270 output = 198, expect = 198
2280 output = 140, expect = 140
2290 output = 74, expect = 74
2300 output = 46, expect = 46
2310 output = 86, expect = 86
2320 output = 197, expect = 197
2330 output = 188, expect = 188
2340 output = 21, expect = 21
2350 output = 42, expect = 42
2360 output = 85, expect = 85
2370 output = 170, expect = 170
2380 output = 34, expect = 34
2390 output = 168, expect = 168
2400 output = 80, expect = 80
2410 output = 161, expect = 161
2420 output = 66, expect = 66
2430 output = 132, expect = 132
2440 output = 8, expect = 8
2450 output = 16, expect = 16
2460 output = 32, expect = 32
2470 output = 64, expect = 64
2480 output = 128, expect = 128
2490 output = 256, expect = 256
2500 output = 512, expect = 512
2510 output = 1024, expect = 1024
2520 output = 2048, expect = 2048
2530 output = 4096, expect = 4096
2540 output = 8192, expect = 8192
2550 output = 16384, expect = 16384
2560 output = 32768, expect = 32768

*****
**                               |__|
**  Congratulations !!         / D.O. |
**                               |__|
**  Simulation PASSED!!        /  -  -  \ |
**                               |__|
*****

$finish called from file "LFSR_tb.v", line 54.
$finish at simulation time: 256000
VCS Simulation Report
Time: 256000 ps
CPU Time: 0.660 seconds; Data structure size: 0.00Mb
Thu Mar 21 16:13:28 2024
CPU time: .583 seconds to compile + .696 seconds to elab + .446 seconds to link + .750 seconds in simulation
vlsi00001/home/user2/vlsi24/vlsi24119/Lab_8/vlsi041191/prb0c

```

6. Show SuperLint coverage



Coverage = (1-(0/32))\*100% = 100%

At last, please write the lesson you learned from Lab4

在這次的實驗課中我學到我們在計算機組織中上到的 **register file** 是如何透過 **verilog** 實作出來的，讓我更了解計算機的架構。此外還學到了 **Manhattan** 和 **Euclidean** 兩種不同計算相對距離的方法，其中 **Manhattan** 是這次第二題所用到的方法，我想如果要用 **verilog** 實作出影像辨識，這應該會是其中一小部分吧？最後是我第一次接觸到的 **LFSR**，我從來沒想過原來我們可以用這種方法用電路實作出接近隨機的數字產生器，沒想過原來 **D type Flip-Flop** 也有如此功用，讓我大開眼界。

*Appendix A : Commands we will use to check your homework*

Problem		Command
Prob A	Compile	% vcs -R regfile_sipo.v -full64
	Simulate	% vcs -R regfile_sipo_tb.v -debug_access+all -full64 +define+FSDB
Prob B	Compile	% vcs -R top.v -full64
	Simulate	% vcs -R top_tb.v -debug_access+all -full64 +define+FSDB
Prob C	Compile	% vcs -R LFSR.v -full64
	Simulate	% vcs -R LFSR_tb.v -debug_access+all -full64 +define+FSDB