

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	
鄭喆嚴	E14096724	
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**

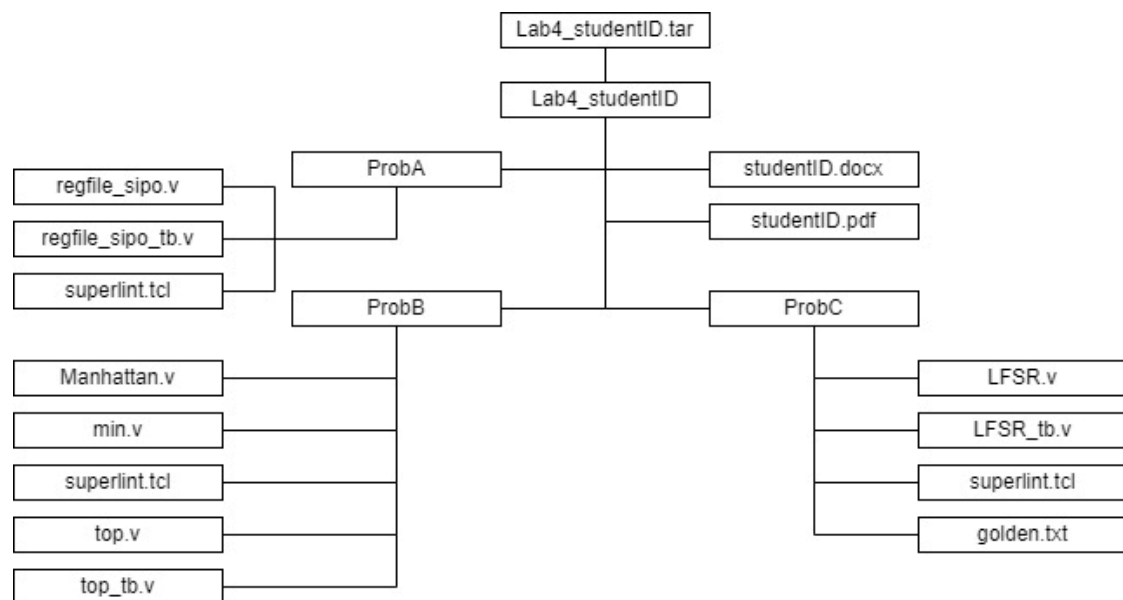
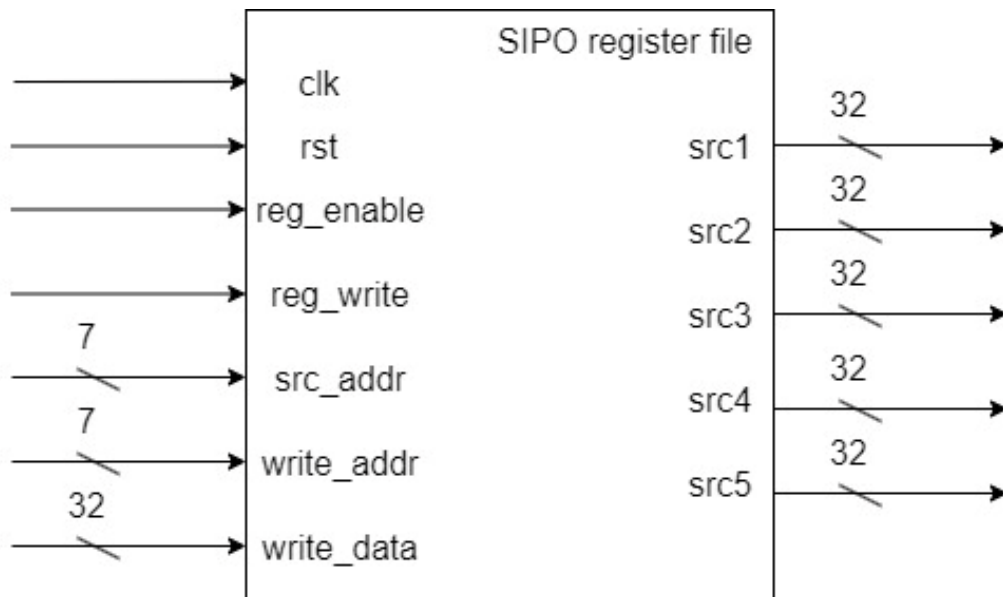


Fig.1 File hierarchy for Homework submission

Prob A: SIPO Register File



1. Based on the SIPO register file structure in LabA, please design a **128 x 32** SIPO register file with **5** output ports.
2. 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.

```

R[ 92] = 00000000
R[ 93] = 00000000
R[ 94] = 00000000
R[ 95] = 00000000
R[ 96] = 00000000
R[ 97] = 00000000
R[ 98] = 00000000
R[ 99] = 00000000
R[100] = 00000000
R[101] = 00000000
R[102] = 00000000
R[103] = 00000000
R[104] = 00000000
R[105] = ffff0005
R[106] = ffff0006
R[107] = ffff0007
R[108] = ffff0008
R[109] = ffff0009
R[110] = 00000000
R[111] = 00000000
R[112] = 00000000
R[113] = 00000000
R[114] = 00000000
R[115] = 00000000
R[116] = 00000000
R[117] = 00000000
R[118] = 00000000
R[119] = 00000000
R[120] = 00000000
R[121] = 00000000
R[122] = 00000000
R[123] = 00000000
R[124] = 00000000
R[125] = 00000000
R[126] = 00000000
R[127] = 00000000

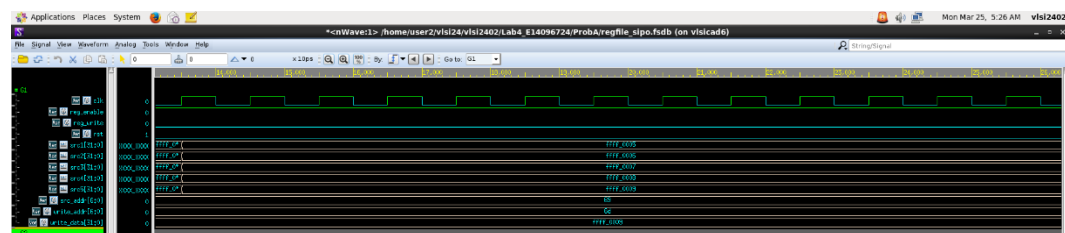
*****
**      /\_/_||
** Congratulations !! **      / 0.0 |
**                               **
** Simulation PASS!! **      + + + +
**                               **
*****

$finish called from file "regfile_sipo.tb.v", line 160.
$finish at simulation time 32500
VCS Simulation Report
Time: 328900 ps
CPU Time: 0.020 seconds: Data structure size: 0.0MB
Sun Mar 24 21:15:48 2024
CPU time: .372 seconds to compile + .443 seconds to elab + .395 seconds to link + .558 seconds in simulation
visicad6:/home/user2/visi24/visi2402/Lab4_E14096724/ProbA %

```

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

Read:



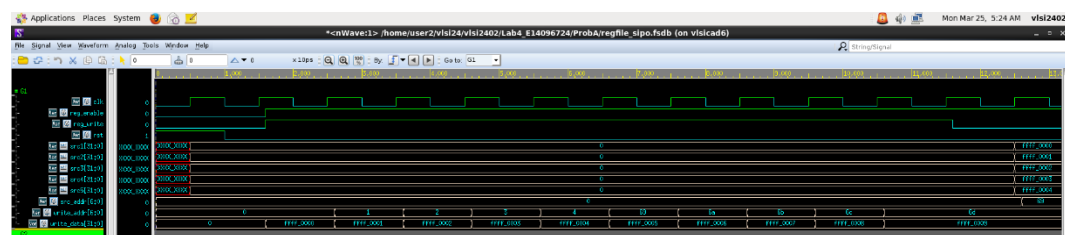
```

R[ 105] = ffff0005
R[ 106] = ffff0006
R[ 107] = ffff0007
R[ 108] = ffff0008
R[ 109] = ffff0009

```

可以從波形中看到，當 `reg_enable` 為 1 且 `reg_write` 為 0，在 `clock` 上升時讀取 `src_addr`(hexadecimal:69,decimal:105)地址中的資料以及後四筆共五筆資料(`src1~src5`)，可以從 register file 中看到讀取正確。

Write:



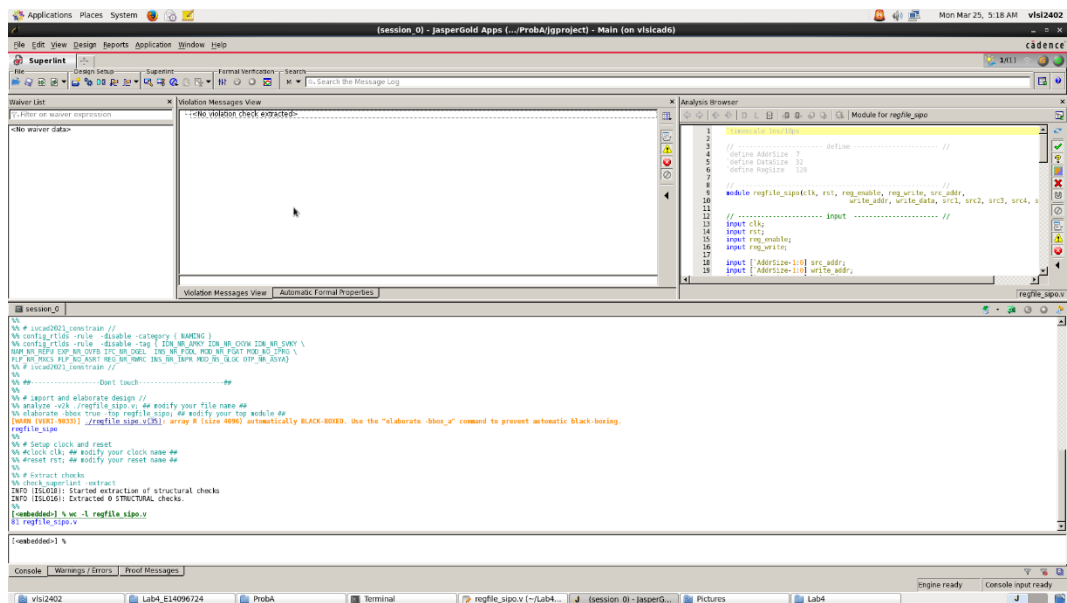
```

R[      0] = ffff0000
R[      1] = ffff0001
R[      2] = ffff0002
R[      3] = ffff0003
R[      4] = ffff0004

```

可以從波形圖中看到，當 reg_enable 為 1 且 reg_write 為 1，在 clock 上升時在 write_addr(hexadecimal:0~5,decimal:0~5)地址寫入 write_data 資料，可以從 register file 中看到讀取正確。

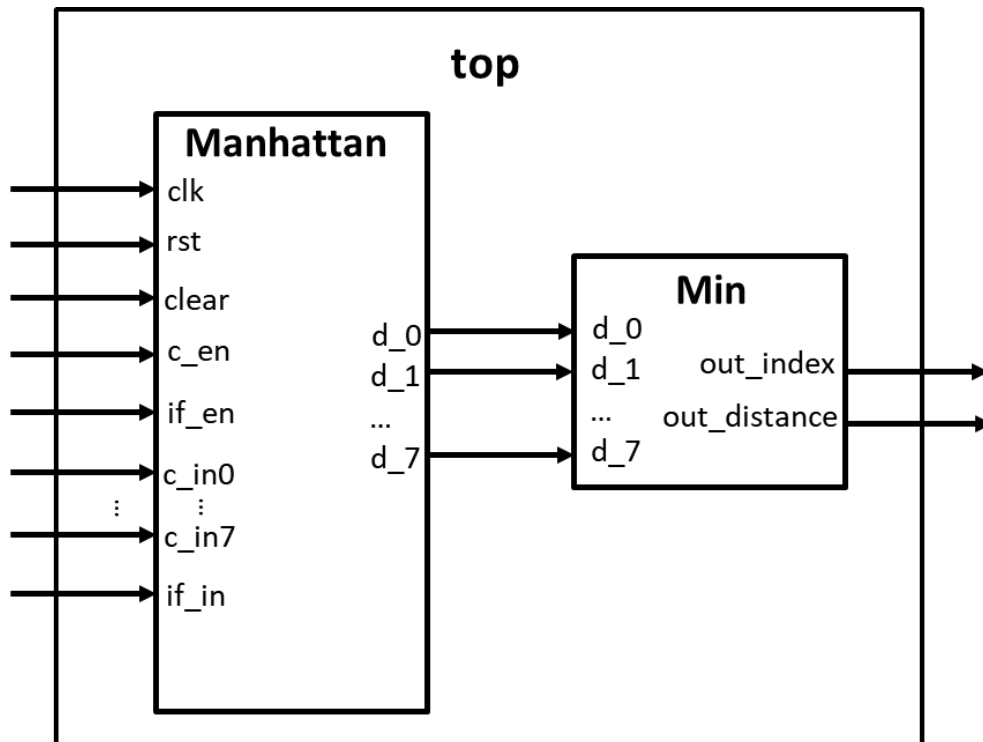
5. Show SuperLint coverage



Coverage :

Regfile_sipo.v : $(1 - 0/81) * 100\% = 100\%$

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.

```

[0] Applications Places System [0] [0] Mon Mar 25, 8:23 AM Vitis2402
Terminal
File Edit View Search Terminal Help
run VCS again.

Info: [VCS_SAVE_RESTORE_INFO] ASLR (Address Space Layout Randomization) is detected on the machine. To enable save functionality, ASLR will be switched off and slmv re-executed.
Please use "no-save slmv switch to avoid re-execution or -suppressASLR_DETECTED_INFO" to suppress this message.
Chronologic VCS simulator copyright 1991-2023
Contains Synopsys proprietary information.
Compiler version U-2023.03-SP2_Full64; Runtime version U-2023.03-SP2_Full64; Mar 25 08:17 2024
Warning : License for product VCS-BASE-RUNTIME will expire within 7 days, on: 31-mar-2024.

If you would like to temporarily disable this message, set
the VCS_LIC_EXPIRE_WARNING_environment variable to the number of days
before expiration that you want this message to start (the minimum is 0).
*Verdi - Loading Libraries vcsW202303.sp
FSDB Dumper for VCS, Release Verdi.U-2023.03-SP2, Linux x86_64/64bit, 08/28/2023
(C) 1990 - 2023 by Synopsys, Inc.
*Verdi - FSDB WARNMsg: The FSDB file already exists. Overwriting the FSDB file may crash the programs that are using this file.
*Verdi - Create FSDB file 'top.fsb'
*Verdi - Begin Traversing the scopes, layer (0).
*Verdi - Enable mda dumping.
*Verdi - End of traversing.

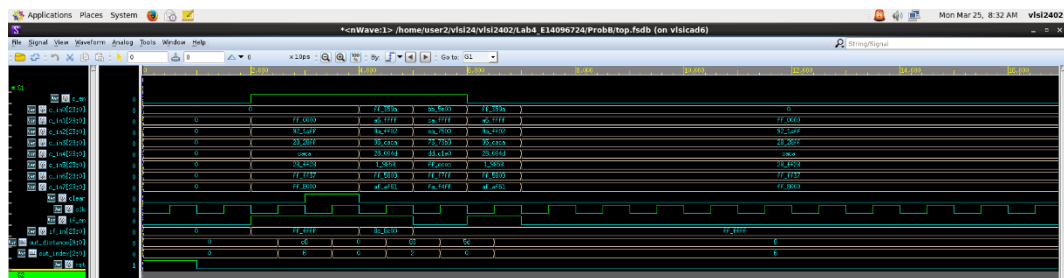
.....
time          35 output is correct
time          40 output is correct
time          45 output is correct
time          55 output is correct
time          60 output is correct
time          75 output is correct
time          85 output is correct
.....

*****
**                               \  | |
** Congratulations !!           / 0.0 |
**                               /---|
** Simulation PASS!!            ^---w|
**                               ^---w|
*****                               \__m |

$finish called from file 'top.tb.v', line 145,
$finish at simulation time 17000
      V C S   S i m u l a t i o n   R e p o r t
Time: 17000 ps
CPU Time:    0.30 seconds;      Data structure size: 0.0Mb
Mon Mar 25 08:18:21 2024
CPU time: .363 seconds in simulation
vcsinfo:/home/user7/vlsim2402/vlsim2402/job4 E14996724/ProbB %

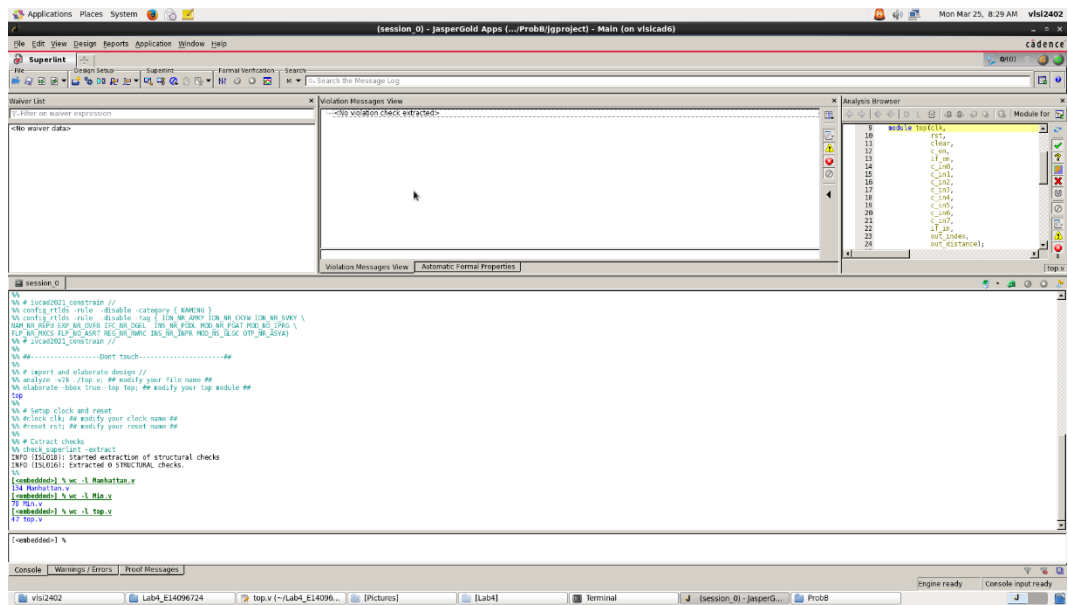
[0] Vitis2402 [0] LabA E14096724 [0] Miniv(-LabA E14096... [0] PICTUrss [0] [LabA] [0] ProbB [0] ProBA [0] DE Terminal [0]
```

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



可以從波形圖中看到，當 reg_enable 為 1 且 reg_write 為 1，在 clock 上升時在 write_addr(hexadecimal:0~5,decimal:0~5)地址寫入 write_data 資料，可以從 register file 中看到讀取正確。

5. Show SuperLint coverage



Coverage :

Manhattan.v : $(1 - 0/134) * 100\% = 100\%$

Min.v : $(1 - 0/78) * 100\% = 100\%$

top.v : $(1 - 0/47) * 100\% = 100\%$

Prob C: LFSR

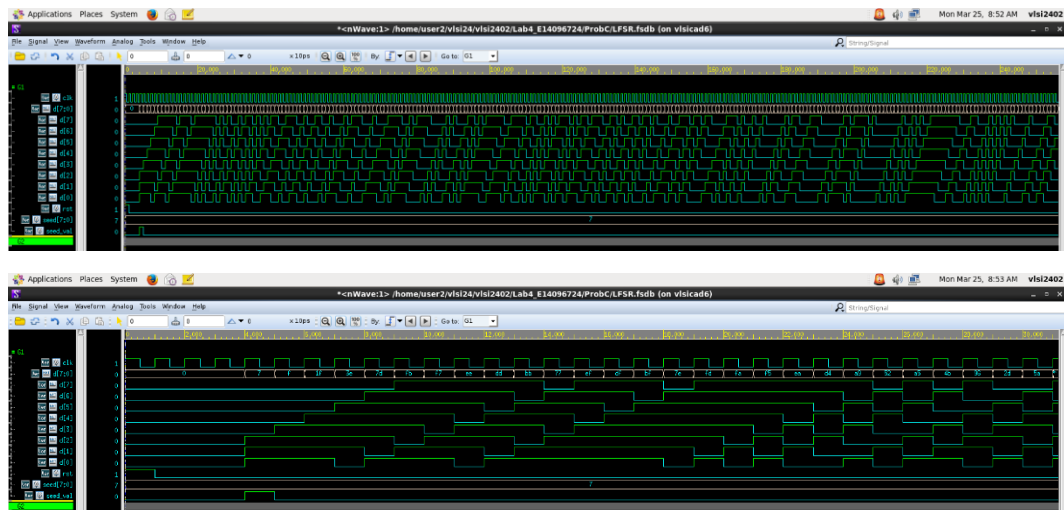
1. Please design an 8-bit-LFSR, with the given feedback function in the LAB4 slide.
2. Port list

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

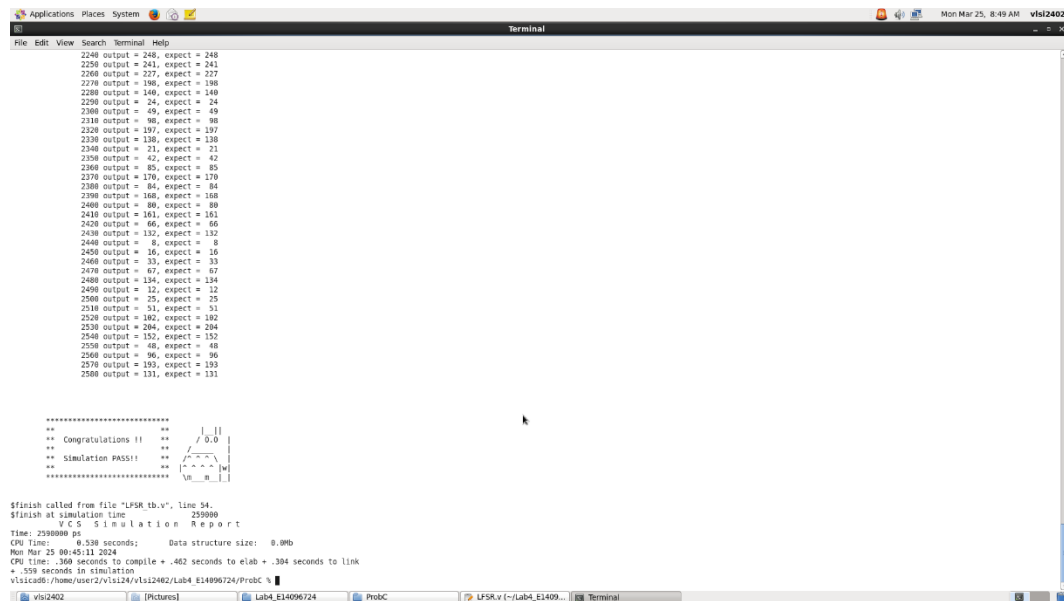
$$d[0] = (d[7] \wedge d[5]) \wedge (d[4] \wedge d[2])$$

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

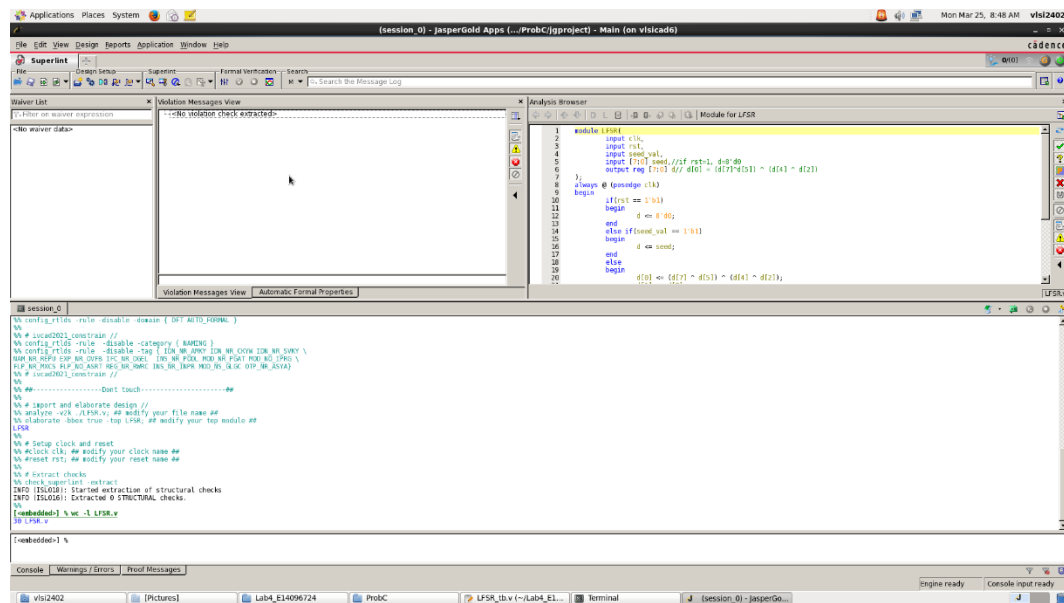


可以從上方的波形圖中看到，在 clock 上升時 d[0]~d[6]會向後傳遞到 d[1]~d[7]，並且可以從下方的波形圖中看到 d[0]確實有依照 $d[0] = (d[7] \wedge d[5]) \wedge (d[4] \wedge d[2])$ 來產生。

5. Show the simulation result on the terminal.



6. Show SuperLint coverage



Coverage :

$$\text{LFSR.v} : (1 - 0/30) * 100\% = 100\%$$

At last, please write the lesson you learned from Lab4

這次的 Lab 讓我學習到有關 SIPO register file、Manhattan、Min 和 8-bit-LFSR 在硬體上設計，SIPO register file 讓我更了解了有關記憶體的操作，Manhattan 和 Min 也讓我學到計算顏色和顏色之間的距離和取最小值的作法，LFSR 也讓我學到亂數是可以用這種傳遞的方式來產生，另外這次的 Lab 讓我學習到如何修改程式來讓 superlint 中的錯誤和警告減少，像是如何修改程式來解決 propagation race、multiple drive 的問題，來讓我程式的 coverage 達到 100%。

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