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	
簡笠恩	E14102305	
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

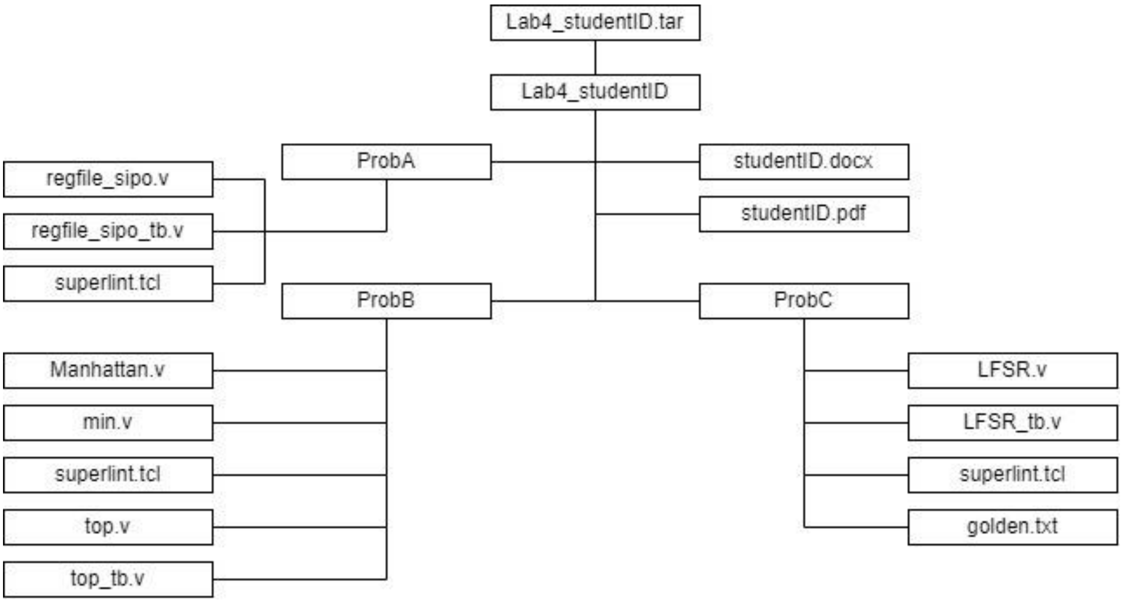
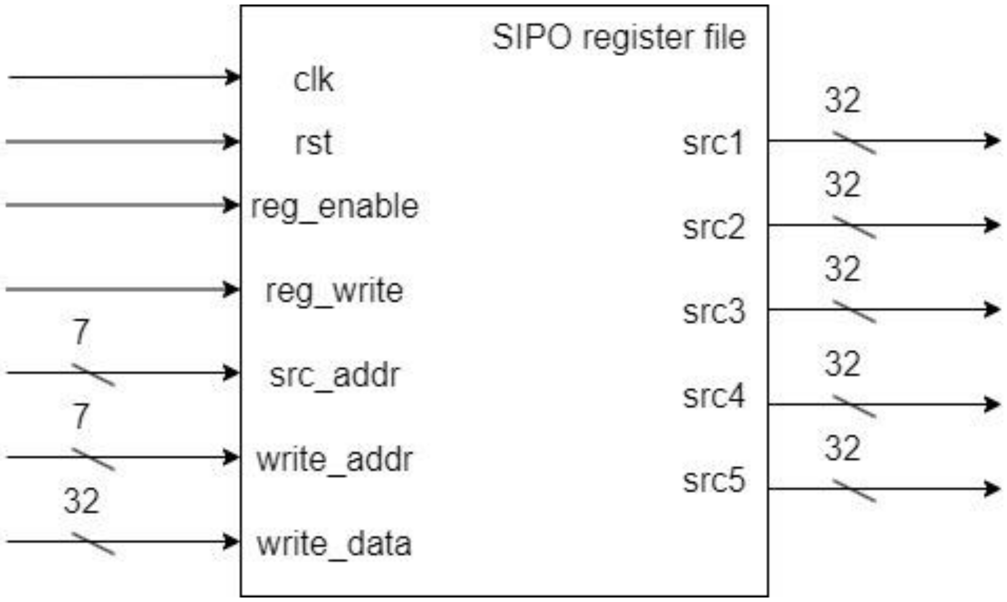


Fig.1 File hierarchy for Homework submission

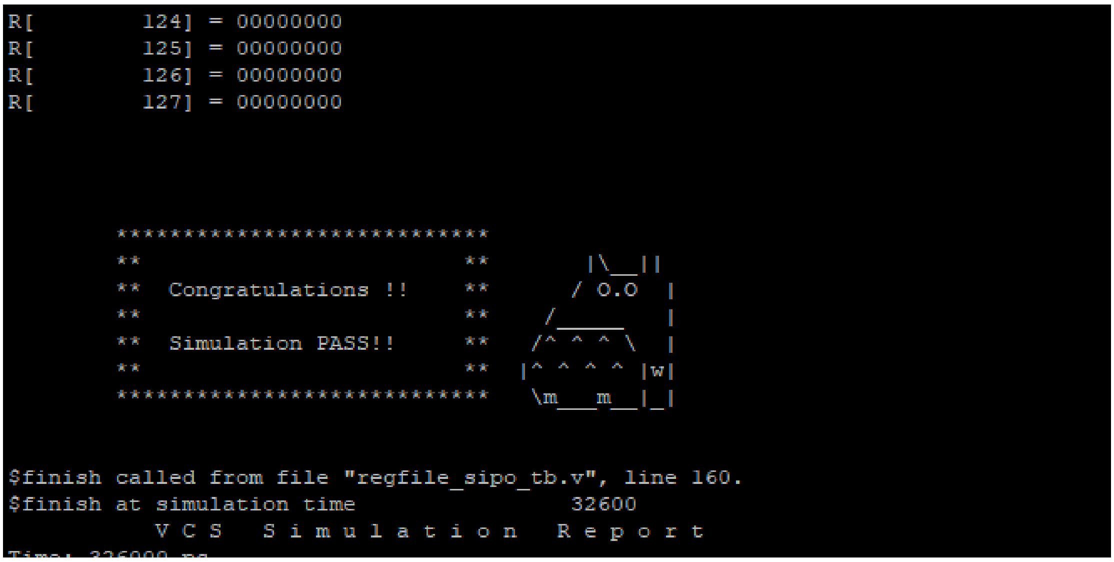
Prob A: SIPO Register File



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.



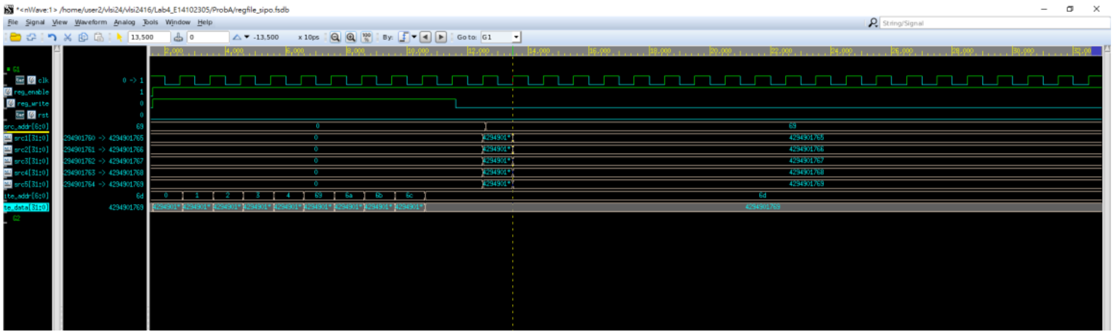
```
R[         124] = 00000000
R[         125] = 00000000
R[         126] = 00000000
R[         127] = 00000000



        ****************************
        **                        **       |\__||
        **  Congratulations !!    **      / O.O  |
        **                        **    /_____   |
        **  Simulation PASS!!     **   /^ ^ ^ \  |
        **                        **  |^ ^ ^ ^ |w|
        ****************************   \m___m__|_|


$finish called from file "regfile_sipo_tb.v", line 160.
$finish at simulation time                32600
           V C S   S i m u l a t i o n   R e p o r t
Time: 326000 ps
```

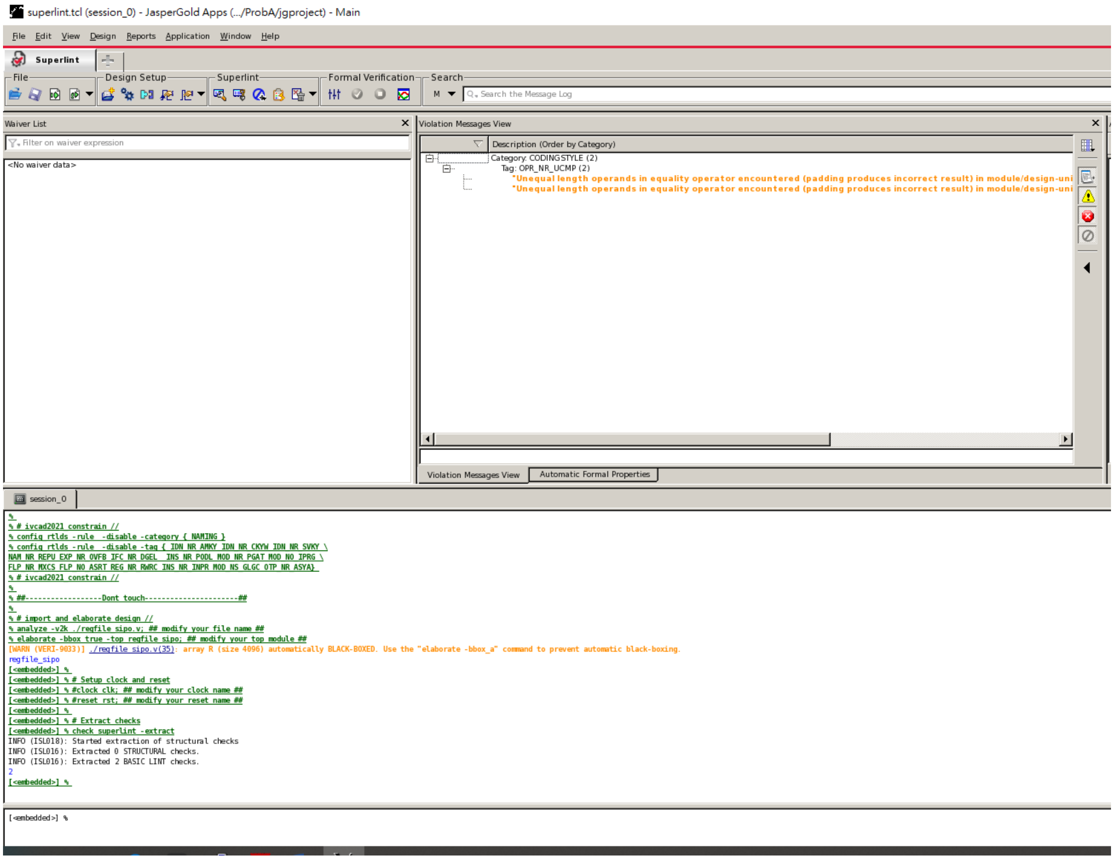
4. Show waveforms to explain that your register work correctly when read and write.



從此圖可知 reg_write = 0 為 read 模式故

我們第一個要讀的是 69 暫存器所以 src1 為其內的值，後面四個為 6a,6b,6c,6d 的值，src5 會是 69 結尾，如圖所示，前面四個也是如此。

5. Show SuperLint coverage


superlint.tcl (session_0) - JasperGold Apps (.../ProbA/jgproject) - Main
File Edit View Design Reports Application Window Help
Superlint
File
Design Setup
Superlint
Formal Verification
Search
M
Search the Message Log
Waiver List
Filter on waiver expression
Violation Messages View
Description (Order by Category)
Violation Messages View
Automatic Formal Properties
session_0

```
<No waiver data>
```

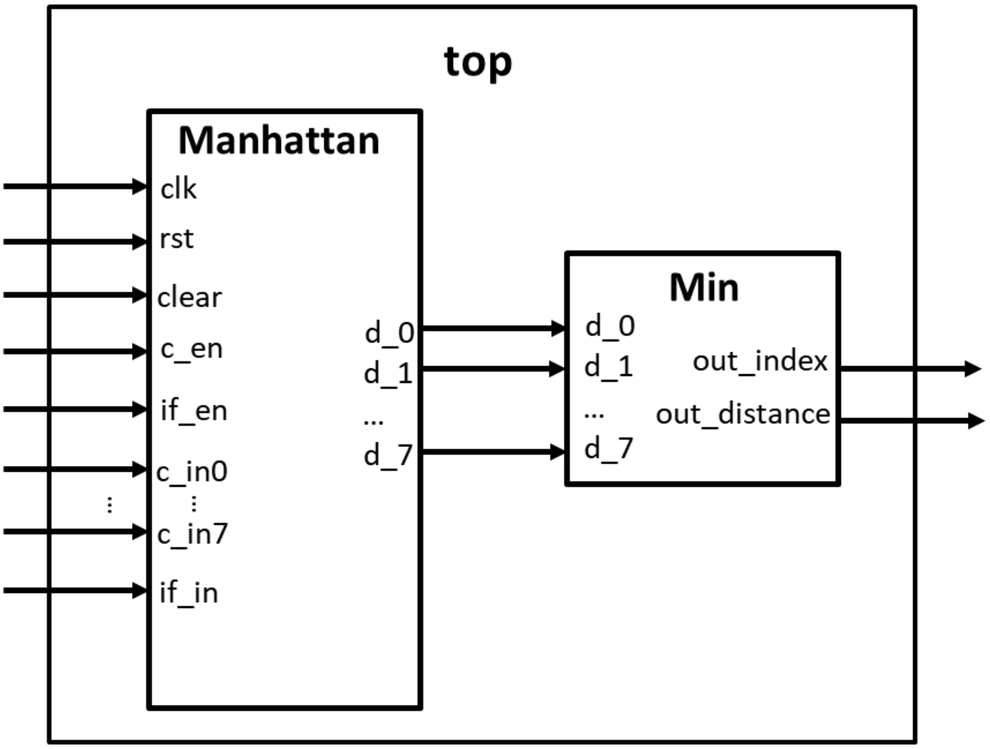
```
Category: CODINGSTYLE (2)
  Tag: OPR_NR_UCMP (2)
    "Unequal length operands in equality operator encountered (padding produces incorrect result) in module/design-uni
    "Unequal length operands in equality operator encountered (padding produces incorrect result) in module/design-uni
```

```
%
% # ivcad2021 constrain //
% config_rtlds -rule  -disable -category { NAMING }
% config_rtlds -rule  -disable -tag { IDN_NR_AMKY IDN_NR_CKYW IDN_NR_SVKY \
NAM_NR_REPU EXP_NR_OVFB IFC_NR_DGEL  INS_NR_PODL MOD_NR_PGAT MOD_NO_IPRG \
FLP_NR_MXCS FLP_NO_ASRT REG_NR_RWRC INS_NR_INPR MOD_NS_GLGC OTP_NR_ASYA}
% # ivcad2021 constrain //
%
% ##-----------------Dont touch---------------------##
%
% # import and elaborate design //
% analyze -v2k ./regfile_sipo.v; ## modify your file name ##
% elaborate -bbox true -top regfile_sipo; ## modify your top module ##
[WARN (VERI-9033)] ./regfile_sipo.v(35): array R (size 4096) automatically BLACK-BOXED. Use the "elaborate -bbox_a" command to prevent automatic black-boxing.
regfile_sipo
[<embedded>] %
[<embedded>] % # Setup clock and reset
[<embedded>] % #clock clk; ## modify your clock name ##
[<embedded>] % #reset rst; ## modify your reset name ##
[<embedded>] %
[<embedded>] % # Extract checks
[<embedded>] % check_superlint -extract
INFO (ISL018): Started extraction of structural checks
INFO (ISL016): Extracted 0 STRUCTURAL checks.
INFO (ISL016): Extracted 2 BASIC LINT checks.
2
[<embedded>] %
```

```
[<embedded>] %
```

1 – 2/69 = 97%

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

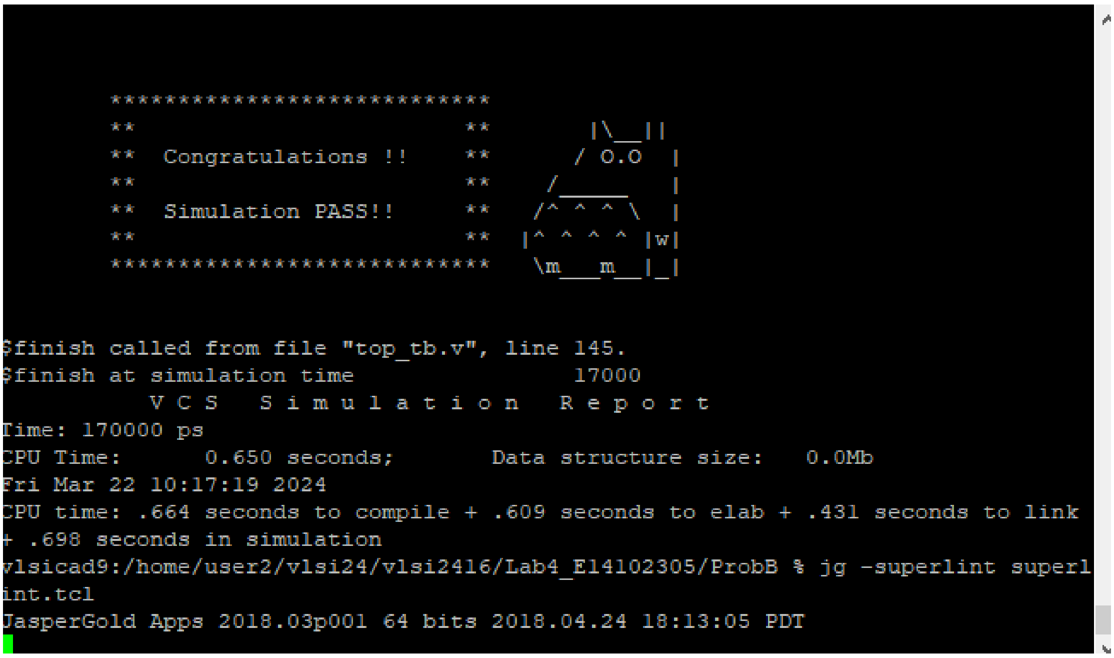
Manhattan:

Signal	Type	Bits	Description
clk	input	1	Clock pin.
rst	input	1	Reset pin.
clear	input	1	Set all registers to 0.
c_en	input	1	Write compared colors enable. When c_en is high, then c_in0~7 is available.
if_en	input	1	Write input pixel enable. When if_en is high, then if_in is available.

Min:

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

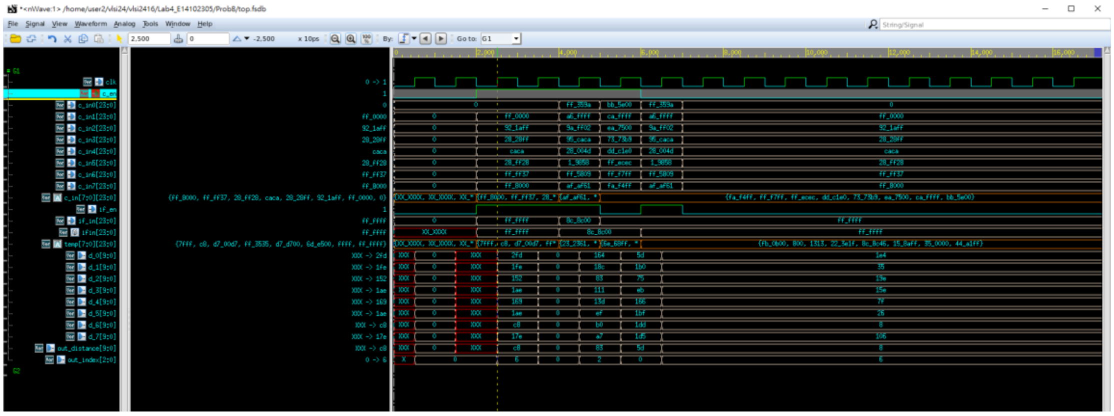
3. Show the simulation result on the terminal.



```
          ****************************
          **                        **        |\__||
          **   Congratulations !!   **       / O.O  |
          **                        **      /_____  |
          **   Simulation PASS!!    **     /^ ^ ^ \  |
          **                        **    |^ ^ ^ ^ |w|
          ****************************     \m___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: 170000 ps
CPU Time:      0.650 seconds;       Data structure size:   0.0Mb
Fri Mar 22 10:17:19 2024
CPU time: .664 seconds to compile + .609 seconds to elab + .431 seconds to link
+ .698 seconds in simulation
vlsicad9:/home/user2/vlsi24/vlsi2416/Lab4_E14102305/ProbB % jg -superlint superl
int.tcl
JasperGold Apps 2018.03p001 64 bits 2018.04.24 18:13:05 PDT
```

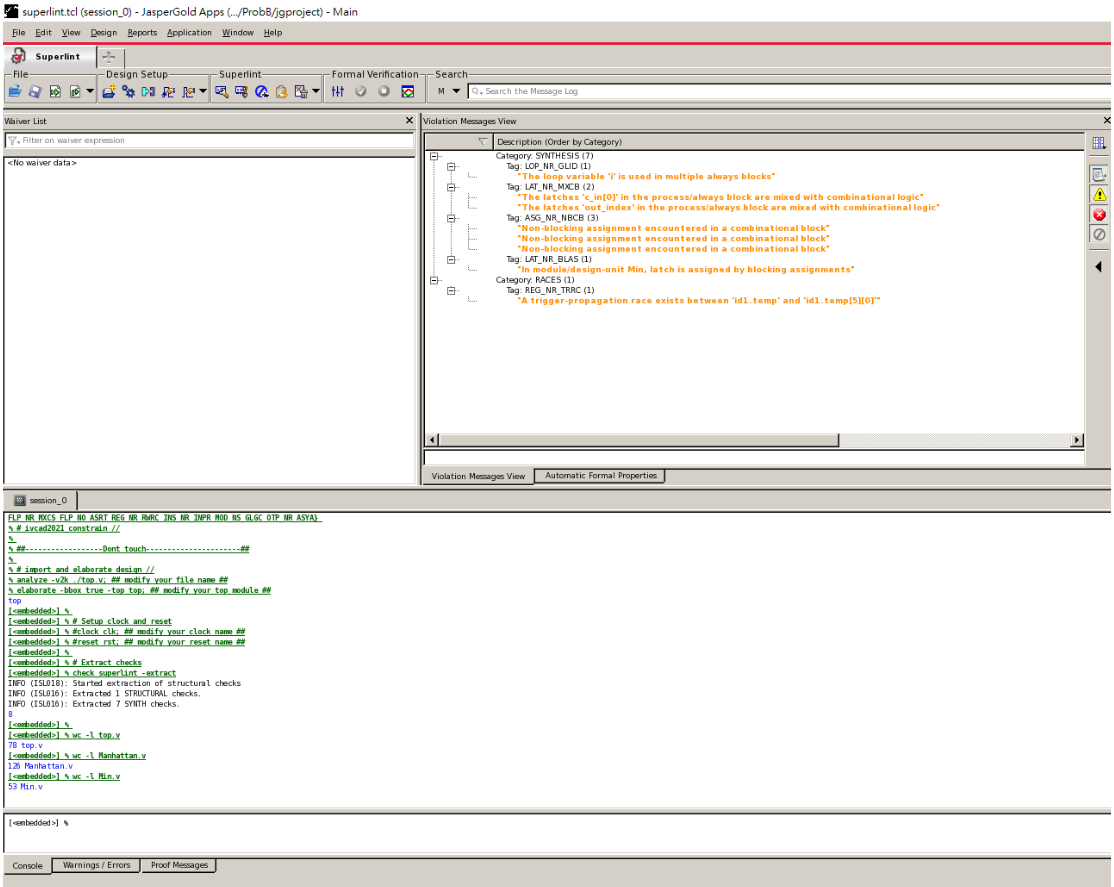
4. Show waveforms to explain that your design works correctly.



```
*<nWave:1> /home/user2/vlsi24/vlsi2416/Lab4_E14102305/ProbB/top.fsdb
G1
clk                  0 -> 1
c_en                 1
c_in0[23:0]          0
c_in1[23:0]          ff_0000
c_in2[23:0]          92_1aff
c_in3[23:0]          28_28ff
c_in4[23:0]          caca
c_in5[23:0]          28_ff28
c_in6[23:0]          ff_ff37
c_in7[23:0]          ff_8000
c_in[7:0][23:0]      {ff_8000, ff_ff37, 28_ff28, caca, 28_28ff, 92_1aff, ff_0000, 0}
if_en                1
if_in[23:0]          ff_ffff
ifin[23:0]           ff_ffff
temp[7:0][23:0]      {7fff, c8, d7_00d7, ff_3535, d7_d700, 6d_e500, ffff, ff_ffff}
d_0[9:0]             XXX -> 2fd
d_1[9:0]             XXX -> 1fe
d_2[9:0]             XXX -> 152
d_3[9:0]             XXX -> 1ae
d_4[9:0]             XXX -> 169
d_5[9:0]             XXX -> 1ae
d_6[9:0]             XXX -> c8
d_7[9:0]             XXX -> 17e
out_distance[9:0]    XXX -> c8
out_index[2:0]       0 -> 6
G2
```

我們已第一題來說，當在 c_en 升起，會將每個 c_in 的值存入自製暫存器，而 input_feature 也是如此，且在他們值有變化時 temp 就會去計算他們之間的最小路徑的 3 個 part(RGB)並存在 temp 裡，然後會在時脈正源時，將 temp 的 RGB 相加存入個別的 d 裡，並且送入 min 裡判斷最小值是誰，最後將最小值送回去 output_distance 裡，並且判斷他是哪個 C_in。

5. Show SuperLint coverage



superlint.tcl (session_0) - JasperGold Apps (.../ProbB/jgproject) - Main
File Edit View Design Reports Application Window Help
Superlint
File
Design Setup
Superlint
Formal Verification
Search
M
Search the Message Log
Waiver List
Filter on waiver expression
<No waiver data>
Violation Messages View
Description (Order by Category)
Violation Messages View
Automatic Formal Properties
session_0
Console
Warnings / Errors
Proof Messages

```
Category: SYNTHESIS (7)
  Tag: LOP_NR_GLID (1)
    "The loop variable 'i' is used in multiple always blocks"
  Tag: LAT_NR_MXCB (2)
    "The latches 'c_in[0]' in the process/always block are mixed with combinational logic"
    "The latches 'out_index' in the process/always block are mixed with combinational logic"
  Tag: ASG_NR_NBCB (3)
    "Non-blocking assignment encountered in a combinational block"
    "Non-blocking assignment encountered in a combinational block"
    "Non-blocking assignment encountered in a combinational block"
  Tag: LAT_NR_BLAS (1)
    "In module/design-unit Min, latch is assigned by blocking assignments"
Category: RACES (1)
  Tag: REG_NR_TRRC (1)
    "A trigger-propagation race exists between 'id1.temp' and 'id1.temp[5][0]'"
```

```
FLP_NR_MXCS FLP_NO_ASRT REG_NR_RWRC INS_NR_INPR MOD_NS_GLGC OTP_NR_ASYA}
% # ivcad2021 constrain //
%
% ##-----------------Dont touch---------------------##
%
% # import and elaborate design //
% analyze -v2k ./top.v; ## modify your file name ##
% elaborate -bbox true -top top; ## modify your top module ##
top
[<embedded>] %
[<embedded>] % # Setup clock and reset
[<embedded>] % #clock clk; ## modify your clock name ##
[<embedded>] % #reset rst; ## modify your reset name ##
[<embedded>] %
[<embedded>] % # Extract checks
[<embedded>] % check_superlint -extract
INFO (ISL018): Started extraction of structural checks
INFO (ISL016): Extracted 1 STRUCTURAL checks.
INFO (ISL016): Extracted 7 SYNTH checks.
8
[<embedded>] %
[<embedded>] % wc -l top.v
78 top.v
[<embedded>] % wc -l Manhattan.v
126 Manhattan.v
[<embedded>] % wc -l Min.v
53 Min.v

[<embedded>] %
```

1-8/78+126+53 = 96.89%

Prob C: LFSR

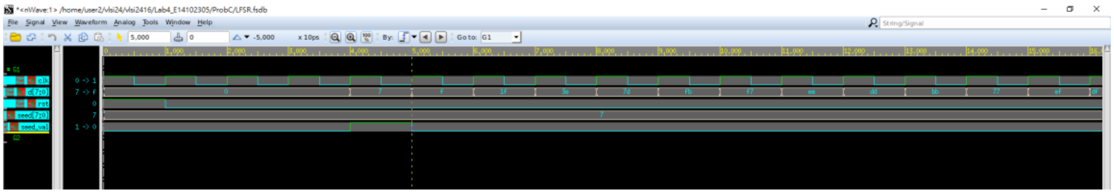
1. Please design an 8-bit-LFSR, with the given feedback function in the LAB4 slide.
2. Port list

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

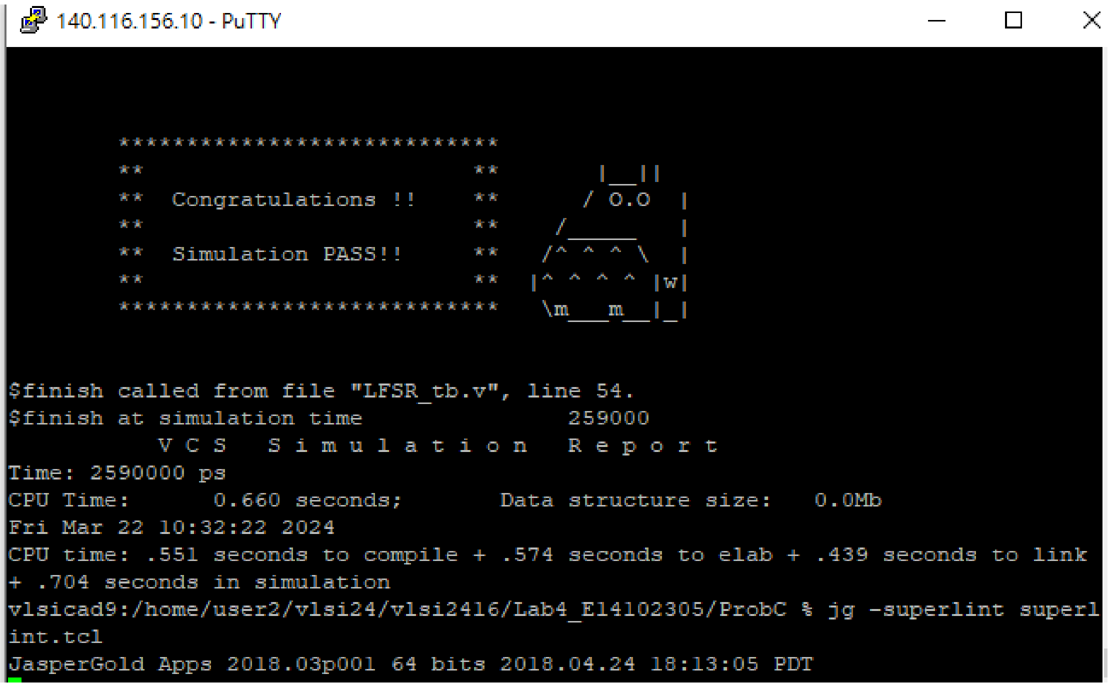
$$d[0] = (d[7] \wedge d[5]) \wedge (d[4] \wedge d[2])$$

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



在 seed_val=1 時，會將 output d 存入 seed 值故為 7，在下一個時脈正源，因為 seed_val=0，故開始線性回饋，起始 7 為 8’b00000111，經過線性 F→d[0] = (0^0)^(0^1)→0^1→1，故會變成 8’b00001111 即為 0f。

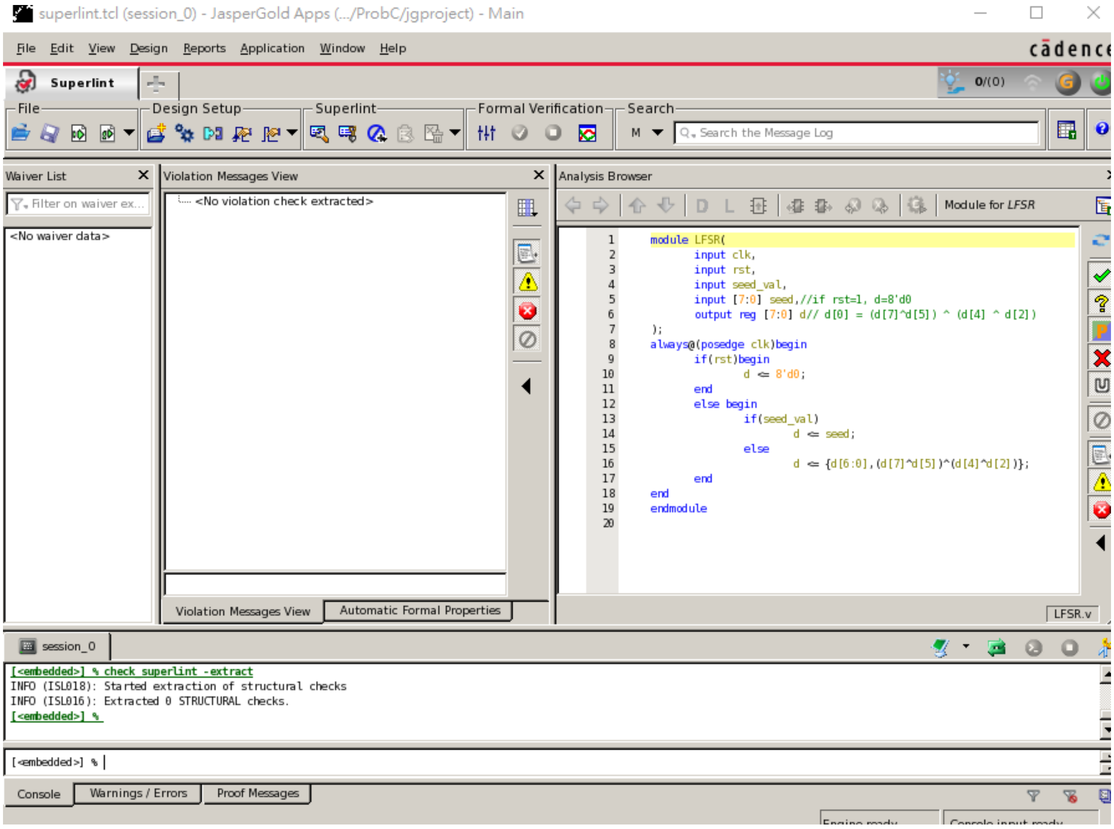
5. Show the simulation result on the terminal.



```
        ****************************
        **                        **       |__||
        **  Congratulations !!    **      / O.O  |
        **                        **    /_____   |
        **  Simulation PASS!!     **   /^ ^ ^ \  |
        **                        **  |^ ^ ^ ^ |w|
        ****************************   \m___m__|_|

$finish called from file "LFSR_tb.v", line 54.
$finish at simulation time              259000
           V C S   S i m u l a t i o n   R e p o r t
Time: 2590000 ps
CPU Time:      0.660 seconds;       Data structure size:   0.0Mb
Fri Mar 22 10:32:22 2024
CPU time: .551 seconds to compile + .574 seconds to elab + .439 seconds to link
+ .704 seconds in simulation
vlsicad9:/home/user2/vlsi24/vlsi2416/Lab4_E14102305/ProbC % jg -superlint superl
int.tcl
JasperGold Apps 2018.03p001 64 bits 2018.04.24 18:13:05 PDT
```

6. Show SuperLint coverage



```
module LFSR(
        input clk,
        input rst,
        input seed_val,
        input [7:0] seed,//if rst=1, d=8'd0
        output reg [7:0] d// d[0] = (d[7]^d[5]) ^ (d[4] ^ d[2])
);
always@(posedge clk)begin
        if(rst)begin
                d <= 8'd0;
        end
        else begin
                if(seed_val)
                        d <= seed;
                else
                        d <= {d[6:0],(d[7]^d[5])^(d[4]^d[2])};
        end
end
endmodule
```

```
[<embedded>] % check_superlint -extract
INFO (ISL018): Started extraction of structural checks
INFO (ISL016): Extracted 0 STRUCTURAL checks.
[<embedded>] %
```

```
[<embedded>] %
```

100%

At last, please write the lesson you learned from Lab4

我覺得學到了關於 sequential 的寫法，且更了解暫存器的運作，還有 non-blocking 與 blocking 的區別，還有 for 迴圈的運用會使得程式較為簡化。

老話一句，好課一生推。

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