National Cheng Kung University

Department of Electrical Engineering

Introduction to VLSI CAD (Spring 2023)

Lab Session 4

Register Files, Manhattan Distance and LFSR

Name	Student ID	
鄭智宇	E94096110	
Practical Sections	**Points**	**Marks**
Prob A	25	
Prob B	25	
Prob C	25	
Report	15	
File hierarchy, naming…etc.	5	
Superlint	5	
Notes:		

Due Date: 15:00, March 23, 2020 @ moodle

Deliverables

1) All Verilog codes including testbenches for each problem should be uploaded.
   NOTE: Please **DO NOT** paste source code in the report!
2) Noted! TA will use commands in Appendix A to check your design in SoC Lab. If TA can not compile your code with the commands, you will not get full credit.
3) If you upload a dead body which we can't even compile, you will get **NO** credit!
4) All Verilog file should get at least 85% SuperLint Coverage.
5) All homework requirements should be uploaded in this file hierarchy or you will not get full credit.
   NOTE: Please **DO NOT** upload waveforms!

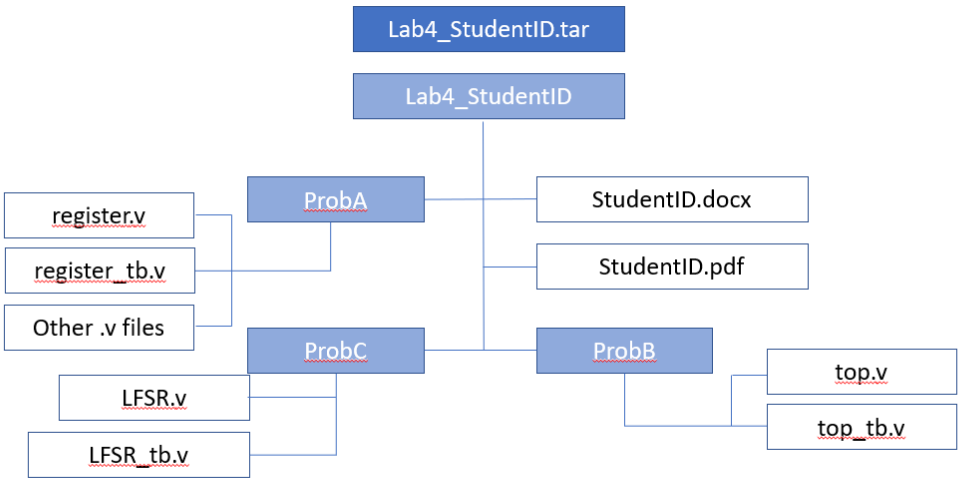
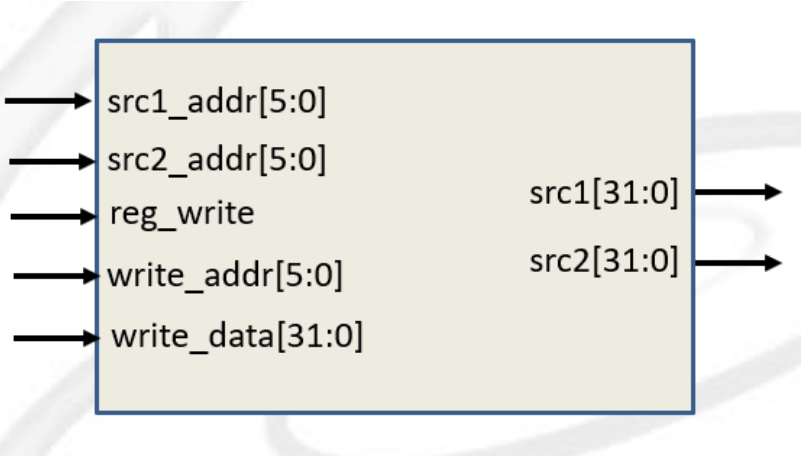


Fig.1 File hierarchy for Homework submission

Prob A: Register File

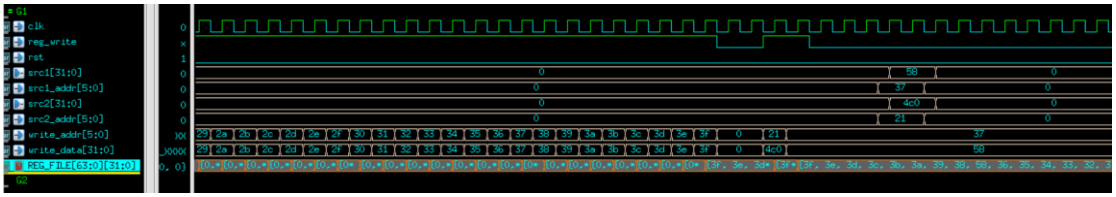


1. Based on the register file structure in LabA, please design a 64 x 32 register file by yourself.
2. Port list

Signal	Type	Bits	Description
clk	input	1	clock
rst	input	1	reset
reg_write	input	1	0 → read, 1 → write
src1_addr	input	6	source1 address
src2_addr	input	6	source2 address
write_addr	input	6	write address
write_data	input	32	write data
src1	output	32	read data source1
src2	output	32	read data source2

3. You should follow the file name rules as follow.
   - Register file
     - File name: **register.v**
     - Module name: **register**
   - Register file testbench
     - File name: **register_tb.v**

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



For REG_FILE

當 reg_write 為 1 時，將 write_data 寫入地址為 write_addr 的 **REG_FILE** 中

當 reg_write 為 0 時，所有地址為的 **REG_FILE** 的值保持不變

For src1, src2

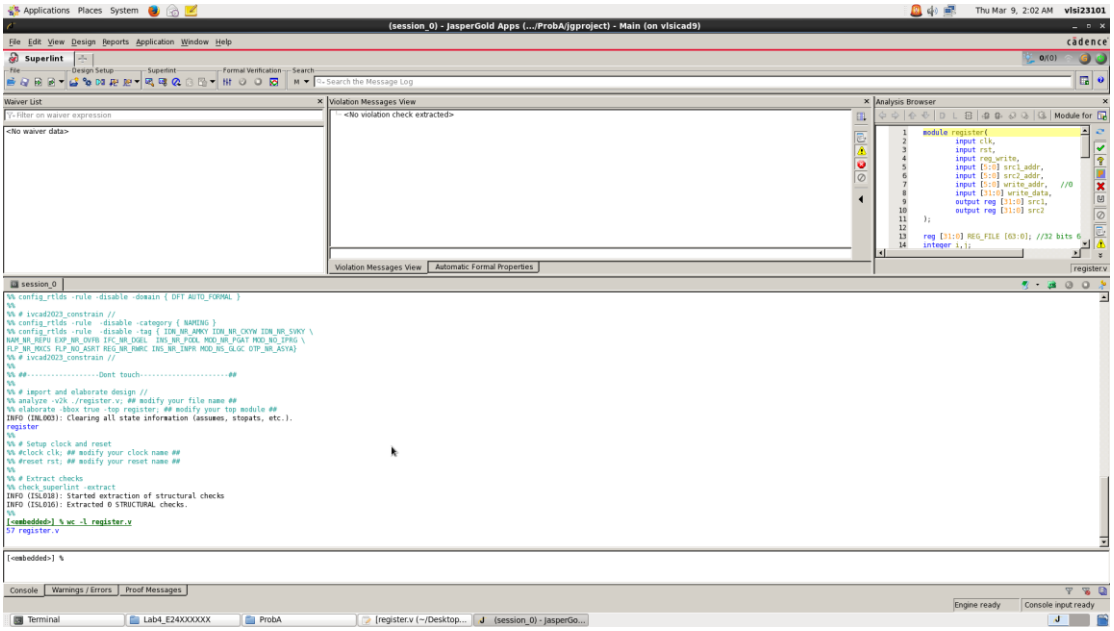
當 reg_write 為 0 時，src1 讀取 src1_addr 指定的暫存器的值

當 reg_write 為 1 時，src1 的值保持不變

當 reg_write 為 0 時，src2 讀取 src2_addr 指定的暫存器的值

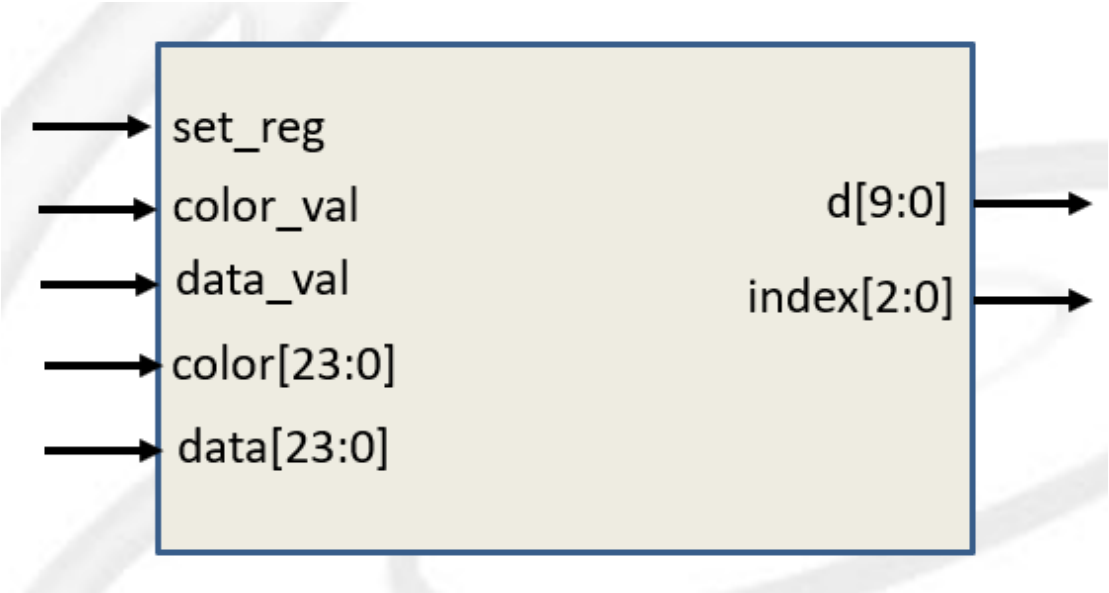
當 reg_write 為 1 時，src2 的值保持不變

5. Show SuperLint coverage



Coverage : 100 %

Prob B: Finding Smallest Distance

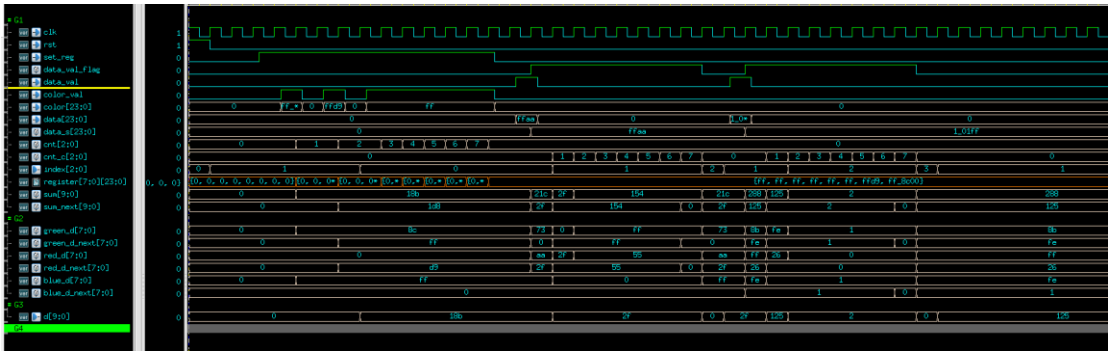


Please design a circuit that will find the smallest distance between the local registers and the input data, based on the structure given in the LAB4 slide.

Port list

Signal	Type	Bits	Description
clk	input	1	Clock pin.
rst	input	1	Reset pin.
set_reg	input	1	1: write mode. In this mode, when the hardware reads color_val, it will fill register[0] with color, wait for the next color_val and fill register[1]… and so on, until set_reg=0. 0: keep the color registers the same.
color_val	input	1	1: color data is valid, color shall be written into the local register if set_reg=1. 0: color data isn't available, keep the local registers the same.
data_val	input	1	1: input data is valid, output shall be available after 8 cycles. 0: input data isn't available.
color	input	24	color data that will be stored in the local register.
data	input	24	Input data that will be compared among all of the color registers.
d	output	10	Output distance data.
index	output	3	Output index. If there are 2 or more colors that has the same smallest distance between input data, output the one that has the smallest index.

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

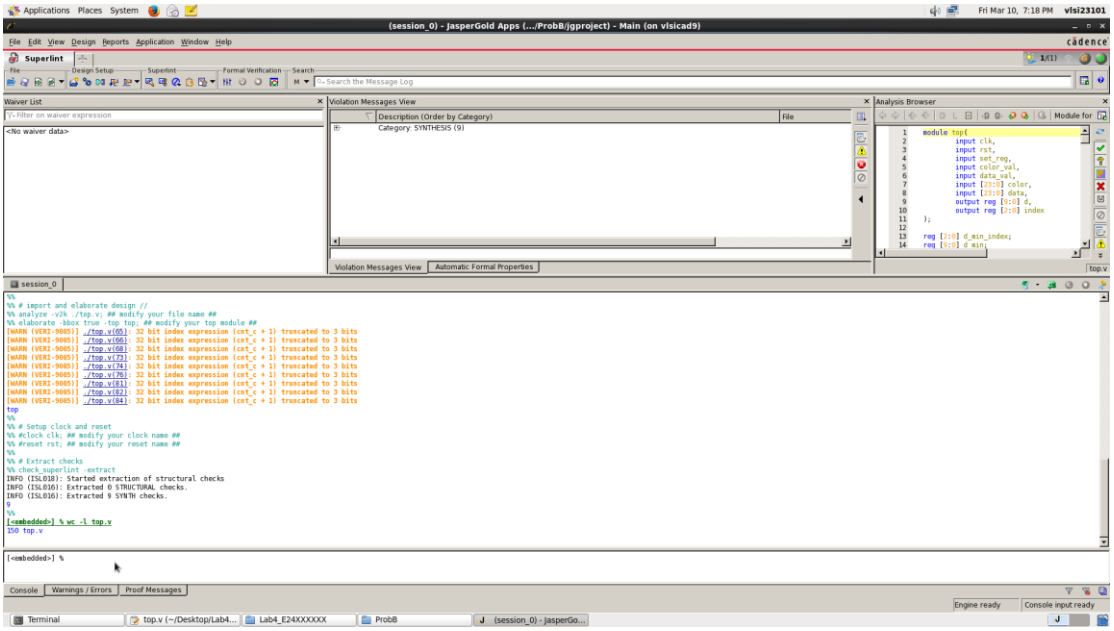


1. register:
    - 8 個元素的 register 陣列，每個元素是一個 24-bit 寬的 reg。
    - 這個陣列用於存儲顏色數據
2. cnt_c:
    - 3-bit 寬的 reg，
    - 用於計算最近的顏色向量與輸入數據之間的距離。
3. cnt:
    - 3-bit 寬的 reg，用於計數要更新哪個顏色向量。
    - cnt 用於指示下一個要更新的顏色向量的索引。
4. data_val_flag:
    - 用於標識輸入數據的有效性。
    - 如果 data_val_flag 是 1，則表示當前輸入數據有效，可以更新 register 中的數據。否則，將忽略輸入數據。
5. blue_d, green_d, red_d:
   這些都是 8-bit 寬的 reg，用於計算輸入數據與 register 中每個顏色向量之間的距離。
6. blue_d_next, green_d_next, red_d_next:
   這些都是 8-bit 寬的 reg，用於計算輸入數據與下一個 register 中的每個顏色向量之間的距離。
7. sum, sum_next:
   這些都是 10-bit 寬的 reg，用於計算輸入數據與 register 中每個顏色向量之間的總距離，用於決定 d 與 index。
8. data_s:
   這是一個 24-bit 寬的 reg，用於存儲最近的輸入數據。當 data_val 信號有效時，data_s 將更新為輸入數據。
9. d, index:
   這些都是輸出信號，分別是 10-bit 寬和 3-bit 寬的 reg。它們用於記錄最近的顏色向量和其索引。當 cnt_c = 0 時，d 和 index 將被更新。

運行步驟:

I. 宣告了一個大小為 8 的暫存器陣列 register，其中每個元素都是 24bits reg。
II. 設置了一個 data_s 變數，其中存儲了從 data 輸入獲取的 24 位數據。
III. 計算與記錄了 data_s 與 register 中指定的寄存器之間的距離，分別是藍色分量、綠色分量和紅色分量，以及與下一個寄存器之間的距離。
IV. 計算並記錄了 sum 和 sum_next，分別表示 data_s 與 register[cnt_c]和 register[cnt_c+1]之間的總距離。
V. 根據 sum 和 sum_next 的值，更新 d 和 index 變數的值，以選擇最接近的顏色寄存器。
VI. 計算和更新了 cnt_c 變數的值，以指示目前正在檢查哪個寄存器。
VII. 計算和更新了 cnt 變數的值，以指示目前正在設置哪個寄存器。
VIII. 初始化了 register 陣列，並在 set_reg 和 color_val 信號接收到時，將顏色值存儲在選定的寄存器中。

2. Show SuperLint coverage



9 warning

Coverage : 94%

Prob C: LFSR

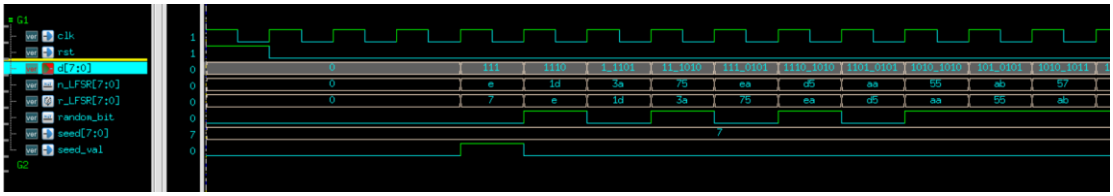
1. Please design an 8-bit-LFSR, with the given feedback function in the LAB4 slide.
2. Port list

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

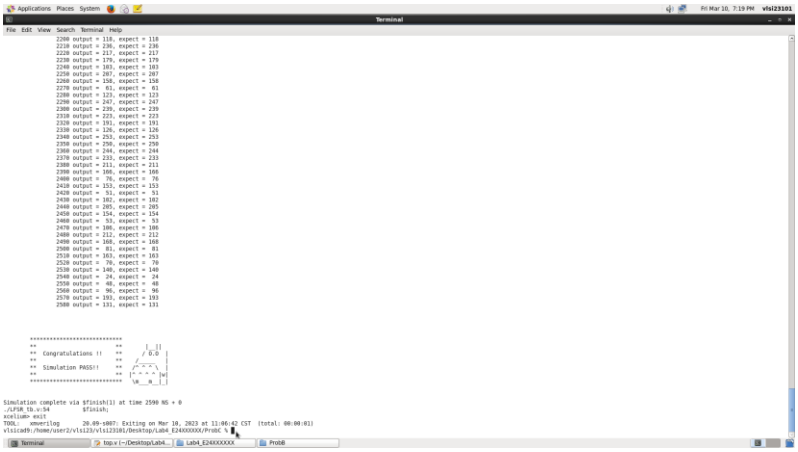
$$d[0] = (d[7] \wedge \sim d[5]) \wedge (d[4] \wedge \sim d[3])$$

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

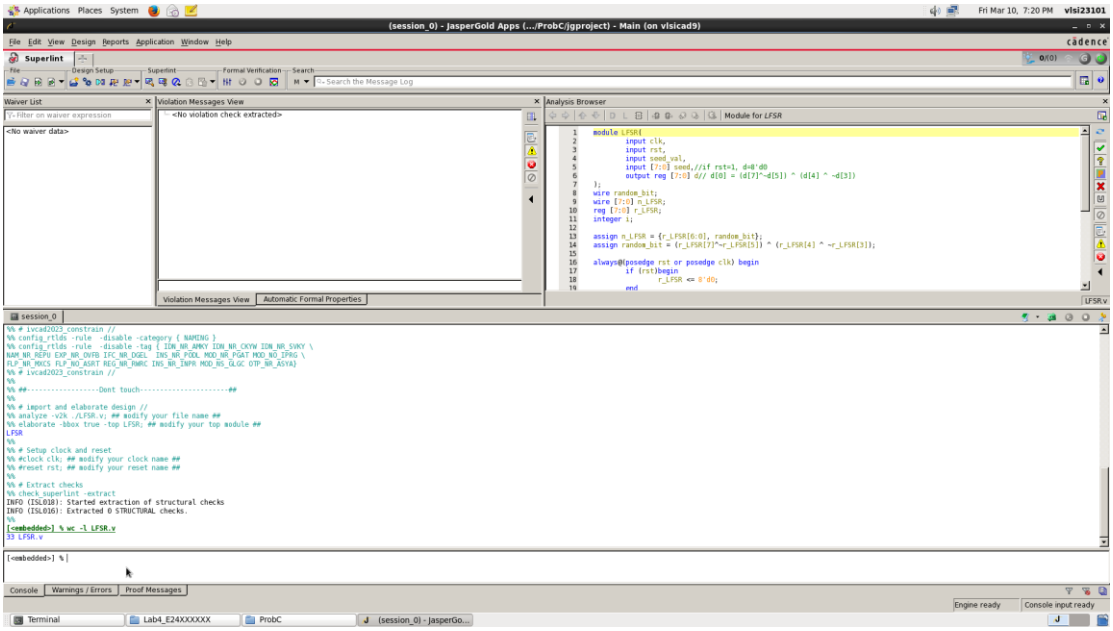


Step1：初始化 d 和 r_LFSR 為 0。
Step2：當時鐘節拍到達時，檢查 rst 是否為 1，是則將 r_LFSR 設為 0。
Step3：若種子值有效 seed_val 為 1，則使用 seed 初始化 r_LFSR。
Step4：計算 random_bit 和 n_LFSR，並將 n_LFSR 賦值給 r_LFSR。
Step5：輸出 d 為 r_LFSR 值，該值的前 8 位是當前的隨機序列值。

5. Show the simulation result on the terminal.



6. Show SuperLint coverage



coverage : 100%

7. At last, please write the lesson you learned from Lab4

組合電路與時序電路得差別與使用時機

Appendix A : Commands we will use to check your homework

Prob	Syn	command
A	pre	ncverilog register_tb.v +define+FSDB +access+r ncverilog register_tb.v +define+FSDB_ALL +access+r
B	pre	ncverilog top_tb.v +define+FSDB +access+r
C	pre	ncverilog LFSR_tb.v +define+FSDB +access+r