National Cheng Kung University

Department of Electrical Engineering

Introduction to VLSI CAD (Spring 2025)

Lab Session 4

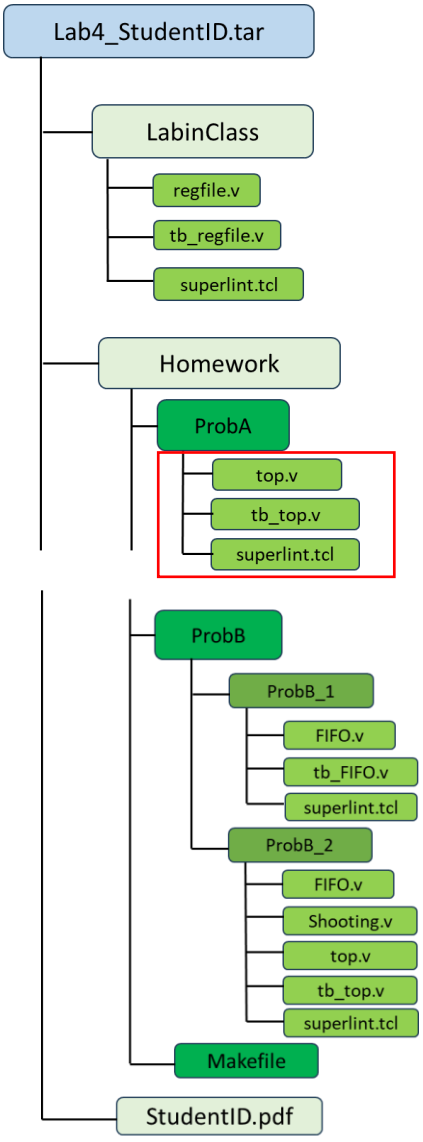
Register File and FIFO Design with Manhattan and Euclidean Distance Applications

Name	Student ID	
Practical Sections	**Points**	**Marks**
Lab in class	10	
Prob A	40	
Prob B-1	10	
Prob B-2	20	
Report	20	
Notes:		

Due Date: 15:10, March 19, 2025 @ moodle

Deliverables

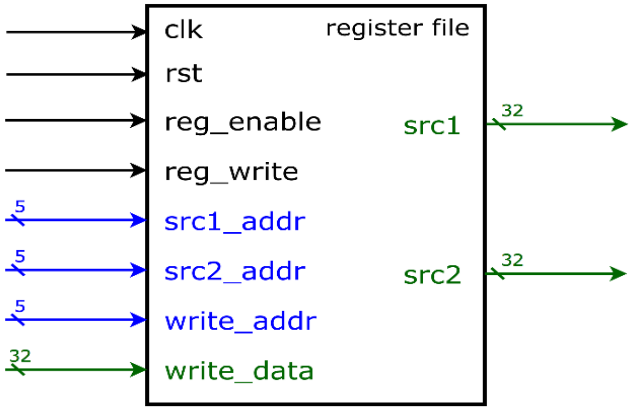
1) All Verilog codes including testbenches for each problem should be uploaded.
   NOTE: Please **DO NOT** include source code in the paper report!
2) All homework requirements should be uploaded in this file hierarchy or you will not get the full credit.
   NOTE: Please **DO NOT** upload waveforms!
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



You also need to add your additional submodule here.

Fig.1 File hierarchy for Homework submission

Lab in class: 32x32 bit Register File



1. Design a 32 x 32 bit register file with 2 output ports.
2. Port list

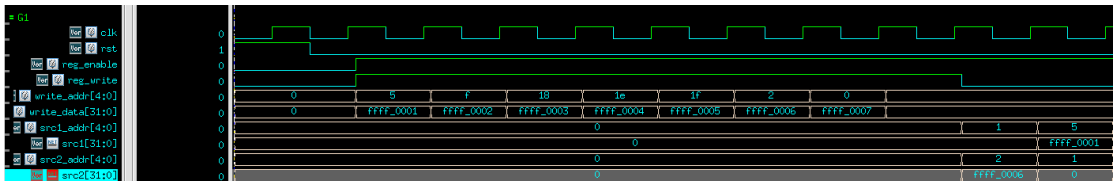
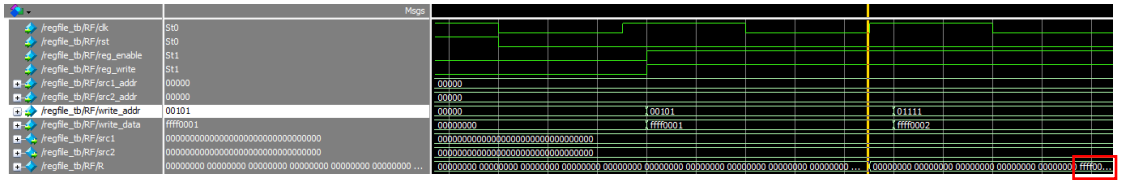
Signal	Type	Bits	Description
clk	input	1	clock
rst	input	1	reset
reg_enable	input	1	0 → off 1 → on
reg_write	input	1	0 → read 1 → write
src1_addr	input	5	source1 address
src2_addr	input	5	source2 address
write_addr	input	5	write address
write_data	input	32	write data
src1	output	32	read data source1
src2	output	32	read data source2

3.

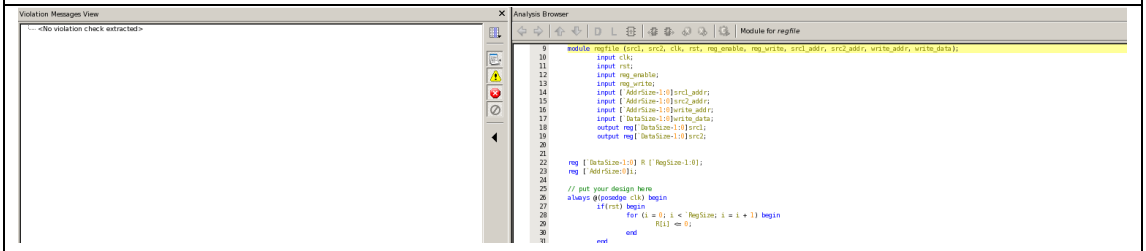
Show the simulation result on the terminal

```
**************************
**                      **       |\__||
**  Congratulations !!  **      / 0.0  |
**                      **     /_____   |
**  Simulation PASS!!   **    /^ ^ ^ \  |
**                      **   |^ ^ ^ ^ |w|
**************************    \m___m__|_|


$finish called from file "tb_regfile.v", line 208.
$finish at simulation time                33600
         V C S   S i m u l a t i o n   R e p o r t
Time: 336000 ps
CPU Time:      0.120 seconds;       Data structure size:   0.0Mb
Fri Mar 14 00:46:06 2025
CPU time: .122 seconds to compile + .139 seconds to elab + .113 seconds to link + .142 sec
```

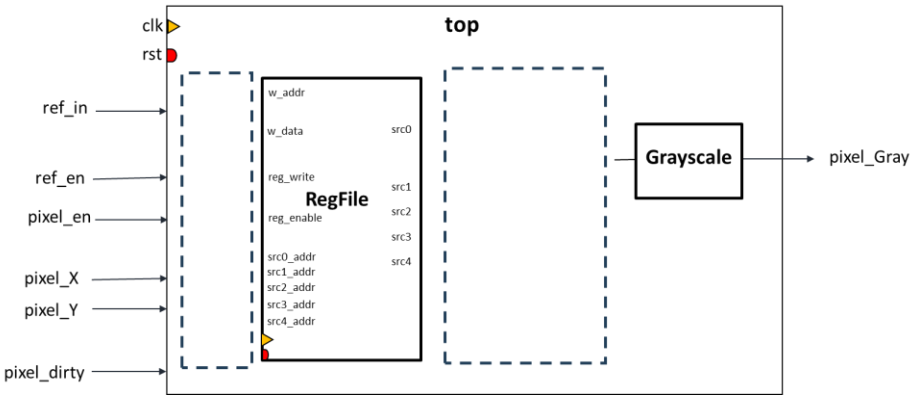
4. Show waveforms to explain that your register work correctly when read and write.

Use waveforms to explain that your register work correctly when read
 在寫入完成後，當 reg_enable = 1, Reg_write = 0 時要讀取 scr1=1 和 scr2=2 的值，因此讀取 R[1]和 R[2]的值。 R[1]因為沒有寫入，維持初始值 0， R[2]有寫入 ffff_0006, 輸出 ffff_0006。
Use waveforms to explain that your register work correctly when write
 Reg_write = 1, reg_enable = 1, write_addr = 5, write_data = ffff0001， 因此在下一個 clk, R[5]內儲存的資料從 0->ffff0001

5.

Show SuperLint coverage
 Coverage = 100%

Prob A: Image Repair



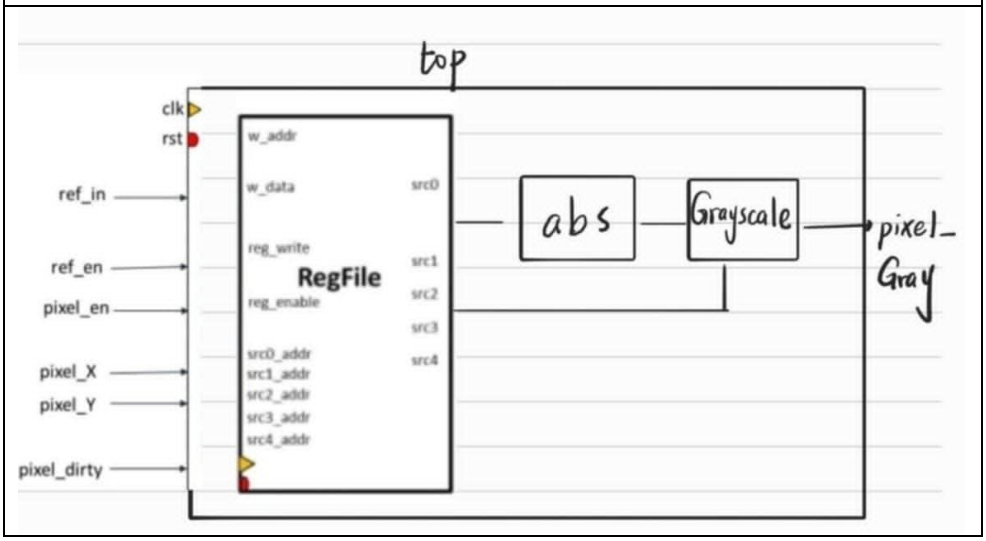
1. Design an image repair circuit in top.v, and set the ports of the top module according to the diagram below.

Signal	Type	Bits	Description
clk	input	1	Clock signal
rst	input	1	Reset, active high
ref_en	input	1	Write compared pixel enable. When ref_en is high, then ref_in is available
ref_in	input	28	Reference pixel
pixel_en	input	1	Write input pixel coordinate enable When pixel_en is high, then pixel_X, pixel_Y, pixel_dirty is available
pixel_X	input	2	x-coordinate
pixel_Y	input	2	y-coordinate
pixel_broken	input	1	1 :Pixel is broken 0: Pixel isn't broken
pixel_Gray	output	8	Grayscale output

2. Design a register file to store the RGB values of each pixel.
3. Design a Grayscale module that can convert RGB to grayscale.
4. Connect all the designed modules in top.v.

   Note: Make sure to include the used modules in top.v.

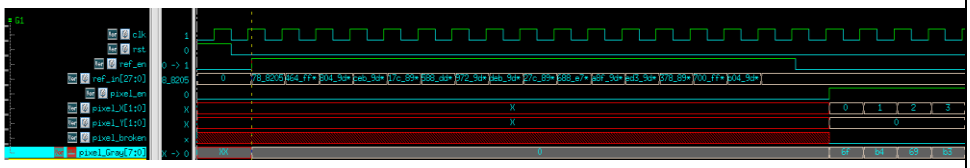
5.

Draw your design as a block diagram.


6.

Show the simulation result on the terminal

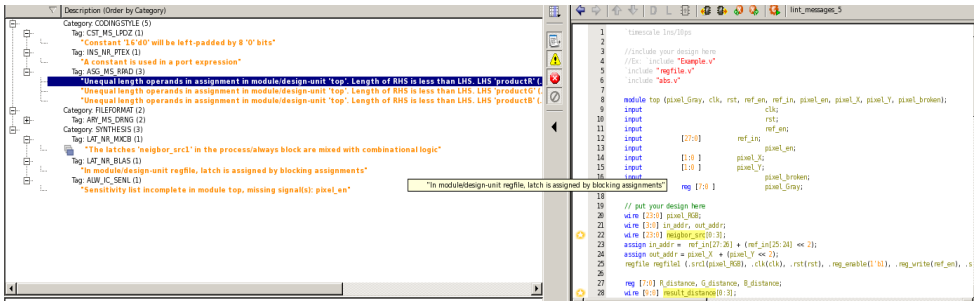

7.

Show waveforms to explain that your design work correctly.


Ref_en = 1 時 ref_in = 28’h78_8205 =
28’b0000_0111_1000_1000_0010_0000_0101
經過拆解可知 x=0, y=0, R=120, G=130, B=5。
因為(x, y)=(0, 0)時 broken 為 0，因此直接轉換成 gray_pixel
將 R*6'b001001+G*6'b010011+B*6'b000011，最後經 rounding 可得 6f

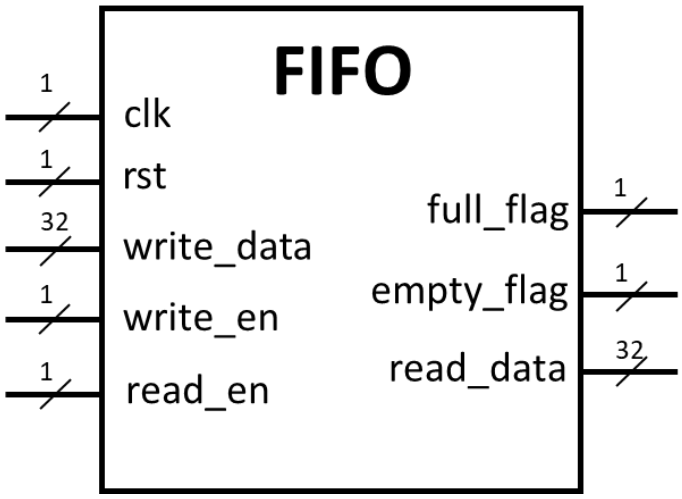
8.

Show SuperLint coverage



Coverage = 142/152 = 93%

Prob B-1: FIFO



1. Design a synchronous FIFO with a depth of 8 (able to store 8 data) and a data bit-width of 32 bits.
2. Port list

Signal	I/O	Bits	Description
clk	Input	1	Clock signal
rst	Input	1	Reset signal(active high)
write_en	Input	1	Write enable
write_data	Input	32	Data to be written into FIFO
read_en	Input	1	Read enable
read_data	Output	32	Data being read out of FIFO
full_flag	Output	1	Indicates FIFO is full or not
empty_flag	Output	1	Indicates FIFO is empty or not

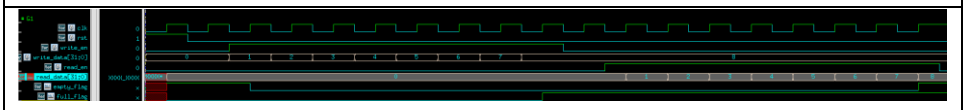
3.

Show the simulation result on the terminal

```
----------------------FIFO test--------------------------
               100  The FIFO is full. full_flag is 1.
               110  The FIFO is full. full_flag is 1.
               120  read_data 00000001 is correct
               130  read_data 00000002 is correct
               140  read_data 00000003 is correct
               150  read_data 00000004 is correct
               160  read_data 00000005 is correct
               170  read_data 00000006 is correct
               180  read_data 00000007 is correct
               190  read_data 00000008 is correct
               190  The FIFO is empty. empty_flag is 1.


      ****************************
      **                        **       |\__||
      **  Congratulations !!    **      / 0.0  |
      **                        **     /_____   |
      **  Simulation PASS!!     **    /^ ^ ^ \  |
      **                        **   |^ ^ ^ ^ |w|
      ****************************    \m___m__|_|

$finish called from file "tb_FIFO.v", line 178.
$finish at simulation time                19000
           V C S   S i m u l a t i o n   R e p o r t
Time: 190000 ps
CPU Time:      0.130 seconds;       Data structure size:   0.0Mb
Fri Mar 14 02:06:51 2025
CPU time: .125 seconds to compile + .148 seconds to elab + .116 seconds to link + .139 seconds in s
E74101034@CSH: >
```

4.

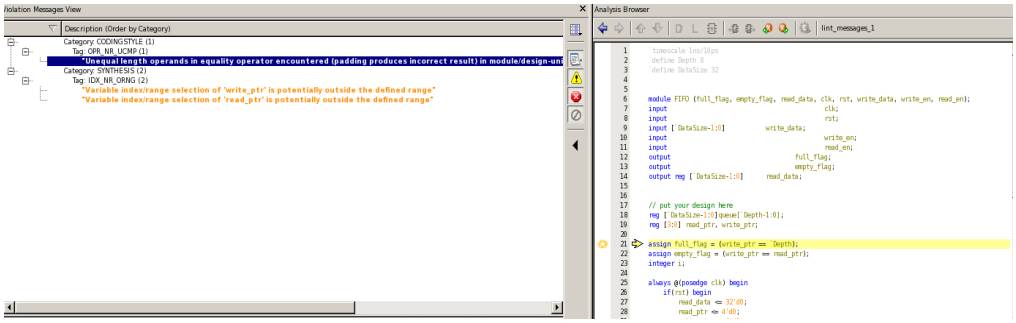
Show waveforms to explain that your design work correctly



一開始先將 input 1, 2, …7, 8 依序讀入 queue, 結束後 full 訊號升起，之後開始做 pop，依序為 1,2,…7,8，此時 empty_flag 升起。

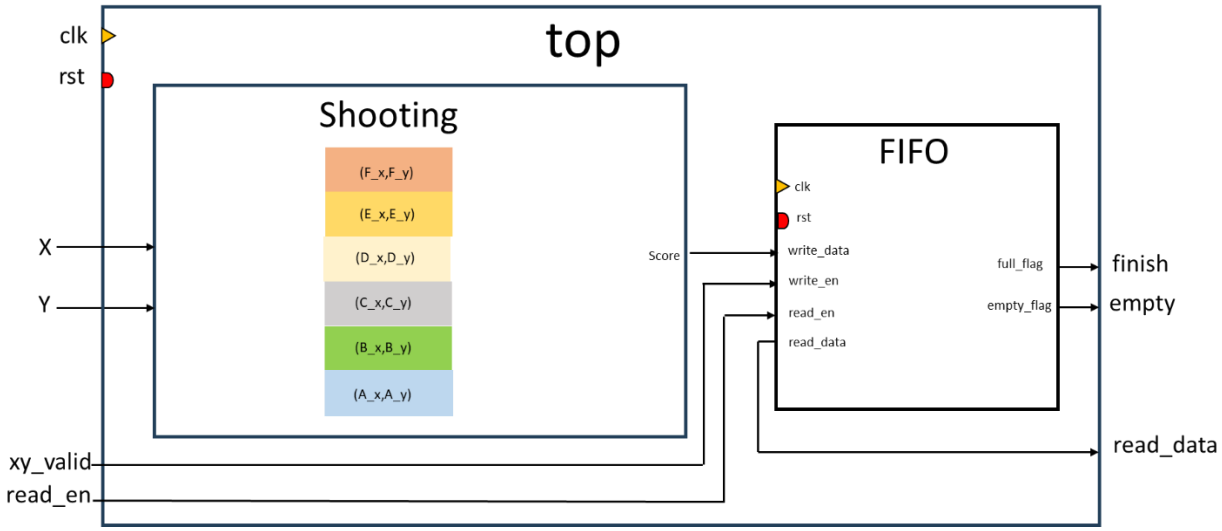
5.

Show SuperLint coverage



Coverage = 47/50 = 94%

Prob B-2: Shooting



1. Design a **Shooting module** based on the port list in the table.

Signal	IO	Bits	Description
X	Input	4	The x-coordinate of the shooting point
Y	Input	4	The y-coordinate of the shooting point
score	Output	32	Shooting point score

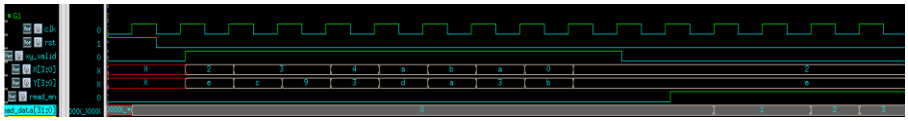
2. Connect the completed **Shooting module** and the **FIFO** from **ProbB-1** in the **top module**.
3.

Show the simulation result on the terminal

```
---------------- Second shot -------------------

FIFO[          0] is correct

FIFO[          1] is correct

FIFO[          2] is correct

FIFO[          3] is correct

FIFO[          4] is correct

FIFO[          5] is correct

FIFO[          6] is correct

FIFO[          7] is correct




        ***************************
        **                       **       |\__||
        **  Congratulations !!   **      / 0.0  |
        **                       **    /_____   |
        **  Simulation PASS!!    **   /^ ^ ^ \  |
        **                       **  |^ ^ ^ ^ |w|
        ***************************   \m___m__|_|

$finish called from file "tb_top.v", line 142.
$finish at simulation time              48100
           V C S   S i m u l a t i o n   R e p o r t
Time: 481000 ps
CPU Time:      0.120 seconds;       Data structure size:   0.0Mb
Fri Mar 14 02:12:25 2025
CPU time: .124 seconds to compile + .143 seconds to elab + .116 seconds to link + .142 sec
```

4.

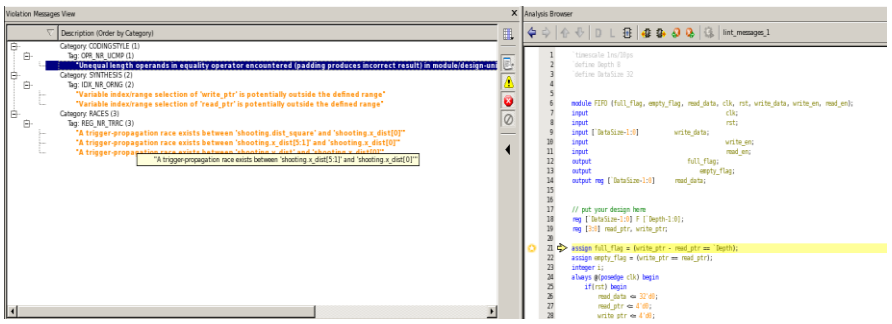
Show waveforms to explain that your design work correctly



Input1 x=2, y=14 因在 A 圈內得 1 分並寫入 queue,
Input2 x=3, y=12 因在 A 圈內得 1 分並寫入 queue,
Input2 x=3, y=9　因在 B 圈內得 2 分並寫入 queue,
依此類推，最後再將值 pop 出來。

5.

Show SuperLint coverage



Coverage = 121/127= 95%

At last, please write the lesson you learned from Lab4

之前練習 IC contest 時有寫到與圓距離相關寫法，這次練習到讓我更確切知道如何判斷點與圓的相對關係。另外，之前在寫 verilog 就有聽說過如果資料比較大，就會需要寫一個 regfile，如果不大可以直接存在陣列，原本不知道是甚麼意思，經過這次親自實作 regfile，讓我更了解這個觀念。
這次 lab 我覺得最複雜的地方是 always@(*)和 always@(posedge clk)使用，我有時候不確定某個訊號要放在哪一種 block。

Appendix A : Commands we will use to check your homework

Problem	Command
Lab in class	% vcs -R tb_regfile.v -debug_access+all -full64 +define+FSDB
ProbA	make probA
ProbB-1	make probB_1
ProbB-1	make probB_2