Due Date: 15:10, March 5, 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 Lab2 tutorial 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**

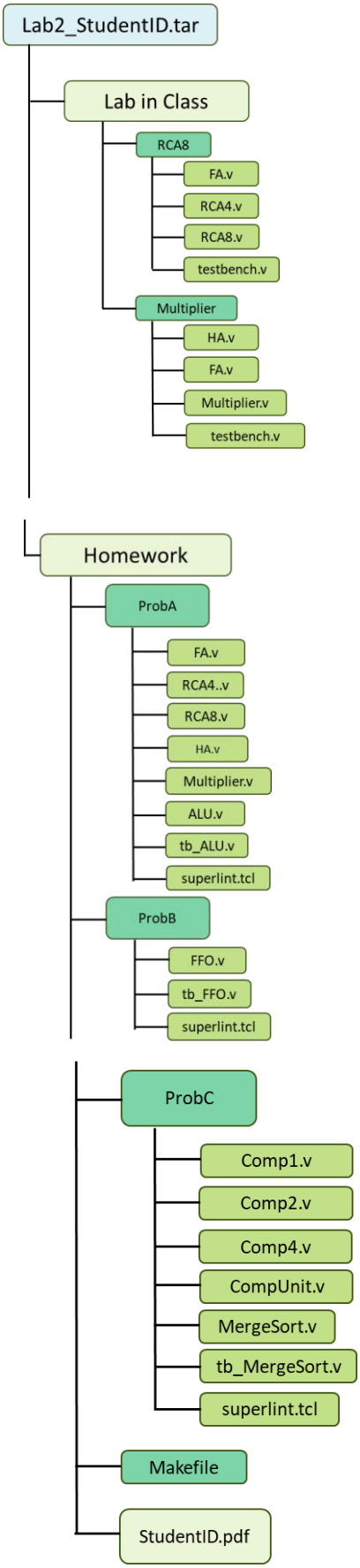


Fig.1 File hierarchy for Homework submission

Objectives:

Help students get familiar with the CAD tools (VCS & Verdi) for digital logic design. Introduce structural and hierarchical Verilog modeling and provide related practice problems. Please go through the hands-on exercise step-by-step.

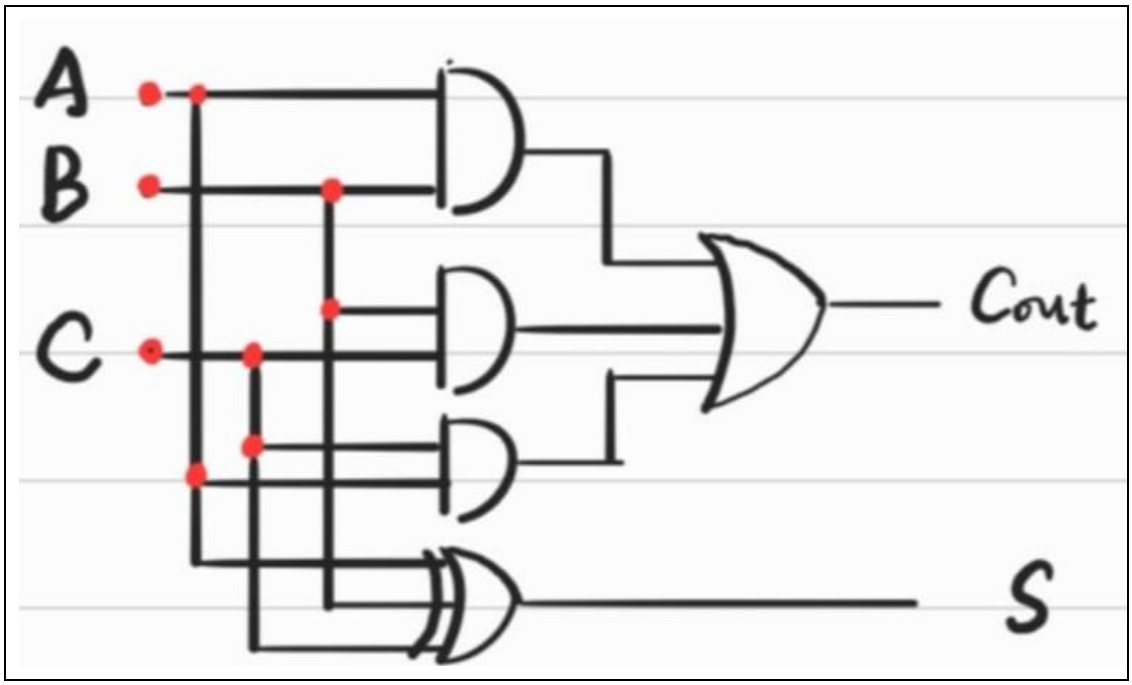
Lab in class: Design Steps & Ripple-Carry Adder & Multiplier

✧ **8-bit unsigned ripple-carry adder**

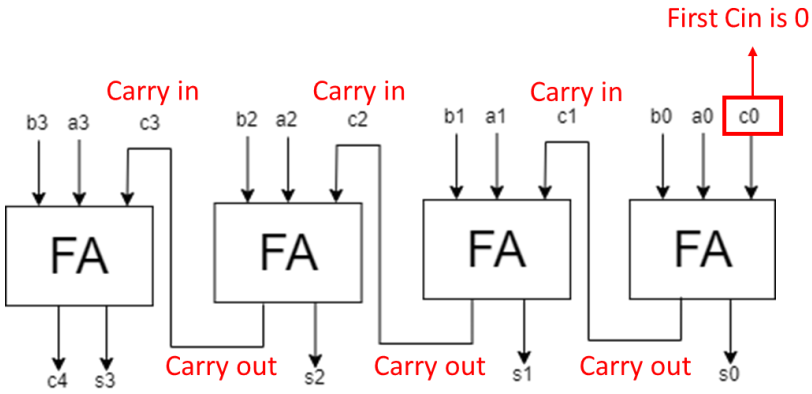
1) **An adder is a digital circuit that performs addition of number. Please design a full adder in gate level.**
2) **The truth table of full adder**

Inputs			Outputs	
A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

3) **Draw a full adder in gate level.**



4) **Design a full adder in Structural coding (The module name should be FA. And the file you include should be FA.v)**
5) **A ripple-carry adder is a basic digital circuit that performs binary addition by propagating carry bits sequentially through a chain of full adders, resulting in a relatively slow operation for large bit-widths.**
6) **Design a 4-bit unsigned ripple-carry adder in Structural coding (The module name should be RCA4. And the file you include should be RCA4.v)**



7) **Design a 8-bit unsigned ripple-carry adder in hierarchical coding using previously designed RCA4 module. (The module name should be RCA8. And the file should be RCA8.v)**

Signal	IO	Bits	Description
a	Input	8	Addend
b	Input	8	Augend
sum	Output	8	Result after calculating
overflow	Output	1	Overflow detection

8) **Simulate your design with the following test pattern in sample testbench.**
9) **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 2) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain your waveform.**

10) **You only need to upload your v-code to moodle, do not paste your code here.**

Your simulation result on the terminal.

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


$finish called from file "testbench.v", line 90.
$finish at simulation time                257000
           V C S   S i m u l a t i o n   R e p o r t
Time: 2570000 ps
CPU Time:      0.620 seconds;       Data structure size:   0.0Mb
Tue Mar  4 17:29:46 2025
CPU time: .477 seconds to compile + .575 seconds to elab + .417 seconds to link
+ .662 seconds in simulation
vlsicad6:/home/user1/vlsi25/vlsi25137/Lab2_Exxxxxxxx/Labinclass/RCA8 %
```

Your waveform :

Explanation of your waveform :

Input A = 11, B = ee, 相加得到 ff 且無 overflow，

Input A = 11, B = ff, 相加產生 overflow, Sum 值為 10。

✧ **8*8-bit unsigned multiplier**

1) **Design a half adder in Structural coding (The module name should be HA. And the file you include should be HA.v)**
2) **Design a full adder in Structural coding (The module name should be FA. And the file you include should be FA.v)**
3) **If you want to use RCA8.v from the first question to build an 8×8-bit multiplier, that is also possible. You can copy RCA4.v and RCA8.v from the first question into the folder and use them as a basis to write your program.**
4) **Design a 8*8-bit unsigned multiplier in Structural and hierarchical coding (The module name should be Multiplier. And the file should be Multiplier.v)**

Signal	IO	Bits	Description

a	Input	8	Multiplicand
b	Input	8	Multiplier
product	Output	16	Result after calculating

5) **Simulate your design with the following test pattern in sample testbench.**
6) **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 1) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain your waveform.**
7) **You only need to upload your v-code to moodle, <u>do not</u> paste your code here.**

Your simulation result on the terminal.

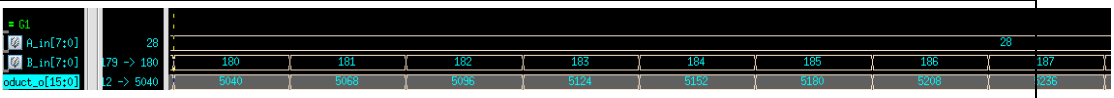
```
The correct answer is A*B = 255 * 254 = 64770
Correct! Your answer is A*B = 255 * 254 = 64770
--------------------------------------------------------------------------------

The correct answer is A*B = 255 * 255 = 65025
Correct! Your answer is A*B = 255 * 255 = 65025
--------------------------------------------------------------------------------

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


$finish called from file "testbench.v", line 76.
$finish at simulation time             32769000
           V C S   S i m u l a t i o n   R e p o r t
Time: 327690000 ps
CPU Time:      5.090 seconds;       Data structure size:   0.0Mb
Tue Mar  4 17:35:26 2025
CPU time: .511 seconds to compile + .593 seconds to elab + .419 seconds to link + 5.129 seconds in simulation
vlsicad6:/home/user1/vlsi25/vlsi25137/Lab2_Exxxxxxxx/Labinclass/Multipler %
```

Your waveform :



Explanation of your waveform :

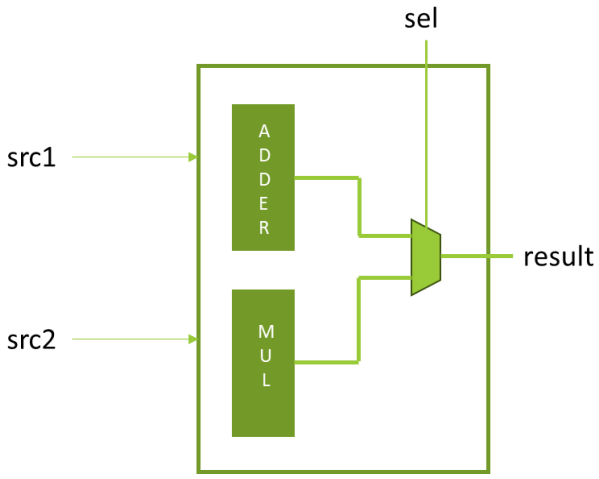
將數字換成 10 進位易觀察。

Input 1 為 28*180 = 5040，

Input 2 為 28*181 = 5068，與波形圖結果符合。

Prob A: 8-bit unsigned ALU

1) **An Arithmetic Logic Unit (ALU) is a digital circuit that performs arithmetic and logical operations as the fundamental processing unit of CPU.**
2) **Using the circuit design RCA8.v and Multiplier.v from lab-in-class to design a 8-bit unsigned ALU in Structural coding (The module name should be ALU. And the file you include should be ALU.v)**
3) **Using a select signal sel to determine whether the ALU should perform addition or multiplication. If sel is 1, the ALU performs addition; otherwise, if sel is 0, it performs multiplication.**

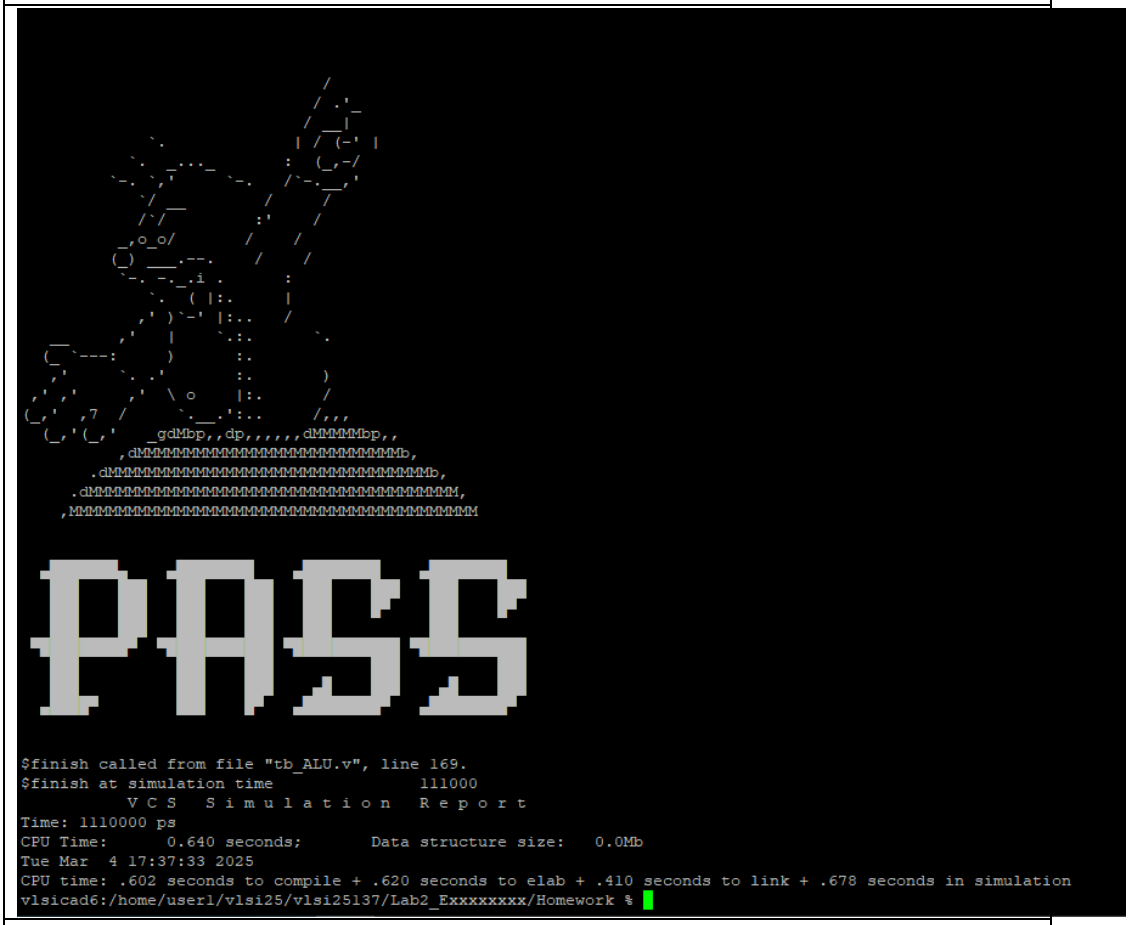


Signal	IO	Bits	Description
src1	Input	8	Source 1
src2	Input	8	Source 2
sel	Input	1	1'b1 : addition 1'b0 : multiplication
result	Output	16	ALU result

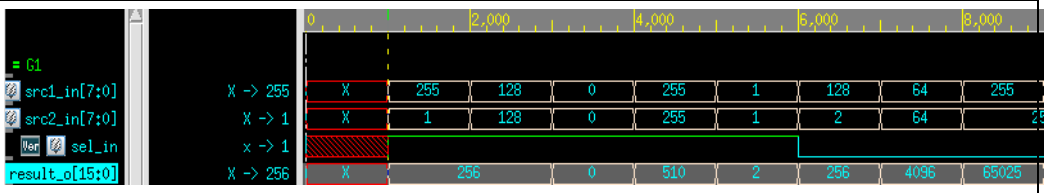
4) **Simulate your design with the following test pattern in sample testbench.**
5) **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 2) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain your waveform.**
6) **In addition, you should check your coding style, and make sure that there are no error messages and coverage with Superlint must > 90 %. Snapshot the result and *calculate Superlint coverage*.**

7) You only need to upload your v-code to moodle, <u>do not</u> paste your code here.

Your simulation result on the terminal.



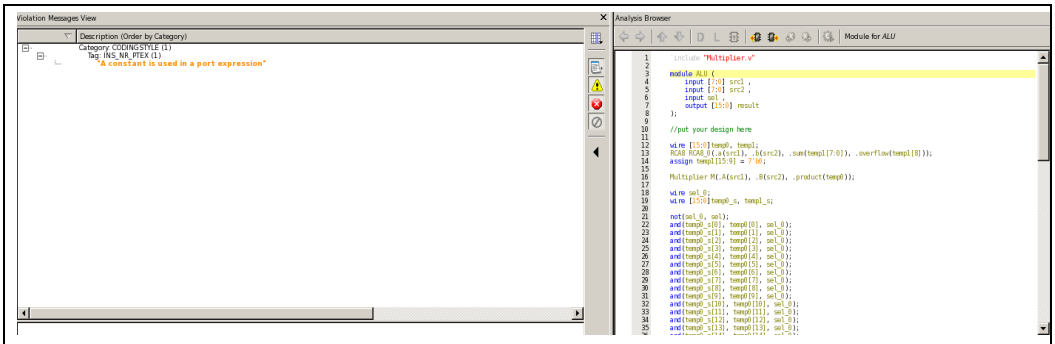
Your waveform :



Explanation of your waveform :

第一個 **input sel = 1**，因此做相加**(255+1 = 256)**,直到第六個 **input sel = 0**，因此做相乘**(128*2= 256)**

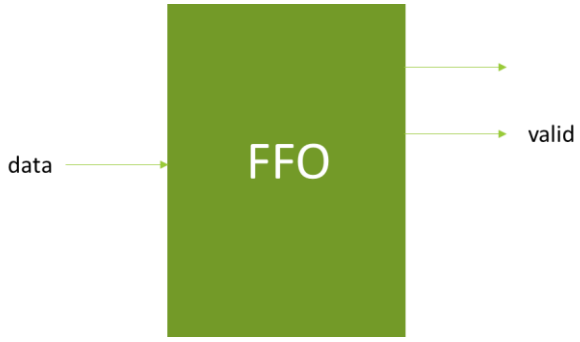
Superlint screenshot and coverage



Coverage : (280/281)*100% = 99.6%

Prob B: Find First One

1) **Design a Find First One circuit in Structural coding (The module name should be FFO. And the file you include should be FFO.v)**
2) **The function of FFO (First Find One) is to locate the first occurrence of '1' in a binary sequence, starting from the LSB. It outputs the position of the first '1' and indicates whether the position is valid. (An invalid position occurs when there are no '1's in the sequence.)**

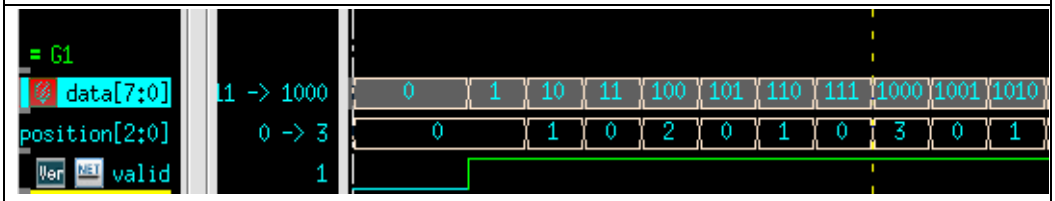


Signal	IO	Bits	Description
data	Input	8	Input sequence
position	Output	3	The position of the first occurrence of 1.
valid	Output	1	The position is valid or not.

3) **Simulate your design with the following test pattern in sample testbench.**
4) **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 1) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain your waveform.**
5) **In addition, you should check your coding style, and make sure that there are no error messages and coverage with Superlint must > 90 %. Snapshot the result and *calculate Superlint coverage*.**
6) **You only need to upload your v-code to moodle, do not paste your code here.**

Your simulation result on the terminal.

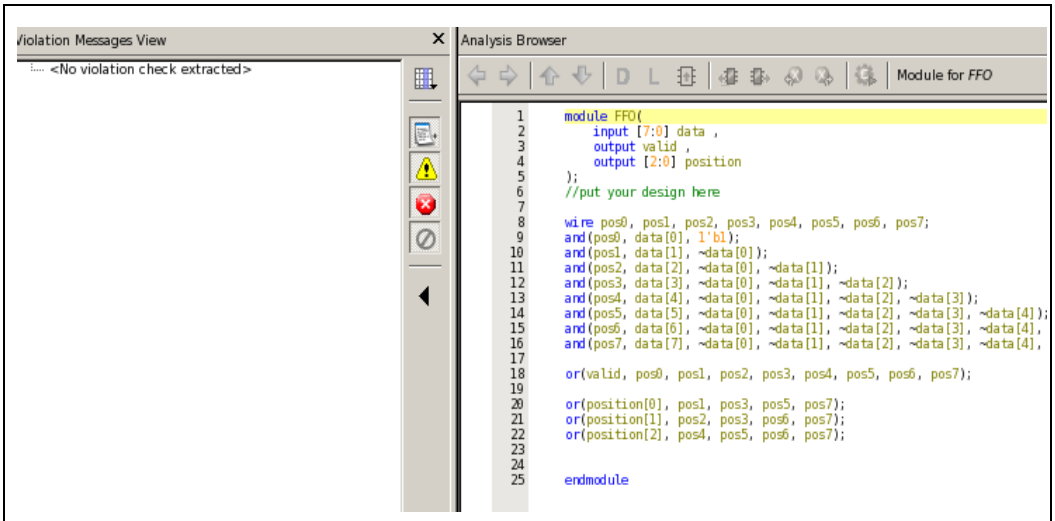
Your waveform :



Explanation of your waveform :

我將 **data** 用 **binary** 的方式顯示，較容易看出 **position** 位置。第一個 **input** 是 **0**，因此 **valid =0,** 第二個 **input** 是 **1**，因此 **valid = 1** 且 **position = 1**。第三個 **input** 是 **10**，**1** 出現在 **bit 1**，所以 **position = 1, valid = 0**。

Superlint screenshot and coverage



Coverage : 100%

Prob C: Merge Sort

1) Design a 4-bit*6 merge sort circuit in Structural coding (The module name should be MergeSort. And the file you include should be MergeSort.v)

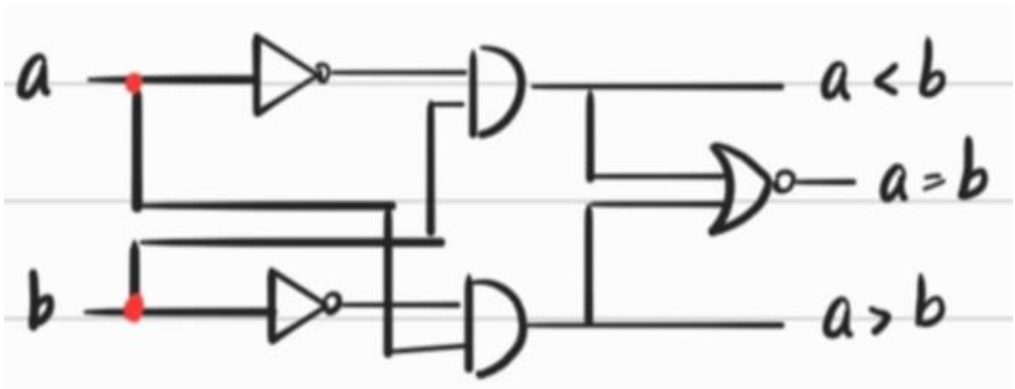
Signal	I/O	Bits	Description
in0	Input	4	Input data(unsigned)
in1	Input	4	Input data(unsigned)
in2	Input	4	Input data(unsigned)
in3	Input	4	Input data(unsigned)
in4	Input	4	Input data(unsigned)
in5	Input	4	Input data(unsigned)
out0	Output	4	Output data(biggest)
out1	Output	4	Output data
out2	Output	4	Output data

out3	Output	4	Output data
out4	Output	4	Output data
out5	Output	4	Output data(smallest)

2) Design a 1-bit comparator in gate level, and draw the 1-bit comparator in gate level. (The module name should be Comp1. And the file you include should be Comp1.v)

Input		Output		
a	b	a<b	a=b	a>b
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

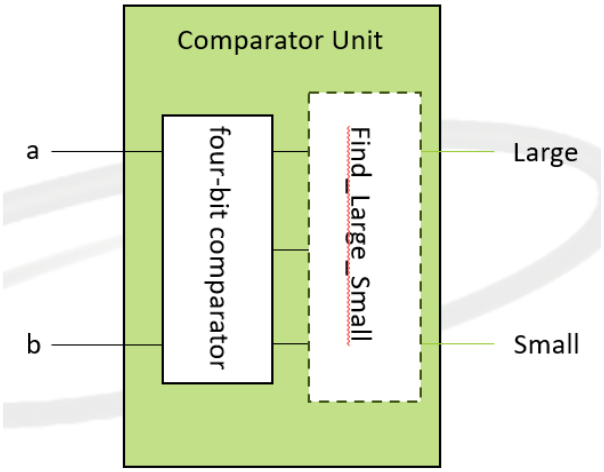
Draw your 1-bit comparator structure.



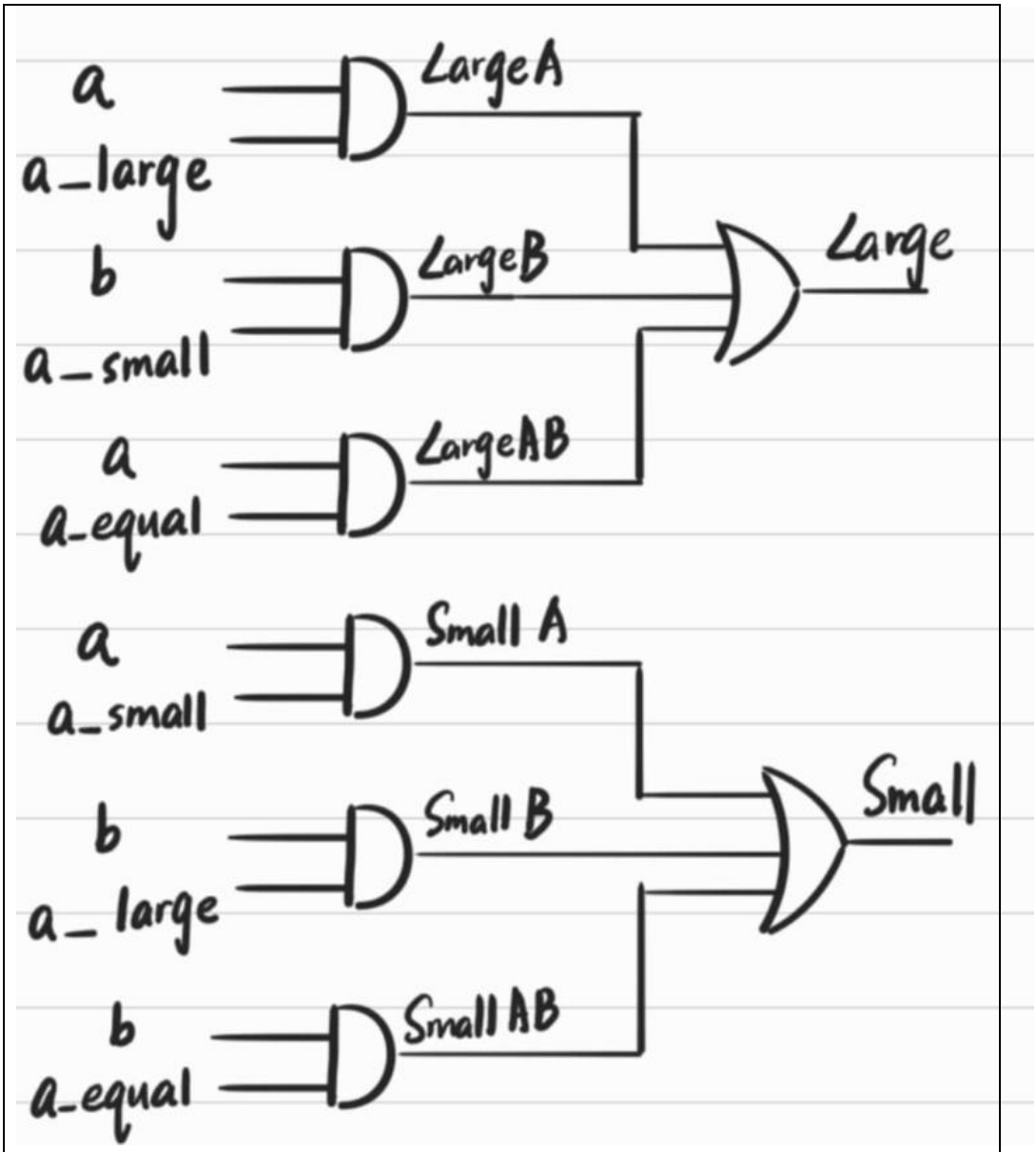
3) Design a 2-bit comparator using a 1-bit comparator. (The module name should be Comp2. And the file you include should be Comp2.v)

4) Design a 4-bit comparator using a 2-bit comparator. (The module name should be Comp4. And the file you include should be Comp4.v)

5) **Design a 4-bit comparator unit using a 4-bit comparator, and draw the Find_Large_Small block in gate level. (The module name should be CompUnit. And the file you include should be CompUnit.v)**



Draw your Find_Large_Small block in gate level.

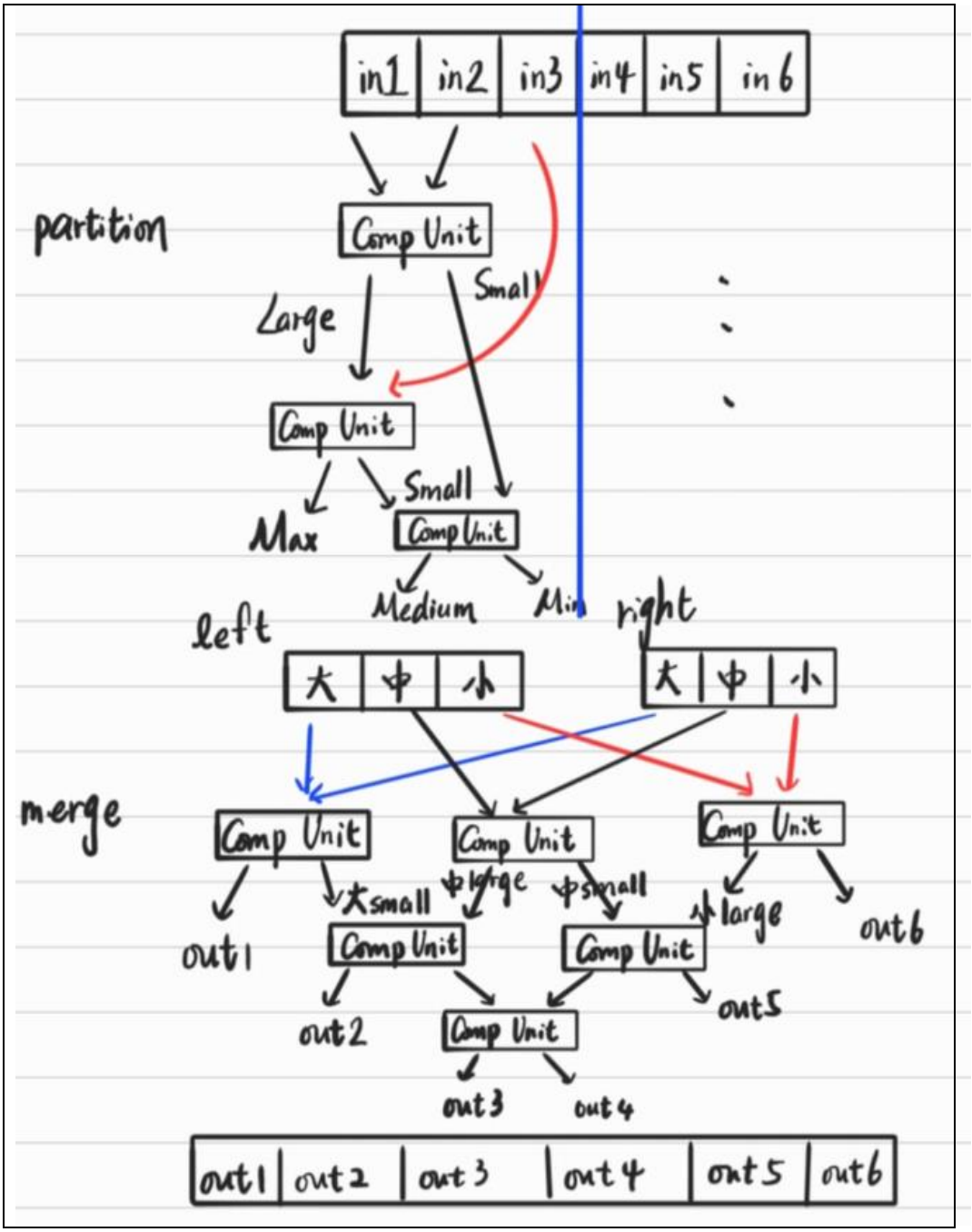


a, b, LargeA, LargeAB, LargeB, SmallA, SmallB, SmallAB 皆為 8bit，因此每個都需要 8 個邏輯閘，因空間關係沒有畫出來。

6) **Design a 4-bit*6 merge sort using a 4-bit comparator.**

7) **Draw the block diagram of the merge sort.**

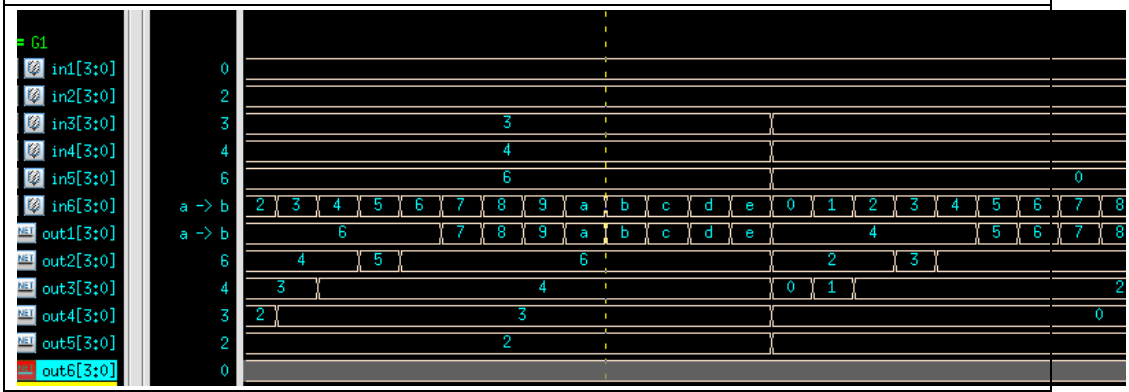
Draw your merge sort block diagram.



8) **You only need to upload your v-code to moodle, do not paste your code here.**
9) **Verify your design by comparing the simulation results with the results you predicted. If the results are not the same, please go back to 2) and revise your code. If the simulation results are correct, please snapshot the simulation result on the terminal and the waveform you dumped and explain your waveform.**
10) **In addition, you should check your coding style, there are no error messages and over 90% coverage with Superlint. Snapshot the result and *calculate Superlint coverage*.**

Your simulation result on the terminal.

Your waveform :

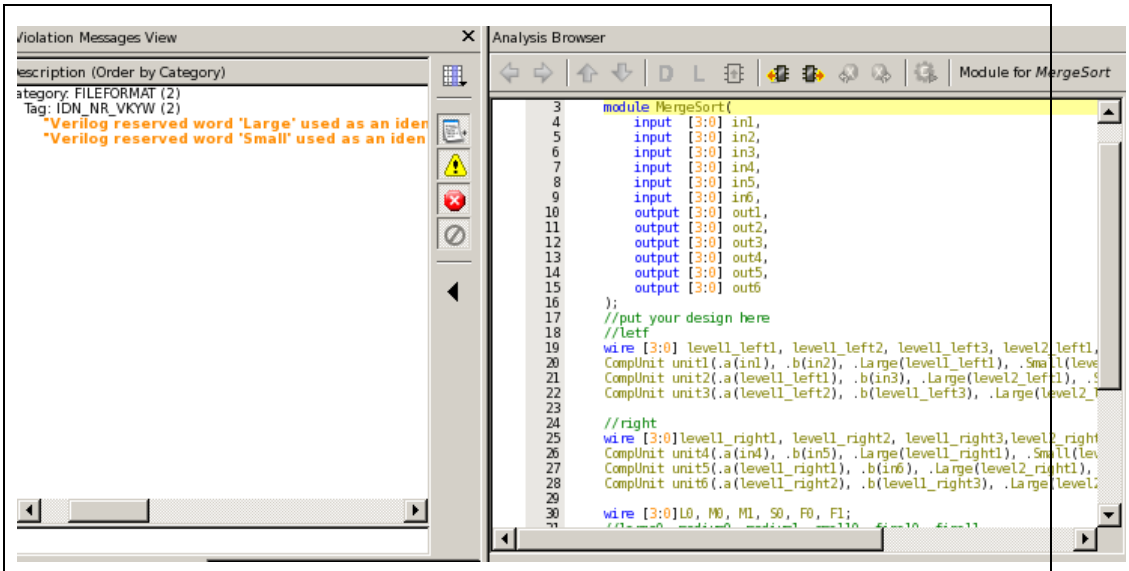


Explanation of your waveform :

第一個 **input (0, 2, 3, 4, 6, 2)**時，**ouput** 為**(6, 4, 3, 2, 2, 0)**，符合預期排序。

第二個 **input (0, 2, 3, 4, 6, 3)**時，**outpu** 為**(6, 4, 3, 3, 2, 0)**，符合預期排序。

Superlint screenshot and coverage



Coverage : 100 %

Appendix A : Commands we will use to check your homework

	Problem	Commands
Lab in class	**RCA**	**% vcs -R testbench.v -debug_access+all -full64 +define+FSDB**
	Multiplier	**% vcs -R testbench.v -debug_access+all -full64 +define+FSDB**
Homework	**ProbA**	**% make probA**
	ProbB	**% make probB**
	ProbC	**% make probC**