EEC S2070 02 Exam 1

15:30 ~ 17:30, November 3, 2020

➢ Instructions

1. There are **three (3) design problems** in this exam, with **five (5) pages** in total.
2. Download the `*.cpp` files from OJ. Change them to `*.zip` and decompress them. (Skip the `*.h` files. The OJ enforces listing a `*.h` file.)
3. Submit each of your Verilog code to OJ **immediately** after it is done.
   a. You have the responsibility to check if the submission is successful.
   b. The module names should be **exam1_A**, **exam1_B**, and **exam1_C**.
   c. The first line of each Verilog code should be a comment with your student ID and name as follows:

      ```
      // 107123456 王小明
      module exam1_A (...
      ```

   d. The **exam1_C.v** should be able to be compiled by Vivado, generating the bit file.
   e. **The submission is due at 17:30**!
4. The score will get deducted if you fail to follow the rules.
5. Hand back this problem sheet **with all the following items checked**. Also **sign your name and student ID**.

☐ I confirm that I follow the naming rule of the modules. And the first line of each answer code shows my student ID and name.

☐ I confirm that all my answers are submitted successfully.

- ✓ Submit the module **exam1_A** and the complete design for Problem A;
- ✓ Submit the module **exam1_B** and the complete design for Problem B;
- ✓ Submit the module **exam1_C** and the complete design for Problem C.

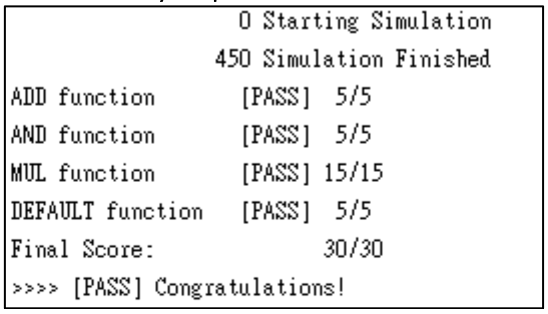
☐ I hereby state that all my answers are done on my own.

ID: ____________________ Name: ____________________

➢ Design Problems

A. [30%] [Verilog Simulation]

1. Implement an **8-bit ALU** with various computation modes.
2. You must complete the Verilog template **exam_1_A.v** with the given testbench, **exam_1_A_tb.v.** DO NOT modify the IO signals.
3. Make sure you pass the simulation with the following PASS message:



```
                    0 Starting Simulation
                  450 Simulation Finished
ADD function        [PASS]  5/5
AND function        [PASS]  5/5
MUL function        [PASS] 15/15
DEFAULT function    [PASS]  5/5
Final Score:               30/30
>>>> [PASS] Congratulations!
```

4. IO signals and description:

Signal Name	**I/O**	**Description**
X[3:0]	Input	ALU input
Y[3:0]	Input	ALU input
select[2:0]	Input	Mode selection
out[7:0]	Output	ALU result

Mode selection:

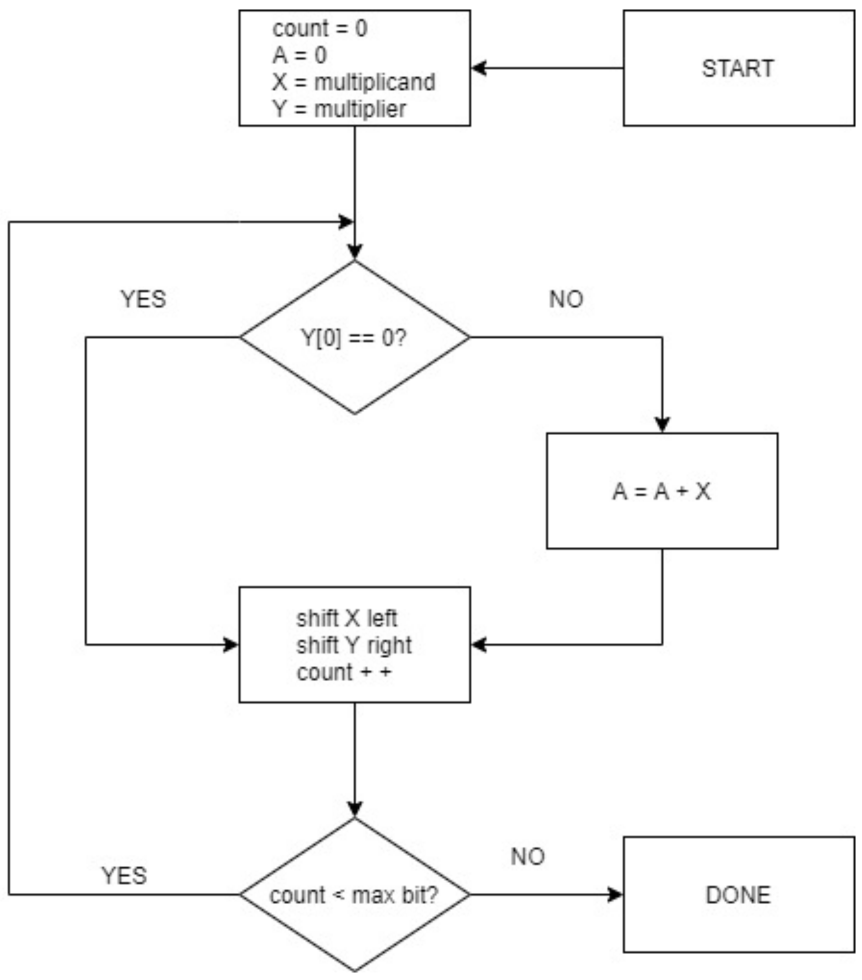
select	Function
0	out = X + Y
1	out = X & Y **with the extension of all 1's vector (see the example)**
2	out = X * Y
Otherwise	out = 5 (The **select** has 3 bits, which can support 8 different computation modes. But currently, only three modes are defined. Others are reserved with a constant output of 5.)

Examples:

(1) select=0, **X**=5 and **Y**=6: **out** is 5+6=11.
(2) select=1, **X**=4'b0110 and **Y**=4'b0111: **out** is 8'b**1111**_0110
(3) select=2, **X**=8 and **Y**=9: **out** is 8 * 9 = 72.
(4) select=3, **X**=6, and **Y**=15: **out** is 5.

5. Assume that **X, Y >= 0** for arithmetic computing. They are **unsigned integers**.
6. **Design Constraint: Using arithmetic multiply operator * is forbidden and will get a ZERO score. That is, out=X*Y is NOT allowed.** Refer to the algorithm below for the multiplication without the multiplier operator. You may use an alternative algorithm to implement the multiplication, as long as it does not use the multiplier operator.

7. The multiplication algorithm is illustrated using the flow chart and one example as follows:



Example of a 4-bit multiplication:

Multiplicand = 4’d12 = 4’b1100, Multiplier = 4’d13 = 4’b1101

Initial: A = 0000_0000, X = 0000_1100, Y = 1101

1: A = 0000_1100, X = 0001_1000, Y = 0110

2: A = 0000_1100, X = 0011_0000, Y = 0011

3: A = 0011_1100, X = 0110_0000, Y = 0001

4: A = 1001_1100, X = 1100_0000, Y = 0000

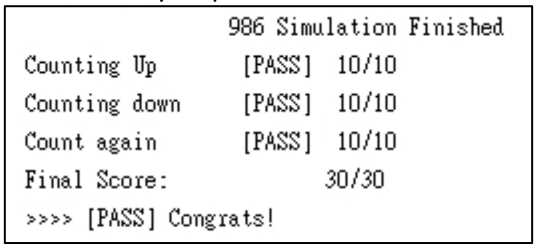
Then **the answer = A = 1001_1100 = 156 = X*Y**

A kindly reminder:

You may deal with the easiest problem first (e.g., you may deal with the multiplication after you complete other design problems to be more efficient for the exam).

B. [30%] [Verilog Simulation]

1. Implement a specific counter with the following rules.
2. You must complete the Verilog template **exam_1_B.v** with the given testbench, **exam_1_B_tb.v.** DO NOT modify the IO signals.
3. Make sure you pass the simulation with the following PASS message:



```
                        986 Simulation Finished
 Counting Up          [PASS]   10/10
 Counting down        [PASS]   10/10
 Count again          [PASS]   10/10
 Final Score:                  30/30
 >>>> [PASS] Congrats!
```

4.
5. IO signals and description:

Signal	I/O	Function
clk	Input	Clock signal (positive-edge triggered).
rst	Input	Active-high reset. You may store two values: the **current value of the counter** and the **previous value of the counter**. When being reset, let the previous value be 0 and the current value be 1.
out[9:0]	Output	The (current) value of the counter

6. The counter will count upward from 1 to 232 and then count downward to 1 repeatedly.
7. The counter will follow the rules:

   When counting upward:

   Let the sum of the current value and the previous value be *S*

   (1) When **S** is a multiple of 3 and is not a multiple of 5, the counter will increase by 2

   (2) When **S** is a multiple of 5 and is not a multiple of 3, the counter will increase by 3.

   (3) When **S** is a multiple of 3 and 5, the counter will increase by 4

   (4) When **S** is **neither** a multiple of 3 **nor** a multiple of 5, the counter will increase by 1

   When counting downward:

   (1) The counter will decrease by **i** when it is the **i**th countdown:

   X → X – 1 → (X – 1) – 2 → (X – 1 – 2) – 3 → ... → 1

8. Your output should look like this:

   1 → 2 → 4 → 6 → ... → 226 → 230 → 232 → (Upward)

   231 → 229 → 226 → ... → 22 → 1 → (Downward)

   2 → 4 → 6 → 9 → ... (Upward again)

9. Design Constraint: **You must NOT use a lookup table or a fixed, hand-coded sequence to generate the counter values.** Otherwise, your design will get a zero score.

C. [40%] [FPGA Implementation]

1. Complete the Verilog template, **exam_1_C.v**, to implement the LED controller. DO NOT modify the IO signals.
2. The LED light moving is synchronous with the clock frequency of 100MHz / 2^{25}.
3. Here is the table showing the function with the I/O connection:
   Note: The **rst**, **start**, **direction**, and **wall** are connected to the four right-most switches.

Name	**Pin**	**Description**
clk	W5	100MHz clock signal
rst	V17	Active-high reset
start	V16	To enable the movement
direction	W16	To control the moving direction when wall = 0
wall	W17	To enable the wall at the left-most and right-most sides
LED[15:0]	LEDs LD15-LD0	To connect to LD15-LD0

   a. When being reset, **the left-most LED (LD15) is ON**, and others are OFF.
   b. When **start** is 1, the light will begin to shift (to turn on and off the LEDs one by one), synchronized to the positive clock edges with the clock frequency of 100MHz / 2^{25}.
   c. When **start** is 0, hold the LEDs unchanged.
   d. Only **one single LED** is ON at one time.
   e. When **wall** = 0 and **direction** = 0, the light shifts to the right. When the light reaches the right-most LED (LD0), it will appear at the left-most LED (LD15) afterward.
   f. When **wall** = 0 and **direction** = 1, the light shifts to the left. When the light reaches the left-most LED (LD15), it will appear at the right-most LED (LD0) afterward.
   g. When **wall** = 1, **direction** makes **no effect**.
   If **wall** = 1 when being reset, the light will shift to the right. Otherwise, the light keeps its moving direction after **wall** turns to 1, no matter **direction** changes to 1 or 0 after that. If the light reaches the left-most side or right-most side, the light will **change its moving direction**. That is, it will bounce at the left-most side and the right-most side. Ex: if the light shifts to the right and reaches the right-most side (LD0), it will then move to LD1 and continue to shift to the left.
4. DO NOT modify the XDC constraint file (**exam_1_C.xdc**). Otherwise, your design will fail to test and get a ZERO score.
5. There is a clock divider in the template. You may modify it if necessary. You must include the clock divider in **exam_1_C.v**.
6. Please refer to the demo video (**exam_1_C_DEMO.mp4**) for further details.

Happy Designing!

(If you have too much time left, there is always a joke for you.)

One day, a mechanical engineer, an electrical engineer, and a computer engineer drove down the street in the same car when it broke down.
The mechanical engineer said, "I think the engine broke. We have to fix it."
The electrical engineer said, "I think there was a spark and something's wrong with the electrical system. We have to fix it."
Both of them turned to the computer engineer and asked, "What do you think?"
The computer engineer replied, "I have no idea what happened. But why don't we close all the windows, and then open the windows again? That always works for me!!"