EECS2070 02 Exam 2

15:30 ~ 17:30, December 6, 2022

➢ Instructions

1. There are **two (2) design problems** in this exam, with the PDF specification of **four (4) pages** in total.
   - You also have the hardcopy of the first page to sign and hand back.
2. Download the ***.cpp** files from OJ. Change them to ***.zip** and decompress them. (Skip the *.h file, which the OJ system enforces to have.)
3. Submit each 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 **exam2_A** and **exam2_B**.
   c. The first line of each Verilog code should be a comment with your student ID and name as follows:

```
// 110123456 王小明
module exam2_A (...
```

   d. The **exam2_A.v** and **exam2_B.v** should be able to be compiled by Vivado, generating the bit file.
   e. **The submission is due at 17:30**!
4. Please **take the OJ password slip with you** when leaving your seat. Do not litter!
5. The score will get deducted if you fail to follow the rules.
6. Hand back this problem sheet **with all the following items checked**. Also, **sign your name with your student ID**.

☐ I confirm that I read the instructions carefully and understand that my score will get deducted if failing to follow the rules.

☐ 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, if finished, are submitted successfully.
- ✓ Submit the module **exam2_A** and the complete design in a single file.
- ✓ Submit the module **exam2_B** and the complete design in a single file.
- ✓ I understand that the generated bit file will be scored. And the incorrect submission will result in a zero score.

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

ID: _____ Name: _____

➢ Design Problems

A. (25%)

1. **This design problem needs the FPGA board.**
2. Use the Verilog template **exam2_A.v** to implement an LED controller. DO NOT modify I/O signals. (You can define I/O signals as either reg or wire.)
3. I/O signal specification:

Name	**I/O**	**Pin**	**Description**
clk	Input	W5	100MHz clock signal
btnC	Input	U18	Reverse the state of all LEDs
btnU	Input	T18	Light up all LEDs
btnR	Input	T17	Right shift all LEDs
sw [15:0]	Input	16-switchs	Control each LEDs
DIGIT [3:0]	Output	4-digits	Display number of light LEDs
DISPLAY [6:0]	Output	7-segment	Display number of light LEDs
led [15:0]	Output	LEDs LD15-LD0	

4. Function description
   (1) Each switch controls the corresponding LED. Set sw[i] on to light up led[i].
   (2) Pressing btnC to reverse the state of all LEDs.
   (3) Pressing btnU to light up all LEDs.
   (4) Pressing btnR to shift the state of all LEDs to the right. The rightmost LED reappears on the left.
   (5) Releasing a button make LEDs return to their original state.
   (6) The seven-segment display shows the number of lighted LEDs. "0000" initially, and "0005" if 5 LEDs are lit.
5. Please refer to the demo video (**exam2_A_DEMO.mp4**) for further details.
6. Refer to the XDC constraint file (**exam2_A.xdc**) for the pin connection. Also, DO NOT modify the constraint file. Otherwise, your design will fail to test and get a ZERO score.
7. There are already several modules in the template, including the clock divider, debounce, and seven-segments modules. You can modify them if necessary. However, the submitted file must be able to generate the bit file, otherwise you will get 0 point in exam2_A.
8. Grading

Function	Score
switchs	5%
btnC	5%
btnU	5%
btnR	5%
Seven segment display	5%

B. (75%)

1. **This design problem needs the FPGA board and keyboard (number keys 0~9).**
2. Use the Verilog template **exam2_B.v** to implement an electric lock, which allows you to set and guess the password. DO NOT modify I/O signals.
   (You can define I/O signals as either reg or wire.)
3. I/O signal specification:

Name	I/O	Pin	Description
clk	Input	W5	100MHz clock signal
rst	Input	U18 // btnC	Reset
en	Input	T17 // btnR	Control the states
PS2_DATA	Inout	B17	Signal used for the keyboard
PS2_CLK	Inout	C17	Signal used for the keyboard
DIGIT [3:0]	Output	4-digits	Display password
DISPLAY [6:0]	Output	7-segment	Display password
led [15:0]	Output	LEDs LD15-LD0	Display state

4. Your design should be reset **asynchronously**.
5. Your design should change its value at the **positive edge** of each clock cycle. Except for the reset.
6. Function description
   (1) There are four states: **INIT, SET, GUESS,** and **CHECK.** In the **SET** state, you can set a 4-digit password for the lock. In the **GUESS** state, you can guess the password. In the **CHECK** state, the module will inform you of the correctness of your guess (whether the password you guessed is equal to the one you set or not).
   (2) LEDs are used to show the current state.
   (3) Keyboard is used to set a password and guess the password.
   (4) The seven-segment display is used to show the password in the **SET** and **GUESS** states and show the correctness in the **CHECK** state.

   ➢ **INIT:**
   - After being reset, the state should be reset to the **INIT** state; and the set password and guessed password should be reset to "0000".
   - led[12] ~ led[15] should be on; others are off.
   - The seven-segment display should show "- - - -".
   - Press the **en** button to go to the **SET** state.

   ➢ **SET:**
   - You can set the 4-digit password in this state. The password is initially "0000".
   - Pressing a number key on the keyboard will shift the password left, and the rightmost digit becomes the pressed number. E.g., if the original password is "0000", pressing the number key 1 changes the password to "0001", and then pressing the number key 5 changes the password to "0015".
   - led[8] ~ led[11] should be on, others are off.
   - The seven-segment display should show the set password.
   - Press the **en** button to finish the setting and go to the **GUESS** state.

   ➢ **GUESS:**
   - You can guess a 4-digit password in this state. The initial value of your guessed password is different in two cases:
     Case1. Enter **GUESS** state from **SET** state. Initial value: "0000".
     Case2. Enter **GUESS** state from **CHECK** state. Initial value: your last guessed password.
   - Use the keyboard to guess the password in the same way as setting the password in the **SET** state.
   - led[4] ~ led[7] should be on; others are off.
   - The seven-segment display should show the guessed password.

- Press the **en** button to finish the guessing and go to the **CHECK** state.

➢ **CHECK:**

- In this state, the lock will check the correctness of your guess.
- led[0] ~ led[3] should be on; others are off.
- The seven-segment display should show “1111” or “0000”, and flash at a frequency of 100MHz/2^{25}: flashing with “1111” if the guessed password is equal to the set password; flashing with “0000” if not equal.
- After 5 flashing (dark-light-dark-light-dark), if the set password and the guessed password are matched, go to the **INIT** state and reset the two passwords to “0000”; if they are NOT the same, go back to the **GUESS** state with the **two passwords remaining unchanged**. Then you can make another guess based on your previous guess.

7. Please refer to the demo video (**exam2_B _DEMO.mp4**) for further details.
8. Refer to the XDC constraint file (**exam2_2.xdc**) for the pin connection. Also, DO NOT modify the constraint file. Otherwise, your design will fail to test and then get a ZERO score.
9. There are already several modules in the template, including the clock divider, debounce, one-pulse, seven-segments, and KeyboardDecoder modules. You can modify them if necessary. However, the submitted file must be able to generate the bit file, otherwise you will get 0 point in exam2_B.
10. Grading

Function	Score
INIT state	13%
SET state	22%
GUESS state	20%
CHECK state	20%

Happy Designing and Good luck!