Lab 5: 1A2B Game Console

Submission Due Dates:

Demo:	2022/11/01 17:20
Source Code:	2022/11/01 18:30
Report:	2022/11/06 23:59

Objective

1 Getting familiar with modeling finite state machines (FSMs) in Verilog.
2 Getting familiar with the FPGA design flow using the demo board.

Description

The 1A2B game rules are listed as follows:

The opponent sets the secret number first, and then the player tries to guess it. The secret number should consist of **four distinct digits**. If the matching digits are in their correct positions, they are "As"; those in incorrect positions are "Bs".

Example:

Secret number: 1234
The player's guess: 4271
→ Answer: 1A2B (The A is "2"; the Bs are "4" and "1".)

- The detailed specification:
  - ✓ Any unspecified inputs (e.g., buttons and switches) should do nothing.
  - ✓ Any unspecified outputs (e.g., the LEDs and 7-segment display) should be turned off.
  - ✓ The FSM may consist of the following states. You can design your own variant instead.

1 IDLE state:

a. After being reset, go to the IDLE state.

b. The 7-segment display will show "----".

c. LEDs light up as follows:

●●●●○○○○○○○○○○○○

(●: LED on, ○: LED off)

d. When the **OK** button (BTNR) is pressed, go to the SET_ANSWER state.

2 SET_ANSWER state:

a. The 7-segment shows “**0000**”, initially.

b. The opponent sets the digit from left to right. When the **minus (-)** button (BTND) (or the **plus (+)** button (BTNU)) is pressed, the value of the adjusted digit will decrease (or increase) by one. After finishing the adjustment, press **OK** to proceed with the next digit.

**Note: assume the four digits are distinct. You don’t need to deal with duplicated numbers.

c. The minimum value of each digit is zero; the maximum value is nine. For example, “9” will still remain “9” after pressing the **plus** button and vice versa.

d. LEDs will light up from left to right when setting each digit.

(Setting the leftmost digit) : ○○○○●○○○○○○○○○○○

↓ : ○○○○○●○○○○○○○○○○

↓ : ○○○○○○●○○○○○○○○○

(Setting the rightmost digit) : ○○○○○○○●○○○○○○○○

e. After adjusting the final (rightmost) digit, press **OK** to go to the GUESS state.

f. When the **cancel** button (BTNL) is pressed, go to the IDLE state.

3 GUESS state:

a. The 7-segment shows “**0000**”, initially.

b. The player adjusts the digit from left to right. When the **minus** (or **plus**) button is pressed, the value of the adjusted digit will decrease (or increase) by one. After finishing the adjustment, press **OK** to proceed with the next digit.

**Note: assume the four digits are distinct. You don’t need to deal with duplicated numbers.

c. The minimum value of each digit is zero; the maximum value is nine. For example, “9” will still remain “9” after pressing the **plus** button and vice versa.

d. LEDs will light up from left to right when guessing each digit.

(Guessing the leftmost digit) : ○○○○○○○○●○○○○○○○

↓ : ○○○○○○○○○●○○○○○○

↓ : ○○○○○○○○○○●○○○○○

(Guessing the rightmost digit) : ○○○○○○○○○○○●○○○○

e. After adjusting the final (rightmost) digit, press **OK** to go to the next state. If the player makes a correct guess, go to the CORRECT state. Otherwise, go to the WRONG state.

f. When **cancel** is pressed, go to the IDLE state.

4 WRONG state:

a. The 7-segment will show the result (i.e., “**?A?b**”).

b. LEDs will light up as follows:

○○○○○○○○○○○○●●●●

c. When **OK** is pressed, go to the GUESS state to guess again.

d. When **cancel** is pressed, go to the IDLE state.

5 CORRECT state:

a. The 7-segment will show “**4A0b**”.

b. All LEDs will be flashing for 5 seconds. (light up first for one second then off, and you may use one cycle of clk/(2^27) as 1 second).

c. The machine goes to the IDLE state after 5 seconds.

6 Demo video:

https://youtu.be/bWnB0hayADQ

I/O signal specification

- ***clk***: clock signal with the frequency of 100MHz (connected to pin **W5**).
- ***rst***: asynchronous active-high reset (connected to **BTNC**).
- ***BTNR*** (OK), ***BTNU*** (+), ***BTND*** (-), ***BTNL*** (cancel): pushbuttons.
  - ✧ Signals from these pushbuttons should be processed by debouncing and one-pulse converters properly.
- ***LED[15:0]***: signals to control LEDs (connected to **LD15 ~ LD0**).
- ***DIGIT[3:0]***: signals to enable one of the 7-segment digits.
- ***DISPLAY[6:0]***: signals to control the digits on the 7-segment display.

Questions and Discussion

Please answer the following questions in your report.

A. Please explain how your design identifies the correct answer set in the SET_ANSWER state when the machine is in the other state.
B. Please explain how your design calculates the number of A and B.
C. Please explain how to control LEDs to blink at a frequency of 1 second in the CORRECT state.

Guidelines for the report

Refer to the guidelines in the report template (or in the previous lab assignments).

Grading policy (subject to change): Part (A): **35%**; Part (B): **50%**; Part (C): **10%**; (D): **5%**

Hint

1 You must design at least one finite state machine (FSM). There should be at least five states. More states or multiple FSMs are acceptable. But remember to explain your design in the report.

2 You have to use the following template for your design:

```
module lab5 (
  input wire clk,
  input wire rst,
  input wire BTNR,
  input wire BTNU,
  input wire BTND,
  input wire BTNL,
  output reg [15:0] LED,
            output reg [3:0] DIGIT,
            output reg [6:0] DISPLAY
        );
          /* Note that output ports can be either reg or wire.
          * It depends on how you design your module. */
          // add your design here
        endmodule
```

Attention

- ✓ DO NOT include the clock_divider, debounce and one-pulse modules in the file you hand in. Please do not integrate them into lab5.v
- ✓ If you have two or more modules used for any specific lab, merge them into one Verilog file before the submission.
- ✓ You should submit one source files, **lab5.v**. DO NOT hand in any compressed ZIP files, which will be considered an incorrect format.
- ✓ You should also hand in your report as **lab5_report_StudentID.pdf** (e.g., lab5_report_110062666.pdf).
- ✓ You should be able to answer the questions for this lab from TA during the demo.
- ✓ You need to prepare the bitstream files before the lab demo to make the demo process smooth.
- ✓ Feel free to ask questions about the specification on the EECLASS forum.