# EECS2070 02 Exam 2

**15:20 ~ 17:50, December 15, 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 **exam2_A**, **exam2_B**, and **exam2_C**.
   c. The first line of each Verilog code should be a comment with your student ID and name as follows:

```
// 108123456 王小明
module exam2_A (...
```

   d. Submit one integrated Verilog file including all modules (clock_divider, debounce, onepulse, SevenSegment, KeyboardDecoder, KeyboardDecoder, everything!) for each design problem, i.e., **exam2_A.v, exam2_B.v or exam2_C.v**. DO NOT submit separate files for one single problem.
   e. The **exam2_A.v**, **exam2_B.v**, and **exam2_C.v** should be able to be compiled by Vivado, generating the bit file.
   f. **The submission is due at 17:50**!
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 of my answers are submitted successfully.

✓ Submit the module **exam2_A** and the complete design for Problem A;

✓ Submit the module **exam2_B** and the complete design for Problem B;

✓ Submit the module **exam2_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. (20%)

1、**This design problem needs the FPGA board.**

2、Design the Verilog module, **exam2_A**, that models a counter that its counting direction can be changed. It behaves as follows:

- Precision is **0.1sec** and the counting range is from **0.00.0 to1.00.0** (60sec).
- Count the time (**0.1sec**) with the frequency of **clk / (2^23)** and show on the 7-segment display.
- In the beginning, press **en_btn** button to start **counting up**. Press **en_btn** button again to stop counting.
- When counting, press **dir_btn** button to change the counting direction.
- When the counter **counts up** to **1.00.0**, stop counting. However, you can change the direction by pressing **dir_btn** button once and start **counting down** immediately.
- When the counter **counts down** to **0.00.0**, stop counting. However, you can change direction by pressing **dir_btn** button once and start **counting up** immediately.

**3、I/O signal specification:**

| | |
|---|---|
| clk | connected to W5 |
| rst | connected to BTND (U17) |
| en_btn | connected to BTNL (W19) |
| dir_btn | connected to BTNU (T18) |
| DISPLAY | signal used for the 7-segment display. |
| DIGIT | signal used for the 7-segment display. |

**4、**You have to use the given template for your **exam2_A.v**. Do not change the input and output signals.

```
// 108123456 王小明
module exam2_A(clk, rst, en_btn, dir_btn, DIGIT, DISPLAY);
    input clk, rst, en_btn, dir_btn;
    output [3:0] DIGIT;
    output [6:0] DISPLAY;

    // add your design here

endmodule
```

## B. (50%)

1. **This design problem needs the FPGA board and keyboard (number keys 0~3).**
2. Design the Verilog module, **exam2_B**, that models a controller with the keyboard. It behaves as follows:

   a. **INITIAL:**

   - After the reset, the 7-segment display shows "0000". All the LEDs are turned off initially. After pressing **control_btn**, change to the **CONTROL** mode.

   b. **CONTROL:**

   - In the CONTROL mode, the user can use the keyboard to turn LEDs on or off.
   - The LEDs are on when the specific key is **pressed**. The LEDs are off when the key is **released**. The following table specifies the correspondence between the numerical keys and the LEDs:

| Keyboard | LEDs |
|---|---|
| Key 0 | LED[0:3] (U16~V19) |
| Key 1 | LED[4:7] (W18~V14) |
| Key 2 | LED[8:11] (V13~U3) |
| Key 3 | LED[12:15] (P3~L1) |

   - The counter starts from **0.00.0**, counting to **1.00.0** (60 sec) with an increment of **0.1 sec**.
   - Show the counter output on the 7-segment display. Use the frequency of **clk / (2^23)** for the increment of **0.1 sec**.
   - Return to the **INITIAL** mode when the user presses the **finish_btn** button or after the counter reaches **1.00.0**.

3. **I/O signal specification:**

| | |
|---|---|
| clk | connected to W5 |
| rst | connected to BTND (U17) |
| control_btn | connected to BTNL (W19) |
| finish_btn | connected to BTNU (T18) |
| PS2_CLK | signal used for the keyboard |

| | |
|---|---|
| PS2_DATA | signal used for the keyboard |
| DISPLAY | signal used for the 7-segment display. |
| DIGIT | signal used for the 7-segment display. |

4. You have to use the given template for your **exam2_B.v**. Do not change the input and output signals.

```
// 108123456 王小明
module exam2_B(
      output wire [6:0] DISPLAY,
      output wire [3:0] DIGIT,
      output wire [15:0] led,
      inout wire PS2_DATA,
      inout wire PS2_CLK,
      input wire rst,
      input wire clk,
      input wire control_btn,
      input wire finish_btn
      );
      parameter [8:0] KEY_CODES [0:3] = {
            9'b0_0111_0000, // right_0 => 70
            9'b0_0110_1001, // right_1 => 69
            9'b0_0111_0010, // right_2 => 72
            9'b0_0111_1010 // right_3 => 7A
      };
    // add your design here
endmodule
```

## C. (30%)

1. **This design problem needs both the FPGA board and keyboard (only Key 0).**
2. Design the Verilog module, **exam2_C**, similar to **exam2_B**, but with extended functionality. It behaves as follows:
   a. **INITIAL:**
      - After being reset, the 7-segment display shows “0000”. All the LEDs are turned off initially. After pressing **record_btn**, change to the **RECORD** mode.

   **b. RECORD:**
      - The counter starts from **0.00.0**, counting to **1.00.0** (60 sec) with an increment of **0.1 sec**.
      - Show the counter output on the 7-segment display. Use the frequency of **clk / (2^23)** for the increment of **0.1 sec**.
      - Control all the LEDs (U16~L1) by Key 0 similar to the CONTROL mode in

Problem B .

- In the **RECORD** mode, you should record how the LEDs are on and off. That is, you should record **the time Key 0 being pressed** and **the time Key 0 being released**.
- Keep recording until the **show_btn** is pressed or the counter reaches **1.00.0**. Then go to the **SHOW** mode

c. **SHOW:**

- Replay how the LEDs were on and off in the **RECORD** mode
- For example, in the **RECORD** mode, if Key 0 was pressed at **2.5 sec** and released at **5.0 sec**, the LEDs were on **between 2.5 sec and 5.0 sec**; in the SHOW mode, the LEDs will be turned on **between 2.5 sec and 5.0 sec.**
- After the replay, start **counting down** to 0.00.0. Then change to the **INITIAL** mode.

3. **Note**

a. When testing your code, TAs will not press Key 0 more than **10** times.

b. You should handle the situation that the user presses key0 repeatedly until changing to the **SHOW** mode (when **show_btn** is pressed or the counter reaches **1.00.0**).

4. **I/O signal specification:**

| | |
|---|---|
| clk | connected to W5 |
| rst | connected to BTND (U17) |
| record_btn | connected to BTNL (W19) |
| show_btn | connected to BTNU (T18) |
| PS2_CLK | signal used for the keyboard |
| PS2_DATA | signal used for the keyboard |
| DISPLAY | signal used for the 7-segment display. |
| DIGIT | signal used for the 7-segment display. |

5. You have to use the given template for your **exam2_C.v**. Do not change the input and output signals.

```
// 108123456 王小明
module exam2_C(
      output wire [6:0] DISPLAY,
      output wire [3:0] DIGIT,
      output wire [15:0] led,
      inout wire PS2_DATA,
      inout wire PS2_CLK,
      input wire rst,
      input wire clk,
      input wire record_btn,
```

```
      input wire show_btn
      );
      parameter [8:0] KEY_CODES [0:3] = {
            9'b0_0111_0000, // right_0 => 70
            9'b0_0110_1001, // right_1 => 69
            9'b0_0111_0010, // right_2 => 72
            9'b0_0111_1010 // right_3 => 7A
      };
    // add your design here
endmodule
```

## ➢ Note

- You should hand in only one Verilog file for each problem. If you have several modules (including clock_divider, debounce, onepulse, SevenSegment, KeyboardDecoder, everything!) in your design, integrate them such as **exam_A.v, exam_B.v, and exam_C.v**.