

## Lab 4: Counters

### Submission Due Dates:

Source Code: 2020/10/27 18:30

Report: 2020/11/01 23:59

### Objective

- 1 Getting familiar with the 7-segment display and switches on the FPGA board with the counter designs.
- 2 Getting familiar with finite-state machines (FSM) in Verilog.
- 3 Every signal connected to the push button (except the reset signal) should be properly processed with the debouncing and/or one-pulse converters.

### Action Items

#### 1 lab4\_1.v

(40%) Design the 2-digit BCD up/down counter and display the counting direction.

- a. I/O list:
  - ✓ Inputs: clk, rst, en, dir
  - ✓ Output: [3:0] DIGIT, [6:0] DISPLAY, max, min
- b. **clk**: (connected to pin **W5**) the clock input with frequency of 100MHZ.
- c. **rst**: (connected to **BTNC**) the positive-edge-triggered (active-high) reset signal to reset the counter value to (decimal) 00.
- d. **en**: (connected to **BTNU**) the control signal to toggle between the *start/resume* and *pause* mode. Press the button to start the counting. Press it again to pause the counting. Press it one more time to resume the counting, and so on.

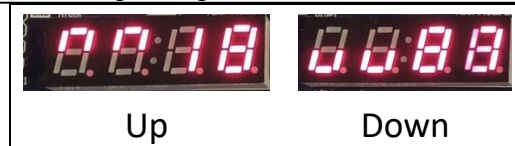
After the reset, the counter is initialized to **00** and remains unchanged. That is, the counter will be in the pause mode initially.

The counter will stop at 99 eventually when counting up; the counter will stop at 00 at the end when counting down.

- e. **dir**: (connected to **BTND**) after the reset, the counter is to count **up**. Press the dir button to change the direction to count down. Press the button one more time to change the direction to count up. Pressing the dir button in the pause mode will change the counting direction as well.

Show the current counting direction on the leftmost two digits of 7-segment. Use two up arrows to indicate the up direction; use two down arrows to indicate the down direction:

Up Arrow:  Down Arrow: 



- f. **DISPLAY[6:0]**: the signals to control the seven LED segments on the 7-segment display.
- g. **DIGIT[3:0]**: the signals to control the four digits on the 7-segment display.  
In this lab, the **two rightmost** digits are used to display the counting number, and the **two leftmost** digits are used to display the counting direction.
- h. **max**: 1 if the counter reaches the largest number (99), and 0 otherwise. The LED0 turns on when max is 1. Otherwise, it turns off.
- i. **min**: 1 if the counter reaches the smallest number (00), and 0 otherwise. the LED1 turns on when min is 1. Otherwise, it turns off.
- j. You have to use the following template for your design:

```
module lab4_1 (
    input clk,
    input rst,
    input en,
    input dir,
    output [3:0] DIGIT,
    output [6:0] DISPLAY,
    output max,
    output min
);
    // add your design here
endmodule
```


Demo: <https://reurl.cc/x0p37b>

## 2 lab4\_2.v

(60%) Implement a stopwatch with the record function. The stopwatch will count up to 2:00.0 (The rightmost digit represents 0.1 seconds. You can ignore the colon ":" and decimal point ".". They are only to help you understand. You don't need to show them on the 7-segment display.)

- a. I/O list:
  - ✓ Inputs: clk, rst, en, record, display\_1, display\_2
  - ✓ Output: [3:0] DIGIT, [6:0] DISPLAY
- b. When the stopwatch is counting, press BTNR to record the current time. The stopwatch can have at most **two time records**. Any further action would not be recorded.
- c. When the stopwatch is paused, use the switch SW0 (or SW1) to show the first (or second) time record on the 7-segment. **You should only change one switch at a**

**time. If you slide up both switches, display  on the 7-segment display.**

- d. When the stopwatch reaches 2:00.0, it will stop automatically. User can check their records as well.
- e. The default record is 0:00.0.
- f. **clk**: (connected to pin **W5**)  
The clock input with the frequency of 100MHz
- g. **rst**: (connected to pin **BTNC**)  
The asynchronous active-high reset. The counter is reset to 0:00.0.
- h. **en**: (connected to pin **BTNU**)  
The control signal to toggle between the count-up mode and pause mode. Press the button to start the counting up. Press it again to pause the counting. Press it one more time to resume the counting, and so on. The stopwatch resets to the pause mode.
- i. **record**: (connected to pin **BTNR**)  
When the stopwatch is counting, push down the button to record the current time.
- j. **display\_1**: (connect to pin **V17**)  
When the stopwatch is paused, slide up the switch SW0 to show the first time record on the 7-segment. (Slide it down before showing the second record.)
- k. **display\_2**: (connect to pin **V16**)  
When the stopwatch is paused, slide up the switch SW1 to show the second time record on the 7-segment. (Slide it down before showing the first record.)  
**Note:** If both switches are slid up, display  on the 7-segment display.
- l. **DISPLAY[6:0]** :  
The signals to control the seven LED segments on the 7-segment display.
- m. **DIGIT[3:0]**:  
The signals to control the four digits on the 7-segment display.
- n. You have to use the following template for your design:

```
module lab4_2 (
    input clk,
    input rst,
    input en,
    input record,
    input display_1,
    input display_2,
    output [3:0] DIGIT,
    output [6:0] DISPLAY
);
    // add your design here
endmodule
```

Demo: <https://reurl.cc/EzgQog>

### 3 Bonus:

(5%) Create the exact 0.1-second timestep for lab4\_2 to count up. You have to explain your implementation both during the demo and in the report to get the extra points. Integrate the bonus feature in lab4\_2, instead of having another stopwatch if you work for the bonus.

## Attention

- ✓ You have to hand in lab4\_1.v, lab4\_2.v. **Upload each source file directly! DO NOT hand in a compressed ZIP file!**
- ✓ You should also hand in your report as lab4\_report\_StudentID.pdf (i.e., lab4\_report\_108456789.pdf).
- ✓ Your report should include diagrams to explain your design (such as FSM, block diagram, etc.).
- ✓ You should be able to answer questions of this lab from TA during the demo.
- ✓ You need to generate the bitstream before the demo.