Lab 04: BCD Counter and Stop Watch

(Oct 16, 2018)

Submission deadlines:

Source Code:	18:30, Oct 16, 2018
Report	23:59, Oct 21, 2018

Objective

To be familiar with the FPGA design flow, the demo-board I/Os, and the design of a simple finite-state machine.

Action Items

1. To get familiar with the 7-Segment Display, please use your switch to provide binary inputs, and display the decimal values (only 0~9, if the value is greater than 9, display 9) on the 7-Segment Display. The four decimal digits should correspond to SW[15:12], SW[11:8], SW[7:4], and SW[3:0].

   I/O signal specification:

   - **clk**: clock signal with the frequency of 100MHz (connected to pin **W5**)
   - **SW:** binary input of each digit. For example, if SW = 16'b0001_1000_0000_0101 7-Segment Display should display 1805.
   - **reset**: **asynchronous** active-high reset (connected to **BTNC**). 7-segment displays should display only value of 1 digit when the **reset** signal is 1
   - **DISPLAY[6:0]**: signals to show the digits on the 7-segment displays.
   - **DIGIT[3:0]**: signals to enable one of the 7-segment displays. You can use the following template for your design.

```
module Lab4_1(DIGIT, DISPLAY, clk, reset, SW);

input [15:0] SW;
input clk;
input reset;
```

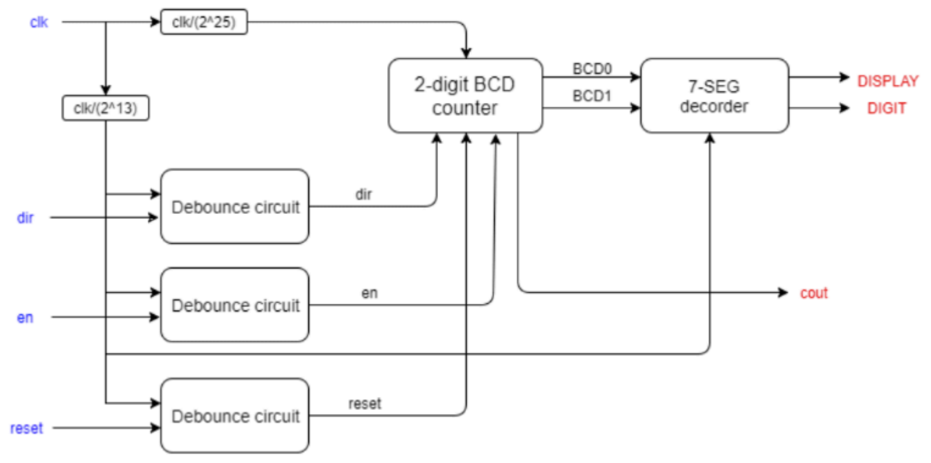
```
output [3:0] DIGIT;
output [6:0] DISPLAY;
    // add your design here
endmodule
```

Demo video: https://youtu.be/YVCiViq4xNM

2. Design and program a 2-digit BCD up/down counter to the FPGA on the demo board. The I/O signals and the block diagram of the counter are shown below.

I/O signal specification:

- **clk**: clock signal with the frequency of 100MHz (connected to pin **W5**)
- **dir** (connected to **BTND**): after the reset, the counter is to count up with the **dir** pushbutton released (dir = 0). Press the **dir** pushbutton to change the direction to count down (dir = 1); When the pushbutton released, the counter will resume to count up (dir = 0). Note: pressing the **dir** pushbutton in the pause mode won't change anything. It only takes effect when the counter is running.
- **reset**: **asynchronous** active-high reset (connected to **BTNC**). The counter is reset to 00 when the **reset** signal is 1
- **en**: toggling between the start/resume and pause. After the reset, the counter stays at **00** (the pause mode). Press the pushbutton to start the counting. Press it again to pause the counting. Press it one more time to resume the counting, and so on and so forth. (connected to **BTNU**)
- **DISPLAY[6:0]**: signals to show the two BCD digits on the 7-segment displays.
- **DIGIT[3:0]**: signals to enable one of the 7-segment displays. In this lab, the two rightmost digits are used.
- **cout**: the carry-out bit of the counter (connected to **LD0**); the LED is turned on when cout is 1, and is turned off otherwise



Color Blue are input signals
Color Red are output signals

Note: We use both the clk/2^13 and clk/2^25 in the block diagram.

You can use the following template for your design.

```
module Lab4_2(DIGIT, DISPLAY, cout, en, reset, dir, clk);

input en;
input reset;
input dir;
input clk;
output [3:0] DIGIT;
output [6:0] DISPLAY;
output cout;
    // add your design here
endmodule
```

Demo video: https://youtu.be/gT8bj_OKtm8

3. Modify your 2-digit BCD up/down counter design a 2-minute time counter. The counter counts up 0:00.0 to 2:00.0 (The right-most digit represents 0.1 second. 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.). The counter will continue counting up when it is enabled. Use a push-bottom with the debounce preprocessing to trigger the counting/stop/restart, and another push-

button with the debounce to reset. The I/O signals are shown below. You have to design a finite-state machine (FSM) with at least three states to control the counter.
Hint: You may have these FSM states: RESET, WAIT, COUNT to complete your design. You may also choose a proper clock divider to approach 0.1 second timescale as much as possible.

I/O signal specification:

- **clk**: clock signal with the frequency of 100MHz (connected to pin **W5**)
- **reset** (connected to **BTNC**): **asynchronous** reset; the stopwatch is reset to 0:00.0 when the **reset** signal is 1. The signal needs to be preprocessed through debounce and onepulse modules.
- **en** (connected to **BTNU**): toggling between the start/resume and pause. After the reset, the stopwatch stays at 0:00.0 (the pause mode). Press the pushbutton to start the counting. Press it again to pause the counting. Press it one more time to resume the counting, and so on and so forth. The signal needs to be preprocessed through debounce and onepulse modules.
- **DISPLAY[6:0]**: signals to show the digits on the 7-segment displays.
- **Dot(connected to dp):** disaplay it on the 7-segment displays to separate the third and fourth digits
- **DIGIT[3:0]**: signals to enable one of the 7-segment displays.

  The four digits represent:

  Digit[3]: Minutes

  Digit[2:1]: Seconds

  Digit[0]: 0.1 Seconds

Debounce&onepulse

The purpose for debounce is to eliminate noise while pressing buttoms.
onepulse yields a one shot pulse signal each time pressing the buttom.

```
module onepulse(pb_debounced,   clk,   pb_1pulse);
      input pb_debounced;
      input   clk;
      output pb_1pulse;
      reg pb_1pulse;
      reg pb_debounced_delay;
      always@(posedgeclk)   begin
            if(pb_debounced==      1'b1      &      pb_debounced_delay==
1'b0)
                  pb_1pulse   <=   1'b1;
            else pb_1pulse   <=   1'b0;
            pb_debounced_delay<=   pb_debounced;
      end
endmodule

module debounce (pb_debounced, pb, clk);
       output  pb_debounced;  //  signal  of  a  pushbutton  after  being
debounced
       input pb; // signal from a pushbutton
       input clk;
       reg [3:0] DFF; // use shift_reg to filter pushbutton bounce
       always @(posedge clk)
       begin
       DFF[3:1] <= DFF[2:0];
       DFF[0] <= pb;
end
assign pb_debounced = ((DFF == 4'b1111) ? 1'b1 : 1'b0);
endmodule
```

You can use the following template for your design.

```
module Lab4_3(DIGIT, DISPLAY, Dot, en, reset, clk);

input en;
input reset;
input clk;
output [3:0] DIGIT;
output [6:0] DISPLAY;
output Dot;
    // add your design here
endmodule
```

Demo video: https://youtu.be/I-sbyC1Xv0o

Challenge:

Try if you can create the exact 0.1 second timestep. You will get extra bonus points for achieving it. You have to explain the design in your report if you can solve this challenge.