ELEX 2117 : Digital Techniques 2
2020 Fall Term

Sequential Logic Design with Verilog

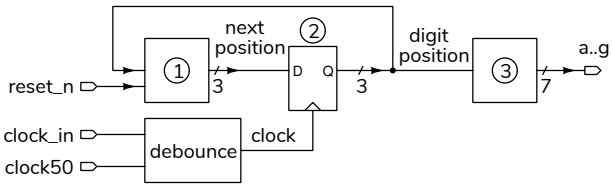
Introduction

In this lab you will implement a circuit that displays successive digits of your BCIT ID on a 7-segment LED display.

You will use the same components, input/output devices and wiring as the previous lab.

Requirements

Your design should implement the following circuit.



It has two pushbutton switch inputs (`reset_n` and `clock_in`), a 50 MHz clock input (`clock50`) and seven segment outputs, (`a` through `g`).

① Is a combinational logic circuit that outputs the next digit position based on the current position and the active-low `reset_n` input. The function is described below. Position values are 3 bits wide and take values between 0 and 7 because BCIT ID's have 8 digits.

② Is a 3-bit register whose value is the digit position being displayed (between 0 and 7). The register's `clock` signal is the "debounced" `clock_in` input.

③ Is a combinational logic circuit that outputs the values of the LED segments that should be lit to show the digit of your BCIT ID for the current digit position.

Your circuit should operate as follows:

- The output should change on the rising edge of the clock (when the `clock_in` pushbutton is *released*).
- If the `reset_n` input is asserted (low) at the rising edge of the clock, the first digit of your BCIT ID should be displayed.
- If the reset input is not asserted at the rising edge of the clock then the next digit of your BCIT ID should be displayed.
- When last digit is reached, if your ID ends with an even number your design should continue from the first digit (position 0), if your ID ends with an odd number your design should continue with the last digit (position 7).

Debouncing

Most mechanical switches briefly interrupt the connection when they switch. This "switch bounce" produces multiple signal edges. So you'll need to "debounce" the `clock_in` switch input [1].

Include a statement such as:

```
debounce debounce0 ( clock_in, clock50, clock ) ;
```

in your Verilog code where `clock_in` and `clock` are the signal names for the un-debounced and debounced clock signals respectively and `clock50` is a 50 MHz clock on the CPLD board that is connected to pin 12.

The module instantiation statement above adds an instance of the `debounce` module to your design with the name `debounce0`, and connects the module inputs and output to the specified signals.

CPLD I/O

The CPLD should be wired up to two pushbutton switches and a seven-segment LED as in the previous lab.

[1] But not `reset_n` since it's level-sensitive rather than edge-sensitive.

Hints

Verilog has an addition operator you can use to compute the next digit position. For example, `p_next = p+1;`.

To save you time, the active-low seven-segment values (`a` to `g`) for digits 0 to 9 are:

```
0  7'h01
1  7'h4f
2  7'h12
3  7'h06
4  7'h4c
5  7'h24
6  7'h20
7  7'h0f
8  7'h00
9  7'h04
```

Examples of the output digit sequences are shown below for the IDs **A00123456** and **A01234567**. The bottom rows show the outputs after the rising edge of the clock ("ʃ" represents a rising edge of the clock):

reset_n	L H H H H L H H H H H H H H H H ...
clock	ʃ ʃ ʃ ʃ ʃ ʃ ʃ ʃ ʃ ʃ ʃ ʃ ʃ ʃ ʃ ʃ ...
A00123456	0 0 0 1 2 0 0 1 2 3 4 5 6 0 0 1 2 ...
A01234567	0 0 1 2 3 0 1 2 3 4 5 6 7 7 7 7 7 ...

Procedure

Follow the procedure in Appendix A of the previous lab to create a project, compile it, and configure your CPLD. Note the following instructions specific to this lab.

Download `debounce.sv` from the course website to your project folder and add it using **Project > Add/Remove Files in Project...**, select the file, **Add > OK**).

You can use **Assignments > Import Assignments** to import your `lab1.qsf` file instead of doing all the pin assignments manually. You'll still need to change `x[0]` to `clock_in` and change `x[1]` to `reset_n` in the pin and pull-up assignments.

Assign pin 12 to the `clock50` 50 MHz oscillator input.

You should end up with the following assignments, possibly using different signal names:

To	Assignment Name	Value
a	Location	PIN_44
b	Location	PIN_42
c	Location	PIN_36
d	Location	PIN_34
e	Location	PIN_30
f	Location	PIN_48
g	Location	PIN_50
clk_50	Location	PIN_12
reset_n	Location	PIN_99
clk_in	Location	PIN_97
clk_in	Weak Pull-Up Resistor	On
reset_n	Weak Pull-Up Resistor	On

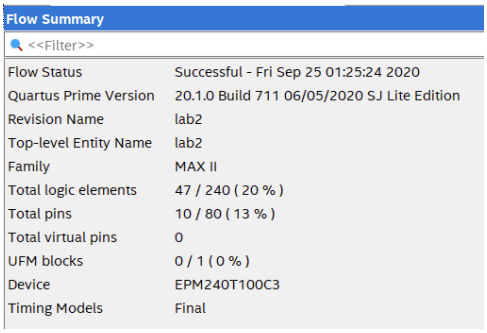
On each release of the pushbutton (rising clock edge) the count value should change to the next digit of your BCIT ID. Holding the reset button and releasing the block button should display the first digit of your BCIT ID. On reaching the last digit of your ID, the display should change (or not) as described above.

Submission

To get credit for completing this lab, submit the following to the Assignment folder for Lab 2 on the course website:

1. A PDF document containing:
   - Your name, BCIT ID, course number and lab number.
   - A listing of your Verilog code. You must follow the coding guidelines given on the "Course Information" section of the course website. Note that these may have changed.

     The listings should be included as text rather than images.
   - a screen capture of your compilation report (**Window > Compilation Report**) similar to this:



Flow Summary	
<<Filter>>	
Flow Status	Successful - Fri Sep 25 01:25:24 2020
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	lab2
Top-level Entity Name	lab2
Family	MAX II
Total logic elements	47 / 240 (20 %)
Total pins	10 / 80 (13 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)
Device	EPM240T100C3
Timing Models	Final

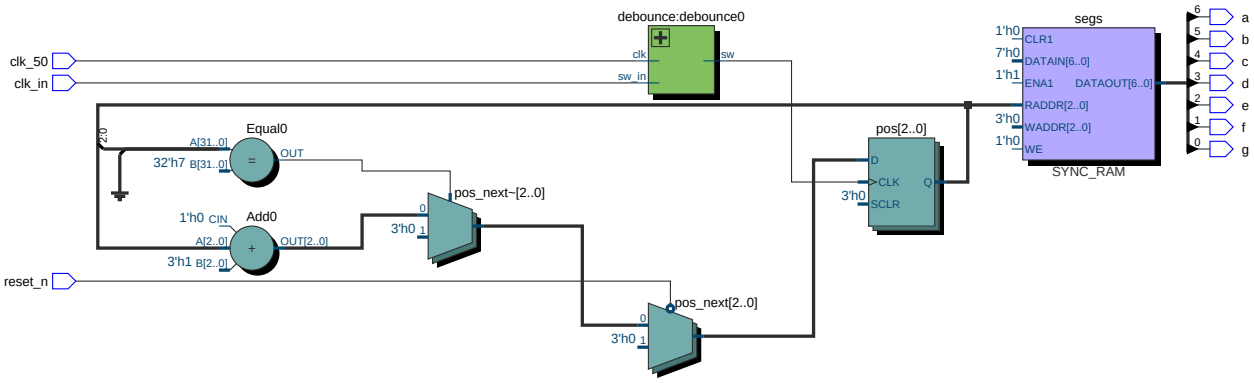


Figure 1: Example RTL Schematic for Lab 2.

2. The PDF file containing the schematic created by **Tools > Netlist Viewers > RTL Viewer** and then **File > Export...** . The file might look like Figure 1.

3. A video showing the pushbuttons and the LED display as you test your design by doing the following:

    - push & hold **`reset_n`**
    - push & release **`clock_in`** once (shows 0)
    - release **`reset_n`**
    - push & release **`clock_in`** three times (shows first three digits)
    - push & hold **`reset_n`**
    - push & release **`clock_in`** once (shows 0)
    - release **`reset_n`**
    - push & release **`clock_in`** eleven times (shows all digits and either stops at the last digit if it's odd or starts again if it's even)