Sequential Logic Design with Verilog

Version 4: added frames around each digit in the display examples.

Introduction

In this lab you will display the last four digits of your BCIT ID on the 4-digit 7-segment LED display.

You will use the same components as in the previous lab, connected the same way.

Multiplexed LED Display

As shown in the schematic of the LED display in the previous lab, the four digits on the LED display share the same segment (anode) connections. The seven segments on each digit have a common cathode connection. You can therefore only display one digit at a time.

Requirements

Each digit should be displayed, in the correct digit position, for one second. The digits should be displayed in order of increasing value, and repeat.

For example, if your BCIT ID were **A0012 6354** the display should show the following: [3], [4], [5], [6], and then repeat starting with [3].

The display should go blank if [*] is not pressed. Pressing [*] should [re]start from the first digit.

As another example, if your BCIT ID were **A0012 1882** the display should show the following: [1], [2], [8], [8], and then repeat.

Design

One possible solution is described below. It consists of two parts.

One part is a register whose output, **en**, enables the digits one after the other. For example, if you have a clock signal **clk** (see the *Clock* section below):

```
output logic [3:0] en,    // active-low digit enable
 ...
always_ff @(posedge clk)
   en <= ... ;      // next value of en (depends
                    // on values of col and en)
```

The value assigned to **en** in the **always_ff** statement would depend on whether [*] was pressed and on the current value of **en**.

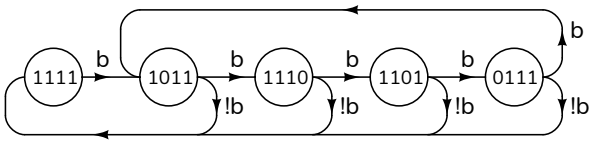
Consider the first example above. The value of **en** should be set to **4'b1111** (all digits off) if [*] is not pressed. If [*] is pressed and **en** is **4'b1111** then **en** should be set to **4'b1011**; if [*] is pressed and **en** is **4'b1011** then it should be set to **4'b1110**; etc. This results in **en** cycling through the four values while [*] is pressed.

Note that the value of **en** only changes on the rising edge (**posedge**) of **clk**.

We can describe this behaviour with a "state transition table":

en	col[3]	next en
any	1	4'b1111
4'b1111	0	4'b1011
4'b1011	0	4'b1110
4'b1101	0	4'b1101
4'b1110	0	4'b0111

or a "state transition diagram:"



b: !col[3]

where the values in the circles show the values of the register **en**. The arrows labelled b show how **en** changes when [*] is pressed and the arrows labelled !b show how **en** changes when [*] is not pressed.

This means **en** is set to **4'b1111**, regardless of its current value, whenever **b** is false (**col[3]==1**). Otherwise the **en**able output cycles through the four other values.

A second part of your design should set the seven segments (**a** through **g**) to the correct values for the digit selected by **en**. This part will be similar to the previous lab.

Component Connections

The CPLD board, keypad and LED display should be connected as in the previous lab. Note that you will need to connect all four digit enables.

Pre-Lab Assignment

Prepare:

1. A table with three columns showing: the digits of your ID in the order they should be displayed, the corresponding segment output values (in hexadecimal), and the corresponding `en` values. For the second example above this would be:

digit	`a...g`	`en`
1	`7'h7e`	`4'b0111`
2	`7'h6d`	`4'b1110`
8	`7'h7f`	`4'b1011`
8	`7'h7f`	`4'b1101`

2. A Quartus Project including pin assignments and a module with the required inputs and outputs and an instantiation of the clock divider module.

Procedure

Follow the Logic Synthesis procedure in the "Software Installation and Use" document along with the changes described below.

Importing Pin Assignments

You can import the pin assignments from the previous lab if you are not changing them – this is recommended. Select **Assignments** / **Import Assignments...** and select `lab1.qsf` from your previous lab's project folder as the file. Click on **Categories...**, un-check everything except **Pin and Location Assignments** and click on **OK**. Check that the pin assignments are correct. You may need to add `en[3]` through `en[1]` which were not used the previous lab. Also check the weak pull-up resistor assignment on the `col` inputs which may not have been imported with the pin numbers.

Clock

The board has a 50 MHz clock signal connected to pin 12 of the CPLD. This signal was not used in the previous lab although it was listed in the sample pin assignments and named `clk50`. Add this pin assignment if you don't have it already.

You will need to use a supplied "clock divider" module to create a 1 Hz clock from the 50 MHz clock. To do this, copy the `clkdiv.sv` file from the course website to your project folder and add it to your project (**Project > Add/remove Files in Project..**). This module has one input, the 50 MHz clock and one output, a clock signal with a configurable frequency. You can instantiate a 1 Hz clock in your `lab2.sv` file with the following two lines:

```
logic clk ;
clkdiv #(1) c0 ( clk50, clk ) ;
```

You can then use the `clk` signal in your register descriptions:

```
always_ff @(posedge clk) en <= ...
```

Lab Report

Submit the following to the appropriate Assignment folder on the course website:

1. A PDF document containing:
   - A listing of your Verilog code.
   - A screen capture of your compilation report.
   - A block diagram of your design. It should include the inputs and outputs, the clock divider, a state register, and the combinational logic driving the segment outputs.
   - a photo of the display when you change the parameter of the `clkdiv` module instantiation from 1 to 5000:

     ```
     clkdiv #(5000) c0 (clk50, clk );
     ```

     and press [*]. The digits in the photo should be upright. For example:

2. If you were not able to demonstrate your solution to the lab instructor during your scheduled lab period, submit a video showing the keypad and the LED display as you: push and hold [*] while your display sequences through at least 5 digits, then push [7], [5], and [3]. Then push [*] to confirm the sequence begins with the first digit. A sample video demonstration is available on the course website.

Follow the *Report and Video Guidelines* and the *Coding Guidelines* documents on the course website.