Lab 1: Display Driver

Due: 11:59:59pm, Sunday April 23rd, 2017

1 Problem Description

7-segment displays are used universally in the digital clocks, meters and other electronic devices. In this lab, we will build a display driver for a simple one-digit 7-segment display, which is connected to a digit keypad as shown in Figure 1.

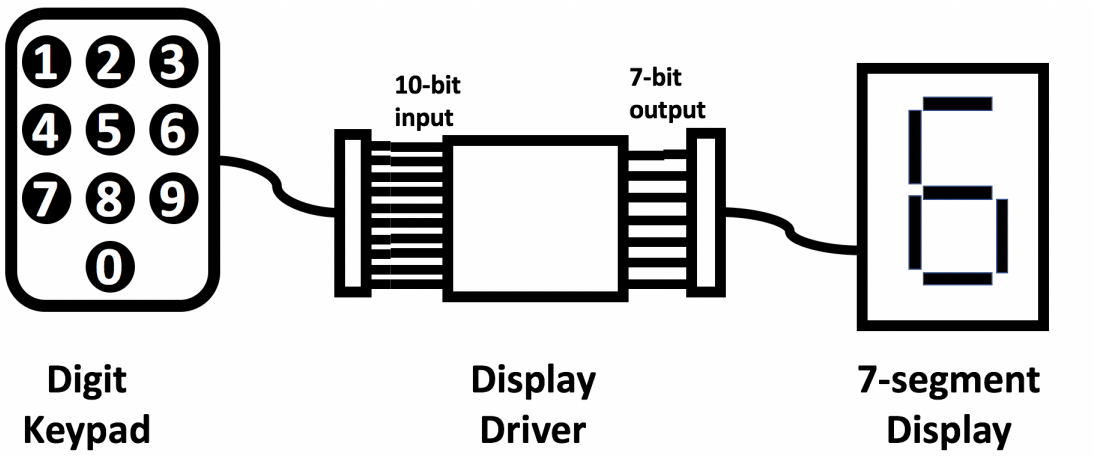

Figure 1: A digit keypad is connected to a display driver, which drives a 7-segment display.

7-Segment Display 7-segment display is composed of seven LEDs which can be controlled individually. The seven segments are numbered 0 to 6 as shown in Figure 2. The display uses combinations of these segments to produce visual representation of digits. Figure 2 also shows how each number is displayed on the 7-segment display (Pay special attention to how digits 1, 6, and 9 are displayed.). The *i-th* segment is ON if *i-th* ($0 \leq i \leq 6$) line is "1" and OFF if it is "0".

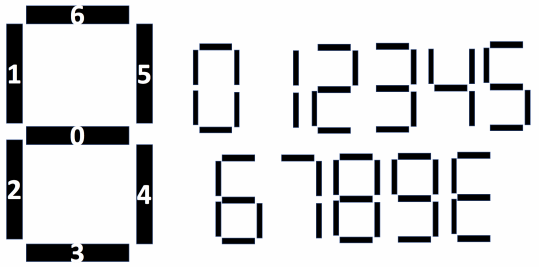

Figure 2: 7-segment display and decimal numbers, and error code (E) representations.

Keypad Input As shown in Figure 1, when you press any button on the keypad, the respective bit in the 10-bit output will be ON. When you press two or more buttons at the same time, the respective bits will all be ON. When no button is pressed, all the 10 output bits will be OFF.

Display Driver Our goal is to write the decoder function that takes a 10-bit input and produces a 7-bit output, one for each segment. It displays the appropriate number if one key is pressed, and `E` if multiple keys are pressed.

2 Lab Assignment

First, make a copy of the source code using the following command

```
$ cp -r  /home/linux/ieng6/cs140g/public/lab1 .
```

Please answer the following questions.

1. Fill the Input/Output table for the display driver given in Figure 2. (20 points)

Table 1: Input/Output Function Table for 7-segment Decoder

Digit	Input Bits										Output Bits						
	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	O6	O5	O4	O3	O2	O1	O0
0	0	0	0	0	0	0	0	0	0	1							
1	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0							
3	0	0	0	0	0	0	1	0	0	0							
4	0	0	0	0	0	1	0	0	0	0							
5	0	0	0	0	1	0	0	0	0	0							
6	0	0	0	1	0	0	0	0	0	0							
7	0	0	1	0	0	0	0	0	0	0							
8	0	1	0	0	0	0	0	0	0	0							
9	1	0	0	0	0	0	0	0	0	0							
No Input	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	Others																

2. Please complete the following BSV function to implement the decoder. (80 points)

   Your score for this part will be proportional to the number of cases you got correct.

```
function Bit#(7) sevenSegmentDriver(Bit#(10) number);
  ....
endfunction
```

3 Submission

For Question 1, write your answers in `Lab1.txt` in the `lab1/` folder. For Question 2, write your answers in `SevenSegmentDriver.bsv` in lab1/ folder. Do not change the function signature (name or the type of the function or the arguments) as we will use a different testbench to evaluate your solution. The provided testbench is only to help you finish this assignment. Use `make` command to compile and run your code.

Write down your name, PID and e-mail address in `Lab1.txt` located in `lab1/` directory. If you worked in a group, write down your partner's information as well. While your solutions can be identical, each group member must make their own submission. You will be scored based on your own submission.

To submit your assignment, simply run the following command from `lab1/` directory:

```
$ bundleP1 <your-email>
```

This command packs all necessary files which you modified in this assignment and turns them in. You can make as many submissions as you want before the deadline but only the last one will be recorded.

Submission verification

You will receive a confirmation email on the given email-id. Check the submission details and report to the TAs in case of any inconsistencies via Piazza. Keep this email for your records.

You can verify that you have submitted successfully by running the bundle script again:

```
$ bundleP1 <your-email>
```

If you have successfully submitted the files, then the following will appear on your screen.

```
A previously turned-in file exists:
...
```

Submission and verification procedures will be similar for all future labs except that script name would change. **Note that it is your responsibility to verify your submission. No requests regarding unsuccessful submission would be entertained later.**

4 Resources

1. BSV by Example `http://csg.csail.mit.edu/6.S078/6_S078_2012_www/resources/bsv_by_example.pdf`
2. BSV Reference Guide `http://csg.csail.mit.edu/6.S078/6_S078_2012_www/resources/reference-guide.pdf`
3. Computer Architecture: A Constructive Approach. `http://csg.csail.mit.edu/6.175/resources/archbook_2015-08-25.pdf`