Lab 8

4-digit Display

Updated: March 23, 2020
Version: SystemVerilog 2017

8.1 Introduction

This lab builds on the previous ones, where you have created a 2-digit, 7-segment display and a BCD converter. Now, you will expand this to a 4-digit display and add the ability to switch between hexadecimal and decimal (BCD) output.

8.2 Objectives

After completing this lab, you should be able to:

- Use parameters to create flexible, reusable modules
- Import modules, modify modules, and use them to design a modular system

8.3 Procedure

8.3.1 Import Previous

Copy your `sseg_decoder`, `bcd11`, and `add3` modules from previous labs. If needed, modify your BCD module to have a single, 16-bit output (instead of 4 separate outputs for the 4 digits).

8.3.2 mux2

Create a new 2-input MUX named `mux2` using a parameter to set the number of bits of the inputs and the output. Use the conditional operator, an `if` statement, or a `case` statement for this MUX. Copy your test bench from the previous lab where you tested a MUX, modify it, and test your new one.

8.3.3 mux4

Create a new 4-input MUX named `mux4` again using a parameter to set the number of bits of the inputs and the output. Use a `case` statement for this MUX. Create a test bench and verify its operation.

8.3.4 Anode Decoder

Previously, we simply used a NOT gate to switch between the two output digits. With four digits, we need a slightly more complex circuit. The truth table is given in Table 8.1. Recall that the output is inverse logic (zero means "on"). Thus, as the input counts 0, 1, 2, 3, we are turning each digit on: 0th, 1st, 2nd, 3rd.

This can be implemented as a set of individual gates or as `case` statement. Put this in a file named `an_decoder`.

Table 8.1: Anode decoder truth table

in	out
00	1110
01	1101
10	1011
11	0111

8.3.5 Four Digit Driver

Create a new file named `sseg4` and implement the diagram shown in Figure 8.1. (You can copy your `sseg2_bcd` file from the previous lab and modify it if you want or simply start a new one. It will be significantly different.)

8.3.6 Top-level File

Create another file named `sseg4_manual` that will interface your `sseg4` module with the FPGA board.

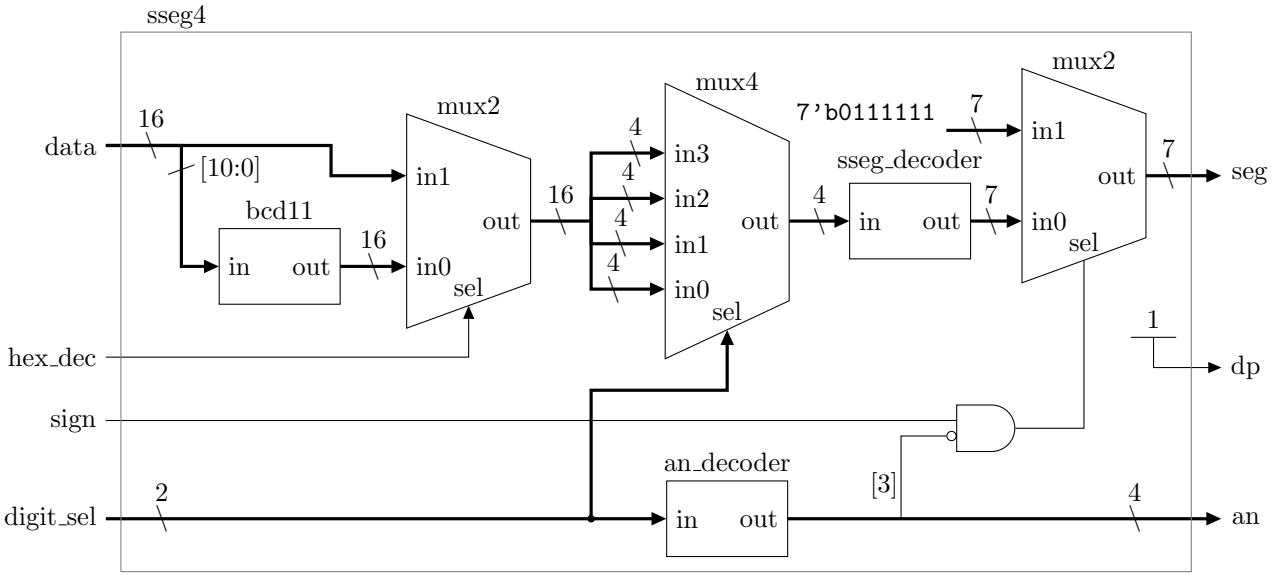

Figure 8.1: 4-digit 7-seg driver (`sseg4`) schematic

Use it to connect the switches to the inputs of `sseg4` and the outputs to the 7-seg wires, as shown in Figure 8.2.

After creating `sseg4_manual`, if you have a Basys3 board available, use the On-Board Testing procedure, otherwise use the Top-Level Simulation procedure.

8.3.7 On-Board Testing

Add the necessary constraint files. Implement your design on the Basys3 board, verify its operation, take a picture after each Operation step, and include the pictures in your report. Make sure that each picture shows all the switch settings in addition to the display.

Operation

Here is how your board should operate:

You should see the first digit lit. Set sw[15] = 1 to output in hex. Change sw[3:0] and set the value to `C`.

Now make sw[15] = 0. The value should change to 2. Make sw[12] = 1. The 2nd digit should be lit and have the number 1. You are now displaying the decimal equivalent of C, which is 12.

Set sw[13:12] = 10. The third digit should be lit and be 0. Change sw[7] = 1. The output should now be a 1, which is the first digit of 128 + 12 = 140. Change sw[13:12] = 01 to confirm that the second digit is a 4, and then to 00 to confirm that the last digit is a 0.

Now change sw[13:12] = 11. The fourth digit should be lit and be 0. Change sw[10] = 1, which adds 1024 and should make the fourth digit a 1. Changing sw[15] = 0 should make the output 0. Is that right? *Hint*: what did you set the upper 4 `data` bits to?

Changing sw[14] = 1, should display a negative sign on the fourth digit. Changing sw[13:12] should change the digits and none of the other should display the negative sign, only the fourth.

8.3.8 Top-Level Simulation

You will need to download the `basys3.sv` and `disp.sv` files from the Lab 8 folder in Canvas. The `basys3` module will serve as a top-level test bench to simulate your `sseg4_manual` module. The `disp` module will facilitate the reporting of displayed values in a more human-readable format. Download `basys3.sv` and `disp.sv` and add them to your Vivado project as simulation sources. Set your simulation top level

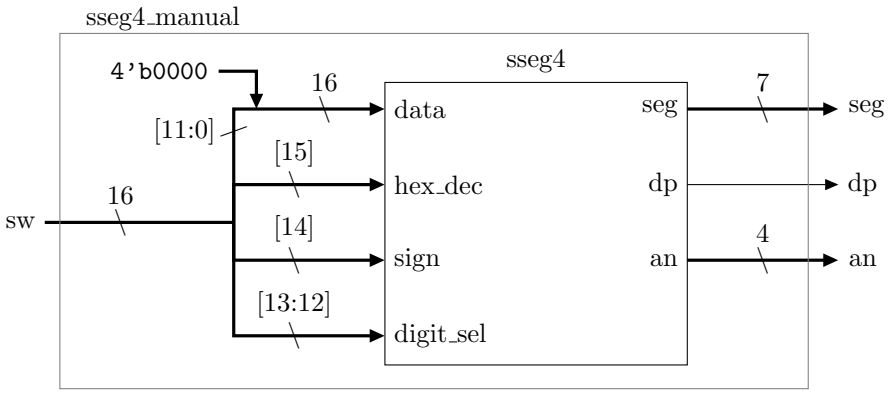

Figure 8.2: Top-level module schematic

to `basys3`. If the simulation runs correctly, you will see the output in Listing 8.1 in the TCL Console tab:

8.4 Deliverables

Submit a report containing the following:

1. Simulations with ERTs for 3 modules (`mux2`, `mux4`, and `an_decoder`)
2. Pictures of your board for each Operation step for On-Board Testing, or a copy of the relevant TCL Console output for Top-Level Simulation
3. Verilog source files written as part of this lab (`mux2`, `mux4`, `an_decoder`, `sseg4`, `sseg4_manual`)

Listing 8.1: Top-level simulation output

```
    +--------- Digit 3 (? = incorrect segment values)
    |+-------- Decimal point
    ||+------- Digit 2 (? = incorrect segment values)
    |||+------ Decimal point
    ||||+----- Digit 1 (? = incorrect segment values)
    |||||+---- Decimal point
    ||||||+--- Digit 0 (? = incorrect segment values)
    |||||||+-- Decimal point
    ||||||||
-->       0
-->       C
-->       2
-->     1
-->   0
-->   1
-->     4
-->       0
--> 0
--> 1
--> 0
--> -
-->   4
-->     8
-->       C
```