Lab 2: Numbers and Displays

CSC 258, University of Toronto

January 20, 2014

About this Lab

Learning Objectives

1. To give you more practice in designing combinational circuits with Verilog.
2. To learn about binary-to-decimal number conversion and binary-coded-decimal (BCD) addition.

Marking Scheme

Read and understand the Pre-lab section before the lab. All items in this document labeled **PRELAB TODO:** are to be completed before arriving to the lab, and are worth 1 mark. You will **not** be allowed to do in lab work unless you have your prelab work handed in. Items marked **INLAB TODO:** are to be completed while your are in the lab. Parts I and II together are worth a mark, as are parts III and IV. Part V is marked as Advanced, and is worth 1 mark if completed.

1 Pre-lab Assignment

You are required to write the Verilog code for all of the following subsections. Perform the tasks marked as **PRELAB TODO:** and bring your prepared Verilog code to the lab to be marked by the TAs.

1.1 Part I

Figure 1*a* shows a circuit for a *full adder*, which has the inputs a, b, and c_i, and produces the outputs s and c_o. Parts b and c of the figure show a circuit symbol and truth table for the full adder, which produces the two-bit binary sum $c_o s = a + b + c_i$. Figure 1*d* shows how four instances of this full adder module can be used to design a circuit that adds two four-bit numbers. This type of circuit is usually called a *ripple-carry* adder, because of the way that the carry signals are passed from one full adder to the next.

PRELAB TODO: Determine the chips and pin assignments needed to implement a 4-bit adder of this type using TTL logic.

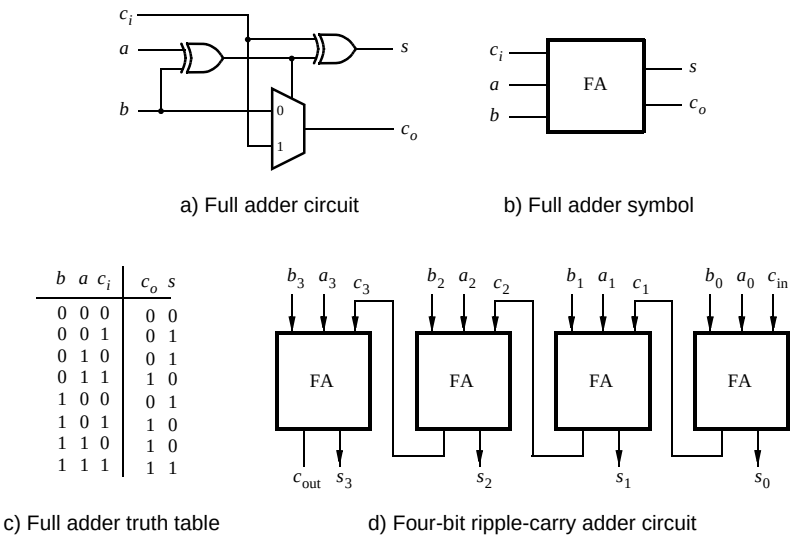

Figure 1: A ripple-carry adder circuit.

1. To implement this ripple-carry adder, you'll need to use the 4-bit adder chip (74LS283). Search the web for 74LS283 to find the pin layout for this chips, and consult the notes and the diagram above on how to wire the adder components together.

2. Note that pins A4 and B4 are the most significant bits of the input values, and that all the pins need to be connected to something (i.e. the c_in pin needs to be grounded if you don't plan on using it). Also note that the carry bits between adders are connected internally within the chip.

3. Label the pin numbers that your design will use to implement this 4-bit adder. When you get to the lab, you will need to connect the switches on the switch board to each of the inputs of the adder, and outputs of the adder circuit to the lights. You may want to have the pins of the ribbon cable labelled as well in anticipation.

4. The rest of this part will be performed in the lab.

1.2 Part II

PRELAB TODO: Write Verilog code that implements this circuit, as described below.

1. Create a new Quartus II project for the adder circuit from Part I. Write a Verilog module for the full adder subcircuit and write a top-level Verilog module that instantiates four instances of this full adder.

2. Use switches SW_{7-4} and SW_{3-0} to represent the inputs A and B, respectively. Use SW_8 for the carry-in c_{in} of the adder. Connect the SW switches to their corresponding red lights LEDR, and connect the outputs of the adder, c_{out} and S, to the green lights LEDG.

3. Include the necessary pin assignments for the DE2 board and compile the circuit.

4. Test your circuit by trying different values for numbers A, B, and c_{in}.

1.3 Part III

It is sometimes useful to build circuits in which numbers are represented in decimal format, where each decimal digit is represented using four bits. This scheme is known as the *binary coded decimal* (BCD) representation. As an example, the decimal value 59 is encoded in BCD form as 0101 1001.

PRELAB TODO: Write Verilog code that converts a four-bit binary number represented by SW_{3-0} into two-digit BCD format. Also the two-digit BCD number must be displayed on the 7-segment displays *HEX1* and *HEX0*.

Let the value V be denoted by SW_{3-0}. We want V to be converted into a two-digit BCD format, in turn displayed on *HEX1* and *HEX0*, respectively. Your circuit should be able to display the digits from 0 to 9 on each 7-segment display. Table 1 shows the required output values.

Binary value	Decimal digits	
0000	0	0
0001	0	1
0010	0	2
...	...	...
1001	0	9
1010	1	0
1011	1	1
1100	1	2
1101	1	3
1110	1	4
1111	1	5

Table 1. Binary-to-decimal conversion values.

A partial design of this circuit is given in Figure 2. It includes a comparator that checks when the value of V is greater than 9, and uses the output of this comparator to produce the correct BCD digits. In Figure 2, the comparator must output a 1 if the binary input has a value greater than 9, and 0 otherwise. If the output of the comparator is 0, the multiplexers set the binary value V as BCD digit d_0. If the comparator output is 1, then Circuit A is responsible for setting the value of d_0 instead.

Circuit B is responsible for sending the correct signals to the second 7-segment. If the output of the comparator is 1, Circuit B must display number 1 on the 7-segment, otherwise number 0 must be displayed.

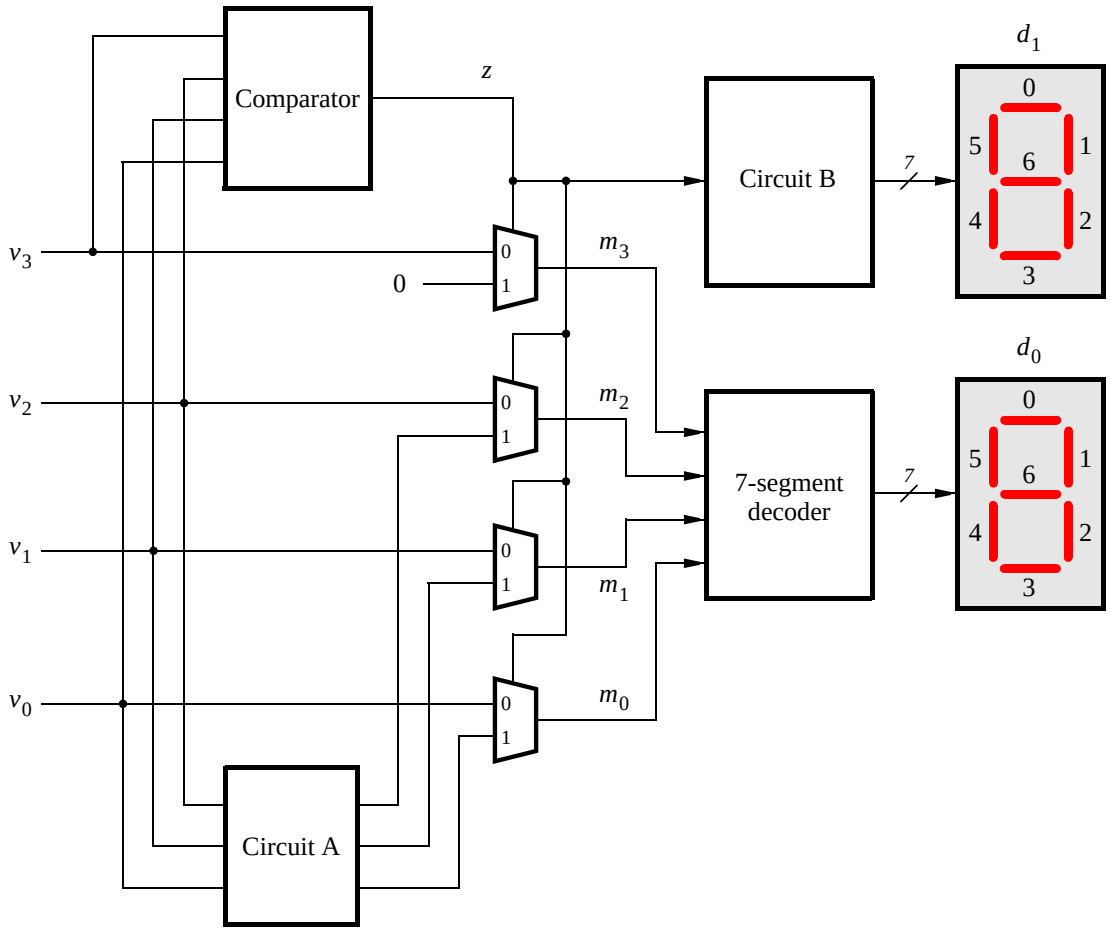

Figure 2: Partial design of the binary-to-decimal conversion circuit.

You are to complete the design of this circuit by creating a Verilog module which includes the comparator, multiplexers, circuit A, circuit B and the 7-segment decoder. Your Verilog module should have the four-bit input V, and the two 7-segment outputs. The intent of this exercise is to use simple Verilog **assign** statements to specify the required logic functions using Boolean expressions. Your Verilog code should not include any **if-else**, **case**, or similar statements.

1. Make a Quartus II project for your Verilog module. Compile the circuit and use functional simulation to verify the correct operation of your comparator, multiplexers, and circuit A, circuit B and the 7-segment decoder.

2. Change the inputs and outputs of your code to use switches SW_{3-0} on the DE2 board to represent the binary number V, and the displays *HEX1* and *HEX0* to show the values of decimal digits d_1 and d_0. Make sure to include in your project the required pin assignments for the DE2 board.

3. Recompile the project. Test your circuit by trying all possible values of V and observing the output displays.

1.4 Part IV

In part III we discussed the conversion of binary numbers into BCD.

PRELAB TODO: Design and implement a circuit that adds two BCD digits and produces a two-digit BCD result. The inputs to the circuit are BCD numbers A and B. The output should be a two-digit BCD sum S_1S_0. Note that the largest sum that needs to be handled by this circuit is $S_1S_0 = 9 + 9 = 18$. Perform the steps given below.

1. Create a new Quartus II project for your BCD adder. You should use the four-bit adder circuit from part III to produce a four-bit sum and carry-out for the operation $A + B$. A circuit that converts this five-bit result, which has the maximum value 18, into two BCD digits S_1S_0 can be designed in a very similar way as the binary-to-decimal converter from part I. Write your Verilog code using simple **assign** statements to specify the required logic functions–do not use other types of Verilog statements such as **if-else** or **case** statements for this part of the exercise.

2. Use switches SW_{7-4} and SW_{3-0} for the inputs A and B, respectively. Connect the SW switches to their corresponding red lights LEDR, and connect the four-bit sum and carry-out produced by the operation $A + B$ to the green lights LEDG. Display the BCD values of A and B on the 7-segment displays *HEX6* and *HEX4*, and display the result S_1S_0 on *HEX1* and *HEX0*.

3. Since your circuit handles only BCD digits, check for the cases when the input A or B is greater than nine. If this occurs, indicate an error by turning on the green light $LEDG_8$.

4. Include the necessary pin assignments for the DE2 board and compile the circuit.

5. Test your circuit by trying different values for numbers A and B.

1.5 Part V - Advanced

In this part, you will design a circuit that can add two 2-digit BCD numbers, A_1A_0 and B_1B_0 to produce the three-digit BCD sum $S_2S_1S_0$. This could be done by creating Verilog code for a two-digit BCD adder by using two instances of the Verilog code for a one-digit BCD adder from part IV. A different approach for describing the two-digit BCD adder in Verilog code is to specify an algorithm like the one represented by the following pseudo-code:

```
1   T0 = A0 + B0
2   if (T0 > 9) then
3       Z0 = 10;
4       c1 = 1;
5   else
6       Z0 = 0;
7       c1 = 0;
8   end if
9   S0 = T0 − Z0

10  T1 = A1 + B1 + c1
11  if (T1 > 9) then
12      Z1 = 10;
13      c2 = 1;
14  else
15      Z1 = 0;
16      c2 = 0;
17  end if
18  S1 = T1 − Z1
19  S2 = c2
```

It is reasonably straightforward to see what circuit could be used to implement this pseudo-code. Lines 1, 9, 10, and 18 represent adders, lines 2-8 and 11-17 correspond to multiplexers, and testing for the conditions $T_0 > 9$ and $T_1 > 9$ requires comparators.

PRELAB TODO: Write Verilog code that corresponds to this pseudo-code. Note that you can perform addition operations in your Verilog code instead of the subtractions shown in lines 9 and 18. The intent of this part of the exercise is to examine the effects of relying more on the Verilog compiler to design the circuit by using **if-else** statements along with the Verilog > and + operators. Perform the following steps:

1. Create a new Quartus II project for your Verilog code.
2. Use switches SW_{15-8} and SW_{7-0} to represent 2-digit BCD numbers A_1A_0 and B_1B_0, respectively. The value of A_1A_0 and B_1B_0 should be displayed on red LEDs. Display the BCD sum, $S_2S_1S_0$, on the 7-segment displays *HEX2*, *HEX1* and *HEX0*. Compile your circuit.
3. Test it by trying different values for numbers A_1A_0 and B_1B_0.

2 In lab work

For each part of the pre-lab, compile and download your code to the FPGA. Test the functionality of your design by toggling the switches and observing the displays.

2.1 Part I

INLAB TODO: Implement your adder design on the protoboard, using four switches on the switch board for each of the two input values, and use the lights as the carry and 4-bit sum output. Once you've got it working, demonstrate it to one of the teaching assistants. Consider implementing the least significant bits first and making sure those work before wiring the other bits of the chip.

2.2 Part II

INLAB TODO: Show your board to your TA once you have completed and verified this part.

2.3 Part III

INLAB TODO: Show your board to your TA once you have completed and verified this part.

2.4 Part IV

INLAB TODO: Show your board to your TA once you have completed and verified this part.

2.5 Part V - Advanced

INLAB TODO: Show your board to your TA once you have completed and verified this part.

2.6 Clean up

INLAB TODO: Clean up your station once you are done with the lab.