Laboratory Exercise 2

Numbers and Displays

This is an exercise in designing combinational circuits that can perform binary-to-decimal number conversion and binary-coded-decimal (BCD) addition.

Part I

We wish to display on the 7-segment displays *HEX3* to *HEX0* the values set by the switches SW_{15-0}. Let the values denoted by SW_{15-12}, SW_{11-8}, SW_{7-4} and SW_{3-0} be displayed on *HEX3*, *HEX2*, *HEX1* and *HEX0*, respectively. Your circuit should be able to display the hex digits from 0 to F.

1. Create a new project which will be used to implement the desired circuit on the Altera DE2 board. The intent of this exercise is to manually derive the logic functions needed for the 7-segment displays.
2. Write a Verilog file that provides the necessary functionality. Include this file in your project and assign the pins on the FPGA to connect to the switches and 7-segment displays, as indicated in the User Manual for the DE2 board. The procedure for making pin assignments is the same as in Homework 1.
3. You should have two modules for this part: Hex7Seg and Part1.
4. Compile the project and download the compiled circuit into the FPGA chip.
5. Test the functionality of your design by toggling the switches and observing the displays.

Part II

You are to design a circuit that converts a four-bit binary number $V = v_3 v_2 v_1 v_0$ into its two-digit decimal equivalent $D = d_1 d_0$. Table 1 shows the required output values. A partial design of this circuit is given in Figure 1. It includes a comparator that checks when the value of V is greater than 9, and uses the output of this comparator in the control of the 7-segment displays. You are to complete the design of this circuit by creating a Verilog module which includes the comparator, multiplexers, and circuit. Your Verilog module should have the four-bit input *V*, and two hex displays.

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.

Perform the following steps:

1. Make a Quartus II project for this part. Structure your circuit as shown in Figures 1a and 1b.
2. Compile the project, and then download the circuit into the FPGA chip.
3. Test your circuit by trying all possible values of V and observing the output displays.

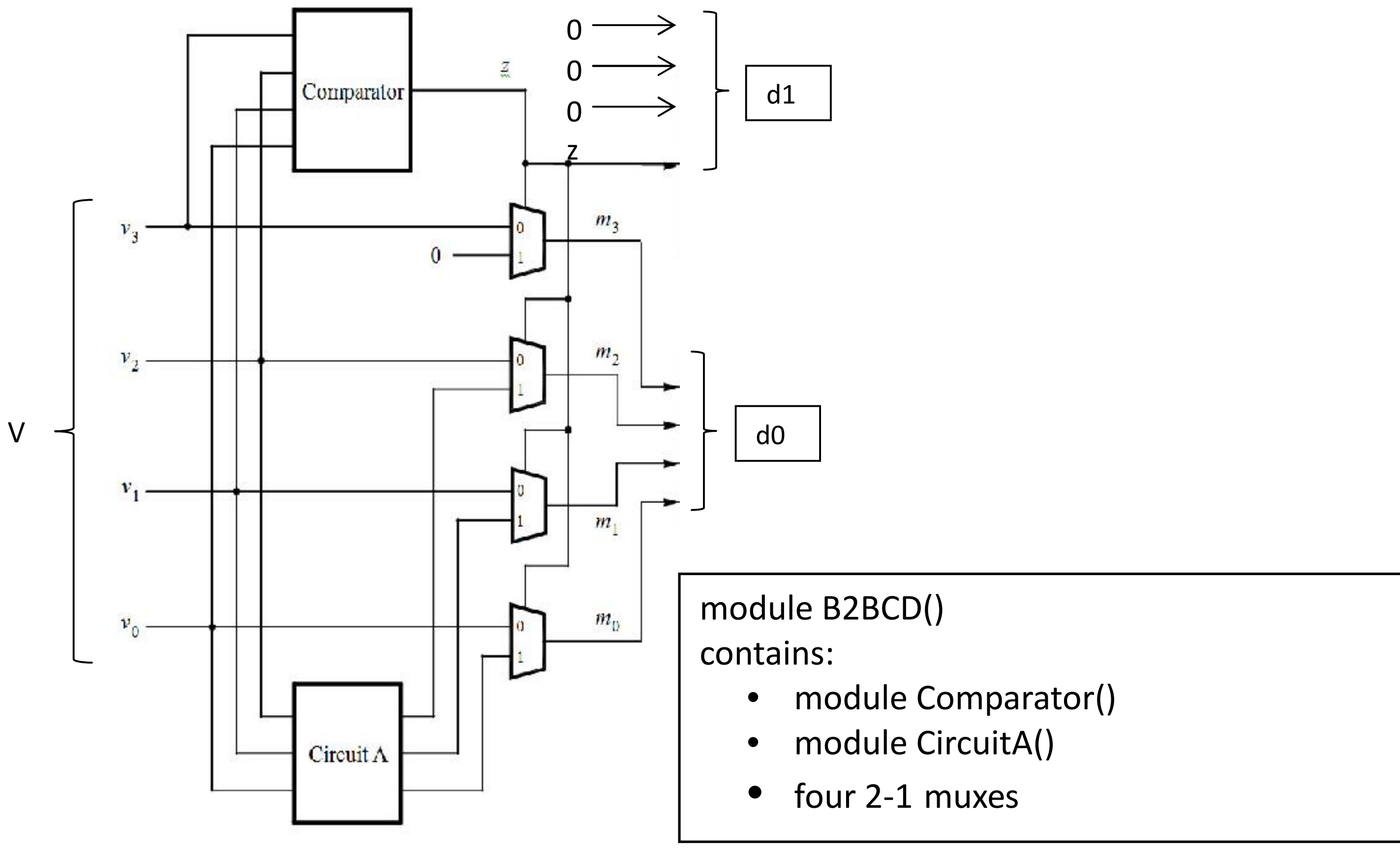


Figure 1a. Partial design of the binary-to-decimal conversion circuit.

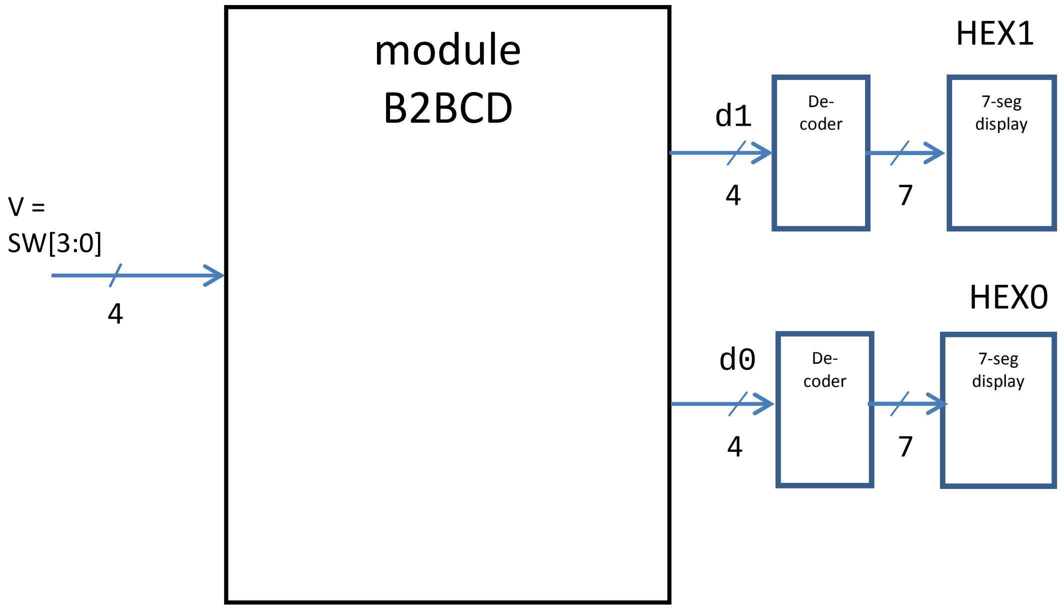
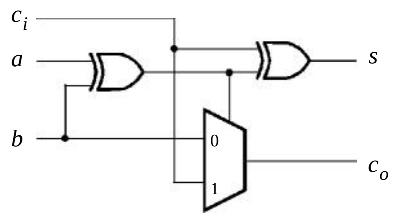


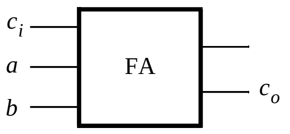
Figure 1b. Module Part2()

Part III

Figure 2*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 2*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. Write Verilog code that implements this circuit, as described below.



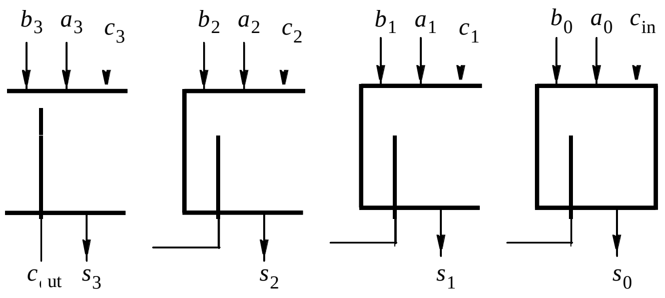
a) Full adder circuit



b) Full adder symbol

b	a	c_i	c_o	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

c) Full adder truth table



d) Four-bit ripple-carry adder circuit
(see figure at end of this document).

Figure 2. A ripple-carry adder circuit.

1. Create a new Quartus II project for a 4-bit adder circuit. Write a Verilog module for the adder circuit and write a top-level Verilog module that instantiates your 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, compile the circuit, and download it into the FPGA chip.
4. Test your circuit by trying different values for numbers A, B, and c_{in}.

Part IV

In part II we discussed the conversion of binary numbers into decimal digits. It is sometimes useful to build circuits that use this method of representing decimal numbers, in which 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.

You are to design a circuit that adds two BCD digits. The inputs to the circuit are BCD numbers A and B, plus a carry-in, c_{in}. The output should be a two-digit BCD sum $S_1 S_0$. Note that the largest sum that needs to be handled by this circuit is $S_1 S_0 = 9 + 9 + 1 = 19$. 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 19, into two BCD digits $S_1 S_0$ can be designed in a very similar way as the binary-to-decimal converter from part II.

2. Use switches SW_{7-4} and SW_{3-0} for the inputs A and B, respectively, and use SW_8 for the carry-in. 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 $S_1 S_0$ 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, compile the circuit, and download it into the FPGA chip.
5. Test your circuit by trying different values for numbers A, B, and c_{in}.

Part V

Design a circuit that can add two 2-digit BCD numbers, $A_1 A_0$ and $B_1 B_0$ to produce the three-digit BCD sum $S_2 S_1 S_0$. Use two instances of your circuit from part IV to build this two-digit BCD adder. Perform the steps below:

1. Use switches SW_{15-8} and SW_{7-0} to represent 2-digit BCD numbers $A_1 A_0$ and $B_1 B_0$, respectively. The value of $A_1 A_0$ should be displayed on the 7-segment displays *HEX7* and *HEX6*, while $B_1 B_0$ should be on *HEX5* and *HEX4*. Display the BCD sum, $S_2 S_1 S_0$, on the 7-segment displays *HEX2*, *HEX1* and *HEX0*.
2. Make the necessary pin assignments and compile the circuit.
3. Download the circuit into the FPGA chip, and test its operation.

Part VI

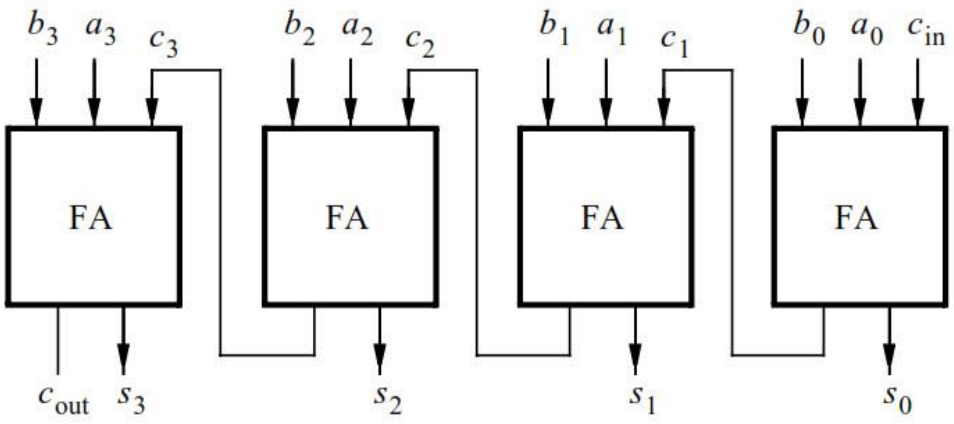
In part V you created Verilog code for a two-digit BCD adder by using two instances of the Verilog code for a one-digit BCD adder from part IV. A different approach for describing the two-digit BCD adder in Verilog code is to specify an algorithm like the one represented by the following pseudo-code:

```
1   $T_0 = A_0 + B_0$
2   if ($T_0 > 9$) then
3       $Z_0 = 10$;
4       $c_1 = 1$;
5   else
6       $Z_0 = 0$;
7       $c_1 = 0$;
8   end if
9   $S_0 = T_0 - Z_0$

10  $T_1 = A_1 + B_1 + c_1$
11  if ($T_1 > 9$) then
12      $Z_1 = 10$;
13      $c_2 = 1$;
14  else
15      $Z_1 = 0$;
16      $c_2 = 0$;
17  end if
18  $S_1 = T_1 - Z_1$
19  $S_2 = c_2$
```

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. You are to 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. Use the same switches, lights, and displays as in part V. Compile your circuit.
2. Use the Quartus II RTL Viewer tool to examine the circuit produced by compiling your Verilog code. Compare the circuit to the one you designed in Part V.
3. Download your circuit onto the DE2 board and test it by trying different values for numbers $A_1 A_0$ and $B_1 B_0$.


b_3 a_3 c_3
b_2 a_2 c_2
b_1 a_1 c_1
b_0 a_0 c_{in}
FA
FA
FA
FA
c_{out} s_3
s_2
s_1
s_0