ECE327 Digital System Design
Lab 1: BCD conversion and Adder

Lab Overview: The purpose of this exercise is to design combinatorial circuits that can perform binary-to-decimal number conversion (Part I). In Part II of this lab, you will create a ripple carry adder. Finally, in Part III of this lab you will use the converters from Part I and the adder from part II in a hierarchical design to form a binary-coded-decimal (BCD) adder. The requirements for this lab consist of completing the QUARTUS designs, submitting VHDL source files to the assign server as assignment 1, and completing the laboratory report using the LaTeX template provided in the lab manual.

Submission Instructions: Please follow all instructions given in the "Lab Report Instructions" and "Assign Server Instructions" in the lab manual. See Announcements for due dates and times.

Background: The DE1-series board provides 10 toggle switches, called SW_{9-0}, that can be used as inputs to a circuit, and 10 red lights, called $LEDR_{9-0}$, that can be used to display output values. Since there are 10 switches and lights, it is convenient to represent them as arrays (STD_LOGIC_VECTOR) in the VHDL code.

The DE1-series board has hardwired connections between its FPGA chip and the switches and lights. To use SW_{9-0} and $LEDR_{9-0}$ it is necessary to include in your Quartus project the correct pin assignments, which are given in the *DE1-series User Manual*. For example, on the DE1-SoC board, SW_0 is connected to the FPGA pin *AB12* and $LEDR_0$ is connected to pin *V16*. A good way to make the required pin assignments is to import into the Quartus II software the file called *DE1.qsf* for the DE1-SoC board, which is provided on Canvas. The procedure for making pin assignments is described in the tutorial Quartus Introduction using VHDL Design, which is also available from Intel.

It is important to realize that the pin assignments in the .qsf file are useful only if the pin names given in the file are exactly the same as the port names used in your VHDL entity. The file uses the names *SW[0] : : : SW[9]* and *LEDR[0] : : : LEDR[9]* for the switches and lights (note that the Quartus II software uses [ ] square brackets for array elements, while the VHDL syntax uses ( ) round brackets).

Part I: You are to design a circuit that converts a four-bit binary number $V = v_3v_2v_1v_0$ into its hardware representation, $M = m_4m_3m_2m_1m_0$. Table 1 shows the required output values as well as the decimal representation used in Part II. Your VHDL entity should have the four-bit input V (the binary number) and the output M (the BCD representation). M should consist of 4-bits representing the least significant BCD digit and another bit representing the upper BCD digit, since it can only take the values 0 and 1.

Binary Value (V)	Decimal Digits (D)		Hardware representation of BCD (M)	
0000	0	0	0	0000
0001	0	1	0	0001
0010	0	2	0	0010
…	…	…	…	…
1001	0	9	0	1001
1010	1	0	1	0000
1011	1	1	1	0001

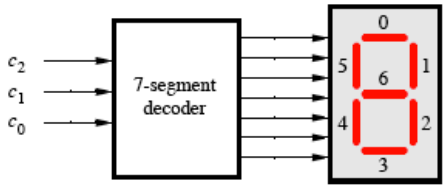
…	…	…	…	…
1111	1	5	1	0101

Simulate with a test bench. Then use switches SW_{3-0} on the DE1-SoC board to represent the binary number *V* and $LEDR_{4-0}$ to represent *M* to test your circuit on the DE1-SoC board.

Part II: Figure 8 shows a *7-segment decoder* module that has the three-bit input $c_2c_1c_0$. This decoder produces seven outputs that are used to display a character on a 7-segment display. The seven segments in the display are identified by the indices 0 to 6 shown in the figure below. Each segment is illuminated by driving it to the logic value 0.

You are to design a decoder that accepts *M* as an input and outputs the appropriate value to two 7-segment displays. The expected values are shown in the table in Part I. Note you will then have more than a 3-bit input and more than 1 vector output due to the BCD representation of your values.

Simulate your design with a testbench. Then use SW_{4-0} as *M* and HEX1 and HEX0 as the two 7-segment displays to test your circuit on the DE1-SoC board.

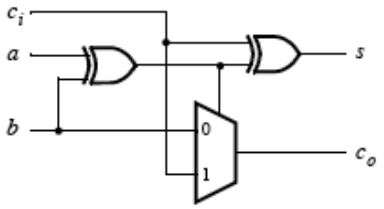


Part III: The figure below shows a circuit for a *full adder*, which has the inputs *a*, *b*, and c_i, and produces the outputs *s* and c_o. Parts *b* and *c* of the figure show a circuit symbol and truth table for the full adder, which produces the two-bit binary sum $c_o s = a + b + c_i$. Figure 2d shows how four instances of this full adder entity can be used to design a circuit that adds two four-bit numbers. This type of circuit is usually called a *ripple-carry* adder (we will discuss this in more detail later in the course), because of the way that the carry signals are passed from one full adder to the next.

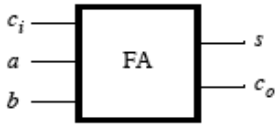
Write VHDL code that implements a full adder and write a top-level VHDL entity that instantiates four instances of this full adder.

Use functional simulations to verify the correct operation of your adder. Test your circuit by trying different values for numbers *A*, *B*, and c_{in} in simulation. As we have discussed in class, make sure you use both appropriate and enough test cases to prove that your design works. **In your report, discuss the test cases that you have chosen and justify that they are sufficient.**

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 LED lights, and connect the outputs of the adder, c_{out} and *S*, to the corresponding LED lights to test on the DE1-SoC board.



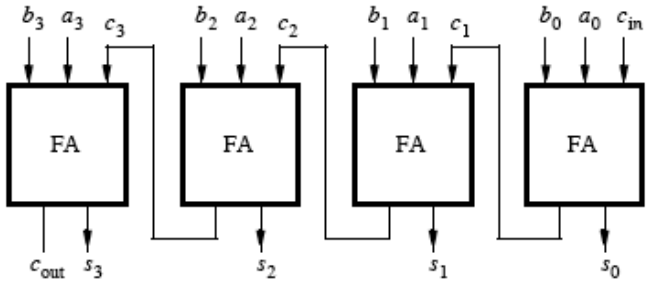
a) Full adder circuit



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

Part IV: In Part I you created a BCD converter. It is often useful to build circuits that use this method of representing decimal numbers, in which each decimal digit is represented using four bits. 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_1S_0 and an error bit. Note that the largest sum that must be handled by this circuit is $S_1S_0 = 9 + 9 + 1 = 19$. If the sum exceeds this value, you should indicate this by setting the error bit.

In your design, 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_1S_0.

Simulate your circuit with a testbench. Then 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 four-bit sum, carryout, and possible error produced by the operation $A + B$ to the LEDs. Display the BCD values of *A*, *B*, and result S_1S_0 on the 7-segment displays.

Rubric	
Report - Proper format - All sections included - Valid images where applicable - Proper grammar, punctuation, and spelling	**50%** - 10% - 30% - 5% - 5%
Demo - Live Demonstration o Includes working code and answering questions from the TA - Comments o Thoughtful comments, not English translations of code	**40%** - 35% - 5%
Proper Assign Server Code Submission	**10%**