ECE 3270: Laboratory One

LAB 1: BCD CONVERSION AND ADDER

September 20, 2017

Rodrigo Ignacio Rojas Garcia
Clemson University
Department of Electrical and Computer Engineering
rrojas@clemson.edu

Abstract

Laboratory one focuses on creating modular VHDL entities to create a single design with multiple purposes. The laboratory was divided in three parts: BCD converter, Ripple-Carry Adder, and BCD Adder. The laboratory consisted on designing the BCD converted and Ripple-Carry Adder to be used in part three and create a BCD Adder. The outcome of the design was to be able to add two four binary number inputs and a single bit and display the sum of the inputs in a 7-segment display in a decimal value. The purpose of the laboratory was to introduce the student to designing software Quartus II, simulation software ModelSim, and programming language VHDL. first two parts were used to designed with the purpose of using them of part three.

1 Introduction

The first laboratory for ECE 3270, Digital Computing Design, had the purpose of introducing the student to the Quartus II and ModelSim software as well as the VHDL language. Laboratory One was divided in three parts. Part one consisted of designing a combinatorial circuit that could perform binary-to-decimal number conversion. Part two consisted of creating a ripple carry adder which consists of adding two four bit binary numbers using multiple Full Adders. Part three consisted of using the hierarchical designs of section one and two to create a binary-coded-decimal adder with the purpose of performing the binary-to-decimal conversion of two four bit binary numbers and obtain the sum of the two.

2 Design of a Binary-to-Decimal Converter

2.1 Binary-to-Decimal Converter

In part one of the laboratory, the design of a Binary-to-Decimal (BCD) number converter was to be created. A BCD converter consists of converting a four bit binary number and converting it to its decimal equivalent. The outcome of the BCD conversion are two outputs, a single bit whose value could be either a 0 or a 1 and a four bit binary number whose value could be between 0 and 9. After the conversion had been processed, the decimal equivalent, whose value was to be between 0 and 15, was to be displayed in two 7-segment displays. The four bit binary number was to be connected to switches SW3-SW0 on the board. The decimal equivalent would be displayed on 7-segment displays HEX1 and HEX0 on the board. The design of a BCD was approached in three steps. First, the Boolean logic of the five bit binary number was calculated. The calculation was made by creating five Karnaugh maps[1](Figures 1-5) based on the desired output of the single bit and four bit binary number from five Truth Tables (Tables 2-6) which were based from the Table provided on laboratory one instruction manual (Table 1). Second, the values of the 7-segment displays were calculated. The calculation was made possible by referring to the 7-segment display pin out from the DE1_SoC manual (Figure 6)[3] and choosing what segments were to be turned on decimal values 0 through 9. Third, software Quartus II was used to design a top level and lower level entity. The BCD converter was made a top-level VHDL entity, which was composed of one input named "V" which would store the four bit binary number. Also, it was composed of two outputs named "HEX1" and "HEX0" which displayed the decimal equivalent of the BCD conversion. The architecture of the BCD converter consisted of an internal signal named "BCDConversion" and a single component called "sevenSegmentDisplay". The purpose of the internal signal was to store the result of the five Boolean logic equations obtained from the Karnaugh Maps[1] with the purpose of converting the four bit binary number input to its decimal equivalent. The component "sevenSegmentDisplay" is a lower VHDL entity which is composed of one input named "M" and two outputs named "HEX1" and "HEX0". This entity creates a process based on its input which corresponds to the BCD conversion of the four bit binary number and calculates what value to store on its two outputs which correspond to

the two 7-segment displays by using case statements (Figure 8). The 7-segment displays contain 7 inputs which can be either given a value of 0 to light up a segment of the display or a 1 to turn it off. The possible values given to the 7-segment display can be seen on Figure 7. The top VHDL entity uses the "sevenSegmentDisplay" by mapping the internal signal "BCDConversion" to the component's input and by mapping its two outputs to the component's output.

2.1.1 Tables and Figures

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

Table 1: BCD Table from Laboratory 1 Instructions

A	B	C	D	F(ABCD)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Table 2: Truth Table for fourth bit

		AB			
		00	01	11	10
CD	00	0	0	1	0
	01	0	0	1	0
	11	0	0	1	1
	10	0	0	1	1

F(ABCD)= A B + A C

Figure 1: Karnaugh Map for fourth bit and its Boolean Logic

A	B	C	D	F(ABCD)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Table 3: Truth Table for third bit

		AB			
		00	01	11	10
CD	00	0	0	0	1
	01	0	0	0	1
	11	0	0	0	0
	10	0	0	0	0

$$F(ABCD) = A\,\overline{B}\,\overline{C}$$

Figure 2: Karnaugh Map for third bit and its Boolean Logic

A	B	C	D	F(ABCD)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Table 4: Truth Table for second bit

		AB			
		00	01	11	10
CD	00	0	1	0	0
	01	0	1	0	0
	11	0	1	1	0
	10	0	1	1	0

$$F(ABCD) = \overline{A}\,B + B\,C$$

Figure 3: Karnaugh Map for second bit and its Boolean Logic

A	B	C	D	F(ABCD)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Table 5: Truth Table for first bit

		AB			
		00	01	11	10
CD	00	0	0	1	0
	01	0	0	1	0
	11	1	1	0	0
	10	1	1	0	0

$$F(ABCD) = \overline{A}\,C + A\,B\,\overline{C}$$

Figure 4: Karnaugh Map for first bit and its Boolean Logic

A	B	C	D	F(ABCD)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Table 6: Truth Table for zeroth bit

		AB			
		00	01	11	10
CD	00	0	0	0	0
	01	1	1	1	1
	11	1	1	1	1
	10	0	0	0	0

F(ABCD)= D

Figure 5: Karnaugh Map for zeroth bit and it Boolean Logic

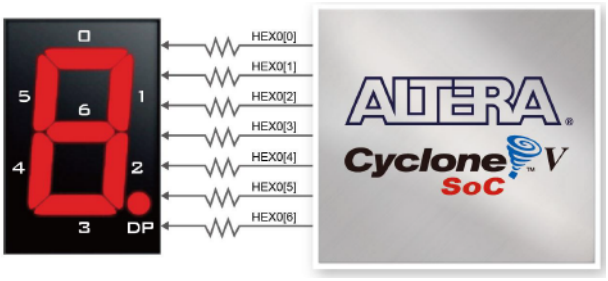

Figure 6: 7-Segment Display Pins

Four Bit Binary Number Input	BCD Conversion Value	HEX1 7-Segment Display Value	HEX0 7-Segment Display
0000	00000	1000000	1000000
0001	00001	1000000	1111001
0010	00010	1000000	0100100
0011	00011	1000000	0110000
0100	00100	1000000	0011001
0101	00101	1000000	0010010
0110	00110	1000000	0000010
0111	00111	1000000	1111000
1000	01000	1000000	0000000
1001	01001	1000000	0011000
1010	10000	1111001	1000000
1011	10001	1111001	1111001
1100	10010	1111001	0100100
1101	10011	1111001	0110000
1110	10100	1111001	0011001
1111	10101	1111001	0010010

Table 7: Input, BCD, and 7-Segment Display Values

```
PROCESS(M)
   BEGIN
      CASE M IS
         WHEN "00000" => HEX1 <= "1000000"; HEX0 <= "1000000";
         WHEN "00001" => HEX1 <= "1000000"; HEX0 <= "1111001";
         WHEN "00010" => HEX1 <= "1000000"; HEX0 <= "0100100";
         WHEN "00011" => HEX1 <= "1000000"; HEX0 <= "0110000";
         WHEN "00100" => HEX1 <= "1000000"; HEX0 <= "0011001";
         WHEN "00101" => HEX1 <= "1000000"; HEX0 <= "0010010";
         WHEN "00110" => HEX1 <= "1000000"; HEX0 <= "0000010";
         WHEN "00111" => HEX1 <= "1000000"; HEX0 <= "1111000";
         WHEN "01000" => HEX1 <= "1000000"; HEX0 <= "0000000";
         WHEN "01001" => HEX1 <= "1000000"; HEX0 <= "0011000";
         WHEN "10000" => HEX1 <= "1111001"; HEX0 <= "1000000";
         WHEN "10001" => HEX1 <= "1111001"; HEX0 <= "1111001";
         WHEN "10010" => HEX1 <= "1111001"; HEX0 <= "0100100";
         WHEN "10011" => HEX1 <= "1111001"; HEX0 <= "0110000";
         WHEN "10100" => HEX1 <= "1111001"; HEX0 <= "0011001";
         WHEN "10101" => HEX1 <= "1111001"; HEX0 <= "0010010";
         WHEN OTHERS => HEX1 <= "1111111"; HEX0 <= "1111111";
      END CASE;
   END PROCESS;
```

Figure 7: Case Statements

2.2 Simulation of Binary-To-Decimal Converter

After the design of the BCD and the 7-segment display entities, simulation was performed on the software ModelSim to test six different cases to confirm that the requirement of converting the four bit binary value to a decimal value and display the output in two 7-segment displays were met. Each test test consisted of entering different four bit binary values into variable "V" which corresponds to pins SW3-SW0 and checking that the outputs "HEX1" and "HEX0" displayed the correct decimal equivalent value by comparing their outputs to the values on Table 7.

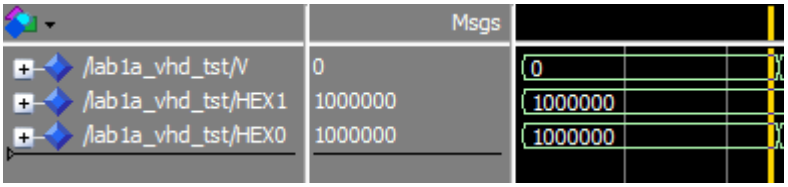

Figure 8: BCD Test Case 1

In the first test case, input "V" is given the value of "0000" (0 decimal) which is expected to convert it to value "00000". The conversion value is expected to be checked by the case statements and assign the values "1000000" and "1000000" to outputs "HEX1" and "HEX0" correspondingly. It can be observed that the outputs values of "HEX1" and "HEX0" are correct by comparing them with Table 7

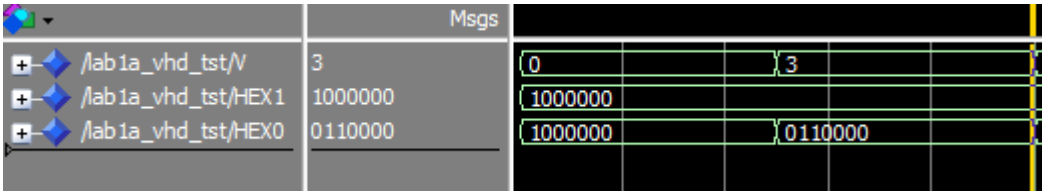

Figure 9: BCD Test Case 2

In the second test case, input "V" is given the value of "0011" (3 decimal) which is expected to convert it to value "00011". The conversion value is expected to be checked by the case statements and assign the values "1000000" and "0110000" to outputs "HEX1" and "HEX0" correspondingly. It can be observed that the outputs values of "HEX1" and "HEX0" are correct by comparing them with Table 7.

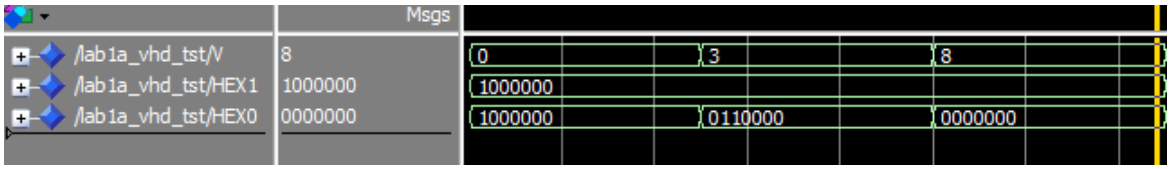

Figure 10: BCD Test Case 3

In the third test case, input "V" is given the value of "1000" (8 decimal) which is expected to convert it to value "01000". The conversion value is expected to be checked by the case statements and assign the values "1000000" and "0000000" to outputs "HEX1" and "HEX0" correspondingly. It can be observed that the outputs values of "HEX1" and "HEX0" are correct by comparing them with Table 7.

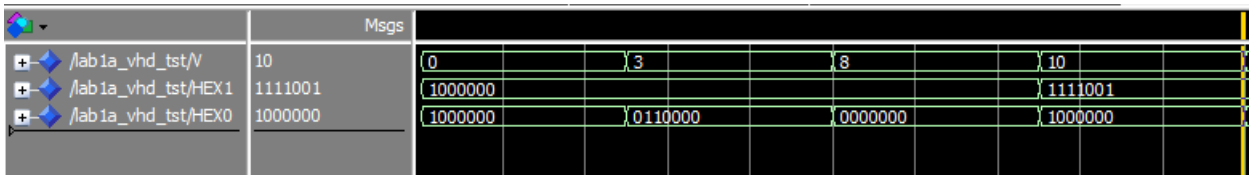

Figure 11: BCD Test Case 4

In the fourth test case, input "V" is given the value of "1010" (10 decimal) which is expected to convert it to value "01000". The conversion value is expected to be checked by the case statements and assign the values "1111001" and "1000000" to outputs "HEX1" and "HEX0" correspondingly. It can be observed that the outputs values of "HEX1" and "HEX0" are correct by comparing them with Table 7.

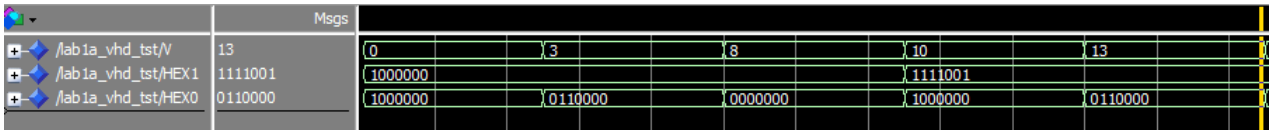

Figure 12: BCD Test Case 5

In the fifth test case, input "V" is given the value of "1101" (13 decimal) which is expected to convert it to value "10011". The conversion value is expected to be checked by the case statements and assign the values "1111001" and "0110000" to outputs "HEX1" and "HEX0" correspondingly. It can be observed that the outputs values of "HEX1" and "HEX0" are correct by comparing them with Table 7.

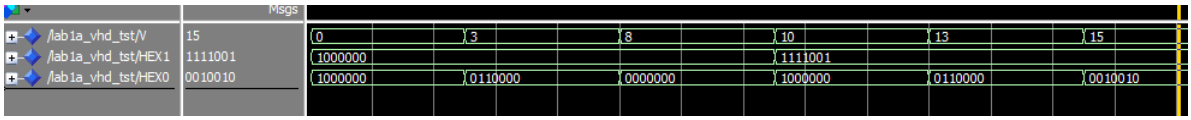

Figure 13: BCD Test Case 6

In the sixth test case, input "V" is given the value of "1111" (13 decimal) which is expected to convert it to value "10101". The conversion value is expected to be checked by the case statements and assign the values "1111001" and "0010010" to outputs "HEX1" and "HEX0" correspondingly. It can be observed that the outputs values of "HEX1" and "HEX0" are correct by comparing them with Table 7.

3 Design of a Ripple-Carry Adder

3.1 Full Adder

In part two of laboratory one, the design of a Ripple-Carry adder was to be created. Before this could be accomplished, the design of a Full Adder must occur first because a Ripple-Carry adder is a combination of Full Adders. A Full Adder is a circuit which performs addition of three one-bit binary numbers, two operands and a carry bit. The Full Adder outputs a sum and a carry bit.[**?**] The design of a Full Adder was approached in two steps. First, the Boolean logic of the sum and carry bit outputs was calculated. The calculation was made by creating two Karnaugh maps[1] (Figures 14-15) based on the desired output of the sum and carry bit from the Truth table provided on the laboratory one instruction manual (Table 8). Second, the Full Adder circuit was designed through the software Quartus II. The Full Adder was made a lower-level VHDL entity which was composed of three inputs, two operands named "A" and "B" and a carry bit named "Cin". The entity was also composed of two outputs, sum and carry bit which were named "Sum" and "Cout" accordingly. Both outputs, "Sum" and "Cout", were assigned their values based on the Boolean logic obtained from the Karnaugh maps[1].

3.2 Ripple-Carry Adder

After the design of a Full Adder, the design of a Ripple-Carry Adder could be performed. The Ripple-Carry adder consists of a combination of Full Adder circuits in which the carry bit of each Full Adder is carried on the succeeding most significant Full Adder.[2] The specification of laboratory one regarding the Ripple-Carry adder consisted on connecting four Full Adders with the purpose of performing the addition of two four bit binary numbers and a carry bit. The two four bit binary numbers could be of value 15 through 0 and carry bit could be of value 0 or 1. The sum of these inputs would produce five outputs which correspond to a five bit binary number of value between 31 and 0. The input of both four bit binary numbers and carry bit were to be connected to switches SW8-SW0 on the board as well as their corresponding LED's 8 through 0 (Figure 16). The sum of these three inputs would be displayed on LED's 4 through 0 only if switch SW9 was turned on. The Ripple-Carry adder was designed through the software Quartus II. The design was made a top-level VHDL entity which was composed of four inputs, two four bit binary numbers named "A" and "B", carry bit named "Cin", and trigger switch named "triggerSwitch". Also, it was composed of one output named "SWLEDS", which was connected to LED's 9 through 0, which displayed the value of the three inputs in binary or displayed the addition of the three inputs in binary. The architecture of this entity used four components of a Full Adder. The four Full Adder components were connected to each other through three internal signals, "Carry1", "Carry2", and "Carry3". These three internal signals were used to store the output carry bit of the Full Adder and use it as input for the Full Adder next to it. The sum of the Ripple-Carry adder gave five outputs. The five outputs were stored on an internal signal called "tempResult". After all calculations were made, a process was made to check if any of the values of inputs "triggerSwitch", "Cin", "A", or "B" had been changed. The purpose of the process was to analyze if the value of the inputs "Cin", "A", and "B" had to be displayed in binary or the sum of these three inputs had to be displayed in binary on LED's 4 through 0. For the process to work properly, and internal variable named "finalResult" was used. The reason of using a variable instead of a signal was that the value of variables change instantly. After variable "finalResult" was given its value based if the triggerSwitch was turned on or not, it stored its value to LED's 9 through 0.

3.2.1 Tables and Figures

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

Table 8: Full Adder Truth Tablet

		AB			
		00	01	11	10
C	0	0	0	1	0
	1	0	1	1	1

F(ABC)= B C + A C + A B

Figure 14: Carry bit Karnaugh Map

		AB			
		00	01	11	10
C	0	0	1	0	1
	1	1	0	1	0

F(ABC)= $\overline{A}$ $\overline{B}$ C + $\overline{A}$ B $\overline{C}$ + A B C + A $\overline{B}$ $\overline{C}$

Figure 15: Sum Karnaugh Map

Figure 16: LED Pins

3.3 Simulation of Ripple-Carry Adder

After the completion of the Full Adder and Ripple-Carry adder designs, simulation was performed through software, ModelSim. The purpose of the simulation was to confirm that the requirements of part two of the laboratory were met. The simulation consisted on seven cases. Each test case consisted of entering different four bit binary values into variables "A" and "B" as well as different value for single bit inputs "Cin" and "triggerSwitch".

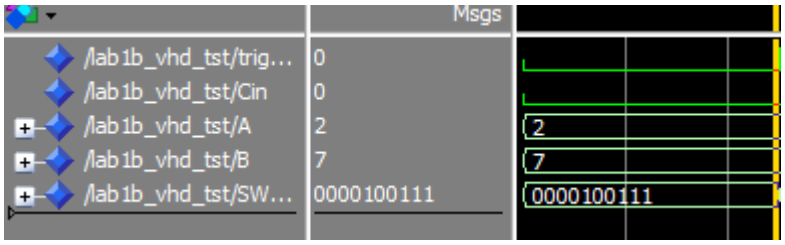

Figure 17: Case 1

In the first test case, inputs "triggerSwitch" and "Cin" were given values of "0" and inputs "A" and "B" were given values of "0010" and "0111" correspondingly. The purpose of this test case was to demonstrate that the binary values of inputs "A" and "B" are displayed on LED's 7 through 0.

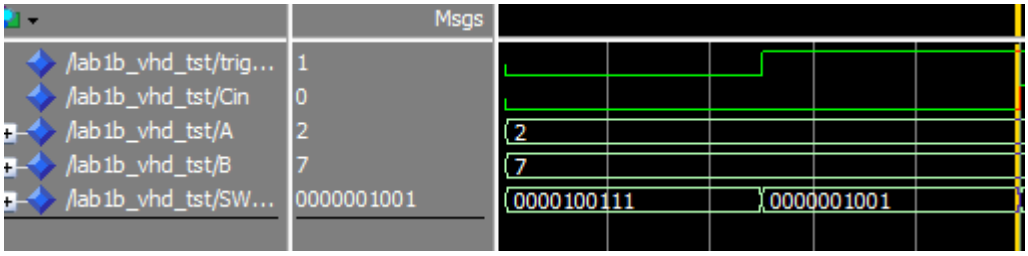

Figure 18: Case 2

In the second test case, inputs "triggerSwitch" and "Cin" were given values of "1" and "0" correspondingly. The inputs "A" and "B" were given values of "0010" and "0111" correspondingly. The expected sum of the inputs is to be 9. The purpose of this test case was to demonstrate that the sum of inputs "A" and "B" was correct which is expected to be value of 9 and to demonstrate that the sum is displayed on LED's 4 through 0.

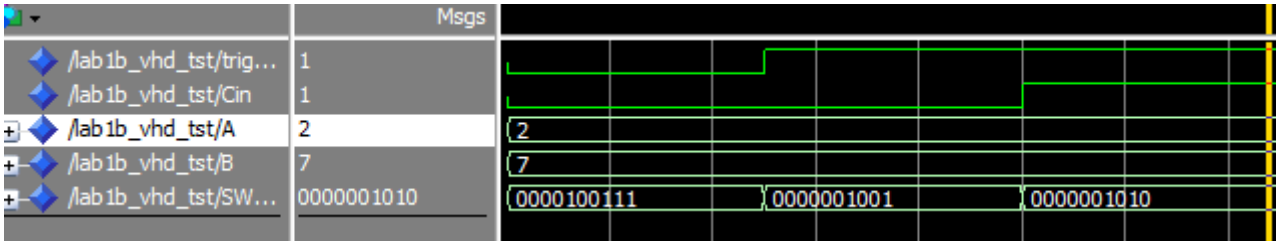

Figure 19: Case 3

In the third test case, inputs "triggerSwitch" and "Cin" were given values of "1". The inputs "A" and "B" were given values of "0010" and "0111" correspondingly. The expected sum of the inputs is to be 10. The purpose of this test case was to demonstrate that the sum of inputs "A", "B", and "Cin" was correct and to demonstrate that the sum is displayed on LED's 4 through 0.

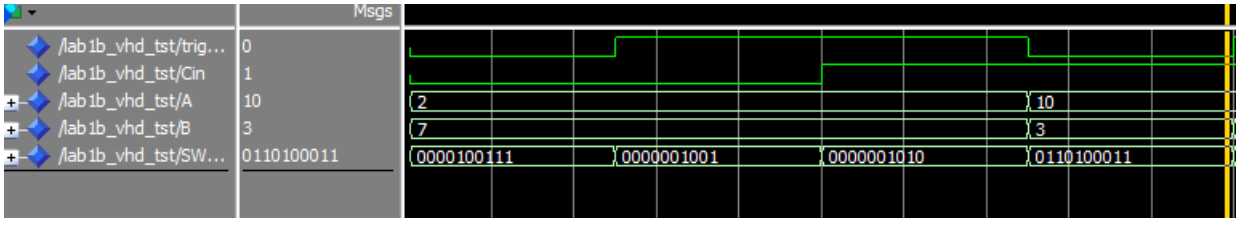

Figure 20: Case 4

In the fourth test case, inputs "triggerSwitch" and "Cin" were given values of "0" and "1" correspondingly. The inputs "A" and "B" were given values of "1010" and "0011" correspondingly. The expected sum of the inputs is to be 14. The purpose of this

test case was to demonstrate that the binary value of inputs "A", "B", and "Cin" are displayed on LED's 8 through 0.

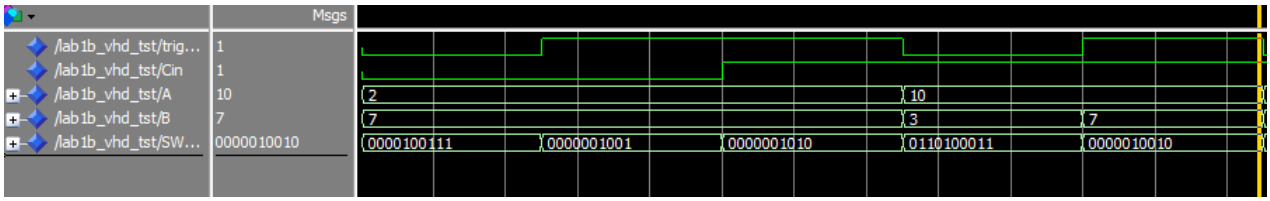

Figure 21: Case 5

In the fifth test case, inputs "triggerSwitch" and "Cin" were given values of "1". The inputs "A" and "B" were given values of "1010" and "0111" correspondingly. The expected sum of the inputs is to be 18. The purpose of this test case was to demonstrate that the sum of inputs "A", "B", and "Cin" are displayed on LED's 4 through 0 when input "A" is a double digit number.

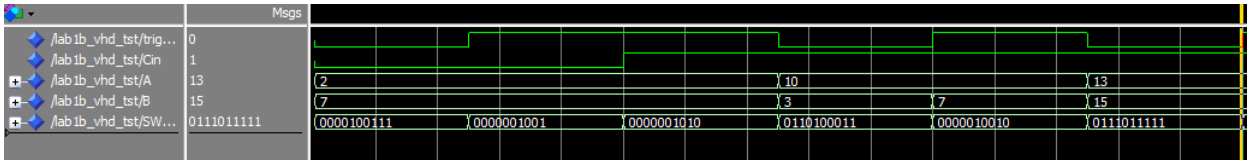

Figure 22: Case 6

In the sixth test case, inputs "triggerSwitch" and "Cin" were given values of "0" and "1" correspondingly. The inputs "A" and "B" were given values of "1101" and "1111" correspondingly. The expected sum of the inputs is to be 29. The purpose of this test case was to demonstrate that the binary value of inputs "A", "B", and "Cin" are displayed on LED's 8 through 0 when inputs "A" and "B" are double digit numbers.

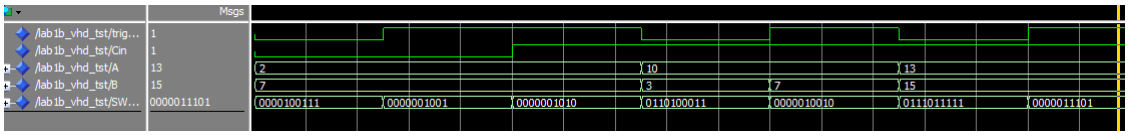

Figure 23: Case 7

In the seventh test case, inputs "triggerSwitch" and "Cin" were given value "1". The inputs "A" and "B" were given values of "1101" and "1111" correspondingly. The expected sum of the inputs is to be 29. The purpose of this test case was to demonstrate that the sum of inputs "A", "B", and "Cin" are displayed on LED's 4 through 0 when inputs "A" and "B" are double digit numbers.

4 Binary-to-Decimal Adder

In part three of the laboratory, the design of a Binary-to-Decimal (BCD) adder was to be created. The BCD Adder consists of the combination of the BCD converter and Ripple-Carry adder. The specification of laboratory one regarding the BCD Adder consisted on performing the addition of two four bit binary numbers and carry bit. The two four bit binary numbers could be of value 15 through 0 and carry bit could be of value 0 or 1. The sum of these inputs would produce five outputs which corresponded to a five bit binary number of value 31 to 0. The input of both four bit binary numbers and carry bit were to be connected to switches SW8-SW0 on the board as well as their corresponding LED's 8 though 0 (Figure 16) and 7-segment displays HEX5-HEX2. The sum of these three inputs would be displayed on 7-segment displays HEX1 and HEX0. The sum would be displayed on LED's 4 through 0 only if switch SW9 was turned on. A restriction on what to display was also specified. If either values of the two four bit binary number inputs were of value greater than 9, LED 9 was to be light up displaying this error and their number would not be displayed on the 7-segment display. Another restriction was that if the sum of the three inputs was of value greater than 19, the sum would not be displayed on the 7-segment displays and LED 9 would be light up. It should be noted that there was no restriction on the value of sum of inputs to be displayed on LED's 4 through 0 if switch SW9 was to be turned on. The design was made a top-level VHDL entity which was composed of four inputs, two four bit binary numbers named "A" and "B", carry bit named "Cin", and trigger switch named "triggerSwitch". Also, it was composed of seven outputs. One output corresponded to LED's 9 through 0, named "SWLEDS", which displayed the value of the three input in binary or displayed the addition of the three inputs in binary. The rest of the outputs were connected to 7-segment displays HEX5-HEX0 named "HEX5, HEX4 ...". The architecture of this entity used three different components: "FourBitBinary", "FourBitFullAdder", and "FiveBitBinary". The "FourBitBinary" component was used to convert the binary value of inputs "A" and "B" and display them on the 7-segment displays HEX5-4 and HEX3-2 correspondingly. The "FourBitFullAdder" component was used to add the three inputs "A", "B", and "Cin". The "FiveBitBinary" component was a new component which was made to perform the BCD conversion of the five bit binary number obtained from the addition in the "FourBitFullAdder" component. Table 9 contains all possible outputs of the conversion. At the end of the architecture, a process was created to analyze if the input "triggerSwitch" had been turned on so either the binary values of the inputs were to be displayed or the sum of the inputs were to be displayed on LED's. The process was also created to check if values of inputs "A" or "B" exceeded value of 9 or if the sum of the three inputs exceeded the value of 19 so it could be determined if LED 9 should be light up or not.

4.1 Tables and Figures

Binary Number Sum	BCD Conversion Value	HEX1 7-Segment Display	HEX0 7-Segment Display
0000	00000	1000000	1000000
0001	00001	1000000	1111001
0010	00010	1000000	0100100
0011	00011	1000000	0110000
0100	00100	1000000	0011001
0101	00101	1000000	0010010
0110	00110	1000000	0000010
0111	00111	1000000	1111000
1000	01000	1000000	0000000
1001	01001	1000000	0011000
1010	10000	1111001	1000000
1011	10001	1111001	1111001
1100	10010	1111001	0100100
1101	10011	1111001	0110000
1110	10100	1111001	0011001
1111	10110	1111001	0010010
10000	10111	1111001	0000010
10001	11000	1111001	1111000
10010	11001	1111001	0000000
10011	11010	1111001	0011000

Table 9: BCD Conversion 5-bit

4.2 Simulation of Binary-to-Decimal Adder

After the completion of the BCD Adder design, simulation was performed through software, ModelSim. The purpose of the simulation was to confirm that the requirements of part three of the laboratory were met. The simulation consisted on eight cases. Each test case consisted of entering different four bit binary values into variables "A" and "B" as well as different value for single bit inputs "Cin" and "triggerSwitch".

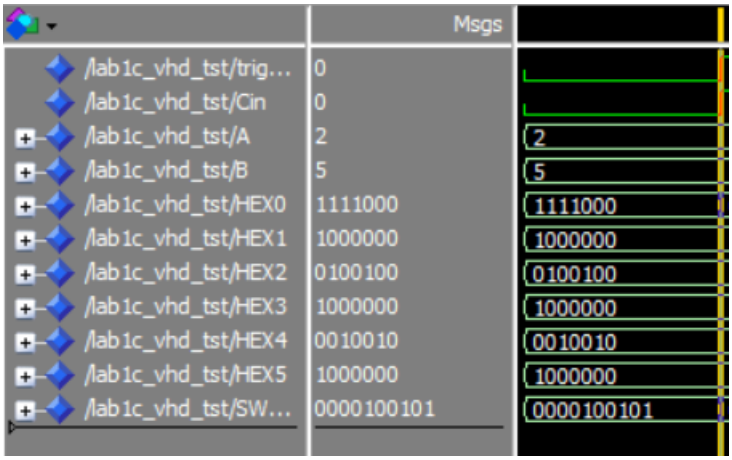

Figure 24: BCD Adder Case 1

In the first test case, inputs "triggerSwitch" and "Cin" were given values of "0". The inputs "A" and "B" were given values of "0010" and "0101" correspondingly. The expected sum of the inputs is to be 7. The purpose of this test case was to demonstrate that the binary value of inputs "A" and "B" are displayed on LED's 7 through 0 and their corresponding 7-segment displays HEX5-0 light up the correct segments when comparing values and Table 8.

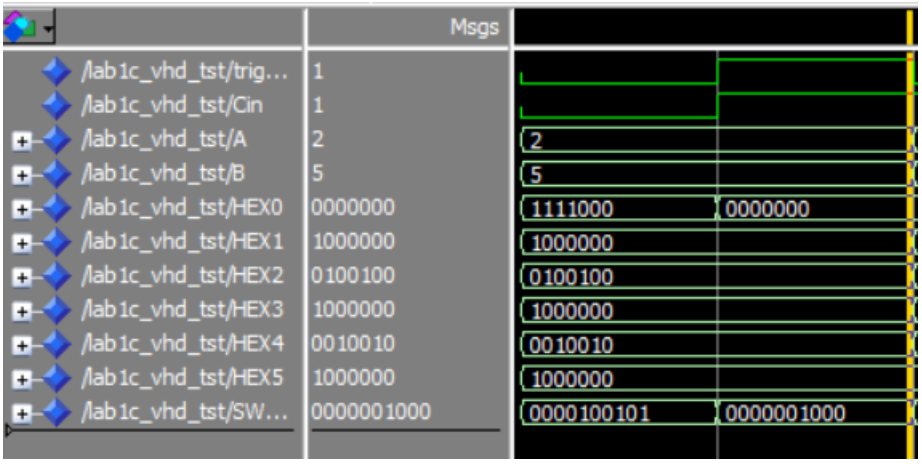

Figure 25: BCD Adder Case 2

In the second test case, inputs "triggerSwitch" and "Cin" were given values of "1". The inputs "A" and "B" were given values of "0010" and "0101" correspondingly. The expected sum of the inputs is to be 8. The purpose of this test case was to demonstrate that the sum of inputs "A", "B", and "Cin" are displayed on LED's 4 through 0 and their corresponding 7-segment displays HEX5-0 light up the correct segments when comparing values and Table 8.

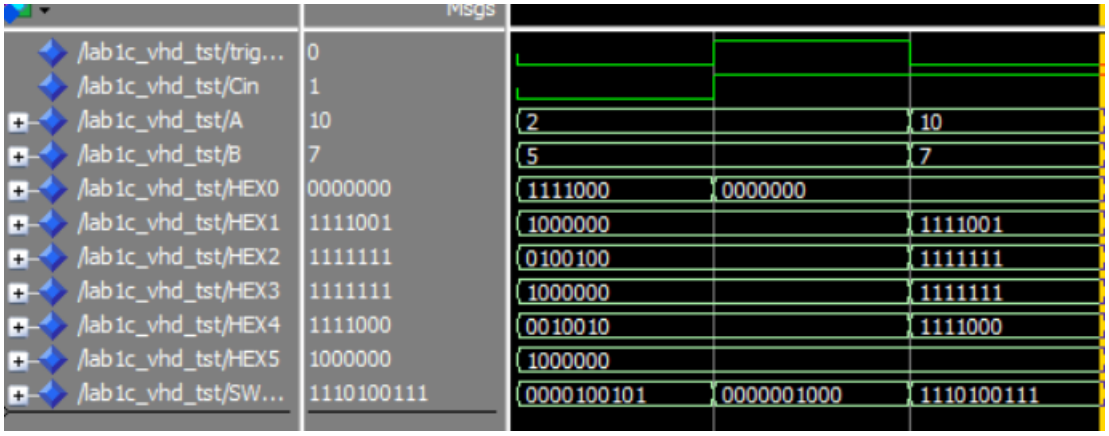

Figure 26: BCD Adder Case 3

In the third test case, inputs "triggerSwitch" and "Cin" were given values of "0" and "1" correspondingly. The inputs "A" and "B" were given values of "1010" and "0111" correspondingly. The expected sum of the inputs is to be 18. The purpose of this test case was to demonstrate that the binary value of inputs "A", "B", and "Cin" are displayed on LED's 8 through 0 and 7-segment display for "A" does not light up since value is greater than 9. Also, it is made to check that LED 9 lights up since input "A" is greater than 9.

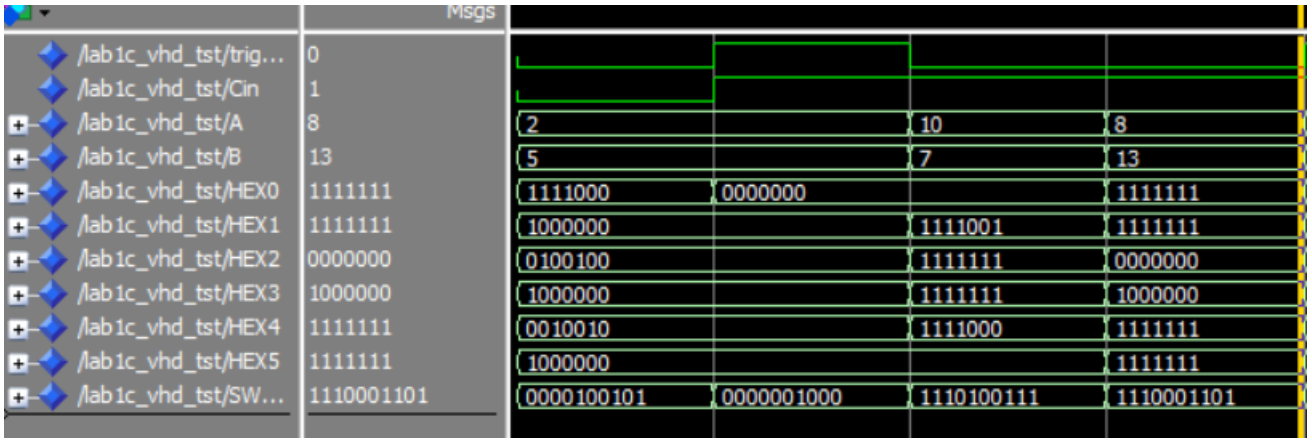

Figure 27: BCD Adder Case 4

In the fourth test case, inputs "triggerSwitch" and "Cin" were given values of "0" and "1" correspondingly. The inputs "A" and "B" were given values of "1000" and "1101" correspondingly. The expected sum of the inputs is to be 22. The purpose of this test case was to demonstrate that the binary value of inputs "A", "B", and "Cin" are displayed on LED's 8 through 0 and 7-segment displays for "B" and the addition does not light up since value is greater than 9 and 19 correspondingly. Also, it is made to check that LED 9 lights up since input "B" is greater than 9 and addition is greater than 19.

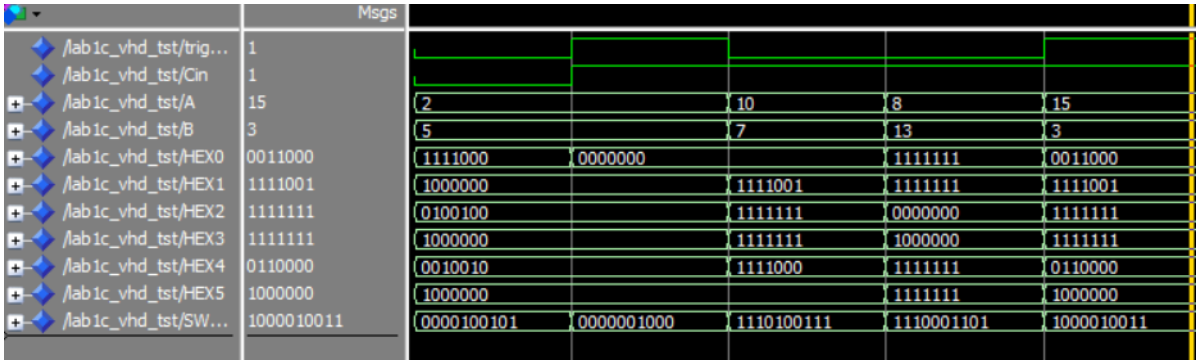

Figure 28: BCD Adder Case 5

In the fifth test case, inputs "triggerSwitch" and "Cin" were given values of "1" and "1" correspondingly. The inputs "A" and "B" were given values of "1111" and "0011" correspondingly. The expected sum of the inputs is to be 19. The purpose of this test case was to demonstrate that the addition of inputs "A", "B", and "Cin" are displayed on LED's 4 through 0 and 7-segment displays for "A" does not light up since value is greater than 9. Also, it is made to check that LED 9 lights up since input "A" is greater than 9.

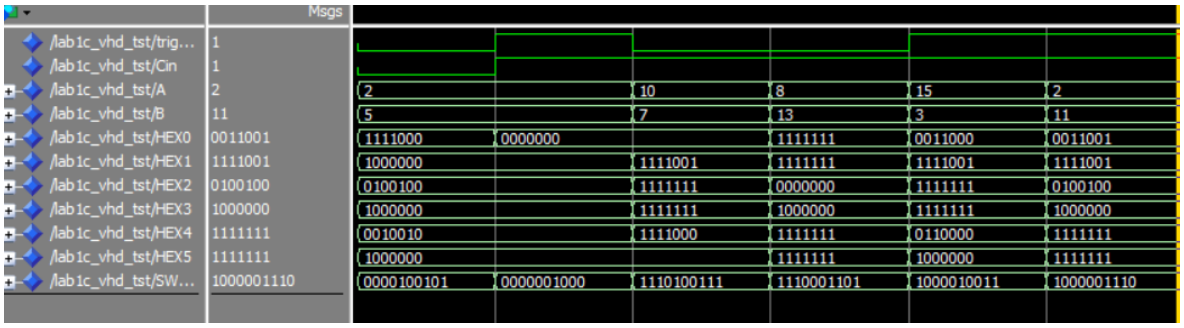

Figure 29: BCD Adder Case 6

In the sixth test case, inputs "triggerSwitch" and "Cin" were given values of "1" and "1" correspondingly. The inputs "A" and "B" were given values of "0010" and "1011" correspondingly. The expected sum of the inputs is to be 14. The purpose of this test case was to demonstrate that the addition of inputs "A", "B", and "Cin" are displayed on LED's 4 through 0 and 7-segment displays for "B" does not light up since value is greater than 9. Also, it is made to check that LED 9 lights up since input "B" is greater than 9.

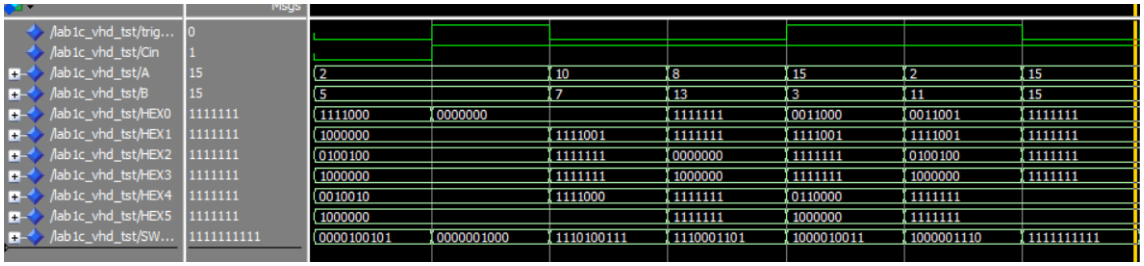

Figure 30: BCD Adder Case 7

In the seventh test case, inputs "triggerSwitch" and "Cin" were given values of "0" and "1" correspondingly. The inputs "A" and "B" were given values of "1111" and "1111" correspondingly. The expected sum of the inputs is to be 31. The purpose of this test case was to demonstrate that the binary value of inputs "A", "B", and "Cin" are displayed on LED's 8 through 0 and none of the 7-segment displays are light up. Also, it is made to check that LED 9 lights up since input "A" and "B" are greater than 9 and addition is greater than 19.

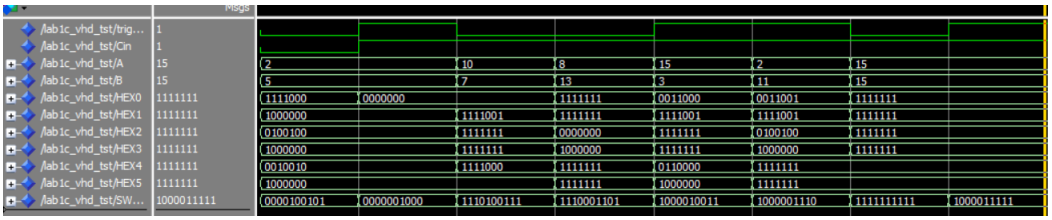

Figure 31: BCD Adder Case 8

In the eighth test case, inputs "triggerSwitch" and "Cin" were given values of "0" and "1" correspondingly. The inputs "A" and "B" were given values of "1111" and "1111" correspondingly. The expected sum of the inputs is to be 31. The purpose of this test case was to demonstrate that the addition value of inputs "A", "B", and "Cin" are displayed on LED's 4 through 0 and none of the 7-segment displays are light up. Also, it is made to check that LED 9 lights up since input "A" and "B" are greater than 9 and addition is greater than 19.

5 Conclusion

The purpose of laboratory one was to introduce the student to the VHDL programming language as well as design software Quartus II and simulation software ModelSim. The outcome of laboratory one taught the student how to create and design modular entities. The laboratory accomplished this by making the student use the BCD and Ripple-Carry adder components created on parts one and two of the laboratory to create a BCD Adder on part three.

References

[1] Calpoly. *Karnaugh Map Solver*. Calpoly, 2017.

[2] Circuittoday. *Ripple Carry Adder*. Circuittoday, 2017.

[3] Terasic. $DE1_SoCManual.Terasic$, 2017.