Lab 1 HALF-Adder, FULL-Adder, 4-Bit-Adder

CSC 343 Fall 2017

October, 2, 2017

Jeter Gutierrez

TABLE OF CONTENTS

1. Objective.

In this lab we will use the instructions and the procedure described in the Master Tutorial in order to design a Half- Adder, Full-Adder using gates, Full-Adder using half adder as a component, and a 4 bit adder. The software used in order to make these circuits is Quartus. We will build the 4 circuits through software and verify validity through simulation in the software and then we will program the FPGA board and test that the circuit works accordingly. The board used to test the circuits is the DE1-SoC board. The circuits we will be designing are:

a. Half adder
b. Full-Adder using gates
c. Full-Adder using Half adder as a component
d. 4-Bit adder using 1 bit Full adder as a component.

2 Half Adder.

2.1 Functionality and Specifications for Half-Adder.

In this step we will be explaining and creating a Half-Adder circuit. A half adder circuit is a digital circuit that is created using two logical gates. It adds two binary numbers such as ab or xy and the output of the circuits are the carry and the total sum. In this example the gates that are used to create the circuit are an and2 gate and a xor 2 gate.

In the half adder truth table the inputs are the GUTI_A and GUTI_B. GUTI_C is the output value for the carry and GUTI_S is the output for the total sum.

Half-Adder Truth Table

GUTI_A	GUTI_B	GUTI_C	GUTI_S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

The boolean function that represents the half adder is.

$$Sum = \overline{A} \cdot B + A \cdot \overline{B}$$

When we apply the boolean function it requires an and gate and an or gate. Both of the inputs go into each of the gates. One of the gates, the and gates, outputs the value of the carry, and the other gate, the or gate outputs the total sum of the two one-bit binary numbers.

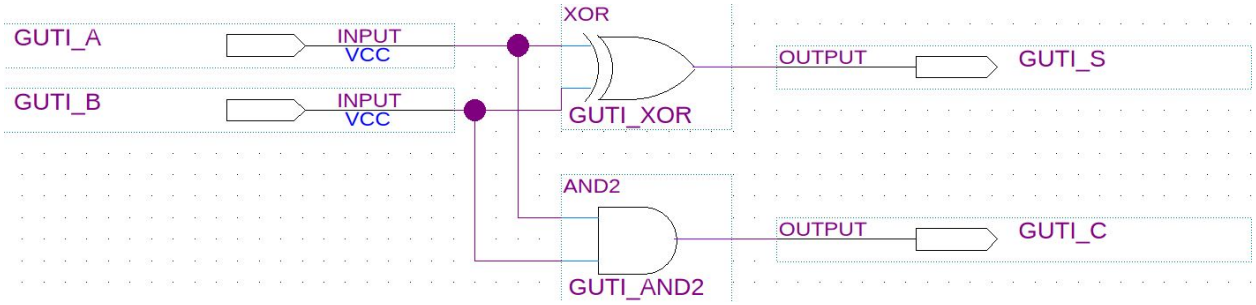

Figure 1: Block Diagram of Half-Adder

We can make the Half adder block diagram into a symbol in order to make using the circuit easier for future use.

Figure 2: Half-Adder Symbol.

2.2 Simulation for Half-Adder.

In the simulation the values for GUTI_A and GUTI_B will be given in different intervals of 0 and 1 and different combinations. We will then observe the output in order to make sure that the

simulation is associated with the truth table correctly. The input values will alternate by 160ns in this simulation. The simulation is read by columns from left to right, the first interval 0ns-160ns the input for GUTI_A and GUIT_B are both 0, and the ouput for GUTI_C and GUIT_S are both 0 just like in the truth table.

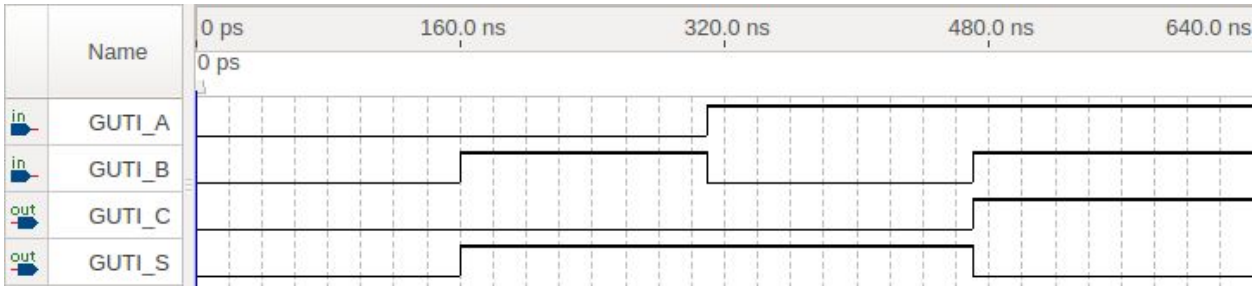

Figure 3: Vector waveform corresponding to Half-Adder.

We can notice that when both GUTI_A and GUTI_B are both 0 the output for both the sum and the carry value are both 0, and that when both inputs are set to 1, the output for the total sum is 0 but the carry is 1 because binary numbers can only have two outputs, 0 or 1, therefore the sum of 1 and 1 is not 2 but instead 0 along with a carry value of 1. Both of the two selected results along with the rest of the values in the simulation compare accordingly with the truth table and match up correctly.

2.3 Demonstration of Half-Adder on De1-SoC Board.

The inputs and outputs that are assigned to pins on the DE1-SoC board are:

GUTI_A is assigned to PIN_AB12 SW[0]

GUTI_B is assigned PIN_AC12 SW[1]

GUTI_C is assigned PIN_V16 LED[0]

GUTI_S is assigned PIN_W16 LED[1]

Figure 4: PIN assignments of the Half-Adder Circuit to the DE1-SoC board.

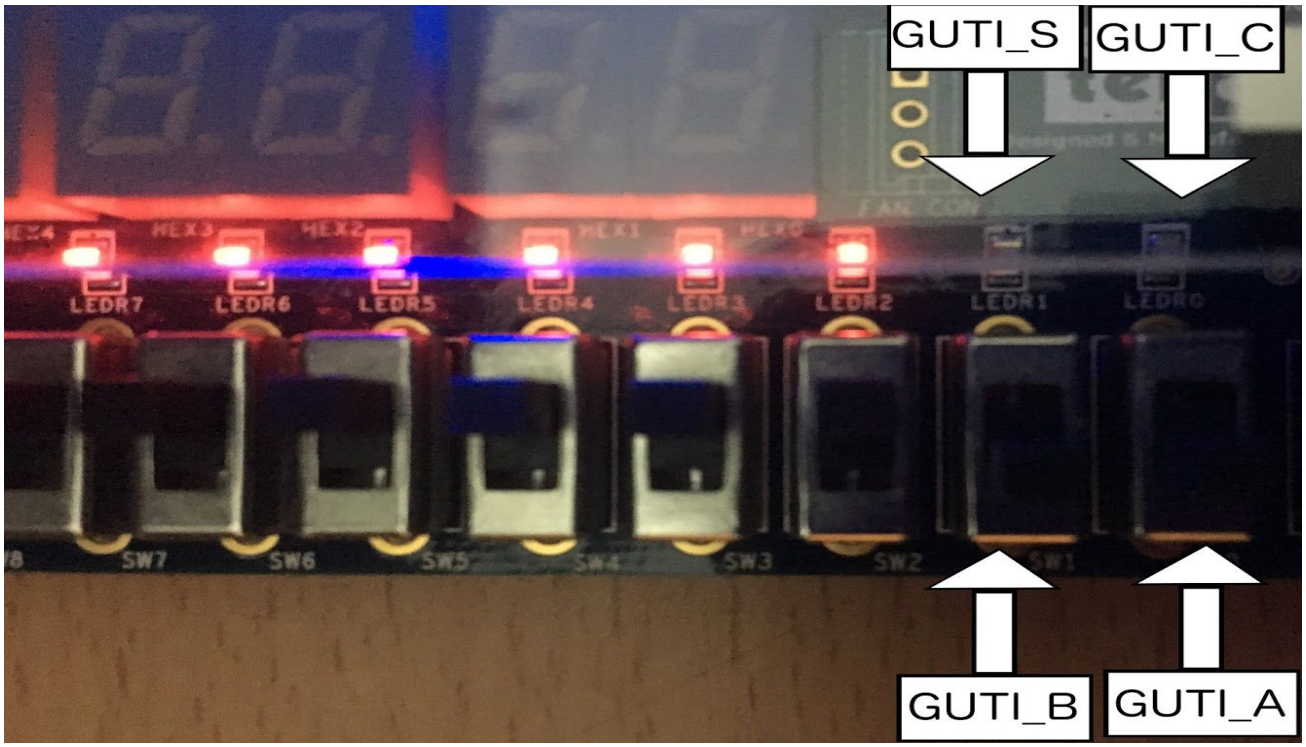

Figure 5: Digital Circuit of HAlf-Adder when GUTI_A is 0, and GUTI_B is also 0. Both of the outputs GUTI_S and GUTI_C are both also 0.

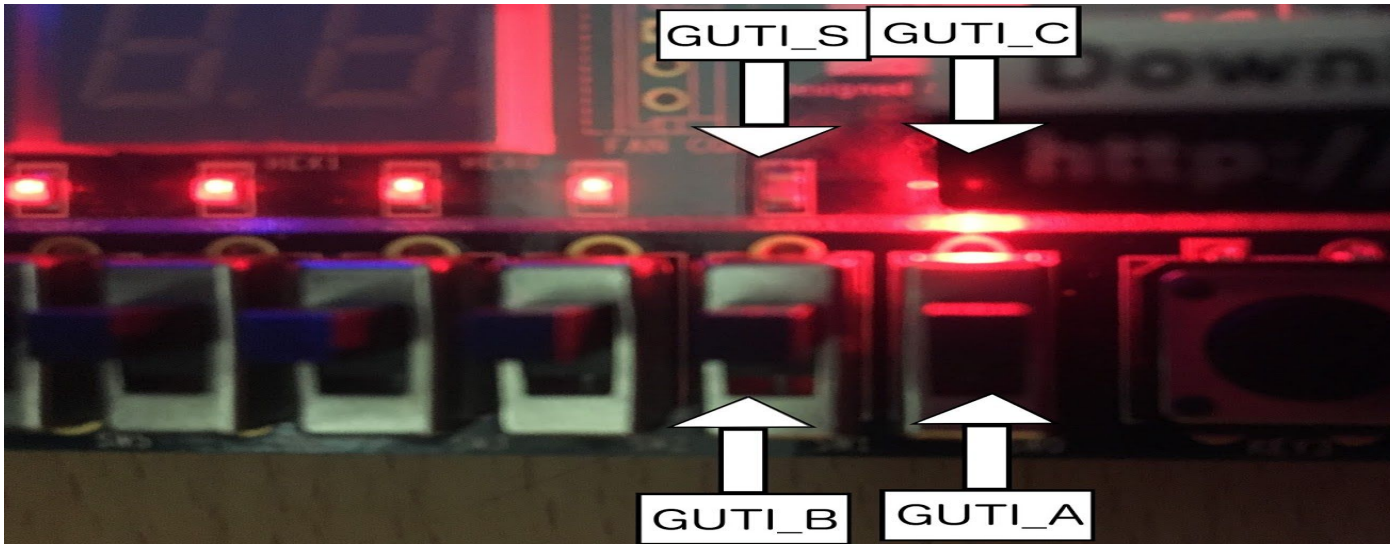

Figure 6: Digital Circuit of Half-Adder when GUTI_A is 1, and GUTI_B is also 1.Output GUTI_S is 0 and GUTI_C is 1.

3 Full-Adder Using Gates.

3.1 Functionality and Specifications Full-Adder Using Gates.

In this step we will be explaining and creating a Full-Adder circuit using gates. A full adder is a circuit that adds three one bit binary numbers instead of two of them and then outputs the sum and the carry value of the 3 binary numbers. It takes three numbers as the input but two of the numbers are numbers being added and the third is the carry that is being used. The design of the half adder using gates consists of 2 xor gates and two and2 gates and one or gate. Both of the inputs GUTI_A and GUTI_B go into an xor gate and an and2 gates, and the third xor gate outputs the total sum of the two binary numbers, and the or2 gate is output of the original carry value and the output from the or2 gate at the end of the circuit. The three input values are GUTI_A, GUTI_B and GUIT_IN_C and the two output values are GUTI_S and GUIT_C_OUT. GUTI_S is the total sum and GUTI_C_OUT is the final carry value.

Full-Adder using Gates Truth Table

GUTI_A	GUTI_B	GUTI_IN_C	GUTI_C_OUT	GUTI_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

The boolean function that represents the full adder is:

$$Sum = \overline{A}\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\overline{C}_{in} + ABC_{in}$$

The output using the boolean function gives the carry value of GUTI_C_OUT through an or gate and the total sum is output through a XOR gate.

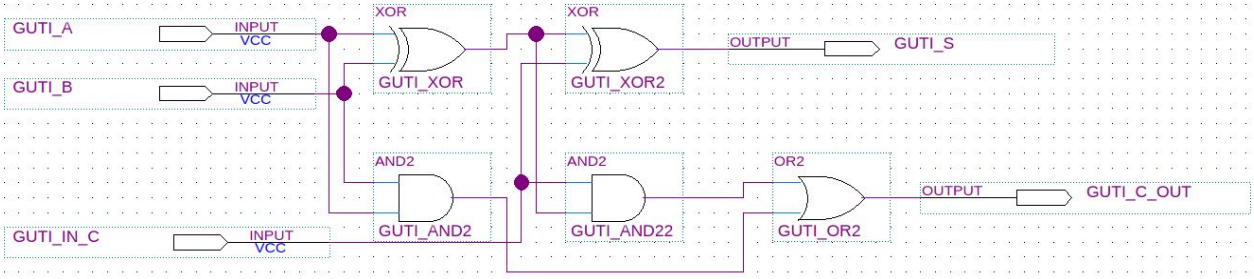

Figure 7:Block diagram of Full-Adder using Gates.

We can make the Full-Adder block diagram into a symbol in order to use the circuit in the software and it looks like this.

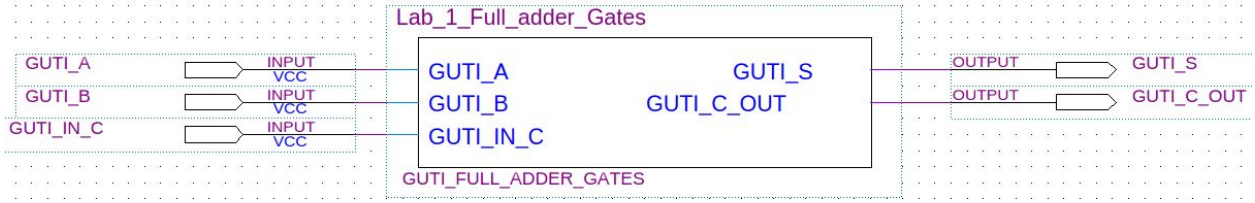

Figure 8: Full-Adder Symbol using Gates.

3.2 Simulation for Full-Adder Using Gates.

In the simulation the values for GUTI_A, GUTI_B, GUTI_IN_C will be given in different intervals of 0 and 1 and that will be done in different combinations. We will then compare the output of the simulation to the truth table in order to make sure the circuit is working correctly. The input values will alternate by 100ns and the simulation is read by reading the columns from left to right, the first interval is 0ns-100ns the input for GUTI_A, GUTI_B and GUTI_IN_C are all 0 in the first column and the output for GUTI_OUT_C and GUTI_S are both 0 which works correctly according to the truth table.

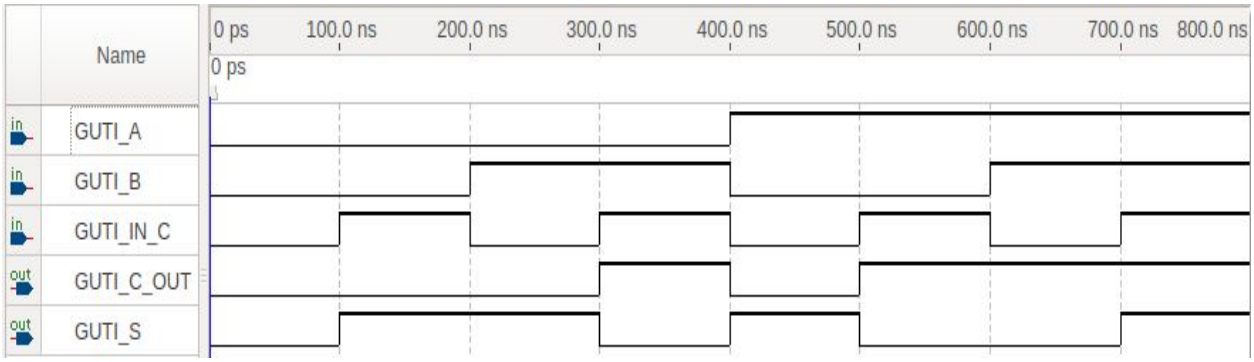

Figure 9: Vector waveform corresponding to Full-Adder Using Gates.

We can notice that when all three input values are are 0 the output values are also 0. When all of the three input values are 1, the output values for the total sum and the total carry are both 1, just like in the truth table. The truth table and the simulation match up correctly in combination.

3.3 Demonstration of Full-Adder on DE1-SoC Board.

The inputs and outputs that are assigned to pins on the DE1-SoC board are:

GUTI_A is assigned to PIN_AB12 SW[0]

GUTI_B is assigned to PIN_AC12 SW[1]

GUTI_IN_C is assigned to PIN_AF9 SW[2]

GUTI_C_OUT is assigned to PIN_V16 LED[0]

GUTI_S is assigned to PIN_W16 LED[1]

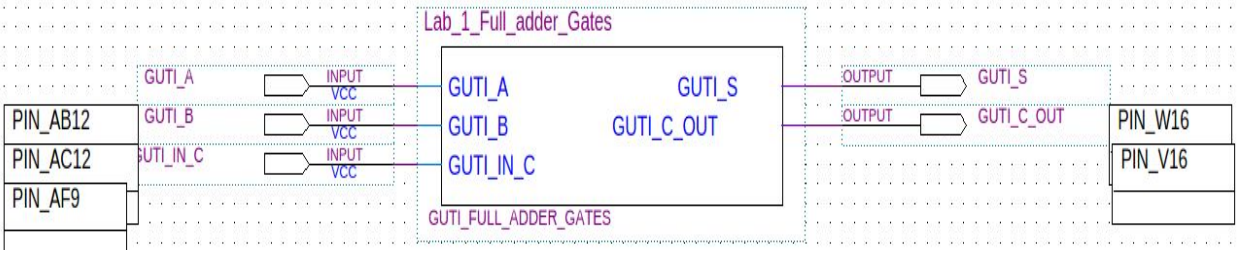

Figure 10: PIN assignments of the Full-Adder Circuit to the DE1-SoC Board.

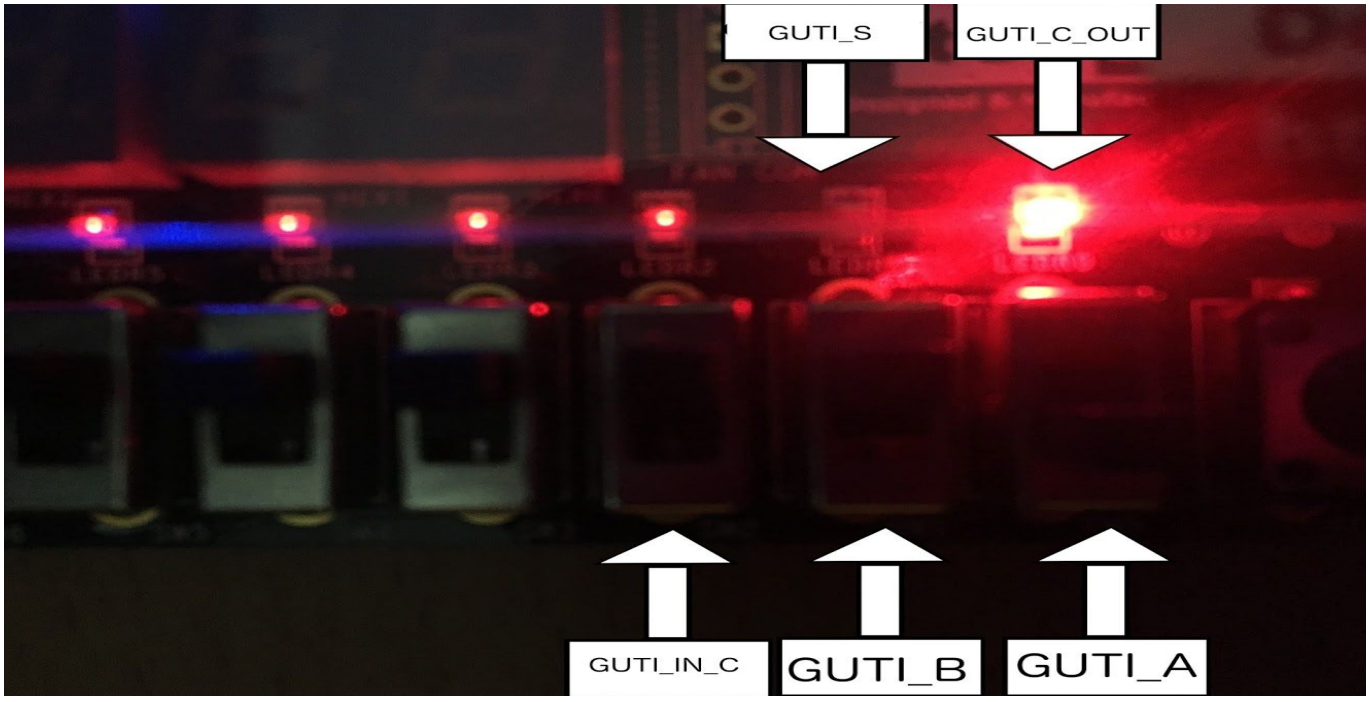

Figure 11:Digital Circuit of Full-Adder Using Gates when GUTI_A and GUTI_B are both 1 and the Carry Value GUTI_IN_C is 0. The output for GUTI_C_OUT is 1 and the output for the GUTI_S is 0.

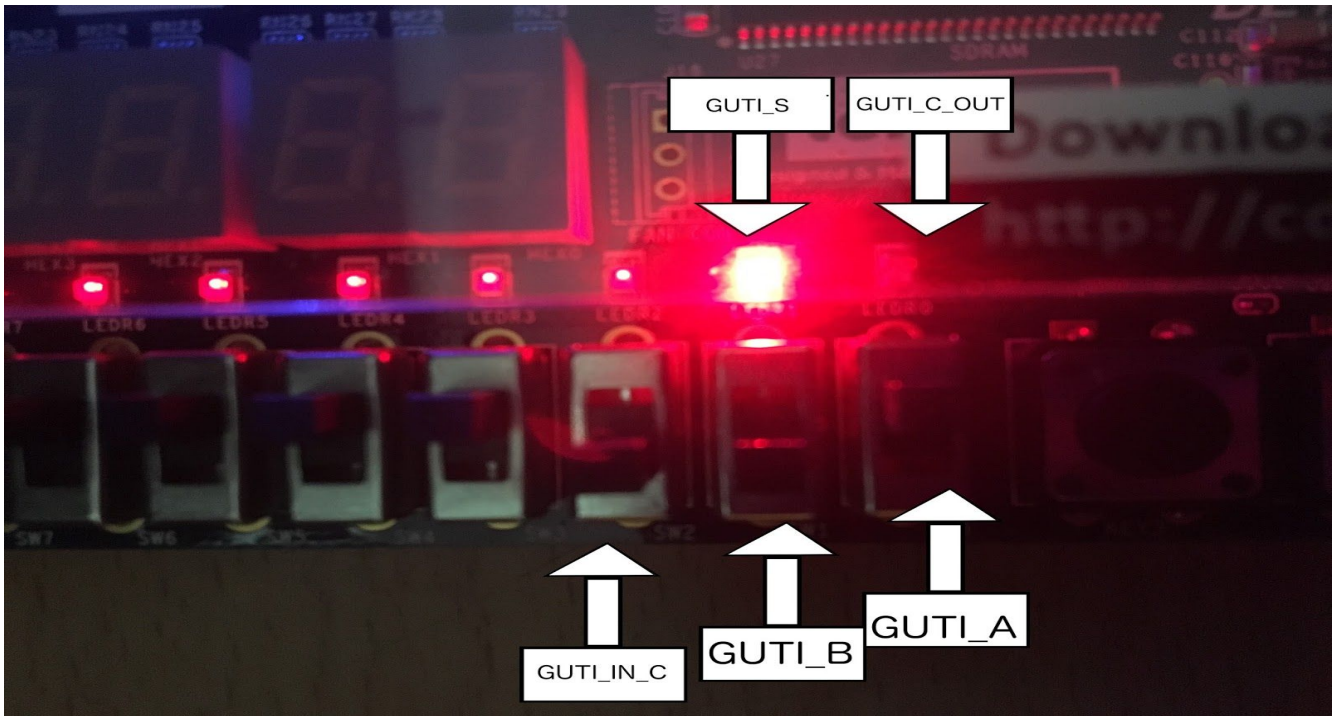

Figure 12: Digital Circuit of Full-Adder Using Gates when GUTI_A is 1 and GUTI_B is 0 and the Carry Value GUTI_IN_C is 0. The output for GUTI_C_OUT is 0 and the output for the GUTI_S is 1.

4 Full-Adder using Half-Adder as a Component.

4.1 Functionality and Specifications for Full-Adder Using Half-Adder as Components.

In this step we will be using half-adder components we created earlier in order to create a full adder circuit. This full adder circuit will follow the same procedure in terms of how many inputs it takes and how many outputs it returns in the same way the Full adder using gates does. The

components used in the full adder are two half adders and one or gate, the or gate is used in order to output the carry value while the second half adder computes the total sum and give it as an output.

Full-Adder using Half-Adder Components Truth table

GUTI_A	GUTI_B	GUTI_CIN	GUTI_COUT	GUTI_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

The boolean function that represents the full adder using half adder components is the same as the equation for the full adder using gates.

$$Sum = \overline{A}B\overline{C_{in}} + \overline{A}\overline{B}C_{in} + A\overline{B}\overline{C_{in}} + ABC_{in}$$

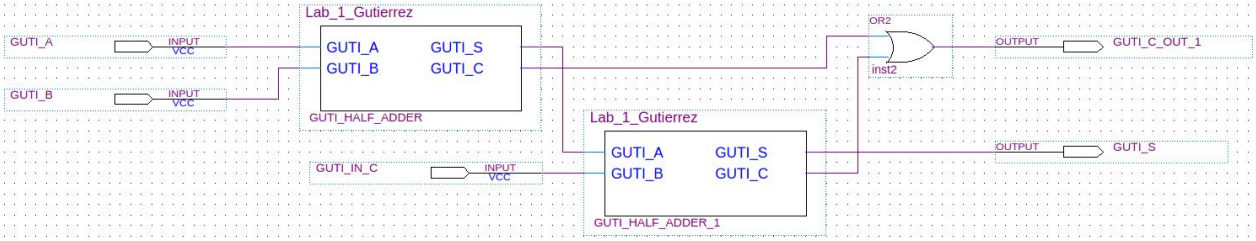

Figure 13: Block Diagram of Full-Adder using Half-Adder components.

The input values are GUTI_A, GUTI_B and GUTI_CIN, the output values come from GUTI_COUT and GUTI_S. The GUTI_S value comes from the second half adder that calculates the final calculation for the three binary numbers.

4.2 Simulation for Full-Adder Using Half-Adder as Components.

In this simulation the values for GUTI_A,GUTI_B, GUTI_IN_C, will be given in different intervals of 0 and 1 and that will be done in different combinations. We will then compare the output used in this simulation to the truth table in order to make sure that tie circuit for the full adder using half adder components is working correctly. The output values are going to be GUTI_C_OUT for the total carry value and the GUIT_S will be the output for the total sum of the binary numbers.

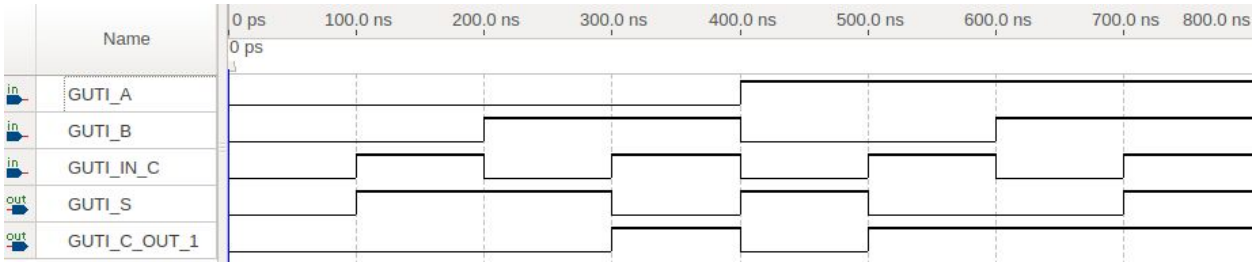

Figure 14: Vector waveform corresponding to the Full-Adder circuit using Half-Adder components.

We can notice that the values in the simulation match the values in the truth table, and that the same values also match the full adder values using gates. That means that both circuits were created correcting bto using gates and using the half adders as components.

4.3 Demonstration of Full-Adder Using Half-Adder as Components on DE1-SoC Board.

The inputs and outputs that are assigned to pins on the DE1-SoC board are:

GUTI_A is assigned to PIN_AB12 SW[0]

GUTI_B is assigned to PIN_AC12 SW[1]

GUTI_IN_C is assigned to PIN_AF9 SW[2]

GUTI_C_OUT_1 is assigned to PIN_V16 LED[0]

GUTI_S is assigned to PIN_W16 LED[1]

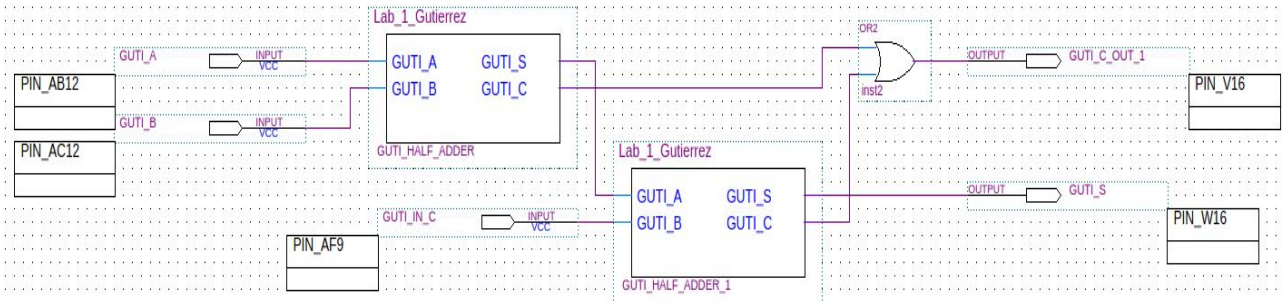

Figure 15: PIN assignments of the Full-Adder Circuit to the DE1-SoC Board.

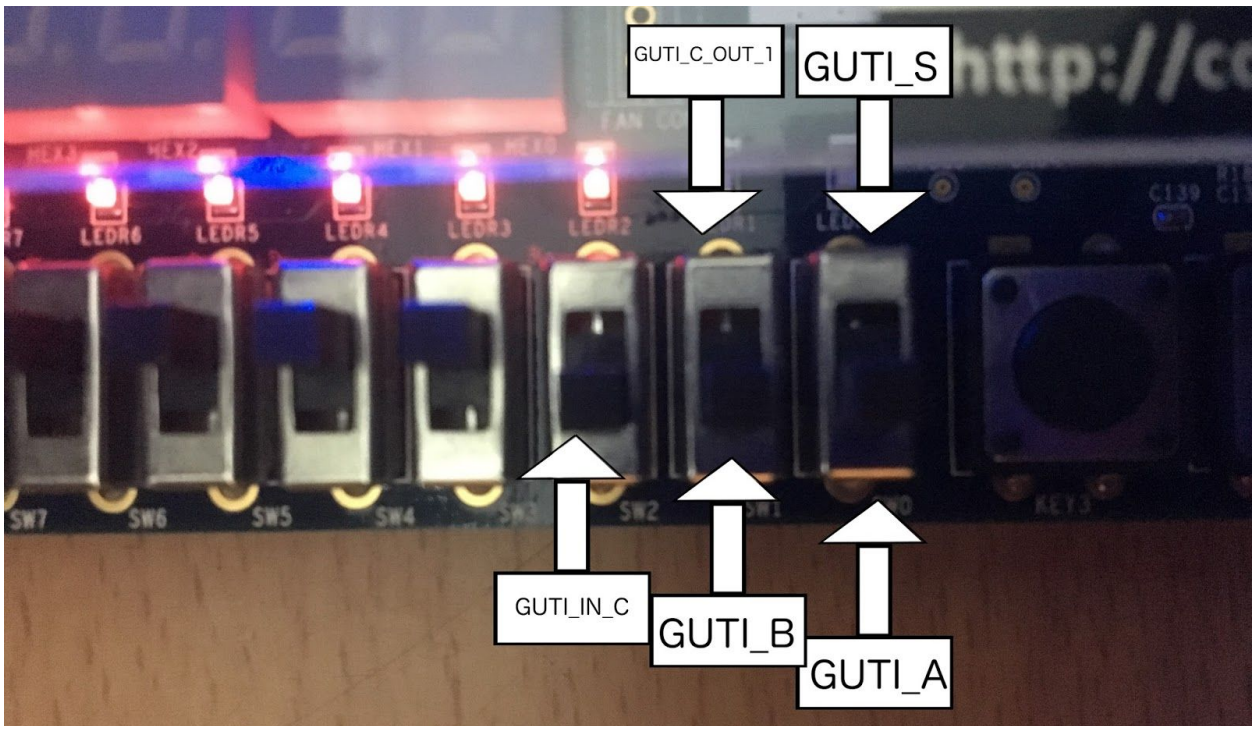

Figure 16: Digital Circuit of Full-Adder Using Half-Adder Components when GUTI_A, GUTI_B and GUTI_IN_C are all 0. The output for GUTI_S and GUTI_C_OUT_1 are both 0.

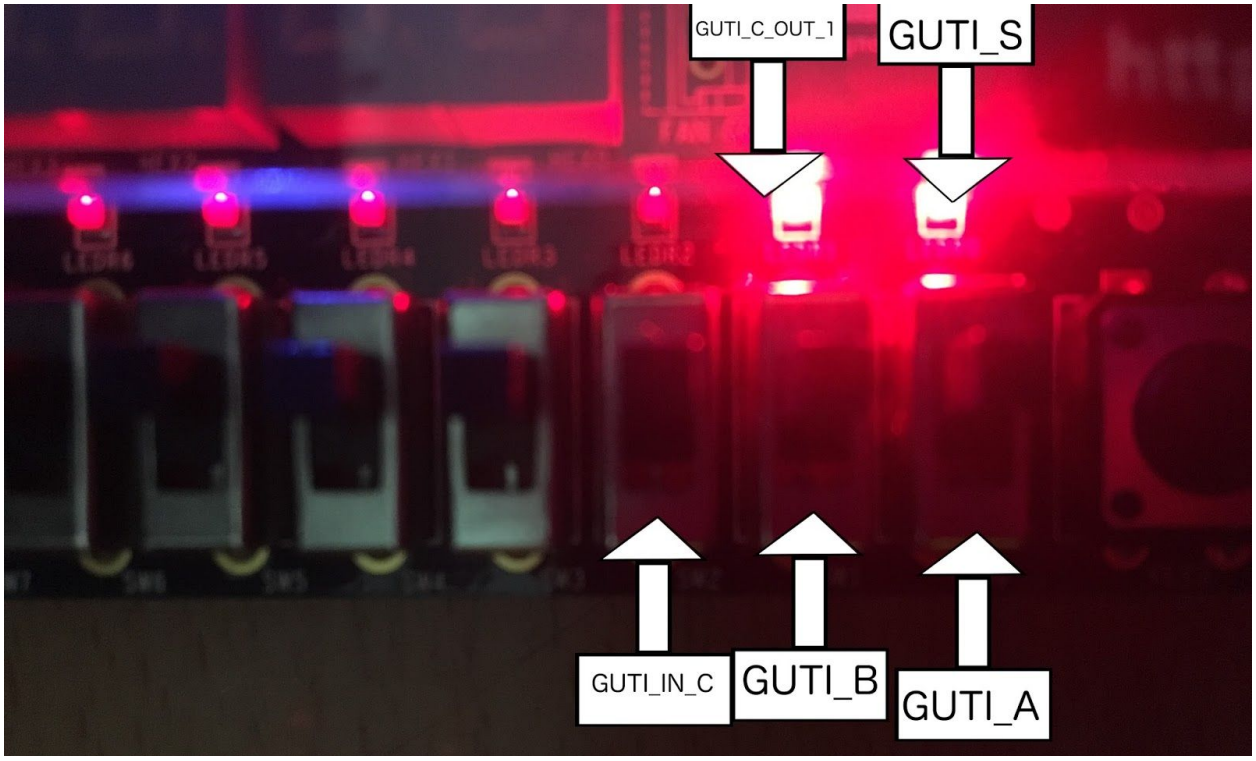

Figure 17: Digital Circuit of Full-Adder using Half-Adder Components when GUTI_A, GUTI_B and GUTI_IN_C are all 1. The output for GUTI_S and GUTI_C_OUT_1 are both 1.

5 4-Bit Adder Using Full-Adder as Component.

5.1 Functionality and Specifications 4-Bit Adder Using Full-Adder as Component.

The purpose of a 4-bit adder is to add 4-bit binary numbers in together completely. The design uses 4 full adders either through gates or with the half adder in order to add each of the 4 bits respectively together. Instead of adding the values by 4 bit values when giving the input we will use 9 different inputs, GUTI_A1, GUTI_A2, GUTI_A3, GUTI_A4, GUTI_B1, GUTI_B2, GUTI_B3, GUTI_B4, GUTI_CIN, are all of the input values where A-1 through A-4 are the first

4 bit of the first binary number and B1-B4 are the second number. GUTI_CIN is the initial carry value which starts at 0. The output values are GUTI_S1, GUTI_S2, GUTI_S3, GUTI_S4, for the total sums of the 4 bit binary numbers and GUTI_COUT for the final remaining carry value.

4-Bit Adder Truth Table

GUTI_A1	GUTI_A2	GUTI_A3	GUTI_A4	GUTI_B1	GUTI_B2	GUTI_B3	GUTI_B4	GUTI_CIN	GUTI_S1	GUTI_S2	GUTI_S3	GUTI_S4	GUTI_COUT
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	0	1	1	0	0	0	1	0	1
0	0	0	0	0	1	1	1	0	0	1	1	0	1
1	0	0	0	1	1	1	1	0	0	0	0	1	0
1	0	0	1	1	1	1	1	0	0	0	0	1	1
1	0	1	1	1	1	1	1	0	0	0	1	1	1
1	1	1	1	1	1	1	1	0	0	1	1	1	1

The values in the truth tables are the values that are going to be tested in the simulation. The 4-bit adder has 2^9 possible rows in the truth table but we will not be testing all of the values, if enough of the values in the truth table also work in the simulation then the circuit is correct and we can assume that all of the values will work giving a correct output.

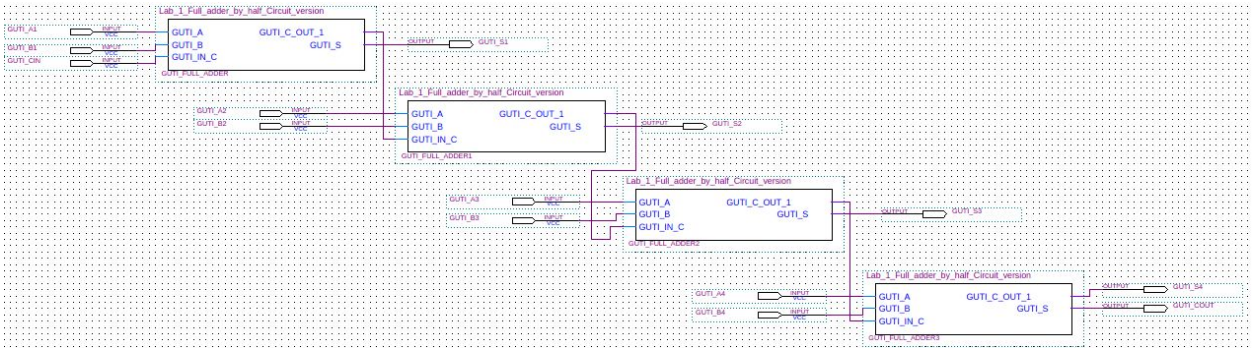

Figure 18 : Block diagram of 4-Bit Adder using Full-Adder Components.

5.2 Simulation for 4-Bit Adder.

In this simulation each of the inputs will be given different interval values of 0 and 1 in different combinations. The values that are tested will also correspond to the truth table because we need to test that the circuit works correctly for at least the 8 combinations in the 4-bit adder truth table.

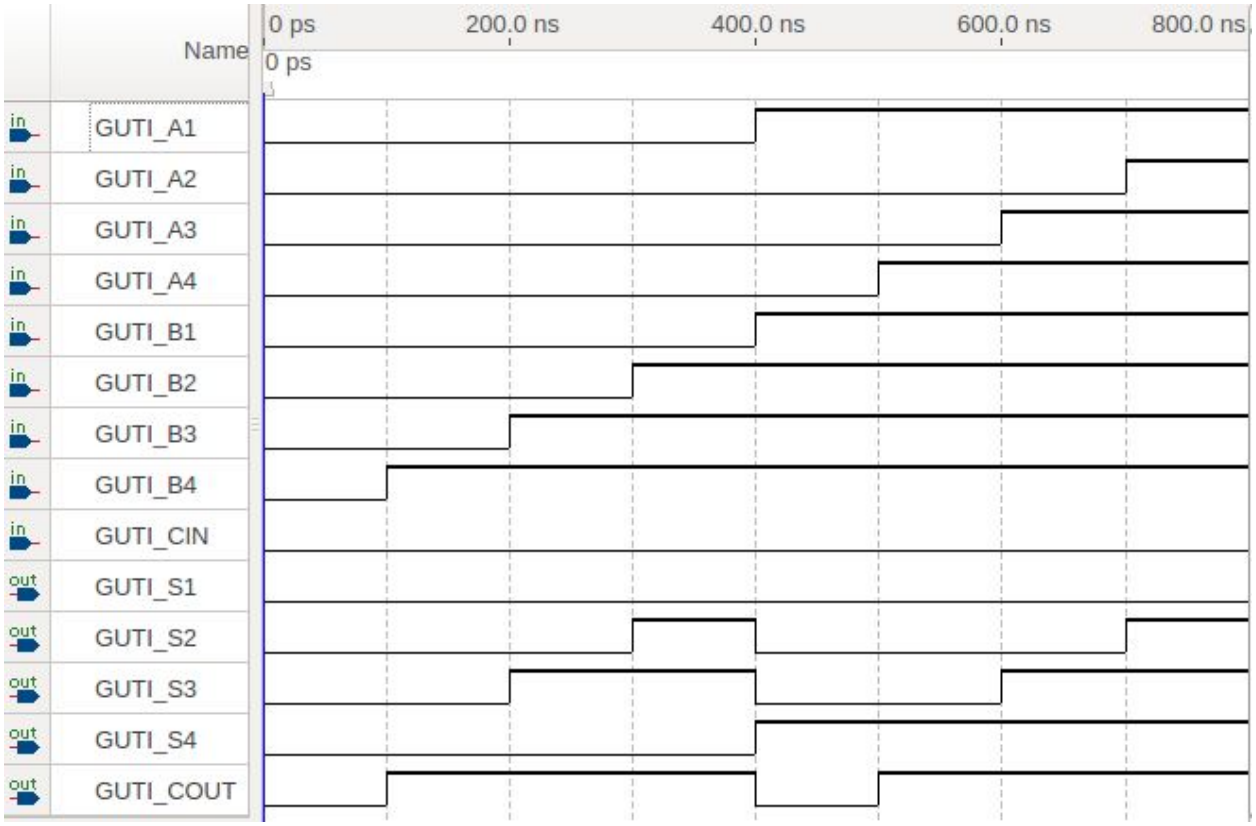

Figure 19: Vector Waveform corresponding to 4-bit adder.

We can notice that when all of the input values are 0 all the carries and the total sum values are alsos 0. When we have all of the input values as 1 not including the carry value, the total sum values GUTI_S2-GUTI_S4 are 1 the initial GUTI_S1 is 0 and the carry value is 1. The values on the simulation correspond accordingly and correctly to the truth table which means our circuit was built correctly.

5.3 Demonstration of 4-Bit Adder on DE1-SoC Board.

The Input and outputs that are assigned to pins on the DE1-SoC board are:

GUTI_A1 is assigned to PIN_AB12 SW[0]

GUTI_A2 is assigned to PIN_AC12 SW[1]

GUTI_A3 is assigned to PIN_AF9 SW[2]

GUTI_A4 is assigned to PIN_AF10 SW[3]

GUTI_B1 is assigned to PIN_AD11 SW[4]

GUTI_B2 is assigned to PIN_AD12 SW[5]

GUTI_B3 is assigned to PIN_AE11 SW[6]

GUTI_B4 is assigned to PIN_AC9 SW[7]

GUTI_CIN is assigned to PIN_AD10 SW[8]

GUTI_S1 is assigned to PIN_V16 LED[0]

GUTI_S2 is assigned to PIN_W16 LED[1]

GUTI_S3 is assigned to PIN_V17 LED[2]

GUTI_S4 is assigned to PIN_V18 LED[3]

GUTI_COUT is assigned to PIN_W17 LED[4]

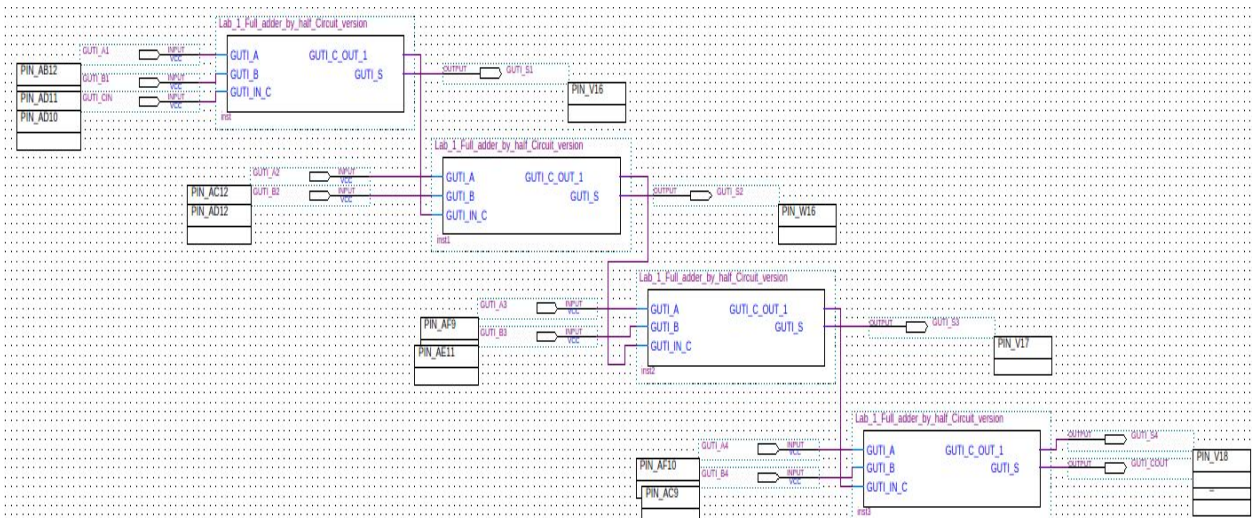

Figure 20: PIN assignments of the 4-Bit adder circuit to the DE1-SoC Board.

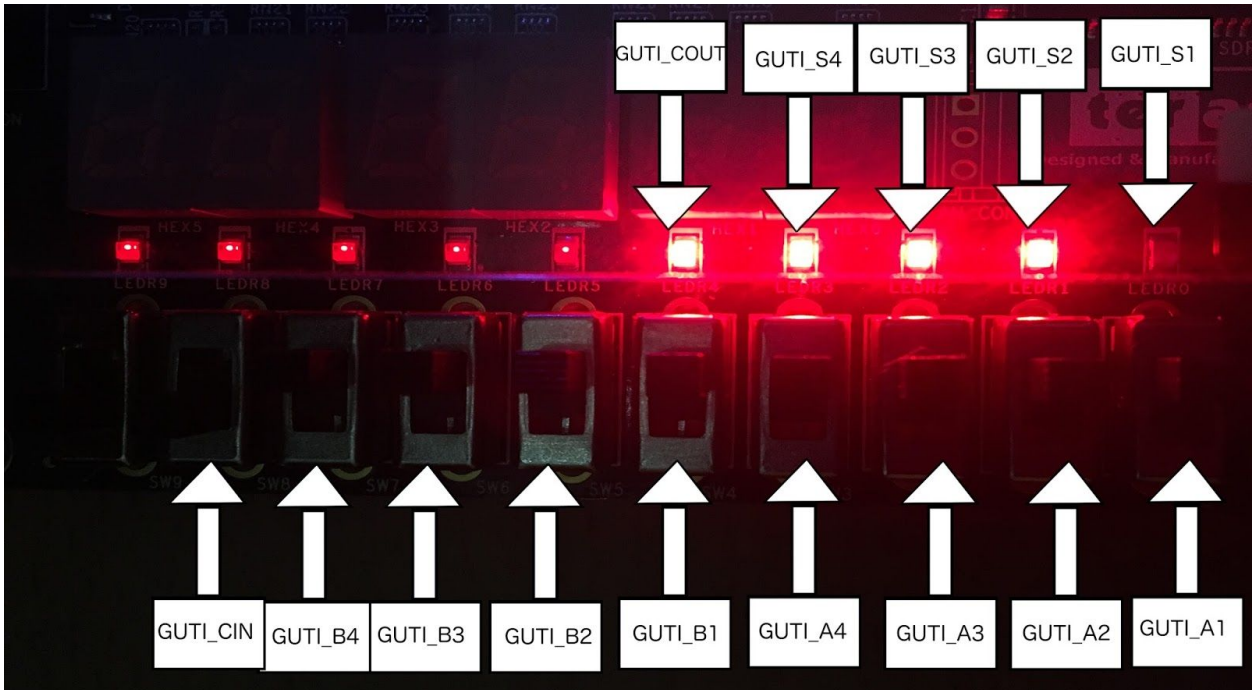

Figure 2: Digital Circuit of 4-Bit Adder Using Gates when GUTI_A1, GUTI_A2, GUTI_A3, GUTI_A4, GUTI_B1, GUTI_B2, GUTI_B3, GUTI_B4, are all 1 and GUTI_CIN is 0. GUTI_S1 returns an output value of 0. The output values of GUTI_S2, GUTI_S3, GUTI_S4 and GUTI_COUT all return a value of 1.

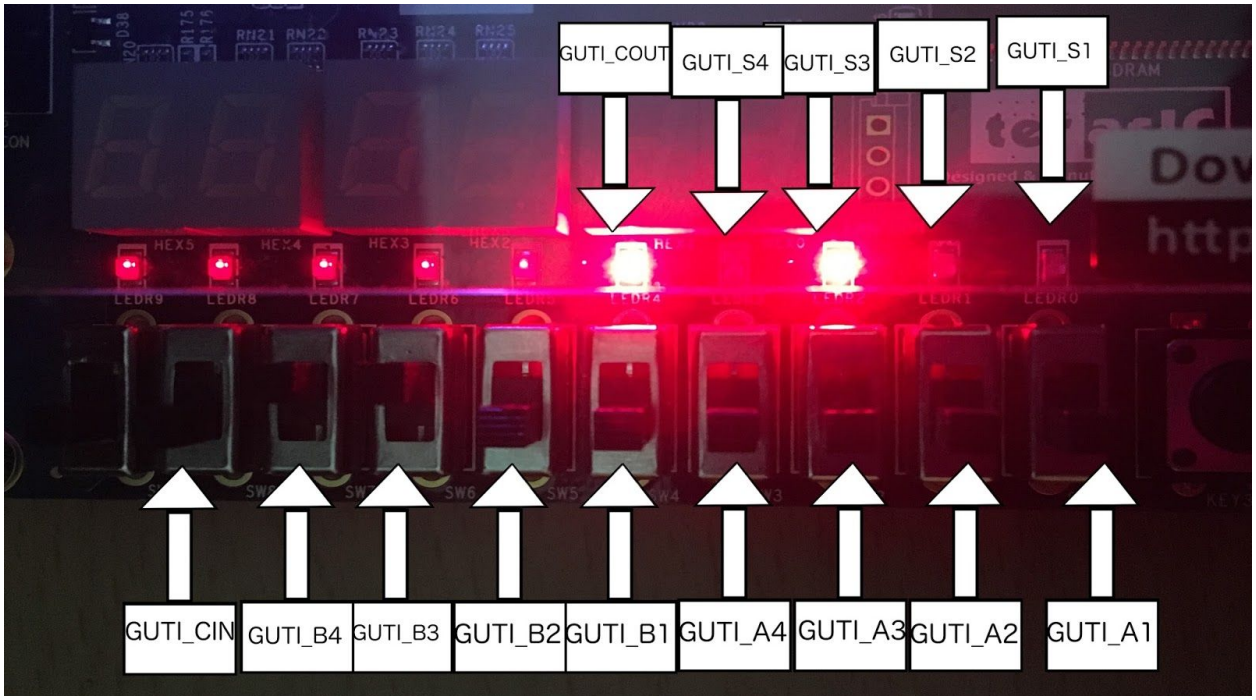

Figure 22: Digital Circuit of 4-Bit Adder Using Gates when GUTI_A1, GUTI_A2, GUTI_A3, GUTI_A4, GUTI_B1, GUTI_B2 are all 0. The input values for GUTI_B3, GUTI_B4 are both 1. The input value for the GUTI_CIN carry value is 0. GUTI_S1, GUTI_S2, GUTI_S4 returns an output value of 0. The output values of GUTI_S3 and GUTI_COUT all return a value of 1.

6 Conclusion

In this lab I used the Master Tutorial given by the professor in order to use the Software Quartus to make circuits. The circuits that were made were; Half-Adder circuit, Full-Adder circuit using Gates, Full-Adder Using Half-Adder as components and a 4-Bit adder. The purpose of making

the four circuits is to understand how each of the circuits work. I learned how to design a circuit and install it onto a DE1-SoC circuit board on a computer and how to use a waveform simulation in order to compare the circuit design and results to a truth table. I was introduced to and gates, xor gates and or gates, I also learned how to assign pins values to the input and output values in the circuit board to the equivalent location designed on the software. As it turns out, the designs I was able to make all worked and produced the correct output, in simulation, on the actual circuit board and when compared to the truth table which means that my understanding of how bit adders work is now correct. Initially working with the 4-bit adder it was hard to identify which of the 2^9 possible combinations of values to test the 4-Bit adder with but i understand that i should just test a few and that if they work, in theory all of the rest of the combinations should also work on the circuit board, that is going to be helpful with the coming labs because i will not be stuck wondering whether or not i should try to figure out how to test that many values. I also haven't learned how to use the simulation properly enough for an automatic input, all of the values that were tested for input had to be put in manually. One of the questions to consider was how much time it would take to compute the sum of two 32-bit words, when denoting Δt as 1-Bit full adder. In this case it would take 16 4-bit adders or 64 1-Bit adders in order to compute the sum of two 32 bit words.

7 Appendix

Pins_Half_Adder.txt

```
1  To, Location
2  GUTI_A,PIN_AB12
3  GUTI_B,PIN_AC12
4  GUTI_C,PIN_V16
5  GUTI_S,PIN_W16
```

Figure 23: Pin Assignment for Half-Adder.

Pins_Full_Adder.txt

```
1  To, Location
2  GUTI_A,PIN_AB12
3  GUTI_B,PIN_AC12
4  GUTI_IN_C,PIN_AF9
5  GUTI_C_OUT,PIN_V16
6  GUTI_S,PIN_W16
```

Figure 24: Pin Assignment for Full-Adder Using Gates.

Pins_Full_Adder_With_Half_Adder.txt

```
1  To, Location
2  GUTI_A,PIN_AB12
3  GUTI_B,PIN_AC12
4  GUTI_IN_C,PIN_AF9
5  GUTI_C_OUT_1,PIN_V16
6  GUTI_S,PIN_W16
```

Figure 25: Pin Assignment for FUll-Adder Using Half-Adder Components.

Pins_4Bit-Adder.txt

```
1  To, Location
2  GUTI_A1,PIN_AB12
3  GUTI_A2,PIN_AC12
4  GUTI_A3,PIN_AF9
5  GUTI_A4,PIN_AF10
6  GUTI_B1,PIN_AD11
7  GUTI_B2,PIN_AD12
8  GUTI_B3,PIN_AE11
9  GUTI_B4,PIN_AC9
10 GUTI_CIN,PIN_AD10
11 GUTI_S1,PIN_V16
12 GUTI_S2,PIN_W16
13 GUTI_S3,PIN_V17
14 GUTI_S4,PIN_V18
15 GUTI_COUT,PIN_W17
```

Figure 26: Pin Assignment for 4-Bit Adder.