

Lab 1 Binary Adder Lab Report

CS211 Summer 2017

June 12, 2017

Xu Mingzhi

Table of Content

1. Objective	3
2. Half-Adder	4
2.1 Functionality and Specifications	4
2.2 Simulation	5
2.3 Demonstration	5
3. One Bit Full Adder using Gates	8
3.1 Functionality and Specifications	8
3.2 Simulation	9
3.3 Demonstration	10
4. One Bit Full Adder using Half-Adder as a component	12
4.1 Functionality and Specifications	12
4.2 Simulation	14
4.3 Demonstration	15
5. 4-Bit Full Adder using 1-Bit Full Adder as a component	17
5.1 Functionality and Specification	17
5.2 Simulation	19
5.3 Demonstration	20
6. Conclusion	22

1. Objective

In this lab, we will be using what we have learned from the tutorial that explained the basic tools for designing circuits in Quartus Prime to build circuits for binary adders, run waveform simulation that correspond to the truth table that defines the circuit, and loading the circuits onto the DE1-SoC board to test if the result corresponds to the simulations and the truth table. The circuit that we will be designing includes:

- a. Half-Adder
- b. One Bit Full Adder using Gates
- c. One Bit Full Adder using Half-Adder
- d. 4-Bit Full Adder using 1-Bit Full Adder

2. Half-Adder

2.1 Functionality and Specifications

The Half-Adder takes two inputs x , y and add them up in base 2 form and produces two outputs, the sum s and carry over c . Since it is a binary adder and binary are in base 2 form so only 0 and 1 are used in this addition. Below is the truth table of all possible input for x and y along with the Boolean function that defines the two outputs s and c .

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$s = x \text{ XOR } y$
 $c = x \text{ AND } y$

When the Boolean function is applied to the block diagram, a XOR gate and an AND gate is used to design the circuit for Half-Adder. The XOR gate will take x and y as inputs and outputs s . The AND gate will also take x and y as inputs and outputs c .

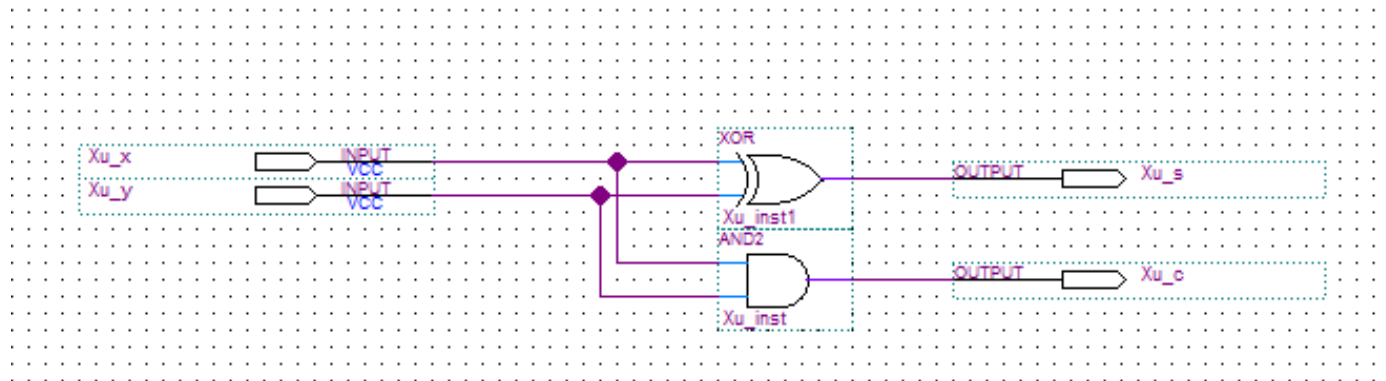


Figure 1: Block diagram of Half-Adder

2.2 Simulation

In the simulation, values of 0 and 1 will be the input for x and y at varying intervals. Input x will have value of 0 and 1 at each 320ns interval. Input y will have values of 0 and 1 at each 160ns interval.

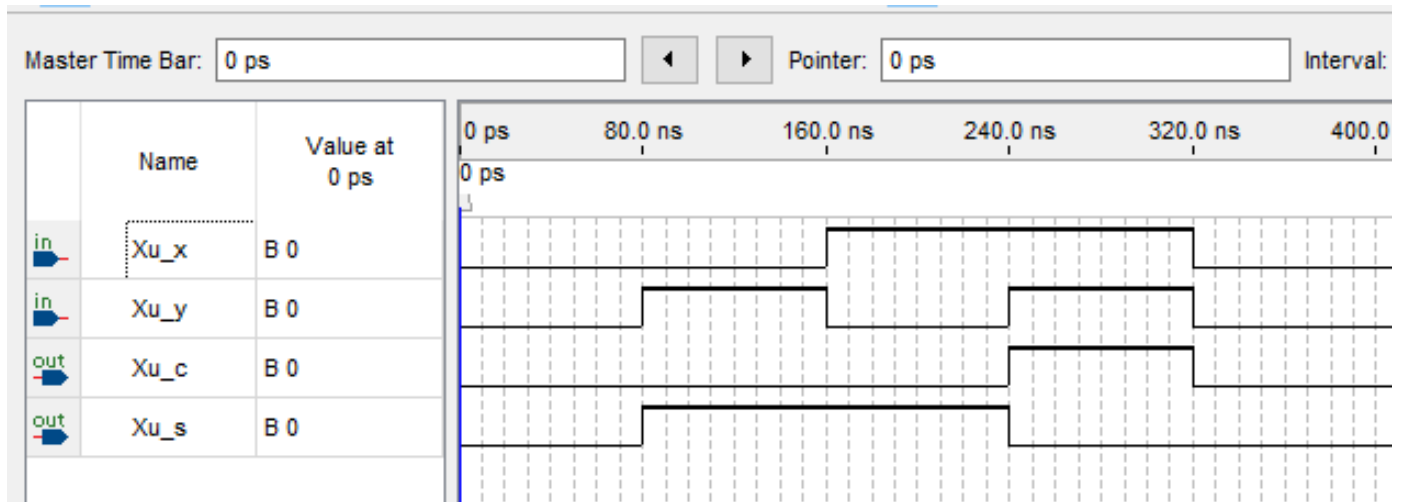


Figure 2: Vector waveform simulation corresponding to Half-Adder

We can observe that when either input x or y has the value of 1 then the output of c will be 0 and the sum s will be 1 and when both input x and y has the value of 1 then the output of c will be 1 and the sum s will be 0. These results correspond to the inputs and outputs shown on the truth table.

2.3 Demonstration

The inputs and outputs are assigned to pins on the DE1-SoC Board.

Xu_x is assigned to SW[0] which is PIN_AB12

Xu_y is assigned to SW[1] which is PIN_AC12

Xu_s is assigned to LEDR[0] which is PIN_V16

Xu_c is assigned to LEDR[1] which is PIN_W16

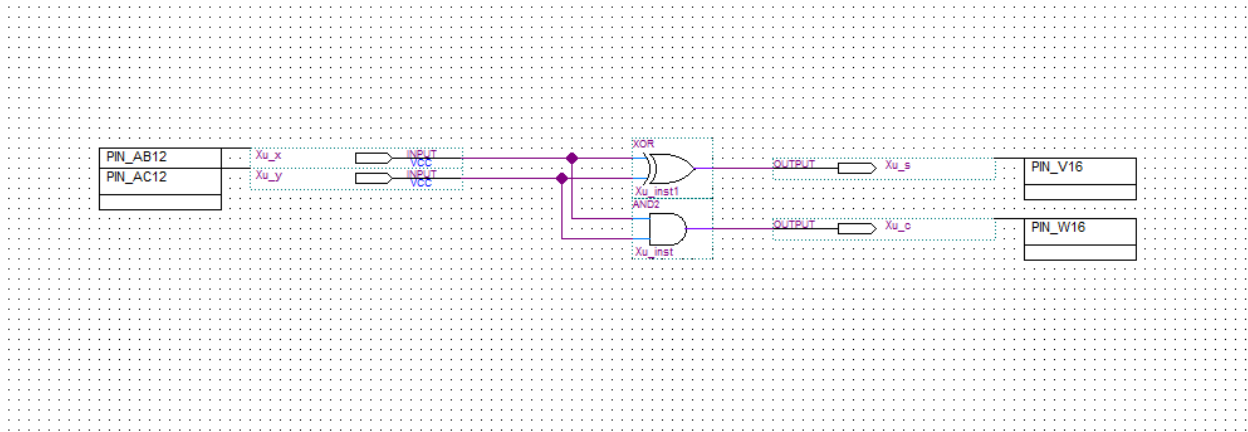


Figure 3: PIN assignments of the Half-Adder circuit to DE1-SoC Board.

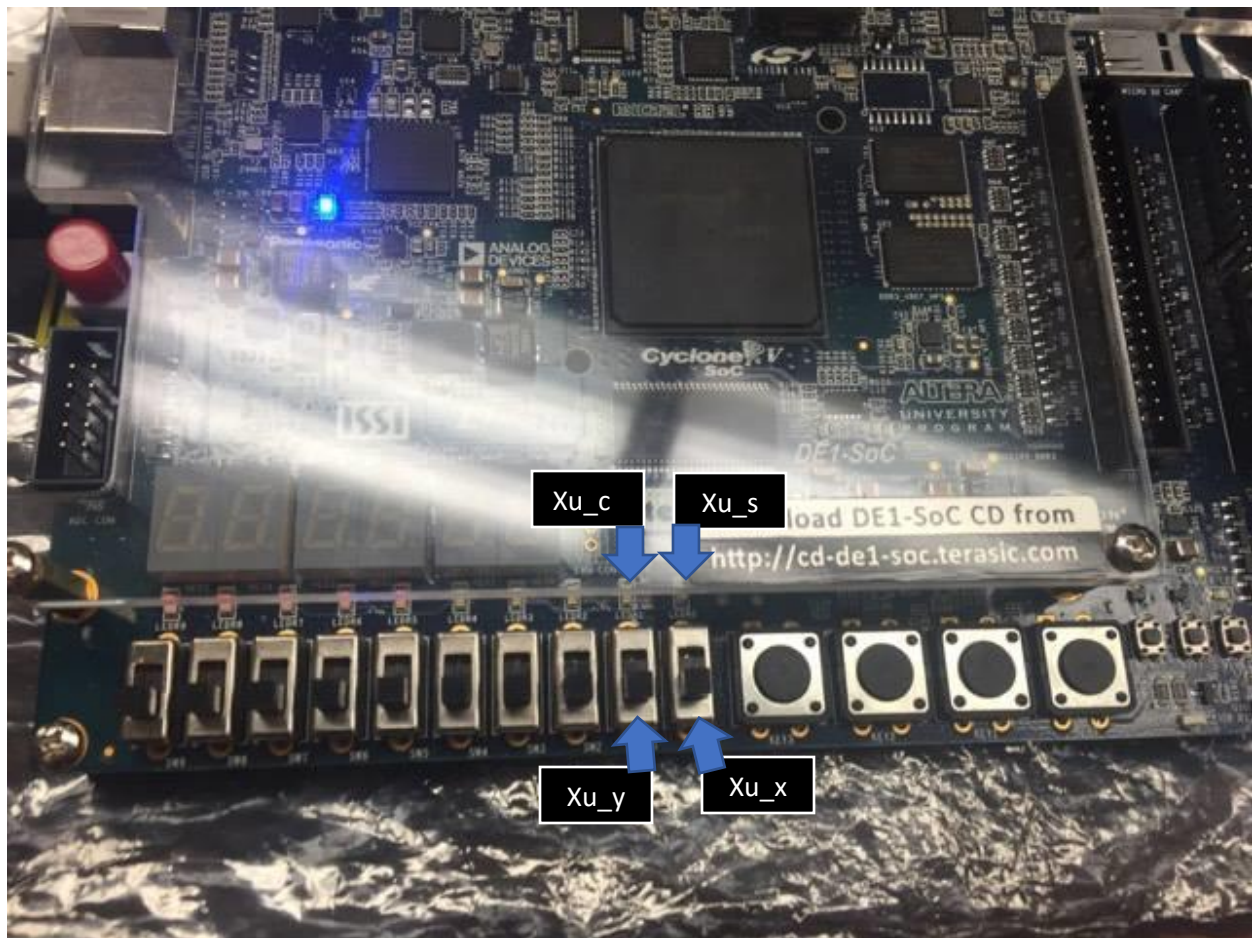


Figure 4: PIN assignment on the DE1-SoC Board



Figure 5: Digital circuit when x is 0 (off) and y is 1 (on), output of s is 1 (on) c is 0 (off)



Figure 6: Digital circuit when x is 1 (on) and y is 1 (on), output of s is 0 (off) c is 1 (on)

3. One Bit Full Adder using Gates

3.1 Functionality and Specifications

A One Bit Full Adder will have three inputs carry-in c , x and y and produces two outputs carry-out $c1$ and sum s . The idea of a Full Adder is similar to a Half-Adder but includes an additional input of a carry-in c . Below is a truth table that displays all of the possible inputs for x , y and carry-in c along with the Boolean function that defines the output sum s and carry-out $c1$.

x	y	c	$c1$	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

$$s = (x \text{ XOR } y) \text{ XOR } c$$

$$c1 = (x \text{ AND } y) \text{ OR } (c \text{ AND } (x \text{ XOR } y))$$

When the Boolean function is applied to the block diagram, two XOR gate, two AND gate, and an OR gate is used to design the circuit for One Bit Full Adder using Gates. A XOR gate takes the output of another XOR that takes x and y as input and the carry-in c to produce the sum s . An

OR gate takes output of x AND y and c AND $(x \text{ XOR } y)$ as inputs produce the carry-out $c1$.

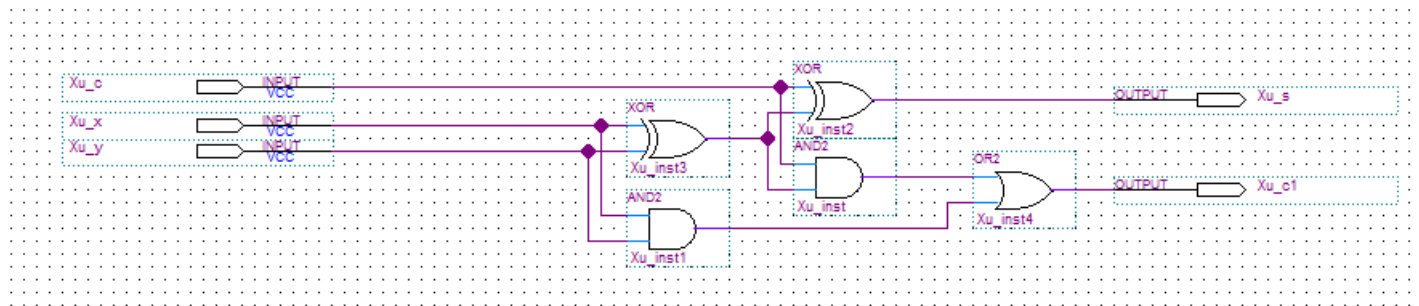


Figure 7: Block diagram of One Bit Full Adder using Gates

3.2 Simulation

In the simulation, values of 0 and 1 will be the input for x , y and carry-in c at varying intervals.

Input x will have value 0 and 1 at each 640ns interval. Input y will have value of 0 and 1 at each 320ns interval. Input carry-in c will have value 0 and 1 at each 160ns interval.

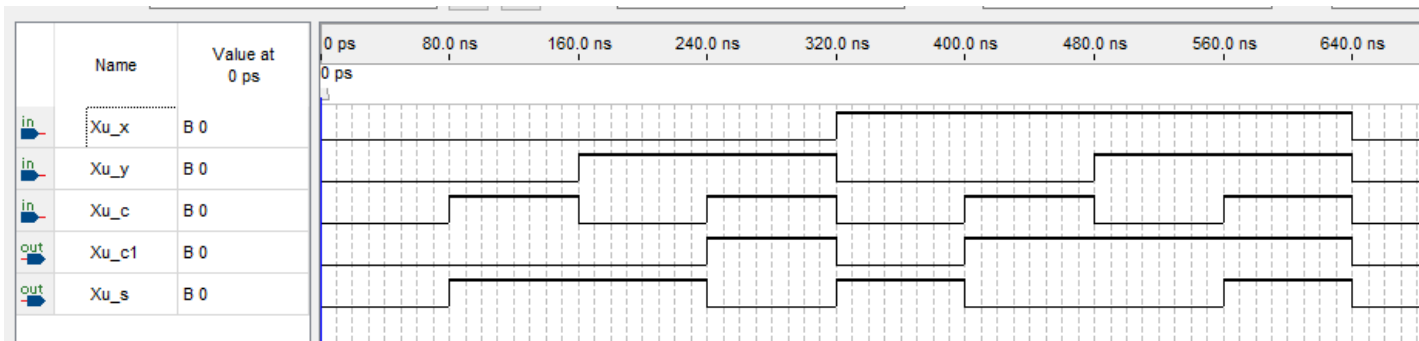


Figure 8: Vector waveform simulation corresponding to One Bit Full Adder using Gates

We can observe that when both x and y inputs are 0 and the carry-in c is 1 the output sum s will be 1 and carry-out $c1$ will be 0. When either x or y is 1 and carry-in c is 0 the output will be sum s is 1 and carry-out $c1$ will be 0. When either x or y is 1 and carry-in c is 1 the output sum s will be 0 and carry-out $c1$ will be 1. When all the inputs are 1 the output for both sum s and carry-out $c1$ will also be 1. The results on the simulation correspond to the truth table that defines One Bit Full Adder using Gates

3.3 Demonstration

The inputs and outputs are assigned to pins on the DE1-SoC Board.

Xu_x is assigned to SW[0] which is PIN_AB12

Xu_y is assigned to SW[1] which is PIN_AC12

Xu_c is assigned to SW[2] which is PIN_AF9

Xu_s is assigned to LEDR[0] which is PIN_V16

Xu_c1 is assigned to LEDR[1] which is PIN_W16

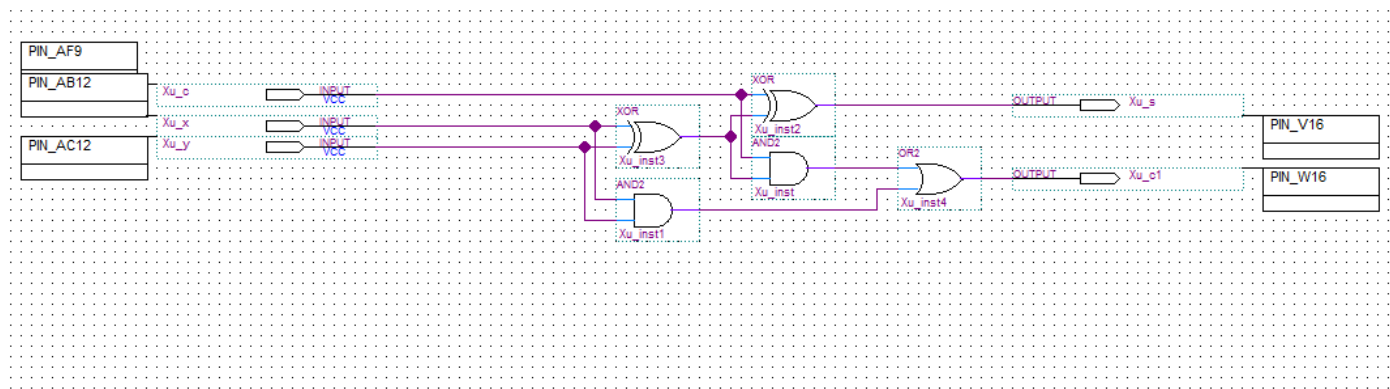


Figure 9: PIN assignments of Full Adder using Gates circuit to DE1-SoC Board

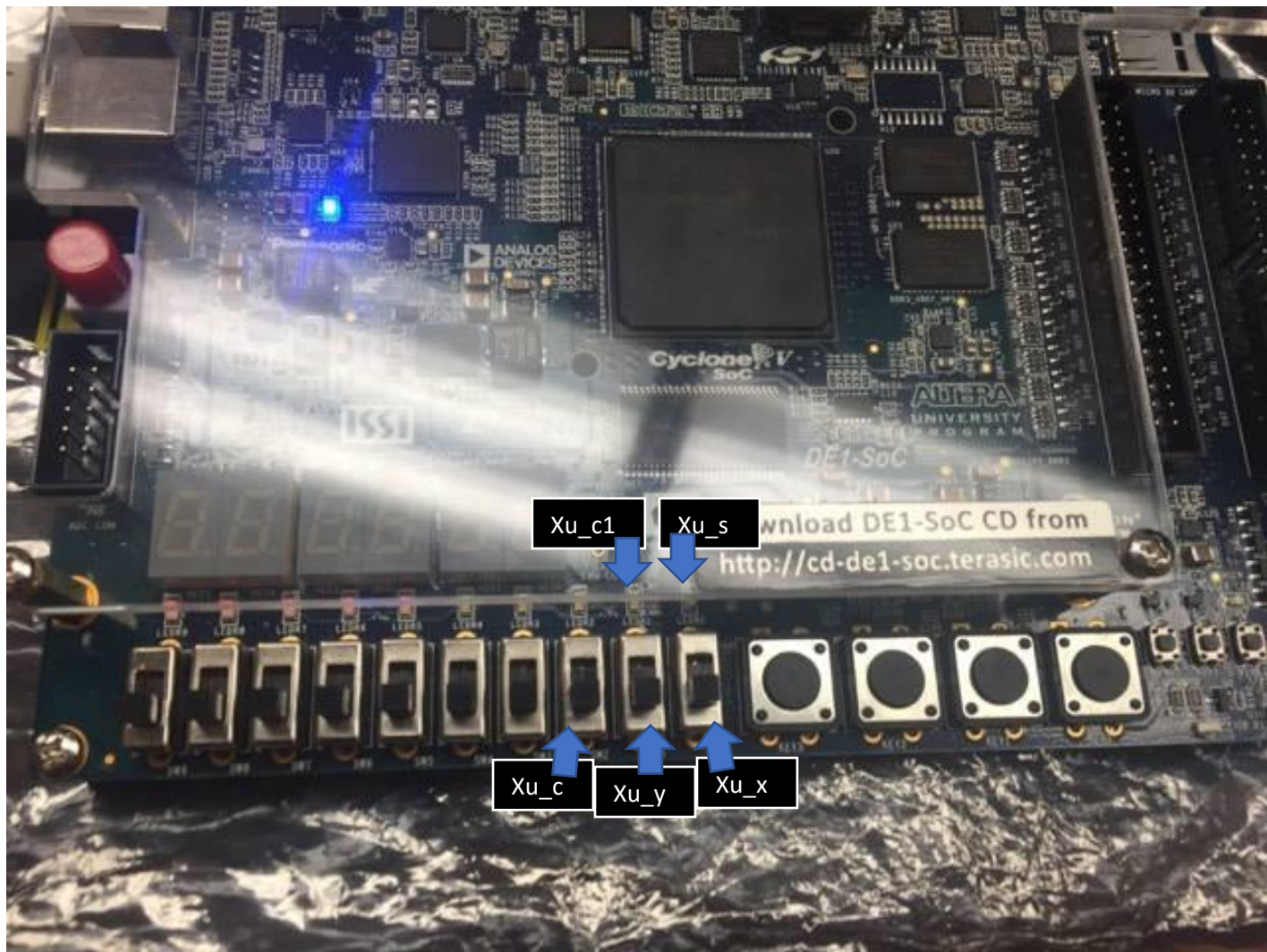


Figure 10: PIN assignment on the DE1-SoC Board



Figure 11: Inputs x is 0 (off) y is 1 (on) c is 0 (off) outputs c1 is 0 (off) s is 1 (on)



Figure 12: Input x is 1 (on) y is 1 (on) c is 1 (on) Outputs c1 is 1 (on) s is 1 (on)

4. One Bit Full Adder using Half-Adder as a component

4.1 Functionality and Specifications

A One Bit Full Adder using Half-Adder as a component is the same as One Bit Full Adder using Gates. Therefore, it will have the same truth table and Boolean function such that it will have three inputs carry-in c, x and y and produces two outputs carry-out c1 and sum s. Below is a truth table that displays all the possible inputs for x, y and carry-in c along with the Boolean function that defines the output sum s and carry-out c1.

x	y	c	c1	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

$$s = (x \text{ XOR } y) \text{ XOR } c$$

$$c1 = (x \text{ AND } y) \text{ OR } (c \text{ AND } (x \text{ XOR } y))$$

When the Boolean function is applied to the block diagram, two XOR gate, two AND gate, and an OR gate is used to design the circuit for One Bit Full Adder using Gates. A XOR gate takes the output of another XOR that takes x and y as input and the carry-in c to produce the sum s . An OR gate takes output of x AND y and c AND $(x$ XOR $y)$ as inputs produce the carry-out $c1$.

Since it is a Full Adder using Half-Adder as a component, and Half-Adder circuit contains one XOR gate and one AND gate, so therefore we can convert the two XOR gate and two AND gate that is used in the Full Adder using Gates into two blocks of Half-Adder as component.

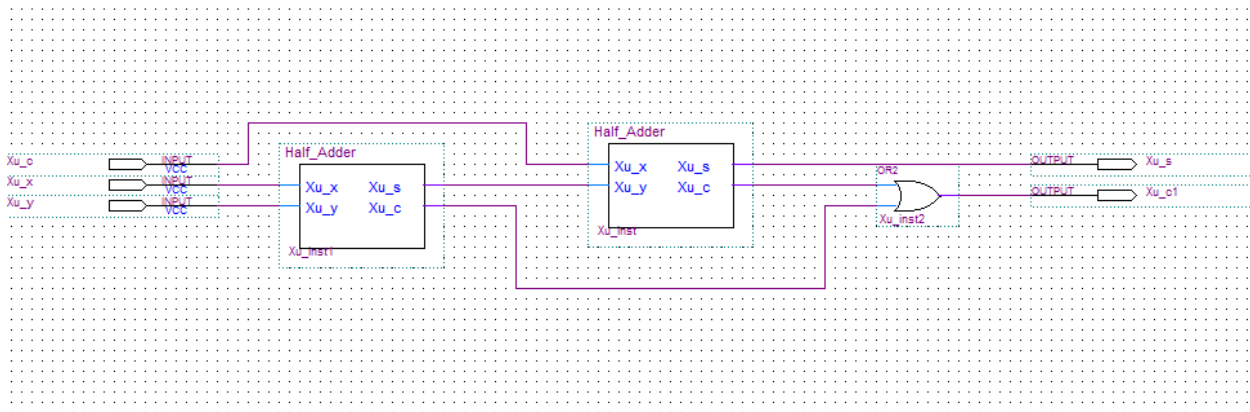


Figure 13: Block diagram of Full Adder using Half-Adder as a component

4.2 Simulation

In the simulation, values of 0 and 1 will be the input for x, y and carry-in c at varying intervals.

Input x will have value 0 and 1 at each 640ns interval. Input y will have value of 0 and 1 at each 320ns interval. Input carry-in c will have value 0 and 1 at each 160ns interval. It is the same as the Full Adder using Gates since it shares the same truth table and Boolean function.

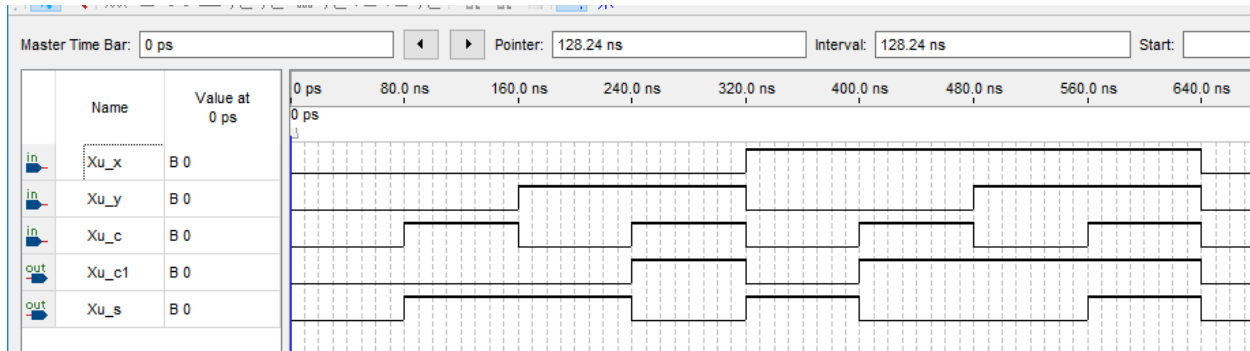


Figure 14: Vector waveform simulation corresponding to One Bit Full Adder using Half-Adders

We can observe that when both x and y inputs are 0 and the carry-in c is 1 the output sum s will be 1 and carry-out c1 will be 0. When either x or y is 1 and carry-in c is 0 the output will be sum s is 1 and carry-out c1 will be 0. When either x or y is 1 and carry-in c is 1 the output sum s will be 0 and carry-out c1 will be 1. When all the inputs are 1 the output for both sum s and carry-out c1 will also be 1. The results on the simulation correspond to the truth table that defines One Bit Full Adder using Half-Adders.

4.3 Demonstration

The inputs and outputs are assigned to pins on the DE1-SoC Board.

Xu_x is assigned to SW[0] which is PIN_AB12

Xu_y is assigned to SW[1] which is PIN_AC12

Xu_c is assigned to SW[2] which is PIN_AF9

Xu_s is assigned to LEDR[0] which is PIN_V16

Xu_c1 is assigned to LEDR[1] which is PIN_W16

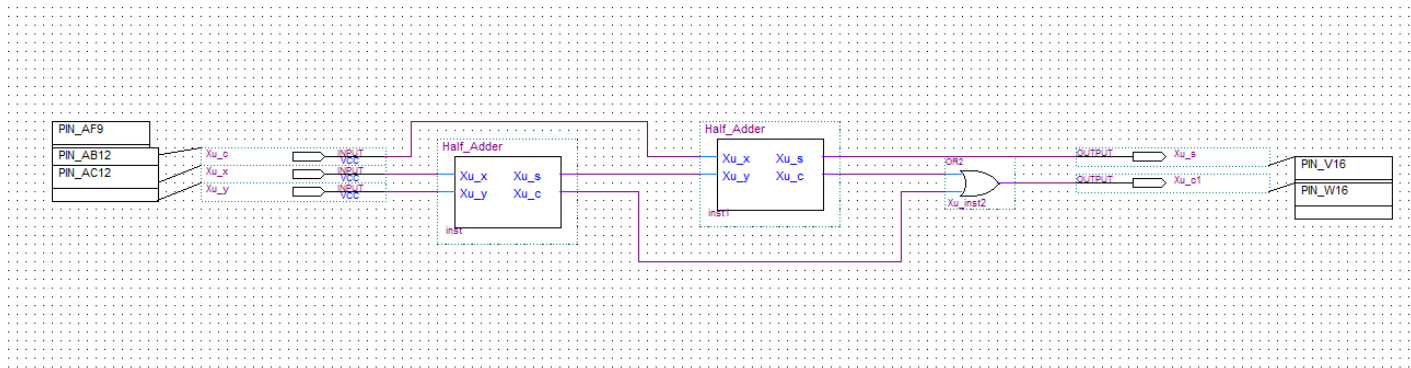


Figure 15: PIN assignments of Full Adder using Gates circuit to DE1-SoC Board

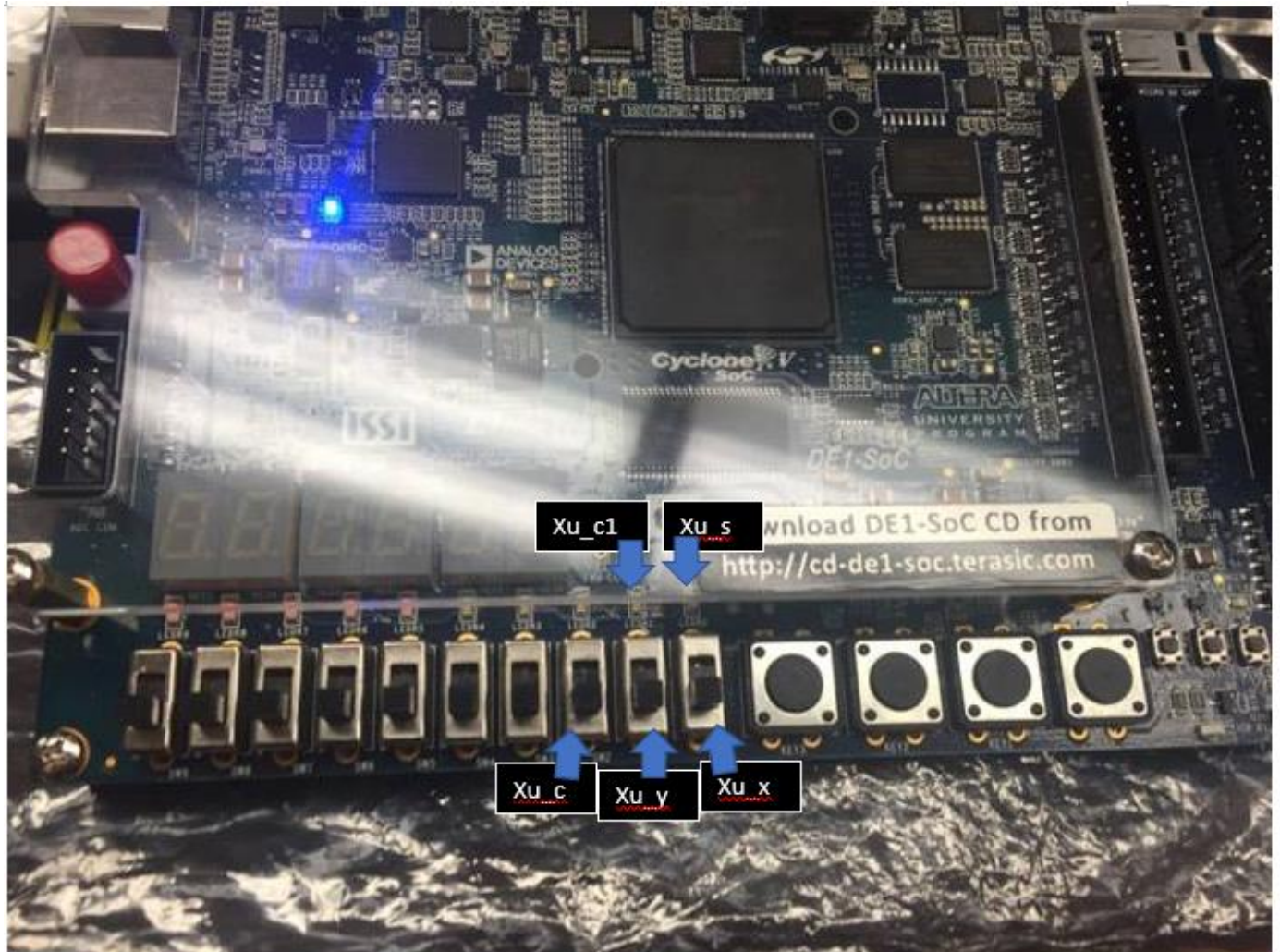


Figure 16: PIN assignment on the DE1-SoC Board



Figure 17: Input x is 0 (off) y is 0 (off) c is 1 (on) Outputs c1 is 0 (off) s is 1 (on)



Figure 18: Input x is 1 (on) y is 1 (on) c is 1 (on) Outputs c1 is 1 (on) s is 1 (on)

5. 4-Bit Full Adder using 1-Bit Full Adder as a component

5.1 Functionality and Specification

A 4-Bit Full Adder using 1-Bit Full Adder as a component would require using 4 Full Adder meaning there will be 9 inputs and producing 5 outputs. Since there are 9 inputs so 2^9 is 512 possible inputs which is going to be overly long if it were to implement onto a truth table. So here below is a shorten truth table of a 4-Bit Full Adder and Boolean functions.

cin	y3	y2	y1	y0	x3	x2	x1	x0	cout	s3	s2	s1	s0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1	0	1	1	1	1
0	1	1	1	1	0	0	0	0	0	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	1	1	1	1	1	0	0	0	0
1	1	1	1	1	0	0	0	0	1	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1
$s0 = (x0 \text{ XOR } y0) \text{ XOR } cin$													
$s1 = (x1 \text{ XOR } y1) \text{ XOR } ((x0 \text{ AND } y0) \text{ OR } (cin \text{ AND } (x0 \text{ XOR } y0)))$													
$s2 = (x2 \text{ XOR } y2) \text{ XOR } ((x1 \text{ AND } y1) \text{ OR } ((x0 \text{ AND } y0) \text{ OR } (cin \text{ AND } (x0 \text{ XOR } y0)) \text{ AND } (x1 \text{ XOR } y1))))$													
$s3 = (x3 \text{ XOR } y3) \text{ XOR } (((x2 \text{ AND } y2) \text{ OR } ((x1 \text{ AND } y1) \text{ OR } ((x0 \text{ AND } y0) \text{ OR } (cin \text{ AND } (x0 \text{ XOR } y0)) \text{ AND } (x1 \text{ XOR } y1)))) \text{ AND } (x2 \text{ XOR } y2))))$													
$cout = (x3 \text{ AND } y3) \text{ OR } (((x2 \text{ AND } y2) \text{ OR } ((x1 \text{ AND } y1) \text{ OR } ((x0 \text{ AND } y0) \text{ OR } (cin \text{ AND } (x0 \text{ XOR } y0)) \text{ AND } (x1 \text{ XOR } y1)))) \text{ AND } (x2 \text{ XOR } y2)))) \text{ AND } (x3 \text{ XOR } y3))$													

When the Boolean function is applied to the block diagram, there will be 8 XOR gates, 8 AND gates, and 4 OR gates. Which makes up 4 One-Bit Full Adder and created into a Full Adder

component.

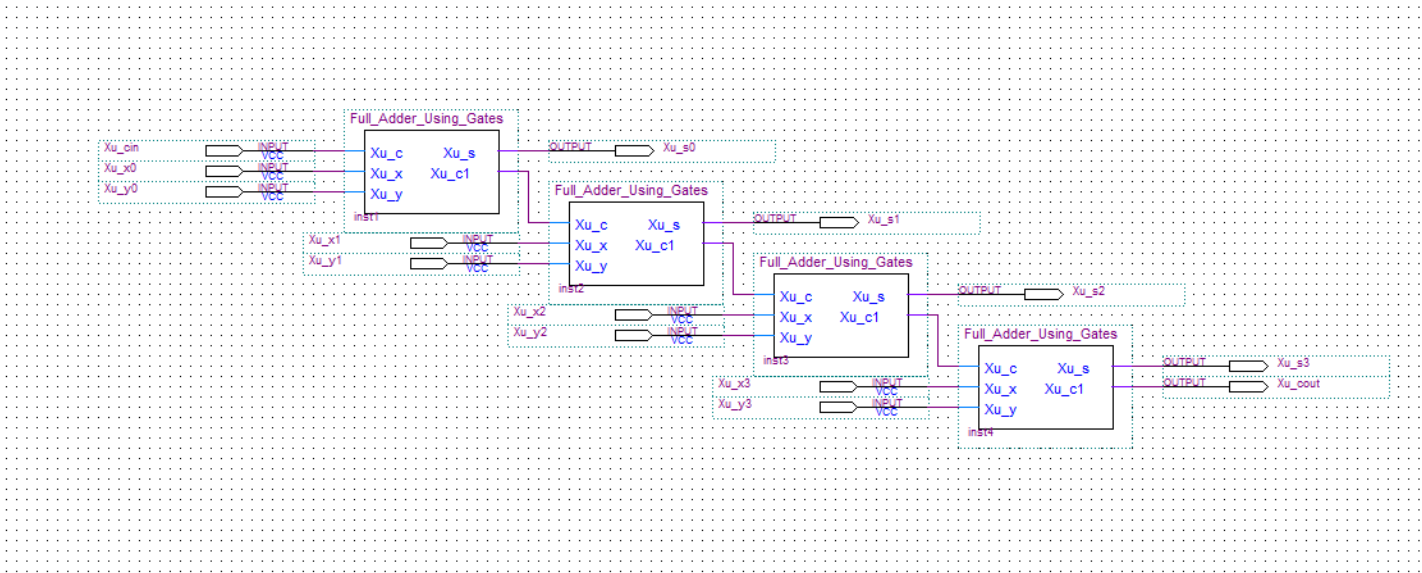


Figure 18: Block diagram of 4-Bit Full Adder using Full Adder as a component

5.2 Simulation

In the simulation, values of 0 and 1 will be the input for cin, x0, x1, x2, x3, y0, y1, y2, y3, cout, s3, s2, s1, and s0 at varying intervals. Input cin will value of 0 and 1 at each 800ns interval. Input x0, x1, x2, and x3 will have value of 0 and 1 at each 200ns interval. Input y0, y1, y2, and y3 will have values of 0 and 1 at each 400ns interval.

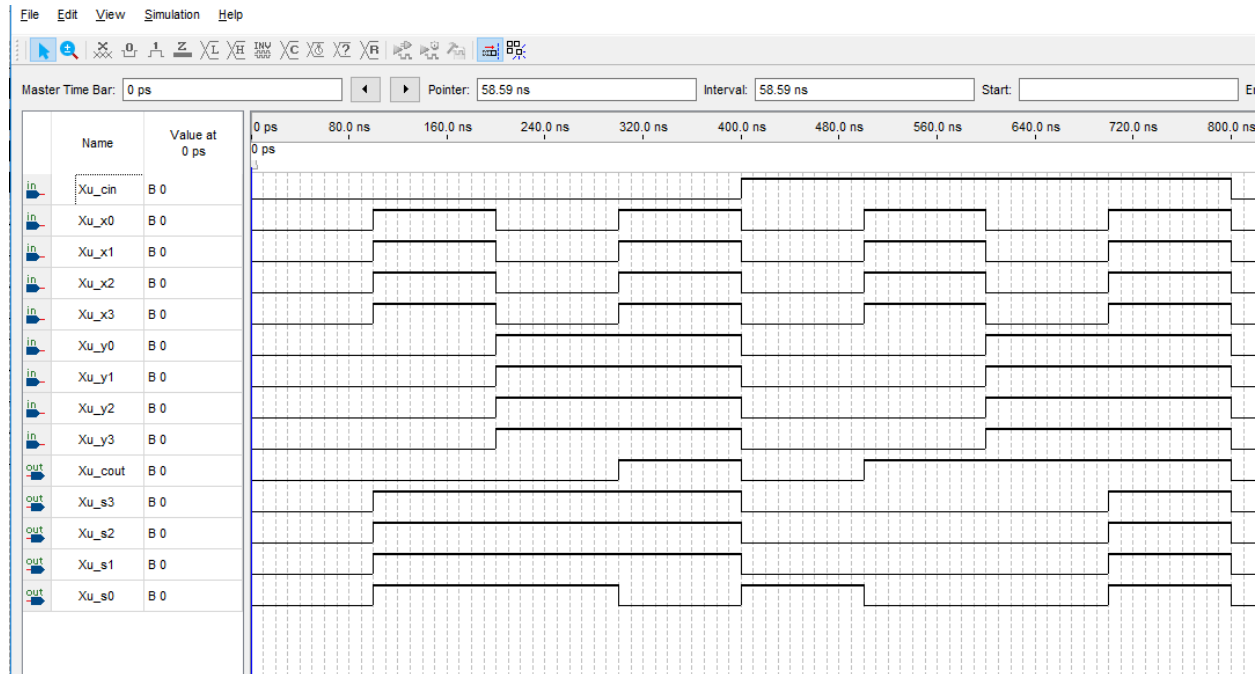


Figure 19: Vector waveform simulation corresponding to 4-Bit Full Adder using Full Adder

We can observe that when input cin is 0 and either all x inputs or y inputs are 1 the output of all s will be 1 and cout will be 0. When cin is 0 and all x and y inputs are 1 the output will be s0 is 0 and all other s output and cout will be 1. When cin is 1 and either x inputs or y inputs are 1 the output will be s0 and cout will be 1 and all other s output will be 0. When all inputs are 1 then all outputs are 1. The results of the simulation corresponds to the truth table that defines the 4-Bit Full Adder using 1-Bit Full Adder as a component.

5.3 Demonstration

The inputs and outputs are assigned to pins on the DE1-SoC Board.

Xu_x0 is assigned to SW[0] which is PIN_AB12

Xu_x1 is assigned to SW[1] which is PIN_AC12

Xu_x2 is assigned to SW[2] which is PIN_AF9

Xu_x3 is assigned to SW[3] which is PIN_AF10

Xu_y0 is assigned to SW[4] which is PIN_AD11

Xu_y1 is assigned to SW[5] which is PIN_AD12

Xu_y2 is assigned to SW[6] which is PIN_AE11

Xu_y3 is assigned to SW[7] which is PIN_AC9

Xu_cin is assigned to SW[8] which is PIN_AD10

Xu_s0 is assigned to LEDR[0] which is PIN_V16

Xu_s1 is assigned to LEDR[1] which is PIN_W16

Xu_s2 is assigned to LEDR[2] which is PIN_V17

Xu_s3 is assigned to LEDR[3] which is PIN_V18

Xu_cout is assigned to LEDR[4] which is PIN_W17

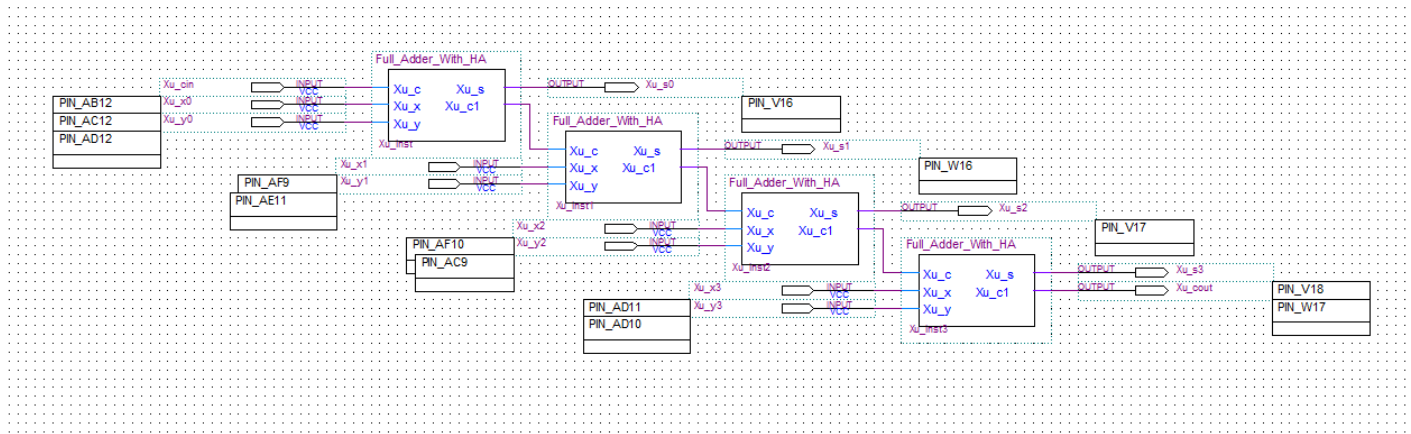


Figure 20: PIN assignments of 4-Bit Full Adder using Full Adders circuit to DE1-SoC Board

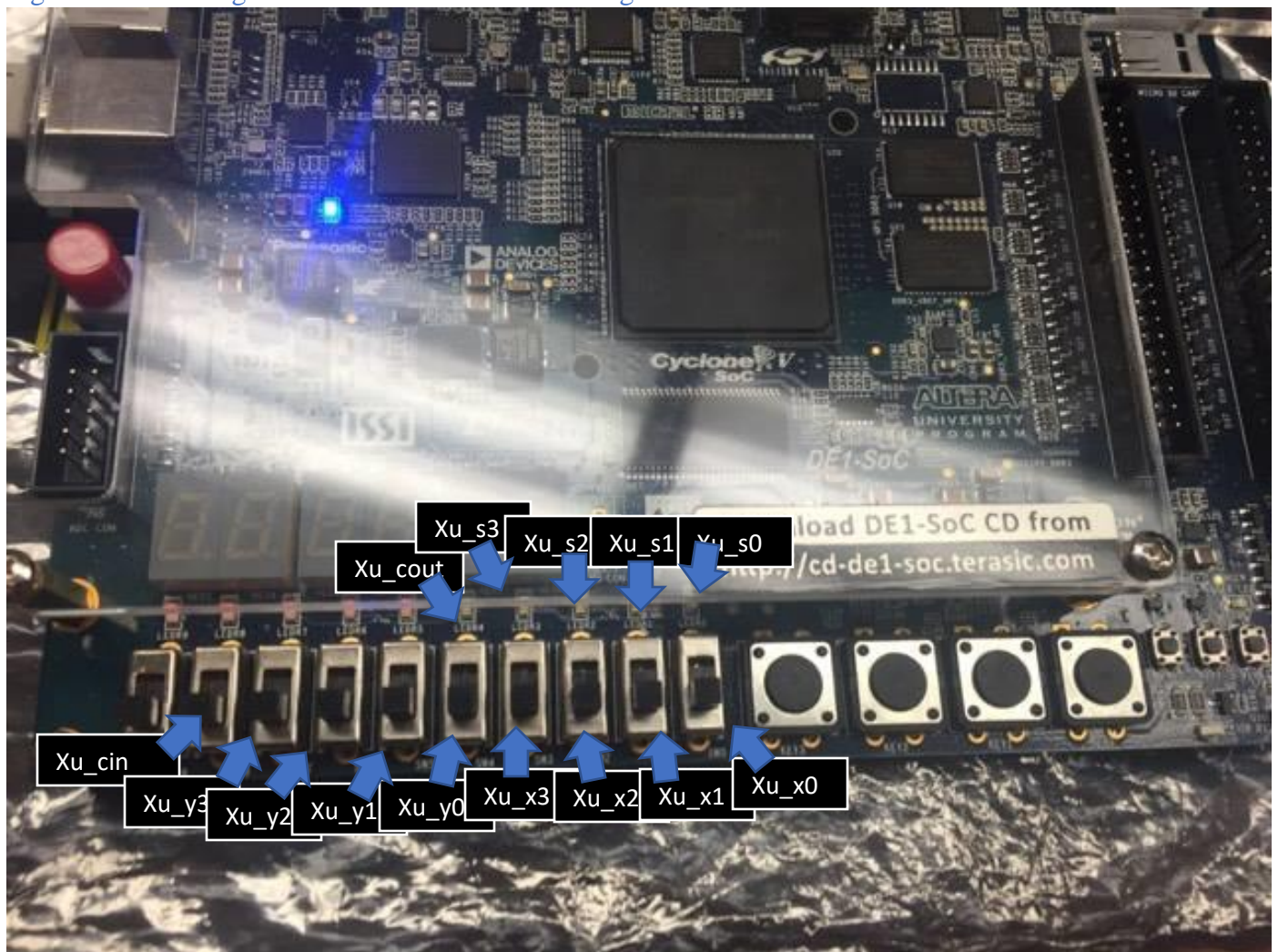


Figure 21: PIN assignment on the DE1-SoC Board



Figure 22: inputs x_0, x_1, x_2, x_3 is 1 (on) y_0, y_1, y_2, y_3 , cin is 0 (off) outputs s_0, s_1, s_2, s_3 is 1 (on) and cout is 0 (off)



Figure 23: inputs $x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3$ is 1 (on) cin is 0 (off) outputs s_0 is 0(off) s_1, s_2, s_3 and cout is 1 (on)

6. Conclusion

Taking from what we have learned from the master tutorial that explained the use of basic tools for circuit design in Quartus Prime to create new project, new block diagram file, designing digital circuit, creating symbol file with designed circuit, assigning pin to inputs and outputs, running waveform simulation corresponding to designed circuit, and loading circuit to the DE1-SoC board and test designed circuit. We have successfully designed circuits for Half-Adder, One Bit Full Adder using Gates only, One Bit Full Adder using Half-Adder as component, and Four Bit Full Adder using Full Adder as component in Quartus Prime, also ran simulations corresponding to truth table that specifies the circuit designed and successfully loading the designed circuit onto the DE1-SoC board which results corresponding to the simulation for the circuit designed.