Lebanese American University

Digital Systems Lab

Section 33

Laboratory manual 1

Nour Kachmar: 201902597

Youssef Kourani: 202004265

15/09/2023

Table of Contents

List of Figures

List of Tables

Introduction

In this lab report, we will build our own 1-bit half adder, 1-bit full adder and 3-bit multiplier in multiple steps. First, we will design those using truth tables, K-maps, logic Boolean expressions or just simple arithmetic operations according to what is needed. We will then implement all of our designs on Altera Quartus II using block diagrams, and finally simulate those using waveforms.

Experiment 1

In this experiment, we are going to design a 1-bit half Adder by using the truth table then draw its block diagram using Altera Quartus II. According to the equation we will get from the truth table, we will deduce what components to use in the design process. Then, we're going to simulate our design using a waveform. Finally, we will add this 1-bit half Adder to a symbol file so we can use it in the next experiment.

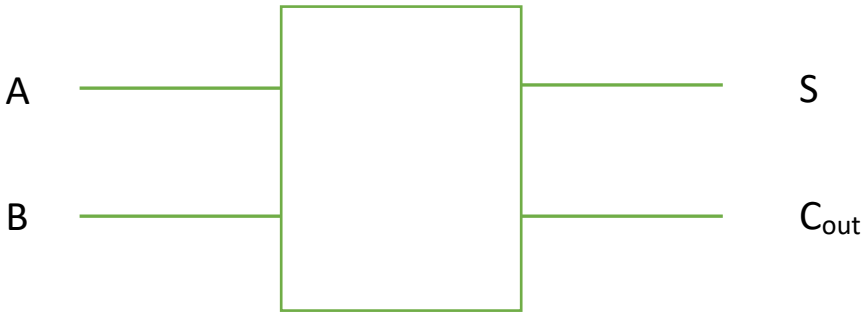

Figure 1- 1-bit half Adder

In figure 1, we displayed the input-output relation in the 1-bit half Adder to use in the truth table below. Providing two inputs A and B yields two outputs, S which is the sum of the two input bits and C_{out} which is the carry out.

Truth table

Table 1-1-bit half Adder truth table

A	B	S	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-map / equations

From the truth table we can deduce that $S = A'B + AB' = A \oplus B$ and $C_{out} = AB$.

PS: we did not need any K-map in this scenario.

Block diagram

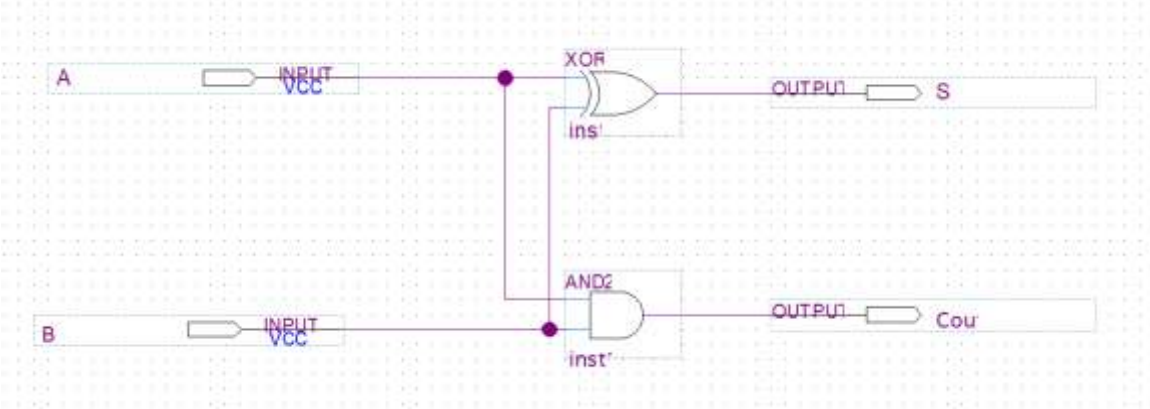

Figure 2- 1-bit half Adder block diagram

In this block diagram, based on the equation we deduced from the truth table, we added two inputs that were first connected to the XOR gate and to the and gate. The output of the XOR gate is S and the output of the AND gate is C_{out}.

Simulation

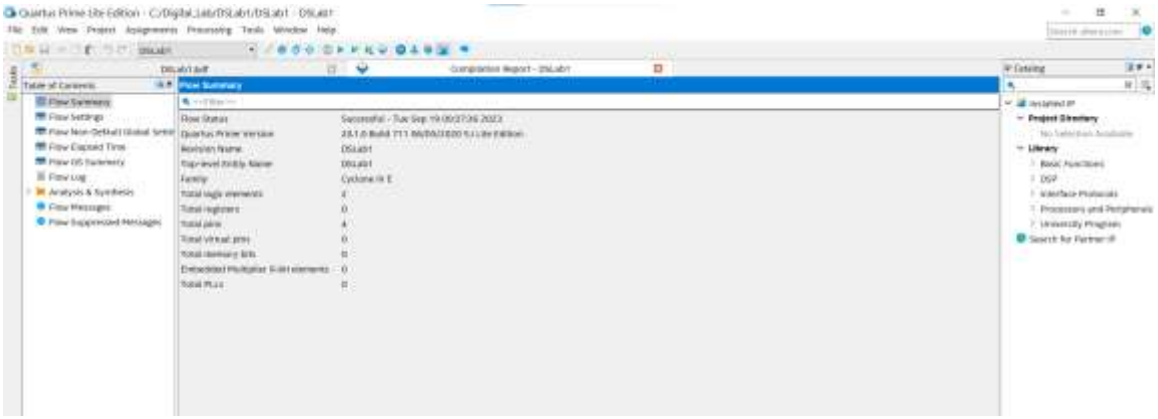

Figure 3- 1-bit half Adder compilation report

Figure 3 shows that the compilation of the 1-bit half Adder named “DSLab1” was successful.

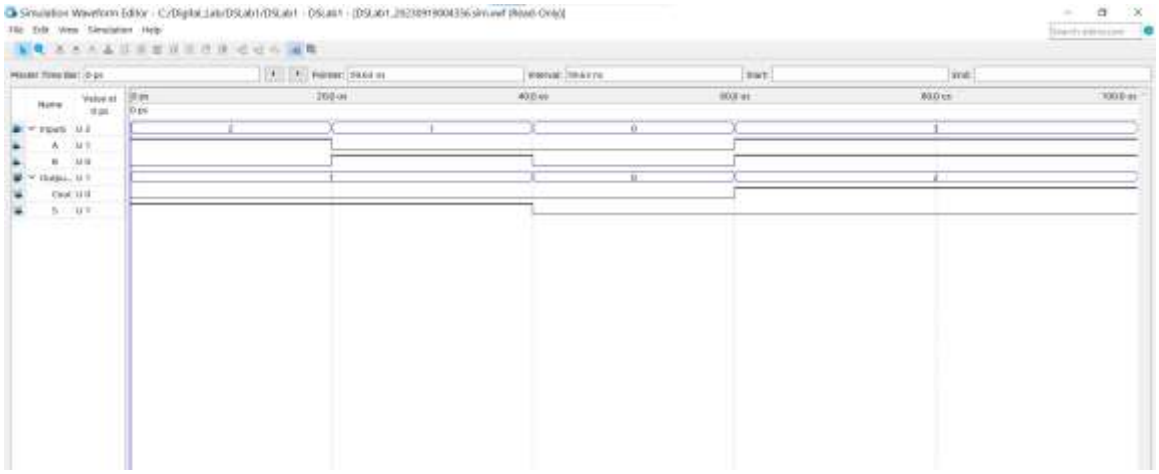

Figure 4- 1-bit half Adder simulation report

Figure 4 shows the simulation of the 1-bit half Adder named “Waveform”. It shows 2 inputs in unsigned decimal grouped as “Inputs” with different values each 20 µs being simulated over a whole interval of 100 µs (maximum allowed by Quartus’ free version). For example, from 0 µs to 20 µs, we can see that A is equal to 1 and B is equal to 0. In unsigned decimal, this corresponds to the value of 2 shown by the” inputs” group on top. Adding 1 and 0 should give us a sum of 1 and a carry out of 0. Indeed, looking at the first interval, we can observe these values in C_{out} representing the carry out and S representing the sum. After the simulation, we created a symbol file of the 1-bit half-Adder so that we don’t have to repeat the whole design in the next steps.

Experiment 2

In this experiment, we’re going to design a 1-bit full Adder by using the truth table, logic Boolean laws and K-maps according to what is needed, and then draw its block diagram using Altera Quartus II. According to the equation we will get from the processes stated, we will deduce what components to use in the design process. We will then simulate our design through a waveform and finally add the whole 1-bit full Adder to a symbol file so that we can use it later on.

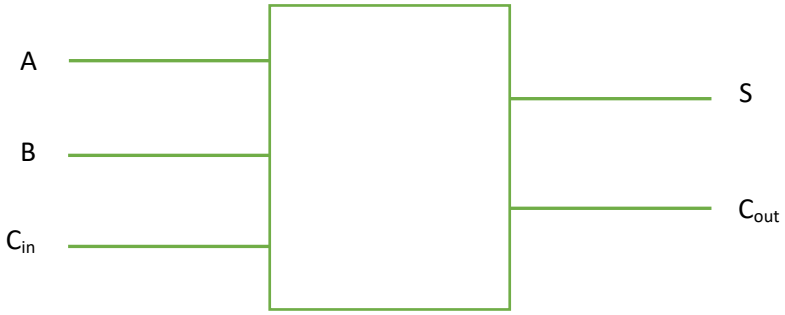

Figure 5- 1-bit full Adder

In figure 5, we displayed the input-output relation in the 1-bit full Adder to use in the truth table below. Providing three inputs A and B and C_{in} yields two outputs, S which is the sum of the three input bits and C_{out} which is the carry out.

Truth table

Table 2- 1-bit full Adder truth table

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

From the truth table, by checking when S is equal to 1 and C_{out} is equal to 1, we come up with the following equations:

$$S = A'B'Cin + A'BCin' + AB'Cin' + ABCin$$

$$Cout = A'BCin + AB'Cin + ABCin' + ABCin$$

K-map / equations

We will simplify the equations as follows:

$$S = A'B'Cin + A'BCin' + AB'Cin' + ABCin$$

$$S = Cin(A'B' + AB) + Cin'(A'B + AB')$$

We can realize that $A'B' + AB$ is none other than a XNOR gate and that $A'B + AB'$ is none other than a XOR gate.

Now suppose C_{in} is denoted as X and A $\oplus$B is denoted as Y which yields:

$$S = XY' + X'Y$$

Again, this is a XOR gate. Our final equation will be:

$$S = Cin \oplus (A \oplus B)$$

To get C_{out}, we will use a K-map which will be filled up by the values in the truth table.

Table 3-K-map for C_{out} in 1-bit full Adder

AB \ C_{in}	0	1
00	0	0
01	0	1
11	1	1
10	0	1

$Cout = AB + BCin + ACin$

.

Assume we have three inputs: A, B and C_{in} . A and B are going through the first 1-bit half adder yielding 2 outputs: $S1 = A \oplus B$ and $Co1 = AB$. The second 1-bit half adder has as inputs S_1 and $C_{in,}$ yielding to two outputs: $S = Cin \oplus (A \oplus B)$ which is also the direct output of the 1-bit full adder. The second output $Co2 = S1.Cin = (A \oplus B)Cin$. From the K-map, Cout differs from Co2. We will manipulate Cout through logical operations to relate Co1 and Co2 to Cout.

$Cout = AB + BCin + ACin$

Using (X+X')=1, we will multiply AB by Cin + Cin'

$Cout = AB(Cin + Cin') + ACin + BCin$

$Cout = ABCin + ABCin' + ACin + BCin$

Factorizing by ACin

$Cout = ACin(B + 1) + ABCin' + BCin$

Since B+1=1,

$Cout = ACin + BCin + ABCin'$

Multiplying BCin by A+A',

$Cout = ACin + BCin(A + A') + ABCin'$

$Cout = ACin + ABCin + A'BCin + ABCin'$

Factorize by ACin and B,

$Cout = ACin(1 + B) + B(A'Cin + ACin')$

Since 1+B=1, and A'Cin+ACin' is a XOR gate,

$Cout = AB + Cin(A \oplus B) = Co1 + Co2$

Meaning that we need to add an OR gate between Co1 and Co2 to get Cout.

Block diagram

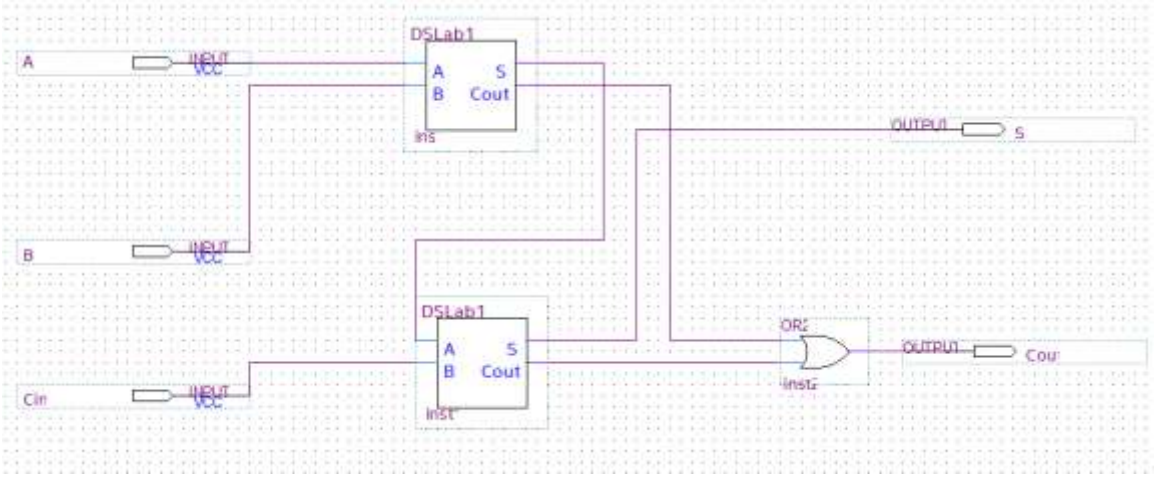

Figure 6- 1-bit full adder block diagram

Simulation

Figure 7- 1-bit full Adder compilation report

Figure 7 shows that the compilation of the 1-bit full Adder named "Block2" was successful.

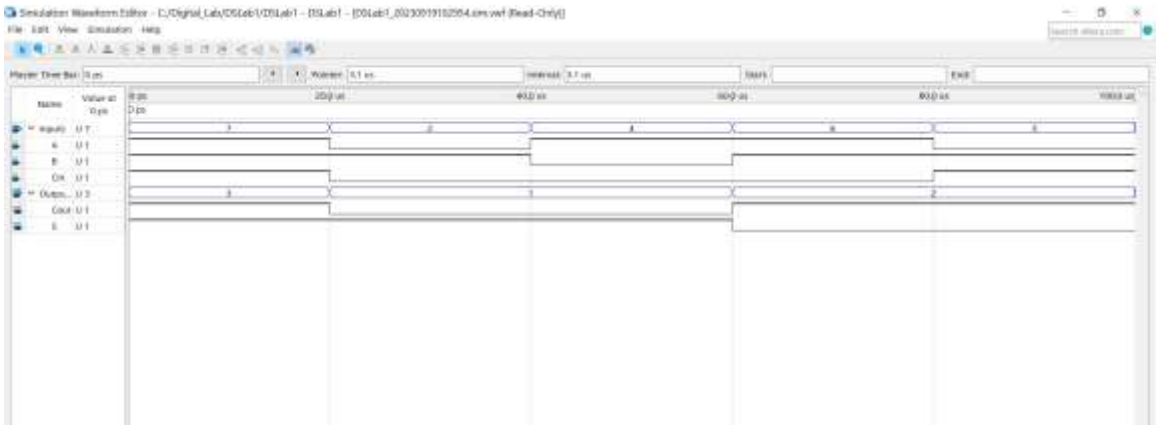

Figure 8- 1-bit full Adder simulation report

Figure 8 shows the simulation of the 1-bit full Adder named “Waveform1”. It shows 3 inputs in unsigned decimal grouped as “Inputs” with different values each 20 µs being simulated over a whole interval of 100 µs. For example, from 0 µs to 20 µs, we can see that A is equal to 1, B is equal to 1 and C is also equal to 1. This is a 7 in unsigned decimal (111) as shown by the value of “Inputs” on the top left of the screen. In unsigned decimal, Adding 1 three times should give us a sum of 1 and a carry out of 1. Indeed, looking at the first interval, we can observe these values in C_{out} representing the carry out and S representing the sum. After the simulation, we created a symbol file of the 1-bit full-Adder so that we don’t have to repeat the whole design in the next steps.

Experiment 3

In this experiment, we’re going to design a 3-bit multiplier by using our previously designed 1-bit full adder, and then draw its block diagram using Altera Quartus II. We will then simulate our design through a waveform.

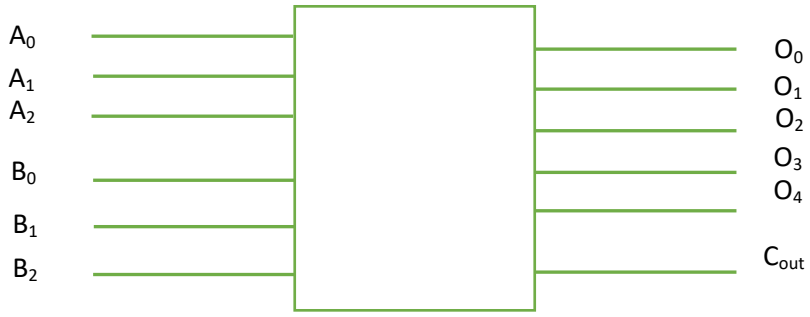

Figure 9- 3-bit multiplier

In figure 9, 2 3-bit inputs A and B with respective input bits A_2, A_1, A_0 and B_2 , B_1, B_0 are being fed to a 3 bit multiplier which will output 6 bits, the first 5 being the magnitude of the output and the most significant bit being the carry out.

Truth table

In this case, doing a truth table will take 2^6 rows, in other terms 64 rows which is not feasible on paper. Rather, we will use basic multiplication skills to be able to determine the design of our 3-bit multiplier.

K-map / equations

Also, no K-maps will be needed since we will get the equations for each output from the multiplication of both A and B.

Block diagram

To draw the block diagram, we need to set up a design first through equations for each output bit. We will multiply A and B and see analyze the equations:

Figure 10- 3-bit multiplication with inputs A and B

From the multiplication, we were able to obtain the above equations for each output bit as shown in Figure 10. By analyzing these equations and trial and error, we were able to obtain a design using 3 full adders and 3 half adders.

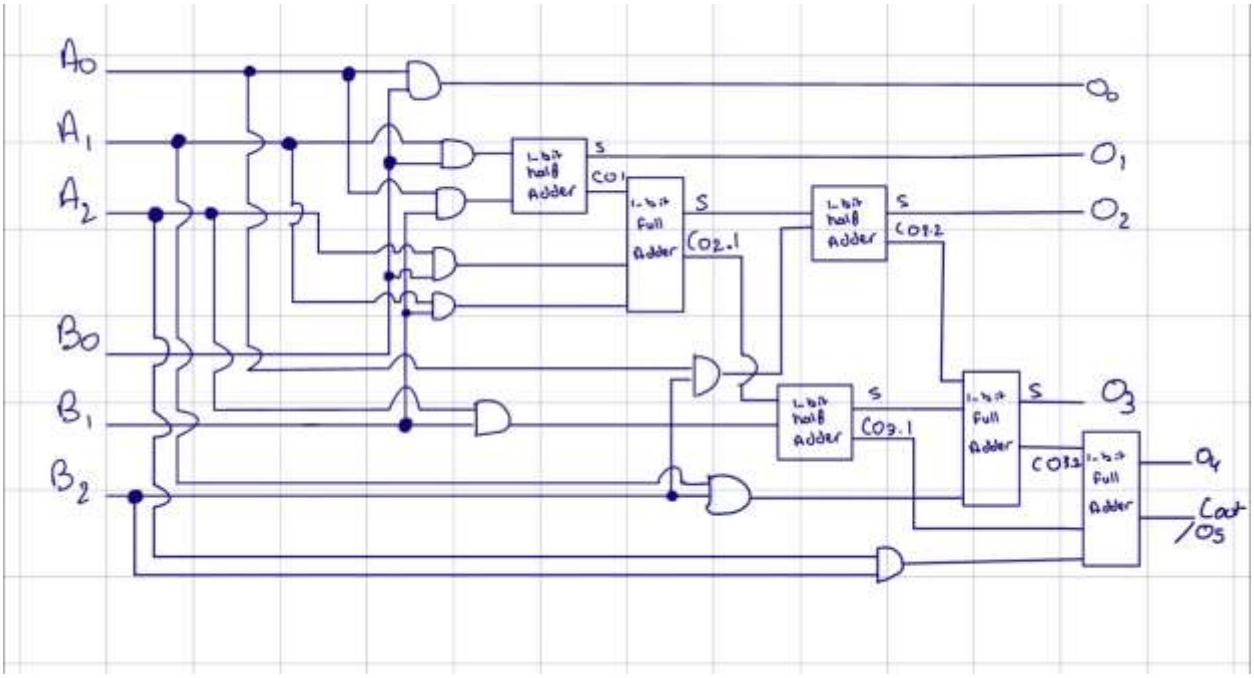

Figure 11- 3-bit mulitplier block diagram design

For O_0, we just needed to use an AND gate with inputs A_0 and B_0 to obtain it.

For O_1, since we have 2 inputs A_1B_0 and A_0B_1, we inputted them to a half bit adder and obtained a carry out CO_1 and O_1.

For O_2, we now have 3 inputs: CO_1 from the previous output, A_2B_0 and A_1B_1. We will input these 3 outputs to a full adder which will give a sum S and a carry output $CO_{2.1}$. However, we did not add all the products so this is not O_2. We still need to add the sum S from the full adder to A_0B_2. Having 2 inputs S and A_0B_2, we will use a half adder which will yield the correct output O_2 and a second carry $CO_{2.2}$.

For O3, we need to add A_2B_1 to the first carry from the previous output named $CO_{2.1}$. Having 2 inputs, we will use a half adder which will yield 2 outputs S and $CO_{3.1}$. The S is not the final output since we still need to add to it the second previous carry $CO_{2.2}$ and A_1B_2. Having 3 inputs, we will use a full adder which will yield the correct output O_3 and a carry out $CO_{3.2}$.

For O_4, we have 3 inputs: $CO_{3.1}$, $CO_{3.2}$ and A_2B_2. Thus, we will use a full adder which will yield the correct output O_4 and a carry out bit.

Being not added to anything, the previous carry bit obtained will be none other than O_5.

In summary, this process explain how we designed a 3-bit multiplier using 3 1-bit full adders and 3 1-bit half adders without having to compute any truth table/K-map, just by using basic multiplication skills.

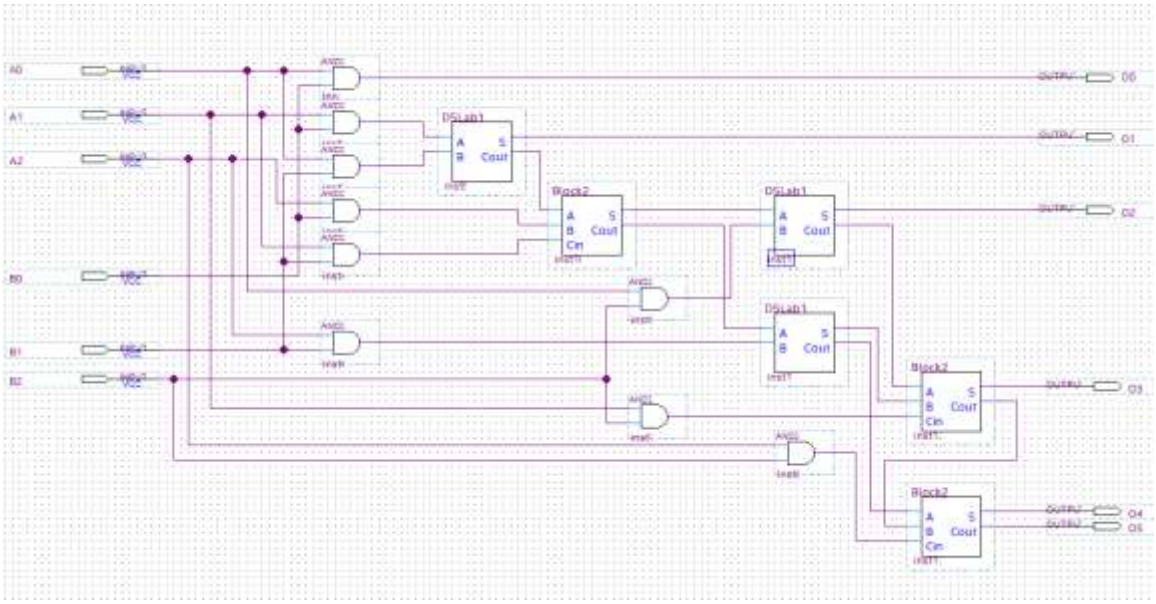

Figure 12- 3-bit multiplier block diagram on Quartus

This is the block diagram design explained before implemented on Quartus II. Please note that the block symbol “DSLab1” refers to a 1-bit half Adder and the block symbol “Block2” refers to a 1-bit full adder.

Figure 13- 3-bit multiplier compilation report

Figure 13 shows that the compilation of the block diagram of the 3-bit multiplier was a success. We will know proceed to the simulation.

Simulation

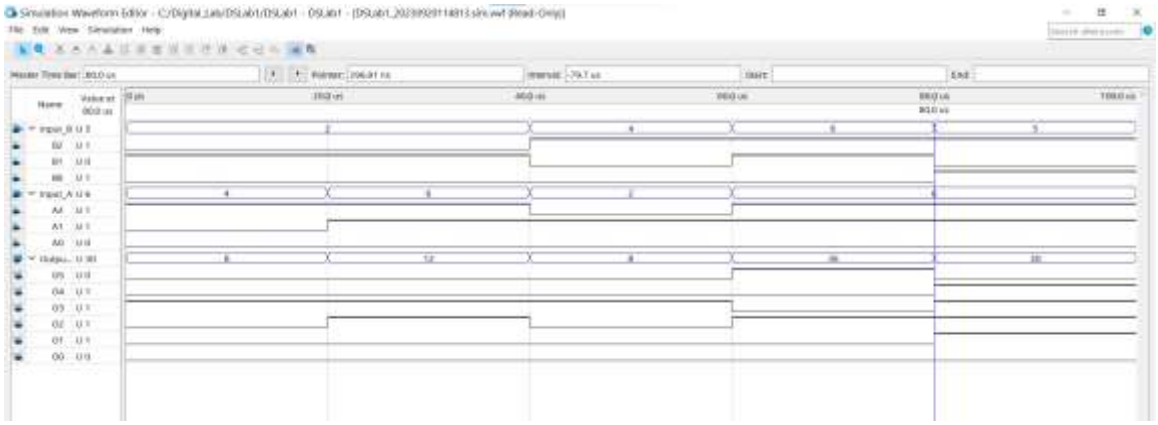

Figure 14: 3-bit multiplier simulation report

Figure 14 shows the simulation of the 3-bit multiplier through a waveform which we named "Waveform3". For example, in the interval 80 µs to 100 µs, we can observe that for 2 inputs A and B of respective values 6 (represented by bits A_0 , A_1 and A_2 as 011 which is a 6 in decimal) and 5 (represented by bits B_0, B_1, and B_2 as 101 which is a 5 in decimal), we were able to obtain an output of 30 (represented by bits O_0, O_1, O_2, O_3, O_4 and O_5 as 011110 which is 2+4+8+16= 30 in decimal); thus proving that our simulation was a successful obtaining correct outputs.

Conclusion

To sum up our work, we were able to design a 1-bit half Adder, a 1-bit full Adder using the previously generated 1-bit half Adder and a 3-bit multiplier using both the 1-bit half Adder and 1-bit full Adder. Finally, we were able to check the correctness of our designs through simulations on Altera Quartus II.