Lebanese American University

Digital Systems Lab

Section 33

Laboratory manual 2

Nour Kachmar: 201902597

Youssef Kourani: 202004265

04/10/2023

Table of Contents

List of Figures

List of Tables

Introduction

In this lab report, we are going to design a BCD-7-segment display decoder by filling up its truth table, obtaining simplified equations for each segment of the BCD-7-segment display decoder from K-Maps, implementing them on Quartus and simulating the whole BCD-7-segment display decoder on waveforms to check if the outputs are correct. Moreover, we will implement a 2-3 bits adder display decoder which uses a 3 bit adder and three BCD-7-segment display decoder. Finally, we will test the 2-3 bits adder display decoder using the FPGA.

Experiment 1

In this experiment, we are going to design a BCD to 7-segment Display Decoder by using the truth table then drawing 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 BCD to 7-segment Display Decoder to a symbol file so we can use it in the next experiment.

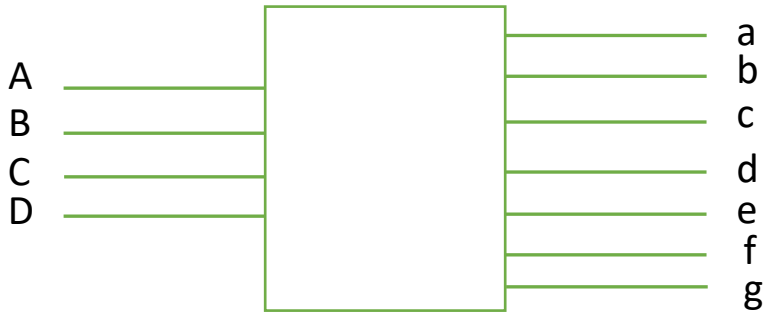


Figure 1- BCD to 7-segment Display Decoder

In figure 1, we displayed the input-output relation in the BCD to 7-segment Display Decoder to use in the truth table below. Providing four inputs A, B, C and D yield seven outputs a, b, c, d, e, f and g.

Truth table

Table 1- BCD to 7-segment Display Decoder truth table

A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	1
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

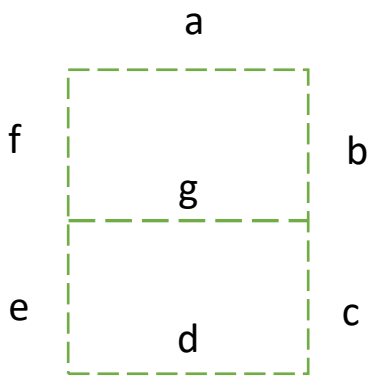


Figure 2- BCD 7-segment representation

To fill the truth table, we used the representation shown in figure 2. Note that numbers ranging from 10 to 15 were represented in a hexadecimal format. Moreover, we represented B as b and D as d to not confuse them with 8 and 0 respectively.

K-map / equations

We will use the values obtained in the truth table to get simplified equations for each output.

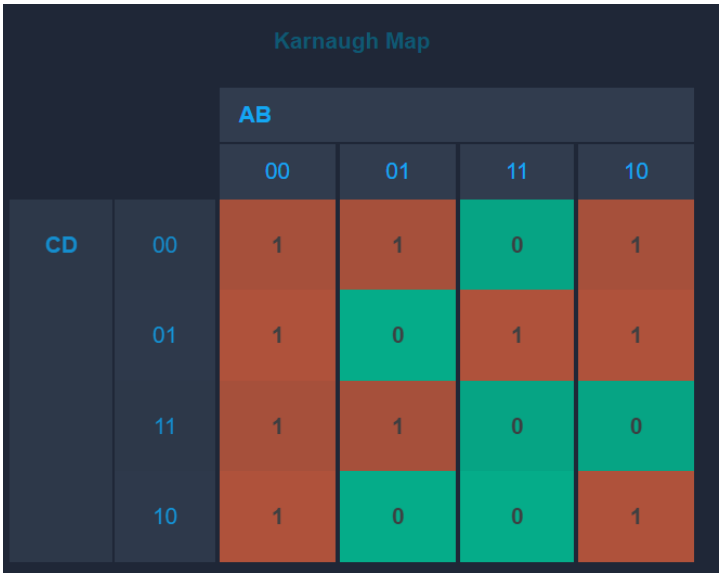
Table 2- K-map of output a

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

We can see that the output a is: $a = B'D' + BC + A'C + AD' + A'BD + AB'C'$

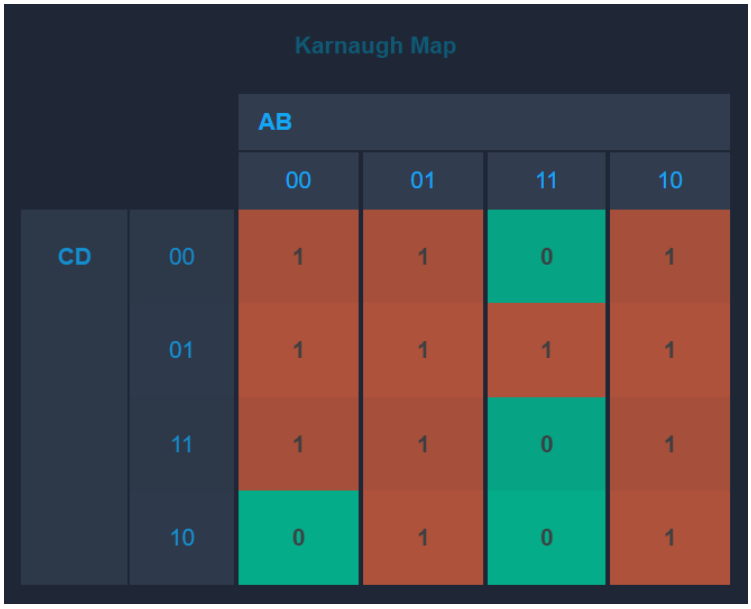
NOTE: We were told in class by our instructor that only one k-map was mandatory by hand. For the others, we can simulate those using online K-map Simulators.

Table 3- K-map for output b



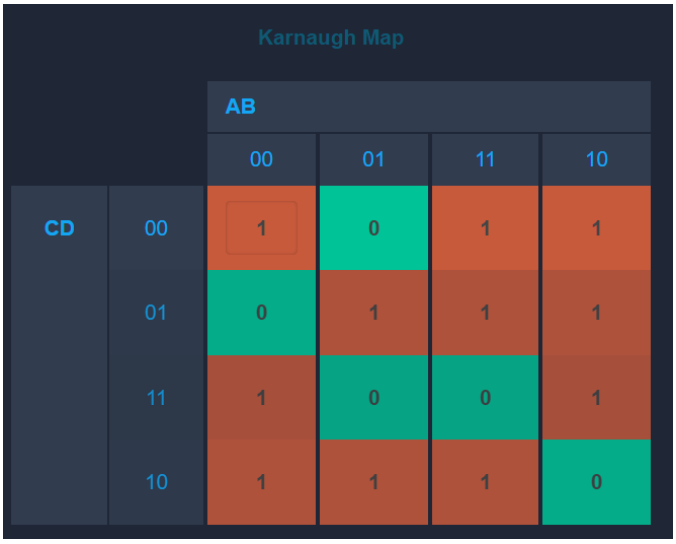
According to the K-map solver, the output b is: $b = B'D' + A'C'D' + A'B' + CDA' + AC'D$

Table 4- K-map for output c



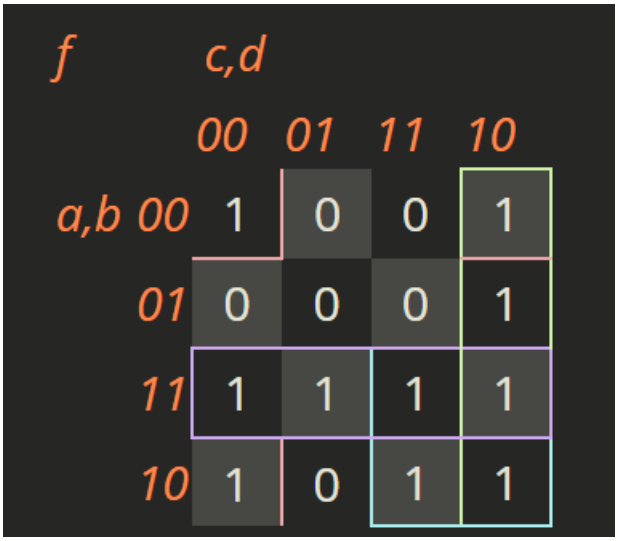
According to the K-map solver, the output c is: $c = A'B + AB' + C'D + A'C' + A'D$

Table 5- K-map for output d



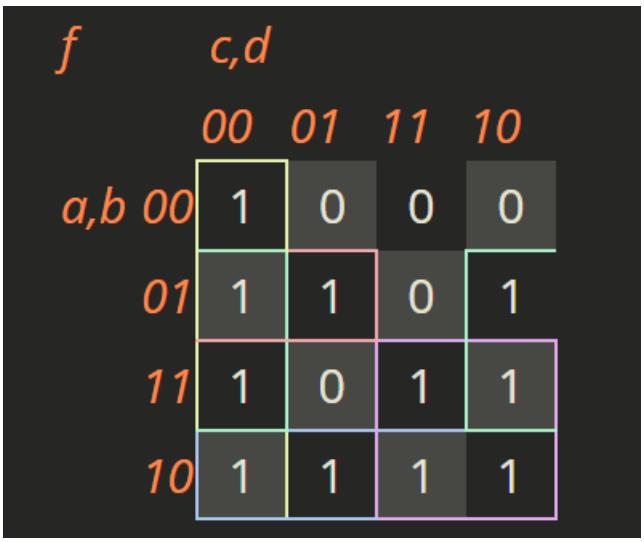
According to the K-map solver, the output d is: $d = AC' + BCD' + B'CD + A'B'D' + BC'D$

Table 6- K-map for output e



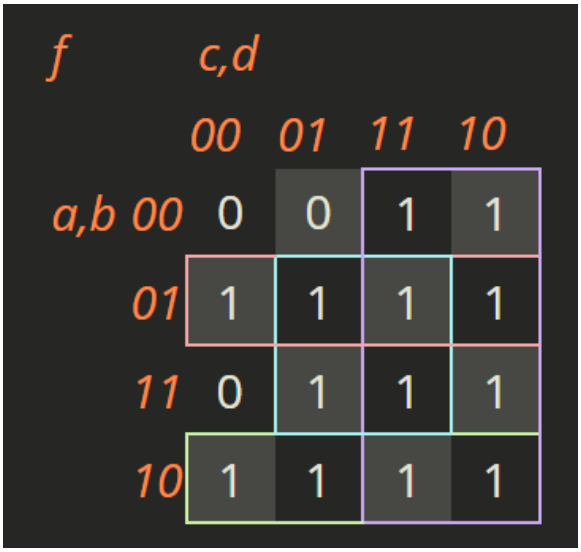
According to the K-map solver, the output e is: $e = B'D' + CD' + AC + AB$

Table 7- K-map for output f



According to the K-map solver, the output f is: $f = C'D' + A'BC' + BD' + AB' + AC$

Table 8- K-map for output g



According to the K-map solver, the output g is: $g = C + A'B + AB' + BD$

Block diagram

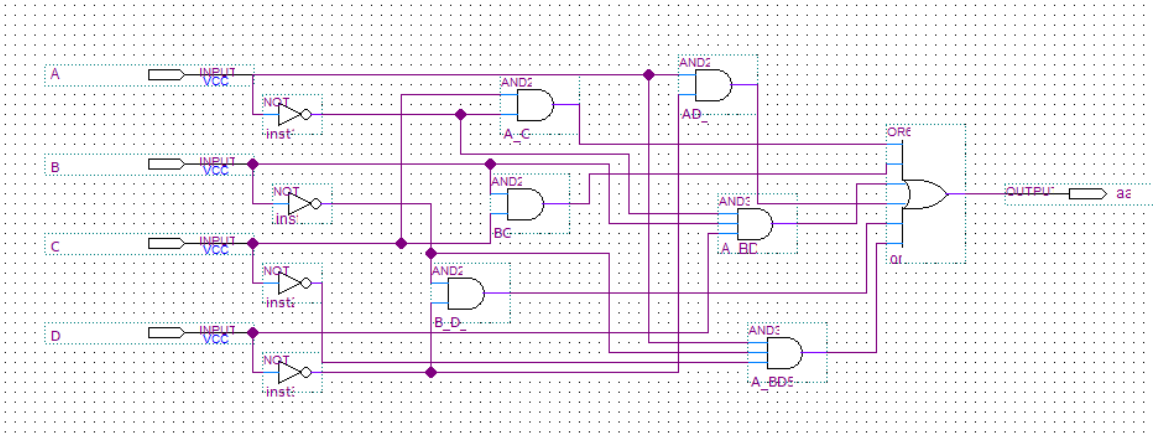


Figure 3- Block Diagram of output a

Figure 3 represents the block diagram implemented using Quartus II of the first output a of our BCD 7-segment display decoder. Note that the file of this block diagram was named "output_a" and we had to name the output in the block diagram "aa" since Quartus II is not case sensitive and we have an input named "A".

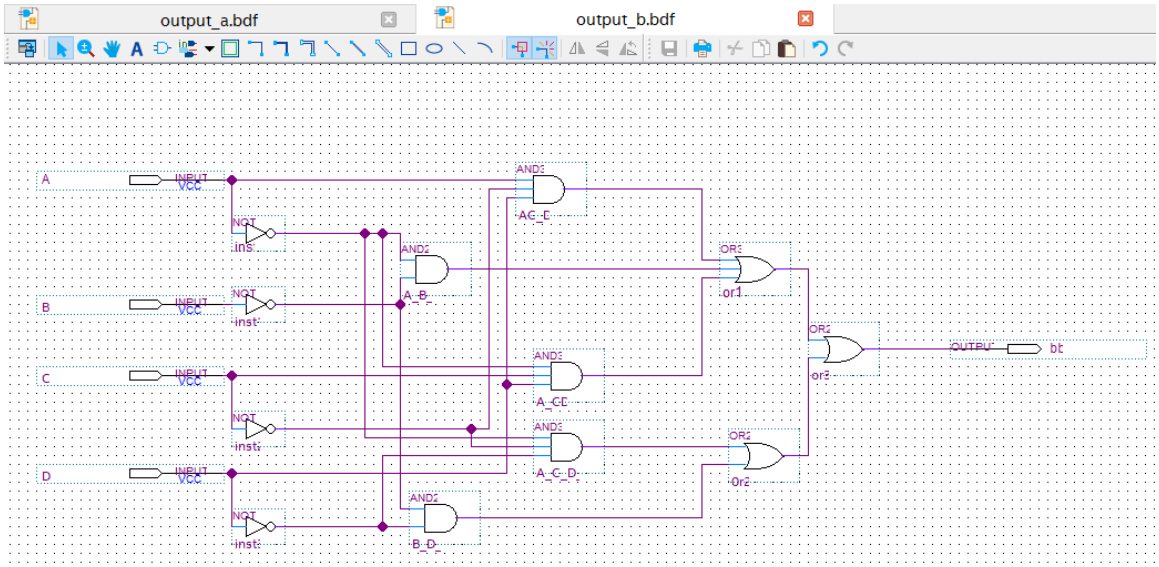


Figure 4- Block Diagram of output b

Figure 4 represents the block diagram implemented using Quartus II of the second output b of our BCD 7-segment display decoder. Note that the file of this block diagram was named "output_b" and we had to name the output in the block diagram "bb" since Quartus II is not case sensitive and we have an input named "B".

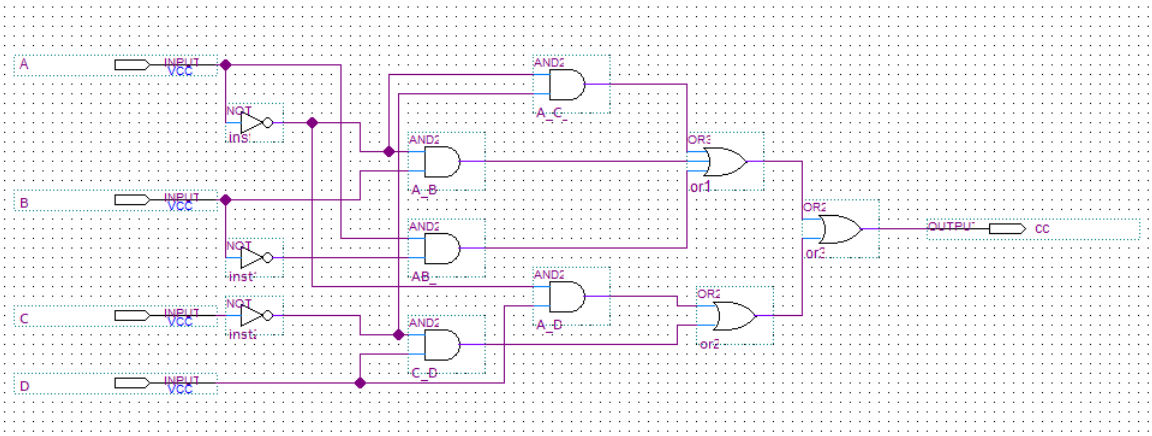


Figure 5- Block Diagram of output c

Figure 5 represents the block diagram implemented using Quartus II of the third output c of our BCD 7-segment display decoder. Note that the file of this block diagram was named “output_c” and we had to name the output in the block diagram “cc” since Quartus II is not case sensitive and we have an input named “C”.

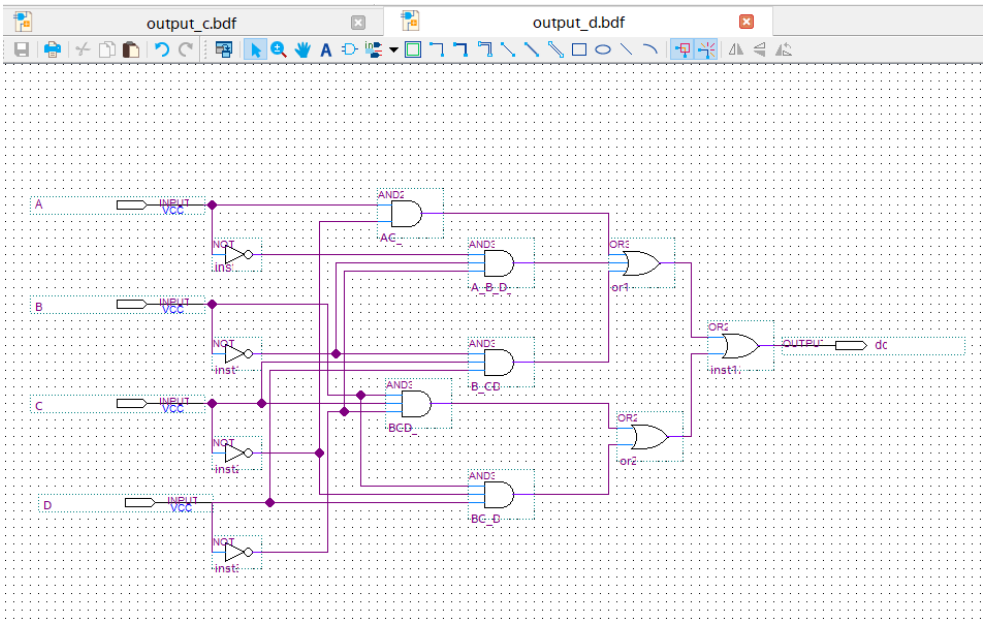


Figure 6- Block Diagram of output d

Figure 6 represents the block diagram implemented using Quartus II of the fourth output d of our BCD 7-segment display decoder. Note that the file of this block diagram was named “output_d” and we had to name the output in the block diagram “dd” since Quartus II is not case sensitive and we have an input named “D”.

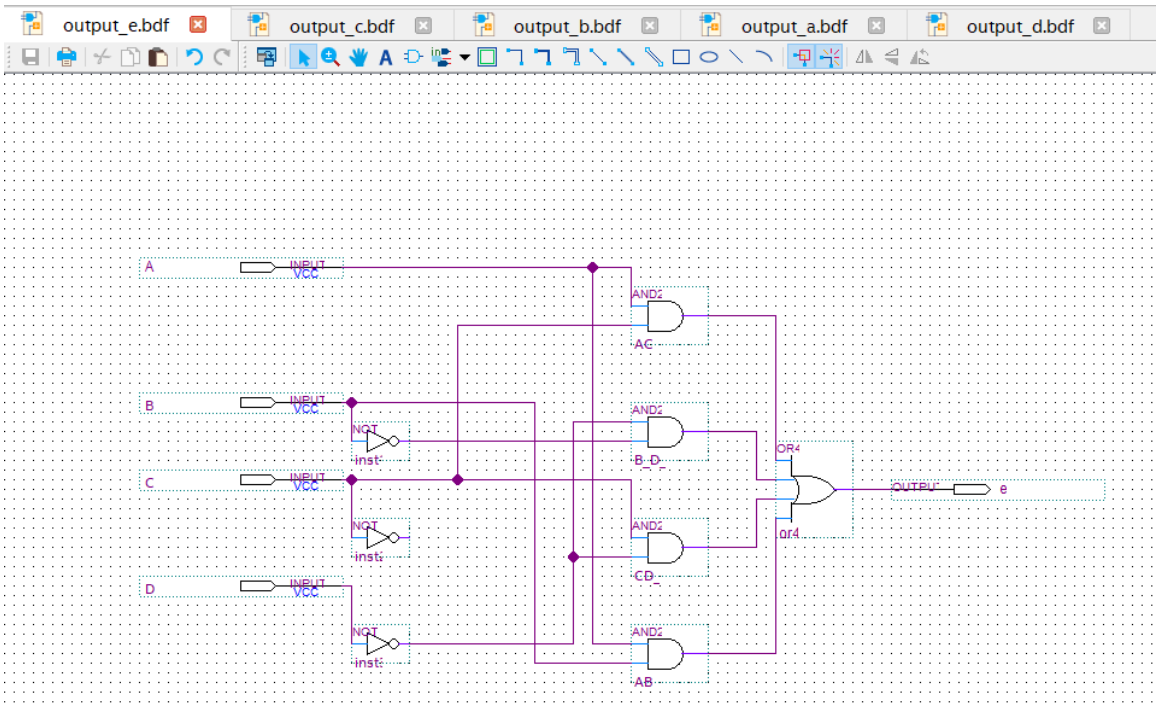


Figure 7- Block Diagram of output e

Figure 7 represents the block diagram implemented using Quartus II of the fifth output e of our BCD 7-segment display decoder. Note that the file of this block diagram was named “output_e”.

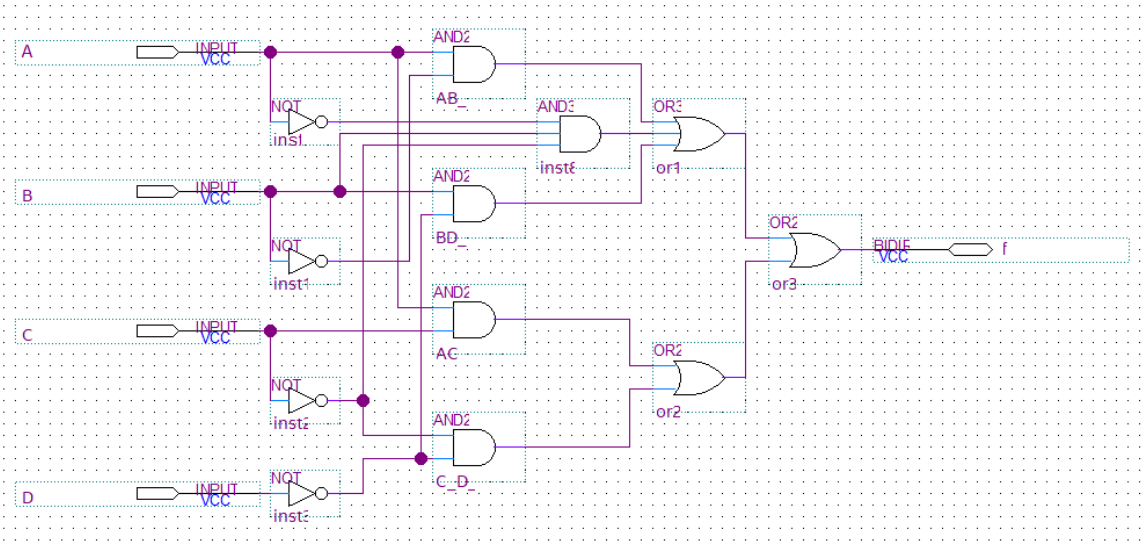


Figure 8- Block Diagram of ouput f

Figure 8 represents the block diagram implemented using Quartus II of the sixth output f of our BCD 7-segment display decoder. Note that the file of this block diagram was named “output_f”.

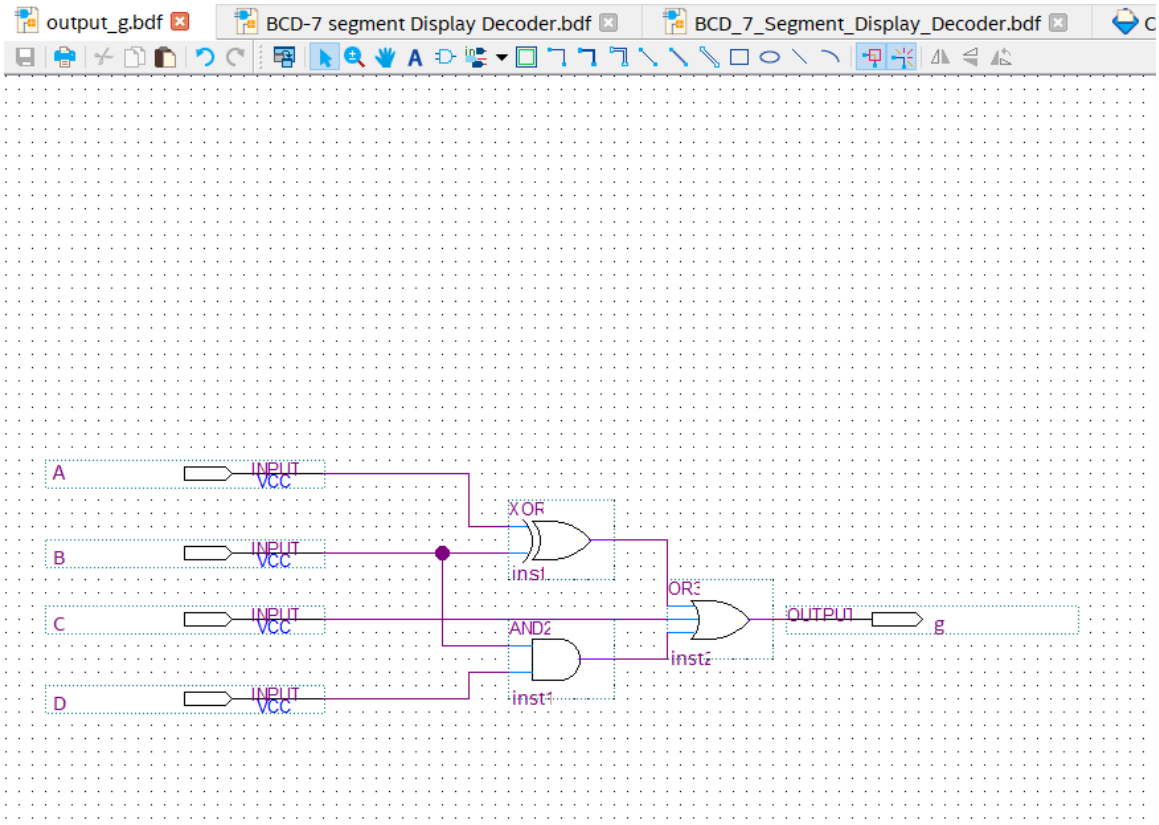


Figure 9- Block Diagram of output g

Figure 9 represents the block diagram implemented using Quartus II of the final seventh output g of our BCD 7-segment display decoder. Note that the file of this block diagram was named “output_g”.

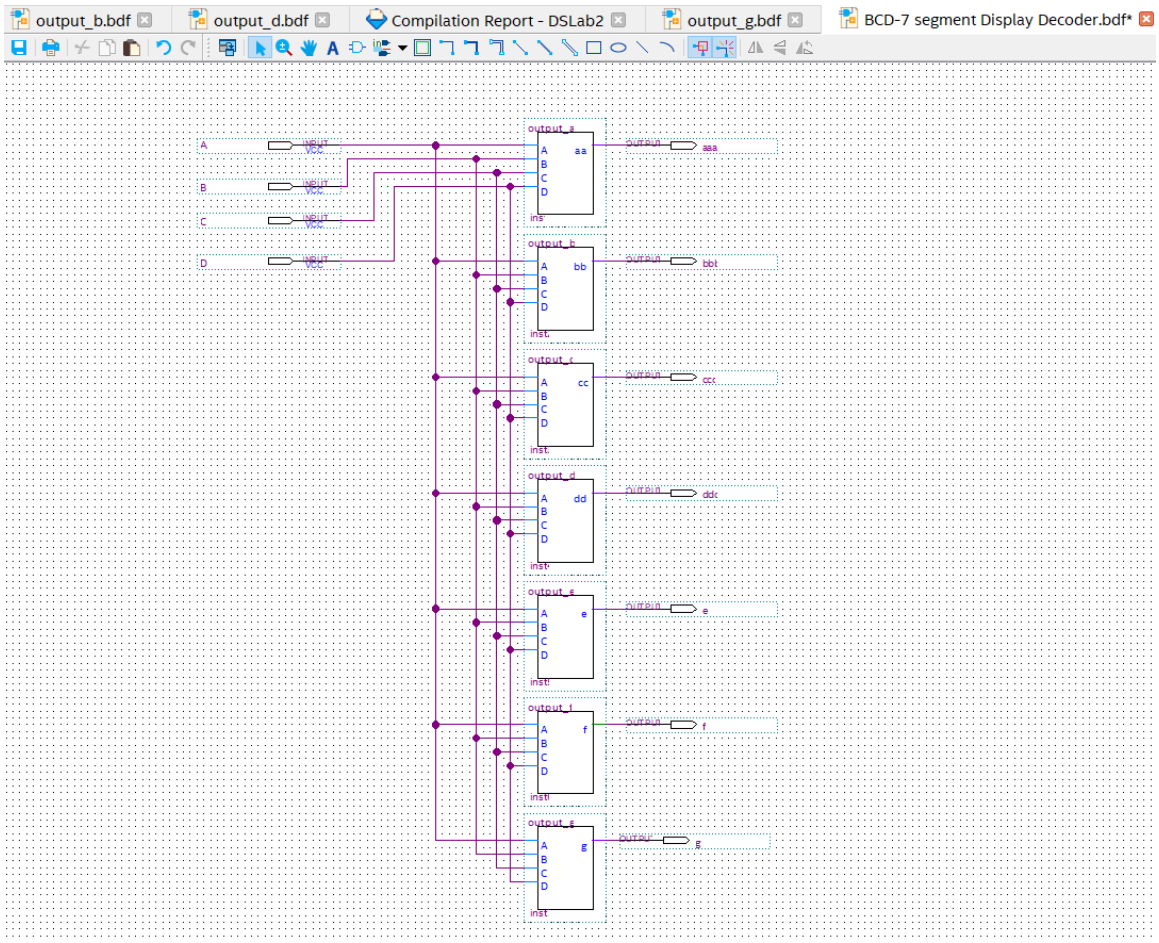


Figure 10- Block Diagram of BCD 7-segment Display Decoder

Figure 10 represents the block diagram implemented using Quartus II of the all the 7 outputs of our BCD 7-segment display decoder. Note that the file of this block diagram was named "BCD_7_Segment_Display_Decoder". Moreover, we created a symbol file for each of our previous outputs to have a scalable design for our BCD 7-segment display decoder.

Simulation

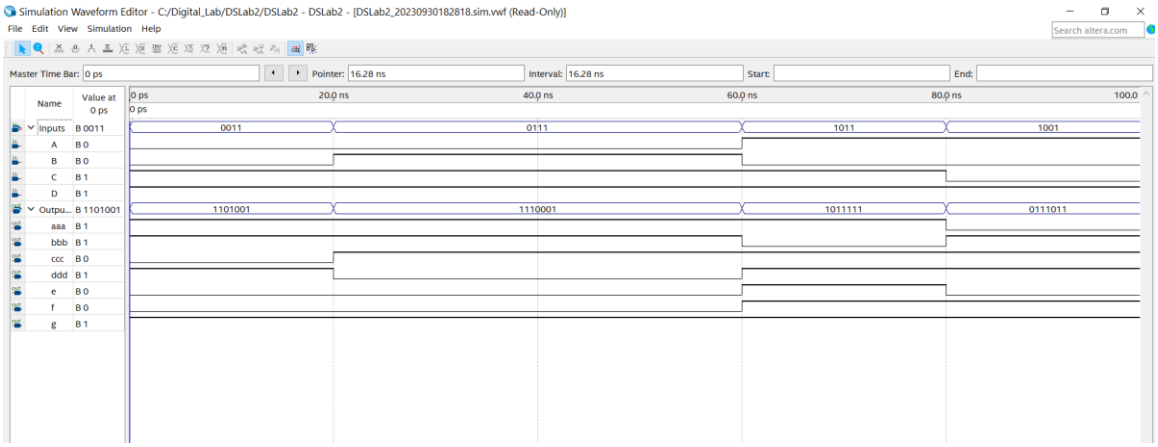


Figure 11- Waveform simulation of the BCD-7-segment Display Decoder

In this waveform named “Waveform1”, we were able to confirm that our BCD 7-segment display decoder gave us accurate values that were shown in our truth table. For example, between 20 ns and 40 ns, the group “Inputs” gives us a value of 0111. In the truth table, when the inputs are 0111, the corresponding output row is 1110001 which correspond to the value shown by the group “Outputs’ testifying for the accuracy of our design.

Experiment 2

In this experiment, we're going to design a system that takes two 3-bit numbers and add them together. The result of the addition will be represented as 4-bit output and then draw its block diagram using Altera Quartus II. We will then simulate our design through a waveform and finally implement it using the board we were given to have a concrete visualization of our outputs,

Truth Table

We did not need any truth table for this part.

K-map / equations

We did not need any K-map for this part.

Block diagram

In our previous Lab (LAB 1), we were able to implement a 1-bit full adder. This lab requires that we implement a 3-bit full adder. Since we created a 1-bit full adder named as “Block2”, we were able to come up with a design that would take use of the 1-bit full adder to create a 3-bit full adder.

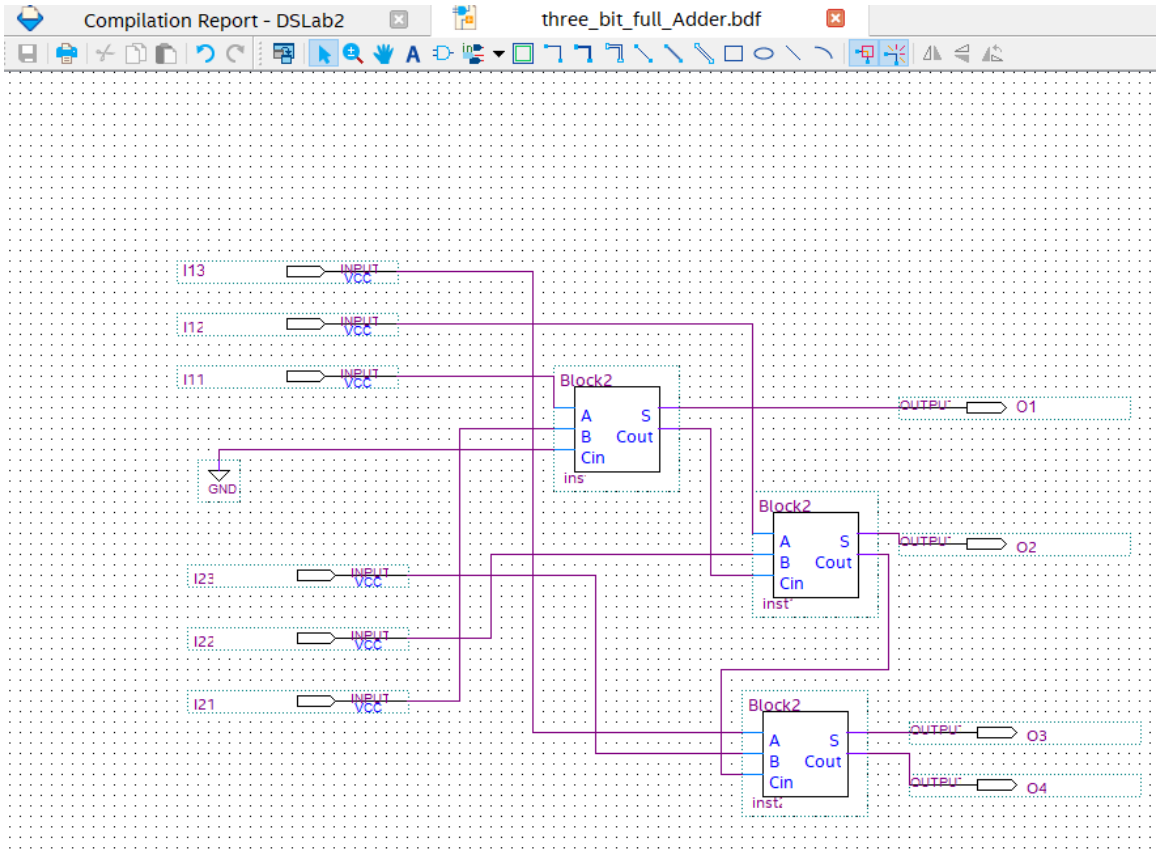


Figure 12-Block Diagram of 3-bit full adder

Figure 12 shows 3 “block2” which are symbol files of our 1-bit full adder. I_{11}, I_{12}, and I_{13} represent the first number to be added from the least significant to the most significant bit and I_{21}, I_{22}, and I_{23} represent the second number to be added. Our approach was to add each bits of same significance together. In other words, the 1-bit full adder would receive I_{11} and I_{21}. Since there is no carry at this point, we decided to ground C_{in}. The first output bit O_1would be the output S (sum) of the first full adder. The second full adder receives I_{12} and I_{22} with the carry of the first full adder as we would have done doing simple arithmetic. The second output bit O_2 would be the output S (sum) of the second full adder. Finally, the third full adder would have as inputs I_{13} and I_{23} with the carry of the previous full adder and give outputs O_3 which is S (sum) and O_4 which is the final carry out, thus giving us the answer to the addition of the two inputs. Note that this block diagram’s name is “three_bit_full_Adder”.

We will include a simulation in the simulation section to test our 3-bit full adder.

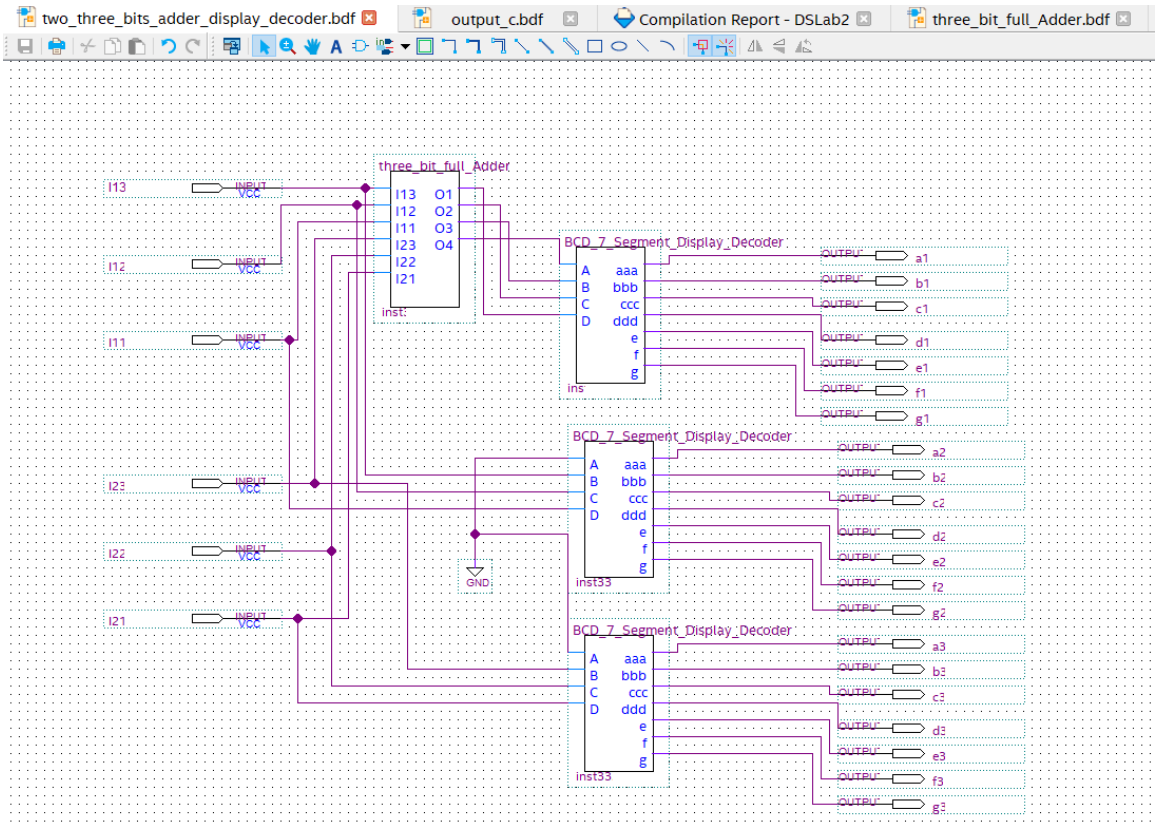


Figure 13- Block Diagram of 2-3 bits adder Display Decoder

Figure 13 shows the final block diagram named “two_three_bits_adder_display_decoder” which represents a 2-3 bits adder Display Decoder which takes as first input I_1 formed of bits I_{11}, I_{12} and I_{13} from least significant to most significant respectfully. And it takes a second input I_2 formed of bits I_{21}, I_{22} and I_{23} from least significant to most significant respectfully. These inputs are connected to the 3-bit full adder block symbol file we previously created. The outputs of the 3-bit full adder are connected to the inputs of the BCD 7-segments display decoder, so that we can converts our output from binary to hexadecimal to be able to visualize it on the board. Moreover, each input’s bits were also respectively connected to two BCD 7-segment display decoder so that we can visualize each input on its respective BCD 7-segment display decoder.

Simulation

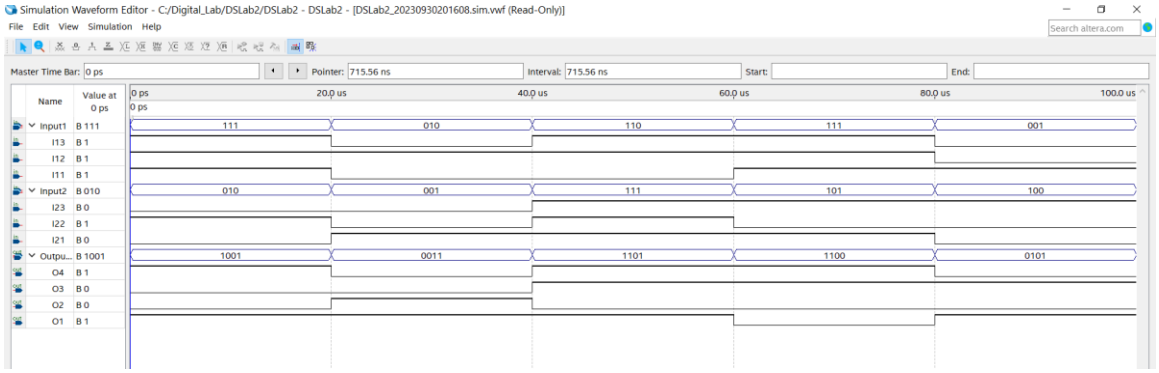


Figure 14- Simulation of the three bit full adder

Figure 14 shows "Waveform" which is the simulation waveform of the 3-bit full adder. As we can see in the first time interval, the first input is $I_{13}I_{12}I_{11}$ is equal to 111 (7) and the second input $I_{23}I_{22}I_{21}$ is equal to 010 (2) which add up to 1001 (9) as shown by $O_4O_3O_2O_1$ proving the correctness of our design.

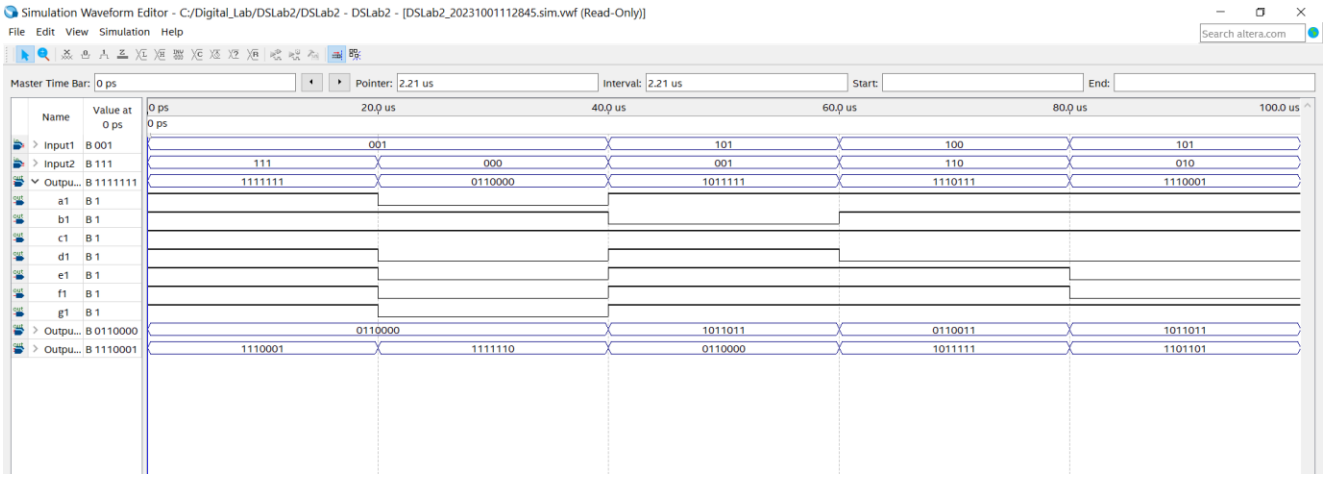


Figure 15- Simulation of the 2-3 bits full adder Display Decoder

Figure 15 shows "Waveform2" which is the simulation waveform of the 2-3 bits full adder Display Decoder. As we can see, in the first time interval, $Input_1$ ($I_{13}I_{12}I_{11}$) is equal to 111 (7) and $Input_2$ ($I_{23}I_{22}I_{21}$) is equal to 001 (1) which represents when summed up an 8 in decimal thus lightning up all of the Display Decoder's segments which will form an 8 so the result OutputSum(a_1 to g_1) 1111111 is correct.

The first interval displays an 8, the second a 1, the third a 6, the fourth a 10 (A), the fifth a 7. We will show in the remaining diagrams functional simulations that covers all the remaining cases (2-3-4-5-9-11(B)-12(C)-13(D)-14(E) as asked from us in the lab's questions.

Note that we cannot represent 15(F) since it is out of range when adding two-3 bits inputs because the maximum is 111 (7) + 111(7) giving a value of 1110(14, or E).

Note that there is no need to cover $Output_1$ and $Output_2$ since they use the same block symbol file for the BCD 7-segments display decoder. Having a correct output in OutputSum assures that $Output_1$ and $Output_2$ are correct.

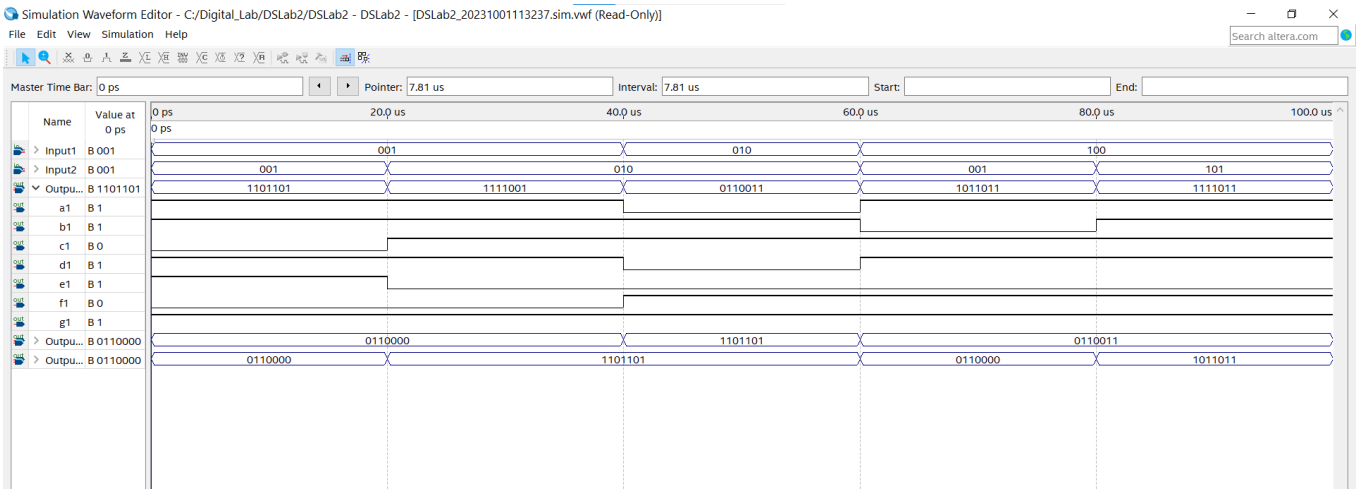


Figure 16- Simulation of OutputSum values 2-3-4-5-9

This figure is the same waveform as the one before but for the remaining values 2-3-4-5-9 respectively shown in each successive interval in the first output "OutputSum". For example, from 0 µs to 20 µs, we have $I_{13}I_{12}I_{11}$ equal to 001(1) added to $I_{23}I_{22}I_{21}$ equal to (001)(1) which lights up a_1,b_1,d_1,e_1,g_1 forming the shape of a 2.

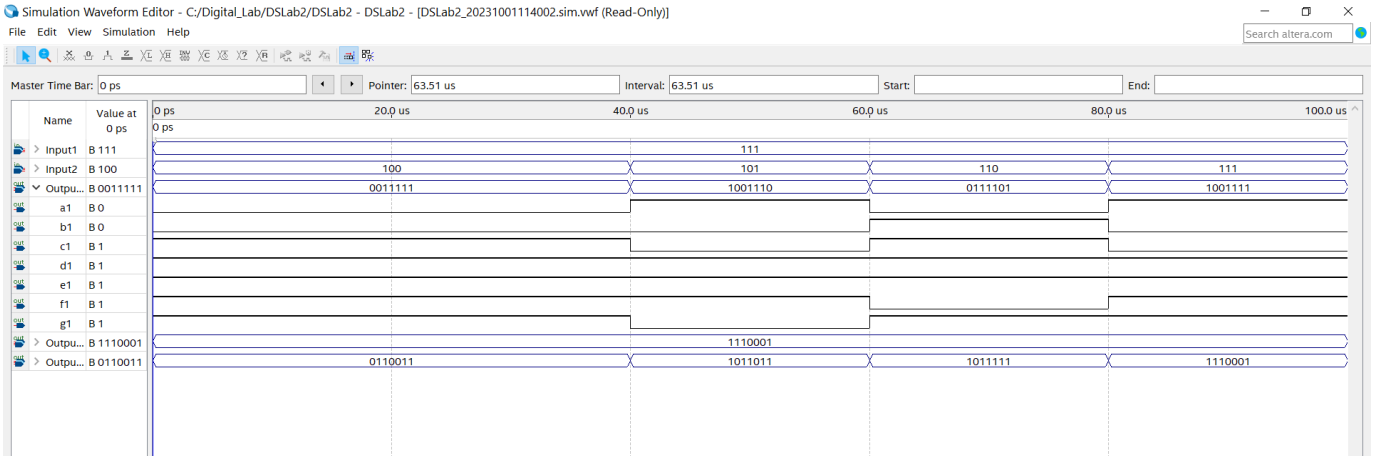


Figure 17: Simulation OutputSum values of b-C-d-E

This figure is the same waveform as the two previous ones before but for the remaining values b-C-d-E respectively shown in each successive interval in the first output "OutputSum". For example, from 0 µs to 40 µs, we have $I_{13}I_{12}I_{11}$ equal to 111(7) added to $I_{23}I_{22}I_{21}$ (100)(4) which lights up c_1, d_1,e_1,f_1 and g1 forming the shape of a b which represents an 11 but in hexadecimal.

Hence, going over all the possible values and getting accurate results proves that our 2-3bits adder Display Decoder is fully operation with 100% accuracy.

Hardware Simulation

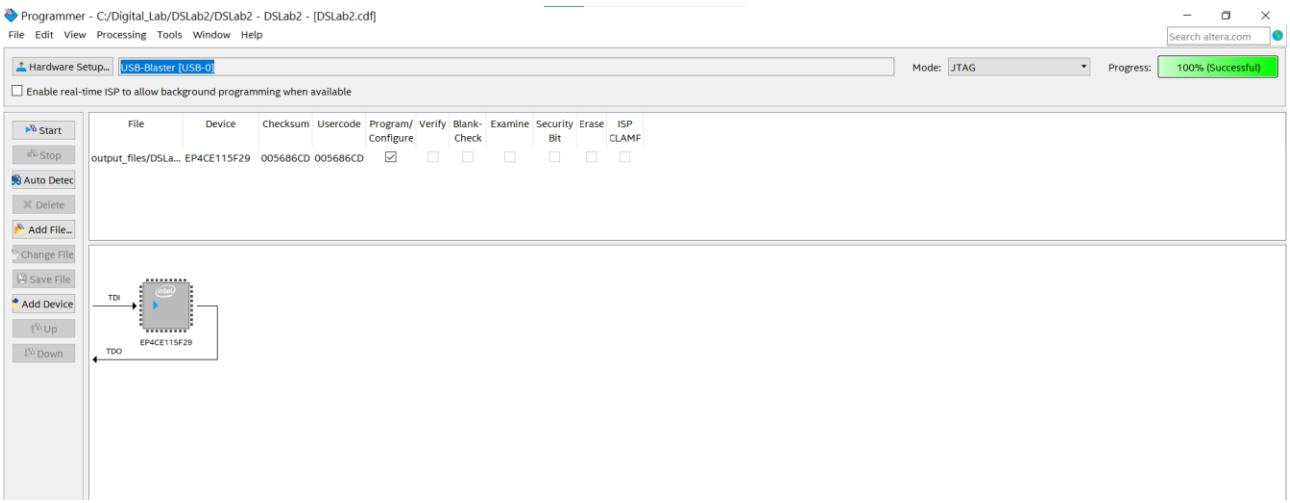


Figure 18- Programmer Interface

Figure 18 shows the programmer interface where we connect our FPGA to Quartus.

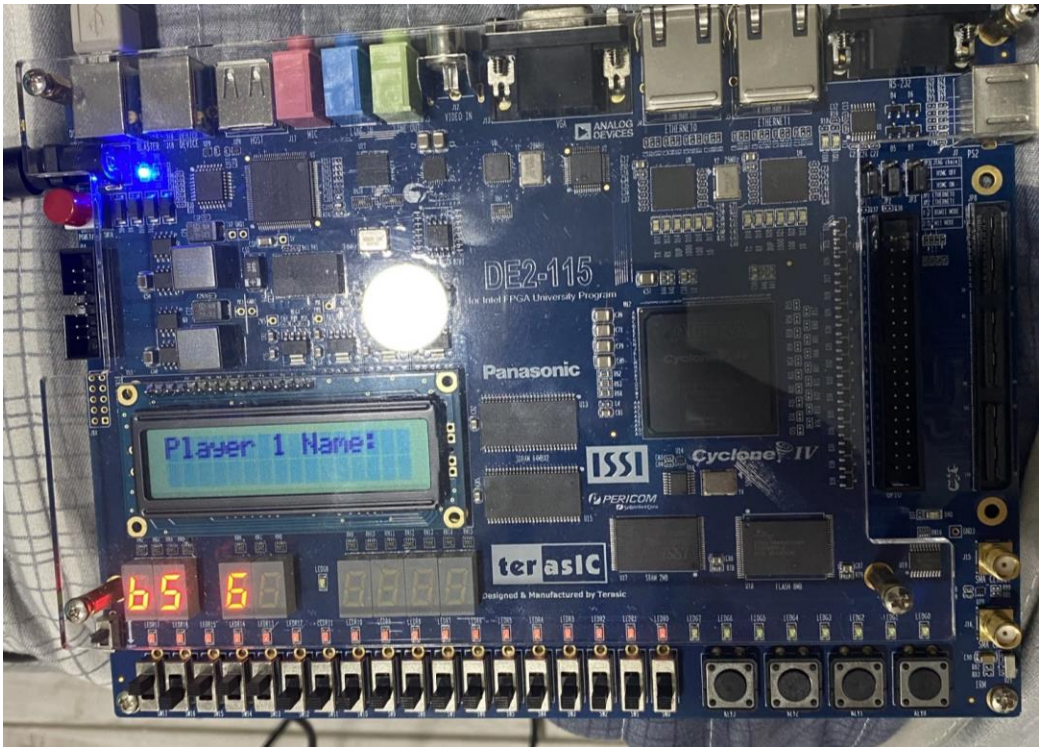


Figure 19- Hardware Simulation on FPGA

Figure 19 shows that 2 inputs were added, 5 and 6 which yield b as a result.

Conclusion

To sum up our work, we designed in the first part a BCD-7- segment display decoder by filling up a truth table, obtaining simplified equations from K-maps, implementing them on Quartus II and testing the results through waveforms. In the second part, we finally implemented a 2-3 bits adder display decoder by using a 3-bit full adder and 3 BCD 7-segment display decoder and implementing them on the FPGA to observe our result.

References

https://www.charlie-coleman.com/experiments/kmap/

https://ictlab.kz/extra/Kmap/