Course Name: Digital Logic Design Laboratory

Course Number: **14:332:233**

Lab #: 1

Section #: 2

Date Performed: February 12, 2021

Date Submitted: February 25, 2021

Submitted by: Qadis Chaudhry 202001055

Declaration of Authorship

I declare that this lab report is my own and has been generated by me as the result of my own work.

I confirm that I have acknowledged all main sources of help and, where I have consulted with others, this is always clearly attributed, including consultations with my TA or classmates.

I have provided the sources or references of all materials included in this report that were taken from other works, including online materials, such as pictures, graphs, diagrams, screenshots, and text quotes.

Moreover, I affirm that I have not let my classmates or other persons copy materials from this report for their own use.

Sign & Date: Qadis Chaudhry February 25, 2021

Qadis Chaudhry
Minning Zhu
Digital Logic Design Lab
February 26, 2021

Lab 1

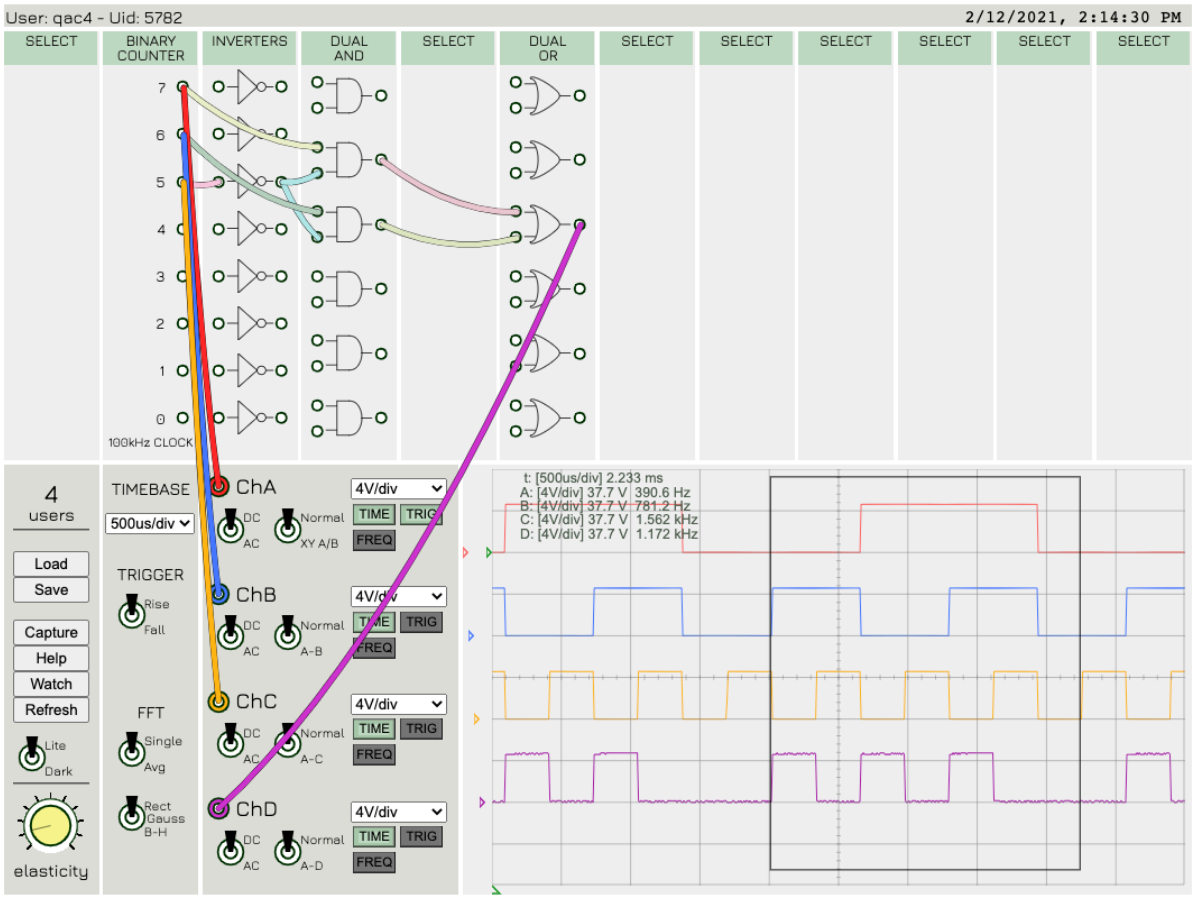
Experiment 1:

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

A	B	C	F(A, B, C)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

The truth table for a function shows the output of the function for all possible combinations of the input variables, $A, B,$ and C. This gives a predictable outcome for the implementation of the function within an integrated circuit. Since the terms of the function are being ORed together, only one of the two terms, $A\overline{C}$ or $B\overline{C}$, has to equal 1 in order for the function to output 1. Since both of the terms contain the inverted variable C, wherever C equals 1, the output of the function will have to equal 0 since both of the terms will equal 0 due to them being ANDed with the value of $\overline{C}$. For the rest of the table, we can look for when either A or B equals 1. This will make either of the terms $A\overline{C}$ or $B\overline{C}$ equal to 1 meaning, the function will be equal to 1. This function actually boils down to being dependent almost only on the value of C, since that variable is present within each of the other terms of the function. For seven of the eight combinations that are possible, the output of the function is the inverse of the variable C. The last case is where all of the variables, A, B, and C are 0. In this case, $\overline{C}$ will be equal to 1 however, A and B are both 0, making both of the terms of the function equal to 0, resulting in 0 as the output of the function.

To implement this function using gates, three types of gates are required, an “AND” gate, an “OR” gate, and an inverter or a “NOT” gate. The output of an AND gate is 1 only if both of the inputs are 1 while the OR gate’s output is 1 if either of the inputs is 1. An inverter does the simple task of inverting whatever input that is fed into it meaning, a 1 will turn into a 0 and a 0 will correspondingly become a 1. Using these three gates, any Boolean function can be implemented since all functions can be written as either a sum of product terms, a collection of terms that are ANDed together and are then all inputted into an OR gate, or the product of sums, the variables are ORed together and then all the terms are then the input of an AND gate. For this particular function, it can be seen that we would require two AND gates that take two inputs, one inverter to invert the value of the variable C, and one OR gate that takes in two inputs.



On the Emona board, the circuit is shown where the inputs are binary counters shown on the left. The value of 7, 6, and 5 correspond to the variables A, B, and C respectively. The value of C can be seen as being inputted into the inverter in the second column and in the third reside that AND gates. Two AND gates are used where one takes in the input variable A directly and

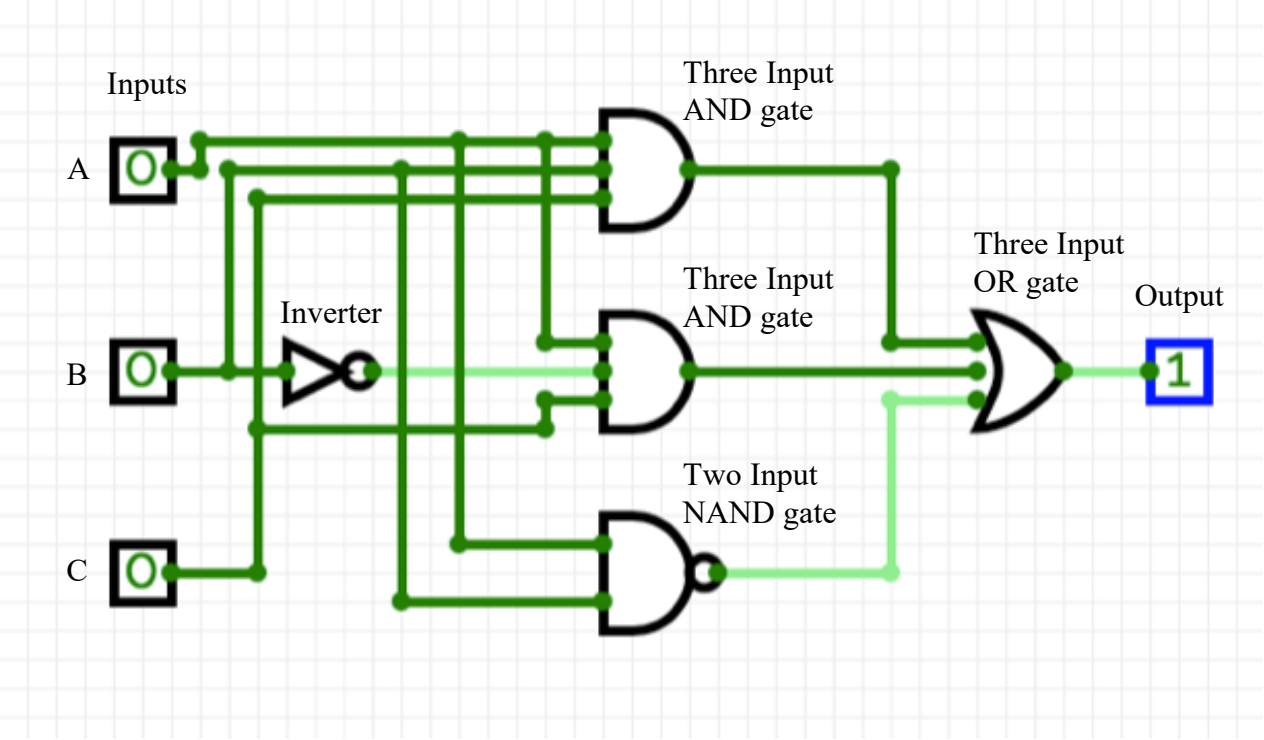
the output of the inverter that took the input variable C. The second AND module takes the input variable B directly and takes the inverted input of the variable C as well. It can then be seen that the two outputs of the AND gates are then fed into a dual input OR module. Lastly, all of the significant readings are fed into a corresponding channel where the input or output is monitored. Channel A is tuned to the value of the input variable A, Channel B is tuned to the variable B, similarly Channel C is tuned to the variable C. Channel D is then given the output of the whole function, the output of the OR module at the right end.

Experiment 2:

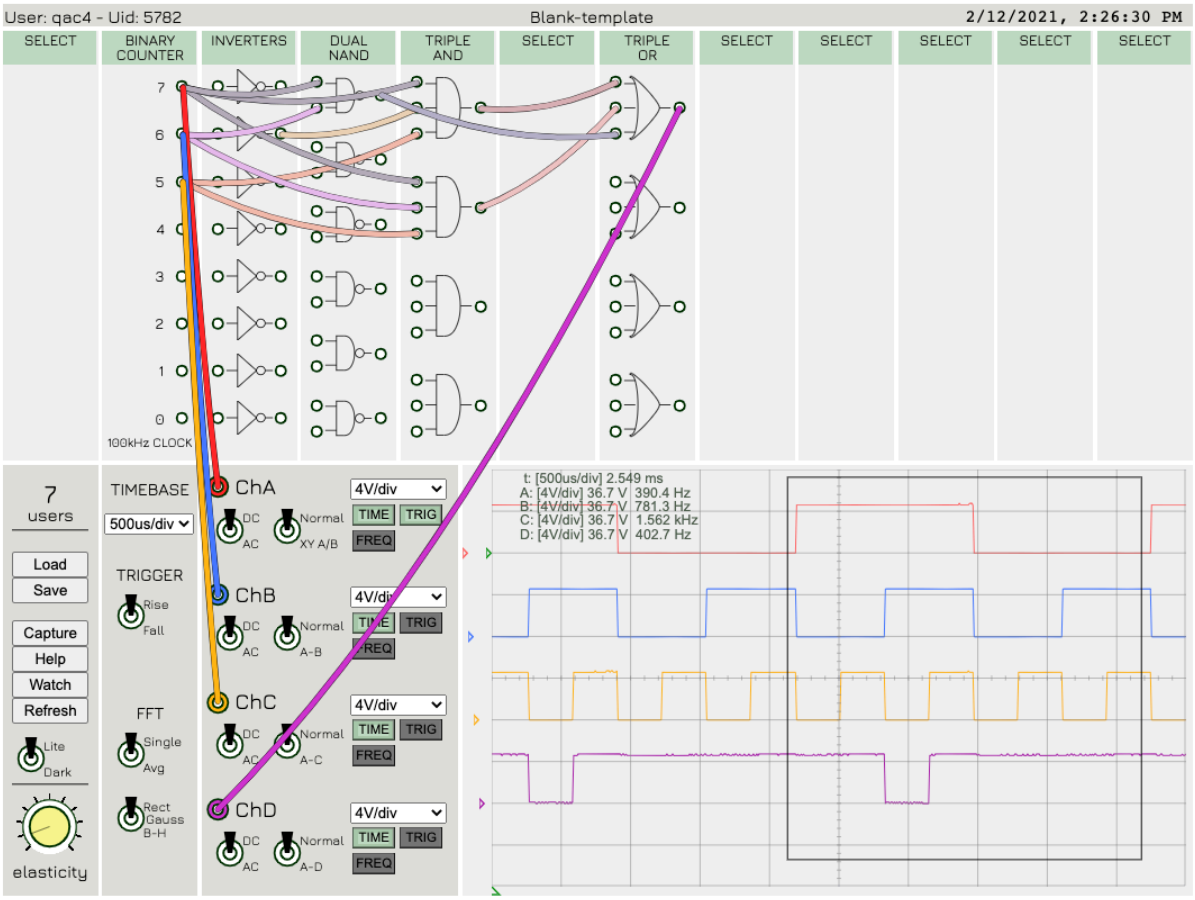
$$F(A, B, C) = A\overline{B}C + ABC + \overline{AB}$$

A	B	C	F(A, B, C)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

The truth table is built with the same reasoning as the in the previous experiment. Here, as before, we can check each of the terms individually to see if they equal 1. This is because they are all being ORed together, if one of the terms is 1, the output of the function is 1. We can see the last term does not include the variable C. Coupled with this, since the value of the term AB is being inverted, we can check to see where the value of AB equals zero and therefore the final term in the function, $\overline{AB}$, will be 1. Since the operation between the variables in AB is the AND operation, this will equal zero when either A or B equals 0. This means that all the outputs of the function up to the last two rows will be 1. The last row will make the value of the term ABC equal to 1, and therefore the output of the function equals 1. The second last row with the input 110, will be false since each of the terms will evaluate to 0 with that combination of values for the variables. Since C is 0, the first two terms will equate to 0 and since A and B are both 1, the last term will be the complement of 1, 0.



The schematic for this function is made up of two AND gates, one NAND gate, one inverter, and one OR gate. The inputs for the variables $A, B,$ and C are defined on the left as either 1 or 0. These inputs can correspond to the individual combinations from the truth table. The AND gate at the top represents the term, ABC as it takes in the values of A, B, and C directly. The AND gate below that represents the term $A\overline{B}C$, as it takes in A and C and the inverted value of the variable B. the last gate is the NAND gate, and this takes in the input, A and B. The NAND gate is by definition the inversion of the output of an AND gate, which means it can be used to describe the term $\overline{AB}$. Lastly, there is an OR gate that takes the outputs of the previous three gates and yields the output to the function, $A\overline{B}C + ABC + \overline{AB}$.



Experiment 3:

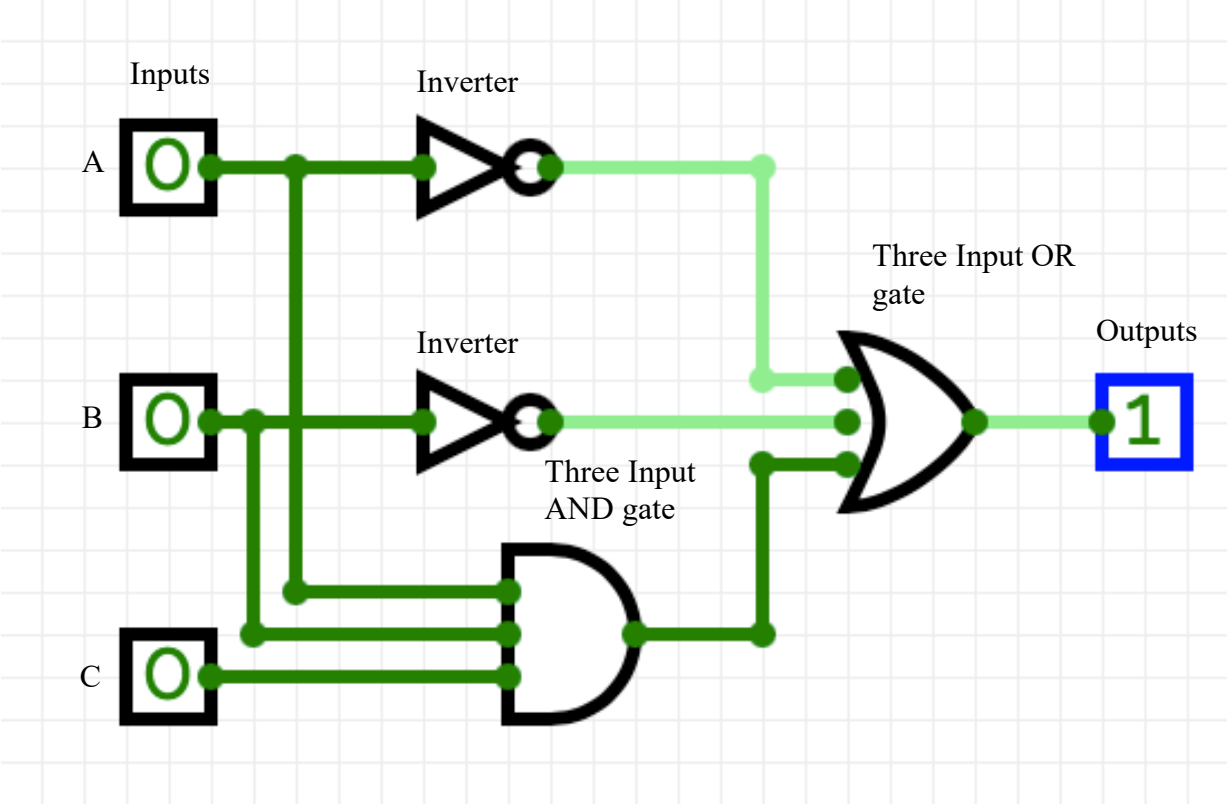
$$F(A,B,C) = \overline{A} + ABC + \overline{B}$$

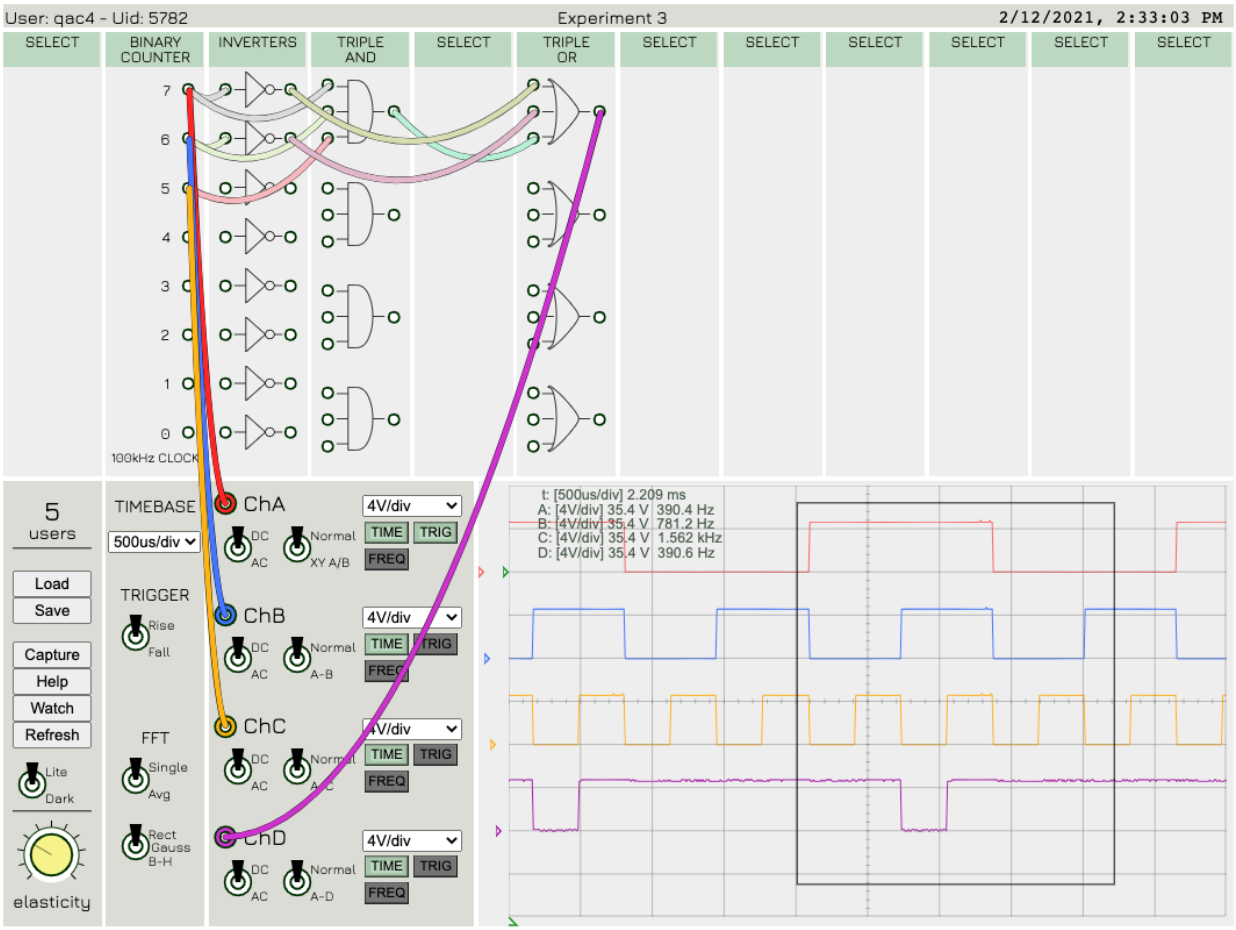
A	B	C	F(A, B, C)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

For this truth table, the same method of inspection can be used as with the others, check individually when each of the terms are 1 and since the terms are being ORed together, one of them being 1 will cause the function to output 1. In this case, since A and B are both inverted and summed in the function as alone terms, this function will be 1 when either A or B is equal to 0.

This means that all the rows up until the last two, the function will output 1. Since the term ABC is present, the last row will output 1 since all the variables are 1 making the term 1 which makes the function equal to 1. The second last row will be the only row that outputs 0 since the two variables A and B will be turned into 0, and the variable C is 0. This makes all the individual terms 0, resulting in the output of the function being 0. This yields the same truth table as the function from the previous experiment.

This circuit is similar to the last however, it is much simpler as it uses less gates. The truth table for this function was the same as the truth table for the function in the second experiment which means that these functions are basically the same function and this circuit is will yield the same outputs as the one before. The values for A and B are inverted and taken directly as inputs to the OR gate. The other term to be represented is the term ABC. This can be done simply with one AND gate whose inputs are each of the variables. The output of that AND gate is then taken as the last input to the OR gate.





Experiment 4:

The results for experiments 2 and 3 were very similar and it can be seen with the Emona circuit as well as with the analysis of the truth table for the two functions. It can be seen that the truth tables for each of the two functions yield the same patterns for the output with the only time the output being 0, is when the input combination for ABC is 110. The Emona board implementation of the circuits shows as well how the two are linked as it can be seen that the voltage readings for both of the functions are very close, essentially equivalent by observation.

This can be explained by the fact that the Boolean expressions can be simplified and rewritten with the theorems of Boolean and Switching Algebra. One of the functions may be able to be written as the other with the use of the rules. Another way this can be proven is by writing the expressions as a completed sum of minterms, meaning every term includes each of the input variables. By doing this, it will be seen that the expressions are equivalent as they possess the same combinations of the variables as the terms of the function.