ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E
DIGITAL CIRCUITS LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	1
EXPERIMENT DATE	:	17.04.2021
LAB SESSION	:	FRIDAY - 10.30
GROUP NO	:	G1

GROUP MEMBERS:

150170908	:	MUHAMMAD RIZA FAIRUZZABADI
070170364	:	AİŞE HÜMEYRA BOZ

SPRING 2021

Contents

1 INTRODUCTION [10 points]

In this project, we were required to do two distinct tasks, first one being preliminary to review our knowledge of Boolean algebra, and the second one is to design and implement logic circuits in Vivado Design Suite software. In preliminary part, we did calculations and simplifications by ourselves, and we used Logisim to draw the circuits. While in the second part, we constructed AND, OR, NOT gates modules then implemented and simulated the given logic functions.

2 MATERIALS AND METHODS [40 points]

2.1 MATERIALS

Tools Used

- Vivado Design Suite - Xilinx
- Latex (overleaf.com)
- Logisim

Firstly we designed our preliminary logic circuits in Logisim and then we design, implement and program the logic circuits to alongside their NOT,AND,OR gates modules in Vivado Design Suite, lastly we used overleaf.com to prepare the report document in LaTex.

2.2 PRELIMINARY

After revising theorems and axioms of Boolean Algebra, we proved given equalities ($A + AB = A$ and $(A + B)(A + B') = A$) below. Since they equal to the same expression ('a') we simplified them together side-by-side using axioms of Boolean Algebra as can be seen below.

$$A + A.B = (A + B).(A + B')$$

(Identity) $(A.1) + A.B = A + (B.B')$ **(Distributive)**

(Distributive) $A.(1 + B) = A + 0$ **(Inverse)**

(Identity) $A.(1 + B).1 = A$ **(Identity)**

(Inverse) $A.(1 + B)(B + B') = A$

(Distributive) $A.(B(1 + B')) = A$

(Identity) $A.(B + B') = A$

(Inverse) $A.1 = A$

(Identity) $A = A$

Next, we determined and proved the duals of the equalities defined above (first equation's dual in left side and second equation's dual in the right side) and explained the axioms we used which steps can be seen below.

$$A.(A + B) = A.B + A.B'$$

(Distributive) $A.A + A.B = A.(B + B')$ **(Distributive)**

(Idempotency) $A + A.B = A.1$ **(Inverse)**

(Absorption) $A = A$ **(Identity)**

In fourth part, we calculated the complementary expression of F $(A.B + A'.C)$ by utilizing De Morgan's theorem which is taking complement of the whole expression. The steps of simplification and the truth table of the expression can be seen below. Additionally, we drew the logic circuits of the expression and its complement using Logisim, as shown in Figure 1 and 2.

$$\begin{aligned}(F)' &= (A.B + A'.C)' \\ &= (A.B)'.(A'.C)' \\ &= (A' + B').(A + C')\end{aligned}$$

A	B	C	A'+B'	A+C'	F	F'
0	0	0	1	1	0	1
0	0	1	1	0	1	0
0	1	0	1	1	0	1
0	1	1	1	0	1	0
1	0	0	1	1	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	1	0	1	1	0

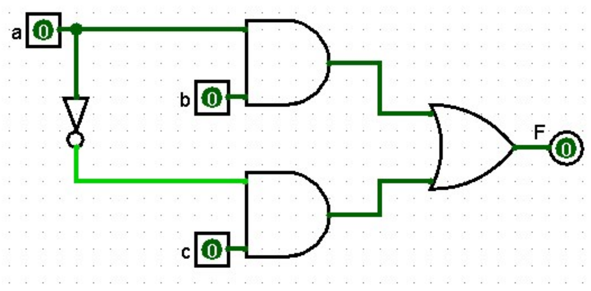


Figure 1: Expression F Logic Circuits Design

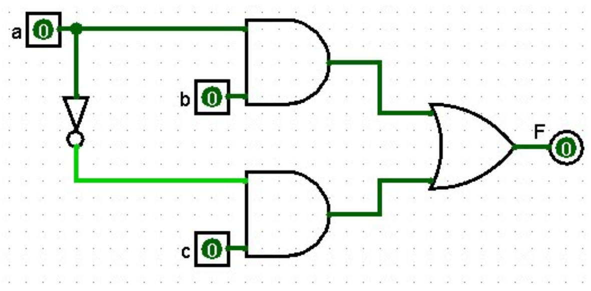
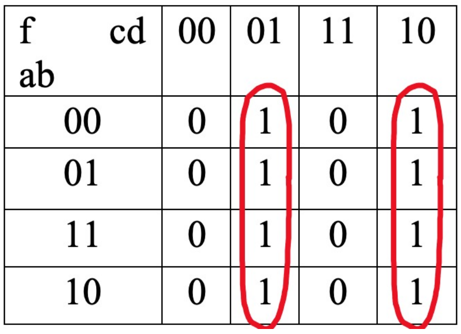


Figure 2: Expression F's Complement Logic Circuits

Lastly, we were given a logical function of $F(a, b, c, d)$= $U_1(1, 2, 5, 6, 9, 10, 13, 14)$ to simplify then draw its logic circuit. First, we used Karnaugh Map (Figure 3) to simplify the equation. By combining adjacent true points (1s) through the map, we obtained the prime implicants $F(a, b, c, d) = c'.d + c.d'$.



f cd ab	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

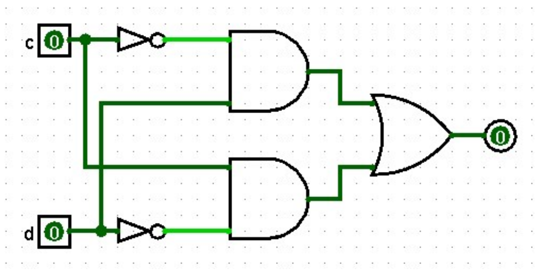


Figure 3: Karnaugh Map and Simplified Equation's Logic Circuits of Expression F

A	B	C	D	C'.D	C.D'	F
0	0	0	0	0	0	0
0	0	0	1	1	0	1
0	0	1	0	0	1	1
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	1	0	1
0	1	1	0	0	1	1
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	0	1	1	0	1
1	0	1	0	0	1	1
1	0	1	1	0	0	0
1	1	0	0	0	0	0
1	1	0	1	1	0	1
1	1	1	0	0	1	1
1	1	1	1	0	0	0

2.3 PART 1

In the first part of this experiment we were asked to implement AND, OR, NOT modules so that we can utilize them in the subsequent experiments, since we were not allowed to benefit from operators. For AND module, we chose A and B as inputs and C as an output. Then we assigned C as A&B using assign statement. For NOT module, A was the input and B was the output. We assigned B as $B = \sim A$. In the last module which is OR, we assigned C as the output and A and B as the inputs. Through assign statement, we obtained C = $A\,|B$. The elaborated design schematics of all three modules are shown in Figure 4 an 5, respectively.

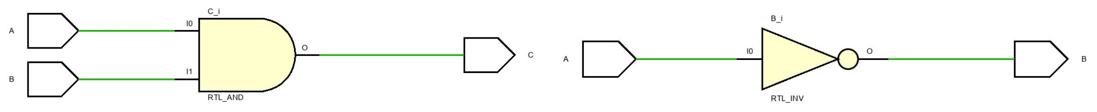


Figure 4: Elaborated Design Schematic of AND and NOT Module

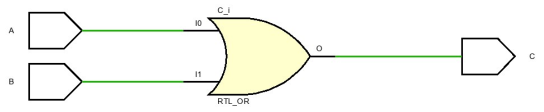


Figure 5: Elaborated Design Schematic of OR Module

2.4 PART 2

In the second part of the experiment, we were tasked to design and implement the logic circuits for given expressions of $A + AB = A$ and $(A + B)(A + B') = A$ by using AND, OR, NOT modules which we implemented in Part 1 and simulate it to validate its correctness. To implement the expression, we add new module in the previously created design source of module.v as F1 and F2. For the first expression of $A + AB = A$, we at first set A and B as Input and a as output. For the AND operations between A and B we instantiate the AND module and set A, B and AB as arguments. Then, we calculate A + AB by instantiating the OR module with A and previous AB output as arguments, alongside lowercase a, giving us the needed output for the expression. For the second expression of $(A + B)(A + B') = A$, we implemented similar techniques. We set A, B as input and lowercase a as output. We then instantiate the module OR with arguments (A, B, o1), module NOT with arguments (B, o2), module OR with arguments (A, o2, o3), and lastly module AND with arguments (o1, o3, a), respectively. The last output of a in the mentioned cascade multiple operations give us the result of expression F2. After implementing the modules for both of the expressions, we created two modules name Part1 and Part2 in the simulation source (simulation.v). In both modules, we set i1, i2 as reg Data Type and o as wire. Then we instantiate F1 and and F2 in each parts with arguments (i1, i2, o). Then we set the values of i1 and i2 to simulate with delay of 50 ps inside initial block. The result of the simulation can be seen in the Figure 6 and 7. Then we can check the correctness of the implementation by comparing the outputs with the truth table below.

A	B	A’	B’	F1	F2’
1	0	1	1	1	1
1	1	1	0	1	1
0	0	0	1	0	0
0	1	0	0	0	0

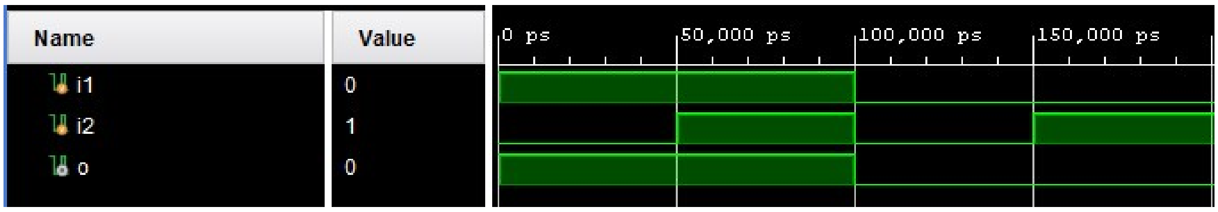


Figure 6: F1 Simulation with Delay 50

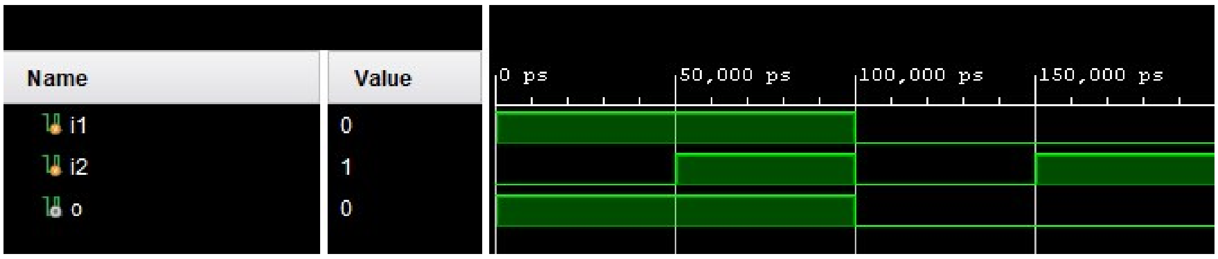


Figure 7: F2 Simulation with Delay 50

It can be seen that not only our implementations of both expressions are correct, both F1 and F2's outputs are equal, indicating its equality as we have proven in the previous preliminary part. Lastly, the elaborated design schematics of F1 and F2 are shown in Figure 8 and 9, respectively.

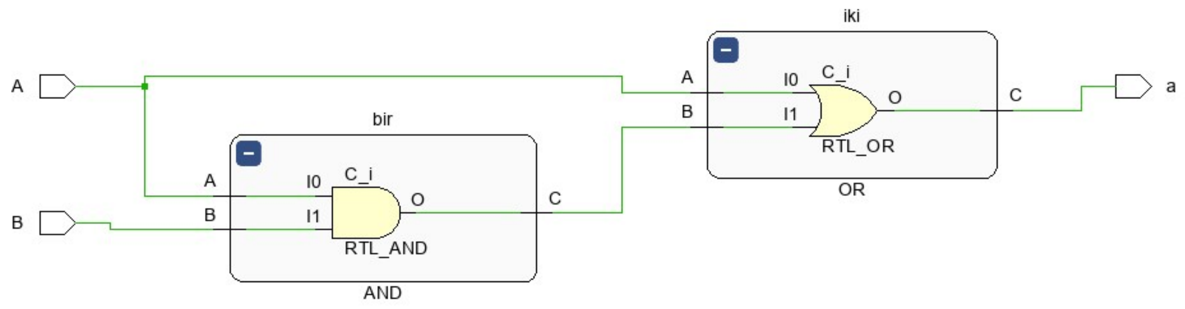


Figure 8: F1 Schematic

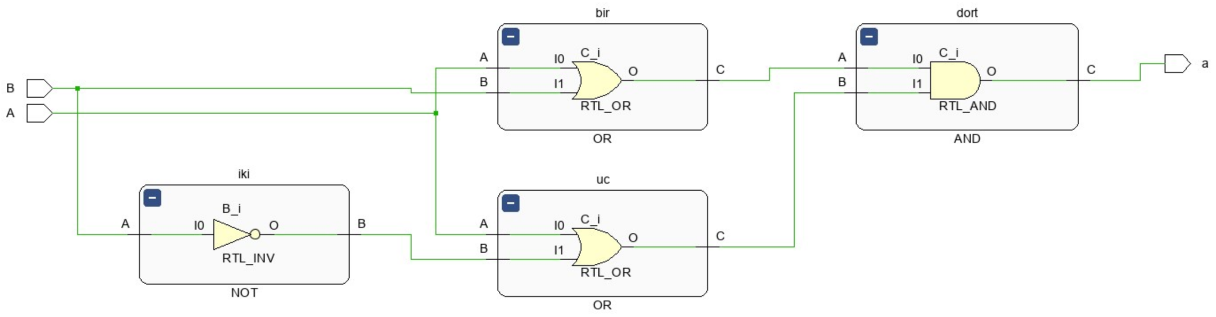


Figure 9: F2 Schematic

2.5 PART 3

In the third part of this experiment, we determined the dual of the given theorem $A + AB = A$ before the experiment, which can be indicated as $A.(A + B)$. We started by creating a design source module named as F3 . We used previously prepared modules of AND, OR, NOT. We assigned A, B as inputs and lowercase a as an output in the newly created module F3. Then we instantiate OR module with arguments A, B and as a result AB. We implemented AND operation between the the last operation's result AB and A input, got lowercase a. Afterwards, we validated the truth of the theorem by implementing the functions for the both sides of the dual theorem using AND, NOT, OR modules. Firstly, we created a module named Part3 in the simulation.v .We chose reg data types for i1,i2,i3,i4 inputs and wire data type for o1,o2 outputs. We instantiated F1 for the first equation and F3 for the dual. We set i1,i2,o1 arguments in F1 module; i3,i4,o2 arguments in F2 module. Then we set the values of i1,i2 with delay 200 ps inside initial block. We constructed another initial block in order to detect the duals by comparing the changes in the outputs using simulation. It can be seen from the truth table and simulation outcome (Figure 10) below. Lastly, the elaborated design schematic of F3 (Dual of F1) is shown in Figure 11.

A	B	A'	B'	F1	F3
1	0	1	1	1	1
1	1	1	0	1	1
0	0	0	1	0	0
0	1	0	0	0	0

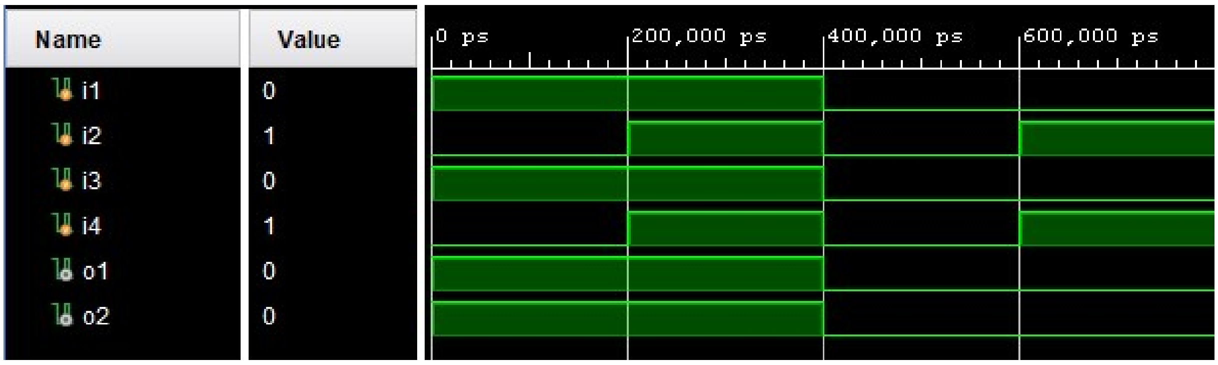


Figure 10: F3 (F1's Dual) Simulation with Delay 200 ps

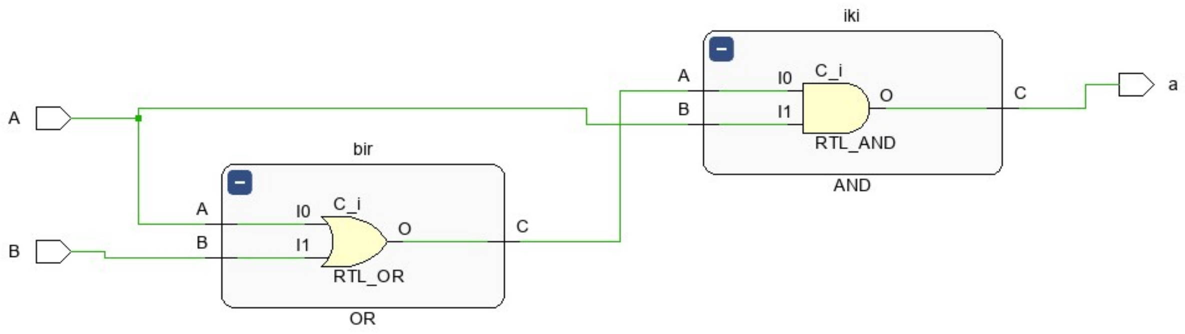


Figure 11: Elaborated Design Schematic of F3 (F1's Dual) Module

2.6 PART 4

In the fourth part of the experiment, we were given an equation $F_3(a, b, c) = A.B+A'.C$ and were asked to implement the circuit that realizes its complement. Using De Morgan's theorem, we have determined its complementary expression as $F = (A + B).(A + C)$. It is worth mentioning that since we have assigned 'F3' as the F1's dual in the previous part, we'll be using the name 'F4' for this expression. As in the previous expressions we have implemented, we implement this F4 expression as module inside Design Source modules.v. Within this F4 module, first we set A, B and C as input, and O as output. Then we again use sequential assigments to cascade multiple operations F4 consists. First we instantiate NOT Module three times for each A, B and C input to get its negates as nota, notb and notc. Then we instantiate OR module with arguments (nota, notb, o1), again OR module with arguments (A, notc, o2), then AND module with arguments o1 and o2 with output O. After we implemented the circuit, we created a simulation module inside simulation.v simulation source as Part4. Inside Part4, we first set i1, i2, and i3 as reg Data Types and o as wire. Then we instantiated F4 with those. To simulate, we entered values for i1, i2, and i3 needed to test inside initial block. The delay is set to 50. The correctness of the implementation can be checked by comparing the values of truth table of F4 below which we have also formed in the preliminary part with the simulation of our implementation shown in Figure 12.

A	B	C	A'+B'	A+C'	F	F'
0	0	0	1	1	0	1
0	0	1	1	0	1	0
0	1	0	1	1	0	1
0	1	1	1	0	1	0
1	0	0	1	1	0	1
1	0	1	1	1	0	1
1	1	0	0	1	1	0
1	1	1	0	1	1	0

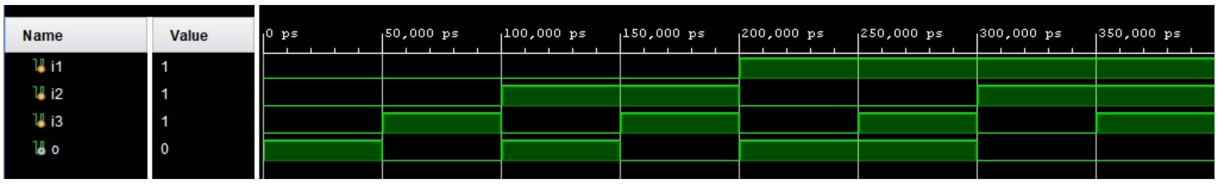


Figure 12: F4 Simulation with Delay 50 ps

It can be seen that values of F' in Truth Table and o in the F4 Simulation are equal for every inputs of ABC/I1I2I3. Lastly, the elaborated design schematics of F4 are shown in Figure 13.

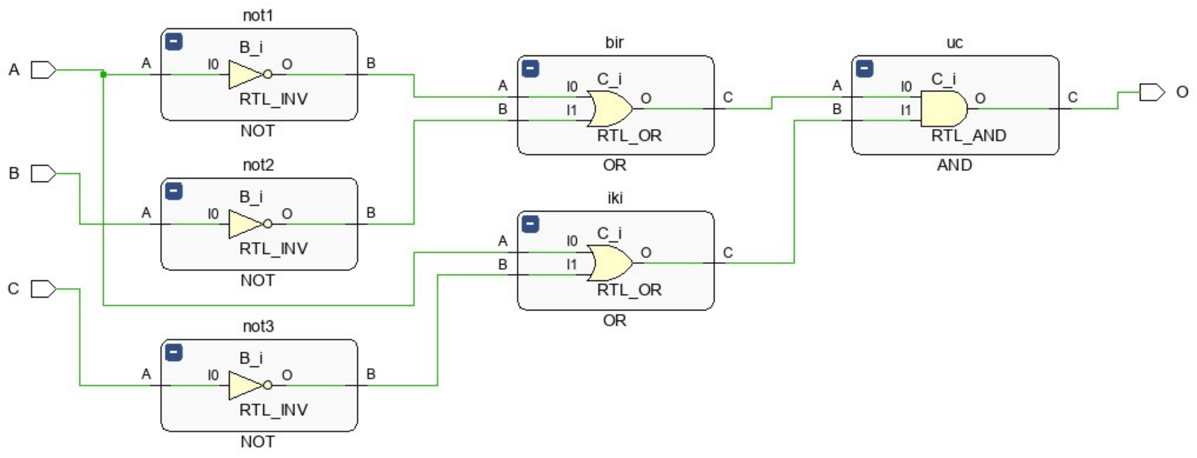


Figure 13: Elaborated Design Schematic of F' = F4 = (A+B).(A+C)

2.7 PART 5

In the fifth part, we obtained the function in numbered (indexed) representation. It was listed the decimal numbers of each input combination which the function generates

’1’. The values are 1,2,5,6,9,10,13,14 and U_1 denotes ’set of 1-generating points’. For simplification, we used Karnaugh Map and minimized the expression in the preliminary part of the report. The expression was minimized into $F(a,b,c,d) = c'.d + c.d'$. We formed a module named F5 having A,B,C,D inputs and O outputs. In order to visualize it, we instantiate two NOT modules for C and D inputs to obtain their complements namely notc, notd. Second, we instantiate two AND gates which first have C, notd inputs and O1 output, the second one consists of D, notc as inputs and O2 output. Finally, we instantiate OR module that has O1,O2,O as arguments. For simulation, we constructed the last module, simulation source Part5 which has reg data types i1,i2,i3,i4 and a wire data type o. We recalled F5 we implemented before. Since we only used C and D inputs in initial begin block we tested only them. The delay is set to 100. The truth table below is minimized version of the truth table regarding F5 in preliminary part. The correctness of the implementation can be detected with the values of truth table of F5 below which we have also created in the preliminary part with the simulation of our implementation shown in Figure 14.

A	B	C	D	C’	D’	C’.D	C.D’	F5
X	X	1	0	0	1	0	1	1
X	X	1	1	0	0	0	0	0
X	X	0	0	1	1	0	0	0
X	X	0	1	1	0	1	0	1

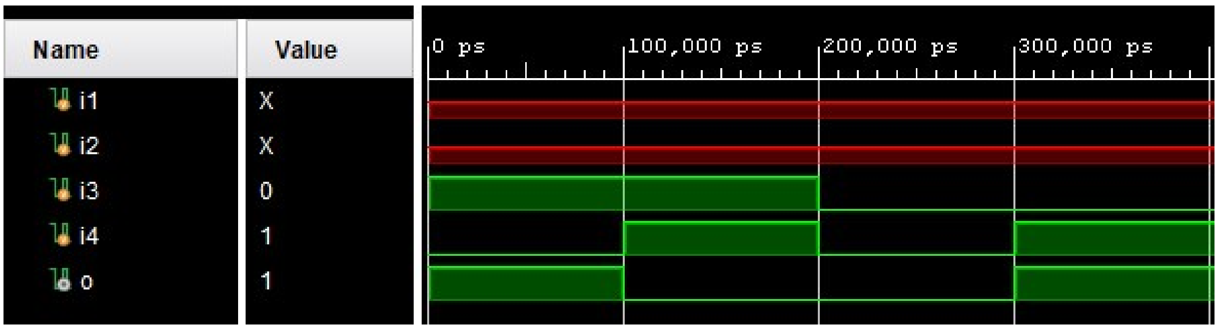


Figure 14: F5 Simulation with Delay 100 ps

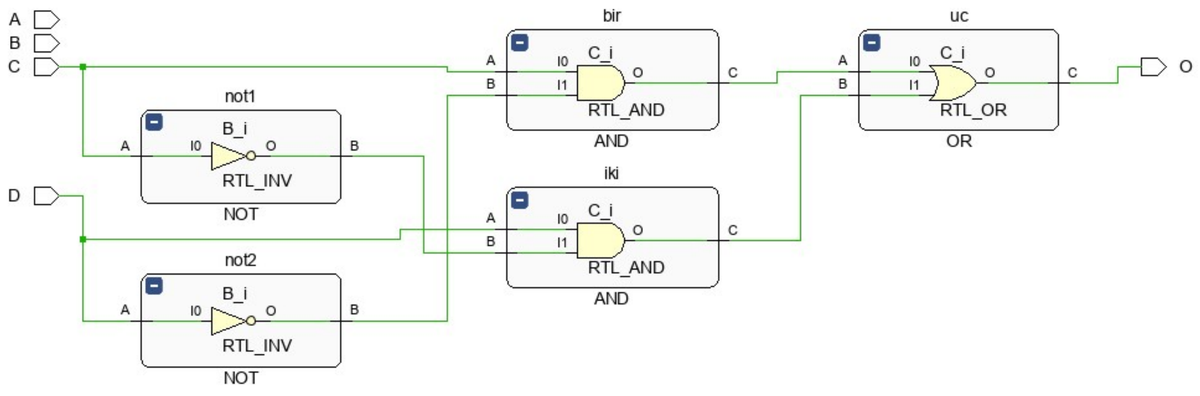


Figure 15: Elaborated Design Schematic of F5

3 RESULTS [15 points]

We managed to complete every tasks given in the experiment. At first we prove the expressions' equality and duality in the preliminary part by hand, but then we got to see them clearly through simulations. We also got the correctness of the simulation results by comparing them with truth tables we set beforehand. In this experiment we learned how to use Vivado Design Suite and to program in Verilog for the first time. Most importantly we learned the needed steps and tricks in using Vivado to make better use of it which we did not know much prior like how to set extend/narrow the simulation window, to set modules at top (or to comment modules we are not currently using), et cetera.

4 DISCUSSION [25 points]

Please explain, analyze, and interpret what have you done during the experiment. In the preliminary part, we prove the equality and duality of the expressions by hand, which then we wrote step by step in its part in the Methods section. We also have created the function tables/truth tables of every expressions again in their corresponding parts. Some of them are also usable to check the correctness of the implementation by comparing them with simulation results. In preliminary part we have shown the logic circuit design of expressions. We also used Karnaugh map solution in one part of the preliminary. In materials and methods section, we did indicate the elaborate design schematic of modules alongside with simulations.

5 CONCLUSION [10 points]

Since it was the first time we used Vivado Design Suite powered by Xilinx, we faced lots of barriers during our learning process. We did know how to implement basic AND, OR, NOT modules, but had no idea about designing modules for each equation then test them. We learnt all in time but it took a lot of effort. In addition, we discovered the main and detailed aspects of LaTex. We revised what we learnt in Digital Circuits course last term by utilizing Logisim. It was challenging to implement circuits with the specified modules rather than using pre-defined operators during the experiment, but it contributed to our learning process.

REFERENCES

[1] LogicLab. Logic lab. *An example journal*, 22(4):10–16, February 2020.

[2] Overleaf documentation https://tr.overleaf.com/learn.