ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E
DIGITAL CIRCUITS LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	3
EXPERIMENT DATE	:	28.02.2020
LAB SESSION	:	FRIDAY - 08.30
GROUP NO	:	G6

GROUP MEMBERS:

150170087	:	Sırrı Batuhan ÇOKSAK
150170039	:	Fatih MURAT
150180905	:	Fatima RAHİMOVA(G9)

SPRING 2020

Contents

1 INTRODUCTION

In this experiment, we were expected to find the prime implicants and the implementations of them with the lowest cost of the given function. We experienced well-known methods such as karnaugh map and Quine-McCluskey algorithm. Beside that, we converted given expressions each other with the given constrains. While doing these, we used AND, OR, NOT, NAND gates, multiplexers and decoders.

2 MATERIALS AND EXPERIMENT

Tools Used

- C.A.D.E.T
- 74000 series ICs
  - 74xx00 - Quadruple 2-input Positive NAND Gates
  - 74xx04 - Hex Inverters
  - 74xx08 - Quadruple 2-input Positive AND Gates
  - 74xx10 - Triple 3-input Positive NAND Gates
  - 74xx11 - Triple 3-input Positive AND Gates
  - 74xx32 - Quadruple 2-input Positive OR Gates
  - 74xx138 - 3:8 Decoder
  - 74xx151 - 8:1 Multiplexer

2.1 PART 1

In the first part of the experiment, we were given set of 1-generators and set of don't cares of the function F_1. From this information, we drew the karnaugh map easily and selected the all prime implicants. For all prime implicants we had cost values(2 for each variable and 1 for complement operation).

- F_1(a,b,c,d) = $U_1(0, 3, 5, 7, 11, 12, 13) + U_\Phi(1, 8, 15)$

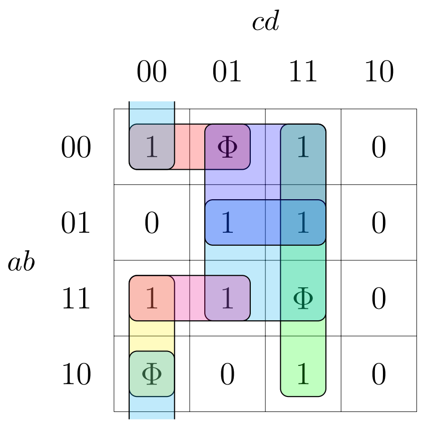

Figure 1: Karnaugh map of F

	$a'b'c'$	$b'c'd'$	$a'd$	cd	bd	abc'	$ac'd'$
SYMBOL	A	B	C	D	E	F	G
COVERED POINTS	0,1	0,8	1,3,5,7	3,7,11,15	5,7,13,15	12,13	8, 12
COST	9	9	5	4	4	7	8

Table 1: All prime implicants

In order to choose proper prime implicants with the lowest cost, we used prime implicant chart method. We did not include the don't care values to our prime implicant chart since we did not need them anymore. Since D is the distinguished point for 11, we had to choose it. After we chose D, we eliminated the D as a row and other covering columns(3,7,11).

	0	3	5	7	11	12	13	COST
A	x							9
B	x							9
C		x	x	x				5
D		x		x	x			4
E			x	x			x	4
F						x	x	7
G						x		8

Table 2: Prime implicant chart.

After the first elimination operation we had new chart(Table 3). In this chart, we had no distinguished points, therefore we compared them each other to pick proper prime implicant with lower cost. Since A and B are the exactly same prime implicant in terms of coverage and cost, we were able to pick one of them. For this experiment, we chose A. After that, we eliminate the A and B's row and related column(0).

	0	5	12	13	COST
A	x				9
B	x				9
C		x			5
E		x		x	4
F			x	x	7
G			x		8

Table 3: Prime implicant chart after first elimination.

After the second elimination operation, we had another new chart(Table 4). In this step, we compared C and E. As it is obvious that E covers C with lower cost, we chose E. After we chose E, we eliminated related rows(C,E) and columns(5,13).

	5	12	13	COST
C	x			5
E	x		x	4
F		x	x	7
G		x		8

Table 4: Prime implicant chart after second elimination.

After the last elimination operation, we were left with F and G for the same point(12). Since F has lower cost than G, we chose F as our last prime implicant. After the elimination, nothing left and we are done with elimination part.

	12	COST
F	x	7
G	x	8

Table 5: Prime implicant chart after third elimination.

As a result of these eliminations, we chose A, D, E and F with 24 total cost. Chosen prime implicants is shown on karnaugh map in Figure 2.

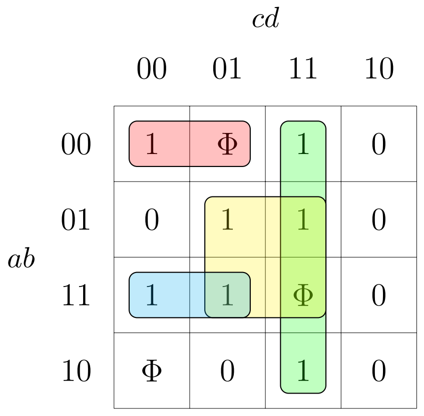

Figure 2: Karnaugh map of F

After the karnaugh map elimination operations, we were supposed to use Quine-McCluskey method. Algorithm steps are shown one by one below. Results from Quine-McCluskey method and karnaugh map verified each other and gave exact same prime implicants as it was expected.

Number	a b c d	
0	0 0 0 0	✓
1	0 0 0 1	✓
8	1 0 0 0	✓
3	0 0 1 1	✓
5	0 1 0 1	✓
12	1 1 0 0	✓
7	0 1 1 1	✓
11	1 0 1 1	✓
13	1 1 0 1	✓
15	1 1 1 1	✓

Table 6: Quine-McCluskey algorithm step-1

Number	a b c d	
0,1	0 0 0 -	Prime I.(a’b’c’)
0,8	- 0 0 1	Prime I.(b’c’d)
1,3	0 0 - 1	✓
1,5	0 - 0 1	✓
8,12	1 - 0 0	Prime I.(ac’d’)
3,7	0 - 1 1	✓
3,11	- 0 1 1	✓
5,7	0 1 - 1	✓
12,13	1 1 0 -	Prime I.(abc’)
7,15	- 1 1 1	✓
11,15	1 - 1 1	✓
13,15	1 1 - 1	✓

Table 7: Quine-McCluskey algorithm step-2

As it is been seen in the Table 8, there exists same expressions more than one. However, we use only one of the same ones.

Number	a b c d	
1,3,5,7	0 - - 1	Prime I.(a’d)
1,5,3,7	0 - - 1	Prime I.(a’d)
3,7,11,15	- - 1 1	Prime I.(cd)
3,11,7,15	- - 1 1	Prime I.(cd)
5,7,13,15	- 1 - 1	Prime I.(bd)

Table 8: Quine-McCluskey algorithm step-3

As a result of all of these operations, we determined our F function and its truth table as shown below;

- F(a,b,c,d) = a'b'c' + cd + bd + abc'

a	b	c	d	a'	b'	c'	a'b'c'	cd	bd	abc'	F(a,b,c,d)
0	0	0	0	1	1	1	1	0	0	0	1
0	0	0	1	1	1	1	1	0	0	0	Φ
0	0	1	0	1	1	0	0	0	0	0	0
0	0	1	1	1	1	0	0	1	0	0	1
0	1	0	0	1	0	1	0	0	0	0	0
0	1	0	1	1	0	1	0	0	1	0	1
0	1	1	0	1	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	1	1	0	1
1	0	0	0	0	1	1	0	0	0	0	Φ
1	0	0	1	0	1	1	0	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0	0
1	0	1	1	0	1	0	0	1	0	0	1
1	1	0	0	0	0	1	0	0	0	1	1
1	1	0	1	0	0	1	0	0	1	1	1
1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	1	1	0	Φ

Table 9: Truth table for F(a,b,c,d)

We implemented our expression using AND, OR and NOT gates as shown below;

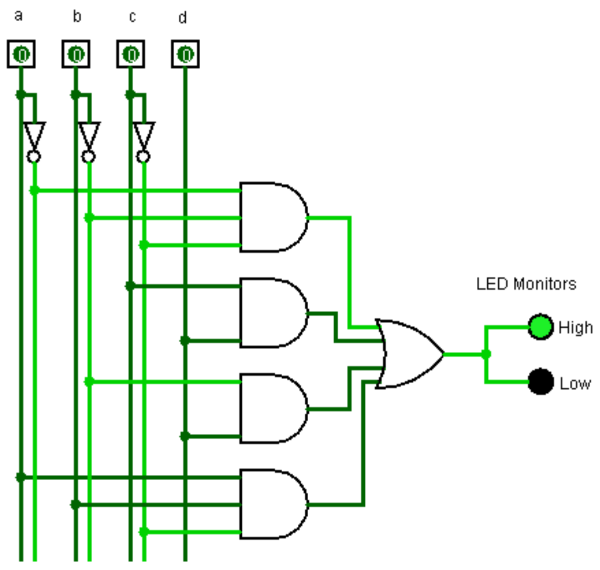

Figure 3: Circuit of $F(a, b, c, d)$

2.2 PART 2

In the second part of the experiment, we were expected to express our F function that we obtained from part-1 by using only NAND and NOT gates and implement the proper circuit.

- F(a,b,c,d) = a'b'c' + cd + bd + abc' (AND, OR, NOT gates are used)
- F(a,b,c,d) = (([(a'b'c')'(cd)'(bd)']')'(abc')')' (NAND and NOT gates are used)

Implementation of the F function using by only NAND and NOT gates is as shown in Figure 4;

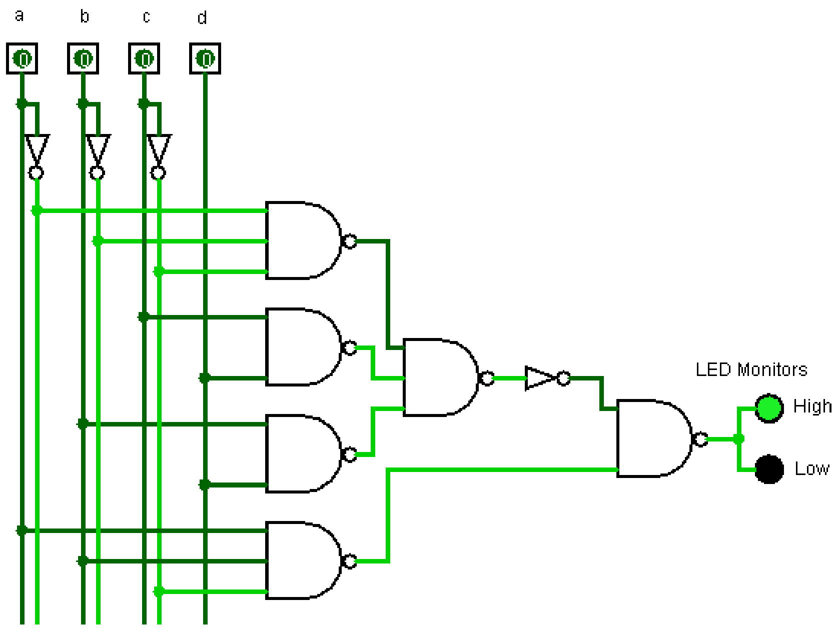

Figure 4: Circuit of $F(a, b, c, d)$

As these 2 expression is same in terms of outputs values, truth tables of them are also same.

2.3 PART 3

In the third part of the experiment, we were expected design and draw the same function using a single 8:1 multiplexer and NOT gates. In order to do that, we chose a,b and c input variables as selector input and d as input. Beside that, we assumed that enable input is always 1 and we did not show that in the logisim implementation. However, during the experiment we used ICs that requires enable input which we connected to high voltage. In this context, we obtained out truth table and the circuit implementation as shown below;

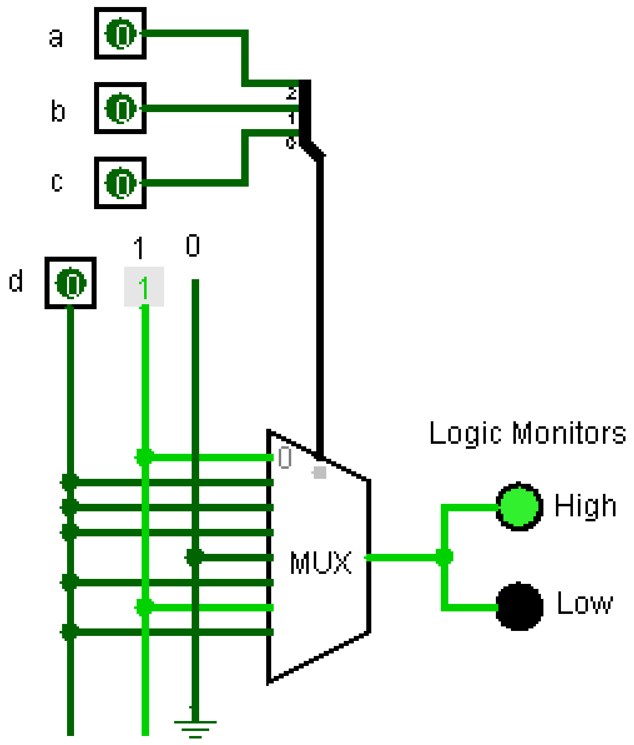

Figure 5: Circuit of $F(a, b, c, d)$

A	B	C	D	F	
0	0	0	0	1	1
0	0	0	1	Φ	
0	0	1	0	0	d
0	0	1	1	1	
0	1	0	0	0	d
0	1	0	1	1	
0	1	1	0	0	d
0	1	1	1	1	
1	0	0	0	Φ	0
1	0	0	1	0	
1	0	1	0	0	d
1	0	1	1	1	
1	1	0	0	1	1
1	1	0	1	1	
1	1	1	0	0	0
1	1	1	1	Φ	

Table 10: Truth table for multiplexer implementation.

2.4 PART 4

In the fourth and last part of the experiment, we were expected to design and draw the functions below using a single 3:8 decoder, OR, and NOT gates.

- F_1(a, b, c) = a'c' + bc
- F_2(a, b, c) = a'b'c' + ab

For the F_1 we drew the truth table and after that, we implemented our function as expected. But there was a important point in the lab session for this implementation; decoder was outputting the inverted form of the expressions. Therefore we needed to implement extra inverters to obtain normal forms(not inverted version) of the outputs. With this touch, we got the expected and correct outputs from the logic monitors.

a	b	c	a'	b'	c'	$a' \cdot c'$	$b \cdot c$	$F_1 = a' \cdot c' + b \cdot c$
0	0	0	1	1	1	1	0	1
0	0	1	1	1	0	0	0	0
0	1	0	1	0	1	1	0	1
0	1	1	1	0	0	0	1	1
1	0	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0	0
1	1	0	0	0	1	0	0	0
1	1	1	0	0	0	0	1	1

Table 11: Truth table for F_1

After we drew truth table for F_1, we implemented our circuit in this way;

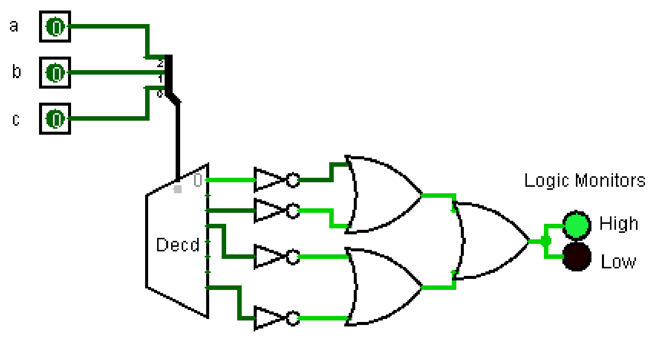

Figure 6: Circuit of $F_1(a, b, c, d)$

For the F_2 we did the same operations as we did for F_1. Simply, we drew the truth table and implemented our circuit as the truth table verified our implementation. Decoder was giving inverted version of the output values as well. Therefore we used inverters to control this situtation again. As we assumed that enable input is always 1 for the former implementations, we did it for the F_2 implementation as well. As a result we obtained both truth table and circuit design as below;

a	b	c	a'	b'	c'	$a' \cdot b' \cdot c'$	$a \cdot b$	$F_2 = a' \cdot b' \cdot c' + a \cdot b$
0	0	0	1	1	1	1	0	1
0	0	1	1	1	0	0	0	0
0	1	0	1	0	1	0	0	0
0	1	1	1	0	0	0	0	0
1	0	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0	0
1	1	0	0	0	1	0	1	1
1	1	1	0	0	0	0	1	1

Table 12: Truth table for F_2

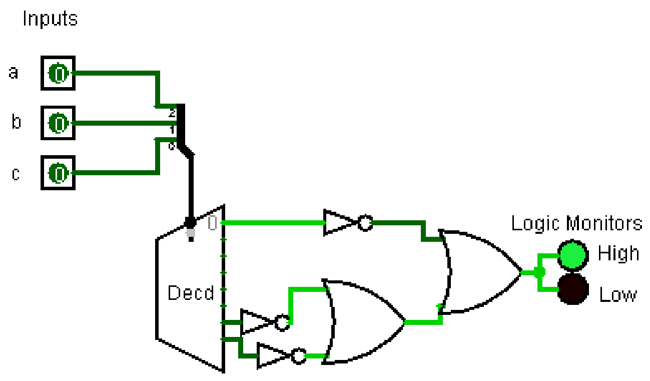

Figure 7: Circuit of $F_2(a, b, c, d)$

3 RESULTS

Throughout the whole experiment the results we obtained was not different from the expected results which we calculated before the implementation of the functions and expressions. Since the equipment we used during the experiment were mostly integrated circuits results were output values of these equipment and the results was certain either one or zero. Hence, we obtained precise results and verified our results using truth tables.

4 DISCUSSION

Generally, the results we found were the same with theoretically as they should have been. In the first part of the experiment, we couldn't get a result for a long time because we plugged the IC in reverse, which caused us to lose time. Other than that, we had some difficulties in the part 3 of the experiment . We could not figure out the pin configuration of 74151 - 8-Input Multiplexer but with the help of the laboratory assistant, we solved the problem and achieved the result. Except for this problem we did not face big problems and outputs of the each part of the experiment in logic monitor met the result we expected.

5 CONCLUSION

As a result, the experiment wasn't too hard but as we mentioned in the discussion part since we did not notice that we had the IC upside down for a long time, there was not enough time left to do F_2 in the fourth part of this experiment. But all in all, we have successfully completed the experiment without harming any IC, cabel etc. in the Logic Circuits Laboratory. Moreover, thanks to this session, we especially concreted our multiplexer and decoder knowledge and practical skills.

[1] Istanbul Technical University Department of Computer Engineering BLG-242E Logic Circuits Laboratory Experiments Booklet Version 1.9.1, spring 2020.
[2] Overleaf documentation https://tr.overleaf.com/learn.