# ISTANBUL TECHNICAL UNIVERSITY

# COMPUTER ENGINEERING DEPARTMENT

## BLG 242E

## DIGITAL CIRCUITS LABORATORY EXPERIMENT REPORT

| | | |
|---|---|---|
| **EXPERIMENT NO** | : | 2 |
| **EXPERIMENT DATE** | : | 24.03.2023 |
| **LAB SESSION** | : | FRIDAY - 16.00 |
| **GROUP NO** | : | G18 |

## GROUP MEMBERS:


150220770 : Onur Baylam

150200916 : Denıs Iurıe Davıdoglu


## SPRING 2023

# Contents

# 1 INTRODUCTION

In this experiment, we found prime implicants of the given function $F$, using both Karnaugh diagram and Quine-McCluskey method. Since Karnaugh map is a visual method to find prime implicants, it can be difficult to be sure if all prime implicants are detected. Therefore, we also found all prime implicants using Quine-McCluskey method to ensure that our findings are complete. Afterwards, in order to obtain the lowest cost expression, we drew prime implicant chart and compose our lowest cost expression. Finally, we implemented it using different ICs.

# 2 MATERIALS AND METHODS

## 2.1 Preliminary

1. Find all prime implicants of the function F below.
   $F(a,b,c,d) = \bigcup_1(1,4,6,11) + \bigcup_\phi(0,2,8,9,14,15)$

   a) using Karnaugh diagram

   We cluster all **1s** located in the adjacent cells in the Karnaugh diagram into same group to obtain prime implicants of $F$. Then, we need to keep the constant literals and discard the changing ones. To write each prime implicant, we use normal form of the variable if its truth value is **'1'**. On the other hand, if its truth value is **'0'**, we use its complementary form.

   **Reminder:** We assume **don't care terms** ($\phi$) to be **true** (1) to create larger groups.

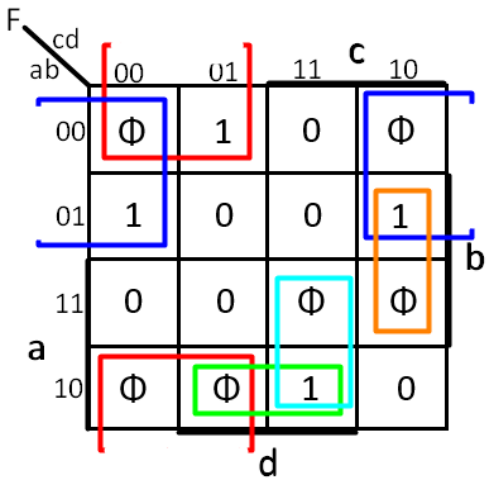



Figure 1: Karnaugh map of function $F$

**Set of all prime implicants:**


Figure 2: Prime implicants of the function $F$

b) using Quine-McCluskey method

**Step 1:** We should place all **1-generating** points and don't cares in a table. Also, we may cluster them based on the number of **1s** they contain to have a better running time.

**Step 2:** We compare all input combinations in the neighboring clusters to detect adjacent pairs of input combinations. When we find an adjacent pair of input combination, we create a new group of them and replace changing literals with "-". Finally, we mark both input combinations with "✓" to show that they are combined. Uncombined ones (without ✓) will be prime implicants.

**Reminder:** We apply step 2 until no further groups can be created.

| Step 1 | |
|---|---|
| Num | abcd |
| 0 | 0000 ✓ |
| 1 | 0001 ✓ |
| 4 | 0100 ✓ |
| 2 | 0010 ✓ |
| 8 | 1000 ✓ |
| 6 | 0110 ✓ |
| 9 | 1001 ✓ |
| 11 | 1011 ✓ |
| 14 | 1110 ✓ |
| 15 | 1111 ✓ |

| Step 2 | |
|---|---|
| Num | abcd |
| 0,1 | 000- ✓ |
| 0,2 | 00-0 ✓ |
| 0,4 | 0-00 ✓ |
| 0,8 | -000 ✓ |
| 1,9 | -001 ✓ |
| 2,6 | 0-10 ✓ |
| 4,6 | 01-0 ✓ |
| 8,9 | 100- ✓ |
| 6,14 | -110 |
| 9,11 | 10-1 |
| 11,15 | 1-11 |
| 14,15 | 111- (X) |

| Step 3 | |
|---|---|
| Num | abcd |
| 0,1,8,9 | -00- |
| 0,2,4,6 | 0–0 |

We have placed "X" beside the combination of $14, 15$ because 14 and 15 are both don't care points, so we do not need them.

So, unmarked rows are prime implicants:


Figure 3: Prime implicants of the function $F$

· Also create the prime implicant chart of the function F and find the expression with the lowest cost. Cost criteria is 2 units for each variable and 1 unit for each complement. Draw the lowest cost expression using AND, OR, and NOT gates.

Firstly, we label each prime implicant with a letter. Also, we write their costs and covered points below them.

| | b'c' | a'd' | bcd' | ab'd | acd |
|---|---|---|---|---|---|
| Symbols: | A | B | C | D | E |
| Costs: | 6 | 6 | 7 | 7 | 6 |
| Covered Points: | 1 | 4,6 | 6 | 11 | 11 |

Figure 4: Prime implicants with covered points

**Reminder:** Since it is not necessary to cover don't care terms ($\phi$), we do not include them to prime implicant chart.

**Prime implicant chart is given below:**

| PI | 1 | 4 | 6 | 11 | Cost |
|---|---|---|---|---|---|
| A | X | | | | 6 |
| B | | X | X | | 6 |
| C | | | X | | 7 |
| D | | | | X | 7 |
| E | | | | X | 6 |

Figure 5: Prime Implicant Chart

In the chart, **A** and **B** are essential prime implicants which cover distinguished points 1 and 4, respectively. So we have to include them to the lowest cost expression.

**We mark A and B because they are selected:**

| PI | 1 | 4 | 6 | 11 | Cost |
|---|---|---|---|---|---|
| A | X | | | | 6 |
| B | | X | X | | 6 |
| C | | | X | | 7 |
| D | | | | X | 7 |
| E | | | | X | 6 |

Figure 6: A and B are selected

**New form of the chart:**

| PI | 11 | Cost |
|---|---|---|
| D | X | 7 |
| E | X | 6 |

Figure 7: New Form of Prime Implicant Chart

To cover point 11, we need to select **E** which has a lower cost than D. So, we mark **E** since it is selected.

| PI | 11 | Cost |
|---|---|---|
| D | X | 7 |
| E | X | 6 |

Figure 8: E is selected

**Lowest Cost Expression (Minimal covering sum):**

F(a, b, c, d) = b'c' + a'd' + acd

Symbols: A + B + E

Total Cost: 6 + 6 + 6 = 18

Figure 9: Lowest Cost Expression

**Lowest cost expression using AND, OR, and NOT gates:**

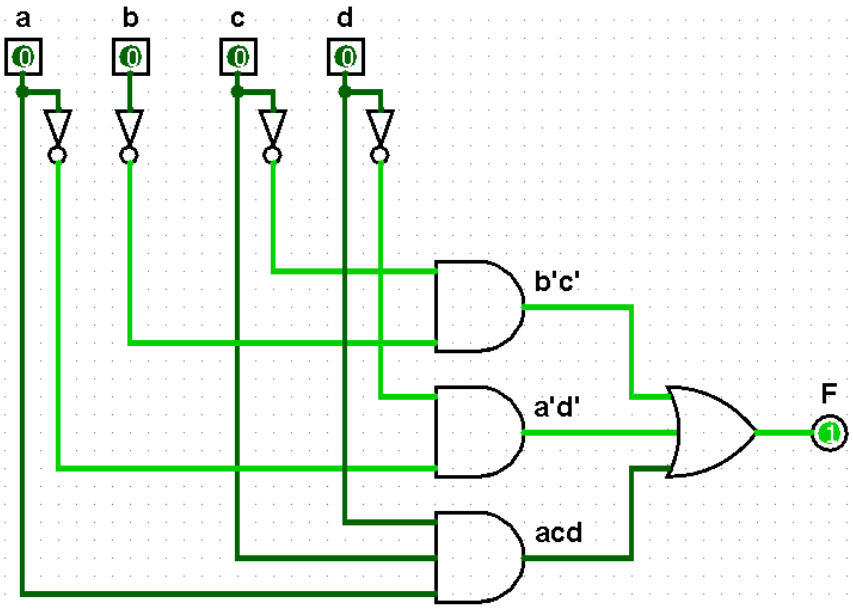
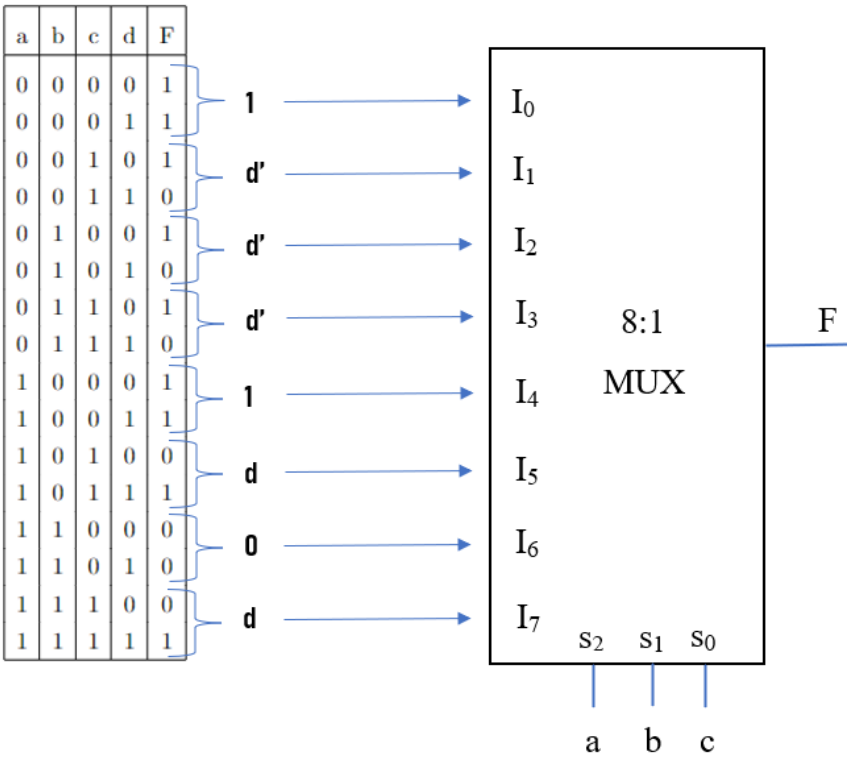



Figure 10: Drawing of Lowest Cost Expression

2. Design and draw the same function using a single 8:1 multiplexer and NOT gates.

   First of all, we create the truth table of function $F$, then using the truth table we will determine which input should be connected to the inputs of 8:1 Multiplexer.



| a | b | c | d | F |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |



Figure 11: Function $F$ using 8:1 multiplexer and NOT gate

**Design and drawing of $F$ using 8:1 Multiplexer and NOT gate:**

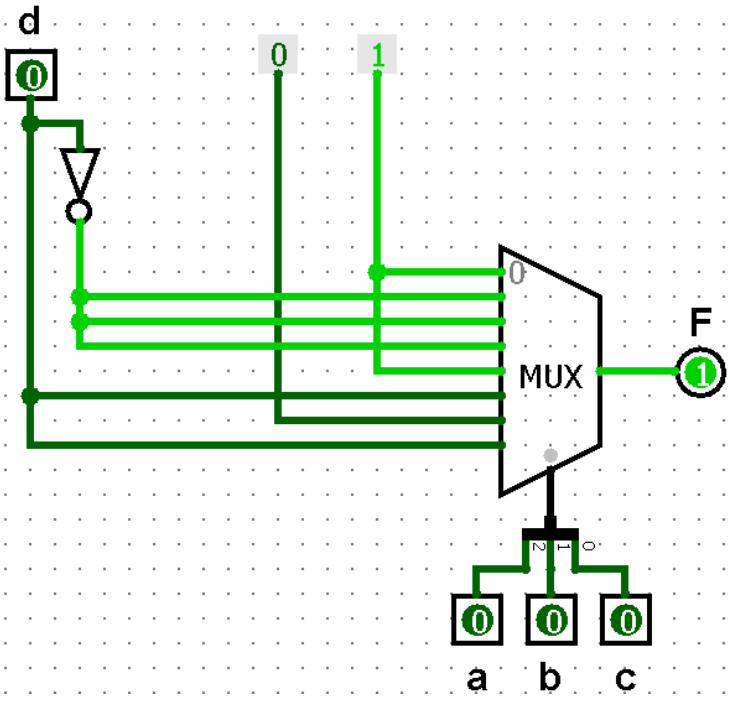



Figure 12: Function $F$ using 8:1 multiplexer and NOT gate

## 2.2 Experiment

- **Part 1:** Build the circuit you have designed in Preliminary Question 1 using the ICs. Fill up a truth table to evaluate your implementation.

**Function $F$ using ICs:**

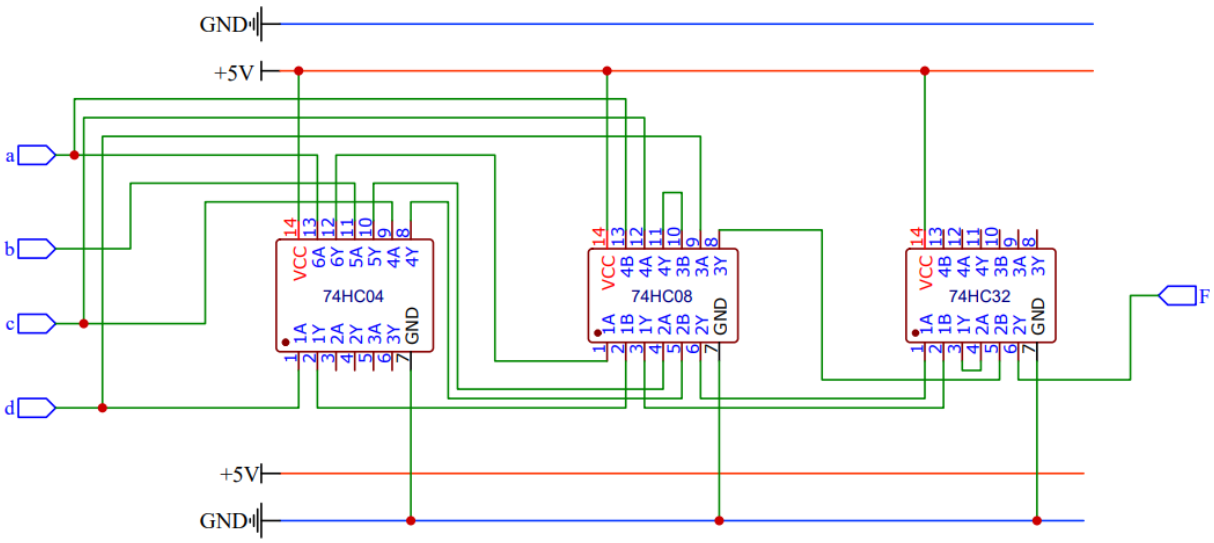



Figure 13: Function $F$ using ICs

**Truth Table of The Circuit:**

| a | b | c | d | F |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

In the experiment, we evaluated the outputs of our implementation utilizing the truth table and the results were correct.

- **Part 2:** Build the circuit you have designed in Preliminary Question 2 using the ICs. Fill up a truth table to evaluate your implementation.

**Function $F$ using ICs:**

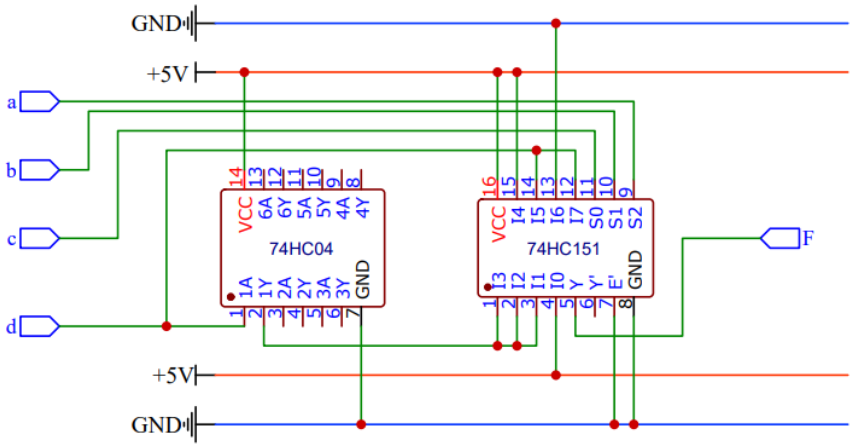



Figure 14: Function $F$ using ICs

**Truth Table of The Circuit:**

| a | b | c | d | F |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

The operation of this circuit is the same as the previous one. Also, we evaluated the outputs of our implementation utilizing the truth table in the experiment and the results were correct.

# 3 RESULTS

In the experiment, we intensified our knowledge on design of basic digital circuitry in a practical way by implementing the given function $F$. During implementation phases, we benefited from various ICs composed of logic gates such as AND, OR, NOT. Moreover, we had the chance to implement the given function using a multiplexer instead of usual ones. Afterwards, we validated our results of implementations and took notes from crucial points. Also, we took some pictures of our implementations. Here is the some images provided from our experience of experiment.

**An Image From Implementation of Part 1:**

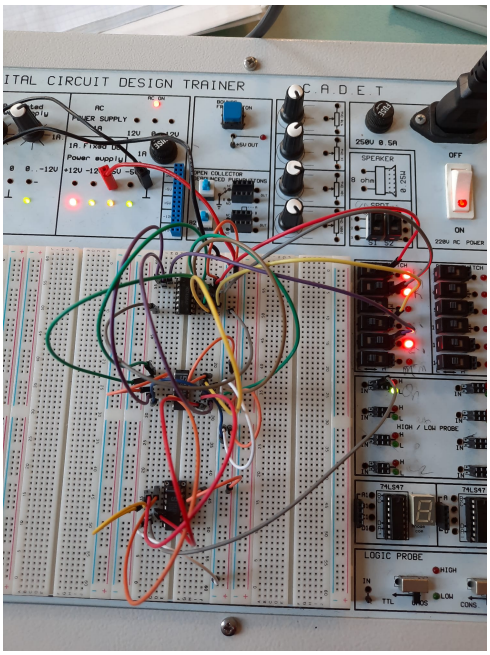



Figure 15: An image from part 1 implementation

**An Image From Implementation of Part 2:**

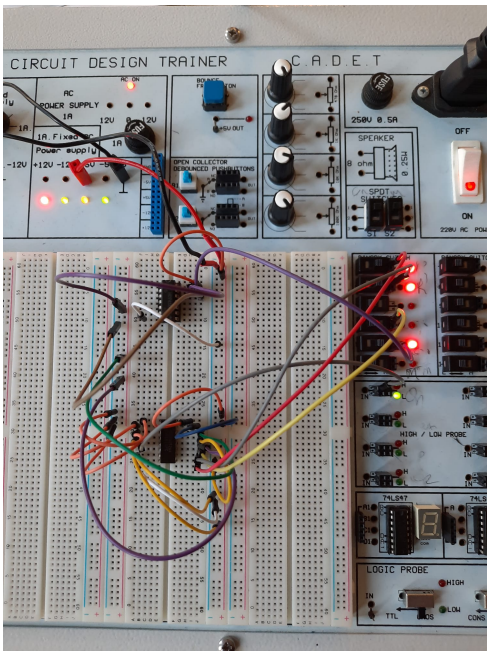



Figure 16: An image from part 2 implementation

# 4 DISCUSSION

In this experiment, we reviewed our learnings on finding the lowest cost expression. To find the lowest cost expression of given function $F$, we used two different procedures that first one is Karnaugh diagram which is a visual method, and second one is Quine-McCluskey method which has certain steps to investigate prime implicants. Each method has different dynamics, but to prevent possible mistakes, it can be better to use Quine-McCluskey method, since it is an algorithmic way of obtaining prime implicants. During the experiment, we implemented the lowest cost expression with ICs of AND, OR, NOT gates. When it comes to second part of the experiment, we made the same implementation using only multiplexer and NOT gates which provided us with a better intuition into digital circuitry. After our implementations, we checked our outputs with the help of truth tables to ensure that our circuits operate accurately.

# 5 CONCLUSION

In conclusion, implementations carried out during the experiment bring about the opportunity of acquiring internal knowledge to logical circuits thanks to realizing the same function using various kinds of ICs. So, this makes us even more comfortable to the components of digital circuitry. Because we discovered that different types of ICs can be used interchangeably. Furthermore, our second implementation was simpler than the first one. So, we learned that the same function can be implemented in a simpler way by just using different components like a multiplexer in our case. In addition, there are also some difficulties we faced during the experiment. First one is related to having a great number of wires which cause us to have difficulty in making connections accurately, but we overcame these difficulties easily by analyzing the circuits carefully and applying some intermediate tests to check if all connections are correct.