Laboratory 3-4:

EE203 Digital Systems Design

Filiz Köse 042301071

Kerim Ercan 042401167

MEF University, Istanbul, Turkey

Section: 1.1

Date Performed: 8.11.2024

Submission Date: 15.11.2024

Abstract

As part of the EE203 Digital Systems Design lab, this paper describes the design and implementation of a 1-bit Arithmetic Logic Unit (ALU). The goal was to use multiplexers and AOI gates to create an ALU module that was both efficient and minimally gate-consuming. This was confirmed by breadboard implementation and Logisim simulation. Testing arithmetic operations in a 2's complement system with overflow detection was one of the main tests. The findings lay the groundwork for more intricate digital systems by supporting scalable designs for upcoming ALU extensions.

1.Introduction

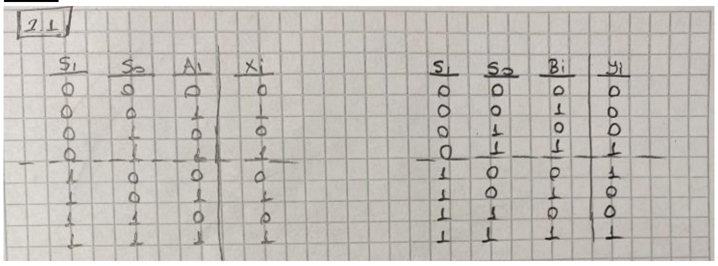
Designing and implementing a 1-bit Arithmetic Logic Unit (ALU), a crucial part of digital electronics used for simple arithmetic and logic operations, was the aim of this exercise. We developed a small and effective ALU with minimal gate consumption by using both multiplexers and AOI gates in a modular design approach.

Our approach integrated overflow detection for accuracy and allowed for both positive and negative numbers while conducting arithmetic operations in a 2's complement system. In order to ensure operation in both virtual and real contexts, the design was first validated through simulations in Logisim and then physically implemented on a breadboard.

The essential abilities for creating scalable ALUs are introduced in this experiment, setting the stage for upcoming experiments involving digital system design and fundamental CPU architecture. The report discusses design choices, implementation procedures, and important testing results.

2.Theory

2.1

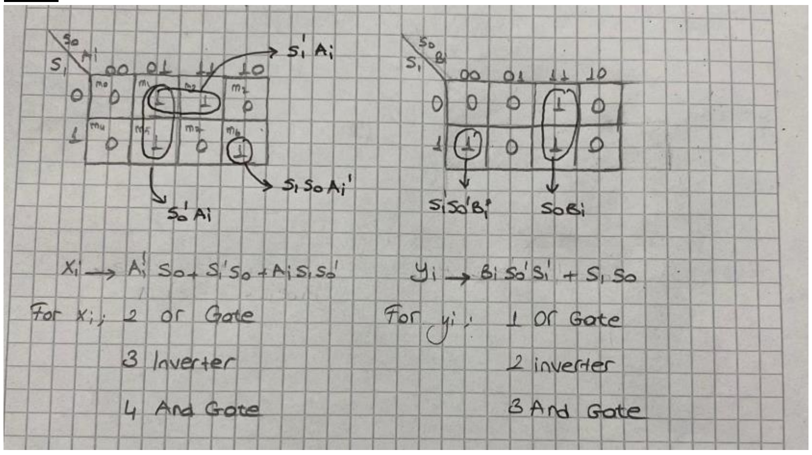


2.1

S_1	S_0	A_i	X_i
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

S_1	S_0	B_i	y_i
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

2.2



$X_i \rightarrow A_i' S_0 + S_1' S_0 + A_i S_1 S_0'$

For X_i; 2 or Gate

3 Inverter

4 And Gote

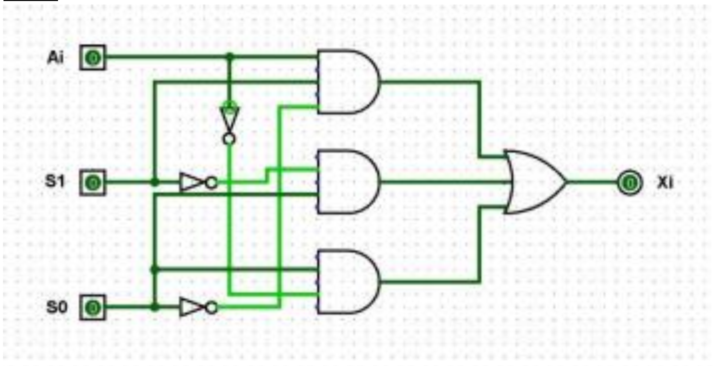
$y_i \rightarrow B_i S_0' S_1' + S_1 S_0$

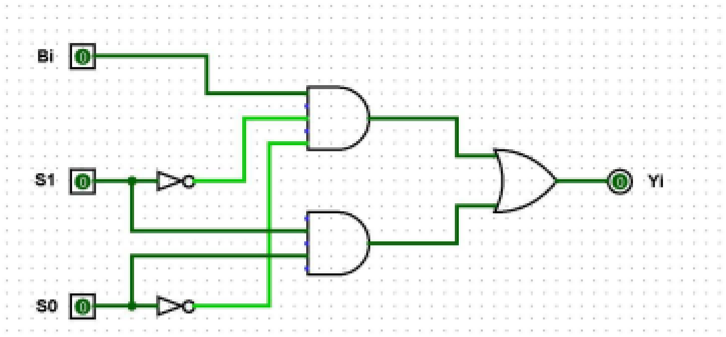
For y_i: 1 Or Gate

2 inverter

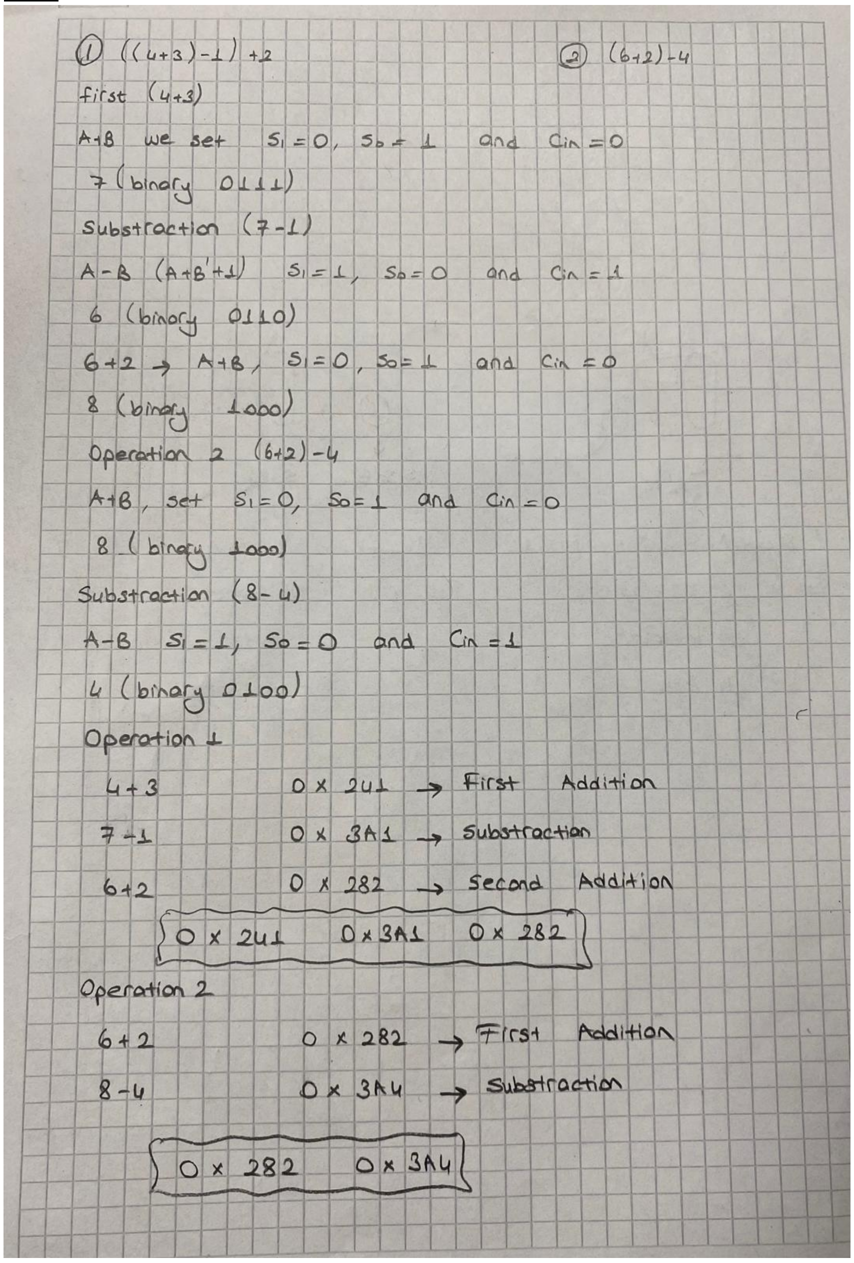
3 And Gate

2.3





2.4



① ((4+3)-1)+2 ② (6+2)-4

first (4+3)

A+B we set $S_1 = 0$, $S_0 = 1$ and $C_{in} = 0$

7 (binary 0111)

Substraction (7-1)

A-B (A+B'+1) $S_1 = 1$, $S_0 = 0$ and $C_{in} = 1$

6 (binary 0110)

6+2 → A+B, $S_1 = 0$, $S_0 = 1$ and $C_{in} = 0$

8 (binary 1000)

Operation 2 (6+2)-4

A+B, set $S_1 = 0$, $S_0 = 1$ and $C_{in} = 0$

8 (binary 1000)

Substraction (8-4)

A-B $S_1 = 1$, $S_0 = 0$ and $C_{in} = 1$

4 (binary 0100)

Operation 1

4+3	0 x 241	→ First Addition
7-1	0 x 3A1	→ Substraction
6+2	0 x 282	→ Second Addition

$\boxed{0 \times 241 \quad 0 \times 3A1 \quad 0 \times 282}$

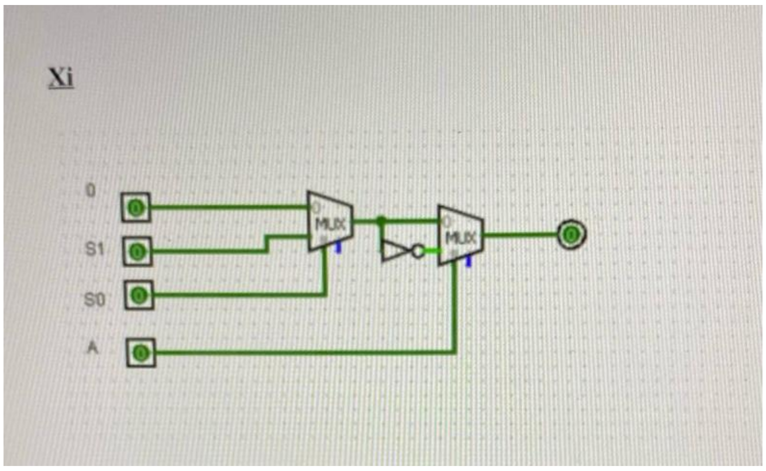
Operation 2

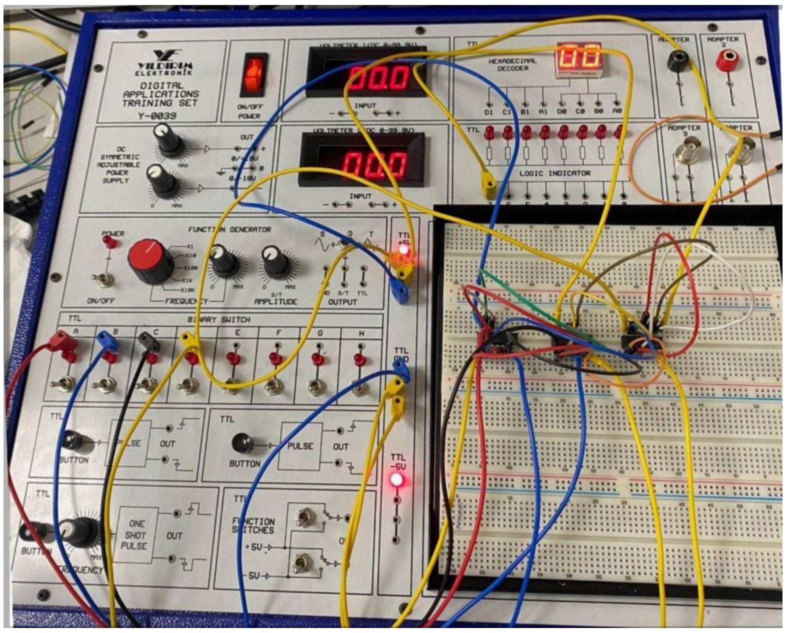
6+2	0 x 282	→ First Addition
8-4	0 x 3A4	→ Substraction

$\boxed{0 \times 282 \quad 0 \times 3A4}$

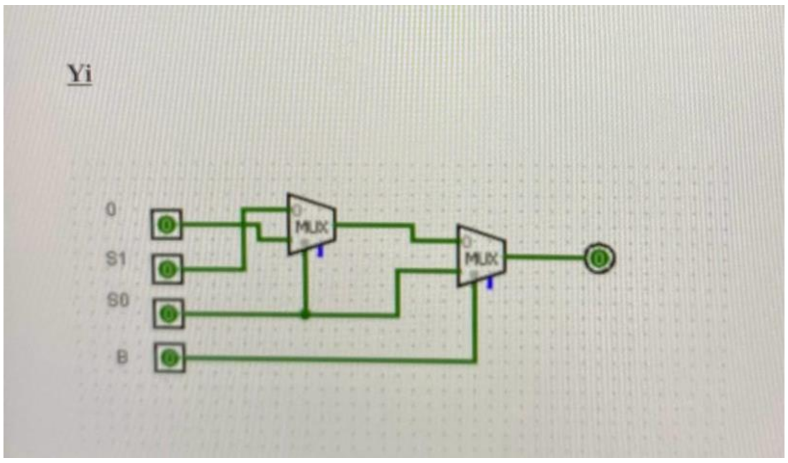
3. Experimental Results

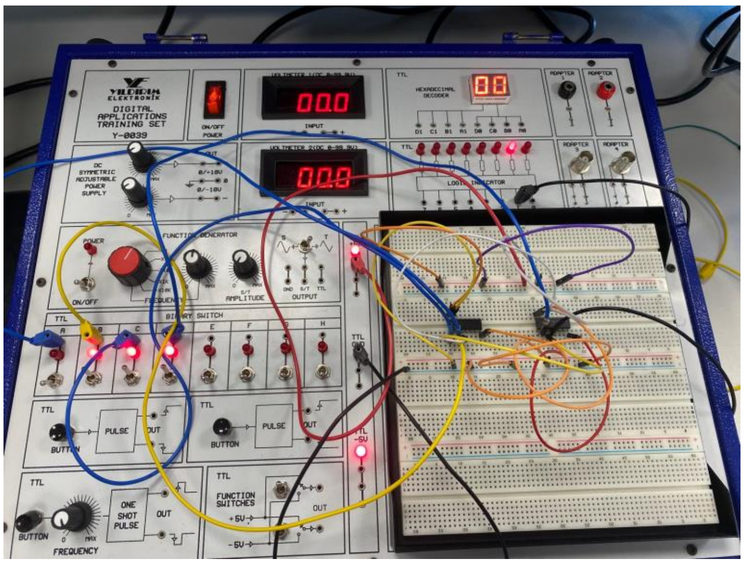
3.1 (X_i)



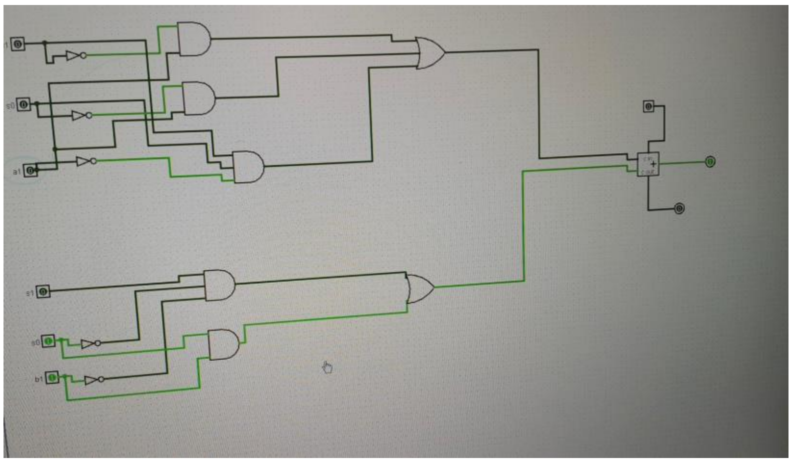


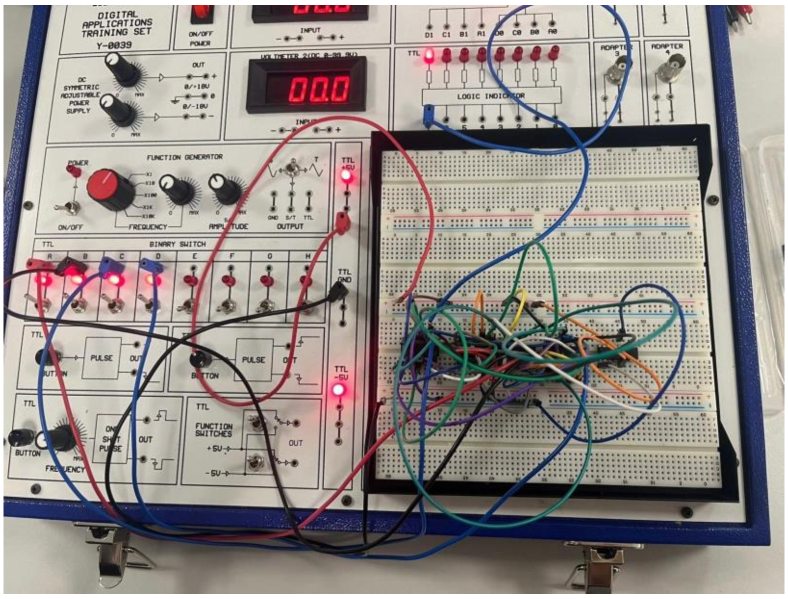
3.1 (Y_i)



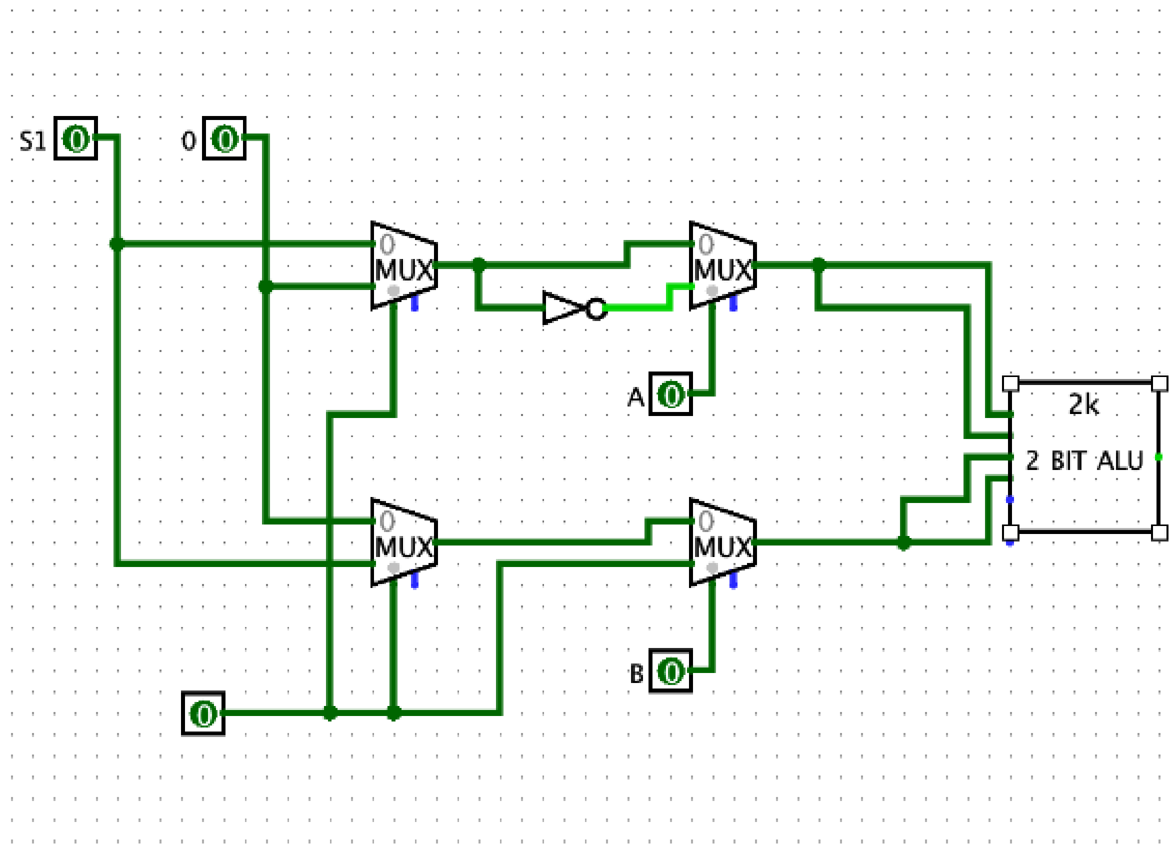


<u>3.2</u>




DIGITAL APPLICATIONS TRAINING SET
Y-0039
VOLTMETER
LOGIC INDICATOR
FUNCTION GENERATOR
BINARY SWITCH
PULSE
FUNCTION SWITCHES

4. Postlab



S3 S2 S1	A1 A0	B1 B0	C0	Operation	Output
0 0 0	01	10	0	AND	0000
0 0 1	01	10	0	OR	0111
0 1 0	01	10	0	XOR	0111
0 1 1	01	10	0	ADD	1011
1 0 0	01	10	0	SUB	0011
1 0 1	01	10	0	SLT	0001
1 1 0	01	10	0	NOR	1000
1 1 1	01	10	0	MUX	1011

Additional operations for 0x13 and 0x14 yield valid results. In the case of 0x5D, the 2-bit ALU limit is exceeded and an error occurs.

5. Discussion of Experiment Result

5.1

Why is the use of multiplexers preferred?
Using a multiplexer makes less logic gates so it makes it more efficient.
What changes when transitioning from a 1-bit ALU to a 2-bit ALU?
2-bit ALU can process more data because of that overflow risk increse.So we need to use more advanced control signals.
What can be done in case of overflow?
Using an overflow detector and providing an error message can be a solution.

5.2

Gates= AND Gates 7408, OR Gates 7432, MUX Gates 74153, and ADDER 7400.
Circuit Board
Cables
Power supply and connections

5.3

An approach to change design that lowers the delay caused by ripple deliver logic in a 4-bit ALU is to use a Carry Look-Ahead (CLA) adder. A Carry Look-Ahead adder may also greatly reduce the propagation time associated with the bring chain in ripple bring adders.
In a ripple carry adder, each bit's bring-out is determined by the convey-in from the bit before it. This sets off a series of events that cause the entire computation to be delayed.
Carry Look-Ahead adders in assessment allow the carry chain to be computed in parallel by instantly computing carry indicators for each bit location.

5.4

Although more complex adder architectures in conjunction with deliver look-beforehand or bring-keep adders can offer better velocity and performance for larger word sizes the choice to use ripple carry adders frequently depends on the particular needs of the utility and the design goals. Despite their limitations in terms of pace and place performance, ripple conveyers are a reasonable choice for academic or small word sizes due to their simplicity, ease of use, and versatility

6. Append

S_1	S_0	A_i	X_i (Logic X)	S_1	S_0	B_i	Y_i (logic Y)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	1
1	0	1	1	1	0	1	0
1	1	0	1	1	1	0	0
1	1	1	0	1	1	1	1

#	*A*	*B*	operation	result	underflow	overflow
1	0101	0010	A+B	0111	-	-
1	0111	0011	A-B	0100	-	-
1	0100	0001	A+B	0101	-	-
2	0101	0010	A+B	0111	-	-
2	1001	011	A-B	11010	1	-

7. Labor

Task	**Group Member Responsible**
Prepare Prelab, Set Up Logic Circuit and Testing, Write Report	Filiz Köse
Set Up Logic Circuit and Testing, Write Report	Kerim Ercan

Filiz Köse **Kerim Ercan**