(CSE111) Logic Design

Sophomore CESS

FALL 2022

Team (20)

Major Task

Name	ID
Ahmed Ehab Mohamed El-Baramony	21P0261
Ahmed Mohamed Mohamed	22P0283
Ahmed Mohamed Hassan El-Henawy	21P0298

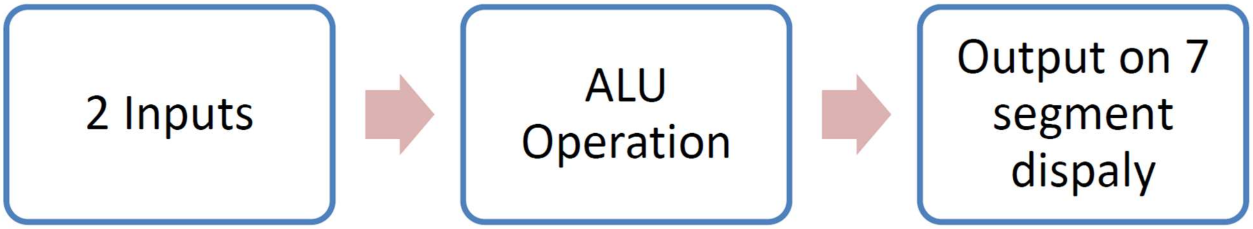
Phase (1)

Combinational Circuit

Overview:

ALU (Arithmetic Logical Unit) that have 2 inputs **4-Bit** each
A {A_0, A_1, A_2, A_3} & **B {B_0 , B_1, B_2, B_3}**
that can perform:

- 2 Arithmetic Operations:
    - **A + B** (Addition)
    - **A + 1** (Increment)
- 2 Logical Operations:
    - **A AND B** (Bitwise AND)
    - **A OR B** (Bitwise OR)



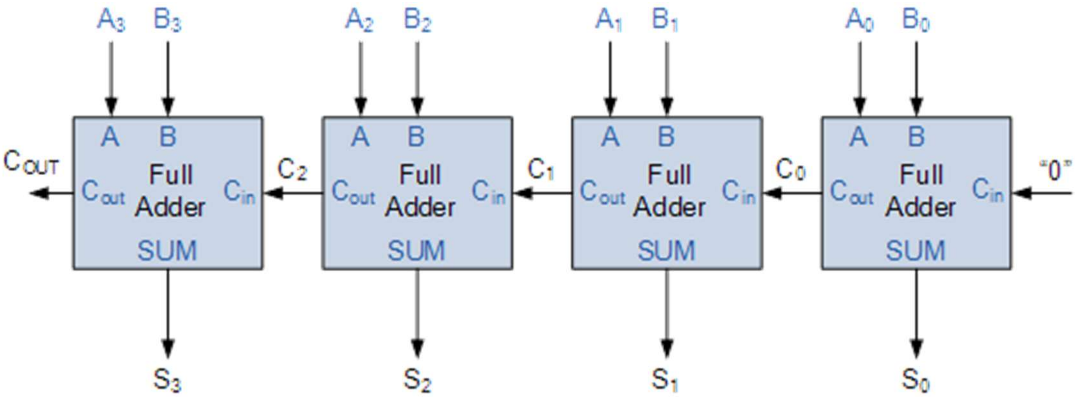
2 Inputs

ALU Operation

Output on 7 segment dispaly

Addition:

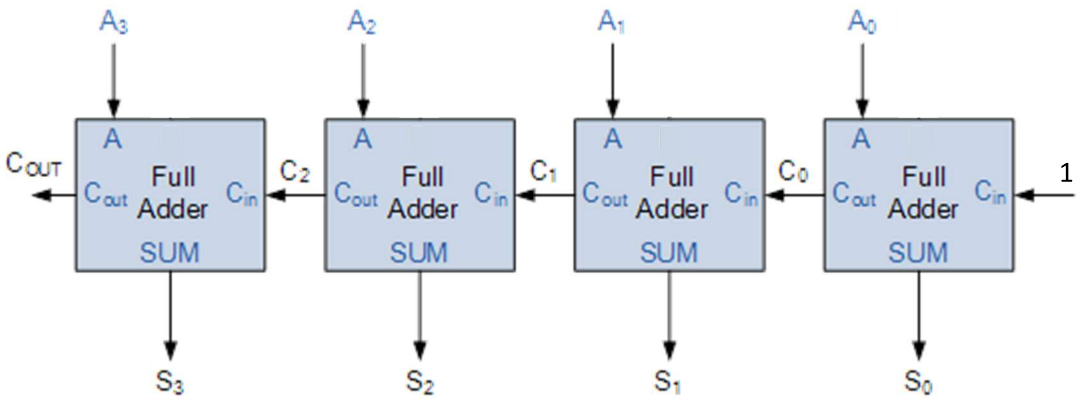
Addition Operation (A+B)													
C_{In}	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	S_3	S_2	S_1	S_0	C_{Out}
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	1	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0
0	0	0	1	1	0	0	1	1	0	1	1	0	0
0	0	1	0	0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	1	0	1	1	0	1	0	0
0	0	1	1	0	0	1	1	0	1	1	0	0	0
0	0	1	1	1	0	1	1	1	1	1	1	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1	0	0	1	0	1
0	1	0	1	0	1	0	1	0	0	1	0	0	1
0	1	0	1	1	1	0	1	1	0	1	1	0	1
0	1	1	0	0	1	1	0	0	1	0	0	0	1
0	1	1	0	1	1	1	0	1	1	0	1	0	1
0	1	1	1	0	1	1	1	0	1	1	0	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0	1



In the **Addition Operation** we used a **4-Bit Adder 7483 IC** with a C_{In} = 0

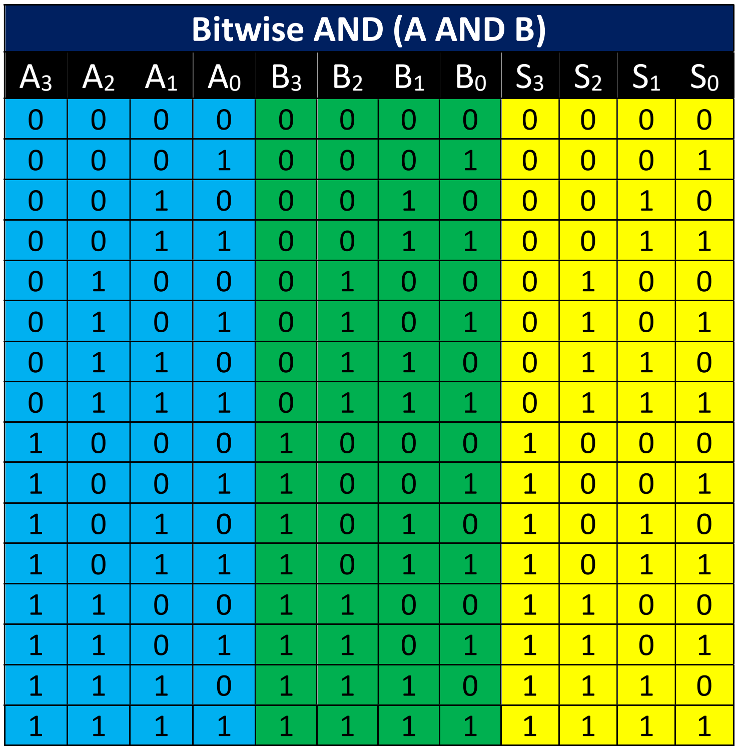
Increment:

Increment By 1									
C_{In}	A_3	A_2	A_1	A_0	S_3	S_2	S_1	S_0	C_{Out}
1	0	0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	1	0	0
1	0	0	1	0	0	0	1	1	0
1	0	0	1	1	0	1	0	0	0
1	0	1	0	0	0	1	0	1	0
1	0	1	0	1	0	1	1	0	0
1	0	1	1	0	0	1	1	1	0
1	0	1	1	1	1	0	0	0	0
1	1	0	0	0	1	0	0	1	0
1	1	0	0	1	1	0	1	0	0
1	1	0	1	0	1	0	1	1	0
1	1	0	1	1	1	1	0	0	0
1	1	1	0	0	1	1	0	1	0
1	1	1	0	1	1	1	1	0	0
1	1	1	1	0	1	1	1	1	0
1	1	1	1	1	0	0	0	0	1



In the **Increment Operation** we used a **4-Bit Adder 7483 IC** with a C_{In} = 1

Bitwise AND:



Bitwise AND (A AND B)											
A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	1	1	0	0	1
1	0	1	0	1	0	1	0	1	0	1	0
1	0	1	1	1	0	1	1	1	0	1	1
1	1	0	0	1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	1	1	1	0	1
1	1	1	0	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1

In the **Bitwise AND Operation** we used a **7408 IC**

Equations:

$A_0 . A_0 = S_0$

$A_1 . A_1 = S_1$

$A_2 . A_2 = S_2$

$A_3 . A_3 = S_3$

Bitwise OR:

Bitwise OR (A OR B)											
A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0	0	1	1	0
0	1	1	1	0	1	1	1	0	1	1	1
1	0	0	0	1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	1	1	0	0	1
1	0	1	0	1	0	1	0	1	0	1	0
1	0	1	1	1	0	1	1	1	0	1	1
1	1	0	0	1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	1	1	1	0	1
1	1	1	0	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1	1

In the **Bitwise OR Operation** we used a **7432 IC**

Equations:

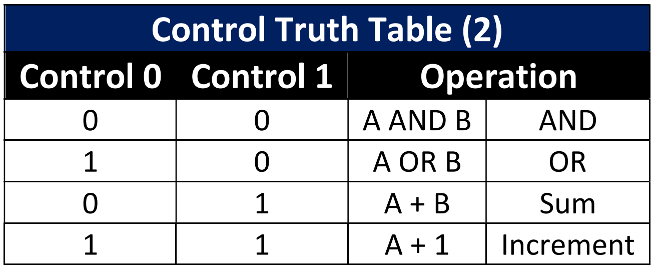
$A_0 + A_0 = S_0$

$A_1 + A_1 = S_1$

$A_2 + A_2 = S_2$

$A_3 + A_3 = S_3$

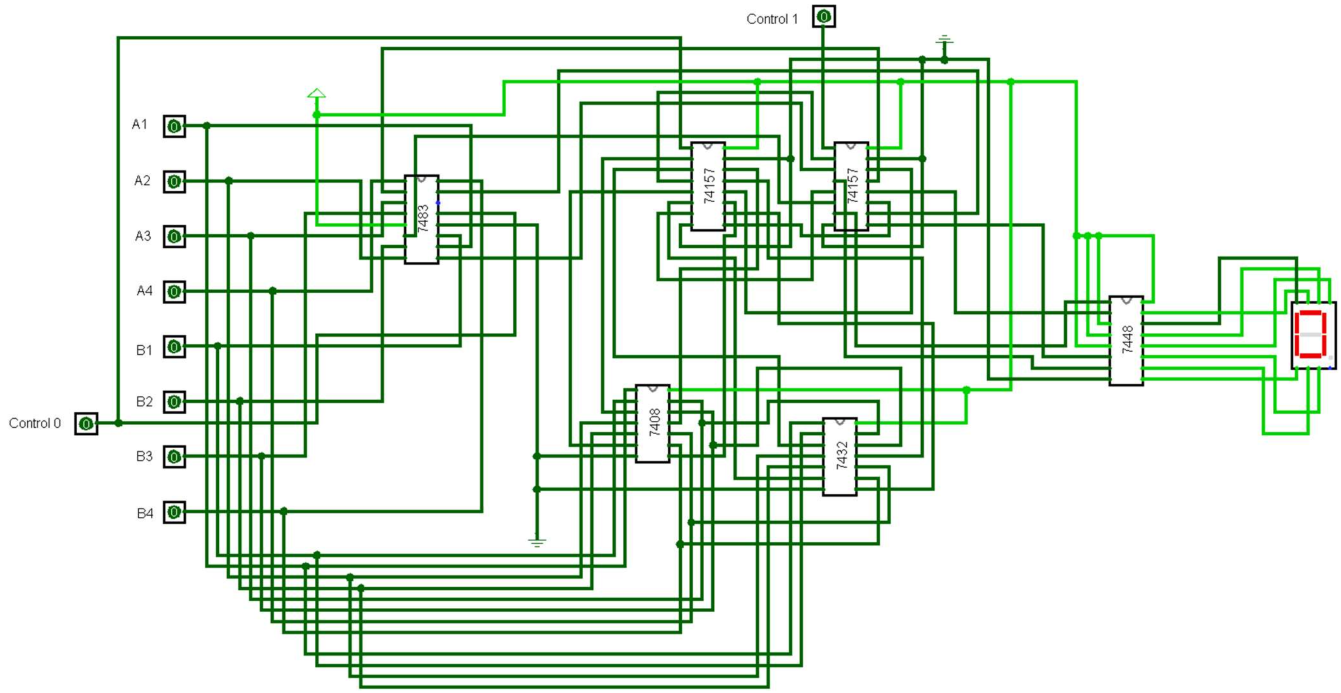
Control Truth Table:



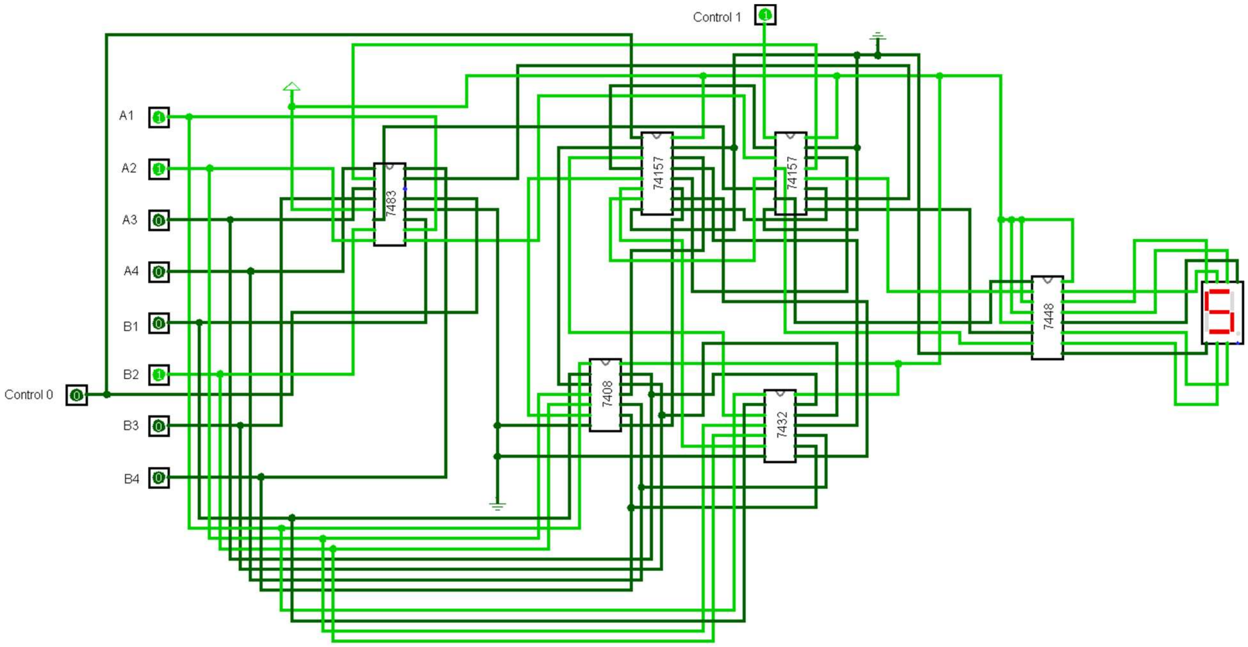
<table>
<tr><th colspan="4">Control Truth Table (2)</th></tr>
<tr><th>Control 0</th><th>Control 1</th><th colspan="2">Operation</th></tr>
<tr><td>0</td><td>0</td><td>A AND B</td><td>AND</td></tr>
<tr><td>1</td><td>0</td><td>A OR B</td><td>OR</td></tr>
<tr><td>0</td><td>1</td><td>A + B</td><td>Sum</td></tr>
<tr><td>1</td><td>1</td><td>A + 1</td><td>Increment</td></tr>
</table>

We used a dip switch to control the operations that is displayed on 7-Segment Display using this table.

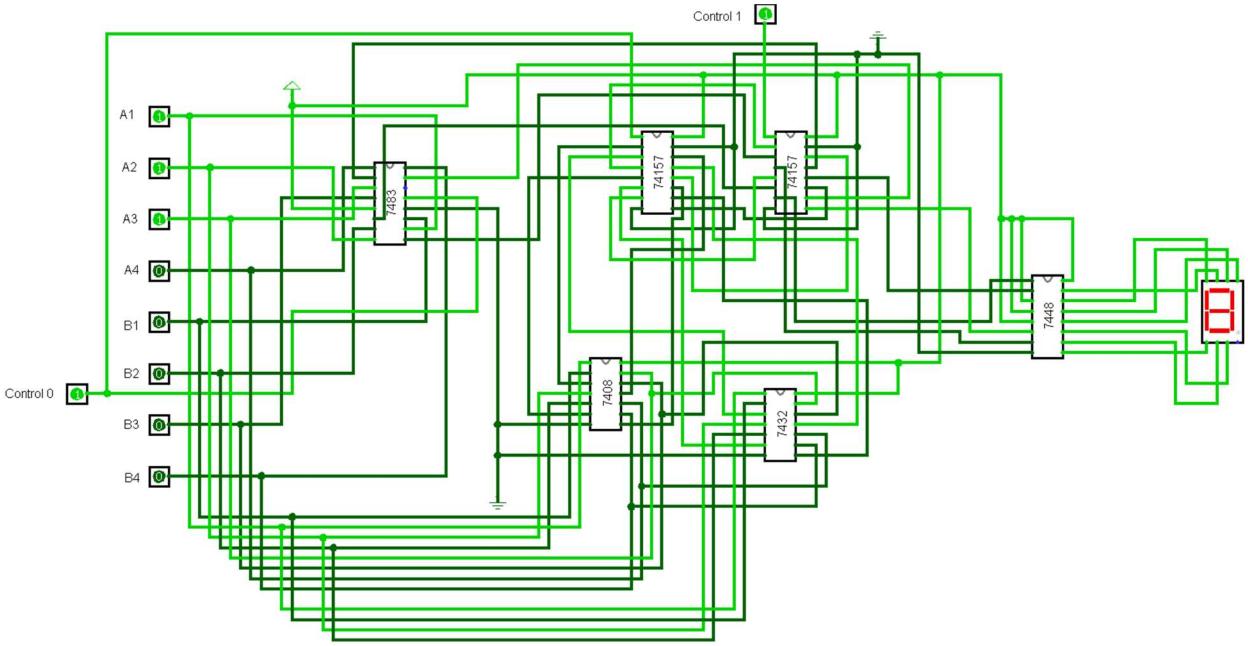
Logic Circuit (Logisim):



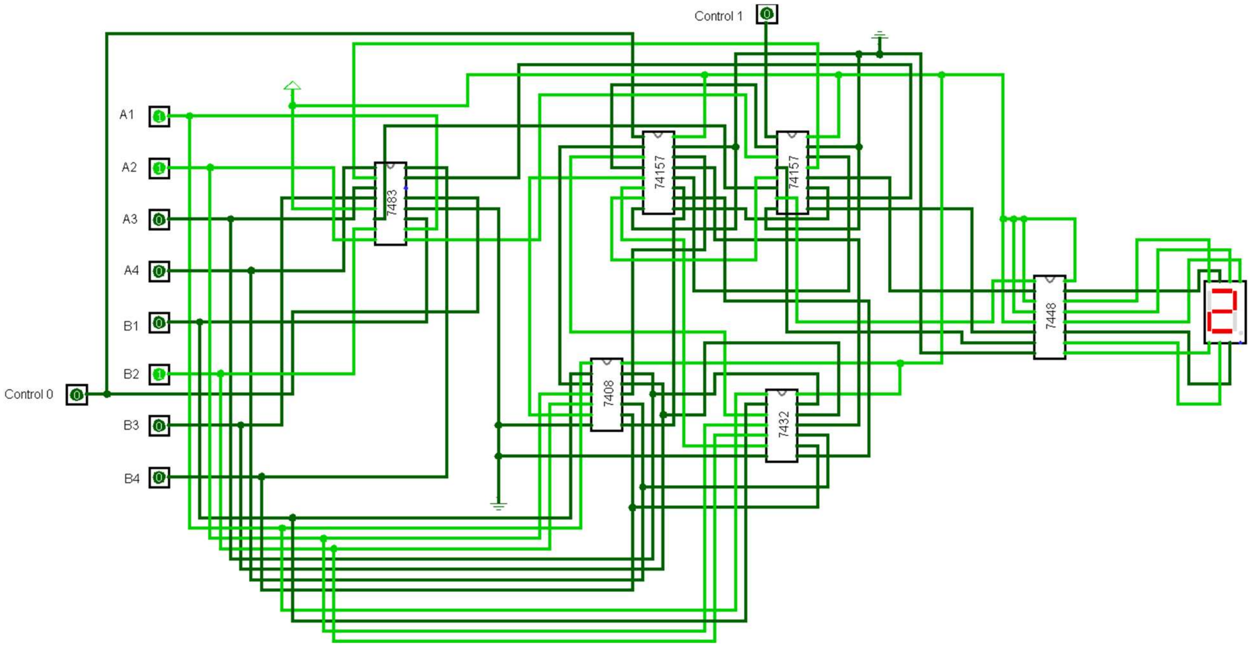
Addition Simulation:



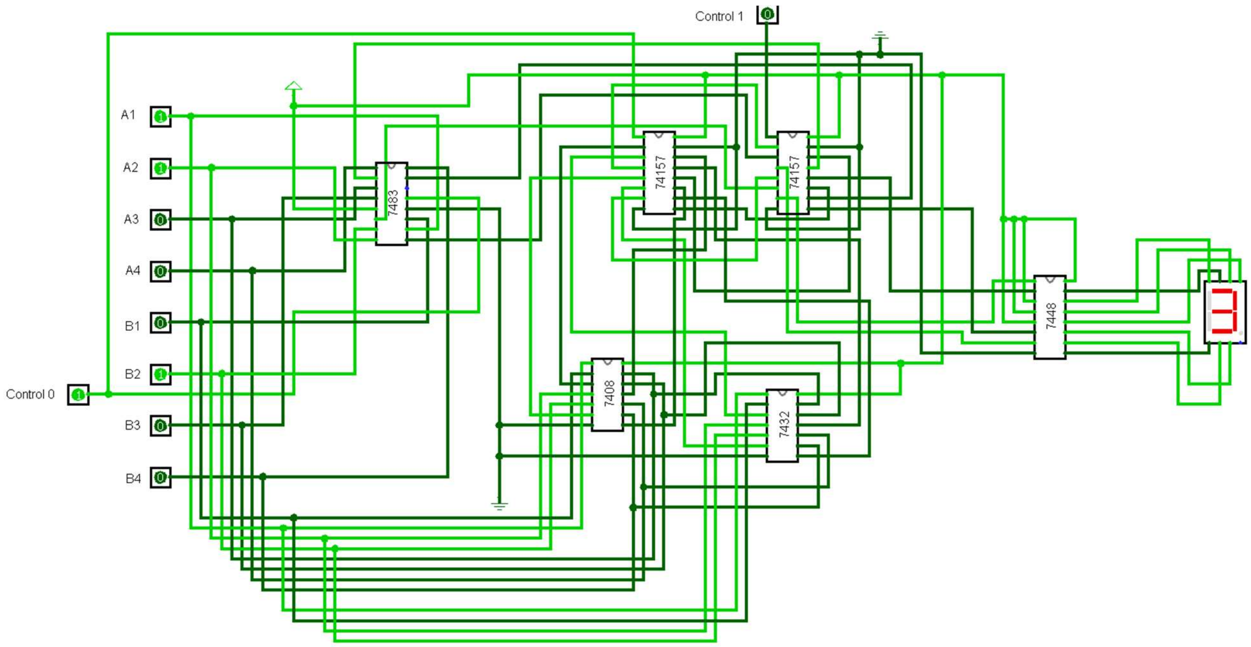
Increment Simulation:



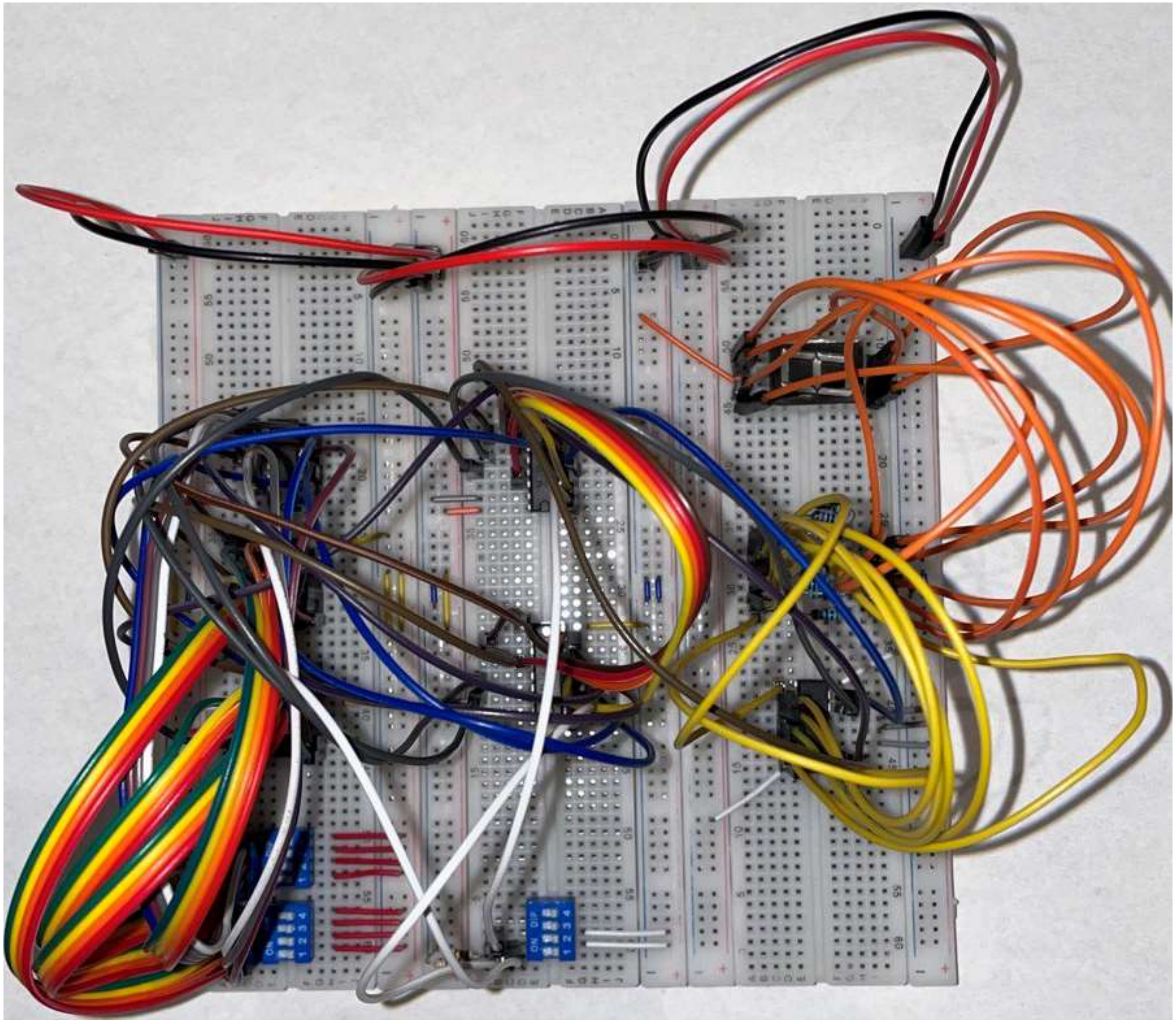
AND Simulation:

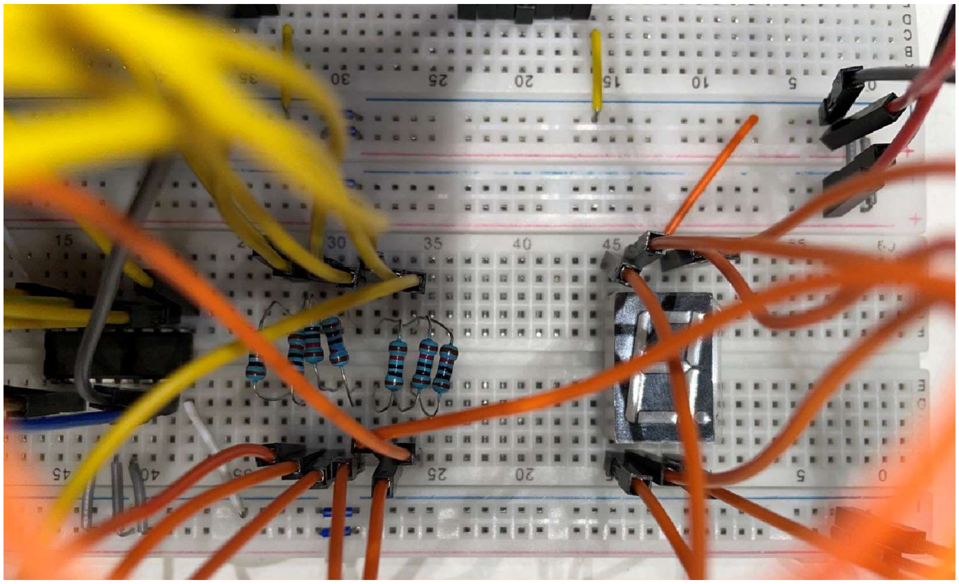


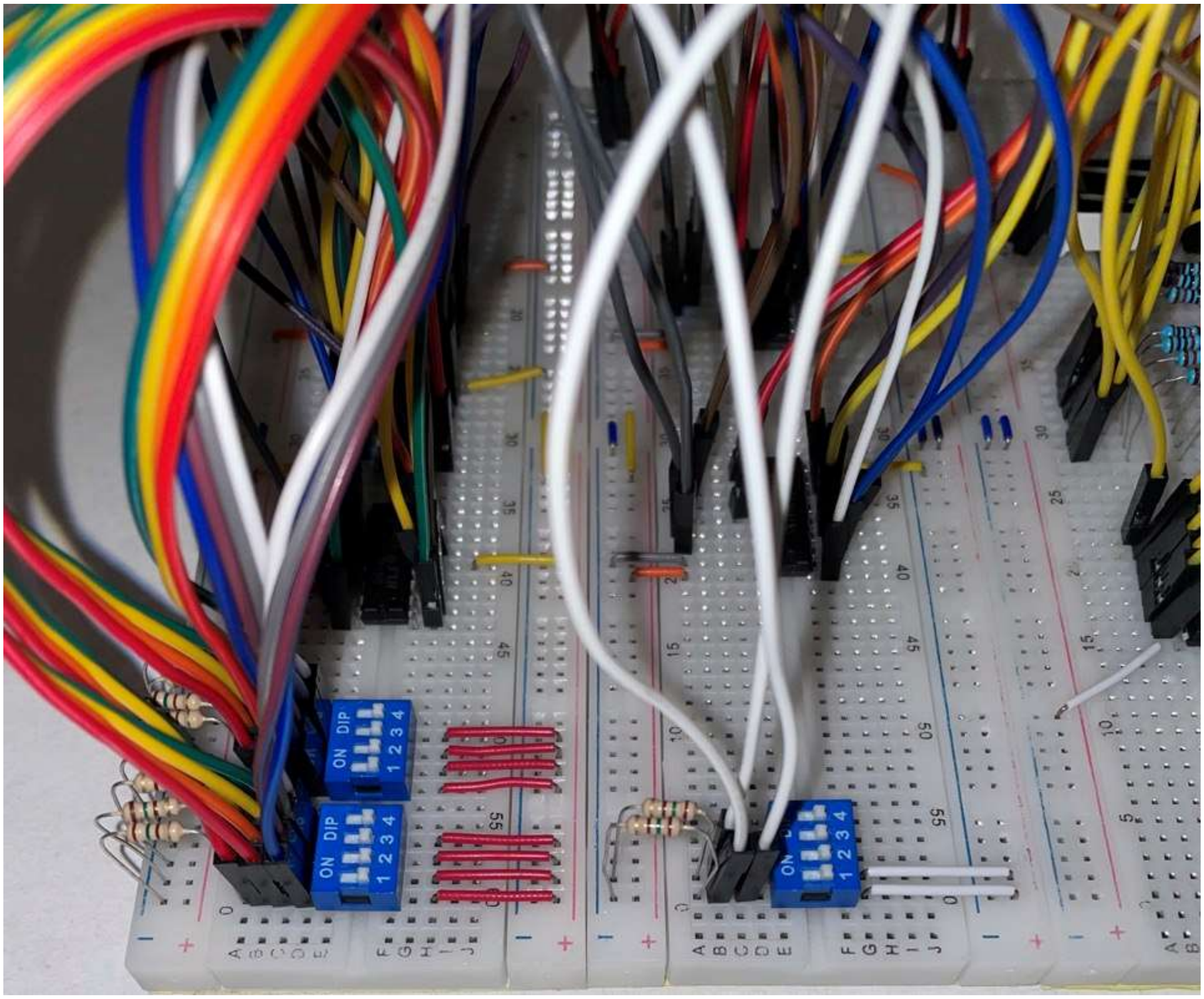
OR Simulation:

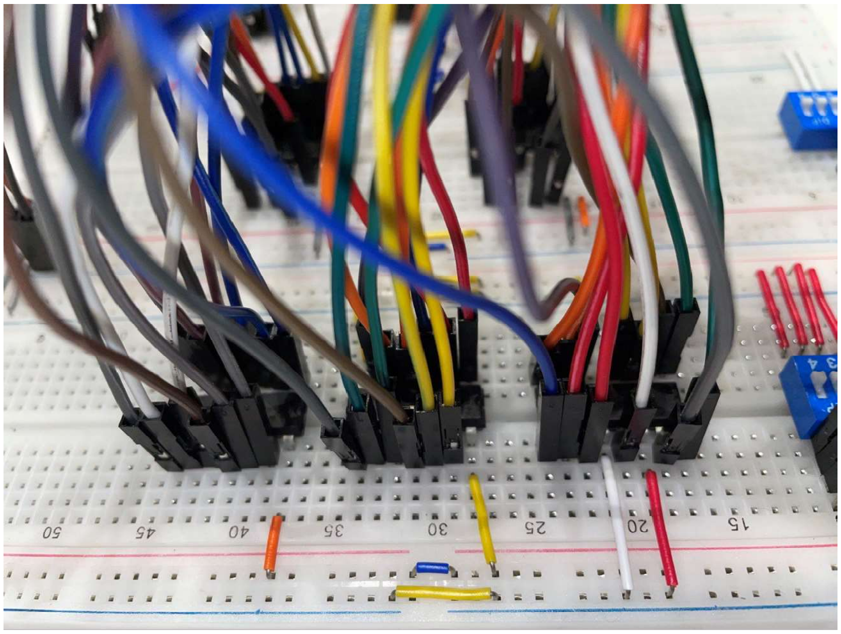


Hardware Implementation:



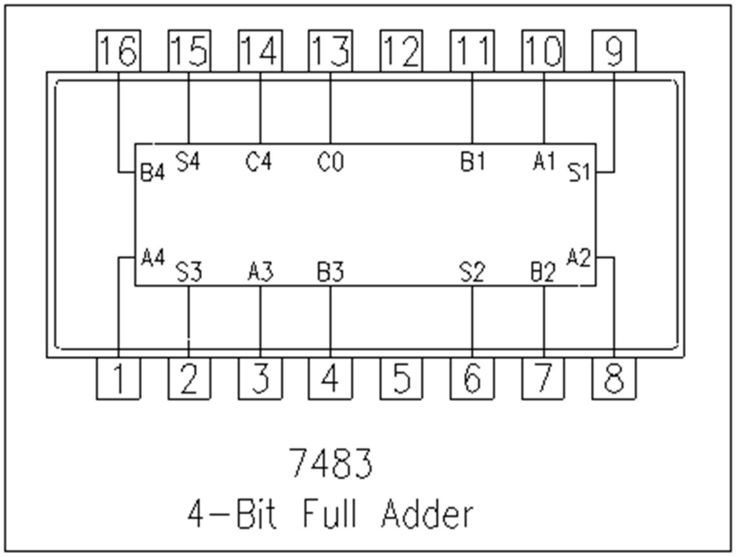


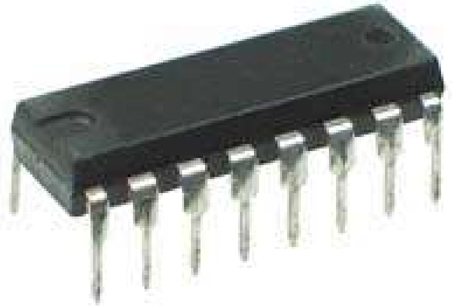




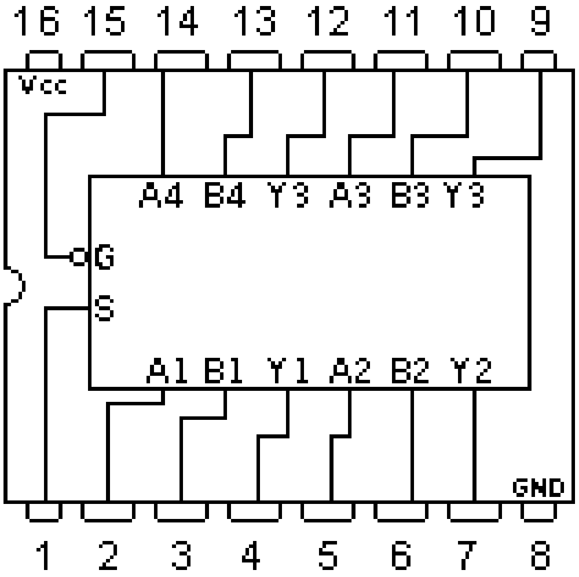
ICs Used In Hardware:

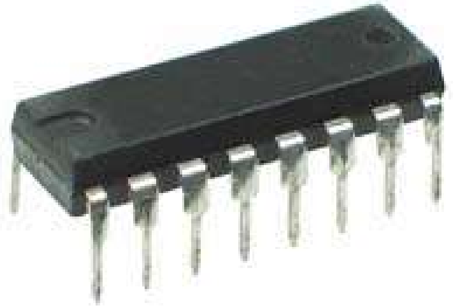
7483 (4-Bit Full Adder)



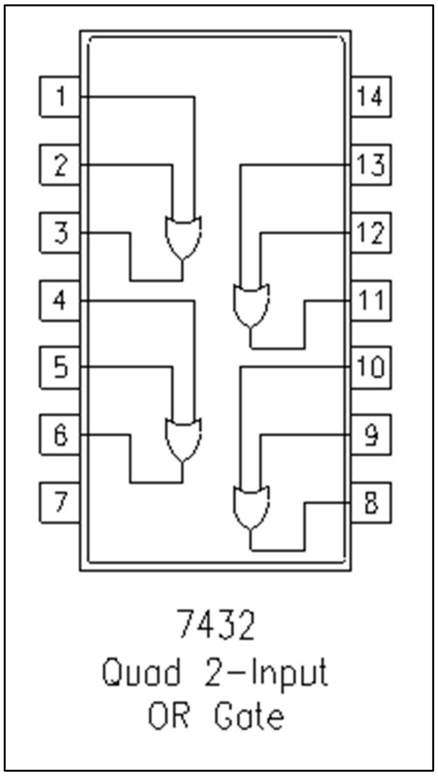


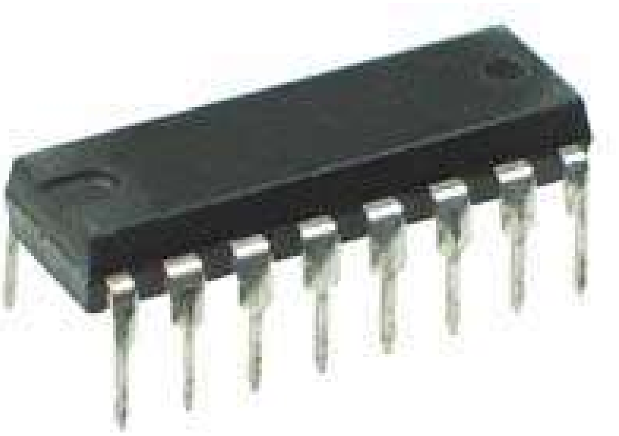
74157 (Quad 2-1 MUX)



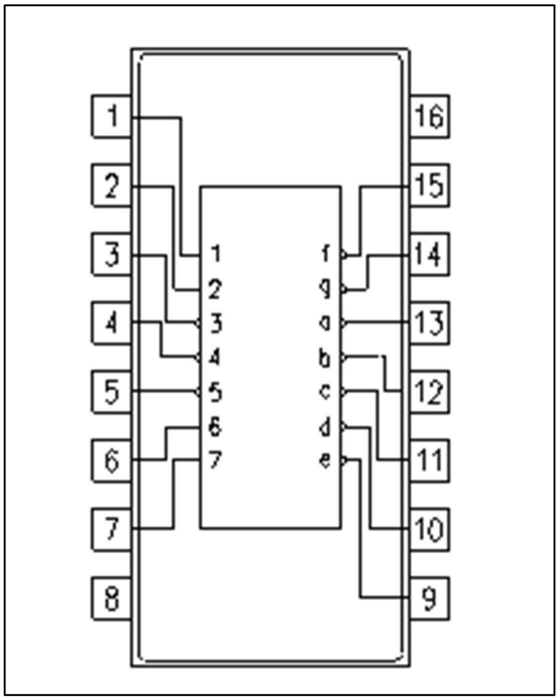


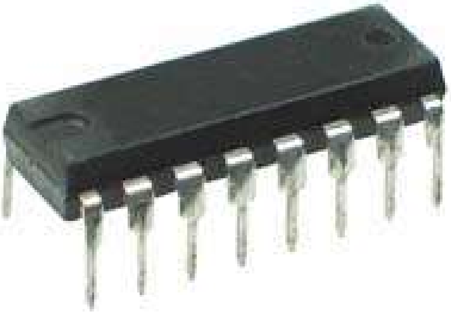
7432 (Quad 2-Input OR Gate)

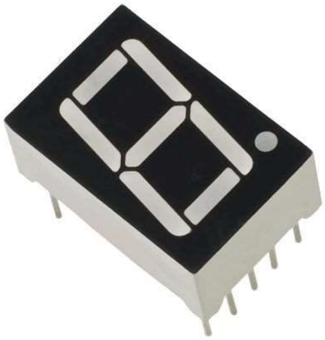




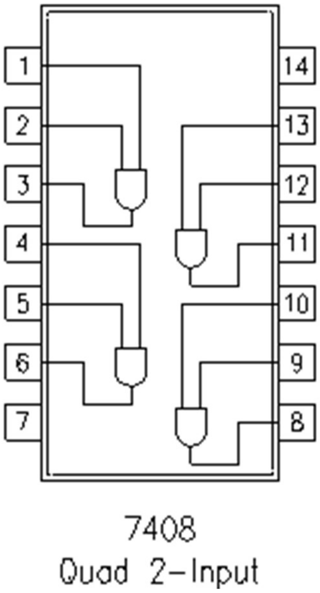
7448 (BCD To 7-Segment Decoder "Common Cathode")

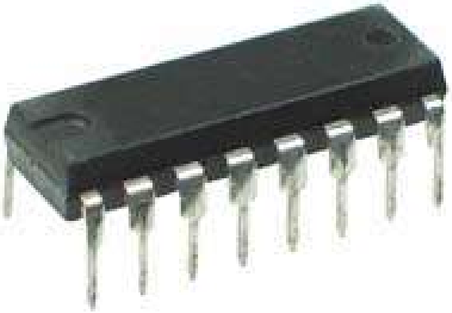






7408 (Quad 2-Input AND)





Phase (2)

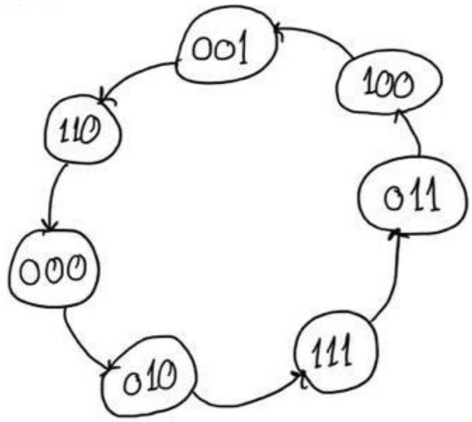
Sequential Circuit

Overview:

Sequential circuit with **JK-type flip-flops** and logic gates to count the sequence:

{1 , 6 , 0 , 2 , 7 , 3 , 4 , 1}, and repeat; then display it on 7-segment.

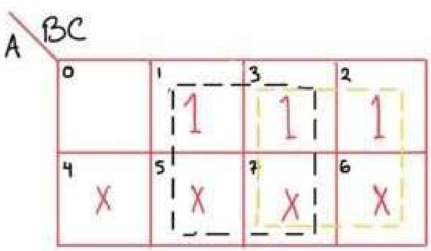
State Diagram:



State Table:

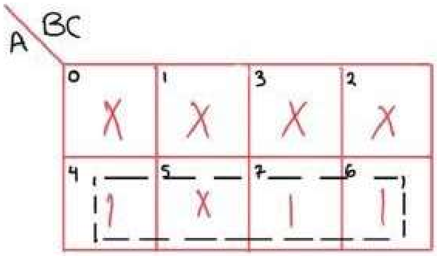
	present state			Next state			F.F inputs					
	A	B	C	A(t+1)	B(t+1)	C(t+1)	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	0	X	1	X	0	X
1	0	0	1	1	1	0	1	X	1	X	X	1
2	0	1	0	1	1	1	1	X	X	0	1	X
3	0	1	1	1	0	0	1	X	X	1	X	1
4	1	0	0	0	0	1	X	1	0	X	1	X
5	1	0	1	X	X	X	X	X	X	X	X	X
6	1	1	0	0	0	0	X	1	X	1	0	X
7	1	1	1	0	1	1	X	1	X	0	X	0

K-Maps:



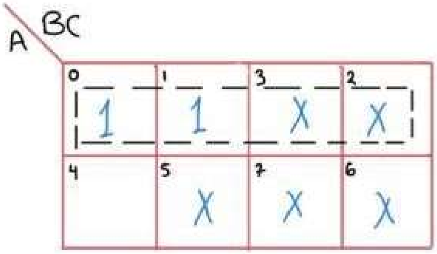
0	1 1	3 1	2 1
4 X	5 X	7 X	6 X

$J_A = C + B$



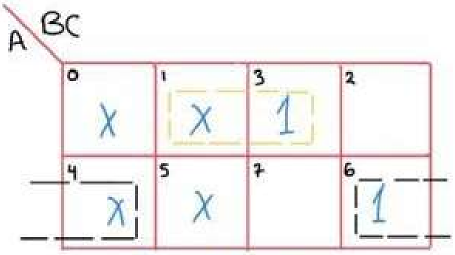
0 X	1 X	3 X	2 X
4 1	5 X	7 1	6 1

$K_A = A$



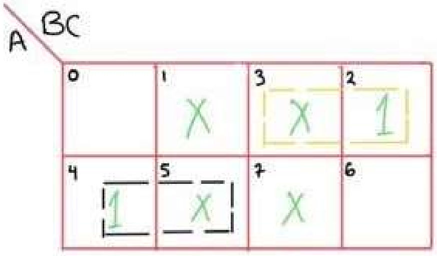
0 1	1 1	3 X	2 X
4	5 X	7 X	6 X

$J_B = A'$



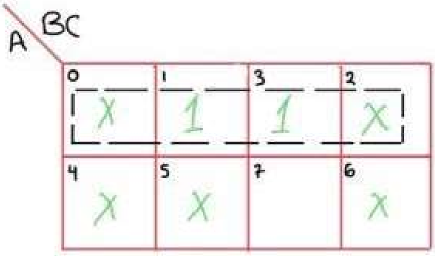
0 X	1 X	3 1	2
4 X	5 X	7	6 1

$k_B = AC' + A'C$
$= A \oplus$



0	1 X	3 X	2 1
4 1	5 X	7 X	6

$J_c = A'B + AB'$
$= A \oplus B$



0 X	1 1	3 1	2 X
4 X	5 X	7	6 X

$K_c = A'$

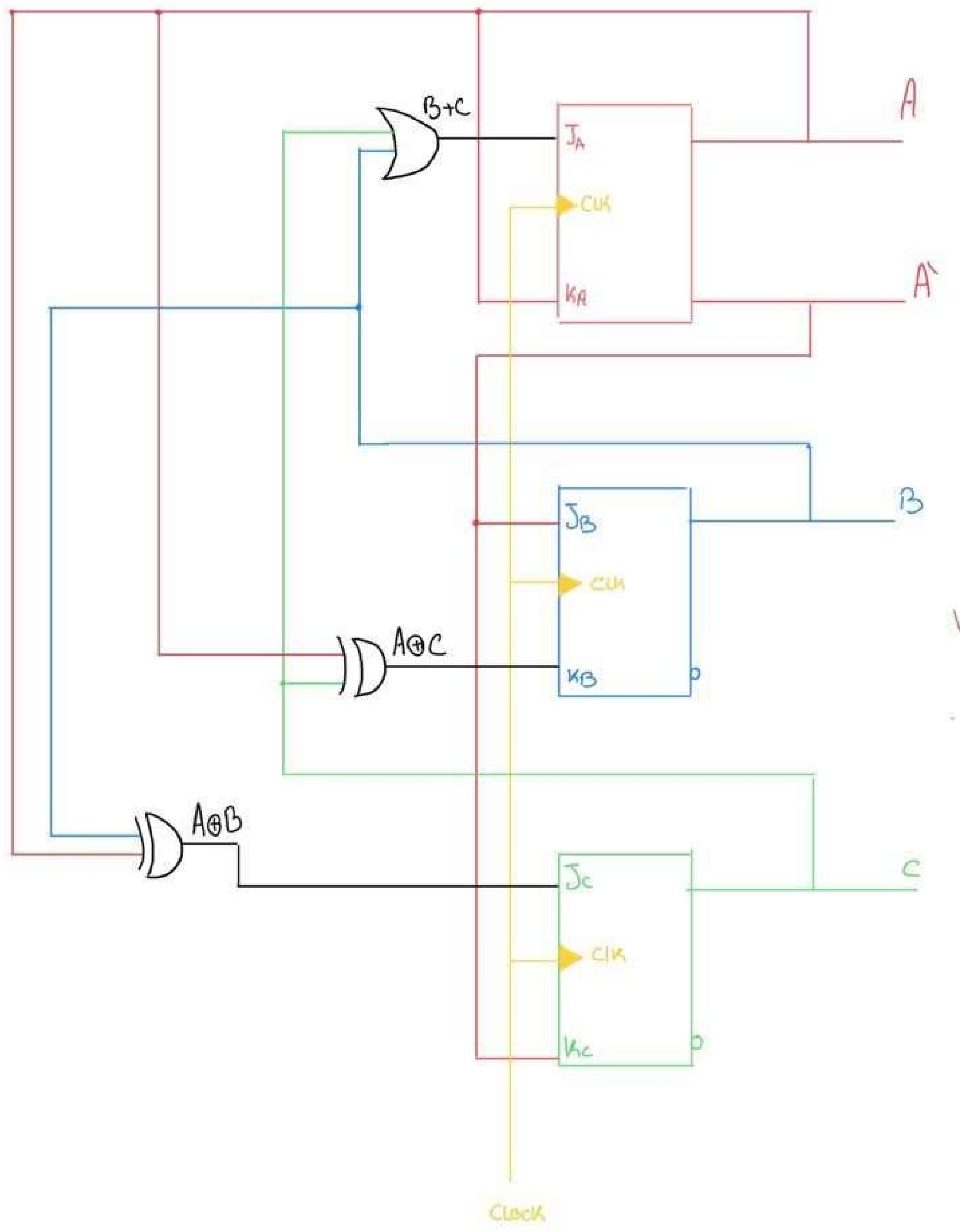
State Equations:

$(J_A = B + C \quad , \quad K_A = A)$

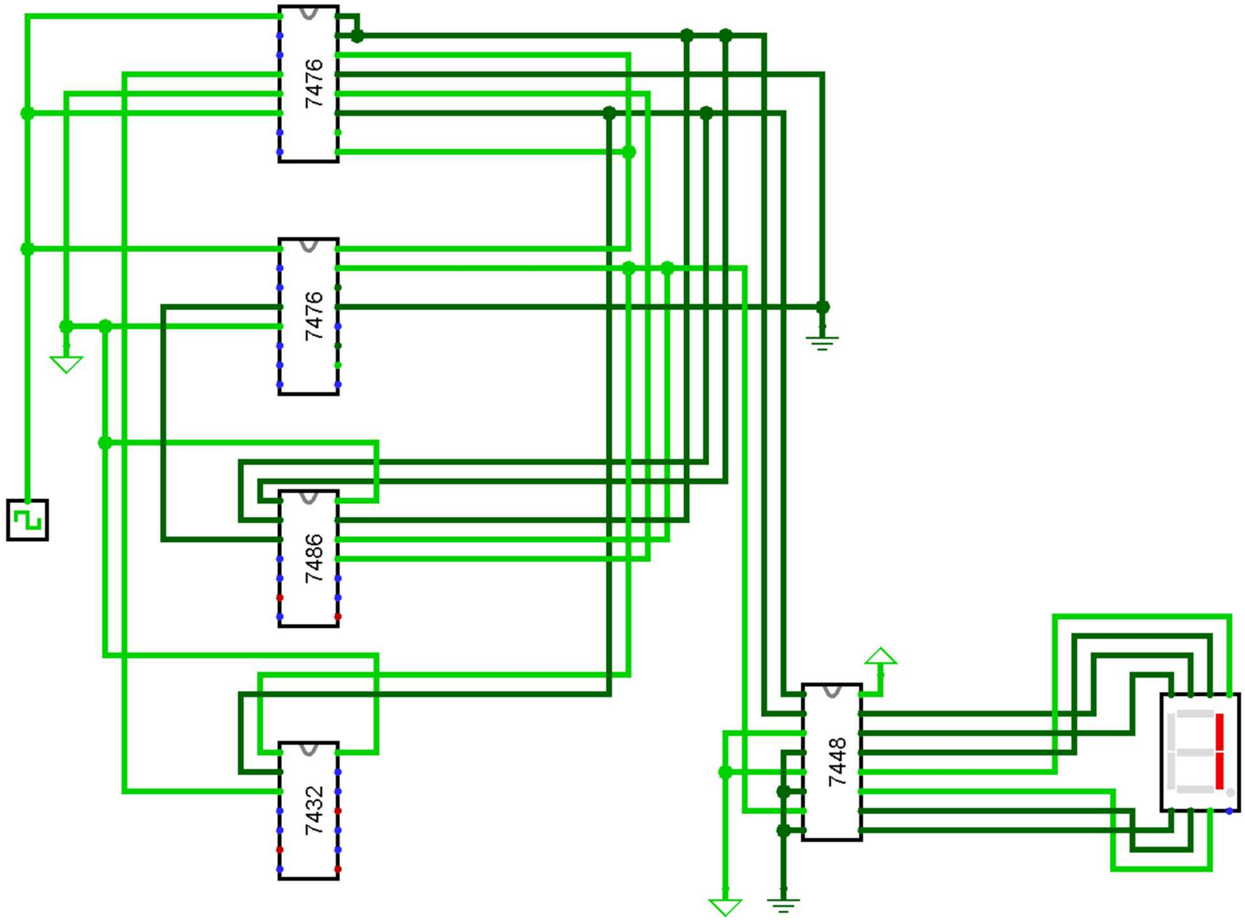
$(J_B = A' \quad , \quad K_B = A \oplus C)$

$(J_c = A \oplus B \quad , \quad K_c = A')$

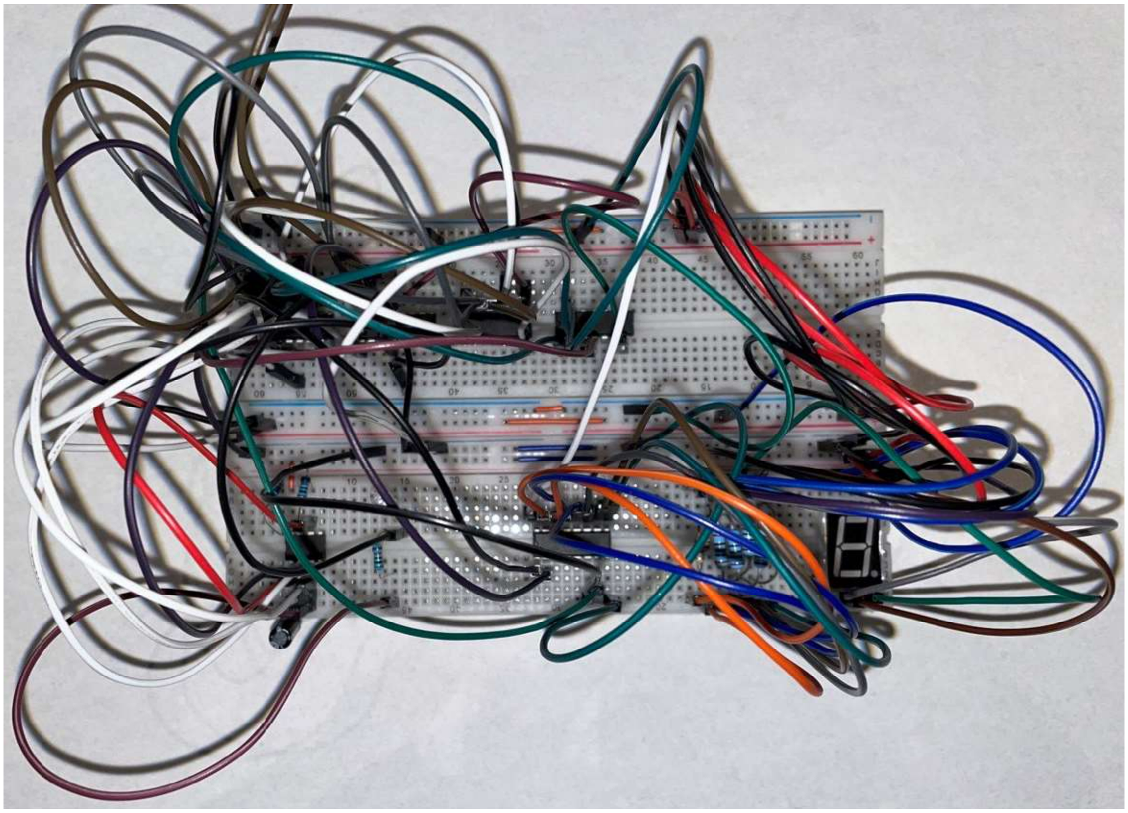
Logic Circuit:

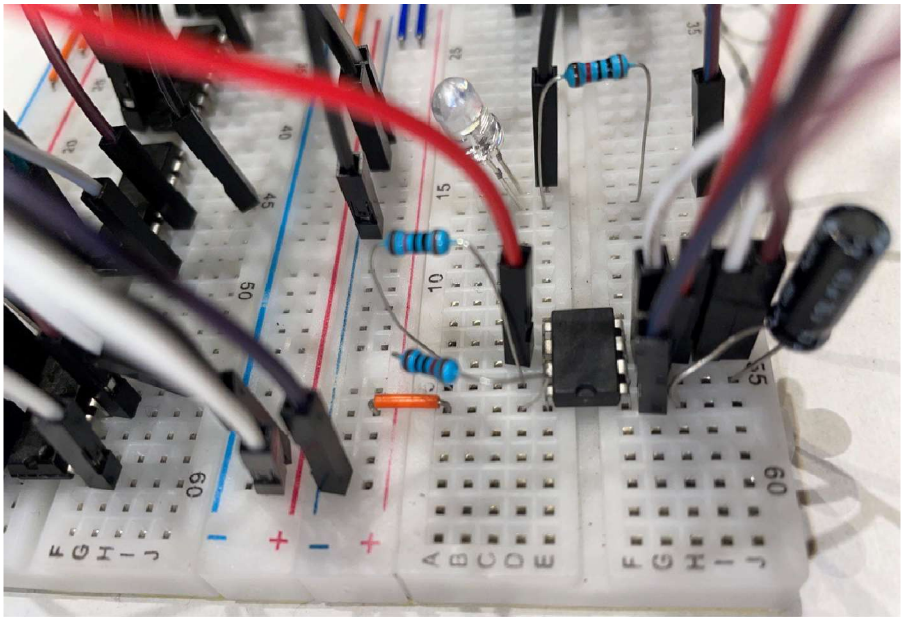


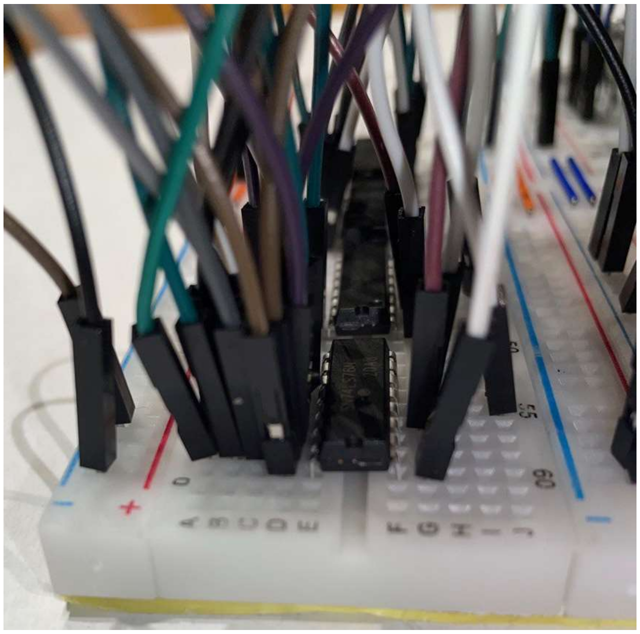
Logic Circuit (Logisim):



Hardware Implementation:

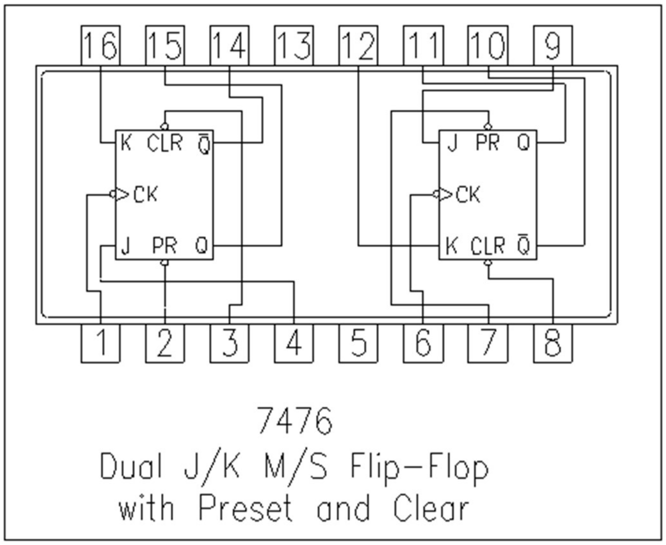


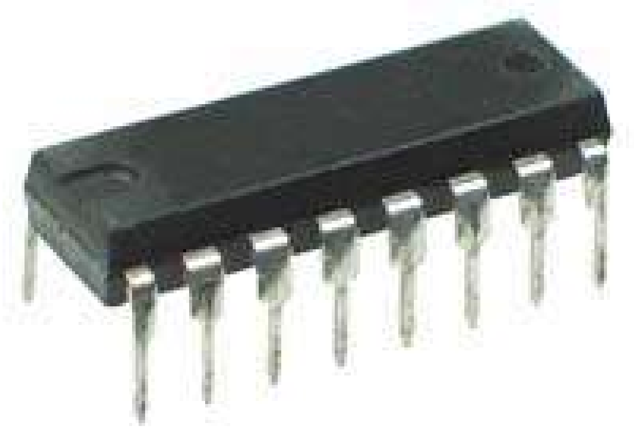




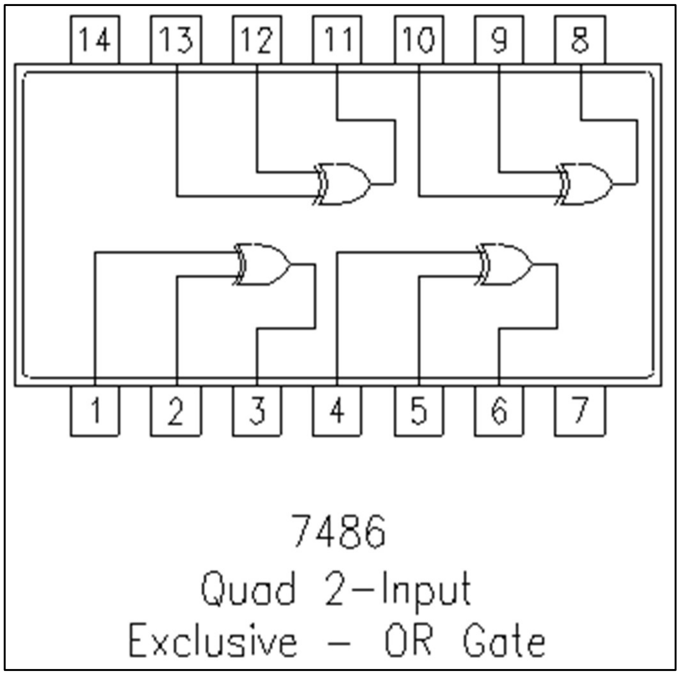
ICs Used In Hardware:

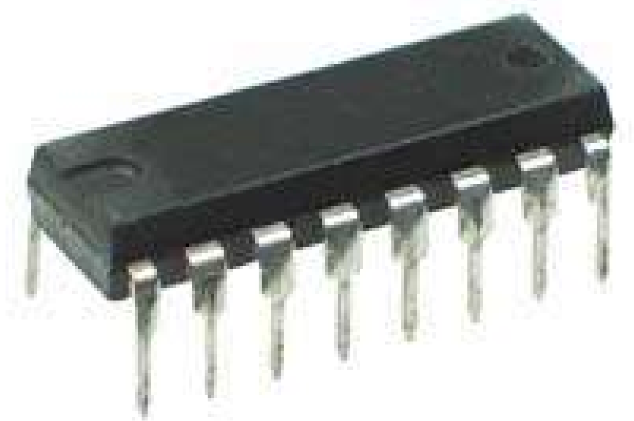
7476 (JK Flip-Flop)



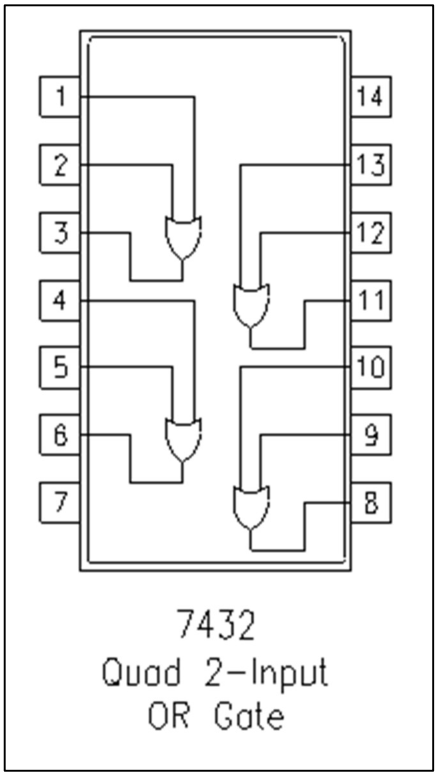


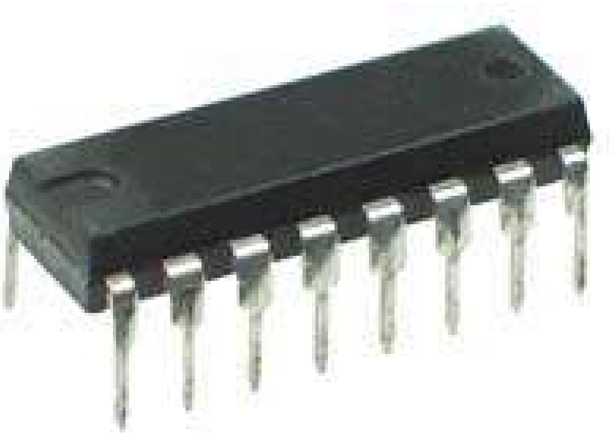
7486 (Quad XOR Gate)



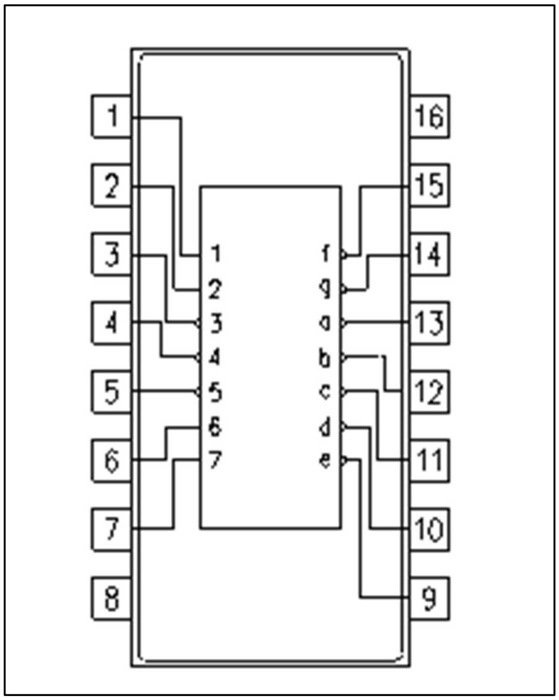


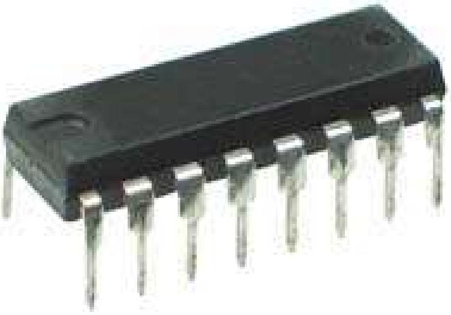
7432 (Quad 2-Input OR Gate)

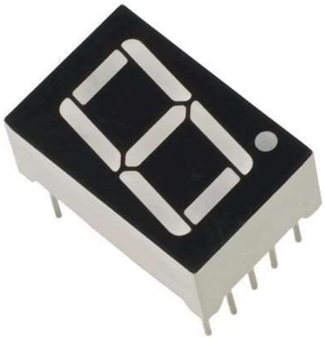




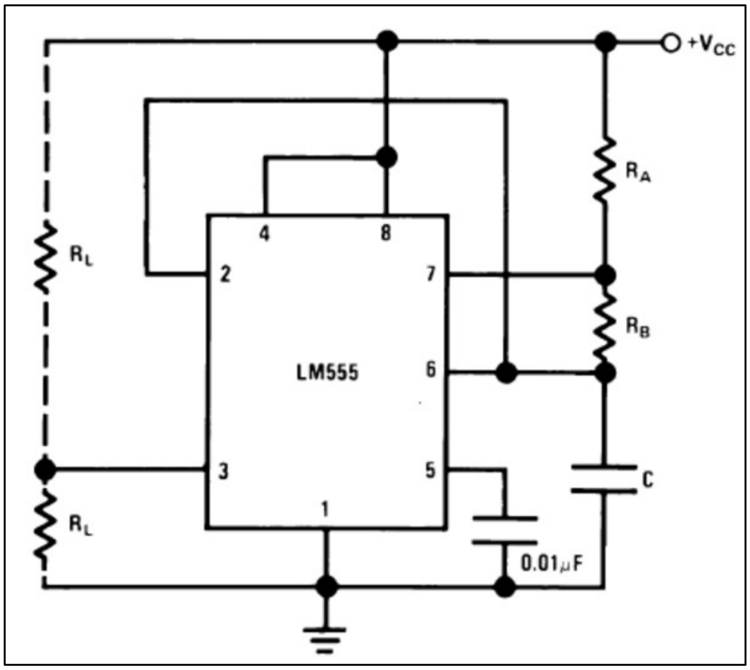
7448 (BCD To 7-Segment Decoder “Common Cathode”)

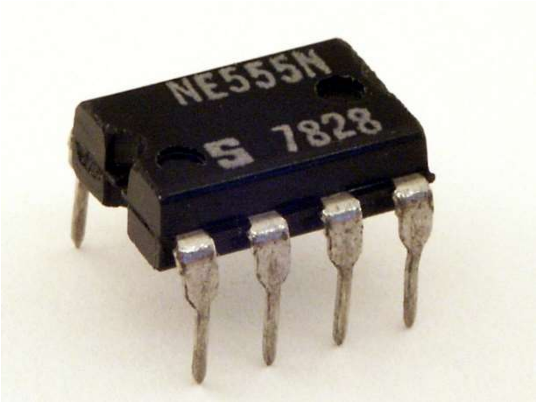


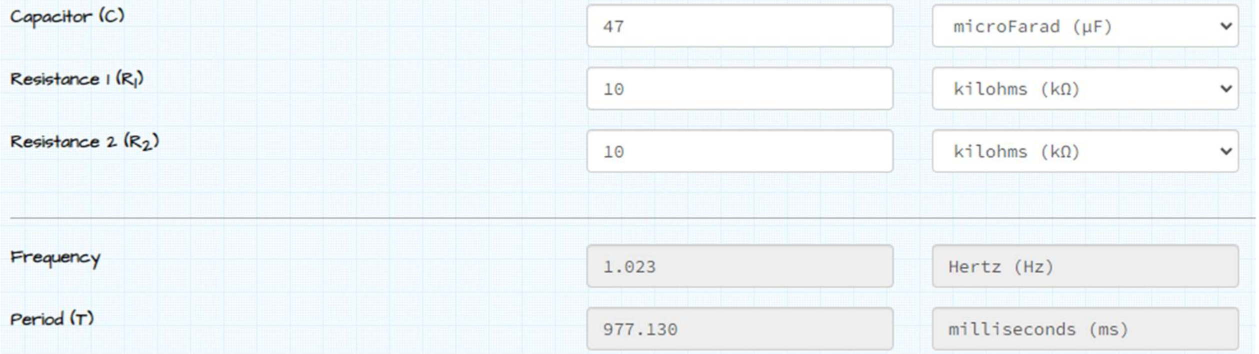




555 Timer







Capacitor (C)	47	microFarad (μF)
Resistance 1 (R_1)	10	kilohms (kΩ)
Resistance 2 (R_2)	10	kilohms (kΩ)
Frequency	1.023	Hertz (Hz)
Period (T)	977.130	milliseconds (ms)

We used **555 Calculator** to calculate the Capacitance & Resistance needed to implement a **1 Hz Clock** using 555 Timer.

Link: https://ohmslawcalculator.com/555-astable-calculator

N.B. Videos & Logisim Circuits Are Attached With The Report In The Zip File