EC-204

<LAB - 3>

NITK SURATHKAL

INBASEKARAN.P

201EC226

Prof: Sumam S

1. Design a circuit to convert 4 bits Binary to Gray and 4 bits Gray to Binary Code using XOR gates

Solution:

Binary to Gray

TURTH TABLE

B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

KMAP

Output: G3

Format: Sum of products

B3, B2 \ B1, B0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

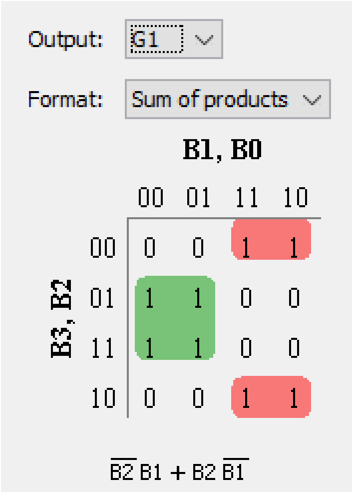
B3

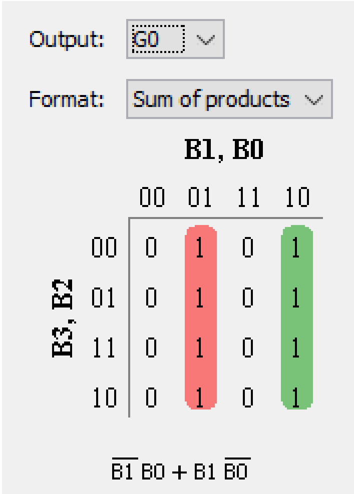
Output: G2

Format: Sum of products

B3, B2 \ B1, B0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

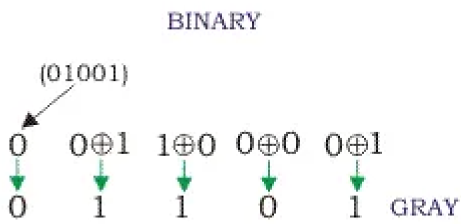
$\overline{B3}\,B2 + B3\,\overline{B2}$



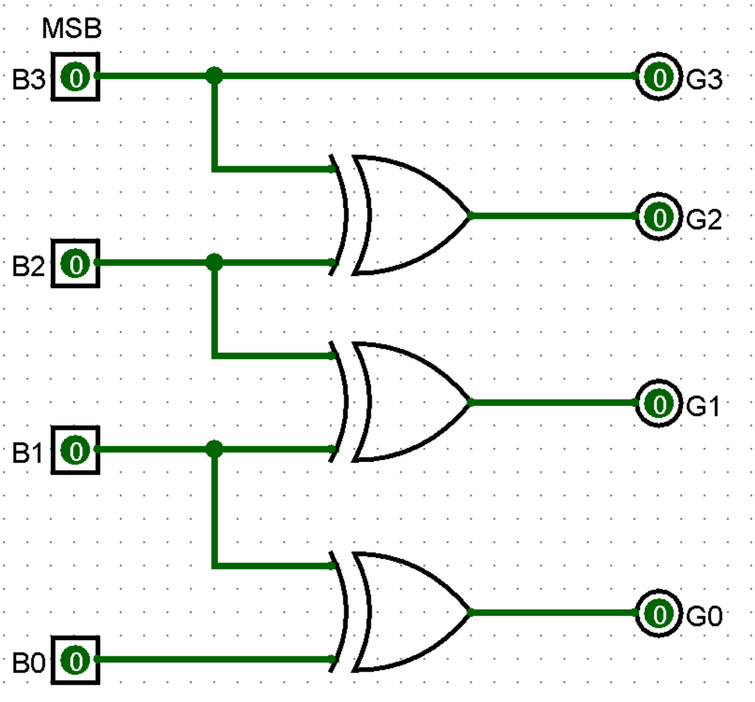


<u>EXPLAIN</u>

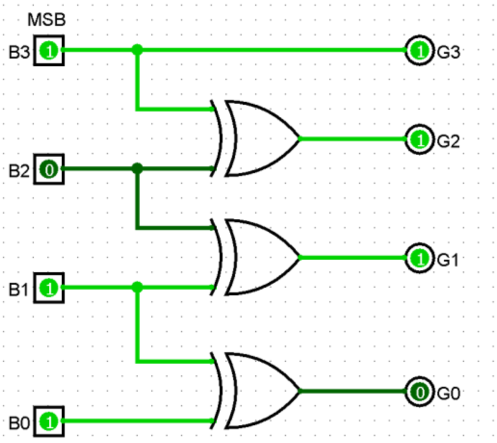
The Binary to Gray code the MSB remains same and the other bits can be calculated by XOR with the adjacent bits.

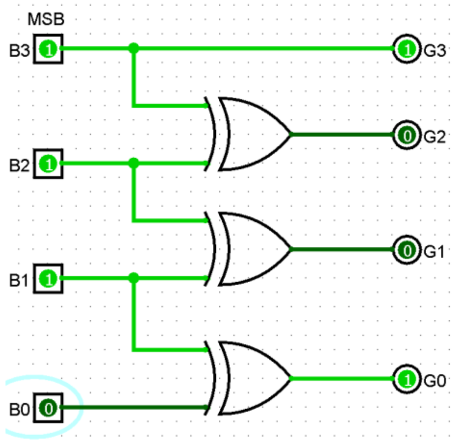


<u>CIRCUIT</u>



RESULT OBTAINED



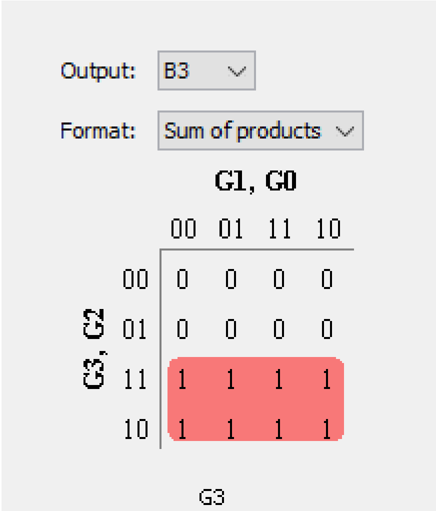


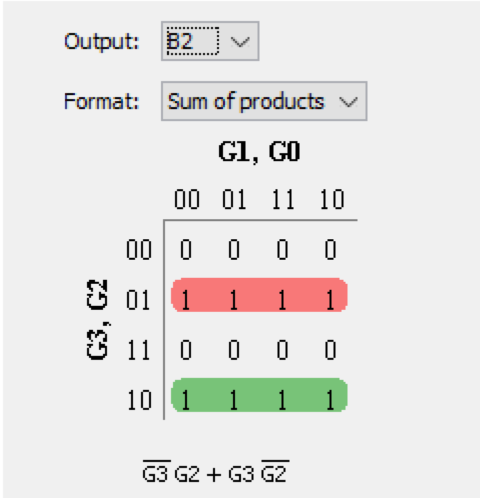
Gray to Binary Code

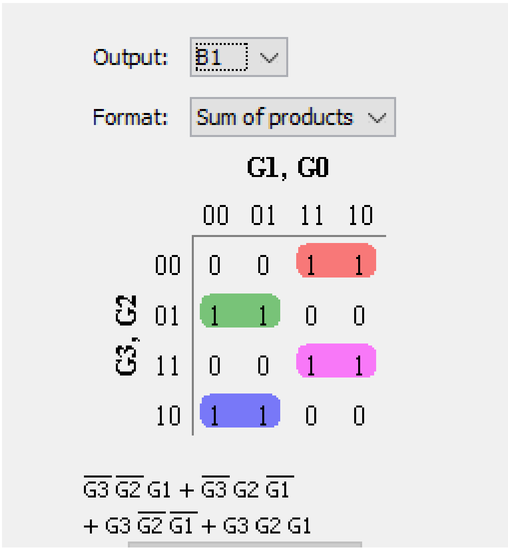
TURTH TABLE

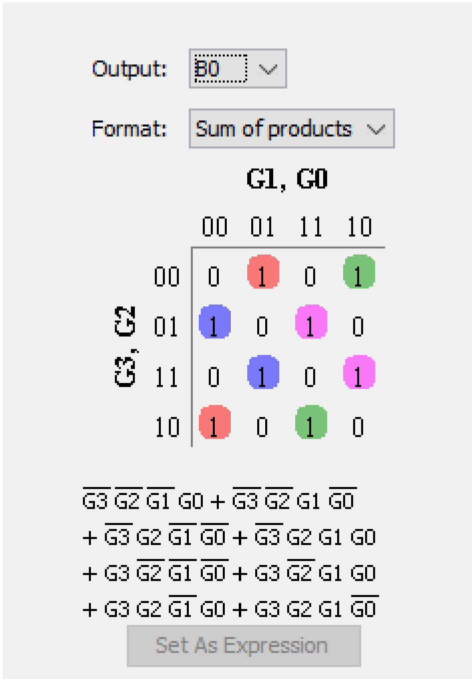
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

KMAP



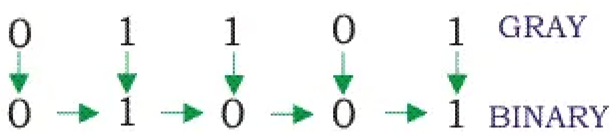




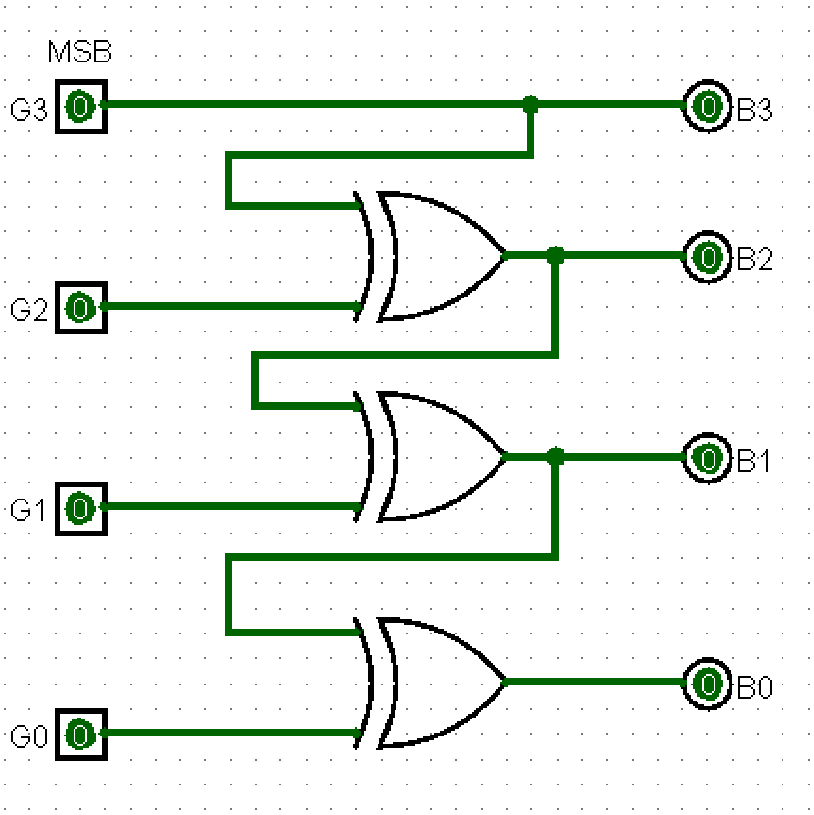


EXPLAIN

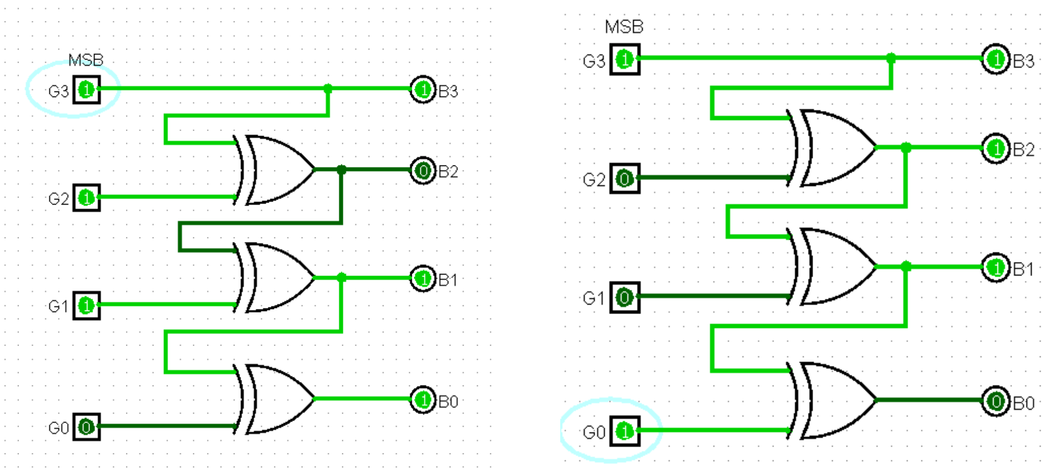
The Gray code to Binary the MSB remains same and the other bits can be calculated by XOR as follows



CIRCUIT



RESULT OBTAINED



2. Design an Excess-3 to BCD code converter and implement using 8:1 multiplexer. Use the multiplexer available in Plexers library in logisim.

Solution:

TURTH TABLE

X3	X2	X1	X0	Y3	Y2	Y1	Y0
0	0	0	0	x	x	x	x
0	0	0	1	x	x	x	x
0	0	1	0	x	x	x	x
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	1	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	0	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

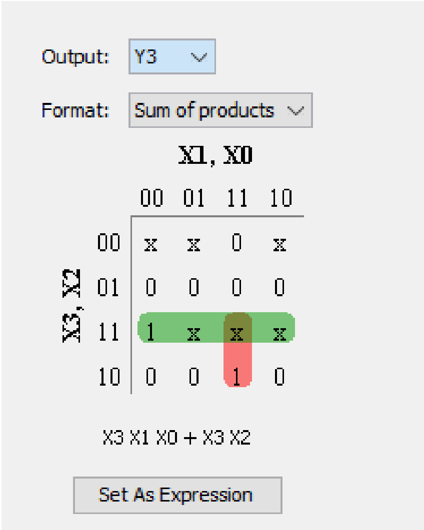
EXPLAIN

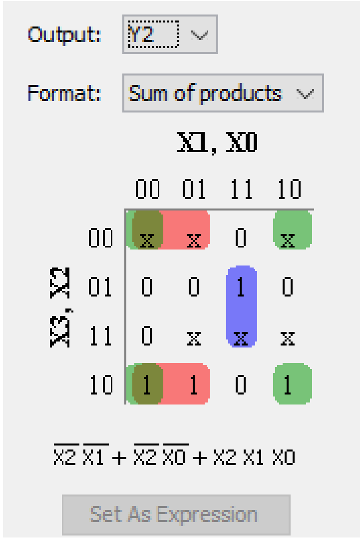
Excess-3 binary code is an unweighted self-complementary BCD code.

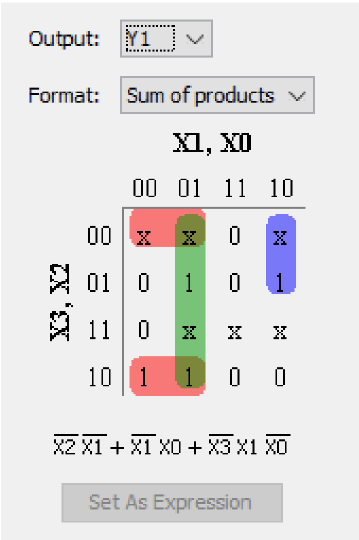
Self-Complementary property means that the 1's complement of an excess-3 number is the excess-3 code of the 9's complement of the corresponding decimal number. This property is useful since a decimal number can be nines' complemented (for subtraction) as easily as a binary number can be ones' complemented; just by inverting all bits.

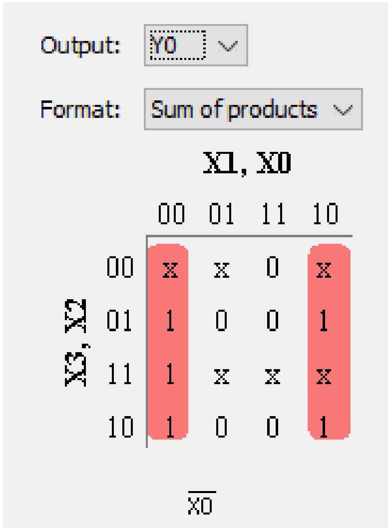
For example, the excess-3 code for 3(0011) is 0110 and to find the excess-3 code of the complement of 3, we just need to find the 1's complement of 0110 -> 1001, which is also the excess-3 code for the 9's complement of 3 -> (9-3) = 6.

KMAP

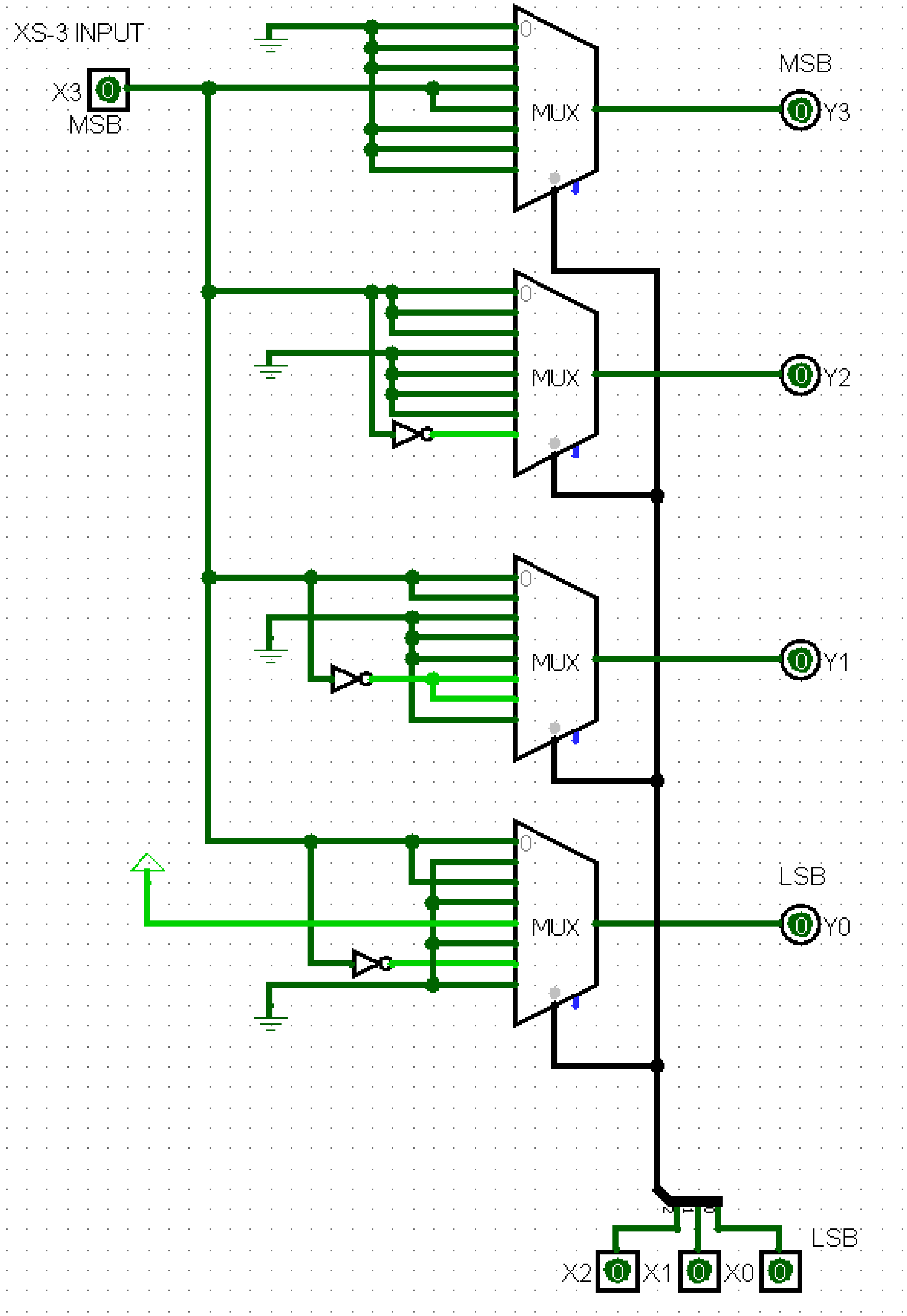

Output: Y3
Format: Sum of products
X1, X0
00 01 11 10
X3, X2
00 x x 0 x
01 0 0 0 0
11 1 x x x
10 0 0 1 0
X3 X1 X0 + X3 X2
Set As Expression


Output: Y2
Format: Sum of products
X1, X0
00 01 11 10
X3, X2
00 x x 0 x
01 0 0 1 0
11 0 x x x
10 1 1 0 1
$\overline{X2}\,\overline{X1} + \overline{X2}\,\overline{X0} + X2\,X1\,X0$
Set As Expression

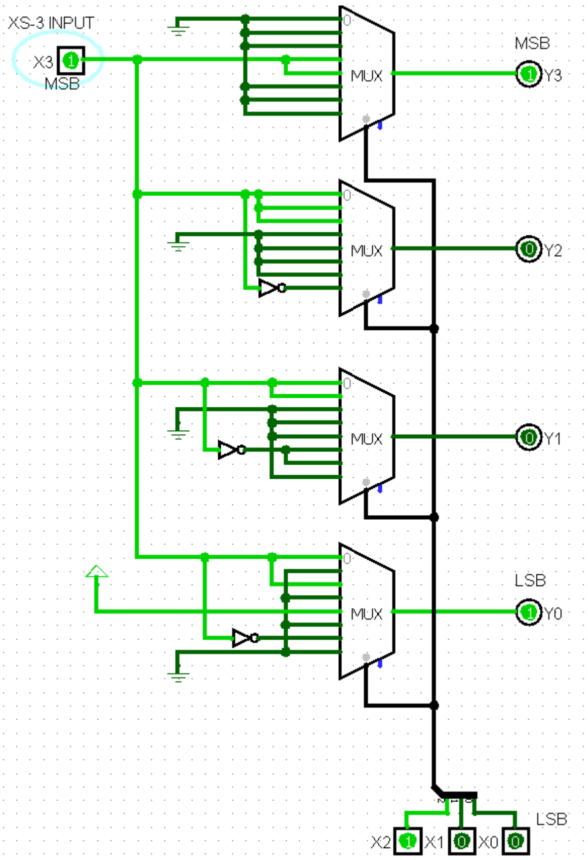

Output: Y1
Format: Sum of products
X1, X0
00 01 11 10
X3, X2
00 x x 0 x
01 0 1 0 1
11 0 x x x
10 1 1 0 0
$\overline{X2}\,\overline{X1} + \overline{X1}\,X0 + \overline{X3}\,X1\,\overline{X0}$
Set As Expression

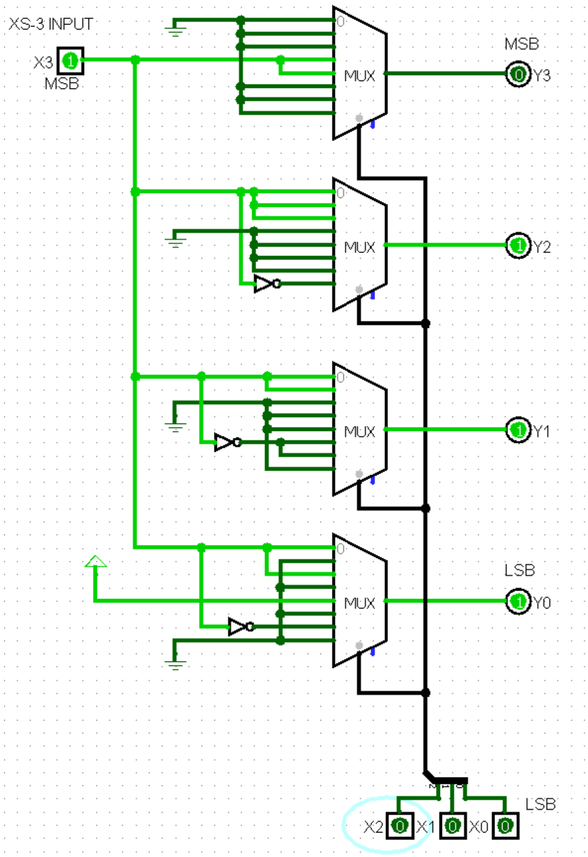

Output: Y0
Format: Sum of products
X1, X0
00 01 11 10
X3, X2
00 x x 0 x
01 1 0 0 1
11 1 x x x
10 1 0 0 1
$\overline{X0}$

CIRCUIT



RESULT OBTAINED





3. Design a circuit to implement the following functions using (a) 4 to 16 decoder (b) 4 to 1 Multiplexer. Use the Multiplexer and decoder available in Plexers library in logisim. XOR gates

$$F_1(A,B,C,D) = \Sigma m\ (11,12,13,14,15)$$

$$F_2(A,B,C,D) = \Sigma m\ (3,7,11,12,13,15)$$

$$F_3(A,B,C,D) = \Sigma m\ (3,7,12,13,14,15)$$

Solution:

4 to 16 decoder

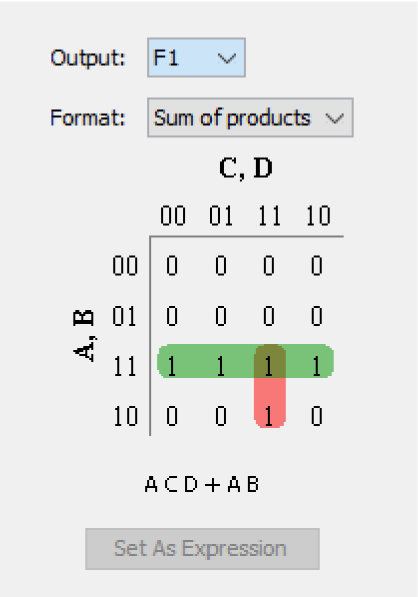
TURTH TABLE

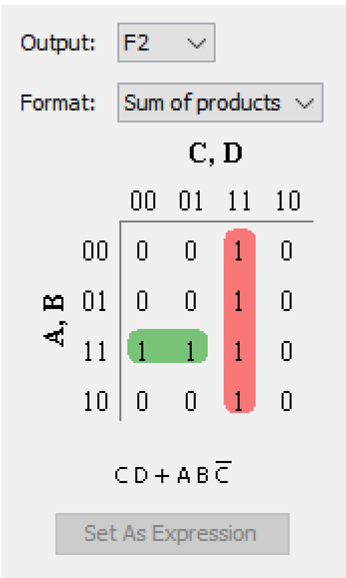
A	B	C	D	F1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

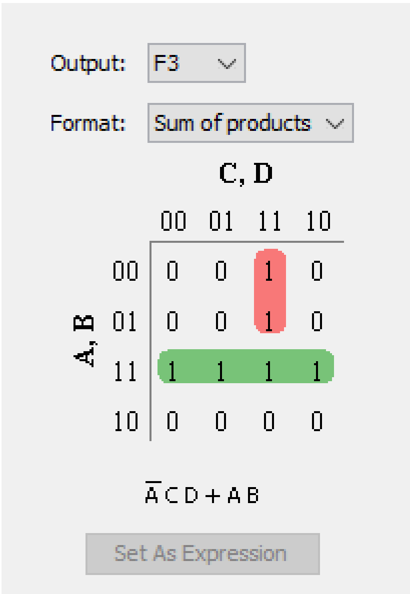
A	B	C	D	F2
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

A	B	C	D	F3
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

KMAP

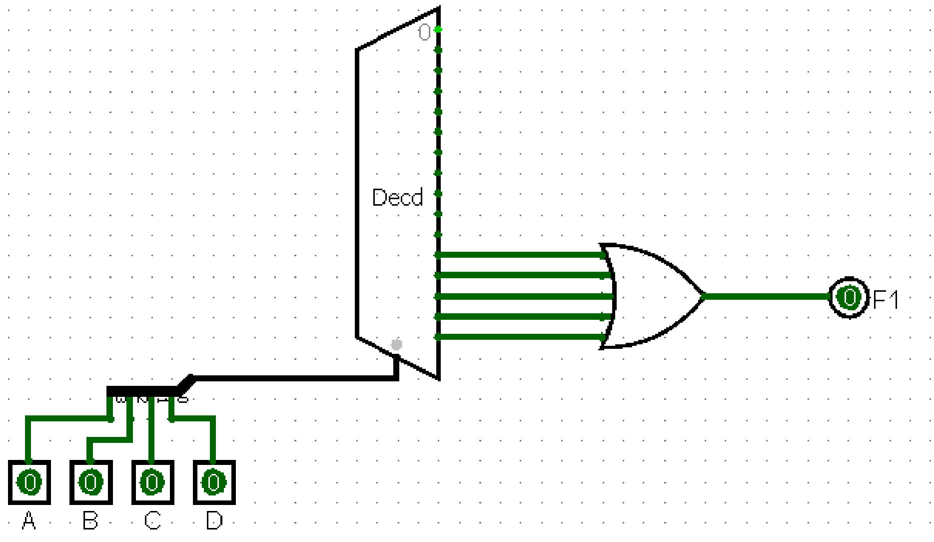


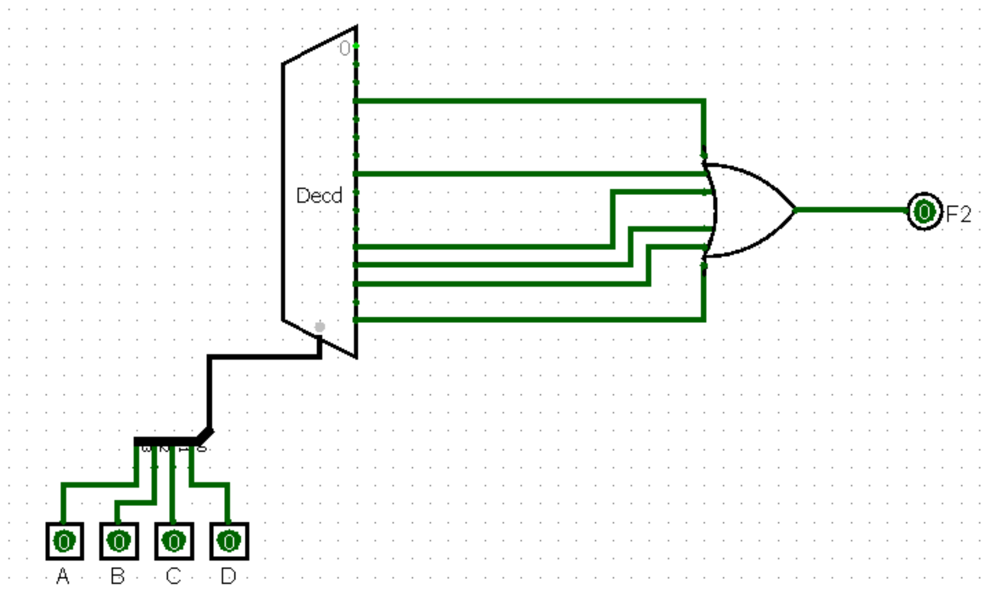


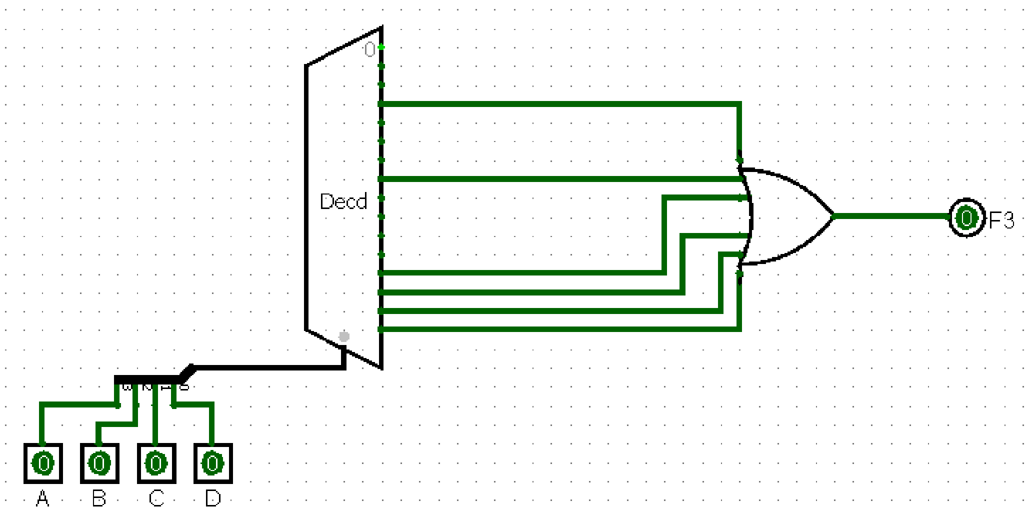


EXPLAIN

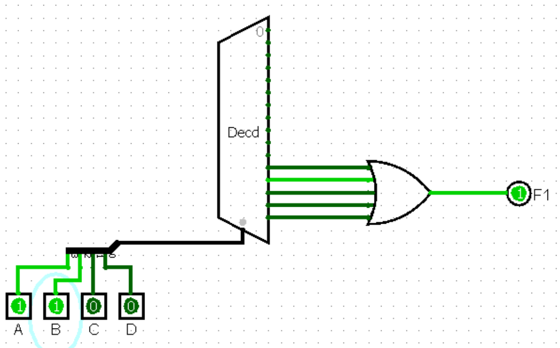
CIRCUIT

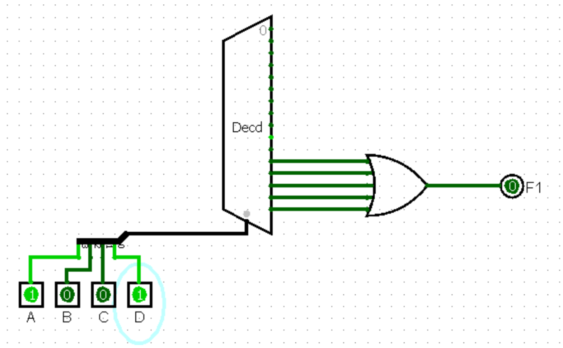


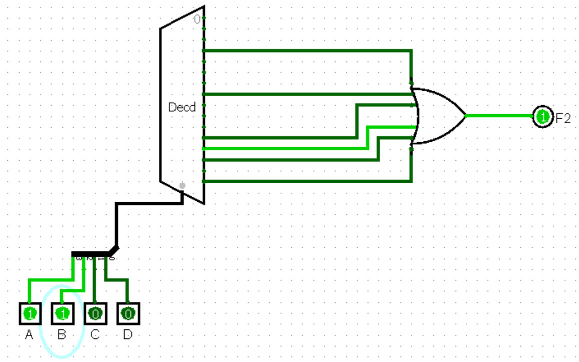


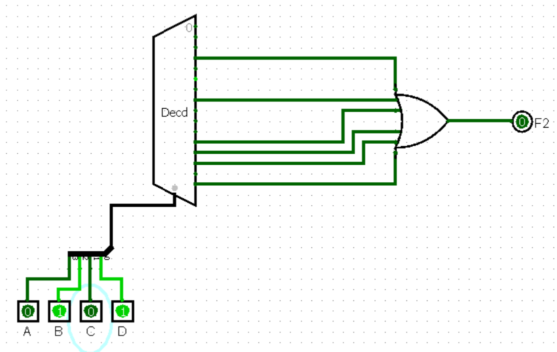


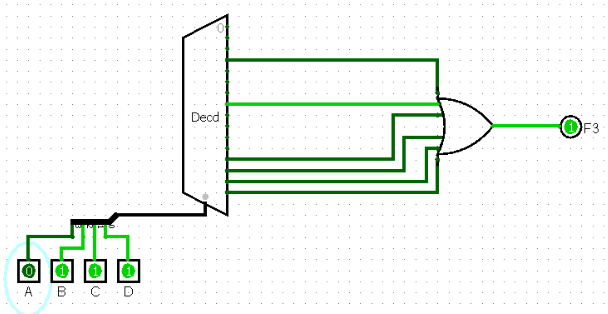
RESULT OBTAINED

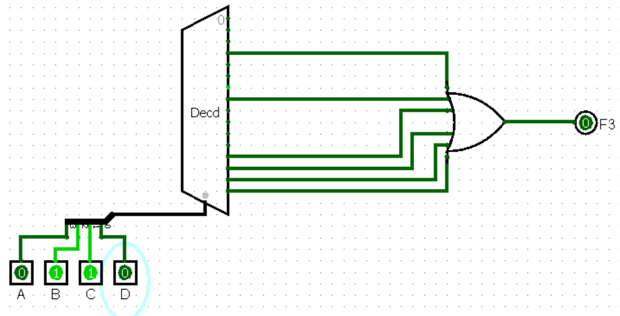












4 to 1 Multiplexer

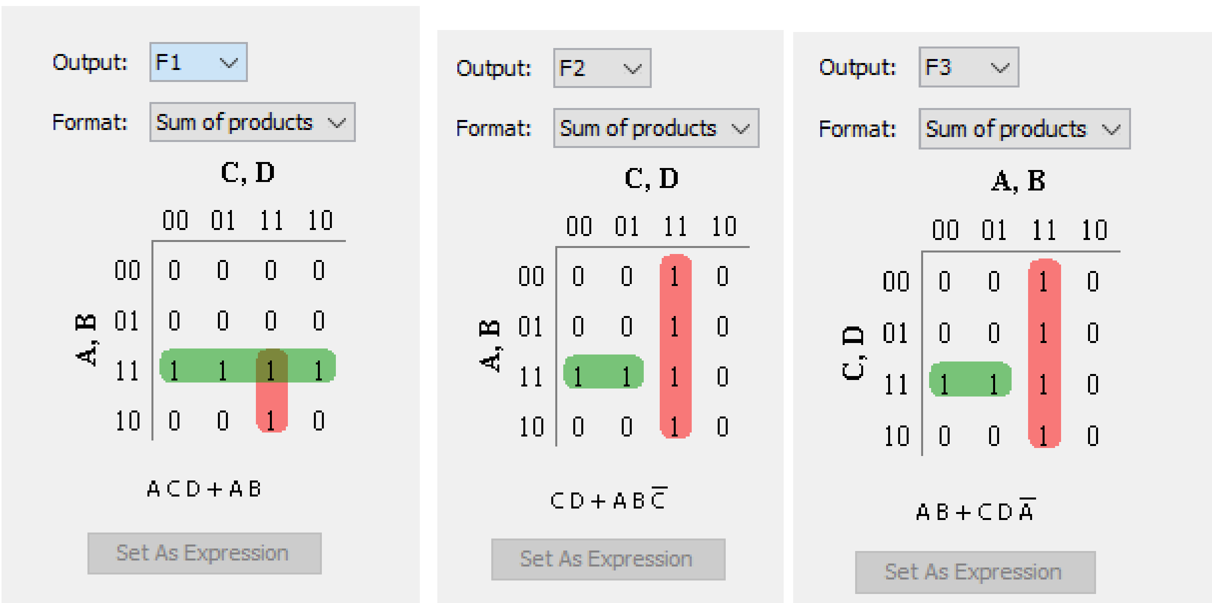
TURTH TABLE

A	B	C	D	F1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

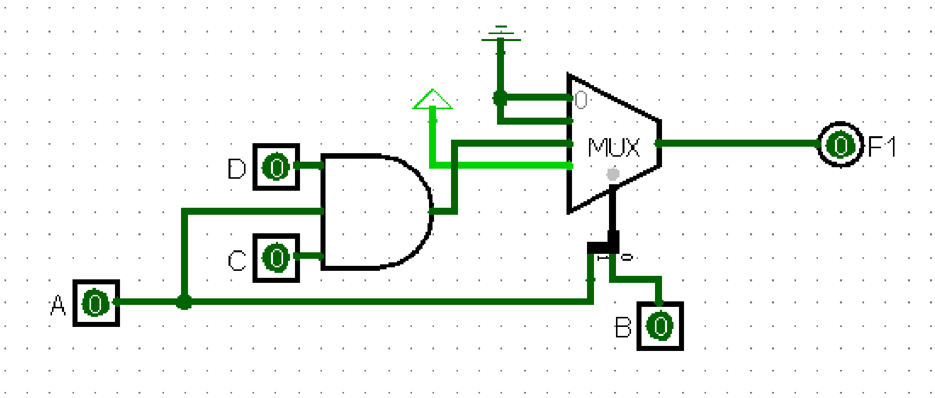
A	B	C	D	F2
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

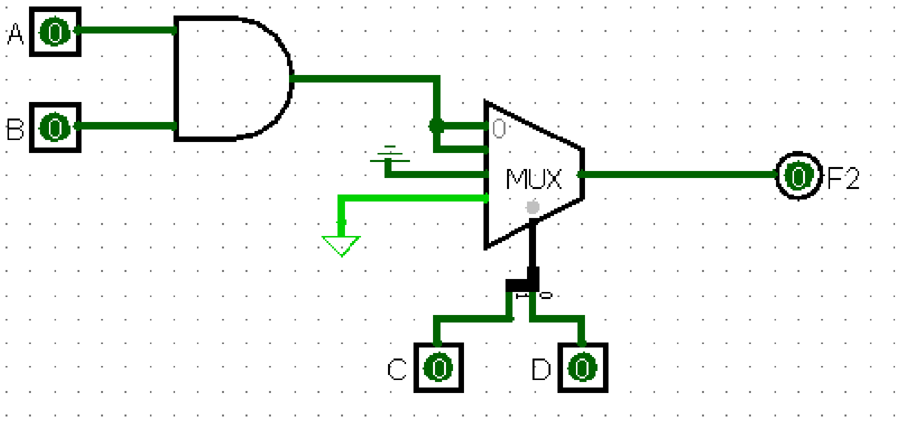
A	B	C	D	F3
0	0	0	0	0
0	0	0	1	0
0	0	1	0	$\boxed{0}$
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

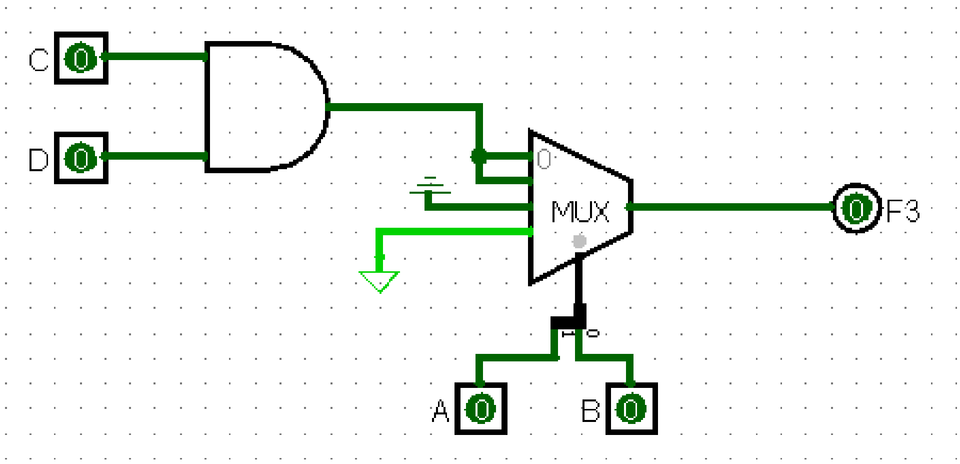
KMAP



CIRCUIT







4. Design a circuit to implement the following function using 16 to 1 Multiplexer. . Use the Multiplexer available in Plexers library in logisim.Solution:

F = A'B'C'D'(R'+S) + AB'C'D' + A'B'C'D + ABC'DQR + AB'C'D + A'B'CD (Q'+T) + ABCDS'T + AB'CD + A'BCD

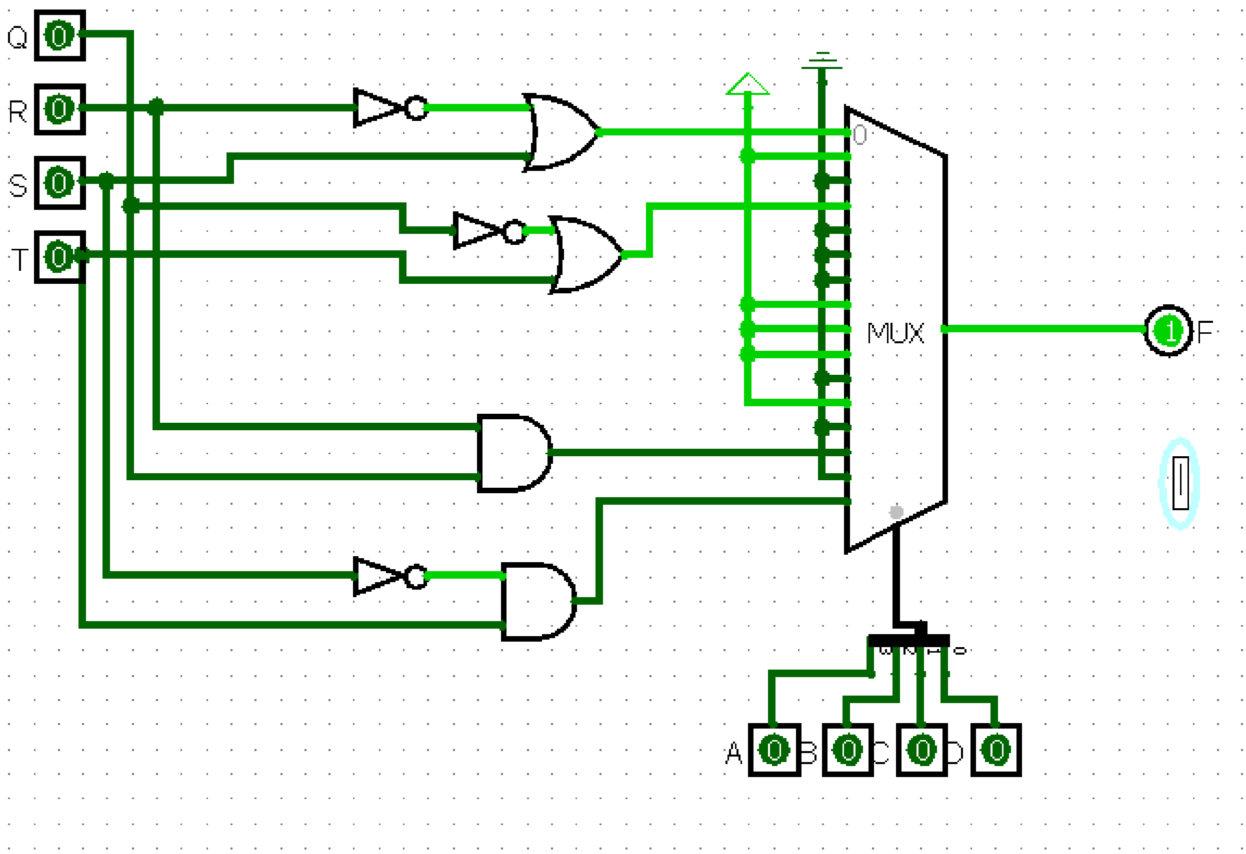
Solution:

EXPLAIN

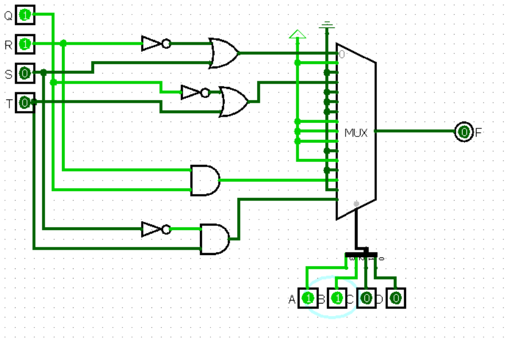
The selector lines can be used as ABCD respectively and the corresponding expression can be found by simply mapping it with the terms given in the question.

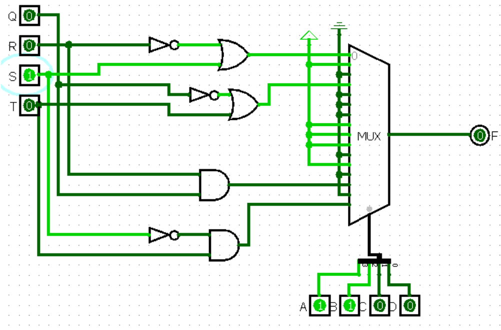
For example ~A~B~C~D is the first input line so we can give (~R + S) as the input and that will result in the following term.

CIRCUIT



RESULT OBTAINED





5. Implement (a) 4 bit controllable adder/subtractor (b) 8 bit controllable adder/subtractor.

   Use the Adder available in Arithmetic library in logisim.

Solution:

4 bit controllable adder/subtractor

TURTH TABLE

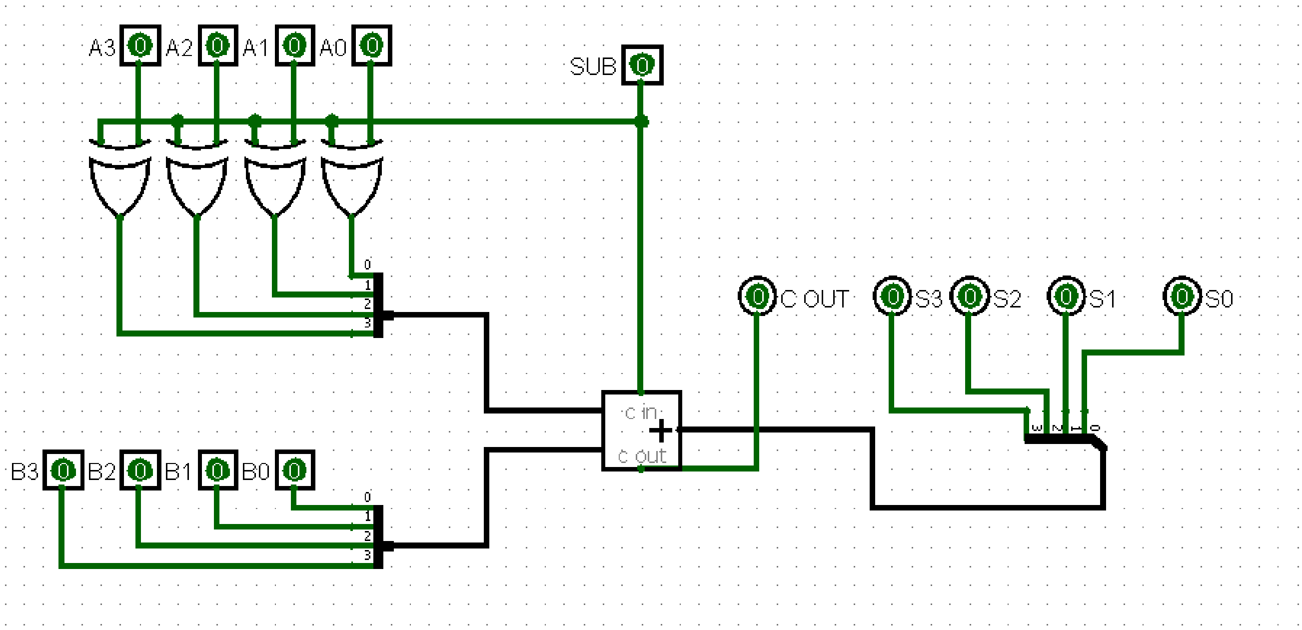
A3	A2	A1	A0	SUB	B3	B2	B1	B0	COUT	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	1	1	0	0	0	1	1
0	0	0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	1	0	0	1	0	1
0	0	0	0	0	0	1	1	0	0	0	1	1	0
0	0	0	0	0	0	1	1	1	0	0	1	1	1
0	0	0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	1	0	1	0	0	1
0	0	0	0	0	1	0	1	0	0	1	0	1	0
0	0	0	0	0	1	0	1	1	0	1	0	1	1
0	0	0	0	0	1	1	0	0	0	1	1	0	0
0	0	0	0	0	1	1	0	1	0	1	1	0	1
0	0	0	0	0	1	1	1	0	0	1	1	1	0
0	0	0	0	0	1	1	1	1	0	1	1	1	1
0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	1	1	0	0	0	1
0	0	0	0	1	0	0	1	0	1	0	0	1	0
0	0	0	0	1	0	0	1	1	1	0	0	1	1
0	0	0	0	1	0	1	0	0	1	0	1	0	0
0	0	0	0	1	0	1	0	1	1	0	1	0	1
0	0	0	0	1	0	1	1	0	1	0	1	1	0
0	0	0	0	1	0	1	1	1	1	0	1	1	1
0	0	0	0	1	1	0	0	0	1	1	0	0	0
0	0	0	0	1	1	0	0	1	1	1	0	0	1
0	0	0	0	1	1	0	1	0	1	1	0	1	0
0	0	0	0	1	1	0	1	1	1	1	0	1	1

… 256 VALUES

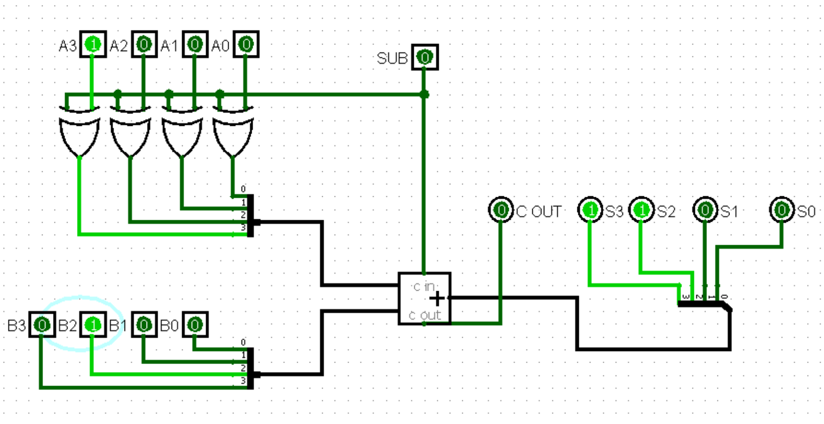
EXPLAIN

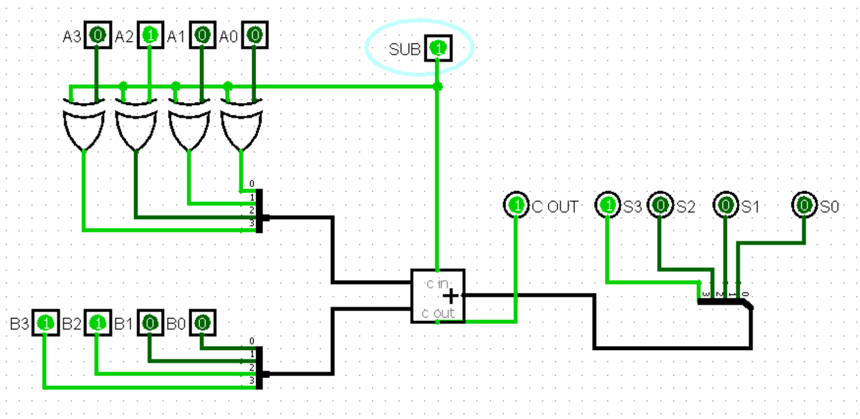
Subtraction can be done using 2s complement. Which is nothing but 1s compliment + 1. For subtraction we pass the cin as 1 and we take 1 XOR with the input bits which is to be subtracted. This results in 1’s complement to this the adder adds 1 (cin) which results in 2’s compliment.

CIRCUIT



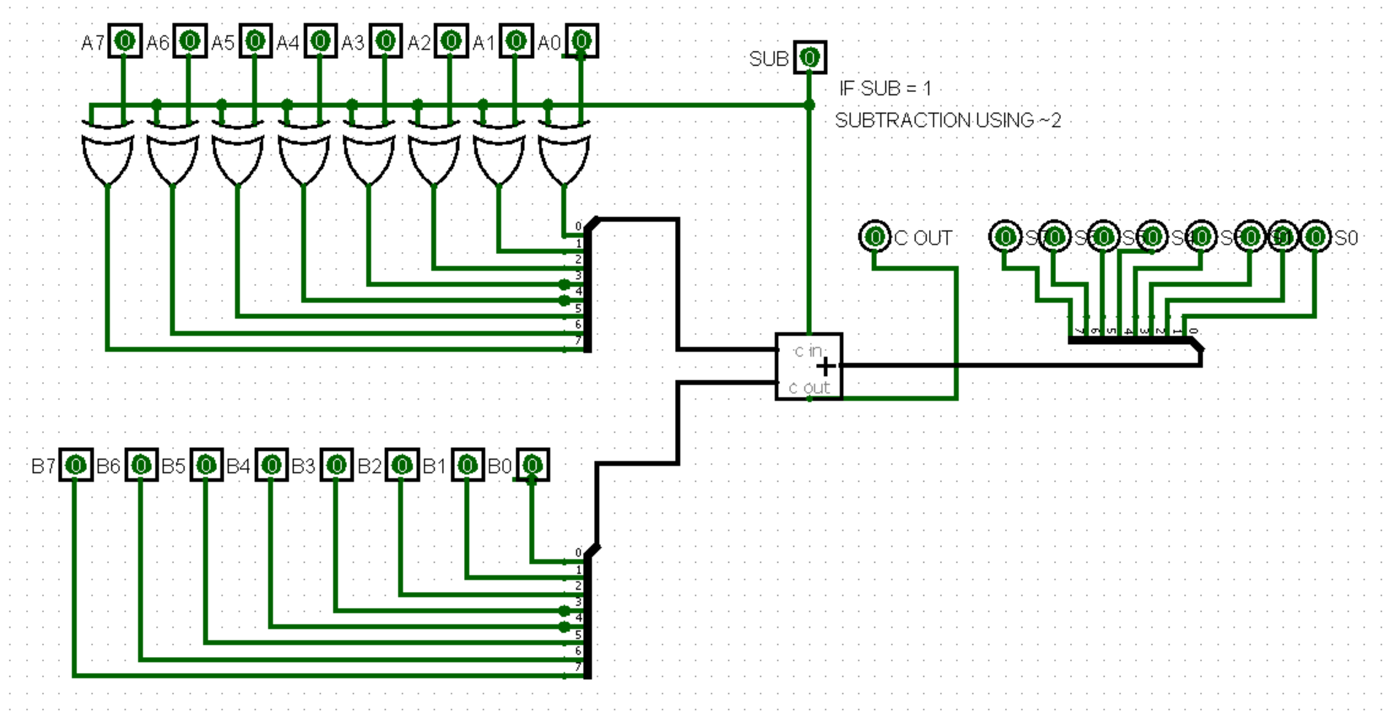
RESULT OBTAINED



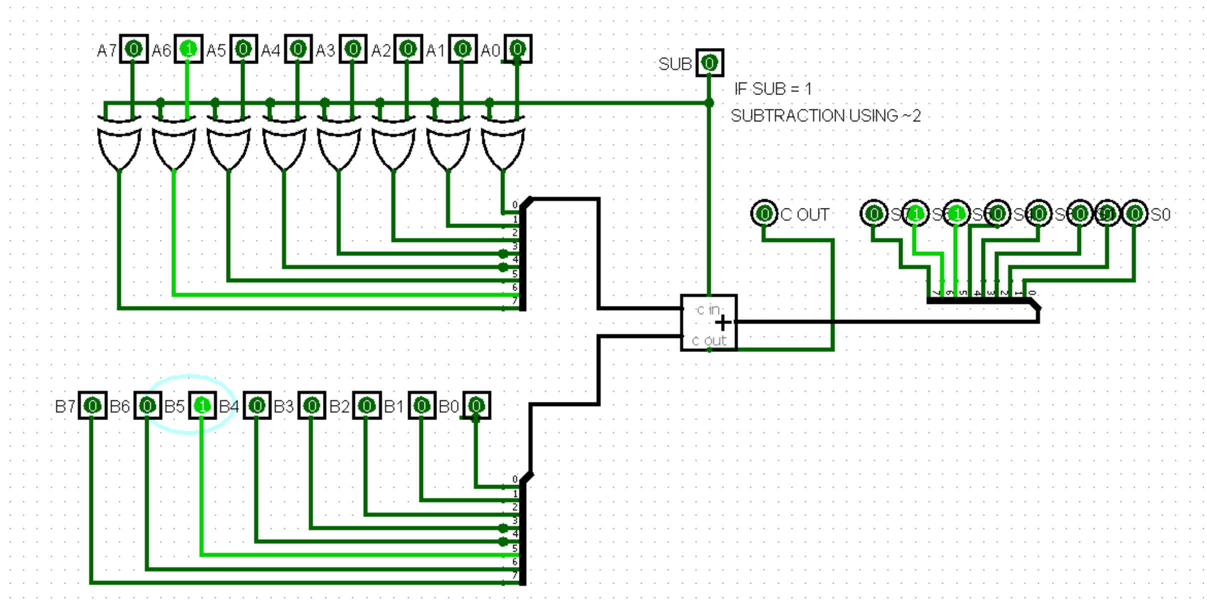


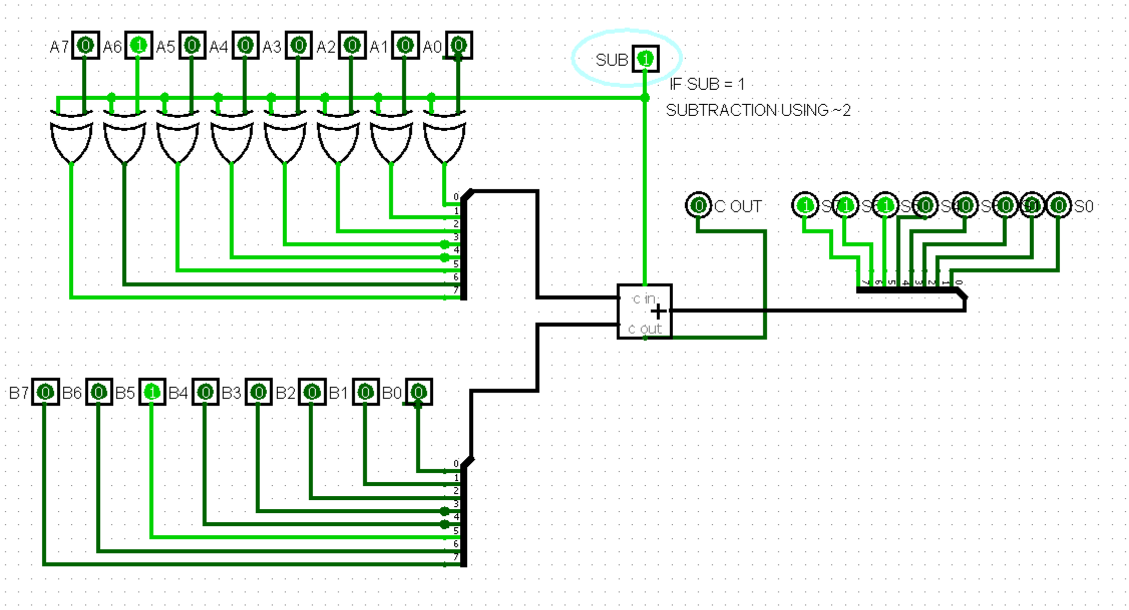
8 bit controllable adder/subtractor

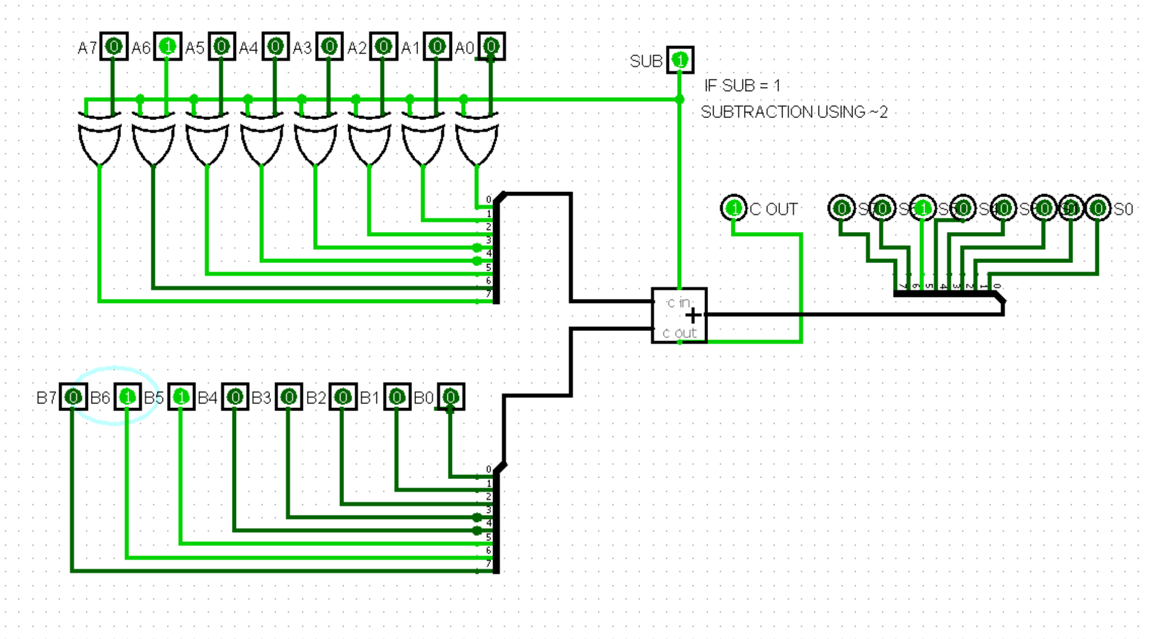
CIRCUIT



RESULT OBTAINED


A7 A6 A5 A4 A3 A2 A1 A0
SUB
IF SUB = 1
SUBTRACTION USING ~2
C OUT
S0
c in
c out
B7 B6 B5 B4 B3 B2 B1 B0


A7 A6 A5 A4 A3 A2 A1 A0
SUB
IF SUB = 1
SUBTRACTION USING ~2
C OUT
S0
c in
c out
B7 B6 B5 B4 B3 B2 B1 B0


A7 A6 A5 A4 A3 A2 A1 A0
SUB
IF SUB = 1
SUBTRACTION USING ~2
C OUT
S0
c in
c out
B7 B6 B5 B4 B3 B2 B1 B0