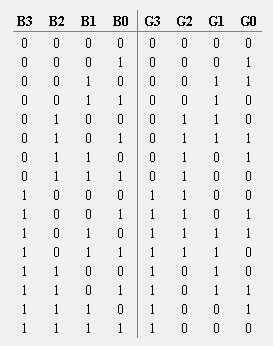


# Design a circuit to convert 4 bits Binary to Gray and 4 bits Gray to Binary Code using XOR gates

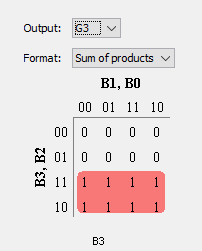
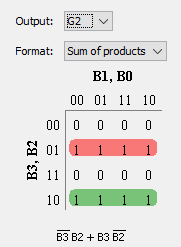
## Solution:

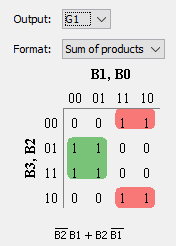
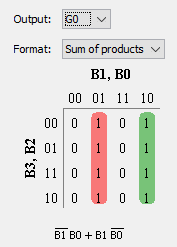
### Binary to Gray

#### TURTH TABLE



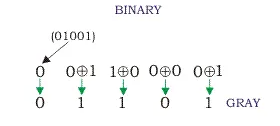
#### KMAP

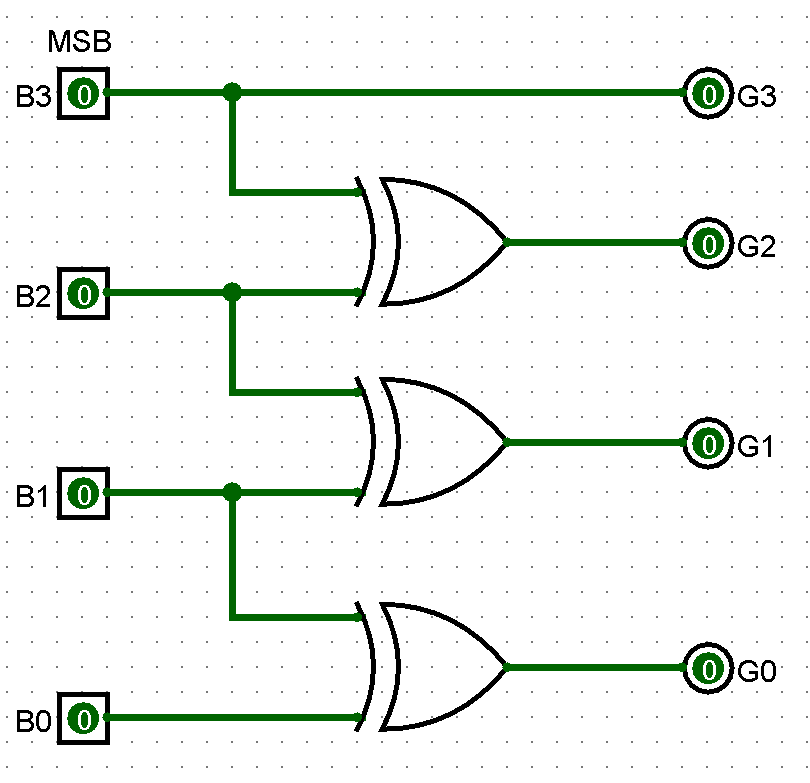
 

#### EXPLAIN

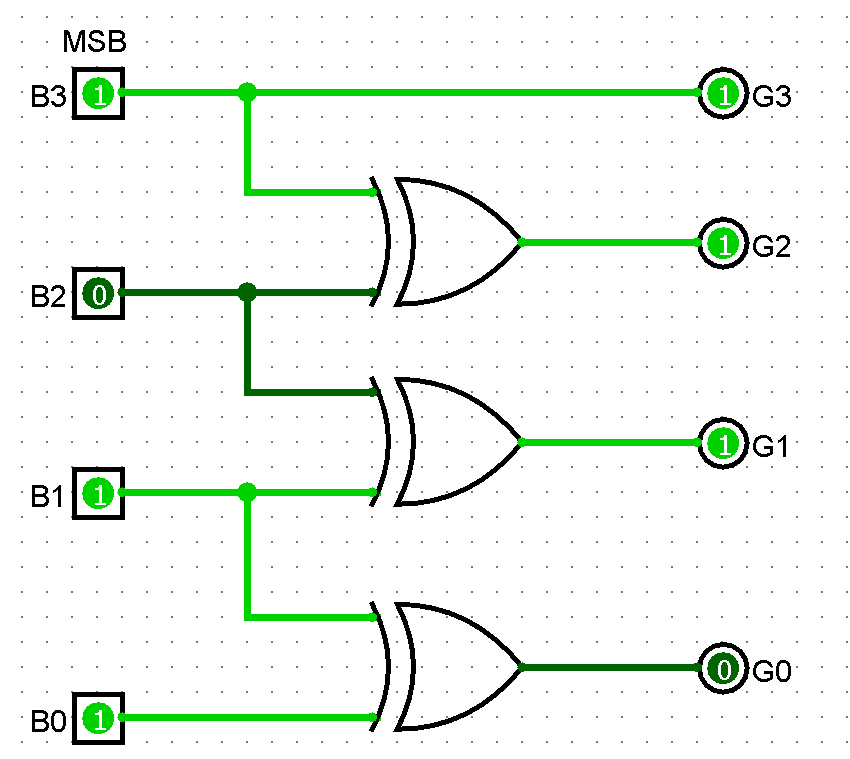
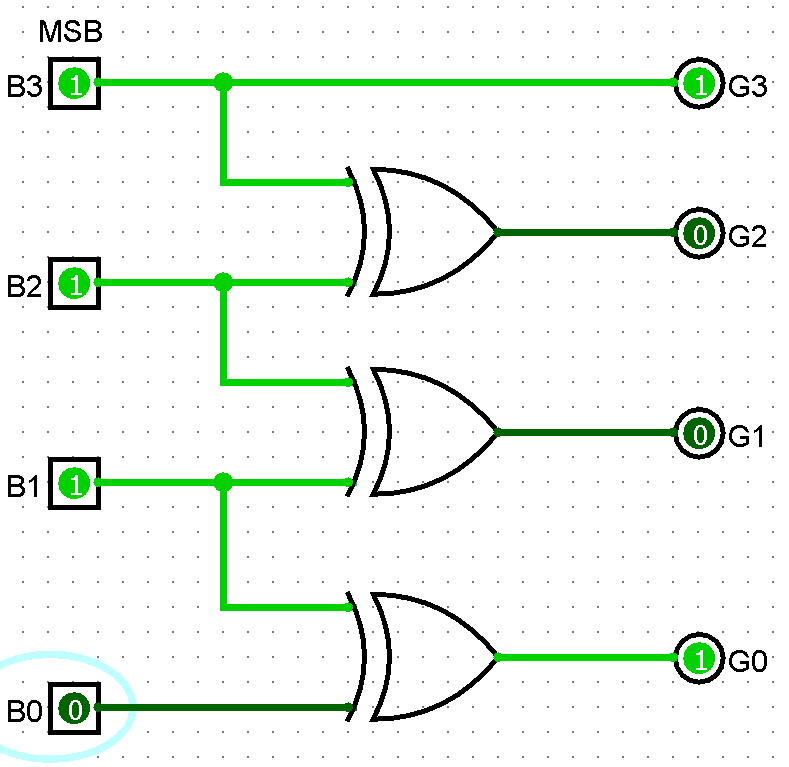
The Binary to Gray code the MSB remains same and the other bits can be calculated by XOR with the adjacent bits.



#### CIRCUIT

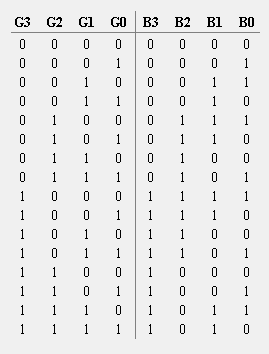


#### RESULT OBTAINED

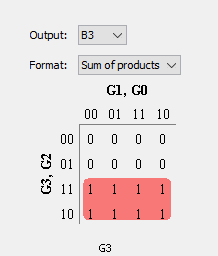
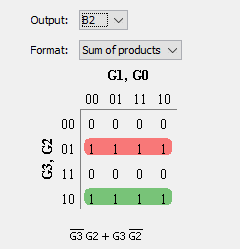
 

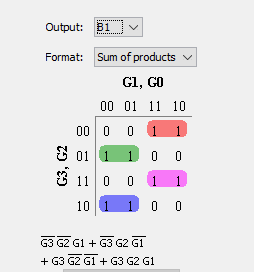
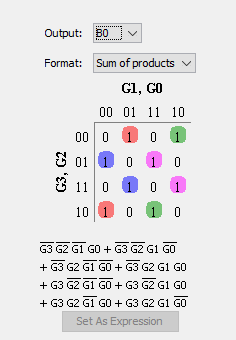
### Gray to Binary Code

#### TURTH TABLE



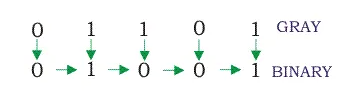
#### KMAP

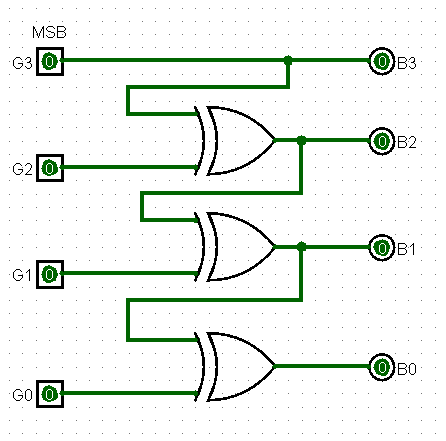
 

#### EXPLAIN

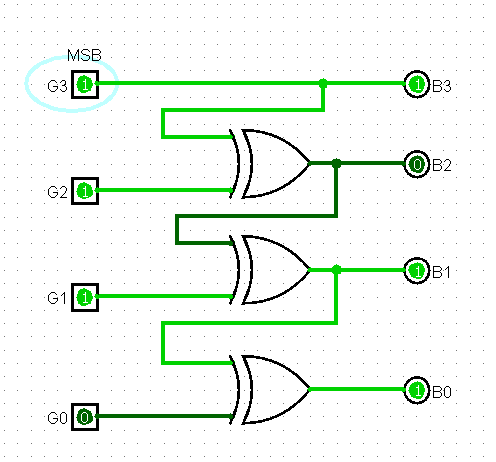
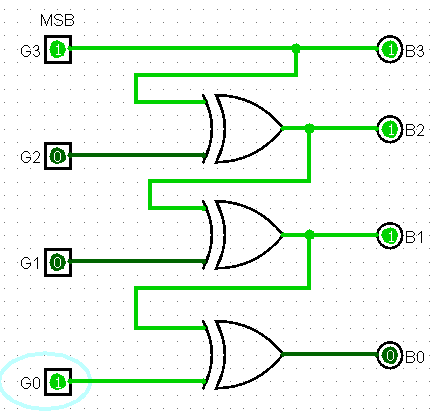
The Gray code to Binary the MSB remains same and the other bits can be calculated by XOR as follows



#### CIRCUIT



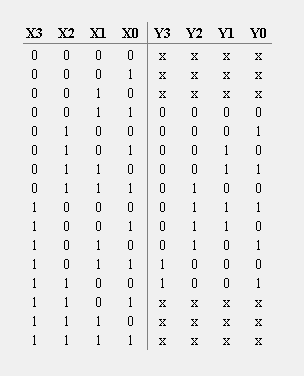
#### RESULT OBTAINED

# 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



#### 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.

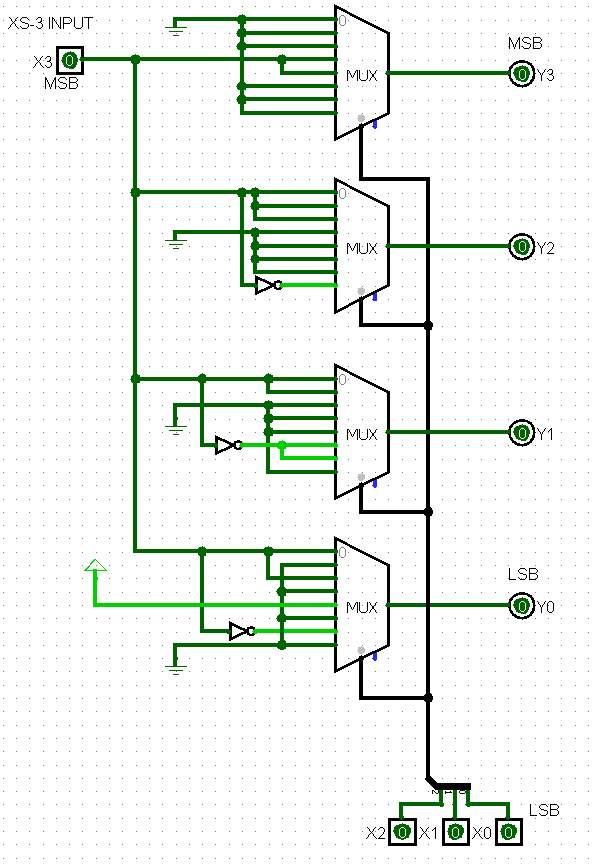
For example, the excess-3 code for 3(0011) is 0110 and to find the excess-3 code of the complement of 3, we just need to find the 1’s complement of 0110 -> 1001, which is also the excess-3 code for the 9’s complement of 3 -> (9-3) = 6.

KMAP

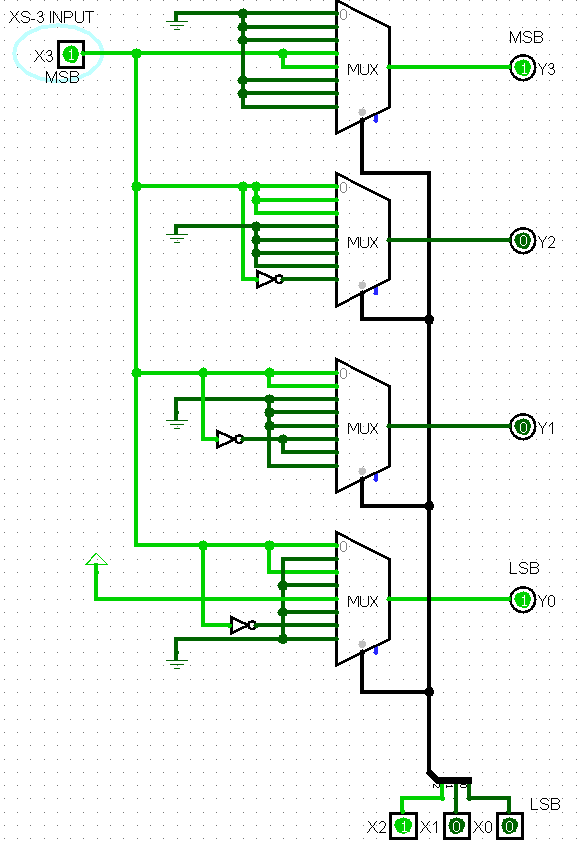
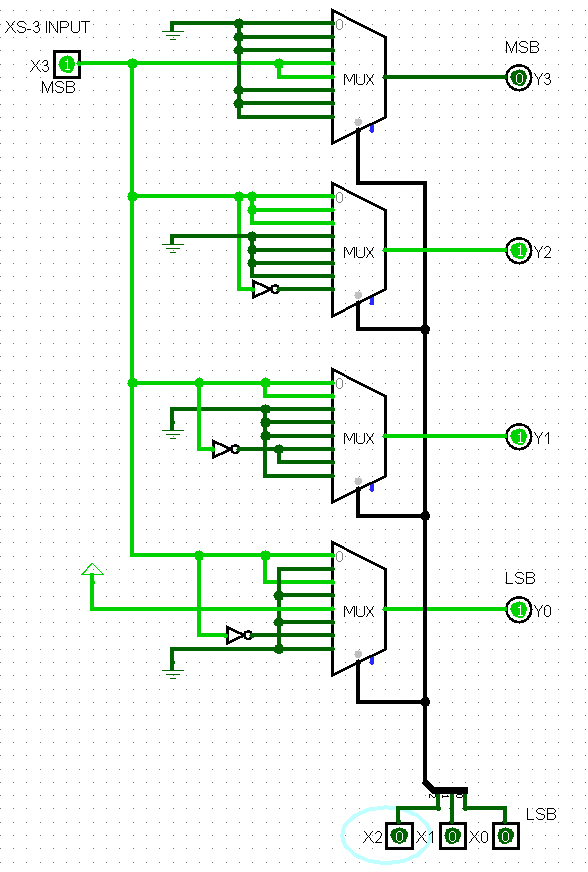
#### 

#### 

#### CIRCUIT



#### RESULT OBTAINED

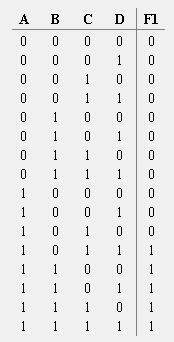
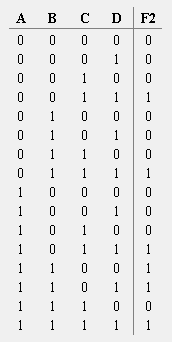
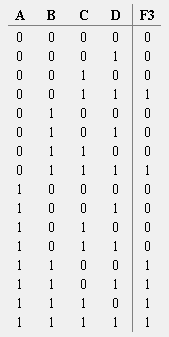
# Design a circuit to implement the following functions using (a) 4 to 16 decoder (b) 4 to 1 Multiplexer. Use the Multiplexer and decoder available in Plexers library in logisim. XOR gates

## 

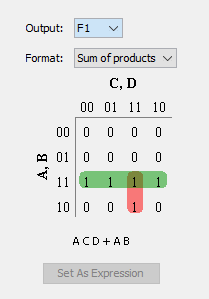
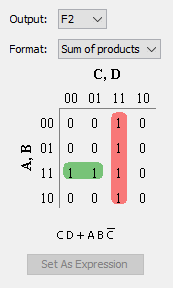
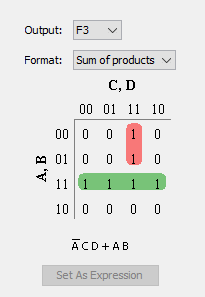
## Solution:

## 4 to 16 decoder

#### TURTH TABLE

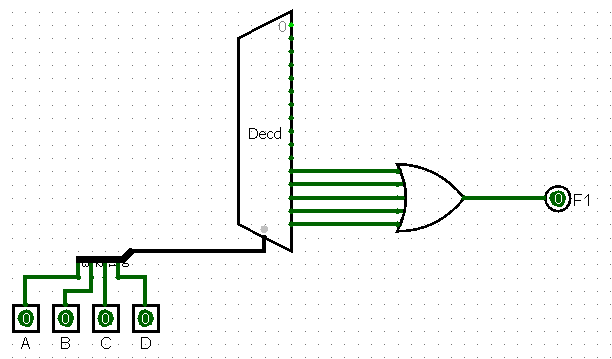
  

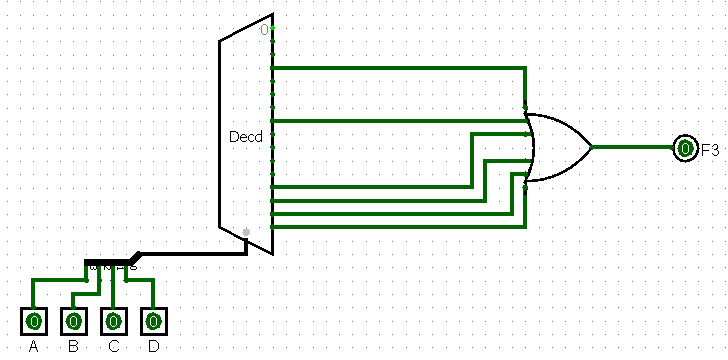
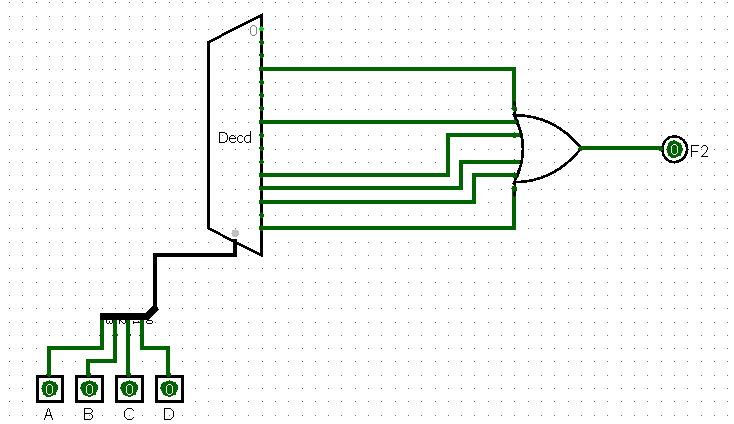
#### KMAP

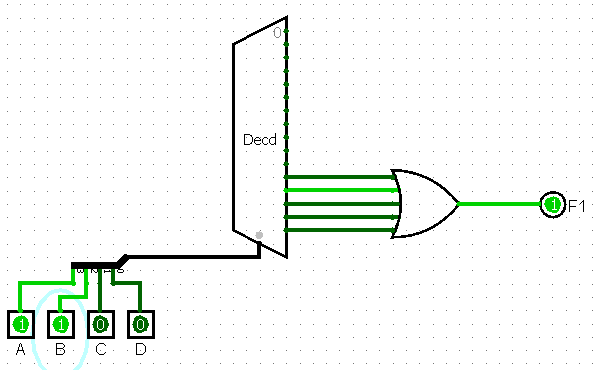
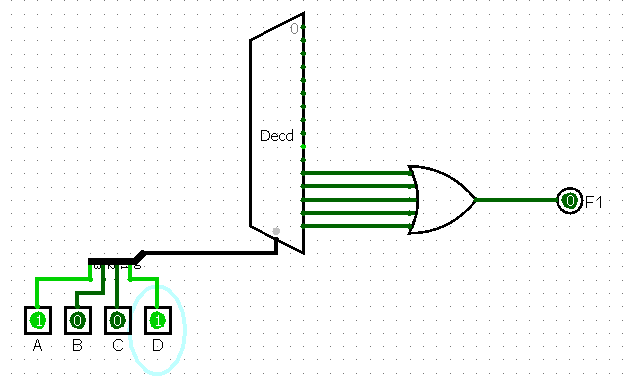
#### EXPLAIN

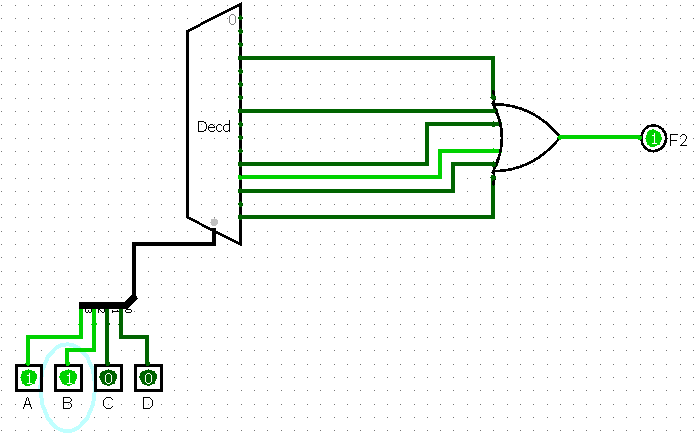
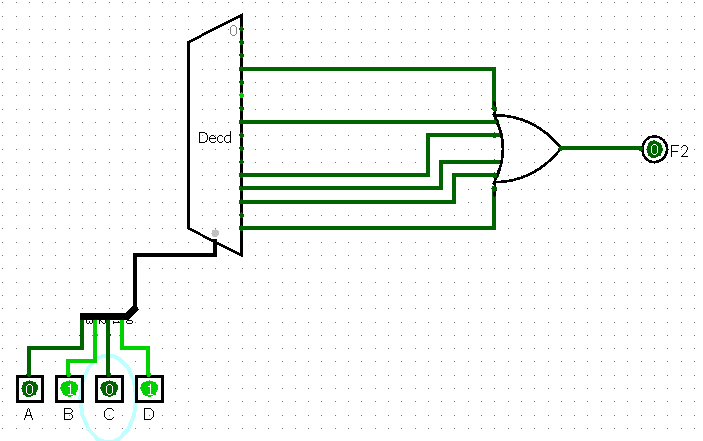
#### CIRCUIT

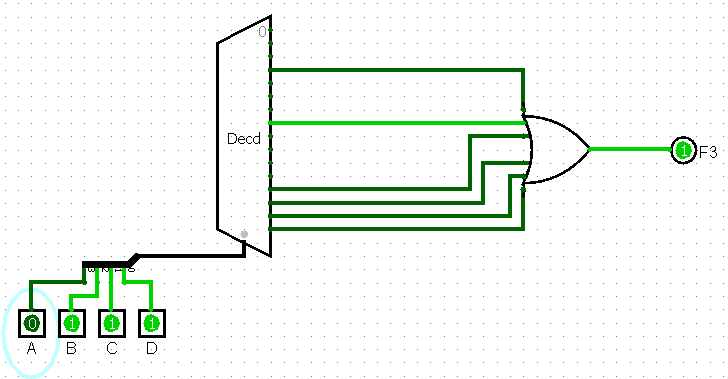
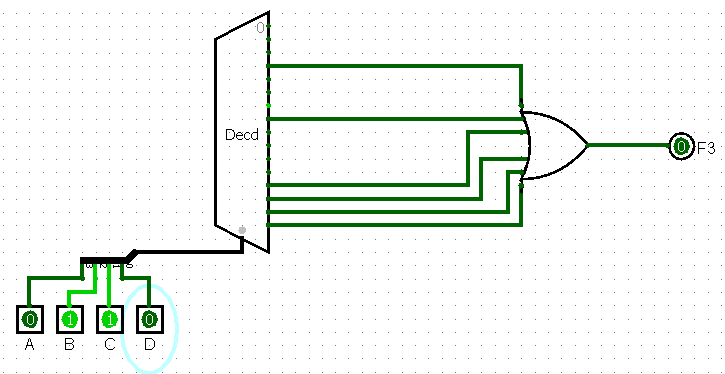




#### RESULT OBTAINED

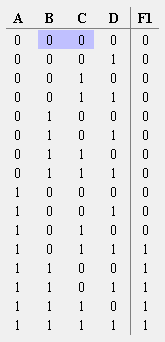
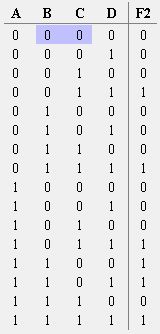
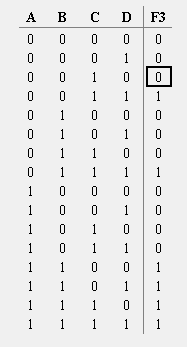
 

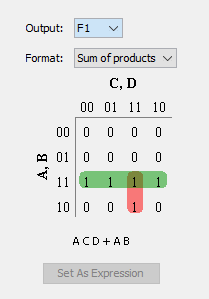
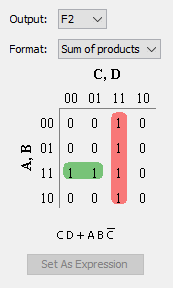
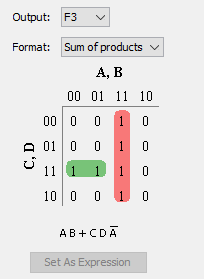
 

#### **4 to 1 Multiplexer**

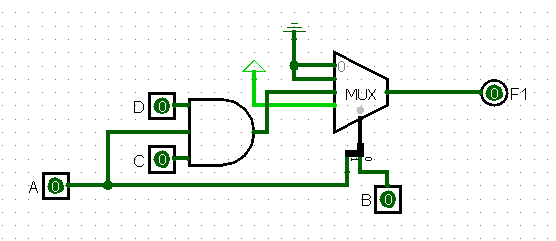
#### TURTH TABLE

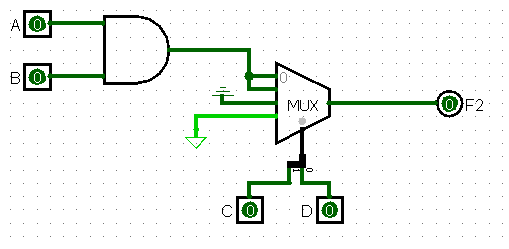
  

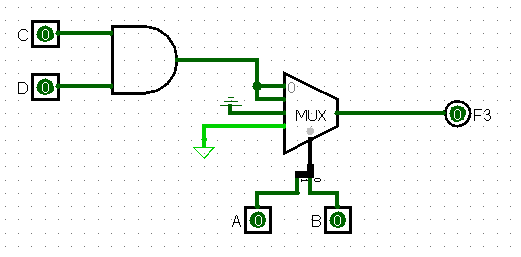
#### KMAP

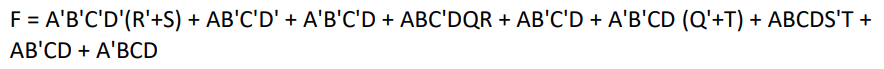
#### CIRCUIT







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



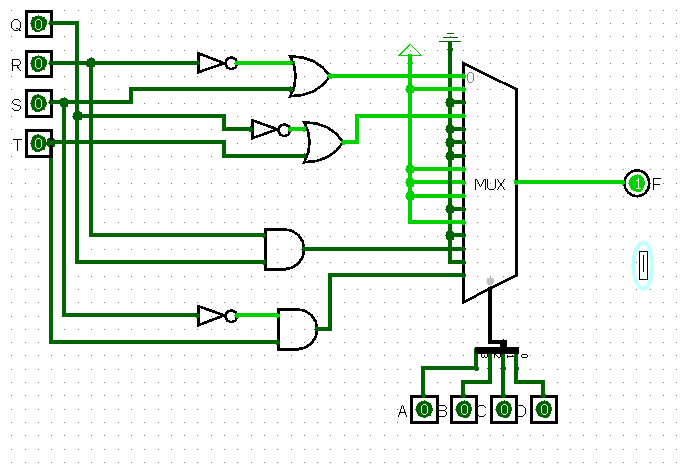
## Solution:

#### EXPLAIN

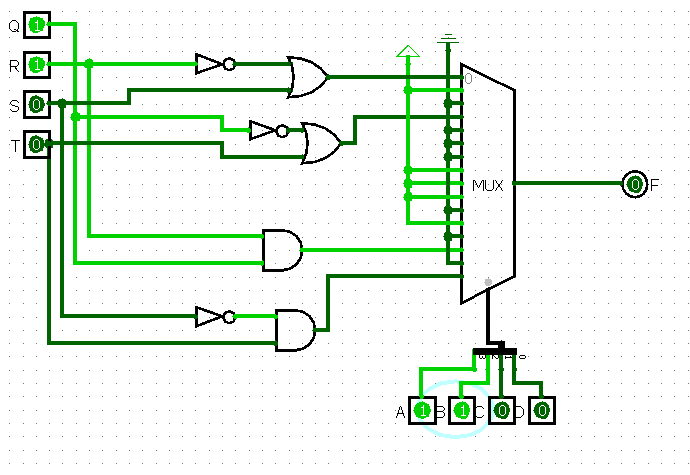
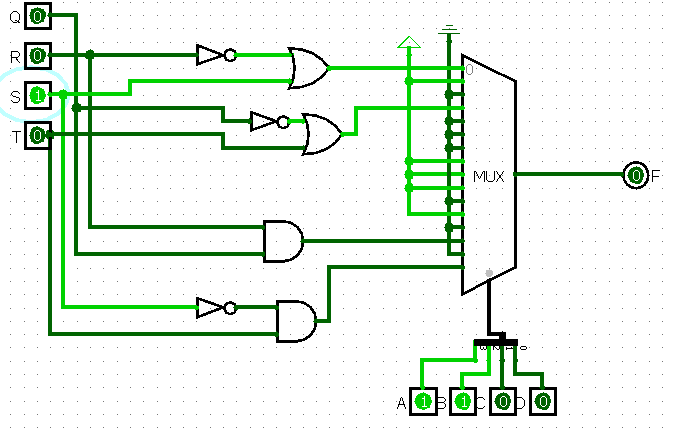
The selector lines can be used as ABCD respectively and the corresponding expression can be found by simply mapping it with the terms given in the question.

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

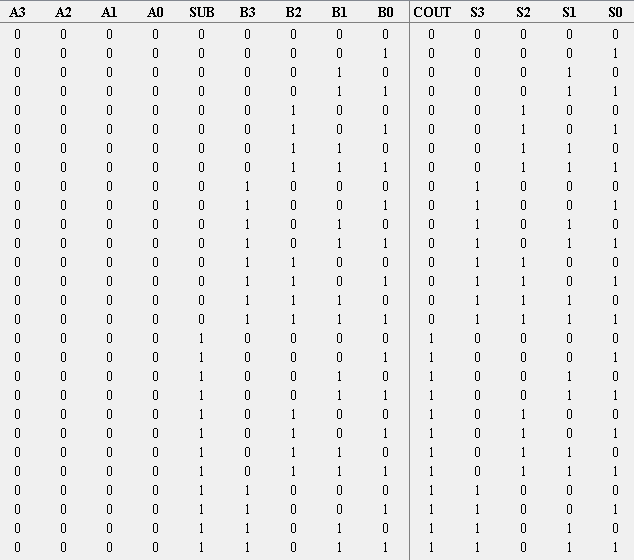
 

# Implement (a) 4 bit controllable adder/subtractor (b) 8 bit controllable adder/subtractor. Use the Adder available in Arithmetic library in logisim.

## Solution:

## 4 bit controllable adder/subtractor

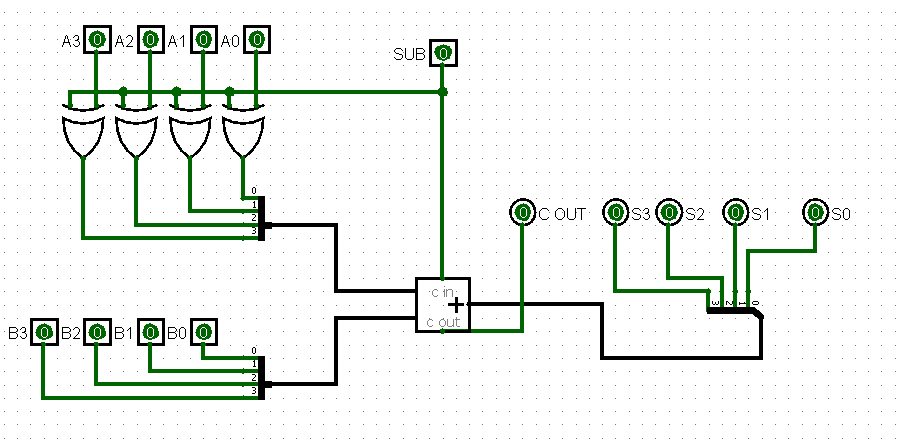
#### TURTH TABLE

 … 256 VALUES

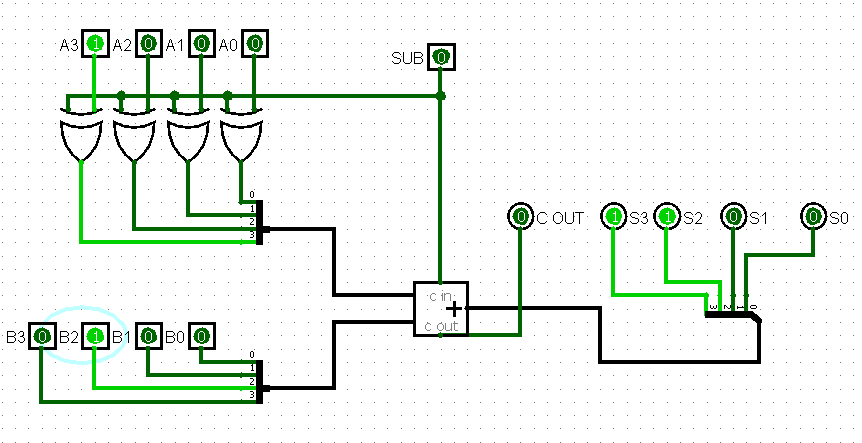
#### EXPLAIN

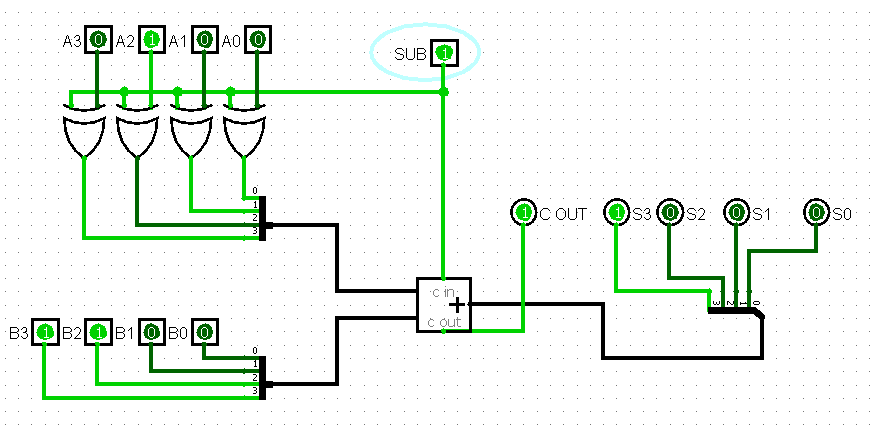
Subtraction can be done using 2s complement. Which is nothing but 1s compliment + 1. For subtraction we pass the cin as 1 and we take 1 XOR with the input bits which is to be subtracted. This results in 1’s complement to this the adder adds 1 (cin) which results in 2’s compliment.

#### CIRCUIT



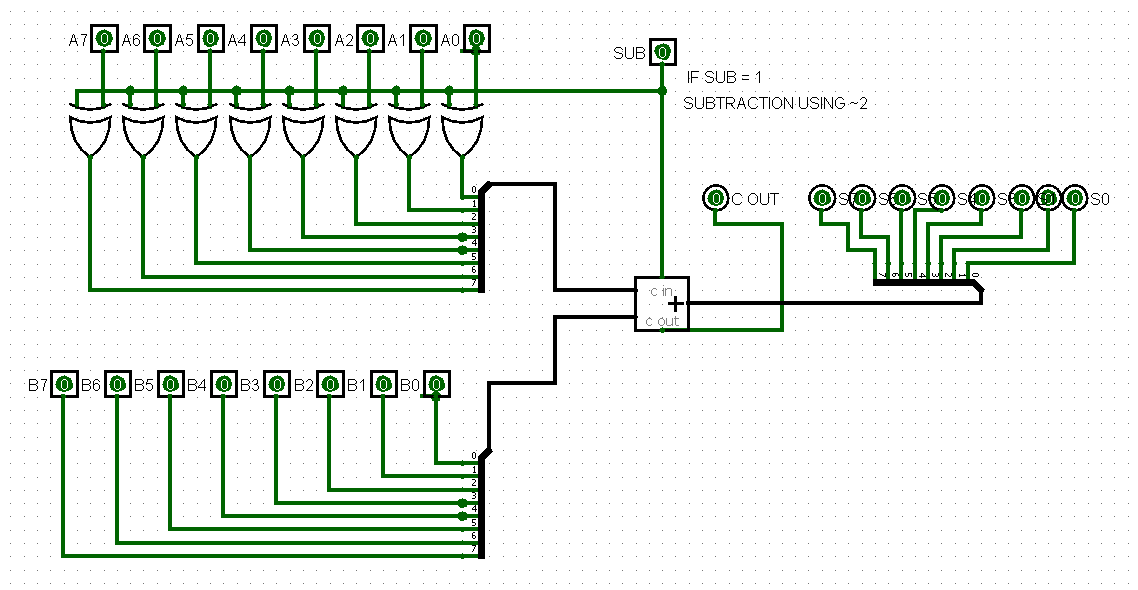
#### RESULT OBTAINED





#### **8 bit controllable adder/subtractor**

#### CIRCUIT



#### RESULT OBTAINED

