

# 1) Design a circuit that will perform addition of two single digit BCD numbers and display the output on seven segment displays. Use the Adder available in Arithmetic library in Logisim.

## Solution:

### Design and explanation

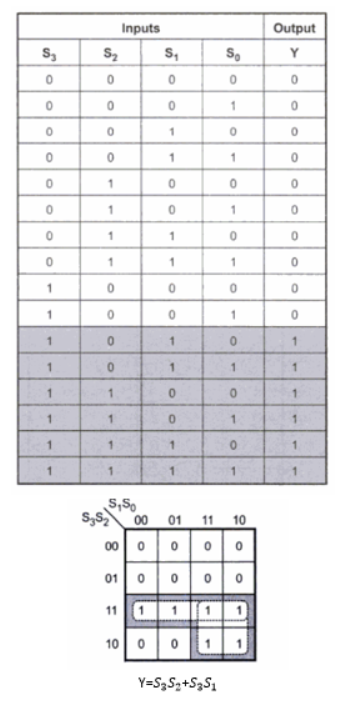
To implement BCD adder, we require:

• 4-bit binary adder for initial addition

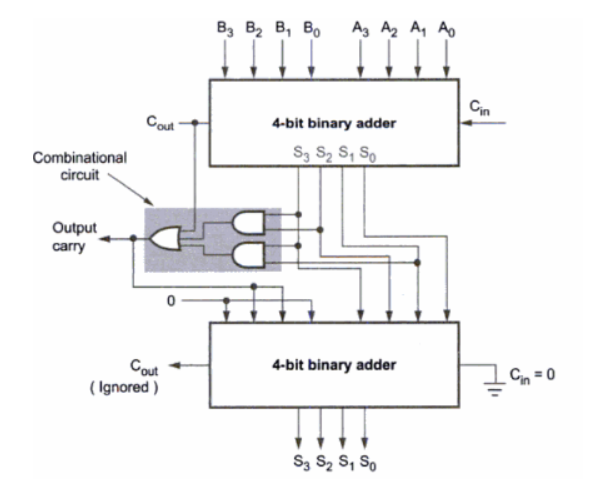
• Logic circuit to detect sum greater than 9

• One more 4-bit adder to add 0110201102 in the sum if sum is greater than 9 or carry is 1

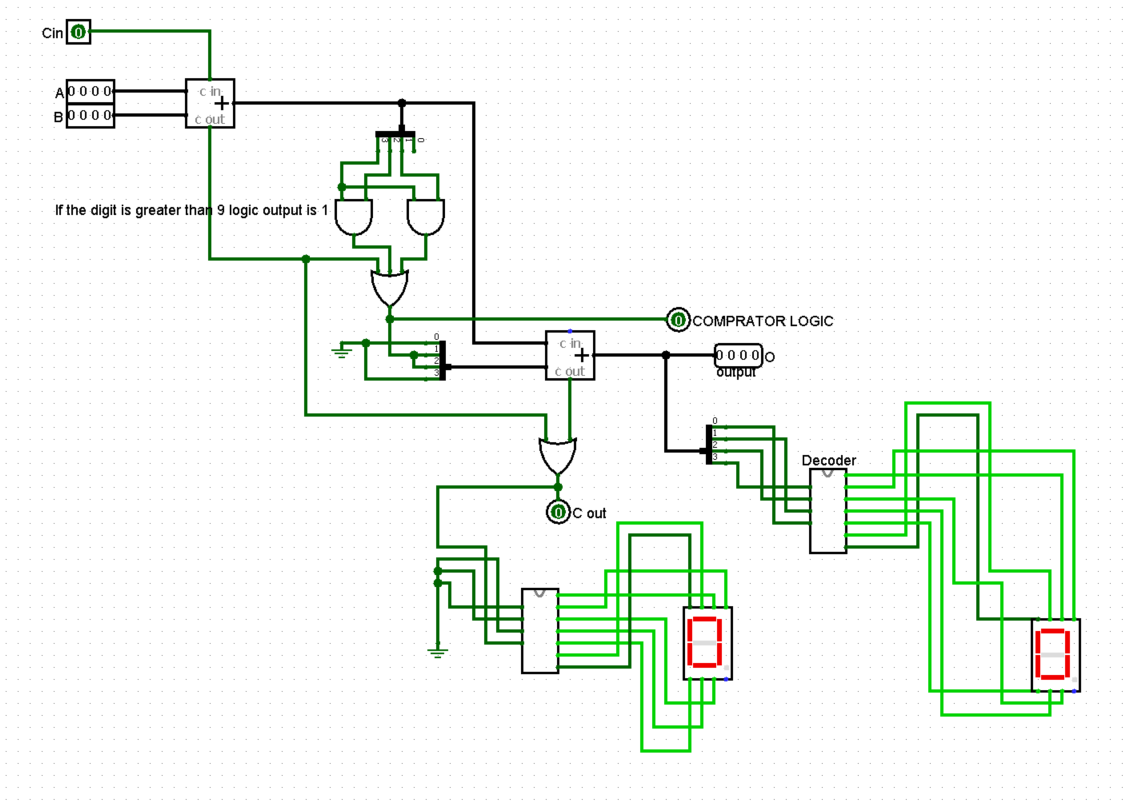
The logic circuit to detect sum greater than 9 can be determined by simplifying the Boolean expression of given truth Table.



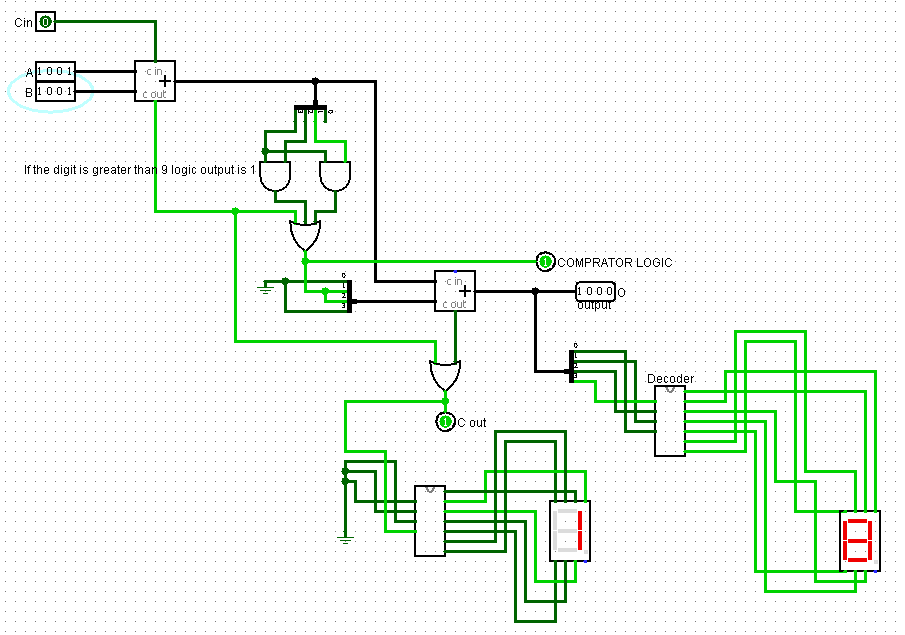
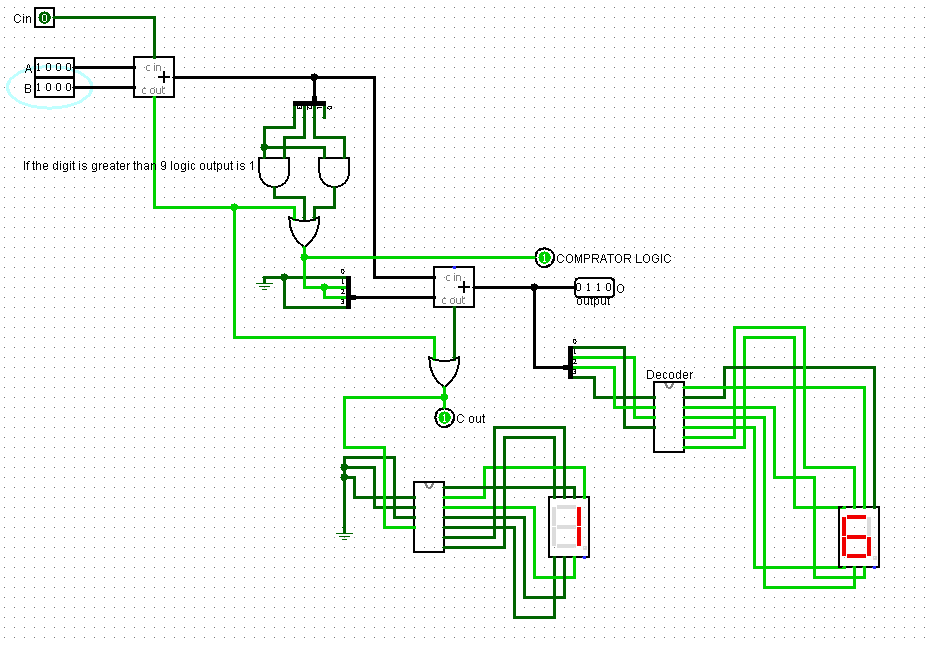
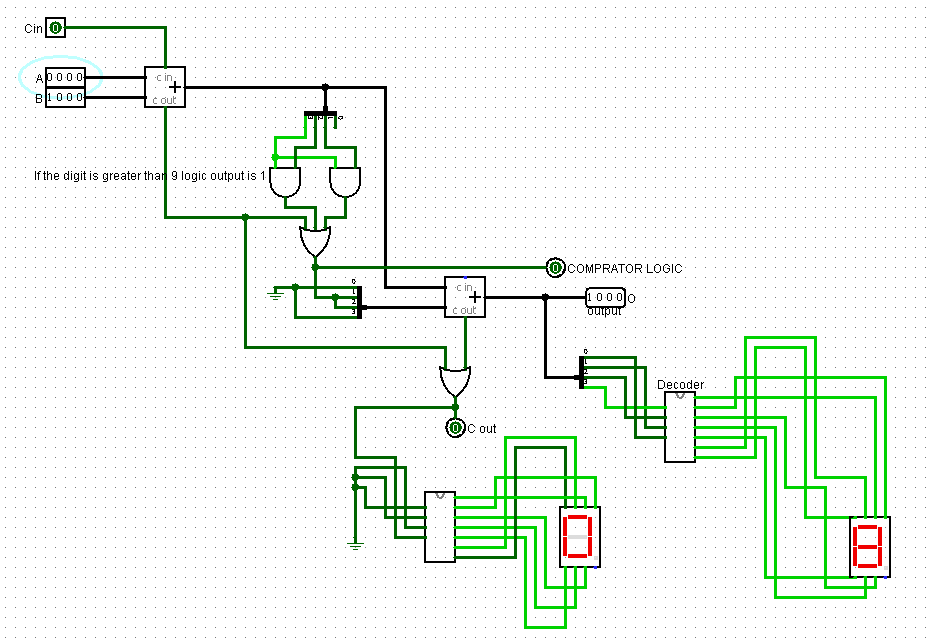
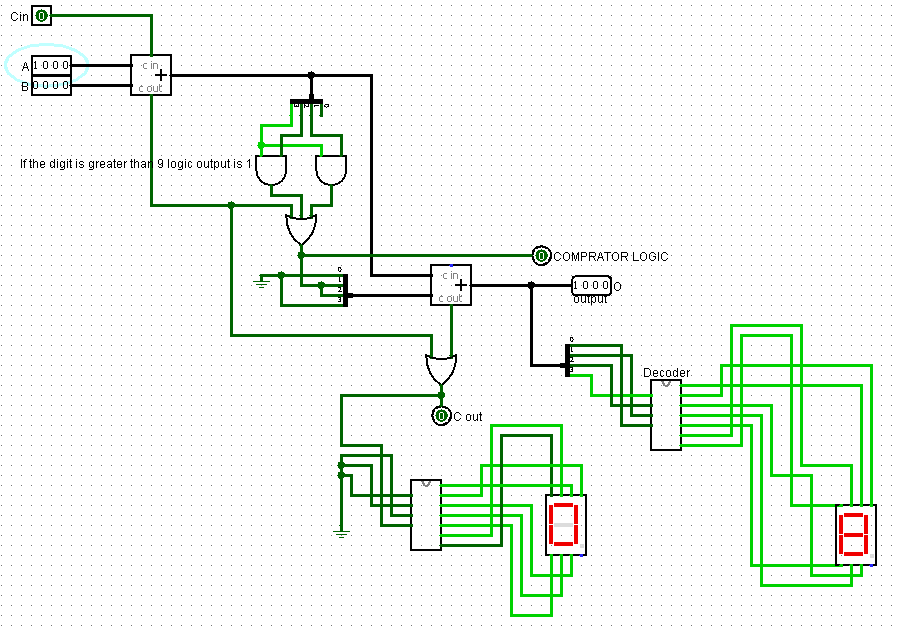
Y=1 indicates sum is greater than 9. We can put one more term, Cout in the above expression to check whether carry is one. If any one condition is satisfied, we add 6(0110) in the sum.

Now we just display the output in the 7 segment LED using the decoder which we implemented in Lab 1.

### CIRCUIT



### RESULT OBTAINED



# 2) Design a circuit that will perform addition of two single digit BCD numbers and display the output on seven segment displays. Use the Adder available in Arithmetic library in Logisim.

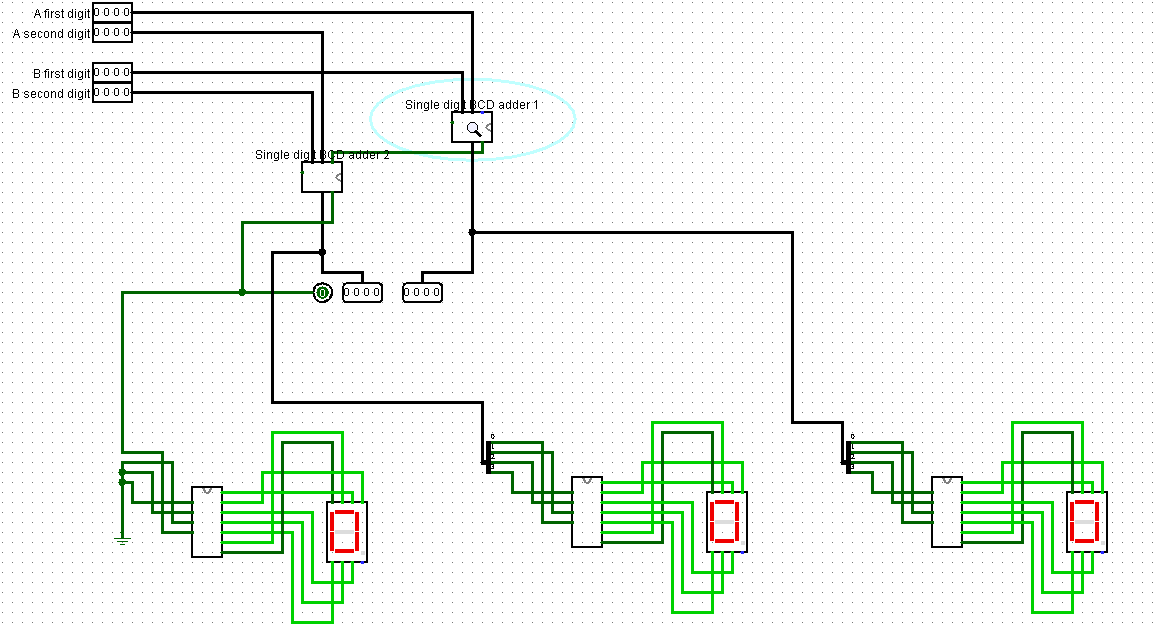
## Solution:

### Design and explanation

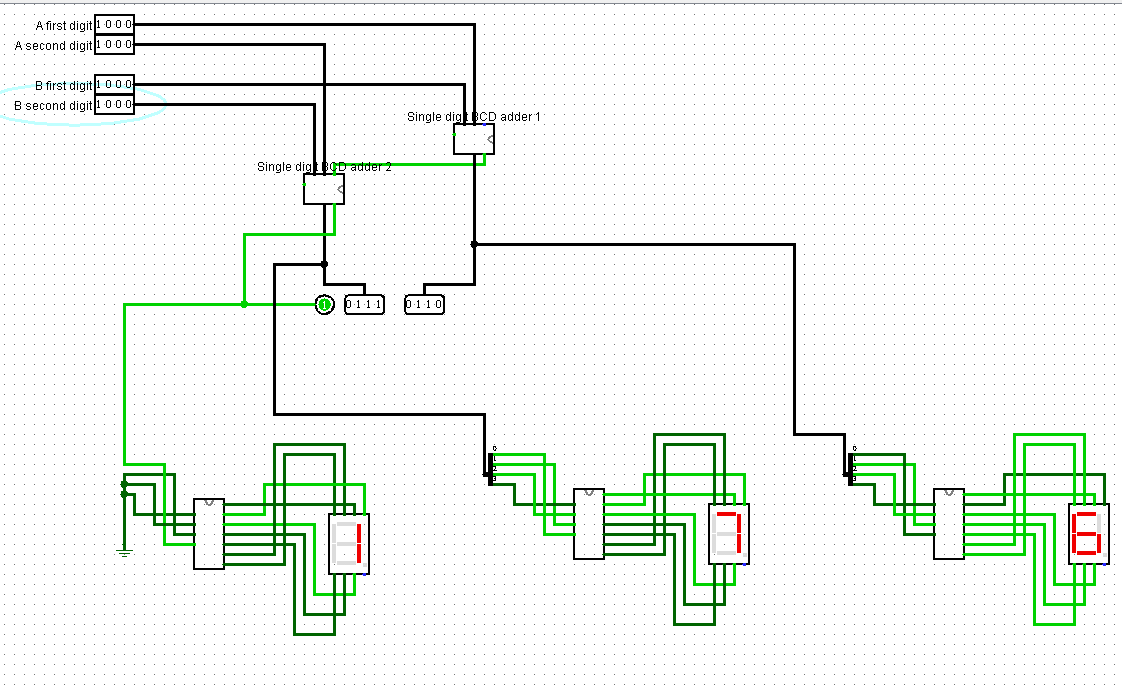
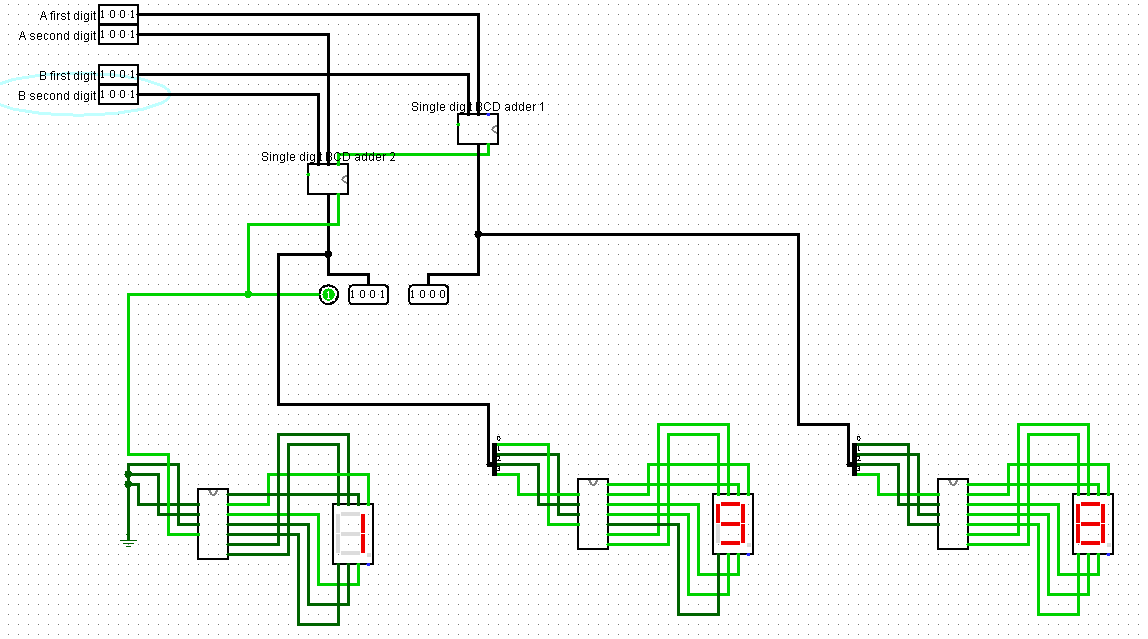
To implement BCD adder, for two digits we can use the previous single digit adder which works up to (9 + 9 = 18)

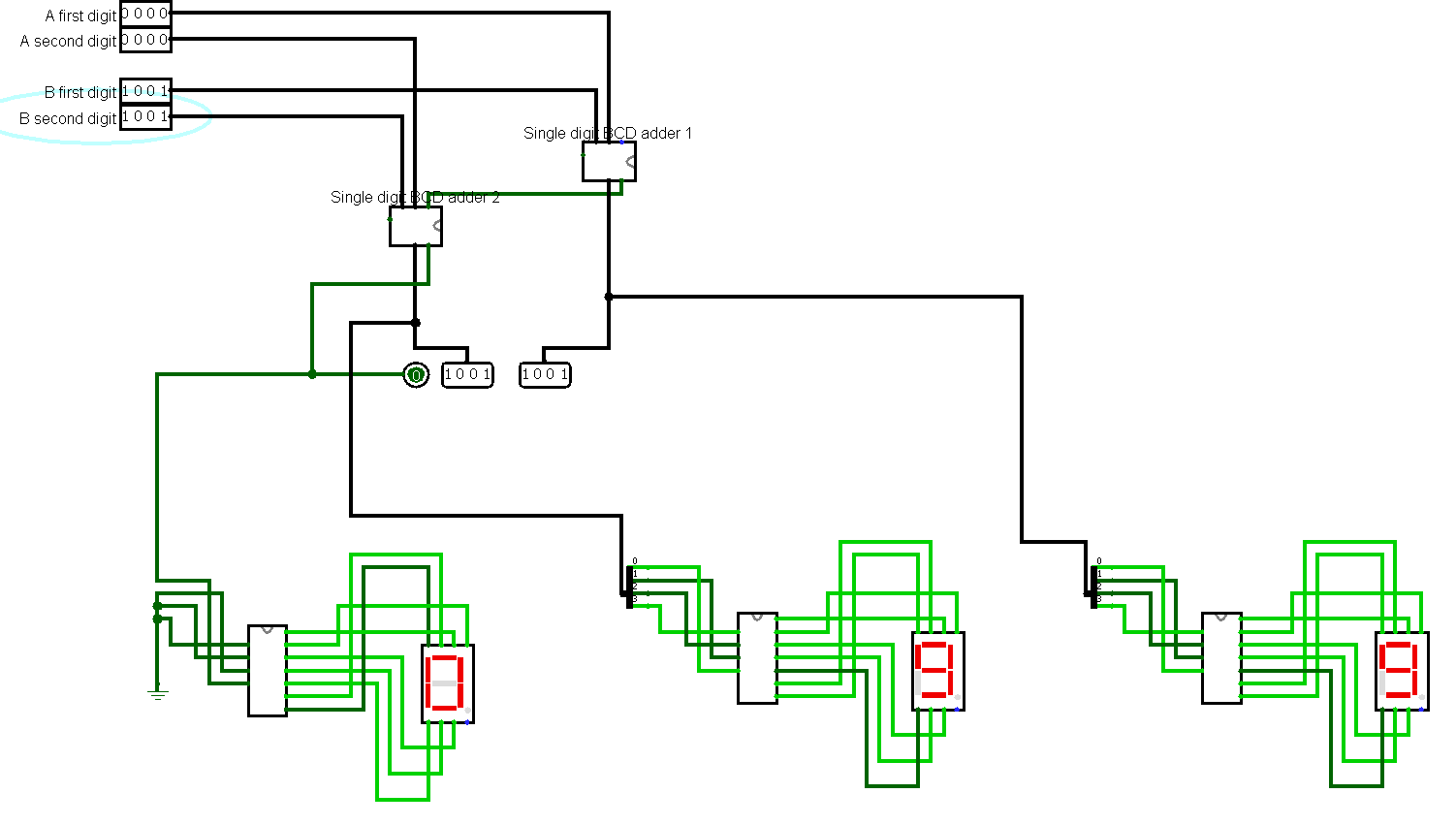
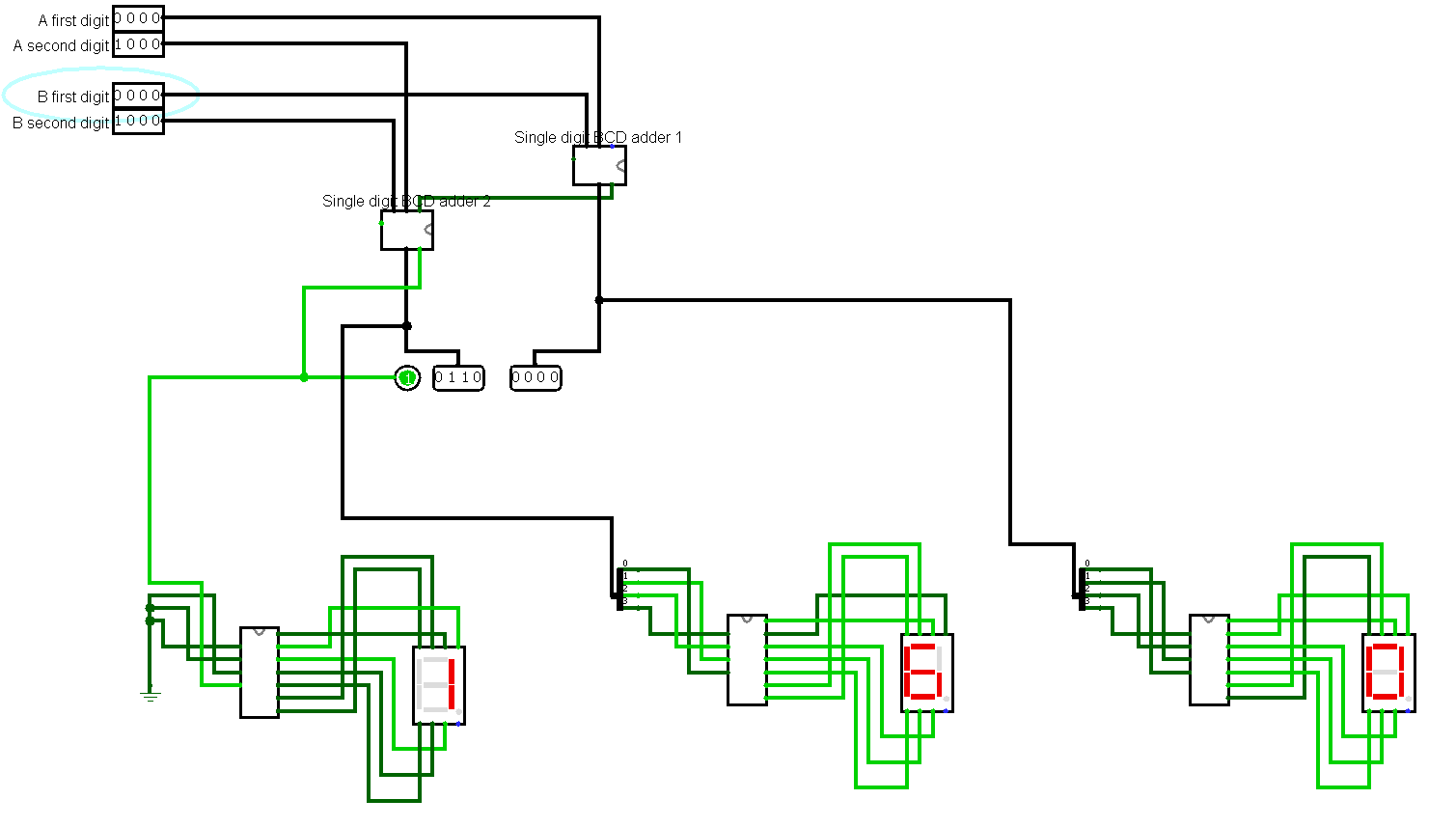
The first adder adds the first digit (BCD), in the second adder we add the second digit of the BCD number, along with the digit we also give the Cin input as the comparator logic, now it will work for two digit BCD numbers, the following output is displayed in the 7segment led.

### CIRCUIT



### RESULT OBTAINED





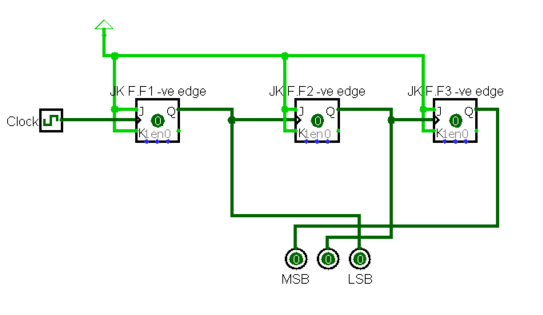
# 3) (a) Design a modulo 8 ripple up counter using JKFF (b) modify it to a modulo-8 ripple up-down counter

## Solution 8 ripple up:

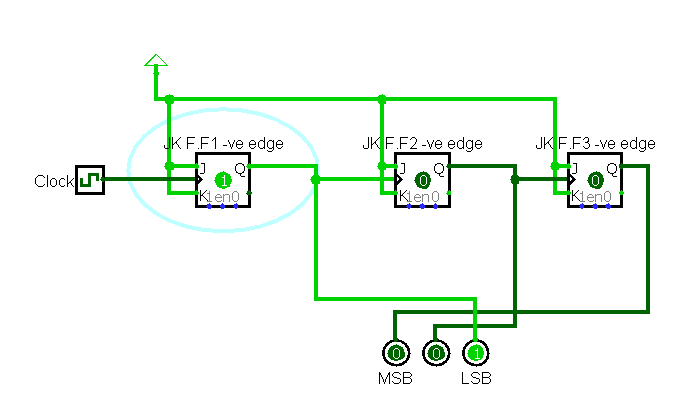
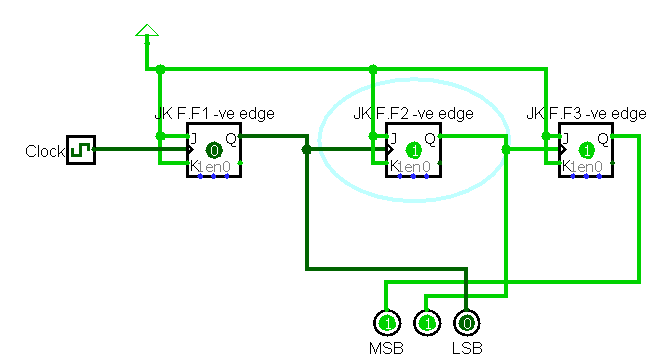
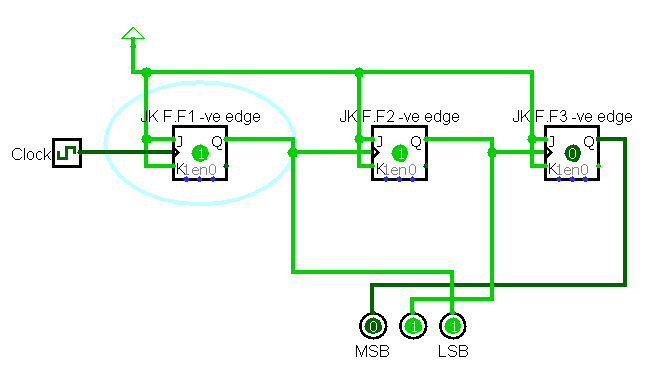
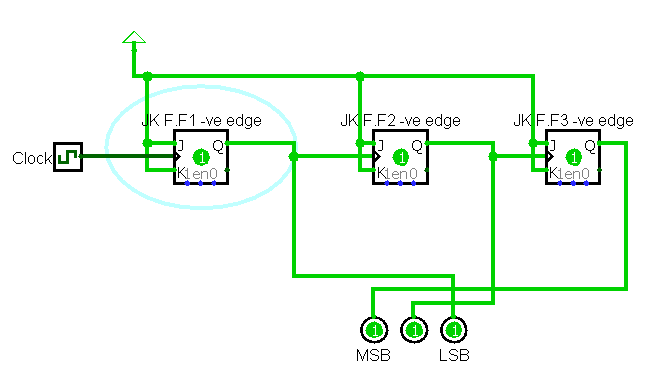
### Design and explanation

We design the mod 8 counter using 3 JK FF in a cascade manner, the values of J and K are always one and the clocks are different for each flipflop, we give the output of the previous flipflop as the clock input to the next flipflop which halves the frequency, by doing this we can count up to 7. The value stored in FF1 is the LSB, and FF3 is the MSB

### CIRCUIT



### RESULT OBTAINED

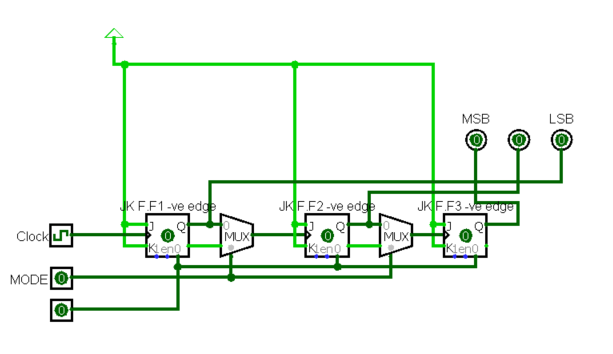
Counts upto 7 and repeats from 0 after that.

## Solution modulo-8 ripple up-down counter:

### Design and explanation

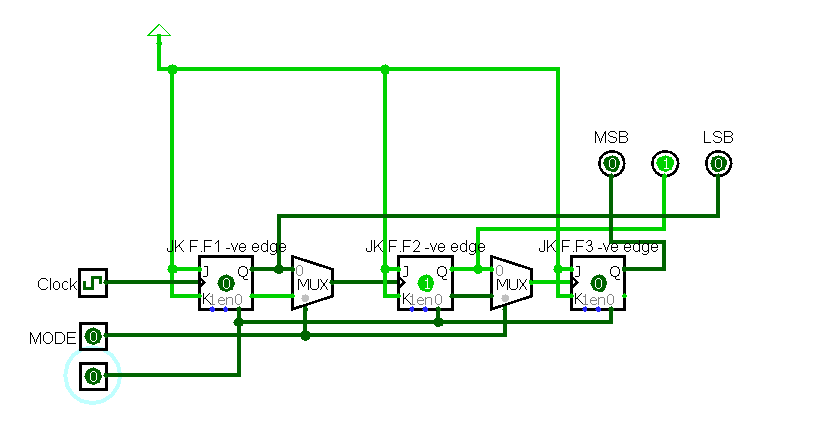
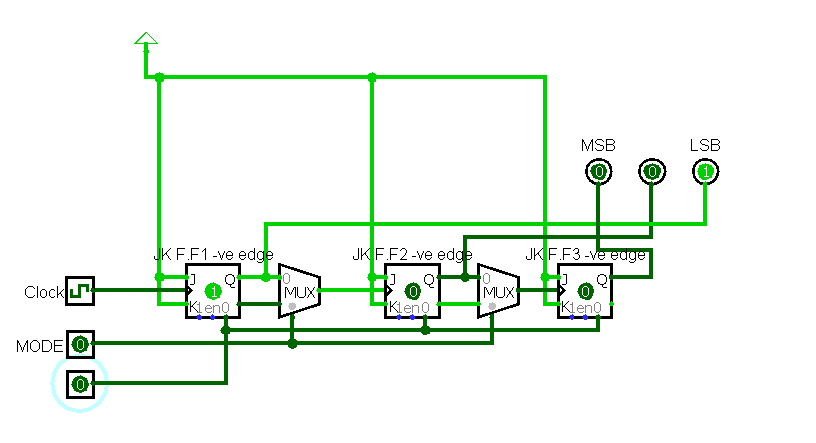
To design an up down counter we can use a Mode variable which when 0 does up counting and 1 does down counting. To perform down counting we just need to give the compliment of the clock to the next flip flop, now the counter counts from backwards when Mode is equal to 1.

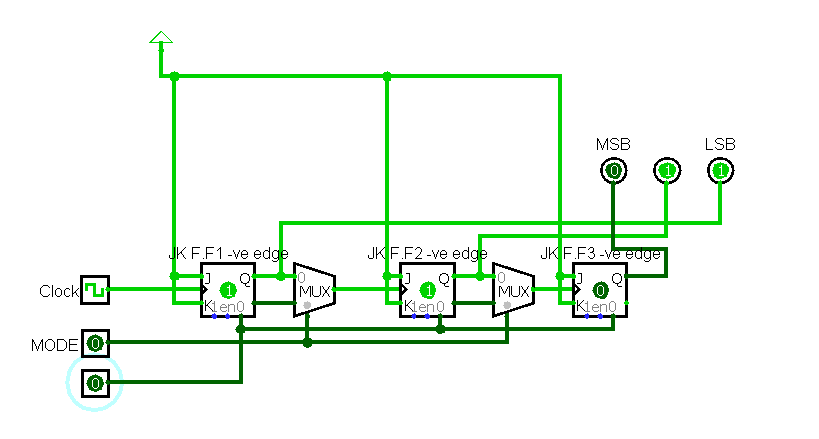
### CIRCUIT



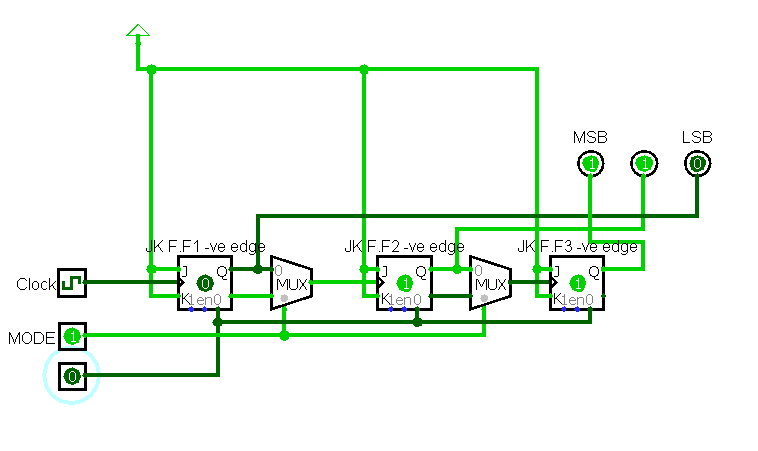
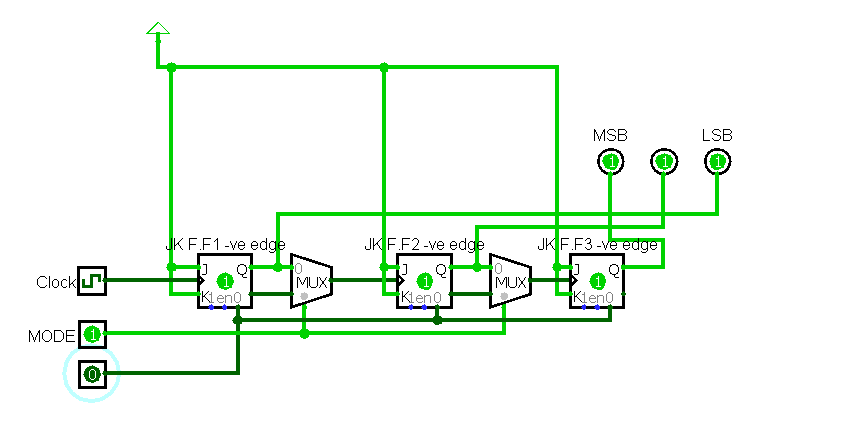
### RESULT OBTAINED

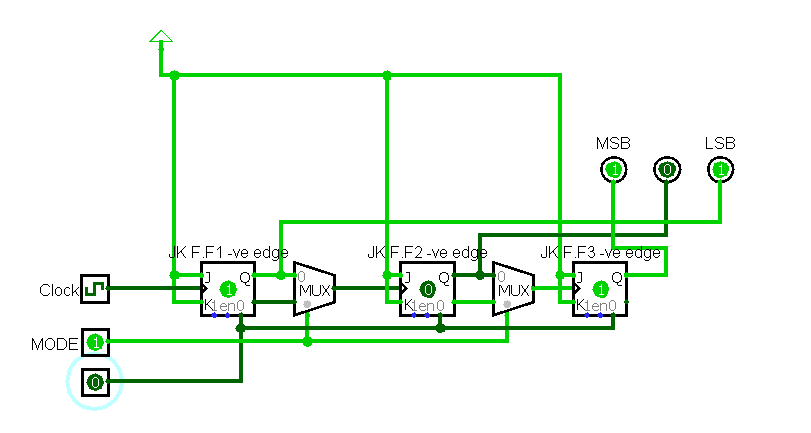
Up count





Down count



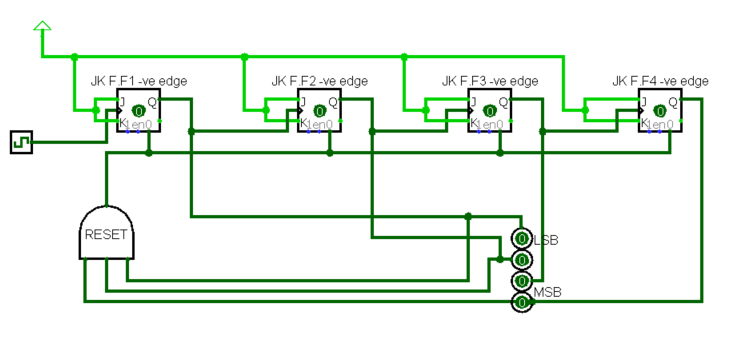


# 4) Design a modulo-11 ripple counter using JKFF

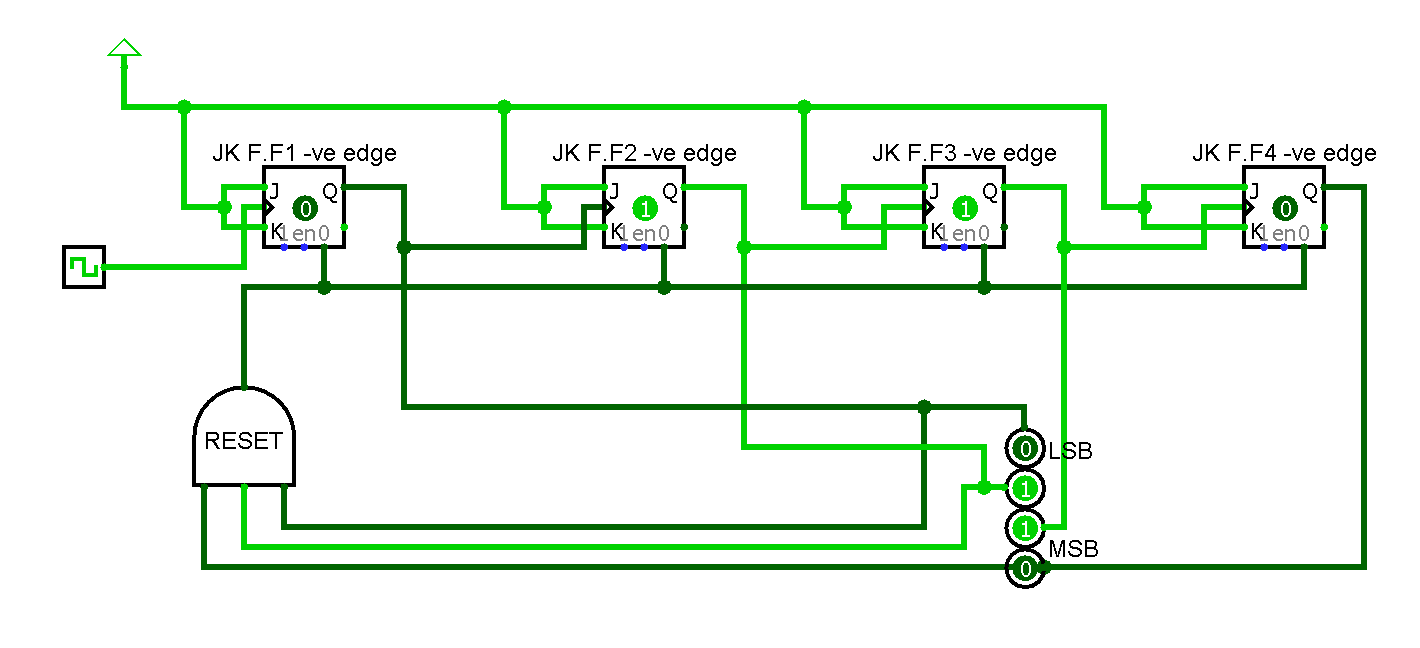
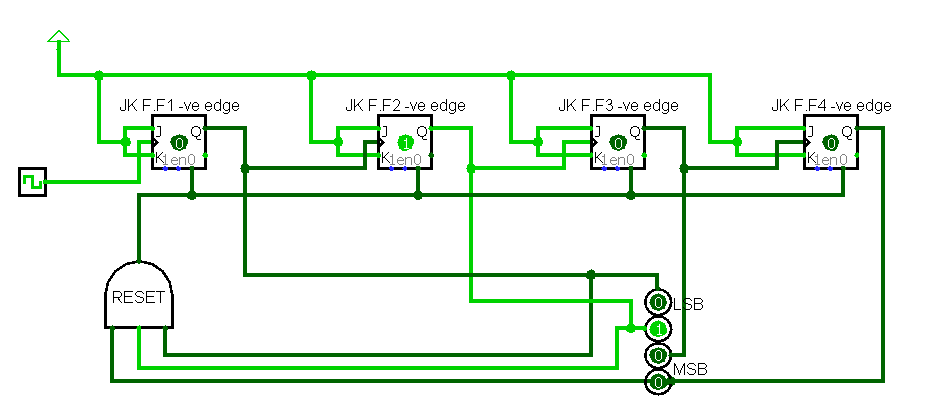
## Solution:

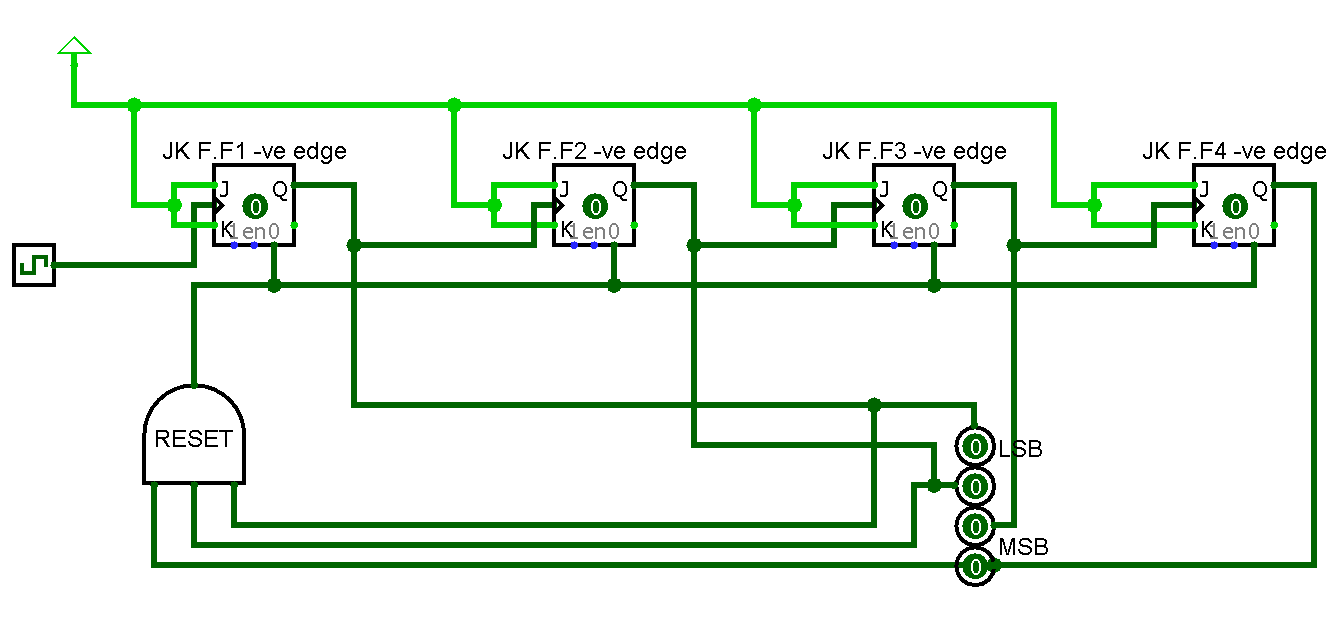
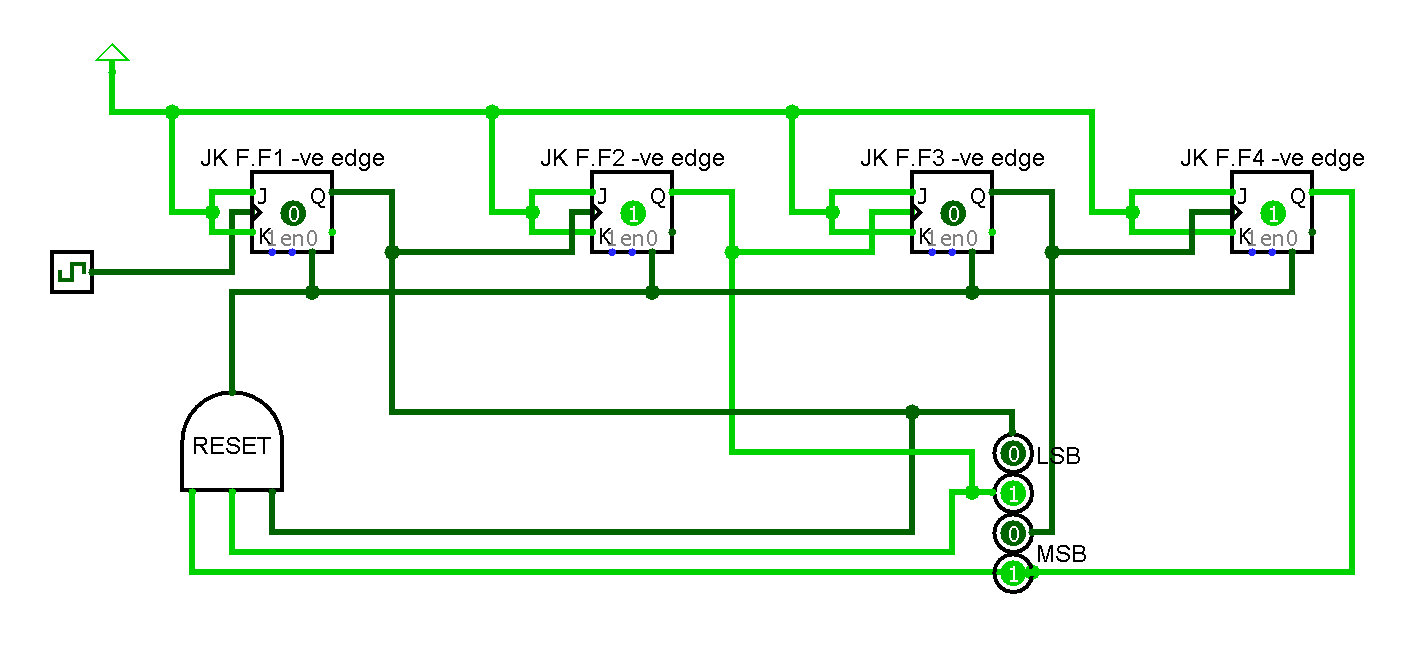
### Design and explanation

### CIRCUIT



### RESULT OBTAINED





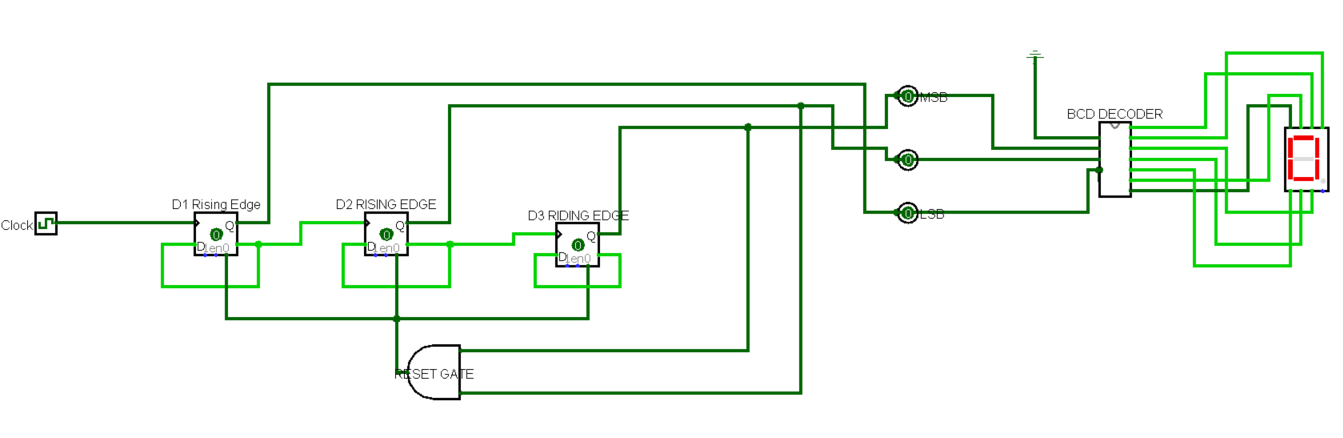
As we can see the counter is reset after 10 when the current state is 11 now we perform the asynchronous reset and start counting form 0 again.

# 5) Design a modulo-6 ripple counter using DFF and display the output using 7 segment LED

## Solution:

### Design and explanation

### CIRCUIT



### RESULT OBTAINED

