ASSIGNMENT - 4

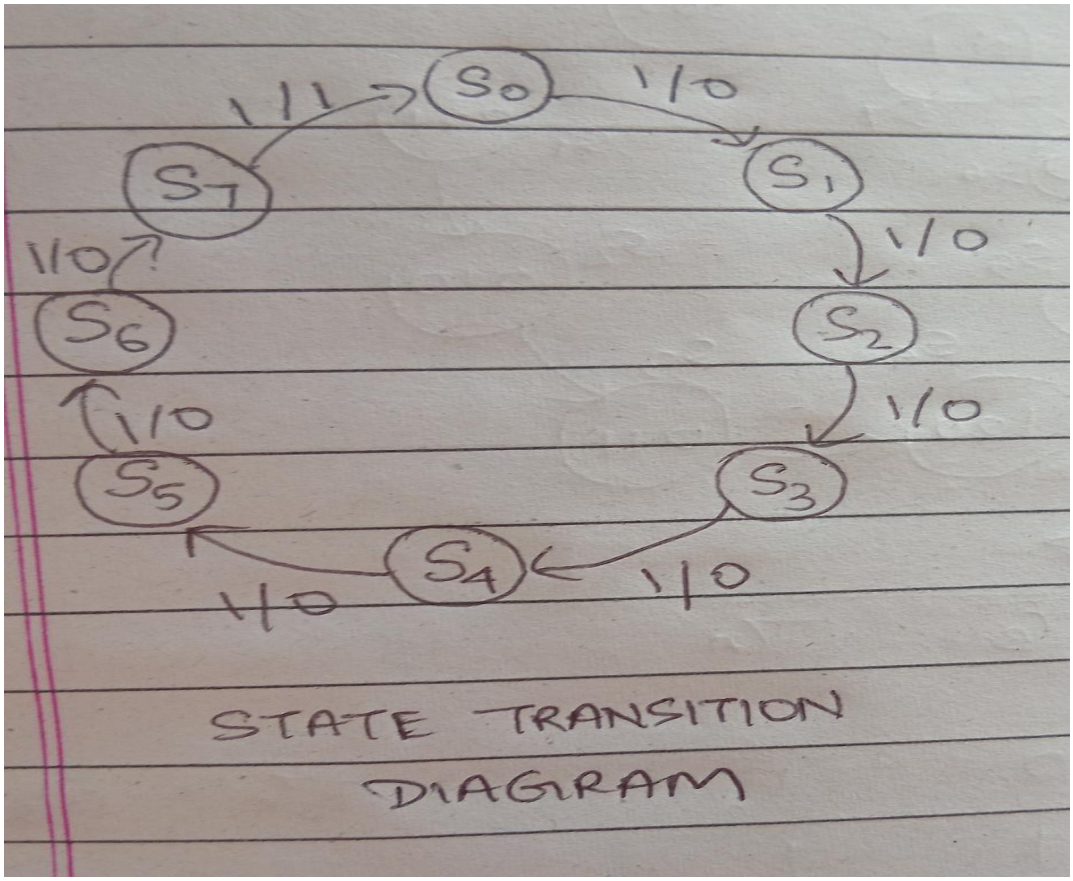
3-bit Gray Code Counter

Approach:

We have used 8 states to implement the 3-bit Gray code counter. The states are as follows : S_0 = 000, S_1 = 001, S_2 = 010, S_3 = 011, S_4 = 100, S_5 = 101, S_6 = 110, S_7 = 111.

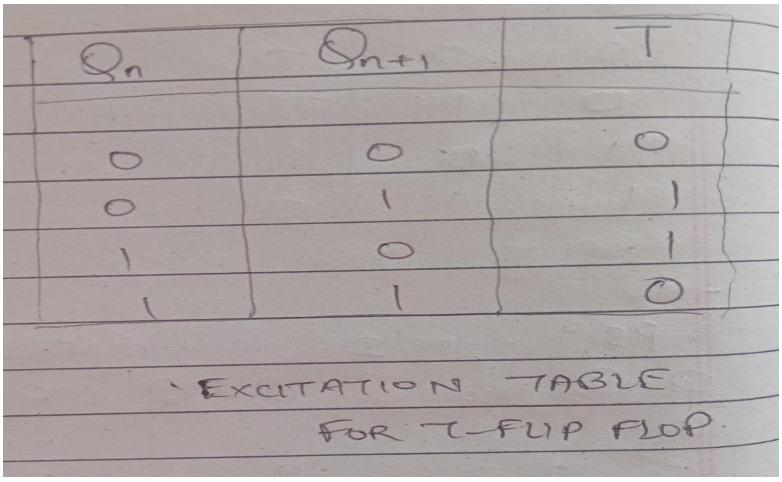
There is an input pulse of 1 bit having value = 1. With each input pulse, a state transition takes place. Output is high only when the state changes from S_7 to S_0. Otherwise, the output is zero.

State transition diagram :



Excitation tables :

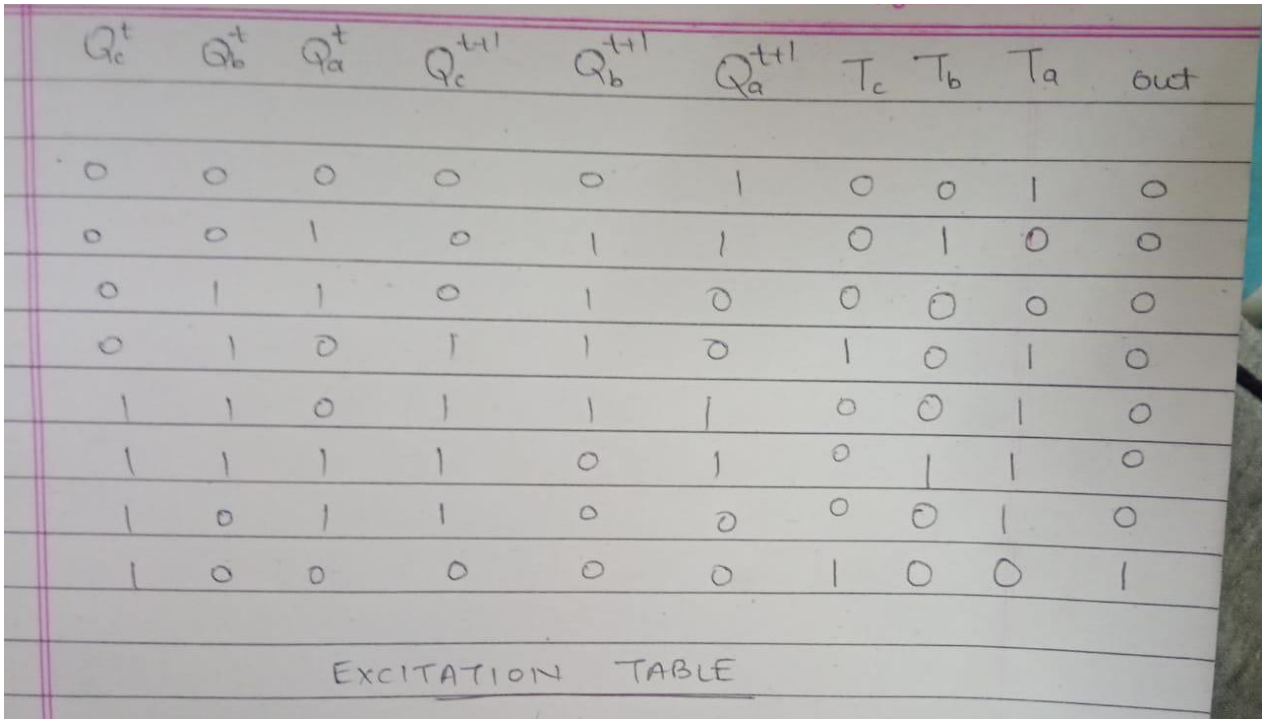
We are using T-flip flops for implementing the gray code counter. The excitation table for the T-flip flop can be shown as follows :



Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

EXCITATION TABLE FOR T-FLIP FLOP

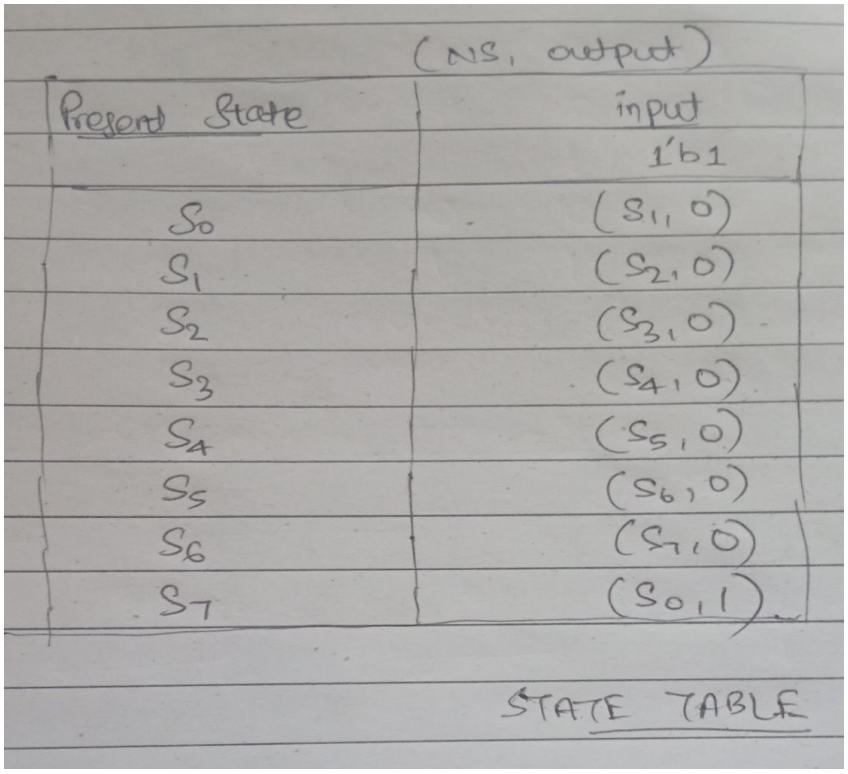
We represent the MSB of the states as Q_c and the LSB as Q_a. The superscript 't' denotes the present state and the superscript 't+1' denotes the next state. The excitation table for all the 3 bits is as follows :



Q_c^t	Q_b^t	Q_a^t	Q_c^{t+1}	Q_b^{t+1}	Q_a^{t+1}	T_c	T_b	T_a	out
0	0	0	0	0	1	0	0	1	0
0	0	1	0	1	1	0	1	0	0
0	1	1	0	1	0	0	0	0	0
0	1	0	1	1	0	1	0	1	0
1	1	0	1	1	1	0	0	1	0
1	1	1	1	0	1	0	1	1	0
1	0	1	1	0	0	0	0	1	0
1	0	0	0	0	0	1	0	0	1

EXCITATION TABLE

State Table:

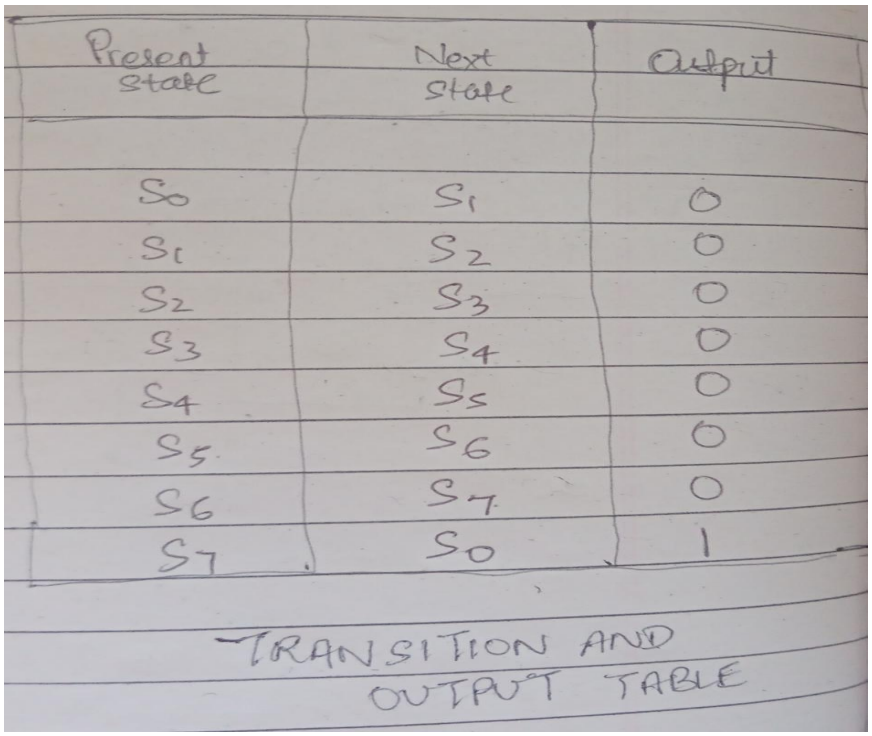


(NS, output)

Present State	input 1'b1
S_0	$(S_1, 0)$
S_1	$(S_2, 0)$
S_2	$(S_3, 0)$
S_3	$(S_4, 0)$
S_4	$(S_5, 0)$
S_5	$(S_6, 0)$
S_6	$(S_7, 0)$
S_7	$(S_0, 1)$

STATE TABLE

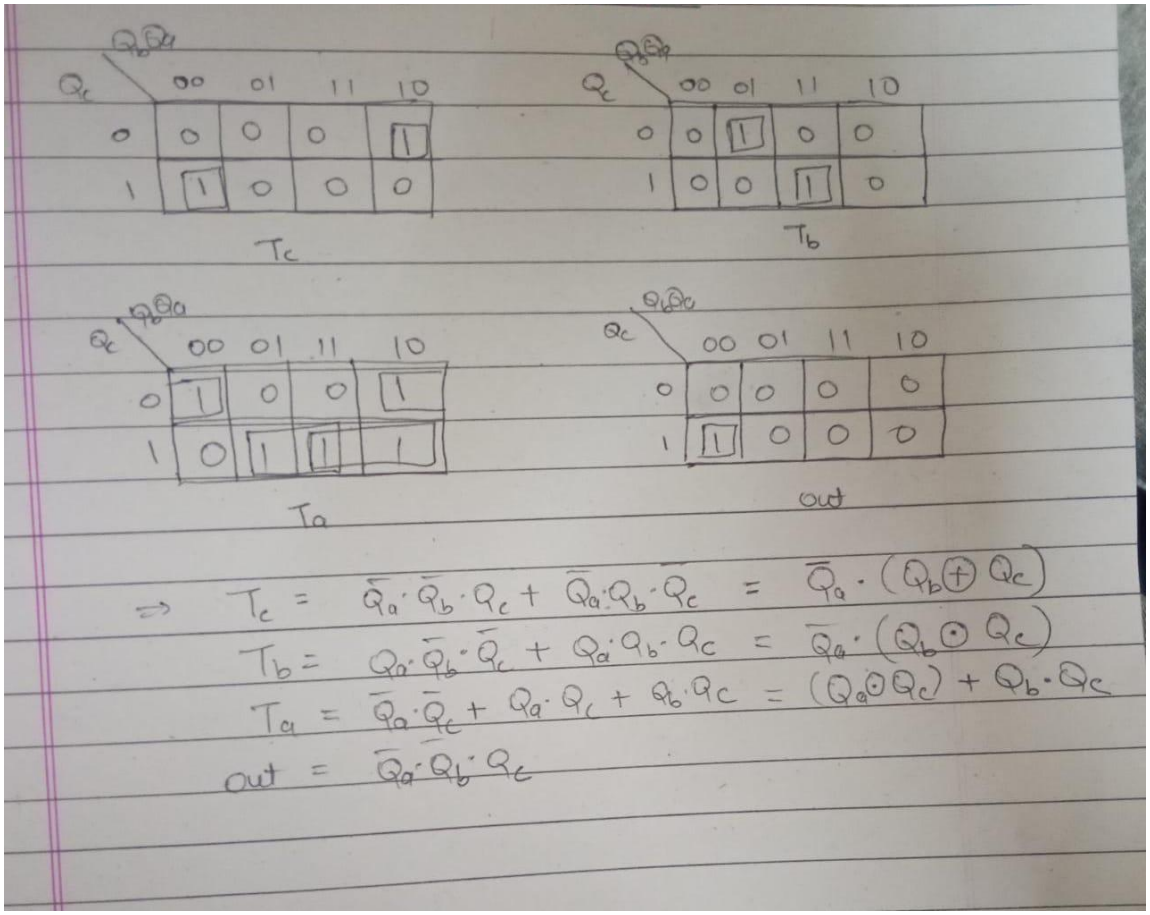
Transition and Output Table:



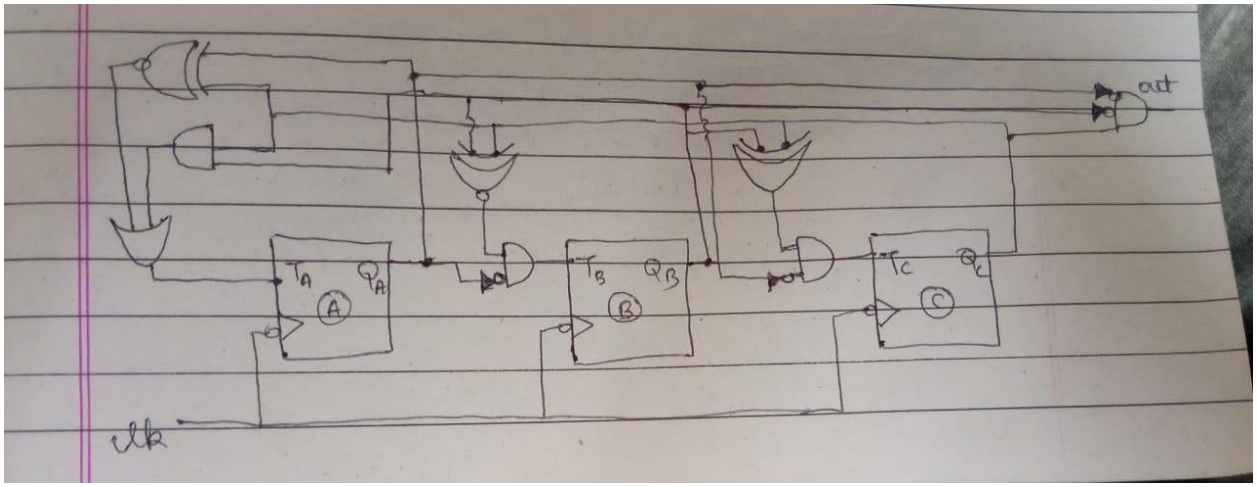
Present state	Next State	Output
S_0	S_1	0
S_1	S_2	0
S_2	S_3	0
S_3	S_4	0
S_4	S_5	0
S_5	S_6	0
S_6	S_7	0
S_7	S_0	1

TRANSITION AND OUTPUT TABLE

K-maps:



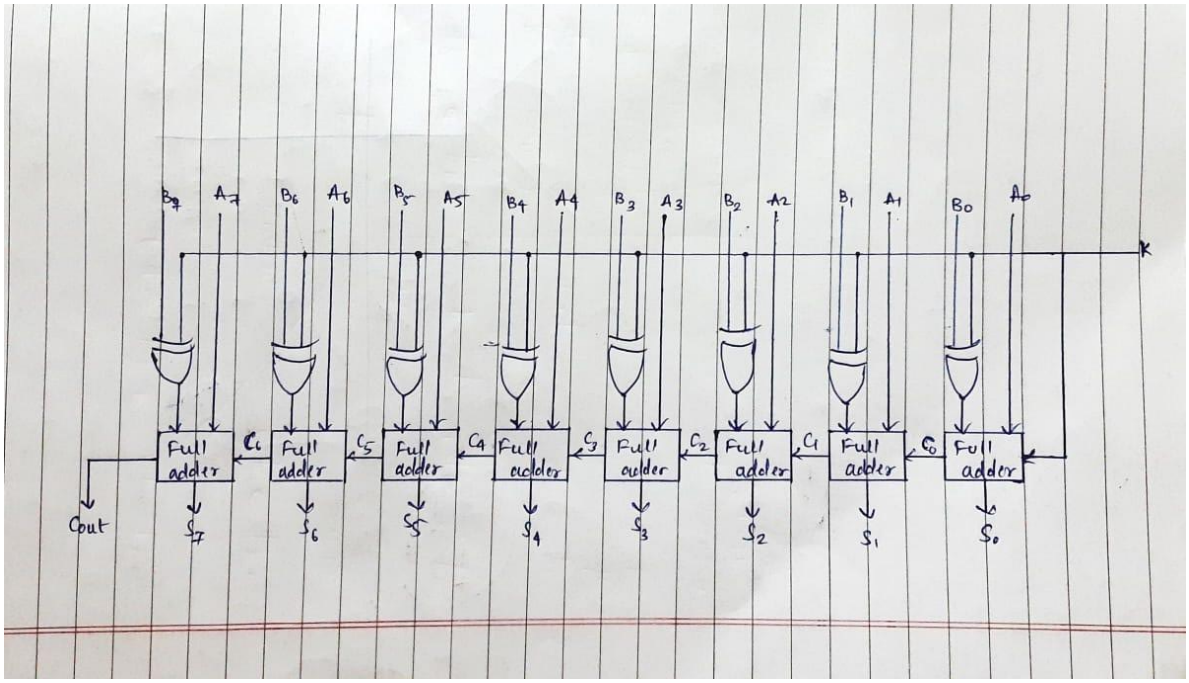
Circuit Diagram :



8-bit Adder/Subtractor

We are given two 8-bit numbers(namely A and B) and we have to either add or subtract them.
opcode = 0 for addition and
opcode = 1 for subtraction.

Circuit:



Here, A_7, A_6, …A_0 are the input bits of A and B_7, B_6, …B_0 are the input bits of B.
S_7, S_6, …., S_0 are the output bits of the result.
C_{out} is the carry out and k is the opcode.

Approach:

We have implemented an 8-bit adder/subtractor using Full Adder and XOR gate.

For addition, k=0 and
For subtraction, k=1

Truth table for XOR gate is as follows:

X	Y	XOR
0	0	0
1	0	1
0	1	1
1	1	0

We observe,
When Y=0, the output XOR for a given input X is **X** itself and
When Y=1, the output XOR for a given input X is $\overline{\mathbf{X}}$.

So, when we want to add the number we simply pass k=0 and
when we want to subtract the number (that is adding 2's complement) we pass k=1.

For adding A and B, the XOR of B and k=0 simply gives B, and we do the bitwise addition of A and B using 1-bit Full Adder by feeding the carry out of the previous bit to the next bit.

We do the subtraction by 2's complement method. The XOR of B and k=1, gives $\overline{B}$ and for LSB of result we add k=1, as 2's complement is obtained by adding one to the 1's complement of B. Hence, the result is obtained by adding 2's complement of B to A using 1-bit Full Adder by feeding the carry out of the previous bit to the next bit.

Full Adder Truth Table:

X	Y	C_{in}	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

We observe,
Sum = $X \oplus Y \oplus Z$
Carry = $X \cdot Y + X \cdot C_{in} + C_{in} \cdot Y$

Overflow condition:

Overflow occurs when $C_6 \neq C_{out}$.

No overflow when $C_6 = C_{out}$.

Truth Table:

C_6	C_{out}	Overflow
0	0	0
0	1	1
1	0	1
1	1	0

Therefore,

Overflow = $C_6 \oplus C_{out}$.