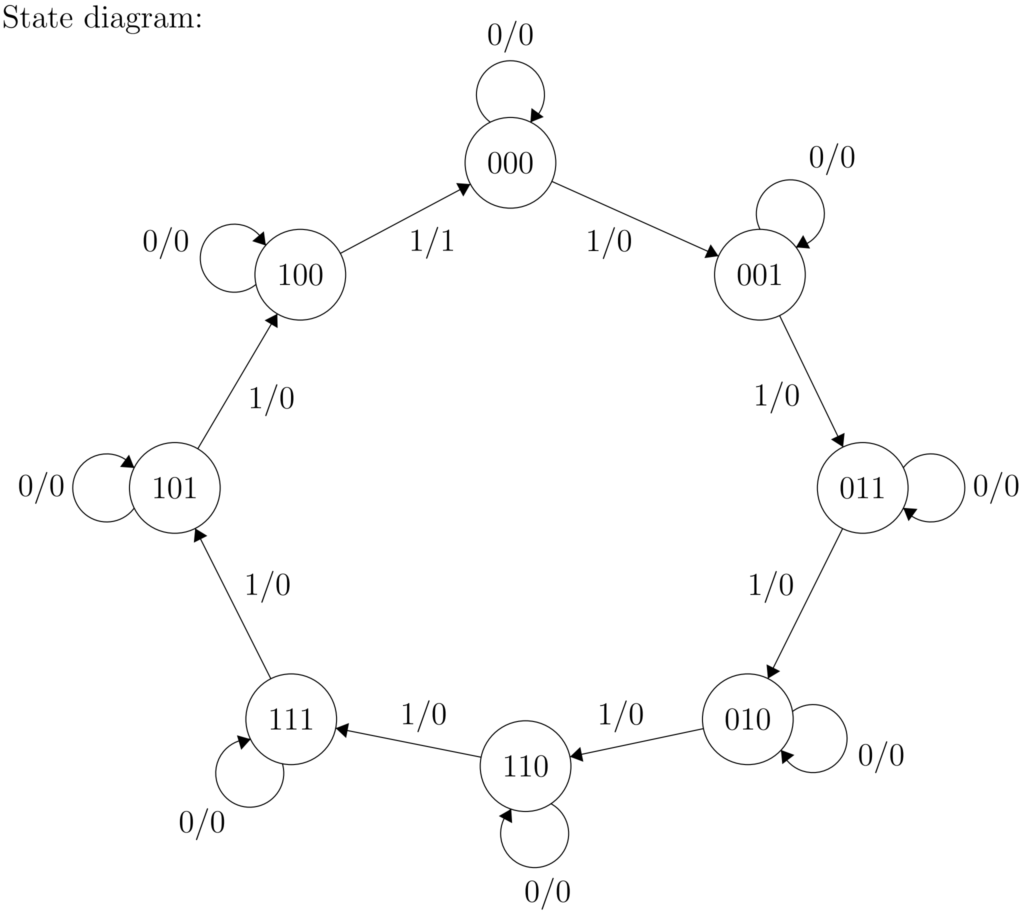
CS 220A — Computer Organization

Group No: 33 **Due Date:** March 6 2022, 23:59
Roll Number: 190616, 190714, 190773 **Assignment Number:** 4

Q1. 3-bit Gray code counter:

(a) State diagram:



Excitation table:

PS (ABC)	in	NS (ABC)
000	0	000
000	1	001
001	0	001
001	1	011
011	0	011
011	1	010
010	0	010
010	1	110
110	0	110
110	1	111
111	0	111
111	1	101
101	0	101
101	1	100
100	0	100
100	1	000

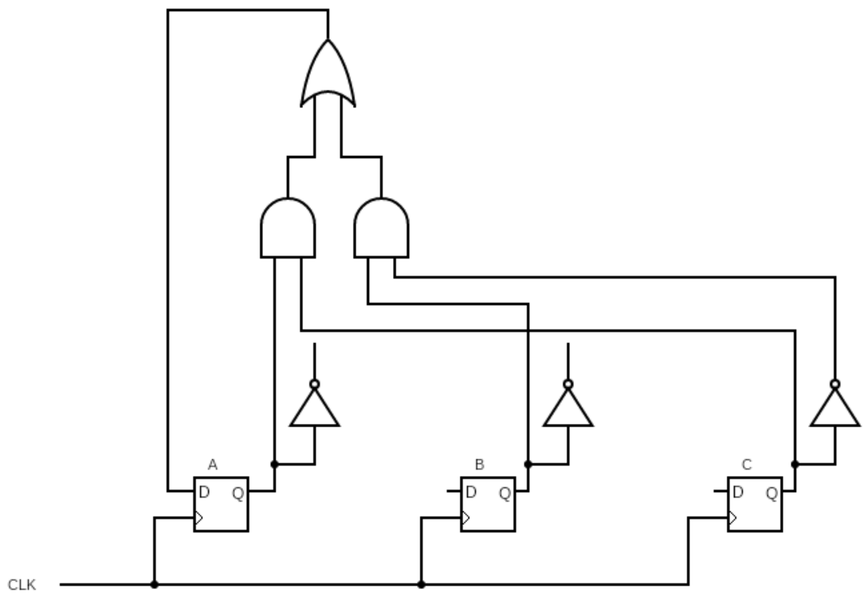
Transition and output table:

PS (ABC)	NS (ABC), O/P	
	in	in
	0	1
000	(000, 0)	(001, 0)
001	(001, 0)	(011, 0)
010	(010, 0)	(110, 0)
011	(011, 0)	(010, 0)
100	(100, 0)	(000, 1)
101	(101, 0)	(100, 0)
110	(110, 0)	(111, 0)
111	(111, 0)	(101, 1)

(b) K-map for A(t+1):

		B(t), C(t)			
		00	01	11	10
A(t)	0	0	0	0	1
	1	0	1	1	1

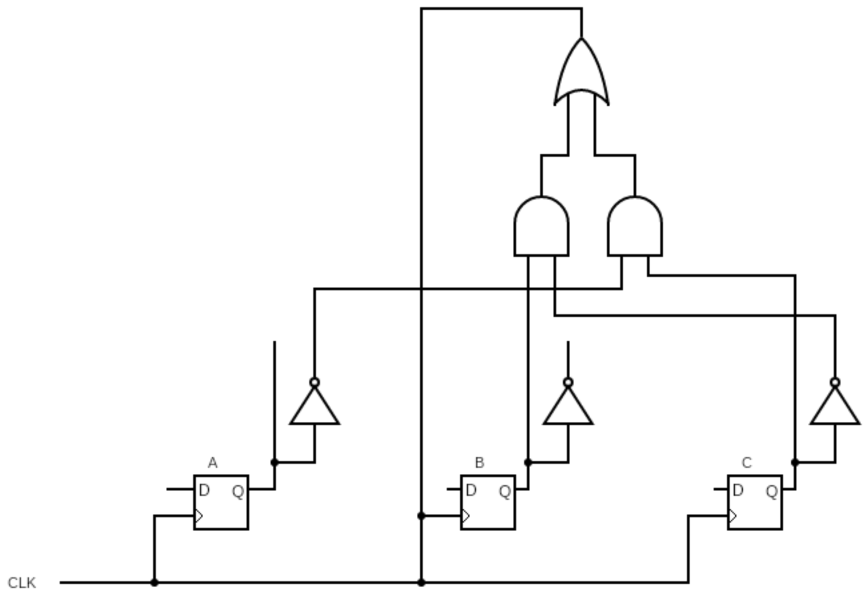
$$A(t+1) = A(t)C(t) + B(t)\bar{C}(t)$$



K-map for B(t+1):

		B(t), C(t)			
		00	01	11	10
A(t)	0	0	1	1	1
	1	0	0	0	1

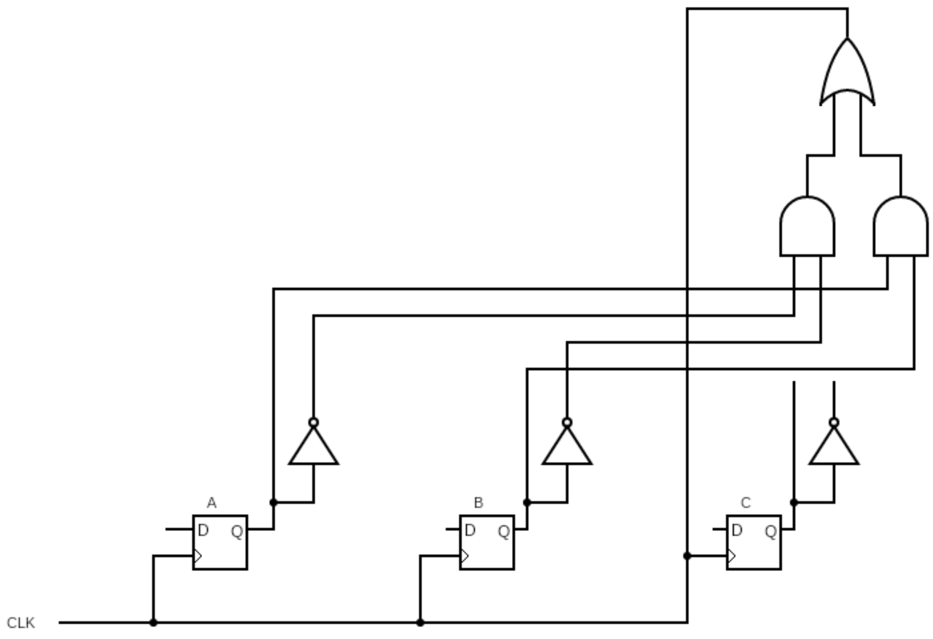
$$B(t+1) = \bar{A}(t)C(t) + B(t)\bar{C}(t)$$



K-map for C(t+1):

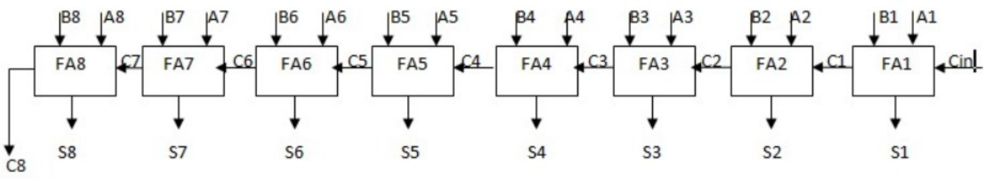
		B(t), C(t)			
		00	01	11	10
A(t)	0	1	1	0	0
	1	0	0	1	1

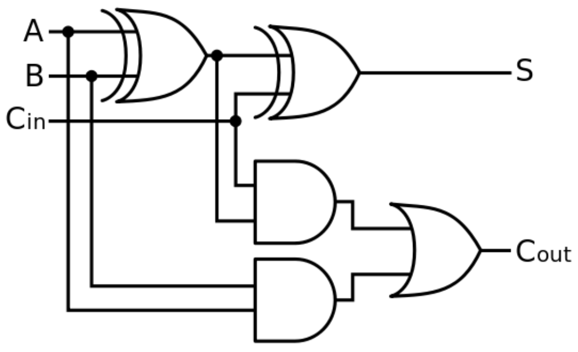
$$C(t+1) = A\bar(t)B\bar(t) + A(t)B(t)$$

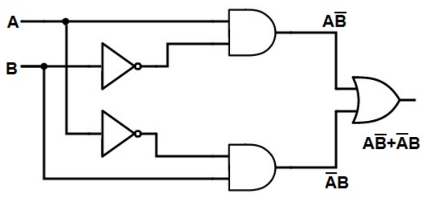


Q2. 8-bit Adder/Subtractor:

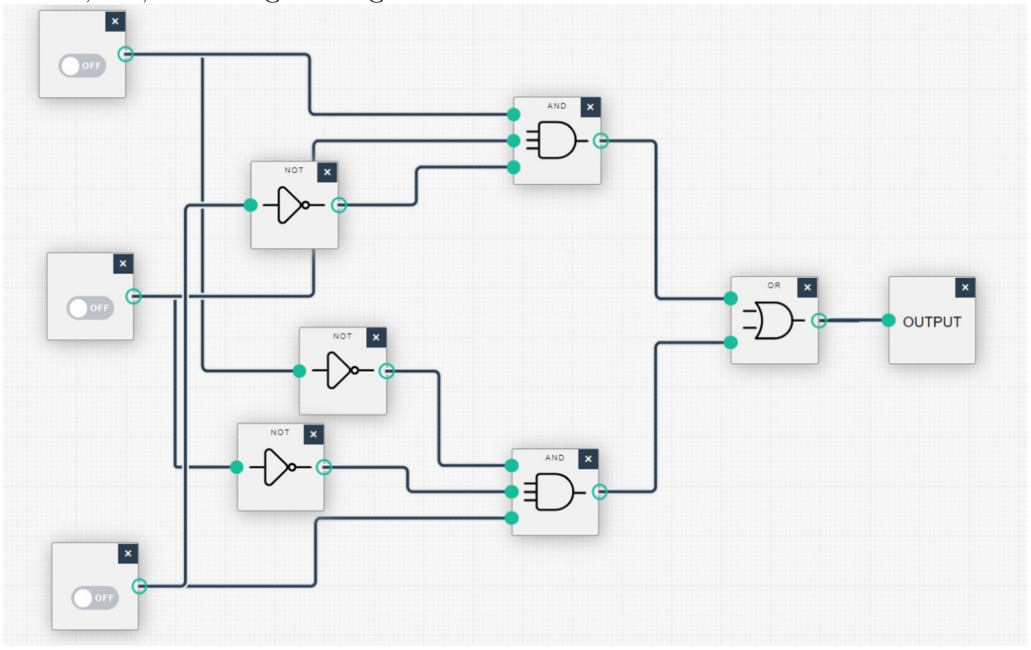
A+B=A+B
A-B=A+(-B)
The 8-bit Adder/Subtractor uses a 1-bit adder/subtractor to implement itself. The 1-bit adder/subtractor has 4 inputs: the first bit, the second bit, opcode, initial carry. The logic is that the second bit will be x-or'ed with opcode so that subtraction operation would imply reversal of the second bit. The 8-bit Adder/Subtractor then passes the two bits along with an initial carry equal to the opcode so that the conversion of B to its 2's complement -B in case of subtraction is complete. The C_out from the 8-bit Adder/Subtractor denotes an overflow if 1. C_out is 1 if A, $(B \oplus opcode)$, sum have the same sign or if A, $(B \oplus opcode)$, sum have the same sign.







In the first circuit, S1, S2.. are sum bits and C8 is carry_out in the code, FA's denote the full adders (one-bit). Each of B's in the circuit represent (B ⊕ opcode). The representative xor circuit is shown in the third diagram. C_out is computed from C8, B8, A8 using the logic stated above the two circuits.



The output (overflow) here is C_out, the inputs from top to bottom are A, B ⊕ opcode and sum.