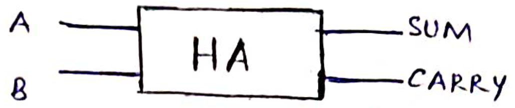
NAME: ANANYA PRASAD
REG NO: 20BCE10093
SUBJECT: ECE2002
SLOT: B11+B12+B13
FACULTY: DR JITENDRA KUMAR

ASSIGNMENT-3

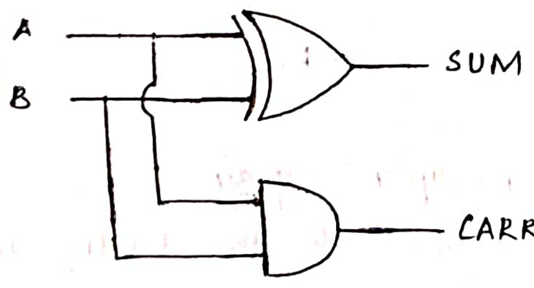
1 Write the function of half adder. Draw and explain various implementations.
A combinational circuit that performs the addition of two bits is called a half adder. There are two input bits and the output consists of sum and carry bits



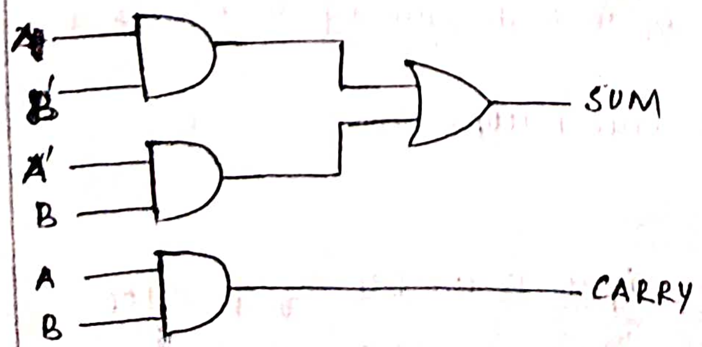
TRUTH TABLE →

A	B	S	C
0	0	0	0
0	1	1	0
1	1	1	0
1	0	0	1

To find the SUM, we have XOR gate ⇒ SUM = A XOR B = $A \oplus B$
To find the carry, we have AND gate ⇒ CARRY = A AND B = AB
LOGIC GATE



LOGIC GATE WITHOUT USING XOR GATE



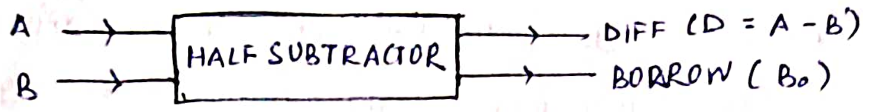
$S = AB' + A'B$
$C = AB$

TRUTH TABE

A	B	S	C
0	0	0	0
0	1	1	0
1	1	1	0
1	0	0	1

2 Explain half subtractor.

Half subtractor is a building block for subtracting two binary numbers. It has two inputs and two outputs. The inputs are two single bit binary numbers and the output are difference and borrow

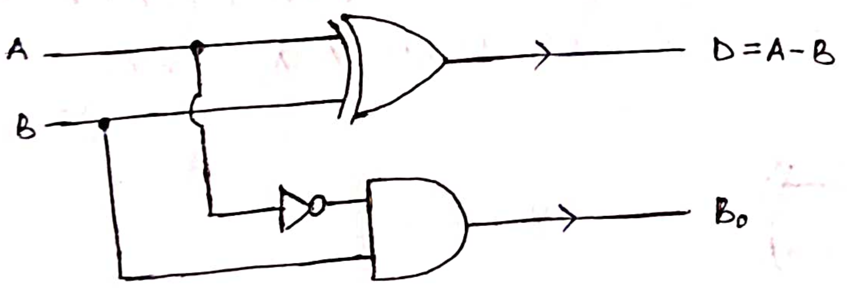


Boolean expression → EXOR for difference
NAND for borrow

TRUTH TABLE

A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

LOGIC GATE



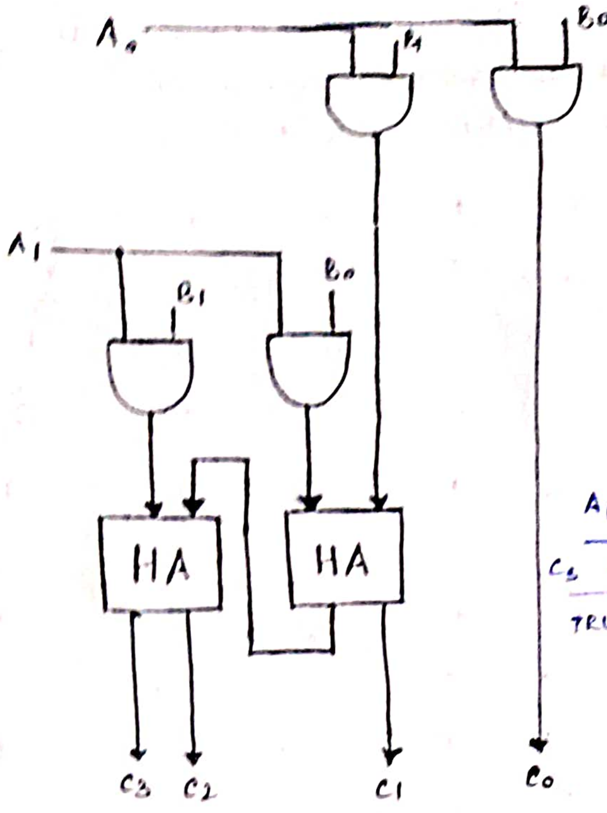
3 What is the function of binary multiplier? Explain

A binary multiplier is used to multiply two binary numbers.
The two numbers are specifically known as multiplicand and multiplier and the result is known as product.
The bit size of the product is equal to the sum of the bit size of multiplier and multiplicand.
For 1 bit → same as ~~binary~~ decimal multiplication

2×2 Bit multiplier

2-Bit Full adder

The partial product of LSBs of inputs is the LSB of product.
The other terms of each partial product should be considered and added using a 2-bit full adder.

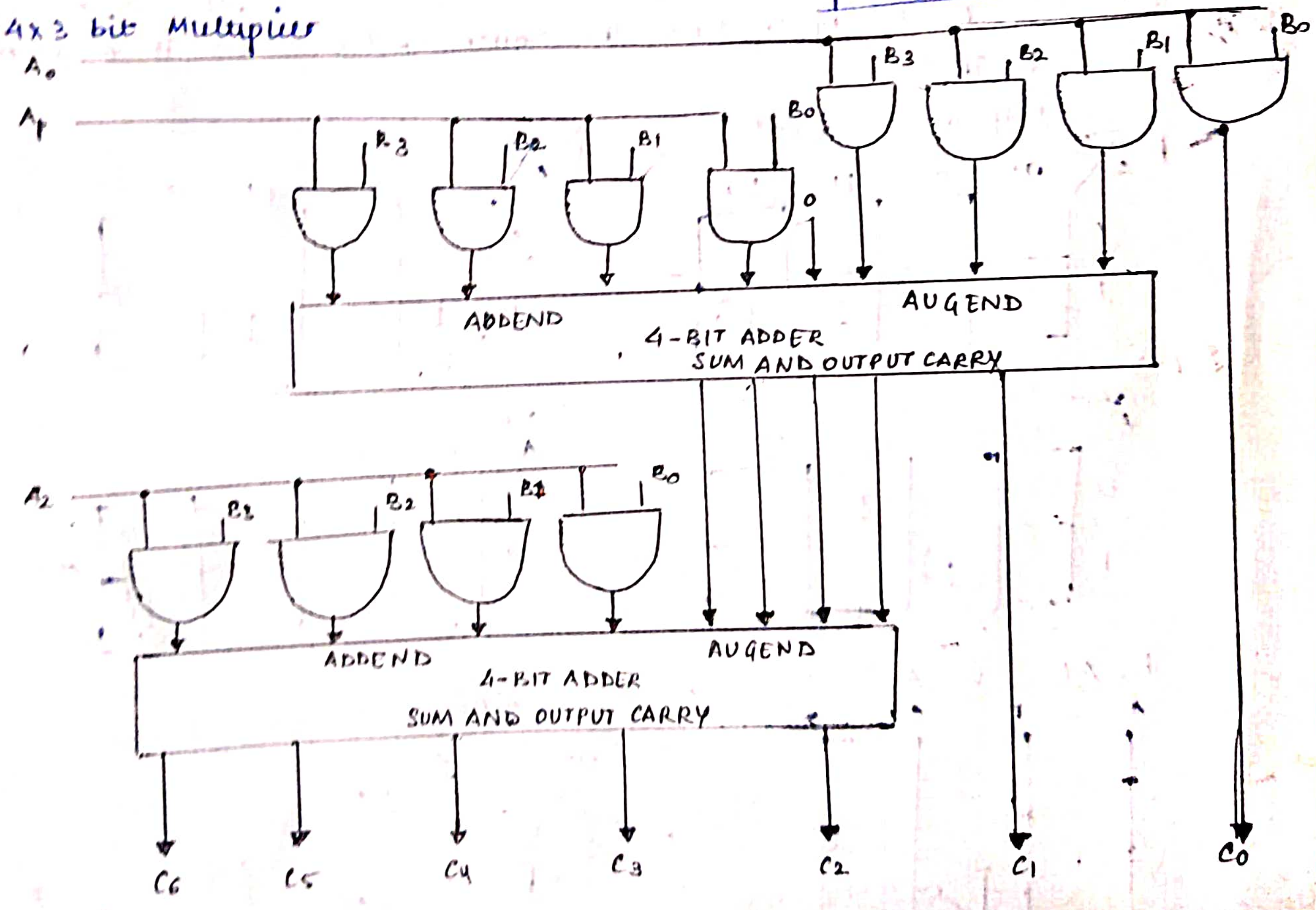


$$\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0b_0 \\
 & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}$$

TRUTH TABLE

MULTIPLIER BITS		MULTIPLICAND	
Y_{i+1}	Y_i	MULTIPLES	IMP.
0	0	0	0
0	1	1	X
1	0	2	Shift left X
1	1	3	(Shift left X)+X

4x3 bit Multiplier



esign a combinational circuit that accepts a 3-bit number and generates an output binary number equal to the square of input number.

A 3-bit register has 2^3 possible combination of inputs.

A	B	C	P	Q	R	S	T	U
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1

Truth table, we have 3bits input and 6 outputs on the left side. We have total of 8 possible inputs. since the square of 8 is 64, so it needs maximum of 6 bits spaces.

For K-Map for P, Q, R and S.

P ⇒

A \ BC	00	01	11	10
0				
1			1	1

Q ⇒

A \ BC	00	01	11	10
0				
1	1	1	1	

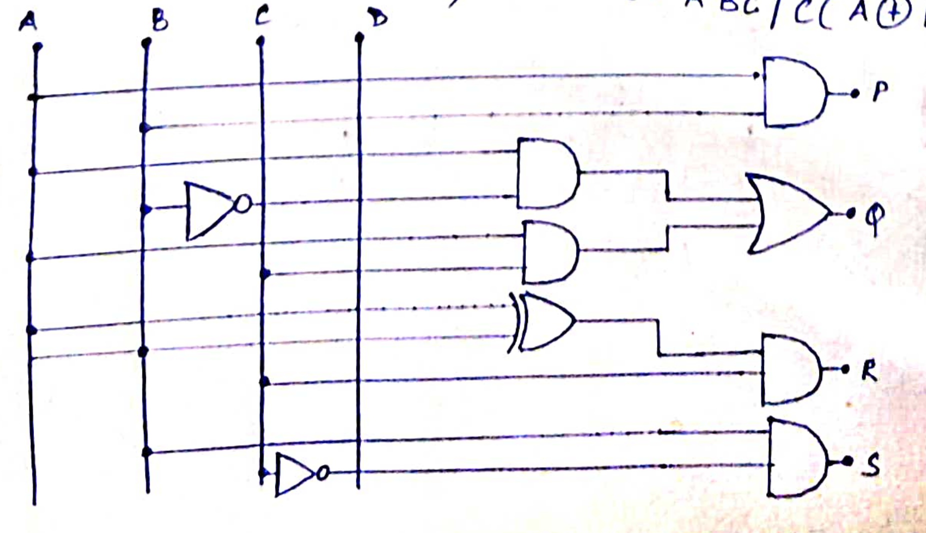
R ⇒

A \ BC	00	01	11	10
0			1	
1		1		

S ⇒

A \ BC	00	01	11	10
0				1
1				1

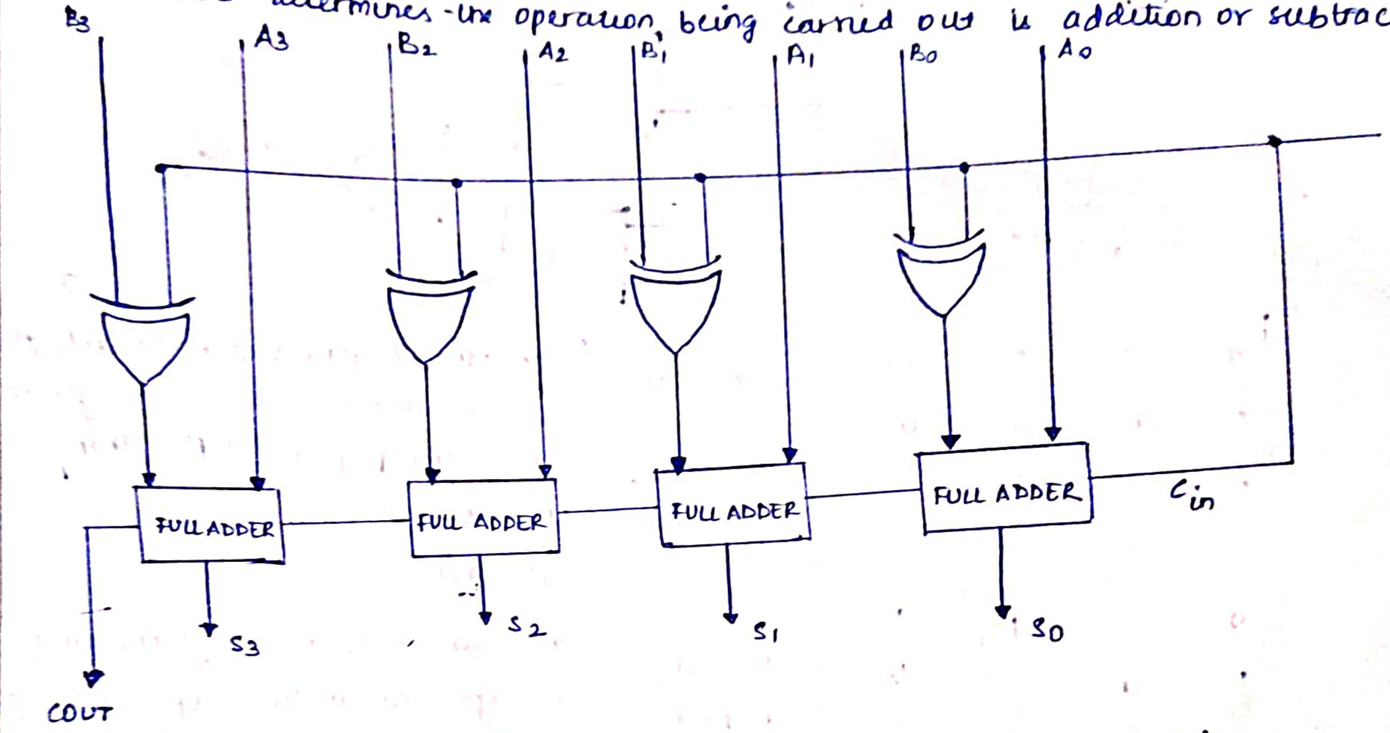
$P = AB$; $Q = AB' + AC$; $R = AB'C + A'BC \,/\, C(A \oplus B)$; $S = BC'$



5 Design a 4-bit adder subtractor circuit and explain operation in details
Let us consider two 4-bit binary numbers A and B as inputs to the digital circuit for the operation with digits

A0 .	A1	A2	A3	for A
B0	B1	B2	B3	for B

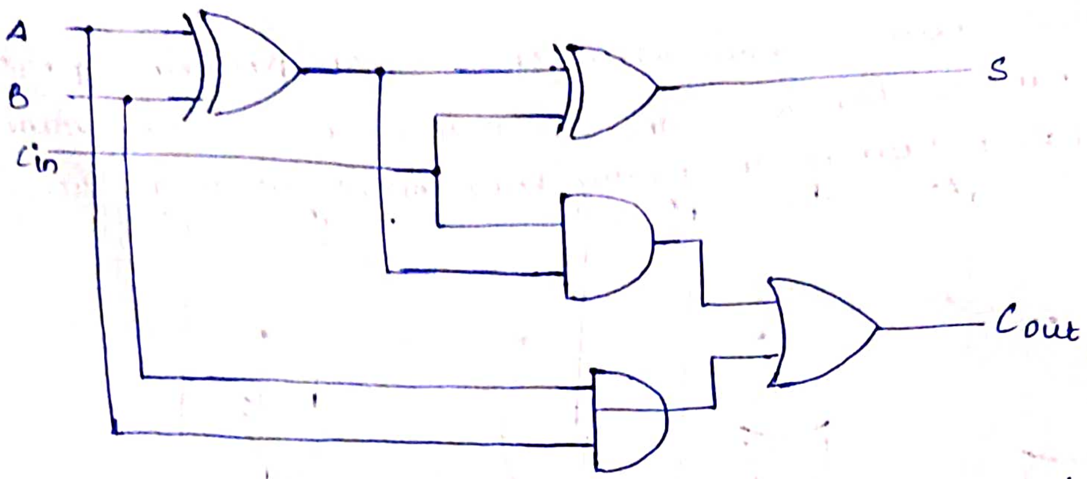
The circuit consists of 4 full adders since we are performing operation on 4-bit numbers. There is a control line K that holds a binary value of either 0 or 1 which determines the operation being carried out is addition or subtraction



As shown, the first full adder has control line directly as its input (input carry C0) The input A0 (The least significant bit of A) is directly input in the full adder. The third input is the EXOR of B0 and K (S in fig, but do not confuse it with sum - S). The two outputs produced are the sum / difference (S0) and carry (C1)
If the value of K (control line) is 1, output of B0 (exor) K = B0' (complement B0)
Thus the operation would be ~~A⊕B0~~ A+(B0'). Now 2's complement subtraction for two numbers A and B is given by A+B'. This suggests that when K=1, the operation being performed on the four bit numbers is subtraction.
Similarly, if the value of K=0, B0 (exor) K = B0. The operation is A + B which simple binary addition. This suggests that when K=0, the operation being performed on the four bit numbers in addition
Then C0 is serially passed to the second full adder as one of its outputs.
The sum / difference S0 is recorded as the least significant bit of the sum/diff.
A1, A2, A3 are direct inputs to the second, third and fourth full adder. Then carry C1, C2 are serially passed to the successive full adder as one of the inputs.
C3 becomes the total carry to the sum / difference. S1, S2 and S3 are recorded to form the result with S0.

6 Draw and explain the working of carry look ahead adder.

A carry look ahead adder reduces the propogation delay by introducing more complex hardware. In this design, the ripple carry design is suitably transformed such that the carry logic over fixed groups of bits of the adder is reduced to two-level logic.



A	B	C	C+1	Condition
0	0	0	0	No carry Generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No Carry Propagate
1	0	0	0	
1	0	1	1	
1	1	0	1	carry generate
1	1	1	1	

Consider the full adder circuit shown has two variables as 'carry generate' G_i and 'carry propagate' P_i, then,

$P_i = A_i \oplus B_i$

$G_i = A_i B_i$

The sum output and carry output can be expressed in terms of carry generate G_i and carry propagate P_i as.

$S_i = P_i \oplus C_i$

$C_{i+1} = G_i + P_i C_i$

where G_i produces the carry when both A_i, B_i are 1 regardless of the input carry. P_i is associated with the propagation of carry from C_i to C_{i+1}.

The carry output Boolean function of each stage in a 4 stage carry look ahead adder can be expressed as

$C_1 = G_0 + P_0 C_{in}$

$C_2 = G_1 + P_1 C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{in}$

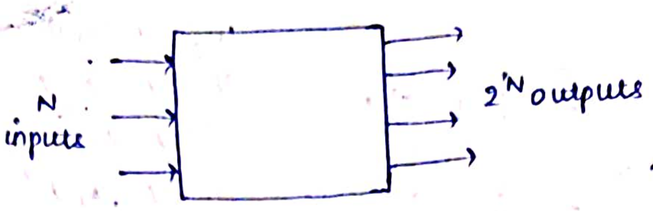
$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$

$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 + P_3 P_2 P_1 P_0 G_{in}$

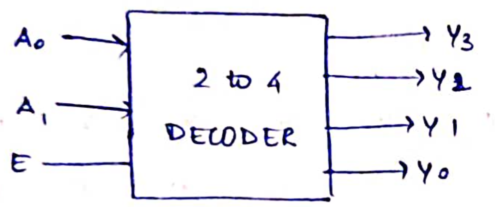
From the above Boolean equations we can observe that C_4 does not have to wait for C_3 and C_2 to propagate but C_4 is propagated at the same time as C_3 and C_2. since the Boolean expression for each carry output is the sum of products so these can be implemented with one level of AND gates followed by an OR gate.

7 Explain the functionality of a decoder

It is a combinational circuit that converts n lines of input into 2^n lines of output. At a time, only one input line is activated for simplicity. The produced 2^n-bit output code is equivalent to the binary information.



There are various types of decoders, to demonstrate, lets discuss 2 to 4 line decoder. There are three inputs A_0, A_1 and E and four outputs Y_0, Y_1, Y_2, Y_3. For each combination of inputs, when the enable E is set to 1, one of these four outputs will be 1. The block diagram and the truth table of 2 to 4 line decoder →



ENABLE	INPUTS		OUTPUTS			
E	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

The logical expression of the term Y_0, Y_1, Y_2 and Y_3 are as follows:

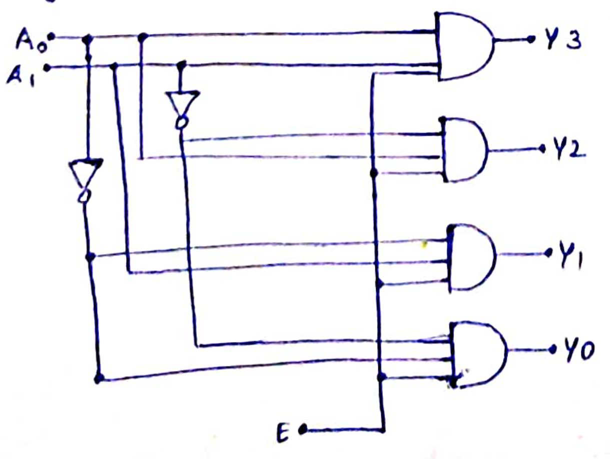
$Y_3 = E.A_1.A_0$

$Y_2 = E.A_1.A_0'$

$Y_1 = E.A_1'.A_0$

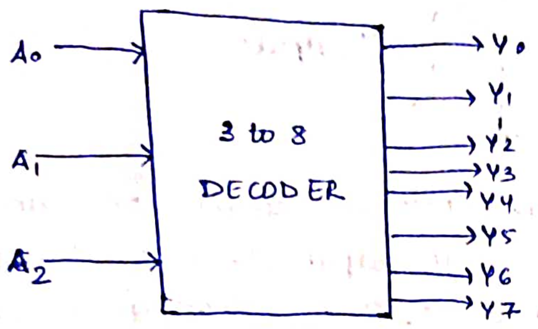
$Y_0 = E.A_1'.A_0'$

Logical circuit.



8 With a neat diagram explain 3-8 decoder.

The 3-8 line decoder is also called Binary to Octal Decoder. There are a total of eight outputs, i.e, $Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7$ and three inputs A_0, A_1, A_2. This circuit has an enable E. When the enable 'E' is set to 1. The block diagram and Truth table →



Logical expressions →

$Y_0 = A_0'.A_1'A_2'$

$Y_1 = A_0.A_1'.A_2'$

$Y_2 = A_0'.A_1.A_2'$

$Y_3 = A_0.A_1A_2'$

$Y_4 = A_0'.A_1'A_2$

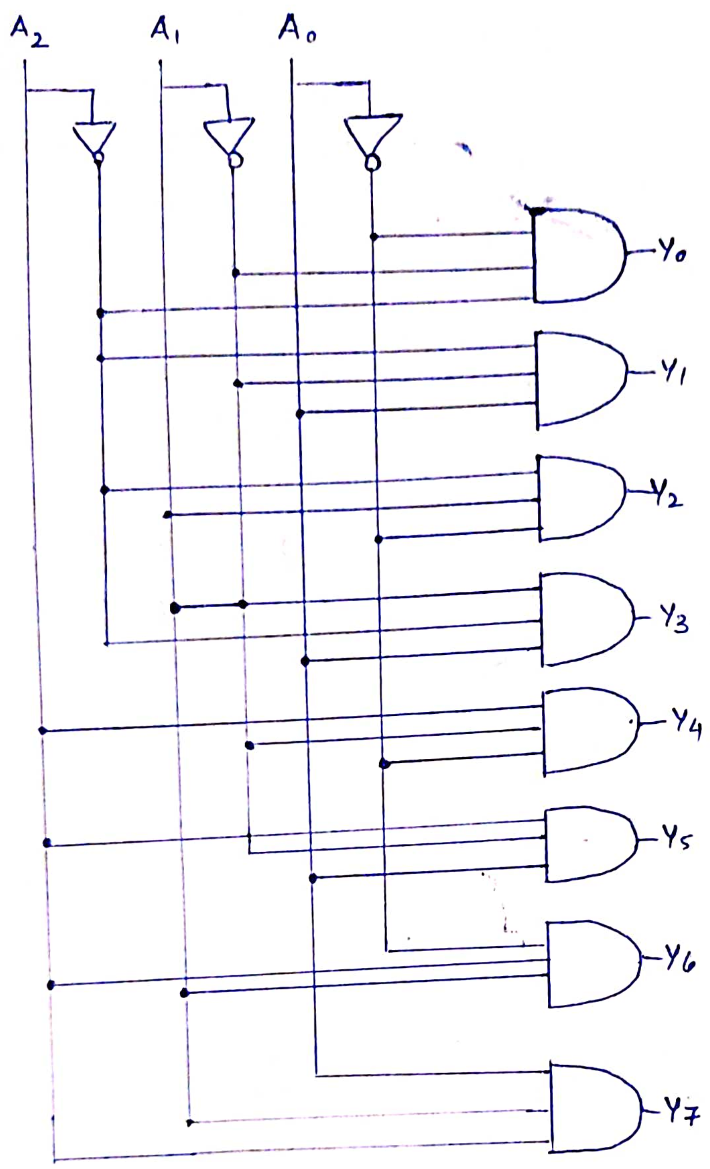
$Y_5 = A_0.A_1'A_2$

$Y_6 = A_0'.A_1.A_2$

$Y_7 = A_0.A_1A_2$

TRUTH TABLE :

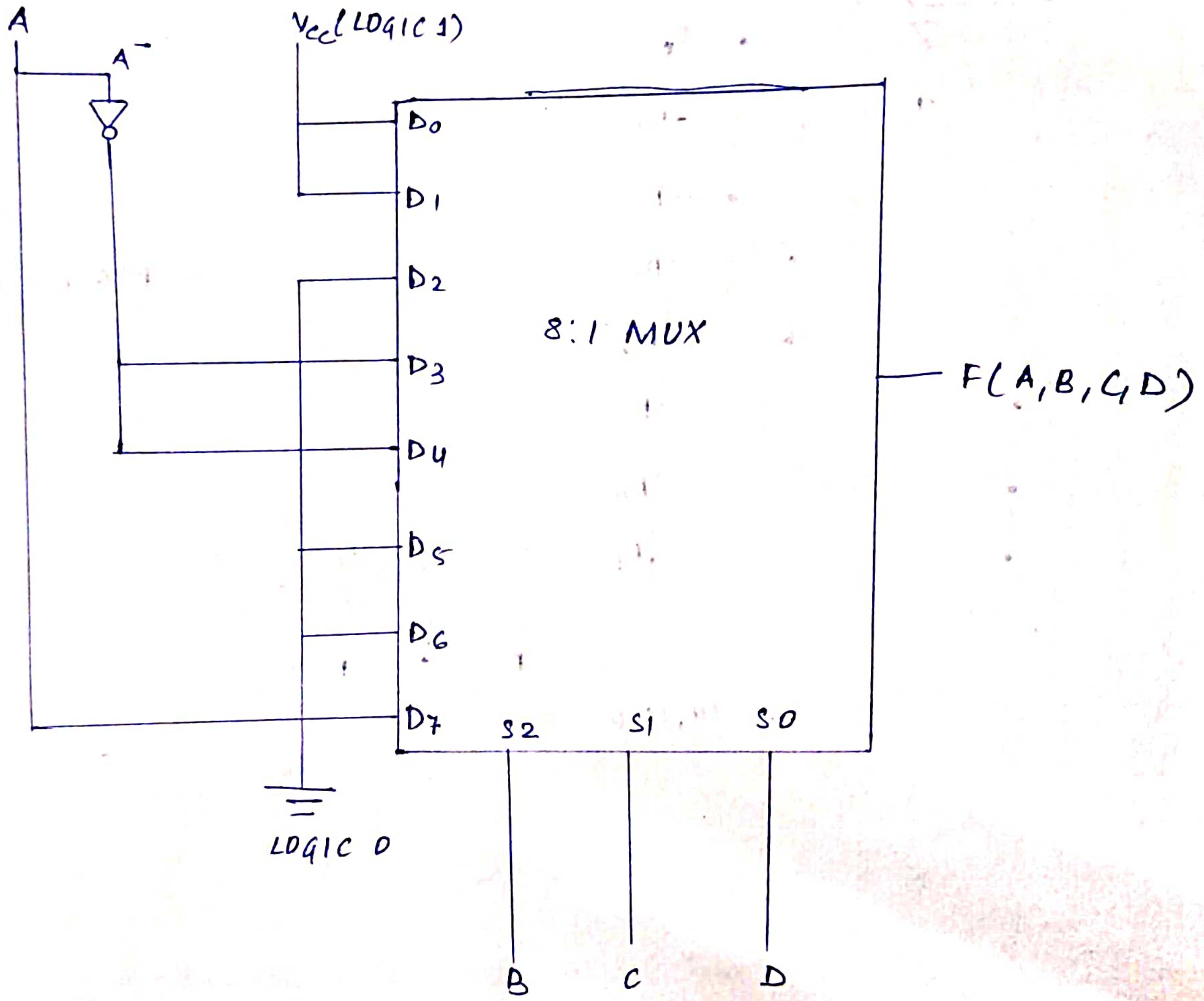
ENABLE	INPUTS			OUTPUTS							
E	A_0	A_1	A_2	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0


A_2
A_1
A_0
Y_0
Y_1
Y_2
Y_3
Y_4
Y_5
Y_6
Y_7

9 Implementa using a 8:1 MUX

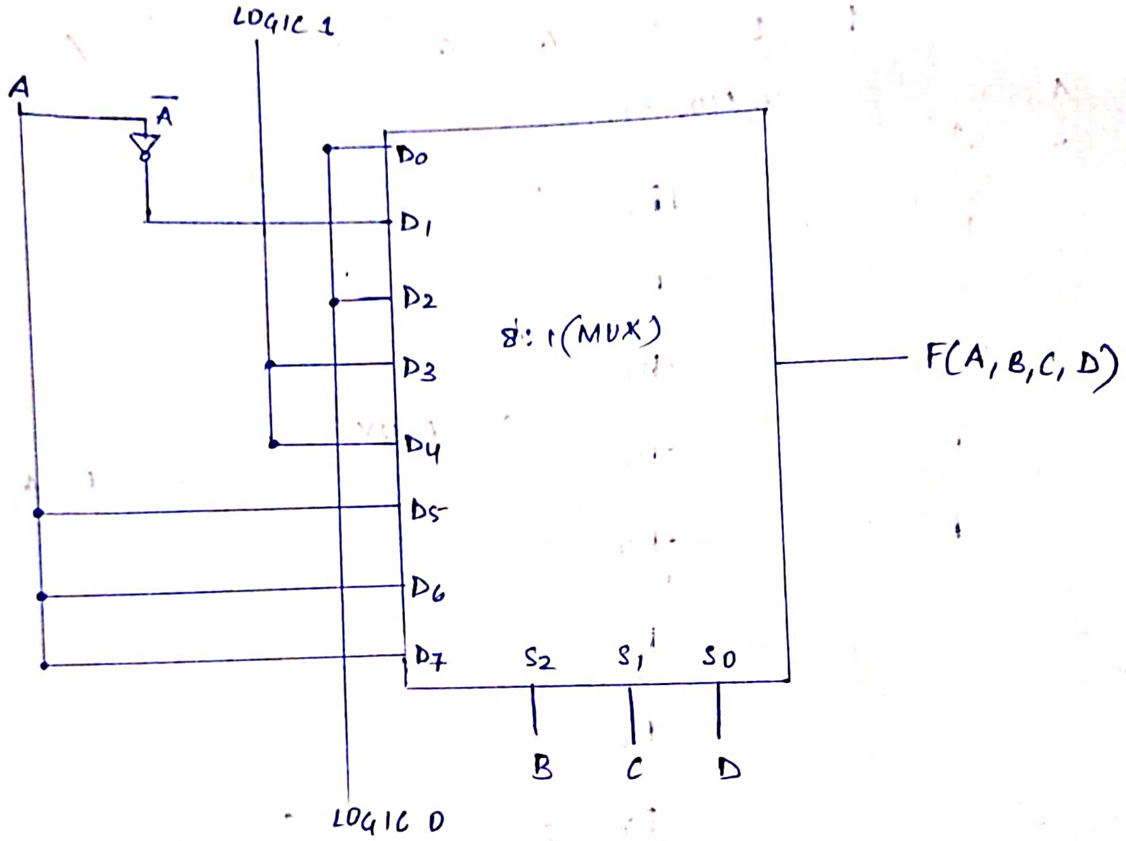
(i) $F(A,B,C,D) = \Sigma(0,1,3,4,8,9,15)$

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	$\textcircled{0}$	$\textcircled{1}$	2	$\textcircled{3}$	$\textcircled{4}$	5	6	7
A	$\textcircled{8}$	$\textcircled{9}$	10	11	12	13	14	$\textcircled{15}$
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A



(iv) $F(A,B,C,D) = \Sigma(1,3,4,11,12,13,14,15)$

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	①	2	③	④	5	6	7
A	8	9	10	⑪	⑫	⑬	⑭	⑮
	0	$\bar{A}$	0	1	1	A	A	A

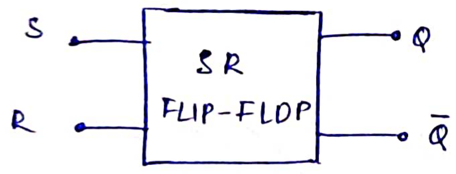


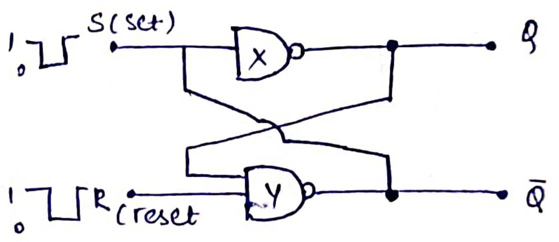
The basic Flipflop circuit with NAND GATE is SR FLIPFLOP.

This simple flipflop is basically a one-bit memory bistable device that has two inputs one which will 'SET' the device (labelled S) and the other which 'reset' the device (labelled R)

SR stands for 'Set-Reset'. The reset input resets the flipflop back to its original state with an output Q that will be either at a 'logic level 1' or logic "0" depending upon set / reset condition

The simplest way to make an SR Flipflop, just connect together a pair of cross coupled 2-input NAND gates, to form a set / reset bistable also known as an active LOW SR NAND gate latch, so there is feedback from each output to one of the other NAND gate inputs. The device has two inputs, one is set (S), other reset (R) and corresponding outputs Q and its complement $\bar{Q}$.





TRUTH TABLE

	S	R	Q		
Set	1	0	0	1	Set $\bar{Q}$ >1
	1	1	0	1	No change
Reset	0	1	1	0	Reset $\bar{Q}$ > 0
	1	1	1	0	no change
Invalid	0	0	1	1	Invalid