

Digital Electronics

- Core subjects for CS/IT Students
- In GATE 7-8 Marks out of 100 Marks, and 5-6 questions on an average
- Most questions are Numerical
- Needs less time, good scoring
- Not asked in Industry

Syllabus

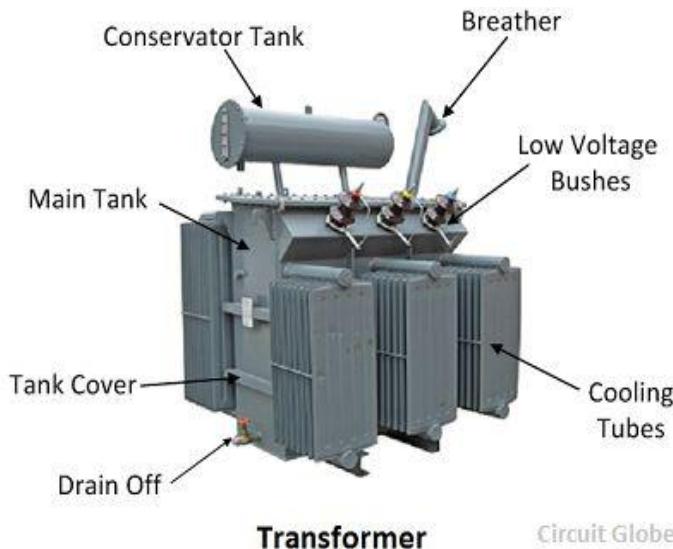
Section 2: Digital Logic

Boolean algebra. Combinational and sequential circuits. Minimization. Number representations and computer arithmetic (fixed and floating point).

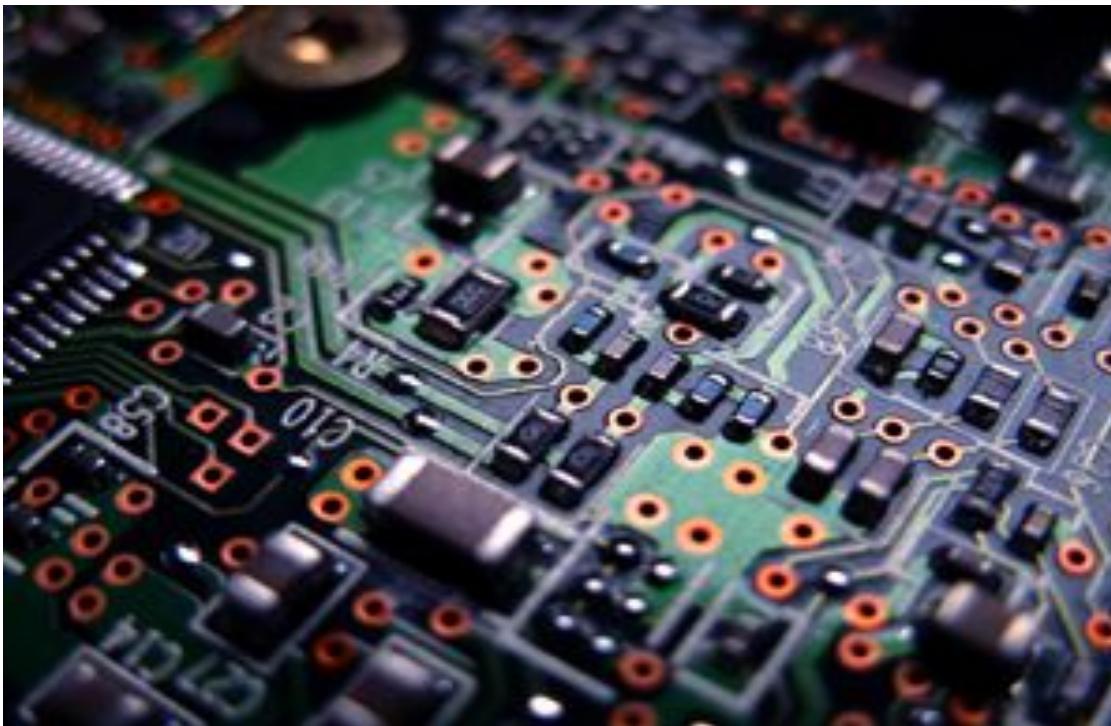
- Understanding Digital Electronics (History and Motive)
- Boolean Algebra Laws and Logic Gates
- Boolean Expression (SOP and POS) (Minimization)
- Combinational Circuit
- Sequential Circuit
- Number System and Number Representation

Basics

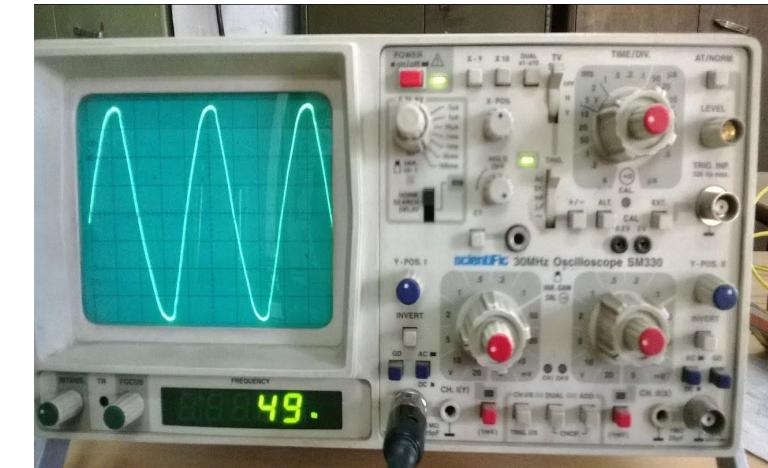
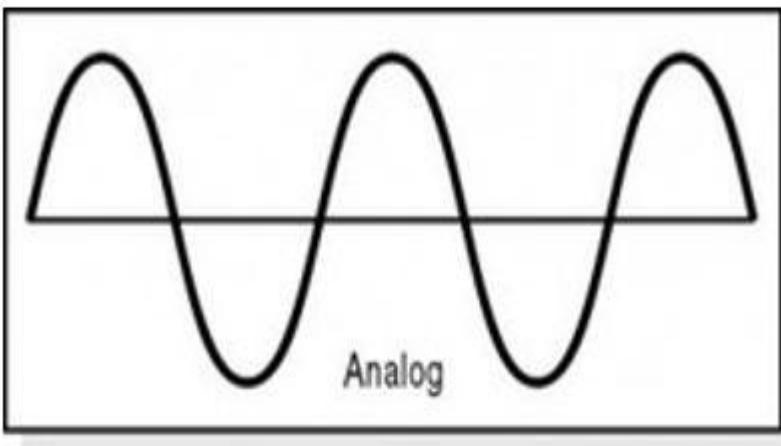
- **Electrical engineering** is a professional engineering discipline that generally deals with the study and application of electricity, electronics and electromagnetism and heavy voltage devices like transformers, motors etc.



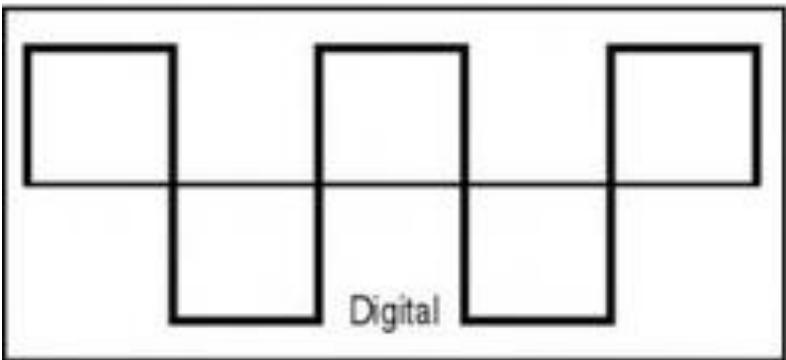
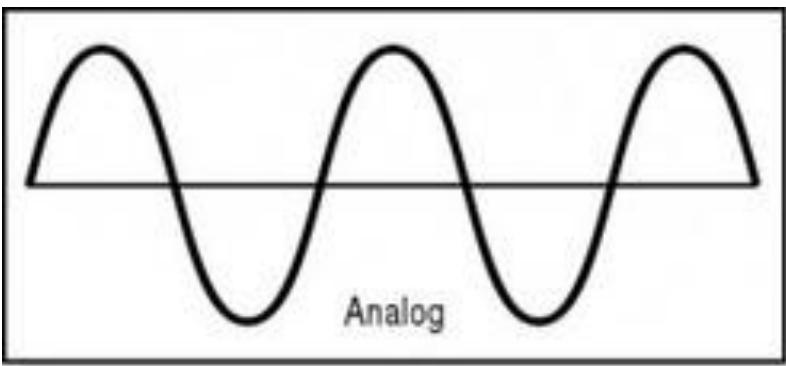
- **Electronic engineering** is an electrical engineering discipline where we work on low voltage devices (such as semiconductor devices, especially transistor, diodes and integrated circuits) to design electronic circuits, VLSI devices and systems.



- Electronic systems are generally of two types –
 - **Analog system** in an analog representation, a continuous value is used to denote the information. Analog signal is defined as any physical quantity which varies continuously with respect to time. e.g. amplifier, cro, ecg. Watch, radio.



- **Digital system** – the information is denoted by a finite sequence of discrete value or digits. Digital signal is defined as any physical quantity having discrete values. E.g. digital watch, calculator, bp machine, thermometer etc.
- Early digital computers were used for numeric computation. The discrete elements were the digits, from this application, the term digital computer emerged.



Advantage of Digital system

- 1) Digital system are easy to design because they do not require sound engineering and mathematical knowledge, uses only switching circuit.
- 2) Storage for long time and processing of information is easy.
- 3) They provide better accuracy and precision compared to analog devices, as digital devices are less affected by noise, sound, electric or magnetic field etc.
- 4) Better modularity and easy fabrication. It means the of digital devices are very small compared to analog devices. E.g. integrated circuits.
- 5) Less cost.

Disadvantage of Digital system

- Only analog signal is available in the real world, so an extra hardware is required to convert this analog signal to digital signal by using analog to digital converter.

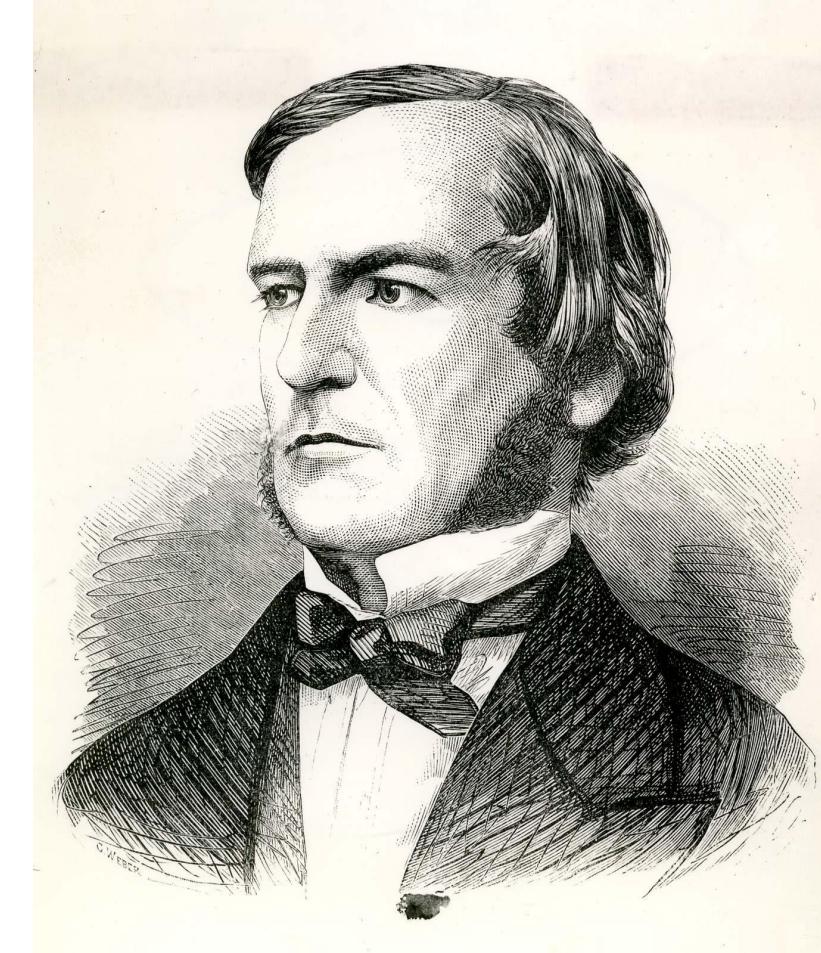
Conclusion

- Digital systems have such a prominent role in everyday life that we refer to the present technological period as digital age.
- Digital system are used in communication, business, transactions, traffic control, space guidance, medical treatment, weather forecasting, the internet, and many other commercial, industrial and scientific enterprises.



Boolean algebra

- The signal in most present day electronic digital system uses just two discrete values and are therefore said to be binary.
- Boolean algebra was introduced by George Boole in his first book The Mathematical Analysis of Logic (1847), and set forth more fully in his An Investigation of the Laws of Thought (1854).
- George boole introduced the concept of binary number system in the studies of the mathematical theory of logic and developed its algebra known as Boolean algebra.



Boolean algebra

- In mathematics and mathematical logic, Boolean algebra is the branch of algebra in which the values of the variables are the truth values true and false, usually denoted 1 and 0 respectively.
- Instead of elementary algebra where the values of the variables are numbers, and the prime operations are addition and multiplication. The main operations of Boolean algebra are
 - The conjunction and denoted as \wedge
 - The disjunction or denoted as \vee
 - The negation not denoted as \neg

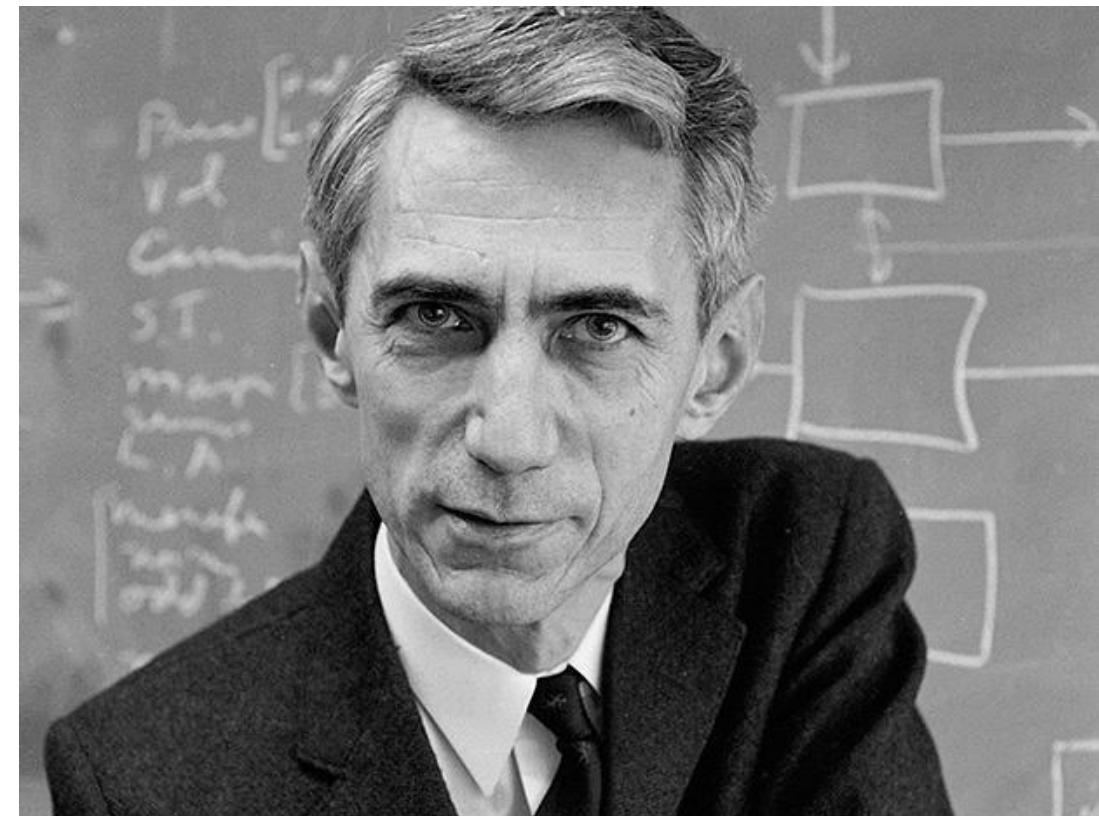
Turing Machine

- *The Church-Turing thesis states that any algorithmic procedure that can be carried out by human beings/computer can be carried out by a Turing machine.(1936)*
- It has been universally accepted by computer scientists that the Turing machine provides an ideal theoretical model of a computer.



Digital System

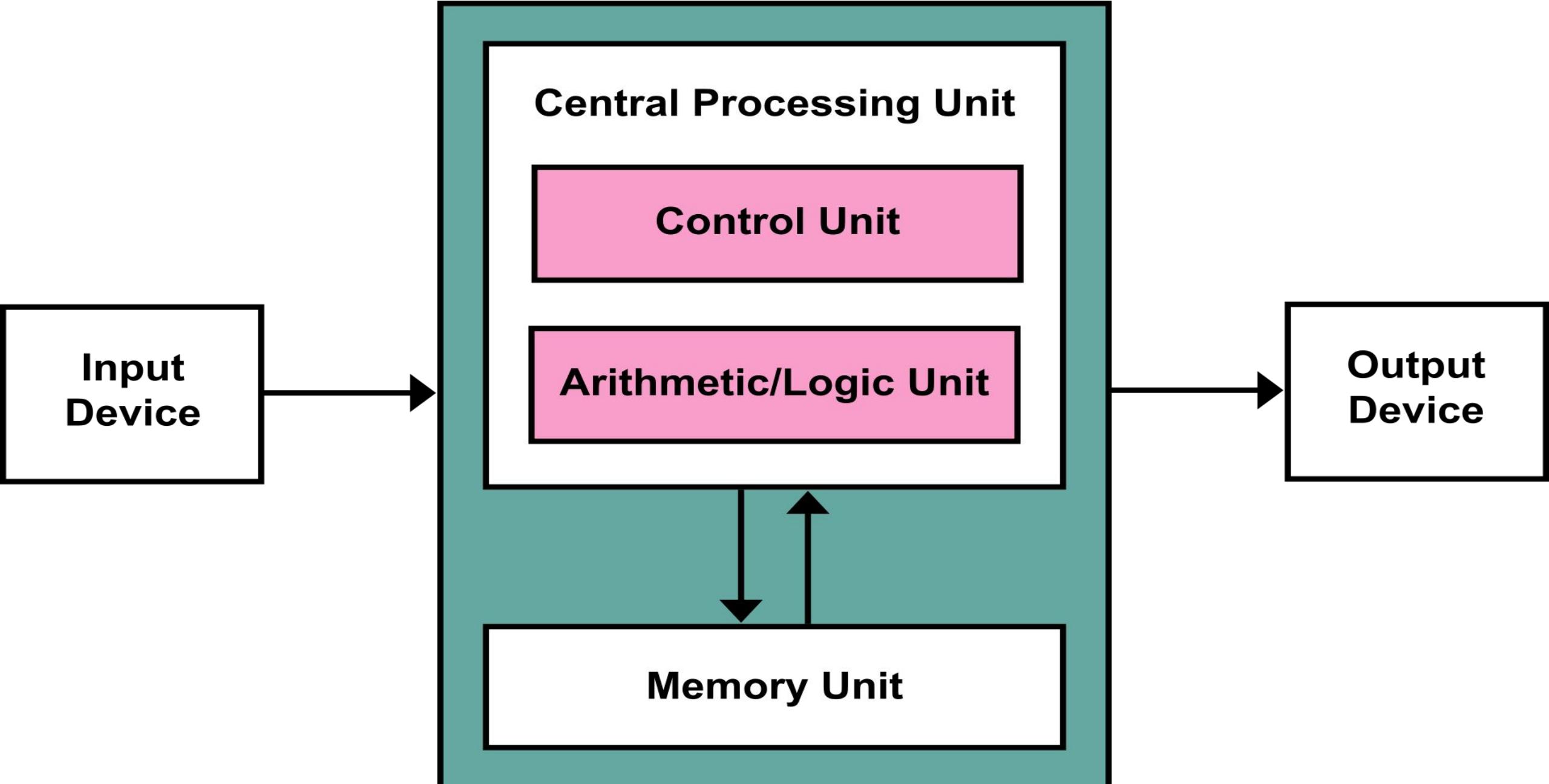
- In the 1930s, while studying switching circuits, Claude Shannon observed that one could also apply the rules of Boole's algebra in this setting, and he introduced switching algebra as a way to analyze and design circuits by algebraic means in terms of logic gates.
- These logic concepts have been adapted for the design of digital hardware since 1938 Claude Shannon (father of information theory), organized and systematized Boole's work.



von Neumann architecture

- The **von Neumann architecture** is a computer architecture based on a 1945 description by John von Neumann. That document describes a design architecture for an electronic digital computer with these components:
 - A processing unit that contains an arithmetic logic unit and processor registers
 - A control unit that contains an instruction register and program counter
 - Memory that stores data and instructions
 - External mass storage
 - Input and output mechanisms





- Digital systems have two logic levels
 - The lower voltage level is called a logic low or logic 0 (0-1 v), which represent digit 0.
 - The higher voltage level is called a logic high or logic 1 (3.5-5 v), which represent digit 1.

Q Design a digital system for a car manufacturing company, where we want to design a warning signal for a car, there are three inputs, lights of the car(L), day or night(D), ignition (on/off).?

Solution

1) Understand the problem- here we will Understand the definition of the problem

Design the truth table –



Light	Day	Engine	Warning
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

2) Write the Boolean expression

- $W(L, D, E) = \sum_m (1, 4, 6, 7)$

- $W(L, D, E) = \prod_M (0, 2, 3, 5)$

Light	Day	Engine	Warning
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

3) Minimize Boolean expression

$$W = a'b'c + ab'c' + abc' + abc$$

	ab	a'b'	a'b	ab	ab'
c	00	01	11	10	
c'	0	0	2	6	4
c	1	1	3	7	5

$$W =$$

Light	Day	Engine	Warning
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

4) Implement the expression using logic gates, draw the implementation using logic gates

$$W = ac' + ab + a'b'c$$

Light	Day	Engine	Warning
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Boolean Algebra Laws

Idempotent Law

$$a \cdot a = a$$

$$a + a = a$$

Associative law

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c$$

$$a + (b + c) = (a + b) + c$$

Commutative law

$$a \cdot b = b \cdot a$$

$$a + b = b + a$$

Distributive law

$$a \cdot (b + c) = a \cdot b + a \cdot c$$

$$a + (b \cdot c) = (a + b) \cdot (a + c)$$

De-Morgan law

$$(a + b)' = a' \cdot b'$$

$$(a \cdot b)' = a' + b'$$

Identity law

$$a + 0 = a$$

$$a \cdot 0 = 0$$

$$a + 1 = 1$$

$$a \cdot 1 = a$$

Complementation law

$$0' = 1$$

$$1' = 0$$

$$a \cdot a' = 0$$

$$a + a' = 1$$

Involution law

$$(a')' = a$$

Q. For a Boolean variable x , which of the following statements is/are FALSE? (Gate 2024 CS) (1 Mark) (MSQ)

(a) $x \cdot 1 = x$

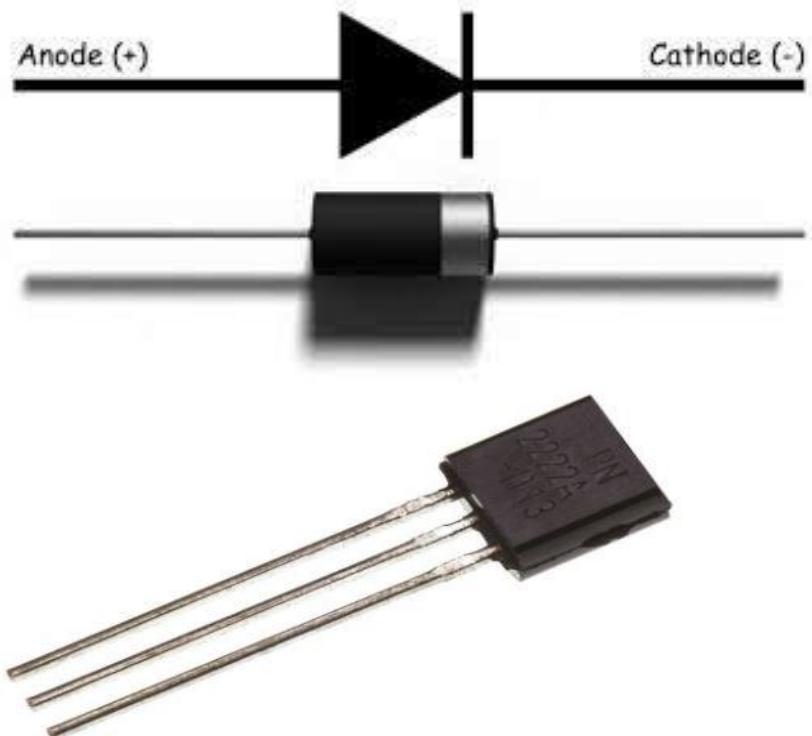
(b) $x + 1 = x$

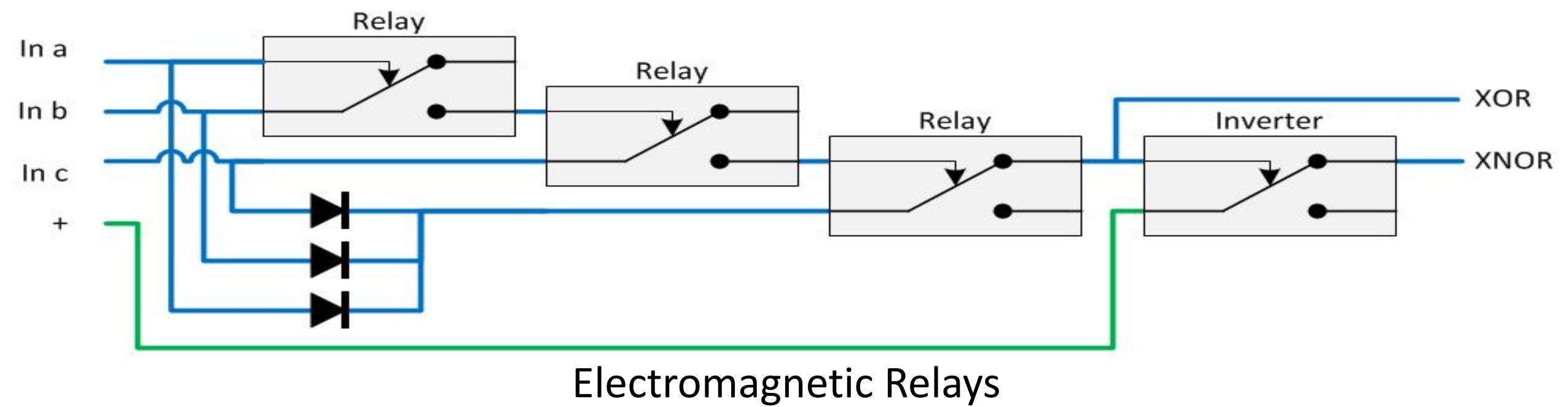
(c) $x \cdot x = 0$

(d) $x + \bar{x} = 1$

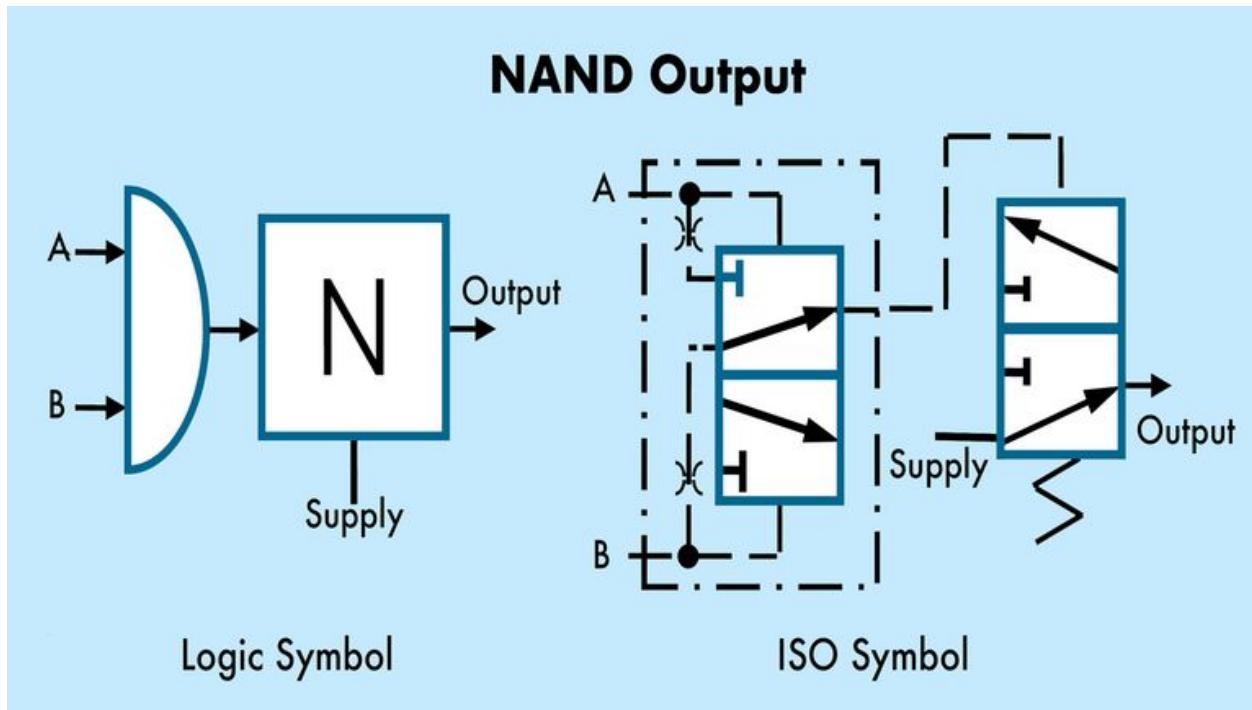
Logic gate

- In electronics, a logic gate is a physical device implementing a Boolean function. Logic gate performs a logical operation on one or more binary inputs signals and produces a single binary output signal. Logic gate is the basic building block from which many kinds of logic circuits can be constructed.
- Logic gates are primarily implemented using diodes or transistors acting as electronic switches, but can also be constructed using vacuum tubes, electromagnetic relays (relay logic), fluidic logic, pneumatic logic, optics, molecules, or even mechanical elements.





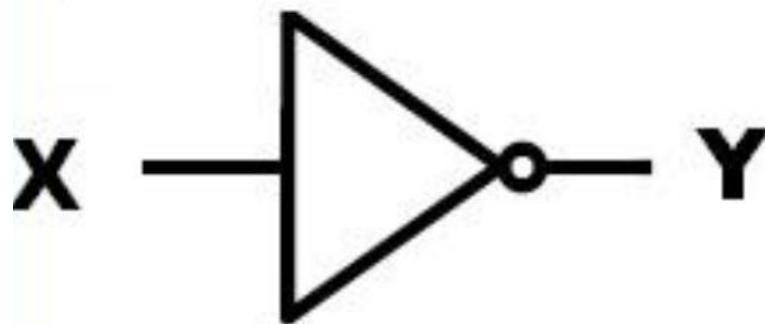
Electromagnetic Relays



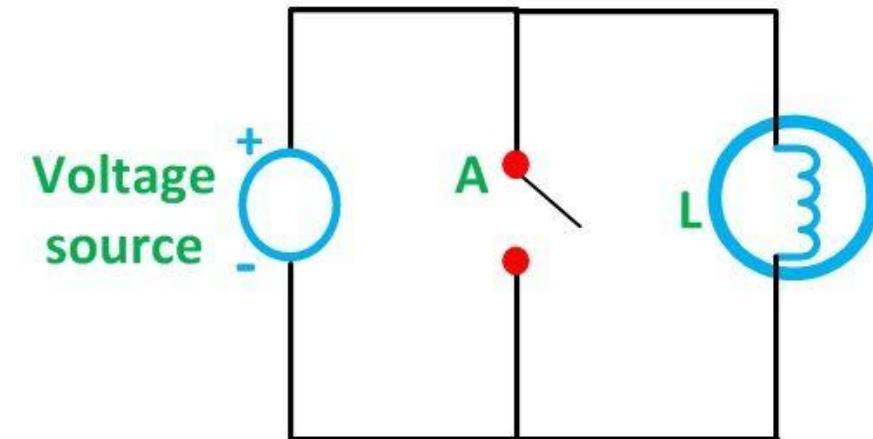
Pneumatic Logic

Not Gate(Inverter)

- It represents not logical operator is also known as inverter, it is a unary operator, which simply complement the input.

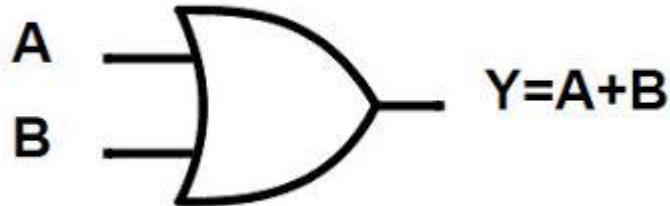


Truth Table	
Input	Output
X	$Y = X'$
0	
1	

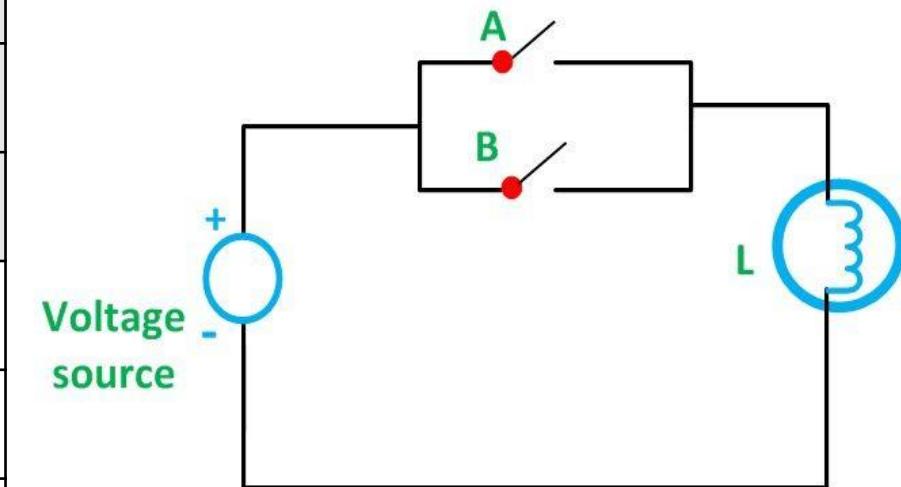


OR Gate

- It is a digital logic gate, that implements logical disjunction. The output will be high if at least one of the input lines is high.



Truth Table		
Input		Output
A	B	$Y = A + B$
0	0	
0	1	
1	0	
1	1	

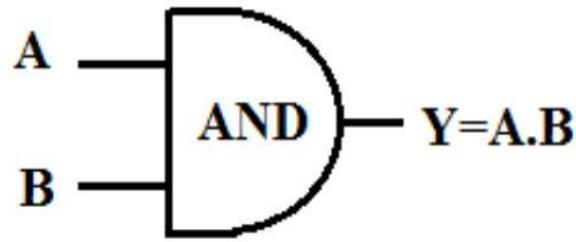


- OR gate satisfy all the 3 rules idempotent, associative, and commutative.

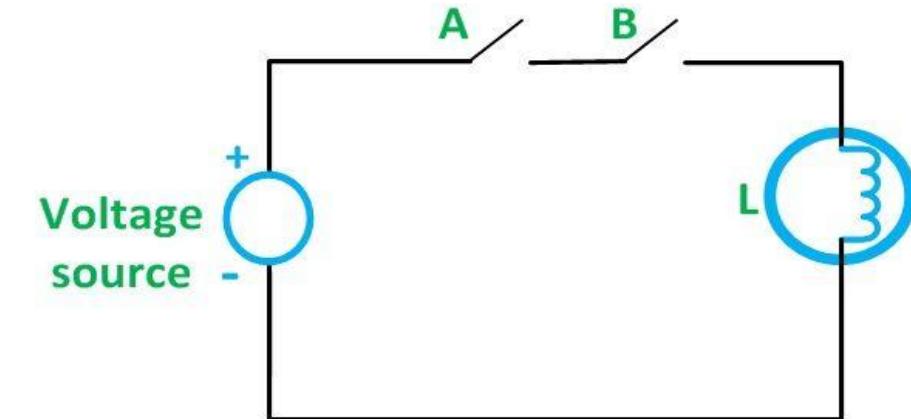
1. **Idempotent Law:** $a + a = a$
2. **Associative law:** $a + (b + c) = (a + b) + c$
3. **Commutative law:** $a + b = b + a$

And Gate

- It is a digital logic gate, that implements logical conjunction. Output will be high if and only if all input are high otherwise low.



Truth Table		
Input		Output
A	B	$Y = A \cdot B$
0	0	
0	1	
1	0	
1	1	



- And gate satisfy all the 3 rules idempotent, associative, and commutative.

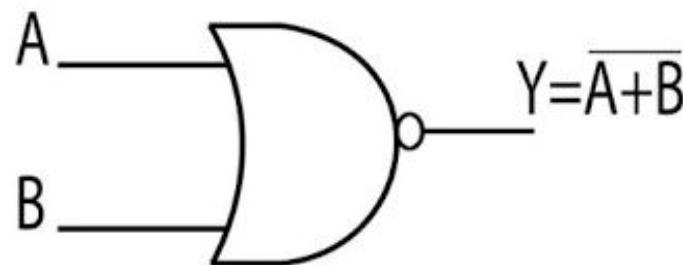
1. **Idempotent Law:** $a \cdot a = a$

2. **Associative law:** $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

3. **Commutative law:** $a \cdot b = b \cdot a$

Nor gate

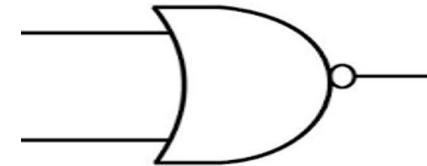
- The output will be high if and only if all inputs are low. Or simply a OR gate followed by an inverter.
- NOR gate is also called universal gate because it can be used to implement any other logic gate. We will cover this property extensively in the next chapter.



Truth Table		
Input		Output
A	B	$Y = (A + B)'$
0	0	
0	1	
1	0	
1	1	

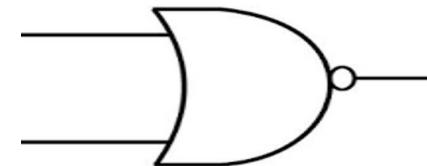
1. NOR with same gives _____

- $(a + a)' =$



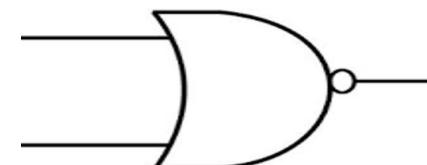
2. NOR with zero gives _____

- $(a + 0)' =$



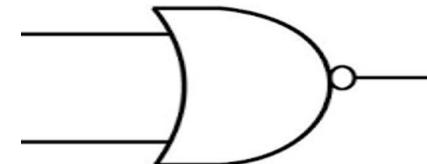
3. NOR with complement gives _____

- $(a + a')' =$



4. NOR with one gives _____

- $(a + 1)' =$



- Idempotent and associative law

- $(a + a)' \boxed{} a$

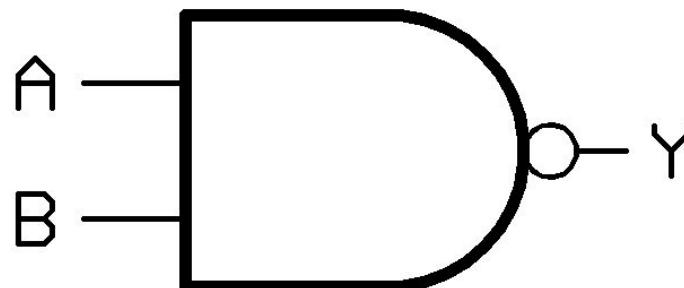
- $((a + b)' + c)' \boxed{} (a + (b + c)')'$

- Commutative law.

- $(a + b)' \boxed{} (b + a)'$

NAND Gate

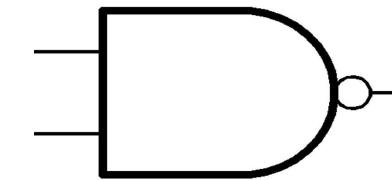
- The output will be low if and only if all inputs are high. Or simply an and gate followed by an inverter
- NAND gate is also called universal gate because it can be used to implement any other logic gate. We will cover this property extensively in the next chapter.



Truth Table		
Input		Output
A	B	$Y = (A \cdot B)'$
0	0	
0	1	
1	0	
1	1	

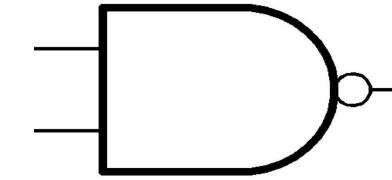
1. NAND with ZERO give _____

- $(a \cdot 0)' =$



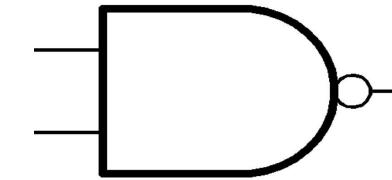
2. NAND with COMPLEMENT give _____

- $(a \cdot a')' =$



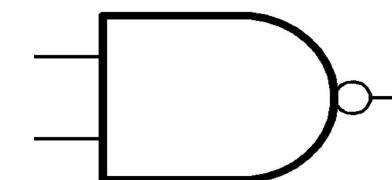
3. NAND with SAME give _____

- $(a \cdot a)' =$



4. NAND with ONE give _____

- $(a \cdot 1)' =$



- NAND with idempotent and associative law

- $(a \cdot a)' \boxed{\quad} a$

- $((a \cdot b)'. c)' \boxed{\quad} (a \cdot (b \cdot c))'$

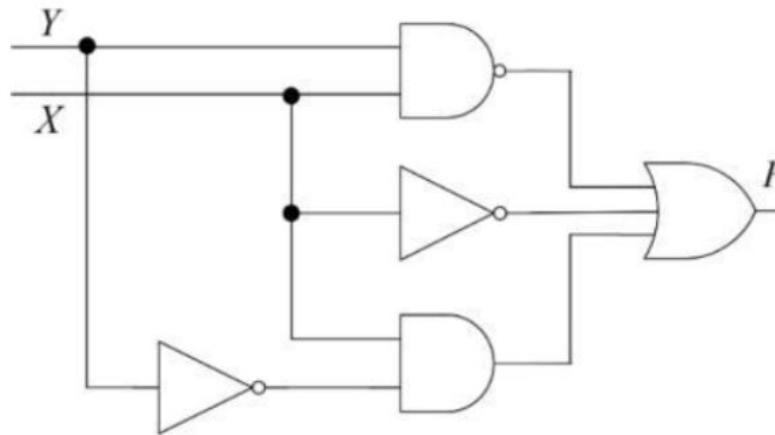
- NAND with commutative law

- $(a \cdot b)' \boxed{\quad} (b \cdot a)'$

Q. Consider the following logic circuit diagram.

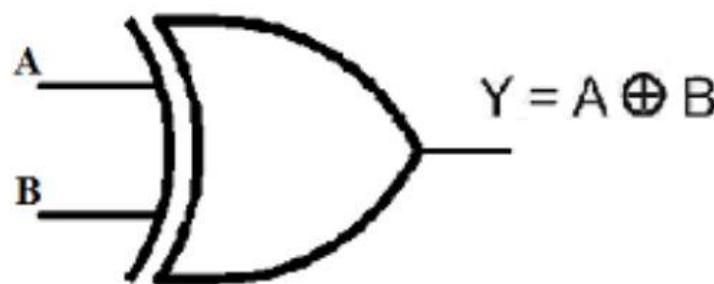
Which is/are the CORRECT option(s) for the output function F ? (GATE 2025)

- A) $X \bar{Y}$
- B) $\bar{X} + \bar{Y} + X \bar{Y}$
- C) $\bar{X} \bar{Y} + \bar{X} + X \bar{Y}$
- D) $X + \bar{Y}$



EX-OR

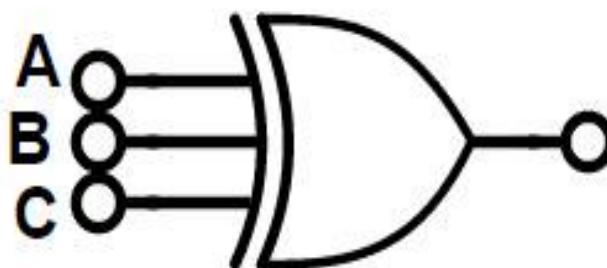
- For two inputs, output will be high if and only if both the input values are different.
- $a \oplus b = a' \cdot b + a \cdot b'$



Truth Table		
Input		Output
A	B	$Y = A \oplus B$
0	0	
0	1	
1	0	
1	1	

EX-OR

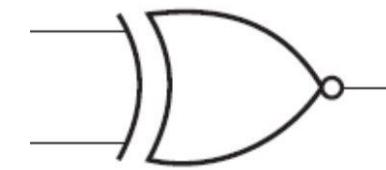
- The XOR gate is a digital logic gate that gives High as output when the number of inputs High are odd.



A	B	C	$a \oplus b \oplus c$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

1. EX-OR with ZERO give _____

- $(a \oplus 0) =$



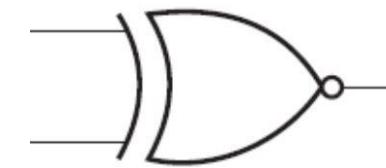
2. EX-OR with ONE give _____

- $(a \oplus 1) =$



3. EX-OR with SAME give _____

- $(a \oplus a) =$



4. EX-OR with COMPLEMENT give _____

- $(a \oplus a') =$



- EX-OR with idempotent law

- $(a \oplus a)$ a

- EX-OR with associative and commutative law.



- $((a \oplus b) \oplus c)$ $(a \oplus (b \oplus c))$

- $(a \oplus b)$ $(b \oplus a)$

Q Consider the Boolean operator with the following properties: (GATE-2016) (1 Marks)

$$x \# 0 = x,$$

$$x \# 1 = x',$$

$$x \# x = 0$$

$$\text{and } x \# x' = 1$$

Then $x \# y$ is equivalent to

- a) $xy' + x'y$
- b) $xy' + x'y'$
- c) $x'y + xy$
- d) $xy + x'y'$

Q The binary operator # is defined by the following truth table. (GATE-2015) (1 Marks)

Which of the following is true about the binary operator # ?

- a) Both commutative and associative
- b) Commutative but not associative
- c) Not commutative but associative
- d) Neither commutative nor associative

p	q	P # q
0	0	0
0	1	1
1	0	1
1	1	0

Q Let \oplus denote the Exclusive OR (XOR) operation. Let '1' and '0' denote the binary constants. Consider the following Boolean expression for F over two variables P and Q.

$$F(P, Q) = ((1 \oplus P) \oplus (P \oplus Q)) \oplus ((P \oplus Q) \oplus (Q \oplus 0))$$

The equivalent expression for F is

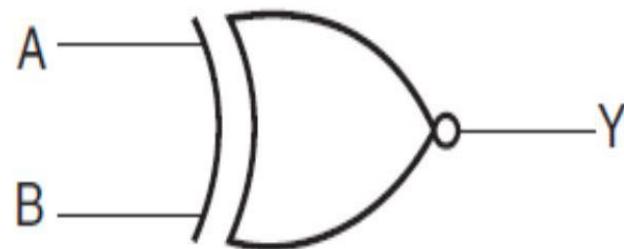
(GATE-2014) (2 Marks)

- (A) $P + Q$
- (B) $(P + Q)'$
- (C) $P \oplus Q$
- (D) $(P \oplus Q)'$

Q A[⊕] P[⊕] A[⊕] P[⊕] B[⊕] P[⊕] B[⊕] P[⊕] B?

EX-NOR

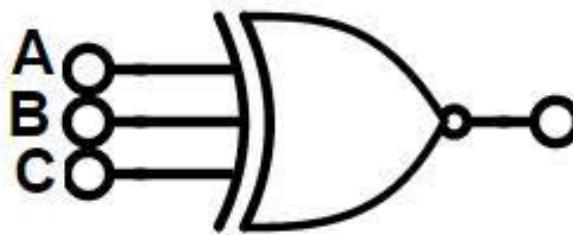
- For two input, output will be high if and only if both the input values are same
- $a \odot b = a'. b' + a. b$



Truth Table		
Input		Output
A	B	$Y = A \odot B$
0	0	
0	1	
1	0	
1	1	

EX-NOR

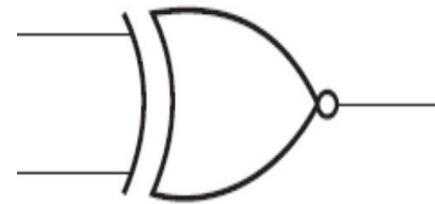
- The EX-NOR gate is a digital logic gate that gives output High when the number of inputs low are even.



A	B	C	$a \odot b \odot c$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

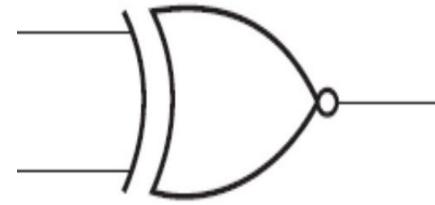
1. EX-NOR with ZERO give _____

- $(a \odot 0) =$



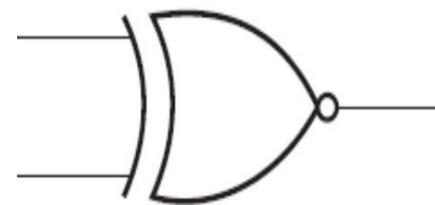
2. EX-NOR with ONE give _____

- $(a \odot 1) =$



3. EX-NOR with SAME give _____

- $(a \odot a) =$



4. EX-NOR with COMPLEMENT give _____

- $(a \odot a') =$

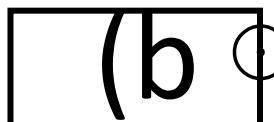


- EX-NOR with idempotent law

- $(a \odot a)$ 

- EX-NOR with associative and commutative law

- $((a \odot b) \odot c)$  $(a \odot (b \odot c))$

- $(a \odot b)$ 

Q Define the connective * for the Boolean variables X and Y as: $X * Y = XY + X' Y'$. Let $Z = X * Y$.
(GATE-2007) (2 Marks)

Consider the following expressions P, Q and R.

P: $X = Y \star Z$

Q: $Y = X \star Z$

R: $X \star Y \star Z = 1$

X	Y	$X \odot Y = Z$

Which of the following is TRUE?

- (A) Only P and Q are valid
- (B) Only Q and R are valid.
- (C) Only P and R are valid.
- (D) All P, Q, R are valid.

A	B	C	$a \oplus b \oplus c$	$a \odot b \odot c$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Conclusion

A	B	C	$a \oplus b \oplus c$	$a \odot b \odot c$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

- Ex-or and ex-nor gate behaves as a complement of each other if the number of input variable is even.
- Ex-or and ex-nor gate behave same if no of input variables are odd.

Relation of EX-OR and EX-NOR

$$a \oplus b = a' \square b = a \square b' = (a \square b)' = (a' \square b')' = a' \square b' = (a' \square b)' = (a \square b)'$$

$$a \odot b = a' \square b = a \square b' = (a \square b)' = (a' \square b')' = a' \square b' = (a' \square b)' = (a \square b)'$$

Q Which one of the following is NOT a valid identity? (GATE-2019) (2 Marks)

(A) $(x \oplus y) \oplus z = x \oplus (y \oplus z)$

(B) $(x + y) \oplus z = x \oplus (y + z)$

(C) $x \oplus y = x + y$, if $xy = 0$

(D) $x \oplus y = (xy + x'y')'$

Q Let \oplus and \odot denote the Exclusive OR and Exclusive NOR operations, respectively. Which one of the following is NOT CORRECT? (GATE-2018) (2 Marks)

- (A) $(P \oplus Q)' = P \odot Q$ (B) $\bar{P} \oplus Q = P \odot Q$
- (C) $\bar{P} \oplus \bar{Q} = P \oplus Q$ (D) $(P \oplus \bar{P}) \oplus Q = (P \odot \bar{P}) \odot \bar{Q}$

Q Let, $X_1 \oplus X_2 \oplus X_3 \oplus X_4 = 0$ where X_1, X_2, X_3, X_4 are Boolean variables, and \oplus is the XOR operator. Which one of the following must always be TRUE? **(GATE-2016) (2 Marks)**

a) $X_1 X_2 X_3 X_4 = 0$

b) $X_1 X_3 + X_2 = 0$

c) $X'_1 \oplus X'_3 = X'_2 \oplus X'_4$

d) $X_1 + X_2 + X_3 + X_4 = 0$

X_1	X_2	X_3	X_4	$X_1 \oplus X_2 \oplus X_3 \oplus X_4 = 0$	$X_1 X_2 X_3 X_4 = 0$	$X_1 X_3 + X_2 = 0$	$X'_1 \oplus X'_3 = X'_2 \oplus X'_4$	$X_1 + X_2 + X_3 + X_4 = 0$
0	0	0	0	0				
1	0	0	0	1				
2	0	0	1	0				
3	0	0	1	1				
4	0	1	0	0				
5	0	1	0	1				
6	0	1	1	0				
7	0	1	1	1				
8	1	0	0	0				
9	1	0	0	1				
10	1	0	1	0				
11	1	0	1	1				
12	1	1	0	0				
13	1	1	0	1				
14	1	1	1	0				
15	1	1	1	1				

Q Which one of the following expressions does NOT represent exclusive NOR of x and y? **(GATE-2013) (1 Marks)**

(A) $xy + x' y'$

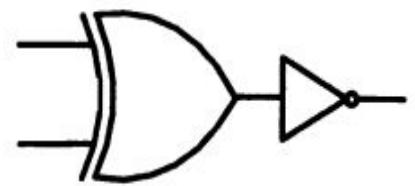
(B) $x \wedge y'$ where \wedge is XOR

(C) $x' \wedge y$ where \wedge is XOR

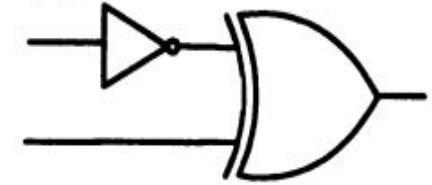
(D) $x' \wedge y'$ where \wedge is XOR

Q Which one of the following circuits is NOT equivalent to a 2-input XNOR (exclusive NOR) gate? **(GATE-2011) (1 Marks)**

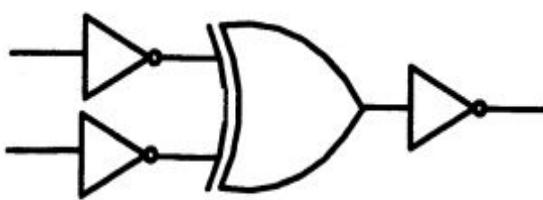
(A)



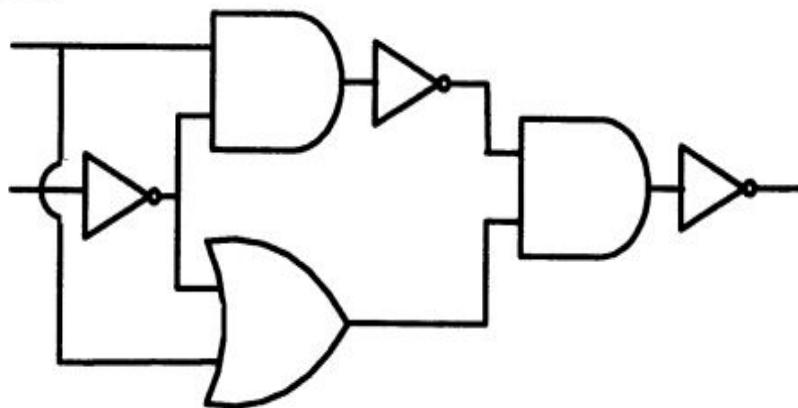
(C)



(B)



(D)



Q The simultaneous equations on the Boolean variables x, y, z and w,

$$x + y + z = 1$$

$$xy = 0$$

$$xz + w = 1$$

$$xy + z'w' = 0$$

have the following solution for x, y, z and w, respectively. **(GATE-2000) (2 Marks)**

- (A) 0 1 0 0
- (B) 1 1 0 1
- (C) 1 0 1 1
- (D) 1 0 0 0

Q. Consider 4-variable functions f_1, f_2, f_3, f_4 expressed in sum-of-minterms form as given below. (Gate 2024 CS) (2 Marks) (MSQ)

$$f_1 = \sum(0,2,3,5,7,8,11,13)$$

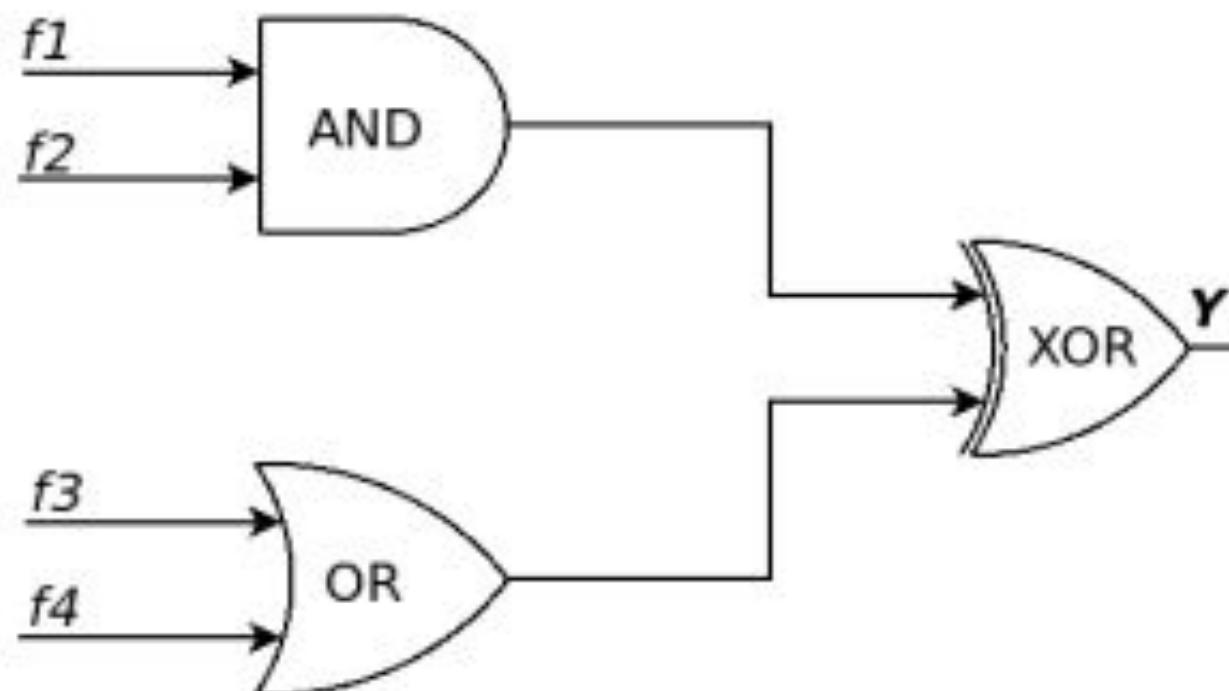
$$f_2 = \sum(1,3,5,7,11,13, 15)$$

$$f_3 = \sum(0,1,4,11)$$

$$f_4 = \sum(0,2,6,13)$$

With respect to the circuit given above, which of the following options is/are CORRECT?

- (a) $Y = \sum(0,1,2,11,13)$
- (b) $Y = \prod(3,4, 5,6,7,8,9,10,12,14,15)$
- (c) $Y = \sum(0,1,2,3,4,5,6,7)$
- (d) $Y = \prod(8,9,10,11,12,13,14,15)$



Boolean Expressions

- Boolean expressions are the method using which we save the information about the Boolean function, that when we get value 1 and when we get value 0 as output.
- So, we convert the truth table of the function into an expression, reading which we can understand where the output is 1 and when it is zero.

Light	Day	Engine	Warning
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



$$\bullet W = a'b'c + ab'c' + abc' + abc$$

- There are two popular approaches for writing these expressions.
 - Sum of Product (SOP) (which remember when we get 1)
 - Product of Sum (POS) (which remember when we get 0)
- Make a note of this as we are studying Boolean function remembering both 0 and 1 is not required, so we can either concentrate on 0 or 1.

- $W(L, D, E) = \sum_m (1, 4, 6, 7)$

- $W(L, D, E) = \prod_M (0, 2, 3, 5)$

Light	Day	Engine	Warning
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

- Power of SOP and POS bother are same, i.e. any Boolean functions can be represented using both SOP and POS.
- Mathematically speaking we should choose (0 or 1), whatever is less in number because then it will be easy to represent.

A	B	$f_1 =$	$f_2 =$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Q The truth table represents the Boolean function **(GATE-2012) (1 Marks)**

- (A) X (B) $X+Y$ (C) $X \oplus Y$ (D) Y

X	Y	$F(X, Y)$
0	0	0
0	1	0
1	0	1
1	1	1

SOP (sum of product)

- A sum of product form expression contains product terms (AND terms) which are sum (OR) together, that's why called sum of product.
- Each product terms (AND terms) consists of one or more literals (variables) appearing either in complements or uncomplemented form. E.g. $a'b + b'c' + abc$
- A product term which contains all the literals (variables) either in complemented or uncomplemented form is called minterm.
- In a n variable function, there will be 2^n minterms.

- The result of a product term must always be 1, If a literal is having value 1 then it is ok, but if not, then we complement those which is 0, it to make it 1.
- There is only 1 input sequence of variables for any minterm on which the output is 1, so it represents information.
- Then the sum of all product term(minterm) to form a function, and functions will have value 1, if at least one of the product term(minterm) is 1.

Binary Representation	Sequence	Minterm	Designation
000	0	$a'b'c'$	m_0
001	1	$a'b'c$	m_1
010	2	$a'bc'$	m_2
011	3	$a'bc$	m_3
100	4	$ab'c'$	m_4
101	5	$ab'c$	m_5
110	6	abc'	m_6
111	7	Abc	m_7

Canonical logic forms

- Either in POS or SOP form it is not essential that all product or sum terms contains all the literals.
- **Canonical SOP form**: - In a sum of product form expression, if each AND term (product term) consists all the literals(variables) appearing either in complements or uncomplemented form. E.g. $a'bc + ab'c' + abc$. Then the form is said to be canonical SOP.

Q The minterm expansion of $f(P, Q, R) = PQ + QR' + PR'$ is **(GATE-2010) (2 Marks)**

- (A) $m_2 + m_4 + m_6 + m_7$
- (B) $m_0 + m_1 + m_3 + m_5$
- (C) $m_0 + m_1 + m_6 + m_7$
- (D) $m_2 + m_3 + m_4 + m_5$

Q. Consider a Boolean expression given by $F(X,Y,Z)=\Sigma(3,5,6,7)$. Which of the following statements is/are CORRECT? (Gate 2024,CS) (2 Marks) (MSQ)

(a) $F(X,Y,Z)=\Pi(0,1,2,4)$

(b) $F(X,Y,Z) = XY + YZ + XZ$

(c) $F(X,Y,Z)$ is independent of input Y

(d) $F(X,Y,Z)$ is independent of input X

	ab	$a'b'$	$a'b$	ab	ab'
c		00	01	11	10
c'	0		0	2	6
c	1		1	3	7

POS (Product of Sum)

- A product of sum (POS) form of expression contains OR (sum) terms which are AND (product) together, that's why called product of sum expression.
- Each OR term (sum term) consists of one or more literals(variables) appearing either in complemented or uncomplemented form. $(a' + b)$. $(b' + c')$. $(a + c)$

- A OR (sum) term which contains all the literals(variables) either in complemented or uncomplemented form is called maxterm. In a n variable function, there will be 2^n maxterms.

Binary Representation	Sequence	Maxterm	Designation
000	0	$a + b + c$	M_0
001	1	$a + b + c'$	M_1
010	2	$a + b' + c$	M_2
011	3	$a + b' + c'$	M_3
100	4	$a' + b + c$	M_4
101	5	$a' + b + c'$	M_5
110	6	$a' + b' + c$	M_6
111	7	$a' + b' + c'$	M_7

- The result of the OR (sum) term must be 0, so If a variable is having value 0 then it is ok, but if not then we complement the variable it to make it 0.
- There is only 1 input sequence for which the output of a Maxterm is 0. Because, it requires all values as zero.
- Then the product of all sum term (maxterm), from a function, and functions will have a value 0, if any of the sum term(maxterm) is 0.

Binary Representation	Sequence	Maxterm	Designation
000	0	$a + b + c$	M_0
001	1	$a + b + c'$	M_1
010	2	$a + b' + c$	M_2
011	3	$a + b' + c'$	M_3
100	4	$a' + b + c$	M_4
101	5	$a' + b + c'$	M_5
110	6	$a' + b' + c$	M_6
111	7	$a' + b' + c'$	M_7

Canonical logic forms

- **Canonical POS form**: - In a product of sum form expression, if each OR term (sum term) consists all the literals(variables) appearing either in complements or uncomplemented form. E.g. $(a' + b + c)$. $(a + b' + c')$. $(a + b + c)$. Then the form is said to be Canonical POS form.

Q Given the function $F = P' + QR$, where F is a function in three Boolean variables P , Q and R and $P' = \neg P$, consider the following statements. **(GATE-2015) (2 Marks)**

S1: $F = \sum (4, 5, 6)$

S2: $F = \sum (0, 1, 2, 3, 7)$

S3: $F = \prod (4, 5, 6)$

S4: $F = \prod (0, 1, 2, 3, 7)$

Which of the following is true?

(A) S1-False, S2-True, S3-True, S4-False

(B) S1-True, S2-False, S3-False, S4-True

(C) S1-False, S2-False, S3-True, S4-True

(D) S1-True, S2-True, S3-False, S4-False

No of functions possible

Q With n-Boolean variables how many different Boolean functions are possible?

No of functions possible

Q With n-Boolean variables how many different Boolean functions are possible?

Solution: with n binary variable we can generate 2^n combinations. And as we know that a Boolean function can also have only 2 possible values either 0 or 1, so total number of different functions possible will be $2^{(2^n)}$.

Similarly, we can generalize this idea as $x^y z$ are the total number of functions possible, x is the nature of the function, y is the nature of the variable and z is the number of variables.

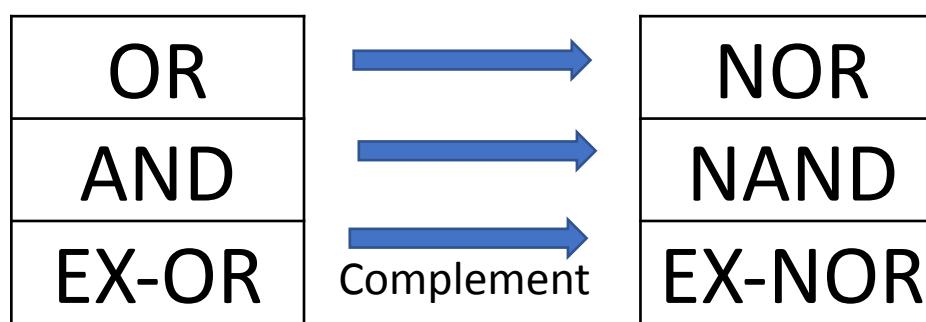
Q What is the maximum number of different Boolean functions involving n Boolean variables? **(GATE-2007) (1 Marks)**

- a) n^2
- b) 2^n
- c) 2^{2^n}
- d) 2^{n^2}

Complementation

- Let us consider a function $f(a, b, c, d, \dots, 0, 1, +, \cdot)$, then the complement of the function is defined as $f'(a', b', c', \dots, z', 1, 0, ., +)$.
- i.e. When all the variables are replaced by their compliments, $0 \square 1, 1 \square 0$, or \square and, and \square or, then we will be called them compliment of a function.

- We can directly put a bar of the entire Boolean expression to find complement of the function.
- We can also directly deal in minterms and maxterm to write complement function.



Q Consider the following Boolean expression $F=(X+Y+Z)(X'+Y)(Y'+Z)$ Which of the following Boolean expressions is/are equivalent to F' (complement of F)? (GATE 2021)(2 MARKS)

(A) $(X'+Y'+Z')(X+Y')(Y+Z')$

(B) $XY'+Z'$

(C) $(X+Z')(Y'+Z')$

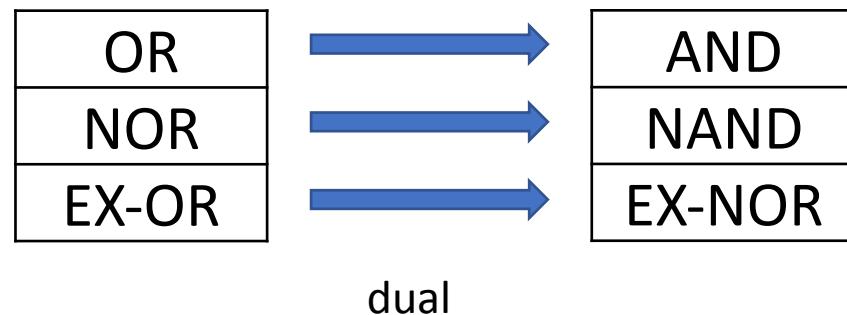
(D) $XY'+YZ'+X'Y'Z'$

Duality

- Let us consider a function $f(a, b, c, d, \dots, z, 0, 1, +, \cdot)$, then the dual of the function is defined as $f^d(a, b, c, \dots, z, 1, 0, \cdot, +)$.
- When the nature of variable remains same but $0 \leftrightarrow 1$, $1 \leftrightarrow 0$, or \wedge and, and \vee or, then they are called dual functions.

a	b		
L 0 H	L 0 H	L 0 H	L 0 H
L 0 H	H 1 L	H 1 L	L 0 H
H 1 L	L 0 H	H 1 L	L 0 H
H 1 L	H 1 L	H 1 L	H 1 L

- When we take dual of a function, then the functionality of a function remains the same but a positive logic system is transformed to negative logic system.
- If a function works correctly in positive logic system, then it must also work correctly in negative logic system as it does not depend on magnitude.



Neutral Function

- A function f is said to be neutral if, it has equal number of minterms and maxterms.

Q. Let X be a 3-variable Boolean function that produces output as '1' when at least two of the input variables are '1'. Which of the following statement(s) is/are CORRECT, where a, b, c, d, e are Boolean variables? **(GATE-2025)**

- A)** $X(a, b, X(c, d, e)) = X(X(a, b, c), d, e)$
- B)** $X(a, b, X(a, b, c)) = X(a, b, c)$
- C)** $X(a, b, X(a, c, d)) = (X(a, b, a) \text{ AND } X(c, d, c))$
- D)** $X(a, b, c) = X(a, X(a, b, c), X(a, c, c))$

Self-Dual

- A function f is said to be self-dual if both the functions and its dual are same.
- $f = f^D$
- $F(a, b, c) = ab + bc + ca$

Q which of the following functions are self-dual?

$$f(a, b, c) = \sum_m (0, 3)$$

$$F(a, b, c) = \sum_m (0, 1, 6, 7)$$

$$F(a, b, c) = \sum_m (0, 1, 2, 4)$$

$$F(a, b, c) = \sum_m (3, 5, 6, 7)$$

Q how to check weather a function is self-dual or not?

- 1) check whether function is neutral or not i.e. (minterm = maxterm)
- 2) A self-dual function does not have any mutually exclusive terms. So in every pair of mutual exclusion term, we can pick only 1 minterm.
- 3) for n variable functions total 2^n minterms are possible, so we will have 2^{n-1} pair of mutually exclusive minterms, in every pair of mutually exclusive minterms we have two choice so, For n variable function total number of $2^{(2^{n-1})}$ self-dual functions are possible.

0□□7

1□□6

2□□5

3□□4

e.g. of Mutual exclusive min terms

Q The dual of a Boolean function $F(X_1, X_2, \dots, X_n, +, *, ')$, written as F^D , is the same expression as that of F with $+$ and $*$ swapped. F is said to be self-dual if $F = F^D$. The number of self-dual functions with n Boolean variables is. **(GATE-2014) (1 Marks)**

- a) 2^n
- b) $2^{(n-1)}$
- c) $2^{(2014^n)}$
- d) $2^{(2^{(n-1)})}$

Q. Consider the following four-variable Boolean function in sum-of-product form:

$$F(b_3, b_2, b_1, b_0) = \Sigma(0, 2, 4, 8, 10, 11, 12)$$

Where the value of the function is computed by considering $b_3b_2b_1b_0$ as a 4-bit binary number, where b_3 is the MSB and b_0 is the LSB.

Which ONE of the following options is the CORRECT minimized Boolean expression for F?
(GATE-2025)

- A) $\neg b_1 \neg b_0 + \neg b_2 \neg b_0 + b_1 \neg b_2 b_3$
- B) $\neg b_1 \neg b_0 + \neg b_2 \neg b_0$
- C) $\neg b_2 \neg b_0 + b_1 b_2 b_3$
- D) $\neg b_0 \neg b_2 + \neg b_3$

Orthogonal

- A function f is said to be orthogonal if the compliment and dual of the function are same. A function can never be both orthogonal and self-dual because it will imply the function is same to its compliment, which is never possible.
- $f' = f^d$
- $f(a, b, c) = a'b'c' + abc + a'bc' + ab'c$

Q how to check weather a function is orthogonal or not?

1) check whether function is neutral or not i.e. (minterm = maxterm)

2) we can choose any pair of mutual exclusive terms, but the minterms and its complement, both must be present.

3) If there is a function f of n-variables, so we will have 2^{n-1} pair of mutually exclusive minterms, out of which we have to select exactly half of the pairs (2^{n-2}), then $(2^{n-1})C(2^{n-2})$ different orthogonal functions are possible.

0□□7

1□□6

2□□5

3□□4

e.g. of Mutual exclusive min terms

Q which of the following functions are self-dual?

$$f(a, b, c) = \sum_m (0, 3)$$

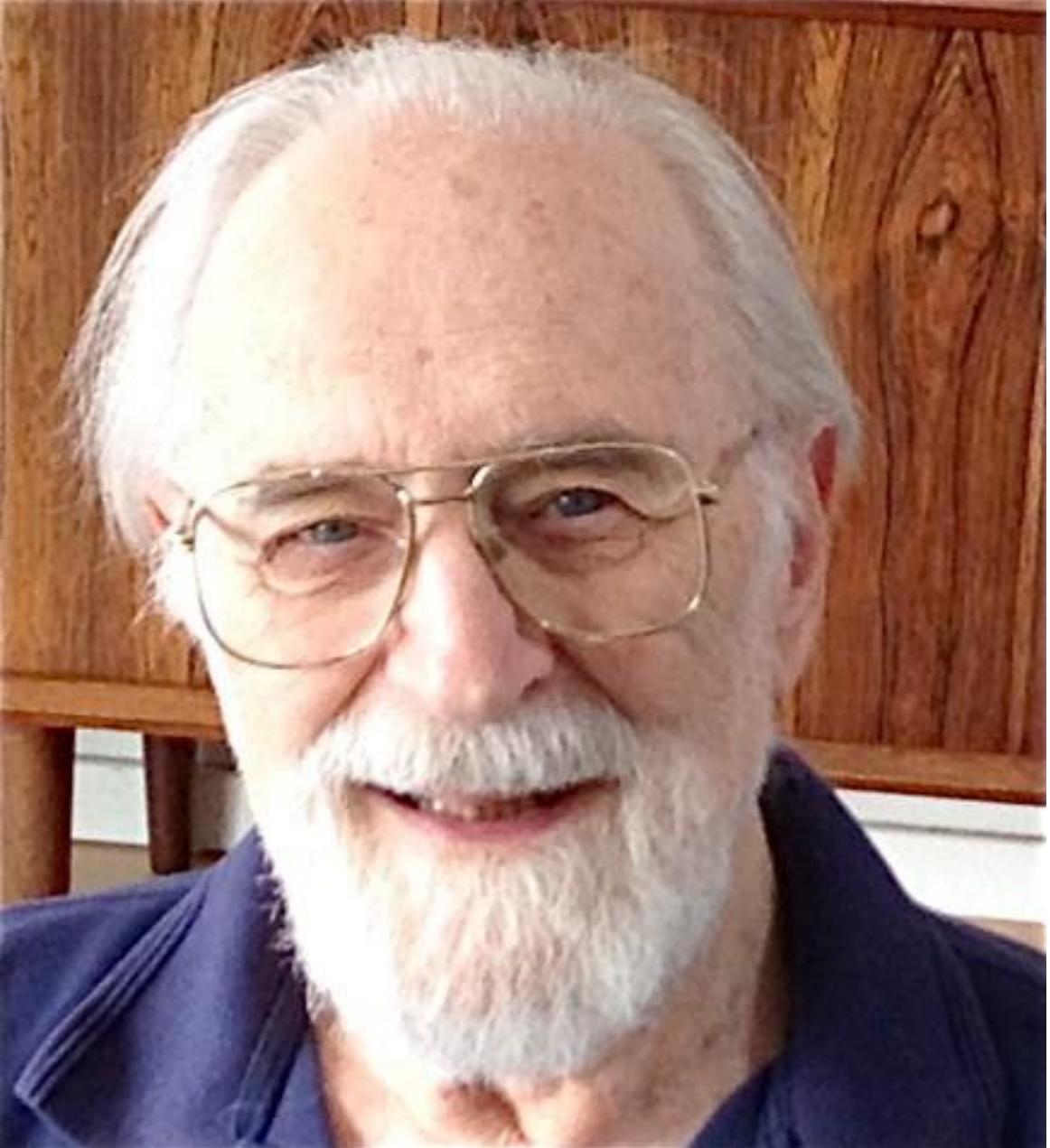
$$F(a, b, c) = \sum_m (0, 1, 6, 7)$$

$$F(a, b, c) = \sum_m (0, 1, 2, 4)$$

$$F(a, b, c) = \sum_m (3, 5, 6, 7)$$

Simplification of Boolean expression

- After deriving a Boolean expression from the truth table next important step is to minimise it, so that the cost of implementation into hardware and delay because of additional gates can be reduced. There are following methods for simplifying a Boolean expression
 - 1) Using Karnaugh-map
 - 2) Using Algebraic method (Boolean Laws)



- Maurice Karnaugh (born 4 October 1924) is an American physicist, mathematician and inventor known for the Karnaugh map used in Boolean algebra.
- Age: 96

Karnaugh Map

- **Problem with other methods:** - The algebraic procedure using Boolean laws and rules for minimising Boolean expression becomes difficult when a function becomes complex, because we have to identify where is the scope of minimization based on some Boolean laws and It is difficult to understand where we reach saturation point or not.

$$a + a'b + a'b'c + a'b'c'd'$$

- Karnaugh map is one of the most extensively used tool, it is a graphical representation, represents truth table by pictorial form, provides a systematic method for simplifying or minimizing a Boolean expression. For a n-variable k-map, there will be 2^n cells addressed by a gray code. Each cell corresponds to one minterm or maxterm.

	a	b	c	d	F
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	

	ab	$a'b'$	$a'b$	ab	ab'	
cd		00	01	11	10	
$c'd'$	00		0	4	12	8
$c'd$	01		1	5	13	9
cd	11		3	7	15	11
cd'	10		2	6	14	10

	cd				
ab					

	ab	$a'b'$	$a'b$	ab	ab'	
cd		00	01	11	10	
$c'd'$	00		0	4	12	8
$c'd$	01		1	5	13	9
cd	11		3	7	15	11
cd'	10		2	6	14	10

	cd	$c'd'$	$c'd$	cd	cd'	
ab		00	01	11	10	
$a'b'$	00		0	1	3	2
$a'b$	01		4	5	7	6
ab	11		12	13	15	14
ab'	10		8	9	11	10

	ad	$a'd'$	$a'd$	ad	ad'
bc		00	01	11	10
$b'c'$	00		0	1	9
$b'c$	01		2	3	11
bc	11		6	7	15
bc'	10		4	5	13

	db	$d'b'$	$d'b$	db	db'
ca		00	01	11	10
$c'a'$	00		0	4	5
$c'a$	01		8	12	13
ca	11		10	14	15
ca'	10		2	6	7

	ab	$a'b'$	$a'b$	ab	ab'
c		00	01	11	10
c'	0	0	2	6	4
c	1	1	3	7	5

	bc	$b'c'$	$b'c$	bc	bc'
a		00	01	11	10
a'	0	0	1	3	2
a	1	4	5	7	6

	a	a'	a
bc		0	1
$b'c'$	00	0	4
$b'c$	01	1	5
bc	11	3	7
bc'	10	2	6

	a	a'	a
b		0	1
b'	0	0	2
b	1	1	3

	b	b'	b
a		0	1
a'	0	0	1
a	1	2	3

5-Variable K-Map

a'

	bc	$b'c'$	$b'c$	bc	bc'
de	00	01	11	10	
$d'e'$	00	0	4	12	8
$d'e$	01	1	5	13	9
de	11	3	7	15	11
de'	10	2	6	14	10

a

	bc	$b'c'$	$b'c$	bc	bc'
de	00	01	11	10	
$d'e'$	00	16	20	28	24
$d'e$	01	17	21	29	25
de	11	19	23	31	27
de'	10	18	22	30	26

6-Variable K-Map

$a'b'$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00	0	4	12	8
e'f	01	1	5	13	9
ef	11	3	7	15	11
ef'	10	2	6	14	10

ab'

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00	32	36	44	40
e'f	01	33	37	45	41
ef	11	35	39	47	43
ef'	10	34	38	46	42

$a'b$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00	16	20	28	24
e'f	01	17	21	29	25
ef	11	19	23	31	27
ef'	10	18	22	30	26

ab

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00	48	52	60	56
e'f	01	49	53	61	57
ef	11	51	55	63	59
ef'	10	50	54	62	58

7-Variable K-Map

$a'b'c'$

	cd	$c'd'$	$c'd$	cd	cd'
ef	00	01	11	10	
$e'f'$	00				
$e'f$	01				
ef	11				
ef'	10				

0-15

$a'b'c$

	cd	$c'd'$	$c'd$	cd	cd'
ef	00	01	11	10	
$e'f'$	00				
$e'f$	01				
ef	11				
ef'	10				

16-31

$a'bc'$

	cd	$c'd'$	$c'd$	cd	cd'
ef	00	01	11	10	
$e'f'$	00				
$e'f$	01				
ef	11				
ef'	10				

32-47

$a'bc$

	cd	$c'd'$	$c'd$	cd	cd'
ef	00	01	11	10	
$e'f'$	00				
$e'f$	01				
ef	11				
ef'	10				

48-63

$ab'c'$

	cd	$c'd'$	$c'd$	cd	cd'
ef	00	01	11	10	
$e'f'$	00				
$e'f$	01				
ef	11				
ef'	10				

64-79

$ab'c$

	cd	$c'd'$	$c'd$	cd	cd'
ef	00	01	11	10	
$e'f'$	00				
$e'f$	01				
ef	11				
ef'	10				

80-95

abc'

	cd	$c'd'$	$c'd$	cd	cd'
ef	00	01	11	10	
$e'f'$	00				
$e'f$	01				
ef	11				
ef'	10				

96-111

abc

	cd	$c'd'$	$c'd$	cd	cd'
ef	00	01	11	10	
$e'f'$	00				
$e'f$	01				
ef	11				
ef'	10				

112-127

$a'b'c'd'$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$a'b'c'd$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$a'bc'd'$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$a'bc'd$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$ab'c'd'$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$ab'c'd$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$abc'd'$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$abc'd$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$a'b'cd'$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$32-47$

$a'b'cd$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$48-63$

$a'bc'd'$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$a'bc'd$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$80-95$

$a'bcd'$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$96-111$

$a'bcd$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$112-127$

$ab'c'd'$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$ab'c'd$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$144-159$

$ab'cd'$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$160-175$

$ab'cd$

	cd	c'd'	c'd	cd	cd'
ef	00	01	11	10	
e'f'	00				
e'f	01				
ef	11				
ef'	10				

$176-191$

$abc'd'$

K-Map for POS

	a+b				
c+d					

K-Map for POS

	$a+b$	$a+b$	$a+b'$	$a'+b'$	$a'+b$
$c+d$		0+0	0+1	1+1	1+0
$c+d$	0+0	0	4	12	8
$c+d'$	0+1	1	5	13	9
$c'+d'$	1+1	3	7	15	11
$c'+d$	1+0	2	6	14	10

- **Minimal function:** - A function is said to be minimal if it is representing the function expression, if it is using minimal no of literals to represent the function.

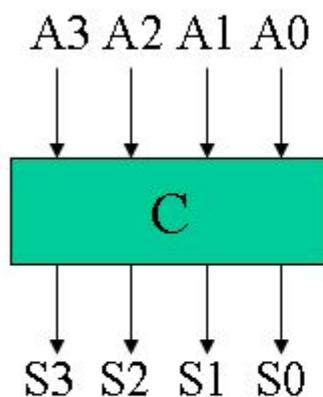
- Rules of grouping: -

1. Every minterm must be covered
2. Group must have contiguous Cells(circular)
3. Group must be in horizontal or vertical fashion(circular)
4. Number of cells in a group must be in power of 2
5. Will Try to make largest group possible, so that number of literals in the expression can be reduced
6. Can also take don't care if it helps in creating the larger groups, other wise don't care.
7. Will consider new implicant if it is covering some new minterm.

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00	0	4	12	8
$c'd$	01	1	5	13	9
cd	11	3	7	15	11
cd'	10	2	6	14	10

Don't care condition

- Don't care cases are those cases which can never occur logically in that function.
- It is not essential to cover don't care conditions, but if don't care are helping to generate bigger prime implicants, then we can use them.



A3	A2	A1	A0	S3	S2	S1	S0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00			1	
$c'd$	01	D	D	1	D
cd	11		1	1	
cd'	10		1	1	

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00		1		
$c'd$	01		1	1	1
cd	11	1	1	1	
cd'	10			1	

First try to cover those minterms which do not have an option.

For some functions more than one minimal Boolean expression are possible

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00		D		D
$c'd$	01	1	1	D	
cd	11		D	1	1
cd'	10		1	D	

- **Implicants**: - Collection of adjacent minterms are called implicants.
- **Prime Implicant (PI)**: - Implicant is called Prime implicant (PI) if it is not subset of any other implicant(group). (overlapping is allowed).
 - We will always try to find Prime Implicant

$$f(a, b, c, d) = \sum_m \{4, 5, 6, 7, 8, 9, 10, 11, 13, 14\}$$

	ab	a'b'	a'b	ab	ab'
cd		00	01	11	10
c'd'	00		0	4	12
c'd	01		1	5	13
cd	11		3	7	15
cd'	10		2	6	14

Essential Prime Implicant (EPI)

- If a prime implicant (PI) have some unique minterm which no other prime implicant covers then, it is called essential prime implicant (EPI).
- Essential prime implicant (EPI) must always be present in the minimal Boolean expression.

$Q f(a, b, c) = \sum_m \{1, 2, 3, 4, 5\}$, find Number of PI, Number of EPI, Number of literals are there in minimal expression, Number of different minimal expression possible?

	ab	$a'b'$	$a'b$	ab	ab'
c	00	01	11	10	
c'	0	0	2	6	4
c	1	1	3	7	5

1. Number of PI
2. Number of EPI
3. Number of different minimal expression possible
4. Number of literals are there in minimal expression

Q Consider the minterm list form of a Boolean function F given below.

$$F(P, Q, R, S) = \sum m(0, 2, 5, 7, 9, 11) + d(3, 8, 10, 12, 14)$$

Here, m denotes a minterm and d denotes a don't care term. The number of essential prime implicants of the function F is _____. **(GATE-2018) (2 Marks)**

	pq	p'q'	p'q	pq	pq'
rs	00	01	11	10	
r's'	00	0	4	12	8
r's	01	1	5	13	9
rs	11	3	7	15	11
rs'	10	2	6	14	10

Q The total number of prime implicants of the function $f(w, x, y, z) = \Sigma(0, 2, 4, 5, 6, 10)$ is _____. (GATE-2015) (1 Marks)

- (A) 2 (B) 3 (C) 4 (D) 5

	wx	$w'x'$	$w'x$	wx	wx'
yz	00	01	11	10	
$y'z'$	00	0	4	12	8
$y'z$	01	1	5	13	9
yz	11	3	7	15	11
yz'	10	2	6	14	10

Q Which are the essential prime implicants of the following Boolean function?

(GATE-2004) (1 Marks)

$$f(a, b, c) = a'c + ac' + b'c$$

(A) $a'c$ and ac'

(B) $a'c$ and $b'c$

(C) $a'c$ only

(D) ac' and bc'

	ab	$a'b'$	$a'b$	ab	ab'
c		00	01	11	10
c'	0	0	2	6	4
c	1	1	3	7	5

Q Consider the functions given below ?

1. Number of PI
2. Number of EPI
3. Number of different minimal expression possible
4. Number of literals are there in minimal expression

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00	1			1
$c'd$	01	1	1		
cd	11		1	1	
cd'	10			1	1

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00	1	D	D	1
$c'd$	01	D			
cd	11				
cd'	10	1			D

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00		1	1	
$c'd$	01	D			1
cd	11	D			1
cd'	10		1	1	D

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00	D	1		1
$c'd$	01		1	D	
cd	11	1	D	D	
cd'	10	D			D

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00			D	1
$c'd$	01		1	D	1
cd	11		1	D	D
cd'	10		1	D	D

Q Consider the functions given below ?

	ab	$a'b'$	$a'b$	ab	ab'
c		00	01	11	10
c'	0	1	1		1
c	1		1	1	1

1. Number of PI
2. Number of EPI
3. Number of different minimal expression possible
4. Number of literals are there in minimal expression

Q Given $f(w, x, y, z) = \sum_m (0, 1, 2, 3, 7, 8, 10) + \sum_d (5, 6, 11, 15)$, where d represents the don't-care condition in Karnaugh maps. Which of the following is a minimum product-of-sums (POS) form of $f(w, x, y, z)$? (GATE-2017) (2 Marks)

- a) $f = (w' + z') (x' + z)$
- b) $f = (w' + z) (x + z)$
- c) $f = (w + z) (x' + z)$
- d) $f = (w + z') (x' + z)$

	wx	$w'x'$	$w'x$	wx	wx'
yz	00	01	11	10	
$y'z'$	00	0	4	12	8
$y'z$	01	1	5	13	9
yz	11	3	7	15	11
yz'	10	2	6	14	10

	$w+x$	$w+x$	$w+x'$	$w'+x'$	$w'+x$
$y+z$		0+0	0+1	1+1	1+0
$y+z$	0+0		0	4	12
$y+z'$	0+1		1	5	13
$y'+z'$	1+1		3	7	15
$y'+z$	1+0		2	6	14

Q Consider the following minterm expression for F: (GATE-2014) (2 Marks)

$$F(P, Q, R, S) = \sum (0, 2, 5, 7, 8, 10, 13, 15)$$

The minterm 2, 7, 8, 13 are 'do not care' term.

The minimum sum-of-products from for F is

a) $QS' + Q'S$

b) $Q'S' + QS$

c) $Q'R'S' + Q'RS' + QR'S + QRS$

d) $P'Q'S' + P'QS + PQS + PQ'S'$

	pq	$p'q'$	$p'q$	pq	pq'	
rs		00	01	11	10	
$r's'$	00		0	4	12	8
$r's$	01		1	5	13	9
rs	11		3	7	15	11
rs'	10		2	6	14	10

Q What is the minimal form of the Karnaugh map shown below? Assume that X denotes a don't care term. **(GATE-2012) (2 Marks)**

- (A)** $b'd'$ **(B)** $b'd' + b'c'$ **(C)** $b'd' + a'b'c'd'$ **(D)** $b'd' + b'c' + c'd'$

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00	1	D	D	1
$c'd$	01	D			1
cd	11				
cd'	10	1			D

Q In the Karnaugh map shown below, X denotes a don't care term. What is the minimal form of the function represented by the Karnaugh map? (GATE-2008) (2 Marks)

- A) $b'd' + a'd'$ B) $a'b' + b'd' + a'bd'$ C) $b'd' + a'bd'$ D) $a'b' + b'd' + a'd'$

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00	1	1		1
$c'd$	01	D			
cd	11	D			
cd'	10	1	1		D

Q Consider the following Boolean function of four variables: (GATE-2007) (2 Marks)

$$f(w, x, y, z) = \sum (1, 3, 4, 6, 9, 11, 12, 14)$$

The function is:

(A) independent of one variable.

(B) independent of two variables.

(C) independent of three variables.

(D) dependent on all the variables.

	WX	W'X'	W'X	WX	WX'
yz	00	01	11	10	
y'z'	00	0	4	12	8
y'z	01	1	5	13	9
yz	11	3	7	15	11
yz'	10	2	6	14	10

Q The switching expression corresponding to $f(A, B, C, D) = \sum (1, 4, 5, 9, 11, 12)$ is
(GATE-2005) (2 Marks)

(A) $BC'D' + A'C'D + AB'D$

(B) $ABC' + ACD + B'C'D$

(C) $ACD' + A'BC' + AC'D'$

(D) $A'BD + ACD' + BCD'$

	ab	$a'b'$	$a'b$	ab	ab'
cd		00	01	11	10
$c'd'$	00	0	4	12	8
$c'd$	01	1	5	13	9
cd	11	3	7	15	11
cd'	10	2	6	14	10

Q The literal count of a Boolean expression is the sum of the number of times each literal appears in the expression. For example, the literal count of $(xy + xz')$ is 4. What are the minimum possible literal counts of the product-of-sum and sum-of-product representations respectively of the function given by the following Karnaugh map? Here, X denotes “don’t care” (GATE-2003) (2 Marks)

(A) (11, 9)

(B) (9, 13)

(C) (9, 10)

(D) (11, 11)

	zw	$z'w'$	$z'w$	zw	zw'
xy	00	01	11	10	
$x'y'$	00	X	1		1
$x'y$	01		1	X	
xy	11	1	X	X	
xy'	10	X			X

	$z+w$	$z+w'$	$z+w'$	$z'+w'$	$z'+w$
$x+y$		0+0	0+1	1+1	1+0
$x+y$	0+0	X		0	
$x+y'$	0+1	0		X	0
$x'+y'$	1+1		X	X	0
$x'+y$	1+0	X	0	0	X

Q Minimum sum of product expression for $f(w, x, y, z)$ shown in Karnaugh-map below is (GATE-2002) (2 Marks)

- (A) $xz + y'z$ (B) $xz' + zx'$ (C) $x'y + zx'$ (D) None of these

	wx	$w'x'$	$w'x$	wx	wx'
yz	00	01	11	10	
$y'z'$	00	D	1	0	1
$y'z$	01	0	1	D	0
yz	11	1	D	D	0
yz'	10	D	0	0	D

	wx	$w'x'$	$w'x$	wx	wx'
yz	00	01	11	10	
$y'z'$	00	D	1		1
$y'z$	01		1	D	
yz	11	1	D	D	
yz'	10	D			D

Q Given the following Karnaugh map, which one of the following represents the minimal Sum-Of-Products of the map? (GATE-2001) (2 Marks)

- (A) $xy + y'z$ (B) $wx'y' + xy + xz$ (C) $w'x + y'z + xy$ (D) $xz + y$

	wx	$w'x'$	$w'x$	wx	wx'
yz		00	01	11	10
$y'z'$	00	0	D	0	D
$y'z$	01	D	1	D	1
yz	11	0	D	1	0
yz'	10	0	1	D	0

	wx	$w'x'$	$w'x$	wx	wx'
yz		00	01	11	10
$y'z'$	00		D		D
$y'z$	01	D	1	D	1
yz	11		D	1	
yz'	10		1	D	

Q Let $f(w, x, y, z) = \sum (0, 4, 5, 7, 8, 9, 13, 15)$. Which of the following expressions are NOT equivalent to f ? (GATE-2007) (2 Marks)

- (A) $x'y'z' + w'xy' + wy'z + xz$
- (B) $w'y'z' + wx'y' + xz$
- (C) $w'y'z' + wx'y' + xyz + xy'z$
- (D) $x'y'z' + wx'y' + w'y$

	wx	$w'x'$	$w'x$	wx	wx'
yz	00	01	11	10	
$y'z'$	00				8
$y'z$	01		1	5	13
yz	11		3	7	15
yz'	10		2	6	14
					10

Q Which function does NOT implement the Karnaugh map given below? (GATE - 2000) (2 Marks)

A) $(W + X) Y$

B) $XY + YW$

C) $(W + X)(W' + Y)(X' + Y)$

D) none of the above

	WZ	W'z'	W'z	Wz	wz'
xy		00	01	11	10
x'y'	00	0	X	0	0
x'y	01	0	X	1	1
xy	11	1	1	1	1
xy'	10	0	X	0	0

Q Consider the following Boolean expression for F: (GATE-2014)(2 Marks)

$$F(P, Q, R, S) = PQ + P'QR + P'QR'S$$

the minimal sum-of-products form of F is

a) $PQ + QR + QS$

b) $P + Q + R + S$

c) $P' + Q' + R' + S'$

d) $P'R + P'R'S + P$

	pq	p'q'	p'q	pq	pq'
rs	00	01	11	10	
r's'	00				
r's	01				
rs	11				
rs'	10				

Q The simplified SOP (Sum of Product) form of the Boolean expression $(P + Q' + R'). (P + Q' + R). (P + Q + R')$ is **(GATE-2011)(1 Marks)**

(A) $(P'.Q + R')$

(B) $(P + Q'.R')$

(C) $(P'.Q + R)$

(D) $(P.Q + R)$

	pq	$p'q'$	$p'q$	pq	pq'
rs	00	01	11	10	
$r's'$	00	0	2	6	4
$r's$	01	1	3	7	5

	$p+q$	$p+q$	$p+q'$	$p'+q'$	$p'+q$
r	0+0	0+1	1+1	1+0	
r	0	0	2	6	4
r'	1	1	3	7	5

Q If P, Q, R are Boolean variables, then $(P + Q')(PQ' + PR)(P'R' + Q')$ simplifies

(GATE-2008) (1 Marks)

(A) PQ'

(B) PR'

(C) PQ' + R

(D) PR' + Q

	pq	p'q'	p'q	pq	pq'
rs	00	01	11	10	
r's'	00	0	2	6	4
r's	01	1	3	7	5

	p+q	p+q	p+q'	p'+q'	p'+q
r	0+0	0+1	1+1	1+0	
r	0	0	2	6	4
r'	1	1	3	7	5

Q The function $AB'C + A'BC + ABC' + A'B'C + AB'C'$ is equivalent to **(GATE-2004) (1 Marks)**

- (A)** $AC' + AB + A'C$ **(B)** $AB' + AC' + A'C$ **(C)** $A'B + AC' + AB'$ **(D)** $A'B + AC + AB'$

	ab	$a'b'$	$a'b$	ab	ab'
c		00	01	11	10
c'	00	0	2	6	4
c	01	1	3	7	5

Q Which of the following expressions is equivalent to $(A \oplus B) \oplus C$ (GATE-2004) (2 Marks)

(A) $(A+B+C)(A'+B'+C')$

(B) $(A+B+C)(A'+B'+C)$

(C) $ABC + A'(B \oplus C) + B'(A \oplus C)$

(D) None

	ab	$a'b'$	$a'b$	ab	ab'
c		00	01	11	10
c'	00		0	2	6
c	01		1	3	7
					5

Q The Boolean function $x'y' + xy + x'y$ is equivalent to **(GATE-2004) (1 Marks)**

- (A)** $x' + y'$ **(B)** $x + y$ **(C)** $x + y'$ **(D)** $x' + y$

	x	x'	x
y		0	1
y'	0	0	2
y	1	1	3

Q If w, x, y, z are Boolean variables, then which one of the following is INCORRECT? (GATE-2017) (2 Marks)

a) $wx + w(x + y) + x(x + y) = x + wy$

b) $(wx'(y + z'))' + w'x = w' + x + y'z$

c) $(wx'(y + xz') + w'x')y = xy'$

d) $(w + y)(wxy + wyz) = wxy + wyz$

	wx	$w'x'$	$w'x$	wx	wx'
yz	00	01	11	10	
$y'z'$	00				
$y'z$	01				
yz	11				
yz'	10				
	0	4	12	8	
	1	5	13	9	
	3	7	15	11	
	2	6	14	10	

Q Let $f(A, B) = A' + B$. Simplified expression for function $f(f(x + y, y), z)$ is :

(GATE-2002) (2 Marks)

- (A)** $x' + z$ **(B)** xyz **(C)** $xy' + z$ **(D)** None of these

Q. Given the following Karnaugh Map for a Boolean function $F(w, x, y, z)$:

$wx \backslash yz$	00	01	10	11
00	1	0	0	1
01	0	1	1	0
10	0	1	1	0
11	1	0	0	1

Which one or more of the following Boolean expression(s) represent(s) F ? **(GATE 2025)**

- A) $\bar{w}\bar{x}\bar{y}z + w\bar{x}\bar{y}z + \bar{w}\bar{x}yz + w\bar{x}\bar{y}z + xz$
- B) $\bar{w}\bar{x}y z + \bar{w}\bar{x}yz + \bar{w}\bar{x}\bar{y}z + xz$
- C) $\bar{w}\bar{x}\bar{y}z + w\bar{x}\bar{y}z + \bar{w}\bar{x}\bar{y}z + xz$
- D) $\bar{x}z + xz$

Q. Which of the following Boolean algebraic equation(s) is/are CORRECT?(GATE 2025)

- A) $\bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}C + ABC = BC + \bar{B}\bar{C} + \bar{A}\bar{B}$
- B) $AB + \bar{A}C + BC = AB + \bar{A}C$
- C) $(A + C)(\bar{A} + B) = AB + \bar{A}C$
- D) $(A + \bar{B} + \bar{D})(C + D)(\bar{A} + C + D)(A + B + \bar{D}) = \bar{A}D + \bar{C}\bar{D}$

Q Consider two functions f_1 and f_2 ?

$$F_1 = \sum_m (1, 2, 5, 6)$$

$$F_2 = \sum_m (2, 3, 4, 5)$$

Find $f_1 \cdot f_2$ and $f_1 + f_2$

Q Consider two functions f_1 and f_2 ?

$$F_1 = \sum_m (1, 2, 5, 6)$$

$$F_2 = \sum_m (2, 3, 4, 5)$$

Find $f_1 \cdot f_2$ and $f_1 + f_2$

	F_1	F_2	$f_1 + f_2$	$f_1 \cdot f_2$
0	0	0		
1	1	0		
2	1	1		
3	0	1		
4	0	1		
5	1	1		
6	1	0		
7	0	0		

Q Consider f_1 & f_2 -

$$F_1(a, b, c) = \sum_m (0, 2, 4) + d (3, 5, 7)$$

$$F_2(a, b, c) = \sum_m (2, 3) + d (1, 6, 7)$$

Find $f_1 \cdot f_2$ and $f_1 + f_2$

	F_1	F_2	$f_1 + f_2$	$f_1 \cdot f_2$
0	1	0		
1	0	D		
2	1	1		
3	D	1		
4	1	0		
5	D	0		
6	0	D		
7	d	d		

Q Consider f_1 & f_2 –

$$F_1(a, b, c, d) = \sum_m (1, 3, 4, 5, 9, 10, 11) + d (6, 8)$$

$$F_2(a, b, c, d) = \sum_m (0, 2, 4, 7, 8, 15) + d (9, 12)$$

Find $f_1 \cdot f_2$ and $f_1 + f_2$

	F_1	F_2	$f_1 + f_2$	$f_1 \cdot f_2$
0	0	1		
1	1	0		
2	0	1		
3	1	0		
4	1	1		
5	1	0		
6	D	0		
7	0	1		
8	D	1		
9	1	D		
10	1	0		
11	1	0		
12	0	D		
13	0	0		
14	0	0		
15	0	1		

Q Consider a function $f(a, b, c) = \sum_m (3, 5, 6)$ is being minimized to $A + BC$. Predict what are the don't care conditions?

- a) d(2,4) b) d(2,7) c) d(4,7) d) d (2,4,7)

	ab	a'b'	a'b	ab	ab'
c	00	01	11	10	
c'	00	0	2	6	4
c	01	1	3	7	5

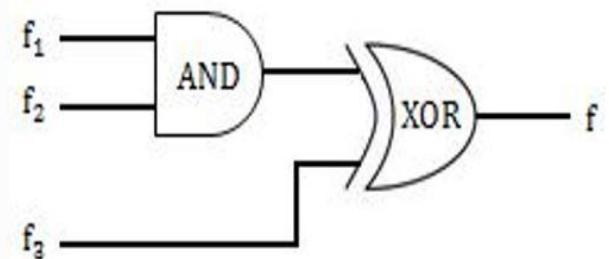
Q Consider three 4-variable functions f_1 , f_2 , and f_3 , which are expressed in sum-of-minterms as

$$f_1 = (0, 2, 5, 8, 14)$$

$$f_2 = (2, 3, 6, 8, 14, 15)$$

$$f_3 = (2, 7, 11, 14)$$

For the following circuit with one AND gate and one XOR gate, the output function f can be expressed as:



For the following circuit with one AND gate and one XOR gate, the output function f can be

expressed as: **(GATE-2019) (2 Marks)**

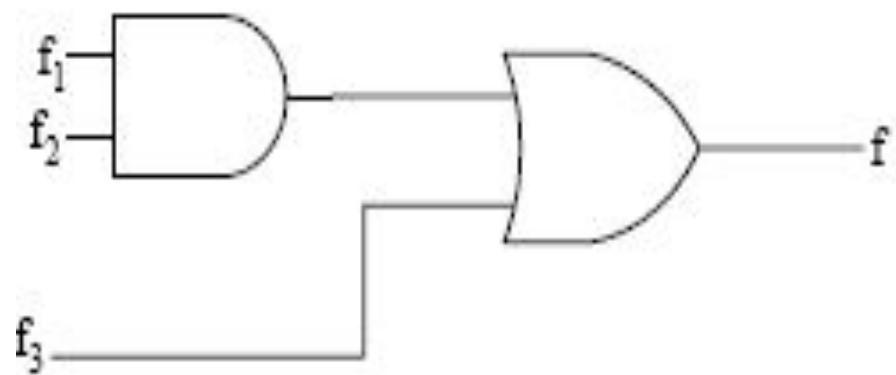
a) $\Sigma(7, 8, 11)$

b) $\Sigma(2, 7, 8, 11, 14)$

c) $\Sigma(2, 14)$

d) $\Sigma(0, 2, 3, 5, 6, 7, 8, 11, 14, 15)$

Q Given f_1 , f_3 and f in canonical sum of products form (in decimal) for the circuit (GATE-2008) (1 Marks)



$$f_1 = \sum m(4, 5, 6, 7, 8)$$

$$f_3 = \sum m(1, 6, 15)$$

$$f = \sum m(1, 6, 8, 15)$$

then f_2 is

A) $\sum_m(4, 6)$

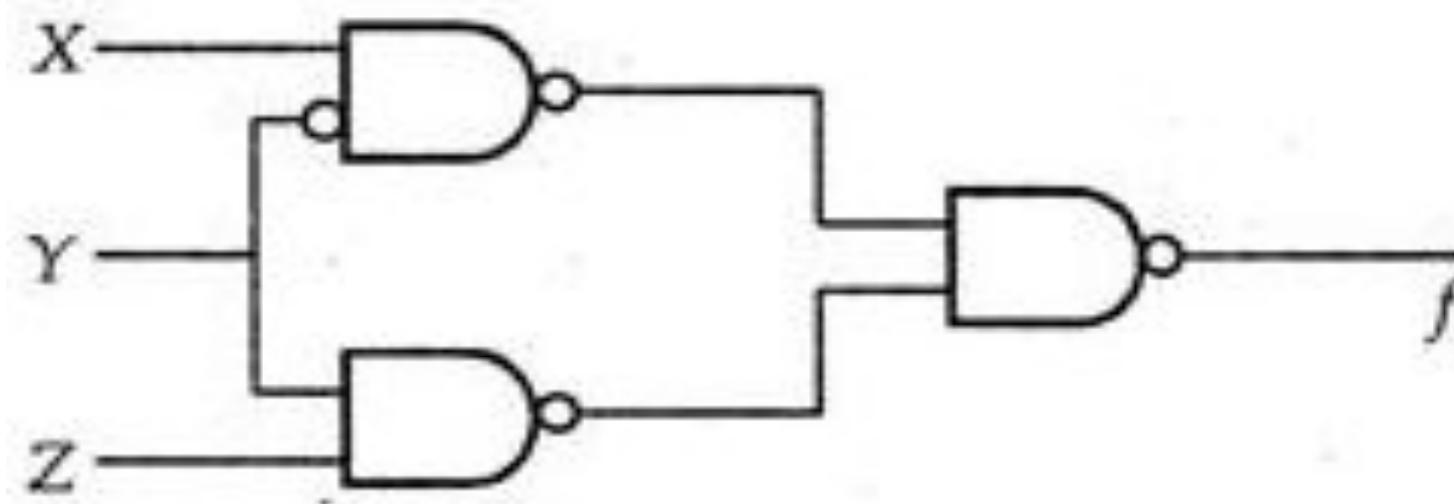
B) $\sum_m(4, 8)$

C) $\sum_m(6, 8)$

D) $\sum_m(4, 6, 8)$

Q Consider the following circuit. (GATE-2005) (1 Marks)

Which one of the following is TRUE?



- (A) f is independent of X
- (B) f is independent of Y
- (C) f is independent of Z
- (D) None of X, Y, Z is redundant

- **Irredundant function**: - A function f is said to be irredundant if no product term can be removed from the function without changing the functional capability of the function.
- **Minimal function**: - A function is said to be minimal if it is representing the function expression, if it is using minimal no of literals to represent the function.
- If a function is irredundant then, it may or may not be minimal, but if a function is minimal, then it is irredundant.
- $F(x, y, z) = xz' + y'z' + yz + xz$, this function is irredundant but not minimal
- $F(x, y, z) = X + yz + y'z'$
- Minimal \subseteq irredundant

	xy	$x'y'$	$x'y$	xy	xy'
z		00	01	11	10
z'	00				
z	01				

0 2 6 4

1 3 7 5

- **Absorption law**

- $a + ab = a$
- $a.(a+b) = a$

- **Compensation theorem**

- $ab + a'c + bc = ab + a'c$
- $(a+b)(a' + c)(b + c) = (a+b)(a' + c)$

- $a + a'b = a + b$
- $a.(a'+b) = a. b$

Q The number of min-terms after minimizing the following Boolean expression is _____ . (GATE-2015) (1 Marks)

$$[D' + AB' + A'C + AC'D + A'C'D]'$$

- (A) 1 (B) 2 (C) 3 (D) 4

	ab	a'b'	a'b	ab	ab'
cd		00	01	11	10
c'd'	00		0	4	12
c'd	01		1	5	13
cd	11		3	7	15
cd'	10		2	6	14

Q Consider an expression & minimize it

$$a + a'b + a'b'c + a'b'c'd'$$

Q Consider the following Boolean expression:

$$F = (x+y+z)(x'+y)(y'+z)$$

Which of the following Boolean expressions is/are equivalent to F' ? (GATE-2021) (2 Marks)

(A) $xy' + yz' + x'y'z'$

(B) $(x'+y'+z')(x+y')(y+z')$

(C) $(x+z')(y'+z')$

(D) $xy' + z'$

	ab	a'b'	a'b	ab	ab'
cd		00	01	11	10
c'd'	00		0	4	12
c'd	01		1	5	13
cd	11		3	7	15
cd'	10		2	6	14

Q Consider a Boolean function $f(w,x,y,z)$ such that (GATE-2021) (2 Marks)

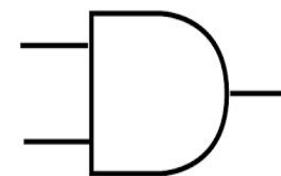
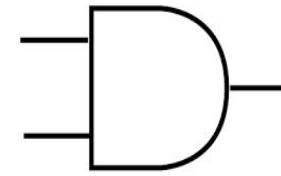
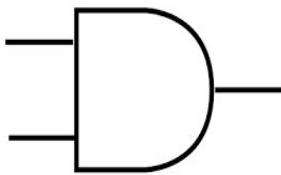
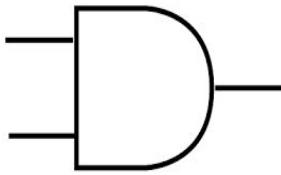
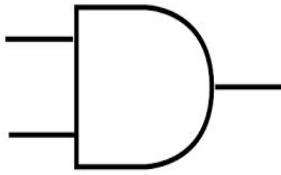
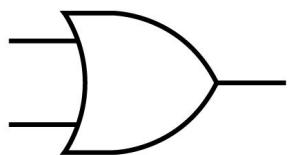
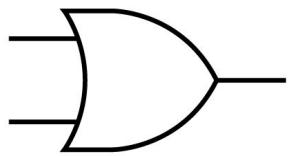
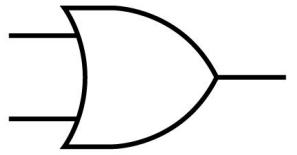
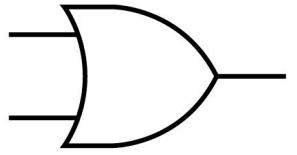
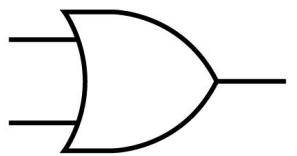
$$f(w,0,0,z) = 1$$

$$f(1,x,1,z) = x+z$$

$$f(w,1,y,z) = wz+y$$

The number of literals in the minimal sum-of-products expression of f is _____

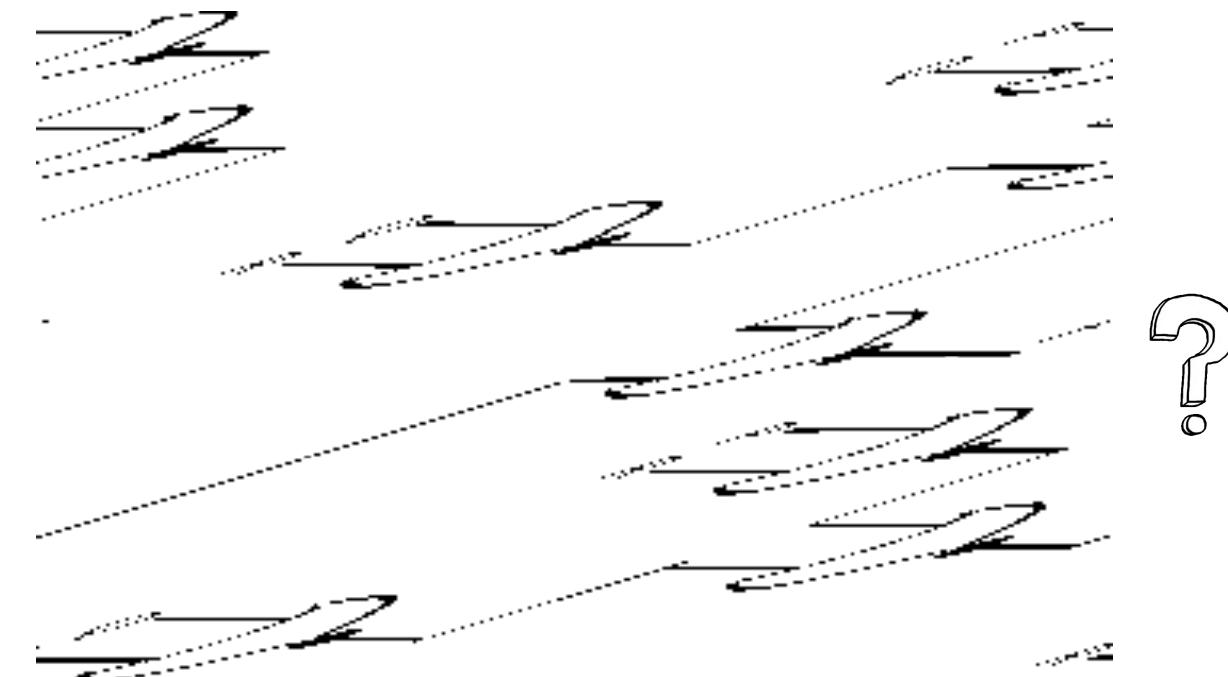
	w	x	y	z	F
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	
6	0	1	1	0	
7	0	1	1	1	
8	1	0	0	0	
9	1	0	0	1	
10	1	0	1	0	
11	1	0	1	1	
12	1	1	0	0	
13	1	1	0	1	
14	1	1	1	0	
15	1	1	1	1	



AND-OR(NAND-NAND) Implementation

OR-AND(NOR-NOR) Implementation

Q What is the Boolean expression for the output f of the combinational logic circuit of NOR gates given below? **(GATE-2010) (1 Marks)**



- (A) $(Q+R)'$
- (B) $(P+Q)'$
- (C) $(P+R)$
- (D) $(P+Q+R)'$

Implementing Every Gate with NAND Gate

NOT

AND

Implementing Every Gate with NAND Gate

OR

NOR

Implementing Every Gate with NAND Gate

EX-OR

EX-NOR

Implementing Every Gate with NOR Gate

NOT

AND

Implementing Every Gate with NOR Gate

OR

NAND

Implementing Every Gate with NOR Gate

EX-OR

EX-NOR

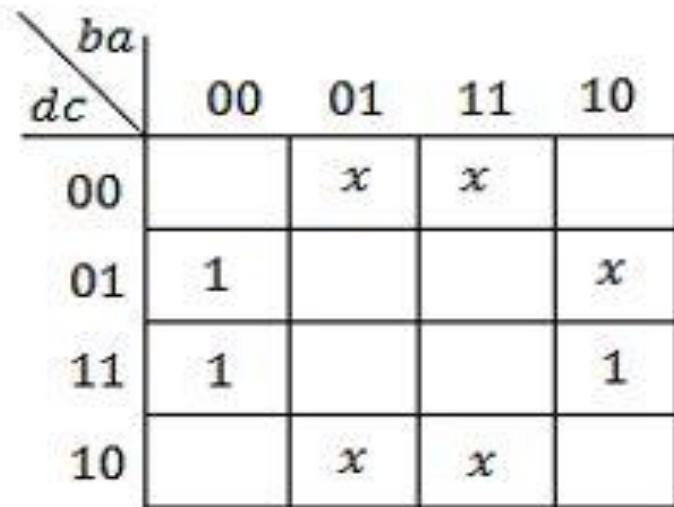
Conclusion

	NOT	AND	OR	NAND	NOR	EX-OR	EX-NOR
NOR							
NAND							

Q What is the minimum number of NAND gates required to implement a 2-input EXCLUSIVE-OR function without using any other logic gate? **(GATE–2004) (1 Marks)**

- (A) 3
- (B) 4
- (C) 5
- (D) 6

Q Consider the Karnaugh map given below, where X represents “don’t care” and blank represents 0. Assume for all inputs (a, c, d) the respective complements (a' , b' , c' , d') are also available. The above logic is implemented 2-input NOR gates only. The minimum number of gates required is _____. (GATE-2017) (1 Marks)



Q what is the minimum number of gates required to implement the Boolean function $(AB+C)$ if we have to use only 2-input NOR gates? **(GATE-2009) (1 Marks)**

- (A) 2
- (B) 3
- (C) 4
- (D) 5

Functionally Complete Function

- As we know there are only three fundamental Boolean operator NOT, AND and OR. All the other operators are derived from these operators
- We understand NOT along with OR(NOR) and NOT along with AND(NAND) are used as universal gates.
- Any Boolean operation or function can be implemented with NAND or NOR operation.
- If any function can implement NOT along with it either AND or OR then, we indirectly prove that the function can also implement any other function, so function can said to be functionally complete.
- Support of 0, 1 or complemented form of variables are not allowed.

Q Consider the operations (GATE-2015) (2 Marks)

$$f(X, Y, Z) = X'YZ + XY' + Y'Z'$$

$$g(X, Y, Z) = X'YZ + X'YZ' + XY$$

Which one of the following is correct?

- (A) Both {f} and {g} are functionally complete
- (B) Only {f} is functionally complete
- (C) Only {g} is functionally complete
- (D) Neither {f} nor {g} is functionally complete

Q which of the following is functionally complete?

- a) \oplus , not
- b) \oplus , 1, +
- c) \oplus , 1, not
- d) \odot , 1, not

$Q f(a, b) = a' + b$ is functionally complete ?

$Q f (a, b, c) = a' + bc'$ is functionally complete

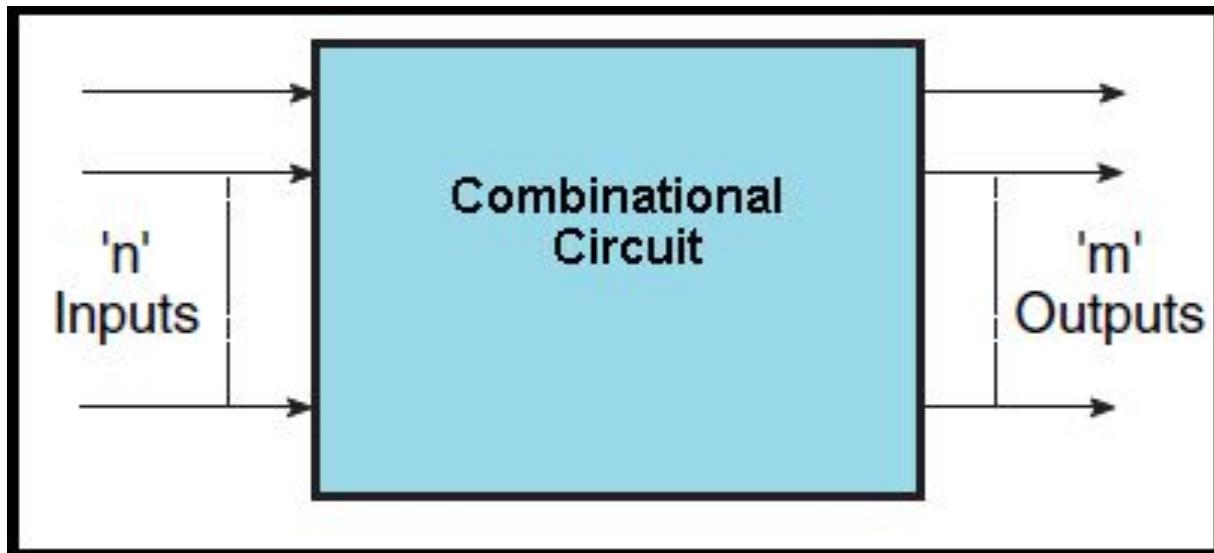
Partially Functionally Complete: - A function is said to be partially functionally complete, if it can be implement any digital circuit with support of logic 0 or 1 as a input line (can't use complemented form).

Q f (a, b) = a' + b is functionally complete?

Q f (a, b) = ab' + a'b not fully or partially complete

Q f (a, b, c) = ab + bc +ca is not fully or partially complete

Combinational Circuits

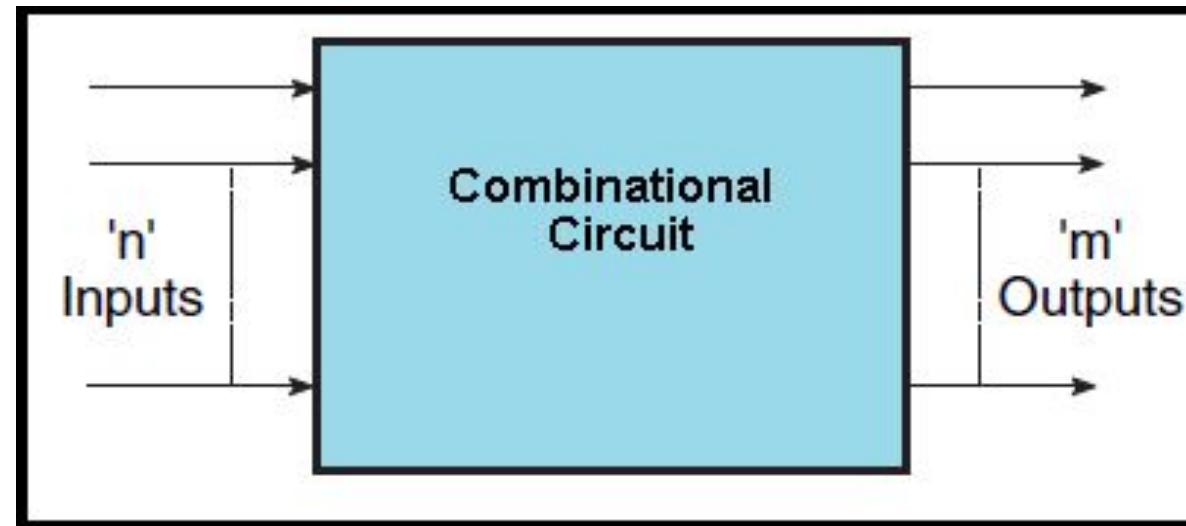


$$O/p = f(i/p)$$

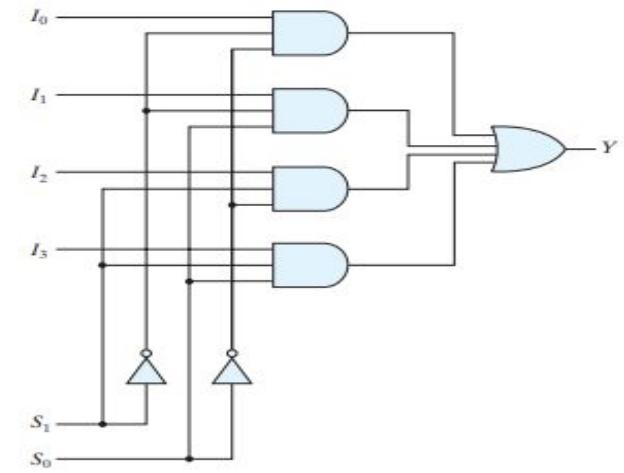
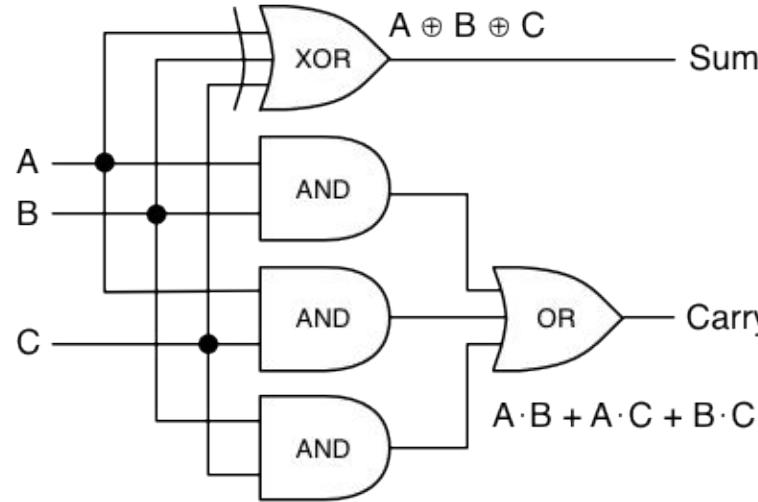
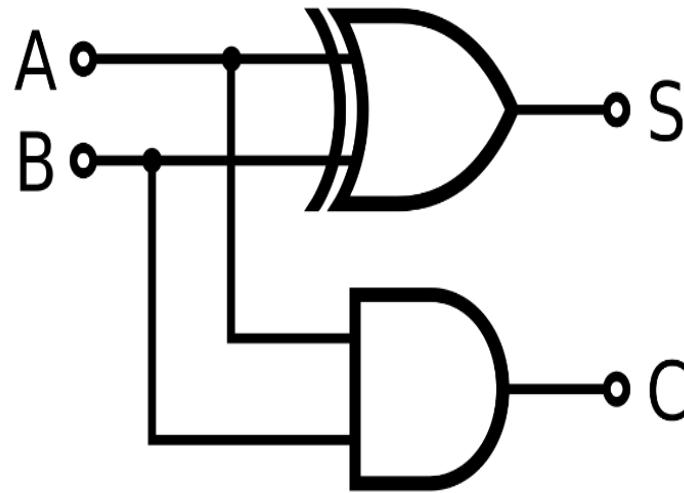


Combinational Circuits

- When logic gates are connected together to produce a specified output on certain specified combinations of input variables, with no memory involved, then the resulting circuit is called a combinational circuit.
- Output depends only on present input. $O/p = f(i/p)$, combinational circuit performs an operation that can be specified logically by a set of Boolean function. A combinational circuit may have n -binary inputs and m -binary outputs.
- Application, Practical computer circuits normally contain combinational and sequential logic. For e.g. the part of ALU, that does mathematical calculations is constructed using combinational logic.



- Other circuits such as half adders, full adders, half subtractors, full subtractors, multiplexers, demultiplexers, encoders and decoders are also made by using combinational logic.

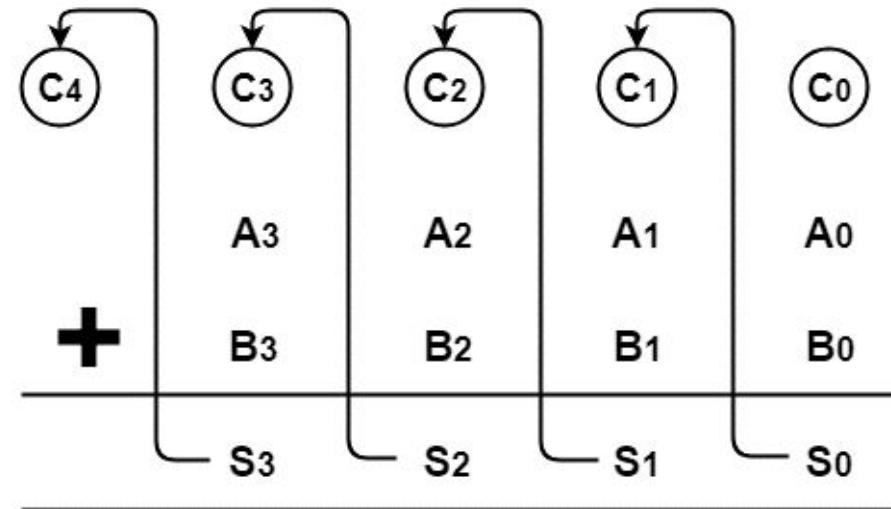


Design procedure: -

1. Analyse the given problem and identify the number of i/p and o/p variables.
2. Write the truth table based on the specification of the problem.
3. Convert the truth table in minimized Boolean expression using k-map.
4. Draw the logic circuit for the above obtained output expression.

Adder

- An **adder** is a digital combinational circuit that performs addition of numbers. Are used in the arithmetic logic units or ALU.
- They are also utilized in other parts of the processor, where they are used to calculate addresses, table indices, increment and decrement operators, and similar operations.
- Although adders can be constructed for many number representations, such as binary-coded decimal or excess-3, the most common adders operate on binary numbers.
- In cases where two's complement or ones' complement is being used to represent negative numbers, it is trivial to modify an adder into an adder–subtractor.

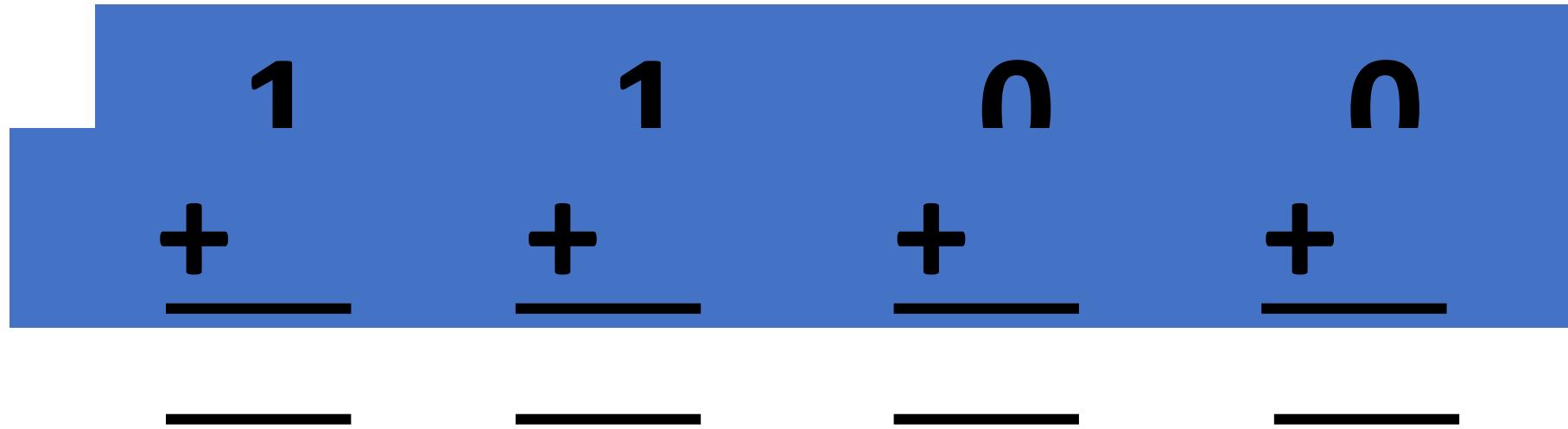


- Basics of addition: - $(A)_x + (B)_x = ?$



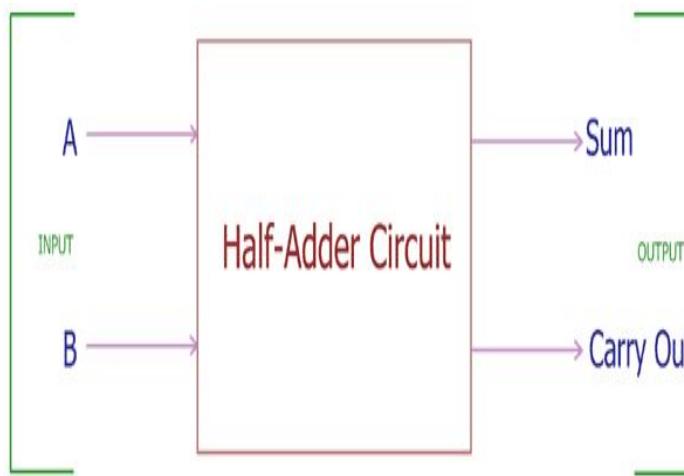
Half adder

- The simplest form of addition is addition of two binary digits, consists of four possible elementary operations

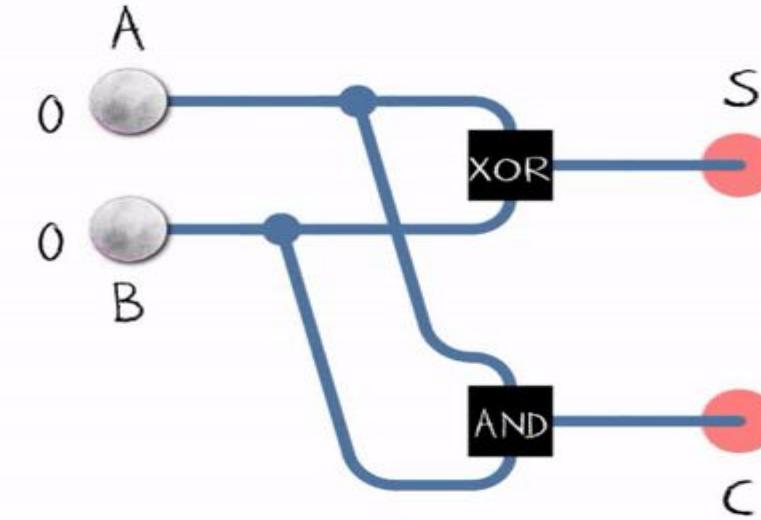
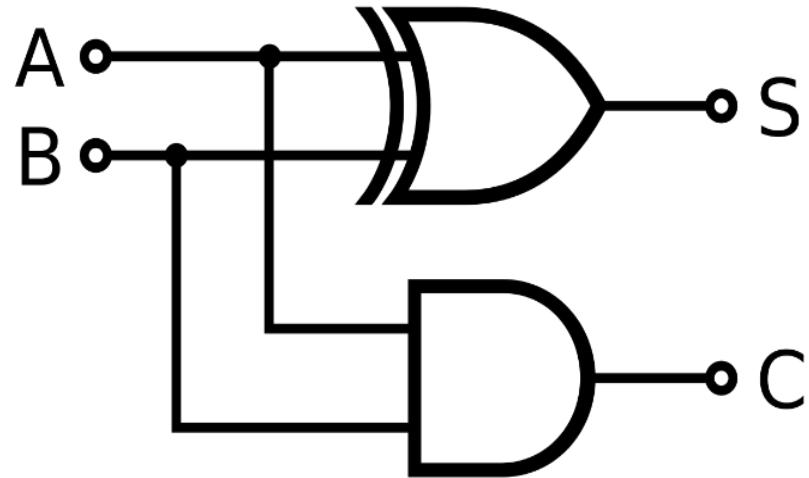


- The first three operations produce a sum of two digits, but when both augend and addend bits are equal to 1, the binary sum consists of two digits. The higher significant bit of this result is called a carry.

- It is a combinational circuit, which perform the arithmetic addition of two one-bit binary numbers is referred to as an half-adder.
- So, in half adder inputs are adds two single binary bits A and B , and two outputs, sum (S) and carry (C).



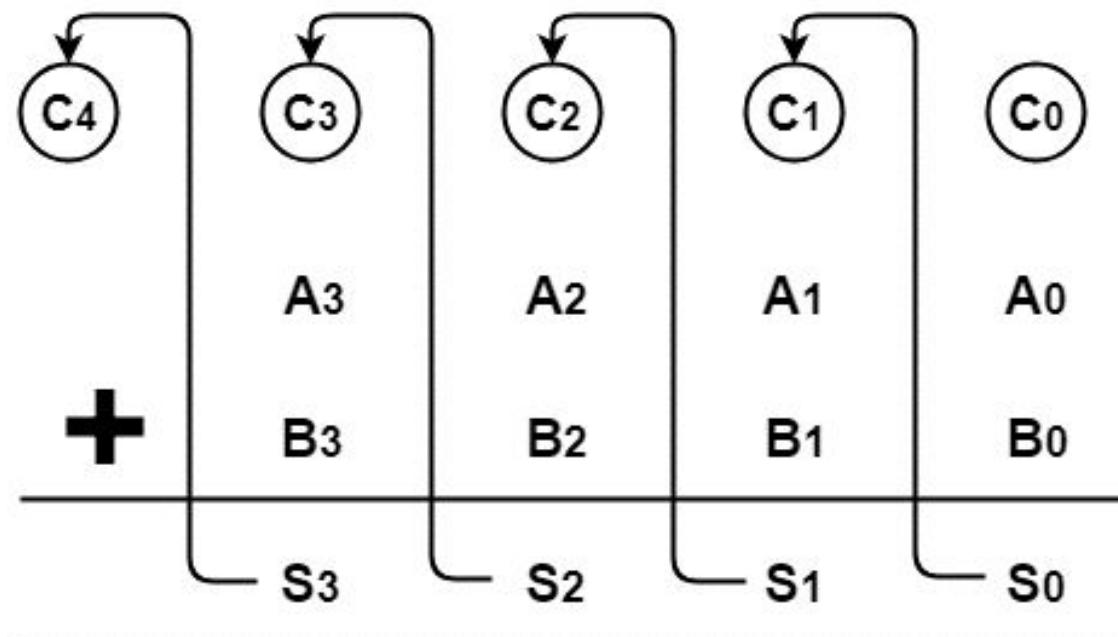
INPUTS		OUTPUTS	
A	B	Carry	Sum
0	0		
0	1		
1	0		
1	1		



1. Cost of implementation a half adder is one EX-OR gate and one AND gate.
2. A half adder has only two inputs and there is no provision to add a carry coming from the lower order bits when multi bit number addition is performed. For this reason, we have designed a full adder.

Full adder

1. A full adder is a combinational logic circuit that performs the arithmetic sum of three input bits.
2. Where A_n, B_n are the n^{th} order bits of the number A and B respectively and C_n is the carry generated from the addition of $(n-1)^{\text{th}}$ order bits.
3. It consists of three input bits, denoted by A (First operand), B (Second operand), C_{in} (Represents carry from the previous lower significant position).



- Two output bits are same as of half adder, which is Sum and Carry_{out}.
- When the augend and addend number contain more significant digits, the carry obtained from the addition of two bits is added to the next higher order pair of significant bits.

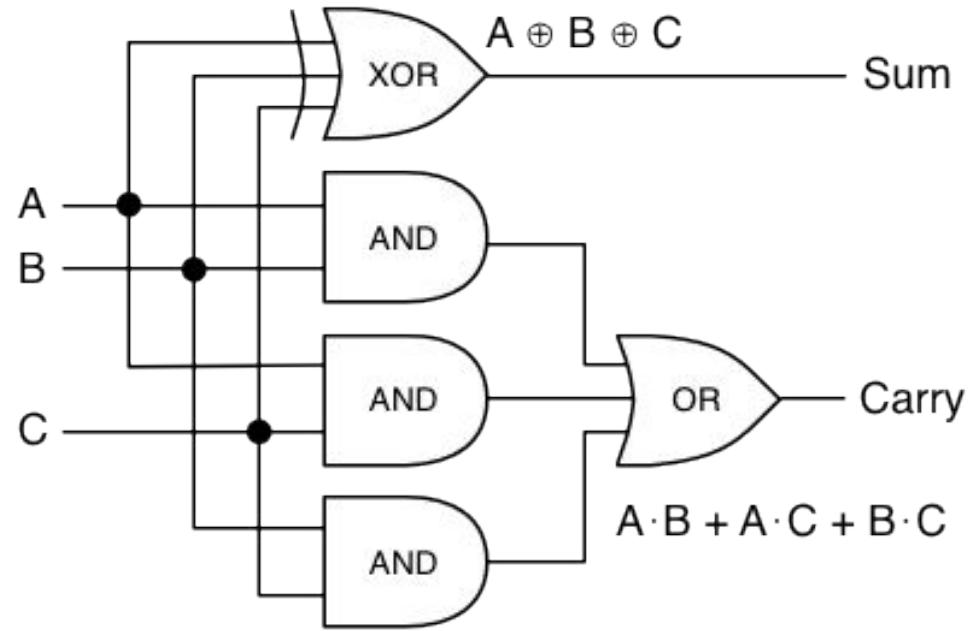


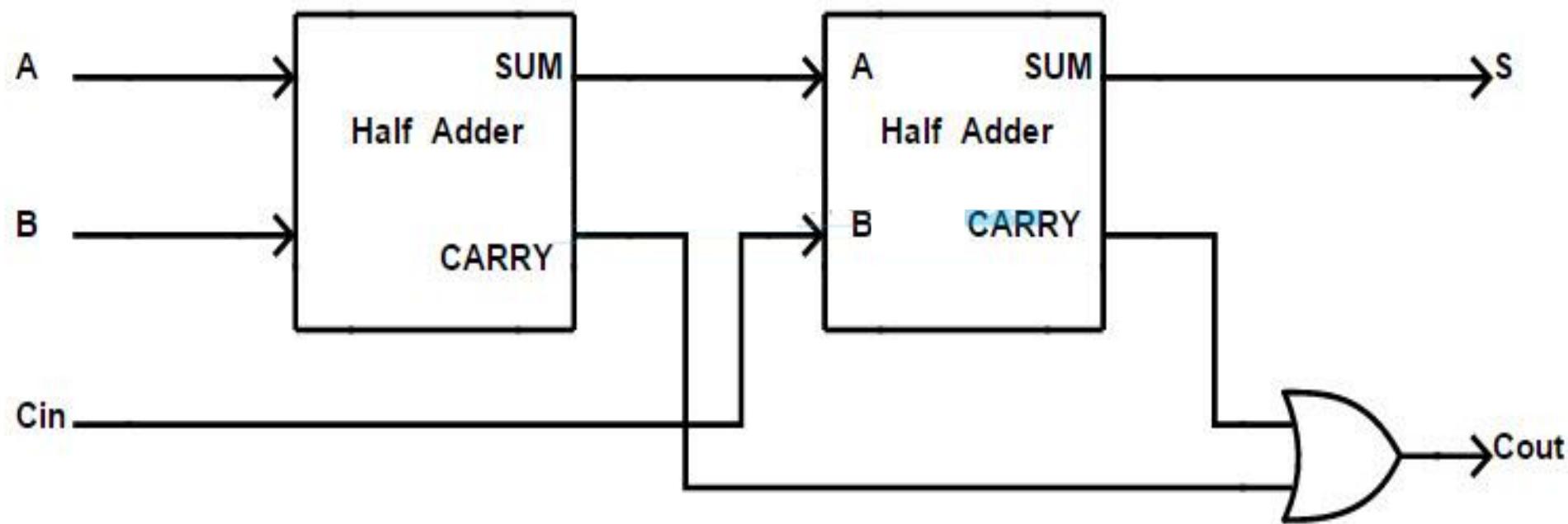
INPUTS			OUTPUTS	
A	B	C _{in}	C _{out}	Sum
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

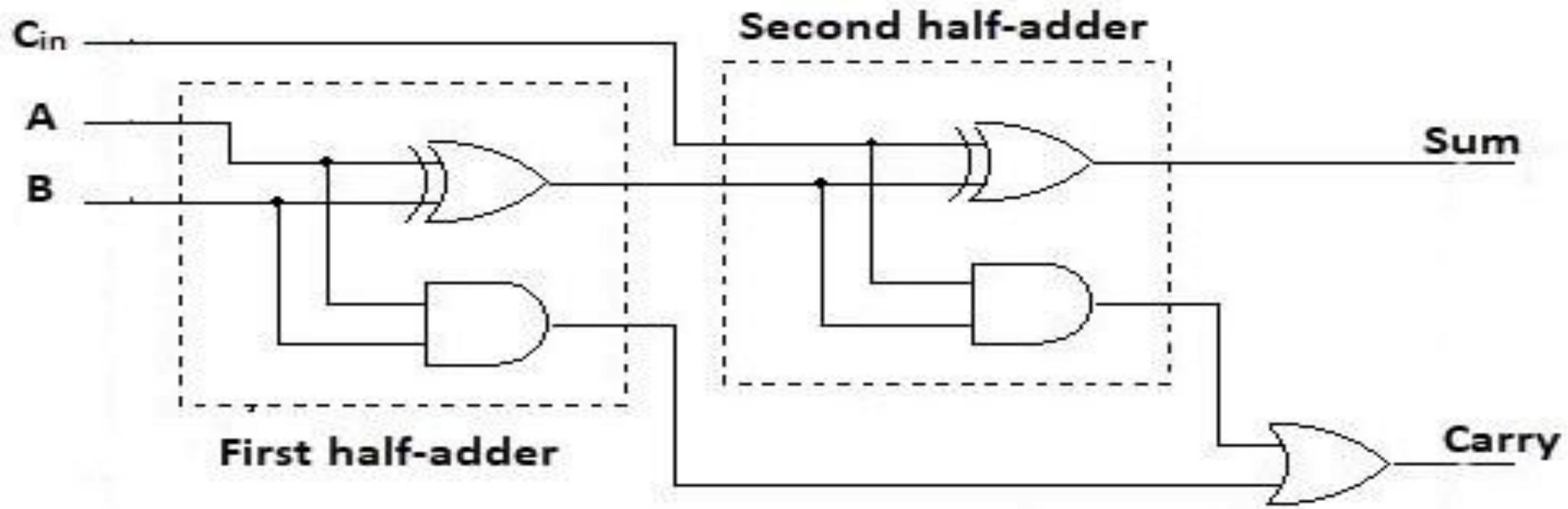
	ab	a'b'	a'b	ab	ab'
C _{in}		00	01	11	10
C _{in} '	0	0	2	6	4
C _{in}	1	1	3	7	5

	ab	a'b'	a'b	ab	ab'
C _{in}		00	01	11	10
C _{in} '	0	0	2	6	4
C _{in}	1	1	3	7	5

INPUTS			OUTPUTS	
A	B	C _{in}	Sum	C _{out}
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

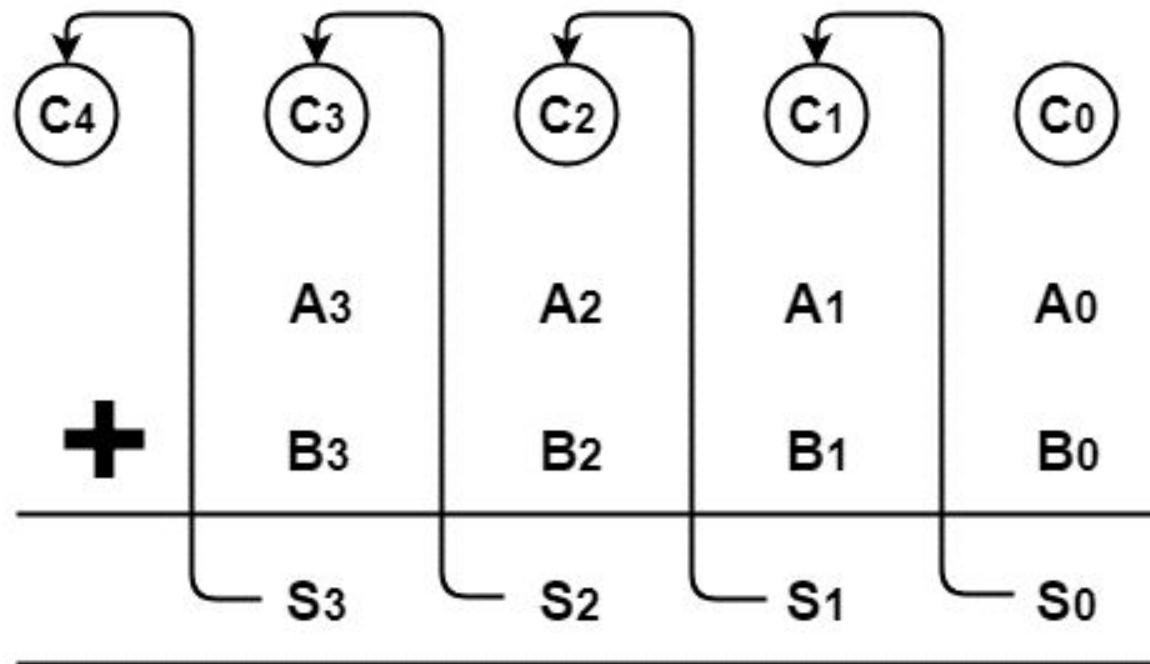




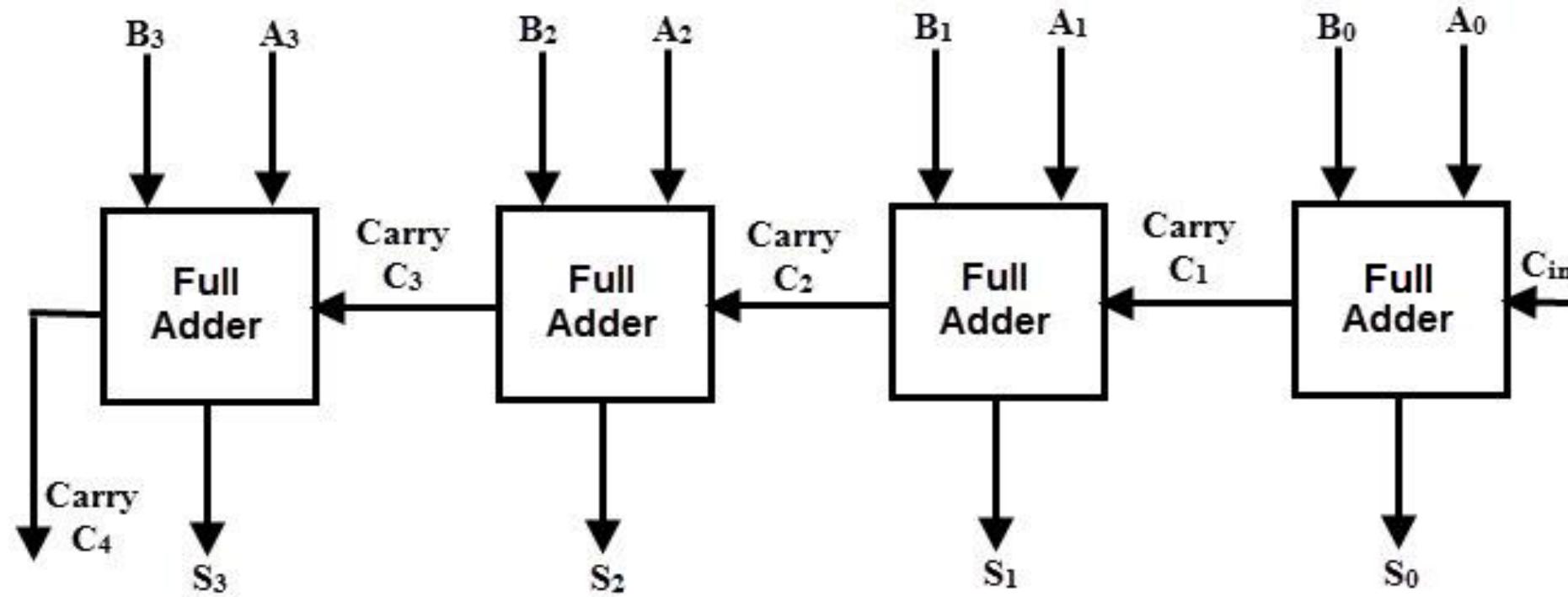


Four-bit parallel binary adder / Ripple adder

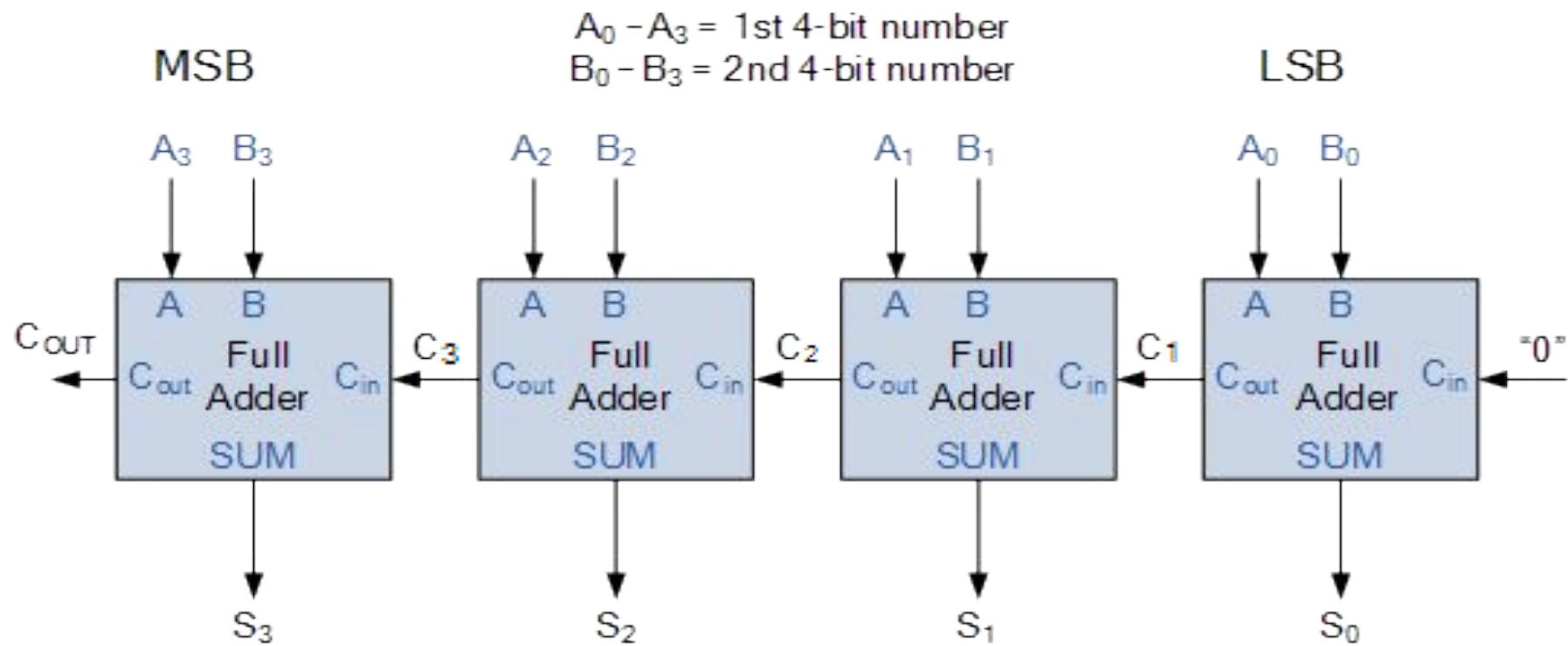
- As we know that full adder is capable of adding two 1 bit number and 1 previous carry, but in order to add binary numbers with more than one bits, additional full adders must be employed. For e.g. a four bit binary adder can be constructed using four full adders.



- These four full adders are connected in cascade, carry output of each adder is connected to the carry input of the next higher-order adder.
- So a n-bit parallel adder is constructed using 'n' number of full adders.



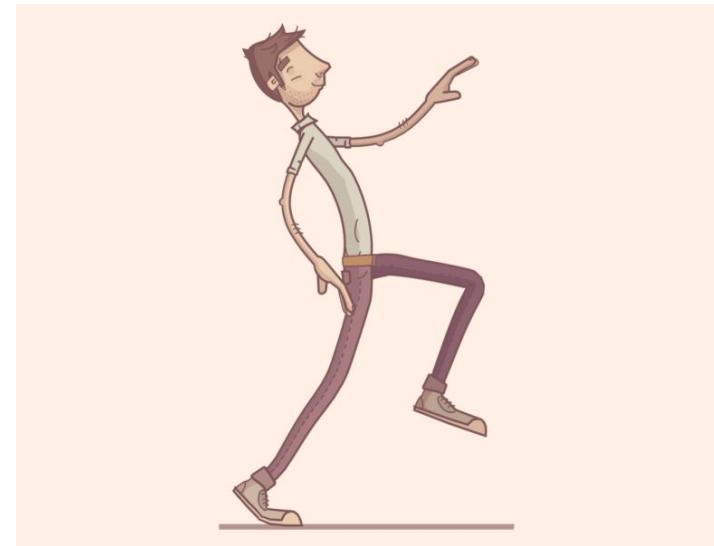
- The longest propagation delay time in an adder is the time it takes the carry to propagate through the full adders. For carry in a 4-bit adder we have 2 gate delays at each adder and for sum we have 1 gate delay.
 - For an n -bit adder, there are **$2n$** gate levels for the **carry to propagate from input to output**. And for Sum we have **$2n-1$** Gate delays.



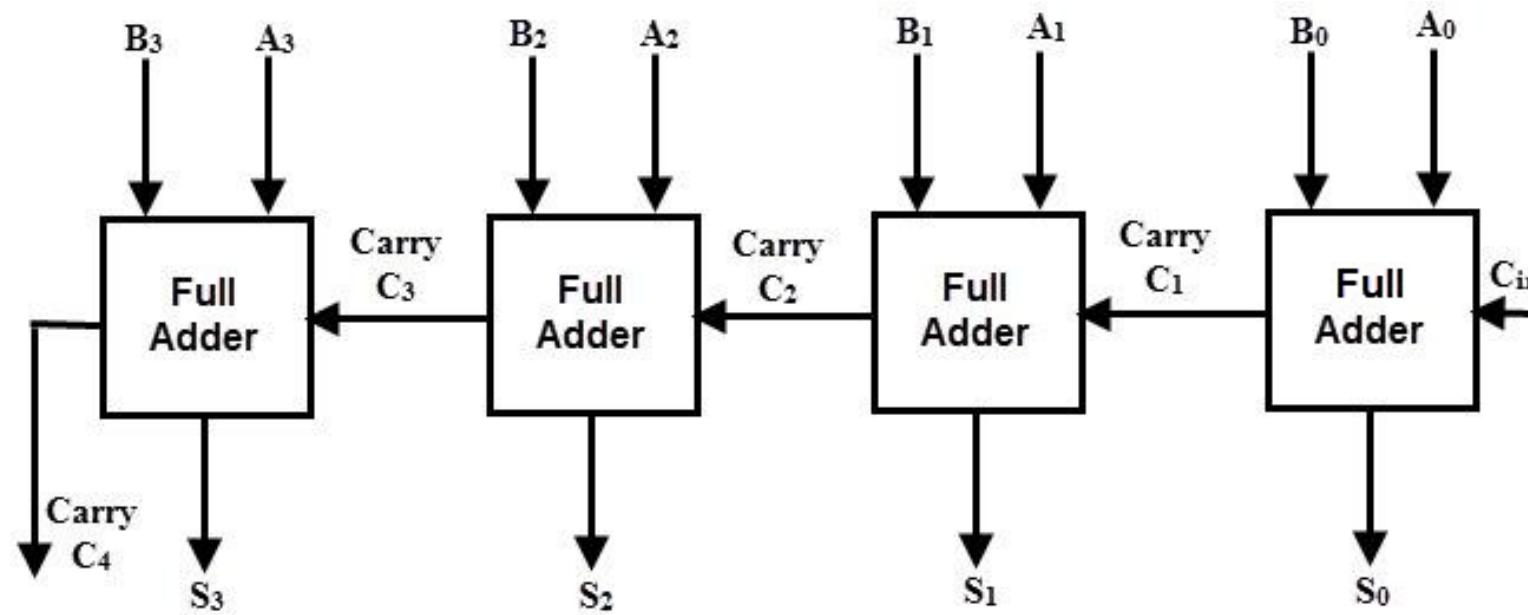
- There are some scope of improvement in adder like

Carry propagation delay

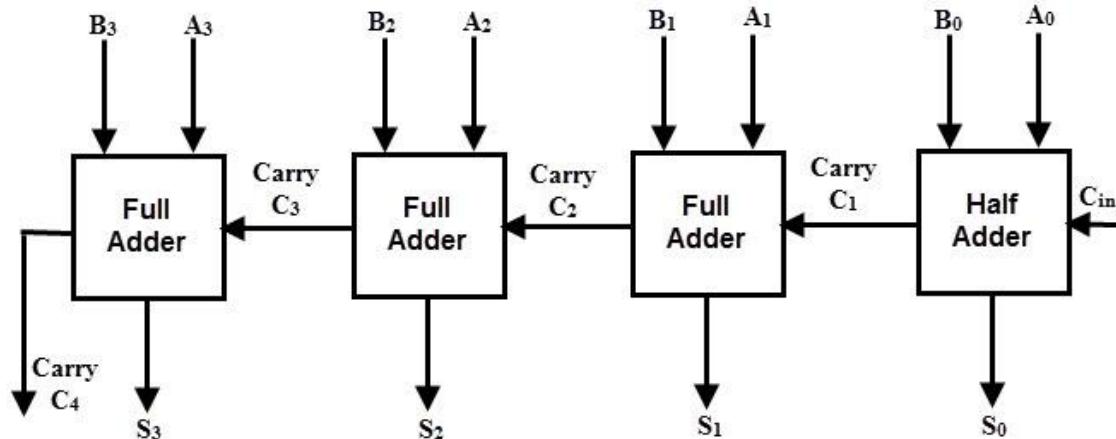
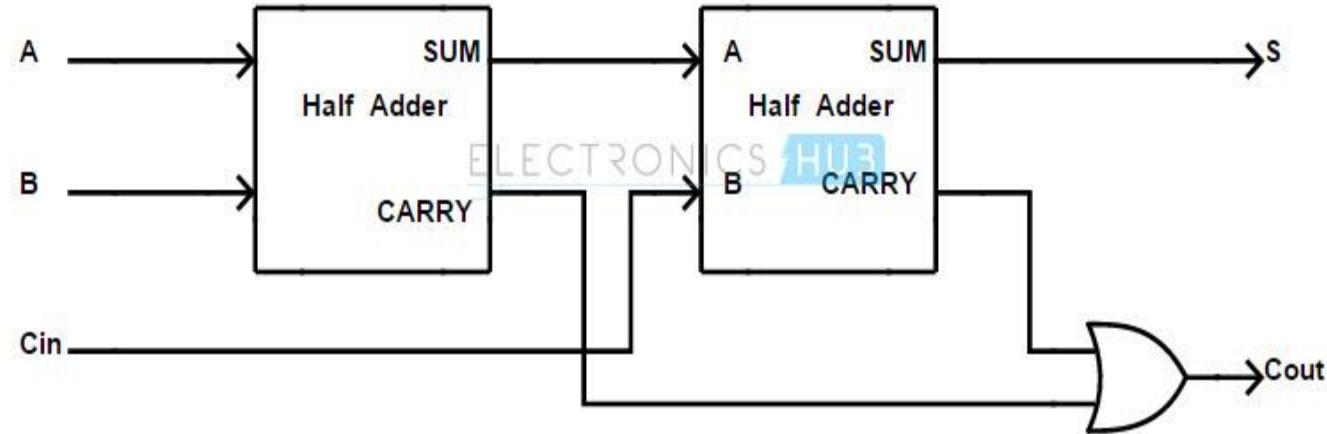
Look ahead Carry Generator

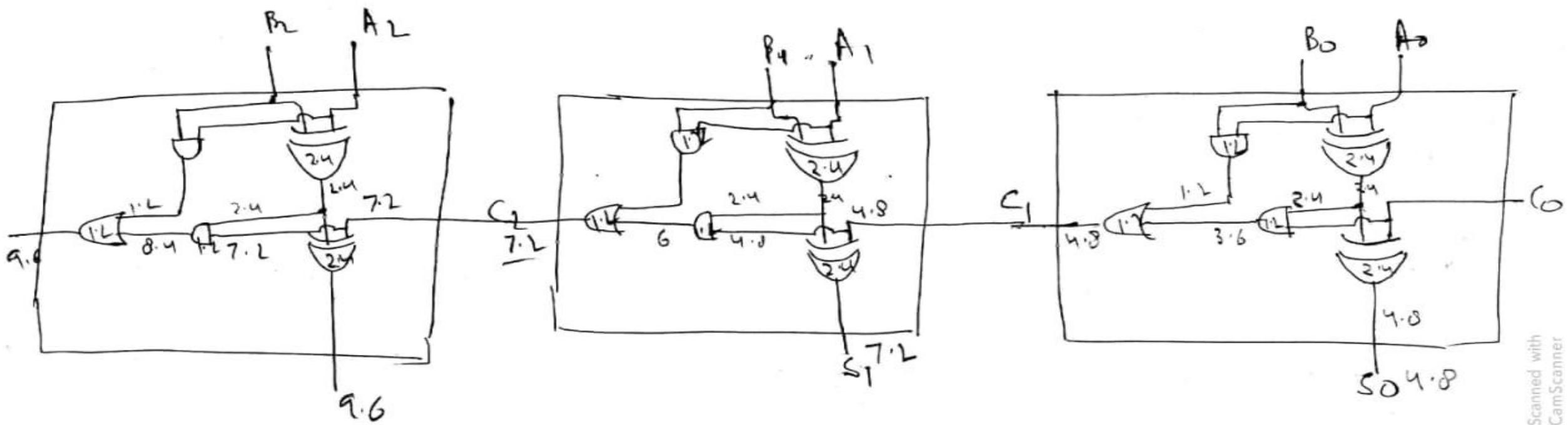


- Can adder be modified to work as subtractor
 - Adder/subtractor or ripple adder



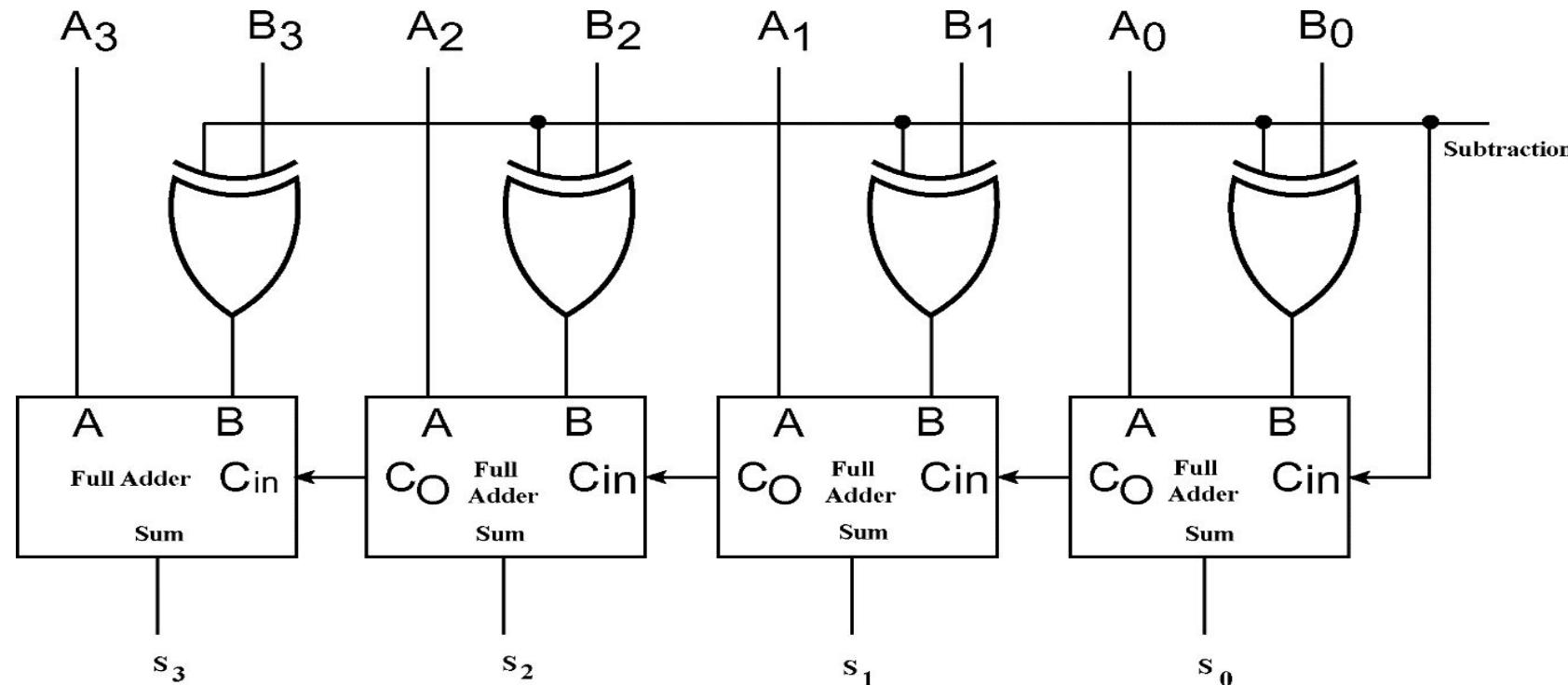
Q A half adder is implemented with XOR and AND gates. A full adder is implemented with two half adders and one OR gate. The propagation delay of an XOR gate is twice that of an AND/OR gate. The propagation delay of an AND/OR gate is 1.2 microseconds. A 4-bit ripple-carry binary adder is implemented by using full adders. The total propagation time of this 4-bit binary adder in microseconds is **(GATE-2015) (2 Marks)**



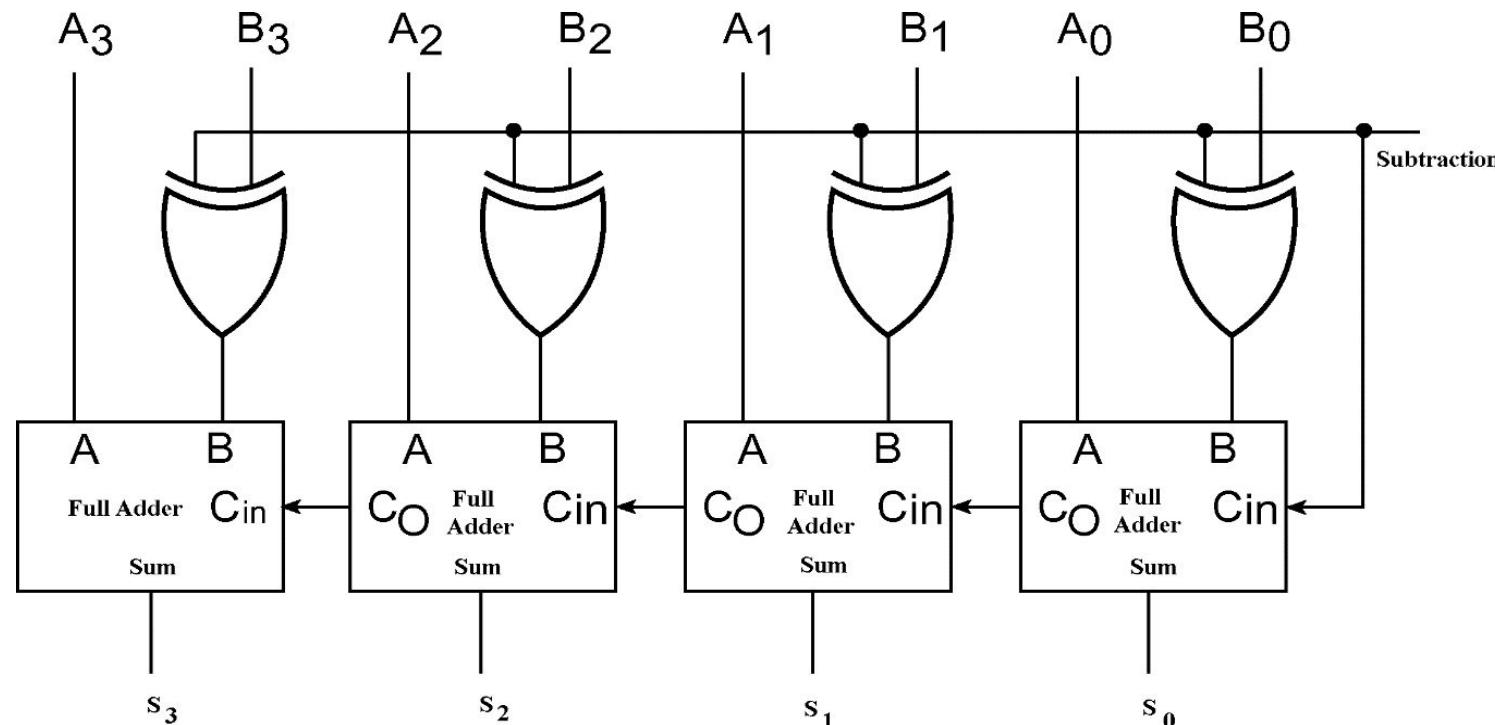


Four-bit ripple adder/subtractor

- The subtraction $A - B$ can be done by taking the 2's complement of B and adding it to A . The 2's complement can be obtained by taking the 1's complement and adding 1 to the least significant pair of bits.
- The 1's complement can be implemented with inverters, and a 1 can be added to the sum through the input carry. The circuit for subtracting $A - B$ consists of an adder with inverters placed between each data input B and the corresponding input of the full adder.

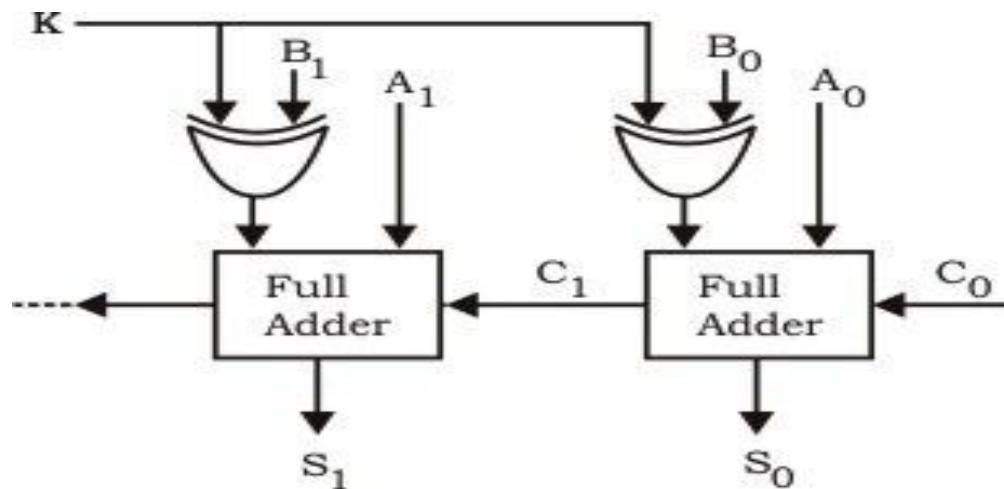


- The mode input M controls the operation. When $M = 0$, the circuit is an adder, and when $M = 1$, the circuit becomes a subtractor.
- When $M = 0$, we have $B \oplus 0 = B$. The full adders receive the value of B, the input carry is 0, and the circuit performs A plus B.
- When $M = 1$, we have $B \oplus 1 = B'$ and $C_0 = 1$. The B inputs are all complemented and a 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B.



Q Consider an eight-bit ripple-carry adder for computing the sum of A and B, where A and B are integers represented in 2's complement form. If the decimal value of A is one, the decimal value of B that leads to the longest latency for the sum to stabilize is _____ (GATE-2016) (2 Marks)

Q Consider the ALU shown below.



If the operands are in 2's complement representation, which of the following operations can be performed by suitably setting the control lines K and C_0 only (+ and – denote addition and subtraction respectively)? (**GATE-2007**)
(2 Marks)

(A) $A + B$, and $A - B$, but not $A + 1$

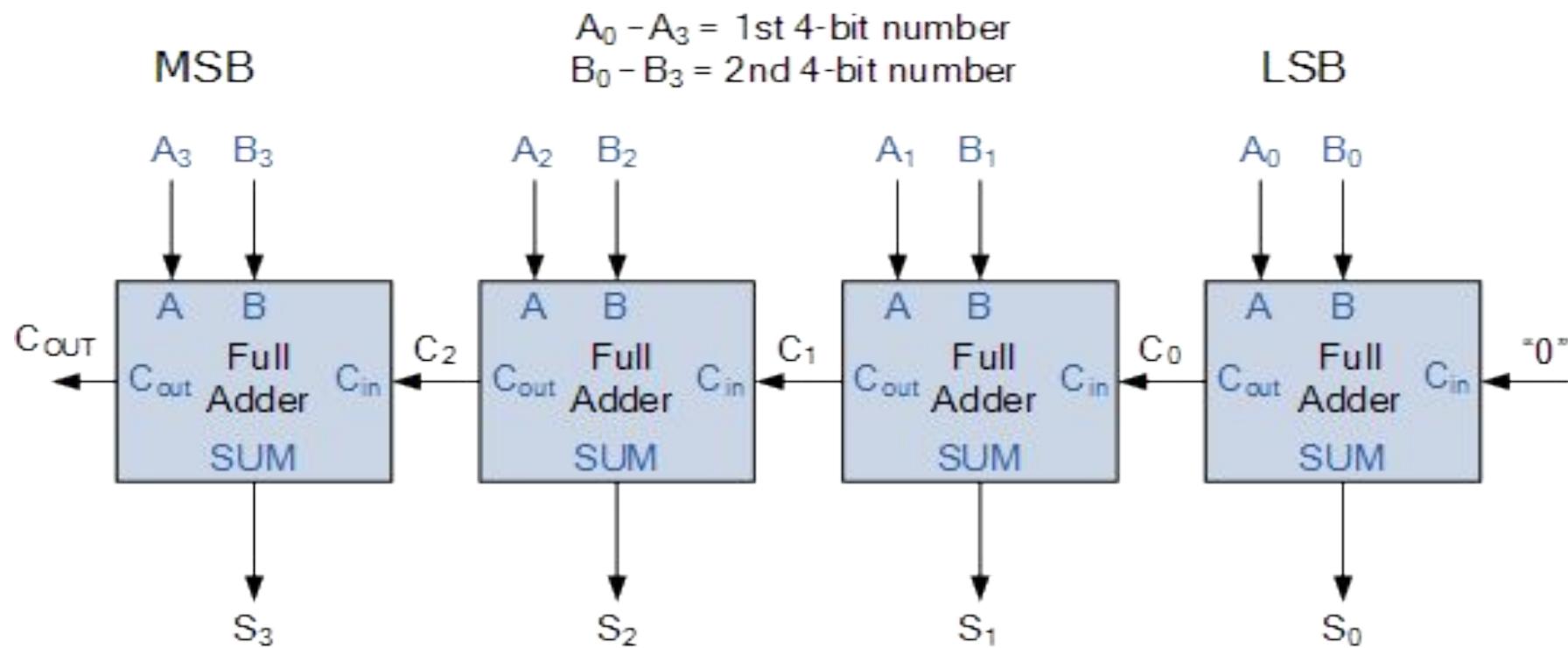
(B) $A + B$, and $A + 1$, but not $A - B$

(C) $A + B$, but not $A - B$, or $A + 1$

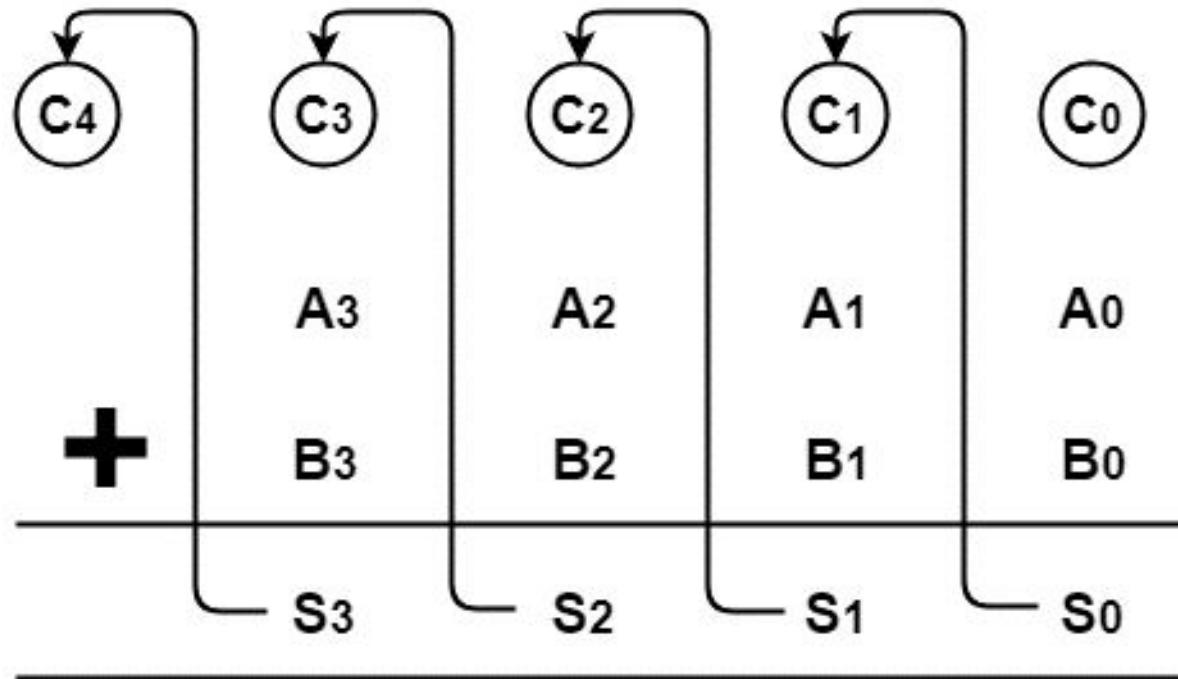
(D) $A + B$, and $A - B$, and $A + 1$

Look ahead carry adder

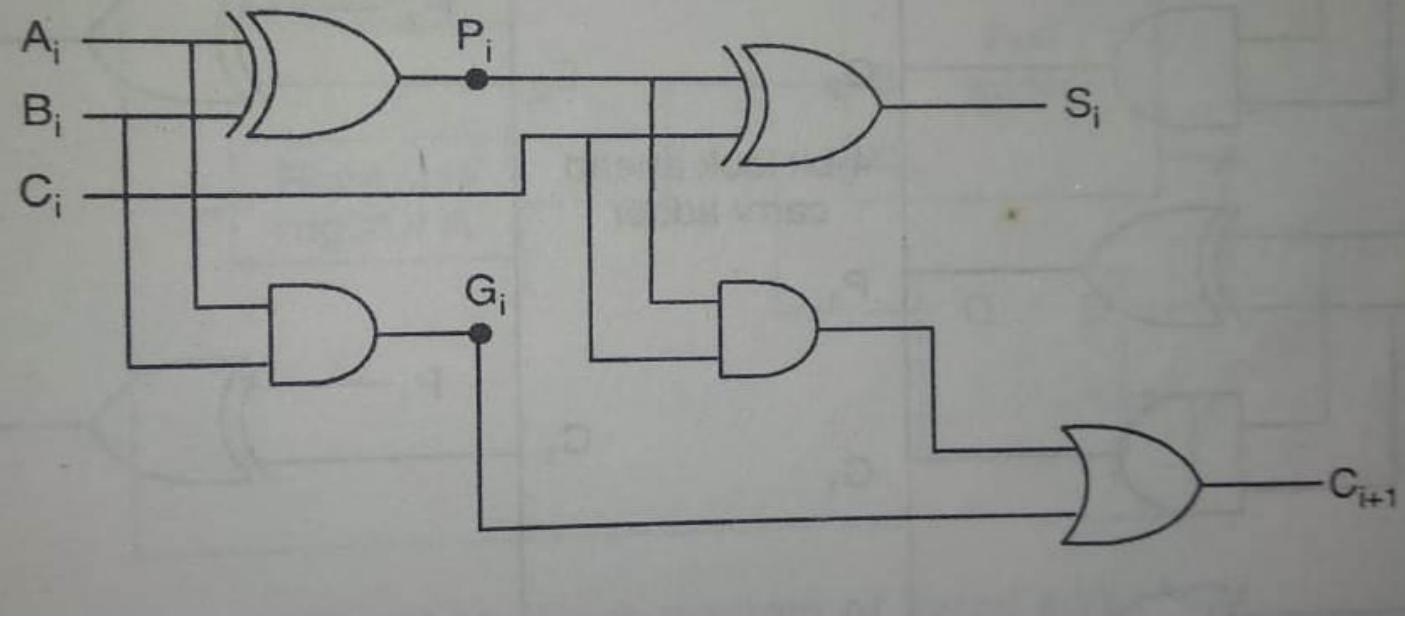
- In parallel adder all bits of augend and addend are available for computation initially, but sum and carry outputs of any stage cannot be produced until the input carry occurs. This leads to delay in the addition process known as carry propagation delay.
 - In any combinational circuit, the signal must propagate through the gates before the correct output sum is available in the output terminals. **The total propagation time is equal to the propagation delay of a typical gate * times the number of gate levels in the circuit.**



- The carry propagation time is an important attribute of the adder because it limits the speed with which two numbers are added. The solution to delay is to increase the complexity of the equipment in such a way that the carry delay time is reduced.
- To solve this problem most widely used technique employs the principle of 'look ahead carry'. This method utilizes logic gates to look at the lower order bits of the augend and addend to see if a higher order carry is to be generated. It uses two functions carry generate G_i and carry propagate P_i



$$\begin{aligned} A(A_3 & A_2 & A_1 & A_0) \\ B(B_3 & B_2 & B_1 & B_0) \end{aligned}$$



- G_i is called a ***carry generate***, and it produces a carry of 1 when both A_i and B_i are 1, regardless of the input carry C_i .
- P_i is called a ***carry propagate***, because it determines whether a carry into stage i will propagate into stage $i + 1$.
- We now write the Boolean functions for the carry outputs of each stage and substitute the value of each C_i from the previous equations:

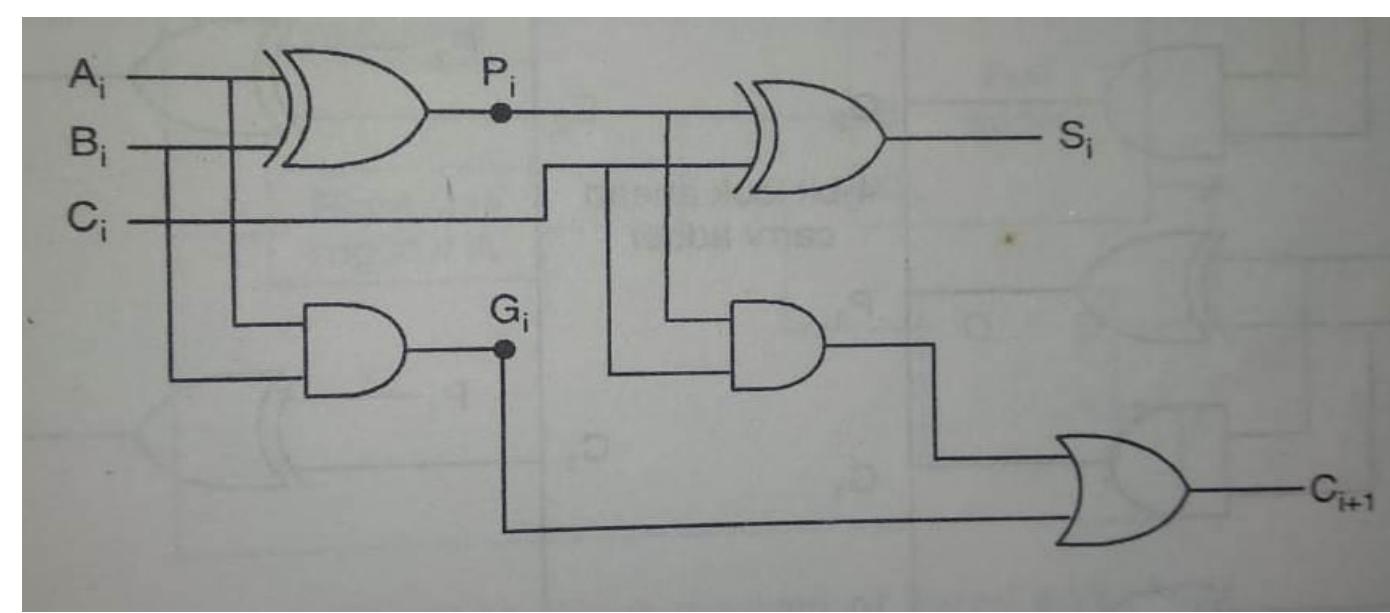
- $P_i = A_i \oplus B_i$
 $G_i = A_i \cdot B_i$
 $S_i = P_i \oplus C_i$
 $C_{i+1} = G_i + P_i C_i$

- $C_0 =$

- $C_1 =$

- $C_2 =$

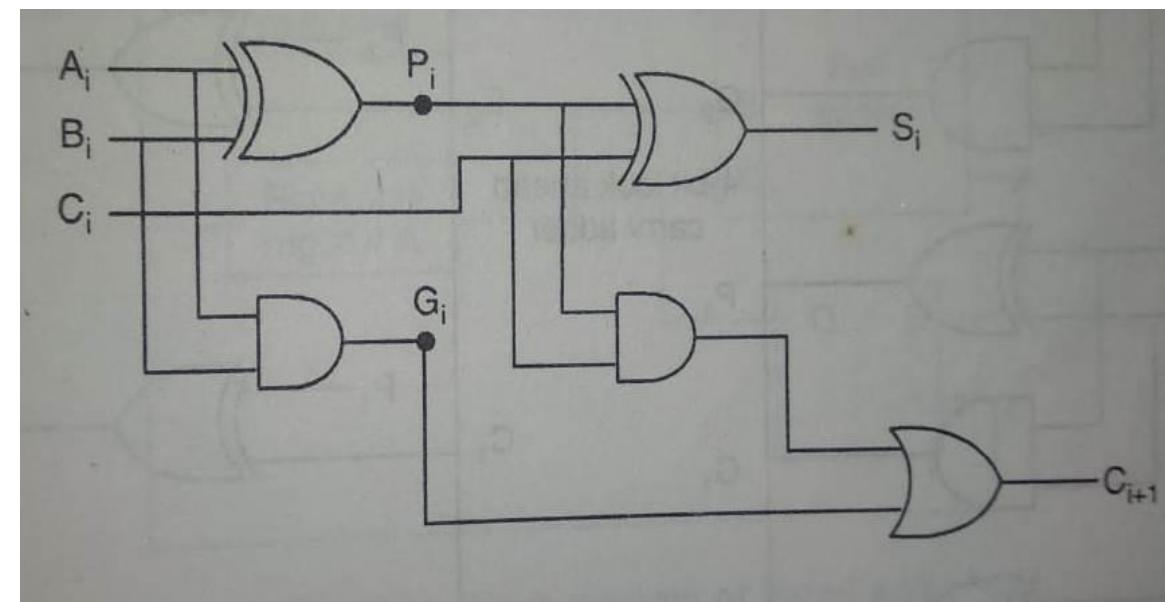
- $C_3 =$



- G_i is called a ***carry generate***, and it produces a carry of 1 when both A_i and B_i are 1, regardless of the input carry C_i .
- P_i is called a ***carry propagate***, because it determines whether a carry into stage i will propagate into stage $i + 1$.
- We now write the Boolean functions for the carry outputs of each stage and substitute the value of each C_i from the previous equations:

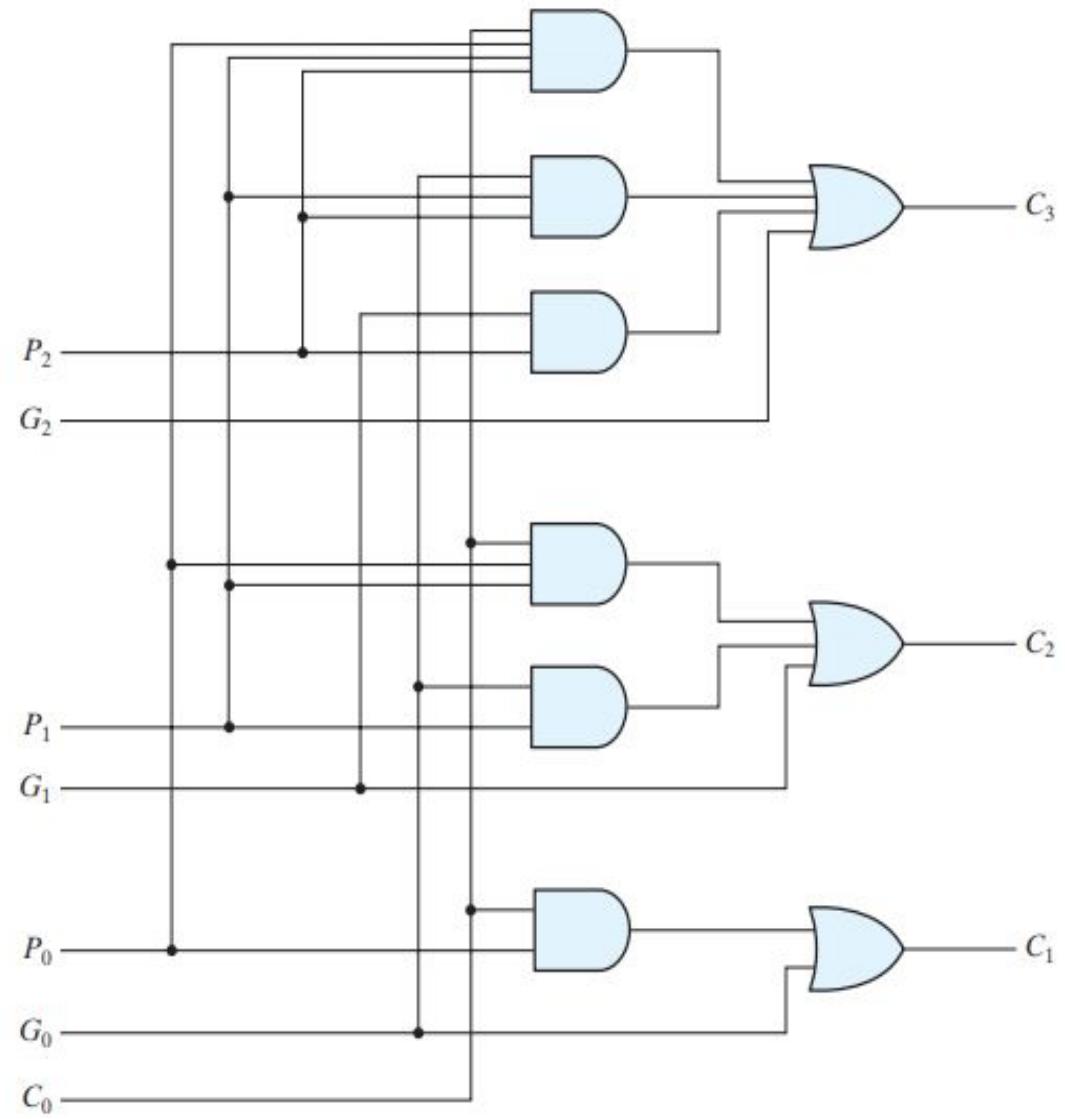
- $P_i = A_i \oplus B_i$
- $G_i = A_i \cdot B_i$
- $S_i = P_i \oplus C_i$
- $C_{i+1} = G_i + P_i C_i$

- $C_0 = 0$
- $C_1 = G_0 + P_0 C_0$
- $C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$
- $C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$
- $C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3$

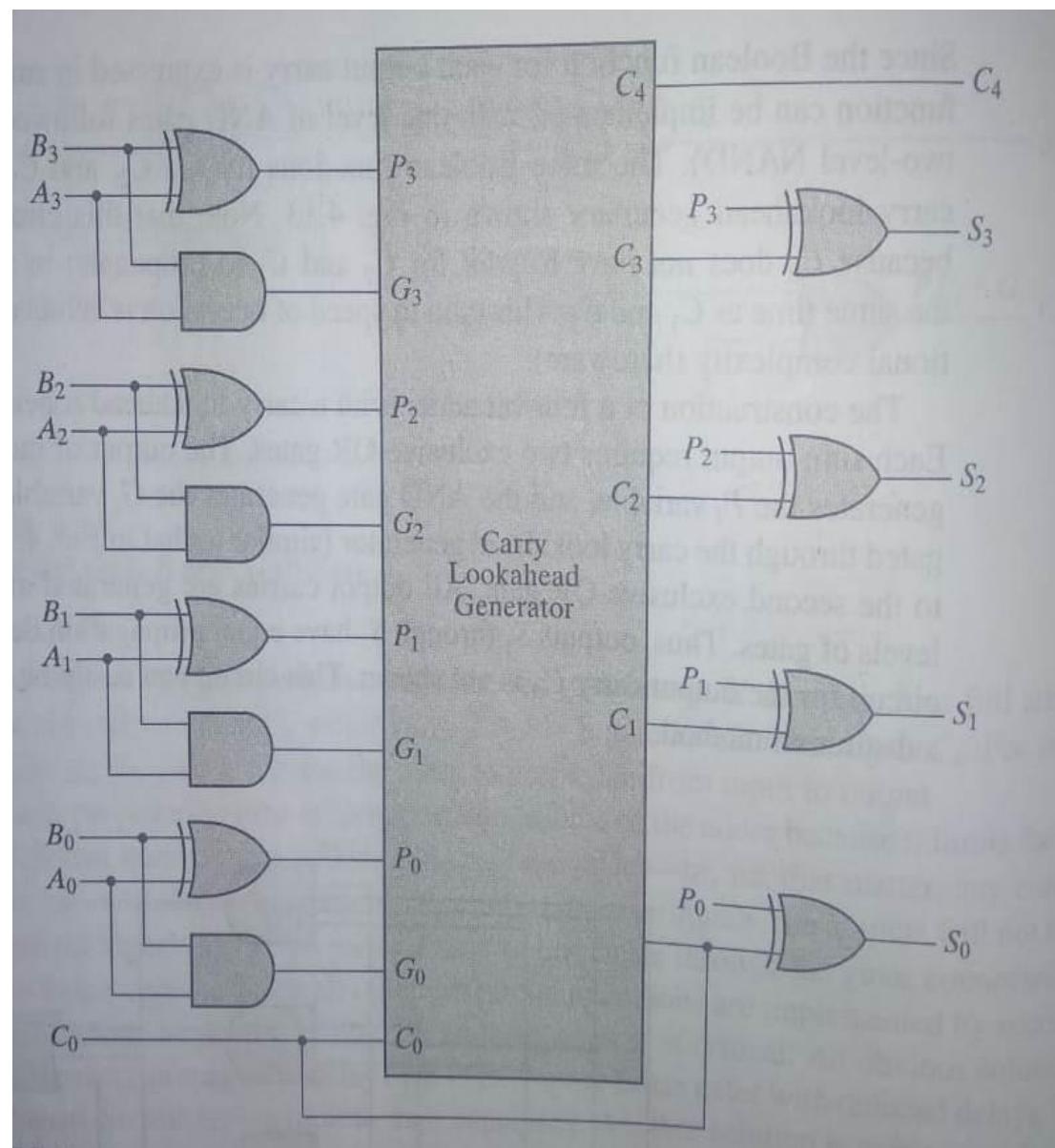
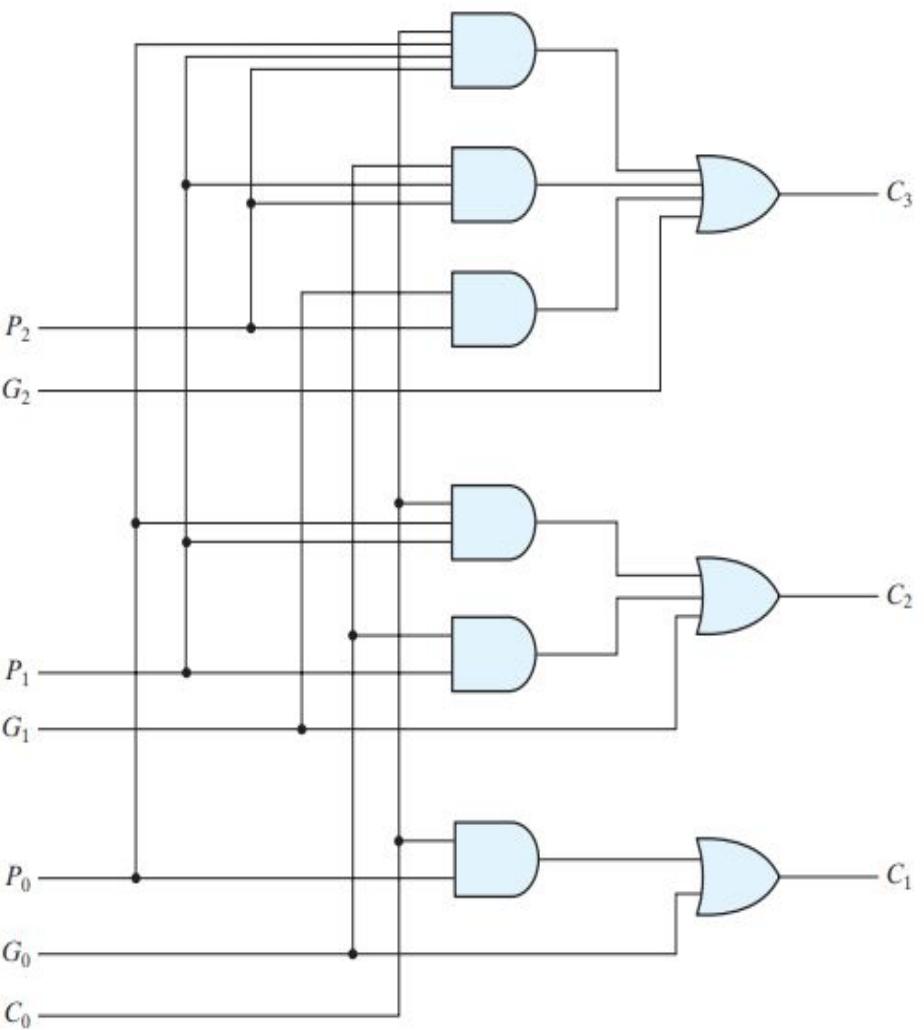


- Since the Boolean function for each output carry is expressed in sum-of-products form. Each function can be implemented with one level of AND gates followed by an OR gate (or by a two-level NAND).

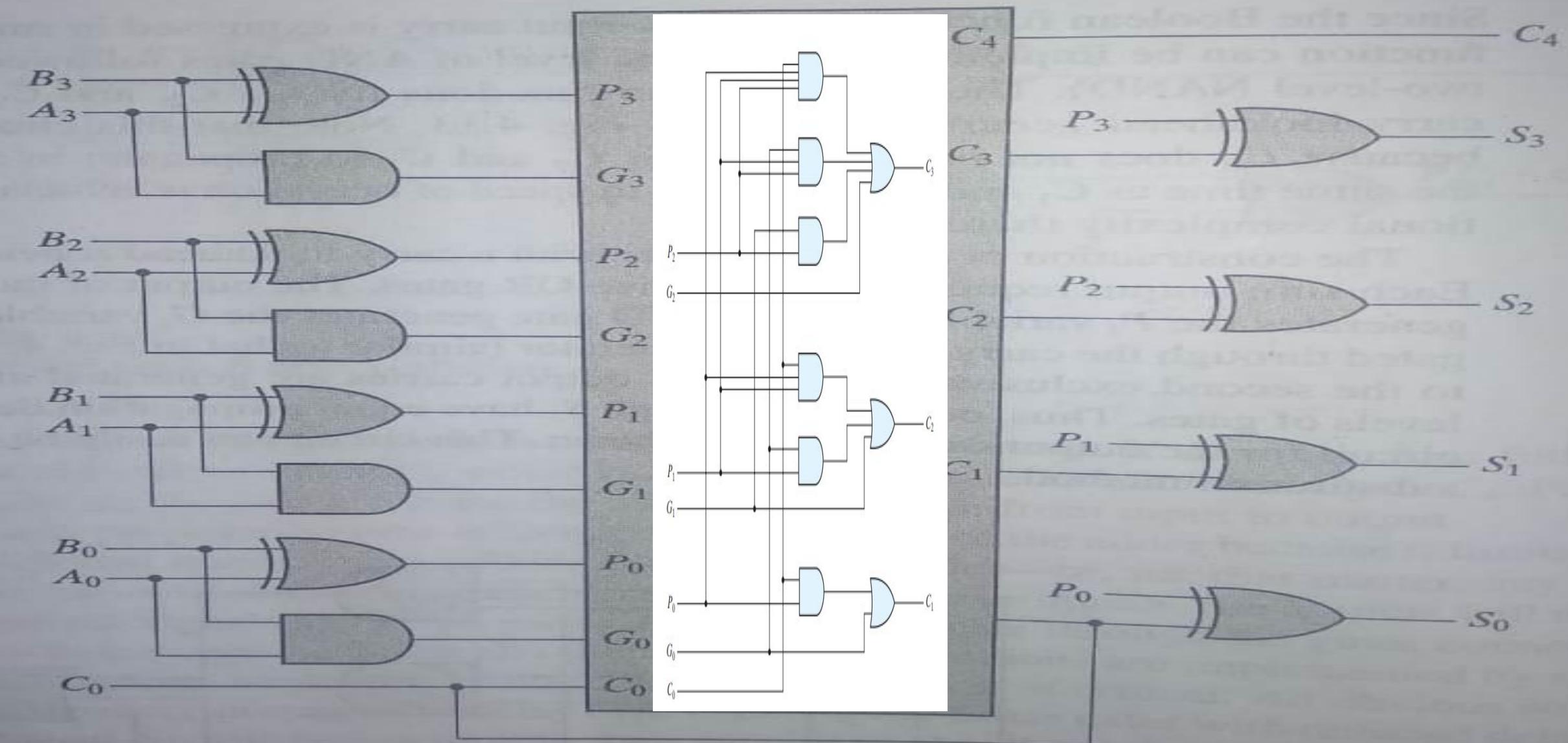
- $C_0 = 0$
- $C_1 = G_0 + P_0 C_0$
- $C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$
- $C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$
- $C_4 = G_3 + G_2 \cdot P_3 + G_1 \cdot P_2 \cdot P_3 + G_0 \cdot P_1 \cdot P_2 \cdot P_3 + C_0 \cdot P_0 \cdot P_1 \cdot P_2 \cdot P_3$



- All output carries are generated after a delay through two levels of gates. Thus, outputs S_1 through S_3 have equal propagation delay times.

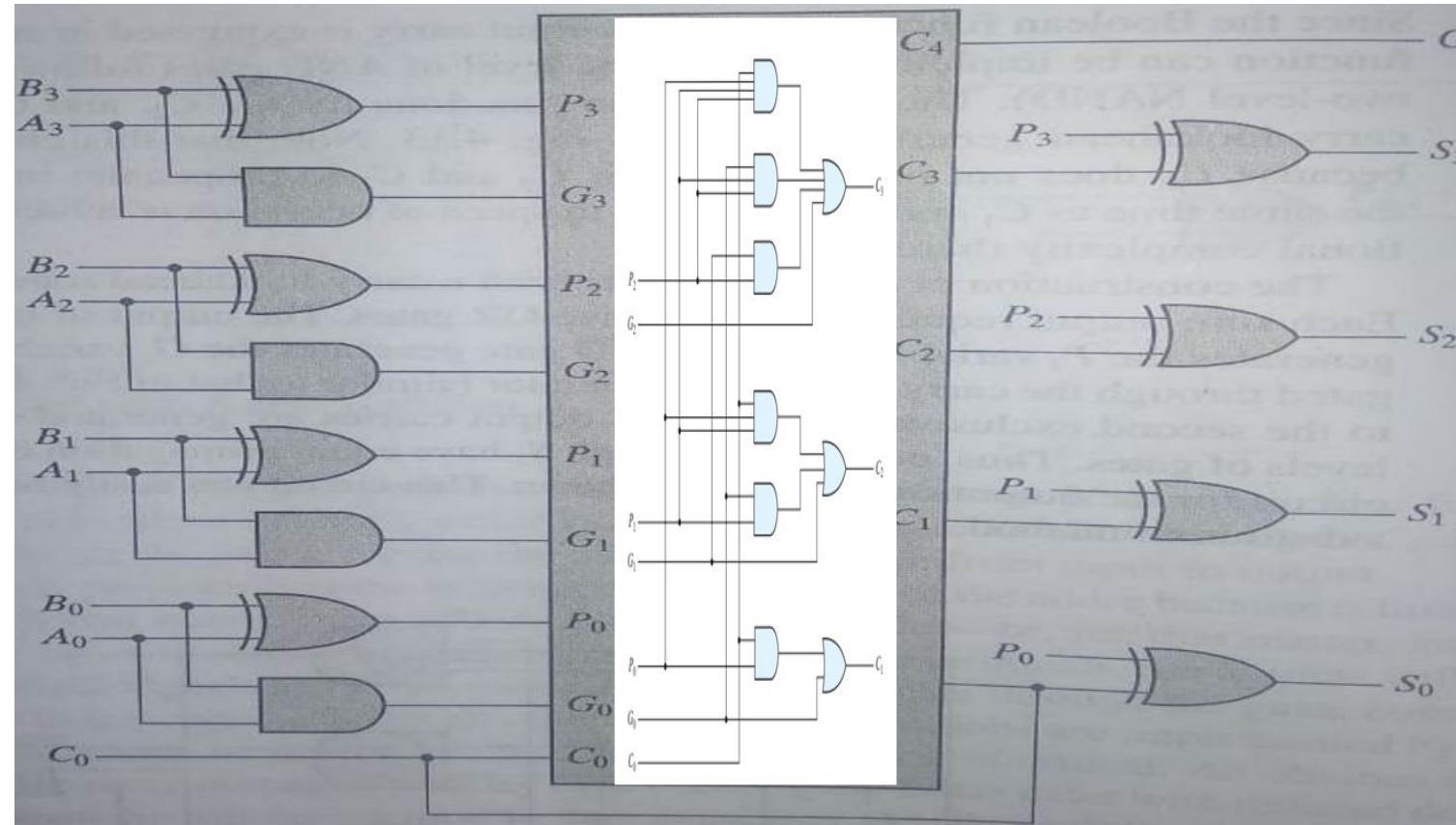


- All output carries are generated after a delay through two levels of gates. Thus, outputs S_1 through S_3 have equal propagation delay times.



Q Consider a carry lookahead adder for adding two n-bit integers, built using gates of fan-in at most two. The time to perform addition using this adder is **(GATE–2016) (2 Marks)**

- (A) $\Theta(1)$ (B) $\Theta(\log(n))$ (C) $\Theta(\sqrt{n})$ (D) $\Theta(n)$



Q In a look-ahead carry generator, the carry generates function G_i and the carry propagate function P_i for inputs A_i and B_i are given by (GATE-2007) (2 Marks)

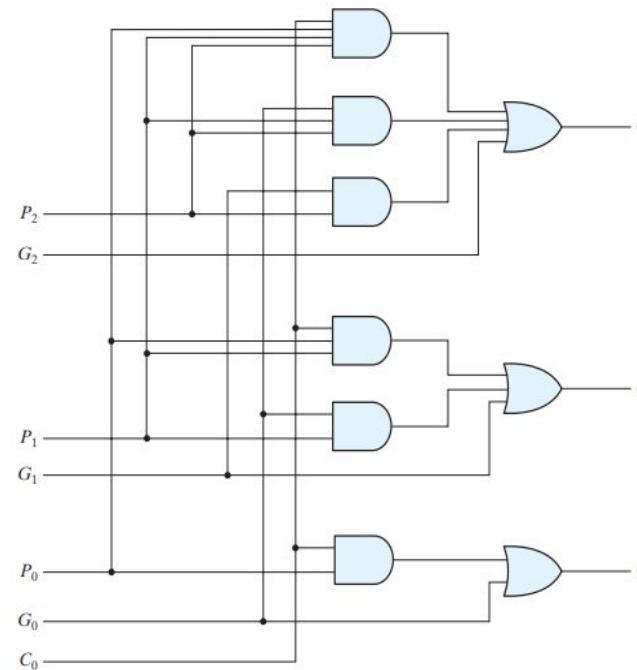
$$P_i = A_i \oplus B_i \text{ and } G_i = A_i B_i$$

The expressions for the sum bit S_i and the carry bit C_{i+1} of the look-ahead carry adder are given by:

$$S_i = P_i \oplus C_i \text{ and } C_{i+1} = G_i + P_i C_i, \text{ where } C_0 \text{ is the input carry.}$$

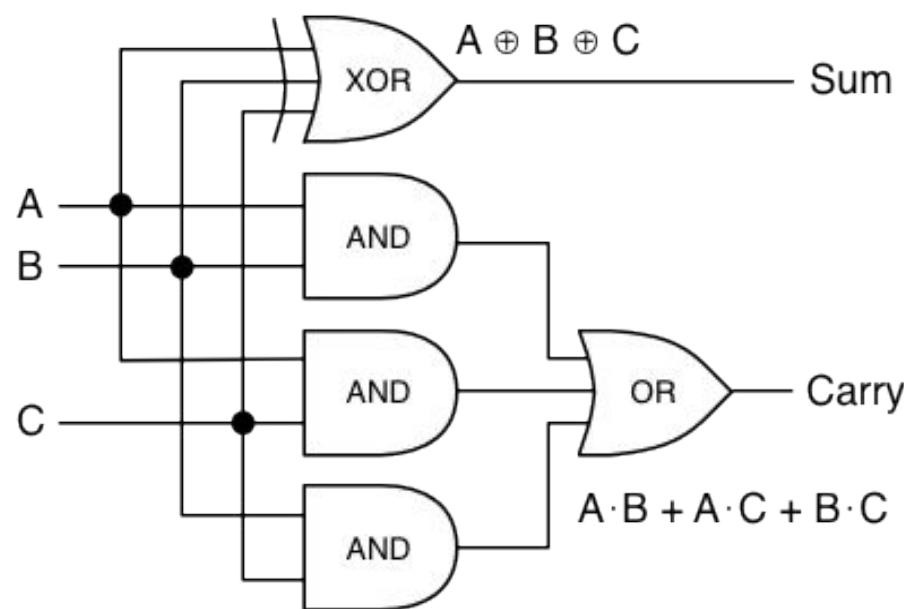
Consider a two-level logic implementation of the look-ahead carry generator. Assume that all P_i and G_i are available for the carry generator circuit and that the AND and OR gates can have any number of inputs. The number of AND gates and OR gates needed to implement the look-ahead carry generator for a 4-bit adder with S_3, S_2, S_1, S_0 and C_4 as its outputs are respectively:

- (A) 6, 3 (B) 10, 4 (C) 6, 4 (D) 10, 5



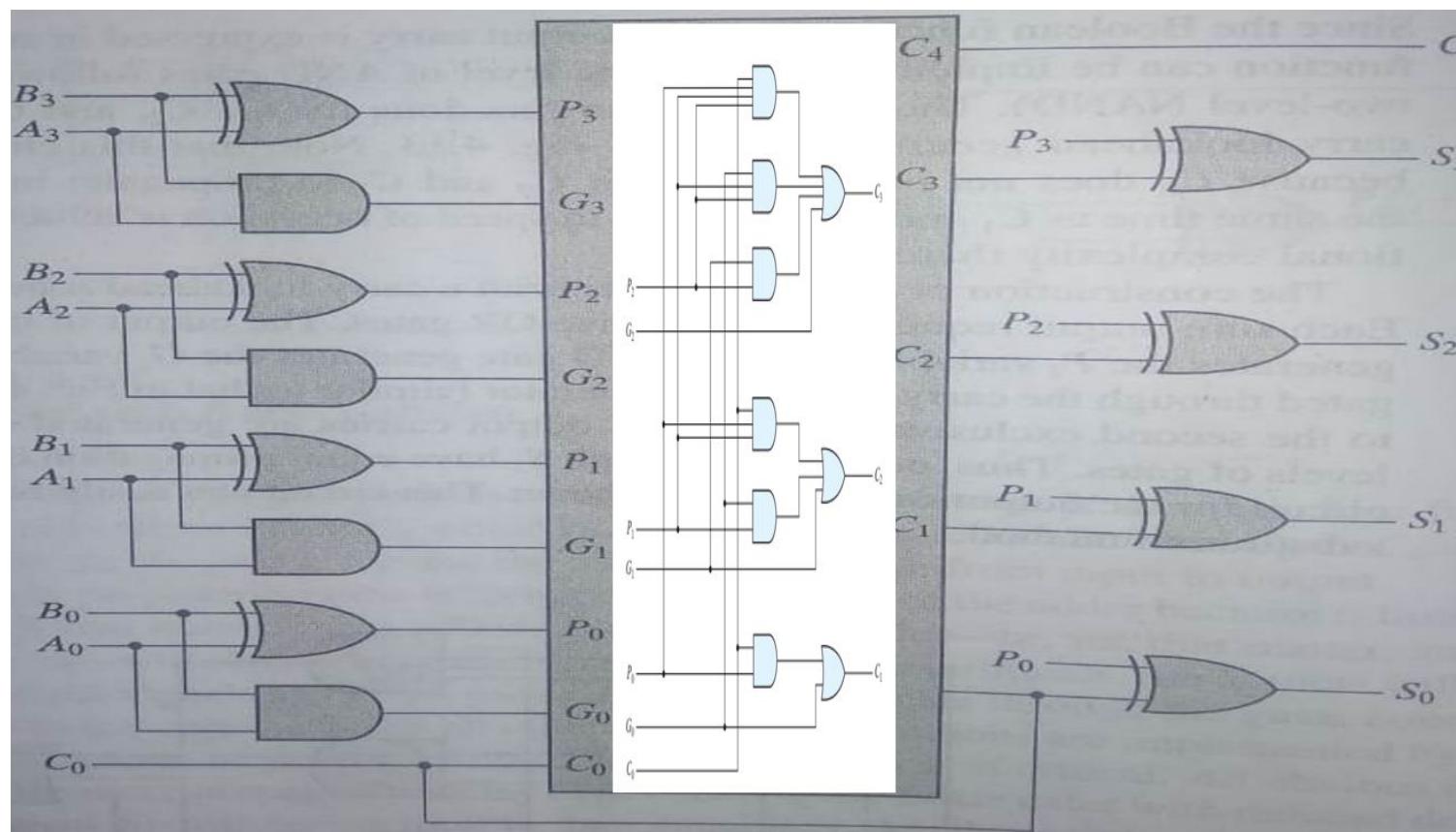
Q Given two three-bit numbers $a_2a_1a_0$ and $b_2b_1b_0$ and c, the carry in, the function that represents the carry generate function when these two numbers are added is **(GATE-2006) (2 Marks)**

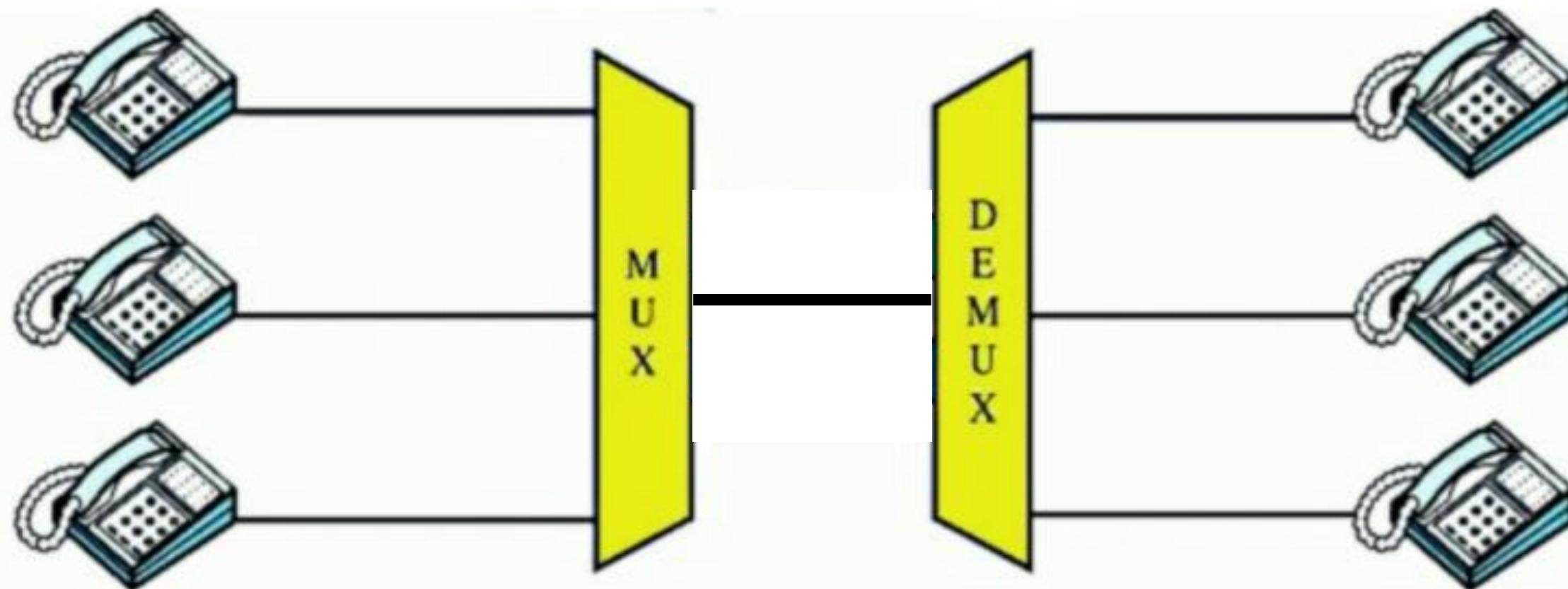
- (A) $a_2b_2 + a_2a_1b_1 + a_2a_1a_0b_0 + a_2a_0b_1b_0 + a_1b_2b_1 + a_1a_0b_2b_0 + a_0b_2b_1b_0$
- (B) $a_2b_2 + a_2b_1b_0 + a_2a_1b_1b_0 + a_1a_0b_2b_1 + a_1a_0b_1 + a_1a_0b_2b_0 + a_2a_0b_1b_0$
- (C) $a_2 + b_2 + (a_2 \oplus b_2)(a_1 + b_1 + (a_1 \oplus b_1)(a_0 + b_0))$
- (D) $a_2b_2 + \overline{a_2}a_1b_1 + \overline{a_2}\overline{a_1}a_0b_0 + \overline{a_2}a_0\overline{b_1}b_0 + a_1\overline{b_2}b_1 + \overline{a_1}a_0\overline{b_2}b_0 + a_0\overline{b_2}b_1b_0$



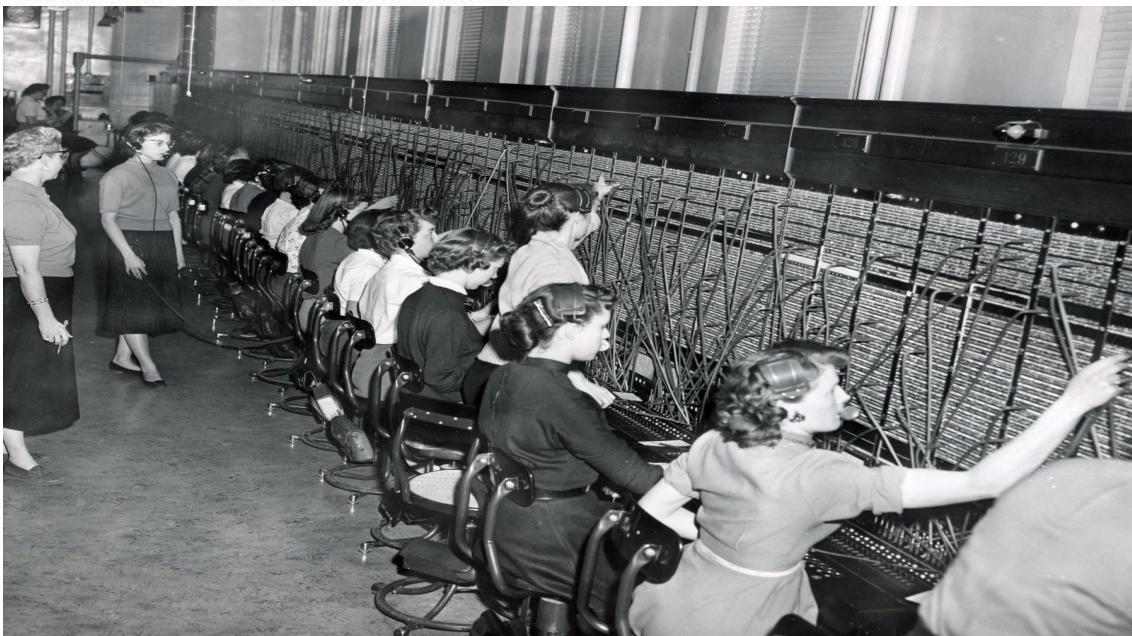
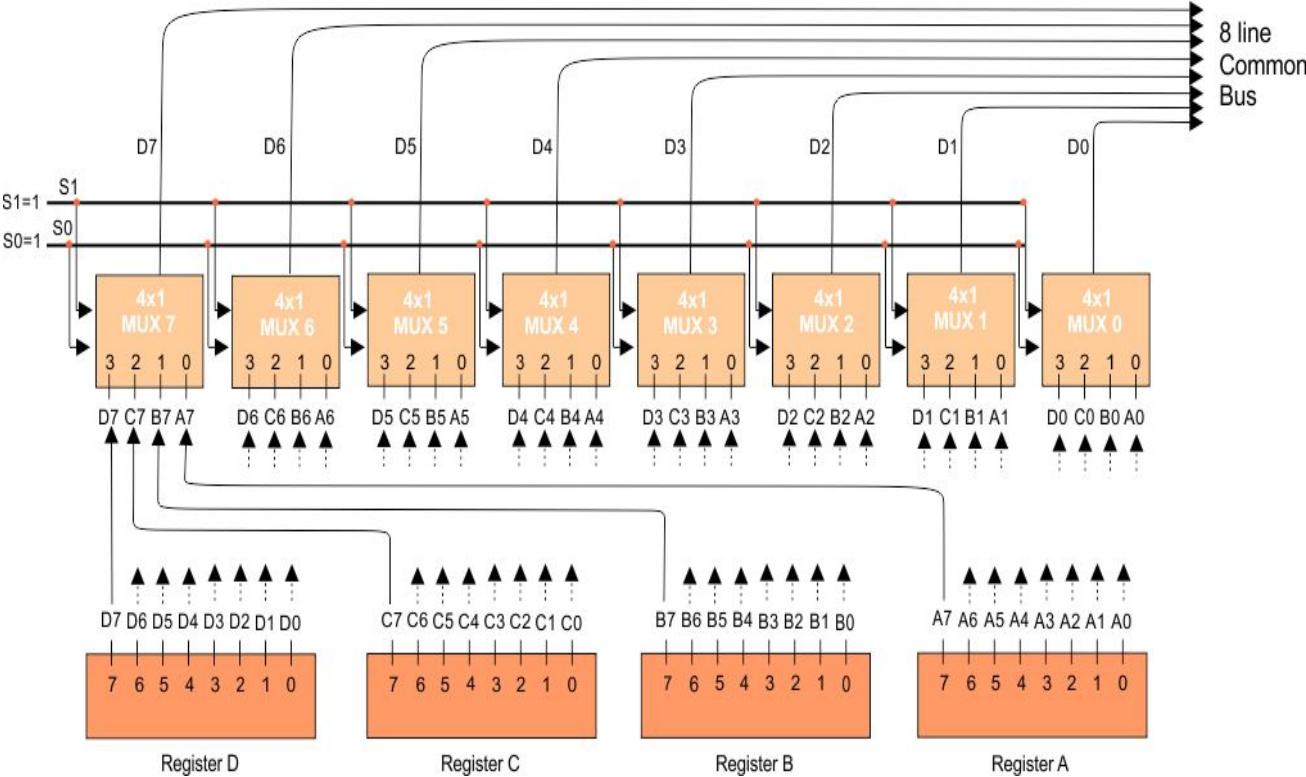
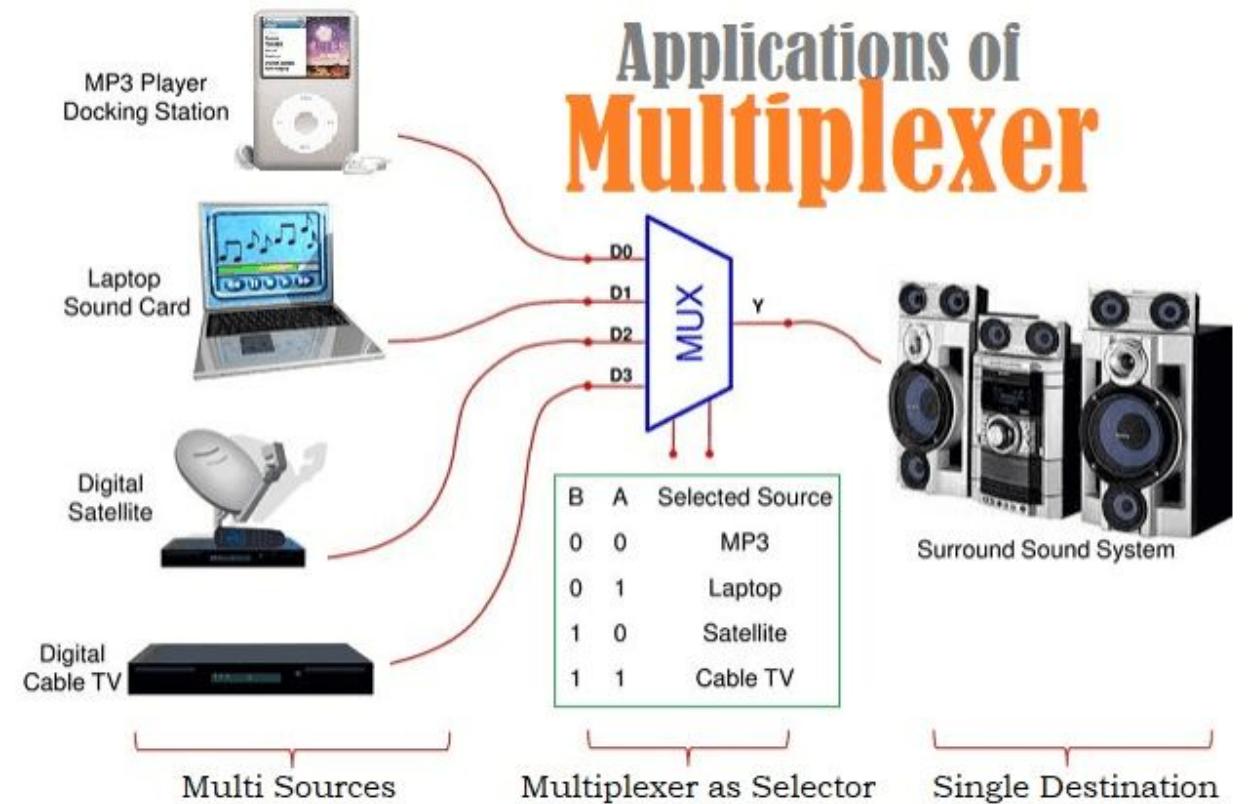
Q A 4-bit carry lookahead adder, which adds two 4-bit numbers, is designed using AND, OR, NOT, NAND, NOR gates only. Assuming that all the inputs are available in both complemented and uncomplemented forms and the delay of each gate is one-time unit, what is the overall propagation delay of the adder? Assume that the carry network has been implemented using two-level AND-OR logic. **(GATE-2004) (2 Marks)**

- (A) 4 time units (B) 6 time units (C) 10 time units (D) 12 time units**



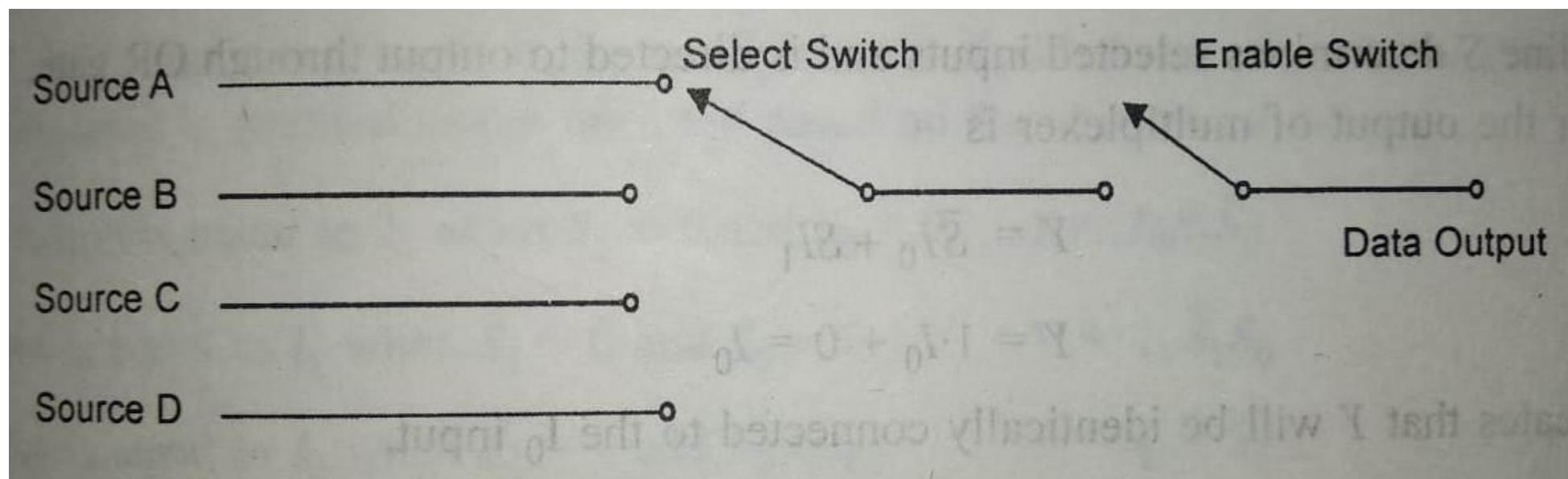


Applications of Multiplexer



Multiplexer (Selector)

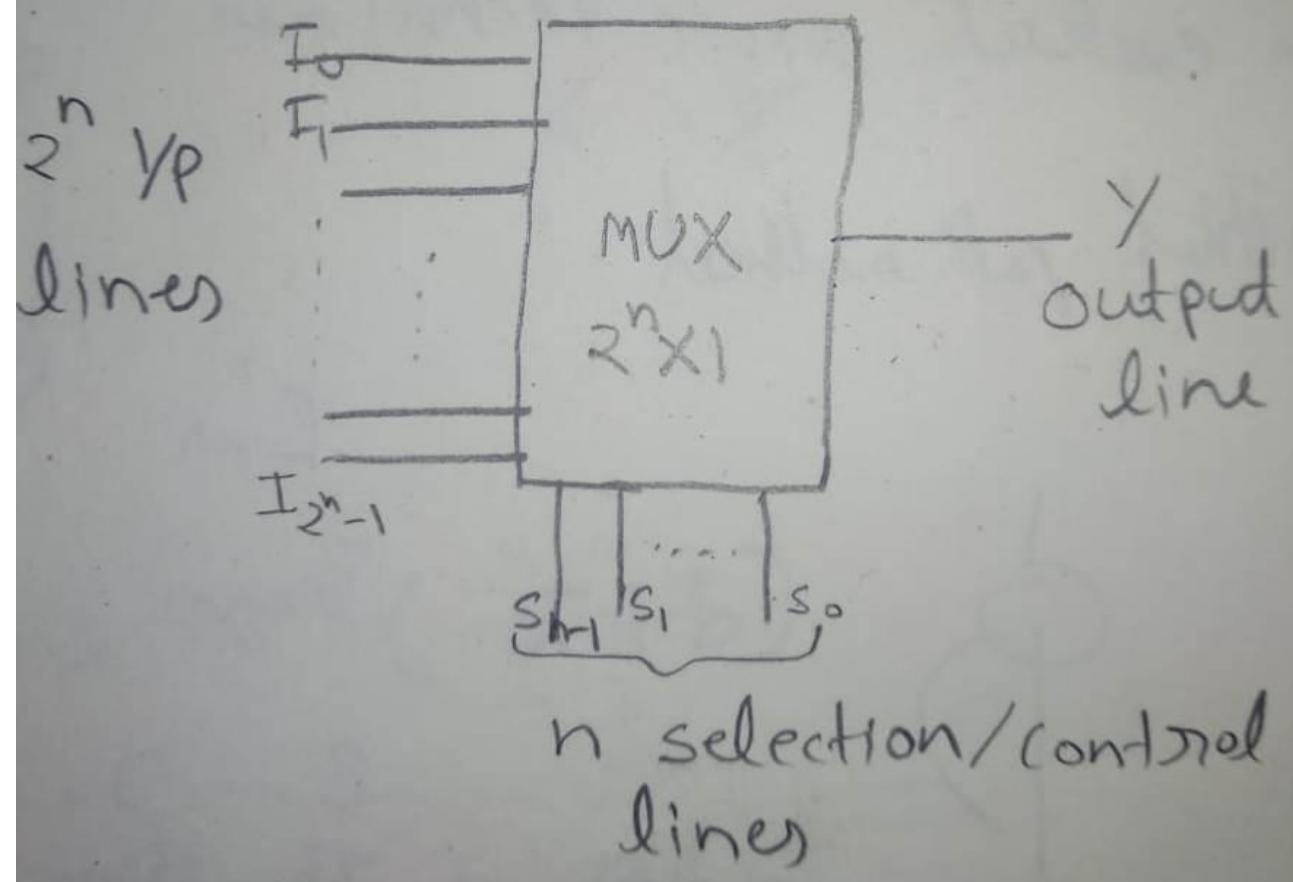
- Multiplexers are special and one of the most widely used combinational circuits.
- Main requirement is out of many inputs we have to select one for e.g. telephone or train leaving the station. So multiplexers do not perform any logical operation or comparison, it just acts as a switch or relay.
- A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. A multiplexer is also called a data selector, since it selects one of many inputs and steers the binary information to the output line.



- The selection of a particular input line is controlled by a set of selection lines.
- There are 2^n input lines and n selection lines whose bit combinations determine which input is to be selected.
- No of Input line $\leq 2^n$
- Note:**
- Can never have two i/p connected to out at any time

Application

- Parallel data to serial data conversion
- Used as data selector, as the output of a multiplexer is directed from one of various inputs
- Used in implementation of Boolean functions
- Used in communication systems, **Computer Memory**, **Telephone Network**, **Transmission from the Computer System of a Satellite**

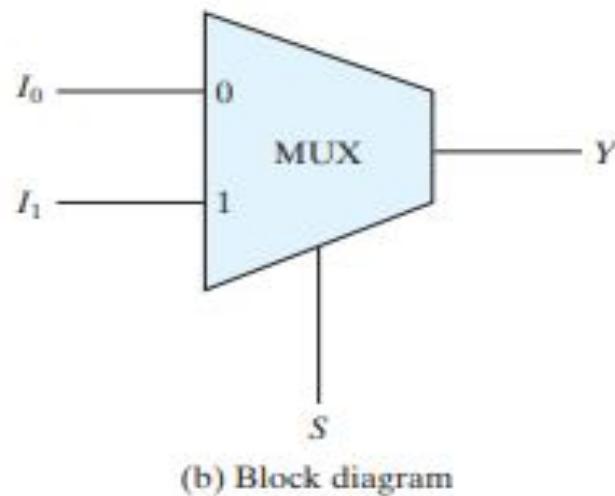


A two-to-one-line multiplexer

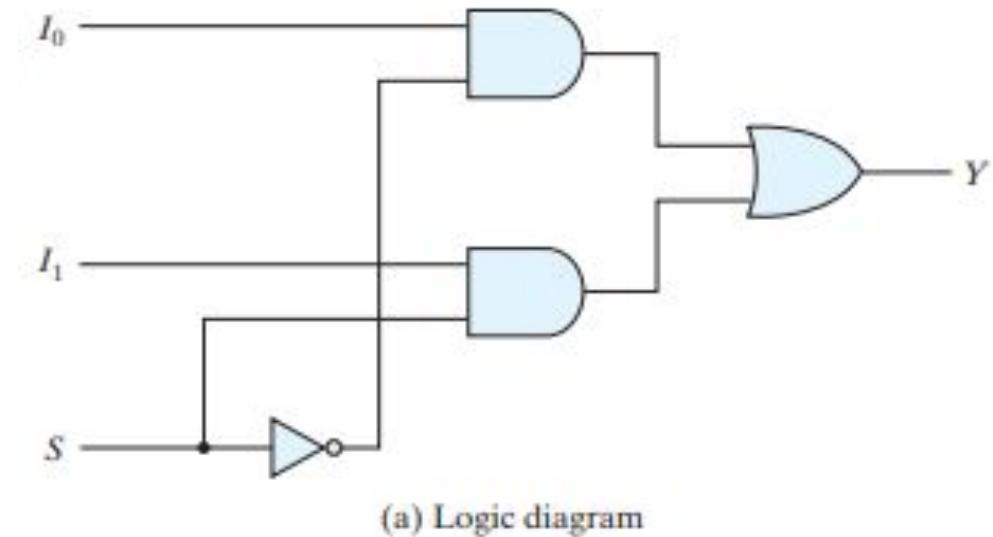
- This multiplexer acts like an electronic switch that selects one of two sources.
- The circuit has two data input lines, one output line, and one selection line S.

A two-to-one-line multiplexer

- When $S = 0$, the upper AND gate is enabled and I_0 has a path to the output.
- When $S = 1$, the lower AND gate is enabled and I_1 has a path to the output.



Input	Output
S	Y
0	I_0
1	I_1



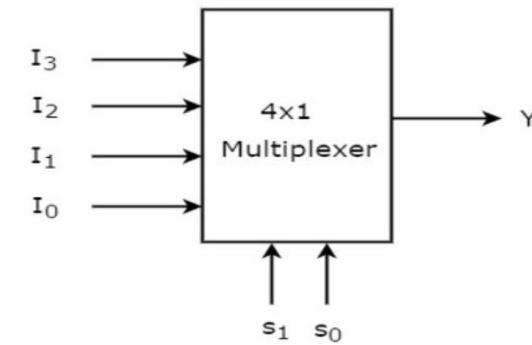
Characteristic equation: $Y = S_0' I_0 + S_0 I_1$

Case study of 4 to 1

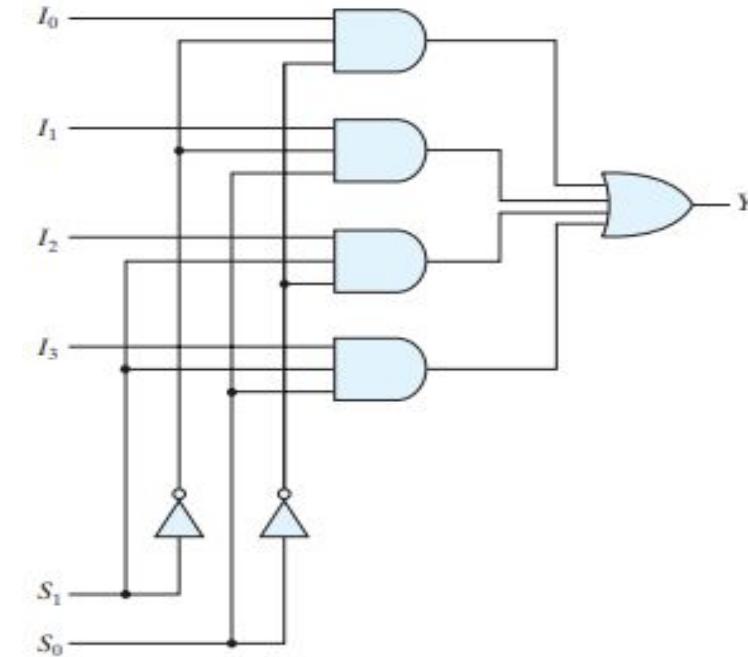
- The circuit has four data input lines I_0, I_1, I_2, I_3 , one output line Y , and two selection line S_0 and S_1 .

Case study of 4 to 1

- Each of the four inputs, I_0 through I_3 , is applied to one input of an AND gate.
- Selection lines S_1 and S_0 are decoded to select a particular AND gate.
- The outputs of the AND gates are applied to a single OR gate that provides the one-line output.



Input		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

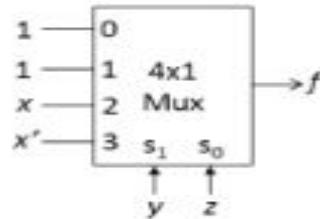


Characteristic equation

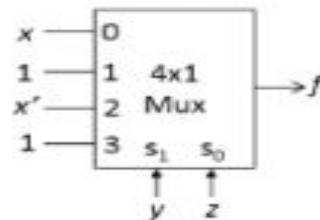
$$Y = \Sigma [(S'_1 S'_0 I_0) + (S'_1 S_0 I_1) + (S_1 S'_0 I_2) + (S_1 S_0 I_3)]$$

Q Which one of the following circuits implements the Boolean function given below? $f(x,y,z) = m_0 + m_1 + m_3 + m_4 + m_5 + m_6$. where m_i is the i th minterm. **(GATE 2021) (1 MARKS)**

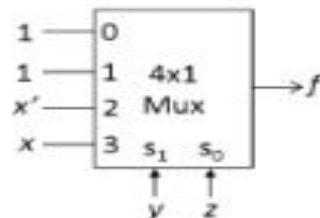
(A)



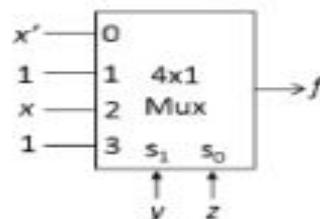
(B)



(C)



(D)



Q Consider a 4-to-1 multiplexer with two select lines S_1 and S_0 , given below

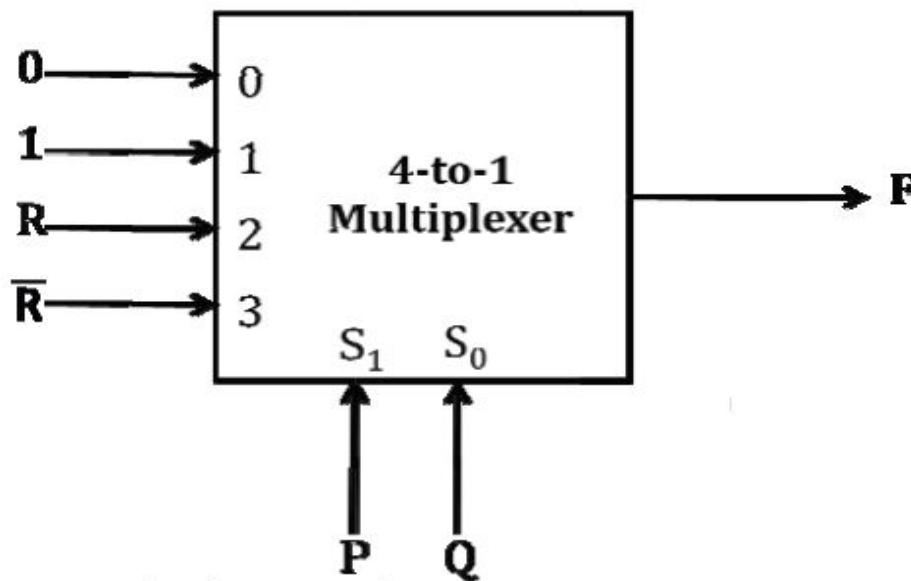
The minimal sum-of-products form of the Boolean expression for the output F of the multiplexer is **(Gate-CS-2014-Set-1) (2 Marks)**

(A) $P'Q + QR' + PQ'R$

(B) $P'Q + P'QR' + PQR' + PQ'R$

(C) $P'QR + P'QR' + QR' + PQ'R$

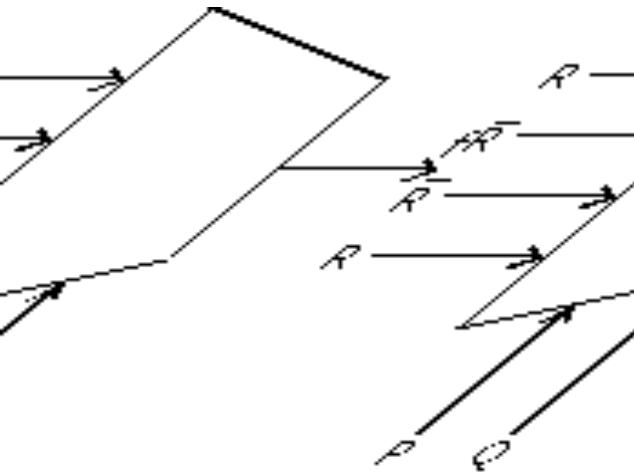
(D) PQR'



$$Y = [(S'_1 S'_0 |_0) + (S'_1 S_0 |_1) + (S_1 S'_0 |_2) + (S_1 S_0 |_3)]$$

Q The Boolean expression for the output 'f' of the multiplexer shown below is **(GATE-2010) (2 Marks)**

- (A) $(P \oplus Q \oplus R)'$** **(B) $P \oplus Q \oplus R$** **(C) $(P + Q + R)'$** **(D) $P + Q + R$**

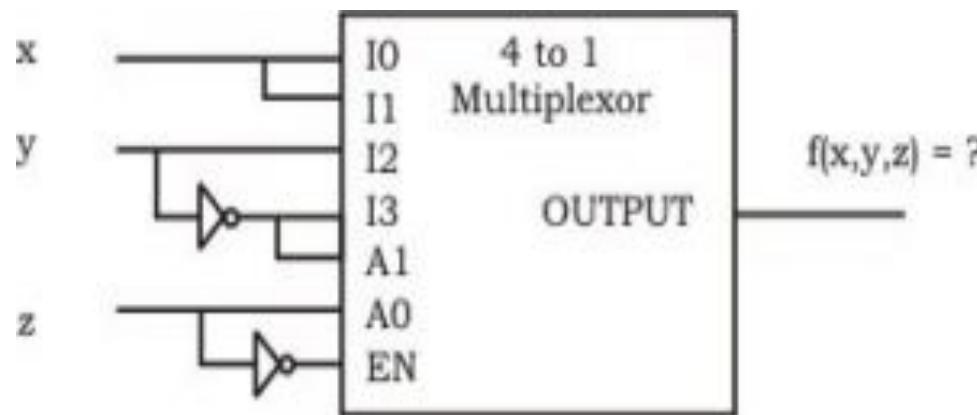


$$Y = [(S'_1 S'_0 I_0) + (S'_1 S_0 I_1) + (S_1 S'_0 I_2) + (S_1 S_0 I_3)]$$

Q Consider the following multiplexor where I_0, I_1, I_2, I_3 are four data input lines selected by two address line combinations $A_1 A_0 = 00, 01, 10, 11$ respectively and f is “the output of the multiplexor. EN is the enable input. (Gate-CS-2002) (2 Marks)

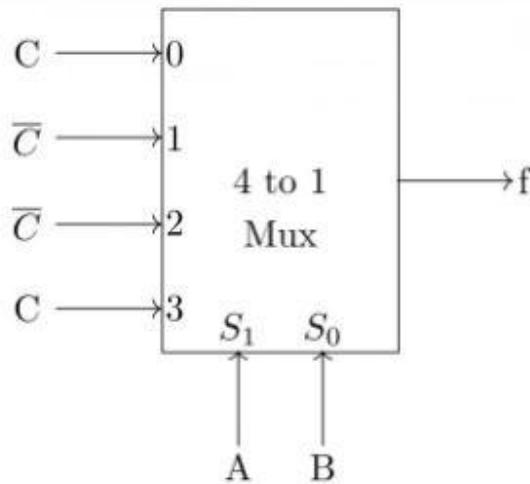
The function $f(x, y, z)$ implemented by the above circuit is :

- (A) xz' (B) $xy + z$ (C) $x + z$ (D) None of these



$$Y = E [(S'_1 S'_0 I_0) + (S'_1 S_0 I_1) + (S_1 S'_0 I_2) + (S_1 S_0 I_3)]$$

Q Consider the circuit in below figure. f implements (GATE-1996) (2 Marks)



(A) $(ABC)' + A'BC' + ABC$

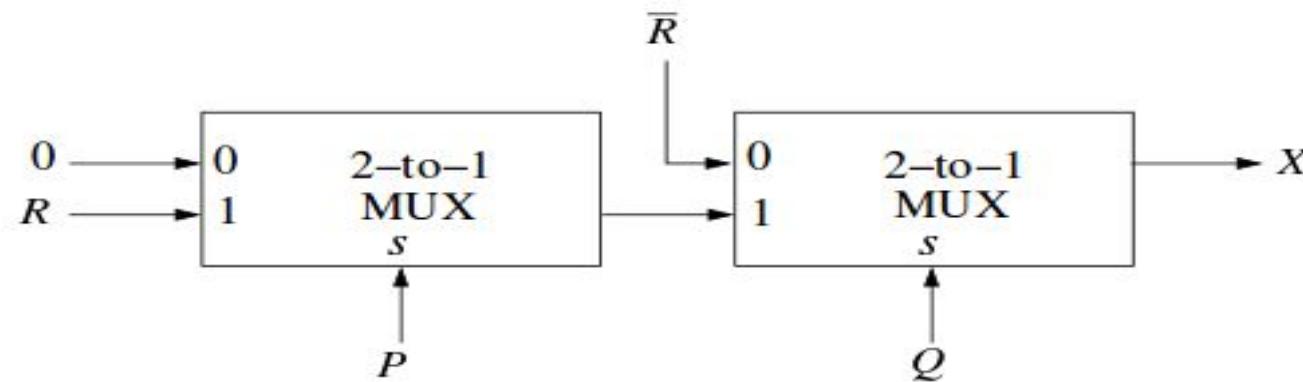
(B) $A + B + C$

(C) $A \oplus B \oplus C$

(D) $AB + BC + CA$

$$Y = [(S'_1 S'_0 I_0) + (S'_1 S_0 I_1) + (S_1 S'_0 I_2) + (S_1 S_0 I_3)]$$

Q Consider the two cascaded 2-to-1 multiplexers as shown in the figure. (GATE-2016) (2 Marks)



$$Y = S_0' I_0 + S_0 I_1$$

The minimal sum of products form of the output X is

- (A) $\bar{P}\bar{Q} + PQR$
- (B) $\bar{P}Q + QR$
- (C) $PQ + \bar{P}\bar{Q}R$
- (D) $\bar{Q}\bar{R} + PQR$

$$Y = S_0' I_0 + S_0 I_1$$

Q The following circuit implements a two-input AND gate using two 2-1 multiplexers.

(GATE-2007) (1 Marks)

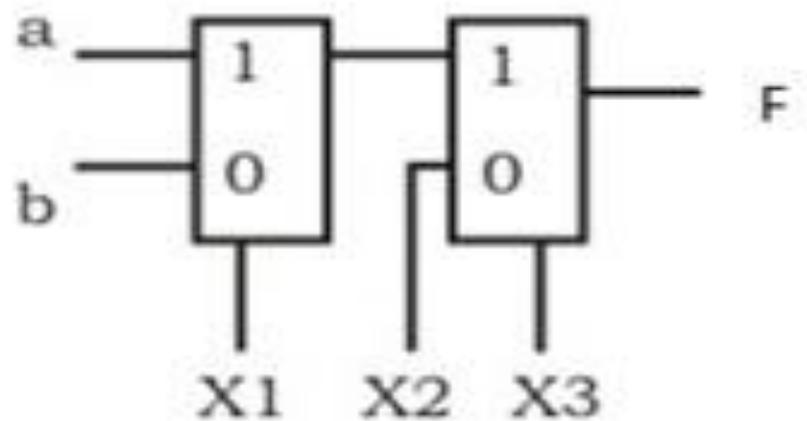
What are the values of X_1, X_2, X_3 ?

(A) $X_1=b, X_2=0, X_3=a$

(C) $X_1=a, X_2=b, X_3=1$

(B) $X_1=b, X_2=1, X_3=b$

(D) $X_1=a, X_2=0, X_3=b$

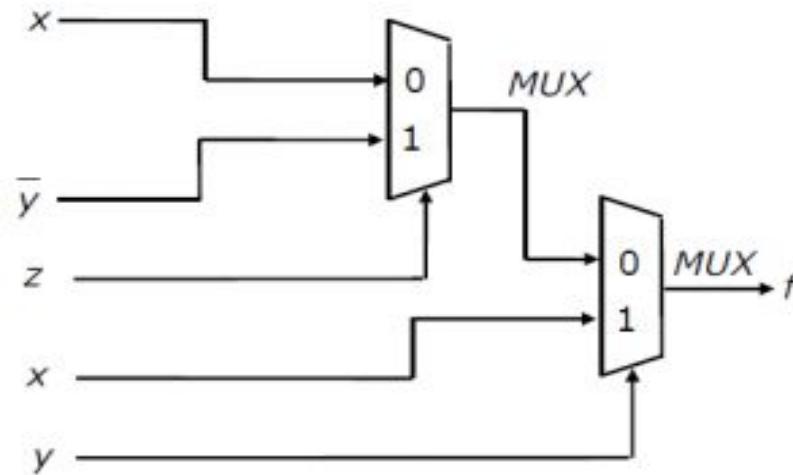


$$Y = S_0' I_0 + S_0 I_1$$

$$Y = S_0' I_0 + S_0 I_1$$

Q Consider the circuit above. Which one of the following options correctly represents $f(x, y, z)$? (Gate-CS-2006) (2 Marks)

- (A) $xz' + xy + y'z$ (B) $xz' + xy + (yz)'$ (C) $xz + xy + (yz)'$ (D) $xz + xy' + y'z$

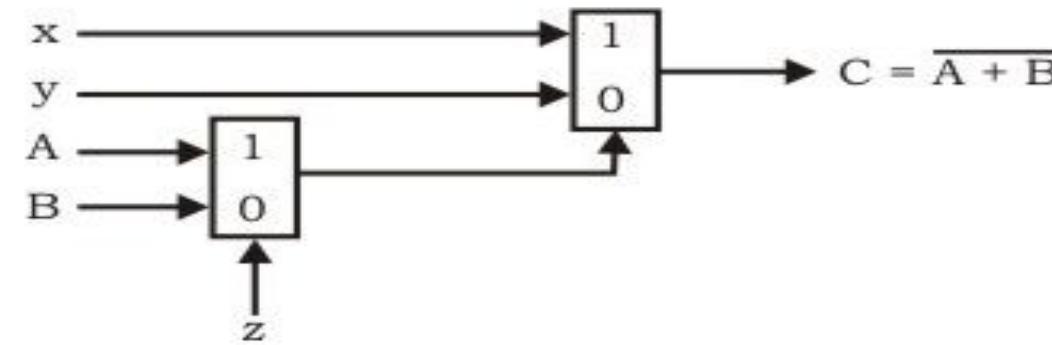


$$Y = S_0' I_0 + S_0 I_1$$

$$Y = S_0' I_0 + S_0 I_1$$

Q The circuit shown below implements a 2-input NOR gate using two 2-4 MUX (control signal 1 selects the upper input). What are the values of signals x, y and z? (GATE-2005) (2 Marks)

- (A) 1, 0, B (B) 1, 0, A (C) 0, 1, B (D) 0, 1, A

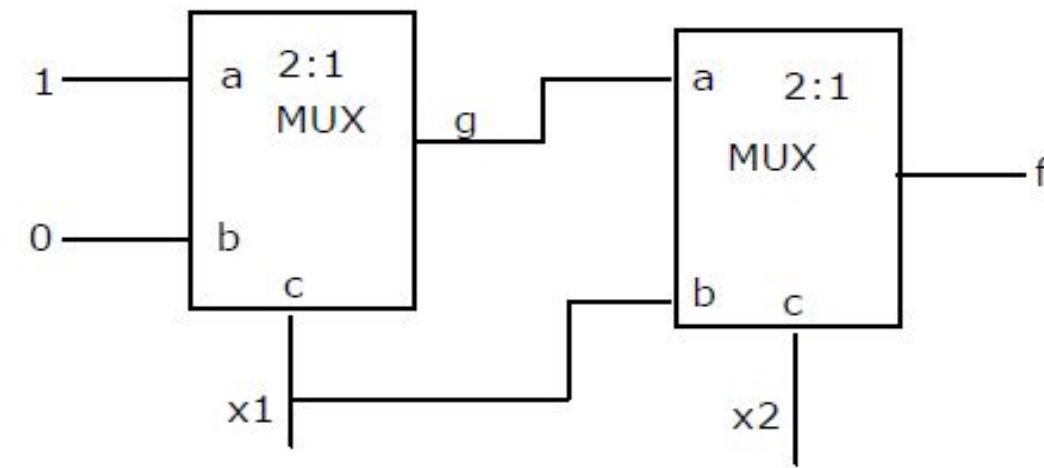


$$Y = S_0' I_0 + S_0 I_1$$

$$Y = S_0' I_0 + S_0 I_1$$

Q Consider the circuit shown below. The output of a 2:1 Mux is given by the function $(ac' + bc)$, Which of the following is true? (GATE-2001) (2 Marks)

- (A) $f = x_1' + x_2$ (B) $f = x_1'x_2 + x_1x_2'$ (C) $f = x_1x_2 + x_1'x_2'$ (D) $f = x_1 + x_2'$

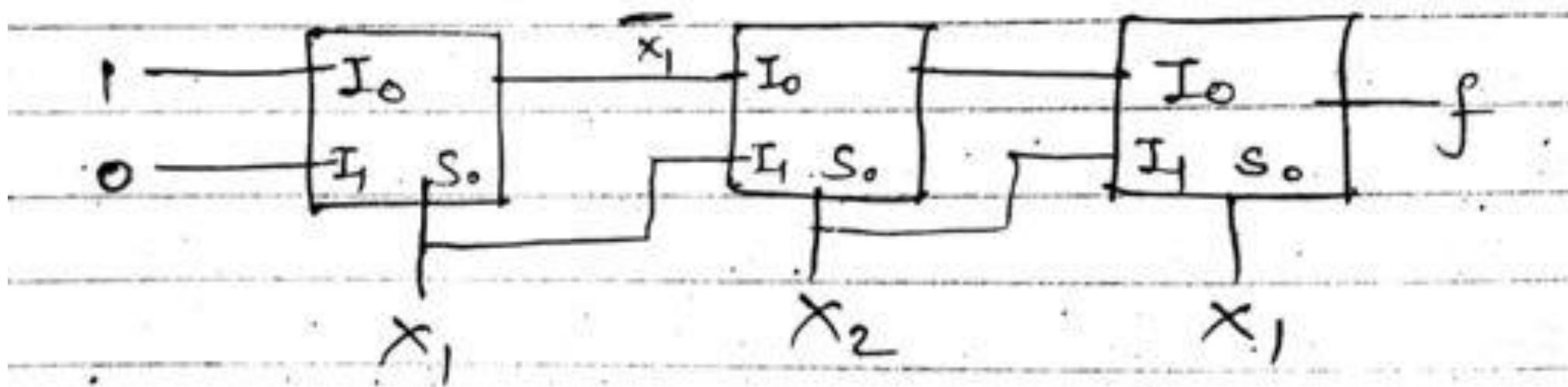


$$Y = S_0' I_0 + S_0 I_1$$

$$Y = S_0' I_0 + S_0 I_1$$

Q Consider the circuit shown below. Which of the following is true?

- (A) $f = x_1' + x_2$ (B) $f = x_1'x_2 + x_1x_2'$ (C) $f = x_1x_2 + x_1'x_2'$ (D) none



$$Y = S_0' I_0 + S_0 I_1$$

$$Y = S_0' I_0 + S_0 I_1$$

$$Y = S_0' I_0 + S_0 I_1$$

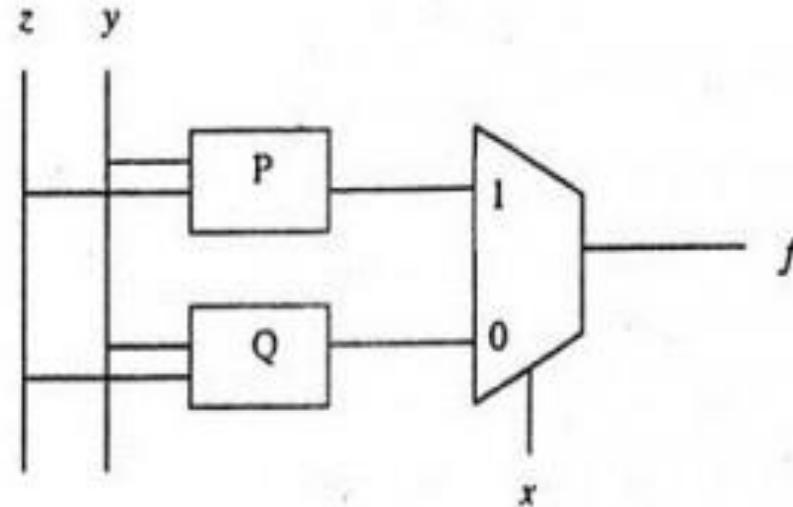
Q Suppose only one multiplexer and one inverter are allowed to be used to implement any Boolean function of n variables. What is the minimum size of the multiplexer needed?

(Gate-CS-2007) (2 Marks)

- (A) 2^n line to 1-line
- (B) 2^{n+1} line to 1 line
- (C) 2^{n-1} line to 1-line
- (D) 2^{n-2} line to 1 line

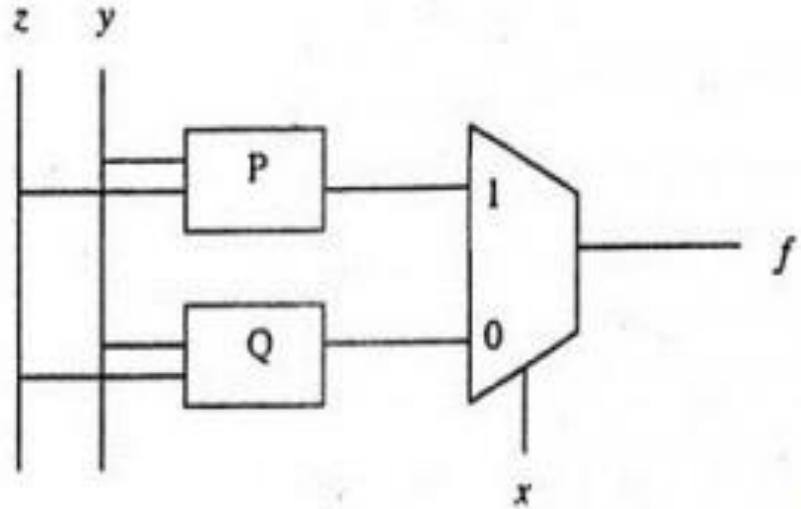
Q The majority function is a Boolean function $f(x, y, z)$ that takes the value 1 whenever a majority of the variables x, y, z is 1. In the circuit diagram for the majority function shown below, the logic gates for the boxes labelled P and Q are, respectively? **(GATE-2006) (2 Marks)**

- (A) XOR, AND
- (B) XOR, XOR
- (C) OR, OR
- (D) OR, AND



Q The majority function is a Boolean function $f(x, y, z)$ that takes the value 1 whenever a majority of the variables x, y, z is 1. In the circuit diagram for the majority function shown below, the logic gates for the boxes labelled P and Q are, respectively? **(GATE-2006) (2 Marks)**

- (A) XOR, AND
- (B) XOR, XOR
- (C) OR, OR
- (D) OR, AND



X	Y	Z	f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Multiplexer Expansion

(Implementation of Higher order MUX using lower order MUX)

Q How many 2:1 MUX are required to Implement 4:1 MUX ?

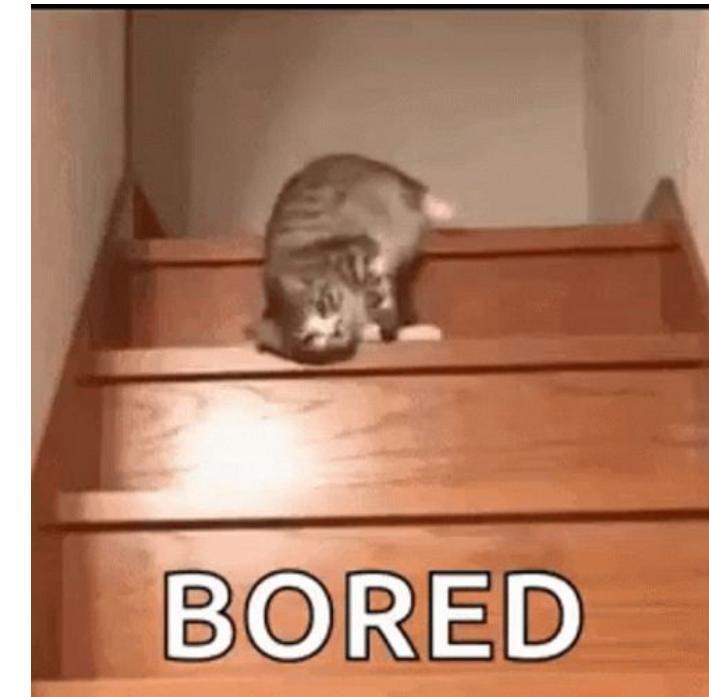
Input		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Q How many 4 : 1 MUX using 16 : 1 MUX ?

Q How many 4X1 Mux are required in order to construct 128X1?

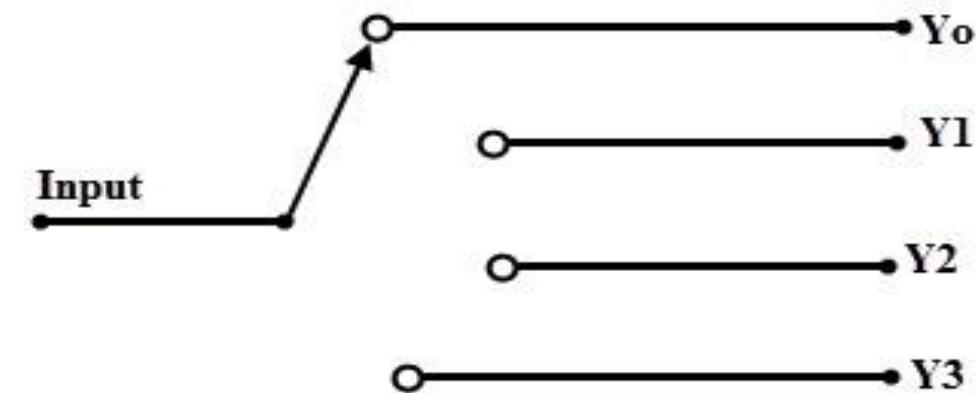
Q How many 8X1 Mux are required in order to construct 4096X1?

Given:	$m \times 1$
Target:	$n \times 1$
No of levels(K)	$\log_m n$
No of Mux at i^{th} level (x_i)	(n/m_i)
Total Mux required	
Maximum capacity	$m^k \times 1$

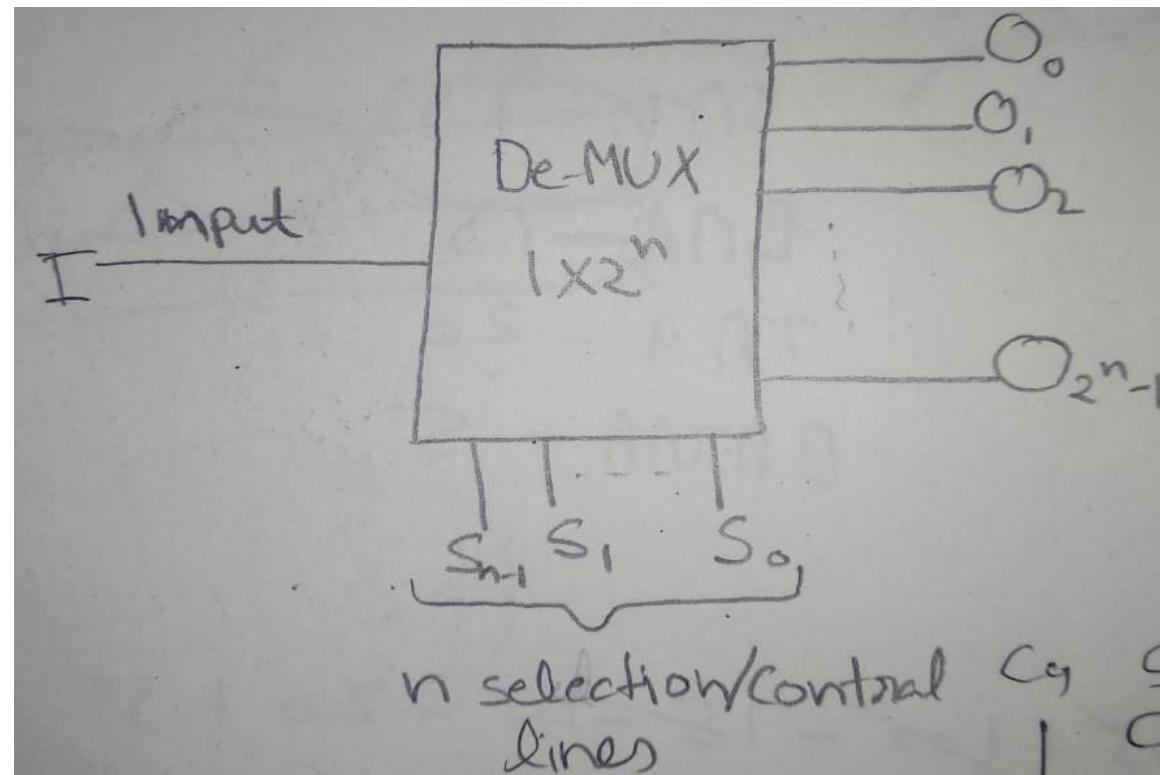


Demultiplexer

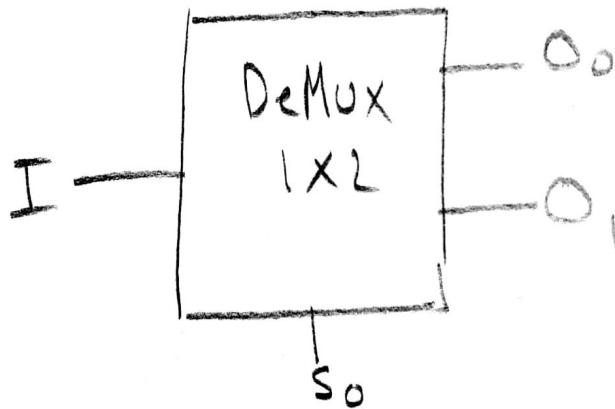
1. A demultiplexer (or DeMux) is a device that takes a single input line and routes it to one of several digital output lines.
2. A demultiplexer is also called a data distributor.
3. It is conceptually same as Mux just with reverse logic.



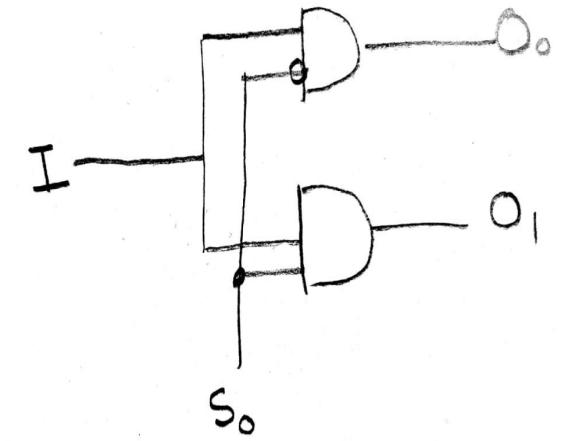
- A demultiplexer of 2^n outputs has n select lines, which are used to select which output line to send the input.
- A DeMux is a combinational circuit, which is used in data communication
- Serial to parallel conversation.
- Different input/output configuration demultiplexers are available in the form of single integrated circuits (ICs).
- Demultiplexers are mainly used in Boolean function generators and decoder circuits.



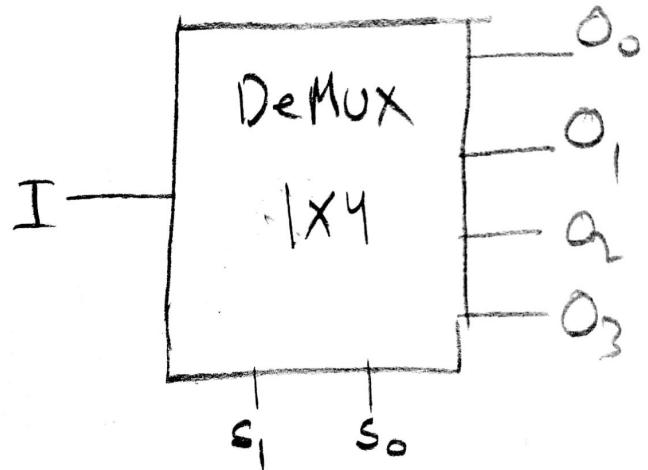
1 to 2 Demultiplexer



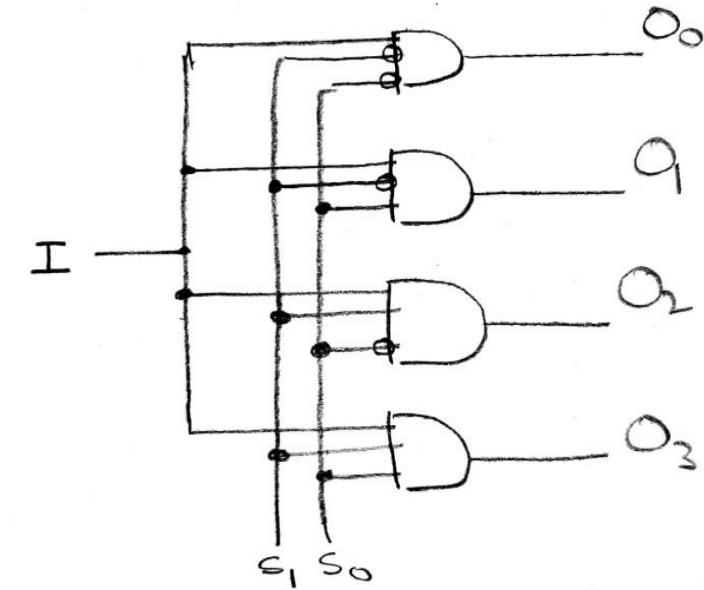
S ₀	O ₁	O ₀
0	0	1
1	1	0



1 to 4 Demultiplexer



s_1	s_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

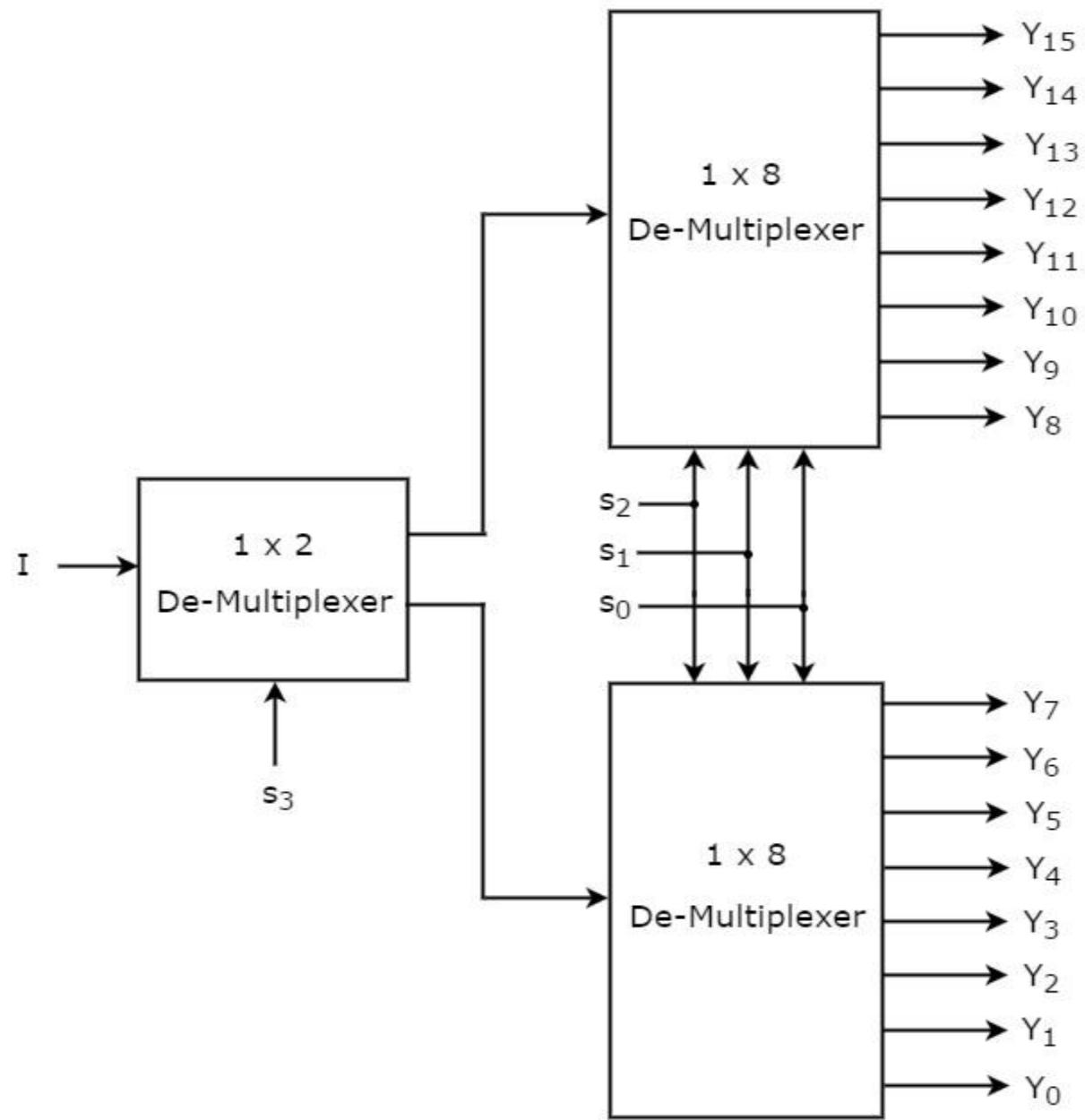


Demultiplexer Expansion

(Implementation of Higher order DeMux using lower order DeMux)

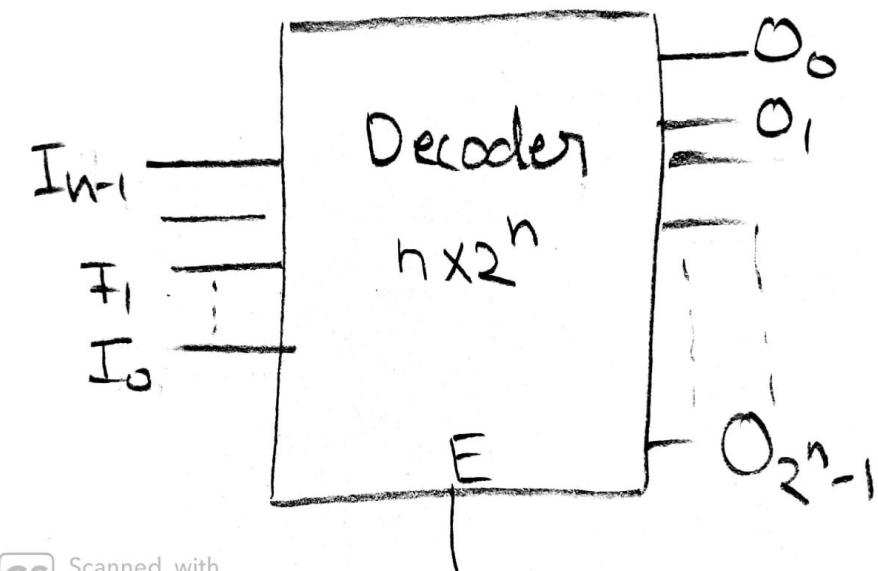
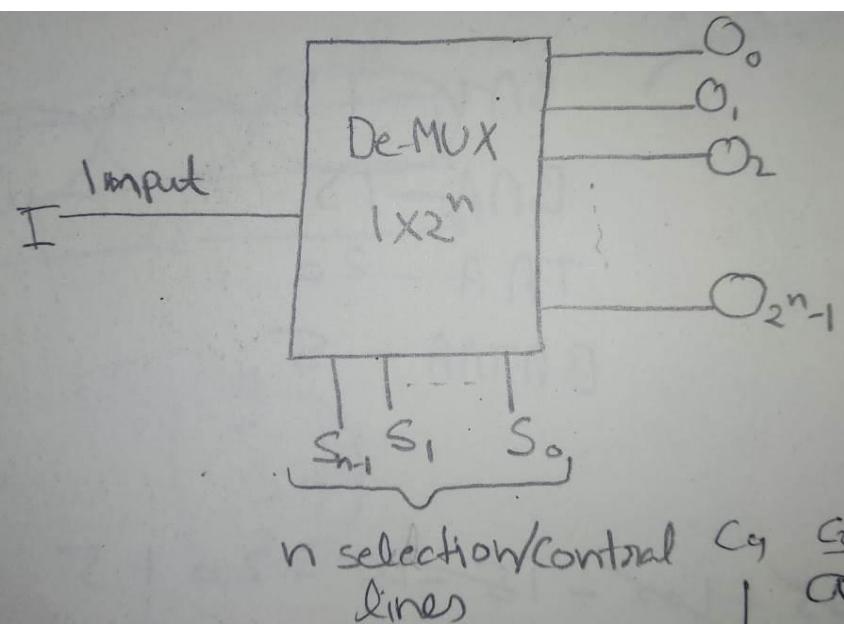
Q 1 : 2 DeMux are required to implement 1 : 4 DeMux ?

Q 1 : 2 DeMux are required to implement 1 : 8 DeMux ?

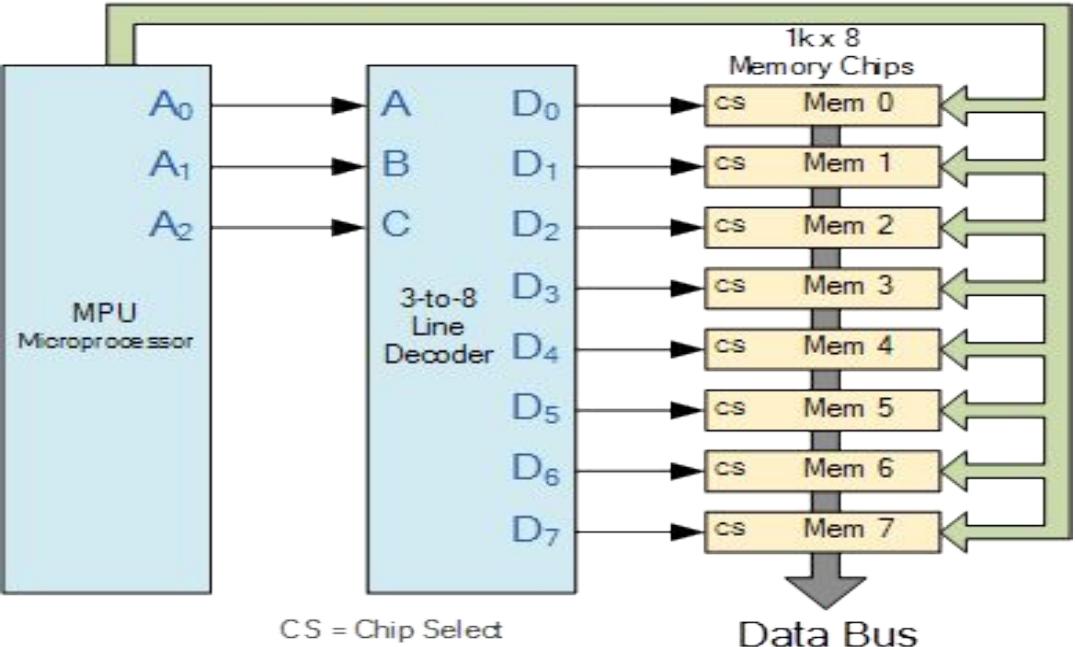


Decoder

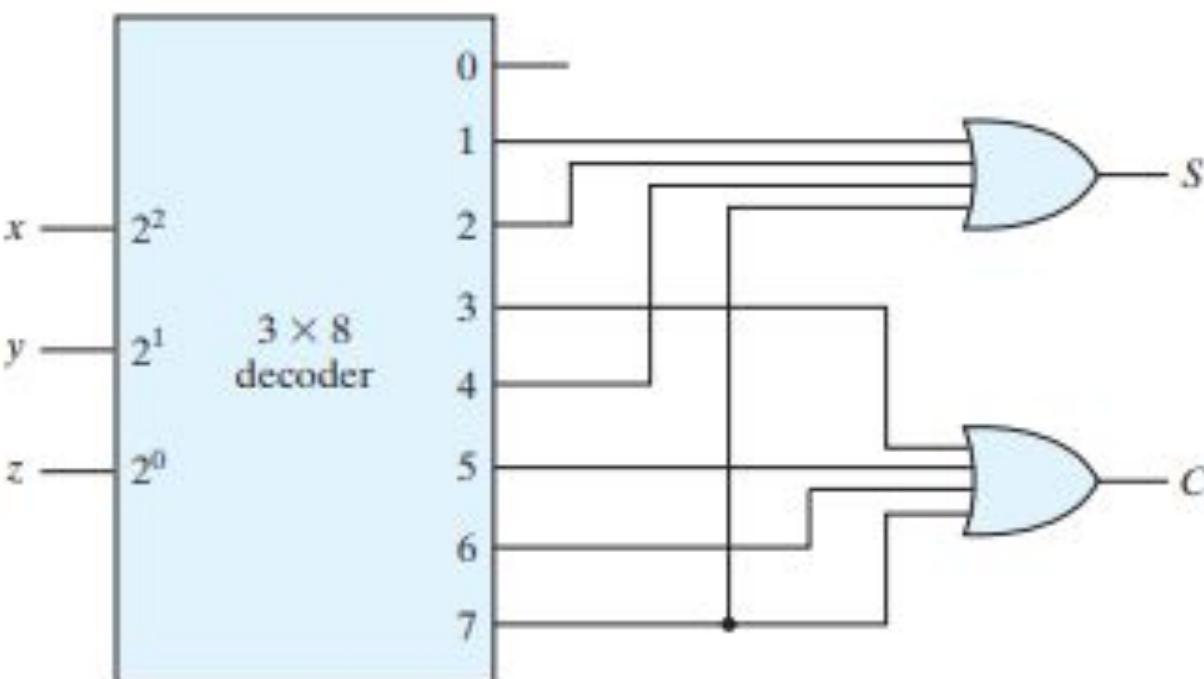
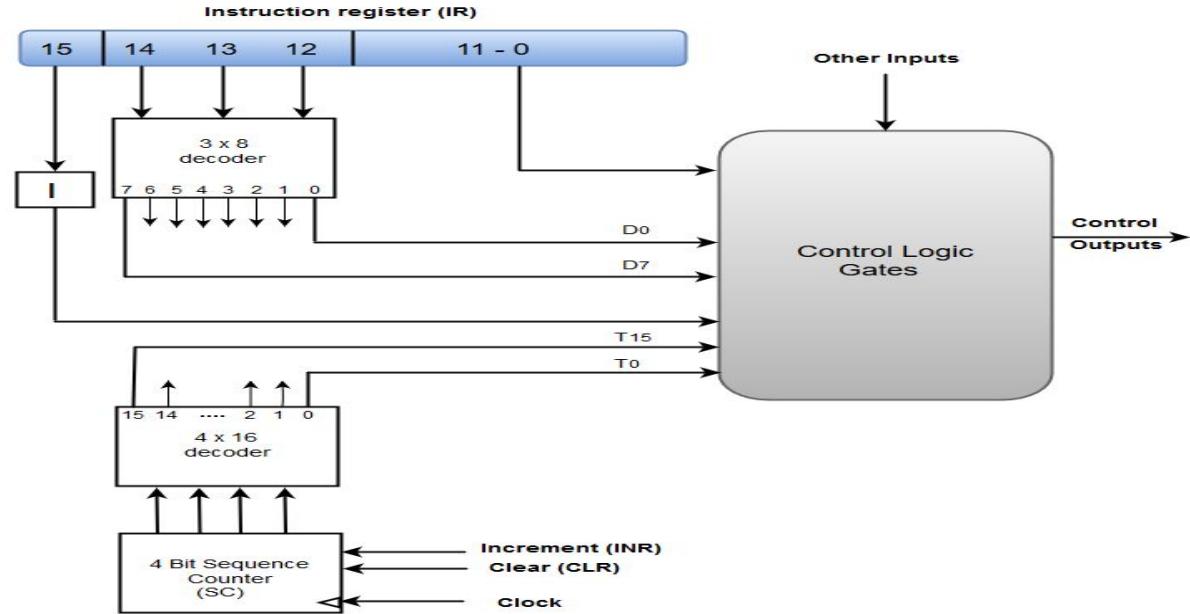
- A decoder is a combinational circuit that decodes binary information from n input lines to a maximum of 2^n unique output lines.
- The decoders are called n -to- m -line decoders, where $m \leq 2^n$.
- Their purpose is to generate the 2^n (or fewer) minterms of n input variables. Each combination of inputs will assert a unique output.
- If the n -bit coded information has unused combinations, the decoder may have fewer than 2^n outputs. Decoders are also combinational circuits logically we can say a DeMux can be converted into a decoder by setting input line as enable line and selection line as input lines.



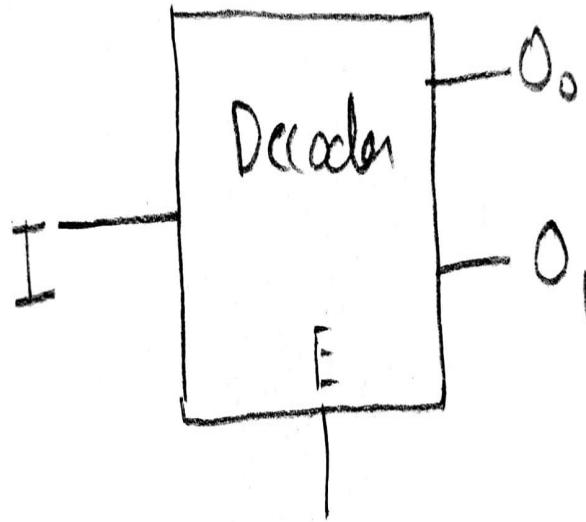
Address Bus - A_3 to A_{10}



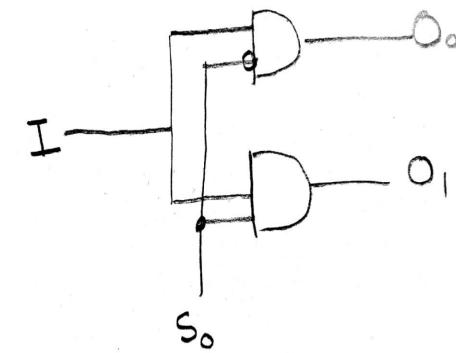
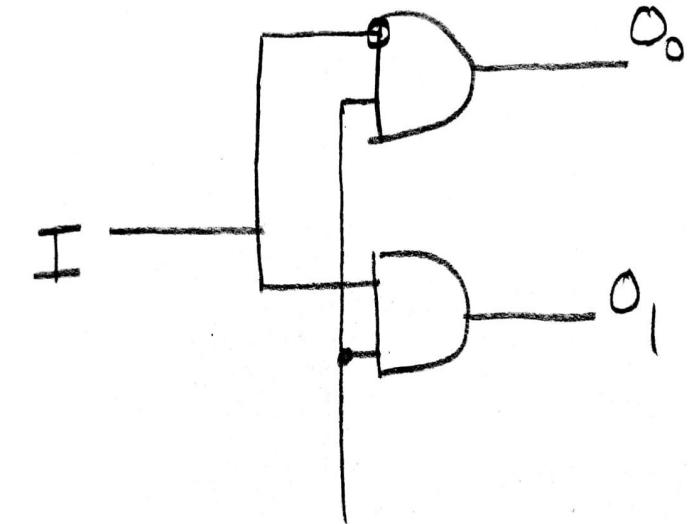
Control Unit of a Basic Computer:

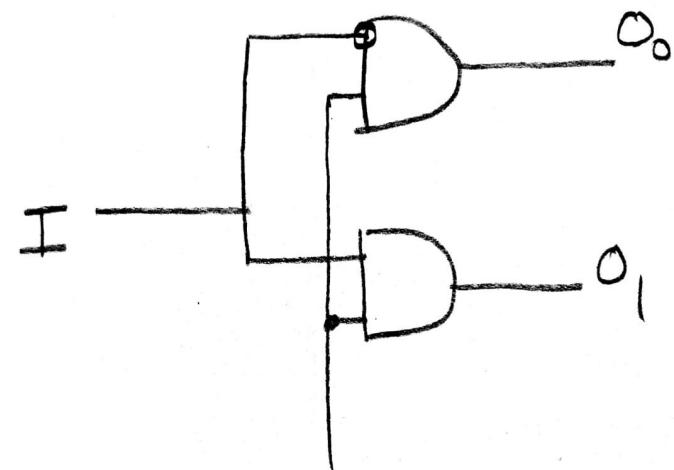
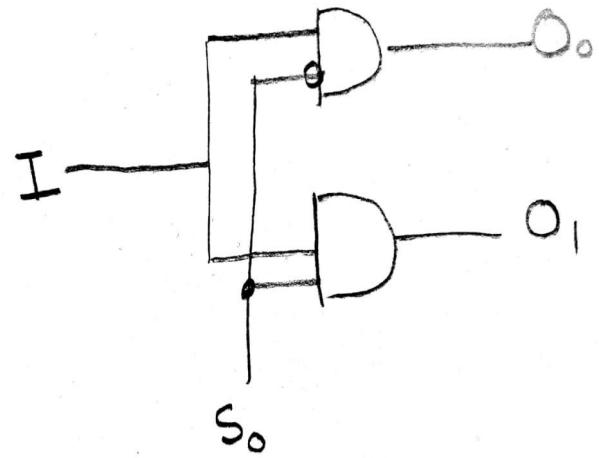
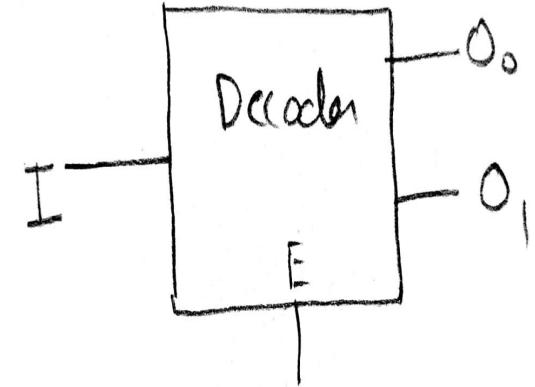
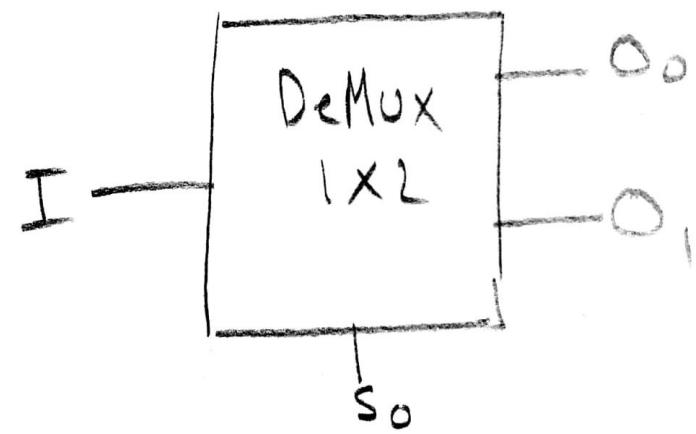


1-to-2 Decoder



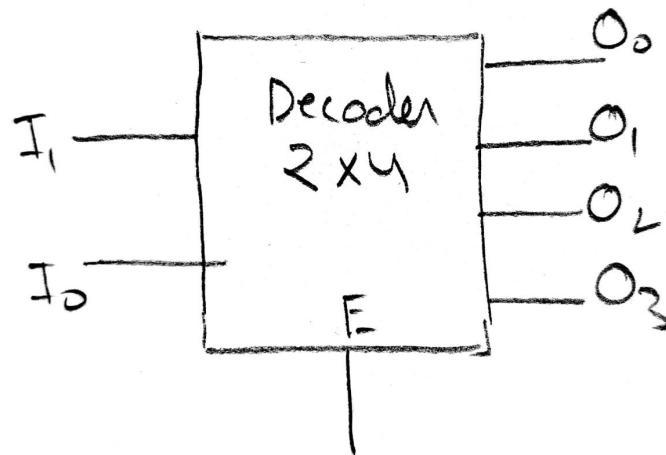
I	O_1	O_0
0	0	1
1	1	0



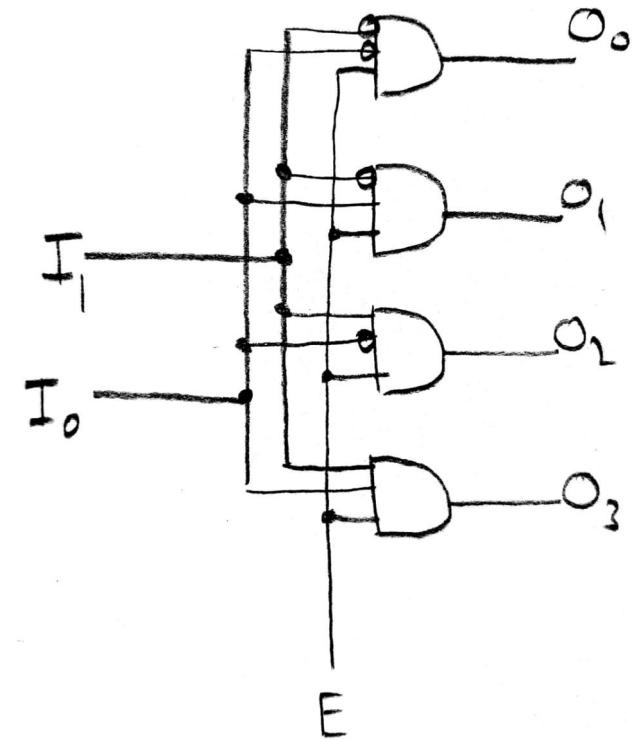


E

2-to-4 Decoder



Input		Output			
I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

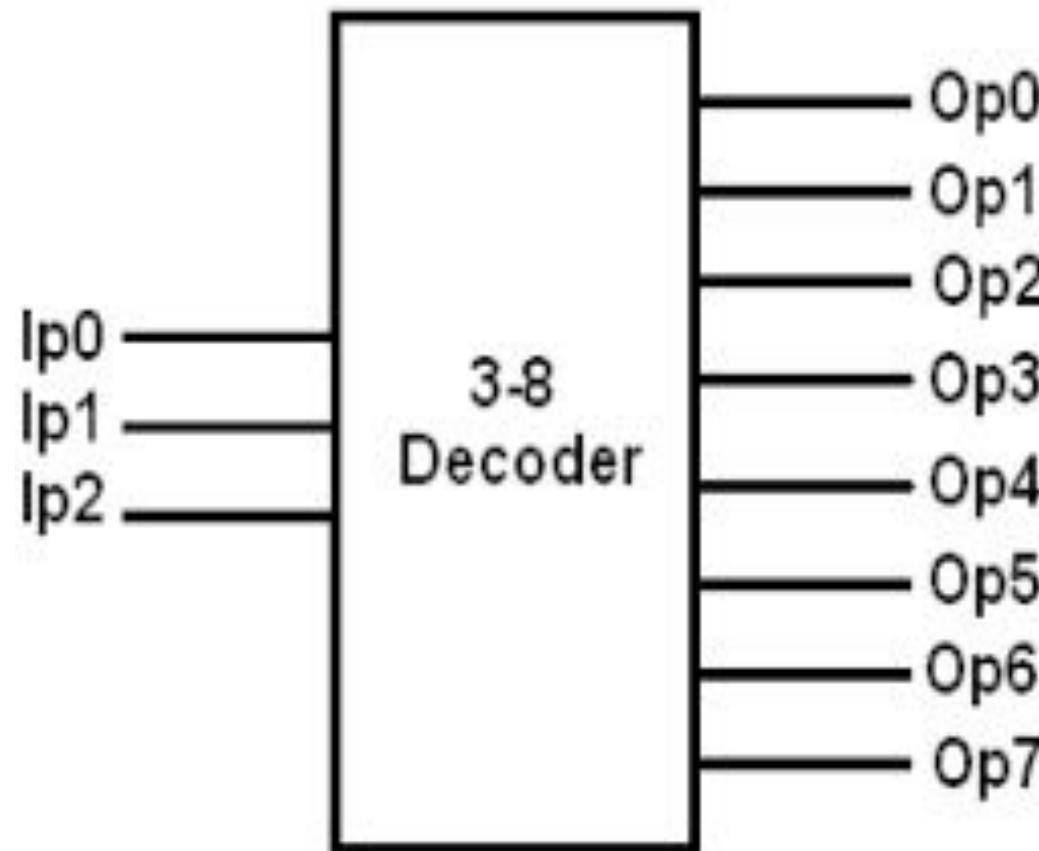


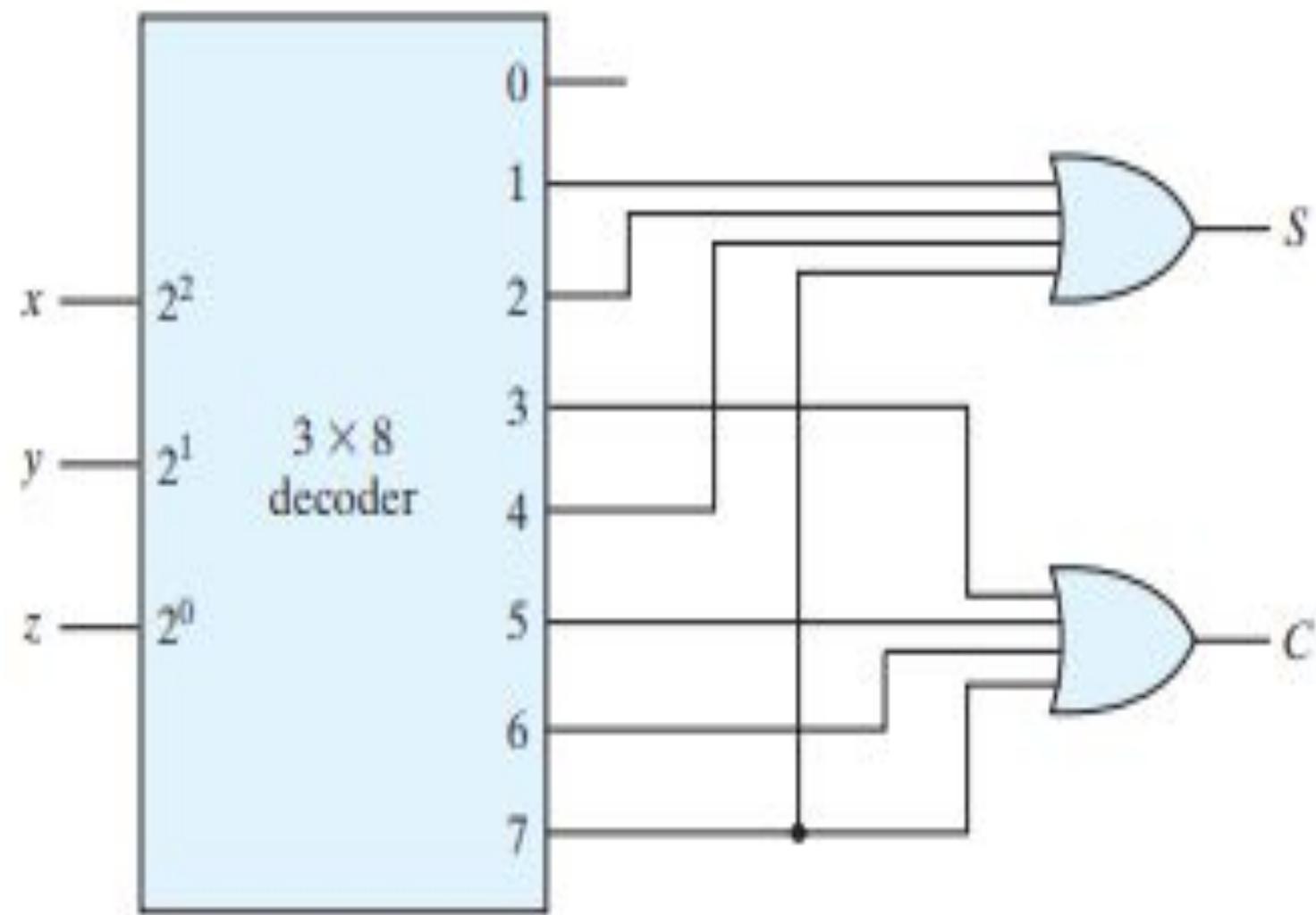
Q Implementation of a full adder with a decoder

From the truth table of the full adder, we obtain the functions for the combinational circuit in sum-of-minterms form:

$$S(x, y, z) = \sum (1, 2, 4, 7)$$

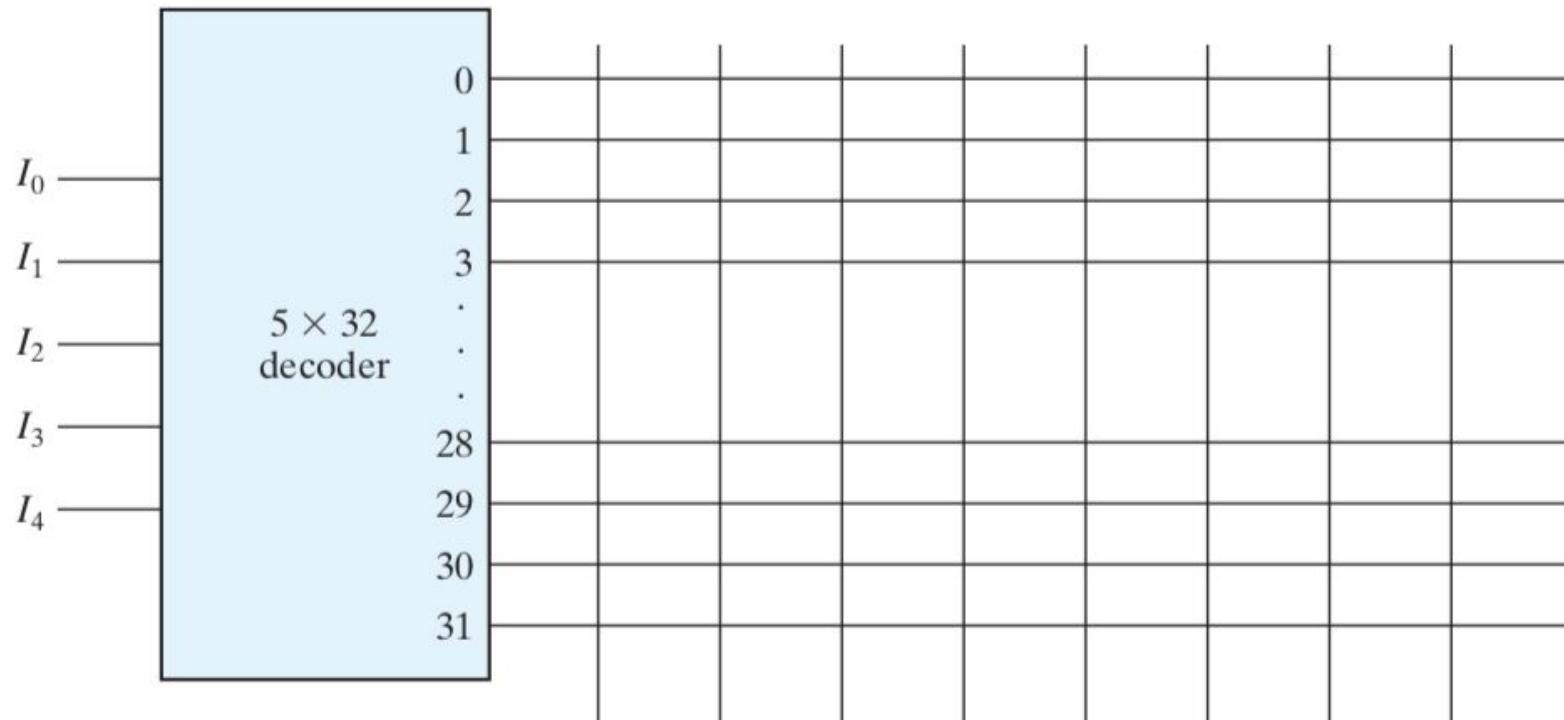
$$C(x, y, z) = \sum (3, 5, 6, 7)$$



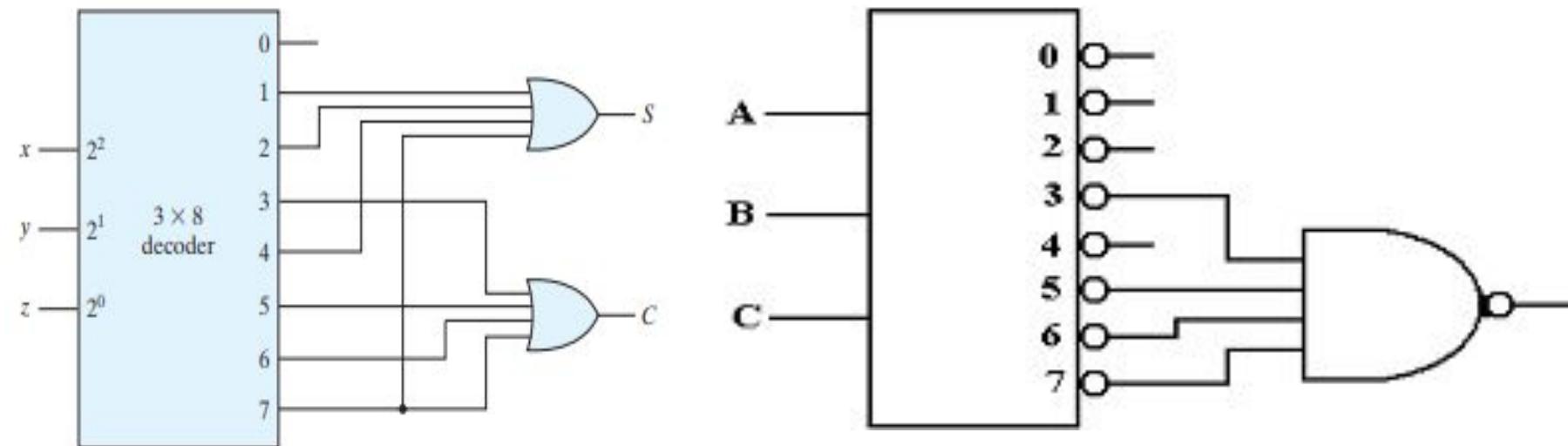


Combinational Logic Implementation

- Any combinational circuit with n inputs and m outputs can be implemented with an n -to- 2^n -line decoder and OR gates.
- The procedure for implementing a combinational circuit by means of a decoder and OR gates requires that the Boolean function for the circuit be expressed as a sum of minterms.
- A decoder is then chosen that generates all the minterms of the input variables. The inputs to each OR gate are selected from the decoder outputs according to the list of minterms of each function.



- **Active High Decoder:-** When output is directly from AND gate, we get exact minterms then, it is called active high decoder. In the 2nd level here, we use OR-gate to find the function.
- **Active Low Decoder:-** In active low decoders, output will be from NAND gates, on the 2nd level we again use NAND gate, as we know NAND-NAND implementation is SAME as AND-OR.



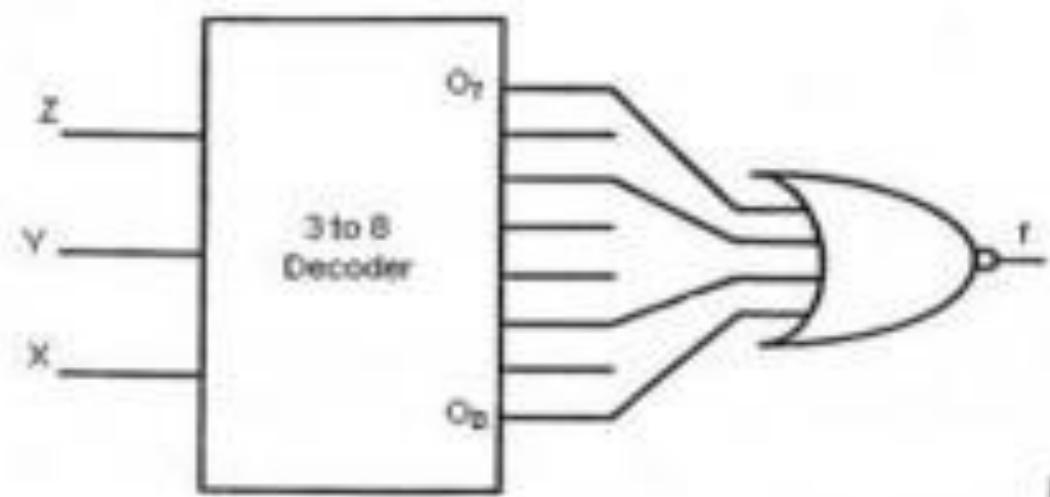
Q What Boolean function does the circuit below realize? **(GATE-2006) (2 Marks)**

(A) $xz + x'z'$

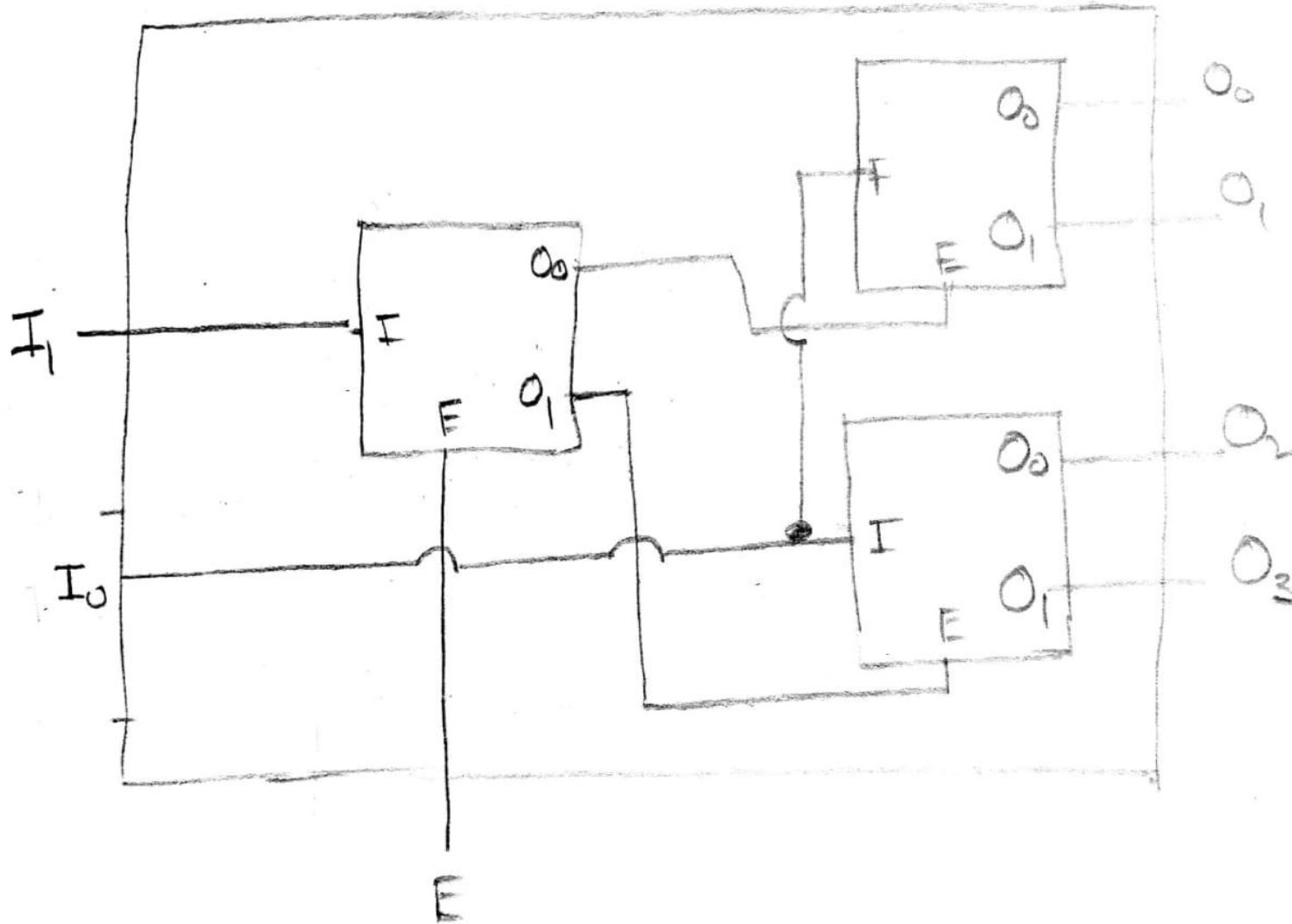
(B) $xz' + x'z$

(C) $x'y' + yz$

(D) $xy + y'z'$



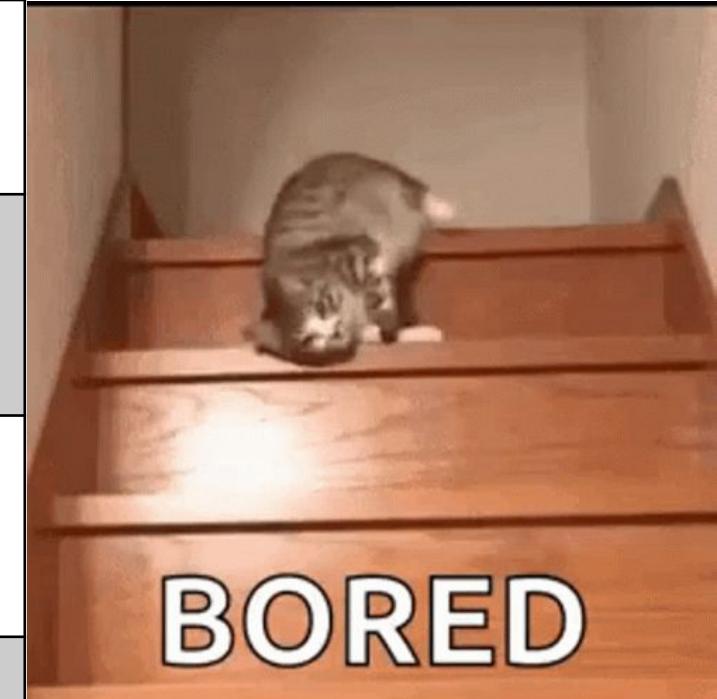
Decoder Expansion



Q 6-to-64 from 2-to-4 ?

Q 7-to-128 from 3-to-8 ?

Target:	$m \times n$
Given:	$p \times q$
No of levels(K)	$\text{floor}(m/p)$
No of Mux at i^{th} level (x_i)	(n/q^{k-i+1})
Total Mux required	
Maximum capacity	$(p \times k) \times q^k$



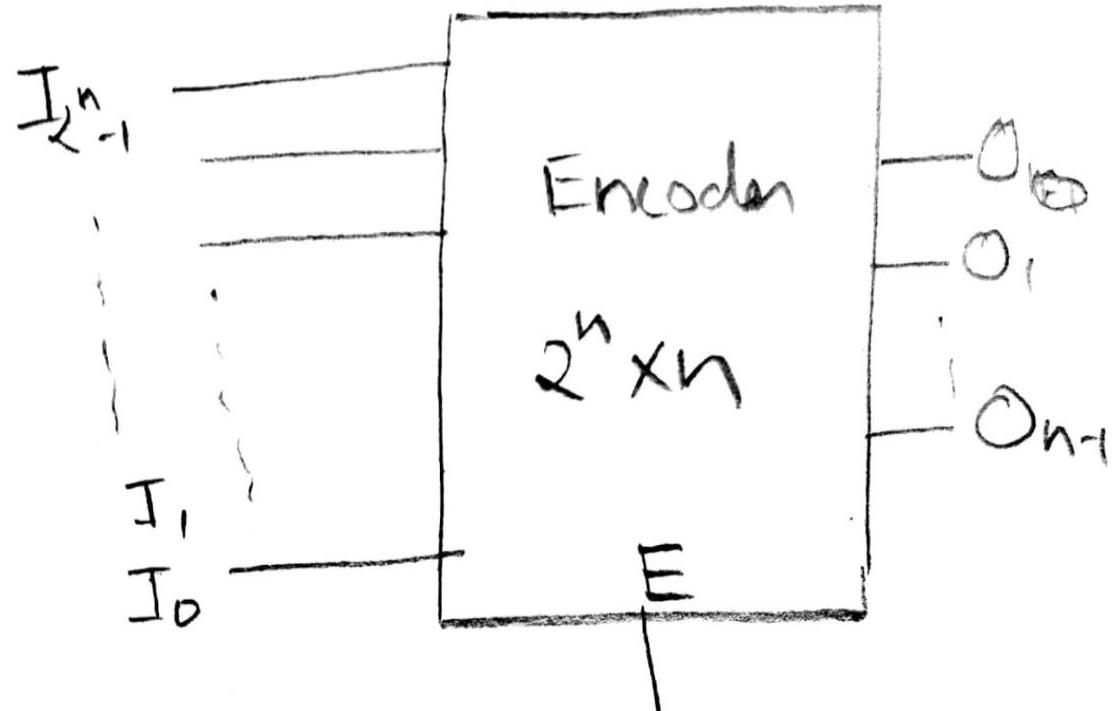
Q How many 3-to-8-line decoders with an enable input are needed to construct a 6-to-64-line decoder without using any other logic gates? **(GATE-2007) (2 Marks)**

- (A) 7**
- (B) 8**
- (C) 9**
- (D) 10**

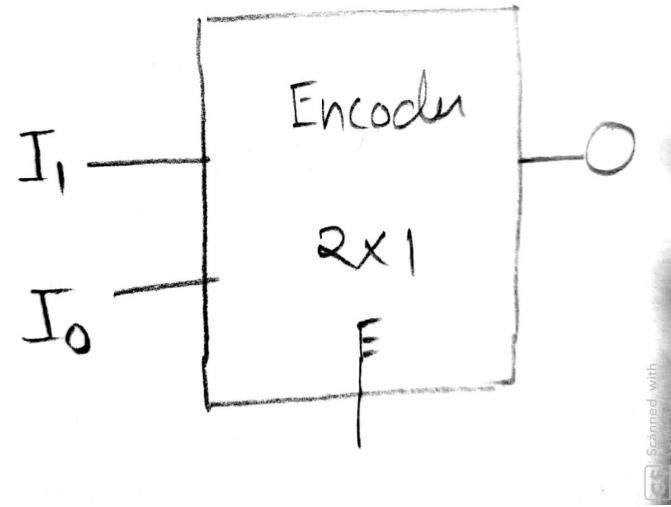
Q Two 3-to-8-line decoders with enable inputs connected to form a 4-to-16-line decoder.

Encoder

- An encoder is a combinational circuit that encode binary information from one of a 2^N input lines and encode it into N output lines, which represent N bit code for the input.
- For simple encoders, it is assumed that only one input line is active at a time.
- Encoder performs the inverse operation of a decoder.

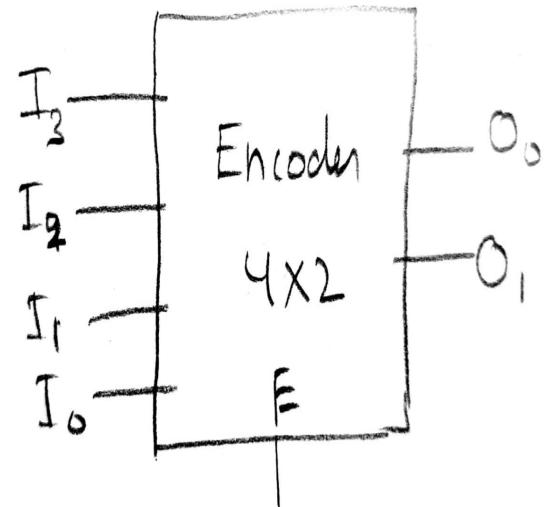


2-to-1 Encoder



I_1	I_0	O_0
0	1	0
1	0	1

4-to-2 Encoder



I_3	I_2	I_1	I_0	O_1	O_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$$\bullet O_0 =$$

$$\bullet O_1 =$$

Priority Encoder

- In some practical cases more than one input can be high at a time, there we can not use simple encoder. In a priority encoder more than one input can be high at a time. A priority encoder is an encoder circuit that includes the priority function.
- The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.
- They are often used to control interrupt requests by acting on the highest priority interrupt input. The *priority encoders* output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored.



$$I_3 > I_2 > I_1 > I_0$$

I_3	I_2	I_1	I_0	O_1	O_0
0	0	0	1		
0	0	1	d		
0	1	d	d		
1	d	d	d		

$$O_0 =$$
$$O_1 =$$

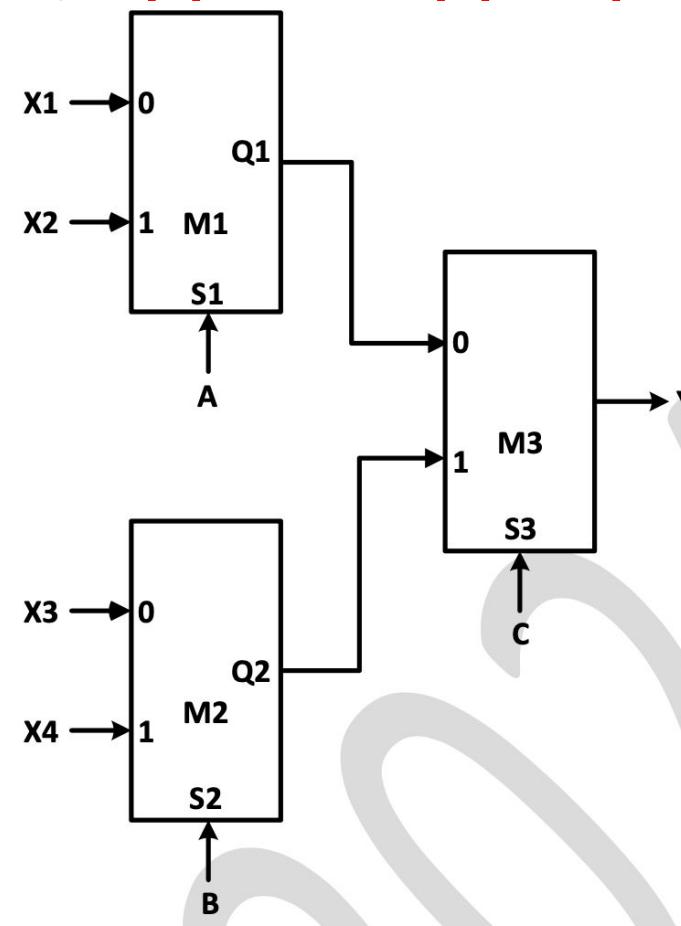
Q In the following truth table, V = 1 if and only if the input is valid. (GATE-2013) (2 Marks)

What function does the truth table represent?

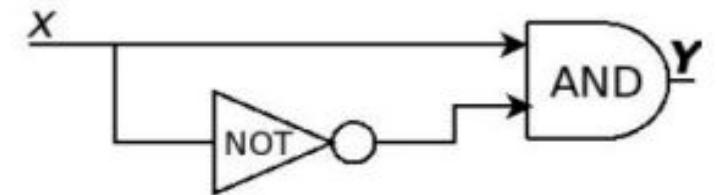
- (A) Priority encoder (B) Decoder (C) Multiplexer (D) Demultiplexer

Inputs				Outputs			
D ₀	D ₁	D ₂	D ₃	X ₀	X ₁	V	
0	0	0	0	x	x	0	
1	0	0	0	0	0	1	
x	1	0	0	0	1	1	
x	x	1	0	1	0	1	
x	x	x	1	1	1	1	

Q. Consider a digital logic circuit consisting of three 2-to-1 multiplexers M_1 , M_2 , and M_3 as shown below. X_1 and X_2 are inputs of M_1 . X_3 and X_4 are inputs of M_2 . A, B and C are select lines of M_1 , M_2 , and M_3 respectively. For an instance of inputs $X_1=1$, $X_2=1$, $X_3=0$, and $X_4=0$ the number of combinations of A,B,C that given the output $Y=1$ is _____. (Gate 2024,CS) (2 Marks) (NAT)



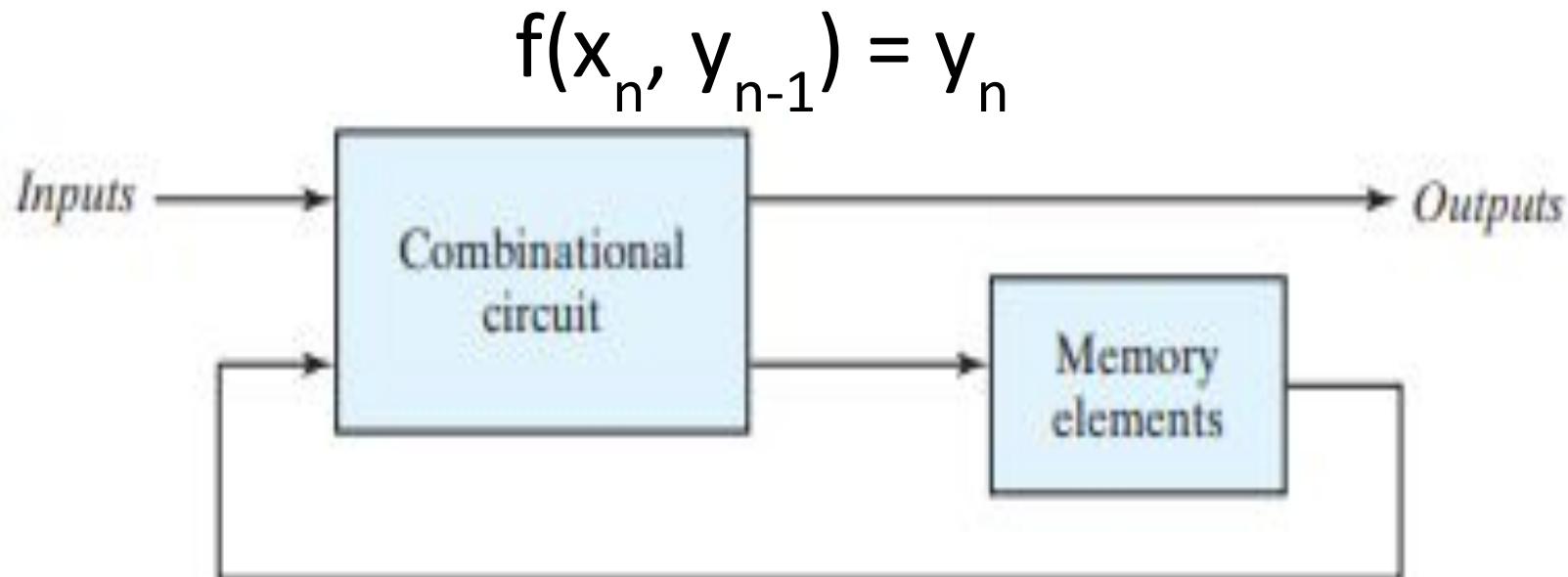
Q. Consider the circuit shown below where the gates may have propagation delays. Assume that all signal transitions occur instantaneously and that wires have no delays. Which of the following statements about the circuit is/are Correct (4 Marks) (MSQ)



- (a) With no propagation delays, the output Y is always logic Zero
- (b) With no propagation delays, the output Y is always logic One
- (c) With propagation delays, the output Y can have transient logic One after X transitions from logic Zero to logic One
- (d) With propagation delays, the output Y can have a transient logic Zero after X transitions from logic One to logic Zero

SEQUENTIAL CIRCUITS

- Sequential Circuits consists of a combinational circuit to which memory elements are connected to form a feedback path. The memory elements are devices capable of storing binary information.
- The binary information stored in these elements at any given time defines the state of the sequential circuit at that time.
- The sequential circuit receives binary information from external inputs (x_n) that, together with the present state (y_{n-1}) of the memory elements, determine the binary value of the outputs(y_n).
- A sequential circuit is specified by a time sequence of inputs, outputs, and internal states.

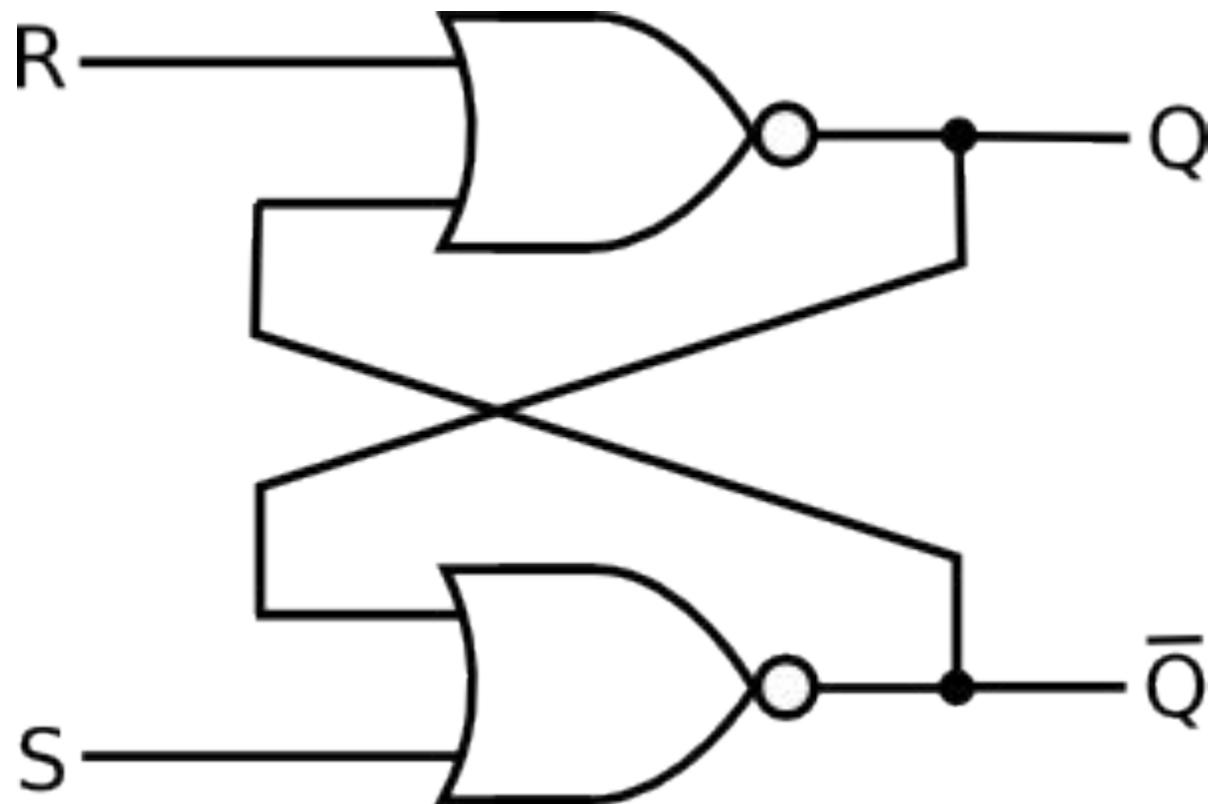


Latches

- Latch means to hold something or something which do not change.
- latches are the basic building blocks of any flip flop and they are capable of holding 1 bit until necessary.
- Storage elements that operate with signal levels are referred to as latches.
- Latches are level sensitive devices.

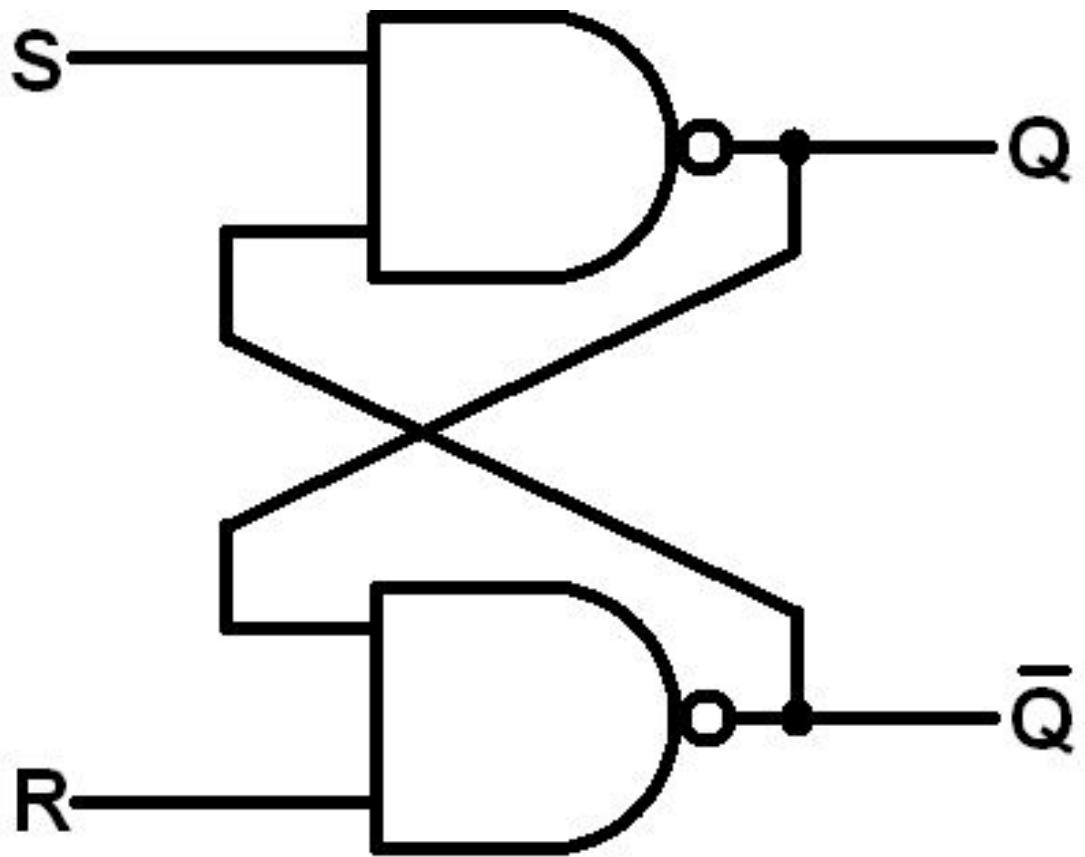
NOR Latch

- The SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates, and two inputs labeled S for set and R for reset.
- Outputs Q_n and \bar{Q}_n are the complement of each other, in valid scenario.



S	R	Q_n	Q_{n+1}
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

NAND Latch



S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

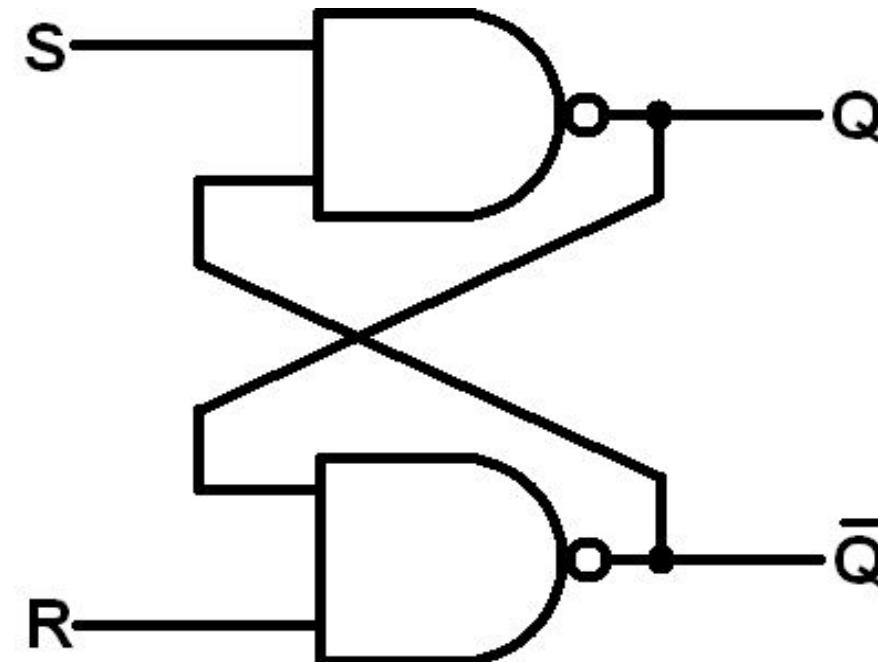
Q In an SR latch made by cross-coupling two NAND gates, if both S and R inputs are set to 0, then it will result in **(GATE-2004) (2 Marks)**

- (A)** $Q = 0, Q' = 1$

- (B)** $Q = 1, Q' = 0$

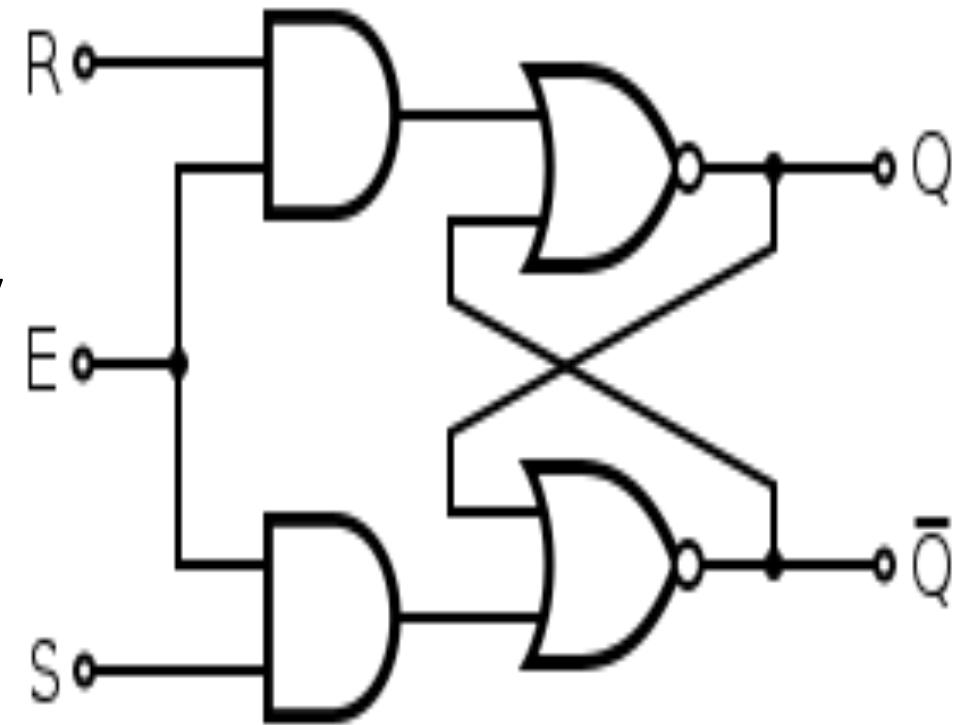
- (C)** $Q = 1, Q' = 1$

- (D)** Indeterminate states

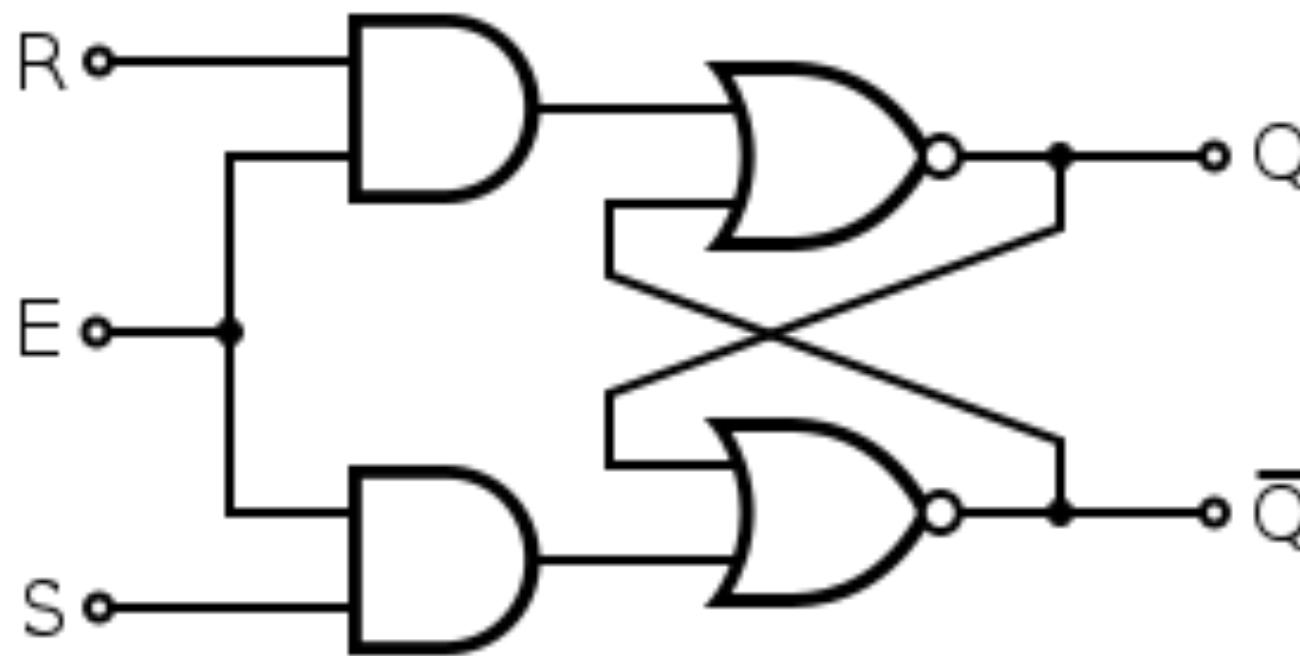


SR flip flop

1. The operation of the basic SR latch can be modified by providing an additional input signal that determines (controls), when the state of the latch can be changed by determining whether S and R (or S and R) can affect the circuit.
2. It consists of the basic SR latch and two additional AND gates. The control input CP / E_n acts as an enable signal for the other two inputs.
3. The outputs of the AND gates stay at the 0 as long as the enable signal remains at 0 as one input of AND gate gets 0 resulting in 0 as output. When the enable input goes to 1, information from the S or R input is allowed to affect the latch.



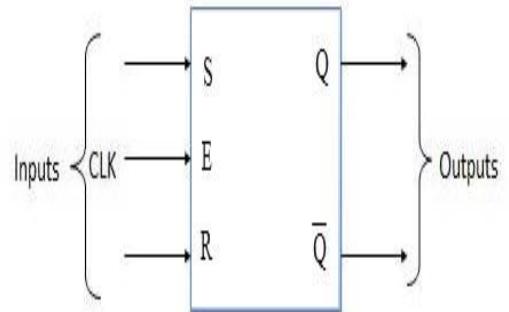
Implementation



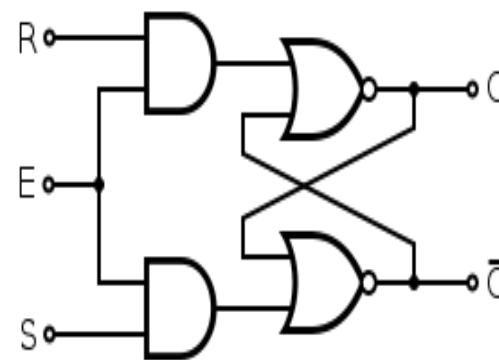
Truth Table

S	R	Q_n	Q_{n+1}
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Block Diagram



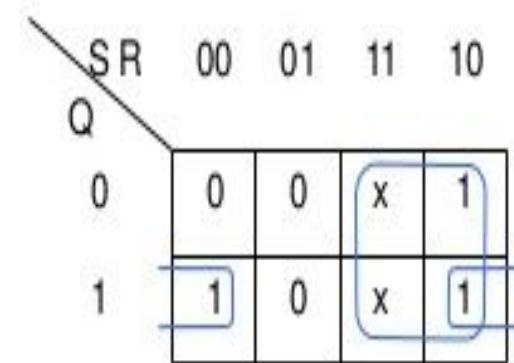
Implementation



Truth Table

S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

K-Map



Characteristics Equation

Function Table

Excitation Table

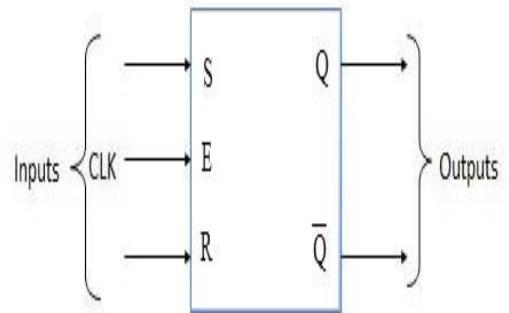
State Diagram

$$Q_{n+1} =$$

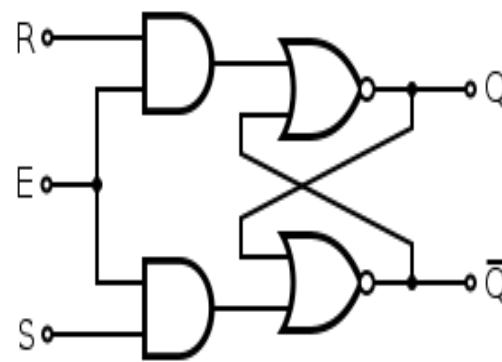
S	R	Q_{n+1}
0	0	
0	1	
1	0	
1	1	

Q_n	Q_{n+1}	S	R
0	0		
0	1		
1	0		
1	1		

Block Diagram



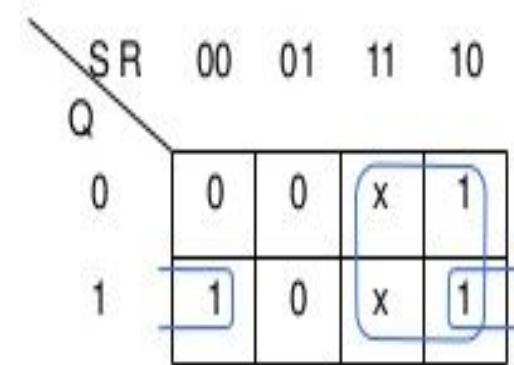
Implementation



Truth Table

S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

K-Map



Characteristics Equation

$$Q_{n+1} = S + R' Q_n$$

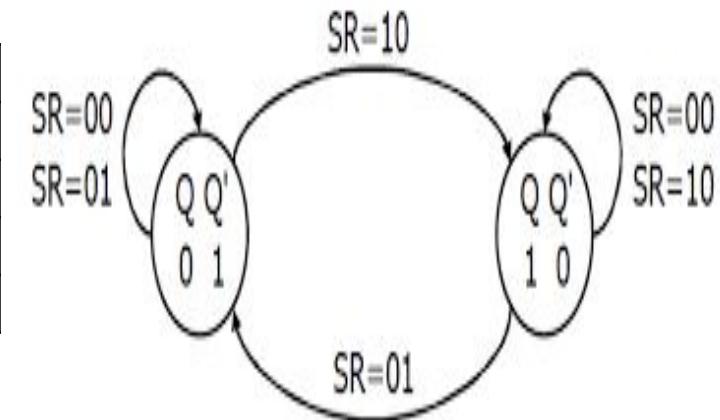
Function Table

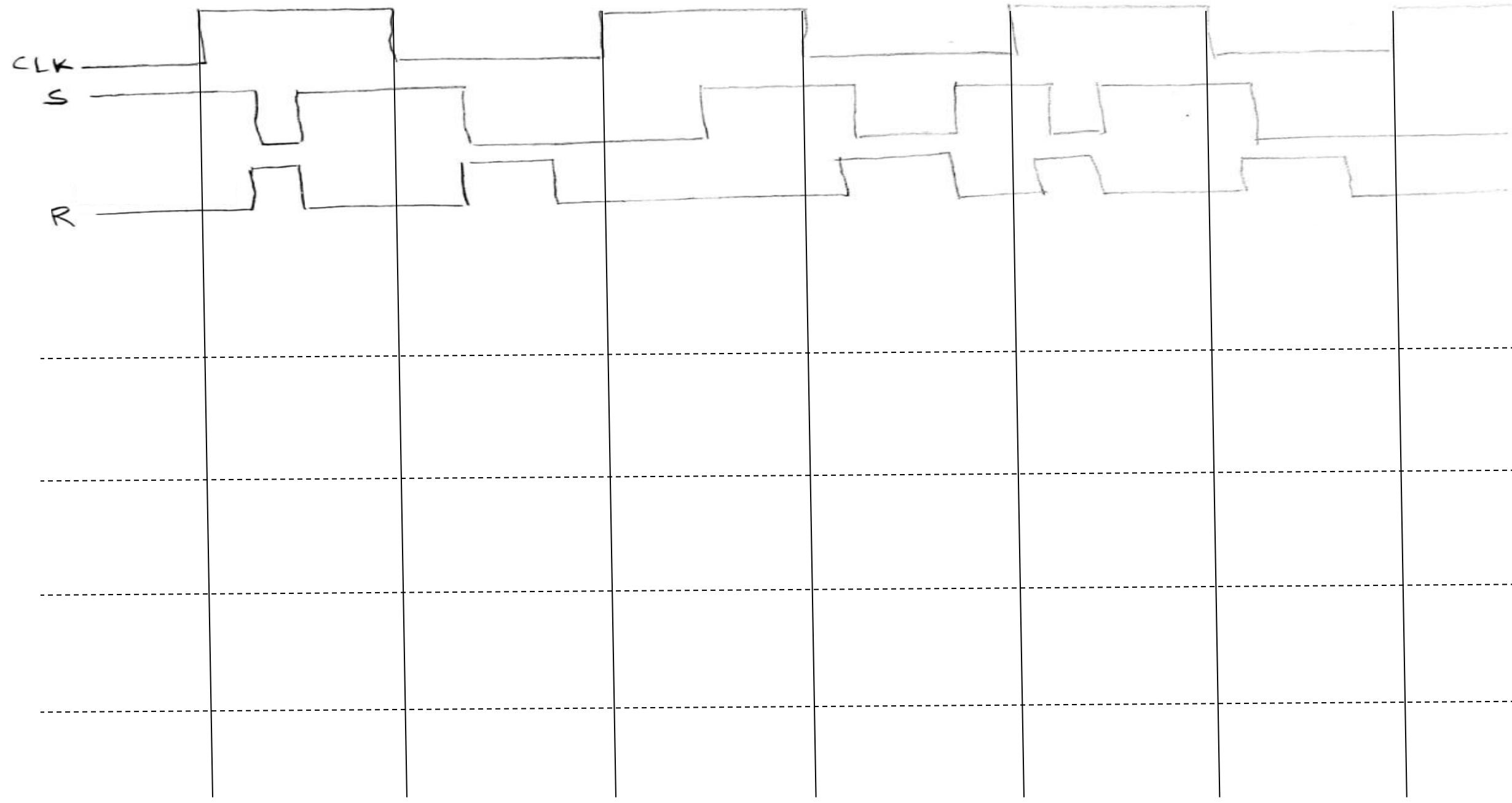
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	X

Excitation Table

Q_n	Q_{n+1}	S	R
0	0	0	d
0	1	1	0
1	0	0	1
1	1	d	0

State Diagram

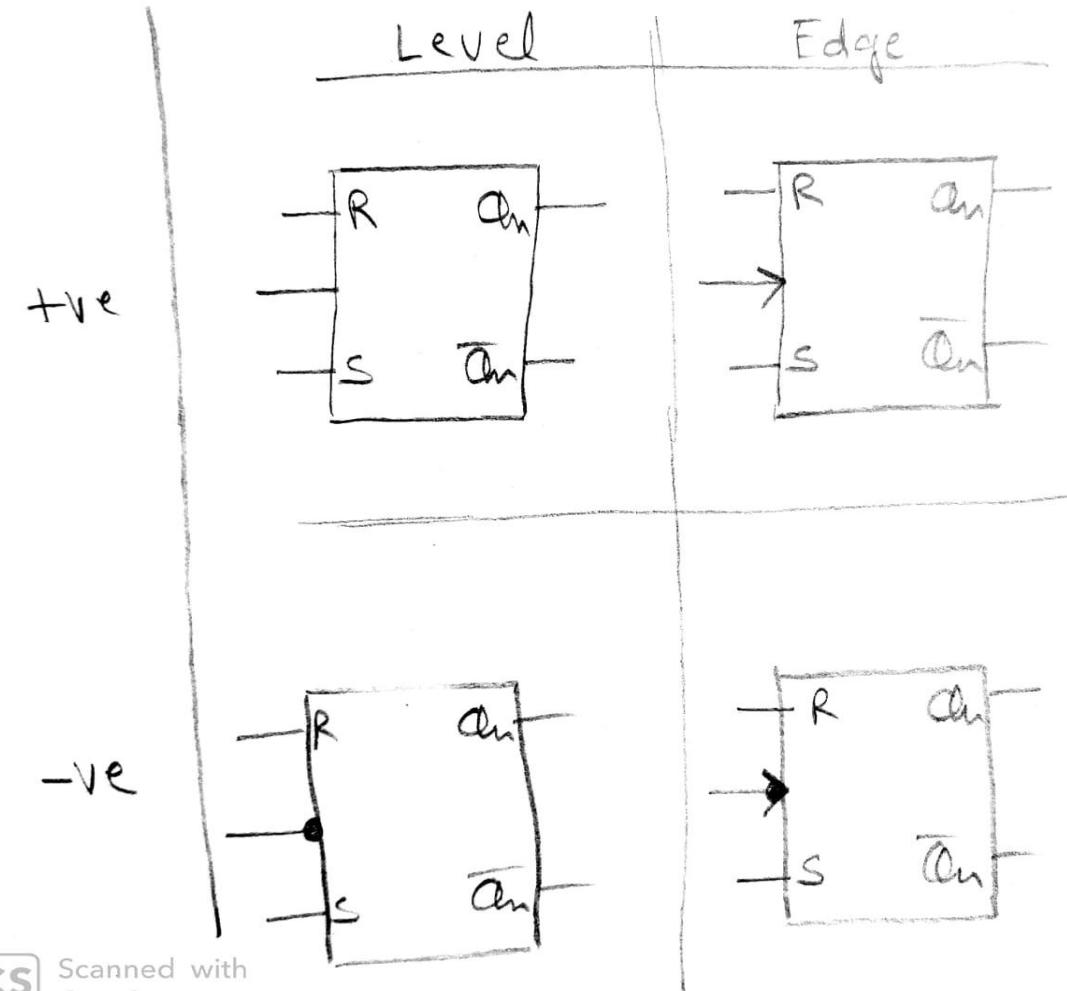




- Edge triggering is always preferred over level triggering because in edge triggering system is in transparent mode for very less amount of time, so system will be stable.

- Order of Priority

- -ve edge
- +ve edge
- -ve level
- +ve level



Scanned with
CamScanner

Notes

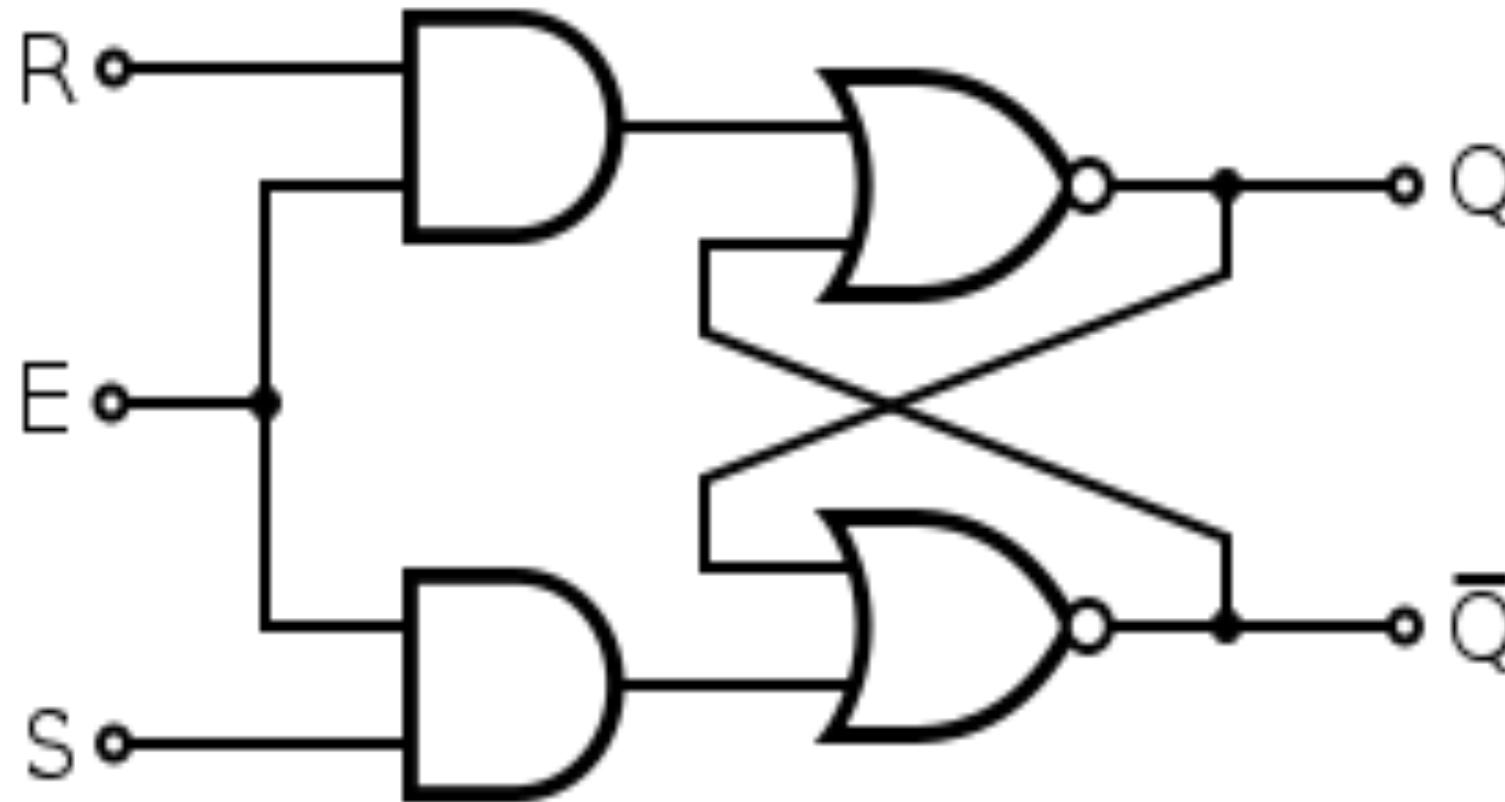
- The key to the proper operation of a flip-flop is to trigger it ***only during a signal transition(edge).***
- This can be accomplished by eliminating the feedback path that is inherent in the operation of the sequential circuit using latches.
- A clock pulse goes through two transitions: from 0 to 1(+ve edge) and the return from 1 to 0 (-ve edge).
- The way that a latch can be modified to form a flip-flop. To produce a flip-flop that triggers only during a signal transition(edge) (from 0 to 1 or from 1 to 0) of the synchronizing signal (clock) and is disabled during the rest of the clock pulse.
- Latches are the basic circuits from which all flip-flops are constructed.
- The storage elements (memory) used in clocked sequential circuits are called flipflops.
- A flip-flop is a binary storage device capable of storing one bit of information. In a stable state, the output of a flip-flop is either 0 or 1 (it is called bi-stable multi-vibrator).

Notes

- A flip-flop is said to be stable if it has complementary behavior.
- Latches are level sensitive devices; flip-flops are edge-sensitive devices.
- Storage elements that operate with signal levels are referred to as latches; those controlled by a clock transition are flip-flops.
- Flip-flops can be either level-triggered (asynchronous, transparent or opaque) or edge-triggered (**synchronous**, or **clocked**).
- The term flip-flop has historically referred generically to both level-triggered and edge-triggered circuits that store a single bit of data using gates.
- Recently, some authors reserve the term *flip-flop* exclusively for discussing clocked circuits;
 - the simple ones are commonly called *transparent latches*. Using this terminology, a level-sensitive flip-flop is called a transparent latch
 - Whereas an edge-triggered flip-flop is simply called a flip-flop.
- Using either terminology, the term "flip-flop" refers to a device that stores a single bit of data. The terms "edge-triggered", and "level-triggered" may be used to avoid ambiguity.

JK flip flop

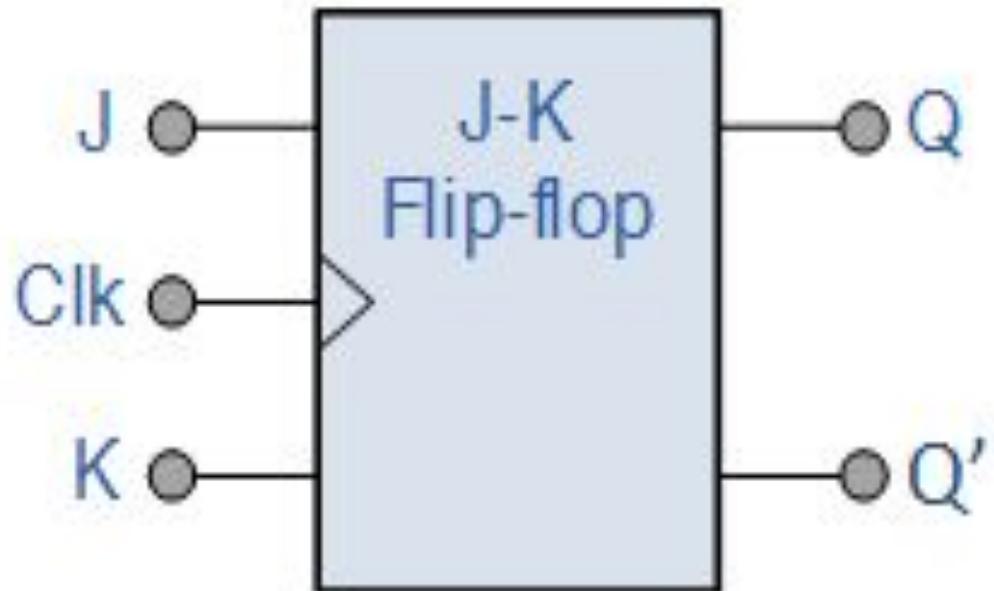
- SR flip flop when both S and R = 1, results in invalid output. To resolve the problem we use JK Flip Flop. We take a feedback from the outputs.



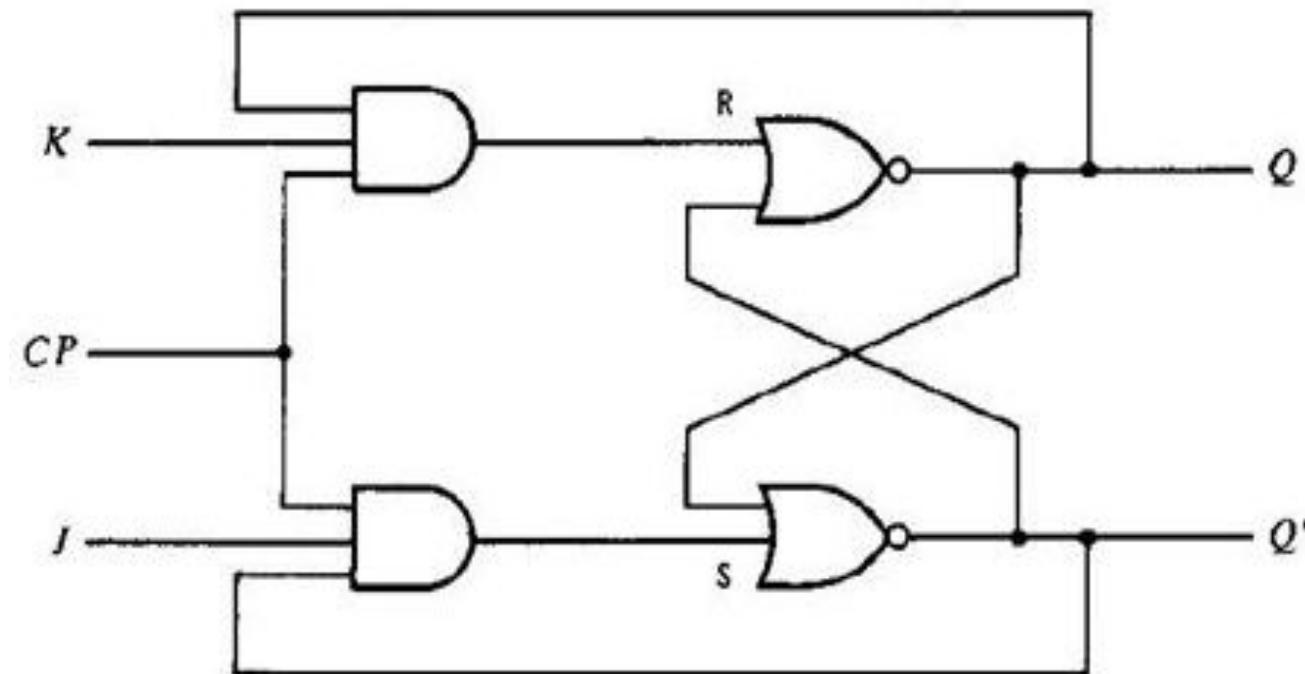
JK flip flop

- SR flip flop when both S and R = 1, results in invalid output. To resolve the problem we use JK Flip Flop. We take a feedback from the outputs.

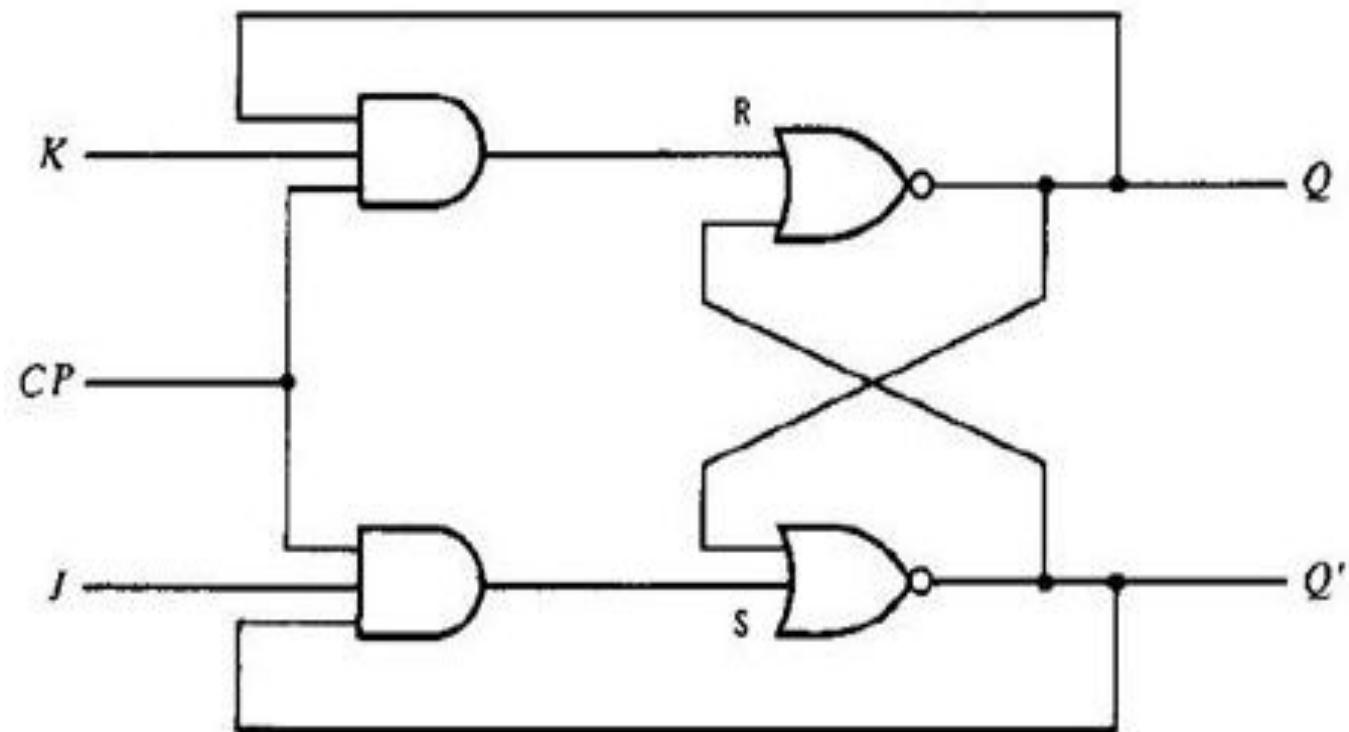
Block Diagram



Implementation



Truth Table



J	K	Q_n	Q_{n+1}
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

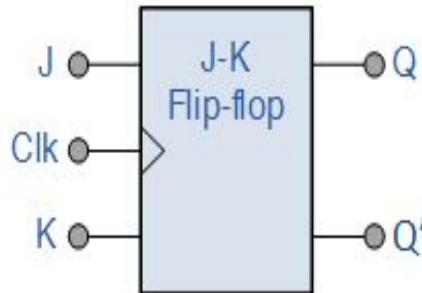
Truth Table

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

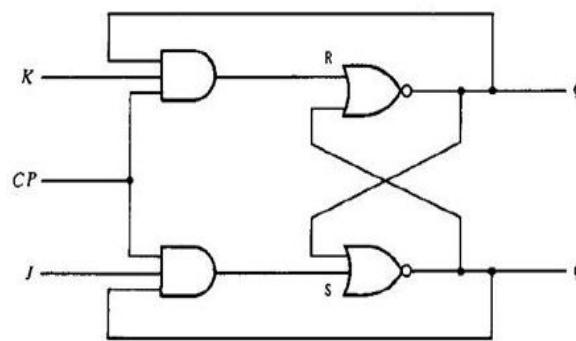
K-Map

	JK	$J'K'$	$J'K$	JK	JK'
q		00	01	11	10
q'	00				
q	01				

Block Diagram



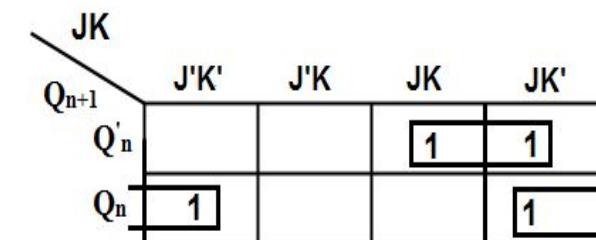
Implementation



Truth Table

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

K-Map



Characteristics Equation

$$Q_{n+1} =$$

Function Table

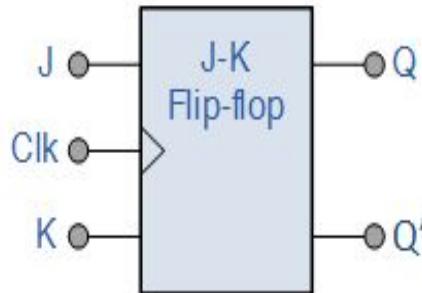
J	K	Q_{n+1}
0	0	
0	1	
1	0	
1	1	

Excitation Table

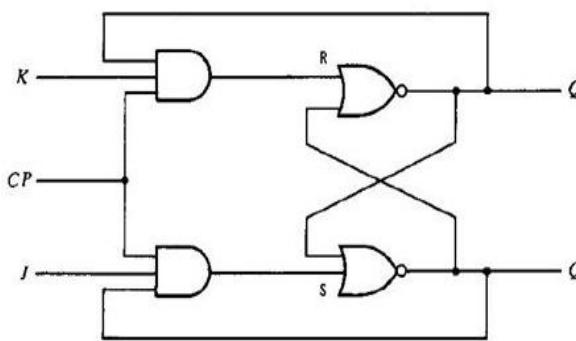
Q_n	Q_{n+1}	J	K
0	0		
0	1		
1	0		
1	1		

State Diagram

Block Diagram



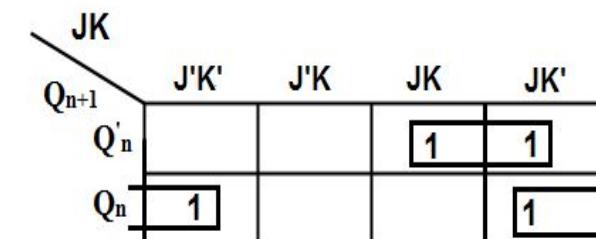
Implementation



Truth Table

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

K-Map



Characteristics Equation

$$Q_{n+1} = J Q'_n + K' Q_n$$

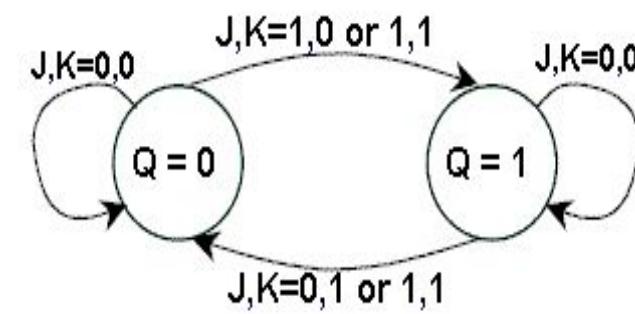
Function Table

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Q'_n

Excitation Table

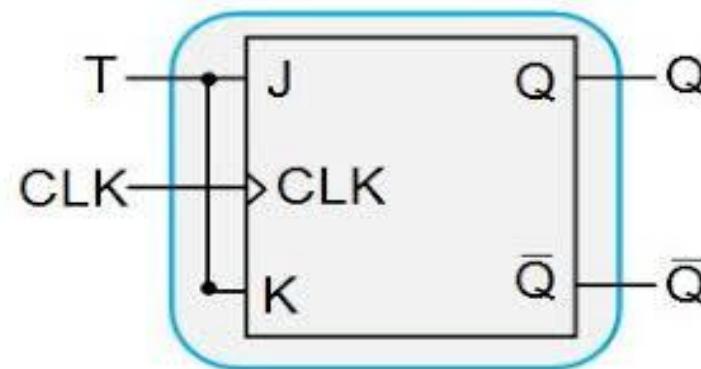
Q_n	Q_{n+1}	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

State Diagram



T(Toggle) flip flop

- The T (toggle) flip-flop is a complementing flip-flop and can be obtained from a JK flip-flop when inputs J and K are tied together.

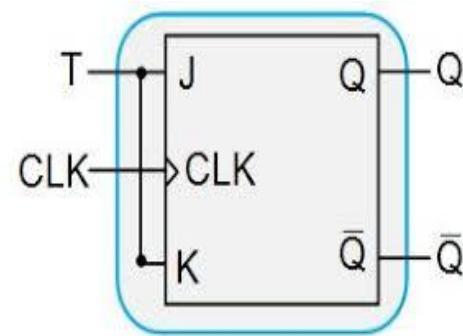


J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Truth Table

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

Block Diagram



Truth Table

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

K-Map

	T	T'	T
q		0	1
q'	0		
q	1		

Characteristics Equation

Function Table

Excitation Table

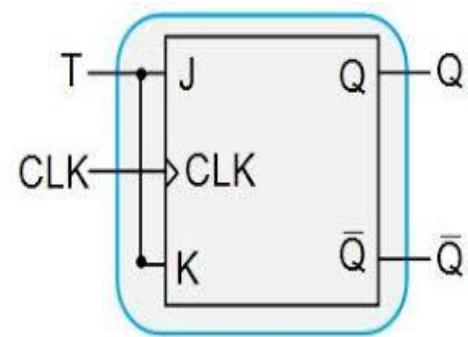
State Diagram

$$Q_{n+1} =$$

T	Q_{n+1}
0	
1	

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

Block Diagram



Truth Table

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

K-Map

	T	T'	T
q		0	1
q'	0		1
q	1	1	

Characteristics Equation

$$Q_{n+1} = T \oplus Q_n$$

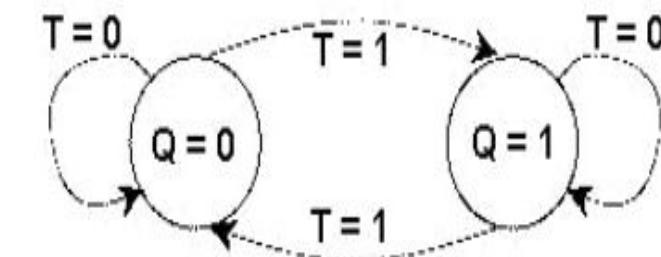
Function Table

T	Q_{n+1}
0	Q_n
1	Q'_n

Excitation Table

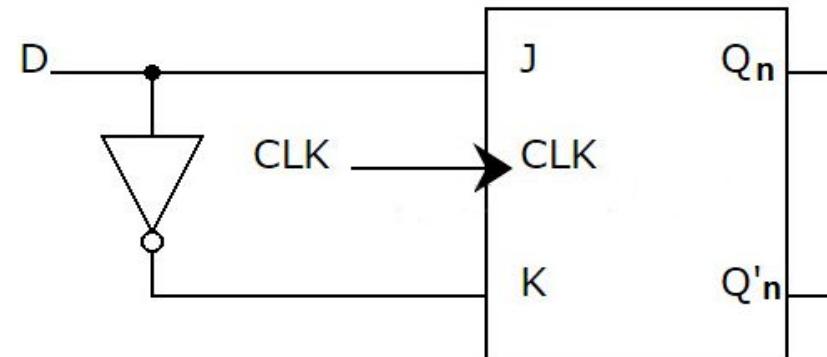
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

State Diagram



D flip flop

- The D (Data/Delay) flip-flop, tracks the input at D and produces the same value as output.

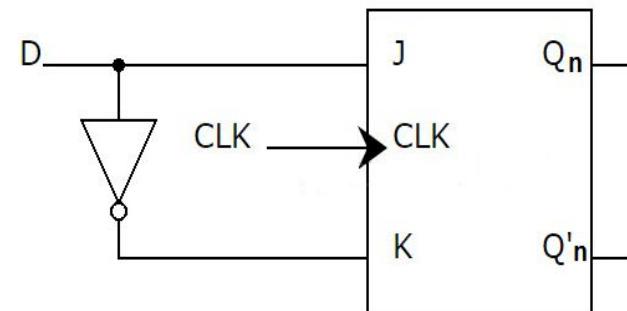


J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Truth Table

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

Block Diagram



Truth Table

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

K-Map

	D	D'	D
q		0	1
q'	0		
q	1		

Characteristics Equation

$$Q_{n+1} =$$

D	Q_{n+1}
0	
1	

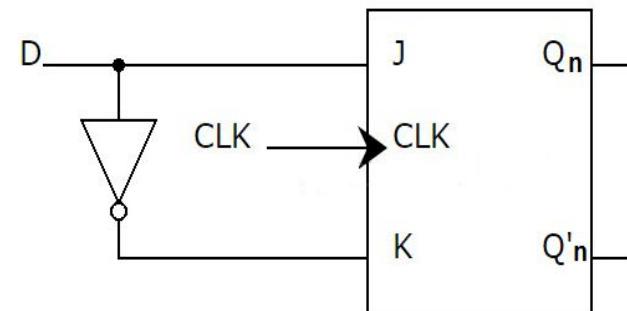
Function Table

Excitation Table

State Diagram

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

Block Diagram



Truth Table

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

K-Map

	D	D'	D
q		0	1
q'	0		1
q	1		1

Characteristics Equation

$$Q_{n+1} = D$$

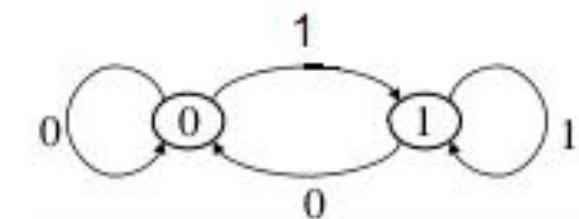
Function Table

D	Q_{n+1}
0	0
1	1

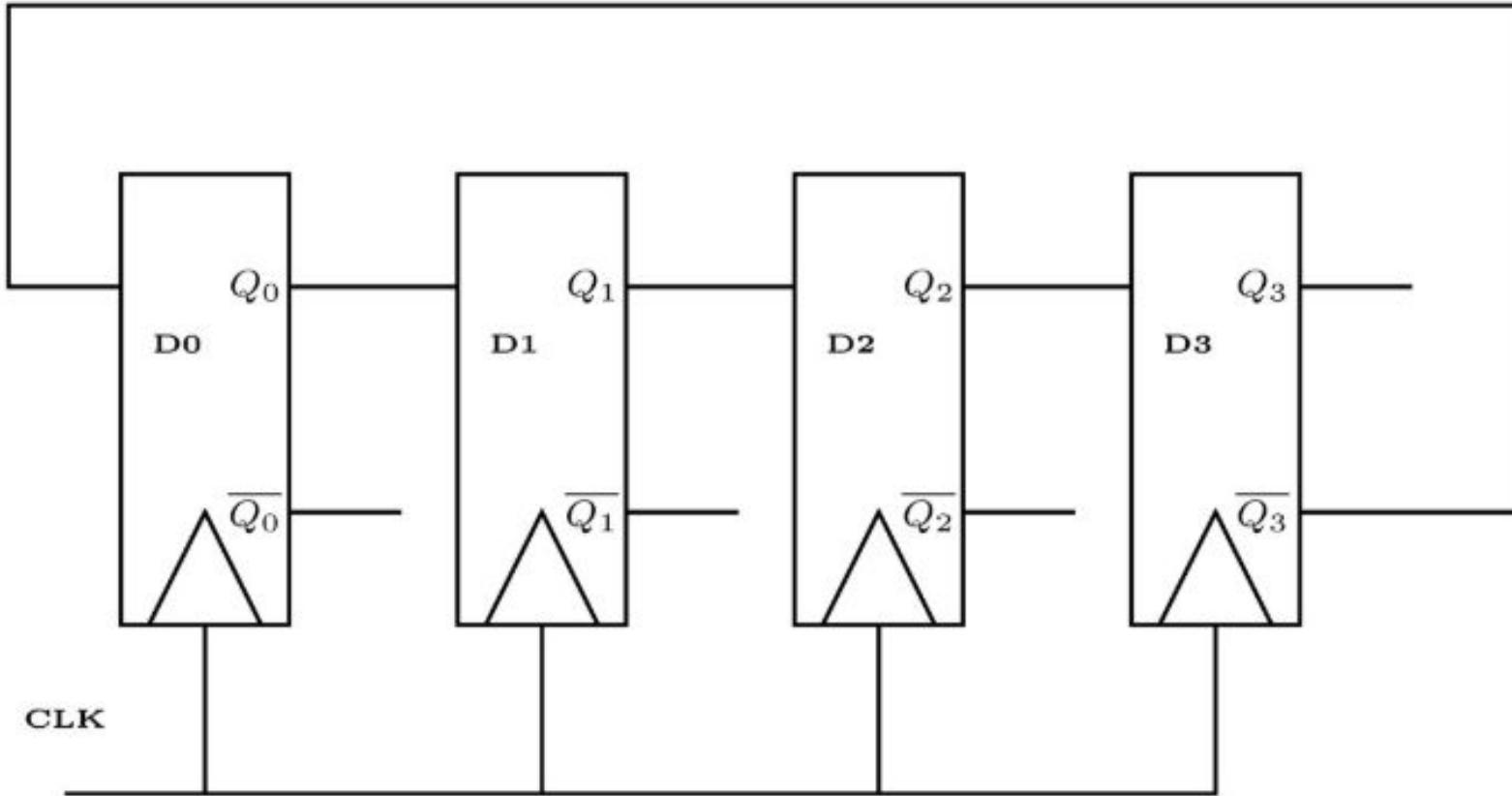
Excitation Table

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

State Diagram



Q. Consider the given sequential circuit designed using D-Flip-flops. The circuit is initialized with some value (initial state). The number of distinct states the circuit will go through before returning back to the initial state is _____. (Answer in integer) (GATE-2025)



Flip Flops Conversion

1. We will require the **Characteristics Table** of target flip flop.
2. We will require the **Excitation Table** of given flip flop.
3. Determine the excitation values for characteristics table.
4. Obtain the expressions for input of given flip flop in terms of target.

Convert D Flip Flop to T Flip Flop

Characteristics table of T- Flip Flop

T	Q _n	Q _{n+1}
0	0	
0	1	
1	0	
1	1	

Excitation Table of D flip flop

T	Q _n	Q _{n+1}	D
0	0	0	
0	1	1	
1	0	1	
1	1	0	

Q _n	Q _{n+1}	D
0	0	
0	1	
1	0	
1	1	

Convert D Flip Flop to T Flip Flop

Excitation values for characteristics table

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

Obtaining simplified expression

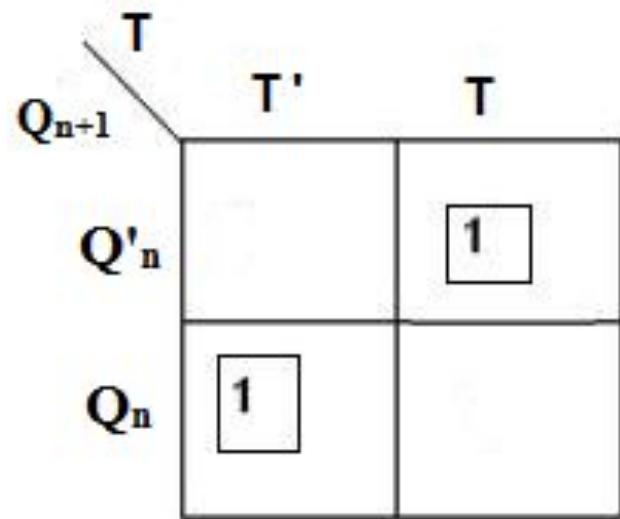
T	Q_n	D
0	0	
0	1	
1	0	
1	1	

Using K-Map to Simplify

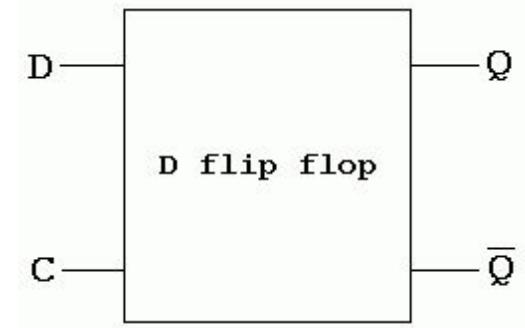
	T	T'	T
q		0	1
q'	0		
q	1		

Convert D Flip Flop to T Flip Flop

Using K-Map to Simplify



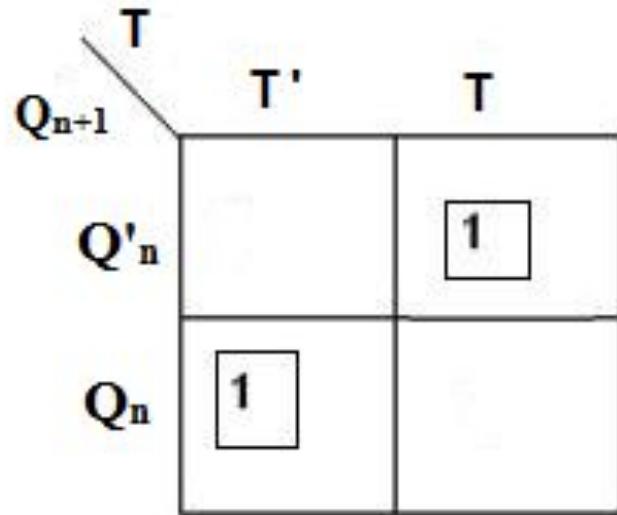
Block Diagram



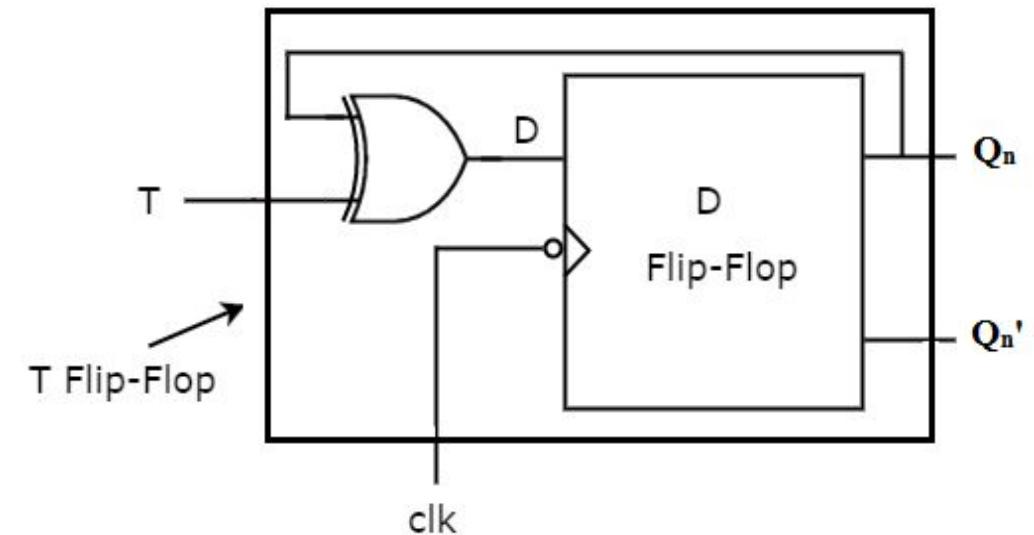
$$D = Q_n \oplus T$$

Convert D Flip Flop to T Flip Flop

Using K-Map to Simplify



Block Diagram



$$D = Q_n \oplus T$$

Convert T flip flop to JK flip flop

Characteristics table of JK- Flip Flop

J	K	Q_n	Q_{n+1}
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Excitation Table of T flip flop

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

J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

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

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

Convert T flip flop to JK flip flop

Excitation values for characteristics table

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

Obtaining simplified expression

J	K	Q_n	T
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Convert T flip flop to JK flip flop

Using K-Map to Simplify

J	K	Q_n	T
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

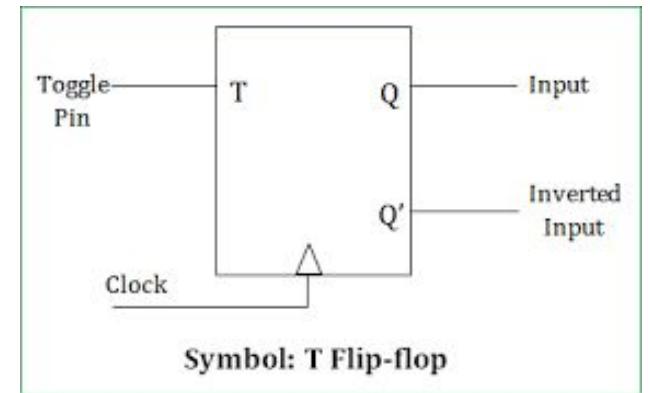
	JK	$J'K'$	$J'K$	JK	JK'
q		00	01	11	10
q'	00				
q	01				

$$T =$$

Convert T flip flop to JK flip flop

Block Diagram

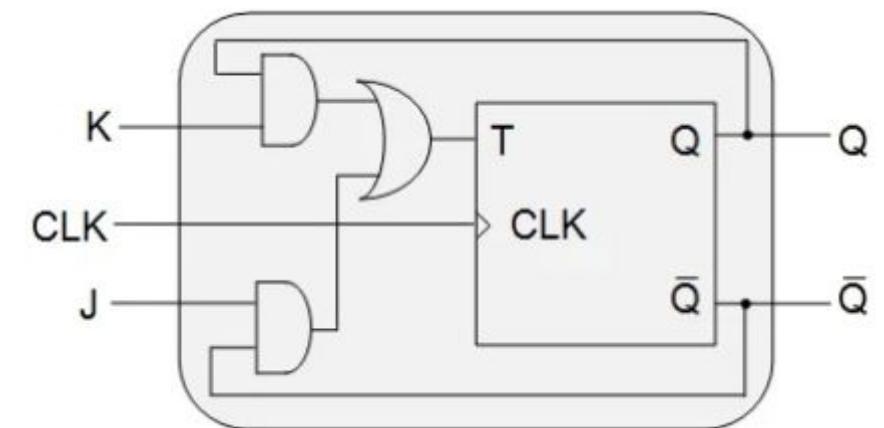
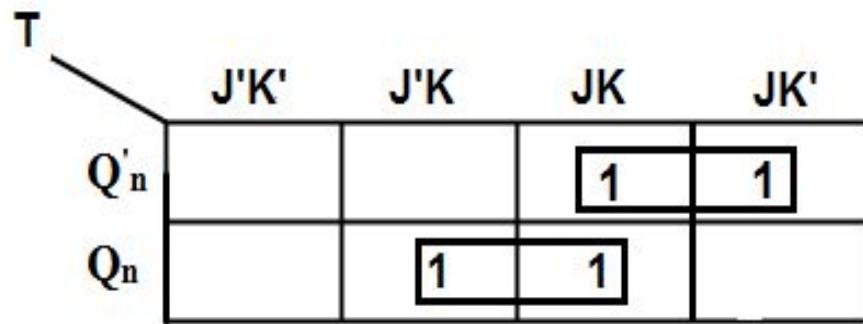
T =



Convert T flip flop to JK flip flop

Using K-Map to Simplify

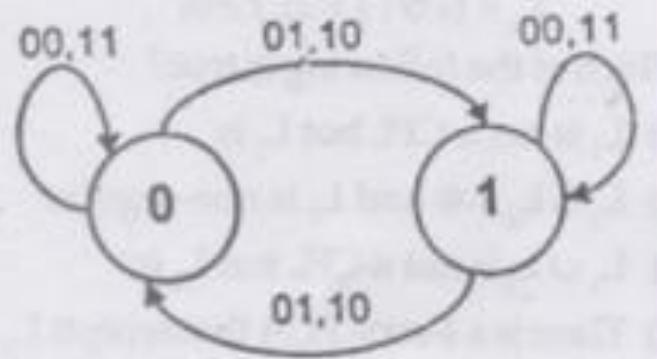
Block Diagram



$$T = J Q_n' + K Q_n$$

Q Consider the following state diagram and its realization by a JK flip flop, the combinational circuit generates J and K in terms of x, y and Q. The Boolean expressions for J and K are : **(GATE-2008) (2 Marks)**

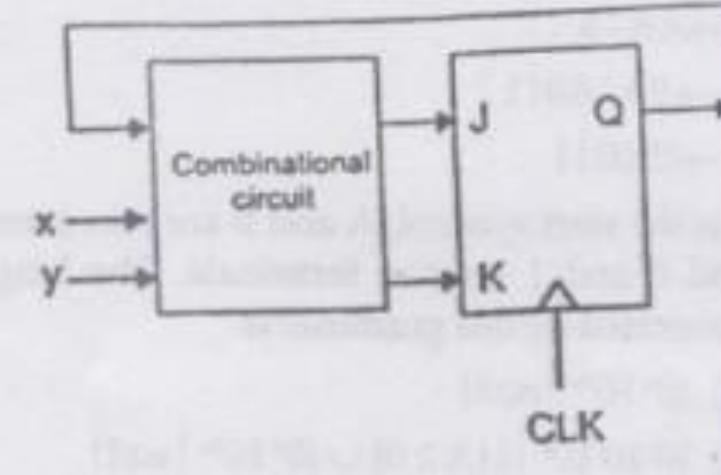
- (A) $(x \oplus y)' \text{ and } x' \oplus y'$
 (B) $(x \oplus y)' \text{ and } x \oplus y$
 (C) $x \oplus y \text{ and } (x \oplus y)'$
 (D) $x \oplus y \text{ and } x \oplus y$



x	y	Q_n	Q_{n+1}
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

x	y	Q_n	Q_{n+1}	J	K
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Q_n	Q_{n+1}	J	K
0	0		
0	1		
1	0		
1	1		



x	y	Q_n	J	K
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

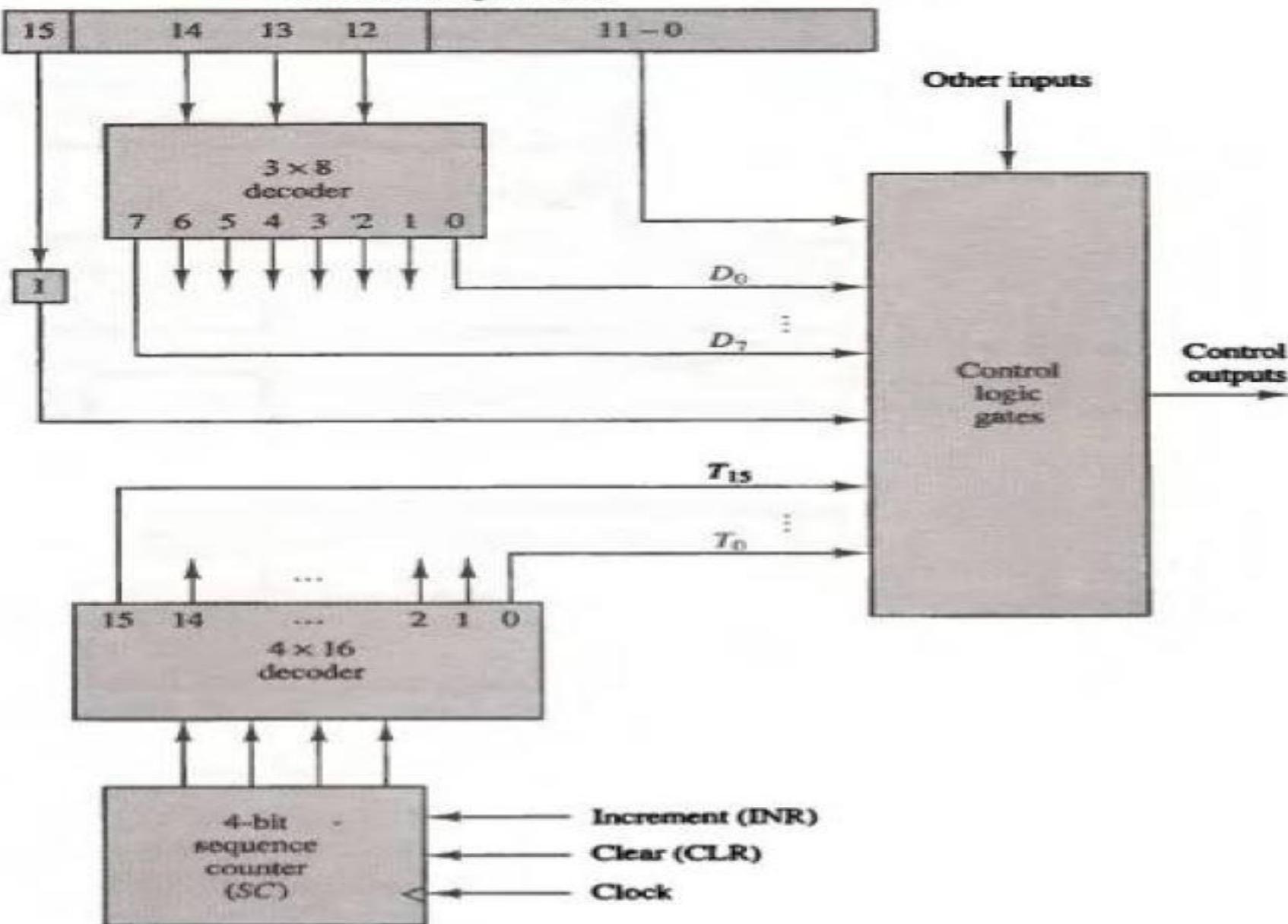
	XY	$X'Y'$	$X'Y$	XY	XY'
q		00	01	11	10
q'	00				
q	01				

	XY	$X'Y'$	$X'Y$	XY	XY'
q		00	01	11	10
q'	00				
q	01				

Basics of counters

- A counter is a sequential circuit that goes through a predetermined sequence of binary states upon the application of input pulses.
- The gates in the counter are connected in such a way as to produce the prescribed sequence of states.
- A counter that follows the binary number sequence is called a binary counter.
- An n - bit binary counter consists of n flip-flops and can count in binary from 0 through $2^n - 1$.
- Counters are available in two categories: **synchronous counters** and **Ripple counters (asynchronous)**.

Instruction register (IR)





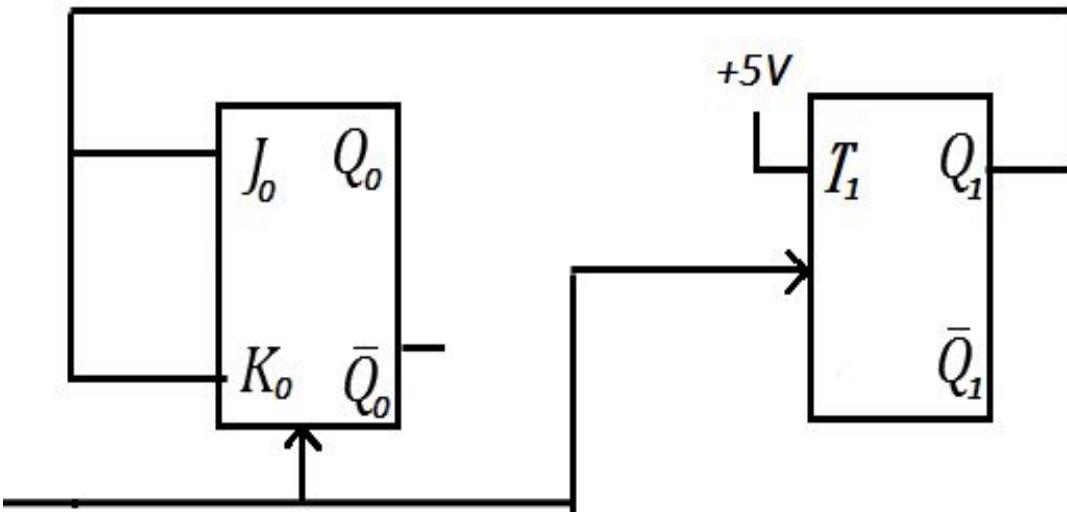
Q Find out the counting sequence for the counter below?

The characteristics equation for JK Flip Flop is $Q_{n+1} =$

$$Q_{0N} =$$

The characteristics equation for T Flip Flop is $Q_{n+1} =$

$$Q_{1N} =$$

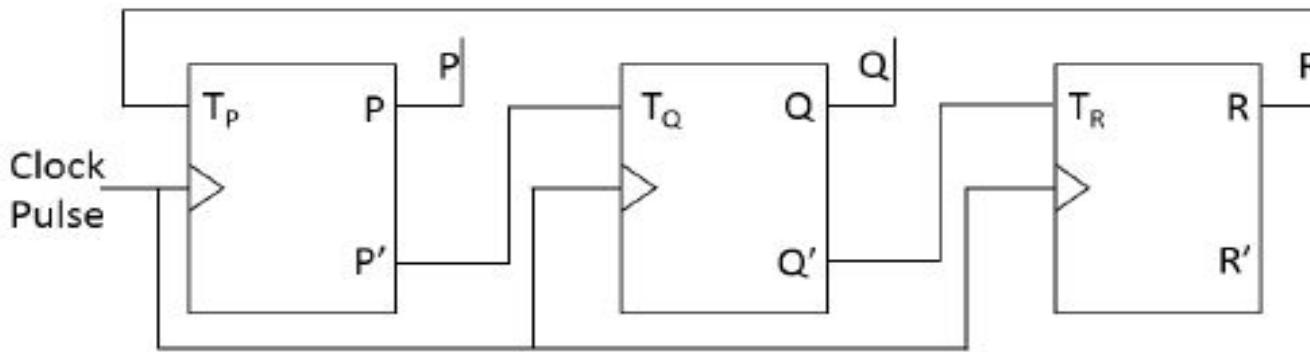


Present State		Next State	
Q _{1p}	Q _{0p}	Q _{1N}	Q _{0N}
0	0		
0	1		
1	0		
1	1		

Q Consider a 3-bit counter, designed using T flip-flops, as shown below: Assuming the initial state of the counter given by PQR as 000, what are the next three states?

(GATE 2021) (2 MARKS)

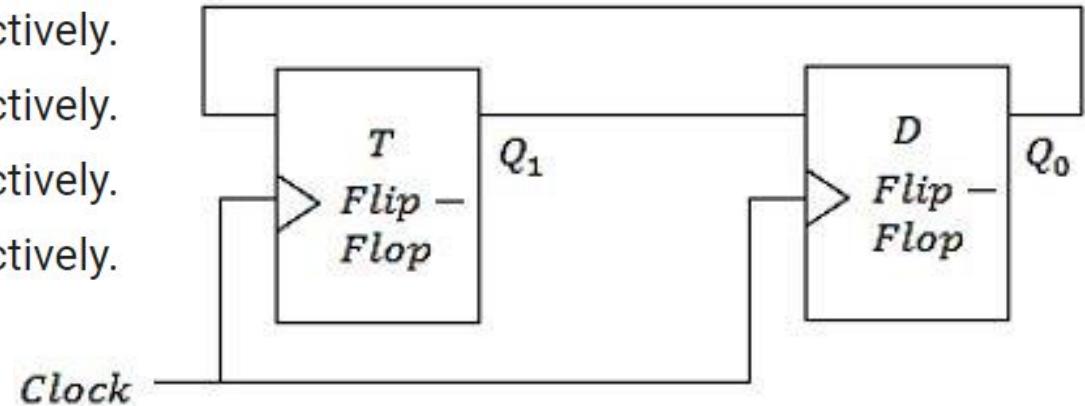
- (A) 011, 101, 000
- (B) 001, 010, 111
- (C) 011, 101, 111
- (D) 001, 010, 000



Present State			Next State		
Q_{2p}	Q_{1p}	Q_{0p}	Q_{2N}	Q_{1N}	Q_{0N}
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Q Consider a combination of T and D flip-flops connected as shown below. The output of the D flip-flop is connected to the input of the T flip-flop and the output of the T flip-flop is connected to the input of the D flip-flop. Initially, both Q_0 and Q_1 , are set to 1 (before the 1st clock cycle). The outputs (GATE-2017) (2 Marks)

- A. $Q_1 Q_0$ after the 3rd cycle are 11 and after the 4th cycle are 00 respectively.
- B. $Q_1 Q_0$ after the 3rd cycle are 11 and after the 4th cycle are 01 respectively.
- C. $Q_1 Q_0$ after the 3rd cycle are 00 and after the 4th cycle are 11 respectively.
- D. $Q_1 Q_0$ after the 3rd cycle are 01 and after the 4th cycle are 01 respectively.



Present State		Next State	
Q_{1p}	Q_{0p}	Q_{1N}	Q_{0N}
0	0		
0	1		
1	0		
1	1		

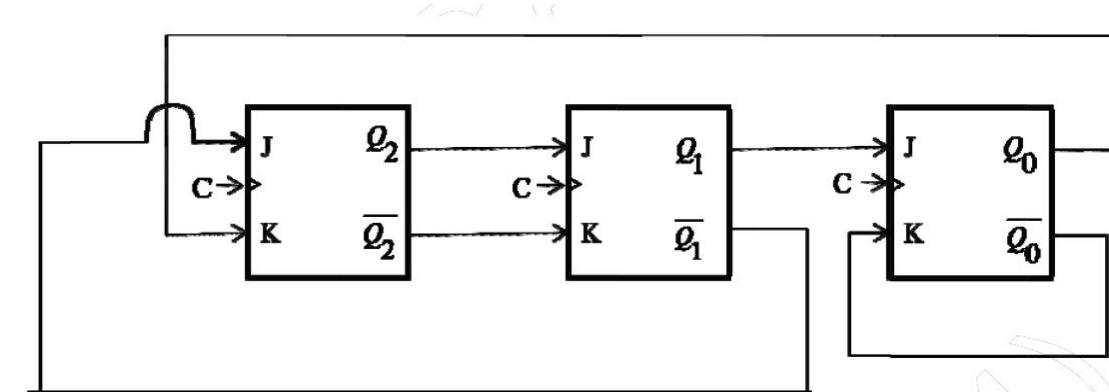
Q The above sequential circuit is built using JK flip-flops is initialized with $Q_2Q_1Q_0 = 000$. The state sequence for this circuit for the next 3 clock cycle is (GATE-2014) (2 Marks)

(A) 001, 010, 011

(B) 111, 110, 101

(C) 100, 110, 111

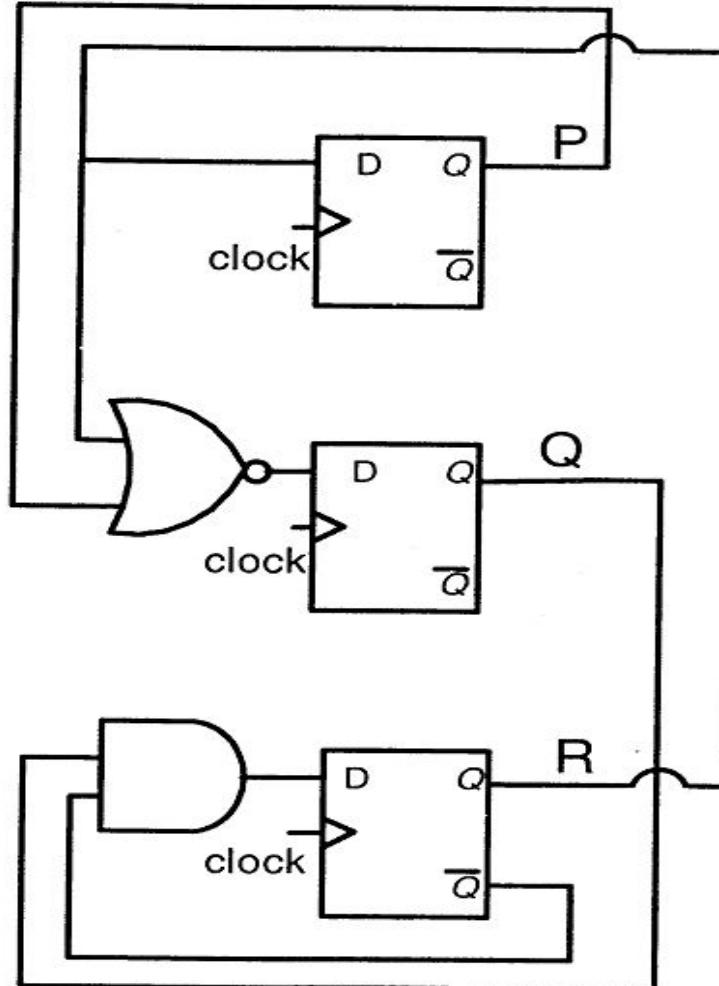
(D) 100, 011, 001



Present State			Next State		
Q_{2p}	Q_{1p}	Q_{0p}	Q_{2N}	Q_{1N}	Q_{0N}
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Q Consider the following circuit involving three D-type flip-flops used in a certain type of counter configuration. If at some instance prior to the occurrence of the clock edge, P, Q and R have a value 0, 1 and 0 respectively, what shall be the value of PQR after the clock edge? (GATE-2011) (2 Marks)

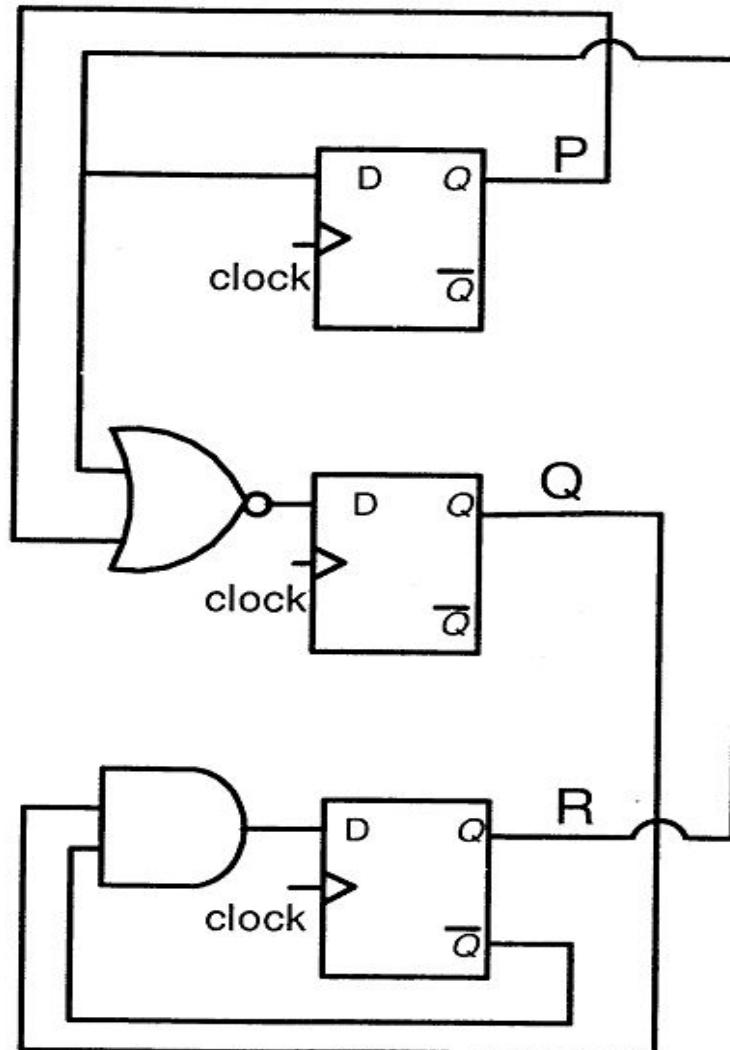
- (A) 000 (B) 001 (C) 010 (D) 011



Present State			Next State		
Q_{2p}	Q_{1p}	Q_{0p}	Q_{2N}	Q_{1N}	Q_{0N}
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Q Consider the data given in previous question. If all the flip-flops were reset to 0 at power on, what is the total number of distinct outputs (states) represented by PQR generated by the counter? (GATE-2011) (2 Marks)

- (A) 3 (B) 4 (C) 5 (D) 6



Q The minimum number of D flip-flops needed to design a mod-258 counter is **(GATE-2011) (1 Marks)**

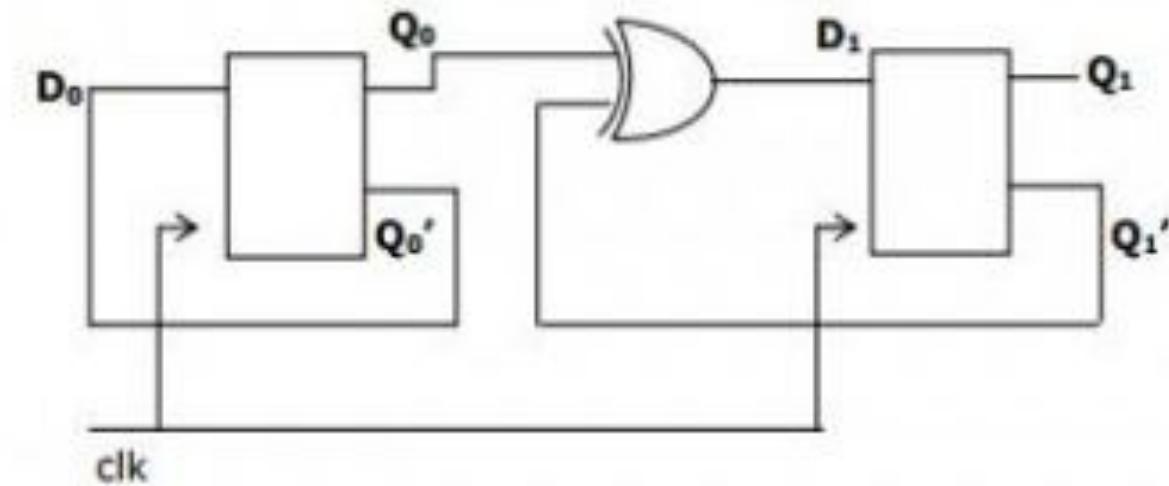
Q Consider the following circuit. The flip-flops are positive edge triggered D FFs. Each state is designated as a two-bit string Q_0Q_1 . Let the initial state be 00. The state transition sequence is: (GATE - 2005) (2 Marks)

a) 00, 11, 01

b) 00, 11

c) 00, 10, 01, 11

d) 00, 11, 01, 10



Present State		Next State	
Q_{0p}	Q_{1p}	Q_{0N}	Q_{1N}
0	0		
0	1		
1	0		
1	1		

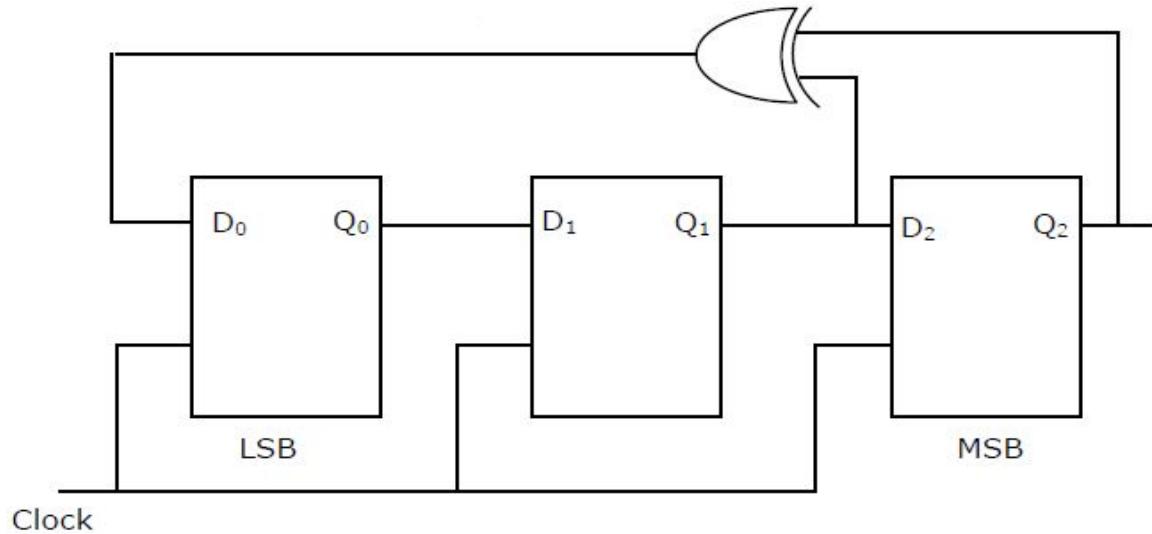
Q Consider the circuit given below with initial state $Q_0 = 1, Q_1 = Q_2 = 0$. The state of the circuit is given by the value $4Q_2 + 2Q_1 + Q_0$, Which one of the following is the correct state sequence of the circuit? (GATE-2001) (2 Marks)

(A) 1,3,4,6,7,5,2

(B) 1,2,5,3,7,6,4

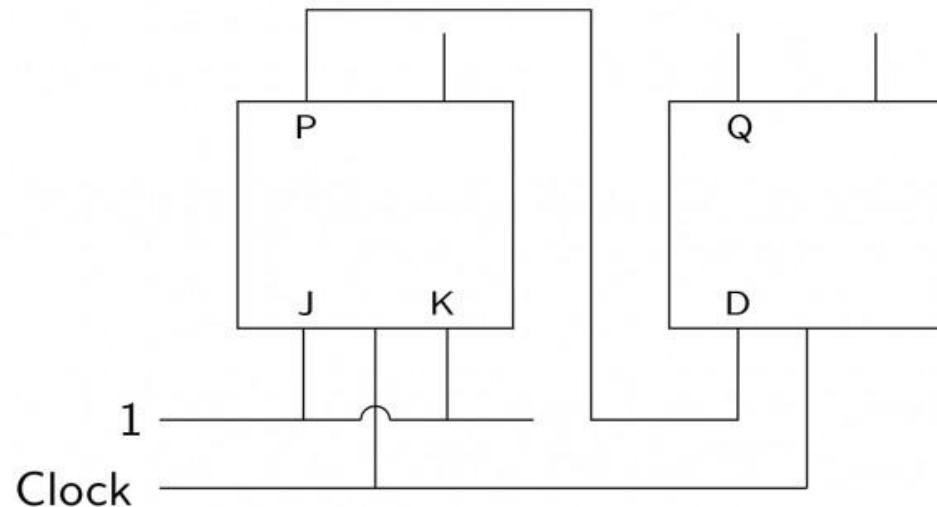
(C) 1,2,7,3,5,6,4

(D) 1,6,5,7,2,3,4



Present State			Next State		
Q _{2p}	Q _{1p}	Q _{0p}	Q _{2N}	Q _{1N}	Q _{0N}
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Q The following arrangement of master-slave flip flops has the initial state of P, Q as 0, 1 (respectively). After three clock cycles the output state P, Q is (respectively), **(GATE-2000) (2 Marks)**



Present State		Next State	
Q_{1p}	Q_{0p}	Q_{1N}	Q_{0N}
0	0		
0	1		
1	0		
1	1		

(A) 1, 0

(B) 1, 1

(C) 0, 0

(D) 0,1

Q The minimum number of JK flip-flops required to construct a synchronous counter with the count sequence (0, 0, 1, 1, 2, 2, 3, 3, 0, 0,...) is _____ (GATE-2015) (2 Marks)

- (A) 0
- (B) 1
- (C) 2
- (D) 3

Q We want to design a synchronous counter that counts the sequence 0-1-0-2-0-3 and then repeats. The minimum number of J-K flip-flops required to implement this counter is **(GATE - 2016)**

Q A positive edge-triggered D flip-flop is connected to a positive edge-triggered JK flipflop as follows. The Q output of the D flip-flop is connected to both the J and K inputs of the JK flip-flop, while the Q output of the JK flip-flop is connected to the input of the D flip-flop. Initially, the output of the D flip-flop is set to logic one and the output of the JK flip-flop is cleared. Which one of the following is the bit sequence (including the initial state) generated at the Q output of the JK flip-flop when the flip-flops are connected to a free-running common clock? Assume that $J = K = 1$ is the toggle mode and $J = K = 0$ is the state-holding mode of the JK flip-flop. Both the flip-flops have non-zero propagation delays. **(GATE-2015) (2 Marks)**

- (A) 0110110...** **(B) 0100100...** **(C) 011101110...** **(D) 011001100...**

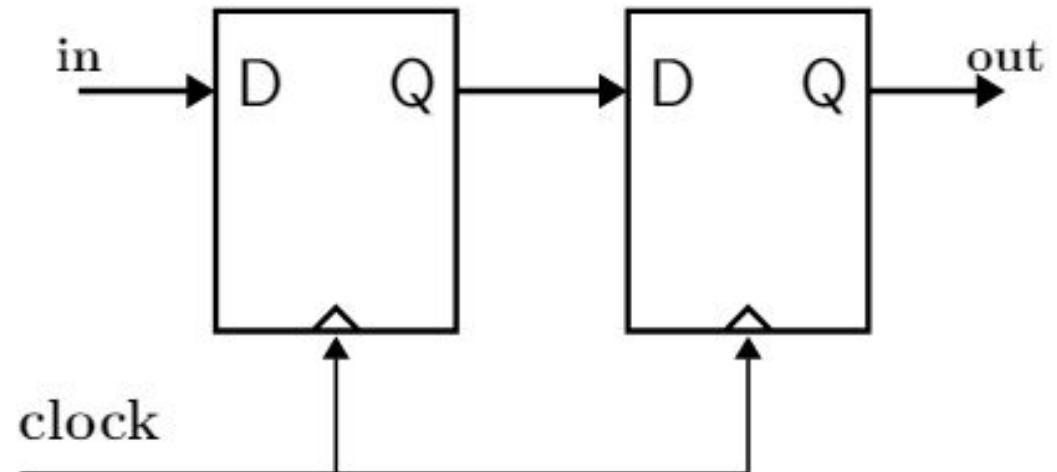
Present State		Next State	
Q_{0p}	Q_{1p}	Q_{0N}	Q_{1N}
0	0		
0	1		
1	0		
1	1		

Q How many pulses are needed to change the contents of an 8-bit up counter from 10101100 to 00100111 (rightmost bit is the LSB)? **(GATE-2005) (2 Marks)**

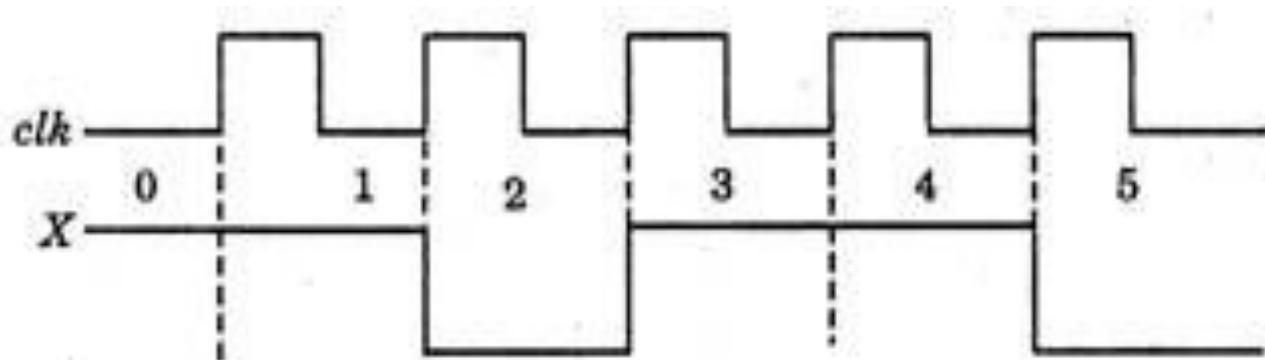
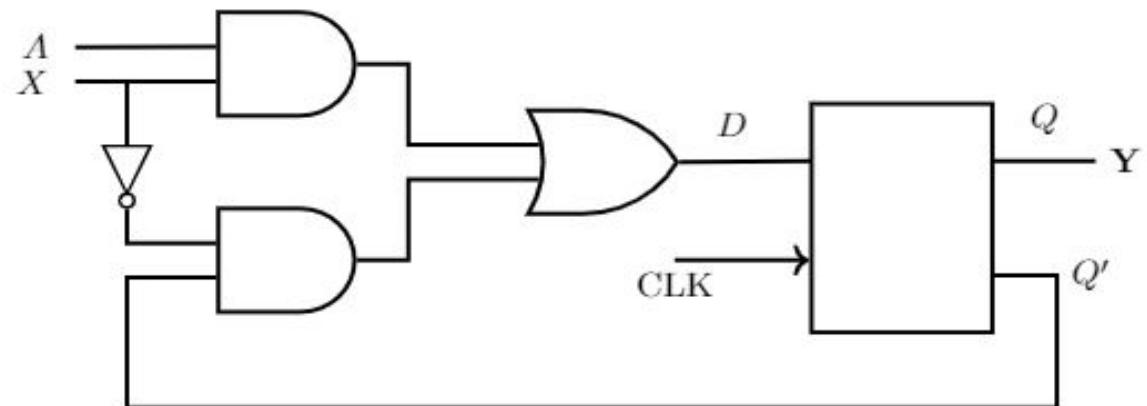
- (A)** 134 **(B)** 133 **(C)** 124 **(D)** 123

Q Consider the sequential circuit shown in the figure, where both flip-flops used are positive edge-triggered D flip-flops. The number of states in the state transition diagram of this circuit that have a transition back to the same state on some value of “in” is _____.(GATE-2018) (2 Marks)

Present State			Next State		
Q_{2p}	Q_{1p}	Q_{0p}	Q_{2N}	Q_{1N}	Q_{0N}
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			



Q Consider the following circuit involving a positive edge triggered D FF. (GATE-2005) (2 Marks)



Consider the following timing diagram. Let A_i represent the logic level on the line A in the i -th clock period.

Let A' represent the complement of A . The correct output sequence on Y over the clock periods 1 through 5 is

- (A) $A_0 A_1 A_1' A_3 A_4$
- (B) $A_0 A_1 A_2' A_3 A_4$
- (C) $A_1 A_2 A_2' A_3 A_4$
- (D) $A_1 A_2' A_3 A_4 A_5'$

Q Design a synchronous counter for sequence: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0$, using T flip flop.

Present State		Next State	
Q_{1p}	Q_{0p}	Q_{1N}	Q_{0N}
0	0		
0	1		
1	0		
1	1		

Q Example: Design a synchronous counter for sequence: $0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 0$, using D flip flop.

Present State		Next State	
Q_{1p}	Q_{0p}	Q_{1N}	Q_{0N}
0	0		
0	1		
1	0		
1	1		

Q Design synchronous counter for sequence: $0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 0$, using T flip-flop.

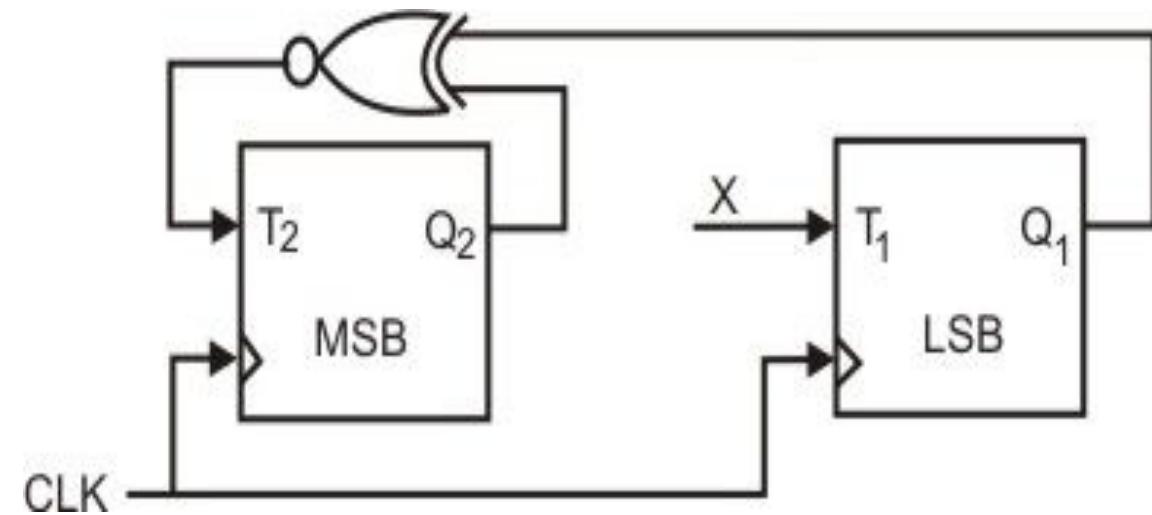
Present State			Next State		
Q_{2p}	Q_{1p}	Q_{0p}	Q_{2N}	Q_{1N}	Q_{0N}
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Q Consider the partial implementation of a 2-bit counter using T flip-flops following the sequence 0-2-3-1-0, as shown below (**GATE-2004**) (2 Marks)

To complete the circuit, the input X should be

- (A) Q_2' (B) $Q_2 + Q_1$ (C) $(Q_1 \oplus Q_2)'$ (D) $Q_1 \oplus Q_2$

Present State		Next State	
Q_{1p}	Q_{0p}	Q_{1N}	Q_{0N}
0	0		
0	1		
1	0		
1	1		



Q The next state table of a 2-bit saturating up-counter is given below. (GATE-2017) (2 Marks)
 The counter is built as synchronous sequential circuit using T flip-flops. The value for T_1 and T_0 are

$$\begin{array}{ll} \text{(A)} \quad T_1 = Q_0 Q_1 & T_0 = Q'_0 Q'_1 \\ \text{(C)} \quad T_1 = Q_1 + Q_0 & T_0 = Q'_1 + Q'_0 \end{array}$$

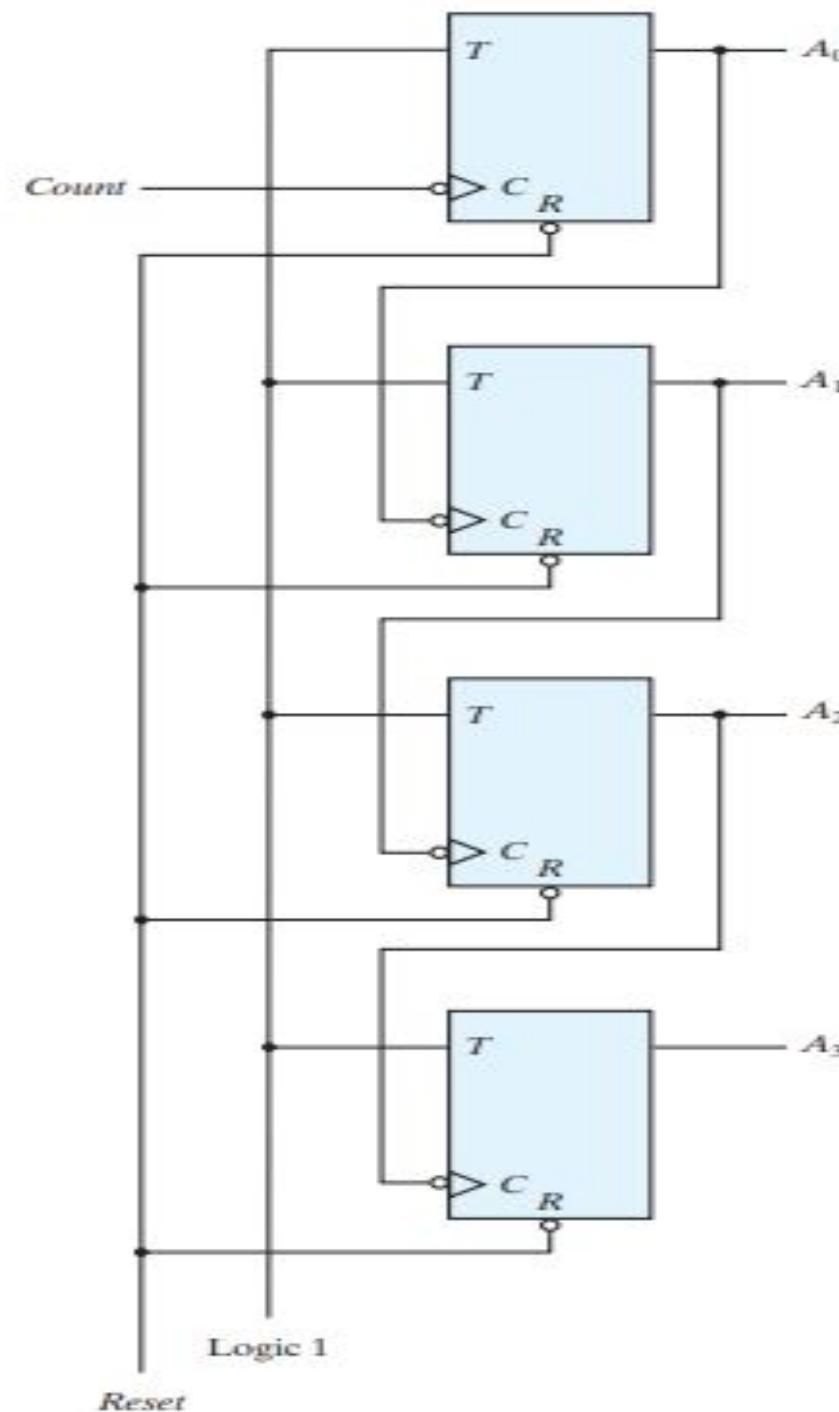
$$\begin{array}{ll} \text{(B)} \quad T_1 = Q'_1 Q_0 & T_0 = Q'_1 + Q'_0 \\ \text{(D)} \quad T_1 = Q'_1 Q_0 & T_0 = Q_1 + Q_0 \end{array}$$

Present State		Next State	
Q_{1p}	Q_{0p}	Q_{1N}	Q_{0N}
0	0		
0	1		
1	0		
1	1		

Q_1	Q_0	Q_1^+	Q_0^+
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	1

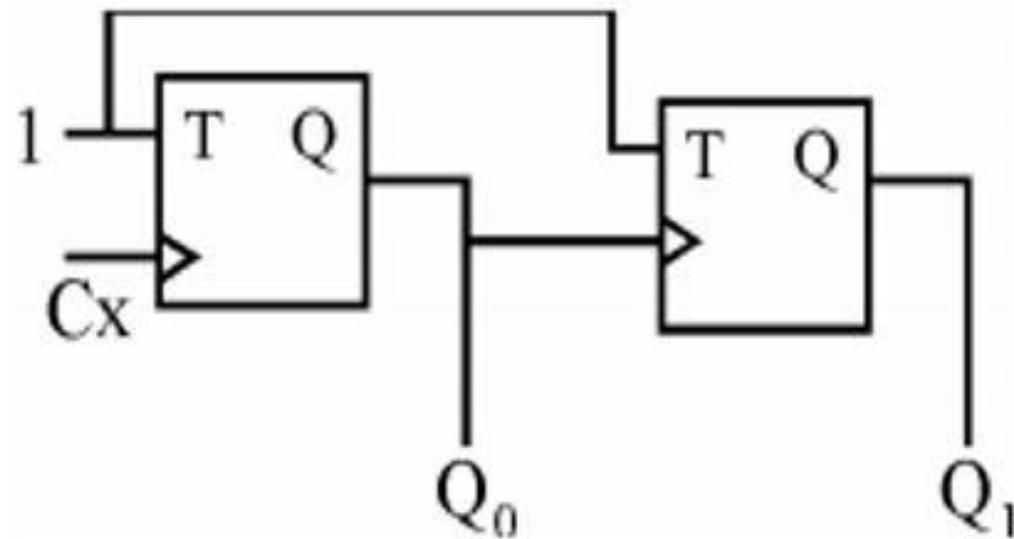
RIPPLE COUNTERS

1. A binary ripple counter consists of a series connection of complementing flip-flops, with the output of each flip-flop connected to the input of the next higher order flip-flop.
2. The flip-flop holding the least significant bit receives the incoming count pulses.

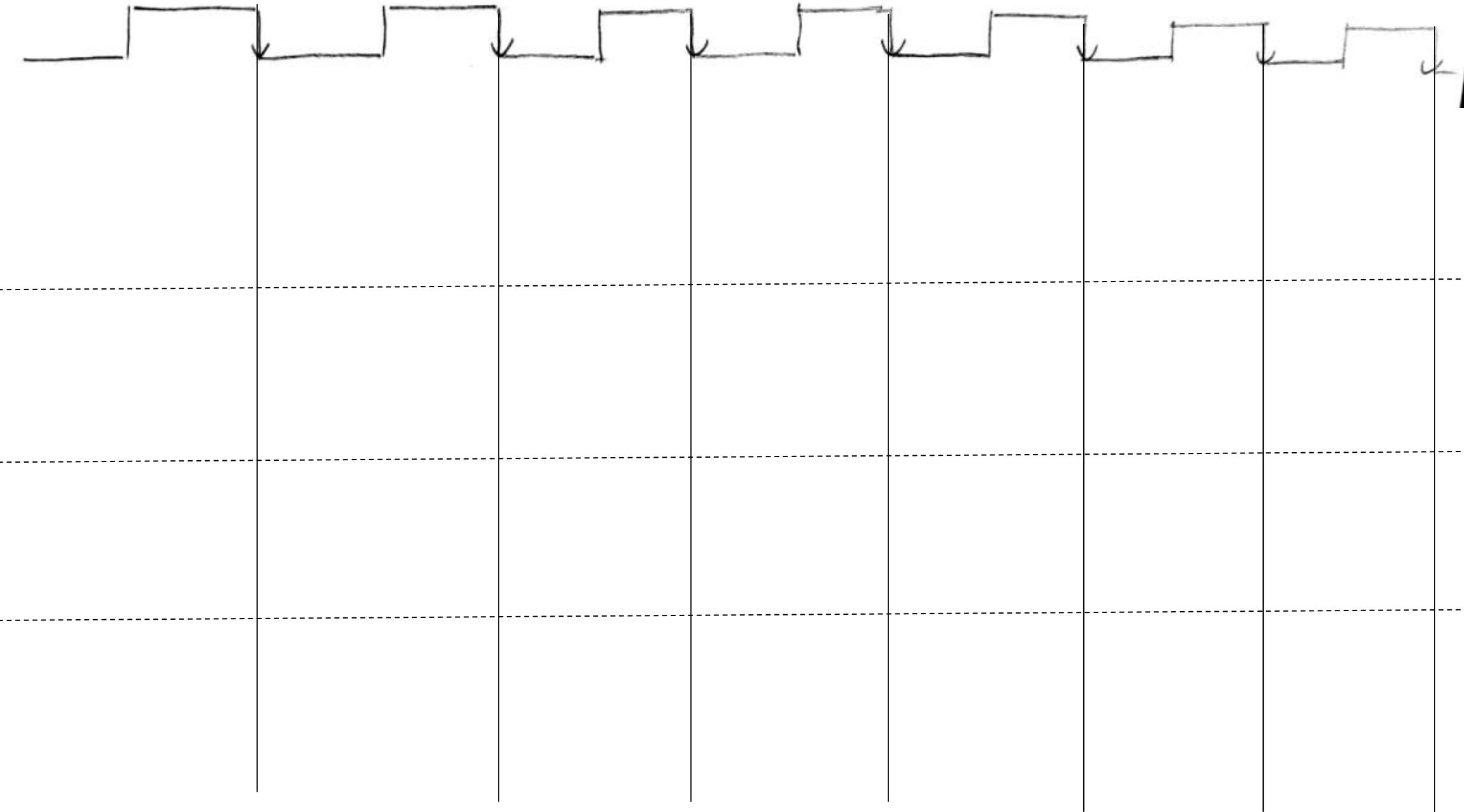


Q In the sequential circuit shown below, if the initial value of the output $Q_1 Q_0$ is 00, what are the next four values of $Q_1 Q_0$? (GATE-2010) (2 Marks)

- (A) 11, 10, 01, 00 (B) 10, 11, 01, 00 (C) 10, 00, 01, 11 (D) 11, 10, 00, 01



Present State		Next State	
Q_{1p}	Q_{0p}	Q_{1N}	Q_{0N}
0	0		
0	1		
1	0		
1	1		



Nature of Clock	Nature of Feedback	Nature of counting
+ve	Q_n'	UP Counting
+ve	Q_n	Down Counting
-ve	Q_n	UP Counting
-ve	Q_n'	Down Counting

Q. In a 4-bit ripple counter, if the period of the waveform at the last flip-flop is 64 microseconds,
then the frequency of the ripple counter in kHz is _____.
(Answer in integer)(GATE 2025)

Asynchronous counters

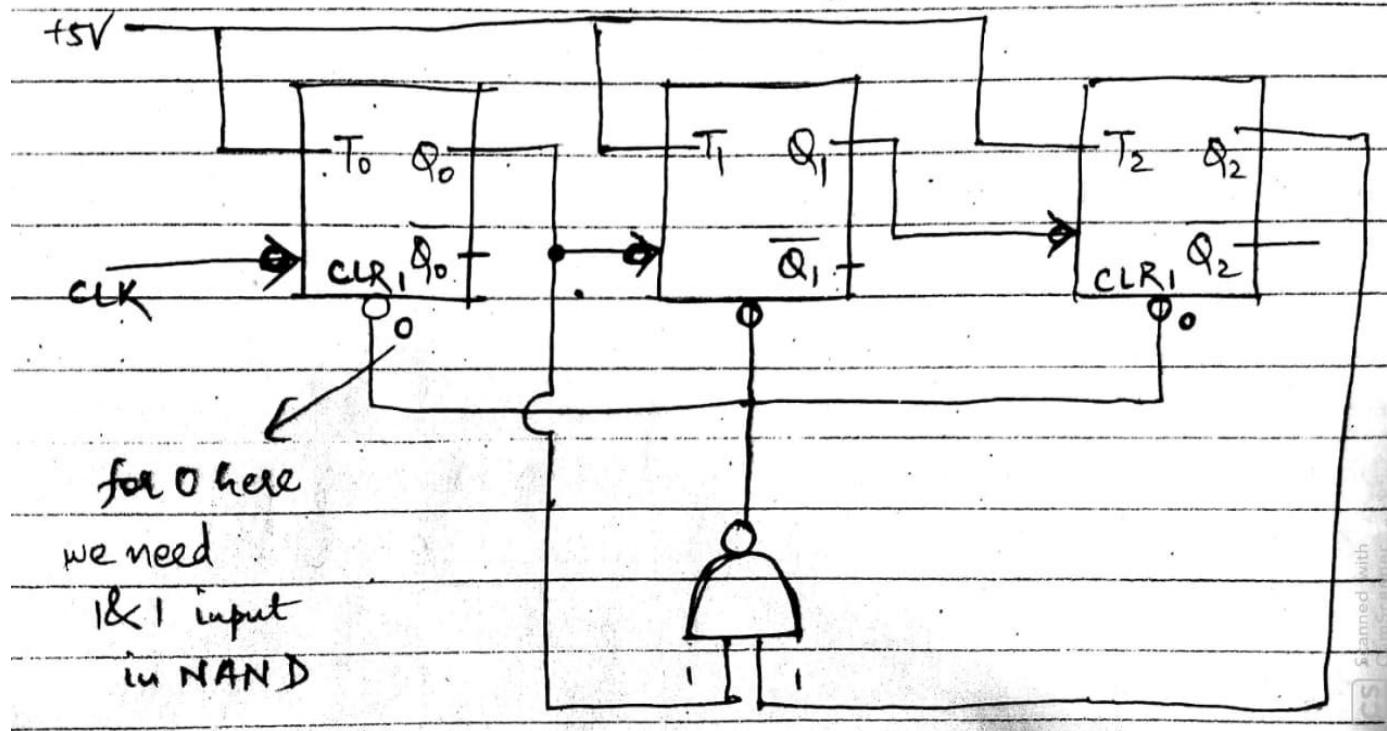
SYNCHRONOUS COUNTER	ASYNCHRONOUS COUNTER
In synchronous counter, all flip flops are triggered with same clock simultaneously.	In asynchronous counter, different flip flops are triggered with different clock, not simultaneously.
Synchronous Counter is faster than asynchronous counter in operation. $T_{\text{delay}} = T_{\text{FF}} + T_{\text{cc}}$	Asynchronous Counter is slower than synchronous counter in operation. $T_{\text{delay}} = n \times T_{\text{FF}} + T_{\text{cc}}$
Synchronous Counter is also called Parallel Counter.	Asynchronous Counter is also called Serial Counter.
Synchronous Counter designing as well implementation are complex due to increasing the number of states.	Asynchronous Counter designing as well as implementation is very easy.
Synchronous Counter will operate in any desired count sequence.	Asynchronous Counter will operate only in fixed count sequence (UP/DOWN).
Synchronous Counter examples are: Ring counter, Johnson counter.	Asynchronous Counter examples are: Ripple UP counter, Ripple DOWN counter.
A synchronous sequential circuit is a system whose behavior can be defined from the knowledge of its signals at discrete instants of time.	The behavior of an asynchronous sequential circuit depends upon the input signals at any instant of time and the order in which the inputs change.

- **Self-Starting Counter**: - A counter is said to be self-starting if it provides the counting sequence irrespective of initial state.
- **Free running counter**: - a counter is said to be free running If it maintains all possible states in the counting sequence.
 - 0 □ 3 □ 2 □ 1 □ 0
 - 0 □ 1 □ 3 □ 2 □ 1
 - 0 □ 3 □ 0, 1 □ 2 □ 3
 - 0 □ 1 □ 2 □ 0, 3 □ 3
 - 0 □ 1 □ 2 □ 3 □ 4 □ 5 □ 6 □ 2, 7 □ 6
 - 1 □ 3 □ 4 □ 5 □ 6 □ 2 □ 1, 7 □ 6
- **Conclusion**: - if a counter is free running, it is also self-starting, but vice-versa not required to be true.

Restricted mod counter

Q Consider a restricted mod counter and find what sequence it is counting?

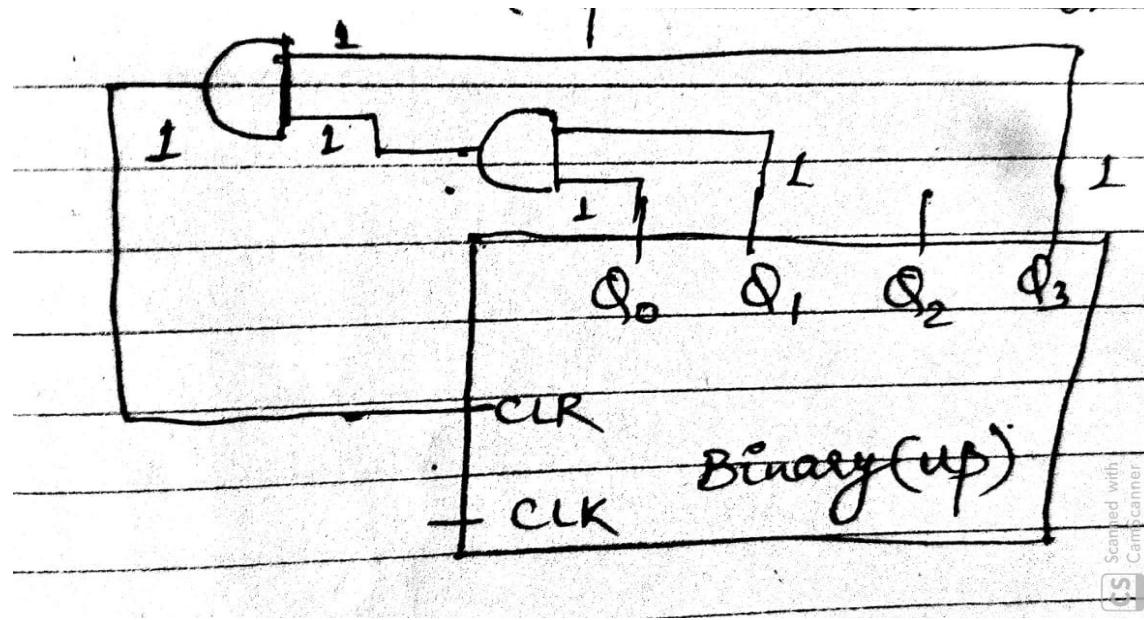
Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1



Restricted mod counter

Q Consider a restricted mod counter and find what sequence it is counting?

Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1



Q The control signal functions of a 4-bit binary counter are given below (where X is “don’t care”)

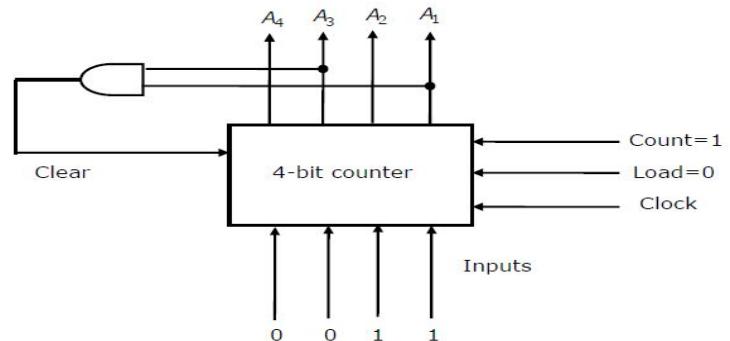
The counter is connected as follows: **(GATE-2007) (2 Marks)**

Assume that the counter and gate delays are negligible. If the counter starts at 0, then it cycles through the following sequence:

- (A) 0, 3, 4
- (B) 0, 3, 4, 5
- (C) 0, 1, 2, 3, 4
- (D) 0, 1, 2, 3, 4, 5

Clear	Clock	Load	Count	Function
1	X	X	X	Clear to 0
0	X	0	0	No change
0	↑	1	X	Load input
0	↑	0	1	Count next

The counter is connected as follows:



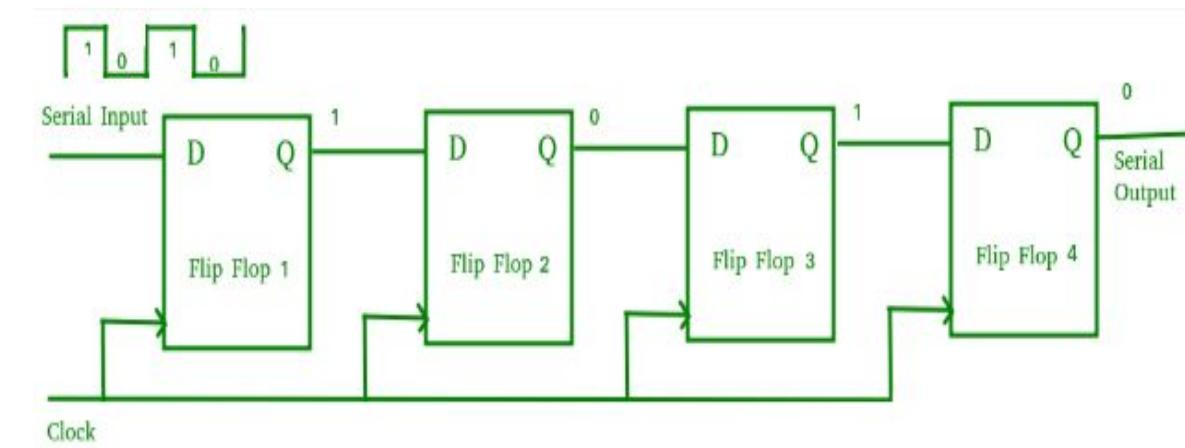
Registers

- Registers are basically storing devices which are also designed using basic element called flip-flop.
- D-flip-flop are most popular choice for register because they don't perform any functionality and output is simply based on current input so, they act as a buffer.
- Apart from storing registers sometimes also be used in performing basic mathematical operation like multiply by 2 by left shift and divide by 2 by right shift.
- There are four different modes in which registers operate.
 - Serial In-Serial Out (SISO)
 - Serial In-Parallel Out (SIPO)
 - Parallel In-Serial Out (PISO)
 - Parallel In- Parallel Out (PIPO)

Serial In-Serial Out (SISO)

- The shift register, which allows serial input (one bit after the other through a single data line) and produces a serial output is known as Serial-In Serial-Out shift register.
- Since there is only one output, the data leaves the shift register one bit at a time in a serial pattern, thus the name Serial-In Serial-Out Shift Register.

Sequence	Input	Q_2	Q_1	Q_0	Output

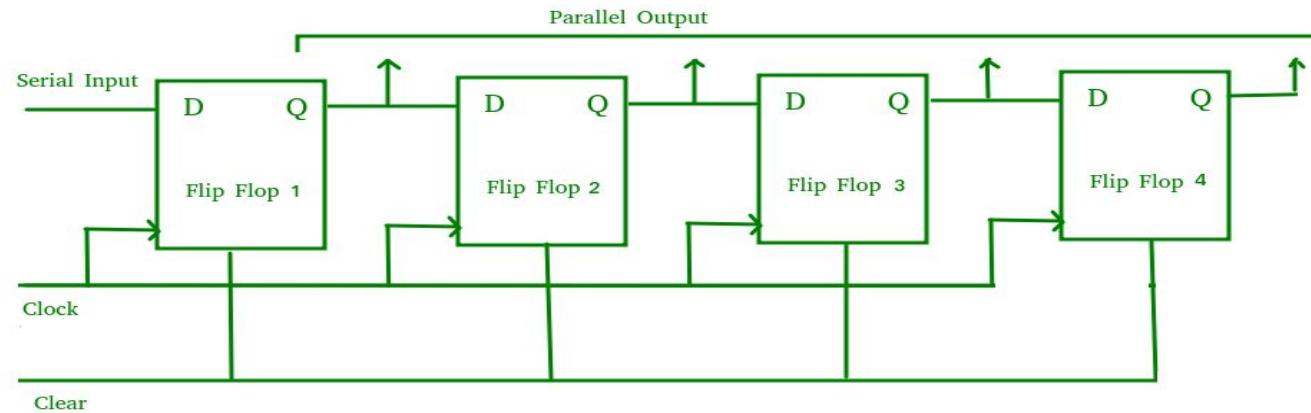


- The main use of a SISO is to act as a delay element.
- In SISO registers to provide n bit data serially in it requires n clock pulse and to provide serial output it requires n - 1 clock pulses.

	No of clock (write)	No of clock (Read)	total
SISO			
SIPO			
PISO			
PIPO			

Serial-In Parallel-Out shift Register (SIPO)

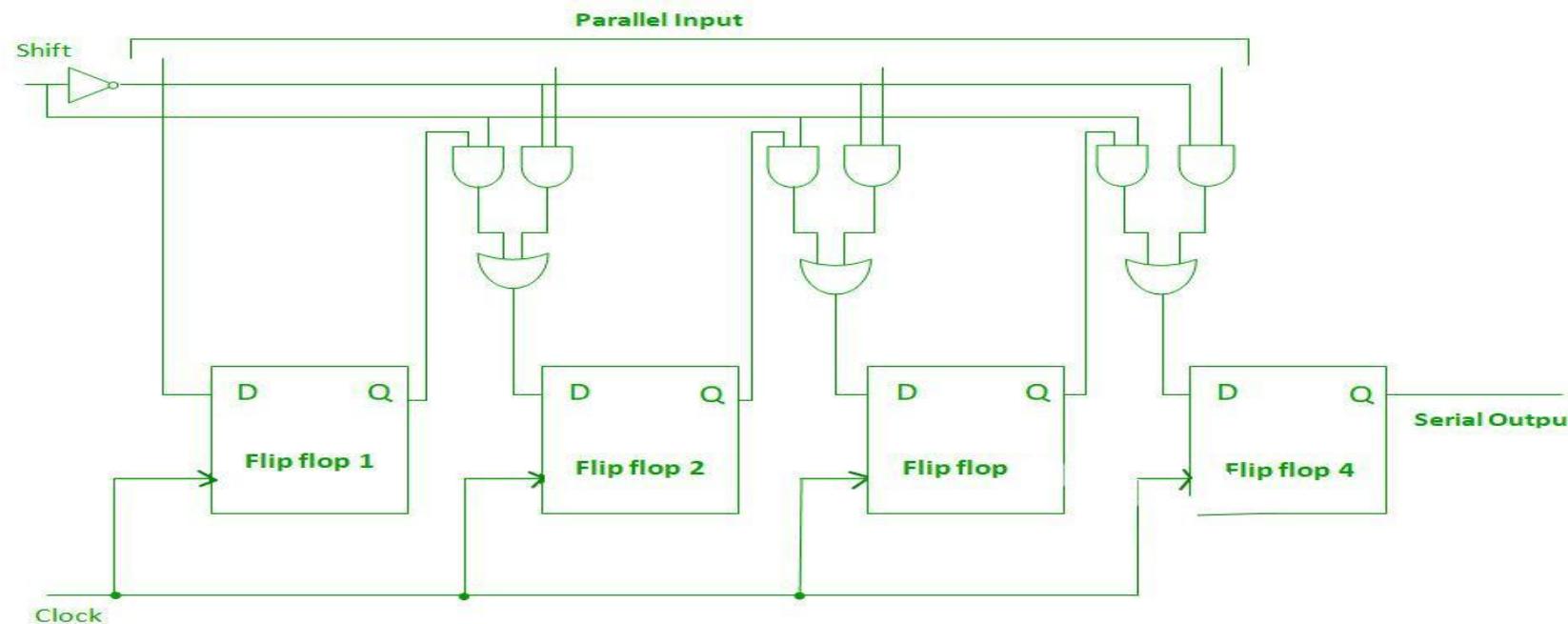
- The shift register, which allows serial input (one bit after the other through a single data line) and produces a parallel output is known as Serial-In Parallel-Out shift register.
- They are used in communication lines where demultiplexing of a data line into several parallel lines is required because the main use of the SIPO register is to convert serial data into parallel data.



	No of clock (write)	No of clock (Read)	total
SISO	n	n-1	2n-1
SIPO			
PISO			
PIPO			

Parallel-In Serial-Out Shift Register (PISO)

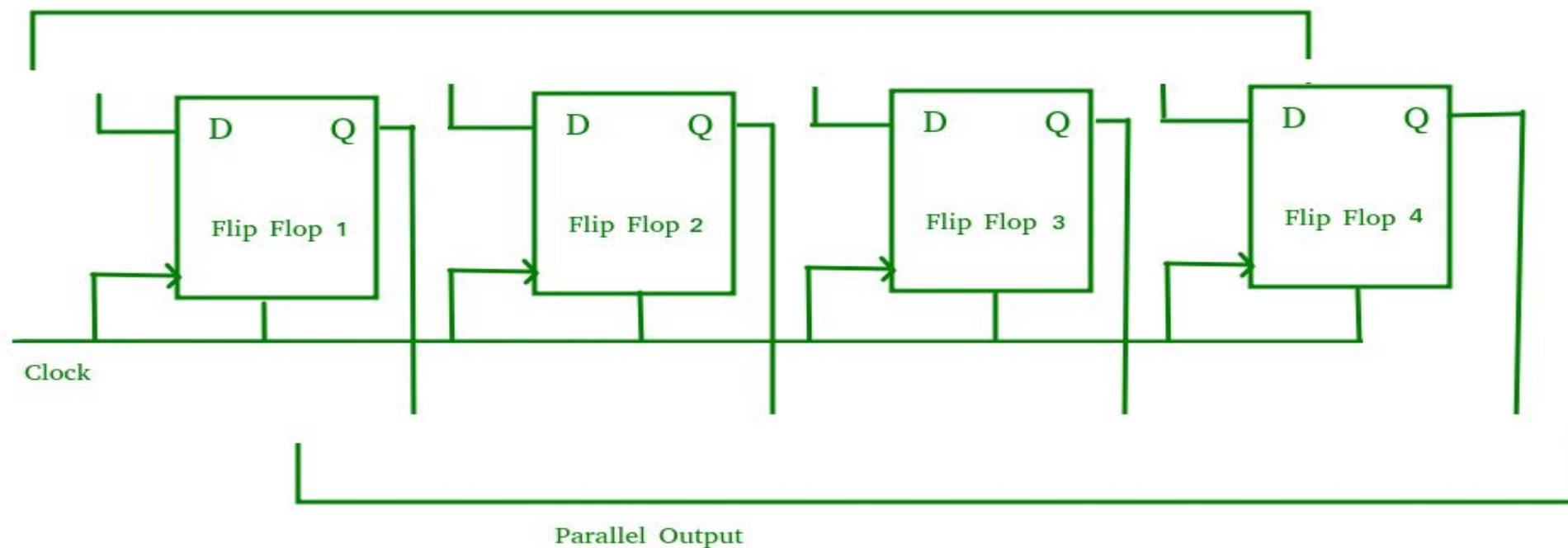
- The shift register, which allows parallel input (data is given separately to each flip flop and in a simultaneous manner) and produces a serial output is known as Parallel-In Serial-Out shift register.
- The circuit consists of four D flip-flops which are connected.
- The clock input is directly connected to all the flip flops but the input data is connected individually to each flip flop through a multiplexer at the input of every flip flop.
- The output of the previous flip flop and parallel data input are connected to the input of the MUX and the output of MUX is connected to the next flip flop.
- A Parallel in Serial out (PISO) shift register us used to convert parallel data to serial data.



	No of clock (write)	No of clock (Read)	total
SISO	n	n-1	$2n-1$
SIPO	n	0	n
PISO			
PIPO			

Parallel-In Parallel-Out Shift Register (PIPO)

- The shift register, which allows parallel input (data is given separately to each flip flop and in a simultaneous manner) and also produces a parallel output is known as Parallel-In parallel-Out shift register.
- The circuit consists of four D flip-flops which are connected.
- In this type of register, there are no interconnections between the individual flip-flops since no serial shifting of the data is required.
- Data is given as input separately for each flip flop and in the same way, output also collected individually from each flip flop.
- A Parallel in Parallel out (PIPO) shift register is used as a temporary storage device and like SISO Shift register it acts as a delay element.
- For parallel input it requires 1 clock pulse and for parallel output it requires 0 clock.

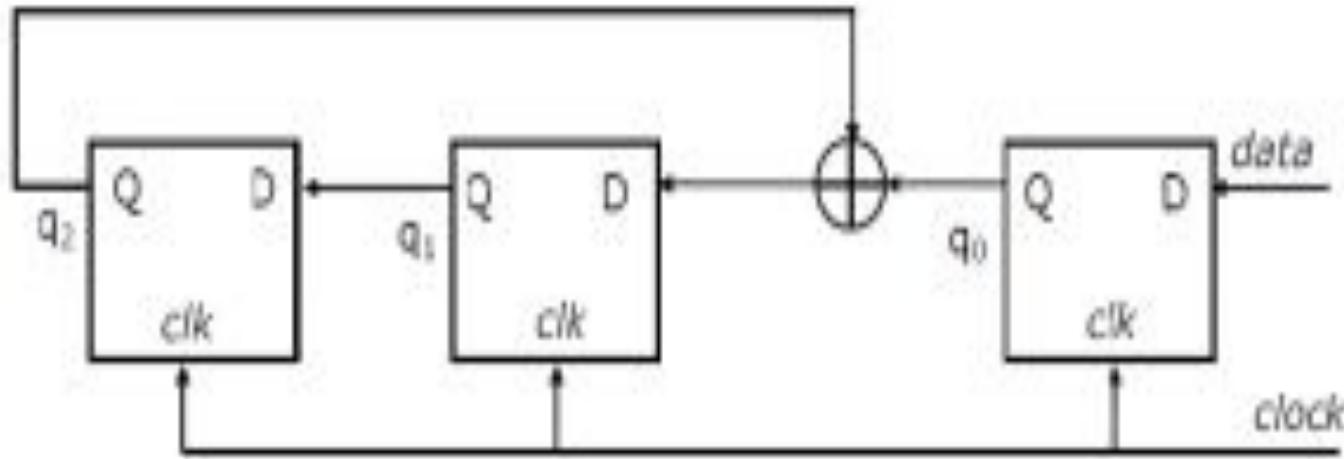


	No of clock (write)	No of clock (Read)	total
SISO	n	n-1	$2n-1$
SIPO	n	0	n
PISO	1	n-1	n
PIPO			

Q Consider a 32-bit shift register which uses a clock of 1 GHz. If register is operated in SISO mode, find total time required to perform loading and reading?

- a) 1 ns
- b) 31 ns
- c) 32 ns
- d) 63 ns

Q Consider the circuit in the diagram. The \oplus operator represents Ex-OR. The D flipflops are initialized to zeroes (cleared). **(GATE-2006) (2 Marks)**



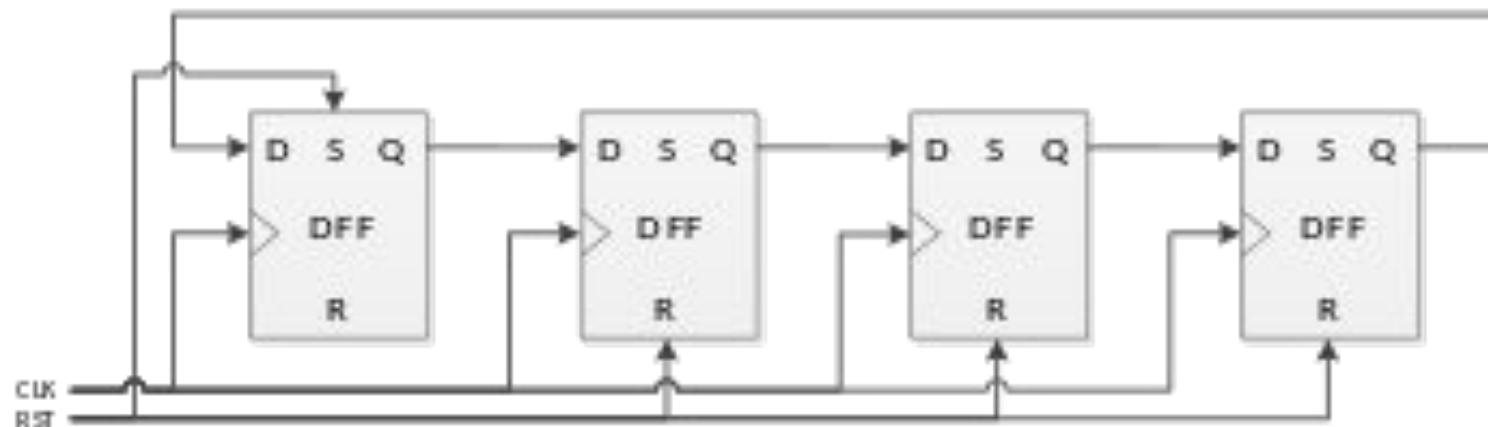
The following data: 100110000 is supplied to the “data” terminal in nine clock cycles. After that the values of $q_2 q_1 q_0$ are:

- (A) 000
- (B) 001
- (C) 010
- (D) 101

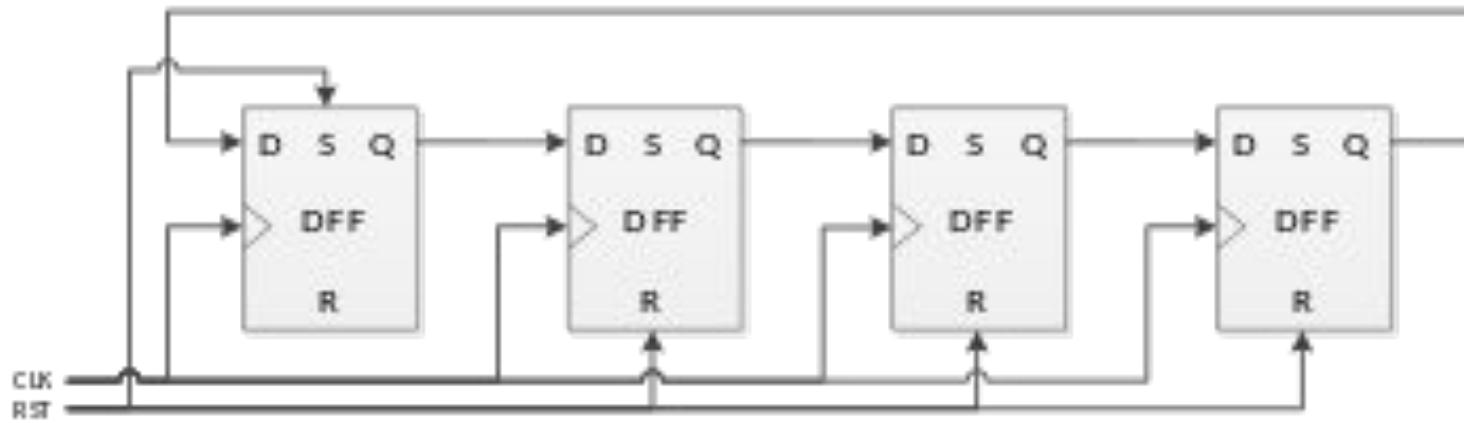
Ring Counter/ Straight Ring Counter

4-Bit Ring Counter

- A ring counter is a circular shift register with only one flip-flop being set at any particular time; all others are cleared.
- The single bit is shifted from one flip-flop to the next to produce the sequence of timing signals.
- Output of the last flip-flop is connected to the input of the first flip-flop in case of ring counter.
- ***No. of states in Ring counter = No. of flip-flop used***



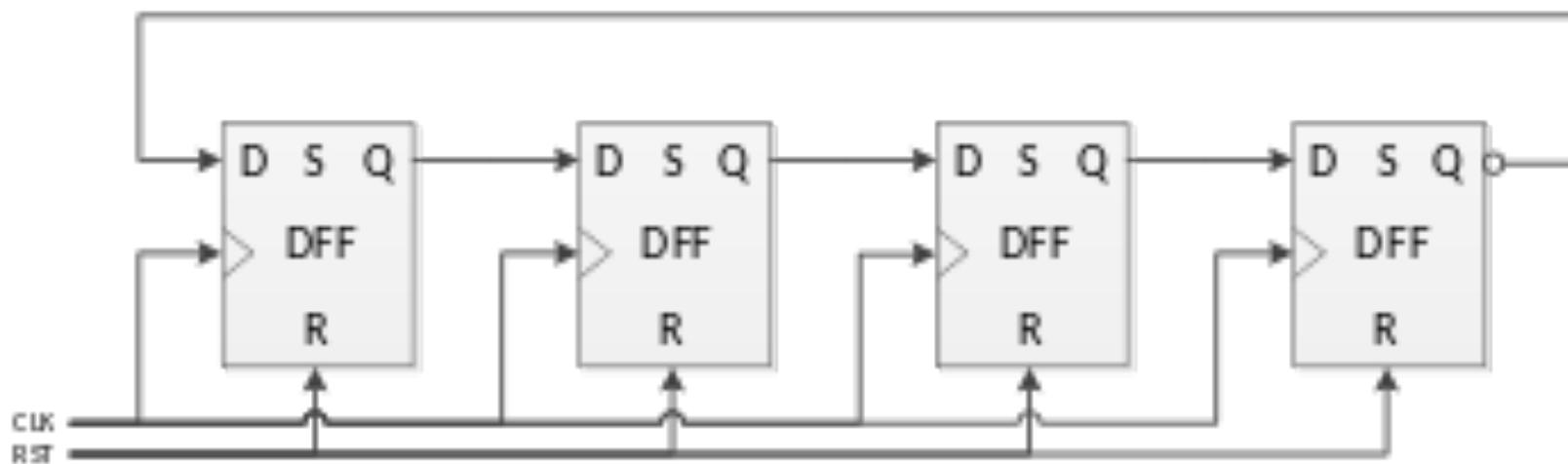
- Number of unused states in Ring Counter is $2^n - n$

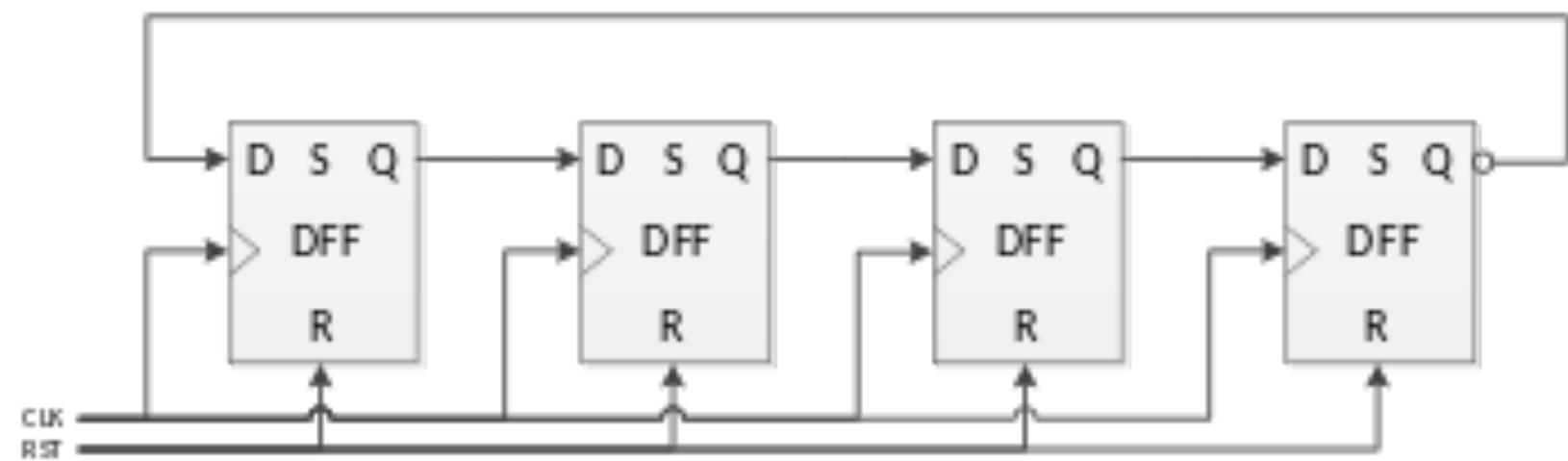


CLK	Q_3	Q_2	Q_1	Q_0
0				
1				
2				
3				
0				

Johnson Counter/ Twisted ring counter/ Switch-tail ring counter/ Walking ring counter

- A k -bit ring counter circulates a single bit among the flip-flops to provide k distinguishable states.
- The number of states can be doubled if the shift register is connected as a switch-tail ring counter. A switch-tail ring counter is a circular shift register with the complemented output of the last flip-flop connected to the input of the first flip-flop.





CLK	Q_3	Q_2	Q_1	Q_0
0				
1				
2				
3				
4				
5				
6				
7				
0				

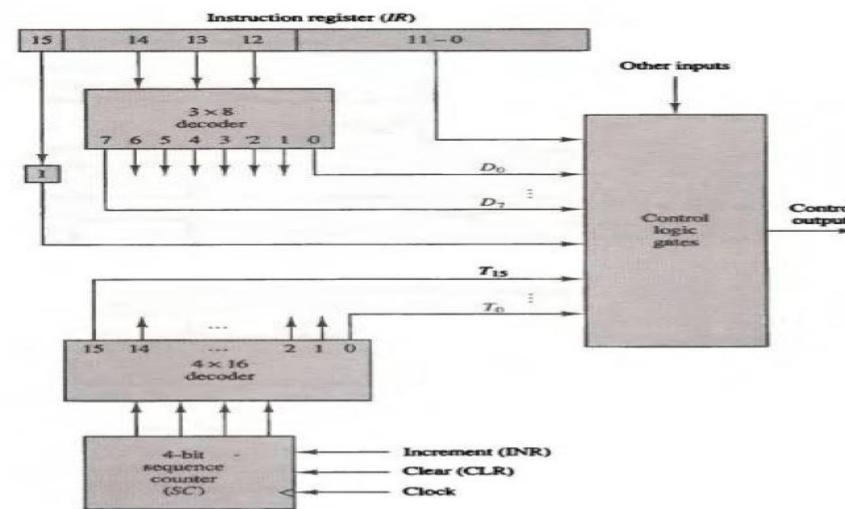
Q Consider a 4-bit Johnson counter with an initial value of 0000. The counting sequence of this counter is **(GATE-2015) (2 Marks)**

- (A) 0, 1, 3, 7, 15, 14, 12, 8, 0
- (B) 0, 1, 3, 5, 7, 9, 11, 13, 15, 0
- (C) 0, 2, 4, 6, 8, 10, 12, 14, 0
- (D) 0, 8, 12, 14, 15, 7, 3, 1, 0

Q Let $k = 2^n$. A circuit is built by giving the output of an n-bit binary counter as input to an n -to- 2^n bit decoder. This circuit is equivalent to a **(CS-2014) (2 Marks)**

(A) k-bit binary up counter. **(B)** k-bit binary down counter.

(C) k-bit ring counter. **(D)** k-bit Johnson counter.



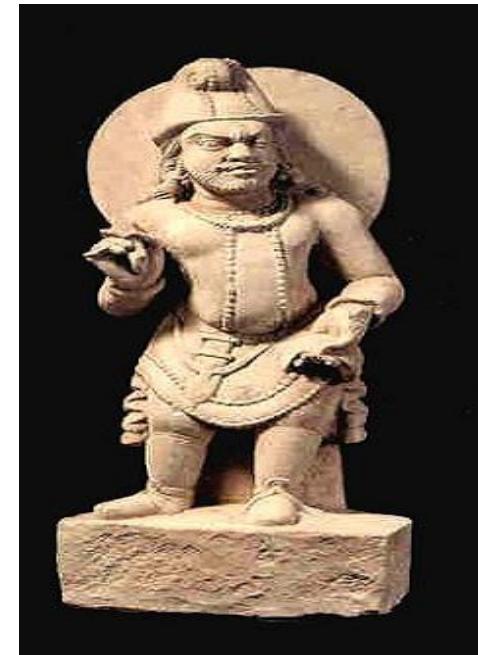
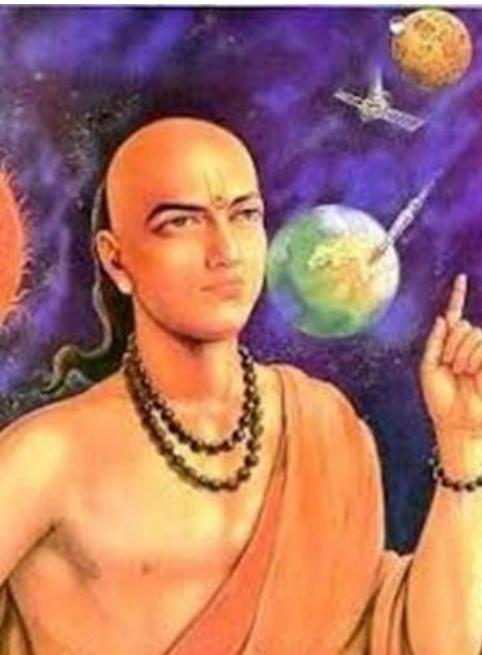
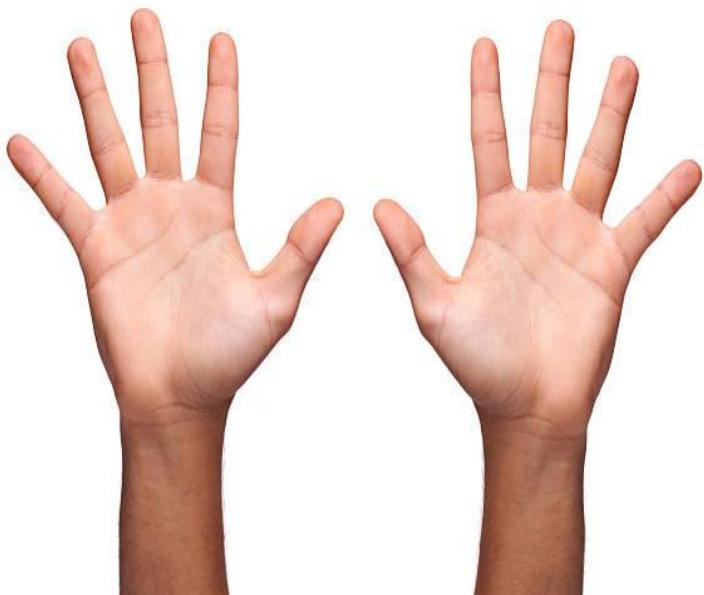
Basics

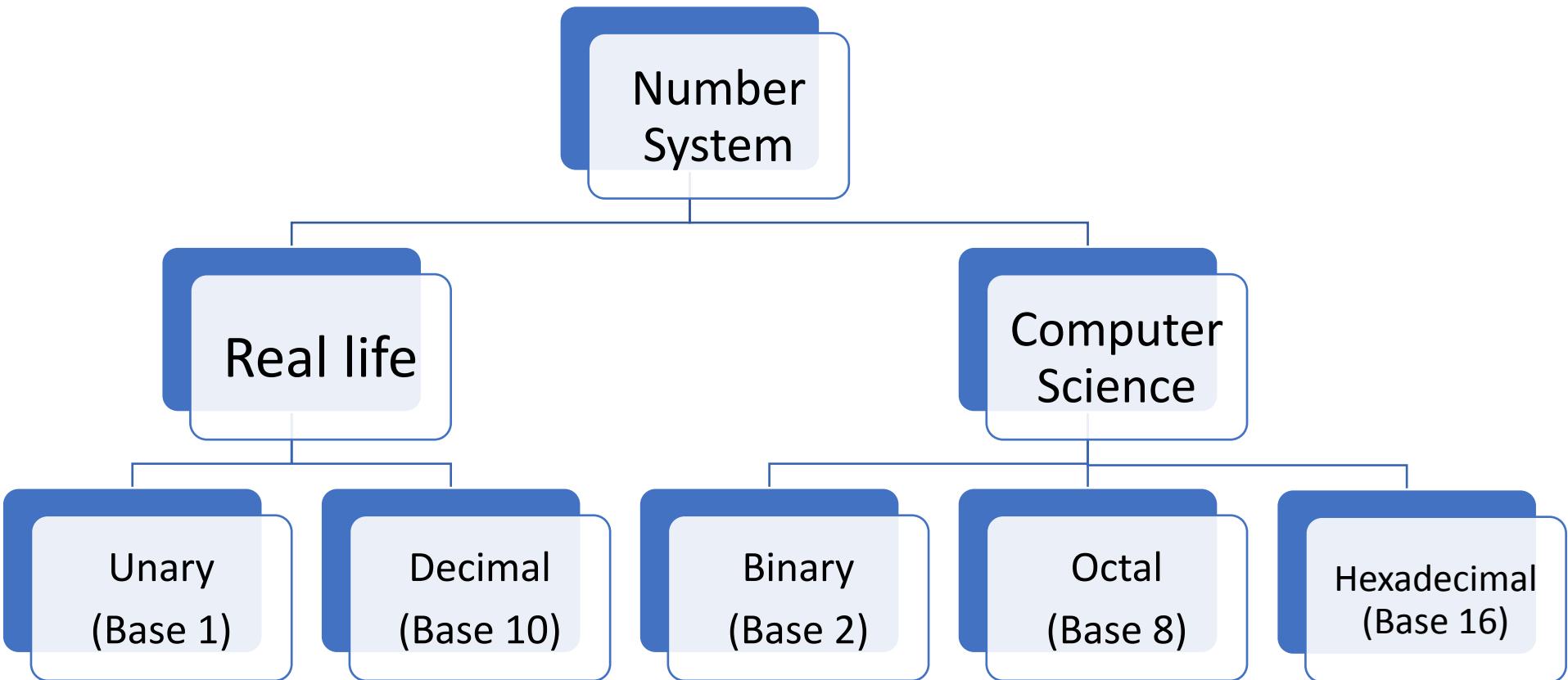
- Main idea in number system is counting and representation of quantity.
- In stone age or even today the basic idea of counting in Unary counting
 - Remember how we started counting on fingers and using abacus
 - How can we use it for counting sheep or anything?
- But when we want to count larger quantity in anything, day to day life, business, maths or research, we cannot work with unary system.



Samsung pays Apple \$1 Billion sending 30 trucks full of 5 cent coins

- So next popular system we use in real life is decimal system, may be because we have 10 fingers on over hands, in history different cultures have been using base 10 for general purpose counting, Indian, British, roman, Arabic etc.
- Thought **Aryabhata (3500 years back)** was the first one known to extensively worked on the idea of zero, pie, number system, trigonometry, quadratic equations, astronomy etc.
- In Indian culture we were working on very complex math from much early time, Aryabhata was may be first one, who have properly written and published his understanding, and some of his work even reached today.





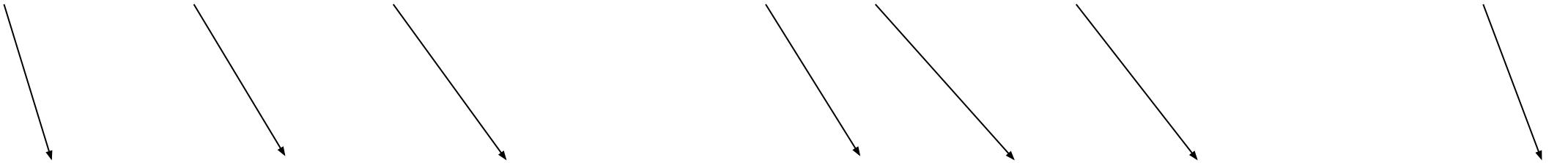
- Formally speaking number system is an ordered set of symbols called digits, from (0 to r-1) if r is this base or radix of the system with rules defined for performing arithmetic operations.
- So, point to be noted is we cannot use a digit higher than r-1 in base r
- Every digit should be of single number, otherwise we may get confused.

Number System	Base or Radix	Digits or symbols
Binary	2	0,1
Octal	8	0,1,2,3,4,5,6,7
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Number System	Base or Radix	Digits or symbols
Unary	1	0/1
Decimal	10	0,1,2,3,4,5,6,7,8,9

- Collection of digits makes a number which has two parts integer and fractional, separated by a Radix point (decimal point)
- The number system we use for counting are weighted number and the idea is

$$(a_{n-1} \ a_{n-2} \ a_{n-3} \dots \ a_1 \ a_0. \ a_{-1} \ a_{-2} \dots \ a_{-m})$$



Value of any base in decimal(any base to decimal conversion)

$$N_r = a_{n-1}r^{n-1} + a_{n-2}r^{n-2} + a_{n-3}r^{n-3} + \dots + a_1r^1 + a_0r^0 + a_{-1}r^{-1} + \dots + a_{-m}r^{-m}$$

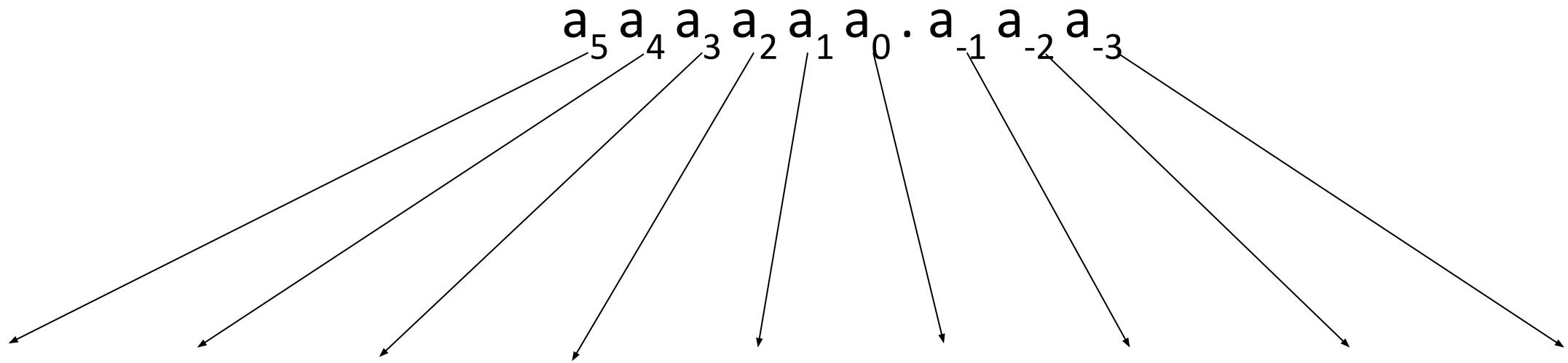
- $Value = \sum_{i=-m}^{n-1} a_i \cdot r^i$

Decimal Number System

- Decimal Number System is a base-10 system that has ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- The decimal number system is said to be of base, or radix, 10 because it uses 10 digits and the coefficients are multiplied by powers of 10.
- This is the base that we often use in our day to day life.
- Example: $(7,392)_{10}$, where 7,392 is a shorthand notation for what should be written as

$$7 * 10^3 + 3 * 10^2 + 9 * 10^1 + 2 * 10^0$$

The diagram illustrates the conversion of the decimal number 7,392 into its expanded form. It features a vertical arrow pointing downwards from the digit 7,392 to the first term of the expansion. From the digit 3, two arrows point to the second term; one arrow points to the 3 itself, and another points to the power of 10. Similarly, from the digit 9, two arrows point to the third term, and from the digit 2, two arrows point to the fourth term. This visualizes how each digit's value is multiplied by a power of 10 based on its position.



$$a_5 \cdot 10^5 + a_4 \cdot 10^4 + a_3 \cdot 10^3 + a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + a_{-2} \cdot 10^{-2} + a_{-3} \cdot 10^{-3}$$

Binary Number System

- The coefficients of the binary number system have only two possible values: 0 and 1.
- Each coefficient a_j is multiplied by a power of the radix, e.g., 2^j , and the results are added to obtain the decimal equivalent of the number.
- $N_r = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + a_{n-3}2^{n-3} + \dots + a_12^1 + a_02^0 + a_{-1}2^{-1} + \dots + a_{-m}2^{-m}$
- $Value = \sum_{i=-m}^{n-1} a_i \cdot 2^i$

Example: $(11010.11)_2$ value in Decimal?

$$1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 + 1 * 2^{-1} + 1 * 2^{-2} = 26.75$$

- Similarly, there can be many bases and they can be easily converted to decimal number system.
- An example of a base-5 number is

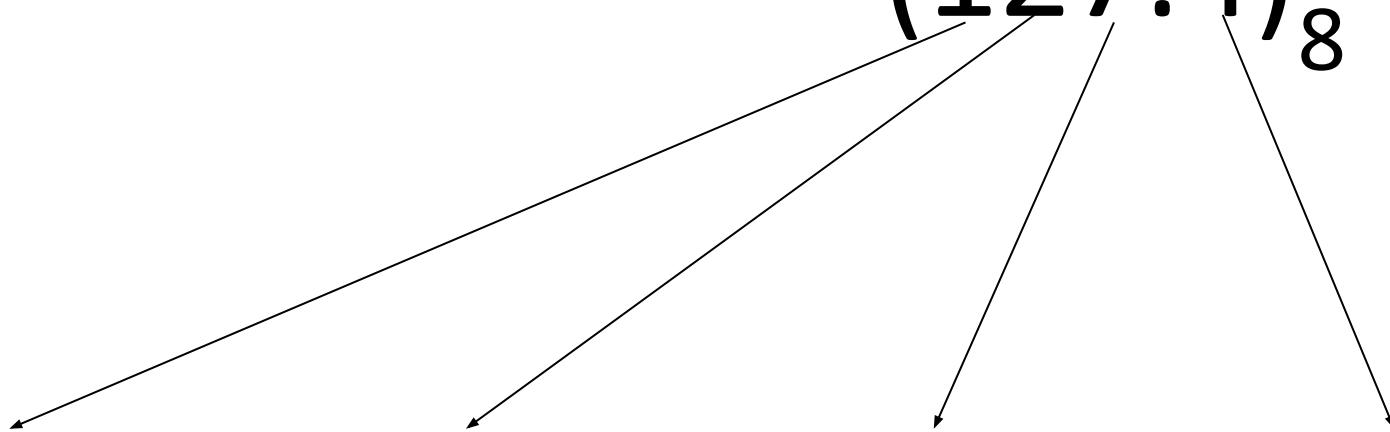
$$(4021.2)_5$$

The diagram shows the base-5 number $(4021.2)_5$ with four arrows pointing from its digits to the corresponding terms in the conversion equation below. The first three digits (4, 0, 2) point to the terms $4 * 5^3$, $0 * 5^2$, and $2 * 5^1$ respectively. The digit 1 points to the term $1 * 5^0$. The digit 2 points to the term $2 * 5^{-1}$.

$$4 * 5^3 + 0 * 5^2 + 2 * 5^1 + 1 * 5^0 + 2 * 5^{-1} = (511.4)_{10}$$

$$(4213.21)_5 \longrightarrow ()_{10} \longrightarrow ()_7$$

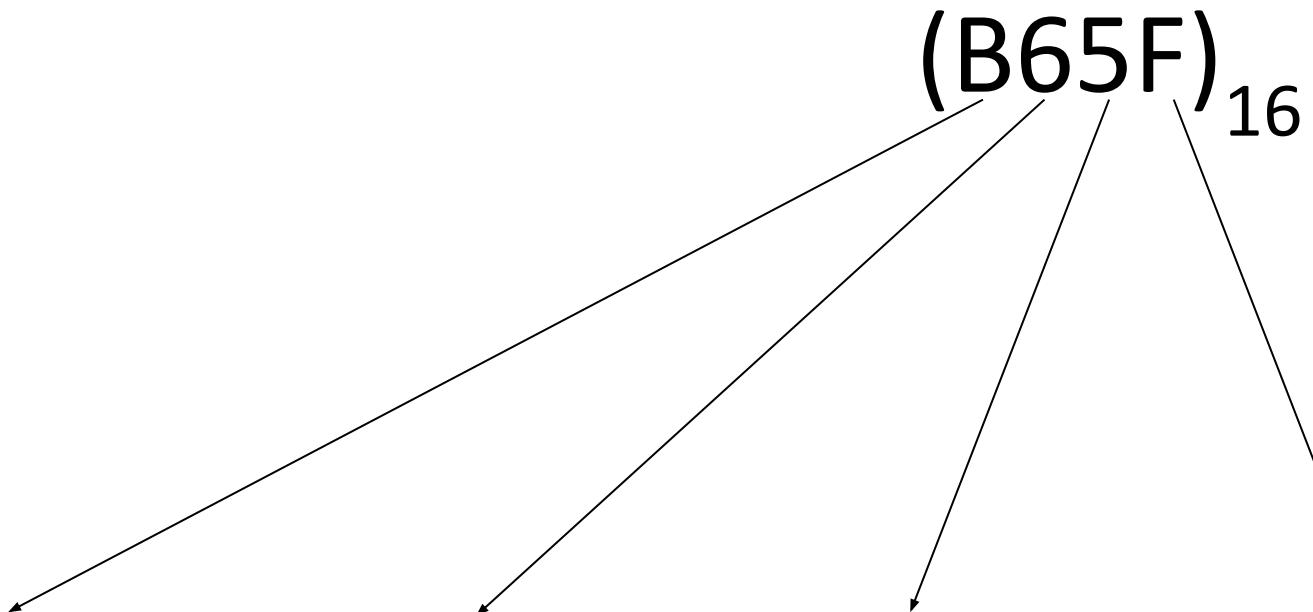
$(127.4)_8$



$$1 * 8^2 + 2 * 8^1 + 7 * 8^0 + 4 * 8^{-1} = (87.5)$$

Hexadecimal Number System

- The *letters of the alphabet* are used to supplement the 10 decimal digits when the base of the number is greater than 10.
- For example, in the hexadecimal (base-16) number system, the first 10 digits are borrowed from the decimal system. The letters A, B, C, D, E, and F are used for the digits 10, 11, 12, 13, 14, and 15, respectively.
- $N_r = a_{n-1}(16)^{n-1} + a_{n-2}(16)^{n-2} + a_{n-3}(16)^{n-3} + \dots + a_1(16)^1 + a_0(16)^0 + a_{-1}(16)^{-1} + \dots + a_{-m}(16)^{-m}$
- $Value = \sum_{i=-m}^{n-1} a_i \cdot (16)^i$



Conversion

- Basic idea is we want to convert from any base to any base. To do that first we convert from any base to decimal and then from decimal to any base. But if the base is in the power to 2, then we can use base to two convert which is relatively easy.
- In case a binary number usually becomes so large so better idea is to club digits in a group of three or four, leading to octal, hexadecimal.

Hexadecimal System / Base 16 System	
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10-A
1011	11-B
1100	12-C
1101	13-D
1110	14-E
1111	15-F

Octal System / Base 8 System	
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

base 4 System	
00	0
01	1
10	2
11	3

Q Let the representation of a number in base 3 be 210. What is the hexadecimal representation of the number? **(GATE 2021)**

(a) 15

(b) 21

(c) D2

(d) 528

Q If x and y are two decimal digits and $(0.1101)_2 = (0.8xy5)_{10}$, the decimal value of x+y is _____. **(GATE 2021) (1 MARKS)**

Q Consider the unsigned 8-bit fixed point binary number representation below,

$b_7 b_6 b_5 b_4 b_3.b_2 b_1 b_0$

where the position of the binary point is between b_3 and b_2 . Assume b_7 is the most significant bit. Some of the decimal numbers listed below cannot be represented exactly in the above representation **(GATE-2017) (2 Marks)**

- (i)** 31.500 **(ii)** 0.875 **(iii)** 12.100 **(iv)** 3.001

Which one of the following statements is true?

- (A)** None of (i), (ii), (iii), (iv) can be exactly represented
(B) Only (ii) cannot be exactly represented
(C) Only (iii) and (iv) cannot be exactly represented
(D) Only (i) and (ii) cannot be exactly represented

Q Consider a quadratic equation $x^2 - 13x + 36 = 0$ with coefficients in a base b . The solutions of this equation in the same base b are $x = 5$ and $x = 6$. Then $b = \underline{\hspace{2cm}}$. (GATE-2017) (2 Marks)

Q The representation of the value of a 16-bit unsigned integer **X** in a hexadecimal number system is **BCA9**. The representation of the value of **X** in octal number system is: **(GATE-2017) (SET-2)**

(A) 571244

(B) 736251

(C) 571247

(D) 136251

Q The n-bit fixed-point representation of an unsigned real number X uses f bits for the fraction part. Let $i = n-f$. The range of decimal values for X in this representation is **(GATE-2017) (2 Marks)**

(A) 2^{-f} to 2^i

(B) 2^{-f} to $(2^i - 2^{-f})$

(C) 0 to 2^i

(D) 0 to $(2^i - 2^{-f})$

Q Consider the equation $(123)_5 = (x8)_y$ with x and y as unknown. The number of possible solutions is _____. **(GATE-2014) (1 Marks)**

(A) 1

$$(123)_5 = (x8)_y$$

(B) 2

(C) 3

(D) 4

The base (or radix) of the number system such that the following equation holds is _____.

$$\frac{312}{20} = 13.1$$

(GATE-2014) (2 Marks)

Q (1217)₈ is equivalent to (GATE-2009) (2 Marks)

A.(1217)₁₆

B.(028F)₁₆

C.(2297)₁₀

D.(0B17)₁₆

Q Let r denote number system radix. The only value(s) of r that satisfy the equation

$$\sqrt{121_r} = 11_r \text{ is/are}$$

(GATE-2008) (2 Marks)

(A) decimal 10

(B) decimal 11

(C) decimal 10 and 11

(D) any value > 2

Q The hexadecimal representation of $(657)_8$ is **(CS-2005) (1 Marks)**

(A) 1AF

$(657)_8$

(B) D78

(C) D71

(D) 32F

Q The number $(123456)_8$ is equivalent to **(GATE-2004) (1 Marks)**

(A) $(A72E)_{16}$ and $(22130232)_4$

$(123456)_8$

(B) $(A72E)_{16}$ and $(22131122)_4$

(C) $(A73E)_{16}$ and $(22130232)_4$

(D) $(A62E)_{16}$ and $(22120232)_4$

Q What will be no of digits required to represent 123-bit binary no into a decimal no?

Q. Which of the following is/are EQUAL to 224 in radix-5 (i.e., base-5) notation? (Gate 2024 CS) (2 Mark) (MSQ)

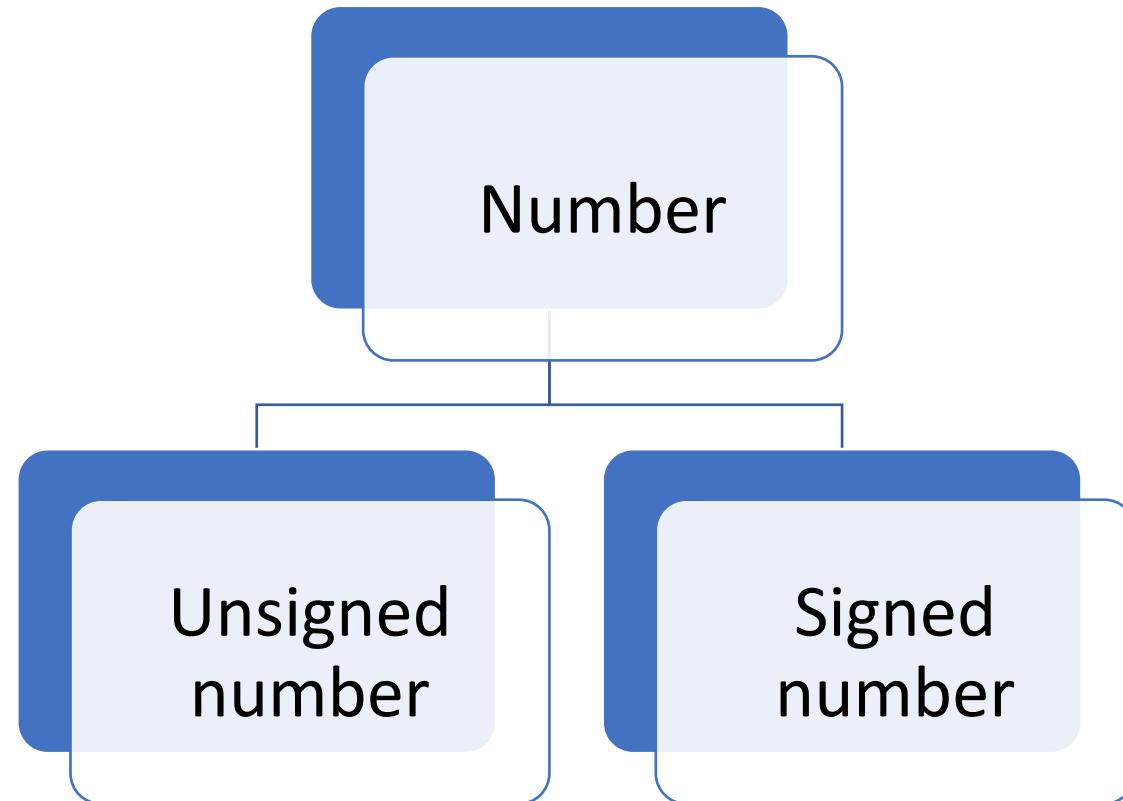
(a) 64 in radix-10

(b) 100 in radix-8

(c) 50 in radix-16

(d) 121 in radix-7

Representation of a number

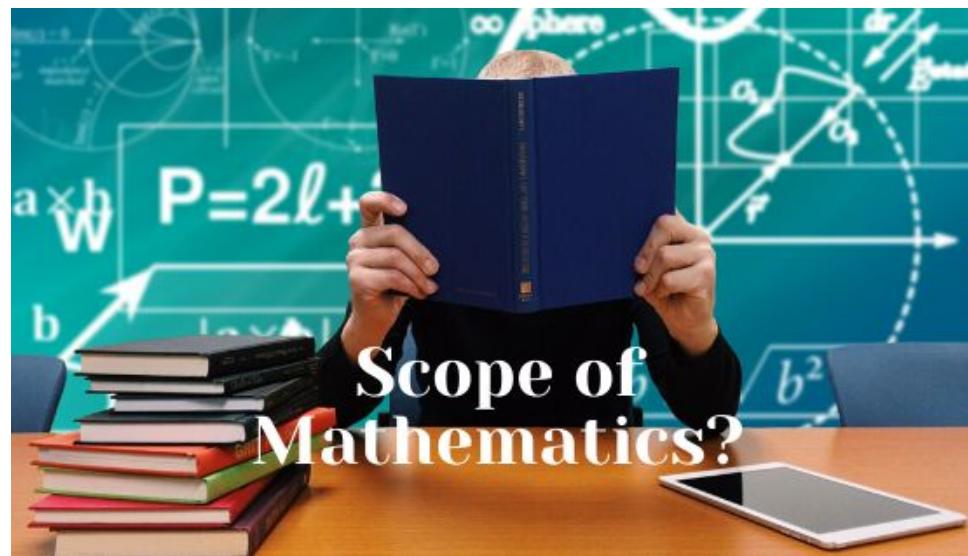
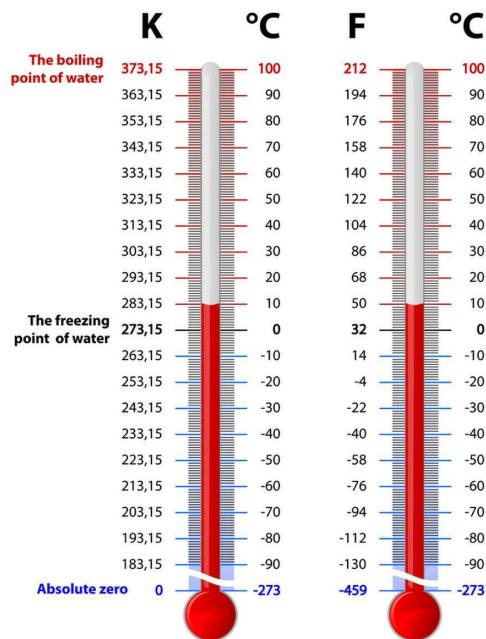


Unsigned number

- Until now, we have considered only unsigned numbers.
- Unsigned numbers without any positive or negative sign, these numbers represent only magnitude.
- In real life, we have only unsigned numbers (considered as positive) (natural numbers) for e.g. number of stars, number of trees etc.
- If n bits are used to store the number, then all the n bits are used to store number / magnitude / absolute value.
- If we have a n bit Unsigned number than range is from 0 to $2^n - 1$

Signed number

- To perform now a day's mathematical work for e.g. profit and loss statement, temperature scale, subtraction, representation of negative numbers is needed.
- So, we use + and – symbols to represent the positive and negative sign respectively, along with magnitude or absolute value. Number without a sign is considered as positive.
- Suppose that an n-bit word is to contain a signed binary number. One bit is reserved to represent the sign of the number, while the remaining n-1 bits indicate its magnitude. To permit uniform processing of all n-bits, the sign is placed to the leftmost magnitude (MSB), 0 and 1 are used to denote plus and minus respectively.



Signed binary number

- A signed number has two components of information
 - The sign of a number, i.e. positive(+ve) or negative(-ve)
 - The magnitude and or absolute value
- In real life when we write a signed number then we represent both information, sign and magnitude or absolute value separately for e.g. +9, -543 etc.
- Because of the limitation of the computer hardware as circuit only understand 0 and 1, every piece of information in computer must be represented in terms of numbers i.e. 0 and 1, whether it is sign or magnitude.
- So, a general convention in first bit (left most) of the number is reserved to represent the sign of the number where 0 means +Ve number and 1 means -Ve number. And the rest of the bits represent the magnitude or absolute value

Signed number representation

Signed magnitude convention(here the negative number is represented by it's sign)

Signed magnitude representation

Signed complement system
(here the negative number is represented by it's complement)

1's complement representation

2's complement representation

Decimal	Signed Magnitude	1's Complement	2's Complement
+7			
+6			
+5			
+4			
+3			
+2			
+1			
+0			
-0			
-1			
-2			
-3			
-4			
-5			
-6			
-7			
-8			

Decimal	Signed Magnitude	1's Complement	2's Complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	0000
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-----	-----	1000

- All Positive number have 0 in the leftmost position, negative numbers have 1 in the leftmost bit position.
- All three notations representations have identical representation if the number is positive.

Sign magnitude representation

- **Introduction:** - It is the simplest form of representation, where in an n-bit word, the right most n-1 bits (from LSB) hold the magnitude of the number and n^{th} bit (Left most, MSB) is assigned for sign, where 0 → +ve and 1 → -ve.
- **Range:** - from $-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$ total $2^n - 1$
- **How to read/write a number:** - Convert the magnitude of the number in binary, then write the number as right as possible, the left most bit must be filled with 0 or 1 according to +ve or -ve number respectively.

- **Advantage:** - Easy to read and write, because it is a weighted code system, directly reading and writing is possible.
- **Disadvantage:** - One drawback in sign-magnitude representation is addition and subtraction require consideration of first signs of the number and second their relative magnitude/absolute value in order to carry out the required operation. This idea is used in real life, but is awkward when employed in computer arithmetic because of the separate handling of the sign and the magnitude.
- One more problem is there are two representation for 0, i.e. -0 & +0, because of which there is a loss of one presentation, and create confusion.
- **Number extension :** - In computer the size of a location is fixed, so it is often required to write a smaller number in a larger space, this can be done by sign extension, where, the magnitude is written as far right as possible, sign bit is written left most, and all the empty cells must be filled/padded with zero.

Q Consider $Z = X - Y$, where X , Y and Z are all in sign-magnitude form. X and Y are each represented in n bits. To avoid overflow, the representation of Z would require a minimum of **(GATE-2019) (1 Marks)**

- (A) n bits
- (B) $n - 1$ bits
- (C) $n + 1$ bits
- (D) $n + 2$ bits

COMPLEMENTS OF NUMBERS

- . Complements are used in digital computers to *simplify the subtraction operation* and for *logical manipulation*.
- . Simplifying operations leads to simpler, less expensive circuits to implement the operations.
- . There are two types of complements for each base- r system: *the radix complements* and the *diminished radix complement*.
- . The first is referred to as the r 's complement and the second as the $(r - 1)$'s complement.

9's Complement

- . Given a number N in base r having n digits, the $(r - 1)$'s complement of N , i.e., its diminished radix complement, is defined as $(r^n - 1) - N$.
- . For decimal numbers, $r = 10$ and $r - 1 = 9$, so the 9's complement of N is $(10^n - 1) - N$.
- . In this case, 10^n represents a number that consists of a single 1 followed by n 0's.
- . $10^n - 1$ is a number represented by n 9's.
- . Example: if $n = 4$, we have $10^4 = 10,000$ and $10^4 - 1 = 9999$.
- . Some examples of 9's complement:
 - The 9's complement of 546700 is $999999 - 546700 = 453299$.
- . The 9's complement of 012398 is $999999 - 012398 = 987601$.

1's Complement

- . For binary numbers, $r = 2$ and $r - 1 = 1$, so the 1's complement of N is $(2^n - 1) - N$.
- . For example, if $n = 4$, we have $2^4 = (10000)_2$ and $2^4 - 1 = (1111)_2$.
- . *"The 1's complement of a binary number is formed by changing 1's to 0's and 0's to 1's."*
- . Example:
 - The 1's complement of 1011000 is 0100111.
 - The 1's complement of 0101101 is 1010010.

Point to Note

- . The $(r - 1)$'s complement of octal or hexadecimal numbers is obtained by subtracting each digit from 7 or F (decimal 15), respectively.

1's Complement Representation

- **Introduction:** - here the negative number is represented by taking complement.
- **Range:** - from $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$, total $2^n - 1$ representation
- **How to read/write a number:** -
 - **how to write a number:** - If the number is +ve then number is written same as of the sign magnitude representation, first bit 0 representing the sign of the number, followed by n-1 bits of the magnitude.
 - But if the number is negative(-x) than first represent its +ve counterpart(+x) then take its 1's complement of the number.
 - 1's complement of a binary number is obtained by changing each 0 to 1 and each 1 to 0, then we get representation for -x.
 - we can directly read 2's complement number and then add 1
- **how to read a number:** - Similarly to read a number, if first bit is 0 then read it directly, otherwise take 1's complement of the number, then read the number if it is +x, then number was -x.
- we can directly write 2's complement number and then subtract 1

- **Advantage:** - it provides a way to easily understand the 2's complement system, easy to calculate, used as logical operations.
- **Disadvantage:** - One drawback in 1's complement representation is addition and subtraction differ from simple addition and subtraction of arithmetic, Not a weighted code system
- there are two representation for 0, i.e. -0 & +0, because of which there is a loss of one presentation.
- Neither we can write a negative number directly nor we can read a negative number directly, as it is not a weighted code system.

• **Number extension**

- In computer the size of a location is fixed, so it is often required to write a smaller number in a larger space, this can be done by sign extension, where,
- if the number is positive extension is done same as that of sign magnitude extension.
- If the number is -ve then we must write the number as far right as possible and then all the empty cells must be filled with 1(in negative logic 0 is represented by 1).

2s complement
representation

- **How to Read: -**
 - Similarly, to read a number, if first bit is 0 then read it directly,
 - otherwise if it is a negative number(-x) first subtract 1 from it to make it 1's complement of the number, then take 1's complement, then read the number(+x) and then change its sign as number was -x.
 - a number written in 2'sc can be read directly, with the following formula. Value = $(-1)(a_{n-1}2^{n-1}) + \sum_{i=0}^{n-2} a_i \cdot 2^i$
- **Number extension**
 - In computer the size of a location is fixed, so it is often required to write a smaller number in a larger space, this can be done by sign extension, where,
 - if the number is positive extension is done same as that of sign magnitude extension.
 - If the number is -ve then we must write the number as far right as possible and then all the empty cells must be filled with 1(in negative logic 0 is represented by 1).

- **Advantage:** -
 - Easy to do arithmetic operations with 2's complement representation. As end round carry can be discarded.
 - Has only one representation for zero which is always positive, therefore provides better clarity, efficiency and more range with no wastage.
 - It is a waited code system, direct reading and writing is possible.
 - Most popular representation used all the computers.
- **Disadvantage:** -
 - One drawback in 2's complement representation is relatively difficult to understand. But in application like floating point representation it is not much required, and it takes more space.

Q Let R1 and R2 be two 4-bit registers that store numbers in 2's complement form. For the operation R1+R2, which one of the following values of R1 and R2 gives an arithmetic overflow? **(GATE 2022) (1 MARKS)**

- (A)** R1 = 1011 and R2 = 1110
- (B)** R1 = 1100 and R2 = 1010
- (C)** R1 = 0011 and R2 = 0100
- (D)** R1 = 1001 and R2 = 1111

Q In 16-bit 2's complement representation, the decimal number -28 is
(GATE-2019) (2 Marks)

(A) 1111 1111 0001 1100

(B) 0000 0000 1110 0100

(C) 1111 1111 1110 0100

(D) 1000 0000 1110 0100

Q When two 8-bit numbers $A_7 \dots A_0$ and $B_7 \dots B_0$ in 2's complement representation (with A_0 and B_0 as the least significant bits) are added using a ripple-carry adder, the sum bits obtained are $S_7 \dots S_0$ and the carry bits are $C_7 \dots C_0$. An overflow is said to have occurred if **(GATE-2017) (2 Marks)**

- (A)** the carry bit C_7 is 1 **(B)** all the carry bits (C_7, \dots, C_0) are 1
- (C)** $(A_7 \cdot B_7 \cdot \bar{S}_7 + \bar{A}_7 \cdot \bar{B}_7 \cdot S_7)$ is 1 $(A_0 \cdot B_0 \cdot \bar{S}_0 + \bar{A}_0 \cdot \bar{B}_0 \cdot S_0)$ is 1

Q The 16-bit 2's complement representation of an integer is 1111 1111 1111 0101; its decimal representation is _____.(GATE-2016) (1 Marks)

1111 1111 1111 0101

Q The addition of 4-bit, two's complement, binary numbers 1101 and 0100 results in **(GATE- 2016) (1 Marks)**

(A) 0001 and an overflow

(B) 1001 and no overflow

(C) 0001 and no overflow

(D) 1001 and an overflow

Q Let X be the number of distinct 16-bit integers in 2's complement representation. Let Y be the number of distinct 16-bit integers in sign magnitude representation. Then $X-Y$ is? **(GATE-2016) (2 Marks)**

(a) 1

(b) 2

(c) 3

(d) 4

Q The smallest integer that can be represented by an 8-bit number in 2's complement form is **(GATE-2013) (1 Marks)**

Q P is a 16-bit signed integer. The 2's complement representation of P is $(F87B)_{16}$.
The 2's complement representation of $8 \times P$ is **(GATE-2010) (2 Marks)**

a) $(C3D8)_{16}$

F87B

b) $(187B)_{16}$

c) $(F878)_{16}$

d) $(987B)_{16}$

Q A processor that has carry, overflow and sign flag bits as part of its program status word (PSW) performs addition of the following two 2's complement numbers 01001101 and 11101001. After the execution of this addition operation, the status of the carry, overflow and sign flags, respectively will be: **(GATE-2008) (2 Marks)**

(A) 1, 1, 0

(B) 1, 0, 0

(C) 0, 1, 0

(D) 1, 0, 1

Q We consider the addition of two 2's complement numbers $b_{n-1} b_{n-2} \dots b_0$ and $a_{n-1} a_{n-2} \dots a_0$. A binary adder for adding unsigned binary numbers is used to add the two numbers. The sum is denoted by $c_{n-1} c_{n-2} \dots c_0$ and the carry-out by c_{out} . Which one of the following options correctly identifies the overflow condition? **(GATE-2006) (2 Marks)**

- (A) $c_{\text{out}} (\overline{a_{n-1}} \oplus \overline{b_{n-1}})$
- (B) $\overline{a_{n-1}} \overline{b_{n-1}} \overline{c_{n-1}} + \overline{a_{n-1}} \overline{b_{n-1}} c_{n-1}$
- (C) $c_{\text{out}} \oplus c_{n-1}$
- (D) $\overline{a_{n-1}} \oplus \overline{b_{n-1}} \oplus c_{n-1}$

Q Using a 4-bit 2's complement arithmetic, which of the following additions will result in an overflow? **(GATE-2004) (2 Marks)**

- (i) $1100 + 1100$ (ii) $0011 + 0111$ (iii) $1111 + 0111$

(A) (i) only

(B) (ii) only

(C) (iii) only

(D) (i) and (iii) only

Q Consider a system that uses 5 bits for representing signed integers in 2's complement format. In this system, two integers A and B are represented as A= 01010 and B=11010. Which one of the following operations will result in either an arithmetic overflow or an arithmetic underflow? **(Gate 2024,CS) (1 Marks) (MCQ)**

(a) A+B

(b) A-B

(b) B-A

(c) 2*B

Q. The number-6 can be represented as 1010 in 4-bit 2's complement representation. Which of the following is/are CORRECT 2's complement representation(s) of -6? **(GATE-2025)**

A) 1000 1010 in 8-bits

B) 1111 1010 in 8-bits

C) 1000 0000 0000 1010 in 16-bits

D) 1111 1111 1111 1010 in 16-bits

Q. The following two signed 2's complement numbers (multiplicand M and multiplier Q) are being multiplied using Booth's algorithm:

M: 1100110111101101

Q: 1010010010101010

The total number of addition and subtraction operations to be performed is _____. (Answer in integer)(**GATE 2025**)

BINARY CODES

Gray Code

- The **reflected binary code (RBC)**, also known just as **reflected binary (RB)** or **Gray code** after Frank Gray, is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit).
- The reflected binary code was originally designed to prevent spurious output from electromechanical switches. Today, Gray codes are widely used to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems.
- A binary number is converted to gray code to reduce switching operation.
- The Gray code is used in applications in which the normal sequence of binary numbers generated by the hardware may produce an error or ambiguity during the transition from one number to the next.
- If binary numbers are used, a change, for example, from 0111 to 1000 may produce an intermediate erroneous number 1001 if the value of the rightmost bit takes longer to change than do the values of the other three bits. This could have serious consequences for the machine using the information.
- The Gray code eliminates this problem, since only one bit changes its value during any transition between two numbers.
- Gray Code is an unweighted code. In gray code two successive values differ in only 1 bit.
- Application
 - Position encoders
 - Mathematical puzzles
 - Genetic algorithms
 - Boolean circuit minimization
 - Error correction

Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

Binary-Coded Decimal Code (BCD)

- In computing and electronic systems, **binary-coded decimal (BCD)** is a class of binary encodings of decimal numbers where each decimal digit is represented by a fixed number of bits, usually four or eight.
- BCD typically encodes two decimal digits within a single byte by taking advantage of the fact that four bits are enough to represent the range 0 to 9.
- BCD's main virtue is its more accurate representation and rounding of decimal quantities as well as an ease of conversion into human-readable representations, in comparison to binary positional systems special used to represent currency value. BCD's principal drawbacks are a small increase in the complexity of the circuits needed to implement basic arithmetic and a slightly less dense storage.
- BCD was used in many early decimal computers, and is implemented in the instruction set of machines such as the IBM System/360 series and its descendants, Digital Equipment Corporation's VAX, the Burroughs B1700, and the Motorola 68000-series processors.
- Although BCD *per se* is not as widely used as in the past and is no longer implemented in newer computers' instruction sets (such as ARM; x86 does not support its BCD instructions in long mode any more), decimal fixed-point and floating-point formats are still important and continue to be used in financial, commercial, and industrial computing, where subtle conversion and fractional rounding errors that are inherent in floating point binary representations cannot be tolerated.

- A number with k decimal digits will require $4k$ bits in BCD.
- A group of 4 bits represents one decimal digit.
- A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9.

Decimal No	BCD Representation
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

- A BCD number greater than equal to 10 looks different from its equivalent binary number, even though both contain 1's and 0's.
- Moreover, the binary combinations 1010 through 1111 are not used and have no meaning in BCD.
- Example: $(185)_{10} = (0001\ 1000\ 0101)$ in BCD = $(10111001)_2$
- 1 - 0001, 8 – 1000 and 5 – 0101
- The BCD value has 12 bits to encode the characters of the decimal value, but the equivalent binary number needs only 8 bits.
- It is obvious that the representation of a BCD number needs more bits than its equivalent binary value.
- The advantage is that it is important to realize that BCD numbers are decimal numbers and not binary numbers, although they use bits in their representation.
- The only difference between a decimal number and BCD is that decimals are written with the symbols 0, 1, 2, ..., 9 and BCD numbers use the binary code 0000, 0001, 0010, .., 1001. The decimal value is exactly the same.

Some Points to Remember

- BCD code is an example of weighted code.
- In a weighted code, each bit position is assigned a weighting factor in such a way that each digit can be evaluated by adding the weights of all the 1's in the coded combination.
- The BCD code has weights of 8, 4, 2, and 1, which correspond to the power of two values of each bit. The bit assignment 0110, for example, is interpreted by the weights to represent decimal 6 because $8 * 0 + 4 * 1 + 2 * 1 + 1 * 0 = 6$.

Excess-3 Code

- **Excess-3, 3-excess or 10-excess-3** binary code (often abbreviated as **XS-3, 3XS** or **X3**) or **Stibitz code** (after George Stibitz, who built a relay-based adding machine in 1937) is a self-complementary binary-coded decimal (BCD) code and numeral system.
- It is a biased representation. Excess-3 code was used on some older computers as well as in cash registers and hand-held portable electronic calculators of the 1970s, among other uses.
- To encode a number such as 127, one simply encodes each of the decimal digits as above, giving (0100, 0101, 1010).
- Excess-3 arithmetic uses different algorithms than normal non-biased BCD or binary positional system numbers. After adding two excess-3 digits, the raw sum is excess-6. For instance, after adding 1 (0100 in excess-3) and 2 (0101 in excess-3), the sum looks like 6 (1001 in excess-3) instead of 3 (0110 in excess-3).
- In order to correct this problem, after adding two digits, it is necessary to remove the extra bias by subtracting binary 0011 (decimal 3 in unbiased binary).

Decimal	Excess-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Excess-3 Code

- Excess-3 codes are examples of self-complementing codes.
- Such codes have the property that the 9's complement of a decimal number is obtained directly by changing 1's to 0's and 0's to 1's (i.e., by complementing each bit in the pattern).
- Excess-3 is an unweighted code in which each coded combination is obtained from the corresponding binary value plus 3.
- Example: 395 in decimal will be represented in excess-3 code as:

0110 1100 1000.

3 9 5

- 3 in excess-3 representation is: $3 + 3 = 6$ (0110)
- 9 in excess-3 representation is: $9 + 3 = 12$ (1100)
- 5 in excess-3 representation is: $5 + 3 = 8$ (1000)
- 9's Complement of 395: $999 - 395 = 604$, its excess-3 representation would be: 1001 0011 0111, which can be obtained by simply doing a bitwise complementation of 395 excess-3 code.