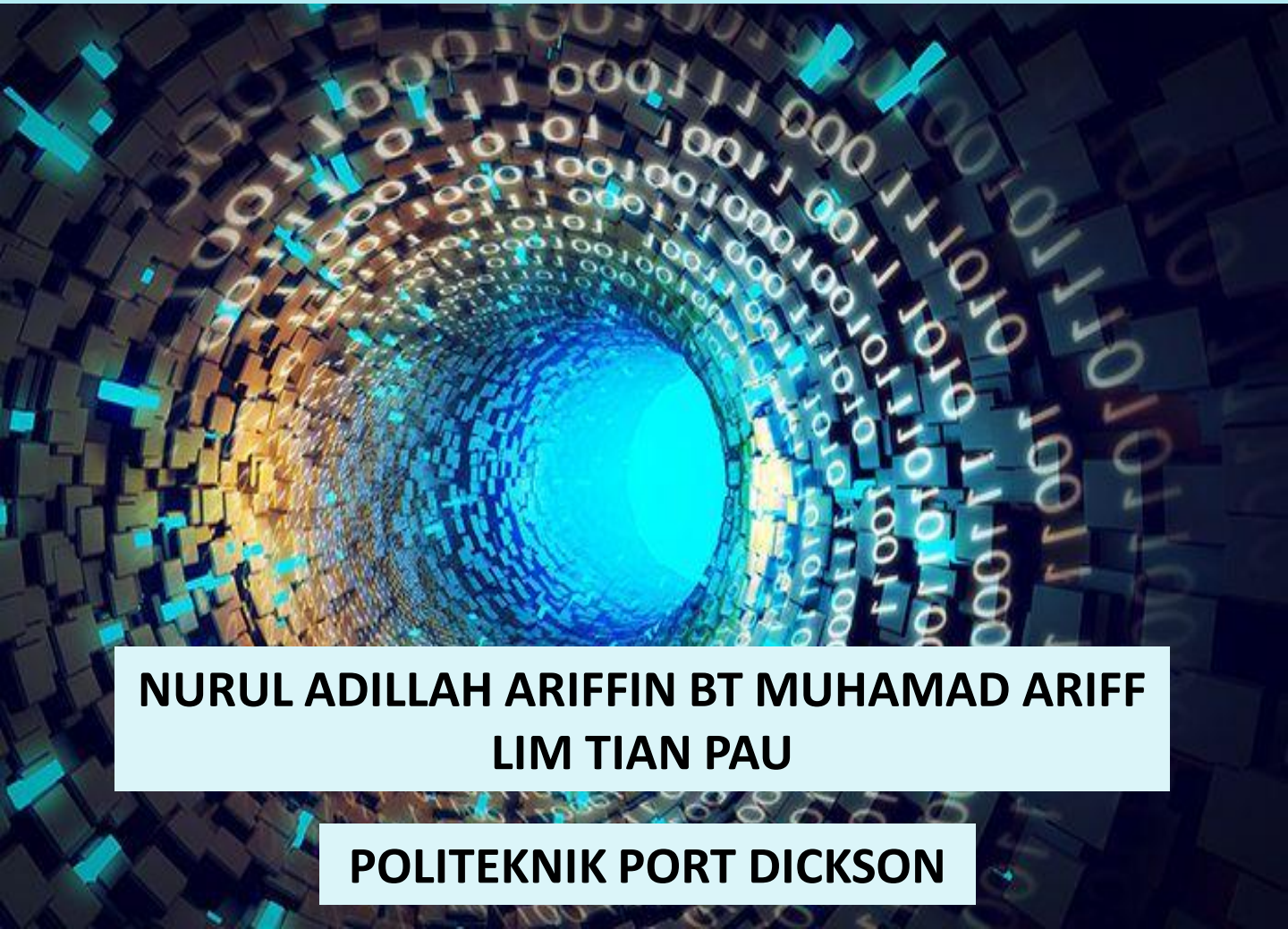
Digital Systems

Principles and Applications

Volume 1



NURUL ADILLAH ARIFFIN BT MUHAMAD ARIFF
LIM TIAN PAU

POLITEKNIK PORT DICKSON

Digital Systems

Principles and Applications

Volume 1

ACKNOWLEDGEMENT

PATRON
Mej (K) Dr. Ishak bin Mohamad
Director, Politeknik Port Dickson

ADVISORS
Abdul Rahim bin Ibrahim
Deputy Director (Academic), Politeknik Port Dickson
Khairun Syatirin bin Md Salleh
Head of Mechanical Engineering Department, Politeknik Port Dickson

EDITOR
Amin Fadilah bin Ahmad
Mechatronic Engineering, Politeknik Port Dickson

FACILITATORS
Che Azlina binti Che Norohoseni
Zuliana binti Zainal Abidin

WRITERS
Nurul Adillah Ariffin Bt Muhamad Ariff
Lim Tian Pau

We would like to convey our utmost gratitude to the Department of Polytechnic and Community College Education particularly the E-learning and Instructional Division (BIPD) for funding our e-book project.

We hereby declare that this module is our original work. To the best of our knowledge it contains no materials previously written or published by another person. However, if there is any, due acknowledgement and credit are mentioned accordingly in the e-book.

Perpustakaan Negara Malaysia Cataloguing-in-Publication Data
e ISBN 978-629-7643-09-0

Cataloguing-in-Publication Data

Perpustakaan Negara Malaysia

A catalogue record for this book is available from the National Library of Malaysia

eISBN 978-629-7643-09-0

PUBLISHED BY:
Politeknik Port Dickson
KM14, Jalan Pantai, 71050 Si Rusa
Port Dickson, Negeri Sembilan

SEPTEMBER 2023

Copyright Each part of this publication may not be reproduced or distributed in any forms by any means or retrieval system without prior written permission.

PREFACE

This e-book Digital Systems: Principles and Applications Volume 1 is published as a reference source for Mechatronic Engineering students of Polytechnic Malaysia.In general, this e-book's content is divided into three major chapters: Number System and Code System, Boolean Expression, and Sequential Logics. The questions included are suitable and compatible with the most recent syllabus created by the Curriculum Development Division at the Department of Polytechnic Studies & Community Colleges (JPPKK), and this reference material is more comprehensive in accordance with the topics of each Digital System course. The applied theory is equally straightforward and furnished with straightforward fundamental ideas for each chapter. Through drills where the information serves as both a source and a guide, the creation of this e-book can be utilized as a reference source.

TABLE OF CONTENTS

NUMBER SYSTEM AND CODE SYSTEM

Introduction

Decimal Number System:

A number system that uses a notation in which each number is expressed in a base of 10.

For example: 3_{10}

Binary Number System:

A number system that uses a notation in which each number is expressed in a base of 2; which uses only two symbols: typically "0" (zero) and "1" (one).

For example: 011011_2

Octal Number System:

A number system that uses a notation in which each number is expressed in a base of 8.

For example: 275_8

Hexadecimal Number System:

A number system that uses a notation in which each number is expressed in a base of 8.

For example: $7DE_{16}$

Convert **BINARY** to **DECIMAL** number:

Example 1

- 111001_2 to decimal number.

Solution:

$= (1x2^5) + (1x2^4) + (1x2^3) + (0x2^2) + (0x2^1) + (1X2^0)$

$= 32 + 16 + 8 + 0 + 0 + 1$

$= \mathbf{57_{10}}$

Example 2

- 111110_2 to decimal number.

Solution:

$= (1x2^5) + (1x2^4) + (1x2^3) + (1x2^2) + (1x2^1) + (0X2^0)$

=

$= \mathbf{????_{10}}$

Example 3

- 100101_2 to decimal number.

Solution:

$= (1x2^5) + (0x2^4) + (0x2^3) + (1x2^2) + (0x2^1) + (1X2^0)$

=

$= \mathbf{????_{10}}$

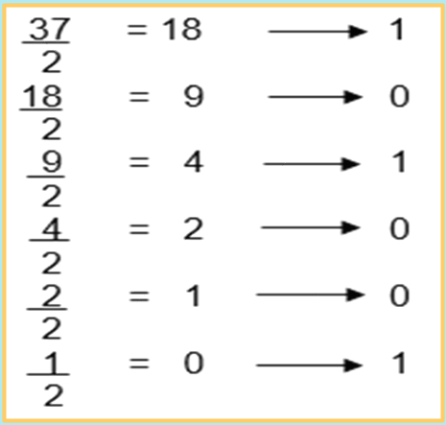
Convert **DECIMAL** to **BINARY** number:

Example 1

- 37_{10} to binary number.

Solution:

= Divide 37 by 2 until the last answer is 1.



So, 37_{10} = **100101_2**

Example 2

- 318_{10} to binary number.

Solution:

= Divide 318 by 2 until the last answer is 1.

=

= **100111110_2**

Example 3

- 156_{10} to binary number.

Solution:

= Divide 156 by 2 until the last answer is 1.

=

= **$??????_2$**

Convert **OCTAL** to **BINARY** number:

Octal Symbol	Binary Equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Conversion Table

Example 1

- 3458 to binary number.

Solution:

- Take Octal number as input
- Convert each digit of octal into binary.
- That will be output as binary number.

= **011100101_2**

Example 2

- 671_8 to binary number.

Solution:

- Take Octal number as input
- Convert each digit of octal into binary.
- That will be output as binary number.

$$6\ 7\ 1_8 = \underbrace{1\ 1\ 0}_{6}\ \underbrace{1\ 1\ 1}_{7}\ \underbrace{0\ 0\ 1}_{1}{}_2$$

= **110111001_2**

Example 3

- 523_8 to binary number.

Solution:

- Take Octal number as input
- Convert each digit of octal into binary.
- That will be output as binary number.

= $????_2$

Example 4

- 416.352_8 to binary number.

Solution:

- Take Octal number as input
- Convert each digit of octal into binary.
- That will be output as binary number.

= $????_2$

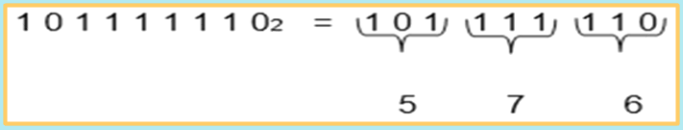
Convert **BINARY** to **OCTAL** number:

Example 1

- 101111110_2 to octal number.

Solution:

- Take Octal number as input
- Convert each digit of octal into binary.
- That will be output as binary number.



= **101111110_2**

= **576_8**

Example 2

- 011010110_2 to octal number.

Solution:

- Take Octal number as input
- Convert each digit of octal into binary.
- That will be output as binary number.

= **326_8**

Example 3

- 01101101_2 to octal number.

Solution:

- Take Octal number as input
- Convert each digit of octal into binary.
- That will be output as binary number.

= **$????_8$**

Convert **OCTAL** to **DECIMAL** number:

Example 1

- 275_8 to decimal number.

Solution:

$= (2x8^2) + (7x8^1) + (5x8^0)$

$= 128$

$= \mathbf{189_{10}}$

Example 2

- 426_8 to decimal number.

Solution:

$= (4x8^2) + (2x8^1) + (6x8^0)$

=

$= \mathbf{278_{10}}$

Example 3

- 1463_8 to decimal number.

Solution:

$= (1x8^3) + (4x8^2) + (6x8^1) + (3x8^0)$

=

$= \mathbf{860_{10}}$

Example 4

- 63_8 to decimal number.

Solution:

$= (6x8^1) + (3x8^0)$

=

$= \mathbf{51_{10}}$

Convert **DECIMAL** to **OCTAL** number:

Example 1

- 179_{10} to octal number.

Solution:

= Divide 179 by 8 until the small answer you get. The final answer of the division cannot be 0.

$$\frac{179}{8} = 22 \longrightarrow 3$$

$$\frac{22}{8} = 2 \longrightarrow 6$$

$$\frac{2}{8} = 0 \longrightarrow 2$$

$= \mathbf{263_8}$

Example 2

- 896_{10} to octal number.

Solution:

= Divide 896 by 8 until the small answer you get. The final answer of the division cannot be 0.

$= \mathbf{????_8}$

Example 3

- 74_{10} to octal number.

Solution:

= Divide 74 by 8 until the small answer you get. The final answer of the division cannot be 0.

$= \mathbf{112_8}$

Convert **HEXADECIMAL** to **BINARY** number:

HEXADECIMAL	BINARY
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

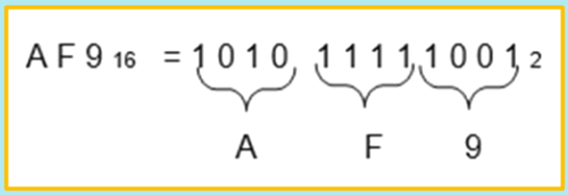
Conversion Table

Example 1

- $AF9_{16}$ to binary number.

Solution:

- Convert each hex digit to 4 binary digits according to the table given



$AF9_{16}$ = **101011111001_2**

Example 2

- $4A01_{16}$ to binary number.

Solution:

- Convert each hex digit to 4 binary digits according to the table given

= **$0100\ 1010\ 0000\ 0001_2$**

Example 3

- $EC32_{16}$ to binary number.

Solution:

- Convert each hex digit to 4 binary digits according to the table given

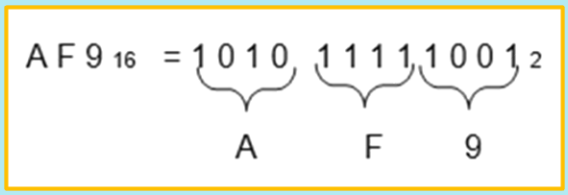
= **$????_2$**

Example 1

- $AF9_{16}$ to binary number.

Solution:

- Convert each hex digit to 4 binary digits according to the table given



$AF9_{16}$ = **101011111001_2**

Example 2

- $4A01_{16}$ to binary number.

Solution:

- Convert each hex digit to 4 binary digits according to the table given

= **$0100\ 1010\ 0000\ 0001_2$**

Example 3

- $EC32_{16}$ to binary number.

Solution:

- Convert each hex digit to 4 binary digits according to the table given

= **$????_2$**

Convert **BINARY** to **HEXADECIMAL** number:

Example 1

- $0100\ 1011\ 1111_2$ to hexadecimal number.

Solution:

- Convert each digit of binary number to hexadecimal digit according to the table given

= $\mathbf{4BF_{16}}$

Example 2

- $0110\ 0111\ 1001_2$ to hexadecimal number.

Solution:

- Convert each digit of binary number to hexadecimal digit according to the table given

= $\mathbf{????_{16}}$

Example 3

- $0101\ 1100\ 1110_2$ to hexadecimal number.

Solution:

- Convert each digit of binary number to hexadecimal digit according to the table given

= $\mathbf{????_{16}}$

Example 4

- $1010\ 0110\ 1111_2$ to hexadecimal number.

Solution:

- Convert each digit of binary number to hexadecimal digit according to the table given

= $\mathbf{????_{16}}$

Example 5

- 0111 1100 1010_2 to hexadecimal number.

Solution:

- Convert each digit of binary number to hexadecimal digit according to the table given

= **$7CA_{16}$**

Example 6

- 0100 1101 0110_2 to binary number.

Solution:

- Convert each digit of binary number to hexadecimal digit according to the table given

= **$????_{16}$**

Convert **HEXADECIMAL** to **DECIMAL** number:

HEXADECIMAL	DECIMAL
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
D	13
E	14
F	15

Conversion Table

Example 1

- $7DE_{16}$ to decimal number.

Solution:

D= 13, E= 14

$= (7x16^2) + (13x16^1) + (14x16^0)$

=

$= \mathbf{2014}_{10}$

Example 2

- $1D9_{16}$ to decimal number.

Solution:

D= 13

$= (1x16^2) + (13x16^1) + (9x16^0)$

=

$= \mathbf{????}_{\mathbf{10}}$

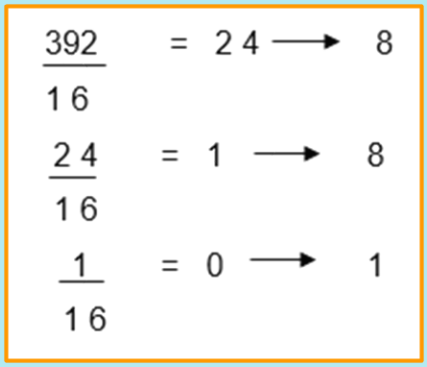
Convert **DECIMAL** to **HEXADECIMAL** number:

Example 1

❑ 392_{10} to hexadecimal number.

Solution:

➢ Divide 392 by 16 until the smallest answer is 0.



= **188_{16}**

Example 2

❑ 479_{10} to hexadecimal number.

Solution:

➢ Divide 479 by 16 until the smallest answer is 0.

= **$????_{16}$**

ONE'S COMPLEMENT

- In the number system, one's complement is used to represent the negative number in 0 & 1 integers.

Example 1:

Decimal Number	Binary Number	1′ complement
0	0000	1111
6	0110	1001
10	0101	0011
12	00001100	11110011

Example 2:

Decimal Number	1′ complement
+ 5	0101
- 5	1010
+7	00000111
-7	11111000

Example 3: $-12_{10} + (8_{10})$

Generate -12 by using Two's – Complement Method	
+12 in 8 bit signed binary	00001100
1' complement	11110011
Add with 1	1
So -12 in 2'complement	11110100
+ 8	00001000
The answer is -4	11111100

Example 4: $-13_{10} + (-15_{10})$

1. Generate -11 by using Two's – Complement Method	
+13 in 8 bit signed binary	00001101
1' complement	11110010
Add with 1	1
-13 in 2'complement	**11110011**
2. Generate -19 by using Two's – Complement Method	
+15 in 8 bit signed binary	00001111
1' complement	11110000
Add with 1	1
-15 in 2'complement	11110001
Addition	
-11 in 2'complement	11110011
-19 in 2'complement	11110001
	111100100
	Discarded(1)11100100
Answer	**11100100**

TWO'S COMPLEMENT IN ADDITION OPERATIONS

Example 1: -13_{10} - (-19_{10})

1. Generate -11 by using Two's – Complement Method	
+13 in 8 bit signed binary	00001101
1' complement	11110010
Add with 1	1
-13 in 2'complement	**11110011**
2. Generate -19 by using Two's – Complement Method	
+15 in 8 bit signed binary	00001111
1' complement	11110000
Add with 1	1
-15 in 2'complement	11110001
Addition	
-11 in 2'complement	11110011
-19 in 2'complement	11110001
	111100100
	Discarded(1)11100100
Answer	**11100100**

BCD 8421 CODES

❖ In computing and electronic system, BCD (binary coded decimal) is a class of binary encodings of decimal numbers where each decimal digit represented by fixed number of bits.

DECIMAL	BCD 8421 CODES
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

ASCII

❖ASCII is defined as **American Standard Code for Information Interchange**

❖Is the most common format for text files in computer.

❖In ASCII file, each alphabetic, numeric or special character is represented with a 7-bit binary number to represent 128 unique characters.

Exercise:

1. Define numbering system
2. List THREE (3) types of numbering system
3. Convert the following BCD codes to binary numbers:
   i. 00111000_{BCD}
   ii. 01001001_{BCD}
4. State clearly the THREE (3) main features of octal
5. Write the following numbers to the number system required (display clear calculation)
   i. 1111010101.11 to hexadecimal number
   ii. 745.23_8 to decimal number
6. Convert each hexadecimal to decimal number
   i. $F8E6_{16}$
   ii. $1E.52B_{16}$
7. Convert each hexadecimal to binary number
   i. $A2D03_{16}$
   ii. $2F.52_{16}$
8. Determine the following calculations by using 2's complement method
   i. 48 - 23
   ii. -48 - 23
   iii. 23 - 48

Answer:

1. Number system are the technique to represent numbers in the computer system architecture, every value that you are saving into or getting from computer memory has a defined system.

2. 1.Decimal
   2.Binary
   3.Octal
   4.Decimal
   5.Hexadecimal

3. Convert the following BCD codes to binary numbers:

   i. 00111000_{BCD}

   0011 1000

   3 8

   So, 38_{10}

   = $\mathbf{100110_2}$

   ii. 01001001_{BCD}

   0100 1001

   4 9

   So, 49_{10}

   = $\mathbf{110001_2}$

4. (i) The octal number system provides a convenient way to express binary numbers and codes.

   (ii) The octal number system is composed of eight digit, which are 0, 1,2,3,4,5,6,7

Answer:

(iii) Counting in octal is similar to counting in decimal, except that the digits 8 and 9 are not used.

5.

i. 1111010101.11 to hexadecimal number

$11\ 1101\ 0101.\ 1100_2$

3 D 5 . C_{16}

ii. 745.23_8 to decimal number

$= (7x8^2) + (4x8^1) + (5x8^0) + (2x8^{-1}) + (3x8^{-2})$

$= 448 + 32 + 5 + 0.25 + 0.046$

$=$ **485.297_{10}**

6.

i. $F8E6_{16}$

$= (Fx16^3) + (8x16^2) + (Ex16^1) + (6x16^0)$

$= (15x16^3) + (8x16^2) + (14x16^1) + (6x16^0)$

$=$ **63718_{10}**

ii. $1E.52B_{16}$

$= (1x16^1) + (Ex16^0) + (5x16^{-1}) + (2x16^{-2}) + (Bx16^{-3})$

$= (11x16^1) + (14x16^0) + (5x16^{-1}) + (2x16^{-2}) + (11x16^{-3})$

$=$ **30.323_{10}**

7.

i. $A2D03_{16}$

A= 1010

2= 0010

D= 0000

3= 0011

= **1010001000000011_2**

ii. $2F.52_{16}$

2= 0010

F= 1111

5= 0101

2= 0010

= **0010111101010010_2**

8. Determine the following calculations by using 2's complement method

i. 48 – 23

+48 = 00110000 (2's complement)

-23 = 11101001 (2's complement)

+48 00110000

-23 +11101001

Discard  **1**00011001 **+25**

ii. -48 - 23

-48 = 11010000 (2's complement)

-23 = 11101001 (2's complement)

-48 11010000

-23 +11101001

Discard  **1**10111001 **-71**

iii. 23 – 48

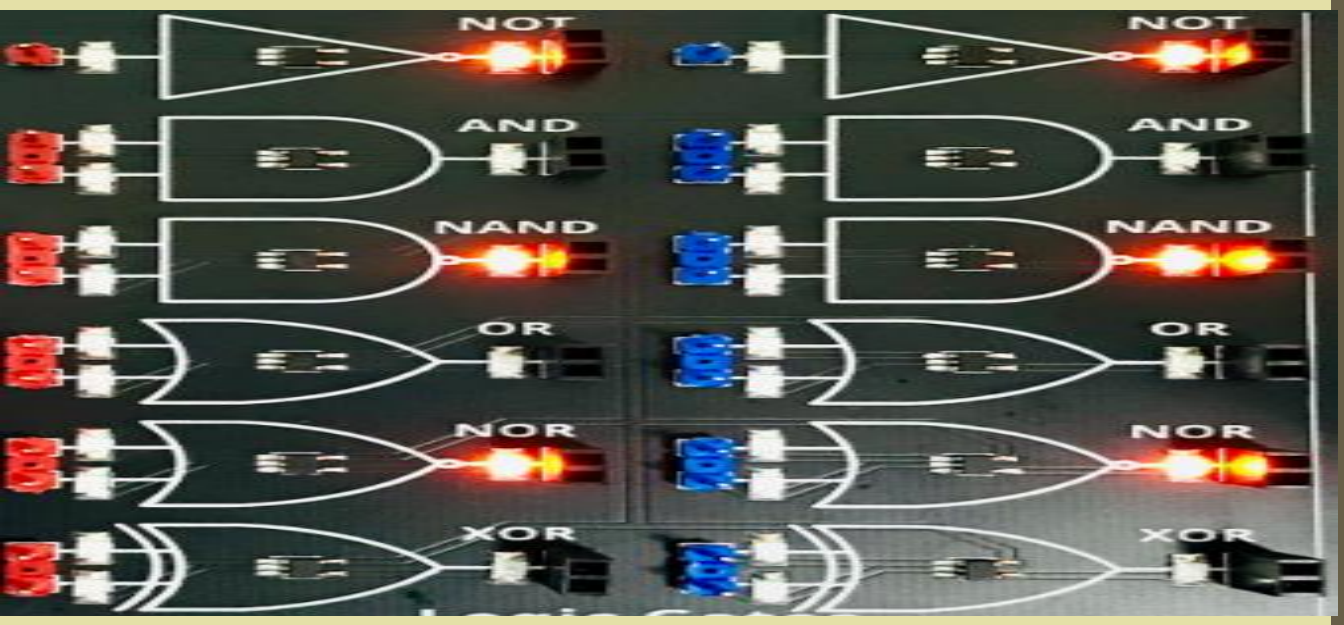
-48 = 11010000 (2's complement)

+23 = 00010111 (2's complement)

-48 11010000

+23 +00010111

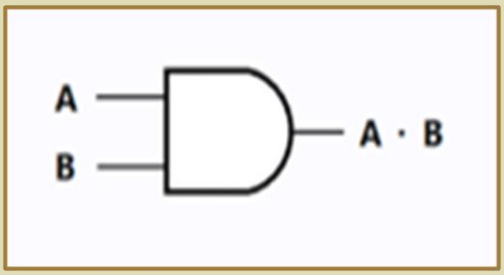
11100111 **-25**



BOOLEAN OPERATIONS

Boolean Operations

AND GATE



*Boolean Expression

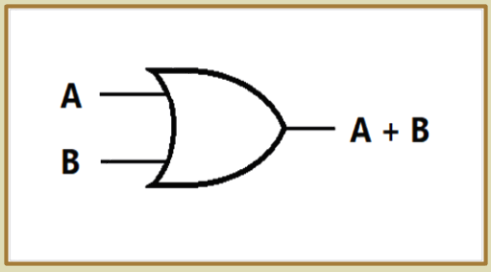
Q= A.B

- If **BOTH A AND B** are **TRUE**, then **Q** is **TRUE**.
  - Q= output
  - True= 1 or High
  - False= 0 or Low

*Truth Table

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

OR GATE



*Boolean Expression

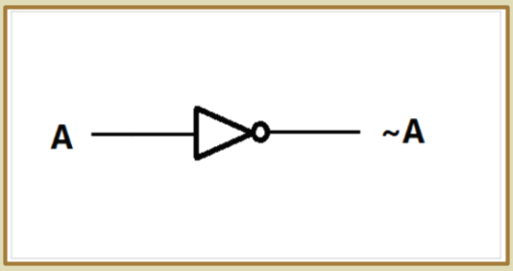
Q= A+B

- If **EITHER A OR B** is **TRUE**, then **Q** is **TRUE**.
  - Q= output
  - True= 1 or High
  - False= 0 or Low

*Truth Table

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

NOT GATE



*Boolean Expression

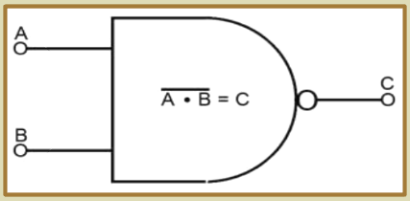
Q= not A or $\overline{A}$

- If **A is TRUE,** then **Q** is **FALSE.**
  - Q= output
  - True= 1 or High
  - False= 0 or Low

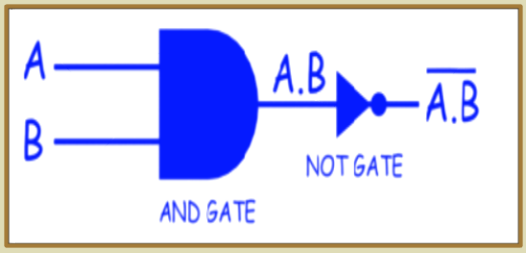
*Truth Table

A	Q
0	1
1	0

NAND GATE



OR



*Boolean Expression

Q= $\overline{A.B}$

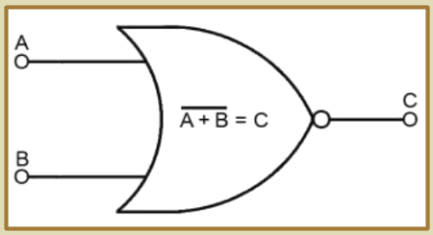
NAND gate is the **OPOSITE** of **AND gate**.

- If **EITHER A OR B are NOT TRUE,** then **Q** is **TRUE.**
  - Q= output
  - True= 1 or High
  - False= 0 or Low

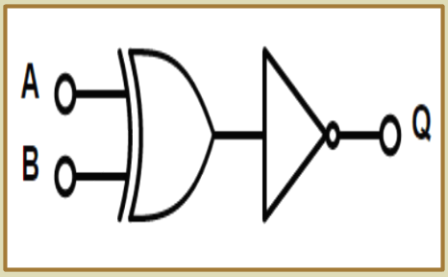
*Truth Table

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

NOR GATE



OR



*Boolean Expression

$$Q = \overline{A+B}$$

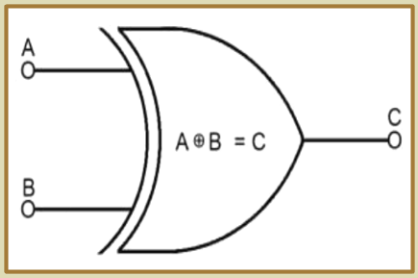
NOR gate is the **OPPOSITE** of **OR gate**.

- If **BOTH A OR B are NOT TRUE,** then **Q** is **TRUE**.
  - Q= output
  - True= 1 or High
  - False= 0 or Low

*Truth Table

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

EXCLUSIVE OR GATE



*Boolean Expression

$$Q = A+B$$

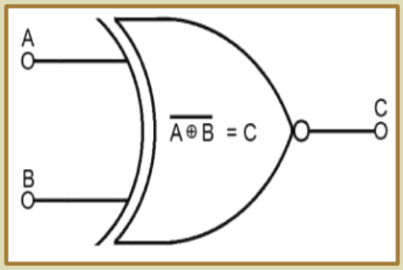
The **OUTPUT** of an Exclusive OR gate **ONLY** goes **HIGH,** when **BOTH** inputs are **DIFFERENT.**

- ✓ Q= output
- ✓ True= 1 or High
- ✓ False= 0 or Low

*Truth Table

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

EXCLUSIVE NOR GATE



*Boolean Expression

$$Q = \overline{A+B}$$

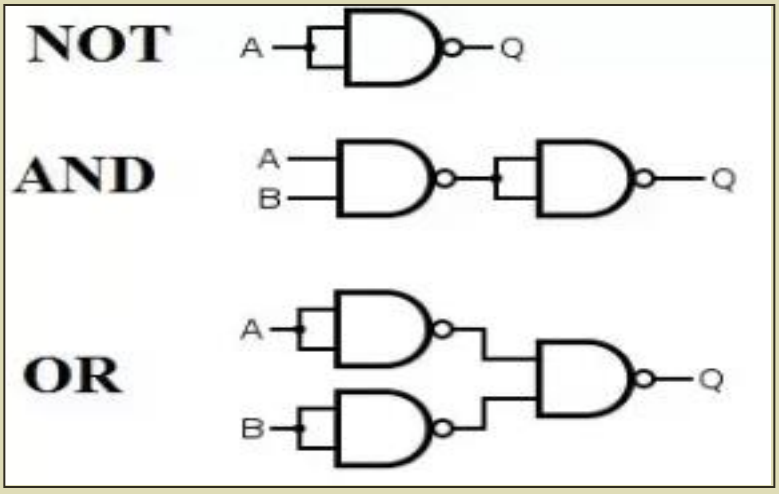
When BOTH A and B are the SAME, Q will be TRUE.

- Q= output
- True= 1 or High
- False= 0 or Low

*Truth Table

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

Construct AND, OR and NOT gates using only NAND gates



Boolean Laws

COMMUNICATIVE LAW

Commutative Law states that the interchanging of the order of operands in a Boolean equation does not change its result.

For example:

Addition (OR gate)

A+B= B+A

0+0= 0

0+1= 1

1+0= 1

1+1= 1

Multiplication (AND gate)

A*B= B*A

0*0= 0

0*1= 0

1*0= 0

1*1= 1

ASSOCIATIVE LAW

Associative Law of multiplication states that the AND operation are done on two or more than two variables.

For example:

A * (B * C) = (A * B) * C

DISTRIBUTIVE LAW

Distributive Law states that the multiplication of two variables and adding the result with a variable will result in the same value as multiplication of addition of the variable with individual variables.

For example:

A + BC = (A + B) (A + C)

DE MORGAN'S THEOREM

DeMorgan's Theorems are two additional simplification techniques that can be used to simplify Boolean expressions. Again, the simpler the Boolean expression the simpler the resulting the Boolean expression, the simpler the resulting logic.

For example:

$$\overline{A+B} = \overline{A} \cdot \overline{B}$$

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

INVOLUTION RULES

The operation of complement in Boolean algebras is an involution. Generally in non-classical logics, negation that satisfies the law of double negation is called involutive.

For example:

$(X')' = X$

Sum of Product (SOP)

- The ***Sum of Product*** **(SOP)** expression comes from the fact that **two or more products (AND) are summed (OR) together.**

- That is the outputs **from two or more AND gates are connected to the input of an OR gate so that they are effectively OR'ed together to create the final AND-OR logical output.**

For example, the following Boolean function is a typical **sum-of-product expression**:

Sum of Product Expressions:

Q = (A.B) + (B.C) + (A.1)
and also
(A.B.C) + (A.C) + (B.C)

- However, Boolean functions can also be expressed in nonstandard sum of products forms like that shown below but they can be converted to a standard SOP form by expanding the expression. So:

$$Q = A.B(C + C) + ABC$$

Becomes in sum-of-product terms:

$$Q = A.B.C + A.B.C + ABC$$

- We can now draw up the truth table for the above expression to show a list of all the possible input combinations for A, B and C which will result in an output "1".

Inputs			Output	Product
C	B	A	Q	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$A.B.\overline{C}$
1	0	0	0	
1	0	1	1	$A.\overline{B}.C$
1	1	0	1	$\overline{A}.B.C$
1	1	1	0	

Sum of Product Truth Table Form

- Then we can clearly see from the truth table that each product row which produces a "1" for its output corresponds to its Boolean multiplication expression with all of the other rows having a "0" output as a "1" is always outputted from an OR gate.

- Clearly the advantage here is that the truth table gives us a visual indication of the Boolean expression allowing us to simplify the expression.

Product of Sum (POS)

Sum-of-Product Example:

The following Boolean Algebra expression is given as:

$$Q = A(BC + BC + BC) + ABC$$

1. Convert this logical equation into an equivalent SOP term.
2. Use a truth table to show all the possible combinations of input conditions that will produces an output.
3. Draw a logic gate diagram for the expression.

Solution:

1. Convert to SOP term

$$Q = A.B.C + A.B.C + A.B.C + A.B.C$$

2. Truth Table

Inputs			Output	Product
C	B	A	Q	
0	0	0	0	
0	0	1	0	
0	1	0	1	$\overline{A}.B.\overline{C}$
0	1	1	0	
1	0	0	1	$\overline{A}.\overline{B}.C$
1	0	1	0	
1	1	0	1	$\overline{A}.B.C$
1	1	1	1	A.B.C

Sum of Product Truth Table Form

- **Product of Sum** expressions are Boolean expressions **made up of sums consisting of one or more variables, either in its normal true form or complemented form or combinations of both, which are then AND'ed together.**

- For example, the following Boolean function is a typical **product-of-sum expression:**

Product of Sum Expressions

$$Q = (A + B).(B + C).(A + 1)$$

and also

$$A + B + C).(A + C).(B + C)$$

- However, Boolean functions can also be expressed in nonstandard product of sum forms like that shown below but they can be converted to a standard POS form by using the distributive law to expand the expression with respect to the sum. Therefore:

$$Q = A + (BC)$$

Becomes in expanded product-of-sum terms:

$$Q = (A + B)(A + C)$$

Another nonstandard example is:

$$Q = (A + B) + (A.C)$$

Becomes as an expanded product-of-sum expession:

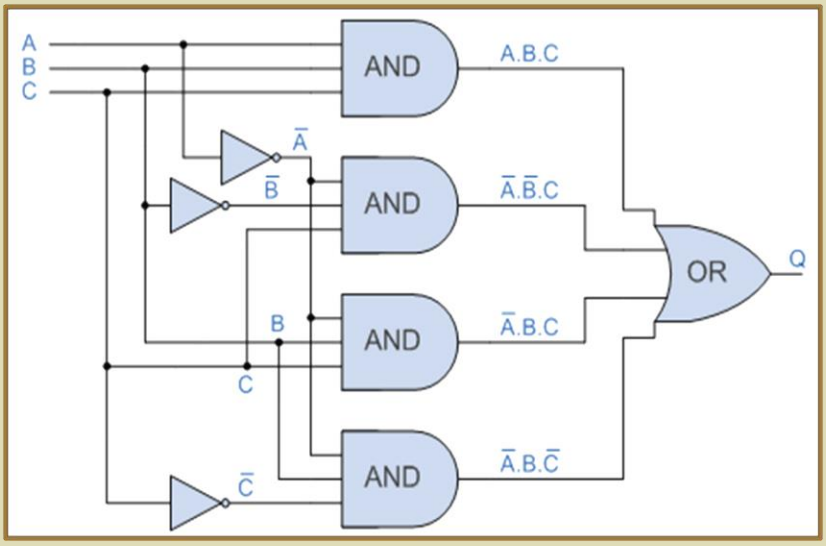
$$Q = (A + B + B)(A + B + C)$$

- which can, if required be reduced using *complement law* and *annulment law)* too:

$$Q = (A + 1)(A + B + C)$$

$$Q = A + B + C$$

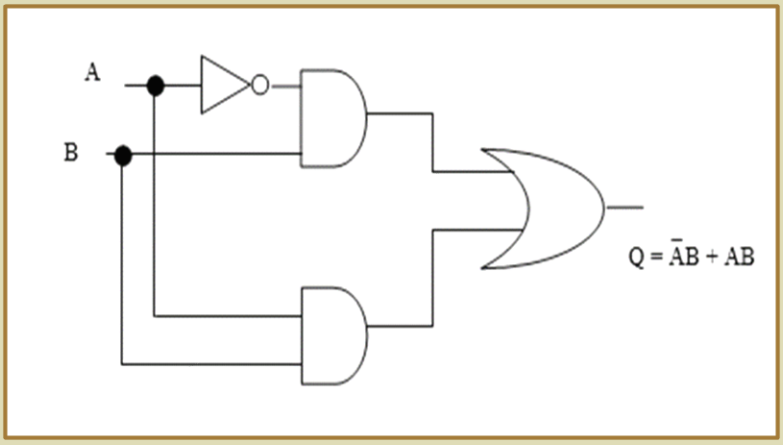
3. Logic Gate SOP Diagram



Construct combinational logic circuits from Boolean Expression:

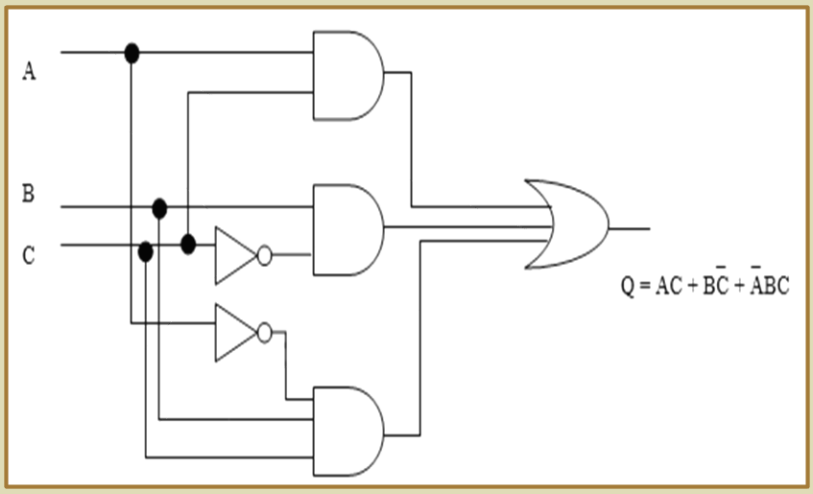
Example 1

$$Q = \bar{A}B + AB$$



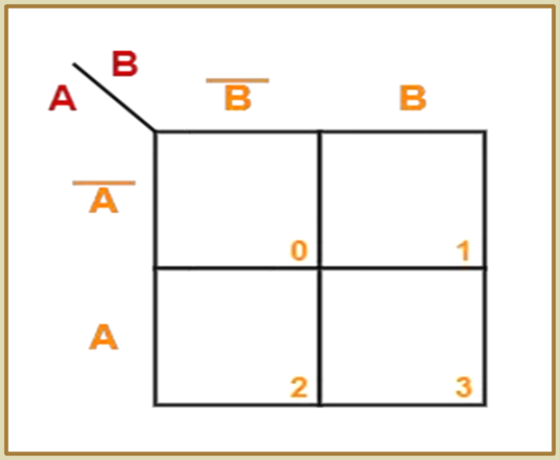
Example 2

$$Q = AC + B\bar{C} + \bar{A}BC$$

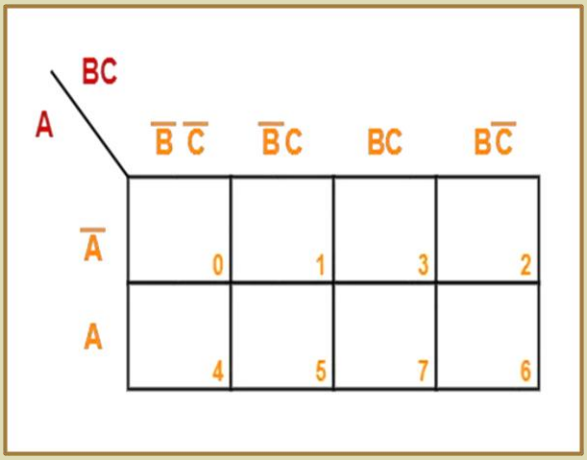


Karnaugh Map

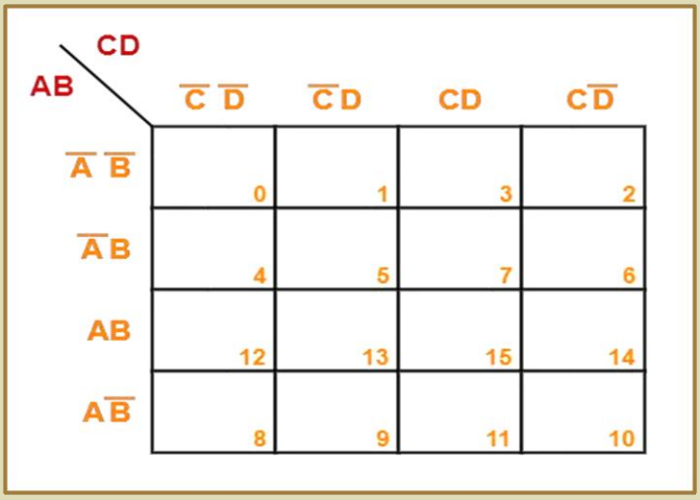
- A Karnaugh map is a method used to simplify the Boolean expressions.
- It resulted less numbers of logic gates and inputs to be used.
- Karnaugh map of 2 to 4 variables are given as below.



Example of Karnaugh Map for 2 variables



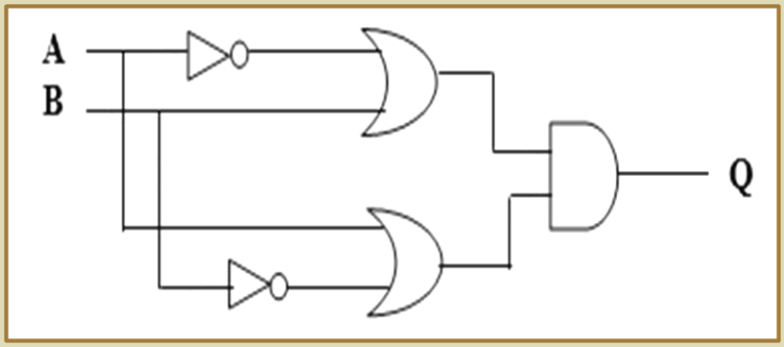
Example of Karnaugh Map for 3 variables



Example of Karnaugh Map for 4 variables

Simplify combinational logic circuits using:
a. Boolean Laws
b. Karnaugh Map

Example 1



Solution

a. Boolean Laws

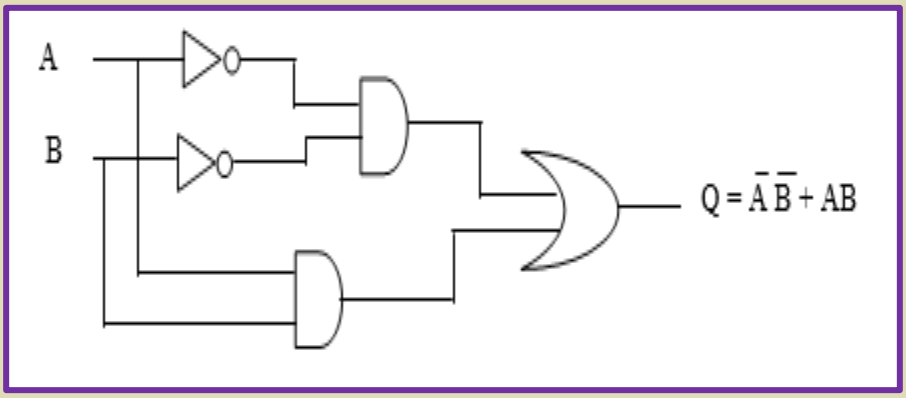
$$Q = (\bar{A} + B)(A + \bar{B})$$

$$Q = \bar{A}A + \bar{A}\bar{B} + BA + B\bar{B}$$

$$Q = \bar{A}\,\bar{B} + AB$$

b. Karnaugh Map

$\bar{A}$ \ $\bar{B}$	$\bar{B}$	B
$\bar{A}$	1	0
A	0	1



Example 2

$$Q = A\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}C + \bar{A}\bar{B}\bar{C}$$

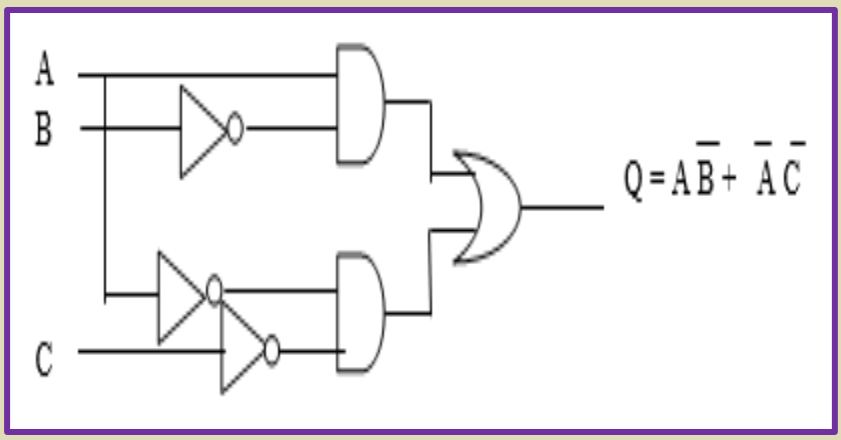
Solutions

a. Boolean Laws

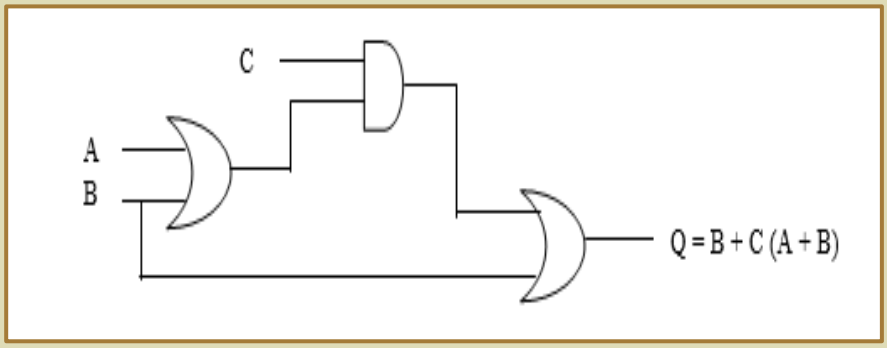
$$\begin{aligned} Q &= A\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}C + A\bar{B}\bar{C} \\ &= A\bar{B}\bar{C} + A\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} \\ &= A\bar{B}(\bar{C} + C) + \bar{A}\bar{C}(B + \bar{B}) \\ &= A\bar{B}(1) + \bar{A}\bar{C}(1) \\ Q &= A\bar{B} + \bar{A}\bar{C} \end{aligned}$$

b. Karnaugh Map

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1	0	0	1
A	1	1	0	0



Example 3



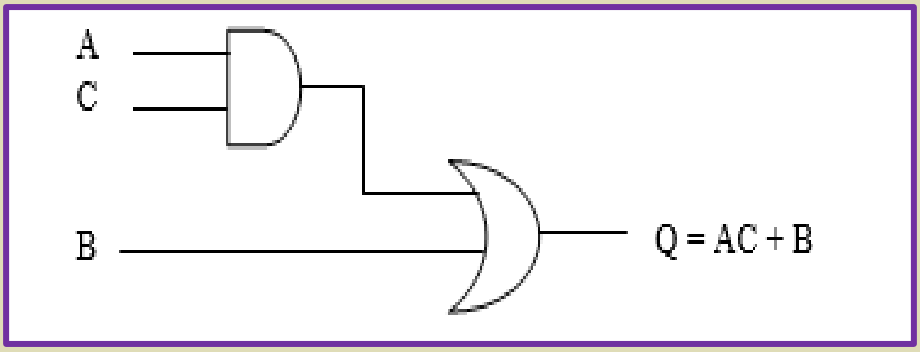
Solutions

a. Boolean Laws

$$
\begin{aligned}
Q &= B + C\,(A + B) \\
&= B + AC + BC \\
&= AC + B + BC \\
&= AC + B\,(1 + C) \\
&= AC + B\,(1) \\
&= AC + B
\end{aligned}
$$

b. Karnaugh Map

Please try to find the answer ☺

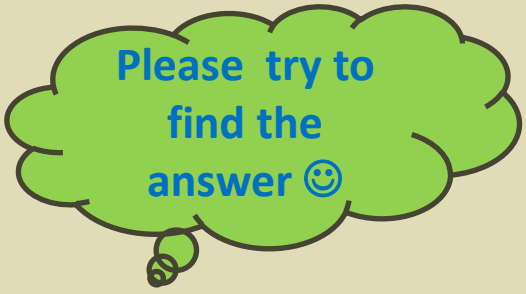


Example 4

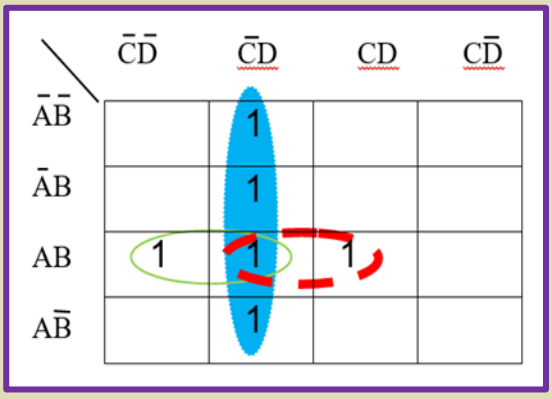
$$Q = AB\bar{C}D + AB\bar{C}\bar{D} + ABCD + \bar{A}B\bar{C}D + \bar{A}\bar{B}\bar{C}D + A\bar{B}\bar{C}D$$

Solutions

a. Boolean Laws



b. Karnaugh Map



$$Q = ABD + AB\bar{C} + C\bar{D}$$

Exercises

1. i. Simplify the following Boolean equations

$$Y = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + A\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

ii. Draw corresponding logic circuits

2. Simplify the following Boolean expression using Karnaugh Map

$$Y = \overline{A}BC + \overline{A}\,\overline{B}\,\overline{C} + ABC + A\overline{B}C + A\overline{B}\,\overline{C}$$

3. i. Express the Boolean expression for three input EX-OR (X-OR) gate.

ii. Write the truth table

iii. Draw the symbol

4. Solve the expression below using Karnaugh Map

$$Y = \overline{A}BC + \overline{A}\,\overline{B}\,\overline{C} + ABC + A\overline{B}C + A\overline{B}\,\overline{C}$$

i. The expression below using Karnaugh Map

ii. Construct the truth table

iii. Draw the logic circuit

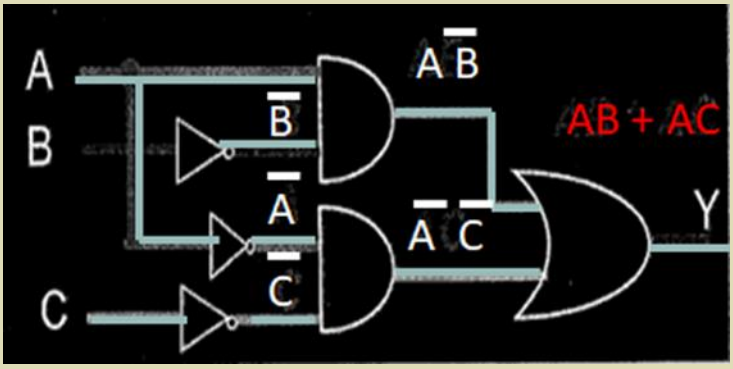
5. Deconstruct a logic circuit having the output expression X= AB + CD using NAND gate only

Solution

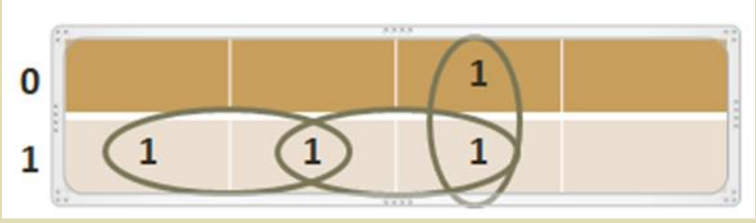
1. i.

$$
\begin{aligned}
Y &= A\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}C + \bar{A}\bar{B}\bar{C} \\
&= A\bar{B}(\bar{C} + C) + \bar{A}\bar{C}(\bar{B} + B) \\
&= A\bar{B}(1) + \bar{A}\bar{C}(1) \\
&= A\bar{B} + \bar{A}\bar{C}
\end{aligned}
$$

ii. Draw corresponding logic circuits



2.



$$X = B\bar{C} + BC + AC$$

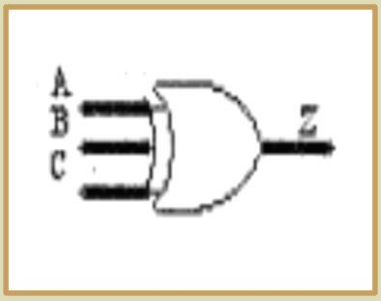
3. i.

$$Z = A \oplus B \oplus C$$

ii. Truth table

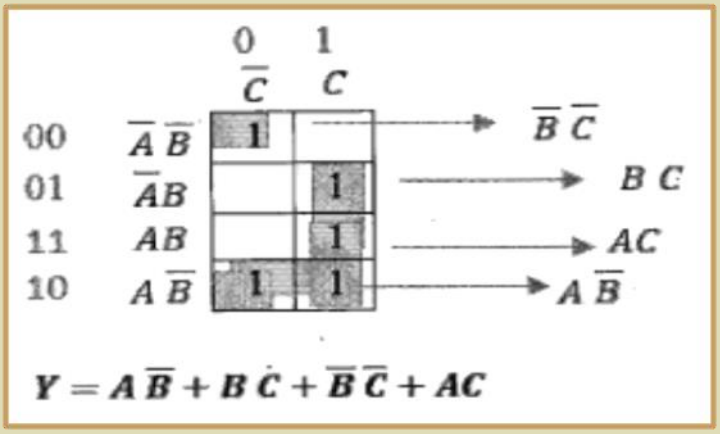
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

iii.



XOR- Exclusive OR gate

4. i. The expression below using Karnaugh Map



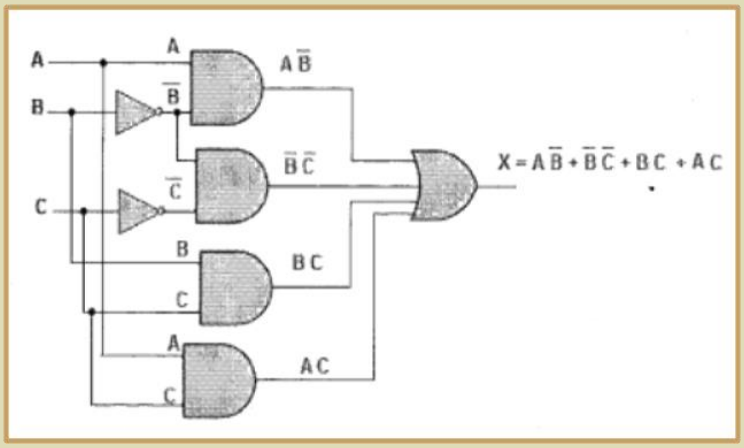
ii. Construct the truth table

$X = \overline{A}BC + \overline{A}\overline{B}\overline{C} + ABC + A\overline{B}C + A\overline{B}\overline{C}$

0 1 1 0 0 0 1 1 1 1 0 1 1 0 0

A	B	C	X	PRODUCT TERM
0	0	0	1	$\overline{A}\overline{B}\overline{C}$
0	0	1	0	
0	1	0	0	
0	1	1	1	$\overline{A}BC$
1	0	0	1	$A\overline{B}\overline{C}$
1	0	1	1	$A\overline{B}C$
1	1	0	0	
1	1	1	1	ABC

iii. Draw the logic circuit



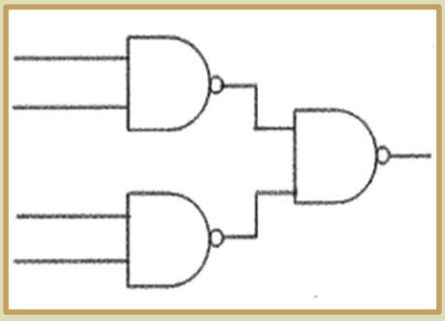
5.

A

B

C

D



SEQUENTIAL LOGICS

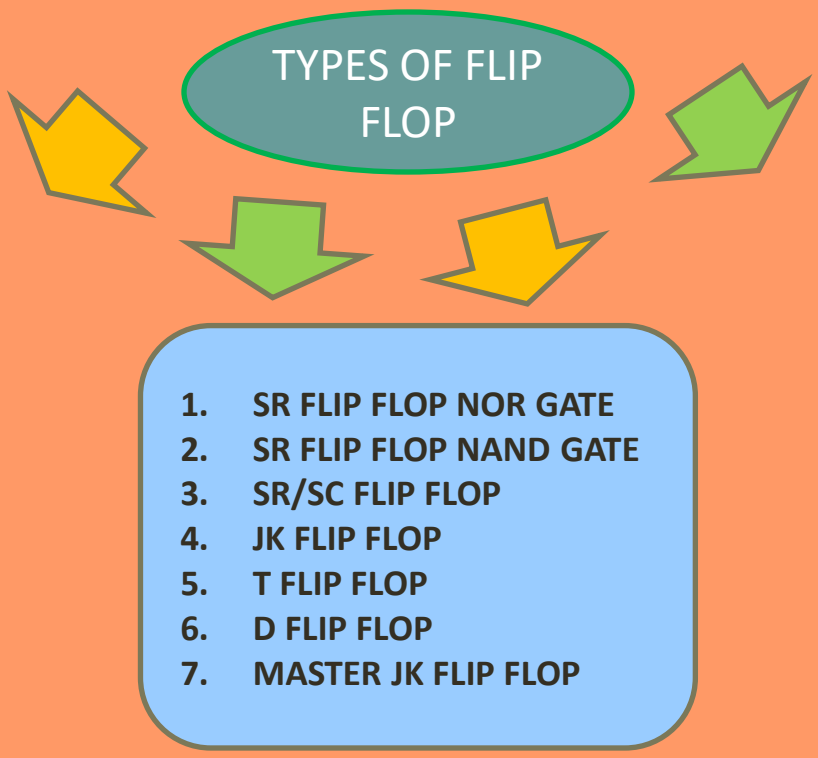
Sequential Logic

DEFINITION:

- **Sequential logic** is a form of a binary circuit design that employs one or more inputs and one or more outputs.

- Each of the inputs and outputs can either attain states:
  0 (low) or logic 1 (high).

- A simple flip flop has two stable states. The flip flop maintains its states indefinitely until an input pulse called a trigger is received. If a trigger is received, the flip flop outputs change their states according to defined rules, and remain in those states until another trigger is received.



Differences Between SR FF NOR Gate and SR FF NAND Gate

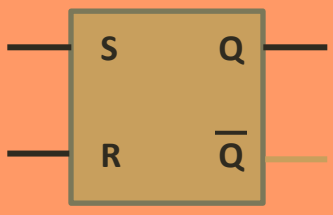
SR FLIP FLOP NOR GATE

- Used combination of **TWO NOR gate** to produce SR flip flop.
- Active **HIGH** flip flop
  0= open
  1= close
- Produced **INVALID** when BOTH S and R become **HIGH**.

SR FLIP FLOP NAND GATE

- Used combination of **TWO NAND** gate to produce SR flip flop.
- Active **LOW** flip flop
  0= close
  1= open
- Produced **NO CHANGE** when BOTH S and R become **LOW.**

SR FLIP FLOP NOR GATE



Symbol of SR flip flop NOR gate

PRESET/SET= 1
CLEAR/RESET= 0
HIGH= 1
LOW= 0
CLK= CLOCK

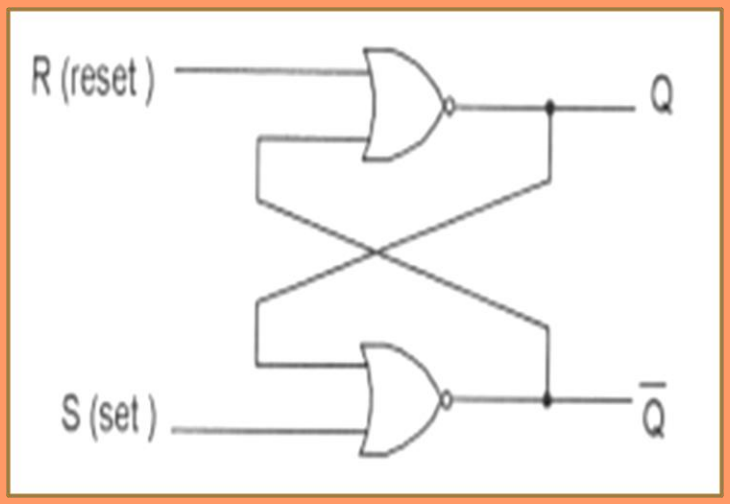
Truth Table

S	R	CLK (PGT)	Q
0	0	↑	NO CHANGE
0	1	↑	1
1	0	↑	0
1	1	↑	INVALID

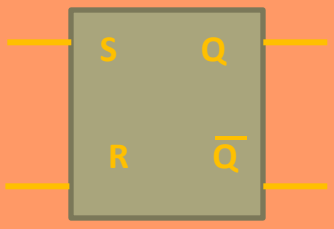
PGT- Positive Going Transition ↑

NGT- Negative Going Transition ↓

Logic Circuit of SR flip flop NOR gate



SR FLIP FLOP NAND GATE



Symbol of SR flip flop NAND gate

PRESET/SET= 1
CLEAR/RESET= 0
HIGH= 1
LOW= 0
CLK= CLOCK

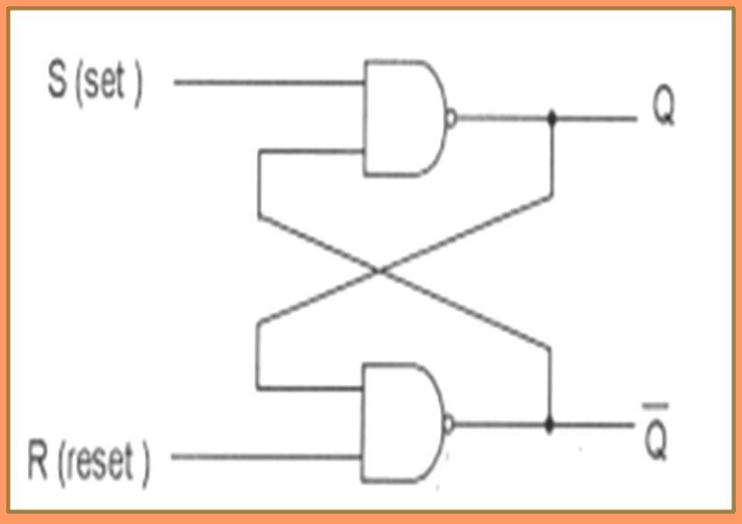
Truth Table

S	R	CLK (PGT)	Q
0	0	↑	NO CHANGE
0	1	↑	0
1	0	↑	1
1	1	↑	INVALID

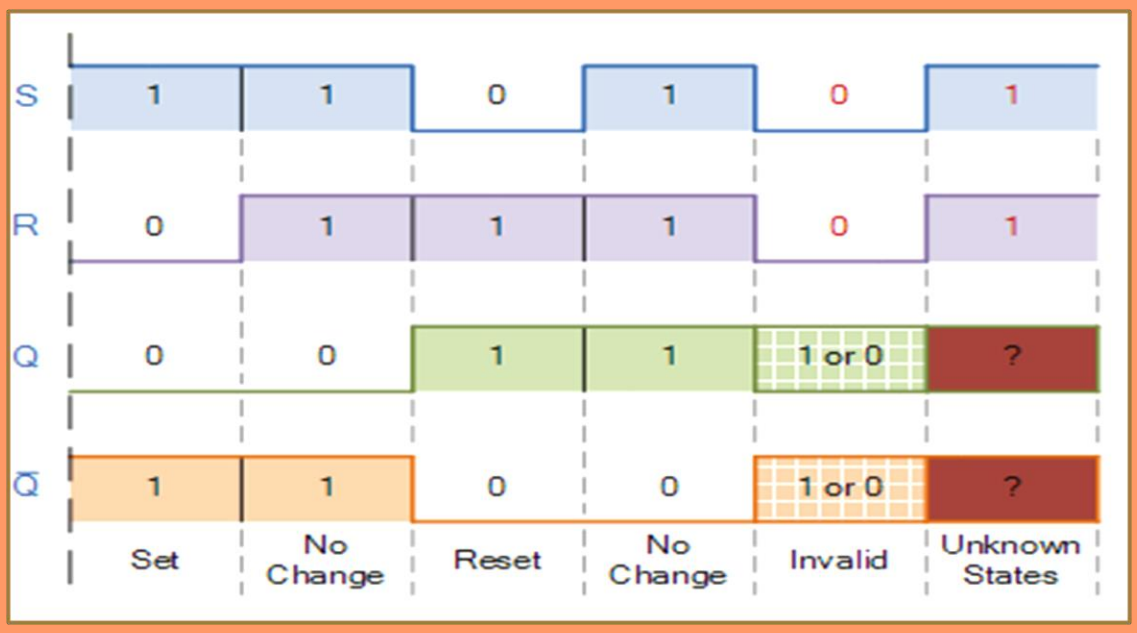
PGT- Positive Going Transition ↑

NGT- Negative Going Transition ↓

Logic Circuit of SR flip flop NAND gate

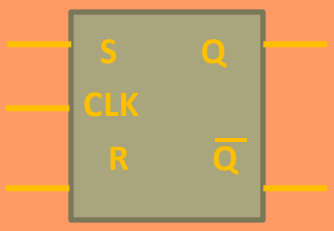


SR FLIP FLOP TIMING DIAGRAM



https://www.electronics-tutorials.ws/sequential/seq_1.html

CLOCKED SR FLIP FLOP NAND GATE



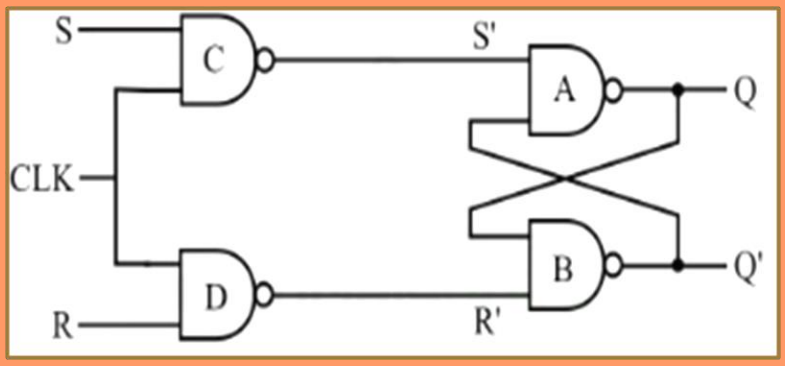
PRESET/SET= 1
CLEAR/RESET= 0
HIGH= 1
LOW= 0
CLK= CLOCK

Symbol of Clocked SR flip flop NAND gate

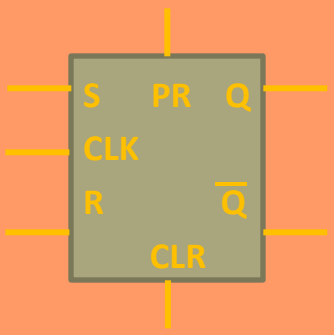
Truth Table

S	R	Q_n	Q_{n+1}	Q
0	0	0	0	NO CHANGE
0	0	1	1	NO CHANGE
0	1	0	0	RESET (0)
0	1	1	0	RESET (0)
1	0	0	1	SET
1	0	1	1	SET
1	1	0	X	FORBIDDEN
1	1	1	X	FORBIDDEN

Logic Circuit of Clocked SR flip flop NAND gate



SR flip flop PRESET and CLEAR control inputs:



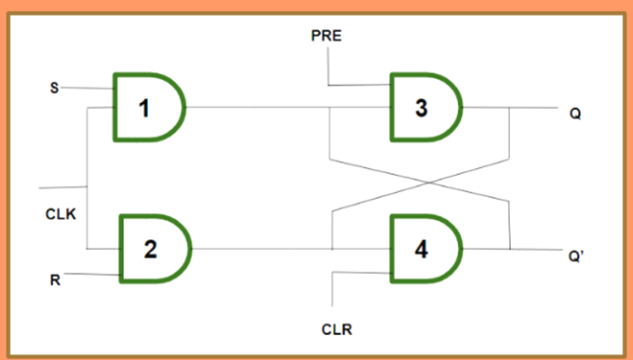
Symbol

PRESET/SET= 1
CLEAR/RESET= 0
HIGH= 1
LOW= 0
CLK= CLOCK

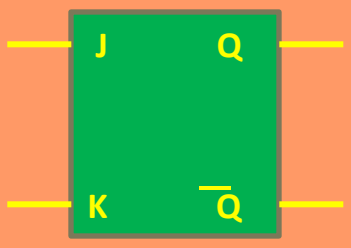
Truth Table

INPUTS					OUTPUTS		
PR	CLR	CLK	S	R	Q(n+1)	$\overline{Q}$(n+1)	Comments
0	1	NA	NA	NA	1	__ 0	SET
1	0	NA	NA	NA	0	__ 1	RESET
1	1	0	NA	NA	Q(n)	Q (n)	NO CHANGE
1	1	1	0	0	Q(n)	Q (n)	NO CHANGE
1	1	1	1	0	1	0	SET
1	1	1	0	1	0	1	RESET
1	1	1	1	1	NA	NA	FORBIDDEN

Logic Circuit of Clocked SR flip flop with PRESET and CLEAR



JK FLIP FLOP



PRESET/SET= 1
CLEAR/RESET= 0
HIGH= 1
LOW= 0
CLK= CLOCK

Symbol of JK flip flop

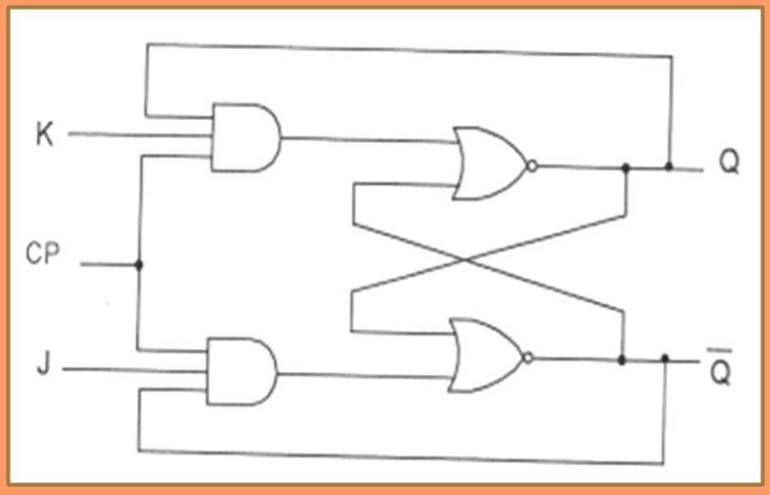
Truth Table

J	K	CLK (PGT)	Q
0	0	↑	HOLD
0	1	↑	0
1	0	↑	1
1	1	↑	TOGGLE

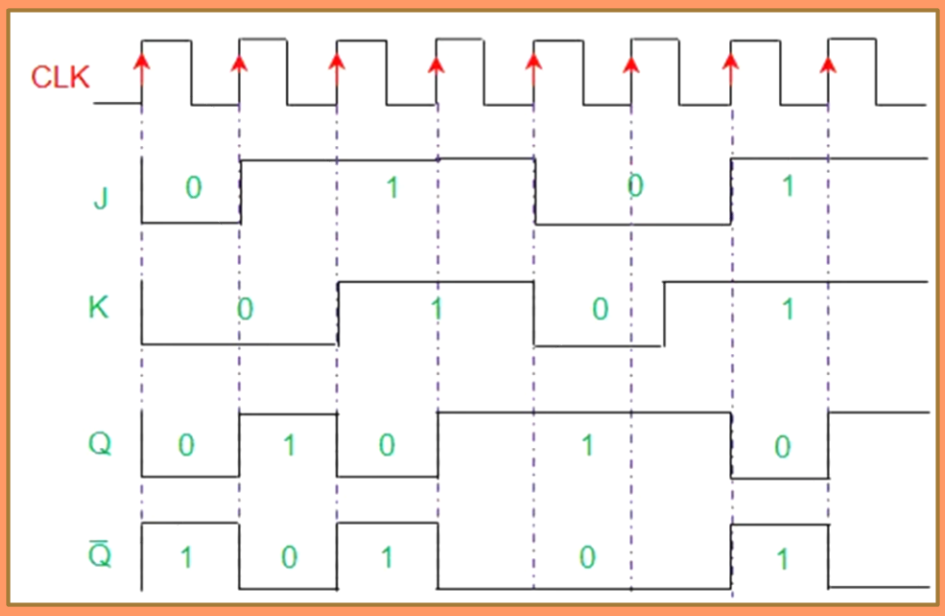
PGT- Positive Going Transition ↑

NGT- Negative Going Transition ↓

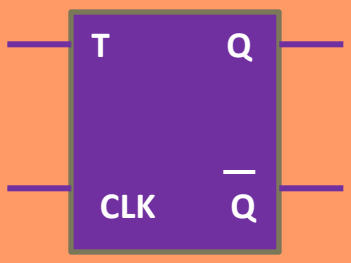
Logic Circuit of JK flip flop



JK FLIP FLOP TIMING DIAGRAM



T FLIP FLOP



PRESET/SET= 1
CLEAR/RESET= 0
HIGH= 1
LOW= 0
CLK= CLOCK

Symbol of T flip flop

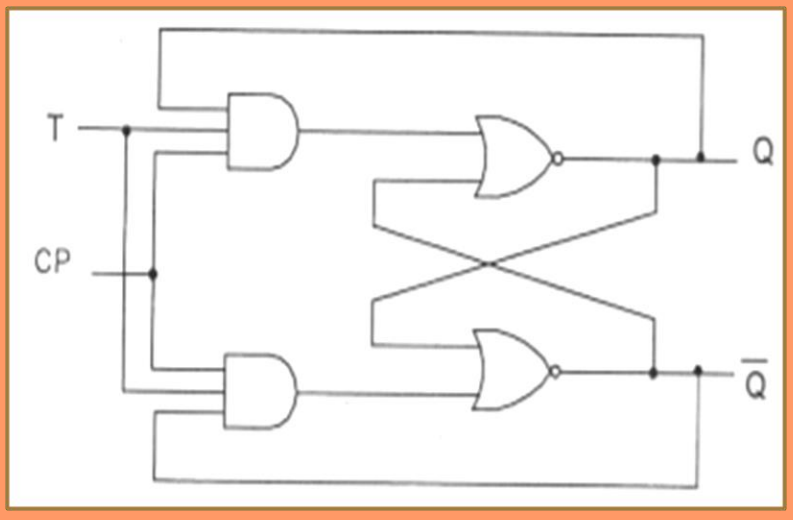
Truth Table

T	CLK (PGT)	Q
0	↑	0
1	↑	1
0	↑	0
1	↑	1

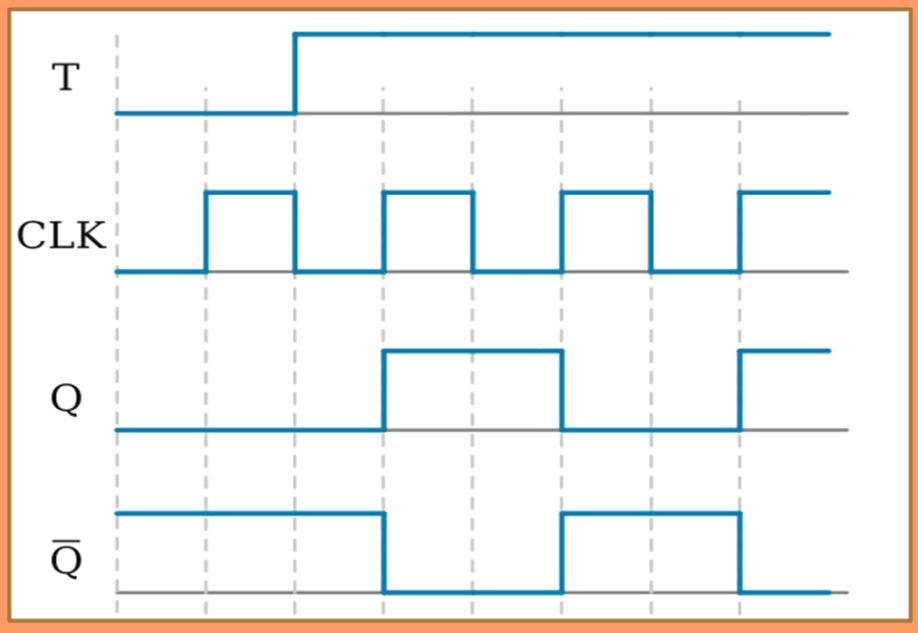
PGT- Positive Going Transition

NGT- Negative Going Transition

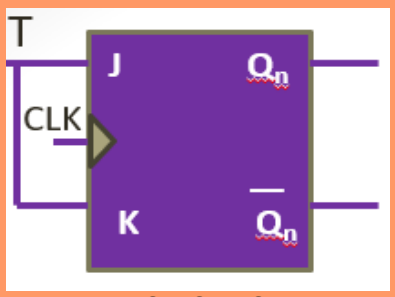
Logic Circuit of T flip flop



T FLIP FLOP TIMING DIAGRAM



Construction of a T FF from JK FF

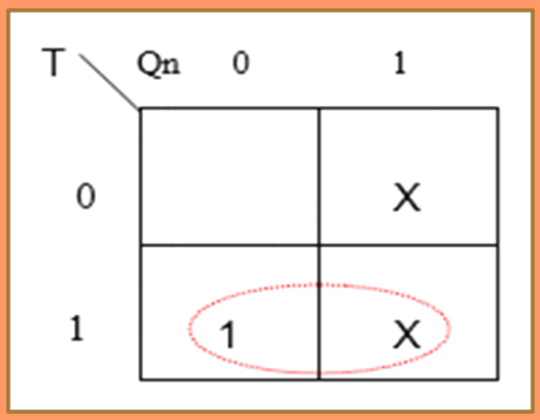


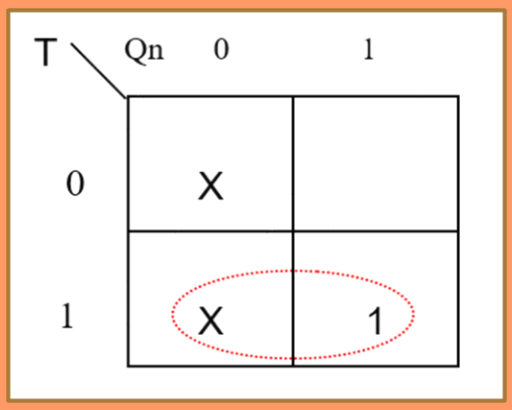
Symbol of T flip flop

Truth Table

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

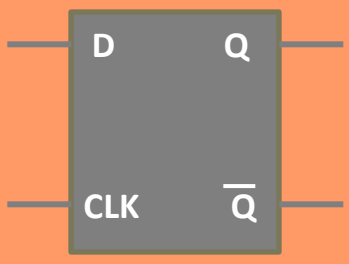
- By Karnaugh map to determine J and K Boolean expression in terms of T





J = T, K = T

D FLIP FLOP



Symbol of D flip flop

PRESET/SET= 1
CLEAR/RESET= 0
HIGH= 1
LOW= 0
CLK= CLOCK

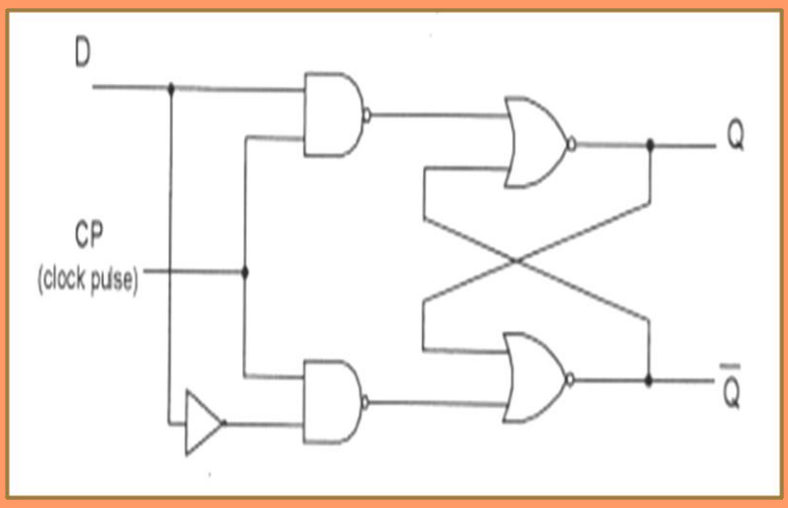
Truth Table

D	CLK (PGT)	Q
0	↑	RESET (0)
1	↑	SET (1)

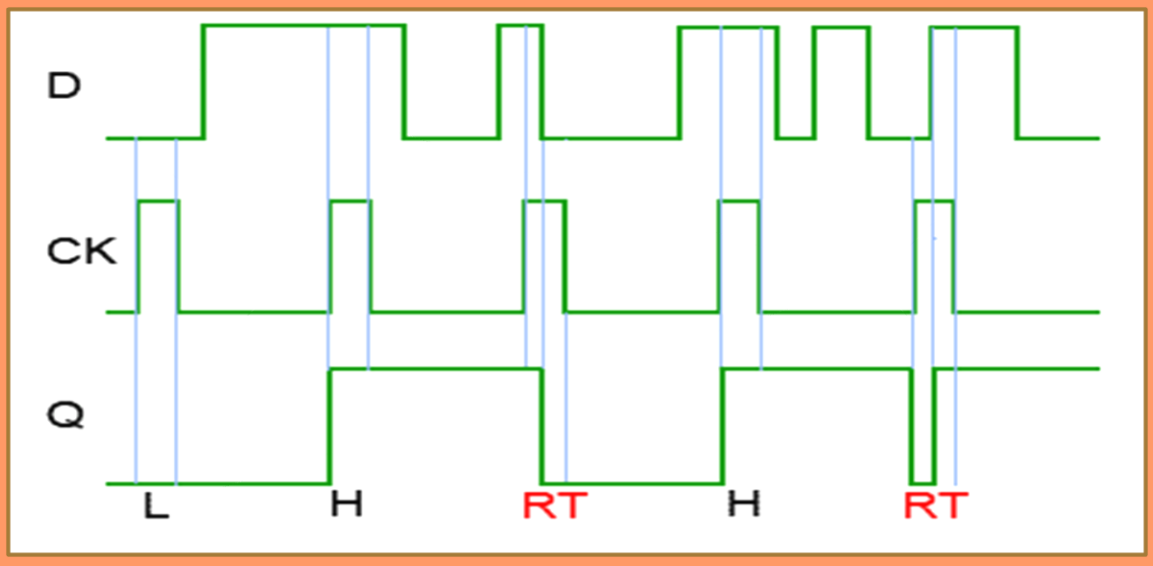
PGT- Positive Going Transition ↑

NGT- Negative Going Transition ↓

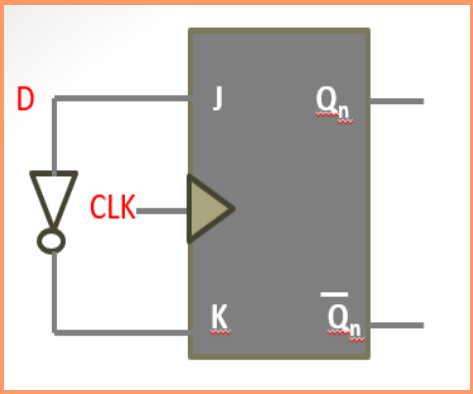
Logic Circuit of D flip flop



D FLIP FLOP TIMING DIAGRAM



Construction of a D FF from JK FF

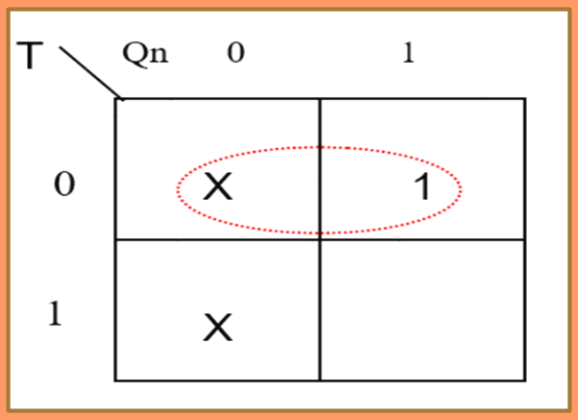


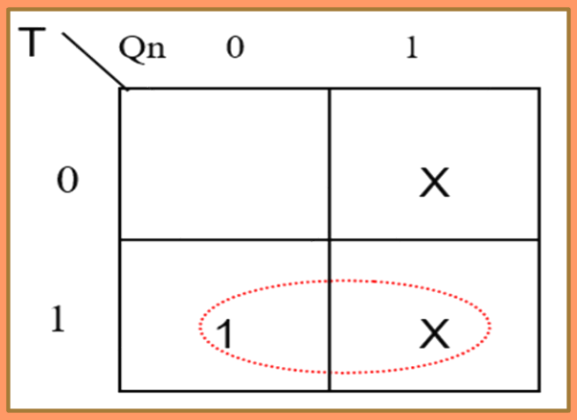
Symbol

Truth Table

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

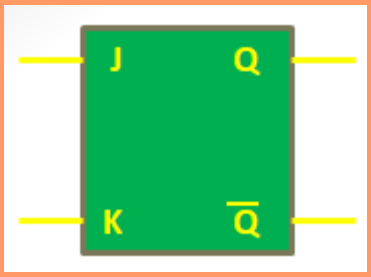
- By Karnaugh map to determine J and K Boolean expression in terms of D





$$J = D, K = \overline{D}$$

MASTER JK FLIP FLOP



Symbol of Master JK flip flop

PRESET/SET= 1
CLEAR/RESET= 0
HIGH= 1
LOW= 0
CLK= CLOCK

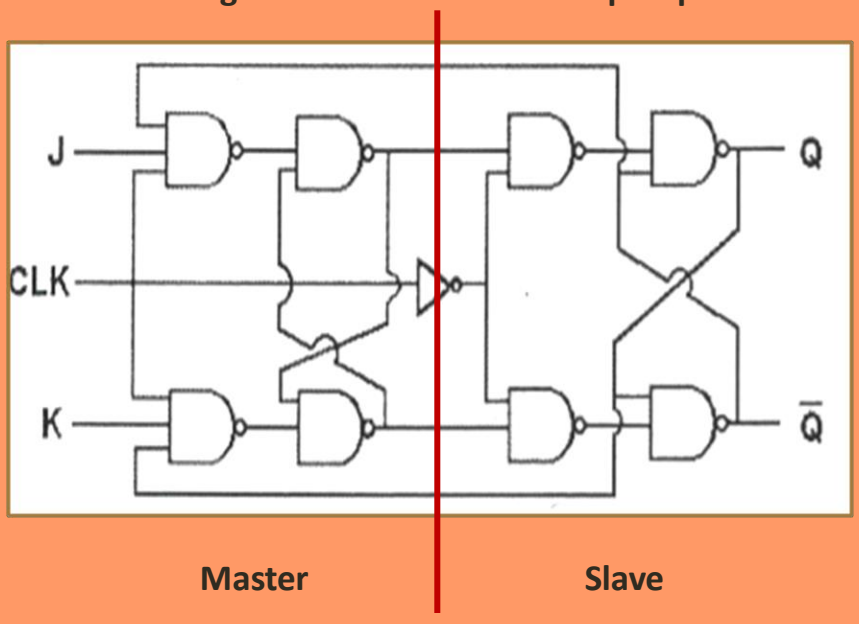
Truth Table

J	K	CLK (PGT)	Q
0	0	↑	NO CHANGE
0	1	↑	RESET (0)
1	0	↑	SET (1)
1	1	↑	TOGGLE

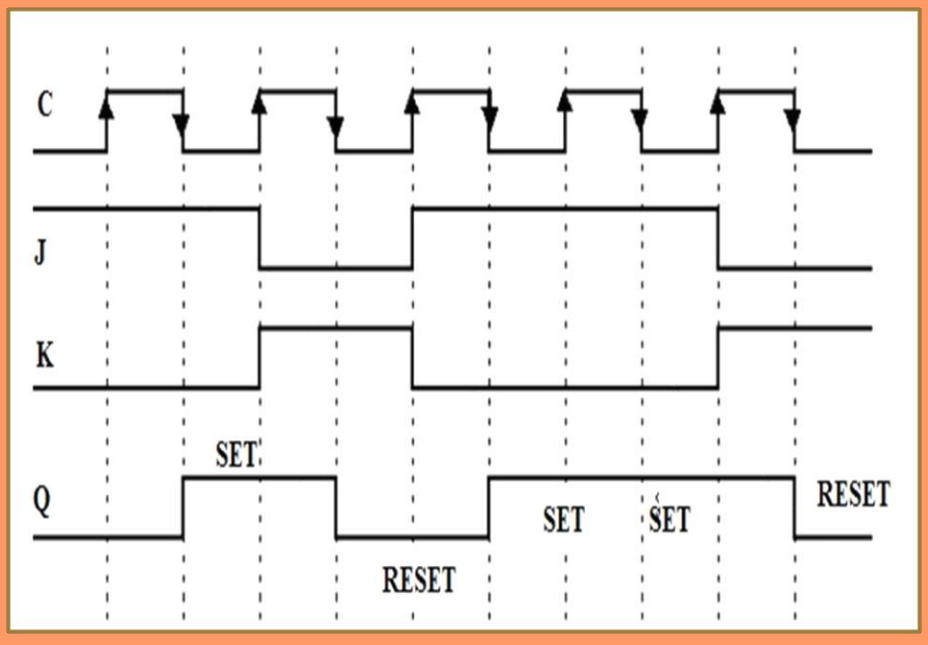
PGT- Positive Going Transition

NGT- Negative Going Transition

Logic Circuit of Master JK flip flop



MASTER JK FLIP FLOP TIMING DIAGRAM



Application of JK flip-flops, T flip-flops and D flip-flops

JK flip-flops	T flip-flops	D flip-flops
Frequency division: create a frequency divider, which is a circuit that divides the frequency of an input signal by a fixed amount. This makes it useful in real-time applications such as audio and video processing.	**Frequency division:** used to divide the frequency of a clock signal by two, making it useful in applications such as digital clocks and frequency synthesizers.	**Shift registers:** store and shift data in digital systems. commonly used in serial communication protocols such as UART, SPI, and I2C.
Synchronization: synchronize data signals between two digital circuits, ensuring that they are operating on the same clock cycle. This makes it useful in applications where timing is critical.	**Frequency multiplication:** multiply the frequency of a clock signal by two, making it useful in applications such as frequency synthesizers and digital signal processing.	**State machines:** used in digital systems to control sequences of events. State machines are commonly used in control systems, automotive applications, and industrial automation.
Counters: conjunction with other digital logic gates to create a binary counter. This makes it useful in real-time applications such as timers and clocks.	**Counters:** conjunction with other digital logic gates to create binary counters that can count up or down depending on the design. This makes them useful in real-time applications such as timers and clocks.	**Counters:** conjunction with other digital logic gates to create binary counters that can count up or down depending on the design. This makes them useful in real-time applications such as timers and clocks.
Data storage: store temporary data in digital systems.	**Data storage:** store a single bit of data, making it useful in applications such as shift registers and memory devices.	**Data storage:** store temporary data in digital systems. They are often used in conjunction with other memory elements to create more complex storage systems.
Refer https://www.electronicsforu.com/technology-trends/learn-electronics/jk-flip-flop-circuit-truth-table-limitations-applications	**Refer https://www.electronicsforu.com/technology-trends/learn-electronics/t-flip-flop-circuit-truth-table-limitations-applications**	**Refer https://www.electronicsforu.com/technology-trends/learn-electronics/d-flip-flop-circuit-truth-table-limitations-applications**

Application of JK flip-flops, T flip-flops and D flip-flops

JK flip-flops	T flip-flops	D flip-flops
https://www.electronicsforu.com/technology-trends/learn-electronics/jk-flip-flop-circuit-truth-table-limitations-applications	https://www.electronicsforu.com/technology-trends/learn-electronics/t-flip-flop-circuit-truth-table-limitations-applications	https://www.electronicsforu.com/technology-trends/learn-electronics/d-flip-flop-circuit-truth-table-limitations-applications

Exercises

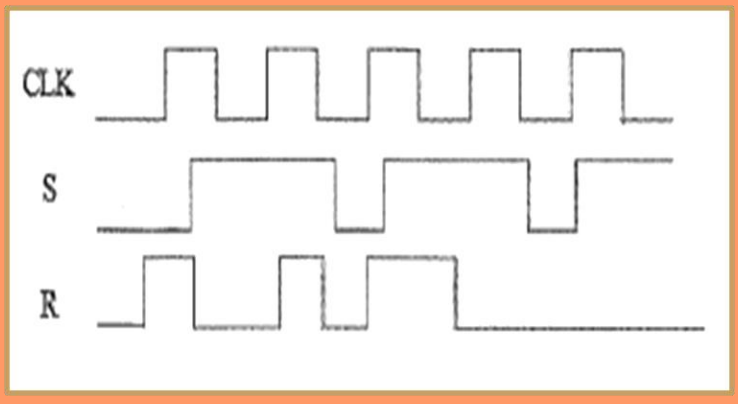
1.

i. Define the flip flop

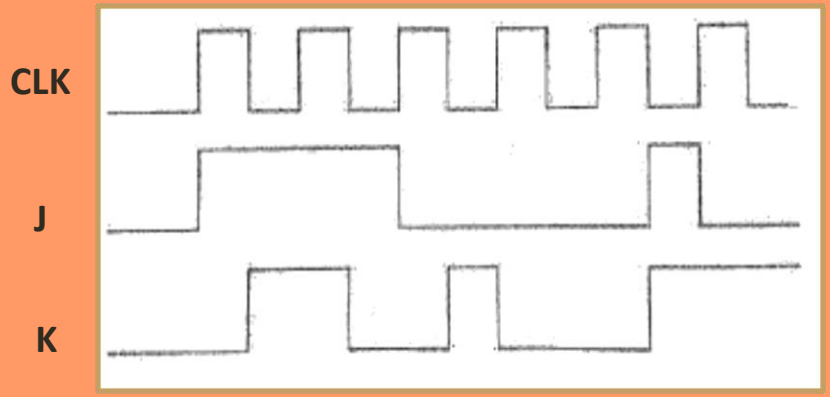
ii. State TWO (2) types of flip flop

2. Discuss THREE (3) differences between SR flip flops NOR and SR flip flop NAND gate

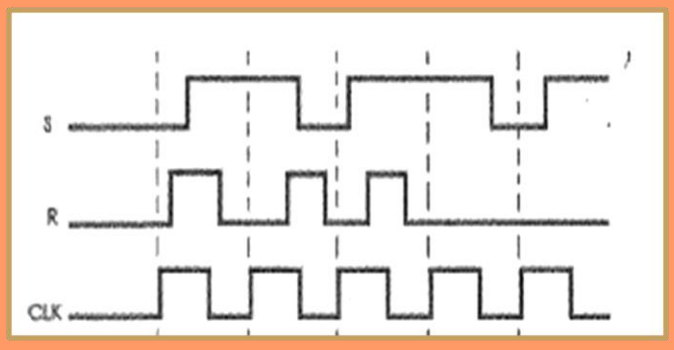
3. Determine the Q and $\underline{Q}$ for the given input of SR Active Low and Positive Edge Trigger Clock (PGT) in figure below. Assume $Q_a = 1$



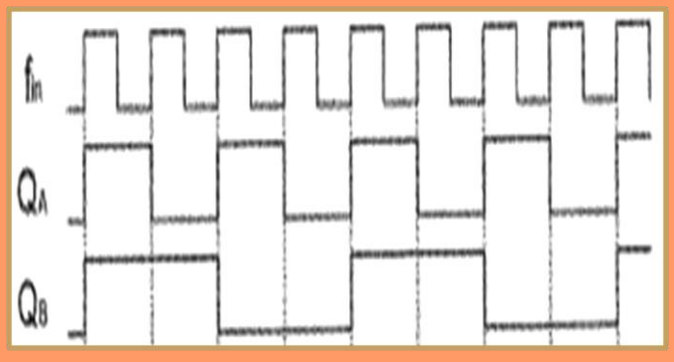
4. Based on figure below, construct the output for the flip flop. Given $Q_{initial} = 1$, Preset= Clear = 1 and clock is negative edge triggered



5. Identify the Q and Q for a given input of SR Active High with Positive Edge Trigger Clock in figure below



6. Sketch f_{out} waveform for the circuit in figure below when 8kHz square wave input is applied to the clock input of flip flop A.



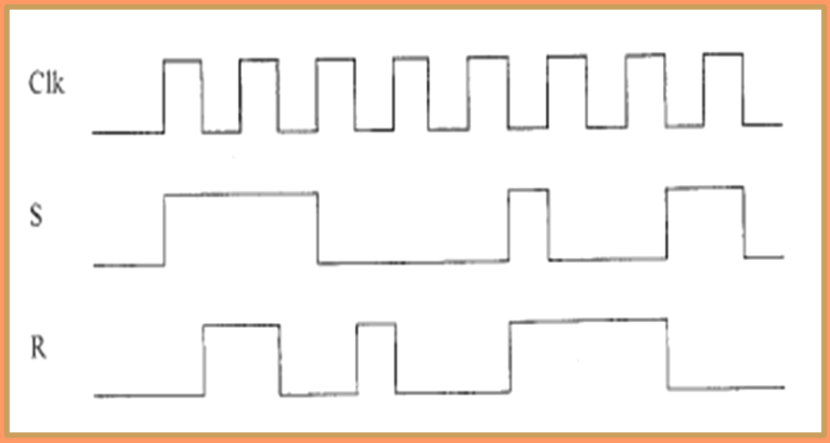
7. For active high SR flip flop, give

i. the logic symbol
ii. the truth table

8. The JK flip flop can be develop by these two flip flops. Express the logic circuit by using:

i. D flip flop
ii. T flip flop

9. Complete the following timing diagram in figure below for the negative edge triggering clocked SR flip flop by assuming the initial conditions Q is 0.



10. The JK flip flops can be develop by these two flip flops. Draw the logic circuit by using:

i. D flip flop
ii. T flip flop

Solution

1.

i. Flip flop is a circuit that has two stable states and can be used to store state information. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs. Is a memory element which change its condition based on clock signal.

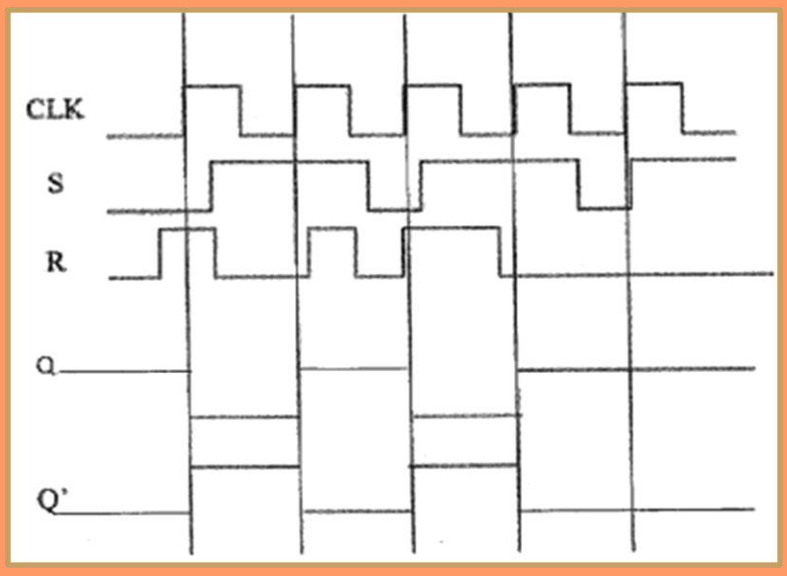
ii.

(a) JK
(b) SR
(c) D
(d) T
(e) Master JK

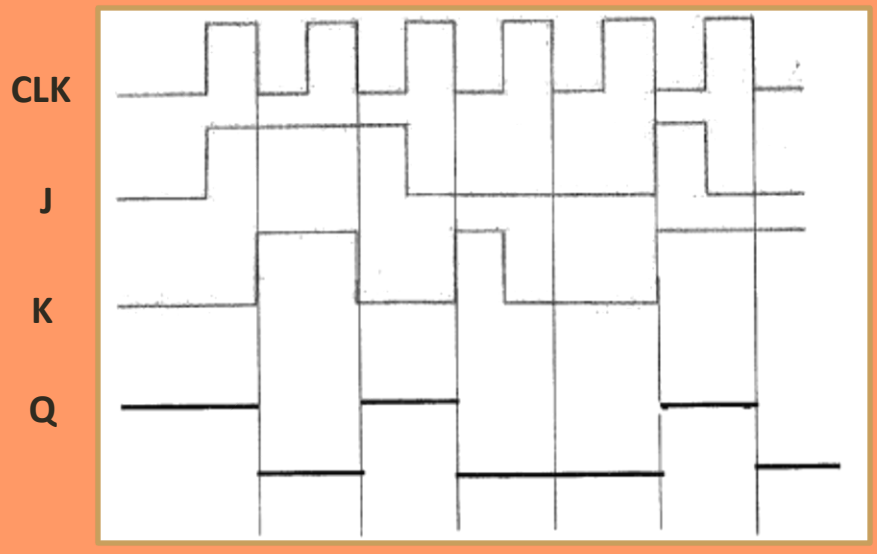
2.

SR NOR	SR NAND
Active high	Active low
Invalid when R=1, S=1	Invalid when R=0, R=0
Set R=0, S=1	Set R=1, S=0

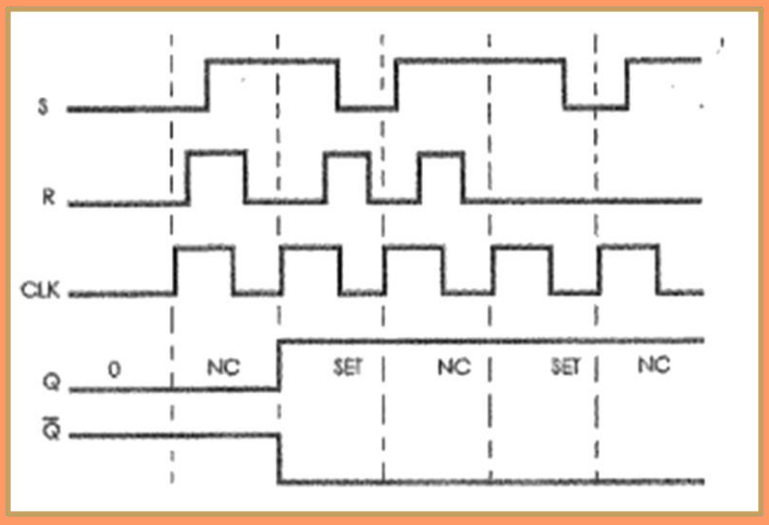
3. Determine the Q and $\overline{Q}$ for the given input of SR Active Low and Positive Edge Trigger Clock (PGT) in figure below. Assume $Q_a = 1$



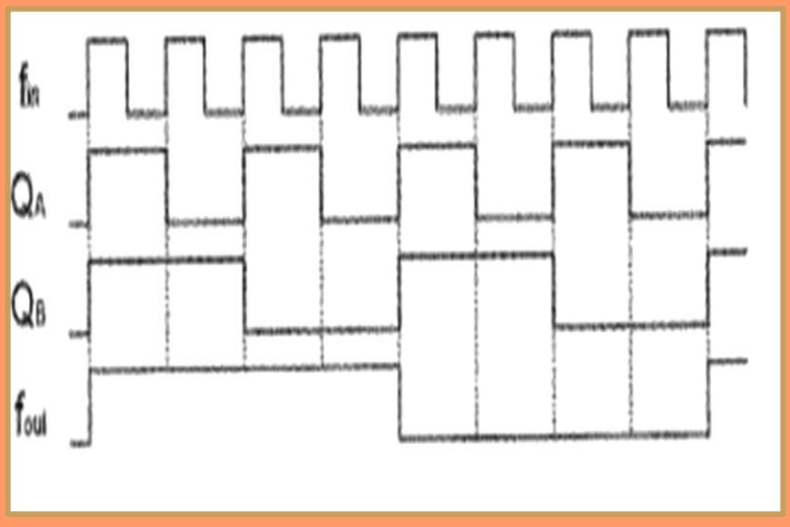
4. Based on figure below, construct the output for the flip flop. Given $Q_{initial} = 1$, Preset= Clear = 1 and clock is negative edge triggered



5.

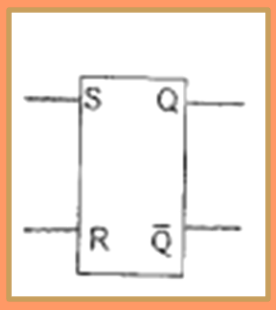


6.



7.

(i) the logic symbol

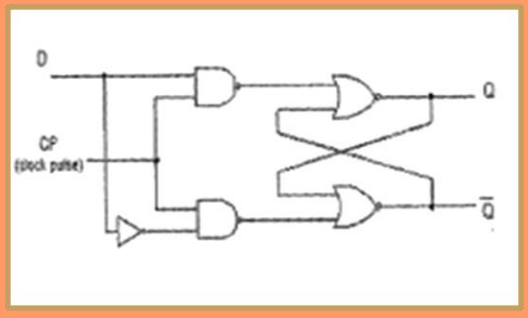


(ii) the truth table

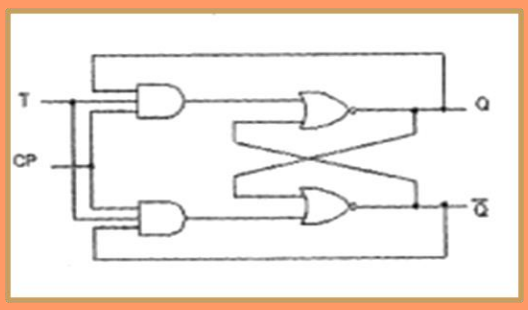
S	R	Q	$\overline{Q}$	Operation
0	0	Q	$\overline{Q}$	Hold
0	1	0	1	Reset
1	0	1	0	Set
1	0	0	0	Invalid

8.

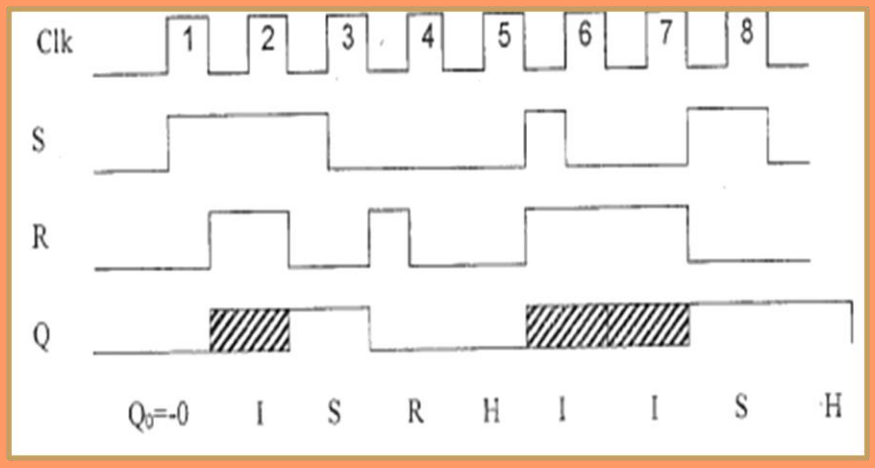
i. D flip flop



ii. T flip flop

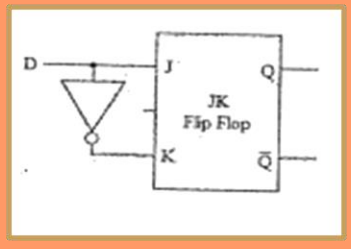


9.

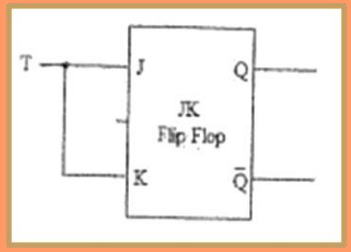


10.

i. D flip flop



ii. T flip flop



References

1. Singhal, D. B. (2018). Digital Electronics. India: S.K Kataria & Sons.
2. Sharma, D. S. (2017). Digital Electronics And Logic Design. India: S.K. Kataria & Sons.
3. Subir Kumar Sarkar, A. K. (2014). Foundation of Digital Electronics and Logic Design. U.S: PAN STANFORD.
4. Ronald J Tocci, N. S. (2011). Digital systems:Principles and Applications. Malaysia: Prentice Hall.
5. https://www.ascii-code.com/ASCII
6. https://www.electronics-tutorials.ws/sequential/seq_1.html
7. https://www.electrical4u.com/jk-flip-flop/
8. https://ecstudiosystems.com/discover/textbooks/basic-electronics/flip-flops/t-flip-flop/
9. https://www.elprocus.com/master-slave-flip-flop-circuit-and-its-working/
10. https://www.electronicsforu.com/technology-trends/learn-electronics/jk-flip-flop-circuit-truth-table-limitations-applications

POLITEKNIK
MALAYSIA
PORT DICKSON

DIGITAL SYSTEM- PRINCIPLES AND APPLICATIONS VOLUME 1

e ISBN 978-629-7643-09-0

9 786297 643090

POLITEKNIK PORT DICKSON

(online)