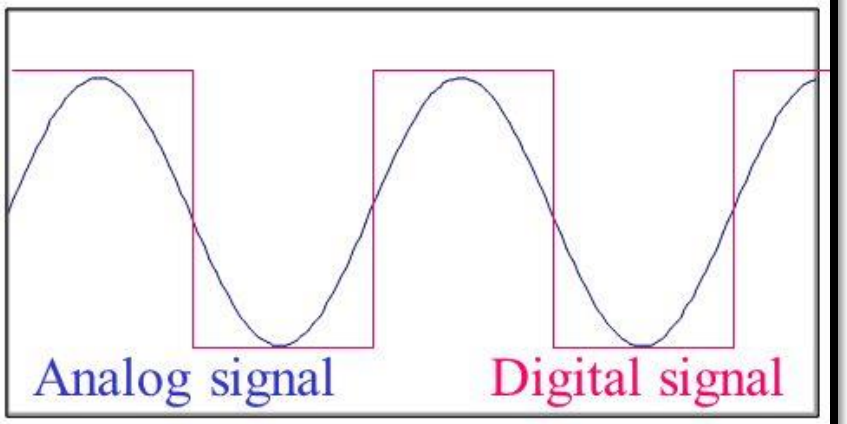
ECE DEPARTMENT
DIGITAL ELECTRONICS
3RD SEM

UNIT 1

Digital Signals versus Analog Signals



- Digital signals have two settings: ON or OFF.
- Analog signals have ranges of settings: dimmer switches, human voices, ocean waves
- Sound: Digital versus analog.
  - Analog is a wave: continuous, gradual
  - Digital is a step: non-continuous, ON/OFF

ANALOG SIGNAL VERSUS DIGITAL SIGNAL

2 KEY DIFFERENCES

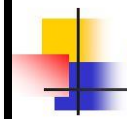
ANALOG SIGNAL	DIGITAL SIGNAL
An analog signal is a continuous signal that changes over a time period.	A digital signal is a discrete signal that carries information in binary form.
A sine wave represents an analog signal.	A square wave represents a digital signal.

Difference between analog and digital signals

S. No.	Analog signal	Digital Signal
1	Analog signals are continuous signals	Digital signals are discrete signals.
2	Analog signal uses continuous values for representing the information.	A digital signal uses discrete values for representing the information.
3	Analog signals can be affected by the noise during the transmission.	Digital signals cannot be affected by the noise during transmission.
4	Accuracy of Analog signal is affected by the noise.	Digital signals are noise-immune hence there accuracy is less affected
5	Devices which are using analog signals are less flexible	Device using digital signals are very flexible
6	Analog signals consumes less bandwidth	Digital signals consume more bandwidth.
7	Analog signal are stored in the form of continuous wave form.	Digital signals are stored in the form of binary bits "0", "1".
8	Analog signals have low cost.	Digital signals have high cost.
9	Analog signals are portable.	Digital signals are not Portable.
10	Analog signals give observation error	Digital Signals doesn't give observation error.

Advantages of Digital Signals

- The main advantage of digital signals over analog signals is that the precise signal level of the digital signal is not vital.
- This means that digital signals are fairly immune to the imperfections of real electronic systems which tend to spoil analog signals.
- Reduced cost
- Flexibility in response to design changes
- Noise immunity
- Easy to control and manipulate



Advantages of Digital Signals

- Digital circuits have only two states so:
  - Changes in value have little effect on digital signals
  - Noise and other forms of interference have little effect on digital signals
  - Little chance of error because voltage in a digital circuit must be in one state or the other
  - Information storage is easy

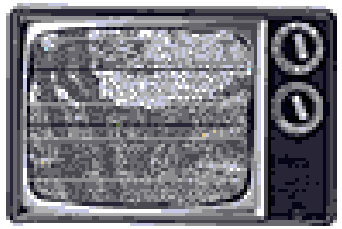
Based on SP Bali (2005), "2000 Solved Problems in Digital Electronics"

Advantages of Digital over Analog

- Digital regenerators eliminate the accumulation of noise that takes place in analog systems
  - It is thus possible to provide long-distance transmission that is nearly independent of distance
- Digital transmission systems can operate with lower signal levels or with greater distances between regenerators
  - This translates into lower overall system cost
- Digital transmission facilitates the monitoring of the quality of a transmission channel in service
  - Nonintrusive monitoring is much more difficult in analog transmission systems
- Digital transmission systems can multiplex and switch any type of information represented in a digital form
- Digital transmission also allows networks to exploit the advances in digital computer technology
  - Error correction, data encryption, various types of network protocols

	Analog	Digital
Bandwidth	Analog signal processing can be done in real time and consumes less bandwidth.	There is no guarantee that digital signal processing can be done in real time and consumes more bandwidth to carry out the same information.
Memory	Stored in the form of wave signal	Stored in the form of binary bit
Power	Analog instrument draws large power	Digital instrument drawS only negligible power
Cost	Low cost and portable	Cost is high and not easily portable
Impedance	Low	High order of 100 megaohm
Errors	Analog instruments usually have a scale which is cramped at lower end and give considerable observational errors.	Digital instruments are free from observational errors like parallax and approximation errors.

Analog Signal



if signal is weak, picture is weak, lots of static

both signals weaken over distance

Digital Signal

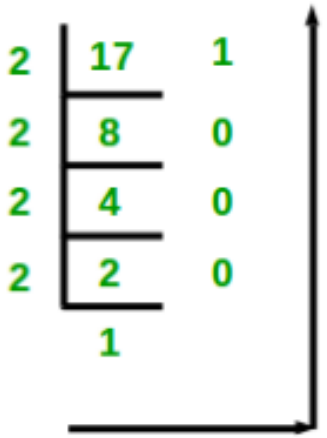
as long as tv is receiving a signal, picture is perfect

Numbering System		
System	Base	Digits
Binary	2	0, 1
Octal	8	0,1,2,3,4,5,6,7
Decimal	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

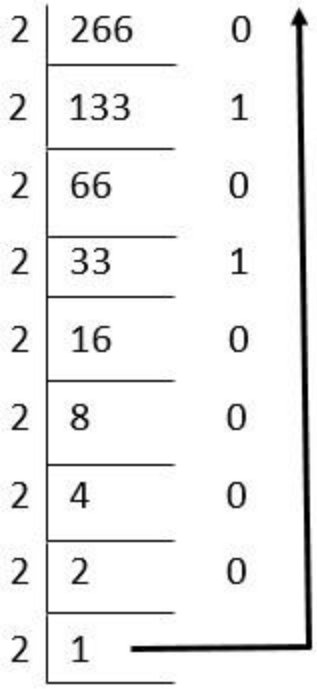
Decimal	Binary	Octal	Hex
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000		8
9	1001		9
10	1010		A
11	1011		B
12	1100		C
13	1101		D
14	1110		E
15	1111		F
MV '98			

DECIMAL TO BINARY

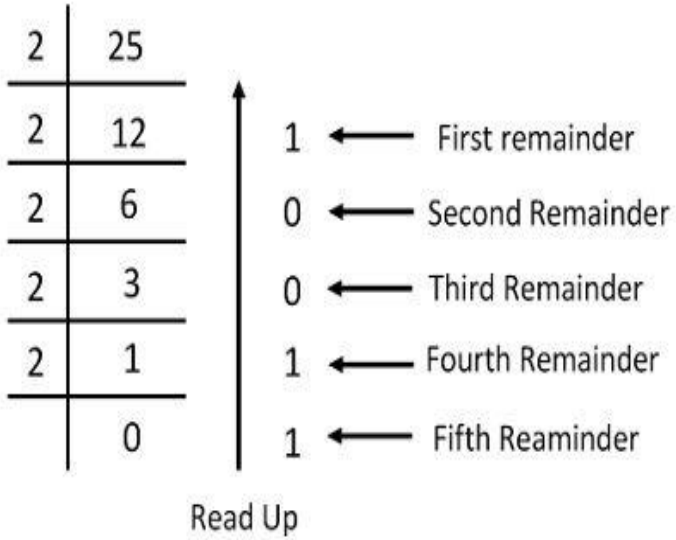
Decimal number : 17



Binary number: 10001



Binary conversion – 1 0 0 0 0 1 0 1 0

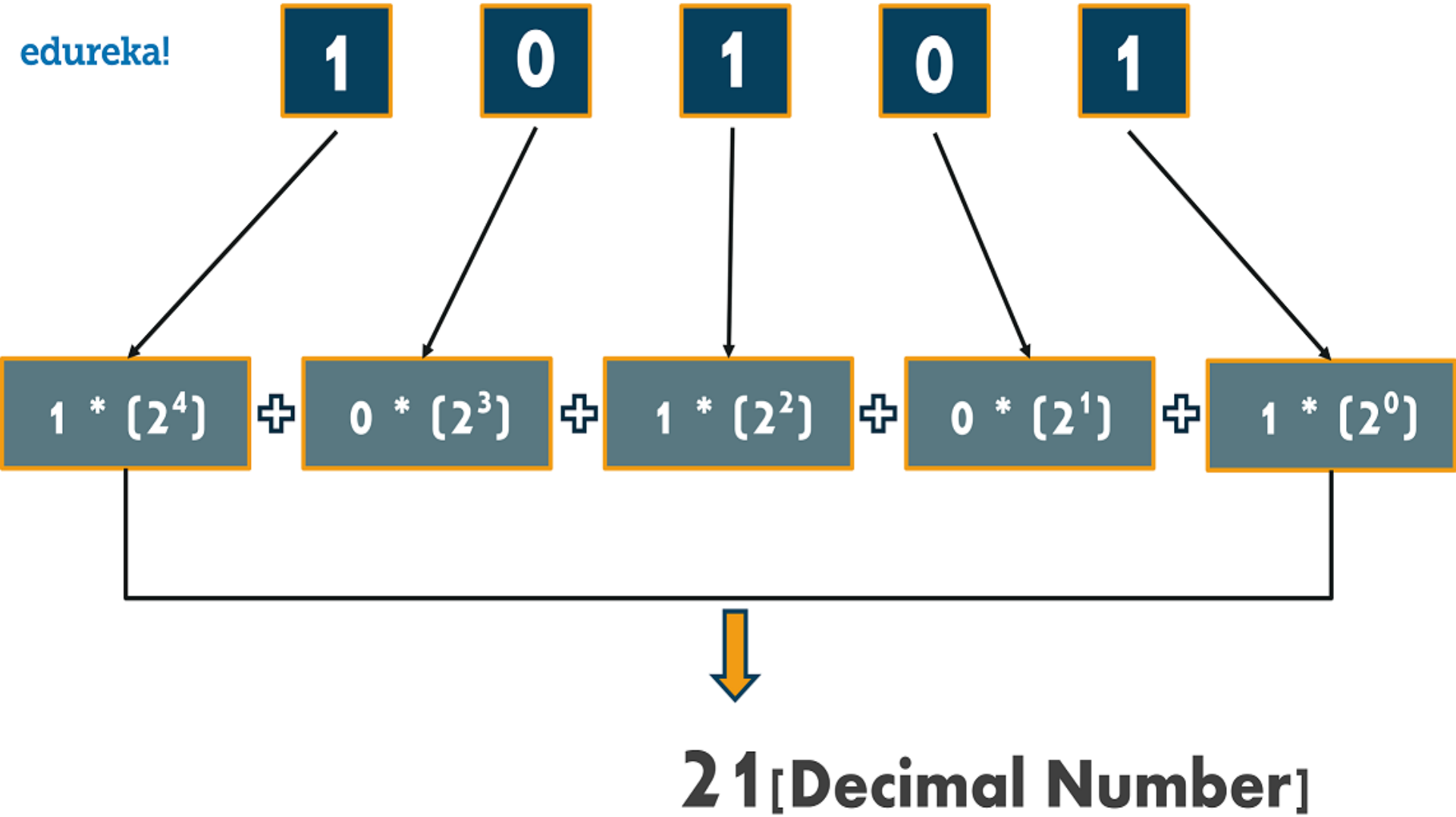


Binary Number = 11001

Circuit Globe

Binary to decimal

edureka!



Binary to decimal

Find the equivalent decimal number for binary 1010_2

Place values	2^3	2^2	2^1	2^0
Binary	1	0	1	0
Conversion	$1x2^3$	$0x2^2$	$1x2^1$	$0x2^0$
Decimal	8 +	0 +	2 +	0

10

Find the Decimal Equivalent for Binary 100101

2^5 2^4 2^3 2^2 2^1 2^0

100101

(1×2^5) (0×2^4) (0×2^3) (1×2^2) (0×2^1) (1×2^0)

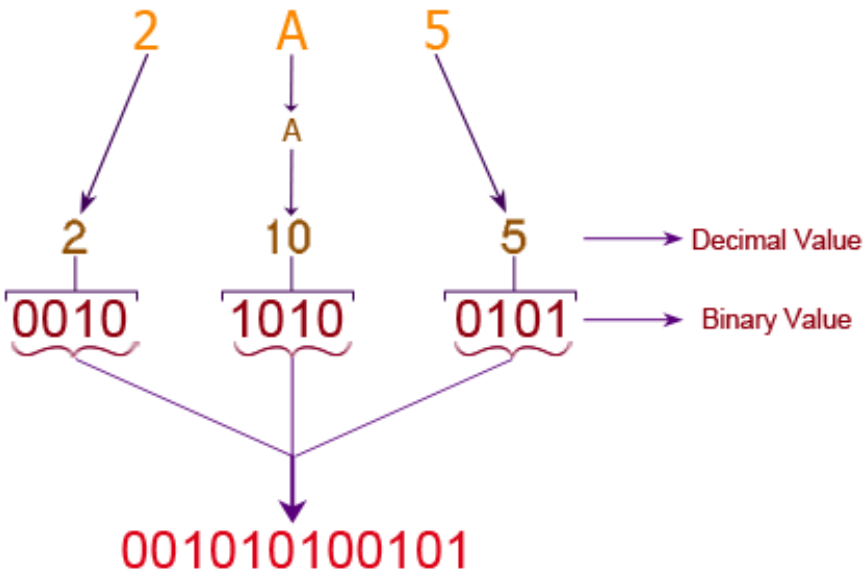
32 + 0 + 0 + 4 + 0 + 1

37

Hexadecimal to Binary

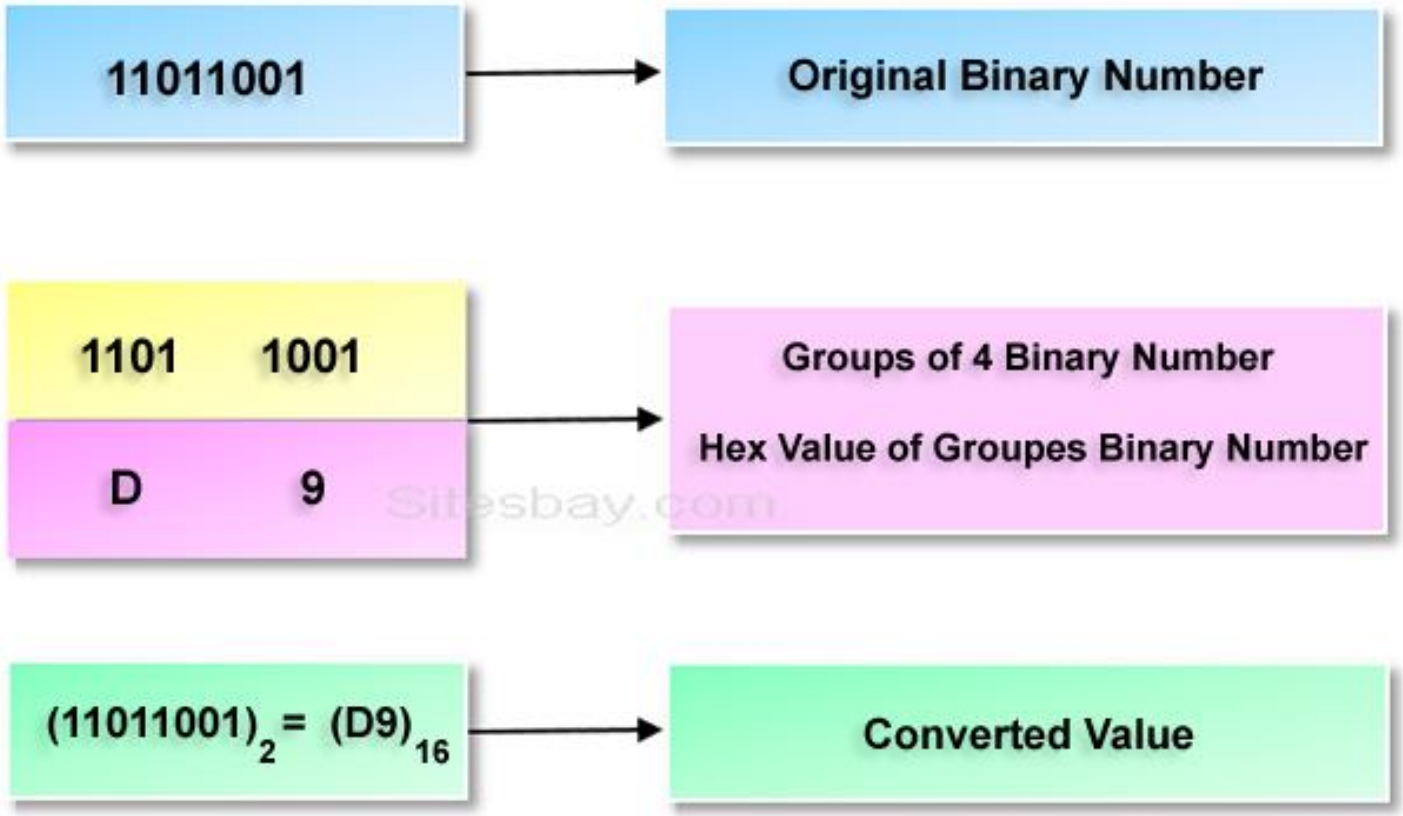
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Hexadecimal Value = 2A5



$$(2A5)_{16} = (1010100101)_2$$

Binary To HexaDecimal Conversion



Addition		Result	Carry
0 + 0	=	0	0
0 + 1	=	1	0
1 + 0	=	1	0
1 + 1	=	0	1

Adding Binary Numbers

- **Binary addition is similar to decimal addition**
  - 0+0 = 0
  - 0+1 = 1
  - 1+0 = 1
  - 1+1 = 10
- **If we have a result greater than 1, we must carry a '1' to the next digit**
  - Produce carry (to left) -> *Carry-out*
  - Get a carry (from right) --> *Carry-In*

```
  0 0 1 0
+ 1 1 0 0
= 1 1 1 0
```

```
    1
  0 0 1 0
+ 1 0 1 0
= 1 1 0 0
```

Carry-Out of bit 1, carry-In to bit 2

```
  1 1 1
  0 1 1 0
+ 1 1 1 0
= 1 0 1 0 0
```

Carries

```
    1  1  1
    1  1  0  1
+   1  1  1  1
-------------
1   1  1  0  0
```

Sol:

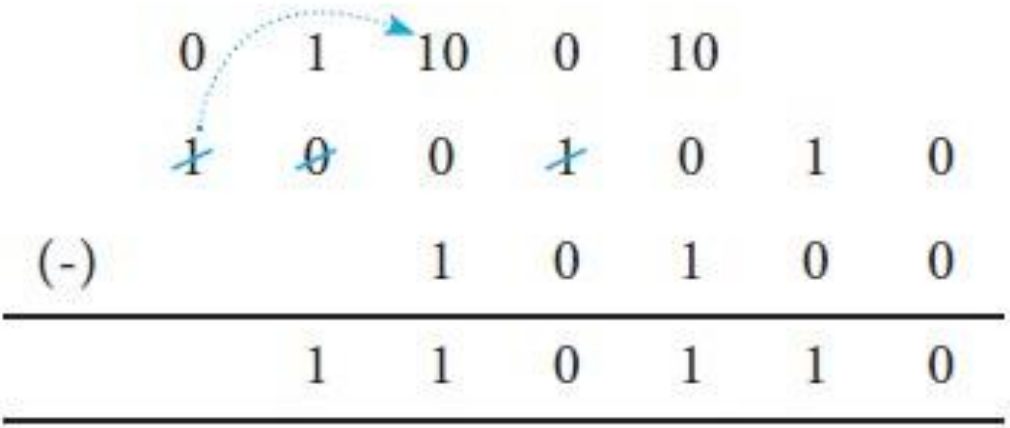
```
    1  1  1  1      <----carries
    1  1  0  1  1
+   1  1  0  1
-------------
1   0  1  0  0  0
```

BINARY SUBTRACTION

TABLE IV
RULES FOR BINARY SUBTRACTION

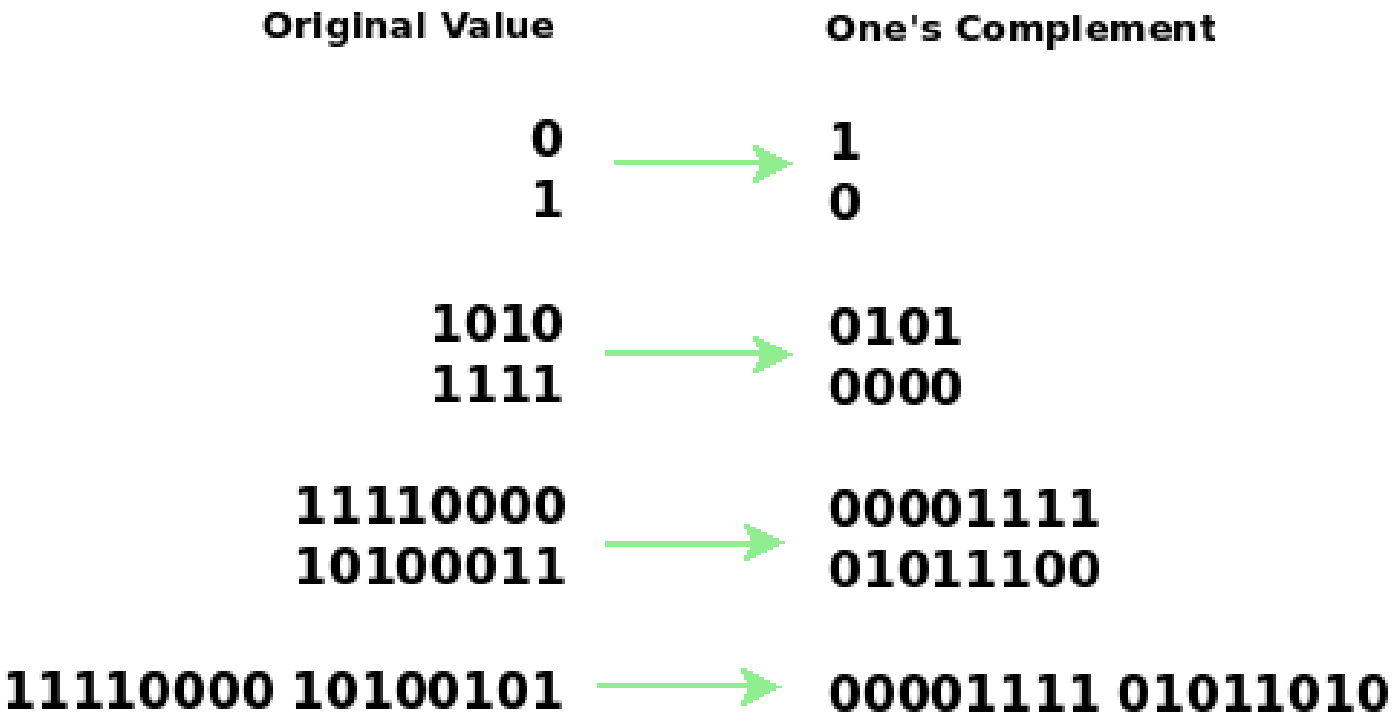
A	B	A-B	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Example Subtract $1001010_2 - 10100_2$



One's Complement

Invert all bits. Each 1 becomes a 0, and each 0 becomes a 1.



2's Complement Examples

Example #1

```
 5 = 00000101
     ↓↓↓↓↓↓↓↓   Complement Digits
     11111010
           +1   Add 1
     --------
-5 = 11111011
```

Example #2

```
-13 = 11110011
      ↓↓↓↓↓↓↓↓   Complement Digits
      00001100
            +1   Add 1
      --------
 13 = 00001101
```

1'S COMPLEMENT SUBTRACTION

- Subtraction of binary numbers using 1's complement allows subtraction only by addition.

A) Subtraction of smaller number from larger number

1) Determine the 1's complement of the smaller number.
2) Add this to the larger number.
3) There is always a carry.
4) Remove the carry and add it to the result

Example: Binary subtraction using 1's complement

Regular Approach:

M - N

M = 0 1 0 1 0 1 0 0

N = 0 1 0 0 0 1 0 0 -

---

0 0 0 1 0 0 0 0

1's complement:

M – N = M + N'

M = 0 1 0 1 0 1 0 0

N = 1 0 1 1 1 0 1 1 +

End Around Carry 1 0 0 0 0 1 1 1 1

1 +

0 0 0 1 0 0 0 0

Regular Approach:

N - M

N = 0 1 0 0 0 1 0 0

M = 0 1 0 1 0 1 0 0 -

- 0 0 0 1 0 0 0 0

1's complement:

N + M'

N = 0 1 0 0 0 1 0 0

M = 1 0 1 0 1 0 1 1 +

No End Carry

1 1 1 0 1 1 1 1

Correction Step Required:

-(1's complement of 11101111) =

-(**00010000**)

BINARY SUBTRACTION USING 1'S COMPLEMENT

Case 1: Carry is produced

Ex 1: $(1\,1\,1\,0\,1\,0)_2 - (1\,0\,1\,0\,1\,1)_2 = (?)_2$

X Y

1) 101011

↓↓↓↓↓↓

010100 ← 1's

2)
```
  1
 111010
 010100 (+)
--------
1001110
```

1 (+)

```
 001111
```

$\rightarrow 1111_2$

Case 2: No Carry

Ex 2: $(1\,0\,0\,1\,1\,0)_2 - (1\,1\,0\,0\,0\,1)_2 = (?)_2$

X Y

1) 110001

↓↓↓↓↓↓

001110 ← 1's

2)
```
   1 1 1
 1 0 0 1 1 0
 0 0 1 1 1 0 (+)
---------------
 1 1 0 1 0 0
```

↓↓↓↓↓↓

−001011

−001011

-1011_2

2's Complement Subtraction

- $16_{10} - 5_{10} \rightarrow 16_{10} + (-5_{10})$
- $\rightarrow 10000_2 + (11011_2)$

$$
\begin{array}{rr}
 & 1\ 0\ 0\ 0\ 0 \\
+ & 1\ 1\ 0\ 1\ 1 \\
\hline
 & \boxed{1}\ 0\ 1\ 0\ 1\ 1
\end{array}
\quad \rightarrow \ 11_{10}
$$

- Faster and easier than signed-magnitude and 1's complement subtraction.

Binary Codes

- Binary codes are codes which are represented in binary system with modification from the original ones.
- Weighted Binary Codes
  - Weighted binary codes are those which obey the positional weighting principles, each position of the number represents a specific weight. The binary counting sequence is an example.
- Non Weighted Codes
  - Non weighted codes are codes that are not positionally weighted. That is, each position within the binary number is not assigned a fixed value.

Weighted Code:-

In weighted code, each digit position has a weight or value. The sum of all digits multiplied by a weight gives the total amount being represented.

We can express any decimal number in tens, hundreds, thousands and so on.

Eg:- Decimal number 4327 can be written as

$4327 = 4000+300+20+7$

In the power of 10, it becomes

$4327 = 4(10^3)+3(10^2)+2(10^1)+7(10^0)$

BCD or 8421 is a type of weighted code where each digit position is being assigned a specific weight.

Non-weighted code:-

In non-weighted code, there is no positional weight i.e. each position within the binary number is not assigned a prefixed value. No specific weights are assigned to bit position in non –weighted code.

The non-weighted codes are:-

a) The Gray code

b) The Excess-3 code

BCD or 8421 code:-

It is composed of four bits representing the decimal digits 0 through 9. The 8421 indicates the binary weights of the four bits(2^3,2^2,2^1,2^0).

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

Binary Code	Decimal	BCD Code
A B C D	Number	B_5 B_4 B_3 B_2 B_1
0 0 0 0	0	0 0 0 0 0
0 0 0 1	1	0 0 0 0 1
0 0 1 0	2	0 0 0 1 0
0 0 1 1	3	0 0 0 1 1
0 1 0 0	4	0 0 1 0 0
0 1 0 1	5	0 0 1 0 1
0 1 1 0	6	0 0 1 1 0
0 1 1 1	7	0 0 1 1 1
1 0 0 0	8	0 1 0 0 0
1 0 0 1	9	0 1 0 0 1
1 0 1 0	10	1 0 0 0 0
1 0 1 1	11	1 0 0 0 1
1 1 0 0	12	1 0 0 1 0
1 1 0 1	13	1 0 0 1 1
1 1 1 0	14	1 0 1 0 0
1 1 1 1	15	1 0 1 0 1

Decimal	8 4 2 1 CODE BCD CODE	XS 3
0	0000 +0011	0011
1	0001 +0011	0100
2	0010 +0011	0101
3	0011 +0011	0110
4	0100 +0011	0111
5	0101 +0011	1000
6	0110 +0011	1001
7	0111 +0011	1010
8	1000 +0011	1011

Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

Excess-3 Code (XS-3)

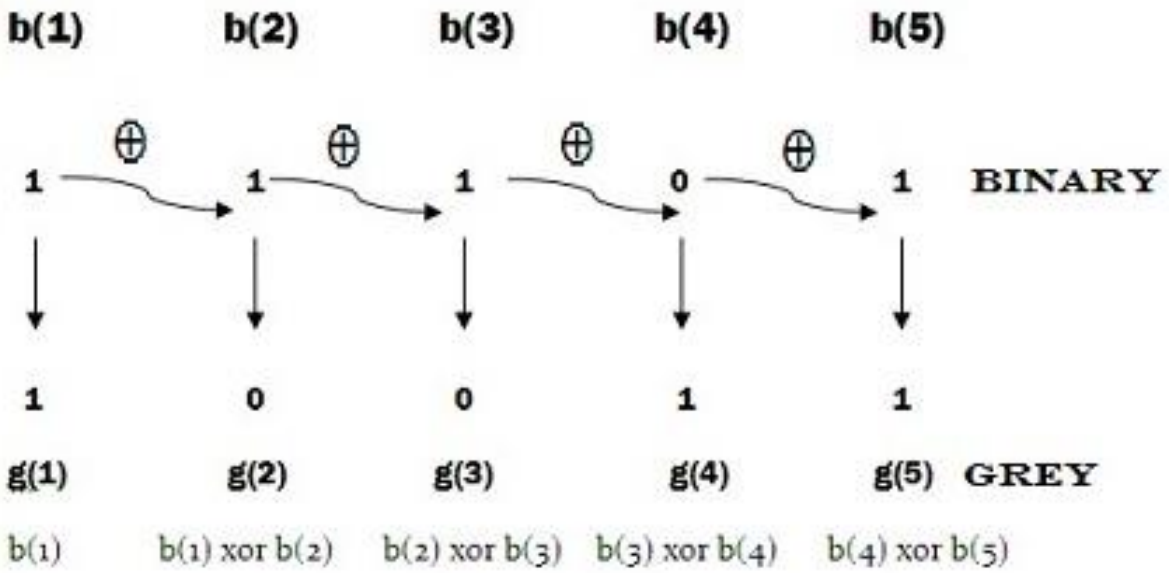
Example 1: Obtain Xs-3 Code for 428 Decimal

	4	2	8
	0100	0010	1000
+	0011	0011	0011
	0111	0101	1011

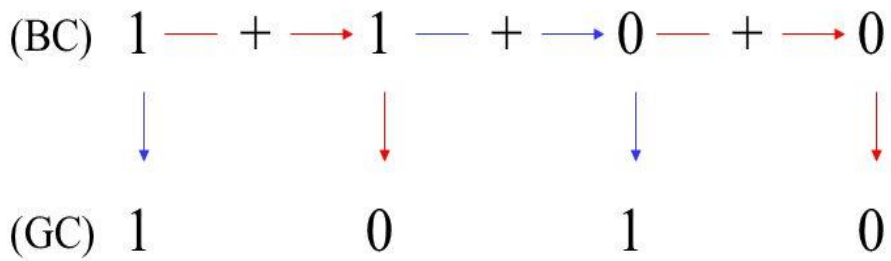
Decimal	Binary	Gray	BCD	Excess-3
0	0	0000	0000	0011
1	1	0001	0001	0100
2	10	0011	0010	0101
3	11	0010	0011	0110
4	100	0110	0100	0111
5	101	0111	0101	1000
6	110	0101	0110	1001
7	111	0100	0111	1010
8	1000	1100	1000	1011
9	1001	1101	1001	1100

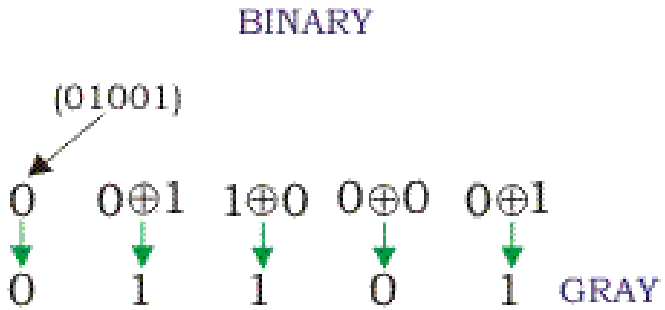
Binary to Grey Code Conversion

Convert the binary 11101_2 to its equivalent Grey code



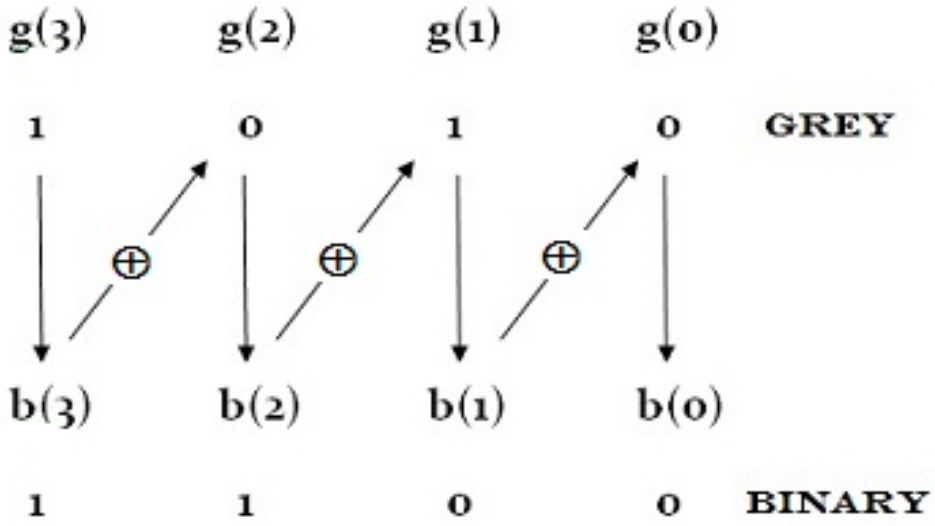
Binary to Gray Code Conversion

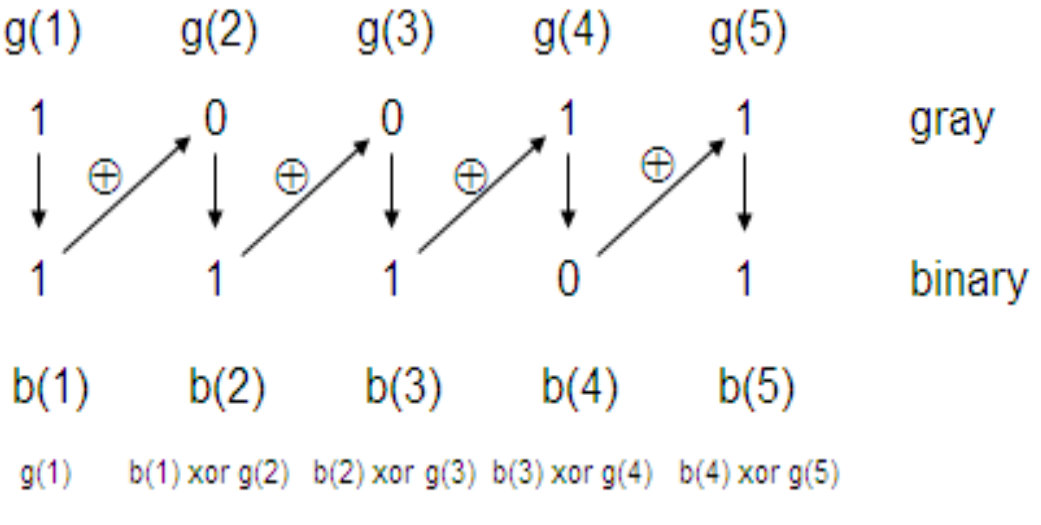




Grey Code to Binary Conversion

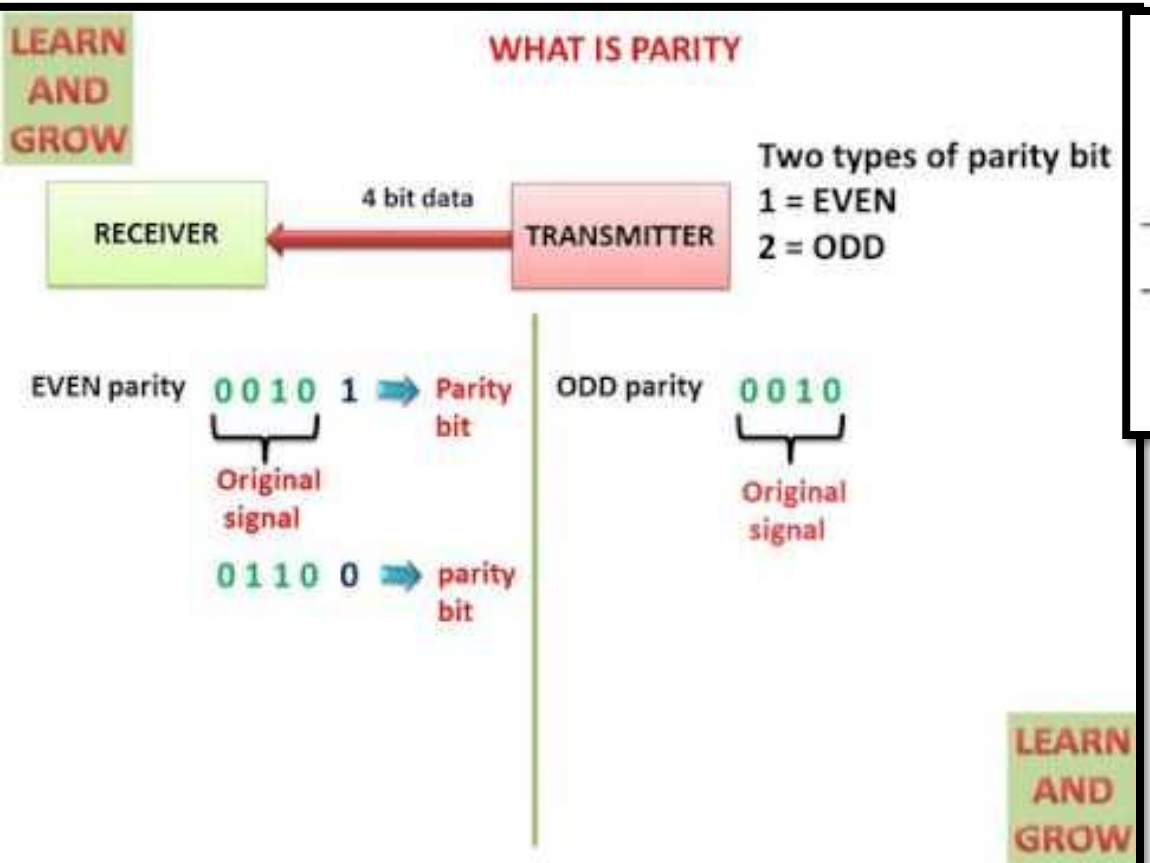
Convert the Grey code 1010 to its equivalent Binary





What is Parity Generator?

- A **Parity Generator** is a Combinational Logic Circuit that Generates the Parity bit in the Transmitter.
- A Parity bit is used for the Purpose of Detecting Errors during Transmissions of binary Information.
- It is an Extra bit Included with a binary Message to Make the Number of 1's either Odd or Even.



Parity bit examples

sequence of seven bits	with eighth even parity bit:	with eighth odd parity bit:
0100010	01000100	01000101
1000000	10000001	10000000

ComputerHope.com

How to Adjust Parity Bits for Even or Odd Parity

ORIGINAL DATA	EVEN PARITY	ODD PARITY
00000000	0	1
01011011	1	0
01010101	0	1
11111111	0	1
10000000	1	0
01001001	1	0

SOURCE: UNIVERSITY OF LONDON BIRKBECK

PARITY BITS

Type of bit parity	Successful transmission scenario
Even parity	A wants to transmit: 1001 A computes parity bit value: 1+0+0+1 = 0 A adds parity bit and sends: 10010 B receives: 10010 B computes parity: 1+0+0+1+0 = 0 (Even) B reports correct transmission after observing expected even result.
Odd parity	A wants to transmit: 1001 A computes parity bit value: 1+0+0+1 = 1 A adds parity bit and sends: 10011 B receives: 10011 B computes overall parity: 1+0+0+1+1 = 1 (Odd) B reports correct transmission after observing expected odd result.

Even Parity

Successful transmission

Transmitter (T)

. T wants to transmit 1011
. T Counts number of 1's (3)
. Parity generator adds 1
. Data is 10111

Receiver (R)

. R receives 10111
. Parity checker checks no of 1's
. Count is even (4)
. No error

Unsuccessful transmission

Transmitter (T)

. T wants to transmit 1011
. T counts number of 1s (3)
. Parity generator adds 1
. Data is 10111

Receiver (R)

. R receives 10011
. Parity checker checks no of 1's
. Count is odd (3)
. Not matched, error message is sent
. Retransmission of data

UNIT 2

Positive and Negative Logic (p.77)

- Positive and Negative Logic
  - Two signal values <=> two logic values
  - Positive logic: H=1; L=0
  - Negative logic: H=0; L=1
- Consider a TTL gate
  - A positive logic AND gate
  - A negative logic OR gate
  - The positive logic is used in this book

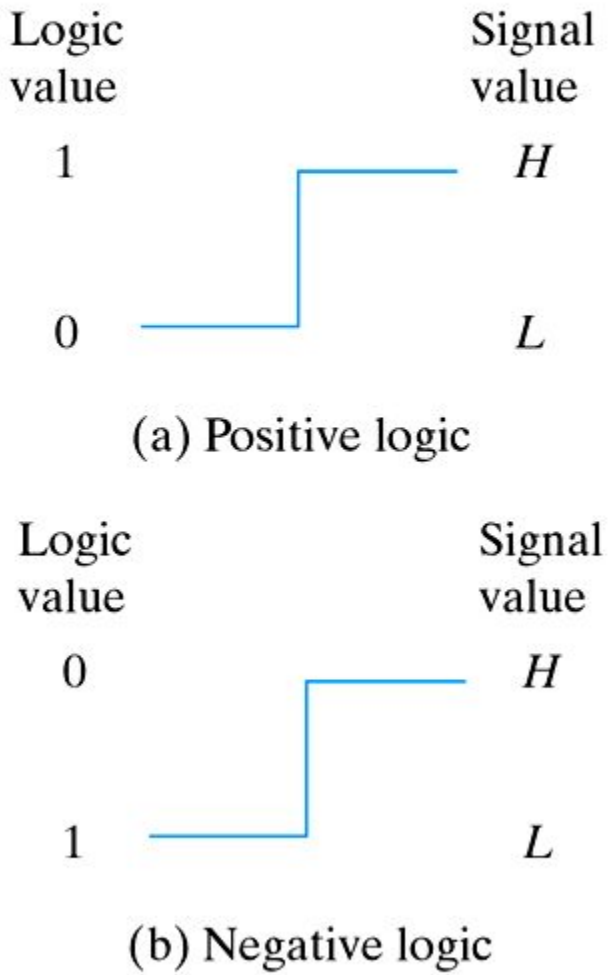


Figure 2.9 Signal assignment and logic polarity

GROW

+ve logic

high voltage correspond to logic '1'

'+5V' = LOGIC "1"

'0V' = LOGIC "0"

-ve logic

high voltage correspond to logic '0'

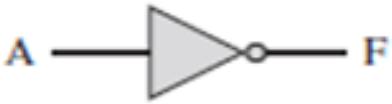
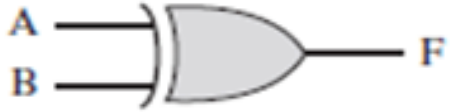
'+5V' = LOGIC "0"

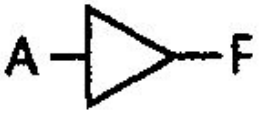
'0V' = LOGIC "1"

EXAMPLE =

Logic 0 = -5V

LEARN

Name	Graphical Symbol	Algebraic Function	Truth Table
AND		$F = A \cdot B$ or $F = AB$	A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = A + B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$F = \overline{A}$ or $F = A'$	A | F 0 | 1 1 | 0
NAND		$F = \overline{AB}$	A B | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{A + B}$	A B | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR		$F = A \oplus B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0



Buffer

$F = A$

A	F
0	0
1	1

AND

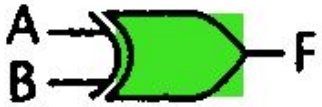
$F = AB$

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

OR

$F = A+B$

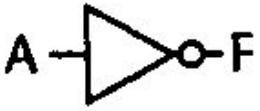
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1



XOR

$F = A \oplus B$

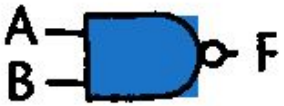
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



Inverter

$F = \overline{A}$

A	F
0	1
1	0



NAND

$F = \overline{AB}$

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

NOR

$F = \overline{A+B}$

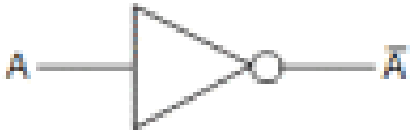
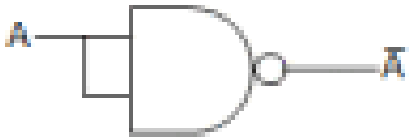
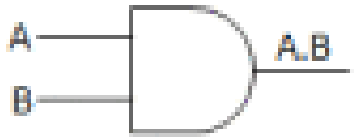
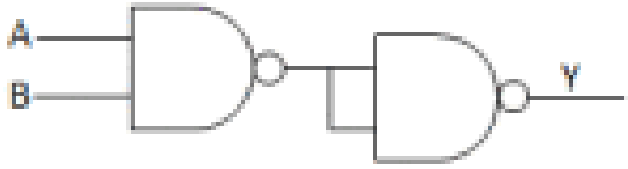
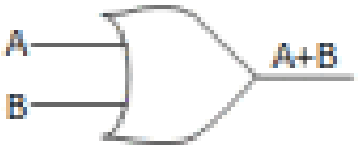
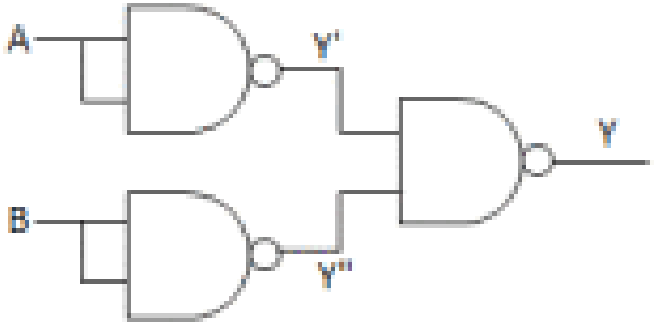
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

XNOR

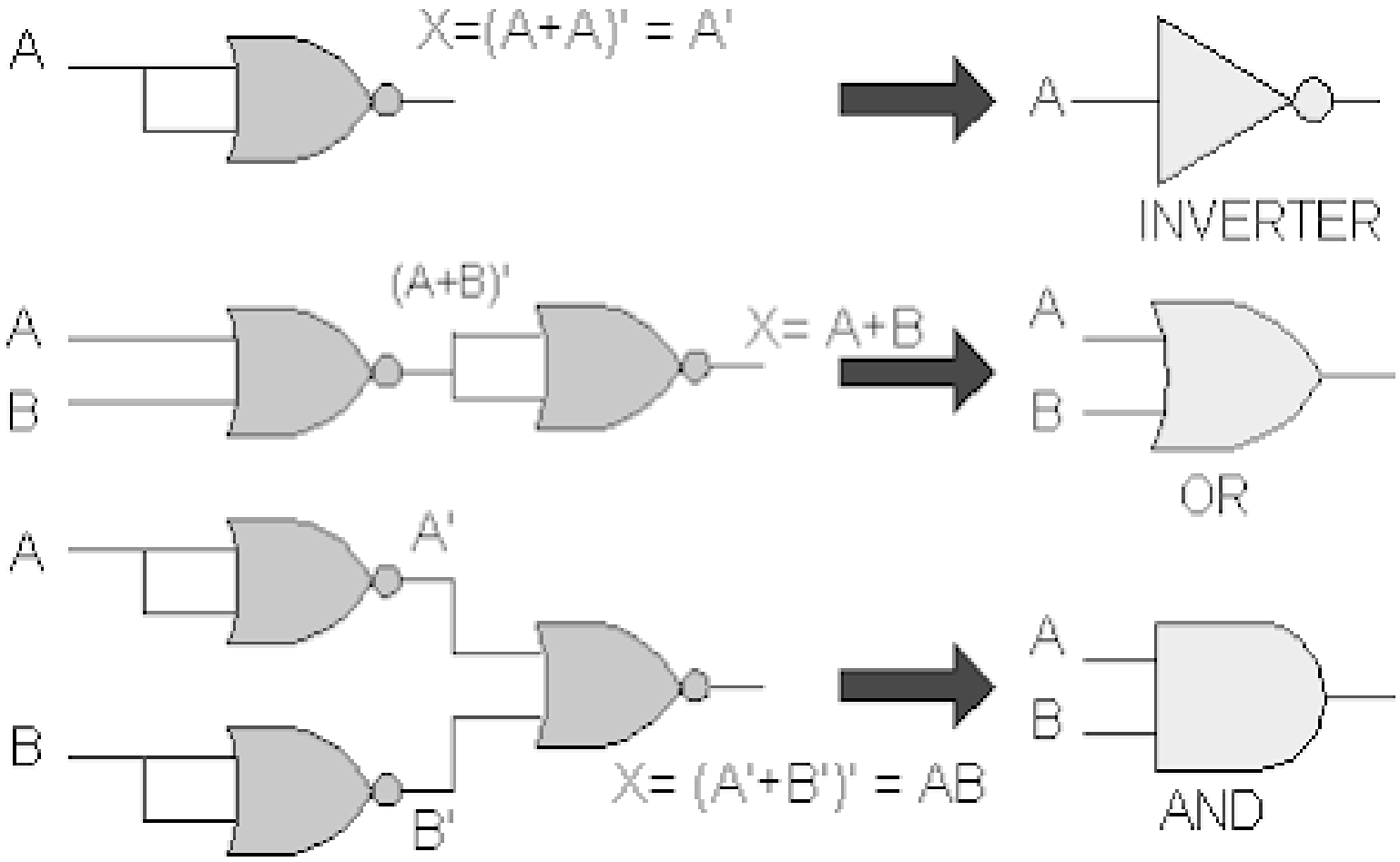
$F = \overline{A \oplus B}$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

NAND AS UNIVERSAL GATE

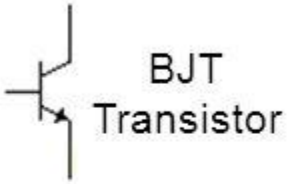
Logic Function Only	Symbol	Circuit using NAND gate
Inverter		
AND		
OR		

NOR AS UNIVERSAL GATE



TTL Vs. CMOS Logic

TTL: Transistor-Transistor Logic



- Constructed from Bipolar Junction Transistors (BJT)
- Advantages:
  - *Faster than CMOS*
  - *Not sensitive to damage from electrostatic-discharge*
- Disadvantages:
  - *Uses more power than CMOS*

CMOS: Complementary Metal Oxide Semiconductor

- Constructed from Metal Oxide Semiconductor Field-Effect Transistors (MOSFET)
- Advantages:
  - *Uses less power than TTL*
- Disadvantages:
  - *Slower than TTL*
  - *Very sensitive to damage from electrostatic-discharge*

Fundamental Theorems & Postulates of Boolean Algebra

Identities:	(1)	$X + 0 = X$	(1D) $X \cdot 1 = X$
Null Elements:	(2)	$X + 1 = 1$	(2D) $X \cdot 0 = 0$
Indempotency:	(3)	$X + X = X$	(3D) $X \cdot X = X$
Involution (Double Negation):	(4)	$(X')' = X$	
Complements:	(5)	$X + X' = 1$	(5D) $X \cdot X' = 0$
Commutativity:	(6)	$X + Y = Y + X$	(6D) $X \cdot Y = Y \cdot X$
Associativity:	(7)	$(X+Y)+Z = X+(Y+Z)$	(7D) $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$
Distributivity:	(8)	$X \cdot Y + X \cdot Z = X \cdot (Y+Z)$	(8D) $(X+Y) \cdot (X+Z) = X+Y \cdot Z$
Combining:	(9)	$X \cdot Y + X \cdot Y' = X$	(9D) $(X+Y) \cdot (X+Y') = X$
Covering:	(10)	$X + X \cdot Y = X$	(10D) $X \cdot (X+Y) = X$
DeMorgan's Laws:	(12)	$(X \cdot Y \cdot Z)' = X' + Y' + Z'$	(12D) $(X+Y+Z)' = X' \cdot Y' \cdot Z'$
Consensus:	(17)	$X \cdot Y + X' \cdot Z + Y \cdot Z = X \cdot Y + X' \cdot Z$	(17D) $(X+Y) \cdot (X'+Z) \cdot (Y+Z) = (X+Y) \cdot (X'+Z)$

Shannon Expansion: (18) $F(X,Y,Z) = X \cdot F(1,Y,Z) + X' \cdot F(0,Y,Z)$

(18D) $F(X,Y,Z) = (X+F(0,Y,Z)) \cdot (X'+F(1,Y,Z))$

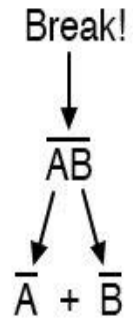
Basic Rules of Boolean Algebra

1. $A + 0 = A$	7. $A \cdot A = A$
2. $A + 1 = 1$	8. $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	9. $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	10. $A + AB = A$
5. $A + A = A$	11. $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	12. $(A + B)(A + C) = A + BC$

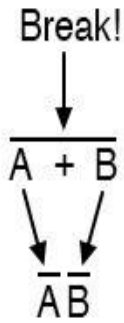
DeMorgan's Theorem

$$\overline{(AB)} = (\overline{A} + \overline{B}) \qquad \overline{(A + B)} = (\overline{A}\ \overline{B})$$

DeMorgan's Theorems

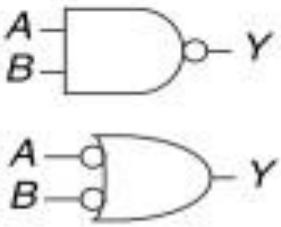


NAND to Negative-OR



NOR to Negative-AND

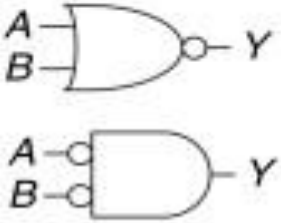
NAND



$Y = \overline{AB} = \overline{A} + \overline{B}$

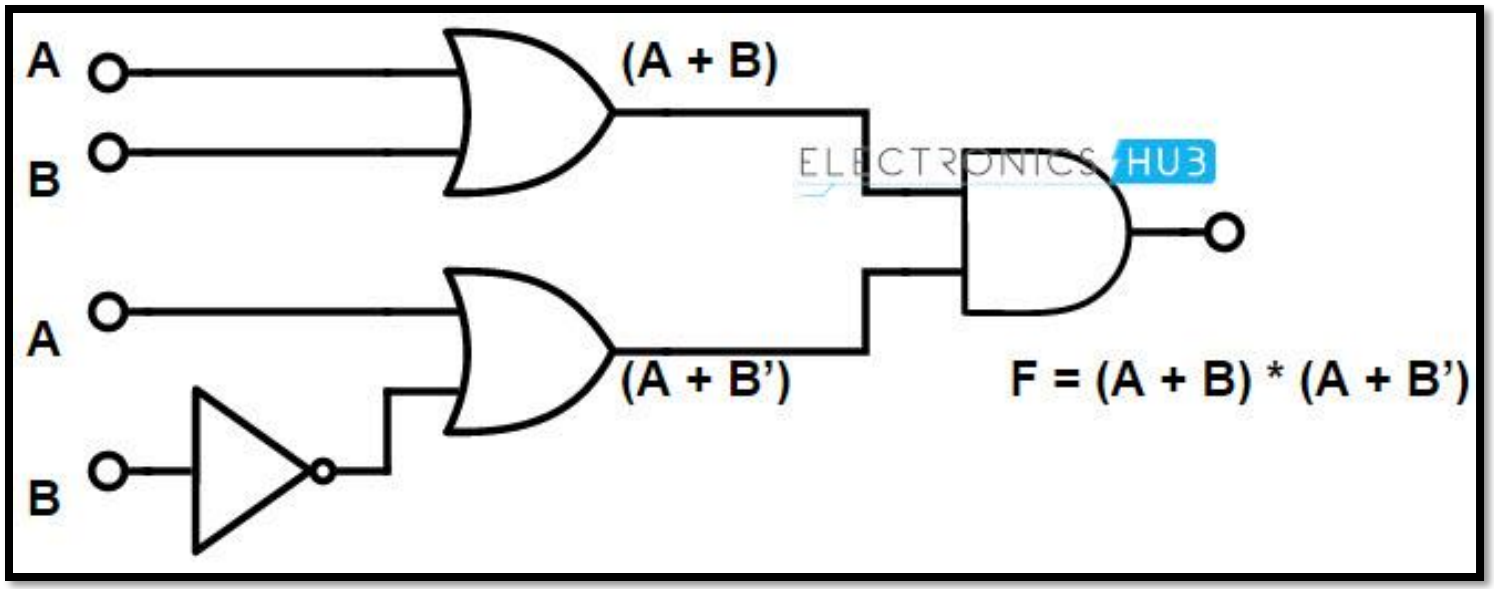
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

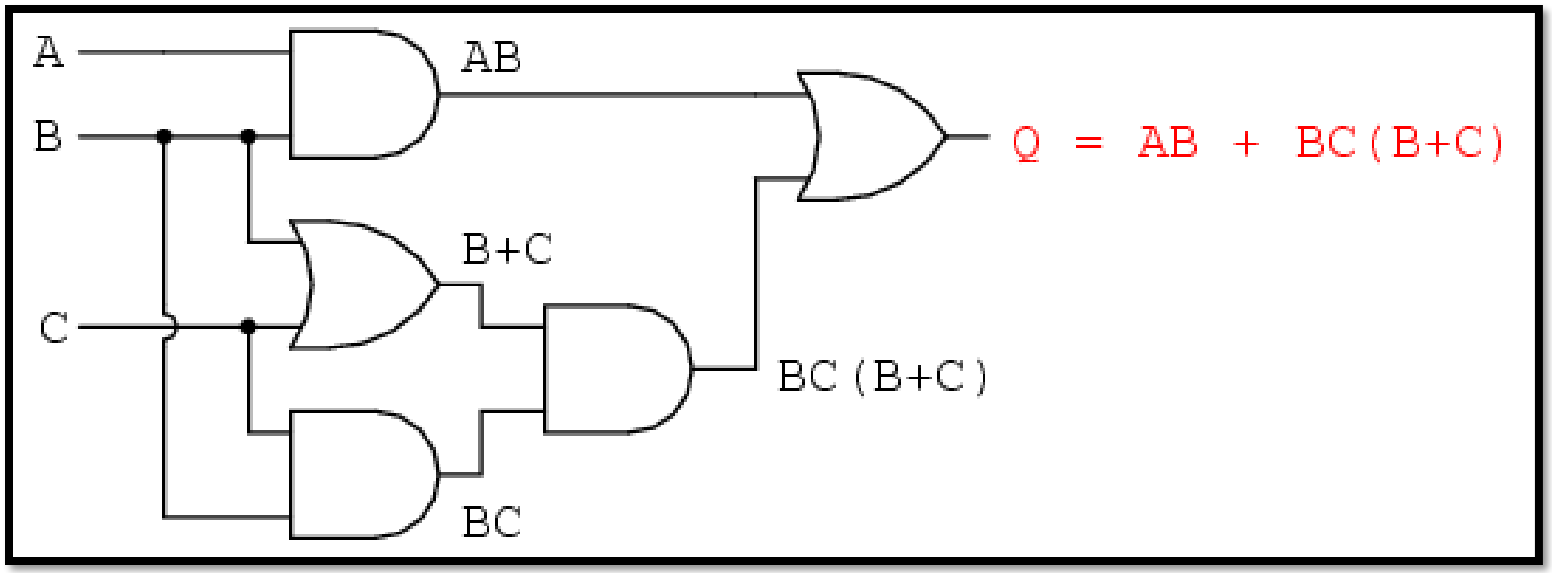
NOR

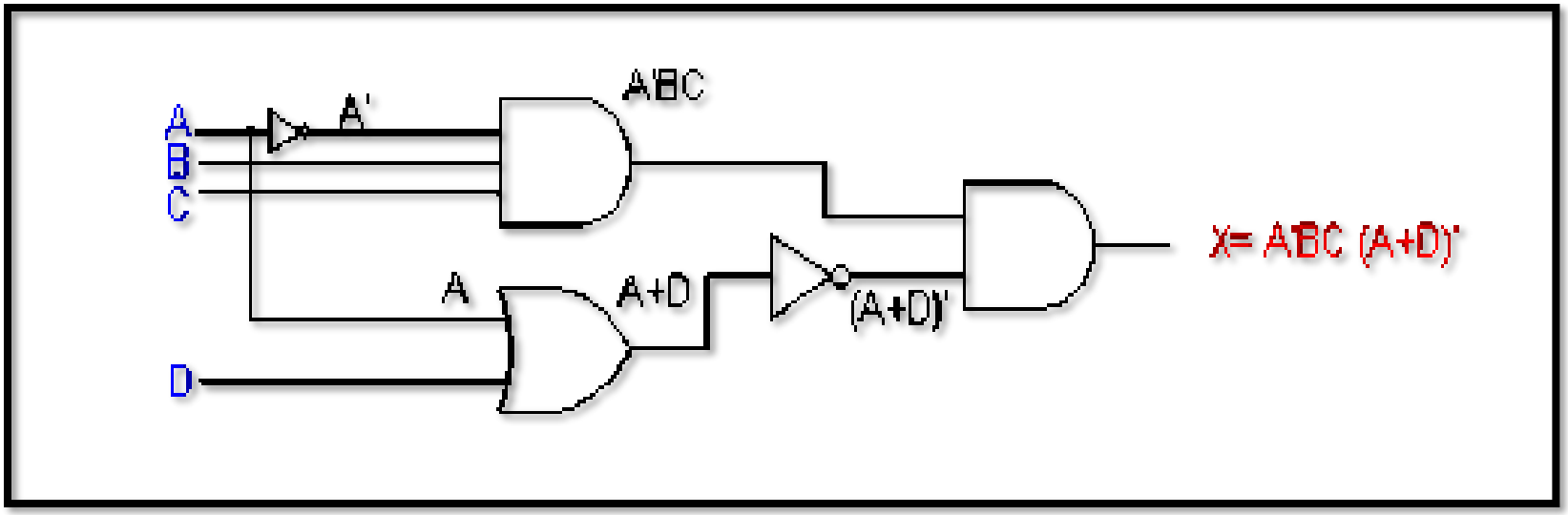


$Y = \overline{A + B} = \overline{A}\,\overline{B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0


A
B
A
B
(A + B)
(A + B')
ELECTRONICS HUB
F = (A + B) * (A + B')

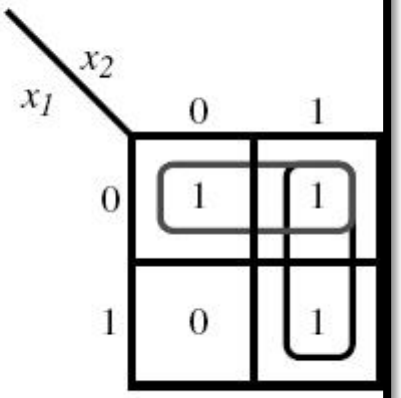

A
B
C
AB
B+C
BC (B+C)
BC
Q = AB + BC(B+C)


A
B
C
A'
A'BC
A
A+D
(A+D)'
D
X= A'BC (A+D)'

K-MAP two variable

A \ B	0	1
0	0 A'B'	1 A'B
1	2 AB'	3 AB

x_1	x_2	f
0	0	1
0	1	1
1	0	0
1	1	1



$f = \bar{x}_1 + x_2$

3-input K-map

Y \ AB / C	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{A}B\bar{C}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{A}BC$	ABC	$A\bar{B}C$

Truth Table

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

K-Map

Y \ AB / C	00	01	11	10
0	0	1	0	0
1	0	1	0	1

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

=

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

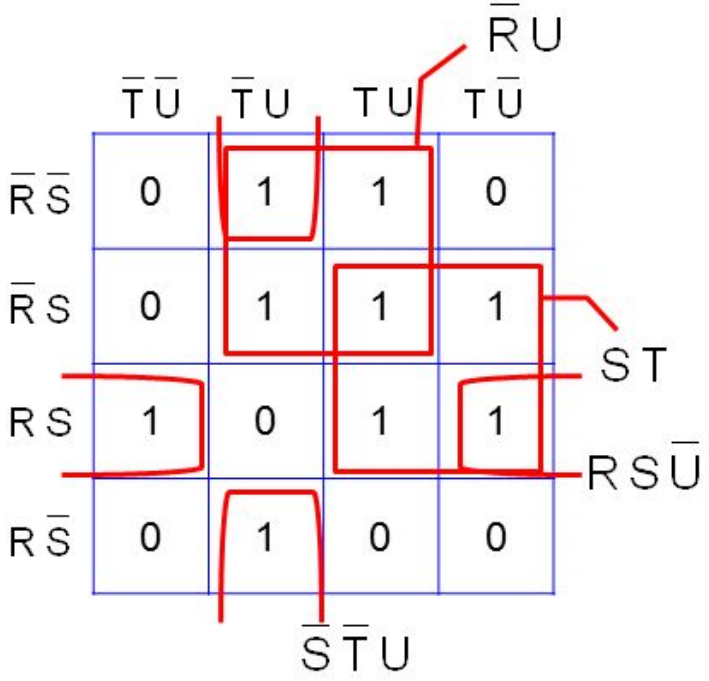
Example #3 : 4 Variable K-Map

Example:

After labeling and transferring the truth-table data into the K-Map, write the simplified sum-of-products (SOP) logic expression for the logic function F_3.

Solution:

R	S	T	U	F_3
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



$$F_3 = RS\overline{U} + \overline{S}\,\overline{T}U + \overline{R}U + ST$$

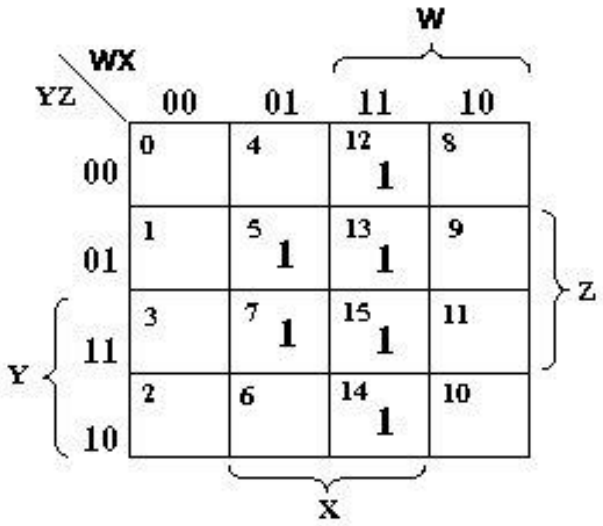
4-Variable K-map Example

For the function $F(W,X,Y,Z) = \Sigma_{W,X,Y,Z}(5,7,12,13,14,15)$

Truth Table:

Row	W	X	Y	Z	F
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

K-map



$$f(a,b,c,d) = \sum(0,1,2,3,4,5,6,7,8,10,13)$$

$$f = \bar{a}\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}\bar{c}d + \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}cd + \bar{a}b\bar{c}\bar{d} + \bar{a}b\bar{c}d + \bar{a}bc\bar{d} + \bar{a}bcd + a\bar{b}\bar{c}\bar{d} + a\bar{b}\bar{c}d + a\bar{b}cd + ab\bar{c}\bar{d}$$

	abcd	f
0	0000	1
1	0001	1
2	0010	1
3	0011	1
4	0100	1
5	0101	1
6	0110	1
7	0111	1
8	1000	1
9	1001	0
10	1010	1
11	1011	0
12	1100	0
13	1101	1
14	1110	0
15	1111	0

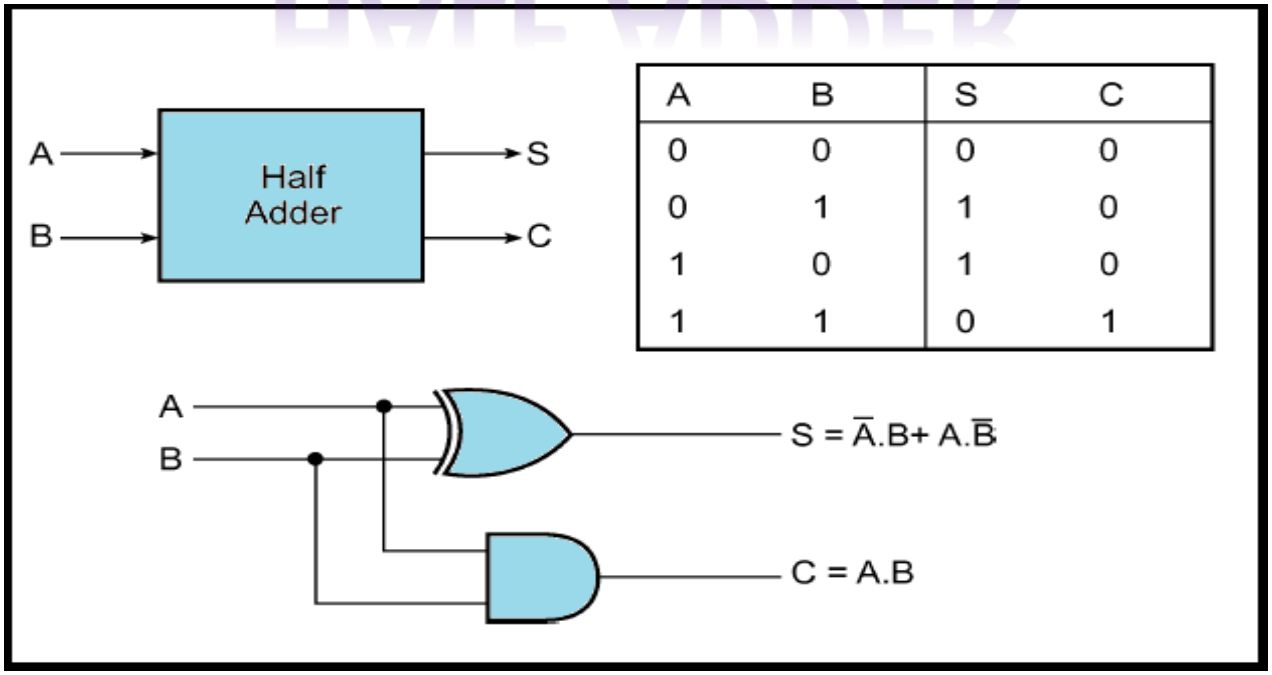
$$f(a,b,c,d) = \prod(9,11,12,14,15)$$

$$f = (\bar{a}+b+c+\bar{d})\cdot(\bar{a}+b+\bar{c}+\bar{d})\cdot(\bar{a}+\bar{b}+c+d)\cdot(\bar{a}+\bar{b}+\bar{c}+d)\cdot(\bar{a}+\bar{b}+\bar{c}+\bar{d})$$

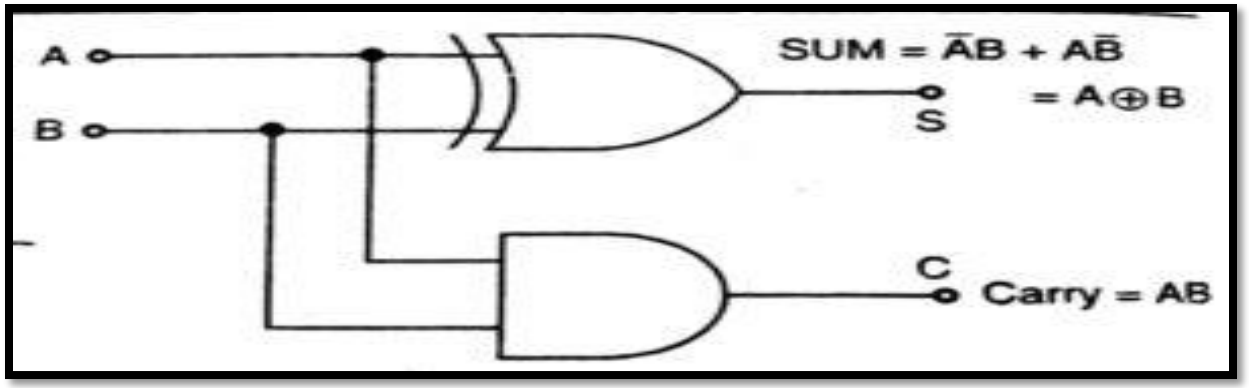
ab \ cd	00	01	11	10
00	0 1	1 1	3 1	2 1
01	4 1	5 1	7 1	6 1
11	12 0	13 1	15 0	14 0
10	8 1	9 0	11 0	10 1

UNIT 3

HALF ADDER

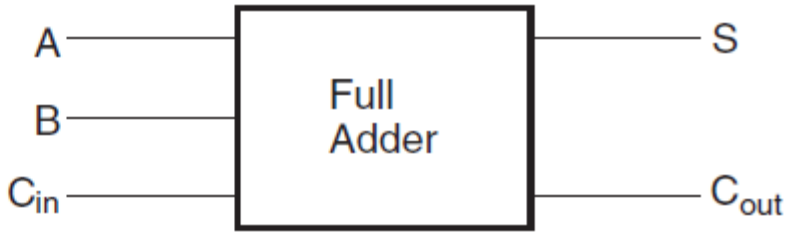


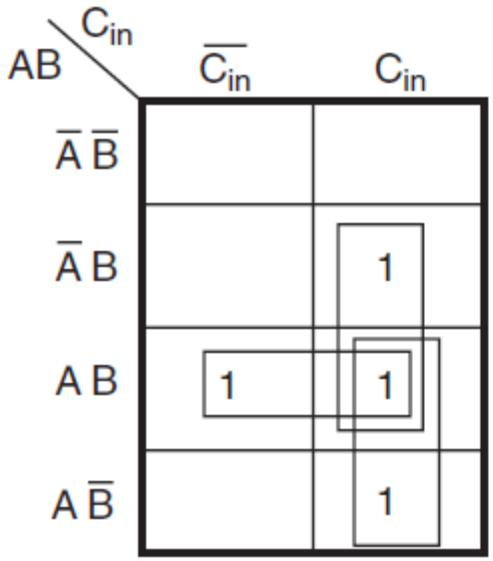
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

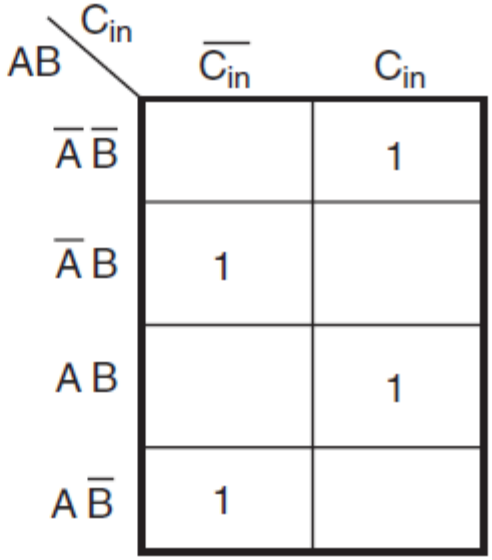


Full Adder

A	B	C_{in}	SUM (S)	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

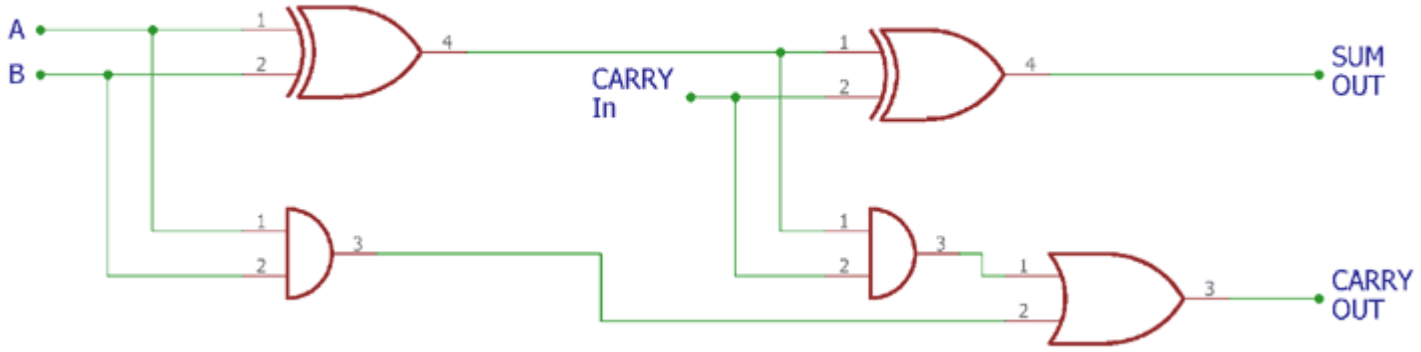






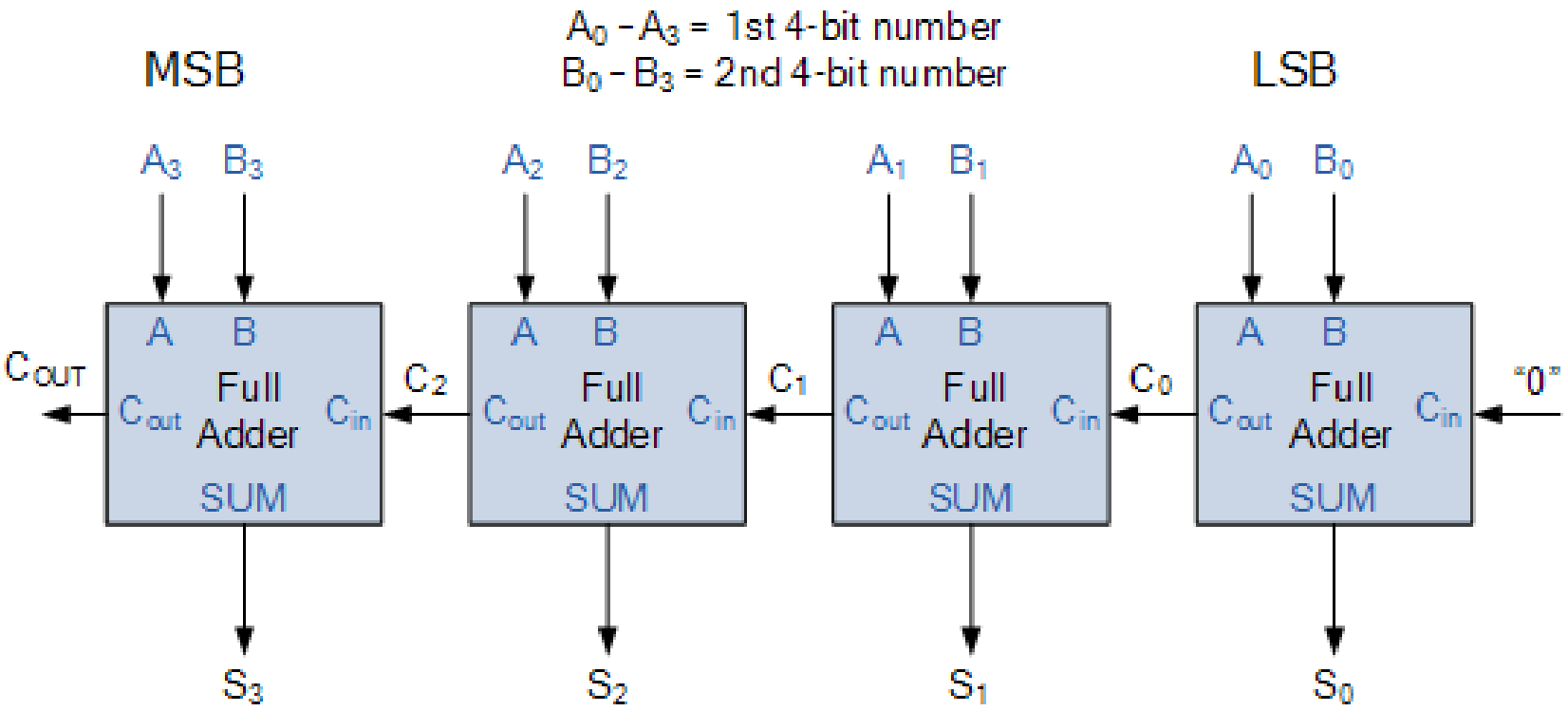
$$C_{out} = \overline{A}.B.C_{in} + A.\overline{B}.C_{in} + A.B.\overline{C}_{in} + A.B.C_{in}$$

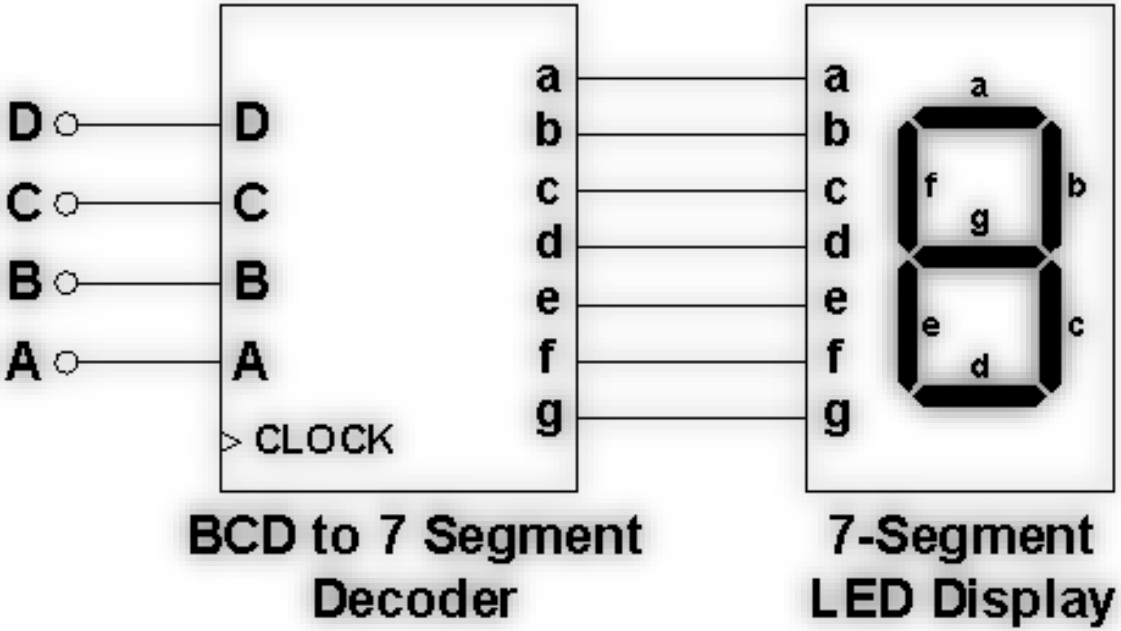
$$S = \overline{A}.\overline{B}.C_{in} + \overline{A}.B.\overline{C}_{in} + A.\overline{B}.\overline{C}_{in} + A.B.C_{in}$$



Full-Adder Circuit

4 bit adder

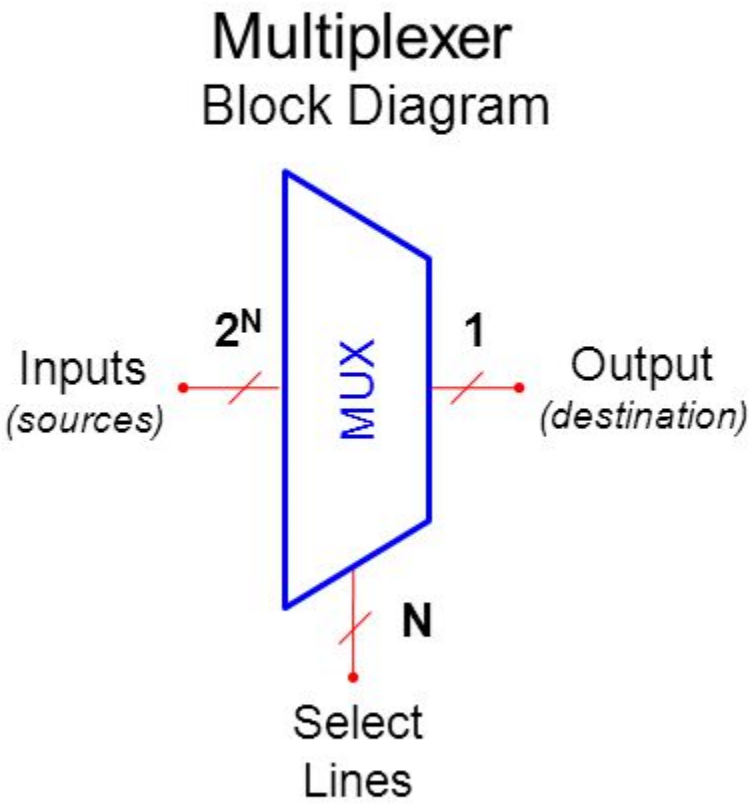

A0 – A3 = 1st 4-bit number
B0 – B3 = 2nd 4-bit number
MSB
LSB
A3
B3
A2
B2
A1
B1
A0
B0
A
B
Cout
Full Adder
Cin
SUM
COUT
C2
C1
C0
"0"
S3
S2
S1
S0

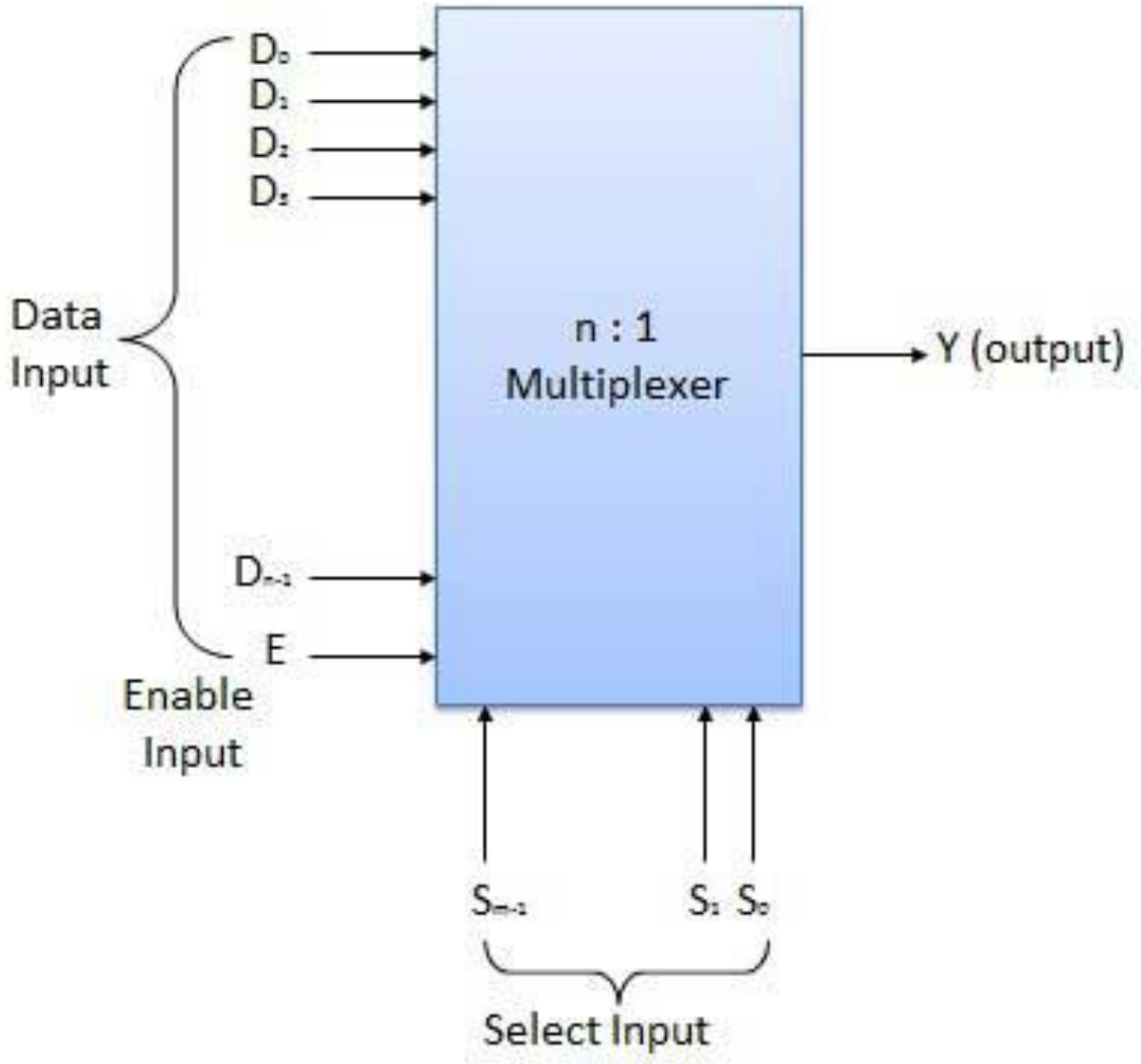


A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

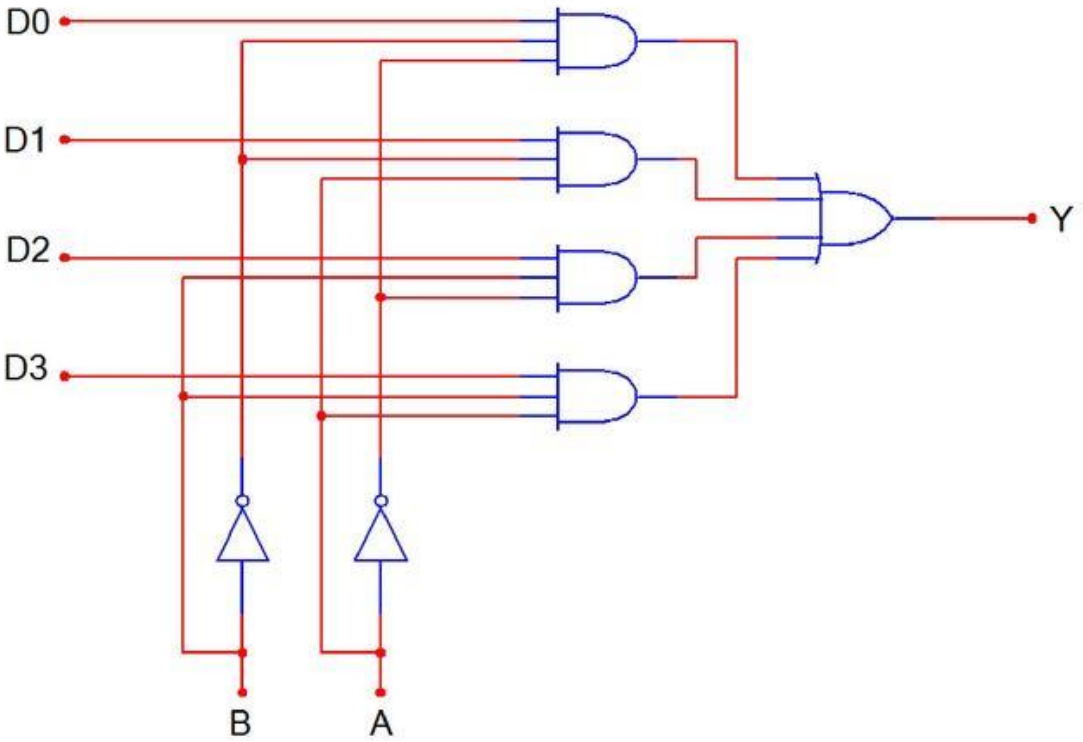
What is a Multiplexer (MUX)?

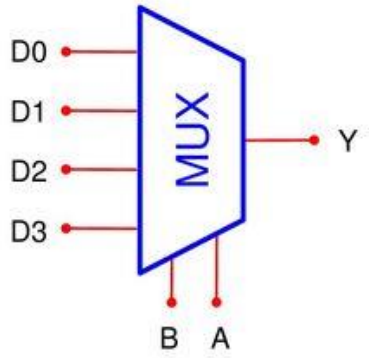
- A MUX is a digital switch that has multiple inputs (sources) and a single output (destination).
- The select lines determine which input is connected to the output.
- MUX Types
  - → 2-to-1 (1 select line)
  - → 4-to-1 (2 select lines)
  - → 8-to-1 (3 select lines)
  - → 16-to-1 (4 select lines)




D0
D1
D2
D3
Data Input
n : 1 Multiplexer
Y (output)
Dn-1
E
Enable Input
Sm-1
S1
S0
Select Input

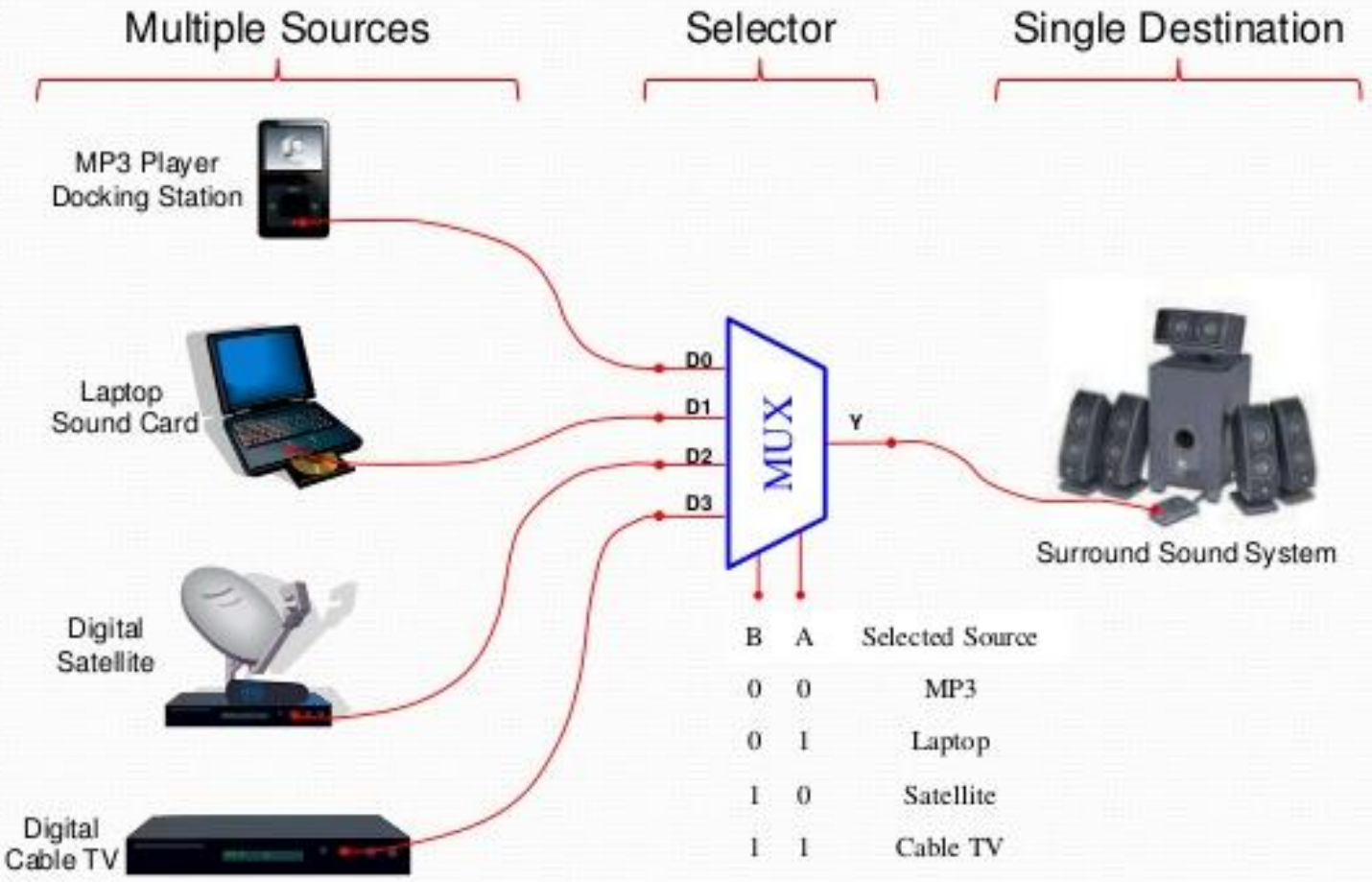
4-to-1 Multiplexer (MUX)





B	A	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Typical Application of a MUX



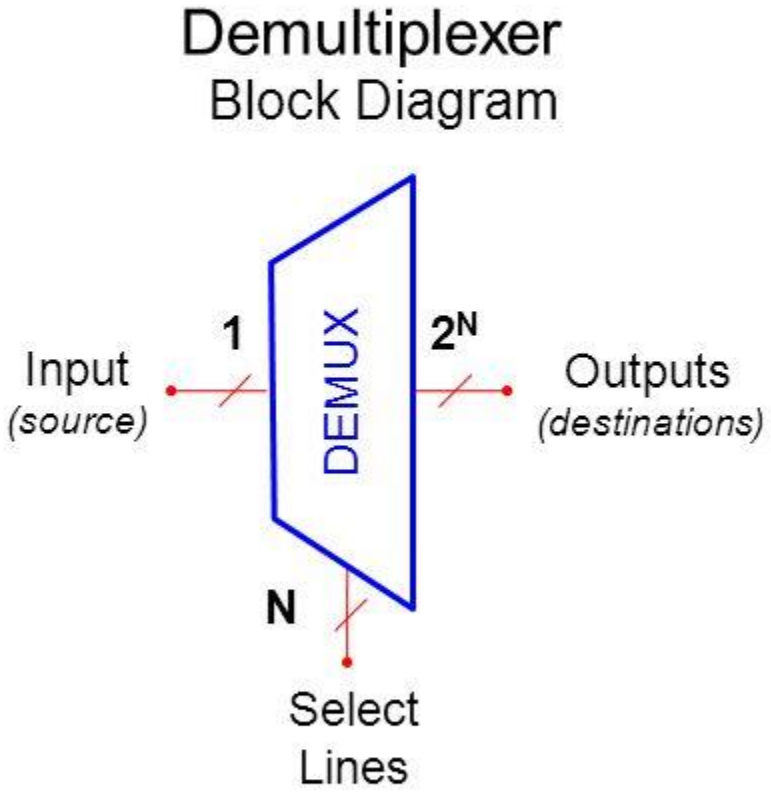
B	A	Selected Source
0	0	MP3
0	1	Laptop
1	0	Satellite
1	1	Cable TV

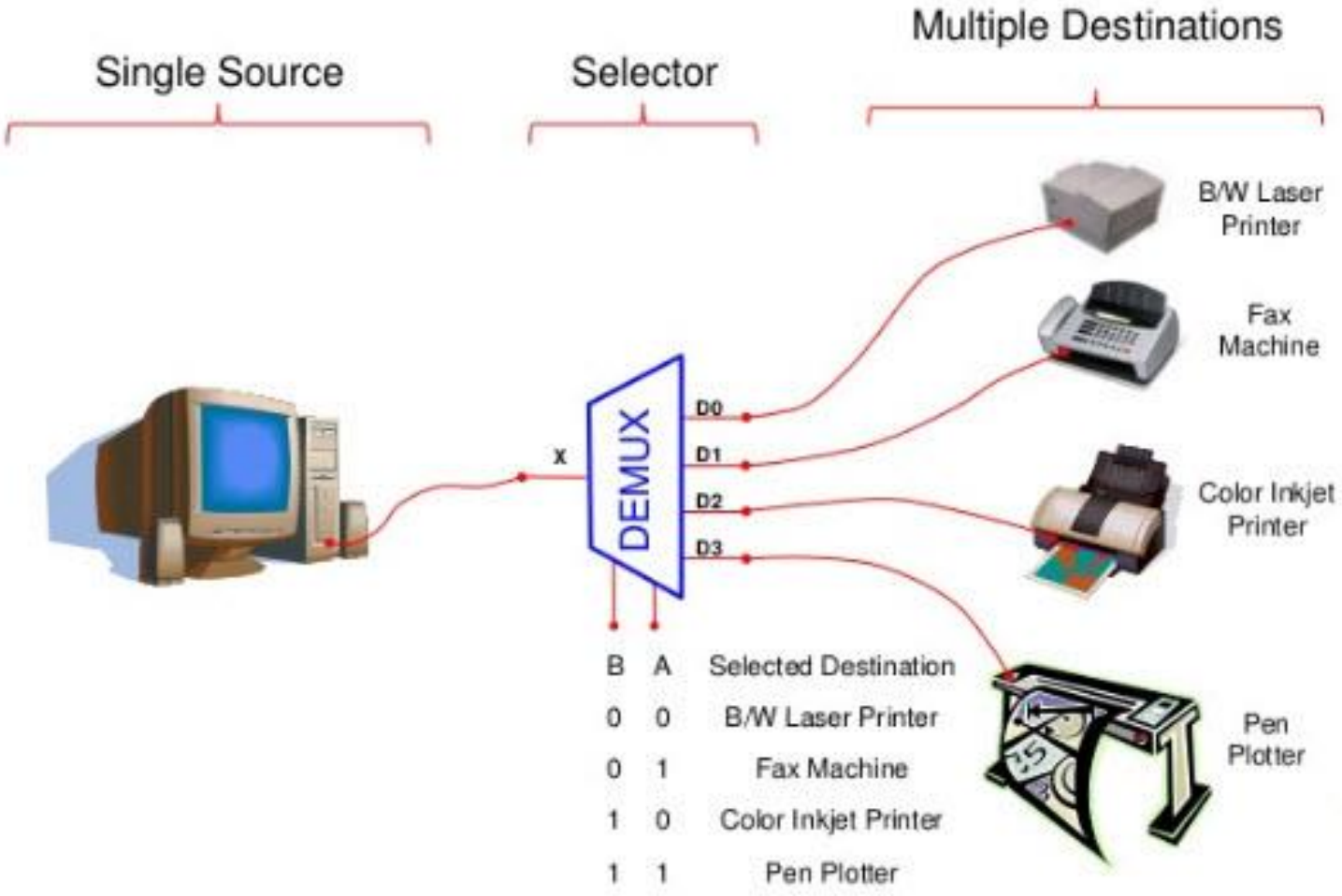
List of ICs which provide multiplexing

S.No.	IC No.	Function	Output State
1	74157	Quad 2:1 mux.	Output same as input given
2	74158	Quad 2:1 mux.	Output is inverted input
0	74153	Dual 4:1 mux.	Output same as input
5	74352	Dual 4:1 mux.	Output is inverted input
9	74151A	16:1 mux.	Both outputs available (i.e., complementary outputs)
6	74151	8:1 mux.	Output is inverted input
7	74150	16:1 mux.	Output is inverted input

What is a Demultiplexer (DEMUX)?

- A DEMUX is a digital switch with a single input (source) and a multiple outputs (destinations).
- The select lines determine which output the input is connected to.
- DEMUX Types
  - → 1-to-2 (1 select line)
  - → 1-to-4 (2 select lines)
  - → 1-to-8 (3 select lines)
  - → 1-to-16 (4 select lines)




Single Source
Selector
Multiple Destinations
B/W Laser Printer
Fax Machine
Color Inkjet Printer
Pen Plotter
DEMUX
X
D0
D1
D2
D3
B A Selected Destination
0 0 B/W Laser Printer
0 1 Fax Machine
1 0 Color Inkjet Printer
1 1 Pen Plotter

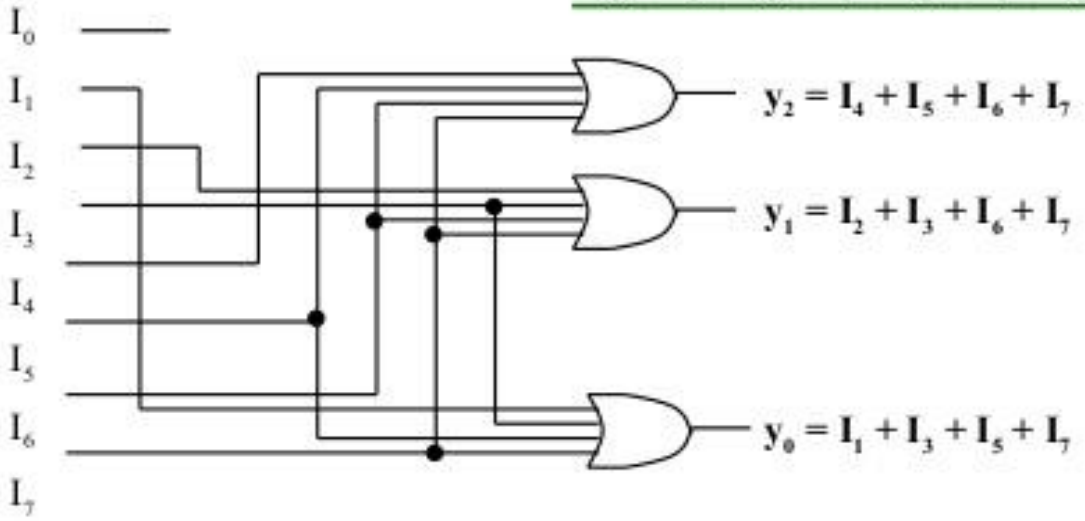
List of ICs which provide demultiplexing

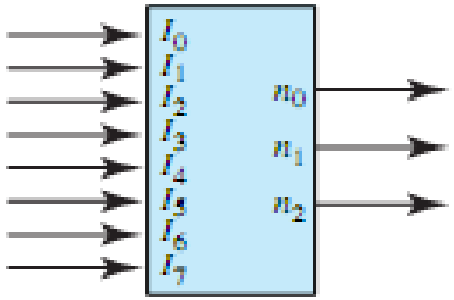
S.No.	IC No. (7400)	IC No. (4000)	Function	Output State
1	74139		Dual 1:4 demux.	Output is inverted input
3	74156		Dual 1:4 demux.	Output is open collector
4	74138		1:8 demux.	Output is inverted input
5	74238		1:8 demux.	
6	74154		1:16 demux.	Output is inverted input
7	74159	CD4514/15	1:16 demux.	Output is open collector and same as input

8-to-3 Binary Encoder

At any one time, only one input line has a value of 1.

Inputs								Outputs		
I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	y_2	y_1	y_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

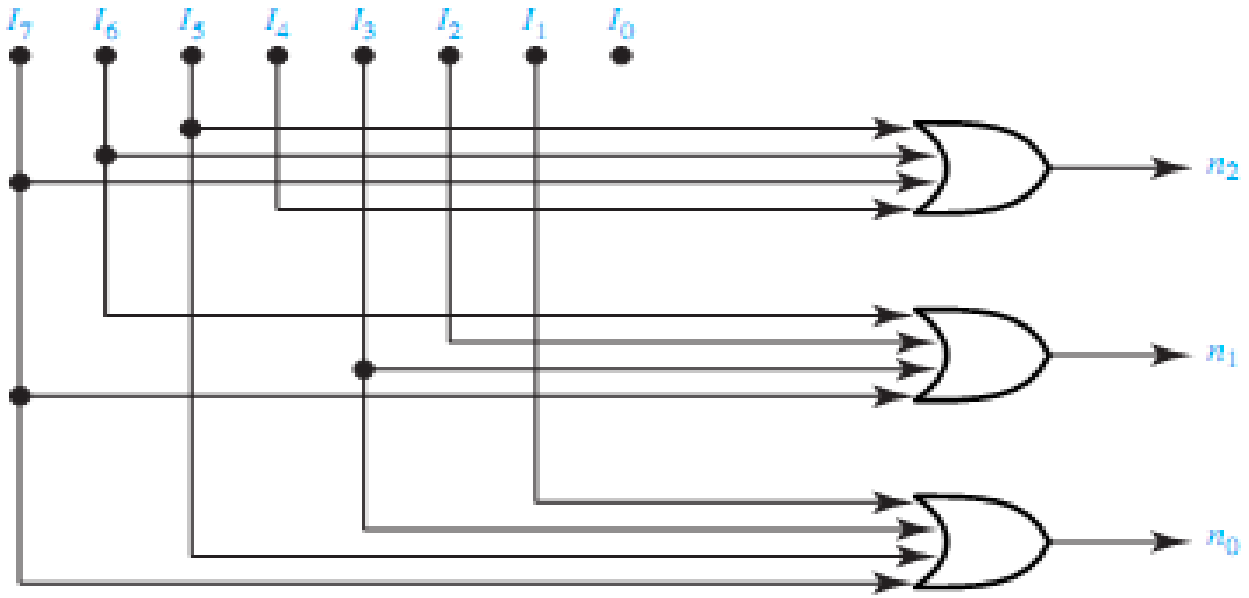




(a)

Inputs								Outputs		
I_7	I_6	I_5	I_4	I_3	I_2	I_1	I_0	n_2	n_1	n_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

(b)



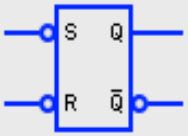
(c)

Figure 8-to-3 encoder. **(a)** Block diagram. **(b)** Truth table. **(c)** Logic diagram.

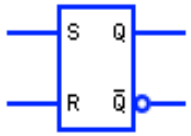
Unit 4

Flip-Flop Symbols

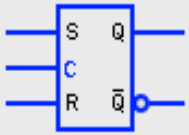
www.electricaltechnology.org



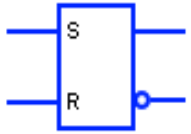
Active Low NAND SR Flipflop



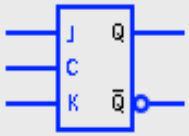
Active High NAND SR Flipflop



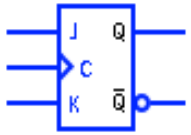
Clocked NAND SR Flipflop



SR Flip-Flop
Set Res et

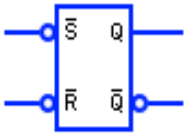


JK Flipflop High level Triggered



JK Flipflop Rising Edge Triggered

JK Flip-Flop
Activated by
the falling edge

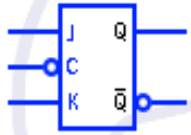


Active Low NOR SR Flipflop

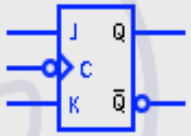
Active High NOR SR Flipflop

Clocked NOR SR Flipflop

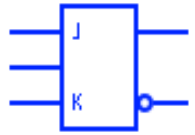
SR Flip-Flop Set Reset



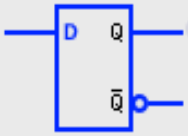
JK Flipflop Low level Triggered



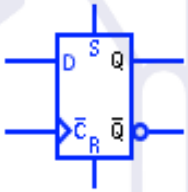
JK Flipflop Falling Edge Triggered



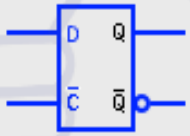
JK Flip-Flop Generic symbol



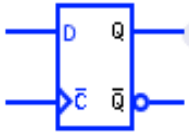
D Flipflop



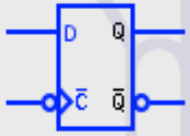
D Flipflop With Preset-Clear



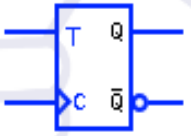
Gated D Flipflop



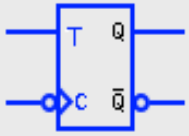
D Flipflop Rising Edge Triggered



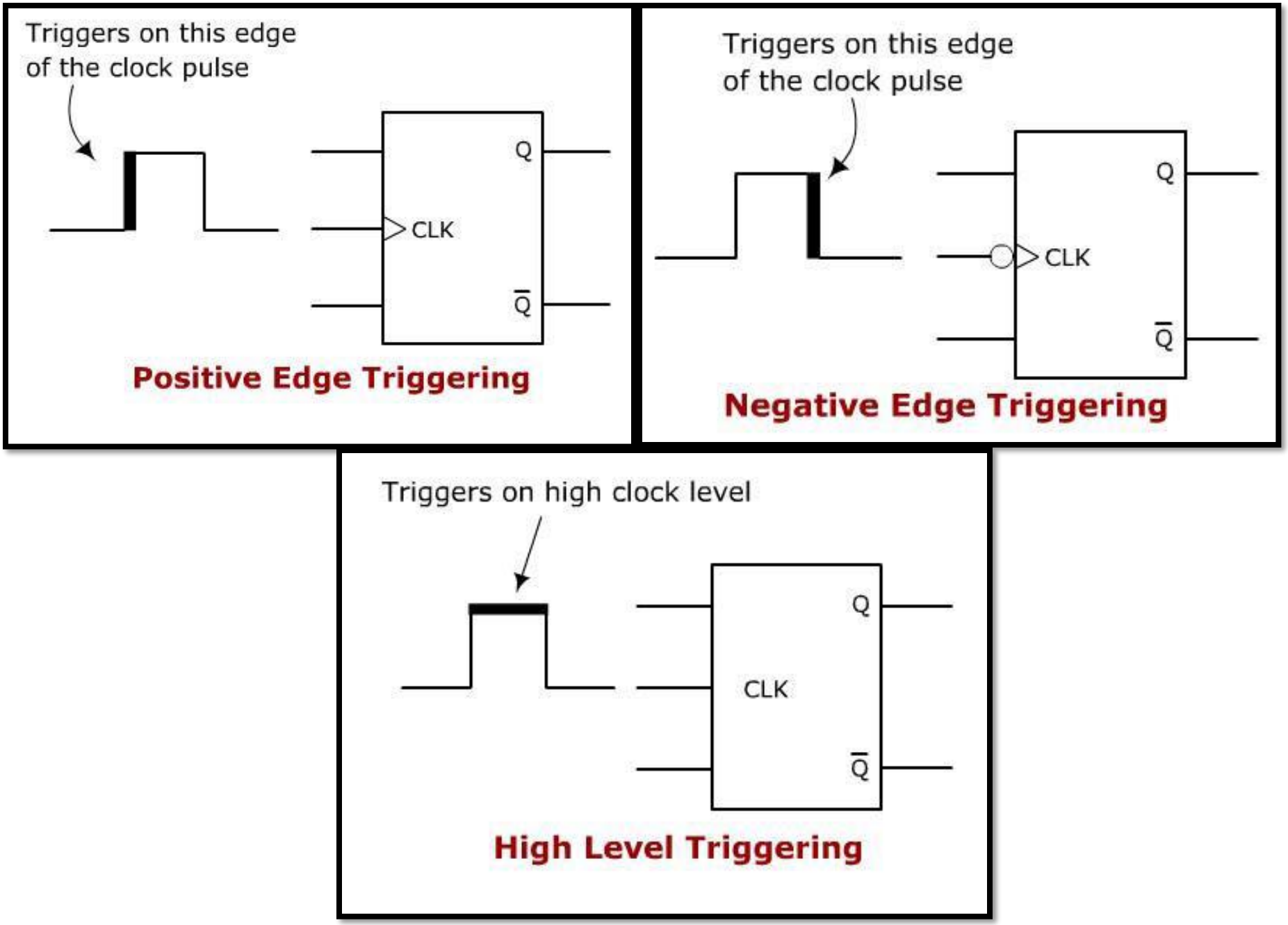
D Flipflop Falling Edge Triggered



T Flipflop Rising Edge Triggered



T Flipflop Falling Edge Triggered


Triggers on this edge
of the clock pulse
Q
CLK
Q̄
Positive Edge Triggering
Triggers on this edge
of the clock pulse
Q
CLK
Q̄
Negative Edge Triggering
Triggers on high clock level
Q
CLK
Q̄
High Level Triggering

- A trigger
  - The state of a latch or flip-flop is switched by a change of the control input
- Level triggered – latches
- Edge triggered – flip-flops

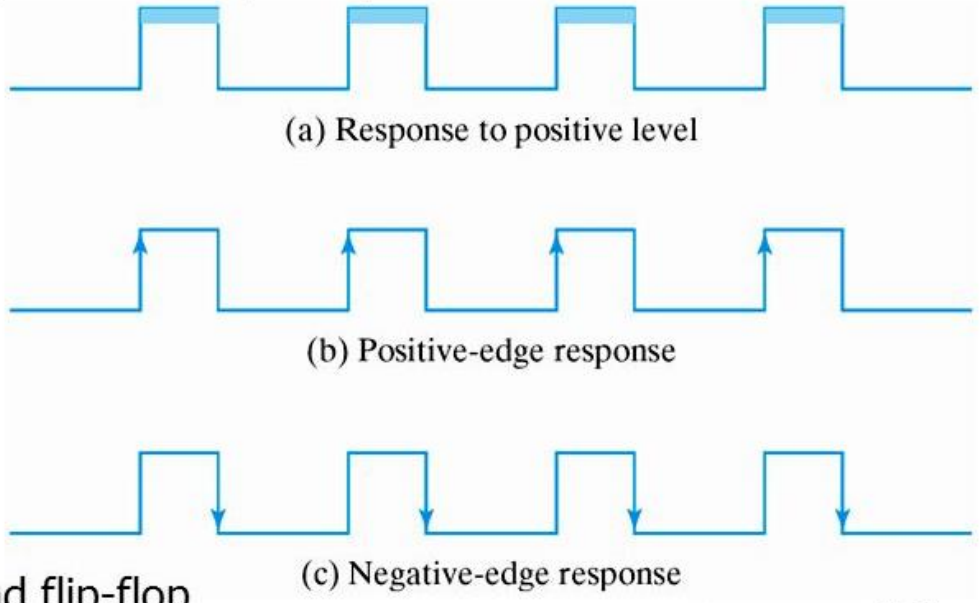
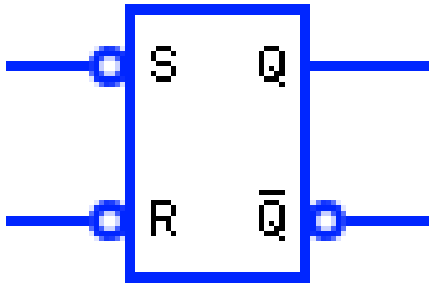


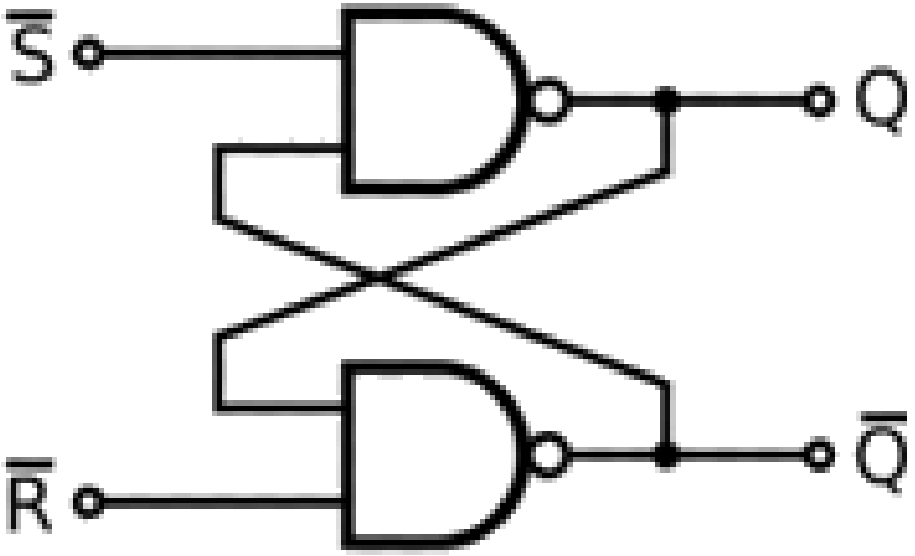
Fig. 5.8
Clock response in latch and flip-flop

A **latch** is an electronic logic **circuit** that has two inputs and one output. One of the inputs is called the SET input; the other is called the RESET input. **Latch circuits** can be either active-high or active-low.

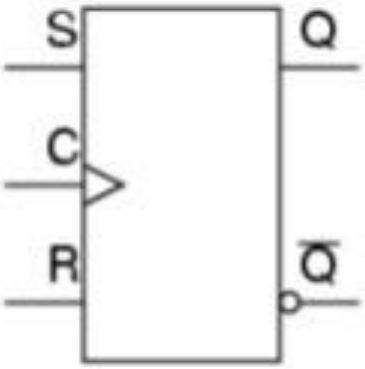
Flip Flop & Latches are sequential circuits & they are the building block of memory units. It stores a single bit of data. Sequential circuit's output depends not only on its current (Present) input but also on its previous output.

This SR Flip flop, also known as SR latch is an asynchronous (Independent of clock signal) sequential circuit made from only NAND gates. S-R represents the "set & reset" function of the flip flop. The bubbles at the input show that it is **Active Low**.





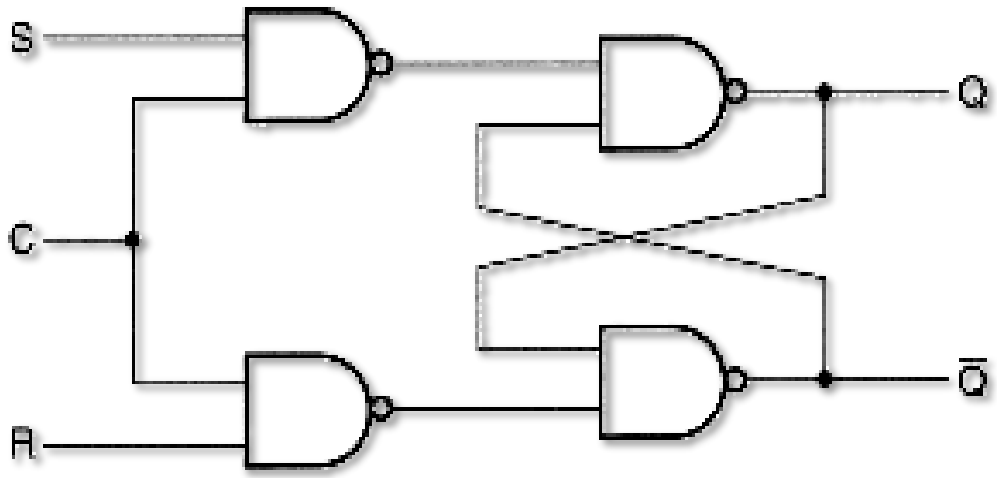
SR Flip-Flop



S	R	Q_{t+1}	
0	0	Q_t	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	?	Indeterminate

- S – Set, R – Reset
- Output changes only when a clock pulse is applied
- When input is 11, output is unpredictable
  - May change to 0 or 1 depending internal delays
- SR should not be pulsed when input is 11
  - Preventing this is difficult → rarely used in circuits

SR flip flop



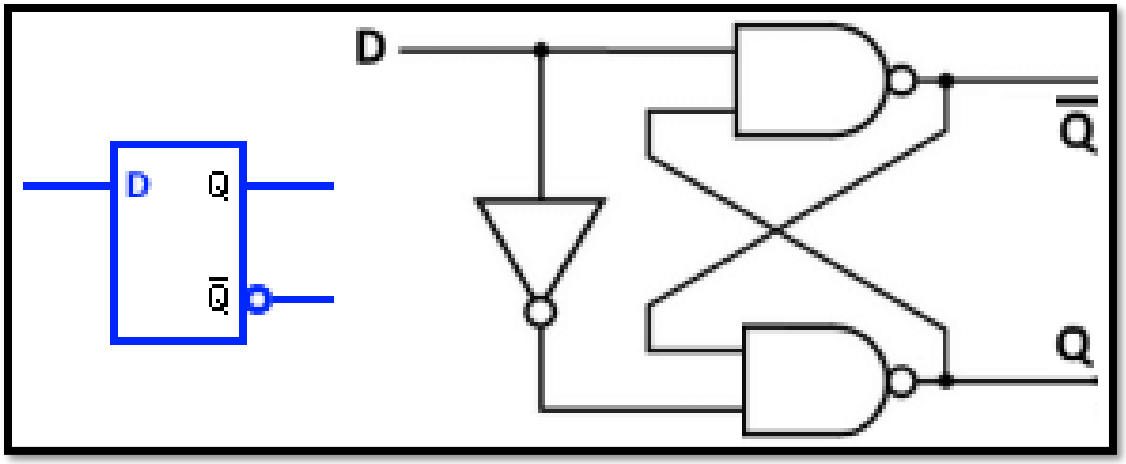
(a) Logic diagram

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

(b) Function table

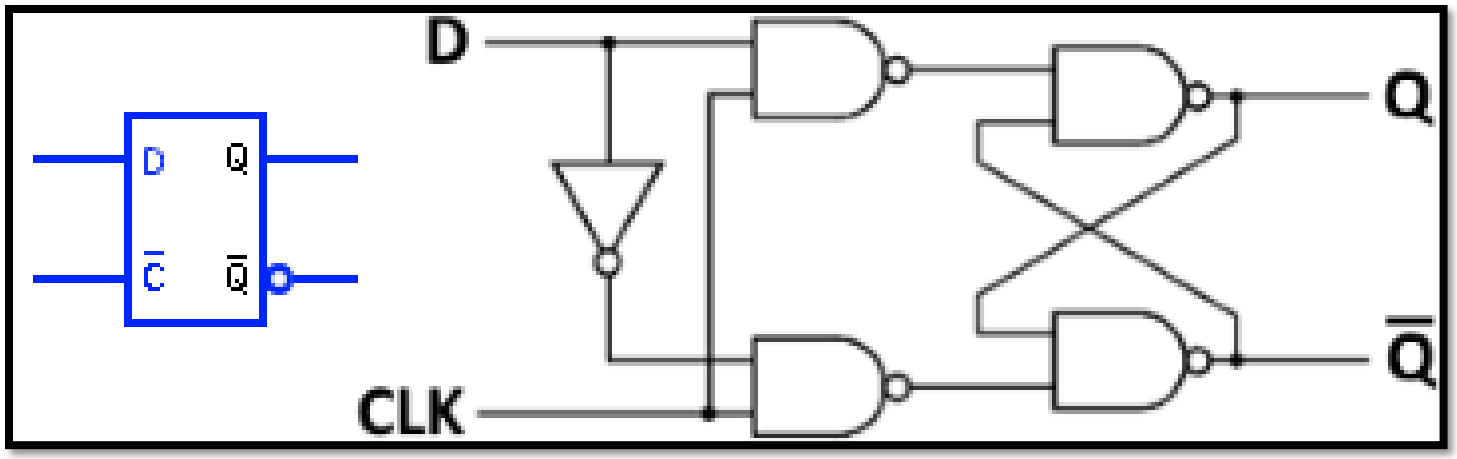
D flip-flop is also known as “DATA” or “DELAY” flip-flop. It is a modified version of SR Flip-flop with a single common input D. It stores a single data bit from the input line D.

When D = 0, the flip flop **reset** & the output Q becomes 0.
When D = 1, the output is **set** & Q becomes 1.



D	State	Q_{next}	Q’
0	Reset	0	1
1	Set	1	0

Gated D flip flop or also known as level triggered D flip flop has an extra control input known as “Enable” or “clock” input. when the CLK = 0, the D flip-flop holds is previous state. When the CLK=1, it operate as a normal D flip-flop.



CLK	D	State	Q_{next}	Q’
0	X	Hold	Previous State	Previous State
1	0	Reset	0	1
1	1	Set	1	0

D Flip-flop

Symbol

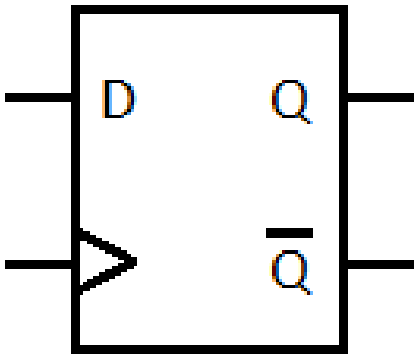


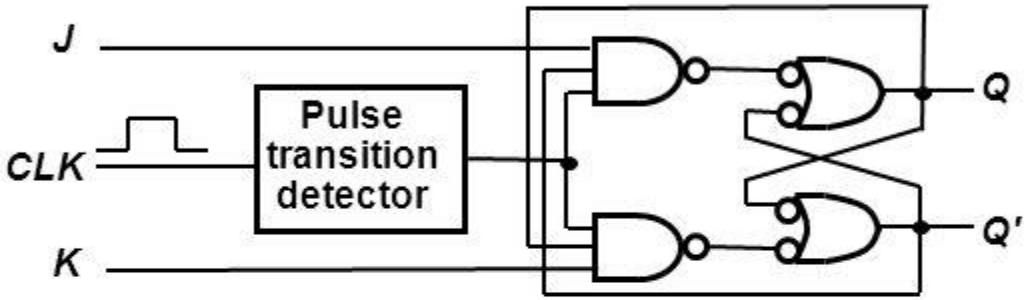
Table of truth:

clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

Input D_n	**Output** Q_{n+1}
0	0
1	1

J-K FLIP-FLOP (2/2)

- *J-K* flip-flop circuit:



- Characteristic table:

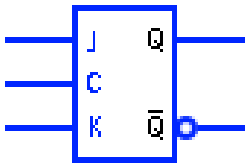
J	K	CLK	Q(t+1)	Comments
0	0	↑	Q(t)	No change
0	1	↑	0	Reset
1	0	↑	1	Set
1	1	↑	Q(t)'	Toggle

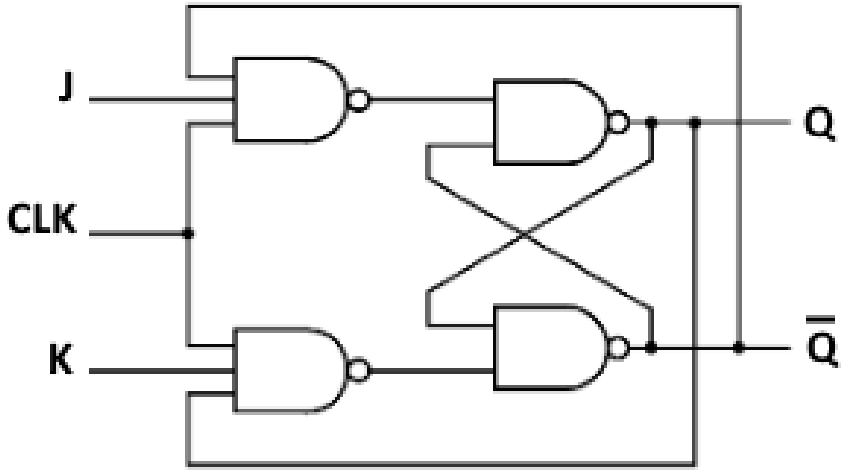
Q(t+1) = ?

Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

The JK flip flop is a modified version of SR flip-flop. The forbidden (Invalid) input in the SR flip flop is is used in JK flip-flop for the toggle function. Apart from toggle function the JK flip-flop works the same as SR flip-flop.

As this is a high level triggered flip-flop, the CLK signal will activate the flip-flop when CLK = 1. The flip-flop holds its state when CLK = 0.



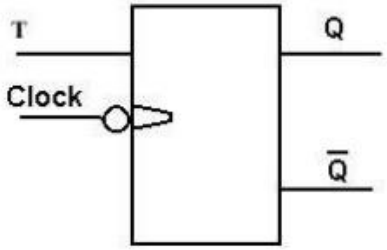


Truth Table

J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

T Flip-Flop

The T flip-flop is obtained from a JK flip-flop when inputs J and K are connected to provide a single input designated by T.



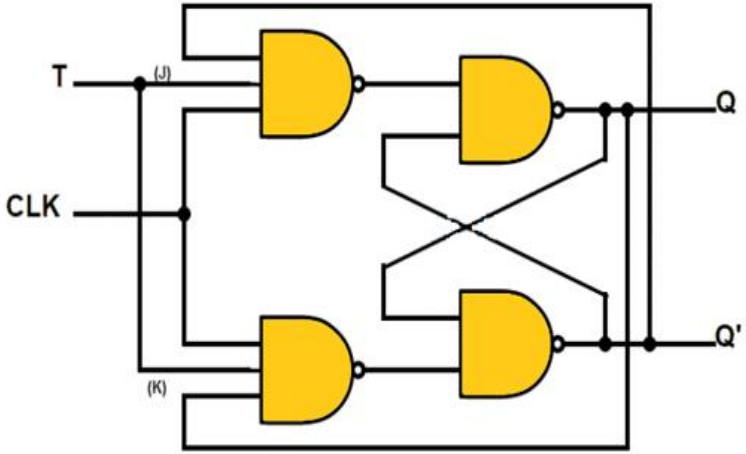
Graphic Symbol

The flip-flop thus has only two conditions.

(d) *T* Flip-Flop

T	$Q(t+1)$	Operation
0	$Q(t)$	No change
1	$\overline{Q}(t)$	Complement

Truth Table



T	Q	Q'
0	0	0
1	0	1
0	1	0
1	1	0

Race Around Condition In JK Flip-flop – For J-K flip-flop, if J=K=1, and if clk=1 for a long period of time, then Q output will toggle as long as CLK is high, which makes the output of the flip-flop unstable or uncertain. This problem is called race around condition in J-K flip-flop. This problem (Race Around Condition) can be avoided by ensuring that the clock input is at logic "1" only for a very short time. This introduced the concept of **Master Slave JK** flip flop.

Master Slave JK flip flop –

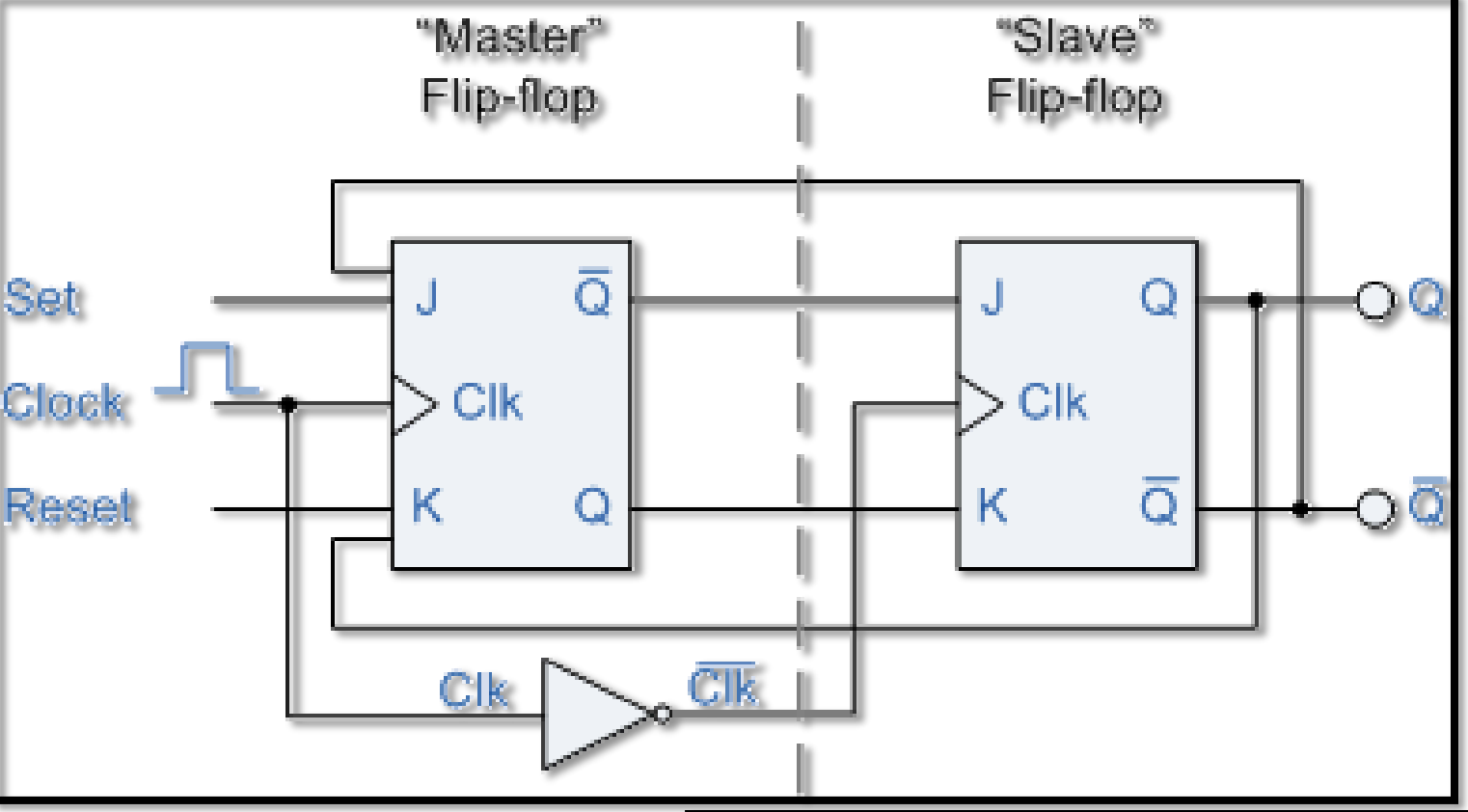
The Master-Slave Flip-Flop is basically a combination of two JK flip-flops connected together in a series configuration. Out of these, one acts as the **"master"** and the other as a **"slave"**. The output from the master flip flop is connected to the two inputs of the slave flip flop whose output is fed back to inputs of the master flip flop.

In addition to these two flip-flops, the circuit also includes an **inverter**. The inverter is connected to clock pulse in such a way that the inverted clock pulse is given to the slave flip-flop. In other words if CP=0 for a master flip-flop, then CP=1 for a slave flip-flop and if CP=1 for master flip flop then it becomes 0 for slave flip flop.

Working of a master slave flip flop –

When the clock pulse goes to 1, the slave is isolated; J and K inputs may affect the state of the system. The slave flip-flop is isolated until the CP goes to 0. When the CP goes back to 0, information is passed from the master flip-flop to the slave and output is obtained. Firstly the master flip flop is positive level triggered and the slave flip flop is negative level triggered, so the master responds before the slave.

If J=0 and K=1, the high Q' output of the master goes to the K input of the slave and the clock forces the slave to reset, thus the slave copies the master.
If J=1 and K=0, the high Q output of the master goes to the J input of the slave and the Negative transition of the clock sets the slave, copying the master.
If J=1 and K=1, it toggles on the positive transition of the clock and thus the slave toggles on the negative transition of the clock.
If J=0 and K=0, the flip flop is disabled and Q remains unchanged.



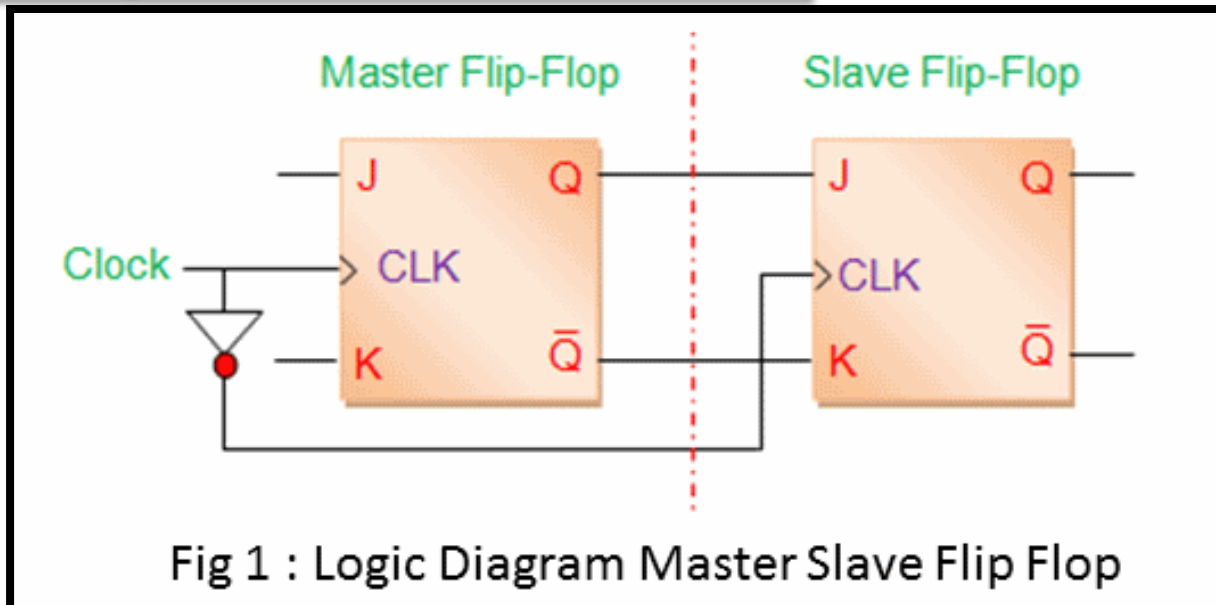
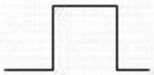
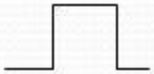
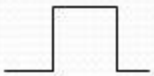


Fig 1 : Logic Diagram Master Slave Flip Flop

Truth Table for Master-Slave J-K Flip Flop

J	K	CLK	Q	Q'	
0	0		Q_0	Q_0'	Hold
0	1		0	1	Reset
1	0		1	0	Set
1	1		Q_0'	Q_0	Toggle (opposite state)

Latches	Flip Flops
Latches are building blocks of sequential circuits and these can be built from logic gates	Flip flops are also building blocks oof sequential circuits. But, these can be built from the latches.
Latch continuously checks its inputs and changes its output correspondingly.	Flip flop continuously checks its inputs and changes its output correspondingly only at times determined by clocking signal
The latch is sensitive to the duration of the pulse and can send or receive the data when the switch is on	Flipflop is sensitive to a signal change. They can transfer data only at the single instant and data cannot be changed until next signal change. Flip flops are used as a register.
It is based on the enable function input	It works on the basis of clock pulses
It is a level triggered, it means that the output of the present state and input of the next state depends on the level that is binary input 1 or 0.	It is an edge triggered, it means that the output and the next state input changes when there is a change in clock pulse whether it may a +ve or -ve clock pulse.

Flip-flop	Latch
A flip-flop samples the inputs only at a clock event (rising edge, etc.)	A Latch samples the inputs continuously *whenever it is enabled*, that is, only when the enable signal is on. (or otherwise, it would be a wire, not a latch).
Flip-Flop are edge sensitive.	Latches are level sensitive.
Flipflop is sensitive to signal change and not on level. They can transfer data only at the single instant and data cannot be changed until next signal change.	Latch is sensitive to duration of pulse and can send or receive the data when the switch is on.
A flip-flop continuously checks its inputs and correspondingly changes its output only at times determined by clocking signal.	Latch is a device which continuously checks all its input and correspondingly changes its output, independent of the time determined by clocking signal.
It work's on the basis of clock pulses.	It is based on enable function input
It is a edge triggered, it mean that the output and the next state input changes when there is a change in clock pulse whether it may a +ve or -ve clock pulse.	It is a level triggered, it mean that the output of present state and input of the next state depends on the level that is binary input 1 or 0.

Applications of Flip Flops

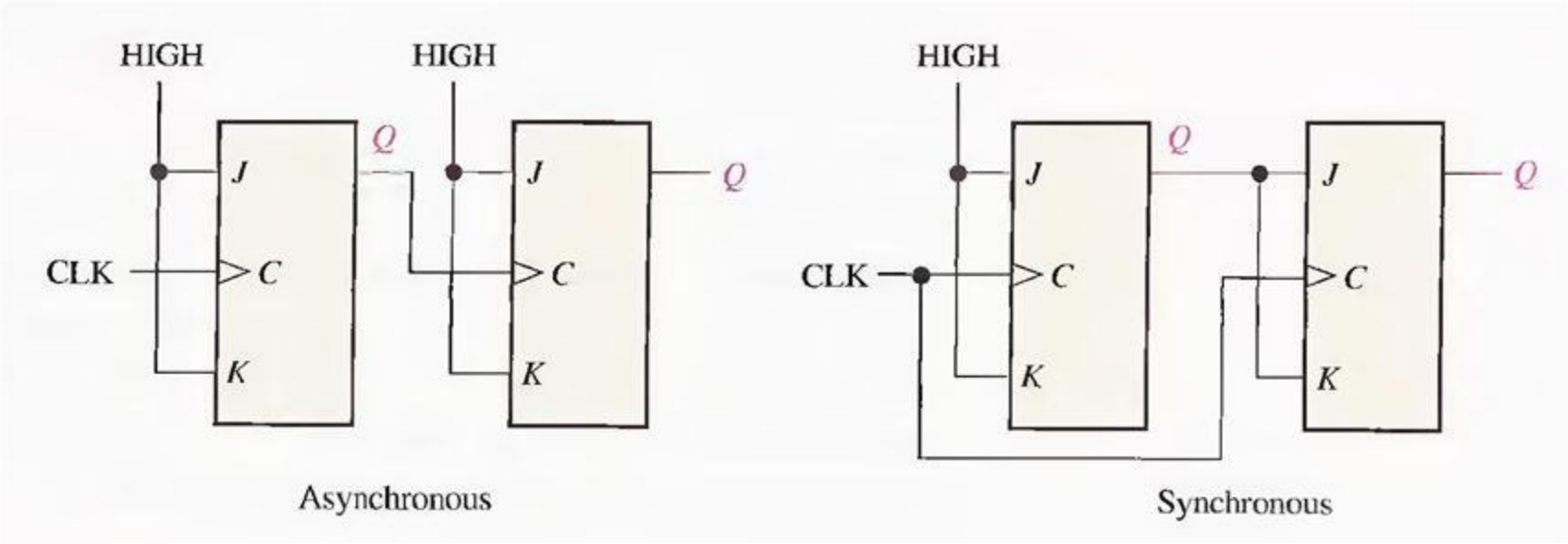
- Flip flops will find their use in many of the fields in digital electronics. Particularly, edge triggered flip flops are very resourceful devices that can be used in wide range of applications like storing of binary data, counter, transferring binary data from one location to other etc. Some of the most common applications of flip – flops are
- Registers
- Counters
- Event Detectors
- Frequency Divider circuits
- All these applications make use of the flip – flop's clocked operation.

Introduction –COUNTERS

- A *counter* is a register that goes through a predetermined sequence of states upon the application of clock pulses.
  - Asynchronous counters
  - Synchronous counters
- **Asynchronous Counters** (or *Ripple counters*)
  - the clock signal (CLK) is only used to clock the first FF.
  - Each FF (except the first FF) is clocked by the preceding FF.
- **Synchronous Counters**,
  - the clock signal (CLK) is applied to all FF, which means that all FF shares the same clock signal,
  - thus the output will change at the same time.

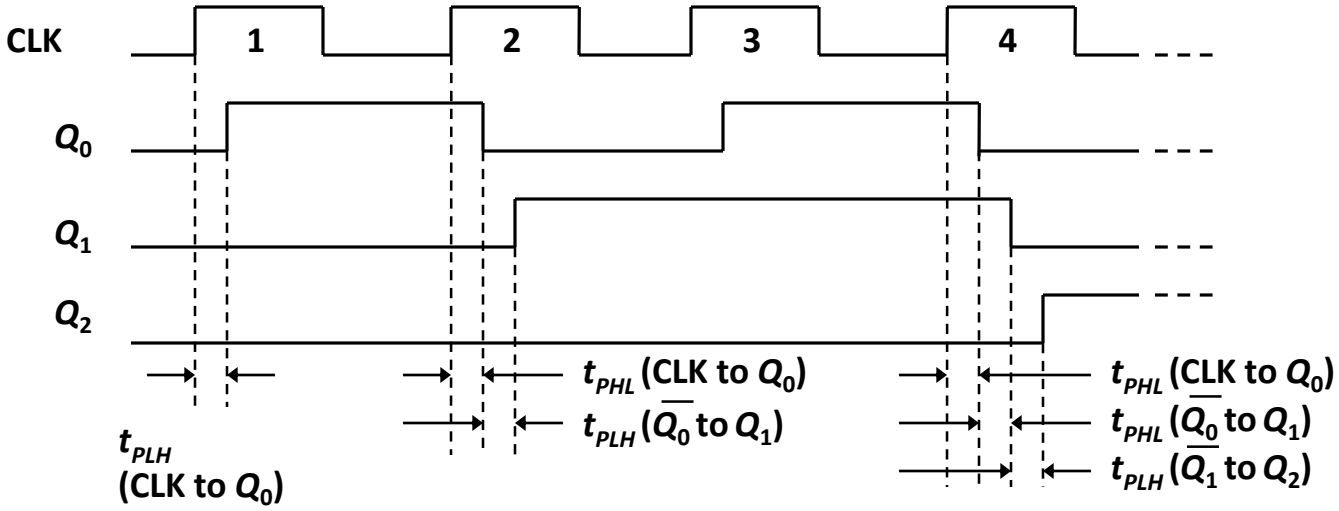
Synchronous Counter	Asynchronous Counter
The counter is clocked in such a way that each flip-flop in the counter is triggered by the same clock signal at the same time.	Only the first flip-flop is clocked by an external clock which in turn drives the clock output of the following flip-flop.
All the flip-flops in the counter change state at the same time in sync with the input clock signal.	The clock input of the flip-flops is not driven by the same clock signal.
It can operate at much higher clock frequencies than its asynchronous counterpart.	It is slower in operation than synchronous counters.
The design involves a complex logic circuit as the number of states increases.	Logic circuit is quite simple even for more number of states.
There is no inherent propagation delay in synchronous counters.	A subsequent propagation delay is encountered from one flip-flop to another.

•Counters can be implemented in either asynchronous and synchronous operated form

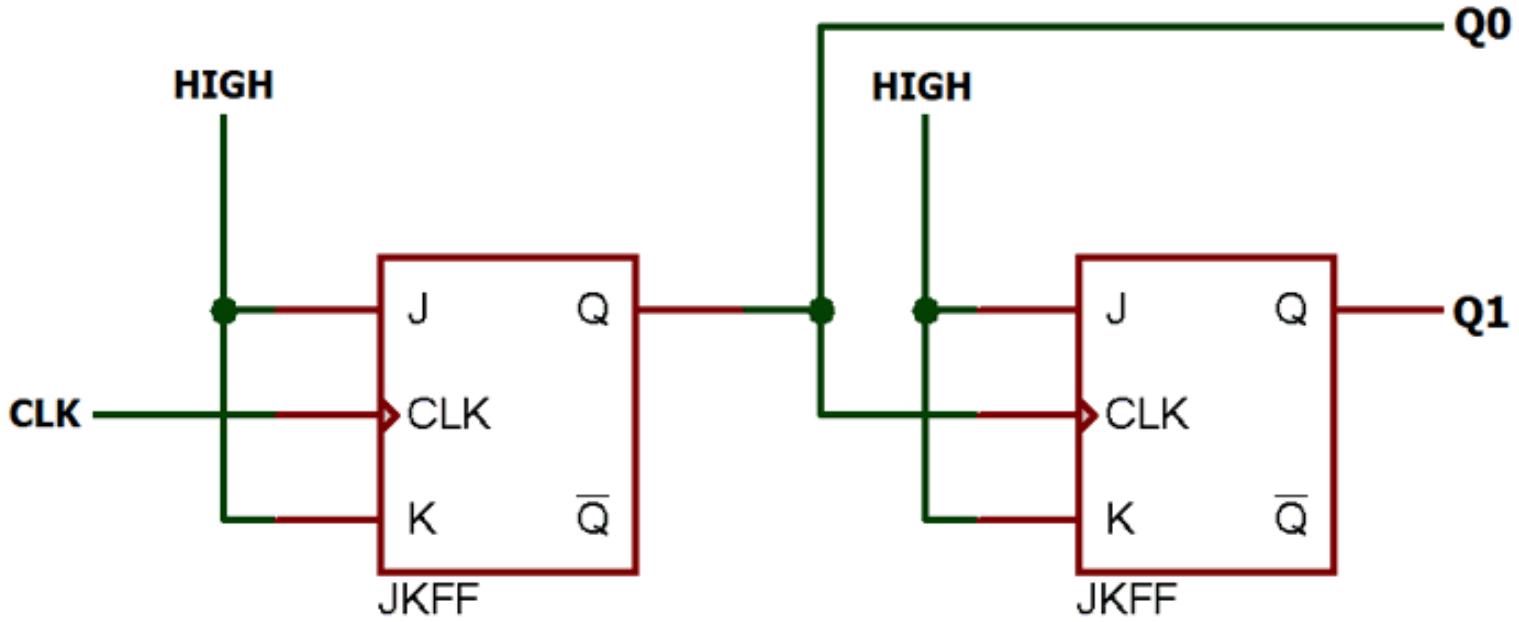


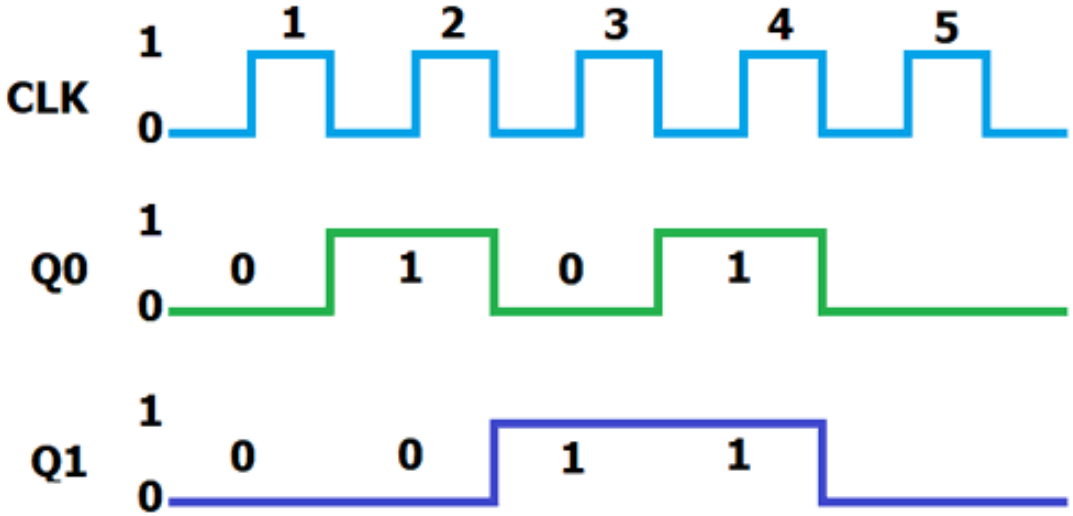
Asynchronous (Ripple) Counters

- Propagation delays in an asynchronous (ripple-clocked) binary counter.
- If the accumulated delay is greater than the clock pulse, some counter states may be misrepresented!



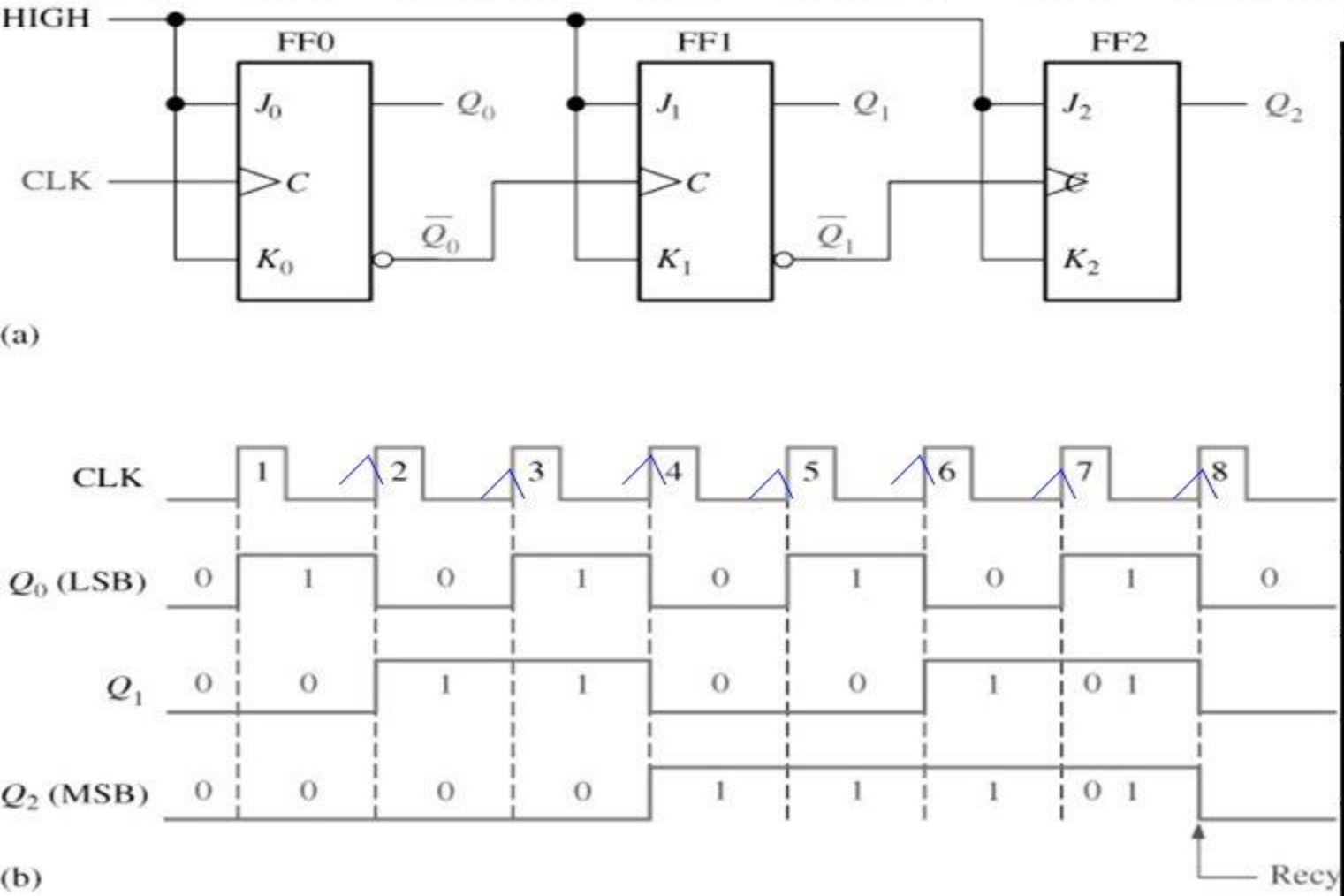
2-Bit Asynchronous Counter





Asynchronous/Ripple Counter

Three-bit asynchronous binary counter and its timing diagram for one cycle.

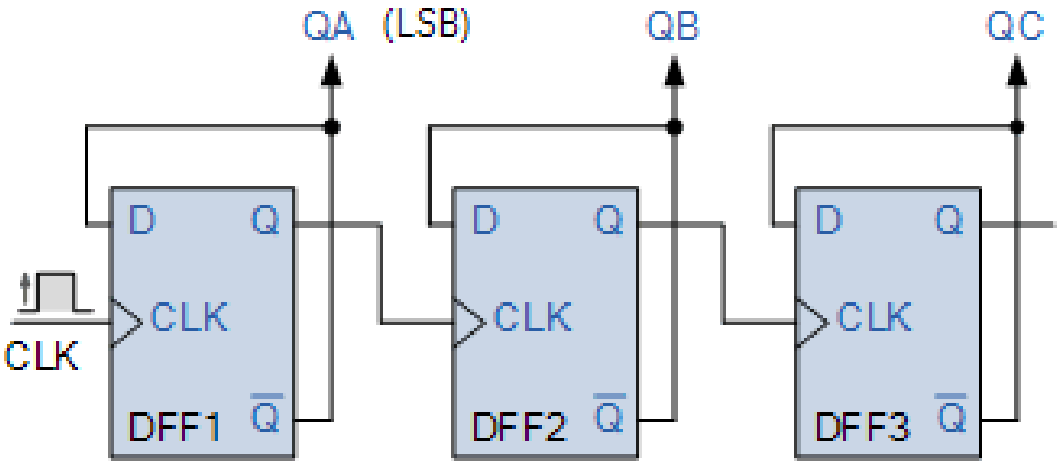


Clk pulse	Q2	Q1	Q0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (REPEAT)	0	0	0

RIPPLE COUNTER UP – PGT AND ALL NON FIRST CLK RECEIVE CLK PLUSE FROM Q'

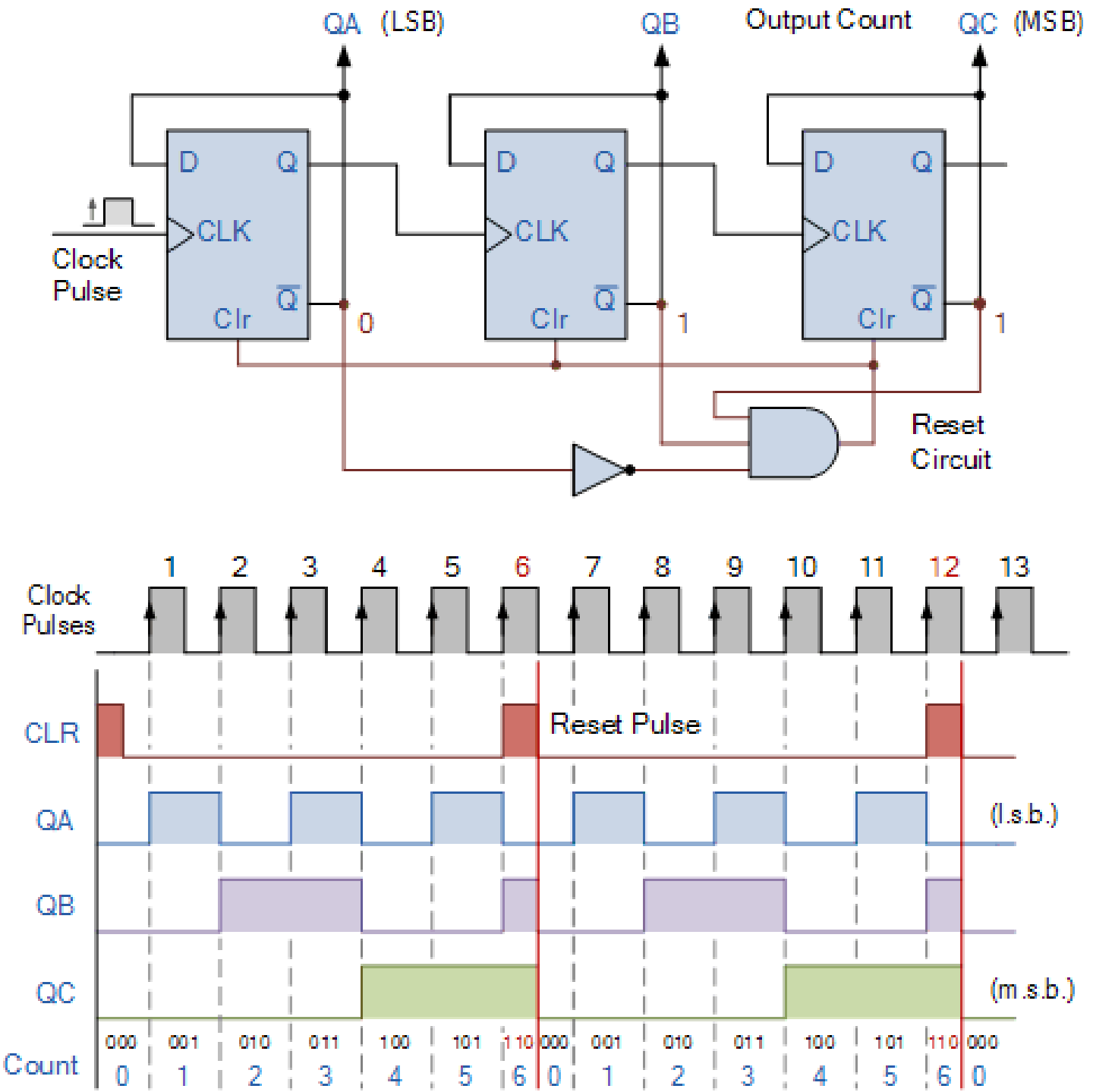
A Modulo-5 Counter

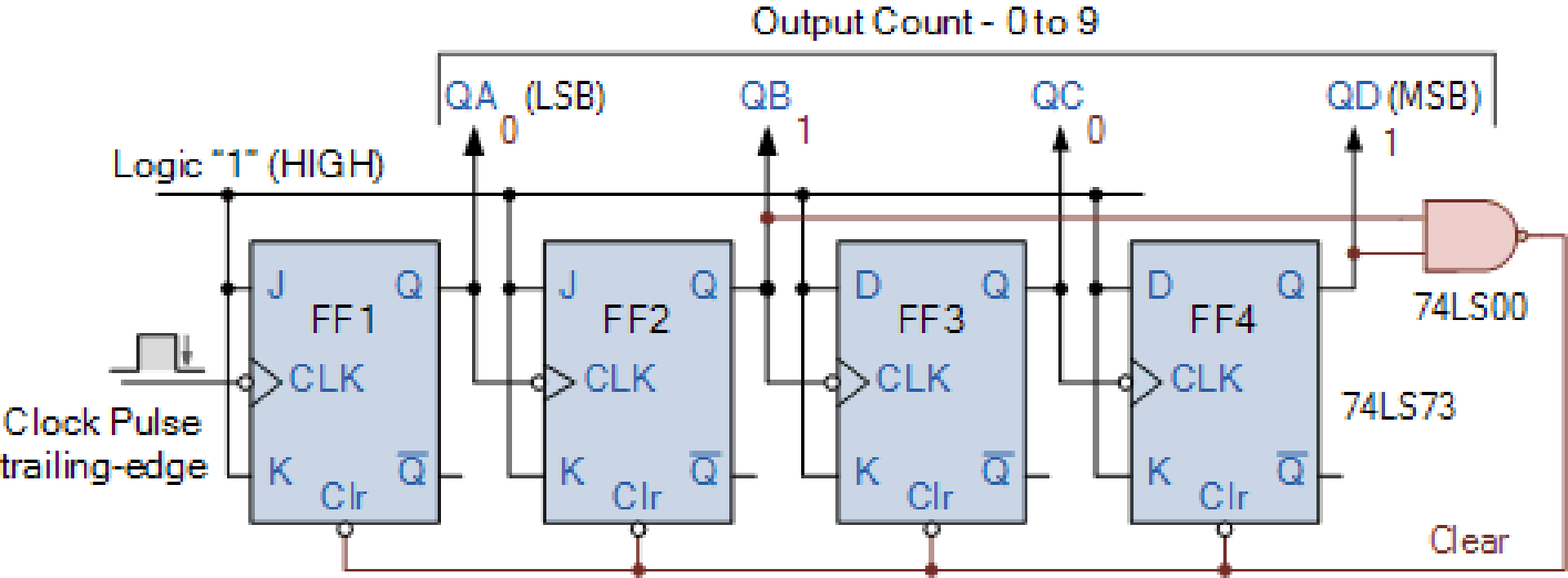
Suppose we want to design a MOD-5 counter, how could we do that. First we know that "m = 5", so 2^n must be greater than 5. As $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, and 8 is greater than 5, then we need a counter with three flip-flops (N = 3) giving us a natural count of 000 to 111 in binary (0 to 7 decimal).



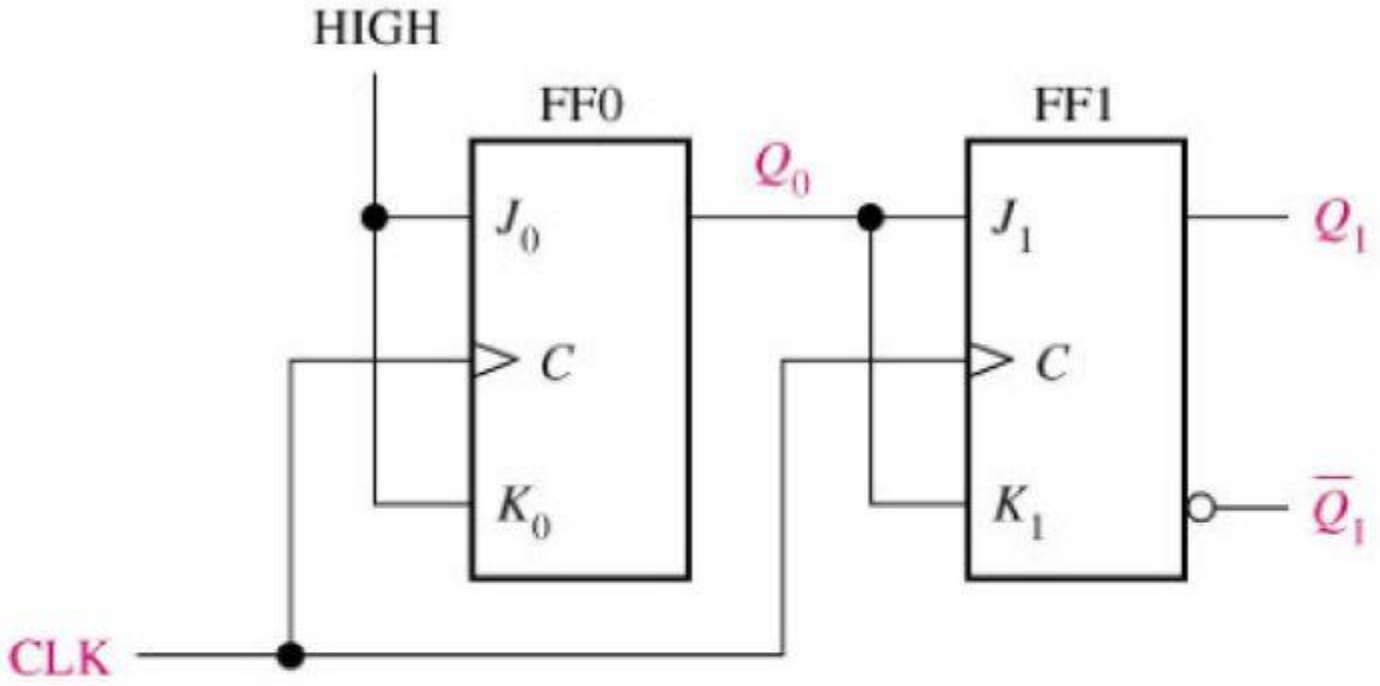
Truth Table

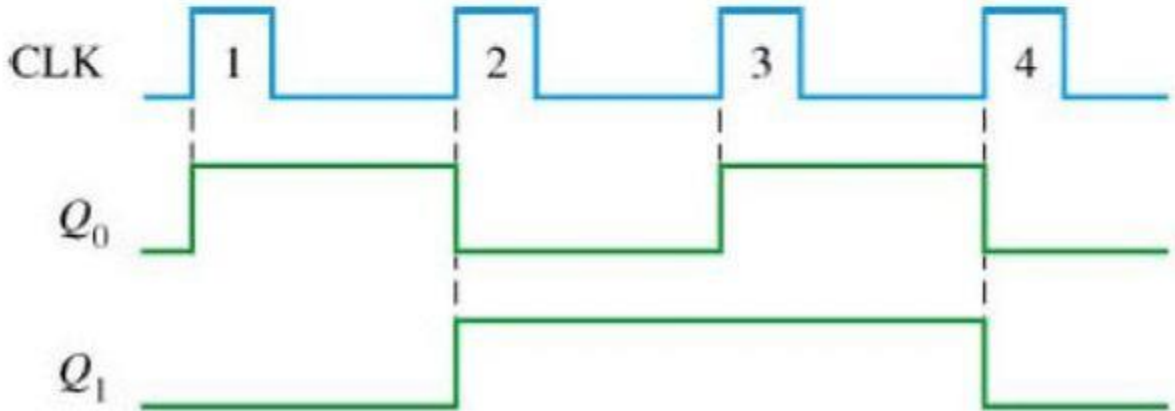
C	QC	QB	QA
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0


QA (LSB)
QB
Output Count
QC (MSB)
D
Q
CLK
Clr
Q̄
Clock
Pulse
0
1
1
Reset
Circuit
Clock
Pulses
1
2
3
4
5
6
7
8
9
10
11
12
13
CLR
Reset Pulse
QA
(l.s.b.)
QB
QC
(m.s.b.)
Count
000
001
010
011
100
101
110
000
001
010
011
100
101
110
000
0
1
2
3
4
5
6
0
1
2
3
4
5
6
0


Output Count - 0 to 9
QA (LSB)
0
QB
1
QC
0
QD (MSB)
1
Logic "1" (HIGH)
J
Q
FF1
CLK
K
Clr
Q
J
Q
FF2
CLK
K
Clr
Q
D
Q
FF3
CLK
K
Clr
Q
D
Q
FF4
CLK
K
Clr
Q
74LS00
74LS73
Clock Pulse
trailing-edge
Clear

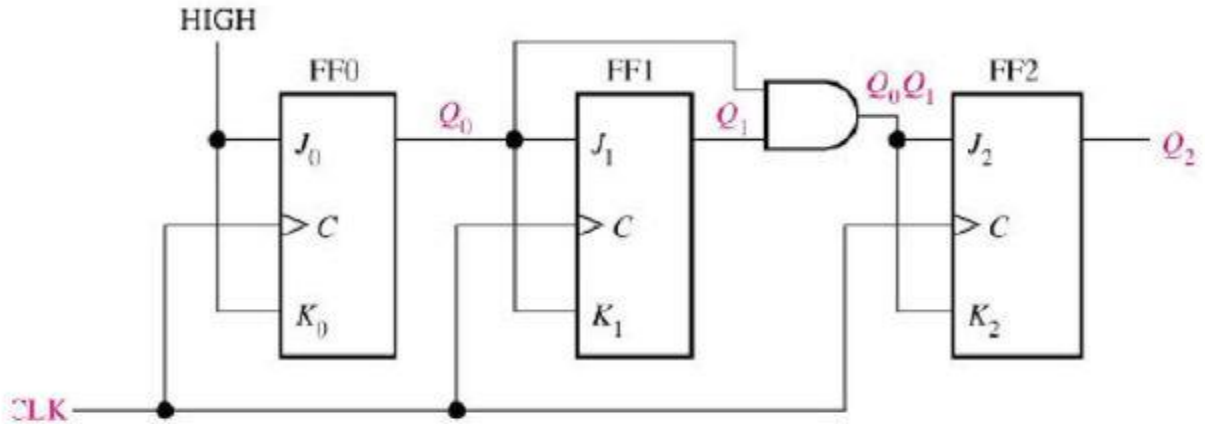
2-bit Synchronous Counter

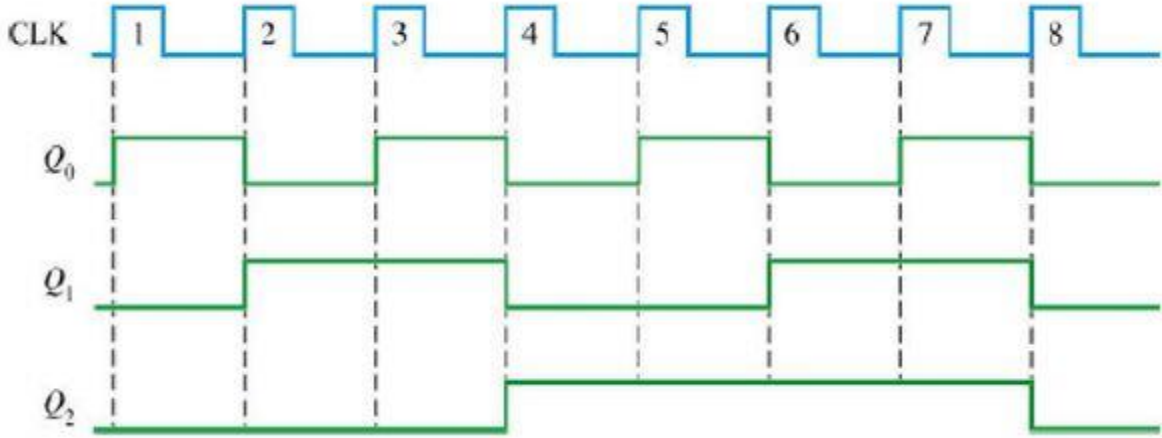




3 bit synchronous counter

- Step 7: Draw circuit diagram

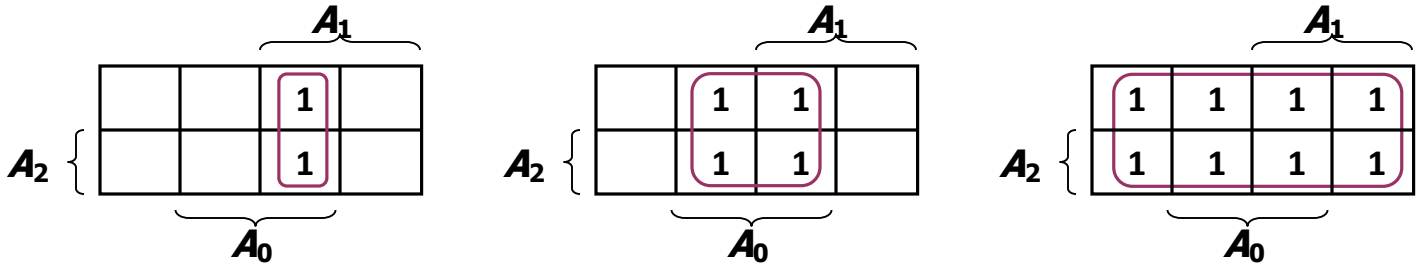




Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J, K inputs).

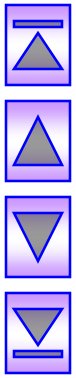
Present state			Next state			Flip-flop inputs		
A_2	A_1	A_0	A_2^+	A_1^+	A_0^+	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



$TA_2 = A_1.A_0$ $TA_1 = A_0$ $TA_0 = 1$

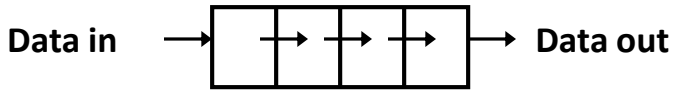
Introduction: Registers

- An *n*-bit register has a group of *n* flip-flops and some logic gates and is capable of storing *n* bits of information.
- The flip-flops store the information while the gates control when and how new information is transferred into the register.
- Some functions of register:
  - retrieve data from register
  - store/load new data into register (serial or parallel)
  - shift the data within register (left or right)

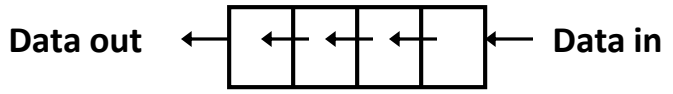


Shift Registers

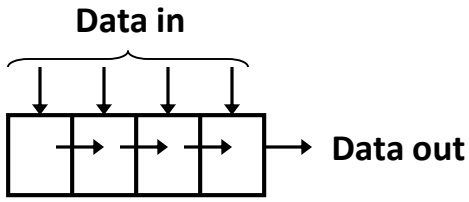
- Basic data movement in shift registers (four bits are used for illustration).



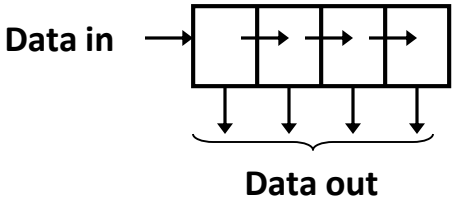
(a) Serial in/shift right/serial out



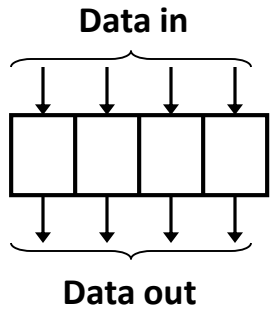
(b) Serial in/shift left/serial out



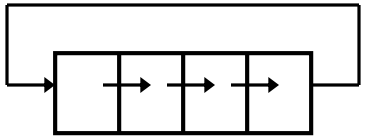
(c) Parallel in/serial out



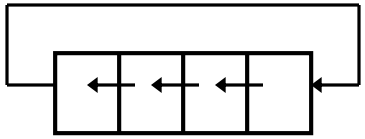
(d) Serial in/parallel out



(e) Parallel in / parallel out



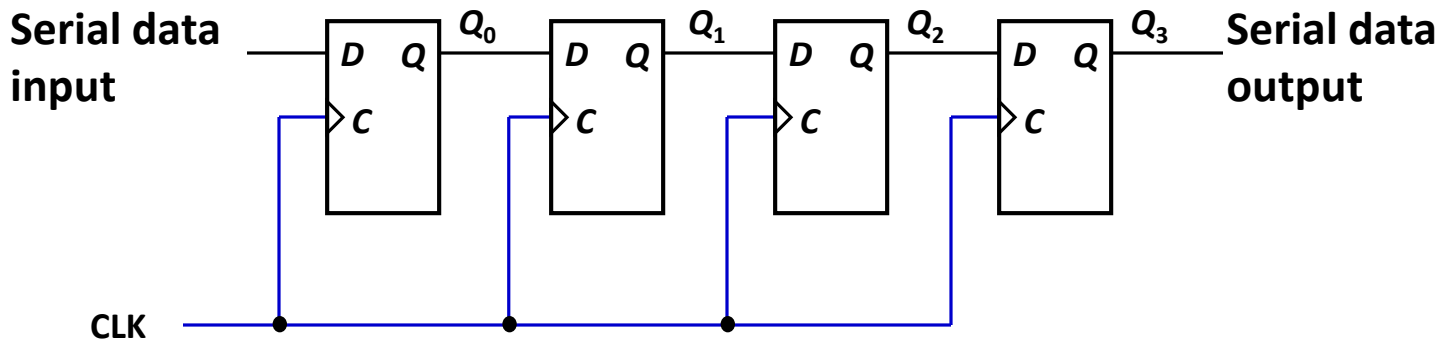
(f) Rotate right



(g) Rotate left

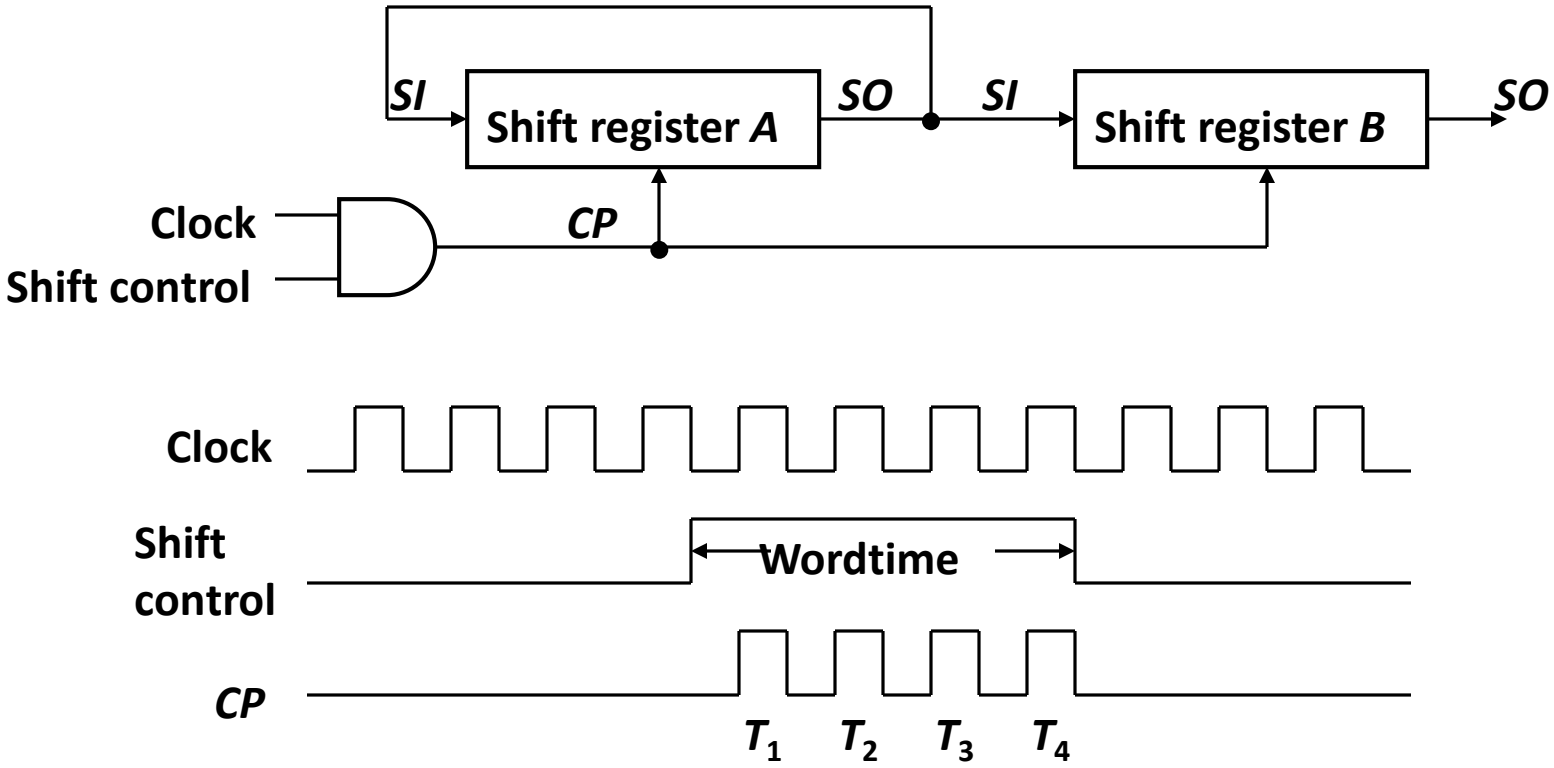
Serial In/Serial Out Shift Registers

- Accepts data serially – one bit at a time – and also produces output serially.



Serial In/Serial Out Shift Registers

- Application: Serial transfer of data from one register to another.



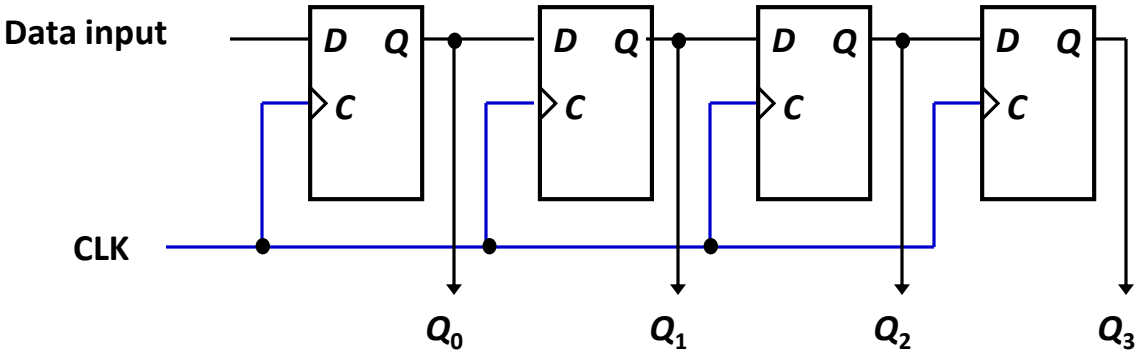
Serial In/Serial Out Shift Registers

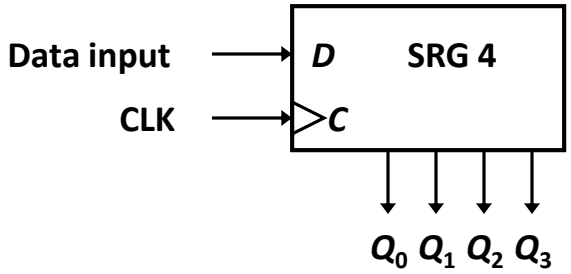
- Serial-transfer example.

Timing Pulse	Shift register *A*				Shift register *B*				Serial output of *B*
Initial value	1	0	1	1	0	0	1	0	0
After T_1	1	1	0	1	1	0	0	1	1
After T_2	1	1	1	0	1	1	0	0	0
After T_3	0	1	1	1	0	1	1	0	0
After T_4	1	0	1	1	1	0	1	1	1

Serial In/Parallel Out Shift Registers

- Accepts data serially.
- Outputs of all stages are available simultaneously.

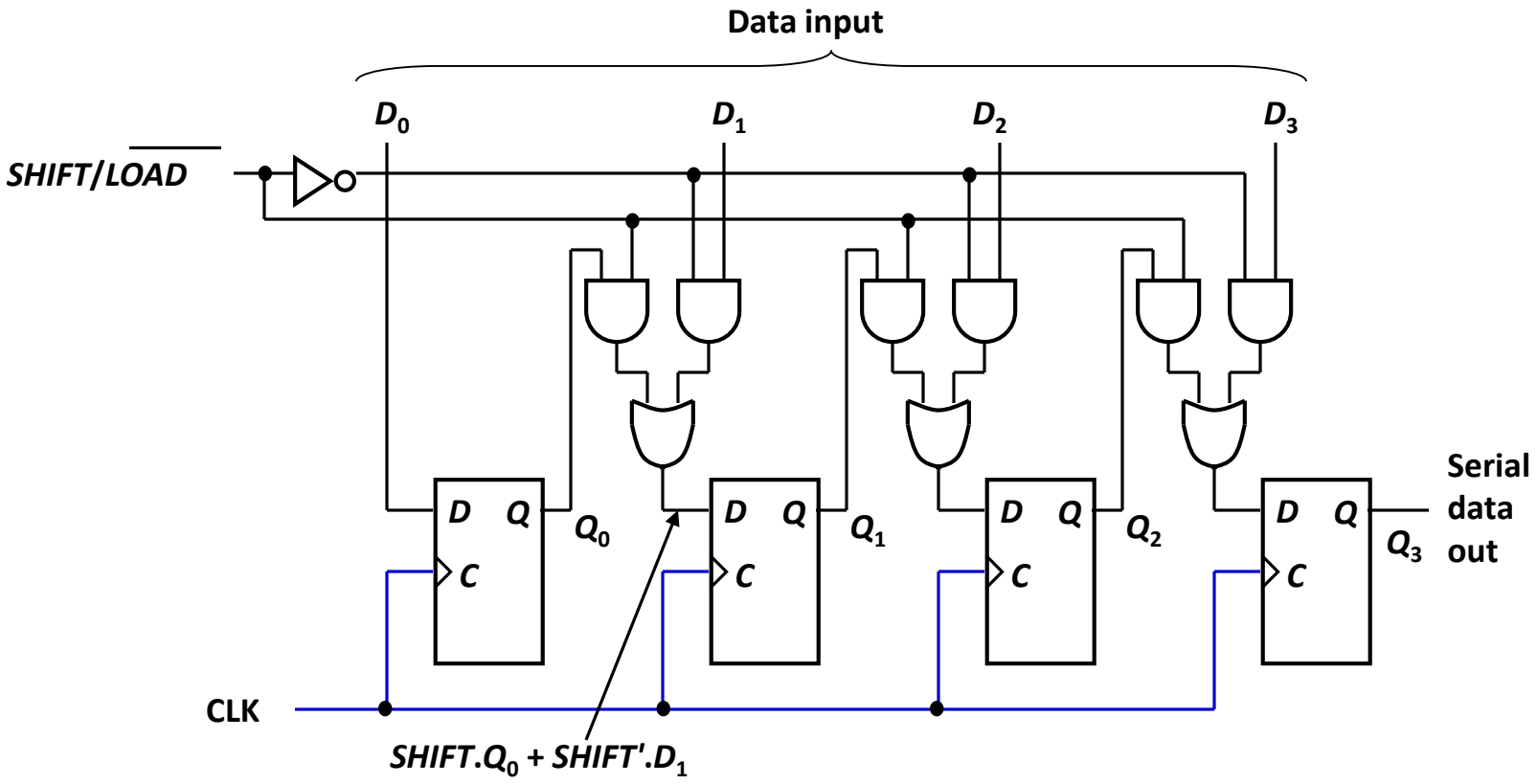




Logic symbol

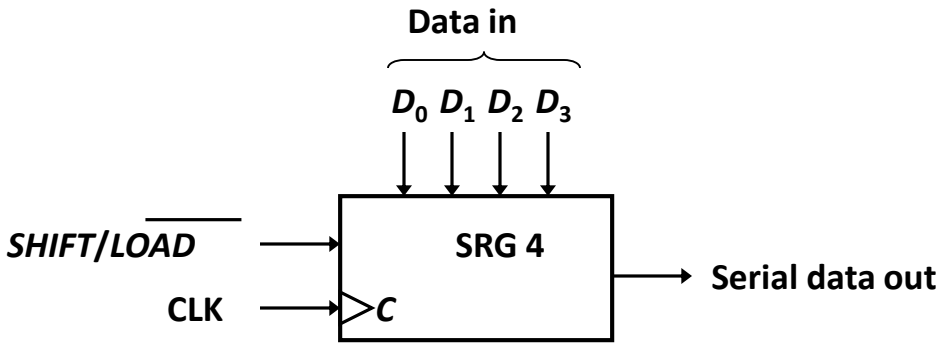
Parallel In/Serial Out Shift Registers

- Bits are entered simultaneously, but output is serial.



Parallel In/Serial Out Shift Registers

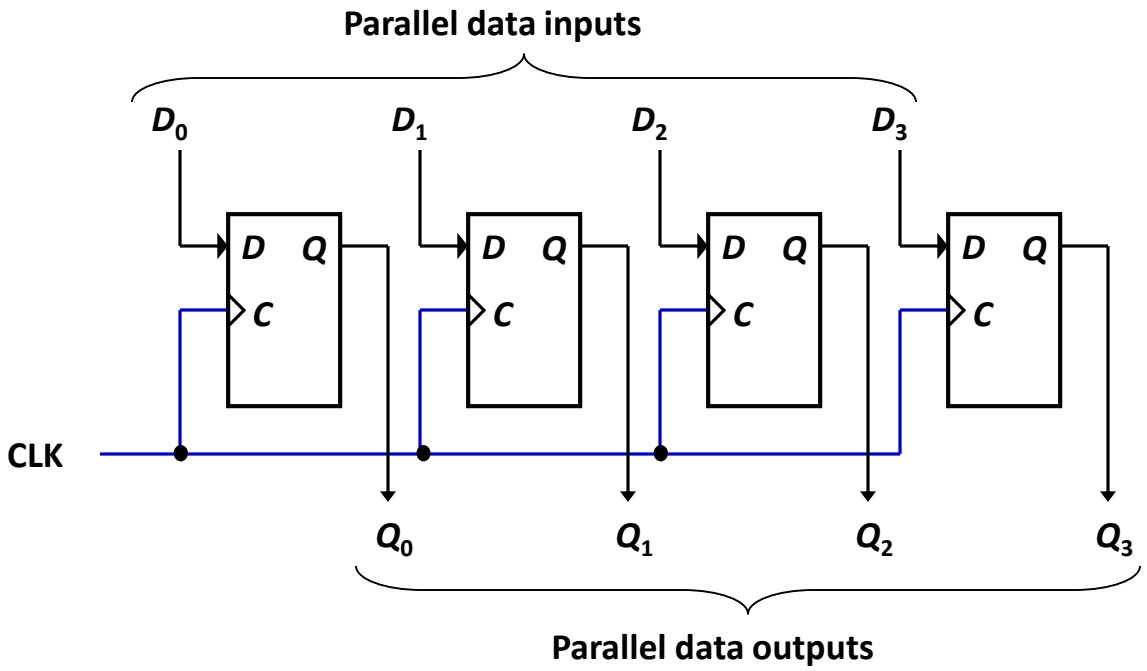
- Bits are entered simultaneously, but output is serial.



Logic symbol

Parallel In/Parallel Out Shift Registers

- Simultaneous input and output of all data bits.



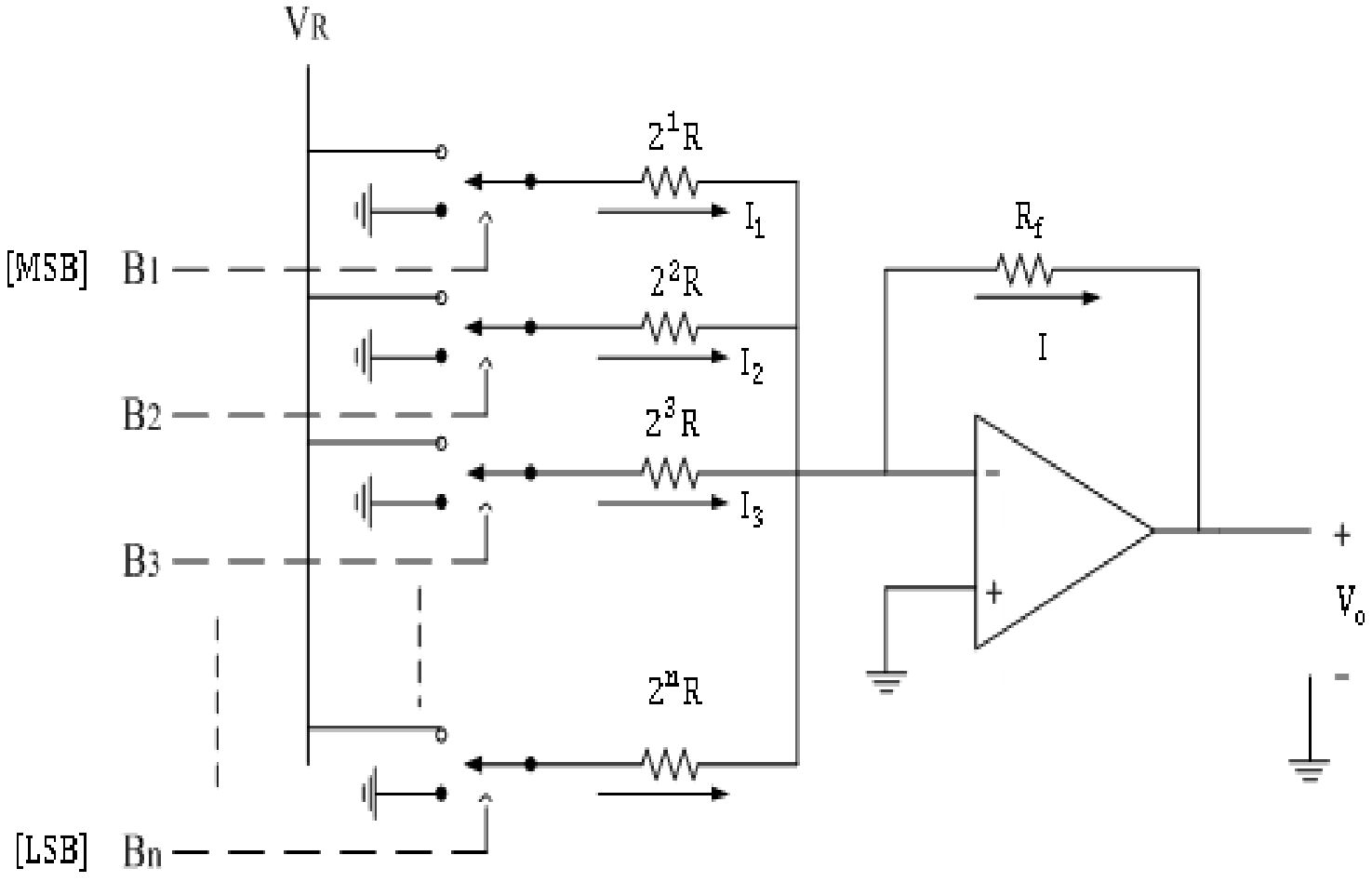
UNIT 5

A/D and D/A Converters

Digital to Analog Converter

In electronics, a digital to analog converter is a system that converts a digital signal into an analog signal. DAC's are commonly used in music players to convert digital data streams into analog audio signals. They are also used in televisions and mobile phones to convert digital video data into analog video signals.

Weighted resistor type DAC



Weighted resistor type DAC

Weighted means each bit is having weight in the sequence

8 4 2 1

MSB LSB

The op-amp is used to produce a weighted sum of the digital inputs , where the weights are proportional to the weights of the bit positions of inputs.

The MSB resistance is 1/8 of the LSB resistance .

The circuit is acting as summing amplifier.

For first input , i.e. , D0 , output will be

= [-Rf / 8R]*D0 equation 1

For second input , i.e. , D1 , output will be

= [-Rf / 4R]*D1 equation 2

For third input , i.e. , D2 , output will be

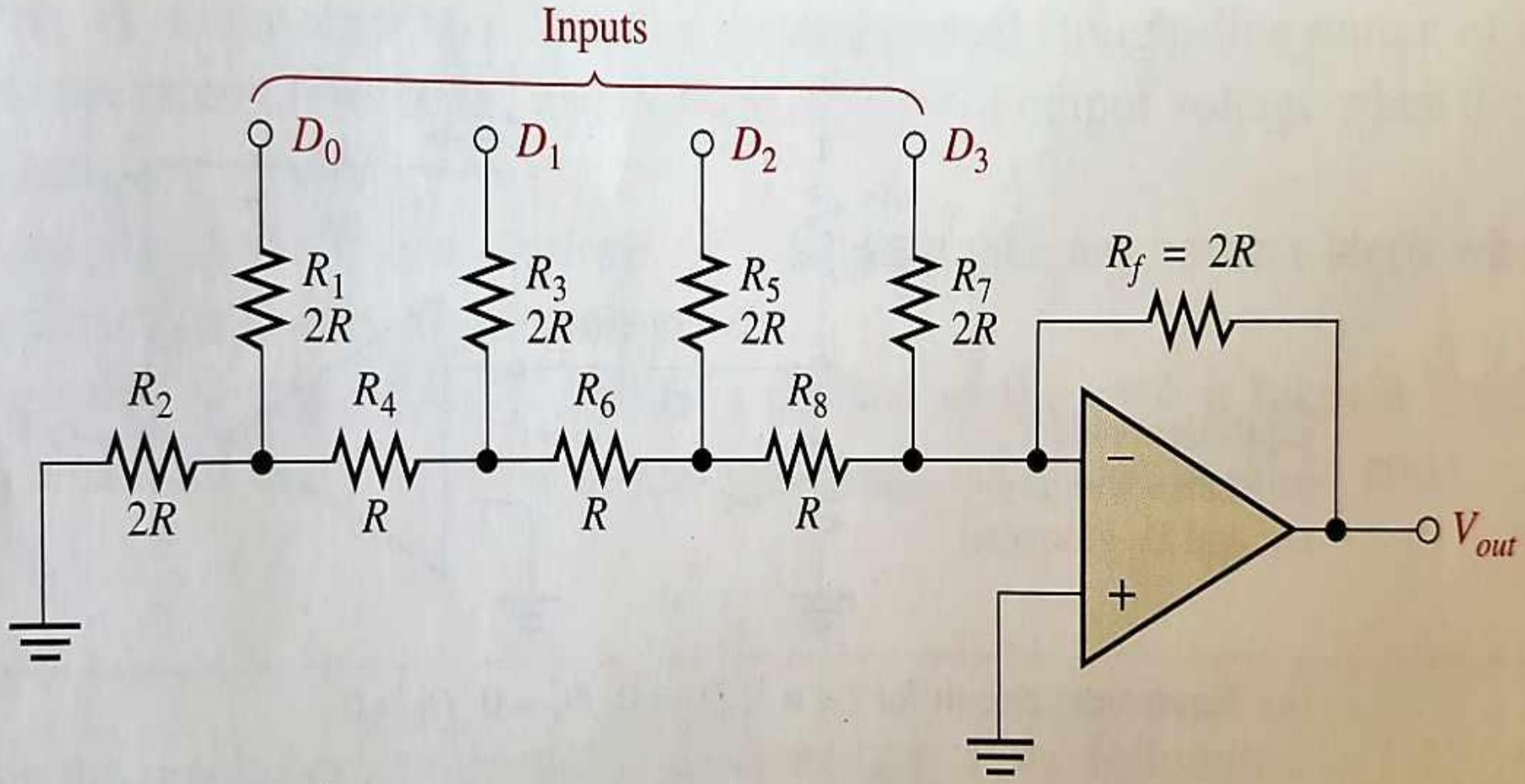
= [-Rf / 2R]*D2 equation 3

For MSB , R , output will be

= [-Rf / R]*D3 equation 4

Adding equation 1 to equation 4

Vout = -Rf / R [D0/8 + DI/4 + D2/2 + D3]


Inputs
D_0
D_1
D_2
D_3
R_1 2R
R_3 2R
R_5 2R
R_7 2R
R_f = 2R
R_2 2R
R_4 R
R_6 R
R_8 R
−
+
V_{out}

R-2R Ladder DAC

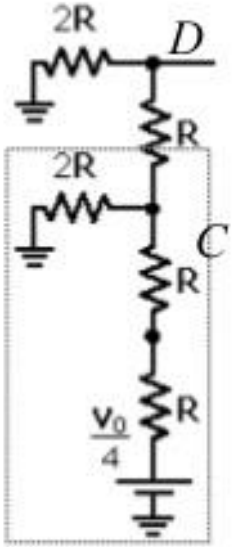
- **Similarly:**

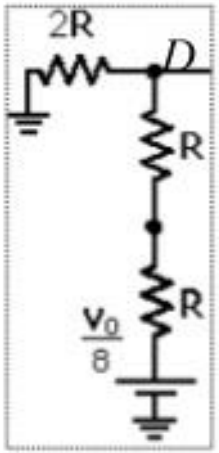
$$E_{th} = V_B = \frac{2R \times \frac{V_0}{2}}{2R + 2R} = \frac{V_0}{4}$$

$$R_{th} = R$$

$$E_{th} = V_C = \frac{2R \times \frac{V_0}{4}}{2R + 2R} = \frac{V_0}{8}$$

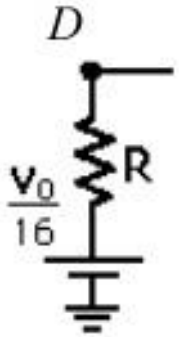
$$R_{th} = R$$

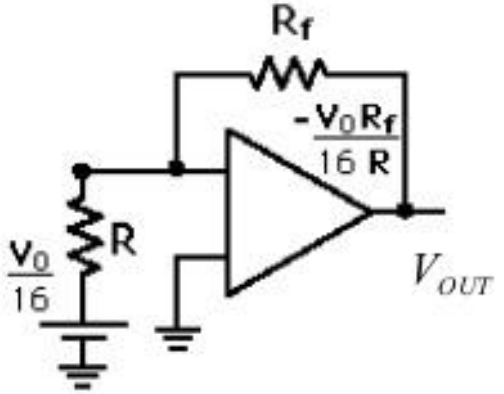




$$E_{th} = V_B = \frac{2R \times \frac{V_0}{2}}{2R + 2R} = \frac{V_0}{4}$$

$$R_{th} = R$$





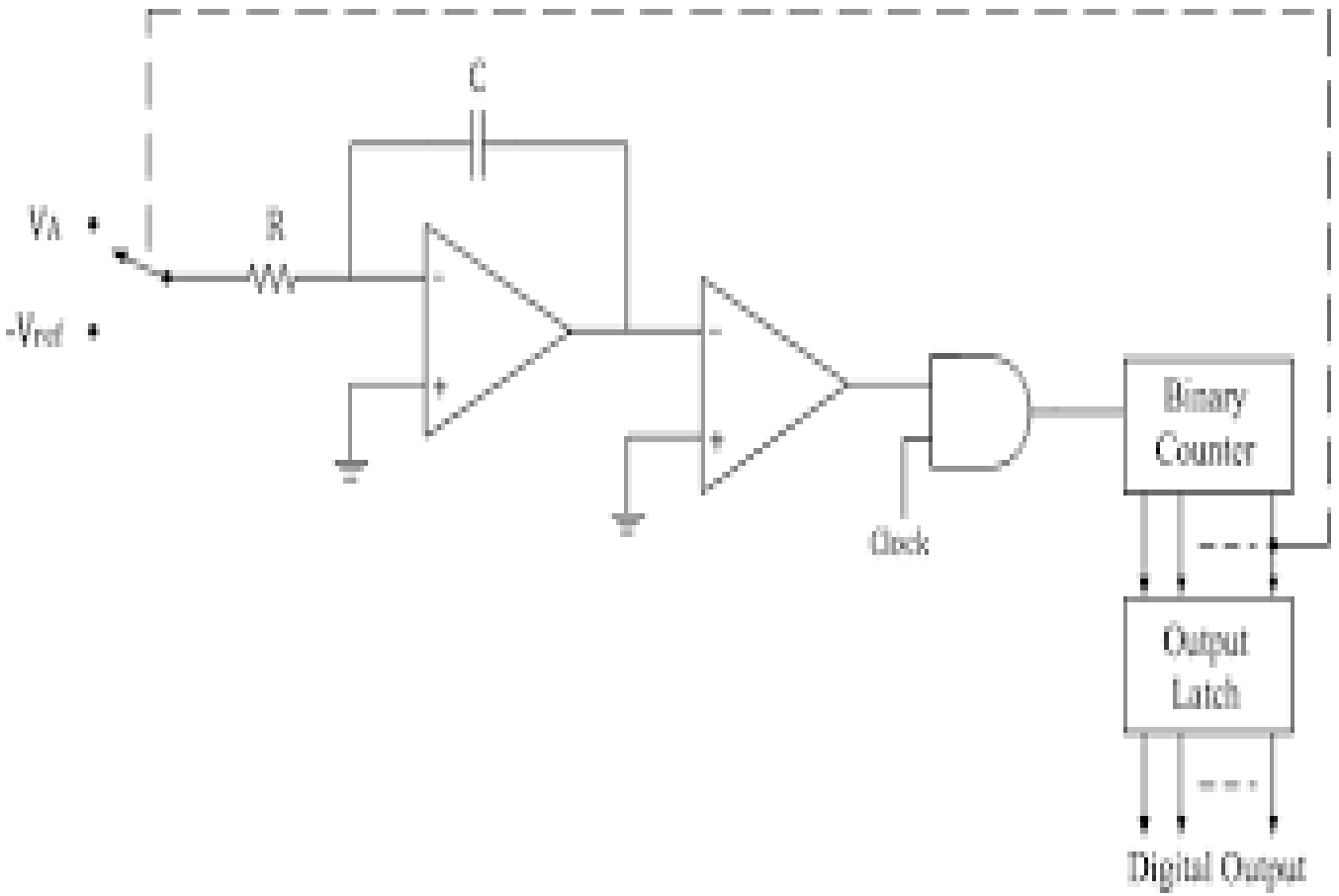
$$Total V_{OUT} = \frac{R_f}{R} \bullet V_{ref} \left[\frac{D_0}{16} + \frac{D_1}{8} + \frac{D_2}{4} + \frac{D_3}{2} \right]$$

D is the binary input

Analog to Digital Converter

In electronics, an analog to digital converter is a system that converts an analog signal, such as sound picked by a microphone or light entering a digital camera, into a digital signal. An ADC is also an electronic device that converts an analog voltage or current to a digital number representing the magnitude of the voltage or current.

Dual slope ADC



Dual slope ADC

Principle

The principle is to first integrate the analog input signal Vin for a fixed duration of 2n clock periods Then it integrates an internal input reference voltage Vr of opposite polarity until the integrator

output is zero . The number n of clock cycle required to return the integrator to zero is proportional to the value of Vin averaged over integration period . Hence n is the desired output

code .

Dual slope ADC

Working

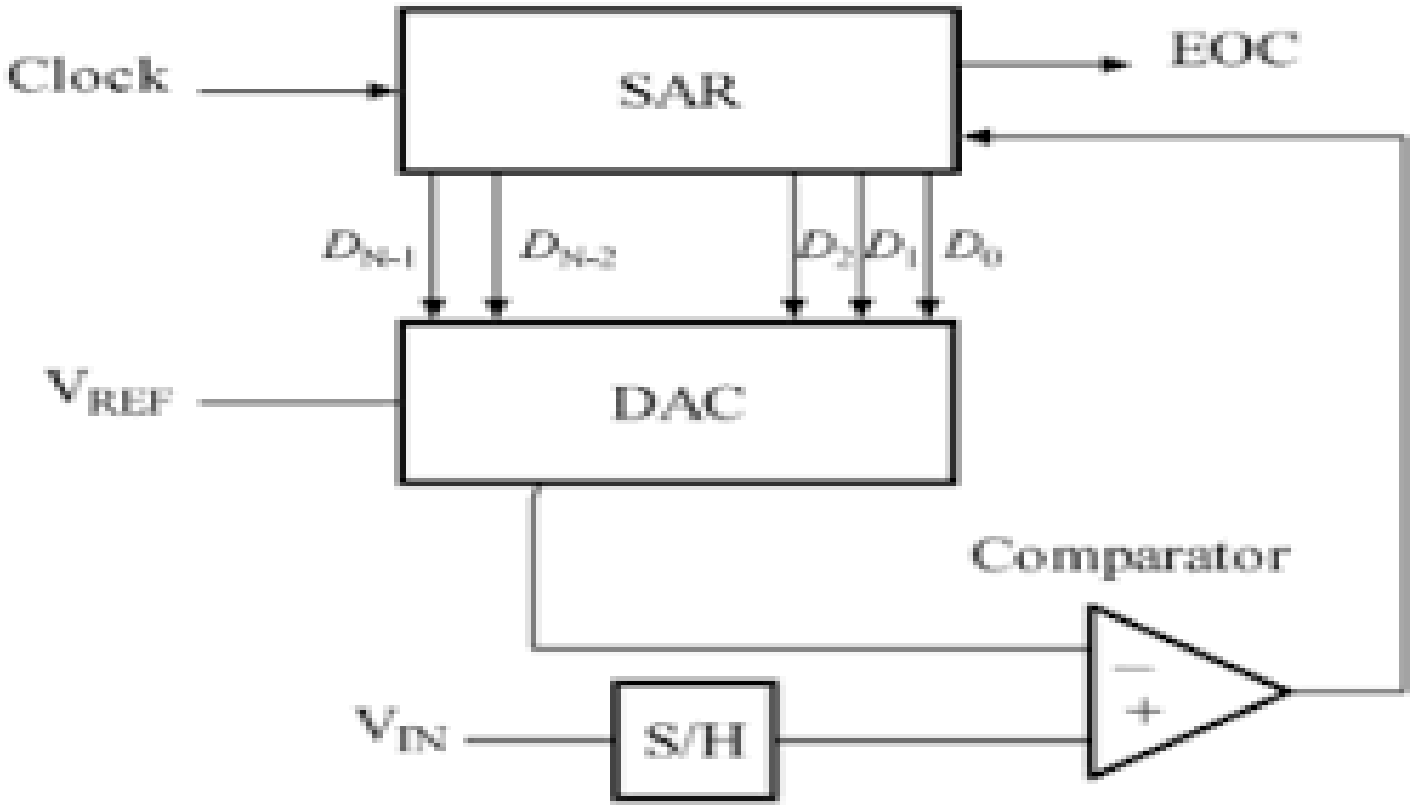
Assume counter is reset and output of integrator is zero.

Connect the switch to Vin . Assume that input is negative and is constant for a period of time, so output of integrator is a positive ramp . The ramp is allowed to continue for a fixed time and voltage it reaches in that time is directly dependent on analog input .

Then counter is reset and switch is connected to a Eref. having positive ramp.

Then AND gate is enabled and counter starts counting . When ramp reaches 0V , comparator output becomes LOW and counter stops counting .

Successive Approximation Method of ADC



Successive Approximation Method of ADC

Principle

The bits of DAC are enabled one at a time , starting with MSB . As each bit is enabled , the comparator produces an output that indicates whether the analog input voltage greater or less than output DAC (Vax) . If D/A output is high , causing the bit in control register to reset and if D/A output is low , the bit is retained in control register . The system enables the MSB first , then the next significant bit and so on .

Successive Approximation Method of ADC

Working

Suppose unknown input voltage Va = 10

On first clock pulse , output register is loaded with 1000 , which is converted by DAC to 8V . So Vdac<Va . So control logic retains that bit , so output is 1000 .

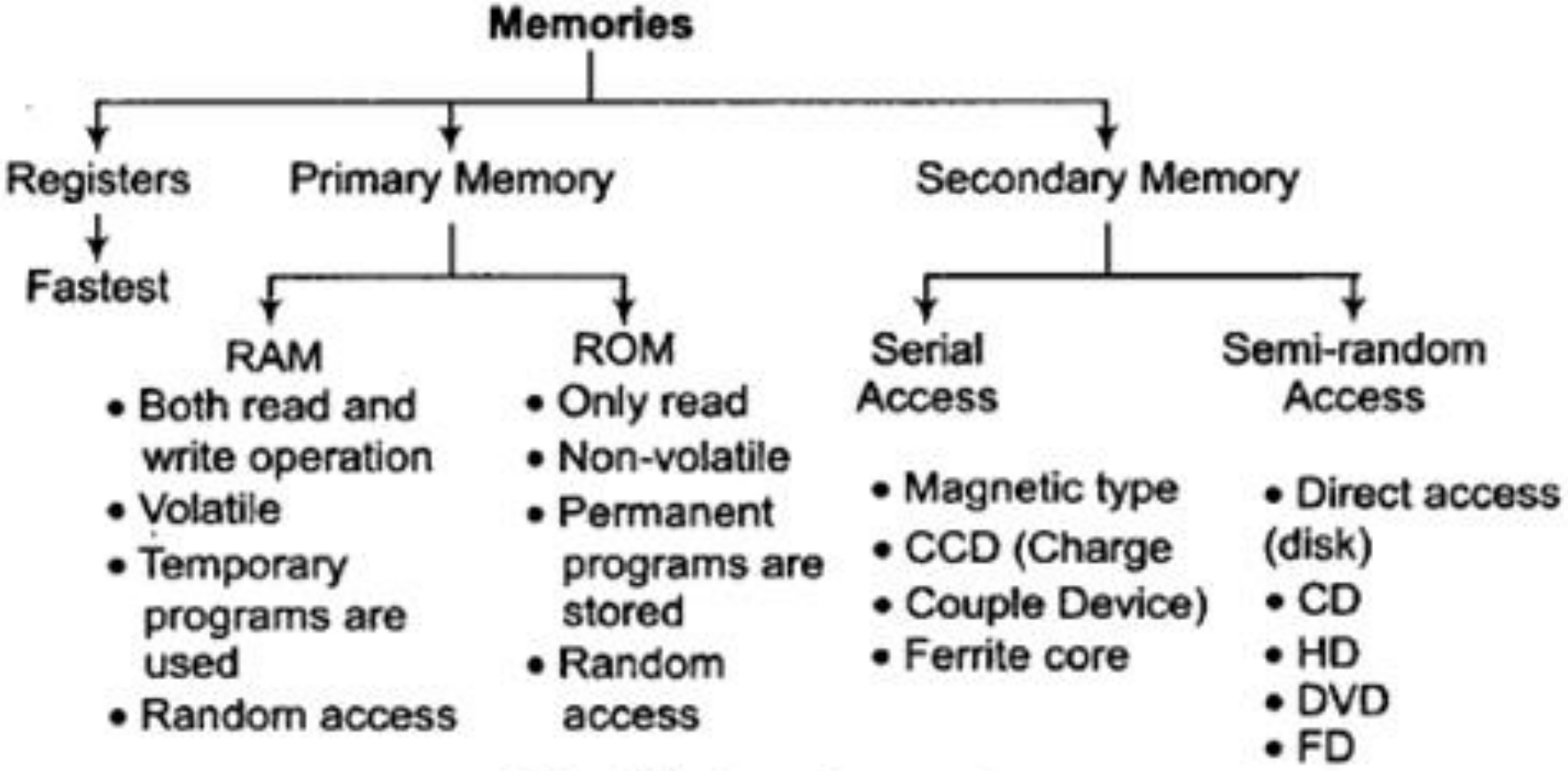
On second clock pulse , output register is loaded with 1100 . So Vdac = 12V , which is greater than Va

So control logic clears that bit , so output = 1000 .

On third clock pulse , output register is loaded with 1010 . So Vdac = 10V which is greater than Va, control logic retains that bit . So output = 1010.

The control logic is loaded with 1011 , Vdac = 11V , which is greater than Va . So output = 1010 , which is nearest integer value to the input 10.3V .

At this point , conversion is complete and EOC is activated by control logic .



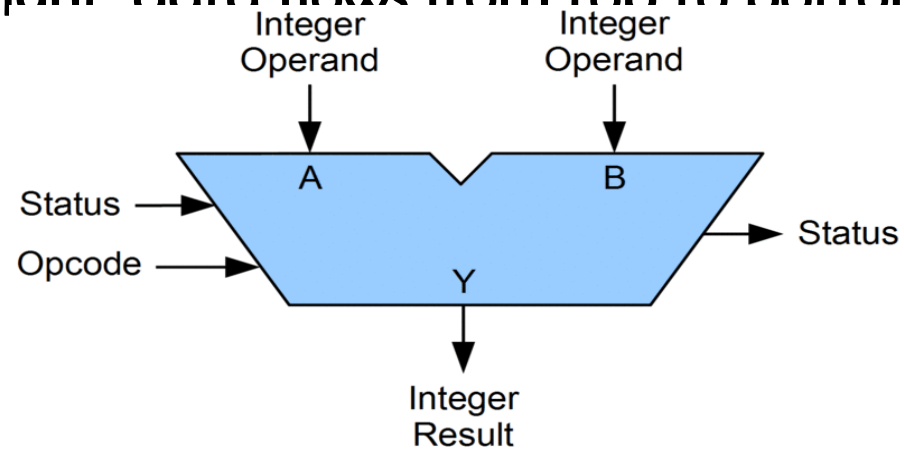
Classification of memories

Arithmetic logic unit (ALU)

An arithmetic logic unit(ALU)is a combinational digital electronic circuit that performs arithmetic and bitwise operations on integer binary

Symbolic Represantation of ALU

- A symbolic representation of an ALU and its input and output signals , indicated by arrows pointing into or out of the ALU, respectively .Each arrow represents one or more signals . Control signals enter from the left and status signals exit on the right; data flows from top to bottom .

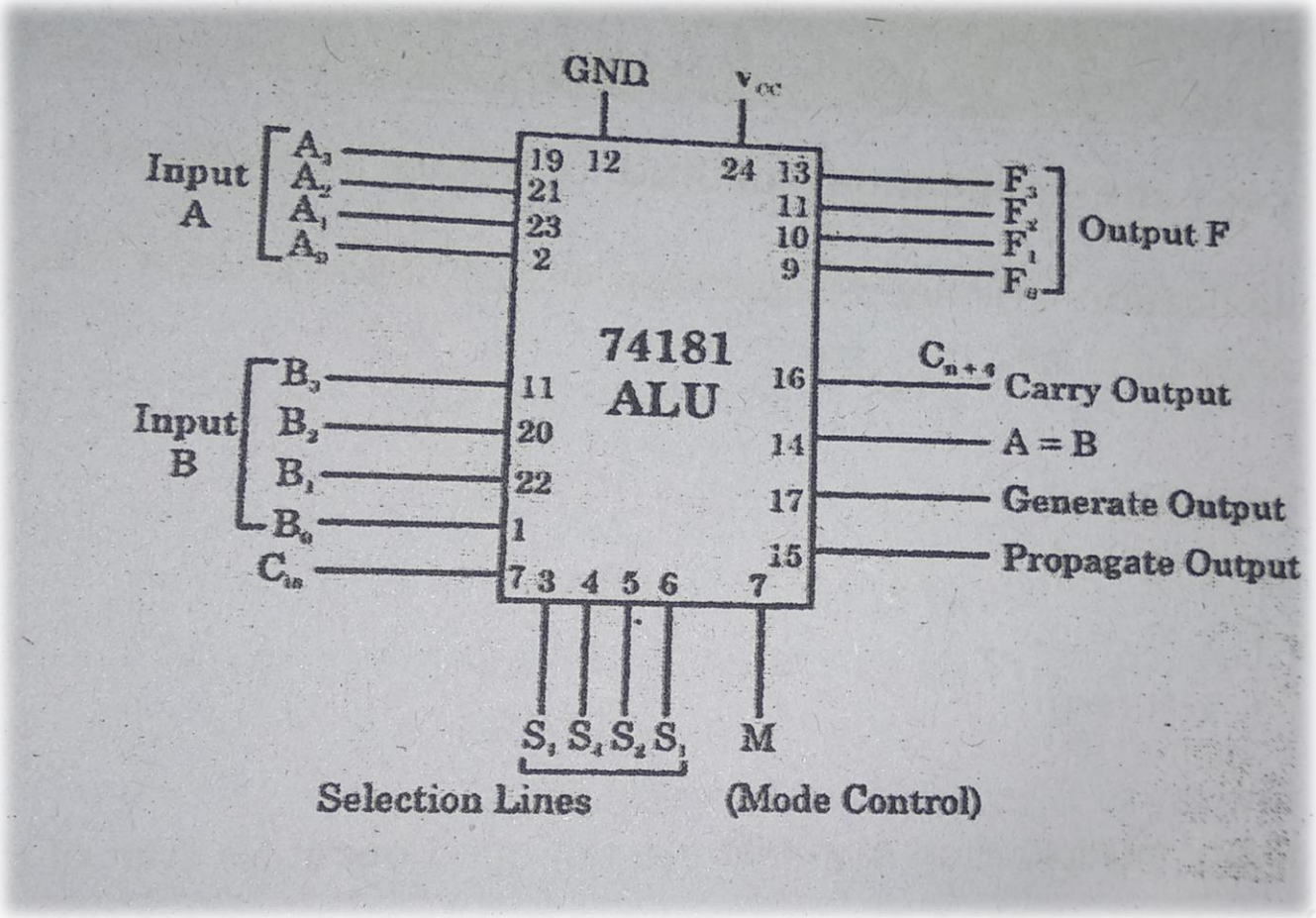


DATA

- A basic ALU has thee parallel data buses consiting of two inputs operands(A and B) and result output (Y).Each data bus is a group of signals that conveys one binary integer number. Typically, the A,B and Y bus widths(the number of signals comprising each bus) ae identical nd match the nativve word size of the external circuitry.

The function of various pins is also shown.Supply is given at Pin.24 and Pin 12 is grounded. Pin 14 performs the function of comparator . Pin7 perform the mode control function. The two inputs A and B are compared by Pin14. Carry output is generated at pin16 and the output is generated at pin17. output is propagated from pin15.IC is capable of performing 16 different operations depending upon the select lines S0,S1,S2 and S3 at pin no 3 to 6

Pin diagram of 74181 ALU



Applications

- The ALU is responsible for complex mathematical calculations,such as floating point math . It does not have any specific additional use as related to databases,except to say that the database ,as any other software ,uses the ALU to perform sums, averages, and so on duriinng queries.