Dijlah University College

Computer Science Department

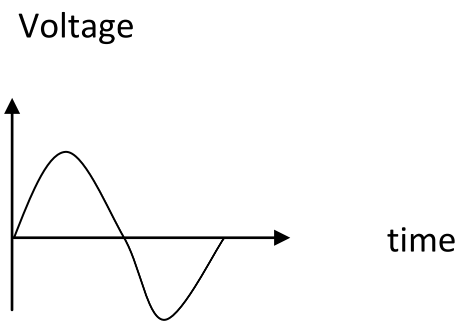
LOGIC   CIRCUIT DESIGN

Dr. Hamza Al-Obaidi

Logic Design

Analog vs. Digital Systems

- Analog

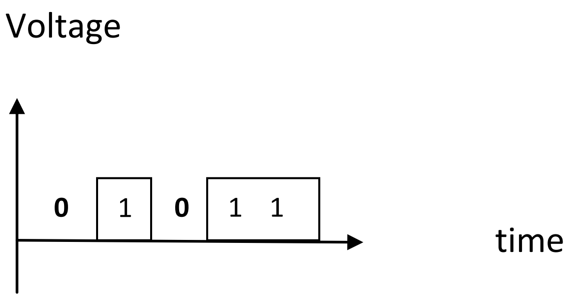


-Continuous time varying voltages ocelot currents

-Basic elements of analog circuits:

. Resistors
. Capacitors
. Inductors
. Transistors

- Digital



- Discrete signals sample in time
- Two possible values

  OV, low, false (logic 0)

  5v, high, true (logic1)
- Basic elements of digital circuits

  Logic gates : AND , OR, NOT

Advantage of Digital System

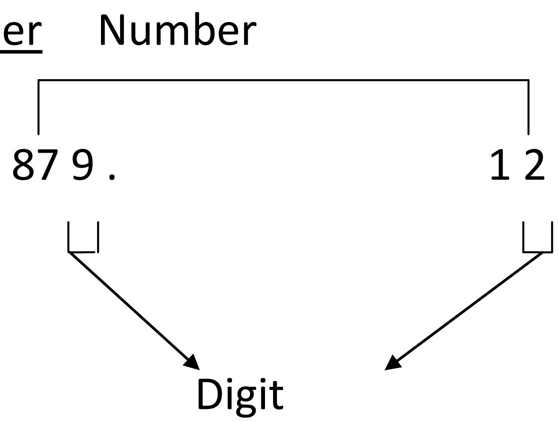
- Reproducible results
- Relative ease of design
- Flexibility and functionality
- High speed
- Small size
- Low cost
- Low power
- Steadily advancing technology
- Programmable logic devices

Number systems and codes

1. Decimal Number System.
2. Binary Number System.
3. Octal Number System.
4. Hexadecimal Numbering System.

Binary Number system is widely using in digital electronic circuits and computer.

Digit and Number



I. Decimal Number System

- Base   10
- Positional notation number system.
  (Positional weight)
- Left most digit is “most significant digit.
- Right most digit is “least significant digit.

- $1,234.56 = 1,234.56_{10}$
- $1 \times 1000 + 2 \times 100 + 3 \times 10 + 4 \times 1 + 5 \times 0.1 + 6 \times 0.01$
- $1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^{\circ} + 5 \times 10^{-1} + 6 \times 10^{-2}$

In general for n digits: $C_{n-1} \times 10^{n-1} + \ldots.. + C_1 \times 10^1 + C_0 \times 10^{\circ}$

-where Ci is the coefficient for the i-th power.

- and Ci is a digit 0 through 9.

Exp.:

1. $128 = 1.10^2 + 2.10^1 + 8.10^{\circ}$

   $100 + 20 + 8 = (128)_{10}$

2. 6239 .45 ⟹

   $= 6.10^3 + 2.10^2 + 3.10^1 + 9.10^{\circ} + 4.10^{-1} + 5.10^{-2}$

   $= 6000 + 200 + 30 + 9 + 0.4 + 0.05$

   $= (6239.45)_{10}$

II.Binary number System

- Base 2

- Only 2 digits = 0 and 1 (exp : 101011 = 6 bits).

  A binary digit is called a bit.

- Positional notation number system .

- Left most bit is “most significant bit” or MSB.

- Right most bit is “last significant bit” or LSB.

In general, for n digits:

$C_{n-1} \times 2^{n-}1 + \ldots\ldots.. + C_1 \times 2^1 + C_0 \times 2^0$

- Where C_i is the coefficient for the i-th power.
- And $\mathbf{C}_i$ is a digit 0 or 1.

Exp: $101010_2 \quad =$

$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 =$

$1 \times 32 + 0 \times 16 + 1 \times 8 + 0 \times 4 + 1 \times 2 + 0 \times 1 = 42_{10}$

Exp: $101.01_2 \quad =$

$101.01_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$

$1 \times 4 + 0 \times 2 + 1 \times 1 + 0 \times ½ + 1 \times ¼ = 5.25_{10}$

- Normally ignore leading Os unless ≠ of bits is specified

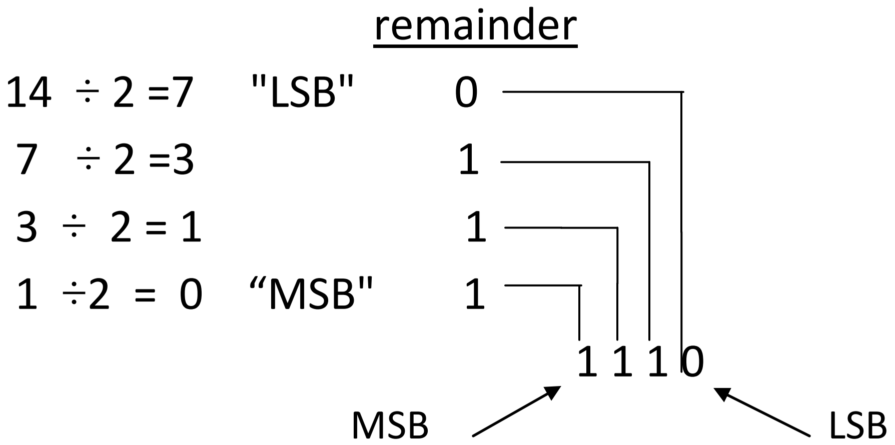
- $00000101_2 = 101_2$
- But if we are dealing with 8 – bit values (for example) we keep all 8 bits.
- Storage unite " Byte " has 8 bits

1 byte = 8 bits.

Decimal - to – Binary Conversion

Two ways:

1. Sum of weights method.
2. Repeated Division - by- 2 method. → Simple and more



remainder

14 ÷ 2 =7 "LSB" 0

7 ÷ 2 =3 1

3 ÷ 2 = 1 1

1 ÷2 = 0 “MSB" 1

1 1 1 0

MSB

LSB

So$(14)_{10} = (1110)_2$

Exp :Conversion of decimal to binary $(25)_{10}$.

25 ÷ 2 = 12 1 (LSB)

12 ÷ 2 = 6 0

6 ÷ 2 = 3 0

3 ÷ 2 = 1 1

1 ÷ 2 = 0 1 (MSB)

$(25)_{10} = (11001)_2$

Conversion of decimal fractions

$(0.3125)_2$	Carried digits		
$0.3125 \times 2 = 0.625$	0	→	(MSB)
$0.625 \times 2 = 1.25$	1		
$0.25 \times 2 = 0.5$	0		
$0.5 \times 2 = 1.0$	1	→	(LSB)

$(0.3125)_2 = (0101)_2$

Binary - to – Decimal Conversion

Exp. :

$(1101001)_2 \longrightarrow (\qquad)_{10}$

2^6	2^5	2^4	2^3	2^2	2^1	2^0	weight
1	1	0	1	0	0	1	bits

$= 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= 64 + 32 + 8 + 1 = (105)_{10}$

Exp. :

$(0.1011)_2 = (\qquad\qquad\qquad)_{10}$

$$\begin{array}{lllll} .\,2^{-1} & & 2^{-2} & 2^{-3} & 2^{-4} \\ 0\,.\,1 & & 0 & 1 & 1 \end{array}$$

$(0.1011)_2 = 1\times2^{-1} + 1\times2^{-3} + 1 \times 2^{-4} = 0.5 + 0.125 + 0.0625$

$= (0.6875)_{10}$

Binary Arithmatic

Binary addition:

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 0$ carry $1 \Longrightarrow 1\,C$

Exp :

$6 + 3 = 9$

$$\begin{array}{rcccc} & 1 & & & \\ & & 1 & 1 & 0 \\ + & & 0 & 1 & 1 \\ \hline & 1 & 0 & 0 & 1 \end{array}$$

Exp :

4 + 3 = 7

$$\begin{array}{rccc} & 1 & 0 & 0 \\ + & 0 & 1 & 1 \\ \hline & 1 & 1 & 1 \end{array}$$

Binary Subtraction

1 – 0 = 1

1 – 1 = 0

0 – 1 = 1 borrow 1

Exp. : 1 0 1 – 0 1 1 ⟹

$$\begin{array}{rccc} & 1 & \overset{\text{borrow 1}}{0} & 1 \\ \_ & 0 & 1 & 1 \\ \hline & 0 & 1 & 0 \end{array}$$

One's and Two's complements of binary

Important to represent negative number.

1. 1's complement ⟹ invert the binary represent for number.

1s become 0s

And 0s become 1s

Exp: $6_{10} = 0\ \ 1\ \ 1\ \ 0_2$

$\_\,6_{10} = 1\ \ 0\ \ 0\ \ 1_2$

2. 2's complement ⟹ take 1's complement and add 1.

Exp: $6_{10} = 0\ \ 1\ \ 1\ \ 0$

$$\Downarrow$$

$1\ \ 0\ \ 0\ \ 1 \longrightarrow$ 1s comp.

$$\underline{+\ \ \ 1}$$

_ 6_{10} = $1\ \ 0\ \ 1\ \ 0 \longrightarrow$ 2s comp.

Alternative method :- right to left copy bits through 1st logic 1, then invert:

Exp.:

$6_{10} = 0\ \ 1\ \ 1\ \ 0$

$$\Downarrow$$

$1\ \ 0\ \ 1\ \ 0 \Longrightarrow$ 2's comp.

Note:

MSB in 1's and 2's complementis sign value.

(0 = + , 1 = - 1)

Representation of signed numbers

Signed magnitude

$6_{10} = \boxed{1}\ 1\ 1\ 0_2$

$\_6_1\ 0 = \boxed{1}\ 1\ 1\ 0_2$

Sign bit

Negative number representations

1. Sign – magnitude
2. 1’s complement
3. 2’scomplement

Addition and subtraction with 2,s comp.

Most often used in hardware implementations of subtractions

$+3_{10} = 0\ \ 0\ \ 1\ \ 1_2$

$+\ +4_{10} = 0\ \ 1\ \ 0\ \ 0_2$

$+7 = 0\ \ 1\ \ 1\ \ 1_2$

$-\ 3_{10} = 1\ \ 1\ \ 0\ \ 1_2$

$+\ -\ 4_{10} = 1\ \ 1\ \ 0\ \ 0_2$

$\_\ 7_{10} = 1\ \ 0\ \ 0\ \ 1$

1001

invert 0110

add 1 0111= 7

$-\ 3_{10} = 1\ \ 1\ \ 0\ \ 1_2$

$+\ +\ 4_{10} = 0\ \ 1\ \ 0\ \ 0_2$

$+\ 1 = 0\ \ 0\ \ 0\ \ 1_2$

$+\ 3_{10} = 0\ \ 0\ \ 1\ \ 1_2$

$+\ \_4_{10} = 1\ \ 1\ \ 0\ \ 0_2$

$\_\ 1_{10} = 1\ \ 1\ \ 1\ \ 1_2$

1111

invert0000

add 10001 = 1

2's complement operation

to show positive result.

The octal Numbering system.

- Other useful number system in digital design.
- Base 8 (octal).

  8 digits: 0 through 7

  $5310_8 = 5 \times 8^3 + 3\times 8^2 + 1\times 8^1 + 0 \times 8^0$

  $5 \times 512 + 3 \times 64 + 1\times 8 + 0 \times 1 = 2760_{10}$

Octal - to – decimal conversion

$(2275)_8$

Number	2	2	7	5
Weights	8^3	8^2	8^1	8^0
	512	64	8	1

Sol.:$(2275)_8 = 2 \times 8^3 + 2 \times 8^2 + 7 \times 8^1 + 5 \times 8^0$

$1024 + 128 + 56 + 5 = (1213)_{10}$

Decimal - to – octal conversion

$35_{10} = (43)_8$

	Remainder	
$35 \div 8 = 4$	3 →	LSB
$4 \div 8 = 0$	4 →	MSB
	43_8	

Exp.:

Convert decimal No. $(150)_{10}$ to octal No.

$(150)_{10} = (226)_8$

	Remainder		
$150 \div 8 = 18$	6	→	LSB
$18 \div 8 = 2$	2		
$2 \div 8 = 0$	2	→	MSB

Decimal fraction – to – octal conversion

$(0.265)_{10} \longrightarrow (0.207534)_8$

	Carry		
$0.265 \times 8 = 2.12$	2	→	MSB
$0.12 \times 8 = 0.96$	0		
$0.96 \times 8 = 7.68$	7		
$0.68 \times 8 = 5.44$	5		
$0.44 \times 8 = 3.52$	3		
$0.52 \times 8 = 4.16$	4	→	LSB

$(0.207534)_8$

Exp.:

Convert $(44.5625)_{10}$ to octal N.S..

	Rem.		carry	
$44 \div 8 = 5$	4	$0.5625 \times 8 = 4.5$	4	MSB
$5 \div 8 = 0$	5	$0.5 \times 8 = 4.0$	4	LSB
	$(54)_8$		$(44)_8$	

S0l. $(44.5625)_{10} = (54.44)_8$

Octal to decimal conversion (with fractions).

$(567.14)_8 \longrightarrow (\qquad)_{10}$

So l.: $5 \times 8^2 + 6 \times 8^1 + 7 \times 8^0 + 1 \times 8^{-1} + 4 \times 8^{-2} =$

$5 \times 64 + 6 \times 8 + 7 \times 1 + 0.125 + 0.0625 =$

$320 + 48 + 7 + 0.125 + 0.0625 = (375.1875)_{10}$

Octal - to – binary conversion

We can represent each octal digital contains (3) bits by binary number.

Octal No.:	0	1	2	3	4	5	6	7
Binary No.:	000	001	010	011	100	101	110	111

$8 = 2^3$ ⟶ 1 octal digital = 3 bits

Croups of 3 bits for octal digital.

Exp: 1. Convert $(357)_8$ to binary No.:

$(357)_8 =$ 3 5 7

011 101 111

$(011101111)_2$

2. $(1276.543)_8$ =

1 2 7 6. 5 4 3

001 010 111 110. 101 1000 011

$(\underline{0}01\ 010111\ 110.\ \ 101\ \ 1000\ \ 011)_2$

↓

×

$(1010\ \ 111\ \ 110.\ \ 101\ \ 1000\ \ 011)_2$

Binary to octal conversion

We have $(1011001011100.00101)_2$

We start from the point to the left and to the right taking ach 3 bits to gather.

001	011	001	011	100.	001	010
1	3	1	3	4	1	2

$= (13134.12)_8$

Arithmetic operations in octal system

Octal addition

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

$$\begin{array}{r} 34 \\ +42 \\ \hline 76 \end{array} \qquad \begin{array}{r} 56 \\ 63 \\ \hline 141 \end{array} \longrightarrow \begin{array}{l} \ \ 6 \\ +3 \\ \hline 9 + 2 = 1 \rightarrow \text{ 1 carry} \\ 5+1 \\ +6 \\ \hline 12 + 2 = 4 \rightarrow \text{ 1 carry} \end{array}$$

Subtraction in octal system

$$\begin{array}{r} 6\ 5\ 7 \\ -3\ 4\ 6 \\ \hline 3\ 1\ 1 \end{array}$$

$$\begin{array}{r} 6\ 2\ 1 \\ \cancel{7}\ \cancel{3}\ 2 \\ -6\ 3\ 4 \\ \hline 0\ 7\ 6 \end{array}$$

Hexadecimal number systems

- Base 16 (hexadecimal, or just “hex” is the common slang)
- 16 digits : 0 through 9 and A through F digits A- F represent decimal values 10 – 15 respectively.

Exp.:

$A1C_{16}$ =

$$A \times 16^2 + 1 \times 16^1 + C \times 16^0 =$$

$$10 \times 256 + 1 \times 16 + 12 \times 1 = 2588_{10}$$

Hex. - to - decimal conversion

$(522.39)_{16}$ =

$$5 \times 16^2 + 2 \times 16^1 + 2 \times 16^0 + 3 \times 16^{-1} + 9 \times 16^{-2} =$$

$$(5 \times 256) + (2 \times 16) + (2 \times 1)+ (3 \times 0.0625) + (9 \times 0.0039062) =$$

$$1280 + 32 + 2 + 0.1875 + 0.00351558 =$$

$$(1314.222655)_{10}$$

Decimal – to – hexadecimal conversion

1. $(97)_{10} = (61)_{16}$

	Rem.		
97 ÷ 16 = 6	1	→	LSB
6 ÷ 16 = 0	6	→	MSB
	$(61)_{16}$		

2. $(314)_{10}$ → $(\ldots\ldots\ldots\ldots)_{16}$

	Rem.		
314 ÷ 16 = 19	A	→	LSB
19 ÷ 16 = 1	3		
1 ÷ 16 = 0	1	→	MSB
	$(13A)_{16}$		

3. $(0.78125)_{10}$ → $(\ldots\ldots\ldots\ldots\ldots)_{16}$

	carry		
o.78125 × 16 = 12.5	C	→	MSB
0.5 × 16 = 8.0	8	→	LSB
	$(0.C8)_{16}$		

Hexadecimal-to-binary conversion

Groups of 4 bits for hex digital

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

$(3A5)_{16} = \quad 3 \qquad A \qquad 5$

$$\underbrace{0011 \qquad 1010 \qquad 0101}_{(001110100101)_2}$$

$(B35, D1)_{16} = \quad B \qquad 3 \qquad 5, \quad D \qquad 1$

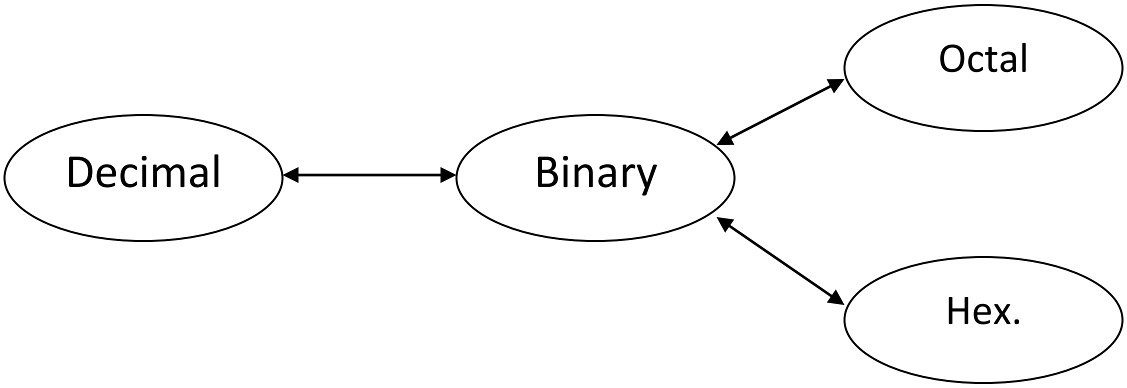
$1011 \quad 0011 \;\; 0101 \quad 1101 \quad 001$

Binary-to-hex conversion

1. $0110 \quad 1110 \quad 1001 \quad = (6E9)_{16}$
2. $\underbrace{0001}_{1} \quad \underbrace{1011}_{B} \quad \underbrace{1101}_{D}\,, \quad \underbrace{1010}_{A} \quad \underbrace{0100}_{Y} \; = (1BD,A4)_{16}$

Octal –to- hexadecimal conversion

Recommended conversion order



First we make conversion to binary, then from binary to hex. :

Exp.:

$(25.342)_8 = (010101.011100010)_2$

$\underline{0001}\quad\underline{0101}\ .\ \underline{0111}\quad\underline{0001}$

1 5 7 1

$(25.342)_8 = (15.71)_{16}$

Hexadecimal – to – octal conversion

Exp:$(AB3E.87D)_{16} = (1010101100111110.100001111101)_2$

$\underline{001}\quad\underline{010}\quad\underline{101}\quad\underline{100}\quad\underline{111}\quad\underline{110}\ ,\ \underline{100}\quad\underline{001}\quad\underline{111}\quad\underline{101}$

1 2 5 4 7 6 4 1 7 5

Sol.= $(AB3E.\ 87D)_{16} = (12547.4175)_8$

Arithmetic operation in hexadecimal system

Hexadecimal addition:

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

1 + 2 = 3 ; 1 + A = B ; 1 + F = 10

D + B = 18 ; 9 + F = 18 ; 5 + B = 10

Exp.:

$(35AB2)_{16} + (1A675)_{16} = (50127)_{16}$

1	1	1		
3	5	A	B	2
+1	A	6	7	5
5	0	1	2	7

Note: more than 16 add 4 and carry 1.

Hexadecimal subtraction

Exp.:

$(F2ABD)_{16} - (EF4CE)_{16} = (35EF)_{16}$

F	2	A	B	D
_ E	F	4	C	E
0	3	5	E	F

Codes

1. 8421 binary coded decimal (or BCD) code.

The 8421 binary coded decimal code is used to represent each decimal character ,0 through 9, with a 4- bit binary code. Decimal number can be easily converted to a binary code by replaying the decimal character by the appropriate 4-bit binary code.

BCD code has ten valid 4-bit binary codes representing the decimal number 0÷ 9 .

There are six invalid 4-bit binary codes that are not part at the BCD code.

To convert from a decimal value to the 8421 BCD.

Decimal	8421BCD	Invalid 8421 BCD codes
0	0000	1010
1	0001	1011
2	0010	1100
3	0011	1101
4	0100	1110
5	0101	1111
6	0110	
7	0111	
8	1000	
9	1001	

Exp.: 1- Decimal to BCD conversion

$(1990)_{10}$ =

$(0001\ \ 1001\ \ 1001\ \ 0000)_{BCD}$

2-$(20.57)_{10}$ =

$(0010\quad 0000.\quad 0101\quad 0111)_{\mathbf{BCD}}$

Exp.:

BCD to decimal conversion

$(1001\quad 0010.\quad 1000\quad 0111)_{\mathbf{BCD}}$**=** $(92.87)_{10}$

9 2. 8 7

2. Excess -3 binary coded Decimal.

decimal	Excess - 3	Invalid excess – 3 code
0	0011	0000
1	0100	0001
2	0101	0010
3	0110	1101
4	0111	1110
5	1000	1111
6	1001	
7	1010	
8	1011	
9	1100	

Exp.: decimal to excess – 3 conversion.

$(20.57)_{10}$ =

$(0101\quad 0011.\quad 1000\quad 1010)_{\mathbf{x-3}}$

3. Gray code.

4. Alpha numeric binary codes.

The two most common alpha numeric codes are the ASCII and EBCDIC codes.

(American standard code for information interchange) 7-bit binary coded represent 2^7 or 128 character. (Table 2-8)

Other binary codes

Other codes sometimes encountered:

- Binary coded decimal (BCD):
  - › Uses binary representation for digits 0-9.
  - › Binary digits grouped in groups of 4 bits.
  - ›53_{10} = 0101 0011$_{BCD}$

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

- Gray code:
  - › as we move through consecutive binary values, only one bit changes.

gray	Dec.
000	0
001	1
011	2
010	3
110	4
111	5
101	6
100	7

- These & other codes have limited uses:

› Error detection in data transmission.

C.E.Stroud"Number Systems &Codes (1/06)

ASCII CODE

American standard code for information interchange – ASCII code.

Table2-8

ASCII	MSBs – $b_6b_5b_4$							
$B_3b_2b_1b_0$	000 – 0	001 – 1	010 – 2	011 – 3	100 – 4	101 – 5	110 – 6	111 - 7
0000 – 0	NUL	DLE	SP	0	@	P	`	p
0001 – 1	SOH	DC1	!	1	A	Q	a	q
0010 – 2	STX	DC2	“	2	B	R	b	r
0011 – 3	ETX	DC3	#	3	C	S	c	s
0100 – 4	EOT	DC4	$	4	D	T	d	t
0101 – 5	ENQ	NAK	%	5	E	U	e	u
0110 – 6	ACK	SYN	&	6	F	V	f	v
0111 – 7	BEL	ETB	‘	7	G	W	g	w
1000 – 8	BS	CAN	(	8	H	X	h	x
1001 – 9	HT	EM	)	9	I	Y	i	y
1010 – A	LF	SUB	*	:	J	Z	j	Z
1011 – B	VT	ESC	+	;	K	[	k	{
1100 – C	FF	FS	,	<	L	\	l	|
1101 – D	CR	GS	-	=	M	]	m	}
1110 – E	SO	RS	.	>	N	^	n	~
111 – F	SI	US	/	?	O	–	o	DEL

C.E.Stroud"Number Systems &Codes (1/06)

ABCDIC (Extend binary coded decimal information code).

8 – bit = 2^8 = 56 characters

Exp:

Encode the following line in ASCII.

ASCII in hex code:

22	**54**	**68**	**65**	**20**	**77**	**65**	**65**	**6B**	**65**	**6E**	**64**	**20**
“	**T**	**h**	**e**	**space**	**w**	**e**	**e**	**k**	**e**	**n**	**d**	**space**

Logic function representation

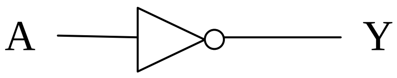
1. Inverter function.

In inverter is a logic gate with one input and one output. The inverter produces an output that is the opposite logic level of the input. The inverter inverters, or complement the input;

LOW → HIGH
HIGH → LOW

Figure shows the logic symbol; equation; and truth table for an inverter. In this figure A is the input and y is the output. The output equation is;

$Y = \bar{A}$, read as "A inverted" , " A bar " or "A NOT ".



$\bar{A}$ The inverter is also known as NOT gate, which performs the NOT function.

Equation $Y = \bar{A}$

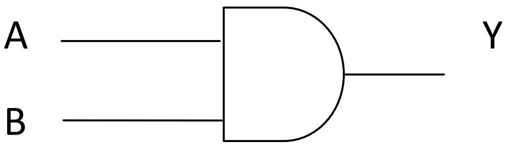
truth table

A	Y
0	1
1	0

One (TTL) transistor – transistor logic integrated circuit (IC),that performs the inverter function is the 7407, which has 6 inverter gates.

2. AND FUNCTION.

An AND gate has two or more input and one output. The only time a logic HIGH is output is when all input are HIGH, if at least one of the inputs is a logic O, the output is a logic 0.



Logic symbol

Truth table

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

Equation

Y = AB = A × B = A.B

AB = BA –commutative

2 inputs = 2^2 output combinations

3 input = 2^3 output combinations

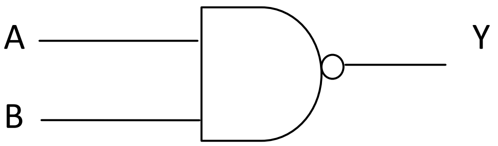
A	B	C	ABC
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

7408 IC s has four two – input AND gates (quadruple) or (quad) two – input AND /c.

3- NAND Function

NAND means “not AND’ and its output is the exact opposite of an AND gate.

A NAND gate has two or more inputs and one output and has opposite function of an AND gate. The NAND function as an AND gate followed by an inverter. The only time a logic o is output is when all inputs are logic 1.



AND gate with inverter bubble on the output

NAND logic symbol

Equation:

$Y = \overline{AB}$

A	B	C	$\overline{ABC}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

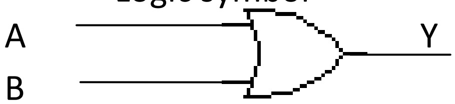
Truth		table
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

TTL ICs containing NAND gates have either one,two,three or four NAND gate per IC. The 7400 has four 2- input NAND gates .

4- OR function

The OR gate is a logic gate with two or more input and one output. The operation of an OR gate is such that of any of the input are a logic 1, the output is a logic 1. The only time an OR has a low or 0 outputs is when all inputs are low.

OR
Logic symbol



Equation :
Y = A + B

“A or B”

The OR function is commutative since the output does not depend on the order of the variables OR ed together.

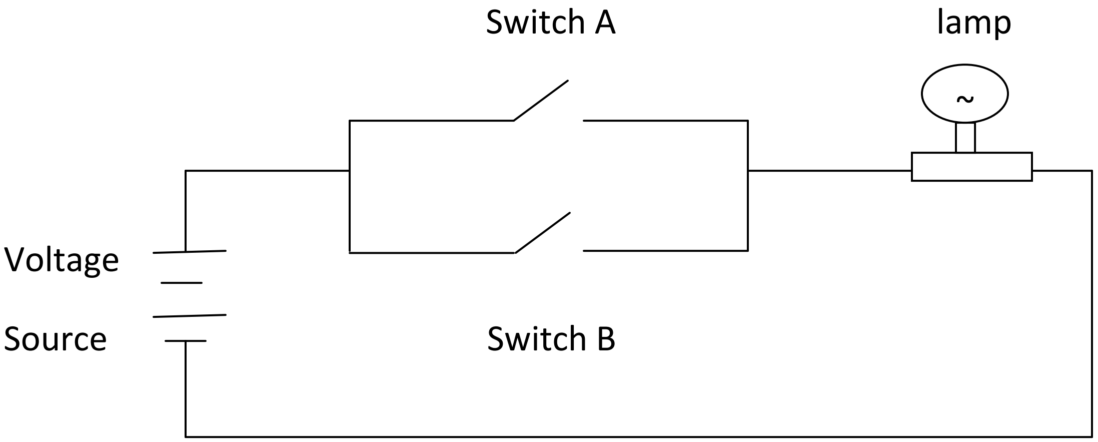
(A + B) = (B +A)

Truth tables

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

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

For more understanding:

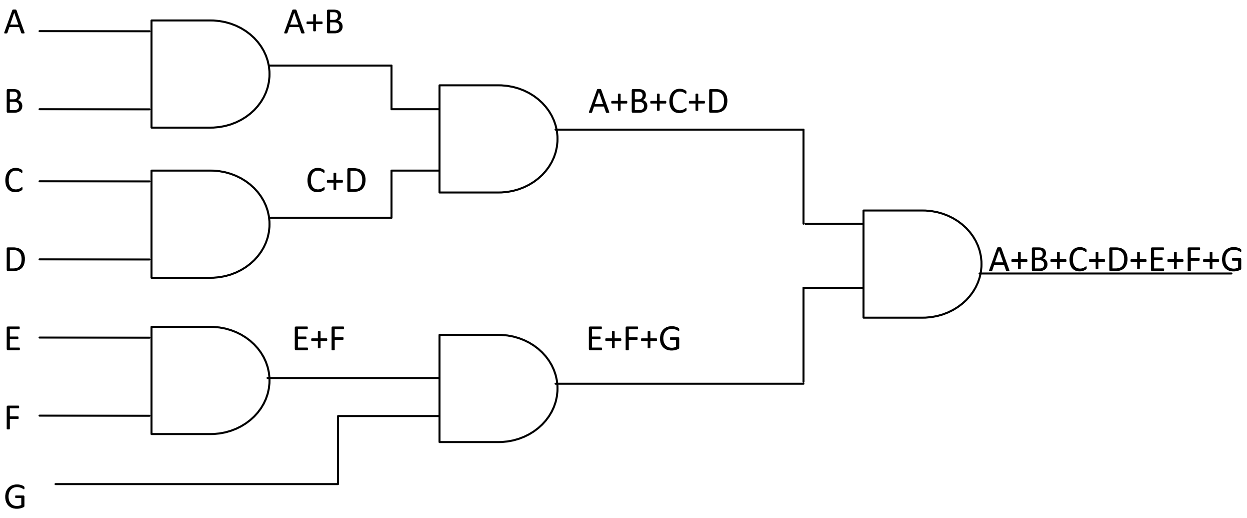


The 7432 IC has four 2- input OR gate.

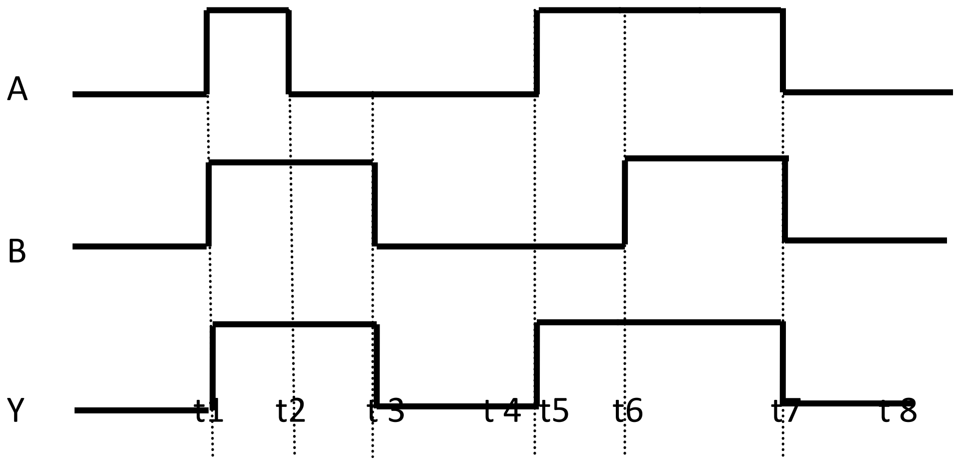
Exp: How many 2- input AND gate will it take to produce the fallowing equation?

X = A + B + C + D + E + F + G

Sol.:



Answer = 6

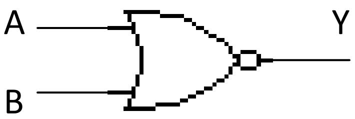


timing diagram

5- NOR FUNCTION.

The NOR function means “NOT OR”. The NOR function as an OR followed by an inverter. The outputs of NOR gate is HIQH only when both inputs are LOW.

NOR logic symbol



Truth table

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

A	B	C	Y = ~~A+B+C~~
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Equation

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

The logic equation for NOR gate has a bar over the entire output. The output of a 3 – input NOR gate will be $\overline{A+ B + C}$, and pronounced “(A or B or C) NOT " or " A NOR B NOR c ”.

6-Exclusive functions: XOR and XNOR.

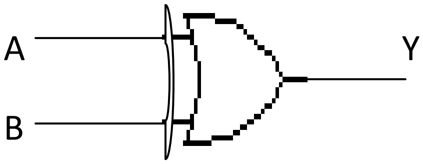
The Exclusive – OR gate is a logic gate with two inputs and one output.

The exclusive – OR is an " inequality " function. The output of XOR is high when the inputs are not equal to each other. If both inputs are HIGH, or it both are LOW , the output at the XOR gate is LOW.

The complement gate to an exclusive – OR gate is the exclusive – NOR, or XNOR. The XNOR is an “equality” function. The only time the output of an exclusive – NOR gate is HIGH is when both inputs are equal, either HIGH or LOW.

XOR

Logic symbol



Equation

$Y = A \oplus B$

“A exclusive –OR B”

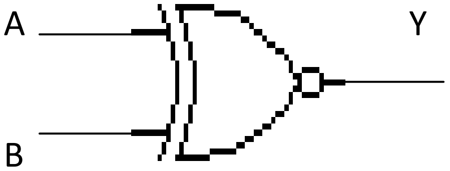
truth table

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

TTL IC 7486 has four 2-input XOR gate. The gate symbol for an XNOR gate is formed by placing the inverter bubble on the output of an XOR gate. As with the XOR, the XNOR function is only a two – input operation.

XNOR

Logic symbol



Truth table

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

Equation

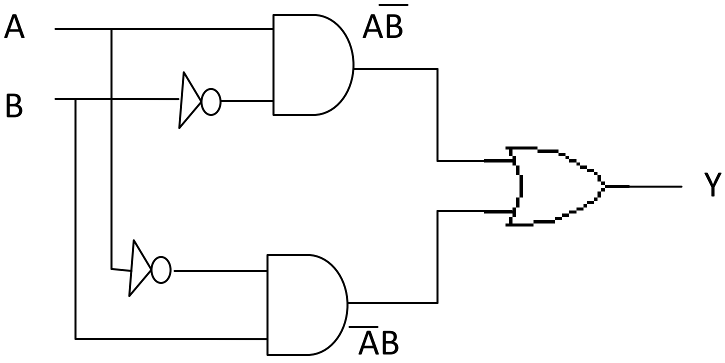
$Y = A \cdot B = \overline{A + B}$

“A exclusive – NOR B”

A 74266 has four 2 – input XNOR gates.

Exp.: $Y = A \oplus B$

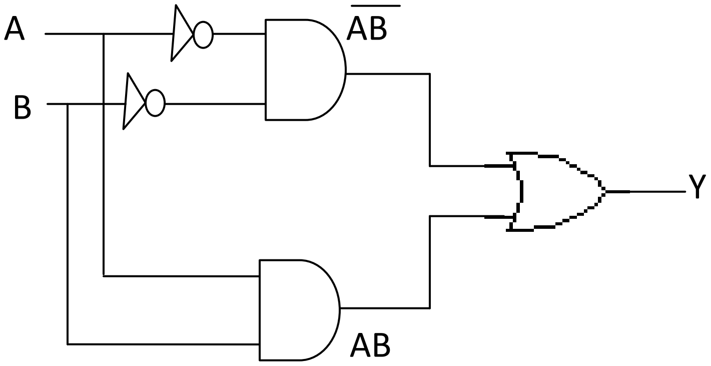
$Y = A\,\overline{B} + \overline{A}\,B$



XOR gates represented by AND, OR, NOT gates.

Exp.: $Y = \overline{A \oplus B}$

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



XNOR is represented by AND, OR, NOT gates.

Exp: draw logic circuit from truth table.

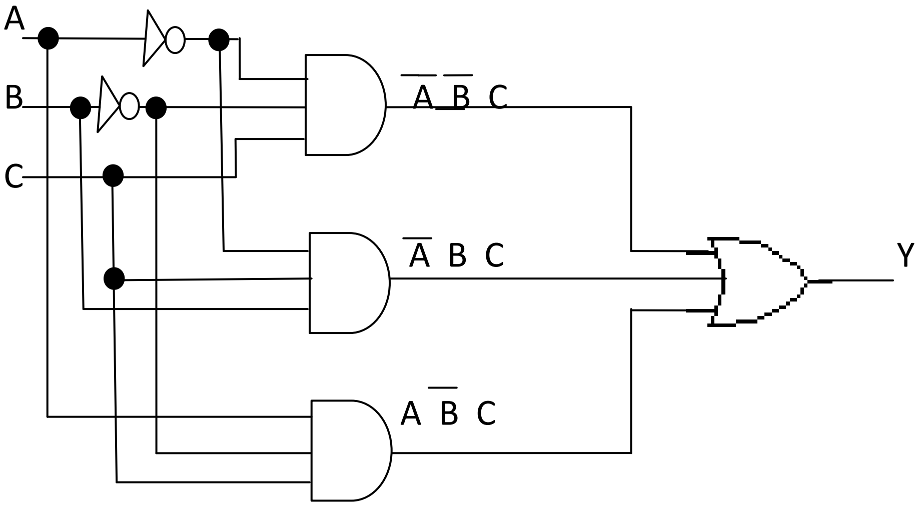
Inputs			outputs
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

$\bar{A}\,\bar{B}\,C = 1$

$\bar{A}\,B\,C = 1$

$A\,\bar{B}\,C = 1$

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



Converting a Boolean expression to a truth table.

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

$\overline{A}\,\overline{B}\,\overline{C} = 0\ 0\ 0$

$\overline{A}\,B\,\overline{C} = 0\ 1\ 0$

$A\,B\,\overline{C} = 1\ 1\ 0$

$A\,B\,C = 1\ 1\ 1$

Output each component is equal

to 1.

input			Output
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Boolean algebra

- Also known switching algebra.
- Invented by mathematician George Boble in 1849.
- Used by Claude Shannon at bell labs in 1938.
  To describe digital circuits built from relays.
- Digital circuits design is based on.
- Boolean algebra : Attribute ينسب الى

  Postulates يفترض

  Theorems نظريات

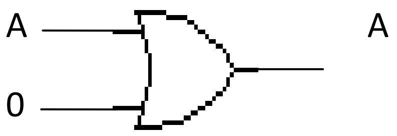
- These allow minimization and manipulation of logic gates for optimizing digital circuits.

Main rules of Boolean algebra

1. $A + 0 = A$	2. $A + 1 = 1$
3- $A \cdot 0 = 0$	4. $A \cdot 1 = A$
5. $A + A = A$	6. $A + \bar{A} = 1$
7. $A \cdot A = A$	8. $A \cdot \bar{A} = 0$
9. $\bar{\bar{A}} = A$	10. $A + A\,B = A$

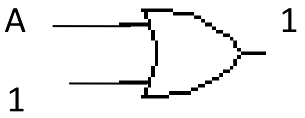
Rule (1): $A + 0 = A$

Any parameter enters gate OR with the zero (0) cause output with the same value of input parameter.



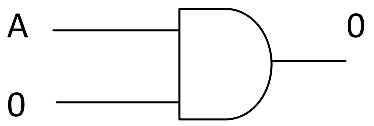
Rule (2): $A + 1 = 1$

Any A input in OR gate with 1, causes output 1 .



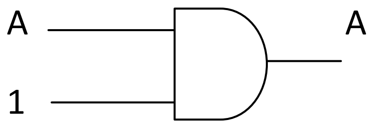
Rule (3): $A \cdot 0 = 0$

Any A input in AND gate with 0, causes 0 output.



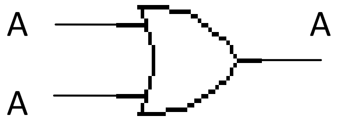
Rule (4): $A \cdot 1 = A$

Any A input in AND gate with 1, causes A output.



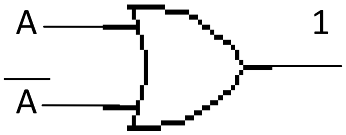
Rule (5): $A + A = A$

If 2 inputs in OR gate are A, output will be A.



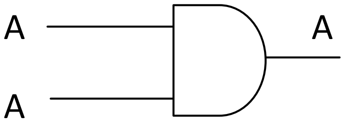
Rule (6): $A + \overline{A} = 1$

If the 2 inputs in OR gates are A , $\overline{A}$, the output will be always 1.



Rule (7): $A . A = A$

If the 2 inputs in AND gate are A , output will be always A.

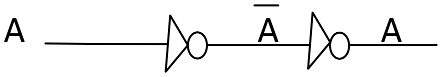


Rule (8): $A . \overline{A} = 0$

If input in AND gate are A , $\overline{A}$ output always will be 0.

Rule (9): $\overline{A} = A$

If invert A twice, output will be A .



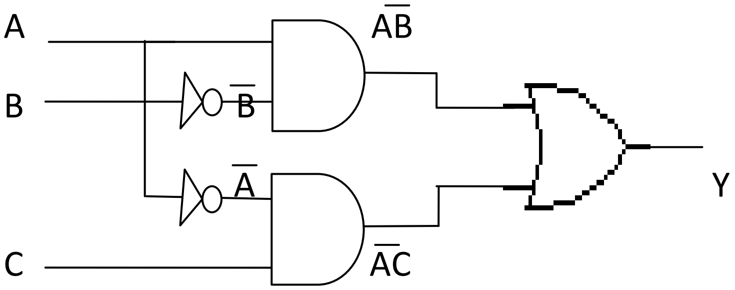
Rule (10): $A + AB = A$

Using rule (2) and (4):

$A + AB = A(1+B) = A \cdot 1= A$

The Boolean expression for logic circuit

To obtain Boolean expression for any logic circuit, we start from the left moving to final output of the circuit by writing the output of each gate. For example we have the following logic circuit.

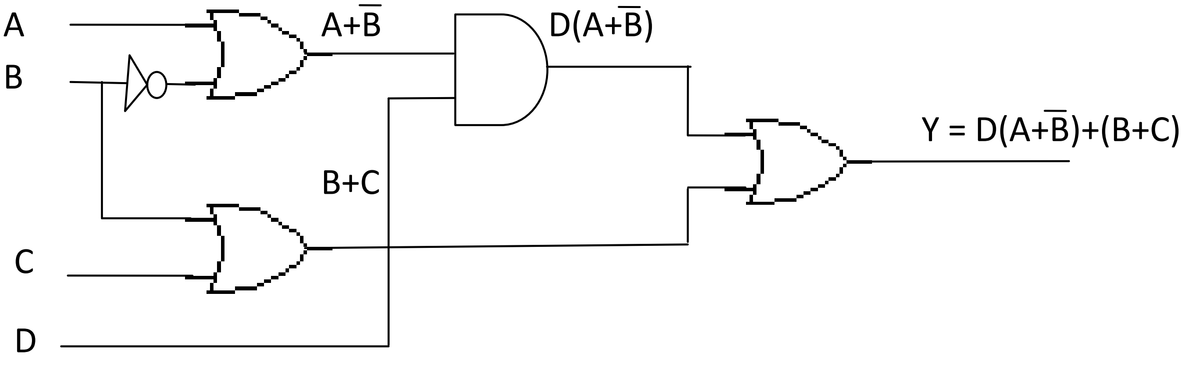


1. Boolean expression for AND gate, which has A and $\overline{B}$ inputsis $A\overline{B}$
2. Boolean expression for AND gate, which has $\overline{A}$ and C is $\overline{A}C$.
3. Boolean expression for OR gate, which has $A\overline{B}$, $\overline{A}C$ inputs is $A\overline{B} + \overline{A}C$

   Final output:

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

<u>Exp</u>: write Boolean expression for following logic circuit.



<u>Output</u> $Y = D\,(A+\overline{B}) + (B+C)$

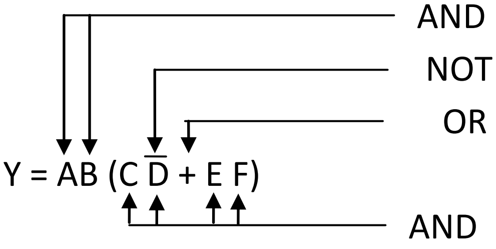
Implementation of logic circuit using a Boolean expression.

To impalement the following Boolean expression:

Y = AB ($C\overline{D}$ + EF)

Analyzing Boolean expression, we find:

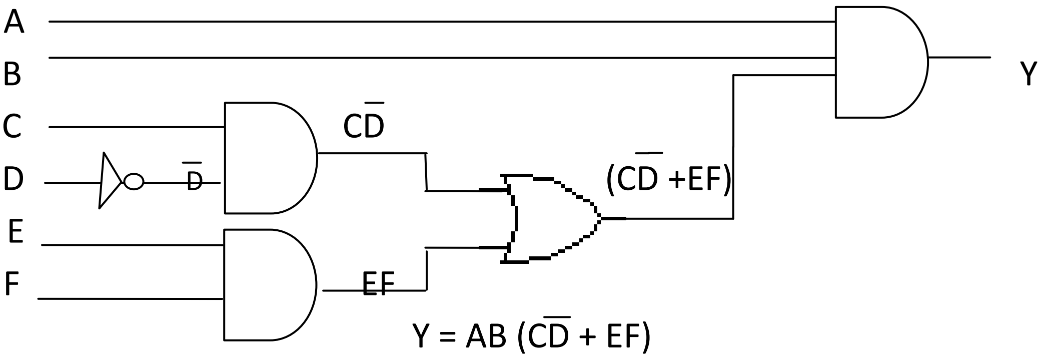
- Three AND gates AB,($C\overline{D}$+EF)
- One OR gate $C\overline{D}$ + EF
- One not gate.



First we implement ($C\overline{D}$ + EF), so we shall obtain the $C\overline{D}$ and EF. Before that, we shall have O.

Following sequence must be obtained:

1. NOT gate to implement $\overline{D}$.
2. Two AND gate with two inputs each to implement $C\overline{D}$ and EF.
3. OR gate with 2- inputs to implement ($C\overline{D}$+EF).
4. AND gate with 3- inputs to implement Y.



Y = AB ($C\overline{D}$ + EF)

Implementation of a logic circuit via a truth table

1. Determine from truth table the combinations of inputs which give Y=1.

3rd line A= 0, B=1, C=0

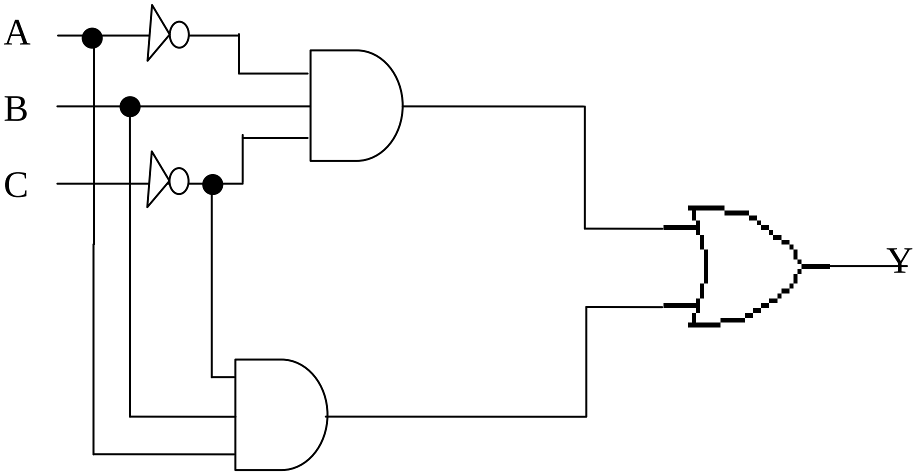
Y =1 , $\overline{A}B\overline{C}$ =1

7th line A =1 , B=1, C=0

$AB\overline{C}$=1

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

inputs			output
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0



Demorgen's theorems

Demorgan's theorems are important part of Boolean algebra. These theorems are using to convert algebra variables from main AND position to OR position and vis uers. Also these theorems help to remove bars from variables.

First Demorgen's theorem $\overline{A + B} = \overline{A}\ .\ \overline{B}$

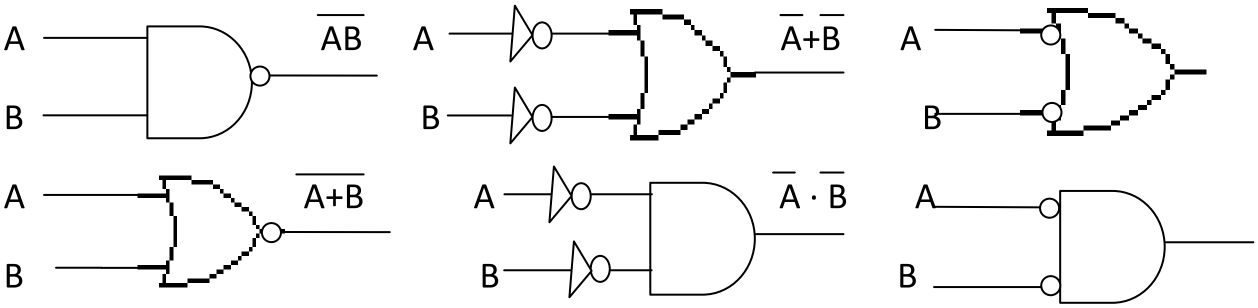
Second Demorgen's theorem $\overline{A\ .\ B} = \overline{A} + \overline{B}$

First theorem changes from main OR position to AND position. To NOR gate equivalent to AND gate, but with inverted inputs. Small circuit in input works as NOT gate.

Demorgen's theorems can be expanded to two or more variables. In equation form for four variables.

$\overline{A+B+C+D} = \overline{A}\ \overline{B}\ \overline{C}\ \overline{D}$ 1 D.T

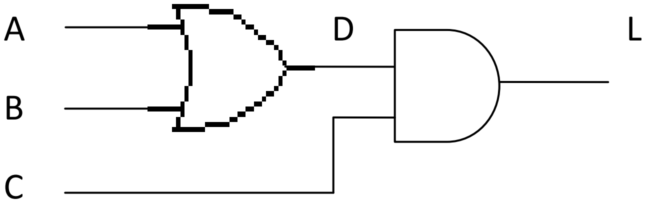
$\overline{A\ B\ C\ D} = \overline{A} + B + C + D$ 2 D.T


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

المدخلات		الخرج	
A	B	$\overline{A.B}$	$\overline{A}+\overline{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Circuit analysis

Exp.:

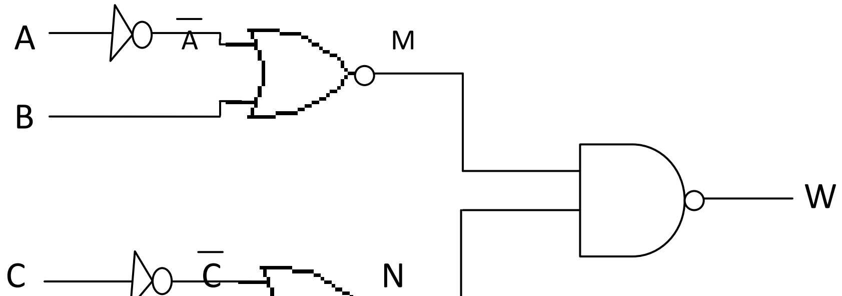


L = CD =C (A +B)

A	B	C	D	L
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

L = C(A +B)

exp.:



$W= MN = \overline{\overline{(\overline{A}+B)}.\overline{(\overline{C}+B)}}$

A	B	C	M	N	W
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	0	0	1
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	0	0	1

$$W = \overline{\overline{(\overline{A} + B)}.\overline{(\overline{C} + B)}}$$

Simplification of Boolean expression using Boolean algebra

Exp.:

$Y = AB + A(A+C) + B(A+C)$

$Y = AB + AA + AC + AB + BC$ Rule 7

$Y = AB + A + AC + AB + BC$ Rule 5

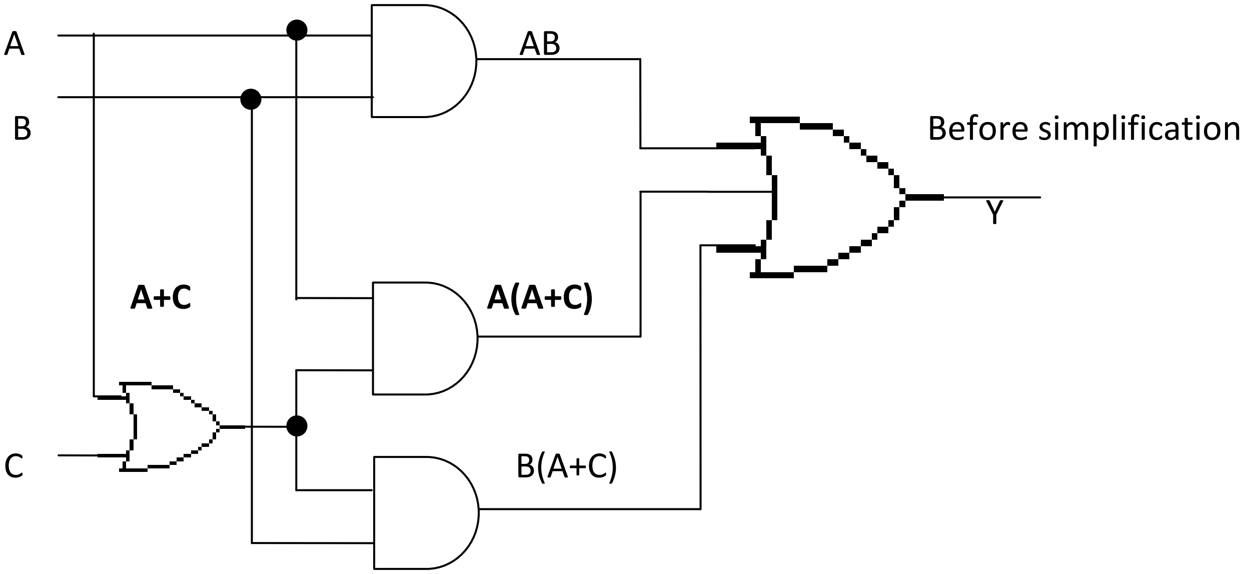
$AB + AB = AB$

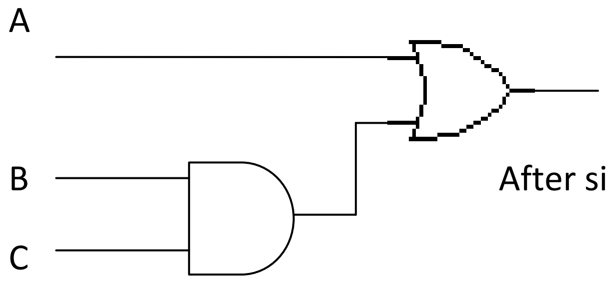
$Y = AB + A + AC + BC$

$Y = A(B+1+C)+BC$

$Y = A . 1 + BC$ ⟶ $A + 1 = 1$

$Y = A + BC$





After simplification

Exp.: Simplify $\overline{\overline{(}A+B+C+D)}+(A\,\overline{B}\,\overline{C}\,D)$ Demorgan's theorems

$= \overline{\overline{A}}\,\overline{B}\,\overline{C}\,\overline{D} + (A\,\overline{B}\,\overline{C}\,D) + A\,\overline{B}\,\overline{C}\,(\overline{D}+D)$

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

EXP.: $(\overline{A}+\overline{B})(A+\overline{B})(\overline{A}+B)$

$=[(\overline{A}+\overline{B})(A+\overline{B})]\,[(\overline{A}+B)(\overline{A}+\overline{B})]$

$= [\overline{B}]\,[\overline{A}]$

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

إيضاح

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

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

$=0+\overline{B}(\overline{A}+A)+\overline{B}$

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

Exp.: Simplify:

$A+\overline{A}B+\overline{A}BC$

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

$=A+\overline{A}B \longrightarrow A(1+B)+\overline{A}B \Longrightarrow (1+B)=1$

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

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

$=A+B$

Exp.:Simplify:

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

$$= \overline{A}\,\overline{B}\,(\,\overline{C} + C\,) + A\,\overline{B}\,\overline{C}$$

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

$$= \overline{B}\,(\,\overline{A} + A\,\overline{C}\,)$$

$$= \overline{B}\,(\,\overline{A} + \overline{\overline{A}}\,\overline{C}\,) \qquad \Longrightarrow \qquad (\,A + \overline{A}\,B\,) = A + B$$

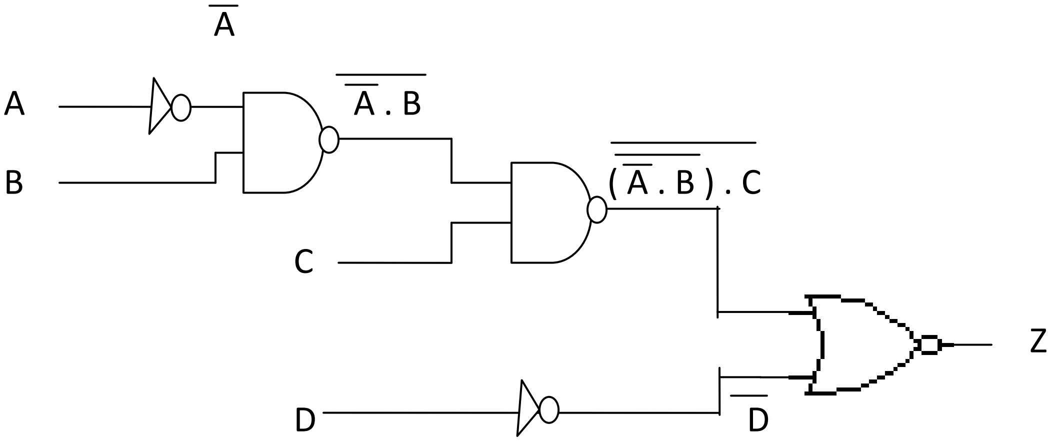
كما في المثال السابق

$$= \overline{B}\,(\,\overline{A} + \overline{C}\,)$$

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

Exp.: Gate level representation from Boolean equation.

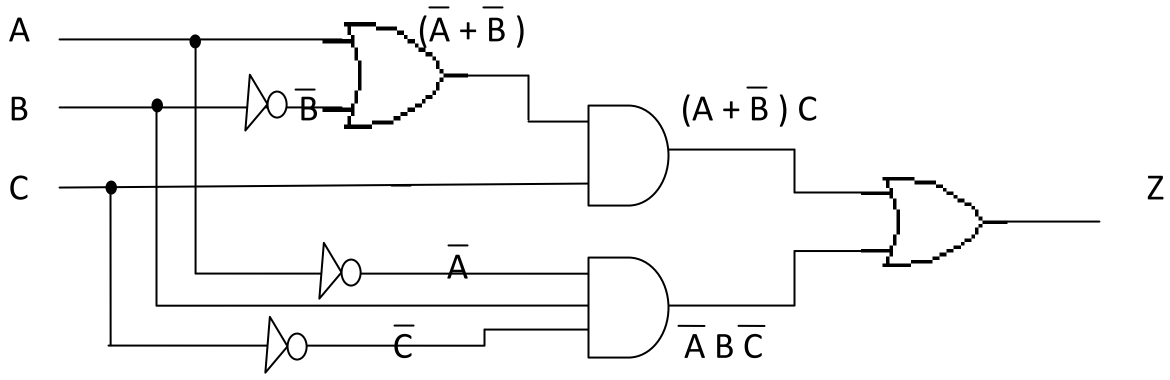
$$Z = \overline{(\,\overline{(\,\overline{\overline{A}\,.\,B}\,)\,.\,C}\,) + \overline{D}}$$



$\overline{A}$
A
B
$\overline{\overline{A}\,.\,B}$
$\overline{(\,\overline{A}\,.\,B\,)\,.\,C}$
C
Z
D
$\overline{D}$

Exp.: Circuit analysis going from gate – level to

a. Truth table!
b. Boolean equation!



$Z = (A + \overline{B})\,C + \overline{A}\,B\,\overline{C}$

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

Truth table

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

Exp. :

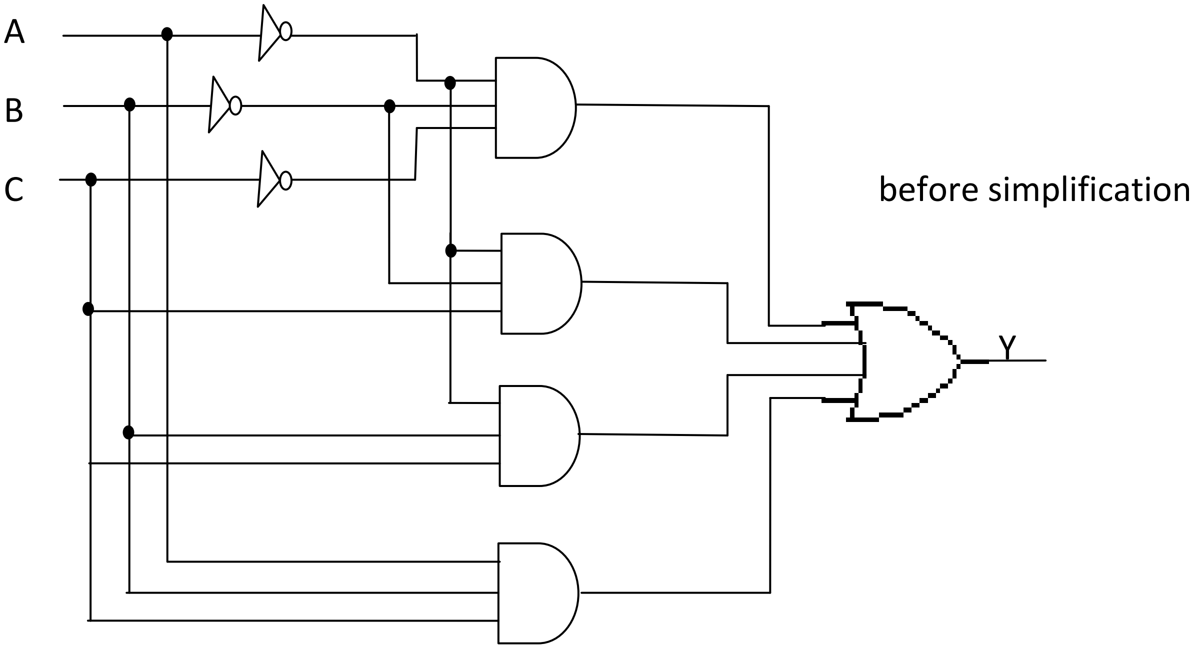
$Y = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\,B\,C$

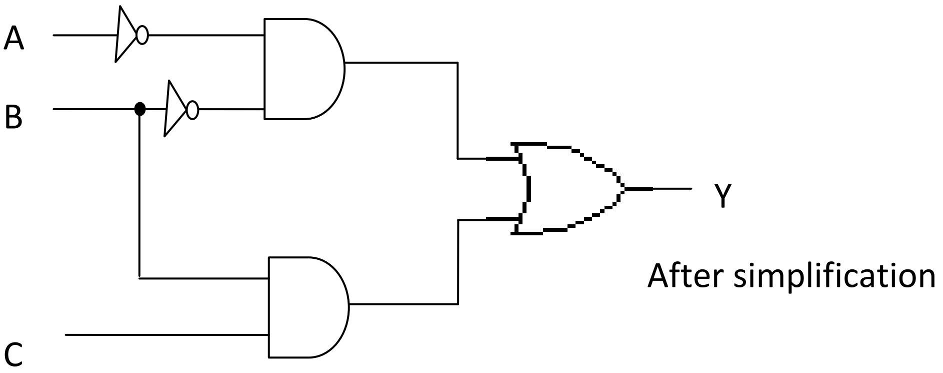
$Y = (\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,C) + (\overline{A}\,B\,C + A\,B\,C)$

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

$Y = \overline{A}\,\overline{B}\,.\,1 + B\,C\,.\,1$ Rule 6

$Y = \overline{A}\,\overline{B} + B\,C$ Rule 4





Truth table

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

$\overline{A}\ \overline{B}\ \overline{C} = 0\,0\,0\,0$

$\overline{A}\,\overline{B}\,C = 0\,0\,1$

$\overline{\mathrm{A}}\,\mathrm{B}\,\mathrm{C} = 0\,1\,1$

$\mathrm{A}\,\mathrm{B}\,\mathrm{C} = 1\,1\,1$

Boolean simplification

Exp: Simplify: $(\overline{A} + \overline{B} + \overline{C})(\overline{A} + \overline{B} + C)(A + \overline{B} + \overline{C})$

$$(\overline{A} + \overline{B} + \overline{C})(\overline{A} + \overline{B} + C)(A + \overline{B} + \overline{C})$$

$$(\overline{A} + \overline{B} + \overline{C})(\overline{A} + \overline{B} + C)(A + \overline{B} + \overline{C})(\overline{A} + \overline{B} + \overline{C}) \qquad \underline{\text{Rule 7}}$$

$$[(\underline{\overline{A} + \overline{B}} + \overline{C})(\underline{\overline{A} + \overline{B}} + C)][(A + \underline{\overline{B} + \overline{C}})(\overline{A} + \underline{\overline{B} + \overline{C}})]$$

Explanation:

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

$$\underbrace{AA}_{\boxed{= A}} + A\overline{B} + AB + \underbrace{B\overline{B}}_{\boxed{= 0}}$$

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

$$A + A(\underbrace{\overline{B} + B}_{\boxed{= 1}}) \qquad A + A = A$$

So: $(\overline{A} + \overline{B})(\overline{B} + \overline{C}) =$

$$\overline{B} + \overline{A}\,\overline{C} \quad \Longrightarrow$$

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

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

$$\overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{B}\underbrace{(1 + \overline{C})}_{\boxed{= 1}}$$

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

$$\overline{B}(\overline{A} + 1) + \overline{A}\,\overline{C} = \overline{B} + \overline{A}\,\overline{C}$$

Exp.: Simplify $\overline{A\,B\,(\,C\,D + E\,F\,)}$

$\overline{A\,B} + (\,\overline{C\,D + E\,F}\,)$ | Demorgan's theorems

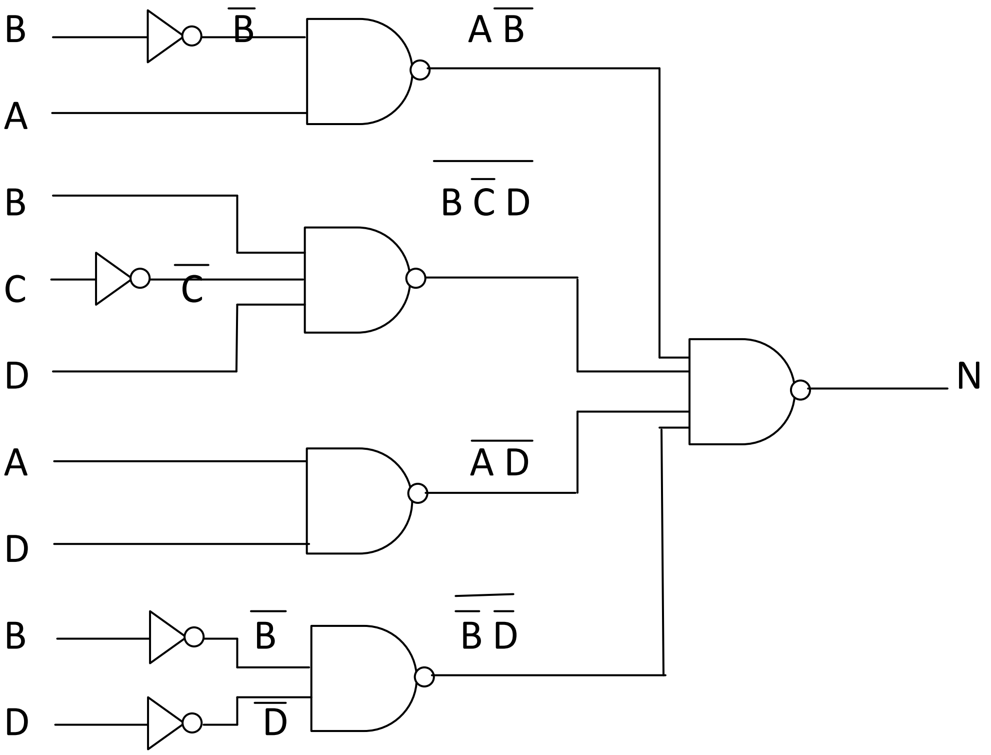
$\overline{A} + \overline{B} + (\,\overline{C\,D}\,)(\,\overline{E\,F}\,)$

$\overline{A} + \overline{B} + (\,\overline{C} + \overline{D}\,)\,(\,\overline{E} + \overline{F}\,)$

$\overline{A} + \overline{B} + \overline{C}\,\overline{E} + \overline{C}\,\overline{F} + \overline{D}\,\overline{E} + \overline{D}\,\overline{F}$

Exp.: Circuit implementation. (NAND- circuits).

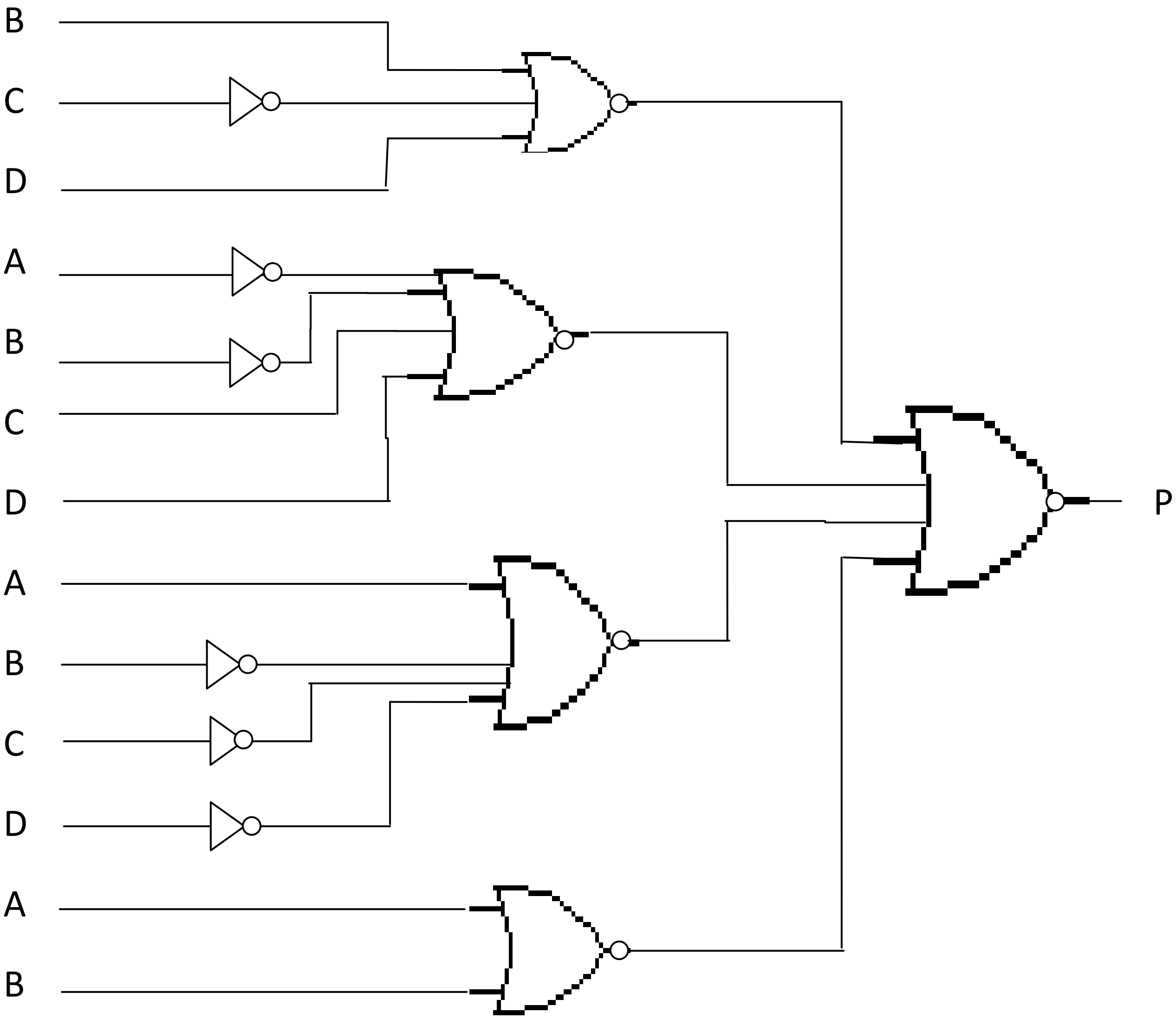
$$N = \overline{\overline{A\,\overline{B}}\;\;\overline{B\,\overline{C}\,D}\;\;\overline{A\,D}\;\;\overline{\overline{B}\,\overline{D}}}$$


B
$\overline{B}$
$\overline{A\,\overline{B}}$
A
B
$\overline{B\,\overline{C}\,D}$
C
$\overline{C}$
D
N
A
$\overline{A\,D}$
D
B
$\overline{B}$
$\overline{\overline{B}\,\overline{D}}$
D
$\overline{D}$

$$N = \overline{\overline{A\,\overline{B}}\;\;\overline{B\,\overline{C}\,D}\;\;\overline{A\,D}\;\;\overline{\overline{B}\,\overline{D}}}$$

Exp.: Circuit implementation

$$P = \overline{\overline{(B + \overline{C} + D)} + \overline{(\overline{A} + \overline{B} + C + D)} + \overline{(A + \overline{B} + \overline{C} + \overline{D})} + \overline{(A + C)}}$$



Truth table derived from a sum of products (sop) Expression

Exp.:

Derive the truth table from the sum of products expression.

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

The equation has three variables. For each AND term in the equation, List the output as a 1 .

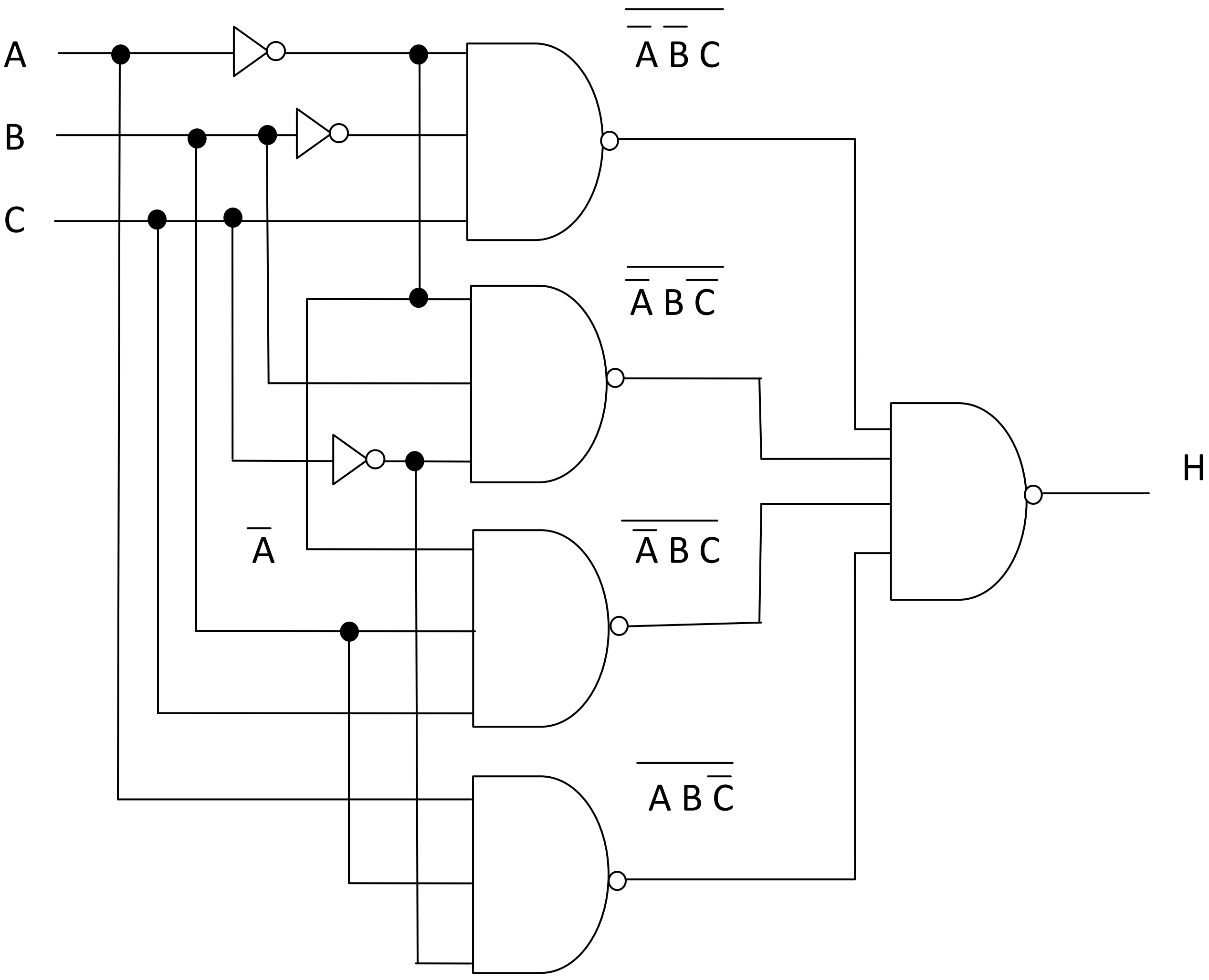
$$H = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}\,B\,C + A\,B\,\overline{C}$$

0 0 1+ 0 1 0 + 0 1 1 + 1 1 0

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

Exp.: How to use only NAND gates to draw logic circuit.

$$H=\overline{\overline{(\overline{A}\,\overline{B}\,C\,)}\,\overline{(\overline{A}\,B\,\overline{C}\,)}\,\overline{(\overline{A}\,B\,C)}\,\overline{(A\,B\,\overline{C})}}$$


A
B
C
$\overline{\bar{A}\,\bar{B}\,C}$
$\overline{\bar{A}\,B\,\bar{C}}$
$\bar{A}$
$\overline{\bar{A}\,B\,C}$
$\overline{A\,B\,\bar{C}}$
H

$$H = \overline{\overline{(\bar{A}\,B\,\bar{C})}\;\overline{(\bar{A}\,B\,\bar{C})}\;\overline{(\bar{A}\,B\,C)}\;\overline{(A\,B\,\bar{C})}}$$

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

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

NOR – NOR productof sums circuit

Exp.: Convert the OR – AND product of sum circuit designed in example to NOR –NOR POS circuit.

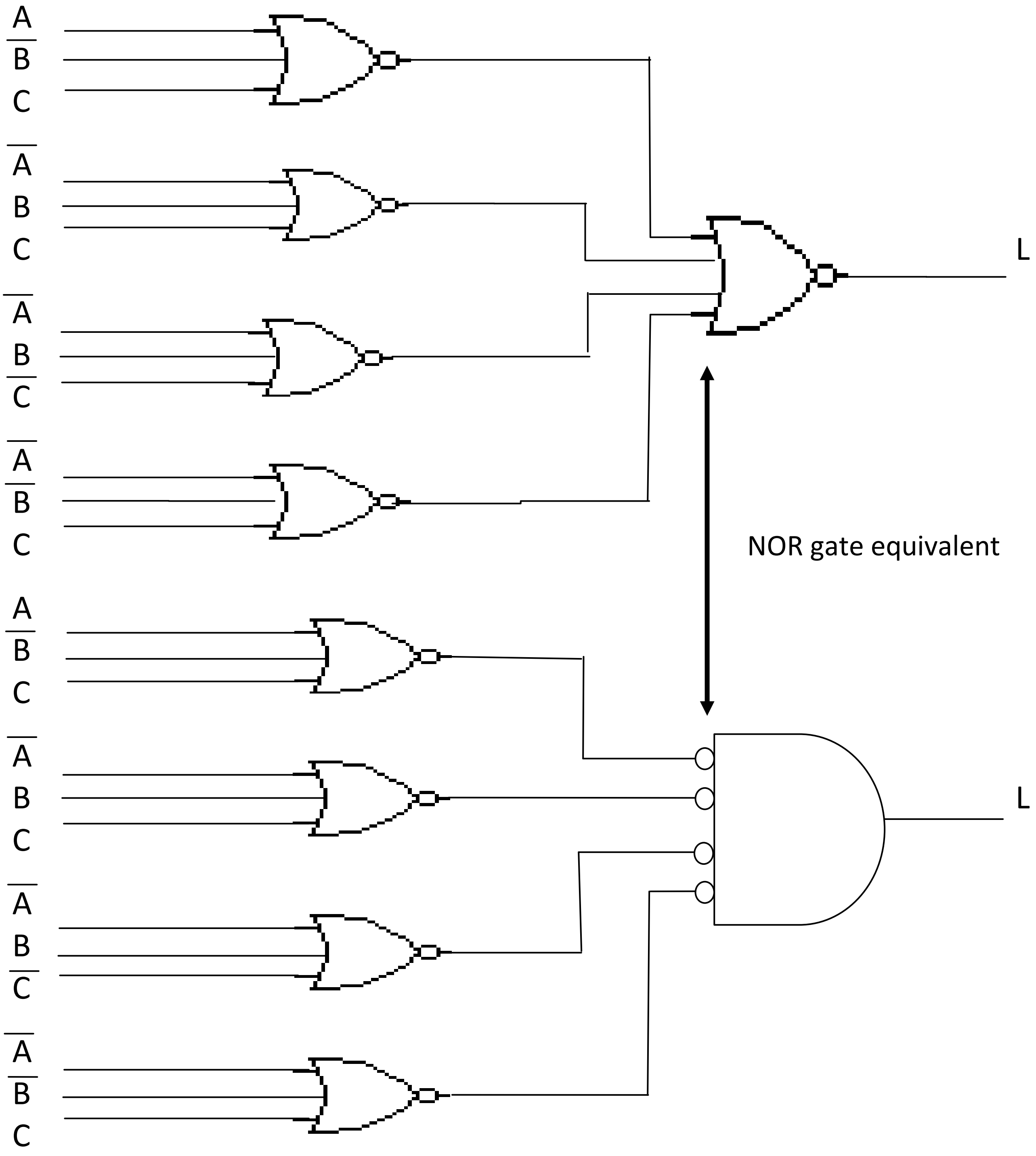
Solution: apply demorgan's theorem to convert OR – AND circuit to a NOR – NOR circuit a according to the following steps:

1. Double inverter the expression for L
2. Keep the top , inversion bar
3. Apply Demorgan's theorem to the bottom inversion bar to elimate the AND operation.
4. Design the NOR – NOR circuit from this eq..

$$L = (A + \overline{B} + C)(\overline{A} + B + C)(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + C)$$

$$L = \overline{\overline{(A + \overline{B} + C)(\overline{A} + B + C)(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + C)}}$$

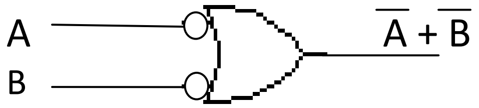
$$L = \overline{\overline{(A + \overline{B} + C)} + \overline{(\overline{A} + B + C)} + \overline{(\overline{A} + B + \overline{C})} + \overline{(\overline{A} + \overline{B} + C)}}$$


A
$\overline{B}$
C
$\overline{A}$
B
C
$\overline{A}$
B
$\overline{C}$
$\overline{A}$
$\overline{B}$
C
L
NOR gate equivalent
A
$\overline{B}$
C
$\overline{A}$
B
C
$\overline{A}$
B
$\overline{C}$
$\overline{A}$
$\overline{B}$
C
L

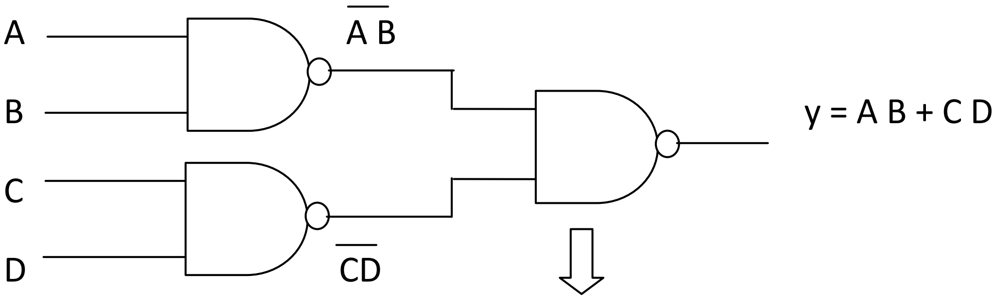
Design of combination logic circuit using NAND an NOR gates.

We will see here how to use NAND and NOR gates to implement logic circuits. NAND gate is equivalent to" negative – OR".NOR gate equivalent to "negative- AND". We will see also, that using AND, "negative - OR 'gates we can read logic diagram at the circuit.

$$\underbrace{\overline{A . B}}_{\text{NAND}} = \underbrace{\overline{A} + \overline{B}}_{\text{negative – OR}}$$



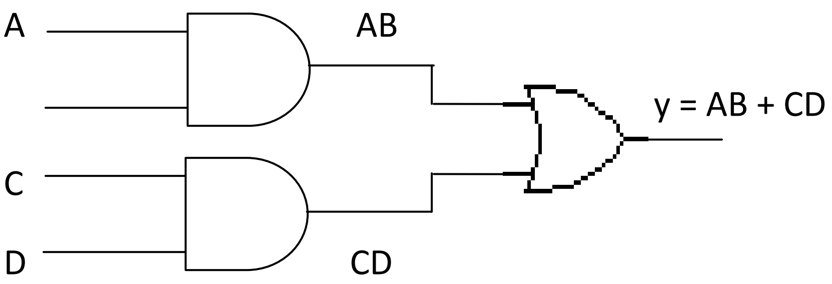
Let us see for example the following logic circuit.



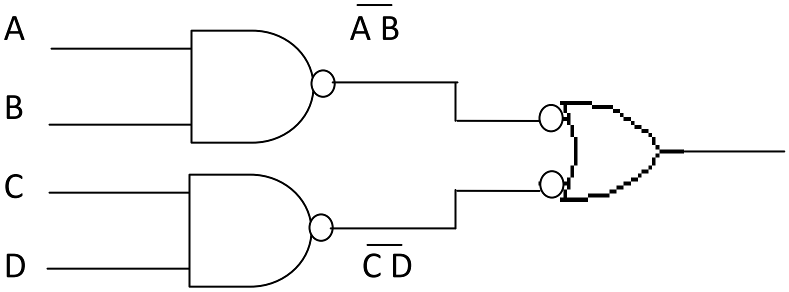
$$y = \overline{(\overline{AB}) . (\overline{CD})} = \overline{\overline{A B}} + \overline{\overline{CD}}$$

$$= AB + CD$$

If we see last equation, we can obtain y , by using two AND gates and one OR gate.

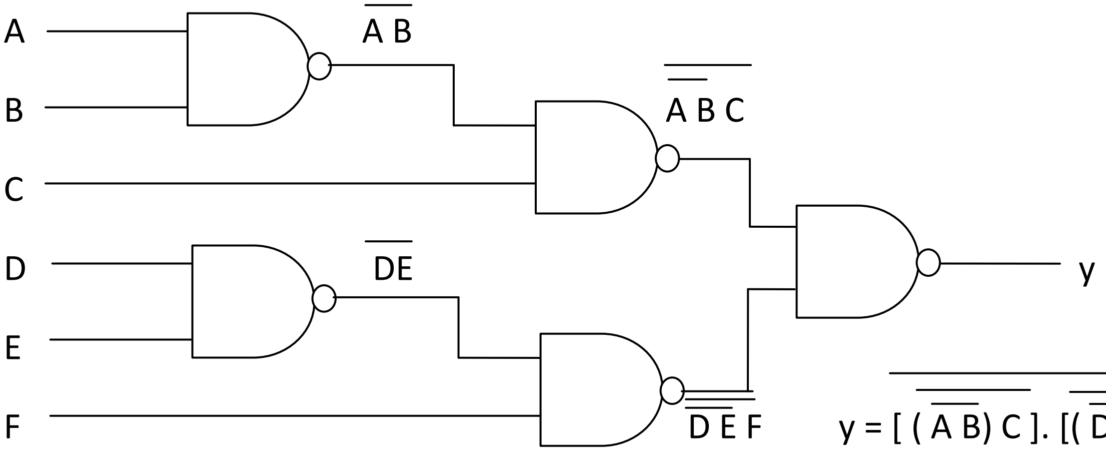


We can also use negative – OR gate instate of NAND gate at the right



$y = AB + CD$

Exp. : We have logic circuit implemented by NAND gates .We need to redraw it by using "negative – OR "gates.



$y = \overline{[\overline{(\overline{AB})C}].[\overline{(\overline{DE})F}]}$

First we shall obtain output equation:

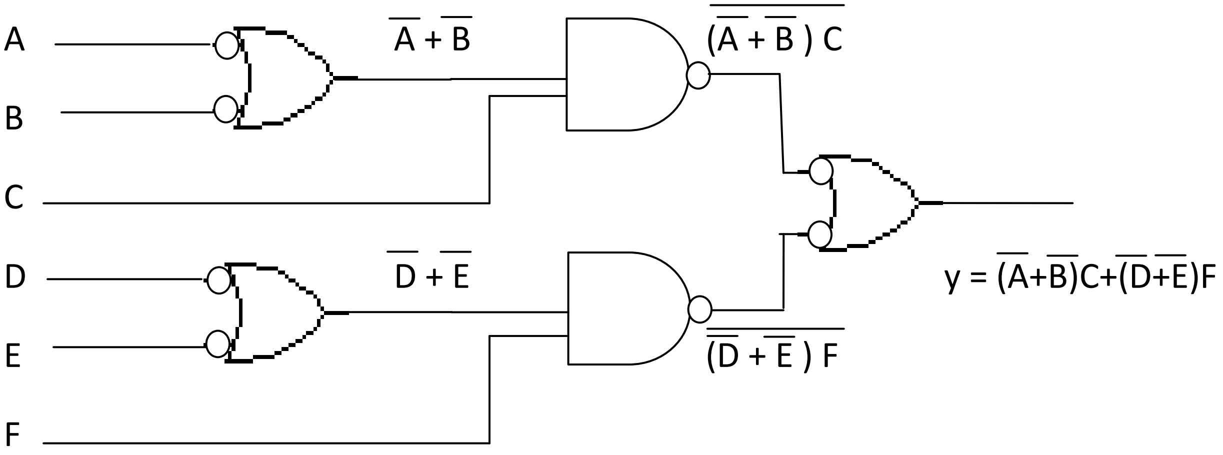
$$Y = \overline{[\overline{(\overline{AB})C}].[\overline{(\overline{DE})F}]}$$

$$Y = \overline{[\overline{(\overline{A}+\overline{B})C}].[\overline{(\overline{D}+\overline{E})F}]}$$

$$Y = \overline{\overline{(\overline{A}+\overline{B})C}} + \overline{\overline{(\overline{D}+\overline{E})F}}$$

$$Y = (\overline{A}+\overline{B})C + (\overline{D}+\overline{E})F$$

Using equivalent" negative –OR" gate to NAND gate, we obtain equivalent circuit:

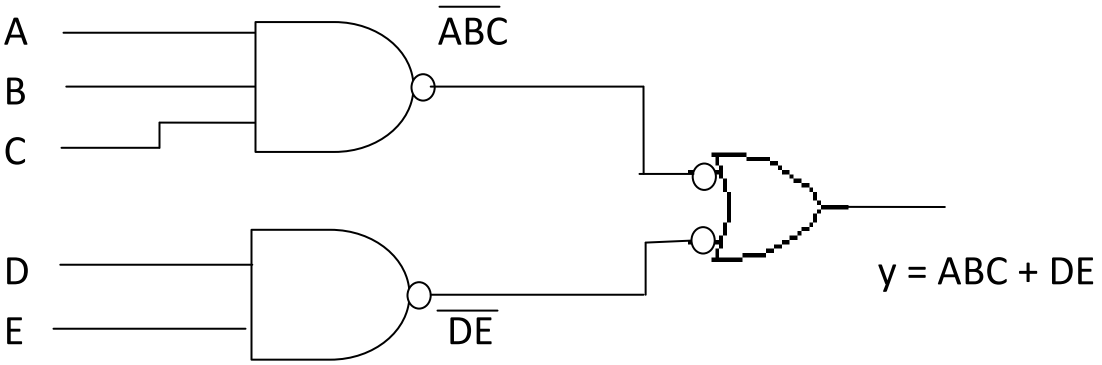

A
B
C
D
E
F
$\overline{A} + \overline{B}$
$\overline{(\overline{A} + \overline{B})\, C}$
$\overline{D} + \overline{E}$
$\overline{(\overline{D} + \overline{E})\, F}$
$y = (\overline{A}+\overline{B})C+(\overline{D}+\overline{E})F$

Exp.: draw logic circuit using NAND gates.

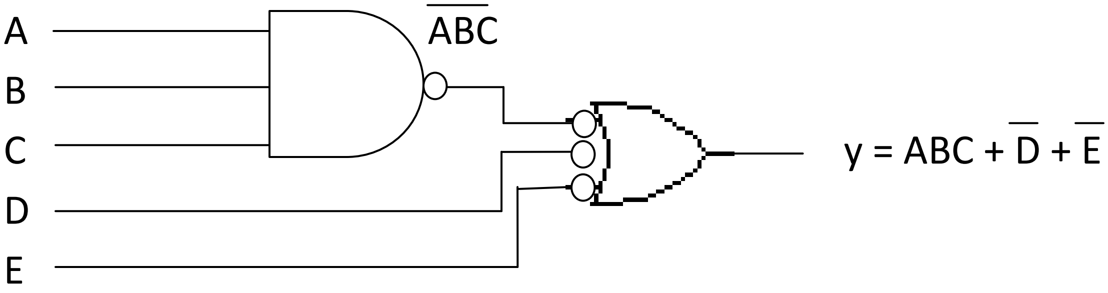
a- $Y = A\ B\ C + D\ E$

b- $Y = A\ B\ C + \overline{D} + \overline{E}$

a//



b//



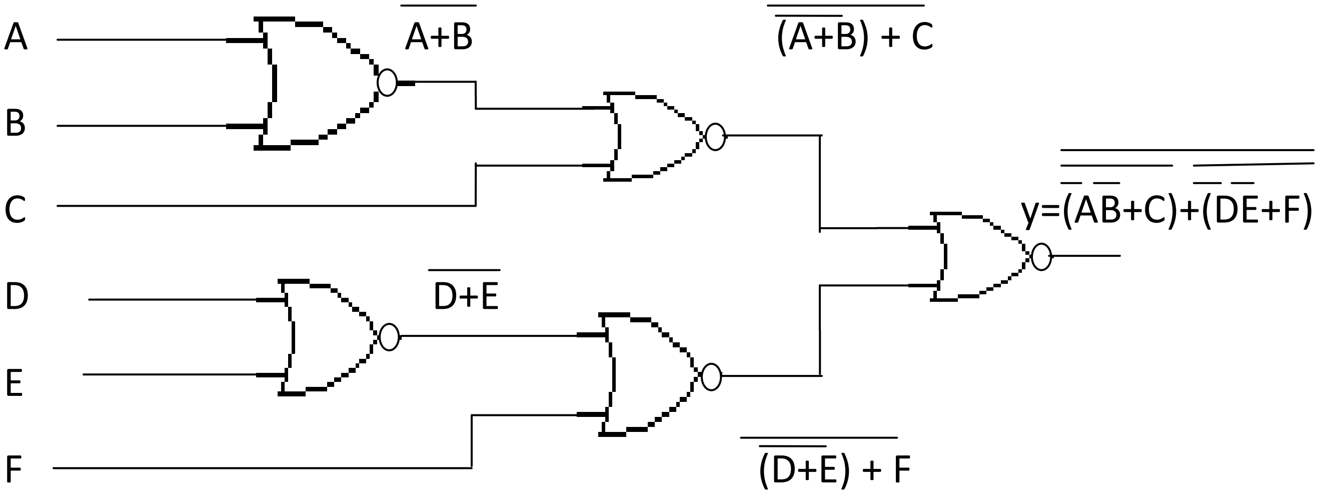
NOR - logic

$$Y = \overline{[\,\overline{(\overline{A + B}) + C}\,] + [\,\overline{(\overline{D + E}) + F}\,]}$$

$$Y = \overline{[\,\overline{\bar{A}\,\bar{B} + C}\,] + [\,\overline{\bar{D}\,\bar{E} + F}\,]}$$

$$= \overline{\overline{(\bar{A}\,\bar{B} + C)}}.\overline{\overline{(\bar{D}\,\bar{E} + F)}}$$

$$= (\bar{A}\,\bar{B} + C).(\bar{D}\,\bar{E} + F)$$


A
B
C
D
E
F
$\overline{A+B}$
$\overline{(\overline{A+B}) + C}$
$\overline{D+E}$
$\overline{(\overline{D+E}) + F}$
$y=\overline{\overline{(\bar{A}\bar{B}+C)}+\overline{(\bar{D}\bar{E}+F)}}$

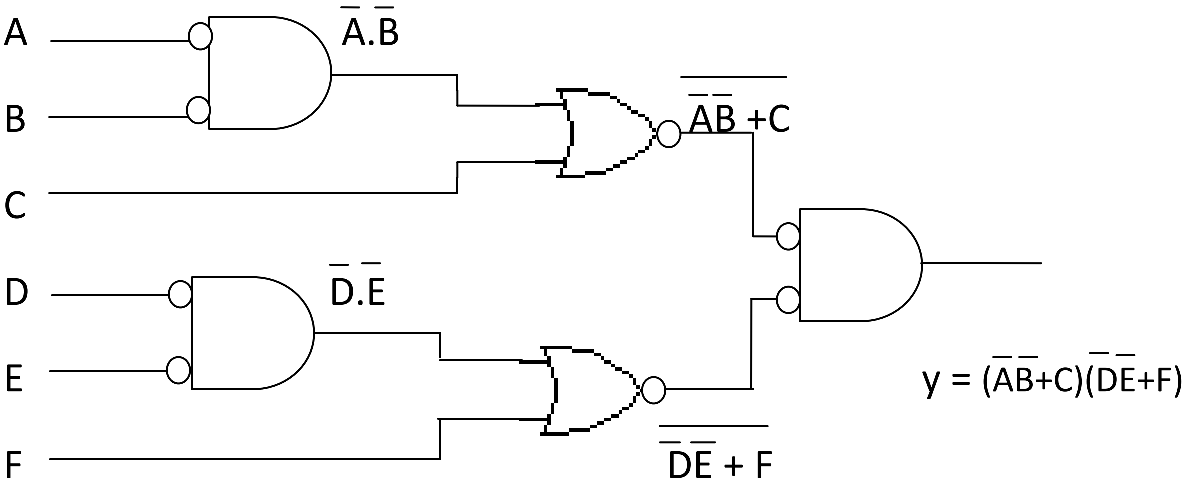
للإيضاح

$$\overline{(A+B)}=\bar{A}\,\bar{B}$$

$$\overline{(D+E)}=\bar{D}\,\bar{E}$$

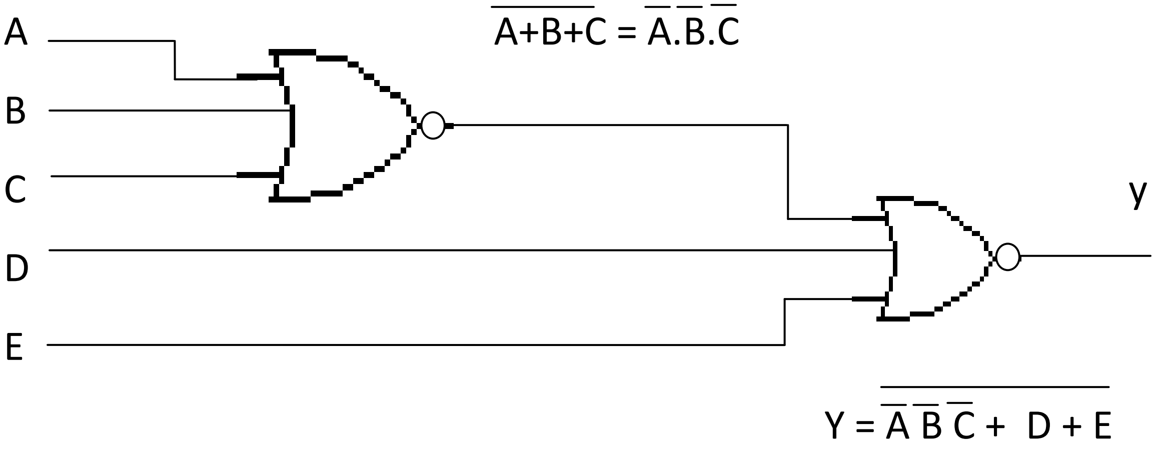
Using negative – AND gates, equivalent to NOR gates.

We obtain:


A
B
C
D
E
F
$\bar{A}.\bar{B}$
$\overline{\bar{A}\bar{B}+C}$
$\bar{D}.\bar{E}$
$\overline{\bar{D}\bar{E}+F}$
$y = (\bar{A}\bar{B}+C)(\bar{D}\bar{E}+F)$

Exp: Using NOR gates only, implement the following equation.

$y = \overline{\bar{A}\bar{B}\bar{C}+ (D + E)}$


A
B
C
D
E
$\overline{A+B+C} = \bar{A}.\bar{B}.\bar{C}$
y
$Y = \overline{\bar{A}\,\bar{B}\,\bar{C} + D + E}$

KARNAUGH MAPING

A karnaugh map, or k-map as it is often referred to, is used to simplify a logic expression and derive from the truth table.

The k- map contains the same information as a truth table, representing the logic function in a slightly different format. K-map contains a cell for each input combination. A two- variable K-map has 4 cells, a three- variable k-map has 8 cells ….. (2^n).

	$\overline{B}$	B
$\overline{A}$		
A		

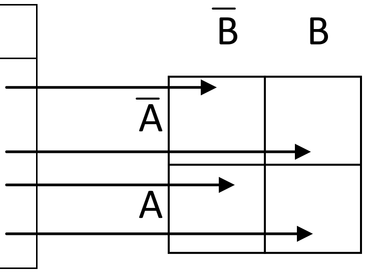
	$\overline{CD}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$				
$\overline{A}B$				
AB				
$A\overline{B}$				

	$\overline{C}$	C
$\overline{AB}$		
$\overline{A}B$		
AB		
$A\overline{B}$		

How to fill k- map ?

Truth table

A	B	Y
0	0	$\overline{A}\,\overline{B}$
0	1	$\overline{A}\,B$
1	0	$A\,\overline{B}$
1	1	$A\,B$



Exp.: make k – map from truth table.

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

truth table

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

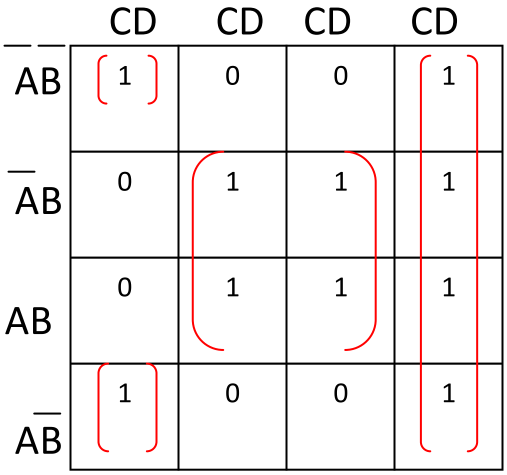
A	B	Y	
0	0	0	
0	1	0	
1	0	1	→ $A\overline{B}$
1	1	1	→ AB

Simplification using k - maps

1. Enter the complete output specification for the logic function in the k – map for every possible input combination.
2. Select one output state to group, either the 1's or 0’s. All the currences of the selected output must be included in the groups formed to specify the function.

3. Group adjacent cells in group of 2^n. Rectangular groups of 1,2,4,8 and 16 are the only allowable groups on k – map: at up to four variables.

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



	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	0	1	1	1
AB	0	1	1	1
$A\bar{B}$	1	0	0	1

4. The larger the group, the more input variables are criminated. For a group containing 2^n cells, n input variables are elimination.
5. The simplified terms contain only the variables common to all output in the group.
6. Each group represents a term in the function equation the tower number of groups, the tower terms are needed in the final equation.
7. Groups can overlap.
8. The first goal in grouping the outputs is to minimize the number of groups.
9. The second goal in grouping the outputs is to maximize the size of the group in order to minimize the number of input variables in each term.
10. There may be more than one correct grouping for a given function.

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	1	0	1
$\overline{A}B$	1	1	0	0
AB	1	0	1	1
$A\overline{B}$	1	1	0	1

To simplify the equations, only those variables common to all 1' s in the group are included in the AND terms.

In other words, if an input variable changes between adjacent cell in a group, but the output remains the same ,that input variable is not needed in the AND term.

Each group of adjacent cells on the k – map is specified by a separate AND term in the equation. The final equation formed by O-Ring the individual AND terms.

When forming the simplified sum of product expression with k- map, the 1' s are grouped in as large a group as possible to minimize the input variables in each term. Each output value of 1 is included in some group, but additional groups are not formed by unnecessary overlapping of extra groups so that the number of AND terms in the final expression is minimized.

Exp.:

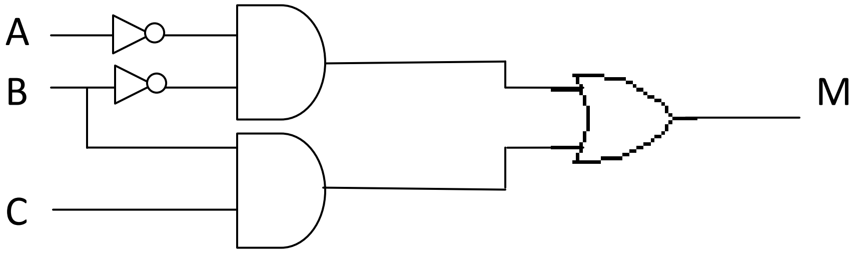
Simplify the logic function specified in the truth table using k –map.

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

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

	$\overline{C}$	C
$\overline{A}\,\overline{B}$	1	1
$\overline{A}B$		1
AB		1
$A\overline{B}$		

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



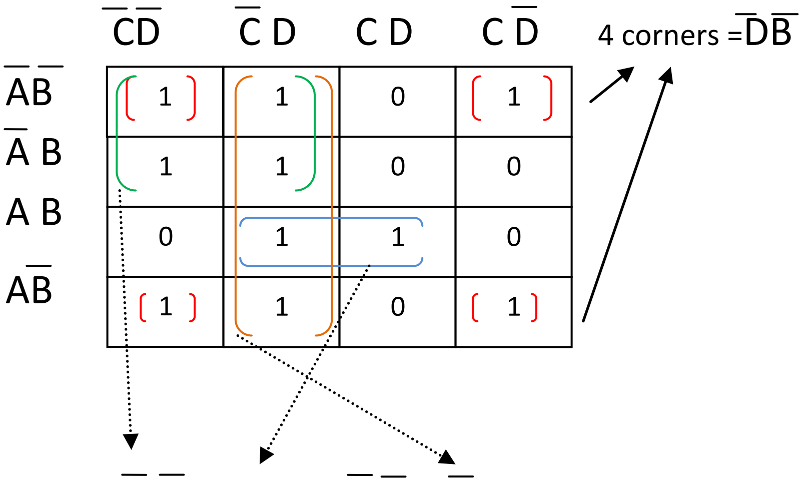
Exp.: before simplification

A B C D	N
0 0 0 0	1
0 0 0 1	1
0 0 1 0	1
0 0 1 1	0
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	0
1 0 0 0	1
1 0 0 1	1
1 0 1 0	1
1 0 1 1	0
1 1 0 0	0
1 1 0 1	1
1 1 1 0	0
1 1 1 1	1

$$N = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,D + \overline{A}\,\overline{B}\,C\,\overline{D} + \overline{A}\,B\,\overline{C}\,\overline{D} +$$

$$\overline{A}\,B\,\overline{C}\,D + A\,\overline{B}\,\overline{C}\,\overline{D} + A\,\overline{B}\,\overline{C}\,D + A\,\overline{B}\,C\,\overline{D} +$$

$$A\,B\,\overline{C}\,D + A\,B\,C\,D$$



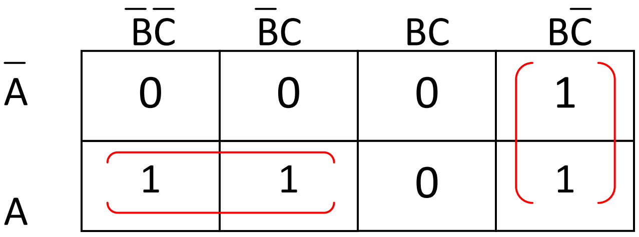
	$\overline{C}\,\overline{D}$	$\overline{C}\,D$	$C\,D$	$C\,\overline{D}$
$\overline{A}\,\overline{B}$	1	1	0	1
$\overline{A}\,B$	1	1	0	0
$A\,B$	0	1	1	0
$A\,\overline{B}$	1	1	0	1

4 corners $=\overline{D}\,\overline{B}$

$$N = \overline{A}\,\overline{C} + A\,B\,D + \overline{B}\,\overline{D} + \overline{C}\,D$$

Exp.: k – map

Design logic circuit in simple view using truth table:



	$\overline{B}\,\overline{C}$	$\overline{B}C$	BC	$B\overline{C}$
$\overline{A}$	0	0	0	1
A	1	1	0	1

inputs			output
A	B	C	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

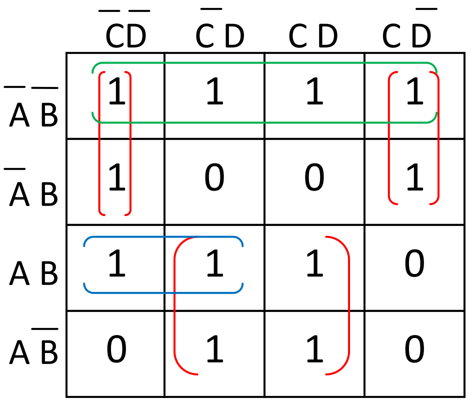
Before simplification:

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

After simplification:

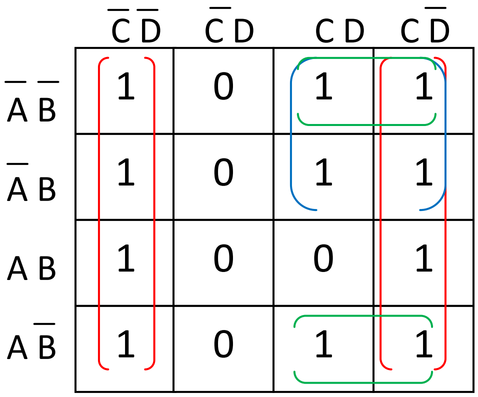
$Y = A\, \overline{B} + B\, \overline{C}$

<u>Examples for k - map</u>



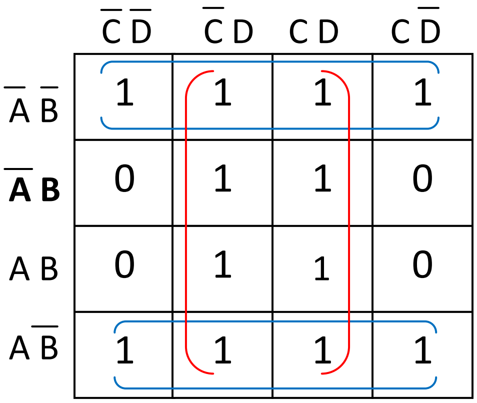
	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	1	1	1
$\overline{A}B$	1	0	0	1
AB	1	1	1	0
$A\overline{B}$	0	1	1	0

A



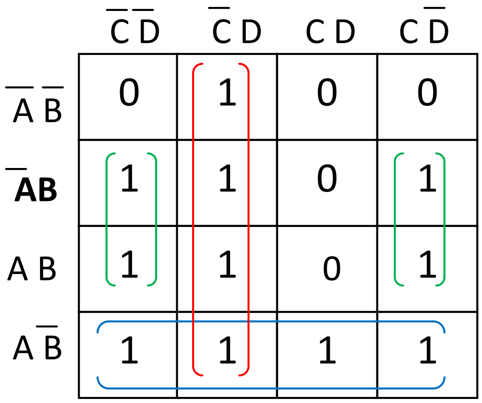
	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	0	1	1
$\overline{A}B$	1	0	1	1
AB	1	0	0	1
$A\overline{B}$	1	0	1	1

B



	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	1	1	1
$\overline{A}B$	0	1	1	0
AB	0	1	1	0
$A\overline{B}$	1	1	1	1

C



	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	0	1	0	0
$\overline{A}B$	1	1	0	1
AB	1	1	0	1
$A\overline{B}$	1	1	1	1

D

$\boxed{A}$: $Y=AB\overline{C}+AD+\overline{A}B\overline{D}+\overline{A}\,\overline{B}$

$\boxed{B}$: $Y=\overline{A}C+\overline{B}C+\overline{D}$

$\boxed{C}$: $Y=\overline{B}+D$

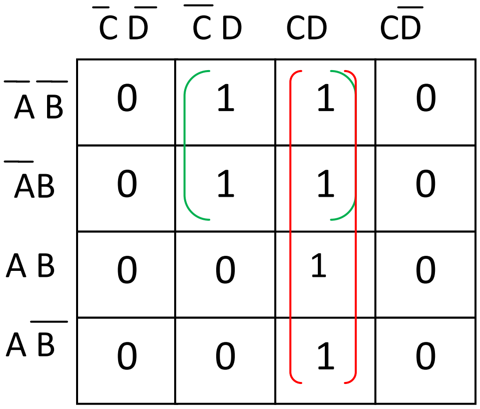
$\boxed{D}$: $Y=\overline{C}D+A\overline{B}+B\overline{D}$

Exp.: Write Boolean equation from truth table and simplify using k – map.

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

Now we draw k – map:

A	B	C	D	y
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	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1



	$\overline{C}\,\overline{D}$	$\overline{C}\,D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	0	1	1	0
$\overline{A}B$	0	1	1	0
$A\,B$	0	0	1	0
$A\,\overline{B}$	0	0	1	0

$y = \overline{A}\,D + CD$

Exp.: XOR and XNOR grouping of k – maps.

A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
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	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

	$\bar{C}\,\bar{D}$	$\bar{C}\,D$	$C\,D$	$C\,\bar{D}$
$\bar{A}\,\bar{B}$	1	0	1	0
$\bar{A}\,B$	0	1	0	1
$A\,B$	1	0	1	0
$A\,\bar{B}$	0	1	0	1

$$F = \bar{A}\,\bar{B}\,\bar{C}\,\bar{D} + \bar{A}\,\bar{B}\,C\,D + \bar{A}\,B\,\bar{C}\,D + \bar{A}\,B\,C\,\bar{D} + A\,B\,\bar{C}\,\bar{D} +$$

$$A\,B\,C\,D + A\,\bar{B}\,\bar{C}\,D + A\,\bar{B}\,C\,\bar{D}$$

$$F = \bar{A}\,\bar{B}\,(\overline{C \oplus D}) + A\,B(\overline{C \oplus D}) + \bar{A}\,B(C \oplus D) + A\,\bar{B}\,(C \oplus D)$$

$$F = (\overline{A \oplus B})(\overline{C \oplus D}) + (A \oplus B)(C \oplus D)$$

$$F = \overline{(A \oplus B) \oplus (C \oplus D)}$$

Binary adders and subtractions

Adders

1. The half – adder circuit.

We stuted four rolls of binary addition with two input A, B and the output were sum (S) and the carry(C). These rolls are:

input		output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

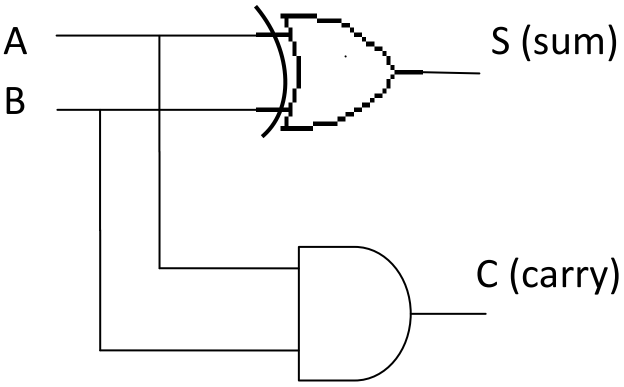
0 + 0 = 0 without carry

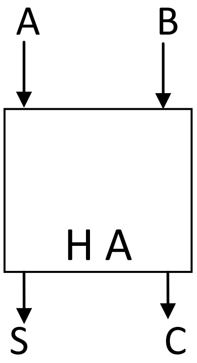
0 + 1 = 1 without carry

1 + 0 = 1 without carry

1 + 1 = 10 or 0 and carry 1

From studding of columns "S", we find that 'S' is equal to the output of XOR gate.
And "C "column is AND gate.
Circui, which implements the output of sum of A, B, is called half-adder circuit.





Box-diagram
HA=Half- adder
Or block diagram

The logic simple equation for outputs "S","C' can be obtained from truth table directly:

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

$$C=AB$$

2.The full adder circuit.

Half- adder can not operate with three inputs. So we need to another new circuit, which can make summation for three inputs at same time. This circuit is called full-adder.

Full – adder circuit is a combination circuit which can make summation for three numbers at same time with two outputs. Two inputs A and B and the third input is C in (input carry). It is the remain or carry number from sum of previes two inputs A, B. There are two outputs" S" (sum) and " C"(carry). Truth table for full- adder is:

inputs			Outputs	
A	B	Cin	S	C
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

0+0+0=0	without carry
0+0+1=1	without carry
0+1+0=1	without carry
0+1+1=10	with carry 1
1+0+0=1	without carry
1+0+1=10	with carry 1
1+1+0=10	with carry 1
1+1+1=11	with carry 1

From truth table we can write the equations for outputs S, C . From output columns we have:

$S = \overline{A}\,\overline{B}\,Cin + \overline{A}\,B\,\overline{C}in + A\,\overline{B}\,\overline{C}in + A\,B\,Cin$

$C = \overline{A}\,B\,Cin + A\,\overline{B}\,Cin + A\,B\,\overline{C}in + A\,B\,Cin$

Starting from S, we can simplify its equation, so:

$S = \overline{A}\,\overline{B}\,Cin + \overline{A}\,B\,\overline{C}in + A\,\overline{B}\,\overline{C}in + A\,B\,Cin$

$(\overline{A}\,B + A\,\overline{B}\,)\,\overline{C}in + (\overline{A}\,\overline{B} + A\,B\,)\,Cin$

$\overline{A}\,B + A\,\overline{B}$ – implements XOR gate:

$\overline{A}\,\overline{B} + A\,B$ - implements XNOR gate:

So:

$S = (A \oplus B\,)\,\overline{C}in + (\overline{A \oplus B}\,)Cin$

$S = (A \oplus B\,) \oplus Cin$

$S = A \oplus B \oplus Cin$

So we can draw or implement equation with two XOR gates. First with two inputes A, B, and the second with the output of the first, as first input and Cin.

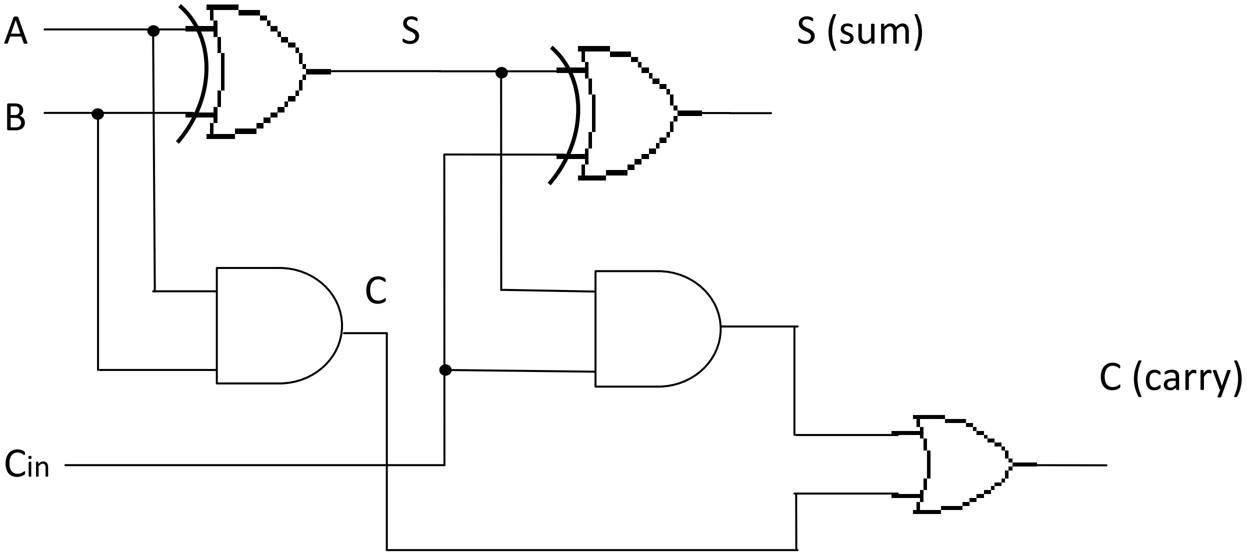
Now we simplify C equation:

$C = \bar{A}\,B\,Cin + A\,\bar{B}\,Cin + A\,B\,\overline{Cin} + A\,B\,Cin$

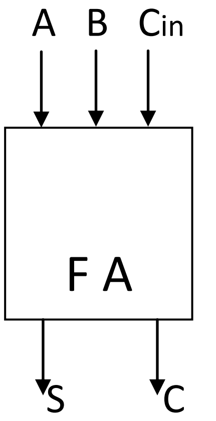
$(\bar{A}\,B + A\,\bar{B})\,C_{in} + A\,B\,(\overline{Cin}+Cin)$

$(A \oplus B)\,Cin + A\,B$

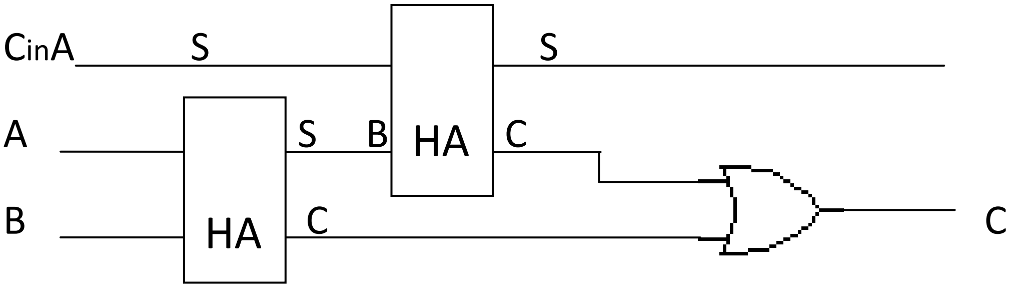
The logic circuit of full – adder is:



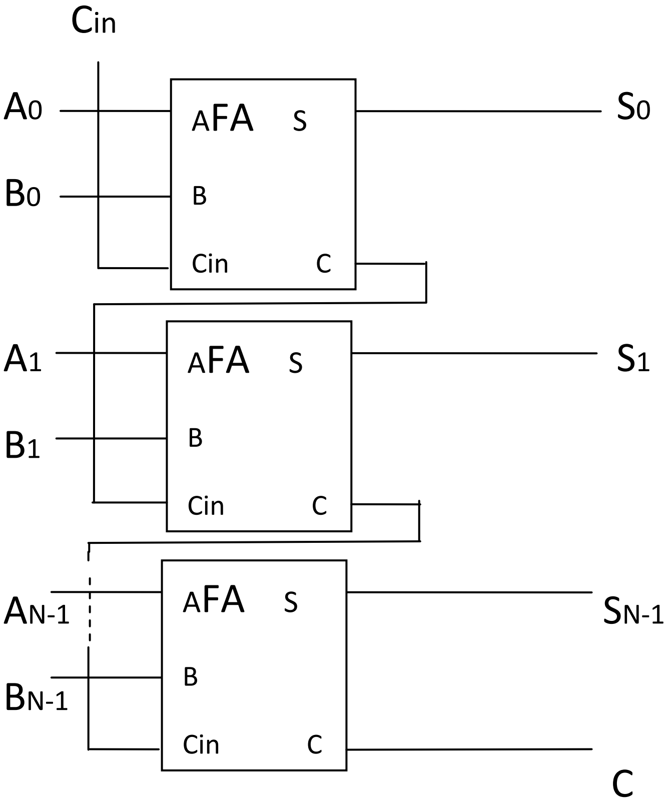
Block – diagram



From full – adder circuit, we see thay full – adder is built from two half – adders with OR gate.



Now we can build an N – bit adder from N full- adder s:



Subtractors

1. Half-subtractor circuit.

Subtractor of two binary numbers is doing by following method:

- Getting complement of second number (it become negative number)
- Adding first number to the complement.

Another way to implement subtractor is to use logic circuit directly.

First way :   7   7 = 111

  -3   3 = 011   2 complement = 101

$$\begin{array}{r} 7 \\ -3 \\ \hline 4 \end{array}$$

$$\begin{array}{r} 7 = 111 \\ -3 = 101\ + \\ \hline \end{array}$$

يهمل $\longrightarrow \boxed{1}\,100$

Second way:

$$\begin{array}{r} 7 = 111 \\ -3 = 011\ - \\ \hline 100 \end{array}$$

Half subtractor is combinational logic circuit gives an output as difference between two inputs (2-bits) and another output equal to 1 in case of borrow, – to do A- B, we have to check each A, B.

IF $A \geq B$ we shall get 3 probabilities.

$$\left.\begin{array}{l}0-0=0\\ 1-0=1\\ 1-1=0\end{array}\right\}\text{ and this bit we call “difference bit”}$$

If $A < B$, we have $0 – 1$. So we must borrow (1) from following bit. 1 will add 2 to A, as in decimal system.

So we shall have $2-1=1$

Half subtractor needs two outputs, first one is the difference and the other is the borrow (B_o)

Truth table, which shows the relationship between inputs and outputs, is:

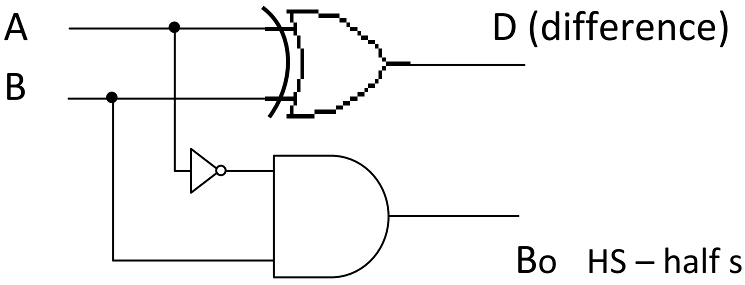
inputs		Outputs	
A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

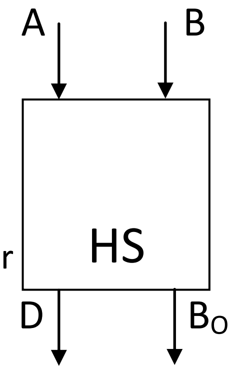
$D = \overline{A}\,B + A\,\overline{B}$

$B_o = \overline{A}\,B$

We see the equation of D is similar to equation of S in half-adder. So we can implement it through XOR gate.

B_o is different from C in HA. B_o can be implemented through AND gate with inputs $\overline{A}$, B.





HS – half subtractor

2. The full- subtractor circuit.

Full subtractor is a combinational logic circuit ,which do subtraction between (2-bits) two numbers, and may be (1) borrowing from followed number this circuit has 3 inputs and two outputs.

Input is A, B, B_{in}. Outputs – D,B_o

The truth table for this circuit is:

inputs			outputs	
A	B	B_{in}	D	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Output is A - B - B_{in}=

The logic equations for D, B_o are:

$D= \bar{A}\ \bar{B}\ B_{in}+\bar{A}\ B\ \bar{B}_{in}+A\ \bar{B}\ \bar{B}_{in}+A\ B\ B_{in}$

It is completelysimilar to S equal

In the HA

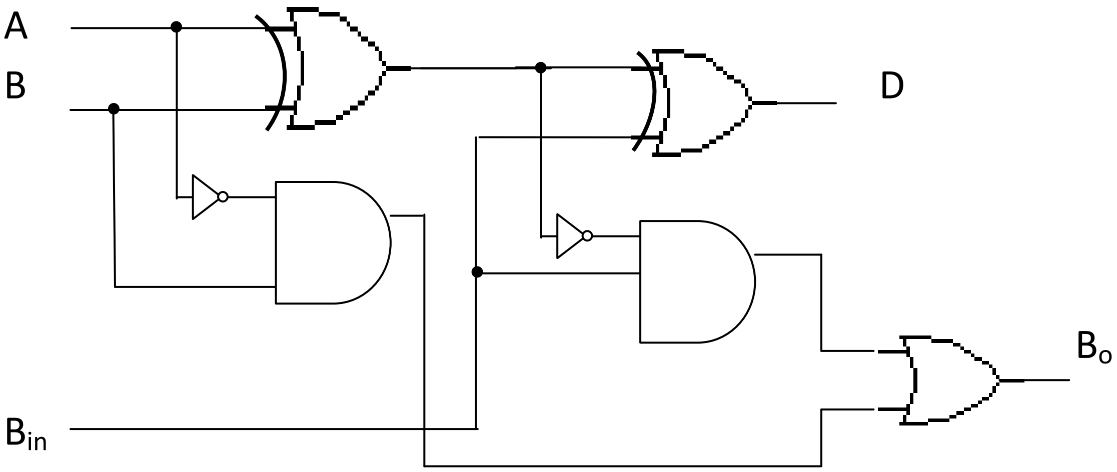
$D = (A \oplus B) \oplus B_{in} = A \oplus B \oplus B_{in}$

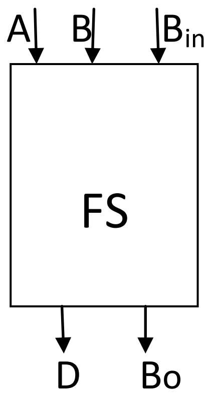
For B_o:

$B_o =\bar{A}\ \bar{B}\ B_{in}+\bar{A}\ B\ \bar{B}_{in}+\bar{A}\ B\ B_{in}+A\ B\ B_{in}$

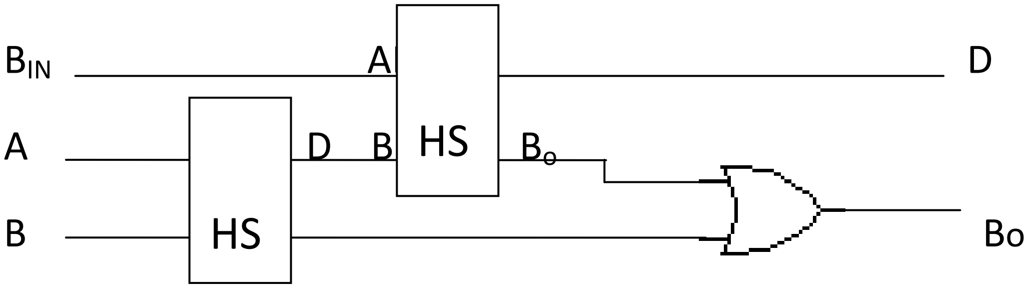
$B_o= B_{in}\ (\bar{A}\bar{B}+AB) + \bar{A}B\ (\ \bar{B}_{in}+B_{in}):\quad \bar{B}_{in}+B_{in} = 1$

$B_o= B_{in}\ (\overline{A \overset{\frown}{+} B})\ +\bar{A}B$





We see, the FS is abtained from 2 HS with OR gate.



Build an N-bit sub tractor from an N –bit adder using 2'S complement

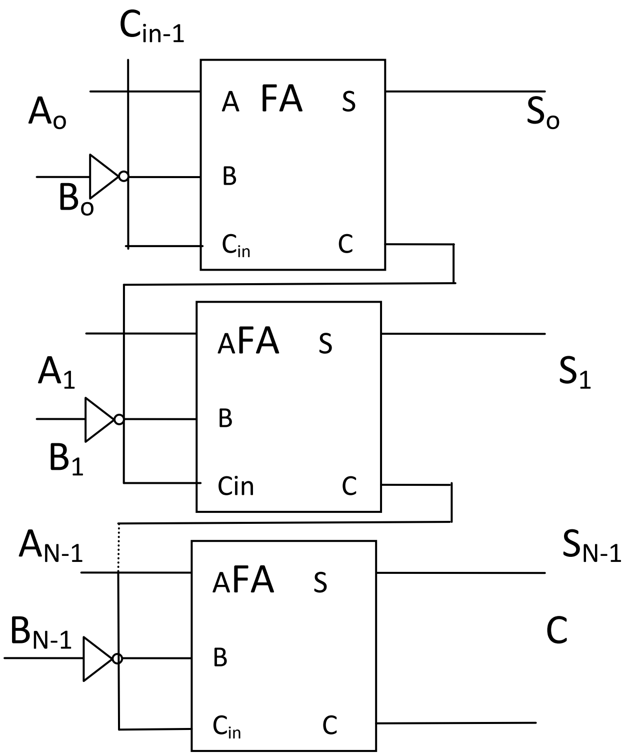
- Recall the 2'S complement.

  Transformation for a negative number:

  1.invert

  2. then add 1

  there we use C_{in} to add 1
- Therefore , S = A – B
- Note that this includes a sign bit (SN-1)



Exp.: two – bit adder

Problem: design a logic circuit to add two 2-bit binary numbers and output the resulting number:

$$\begin{array}{ccc} & A & B \\ + & y & Z \\ \hline S_3 & S_2 & S_1 \end{array}$$

Solution: four inputs are needed one for each bit that is added, three outputs are required for a sum bits and the carry out bit.

A	B	y	Z	S_3	S_2	S_1
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

	$\bar{y}\bar{Z}$	$\bar{y}Z$	yZ	$y\bar{Z}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	0	1	0
AB	0	1	1	1
$A\bar{B}$	0	0	1	1

$$S_3 = Ay + ABZ + ByZ$$

$$= BZ(A+y) + Ay$$

	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{A}\bar{B}$	0	0	1	1
$\bar{A}B$	0	1	0	1
AB	1	0	1	0
$A\bar{B}$	1	1	0	0

$$S_2 = \bar{A}\bar{B}y+\bar{A}y\bar{Z} +A\bar{y}\bar{Z} +A\bar{B}\bar{y} +\bar{A}B\bar{y}Z +AByZ$$

$$= (A \oplus y) \oplus (BZ)$$

	$\bar{y}\bar{Z}$	$\bar{y}Z$	yZ	$y\bar{Z}$
$\bar{A}\bar{B}$	0	1	1	0
$\bar{A}B$	1	0	0	1
AB	1	0	0	1
$A\bar{B}$	0	1	1	0

$$S_1 = B\bar{Z} + \bar{B}Z$$

$$= B \oplus Z$$

$$S_2=\bar{A}Y(\bar{B}+\bar{Z})+A\bar{Y}(\bar{B}+\bar{Z})+(\bar{A}\bar{Y}+AY)BZ$$

$$S_2=(\bar{A}Y+A\bar{Y})(\bar{B}+\bar{Z})+(\bar{A}\bar{Y}+AY)BZ$$

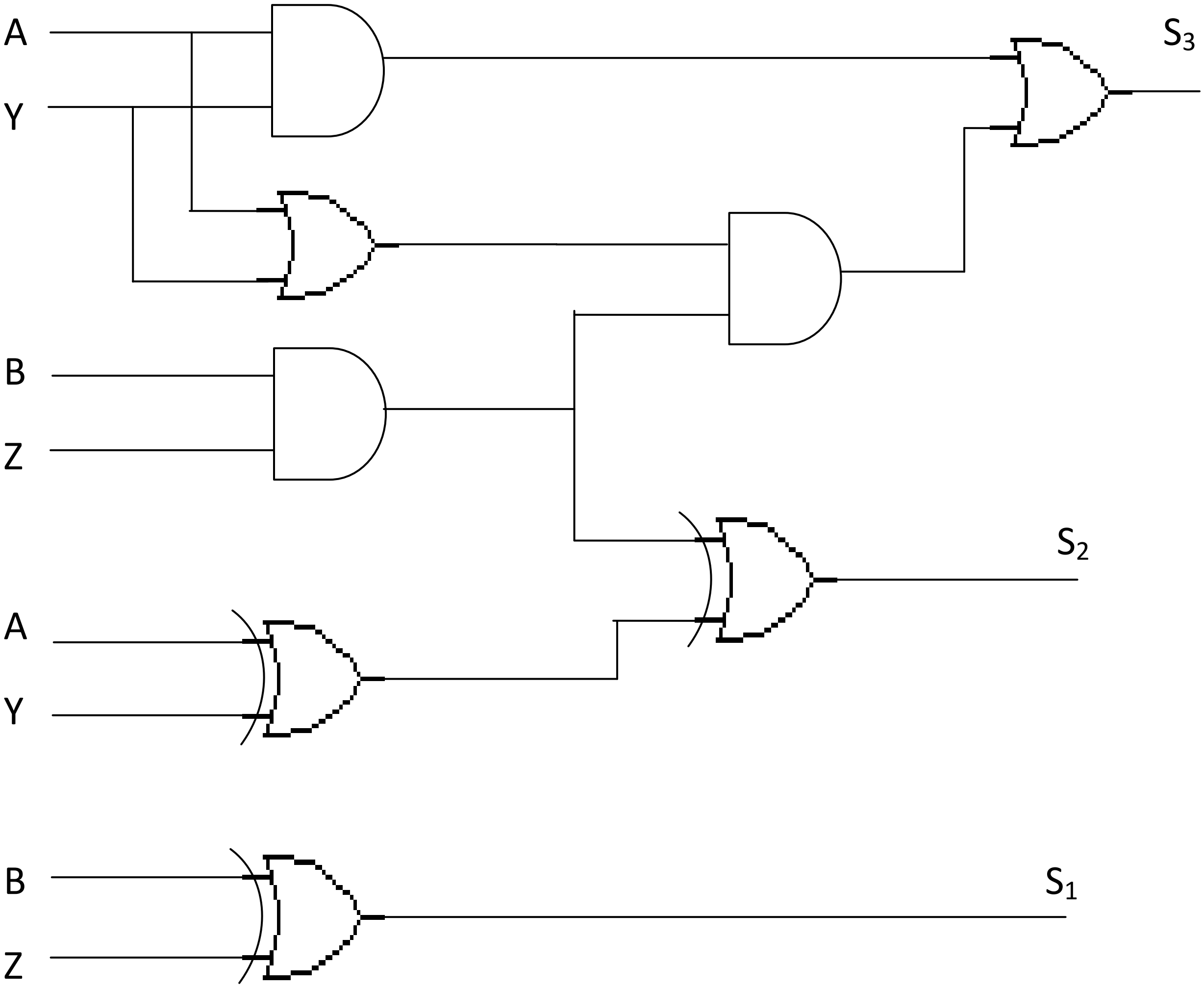
$$S_2=(A\oplus Y)(\bar{B}+\bar{Z})+\overline{(A\oplus Y)}BZ$$

$$S_2=(A\oplus Y)(\overline{BZ})+\overline{(A\oplus Y)}BZ$$

$$S_2=(A\oplus Y)\oplus BZ$$

S2 إيضاح

Logic functions and combinational logic



نـهـايـة

الفصل الأول

Comparators

Comparators are logic circuit that determine of input values are less than, greater than, or equal to each other. Comparators are used in numerous applications, many of which are arithmetic operations. They can be used to evaluate input values from many processes to determine if they are in an acceptable range; to verify relationship between values such as when one value must remain above or below the other value; and as magnitude detectors to indicate in valid of conditions. Comparators are one of the digital blocks of analog-in digital conversion. They can also be used in combination with arithmetic operation to signal overflow or other conditions that require corrective measures.

Comparators are relatively simple combinational logic circuit that can be specified in a truth table and designed from logic gates. The comparator examines two values and usually has up to three outputs.

Comparators operations

The process is simple since the bits can only take on the values of 0 and 1. The exclusive-NOR gate is the simplest comparator to indicate equivalency between 2- bits.

Comparator can be designed and built from logic gates by evaluating like bit positions, or by means of truth table.

1. Equal:

   To determine of the number are equal to each other, each equivalent bit weight must have the same value in its respective position.

   A” equal to”° B

A = B

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

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

To compare two 4-bit numbers is show below.

$A_4\,A_3\,A_2\,A_1\ :\ B_4\,B_3\,B_2\,B_1$

$A = \{\ A_4, A_3, A_2, A_1\ \}$

$B = \{\ B_4, B_3, B_2, B_1\ \}$

Exp. : 0 1 1 0 (4-bit)

$A = B : (\overline{A_4 \oplus B_4})(\overline{A_3 \oplus B_3})(\overline{A_2 \oplus B_2})(\overline{A_1 \oplus B_1})$

2. Greator than

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

A > B

$Y = A\,\bar{B}$

For 4-bits two parameters

$$A > B: \quad A_4\overline{B}_4 + (\overline{A_4 \oplus B_4})(A_3\overline{B}_3) + (\overline{A_4 \oplus B_4})(\overline{A_3 \oplus B_3})(A_2\overline{B}_2) + $$
$$+ (\overline{A_4 \oplus B_4})(\overline{A_3 \oplus B_3})(\overline{A_2 \oplus B_2})(A_1\overline{B}_1)$$

$$\left.\begin{matrix} A_4\,A_3\,A_2\,A_1 \\ B_4\,B_3\,B_2\,B_1 \end{matrix}\right\}$$

analys:

1. $A_1 > B_1$, $A_2 = B_2$, $A_3 = B_3$, $A_4 = B_4$
2. $A_2 > B_2$, $A_3 = B_3$, $A_4 = B_4$
3. $A_3 > B_3$, $A_4 = B_4$
4. $A_4 > B_4$

$$\begin{matrix} 0\,0\,0\,1 & \quad & 1\,1\,1\,1 & \quad & 0\,1\,1\,1 & \quad & 1\,0\,1\,0 \\ \underbrace{0\,0\,0\,0}_{1} & & \underbrace{1\,1\,0\,1}_{2} & & \underbrace{0\,0\,0\,1}_{3} & & \underbrace{0\,1\,1\,1}_{4} \end{matrix}$$

3. Less than

$A < B$

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

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

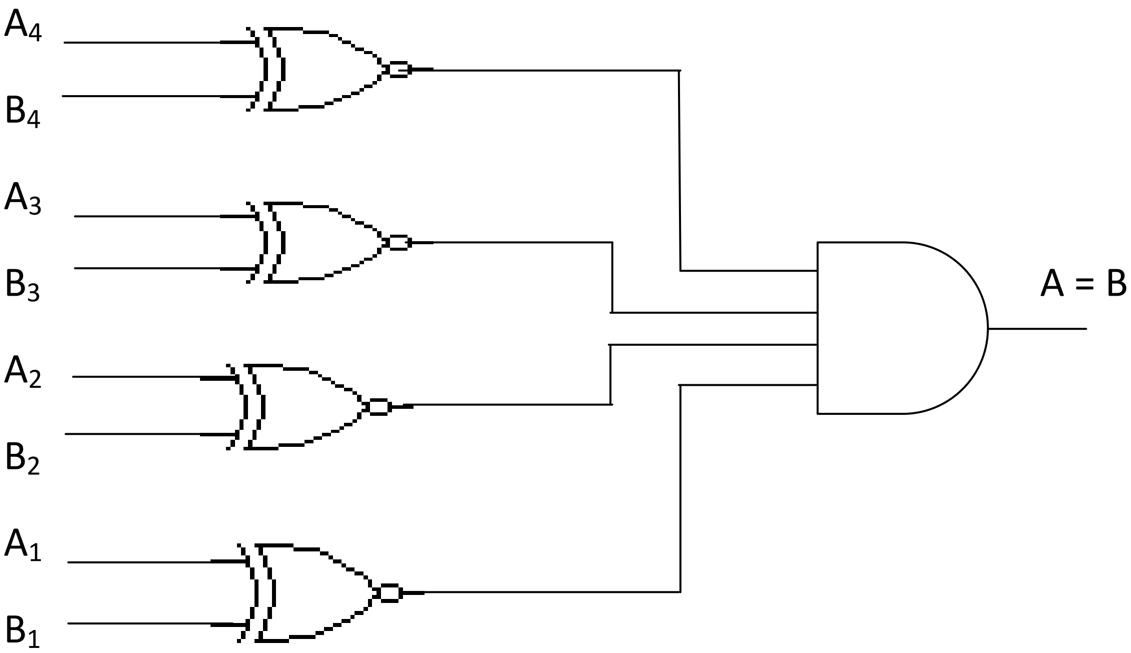
For two 4-bit inputs:

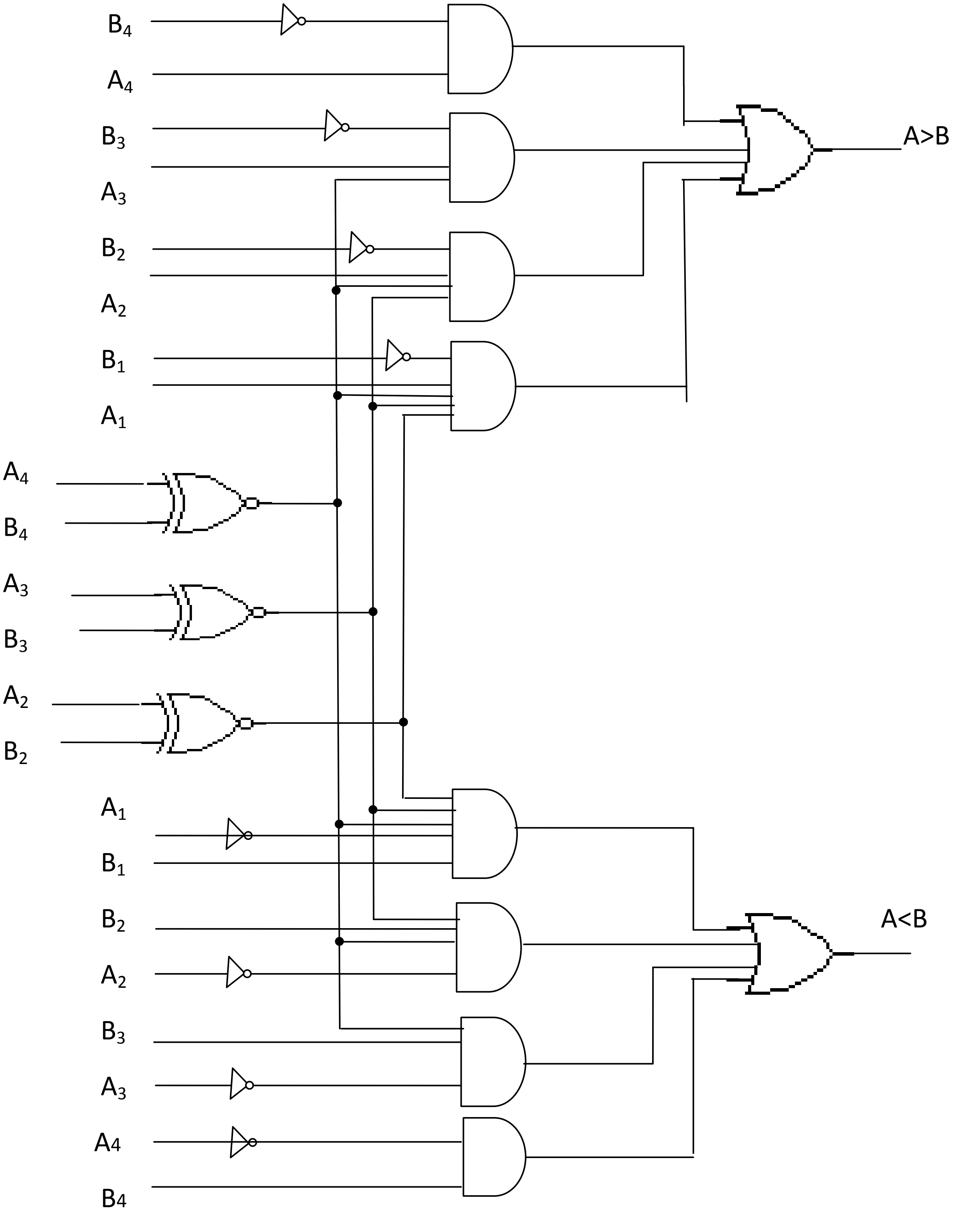
$$\left.\begin{matrix} A_4\,A_3\,A_2\,A_1 \\ B_4\,B_3\,B_2\,B_1 \end{matrix}\right\}$$

A < B :

$$= \overline{A}_4 B_4 + (\overline{A_4 \oplus B_4})(\overline{A}_3\,B_3) + (\overline{A_4 \oplus B_4})(\overline{A_3 \oplus B_3})(\overline{A}_2 B_2) +$$
$$+ (\overline{A_4 \oplus B_4})(\overline{A_3 \oplus B_3})(\overline{A_2 \oplus B_2})(\overline{A}_1 B_1)$$

Logic circuits for a 4-bit comparator that test for A “equal to “ B , A “grater than “B , and A “ less than “ B is shown bellow:


A_4
B_4
A_3
B_3
A_2
B_2
A_1
B_1
A = B


B_4
A_4
B_3
A_3
B_2
A_2
B_1
A_1
A>B
A_4
B_4
A_3
B_3
A_2
B_2
A_1
B_1
B_2
A_2
B_3
A_3
A4
B4
A<B

Multiplexers

Multiplexers accept date from one of many inputs and route the data to input. A multiplexer can be thought of an electronic switch. This simple function of selecting and routing data serves many puposes such as routing data, generating logic functions, transmitting data from several sources on one common transmission line, and using display multiplexing.

Function description.

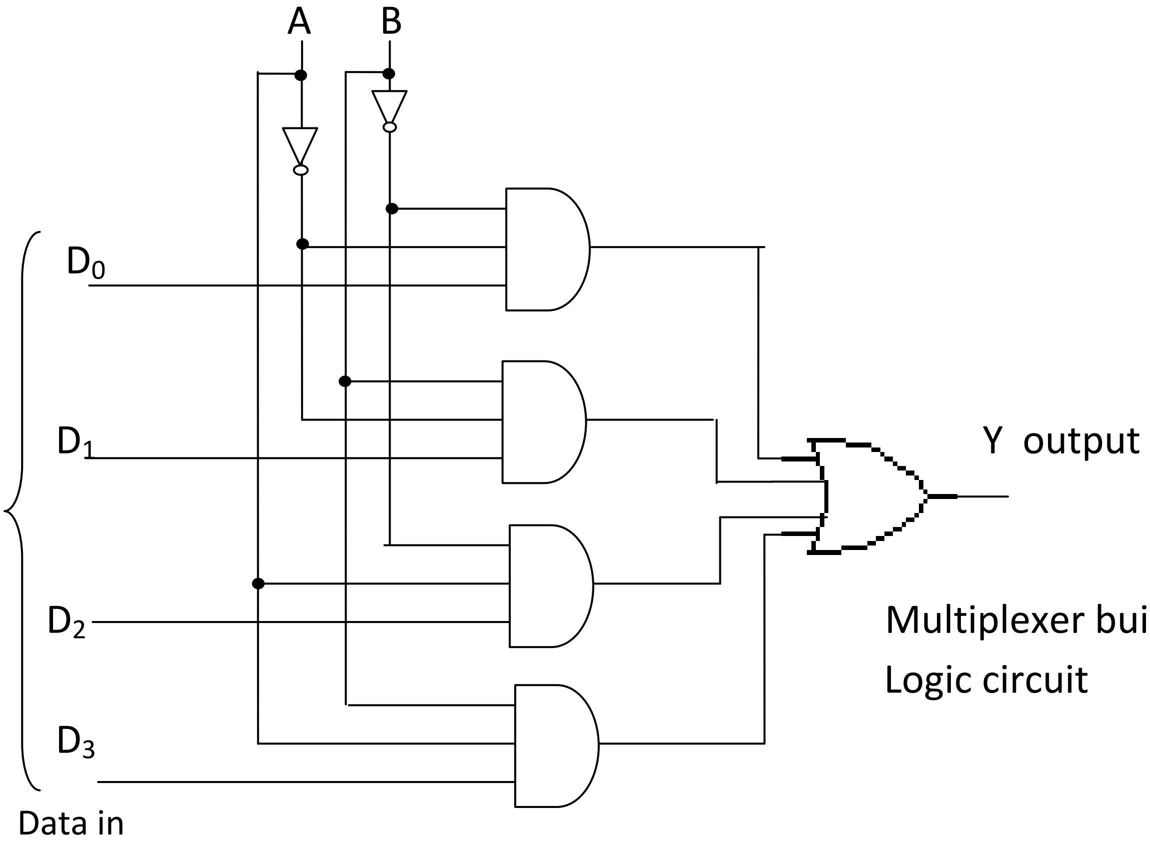
A multiplexer has data inputs, select inputs and an output. The data inputs are used to input the binary logic level, or data, to be switched to the output of the multiplexer. A multiplexer has several data inputs, but only one data input can be switched to the output at any one time. Some multiplexer invert the data during the switching process where as others leave the data unchanged. Other than the possible inversion, the data is not altered as it is routed to the output of the multiplexer.

The select inputs are the address input for the multiplexer. The address on the select inputs determines which data input is switched to the output. Each data input requires its own address, which is a unique binary combination that is specified on the select inputs. For instance data input 1, or D_1, as it after labeled has a 4-bits binary address of 0 0 0 1, D_{15} has a binary address of 1 1 1 1. The multiplexer is programmed through the select inputs to establish a connection between the output and addressed data input.

Gate equivalent circuits.

A multiplexer is a simple two- level logic circuit comprised of one OR level and one AND level , plus inverted and non inverted select inputs. The following circuit is an example of a 4:1 multiplexer (MUX) built out of AND

and OR gate. The logic equation of the output is read off the truth table in the sum of products (sop) from. The output equation has all input variable produce terms ANDed with the respective data input. The logic output equation and truth table description of the operation appear bellow. With an output equation of this form, multiplexer can generate any logic function.



Multiplexer built from Logic circuit

Select input output

A	B	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$Y = \bar{A}\bar{B}D_0 + \bar{A}BD_1 + A\bar{B}D_2 + ABD_3$$

multiplexer truth table

Logic function generator

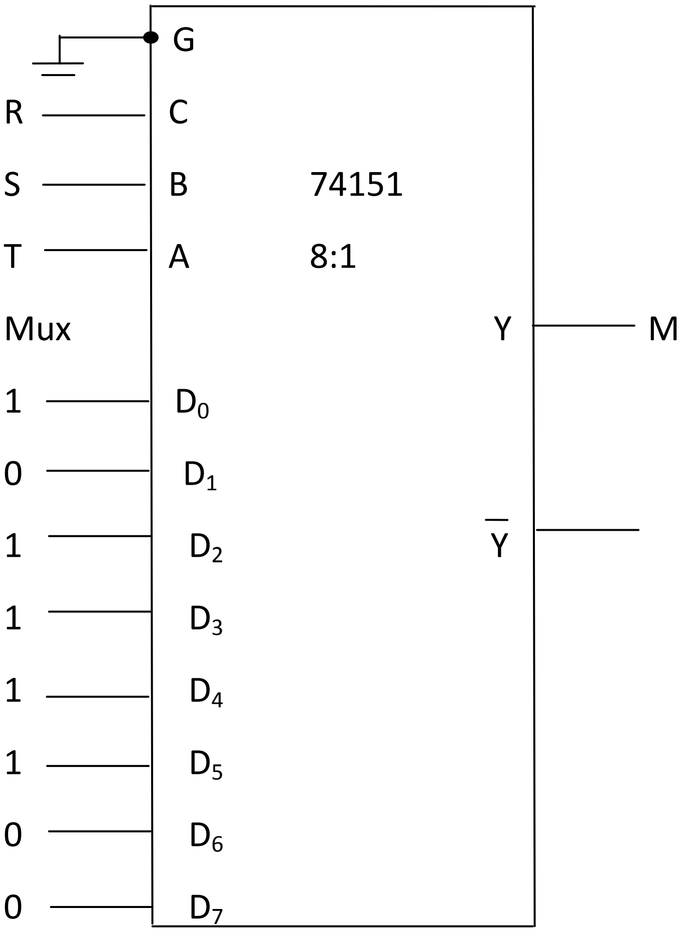
Exp.: implement the logic function specified in the truth table using 74151- 8:1 mux: 8:1 means 8 output probabilities ,so 3 inputs must be.

Solution: the logic function specified by truth table has three input variable and eight output states. The variable inputs R,S and T, are assigned to the select inputs of the multiplexer. The output states are assigned to the data input on the multiplexer. The address of the data input should be the R,S,T input variable combination requited for that output state. The multiplexer circuit is shown below:

R	S	T	M
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Select inputs			Mux outputs
C	B	A	Y
R	S	T	M
0	0	0	0 = D_0
0	0	1	1 = D_1
0	1	0	1 = D_2
0	1	1	1 = D_3
1	0	0	1 = D_4
1	0	1	1 = D_5
1	1	0	0 = D_6
1	1	1	0 = D_7

Output Mux datainputs



Demultiplexer:

A Demultiplexer, or DMUX, performs the opposite operation from that of a multiplexer.

Code conversion circuits

Code conversion circuits is a general category of logic circuit used to perform the 0peration of converting from one code to another. Code conversion is performed be decoder, encoder, and code convertor circuits.

Code conversion operations, which occure extensively in digital applications, are required when signals are entered into a system or displayed as
the output from a system.

Decoders , encoders and code converter circuits are manufactured as medium scale integration (MSI) integrated circuits (IC$_s$).

Decoders convert a binary number or binary address to a single output bit.

Encoders convert a single input bit to binary number.

Code converters convert from one binary code to another. Such as converting from binary coded decimal (BCD) to binary.

Decoders

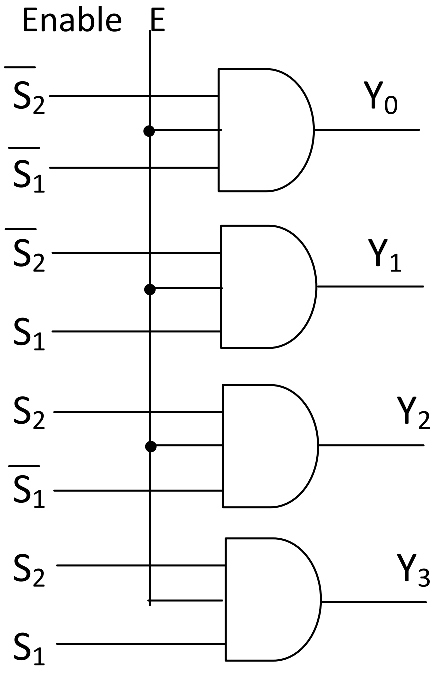
Active – HIGH output

Address inputs		decoded outputs				
S_2	S_1	Y_0	Y_1	Y_2	Y_3	
0	0	1	0	0	0	$Y_0 = \overline{S_2}\,\overline{S_1}$
0	1	0	1	0	0	$Y_1 = \overline{S_2}\,S_1$
1	0	0	0	1	0	$Y_2 = S_2\,\overline{S_1}$
1	1	0	0	0	1	$Y_3 = S_2\,S_1$

Active – LOW output

Address inputs		decoded outputs				
S_2	S_1	Y_0	Y_1	Y_2	Y_3	
0	0	0	1	1	1	$\overline{Y_0} = \overline{S_2}\,\overline{S_1}$
0	1	1	0	1	1	$\overline{Y_1} = \overline{S_2}\,S_1$
1	0	1	1	0	1	$\overline{Y_2} = S_2\,\overline{S_1}$
1	1	1	1	1	0	$\overline{Y_3} = S_2\,S_1$

Active – HIGH decoder

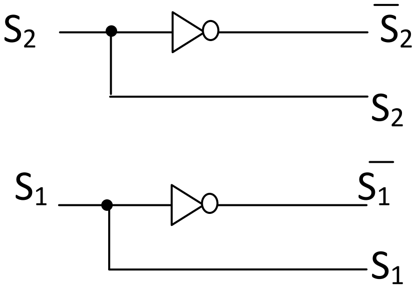


$Y_0 = E\overline{S_2}\,\overline{S_1}$

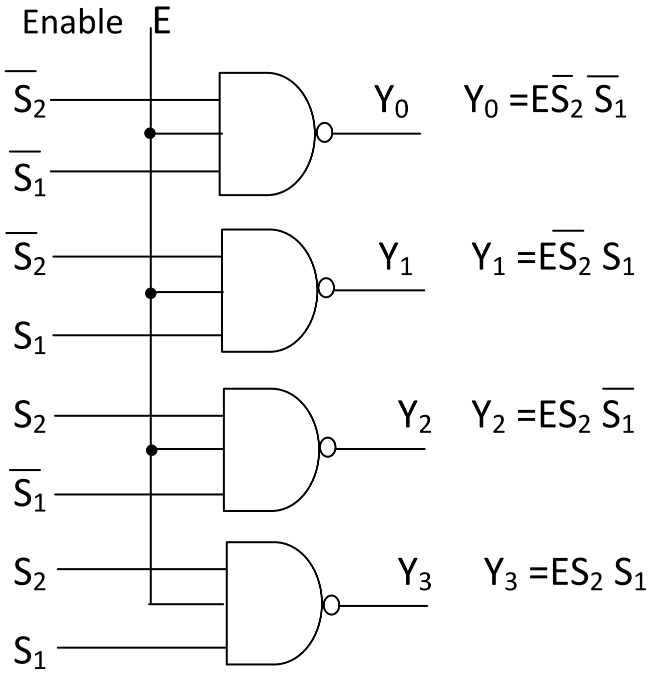
$Y_1 = E\overline{S_2}\,S_1$

$Y_2 = ES_2\,\overline{S_1}$

$Y_3 = ES_2\,S_1$



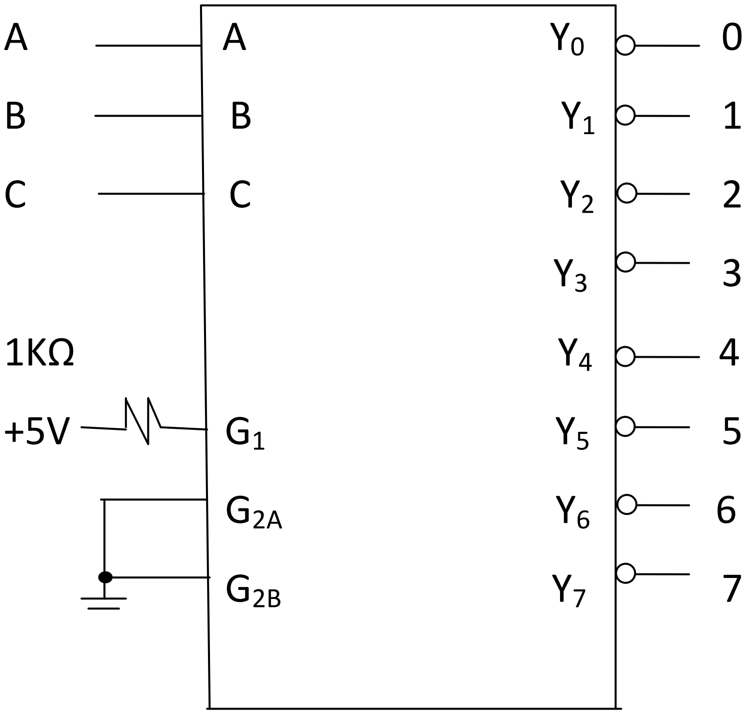
Active – LOW decoder



Exp: Binary to octal conversion determine the binary input required to produce the specified random output sequence. Using the 74138 for code conversion. outputs sequence: 7,3,0,2,1,5,4,6

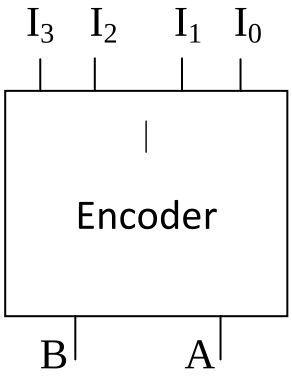
Solution: the input combination must be the binary equivalent of the observed output since the 74138 is designed to perform binary to octal code conversion. The required input sequence is as follows:

Input address			Output activated
C	B	A	
1	1	1	Y_7
0	1	1	Y_3
0	0	0	Y_0
0	1	0	Y_2
0	0	1	Y_1
1	0	1	Y_5
1	0	0	Y_4
1	1	0	Y_6



Encoders :

Encoder circuit perform the opposite function of decoder circuits ,only one input value is active at any one time, and its activation produces a specific code combination on the outputs. Encoders are used extensively in keyboard applications ,where activation at a single key must produce a unique binary code to represent its character or value.



Active HIGH inputs				Encoded outputs	
I_0	I_1	I_2	I_3	B	A
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

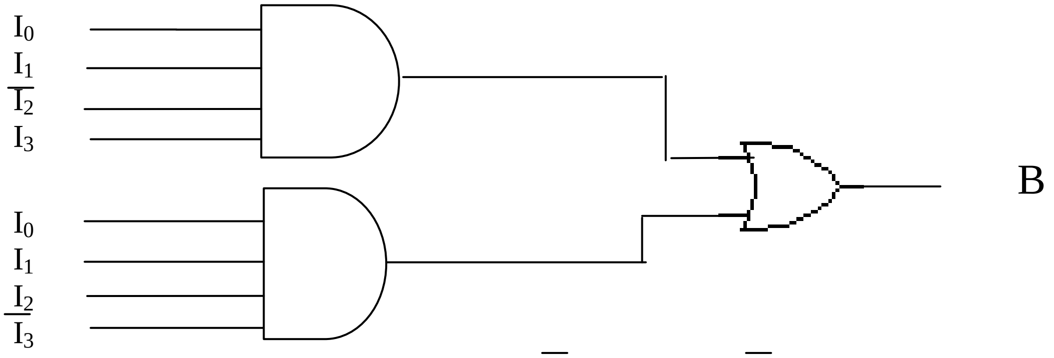
$$B = \overline{I}_0\,\overline{I}_1\,I_2\overline{I}_3 + \overline{I}_0\,\overline{I}_1\,\overline{I}_2 I_3$$

$$A = \overline{I}_0\,I_1\,\overline{I}_2\overline{I}_3 + \overline{I}_0\,\overline{I}_1\,\overline{I}_2 I_3$$

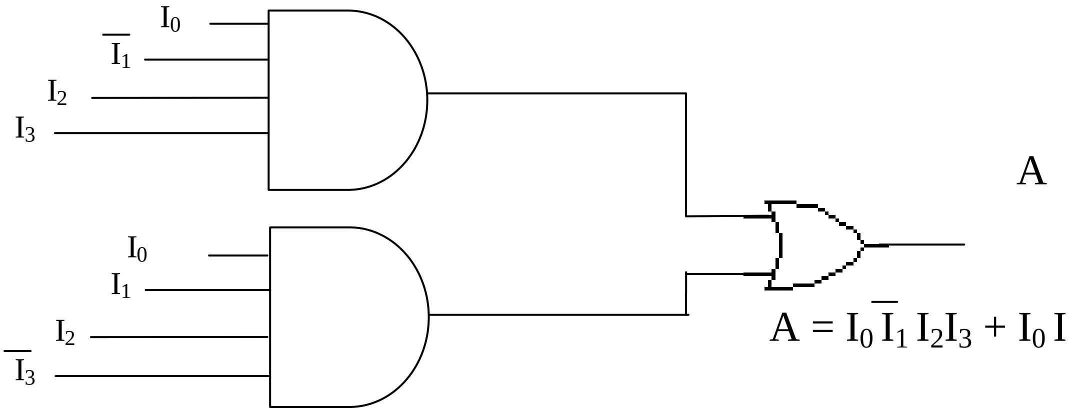
Active LOW inputs				Encoded outputs	
I_0	I_1	I_2	I_3	B	A
0	1	1	1	0	0
1	0	1	1	0	1
1	1	0	1	1	0
1	1	1	0	1	1

$$B = I_0\,I_1\,\overline{I}_2 I_3 + I_0\,I_1\,I_2\overline{I}_3$$

$$A = \overline{I}_0\,I_1\,\overline{I}_2\overline{I}_3 + \overline{I}_0\,\overline{I}_1\,\overline{I}_2 I_3$$



$$B = I_0\,I_1\,\overline{I_2}I_3 + I_0\,I_1\,I_2\overline{I_3}$$



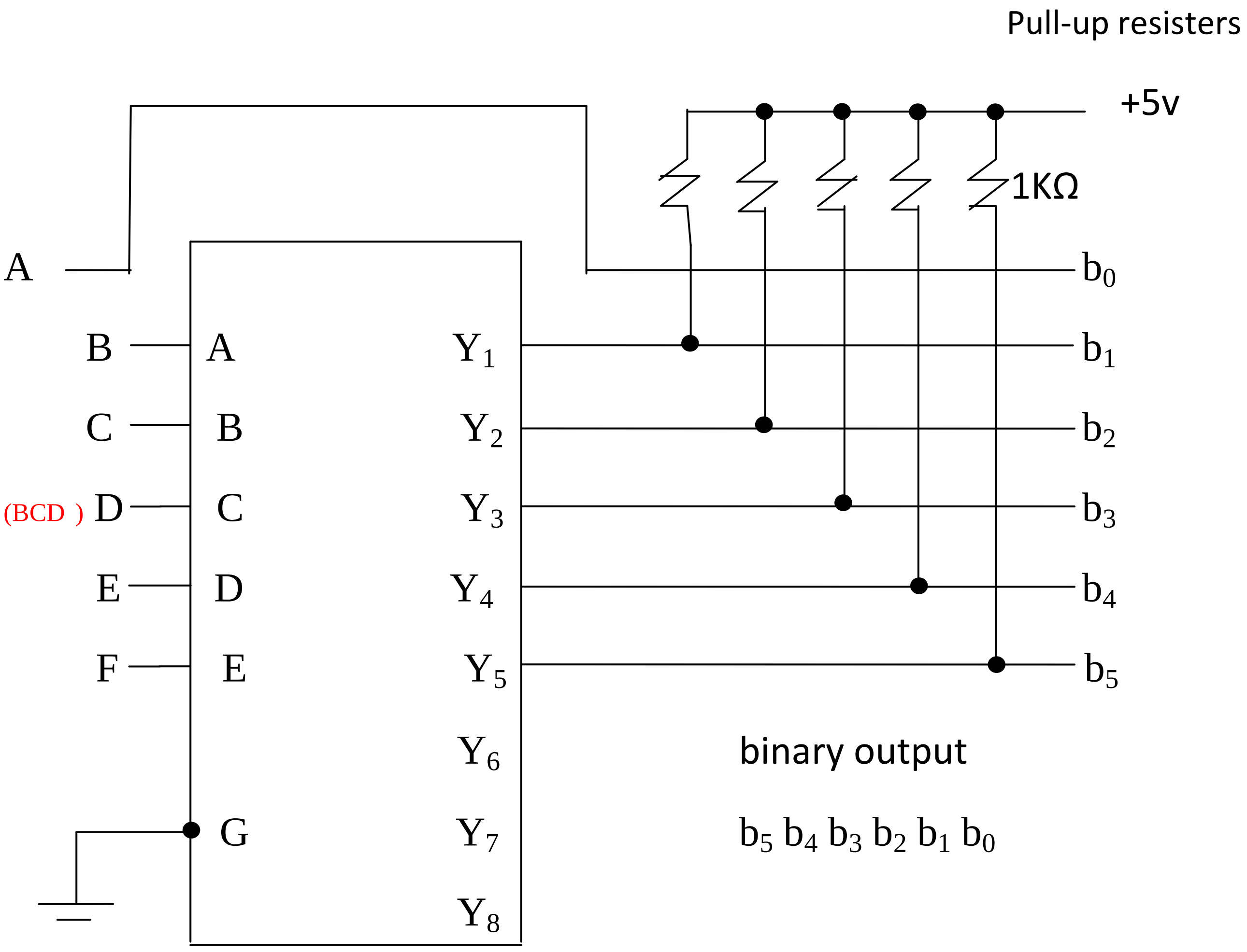
$$A = I_0\,\overline{I_1}\,I_2I_3 + I_0\,I_1\,I_2\overline{I_3}$$

Code converters

Code converter circuits are specially programmed logic circuits designed to perform a specific code conversion. Code converters are used to convert from one number system or binary code to another code conversion circuits can be used to perform the operations to convert from decimal to binary, from 8421 BCD to gray code or excess – 3,and even from BCD to the nine's or ten's complement from of the number.

Binary to BCD conversion.

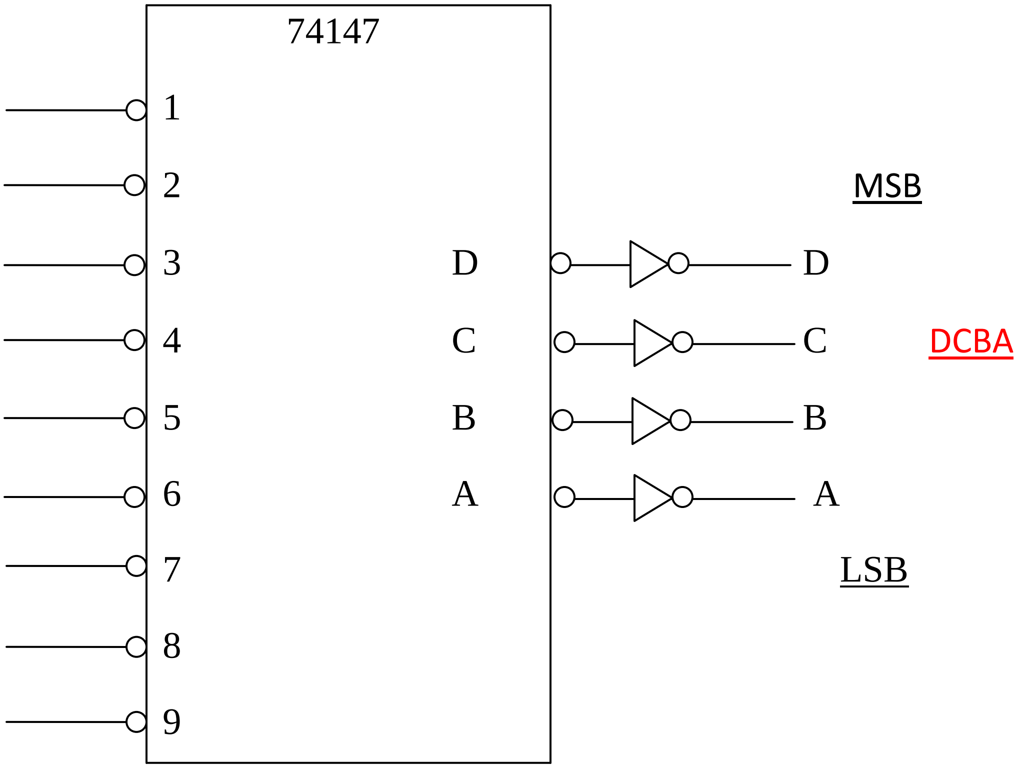
Binary to BCD conversion is complicated for logic gate implementation when binary values requiring over 5 bits must be converted to BCD. The 74185 is a read only memory (ROM) device designed to convert 6 bit binary values to 6-bit BCD values. A single 74185 IC can convert binary values of 0 through 63 to their BCD representation. Lager numbers are converted to BCD by cascading multiple 71485 IC_5



Exp.: decimal to binary encoding

Problem: Use the 74147 encoding and any additional logic gates to design a decimal to binary keypad encoding that has a priority output signal indicating that a keypad is pressed. The 0 keypad must activate the priority output signal, as well as the keypads for 1 through 9.

keypad										Binary output			
0	1	2	3	4	5	6	7	8	9	D	C	B	A
0	1	1	1	1	1	1	1	1	1	0	0	0	0
*	0	1	1	1	1	1	1	1	1	0	0	0	1
*	*	0	1	1	1	1	1	1	1	0	0	1	0
*	*	*	0	1	1	1	1	1	1	0	0	1	1
*	*	*	*	0	1	1	1	1	1	0	1	0	0
*	*	*	*	*	0	1	1	1	1	0	1	0	1
*	*	*	*	*	*	0	1	1	1	0	1	1	0
*	*	*	*	*	*	*	0	1	1	0	1	1	1
*	*	*	*	*	*	*	*	0	1	1	0	0	0
*	*	*	*	*	*	*	*	*	0	1	0	0	1



Sequential logic circuit

Logic circuits divide into two types. First type is called “combinational logic circuit” in which the output depends at any time on input at that time. We studied similar circuit in previous lessons. Second type is called “sequential logic circuit " this type of logic circuit has" memory” in which the output of the circuit depends at any time on current input and previous output of the circuit.

In the combinational circuits the building unit (or item) is the logic gates, while in the sequential logic circuit the building item will be flip-flop circuit. Flip-flop is a digital logic circuit, has main job storing information in fact of one bit information, one or zero.

Flip-flop may be find in one of two stable cases. One of them is the binary number (1) or logic (1). The second implements binary number (0) or logic(0). If flip-flop was put is one of two stable cases, it will be at that case while it is under power or will change this case after application of suitable logic input levels. That flip-flops are called " bistable multivibrator” flip-flops can be built from NAND or NOR gates or buy them as complete digital integrated circuits (IC_S). we can make from flip-flops logic circuits " timers" ,"counters” and” shift registers” and others.

Bistable logic circuit

Latches

Latch: is a basic digital storage device that is capable of storing one bit of information. The storage feature of the latch is very useful in digital circuit. The primary purpose of a latch to temporarily store a logic level. Latches are used to store small a amounts of data temporarily as compared to semiconductor memories which are used for long term storage and storage of large amount of data.

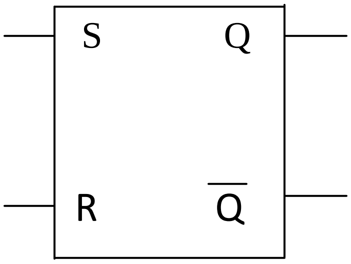
S-R and $\overline{S}$-$\overline{R}$ latch

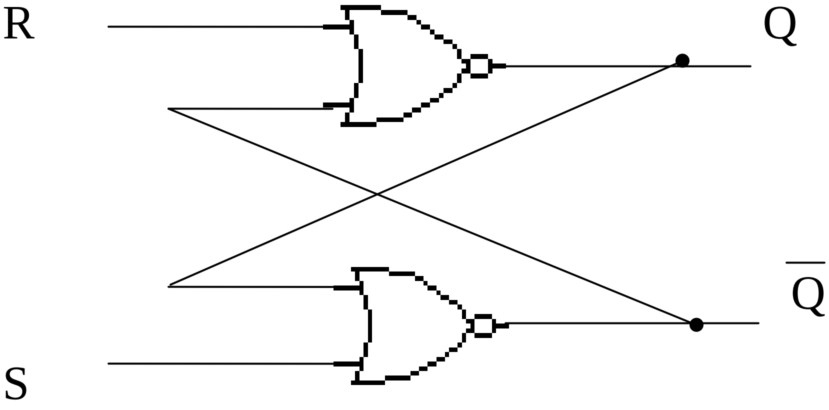
S-R and $\overline{S}$-$\overline{R}$ latches are bistable devices each of which is capable of storing 1 bit of data. These types of latches can be built with two basic logic gates utilizing feedback.

S-R latch

Another common type of latch is the S-R or set-reset latch. The S-R latch has 2 inputs S and R ,one true output Q and one complementary output $\overline{Q}$.
The S-R latch is easily constructed by using two NOR gates, with the $\overline{Q}$ output NORed with the R input, and the Q output NORed with the S input. Thus, the output is fed back to the input, which is known electronically as feedback. Also note that the output of one of the NOR gates is connected to the input of the other NOR gate.

S	R	Q	$\overline{Q}$	
0	0	Q_0	$\overline{Q}_0$	Hdd (no change)
0	1	0	1	Reset
1	0	1	0	Set
1	1	0*	0*	Invalid condition



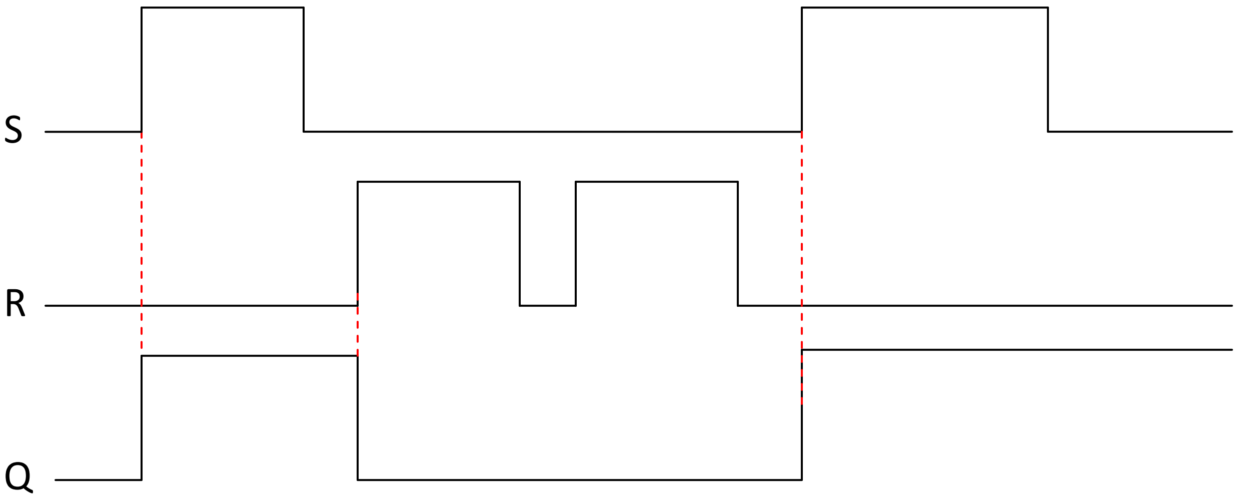


The truth table implicates that the S-R latch can be set (Q =1) or reset (Q = 0)

Illustration the primary steps to undertake when analyzing sequential logic devices and circuits.

Present input		present state	Next state	State comment
S	R	Q_n	Q_{n+1}	
0	0	0	0	hold
0	0	1	1	hold
0	1	0	0	Reset
0	1	1	0	Reset
1	0	0	1	Set
1	0	1	1	Set
1	1	0	nonstable	invalid
1	1	1	nonstable	Invalid

S-R latch operation

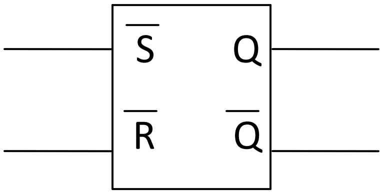


$\overline{S}$-$\overline{R}$ latch

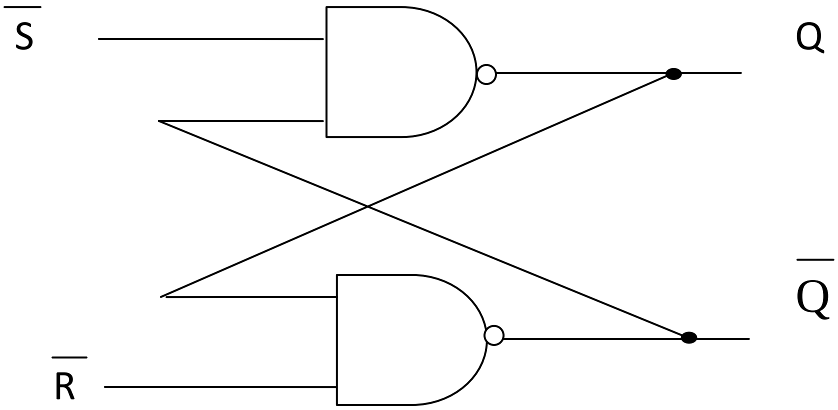
Another type of set-reset latch is the $\overline{S}$-$\overline{R}$ latch. This latch has the same possible output conditions as the S-R latch ,but inputs opposite to those of S-R latch are used to produce the latch states. These, this latch is other referred as an S-R latch with active LOW input.

The $\overline{S}$-$\overline{R}$ latch can be built from two cross- coupled NAND gates. Bellow the $\overline{S}$-$\overline{R}$ latch is shown as block diagram truth table and cress- coupled NAND circuit used to construct the latch.

$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	State comment
1	1	Q_0	$\overline{Q_0}$	Hold(no change)
1	0	0	1	Reset
0	1	1	0	Set
0	0	1*	1*	Invalid



The outputs for $\overline{S}$= 0, $\overline{R}$=0 input condition will both be HIGH or invalid condition.



74279 IC

$\overline{S}$-$\overline{R}$ latch

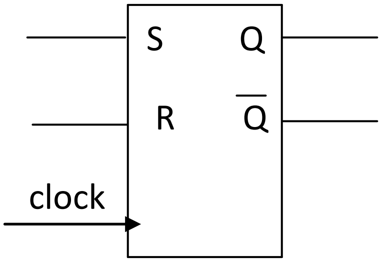
In addition to the change in gate type between the S-R and $\overline{S}$ - $\overline{R}$ cross- coupled latch circuits, it's important to note the input and output labeling. The S-R inputs and the Q and $\overline{Q}$ outputs are not identical for two circuits.

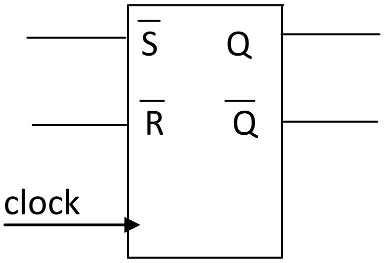
Exp.: problem: an $\overline{S}$-$\overline{R}$ latch has a HIGH input in to $\overline{S}$ and LOW input into $\overline{R}$. what is the output " Q "?

Solution: referring to the truth table , Q=0 ,reset position.

Clocked S-R Flip-Flop

An S-R flip-flop also as set-reset flip-flop, has two state control inputs S and R on clock input a truth output Q and a complement output $\overline{Q}$. If the S-R flip-flop has active- LOW inputs it's known as an $\overline{S}$-$\overline{R}$ flip-flop. The symbols and truth table for S-R and $\overline{S}$-$\overline{R}$ flip-flops are shown bellow.





S	R	clock	Q	operation
0	0	x	Q_0	Hold
0	1	↑	0	Clear
1	0	↑	1	Set
1	1	↑	?	unstable

$\overline{S}$	$\overline{R}$	clock	Q	operation
0	0	↑	?	unstable
0	1	↑	1	Set
1	0	↑	0	Clear
1	1	x	Q_0	Hold

Q_0 = initial state

↑ = leading edge-triggered clock

? = indeterminate

X = don't care

The hold operation for flip-flop means that whatever the output of the flip-flop is in the present state, when the clock triggers the flip-flop next state output will be the same. This is no change in the output for this input condition when the device is triggered.

The clear operation, also known as the reset, means that the output of the flip-flop will change to 0 -be cleared when the clock triggers. The set operation for a flip-flop indicates that the output will change to 1- be set- when the clock triggers.

In addition to the hold ,clear and set operations, the S-R flip-flops have an unstable or invalid state. The invalid state results in an invalid flip-flop operation if both S-R inputs are HIGH. Thus the invalid condition must be avoided by the logic designer when using S-R flip-flops.

Flip-plop triggering

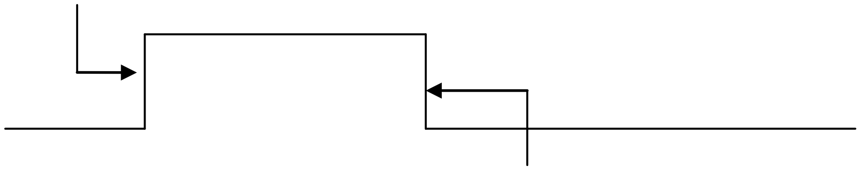
A flip-flop changes it's output state based on its data inputs and the transition of the clock input. Devices that changes state based on the transition of on input timing wave from are known as edge- triggered devices.

Types of edge- triggering

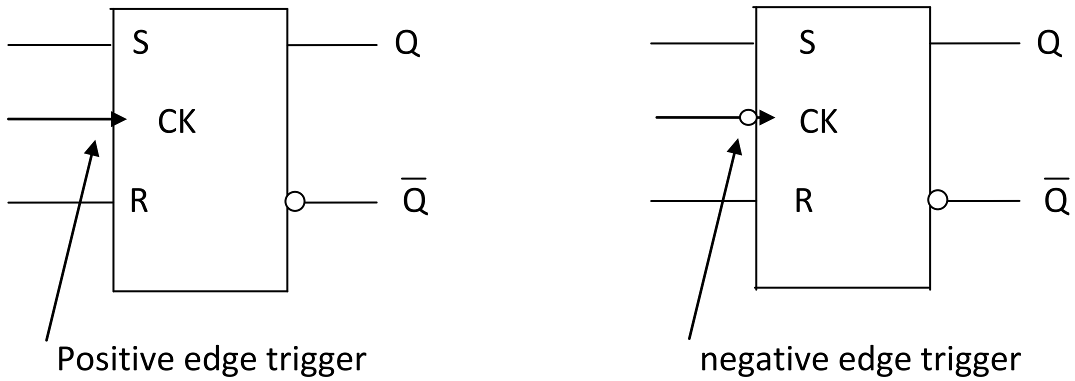
Synchronous digital devices rely on a timing input. The clock, to control when the device recognizes the signals placed on the data inputs and responds accordingly. One way of charactering flip-flop is according to the type of edge-triggering employed to control the function of flip-flop. The type of edge- triggering is denoted on the flip-flop symbol,

As shown for S-R flip-flops (discussed before) bellow.

Leading edge- triggered or positive edge triggered from logic LOW –to logic HIGH

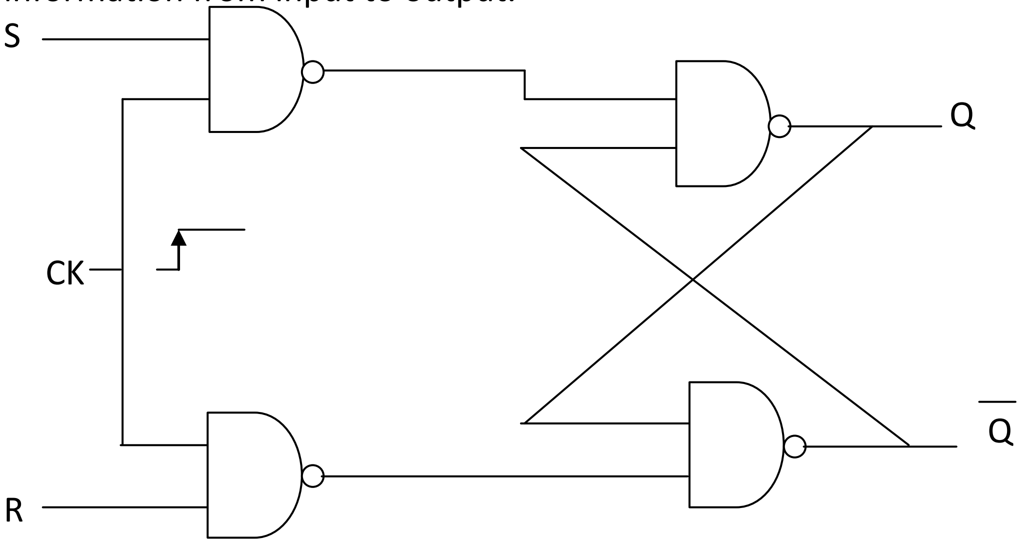


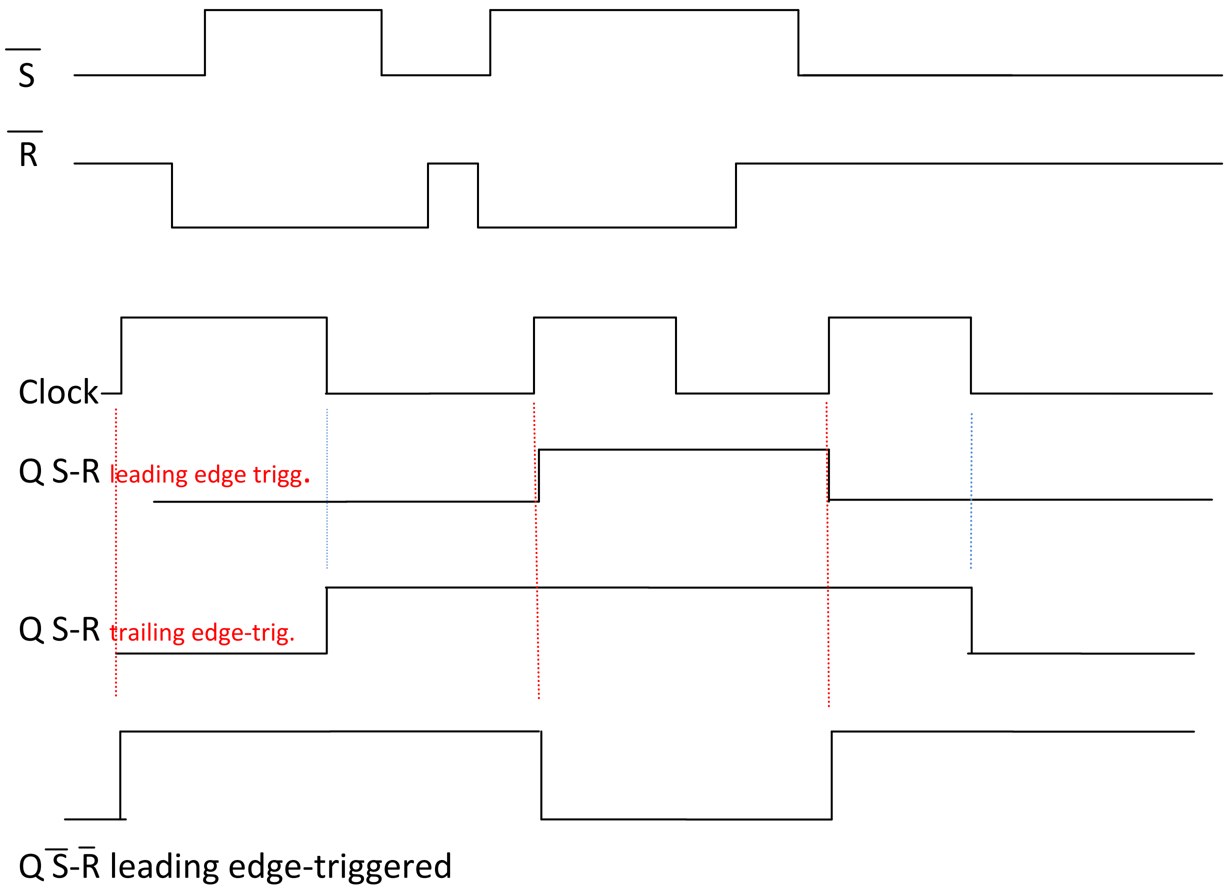
Trailing edge-triggered or negative edge-triggered from logic HIGH to logic LOW



Bellow sequential S-R flip-flop by using NAND gates, two NAND gate have been added to the latch main circuit for sequential purposes.

In formation transition, which are on the inputs S-R to the output Q when clock pulls will be at the positive edge-trigged ,which is worked as permission pulse to transit information from input to output.·




S̄
R̄
Clock
Q S-R leading edge trigg.
Q S-R trailing edge-trig.
Q S̄-R̄ leading edge-triggered

Flip – Flops

Flip- flops are bistable storage devices. Each flip-flop is capable of storing 1 bit of data. Flip-flops differ from latches in the method of triggering employed in the operation of the device. Several deferent types of triggering are used for the various types of flip-flop ICs available. Flip-flops have timing dependent synchronous inputs and timing independent asynchronous input.

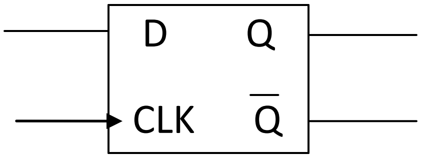
D flip-flop

D flip-flop are widely used in digital logic designs. A D flip-flop steers one bit of information clocked into the flip-flop.

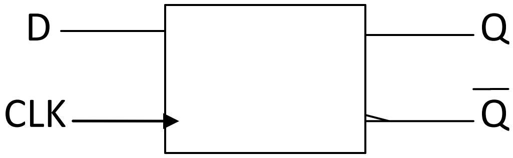
D flip-flop operation:

A D flip-flop, also known as a delay or data flip-flop has one data input D, a clock input and a true output, Q some D F-F also have a complement output $\overline{Q}$. The standard and IEEE symbol for a D F-F are shown below:

IEEE = institute of electrical and electronic engineering .



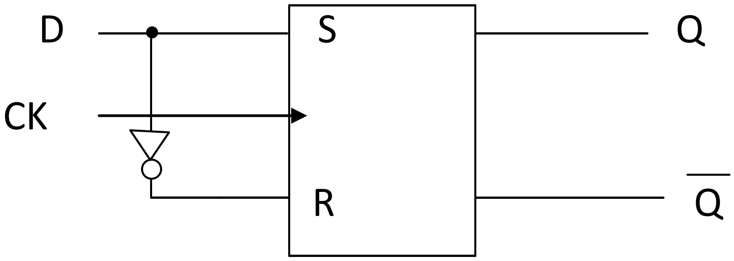
Standard symbol



IEEE symbol

D flip-flops have the ability to store 1 bit of data. The data bit that is placed on the D input is stored in the flip-flop when the clock trigger occurs. The data stored in logic level on the D input changes, the flip-flop value will change to the logic level on the input D at the clock trigger.

D flip-flop can be gotten from S-R F-F after adding NOT gate to the S-R flip-flop circuit.



Truth table:

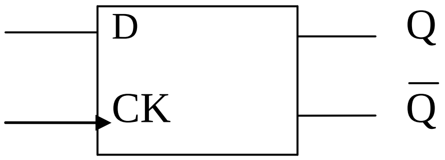
input				outputs		
D	S	CK	R	Q	$\overline{Q}$	
1	1	↑	0	1	0	set
0	0	↑	1	0	1	reset

$$\text{if input}\quad \left.\begin{array}{ll} D = 1 & \quad Q = 1 \\ D = 0 & \quad Q = 0 \end{array}\right\} \text{ when clock pulse comes}$$

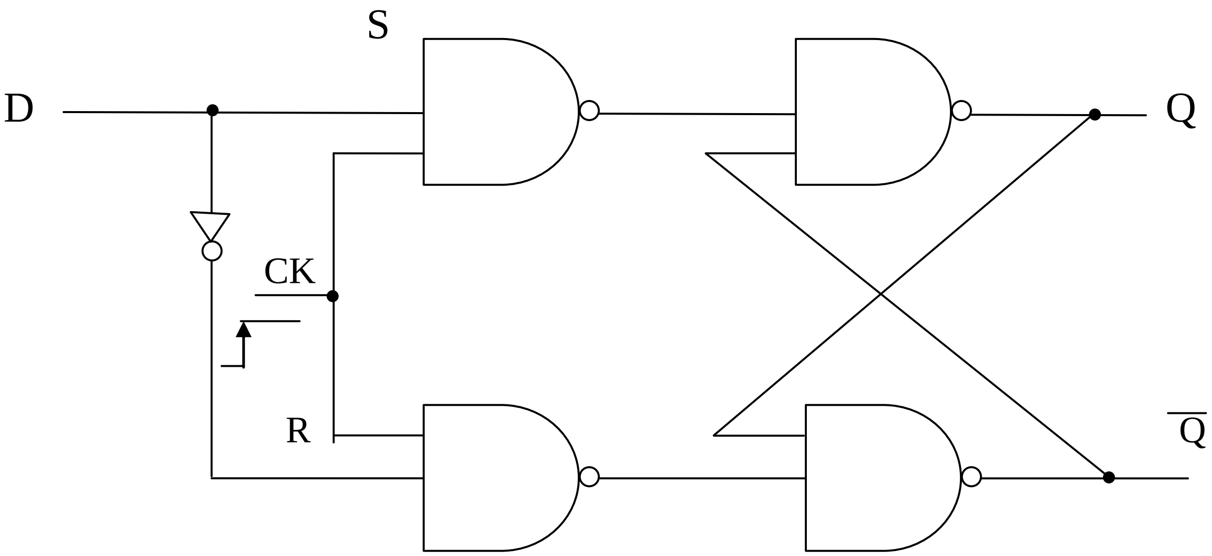
if Q = 1 , we say D flip-flop stores (1)

if Q = 0 , D flip-flop stores (0)

D flip-flop logic circuit using NAND gates

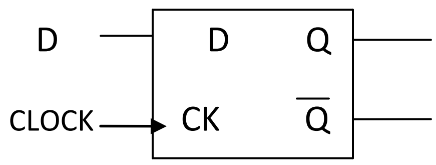


Logic symbol

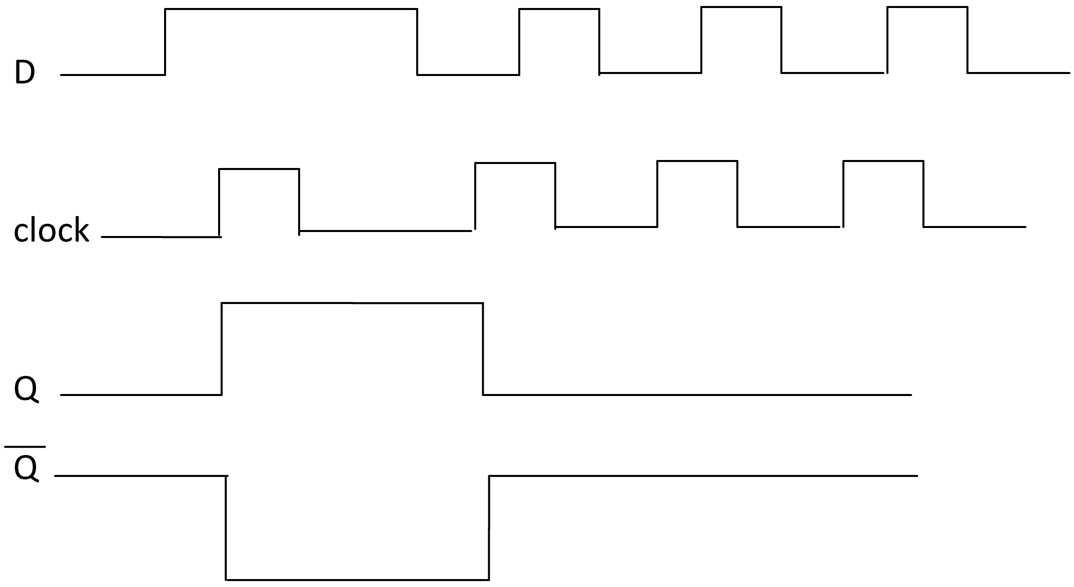


Exp.: Bellow logic symbol of D flip-flop, which operates at leading edge-triggered. Note that the only time the output Q can change state is on the leading edge of the clock pulse.

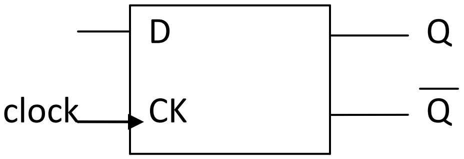
Most of the standard D flip-flops are leading edge- triggered.

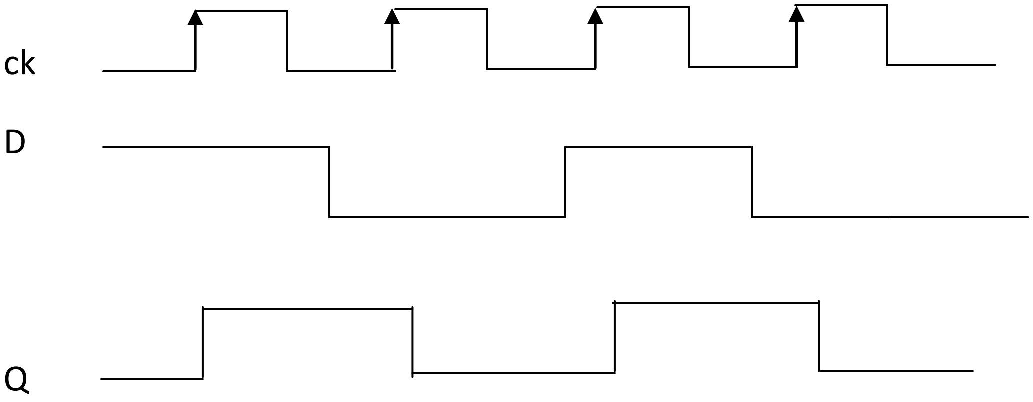


Q (0) = initial flip-flop state = 0

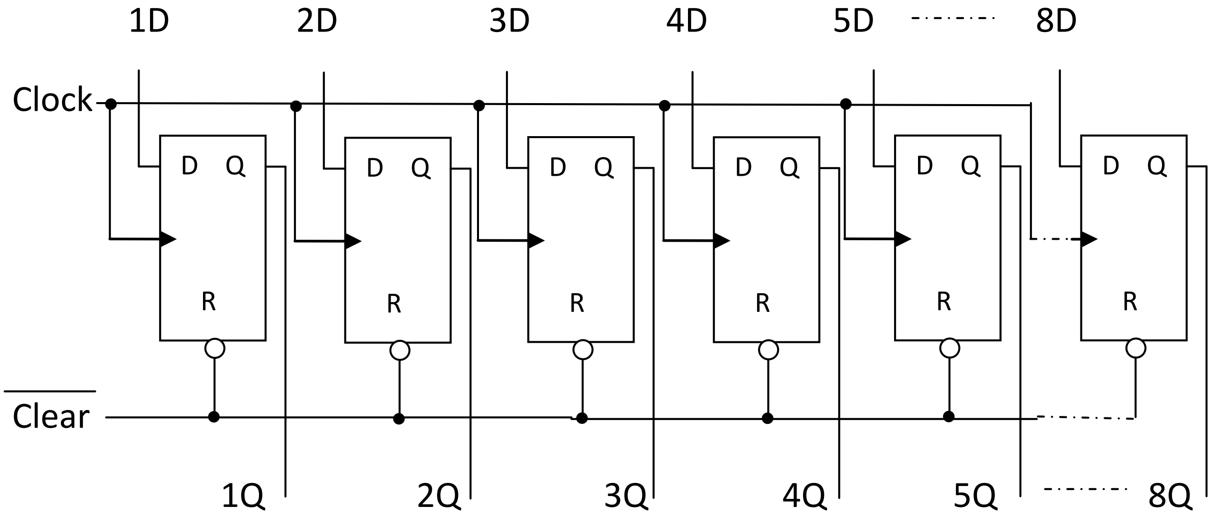


Exp.: draw the output pulses Q for D flip-flop if the input pusses D are shown bellow. Initial flip-flop state Q = 0





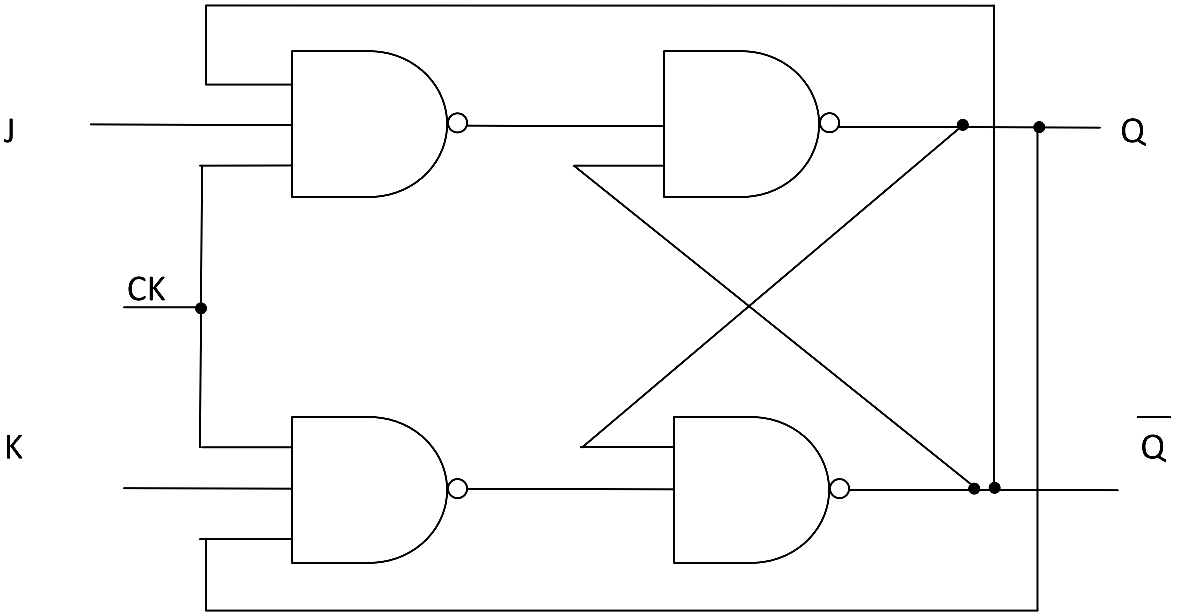
Problem: Construct a one IC digital TTL (transistor logic) flip-flop circuit ,that can store a byte of data on the leading edge of timing clock with an asynchronous clear input

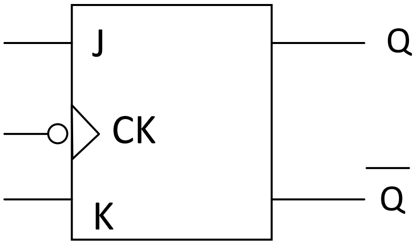


J-K Flip- Flop

The J-K Flip=Flop is one of the most widely used types of IC flip-flop. A J-K flip-flop can be wired to operation as a D or S-R flip-flop in addition to it's own truth table operation. Unlike the S-R flip-flop an unstable output state does exist when both control input are HIGH (1). Thus the J-K- flip-flop is very useful in digital design.

The logic symbol and logic circuit of J-K flip-flop are shown below:





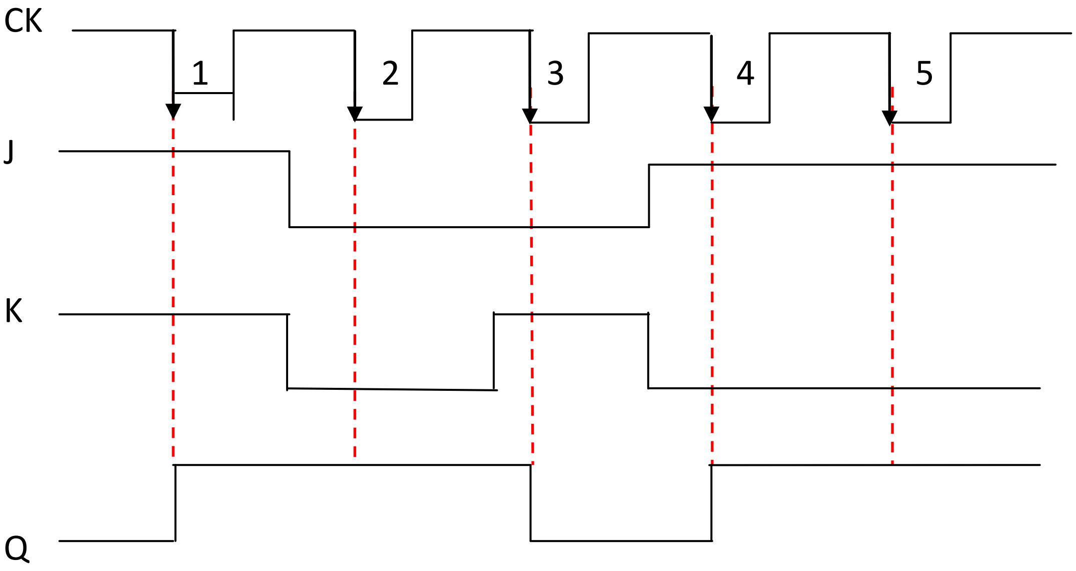
Logic symbol

Truth table

inputs			output	Mode of operation
J	K	CK	Q	
0	0	↓	Q_0	Hold(no change)
0	1	↓	0	Reset
1	0	↓	1	Set
1	1	↓	$\overline{Q_0}$	toggle

J-K flip-flop can star 1 bit data. J-K flip-flops are available as leading or trailing edge-triggered devices. The 74LS76 is as an example.

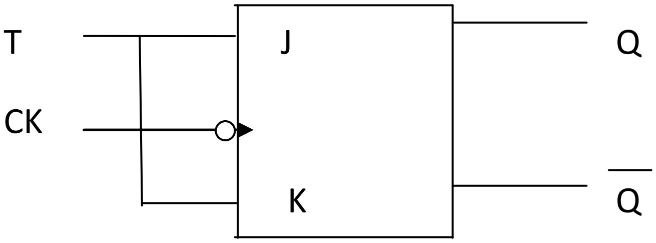
Exp.: draw outputs pulses Q for J-K flip flop the type of inputs pulses for each J-K and clock are shown. Initial state of $Q_{0\,=}\,0$ before first clock pulse.



1. By receiving first pulse CK J =1 ; K = 1 ; state is toggle, so Q_0 = 0 →Q = 1.
2. By receiving second CK pulse, J = 0 ; K = 0 state is no change ; Q = 1.
3. By receiving third CK pulse J = 0 K =1 state is reset, Q becomes = 0.
4. By receiving forth CK pulse J = 1 ; K = 0 state is set , Q becomes = 1.
5. By receiving fifth CK pulse J = 1 ; K = 0 state is still set, so Q continues to be 1.

T flip-flop

T flop-flop can be built J-K flip-flop by connecting two inputs J and K together, so T flip-flop has one input T in addition to clock pulses. T is first letter from "toggle" it means change the state.



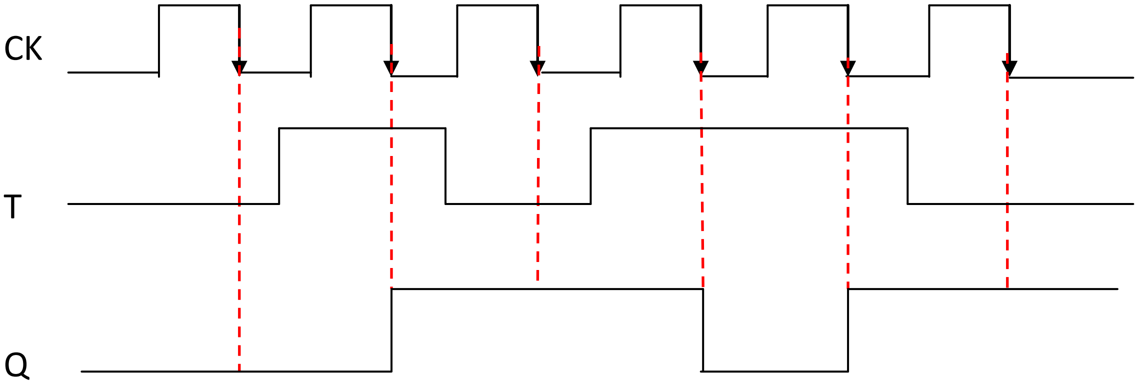
By receiving HIGH input in T (1) with the sequence pulses of clock, the output starts change it's state. In the figure the change happens at the negative edge triggered.

Truth table

inputs		output	Made of operation
T	CK	Q	
0	↓	Q_0	No change
1	↓	$\overline{Q}_0$	toggle

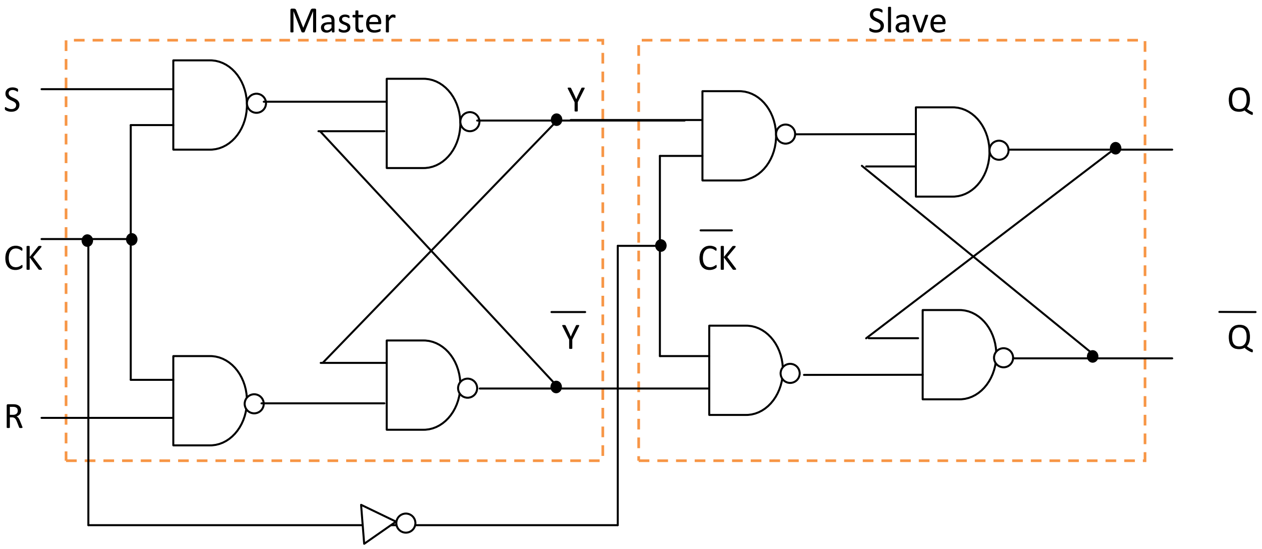
One important property of T flip-flop in toggle mode is that the output frequency is one half the input clock frequency.

Exp.: Draw output pulses Q for T flip-flop if input T and CK are shown. $Q_0 = 0$



Master – slave – flip- flop

From last studies of flip-flops, we knew, how to control operation of flip-flop by leading or traiIling edges triggered. There are other flip-flops, which can be controlled by response to pulse level (pulse triggered). These flip-flop called "master- slave flip-flop”. These master- slave flip-flops need complete block pulse to change output state. Bellow S-R type master- slave flip-flops which has two circuits of S-R flip-flop (sequential) and called (master). Another called (slave). First stage of (master) receives clock pulse (CK) directly, while second stage (slave) receives inverted clock pulse ($\overline{CK}$).



From the drawing we see that the master operation when clock pulse is positive. Slave operates when clock pulse is negative ,it means CK=0 but $\overline{CK}$ = 1 so slave operates, when $\overline{CK}$ = 1.

There are two steps happen before changing Q and $\overline{Q}$ a cording to inputs S,R.

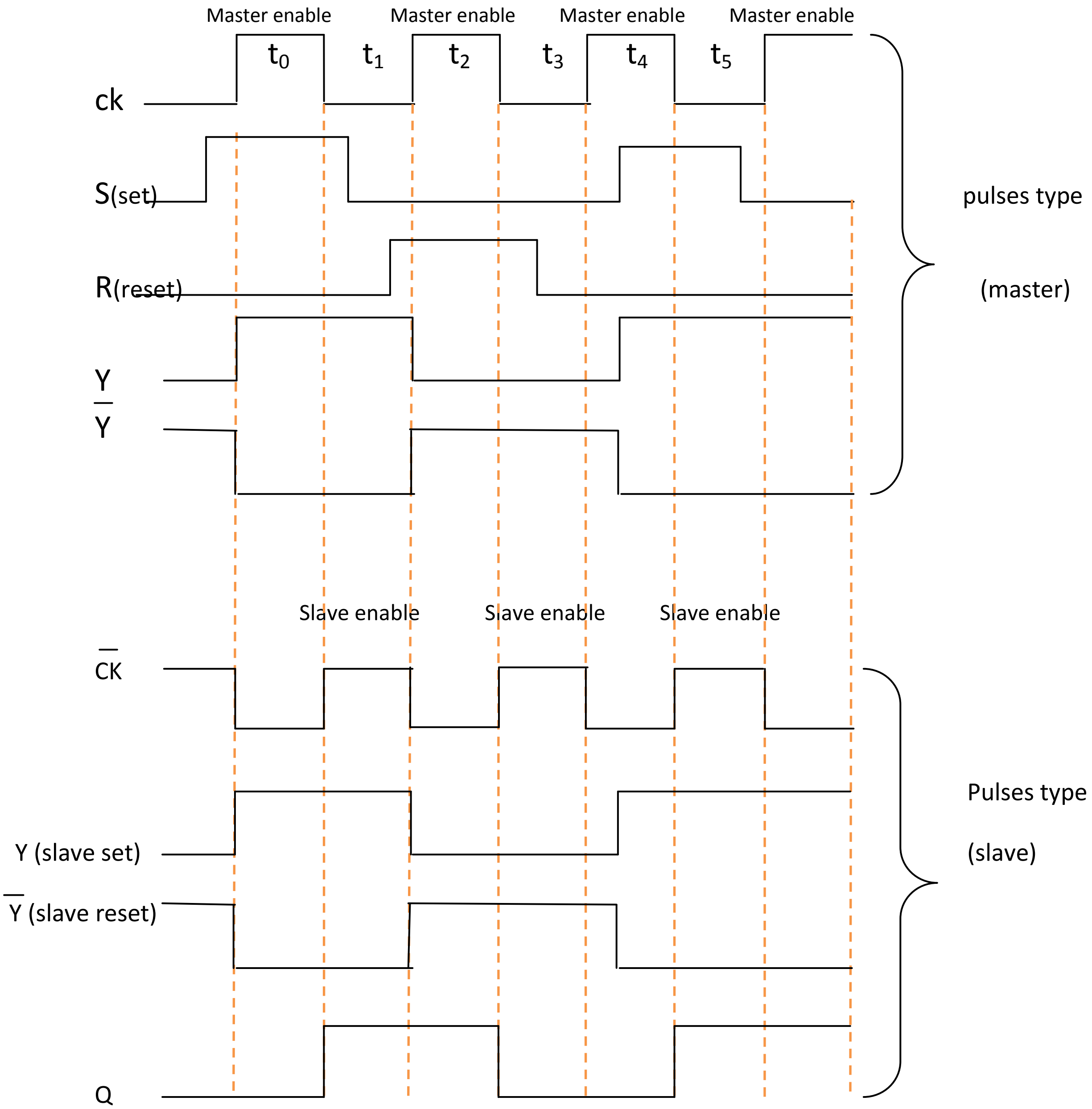
1. In HIGH level of CK, master will operate in position (enabled). Output will be in “set” or “reset” or in "hold" position.
2. In LOW level of CK, slave will be in operation position (enabled). Its output Q will depend on logic level in input y.

Truth table bellow explains the S-R master- slave operation. It is similar to S-R flip-flop will clock pulse. In the table we see complete clock pulse. Master- slave circuit needs (HIGH) a (LOW) level to operate each port. Also bellow, we see the pulse diagram for S-R master-slave. We will move from time t_0 to t_5 t0 see how the circuit will operate according to input values input $\overline{CK}$ y, $\overline{y}$ are the inputs of "slave" circuit so the output Q will be in set state (y = 1 ; $\overline{y}$ = o) , Q = 1. It means that slave circuit takes what are in its inputs one puts them in outputs, when set state is been by clock pulse. This case we say that second flip-flop is slaved the first flip-flop.

3. At (t_2) master circuit is in operation position by positives (ck) pulse(HIGH) , at this moment S= 0, R = 1. The outputs Y = 0 ; $\overline{Y}$ = 1, reset state.

inputs			output	Made of operation
S	R	CK	Q	
0	0	⎍	Q_0	Hold
0	1	⎍	0	Reset
1	0	⎍	1	Set
1	1	⎍	?	Invalid Condition

4. At (t_3) master circuit is switched off by negative (LOW) CK pulse, while is in operating. Inputs in slave circuit implement reset state: Q = 0.
5. At(t_4): S = 0 : R = 0 : Y = 0 in the middle of t_4 S become HIGH , so y became 1 also.
6. At (t_5) master switched off, while slave circuit in operation made. So y = 1 : Q =1.



- At (to) master circuit to in operation state (enable) also to positive level (HIGH) of clock pulse. In this time S = 1 ; R = 0 . And this is "set" state. Outputs Y = 1 ($\overline{Y}$ = 0).
- At(t_1) master circuit is (disabled) through LOW level CK, while (slave) circuit is in operation state (enabled) through positive (HIGH)

Quations:

Q1. A: Compare two 3 bits numbers ($A_3 A_2A_1$), ($B_3B_2B_1$) to equal. Draw the logic circuit and write the equation.

B:. ……………………." to A greater than B" ……………………….

C: ……………………. "to A less than B " …………………….

Q2.A: In active HIGH outputs decoder S_2, S_1 are address inputs; Y_0,Y_1,Y_2,Y_3 are decoded outputs. Determine the input S_2,S_1, if the $Y_2 = 1$. And draw its logic circuit.

B: ………………… if the $Y_3 =1$ …………………..

C: ………………… if the $Y_1 =1$ ………..

Q3.A: In active LOW output decoded, S_2, S_1 , are address inputs, Y_0,Y_1,Y_2,Y_3 , are decoded outputs determine the input S_2,S_1, if the Y_2=1 and draw its logic circuit.

B:. ………………….. Y_3= 1 ……………….

C:………………….. Y_1 = 1 …………………..

Q4.A: In active – HIGH outputs encoder, I_0, I_1, I_2,I_3, are address inputs, B and A are encoded outputs. Determine the outputs A at the $I_2 = 1$.

B:.. ………..$I_1 = 1$.

C: ………… $I_3 = 1$.

Q5. A:In active- HIGH encoder I_0,I_1,I_2,I_3, are inputs B , A are outputs determine when dose the A equal to 1.

B: ………………B equal to 1

C: …………….A equal to 0

D:. …………. B equal to 0.

Q6.A:In active –LOW encoder, I_0,I_1,I_2,I_3 are inputs, B and A are outputs, determine when does A equal to 1.

B: ………………A equal to 0.

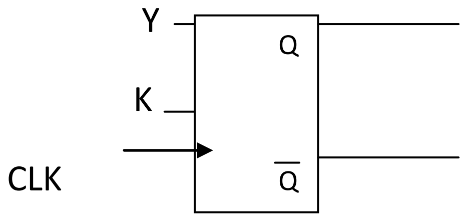
C: ………………….. B equal to 1.

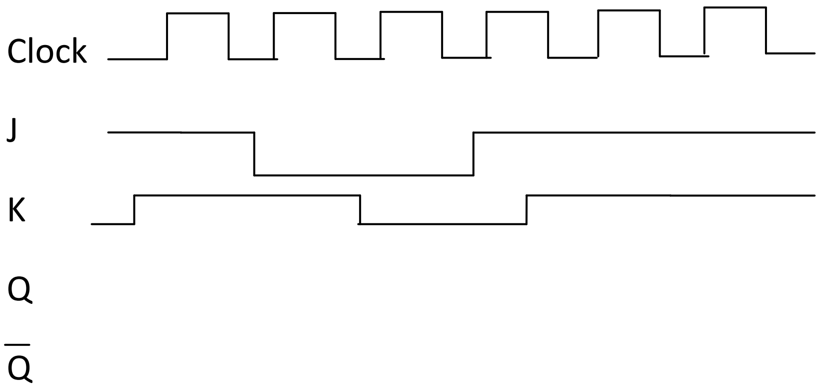
D: ………………….B equal to 0.

Q7. Put correct mark ”√ ” or error mark “×” to the following statements and correct the error statements.

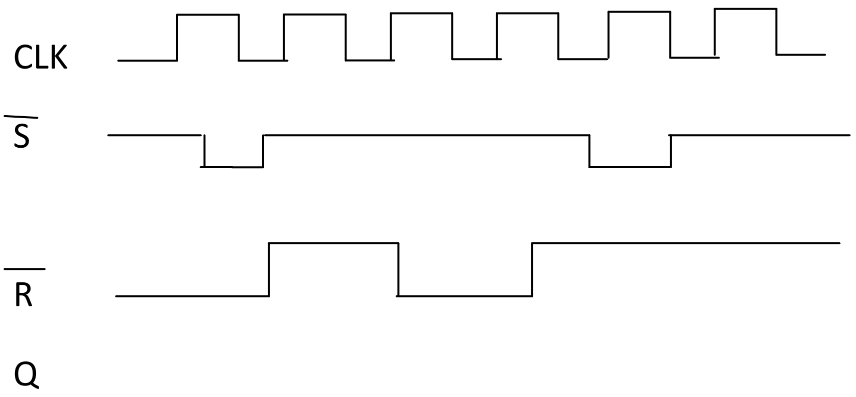
a. S-R latch has three inputs S,R and CLK.
b. D flip-flop has four operational modes.
c. $\overline{S}$ = 1 : $\overline{R}$ = 1 is invalid condition in $\overline{S}$-$\overline{R}$ latch.
d. J-K flip-flop can store 2- bits binary number.
e. S-R latch operates on leading edge triggered.
f. $\overline{S}$-$\overline{R}$ latch operates on trailing edge triggered.
g. There is no deference between S-R latch and S-R flip-flop.

Q8. Given the flip-flop and wave form shown it bellow. Determine Q and $\overline{Q}$.

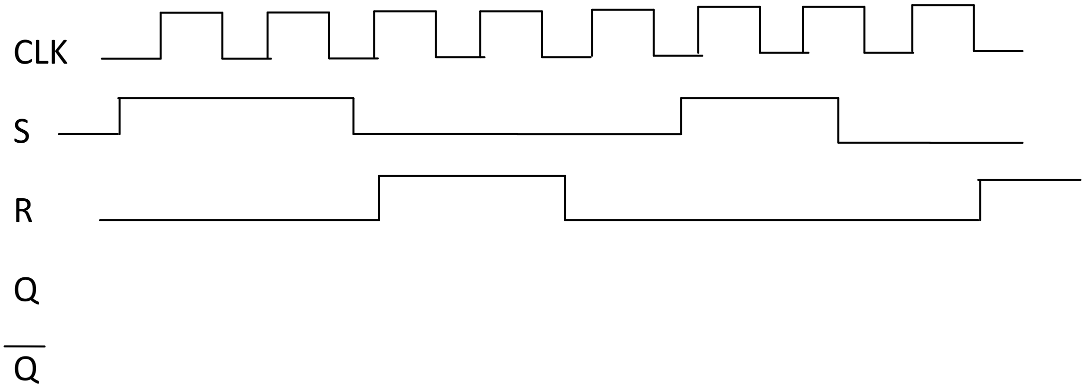




Q9. Given the wave forms shown bellow determine Q for positive edge triggered $\overline{S}$-$\overline{R}$ flop-flop.



Q10.A leading edge - triggered S-R flip-flop has inputs as shown bellow. Determine Q and $\overline{Q}$.



Q11. What is the truth table of the $\overline{S}$-$\overline{R}$ latch. Draw the logic circuit.
Q12. What is the truth table of the $\overline{S}$-$\overline{R}$ flip-flop. Draw the logic circuit.
Q13. What is the ……………. Of the J-K flip-flop ………….
Q14. What is the ……………………….. D flip-flop. …………
Q15. What is the ………………… T flip-flop. Draw.
Q16. What is …………………… S-R flip-flop. Draw.

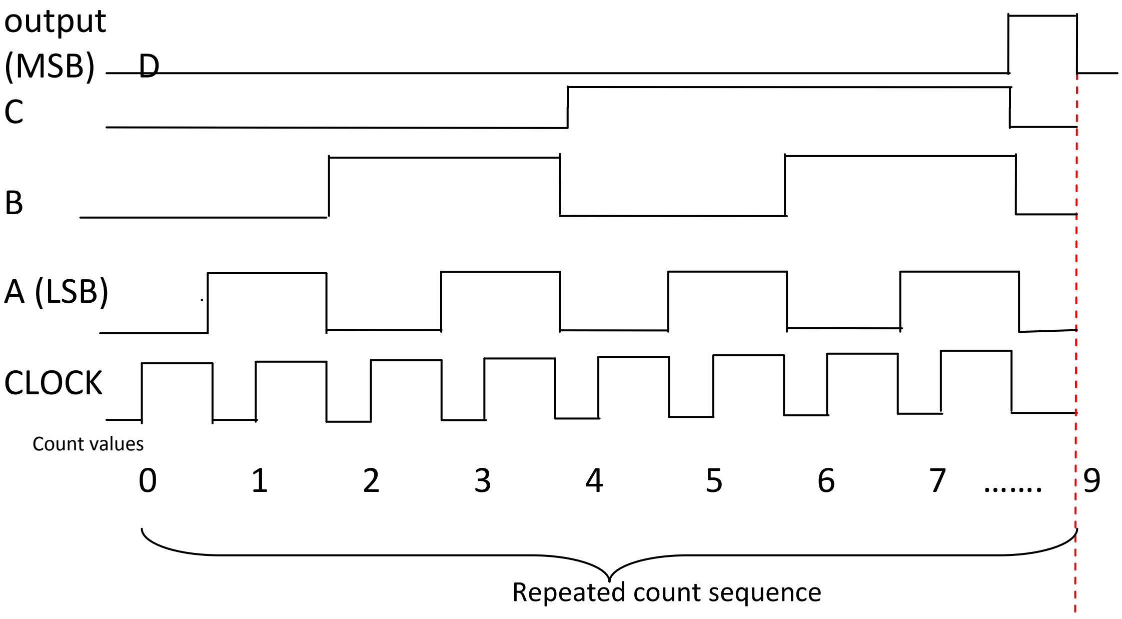
Counter circuits

Digital logic counters can be classified according to their operational characteristics. The key characteristics, that must be determined through analyzing the counter circuit are the count modulus, counter stages, output bits, frequency division, asynchronous operation, synchronies operation and trigger characteristics.

Count modulus:

The count modulus is the total number of states or values generated by the a counter as it progresses through its specified sequence. The modulus, or MOD is one of the most important characteristics to specify when classifying a counter since it identifies the number of values in the count sequence and determines the frequency division capabilities of the counter. Decade counters always have modulus of 10 and sometimes referred to MOD 10 counters. Binary counters of n bits have a modulus of 2^n.

A decade counter counts a sequence of ten numbers, ranging from 0 to 9. The counter generates four output bits whose logic levels correspond to the number in the count sequence. Bellow is the output wave forms of a decade counter.



So 4-bit output generated by decade counter and binary counter

Binary count

← (MSB) (LSB) →

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

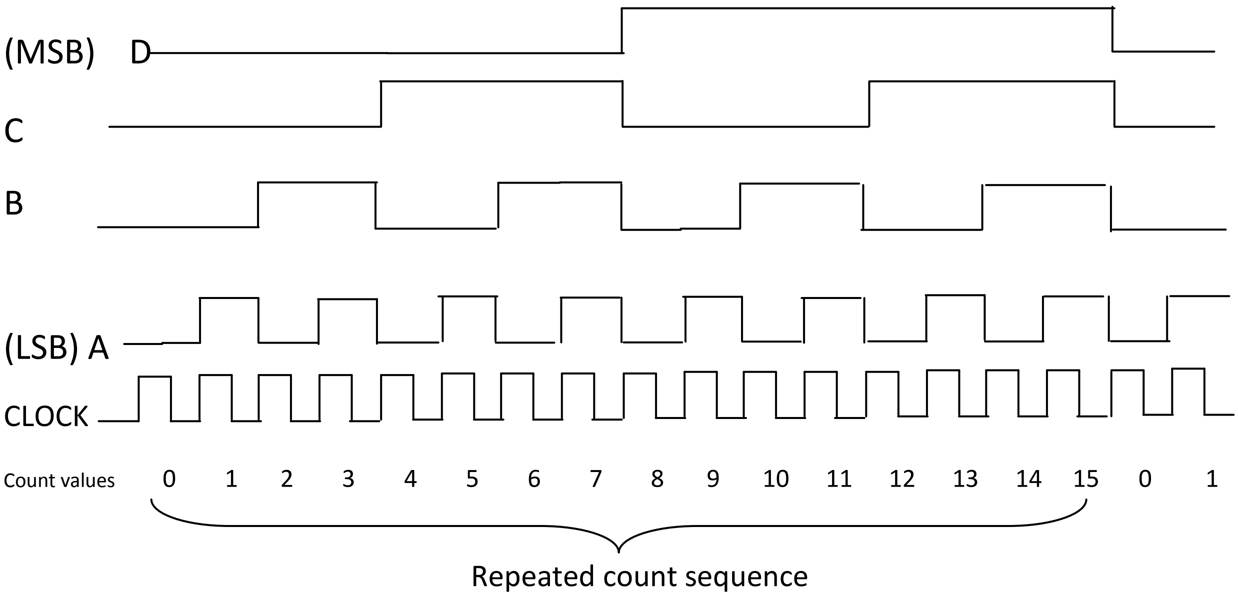
decode counter

← (MSB) (LSB) →

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

A binary counter accounts a sequence of binary numbers. A binary counter with four output bits count 2^4 or 16 numbers in its sequence, ranging from 0 to 15. Bellow is

output bits



Decade count sequence = 10 × clock period.

Binary count sequence = 16 × clock period.

Exp.: Determine the time sequence for the counter to generate the entire count sequence if the input clock has a period of 1.5 ms.

a. For decade counter.
b. For binary counter.

Solution:

a. 1.5 × 10 = 15 ms decade count sequence.
b. 1.5 × 16 = 24 ms binary count sequence.

Counter stages or bits.

The number of output bits of a counter is equal to the flip-flop stages of the counter. A MOD-2^n counter requires n stages or flip-flops in order to produce a count sequence of the desired length. The first stage of a counter is the least significant bit (LSB). The last stage of a counter is the most significant bit (MSB).

Frequency division.

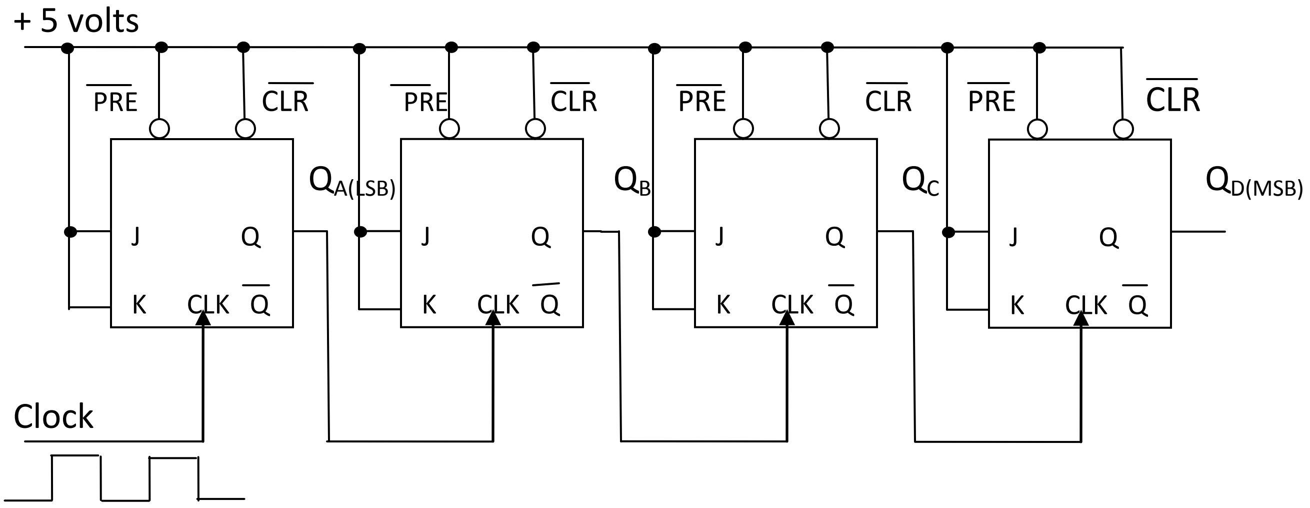
Digital counters function as frequency dividers since they divide the input control clock frequency by the modulus of the counter. As shown in binary counters that count up or down in a country sequence. The clock frequency is divided.

By 2 of the output of each state of the counter. The output wave form produced of the most significant output state a frequency of 1 / modulus of the counter. Therefore a “MOD-16” counter can also be referred to as a “alivie- by 16” counter.

Frequency division is an important property that is required when designing digital clock circuits or other applications needing several frequencies that are integer factors of the original clock signal. Counters serve the significant function of producing output wave forms that are synchronous with the incoming clock but are divided by a factor of 2 of each stage of the counter.

Asynchronous counters:

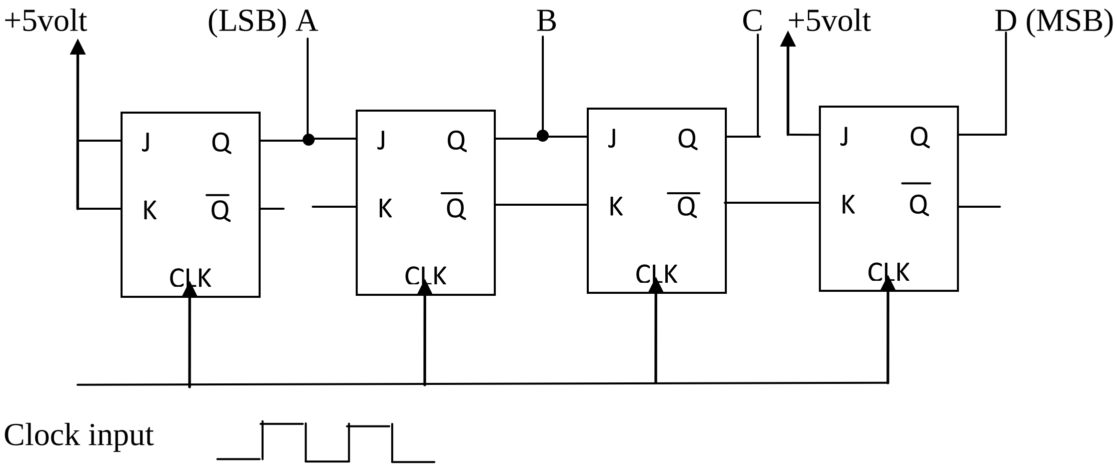
A synchronous counters are also refer to as ripple counters. The incoming clock wave form is routed in to the first stage of the counter, which generates the LSB of the numbers in the count sequence. The output of the LSB stage serves as the clock input to the next stage. Each stage of an asynchronous counter obtains its signal from the output of the prior stage, which results in the clock signal rippling through all the flip-flops of the counter. A 4 –bit asynchronous counter is shown bellow.



Synchronous counters

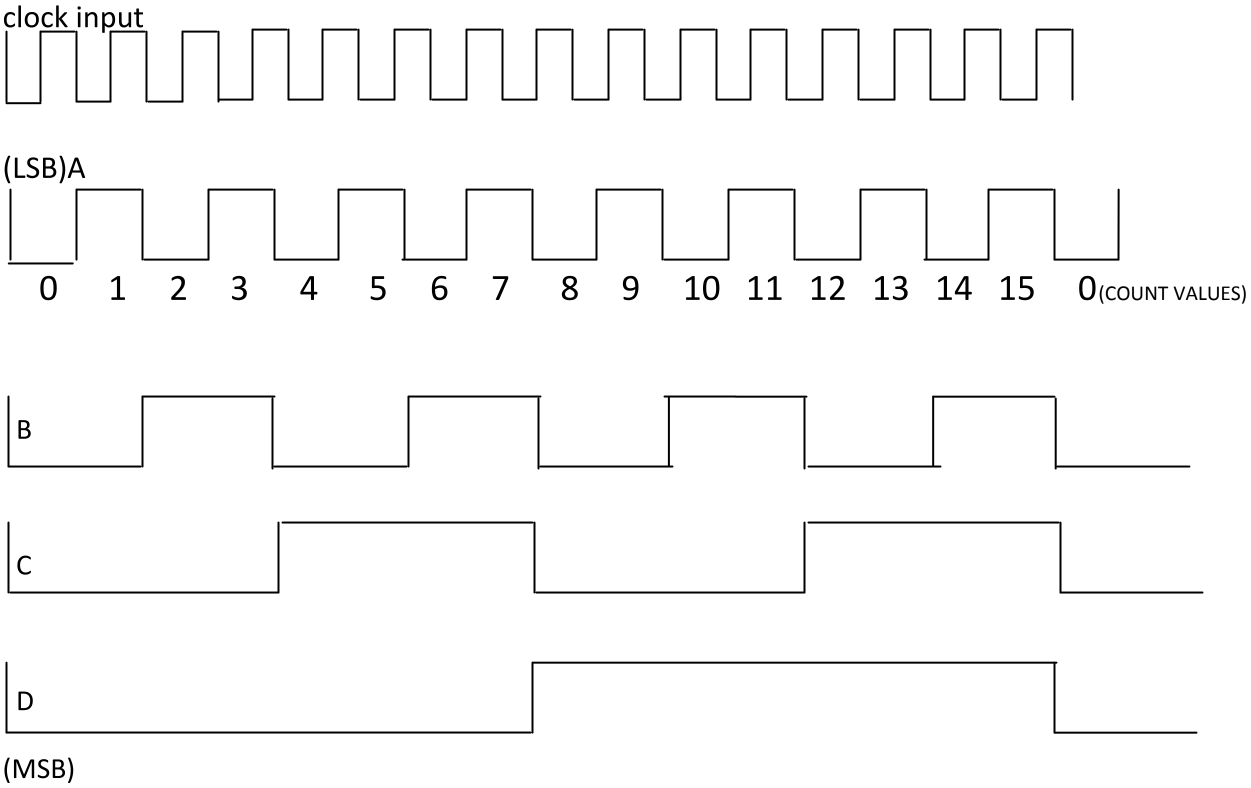
Synchronous counters are construction with one common clock signal as the input to all the flip-flops simultaneously. The clock does not ripple through the counter stage. Synchronous counters are also referred to as parallel counters due to the parallel manner that the clock is fed to all the counter stages.

A basic configuration of a 4 bit synchronous counter is shown below:

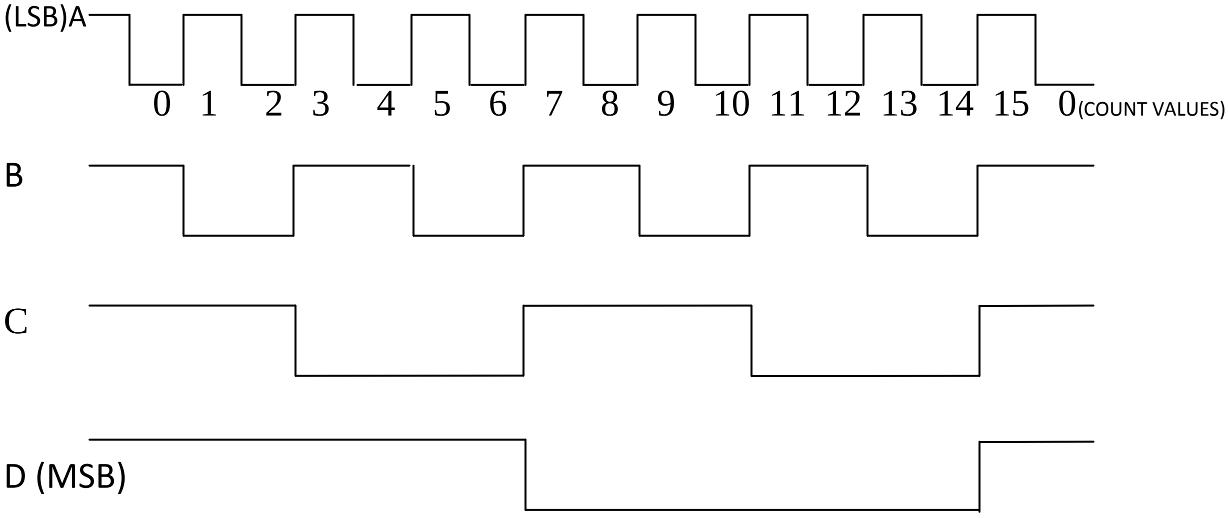


Counter triggering:

A counter is classified as a positive edge- triggered or negative edge-triggered device, depending on the trigger characteristic of the flip-flop circuit that form the counter. It is impotent to know the triggering of the counter because the count staTes only change on the triggering edge of the incoming clock. The figure bellow compares the output wave from for a positive edge-triggered and negative edge-triggered 4 – bit counter. Notice the timing shift relative to the incoming clock pulse:



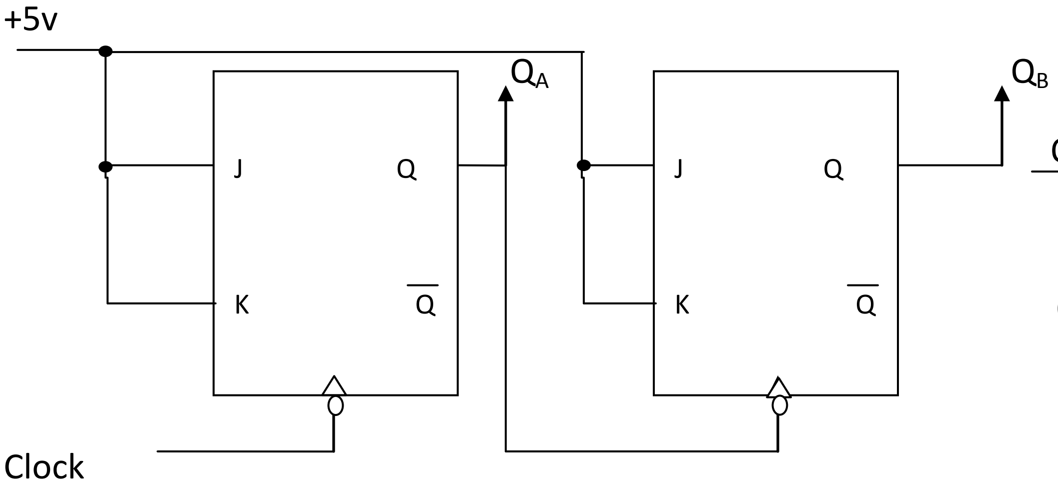
Negative edge triggered counter



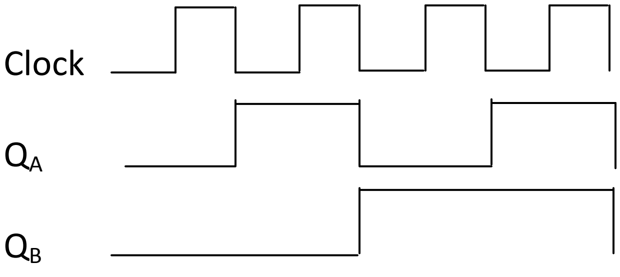
Positive edge- triggered counter

Exp.: Two- bit binary counter analysis.

Problem: analyze the 2 bit counter show bellow:



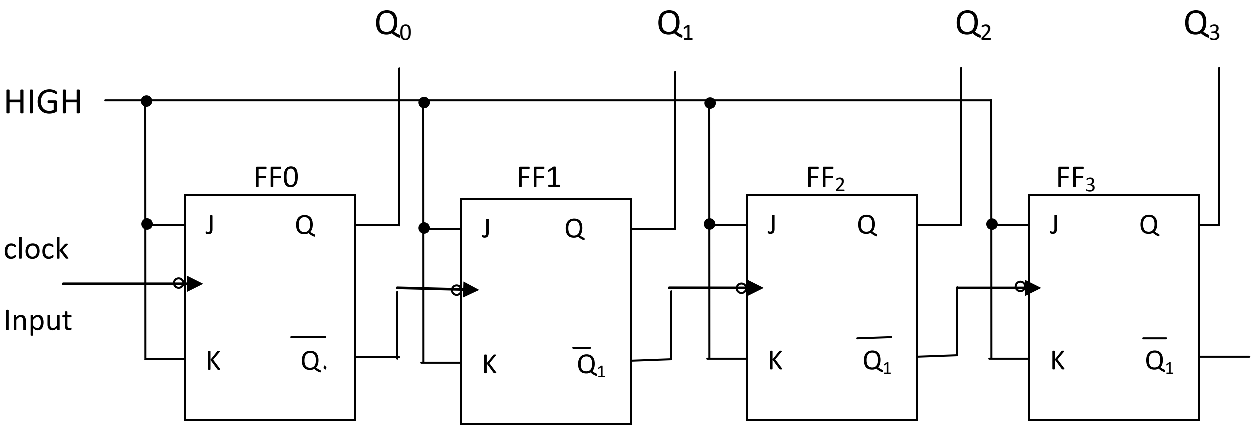
QB	QA
0	0
0	1
1	0
1	1

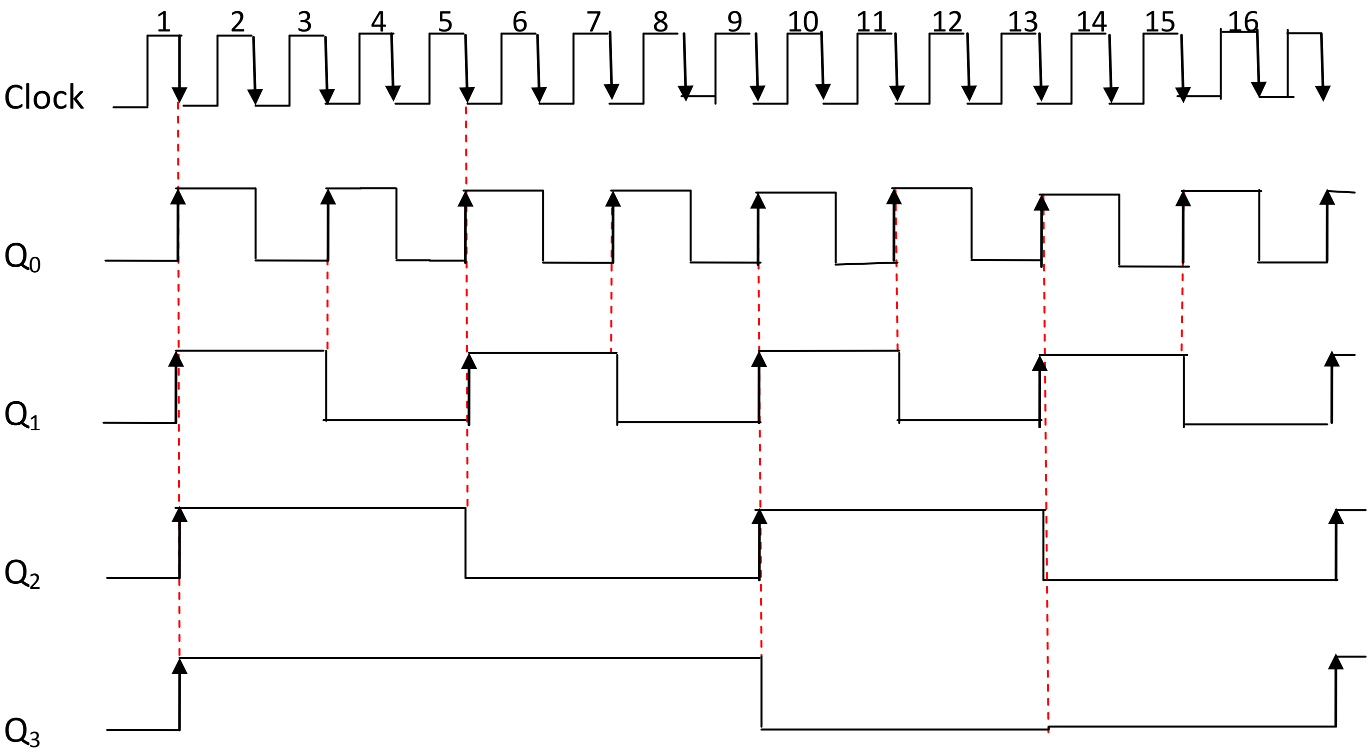


Solution: The counter has "2" flip-flops and 2 output bits, therefore it is a stage counter. The input clock does not trigger both flip-flops, and thus it is an asynchronous counter circuit. The Q_A output is used as the clock to the second stage of the flip-flop. The J-K flip-flops have the J and K inputs tied high, so they are considered Toggle flip-flop .The flip-flops are shown as negative edge triggered devices. $Q_{A0} = 0$; $Q_{B0} =0$; Q_A is LSB. Q_B is MSB. Q_A frequency is half clock frequency: Q_B frequency is quarter clock frequency. MOD-4.

Asynchronous binary down counters.

In binary up counter. The clock pulse make the counter output increasing by "1" by simple modification in binary up counters circuit, we can get binary down counter which decrease the output each pulse by "1". Figure bellow shows, how we can build down counter from four stages. Using for J-K flip-flop logic circuits. There we see the use of $\overline{Q}$ as clock pulse for each stage instate of Q.





Shift registers

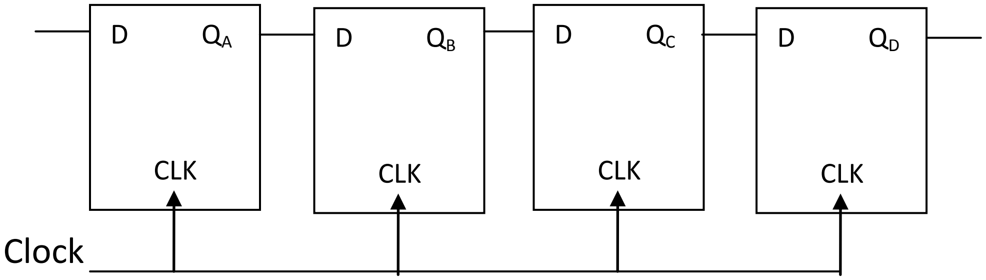
Shift registers are made up of an array of flip-flop stage to store and transport data. Shift registers are commonly use to hold several bits of data for storage or buffering prior to transmission, for conversion between serial and parallel formats.

Data storage and transfer.

Binary data must be transferred and stored in many logic applications. Bits that are generated or transferred one bit at a time are classified as serial data. Multiple bits that are generated simultaneously are classified as parallel data. In many application data must be transmitted from one area of a large system serial format or in parallel format. Data must be converted from a serial to a parallel format if the is generated

one bit at a time and must be processed in a parallel format. However, data must be converted from a parallel to a serial format in many data transmission application.
Shift registers are digital routers that transfer data in serial or parallel formats and convert between the serial and parallel formats.
Shift registers are used to control the two of data to other digital devices through format conversion, data are conversion, buffering and the delay of the data.
A shift register consist of D flip-flop stages connected by common clock that controls the input and output timing action of the flip-flops, as shown bellow:

Four- bit shift register



A shift register is not a counter. A shift register has no define of sequence. It is designed to accept data from a single input (serial input) or multiple inputs (parallel inputs) and to transfer the data to a single output (serial output) or to multiple outputs (parallel outputs).

Data enters the shift register and is transferred only on the triggering edge of the clock pulse. There are four categories of shift register operation.

1. Serial-in serial-out. Data enters the shift register (bit out a time and exist the shift register 1 bit at a time. With this type of operation, shift registers serve as temporary data storage device. There is no format conversion.
2. Serial-in, parallel-out. Data enters the shift register 1 bit at a time and exists the shift register as a multi bit binary word with all bits simultaneously available. In this type of operation the shift register converts from a serial to a parallel format

and offer serves as a data buffer. Serial computer communication system commonly transmits data in a serial format. This must be received and converted to a parallel format prior to arriving device.

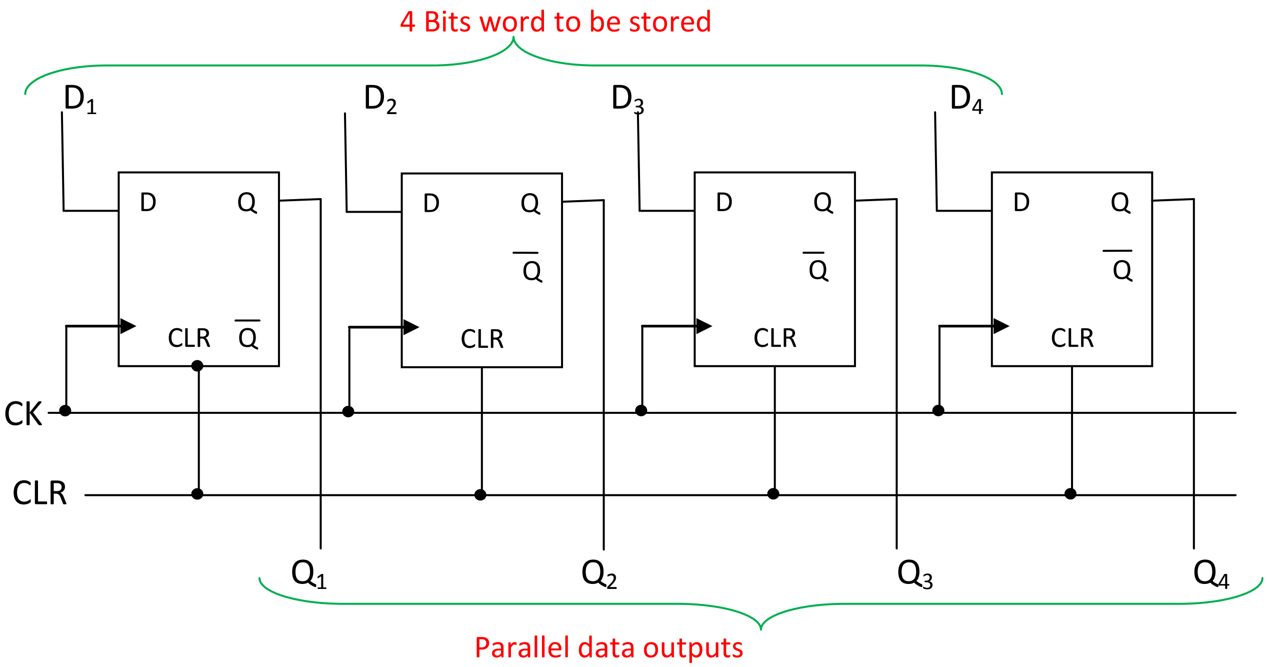
3. Parallel-in,parallel-out. Data enters the shift register as several simultaneous bits and exits the shift register as several simultaneous bit. No format conversion takes place. The shift register serves as a temporary data storage device.
4. Parallel-in, serial-out. Data enters the shift register as several simultaneous bits and exits the shift register 1 bit at a time. The shift register converts the data from a parallel format to a serial format and can temporary store the data.

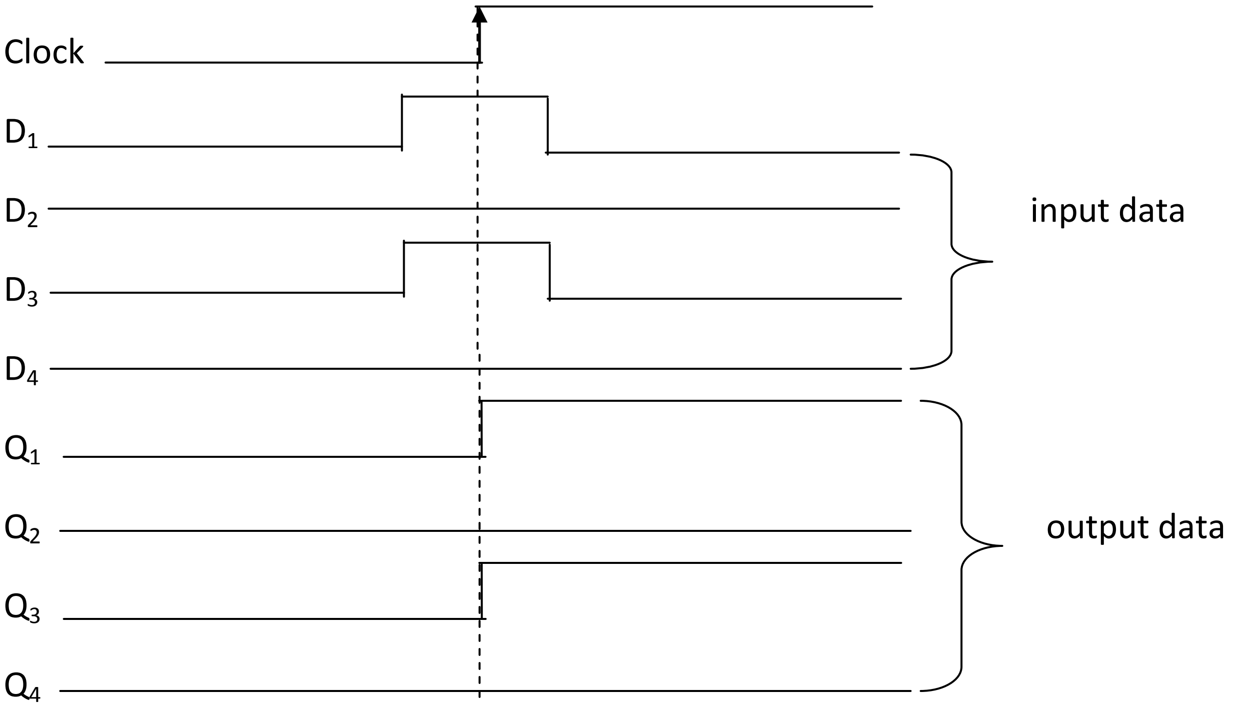
Serial data requires several clock pulses for data to be loaded into or out of the shift register. One clock pulses is required for each flip-flop stage of the register to lead the data sequentially into the register or to transfer it out of the register. Only one input and output line is registered for serial transmission.

Parallel data register only one clock pulse for data to be leaded into or out of the shift register since all the bits are transferred simultaneously into or out of the flip-flops. Although parallel transmission is much quicker than serial transmission, an input line and an output line are required for each bit.

Buffer register

Buffer register is simply used for storage “digital word” built from group of binary numbers(bits). Figure bellow is shown how to build buffer register from 4 stages, using D- flip-flops- flaps, which are worked at the positive. Edge- triggered.


4 Bits word to be stored
D1
D2
D3
D4
D
Q
CLR
Q̄
CK
CLR
Q1
Q2
Q3
Q4
Parallel data outputs

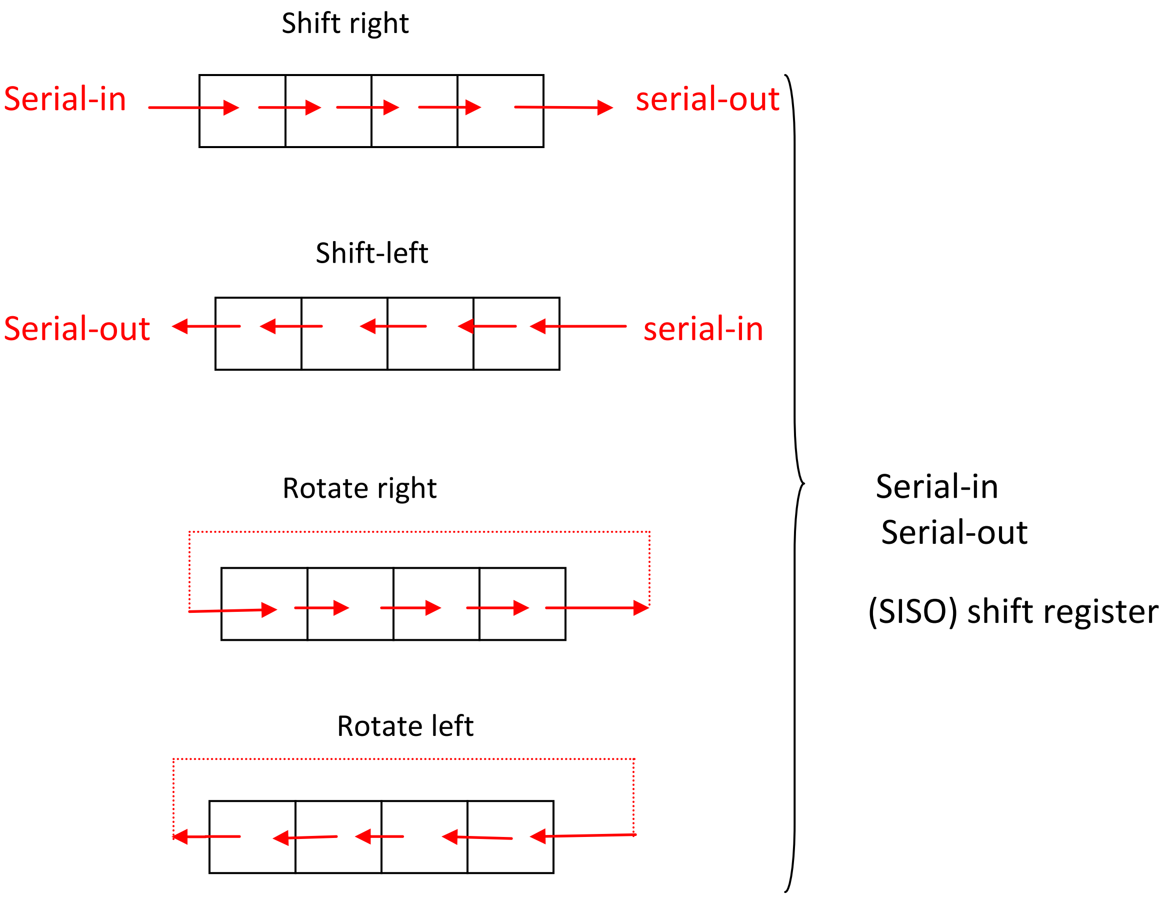

Clock
D1
D2
D3
D4
input data
Q1
Q2
Q3
Q4
output data

Data, which is required to store (4-bits binary number) applies in inputs D_1, D_2, D_3 D_4, and appears as output Q_1, Q_2, Q_3,Q_4, at first clock pulse (positive edge triggered). Data in the outputs Q_1, Q_2, Q_3,Q_4 is stabile at the outputs.

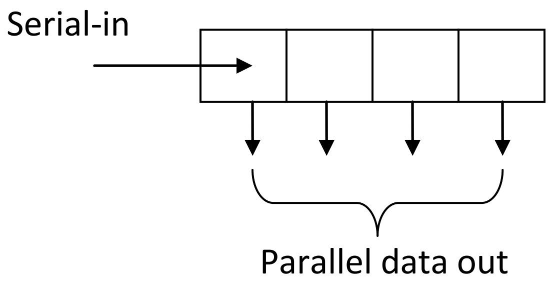
We entered a word from 4- bits in parallel into register inputs, and get in parallel also at the output, so in general buffer registers are called parallel-in, parallel-out registers. Clear-input in active low is used for clear a word from all flip-flops.

Shift registers مسجلات الازاحة

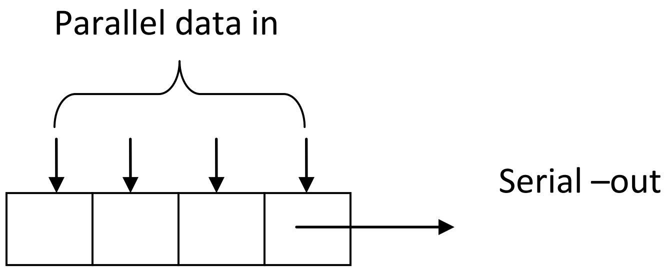
Shift register is a register for storage data preliminary to move it or shift to the left on to the right. There are three main types of shift register:



Serial-in, parallel-out (SIPO) shift register



Parallel-in, serial-out (PISO) shift register



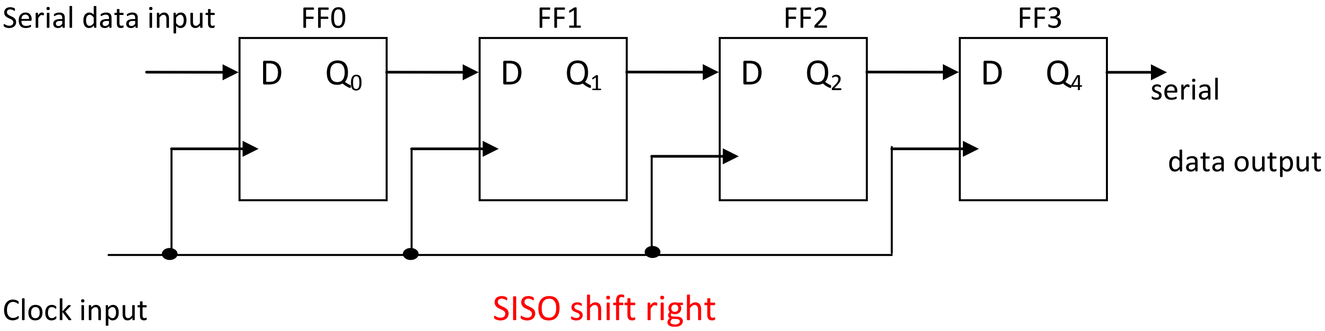
Serial-in, serial-out (SISO) shift registers

We will start with truth table. In this example, we find the register contains the initial data (0110). While the output serial data is (1001), is waiting in the register for shifting.

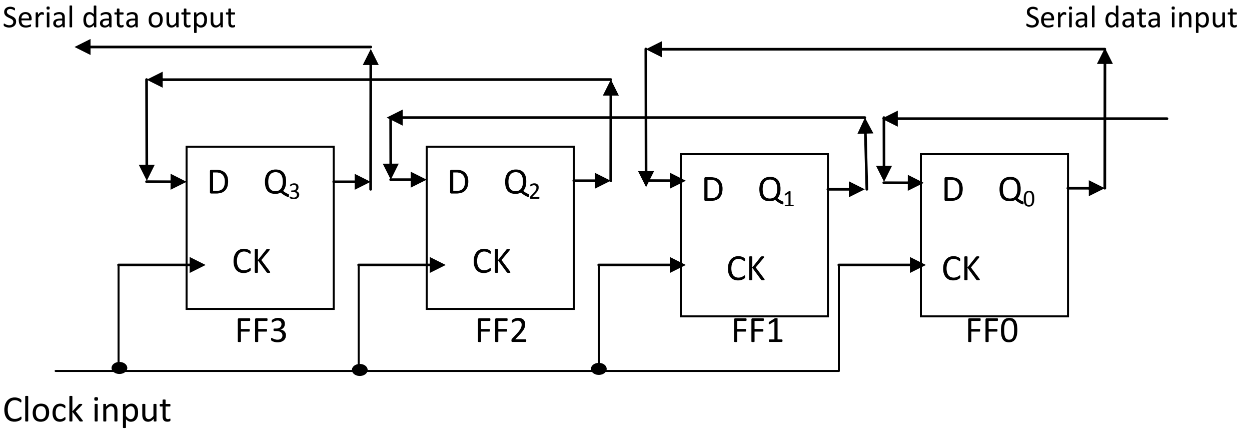
Clock pulse	Data needs to store	Register outputs			
clk	input	Q_0	Q_1	Q_2	Q_3
-	-	0	1	1	0
1st	1	1	0	1	1
2nd	0	0	1	0	1
3rd	0	0	0	1	0
4th	1	1	0	0	1

After first clock pulse storage data in the register will be shifted one step (one bit) to the right. At the same time the first number from the serial input data will be shifted inside the first bit in the register. After second clock pulse (2nd) two numbers from storage numbers (0110) were shifted out of register, while two numbers from serial inputs (1001) will be stored. After third clock pulse, three shifts the right were done. And so for fourth pulse.

Digital logic circuit:



SISO shift right

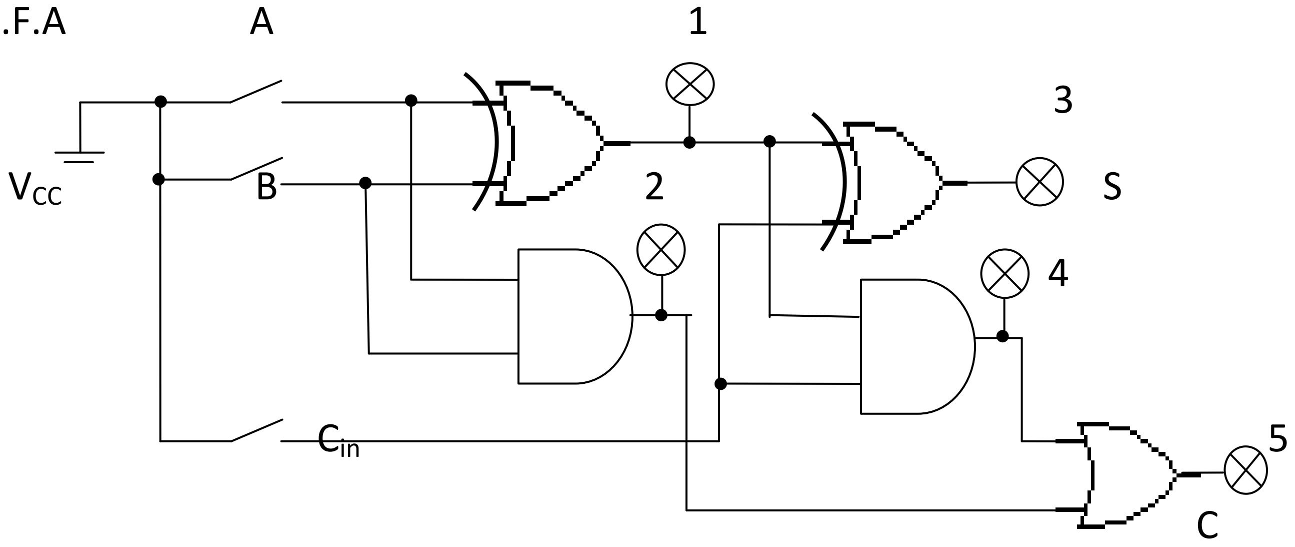


SISO shift left

L.W.No:3

Full-adder and full-subtractor

1.F.A



Inputs outputs

A	B	C_{in}	1	2	3	4	5
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	1	0	0	1	1
1	0	0	1	0	1	0	0
1	0	1	1	0	0	1	1
1	1	0	0	1	0	0	1
1	1	1	0	1	1	0	1

Second tearm

L.W No. 4

Comparators

Comparators are logic circuits, that determine of input values are less than, greater than, or equal to each other. Comparators are used in numerous applications, many of which are arithmetic operations.

1. Equal: for 4 –bits binary numbers equal equation is:

   $$A = B\ ;\ Y = (\overline{A_4 \oplus B_4})(\overline{A_3 \oplus B_3})(\overline{A_2 \oplus B_2})(\overline{A_1 \oplus B_1})$$

2. Greater than: for 4-bits binary numbers:

   A>B;

$$Y= A_4\,\overline{B_4}+\ A_3\,\overline{B_3}\,(\overline{A_4 \oplus B_4})+A_2\,\overline{B_2}\,(\overline{A_4 \oplus B_4})\,(\overline{A_3 \oplus B_3})+ A_1\overline{B_1}(\overline{A_4 \oplus B_4})$$

$$(\overline{A_3 \oplus B_3})\,(\overline{A_2 \oplus B_2})$$

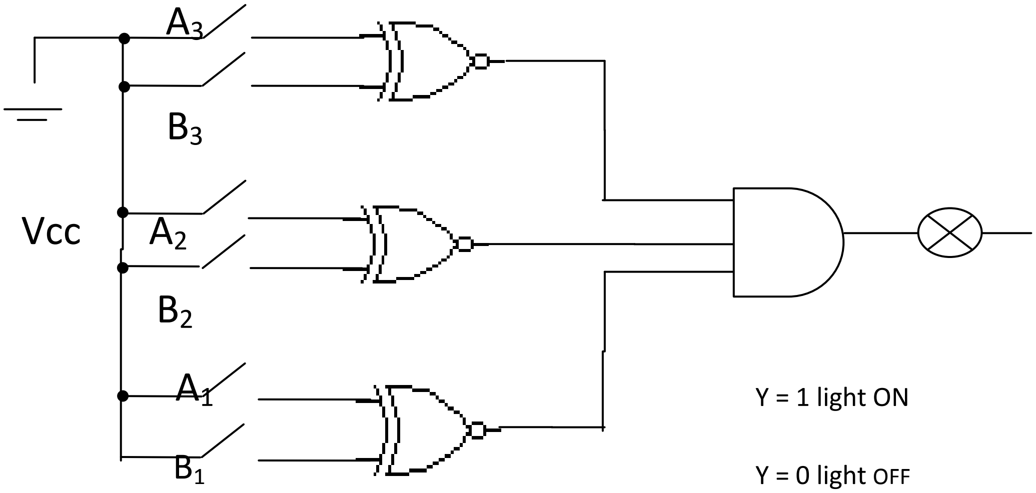
3. Less than: for 4-bits binary numbers:

   A<B ;

   $$Y= \overline{A_4}\,B_4+\ \overline{A_3}\,B_3\,(\overline{A_4 \oplus B_4})+\overline{A_2}\,B_2\,(\overline{A_4 \oplus B_4})\,(\overline{A_3 \oplus B_3})+ \overline{A_1}B_1(\overline{A_4 \oplus B_4})$$

   $$(\overline{A_3 \oplus B_3})\,(\overline{A_2 \oplus B_2})$$

Experiment No.1 logic circuit for 3-bits comparator that test for "A "equal to" B"



Y = 1 light ON

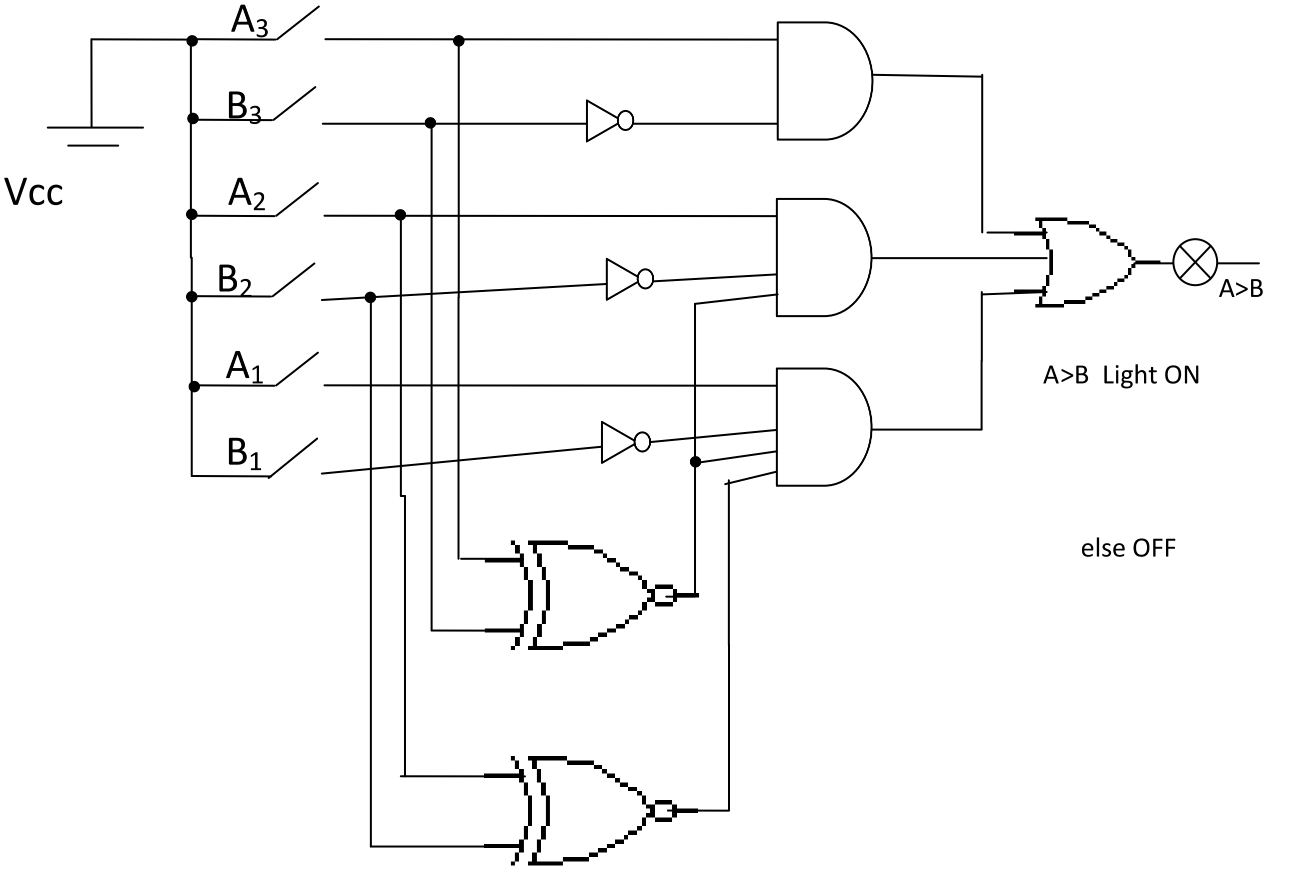
Y = 0 light OFF

1	A = 1 0 1
	B = 1 1 1
		Y=0	Equal: yes
			[✓ No]

2	A = 1 1 0
	B =1 1 0
		Y=1	Equal: [✓ yes]
			No

3	A = 0 1 0
	B = 0 1 1
		Y=0	Equal: [✓ yes]
			No

Experement No:2: logic circuit for 3-bitscomparator,that test "A" greater than "B".



Test table

1	A = 0 1 0
	B = 1 0 0
		Y = 0	Greater than
			Yes No

2	A = 1 1 0
	B = 1 0 1
		Y = 1	Greater than
			Yes No

3	A = 1 0 1
	B = 0 1 1
		Y = 1	Greater than
			Yes No

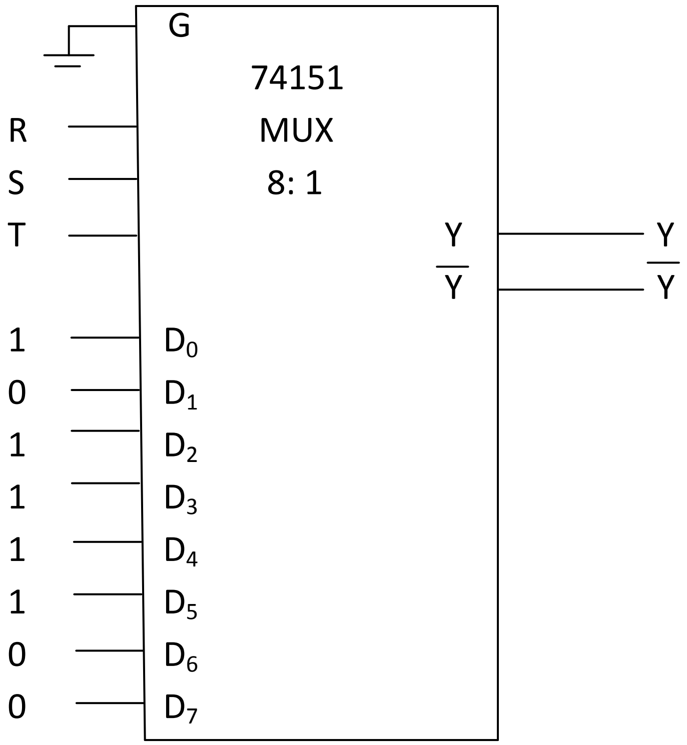
Questions:

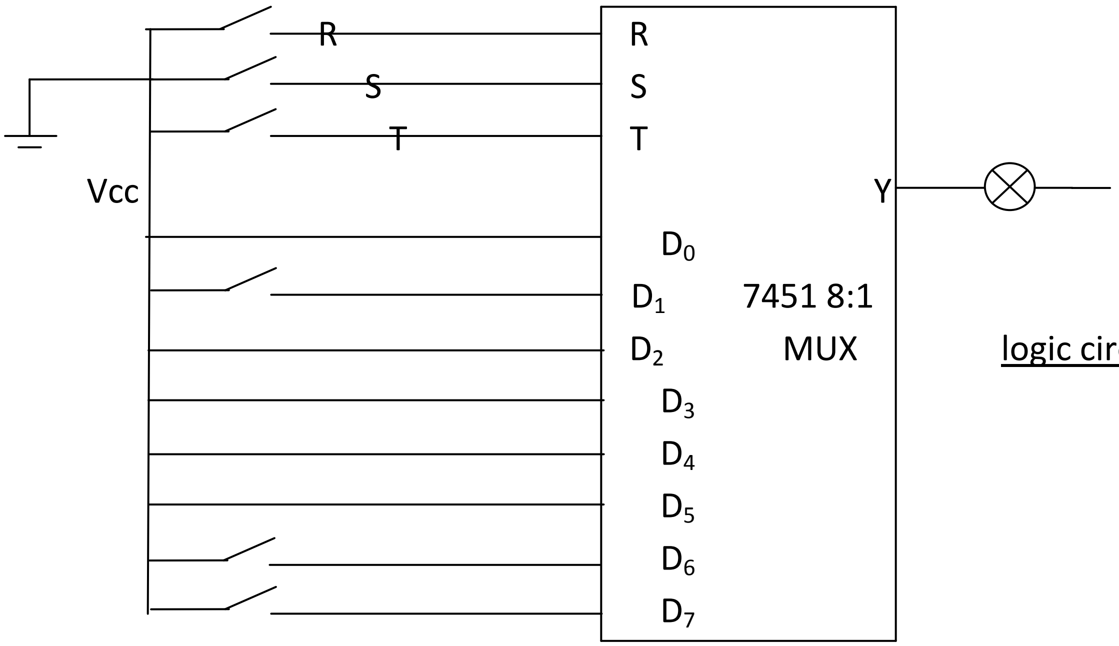
1. Draw logic circuit for 2-bits comparator that test for “A ” leas than “B”.
2. How many X-NOR gets in logic circuit for 5-bits comparator that test for “A” greater than “B”.
3. Draw one logic circuit for 2-bits comparator that test for “A “ equal to “B” and “A” greater than “B”.

Second term
L.W No.5
Multiplexer and decoder

1. Experiment No.1 multiplexer
   Implementation of logic circuit function specified in the truth table using 74151 8 : 1 mux.

R	S	T	Y	
0	0	0	1	D_0
0	0	1	0	D_1
0	1	0	1	D_2
0	1	1	1	D_3
1	0	0	1	D_4
1	0	1	1	D_5
1	1	0	0	D_6
1	1	1	0	D7





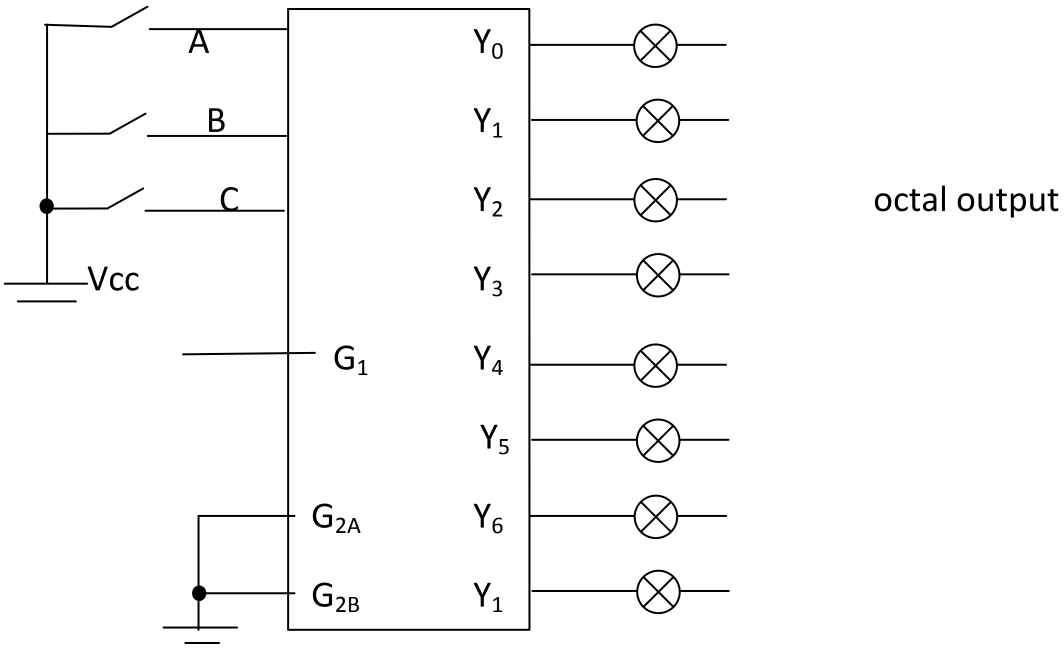
logic circuit

Test table

R	S	T	Y
1	0	1	1
1	1	0	0
0	0	0	1
0	1	1	1
1	0	0	1
0	0	1	0
0	1	0	1

2. Experiment No.2 : decoder
   Determination the binary input required to produce the specified random output sequences using the 74138 for code conversion. Output sequence: 7,3,0,2,1,5,4,6

Logic circuit



Test table

input			Output activated
C	B	A	
1	1	1	$Y_7 = 0$
0	1	1	$Y_3 = 0$
0	0	0	$Y_0 = 0$
0	1	0	$Y_2 = 0$
0	0	1	$Y_1 = 0$
1	0	1	$Y_5 = 0$
1	0	0	$Y_4 = 0$
1	1	0	$Y_6 = 0$

Question:

1. What is the function of multiplexer?
2. What is the function of decedent?

Second term

L.W No.6

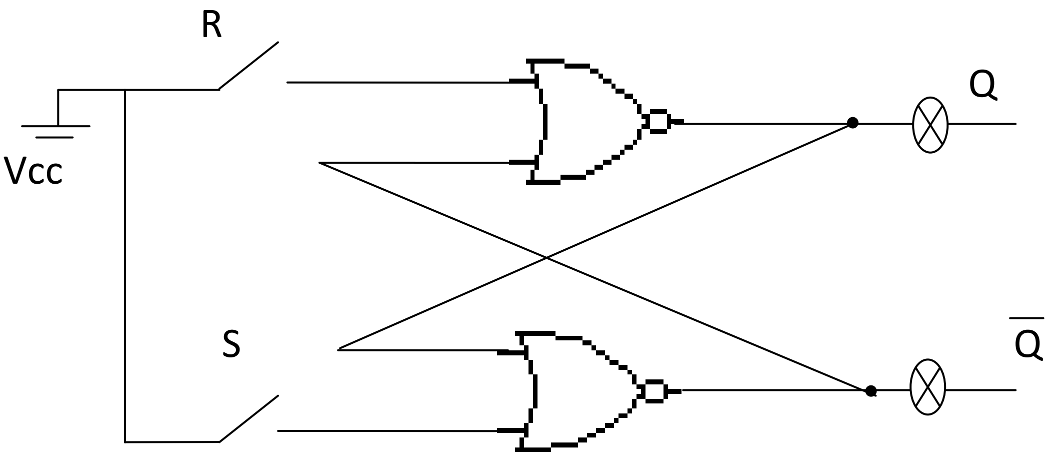
Latches

Latch: is a basic digital storage device that is capable of storing one bit of information.

S-R and $\overline{S}$-$\overline{R}$ latches:

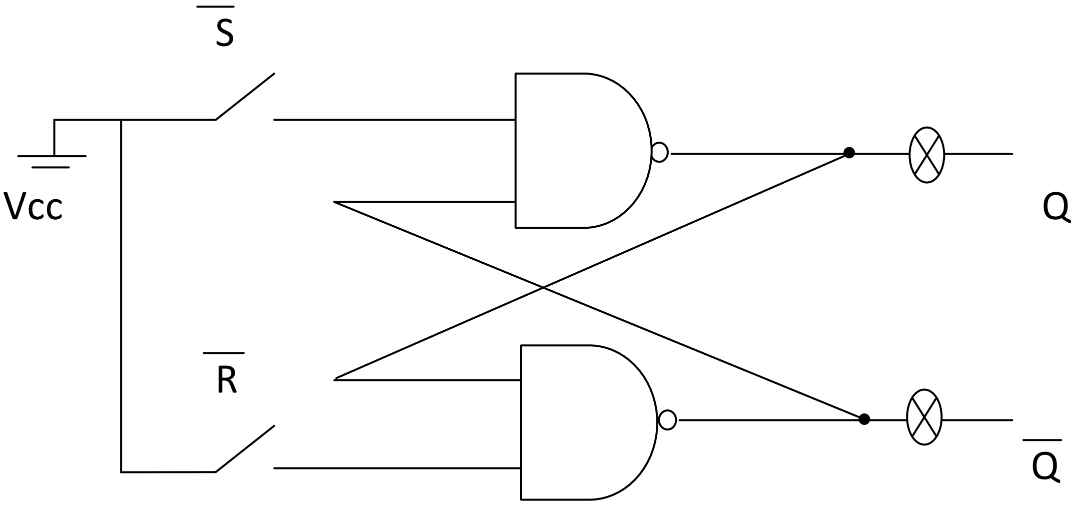
S-R and S-R latches are bistable device, each of which is capable of storing 1 bit of data. These types of latched can be built with two basic logic gates utilizing feedback.

S-R latch



S	R	Q $\overline{Q}$
0	0	
0	1	
1	0	
1 - 1		invalid

$\overline{S}$-$\overline{R}$ latches



S	R	Q Q
0	0	invalid
0	1	
1	0	
1	1	

Questions:

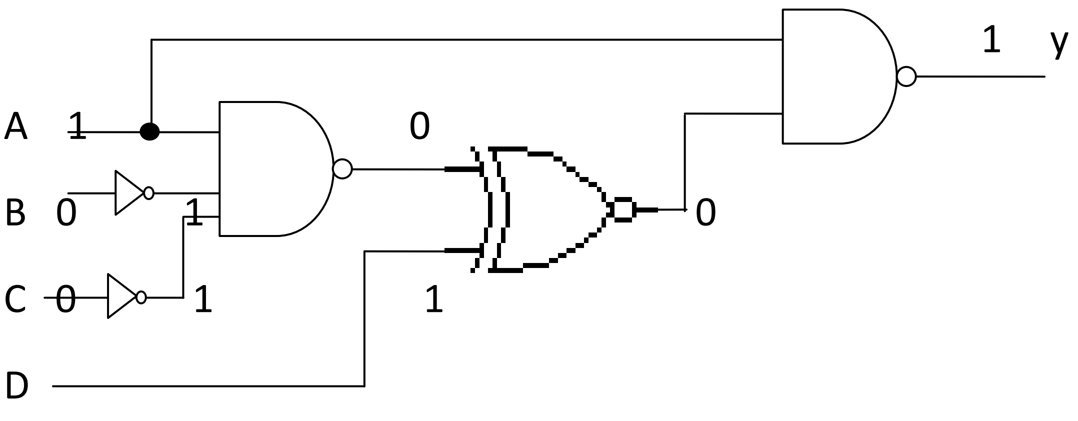
1. Write the truth table of j–k flip-plop?
2. Draw the logic circuit of D flip-plop?

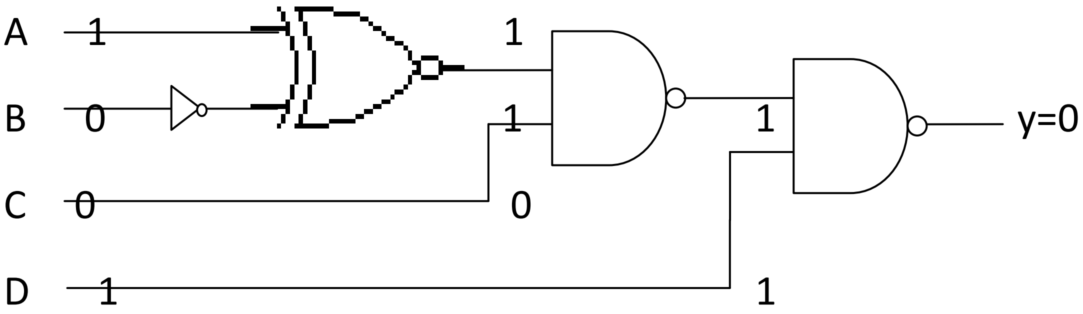
Quations

Q1. Draw a logic circuit from truth table using k-map:

A	B	C	(For A group) y	(For B group) y
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	1
1	1	1	0	0

Q2 write Boolean expression for this logic circuit and what is the output logic number (0 or 1)(y=…..) if A = 1, B = 0, C = 0, D = 1



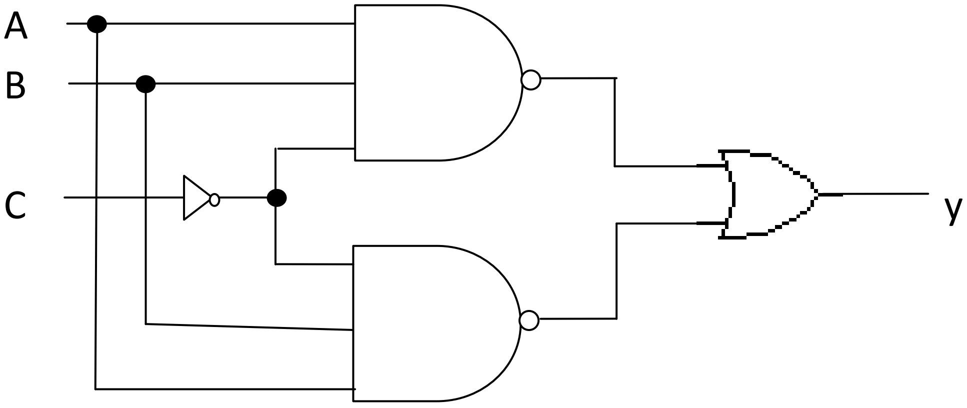


Q3. Draw logic circuit from Boolean expression using k- map.

a- $Y = \overline{A}\ B\ \overline{C}\ D + \overline{A}\ \overline{B}\ \overline{C}\ D + A\ B\ C\ \overline{D} + A\ B\ \overline{C}\ D + \overline{A}\ \overline{B}\ \overline{C}\ \overline{D}$

b- $Y = A\ B\ C\ D + A\ \overline{B}\ C\ \overline{D} + A\ B\ C\ \overline{D} + \overline{A}\ \overline{B}\ \overline{C}\ D + \overline{A}\ \overline{B}\ C\ \overline{D}$

4-a :Write Boolean expression for this logic circuit.



b.Draw logic circuit from logic Boolean expression:

$\overline{A}\,B\,(C + \overline{D}\,)$

5.Draw logic circuit from truth table using k-map:

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

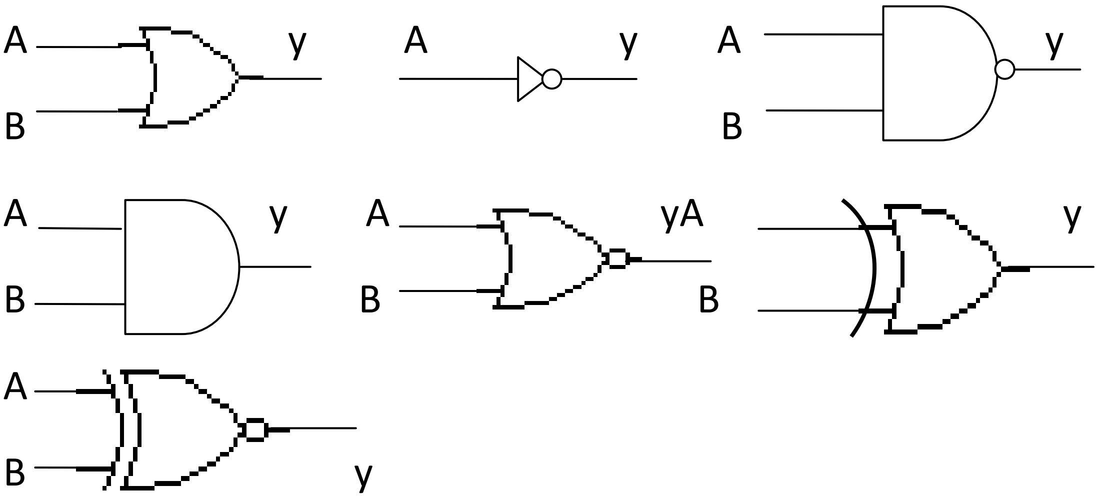
6.Draw logic circuit from truth table using k-map:

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	0	0	1
$\overline{A}B$	0	1	1	0
AB	0	1	1	0
$A\overline{B}$	1	0	0	1

7.a:Use De-Morgen theorem to simplify:

$\overline{A\,B\,(CD + E\,F)}$

b.Write the Boolean expression for each of these logic gates
(y = ……….):



8-a:Draw logic circuit using NOR gate only:

$\overline{\overline{A}\ \overline{B}\ \overline{C}(\ D + \overline{E})}$

b.Find difference of:

$(F\ I\ B\ )_{16}\ -\ (7A3)_{16} =$

9-a:Convert octal number $(14367{,}12)_8$ to decimal number:

b.Find the sum of:

$(11D)_{16} + (2E1)_{16}\ =$

c.Convert octal number $(1600{,}524)_8$ to hex number.

10.Write Boolean expression from this k-map and draw the logic circuit.

	$\overline{C}\,\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\,\overline{B}$	1	1	0	1
$\overline{A}B$	0	0	0	0
AB	1	1	1	1
$A\overline{B}$	1	1	1	1

11. Write Boolean expression from this k-map and draw the logic circuit.

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	0	0
$\bar{A}B$	1	0	0	1
AB	0	1	1	1
$A\bar{B}$	0	1	0	0

12. Use Boolean algebra to simplify the following expression then draw a logic circuit from simplified expression.

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

13. Draw a logic circuit from this function.

$A\ \bar{B} + \bar{C}\ (A + B)$

14. Simplify using Boolean algebra rolls:

a. $A\ B + (\ \bar{A} + \bar{B}\ )\ C + A\ B$

b. $\bar{A}\ B + \bar{A}\ B\ \bar{C} + \bar{A}\ B\ C\ D + \bar{A}\ B\ \bar{C}\ \bar{D}\ E$

15. Implement logic circuit from following Boolean expression using NAND gates only:

$Y = (\ \bar{A} + \bar{B})\ C + (\bar{D} + \bar{E})\ F$

16.Implement logic circuit from following Boolean expression using NOR gates only:

$$Y = (\overline{A}\,\overline{B} + C) . (\overline{D}\,\overline{E} + F)$$

17.What are the logic input values (A,B C_{in}) of full-adder when give the following output logic values:

a. S =0 : c = 0

b. S = 1 : c = 0

c. S =1 : c = 1

d. S = 0 : c = 1

18.What are the logic input values (A,B,B_{in}) of full- subtractor which gives the following output logic values:

a. D =0 : B_0 = 0

b. D = 1 : B_0 = 0

c. D =1 : B_0 = 1

d. D= 0 : B_0 = 1

19.Using k-map simplify the following Boolean expression:

$$M = A\,B\,C\,\overline{D} + \overline{A}\,B\,\overline{C}\,D + A\,\overline{B}\,\overline{C}\,D + A\,B\,\overline{C}\,D + \overline{A}\,B\,C\,\overline{D} +$$
$$A\,B\,C\,\overline{D} + A\,\overline{B}\,C\,\overline{D}$$

20.Using k-map, simplify the following Boolean expression:

$$N = \overline{A}\,B\,\overline{C}\,D + \overline{A}\,B\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,\overline{D} + A\,B\,\overline{C}\,D + A\,\overline{B}\,\overline{C}\,D$$