DIGITAL LOGIC DESIGN

NUMBER SYSTEM

Presented by

Mr. K.M.D.Rajesh babu,

Assistant Professor,

Department of ECE, Vemu Institute of Technology

MODULE-III
NUMBERSYSTEMS

Number systems: Complements of Numbers, Codes- Weighted and Non-weighted codes and its Properties, Parity check code and Hamming code.

Boolean Algebra: Basic Theorems and Properties, Switching Functions- Canonical and Standard Form, Algebraic Simplification, Digital Logic Gates, EX-OR gates, Universal Gates, Multilevel NAND/NOR realizations

- Number System is a way to represent the numbers in the computer architecture. There are four different types of the number system, such as:

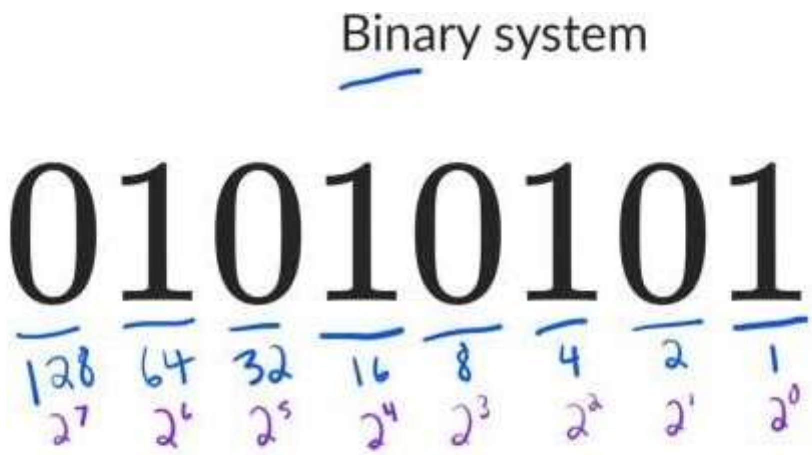
  - Binary number system (base 2)
  - Octal number system (base 8)
  - Decimal number system(base 10)
  - Hexadecimal number system (base 16).

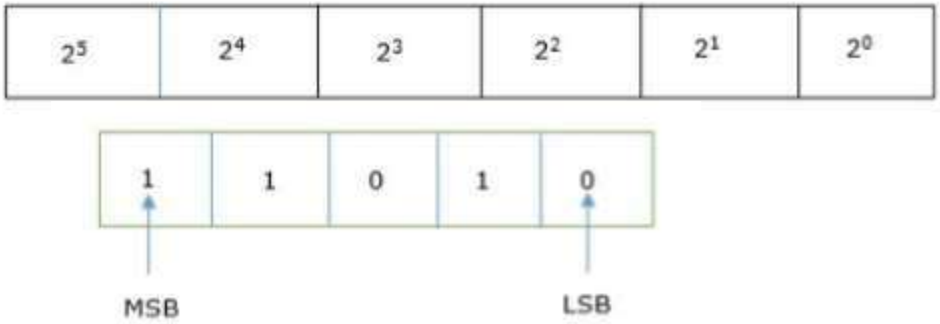
BINARY	OCTAL	DECIMAL	HEXA DECIMAL
Has 2 symbols	Has 8 symbols	Has 10symbols	Has 16 symbols
Symbols are 0,1	Symbols are 0,1,2,3,4,5,6 and7	Symbols are from 0-9	Symbols are from 0-9 and A,B,C,D,E,F
BIT position value system	Position value system	Position value system	Position value system
Value expressed in base of 2	Value expressed in base of 8	Value expressed in base of 10	Value expressed in base of 16
$(101010)_2$	$(765)_8$	$(925)_{10}$	$(F2F1)_{16}$

REPRESENTATION OF NUMBERS: Binary

Facts to Remember:

- Binary numbers are made up of only 0’s and 1’s.
- A binary number is represented with a base-2
- A bit is a single binary digit.





REPRESENTATION OF NUMBERS: Octal

$$etc \ldots\ldots\ldots 8^3 \quad 8^2 \quad 8^1 \quad 8^0 \;.\; 8^{-1} \quad 8^{-2} \quad 8^{-3} \ldots\ldots\ldots etc$$

Octal Point

Octal Numbering System (base 8)

Characters = 0,1,2,3,4,5,6,7

4	3	7
64's place	8's place	1's place

$= 4x64 + 3x8 + 7x1$

written 437_o or 437_8

7	1	2	6	3
8^4	8^3	8^2	8^1	8^0

decimal:

$3 \times 8^0 = 3$

$6 \times 8^1 = 48$

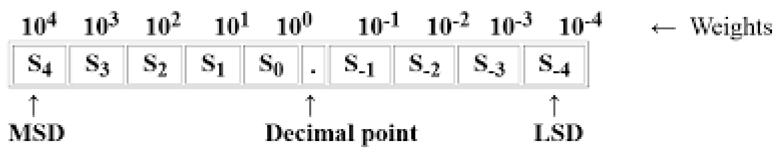
$2 \times 8^2 = 128$

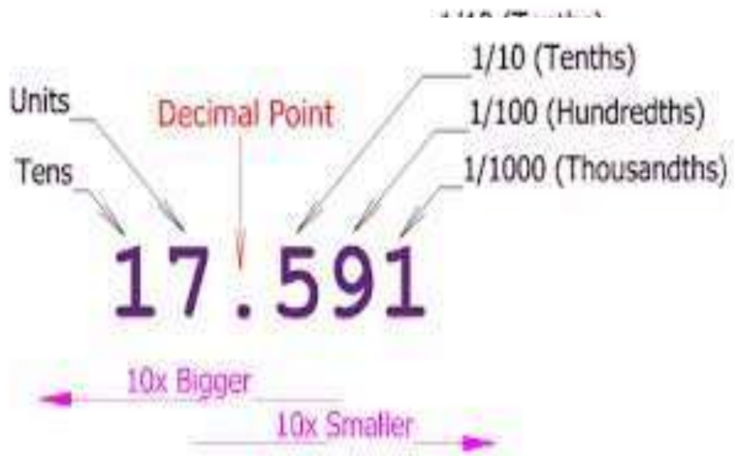
$1 \times 8^3 = 512$

$7 \times 8^4 = 28672$

29363

REPRESENTATION OF NUMBERS: Decimal

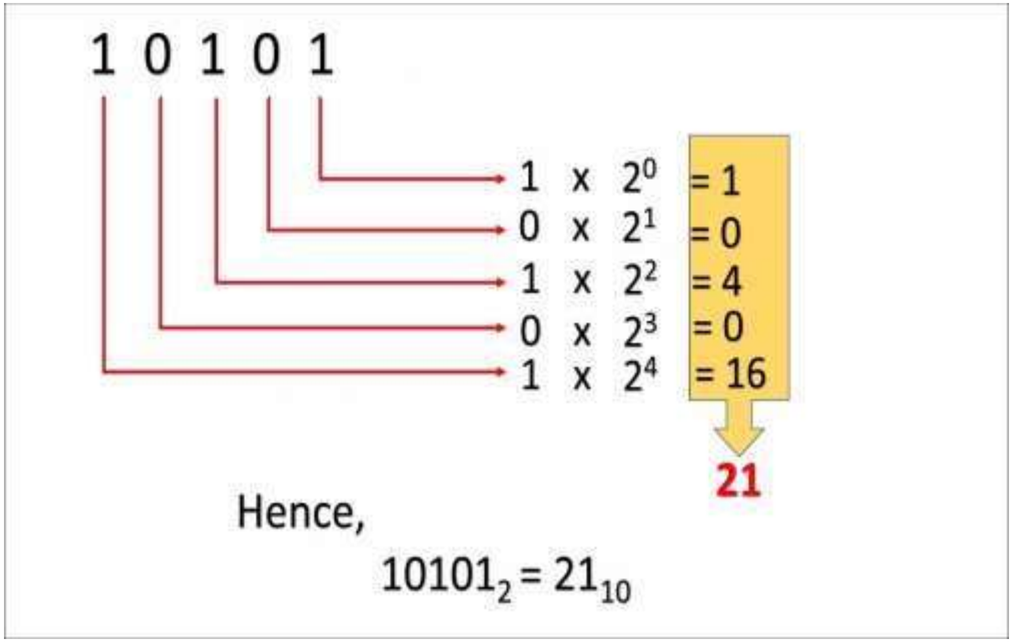


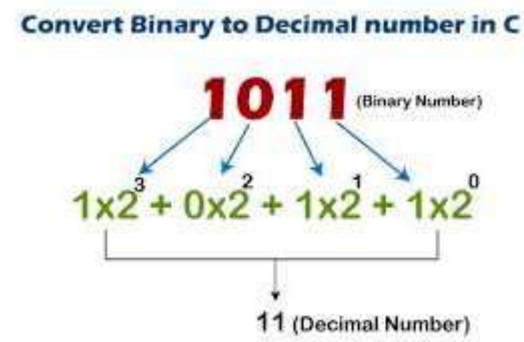


REPRESENTATION OF NUMBERS: Hexa decimal

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

Conversion between Numbers: Binary to Decimal





Find the equivalent decimal number for binary 1010_2

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

10

Conversion between Numbers: Octal to Decimal

2	3	7	.	4	5
8^2	8^1	8^0	Octal Point	8^{-1}	8^{-2}

2x64 + 3x8 + 7x1 (4x1/8) + (5x1/64)

128 + 24 + 7 0.5 + 0.078125

= 159 = 0.578125

Add

= 159.578125

$\boxed{\text{Decimal No.}}$

$(1\ 7\ 2)_8$ → Octal number

$8^0 * 2 =$ 2

$8^1 * 7 =$ 56

$8^2 * 1 =$ 64

$$(172)_8 = (2+56+64)_{10}$$
$$(172)_8 = (122)_{10}$$

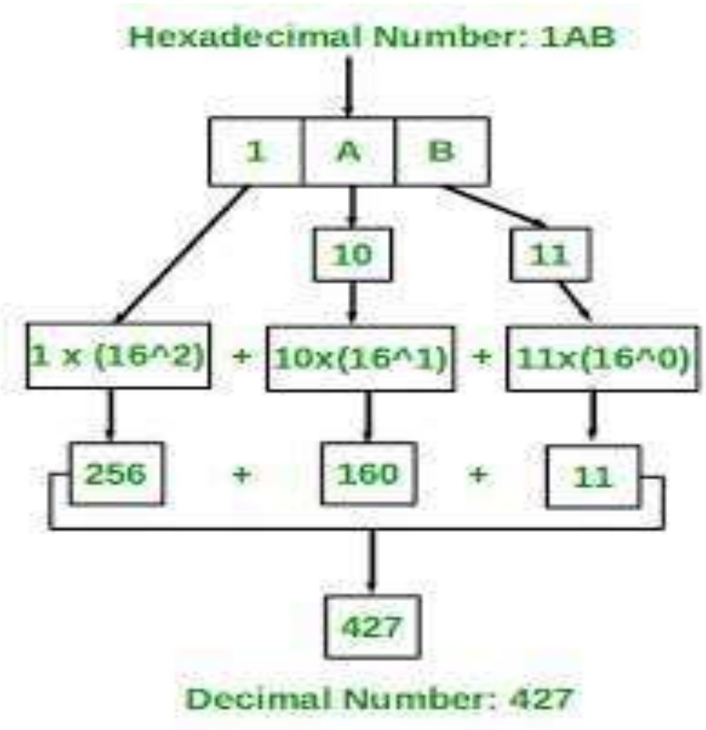
Octal to Decimal Conversion

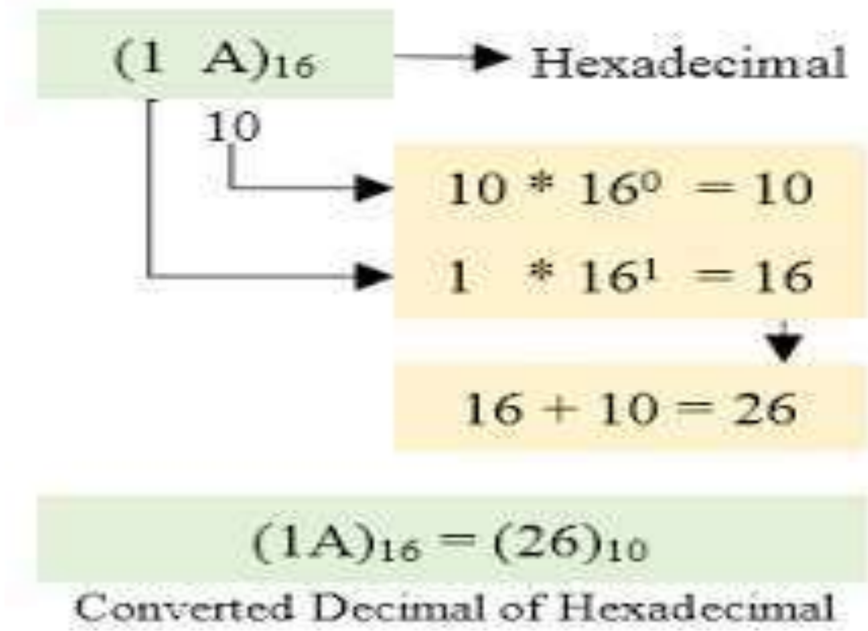
Find the equivalent decimal number for octal 143_8

Place values	8^2	8^1	8^0
Octal	1	4	3
Conversion	$1x8^2$	$4x8^1$	$3x8^0$
Decimal	64 +	32 +	3

99

Conversion between Numbers: Hexadecimal to Decimal





Conversion between Numbers: Decimal to Binary

Successive Division by 2

Division	Remainders	
2 | 29	1	LSB
2 | 14	0	
2 | 7	1	
2 | 3	1	
2 | 1	1	MSB
0		

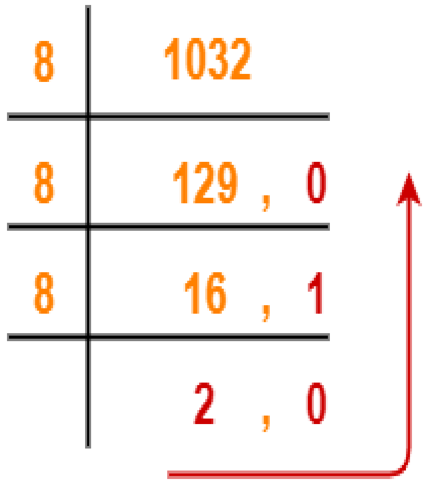
Read the remainders from the bottom up

29 decimal = 11101 binary

		MSB
$0.188 \times 2 = 0.376$	carry = 0	
$0.376 \times 2 = 0.752$	carry = 0	
$0.752 \times 2 = 1.504$	carry = 1	
$0.504 \times 2 = 1.008$	carry = 1	
$0.008 \times 2 = 0.016$	carry = 0	

Answer = .00110 (for five significant digits)

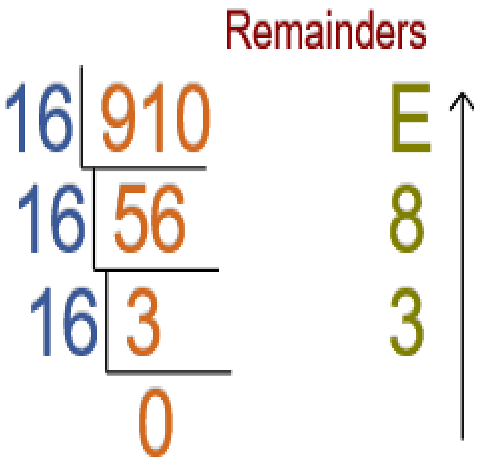
Conversion between Numbers: Decimal to Octal



Multiplication	Result	Integer Portion	Fraction Portion
0.45 x 8	3.60	3	.60
0.60 x 8	4.80	4	.80
0.80 x 8	6.40	6	.40
0.40 x 8	3.20	3	.20
0.20 x 8	1.60	1	.60

$(.45)_{10} = (.34631..)_8$

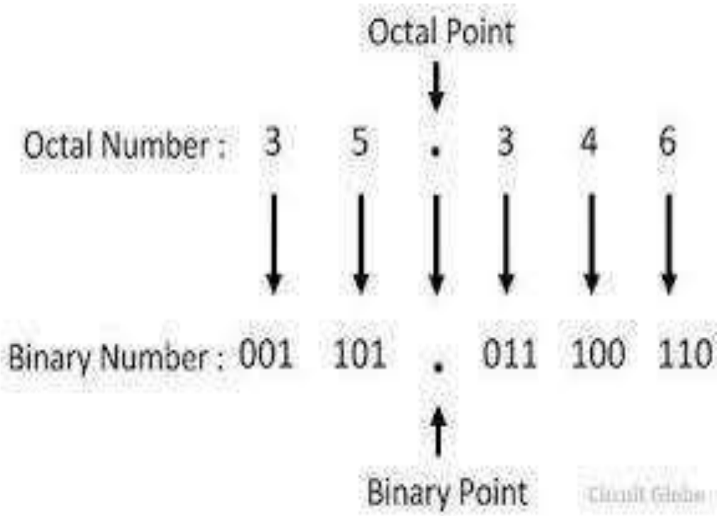
Conversion between Numbers: Decimal to Hexadecimal



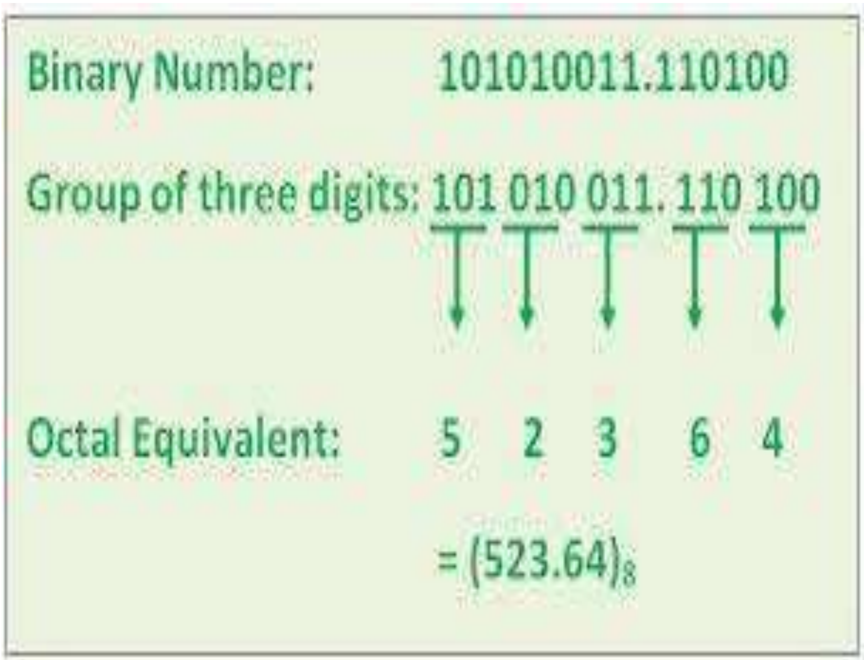
Multiplication	Result	Integer Portion	Fraction Portion
0.85 x 16	13.60	13(D)	.60
0.60 x 16	9.60	9	.60
0.60 x 16	9.60	9	.60

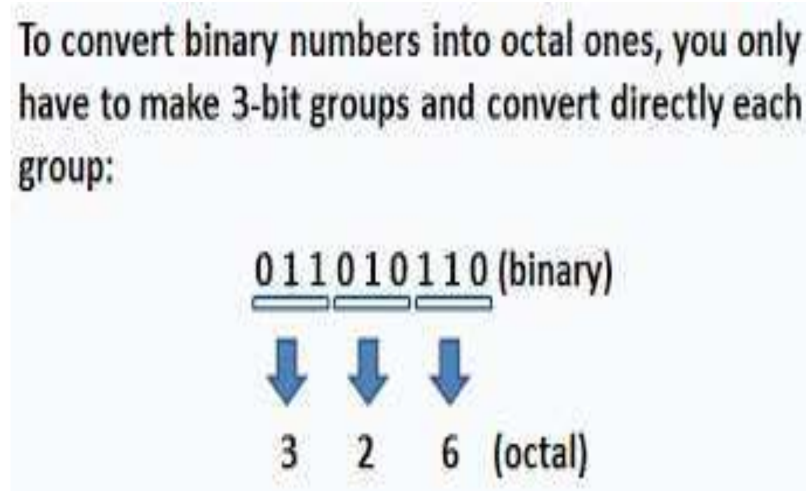
$(.85)_{10} = (.D99...)_{16}$

Octal to Binary

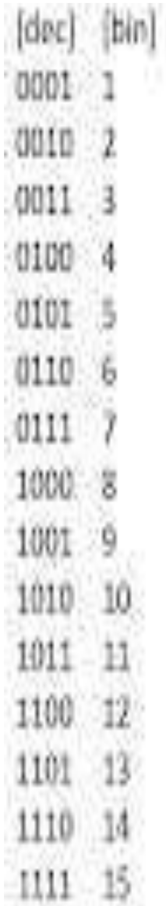


Binary to Octal



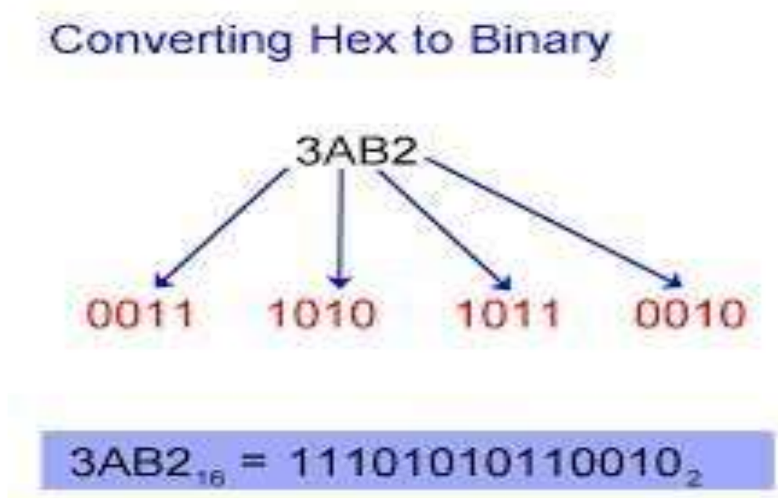


Hexa to Binary

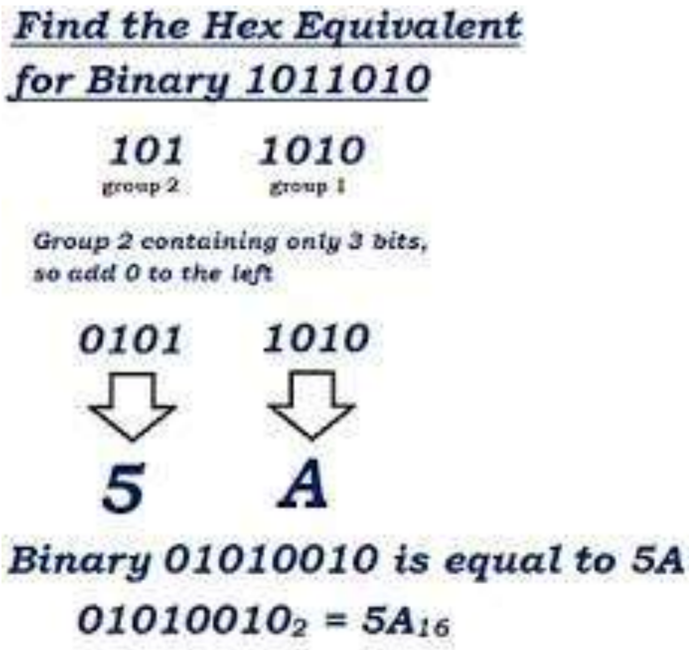


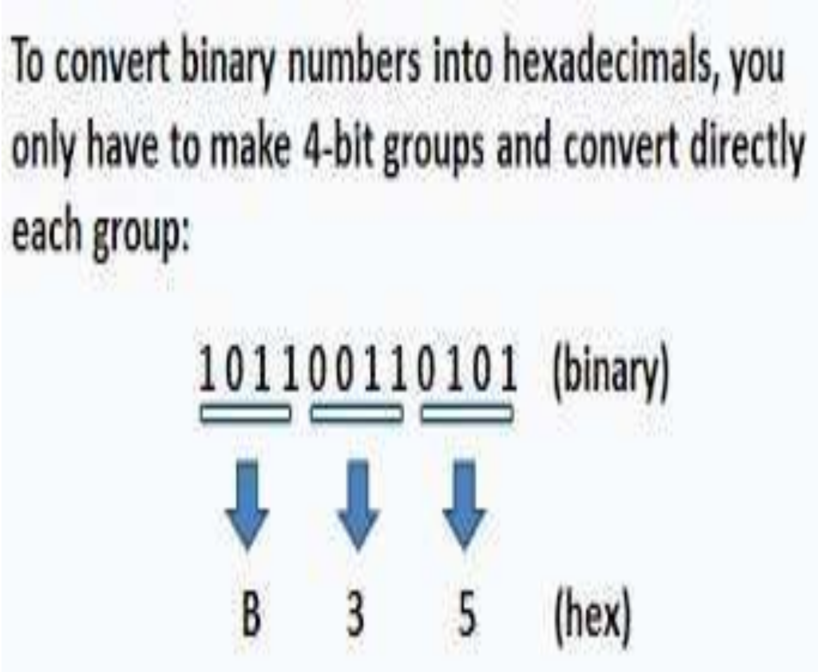
[dec]	[bin]
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

[hex]	[dec]
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

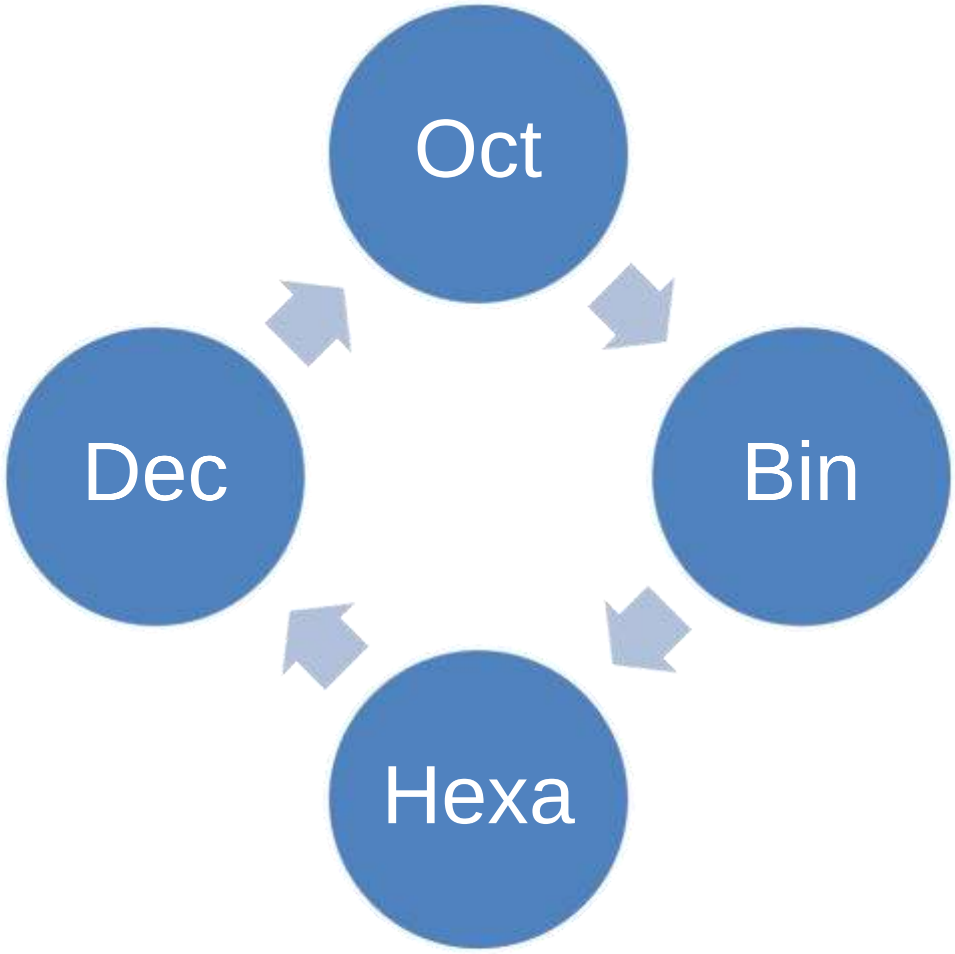


Binary to Hexa

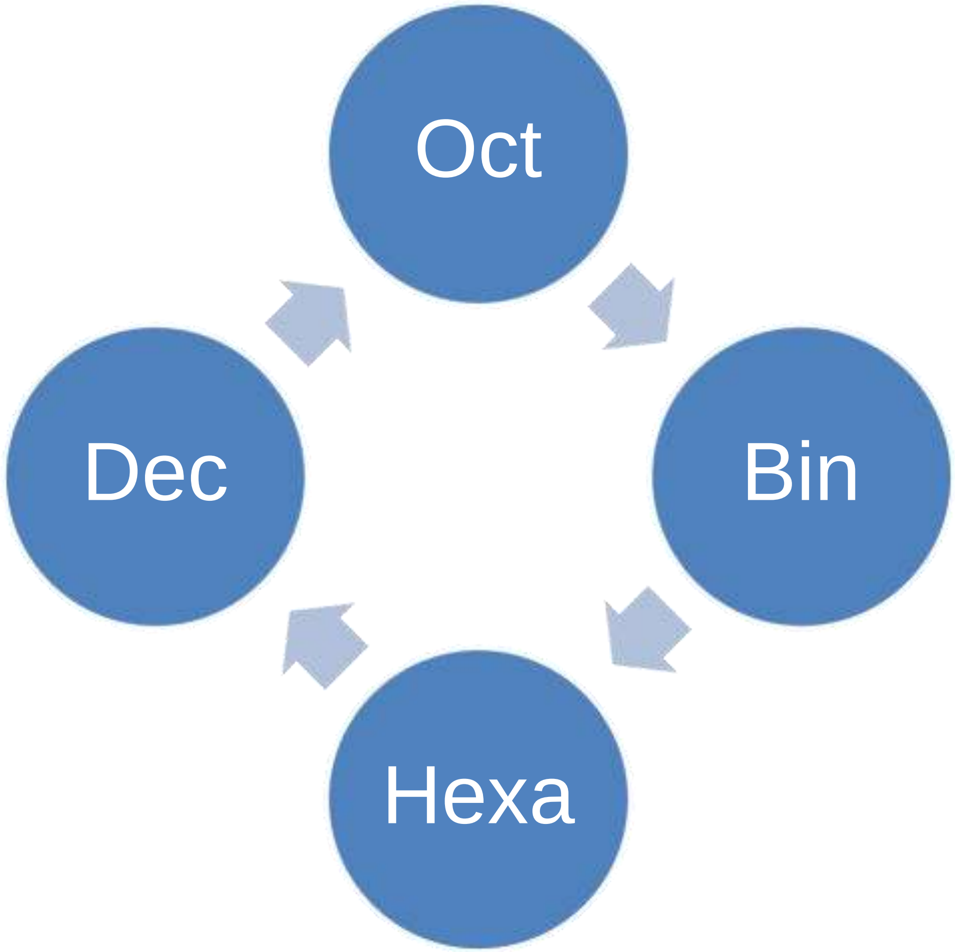




Number Conversion chart



Number Conversion chart



Complements

r's Complements

Ex: 2's and 10's .

r's compliment of N is defined as

$$r^n - N$$

r-1 's complements

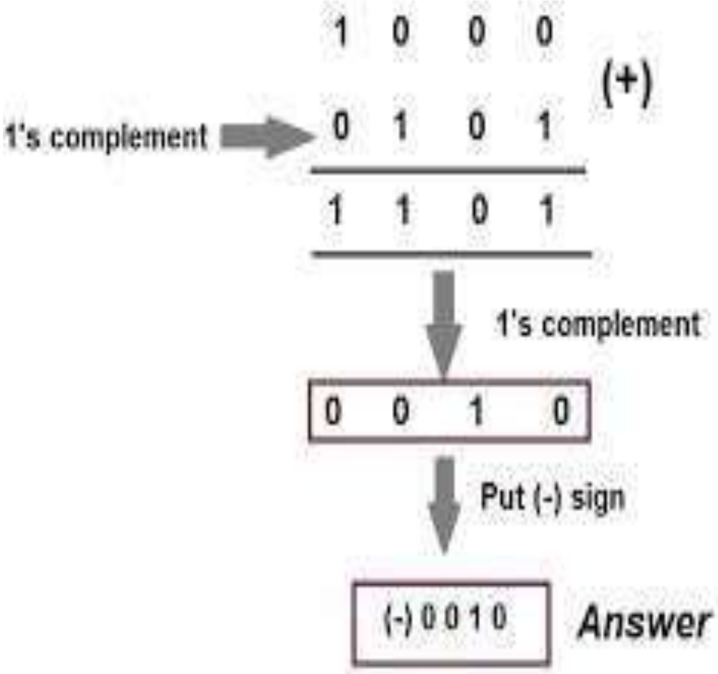
Ex: 1's and 9's

(r-1)'s compliment of N is defined as

$$(r^n - 1) - N$$

Subtraction using 1's Complement

1. Take the minuend as it is and 1'c of subtrahend
2. Add the 1'C of subtrahend to minuend
3. If carry come in MSB remove the carry and add it to the 'sum' to get result
4. If carry does not come in MSB 1'C of 'sum' is the result



Subtraction using 1's Complement

1. Take the minuend as it is and 1'c of subtrahend
2. Add the 1'C of subtrahend to minuend
3. If carry come in MSB remove the carry and add it to the 'sum' to get result
4. If carry does not come in MSB 1'C of 'sum' is the result

```
    1 1 1 1
+
    0 1 0 1
   ---------
Carry (1) 0 1 0 0
   +        1
   ---------
  | 0 1 0 1 | Answer
```

```
          1 1 1 1
      +
          0 1 0 1
        ---------
Carry   1 0 1 0 0
           +     1
        ---------
          0 1 0 1   Answer
```

Subtraction using 2's Complement

1. Take the minuend as it is and 2'c of subtrahend
2. Add the 2'C of subtrahend to minuend
3. If carry come in MSB discord the end carry and remaining value is the result
4. If carry does not come in MSB 2'C of 'sum' is the result

Given the two binary numbers $X = 1010100$ and $Y = 1000011$, perform the subtraction (a) $X - Y$ and (b) $Y - X$ using 2's complements.

(a)

$X =$	1010100
2's complement of $Y =$	$+\ \underline{0111101}$
Sum $=$	10010001
Discard end carry $2^7 =$	$-\underline{10000000}$
Answer: $X - Y =$	0010001

Subtraction using 2's Complement

1. Take the minuend as it is and 2'c of subtrahend
2. Add the 2'C of subtrahend to minuend
3. If carry come in MSB discord the end carry and remaining value is the result
4. If carry does not come in MSB 2'C of 'sum' is the result

$$
\begin{array}{rr}
Y = & 1000011 \\
\text{2's complement of } X = & +\ \underline{0101100} \\
\text{Sum} = & 1101111
\end{array}
$$

There is no end carry.

Answer: $Y - X = -(\text{2's complement of } 1101111) = -0010001$

Using 10's complement, subtract 72532 − 3250.

$$
\begin{aligned}
M &= 72532 \\
\text{10's complement of } N &= +\ \underline{96750} \\
\text{Sum} &= 169282 \\
\text{Discard end carry } 10^5 &= -\underline{100000} \\
\text{Answer} &= 69282
\end{aligned}
$$

ANALOG AND DIGITAL ELECTRONICS

BINARY CODES

1. The main characteristic of a weighted code is, each binary bit is assigned by a “weight” and values depend on the position of the binary bit.

2. The sum of the weights of these binary bits, whose value is 1 is equal to the decimal digit which they represent.

3. In other words, if w1, w2, w3 and w4 are the weights of the binary digits, and x1, x2, x3 and x4 are the corresponding bit values, then the decimal digit N=w4x4 + w3x3+w2x2+w1x1 is represented by the binary sequence x4x3x2x1.

- Two types binary codes

  1.Weighted Binary Systems and

  2.Non Weighted Codes.

- Weighted binary codes are those which follow the positional weighting principles wherein each position of the number represents a specific weight.

  Ex: BCD(8421), 84-2-1, 2421, and 5043210 …

- Non-weighted codes are codes that are not placed weighted. It means that each position within the binary number is not assigned a fixed value.

  Ex: Excess-3 and Gray codes

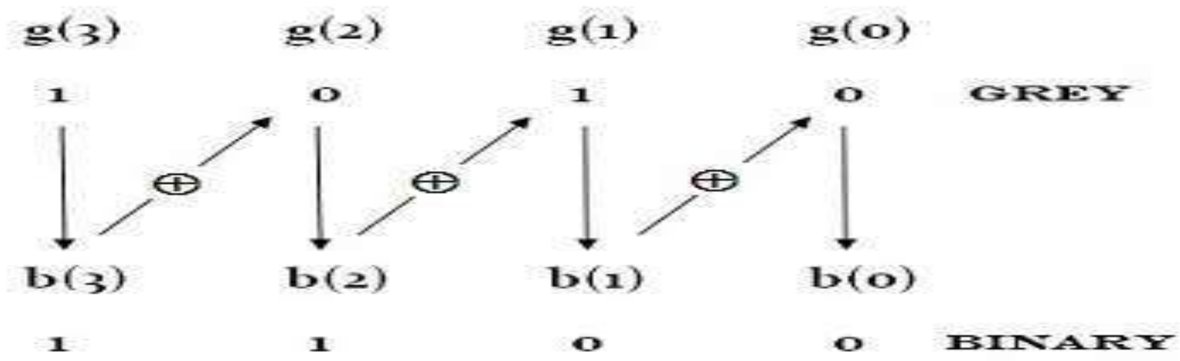
Binary codes for the decimal digits

Decimal digit	(BCD) 8421	Excess-3	84-2-1	2421	(Biquinary) 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

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

Grey Code to Binary Conversion

Convert the Grey code 1010 to its equivalent Binary



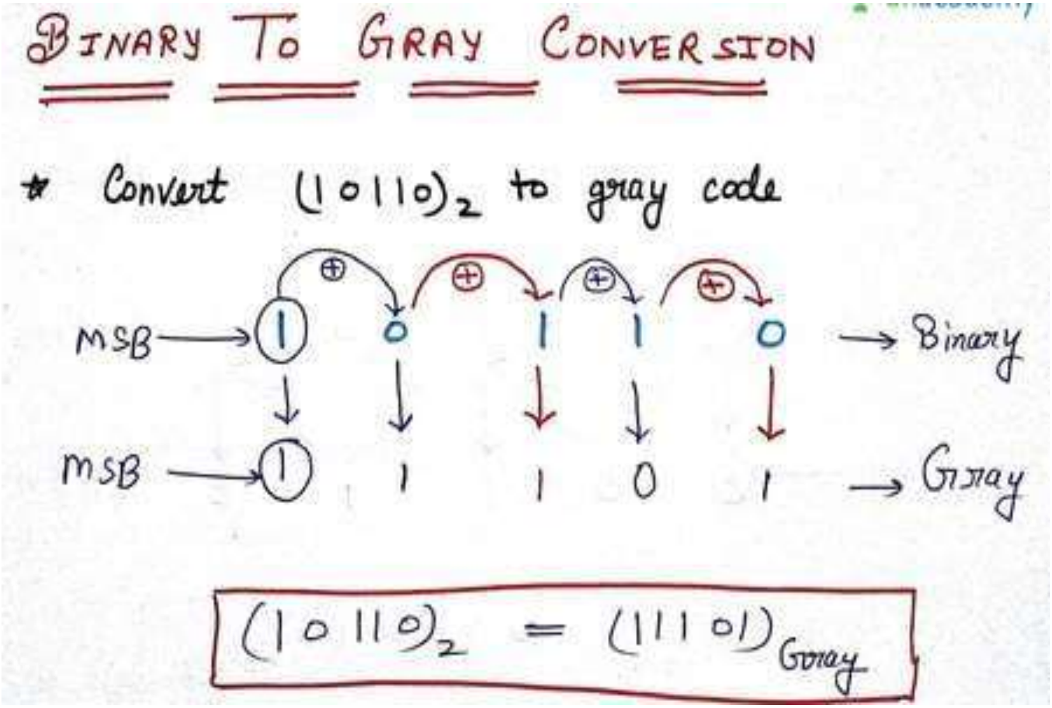
i.e

$$b(3) = g(3)$$

$$b(2) = b(3) \oplus g(2)$$

$$b(1) = b(2) \oplus g(1)$$

$$b(0) = b(1) \oplus g(0)$$


BINARY TO GRAY CONVERSION
Convert (10110)2 to gray code
MSB
1 0 1 1 0
Binary
MSB
1 1 1 0 1
Gray
(10110)2 = (11101)Gray

Decimal	BCD	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

DIGITAL LOGIC DESIGN

PARITY CHECK CODE AND HAMMING CODE.

Presented by

Mr. K.M.D.Rajesh babu,

Assistant Professor,

Department of ECE, Vemu institute of Technology

Parity check code

- Parity check is a simple way to add redundancy bits to the packets such that the total number of 1's is even (or odd).
- A single bit is appended to the end of each frame, the bit is 1 if the data portion of the frame has odd number of 1's. Otherwise, it is 0.
- The total number of 1's in each data frame is always even.
- The problem with this approach is that if there are even number of errors, it can not be detected
- Therefore it has a one bit *error detection* capability but no *error correction* capability.

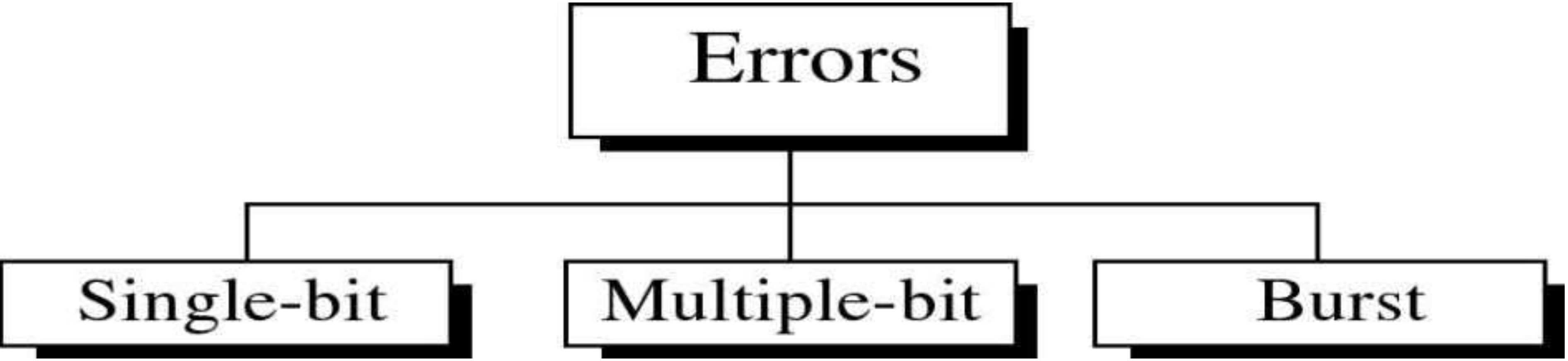
Networks must be able to transfer data from one device to another with complete accuracy.

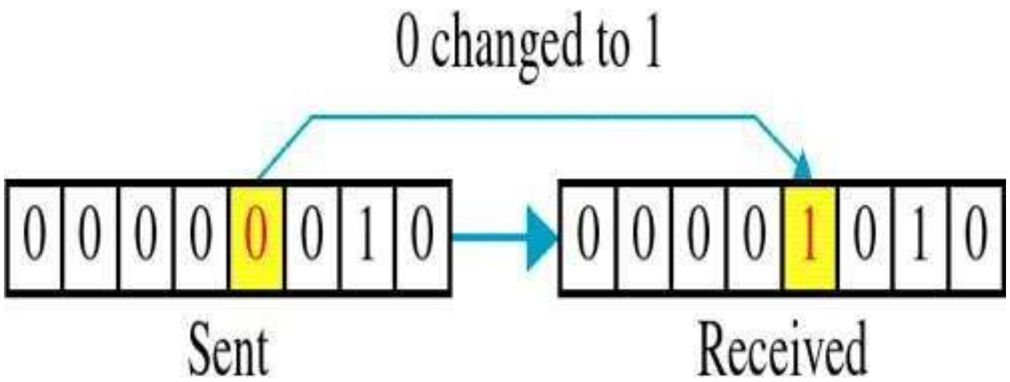
★ Data can be corrupted during transmission.

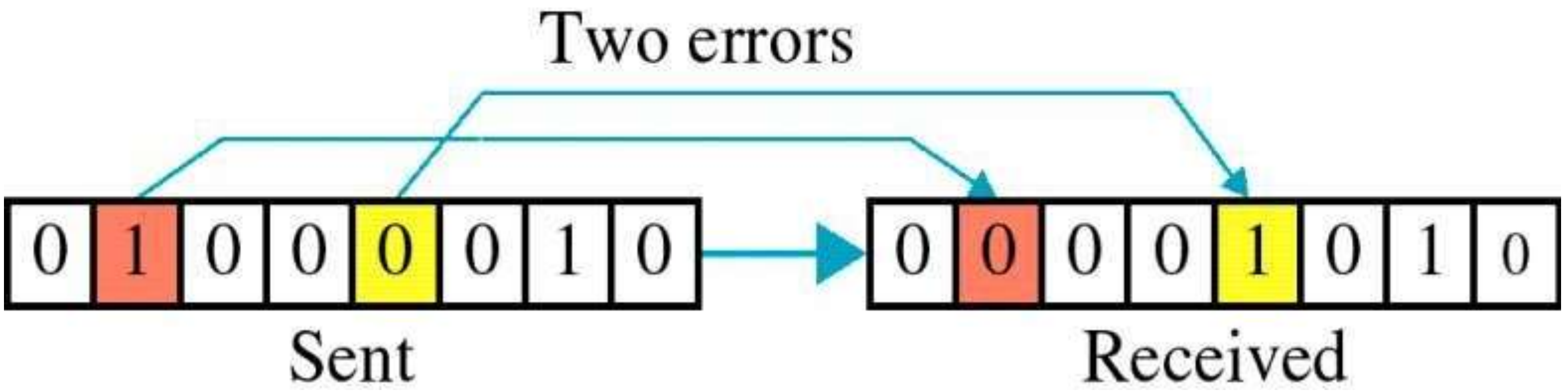
★ For reliable communication, errors must be detected and corrected.

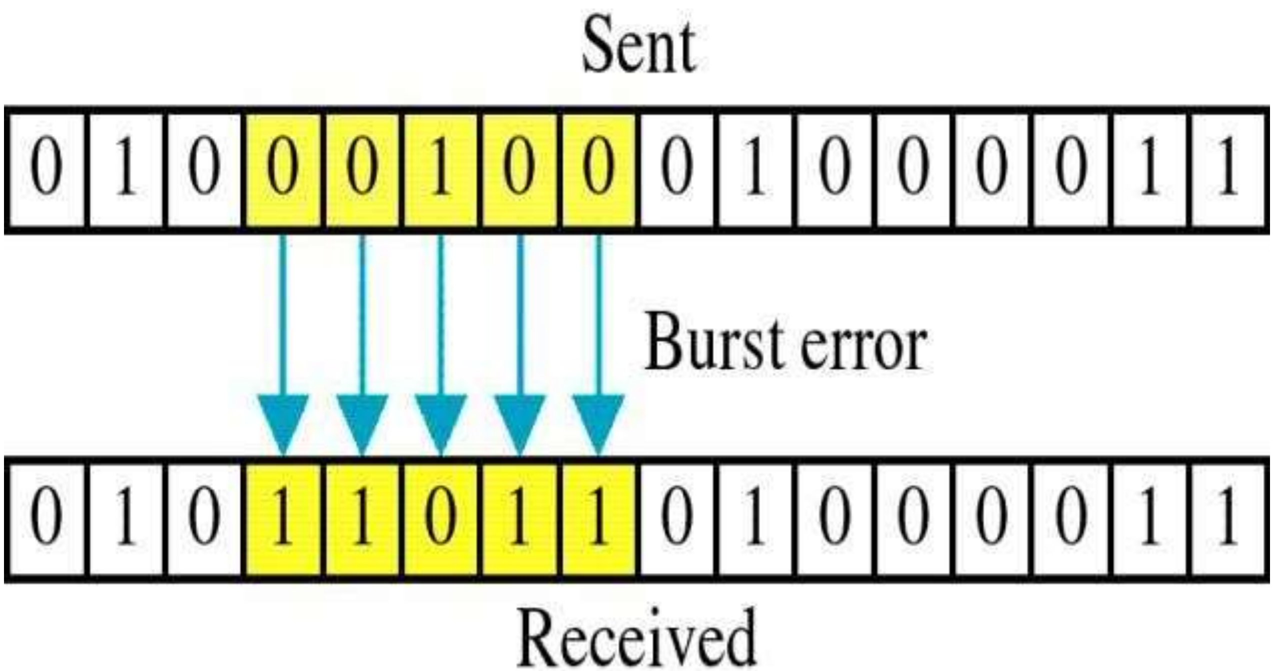
★ **Error detection and correction** are implemented either at the **data link layer** or the **transport layer** of the OSI model.

Types of errors






Two errors
0 1 0 0 0 0 1 0
Sent
0 0 0 0 1 0 1 0
Received


Sent
0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1
Burst error
0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 1
Received

Parity bits

Decimal no.	Message bits	Parity bits (even)	Parity bits (odd)
0	000	0	1
1	001	1	0
2	010	1	0
3	011	0	1
4	100	1	0
5	101	0	1
6	110	0	1
7	111	1	0

Hamming code

- It is an error detection and correction code
- Invented by Richard W. Hamming
- Steps involved in the Hamming code
  1. Selecting the number of redundant bits
  2. Choosing the location of redundant bits
  3. Assigning the values to redundant bits
  4. How to detect and correct the error in the hamming code?

- Selecting the number of parity bits

$$2^P \geq n + P + 1$$

For example msg bits n=4

Let p=2

$$2^2 \geq 4 + 2 + 1$$

4≥ 7(condition fail)

Let p=3

$$2^3 \geq 4 + 2 + 1$$

8≥8(condition true)

So select 3 parity bits for 4 bit message to create hamming code

2. Choosing the location of parity bits

Bit Location	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Bit designation	D4	D3	D2	**P3**	D1	**P2**	**P1**
Binary representation	111	110	101	100	011	010	001

3. Assigning the values to parity bits

Bit Location	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Bit designation	D4	D3	D2	**P3**	D1	**P2**	**P1**
Binary representation	111	110	101	100	011	010	001
(data bits)							
(parity bits)							

$$P_1 = 3\ xor\ 5\ \ xor 7$$

$$P_2 = 3\ xor\ 6\ \ xor 7$$

$$P_3 = 5\ xor\ 6\ \ xor 7$$

- **How to detect and correct the error in the hamming code?.**
- Given the 4 bit data word 1010, generate the 18 bit composite word for the hamming code that corrects and detects single errors.

Bit Location	7	6	5	4	3	2	1
Bit designation	D4	D3	D2	**P3**	D1	**P2**	**P1**
Binary representation	111	110	101	100	011	010	001
(data bits)							
(parity bits)							

DIGITAL LOGIC DESIGN

BOOLEAN ALGEBRA

Presented by

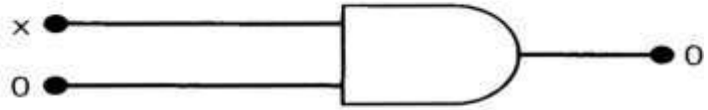
Mr. K.M.D.Rajesh babu,

Assistant Professor,

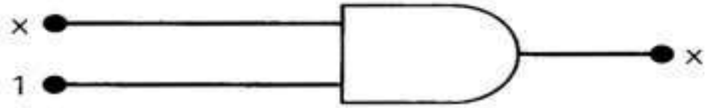
Department of ECE, Vemu Institute of Technology

Theorem Name	AND form	OR form
Identity	1.A=A	0+A=A
Null law	0.A=0	1+A=1
Idempotent law	A. A = A	A + A = A
Inverse law	A A' = 0	A + A' = 1
Commutative law	AB =BA	A+B =B+A
Associate law	(AB)C=A(BC)	(A+B)+C=A+(B+C)
Distributive law	A+BC = (A+B)(A+C)	A(B+C)= AB+AC
Absorption law	A(A+B)=A	A+AB= A
De Morgan's law	(AB)' = A' +B'	(A+B)' = A' B'

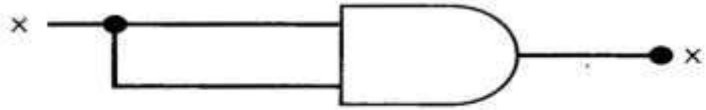
(1) $x \cdot 0 = 0$



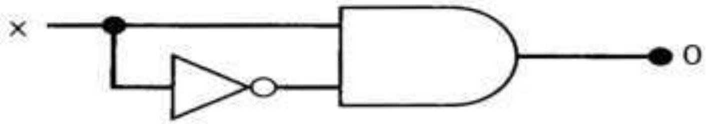
(2) $x \cdot 1 = x$



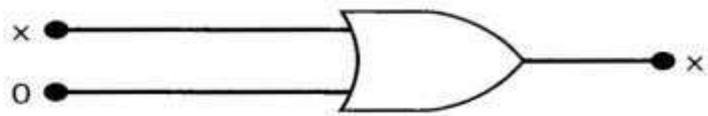
(3) $x \cdot x = x$



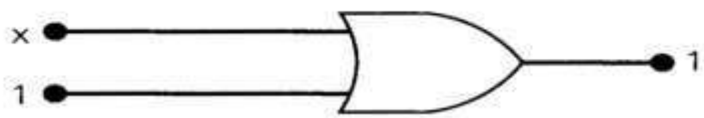
(4) $x \cdot \overline{x} = 0$



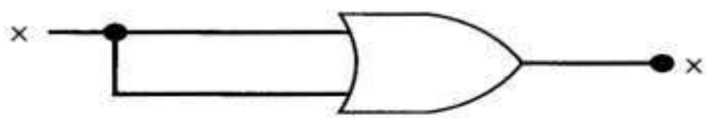
(5) $x + 0 = x$



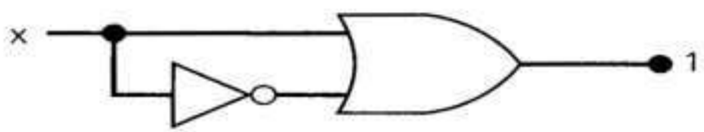
(6) $x + 1 = 1$



(7) $x + x = x$



(8) $x + \overline{x} = 1$



- **Consensus**

  $AB+A'C+BC=AB+A'C$

  $(A+B)\,(A'+C)\,(B+C)=(A+B)\,(A'+C)$

- **Transposition theorem**

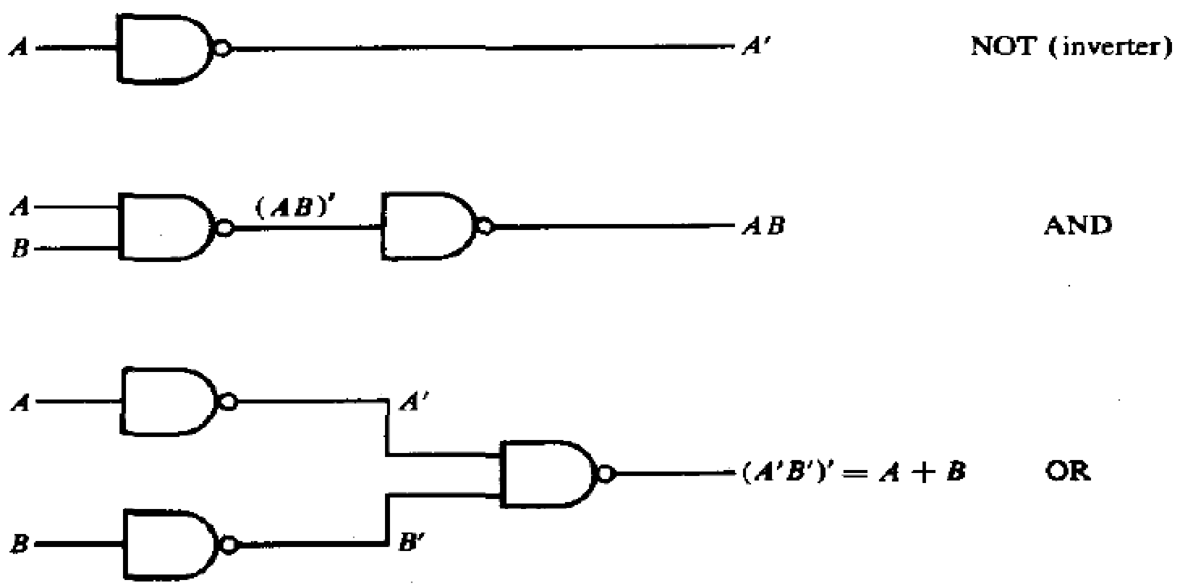
  $(A+B)(A+C)= A + BC$

- Simplify the expression $y = AB'D + AB'D'$

- Simplify the expression $X = ACD + A'BCD$

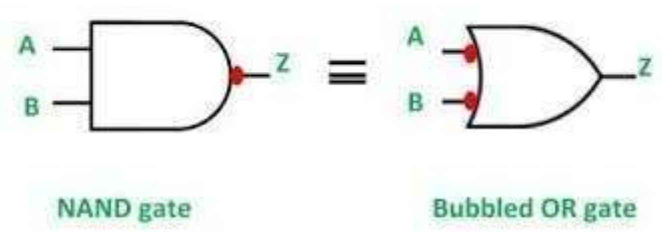
- Simplify $AB' + ABC' + AB'C'D = AB' + AC'$

- Combinational circuits are more frequently constructed with NAND or NOR gates rather than AND and OR gates.
- It is important to be able to recognize the relationships between AND-OR and NAND or NOR.
- NAND and NOR universal gates.



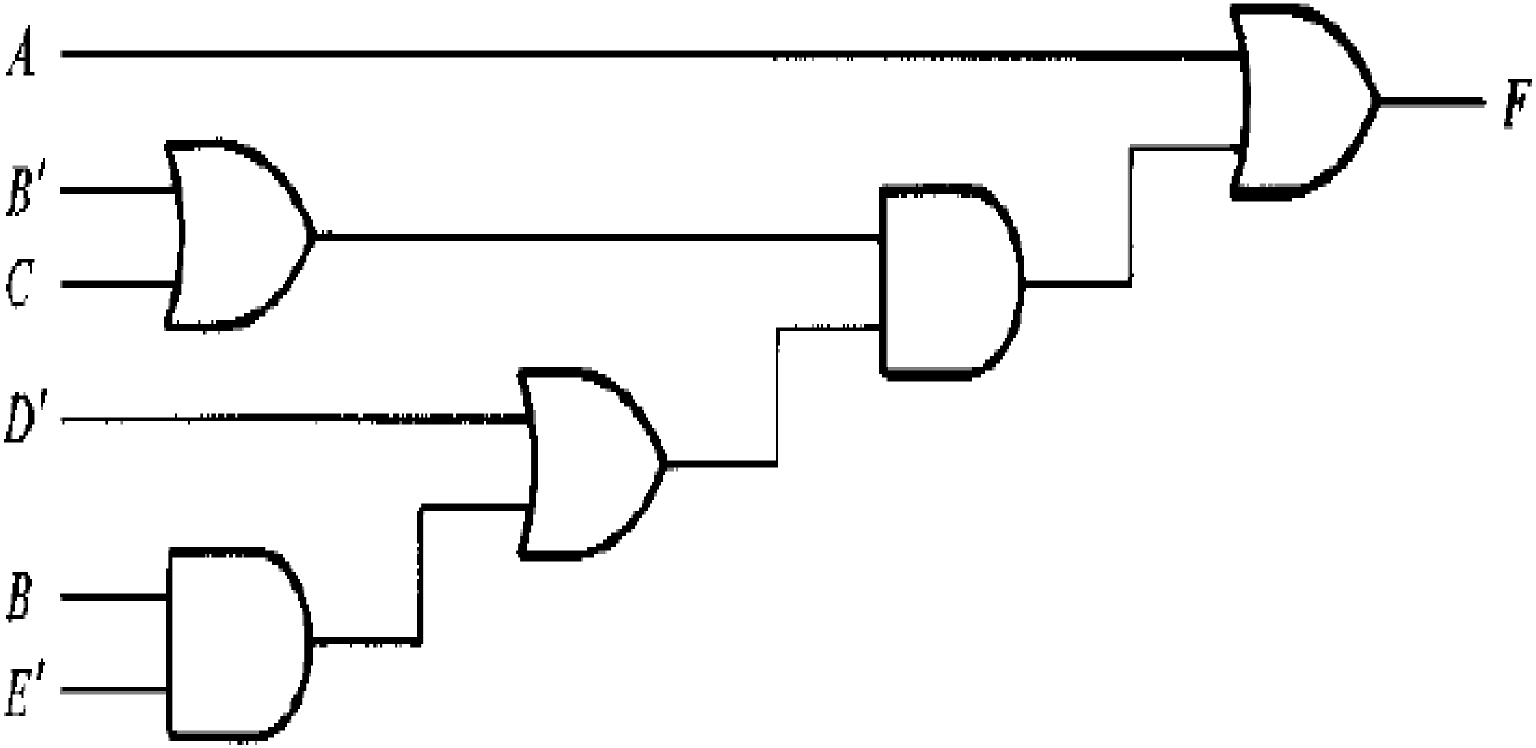
multilevel NAND diagram

- From the given Boolean expression, draw the logic diagram with AND, OR, and inverter gates. Assume that both the normal and complement inputs are available.
- Convert all AND gates to NAND gates with AND-invert graphic symbols.
- Convert all OR gates to NAND gates with invert-OR graphic symbols.
- Check all small circles in the diagram. For every small circle that is not compensated by another small circle along the same line, insert an inverter (one-input NAND gate) or complement the input variable.



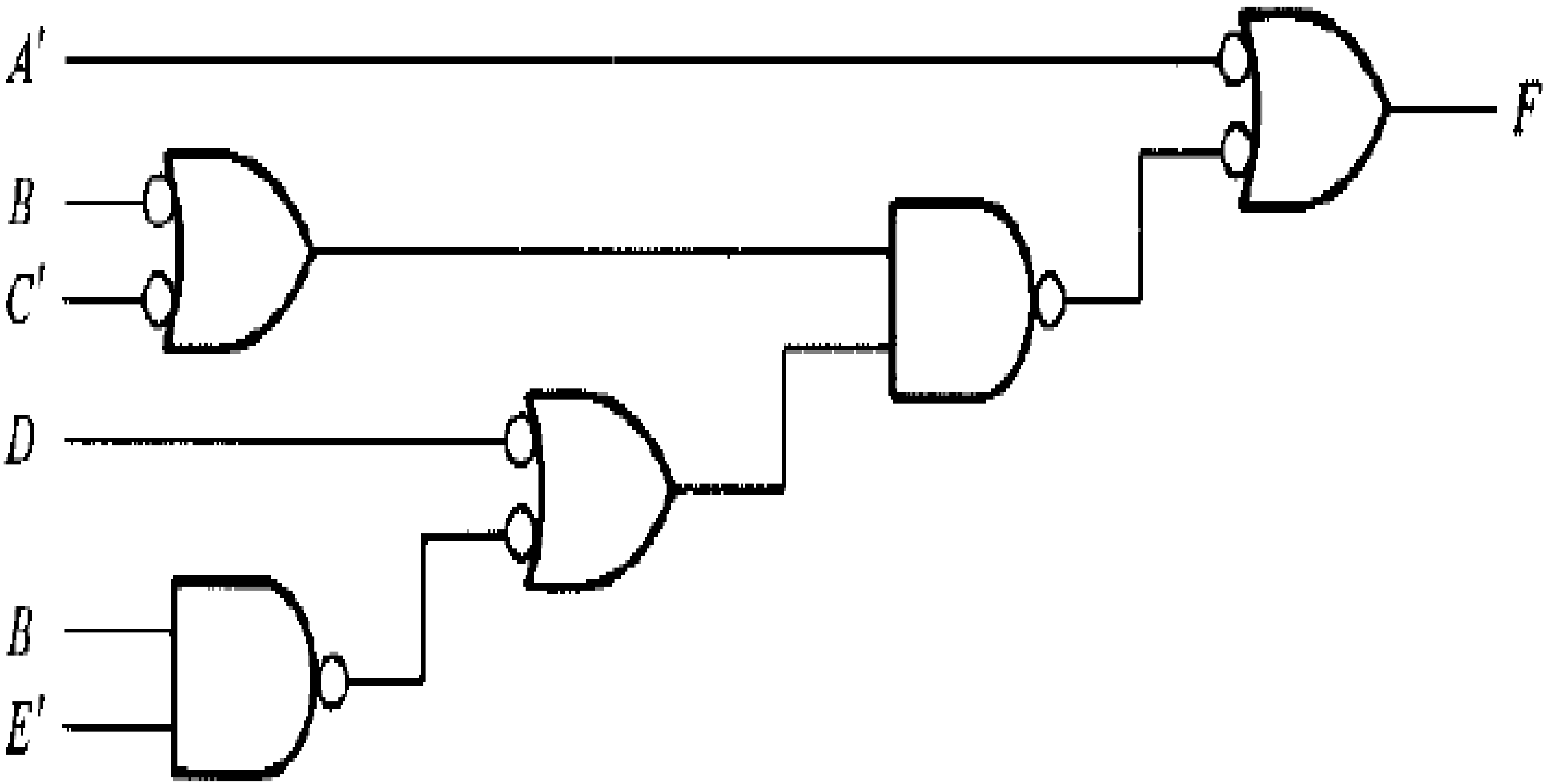
- $F = A + (B' + C)(D' + BE')$

- From the given Boolean expression, draw the logic diagram with AND, OR, and inverter gates

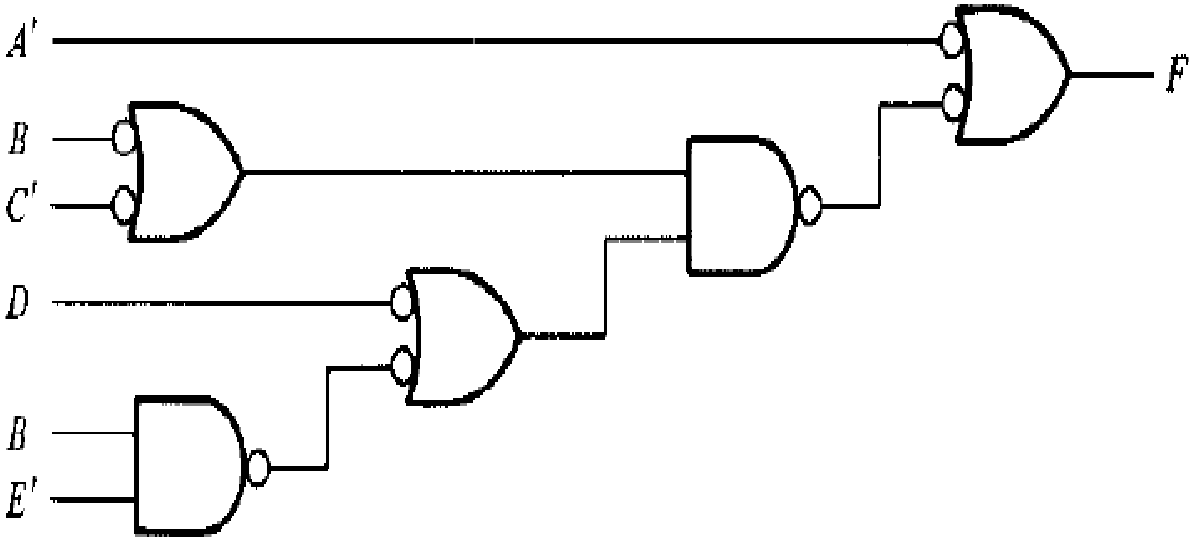


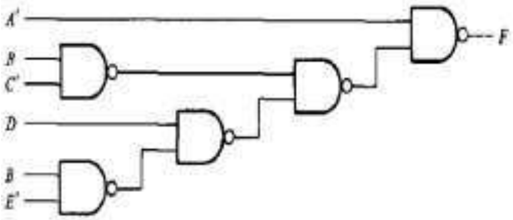
(a) AND-OR diagram

- Convert all AND gates to NAND gates with AND-invert graphic symbols.
- Convert all OR gates to NAND gates with invert-OR graphic symbols

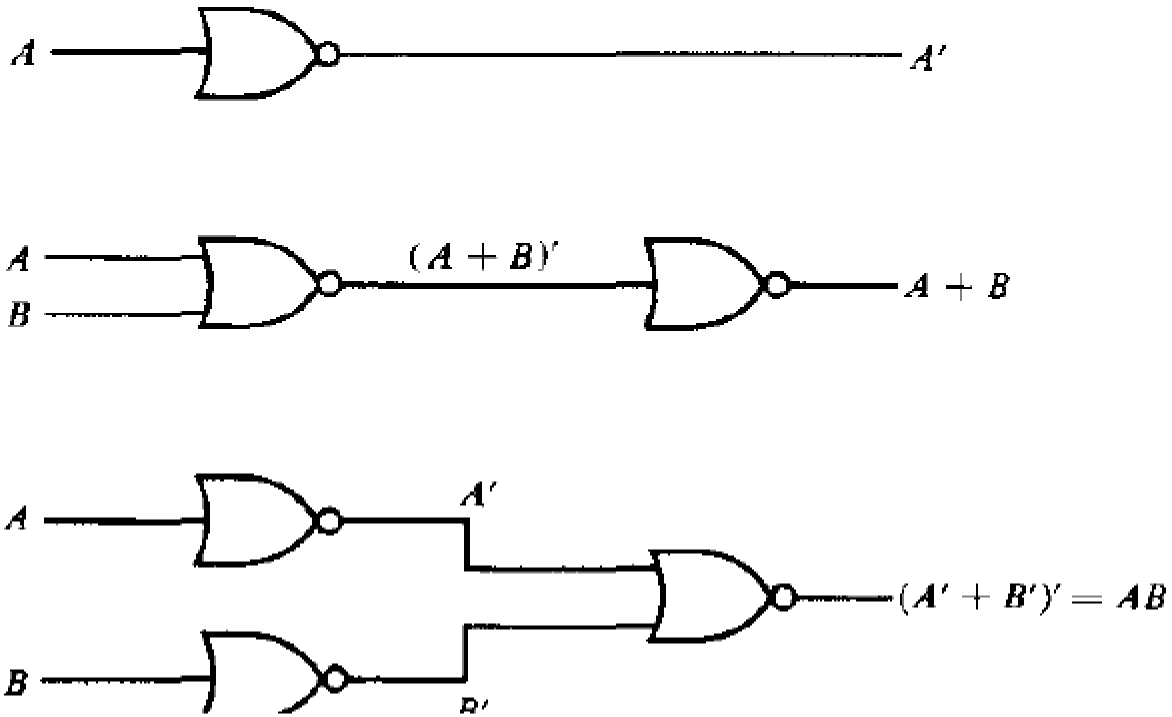


- NAND logic



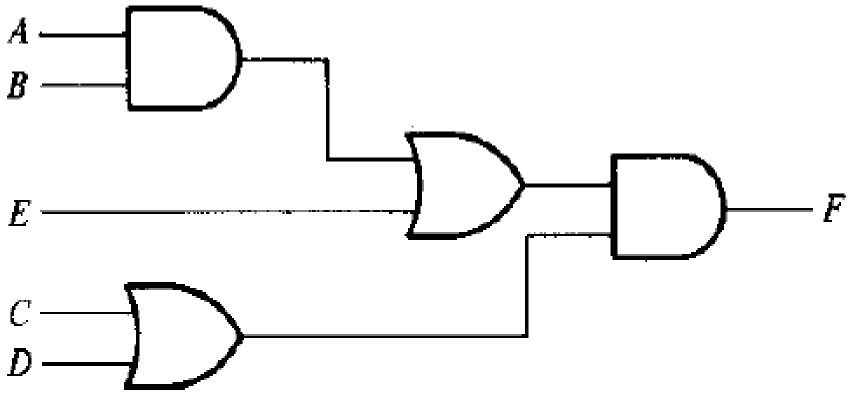


- MULTI LEVEL NOR CIRCUITS



- From the given Boolean expression, draw the logic diagram with AND, OR, and inverter gates. Assume that both the normal and complement inputs are available.
- Convert all AND gates to NAND gates with AND-invert graphic symbols.
- Convert all OR gates to NAND gates with invert-OR graphic symbols.
- Check all small circles in the diagram. For every small circle that is not compensated by another small circle along the same line, insert an inverter (one-input NAND gate) or complement the input variable.

- $F = (AB + E)(C + D)$



- **Karnaugh Map Method** - Up to five Variables, Don't Care Map Entries, Tabular Method.

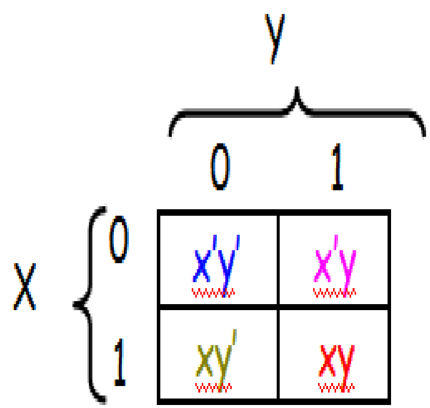
- **Combinational Logic Circuits:** Adders, Subtractors, comparators, Multiplexers, Demultiplexers, Encoders, Decoders and Code converters, Hazards and Hazard Free Relations.

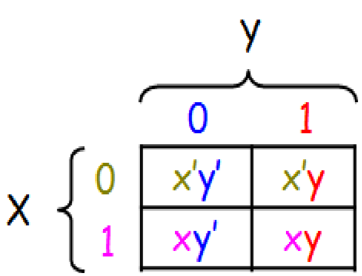
KARANAUGH MAP

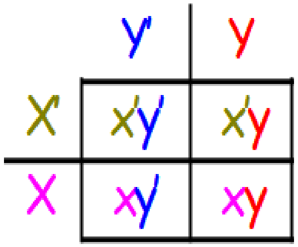
- Boolean functions may be simplified by algebraic theorems. However, this procedure of minimization is awkward.
- KARANAUGH Map is a simple straightforward procedure.
- The map is made up of squares. Each square represents one minterm
  - A two-variable function has four possible minterms. We can re-arrange these minterms into a Karnaugh map.

- By recognizing various patterns, the user can derive alternative algebraic expressions for the same function, from which he can select the simplest one. We shall assume that the simplest algebraic expression is anyone in a sum of products or product of sums that has a minimum number of literals.

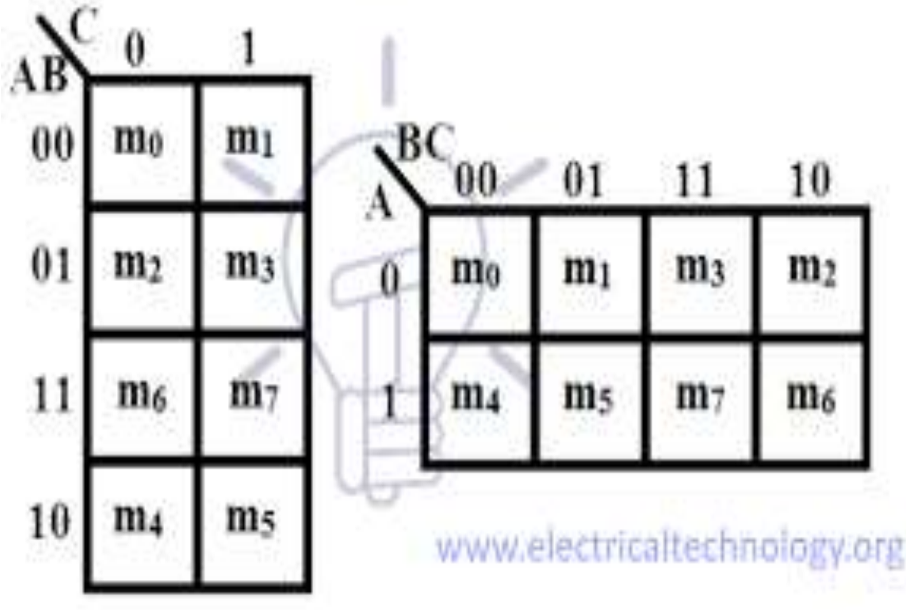
x	y	minterm
0	0	x'y'
0	1	x'y
1	0	xy'
1	1	xy

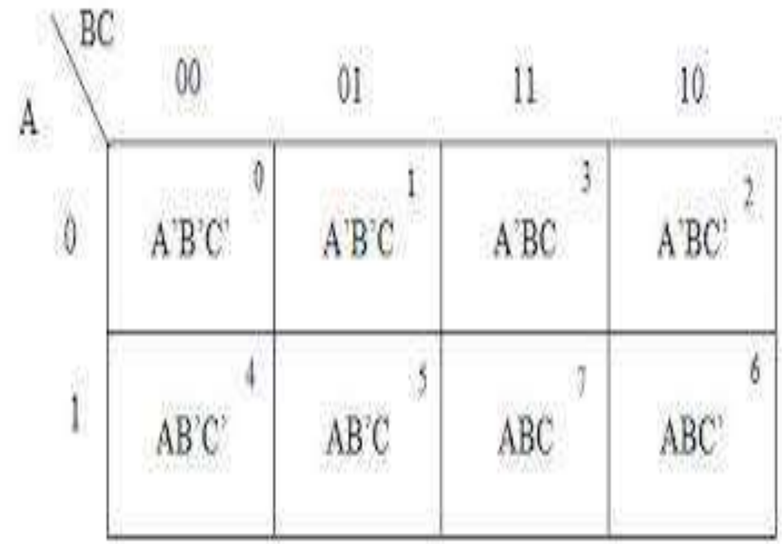






3 variable function





- One square represents one minterm, giving a term of three literals.
- Two adjacent squares represent a term of two literals.
- Four adjacent squares represent a term of one literal.
- Eight adjacent squares encompass the entire map and produce a function that is always equal to 1.

Simplify the Boolean function $F(x, y, z) = \sum (2, 3, 4, 5)$

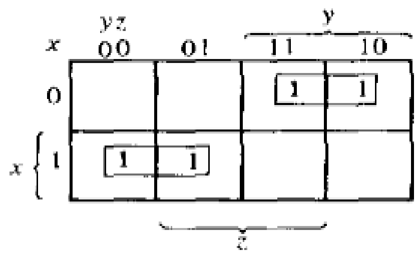


FIGURE 3-4
Map for Example 3-1; $F(x, y, z) = \Sigma (2, 3, 4, 5) = x'y + xy'$

Given the following Boolean function:

$$F = A'C + A'B + AB'C + BC$$

(a) Express it in sum of minterms,

(b) Find the minimal sum of products expression,

4-variable K-map

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

$w'x'y'z'$	$w'x'y'z$	$w'x'y\,z$	$w'x'y\,z'$
$w'x\,y'z'$	$w'x\,y'z$	$w'x\,y\,z$	$w'x\,y\,z'$
$w\,x\,y'z'$	$w\,x\,y'z$	$w\,x\,y\,z$	$w\,x\,y\,z'$
$w\,x'y'z'$	$w\,x'y'z$	$w\,x'\,y\,z$	$w\,x'\,y\,z'$

- 4 variable k-map has maximum of 16 minterms.
- Map has 16 square boxes
- Possibilities of adjacent squares

- One square represents one minterm, giving a term of four literals.
- Two adjacent squares represent a term of three literals.
- Four adjacent squares represent a term of two literals.
- Eight adjacent squares represent a term of one literal.
- Sixteen adjacent squares represent the function equal to 1.

- Simplify the Boolean function

$F(w, x, y, z) = \sum(0, 1,2,4,5,6,8,9, 12, 13, 14)$

- The map method of simplification: convenient for < 5 variable
- The tabulation method overcomes this difficulty: specific step-by-step procedure
- It is also known as the Quine-McCluskey method.
- The tabular method of simplification consists of two parts
- Exhaustive search for *prime implicant*
- Find least number of literals from *prime implicant search*

Don’t write or place any image in this area

DETERMINATION OF PRIME IMPlICANTS

- List of minterms that specify the function in first coloumn.
- The process compares each min term with every other minterm. If two min terms differ in only one variable, that variable is removed and remaining variables are considered.
- This process is repeated for every minterm until no further elimination of literals

Simplify the following Boolean function by using the tabulation method:

$F(w, x, y, z) = \sum (0, 1,2,8,10,11,14, I5)$

Don’t write or place any image in this area

Step I: Group binary representation of the minterms according to the number of 1's contained

Step2: Any two min terms that differ from each other by only one variable can be combined,

and the unmatched variable removed. The minterms in one section are compared with

those of the next section down only.

Step 3: The terms of column (b) have only three variables. The searching and comparing

process is repeated for the terms in column (b) to form the two-variable terms of column (c).

Step 4: The unchecked terms in the table form the prime implicant

Don't write or place any image in this area

Simplify the following Boolean function by using the tabulation method: *F(w, x, y, z)* = ∑ (0, 1,2,8,10,11,14, I5)

Quine-McCluskey minimization method

	a	b	c
	w x y z	w x y z	w x y z
	0- 0 0 0 0	(0,1)- 0 0 0 - (0,2)- 0 0 - 0 (0,8)- - 0 0 0	(0, 2 8,10) - 0 - 0 (0, 8 2,10) - 0 - 0
	1- 0 0 0 1 2- 0 0 1 0 8- 1 0 0 0		(10,11,14,15) 1 - 1 – (10, 14, 11, 15) 1 - 1 -
	10-1 0 1 0	(2,10) - 0 1 0 (8, 10) 1 0 - 0	F= W' X'Y' + X'Z'+ WY
	11-1 0 1 1 14-1 1 1 0	(10,11) 1 0 1 – (10,14) 1 – 1 0	
	15-1 1 1 1	(11,15) 1 - 1 1	

Quine-McCluskey minimization method

Don’t write or place any image in this area

Quine-McCluskey minimization method

Simplify the following Boolean function by using the tabulation method: $F(w, x, y, z) = \sum(1,4,6,7,8,9, 10, 11, 15)$

(a)			(b)			(c)
0001	1	√	1, 9	(8)		8, 9, 10, 11 (1, 2)
0100	4	√	4, 6	(2)		8, 9, 10, 11 (1, 2)
1000	8	√	8, 9	(1)	√	
			8, 10	(2)	√	
0110	6	√				
1001	9	√	6, 7	(1)		
1010	10	√	9, 11	(2)	√	
			10, 11	(1)	√	
0111	7	√				
1011	11	√	7, 15	(8)		
			11, 15	(4)		
1111	15	√				

Don’t write or place any image in this area

Quine-McCluskey minimization

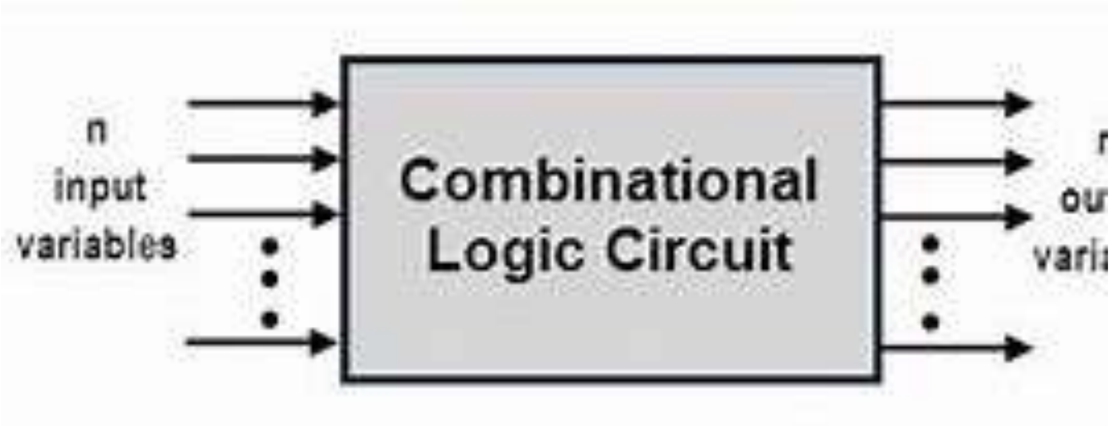
Prime implicants

Decimal	w	x	y	z	Term
	Binary				
1, 9 (8)	–	0	0	1	$x'y'z$
4, 6 (2)	0	1	–	0	$w'xz'$
6, 7 (1)	0	1	1	–	$w'xy$
7, 15 (8)	–	1	1	1	xyz
11, 15 (4)	1	–	1	1	wyz
8, 9, 10, 11 (1, 2)	1	0	–	–	wx'

		1	4	6	7	8	9	10	11	15
√ $x'y'z$	1, 9	X					X			
√ $w'xz'$	4, 6		X	X						
$w'xy$	6, 7			X	X					
xyz	7, 15				X					X
wyz	11, 15								X	X
√ wx'	8, 9, 10, 11					X	X	X	X	
		√	√	√		√	√	√	√	

Cominational Logic Circuits

Logic circuits for digital systems may be combinational or sequential



Don’t write or place any image in this area

DESIGN PROCEDURE

1. The problem is stated.
2. Define input and output variables.
3. The input and output variables are assigned letter symbols.
4. Derive truth table (The truth table that defines the required relationships between inputs and outputs is derived).
5. The simplified Boolean function for each output is obtained.

Don't write or place any image in this area

ADDERS

- Digital computers perform the **basic arithmetic** operation is the addition of two binary digits.
- A combinational circuit that performs the addition of two bits is called a *half-adder*. Performs the addition of three bits is *Full-adder*.

Half-Adder.

1. Define problem: Addition of two binary digits.

2. **Define i/o variables**: input variables are 2 and output are 2

3. **Assign** x, y as input and Sum, Carry are output variables .

Don't write or place any image in this area

4. Define TT

x	y	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

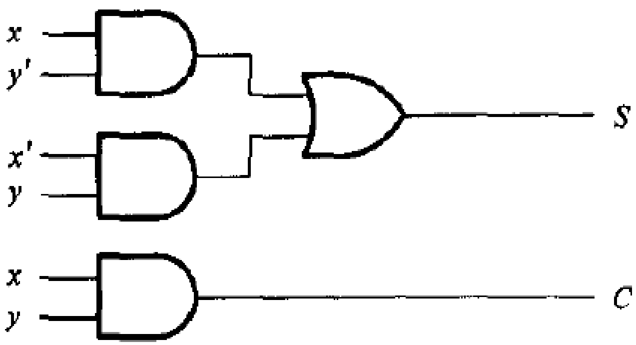
5. The simplify Boolean function

$$S = \bar{x}\,y + x\,\overline{y}$$

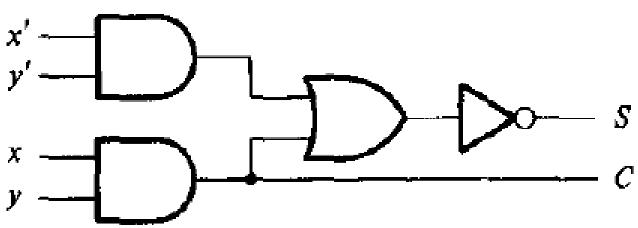
$$C = xy$$

Don’t write or place any image in this area

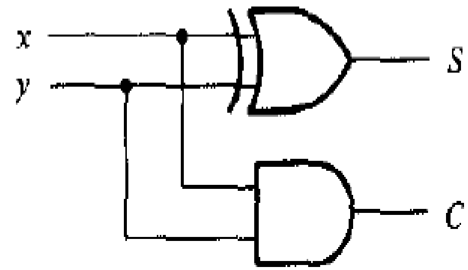
6. Draw the logic diagram



(a) $S = xy' + x'y$
$C = xy$



(c) $S = (C + x'y')'$
$C = xy$



(e) $S = x \oplus y$
$C = xy$

Don't write or place any image in this area

1. Problem statement : A full-adder is a combinational circuit that forms the arithmetic sum of three input bits.
2. Define I/O: It consists of three inputs and two outputs
3. Notation: Input variables are denoted by x,y and Z and output are denoted by S,C.

Don’t write or place any image in this area

4. Truth table

xyz	S	C
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	0	1
110	0	1
111	1	1

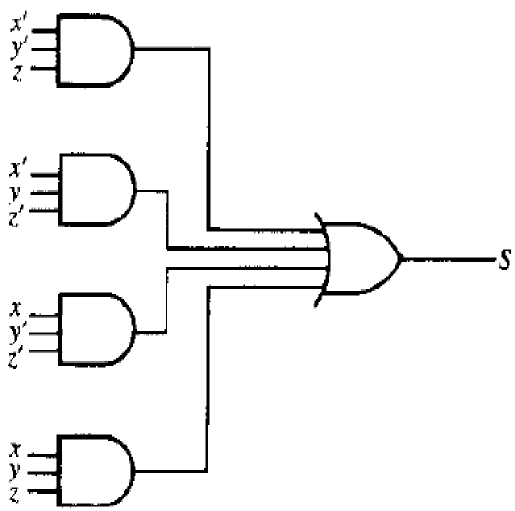
5. Simplify the boolean function

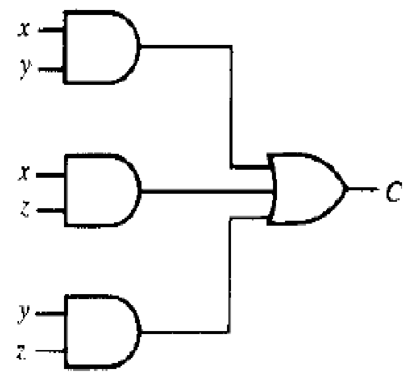
$S = x'y'z + x'yz' + xy'z' + xyz$

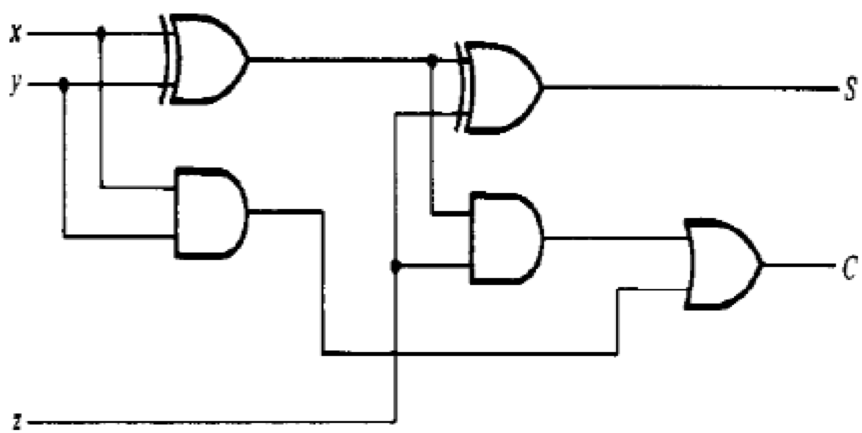
$C = xy + xz + yz$

Don't write or place any image in this area

6. Draw the logic diagram

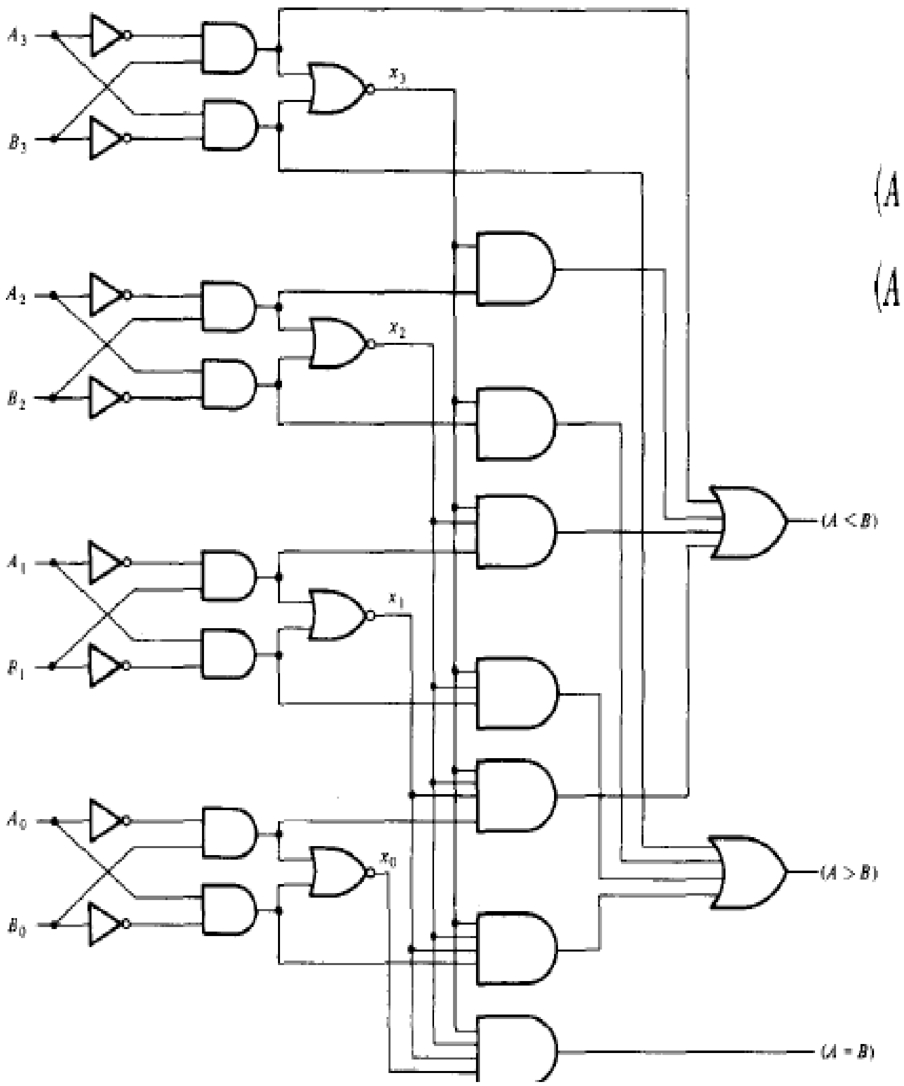






Don’t write or place any image in this area

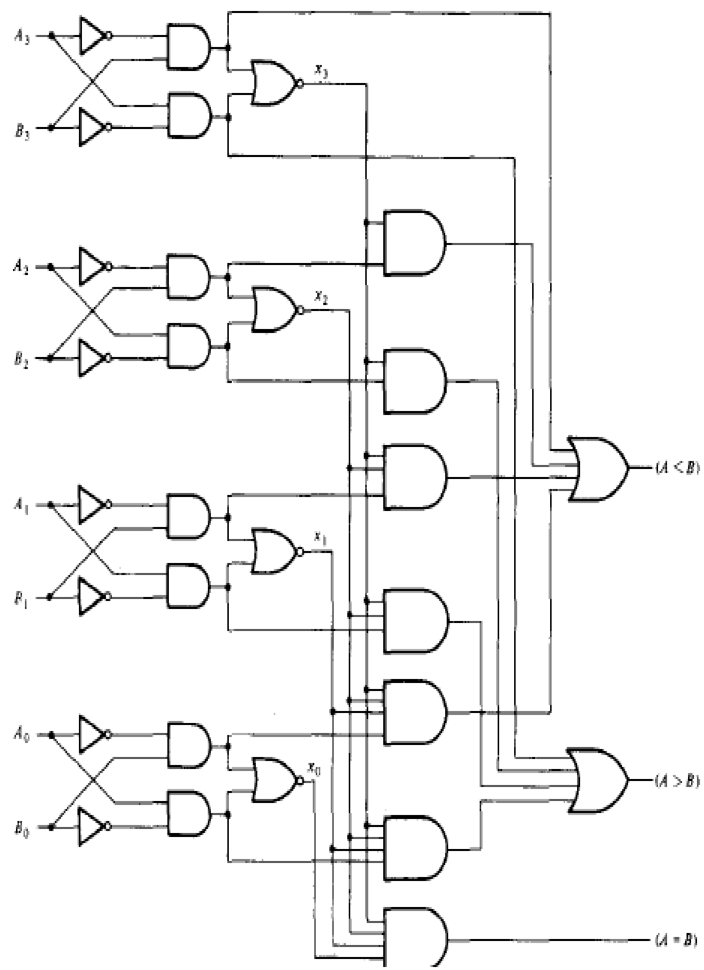
MAGNITUDE COMPARATOR



$$(A > B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$

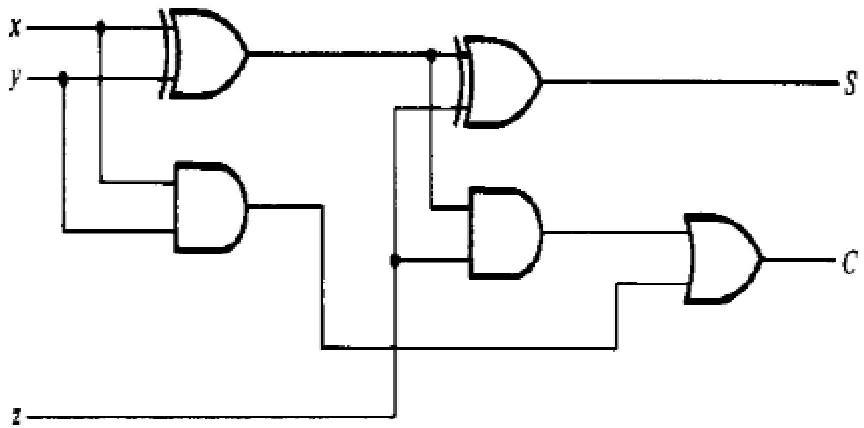
$$(A < B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$$

Don’t write or place any image in this area



A_3
B_3
x_3
A_2
B_2
x_2
A_1
B_1
x_1
A_0
B_0
x_0
$(A < B)$
$(A > B)$
$(A = B)$

6. Draw the logic diagram



$$
\begin{aligned}
S &= z \oplus (x \oplus y) \\
&= z'(xy' + x'y) + z(xy' + x'y)' \\
&= z'(xy' + x'y) + z(xy + x'y') \\
&= xy'z' + x'yz' + xyz + x'y'z
\end{aligned}
$$

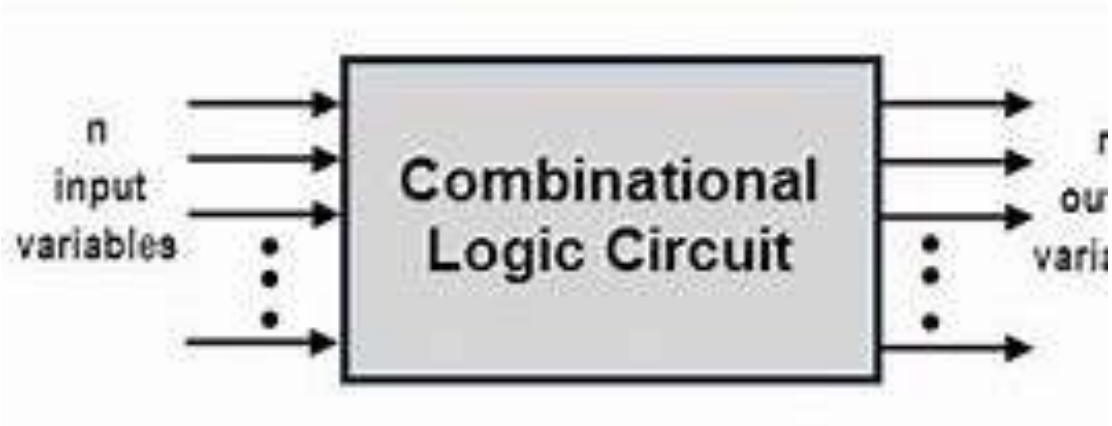
and the carry output is

$$C = z(xy' + x'y) + xy = xy'z + x'yz + xy$$

Don’t write or place any image in this area

Cominational Logic Circuits

Logic circuits for digital systems may be combinational or sequential



Don’t write or place any image in this area

DESIGN PROCEDURE

1. The problem is stated.
2. Define input and output variables.
3. The input and output variables are assigned letter symbols.
4. Derive truth table (The truth table that defines the required relationships between inputs and outputs is derived).
5. The simplified Boolean function for each output is obtained.

Don't write or place any image in this area

ADDERS

- Digital computers perform the **basic arithmetic** operation is the addition of two binary digits.
- A combinational circuit that performs the addition of two bits is called a *half-adder*. Performs the addition of three bits is *Full-adder*.

Half-Adder.

1. Define problem: Addition of two binary digits.

2. **Define i/o variables**: input variables are 2 and output are 2

3. **Assign** x, y as input and Sum, Carry are output variables .

Don't write or place any image in this area

4. Define TT

x	y	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

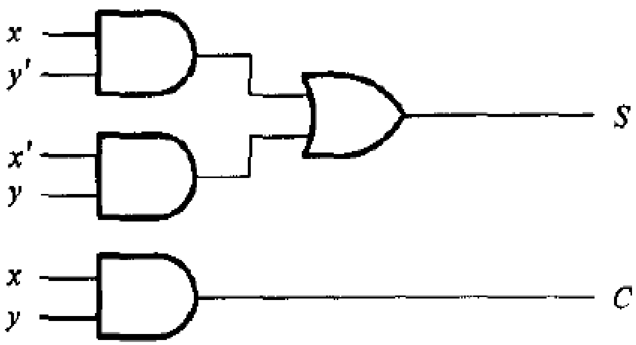
5. The simplify Boolean function

$$S = \bar{x}\, y + x\,\overline{y}$$

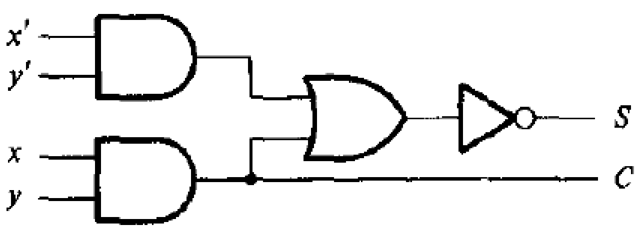
$$C = xy$$

Don’t write or place any image in this area

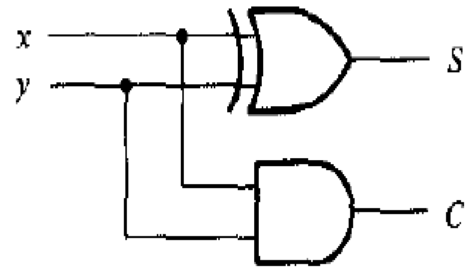
6. Draw the logic diagram



(a) $S = xy' + x'y$
$C = xy$



(c) $S = (C + x'y')'$
$C = xy$



(e) $S = x \oplus y$
$C = xy$

Don’t write or place any image in this area

FULL-ADDER

1. Problem statement : A full-adder is a combinational circuit that forms the arithmetic sum of three input bits.
2. Define I/O: It consists of three inputs and two outputs
3. Notation: Input variables are denoted by x,y and Z and output are denoted by S,C.

Don't write or place any image in this area

4. Truth table

xyz	S	C
000	0	0
001	1	0
010	1	0
011	0	1
100	1	0
101	0	1
110	0	1
111	1	1

Don’t write or place any image in this area

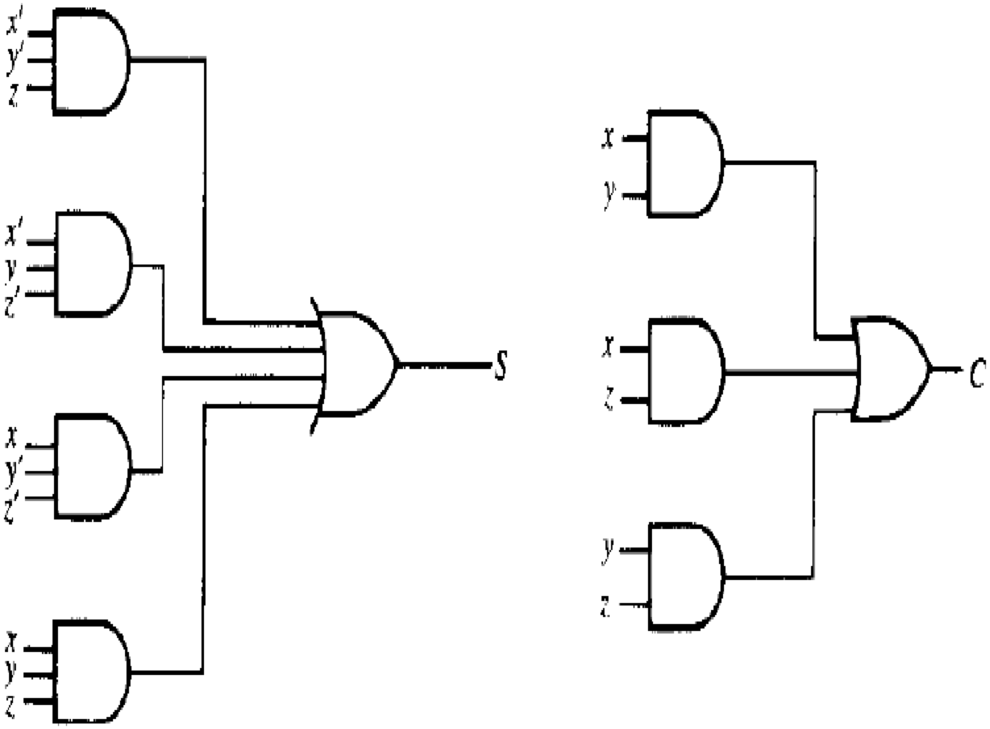
5. Simplify the boolean function

$S = x'y'z + x'yz' + xy'z' + xyz$

$C = xy + xz + yz$

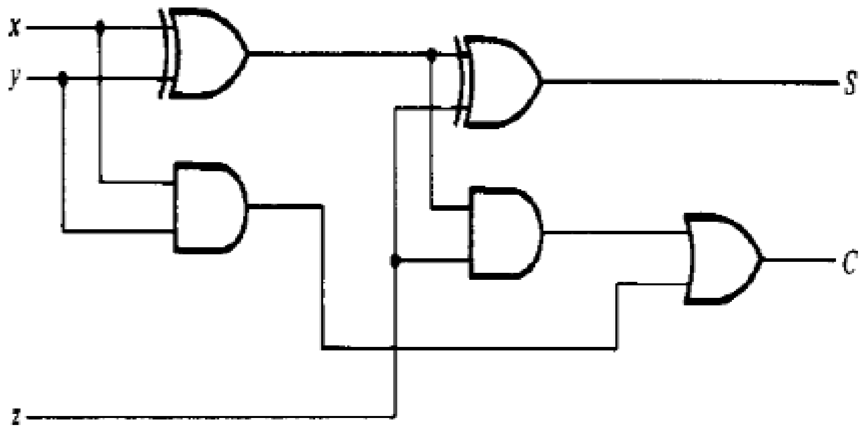
Don't write or place any image in this area

6. Draw the logic diagram



Don’t write or place any image in this area

6. Draw the logic diagram



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

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

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

$$= xy'z' + x'yz' + xyz + x'y'z$$

and the carry output is

$$C = z(xy' + x'y) + xy = xy'z + x'yz + xy$$

Don’t write or place any image in this area

Full-Subtractor

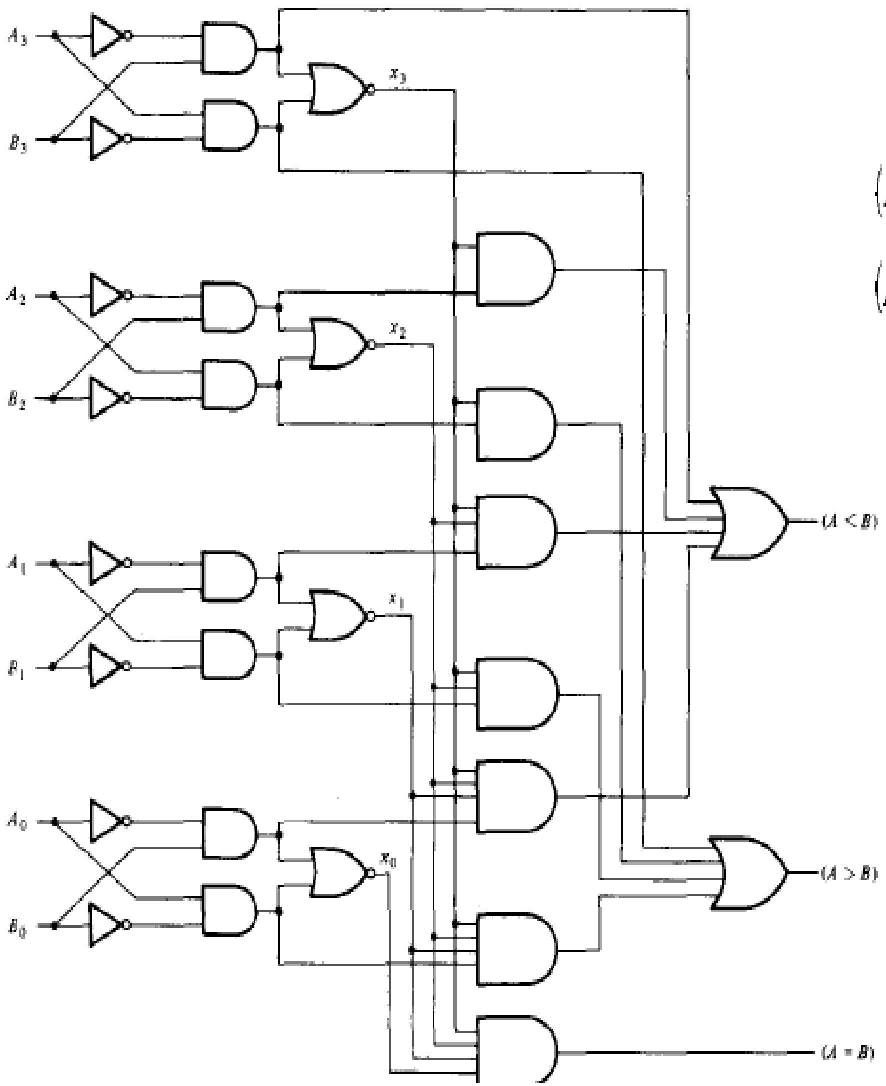
x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$D = x'y'z + x'yz' + xy'z' + xyz$$

$$B = x'y + x'z + yz$$

Don’t write or place any image in this area

MAGNITUDE COMPARATOR



$$(A > B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$

$$(A < B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$$

Don’t write or place any image in this area

Decoder

TABLE 5-2
Truth Table of a 3-to-8-Line Decoder

Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

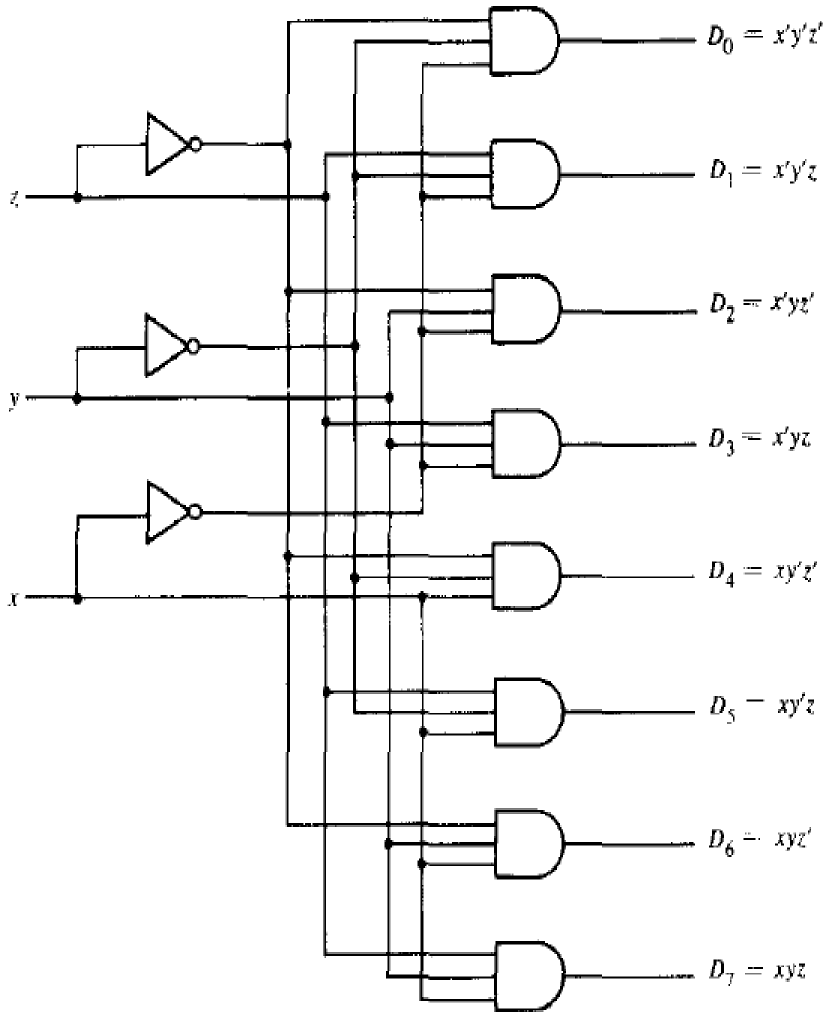


FIGURE 5-8

A 3-to-8 line decoder

8X3 Encoder

TABLE 5-3
Truth Table of Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$$z = D_1 + D_3 + D_5 + D_7$$

$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

Don’t write or place any image in this area

Encoder

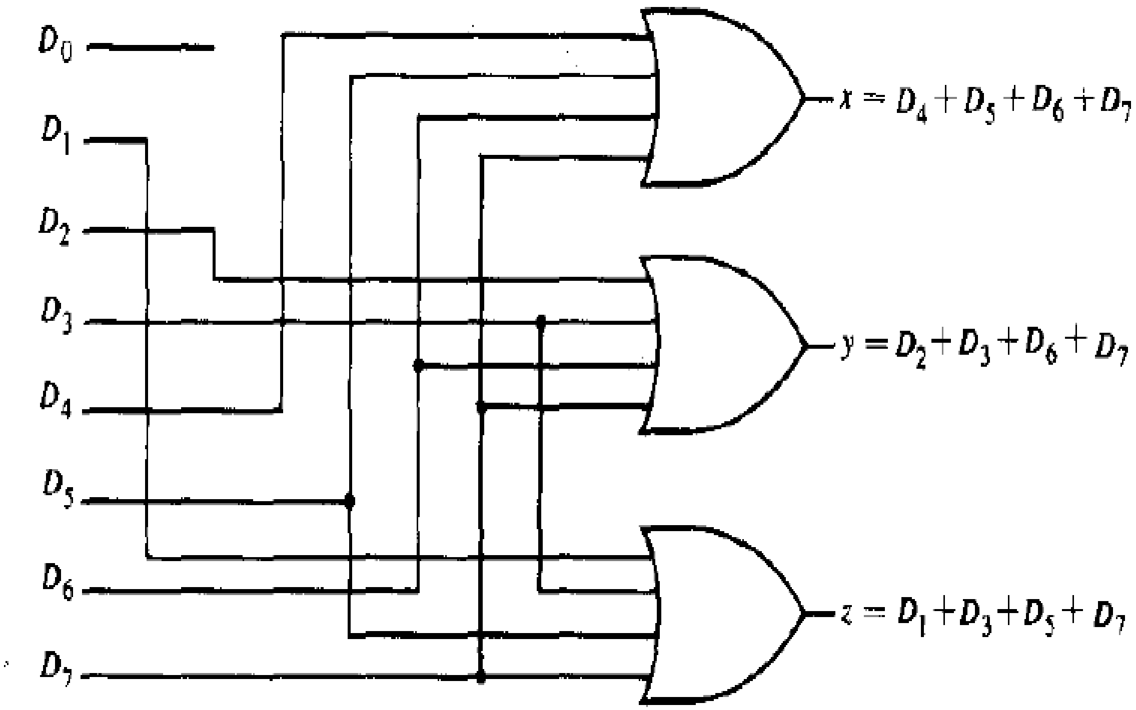
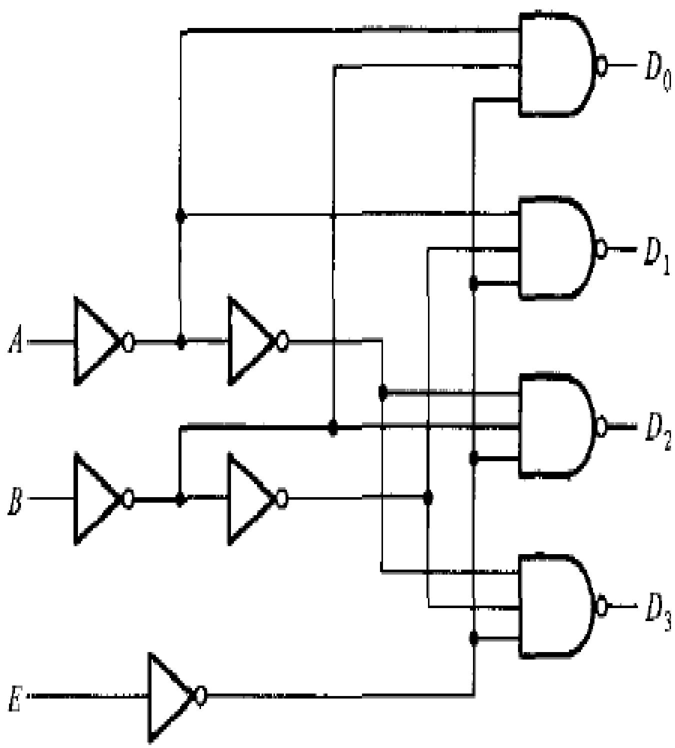


FIGURE 5-13

Octal-to-binary encoder

Don’t write or place any image in this area

Enabled Decoder



(a) Logic diagram

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table

Implement a full-adder circuit with a decoder and two OR gates.

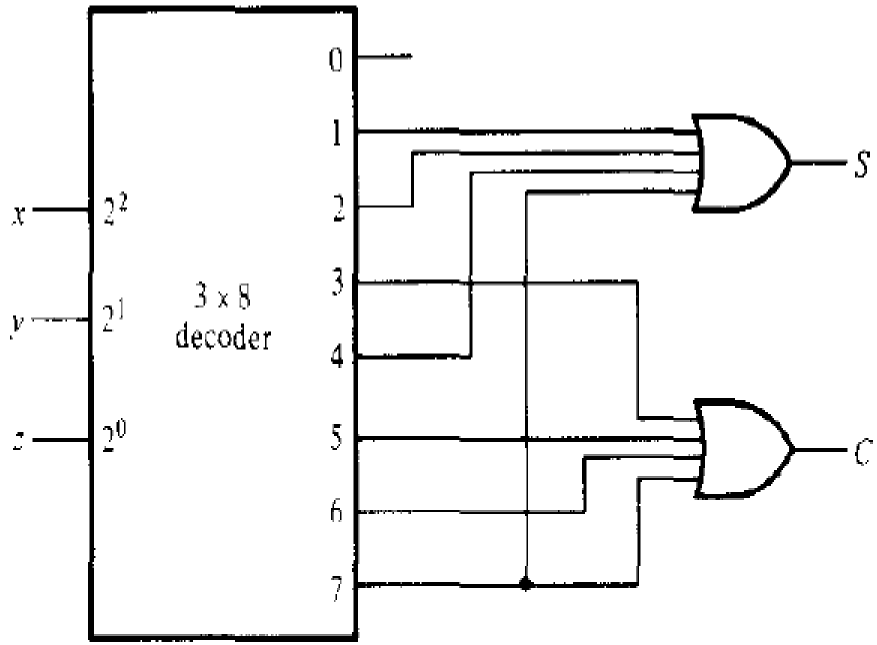


FIGURE 5-9

Implementation of a full-adder with a decoder

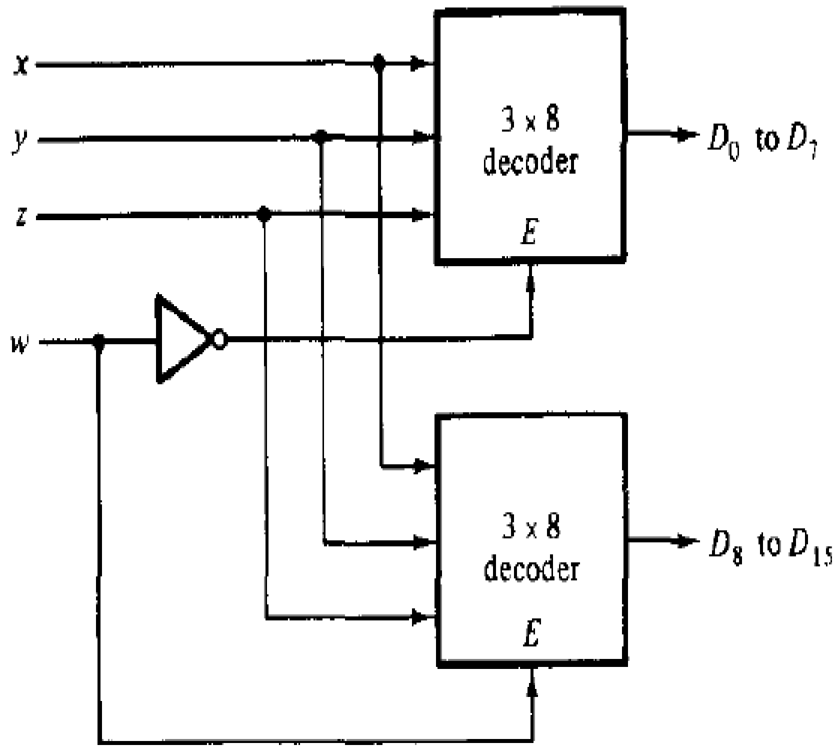
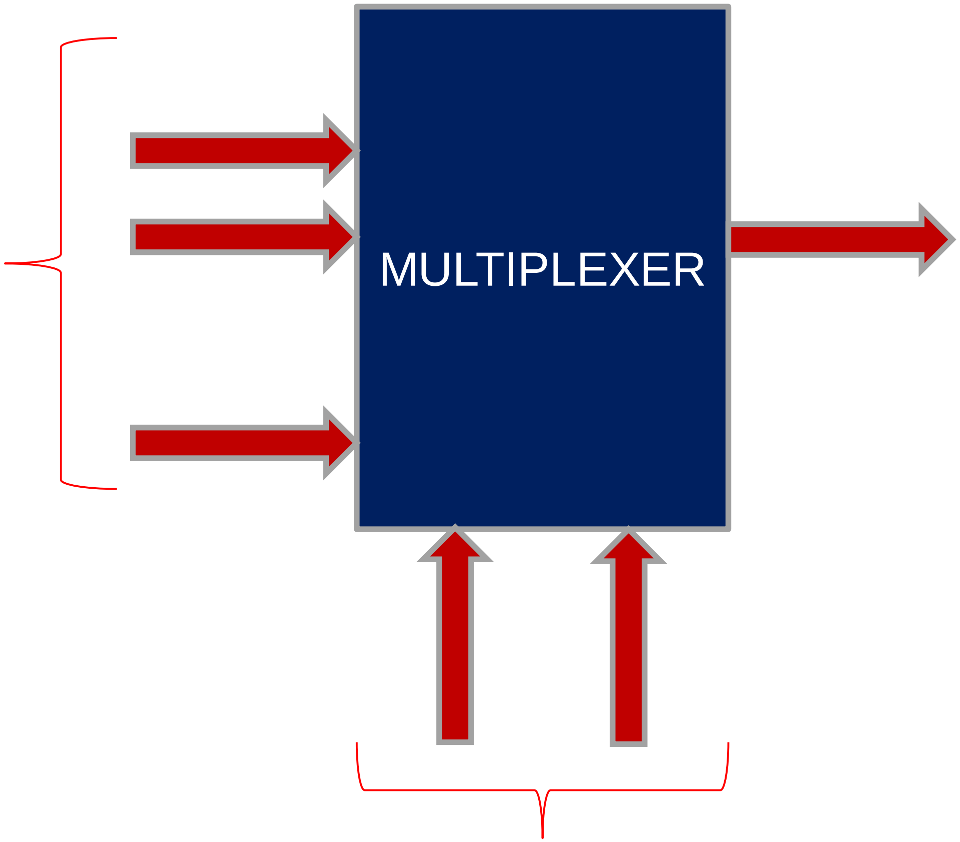


FIGURE 5-12

A 4 × 16 decoder constructed with two 3 × 8 decoders

Multiplexer



- It is a combinational circuit
- Many input and one output
- Depending on select i/p, one of the data i/p is transferred to the o/p

Don’t write or place any image in this area

Multiplexer

- Example of multiplexers are $2^n X1$ where $\mathrm{n} = 1,2,3,4, \ldots$ etc.
- For example 2x1
- Implement by following combinational design steps.

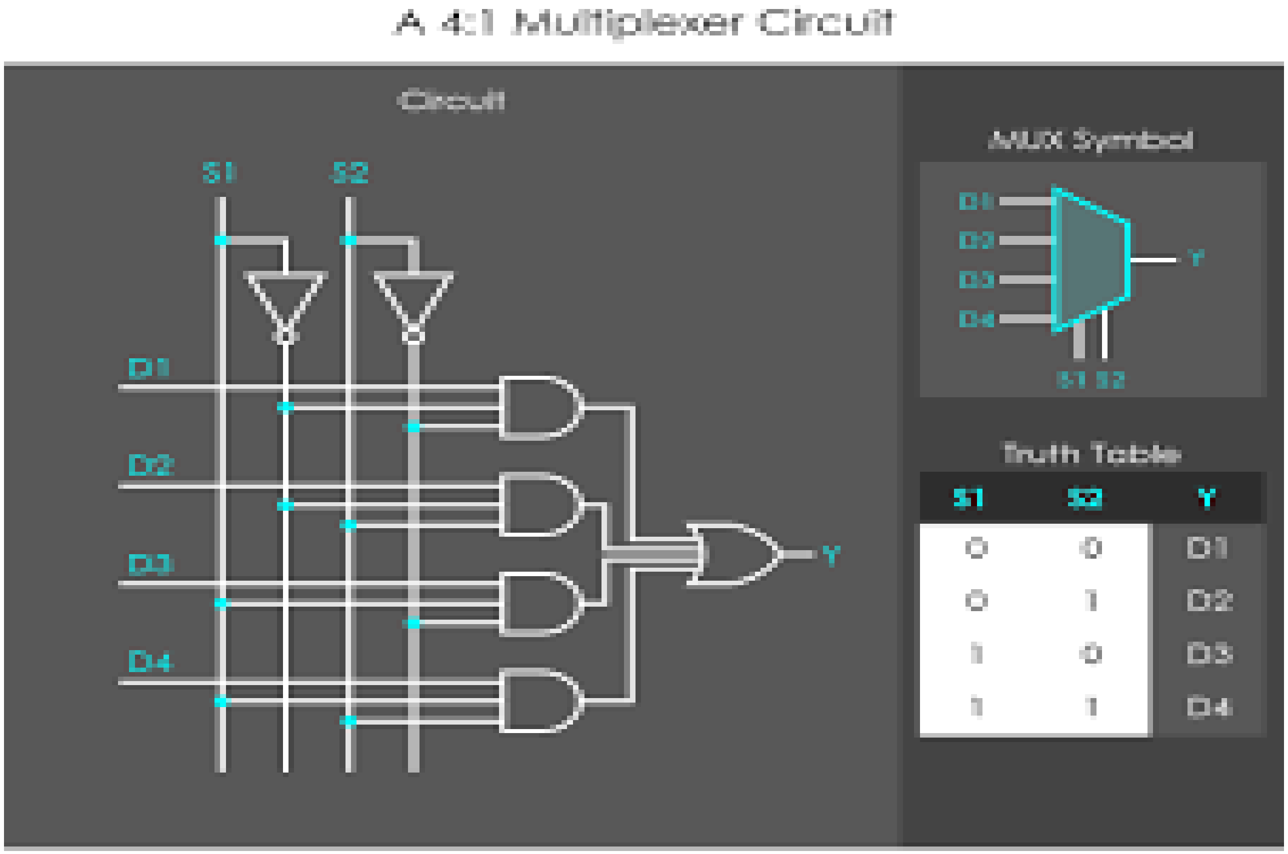
Don't write or place any image in this area

Multiplexer

- Example of multiplexers are $2^n X1$ where $\mathrm{n} = 1,2,3,4, \ldots$ etc.

Don't write or place any image in this area

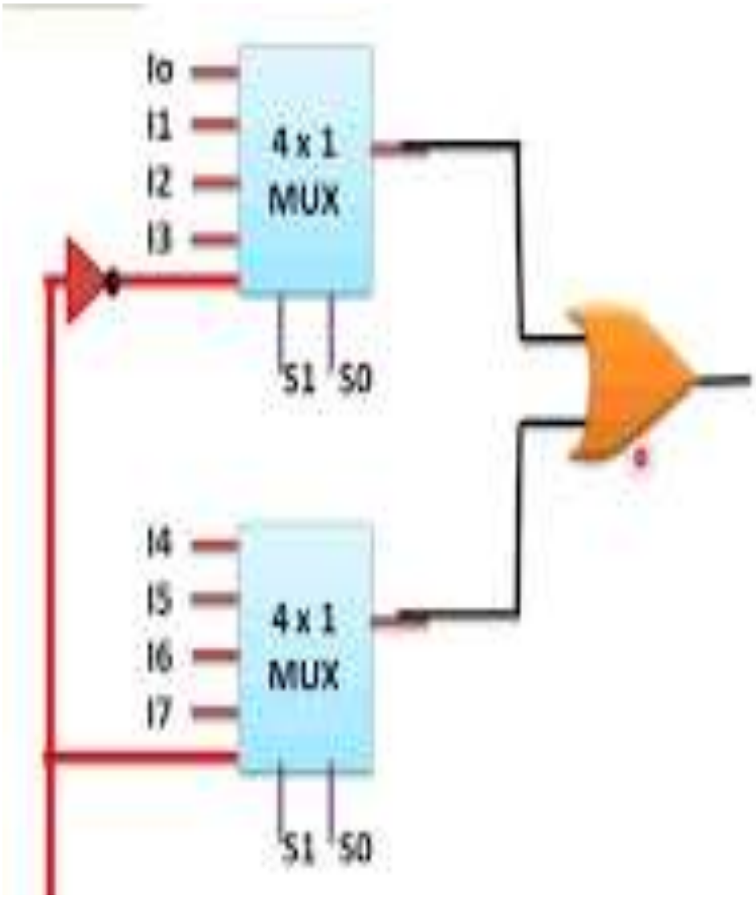
4x1 Multiplexer



Don’t write or place any image in this area

Multiplexer

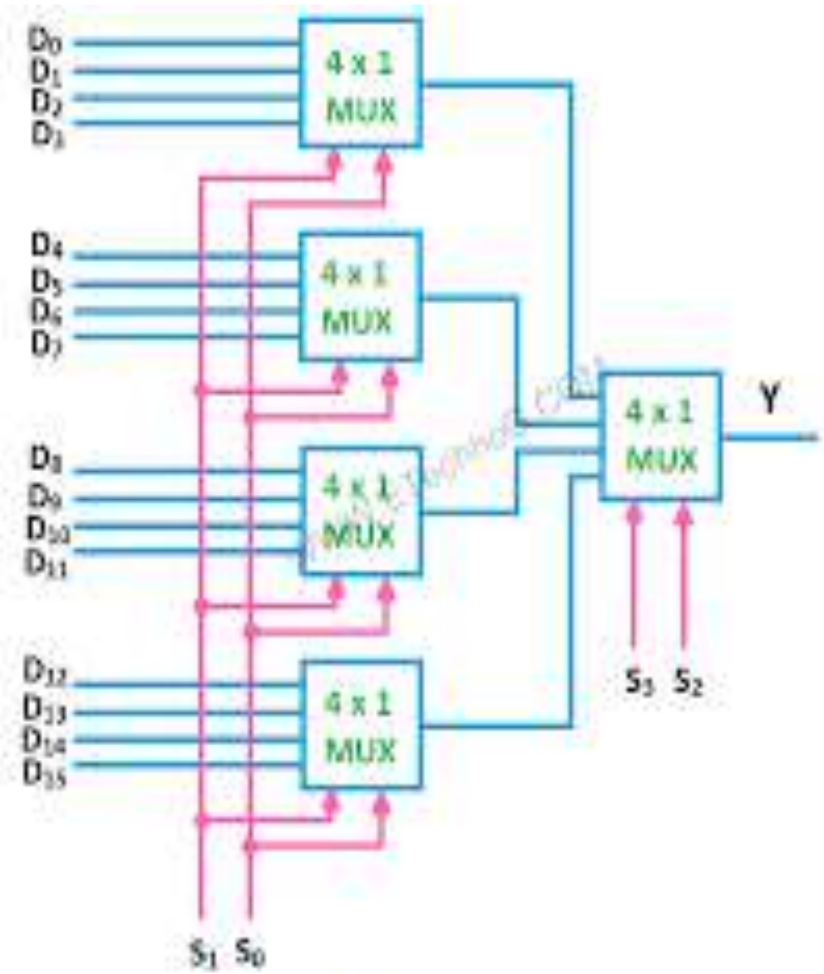
Construct a 8X1multiplexer using 4X1multiplexer



S2	S1	S0	Y
0	0	0	Io
0	0	1	I1
0	1	0	I2
0	1	1	I3
1	0	0	I4
1	0	1	I5
1	1	0	I6
1	1	1	I7

Don’t write or place any image in this area

16 X 1 muxer



16 to 1 Multiplexer

Don’t write or place any image in this area

16 X 1 muxer

Construct a 16 : 1 Mux using only 2 : 1 Mux

Implement a 64:1 MUX using 8:1 MUXs.

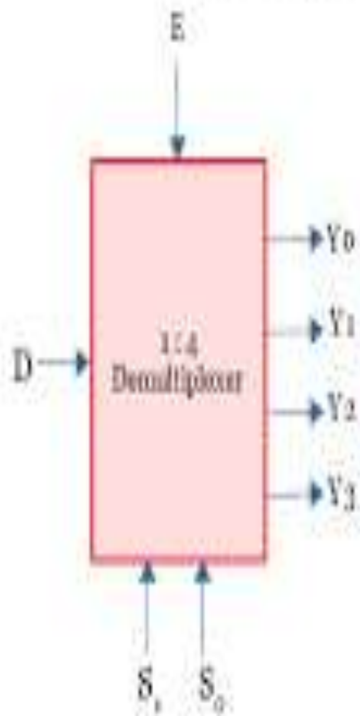
Don’t write or place any image in this area

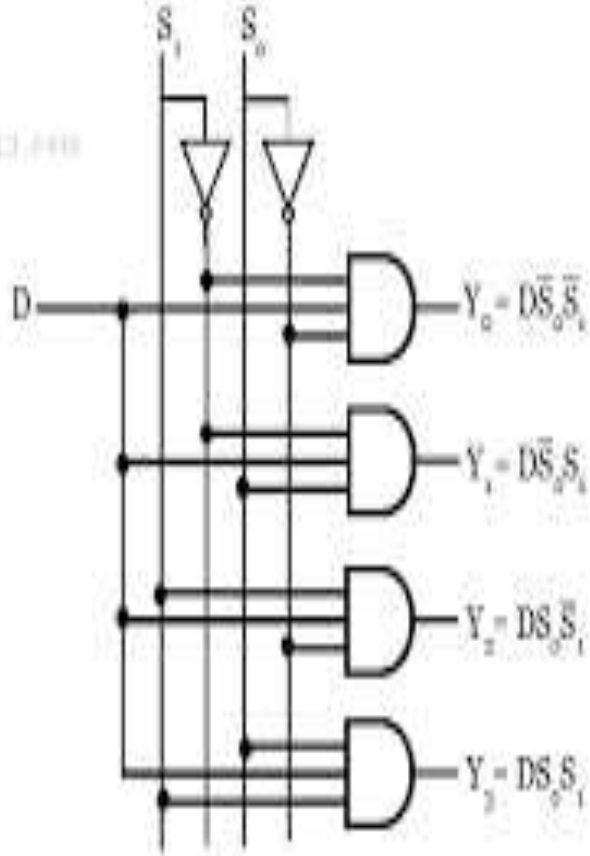
Demultiplexer

- It is a combinational circuit
- Single input and many outputs
- Depending on select i/p, i/p is transferred to the any one of the selected o/p.
- Also known as data distributor.

Don’t write or place any image in this area

Demultiplexer





Don’t write or place any image in this area

Demultiplexer

Don’t write or place any image in this area

Code convertor

TABLE 4-1
Truth Table for Code-Conversion Example

Input BCD				Output Excess-3 Code			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Don't write or place any image in this area

Code convertor

$$z = D'$$

$$y = CD + C'D' = CD + (C + D)'$$

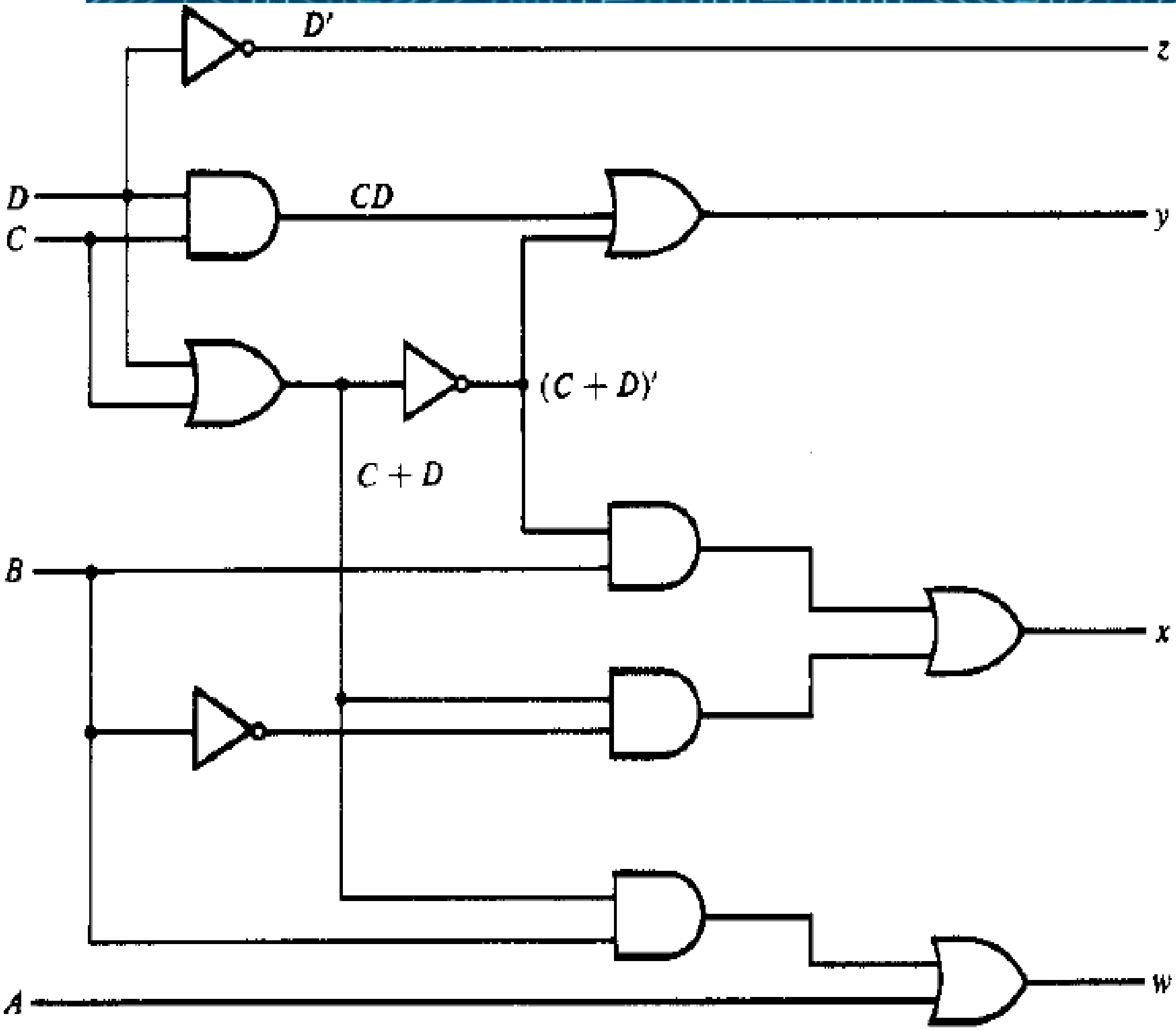
$$x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$

$$= B'(C + D) + B(C + D)'$$

$$w = A + BC + BD = A + B(C + D)$$

Don’t write or place any image in this area

Code convertor



Don't write or place any image in this area