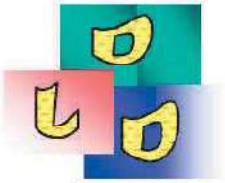
UNIT – III

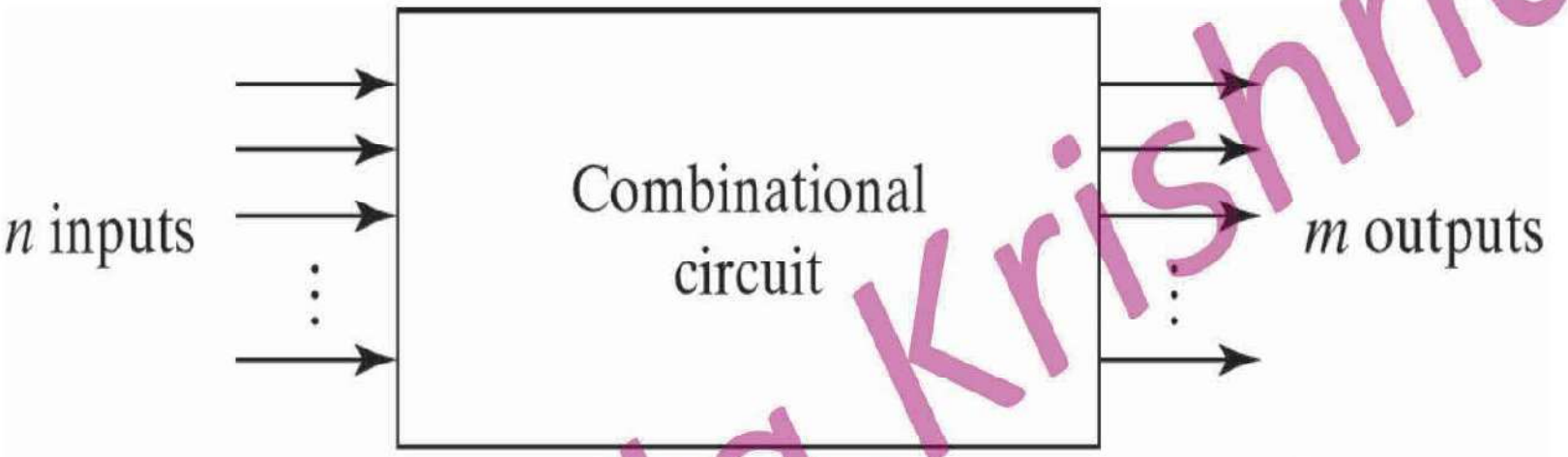
COMBINATIONAL LOGIC



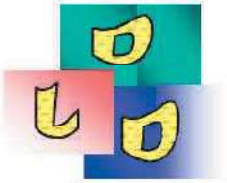
Combinational Circuits

- A combinational circuit is a circuit consists of logic gates whose outputs at any time are determined from the present input combination.
- A combinational circuit performs an operation that can be specified logically by a set of Boolean functions.
- A combinational circuit consists of input variables, logic gates, and output variables.
- The logic gates accepts signals from the inputs and generate signals to the outputs. This process transforms binary information from the given input data to a required output data.

Block Diagram of a Combinational Circuit



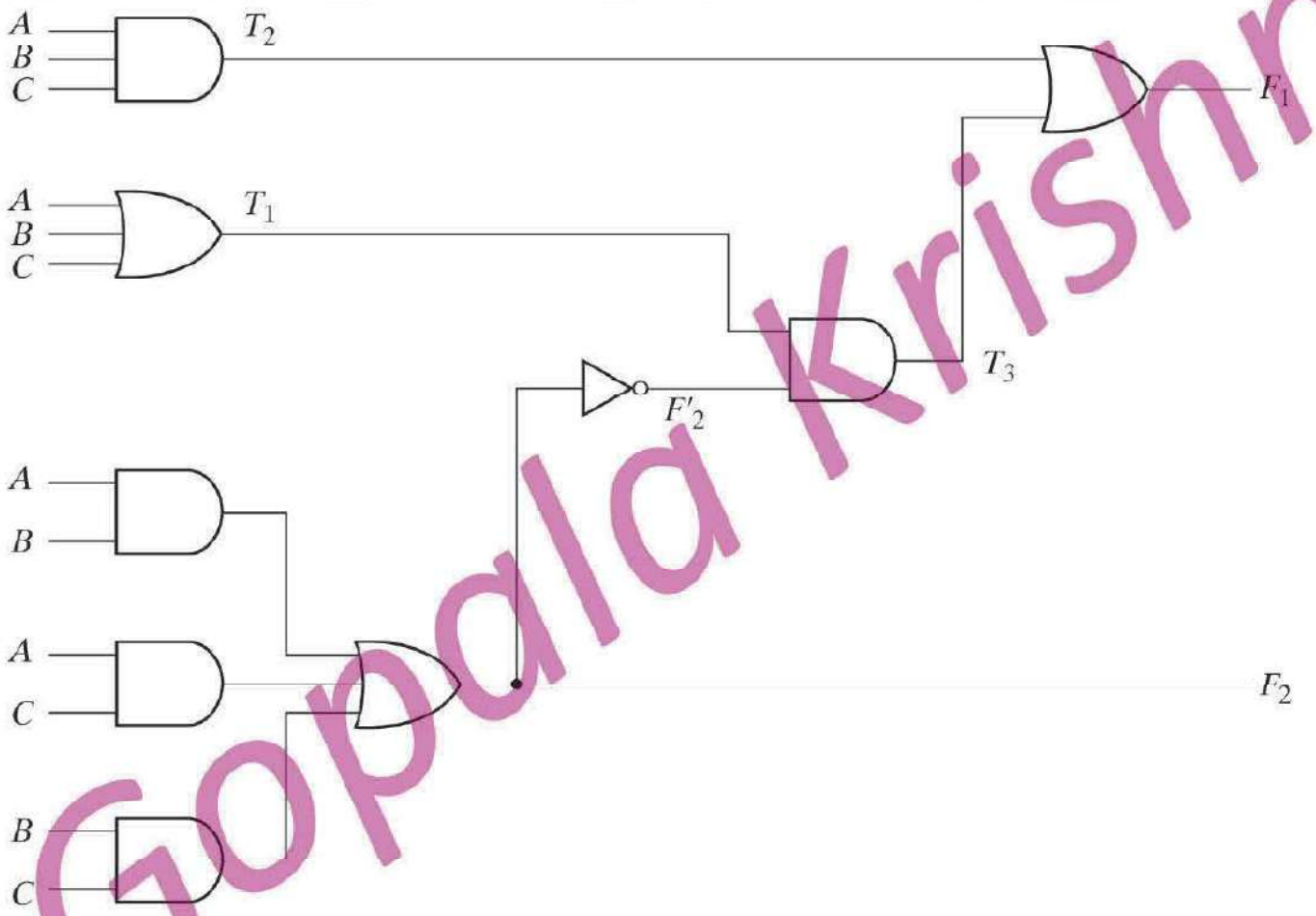
- For n input variables, there are 2^n possible input combinations.
- For each possible input combination, there is one output value. Thus, a combinational circuit can be specified with a truth table that lists the output values for each input combination.
- A combinational circuit can be described by m output Boolean functions, one for each output variable. Each output function is expressed interms of input variables.

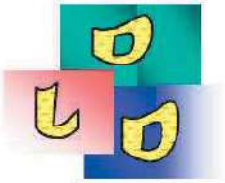


Analysis Procedure

- The analysis of a combinational circuit requires that we determine the function that the circuit implements.
- The analysis procedure is a procedure to obtain Boolean functions from a logic diagram as follows:
  - Label all gate outputs that are a function of input variables with arbitrary symbols. Determine the Boolean functions for each gate output.
  - Label the gate that are a function of input variables and previously labeled gates with other arbitrary symbols. Find the Boolean functions for these gates.
  - Repeat the process outlined in step2 until the outputs of the circuit are obtained.
  - By repeated substitution of previously defined functions, obtain the output Boolean function interms of input variables.

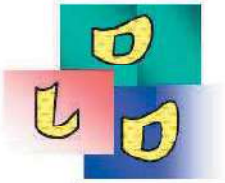
Example





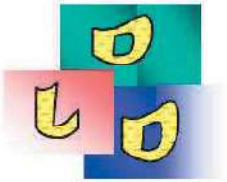
- $F_2=AB+AC+BC$
- $T_1=A+B+C$
- $T_2=ABC$
- Next, we consider outputs of gates that are a function of already defined symbols:
- $T_3=F_2{}^1T_1$
- $F_1=T_3+T_2$
- To obtain F_1 as a function of A,B,C form a series of substitutions as follows:

$$F_1=T_3+T_2=F_2{}^1T_1+ABC$$
$$=(AB+AC+BC)^1(A+B+C)+ABC$$
$$=(A^1+B^1)(A^1+C^1)(B^1+C^1)(A+B+C)+ABC$$
$$=A^1BC^1+A^1B^1C+AB^1C^1+ABC$$

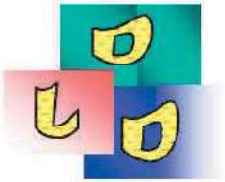


Deriving Truth Table Equivalent

- The derivation of the truth table for the circuit is a straight forward process once the output Boolean functions are known.
- To obtain the truth table directly from logic diagram without going through the derivations of the Boolean function, proceed as follows:
- Determine the number of input variables in the circuit. For n inputs, form the 2^n possible input combinations and list the binary numbers from 0 to 2^n -1 in a table.
- Label the outputs of selected gates with arbitrary symbols.
- Obtain the truth table for the outputs of those gates that are a function of the input variables only.
- Proceed to obtain the truth table for the outputs of those gates that are a function of previously defined values until the columns for all outputs are determined.

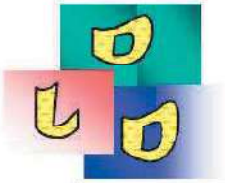


A	B	C	F_2	F_2^1	T_1	T_2	T_3	F_1
0	0	0	0	1	0	1	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	0	0	1



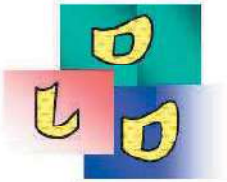
Design Procedure

- The design of combinational circuit starts from the specification of the problem and culminates in a logic circuit diagram or a set of Boolean Functions from which the logic diagram can be obtained.
- The procedure involves the following steps:
  - From the specification of the circuit, determine the required number of inputs and outputs and assign symbol to each
  - Derive the truth table that defines the required relationship between inputs and outputs.
  - Obtain the simplified Boolean functions for each output as a function of the input variables.
  - Draw the logic diagram and verify the correctness of the design.



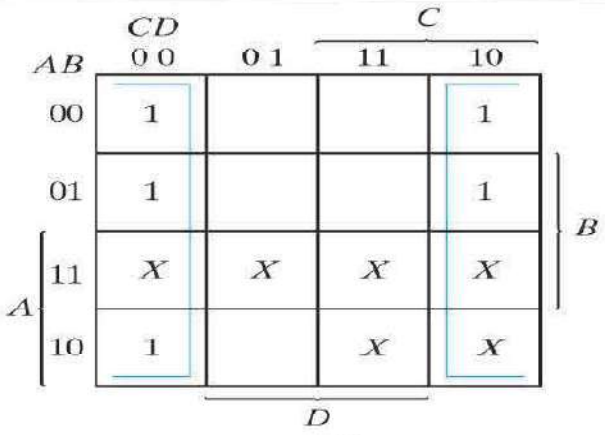
Code Conversion Example

- To convert from binary code A to binary code B, the input lines must supply the bit combination of elements specified by code A and the output lines must generate the corresponding bit combination of code B.
- A combinational circuit performs this transformation by means of logic gates.
- The design procedure will be illustrated by an example that converts the binary coded decimal (BCD) to the excess-3 code for the decimal digits.

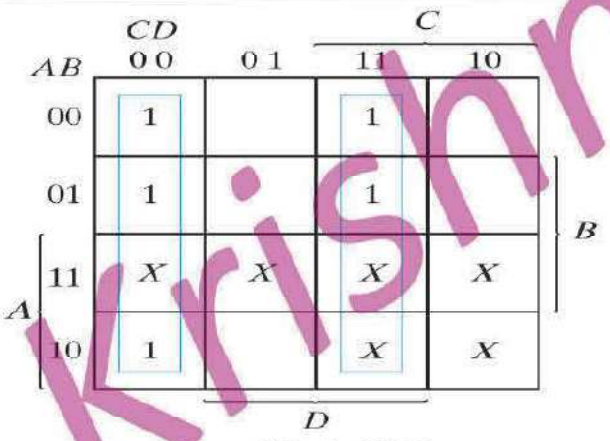


Input BCD				Output Excess – 3 Code			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
0	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

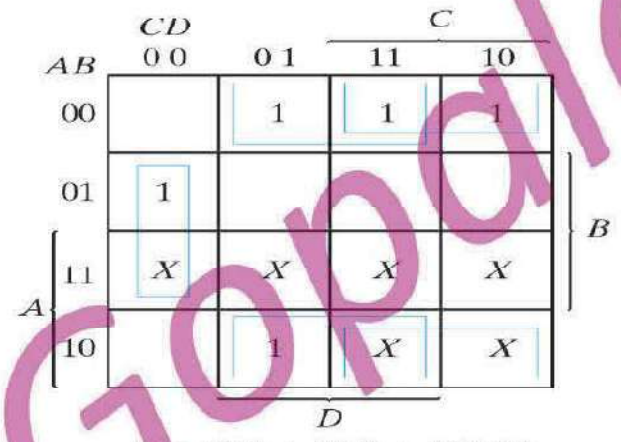
Maps for Code Conversion



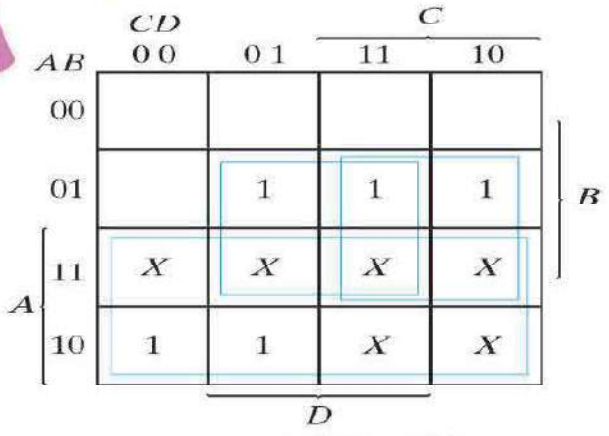
$z = D'$



$y = CD + C'D'$

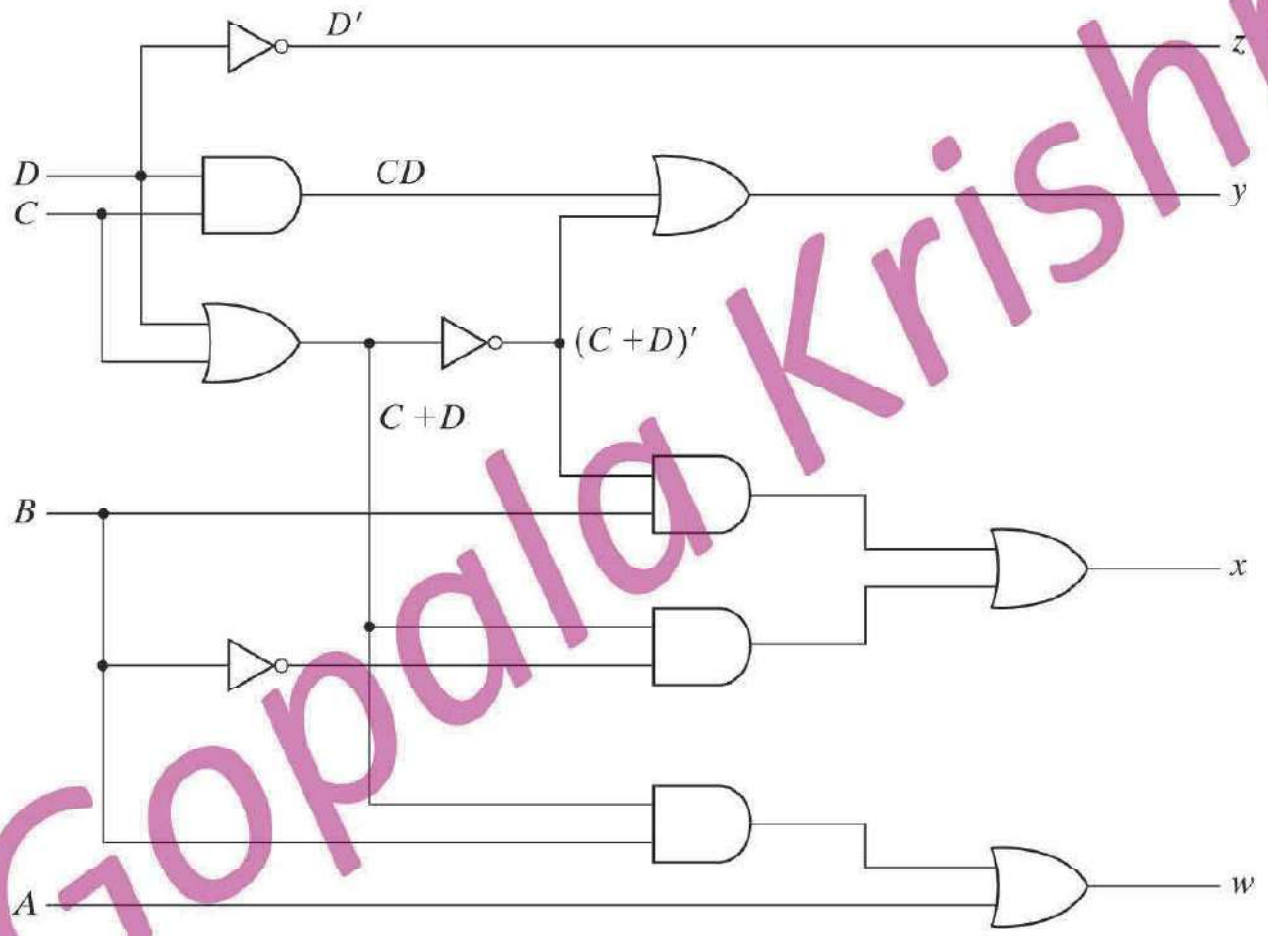


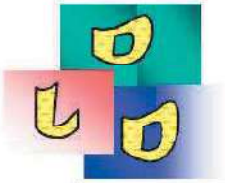
$X = B'C + B'D + BC'D'$



$w = A + BC + BD$

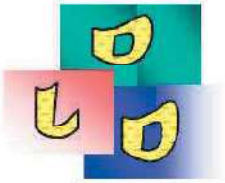
Logic Diagram for Code Conversion





Binary Adder - Subtractor

- A combinational circuit that performs the addition of two bits is called half adder.
- A combinational circuit that performs the addition of three bits is called full adder.
- A binary adder subtractor is a combinational circuit that performs the arithmetic operations of addition and subtraction with binary numbers.
- The design is hierarchical that the half adder is designed first from which we develop a full adder. Connecting n full adders in cascade produces a binary adder for two n bit numbers. The subtraction circuit is included by providing a complementing circuit.



Half Adder

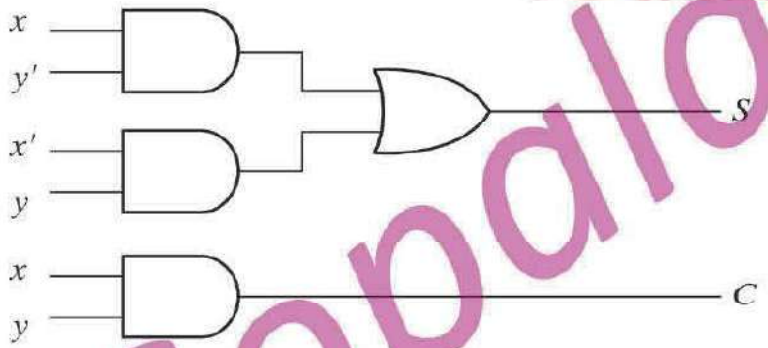
- From the verbal explanation of half adder, we find that this circuit needs two binary inputs and two binary outputs.
- The input variables designate the augend and addend bits; the output variables produce the sum and carry.
- We assign symbols x and y to inputs and S and C to the outputs.
- The truth table for the half adder is:

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

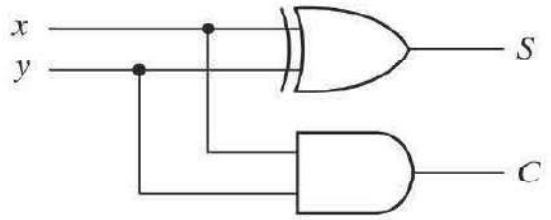
- The simplified Boolean functions for the two outputs can be obtained directly from the truth table. The simplified sum of products expressions are

$$S=x^1y+xy^1$$

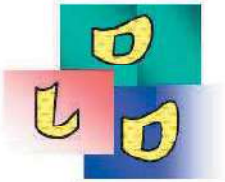
$$C=xy$$



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



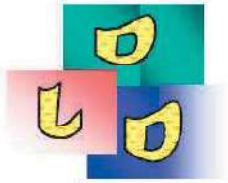
(b) $S = x \oplus y$
$C = xy$



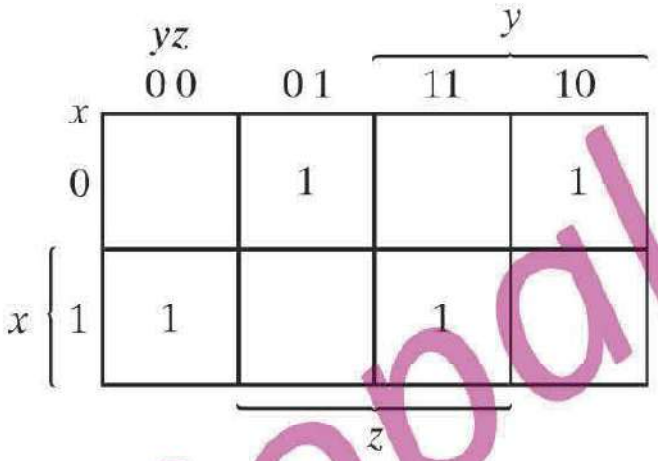
Full Adder

- A full adder is a combinational circuit that forms the arithmetic sum of three bits.
- It consists of three inputs and two outputs. The two of the inputs are denoted by x and y, represent the two significant bits to be added. The third bit z, represents the carry from the previous lower significant position.

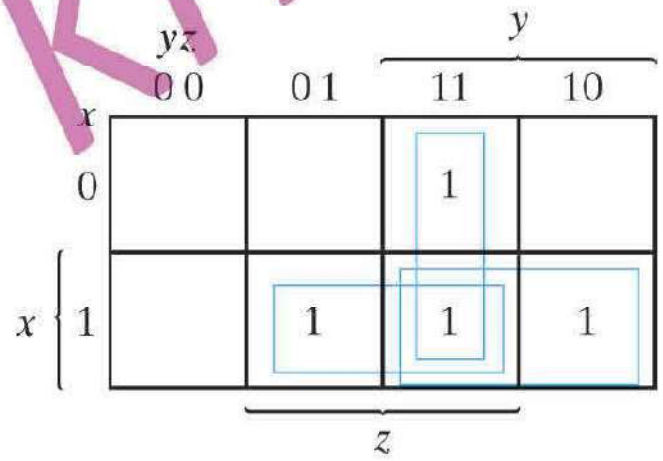
X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



- The simplified Boolean expressions for sum and carry can be obtained from map method.

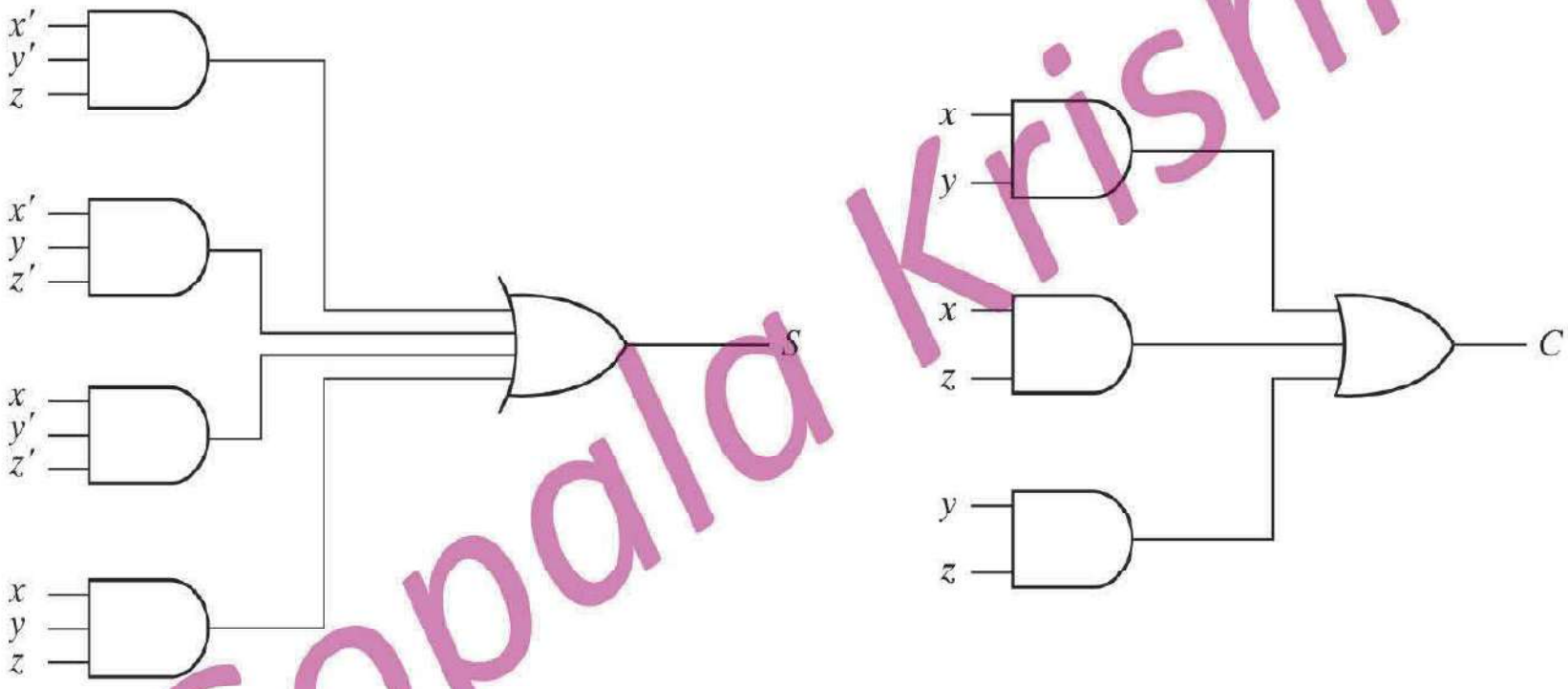


$$S = x'y'z + x'yz' + xy'z' + xyz$$

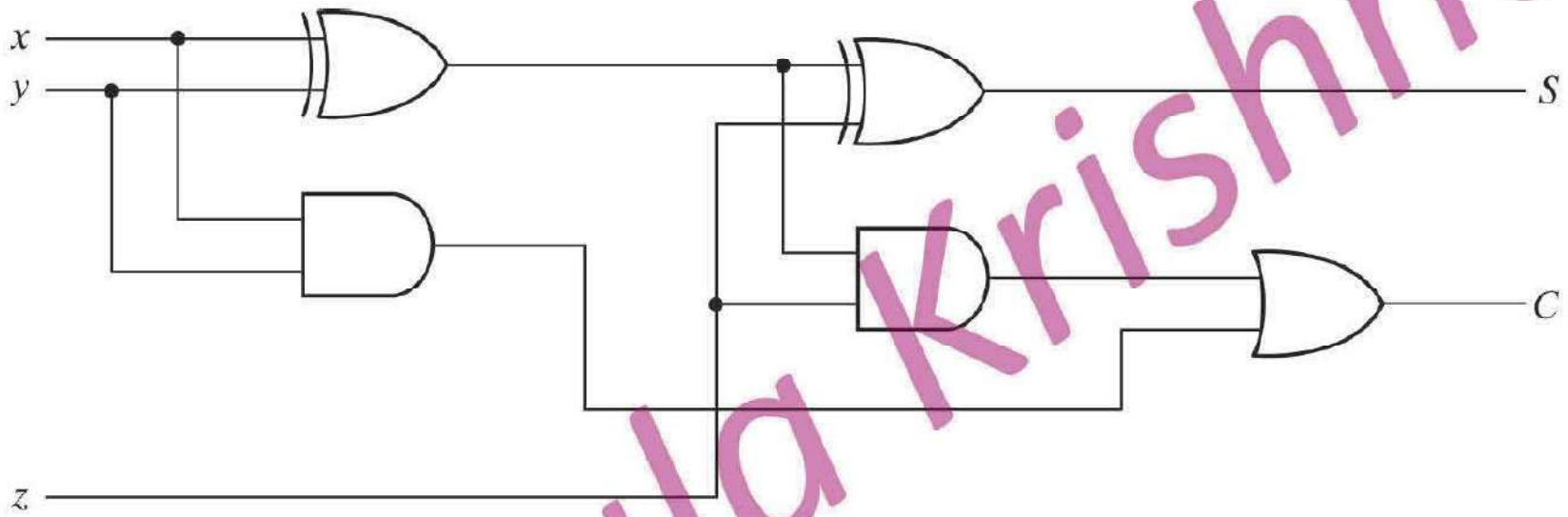


$$C = xy + xz + yz$$

Logic Diagram for Full Adder

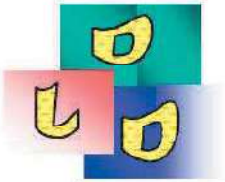


Implementation of Full Adder with two Half Adders



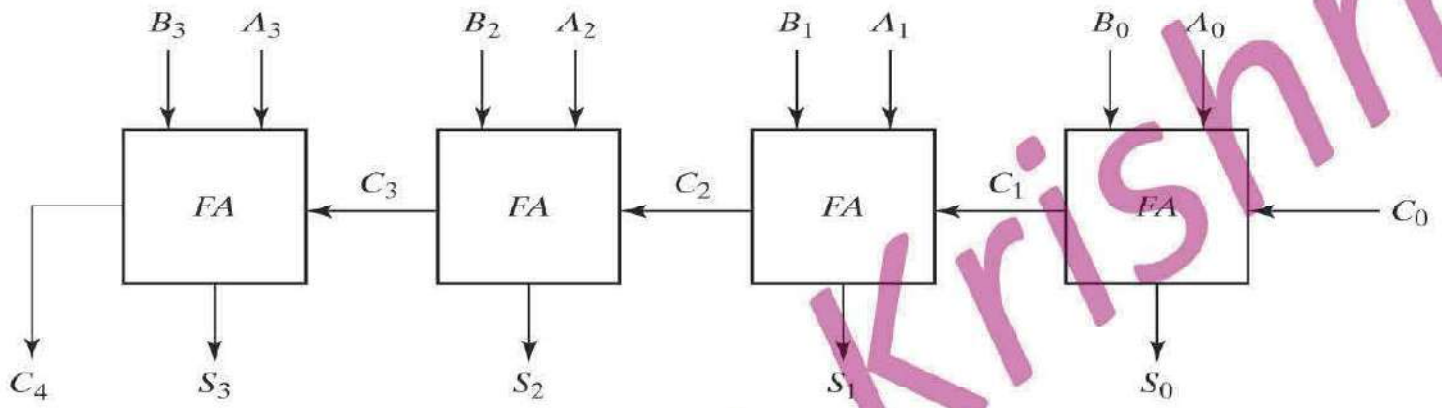
$C=z(xy^1+x^1y)+xy=xy^1z+x^1yz+xy$

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



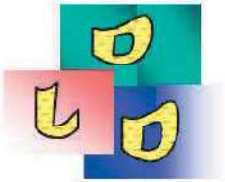
Binary Adder

- A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of the next full adder in the chain.



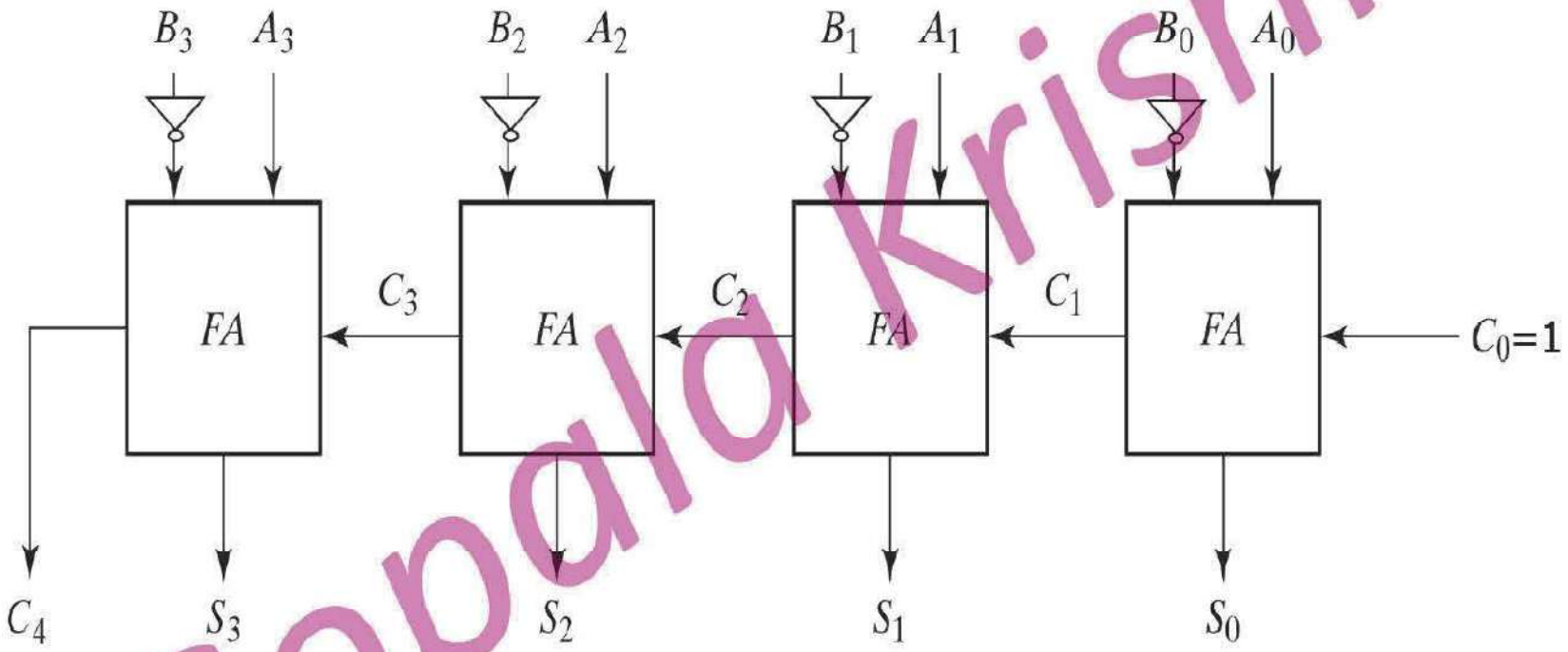
To demonstrate with specific example, consider the two binary numbers, A=1011 and B=0011. Their sum S=1110 is formed with the four bit adder as follows:

Subscript i:	**3**	**2**	**1**	**0**	
Input carry	0	1	1	0	C_i
Augend	1	0	1	1	A_i
Addend	0	0	1	1	B_i
Sum	1	1	1	0	S_i
Output Carry	0	0	1	1	C_{i+1}



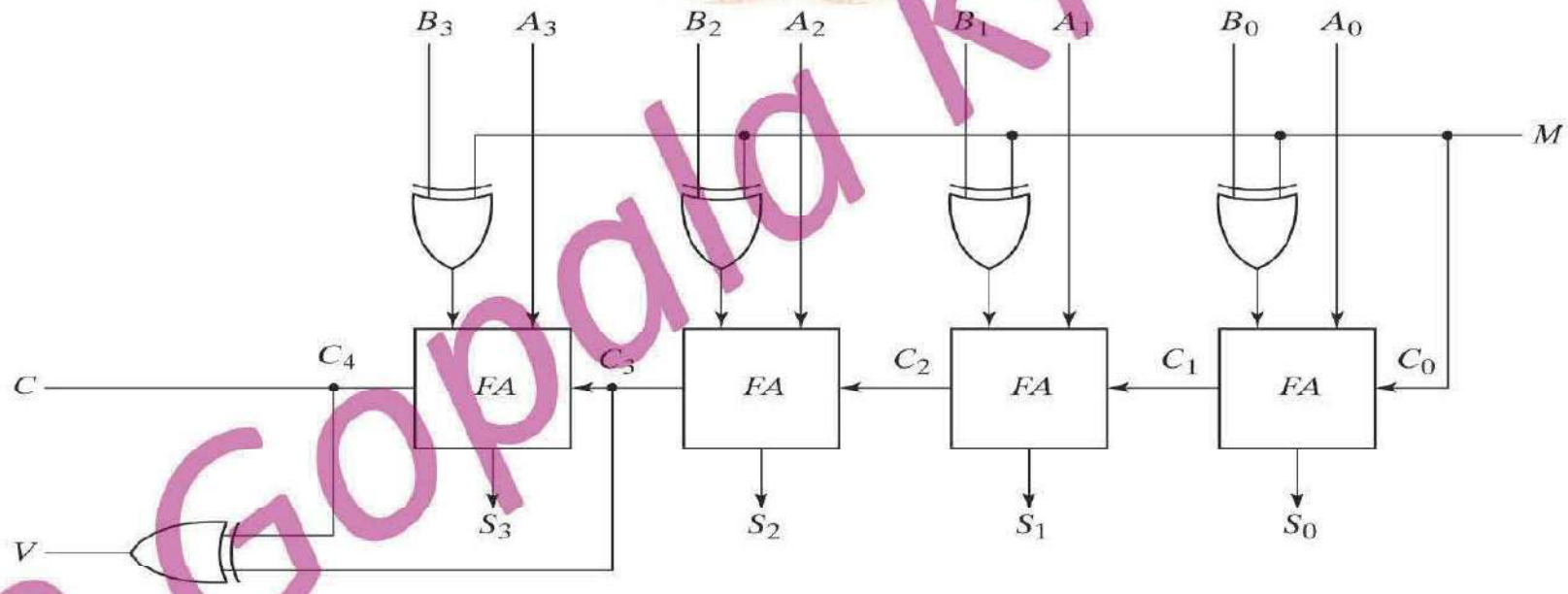
Binary Subtractor

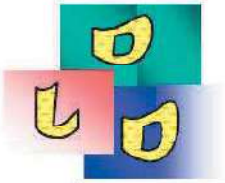
- The subtraction of unsigned binary numbers can be done most conveniently by means of complements.
- Remember that subtraction A-B can be done by taking 2's complement of B and adding it A. The 2's complement can be obtained by taking 1's compliment and adding one to the least significant pair of bits.
- The 1's complement can be implemented with inverters and a one can be added to the sum through the input carry.



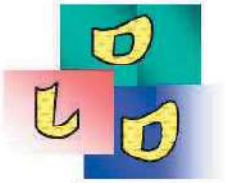
Binary Adder - Subtractor

- The addition and subtraction operations can be combined into one circuit with one common binary adder.
- This can be done by including an exclusive OR gate with each full adder.



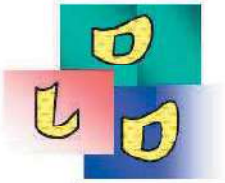


- In the circuit diagram, the mode input M controls the operations.
- When M=0, the circuit is an adder, and when M=1 the circuit becomes a subtractor.
- Each exclusive OR gate receives the input from M and one of the inputs of B.
- When M=0, then we have $B \oplus 0=B$. the full adder receives the value of B, the input carry is zero and the circuit performs A plus B.
- When M=1, we have $B \oplus 1=B^1$ and $c_0=1$. the B inputs are all complemented and a 1 is added through the input carry. The circuit performs the operation A plus two's complement of B (i.e., subtraction).



Decimal Adder

- Computers or calculators that perform arithmetic operations directly in the decimal number system represent decimal numbers in binary coded form.
- An adder for such a computer must employ arithmetic circuits that accept coded decimal numbers and present the results in the same code.
- For binary addition it is sufficient to consider a pair of significant bits together with a previous carry. A decimal adder requires a minimum of nine inputs and five outputs.

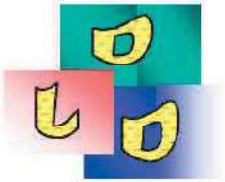


BCD Adder

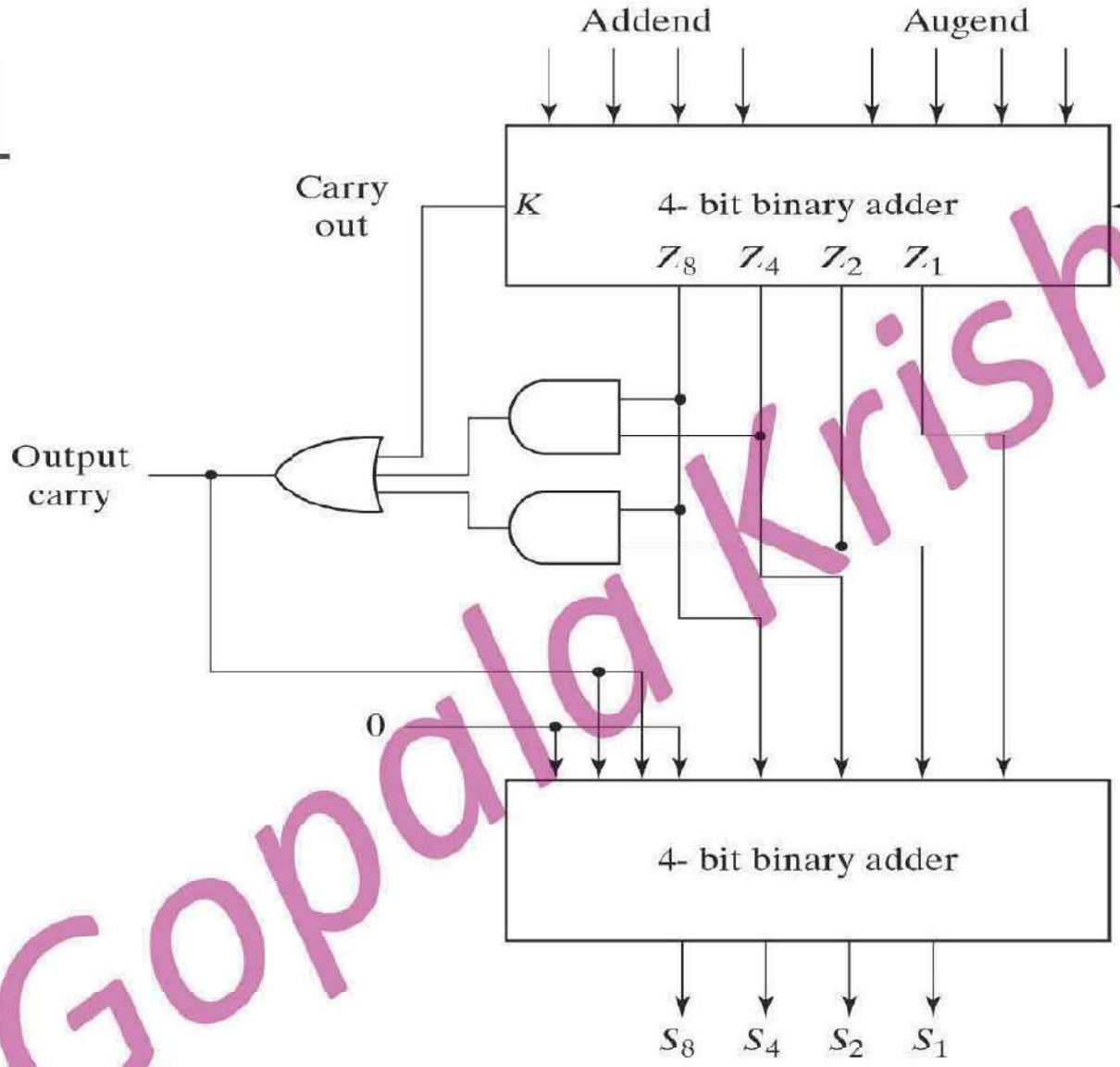
- Consider the arithmetic addition of two decimal digits in BCD, together with an input carry from a previous stage. Since each input digit does not exceed 9, the output sum cannot exceeds 19(9+9+1), the 1 in the sum being an input carry.
- Suppose we apply two BCD digits to a 4-bit binary adder. The adder will form the sum in binary and produce a result that ranges from 0 through 19.

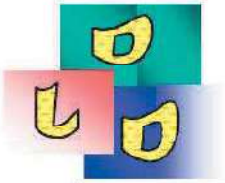
Derivation of BCD Adder

Binary sum					BCD Sum					
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	Decimal
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19



- The columns under the binary sum list the binary value that appears in the outputs of the 4-bit binary adder.
- The output sum of two decimal digits must be represented in BCD and should appear in the form listed in the columns under BCD Sum.
- The problem is to find a rule by which binary sum is converted to the correct BCD digit representation of the number in the BCD Sum.
- In examining the contents of the table, it is apparent that when the binary sum is equal to or less than 1001, the corresponding BCD number is identical, and therefore no conversion is required.
- When the binary sum is greater than 1001, we obtain a non valid BCD.
- The logic circuit that detects the necessary correction can be derived from the table entries.
  - It is obvious that a correction is needed when the binary sum has an output carry k=1.
  - The other six combinations from 1010 through 1111 that need a correction have a 1 in position z8.
  - To distinguish them from binary 1000 and 1001, which also have a 1 in z8, we specify further that either z4 or z2 must have a 1.
  - So $C=K+Z_8Z_4+Z_8Z_2$
- When C=1, it is necessary to add 0110 to the binary sum and provide an output carry for the next stage.



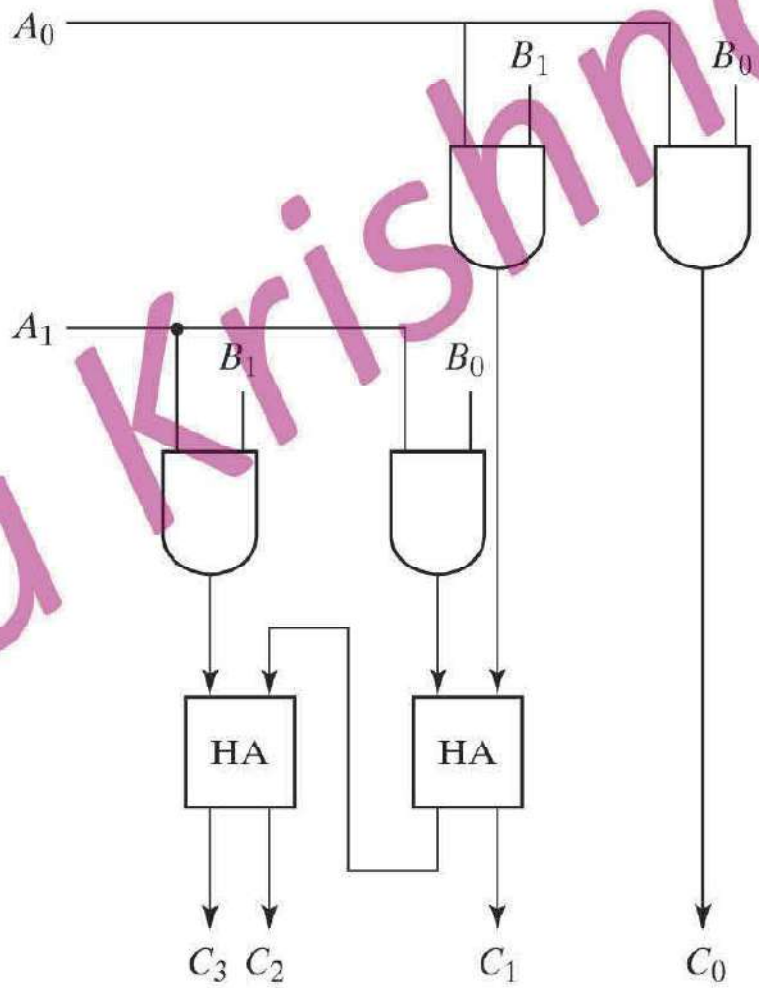


Binary Multiplier

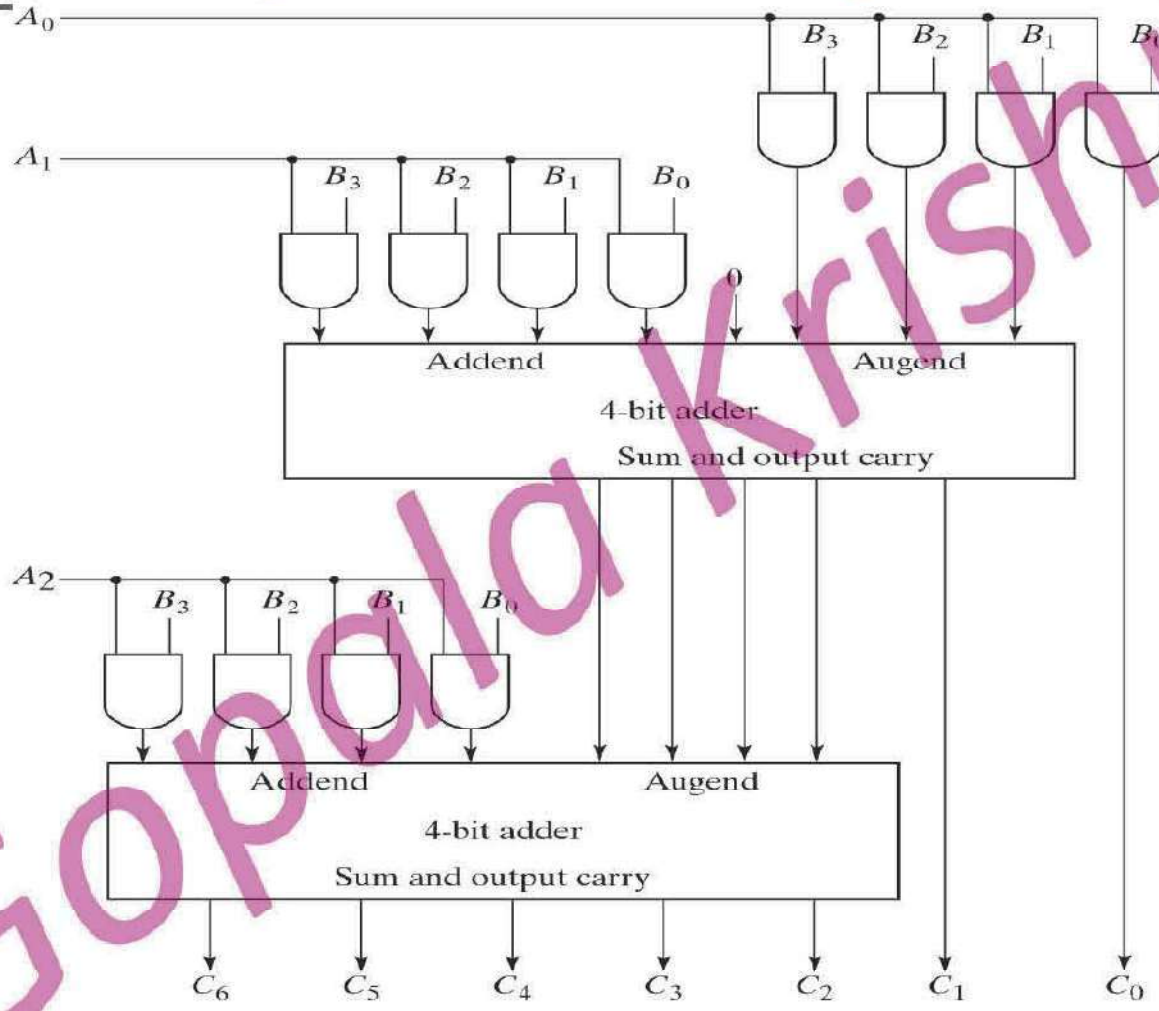
- Multiplication of binary numbers is performed in the same way as in decimal numbers. The multiplicand is multiplied by each bit of the multiplier starting from the least significant bit. Each such multiplication forms a partial product. Successive partial products are shifted one position to the left. The final product is obtained from the sum of the partial products.
- For example, consider the multiplication of two 2-bit numbers. The multiplicand bits are B1 and B0, the multiplier bits are A1 and A0, and the product is C3 C2 C1 C0. The first partial product is formed by multiplying A0 by B1B0.
- The multiplication of two bits such as A0 and B0 produces 1 if both bits are 1; otherwise it produces a 0. this is identical to AND Operation.

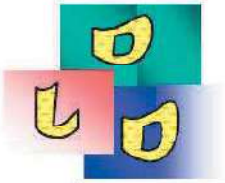
2- bit by 2- bit binary multiplier

$$
\begin{array}{cccc}
 & & B_1 & B_0 \\
 & & A_1 & A_0 \\
\hline
 & & A_0B_1 & A_0B_0 \\
 & A_1B_1 & A_1B_0 & \\
\hline
C_3 & C_2 & C_1 & C_0
\end{array}
$$



4- bit by 3-bit Binary Multiplier





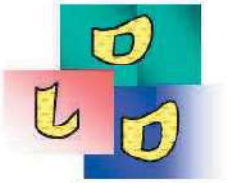
Magnitude Comparator

- The comparison of two numbers is an operation that determines if one number is greater than, less than, or equal to the other number.
- A magnitude comparator is a combinational circuit that compares two numbers, A and B, and determines their relative magnitudes.
- The outcome of the comparison is specified by three binary variables that indicate A>B, A=B, or A<B.
- The algorithm is a direct application of the procedure uses to compare the relative magnitudes of two numbers. Consider two numbers, A and B, with four digits each. Write the coefficients of the numbers with descending significance

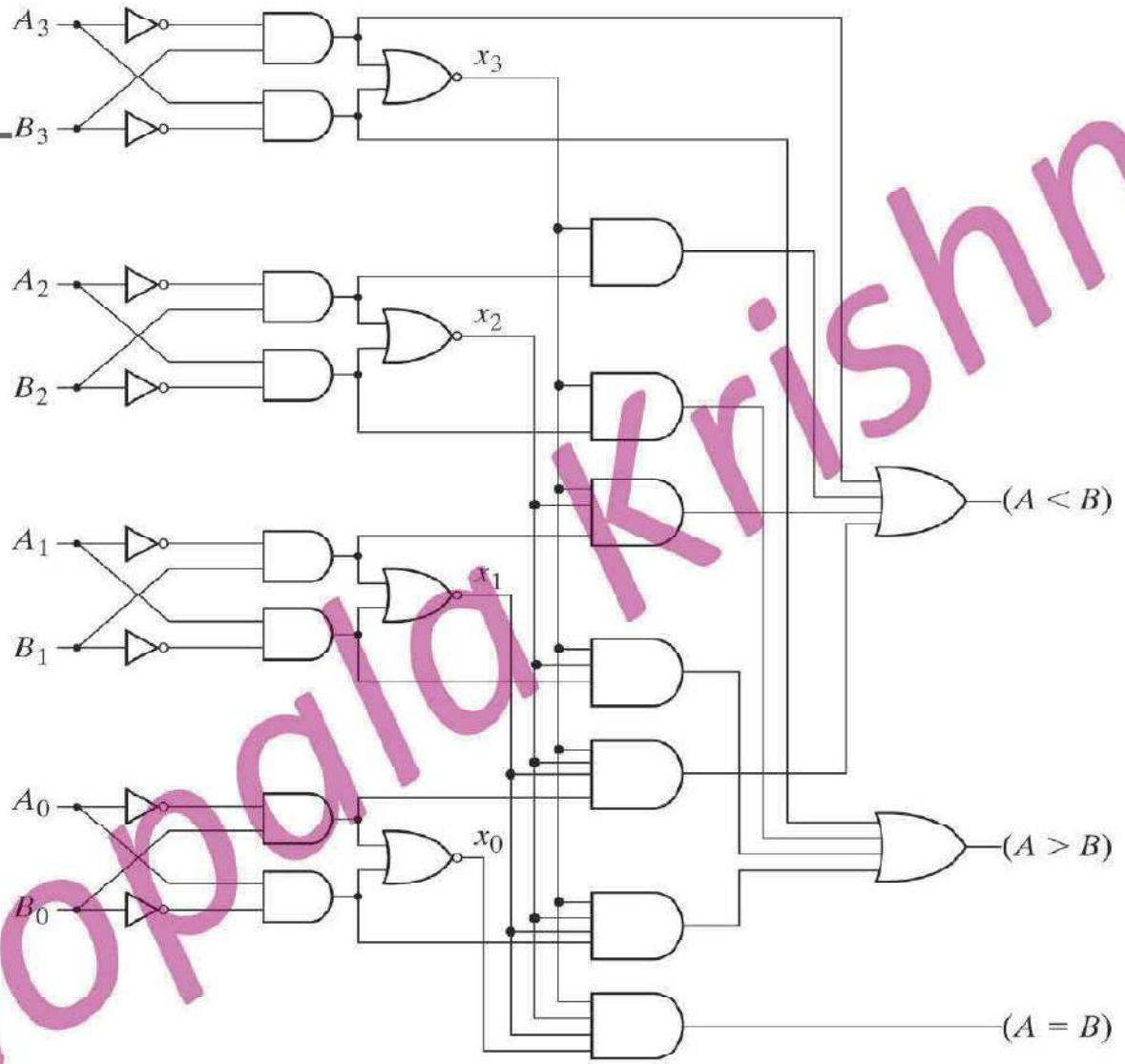
  $A=A_3A_2A_1A_0$

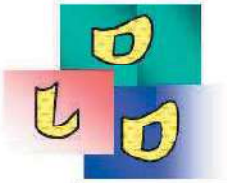
  $B=B_3B_2B_1B_0$
- Each subscripted letter represents one of the digits in the number. The two numbers are equal if all pairs of significant digits are equal:

  $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$, $A_0 = B_0$
- When the numbers are binary the digits are either 1 or 0, and the equality relation of each pair of bits an be expressed logically with an exclusive NOR function as $x_i=A_iB_i+A_i^1B_i^1$ for i=0,1,2,3.....where $x_i=1$ only if the pair of bits in position i are equal (i.e., if both are 1 or both are 0).



- The equality of two numbers, A and B, is displayed in a combinational circuit by an output binary variable that we designate by the symbol (A=B).
- $(A=B) = x_3x_2x_1x_0$
- $(A>B)=A_3B_3^1+x_3A_2B_2^1+x_3x_2A_1B_1^1+x_3x_2x_1A_0B_0^1$
- $(A<B)=A_3^1B_3+x_3A_2^1B_2+x_3x_2A_1^1B_1+x_3x_2x_1A_0^1B_0$

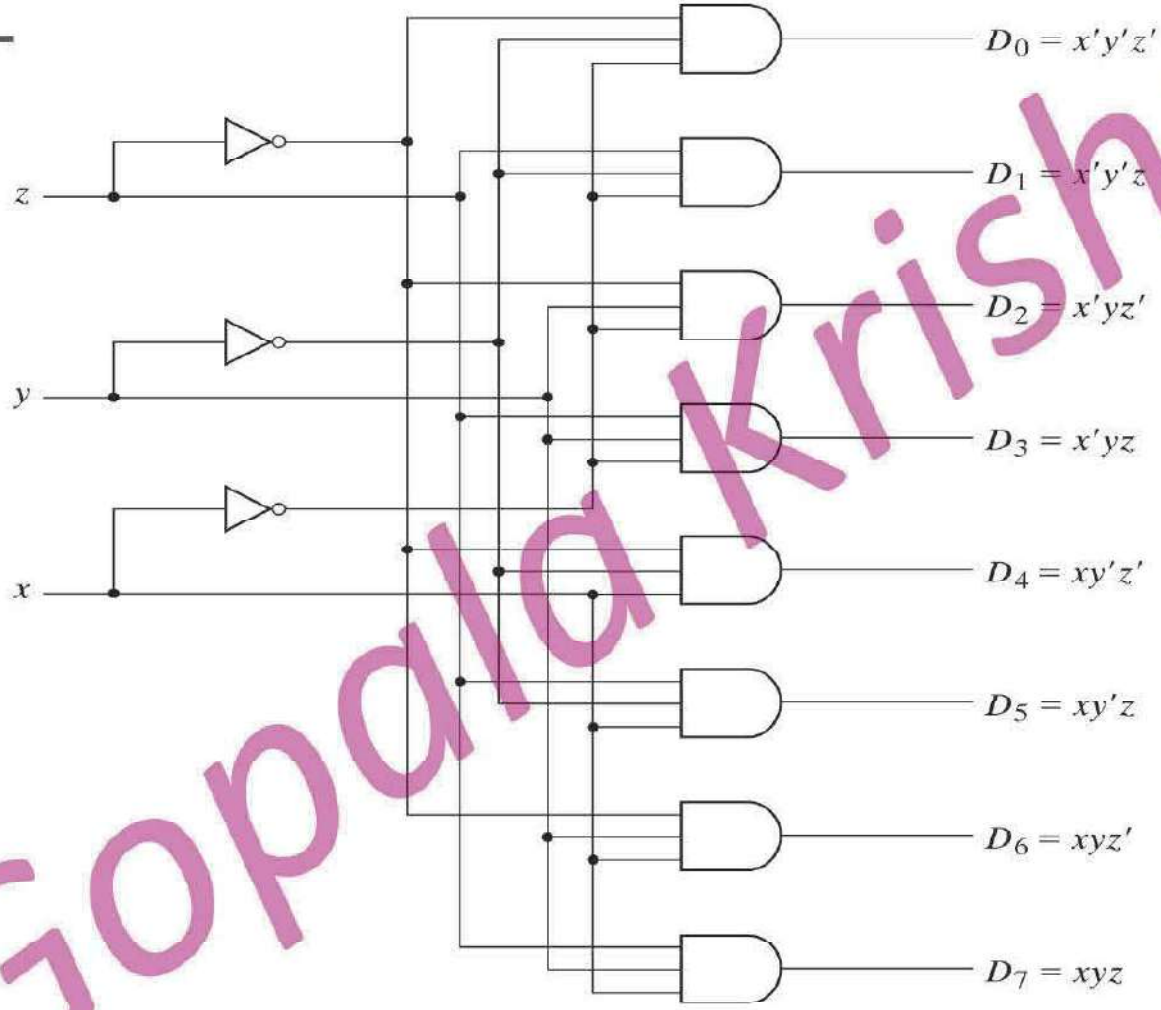


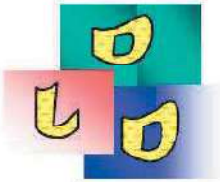


Decoders

- Discrete quantities of information are represented in digital system by binary codes. A binary code of n bits is capable of representing up to 2^n distinct elements of coded information. A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.
- The purpose of a decoder is to generate the 2^n minterms of n input variables.

3-to-8 Line Decoder

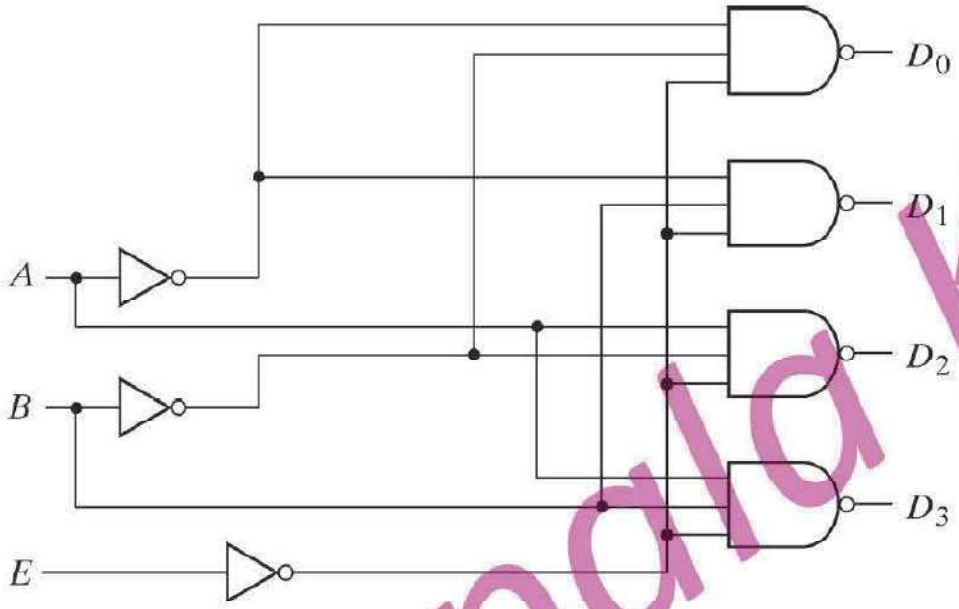




Truth table of a 3-to-8 Line Decoder

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

2-to-4 Line Decoder with Enable Input

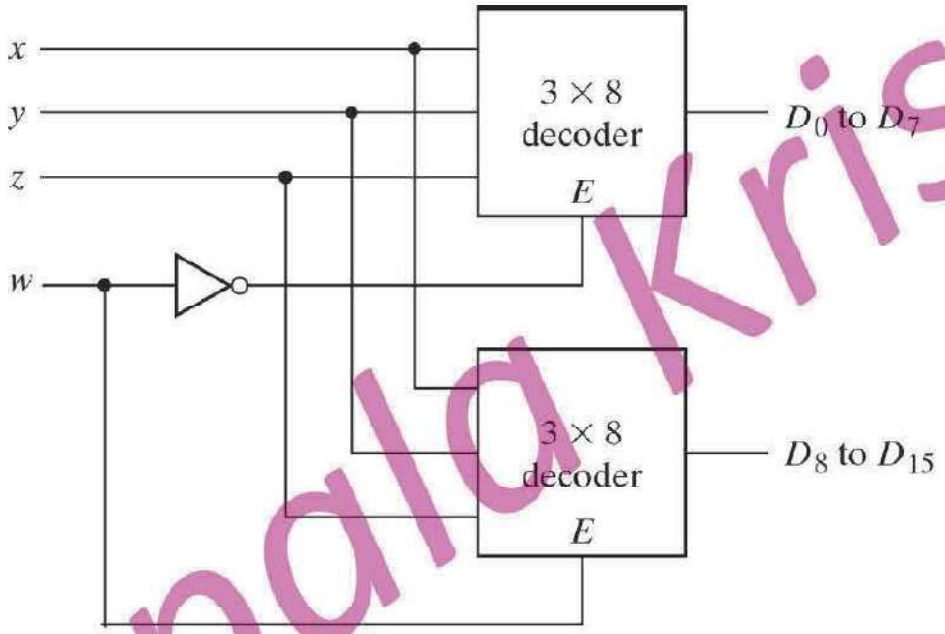


(a) Logic diagram

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

4x16 decoder with two 3x8 decoders



Combinational Logic Implementation with Decoders

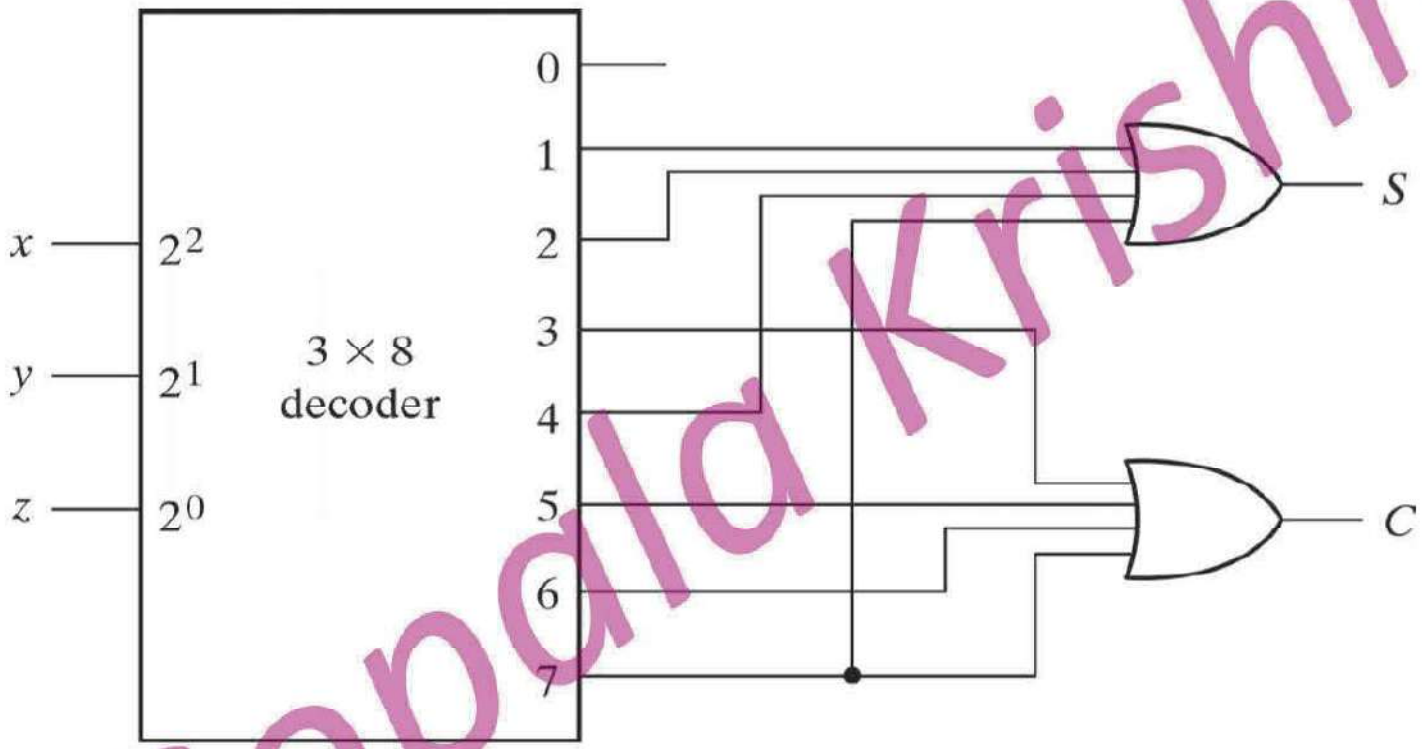
- A decoder provides the 2^n minterm of n input variables. Since any Boolean function can be expressed in sum of minterms, once can use a decoder to generate the minterms and an external OR gate to form the logical sum.
- In this way, any combinational circuit with n inputs and m outputs can be implemented with an n-to-2^n line decoder and m OR gates.
- For example, implementation of full adder circuit is as follows:

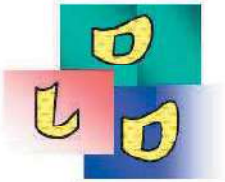
  $S(x,y,z)=\Sigma(1,2,4,7)$

  $c(x,y,z)=\Sigma(3,5,6,7)$

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

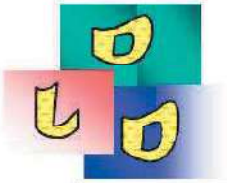
Implementation of Full Adder with a Decoder





Encoders

- An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n input lines and n output lines. The output line generates the binary code corresponding to the input value.
- An example of an encoder is the octal to binary encoder.



Octal to Binary Encoder

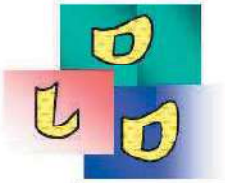
Inputs								outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

- This encoder can be implemented with OR Gates whose inputs are determined from the truth table.

$$Z=D_1+D_3+D_5+D_7$$

$$Y=D_2+D_3+D_6+D_7$$

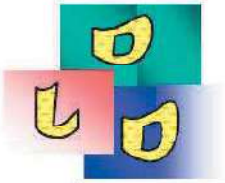
$$X=D_4+D_5+D_6+D_7$$



Priority Encoder

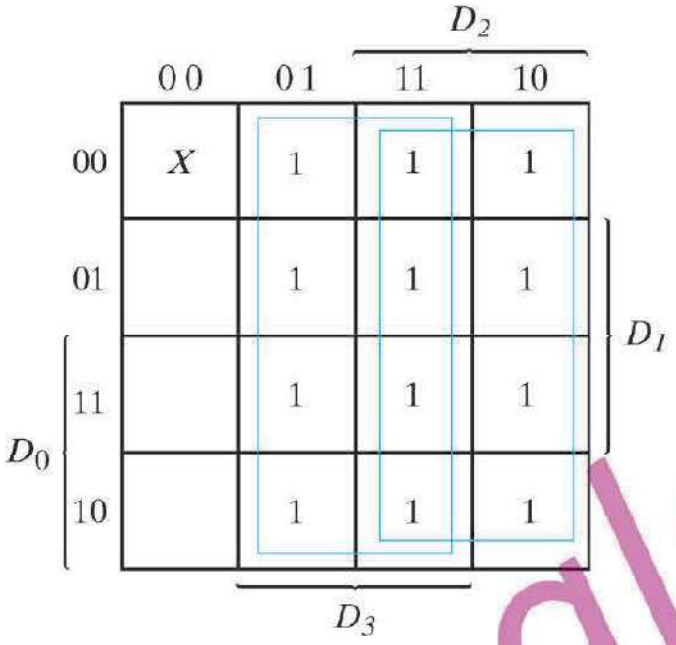
- A priority encoder is an encoder circuit that includes the priority function. The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

Inputs				Outputs		
D_0	D_1	D_2	D_3	x	y	v
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

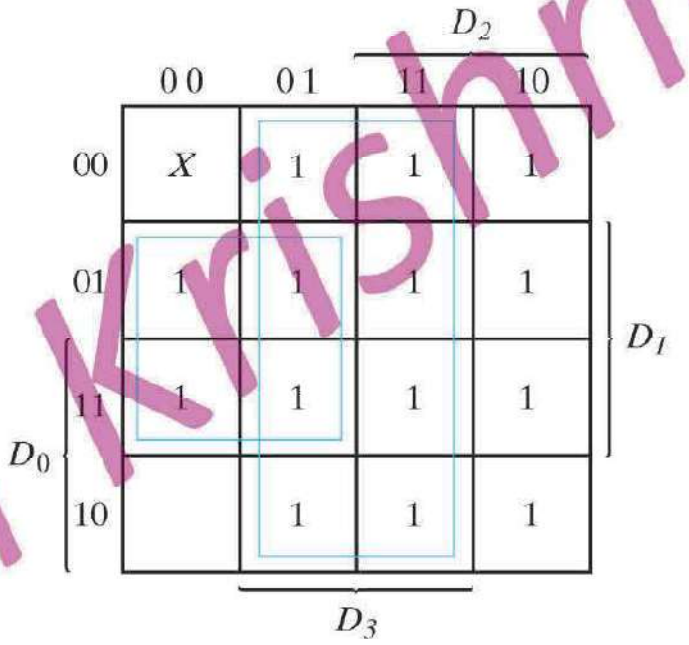


- In addition to the two outputs x and y, the circuit has a third output designated by V; this is a valid bit indicator that is set to 1 when one or more inputs are equal to 1. if all inputs are 0, there is no valid input and V is equal to 0.
- The other two outputs are not inspected when V equals 0 and are specified as don't care conditions.

Maps for Priority Encoder

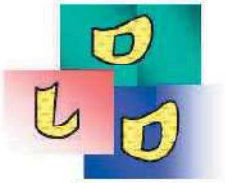


$x = D_2 + D_3$



$y = D_3 + D_1D'_2$

$x=D_2+D_3$
$y=D_3+D_1D_2^1$
$v=D_0+D_1+D_2+D_3$

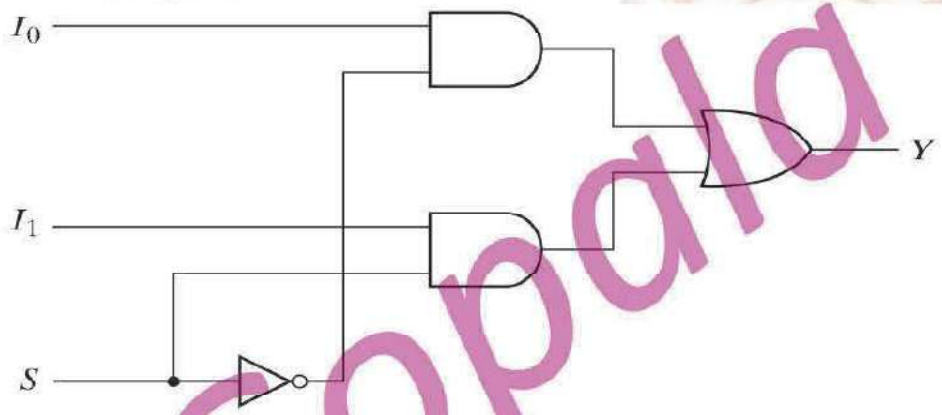


Multiplexers

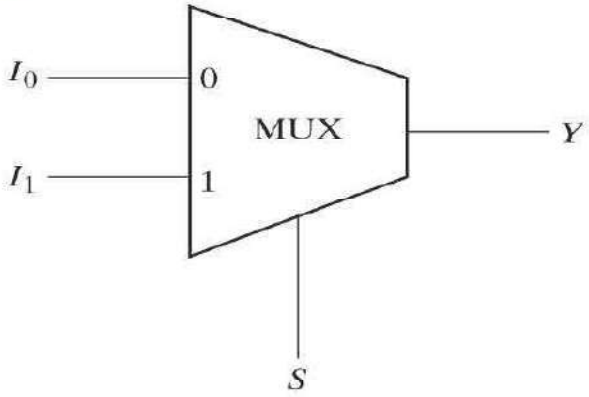
- A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.
- The selection of a particular input line is controlled by a set of selection lines.
- Normally there are 2^n input lines and n selection lines whose bit configurations determine which input line is selected.
- The multiplexer acts like an electronic switch that selects one of the sources.
- A multiplexer is also called as a data selector, since it selects one of many inputs and steers the binary information to the output line.
- The multiplexer is often labeled as MUX in block diagrams.

2-to-1-Line Multiplexer

- A 2-to-1 line multiplexer connects one of two 1-bit sources to a common destination.
- The circuit has two data input lines, one output line, and one selection line S.
- When S=0, the upper AND gate is enabled and I0 has a path to the output.
- When S=1, the lower AND gate is enabled and I1 has a path to the output.

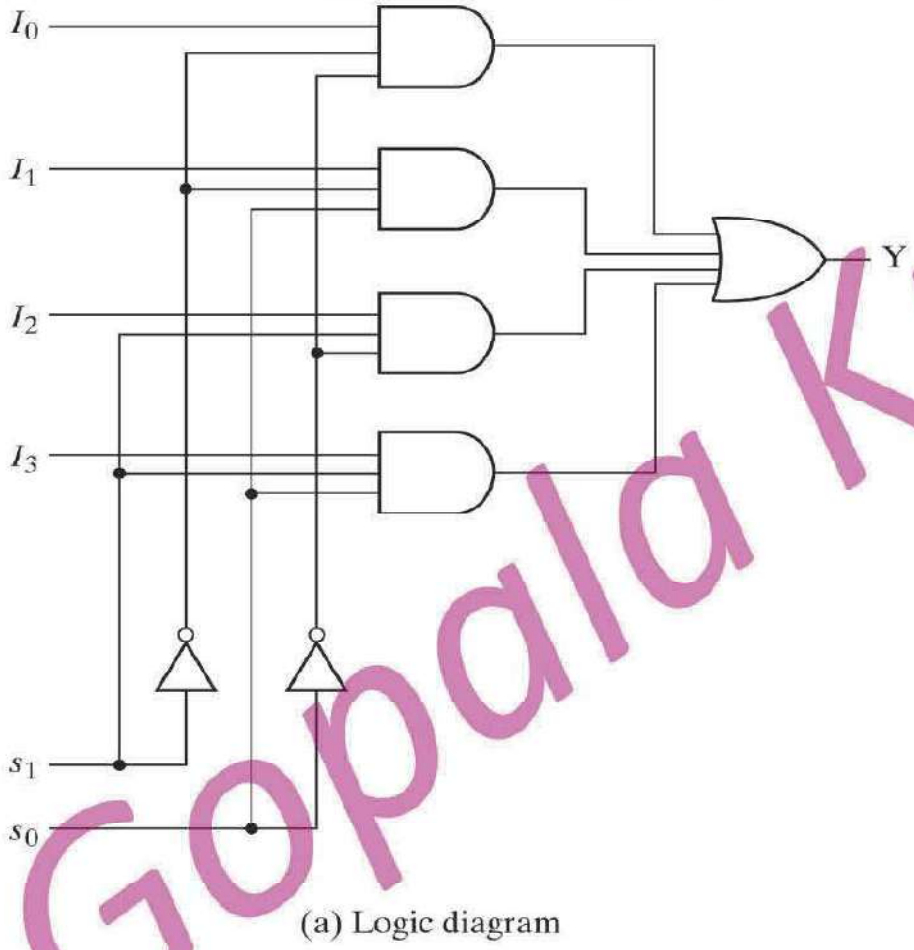


(a) Logic diagram



(b) Block diagram

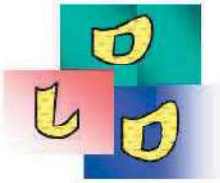
4-to-1-Line Multiplexer



(a) Logic diagram

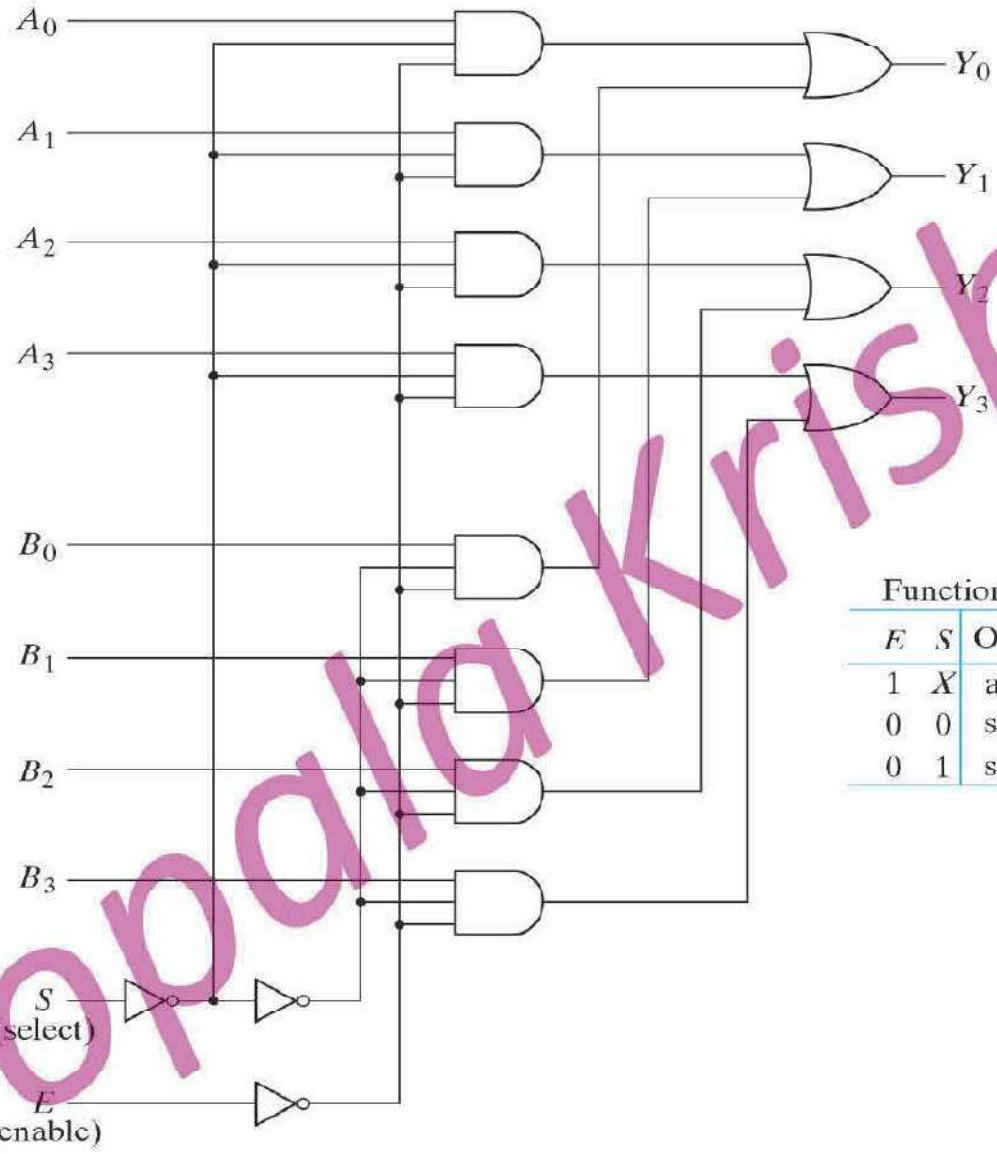
s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table



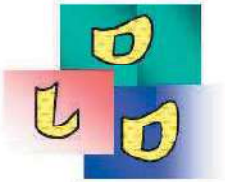
Quadruple 2-to-1 Line Multiplexer

- Multiplexer circuit can be combined with common selection inputs to provide multiple-bit selection logic.
- For example, a quadruple 2-to-1 line multiplexer has four multiplexers, each capable of selecting one of two input lines



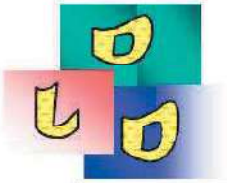
Function table

E	S	Output Y
1	X	all 0's
0	0	select A
0	1	select B



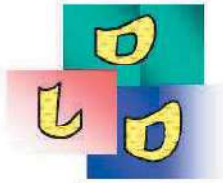
Boolean Function Implementation Using MUX

- The method of implementing a boolean function with a MUX:
  - The boolean function is first listed in a truth table.
  - A boolean function with n variables can be implemented with a multiplexer that has (n-1) selection inputs.
  - The first (n-1) variables of the function are connected to the selection inputs of the multiplexer.
  - The remaining single variable of the function is used for the data inputs.
  - For each combination of the selection variables, we evaluate the output as a function of the left out variable.
  - The function can be 0,1, the variable or the complement of the variable.
  - These values are then applied to the data inputs in the proper order.



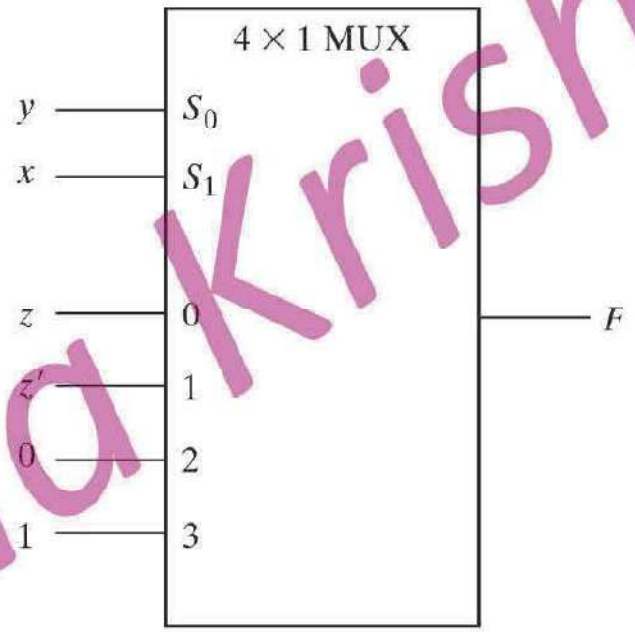
Example 1

- Implement the following boolean function with a Multiplexer: $F(x,y,z)=\Sigma(1,2,6,7)$
- This function can be implemented with a 4-to-1 line multiplexer.
- The two variables x and y are connected to the selection lines in that order.
- The values of the data input lines are determined from the truth table of the function.



x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table

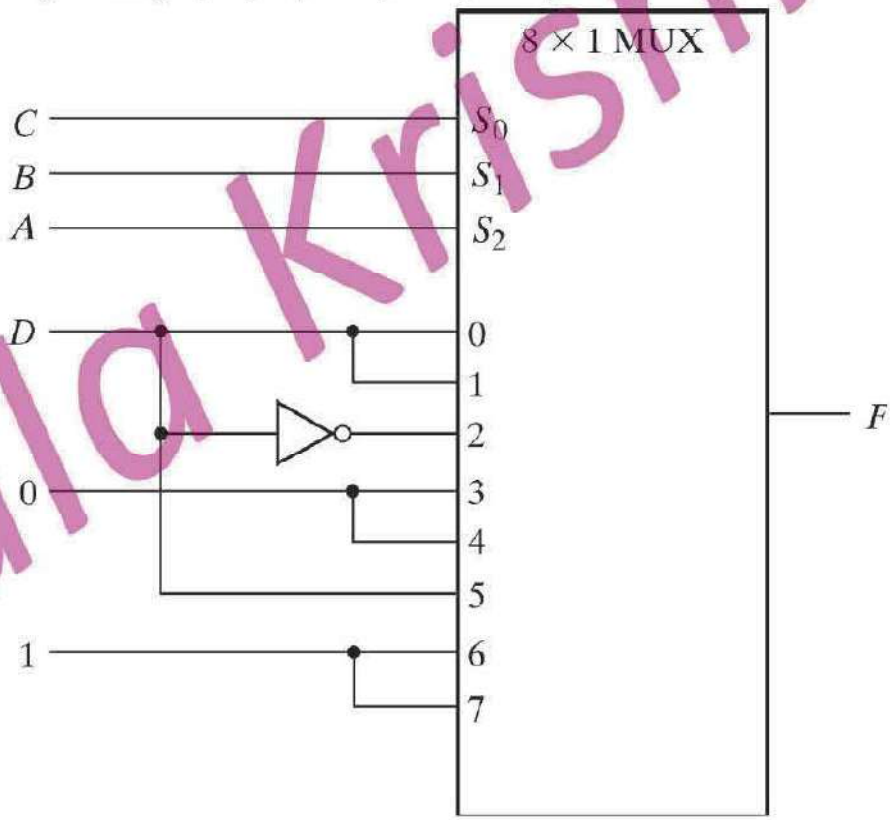


(b) Multiplexer implementation

Example 2

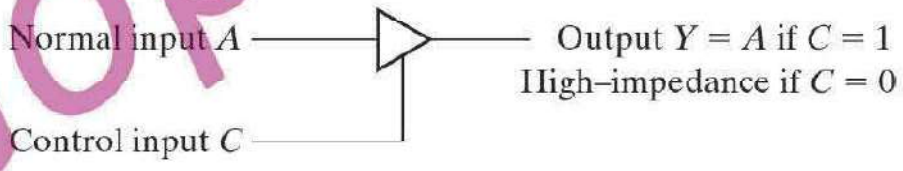
- Implement the following boolean function with a Multiplexer: F(A,B,C,D)=∑(1,3,4,11,12,13,14,15)

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	

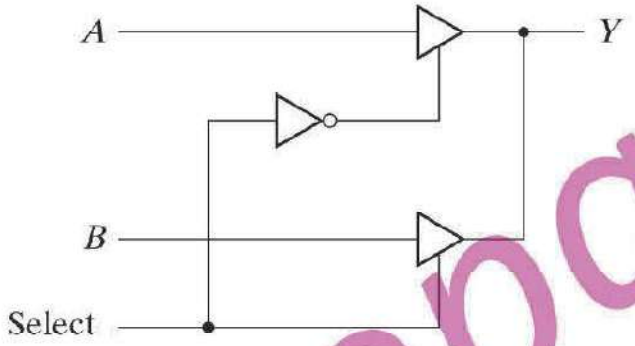


Three State Gates

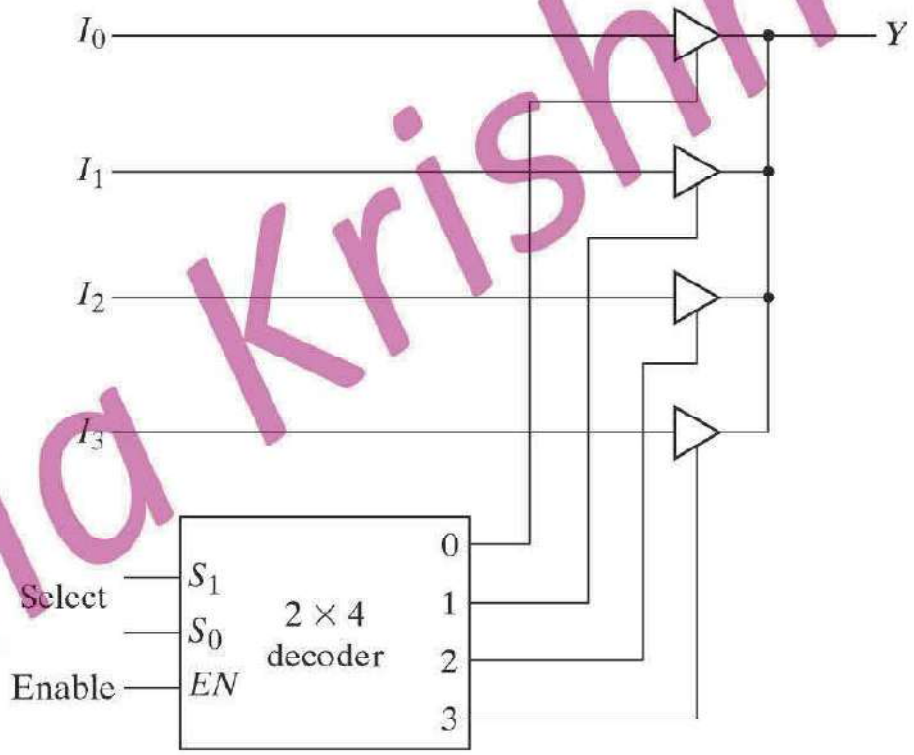
- A three state gate is a digital circuit that exhibits three states.
- Two of the states are signals equivalent to logic 1 and 0 as in conventional gate.
- The third state is a high impedance state. This behaves like a an open circuit , which means that the output appears to be disconnected and the circuit has no logic significance.
- The graphic symbol of a three state buffer gate has input control line entering the bottom of the gate symbol. The buffer has normal input, an output and a control input that determines the state of the output.



Multiplexers with Three State Gates



(a) 2-to-1- line mux



(b) 4 - to - 1 line mux