

Unit 3

Half adder, Full adder, Half subtractor, Parallel adder, study of Multiplexer (4:1) and Demultiplexer (1:4), Encoder (Decimal to BCD encoder and 3 bit priority encoder), Decoder (3 to 8 line decoder using gates only).

Introduction

- Digital logic circuits are classified into two main categories namely Combinational and sequential circuits.
- Combinational logic is a type of digital logic which is implemented using logic gates. **The output of combinational circuit is the function of combination of present inputs only.**
- Whereas in sequential circuit the output depends not only on the present input but also on the previous inputs i.e. history of inputs.
- In other words, **a sequential circuit has memory while combinational circuit does not.**
- Combinational circuit is able to generate an output simply from knowing what current input values are. Combinational circuits do not need to know the history of past inputs.

Combinational Circuit basics

- A combinational circuit consists of input variables, logic gates, and output variables. The logic gates accept signals from the inputs and generate signals to the outputs.
- This process transforms binary information from the given input data to the required output data.
- Obviously, both input and output data are represented by binary signals, i.e. they exist in two possible values, one representing logic-1 and the other logic-0. A block diagram of a combinational circuit is shown in Fig. 1.

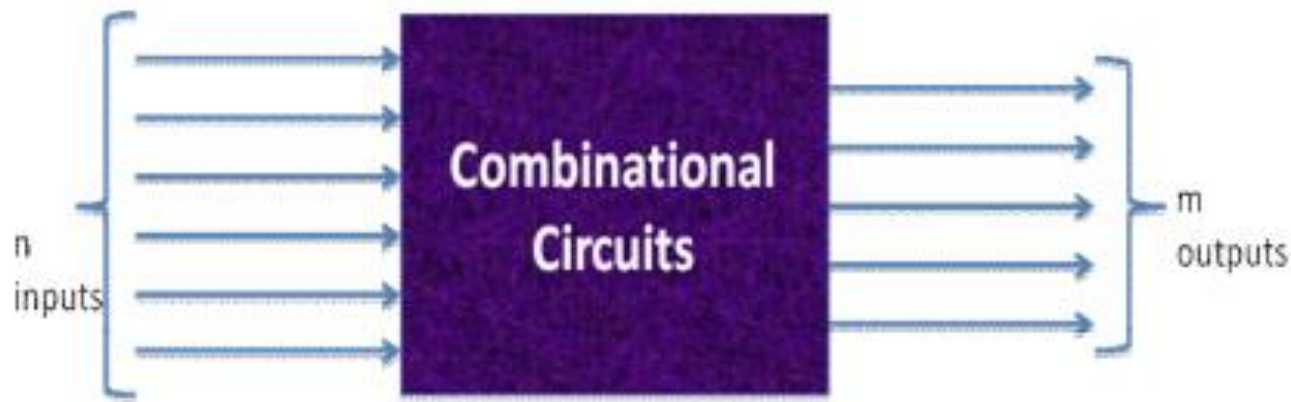


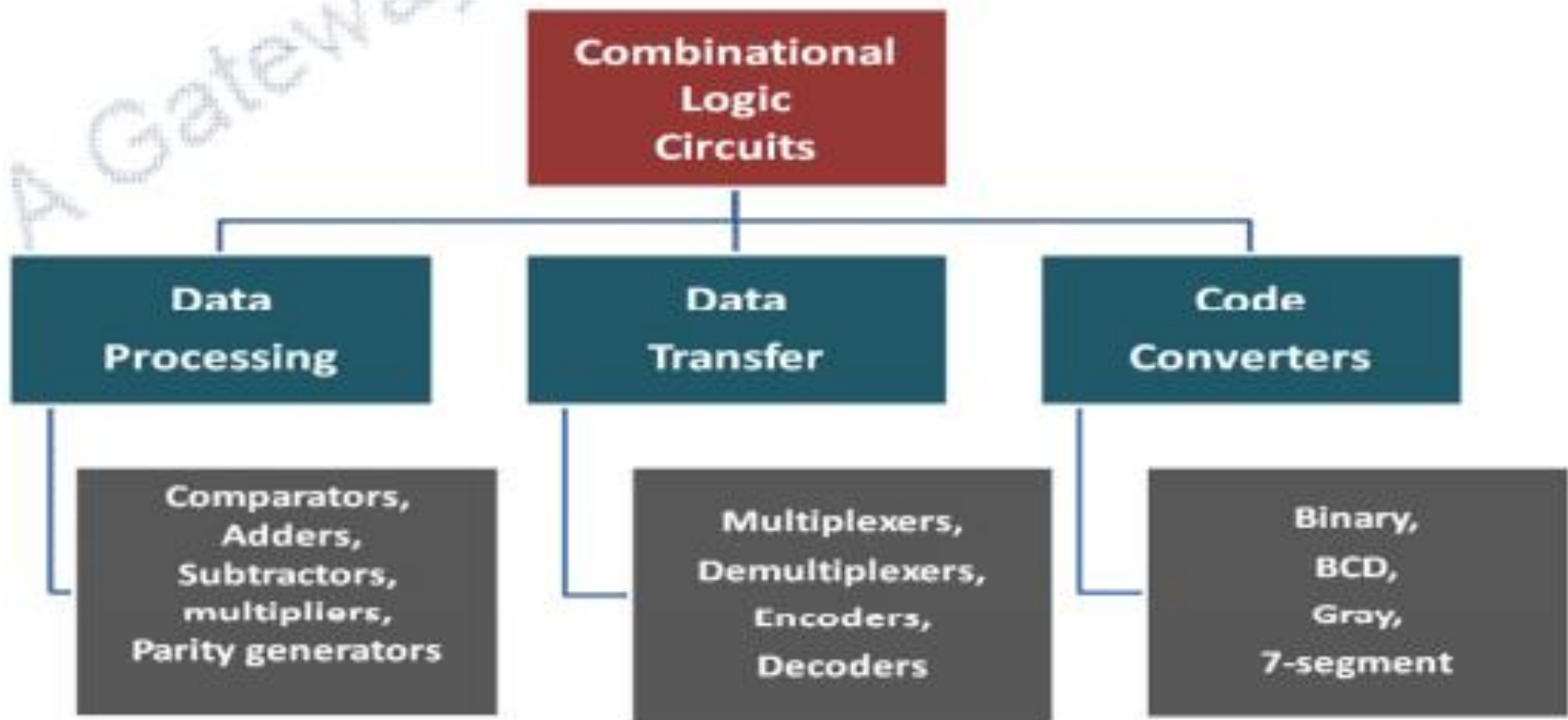
Figure 1: Combinational logic circuit

Combination Logic Circuits are made up from basic gates (AND, OR, NOT) or universal gates (NAND, NOR) gates that are "combined" or connected together to produce more complicated logic circuits.

These logic gates are the building blocks of combinational logic circuits. In these circuits "the outputs at any instant of time depends on the inputs present at that instant only."

For n input variables, there are 2^n possible combinations of binary input values. For each possible input combination, there is one and only one possible output combination. A combinational circuit can be described by m Boolean functions, one for each output variable. Each output function is expressed in terms of the n input variables.

Types of combinational circuits



Types of combinational circuits

- There are three major types of combinational circuits:
- (a) Data processing circuits that process or transform data e.g. Adder, subtractor, comparator, multiplier etc.
- (b) Data transfer circuits to control the flow of logic e.g. Multiplexer, demultiplexer, encoder, decoder etc.
- (c) Code converters for conversion from binary to any other code and vice versa e.g. Binary, BCD, Gray, 7-segment converters

Design procedure of combinational circuits

For the design of Combinational digital circuits basic gates (AND, OR, NOT) or universal gates (NAND, NOR) are used.

The design of combinational circuits begins with problem definition and ends with a logic circuit or set of Boolean expressions.

1. State the problem clearly.
2. Determine the available inputs and required outputs from problem.
3. Assign a letter symbols to input and outputs of the circuits.
4. Derive the truth table that relates inputs and outputs.
5. Write the Boolean functions for the output and further simplify to get reduced number of variables.
6. Draw the logic diagram.

Arithmetic Circuits

- Nowadays, computer is the most important part of our life. It consists of three following important units: (i) Input / Output devices, (ii) Memory and (iii) CPU.
- CPU is the heart of computer. It is central processing unit. Its basic element is ALU i.e. Arithmetic Logic Unit. It performs the arithmetic operations like addition, subtraction, multiplication, division etc.
- The logic operations like AND, OR, NOT etc. are performed. But addition and subtraction are basic operations. These are performed by arithmetic section of ALU.
- We are very familiar with decimal addition but computer does not do decimal addition. It converts the decimal number into binary numbers and then adds them. The other operations are multiple additions or subtraction. Subtraction is also an addition. Therefore addition is the most important operation in arithmetic circuits.
- In this module, you will learn about the basics of binary addition and subtraction

Half Adder

The most basic arithmetic operation is the addition of two binary digits. This simple addition

consists of four possible elementary operations:

$$0 + 0 = 0,$$

$$0 + 1 = 1,$$

$$1 + 0 = 1, \text{ and}$$

$$1 + 1 = 10$$

A combinational circuit that performs the addition of two bits is called a half adder. One that performs the addition of three bits (two significant bits and a previous carry) is a full adder.

The names of the circuits stem from the fact that two half adders can be employed to implement a full adder.

Let us consider x and y as the inputs to the half adder where as Sum and Carry are the outputs. The Truth table, K-maps for sum and carry along with the logic diagram are shown in figure 1.

Truth table for half adder

x	y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-map for sum

	x'	x
y'		1
y	1	

Boolean expression for Sum

$$\text{Sum} = x'y + xy'$$

K-map for carry

	x'	x
y'		
y		1

Boolean expression for Carry

$$\text{Carry} = x.y$$

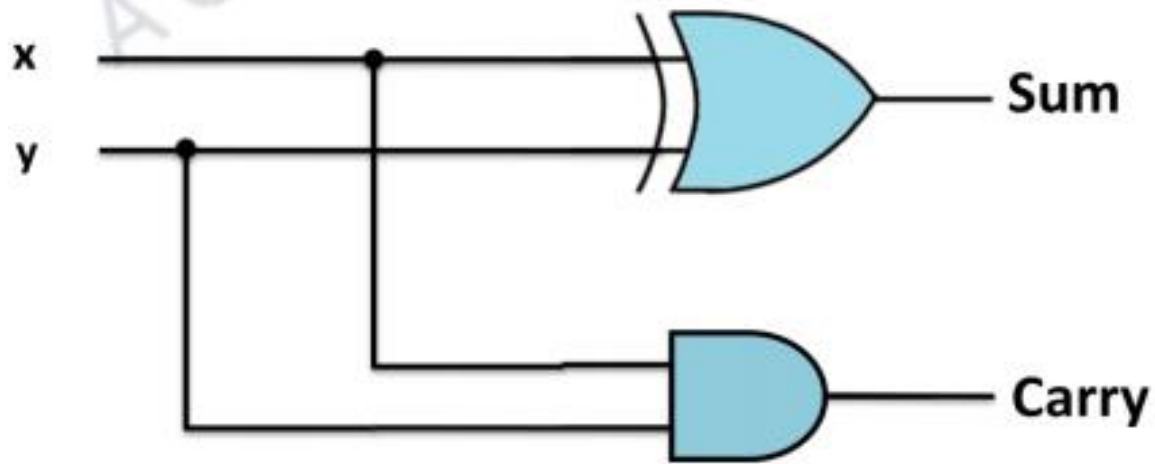


Figure-1: Truth table, K-maps for sum and carry along with the logic circuit for half adder

A= 0111

B= 0101

-----1-----

0

- Let us consider the truth table of half adder. There are four possible combinations for the inputs.
- When both x and y are at logic '0' then sum as well as carry are at logic '0'. If any one of the input either x or y is '1' the sum is at '1' and carry is '0'. When both the inputs are '1' then the sum is '0' and carry becomes '1'.
- The k-map for sum and carry need to be prepared separately. The k-map is of 2-variables for sum and carry as shown in figure -1.
- After plotting the outputs into the k-map the two Boolean expressions for sum and carry are obtained. These equations can be used to draw the logic diagram for half adder.
- The equation for sum indicate, there is a need of an EXOR gate and AND gate for the carry.

Full Adder

- Full adder is a one of the most important blocks of an ALU. One that performs the addition of three bits (two significant bits and a previous carry) is a full adder.
- The names of the circuits stem from the fact that two half adders can be employed to implement a full adder. Let us begin
- with the preparation of Truth table for Full adder.

Truth table for Full Adder

INPUTS			OUTPUT	
X	Y	Z	SUM	CARRY
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

x	y	z	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\text{Sum} = xy'z' + x'y'z + xyz + x'yz'$$

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

consider $(y'z + yz') = m = y \text{ xor } z$

$$= x \cdot m' + x' \cdot m$$

$$= x \text{ xor } m$$

K-map for Sum

	$y'z'$	$y'z$	yz	yz'
x'		1		1
x	1		1	

K-map for Carry

	$y'z'$	$y'z$	yz	yz'
x'			1	
x		1	1	1

Boolean expressions for Sum

$$\text{Sum} = z \oplus (x \oplus y)$$

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

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

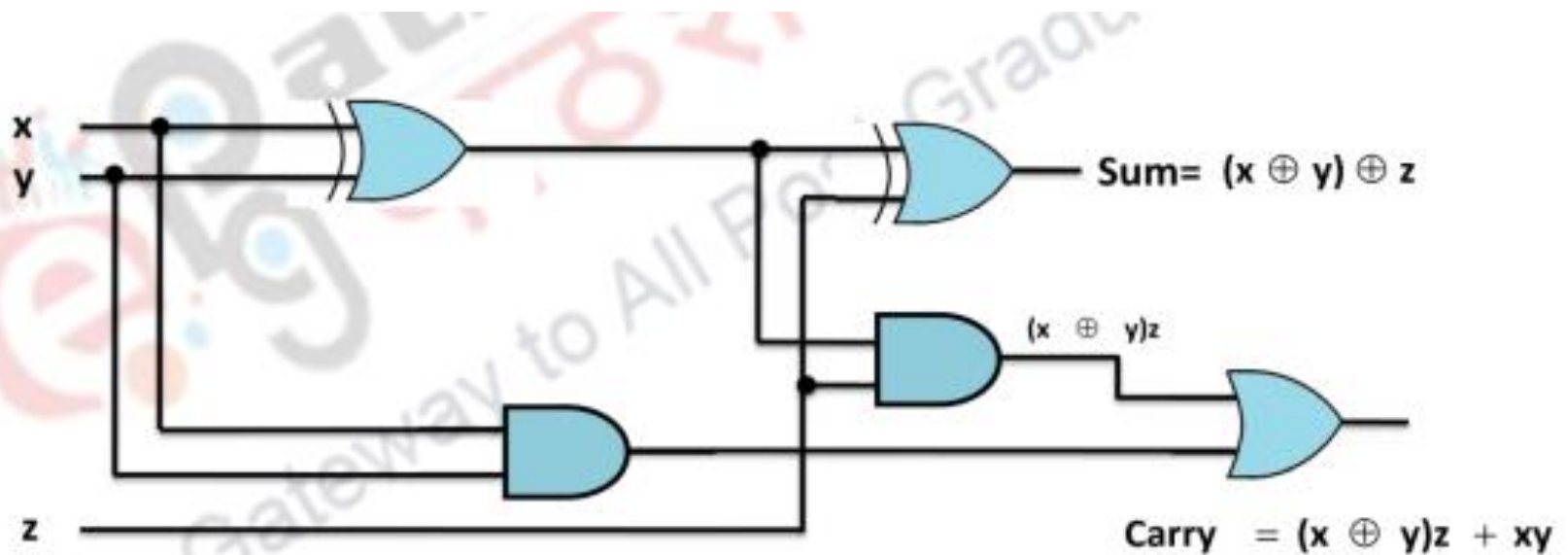
$$= xy'z' + x'yz' + xyz + x'y'z$$

Boolean expressions for Carry

$$\text{Carry} = xy + xz + yz$$

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

$$= xy + z(x \oplus y)$$



Let us consider the truth table of full adder. There are eight possible combinations for the inputs. When all the inputs are at logic '0' then sum as well as carry are at logic '0'. If any one of the input either x or y or z is '1' the sum is at '1' and carry is '0'. When any two of the inputs are '1' then the sum is '0' and carry becomes '1'. When all the inputs are '1' then the sum as well as carry is at logic '1'.

- The k-map for sum and carry need to be prepared separately. The k-map is of 3-variables for sum and carry as shown in figure - 2. After filling the outputs into the k-map and simplification, the two Boolean expressions for Sum and Carry are obtained. These equations can be used to draw the logic diagram for full adder.
- The full adders using two half adders and OR gate is obtained by expressions of sum and carry. In short, we require two EXOR gates, 2 AND gates and one OR gate to obtain the logic diagram for Full Adder.

Binary Adder/Parallel Adder

- Multiple bits binary adder is obtained by using number of full adders connected in cascade. For the first full adder the least significant bit of Augend and Addend along with the carry in are the inputs.
- Its outputs are sum and carry out. The carry out is used as carry in for the next full adder.
- Other inputs for the next full adder are the next significant bits. It again generates the sum and carry.
- The process is repeated till the last MSB. Figure – 3 indicates the logic diagram for 4-bit binary adder.

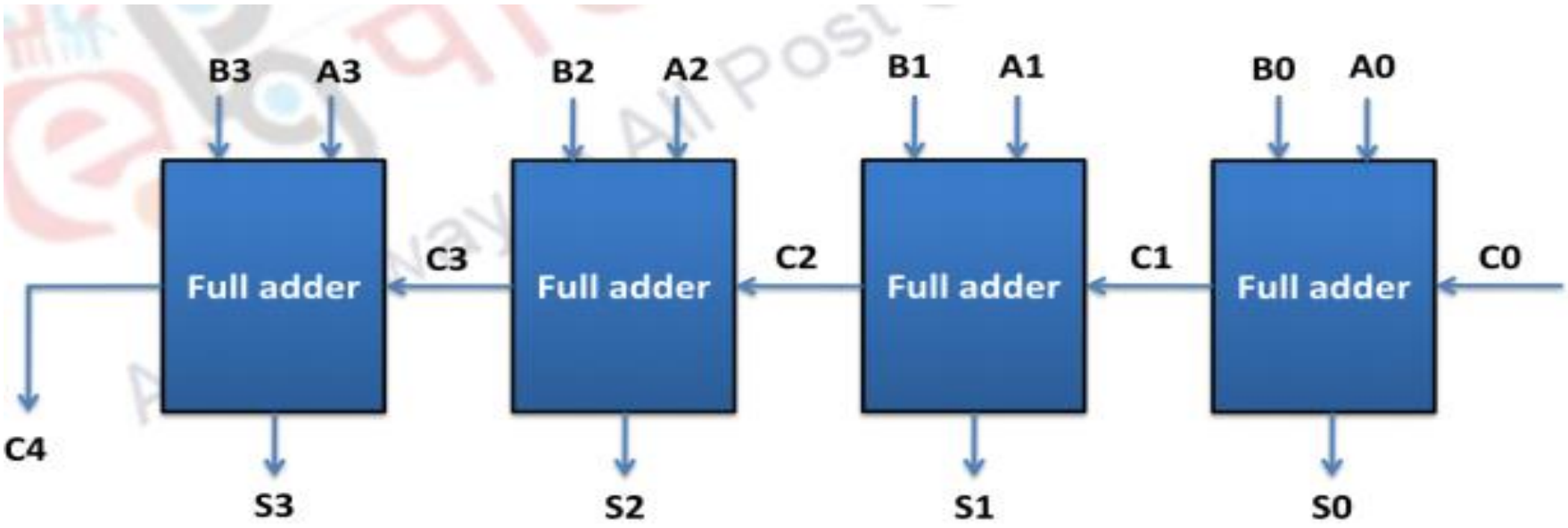
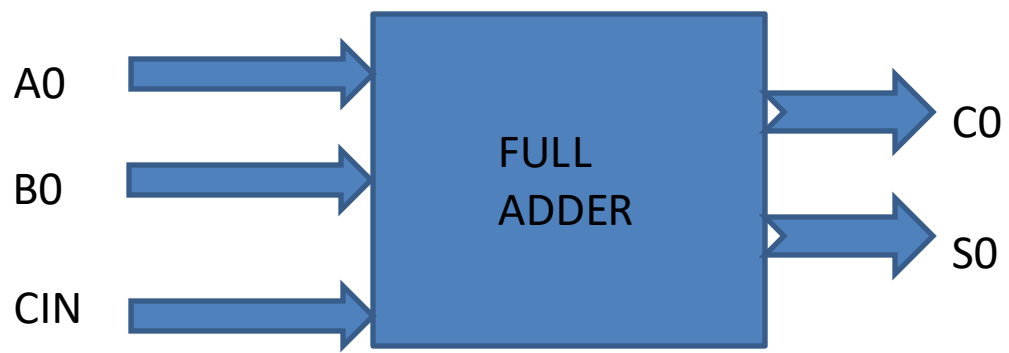


Figure 3: 4 bit binary adder

$A = A_3A_2A_1A_0$
 $B = B_3B_2B_1B_0$
 ---C3-C2C1-----
 C4 S4 S3 S1 S0



Let us consider the example of 4-bit binary addition. Let us take the decimal numbers 11 and 3 for the addition.

Augend	11
Addend	3

Sum	14

The binary addition for these numbers is shown below.

Augend (Ai)	1	0	1	1
Addend (Bi)	0	0	1	1
Input carry (Ci)	0	1	1	0
Sum (Si)	1	1	1	0
Output carry (Ci+1)	0			

Note that the result 1110 in binary is equivalent to decimal 14. So the logic circuit performs the addition of two numbers in binary.

5. Binary subtractor

Let us now begin with process of subtraction. This simple subtraction consists of four possible elementary operations:

$$0 - 0 = 0,$$

$$0 - 1 = 1 \text{ with borrow } 1,$$

$$1 - 0 = 1,$$

$$1 - 1 = 0$$

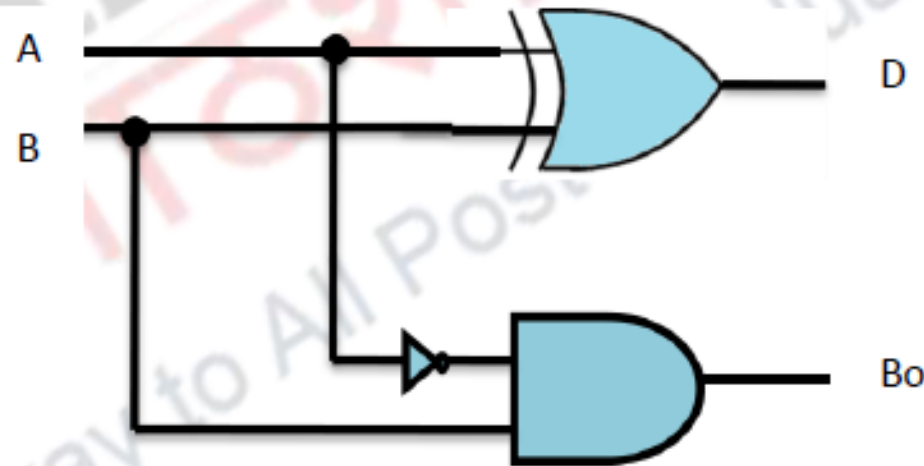
A combinational circuit that performs the subtraction of two bits is called a **half subtractor**. One that performs the subtraction of three bits (two significant bits and a previous borrow) is a

full subtractor. The names of the circuits stem from the fact that two half subtractors can be employed to implement a full subtractor.

Half Subtractor

Let us consider the truth table of half subtractor. There are four possible combinations for the inputs. When both x and y are at logic '0' then difference as well as borrow are at logic '0'. If $A=0$ and $B=1$ then difference is '1' and borrow is also at '1'. When $A=1$ and $B=0$ then the difference is 1 and borrow is 0. When both the inputs are '1' then the both difference and borrow are at logic '0'.

From the truth table it is clear that, we need additional NOT gate while implementing the borrow output. There is a need of an EXOR gate and AND gate for implementation.



A	B	Difference (D)	Borrow (Bo)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Figure-4: Logic diagram and truth table of Half Subtractor

Full Subtractor

One that performs the subtraction of three bits (two significant bits and a previous borrow) is a **full subtractor**. The names of the circuits stem from the fact that two half subtractors can be employed to implement a full subtractor.

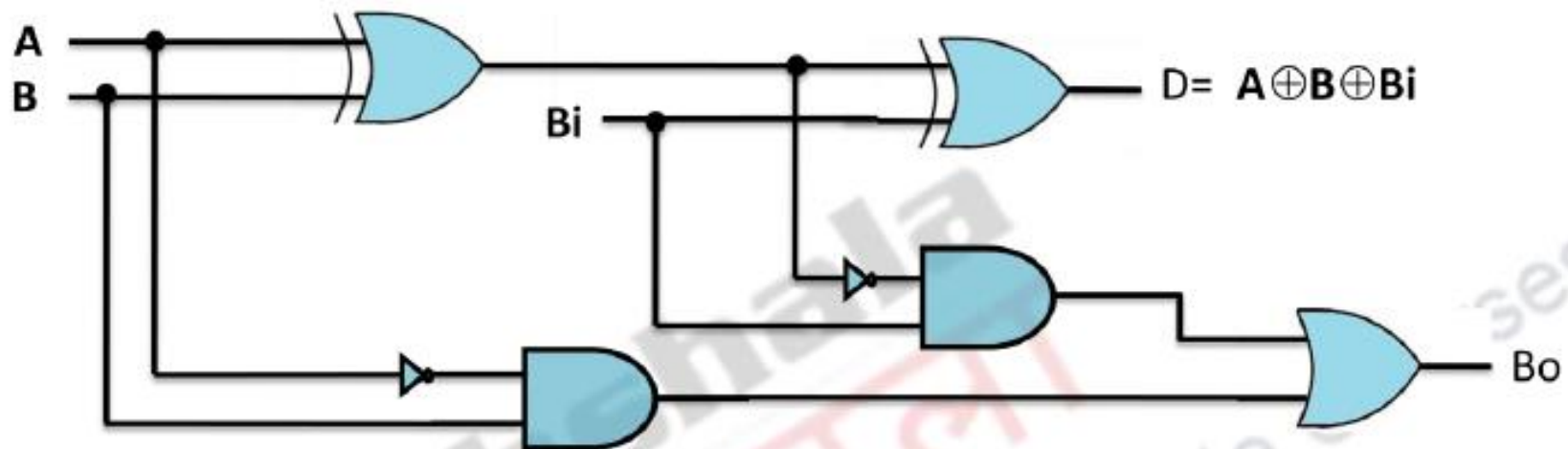
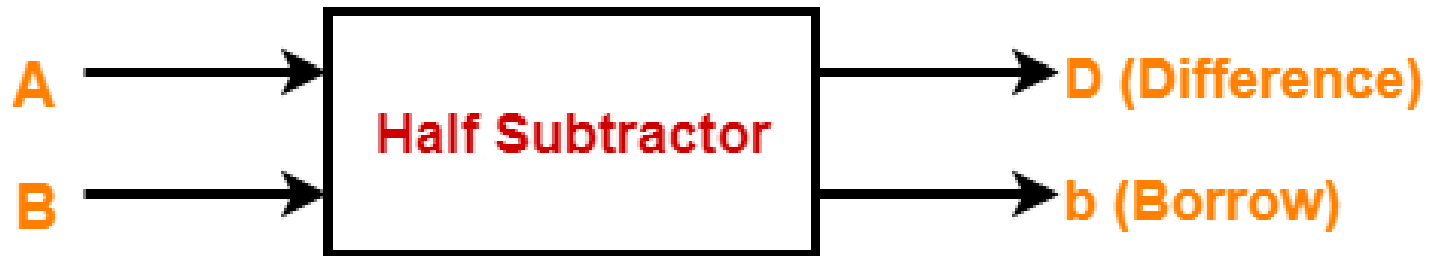


Figure-5: Full Subtractor

Half Subtractors

Half Subtractor-

- Half Subtractor is a combinational logic circuit.
- It is used for the purpose of subtracting two single bit numbers.
- It contains 2 inputs and 2 outputs (difference and borrow).



Half Subtractor Designing-

Step-01:

Identify the input and output variables-

- Input variables = A, B (either 0 or 1)
- Output variables = D, b where D = Difference and b = borrow

Step-02:

Draw the truth table-

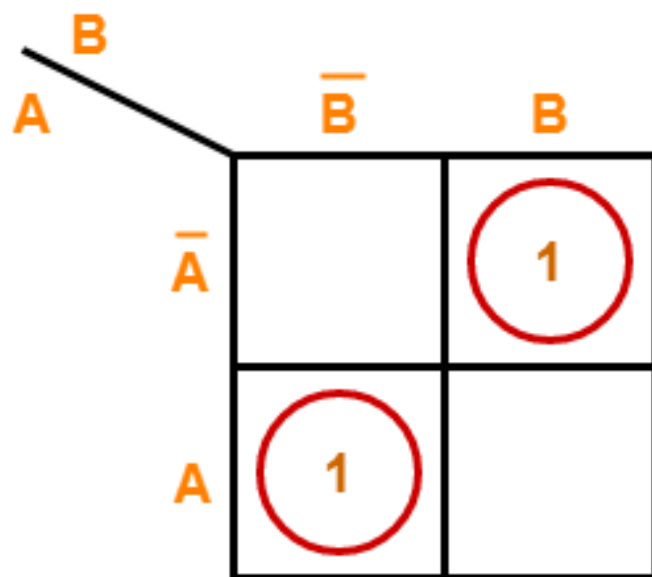
Inputs		Outputs	
A	B	D (Difference)	b (Borrow)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Truth Table

Step-03:

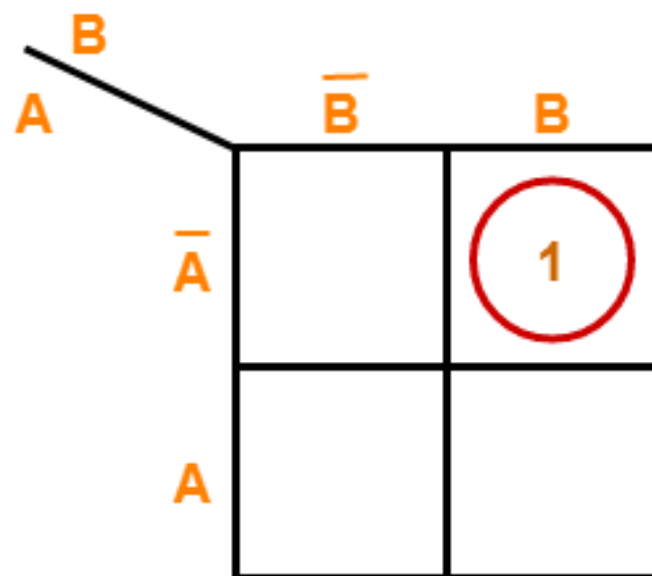
Draw K-maps using the above truth table and determine the simplified Boolean expressions-

For D:



$$D = A \oplus B$$

For b:

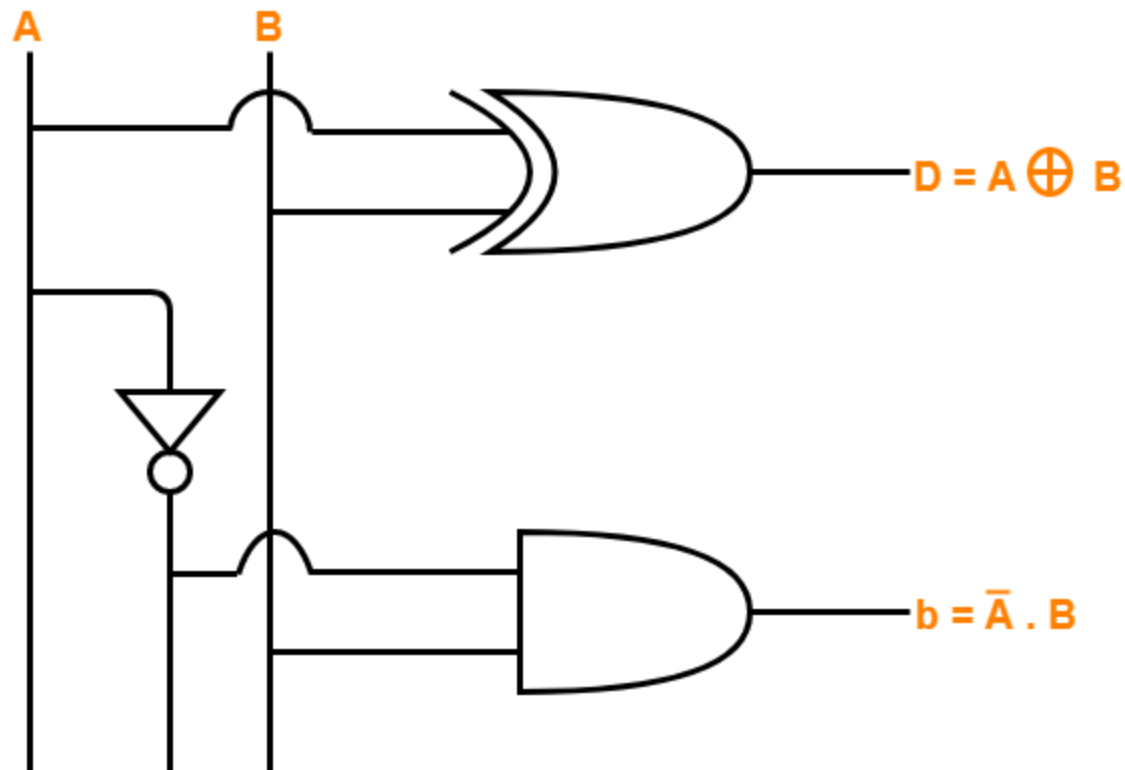


$$b = \bar{A}.B$$

Step-04:

Draw the logic diagram.

The implementation of half subtractor using 1 XOR gate, 1 NOT gate and 1 AND gate is as shown below-



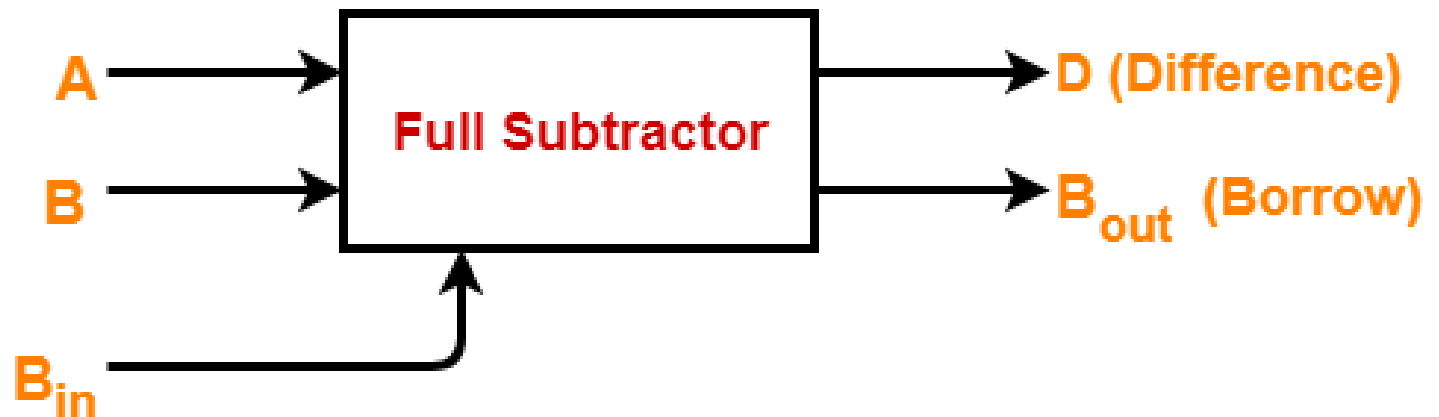
Half Subtractor Logic Diagram

Limitation of Half Subtractor-

- Half subtractors do not take into account "Borrow-in" from the previous circuit.
- This is a major drawback of half subtractors.
- This is because real time scenarios involve subtracting the multiple number of bits which can not be accomplished using half subtractors.

Full Subtractor-

- Full Subtractor is a combinational logic circuit.
- It is used for the purpose of subtracting two single bit numbers.
- It also takes into consideration borrow of the lower significant stage.
- Thus, full subtractor has the ability to perform the subtraction of three bits.
- Full subtractor contains 3 inputs and 2 outputs (Difference and Borrow) as shown-



Designing a Full Subtractor-

Full subtractor is designed in the following steps-

Step-01:

Identify the input and output variables-

- Input variables = A , B , B_{in} (either 0 or 1)
- Output variables = D , B_{out} where D = Difference and B_{out} = Borrow

Step-02:

Draw the truth table-

Inputs			Outputs	
A	B	B _{in}	B _{out} (Borrow)	D (Difference)
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

Truth Table

Step-03:

Draw K-maps using the above truth table and determine the simplified Boolean expressions-

For D:

A	$B B_{in}$				$\bar{B} \bar{B}_{in}$				$\bar{B} B_{in}$				$B \bar{B}_{in}$			
	\bar{A}				A											
			1					1							1	
	1				1											

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

For B_{in} :

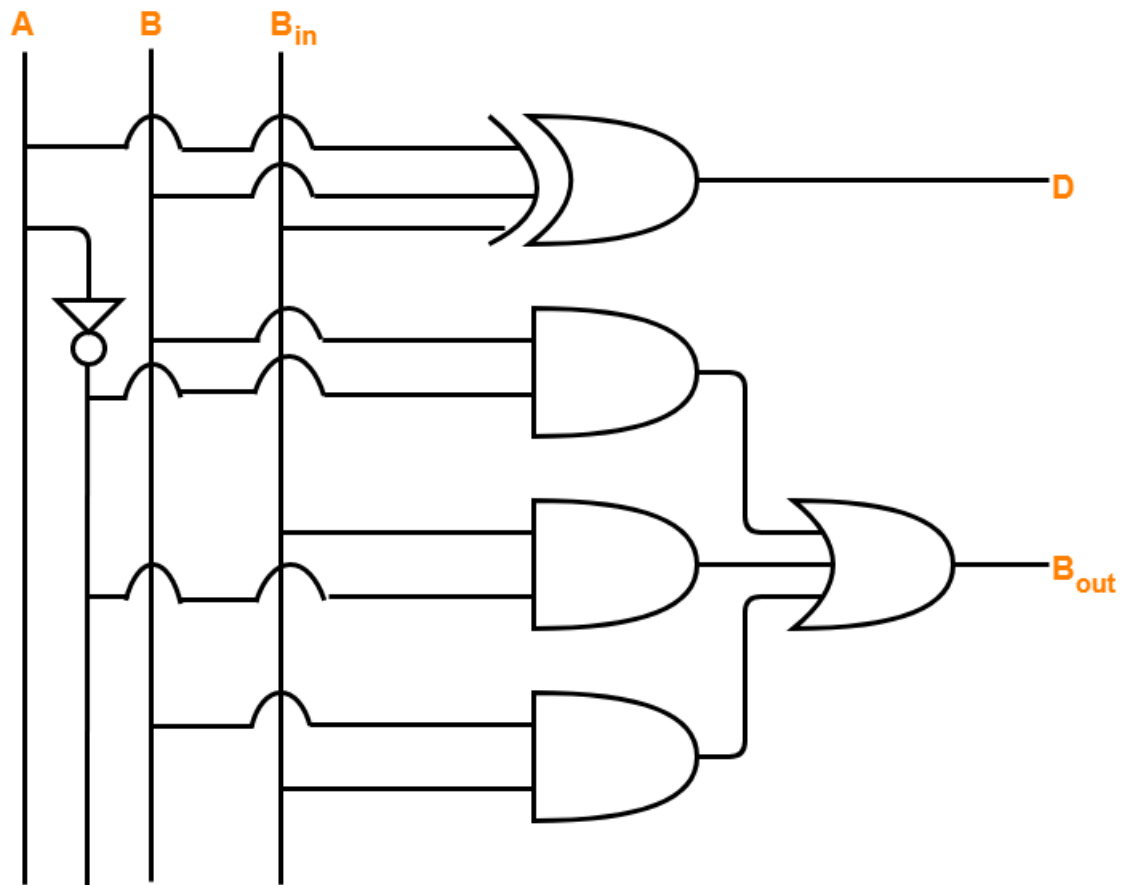
A	$B B_{in}$				$\bar{B} \bar{B}_{in}$				$\bar{B} B_{in}$				$B \bar{B}_{in}$			
	\bar{A}				A											
			1					1							1	
												1				

$$B_{out} = \bar{A} B + (\bar{A} + B) B_{in}$$

Step-04:

Draw the logic diagram.

The implementation of full adder using 1 XOR gate, 3 AND gates, 1 NOT gate and 1 OR gate is as shown below-



Full Subtractor Logic Diagram

Multiplexer & Demultiplexer

- **Introduction**
- There are many applications where many devices or system needs to access a common transmission media.
- Let us consider our cable TV system, in which multiple TV channels are received at home via a shared transmission media.
- Similarly, in computer networking, many computer needs to access a common printer or other shared hardware.
- In Smart instrumentation, a PC or microcontroller can acquire data gathered from many sensors through a single ADC.
- Our telephone exchanges utilize these multiplexers and demultiplexers extensively.

Introduction

Multiplexer and demultiplexers are normally used for sharing resources.

Multiplexers routes the data from many sources to one destination and demultiplexer redistributes data back from one source to many destinations.

In this module you will learn about the basic architecture of multiplexer and demultiplexers along with real life applications.

Principles of Multiplexing and Demultiplexing

- Let us now begin with principles of multiplexing and demultiplexing.
- Let us consider a system with n -inputs. Multiplexing means sharing. It means many to one.
- A multiplexers (MUX) is a device that allows digital information from several inputs to be routed onto a single line for transmission as shown in Fig. 1.

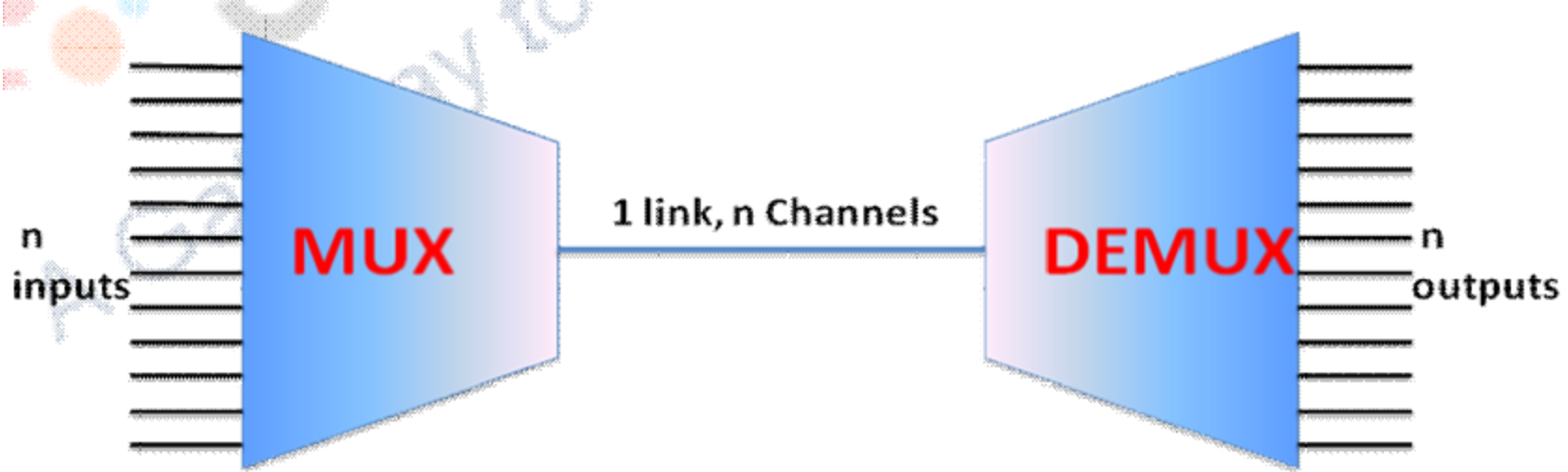


Figure 1: Principle of multiplexing and Demultiplexing

- The multiplexed data is transmitted over a single line to a common destination.
- Demultiplexer accepts this data and redistributes it to the n outputs. Data select lines are used in many into one operation of multiplexer and one to many operation of demultiplexer.
- In short, multiplexers and demultiplexers are combinational circuits designed to provide sharing of resources. The keyword is **sharing**.

Multiplexer

- Multiplexer means many to one. A multiplexer (MUX) is a combinational circuit which is often used when the information from many sources must be transmitted over long distances and it is less expensive to multiplex data onto a single wire for transmission.
- Multiplexer can be considered as multi-position or rotary switch as shown in fig. 2.
- There are n –inputs and one output. The switch position is controlled by the selector lines.
- The select inputs decide which input is connected to the output.

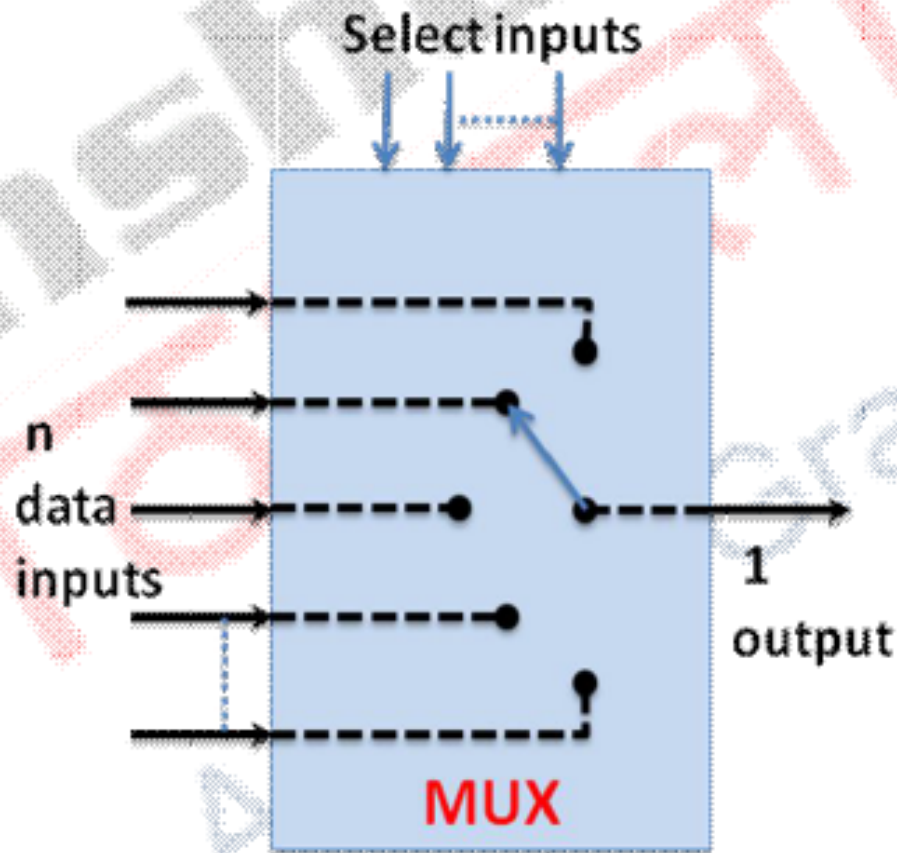


Figure 2: Multiplexer as multi-position or rotary switch

The basic operation is controlled by a selector lines that routes one of many input signals to the output. Fig.3 shows the logic symbol of general symbol of multiplexer.

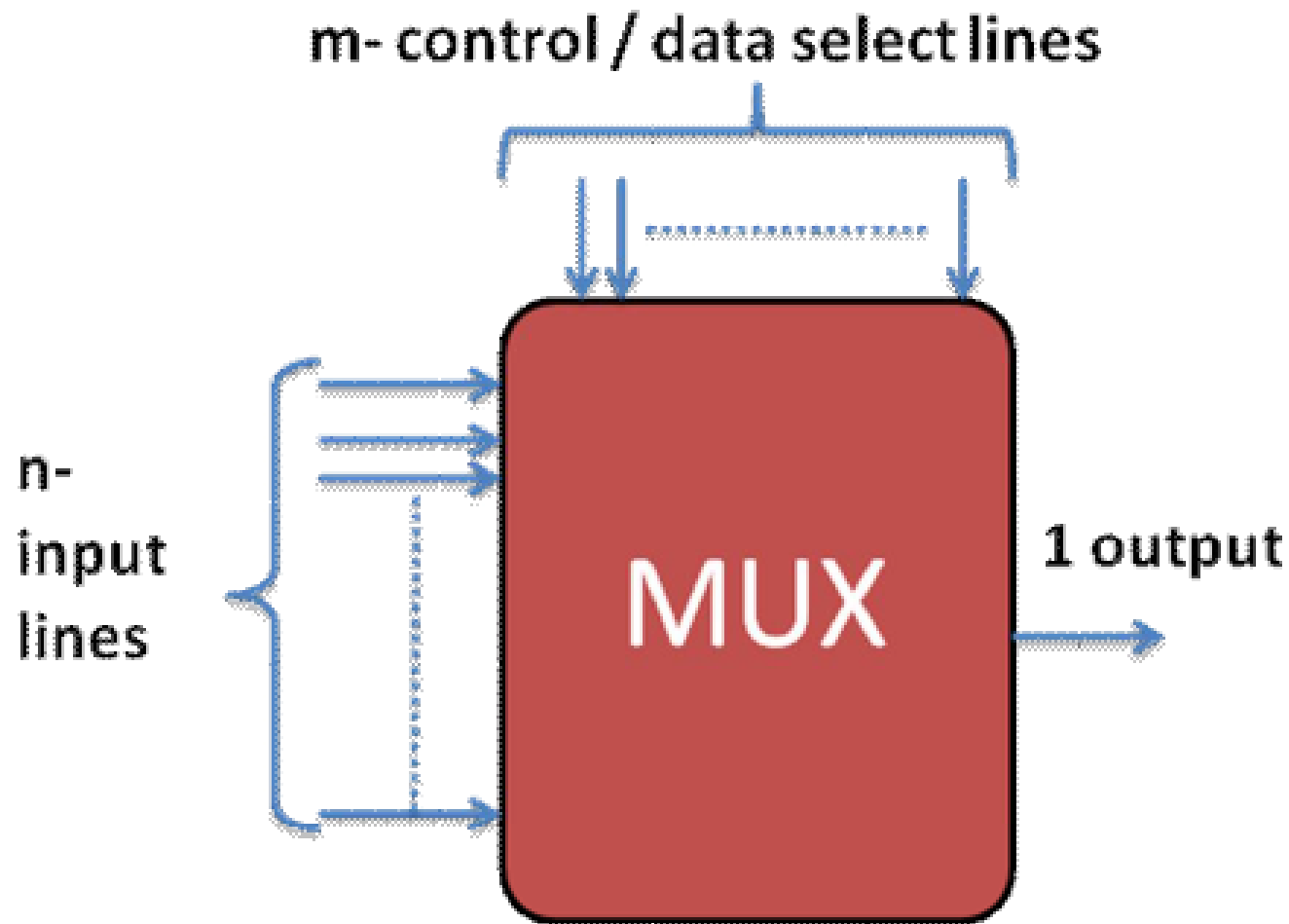
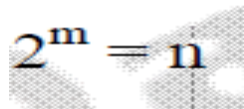


Figure 3: Logic Symbol for multiplexer

Multiplexer

- Multiplexer are also called as **DATA Selector** or router because it accepts several data inputs and allows only one of them to get through to the output at a time.
- The basic multiplexer has n input lines and single output line. It also has m – select or control lines.
- The relation between number of select lines and number of data inputs are

A small, low-resolution image of the equation $2^m = n$. The '2' is blue, the 'm' is red, and the 'n' is blue. The background is a light gray with a subtle grid pattern.

- As multiplexer selects one out of many, it is often called as 2^m to 1 line converter.

3.1 Types of multiplexer

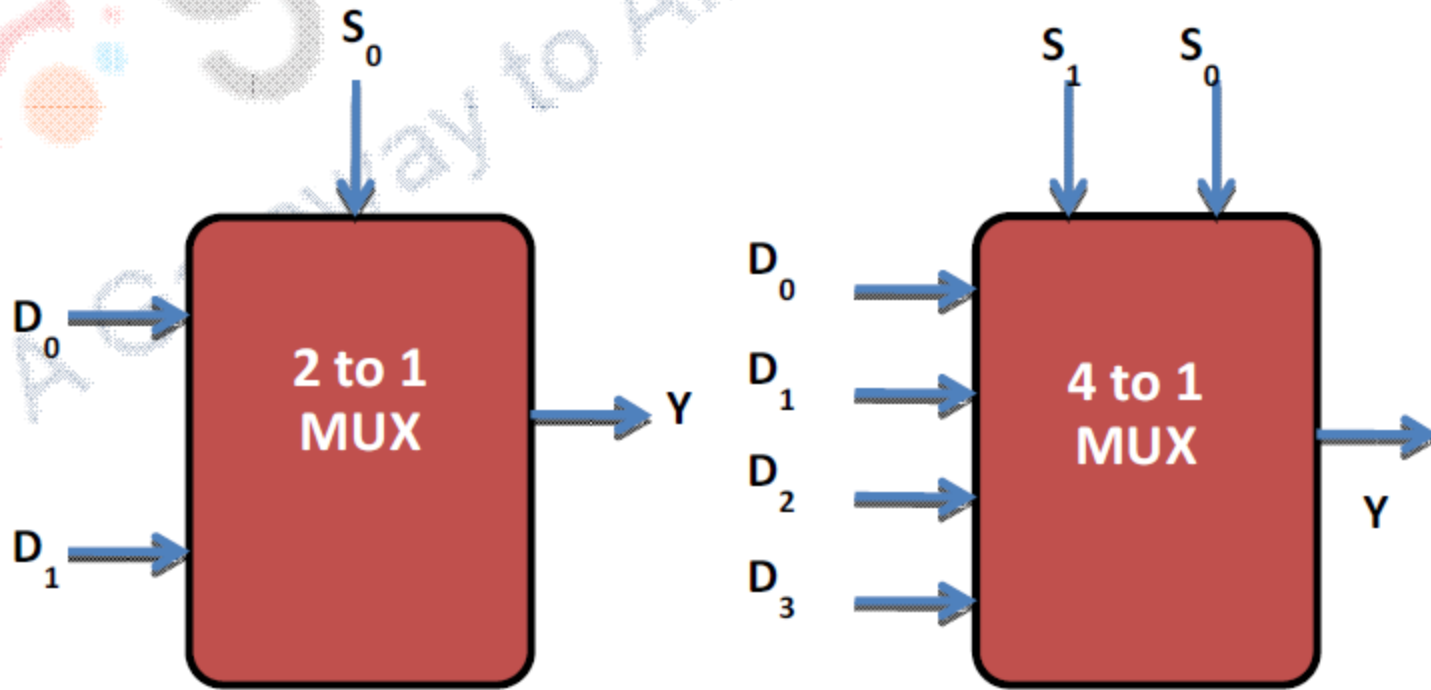
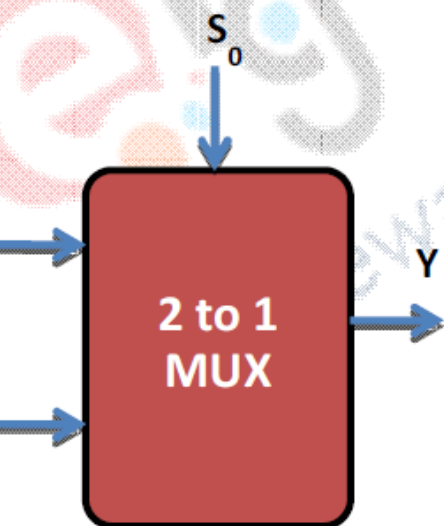


Figure-4(a) : Logic symbols of 2 to 1 and 4 to 1 multiplexers

A 2 to 1 multiplexer



Select input S_0	Output Y
0	D_0
1	D_1

4 2 1

S_0	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Figure 5: Logic symbol, function table and Truth table of 2 to 1 Multiplexer

Figure 5 indicates the logic symbol, function table and truth table of 2 to 1 multiplexer. In this multiplexer, D_0 and D_1 are the data inputs, S_0 is the select line and Y is the output of the

multiplexer. In a function table, Select line S0 is shown as the input and Y is the output. When $S_0 = 0$ then the data D0 appears at output Y and when $S_0=1$, the output Y receives the data D1.

Let us now prepare a detailed truth table for 2 to 1 mux. In this truth table S0, D1, D0 are the inputs and Y is the output. For $S_0=0$, we have 4 possible combinations for the inputs 00,01,10 and 11. It is observed that Y always follows D0. For $S_0=1$, we have similar 4 possible combinations for the inputs 00,01,10 and 11. It is observed that Y always follows D1.

The k-map required for this multiplexer is of three variables. Let us consider two rows for S0 and 4 columns for the data D1D0. Map the truth table into k-map by writing '1' to the appropriate minterm as shown in fig. 6.



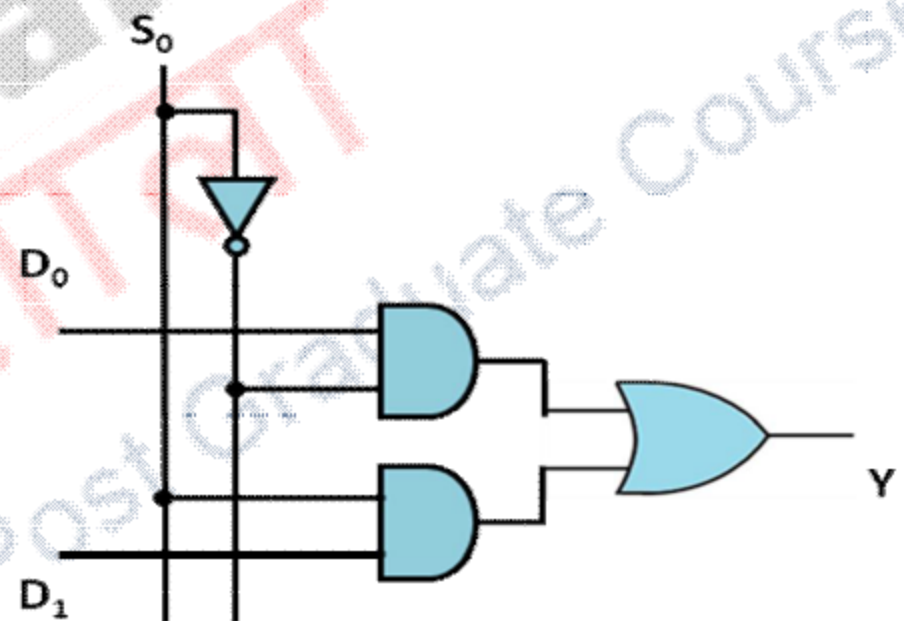
00

01

11

10

	$\overline{D_1} \overline{D_0}$	$\overline{D_1} D_0$	$D_1 D_0$	$D_1 \overline{D_0}$
$\overline{S_0}$		1	1	
S_0			1	1



$$Y = \overline{S_0} D_0 + S_0 D_1$$

From the first pair we can eliminate the D_1 , as it is changing in the pair. Thus this pair contribute $\overline{S_0} D_0$ in the output. Similarly, in the second group D_0 gets eliminated as it is changing in the pair. Therefore, we get $S_0 D_1$ as the second term in the Boolean expression.

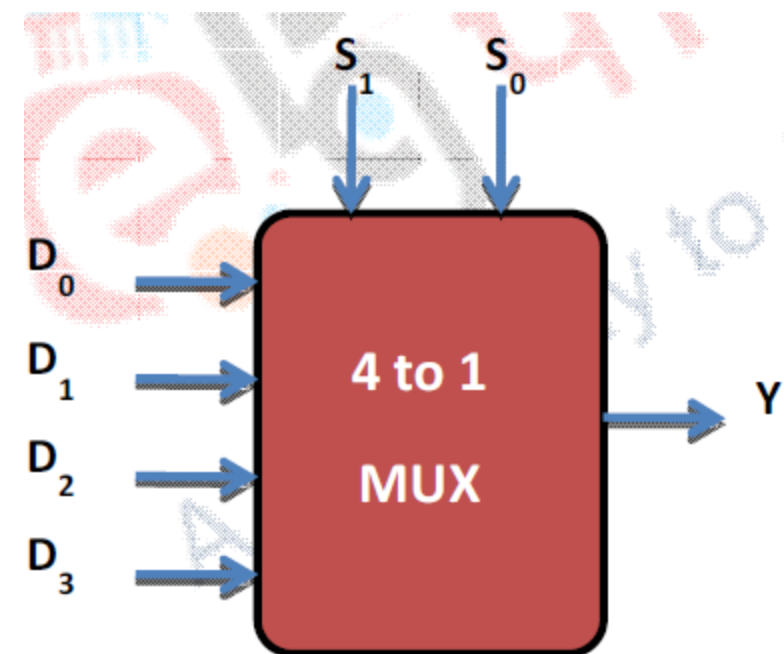
The final Boolean expression is in sum of two product terms as shown here. This indicates either D_0 or D_1 will appear at the output decided by the value of Y .

3.3 A 4 to 1 multiplexer

The second type of multiplexer is 4 to 1 multiplexer. The logic symbol (as shown in fig. 7) indicates that there are 4 data inputs namely D_0 , D_1 , D_2 , D_3 and single output (Y). For 4 data inputs there are two select lines namely S_0 and S_1 . While preparing the function table we write S_1 as MSB and S_0 as LSB. The 2 select inputs provide 4 input combinations for selecting the proper data input at the output Y .

For 4 to 1 multiplexer, two bit binary code on select inputs (S_1S_0) allows the data from selected input (either from D_0, D_1, D_2, D_3) to pass to the output.

The output Y receives D_0 only when $S_1=0$ and $S_0=0$. Similarly, output Y receives D_1 only when $S_1S_0=01$. Output Y receives D_2 only when $S_1S_0=10$. Output Y receives D_3 only when $S_1S_0=11$.



Select inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$Y = S_1' S_0' D_0 + S_1' S_0 D_1 + S_1 S_0' D_2 + S_1 S_0 D_3$$

Figure 8: Logic symbol, Function table and Boolean expression for 4 to multiplexer

From the function table, the Boolean Expression can be written in SOP form. Each row of the function table provides the product term. Referring to the Boolean expression, it is possible to draw the logic circuit consisting of a OR gate with 4 inputs. Each product term is represented by three input AND gate. One of the input of AND gate is the respective data input. The Select lines S1 and S0 along with inverter provide select input either in un-complemented or complemented form. Figure 8 indicates the logic diagram of 4 to 1 multiplexer. The data inputs and select lines are connected to the AND gates as per the requirement of the product term to generate the desired output.

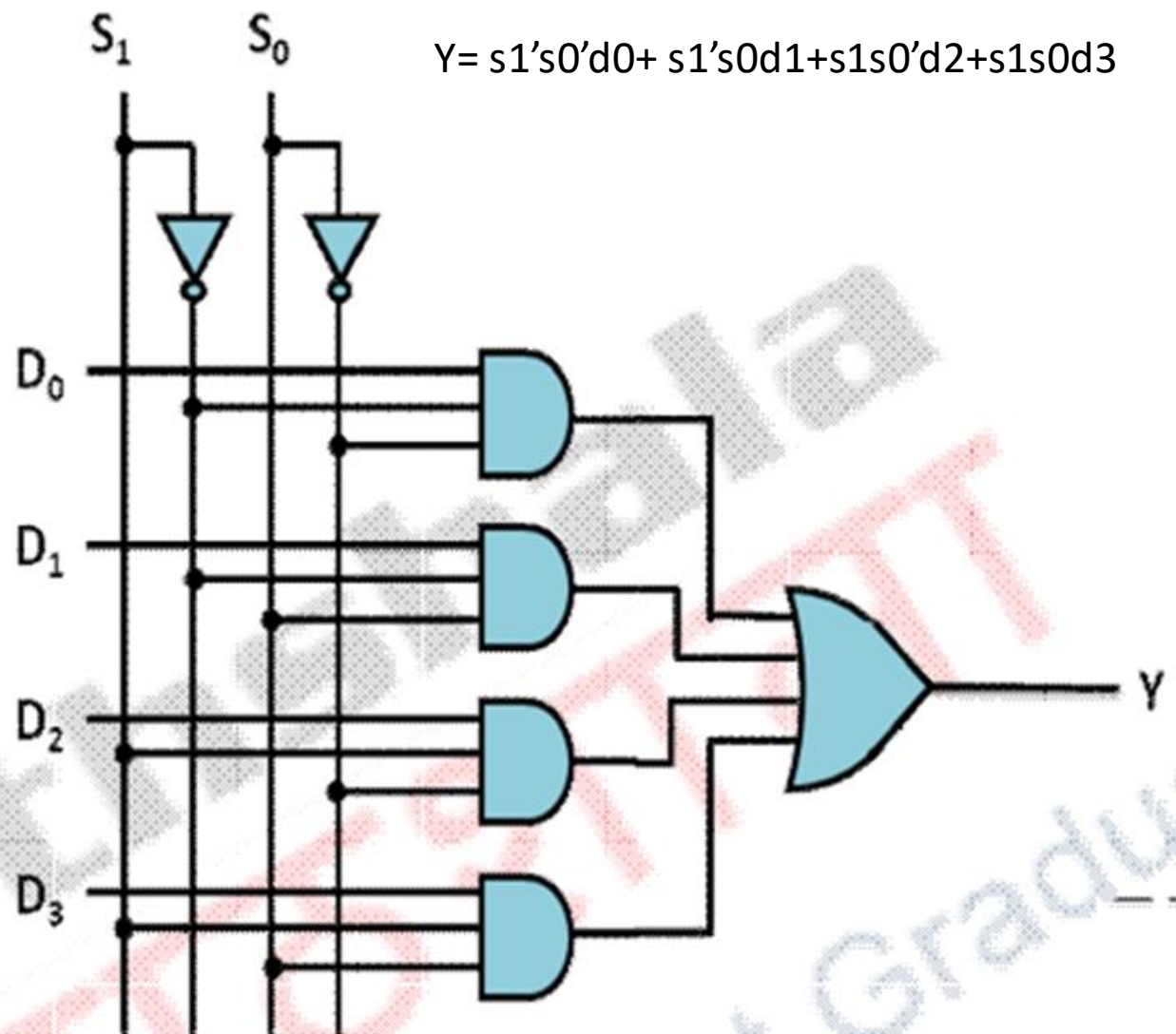
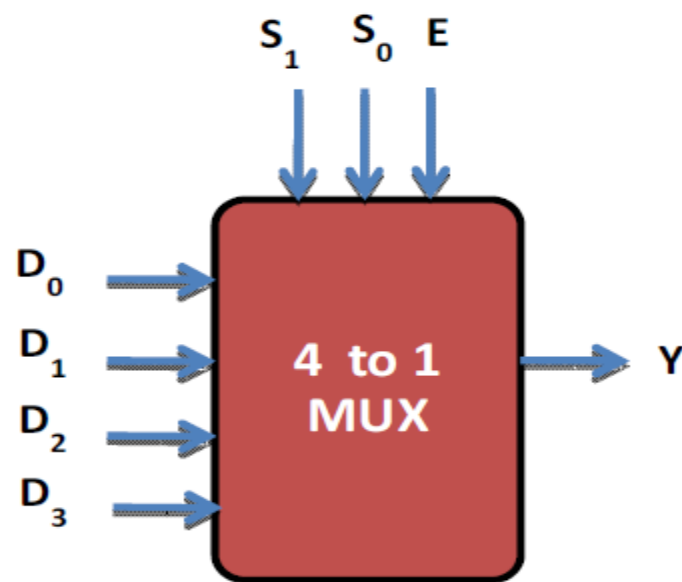


Figure 8: 4 to 1 multiplexer

The first AND gate receives D_0 , S_1' and S_0' as inputs. Similarly rest of the AND gates receives the appropriate inputs as per the product term.

As the data can be selected from any of the input lines, the multiplexer is also known as a **data selector**. In this fashion, it is possible to construct 8 to 1 multiplexer and 16 to 1 multiplexer. Note that the 8 to 1 multiplexer requires three select inputs to select data from 8 inputs to the 1 output whereas the 16 to 1 multiplexer requires 4 select lines.



Select Inputs			Output
S_1	S_0	E	Y
X	X	0	X
0	0	1	D_0
0	1	1	D_1
1	0	1	D_2
1	1	1	D_3

Figure 11: Multiplexer with Enable input

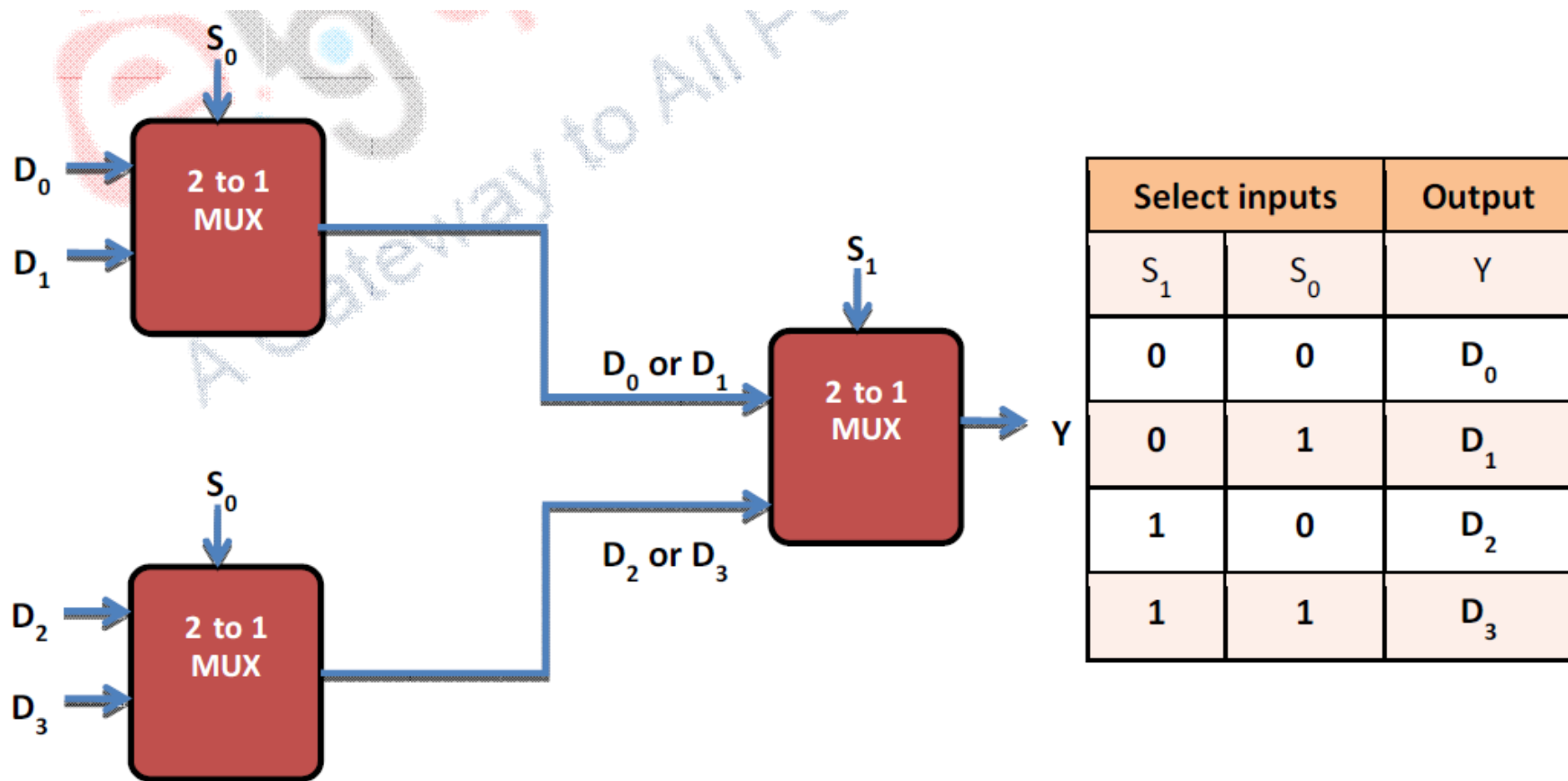


Figure. 12: Construction of 4 to 1 multiplexer using two 2 to 1 multiplexer.

Applications of multiplexers

1. Data routing
2. Data bussing
3. Switch setting comparator
4. Multiplexer as a function generator
5. Parallel to serial converter
6. Cable TV signal distribution
7. Telephone network
8. Sharing printer /resources

Demultiplex means one into many. A demultiplexer reverses the multiplexing operation. In other words, the demultiplexer takes one data input source and selectively distributes it to 1 of N output channels just like multi-position switch.

It also has ‘**m**’ select lines for selecting the desired output for the input data as shown in fig. 21. The mathematical relation between select lines and ‘**n**’ output are:

$$2^m = n$$

M(select i/p)	N(no of outputs)
2	1:4
3	1:8
4	1:16

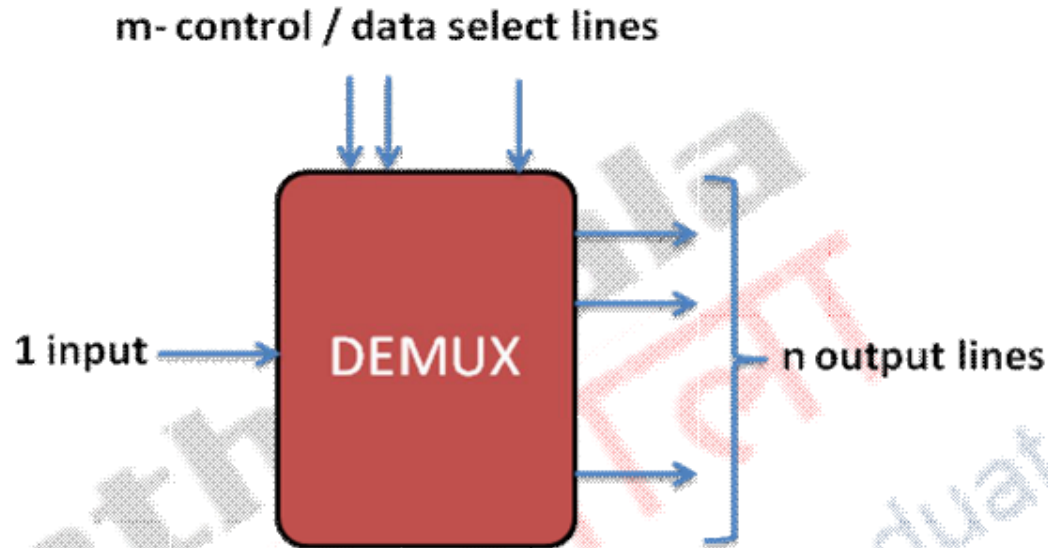
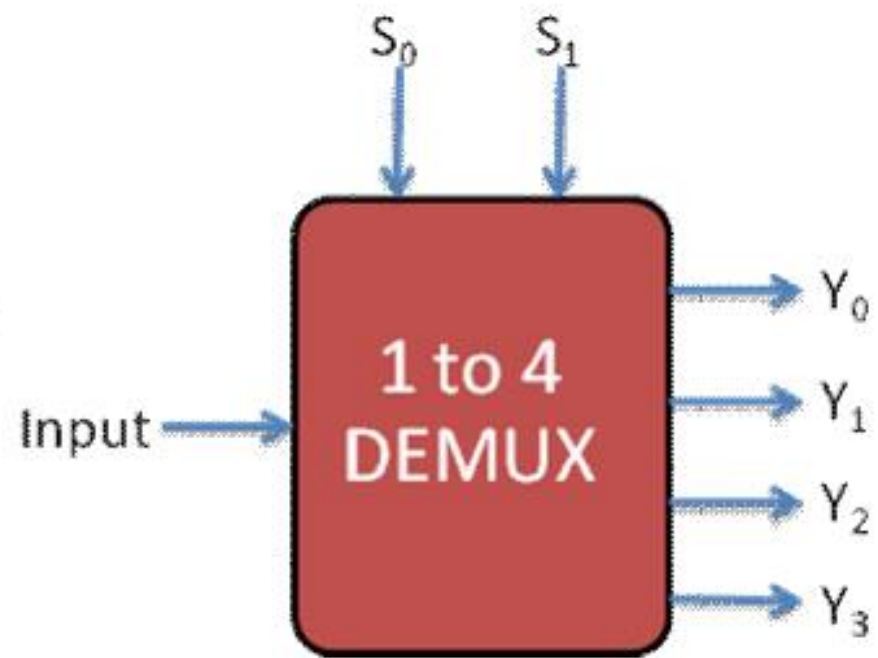
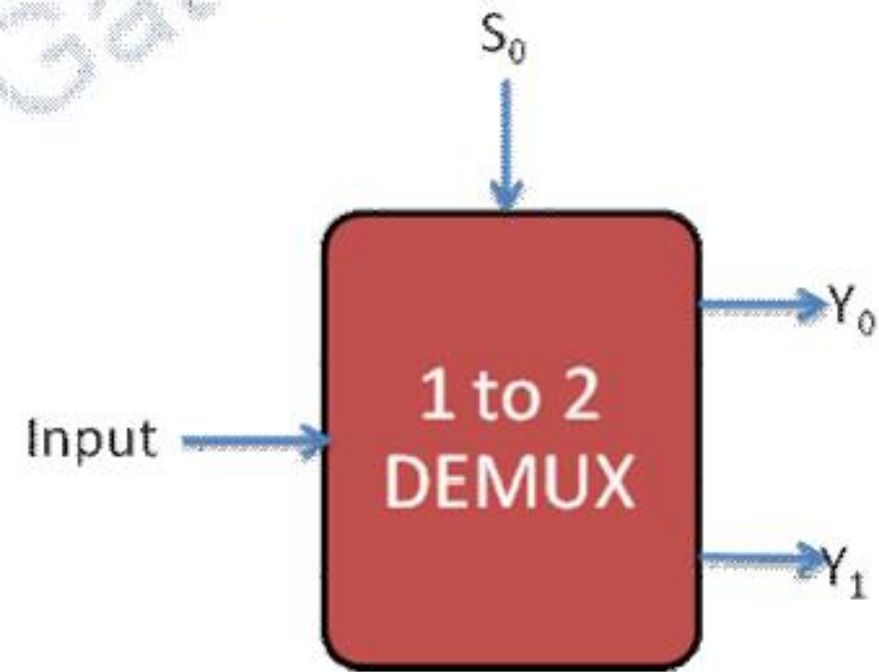


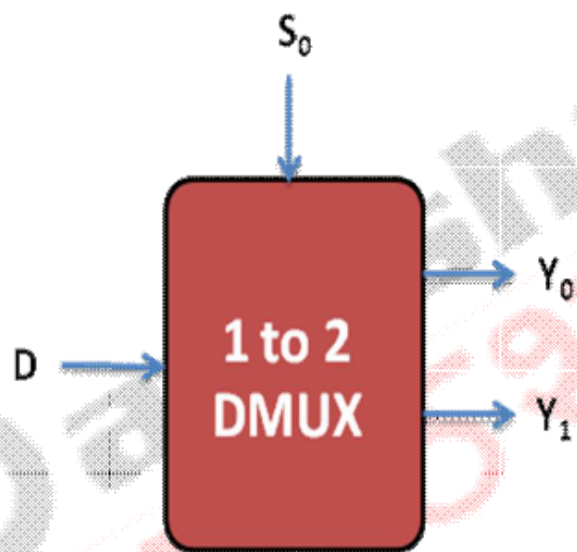
Figure 21: Logic symbol of basic demultiplexer

As a demultiplexer takes data from one input line and distributes over a 2^m output line, hence it is often referred to as **1 to 2^m line converter**. There are four basic types demultiplexers: 1 to 2 demultiplexer, 1 to 4 demultiplexer, 1 to 8 demultiplexer and 1 to 16 demultiplexer as shown in fig. 22. Number of select lines decides this classification.

Gate



4.1 A 1to 2 demultiplexer



Select input	Outputs	
S ₀	Y ₀	Y ₁
0	D	0
1	0	D

Figure 23: Logic symbol and function table of a 1 to 2 demultiplexer

As shown in fig.22, in 1 to 2 demultiplexer, with $S_0=0$ the Y_0 output of demultiplexer receive the input data. Similarly when S_0 becomes '1', the Y_1 output of demultiplexer receives the input data. Thus the Select or control line selects the desired output to which the input data is transferred or distributed. Hence, demultiplexer is also known as **data distributor**.

To distribute the input data D to Y0, the select input S₀ should be 0 and Y1 will receive data input D when S₀=1. The Boolean expressions for the outputs are

$$Y_0 = \overline{S_0} D$$

$$Y_1 = S_0 D$$

The implementation of 1 to 2 demultiplexer requires two 2 input AND gate and a NOT gate as shown in fig. 24. The product term for the output decides the interconnections between the gates: data input and select lines.

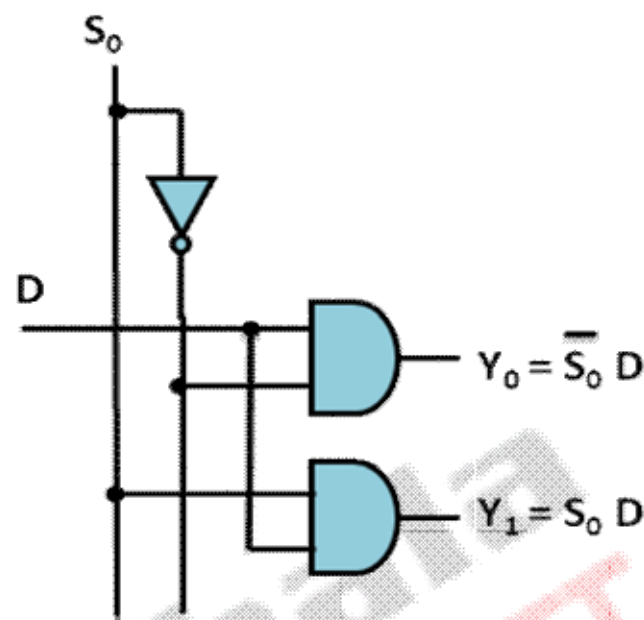
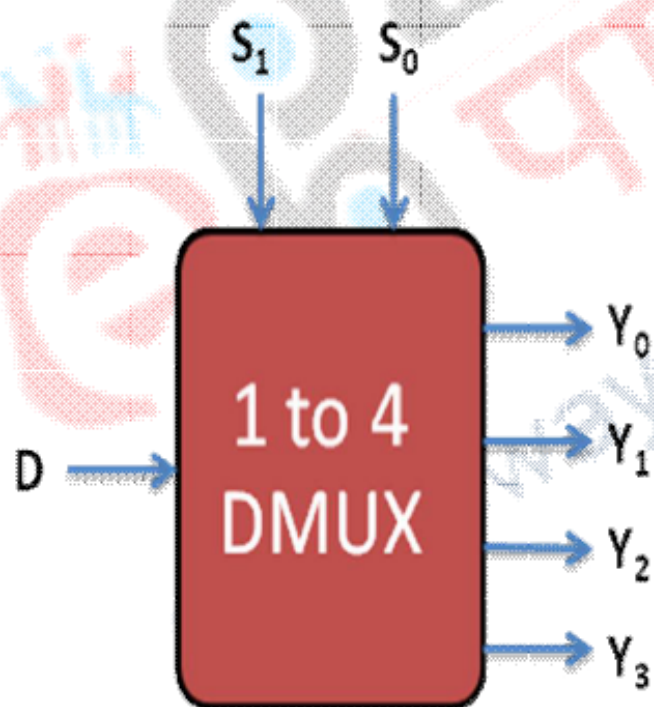


Figure 24: Logic diagram for 1 to 2 demultiplexer

4.2 A 1 to 4 demultiplexer



Inputs		Outputs			
S ₁	S ₀	Y ₀	Y ₁	Y ₂	Y ₃
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

Figure 25: Logic symbol and function table of a 1 to 4 demultiplexer

Fig. 25 indicates the logic symbol and function table of 1 to 4 demultiplexer. In 1 to 4 demultiplexer, the input data can be distributed to 1 of the 4 outputs. Selection of the output is decided by the binary word applied to the select lines. With $S_1S_0=00$, the Y_0 output of demultiplexer receive the input data. For $S_1S_0=01$, the output Y_1 receives the input data. With

$S_1S_0=10$, the input data is distributed to Y_2 and when $S_1S_0=11$, the output Y_3 receives the input data.

The Boolean expressions for the outputs are:

$$Y_0 = \overline{S_1} \overline{S_0} D$$

$$Y_1 = \overline{S_1} S_0 D$$

$$Y_2 = S_1 \overline{S_0} D$$

$$Y_3 = S_1 S_0 D$$

The implementation of 1 to 4 demultiplexer requires four 3 input AND gates and two NOT gates as shown in fig. 26. The product term for the output decides the interconnections between the gates data input and select lines.

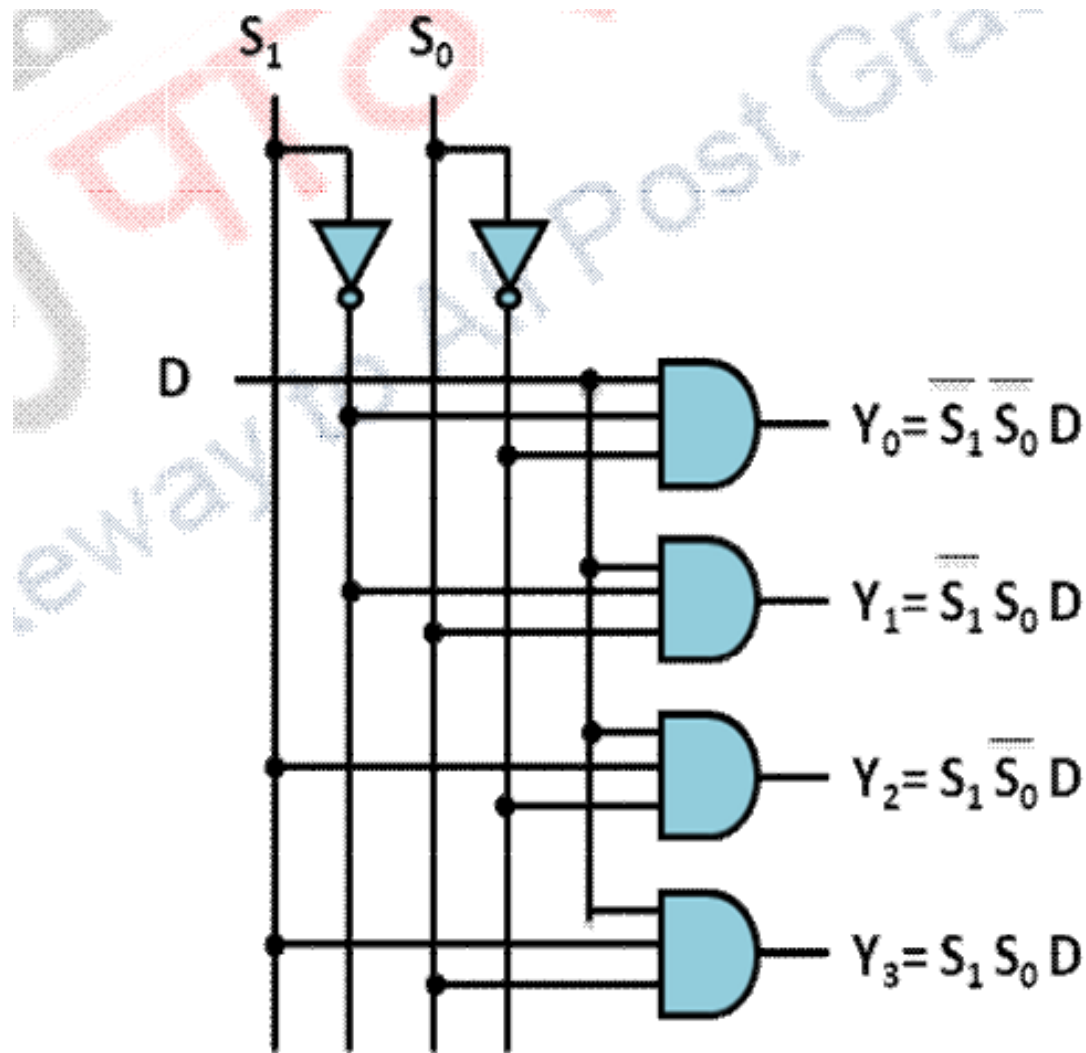


Figure 25: Logic diagram for 1 to 4 demultiplexer

Encoder

1. Introduction

Numbers are usually coded in one form or another so as to represent or use it as required. In modern communication system, data communication involves a process encoding and decoding. Encoding is the process of converting human readable characters into binary information. Decoding is the reverse process of conversion of binary or encoded format into human readable code. This process is also referred to as code conversion.

3. Basics of Encoder

The process of converting from human readable code to machine readable code i.e. binary is known as **encoding**. An encoder is a combinational circuit that converts more familiar numbers, symbols or character into binary code. An encoder has a number of input lines but only one of them is activated at a time representing a digit or character and produces a binary code depending on which input is activated. Figure 2 is the logic symbol of encoder with 'm' inputs and 'n' outputs. In short, it is multiple inputs and multiple outputs device with proper conversion system. Note that encoder performs the reverse operation of the decoder.

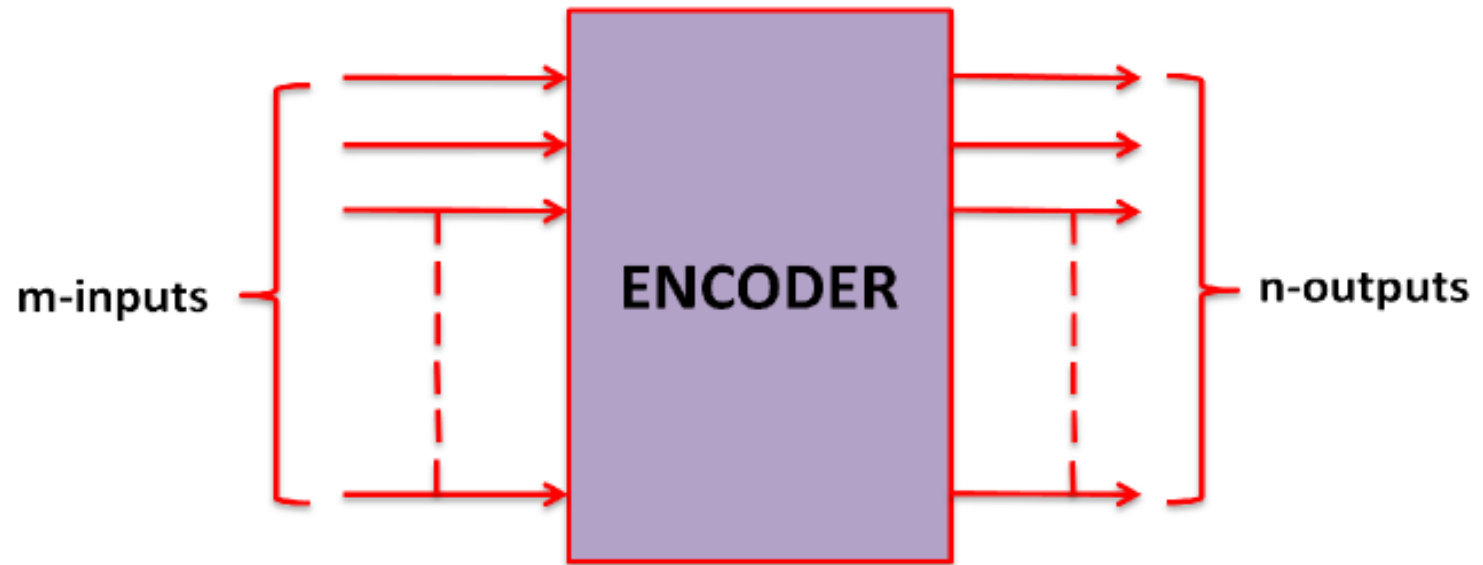


Fig. 2: Logic Symbol of Encoder

Decimal to BCD encoder

- A decimal to BCD (binary coded decimal) encoder is also known as 10-line to 4-line encoder. It accepts 10- inputs and produces a 4-bit output corresponding to the activated decimal input.
- Figure-5 shows the logic symbol of decimal to BCD encoder. **Fig. 5: Decimal to BCD Encoder**

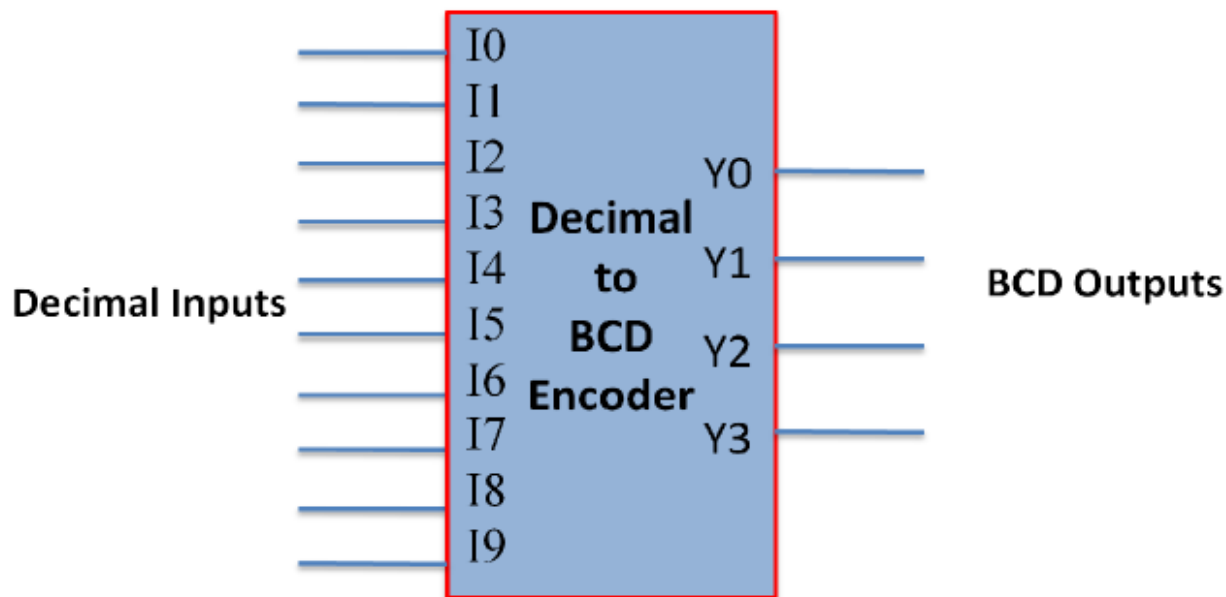


Fig. 5: Decimal to BCD Encoder

The truth table of Decimal to BCD encoder is shown in Table-2. There are ten inputs corresponding to 10 decimal inputs (I0, I1, I2, I3, I4, I5, I6, I7, I8, I9) and 4 Outputs (Y0, Y1, Y2, Y3). Let us look at the table carefully. Note the encoder assumption that only one of the inputs is activated (logic 1) and other inputs are not activated (i.e. at logic 0).

Decimal Inputs										BCD outputs			
										8	4	2	1
I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	B3	B2	B1	B0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

$$B0 = I1 + I3 + I5 + I7 + I9$$

$$B1 = I2 + I3 + I6 + I7$$

$$B2 = I4 + I5 + I6 + I7$$

$$B3 = I8 + I9$$

						hEXADecimal Inputs										Binary outputs			
																8	4	2	1
				B	A	I9	I8	I7	I6	I5	I4	I3	I2	I1	I0	B3	B2	B1	B0
						0	0	0	0	0	0	0	0	0	1	0	0	0	0
						0	0	0	0	0	0	0	0	1	0	0	0	0	1
						0	0	0	0	0	0	0	1	0	0	0	0	1	0
						0	0	0	0	0	0	1	0	0	0	0	0	1	1
						0	0	0	0	0	1	0	0	0	0	0	1	0	0
						0	0	0	0	1	0	0	0	0	0	0	1	0	1
						0	0	0	1	0	0	0	0	0	0	0	1	1	0
						0	0	1	0	0	0	0	0	0	0	0	1	1	1
						0	1	0	0	0	0	0	0	0	0	1	0	0	0
						1	0	0	0	0	0	0	0	0	0	1	0	0	1
					1											1	0	1	0
				BOF I1+I3+I5+I7+I9												1	0	1	1
				B1= i2+i3+i6+i7															
				B2=4+5+6+7															
				B3=8+9															

- In the truth table, the input variable I0 represents the least significant digit (LSD) and I9 represents most significant digit (MSD).
- Similarly, in the outputs Y0 represents least significant bit (LSB) and Y3 is the most significant bit (MSB). The truth table includes only all valid combinations of the inputs.
- The valid combinations are those which have exactly one input equal to logic 1 while all other inputs are logic 0's.
- As the number of inputs is 10, K-maps cannot be used to derive the Boolean expressions. The Boolean expression can be directly derived from the truth table by visual inspection. Let us obtain the Boolean expression for each output.
- Output Y0 is 1 if any of the inputs I1 or I3 or I5 or I7 is 1. Then, the Boolean expression

$$Y0 = I1 + I3 + I5 + I7 + I9$$

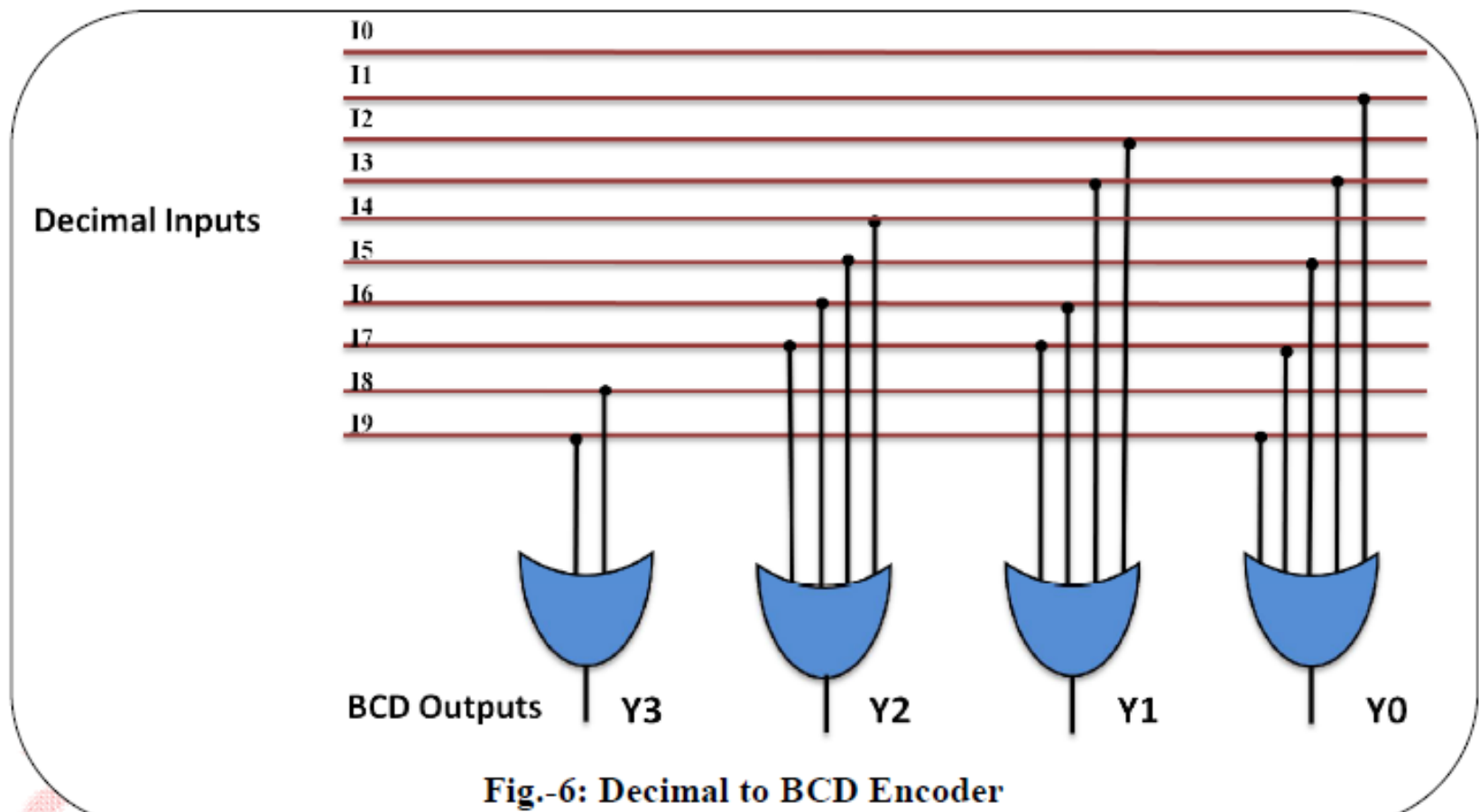
Similarly,

$$Y1 = I2 + I3 + I6 + I7$$

$$Y_2 = I_4 + I_5 + I_6 + I_7$$

$$Y_3 = I_8 + I_9$$

From these Boolean expressions, the Decimal to BCD Encoder can be implemented by using simply three 4 OR gates. Figure-6 indicates the logic diagram for Decimal to BCD Encoder.

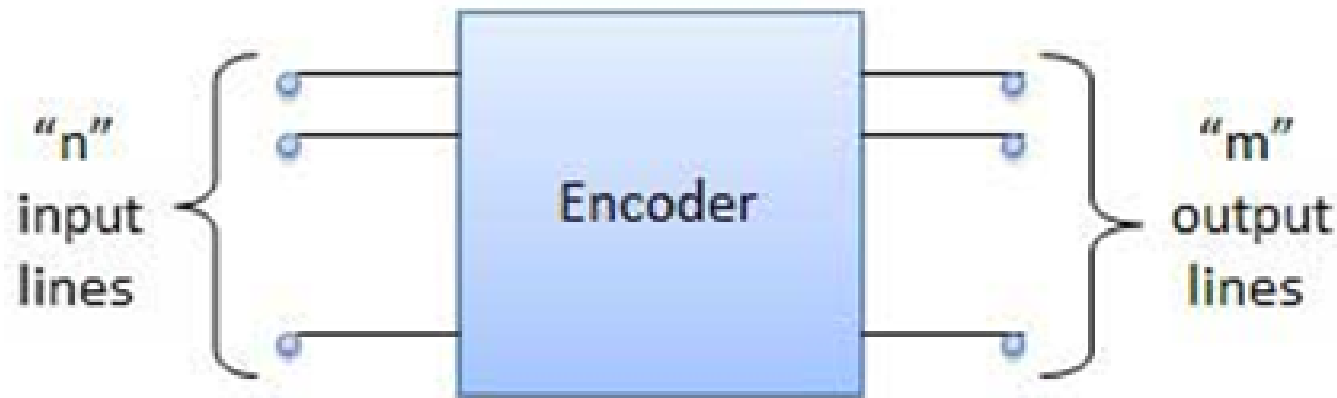


Hexadecimal to Binary

Encoder

- Encoder is a combinational circuit which is designed to perform the inverse operation of the decoder.
- An encoder has n number of input lines and m number of output lines. An encoder produces an m bit binary code corresponding to the digital input number.
- The encoder accepts an n input digital word and converts it into an m bit another digital word.

Block diagram



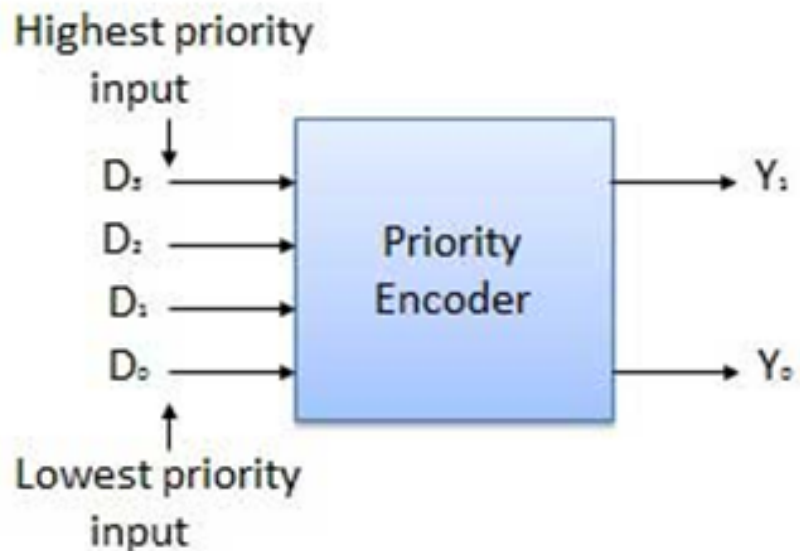
Examples of Encoders are following.

- Priority encoders
- Decimal to BCD encoder
- Octal to binary encoder
- Hexadecimal to binary encoder

4:2 Line Priority Encoder

This is a special type of encoder. Priority is given to the input lines. If two or more input lines are 1 at the same time, then the input line with highest priority will be considered. There are four input D_0, D_1, D_2, D_3 and two output Y_0, Y_1 . Out of the four input D_3 has the highest priority and D_0 has the lowest priority. That means if $D_3 = 1$ then $Y_1 Y_0 = 11$ irrespective of the other inputs. Similarly if $D_3 = 0$ and $D_2 = 1$ then $Y_1 Y_0 = 10$ irrespective of the other inputs.

Block diagram

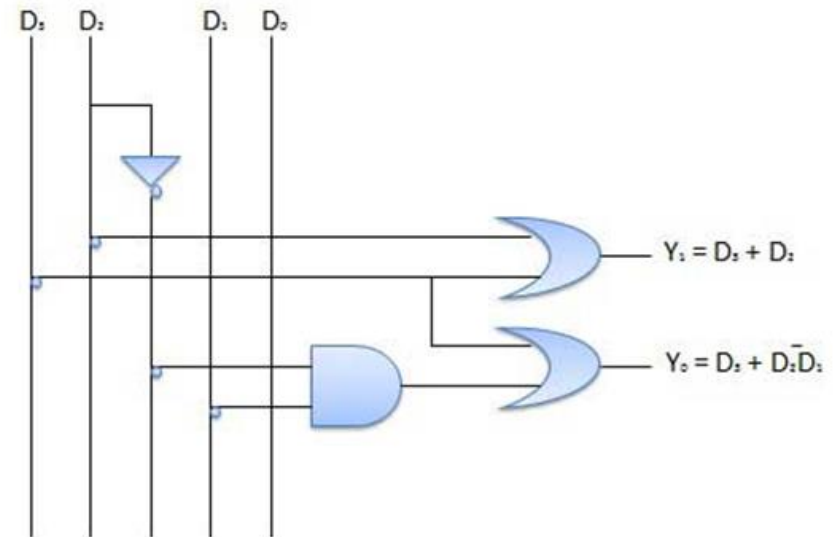


Priority Encoder

Truth Table

Highest	Inputs		Lowest	Outputs	
D_3	D_2	D_1	D_0	Y_2	Y_1
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

Logic Circuit



Highest	Inputs		Lowest	Outputs	
D ₃	D ₂	D ₁	D ₀	Y ₀	Y ₁
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

8421

$$Y_0 = D_2 + D_3$$

$$Y_1 = D_3 + D_2'D_1$$

Y ₀	d ₁ d ₀	00	01	11	10
d ₃ d ₂					
00		X	0	0	0
01		1	1	1	1
11		1	1	1	1
10		1	1	1	1

Y ₁	d ₁ d ₀	00	01	11	10
d ₃ d ₂					
00		X	0	1	1
01		0	0	0	0
11		1	1	1	1
10		1	1	1	1

Decoder

- Define decoder : A combinational circuit which will take binary input and convert it into human readable output.
- Draw truth table for 3:8 line decoder
- Get output equations
- Draw logic diagram
- Applications of decoder

3: 8 decoder

4	2	1								
i2	i1	i0	y7	y6	y5	y4	y3	y2	y1	y0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

$Y0 = i0' i1' i2'$

$Y1 = i2' i1' i0$

Etc