Module III

Combinational Circuit

This module discusses several topics related to combinational circuits, including their analysis and design procedure. It also explores specific types of combinational circuits such as binary adder-subtractors, decimal adders, binary multipliers, magnitude comparators, decoders, encoders, multiplexers, and de-multiplexers.

3.1 Combinational Circuit

A combinational circuit is comprised of a network of logic gates that are interconnected. Combinational circuits respond to the input values they receive and generate an output signal that represents the desired output data, thereby converting binary information from the provided input data.

The above diagram illustrates a block diagram representation of a combinational circuit

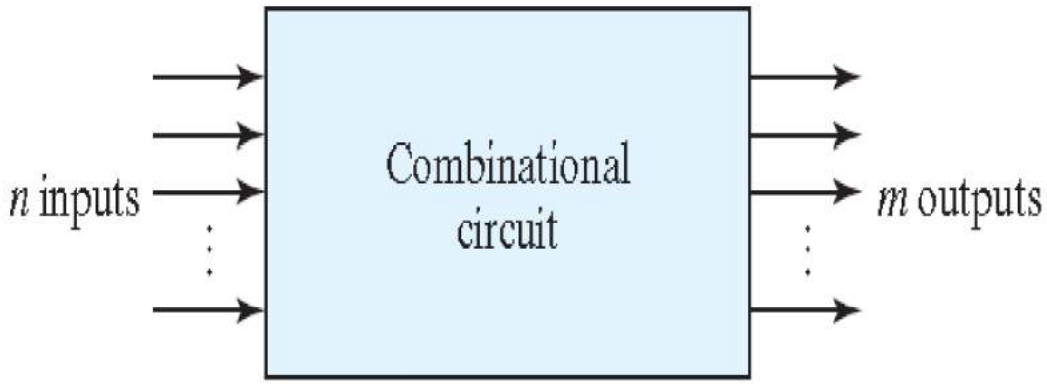


Figure. The schematic representation of a combinational circuit is commonly referred to as a block diagram.The inputs, denoted as n, are sourced externally, while the outputs, denoted as m, are generated by the combinational circuit and directed towards an external destination. The input and output variables in this context pertain to analog electrical signals, which are then translated as binary signals representing logic 1 and logic 0. For a given number of input variables, the total number of possible combinations is equal to twice the value of the number of input variables.

For every conceivable combination of inputs, there exists a unique value for each corresponding output.Therefore, a combinational circuit can be

precisely defined by a truth table that enumerates the output values corresponding to every possible combination of input variables.

A combinational circuit can be characterized by m Boolean functions, with each function corresponding to an output variable.The expression of each output function is dependent on the n input variables.

A variety of widely employed combinational circuits, including adders, subtractors, multipliers, comparators, decoders, encoders, and multiplexers, are readily accessible in conventional integrated circuit components and are utilized as standard cells within intricate very large scale integrated (VLSI) circuits.

3.2 Methodology for Analysis

An established circuit's operational features and behavior are analyzed.The process begins with a logic diagram and ends with Boolean functions, a truth table, or a circuit explanation. The analysis can be done manually by identifying Boolean functions or building a truth table, or using a computer simulation tool.The analysis begins by checking if the circuit is combinational or sequential. Combinational circuits lack feedback channels and memory.Feedback paths link the output of one gate to the input of a second gate, which adds to the input of the first gate. Sequential digital circuits have feedback loops. To get output boolean functions, perform these steps Steps to get Boolean functions from a logic diagram:

Give gate outputs dependent on input variables meaningful labels using arbitrary symbols. Find each gate's output Boolean functions.

Assign input variable-dependent gates and those with symbols different symbols. Find the gates' Boolean functions.

Iterate step 2 until the circuit achieves its goals.

Derive Boolean functions related to input variables by iteratively substituting previously specified functions..

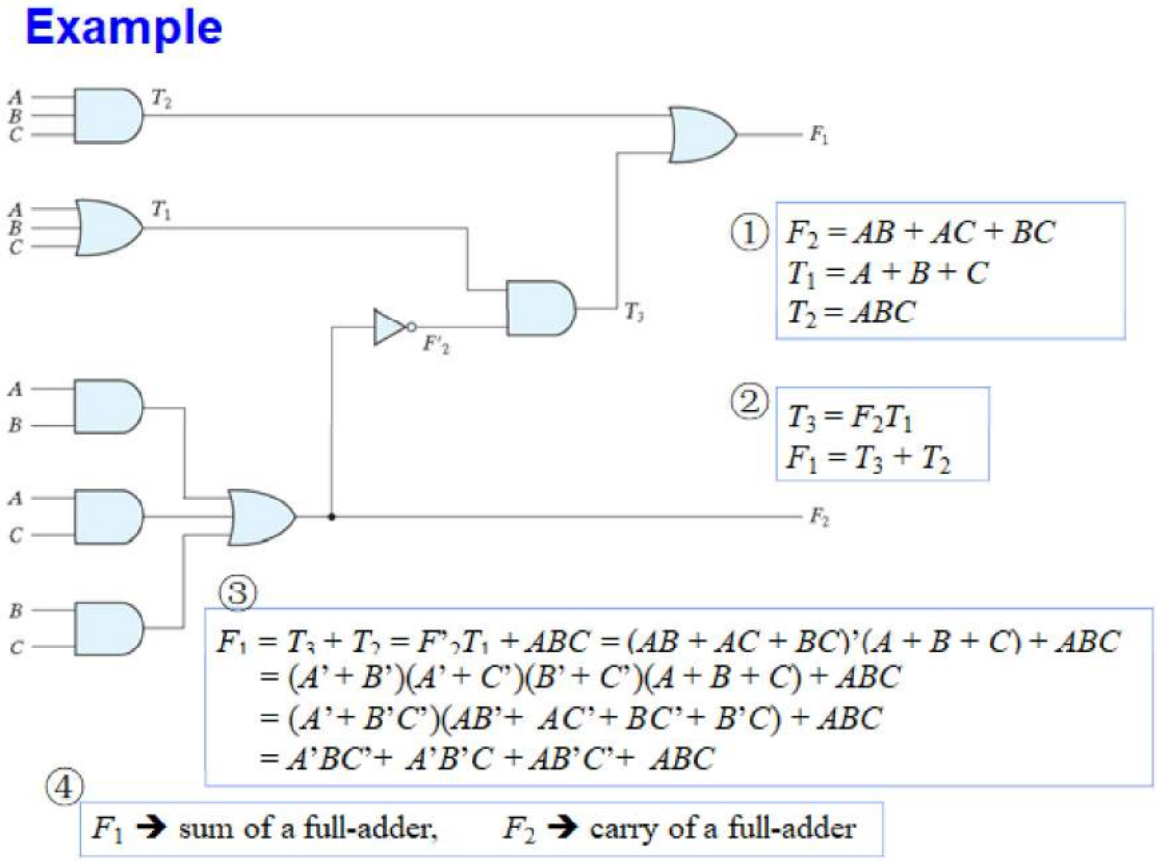


Fig. The analysis procedure refers to the systematic approach employed to examine and interpret data in a research study. It involves a series of steps

Steps to Obtain Truth Table

In order to get the resultant Boolean functions from a given logic diagram, the following steps are undertaken:

Assign meaningful labels to all gate outputs that are dependent on input variables, using arbitrary symbols. Identify the Boolean functions corresponding to the output of each gate.

Assign distinct symbols to the gates that depend on input variables and those that have already been assigned symbols. Determine the Boolean functions associated with the given gates.

Continue to iterate the procedure defined in step 2 until the desired results of the circuit are achieved.

By iteratively replacing previously defined functions, derive the resulting Boolean functions expressed in relation to the input variables.

.

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

To ascertain the quantity of input variables. To generate a comprehensive list of input combinations for a given number of inputs, n, it is necessary to produce 2n distinct combinations. These combinations can be represented as binary values ranging from 0 to (2n - 1). These binary numbers can then be organized and presented in a tabular format.

3.3 Design Procedure

The purpose of the design is to obtain a logic circuit or a collection of Boolean functions based on the specified design objective.

The design procedure encompasses a series of sequential steps that are undertaken in order to achieve a desired outcome.

Based on the given circuit parameters, it is necessary to ascertain the requisite quantity of inputs and outputs, and thereafter designate a distinct symbol to each of them.

The truth table that delineates the necessary correlation between input values and output values will now be derived.

Derive the simplified Boolean functions for each output by expressing them as a function of the input variables.

Produce a logic diagram and assess the accuracy of the design by manual examination or simulation

3.4 Binary Adder

3.4.1The Design and Functionality of Half and Full Adder Circuits

Digital computers and calculators are capable of executing a wide range of arithmetic operations on numerical values that are encoded in binary format. All of these actions are executed within the arithmetic logic unit (ALU) of a computer. The integration of logic gates and flip-flops within the arithmetic logic unit facilitates the execution of fundamental arithmetic operations like as addition, subtraction, multiplication, and division. These circuits exhibit computational capabilities that beyond the limits of human performance in terms of speed. In this section, we will examine the addition operation circuit, a significant mathematical operation within digital systems.

Binary addition is a fundamental operation in computer science and mathematics that involves adding two binary numbers together. It is a process that follows certain

The concept of the Half Adder

Binary Addition

The concept of the Half Adder

When the sum of two digits is computed, it yields two distinct components, namely the sum itself and the carry. Given that there are only four potential combinations of two binary digits, it is evident that one of these combinations will result in a carry. This can be demonstrated as.

		0				1				0				1	
		+0				+0				+1				+1	
Carry		Sum		Carry		Sum		Carry		Sum		Carry		Sum	
0		0		0		1		0		1		1		0	

The truth table illustrating the addition of two binary variables A and B is presented below

The Half Adder

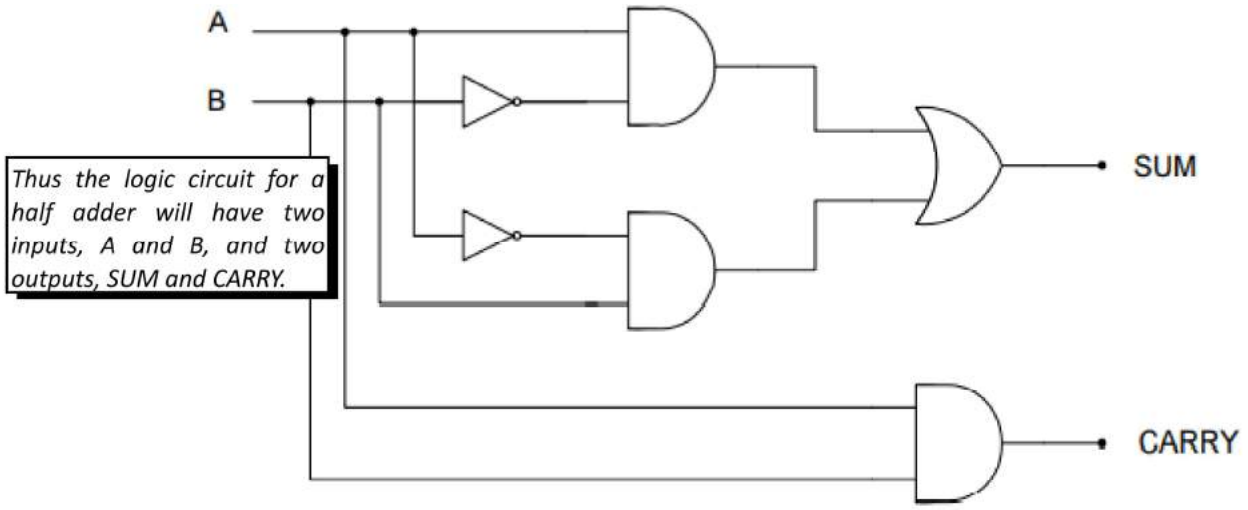
A	B	Sum	Carry	
0	0	0	0	
0	1	1	0	
1	0	1	0	
1	1	0	1	

Table. Half adder truth table.

Based on the analysis of the truth table, it is evident that the output labeled as "SUM" is only generated when the input variable A is set to 1 and the input variable B is set to 0, or when A is set to 0 and B is set to 1. A carry is generated exclusively when both inputs A and B are in a logical high state. Therefore, the Boolean expression representing the SUM and CARRY outputs derived from the aforementioned truth table can be expressed as follows :

$$SUM = A \bullet \bar{B} + \bar{A} \bullet B$$
$$CARRY = A \bullet B$$

The concept of a half adder is a fundamental building block in digital logic circuits. It is used to perform basic addition operations on binary numbers. The half adder consists.



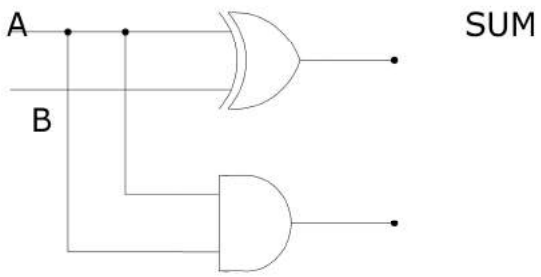
Thus the logic circuit for a half adder will have two inputs, A and B, and two outputs, SUM and CARRY.

Fig. SUM and CARRY

The sum can be easily generated by an Exclusive OR gate since

$$A \oplus B = A \bullet \bar{B} + \bar{A} \bullet B$$

Consequently the resulting circuit is :



Half adder circuit.

Fig sum and carry

The term "carry" refers to the act of transporting or conveying an object or load from

Figure. The concepts of "SUM" and "CARRY" are fundamental components of digital arithmetic. of binary addition, the "SUM" refers to the result obtained when adding two

We currently possess a circuit that is capable of executing binary addition. The circuit under consideration is commonly referred to as a half adder circuit.

The full adder is a digital circuit that performs addition of two binary numbers. It is an essential component in arithmetic logic units (ALUs) and other digital systems.

Computers execute the process of addition on two binary numbers simultaneously, with each binary number potentially consisting of several binary digits. The process of addition commences by summing the least significant bits (LSBs) of the two given numbers. As an illustration

The Full Adder

Computer performs the addition operation on two binary number at a time, where each binary number can have several binary digits. The addition process starts by adding the least significant bits (LSBS) of the two numbers. For Example

01			0			1	
+11			+1			+1	
	Carry		Sum		Carry	Sum	
	0		1		1	0	
			1				
	Carry		Sum				
	1		0				
	↓						
	= 1	0	0				

There exist a total of eight distinct scenarios when considering three input variables, with each scenario having a corresponding set of desirable output values. As an illustration, let us examine the scenario where A is assigned the value of 1, B is assigned the value of 0, and C is assigned the value of 1. The full adder, commonly denoted as FA, is required to perform the addition of these binary bits in order to yield a sum of 0 and a carry out of 1.

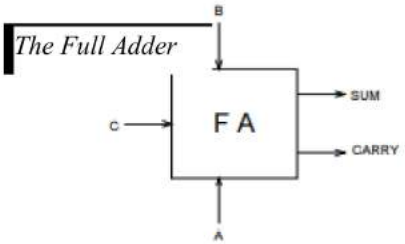
Given the presence of two outputs, it is necessary to develop separate circuitry for each output. The initial focus will be on designing the circuitry for the s output. The above table illustrates four instances in which the variable s is assigned a value of 1. By employing the sum-of-products approach, we may phrase the given statement as follows:

The provided information is in the form of a table. The truth table for a full adder is as follows:

A truth table for full adder.

A	B	C (Carry in)	S (Sum)	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

The Full Adder



truth table for a full adder.

n the present analysis, we will endeavor to simplify the given statement by the process of factoring. Regrettably, it is observed that none of the phrases within the given expression possess two variables that are shared with any other term. However, it is possible to factor out the variable A from the first two terms, and also factor out A from the last two terms.

Since there are two output, we will design the circuitry for each output individually, starting with the s output. The truth table shows that there are four cases where s is to be a 1. Using the sum-of products method, we can write for the expression as,

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

The initial expression within the parentheses should be identified as the logical operation of exclusive OR between variables B and C, denoted as B C. The subsequent expression within the parentheses should be recognized as the logical operation of exclusive NOR between variables B and C, denoted as B $\oplus$ C. Hence, the equation representing the value of S can be derived as follows.

The mathematical representation of the summation operation

The expression for SUM.

$$S = \bar{A}\,(\,\bar{B}C + B\,\bar{C}\,) + A(\,\bar{B}\,\bar{C} + BC\,)$$

The first term in parentheses should be recognized as the exclusive OR combination of B and C This can be written as B C. The second term in the parenthesis should be recognized as the exclusive NOR of B and C And this can be written as B $\oplus$ C . Thus the expression for S becomes.

$$S = \bar{A}(B \oplus C) + A(\overline{B \oplus C})$$

Let us consider the equation X = B ⊕ C. The aforementioned equation can be expressed as,

$$S = \bar{A}X + A\bar{X} = A \oplus X$$

The equation (1) can be expressed as S equals A XOR (B XOR C), where XOR represents the exclusive OR operation.

S = A ⊕ [B ⊕ C]

(1)

Let us now examine the output carry out as presented in truth table 2.2. The sum-of-products expression for the carry out can be written as follows

The expression for CARRY

$$\text{Carry out} = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

The given expression can be simplified using the process of factoring. In this statement, we will utilize a technique by using the ABC term on three separate occasions. This phenomenon occurs due to the presence of shared factors between the given word and each of the remaining terms. Therefore,

$$\begin{aligned}\text{Carry out} &= ABC + \bar{A}BC + ABC + A\bar{B}C + ABC + AB\bar{C} \\ &= BC(A + \bar{A}) + AC(\bar{B} + B) + AB(C + \bar{C}) \\ &= BC + AC + AB \qquad (2)\end{aligned}$$

The given expression has reached its simplest form and cannot be further reduced. Expressions (1) and (2) can be implemented in the manner depicted in Figure 1. The entire adder is represented by a comprehensive circuit that includes inputs A, B, and C, as well as outputs S and carry out.

A full adder circuit constructed by using two half adder circuit.

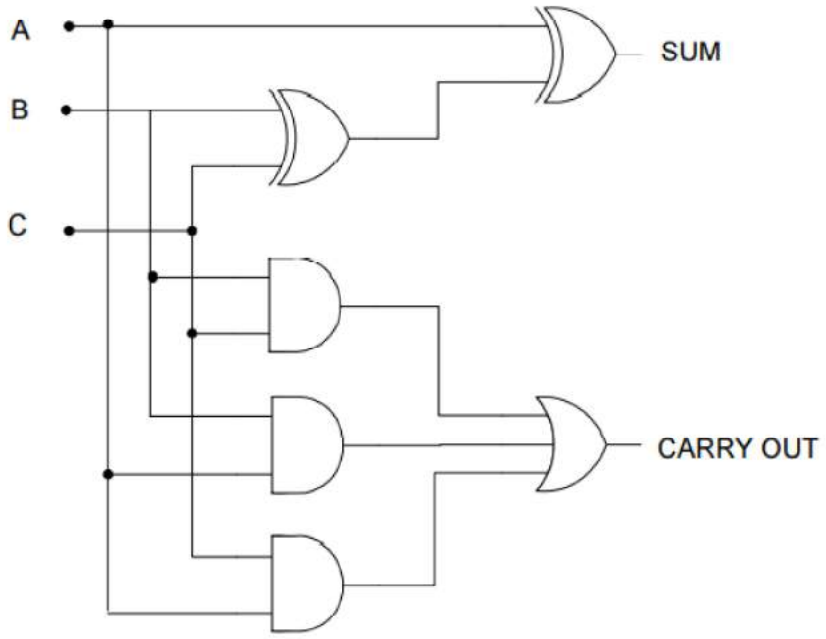


Fig. Complete circuitry of a full adder.

A full adder circuit can be constructed using two half adders. Two of the three inputs are connected to the first half adder which produces a partial sum and partial carry output. The partial sum is fed to the second half adder along with the third of the original inputs. This causes the final sum to be produced and also another partial carry.

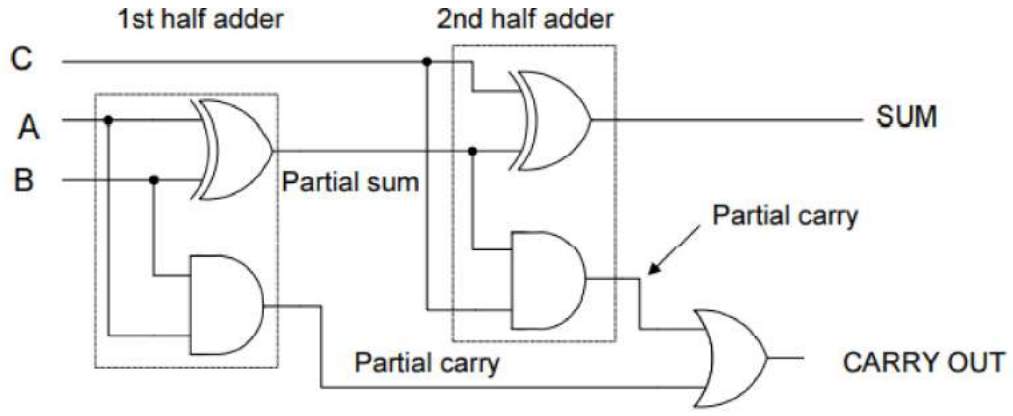


Fig. A full adder circuit using half adder.

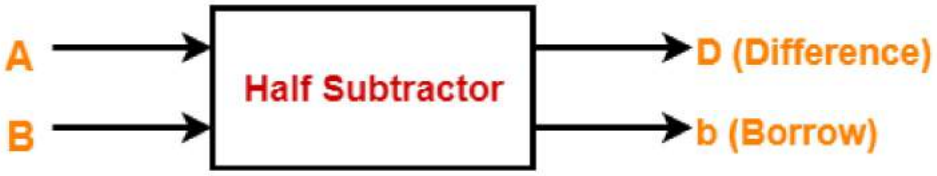
The complete circuitry of a full adder encompasses the integration of several electronic components and logic gates, which collectively enable the addition of two binary numbers.

A complete adder circuit can be created by employing two half adders. The first half adder is connected to two out of the three inputs, resulting in the generation of a partial sum and partial carry output. The partial total is transmitted to the second half adder in conjunction with one-third of the initial inputs. This results in the generation of the final amount as well as an additional partial carry.

3.4.2 Half Subtractor and full Subtractor

Half Subtractor

Half Subtractor and Full Subtractor In the realm of digital electronics, the concepts of half subtractors and full subtractors hold significant importance. These subtractors are fundamental building blocks that facilitate subtraction operations in binary arithmetic. A half subtractor is a combinational circuit



The figure. The following is a depiction of a block diagram illustrating the functioning of a half subtractor

Half Subtractor Design

- The design of a half subtractor is a fundamental component in digital logic circuits used for arithmetic operations. It is responsible for subtracting two single-bit binary numbers and producing the

The first step is to...

The input and output variables will be identified.

The input variables, denoted as A and B, can take on binary values of either 0 or 1.

The output variables in this context are denoted as D and b. Here, D represents the difference, while b represents the amount borrowed.

The second step is to...

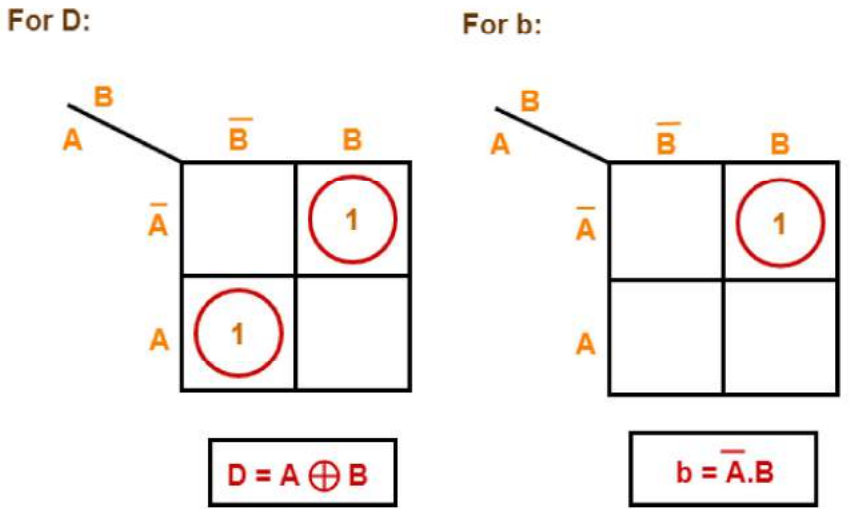
Please construct the truth table.

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:

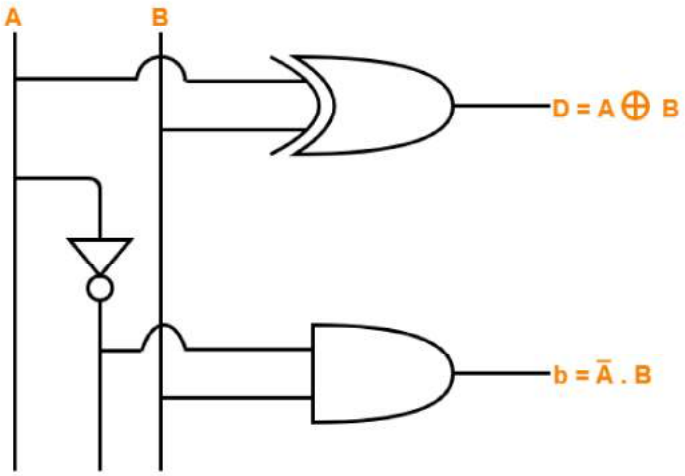
The K-maps should be constructed based on the provided truth table, and subsequently, the simplified Boolean expressions can be determined.



K Maps

Please create a logic diagram.

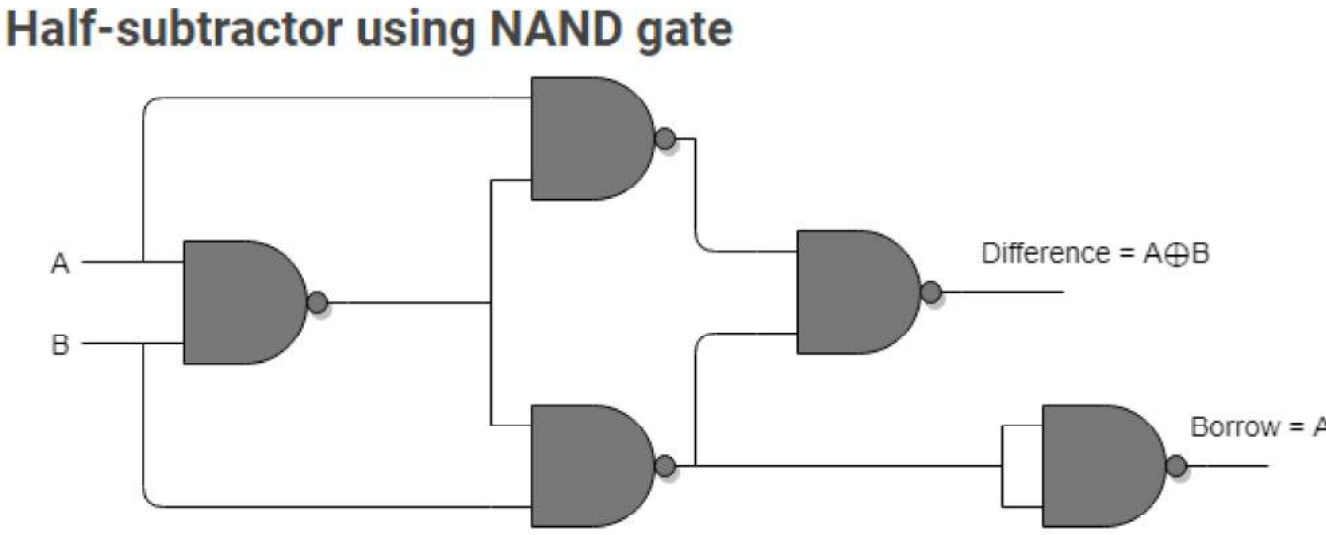
- The following diagram illustrates the creation of a half subtractor utilizing one XOR gate, one NOT gate, and one AND gat



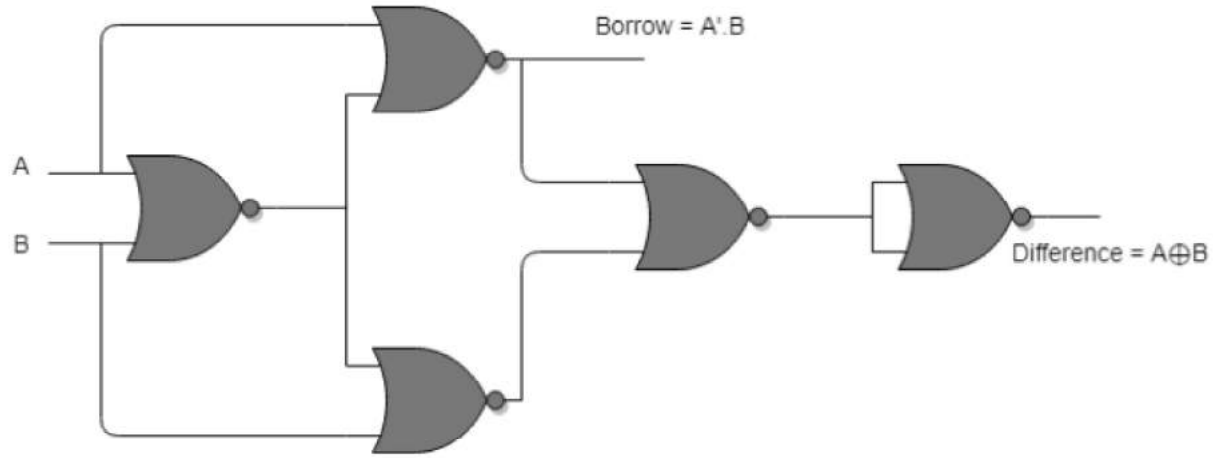
Half Subtractor Logic Diagram

Figure. The following is a depiction of a logic diagram representing a half subtractor.

the implementation of a half subtractor circuit utilizing NAND and NOR gates.



Half-subtractor using NOR gate



Full Subtractor

The Full Subtractor is a type of combinational logic circuit.

The function of this operation is to perform the subtraction of two binary numbers consisting of a single bit each.

It also considers the borrowing of the lower significant stage.

The complete subtractor is capable of doing the subtraction operation on three bits.

The full subtractor is a digital circuit that consists of three inputs and two outputs, namely the Difference and Borrow.

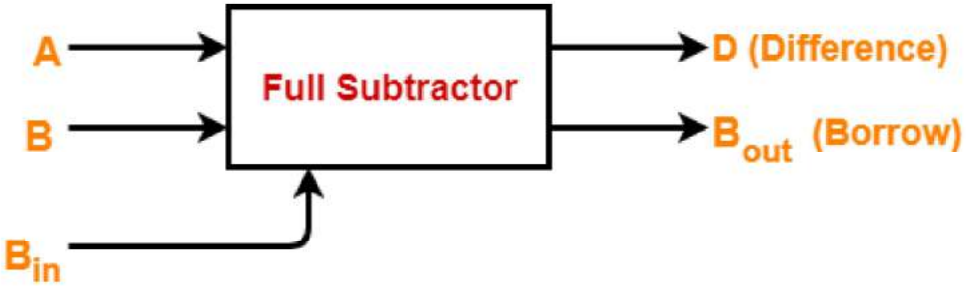


Figure 1 illustrates the block diagram of a complete subtractor.

Designing a Full Subtractor

The first step is to...

The input and output variables should be identified in the given context.

The input variables in this study are denoted as A, B, and Bin, where Bin can take on values of either 0 or 1.

The output variables in this context are denoted as D, representing the difference, and Bout, representing the borrow.

Step-02:

Please construct the truth table.

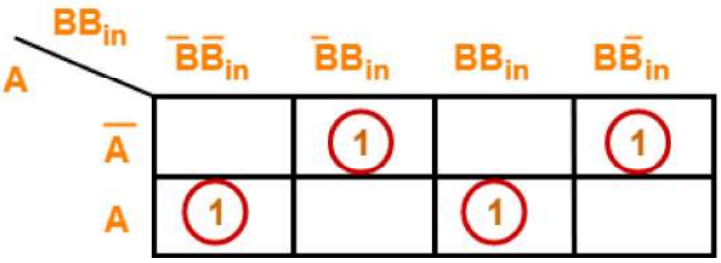
Inputs			Outputs	
A	B	B_{in}	B_{out} (Borrow)	D (Difference)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Truth Table

The third step in the process is as follows:

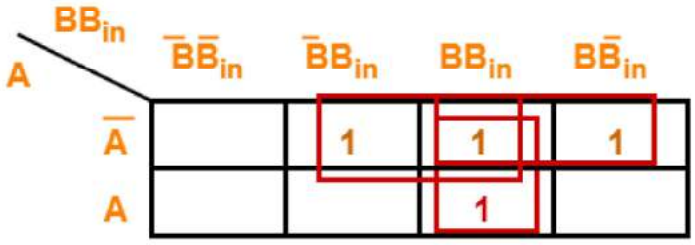
The K-maps should be constructed based on the provided truth table, and subsequently, the simplified Boolean expressions can be determined.

For D:



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

For B_{in}:



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

In the fourth step,

Please create a logic diagram.

The following diagram illustrates the implementation of a full adder utilizing one XOR gate, three AND gates, one NOT gate, and one OR gate.

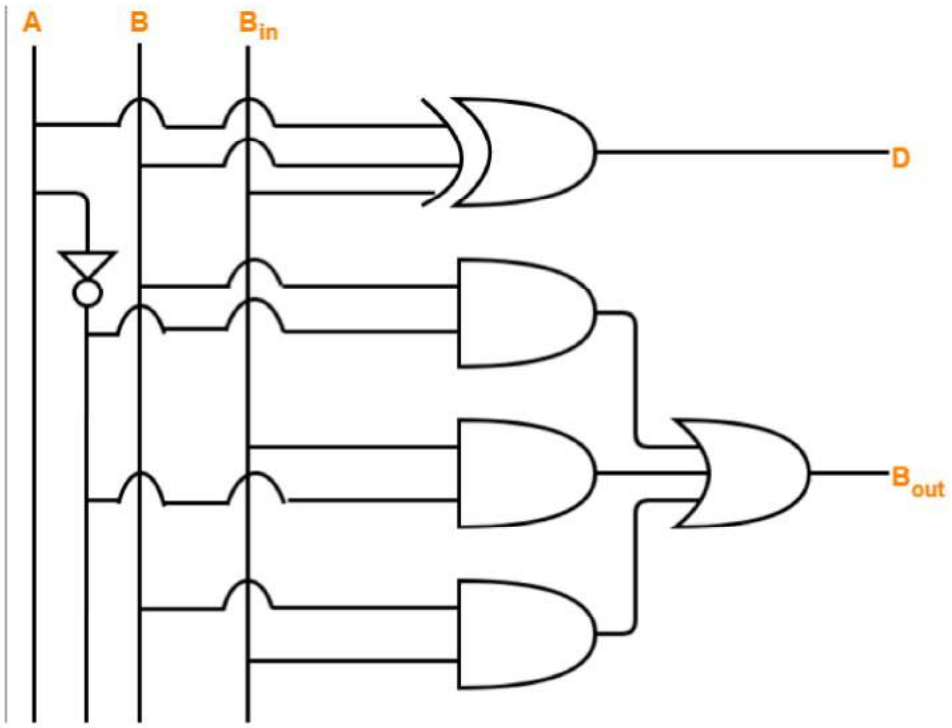


Figure. The purpose of this response is to provide an academic rewrite of the user's text without adding any additional information. A logic diagram illustrating the functionality of a full subtractor will be presented.

3.4.3 Parallel Adder

The topic of discussion is the parallel adder, specifically in the context of digital circuits.

A solitary complete adder executes the operation of adding two one-bit values along with an input carry.

The structure is comprised of a series of interconnected full adders, in which the output carry of each full adder is linked to the carry input of the subsequent full adder of higher order within the chain.

To execute the operation, an n-bit parallel adder necessitates the utilization of n complete adders.

In the case of a two-bit number, it is necessary to employ two adders, whereas for a four-bit number, four adders are required, and this pattern continues for higher bit numbers.

In the realm of digital circuit design, it is customary for parallel adders to integrate carry lookahead logic as a means of guaranteeing that the speed of addition is not hindered by the propagation of carry signals between consecutive stages of the addition process.

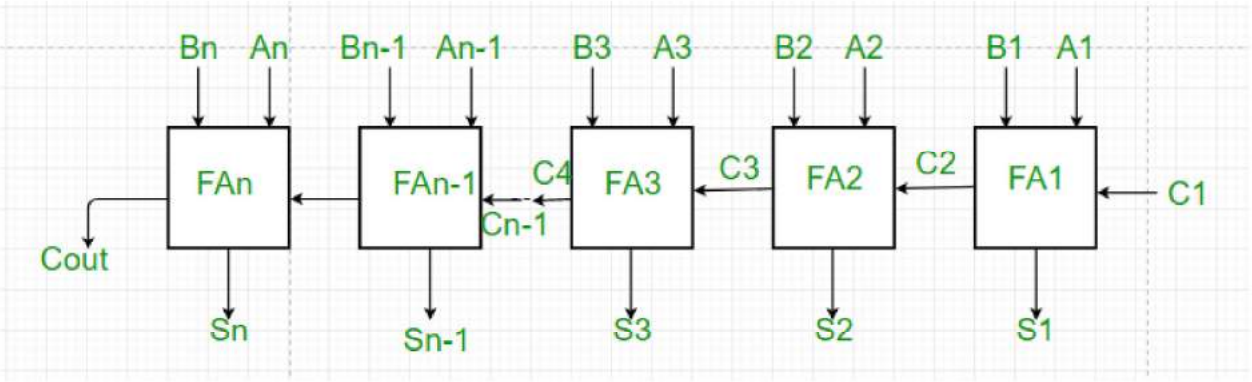


Figure. A parallel adder is a digital circuit that performs the addition operation on multiple binary numbers simultaneously.

Working of parallel Adder

The operation of a parallel adder

The initial step involves the addition of A1 and B1, along with the carry C1, by the full adder FA1. This operation produces the total S1, which represents the first bit of the output sum, as well as the carry C2, which is subsequently linked to the subsequent adder in the chain.

Subsequently, the full adder FA2 incorporates the carry bit C2 in order to perform addition with the input bits A2 and B2, resulting in the generation of the sum S2 (representing the second bit of the output sum) and the carry C3. The carry C3 is subsequently connected to the subsequent adder in the chain, thereby continuing the process.

The process persists until the last full adder (FAn) utilizes the carry bit (Cn) to perform addition with its input values (An and Bn), resulting in the generation of the final bit of the output and the last carry bit (Cout).

Parallel Subtractor

The parallel subtractor is a digital circuit used to perform subtraction operations in parallel. It is commonly used in computer systems and other electronic

A Parallel Subtractor refers to a digital circuit that possesses the ability to calculate the arithmetic difference between two binary integers, which are longer than a single bit, by simultaneously processing corresponding pairs of bits.

The parallel subtractor can be implemented using several design approaches, such as a combination of half and full subtractors, exclusively full subtractors, or exclusively full adders with a subtrahend complement input.

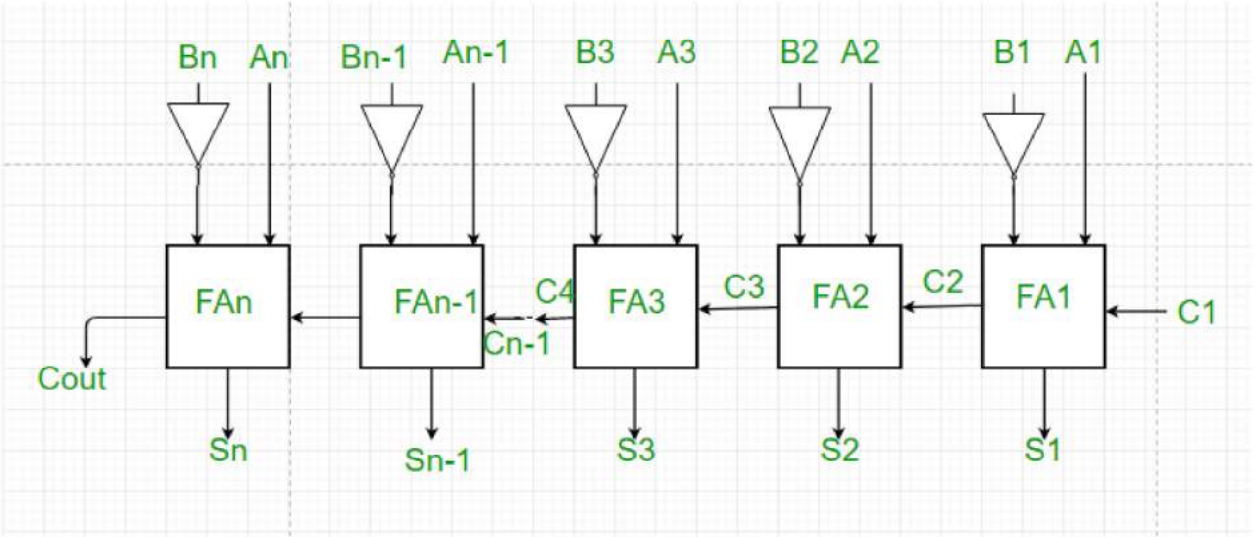


Figure. The concept of a parallel subtractor refers to a computational device or circuit that is designed to perform the operation of subtraction in a parallel

The operational principles of a parallel subtractor

The parallel binary subtractor is constructed by integrating several complete adders with a subtrahend complement input.

This operation postulates that the sum of the minuend and the 2's complement of the subtrahend is equivalent to their difference.

The 1's complement of B is derived by using the NOT gate operation. The 2's complement of B can then be acquired by adding 1 through the carry. Additionally, A is utilized to perform the operation of arithmetic subtraction.

The process is iterated until the final complete adder, denoted as FAn, utilizes the carry bit Cn to perform addition with its input An and the 2's complement of Bn. This results in the generation of the final bit of the output, as well as the last carry bit Cout.

-

The parallel Adder/Subtractor possesses several advantages.

The parallel adder/subtractor has a higher speed of operation in comparison to the serial adder/subtractor.

The duration of the addition process is independent of the number of bits involved.

The resulting output is in parallel form, meaning that all the bits are simultaneously added or deleted.

It is more cost-effective.

There are some drawbacks associated with the implementation of parallel Adder/Subtractor.

Every individual adder must wait for the carry signal to be created by the preceding adder in the chain.

The increase in the number of bits to be added is observed to result in an increase in the propagation delay, which is the delay associated with the passage of the carry bit.

3.5 Decimal or Binary-Coded Decimal (BCD) Adder

The BCD-Adder is commonly employed in computer systems and calculators for performing arithmetic operations directly within the decimal number system.

The BCD-Adder is designed to process decimal numbers in their binary-coded representation. The Decimal-Adder necessitates a minimum of nine input variables and five output variables.

In the BCD code, the representation of the decimal number necessitates the utilization of 4 bits. Additionally, the circuit must possess both an input carry and an output carry.

The method of binary coded decimal addition is depicted as follows:

The user's text is already academic and does not need to be rewritten. Please provide the BCD code groups for each decimal digit location by use standard binary addition.

The user's text is too short to be rewritten in an academic manner. For positions in which the sum is equal to or less than 9, the amount is represented in the right Binary-Coded Decimal (BCD) format, and no adjustment is required.

The user's text does not provide any information to rewrite in an academic manner. In cases when the sum of two digits exceeds 9, it is necessary to apply a correction of 0110 to the sum in order to obtain the correct Binary-Coded Decimal (BCD) representation. This will result in a carry that will be added to the subsequent decimal position.

The process of binary coded decimal addition is depicted in the following manner:

The user's text is already academic. Please provide the BCD code groups for each decimal digit location by use standard binary addition.

The user's text is too short to be rewritten academically. For those positions where the sum is 9 or fewer, the sum is in proper BCD form and no correction is needed.

- 3. When the sum of two digits is more than 9, a correction of 0110 should be added to that sum to achieve the right BCD result. This will cause a carry to be added to the next decimal position.

For example, if the two BCD code groups represented by, A3A2A1A0 and B3B2B1B0 , respectively, are applied to a 4-bit parallel adder, the adder will perform the following operation :

-

$$
\begin{aligned}
&A_3A_2A_1A_0 \leftarrow \text{BCD code group}\\
+&\underline{B_3B_2B_1B_0} \leftarrow \text{BCD code group}\\
&S_4S_3S_2S_1S_0 \leftarrow \text{straight binary sum}
\end{aligned}
$$

S_4 is a actually C_4, the carry out of the MSB.

Inputs				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Table: Truth table for BCD adder

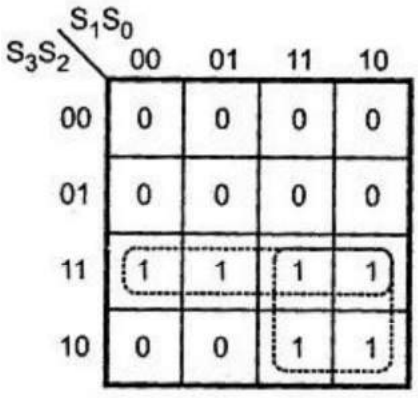


Fig: K-map

Let's define y as a logic output that will go HIGH only when the sum is larger than 01001. If we investigate these situations, it may be reasoned that y will be HIGH for any of the following conditions.

1. Whenever S4 = 1 (sums more than 15)

2. Whenever S3 = 1 and either S2 or S1 or both are 1 (sums 10 to 15)

This can be represented as y = S4 + S3 (S2 + S1)

Whenever y = 1, it is essential to apply the correction 0110 to the sum bits and to generate a carry.

Figure displays the complete circuitry for a BCD adder, including the logic-circuit implementation for y.

•

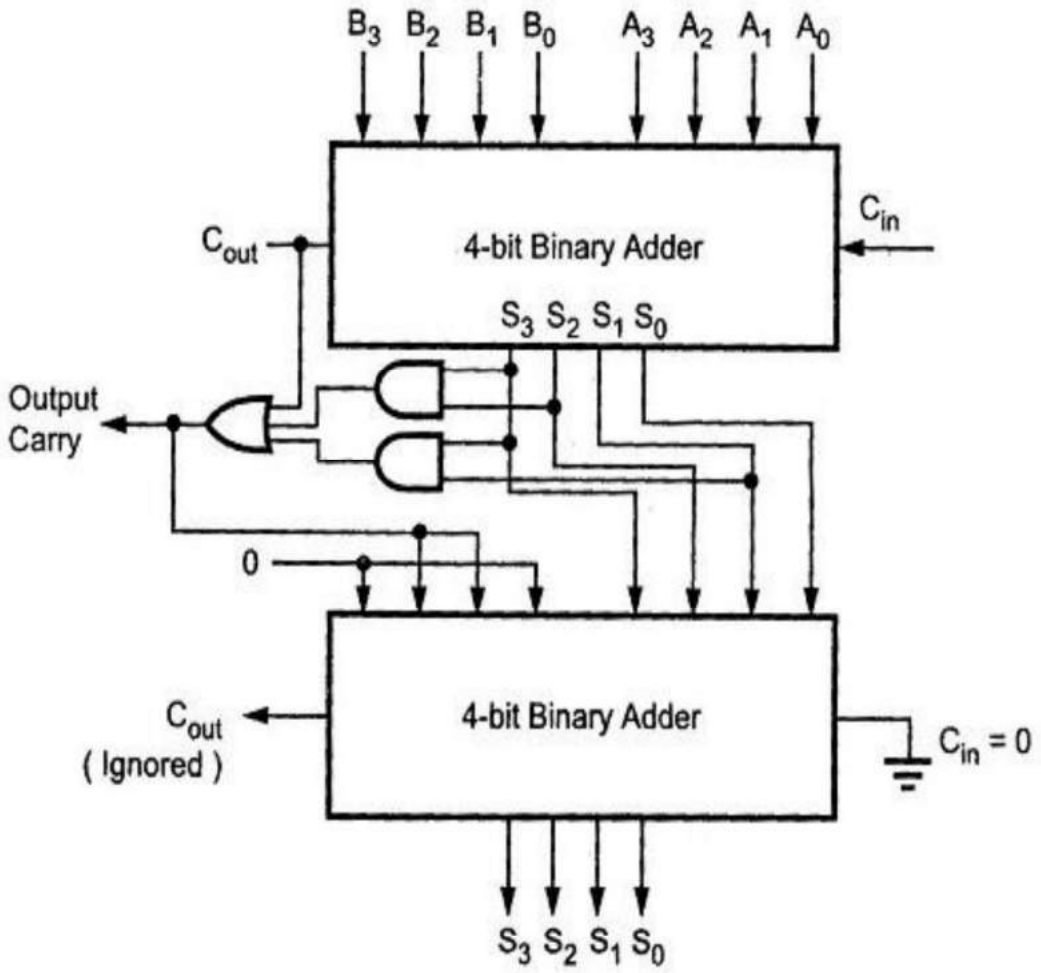


Fig. A BCD adder has two 4-bit adders and correction detector circuit.

The circuit consists of three fundamental elements.

The two code groups A3A2A1A0 and B3B2B1B0 are combined together in the upper 4-bit adder to give the sum S4S3S2S1S0.

The logic gates implement the expression for y.

The two BCD integers, combined with input carry, are first added in the top 4-bit binary adder to obtain a binary sum.

When the output carry is equal to zero (i.e. when sum ≤ 9 and Cout = 0) nothing (zero) is added to the binary total.

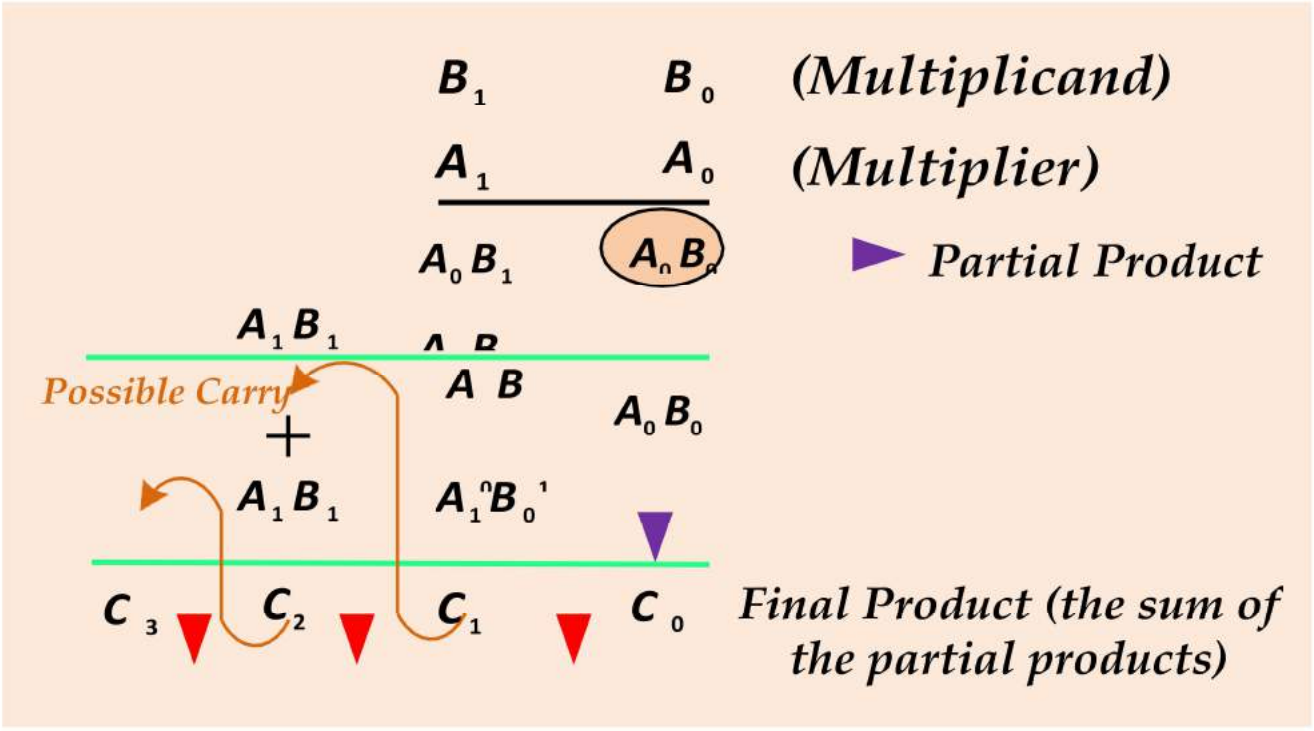
When it is equal to one (i.e. when sum > 9 or Cout = 1), binary 0110 is added to the binary sum through the bottom 4-bit binary adder.

The output carry generated from the bottom binary adder can be ignored, since it offers information already present at the output-carry terminal.

3.6 Binary Multipliers:

Multiplication of binary numbers is performed in the same way as multiplication of decimal numbers:

Multiplication of two 2-bit numbers.



Combinational circuit

The multiplication of two bits such as $A0$ and $B0$ produces a 1 if both bits are

1; otherwise, it produces a 0, this is identical to an AND operation.

The two partial products are added with two half-adder (HA) circuits (if there are more than two bits, we must use full adder (FA)).

☐ circuits (ifthere are more than two bits, we must use full adder (***FA***)).

Combinational circuit of binary multiplier with more bits.

☐ For J bits ***multiplier*** and K bits ***multiplicand***, we need:

$J \times K$ ***AND*** gates.

$(J - 1)K$-bits adders to produce a product of $J + K$ bits.

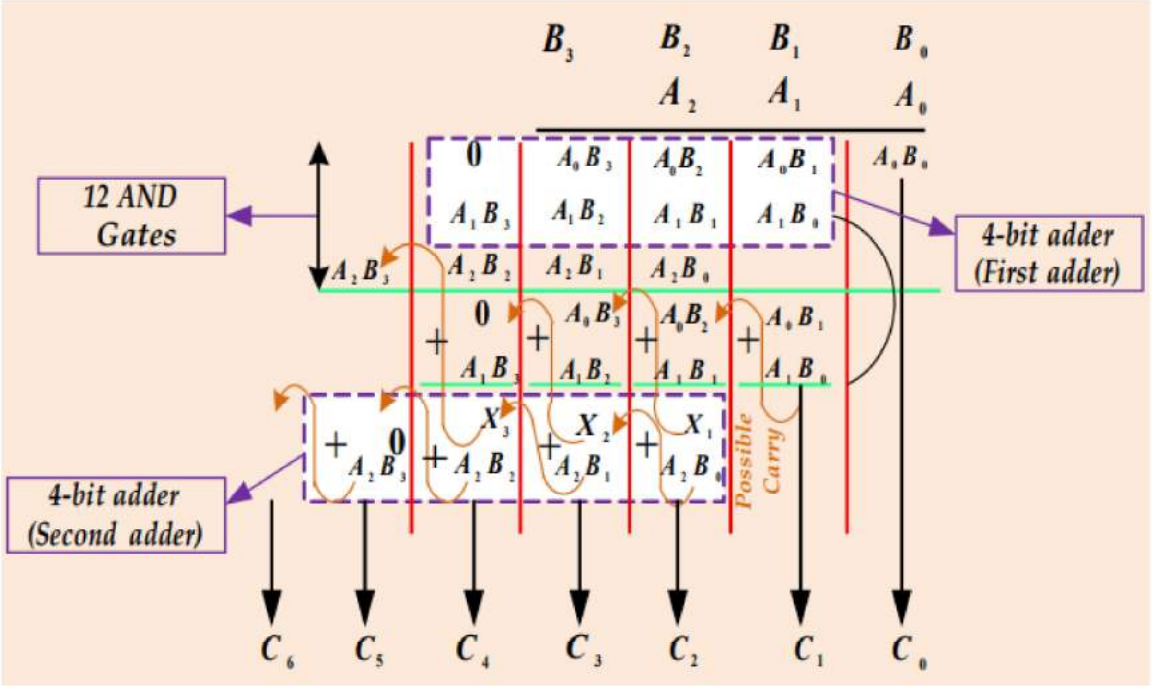
Example: - Create a logic circuit for

$$\begin{array}{r} B_3 B_2 B_1 B_0 \quad \times \\ A_2 A_1 A_0 \\ \hline C_6 C_5 C_4 C_3 C_2 C_1 C_0 \end{array} \quad \textit{Answer}$$

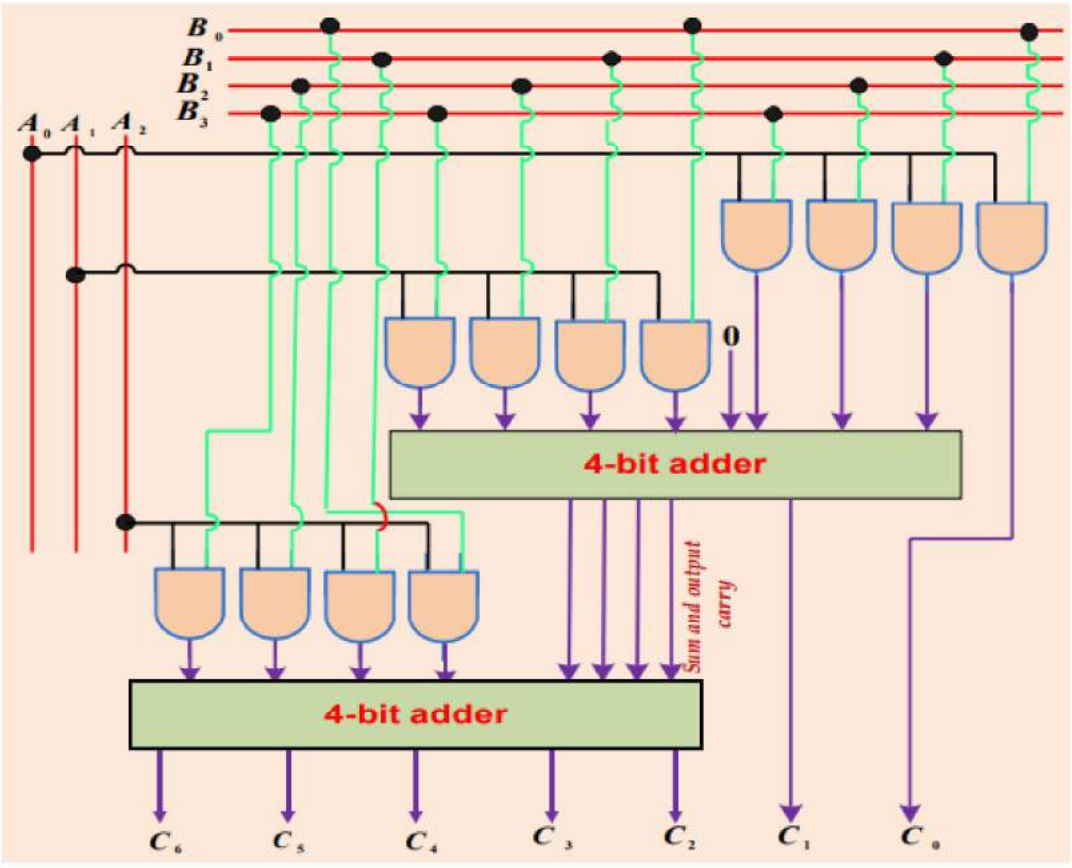
So:

We need ***12 AND*** gates and ***2 four-bit adders*** to produce a product of ***seven bits***:

$C_6C_5C_4C_3C_2C_1C_0$.



Circuit Implementation:



3.7 Magnitude Comparator in Digital Logic

A magnitude digital Comparator is a combinational circuit that compares two digital or binary numbers to determine if one is equal, less than, or more. We create a circuit with two inputs (A and B) and three outputs (A

> B, A = B, and A < B).

1-Bit Magnitude Comparator – A comparator used to compare two bits is called a single bit comparator. It consists of two inputs each for two single bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers. The truth table for a 1-bit comparator is given below:

A	B	A<B	A=B	A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

From the above truth table logical expressions for each output can be expressed as follows:

```
A>B: AB'
A<B: A'B
A=B: A'B' + AB
```

from the above expressions we can derive the following formula:

(A<B)+(A>B) = A'B+AB'
Taking complement both sides

((A<B) + (A>B))' = (A'B + AB')'

((A<B) + (A>B))' = (A'B)' (AB')'

((A<B) + (A>B))' = (A + B') (A' +B)

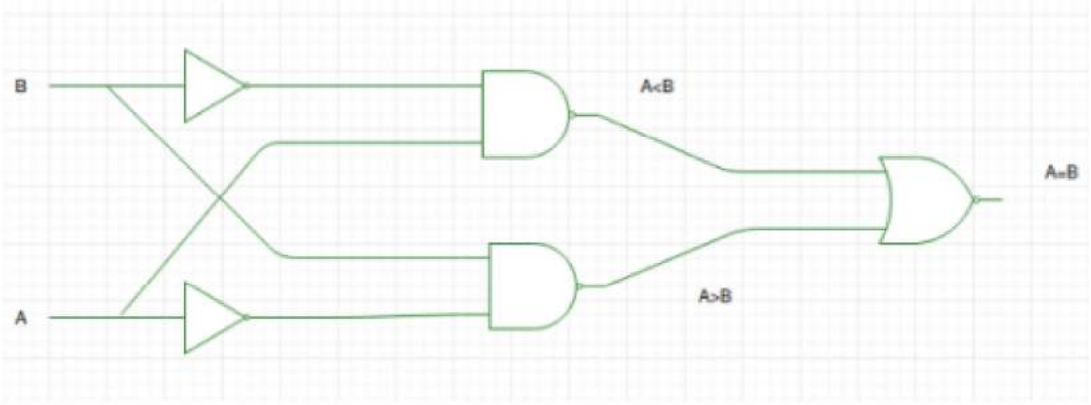
((A<B) + (A>B))' =(AA' + AB + A'B' +BB')

" " = (AB + A'B')

Thus,

((A<B) + (A > B))' = (A = B)

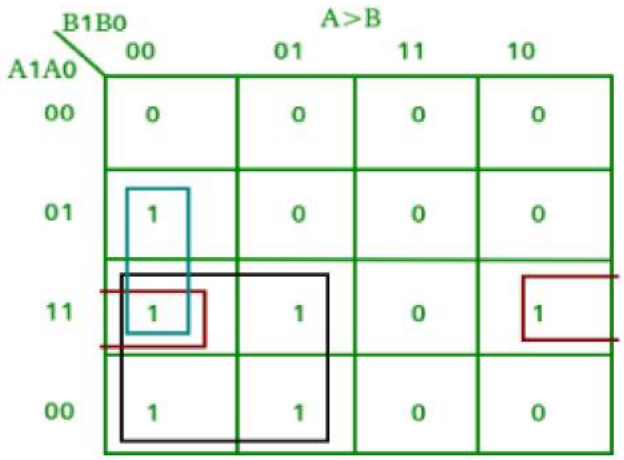
By using these Boolean expressions, we can implement a logic circuit for this comparator as given

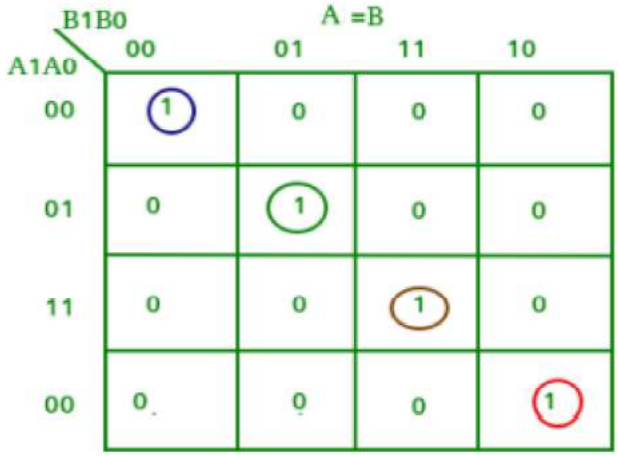


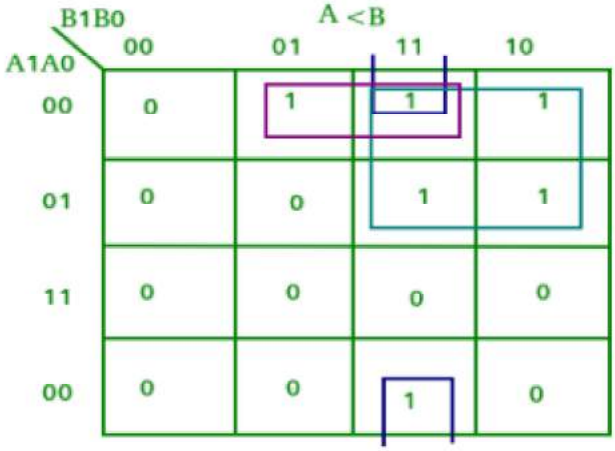
2-Bit Magnitude Comparator – A comparator used to compare two binary numbers each of two bits is called a 2-bit Magnitude comparator. It consists of four inputs and three outputs to generate less than, equal to and greater than between two binary numbers. The truth table for a 2-bit comparator is given below:

INPUT				OUTPUT		
A1	**A0**	**B1**	**B0**	**A<B**	**A=B**	**A>B**
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

From the above truth table K-map for each output can be drawn as follows:







From the above K-maps logical expressions for each output can be expressed as follows

A>B:A1B1’ + A0B1’B0’ + A1A0B0’

A=B: A1’A0’B1’B0’ + A1’A0B1’B0 + A1A0B1B0 + A1A0’B1B0’

: A1’B1’ (A0’B0’ + A0B0) + A1B1 (A0B0 + A0’B0’)

: (A0B0 + A0’B0’) (A1B1 + A1’B1’)

: (A0 Ex-Nor B0) (A1 Ex-Nor B1)

A<B:A1’B1 + A0’B1B0 + A1’A0’B0

OR

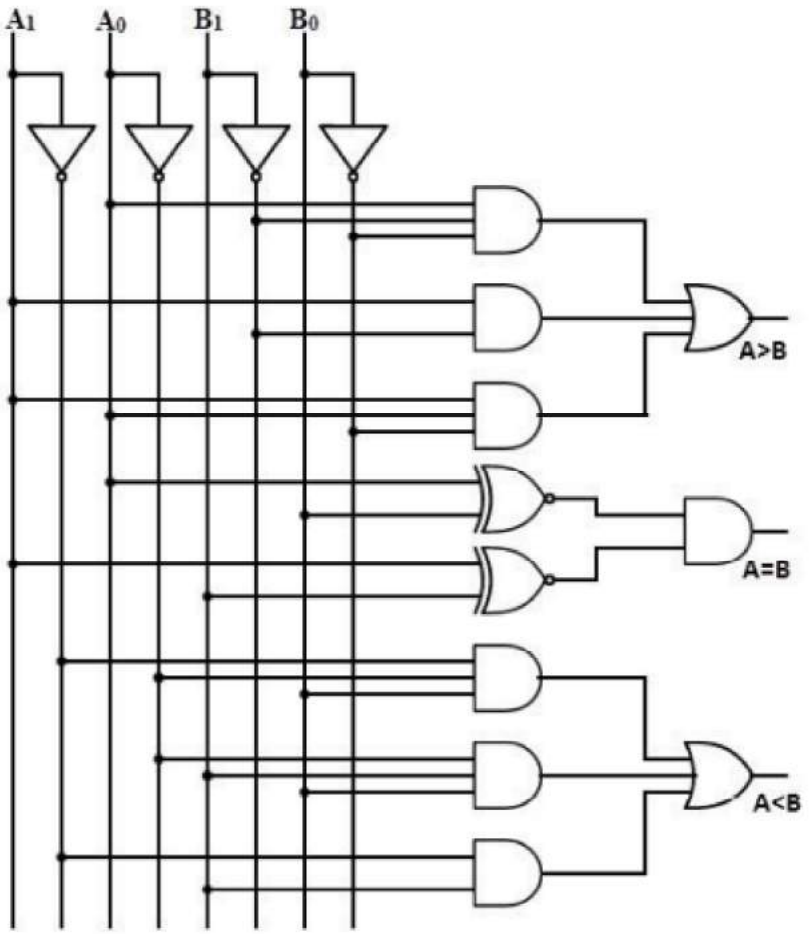
A>B = A1B1’ + B0 (A0B1’ + A0A1)

A=B = $\overline{A0 \oplus B0} \cdot \overline{A1 \oplus B1}$

A<B = B1A1’ + B0B1A0’ + A1’A0’B0

:

By using these Boolean expressions, we can implement a logic circuit for this comparator.



Applications of Comparators –

1. Comparators are used in central processing units (CPUs) and microcontrollers (MCUs).

2. These are used in control applications in which the binary numbers representing physical variables such as temperature, position, etc. are compared with a reference value.

3. Comparators are also used as process controllers and for Servo motor control.

4. Used in password verification and biometric applications

3.8 Decoder

A binary code of n bits is capable of representing up to 2n distinct elements of coded information. A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2n unique output lines. If the n -bit coded information has unused combinations, the decoder may have fewer than 2n outputs.

Figure shows a three-to-eight-line decoder circuit. One of the three input variables' minterms is represented by each of the eight outputs. The three inverters complement the inputs, and each of the eight AND gates yields a minterm. Using this decoder for binary-to-octal conversion. The input variables are binary numbers, while the outputs are octal numbers. However, a three-to-eight-line decoder can decode any three-bit code to produce eight outputs, one for each element.

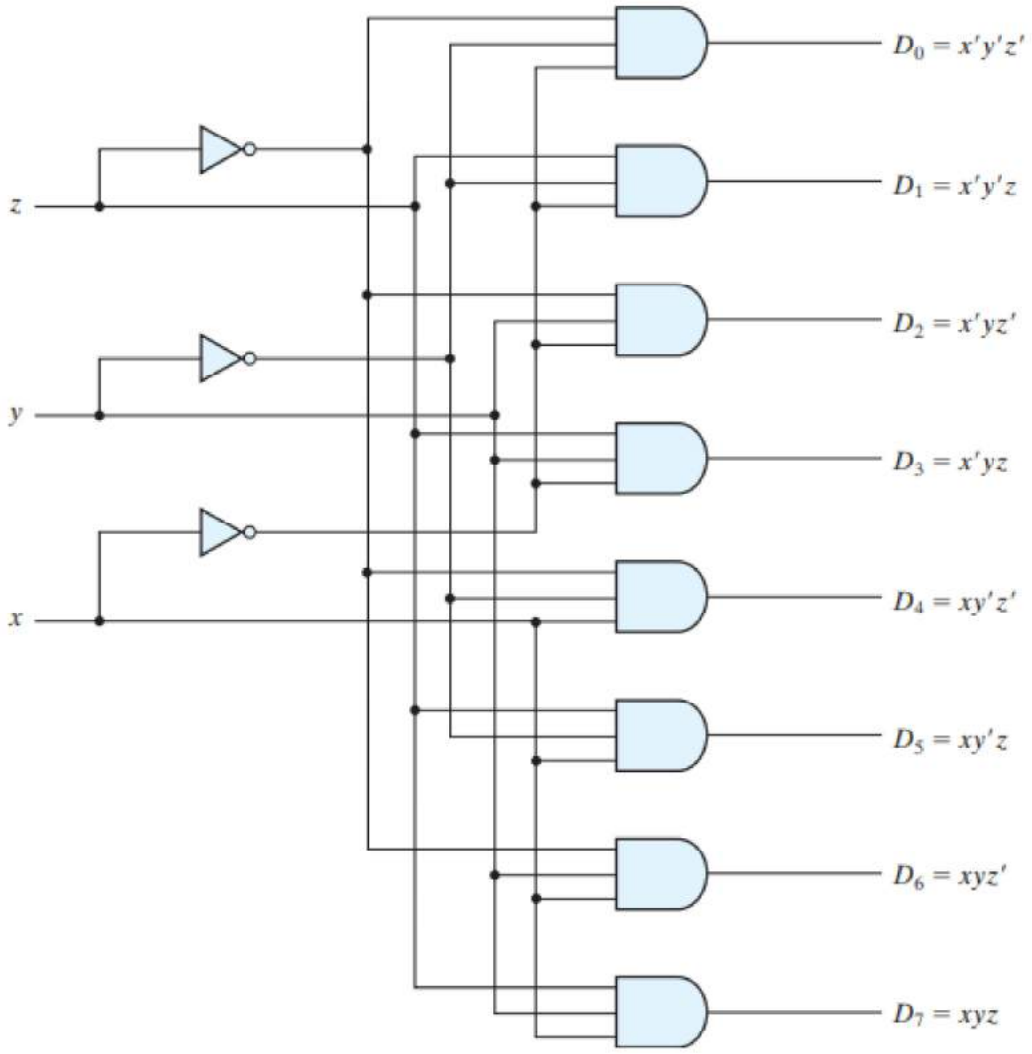


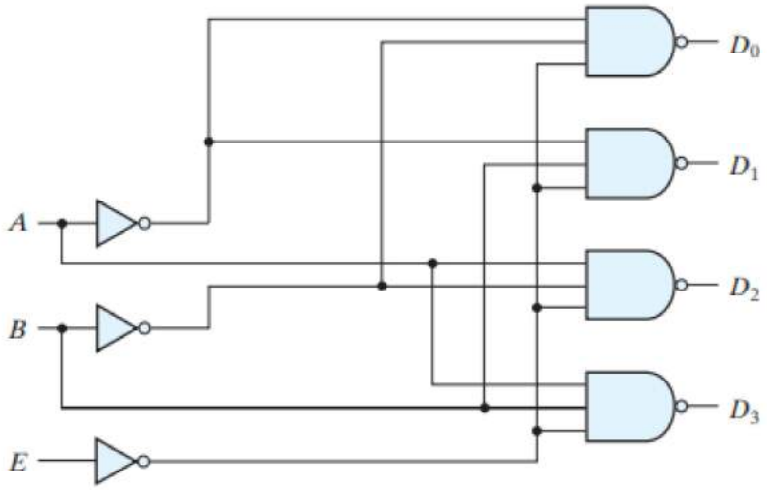
Fig.Three-to-eight-line decoder

Truth Table of a Three-to-Eight-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 shows a three-to-eight-line decoder circuit. One of the three input variables' minterms is represented by each of the eight outputs. The

three inverters complement the inputs, and each of the eight AND gates yields a minterm. Using this decoder for binary-to-octal conversion. The input variables are binary numbers, while the outputs are octal numbers. A three-to-eight-line decoder can decode any three-bit code into eight outputs, one for each element.



(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	1

(b) Truth table

Fig. Two-to-four-line decoder with enable input

There are seven 0 outputs and one 1 output for each input combination. A 1 output represents the minterm equivalent of the binary number in the input lines. Some decoders use NAND gates.

Decoders have enable inputs to control circuit activity. Figure following shows a two-to-four-line decoder with an enable input made of NAND gates. With complement enabling input and outputs, the circuit works. E=0 (active-low enable) enables the decoder. The truth table shows that only one output can be 0 at a time; all others are 1. The minterm determined by inputs A and B is the 0 output. The circuit is disabled when E = 1, independent of the other two inputs. No outputs are 0 and no minterms are selected when the circuit is disabled..

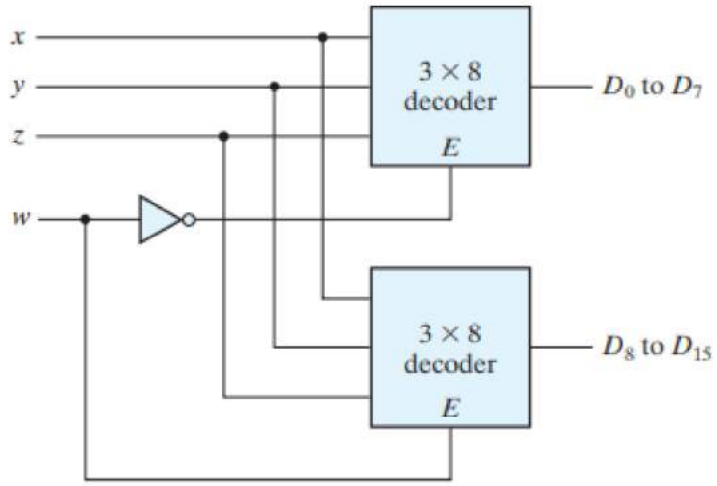


Fig. 4 * 16decoder constructed with two 3 * 8decoders

Ex. From the truth table of the full adder, we obtain the functions for the combinational circuit in sum-of-minterms form: S(x, y, z) = 1, 2, 4, 7) C(x, y, z) =3, 5, 6, 7)

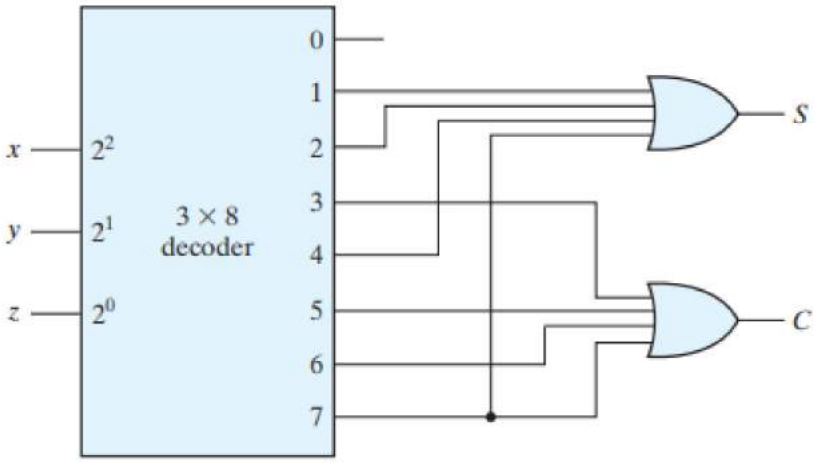


Fig. Implementation of a full adder with a decoder

3.9 Encoders

An Encoder is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2n input lines and 'n' output lines. It will

produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2n input lines with 'n' bits. It is optional to represent the enable signal in encoders.

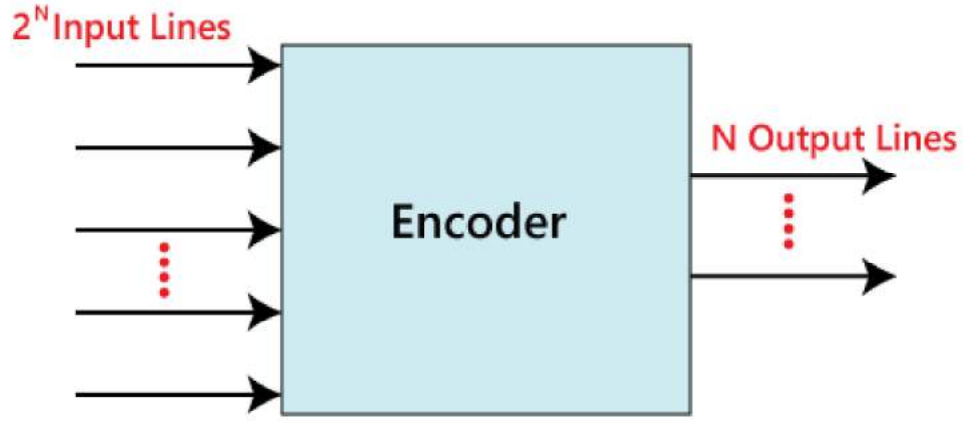


Fig.Block diagram of encoder

4 to 2 Encoder

Let 4 to 2 Encoder has four inputs Y3, Y2, Y1 & Y0 and two outputs A1 & A0. The block diagram of 4 to 2 Encoder is shown in the following figure.

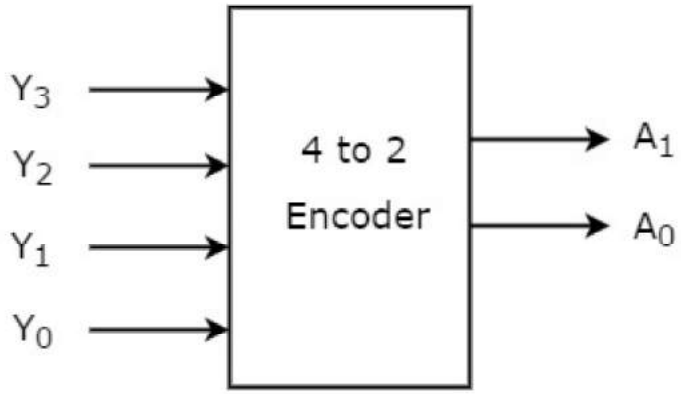


Fig. 4 to 2 Encoder

At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output. The Truth table of 4 to 2 encoder is shown below.

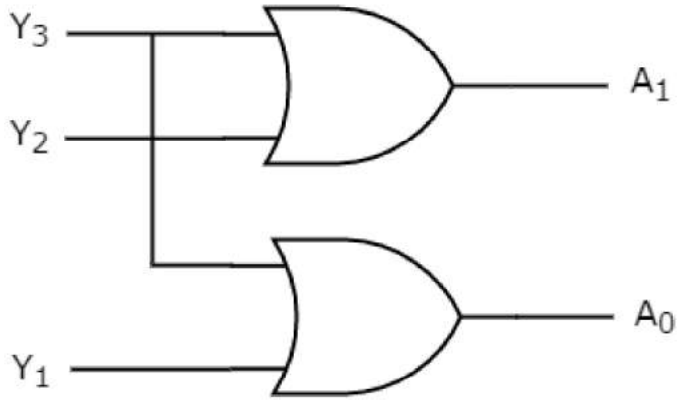
Inputs				Outputs	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

From Truth table, we can write the Boolean functions for each output as

A1=Y3+Y2

A0=Y3+Y1

We can implement the above two Boolean functions by using two input OR gates. The circuit diagram of 4 to 2 encoder is shown in the following figure.



The above circuit diagram contains two OR gates. These OR gates encode the four inputs with two bits.

8 to 3 line Encoder

The 8 to 3 line Encoder is also known as Octal to Binary Encoder. In 8 to 3 line encoder, there is a total of eight inputs, i.e., Y0, Y1, Y2, Y3, Y4, Y5, Y6, and Y7 and three outputs, i.e., A0, A1, and A2. In 8-input lines, one input-line is set to true at a time to get the respective binary code in the output side. Below are the block diagram and the truth table of the 8 to 3 line encoder.

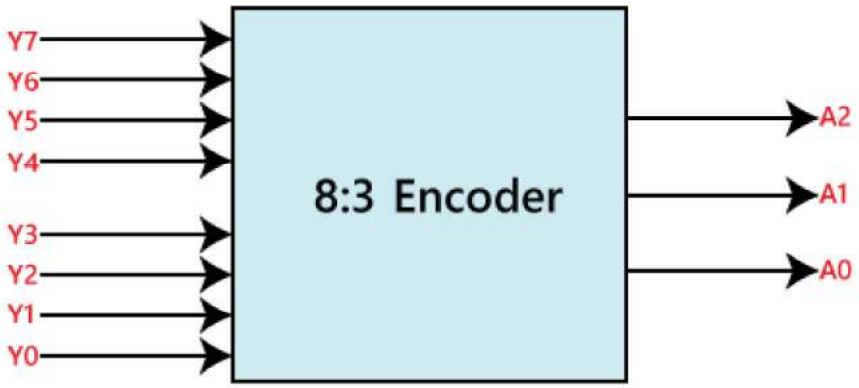


Fig. Block diagram of 8x3 encoder

At any time, only one of these eight inputs can be '1' in order to get the respective binary code. The Truth table of octal to binary encoder is shown below.

INPUTS								OUTPUTS		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Truth table

Truth table

The logical expression of the term A0, A1, and A2 are as follows:

A2=Y4+Y5+Y6+Y7A1=Y2+Y3+Y6+Y7A0=Y7+Y5+Y3+Y1

Logical circuit of the above expressions is given below:

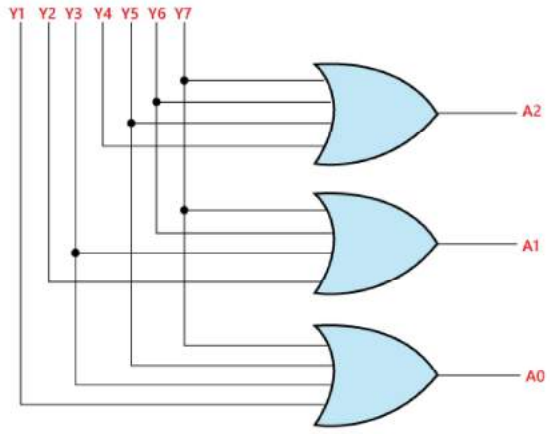
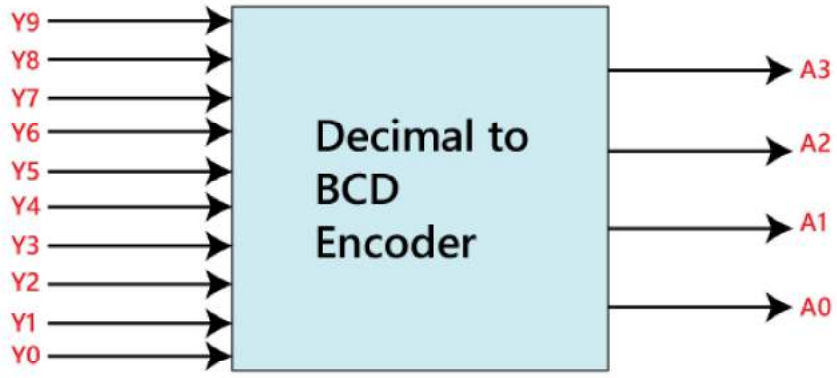


Fig. Logic diagram of 8x3 encoder

Decimal to BCD Encoder

Octal to Binary Encoders are also called 10 to 4 line encoders. The 10 to 4 line encoder has ten inputs (Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, and Y9) and four outputs (A0, A1, A2, and A3). Output BCD codes are obtained by setting one input line to true at a time in 10-input lines. Below are the decimal to BCD encoder block schematic and truth table..

Block Diagram



Truth Table:

INPUTS										OUTPUTS			
Y_9	Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_3	A_2	A_1	A_0
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

The logical expression of the term A0, A1, A2, and A3 is as follows:

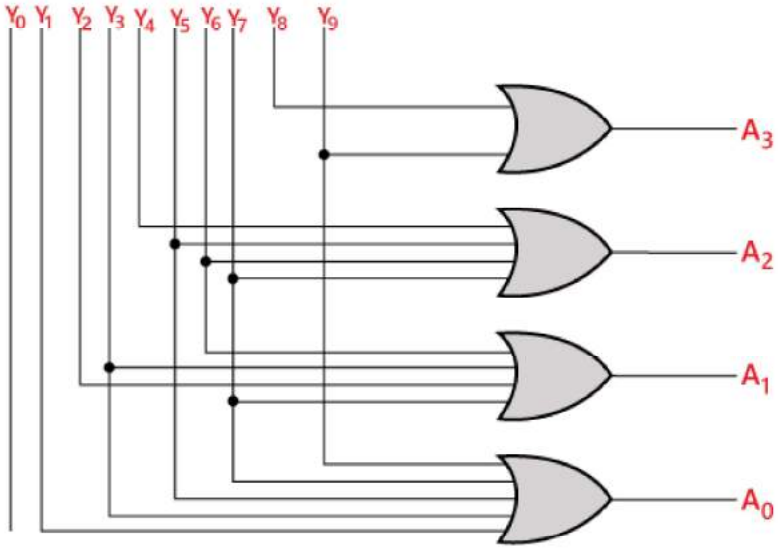
A3 = Y9 + Y8

A2 = Y7 + Y6 + Y5 +Y4

A1 = Y7 + Y6 + Y3 +Y2

A0 = Y9 + Y7 +Y5 +Y3 + Y1

Logical circuit of the above expressions is given below:



Drawbacks of Encoder

Following are the drawbacks of normal encoder.

There is an ambiguity, when all outputs of encoder are equal to zero. Because, it could be the code corresponding to the inputs, when only least significant input is one or when all inputs are zero.

If more than one input is active High, then the encoder produces an output, which may not be the correct code. For example, if both Y3 and Y6 are '1', then the encoder produces 111 at the output. This is neither equivalent code corresponding to Y3, when it is '1' nor the equivalent code corresponding to Y6, when it is '1'.

So, to overcome these difficulties, we should assign priorities to each input of encoder. Then, the output of encoder will be the binary code corresponding to the active High inputs, which has higher priority. This encoder is called as priority encoder.

3.10 Priority Encoder:

4 to 2 line Priority Encoder

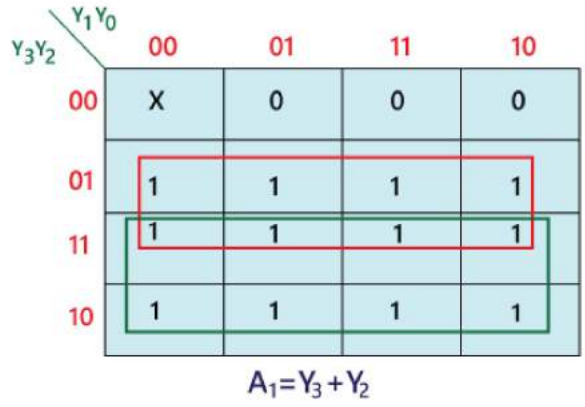
In this priority encoder, there are total of 4 inputs, i.e., Y0, Y1, Y2, and Y3, and two outputs, i.e., A0 and A1. The Y3 has high and Y0 has low priority inputs. When more than one input is '1' at the same time, the output will be the (binary) code corresponding to the higher priority input. Below is the truth table of the 4 to 2 line priority encoder.

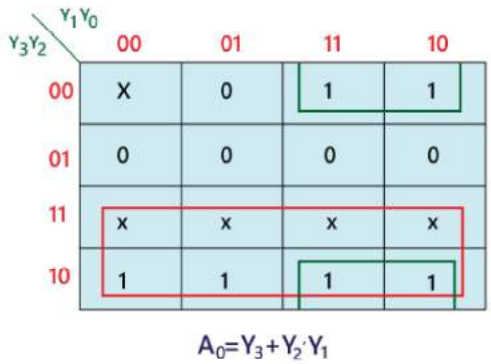
.

Truth Table

INPUTS				OUTPUTS		
Y_3	Y_2	Y_1	Y_0	A_1	A_0	V
0	0	0	0	X	x	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	x	X	1	0	1
1	X	x	X	1	1	1

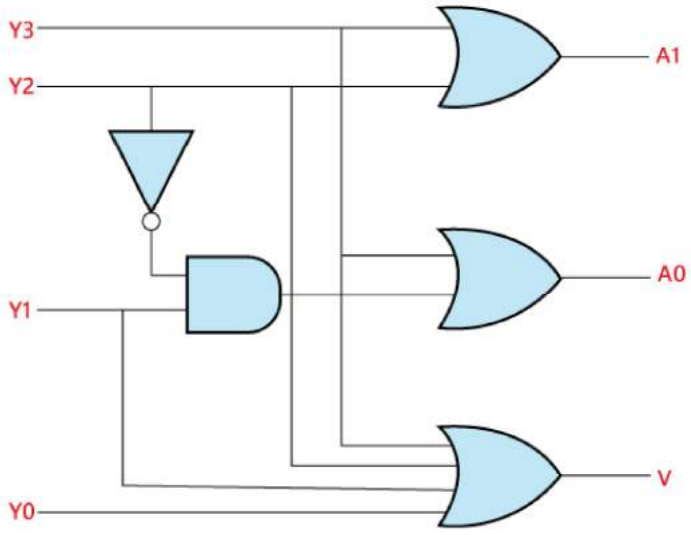
The logical expression of the term A0 and A1 can be found using **K-map** as:





A1=Y3+Y2A0=Y3+Y2'.Y1

Logical circuit of the above expressions is given below:



The above circuit diagram contains two 2-input OR gates, one 4-input OR gate, one 2input AND gate & an inverter. Here AND gate & inverter combination are used for producing a valid code at the outputs, even when multiple inputs are equal to '1' at the same time. Hence, this circuit encodes the four inputs with two bits based on the priority assigned to each input.

Uses of Encoders:

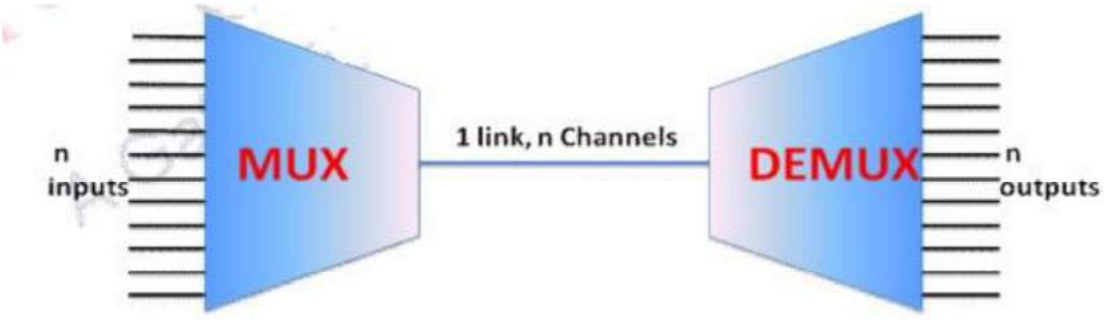
These systems are very easy to use in all digital systems.

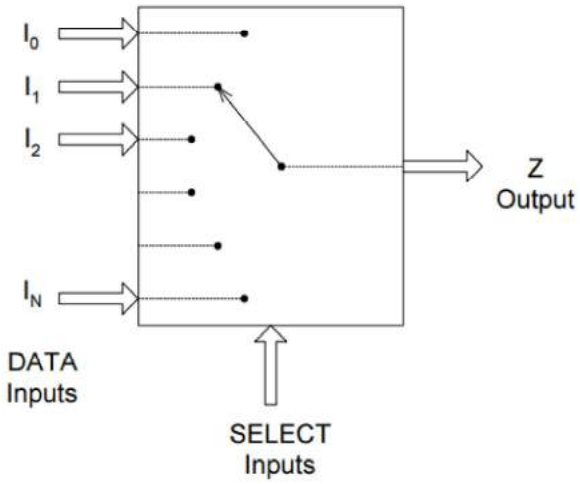
Encoders are used to convert a decimal number into the binary number. The objective is to perform a binary operation such as addition, subtraction, multiplication, etc.

3.11 Multiplexers (Data Selectors)

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. 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.

A digital multiplexer or data selector Multiplexer is a logic circuit that accepts several digital data inputs and selects one of them at a time to pass on to the output. Thus multiplexer transmits several signals on the same line. The purpose of the multiplexer is to save on the number of wires/lines needed to transmit data from multiple sources.





The wide arrow indicates that they may actually be more than one signal line. The multiplexer acts like a rotary switch connecting one of several inputs to a single output. The selection of which input to connect to the output is determined by additional inputs called SELECTOR control lines. For example, output Z will equal data input I0 for some particular SELECT input code, Z will equal I1 for another particular SELECT input code; and so on. Thus, a multiplexer selects 1 out of N input data sources and transmits the selected data to a single output chancel. This is called multiplexing.

Types of multiplexer

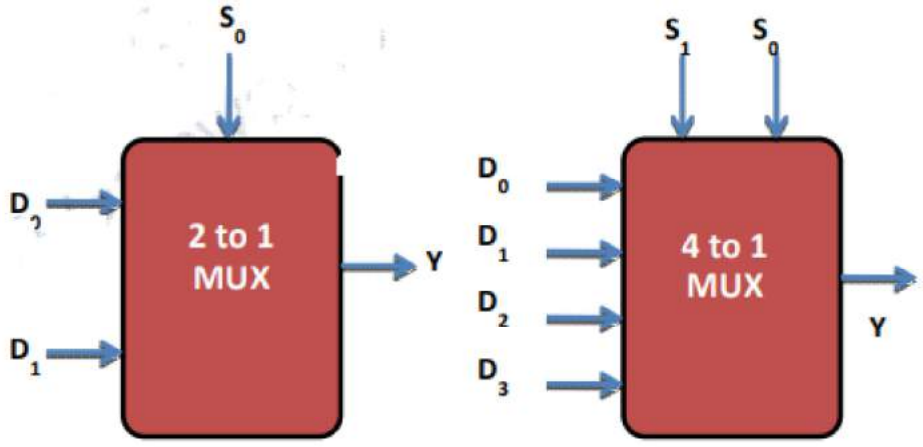


Fig. Logic symbols of 2 to 1 and 4 to 1 multiplexers

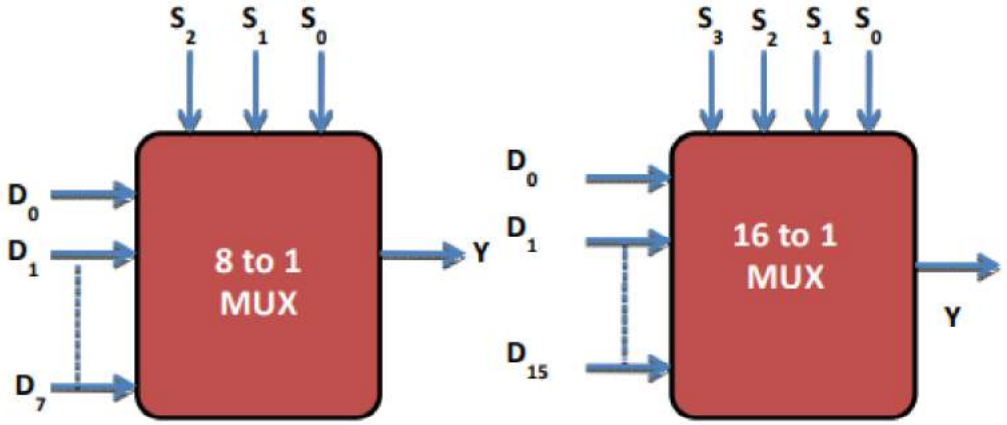
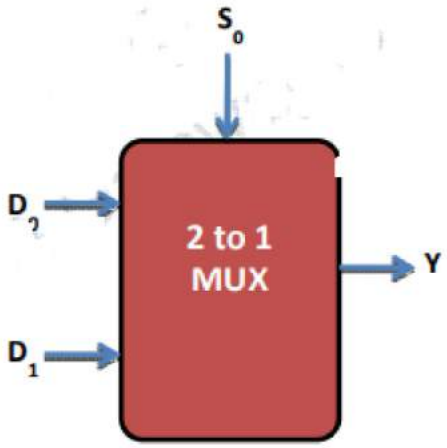


Fig. Logic symbols of 8 to 1 and 16 to 1 multiplexers

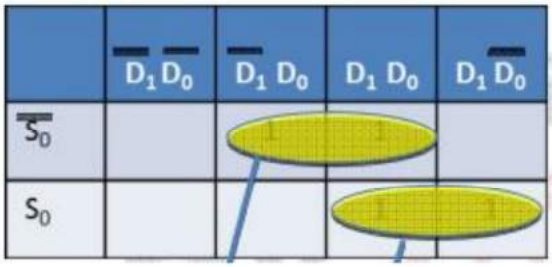
Basic Two-Input Multiplexer



Select input S_0	Output Y
0	D_0
1	D_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

In this multiplexer, D0 and D1 are the data inputs, S0 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 S0 = 0 then the data D0 appears at output Y and when S0=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 S0=0, we have 4 possible combinations for the inputs 00,01,10 and 11. It is observed that Y always follows D0. For S0=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 figure.



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

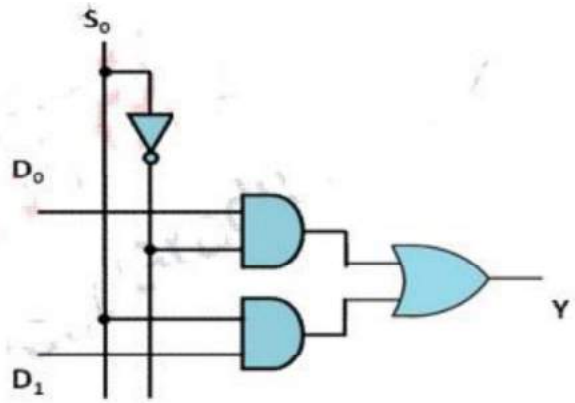


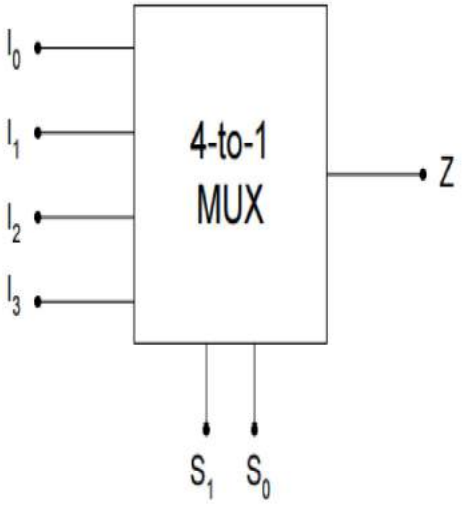
Fig. Simplification using K-map and Logic diagram of 2 to 1 Multiplexer of From the first pair we can eliminate the D1, as it is changing in the pair. Thus this pair contribute S0'D0 in the output. Similarly, in the second group D0 gets eliminated as it is changing in the pair. Therefore, we get S0D1 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 D0 or D1 will appear at the output decided by the value of Y.

The 2 to 1 multiplexer can be implemented using two AND gates and one OR. The Not gate provides complemented and un-complemented form of S0 i.e. select line. The interconnections are made to AND gate to provide the necessary product term.

Four-Input Multiplexer

The second type of multiplexer is 4 to 1 multiplexer. The logic symbol (as shown in figure) indicates that there are 4 data inputs namely D0, D1, D2,D3 and single output (Y) . For 4 data inputs there are two select lines namely S0 and S1. While preparing the function table we write S1 as MSB and S0 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 (S1S0) allows the data from selected input (either from D0,D1,D2,D3) to pass to the output. The output Y receives D0 only when S1=0 and S0=0. Similarly, output Y receives D1 only when S1S0=01. Output Y receives D2 only when S1S0=10. Output Y receives D3 only when S1S0=11.



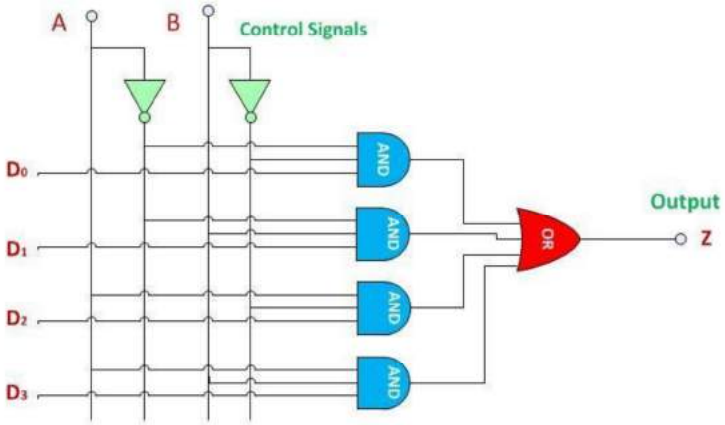
Select inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

The Boolean expression for the output is,

$$Y = S'_1 S'_0 D_0 + S'_1 S_0 D_1 + S_1 S'_0 D_2 + S_1 S_0 D_3$$

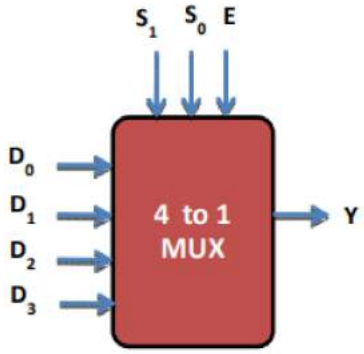
Fig. Logic symbol, Function table and Boolean expression for 4 to multiplexer

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 an 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.



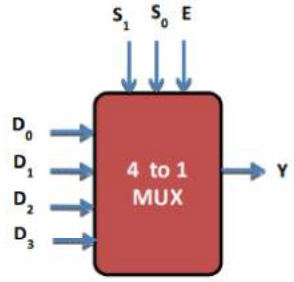
Four – To – One Multiplexer

The first AND gate receives D0, S1′ and S0′ as inputs. Similarly rest of the AND gates receives the appropriate inputs as per the product term. As the data can be selected form 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 require three select inputs to select data from 8 inputs to the 1 output whereas the 16 to 1 multiplexer require 4 select lines.



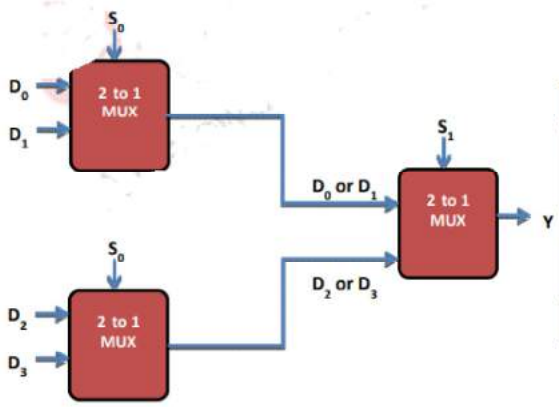
Select Inputs			Output
S_1	S_0	E	Y
X	X	0	X
0	0	1	D_0
0	1	1	D_1
1	0	1	D_2
1	1	1	D_3

The multiplexer will be enables or operative only when the Enable input is active. In this case Enable is active high. When E=0, Multiplexer function is disabled. To enable multiplexer, E must be set to '1'.



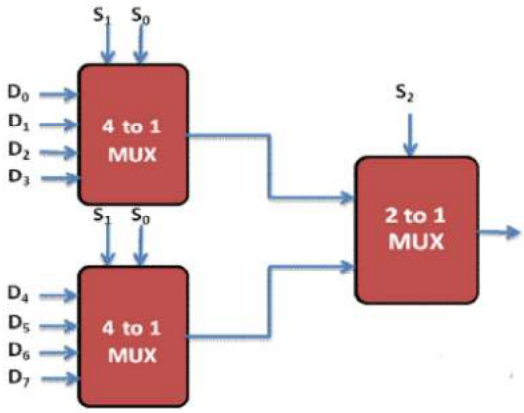
Select Inputs			Output
S_1	S_0	E	Y
X	X	0	X
0	0	1	D_0
0	1	1	D_1
1	0	1	D_2
1	1	1	D_3

Construction of 4 to 1 multiplexer using two 2 to 1 multiplexer:



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

Construction of 8 to 1 multiplexer 4 to 1 and 2 to 1 multiplexer:



Select inputs			Output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

3.12 Demultiplexer

Demultiplexer has a single input and n output lines. Demultiplexer can be visualized as reverse multi-position switch. The select lines permit input data from single line to be switched to any one of the many output lines.

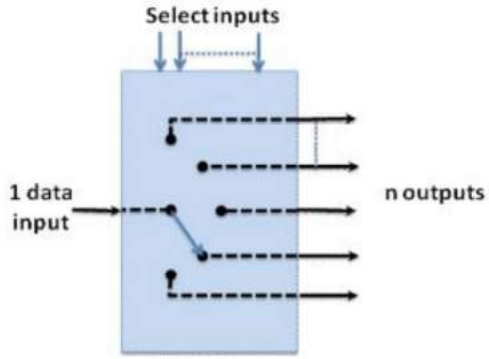


Fig. Multi-position switch as Demultiplexer

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 below figure. The mathematical relation between select lines and 'n' output are: $2m = n$

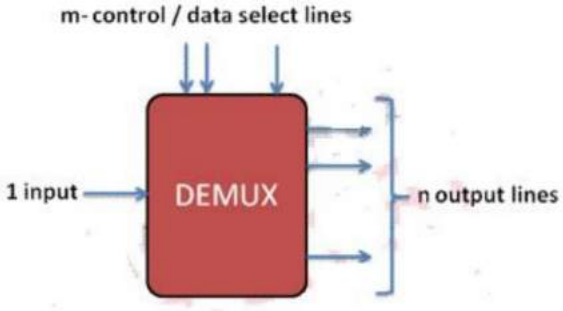


Fig. Logic symbol of basic demultiplexer

As a demultiplexer takes data from one input line and distributes over a 2m output line, hence it is often referred to as 1 to 2m 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 figure. Number of select lines decides this classification.

.

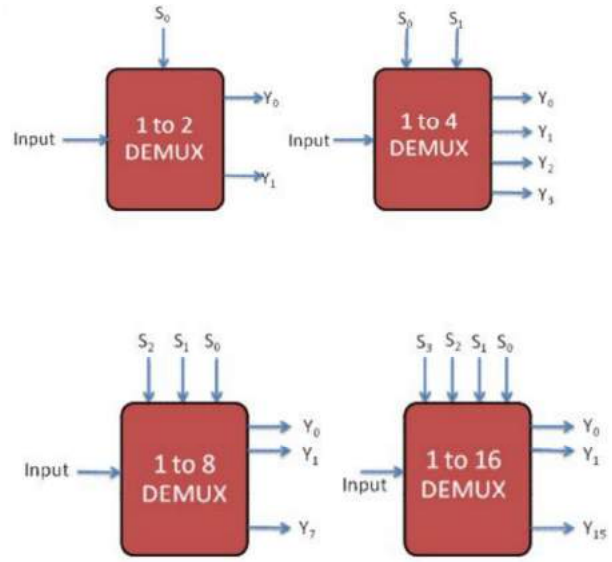
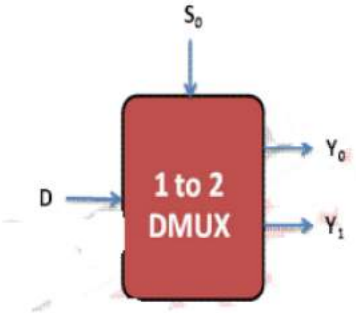


Fig. Types of Demultiplexers

A 1 to 2 demultiplexer



Select input	Outputs	
S0	Y0	Y1
0	D	0
1	0	D

Fig. Logic symbol and function table of a 1 to 2 demultiplexer

As a demultiplexer takes data from one input line and distributes over a 2m output line, hence it is often referred to as 1 to 2m 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 figure. Number of select lines decides this classification.

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

$$Y_1 = S_0\, D$$

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

.

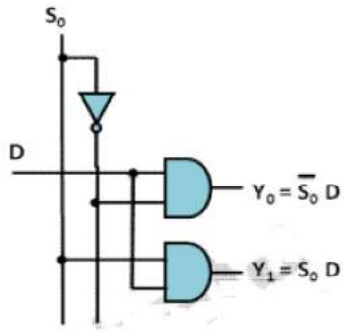
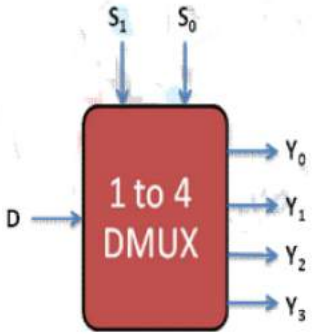


Fig. Logic diagram for 1 to 2 demultiplexer

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

Fig. Logic symbol and function table of a 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 S1S0=00, the Y0 output of demultiplexer receive the input data. For S1S0=01, the output Y1 receives the input data. With S1S0=10, the input data is distributed to Y2 and when S1S0=11, the output Y3 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 figure. The product term for the output decides the interconnections between the gates data input and select lines

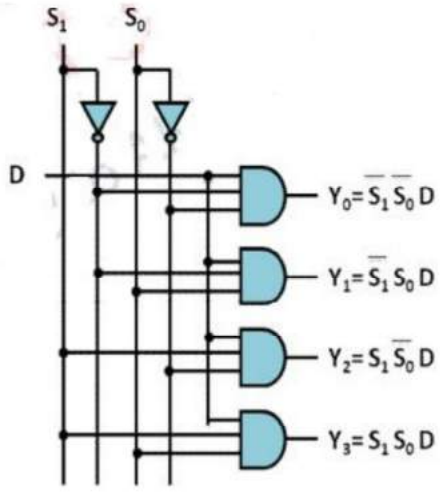


Fig. Logic diagram for 1 to 4 demultiplexer