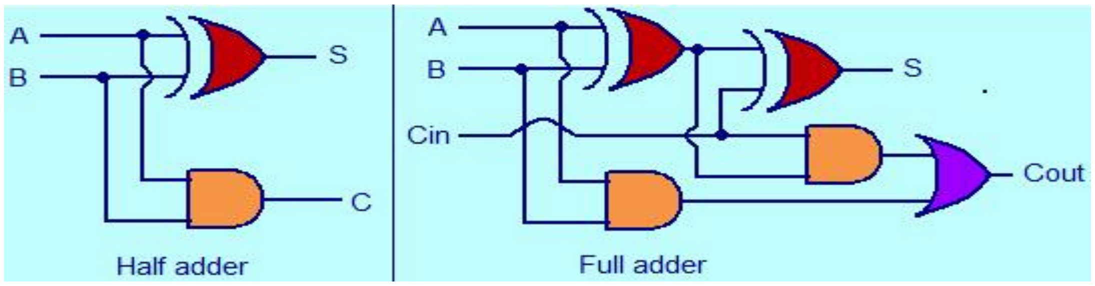
Half Adder and Full Adder

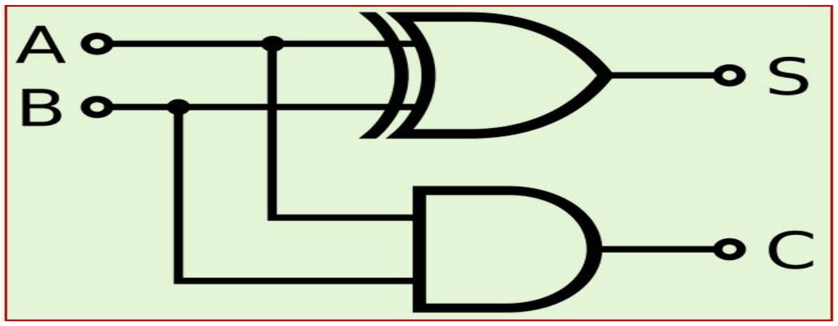
- An adder is a digital circuit that performs addition of numbers.
- The half adder adds two binary digits called as augend and addend and produces two outputs as sum and carry; XOR is applied to both inputs to produce sum and AND gate is applied to both inputs to produce carry.
- The full adder adds 3 one bit numbers, where two can be referred to as operands and one can be referred to as bit carried in. And produces 2-bit output, and these can be referred to as output carry and sum.

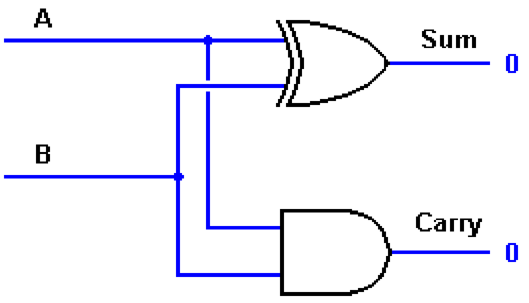


==

Half Adder

With the help of half adder, we can design circuits that are capable of performing simple addition with the help of logic gates.

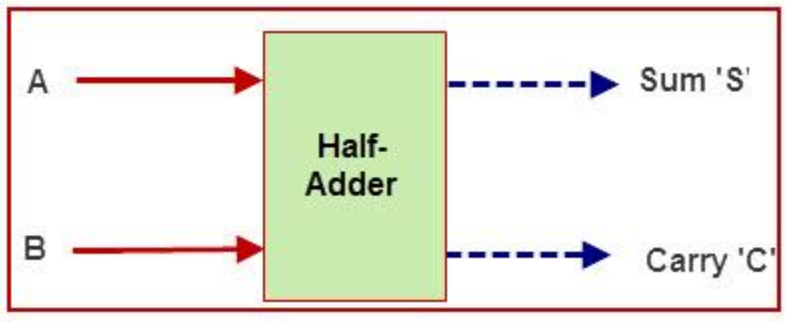




SUM= $A \oplus B$ CARRY=A.B

SUM = A XOR B = A $\oplus$ B

CARRY = A AND B = A.B



Let us first take a look at the addition of single bits.

0+0 = 0

0+1 = 1

1+0 = 1

1+1 = 10

These are the least possible single-bit combinations. But the result for 1+1 is 10. Though this problem can be solved with the help of an EXOR Gate, if you do care about the output, the sum result must be re-written as a 2-bit output.

Thus the above equations can be written as

0+0 = 00

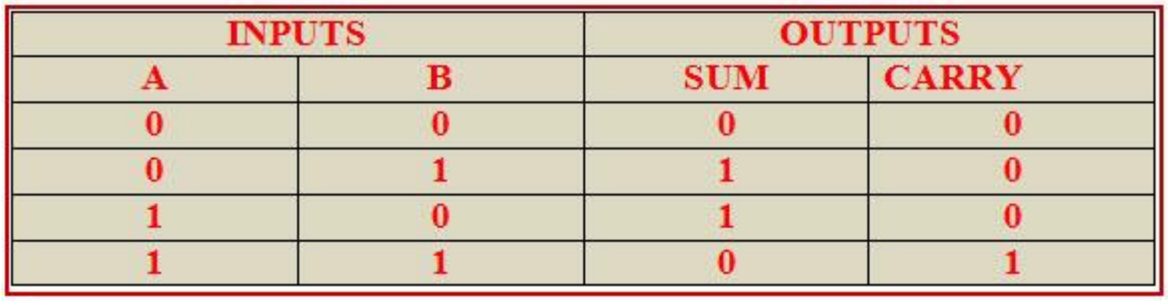
0+1 = 01

1+0 = 01

1+1 = 10

Here the output ‘1’of ‘10’ becomes the carry-out. The result is shown in a truth-table below. ‘SUM’ is the normal output and ‘CARRY’ is the carry-out.

Half Adder Truth Table



INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

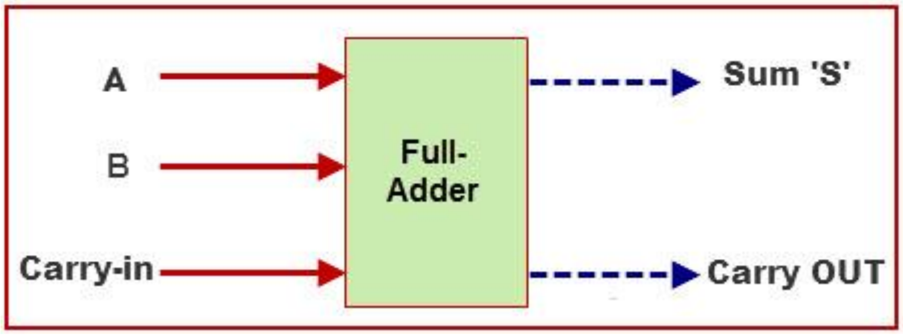
From the equation it is clear that this 1-bit adder can be easily implemented with the help of EXOR Gate for the output 'SUM' and an AND Gate for the carry. Take a look at the implementation below.

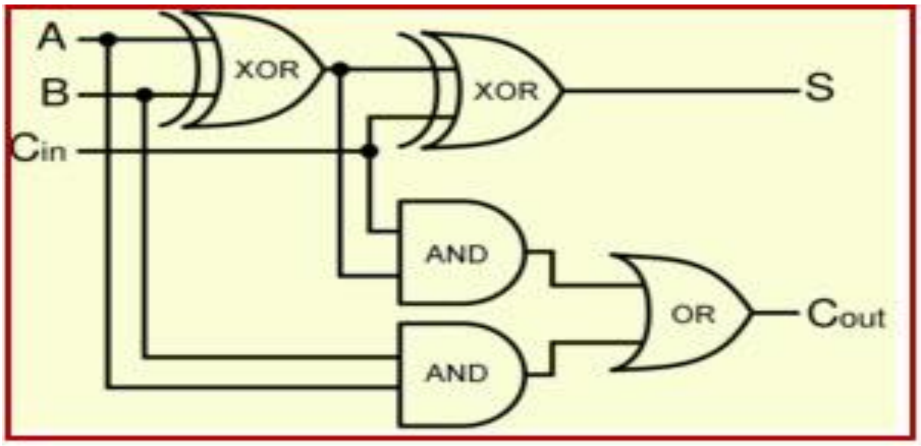
==

Full Adder

The main difference between a half-adder and a full-adder is that the full-adder has three inputs and two outputs.

The first two inputs are A and B and the third input is an input carry designated as C_{IN}. When a full adder logic is designed we will be able to string eight of them together to create a byte-wide adder and cascade the carry bit from one adder to the next.

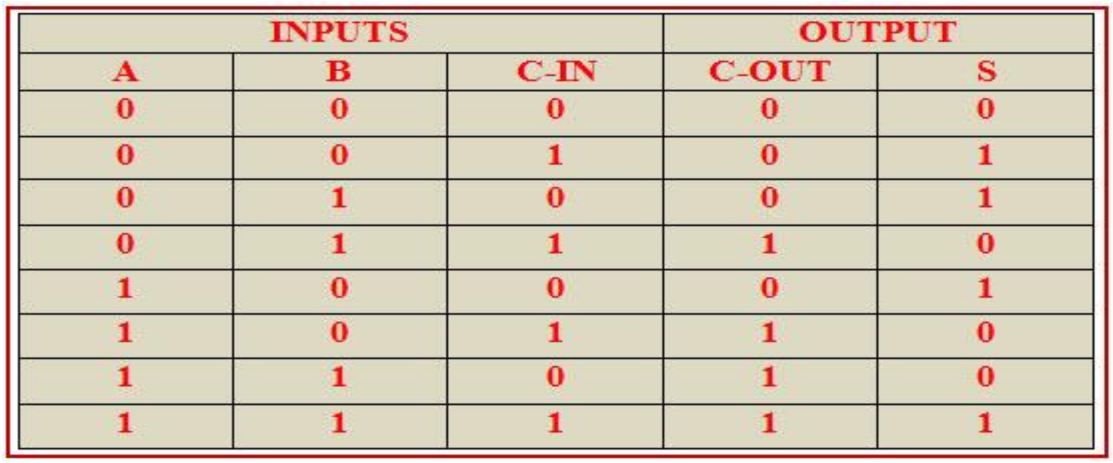




SUM = (A XOR B) XOR Cin = $(A \oplus B) \oplus Cin$

CARRY-OUT = A AND B OR Cin(A XOR B) = $A.B + Cin(A \oplus B)$

Full Adder Truth Table:

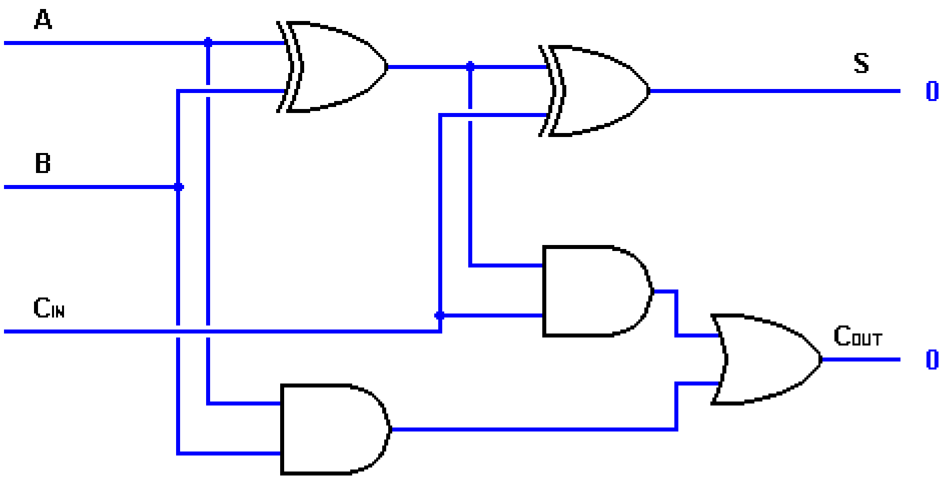


INPUTS			OUTPUT	
A	B	C-IN	C-OUT	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The output carry is designated as C_{OUT} and the normal output is designated as S. Take a look at the truth-table.

From the above truth-table, the full adder logic can be implemented. We can see that the output S is an EXOR between the input A and the half-adder SUM output with B and C_{IN} inputs. We must also note that the C_{OUT} will only be true if any of the two inputs out of the three are HIGH.

Thus, we can implement a full adder circuit with the help of two half adder circuits. The first will half adder will be used to add A and B to produce a partial Sum. The second half adder logic can be used to add CIN to the Sum produced by the first half adder to get the final S output. If any of the half adder logic produces a carry, there will be an output carry. Thus, C_{OUT} will be an OR function of the half-adder Carry outputs. Take a look at the implementation of the full adder circuit shown below.



==

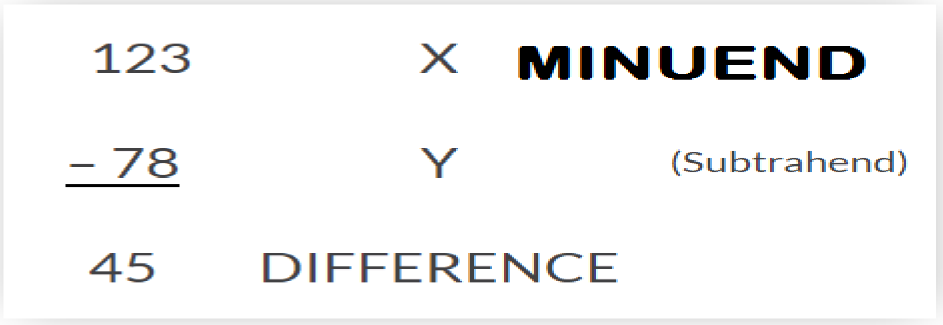
Binary Subtractor

The Binary Subtractor is another type of combinational arithmetic circuit that produces an output which is the subtraction of two binary numbers.

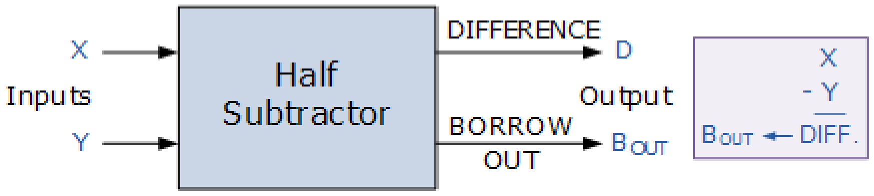
A Half Subtractor Circuit

A half subtractor is a logical circuit that performs a subtraction operation on two binary digits. The half subtractor produces a sum and a borrow bit for the next stage. As like addition operation of 2 binary digits, which produces SUM and CARRY, the subtraction of 2 binary digits also produces two outputs which are termed as **difference and borrow**.

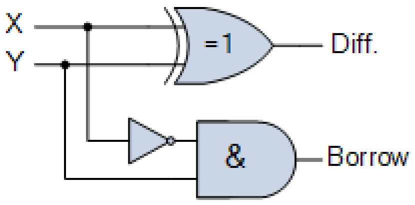
Here, the binary digit from which the other digit is subtracted is called minuend and the binary digit which is to be subtracted is known as the subtrahend.



Half Subtractor Block Diagram



Half subtractor circuit diagram :



Half subtractor truth table :

Inputs		Outputs	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

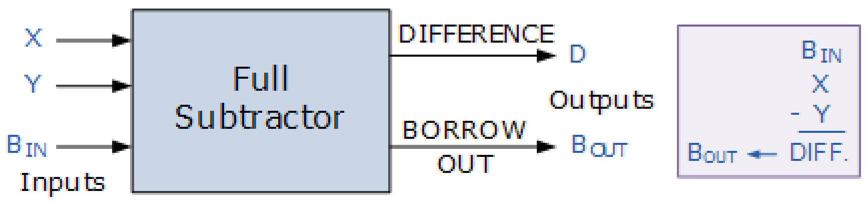
$D = (X'Y + XY') = X \oplus Y$
$B = X'Y$

--

A Full Binary Subtractor Circuit

The main difference between the Full Subtractor and the previous Half Subtractor circuit is that a full subtractor has three inputs. The two single bit data inputs X (minuend) and Y (subtrahend) the same as before plus an additional Borrow-in (B-in) input to receive the borrow generated by the subtraction process from a previous stage as shown below.

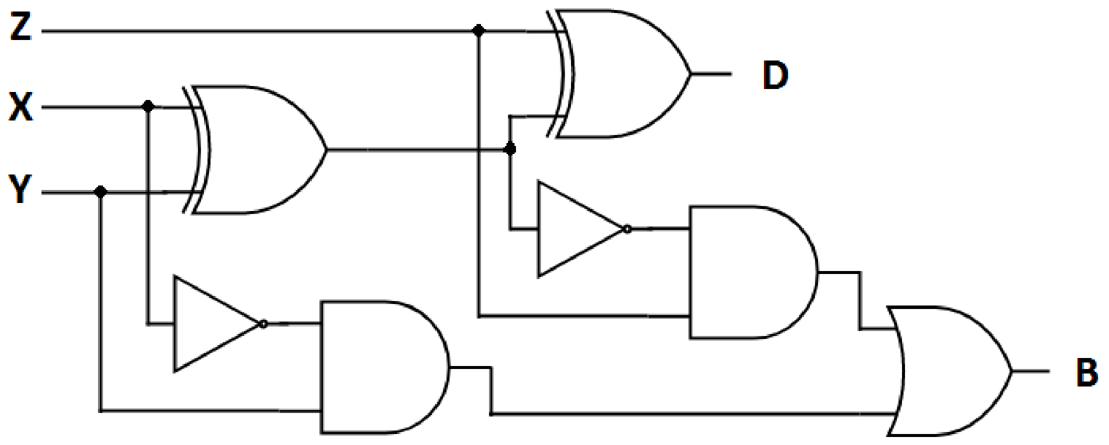
Full Subtractor Block Diagram



Then the combinational circuit of a “full subtractor” performs the operation of subtraction on three binary bits producing outputs for the difference D and borrow B-out. Just like the binary adder circuit, the full subtractor can also be thought of as two half subtractors connected together, with the first half subtractor passing its borrow to the second half subtractor as follows.

As the full subtractor circuit above represents two half subtractors cascaded together, the truth table for the full subtractor will have eight different input combinations as there are three input variables, the data bits and the Borrow-in, BIN input. Also includes the difference output, D and the Borrow-out, BOUT bit.

Full Subtractor Circuit Diagram



Full subtractor truth table :

Inputs			Outputs	
X	Y	Z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

The full subtractor boolean expressions are :

DIFF= (X'Y'Z + X'YZ' + XY'Z' + XYZ) = X ⊕ Y ⊕ Z

BORROW=(X'Y'Z + X'YZ' + X'YZ + XYZ) = X'(Y ⊕ Z) + YZ

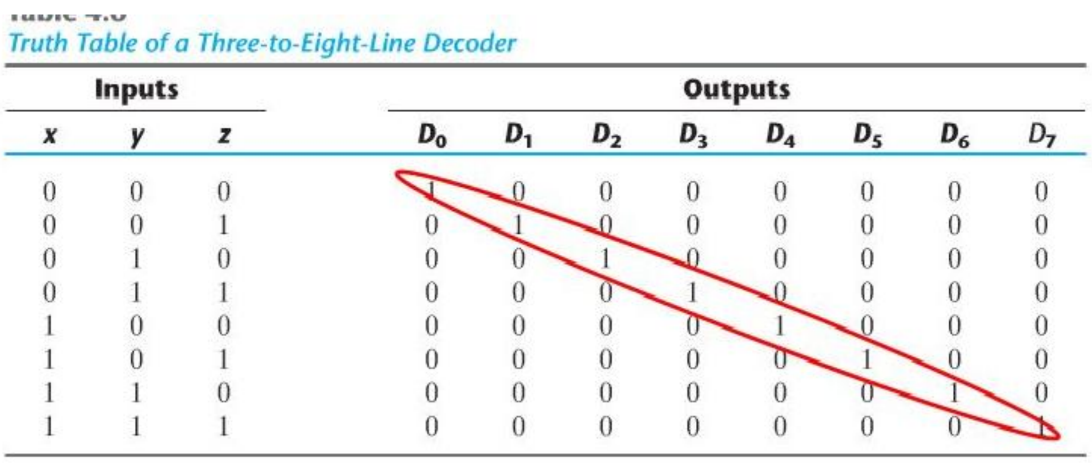
DECODERS

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n-bit decoded information has unused or don't-care combinations, the decoder output will have fewer than 2^noutputs.

The decoders presented here are called **n-to-m-line decoders**, where $m \leq 2^n$. Their purpose is to generate the 2^n (or fewer) minterms of n input variables.

As an example, consider the 3-to-8-line decoder circuit of Fig. 5-8. The three inputs are decoded into eight outputs, each output representing one of the minterms of the 3- input variables. The three inverters provide the complement of the inputs, and each one of the eight AND gates generates one of the minterms. A particular application of this decoder would be a binary-to-octal conversion. The input variables may represent a binary number, and the outputs will then represent the eight digits in the octal number.

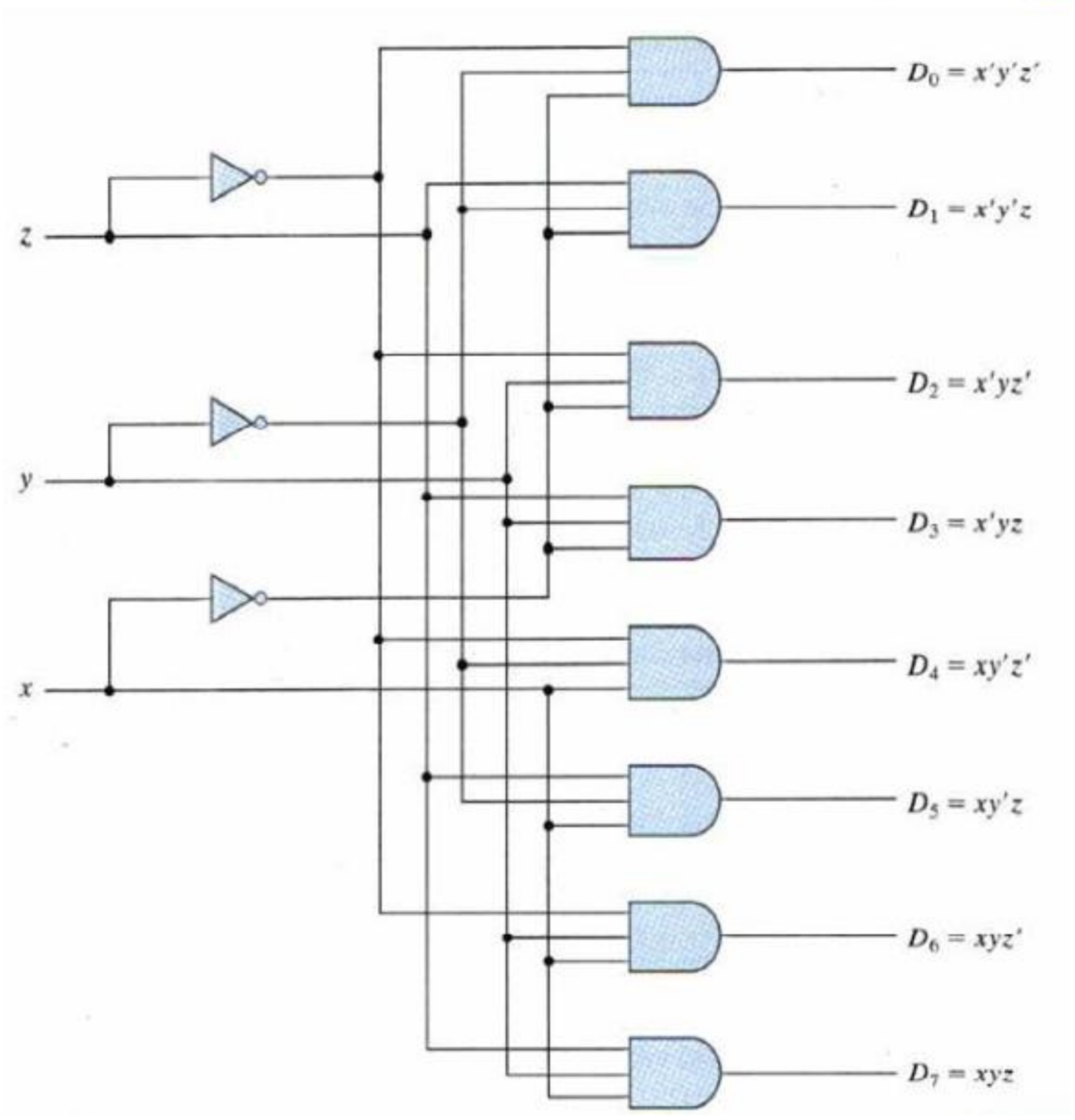
Table 4.6
Truth Table of a Three-to-Eight-Line Decoder



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

Convert binary information from *n* input lines to 2^n unique output lines.

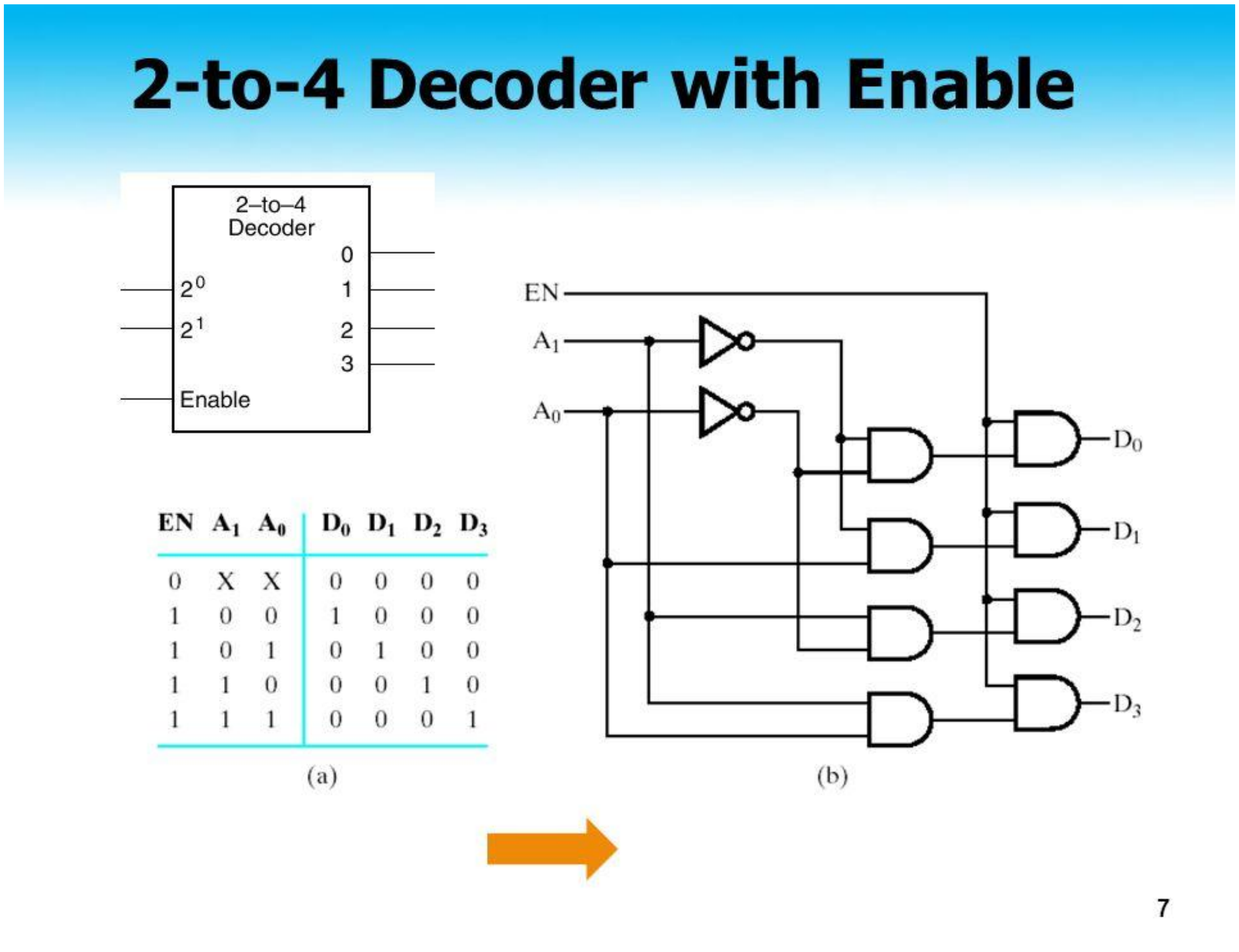
This particular circuit take a binary number and convert it to an octal number.



A 2-to-4-line decoder

A 2-to-4-line decoder with an enable input constructed with NAND gates is shown in Fig. 5- 10. All outputs are equal to I if enable input E is 1, regardless of the values of inputs A and B. When the enable input is 0, the circuit operates as a decoder with complemented outputs. The truth table lists these conditions. The X's under A and B are don't-care conditions. Normal decoder operation occurs only with E = 0, and the outputs are selected when they are in the 0 state.

The block diagram of the decoder is shown in Fig. 5-1 I(a). The small circle at input E indicates that the decoder is enabled when E = 0. The small circles at the outputs indicate that all outputs are complemented.


2-to-4 Decoder with Enable

EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(a) (b)

7

ENCODERS

An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has $2^{\mathbf{n}}$ (or fewer) input lines and n output lines. The output lines generate the binary code corresponding to the input value. An example of an encoder is the octal to- binary encoder whose truth table is given in Table 5-3. It has eight inputs, one for each of the octal digits, and three outputs that generate the corresponding binary number.

It is assumed that only one input has a value of 1 at any given time; otherwise the circuit has no meaning.

The encoder can be implemented with OR gates whose inputs are determined directly from the truth table.

- **Output z is equal to 1** when the input octal digit is **1 or 3 or 5 or 7.**
- **Output y is 1** for octal digits **2, 3, 6, or 7**
- **Output x is 1** for digits **4, 5, 6, or 7.**

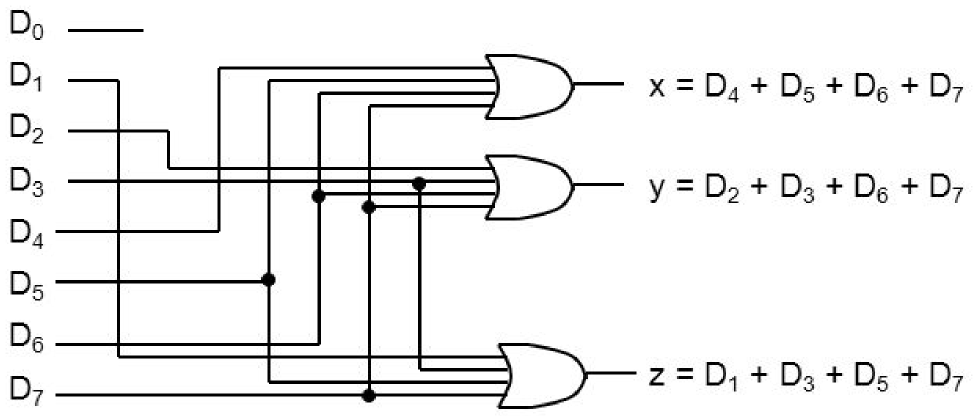
These conditions can be expressed by the following output Boolean functions:

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

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

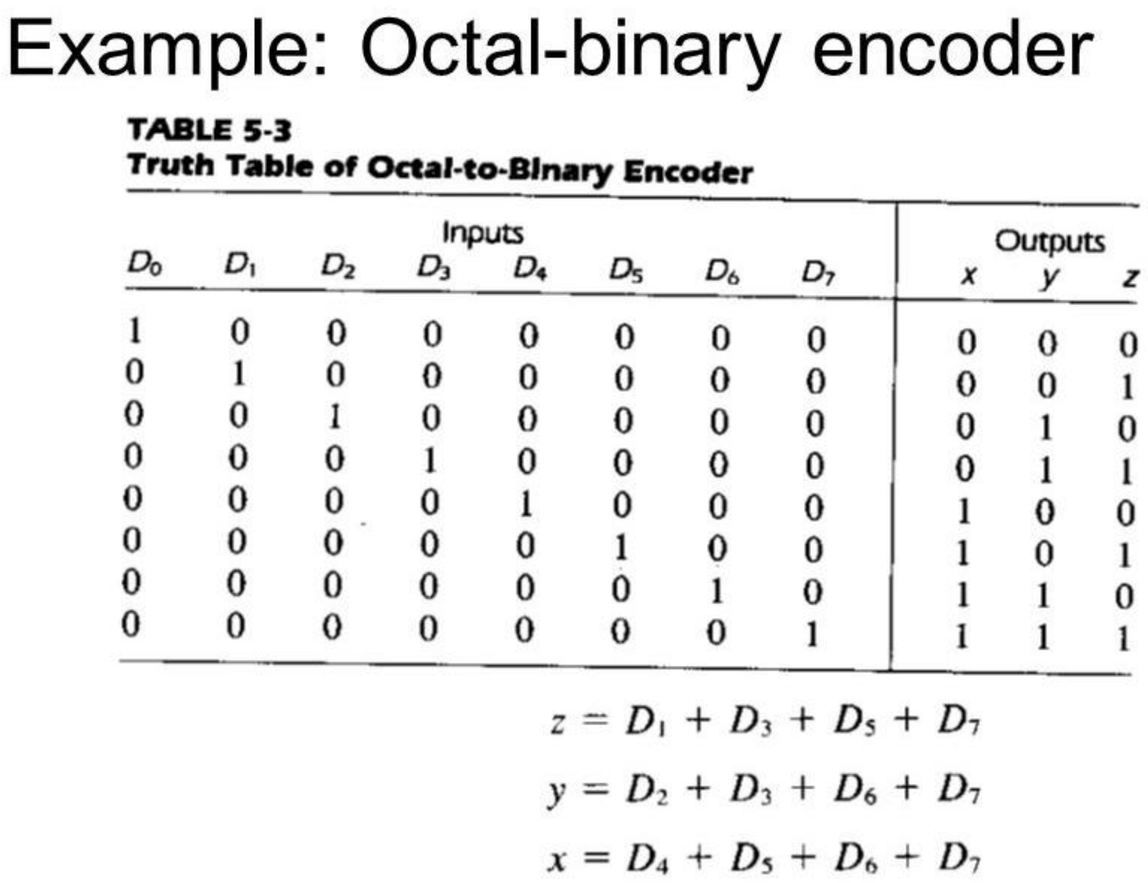
$$x = D_4 + D_5 + D_6 + D_7$$

- Example: Octal-to-binary encoder.



D_0
D_1
D_2
D_3
D_4
D_5
D_6
D_7
$x = D_4 + D_5 + D_6 + D_7$
$y = D_2 + D_3 + D_6 + D_7$
$z = D_1 + D_3 + D_5 + D_7$

8-to-3 encoder



Example: Octal-binary encoder

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

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

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

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

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

===

Priority Encoder

A priority encoder is an encoder circuit that includes the priority function. The operation of the priority encoder is such that if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence. The truth table of a four-input priority encoder is given in Table 5-4. The X's are don't-care conditions that designate the fact that the binary value may be equal either to 0 or 1.

Input D, has the highest priority; so regardless of the values of the other inputs, when this input is 1, the output for xy is 11 (binary 3).

D2 has the next priority level. The output is 10 if D, = 1 provided that D, = 0, regardless of the values of the other two lower-priority inputs.

The output for D, is generated only if higher-priority inputs are 0, and so on down the priority level. A valid-output indicator, designated by V, is set to 1 only when one or more of the inputs are equal to 1. If all inputs are 0, V is equal to 0, and the other two outputs of the circuit are not used.

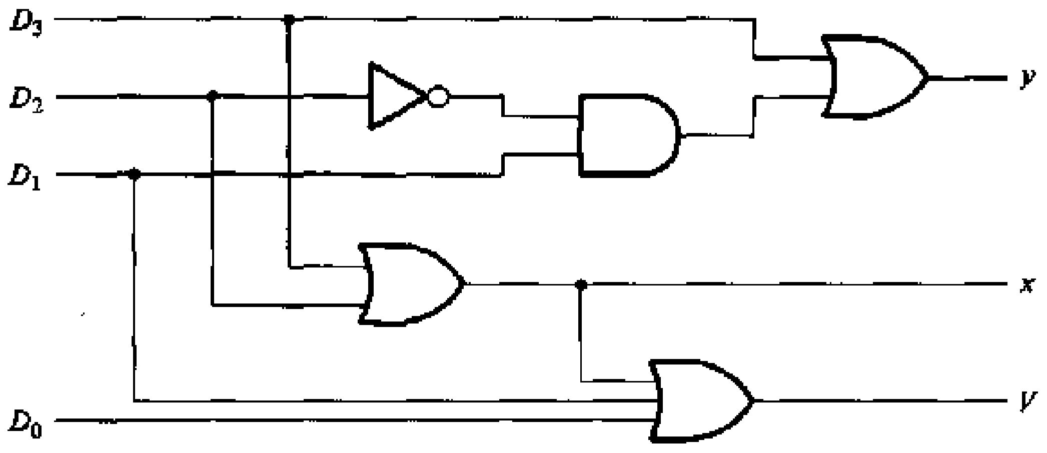


FIGURE 5-15
4-input priority encoder

TABLE 5-4
Truth Table of a Priority Encoder

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

<u>MULTIPLEXERS (DATA SELECTOR)</u>

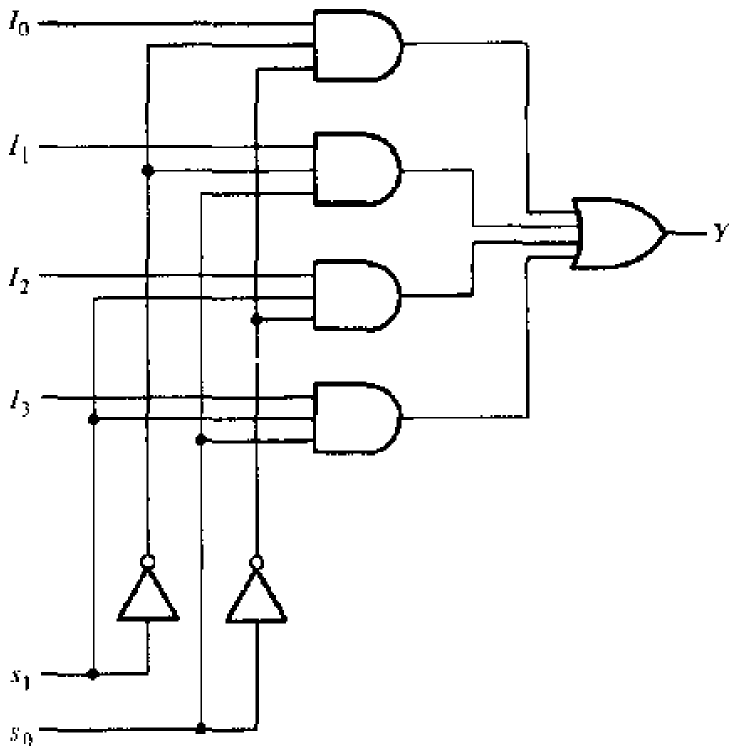
- **A digital multiplexer (data selector) is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines.**
- Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.

A 4-to-l-line multiplexer is shown in Fig. 5-16. Each of the four input lines, **I_0 to I_3** is applied to one input of an AND gate.

Selection lines **S_1 and S_0** are decoded to select a particular AND gate. To demonstrate the circuit operation, consider the case

- When S_1S_0 = 00. The AND gate associated with input I_0, is connected to Y.
- When S_1S_0 = 01. The AND gate associated with input I_1, is connected to Y.
- When S_1S_0 = 10. The AND gate associated with input I_2, is connected to Y.
- When S_1S_0 = 11. The AND gate associated with input I_3, is connected to Y.

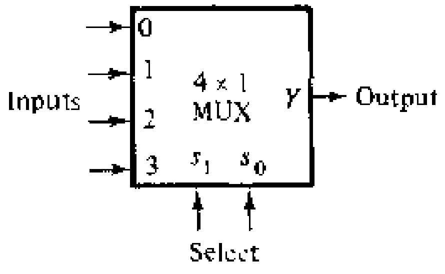
A multiplexer is often abbreviated as MUX.



(a) Logic diagram

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

(b) Function table

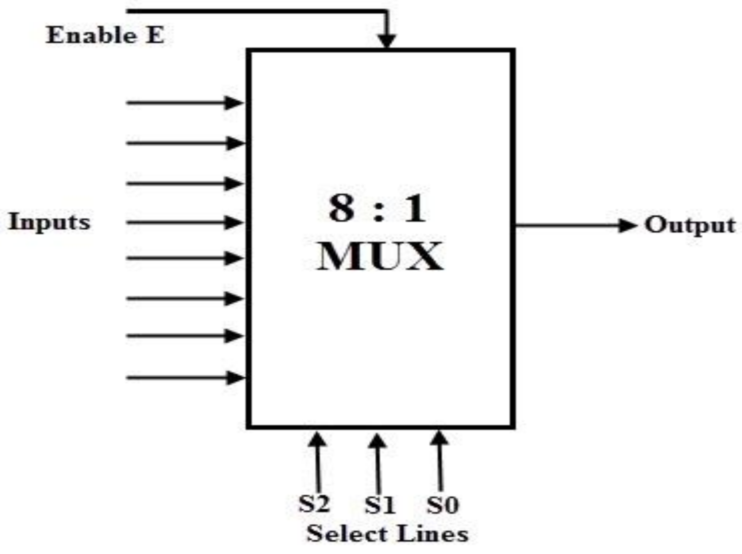


(c) Block diagram

<u>8-to-1 Multiplexer</u>

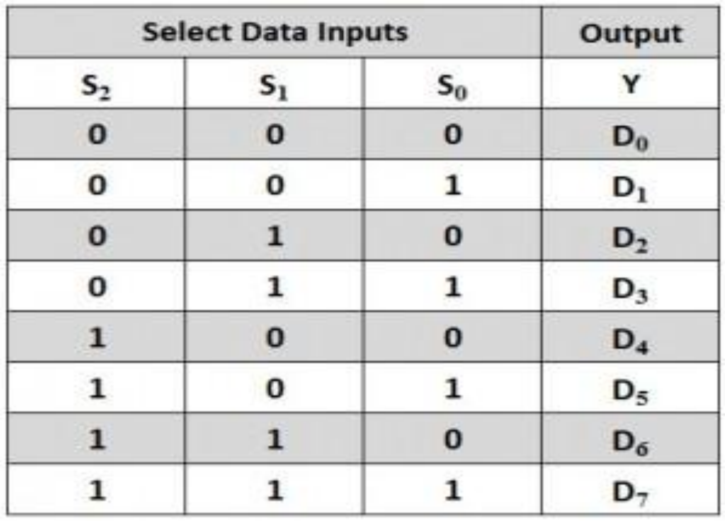
An 8-to-1 multiplexer consists of eight data inputs D0 through D7, three input select lines S2 through S0 and a single output line Y. Depending on the select lines combinations, multiplexer decodes the inputs.

The below figure shows the block diagram of an 8-to-1 multiplexer with enable input that enable or disable the multiplexer. Since the number data bits given to the MUX are eight then 3 bits (23=8) are needed to select one of the eight data bits.



The truth table for an 8-to1 multiplexer is given below with eight combinations of inputs so as to generate each output corresponds to input.

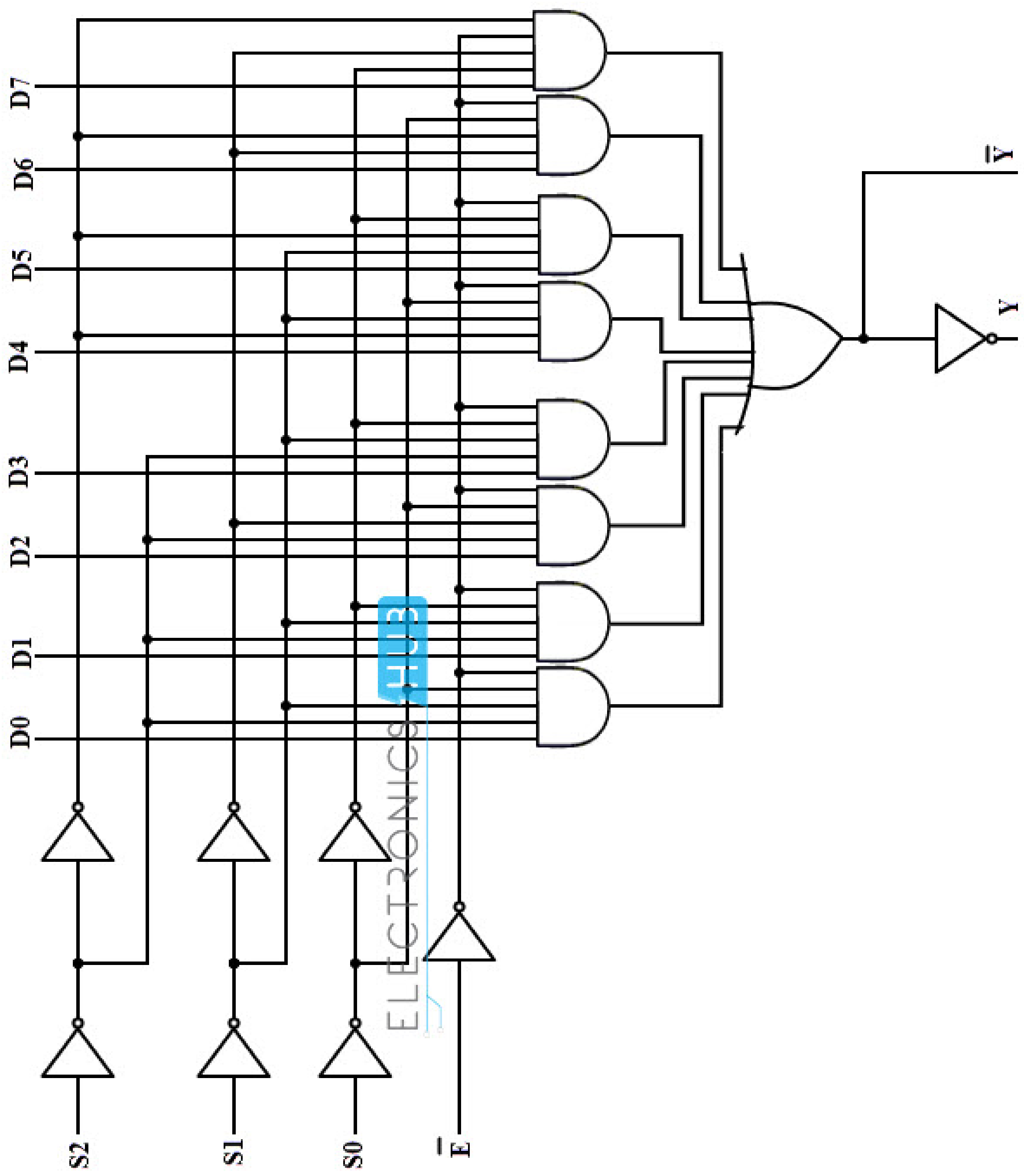
For example, if S2= 0, S1=1 and S0=0 then the data output Y is equal to D2. Similarly the data outputs D0 to D7 will be selected through the combinations of S2, S1 and S0 as shown in below figure.



Select Data 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

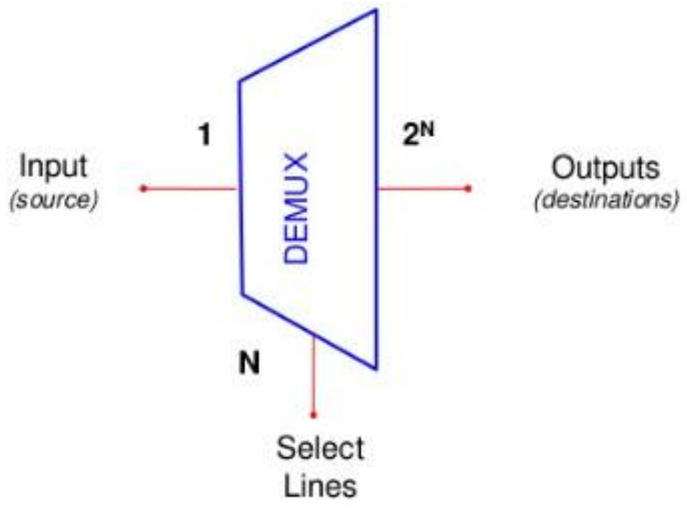
From the above truth table, the Boolean equation for the output is given as

$$Y = D0\,\overline{S2}\,\overline{S1}\,\overline{S0} + D1\,\overline{S2}\,\overline{S1}\,S0 + D2\,\overline{S2}\,S1\,\overline{S0} + D3\,\overline{S2}\,S1\,S0 + D4\,S2\,\overline{S1}\,\overline{S0} + D5\,S2\,\overline{S1}\,S0 + D6\,S2\,S1\,\overline{S0} + D7\,S2\,S1\,S0$$



DEMULTIPLEXERS (data distributor)

A Demultiplexer is a data distributor read as DEMUX. It is quite opposite to multiplexer or MUX. It is a process of taking information from one input and transmitting over one of many outputs.

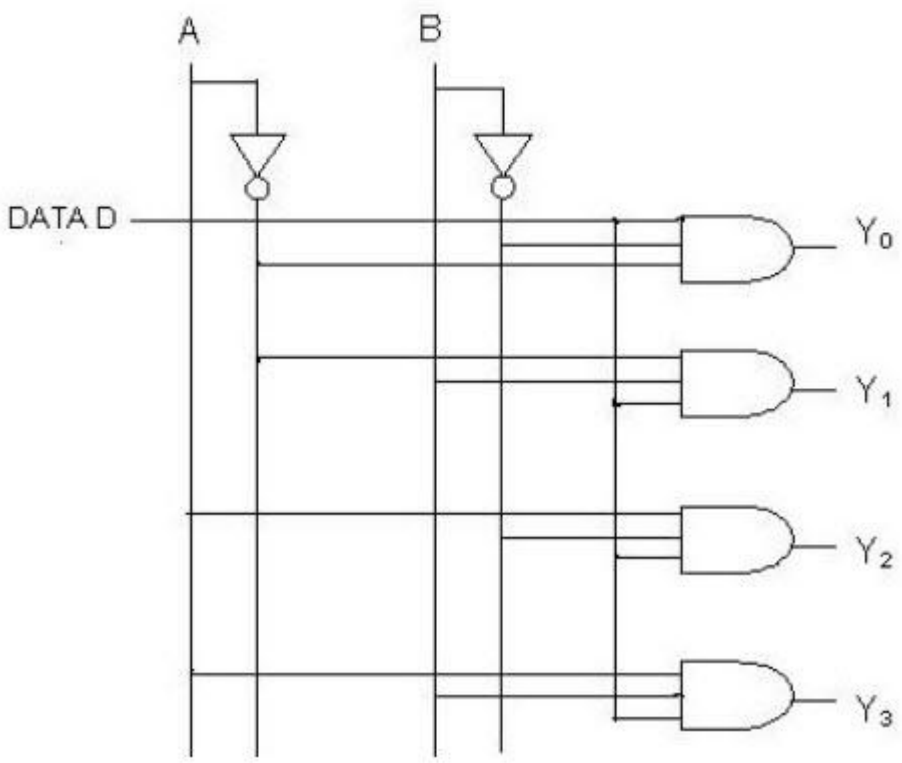


DEMUX are used to implement general-purpose logic systems. A demultiplexer takes one single input data line and distributes it to any one of a number of individual output lines one at a time. Demultiplexing is the process of converting a signal containing multiple analog or digital signals backs into the original and separate signals. A demultiplexer of 2^n outputs has n select lines.

Types of Demultiplexers

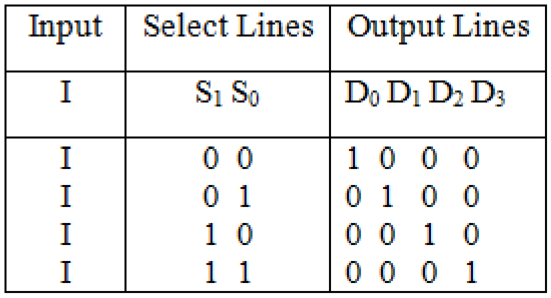
1 to 4 Demultiplexer

The 1 to 4 demultiplexer consists of one input, four outputs, and two control lines to make selections The below diagram shows the circuit of 1 to 4 demultiplexer.



Truth Table

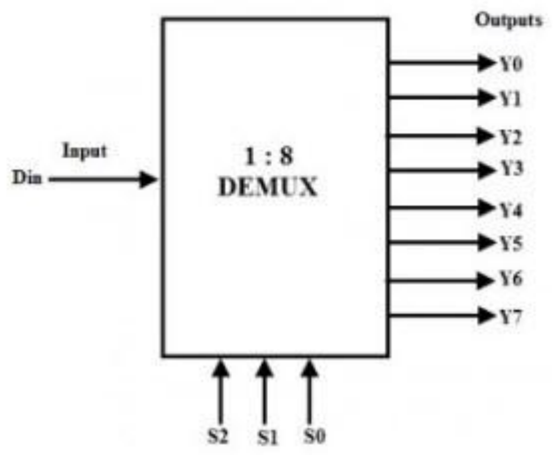
The below is the truth table for the 1 to 4demultiplexer.



Input	Select Lines	Output Lines
I	S_1 S_0	D_0 D_1 D_2 D_3
I	0 0	1 0 0 0
I	0 1	0 1 0 0
I	1 0	0 0 1 0
I	1 1	0 0 0 1

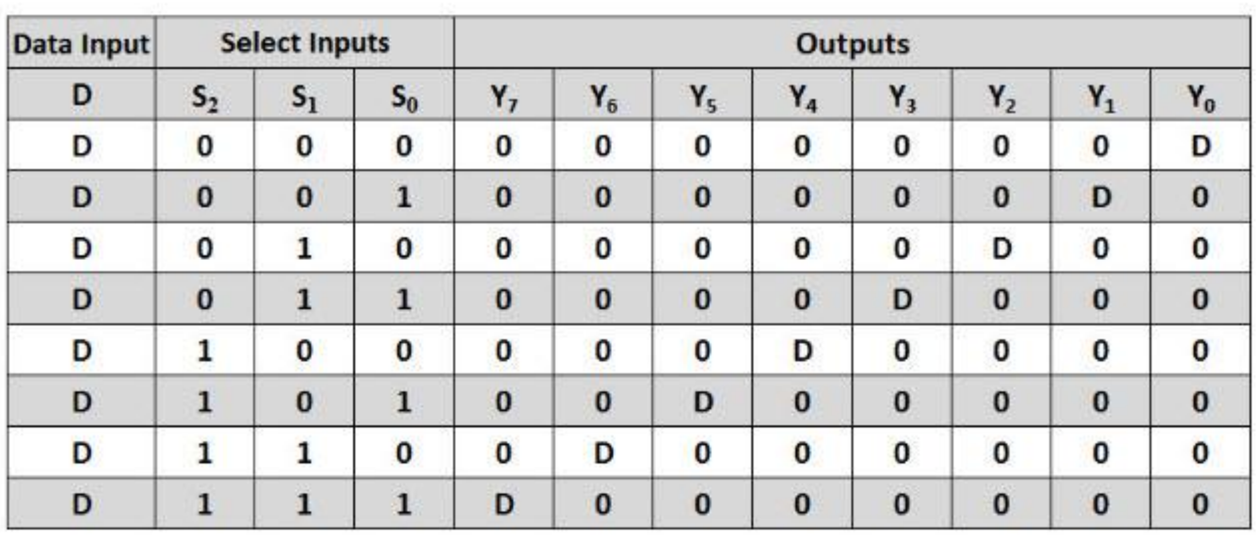
1 to 8 Demultiplexer

A 1 to 8 demultiplexer consists of one input line, 8 output lines and 3 select lines. Let the input be D, S1 and S2 are two select lines and eight outputs from Y0 to Y7. It is also called as 3 to 8 demux because of the 3 selection lines. Below is the block diagram of 1 to 8 demux.



Truth Table

The below is the truth table for 1 to 8 demultiplexer. It tells the functionality of the demux, like, if S1S2S0=000, then the output is seen at Y0 and so on.



Data Input	Select Inputs			Outputs							
D	S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	0	0	0	0	0	D
D	0	0	1	0	0	0	0	0	0	D	0
D	0	1	0	0	0	0	0	0	D	0	0
D	0	1	1	0	0	0	0	D	0	0	0
D	1	0	0	0	0	0	D	0	0	0	0
D	1	0	1	0	0	D	0	0	0	0	0
D	1	1	0	0	D	0	0	0	0	0	0
D	1	1	1	D	0	0	0	0	0	0	0