Nagarathna K
SGL, E&C Dept
Subject Code: 15EC32T

DIGITAL ELECTRONICS

Unit 1 - Combinational Logic Circuits

Combinational logic circuit is a digital circuit which is implemented by a combination of several logic gates whose output at any time depends on present inputs only.

For n variables there are 2^npossible binary input combinations. For each binary combination there is one possible output. Combinational logic circuit has no memory hence its output depends only on present inputs.

Applications of Combinational Logic Circuits

- Multiplexers
- Demultiplexers
- Encoders
- Decoders
- Priority encoders
- Adders and Subtractors
- Code Convertors

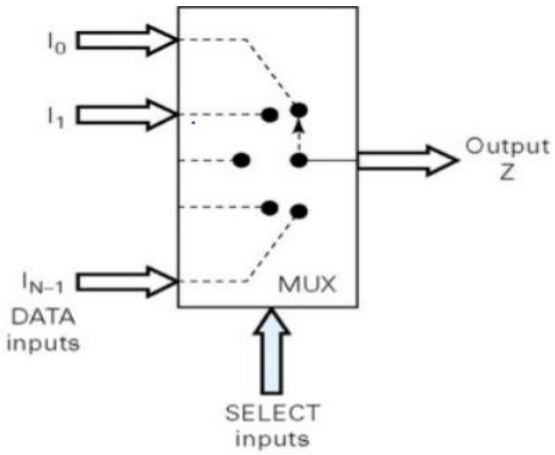
Multiplexers

Definition: Multiplexer is a combinational logic circuit that can select one out of many inputs and pass it on to output.

The term Multiplex means “many into one”. Multiplexing is the process of transmitting a large number of information over a single line

It has many inputs and one output. Multiplexer is also called as data selector. Multiplexer is also called as digital switch.

Functional block diagram of mux:



It has n input lines, some m select lines, one output line. If there are n data input lines and m select lines then $2^m = n$. Example, if $m = 3$ then data input lines will be 8.

Types Of Multiplexers

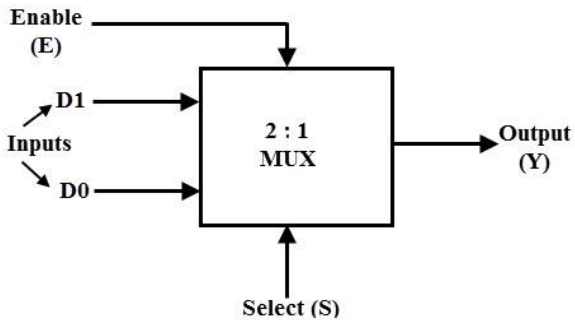
- 2:1 mux (1 select input)
- 4:1mux (2 select input)
- 8:1mux (3 select input)
- 16:1mux (4 select input)
- 32:1 mux (5 select input

Applications of Mux

- Data selection
- Data routing
- Parallel to serial conversion
- Logic function generation
- Operation Sequencing

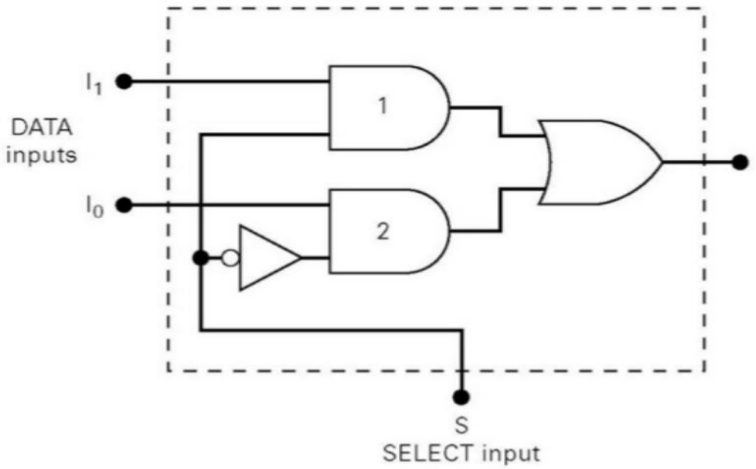
2:1 Mux

A 2:1 mux has 2 data inputs D_0 and D_1 and 1 select input line S_0 and 1 output Y.



Select i/p	o/p Y
0	D_0
1	D_1

Logic Diagram Of 2:1 MUX



A 2:1 mux consists of 2 data inputs D_0 and D_1 single output Y and 1 select input S_0. The select input S_0 selects one out of two inputs and passes it to the output.

Figure shows 2:1 mux symbol, truth table and logic diagram

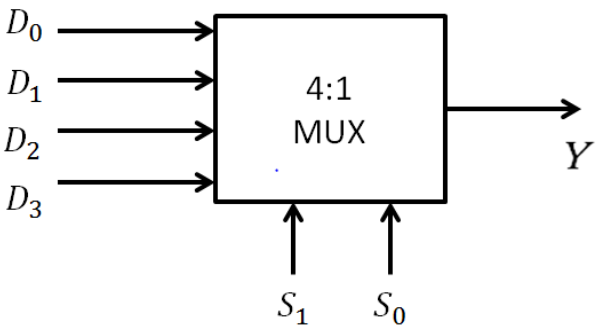
From the truth table if $S_0 = 0$ then $Y = D_0$. Similarly if $S_0 = 1$ then $Y = D_1$.

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

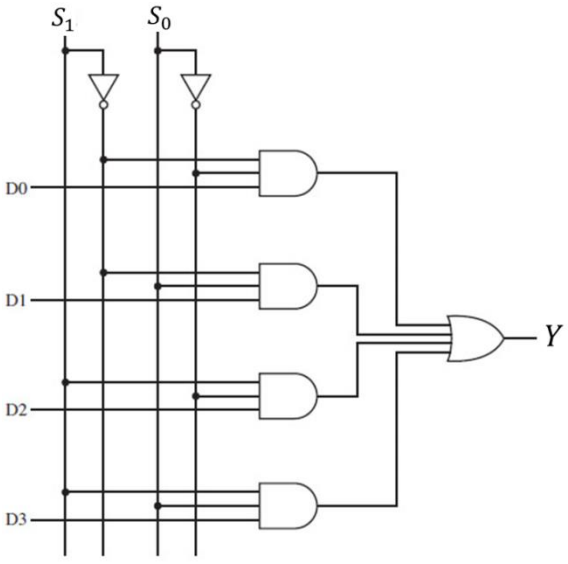
4:1 Mux

A 4:1 mux consists of four inputs D_0 to D_3,two selects inputs S_0 and S_1 and one outputs.

Select input selects one out of four inputs and passes on to the output Y.



Select i/p		o/p
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3



A 4:1 mux consists of four data inputs D_0 to D_3, single output Y and two select inputs S_0 and S_1. The select inputs S_0 and S_1 selects one out of four inputs and passes it to the output.

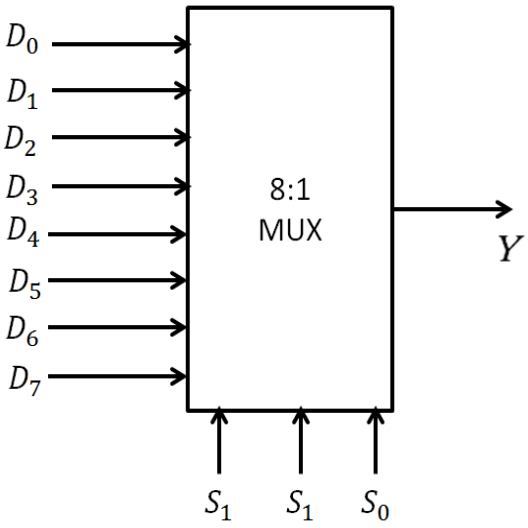
Figure shows 4:1 mux symbol, truth table and logic diagram. From the truth table if $S_0 = 0$ and $S_1 = 0$ then $Y = D_0$ and so on.

LogicExpression: $Y = \overline{S_1}\overline{S_0}D_0 + \overline{S_1}S_0D_1 + S_1\overline{S_0}D_2 + S_1S_0D_3$

8:1 Mux

A 8:1 mux consists of eight inputs D_0 to D_7,three selects inputs S_0 S_1 and S_2 and one output Y.

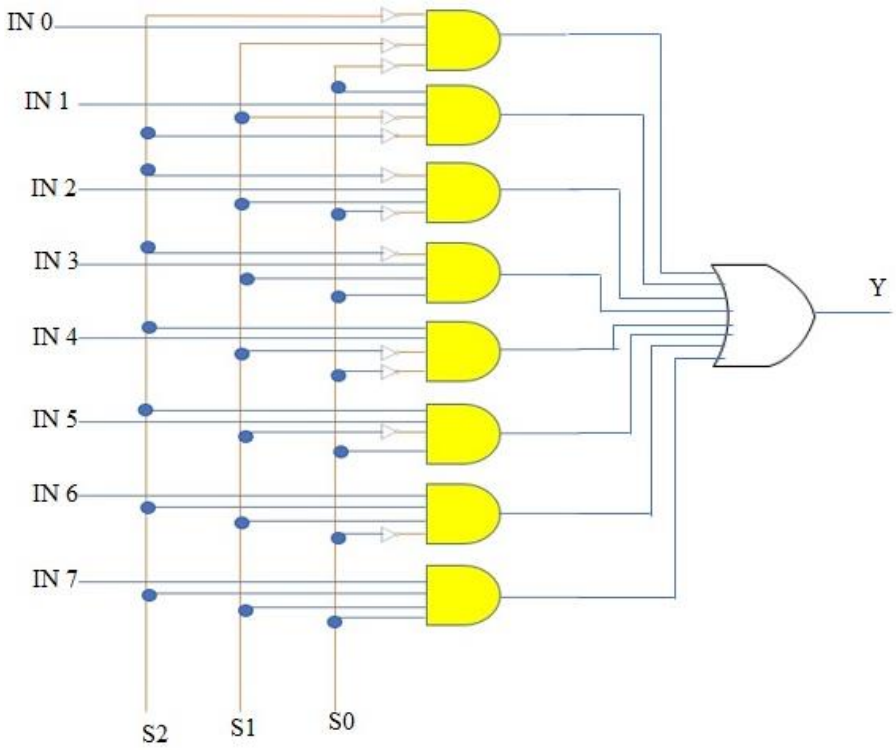
Select input selects one out of eight inputs and passes on to the output Y.



Functional block diagram

Select i/p			o/p
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

Truth table



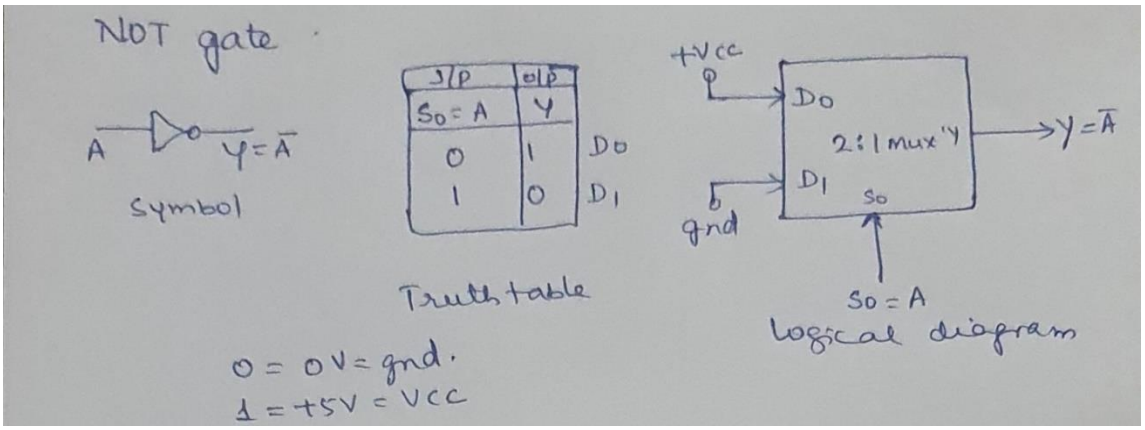
A 8:1 mux consists of eight data inputs D_0 to D_7, single output Y and three select inputs S_0 S_1 and S_2. The select inputs S_0 S_1 and S_2 selects one out of eight inputs and passes it to the output.

Figure shows 8:1 mux symbol, truth table and logic diagram. From truth table if $S_0 = 0$, $S_1 = 0$ and $S_2 = 0$ then $Y = D_0$ and so on.

Logic Expression:

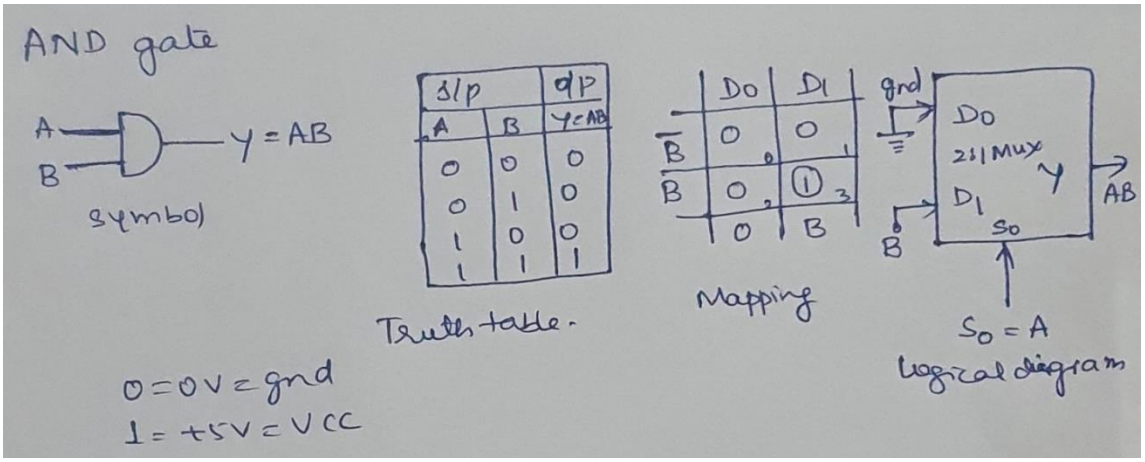
$$Y = \overline{S_2}\,\overline{S_1}\,\overline{S_0}D_0 + \overline{S_2}\,\overline{S_1}S_0D_1 + \overline{S_2}S_1\overline{S_0}D_2 + \overline{S_2}S_1S_0D_3 + S_2\overline{S_1}\,\overline{S_0}D_4 + S_2\overline{S_1}S_0D_5 + S_2S_1\overline{S_0}D_6 + S_2S_1S_0D_7$$

Implementation Of Logic Gates Using 2:1mux



Data input A is taken as select input of mux. Data input D_0 is connected to V_{CC} and D_1 is connected to ground. If $S_0 = 0$, output $Y = 1$ that equal V_{CC} similarly $S_0 = 1$, output $Y = 0$ = ground. Thus a NOT gate is implemented using 2:1 Mux. This also called as inverter.

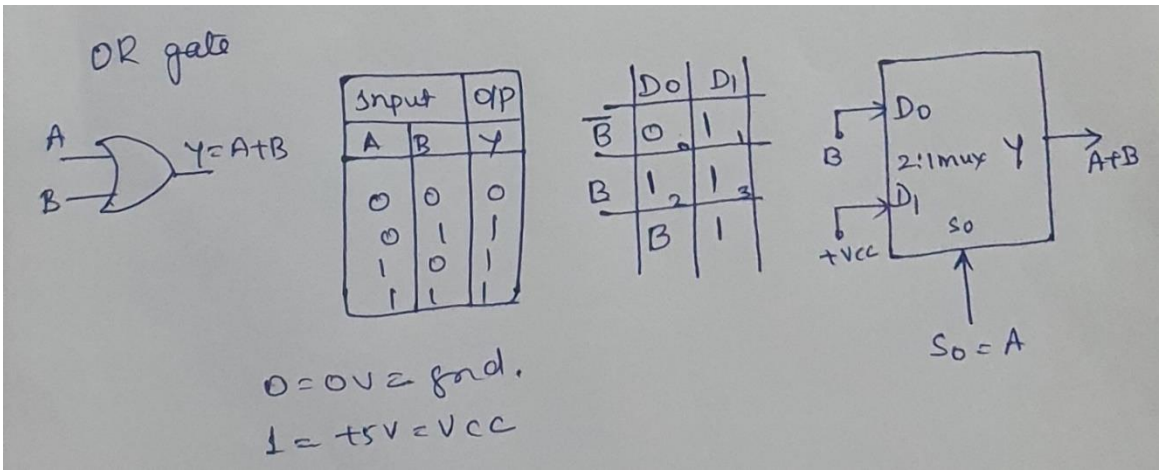
Implementation Of AND Gate Using 2:1 Mux



AND gate performs logical multiplication it has 2 or more inputs and one output. The output is high if the input are high or else low for all other input conditions.

The input is connected to ground (0) and D_1 is connected to input B select input to input A. Thus **2:1 mux can be used to implement AND gate.**

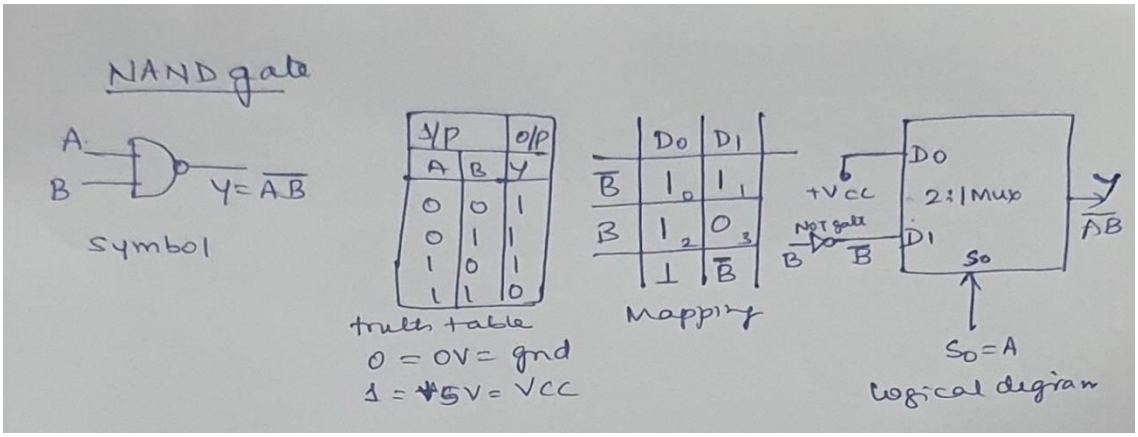
Implementation of OR Gate Using 2:1 Mux



OR gate performs logical addition it has 2 or more inputs and one output. The output is low when the inputs are low or else high for all other input conditions.

The input D_0 is connected to input B and D_1 is connected to $+V_{CC}$. Select input is taken as input A. Thus 2:1 mux can be used to implement OR gate.

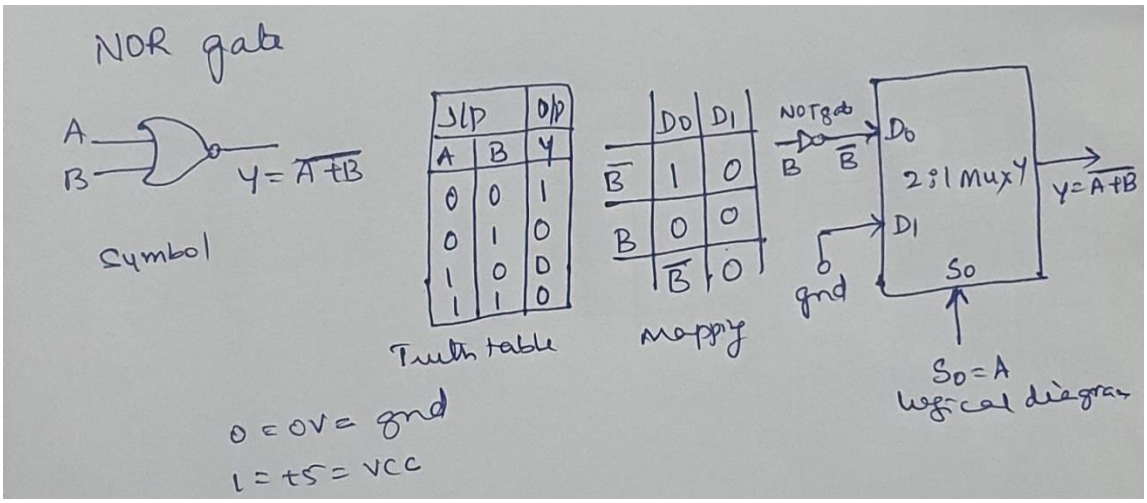
Implementation of NAND Gate Using 2:1 Mux



NAND is called universal gate, it has 2 or more inputs and one output. The output is low if the both inputs are low or else high for all other input conditions.

The input D_0 is connected to V_{CC} and D_1 is connected to $\overline{B}$ through an inverter. Select input is taken A input. Thus 2:1 mux can be used to implement NAND gate.

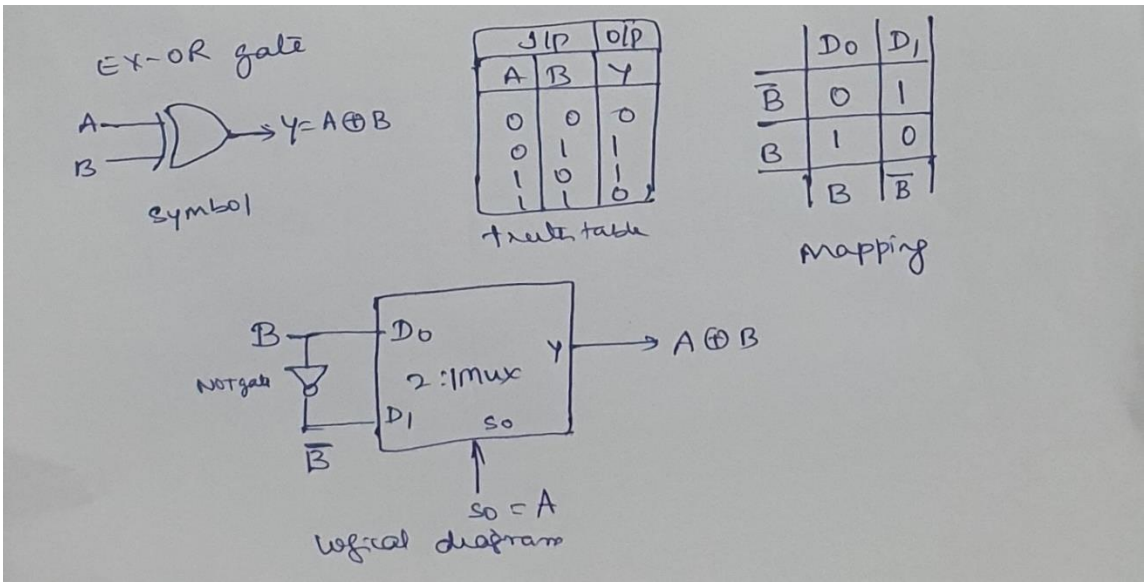
Implementation Of NOR Gate Using 2:1 Mux



NOR is called universal gate, it has 2 or more inputs and one output. The output is high if both inputs are low or else the output is low for all other input conditions.

Here D_0 is connected to $\bar{B}$ through an inverter and D_1 is connected ground and select input is taken A input. Thus 2:1 mux can be used to implemented NOR gate.

Implementation Of EX-OR Gate Using 2:1 Mux



EX-OR is special gate, it has 2 or more inputs and one output. The output is high if both inputs are different and low for same inputs.

The input D_0 is connected to B and D_1 is connected to $\bar{B}$ through an inverter. Select input is taken as A input. Thus 2:1 mux can be used to implement EX-OR gate

Realization of higher –order multiplexer using Lower-order multiplexer Ics.

Now, let us implement the following two higher-order Multiplexers using lower-order.

Multiplexers:

- 8x1 Multiplexer
- 16x1 Multiplexer

8x1 Multiplexer

In this section, let us implement 8x1 multiplexer using 4x1 multiplexers and 2x1 multiplexer. We know that 4x1 multiplexer has 4 data inputs, 2 selection lines and one output. Whereas, 8x1 multiplexer has 8 data inputs, 3 selection lines and one output.

So, we require two 4x1 multiplexers in first stage in order to get the 8 data inputs. Since, each 4x1 multiplexer produces one output, we require a 2x1 multiplexer in second stage by considering the outputs of first stage as inputs and to produce the final output.

Let the 8x1 multiplexer has eight data inputs I_7 to I_0, three selection lines S_2, S_1 and S_0 and one output Y. The Truth table of 8x1 multiplexer is shown below.

Select i/p			**o/p**
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3

1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

We can implement 8x1 Multiplexer using lower order Multiplexers easily by considering the above Truth table. The block diagram of 8x1 Multiplexer is shown in the following figure.

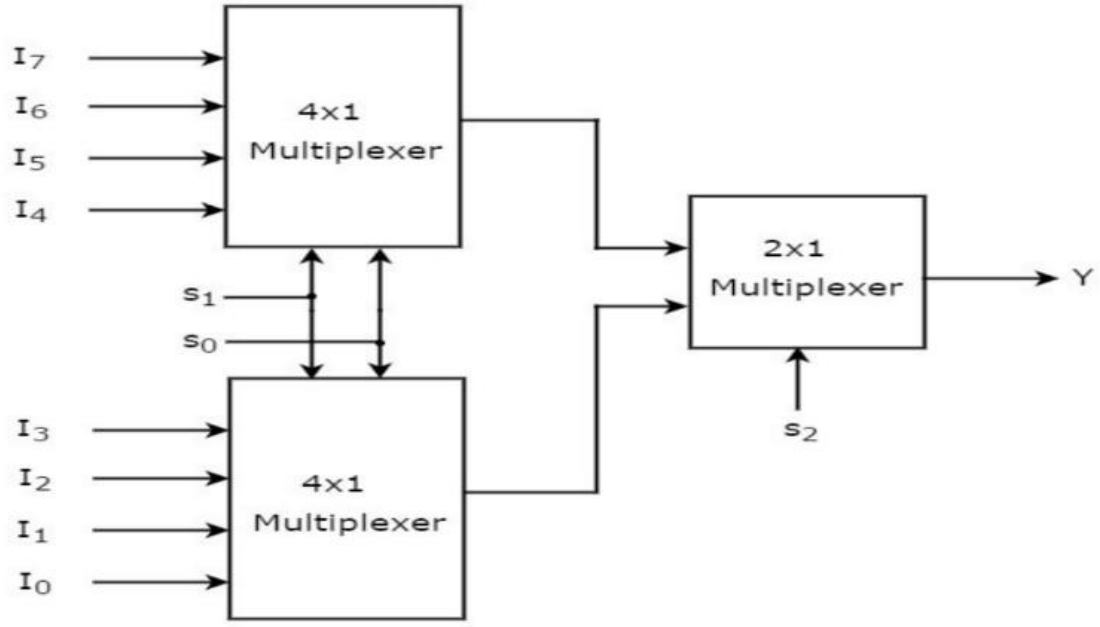


Fig shows how to implement 8x1 Multiplexer using 4x1 Multiplexers and 2x1 mux.

The same selection lines, S_1 & S_0 are applied to both 4x1 Multiplexers. The data inputs of upper 4x1 Multiplexer are I_7 to $I_4$and the data inputs of lower 4x1 Multiplexer are I_3 to I_0. Therefore, each 4x1 Multiplexer produces an output based on the values of selection lines, $S_1$and S_0. The outputs of first stage 4x1 Multiplexers are applied as inputs of 2x1 Multiplexer that is present in second stage. The other selection line, S_2 is applied to 2x1 Multiplexer.

If S_2 is zero, then the output of 2x1 Multiplexer will be one of the 4 inputs I_3 to I_0 based on the values of selection lines $S_1$and S_0.

If S_2 is one, then the output of 2x1 Multiplexer will be one of the 4 inputs I_7 to I_4based on the values of selection lines $S_1$and S_0.

Therefore, the overall combination of two 4x1 Multiplexers and one 2x1 Multiplexer performs as one 8x1 Multiplexer.

16x1 Multiplexer

In this section, let us implement 16x1 Multiplexer using 8x1 Multiplexers and 2x1 Multiplexer. We know that 8x1 Multiplexer has 8 data inputs, 3 selection lines and one output. Whereas, 16x1 Multiplexer has 16 data inputs, 4 selection lines and one output. So, we require two 8x1 Multiplexers in first stage in order to get the 16 data inputs. Since, each 8x1 Multiplexer produces one output, we require a 2x1 Multiplexer in second stage by considering the outputs of first stage as inputs and to produce the final output. Let the 16x1 Multiplexer has sixteen data inputs I_{15} to I_0, four selection lines S_3 to S_0 and one output Y. The Truth table of 16x1 Multiplexer is shown below.

Select i/p				**o/p**
S_3	S_2	S_1	S_0	Y
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5

0	1	1	0	I_6
0	1	1	1	I_7
1	0	0	0	I_8
1	0	0	1	I_9
1	0	1	0	I_{10}
1	0	1	1	I_{11}
1	1	0	0	I_{12}
1	1	0	1	I_{13}
1	1	1	0	I_{14}
1	1	1	1	I_{15}

We can implement 16x1 Multiplexer using lower order Multiplexers easily by considering the above Truth table. The block diagram of 16x1 Multiplexer is shown in the following figure.

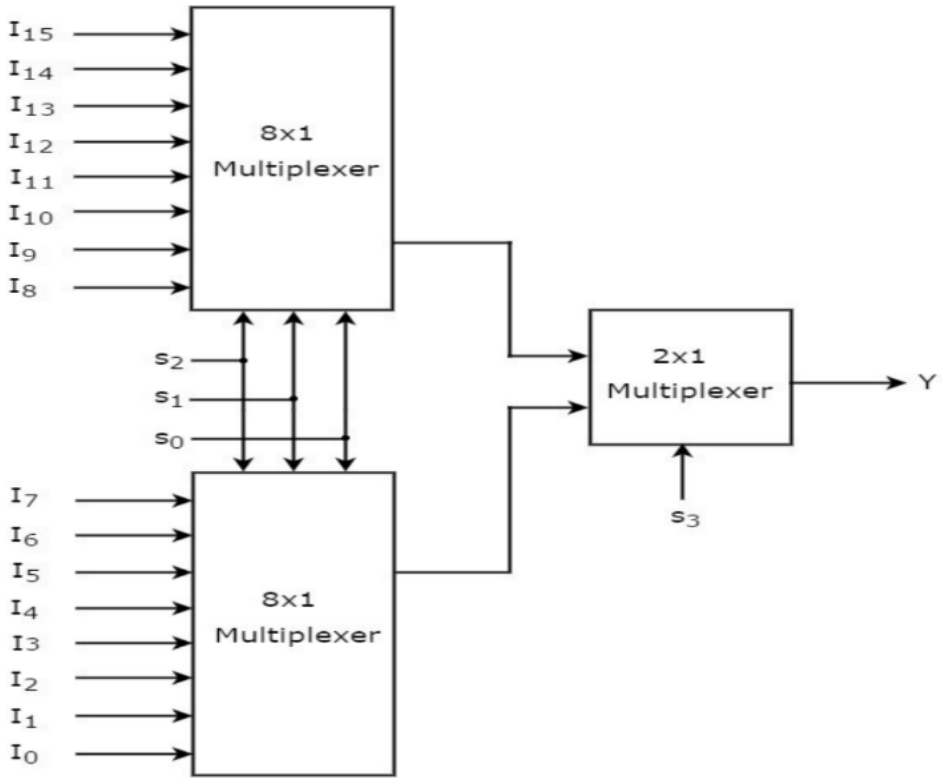


Fig shows how to implement 16x1 Multiplexer using 8x1 Multiplexers and 2x1.

Multiplexer ICS And Features.

IC number	Description	Output
74150	16 to 1 mux	Inverted input
74151	8 to 1 mux	Complementary outputs
74152	8 to 1 mux	Inverted input
74153	Dual 4 to 1 mux	Same as input
74157	Quad 2 to 1 mux	Same as input
74158	Quad 2 to 1 mux	Inverted input

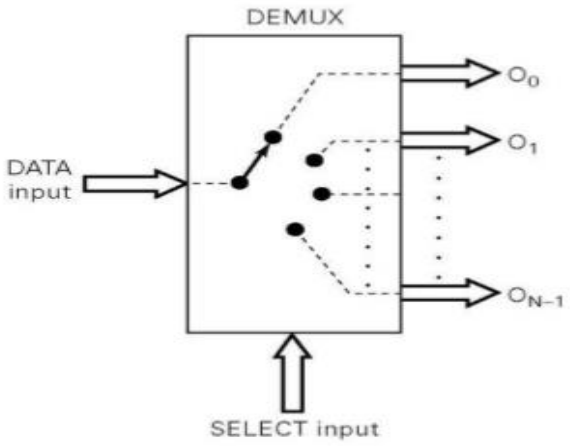
Multiplexers are available as MSI- IC format. The table indicates Multiplexer IC numbers for TTL logic family. CMOS Multiplexer ICs are also popular amongst the

Digital system designer because of low power consumption. Below table indicates list of few CMOS multiplexer ICs.

Demultiplexer

Demultiplexer has a single input and n output line.

The word "Demultiplex" means "One into many". DeMultiplexing is the process of taking information from one input and transmitting the same over one of several outputs.



It has one data input, m select inputs and n output lines. A demux selects one input and distributes to one out of n output lines using select lines. Thus demux is also called data distributer.

Classification

- 1:2 Demux (1 select input)
- 1:4 Demux (2 select input)
- 1:8 Demux (3 select input)
- 1:16 Demux (4 select input)
- 1:32 Demux (5 select input)

Realization Of 1:2 Demultiplexer.

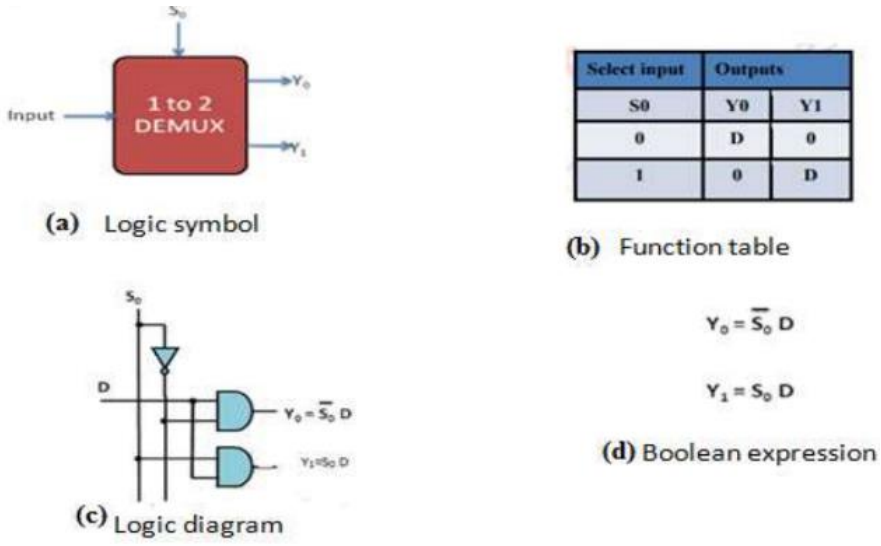


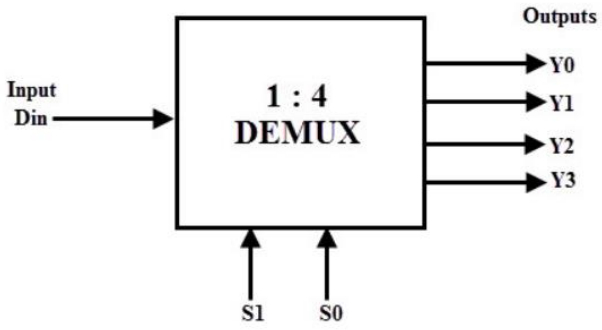
Fig:1.9 - 1:2 Demultipexer

shows the logic symbol, function table ,logic diagram and Boolean expression respectively, for 1:2 demultiplexer. The input data bit is labeled D. The input data line is connected to both AND gates. The select line S enables only one gate at a time. The data D will pass through the enabled gate to the output line. When $S = 0$ the upper AND gate is enabled while the lower and gate is disabled. Therefore data bit D is transmitted only to the Y_0output, giving $Y_0 = D$. If D is low, Y_0 is low. If D is high, Y_0 is high. The value of Y_0 depends on value of D. The other output Y_1 is low state. If control input is changed to $S = 1$, the lower AND gate is enabled while the upper AND gate is disabled. Then D is transmitted only to the Y_1 output and $Y_1 = D$. The Boolean expressions for the output as follows:

$$Y_0 = \bar{S}D$$

$$Y_1 = SD$$

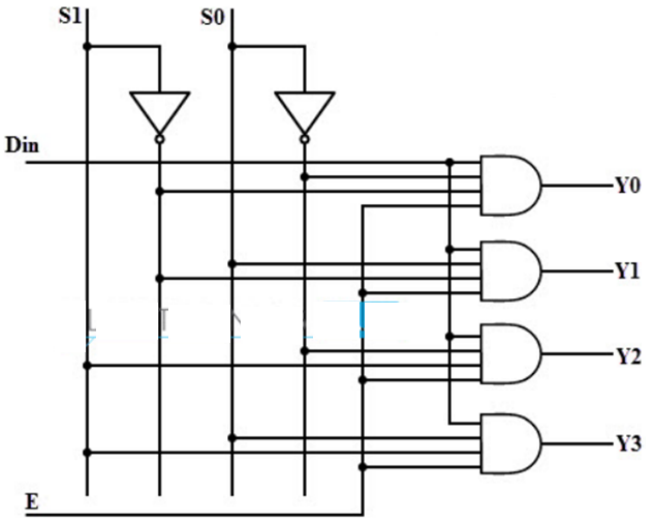
1:4 Demux



A 1:4 demux has a single input D, two selection inputs S_1 and S_0 and four outputs Y_0 to Y_3.

The input data goes to any one of the four outputs at a given time for a particular combination of select lines. This demux is also called as a 2-to-4 demux which means that two select lines and four output lines.

Data Input	Select Inputs		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0



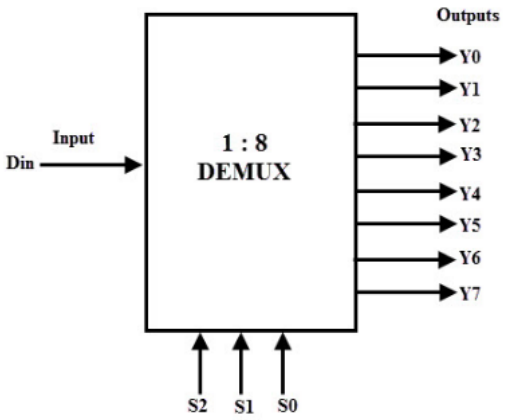
$$Y_0 = \bar{S_1}\bar{S_0}\mathrm{D}$$
$$Y_1 = \bar{S_1}S_0\mathrm{D}$$
$$Y_2 = S_1\bar{S_0}\mathrm{D}$$
$$Y_3 = S_1S_0\mathrm{D}$$

The functional block diagram and symbol and truth table and logic diagram of 1:4 Demux. It as data input D, four output Y_3 to Y_0, two select inputs S_1 and S_0.

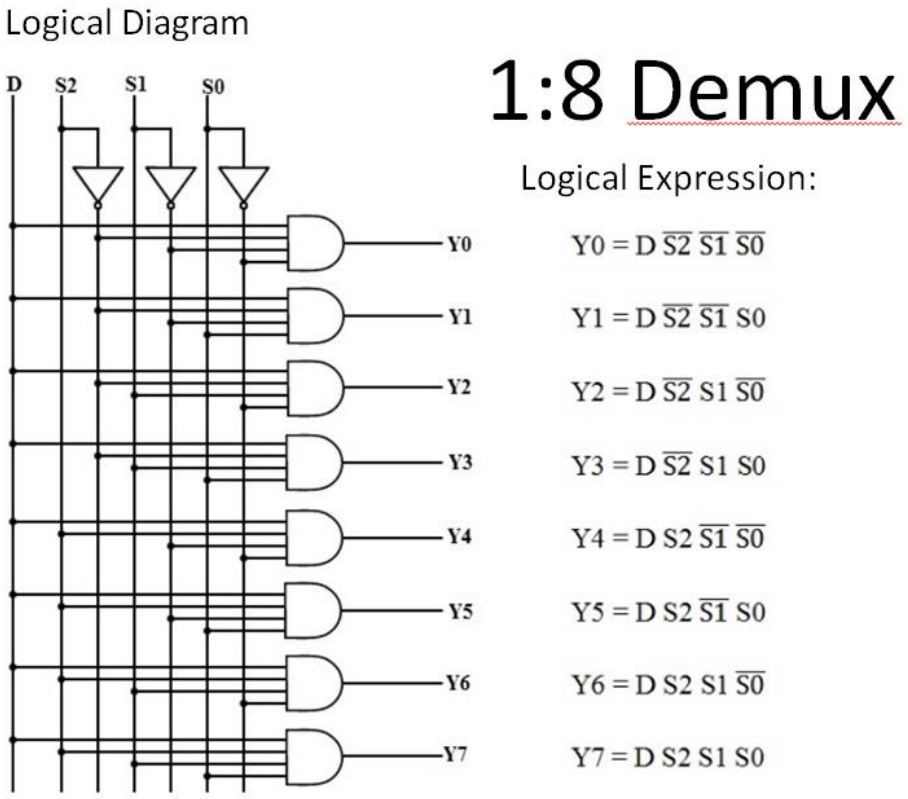
The input data appears on one out of four outputs depending on select inputs.

From the truth table, it is clear that the data input is connected to output Y_0 when $S_1 = 0$ and $S_0 = 0$ and the data input is connected to output Y_1 when $S_1 = 0$ and $S_0 = 1$. Similarly, the data input is connected to outputs Y_2 when $S_1 = 1$ and $S_0 = 0$ and output Y_3 when $S_1 = 1$ and $S_0 = 1$.

1:8 Demux:



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



The functional block diagram and symbol and truth table and logic diagram of 1:8 Demux. It as data input D, eight output Y_7 to Y_0, three select inputs S_2, S_1 and S_0.

The input data appears on one out of eight outputs depending on select inputs.

From the truth table, it is clear that the data input is connected to output Y_0 when $S_2 = 0$, $S_1 = 0$ and $S_0 = 0$ and the data input is connected to output Y_1 when $S_2 = 0$, $S_1 = 0$ and $S_0 = 1$ and so on.

A 1:8 demux has a single input D, three selection inputs S_2, S_1 and S_0 and eight outputs Y_0 to Y_7.

The input data goes to any one of the eight outputs at a given time for a particular combination of select lines. This demux is also called as a 3-to-8 demux which means that three select lines and eight output lines.

Demultiplexer Ics

So far we have discussed construction of demultiplexers using discreet logic gates. Commercially, demultiplexers are available as MSI- IC format. Below indicates a table of demultiplexer IC numbers for TTL logic family

IC number	Description	Output
74138	1 to 8 demux/decoder	Inverted input
74139	Dual 1 to 4 demux/decoder	Inverted input
74155	Dual 1 to 4 demux/decoder	Complementary inputs
74156	Dual 1 to 4 demux/decoder	Open collecter – complementary inputs
74154	1 to 16 demux/decoder	Same as input
74159	1 to 16 demux/decoder	Open collecter- same as input

Applications of Demultiplexers

1. Data demultiplexing
2. Clock demultiplexing
3. Memory addressing
4. Four phase clock generator
5. Function generation using DMUX
6. Switch encoding
7. Serial to parallel converter

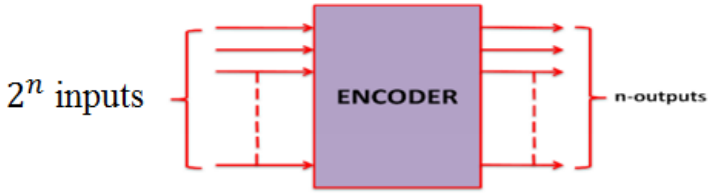
Encoder

Encoder is combinational logic circuit that converts symbols and numbers into a coded form such as binary.

An encoder has 2^n number of input lines and 'n' number of output lines. Out of 2^n inputs only one input line will be active at a given time to produce n-bit code. For if $n = 3$, then $2^3 = 8$, input lines thus it is called 8-line to 3-line encoder or 8:3 encoder..

Some of the most commonly used encoders are

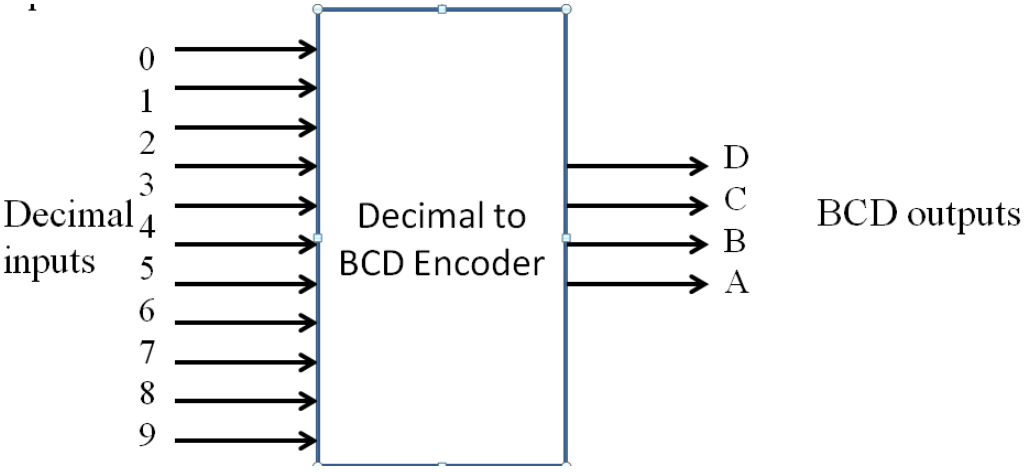
- Octal to binary,
- Decimal to BCD
- Priority encoders.



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



Decimal Digit	BCD Code			
	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Truth Table

Below Boolean functions are formed by OR-ing all the input lines for which output is 1. For instance Y_0 is 1 for D_1, D_3, D_5, D_7 & D_9 input lines.

$$\mathbf{A = D_1 + D_3 + D_5 + D_7 + D_9}$$

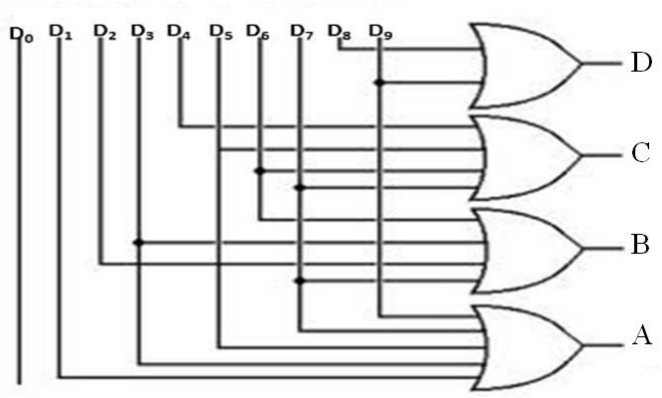
$$\mathbf{B = D_2 + D_3 + D_6 + D_7}$$

$$\mathbf{C = D_4 + D_5 + D_6 + D_7}$$

$$\mathbf{D = D_8 + D_9}$$

The decimal to BCD encoder can therefore be implemented with OR gates whose inputs are determined directly from truth table.

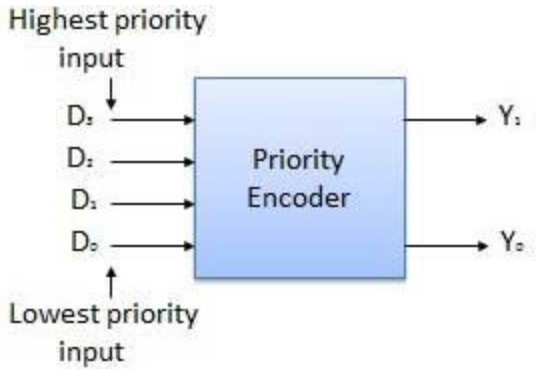
From these Boolean expressions, the Decimal to BCD Encoder can be implemented by using simply three 4 OR gates.



When any decimal input is high the appropriate level occurs on 4 BCD output lines. For ex, decimal 3 is closed then BCD output is 0011

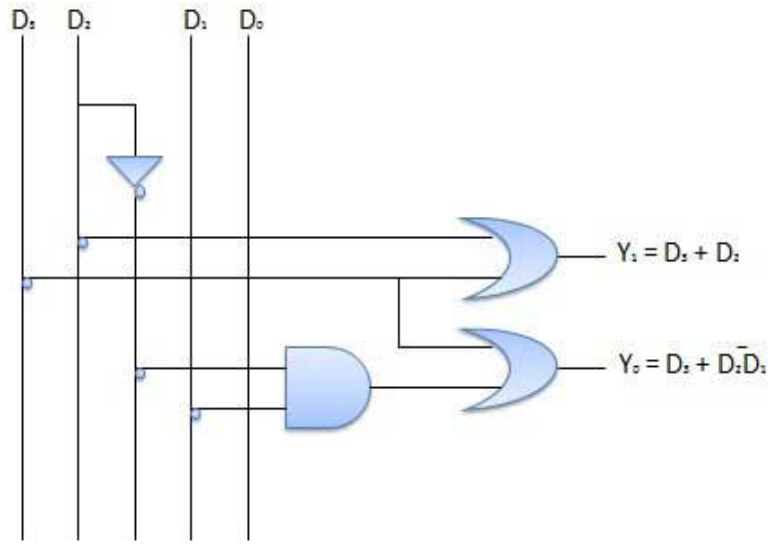
Priority Encoder

This is a special type of encoder. Priority is given to the input lines. If two or more input line 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 $Y_1 = 11$ 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.

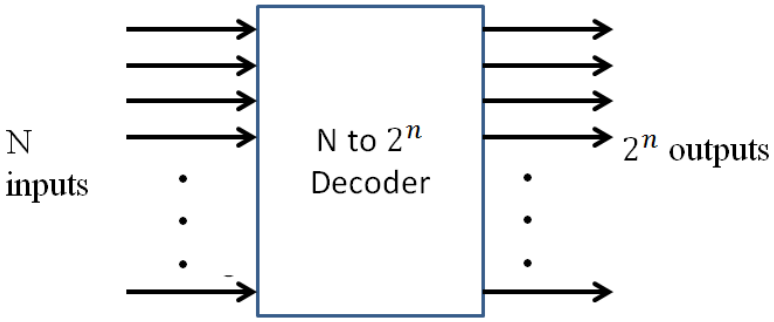
Highest	Inputs		Lowest	Outputs	
D_3	D_2	D_1	D_0	Y_0	Y_1
0	0	0	0	x	x
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1



Decoder

A decoder is a combinational logic circuit that converts a binary code of n input lines into a one out of 2^n output code.

One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled.

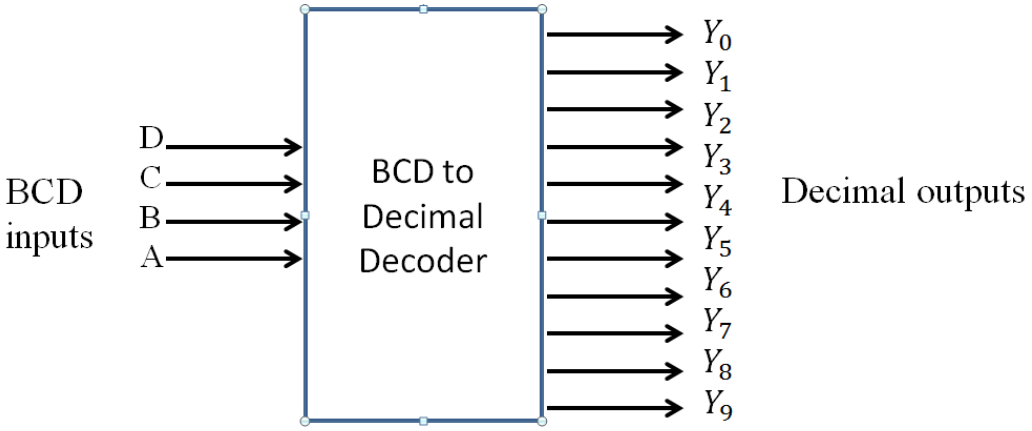


If a binary decoder receives n inputs it activates one and only one of its 2^n outputs based on that input with all other outputs deactivated. If the n -bit coded information has unused combinations, the decoder may have fewer than 2^n outputs.

BCD To Decimal Decoder

BCD to decimal decoder converts each BCD code word into one of the ten possible decimal digits. It has 4 input lines A,B,C,D and 10 output lines $Y_0, Y_1, Y_2, Y_3, .., Y_9$

Hence it is also called 4 lines to 10 lines decoder



Truth Table													
BCD Inputs				Outputs									
A	B	C	D	Y_9	Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	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	0	1	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0

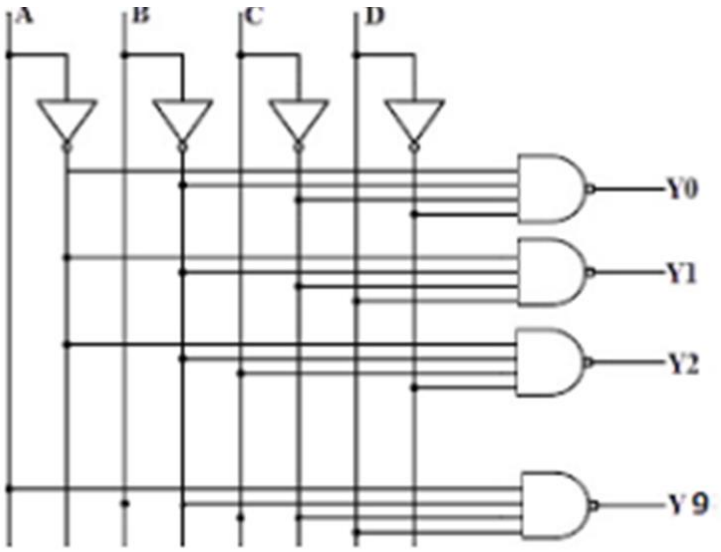
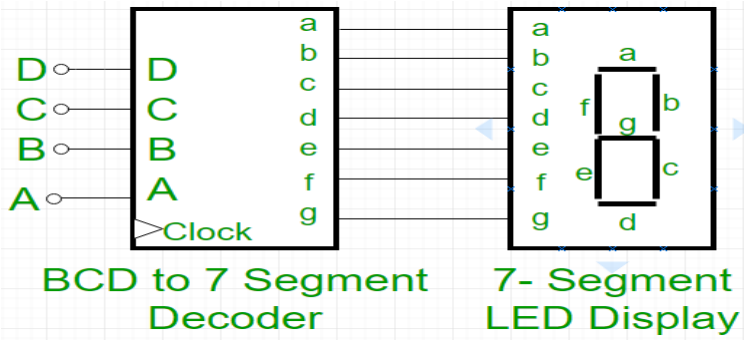


Figure shows truth table logic symbol and logic diagram of BCD to decimal decoder.

From truth table if ABCD = 0000 then Y_0 will be high(1) and all others will be low(0) and so on.

Thus BCD to decimal decoder can be implemented using 4 NOT gates and 10 AND gates as shown in logic diagram.

BCD To 7-Segment Decoder



To display the characters and numbers in order to produce decimal readout 7 segment display are used.

A display decoder is used to convert BCD or a binary code into 7 segment code.

It has 4 input lines and 7 output lines as shown in the figure below.

Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

For ex, to display number 3 segments a,b,c,d and g will be illuminated. Similarly to display number 9 segments a,b,c,d,f and g will be illuminated. Thus a 7 segment display is used in latches, decade counters etc

	ENCODER	DECODER
1	Encoder circuit basically converts the applied information signal into a coded digital bit stream.	Decoder performs reverse operation and recovers the original information signal from the coded bits.
2	In case of encoder, the applied signal is the active signal input.	Decoder accepts coded binary data as its input.
3	The number of inputs accepted by an encoder is 2n.	The number of input accepted by decoder is only n inputs.
4	The output lines for an encoder is n.	The output lines of an decoder is 2n.
5	The encoder generates coded data bits as its output.	The decoder generates an active output signal in response to the coded data bits.
6	The operation performed is simple.	The operation performed is complex.
7	The encoder circuit is installed at the transmitting end.	The decoder circuit is installed at the receiving side.
8	OR gate is the basic logic element used in it.	AND gate along with NOT gate is the basic logic element used in it.
9	It is used in E-mail, video encoders etc.	It is used in Microprocessors, memory chips etc.