Indian Institute of Technology, Guwahati

***Scribe CS221 Digital Design***

9 Sept 2017



**Made By:**

**Arpan Konar (160101014)**

**Daman Tekchandani (160101024)**

**Topics Covered : Comparators, Encoders, Decoders, Multiplexers and DeMultiplexers**

TABLE OF CONTENTS

[**Encoder** 5](#_gjdgxs)

[**Decoder** 12](#_30j0zll)

[**DEMULTIPLEXOR** 18](#_1fob9te)

[**COMPARATOR** 25](#_2et92p0)

**MULTIPLEXOR** [35](#_1t3h5sf)

Encoder

INTRODUCTION

Digital Encoder, more commonly called a Binary Encoder takes all its data inputs one at a time and then converts them into a single encoded output. So, we can say that a binary encoder is a multi-input combinational logic circuit that converts the logic level “1” data at its inputs into an equivalent binary code at its output.

Encoders perform exactly reverse operation than decoder. An encoder has n input and m output lines. Out of n input lines only one is activated at a time and produces equivalent code on output m lines. If a device output code has fewer bits than the input code has, the device is usually called an encoder.



*Figure 1: Encoder*

The encoder in *Figure 1* has n input lines and m output lines where n=2m.

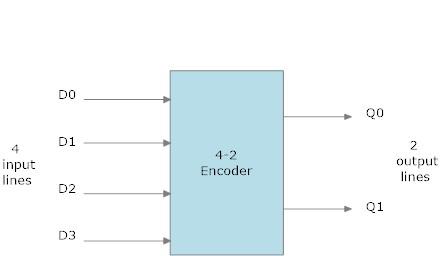
Generally, digital encoders produce outputs of 2-bit, 3-bit or 4-bit codes depending upon the number of data input lines. An “n-bit” binary encoder has 2n input lines and n-bit output lines with common types that include 4-to-2, 8-to-3 and 16-to-4 line configurations.

The output lines of a digital encoder generate the binary equivalent of the input line whose value is equal to “1” and are available to encode either a decimal or hexadecimal input pattern to typically a binary or BCD (binary coded decimal) output code.

**TYPES AND EXAMPLES**

**4-to-2 Bit Binary Encoder:**

The four-bit encoder allows only four inputs such as D0, D1, D2, D3 and generates the two outputs Q0, Q1, as shown in below diagram.

**

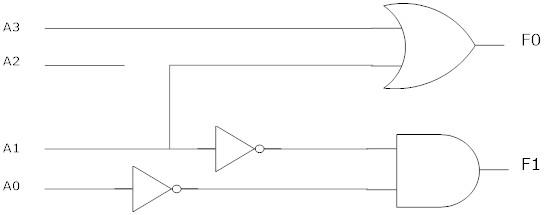
*: Block diagram of 4-to-2-bit encoder*

Truth table of a 4-to-2-bit encoder is shown below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| D3 | D2 | D1 | D0 | Q1 | Q0 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 |

*Truth Table of 4-to-2-bit encoder*

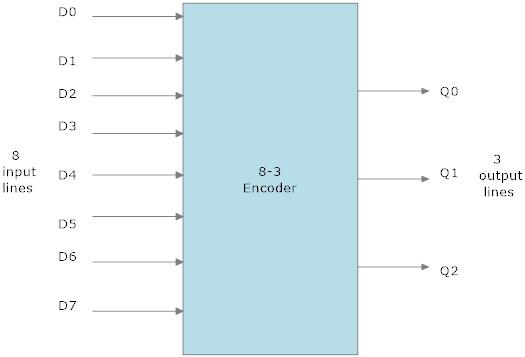
The logic diagram of 4-to-2-bit encoder is given below:

**

*Logic Diagram of 4-to-2 bit encoder*

Octal to binary (8-3 bit) encoder:

Octal-to-Binary encoder take 8 inputs and provides 3 outputs. At any one time, only one input line has a value of 1. The figure below shows the block diagram of an Octal-to-binary encoder.



*Block diagram of 8-to-3-bit encoder*

The truth table of 8-to-3-bit encoder is as follows:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Y2 | Y1 | Y0 |
| 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 of 8-to-3-bit encoder*

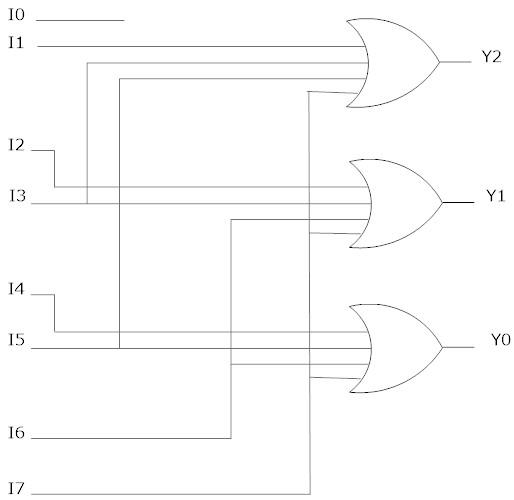
For an 8-to-3 binary encoder with inputs I0-I7 the logic expressions of the outputs Y0-Y2 are:

Y0 = D1 + D3 + D5 + D7

Y1 = D2 + D3 + D6 + D7

Y2 = D4 + D5 + D6 + D7

The logic Diagram and Block Diagram are as follows:



*Logic Diagram of 8-to-3-bit encoder*

Disadvantages of Standard Encoders:

One of the main disadvantages of standard digital encoders is that they can generate the wrong output code when there is more than one input present at logic level “1”. For example, in our first example of 4-2 encoder, if we make inputs A1 and A2 HIGH at logic “1” both at the same time, the resulting output is neither at “01” or at “10” but will be at “11” which is an output binary number that is different to the actual input present. Also, an output code of all logic “0” s can be generated when all of its inputs are at “0” OR when input A0 is equal to one.

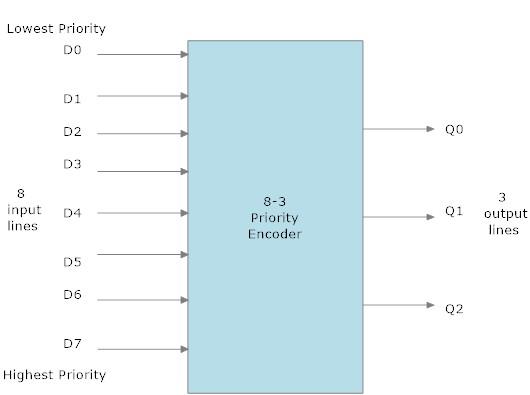
One simple way to overcome this problem is to “Prioritise” the level of each input pin and if there was more than one input at logic level “1” the actual output code would only correspond to the input with the highest designated priority. Then this type of digital encoder is known commonly as a Priority Encoder or P-encoder for short.

PRIORITY ENCODER

The Priority Encoder solves the problems mentioned above by allocating a priority level to each input. The priority encoder’s output corresponds to the currently active input which has the highest priority. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored.

8-to-3 Bit Priority Encoder

TTL 74LS148 is an 8-to-3 bit priority encoder which has eight active LOW (logic “0”) inputs and provides a 3-bit code of the highest ranked input at its output.



*Block Diagram of 8-to-3 bit priority encoder*

Priority encoders output the highest order input first for example, if input lines “D2“, “D3” and “D5” are applied simultaneously the output code would be for input “D5” (“101”) as this has the highest order out of the 3 inputs. Once input “D5” had been removed the next highest output code would be for input “D3” (“011”), and so on.

The truth table for a 8-to-3 bit priority encoder is given as:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Digital inputs | | | | | | | | Binary Outputs | | |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | Q2 | Q1 | Q0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | X | X | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | X | X | X | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | X | X | X | X | 1 | 0 | 0 |
| 0 | 0 | 1 | X | X | X | X | X | 1 | 0 | 1 |
| 0 | 1 | X | X | X | X | X | X | 1 | 1 | 0 |
| 1 | X | X | X | X | X | X | X | 1 | 1 | 1 |

*Truth Table for 8-3 encoder*

Where X equals “don’t care”, that is logic “0” or a logic “1”.

From this truth table, the Boolean expression for the encoder above with data inputs D0 to D7 and outputs Q0, Q1, Q2 is given as:

Output Q0:

Q0=∑ (1,3,5,7)

Output Q1:

Q1 = ∑ (2,3,6,7)

Output Q2:

Q2 = ∑ (4,5,6,7)

Then the final Boolean expression for the priority encoder including the zero inputs is defined as:

***Q0 =∑ (6(42D1 + 4D3 + D5)+ D7)***

***Q1 = ∑ (54(D2+D3)+ D6 + D7)***

***Q2 = ∑ (D4 + D5 + D6 + D7)***

In practice these zero inputs would be ignored allowing the implementation of the final Boolean expression for the outputs of the 8-to-3 **priority encoder.**

Decoder

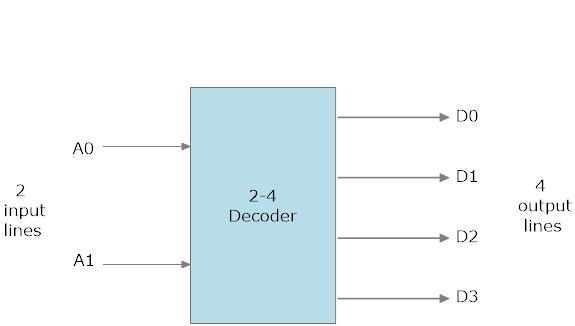
INTRODUCTION

The decoder is an electronic device that is used to convert digital signal to an analogue signal. It allows single input line and produces multiple output lines. The decoders are used in many communication projects that are used to communicate between two devices. The decoder allows n inputs and generates 2n numbers of outputs. For example, if we give 2 inputs it will produce 4 outputs by using 2-4 decoder.

**TYPES AND EXAMPLES**

2-4 decoder

In this type of decoder, it contains two inputs A0, A1, and four outputs represented by D0, D1, D2, and D3.



*Block Diagram of 2-to-4 decoder*

As you can see in the truth table – for each input combination, one output line is activated.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A1 | A0 | D3 | D2 | D1 | D0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

*Truth table of 2-to-4 decoder*

In this example, you can notice that, each output of the decoder is actually a minterm, resulting from a certain inputs combination, that is:

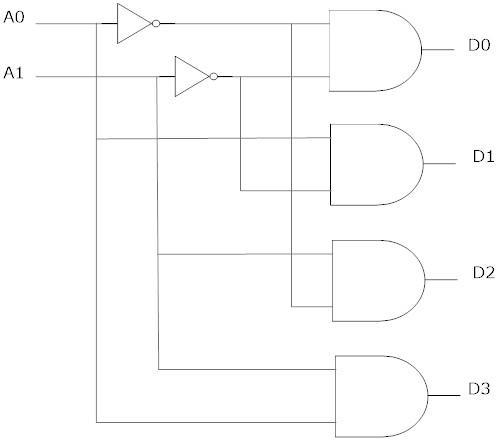
*D0 = 10*, (minterm m0) which corresponds to input 00

*D1 = 1A0*, (minterm m1) which corresponds to input 01

*D2 = A10*, (minterm m2) which corresponds to input 10

*D3 = A1A0*, (minterm m3) which corresponds to input 11

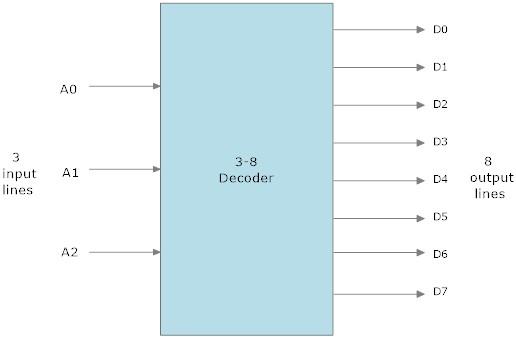
The circuit is implemented with AND gates, as shown in the figure. Each output of the decoder will be generated according to the input combination.



*Logic Diagram of 2-to-4 decoder*

3-8 Decoder

The 3-8 decoder uses all AND gates, and therefore, the outputs are active- high. For active- low outputs, NAND gates are used. It has 3 input lines and 8 output lines. It is also called as binary to octal decoder it takes a 3-bit binary input code and activates one of the 8(octal) outputs corresponding to that code.



*3-to-8-bit Decoder*

The truth table is as follows:

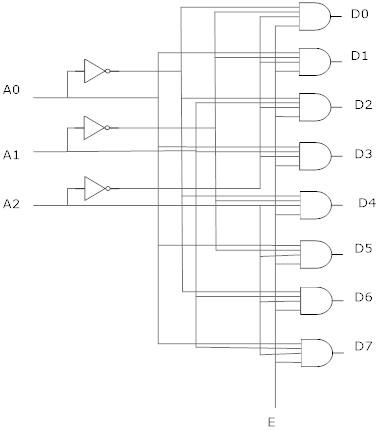
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A2 | A1 | A0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Truth table of 3-to-8 decoder*

In this example, you can notice that, each output of the decoder is actually a minterm, resulting from a certain inputs combination, that is;

* *D0 =210*, (minterm m0) which corresponds to input 000
* *D1 = 21A0*, (minterm m1) which corresponds to input 001
* *D2 = 2A10*, (minterm m2) which corresponds to input 010
* *D3 = 2A1A0*, (minterm m3) which corresponds to input 011
* *D4 = A210*, (minterm m0) which corresponds to input 100
* *D5 = A21A0*, (minterm m1) which corresponds to input 101
* *D6 = A2A10*, (minterm m2) which corresponds to input 110
* *D7 = A2A1A0*, (minterm m3) which corresponds to input 111

The circuit is implemented with AND gates, as shown in the figure.

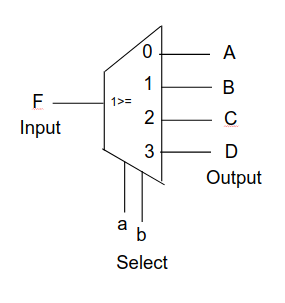


*Logic diagram of 3-to-8-bit encoder*

Demultiplexor

INTRODUCTION

A demultiplexer (or demux) is a device taking a single input signal and selecting one of many data-output-lines, which is connected to the single input. The demultiplexer converts a serial data signal at the input to a parallel data at its output lines as shown below.



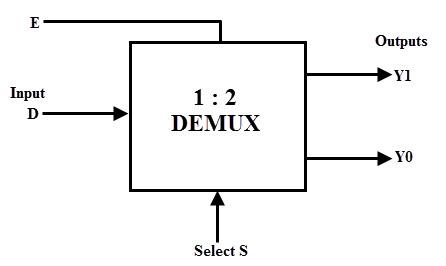
*Demux*

The function of the Demultiplexer is to switch one common data input line to any one of the 4 output data lines A to D in our example above.

**TYPES AND EXAMPLES**

1-to-2 Demux

A 1-to-2 demultiplexer consists of one input line(D), two output lines (Y0 , Y1) and one select line(S).

**

*Block diagram of 1-to-2 Demux*

The truth table of a 1-to-2 demultiplexer is shown below in which the input is routed to Y0 and Y1 depends on the value of select input S.

|  |  |  |  |
| --- | --- | --- | --- |
| S | D | Y0 | Y1 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |

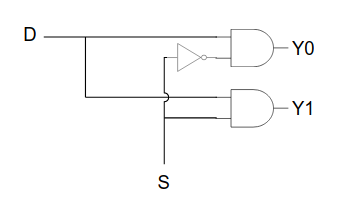
*Truth table of 1-to-2 Demux*

From the truth table we can write the Boolean expression of Y0 and Y1 as:

Y0 = SD

Y1 = S̅D

From the above truth table, the logic diagram of this demultiplexer can be designed by using two AND gates and one NOT gate as shown in below figure.

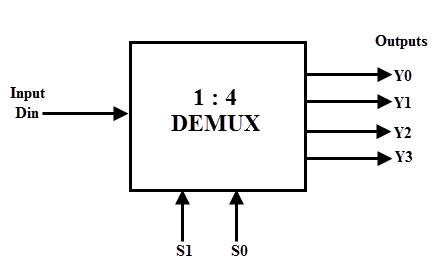


*Logic Diagram of 1-to-2 Demux*

1-to-4 Demux

A 1-to-4 demultiplexer has a single input (D), two selection lines (S1 and S0) and four outputs (Y0 to Y3).

The block diagram of 1:4 DEMUX is shown below.



*Block Diagram of 1-to-4 Demux*

The truth table of 1-to-4 demux is given below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input | Select Lines | | Output Lines | | | |
| D | S1 | S0 | Y3 | Y2 | Y1 | Y0 |
| 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 |

*Truth table of 1-to-4 Demux*

From the truth table we can write the Boolean expression for Y0 – Y3 as:

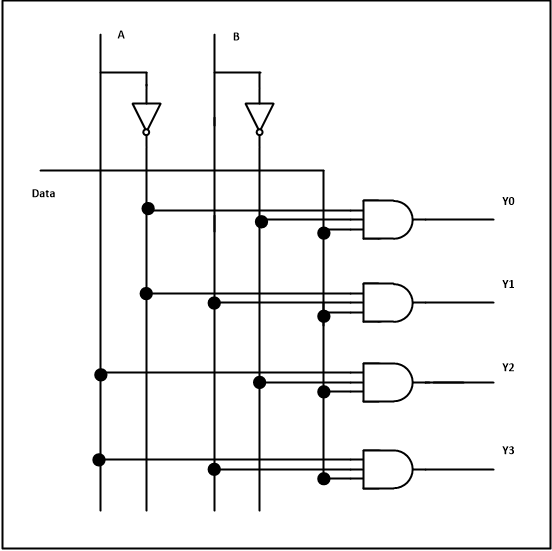
Y0 = S̅0S̅1D

Y1 = S̅0S1D

Y2 = S0S̅1D

Y3 = S0S1D

From the above Boolean expressions, a 1-to-4 demux can be implemented by using four 3-input AND gates and two NOT gates as shown in figure below. The two selection lines enable the particular gate at a time.

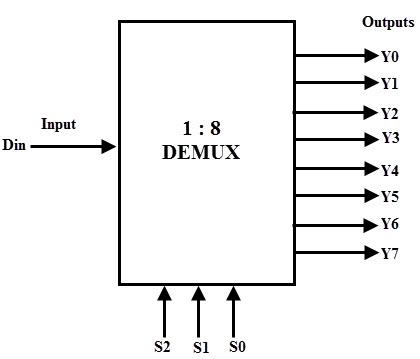


*Logic Diagram of 1-to-4 Demux*

1-to-8 Demux

1-to-8 demultiplexer consists of single input D, three select inputs S2, S1 and S0 and eight outputs from Y0 to Y7.

The block diagram of 1:8 Demux is shown below.



*Block diagram of 1-to-8 Demux*

The truth table of 1-to-8 demux is given below:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input | Select Lines | | | Output Lines | | | | | | | |
| D | S2 | S1 | S0 | Y7 | Y6 | Y5 | Y4 | Y3 | Y2 | Y1 | Y0 |
| 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 |

*Truth table of 1-to-8 demux*

From the truth table we can write the Boolean expression for Y0 – Y7 as:

Y0 = S̅0S̅1S̅2D

Y1 = S̅0S̅1S2D

Y2 = S̅0S1S̅2D

Y3 = S̅0S1S2D

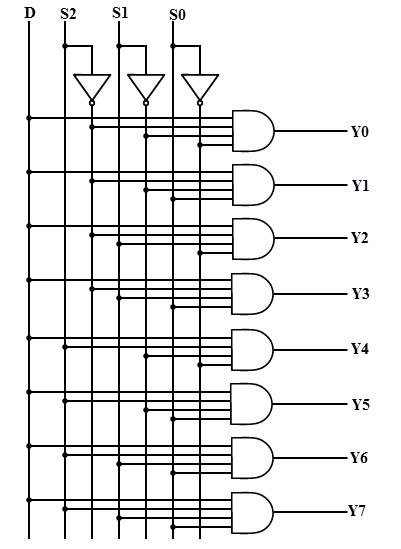
Y4 = S0S̅1S̅2D

Y5 = S0S̅1S2D

Y6 = S0S1S̅2D

Y7 = S0S1S2D

From these obtained equations, the logic diagram of this demultiplexer can be implemented by using eight AND gates and three NOT gates as shown in below figure.



*Logic Diagram of 1-to-8 Demux*

COMPARATORS

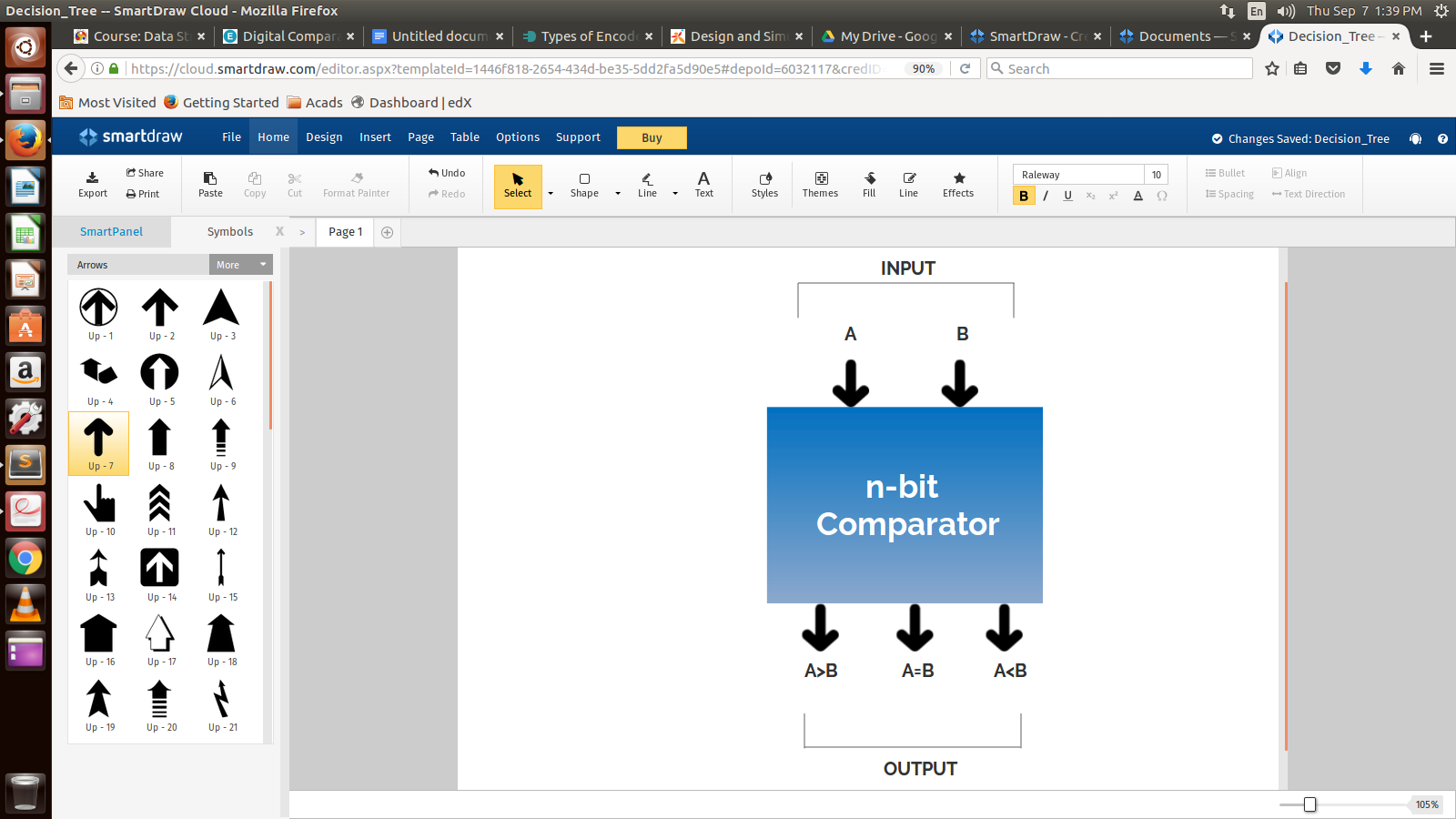
INTRODUCTION

Data comparison is needed in digital systems while performing arithmetic or logical operations. This comparison determines whether one number is greater than, equal, or less than the other number. A digital comparator is widely used in combinational system and is specially designed to compare the relative magnitudes of binary numbers.

Digital comparators can be of two types:

**Identity Comparator:** Comparators that have only one output terminal and produces the output either low or high are identity comparators.

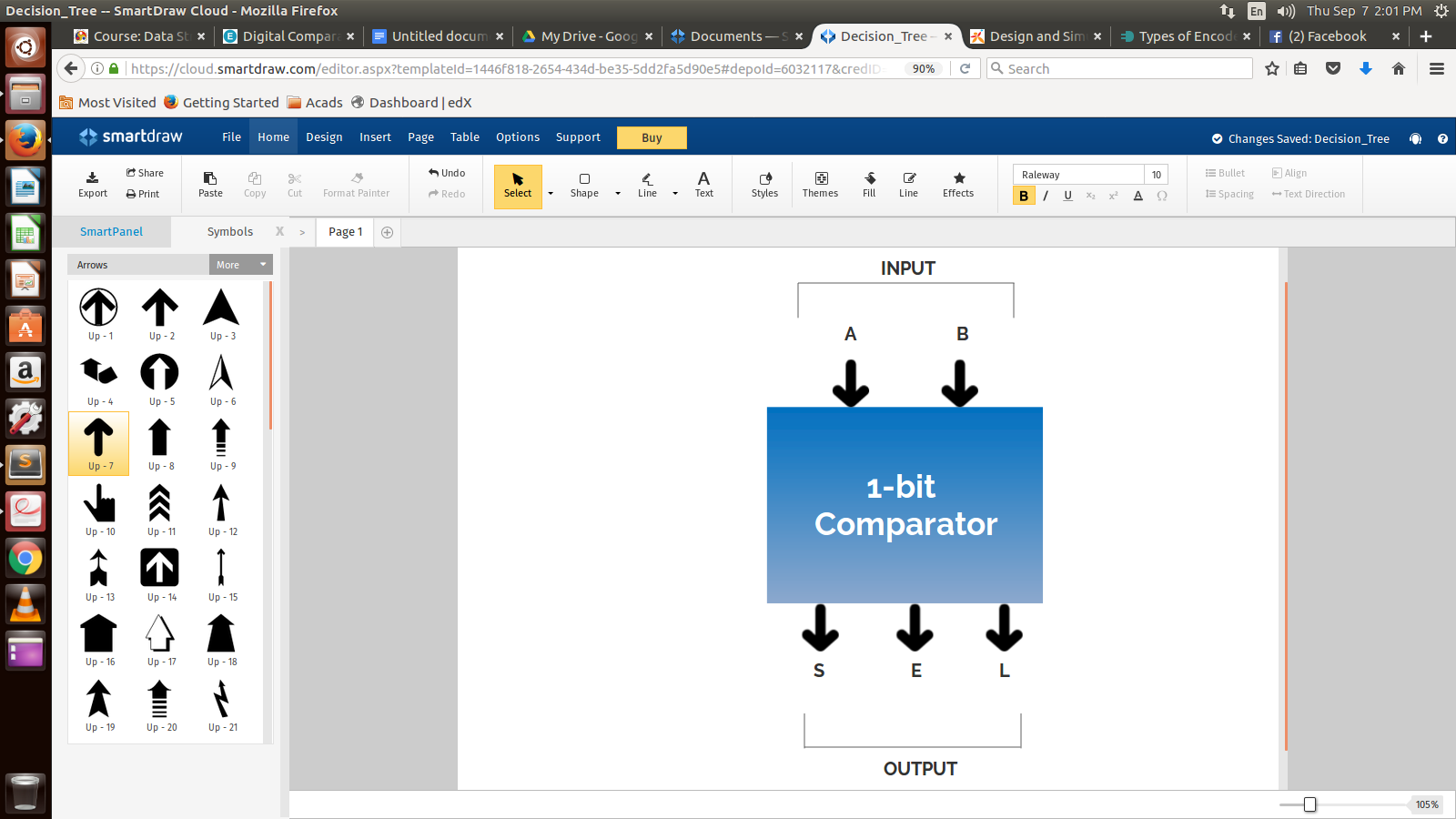
**Magnitude Comparator:** Comparators with three output terminals and checks for three conditions i.e greater than or less than or equal to is magnitude comparator.

Magnitude Digital Comparator

A magnitude digital comparator is a combinational circuit that compares two digital or binary numbers (consider A and B) and determines their relative magnitudes in order to find out whether one number is equal, less than or greater than the other digital number.

Three binary variables are used to indicate the outcome of the comparison as A>B, A<B, or A=B. The figure on right side shows the block diagram of a n-bit comparator which compares the two numbers of n-bit length and generates their relation between themselves.

**1-bit comparator**



A comparator used to compare two bits, i.e., two numbers each of single bit is called a single bit comparator. It consists of two inputs for allowing two single bit numbers and three outputs to generate less than, equal and greater than comparison outputs.

The figure on the right shows the block diagram of a single bit magnitude comparator. This comparator compares the two bits and produces one of the 3 outputs as S (A<B), E (A=B) and L (A>B).

The truth table for the single bit comparator is given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A0 | B0 | S | E | L |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

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

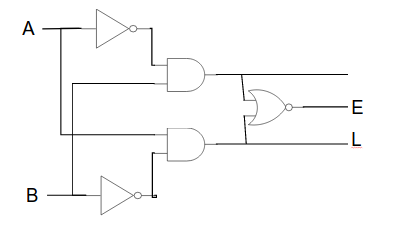
A0 < B0 : S = B0

A0 = B0 : E = + A0 B0

A0 > B0 : L = A0

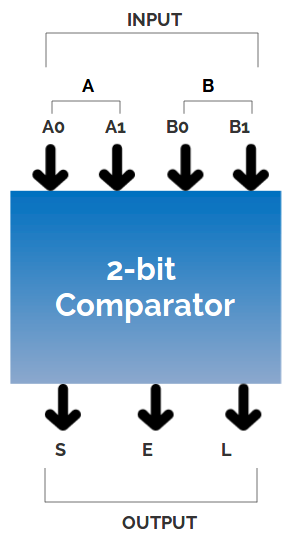
It is to be noted that E can be realized as .

The Logic gate implementation for the same is given below:



Now moving on to the main topic that is explaining 2-bit and 4-bit comparator using K-map

2-bit comparator



A 2-bit comparator compares two binary numbers, each of two bits and produces their relation such as one number is equal or greater than or less than the other. The figure on the right shows the block diagram of a two-bit comparator which has four inputs and three outputs.

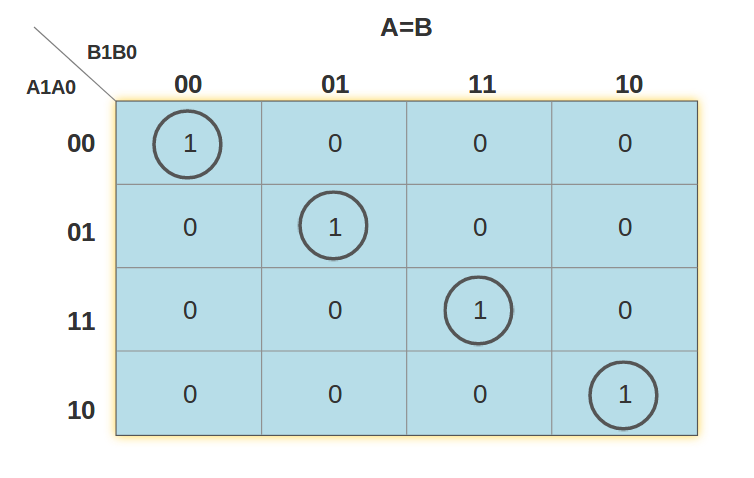
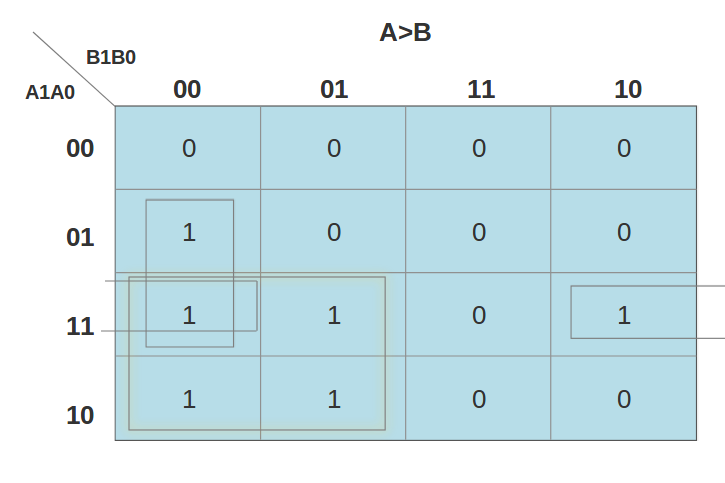
The first number A is designated as A = A1A0 and the second number is designated as B = B1B0. This comparator produces three outputs as L (L = 1 if A>B), E (E = 1, if A = B) and S (S = 1 if A<B).

The truth table of this comparator is shown below which depicting various input and output states.

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

Truth Table for a 2-bit Comparator

Simplifying above table using K-map :



A>B : L = A0+ A1+ A1 A0

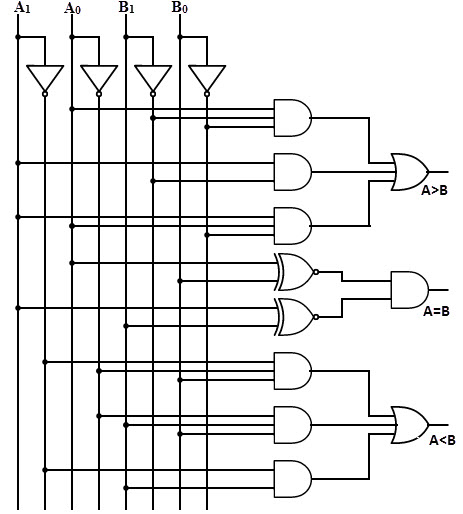
A = B : E = + A0 B0 + A1 A0 B1 B0 + A1 B1

=(+ A0 B0) + A1 B1 (A0 B0 + )

= (A0 B0 + )(A1 B1 + )

= (A0 Ex-NOR B0)(A1 Ex-NOR B1)

A<B : S = B1 +B1 B0 + B0



Logic Diagram of a 2-bit Comparator

4-bit comparator

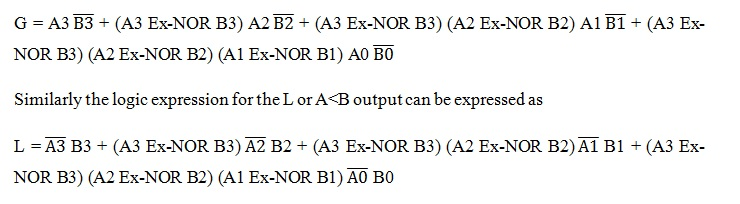
It can be used to compare two four-bit words. The two 4-bit numbers are A = A3 A2 A1 A0 and B3 B2 B1 B0 where A3 and B3 are the most significant bits.

It compares each of these bits in one number with bits in that of other number and produces one of the following outputs as A = B, A < B and A>B.

The output logic statements of this converter are

* If A3 = 1 and B3 = 0, then A is greater than B (A>B). Or
* If A3 and B3 are equal, and if A2 = 1 and B2 = 0, then A > B. Or
* If A3 and B3 are equal & A2 and B2 are equal, and if A1 = 1, and B1 = 0, then A>B. Or
* If A3 and B3 are equal, A2 and B2 are equal and A1 and B1 are equal, and if A0 = 1 and B0 = 0, then A > B.

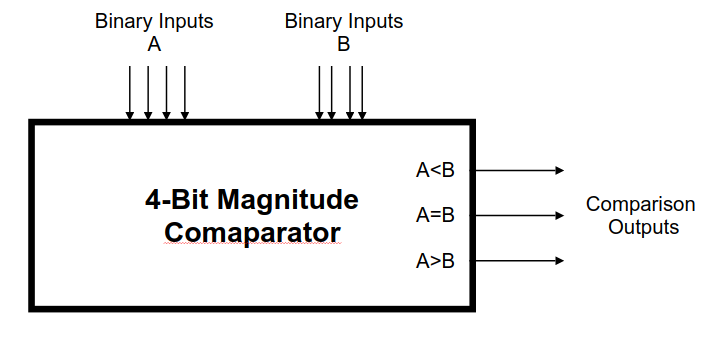
From the above statements, the output A > B logic expression can be written as



The equal output is produced when all the individual bits of one number are exactly coincides with corresponding bits of another number. Then the logical expression for A=B output can be written as

E = (A3 Ex-NOR B3) (A2 Ex-NOR B2) (A1 Ex-NOR B1) (A0 Ex-NOR B0)

We can represent the 4-bit Comparator simply by this block diagram shown below



Multiplexer

INTRODUCTION

Multiplexing is the property of combining one or more signals and transmitting on a single channel .This is achieved by the device multiplexer. A multiplexer is the most frequently used combinational circuits and important building block in many in digital systems.

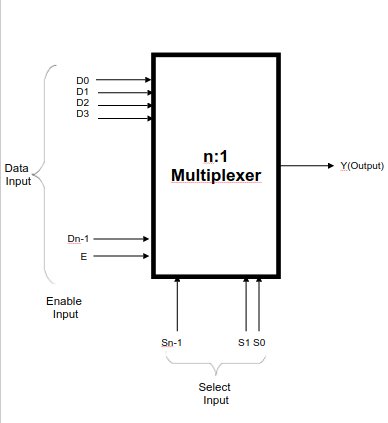
These are mostly used to form a selected path between multiple sources and a single destination. A basic multiplexer has various data input lines and a single output line. These are found in many digital system applications such as data selection and data routing, logic function generators, digital counters with multiplexed displays, telephone network, communication systems, waveform generators, etc. In this article we are going to discuss about types of multiplexers and its design.

#### What is a Multiplexer?

The multiplexer or MUX is a digital switch, also called as data selector. It is a combinational circuit with more than one input line, one output line and more than one select line. It allows the binary information from several input lines or sources and depending on the set of select lines , particular input line , is routed onto a single output line.

The basic idea of multiplexing is that data from several sources are routed to the single output line when the enable switch is ON. That is how the multiplexers are also called as ‘many to one’ combinational circuits.

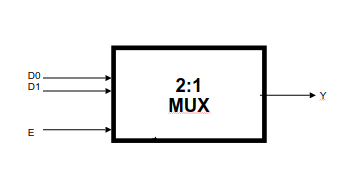
Generally the number of data inputs to a multiplexer is a power of two such as 2, 4, 8, 16, etc. Some of the mostly used multiplexers include 2-to-1, 4-to-1, 8-to-1 and 16-to-1 multiplexers.



Block Diagram of n:1 Multiplexer

### 2-to-1 Multiplexer

A 2-to-1 multiplexer consists of two inputs D0 and D1, one select input S and one output Y. Depends on the select signal, the output is connected to either of the inputs. Since there are two input signals only two ways are possible to connect the inputs to the outputs, so one select is needed to do these operations.

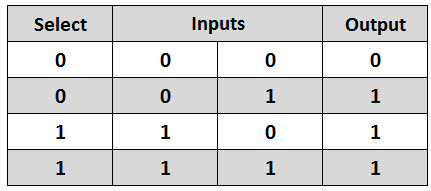


If the select line is low, then the output will be switched to D0 input, whereas if select line is high, then the output will be switched to D1 input. The figure on the right shows the block diagram of a 2-to-1 multiplexer which connects two 1-bit inputs to a common destination.

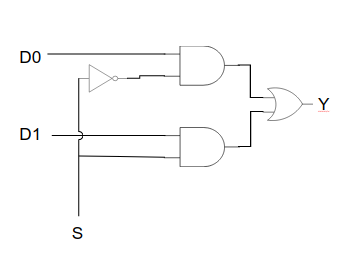
The truth table of the 2-to-1 multiplexer is shown below. Depending on the selector switching the inputs are produced at outputs , i.e., D0 , D1 and are switched to the output for S=0 and S=1 respectively . Thus, the Boolean expression for the output becomes D0 when S=0 and output is D1 when S=1.

From the truth table the Boolean expression of the output is given as

**Y = D0 + D1 S**



From the above output expression, the logic circuit of 2-to-1 multiplexer can be implemented using logic gates as shown in figure.

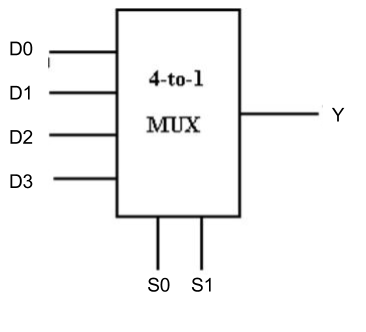


2 to 1 mux logic diagram

### 4-to-1 Multiplexer

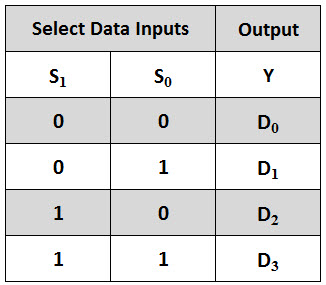
A 4-to-1 multiplexer consists four data input lines as D0 to D3, two select lines as S0 and S1 and a single output line Y. The select lines S1 and S2 select one of the four input lines to connect the output line. The particular input combination on select lines selects one of input (D0 through D3) to the output.

The figure below shows the block diagram of a 4-to-1 multiplexer in which the multiplexer decodes the input through select line.



4:1 Multiplexer Block Diagram

The truth table of a 4-to-1 multiplexer is shown below in which four input combinations 00, 10, 01 and 11 on the select lines respectively switches the inputs D0, D2, D1 and D3 to the output. That means when S1=0 and S0 =0, the output at Y is D0, similarly Y is D1 if the select inputs S1=0 and S0= 1 and so on.

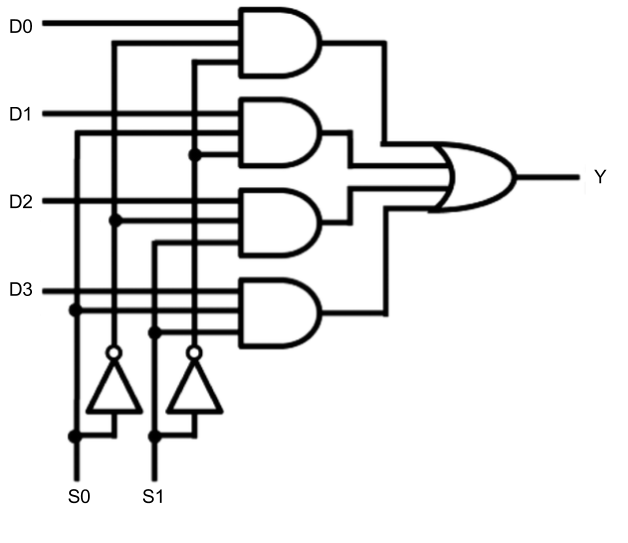


From the above truth table, final Boolean expression of this multiplexer can be given as

**Y = D1 + D1S0 + D2 S1 + D3 S1 S0**

From the above expression of the output, a 4-to-1 multiplexer can be implemented by using basic logic gates.

The below figure shows the logic circuit of 4:1 MUX which is implemented by four 3-inputs AND gates, two 1-input NOT gates, and one 4-inputs OR gate.



4 to 1 mux logic diagram

### 

### 

### 

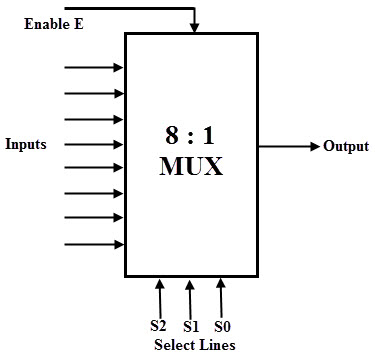
### 

### 

### 8-to-1 Multiplexer

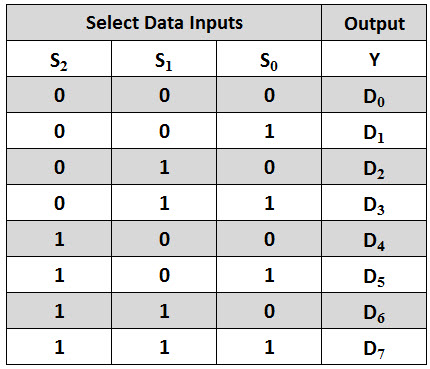
An 8-to-1 multiplexer consists of eight data inputs D0 through D7, three input select lines S2 through S0 and a single output line Y. Depending on the select lines combinations, multiplexer decodes the inputs.

The below figure shows the block diagram of an 8-to-1 multiplexer with enable input that enable or disable the multiplexer.



8:1 Multiplexer Block Diagram

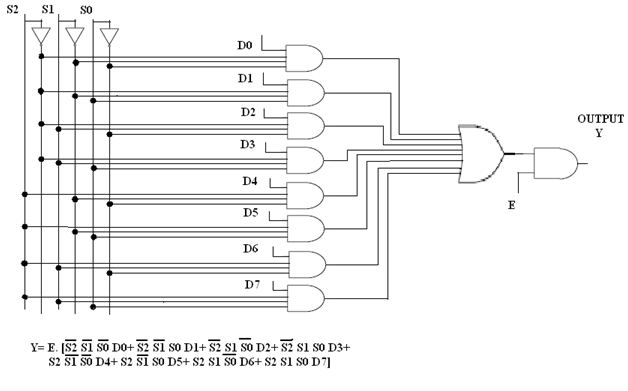
The truth table for an 8-to1 multiplexer is given below with eight combinations of inputs so as to generate each output corresponds to input.



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

****

From the above Boolean equation, the logic circuit diagram of an 8-to-1 multiplexer can be implemented by using 8 AND gates, 1 OR gate and 7 NOT gates as shown in below figure.



8 to 1 mux logic diagram

### Application of Multiplexer

In all types of digital system applications, multiplexers find its immense usage. Since these allows multiple inputs to be connected independently to a single output, these are found in variety of applications including data routing, logic function generators, control sequencers, parallel-to-serial converters, etc.

We will not go in discussion of these applications here.