COE 272
Digital Systems

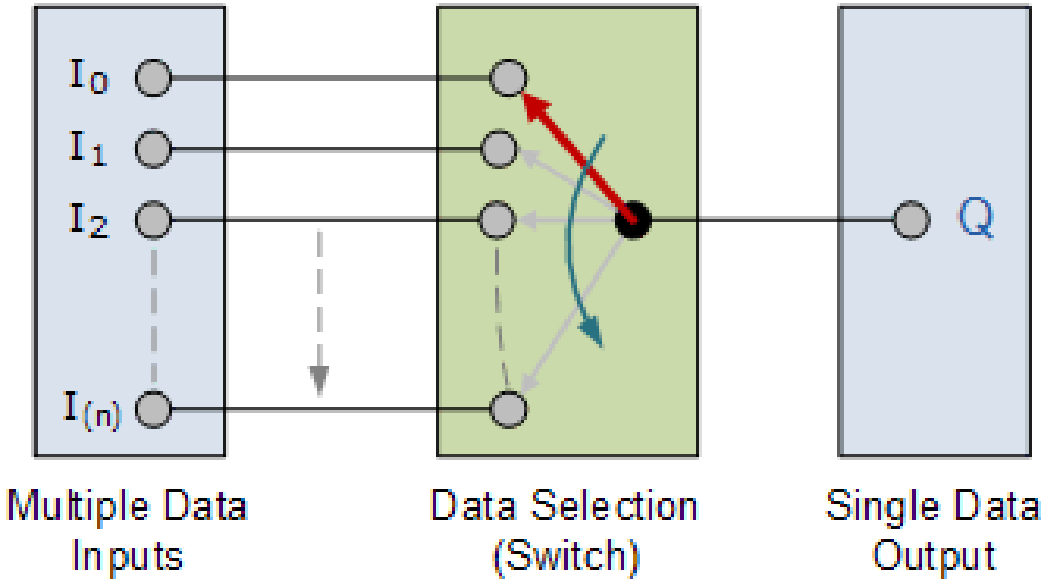
Combinational Circuits Pt.2

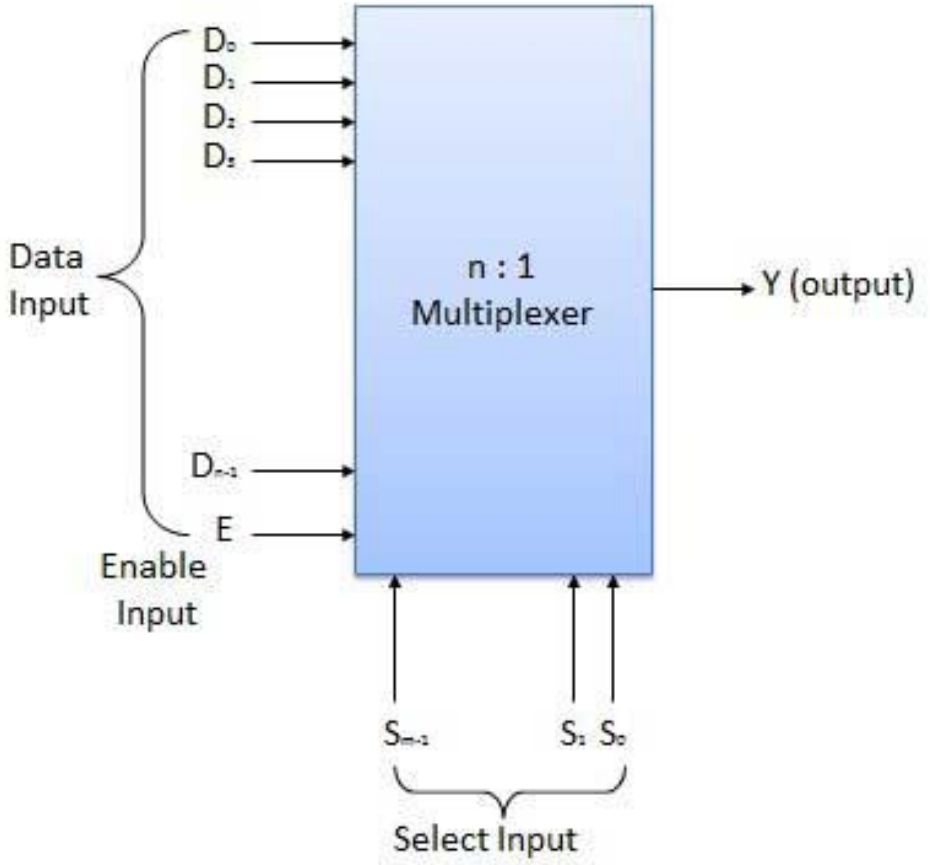
(De/Multiplexers and En/Decoders)

Multiplexer

- A multiplexer is a special type of combinational circuit.
- It selects one out of n data inputs and routes it to the output
- To select n inputs, we require m select lines such that $2^m = n$.
- Depending on the digital code applied at the select inputs, one out of the n data sources is selected and transmitted to the single output Y.

Multiplexer





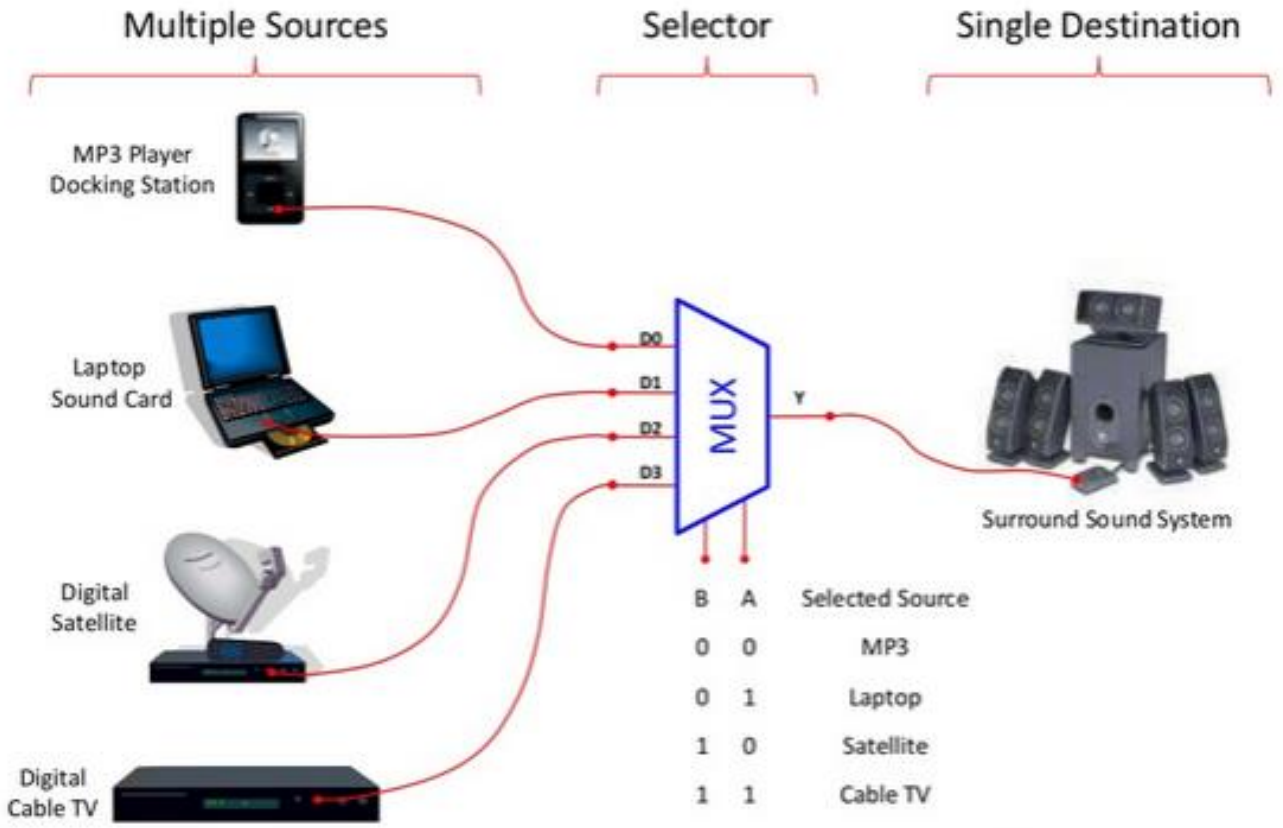
Necessity of multiplexers

- Mostly found in electronic systems, digital data is available on more than one line.
- It is essential to route this data over a single line
- Under such circumstances a circuit is needed to implement this function; the multiplexer is such a circuit.
- Multiplexers improve reliability of digital systems because it reduces the number of external wired connections

Advantages of multiplexers

- The use of multiplexers provide the following advantages:
  - It reduces the number of wires
  - Reduces the circuit complexity and cost
  - It simplifies the logic design
  - It does not need K-Maps and simplification

Typical application of MUX

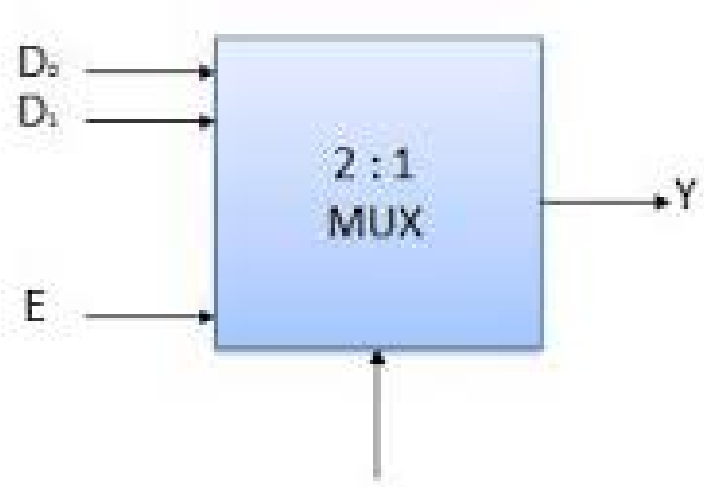


Classification of multiplexers

- Multiplexers are classified as under:
  - 2 : 1 multiplexer
  - 4 : 1 multiplexer
  - 8 : 1 multiplexer
  - 16 : 1 multiplexer

2 : 1 multiplexer

- It has two data inputs Do and D1.
- It has one select input S, an enable input and one output



Enable	Select i/p (S)	Output Y
0	X	0
1	0	Do
1	1	D1

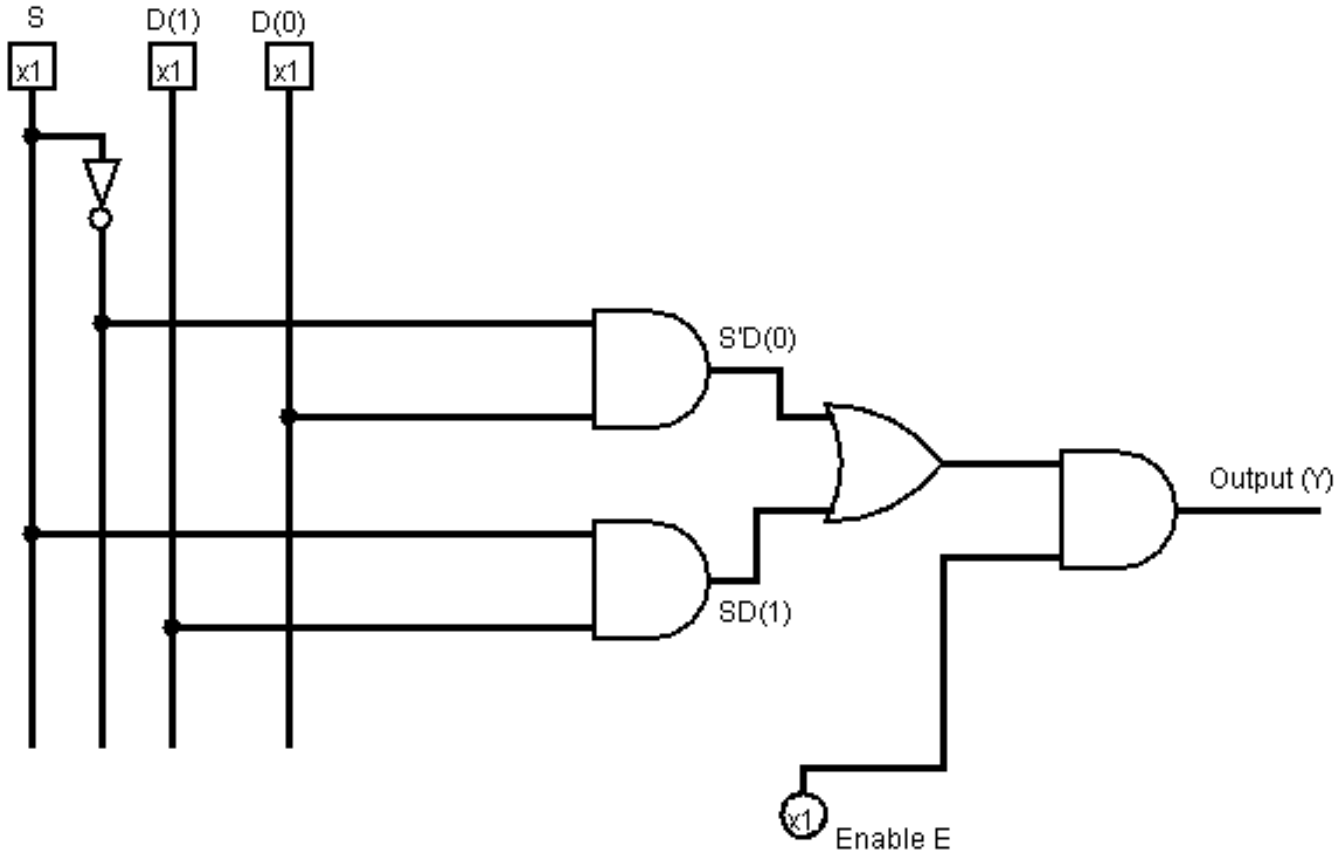
Truth table for 2 : 1 Mux

2 : 1 multiplexer

Enable E	Select S	D_1	Do	Output Y	
0	X	X	X	0	
1	0	X	0	0	
1	0	X	1	1	← $Y = ES'Do$
1	1	0	X	0	
1	1	1	X	1	← $Y = ESD_1$

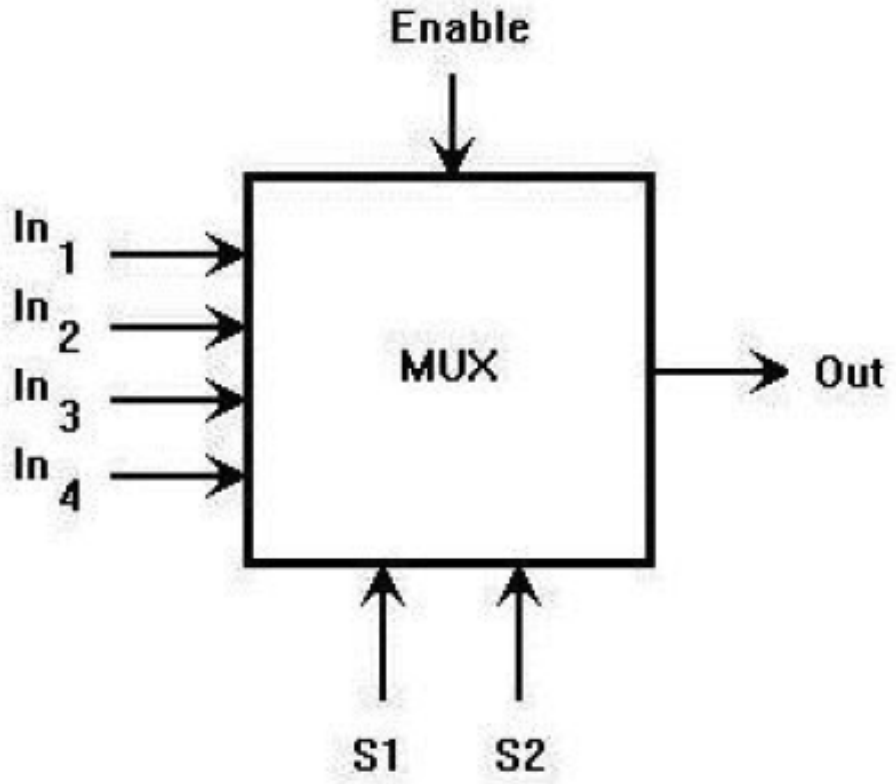
$$Y = ES'Do + ESD_1$$

2 : 1 multiplexer



4 : 1 multiplexer

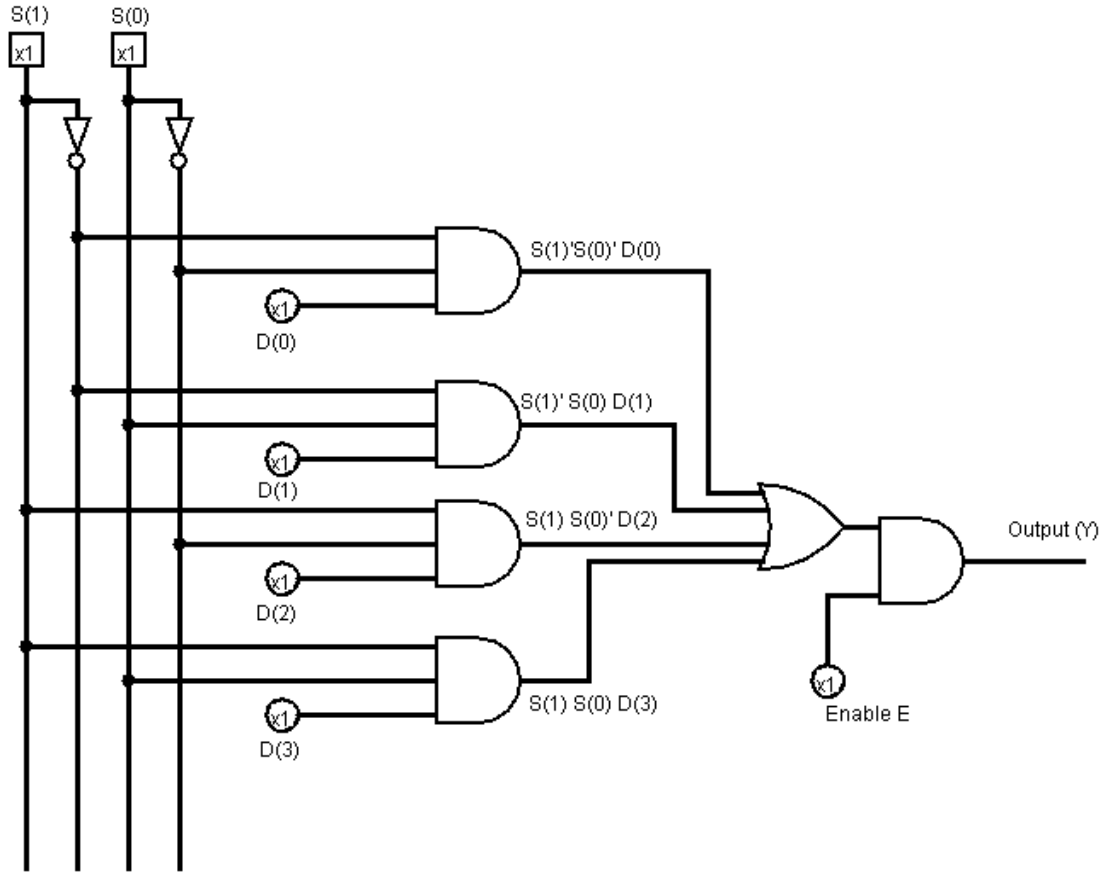
- It has four data input lines, two select lines and one output line



4 : 1 multiplexer

Select Inputs		Output
S1	S0	
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

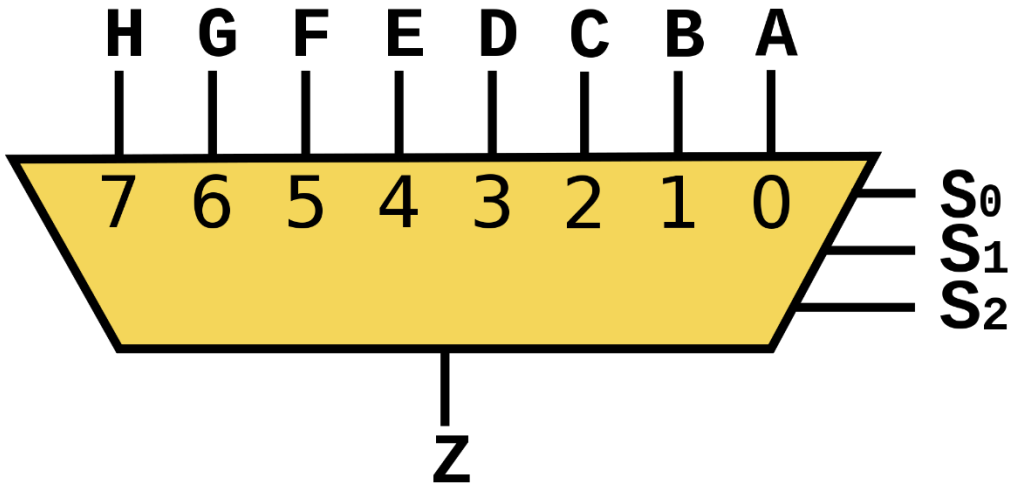
4 : 1 multiplexer



$$Y = \bar{S}_1 \bar{S}_0 D_0 + \bar{S}_1 S_0 D_1 + S_1 \bar{S}_0 D_2 + S_1 S_0 D_3$$

8 : 1 multiplexer

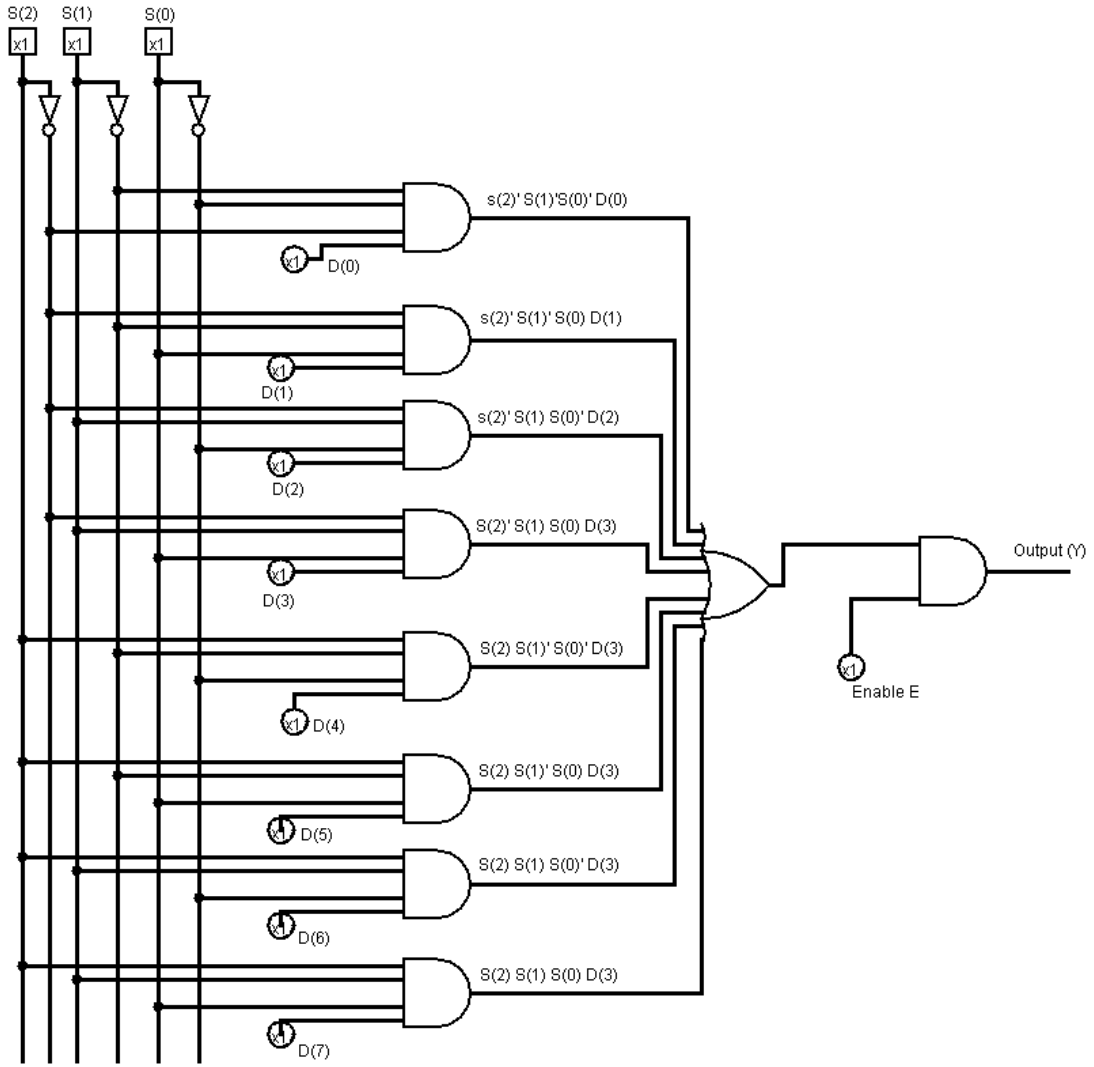
- It has eight data inputs, one enable input, three select inputs and one output



8 : 1 multiplexer

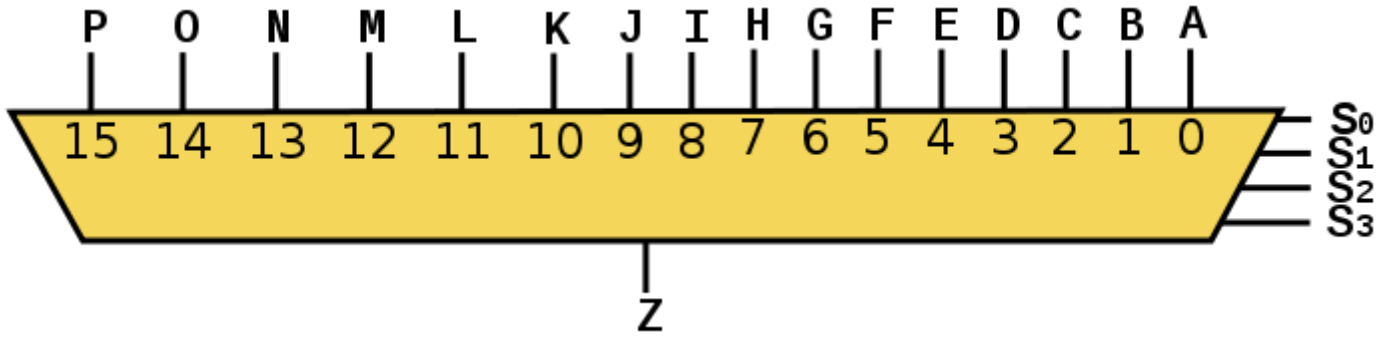
Enable E	Select Inputs			Output (Y) Y
	S_2	S_1	S_0	
0	X	X	X	0
1	0	0	0	D_0
1	0	0	1	D_1
1	0	1	0	D_2
1	0	1	1	D3
1	1	0	0	D_4
1	1	0	1	D_5
1	1	1	0	D_6
1	1	1	1	D_7

8 : 1 multiplexer



16 : 1 multiplexer

- When the enable input is 0 the multiplexer outputs a 0
- When E = 1, we can select one of the 16 inputs with the help of the select input lines



16 : 1 multiplexer

Enable E	Select Input				Output Y
	S3	S2	S1	S0	
0	0	0	0	0	D0
1	0	0	0	1	D1
1	0	0	1	0	D2
1	0	0	1	1	D3
1	0	1	0	0	D4
1	0	1	0	1	D5
1	0	1	1	0	D6
1	0	1	1	1	D7
1	1	0	0	0	D8
1	1	0	0	1	D9
1	1	0	1	0	D10
1	1	0	1	1	D11
1	1	1	0	0	D12
1	1	1	0	1	D13
1	1	1	1	0	D14
1	1	1	1	1	D15

Applications of multiplexers

- The following are some application areas of MUX
  - It can be used as a data selector to select one out of many inputs
  - It can be used for simplification of logic design
  - In designing combinational circuits
  - In D/A converters
  - To minimize number of connections

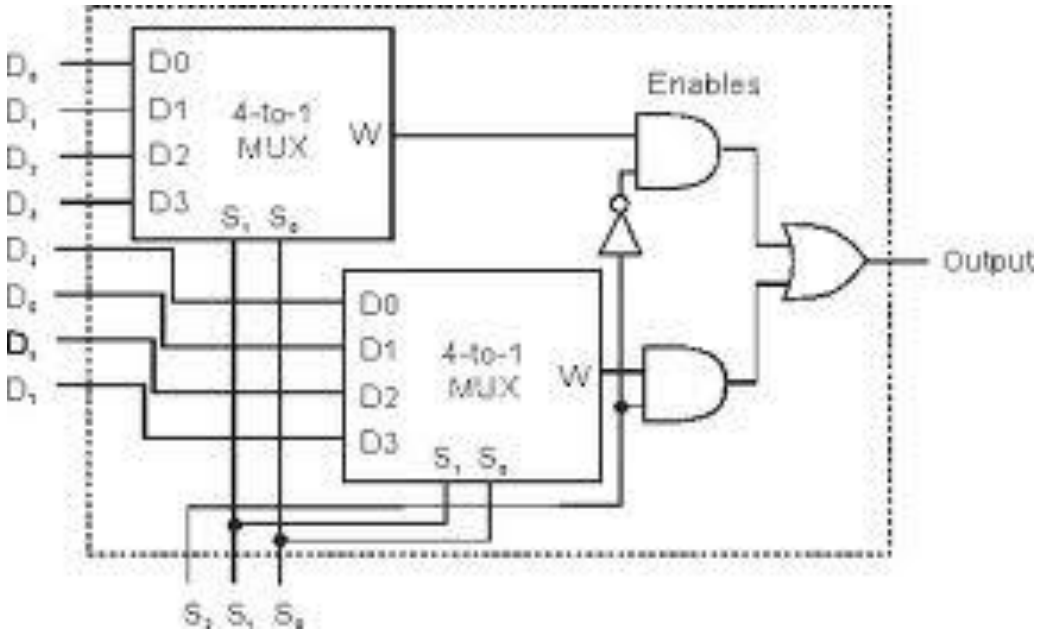
Multiplexer Tree

- Multiplexers having more number of inputs can be obtained by cascading two or more MUX with less number of inputs.

- This phenomenon is known as **multiplexer tree**

Example

- Obtain an 8 : 1 MUX using 4 : 1 MUX



Select Lines			Output Y
S2	S1	S0	
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

Practice Example

- Implement a 4 : 1 MUX using 2 : 1 MUX
- Obtain an 16 : 1 MUX using 4 : 1 MUX
- Obtain an 16 : 1 MUX using 8 : 1 MUX
- Draw a 32 : 1 MUX tree using 4 : 1 MUX

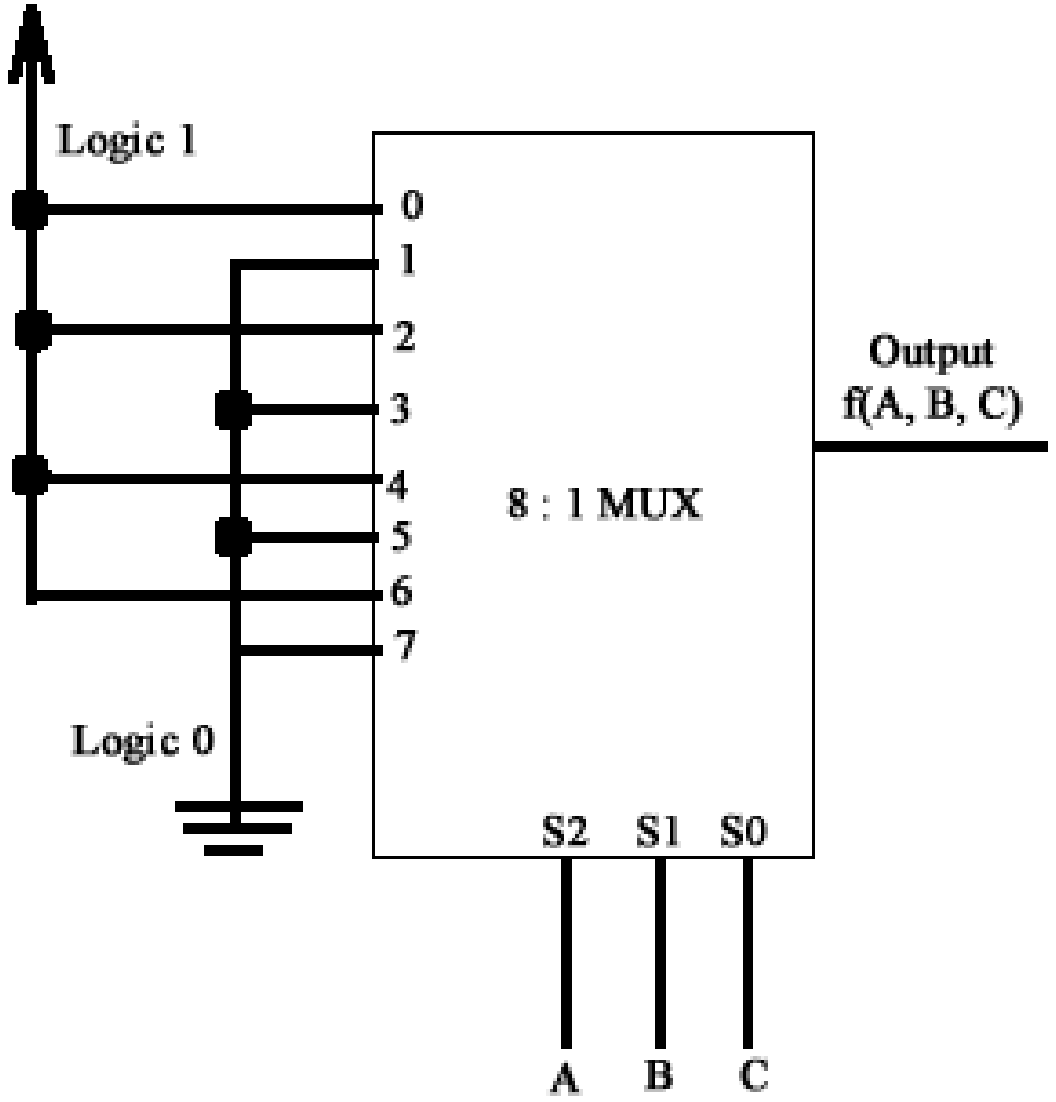
Combinational Logic using MUX

- The design procedure is as follows:
  - First, identify the decimal number corresponding to each minterm/maxterm in the expression
  - The input lines of the MUX, corresponding to these numbers are connected to logic 1 level.
  - All other inputs of the MUX are connected to logic 0 level.
  - The input (literals) are connected to the select inputs.

Example

- Implement the following expression using a MUX:
  - F(A, B, C) = $\sum m\,(0, 2, 4, 6)$

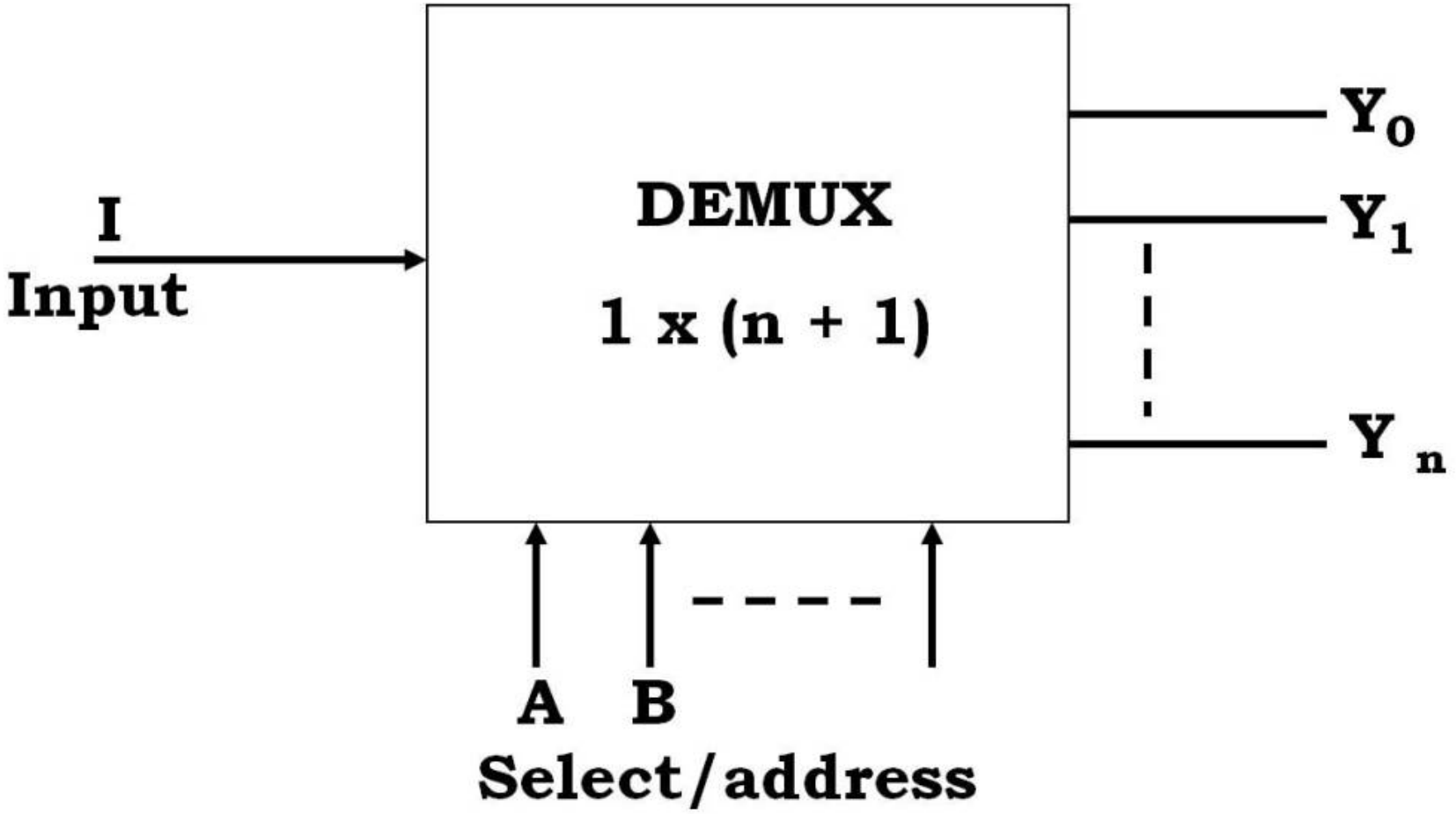
Solution



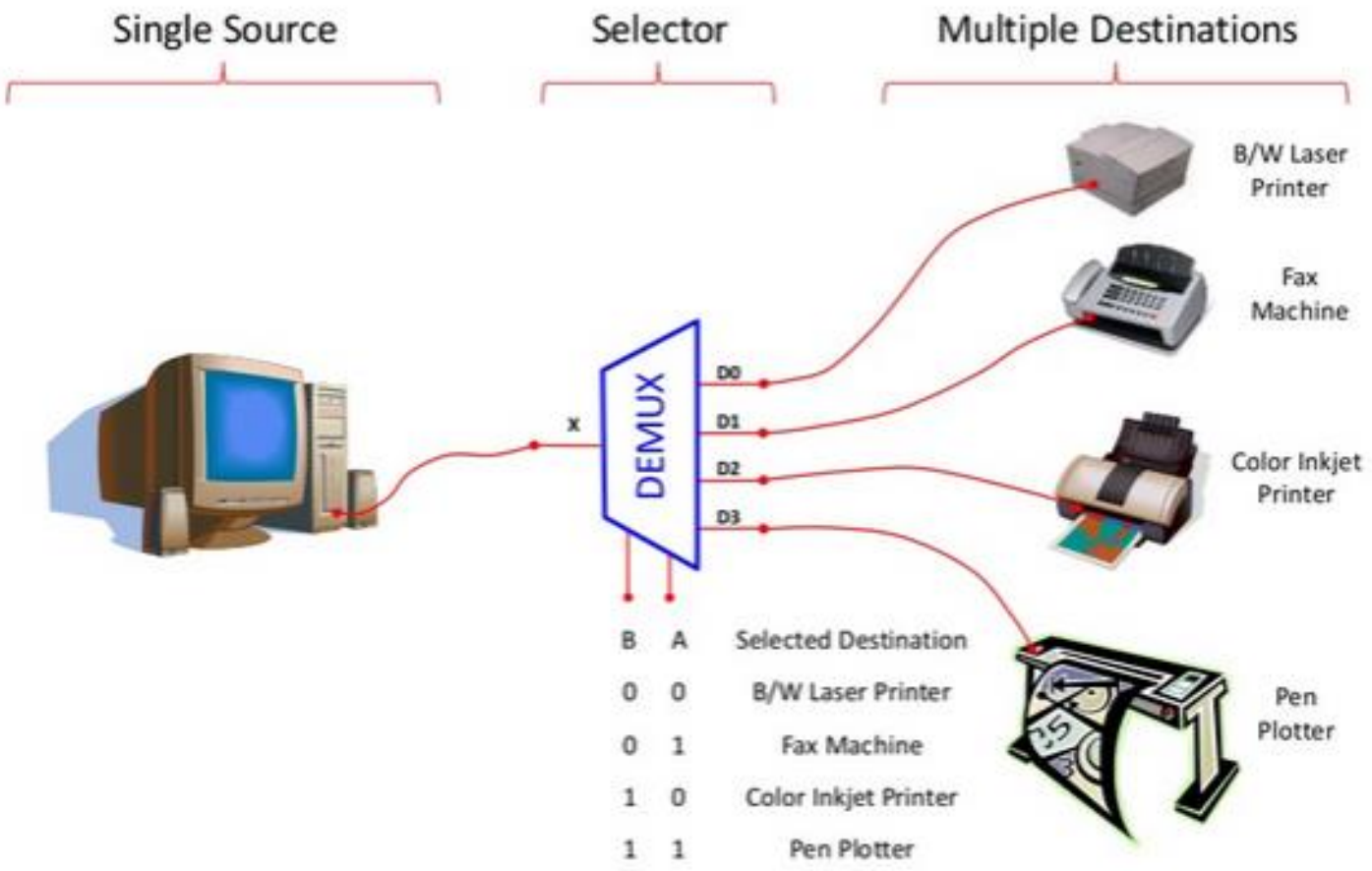
Demultiplexers

- Demultiplexers are used at the other end of the data path to separate signals that are sharing a data path and distribute them to their respective destinations
  - *A demultiplexer performs the reverse operation of a multiplexer*
- At a time, only one output line is selected by the select lines and the input is transmitted to the select output line.
- The Enable (E) enables the demultiplexer

DEMUX



Typical Application of a DEMUX



Classification of Demultiplexers

- 1 : 2 demultiplexer
- 1 : 4 demultiplexer
- 1 : 8 demultiplexer
- 1 : 16 demultiplexer

1 : 2 demultiplexer

- It has one data input Din, one select output So, one enable (E) input, and two outputs Y0 and Y1

Enable E	Select So	Output Y1	Y2
0	X	0	0
1	0	0	Din
1	1	Din	0

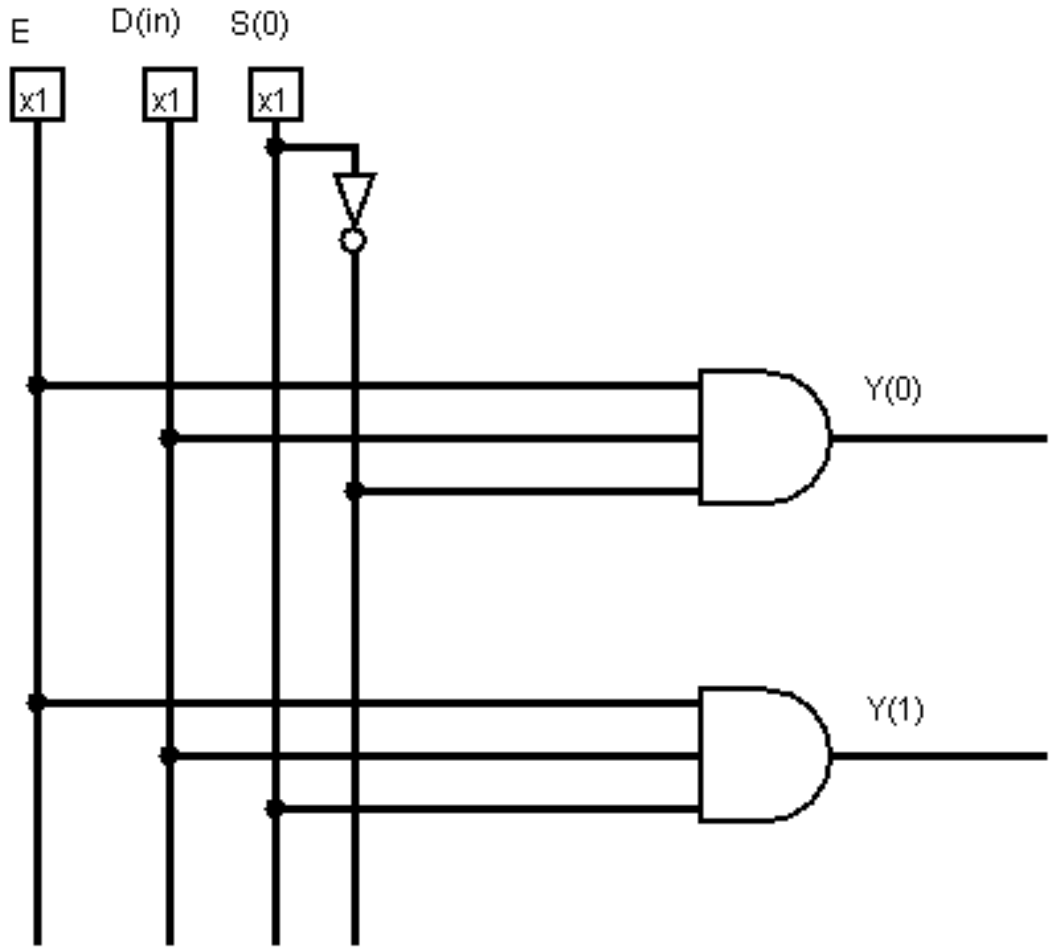
1 : 2 demultiplexer

- Realization of 1 : 2 DEMUX

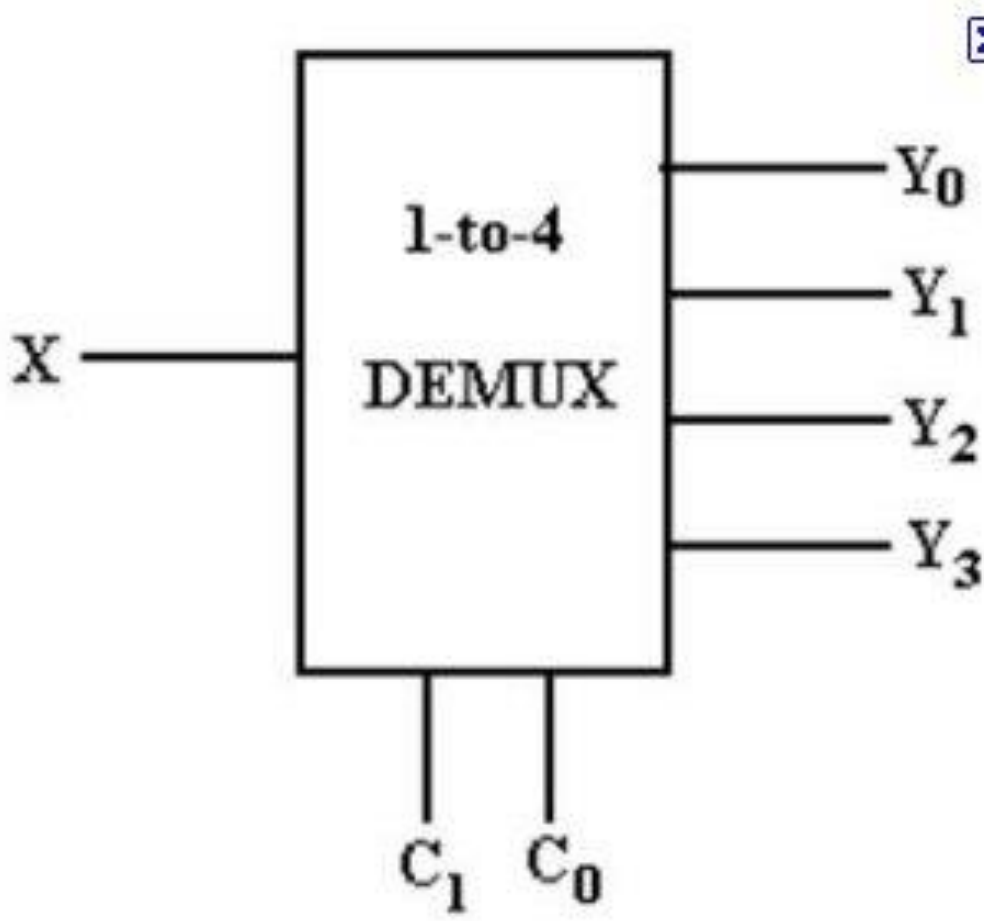
Enable E	Data Din	Select So	Output Y_1	Y_0
0	X	X	0	0
1	0	0	0	0
1	1	0	0	1
1	0	1	0	0
1	1	1	1	0

Truth table for 1 : 2 demux

1 : 2 demultiplexer



1 : 4 demultiplexer



1 : 4 demultiplexer

- Realization of a 1 : 4 demultiplexer

E	Din	Select		Output			
E	Din	S1	S0	Y0	Y1	Y2	Y3
1	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0
1	0	0	1	0	0	0	0
1	1	0	1	0	1	0	0
1	0	1	0	0	0	0	0
1	1	1	0	0	0	1	0
1	0	1	1	0	0	0	0
1	1	1	1	0	0	0	1

Y0 is connected to Din

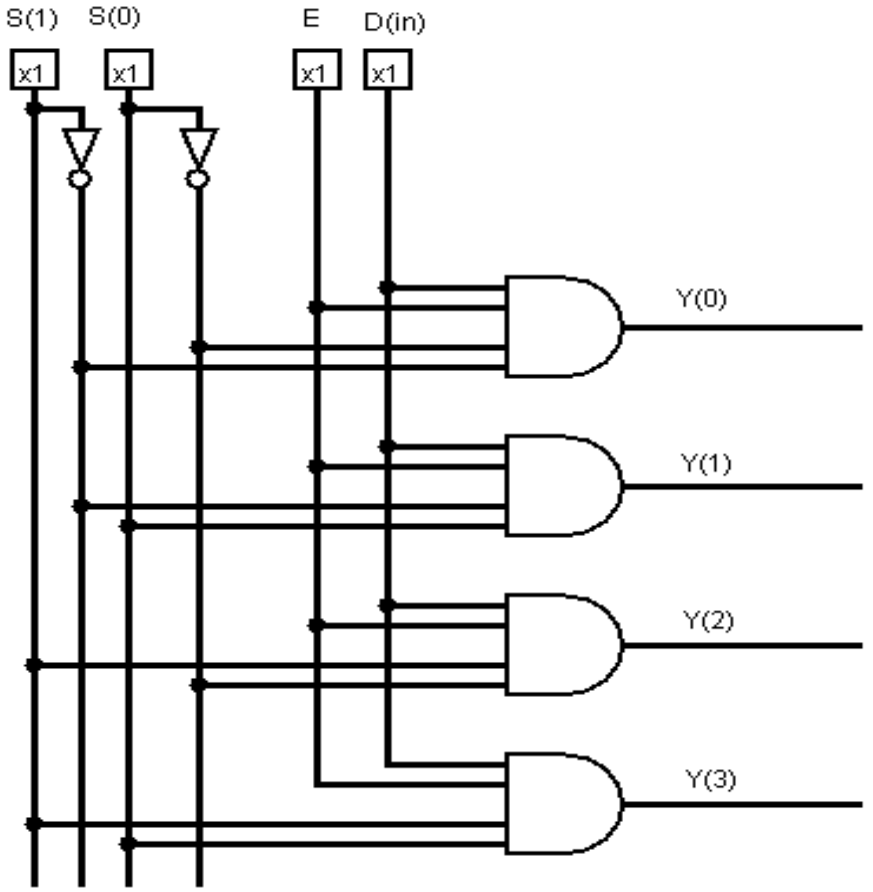
Y1 is connected to Din

Y2 is connected to Din

Y3 is connected to Din

Truth table for 1 : 4 demux

1 : 4 demultiplexer



1 : 8 demultiplexer

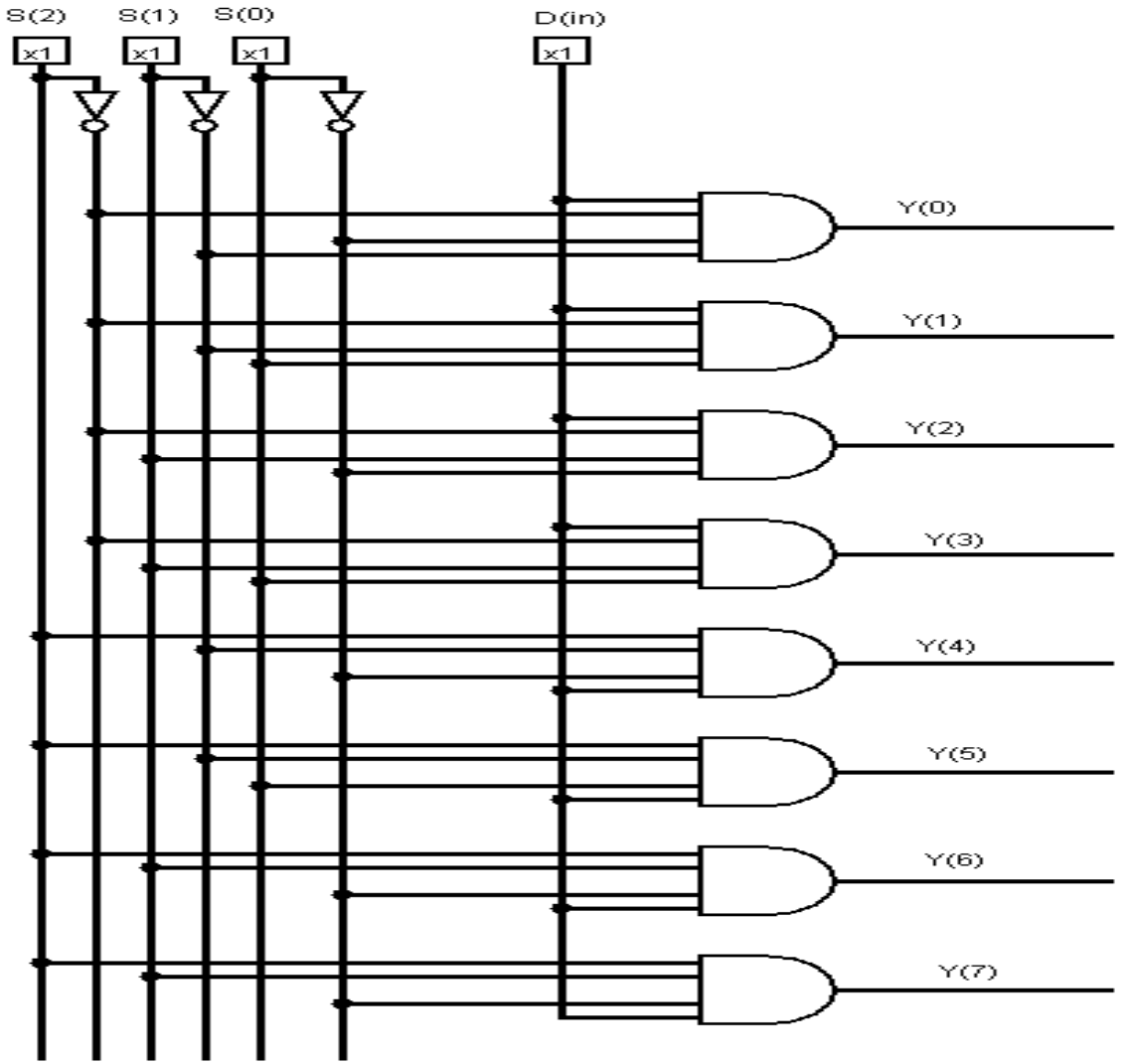
- It has one data input, eight data output, three select inputs and an enable input E.

1 : 8 demultiplexer

Enable E	Select			Outputs							
	S2	S1	S0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	Din
1	0	0	1	0	0	0	0	0	0	Din	0
1	0	1	0	0	0	0	0	0	Din	0	0
1	0	1	1	0	0	0	0	Din	0	0	0
1	1	0	0	0	0	0	Din	0	0	0	0
1	1	0	1	0	0	Din	0	0	0	0	0
1	1	1	0	0	Din	0	0	0	0	0	0
1	1	1	1	Din	0	0	0	0	0	0	0

Truth table for 1 : 8 demux

1 : 8 demultiplexer



1 : 16 demultiplexer

- It has one data input Din, four select lines S0, S1, S2, S3.

- It has sixteen outputs and one enable input

Exercise

- Write the truth table for the 1 : 16 DEMUX

- Implement the above truth table using logic gates

Demultiplexer Tree (Exercise)

- Obtain a 1 : 4 demultiplexer using 1 : 2 demultiplexer
- Obtain a 1 : 8 demultiplexer using two 1 : 4 demultiplexer
- Obtain a 1 : 32 demultiplexer using four 1 : 8 demultiplexer and one 1 : 4 demultiplexer

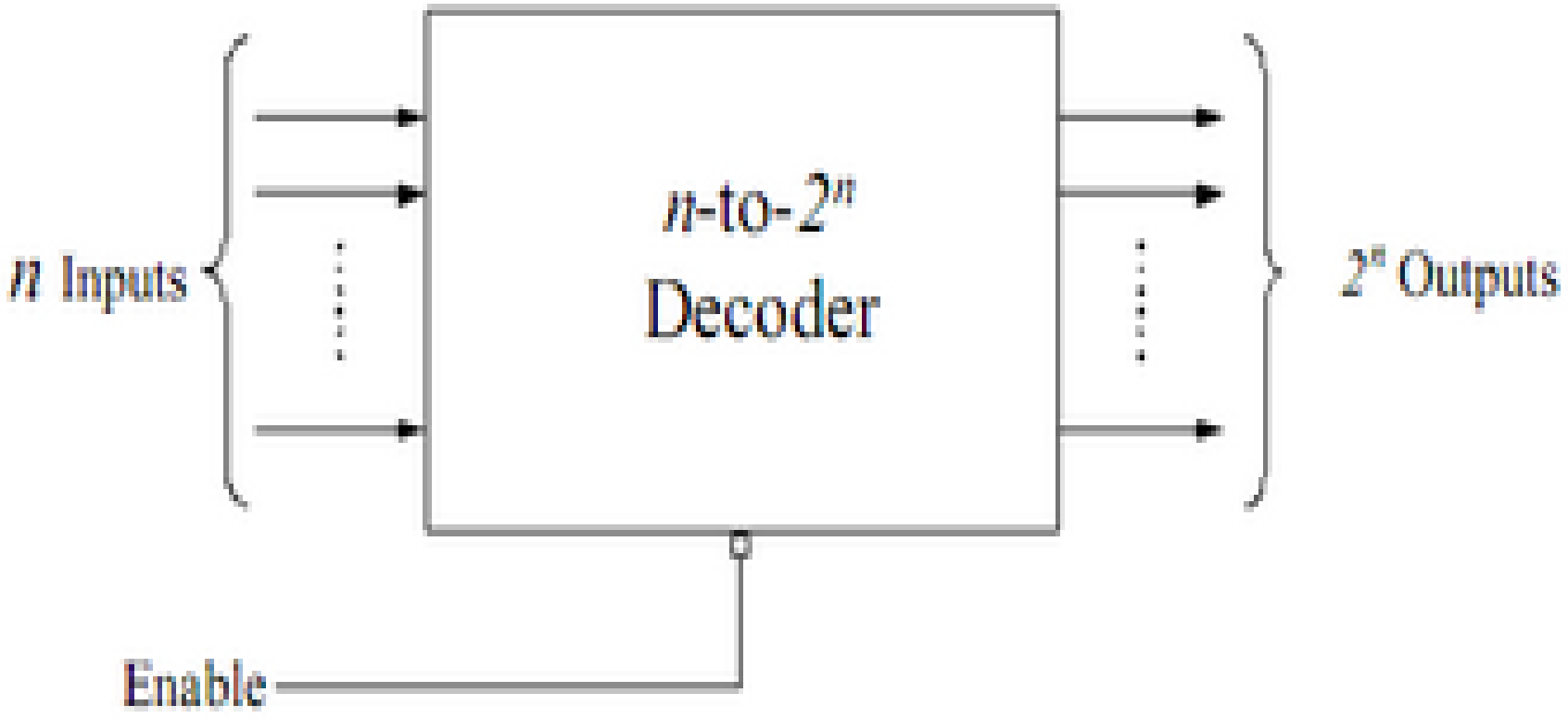
Performance Comparison between MUX and DEMUX

Parameter	Multiplexer	Demultiplexer
Type of Logic Circuit	Combinational	Combinational
Number of Data Inputs	n	1
Number of Select Inputs	m	m
Number of Data Output	1	n
Relation between input/output lines and select lines	$n = 2^m$	$n = 2^m$
Operation principle	Many to 1 or as data selector	1 to many or data distributor
Application	• As a universal logic circuit, i.e. we can implement any combinational circuit using a MUX • In time division multiplexing (TDM) at the sending end	• We can implement some combinational circuit • In TDM systems at the receiving end

Decoder

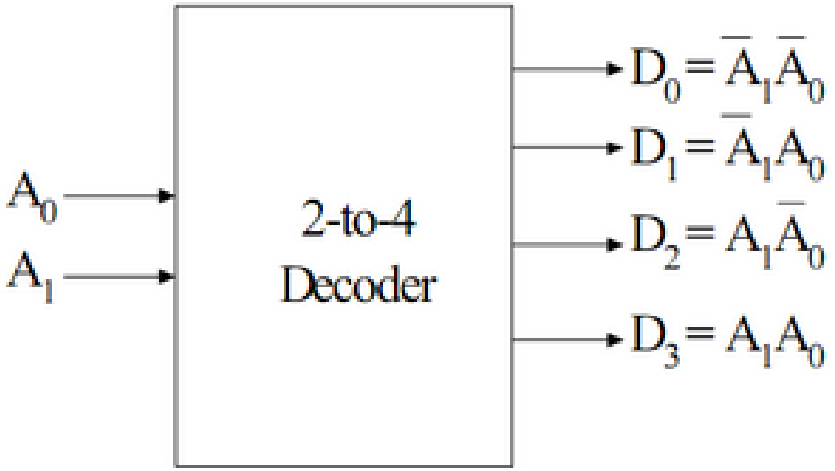
- A decoder is a combinational circuit
- It converts n-bit binary information at its input into a maximum of 2^n output lines.
  - E.g. if n = 2, then we can design up to a 2 : 4 decoder
- Decoders are widely used in the memory system of the computer where they respond to the address code generated by the central processor to activate a particular memory location

Decoder



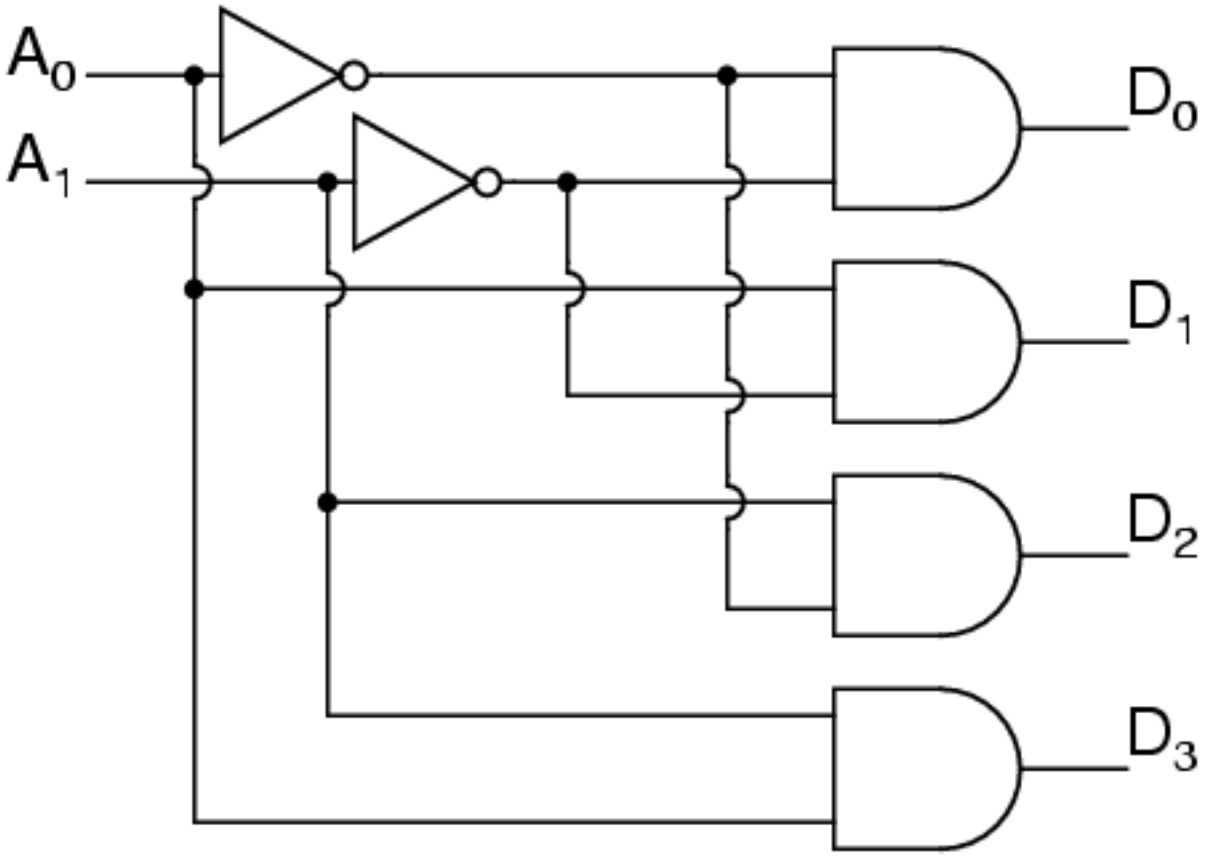
2 to 4 Line Decoder

- There are two inputs (A0 and A1) and 4 outputs (D0, D1, D2, D3).
- The truth table below explains the operation of the 2 to 4 line decoder

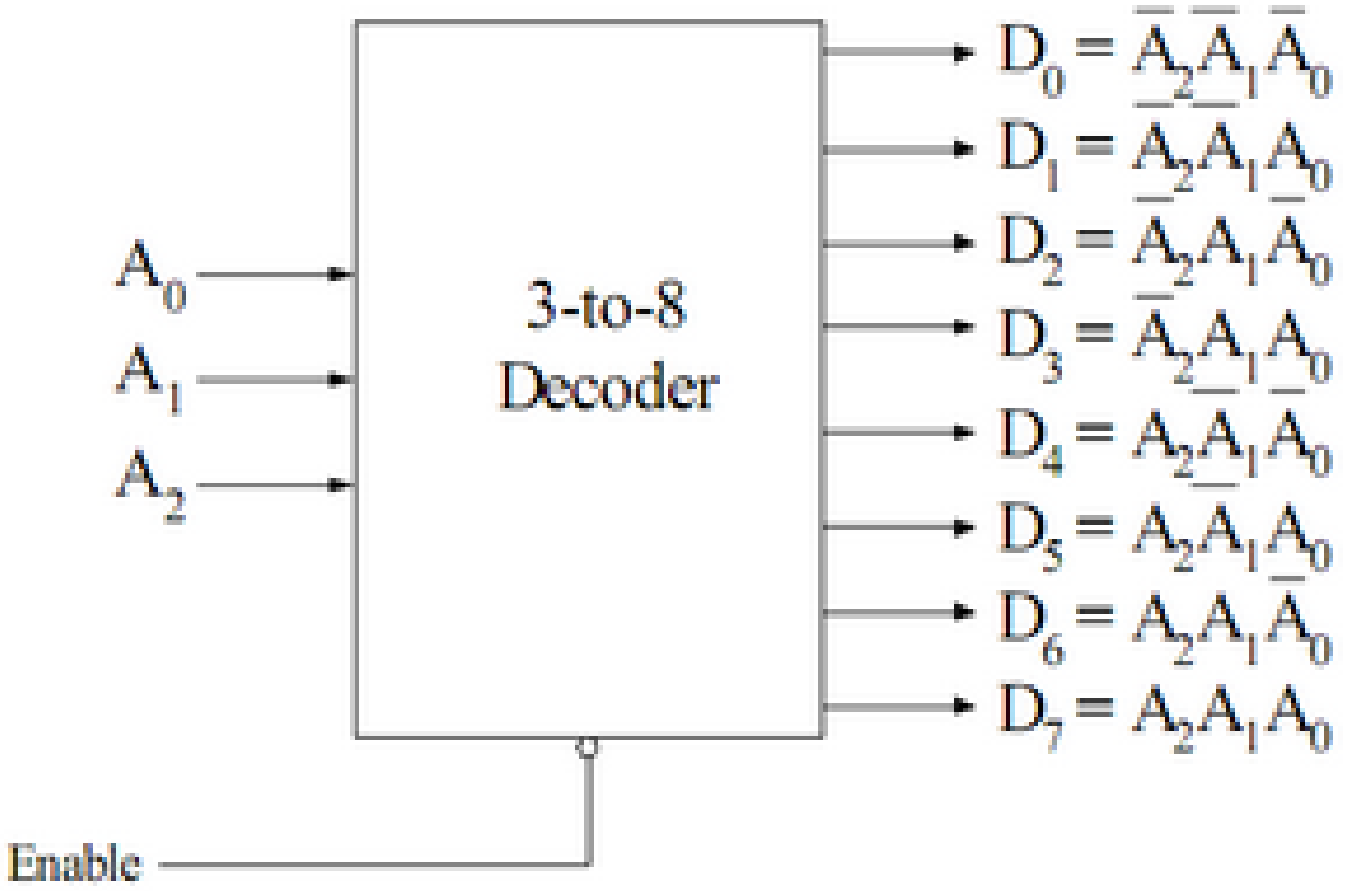


A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

2 to 4 Line Decoder



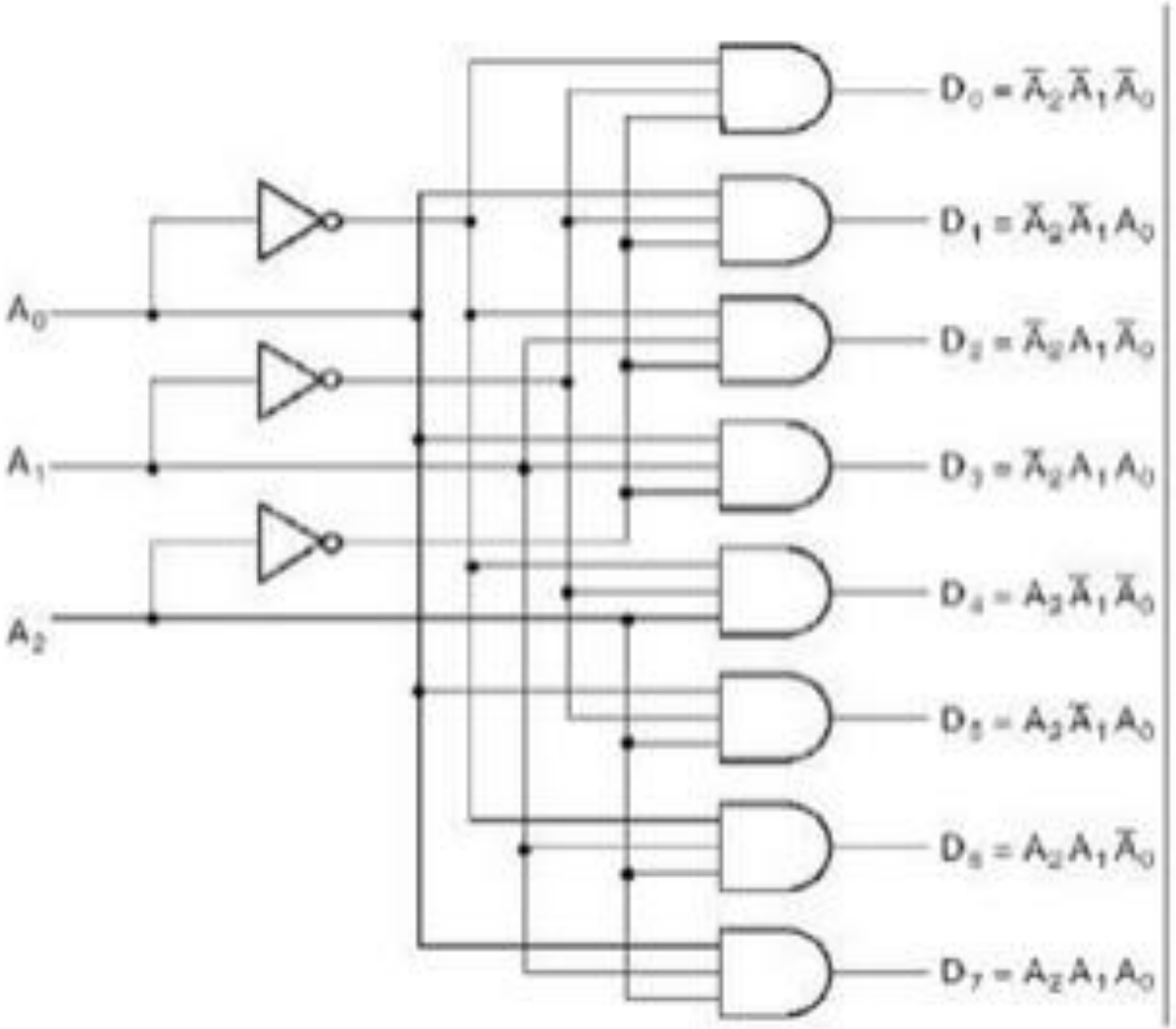
3 to 8 Line Decoder



3 to 8 Line Decoder

Dec. Code	Inputs			Outputs							
	A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

3 to 8 Line Decoder

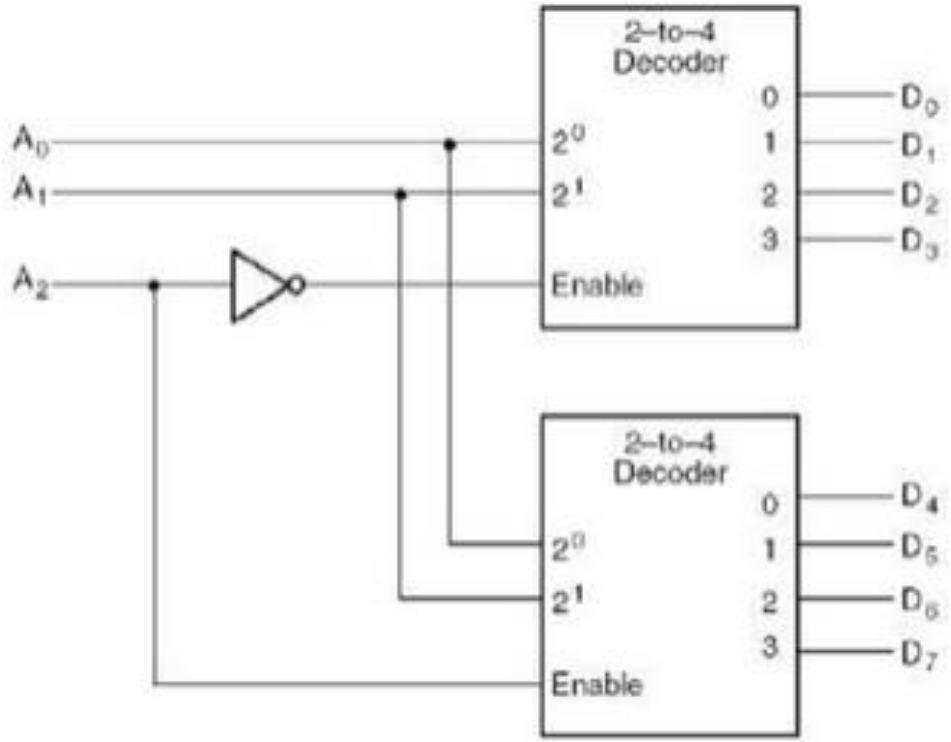


Decoder Expansion

- It is possible to build larger decoders using two or more smaller ones
  - A 6 –to- 64 decoder can be designed with four 4-to-16 decoders and one 2-to-4 decoder

Example

- Construct a 3-to-8 decoder using two 2-to-4 decoders with enable inputs



Combinational Circuit Implementation using Decoders

- A decoder provides the 2^n minterms of n input variables
- Since any Boolean functions can be expressed as a sum of minterms, one can use a decoder to implement any function of n variables.
- The decoder is used to generate the minterms and an additional OR gate is used to generate the sum of the required minterms
- In this way, any combinational circuit with n inputs and m outputs can be implemented using an n-to-2^n decoder in addition to m OR gates

N/B: No simplification is required, because we need SSOP in order to work

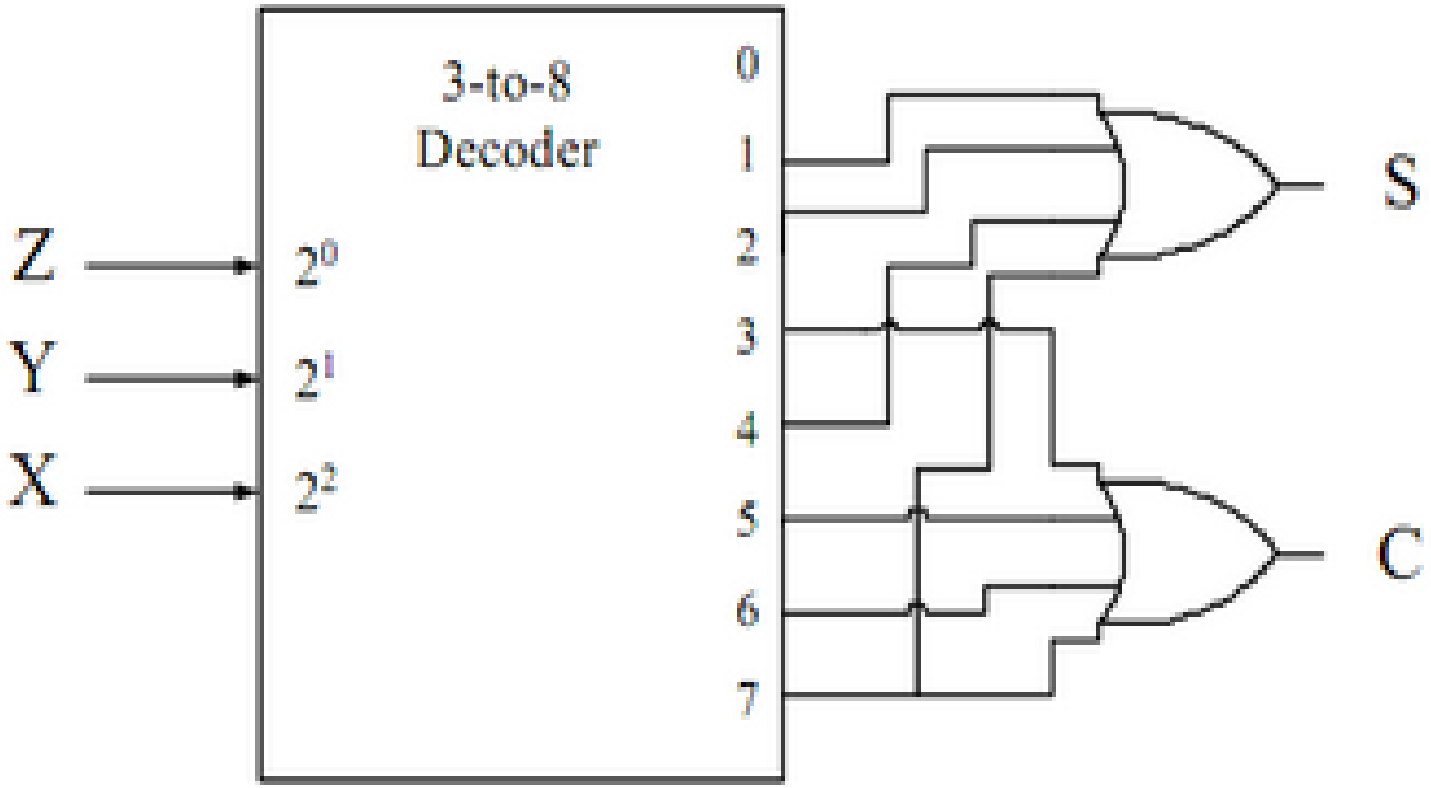
Decoder Implementation of a Full Adder

Decimal value	Input			Output	
	X	Y	Z	S	C
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

$S(X, Y, Z) = m(1, 2. 4. 7)$

$C(X, Y, Z) = m(3, 5, 6, 7)$

Decoder Implementation of a Full Adder



Practice Exercise

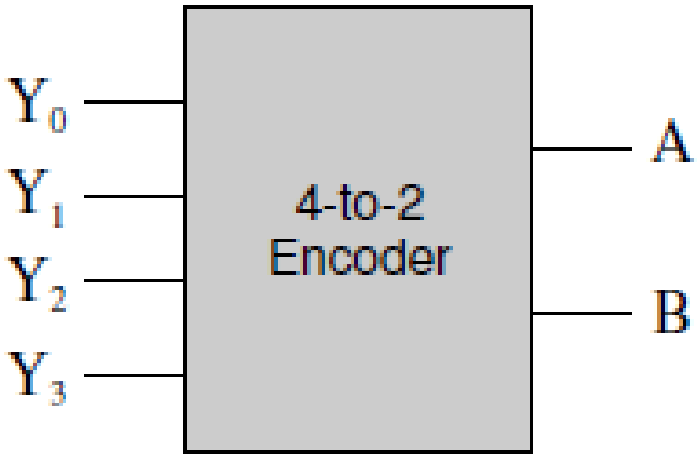
- Implement a 4 : 16 decoder using 3 : 8 decoders

- Implement a subtractor circuit using decoders

Encoders

- An encoder has
  - 2^n inputs
  - n outputs
- Its outputs the binary value of the selected (or active) input
- Performs the inverse operation of a decoder
- Issues
  - What if more than one input is active?
  - What if no inputs are active?

Encoders



Y_0	Y_1	Y_2	Y_3	A	B
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

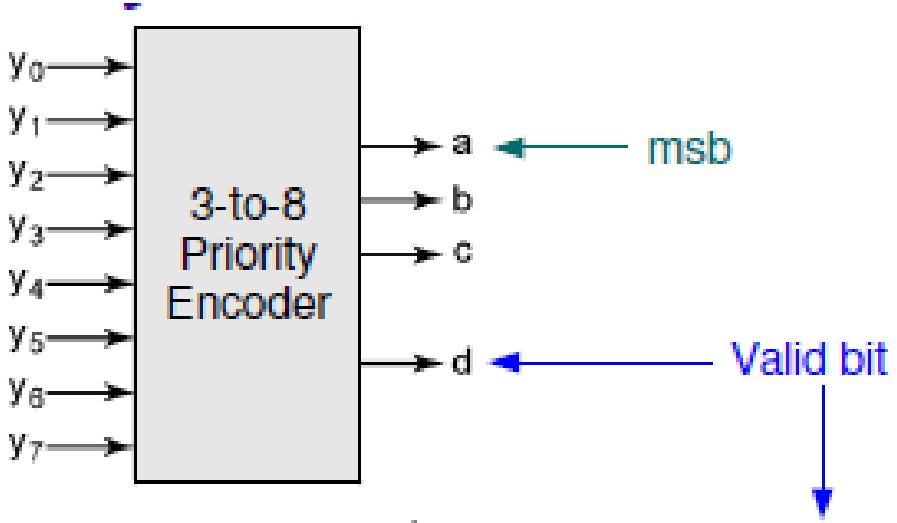
Classification of Encoders

- Priority Encoder
- Decimal to BCD Encoder
- Octal to Binary Encoder
- Hexadecimal to Binary Encoder

Priority Encoder

- If more than one input is active, the higher order input has priority over the lower-order input
  - The higher value is encoded on thee output
- A valid indicator, d, is included to indicate whether or not the output is valid
  - Output is invalid when no inputs are active
    - d = 0
  - Output is valid when at least one input is active
    - d = 1

Priority Encoders



y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	a	b	c	d
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
X	1	0	0	0	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0	1
X	X	X	1	0	0	0	0	0	1	1	1
X	X	X	X	1	0	0	0	1	0	0	1
X	X	X	X	X	1	0	0	1	0	1	1
X	X	X	X	X	X	1	0	1	1	0	1
X	X	X	X	X	X	X	1	1	1	1	1