Unit-6: LSI and MSI component on combinational logic

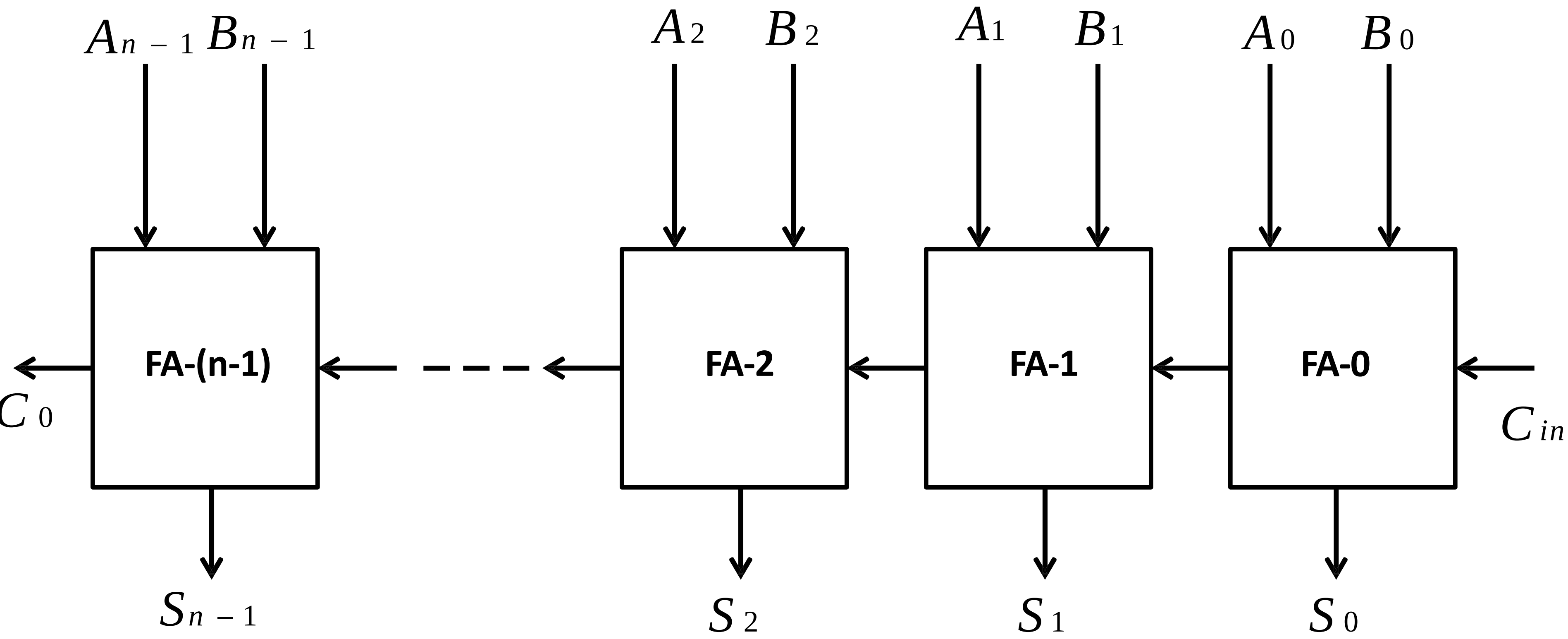
- Binary adder and substractor, decimal adder,
- Magnitude comparator,
- decoder and encoder,
- Multiplexer and demultiplexer,
- Read-only memory (ROM), programmable,

  Logic Array (PLA), Programmable Array Logic (PAL).

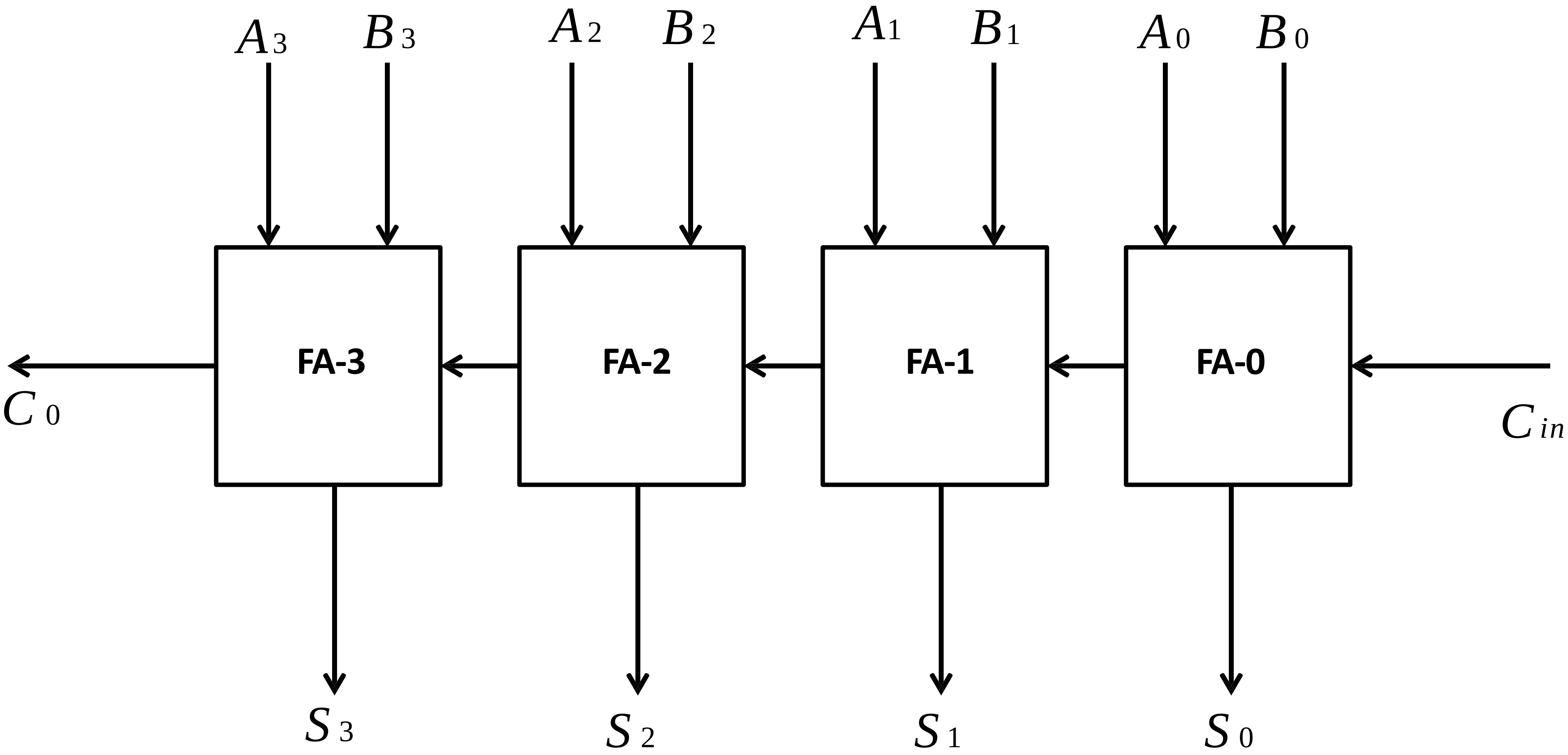
N – Bit Parallel Adder

- ✓ The full adder is capable of adding two single digit binary numbers along with a carry input.
- ✓ But in practice we need to add binary numbers which are much longer than one bit.
- ✓ To add two n-bit binary numbers we need to use the n-bit parallel adder.
- ✓ It uses a number of full adders in cascade.
- ✓ The carry output of the previous full adder is connected to the carry input of the next full adder..

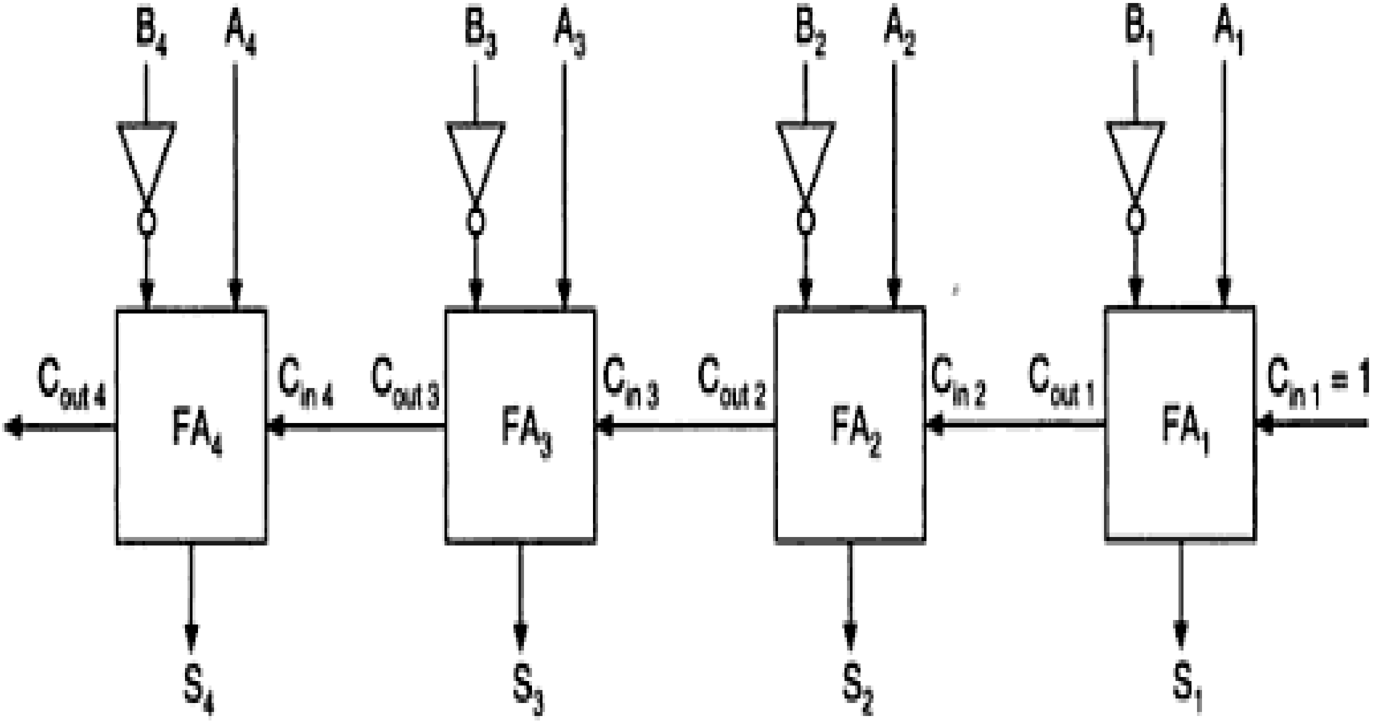
N – Bit Parallel Adder



4 – Bit Parallel Adder using full adder

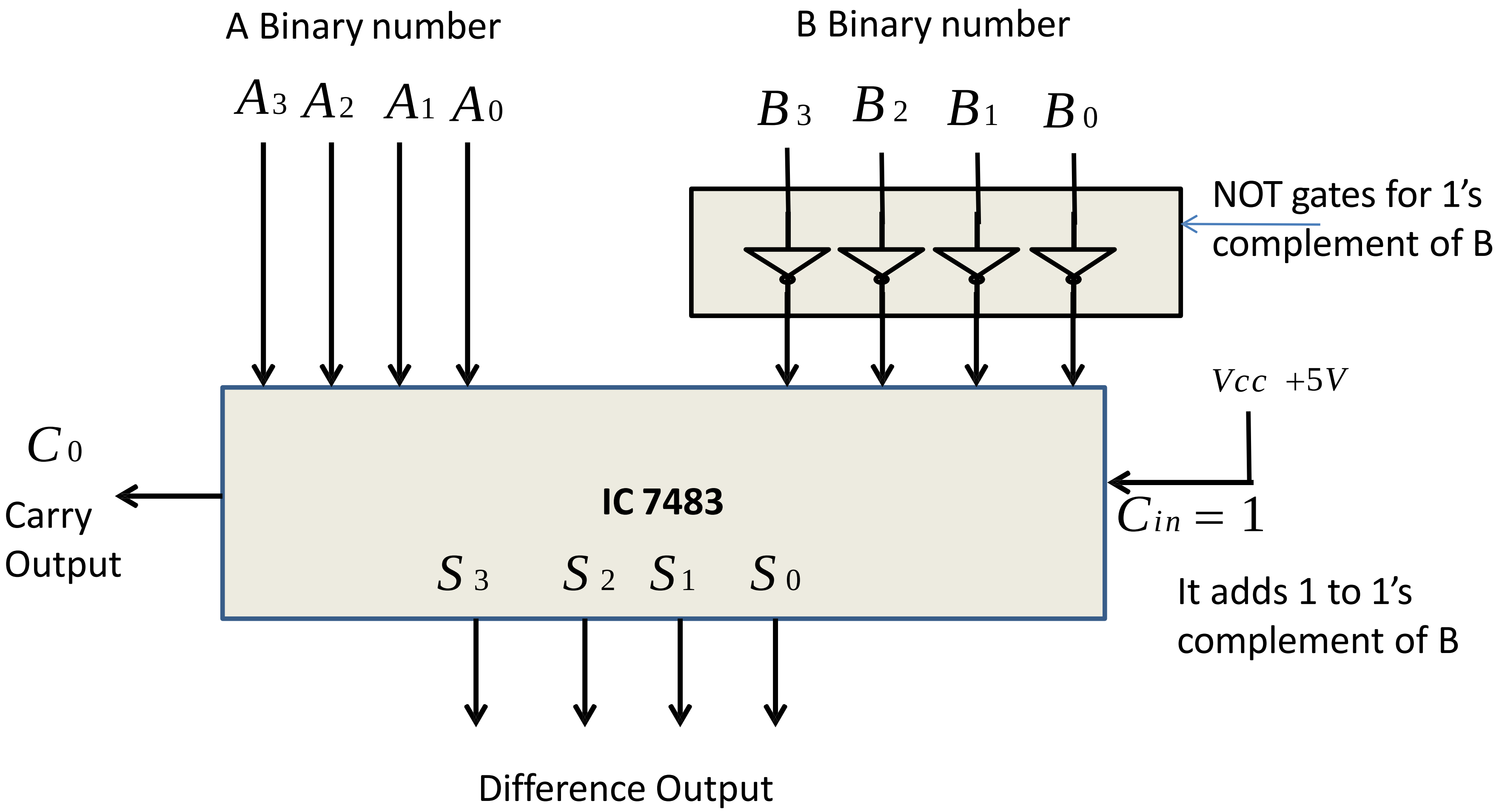


4 bit parallel subtractor:



Logic diagram of a 4-bit parallel subtractor.

4 Bit Binary Parallel Subtractor using IC 7483



A Binary number

B Binary number

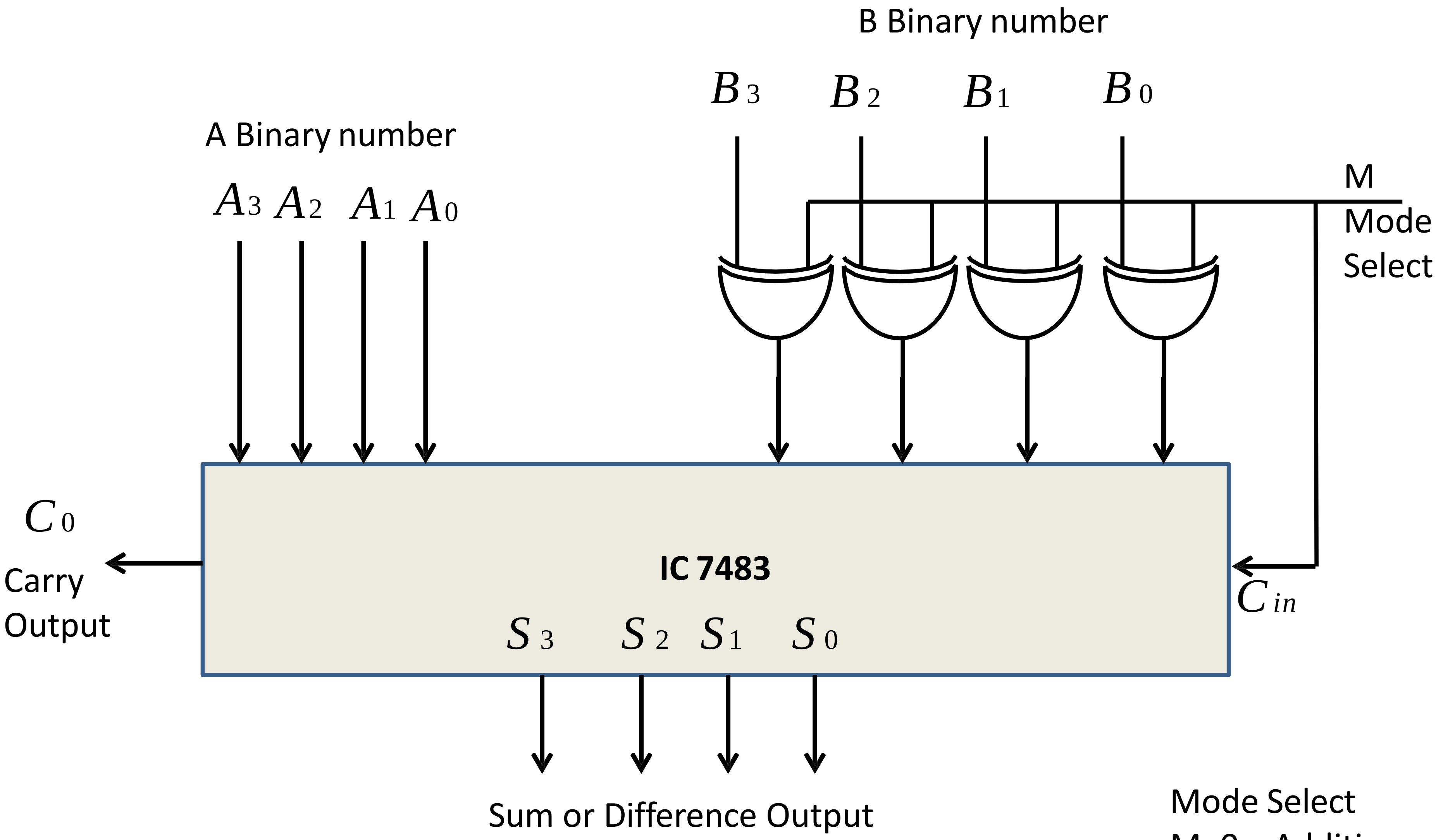
NOT gates for 1’s complement of B

Carry Output

It adds 1 to 1’s complement of B

Difference Output

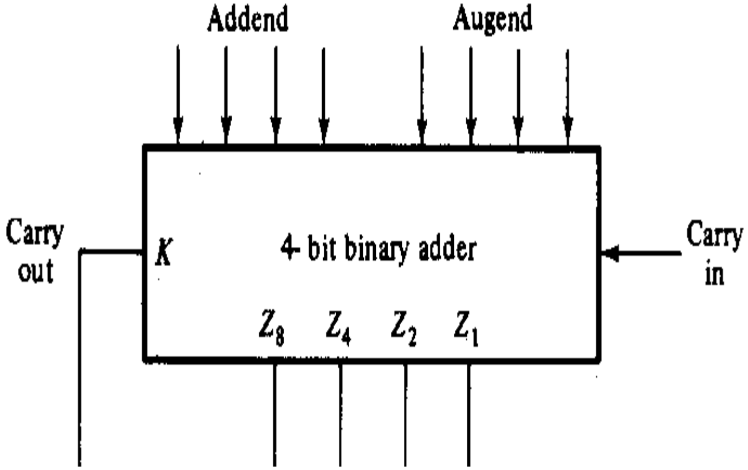
IC 7483 as Parallel Adder/Subtractor



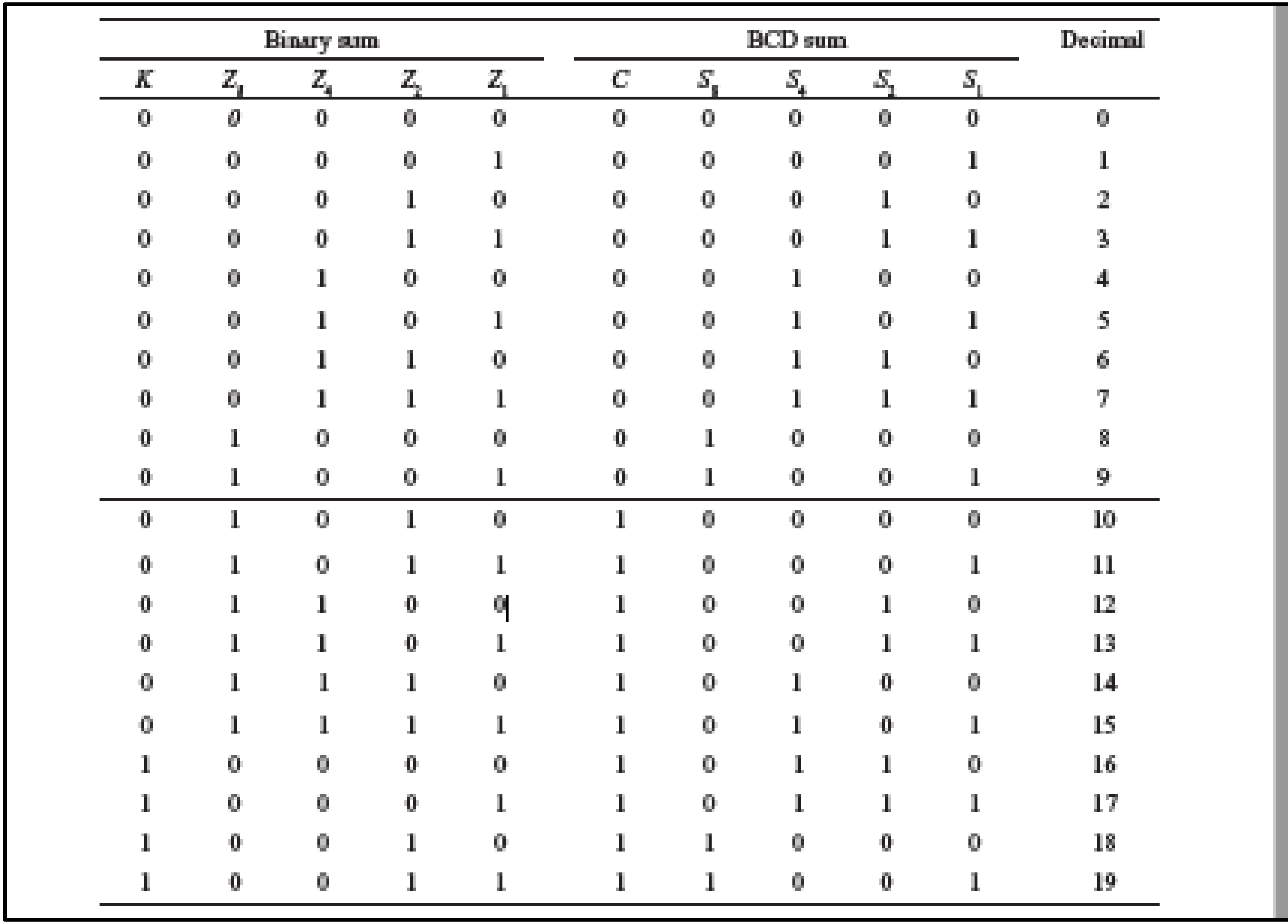
Mode Select
M=0 Addition
M=1 Subtraction

BCD adder/ Decimal adder:

- If we add two decimal digits, the maximum result will be 19 (9+9+1).



- The above circuit gives the binary sum. Now we have to convert the binary sum into BCD sum. For this, we have to design a circuit which converts the binary sum into BCD sum. The truth table is as:



Binary sum					BCD sum					Decimal
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

For 0-9=>Binary sum = BCD sum
For10-19=> Binary sum + 0110= BCD sum.

- When C=1, it is necessary to add binary 0110 to the binary sum and provide an o/p carry to the next stage.
- A correction is needed when the binary sum has an o/p carry K=1.
- Binary 1010-1111 also needs correction and have a 1 in position Z8.
- To distinguish them from binary 1000 and 1001, which also have a 1 in Z8 position, we specify further that either Z4 or Z2 must have a 1.
- The condition for the correction and o/p carry can be:

  C=K+Z8Z4+Z8Z2=1

(Or it can be derived using K-map for 5 variables from the truth table.)

The circuit diagram of BCD adder is as:

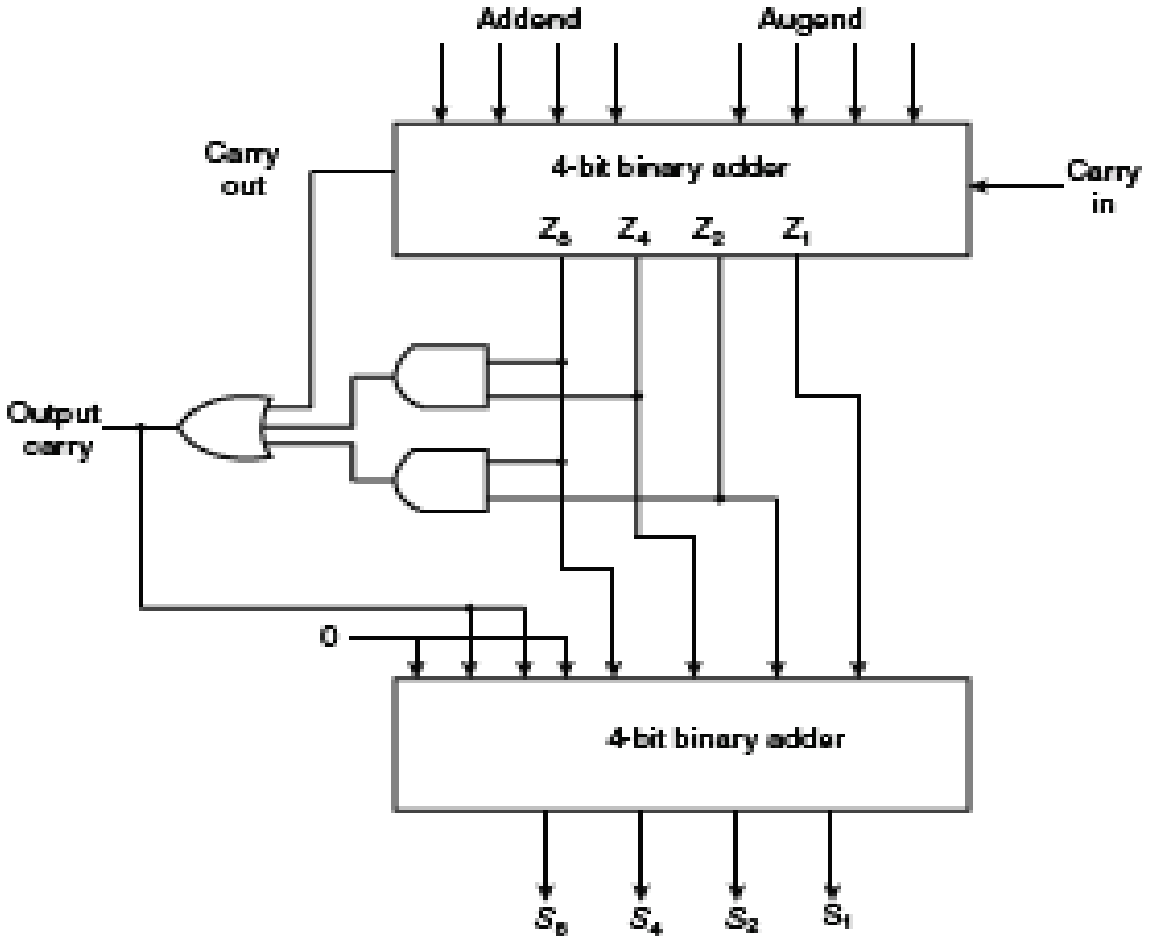
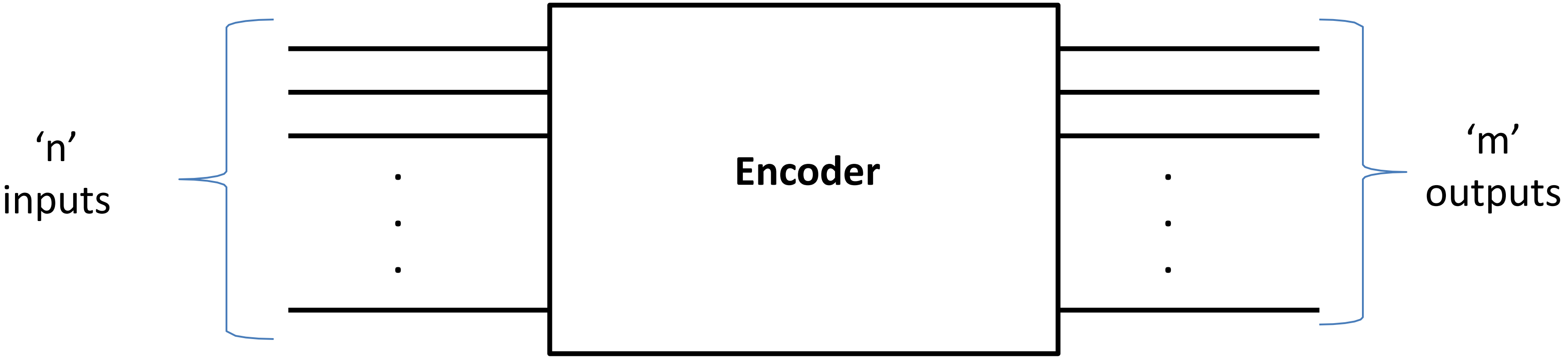


Figure 5-6 Block diagram of a BCD adder

Encoder

- ✓ Encoder is a combinational circuit which is designed to perform the inverse operation of decoder.
- ✓ An encoder has ‘n’ number of input lines and ‘m’ number of output lines.
- ✓ An encoder produces an m bit binary code corresponding to the digital input number.
- ✓ The encoder accepts an n input digital word and converts it into m bit another digital word

Encoder



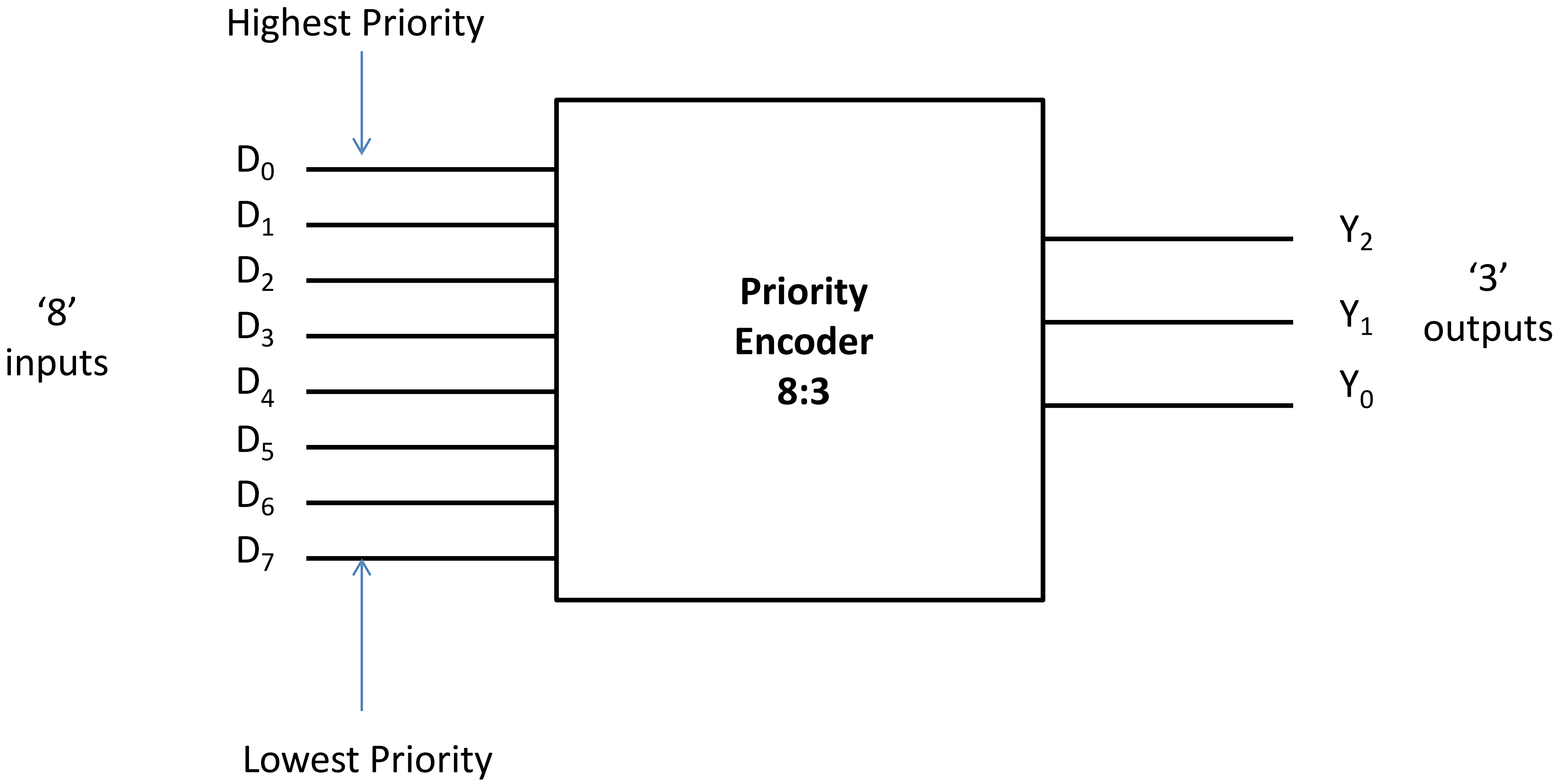
Types of Encoders

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

Priority Encoder

- ✓ This is a special type of encoder.
- ✓ Priorities are given to the input lines.
- ✓ If two or more input lines are “1” at the same time, then the input line with highest priority will be considered.

Priority Encoder 8:3



Highest Priority

D_0

D_1

D_2

D_3

D_4

D_5

D_6

D_7

Lowest Priority

‘8’ inputs

Priority Encoder 8:3

Y_2

Y_1

Y_0

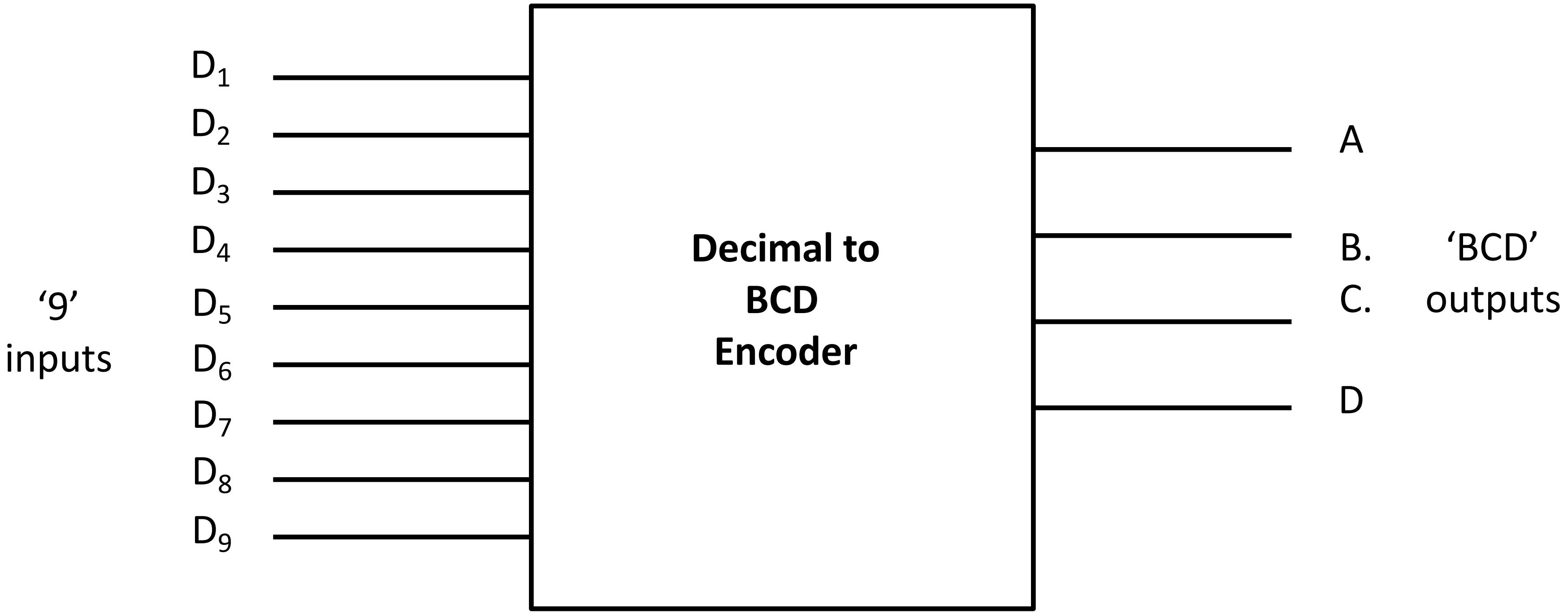
‘3’ outputs

Priority Encoder 8:3

Truth Table:

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	X	X	X
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

Decimal to BCD Encoder



Decimal to BCD Encoder

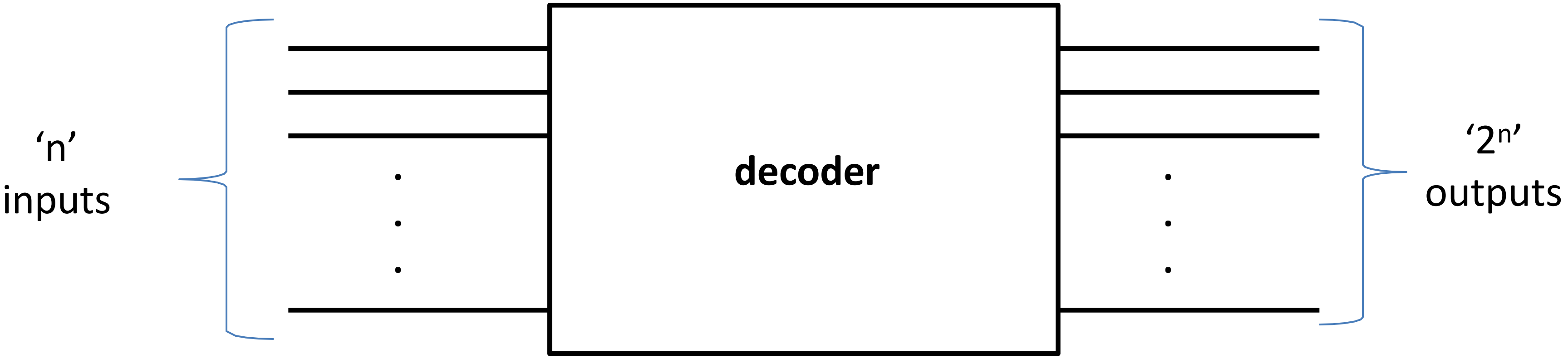
Truth Table:

Inputs									Outputs			
D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D	C	B	A
0	0	0	0	0	0	0	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	1	X	0	0	1	0
0	0	0	0	0	0	1	X	X	0	0	1	1
0	0	0	0	0	1	X	X	X	0	1	0	0
0	0	0	0	1	X	X	X	X	0	1	0	1
0	0	0	1	X	X	X	X	X	0	1	1	0
0	0	1	X	X	X	X	X	X	0	1	1	1
0	1	X	X	X	X	X	X	X	1	0	0	0
1	X	X	X	X	X	X	X	X	1	0	0	1

Decoder

- ✓ Decoder is a combinational circuit which is designed to perform the inverse operation of encoder.
- ✓ An decoder has ‘n’ number of input lines and maximum ‘2^n’ number of output lines.
- ✓ Decoder is identical to a demultiplexer without data input.

Decoder



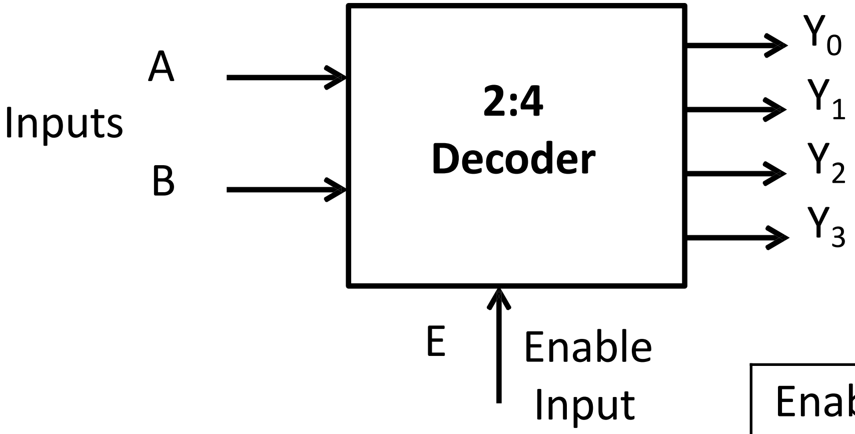
Typical applications of Decoders

- ✓ Code Converters
- ✓ BCD to 7 segment decoders
- ✓ Nixie tube decoders
- ✓ Relay actuators

Types of Decoders

- ✓2 to 4 line Decoder
- ✓3 to 8 line Decoder
- ✓BCD to 7 Segment Decoder

2 to 4 Line Decoder

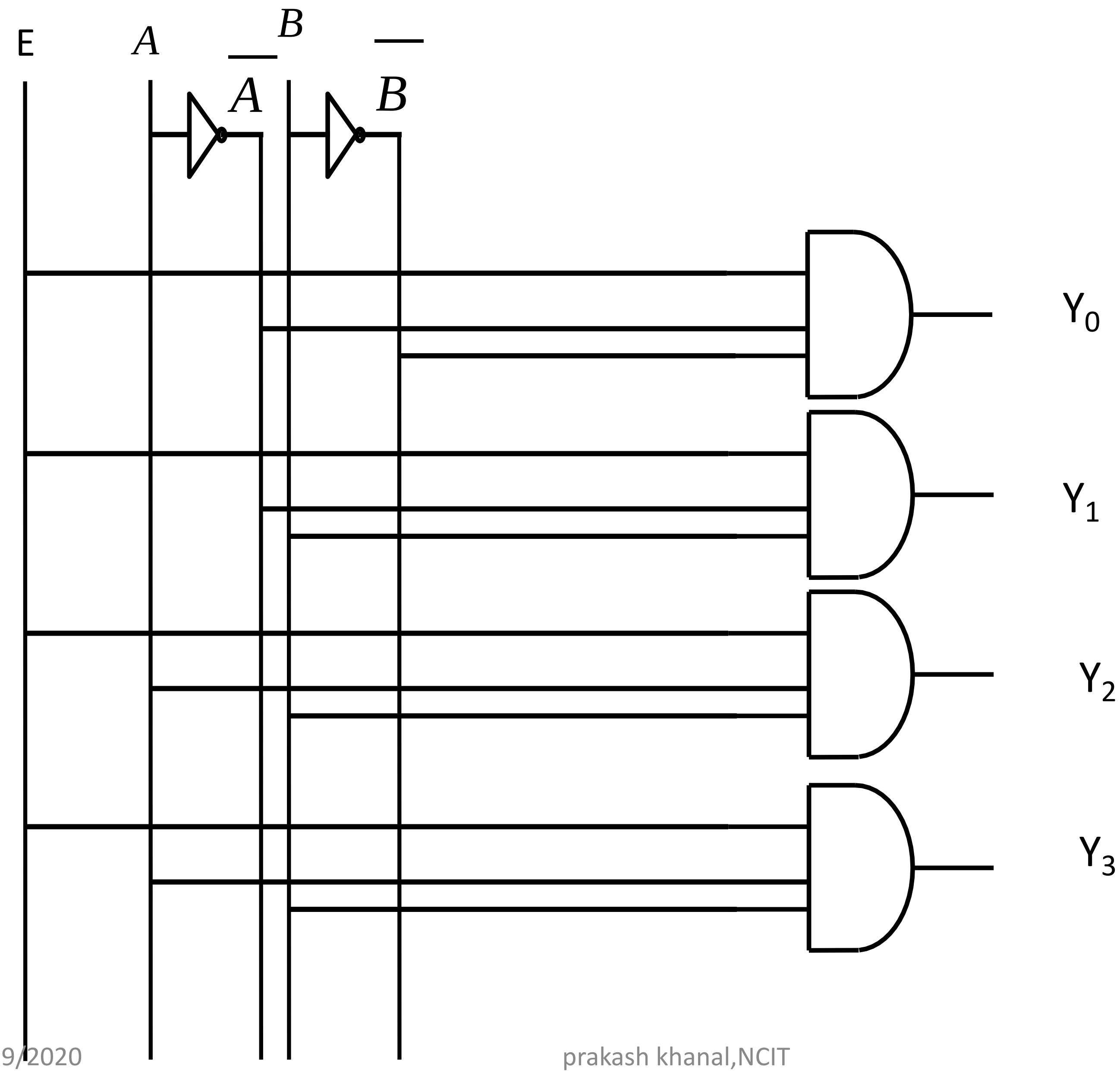


Block Diagram

Truth Table

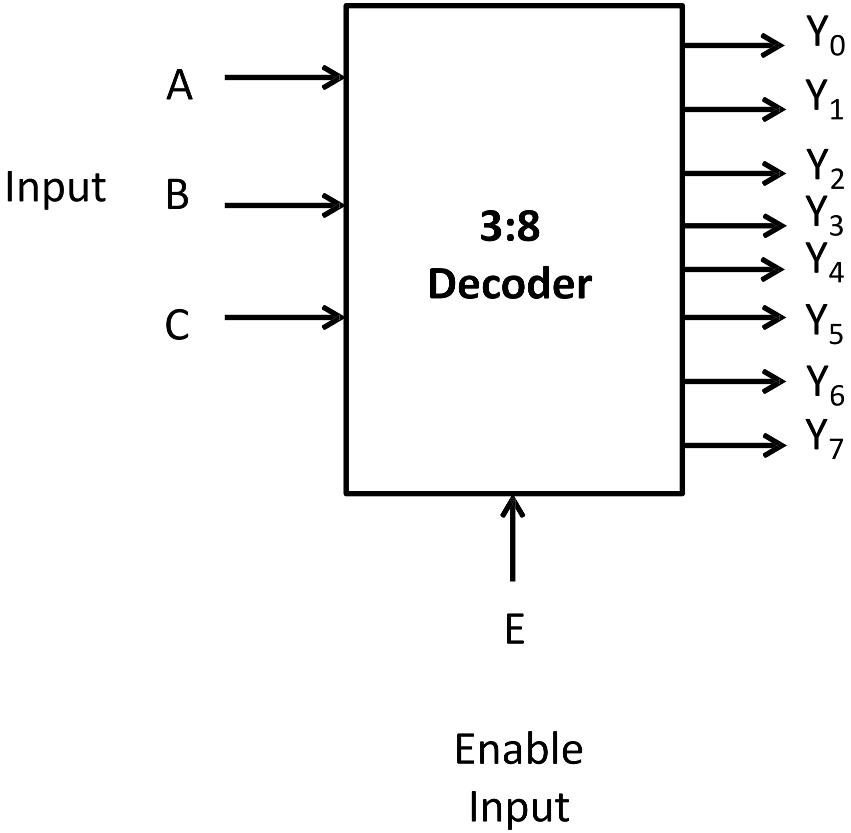
Enable i/p	Data Inputs		Outputs			
E	A	B	Y_0	Y_1	Y_2	Y_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

2 to 4 Line Decoder



3 to 8 Line Decoder

Block Diagram

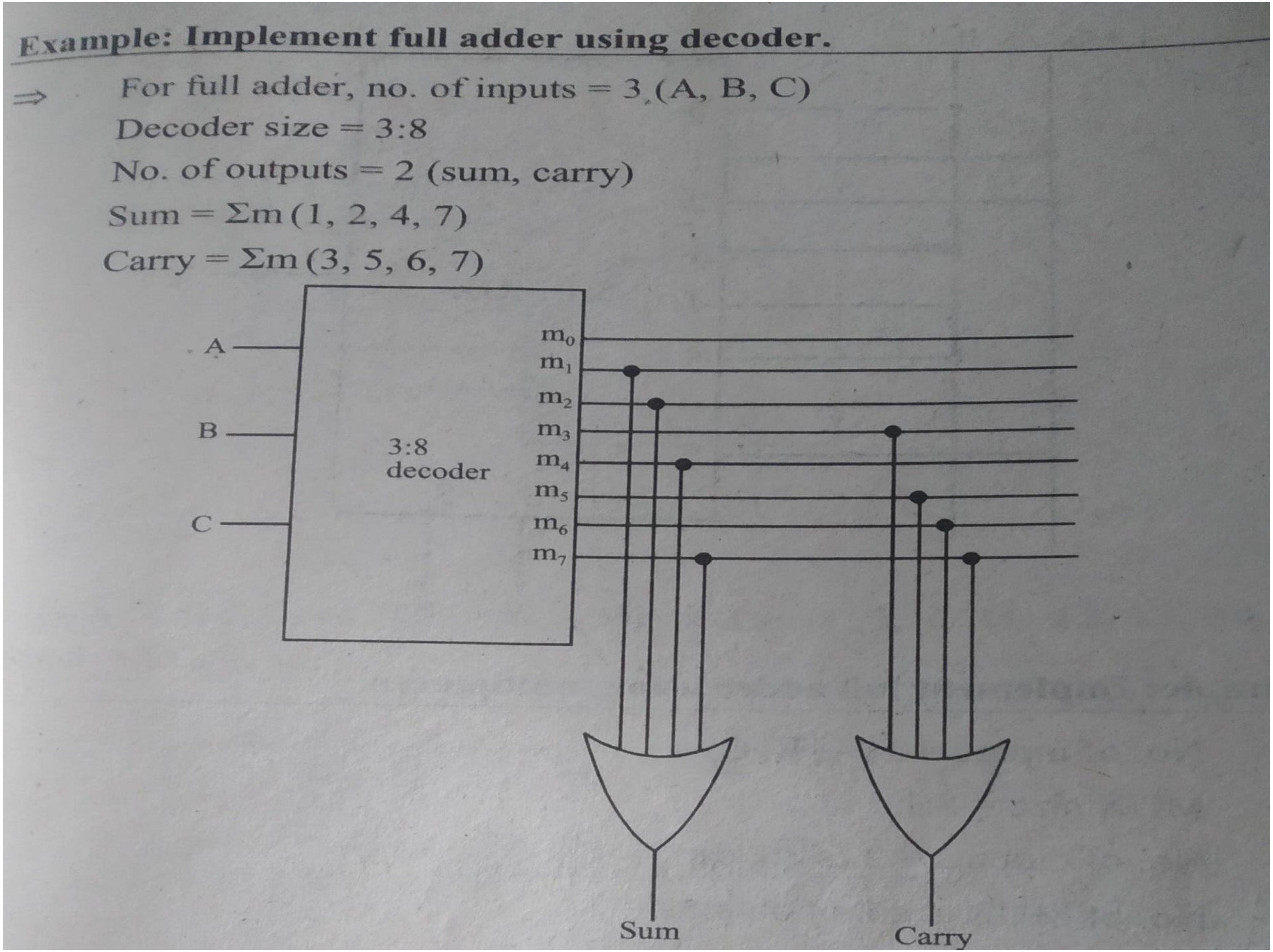


3 to 8 Line Decoder

Truth Table

Enable i/p	Inputs			Outputs							
E	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Implement a full adder using decoder



Example: Implement full adder using decoder.

⇒ For full adder, no. of inputs = 3 (A, B, C)

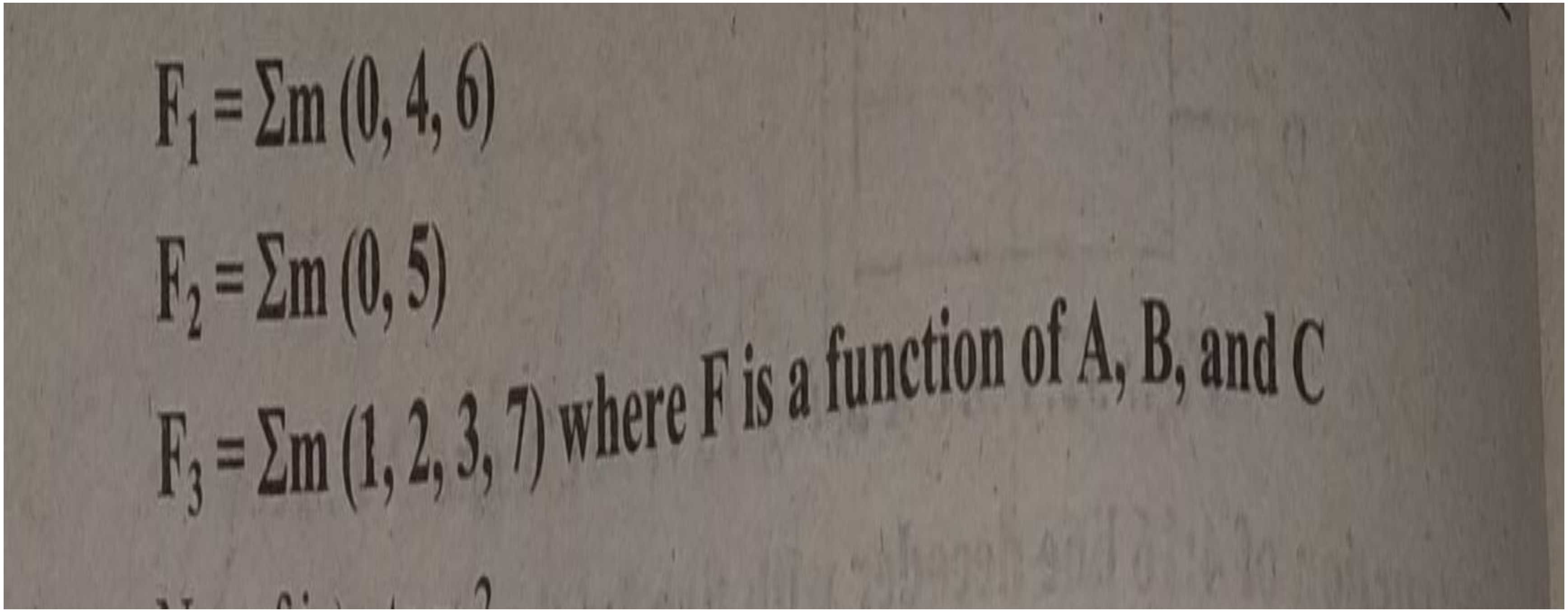
Decoder size = 3:8

No. of outputs = 2 (sum, carry)

Sum = $\Sigma m(1, 2, 4, 7)$

Carry = $\Sigma m(3, 5, 6, 7)$

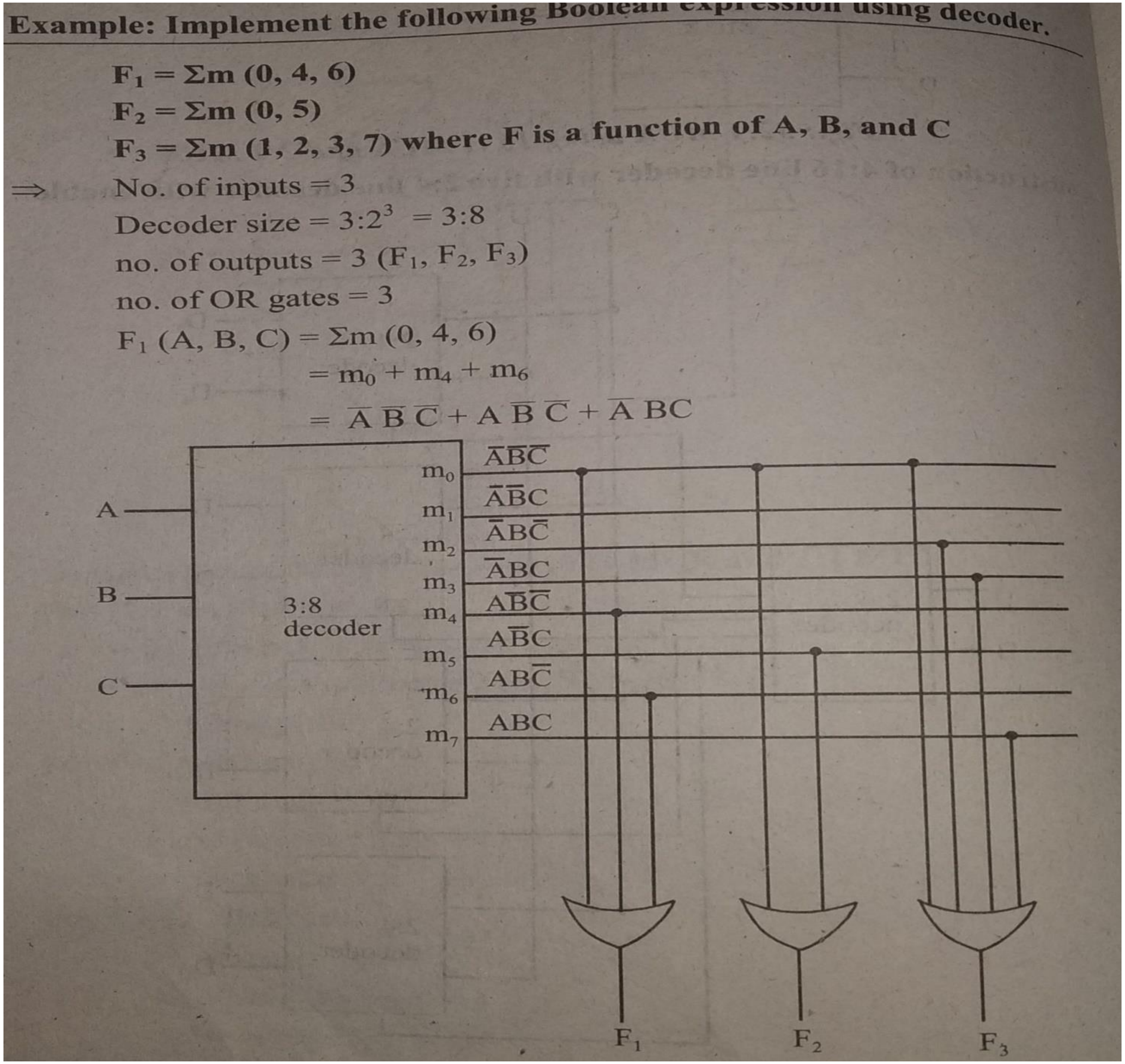
Implement the following Boolean function using decoder



$F_1 = \Sigma m\,(0, 4, 6)$

$F_2 = \Sigma m\,(0, 5)$

$F_3 = \Sigma m\,(1, 2, 3, 7)$ where F is a function of A, B, and C



Example: Implement the following Boolean expression using decoder.

$F_1 = \Sigma m\ (0, 4, 6)$

$F_2 = \Sigma m\ (0, 5)$

$F_3 = \Sigma m\ (1, 2, 3, 7)$ where F is a function of A, B, and C

⇒ No. of inputs = 3

Decoder size = $3{:}2^3 = 3{:}8$

no. of outputs = 3 (F_1, F_2, F_3)

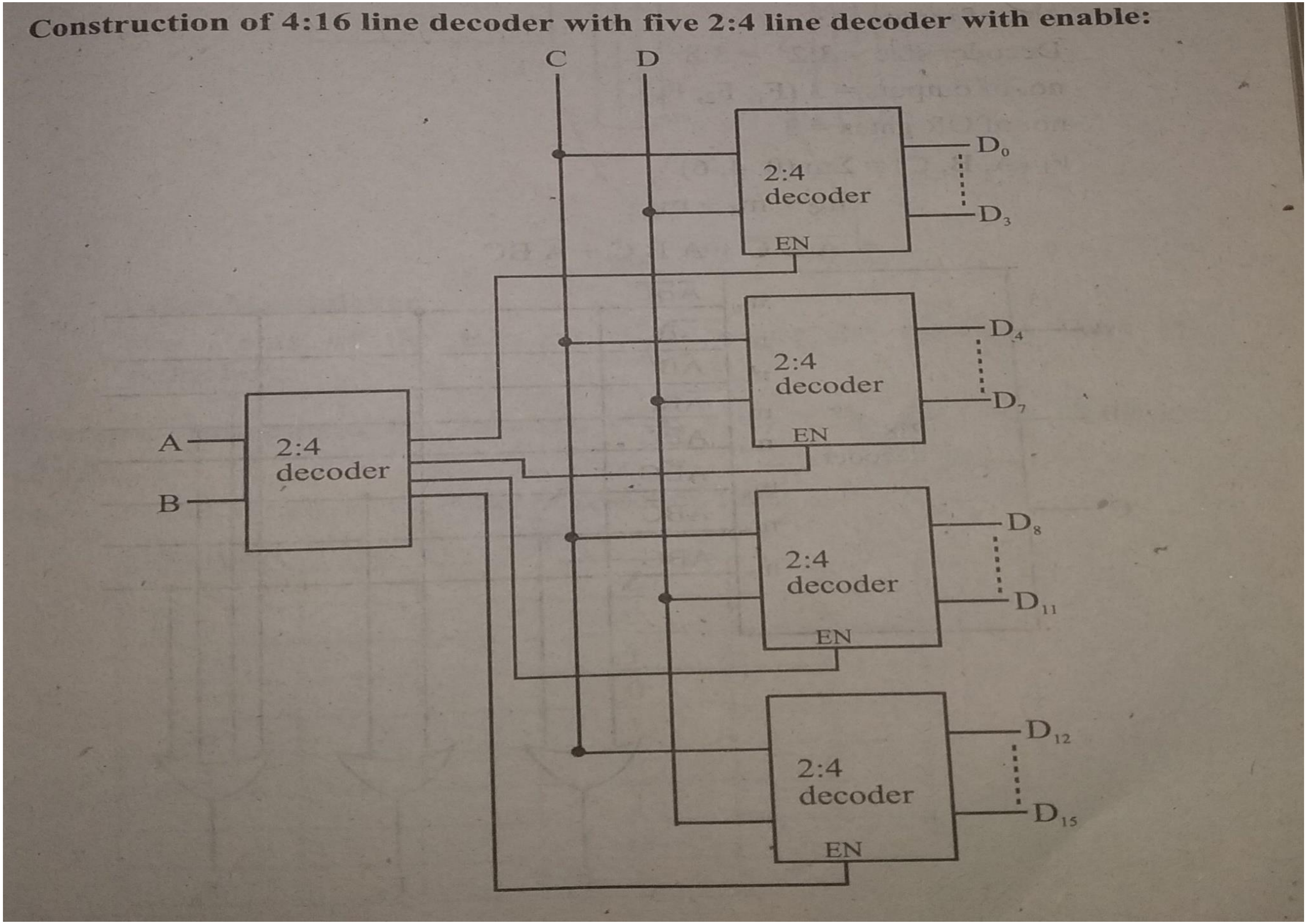
no. of OR gates = 3

$F_1\ (A, B, C) = \Sigma m\ (0, 4, 6)$

$= m_0 + m_4 + m_6$

$= \overline{A}\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,BC$

Construct 4:16 line decoder with five 2:4 decoder with enable

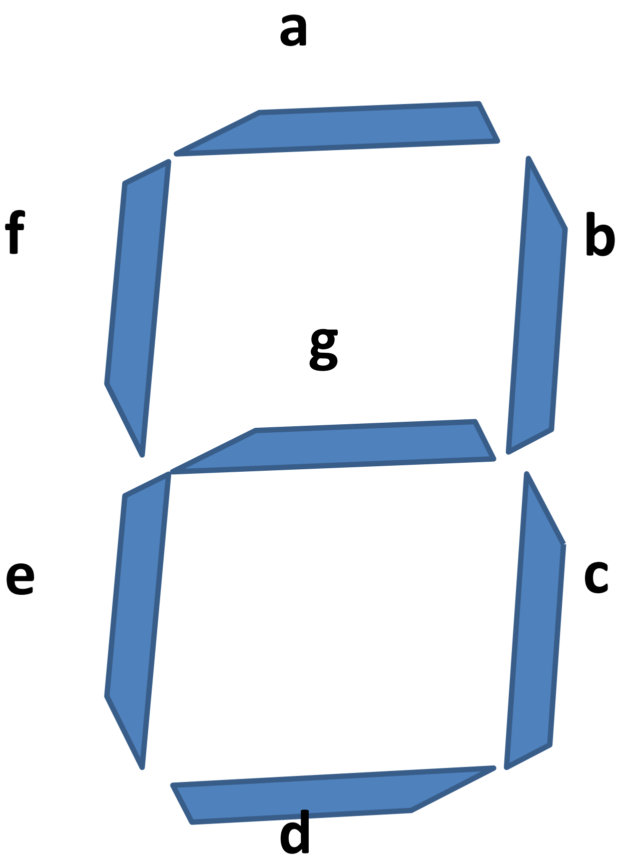


Construction of 4:16 line decoder with five 2:4 line decoder with enable:

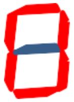
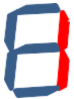
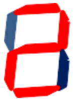
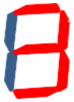
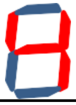
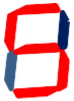
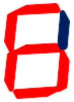
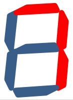
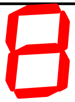
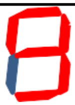
Comparison between Encoder & Decoder

Sr. No.	Parameter	Encoder	Decoder
1	Input applied	Active input signal (original message signal)	Coded binary input
2	Output generated	Coded binary output	Active output signal (original message)
3	Input lines	2^n	n
4	Output lines	N	2^n
5	Operation	Simple	Complex
6	Applications	E-mail , video encoders etc.	Microprocessors, memory chips etc.

BCD to 7 Segment Decoder - Seven Segment Display



Seven Segment Display

Segment s							Display Number	Seven Segment Display
a	b	c	d	e	f	g		
ON	ON	ON	ON	ON	ON	OFF	0	
OFF	ON	ON	OFF	OFF	OFF	OFF	1	
ON	ON	OFF	ON	ON	OFF	ON	2	
ON	ON	ON	ON	OFF	OFF	ON	3	
OFF	ON	ON	OFF	OFF	ON	ON	4	
ON	OFF	ON	ON	OFF	ON	ON	5	
ON	OFF	ON	ON	ON	ON	ON	6	
ON	ON	ON	OFF	OFF	OFF	OFF	7	
ON	ON	ON	ON	ON	ON	ON	8	
ON	ON	ON	ON	OFF	ON	ON	9	

Magnitude comparator:

- Magnitude comparator is a combinational circuit, designed to compare two n bits words applied as its input.
- The comparator has three output namely greater(A>B), lesser(A<B) and equal (A=B). Depending upon the result of comparison, one of these outputs will go high.

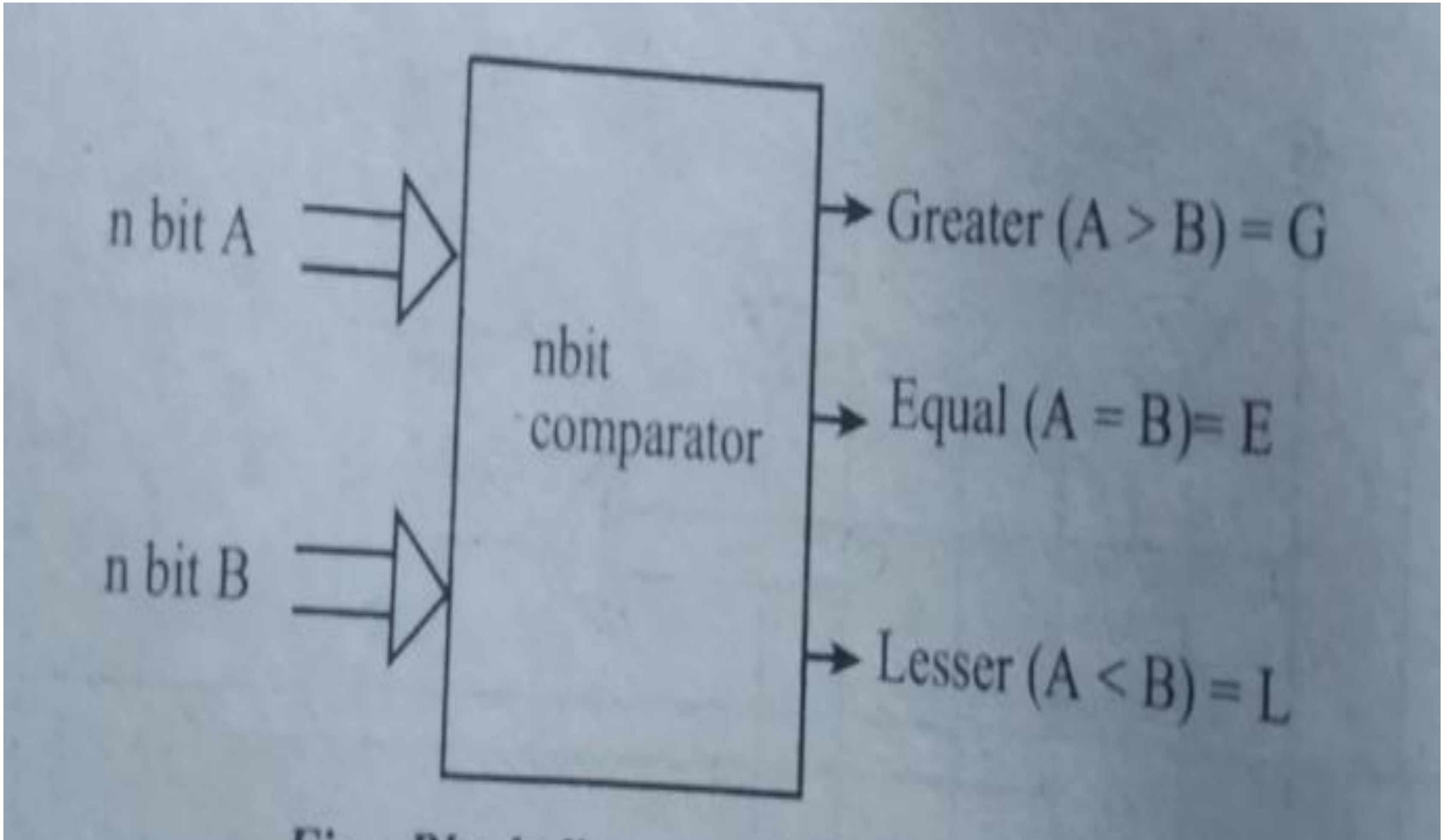
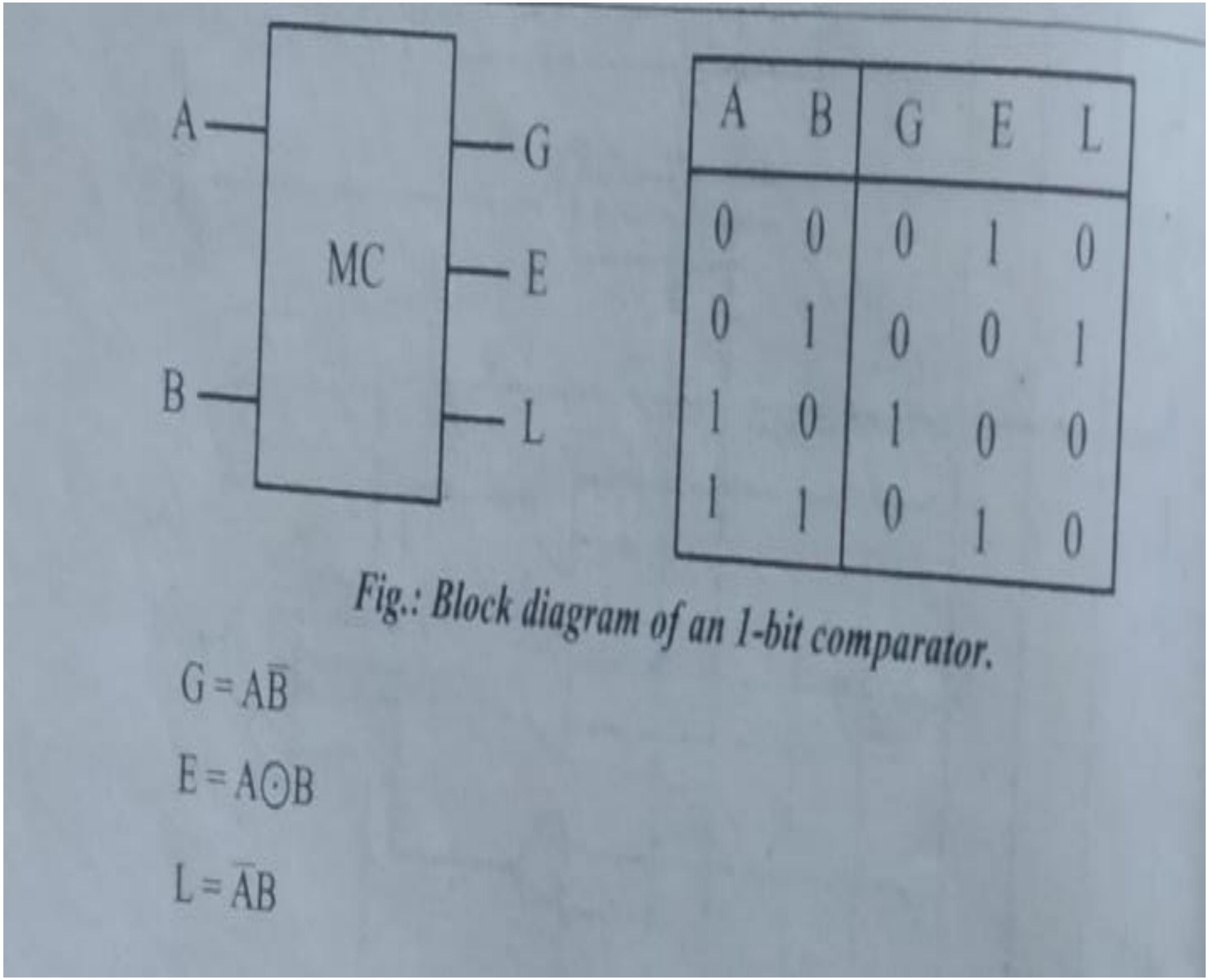


Fig.: Block diagram of an n-bit comparator.



A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Fig.: Block diagram of an 1-bit comparator.

$G = A\overline{B}$

$E = A \odot B$

$L = \overline{A}B$

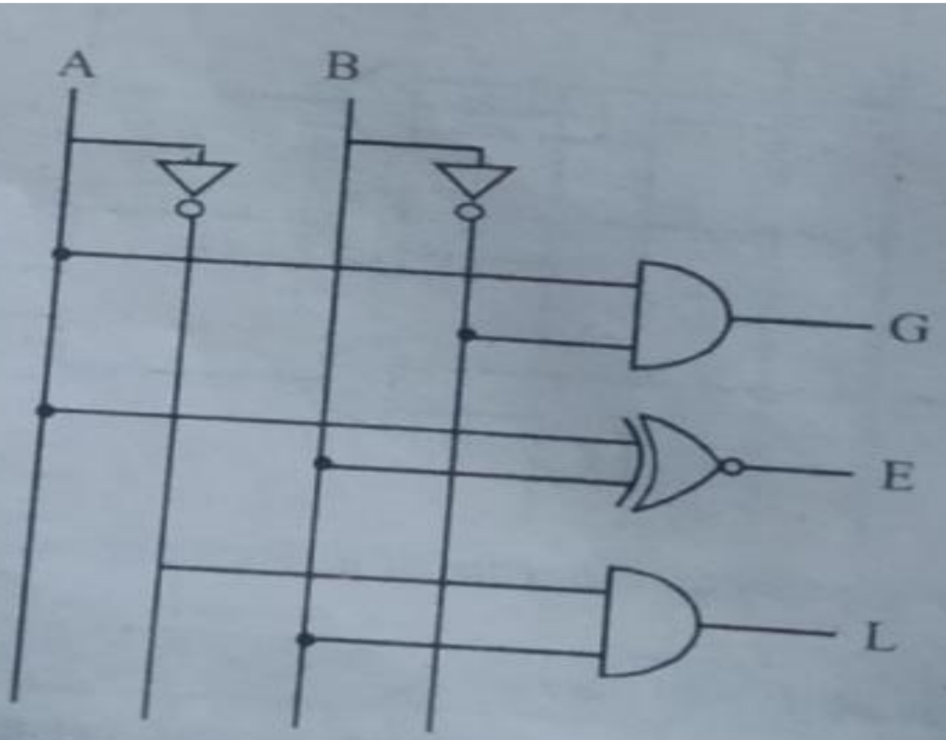


Fig.: Logic diagram of 1-bit magnitude comparator

Multiplexers

- ✓ Multiplexer is a circuit which has a number of inputs but only one output.
- ✓ Multiplexer is a circuit which transmits large number of information signals over a single line.
- ✓ Multiplexer is also known as “Data Selector” or MUX.

Necessity of Multiplexers

- ✓ In most of the electronic systems, the digital data is available on more than one lines. It is necessary to route this data over a single line.
- ✓ Under such circumstances we require a circuit which select one of the many inputs at a time.
- ✓ This circuit is nothing but a multiplexer. Which has many inputs, one output and some select lines.
- ✓ Multiplexer improves the reliability of the digital system because it reduces the number of external wired connections.

Advantages of Multiplexers

- ✓ It reduces the number of wires.
- ✓ So it reduces the circuit complexity and cost.
- ✓ We can implement many combinational circuits using Mux.
- ✓ It simplifies the logic design.
- ✓ It does not need the k-map and simplification.

Applications of Multiplexers

- ✓ It is used as a data selector to select one out of many data inputs.
- ✓ It is used for simplification of logic design.
- ✓ It is used in data acquisition system.
- ✓ In designing the combinational circuits.
- ✓ In D to A converters.
- ✓ To minimize the number of connections.

Block Diagram of Multiplexer

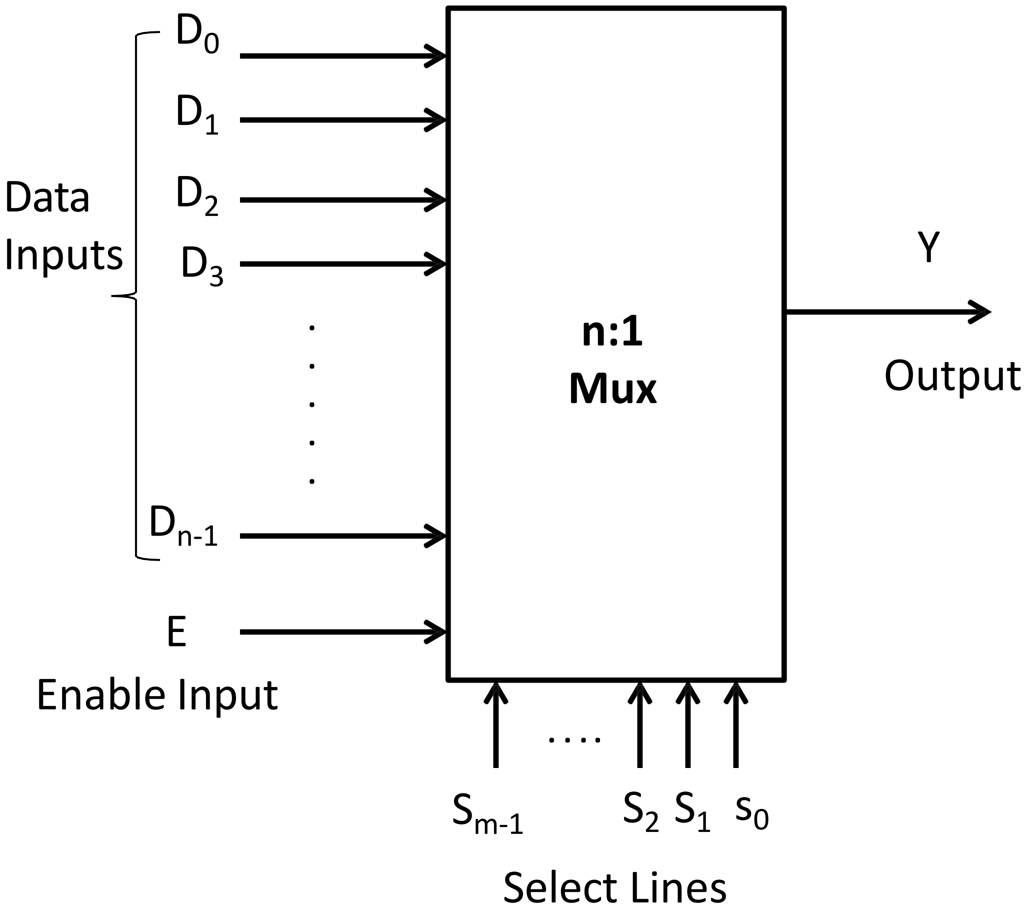


Fig. General Block Diagram

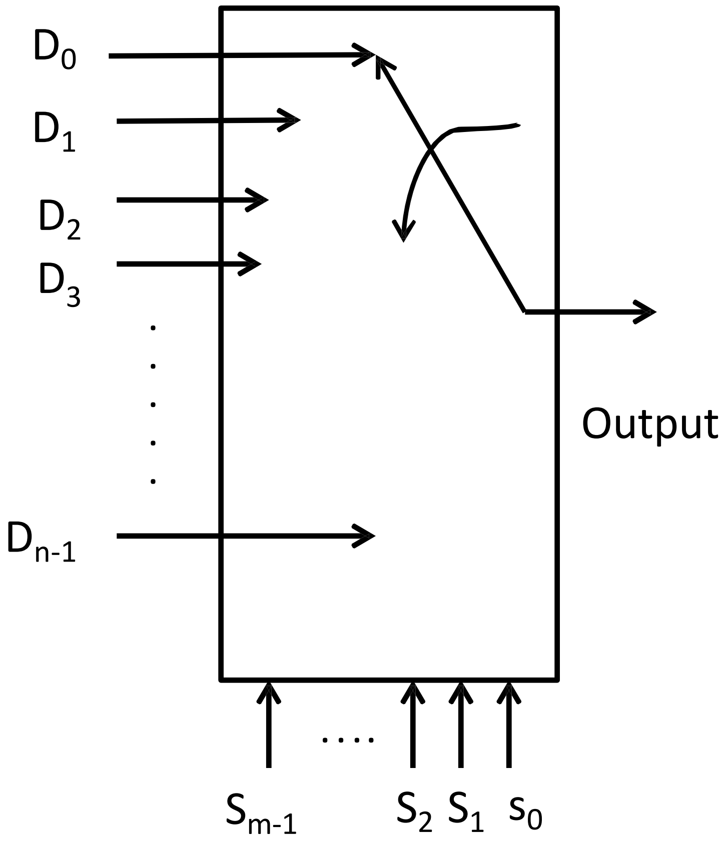


Fig. Equivalent Circuit

Relation between Data Input Lines & Select Lines

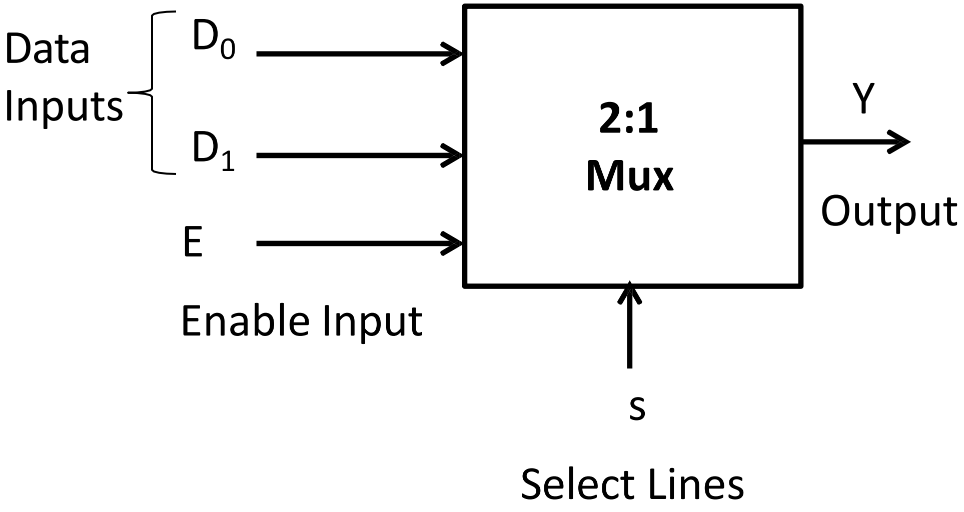
- ✓ In general multiplexer contains , n data lines, one output line and m select lines.
- ✓ To select n inputs we need m select lines such that $2^m=n$.

Types of Multiplexers

- ✓ 2:1 Multiplexer
- ✓ 4:1 Multiplexer
- ✓ 8:1 Multiplexer
- ✓ 16:1 Multiplexer
- ✓ 32:1 Multiplexer
- ✓ 64:1 Multiplexer

and so on…………

2:1 Multiplexer

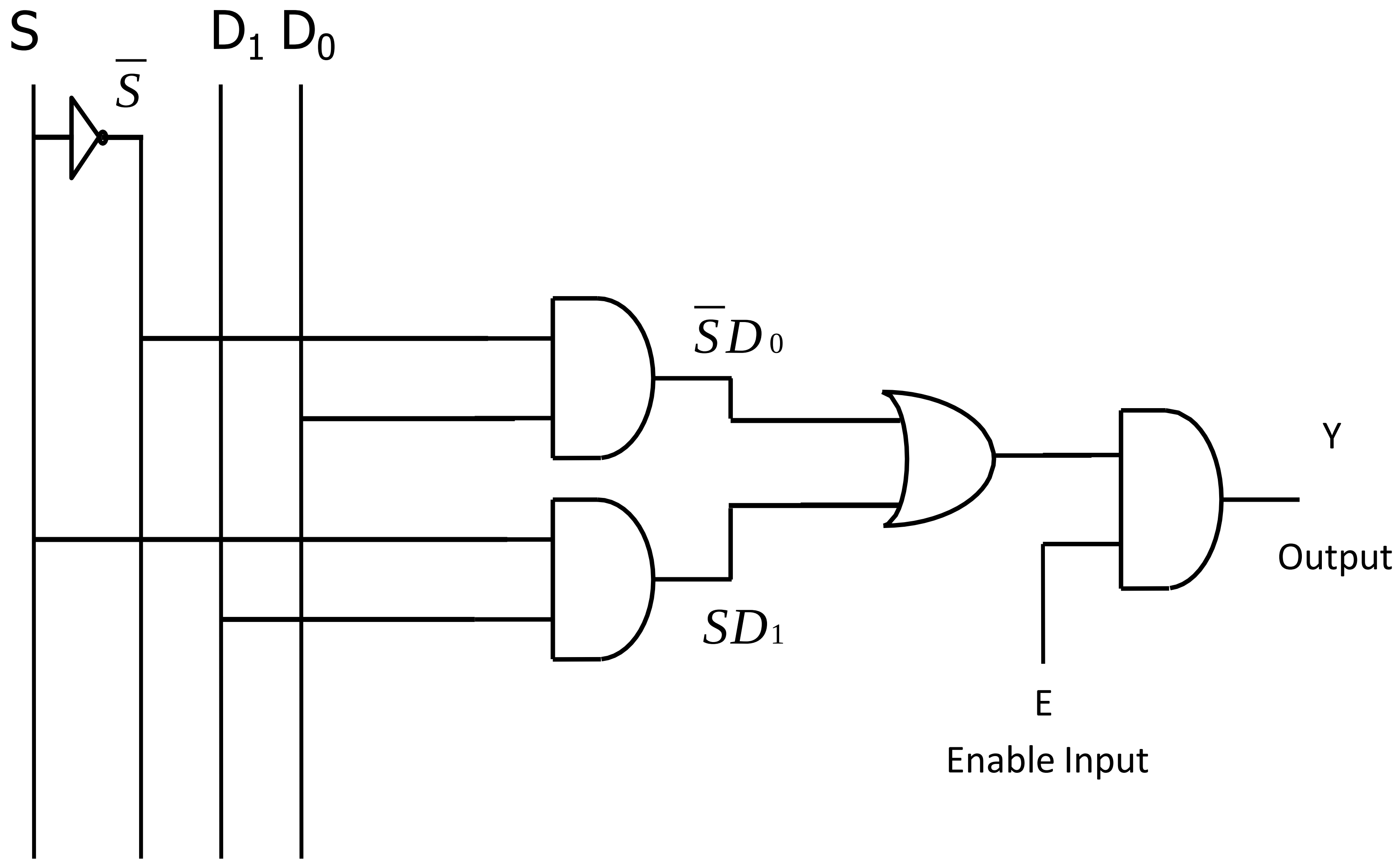


Block Diagram

Truth Table

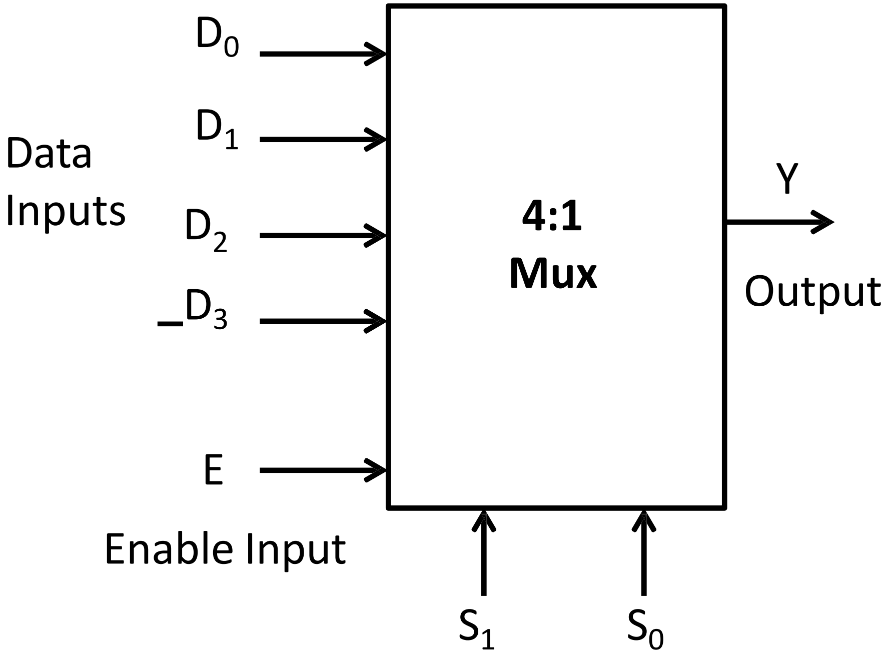
Enable i/p (E)	**Select i/p (S)**	**Output (Y)**
0	X	0
1	0	D_0
1	1	D_1

Realization of 2:1 Mux using gates



4:1 Multiplexer

Block Diagram

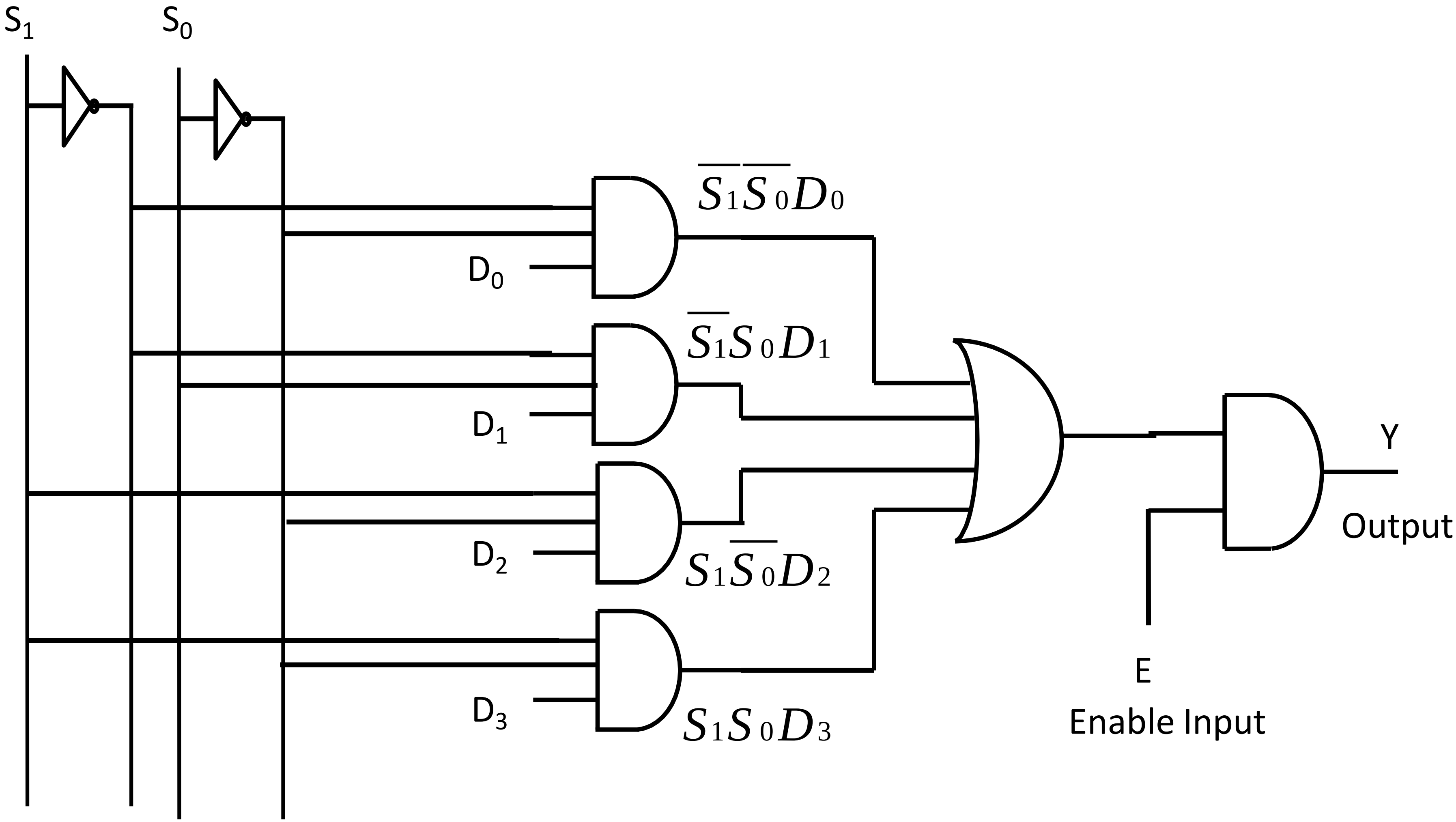


Select Lines

Truth Table

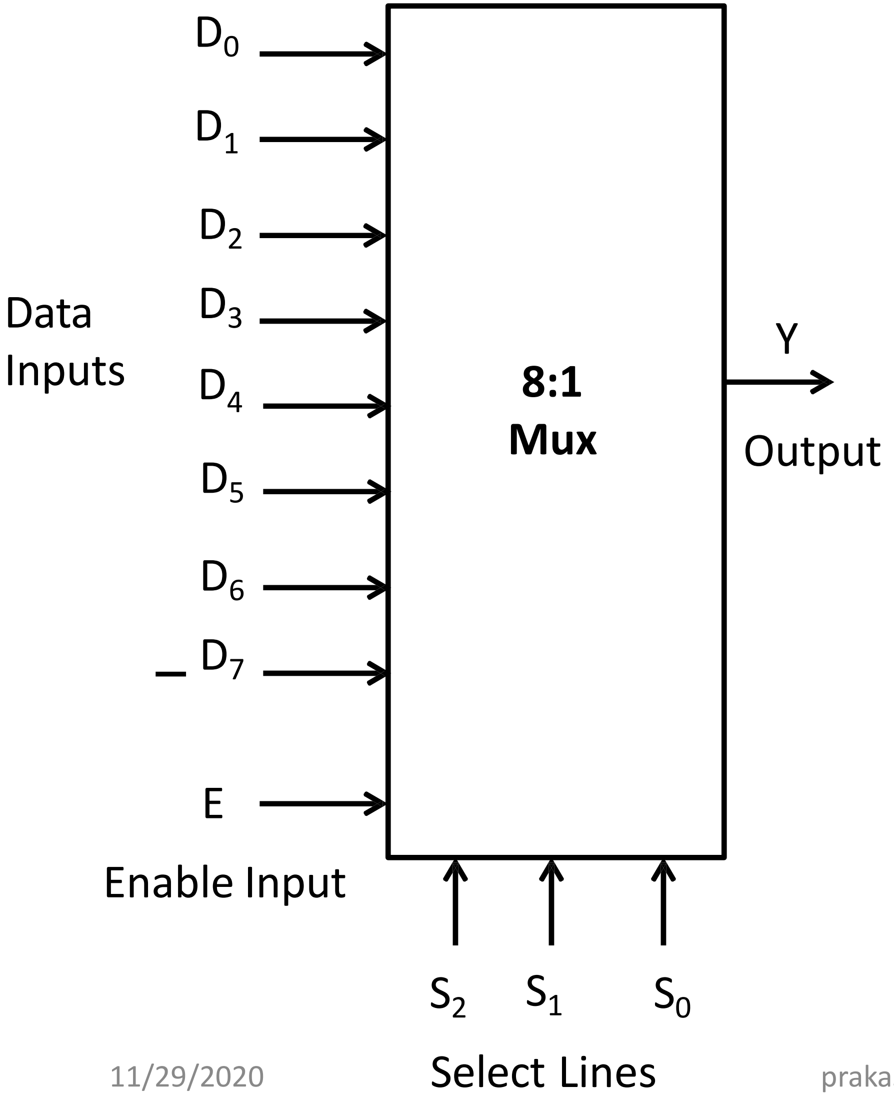
Enable i/p	Select i/p		Output
E	**S_1**	**S_0**	**Y**
0	X	X	0
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3

Realization of 4:1 Mux using gates



8:1 Multiplexer

Block Diagram

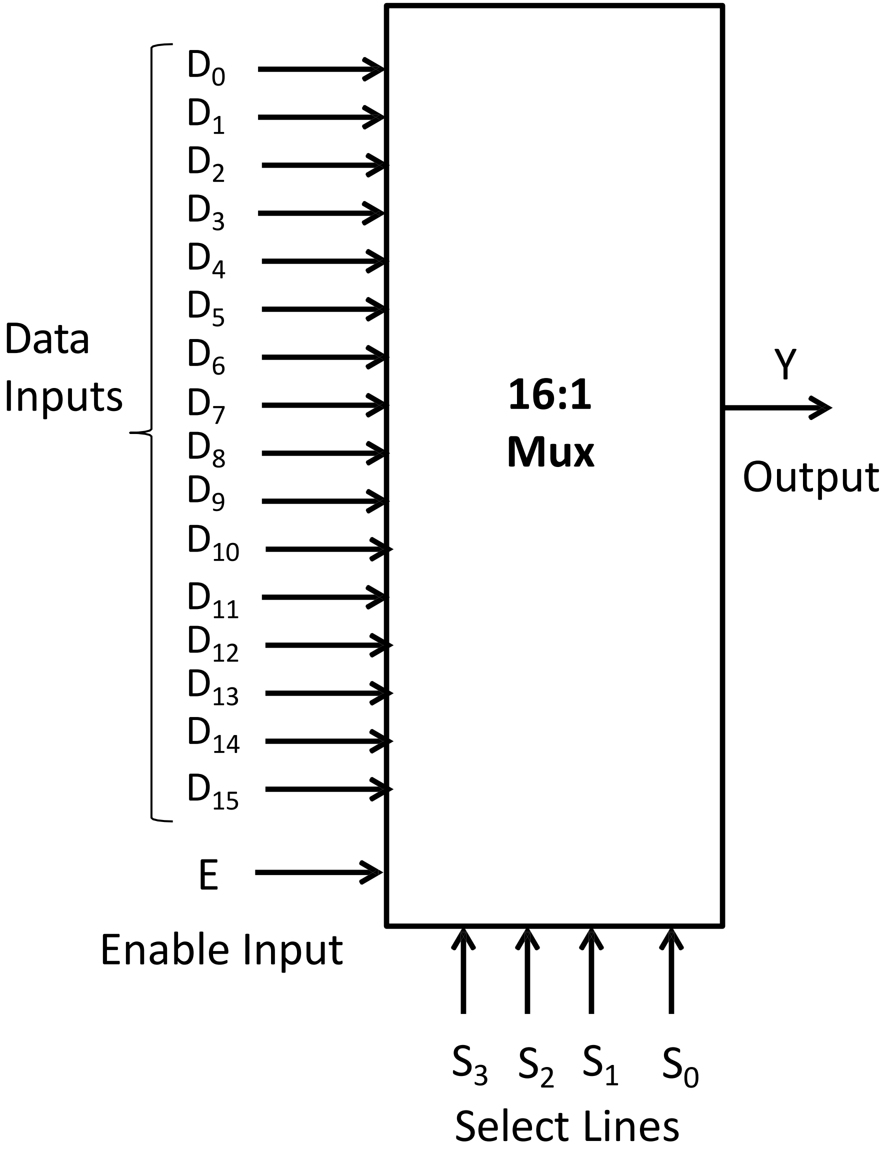


Truth Table

Enable i/p	Select i/p			Outp ut
E	S_2	S_1	S_0	Y
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	D_3
1	1	0	0	D_4
1	1	0	1	D_5
1	1	1	0	D_6
1	1	1	1	D_7

16:1 Multiplexer

Block Diagram



16:1 Multiplexer

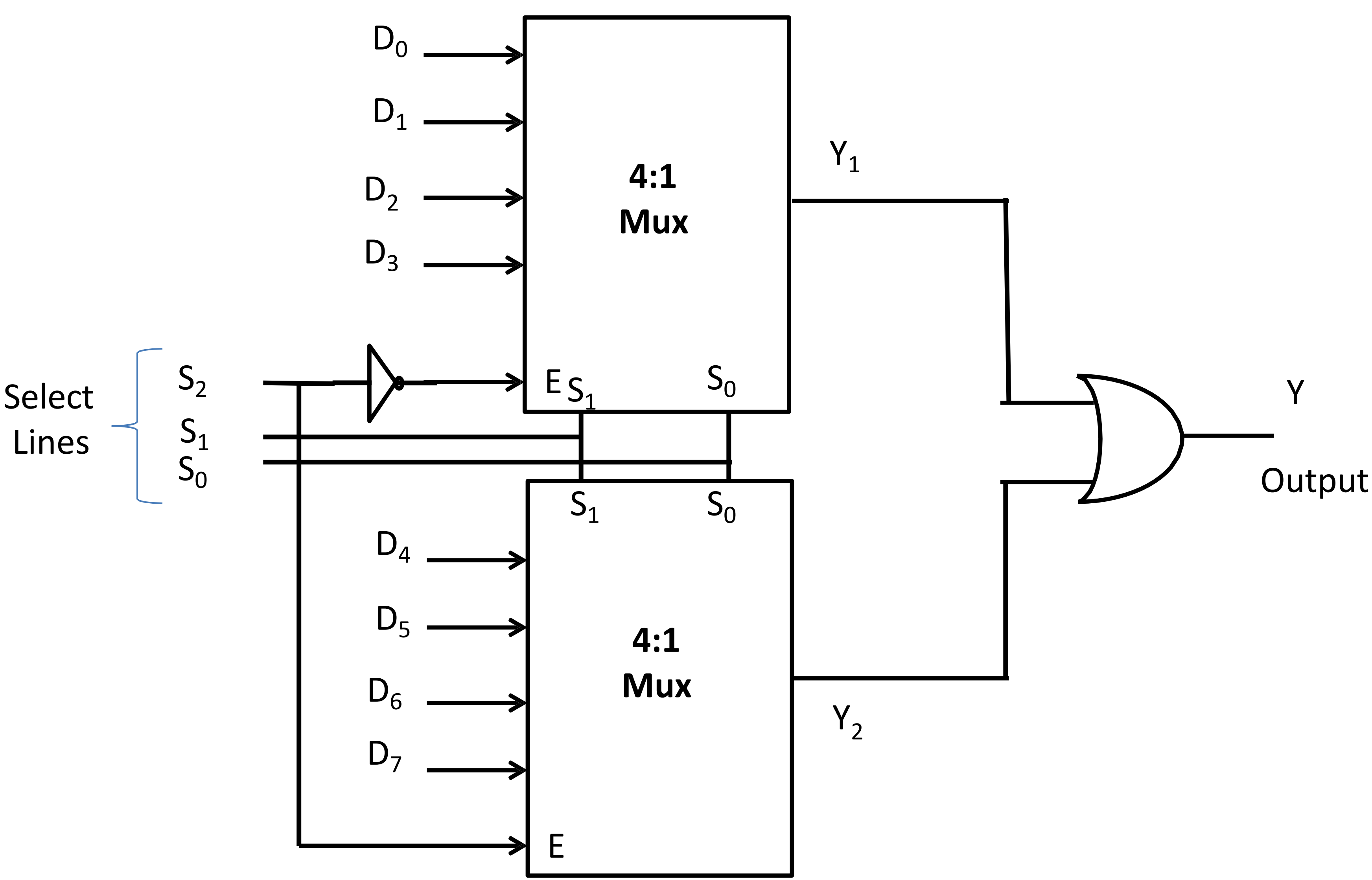
Truth Table

Enable	**Select Lines**				**Output**
E	**S_3**	**S_2**	**S_1**	**S_0**	**Y**
0	X	X	X	X	0
1	0	0	0	0	D_0
1	0	0	0	1	D_1
1	0	0	1	0	D_2
1	0	0	1	1	D_3
1	0	1	0	0	D_4
1	0	1	0	1	D_5
1	0	1	1	0	D_6
1	0	1	1	1	D_7
1	1	0	0	0	D_8
1	1	0	0	1	D_9
1	1	0	1	0	D_{10}
1	1	0	1	1	D_{11}
1	1	1	0	0	D_{12}
1	1	1	0	1	D_{13}
1	1	1	1	0	D_{14}
1	1	1	1	1	D15

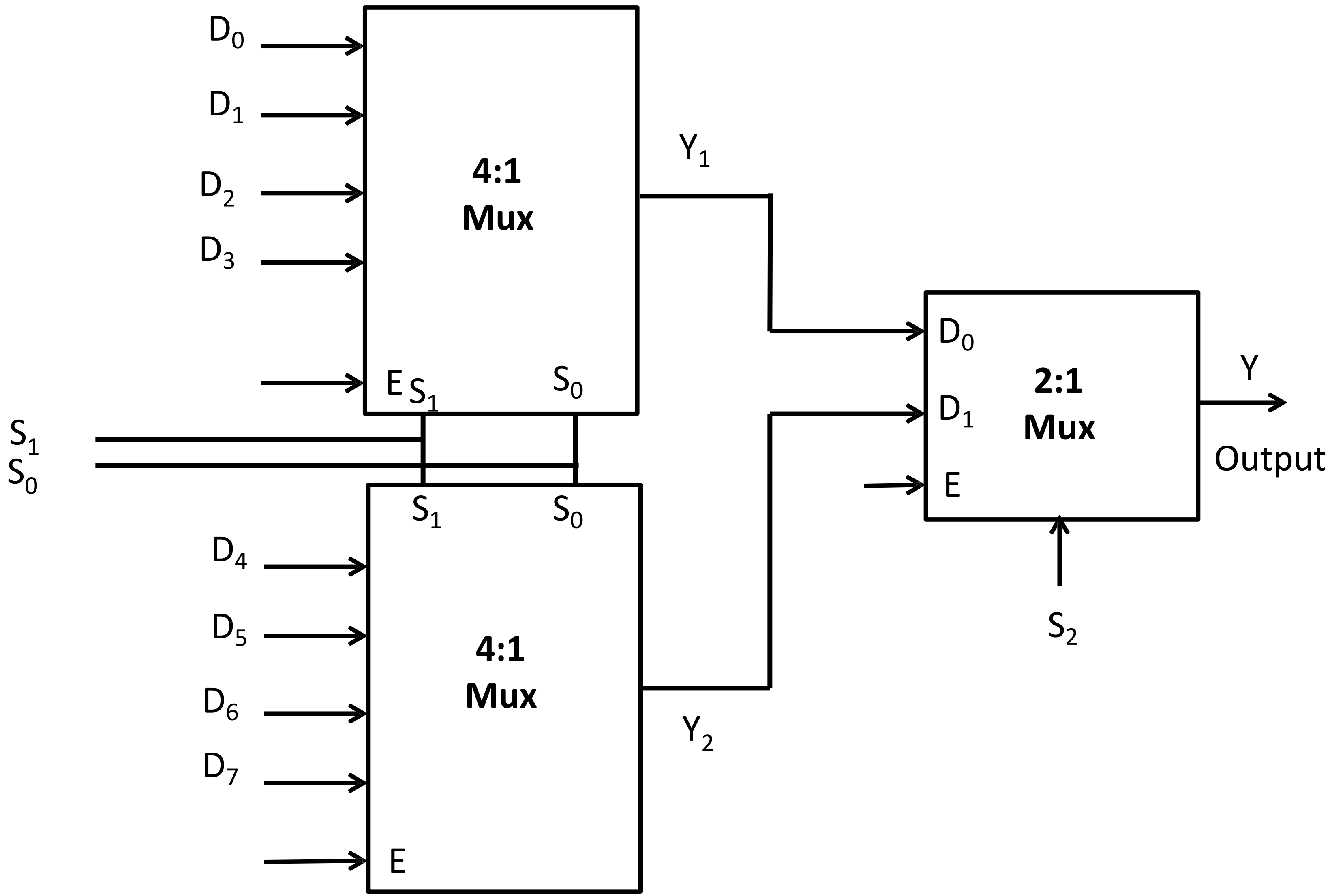
Mux Tree

- ✓ The multiplexers having more number of inputs can be obtained by cascading two or more multiplexers with less number of inputs. This is called as Multiplexer Tree.
- ✓ For example, 32:1 mux can be realized using two 16:1 mux and one 2:1 mux.

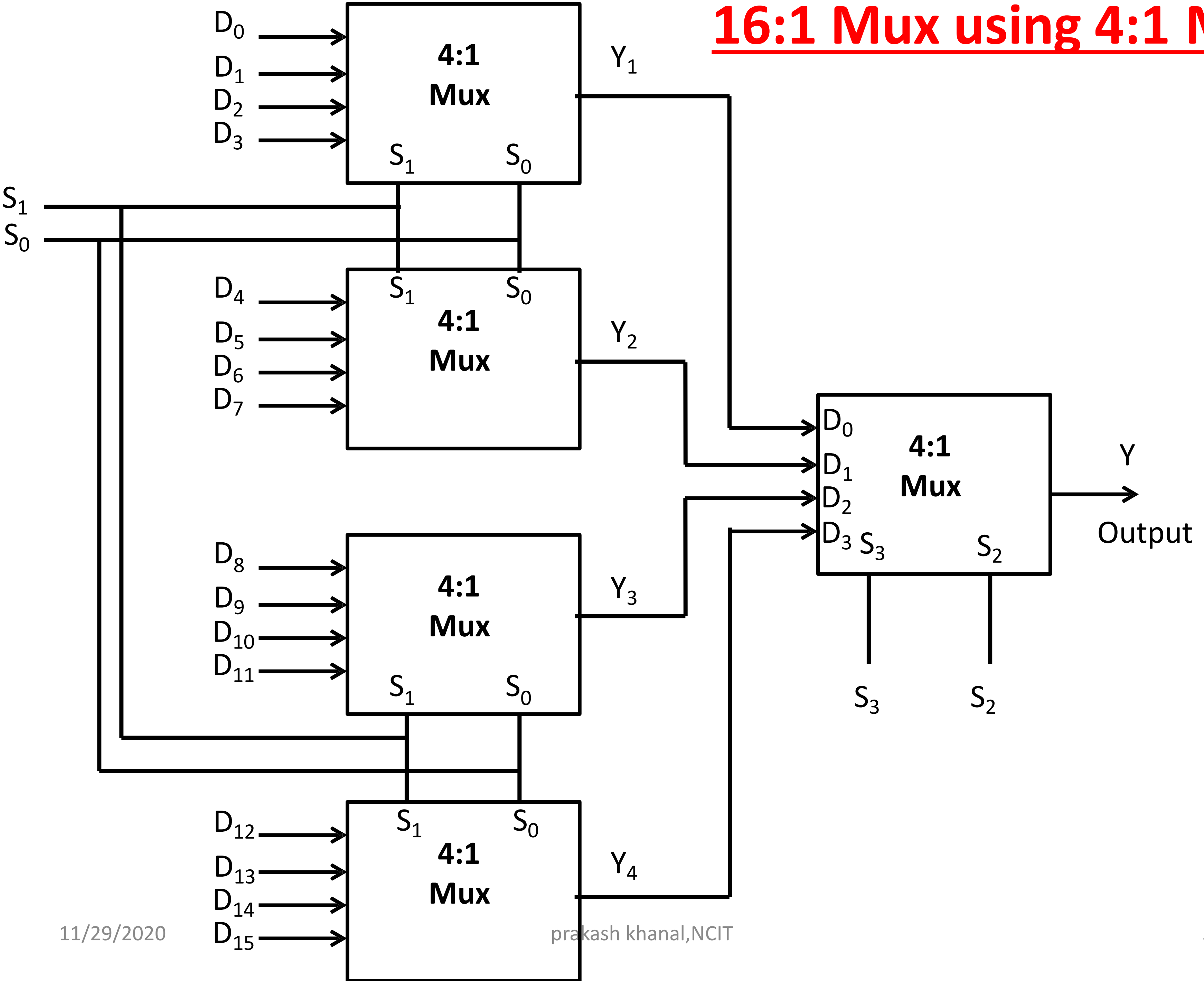
8:1 Multiplexer using 4:1 Multiplexer



8:1 Multiplexer using 4:1 Multiplexer



16:1 Mux using 4:1 Mux



Realization of Boolean expression using Mux

- ✓ We can implement any Boolean expression using Multiplexers.
- ✓ It reduces circuit complexity.
- ✓ It does not require any simplification

Example 1

Implement following Boolean expression using multiplexer

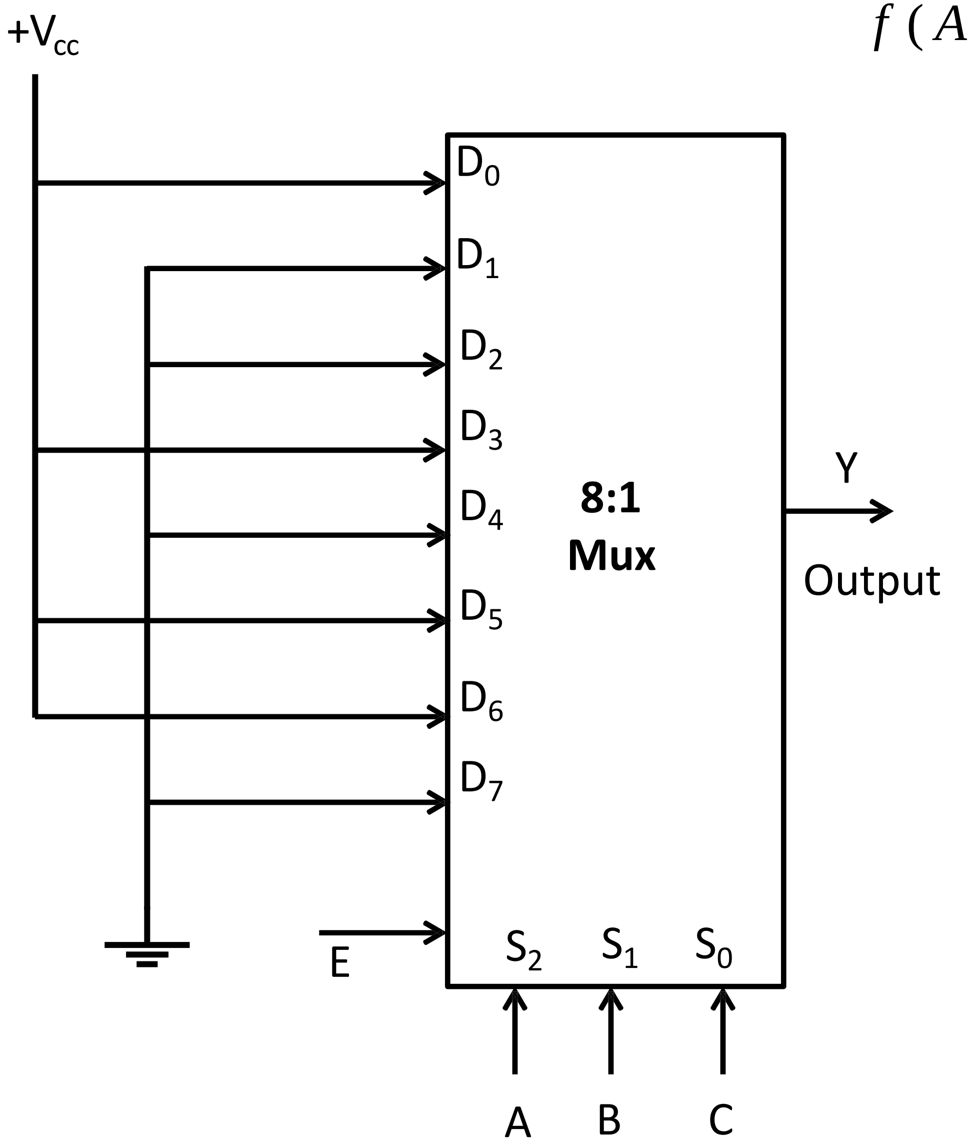
$$f(A, B, C) = \sum m(0, 3, 5, 6)$$

- ✓ Since there are three variables, therefore a multiplexer with three select input is required i.e. 8:1 multiplexer is required
- ✓ The 8:1 multiplexer is configured as below to implement given Boolean expression

Example 1

continue.....

$$f(A, B, C) = \sum m(0, 3, 5, 6)$$



Example 2

Implement following Boolean expression using multiplexer

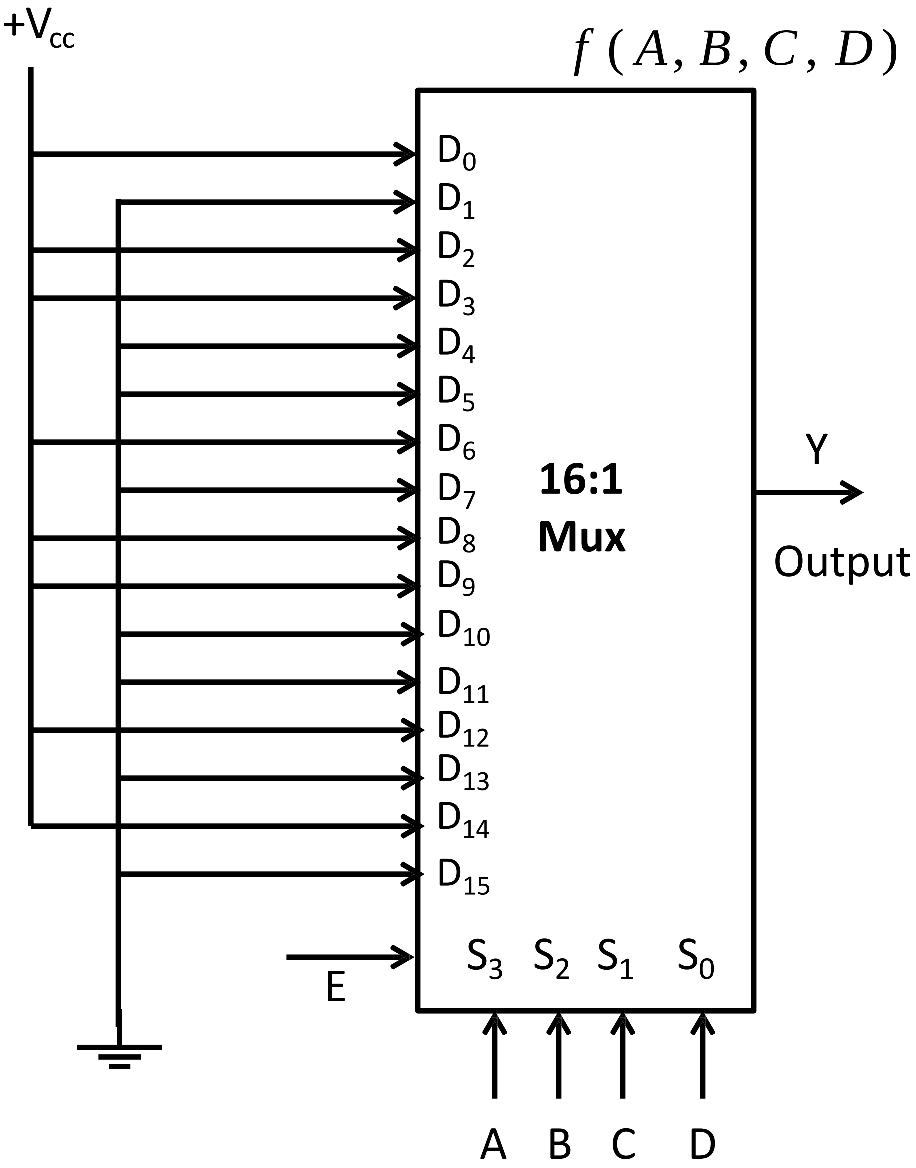
$$f(A, B, C, D) = \sum m(0, 2, 3, 6, 8, 9, 12, 14)$$

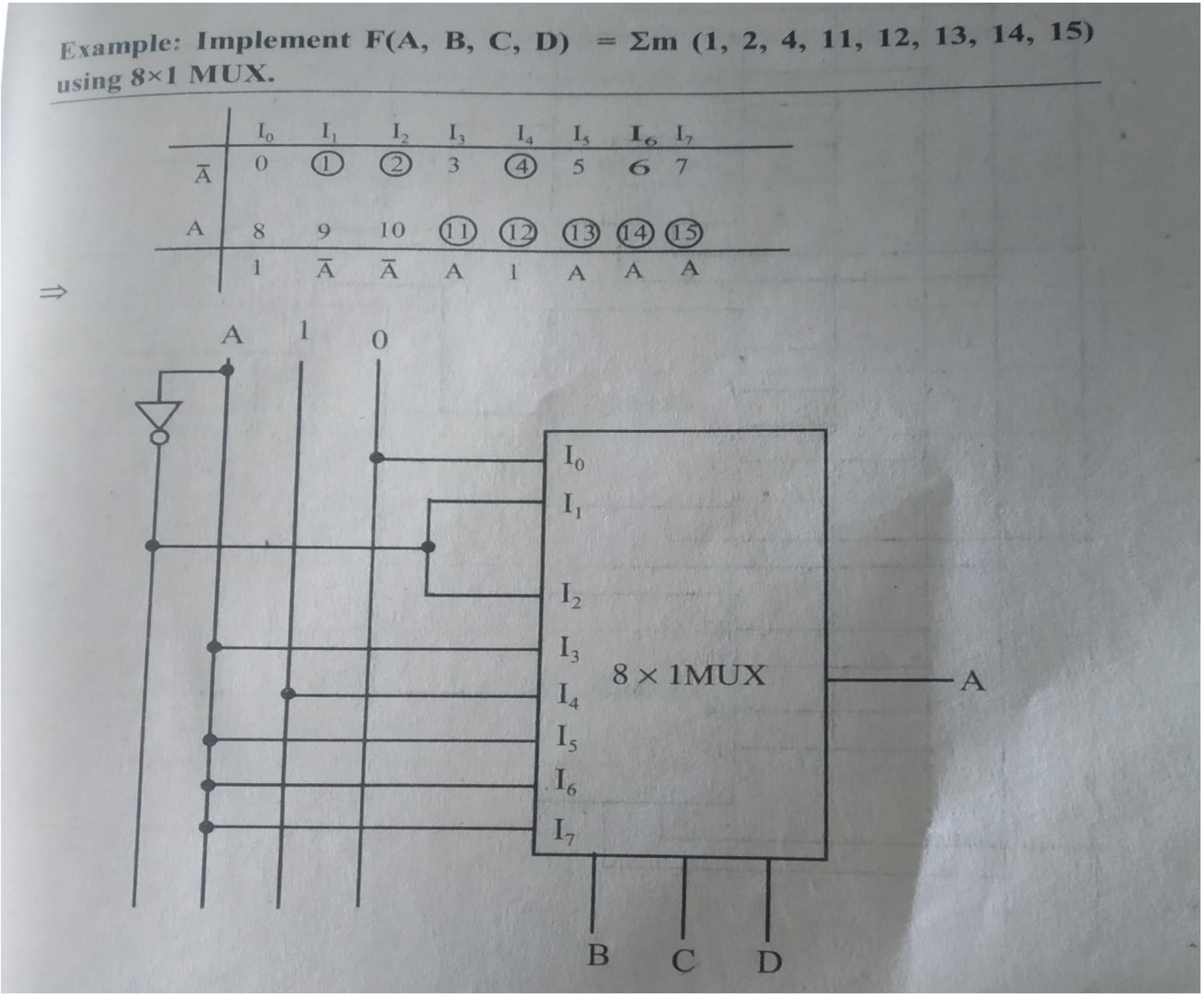
- ✓ Since there are four variables, therefore a multiplexer with four select input is required i.e. 16:1 multiplexer is required
- ✓ The 16:1 multiplexer is configured as below to implement given Boolean expression

Example 2

continue.....

$$f(A,B,C,D) = \sum m(0,2,3,6,8,9,12,14)$$





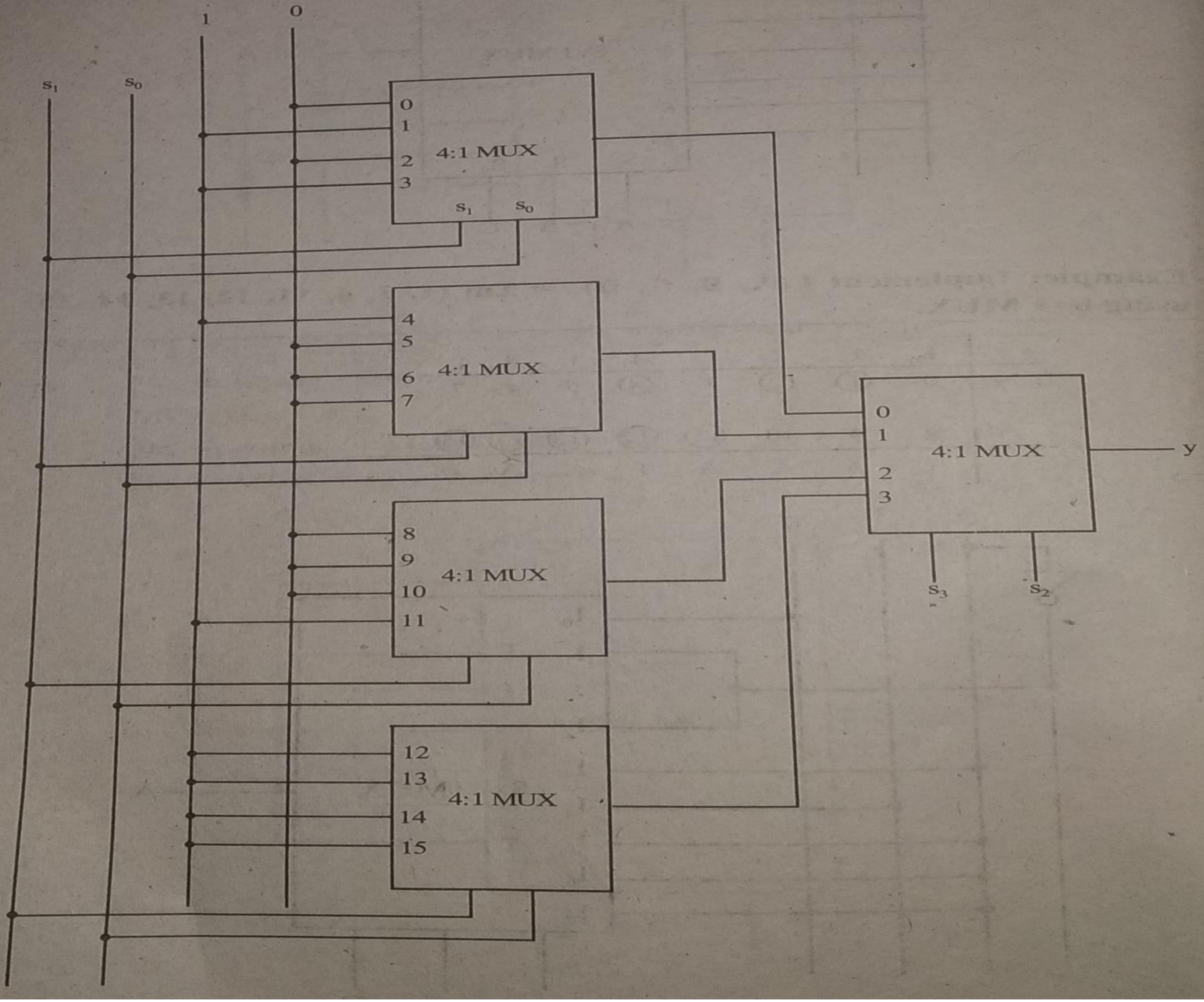
Example: Implement F(A, B, C, D) = Σm (1, 2, 4, 11, 12, 13, 14, 15) using 8×1 MUX.

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	0	①	②	3	④	5	6	7
A	8	9	10	⑪	⑫	⑬	⑭	⑮
	1	$\bar{A}$	$\bar{A}$	A	1	A	A	A

⇒

Example: Implement F(A, B, C, D) = Σm (1, 3, 4, 11, 12, 13, 14, 15) using 4×1 MUX.

⇒ No. of selection lines = 4 (S_3, S_2, S_1, S_0).



De-multiplexer

- ✓ A de-multiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs.
- ✓ At a time only one output line is selected by the select lines and the input is transmitted to the selected output line.
- ✓ It has only one input line, n number of output lines and m number of select lines.

Block Diagram of De-multiplexer

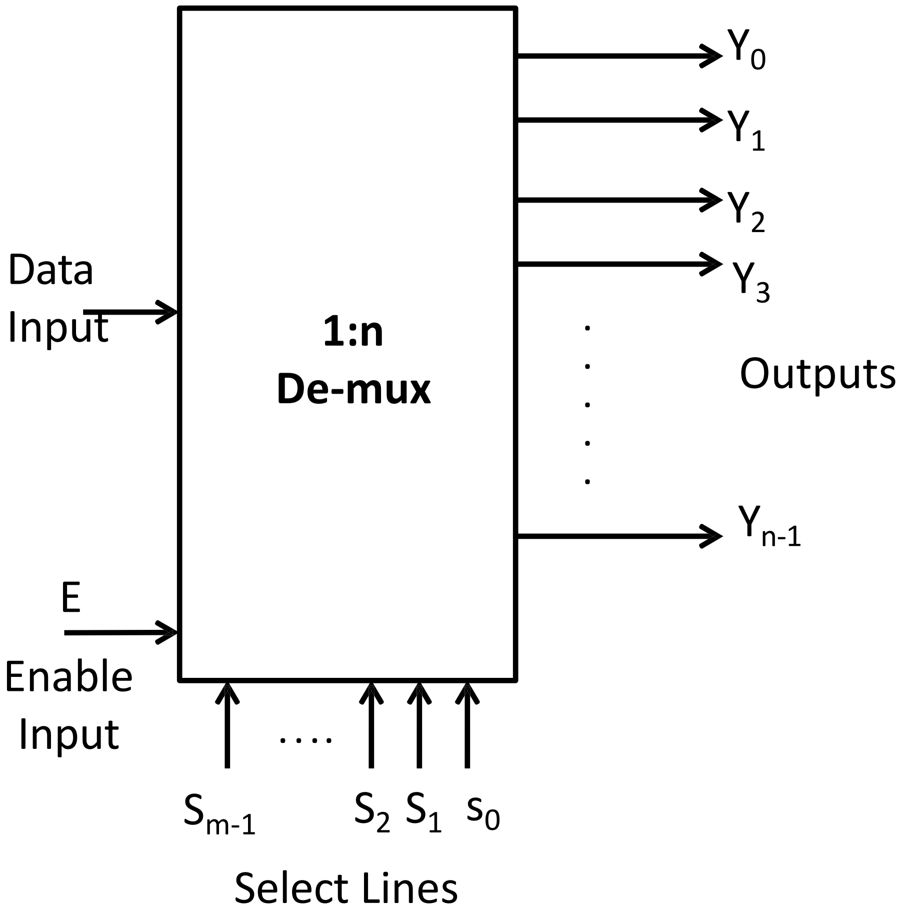


Fig. General Block Diagram

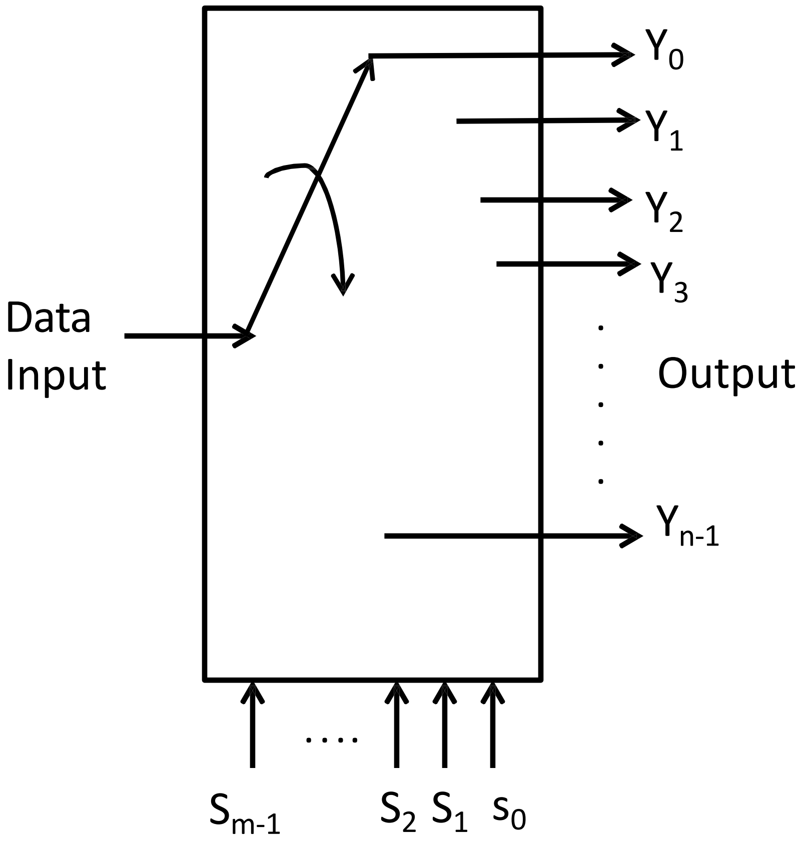


Fig. Equivalent Circuit

Relation between Data Output Lines & Select Lines

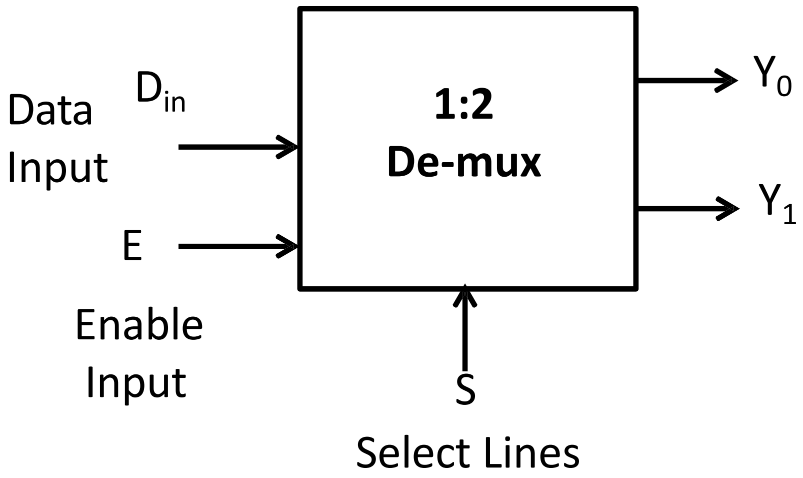
- ✓ In general de-multiplexer contains , n output lines, one input line and m select lines.
- ✓ To select n outputs we need m select lines such that $n=2^m$.

Types of De-multiplexers

- ✓1:2 De-multiplexer
- ✓1:4 De-multiplexer
- ✓1:8 De-multiplexer
- ✓1:16 De-multiplexer
- ✓1:32 De-multiplexer
- ✓1:64 De-multiplexer

and so on…………

1: 2 De-multiplexer

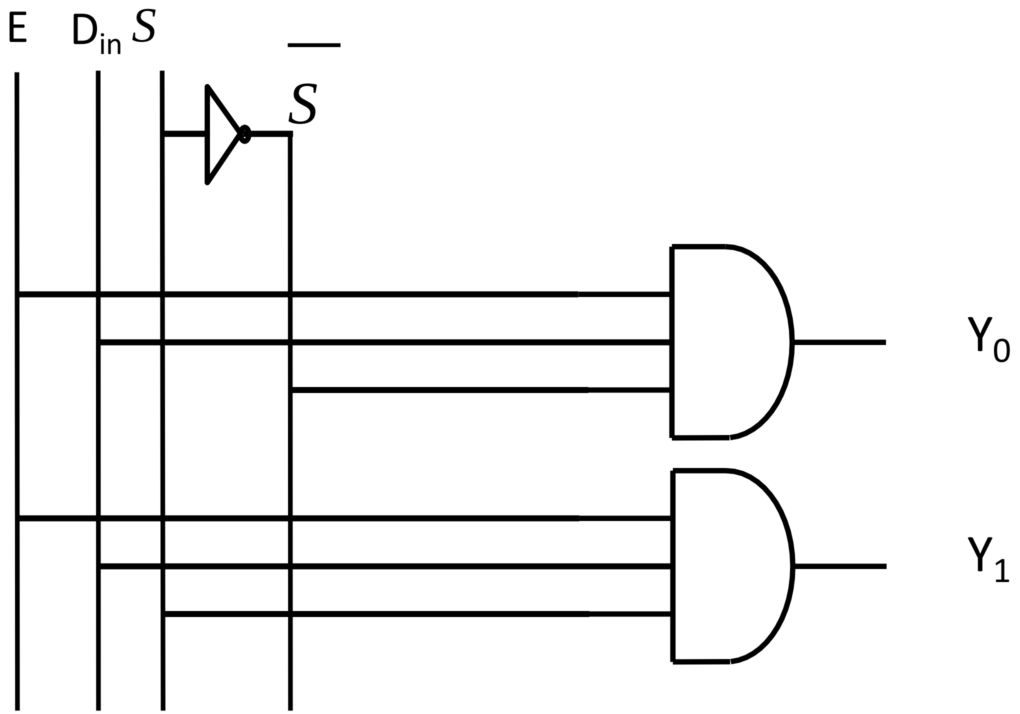


Block Diagram

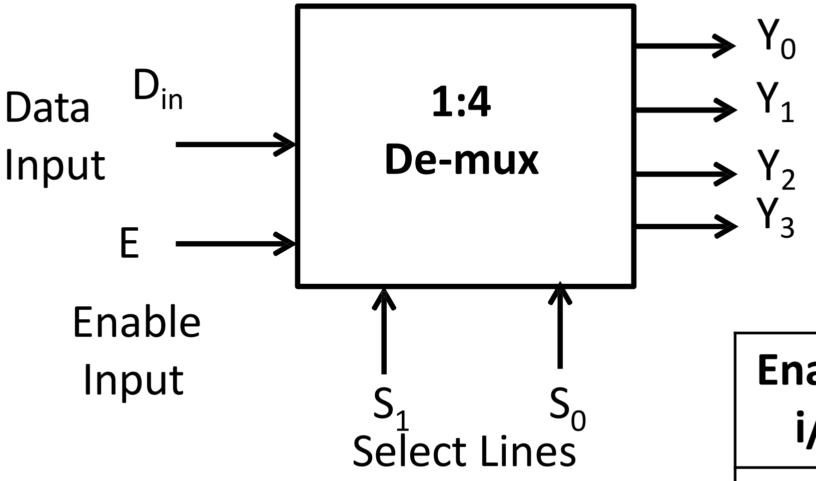
Truth Table

Enable i/p	**Select i/p**	**Outputs**	
E	**S**	Y_0	Y_1
0	X	0	0
1	0	D_{in}	0
1	1	0	D_{in}

1:2 De-mux using basic gates



1: 4 De-multiplexer

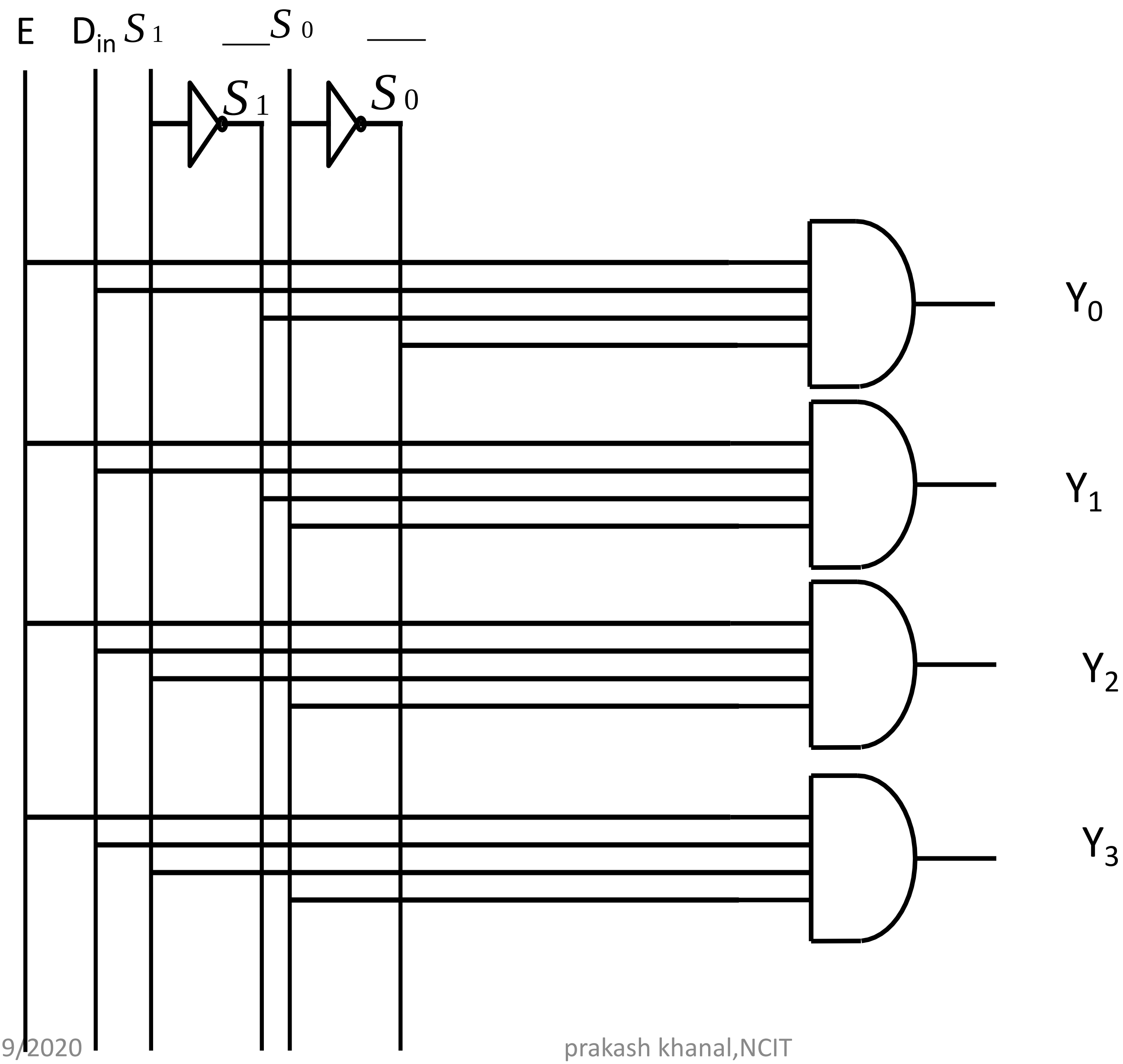


Block Diagram

Enable i/p	Select i/p		Outputs			
E	**S_1**	**S_0**	**Y_0**	**Y_1**	**Y_2**	**Y_3**
0	X	X	0	0	0	0
1	0	0	D_{in}	0	0	0
1	0	1	0	D_{in}	0	0
1	1	0	0	0	D_{in}	0
1	1	1	0	0	0	D_{in}

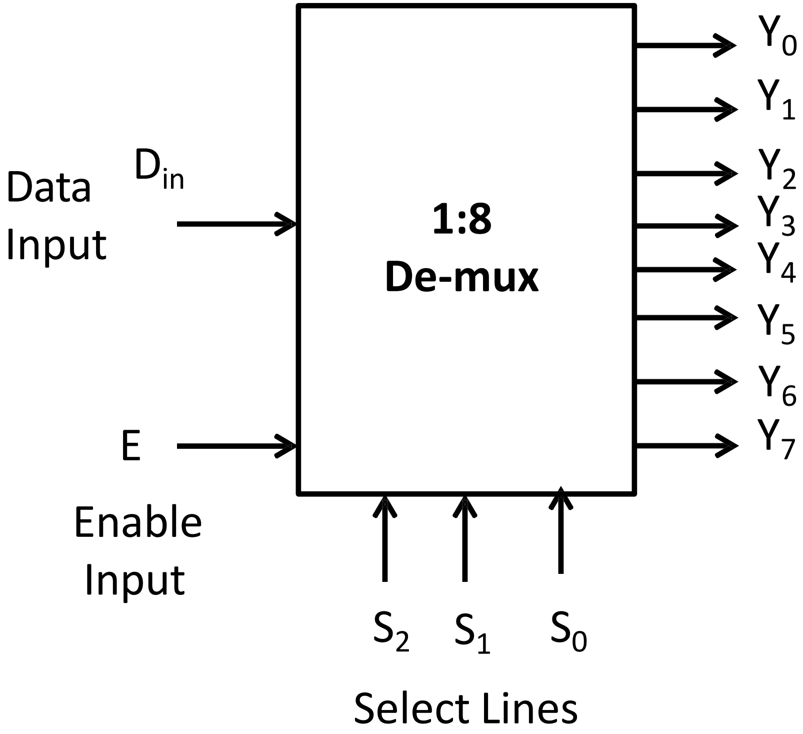
Truth Table

1:4 De-mux using basic gates



1: 8 De-multiplexer

Block Diagram



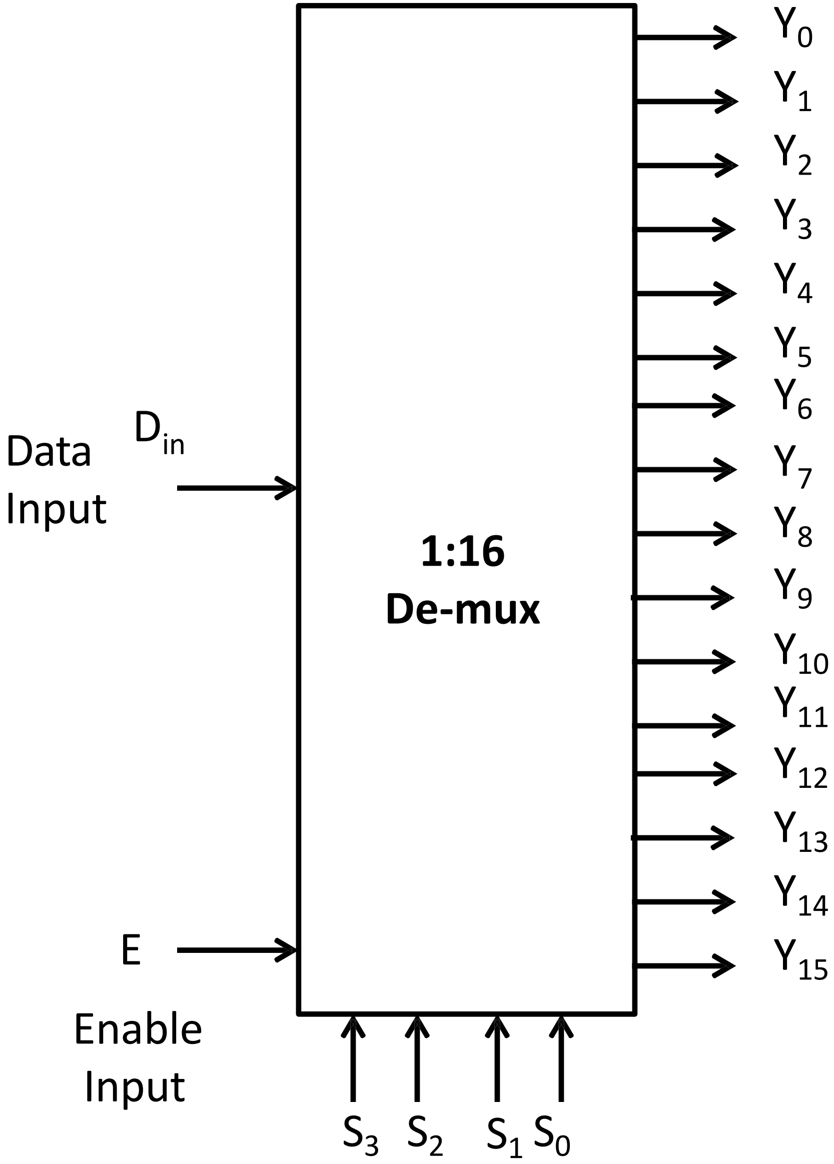
1: 8 De-multiplexer

Truth Table

Enable i/p	Select i/p			Outputs							
E	**S_2**	**S_1**	**S_0**	**Y_7**	**Y_6**	**Y_5**	**Y_4**	**Y_3**	**Y_2**	**Y_1**	**Y_0**
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	D_{in}
1	0	0	1	0	0	0	0	0	0	D_{in}	0
1	0	1	0	0	0	0	0	0	D_{in}	0	0
1	0	1	1	0	0	0	0	D_{in}	0	0	0
1	1	0	0	0	0	0	D_{in}	0	0	0	0
1	1	0	1	0	0	D_{in}	0	0	0	0	0
1	1	1	0	0	D_{in}	0	0	0	0	0	0
1	1	1	1	D_{in}	0	0	0	0	0	0	0

1: 16 De-multiplexer

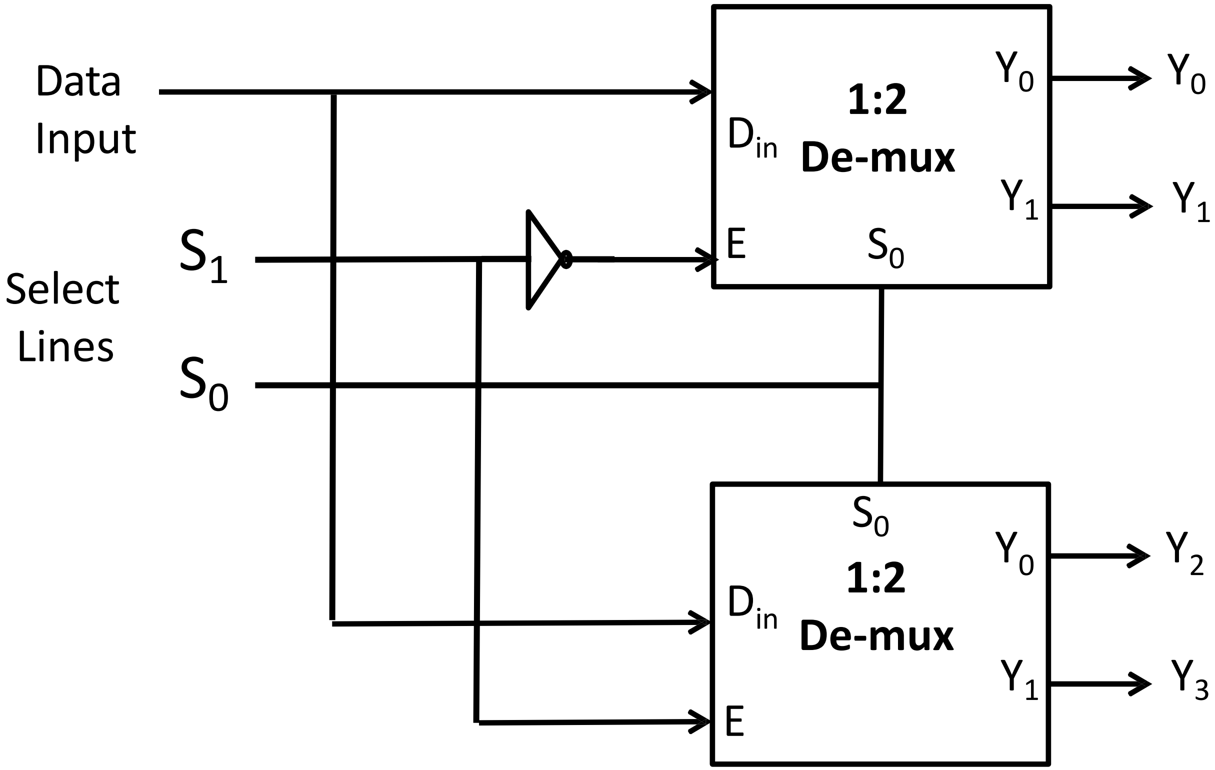
Block Diagram



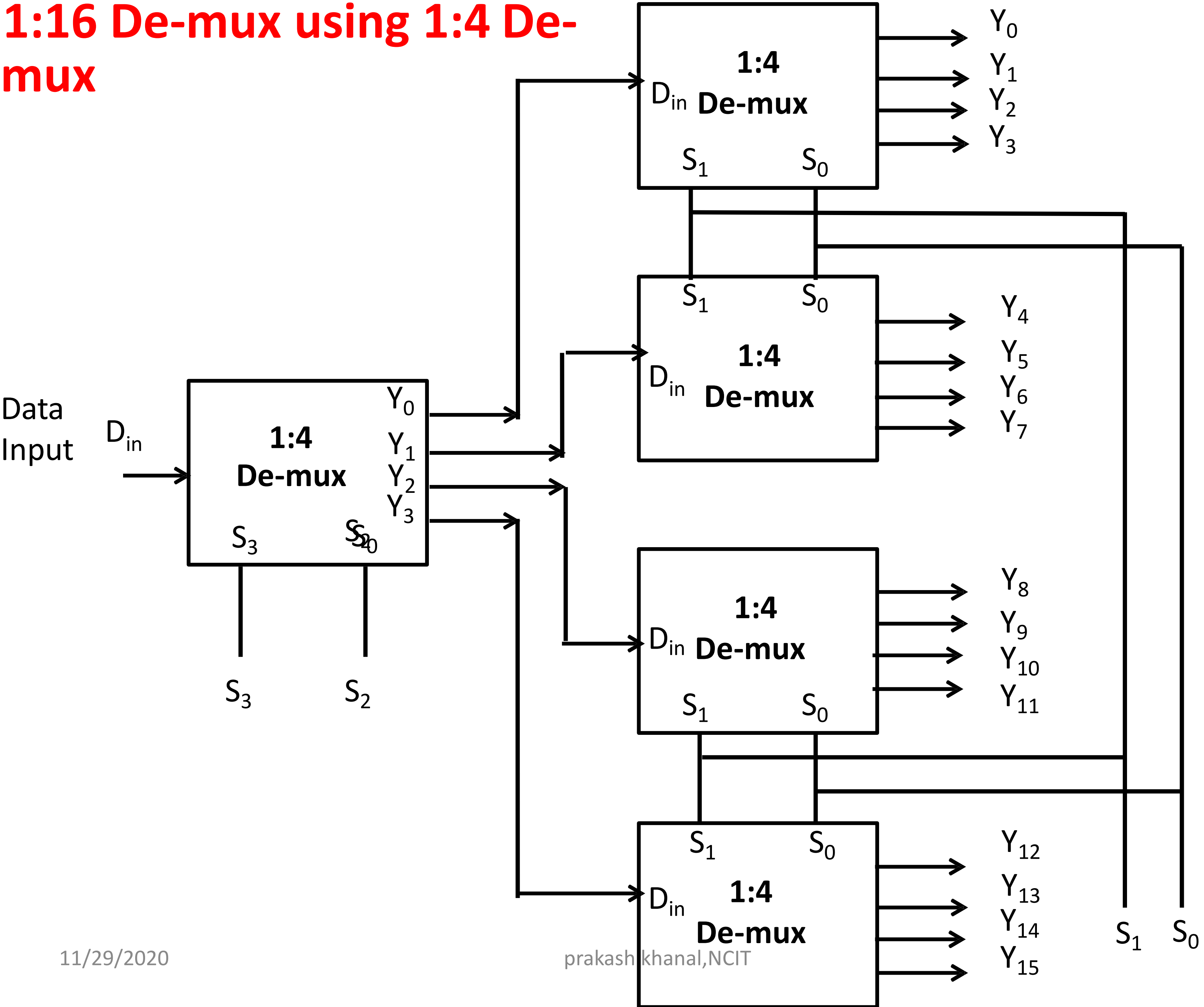
De-mux Tree

- ✓ Similar to multiplexer we can construct the de-multiplexer with more number of lines using de-multiplexer having less number of lines. This is call as “De-mux Tree”.

1:4 De-mux using 1:2 De-mux



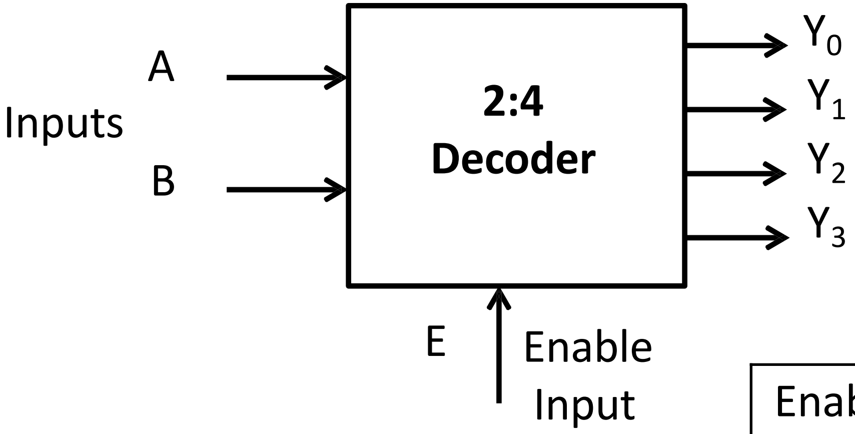
1:16 De-mux using 1:4 De-mux



Decoder

- ✓ Decoder is a combinational circuit.
- ✓ It converts n bit binary information at its input into a maximum of 2_n output lines.
- ✓ For example, if n=2 then we can design upto 2:4 decoder

2:4 Decoder



Block Diagram

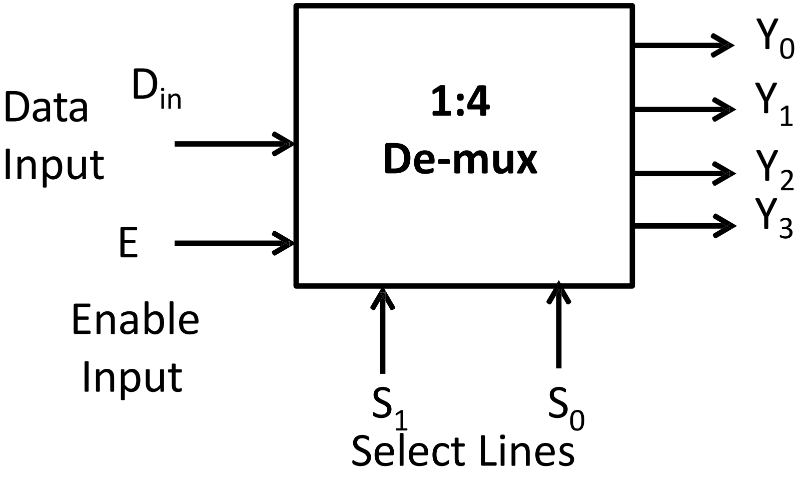
Truth Table

Enable i/p	Data Inputs		Outputs			
E	A	B	Y_0	Y_1	Y_2	Y_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

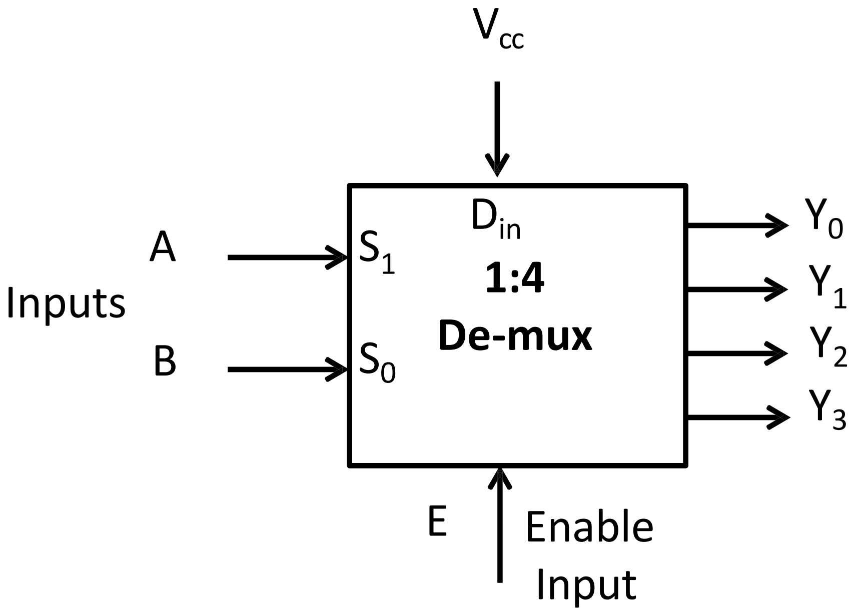
De-multiplexer as Decoder

- ✓ It is possible to operate a de-multiplexer as a decoder.
- ✓ Let us consider an example of 1:4 de-mux can be used as 2:4 decoder

1:4 De-multiplexer as 2:4 Decoder



1: 4 De-multiplexer



1: 4 De-multiplexer as 2:4 Decoder

Realization of Boolean expression using De-mux

- ✓ We can implement any Boolean expression using de-multiplexers.
- ✓ It reduces circuit complexity.
- ✓ It does not require any simplification

Example 1

Implement following Boolean expression using de-multiplexer

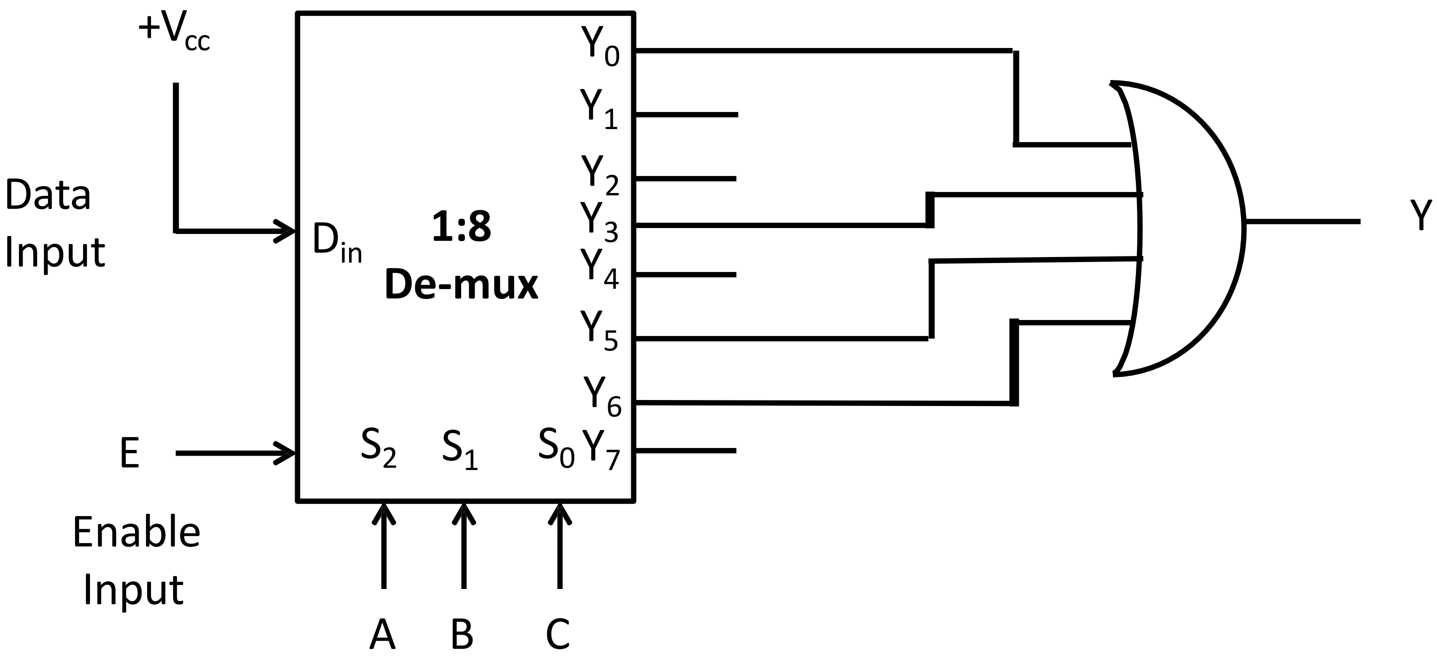
$$f(A, B, C) = \sum m(0, 3, 5, 6)$$

- ✓ Since there are three variables, therefore a de-multiplexer with three select input is required i.e. 1:8 de-multiplexer is required
- ✓ The 1:8 de-multiplexer is configured as below to implement given Boolean expression

Example 1

continue.....

$$f(A,B,C) = \sum m(0,3,5,6)$$



Example 2

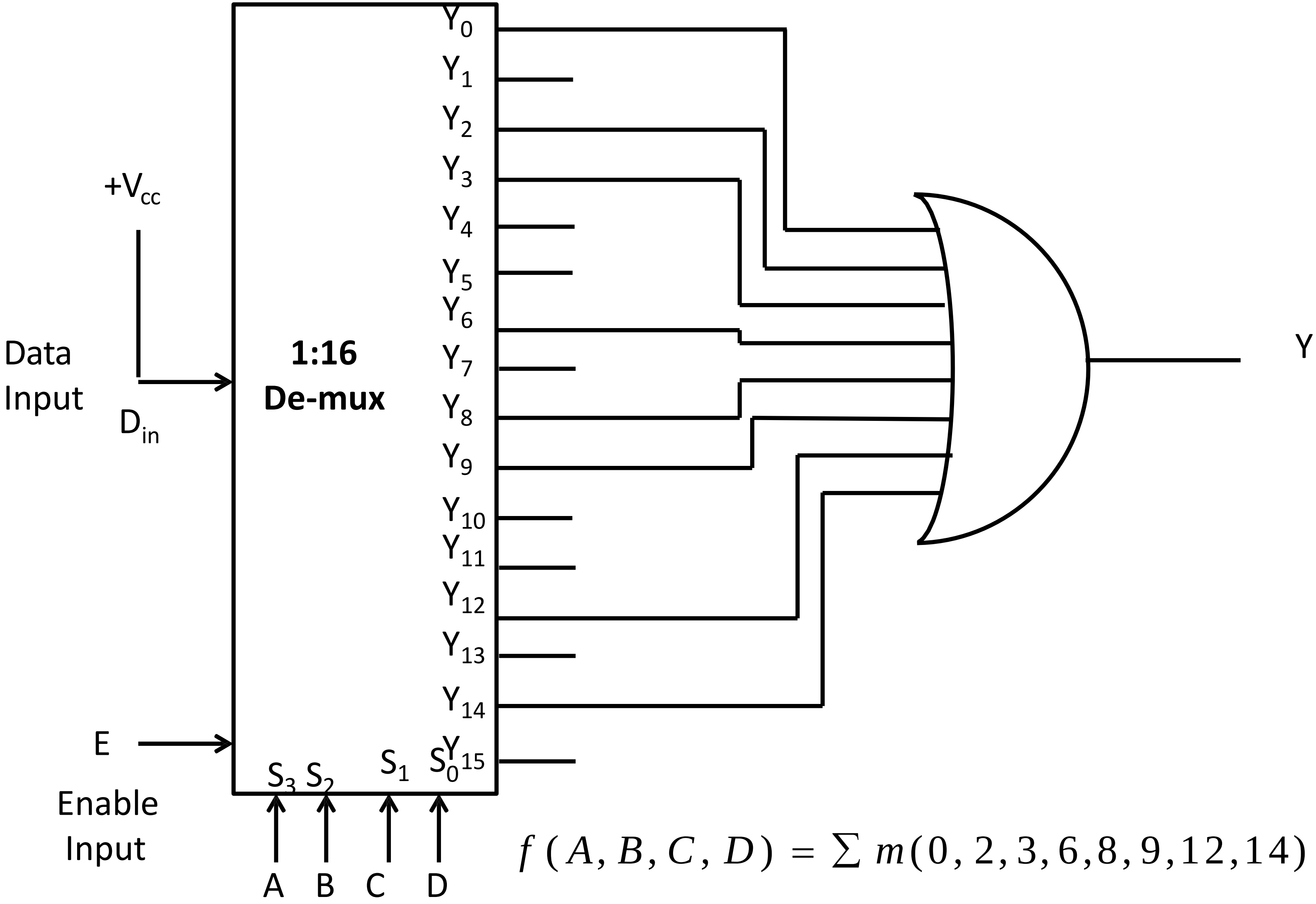
Implement following Boolean expression using de-multiplexer

$$f(A,B,C,D) = \sum m(0,2,3,6,8,9,12,14)$$

- ✓ Since there are four variables, therefore a de-multiplexer with four select input is required i.e. 1:16 de-multiplexer is required
- ✓ The 1:16 de-multiplexer is configured as below to implement given Boolean expression

Example 2

continue.....



$$f\,(A, B, C, D) = \sum m(0, 2, 3, 6, 8, 9, 12, 14)$$

Tristate Logic

- ✓ In digital electronics three-state, tri-state, or 3-state logic allows an output port to assume a high impedance state in addition to the 0 and 1 logic levels, effectively removing the output from the circuit.