CSE1003-Digital Logic Design

Module-4 COMBINATIONAL CIRCUITS-II

Module:3	**COMBINATIONAL CIRCUIT - I**	**4 hours**
Adder - Subtractor - Code Converter - Analyzing a Combinational Circuit		
Module:4	**COMBINATIONAL CIRCUIT –II**	**6 hours**
Binary Parallel Adder- Look ahead carry - Magnitude Comparator - Decoders – Encoders - Multiplexers –Demultiplexers.		

Remember

- **Combinational**
  - The outputs depend only on the current input values
  - It uses only logic gates
- **Sequential**
  - The outputs depend on the current and past input values
  - It uses logic gates and storage elements

Notes

- If there are *n* input variables, there are 2^n input combinations
- For each input combination, there is one output value
- Truth tables are used to list all possible combinations of inputs and corresponding output values

Basic Combinational Circuits

- Adders
- Subtractors
- Multipliers
- Comparators
- Decoders
- Encoders
- Multiplexers
- Demultiplexers

Part-1-Design Procedure

- Determine the inputs and outputs
- Assign a symbol for each
- Derive the truth table
- Get the simplified boolean expression for each output
- Draw the network diagram

Magnitude Comparator

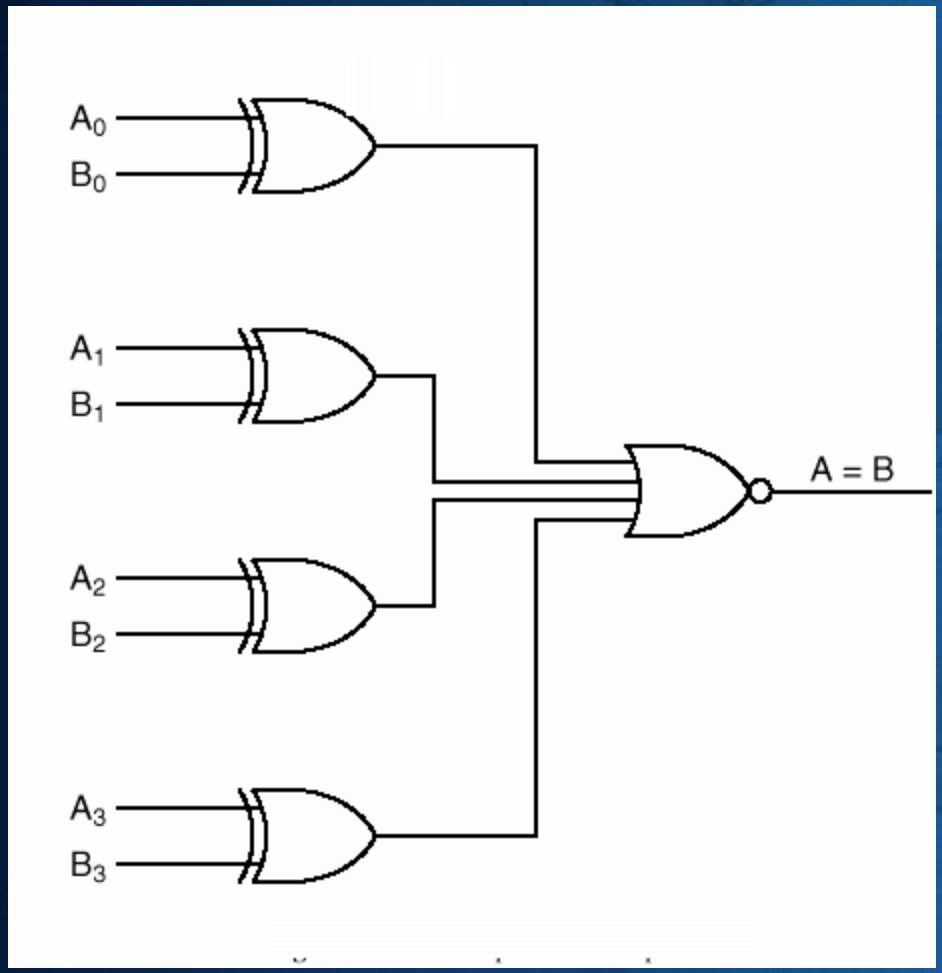
- A magnitude digital comparator is a combinational circuit that compares two digital or binary numbers (consider A and B) and determines their relative magnitudes in order to find out whether one number is equal, less than or greater than the other digital number.
- Three binary variables are used to indicate the outcome of the comparison as A>B, A<B, or A=B. The below figure shows the block diagram of a n-bit comparator which compares the two numbers of n-bit length and generates their relation between themselves.

MAGNITUDE COMPARATOR: DIGITAL COMPARATOR

- It is a combinational logic circuit.
- Digital Comparator is used to compare the value of two binary digits.
- There are two types of digital comparator (i) Identity Comparator (ii) Magnitude Comparator.
- IDENTITY COMPARATOR: This comparator has only one output terminal for when A=B, either A=B=1 (High) or A=B=0 (Low)
- MAGNITUDE COMPARATOR: This Comparator has three output terminals namely A>B, A=B, A<B. Depending on the result of comparison, one of these output will be high (1)
- Block Diagram of Magnitude Comparator is shown in Fig. 1

Identity Comparator

- Compare two input words



- Returns 1 if A=B, 0 otherwise

BLOCK DIAGRAM OF MAGNITUDE COMPARATOR

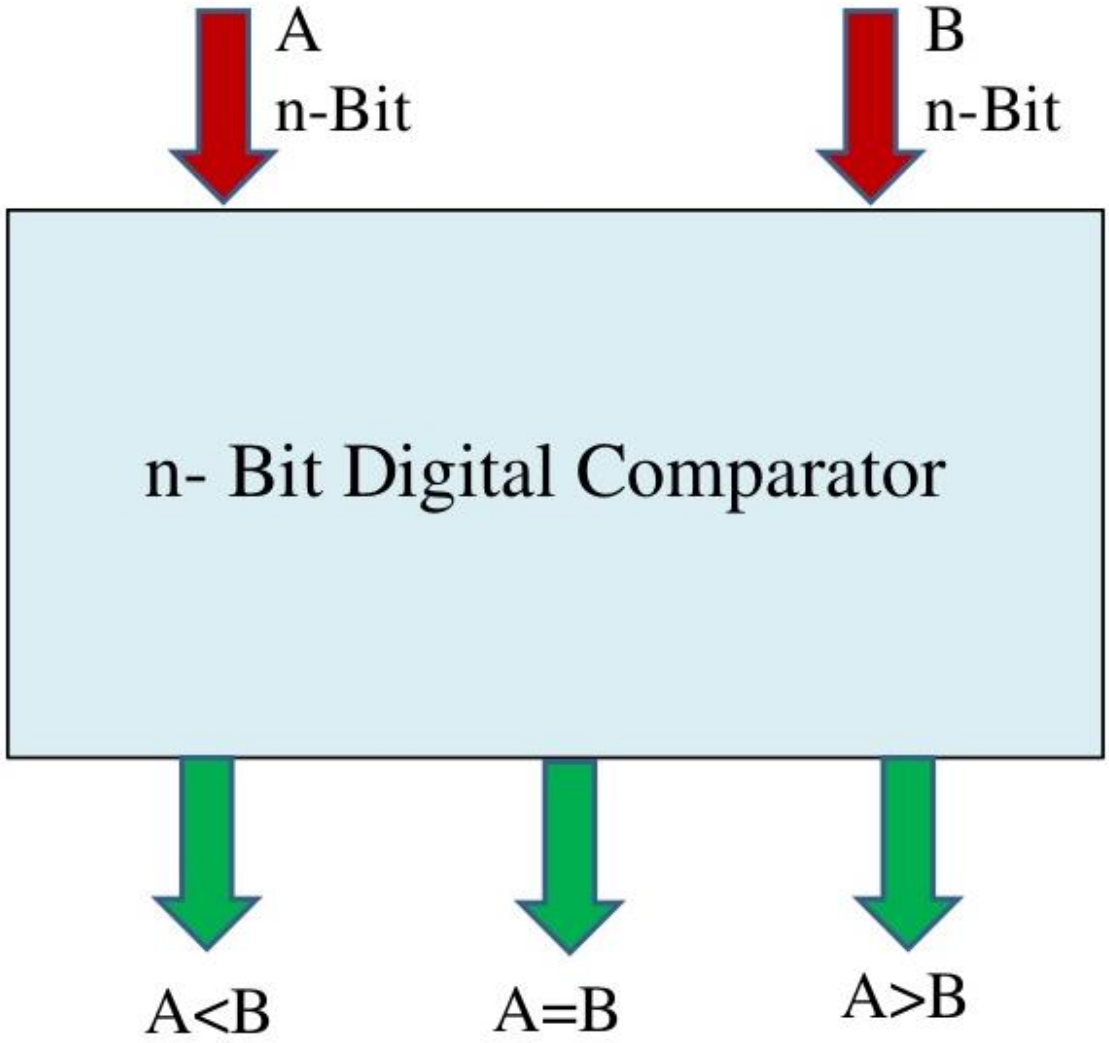


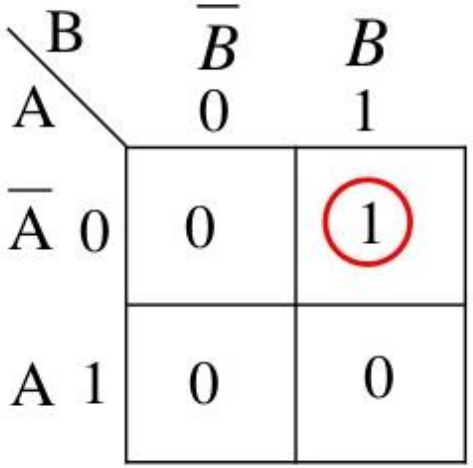
Fig. 1

1- Bit Magnitude Comparator:

- This magnitude comparator has two inputs A and B and three outputs A<B, A=B and A>B.
- This magnitude comparator compares the two numbers of single bits.
- Truth Table of 1-Bit Comparator

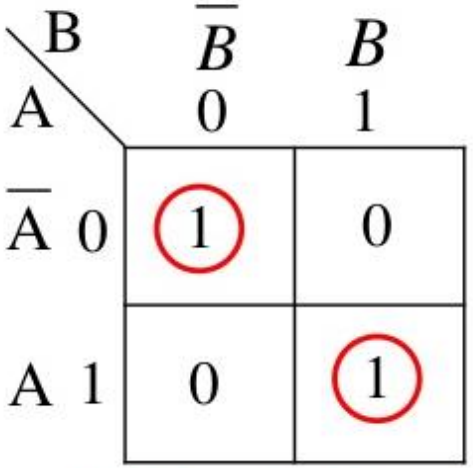
INPUTS		OUTPUTS		
A	B	Y_1 (A<B)	Y_2 (A=B)	Y_3 (A>B)
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

K-Maps For All Three Outputs :



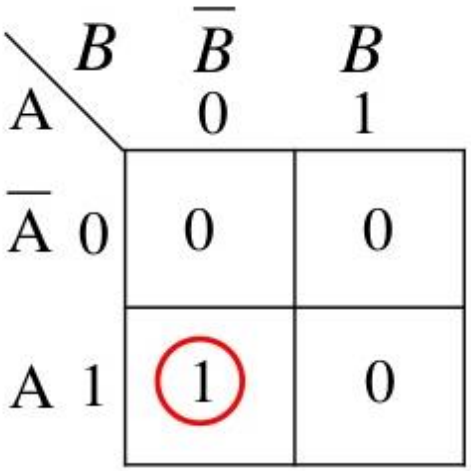
K-Map for Y_1 : A<B

$$Y_1 = \overline{A}B$$



K-Map for Y_2 : A=B

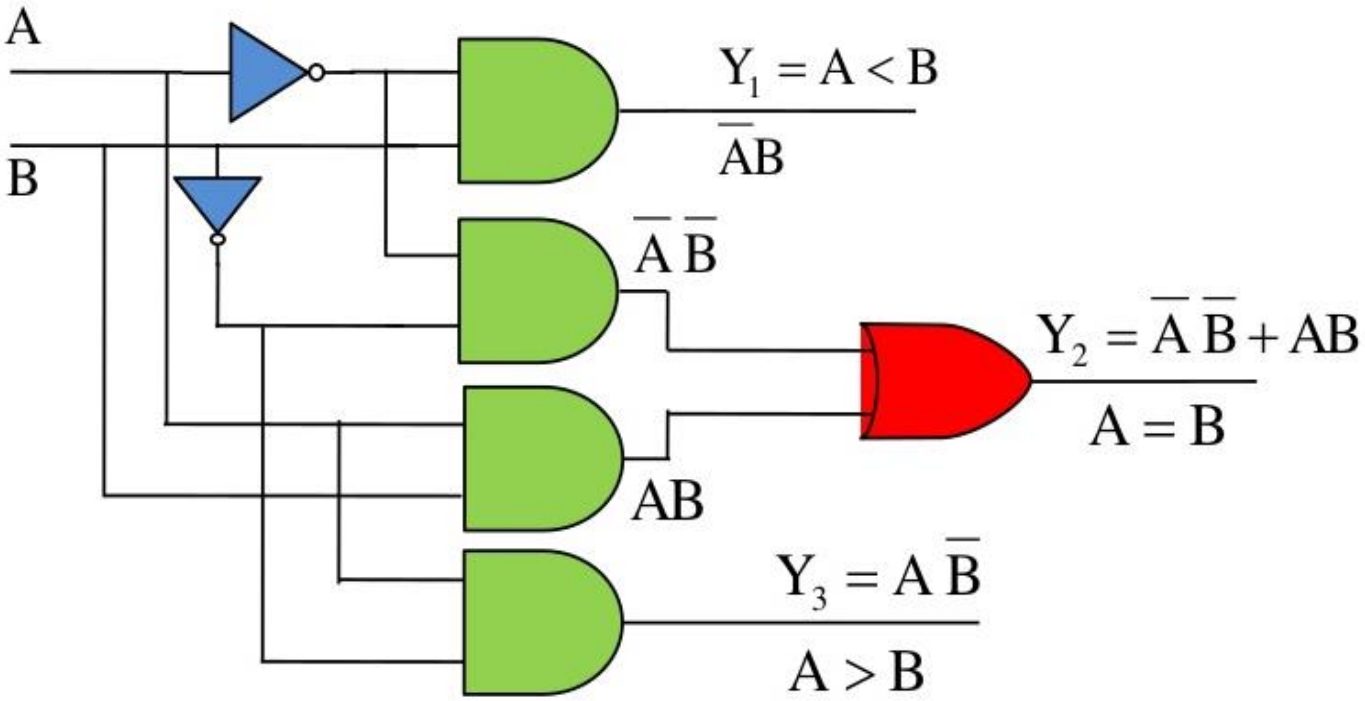
$$Y_2 = \overline{A}\,\overline{B} + AB$$



K-Map for Y_2 : A>B

$$Y_3 = A\overline{B}$$

Realization of One Bit Comparator



$$Y_1 = \overline{A}B$$

$$Y_2 = \overline{A}\,\overline{B} + AB$$

$$Y_3 = A\,\overline{B}$$

2-Bit Comparator:

- A comparator which is used to compare two binary numbers each of two bits is called a 2-bit magnitude comparator.
- Fig. 2 shows the block diagram of 2-Bit magnitude comparator.
- It has four inputs and three outputs.
- Inputs are A_0 ,A_1,B_0 and B_1 and Outputs are Y_1, Y_2 and Y_3

Fig. 2

GREATER THAN (A>B)

A_1	A_0	B_1	B_0
1	0	0	1
1	1	1	0
0	1	0	0

1. If A_1= 1 and B_1= 0 then A>B
2. If A_1 and B_1 are same, i.e A_1=B_1=1 or A_1=B_1=0 and A_0=1, B_0=0 then A>B

LESS THAN (A<B)

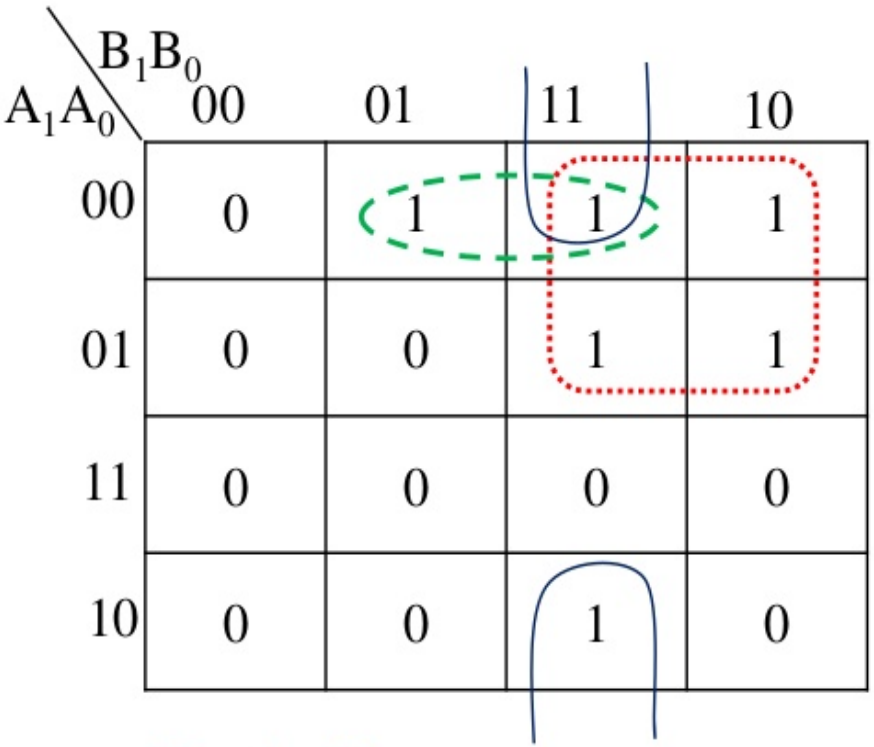
Similarly,

1. If A_1= B_1=1 and A_0= 0, B_0=1, then A<B
2. If A_1= B_1= 0 and A_0= 0, B_0=1 then A<B

TRUTH TABLE

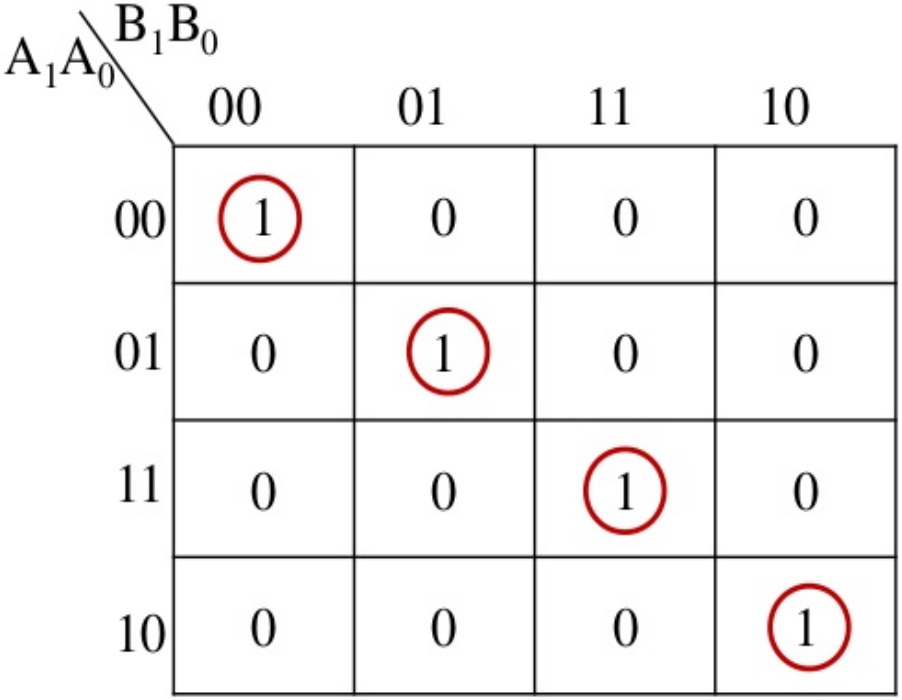
INPUT				OUTPUT		
A_1	A_0	B_1	B_0	Y_1=A<B	Y_2=(A=B)	Y_3=A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

K-Map for A<B:



A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

K-Map for A=B:



A_1A_0 \ B_1B_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

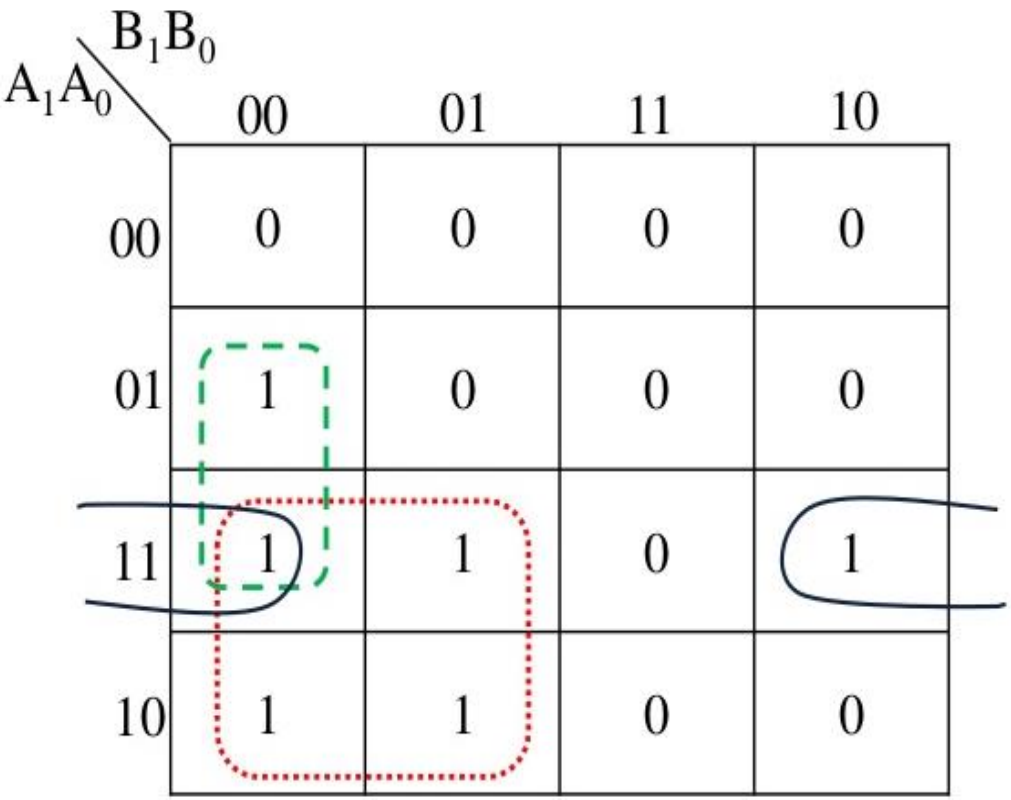
For A<B

$$Y_1 = \overline{A_1}\,\overline{A_0}\,B_0 + \overline{A_1}\,B_1 + \overline{A_0}\,B_1\,B_0$$

For A=B

$$Y_2 = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}\,A_0\,\overline{B_1}\,B_0 + A_1A_0B_1B_0 + A_1\,\overline{A_0}\,B_1\,\overline{B_0}$$

K-Map For A>B



$$Y_3 = A_0 \overline{B_1}\,\overline{B_0} + A_1 \overline{B_1} + A_1 A_0 \overline{B_0}$$

For A=B From K-Map

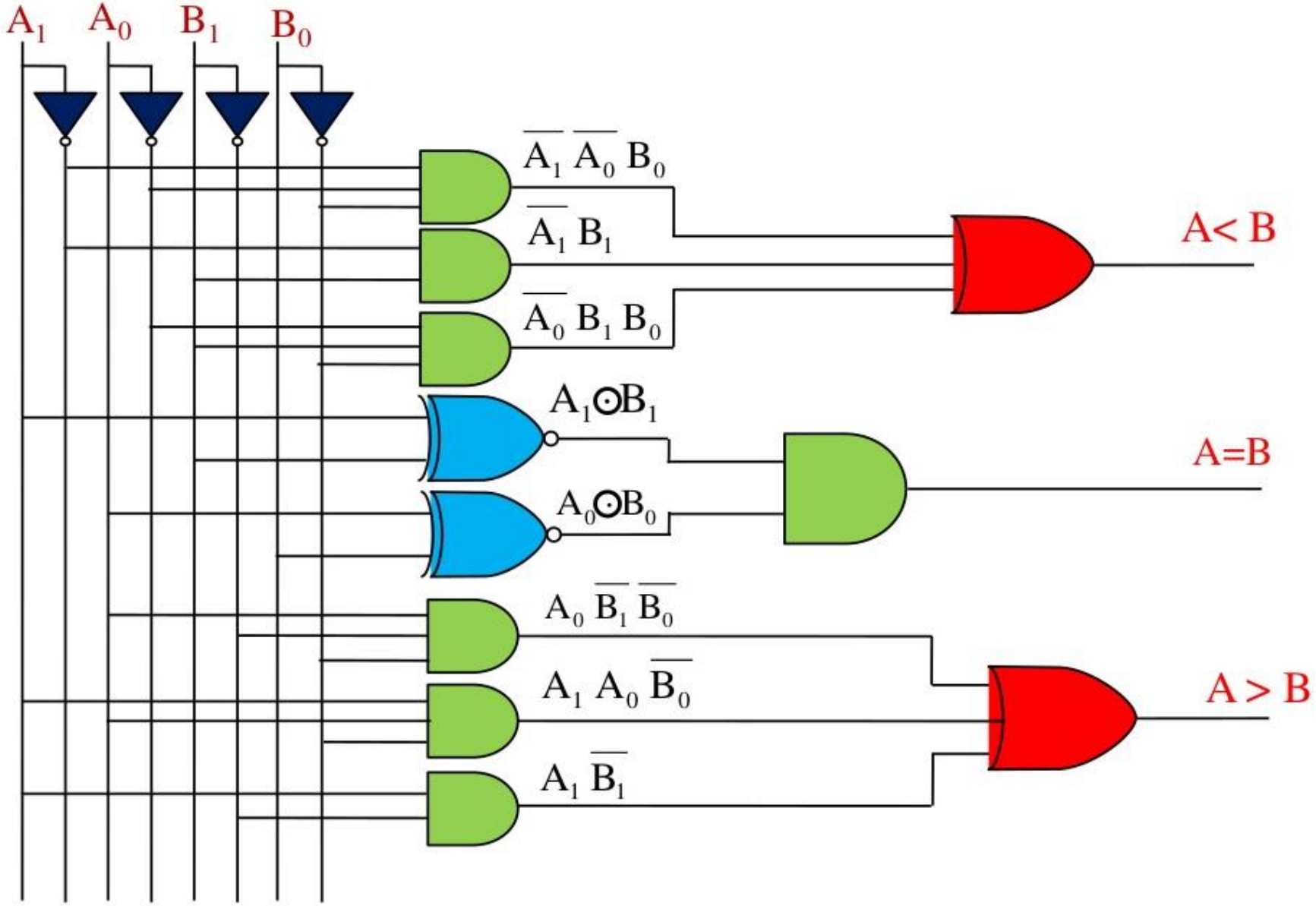
$$Y_2 = \overline{A_1}\,\overline{A_0}\,\overline{B_1}\,\overline{B_0} + \overline{A_1}\,A_0\,\overline{B_1}\,B_0 + \overline{A_1}\overline{A_0}B_1B_0 + A_1\,\overline{A_0}\,B_1\,\overline{B_0}$$

$$Y_2 = \overline{A_0}\,\overline{B_0}(\overline{A_1}\,\overline{B_1} + A_1B_1) + A_0B_0(\overline{A_1}\,\overline{B_1} + A_1B_1)$$

$$Y_2 = (\overline{A_1}\,\overline{B_1} + A_1B_1)\;(\overline{A_0}\,\overline{B_0} + A_0B_0)$$

$$Y_2 = (A_1 \odot B_1)\,(A_0 \odot B_0)$$

LOGIC DIAGRAM OF 2-BIT COMPARATOR:



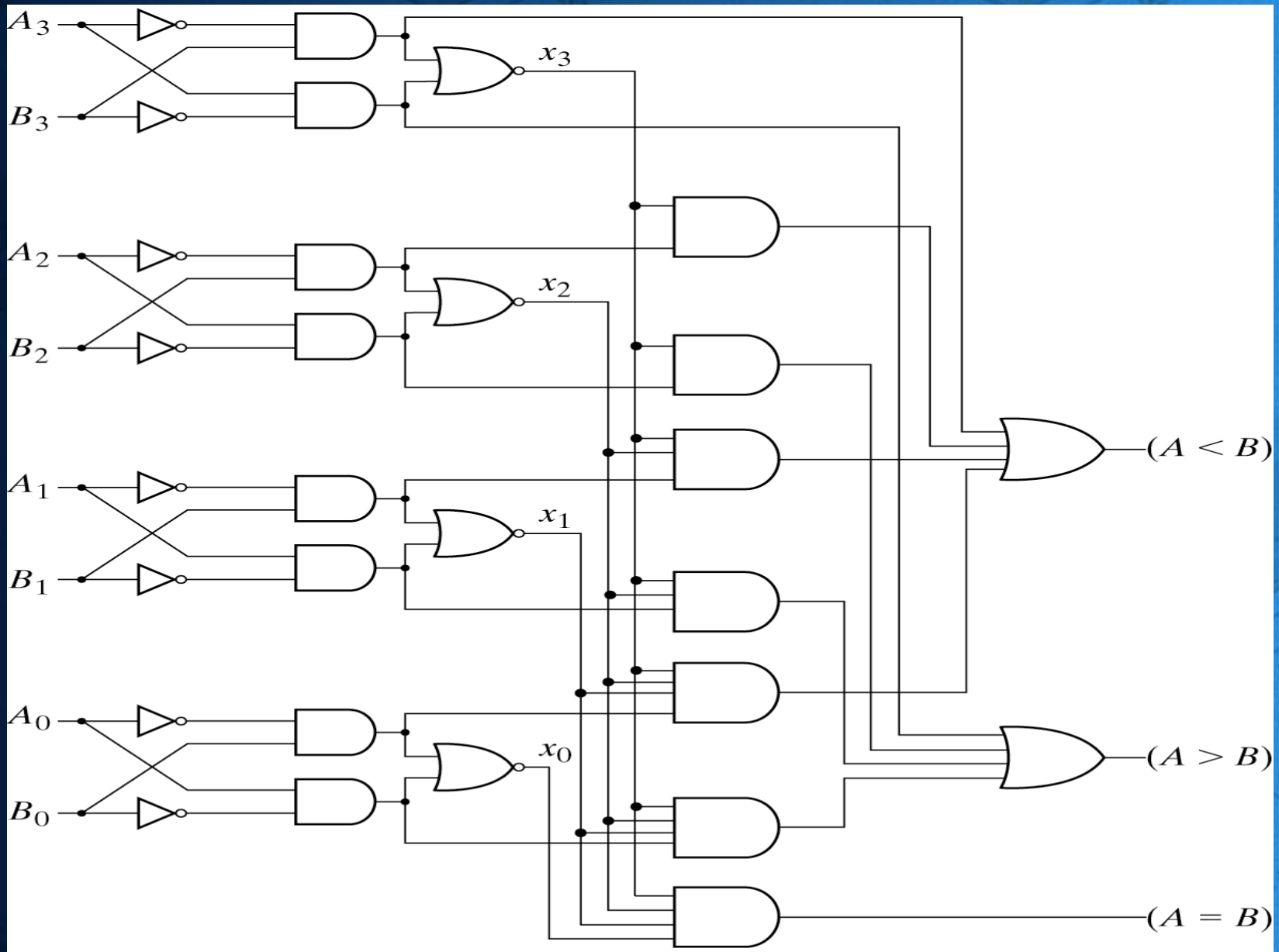


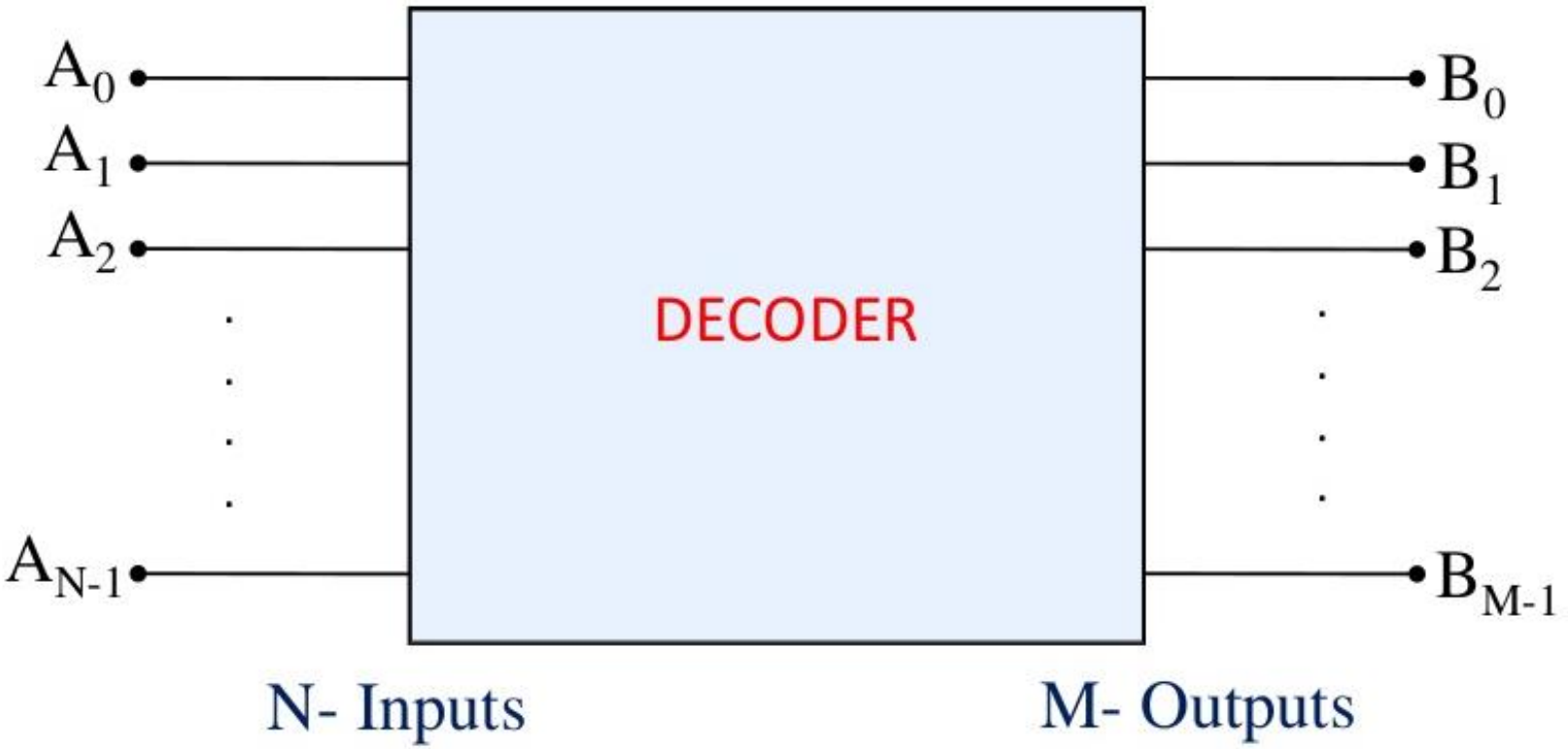
Fig. 4-17 4-Bit Magnitude Comparator

From course text book

DECODER

- A decoder is a combinational circuit.
- A decoder accepts a set of inputs that represents a binary number and activates only that output corresponding to the input number. All other outputs remain inactive.
- Fig. 1 shows the block diagram of decoder with 'N' inputs and 'M' outputs.
- There are 2^N possible input combinations, for each of these input combination only one output will be HIGH (active) all other outputs are LOW
- Some decoder have one or more ENABLE (E) inputs that are used to control the operation of decoder.

BLOCK DIAGRAM OF DECODER



Only one output is High for each input

Fig. 1

2 to 4 Line Decoder:

- Block diagram of 2 to 4 decoder is shown in fig. 2
- A and B are the inputs. (No. of inputs =2)
- No. of possible input combinations: $2^2=4$
- No. of Outputs : $2^2=4$, they are indicated by D_0, D_1, D_2 and D_3
- From the Truth Table it is clear that each output is "1" for only specific combination of inputs.

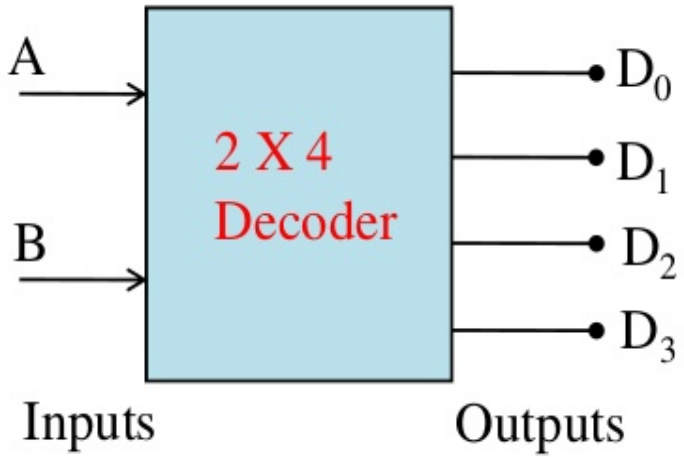


Fig. 2

TRUTH TABLE

INPUTS		OUTPUTS			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

BOOLEAN EXPRESSION:

From Truth Table

$$D_0 = \overline{A}\,\overline{B} \qquad D_1 = \overline{A}\,B$$

$$D_2 = A\,\overline{B} \qquad D_3 = AB$$

LOGIC DIAGRAM:

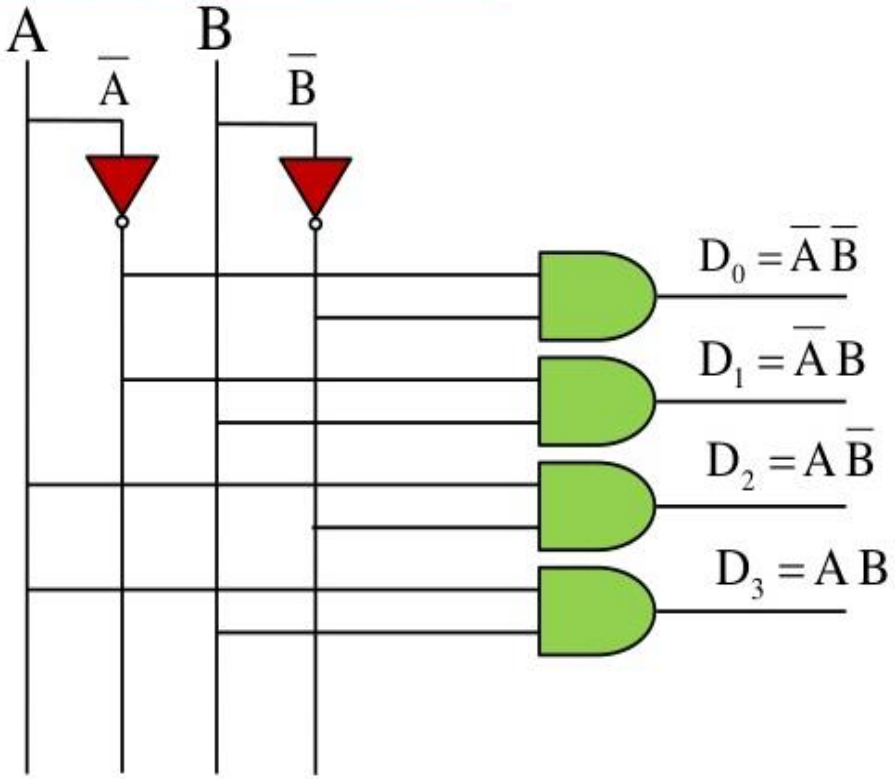
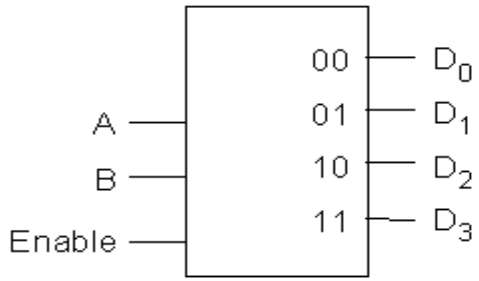


Fig. 3

Decoder

- n by 2^n decoder
  - Converts information from n input lines into 2^n output lines
- Example -2x4 Decoder, 3x8 Decoder, 4x16 Decoder

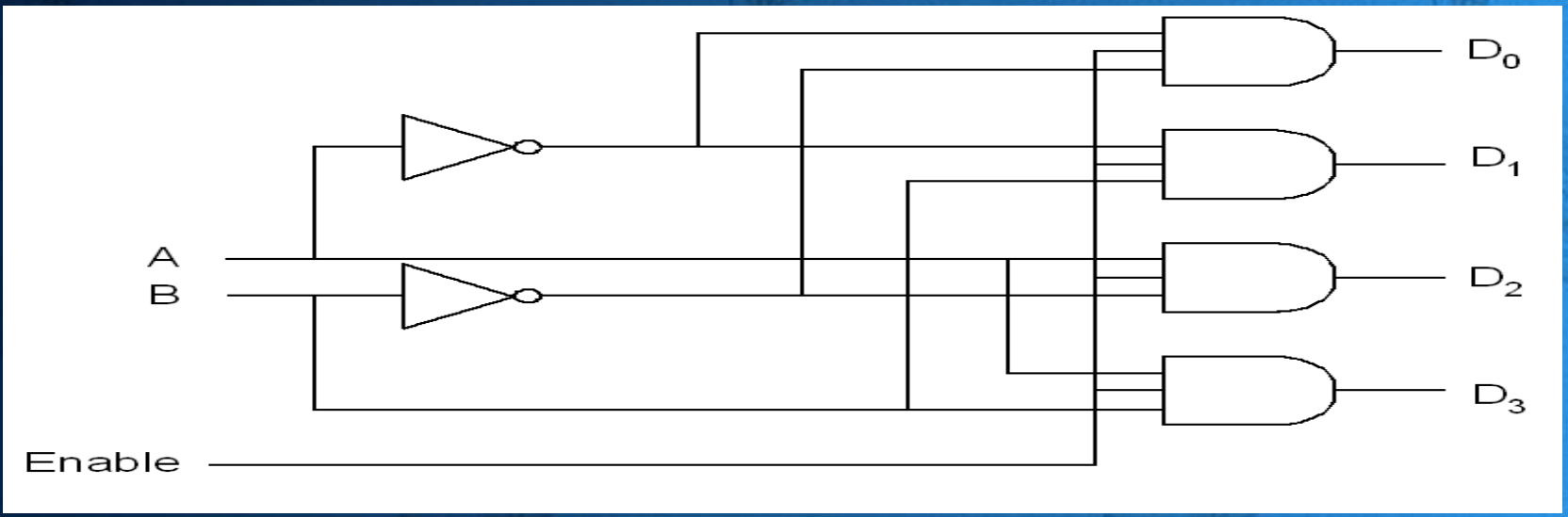
2x4 Decoder



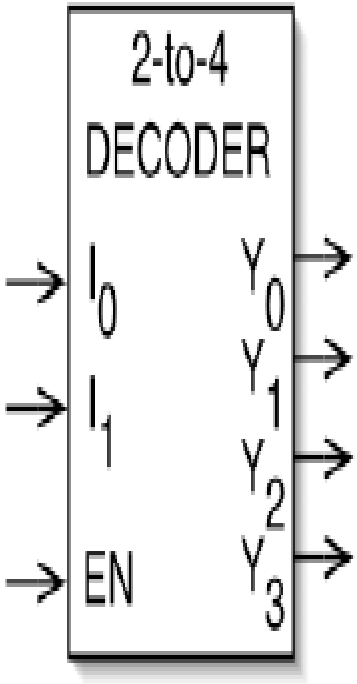
Enable = 1					
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Enable = 0					
A	B	D_0	D_1	D_2	D_3
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0

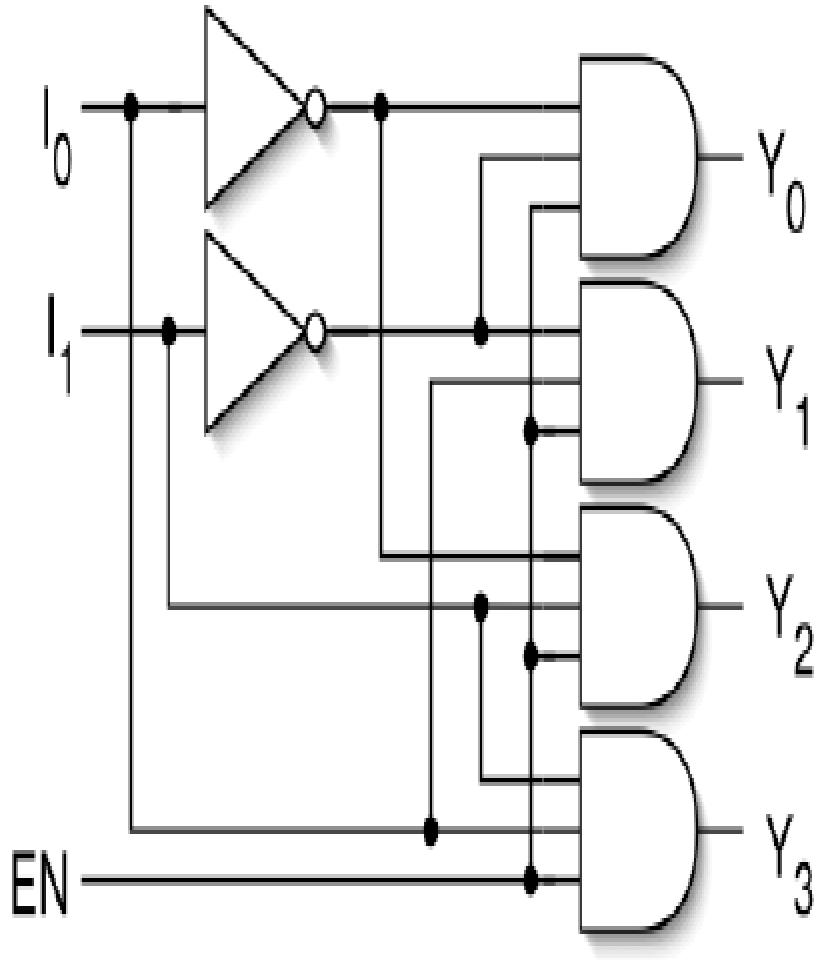
Internal Structure of 2x4 Decoder



Another View



EN	I_1	I_0	Y_3	Y_2	Y_1	Y_0
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



3 to 8 Line Decoder:

- Block diagram of 3 to 8 decoder is shown in fig. 4
- A , B and C are the inputs. (No. of inputs =3)
- No. of possible input combinations: $2^3=8$
- No. of Outputs : $2^3=8$, they are indicated by D_0 to D_7
- From the Truth Table it is clear that each output is "1" for only specific combination of inputs.

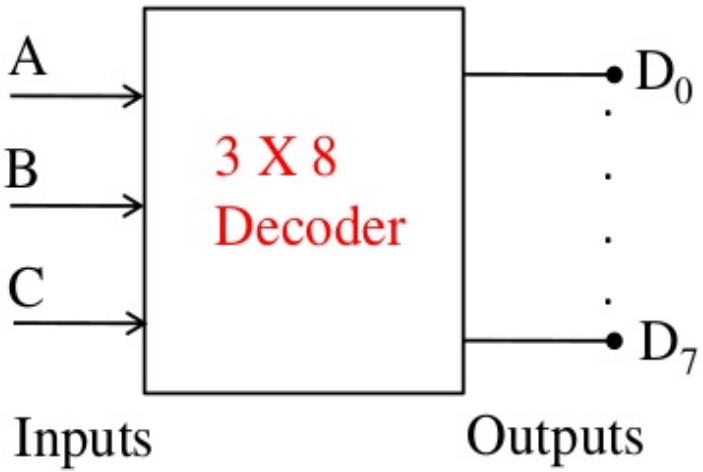


Fig. 4

TRUTH TABLE FOR 3 X 8 DECODER:

INPUTS			OUTPUTS								
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7	
0	0	0	1	0	0	0	0	0	0	0	$D_0 = \overline{A}\,\overline{B}\,\overline{C}$
0	0	1	0	1	0	0	0	0	0	0	$D_1 = \overline{A}\,\overline{B}\,C$
0	1	0	0	0	1	0	0	0	0	0	$D_2 = \overline{A}\,B\,\overline{C}$
0	1	1	0	0	0	1	0	0	0	0	$D_3 = \overline{A}\,B\,C$
1	0	0	0	0	0	0	1	0	0	0	$D_4 = A\,\overline{B}\,\overline{C}$
1	0	1	0	0	0	0	0	1	0	0	$D_5 = A\,\overline{B}\,C$
1	1	0	0	0	0	0	0	0	1	0	$D_6 = A\,B\,\overline{C}$
1	1	1	0	0	0	0	0	0	0	1	$D_7 = A\,B\,C$

LOGIC DIAGRAM OF 3 X 8 DECODER:

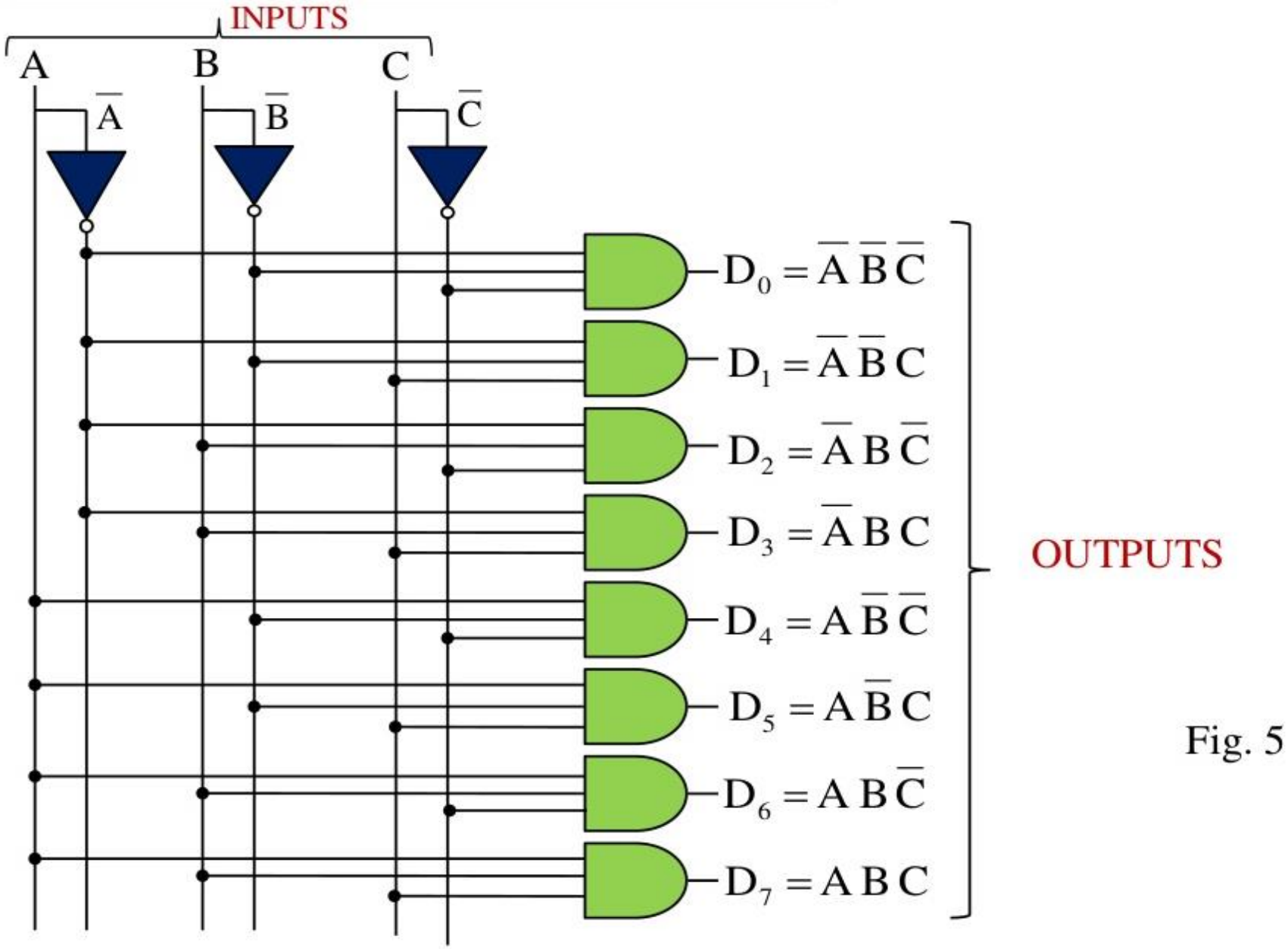


Fig. 5

EXPANSION OF DECODERS:

The number of lower order Decoder for implementing higher order Decoder can be find as

No. of lower order required = m_2/m_1

Where, m_1=No. of Outputs of lower order Decoder

m_2=No. of Outputs of higher order Decoder

3 x 8 Decoder From 2 x 4 Decoder:

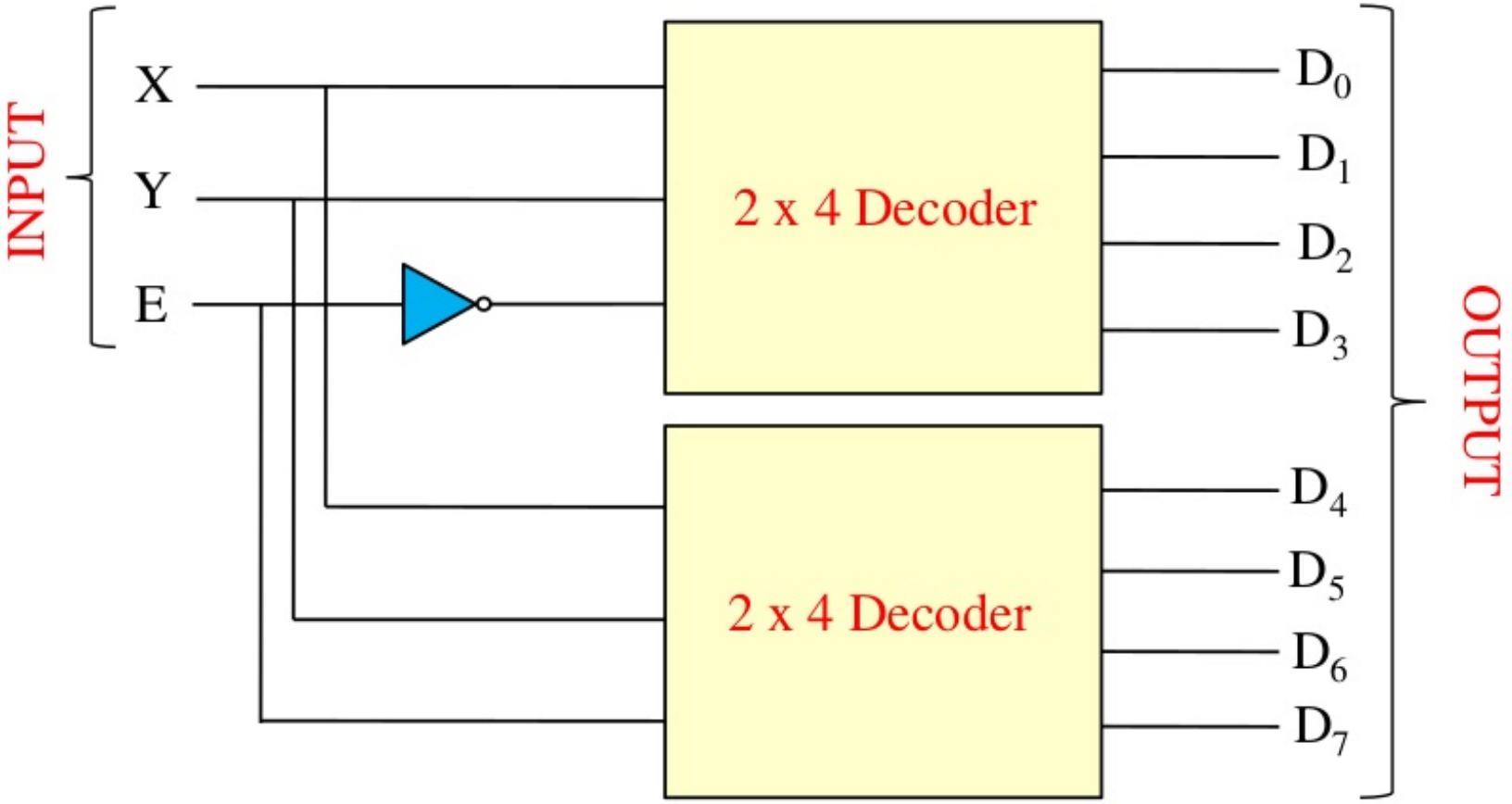


Fig. 6

4x16 Decoder

From course text book

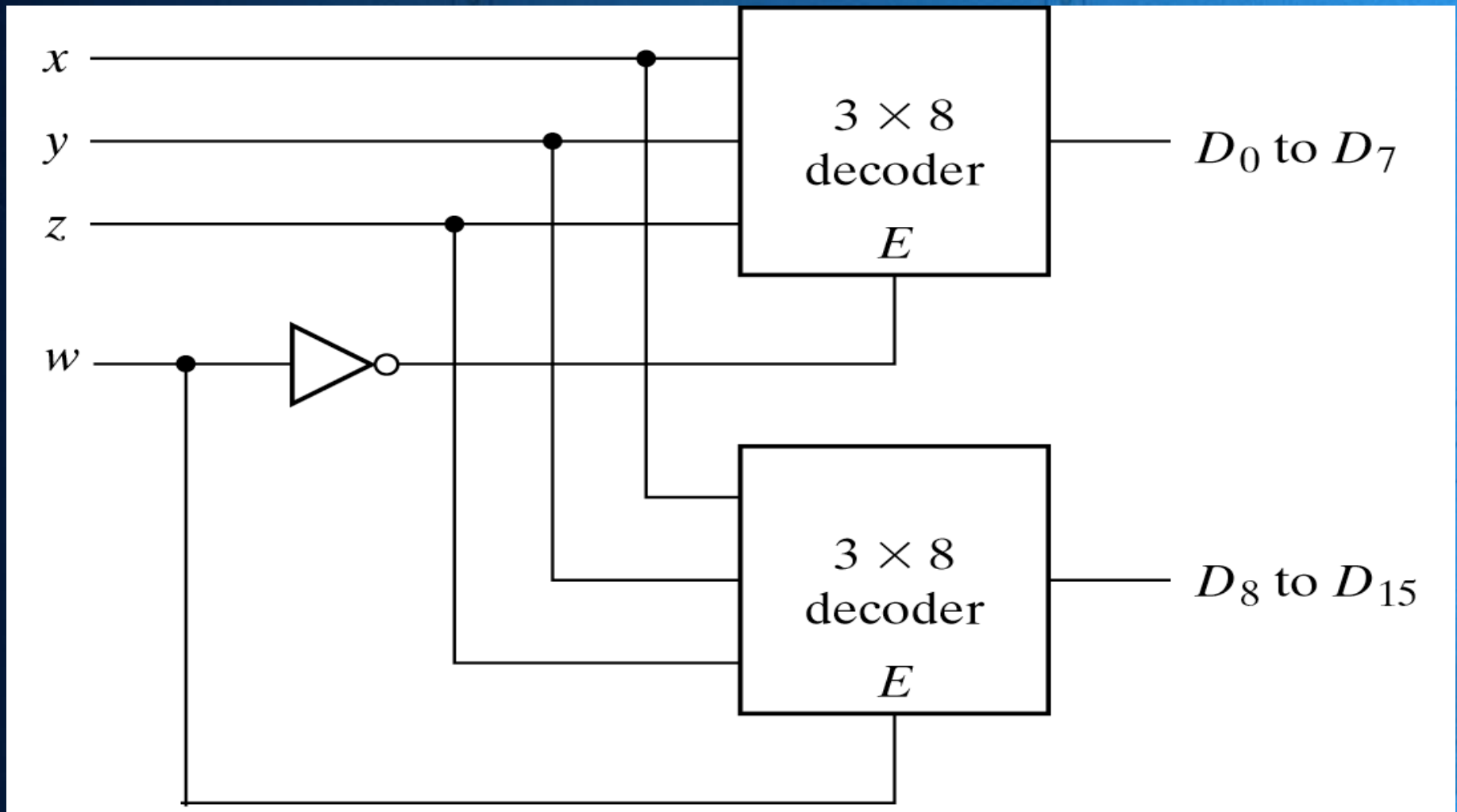


Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders

Example: Implement the following multiple output function using a suitable Decoder.

$f_1(A, B, C) = \sum m(0,4,7) + d(2,3)$

$f_2(A, B, C) = \sum m(1,5,6)$

$f_3(A, B, C) = \sum m(0,2,4,6)$

Solution: f1 consists of don't care conditions. So we consider them to be logic 1.

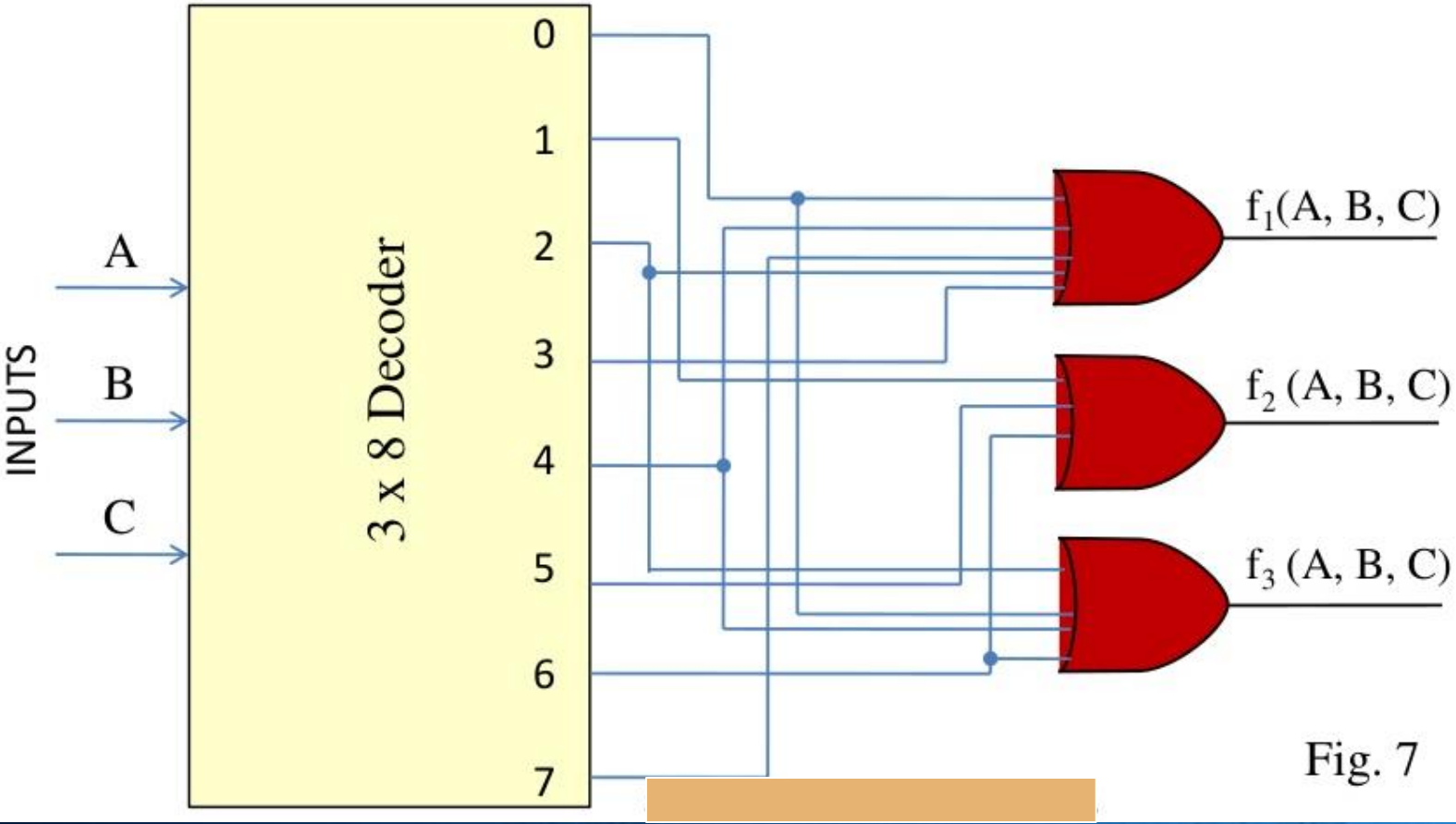


Fig. 7

EXAMPLE: Implement the following Boolean function using suitable Decoder.

$f_1(x,y,z)=\sum m(1,5,7)$

$f_2(x,y,z)=\sum m(0,3)$

$f_3(x,y,z)=\sum m(2,4,5)$

Solution:

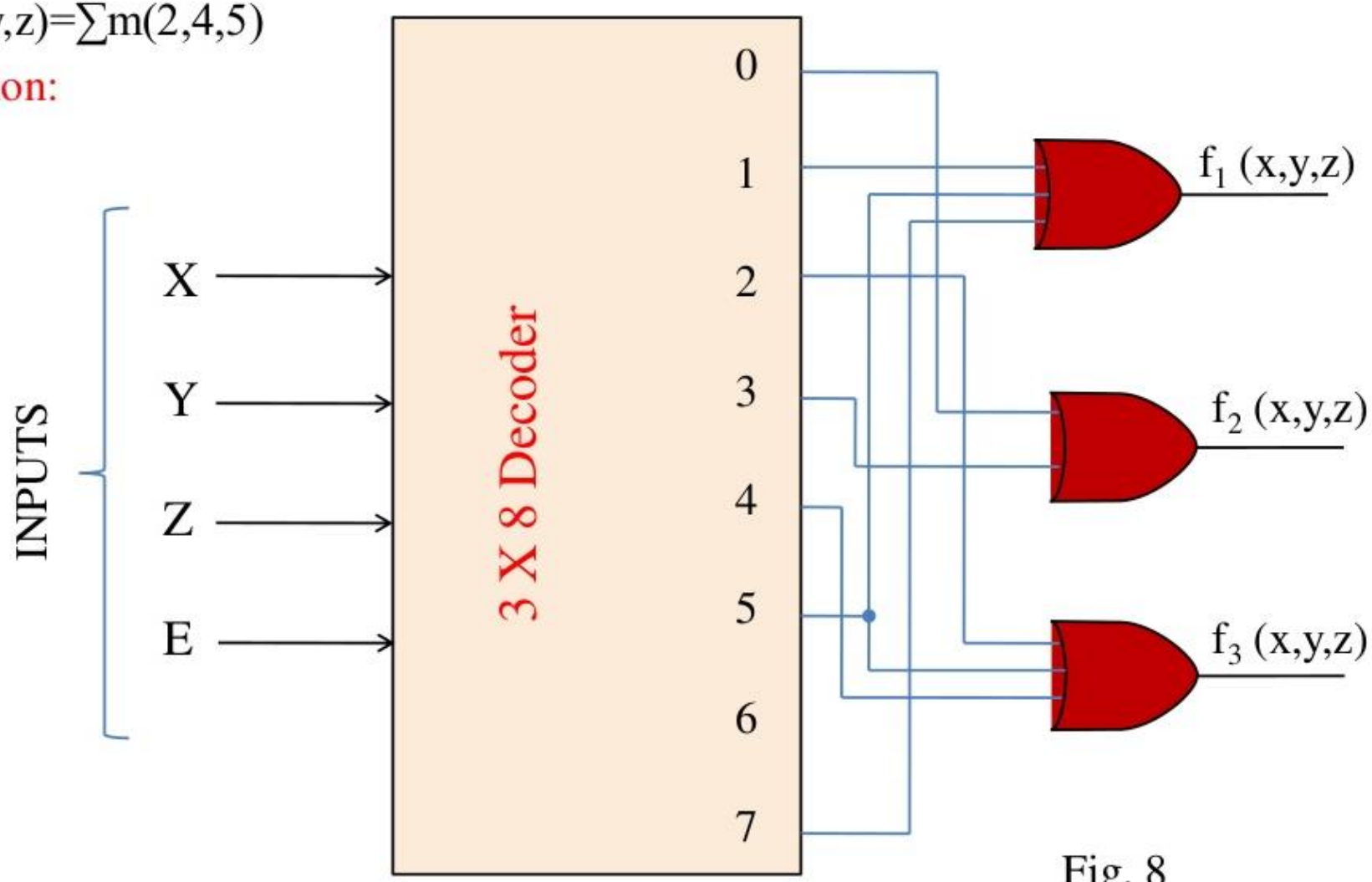


Fig. 8

EXAMPLE: A combinational circuit is defined by the following Boolean function. Design circuit with a Decoder and external gate.

$$F_1(x, y, z) = \bar{x}\,\bar{y}\,\bar{z} + x\,z$$

$$F_2(x, y, z) = x\,y\,\bar{z} + \bar{x}\,z$$

SOLUTION: STEP 1: Write the given function F_1 in SOP form

$$F_1(x, y, z) = \bar{x}\,\bar{y}\,\bar{z} + (y + \bar{y})\,x\,z$$

$$F_1(x, y, z) = \bar{x}\,\bar{y}\,\bar{z} + x\,y\,z + x\,\bar{y}\,z$$

$$F_1(x, y, z) = \Sigma m\,(0,5,7)$$

$$F_2(x, y, z) = x\,y\,\bar{z} + \bar{x}\,z$$

$$F_2(x, y, z) = x\,y\,\bar{z} + (y + \bar{y})\,\bar{x}\,z$$

$$F_2(x, y, z) = x\,y\,\bar{z} + \bar{x}\,y\,z + \bar{x}\,\bar{y}\,z$$

$$F_2(x, y, z) = \Sigma m\,(1,3,6)$$

Boolean Function using Decoder:

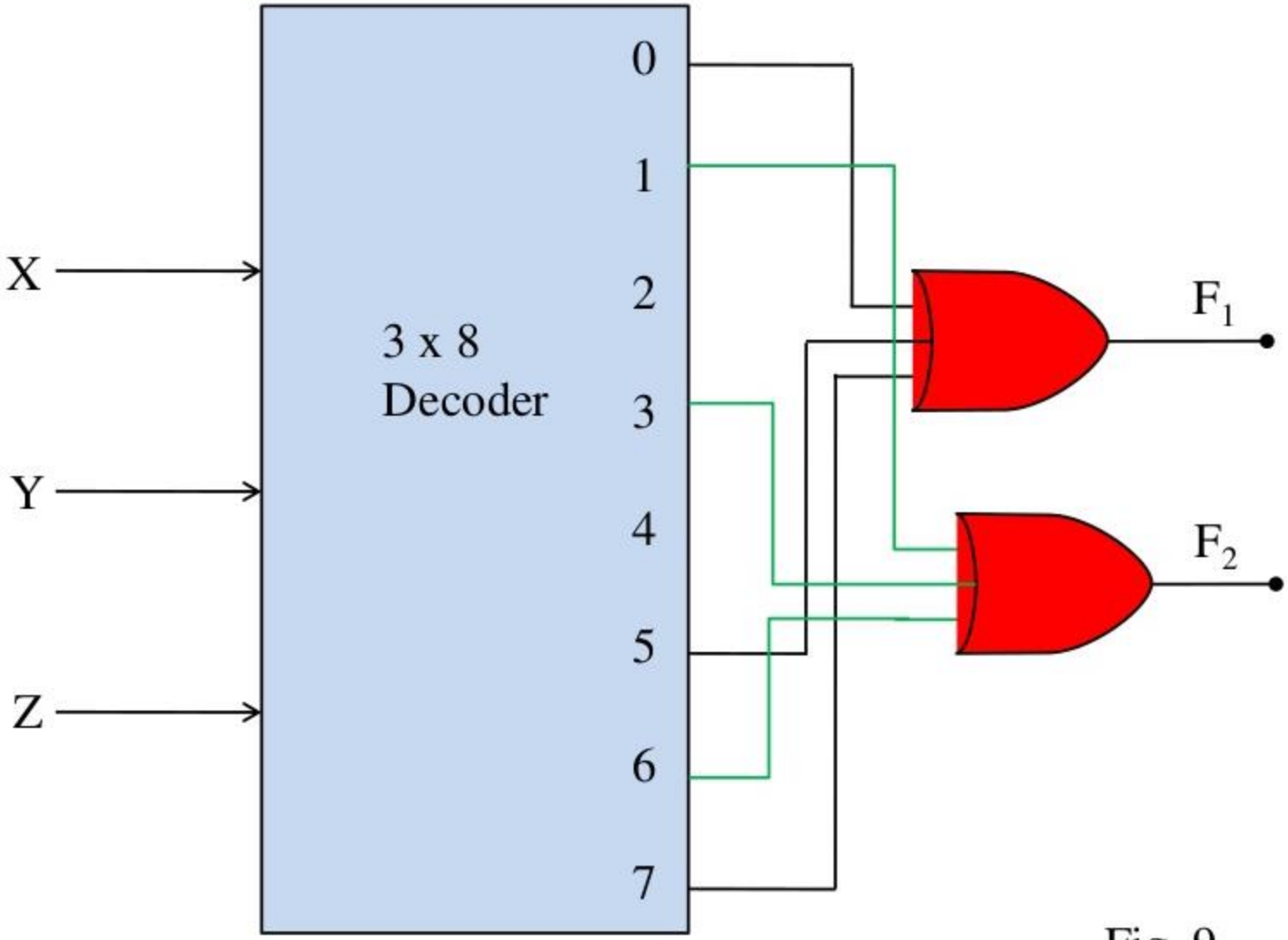


Fig. 9

3x8 Decoder

From course text book

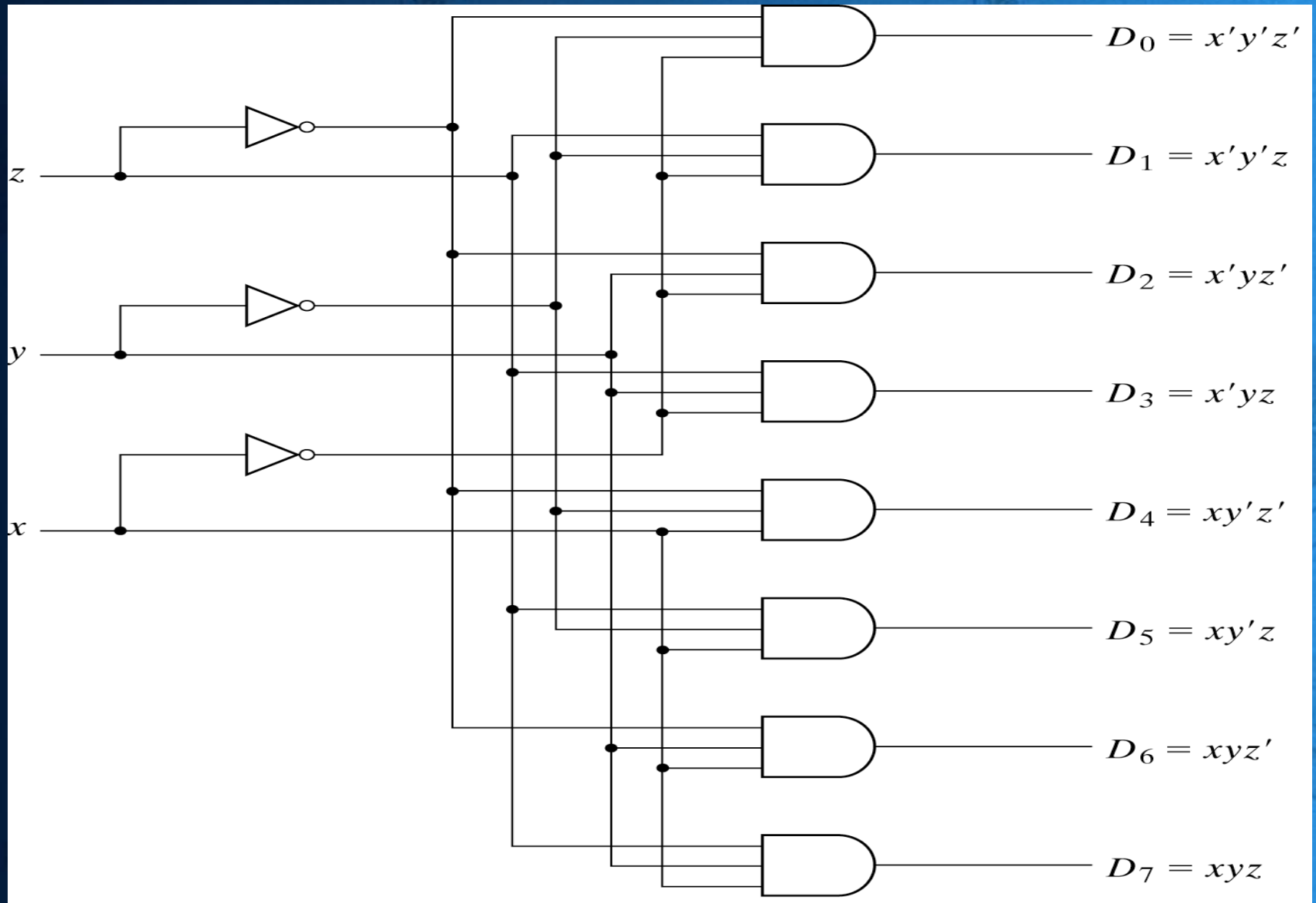
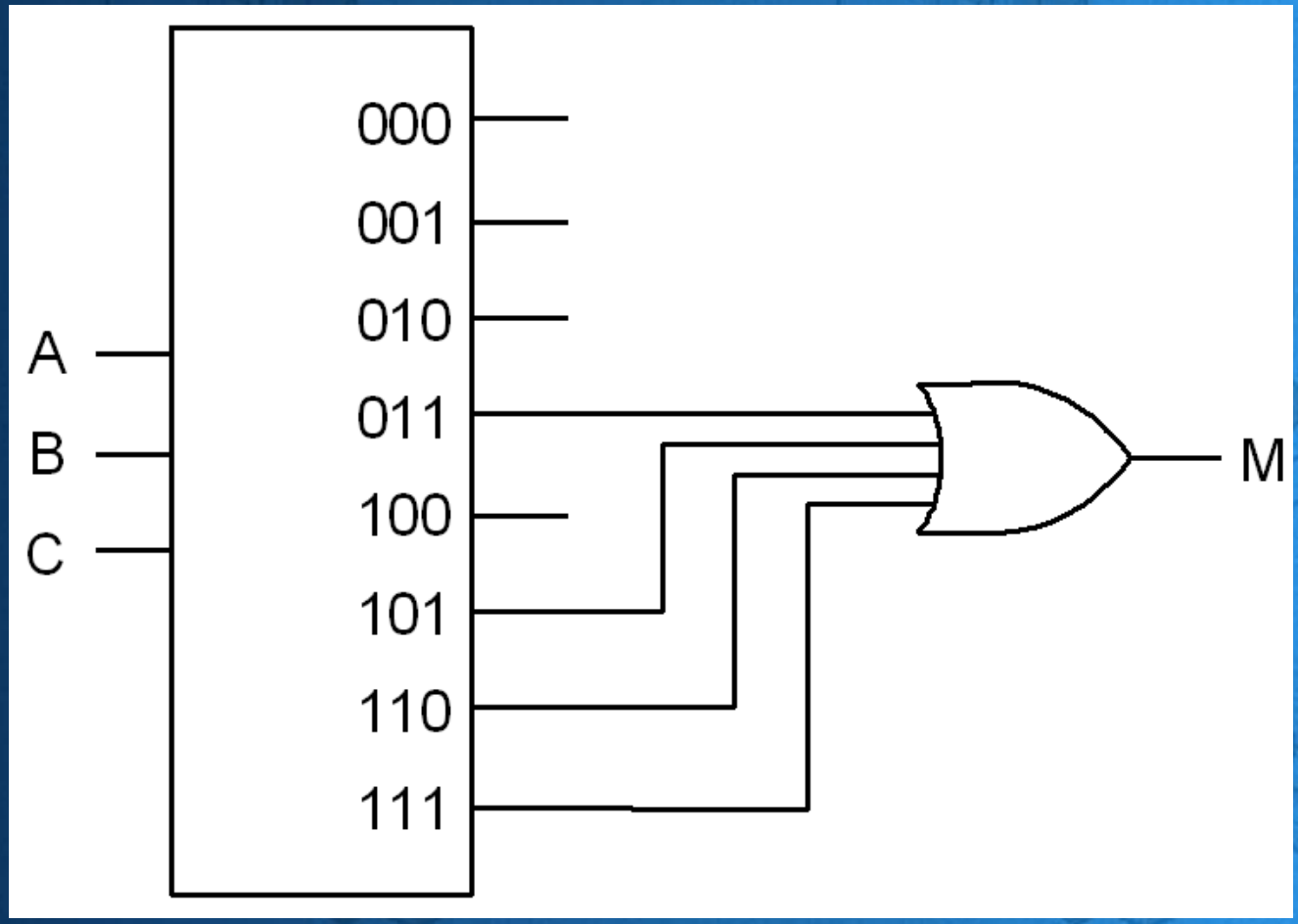


Fig. 4-18 3-to-8-Line Decoder

Example

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Full Adder with Decoder

A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

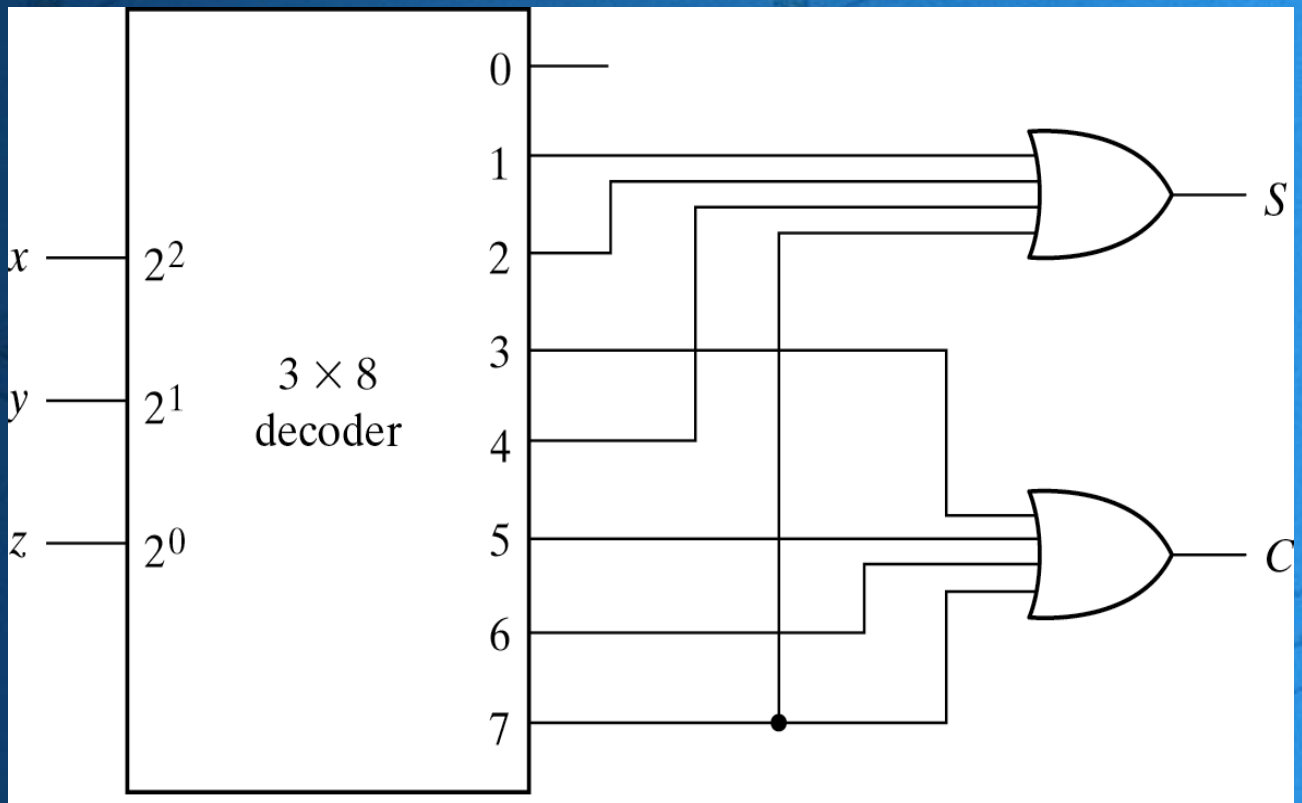


Fig. 4-21 Implementation of a Full Adder with a Decoder

$$S_i = A_i \oplus B_i \oplus C_i$$

$$C_{i+1} = A_i B_i' C_i + A_i' B_i C_i + A_i B_i$$

ENCODER

- An Encoder is a combinational logic circuit.
- It performs the inverse operation of Decoder.
- The opposite process of decoding is known as Encoding.
- An Encoder converts an active input signal into a coded output signal.
- Block diagram of Encoder is shown in Fig.10. It has 'M' inputs and 'N' outputs.
- An Encoder has 'M' input lines, only one of which is activated at a given time, and produces an N-bit output code, depending on which input is activated.

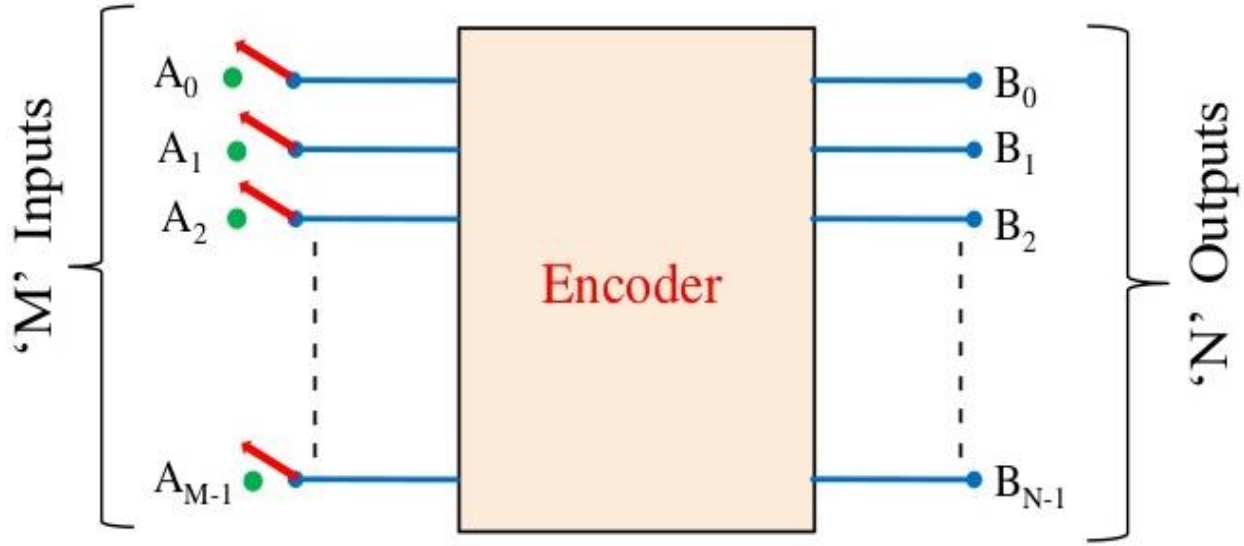


Fig. 10

- Encoders are used to translate the rotary or linear motion into a digital signal.
- The difference between Decoder and Encoder is that Decoder has Binary Code as an input while Encoder has Binary Code as an output.
- Encoder is an Electronics device that converts the analog signal to digital signal such as BCD Code.
- Types of Encoders

i. Priority Encoder

ii. Decimal to BCD Encoder

iii. Octal to Binary Encoder

iv. Hexadecimal to Binary Encoder

ENCODER

M=4

$M=2^2$

$M=2^N$

'M' is the input and

'N' is the output

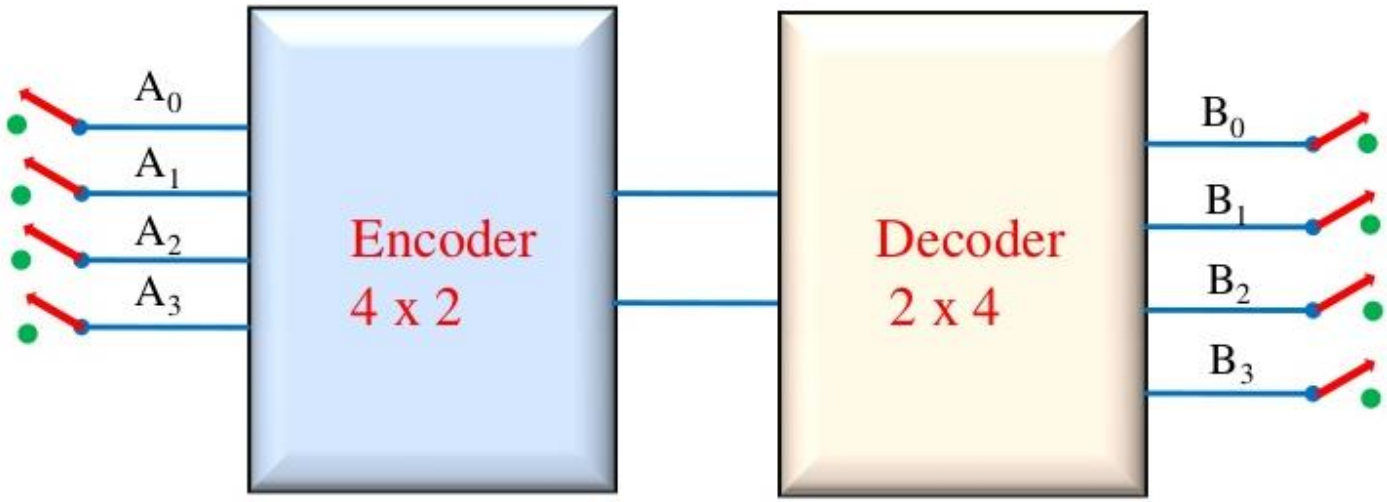


Fig. 11

ENCODER

M=4

$M=2^2$

$M=2^N$

'M' is the input and

'N' is the output

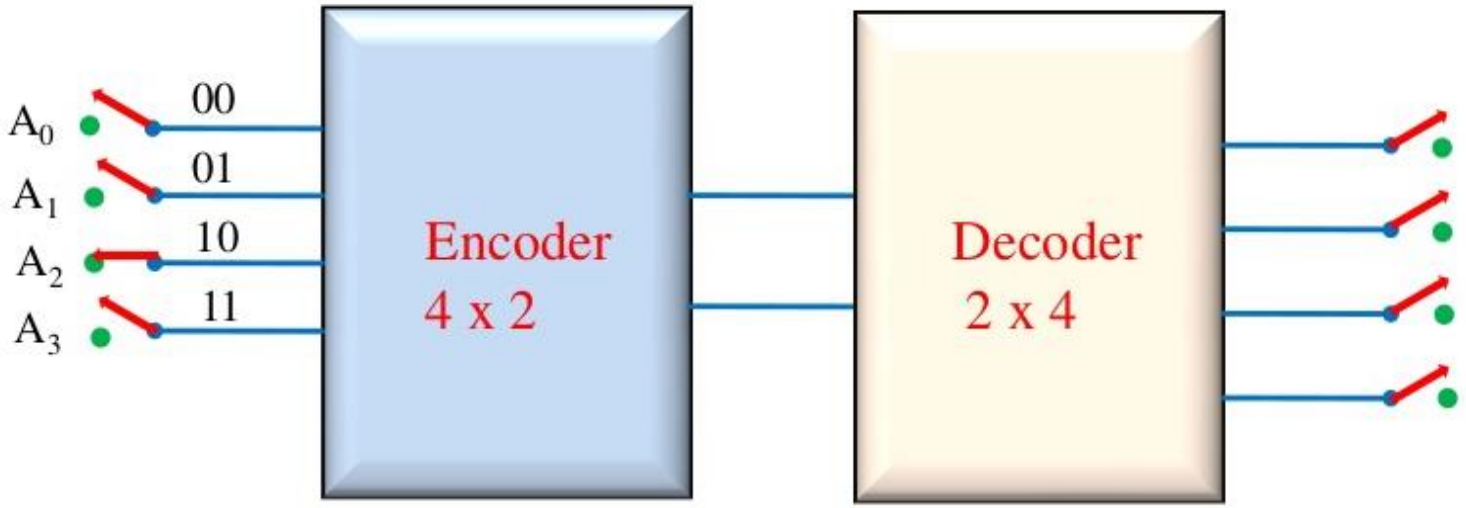


Fig. 12

ENCODER

M=4

$M=2^2$

$M=2^N$

'M' is the input and

'N' is the output

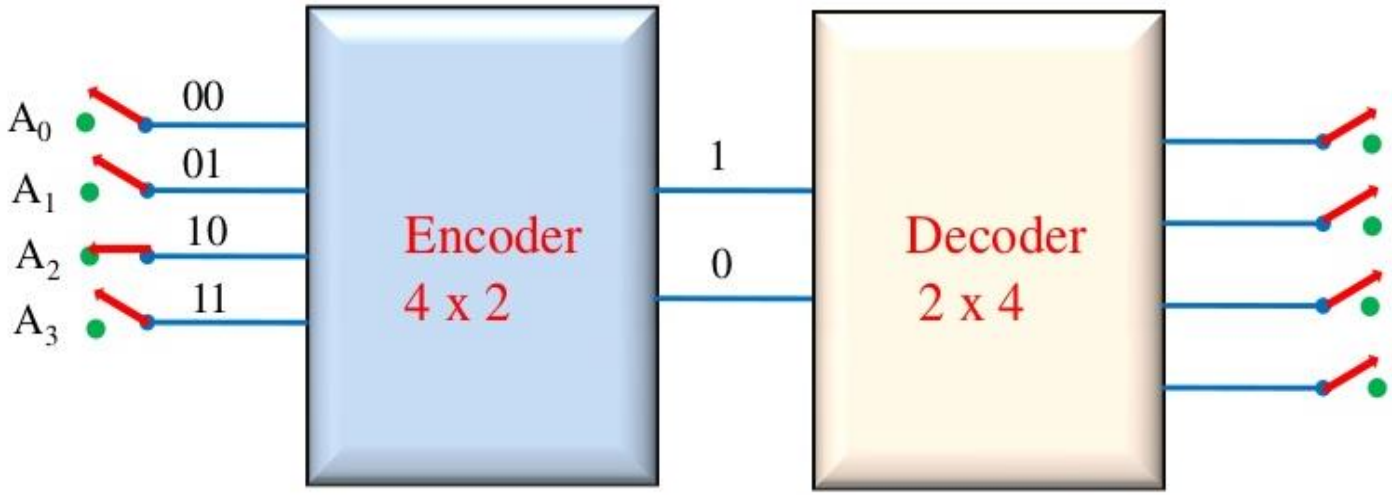


Fig. 13

ENCODER

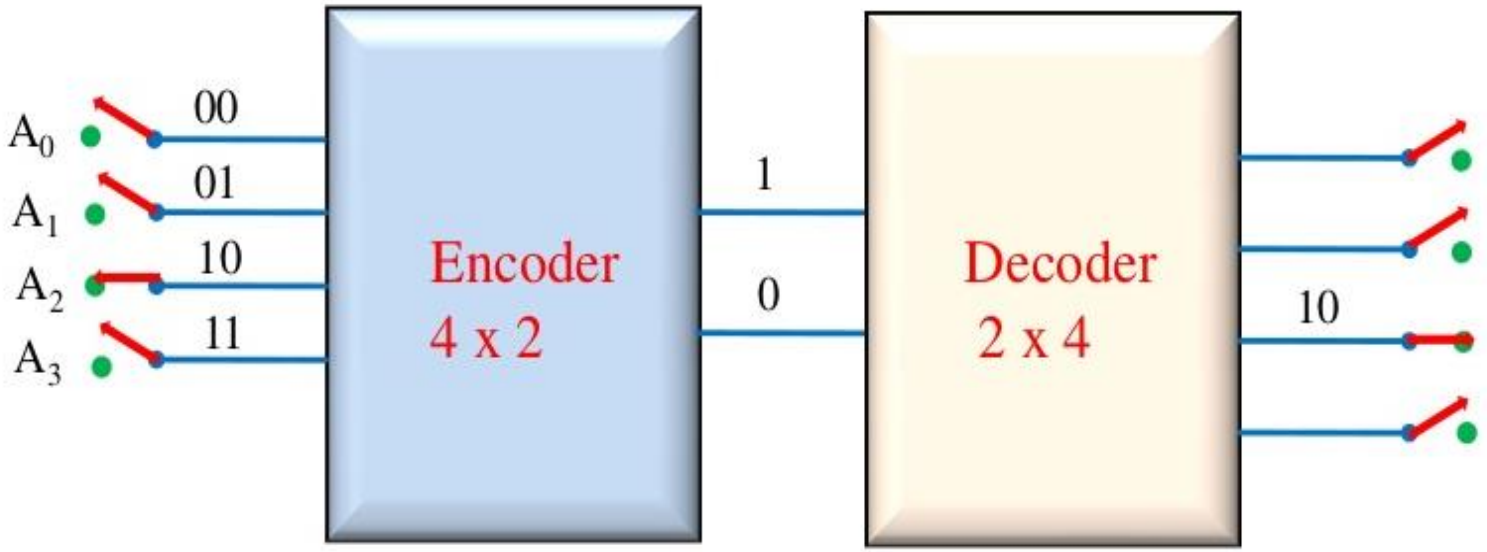
M=4

$M=2^2$

$M=2^N$

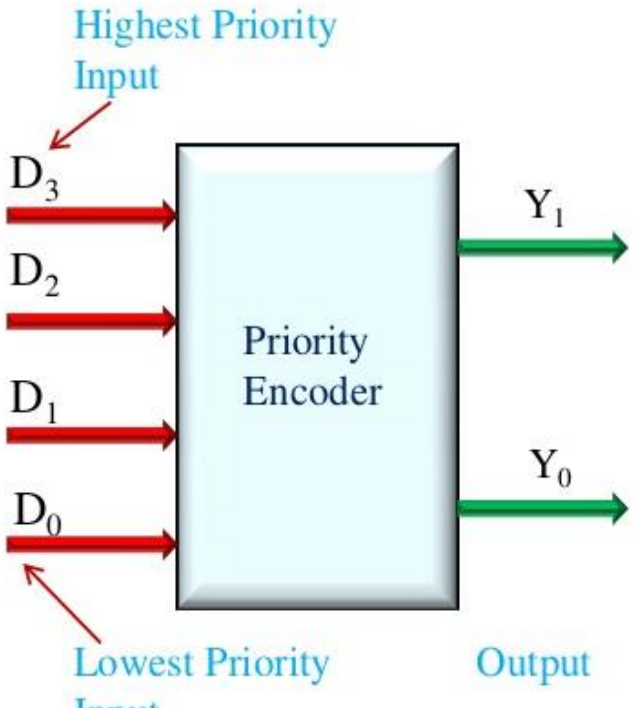
'M' is the input and

'N' is the output



PRIORITY ENCODER:

- As the name indicates, the priority is given to inputs line.
- If two or more input lines are high at the same time i.e 1 at the same time, then the input line with high priority shall be considered.
- Block diagram and Truth table of Priority Encoder are shown in fig.15



Block Diagram of Priority Encoder

Fig.15

TRUTH TABLE:

INPUTS				OUTPUTS		V
D_3	D_2	D_1	D_0	Y_1	Y_0	
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

- There are four inputs D_0, D_1,D_2, D_3 and two outputs Y_1 and Y_2.
- D_3 has highest priority and D_0 is at lowest priority.
- If D_3=1 irrespective of other inputs then output Y_1Y_0=11.
- D_3 is at highest priority so other inputs are considered as don't care.

K-map for Y_1 and Y_0

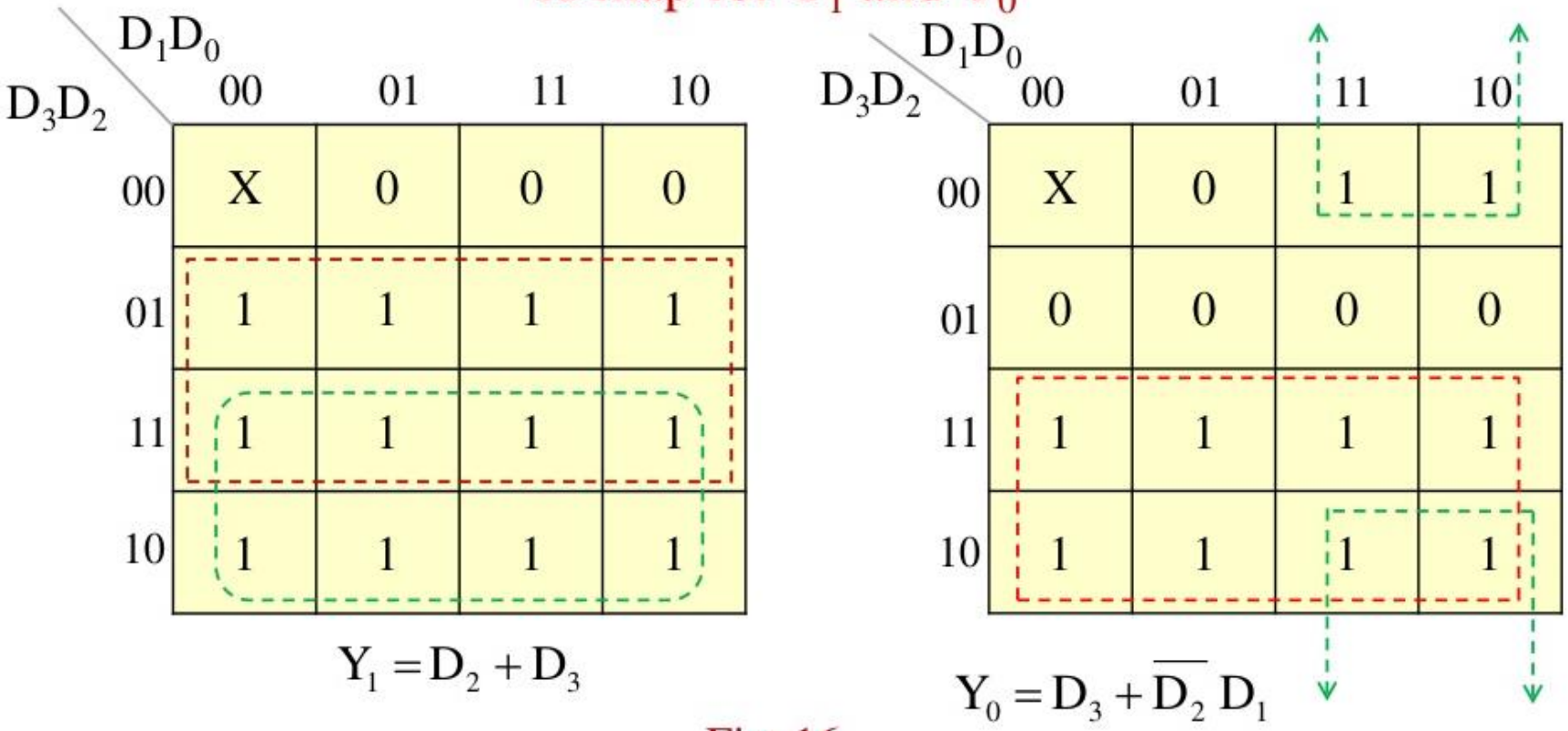


Fig. 16

LOGIC DIAGRAM OF PRIORITY ENCODER:

$$Y_1 = D_2 + D_3$$

$$Y_0 = D_3 + \overline{D_2}\, D_1$$

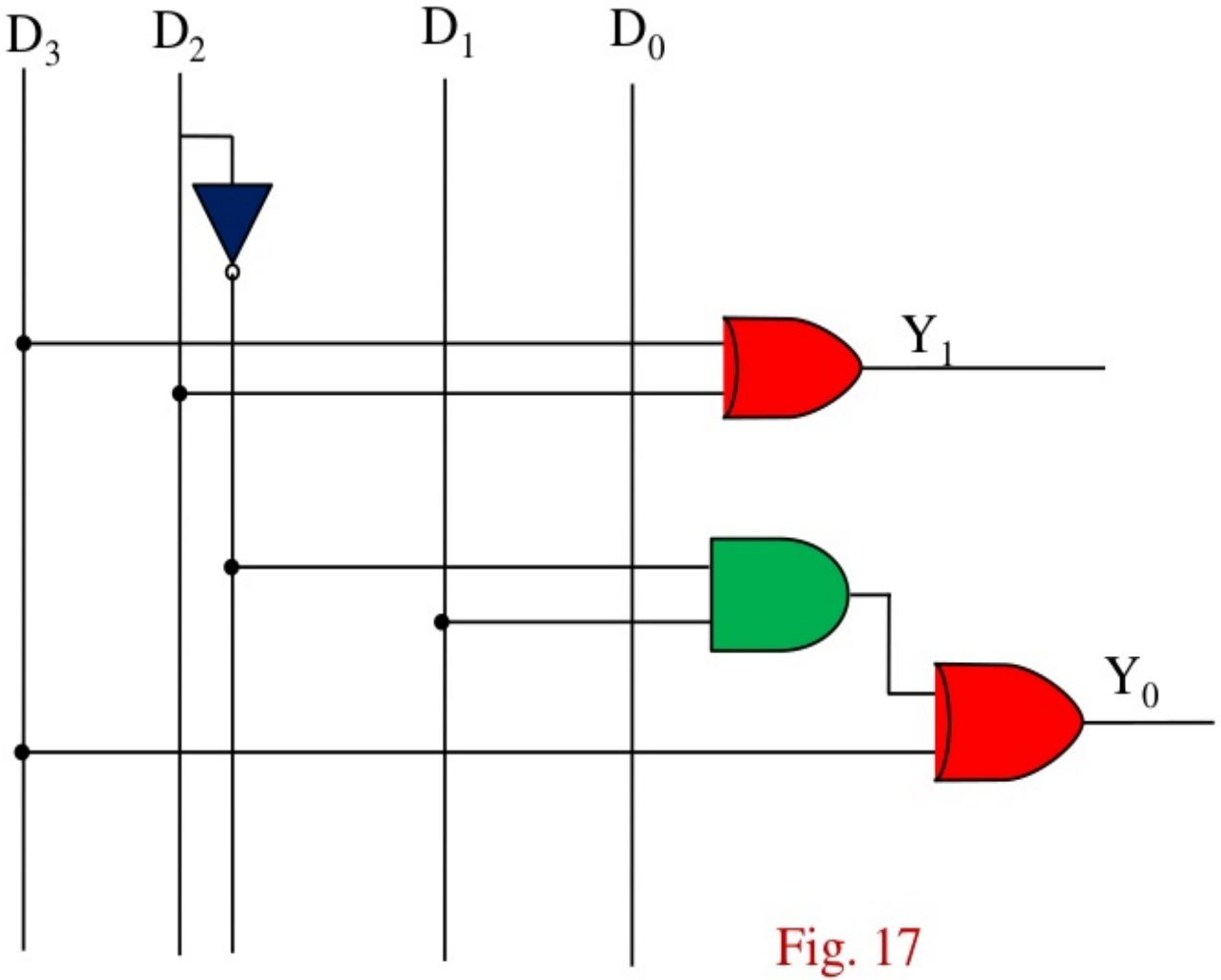


Fig. 17

DECIMAL TO BCD ENCODER:

- It has ten inputs corresponding to ten decimal digits (from 0 to 9) and four outputs (A,B,C,D) representing the BCD.
- The block diagram is shown in fig.18 and Truth table in fig.19

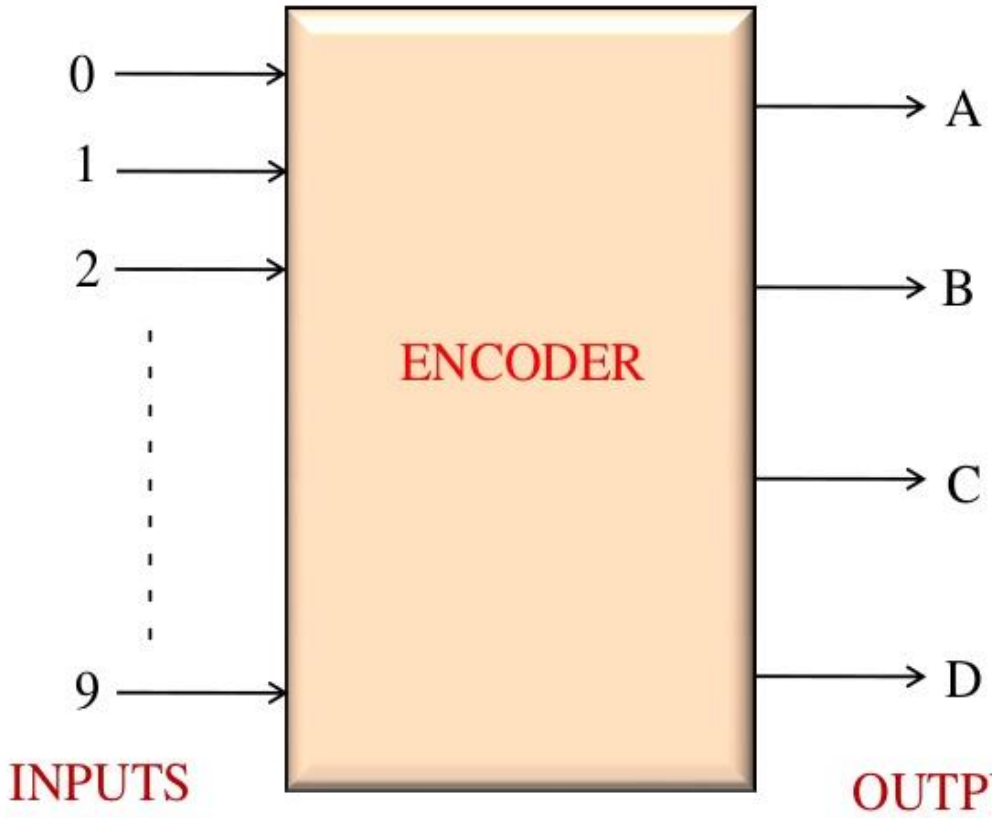


Fig. 18

Truth table:

INPUTS										BCD OUTPUTS			
0	1	2	3	4	5	6	7	8	9	A	B	C	D
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	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	0	0	1	0	0	0	0	1	1	0
0	0	0	0	0	0	0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	1	0	0	1

Fig. 19

- From Truth Table it is clear that the output A is HIGH when input is 8 OR 9 is HIGH

  Therefore A=8+9
- The output B is HIGH when 4 OR 5 OR 6 OR 7 is HIGH

  Therefore B=4+5+6+7
- The output C is HIGH when 2 OR 3 OR 6 OR 7 is HIGH

  Therefore C=2+3+6+7
- Similarly D=1+3+5+7+9

  Logic Diagram is shown in fig.20

DECIMAL TO BCD ENCODER

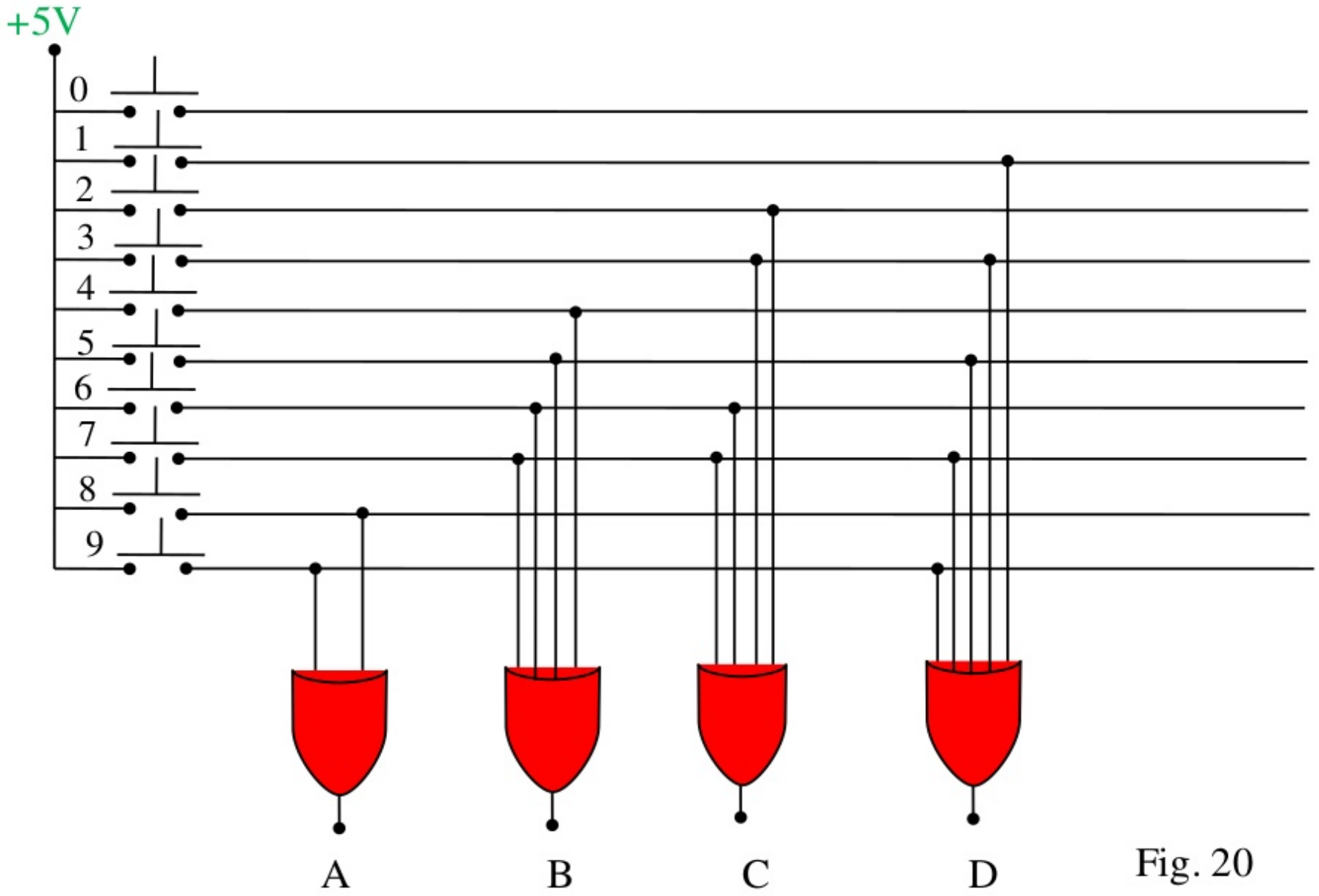


Fig. 20

OCTAL TO BINARY ENCODER:

- Block Diagram of Octal to Binary Encoder is shown in Fig. 21
- It has eight inputs and three outputs.
- Only one input has one value at any given time.
- Each input corresponds to each octal digit and output generates corresponding Binary Code.

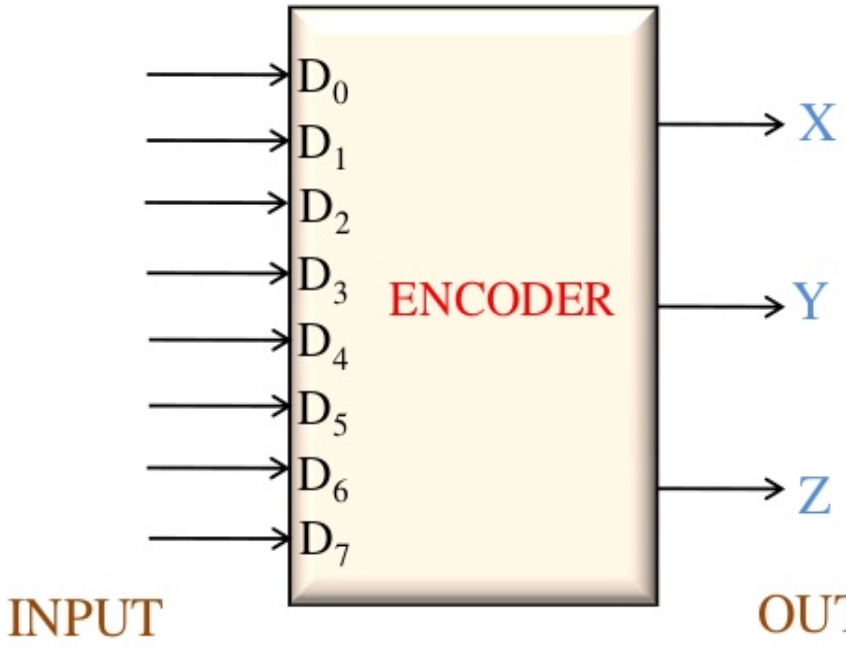


Fig. 21

TRUTH TABLE:

INPUT								OUTPUT		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Fig. 22

From Truth table:

$$X = D_4 + D_5 + D_6 + D_7$$

$$Y = D_2 + D_3 + D_6 + D_7$$

$$Z = D_1 + D_3 + D_5 + D_7$$

- It is assume that only one input is HIGH at any given time. If two outputs are HIGH then undefined output will produced. For example D_3 and D_6 are HIGH, then output of Encoder will be 111. This output neither equivalent code corresponding to D_3 nor to D_6.
- To overcome this problem, priorities should be assigned to each input.
- Form the truth table it is clear that the output X becomes 1 if any of the digit D_4 or D_5 or D_6 or D_7 is 1.
- D_0 is considered as don't care because it is not shown in expression.
- If inputs are zero then output will be zero. Similarly if D_0 is one, the output will be zero.
-

$$X = D_4 + D_5 + D_6 + D_7$$

$$Y = D_2 + D_3 + D_6 + D_7$$

$$Z = D_1 + D_3 + D_5 + D_7$$

LOGIC DIAGRAM:

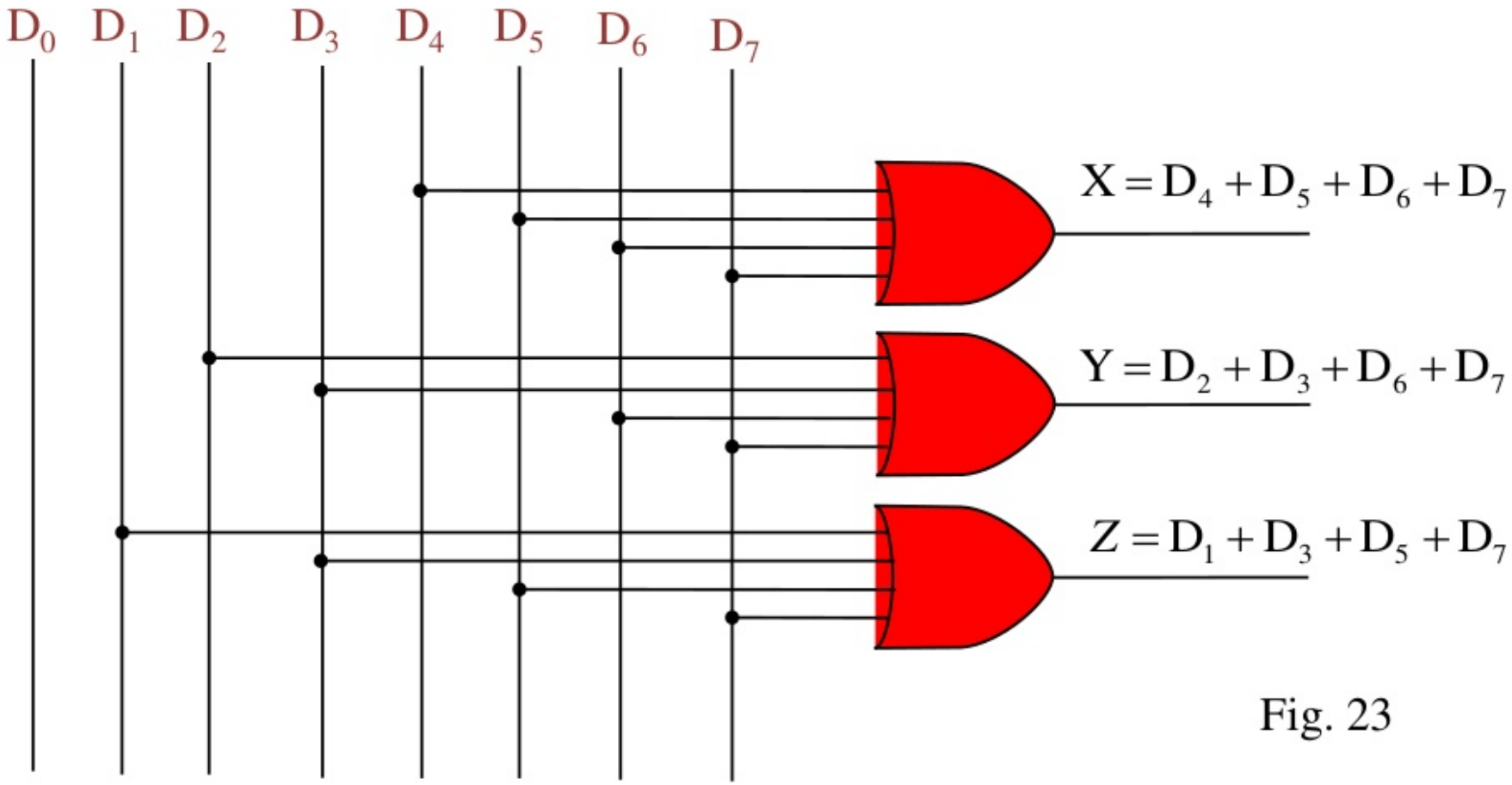
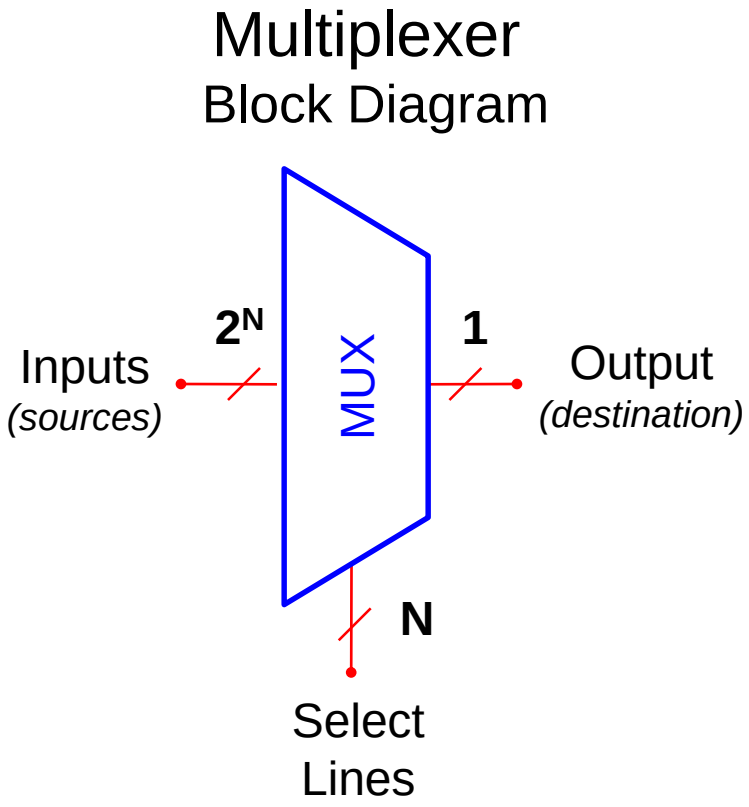


Fig. 23

What is a Multiplexer (MUX)?

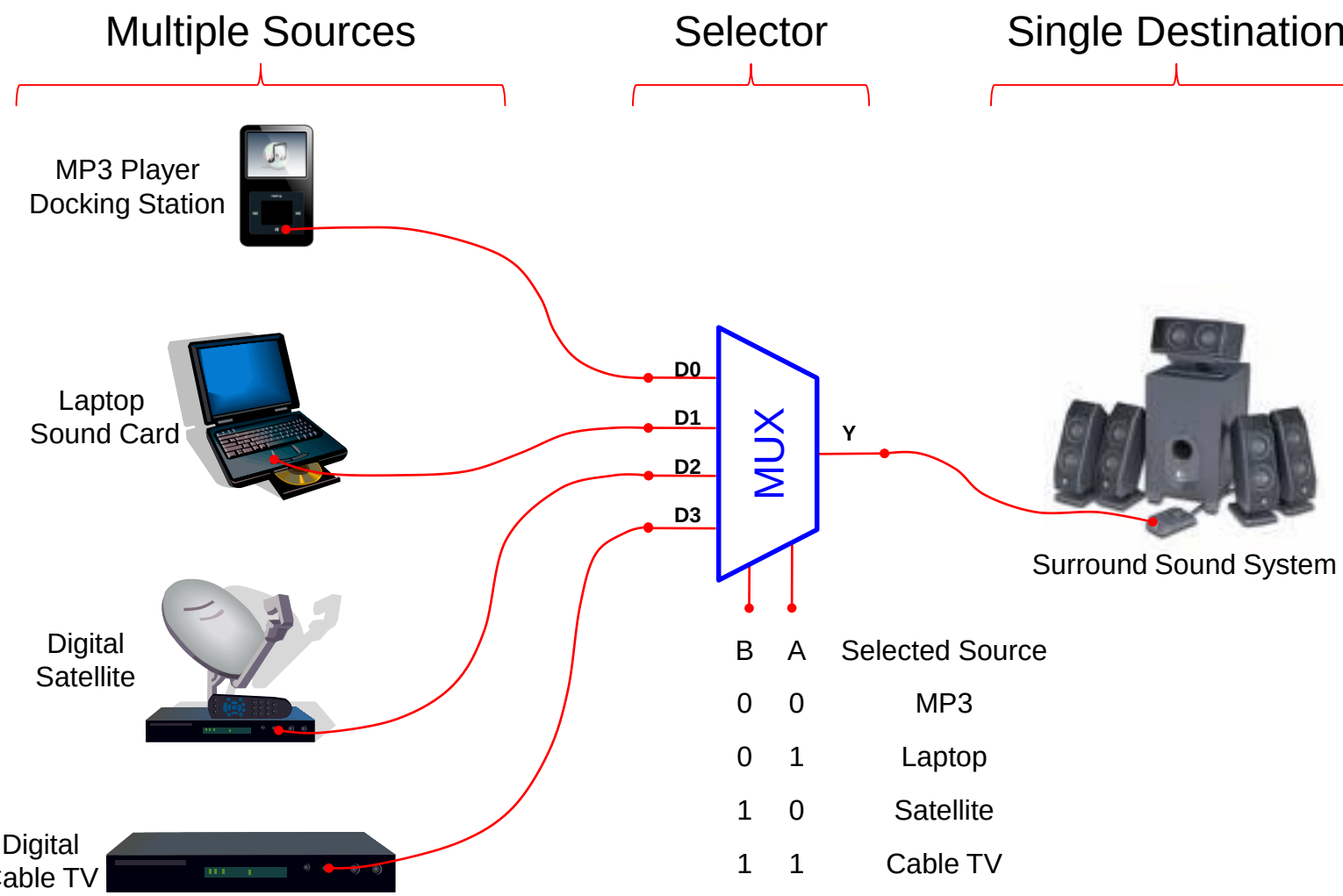
- A MUX is a digital switch that has multiple inputs (sources) and a single output (destination).
- The select lines determine which input is connected to the output.
- MUX Types
  - → 2-to-1 (1 select line)
  - → 4-to-1 (2 select lines)
  - → 8-to-1 (3 select lines)
  - → 16-to-1 (4 select lines)



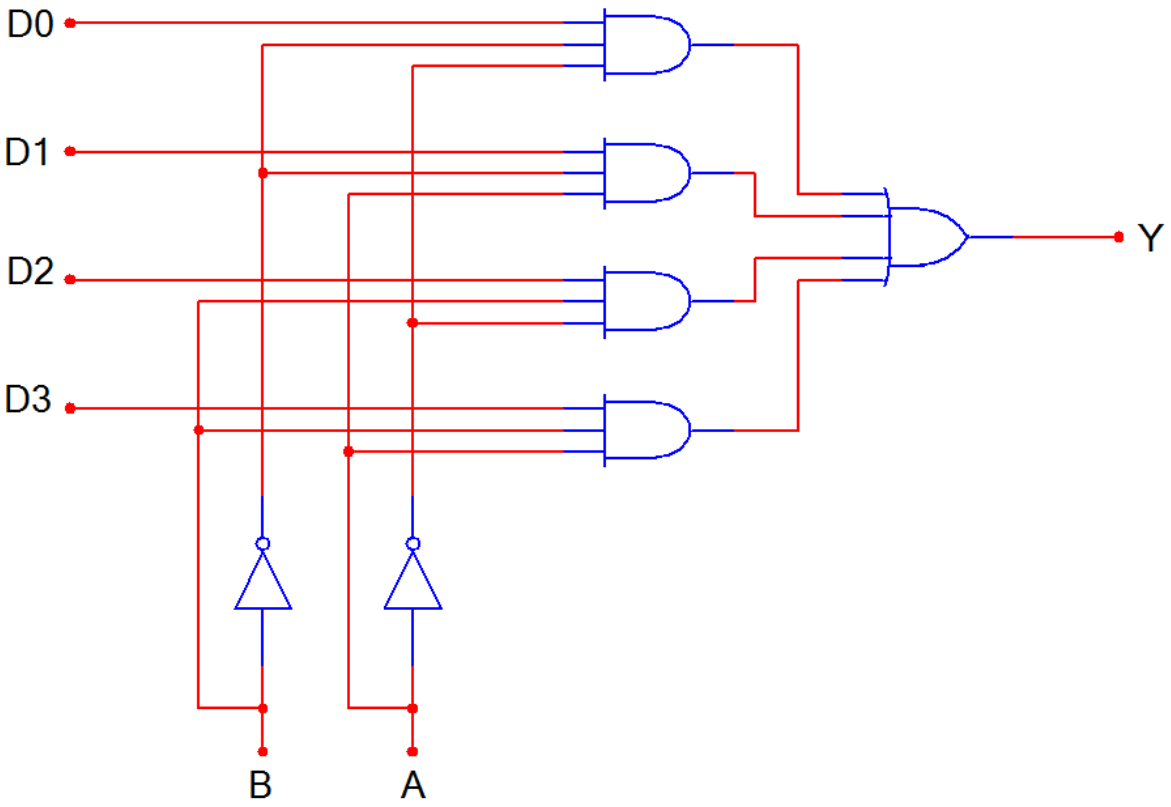
Multiplexer(contd..)

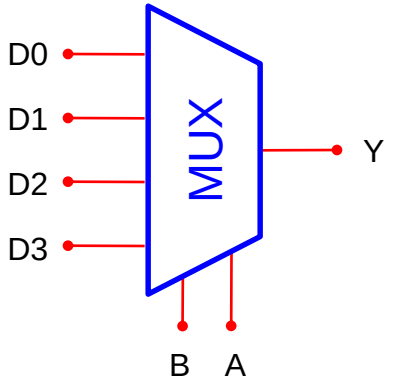
- Also called data selectors.
- Basic function: select one of its 2^n data input lines and place the corresponding information onto a single output line.
- n input bits needed to specify which input line is to be selected.
  - Place binary code for a desired data input line onto its n select input lines.

Typical Application of a MUX



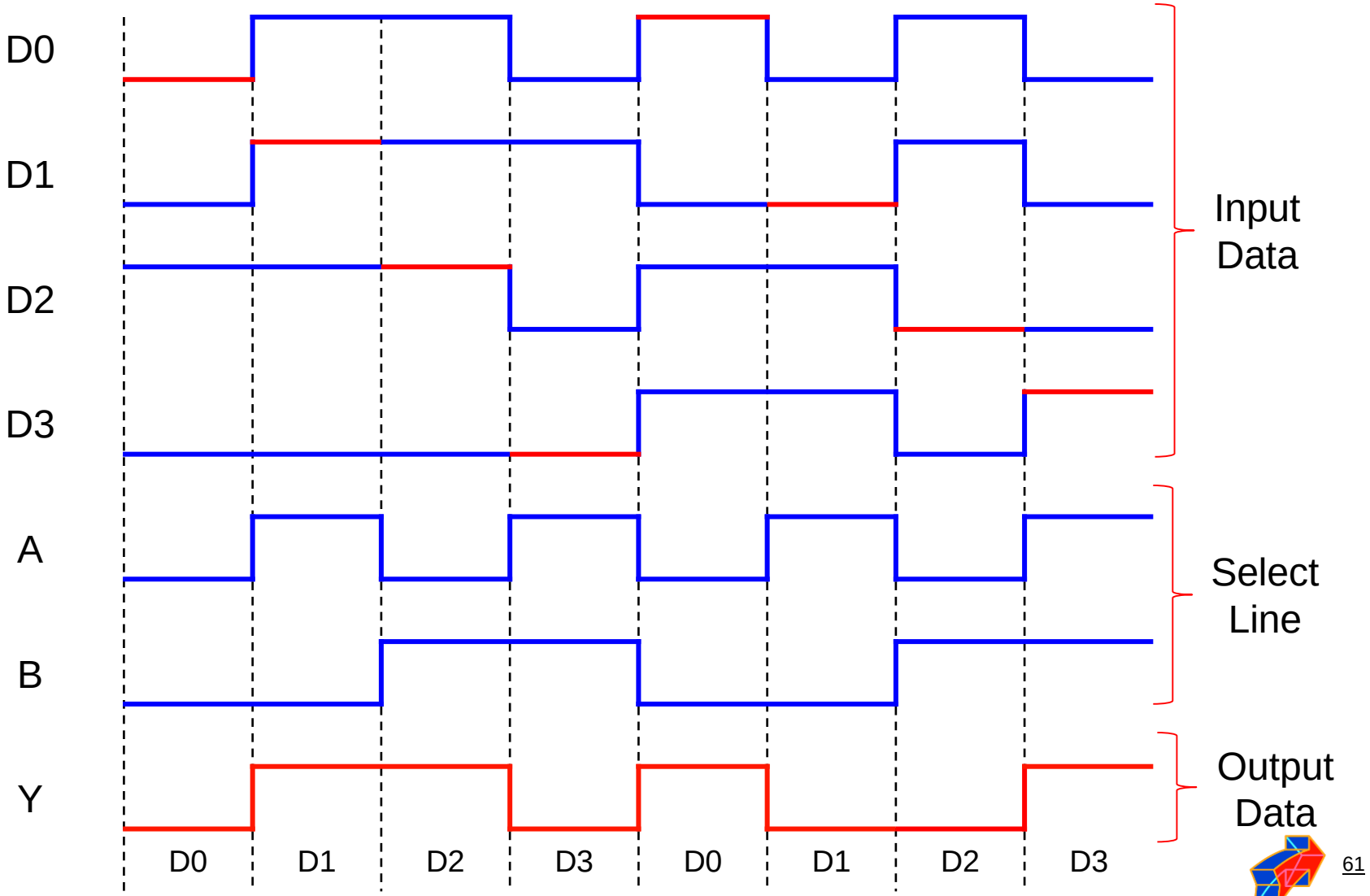
4-to-1 Multiplexer (MUX)





B	A	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

4-to-1 Multiplexer Waveforms



Medium Scale Integration MUX

4-to-1 MUX

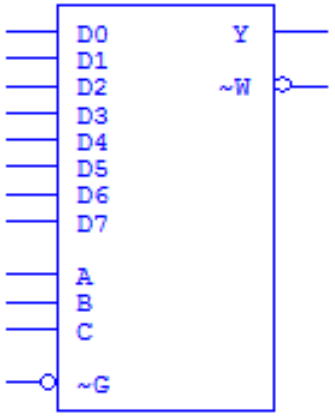
Inputs

Select

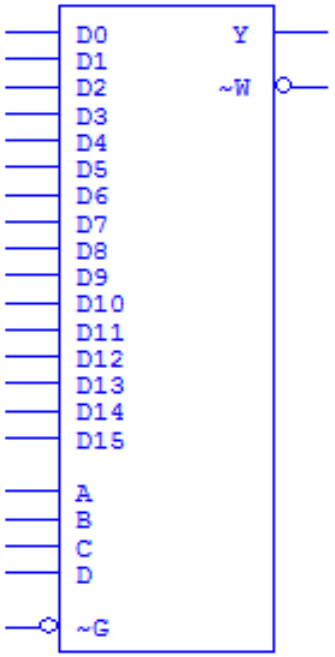
Enable

Output (Y)
(and inverted output)

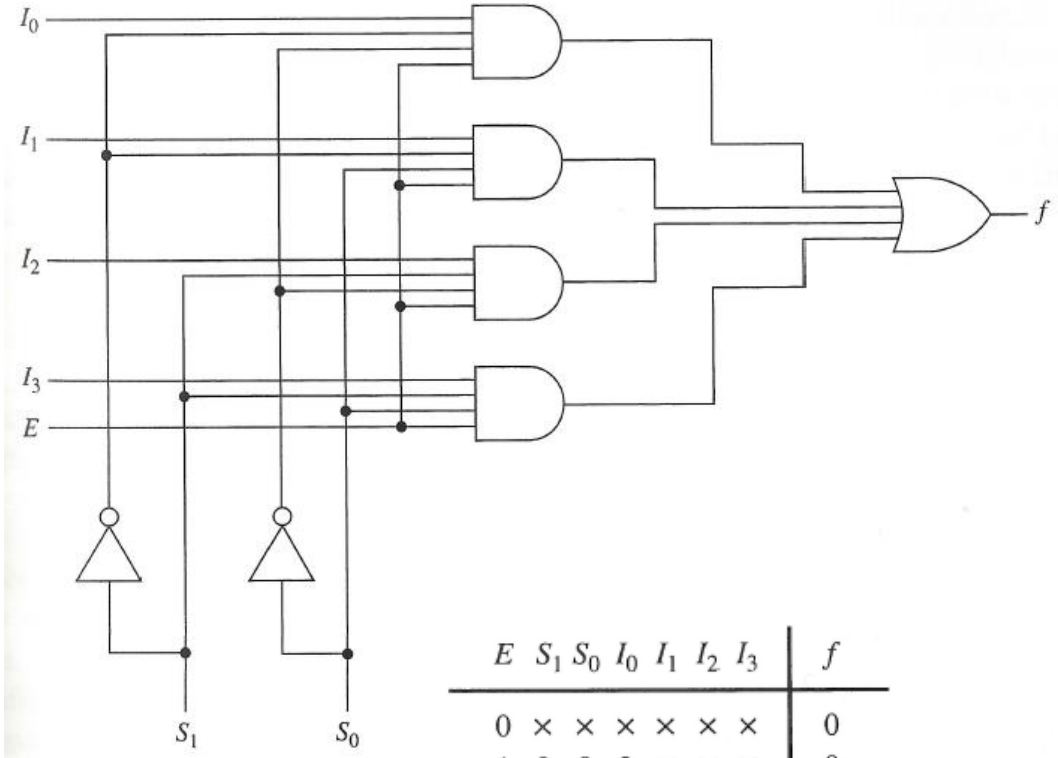
8-to-1 MUX



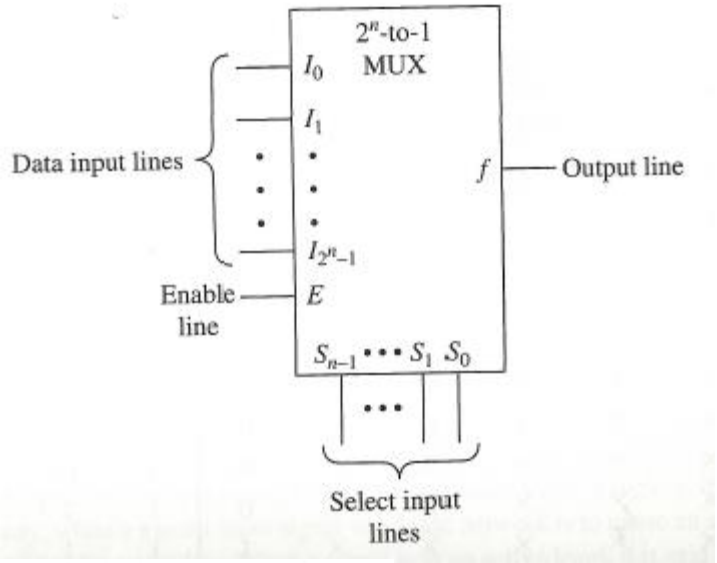
16-to-1 MUX



Realization of 4-to-1 line multiplexer

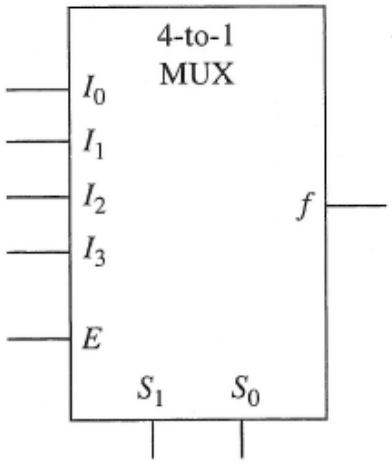


Logic Diagram



E	S_1	S_0	I_0	I_1	I_2	I_3	f
0	×	×	×	×	×	×	0
1	0	0	0	×	×	×	0
1	0	0	1	×	×	×	1
1	0	1	×	0	×	×	0
1	0	1	×	1	×	×	1
1	1	0	×	×	0	×	0
1	1	0	×	×	1	×	1
1	1	1	×	×	×	0	0
1	1	1	×	×	×	1	1

Truth Table



Symbol

Realization of 4-to-1 line multiplexer

- Alternate description:

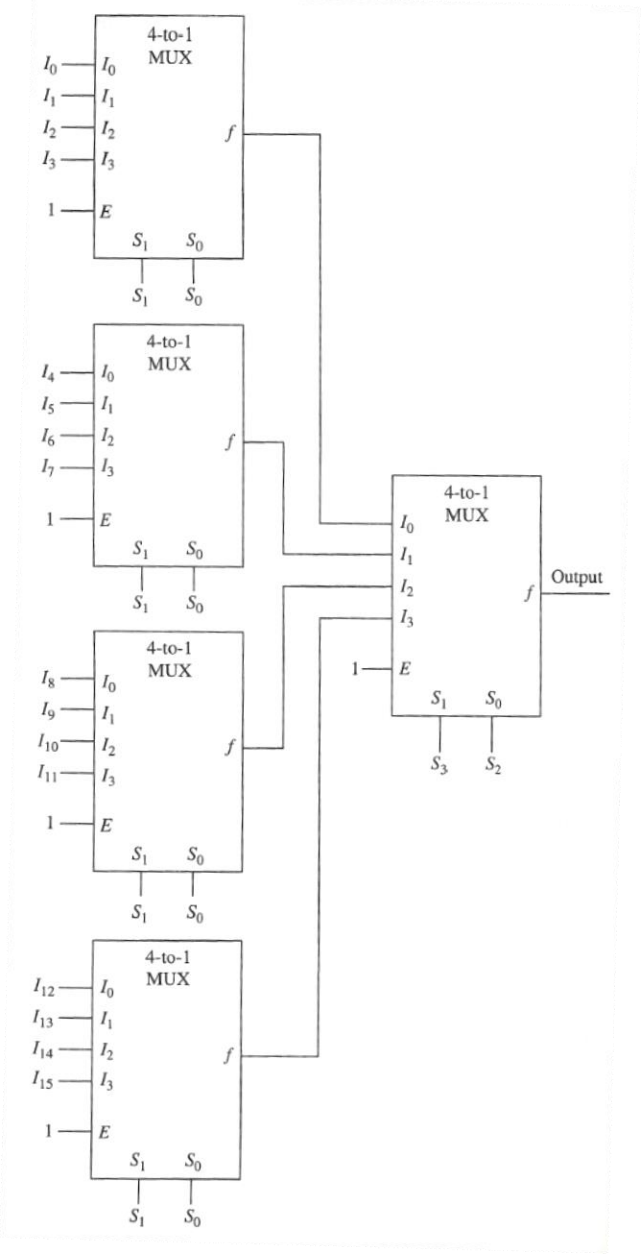
Table 5.6 Function table for a 4-to-1-line multiplexer

E	S_1	S_0	f
0	×	×	0
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3

- Algebraic description of multiplexer:

$$f = \left(I_0\overline{S}_1\overline{S}_0 + I_1\overline{S}_1S_0 + I_2S_1\overline{S}_0 + I_3S_1S_2\right)E$$

Building a Large Multiplexer



Multiplexers

- One of the primary applications of multiplexers is to provide for the transmission of information from several sources over a single path.
- This process is known as multiplexing.
- Demultiplexer = decoder with an enable input.

Multiplexer/Demultiplexer for information transmission

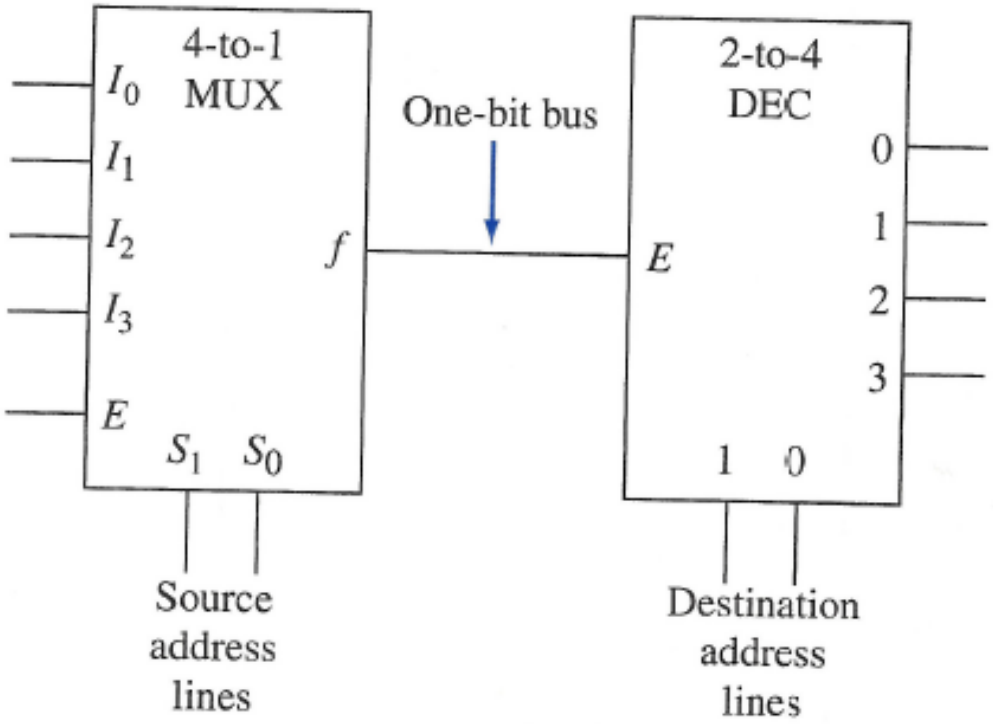


Figure 5.35 A multiplexer/demultiplexer arrangement for information transmission.

Logic Design with Multiplexers

x	y	z	f
0	0	0	f_0
0	0	1	f_1
0	1	0	f_2
0	1	1	f_3
1	0	0	f_4
1	0	1	f_5
1	1	0	f_6
1	1	1	f_7

The Boolean expression corresponding to this truth table can be written as:

$$f(x, y, z)$$
$$= f_0 \cdot \overline{x}\,\overline{y}\,\overline{z} + f_1 \cdot \overline{x}\,\overline{y}\,z + f_2 \cdot \overline{x}y\overline{z} + f_3 \cdot \overline{x}yz + f_4 \cdot x\overline{y}\,\overline{z} + f_5 \cdot x\overline{y}z$$
$$+ f_6 xy\overline{z} + f_7 \cdot xyz.$$

Logic Design with Multiplexers

- The Boolean expression corresponding to this truth table can be written as:

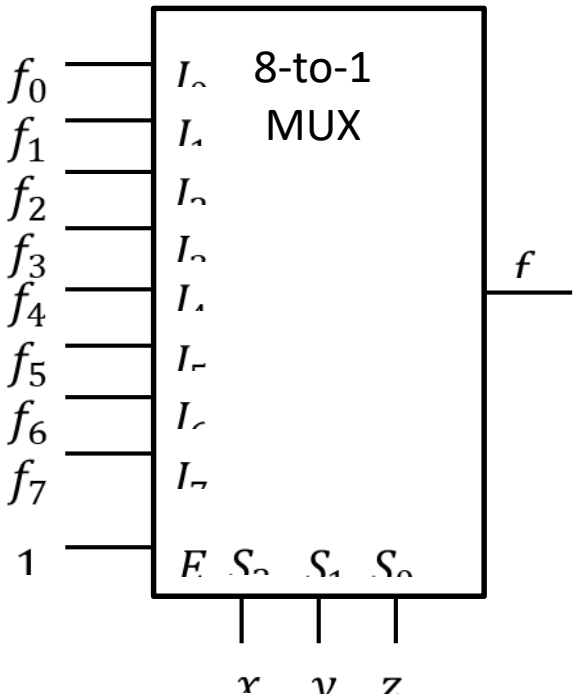
$$f(x, y, z) = f_0 \cdot \overline{x}\,\overline{y}\,\overline{z} + f_1 \cdot \overline{x}\,\overline{y}\,z + f_2 \cdot \overline{x}y\overline{z} + f_3 \cdot \overline{x}yz + f_4 \cdot x\overline{y}\,\overline{z} + f_5 \cdot x\overline{y}z + f_6 xy\overline{z} + f_7 \cdot xyz.$$

- The Boolean expression for an 8-to-1-line multiplexer is:

$$f = \left(I_0\overline{S}_2\overline{S}_1\overline{S}_0 + I_1\overline{S}_2\overline{S}_1S_0 + I_2\overline{S}_2S_1\overline{S}_0 + I_3\overline{S}_2S_1S_0 + I_4S_2\overline{S}_1\overline{S}_0 + I_5S_2\overline{S}_1S_0 + I_6S_2S_1\overline{S}_0 + I_7S_2S_1S_0\right).$$

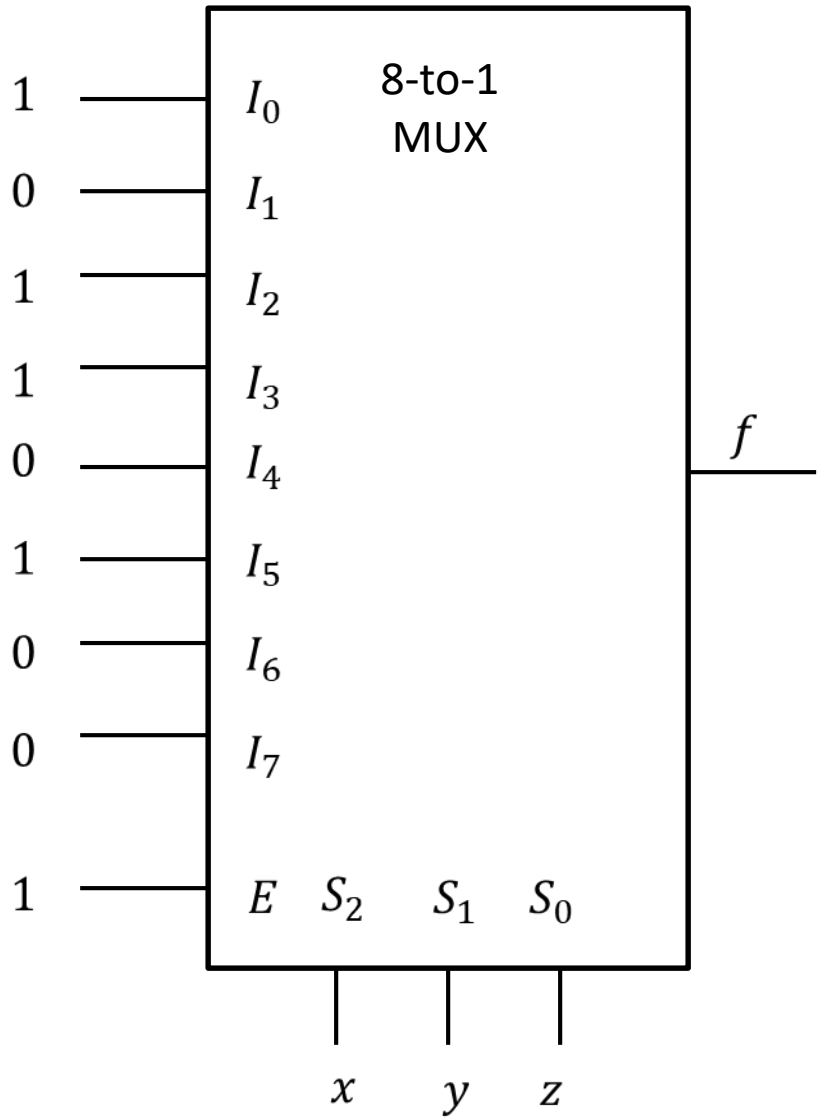
Logic Design with Multiplexers

- If E is logic-1 then the latter is transformed into the former by replacing I_i with f_i, S_2 with x, S_1 with y, and S_0 with z.
- Placing x, y, z on the select lines S_2, S_1, S_0, respectively and placing the functional values f_i on data input lines I_i.



Example:

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Logic Design with Multiplexers

- If at least one input variable of a Boolean function is available in both its complemented and uncomplemented form, any n-variable function is realizable with a 2^{n-1}-to-1-line multiplexer.
- For the case of a 3-variable function, only a 4-to-1 multiplexer is needed.
- $f(x, y, z) = f_0 \cdot \overline{x}\,\overline{y}\,\overline{z} + f_1 \cdot \overline{x}\,\overline{y}\,z + f_2 \cdot \overline{x}y\overline{z} + f_3 \cdot \overline{x}yz + f_4 \cdot x\overline{y}\,\overline{z} + f_5 \cdot x\overline{y}z + f_6 xy\overline{z} + f_7 \cdot xyz$

$$= (f_0 \cdot \overline{z} + f_1 \cdot z)\overline{x}\,\overline{y} + (f_2 \cdot \overline{z} + f_3 \cdot z)\overline{x}y$$
$$+ (f_4 \cdot \overline{z} + f_5 \cdot z)x\overline{y} + (f_6 \cdot \overline{z} + f_7 \cdot z)xy$$

- When E = 1, 4-to-1 Multiplexer has the form

$$I_0\overline{S}_1\overline{S}_0 + I_1\overline{S}_1S_0 + I_2S_1\overline{S}_0 + I_3S_1S_2$$

Logic Design with Multiplexers

$$f(x, y, z)$$
$$= (f_0 \cdot \overline{z} + f_1 \cdot z)\overline{x}\,\overline{y} + (f_2 \cdot \overline{z} + f_3 \cdot z)\overline{x}y$$
$$+ (f_4 \cdot \overline{z} + f_5 \cdot z)x\overline{y} + (f_6 \cdot \overline{z} + f_7 \cdot z)xy$$

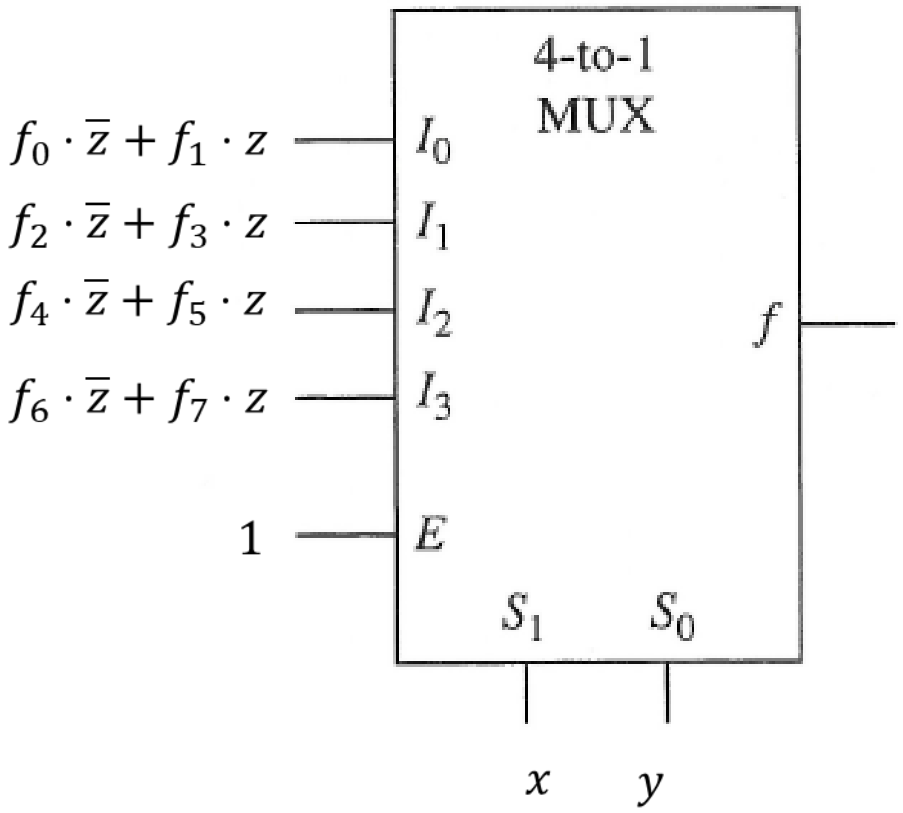
4-to-1 Multiplexer has the form

$$f = I_0\overline{S}_1\overline{S}_0 + I_1\overline{S}_1S_0 + I_2S_1\overline{S}_0 + I_3S_1S_2$$

- Realization of $f(x, y, z)$ is obtained by placing the x and y variables on the S_1, S_0 select lines, the single variable functions $f_i \cdot \overline{z} + f_j \cdot z$ on the data input lines and let E = 1.
- Note: $f_i \cdot \overline{z} + f_j \cdot z$ reduce to 0,1,z or $\overline{z}$.

Example

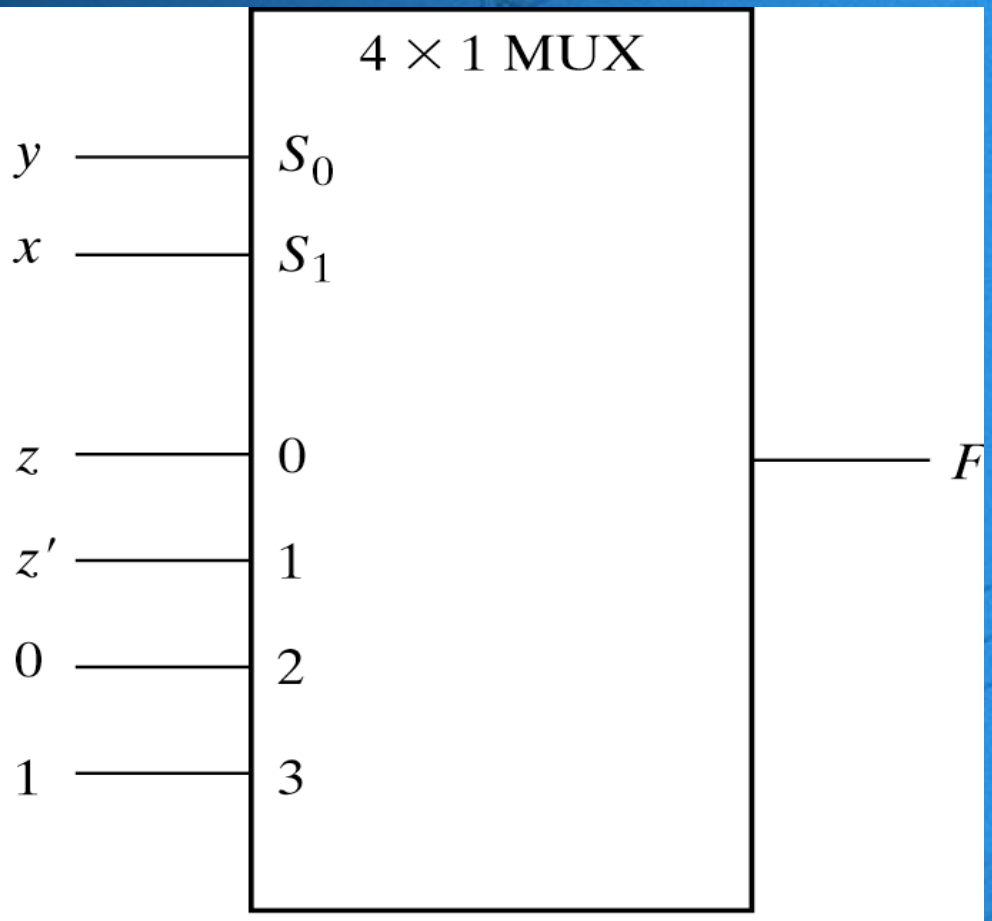
x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Boolean Function Implementation

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table



(b) Multiplexer implementation

Fig. 4-27 Implementing a Boolean Function with a Multiplexer

Example

How do you implement it with 8x1 MUX?

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	

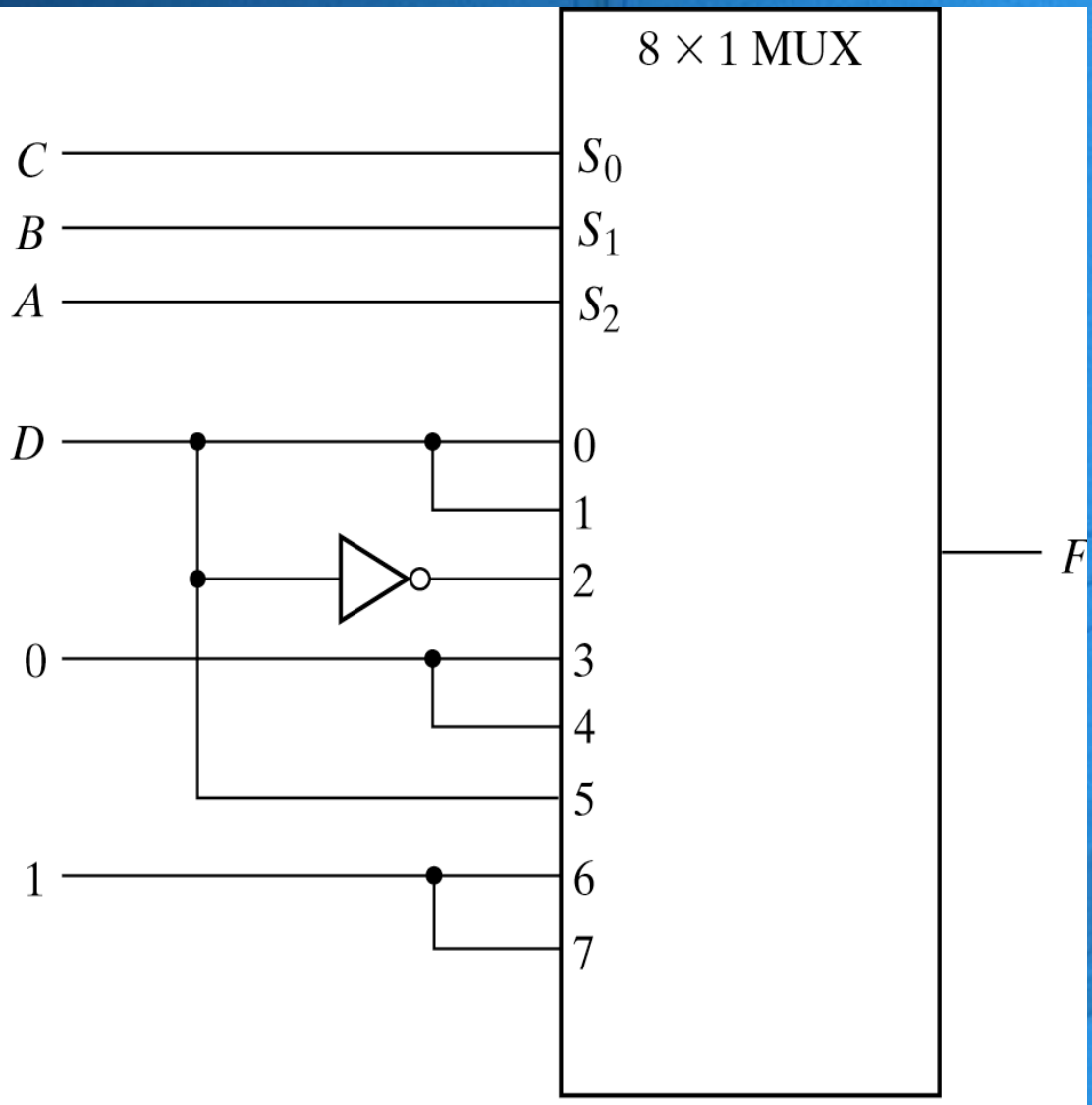
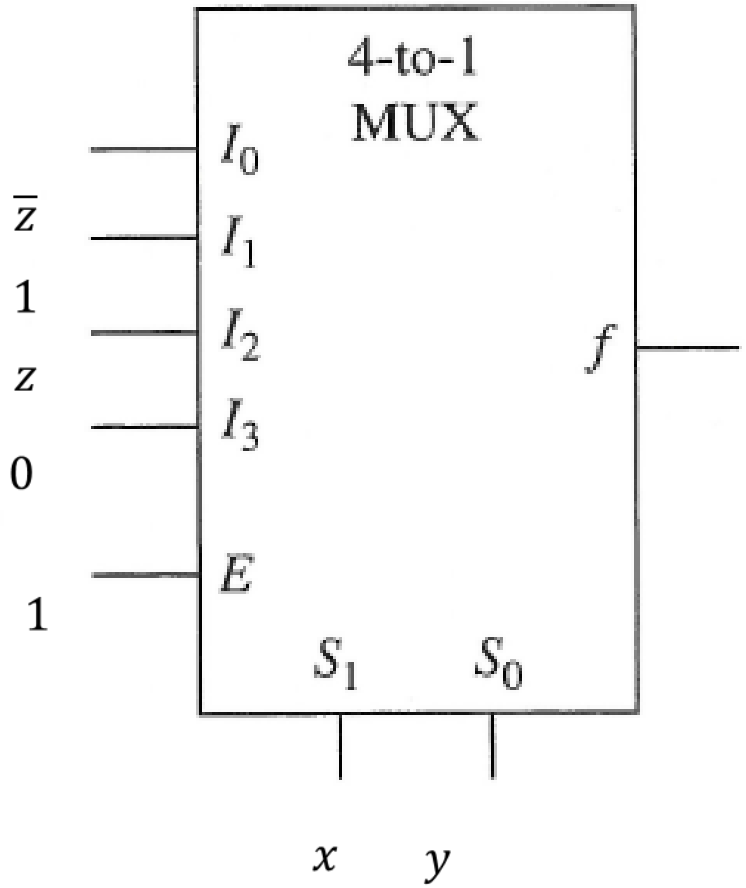


Fig. 4-28 Implementing a 4-Input Function with a Multiplexer

Example

How do you implement it with 4x1 MUX?

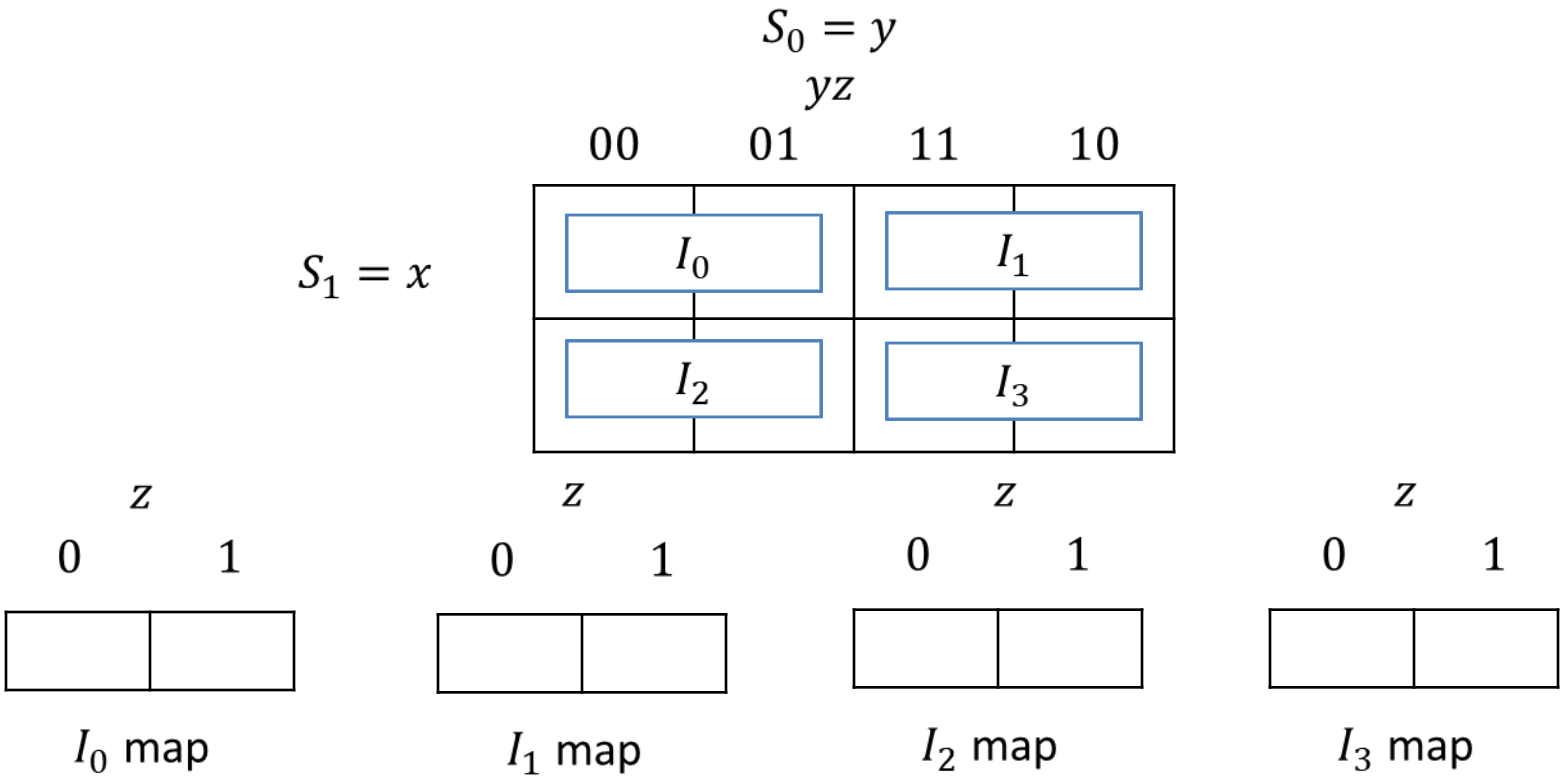
x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Logic Design with Multiplexers and K-maps

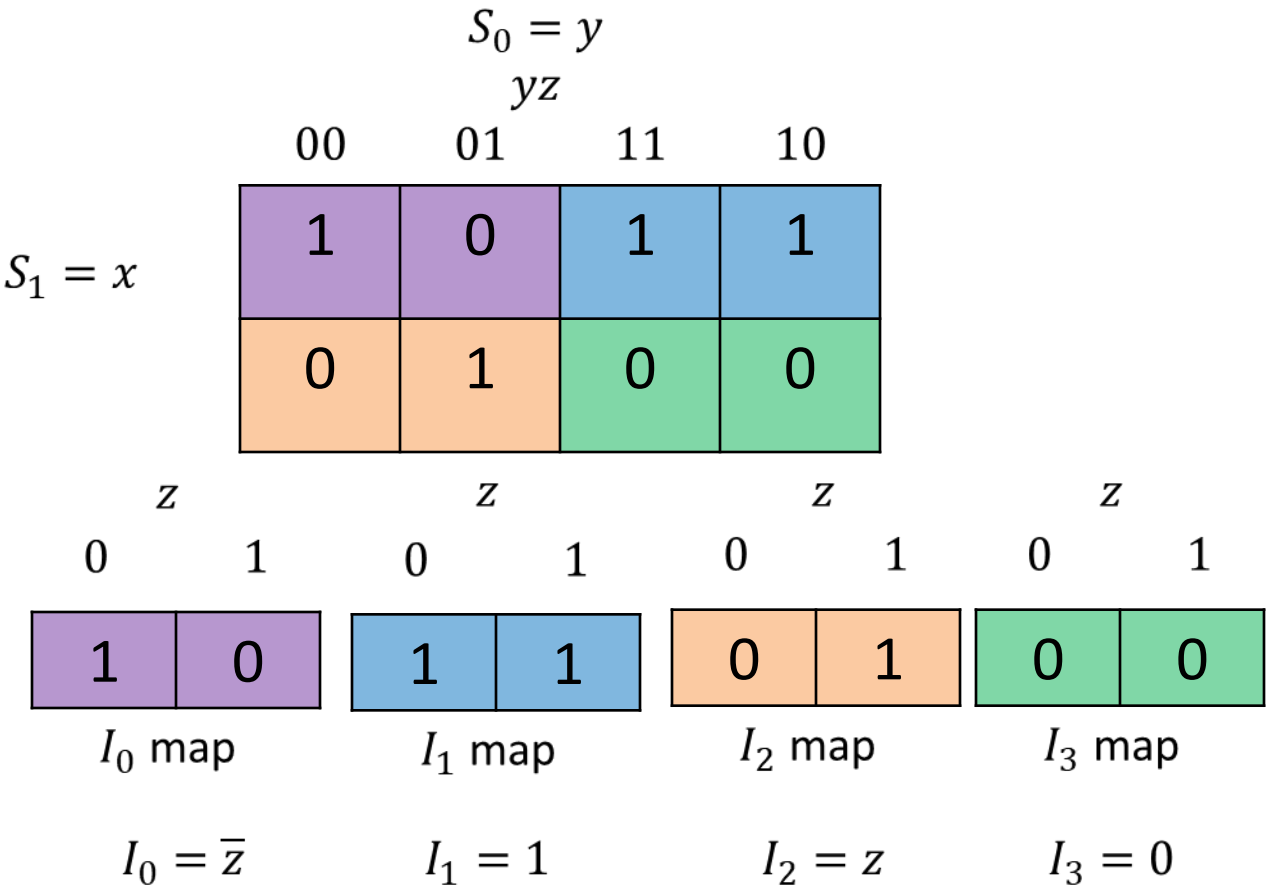
- Consider 3-variable Karnaugh map. Assume x is placed on the S_1 line and y is placed on the S_0 line.
- We get that the output is: $I_0\overline{x}\,\overline{y} + I_1\overline{x}y + I_2x\overline{y} + I_3xy$
- $I_0\overline{x}\,\overline{y}$ corresponds to those cells in which $x = 0, y = 0$
- $I_1\overline{x}y$ corresponds to those cells in which $x = 0, y = 1$
- $I_2x\overline{y}$ corresponds to those cells in which $x = 1, y = 0$
- I_3xy corresponds to those cells in which $x = 1, y = 1$

K-map representation



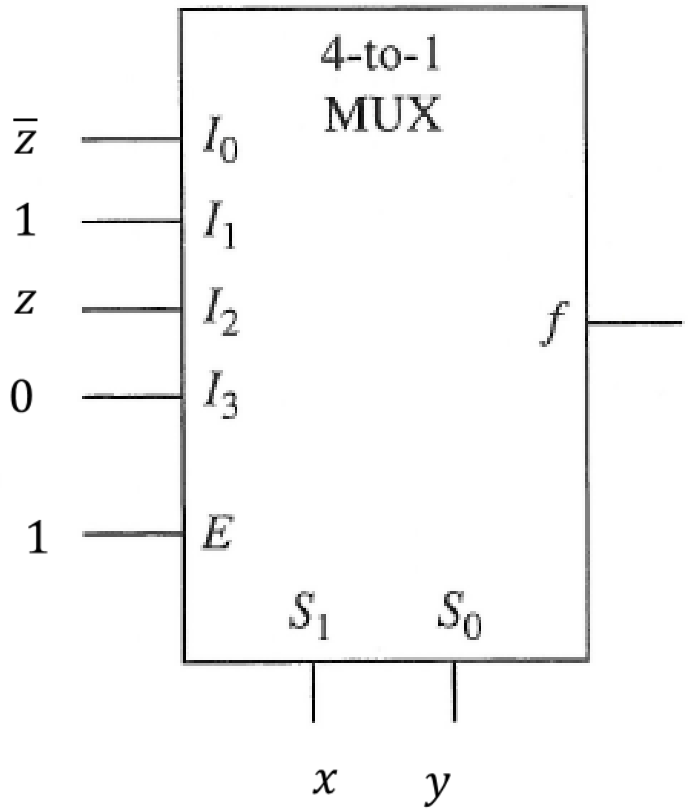
Example

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



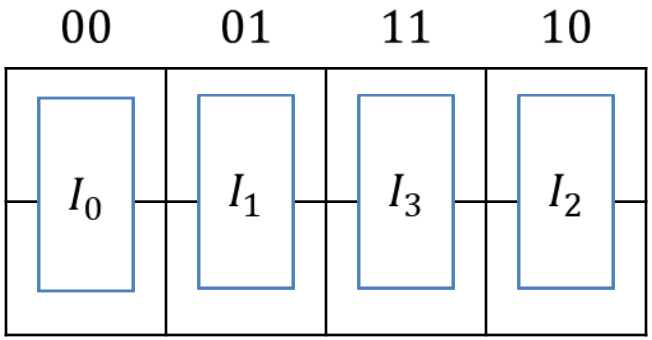
Realization

x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

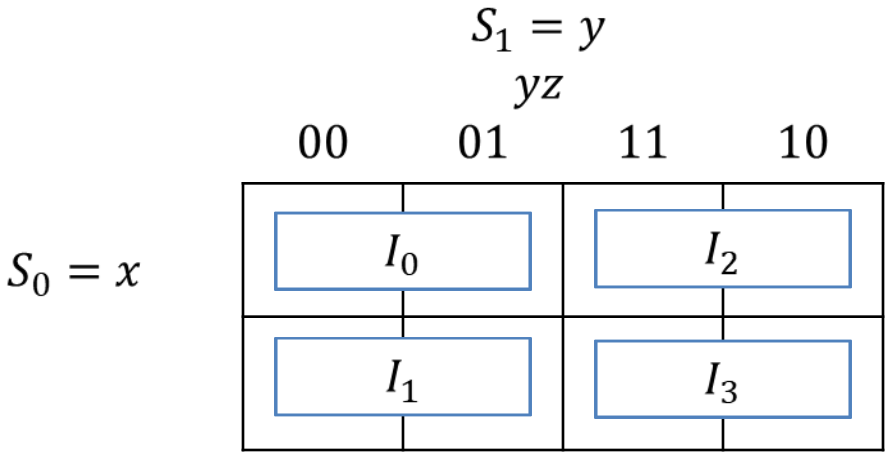


Alternative Structures

$S_1 = y, S_0 = z$



$S_1 = y$



Note that order of variables on input lines matters!

8-to-1-line multiplexers and 4-variable Boolean functions

- Can do the same thing, three variables are placed on select lines, inputs to the data lines are single-variable functions.
- Example:

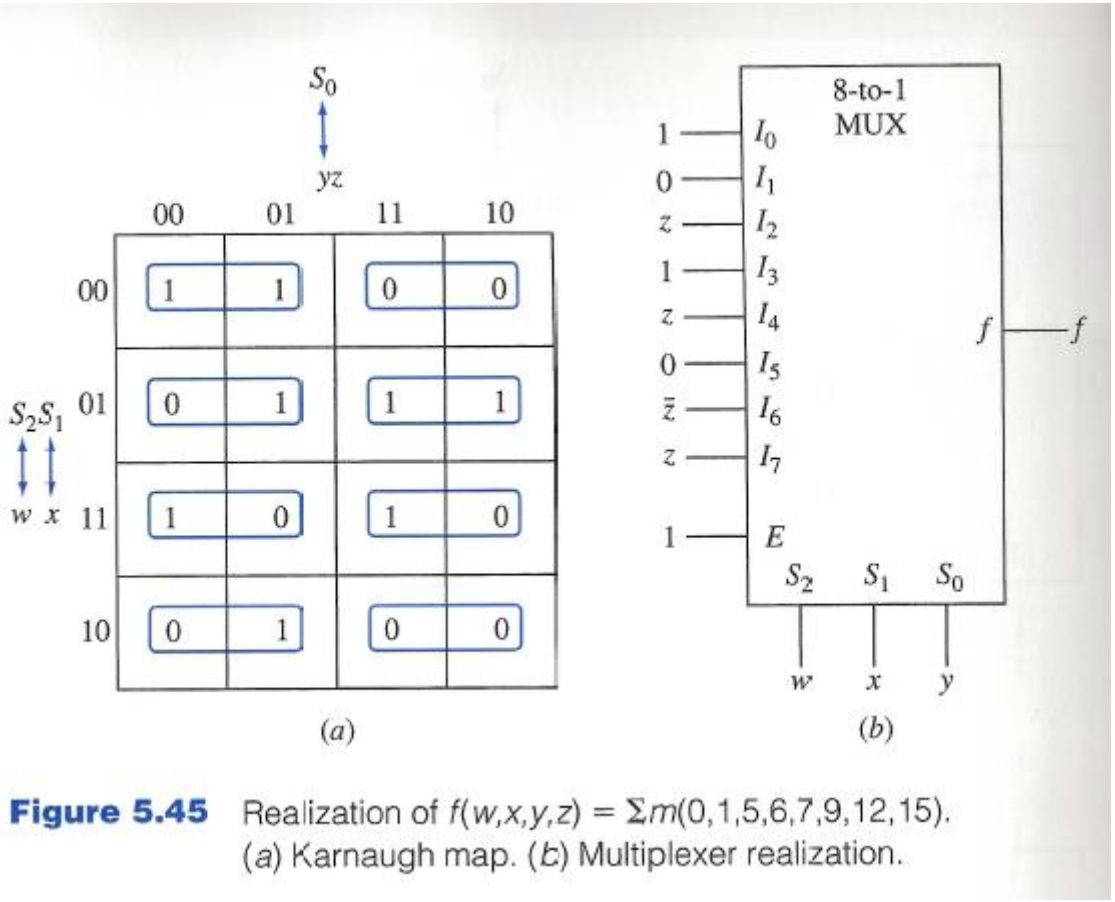


Figure 5.45 Realization of $f(w,x,y,z) = \Sigma m(0,1,5,6,7,9,12,15)$. (*a*) Karnaugh map. (*b*) Multiplexer realization.

Can we do better?

- By allowing realizations of m-variable functions as inputs to the data input lines, 2^n-to-1-line multiplexers can be used in the realization of $(n + m)$-variable functions.
- E.g.: input variables w and x are applied to the S_1, S_0 select inputs. Functions of the y and z variables appear at the data input lines.

K-map Structure

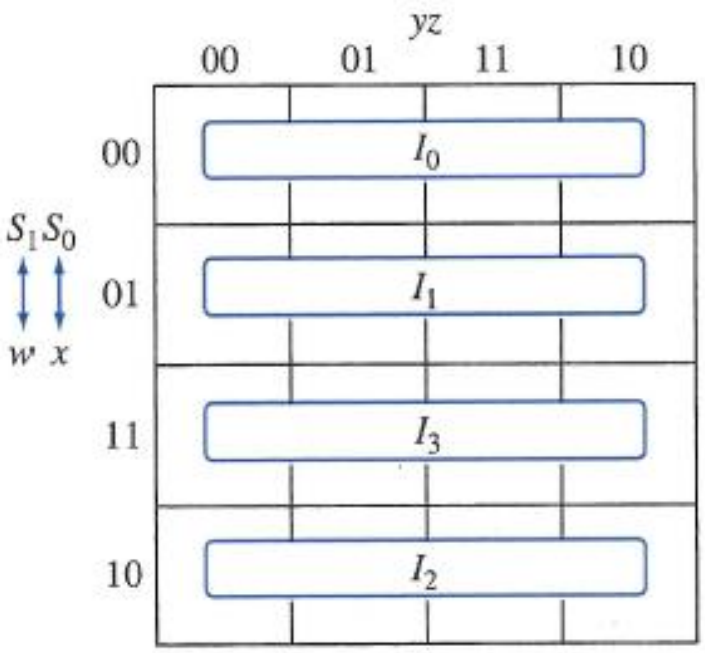
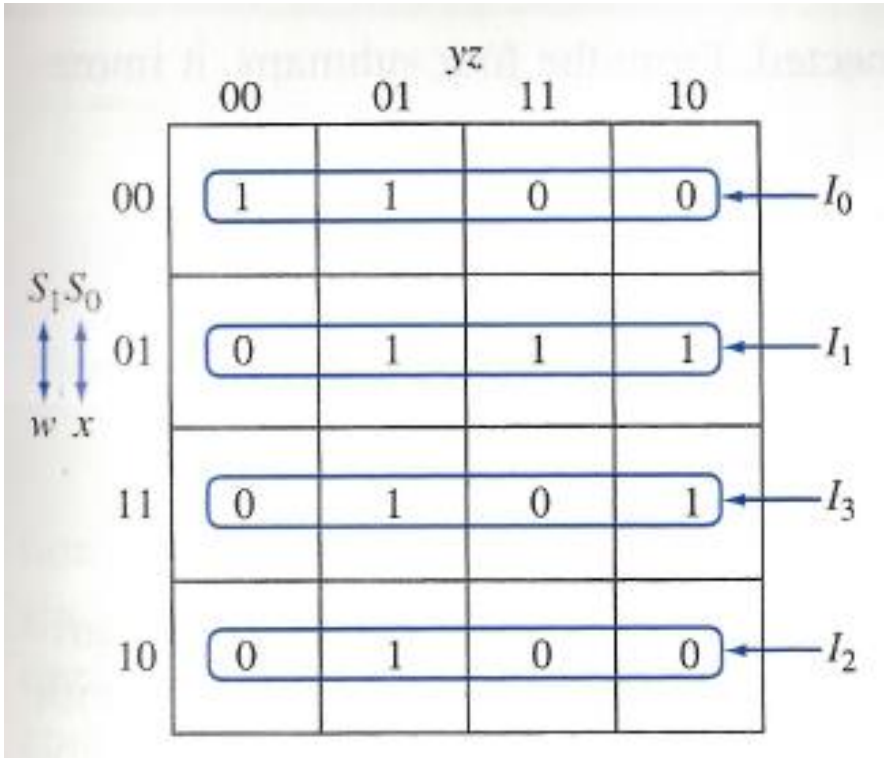
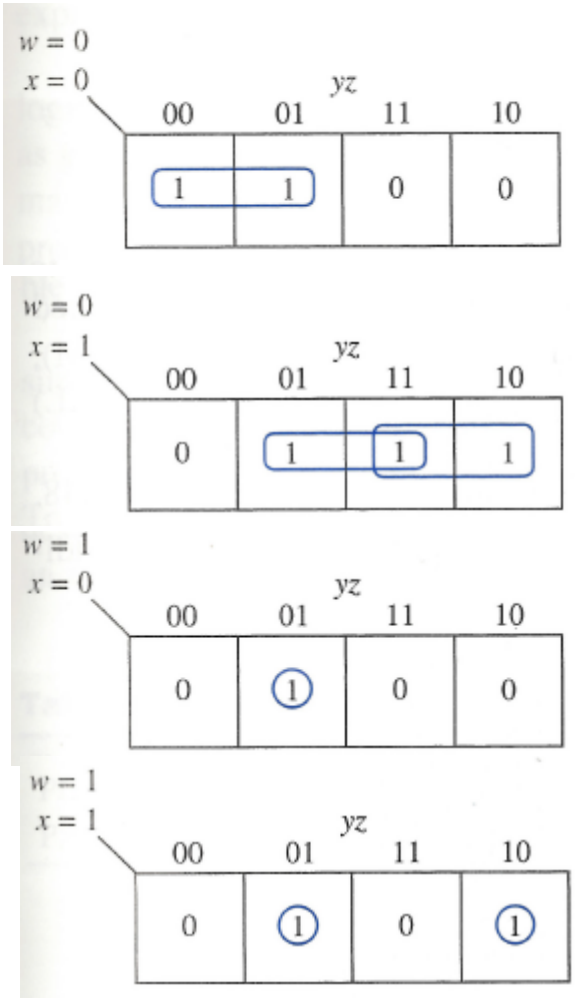


Figure 5.46 Using a four-variable Karnaugh map to obtain a Boolean function realization with a 4-to-1-line multiplexer.

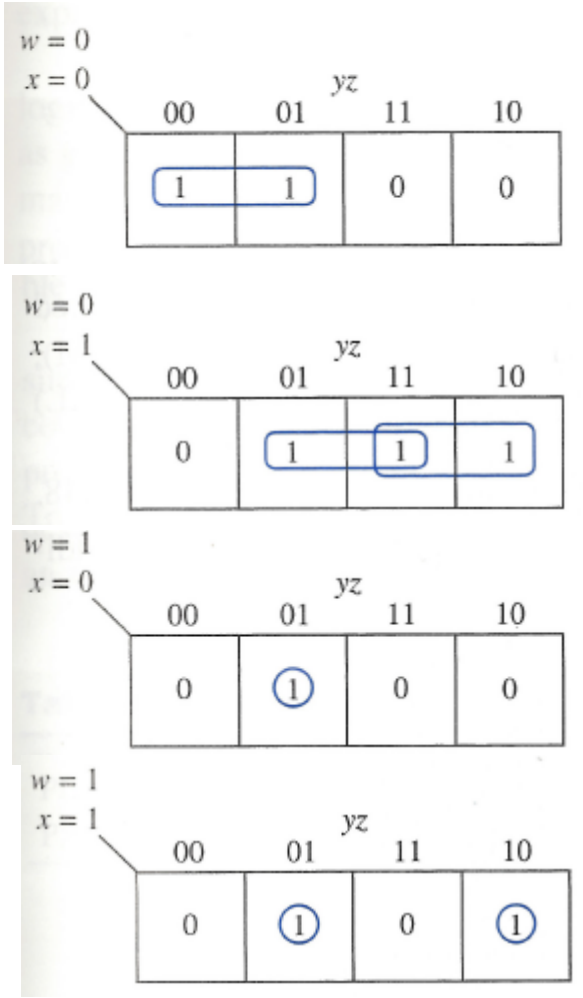
Example:

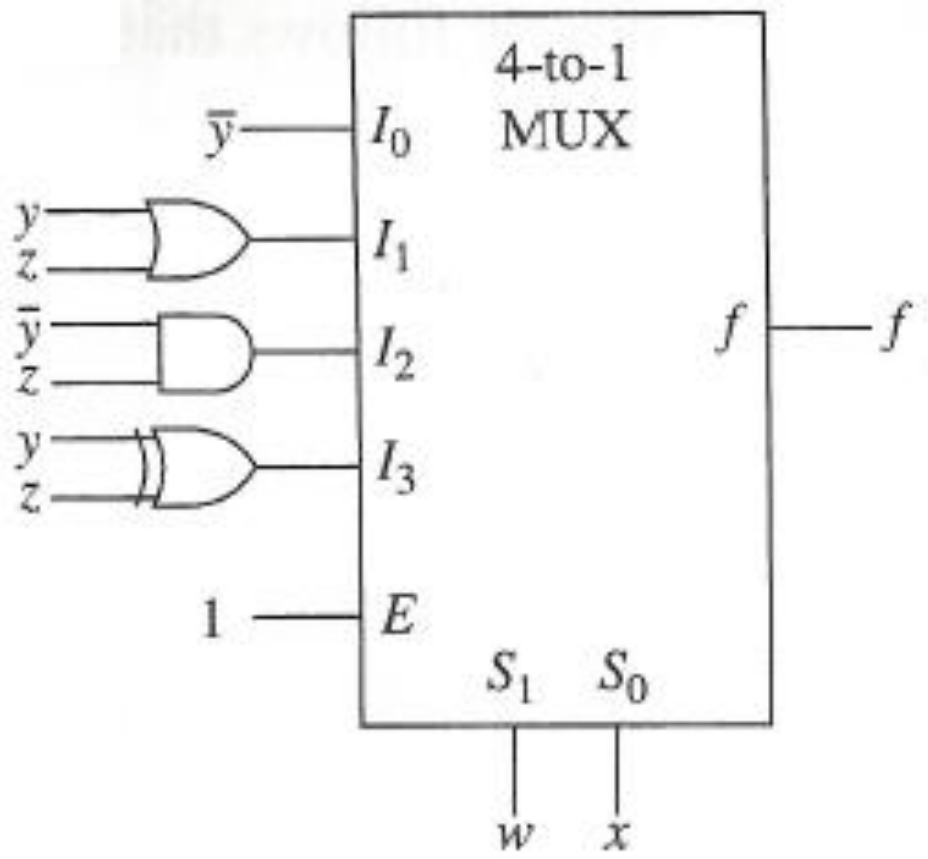
$$f(x, y, z) = \sum m(0,1,5,6,7,9,13,14)$$



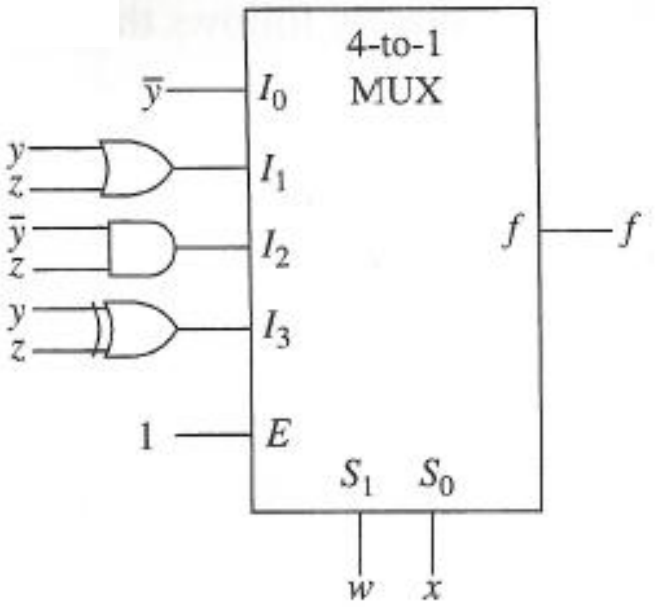


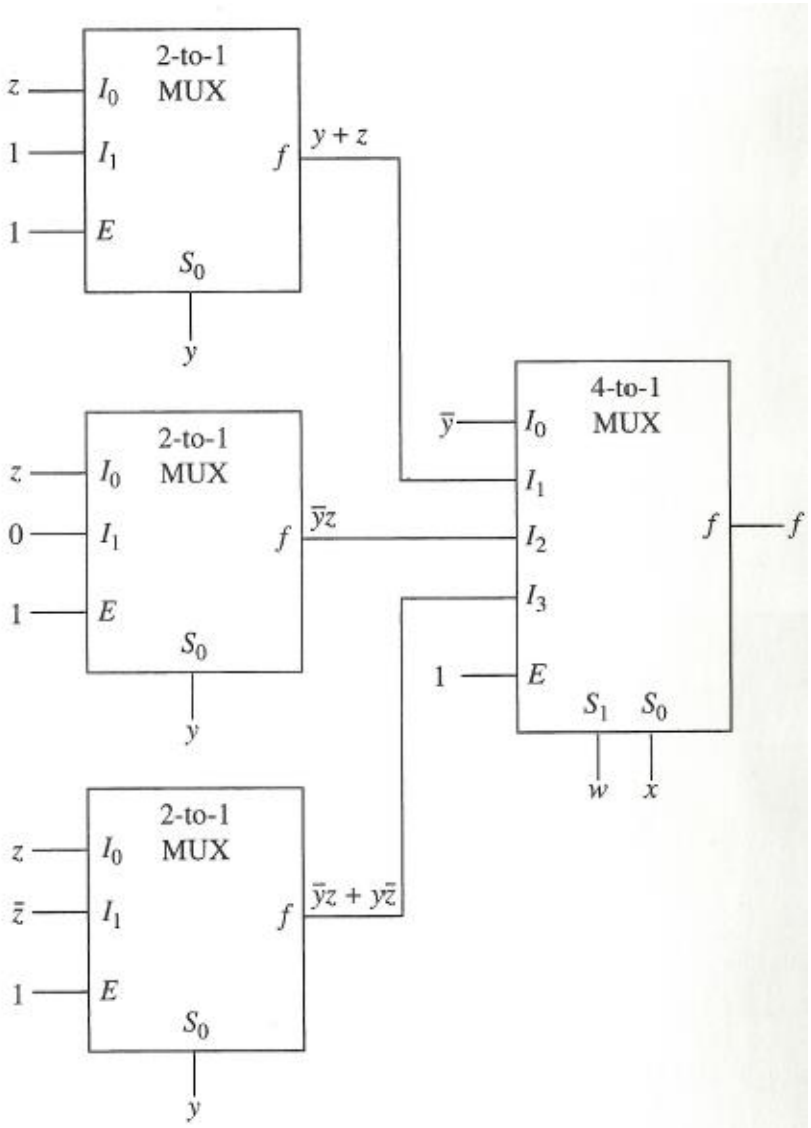
Example


w = 0
x = 0
yz
00 01 11 10
1 1 0 0
w = 0
x = 1
yz
00 01 11 10
0 1 1 1
w = 1
x = 0
yz
00 01 11 10
0 1 0 0
w = 1
x = 1
yz
00 01 11 10
0 1 0 1


4-to-1
MUX
$\bar{y}$
I_0
y
z
I_1
$\bar{y}$
z
I_2
y
z
I_3
1
E
S_1
S_0
f
f
w
x

Example



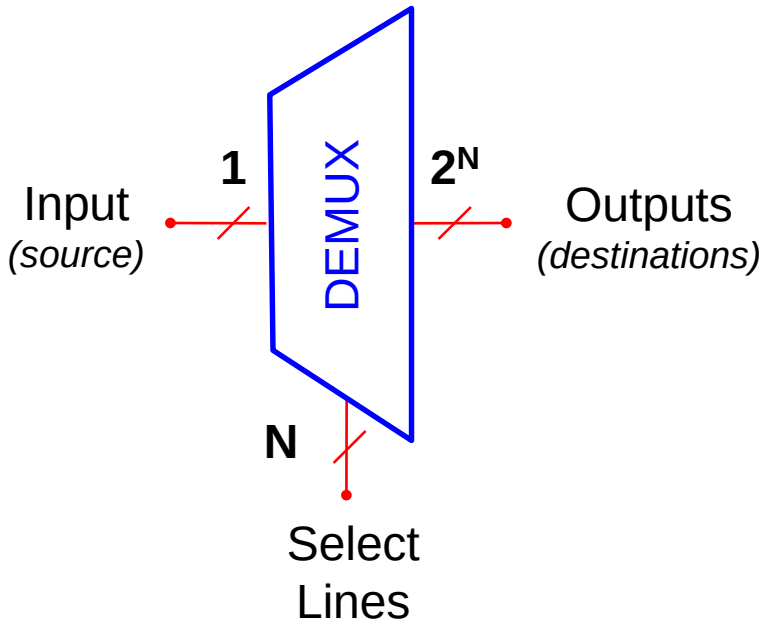


Multiplexer Tree

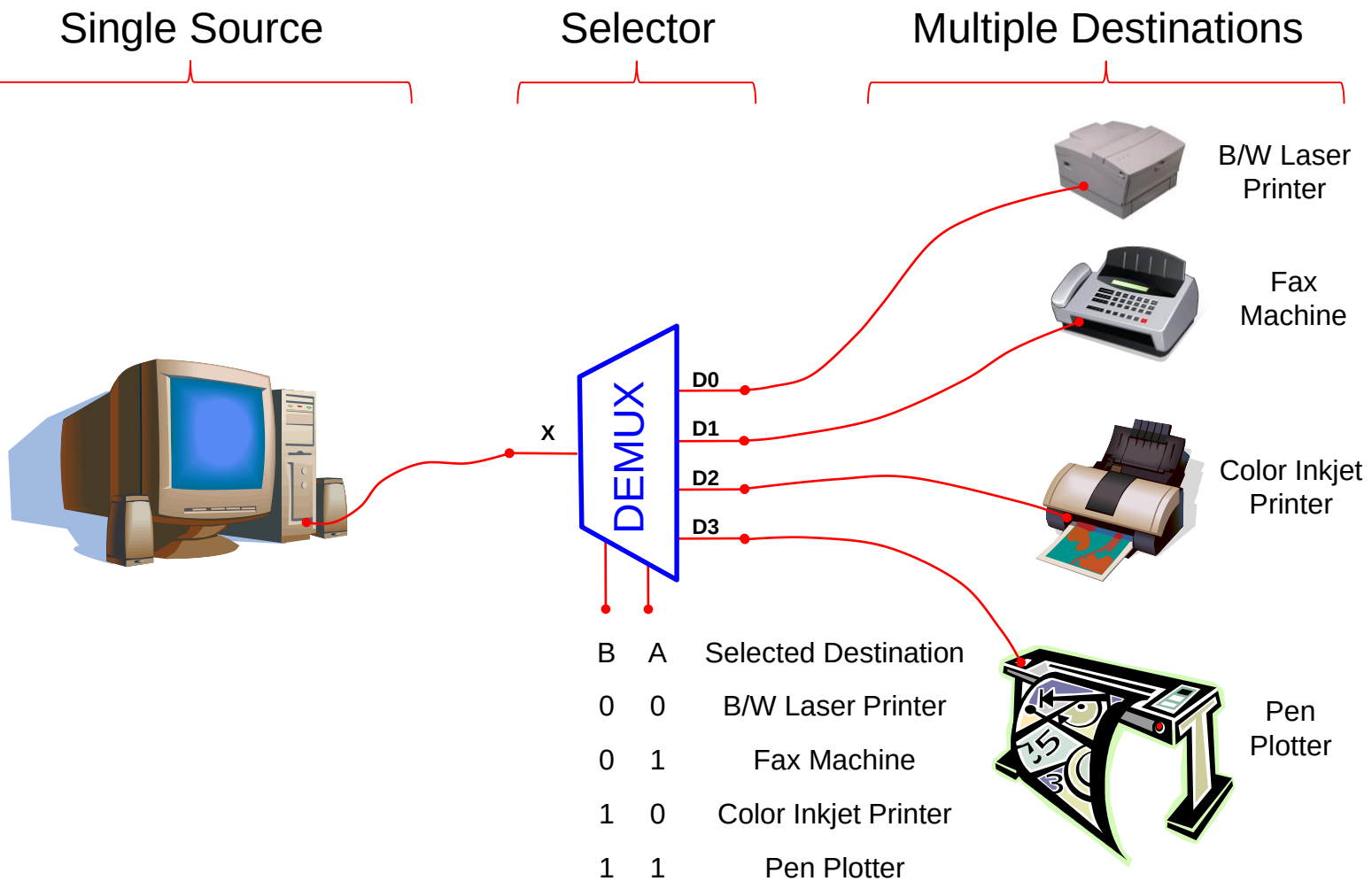
What is a Demultiplexer (DEMUX)?

- A DEMUX is a digital switch with a single input (source) and a multiple outputs (destinations).
- The select lines determine which output the input is connected to.
- DEMUX Types
  - → 1-to-2 (1 select line)
  - → 1-to-4 (2 select lines)
  - → 1-to-8 (3 select lines)
  - → 1-to-16 (4 select lines)

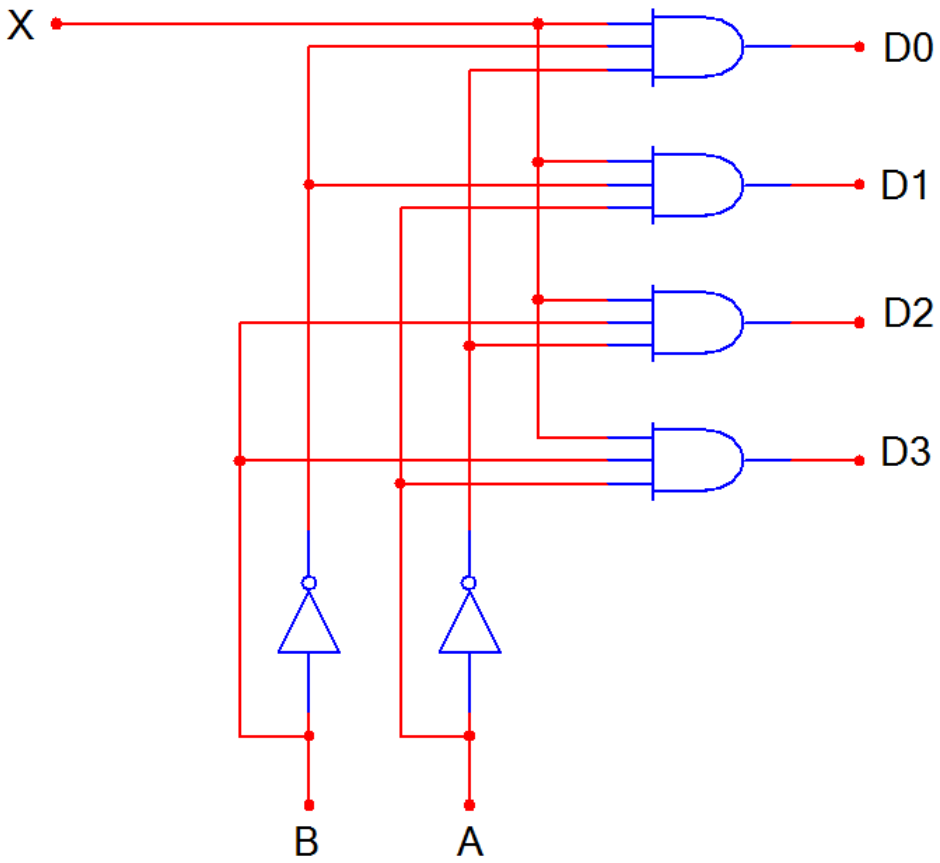
Demultiplexer
Block Diagram

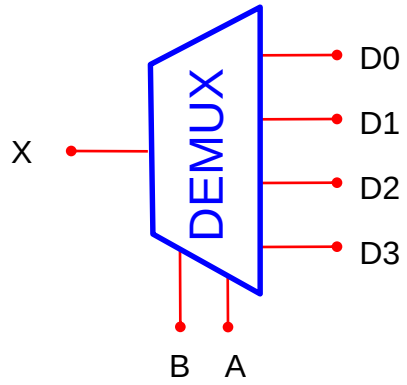


Typical Application of a DEMUX



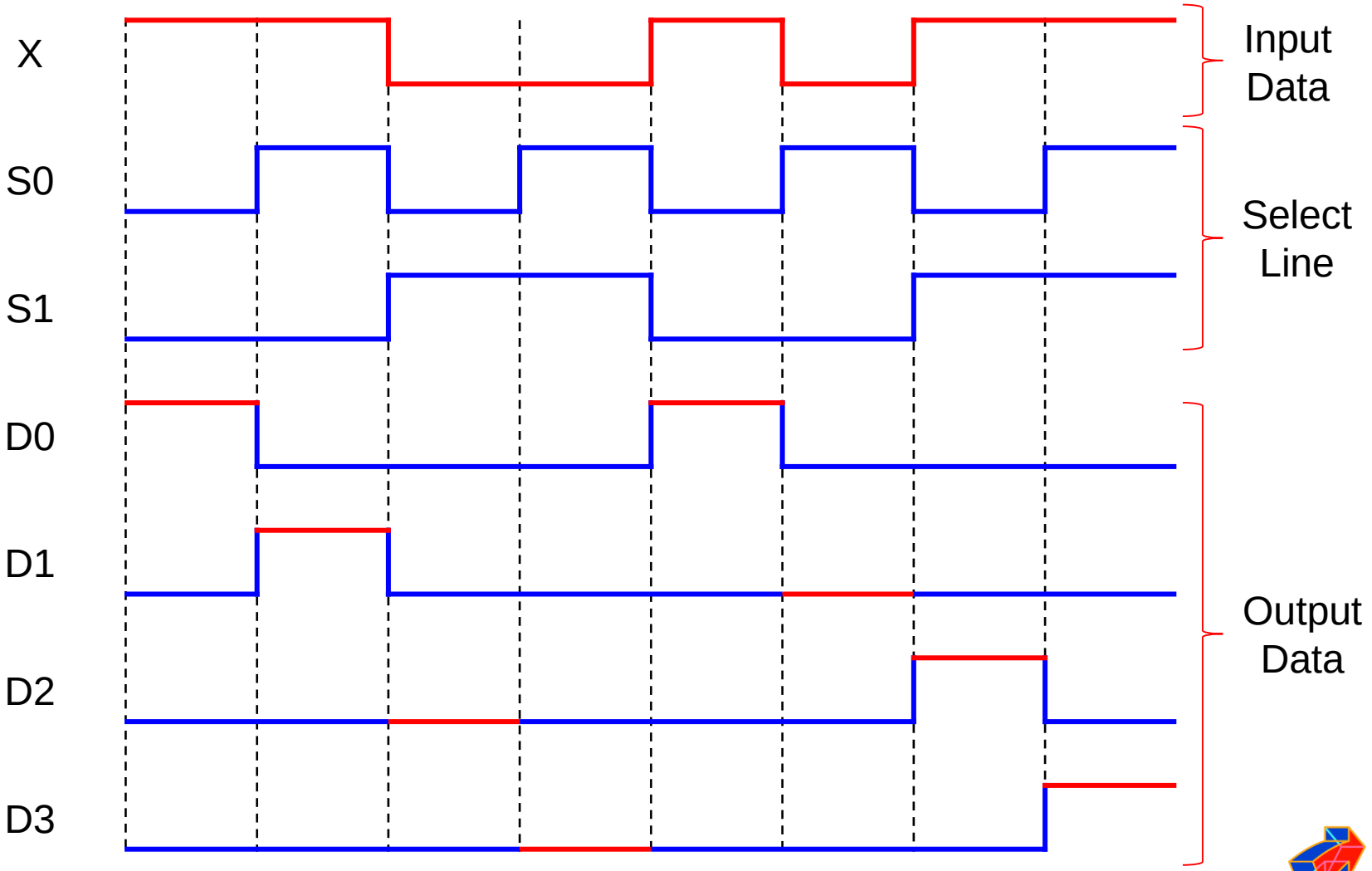
1-to-4 De-Multiplexer (DEMUX)





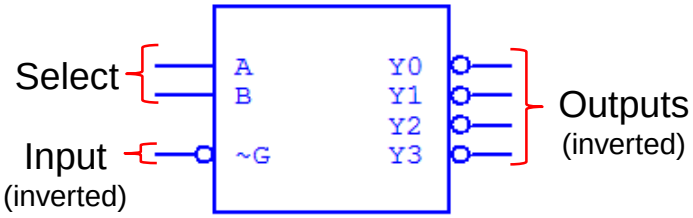
B	A	D0	D1	D2	D3
0	0	X	0	0	0
0	1	0	X	0	0
1	0	0	0	X	0
1	1	0	0	0	X

1-to-4 De-Multiplexer Waveforms

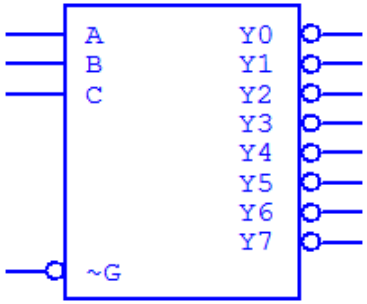


Medium Scale Integration DEMUX

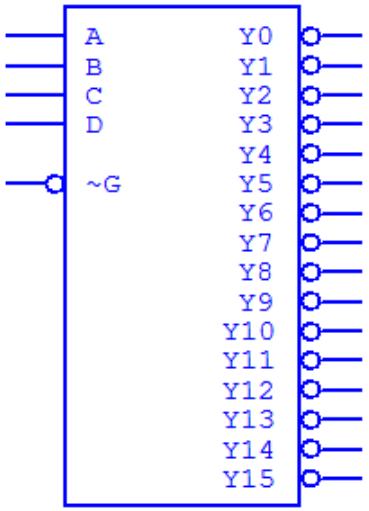
1-to-4 DEMUX



1-to-8 DEMUX



1-to-16 DEMUX



Note : Most Medium Scale Integrated (MSI) DEMUXs , like the three shown, have outputs that are inverted. This is done because it requires few logic gates to implement DEMUXs with inverted outputs rather than no-inverted outputs.

Study Problem

- **Course Book Chapter – 4 Problems**
  - **4 – 31**
    - ✓ **Construct a 16x1 multiplexer with two 8x1 and one 2x1 multiplexer. Use block diagrams**

Study Problem

- **Course Book Chapter – 4 Problems**
  - **4 – 34**

An 8x1 multiplexe r has inputs A, B, and C connected to the selection inputs S_2, S_1, and S_0 respectively.

The data inputs

$I_1 = I_2 = I_7 = 0;$

$I_3 = I_5 = 1;$

$I_0 = I_4 = D;$

$I_6 = D'$

Determine the Boolean function that the multiplexe r implements

Study Problems

- Course Book Chapter – 4 Problems
  - 4 – 1
  - 4 – 4
  - 4 – 6
  - 4 – 11
  - 4 – 20
  - 4 – 21
  - 4 – 25
  - 4 – 32
  - 4 – 33
  - 4 – 35