Implementing Boolean Functions with Multiplexers

- The most common applications of a multiplexer is its use for implementation of combinational logic Boolean functions

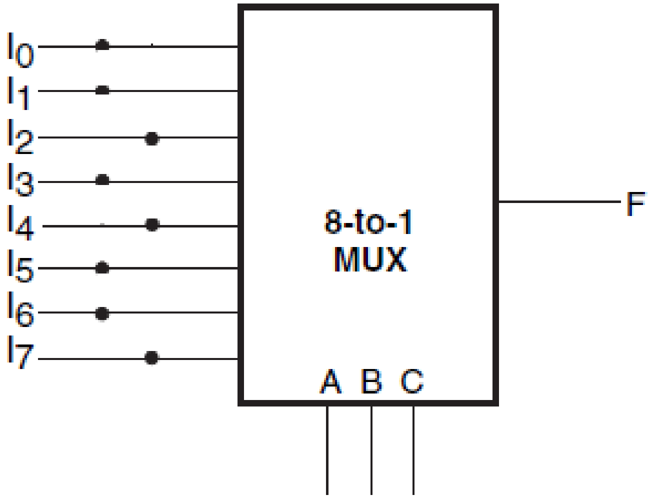
- The simplest technique for doing so is to employ a 2n-to-1 MUX to implement an n-variable Boolean function..

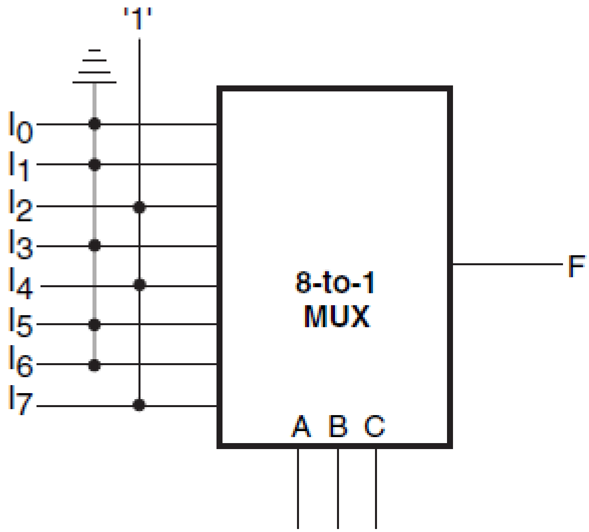
- The input lines corresponding to each of the minterms present in the Boolean function are made equal to logic ‘1’ state.

- The remaining minterms that are absent in the Boolean function are disabled by making their corresponding input lines equal to logic ‘0’.

8-to-1 MUX for implementing the Boolean function given by the equation

$$f(A, B, C) = \sum 2, 4, 7$$





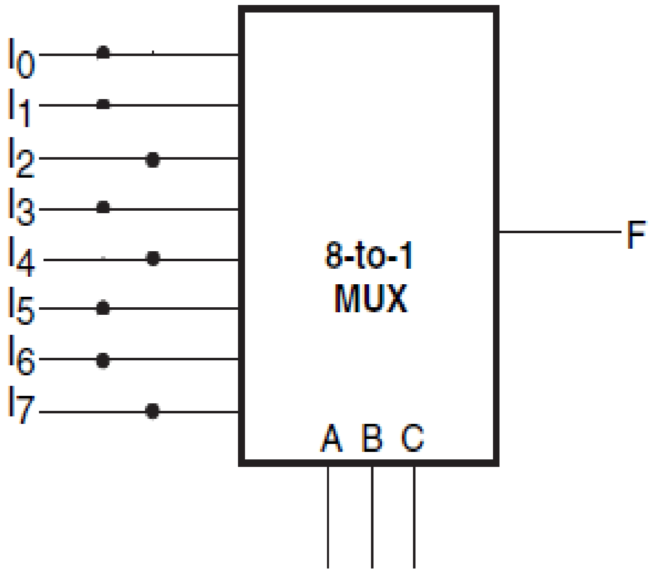
Cascading Multiplexer Circuits

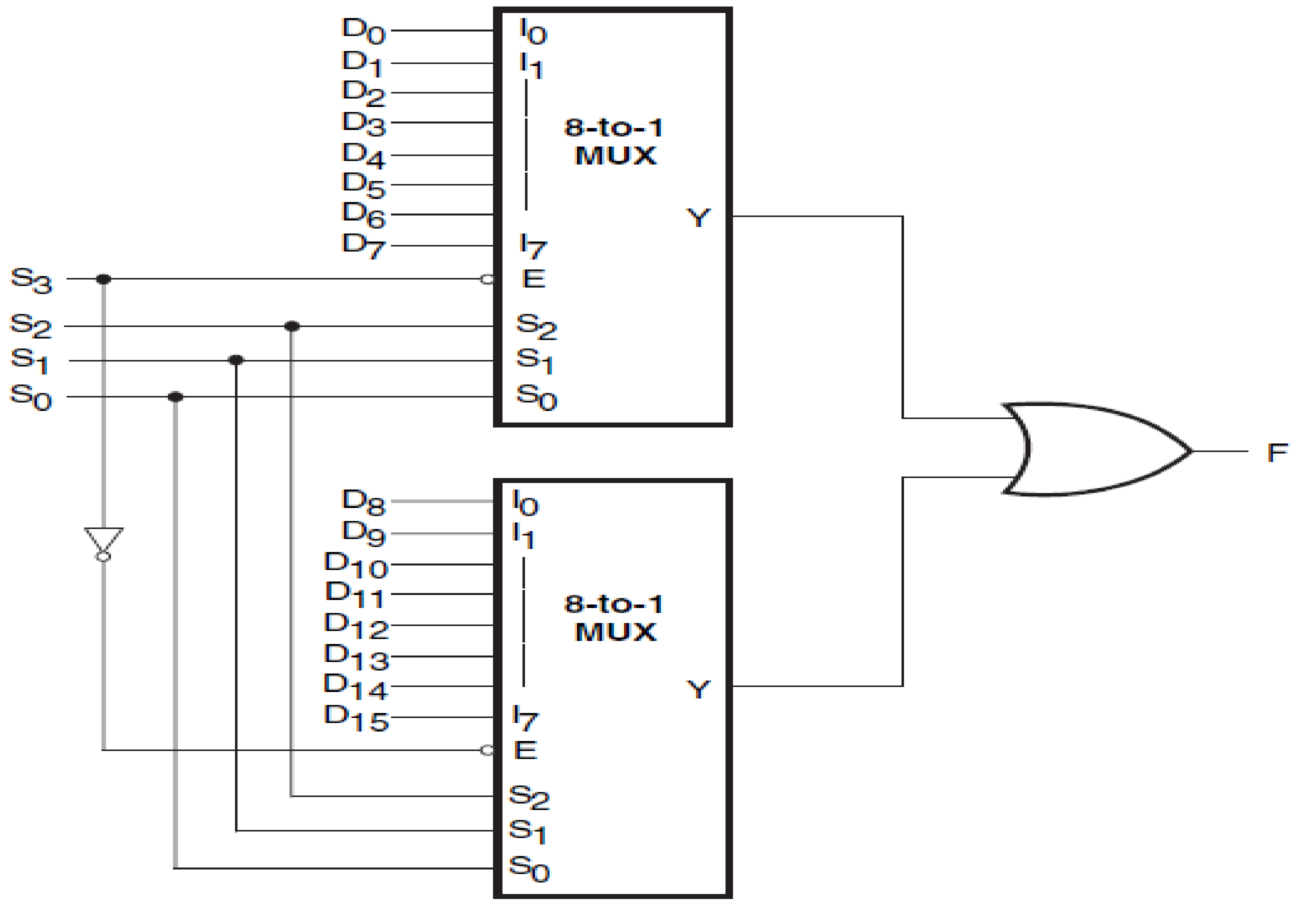
- A multiple number of devices of a given size can be used to construct multiplexers that can handle a larger number of input channels

- For instance, 8-to-1 multiplexers can be used to construct 16-to-1 or 32-to-1 or even larger multiplexer circuits.

Example 8.3

- *Design a 16-to-1 multiplexer using two 8-to-1 multiplexers having an active LOW ENABLE input.*
- Two 8-to-1 multiplexers having an ENABLE input.
  - The ENABLE input is taken as the fourth selection variable occupying the MSB position.
  - Figure shows the complete logic circuit diagram




D0
D1
D2
D3
D4
D5
D6
D7
I0
I1
I7
E
S2
S1
S0
8-to-1
MUX
Y
S3
S2
S1
S0
D8
D9
D10
D11
D12
D13
D14
D15
I0
I1
I7
E
S2
S1
S0
8-to-1
MUX
Y
F

Implementing Functions Using Decoders

- Any n-variable logic function can be implemented using a single n-to-2^n decoder to generate the minterms
  - OR gate forms the sum.
  - The output lines of the decoder corresponding to the minterms of the function are used as inputs to the or gate.

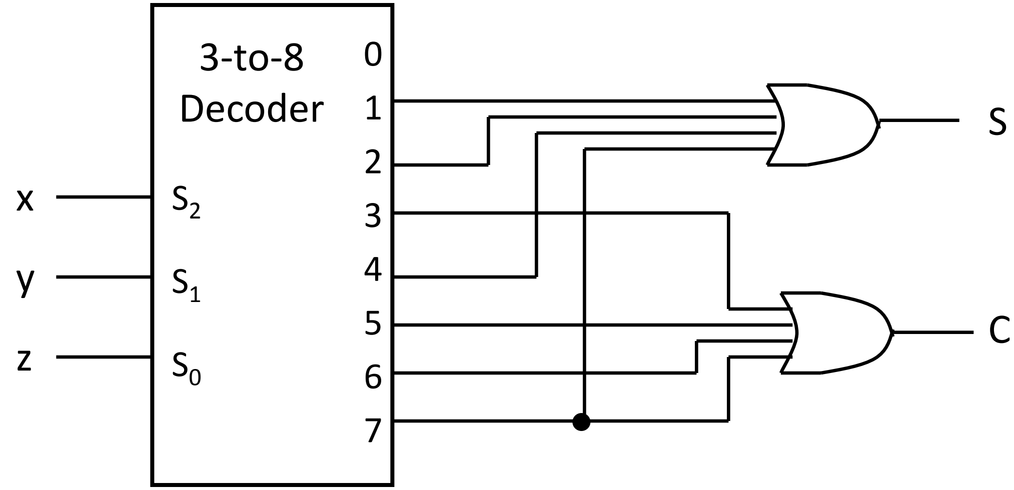
- Any combinational circuit with *n* inputs and *m* outputs can be implemented with an n-to-2^n decoder with *m* OR gates.
- Suitable when a circuit has many outputs, and each output function is expressed with few minterms.

Implementing Functions Using Decoders

- Example: Full adder

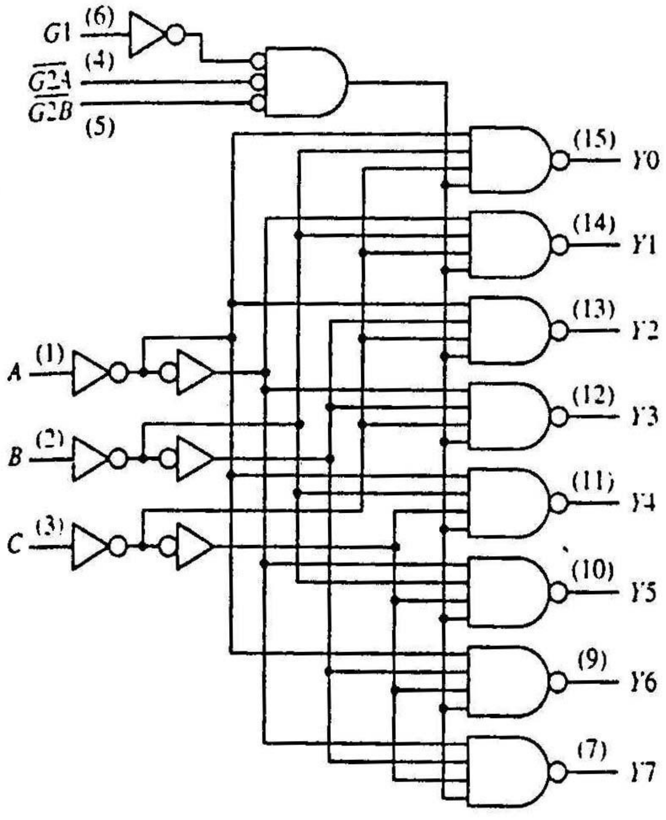
  $S(x, y, z) = \Sigma\ (1,2,4,7)$
  $C(x, y, z) = \Sigma\ (3,5,6,7)$

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

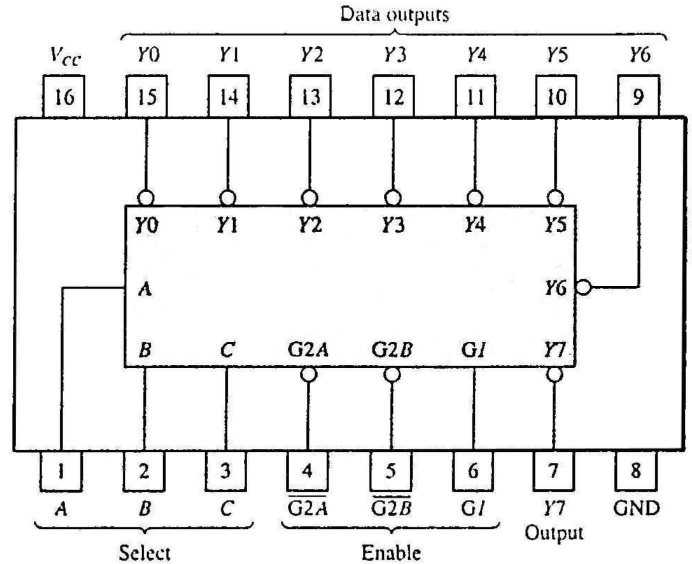


Standard MSI Binary Decoders Example

74138 (3-to-8 decoder)



(a)



(b)

Inputs					Outputs							
Enable		Select										
G1	$\overline{G2}$*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L
×	H	×	×	×	H	H	H	H	H	H	H	H
L	×	×	×	×	H	H	H	H	H	H	H	H

$$\overline{G2}^* = \overline{G2A} + \overline{G2B}$$

(c)

(a) Logic circuit.
(b) Package pin configuration.
(c) Function table.

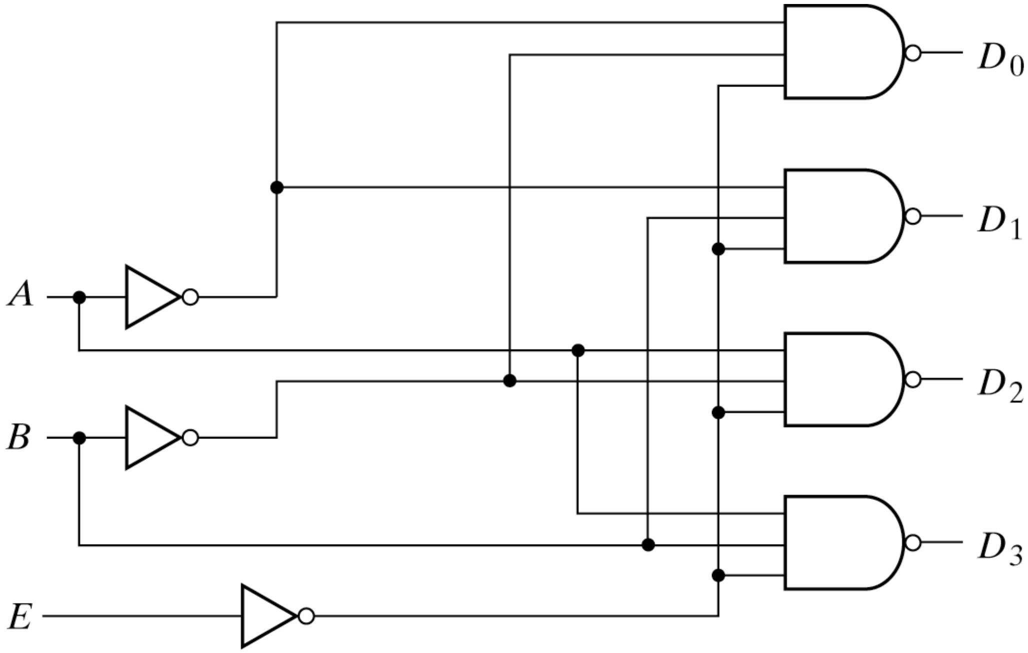
Building a Binary Decoder with NAND Gates

- Start with a 2-bit decoder
  - Add an enable signal (E)

Note: use of NANDs

only one 0 active!

if E = 0



(a) Logic diagram

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table

Fig. 4-19 2-to-4-Line Decoder with Enable Input

Use two 3 to 8 decoders to make 4 to 16 decoder

- Enable can also be active high
- In this example, only one decoder can be active at a time.
- x, y, z effectively select output line for w

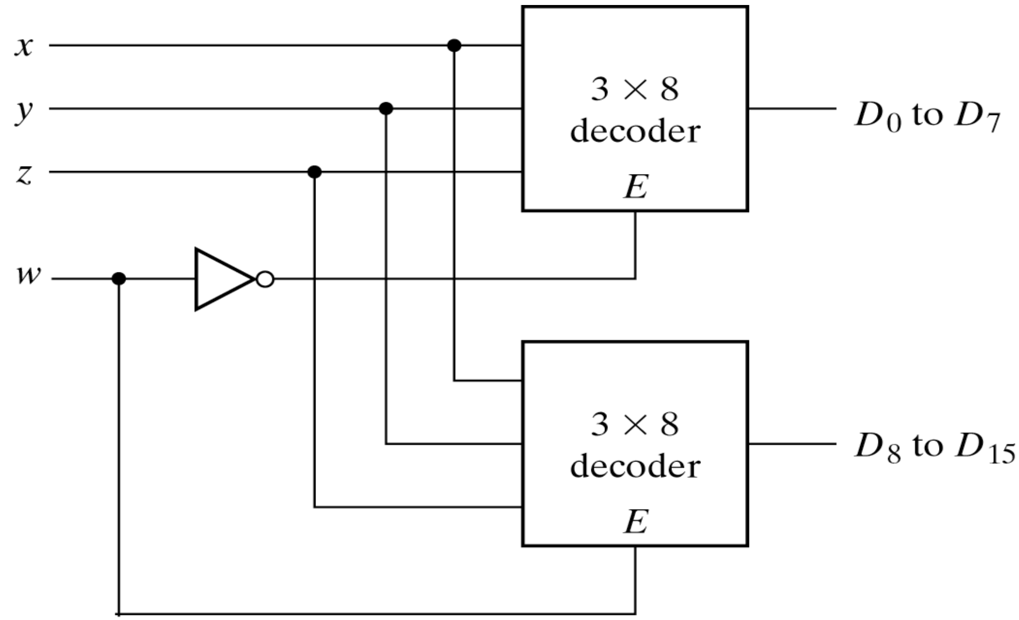


Fig. 4-20 4 × 16 Decoder Constructed with Two 3 × 8 Decoders

BCD Display or 7 segment

decimal		A	B	C	D		a	b	c	d	e	f	g
0		0	0	0	0		**1**	**1**	**1**	**1**	**1**	**1**	**0**
1		0	0	0	1		**0**	**1**	**1**	**0**	**0**	**0**	**0**
2		0	0	1	0		**1**	**1**	**0**	**1**	**1**	**0**	**1**
3		0	0	1	1		**1**	**1**	**1**	**1**	**0**	**0**	**1**
4		0	1	0	0		**0**	**1**	**1**	**0**	**0**	**1**	**1**
5		0	1	0	1		**1**	**0**	**1**	**1**	**0**	**1**	**1**
6		0	1	1	0		**1**	**0**	**1**	**1**	**1**	**1**	**1**
7		0	1	1	1		**1**	**1**	**1**	**0**	**0**	**0**	**0**
8		1	0	0	0		**1**	**1**	**1**	**1**	**1**	**1**	**1**
9		1	0	0	1		**1**	**1**	**1**	**1**	**0**	**1**	**1**
10		1	0	1	0		X	X	X	X	X	X	X
11		1	0	1	1		X	X	X	X	X	X	X
12		1	1	0	0		X	X	X	X	X	X	X
13		1	1	0	1		X	X	X	X	X	X	X
14		1	1	1	0		X	X	X	X	X	X	X
15		1	1	1	1		X	X	X	X	X	X	X

For ‘a’ →

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
	1	0	1	1
	0	1	1	1
	×	×	×	×
	1	1	×	×

For ‘b’ →

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	1	1
$\bar{A}B$	1	0	1	0
AB	×	×	×	×
$A\bar{B}$	1	1	×	×

For ‘c’ →

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	1	0
$\bar{A}B$	1	1	1	1
AB	×	×	×	×
$A\bar{B}$	1	1	×	×

$a = A + C + BD + \bar{B}\bar{D}$

$b = \bar{B} + \bar{C}\bar{D} + CD$

$c = B + \bar{C} + D$

For 'd' $\rightarrow$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
	1	0	1	1
	0	1	0	1
	×	×	×	×
	1	1	×	×

For 'e' $\rightarrow$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	0	0	0	1
AB	×	×	×	×
$A\bar{B}$	1	0	×	×

$\therefore\ c = B + \bar{C} + D$

For 'f' $\rightarrow$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	0
$\bar{A}B$	1	1	0	1
AB	×	×	×	×
$A\bar{B}$	1	1	×	×

$$d = \bar{B}\bar{D} + \bar{B}C + B\bar{C}D + A$$

$$e = \bar{B}\bar{D} + C\bar{D}$$

$$f = A + \bar{C}\bar{D} + B\bar{C} + B\bar{D}$$

For ‘g’ →

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	1	1
$\bar{A}B$	1	1	0	1
AB	×	×	×	×
$A\bar{B}$	1	1	×	×

$$g = A + B\bar{C} + \bar{B}C + C\bar{D}$$