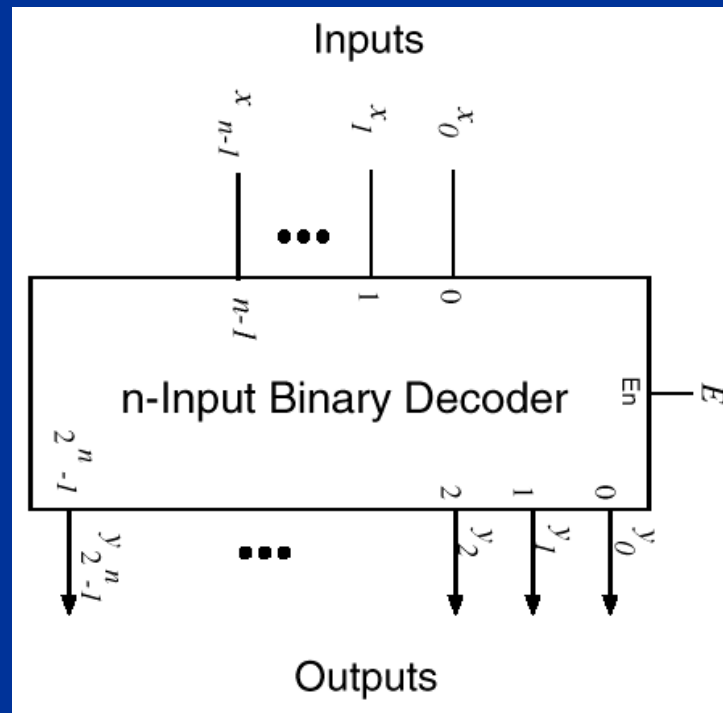# Decoders

Decoders

- Digital information represented in some binary form must be converted into some alternate binary form

- A Decoder is a combinational circuit that converts binary information from n coded inputs to a maximum 2^n coded outputs $\rightarrow$ n-to- 2^n-line decoder

- n-to-m decoder, $m \leq 2^n$

- Only one of the 2^n output lines responds to a given input combination of values on its n-input lines
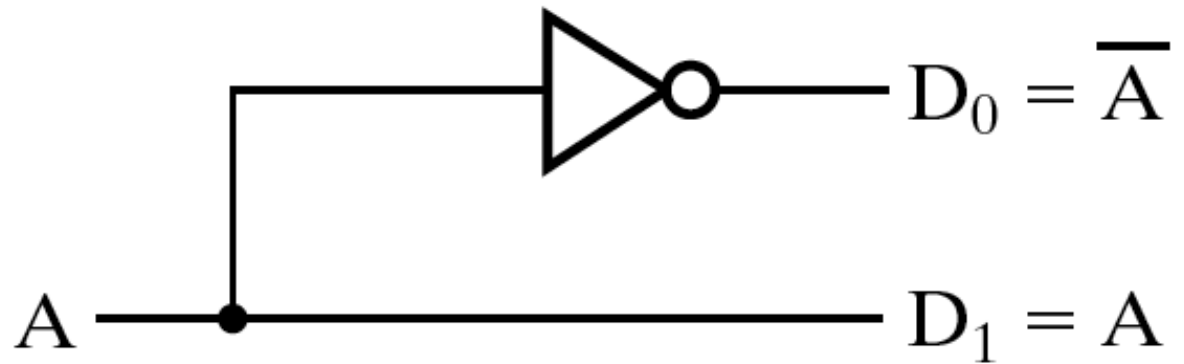
Decoders (cont.)

- Examples:
  - *2-to-4 decoder, where n=2 and m=4*
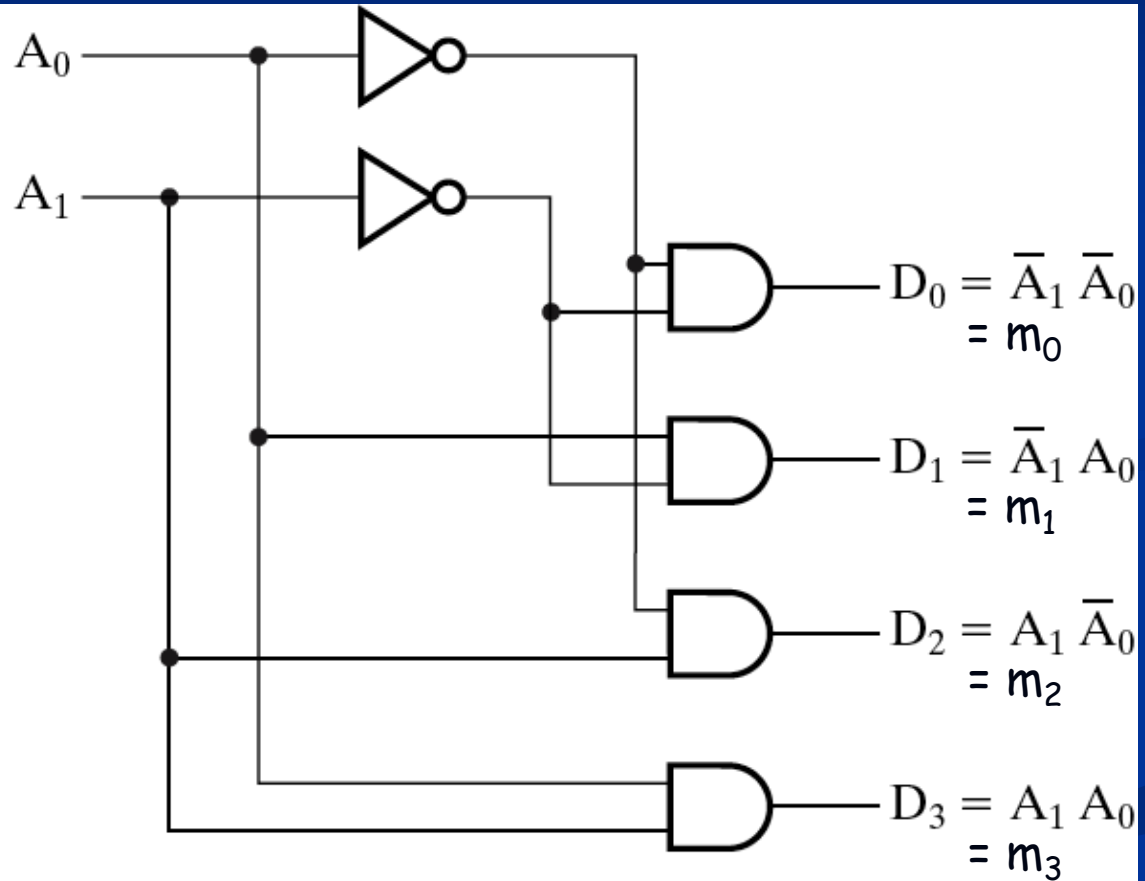 - BCD-to-7-segment decoder, where *n=4* and *m=10*

1-to-2 Decoder

A	D_0	D_1
0	1	0
1	0	1

(a)

$D_0 = \overline{A}$

$D_1 = A$

(b)

2-to-4 Decoder using AND (Active High Decoder)

A_1	A_0	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

(a)

A_0

A_1

$D_0 = \overline{A}_1 \overline{A}_0 = m_0$

$D_1 = \overline{A}_1 A_0 = m_1$

$D_2 = A_1 \overline{A}_0 = m_2$

$D_3 = A_1 A_0 = m_3$

(b)

2-to-4 Decoder using NAND (Active Low Decoder)

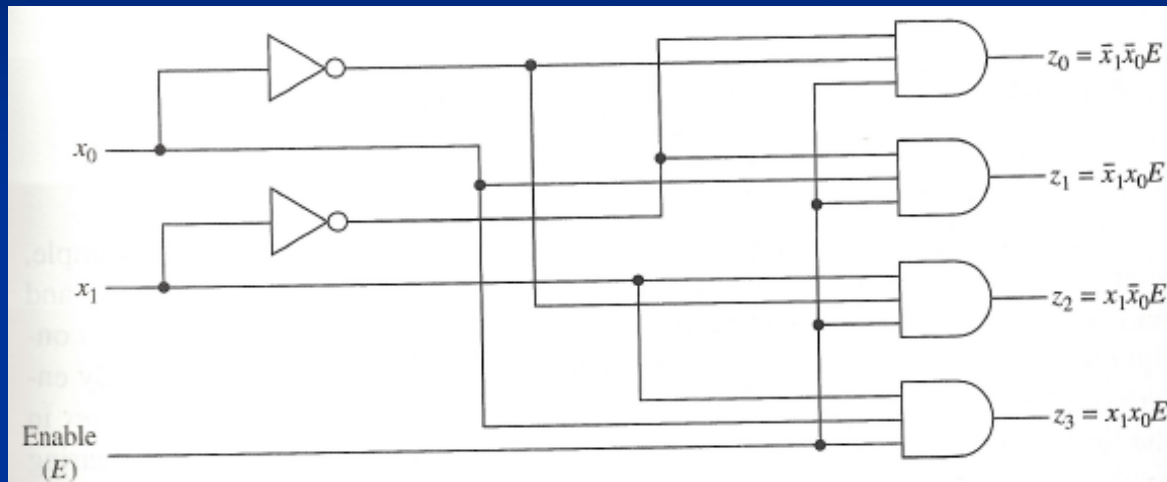| a	b	D_0	D_1	D_2	D_3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

$D_0 = (a'b')' = a + b = M_0$
$D_1 = (a'b)' = a + b' = M_1$
$D_2 = (ab')' = a' + b = M_2$
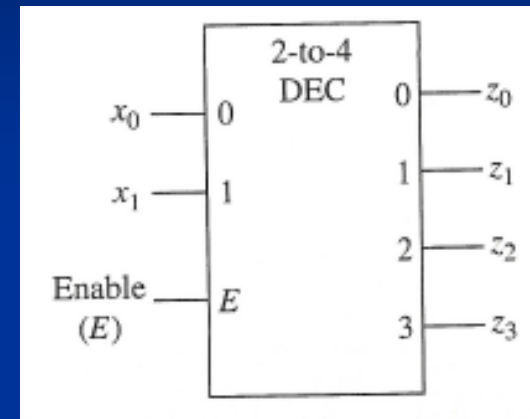$D_3 = (ab)' = a' + b' = M_3$
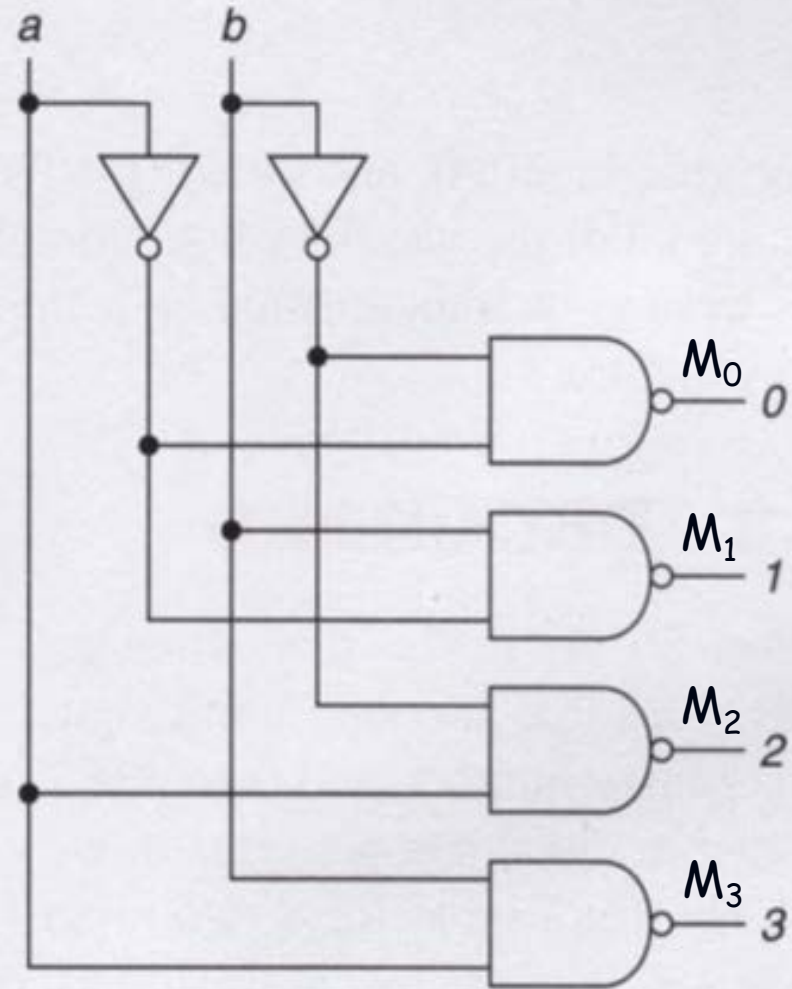
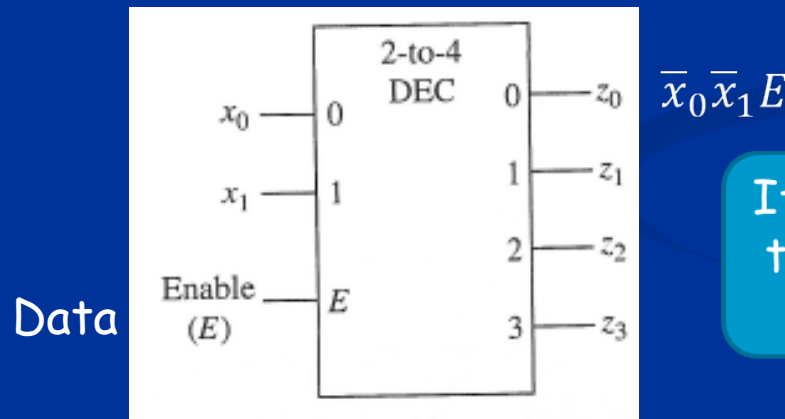# 2-to-4 Decoder using AND with Enable Input

Logic Diagram
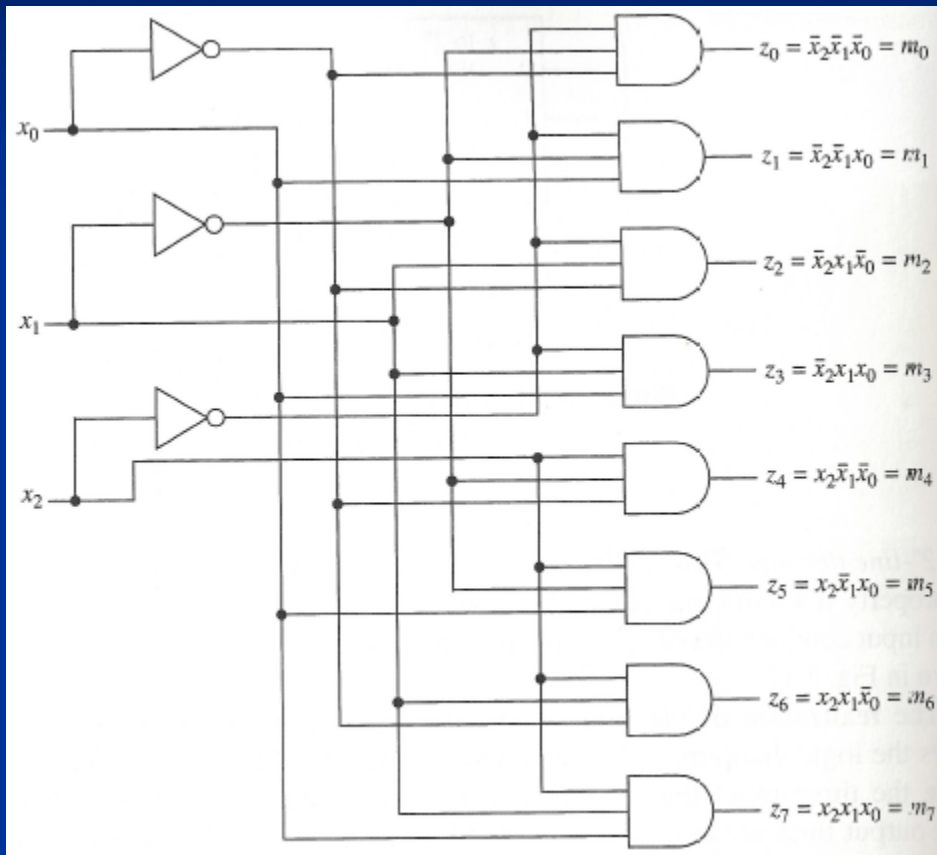

Symbol

Truth Table

Decoders with Enable Inputs

- When disabled, all outputs of the decoder can either be at logic-0 or logic-1

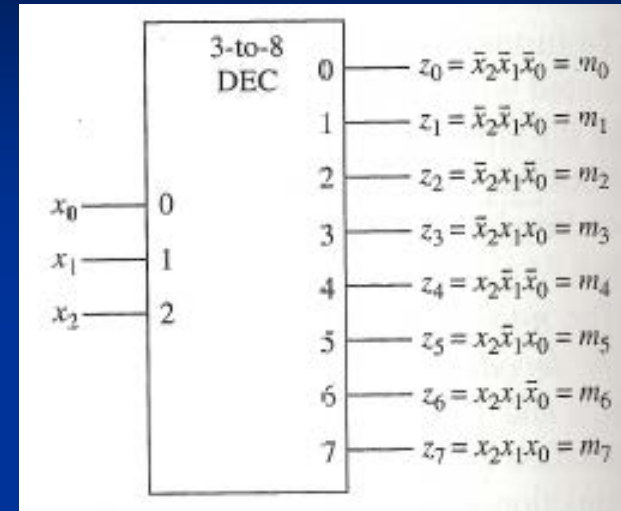- Enable input provides the decoder with additional flexibility

$\bar{x}_0 \bar{x}_1 E$

If $E = 1, x_0 = 0, x_1 = 0$ then data appears on line z_0.

- Enable inputs are useful when constructing larger decoders from smaller decoders
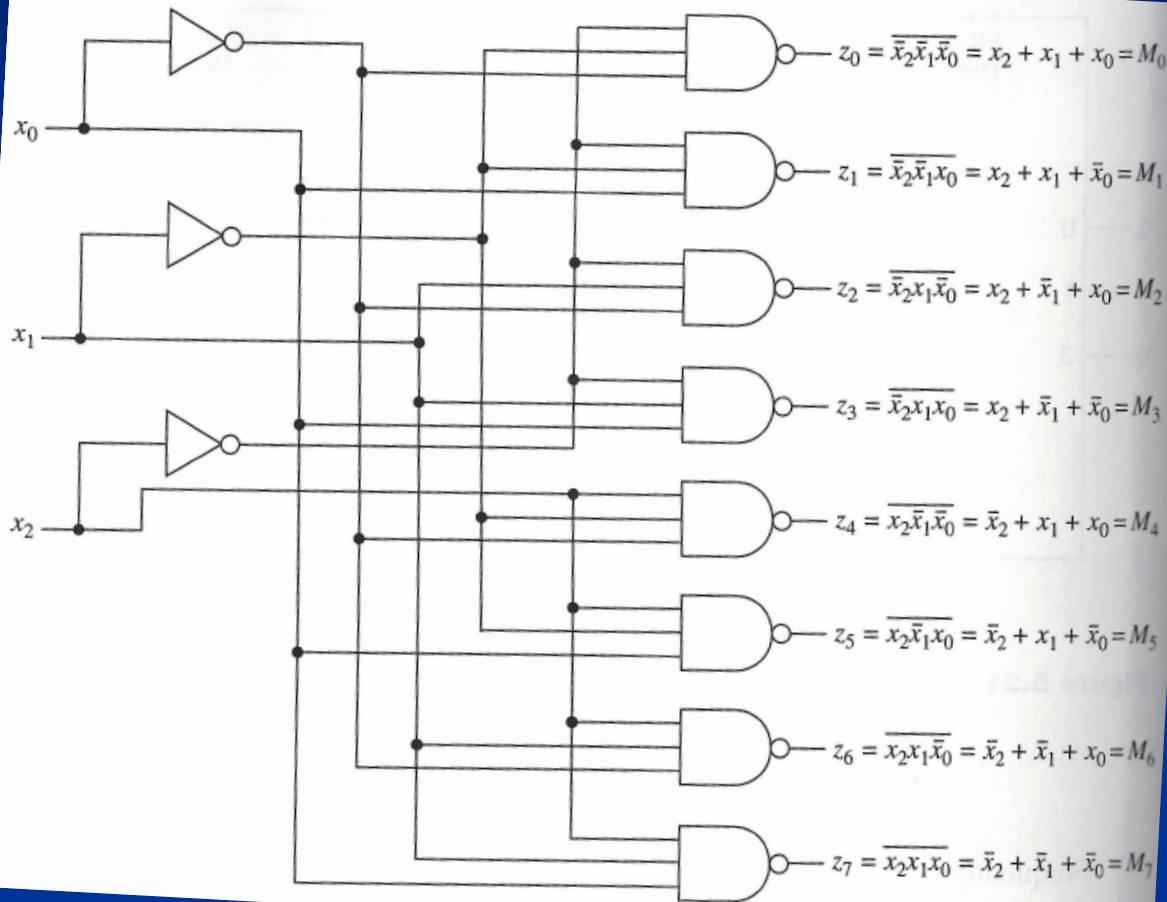
3-to-8 Decoder using AND (cont.)

Logic Diagram

Symbol

Inputs			Outputs							
x_2	x_1	x_0	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Truth Table

3-to-8 Decoder using NAND (cont.)

Logic Diagram

Symbol

Truth Table

Implementing Boolean Functions using Decoders

- **<u>Any</u> combinational circuit can be constructed using decoders! Why?**
  - An n-to-2^n line decoder is a minterm/maxterm generator
  - By using or/nor/and-gates in conjunction with an n-to-2^n line decoder, realizations of Boolean functions are possible
- **Here is an example:**
  - Implement a full adder circuit with a decoder
  - Recall full adder equations, and let X, Y, and Z be the inputs:
    - $S(X,Y,Z) = \Sigma m(1,2,4,7)$
    - $C(X,Y,Z) = \Sigma m(3, 5, 6, 7)$
  - Since there are 3 inputs and a total of 8 minterms, we need a 3-to-8 decoder

Implementing a Full Adder Using a Decoder

$$S(X,Y,Z) = \Sigma_m(1,2,4,7)$$

$$C(X,Y,Z) = \Sigma_m(3,5,6,7)$$
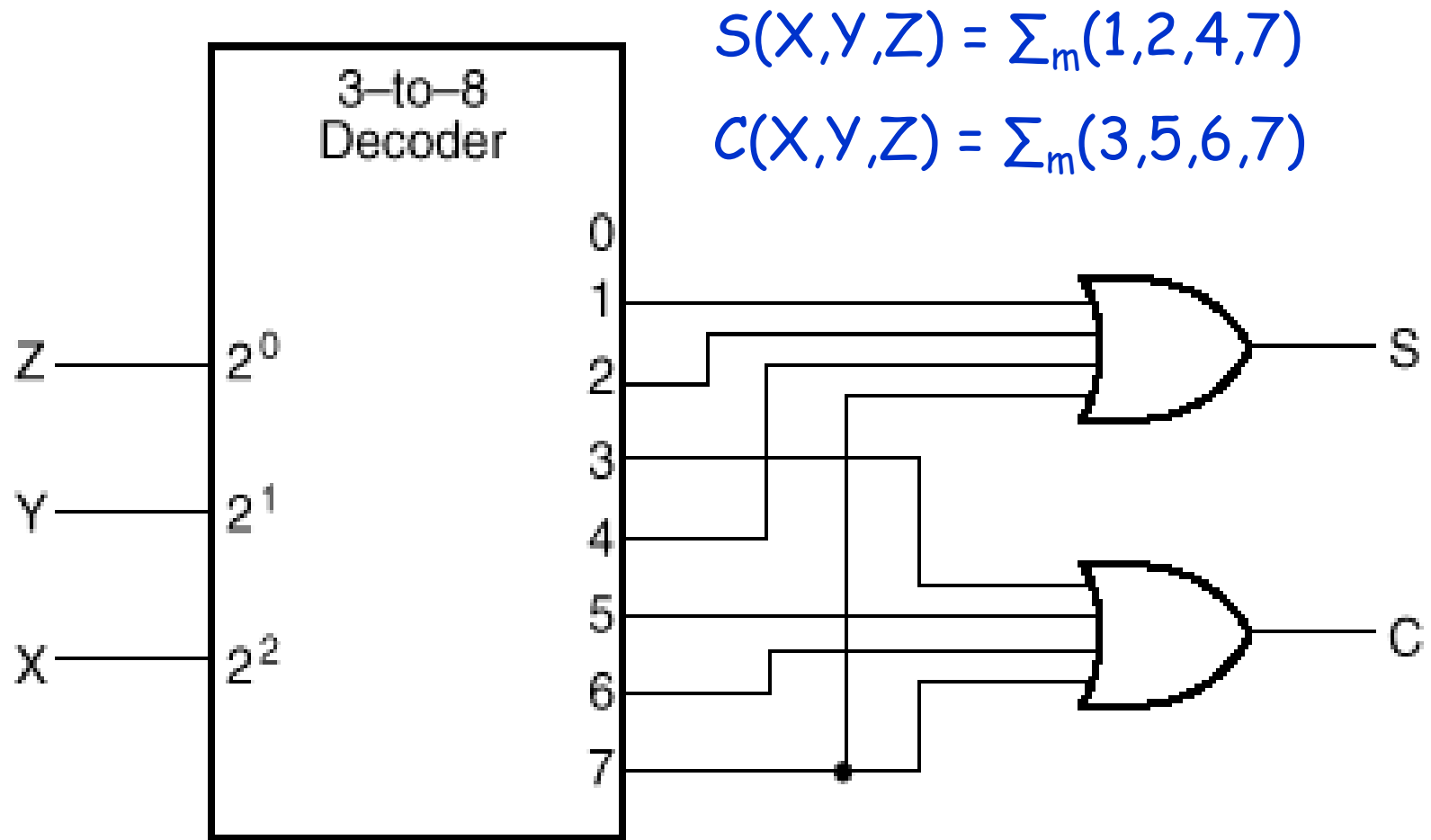
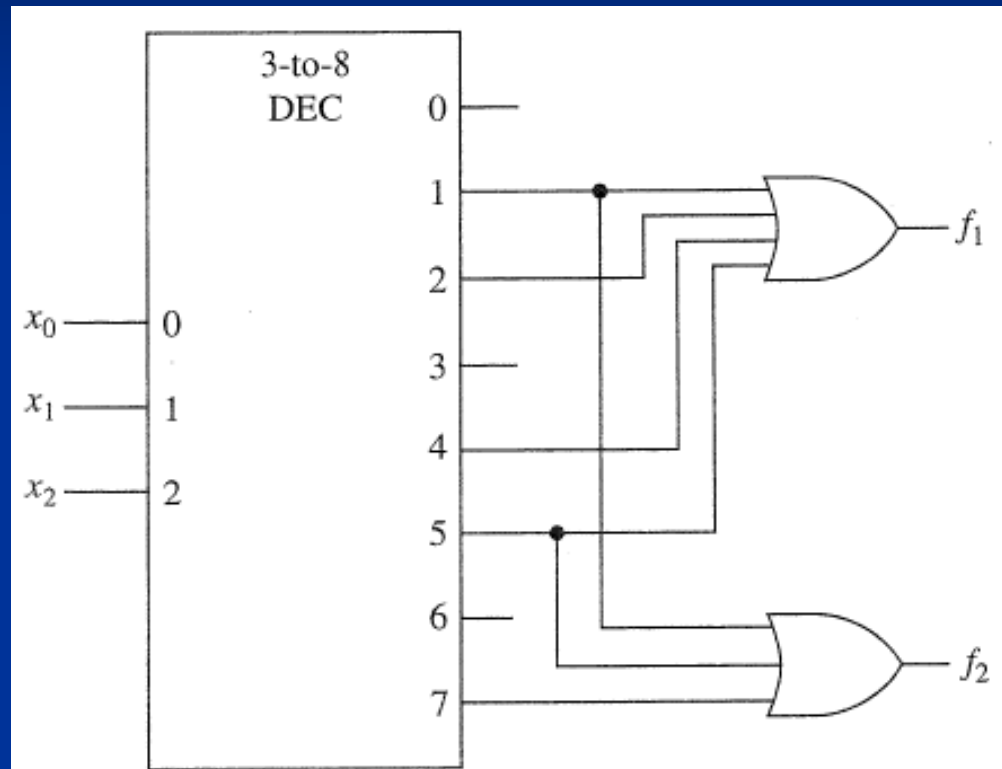# Implementing Boolean Functions using Decoders (cont.)

Figure 5.19 Realization of the Boolean expressions
$f_1(x_2, x_1, x_0) = \Sigma m(1,2,4,5)$ and
$f_2(x_2, x_1, x_0) = \Sigma m(1,5,7)$ with a 3-to-8-

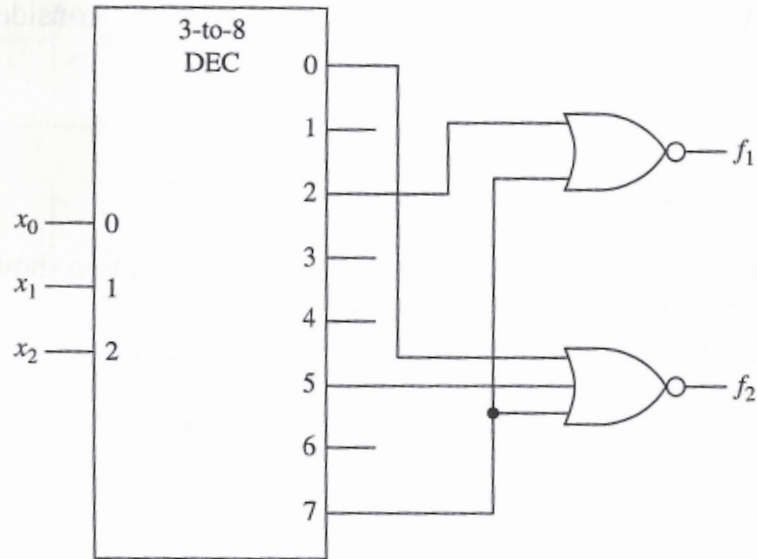# Implementing Boolean Functions using Decoders (cont.)

Figure 5.20 Realization of the Boolean expressions $f_1(x_2,x_1,x_0) = \Sigma m(0,1,3,4,5,6) = \overline{\Sigma m(2,7)}$ and $f_2(x_2,x_1,x_0) = \Sigma m(1,2,3,4,6) = \overline{\Sigma m(0,5,7)}$ with a 3-to-8-line decoder and two nor-gates.
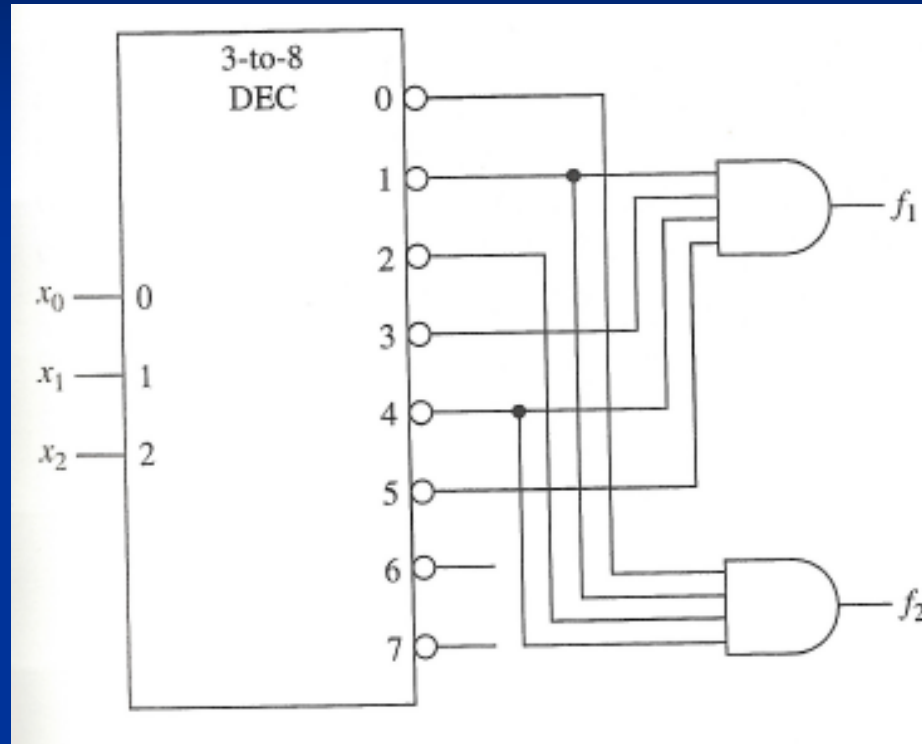
$f_1(x_2,x_1,x_0) = \Sigma_m (0,1,3,4,5,6)$
$f_1'(x_2,x_1,x_0) = \Sigma_m (2,7) \rightarrow f_1(x_2,x_1,x_0) = \overline{\Sigma_m (2,7)}$

$f_2(x_2,x_1,x_0) = \Sigma_m (1,2,3,4,6)$
$f_2'(x_2,x_1,x_0) = \Sigma_m (0,5,7) \rightarrow f_2(x_2,x_1,x_0) = \overline{\Sigma_m (0,5,7)}$

Implementing Boolean Functions using Decoders (cont.)

$$f_1 = \sum_m (0,2,6,7) = \Pi_M(1,3,4,5)$$
$$f_2 = \sum m(3,5,6,7) = \Pi_M(0,1,2,4)$$

Constructing Larger Decoders

- Enable inputs are useful when constructing larger decoders from smaller decoders
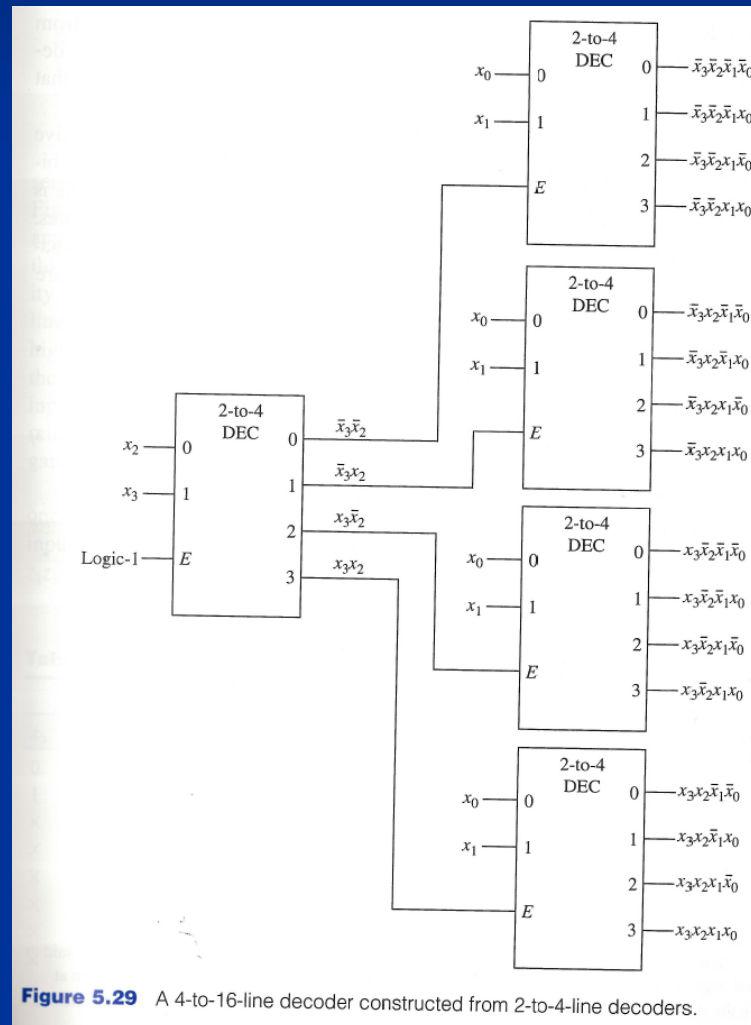

Figure 5.29 A 4-to-16-line decoder constructed from 2-to-4-line decoders.