Digital Logic Design
ENEE 244-020x

Lecture 13

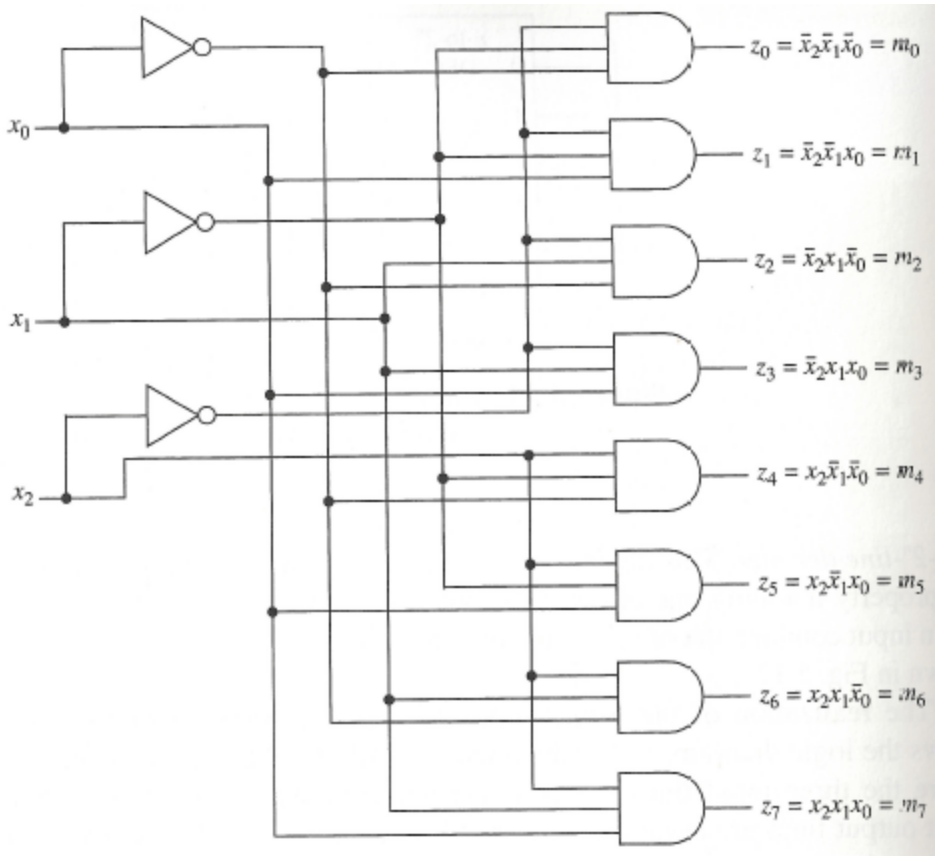
Announcements

- Homework 4 due ~~Thursday 10/29~~ Sunday 11/1 @11:59pm
  - Check updates on course webpage + canvas/ELMS

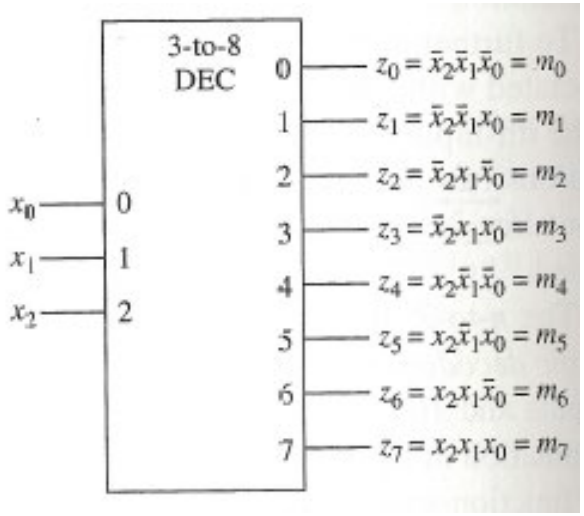
Agenda

- Last time:
  - Decimal Adders (4.6)
  - Comparators (4.8)
  - Decoders (4.9)
  - Encoders (4.10)
  - Multiplexers (4.11)
- This time:
  - Logic Design with Decoders and Multiplexers (4.9, 4.11)

Logic Design Using Decoders



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

Logic Design Using Decoders

- An n-to-2^n line decoder is a minterm generator.
- By using or-gates in conjunction with an n-to-2^n line decoder, realizations of Boolean functions are possible.
- Do not correspond to minimal sum-of-products.
- Are simple to produce. Particularly convenient when several functions of the same variable have to be realized.

correspond to canonical

Minterms using OR Gates

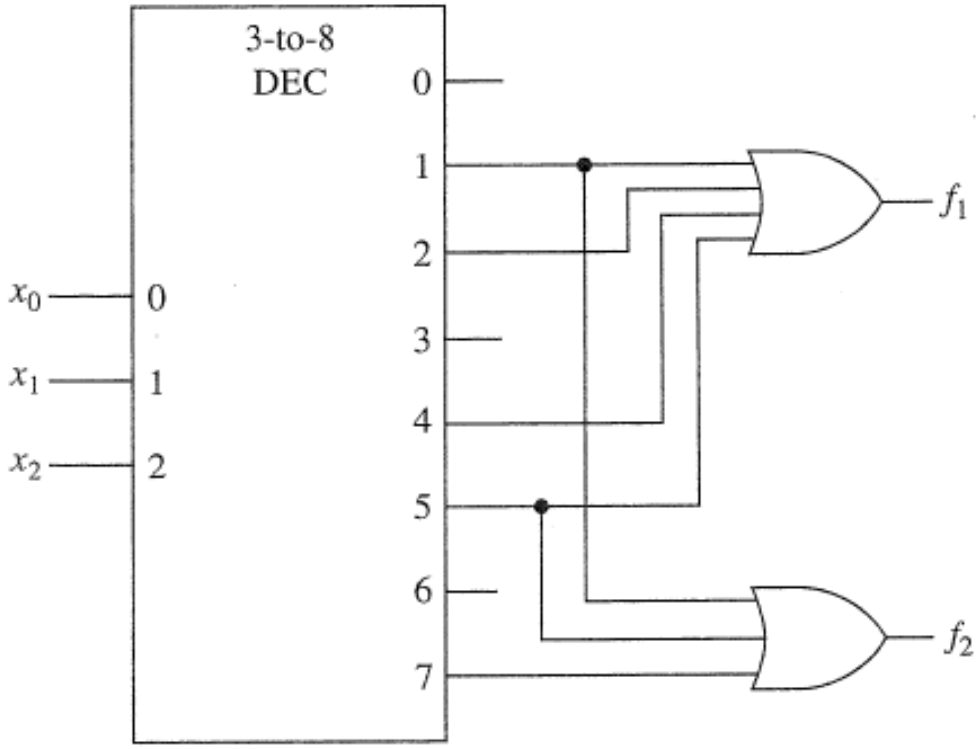


Figure 5.19 Realization of the Boolean expressions $f_1(x_2,x_1,x_0) = \Sigma m(1,2,4,5)$ and $f_2(x_2,x_1,x_0) = \Sigma m(1,5,7)$ with a 3-to-8-

Minterms using NOR Gates

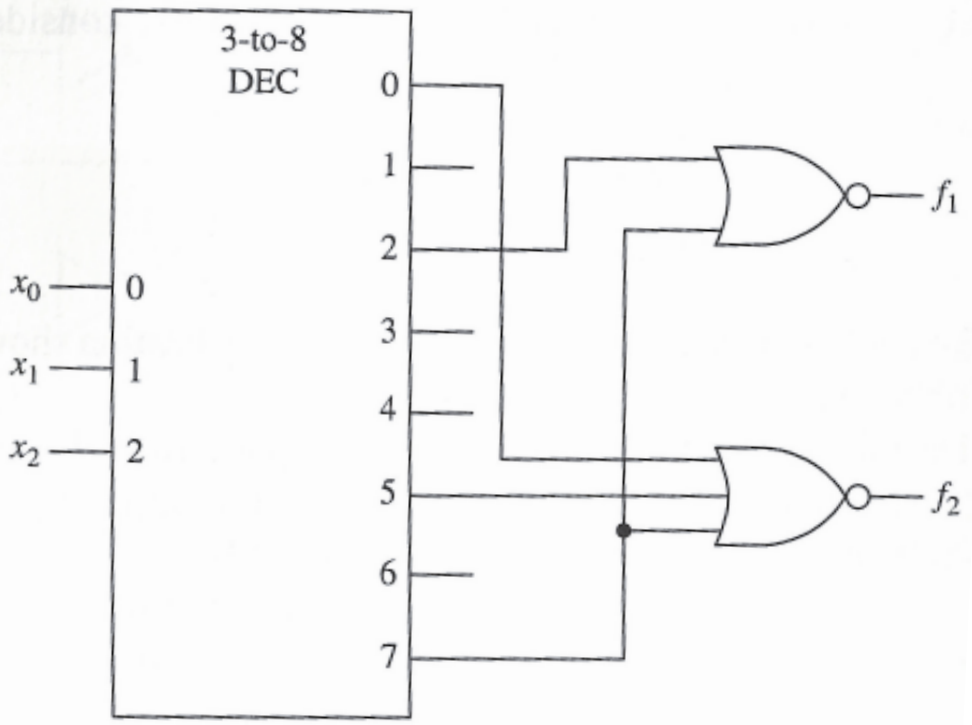
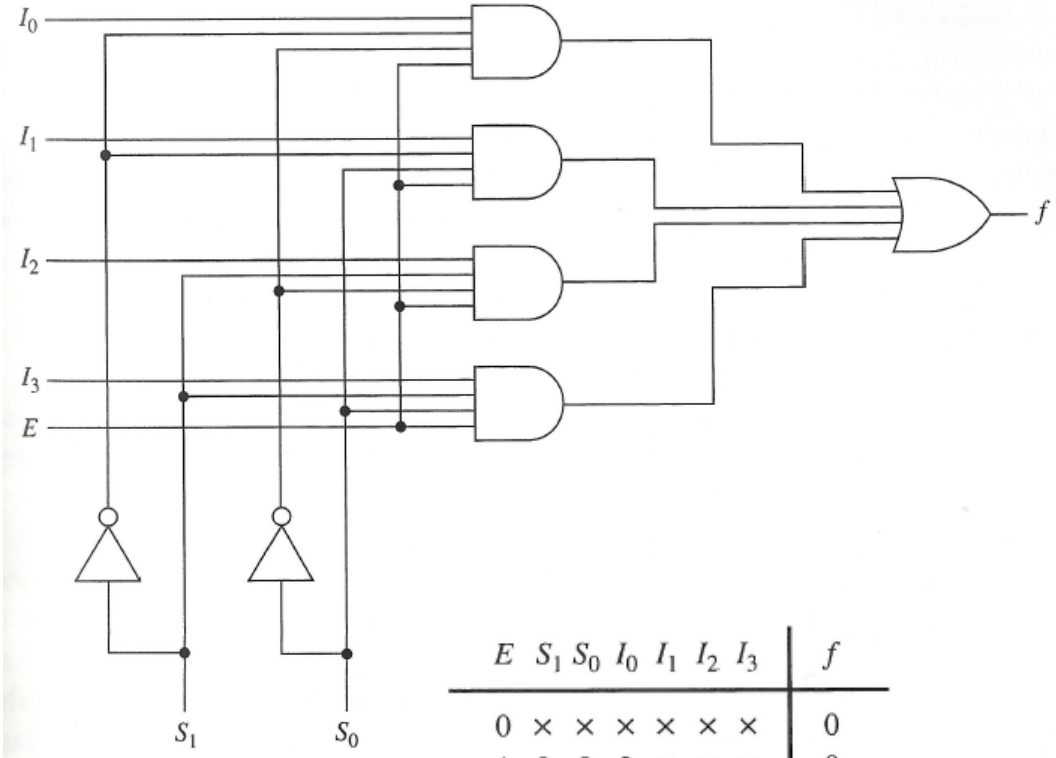
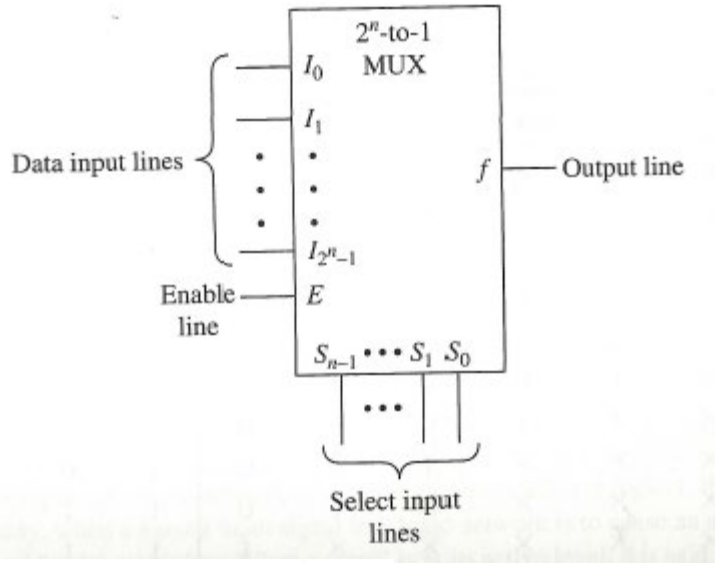


Figure 5.20 Realization of the Boolean expressions $f_1(x_2,x_1,x_0) = \Sigma m(0,1,3,4,5,6) = \overline{\Sigma m(2,7)}$ and $f_2(x_2,x_1,x_0) = \Sigma m(1,2,3,4,6) = \overline{\Sigma m(0,5,7)}$ with a 3-to-8-line decoder and two nor-gates.

Logic Design with Multiplexers

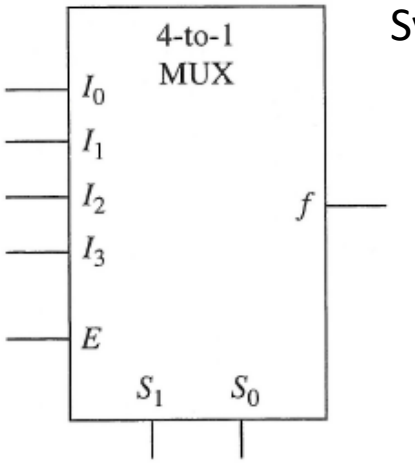


Logic Diagram



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

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:

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

Logic Design with Multiplexers

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

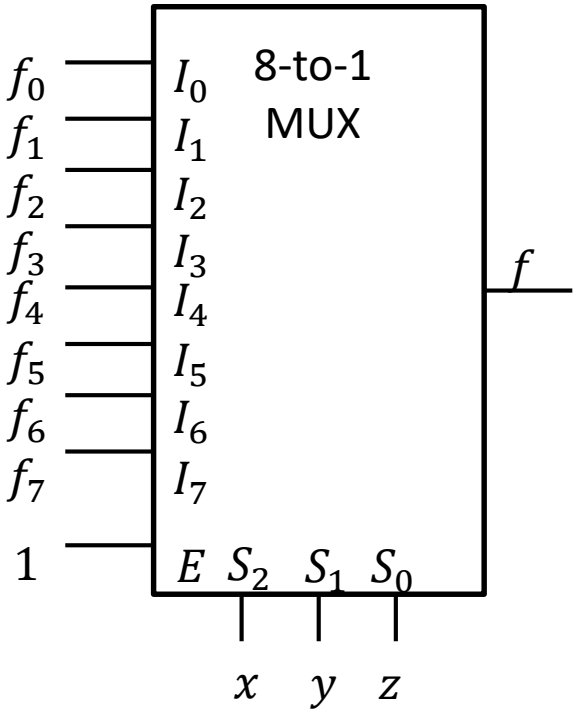
$$f(x, y, z) = f_0 \cdot x'y'z' + f_1 \cdot x'y'z + f_2 \cdot x'yz' + f_3 \cdot x'yz + f_4 \cdot xy'z' + f_5 \cdot xy'z + f_6 xyz' + f_7 \cdot xyz.$$

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

$$f = (I_0 S'_2 S'_1 S'_0 + I_1 S'_2 S'_1 S_0 + I_2 S'_2 S_1 S'_0 + I_3 S'_2 S_1 S_0$$

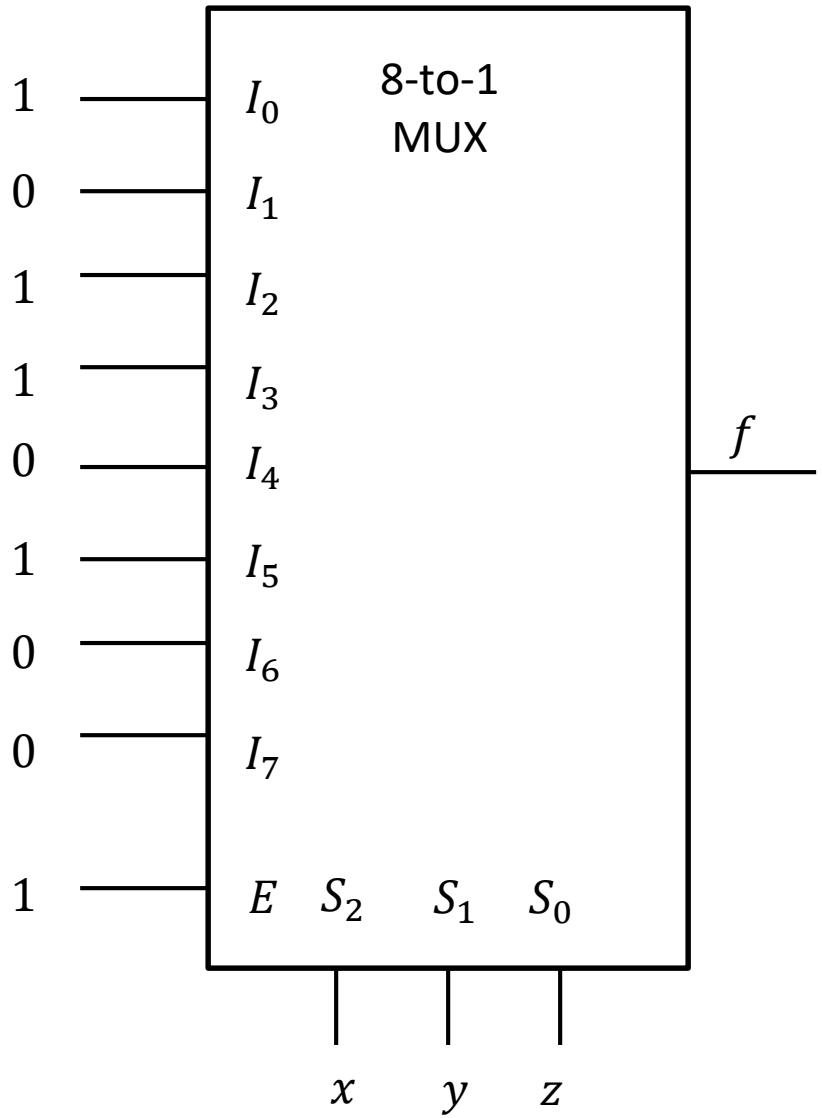
Logic Design with Multiplexers

- 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.
- $f(x, y, z) = f_0 \cdot x'y'z' + f_1 \cdot x'y'z + f_2 \cdot x'yz' + f_3 \cdot x'yz + f_4 \cdot xy'z' + f_5 \cdot xy'z + f_6xyz' + f_7 \cdot xyz$

$$= (f_0 \cdot z' + f_1 \cdot z)x'y' + (f_2 \cdot z' + f_3 \cdot z)x'y + (f_4 \cdot z' + f_5 \cdot z)xy' + (f_6 \cdot z' + f_7 \cdot z)xy$$

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

$$I_0S'_1S'_0 + I_1S'_1S_0 + I_2S_1S'_0 + I_3S_1S_2$$

Logic Design with Multiplexers

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

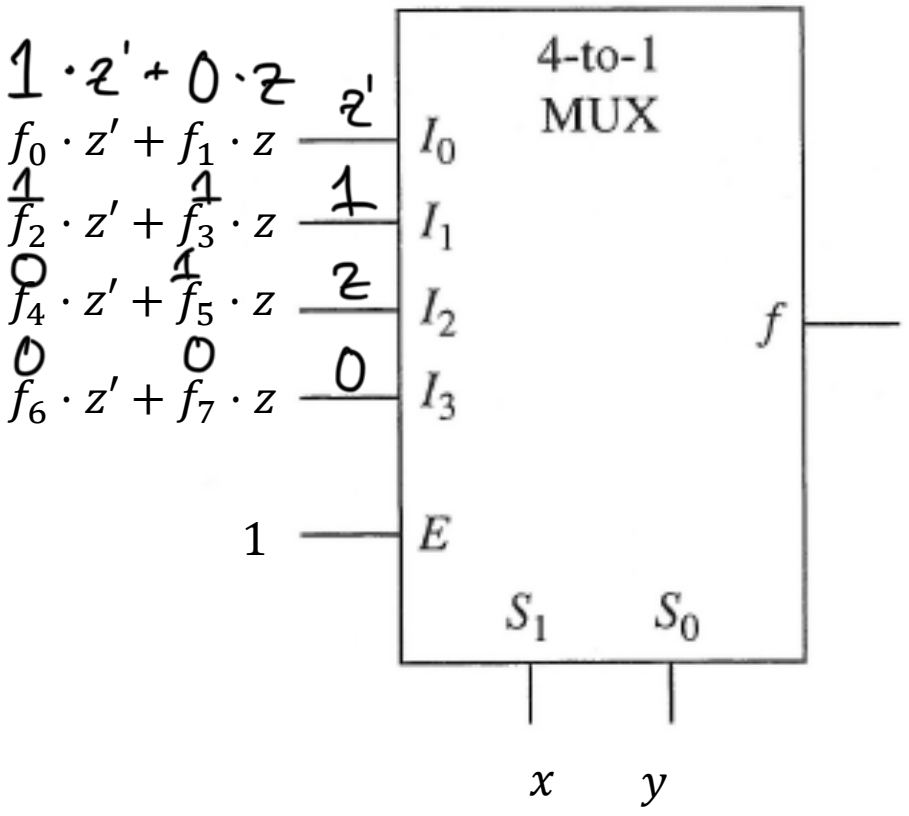
4-to-1 Multiplexer has the form

$$f = I_0 S'_1 S'_0 + I_1 S'_1 S_0 + I_2 S_1 S'_0 + I_3 S_1 S_2$$

- 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 $f_i \cdot z' + f_j \cdot z$ on the data input lines and let E = 1.
- Note: $f_i \cdot z' + f_j \cdot z$ reduce to 0,1,z or z'.

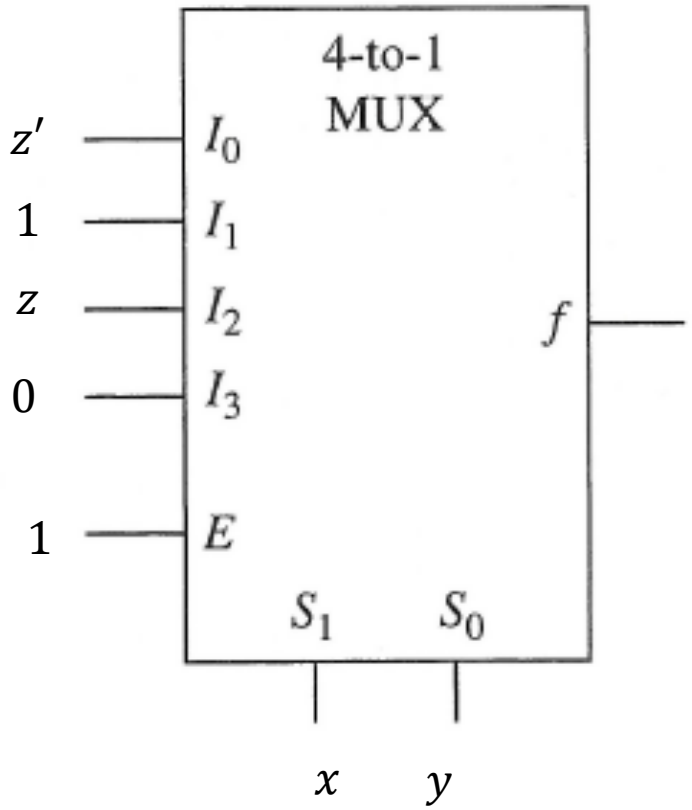
Example

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



Example

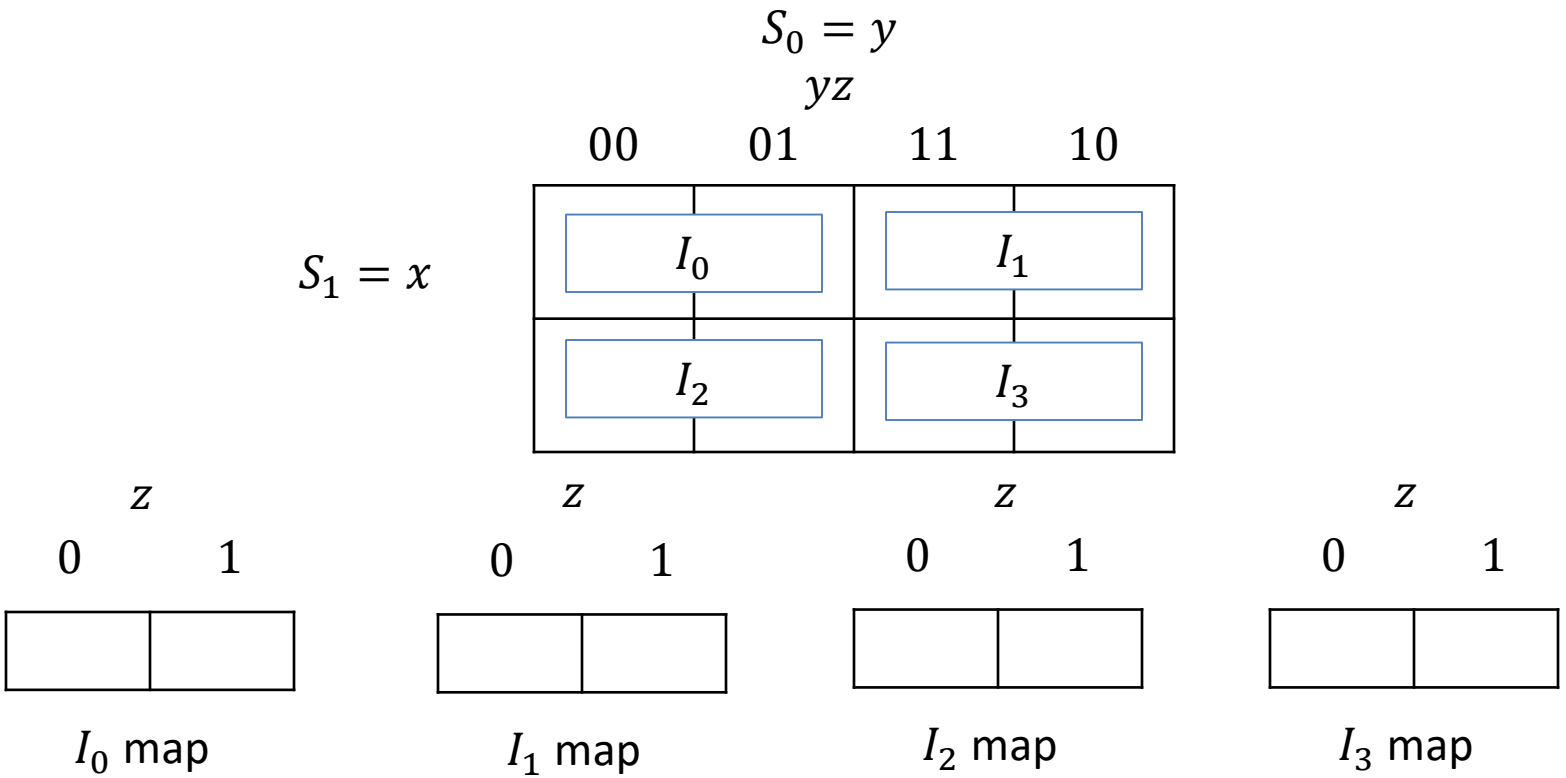
x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Logic Design with Multiplexers and K-maps

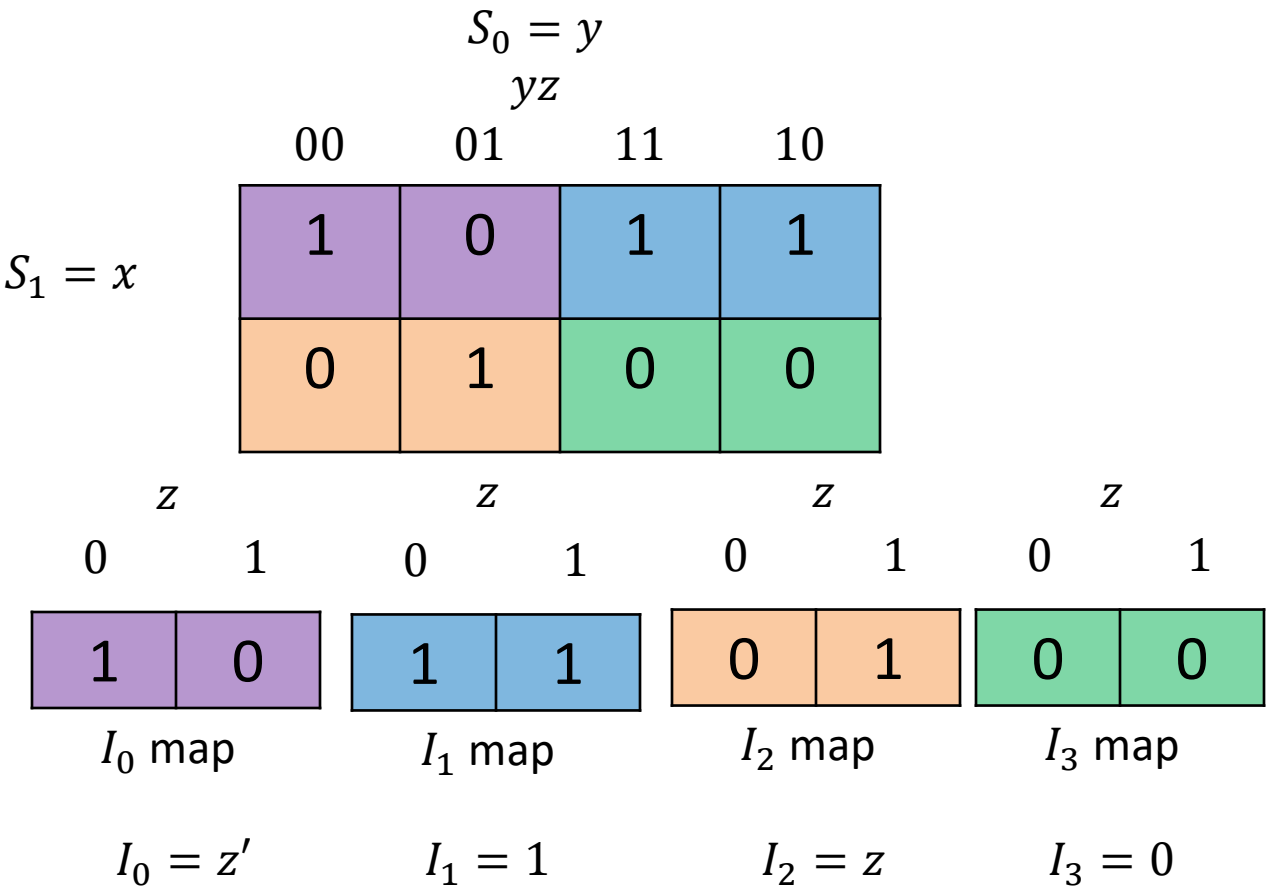
- Consider 3-variable Karnaugh map. Assume x is placed on the S_1 line and y is placed on the S_0 line.
- We get that the output is: $I_0x'y' + I_1x'y + I_2xy' + I_3xy$
- $I_0x'y'$ corresponds to those cells in which $x = 0, y = 0$
- $I_1x'y$ corresponds to those cells in which $x = 0, y = 1$
- I_2xy' 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



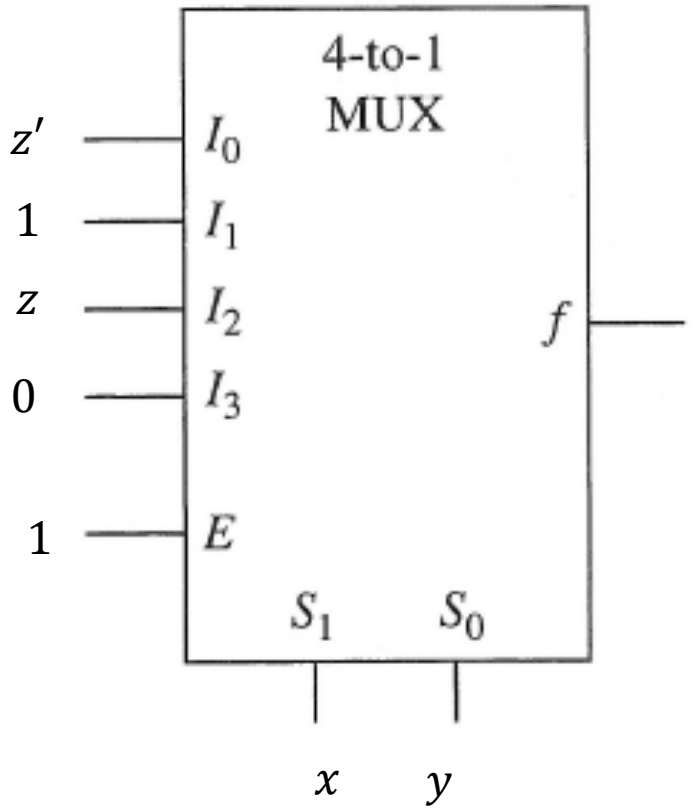
Example

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



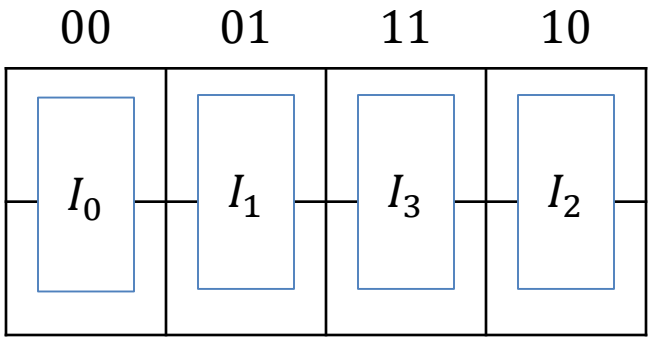
Realization

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

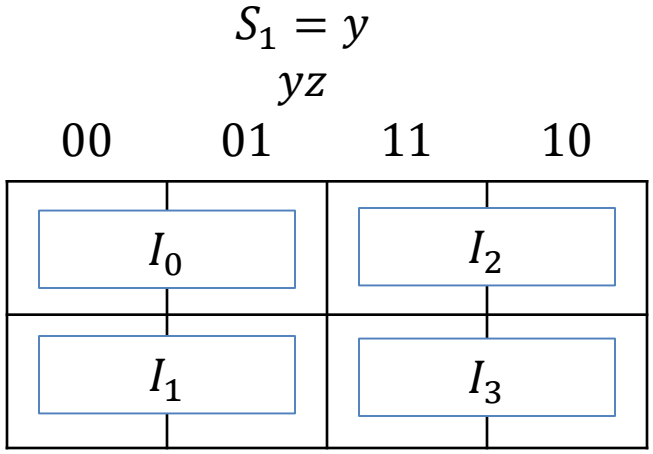


Alternative Structures

$S_1 = y, S_0 = z$



$S_1 = y$



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:

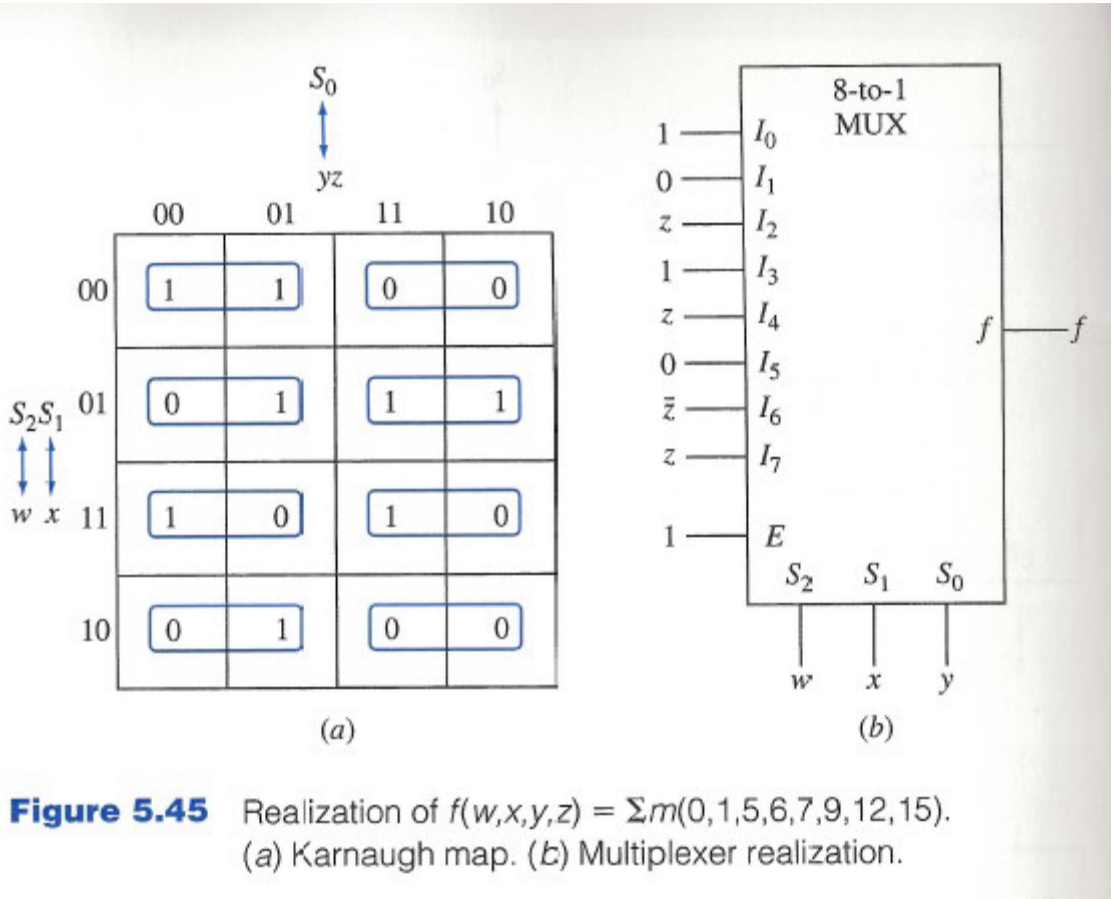


Figure 5.45 Realization of $f(w,x,y,z) = \Sigma m(0,1,5,6,7,9,12,15)$. (*a*) Karnaugh map. (*b*) Multiplexer realization.

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

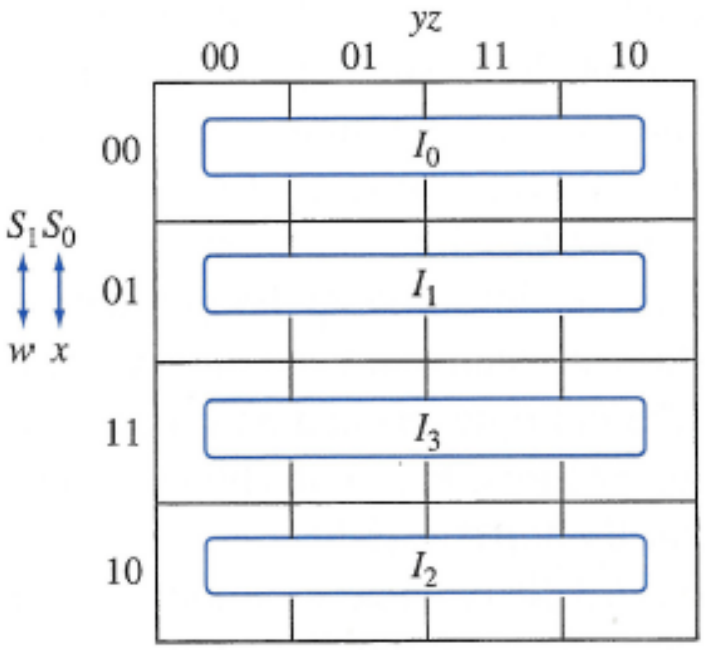
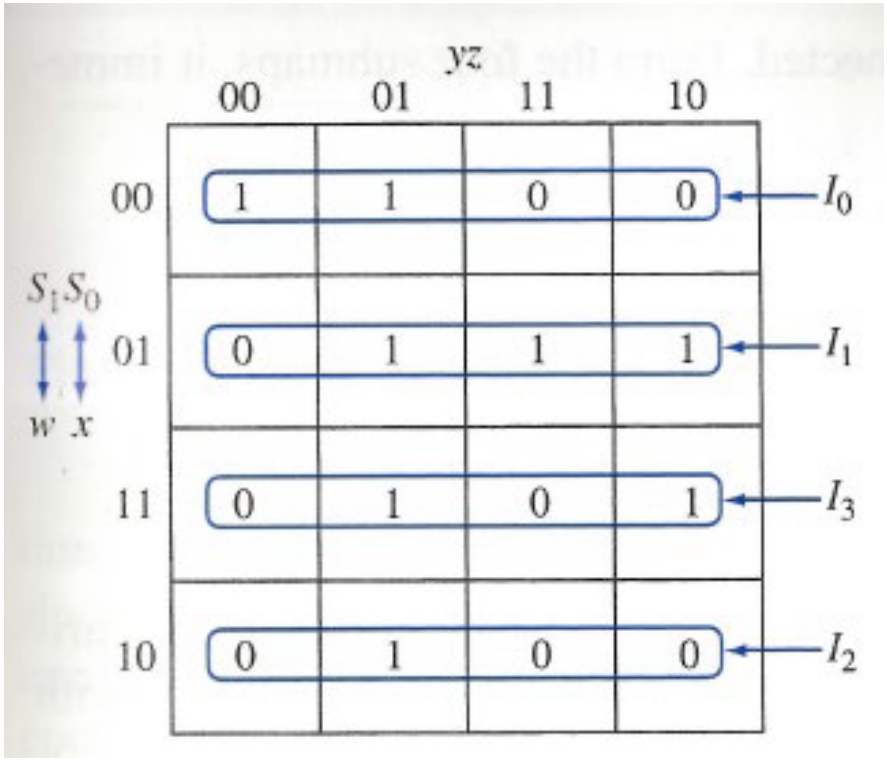
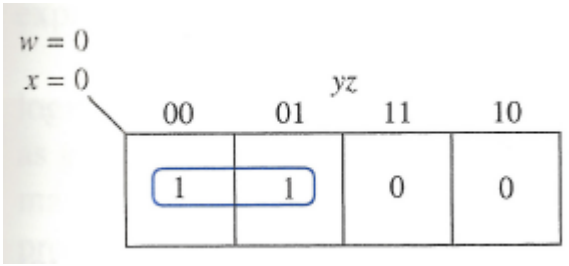


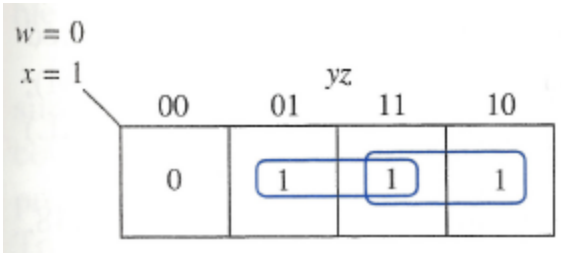
Figure 5.46 Using a four-variable Karnaugh map to obtain a Boolean function realization with a 4-to-1-line multiplexer.

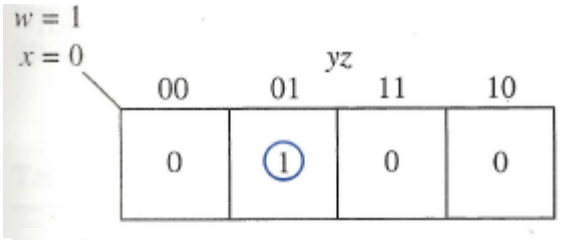
Example:

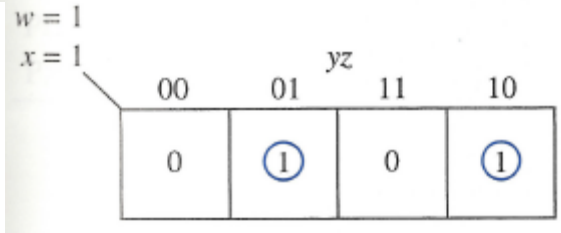
$f(w, x, y, z) = \sum m(0,1,5,6,7,9,13,14)$



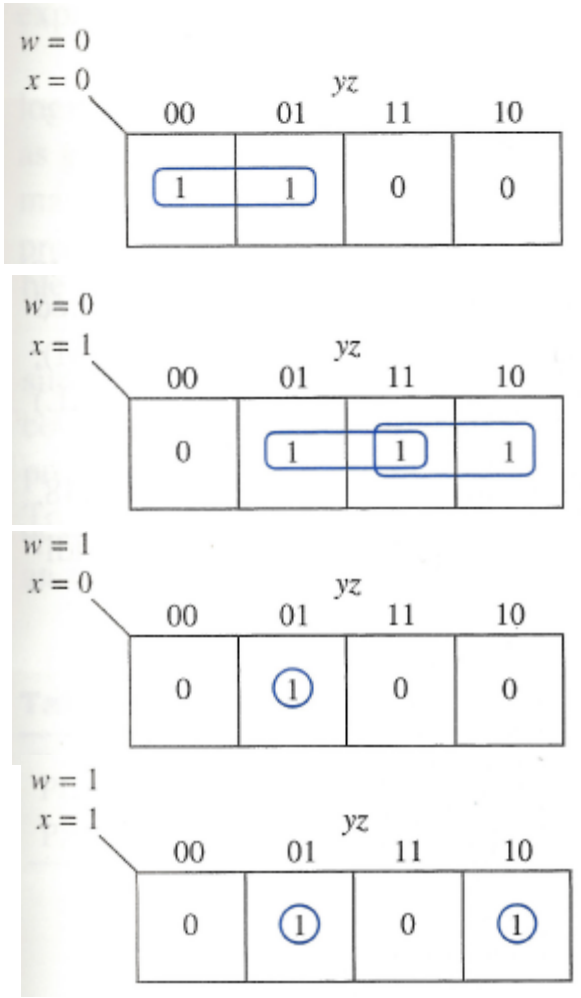


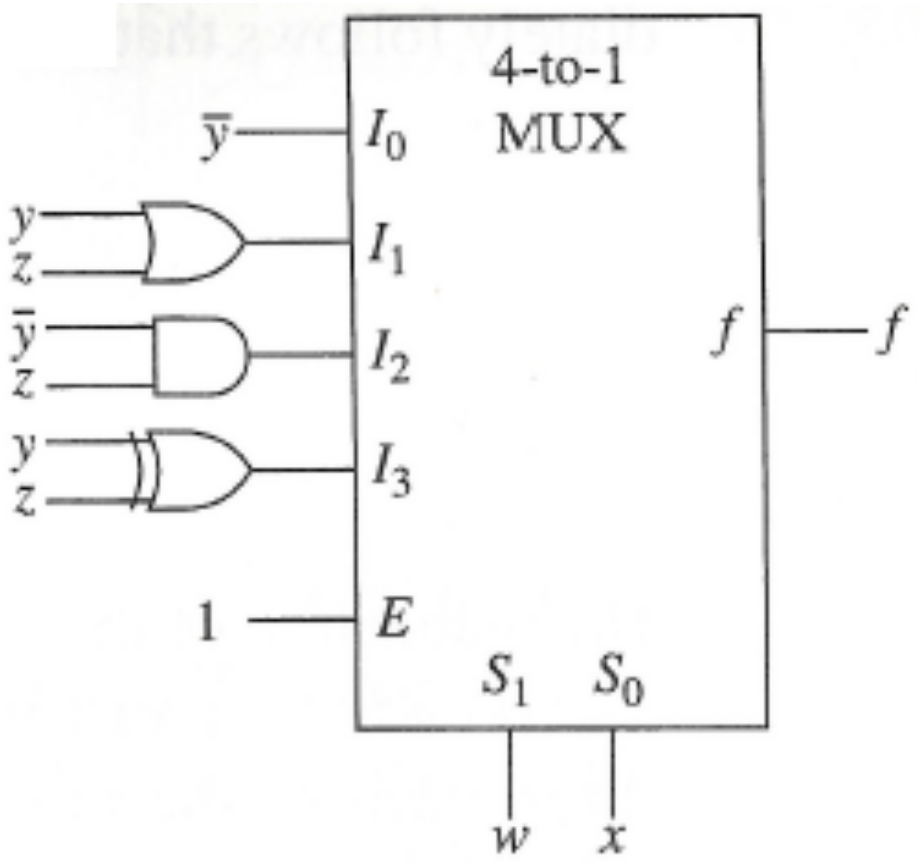




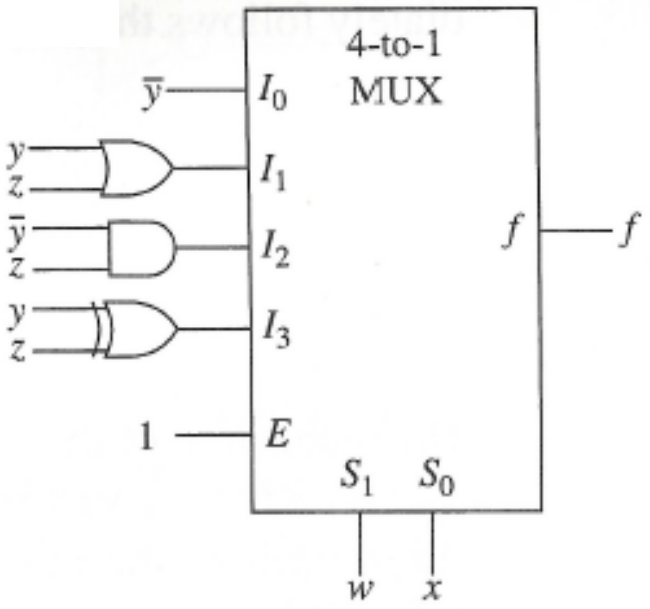


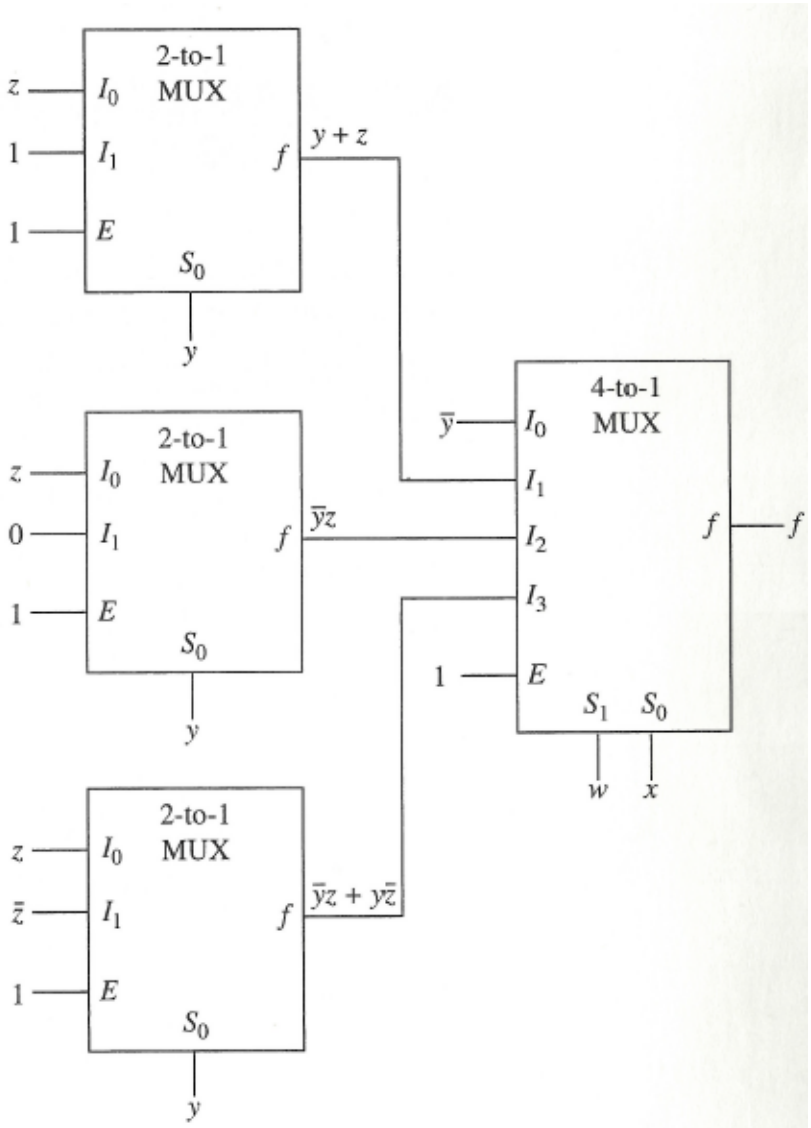
Example


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