



# **CprE 281: Digital Logic**

**Instructor: Alexander Stoytchev**

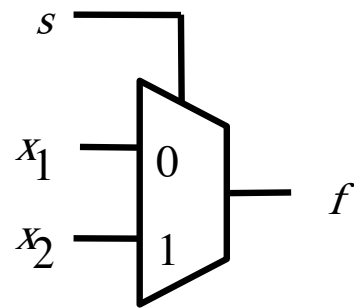
**<http://www.ece.iastate.edu/~alexs/classes/>**

# Decoders and Encoders

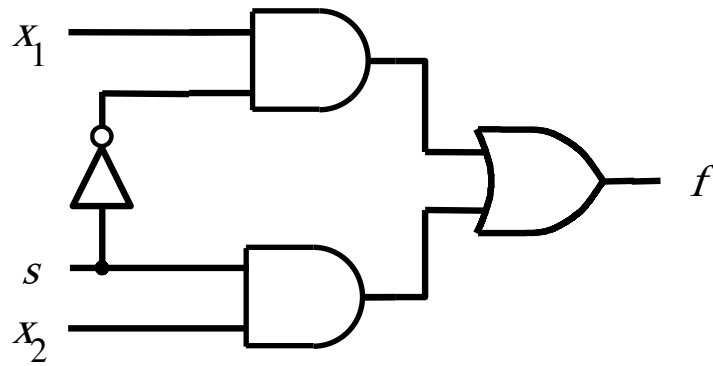
*CprE 281: Digital Logic*  
*Iowa State University, Ames, IA*  
*Copyright © Alexander Stoytchev*

# Quick Review

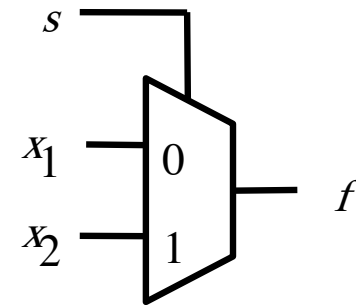
# Graphical Symbol for a 2-1 Multiplexer



# Circuit for 2-1 Multiplexer



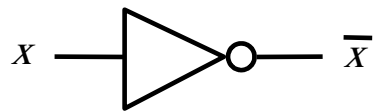
(b) Circuit



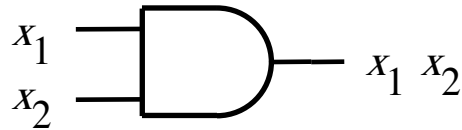
(c) Graphical symbol

$$f(s, x_1, x_2) = \bar{s} x_1 + s x_2$$

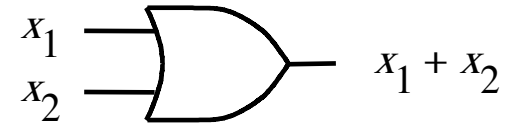
# The Three Basic Logic Gates



NOT gate

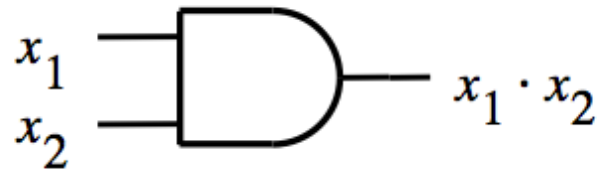


AND gate



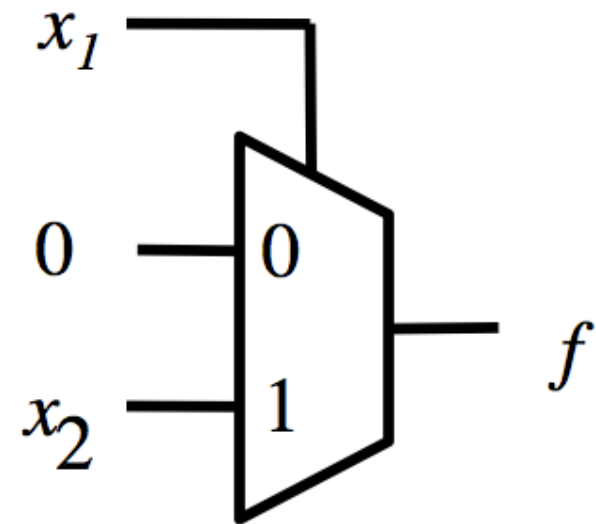
OR gate

# Building an AND Gate with 2-to-1 Mux

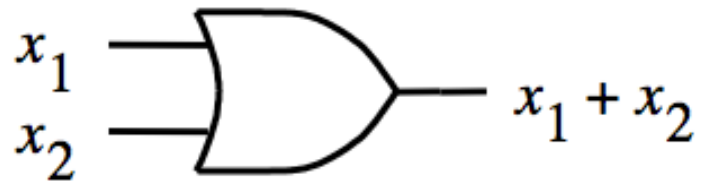


$x_1$	$x_2$	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Red annotations: A vertical red line is drawn between the  $x_1$  and  $x_2$  columns. A horizontal red line is drawn between the second and third rows. Red curly braces group the output values: the first two rows (0, 0) are grouped and labeled  $0$ ; the last two rows (0, 1) are grouped and labeled  $x_2$ .

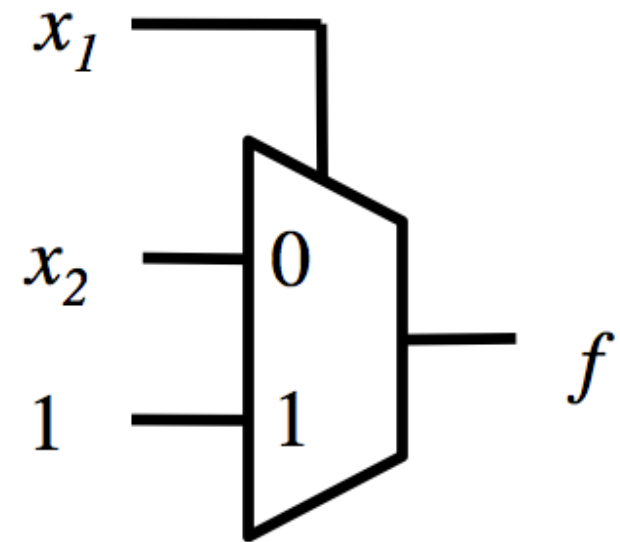


# Building an OR Gate with 2-to-1 Mux



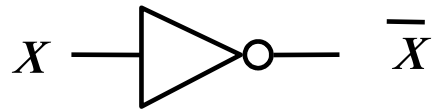
$x_1$	$x_2$	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

Red annotations: A vertical red line is between the  $x_1$  and  $x_2$  columns. A horizontal red line is between the second and third rows. Red curly braces on the right group the output values: the first two rows are grouped and labeled  $x_2$ , and the last two rows are grouped and labeled  $1$ .

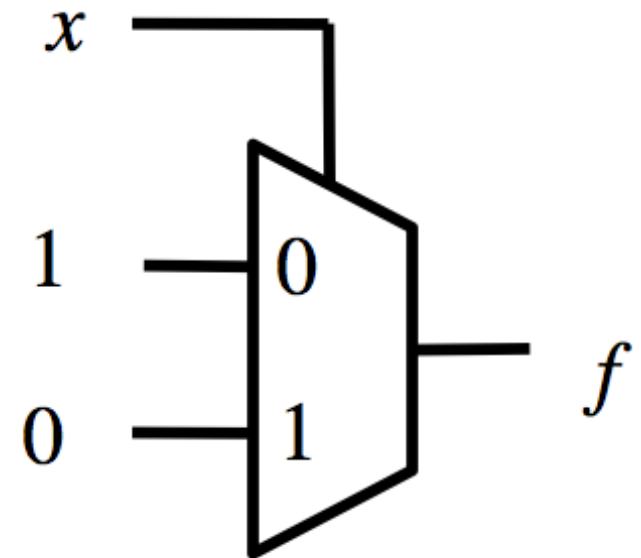




# Building a NOT Gate with 2-to-1 Mux



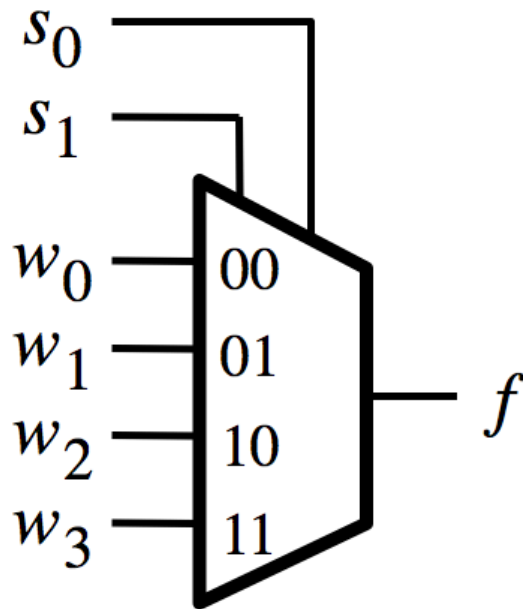
$x$	$\bar{x}$
0	1
1	0



# Implications

**Any Boolean function can be implemented  
using only 2-to-1 multiplexers!**

# 4-to-1 Multiplexer: Graphical Symbol and Truth Table

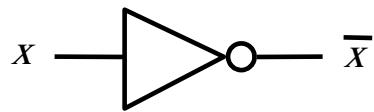


(a) Graphic symbol

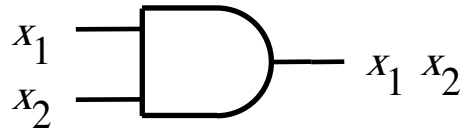
$s_1$	$s_0$	$f$
0	0	$w_0$
0	1	$w_1$
1	0	$w_2$
1	1	$w_3$

(b) Truth table

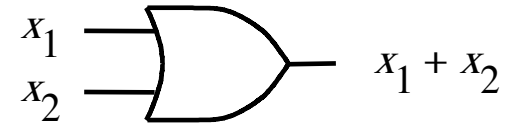
# The Three Basic Logic Gates



NOT gate

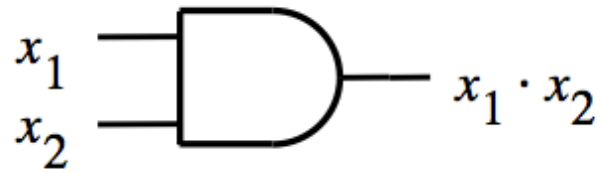


AND gate

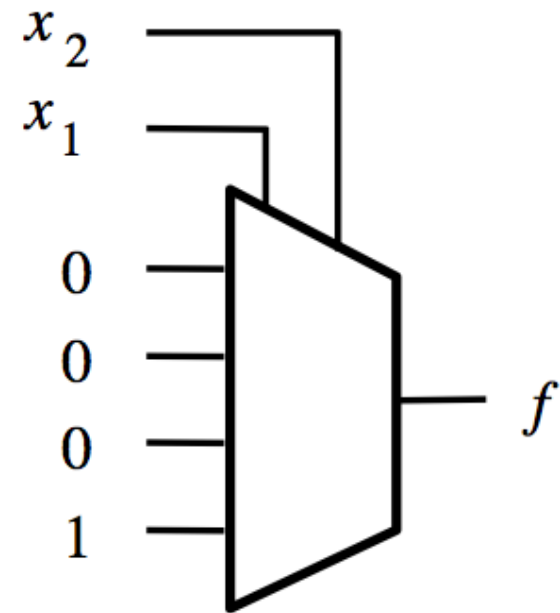


OR gate

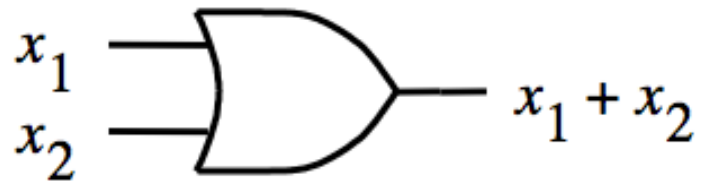
# Building an AND Gate with 4-to-1 Mux



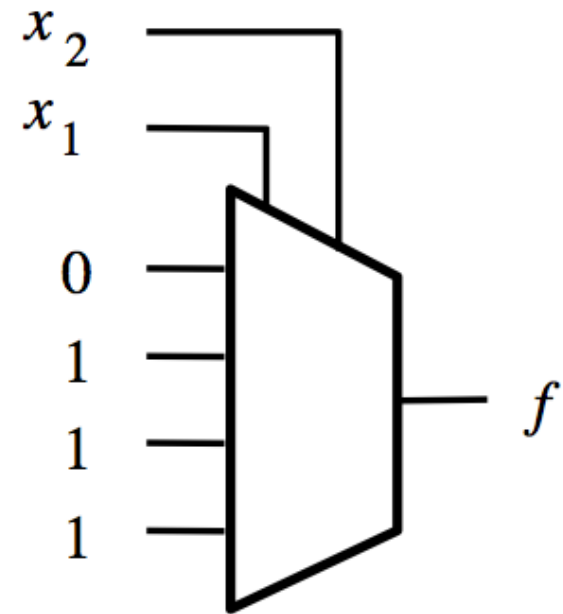
$x_1$	$x_2$	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1



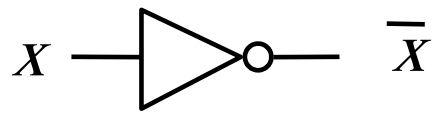
# Building an OR Gate with 4-to-1 Mux



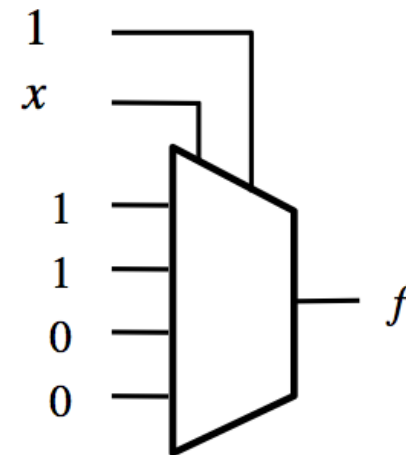
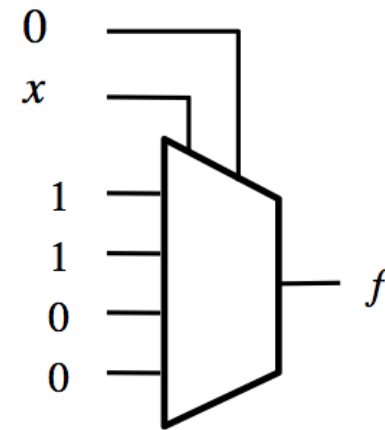
$x_1$	$x_2$	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1



# Building a NOT Gate with 4-to-1 Mux



$x$	$\bar{x}$
0	1
1	0



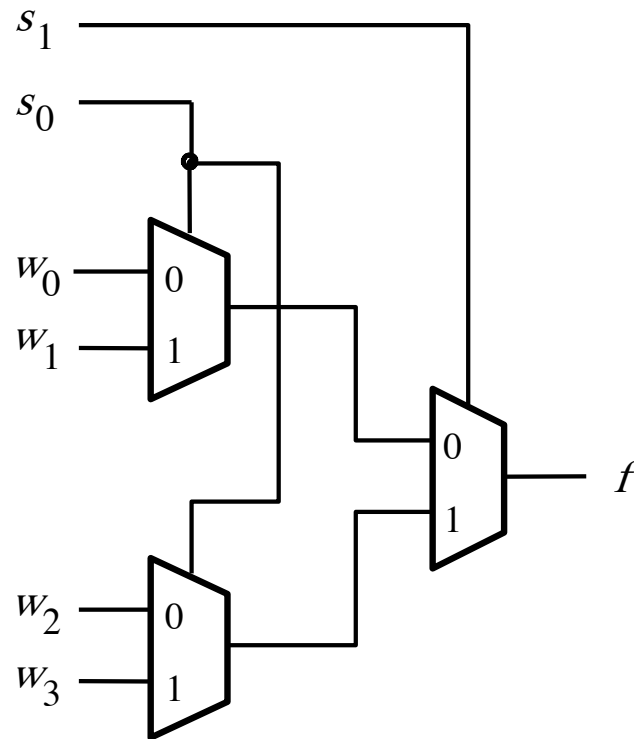
Two alternative solutions.

# Implications

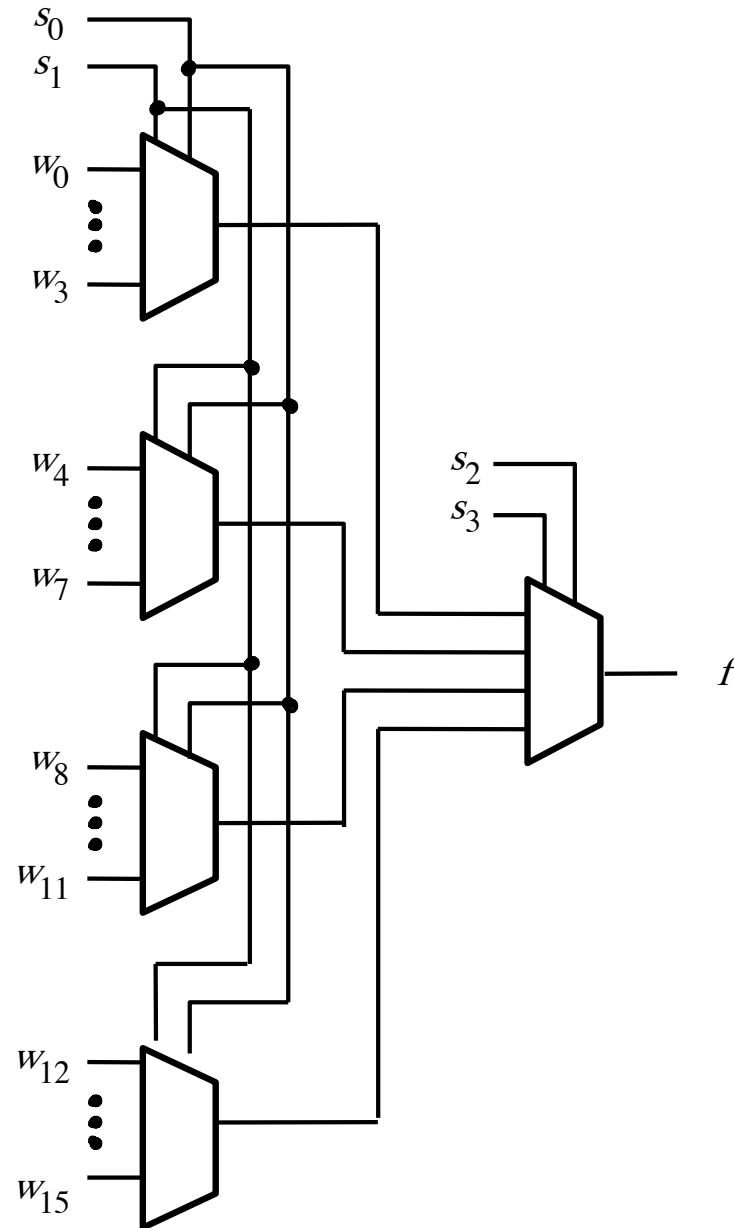
**Any Boolean function can be implemented  
using only 4-to-1 multiplexers!**



# Using three 2-to-1 multiplexers to build one 4-to-1 multiplexer



# 16-1 Multiplexer

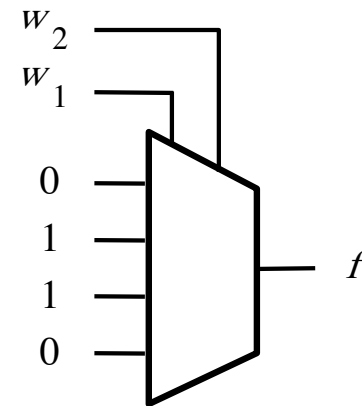


[ Figure 4.4 from the textbook ]

# **Synthesis of Logic Circuits Using Multiplexers**

# Implementation of a logic function with a 4x1 multiplexer

$w_1$	$w_2$	$f$
0	0	0
0	1	1
1	0	1
1	1	0



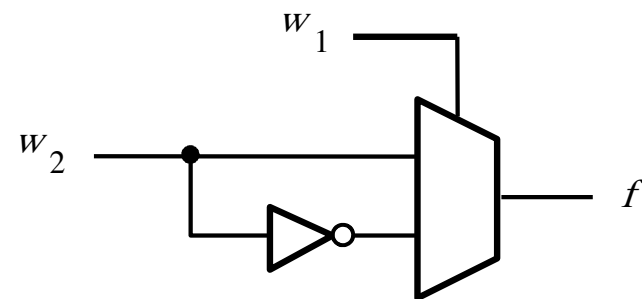
# Implementation of the same logic function with a 2x1 multiplexer

$w_1$	$w_2$	$f$
0	0	0
0	1	1
1	0	1
1	1	0

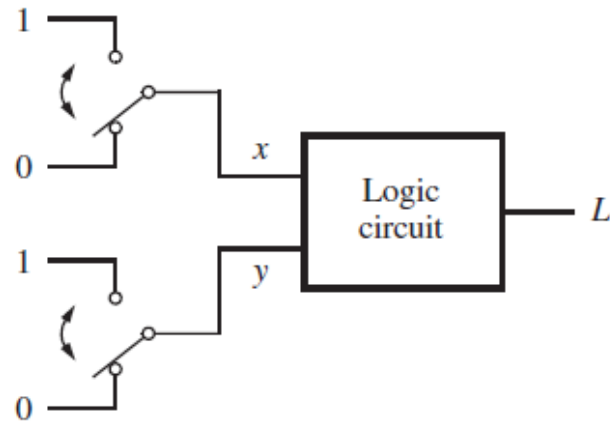
$w_1$	$f$
0	$w_2$
1	$\bar{w}_2$

(b) Modified truth table



(c) Circuit

# The XOR Logic Gate

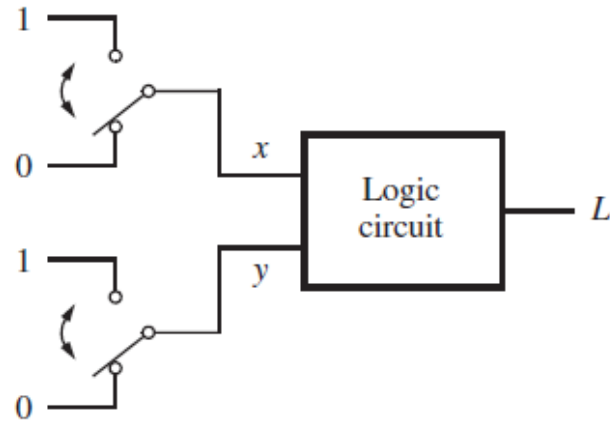


(a) Two switches that control a light

$x$	$y$	$L$
0	0	0
0	1	1
1	0	1
1	1	0

(b) Truth table

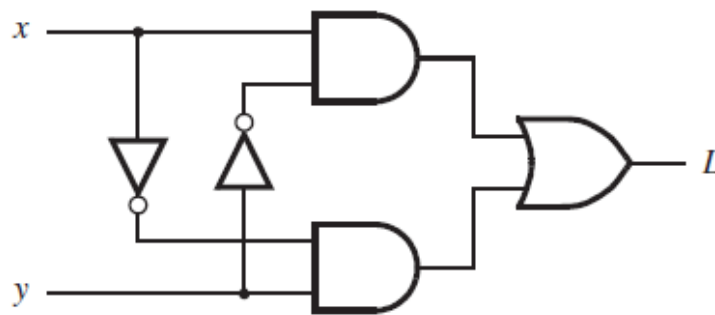
# The XOR Logic Gate



(a) Two switches that control a light

$x$	$y$	$L$
0	0	0
0	1	1
1	0	1
1	1	0

(b) Truth table

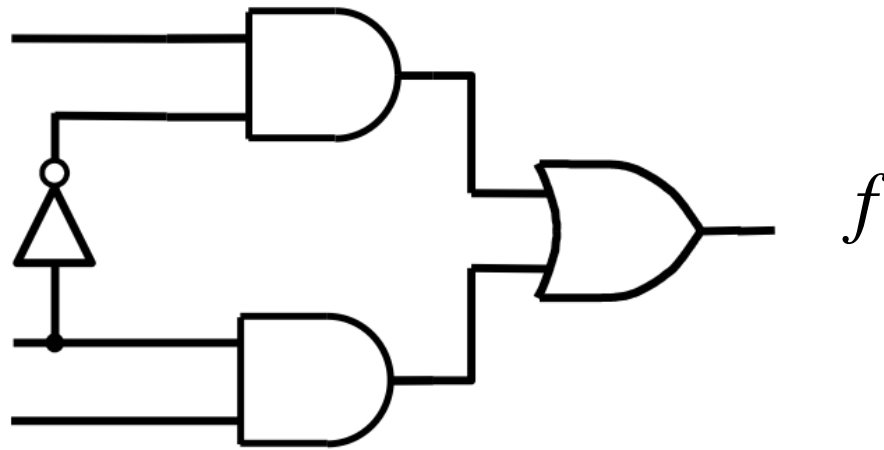


(c) Logic network



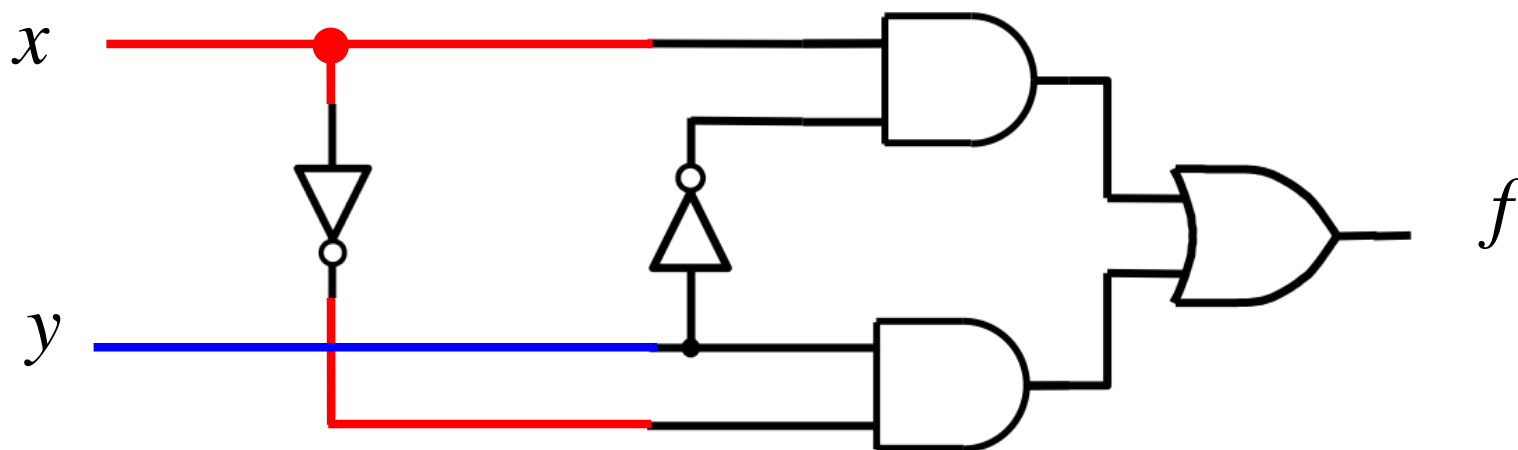
(d) XOR gate symbol

# Implementation of the XOR Logic Gate with a 2-to-1 multiplexer and one NOT

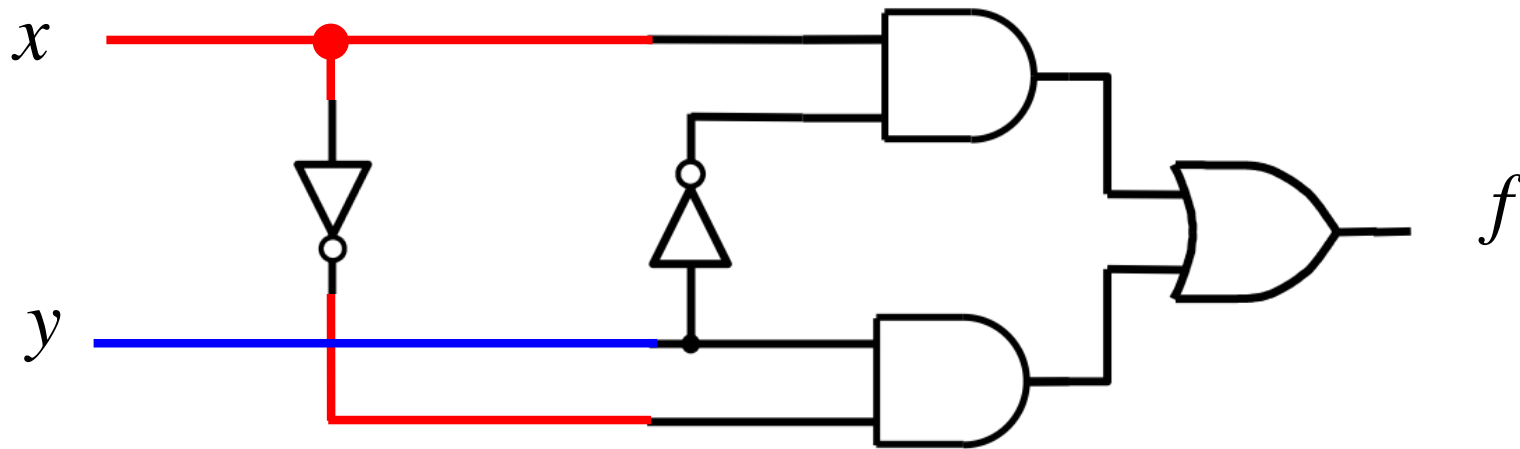




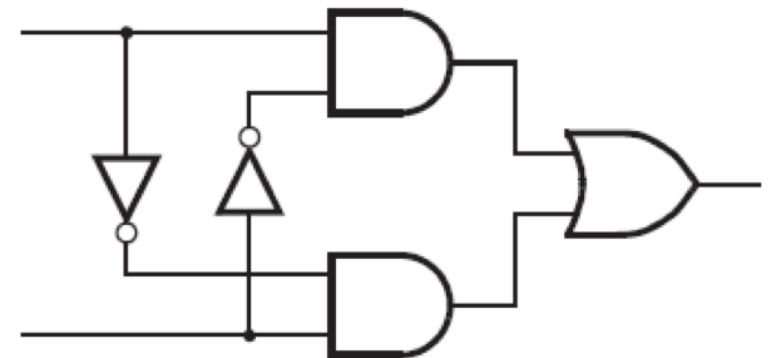
# Implementation of the XOR Logic Gate with a 2-to-1 multiplexer and one NOT



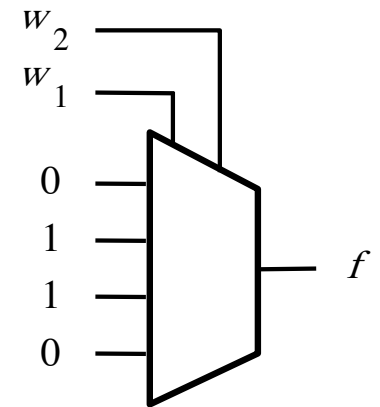
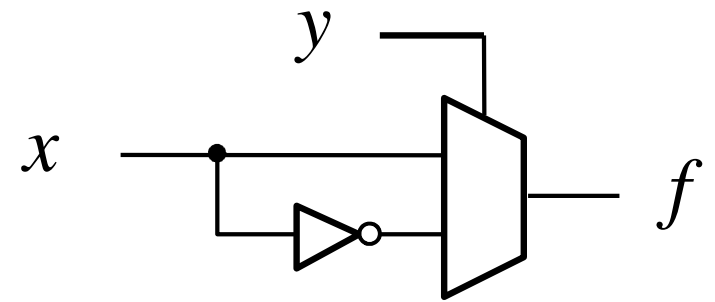
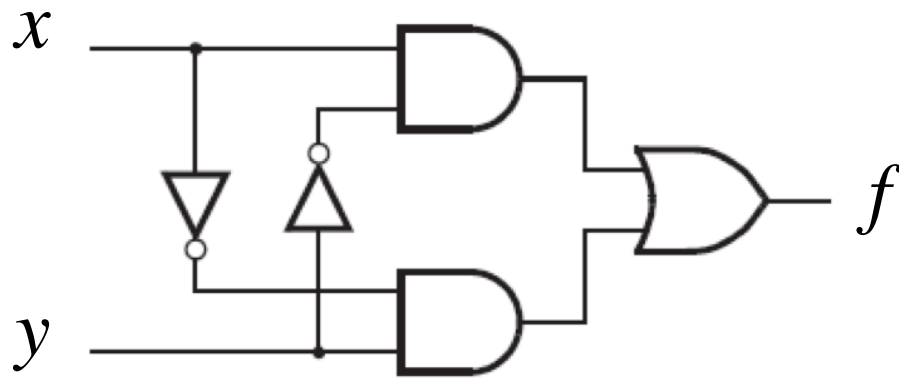
# Implementation of the XOR Logic Gate with a 2-to-1 multiplexer and one NOT



These two circuits are equivalent  
(the wires of the bottom AND gate are flipped)



**In other words,  
all four of these are equivalent!**



# **Another Example (3-input XOR)**

# Implementation of 3-input XOR with 2-to-1 Multiplexers

$w_1$	$w_2$	$w_3$	$f$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

# Implementation of 3-input XOR with 2-to-1 Multiplexers

$w_1$	$w_2$	$w_3$	$f$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

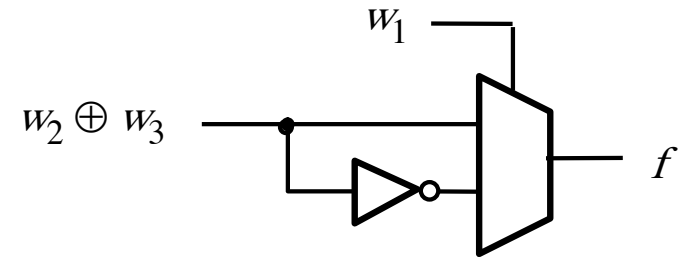
Red annotations in the table:

- A vertical red line is drawn between the  $w_1$  and  $w_2$  columns.
- A horizontal red line is drawn between the  $w_3$  and  $f$  columns.
- Red curly braces group the  $f$  values for  $w_1 = 0$  and  $w_1 = 1$ .
- Red text  $w_2 \oplus w_3$  is placed to the right of the first brace.
- Red text  $\overline{w_2 \oplus w_3}$  is placed to the right of the second brace.

# Implementation of 3-input XOR with 2-to-1 Multiplexers

$w_1$	$w_2$	$w_3$	$f$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(a) Truth table

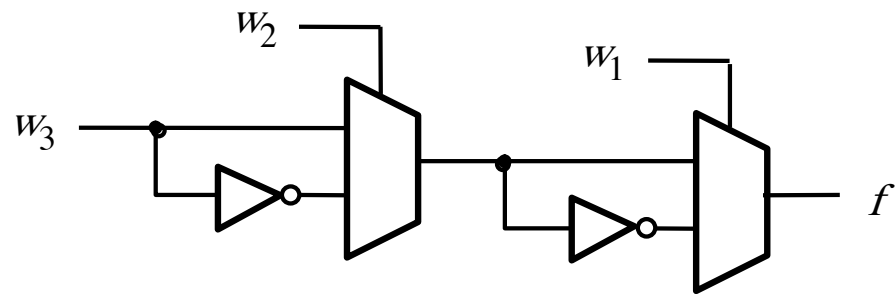


(b) Circuit

# Implementation of 3-input XOR with 2-to-1 Multiplexers

$w_1$	$w_2$	$w_3$	$f$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(a) Truth table



(b) Circuit



# Implementation of 3-input XOR with a 4-to-1 Multiplexer

$w_1$	$w_2$	$w_3$	$f$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

# Implementation of 3-input XOR with a 4-to-1 Multiplexer

$w_1$	$w_2$	$w_3$	$f$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

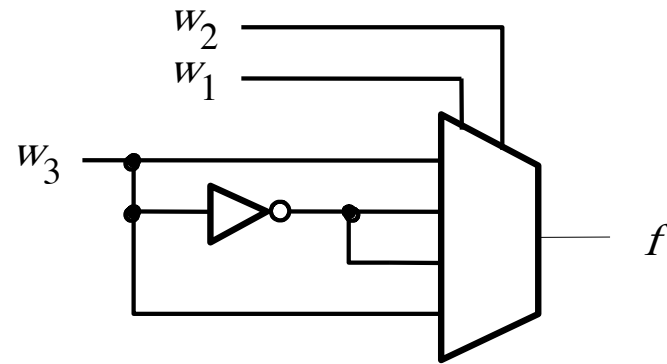
# Implementation of 3-input XOR with a 4-to-1 Multiplexer

$w_1$	$w_2$	$w_3$	$f$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

# Implementation of 3-input XOR with a 4-to-1 Multiplexer

$w_1$	$w_2$	$w_3$	$f$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(a) Truth table



(b) Circuit

# **Multiplexor Synthesis Using Shannon's Expansion**

# Three-input majority function

$w_1$	$w_2$	$w_3$	$f$
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

# Three-input majority function

$w_1$	$w_2$	$w_3$	$f$
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

$w_1$	$f$
0	0
1	1

# Three-input majority function

$w_1$	$w_2$	$w_3$	$f$
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

$w_1$	$f$
0	$w_2 w_3$
1	$w_2 + w_3$



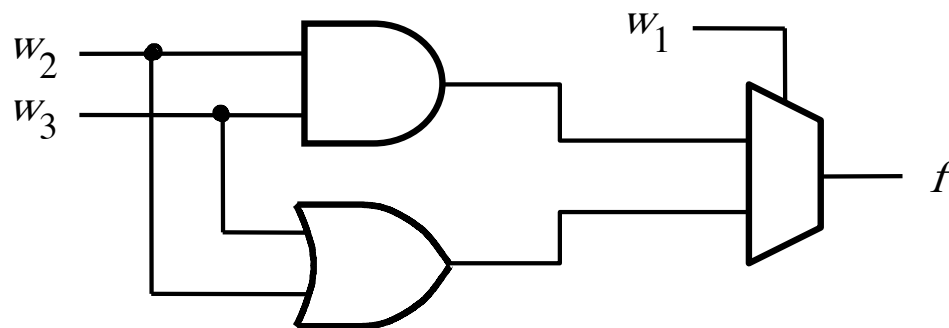
# Three-input majority function

$w_1$	$w_2$	$w_3$	$f$
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

$w_1$	$f$
0	$w_2 w_3$
1	$w_2 + w_3$

(b) Truth table

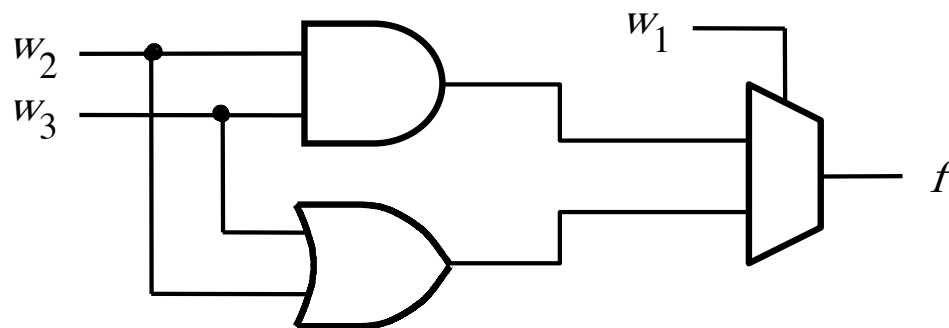


(b) Circuit

# Three-input majority function

$$f = \bar{w}_1 w_2 w_3 + w_1 \bar{w}_2 w_3 + w_1 w_2 \bar{w}_3 + w_1 w_2 w_3$$

$$\begin{aligned} f &= \bar{w}_1 (w_2 w_3) + w_1 (\bar{w}_2 w_3 + w_2 \bar{w}_3 + w_2 w_3) \\ &= \bar{w}_1 (w_2 w_3) + w_1 (w_2 + w_3) \end{aligned}$$



# Shannon's Expansion Theorem

Any Boolean function  $f(w_1, \dots, w_n)$  can be rewritten in the form:

$$f(w_1, w_2, \dots, w_n) = \bar{w}_1 \cdot f(0, w_2, \dots, w_n) + w_1 \cdot f(1, w_2, \dots, w_n)$$

# Shannon's Expansion Theorem

Any Boolean function  $f(w_1, \dots, w_n)$  can be rewritten in the form:

$$f(w_1, w_2, \dots, w_n) = \bar{w}_1 \cdot f(0, w_2, \dots, w_n) + w_1 \cdot f(1, w_2, \dots, w_n)$$

$$f = \bar{w}_1 f_{\bar{w}_1} + w_1 f_{w_1}$$

# Shannon's Expansion Theorem

Any Boolean function  $f(w_1, \dots, w_n)$  can be rewritten in the form:

$$f(w_1, w_2, \dots, w_n) = \bar{w}_1 \cdot f(0, w_2, \dots, w_n) + w_1 \cdot f(1, w_2, \dots, w_n)$$

$$f = \bar{w}_1 f_{\bar{w}_1} + w_1 f_{w_1}$$

cofactor

cofactor

# Shannon's Expansion Theorem (Example)

$$f(w_1, w_2, w_3) = w_1w_2 + w_1w_3 + w_2w_3$$

# Shannon's Expansion Theorem (Example)

$$f(w_1, w_2, w_3) = w_1w_2 + w_1w_3 + w_2w_3$$

$$f(w_1, w_2, w_3) = w_1w_2 + w_1w_3 + w_2w_3 (\overline{w_1} + w_1)$$

# Shannon's Expansion Theorem (Example)

$$f(w_1, w_2, w_3) = w_1w_2 + w_1w_3 + w_2w_3$$

$$f(w_1, w_2, w_3) = w_1w_2 + w_1w_3 + w_2w_3 (\bar{w}_1 + w_1)$$

$$\begin{aligned} f &= \bar{w}_1(0 \cdot w_2 + 0 \cdot w_3 + w_2w_3) + w_1(1 \cdot w_2 + 1 \cdot w_3 + w_2w_3) \\ &= \bar{w}_1(w_2w_3) + w_1(w_2 + w_3) \end{aligned}$$



# Shannon's Expansion Theorem (In terms of more than one variable)

$$f(w_1, \dots, w_n) = \bar{w}_1 \bar{w}_2 \cdot f(0, 0, w_3, \dots, w_n) + \bar{w}_1 w_2 \cdot f(0, 1, w_3, \dots, w_n) \\ + w_1 \bar{w}_2 \cdot f(1, 0, w_3, \dots, w_n) + w_1 w_2 \cdot f(1, 1, w_3, \dots, w_n)$$

This form is suitable for implementation with a 4x1 multiplexer.

# **Another Example**

**Factor and implement the following function with a 2x1 multiplexer**

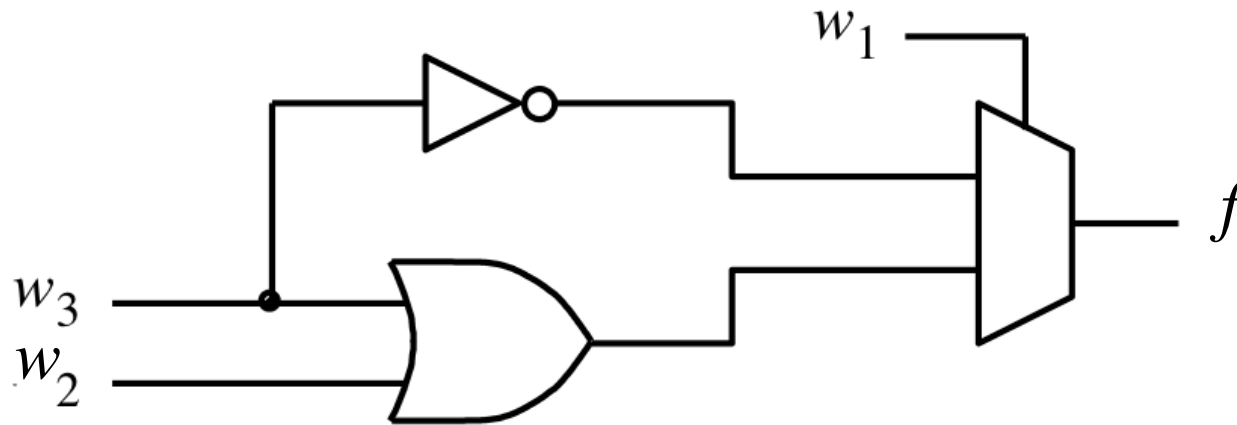
$$f = \bar{w}_1 \bar{w}_3 + w_1 w_2 + w_1 w_3$$

**Factor and implement the following function with a 2x1 multiplexer**

$$f = \bar{w}_1 \bar{w}_3 + w_1 w_2 + w_1 w_3$$

$$\begin{aligned} f &= \bar{w}_1 f_{\bar{w}_1} + w_1 f_{w_1} \\ &= \bar{w}_1 (\bar{w}_3) + w_1 (w_2 + w_3) \end{aligned}$$

# Factor and implement the following function with a 2x1 multiplexer



$$\begin{aligned} f &= \bar{w}_1 f_{\bar{w}_1} + w_1 f_{w_1} \\ &= \bar{w}_1 (\bar{w}_3) + w_1 (w_2 + w_3) \end{aligned}$$

**Factor and implement the following function with a 4x1 multiplexer**

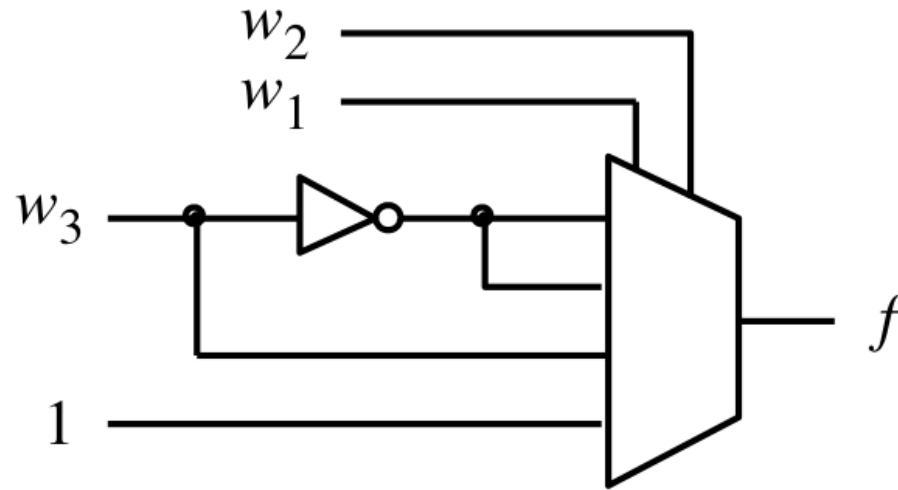
$$f = \bar{w}_1 \bar{w}_3 + w_1 w_2 + w_1 w_3$$

**Factor and implement the following function with a 4x1 multiplexer**

$$f = \bar{w}_1\bar{w}_3 + w_1w_2 + w_1w_3$$

$$\begin{aligned} f &= \bar{w}_1\bar{w}_2f_{\bar{w}_1\bar{w}_2} + \bar{w}_1w_2f_{\bar{w}_1w_2} + w_1\bar{w}_2f_{w_1\bar{w}_2} + w_1w_2f_{w_1w_2} \\ &= \bar{w}_1\bar{w}_2(\bar{w}_3) + \bar{w}_1w_2(\bar{w}_3) + w_1\bar{w}_2(w_3) + w_1w_2(1) \end{aligned}$$

# Factor and implement the following function with a 4x1 multiplexer



$$\begin{aligned} f &= \bar{w}_1 \bar{w}_2 f_{\bar{w}_1 \bar{w}_2} + \bar{w}_1 w_2 f_{\bar{w}_1 w_2} + w_1 \bar{w}_2 f_{w_1 \bar{w}_2} + w_1 w_2 f_{w_1 w_2} \\ &= \bar{w}_1 \bar{w}_2 (\bar{w}_3) + \bar{w}_1 w_2 (\bar{w}_3) + w_1 \bar{w}_2 (w_3) + w_1 w_2 (1) \end{aligned}$$



# **Yet Another Example**

**Factor and implement the following function using only 2x1 multiplexers**

$$f = w_1w_2 + w_1w_3 + w_2w_3$$

**Factor and implement the following function using only 2x1 multiplexers**

$$f = w_1w_2 + w_1w_3 + w_2w_3$$

$$\begin{aligned} f &= \bar{w}_1(w_2w_3) + w_1(w_2 + w_3 + w_2w_3) \\ &= \bar{w}_1(w_2w_3) + w_1(w_2 + w_3) \end{aligned}$$

**Factor and implement the following function using only 2x1 multiplexers**

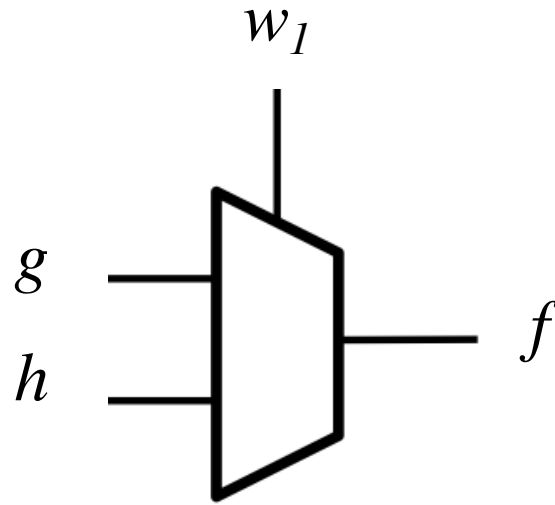
$$f = w_1w_2 + w_1w_3 + w_2w_3$$

$$f = \bar{w}_1(w_2w_3) + w_1(w_2 + w_3 + w_2w_3)$$

$$= \bar{w}_1 \underbrace{(w_2w_3)}_g + w_1 \underbrace{(w_2 + w_3)}_h$$

$$g = w_2w_3 \quad h = w_2 + w_3$$

**Factor and implement the following function using only 2x1 multiplexers**



$$f = \bar{w}_1(w_2w_3) + w_1(w_2 + w_3 + w_2w_3)$$

$$= \bar{w}_1(\underbrace{w_2w_3}) + w_1(\underbrace{w_2 + w_3})$$

$$g = w_2w_3 \quad h = w_2 + w_3$$

**Factor and implement the following function using only 2x1 multiplexers**

$$g = w_2 w_3$$

$$h = w_2 + w_3$$

**Factor and implement the following function using only 2x1 multiplexers**

$$g = w_2 w_3$$



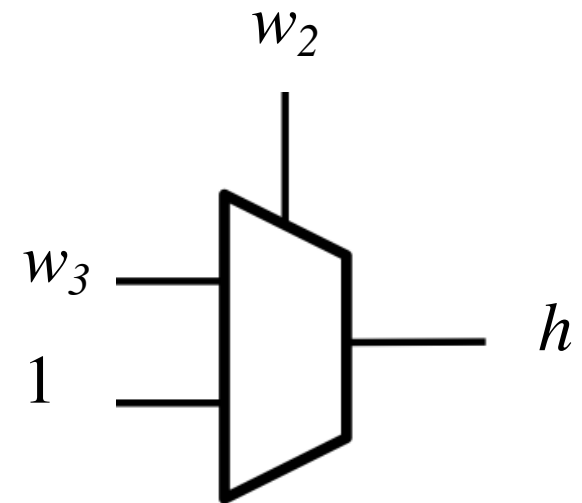
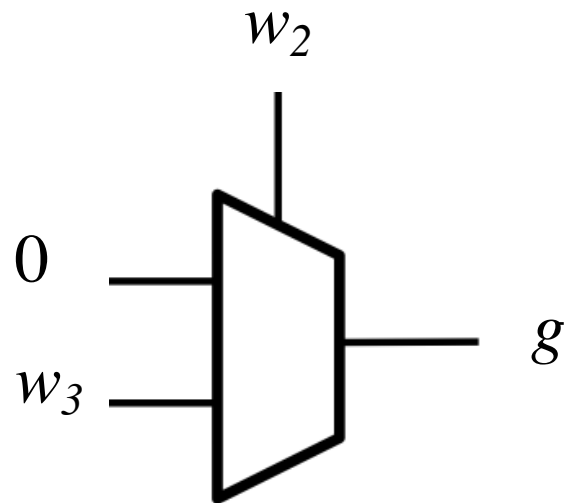
$$g = \bar{w}_2(0) + w_2(w_3)$$

$$h = w_2 + w_3$$



$$h = \bar{w}_2(w_3) + w_2(1)$$

# Factor and implement the following function using only 2x1 multiplexers

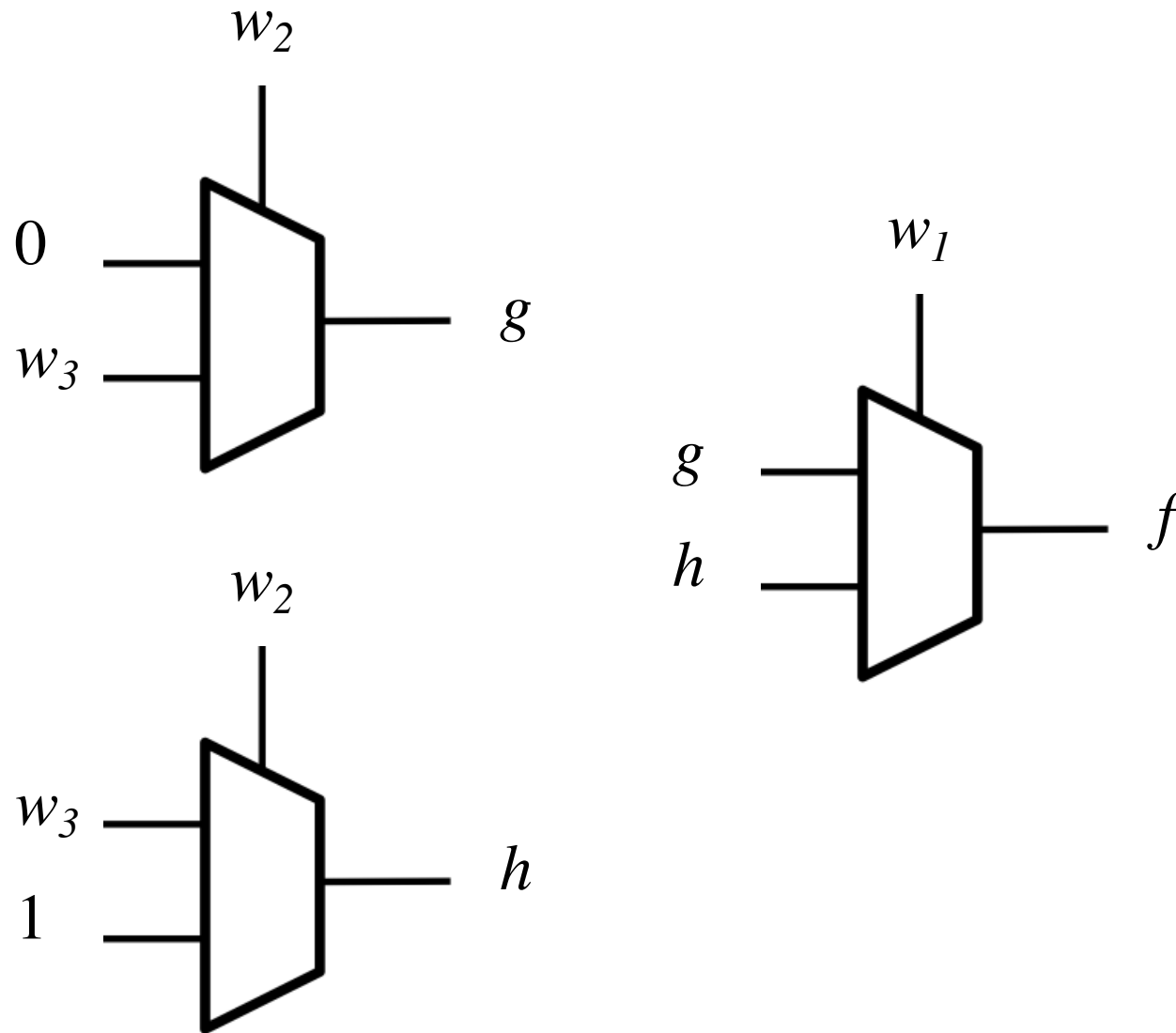


$$g = \bar{w}_2(0) + w_2(w_3)$$

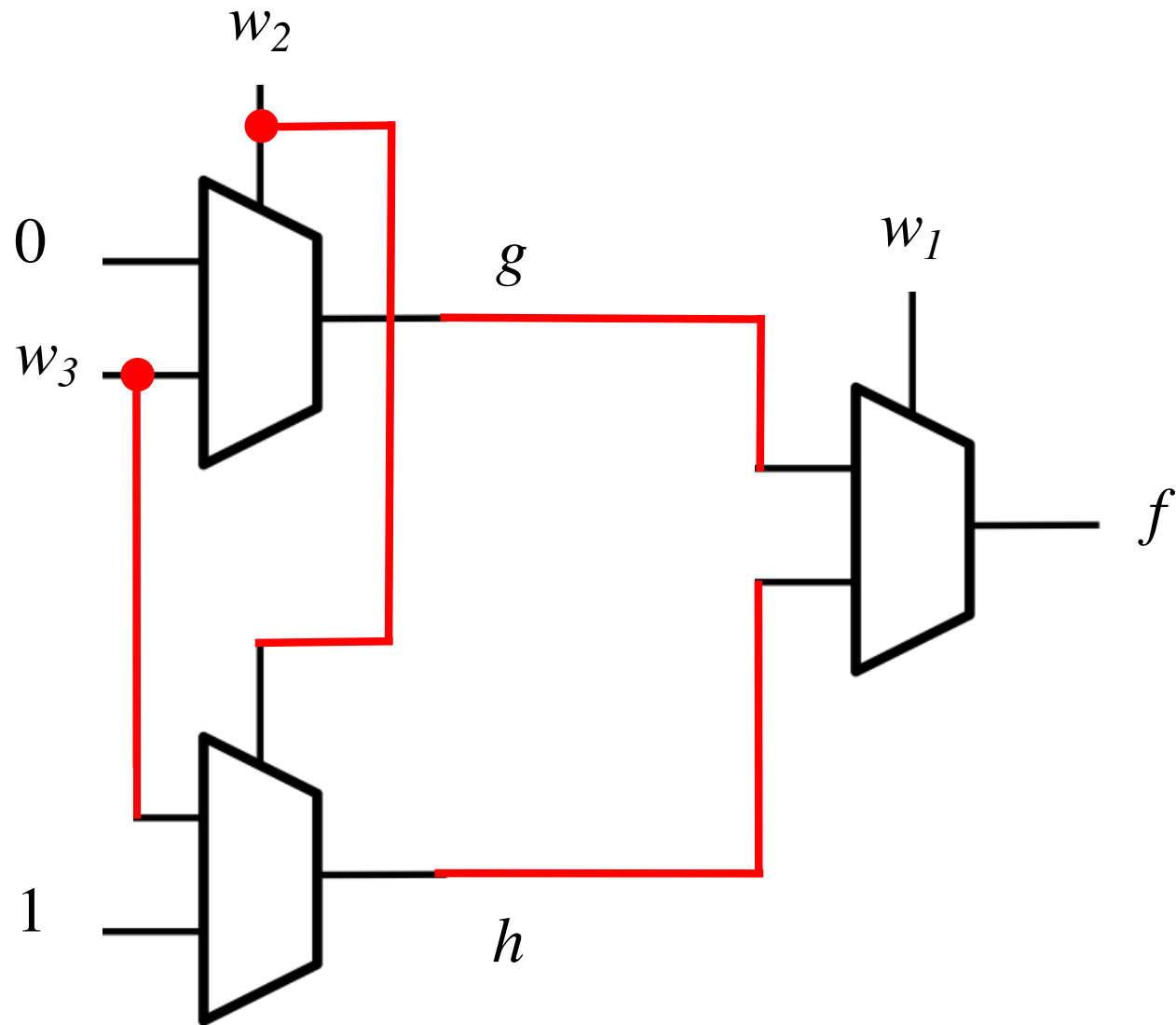
$$h = \bar{w}_2(w_3) + w_2(1)$$



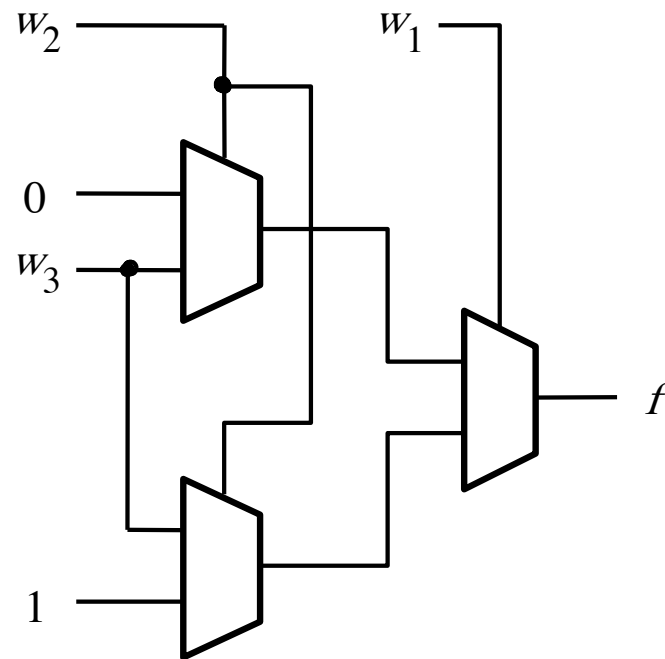
# Finally, we are ready to draw the circuit



**Finally, we are ready to draw the circuit**



# Finally, we are ready to draw the circuit



[ Figure 4.12 from the textbook ]

# Decoders

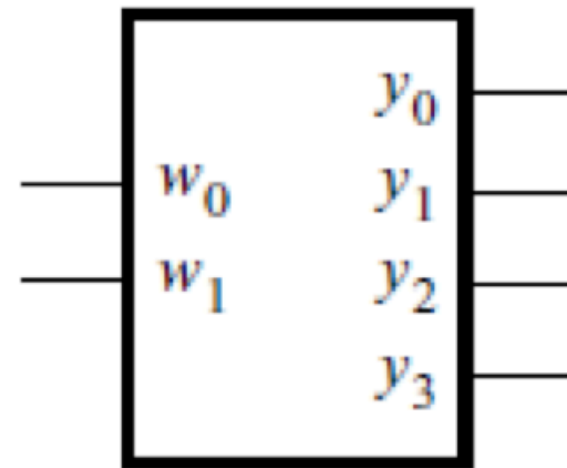
# 2-to-4 Decoder (Definition)

- Has two inputs:  $w_1$  and  $w_0$
- Has four outputs:  $y_0$  ,  $y_1$  ,  $y_2$  , and  $y_3$
- If  $w_1=0$  and  $w_0=0$ , then the output  $y_0$  is set to 1
- If  $w_1=0$  and  $w_0=1$ , then the output  $y_1$  is set to 1
- If  $w_1=1$  and  $w_0=0$ , then the output  $y_2$  is set to 1
- If  $w_1=1$  and  $w_0=1$ , then the output  $y_3$  is set to 1
- Only one output is set to 1. All others are set to 0.

# Truth Table and Graphical Symbol for a 2-to-4 Decoder

$w_1$	$w_0$	$y_0$	$y_1$	$y_2$	$y_3$
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) Truth table



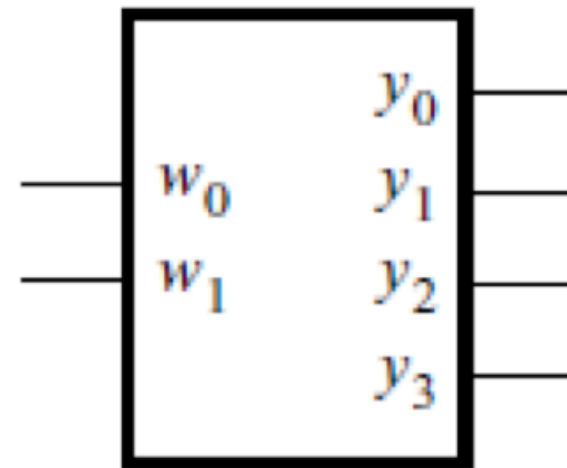
(b) Graphical symbol

# Truth Table and Graphical Symbol for a 2-to-4 Decoder

$w_1$	$w_0$	$y_0$	$y_1$	$y_2$	$y_3$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

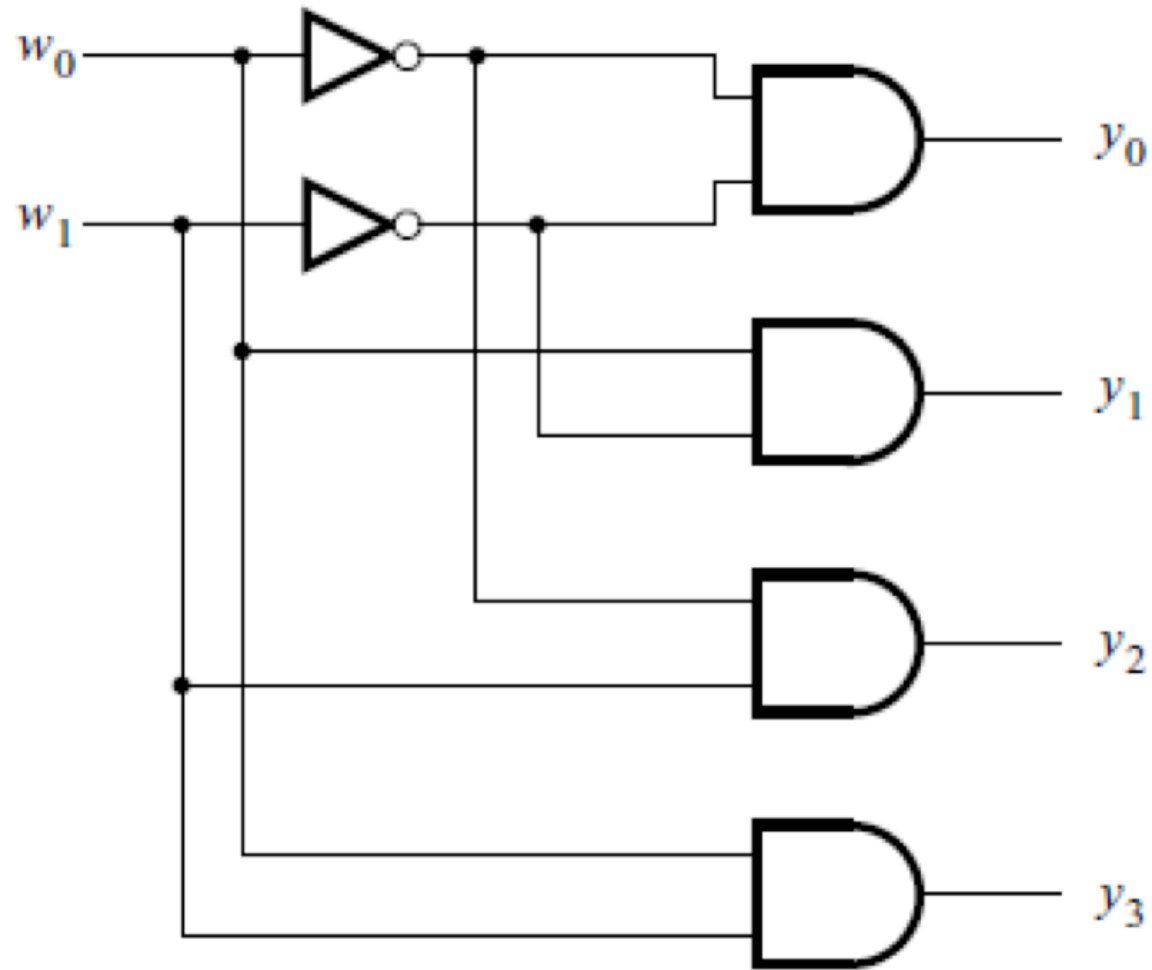
The outputs are “one-hot” encoded

(a) Truth table



(b) Graphical symbol

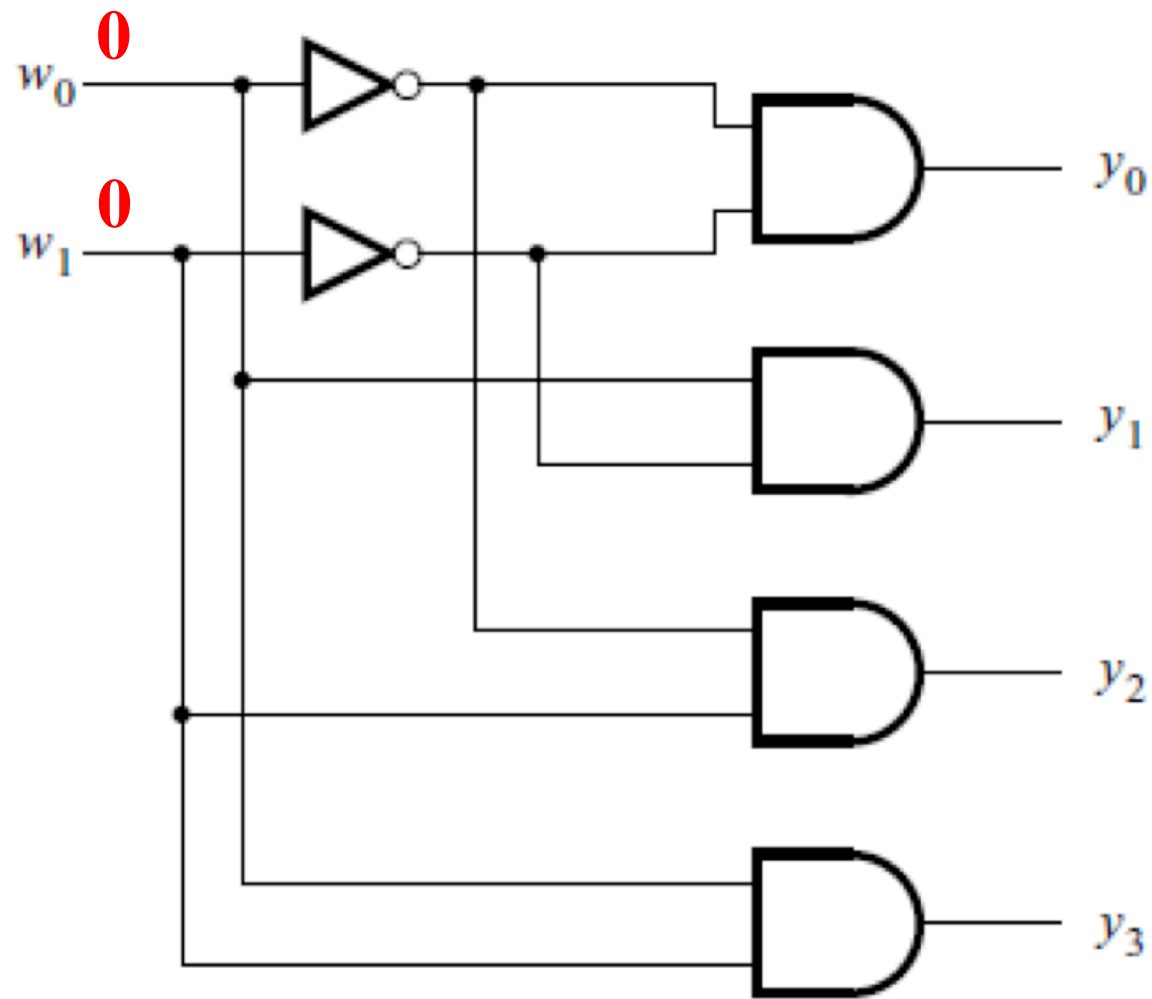
# The Logic Circuit for a 2-to-4 Decoder



[ Figure 4.13c from the textbook ]

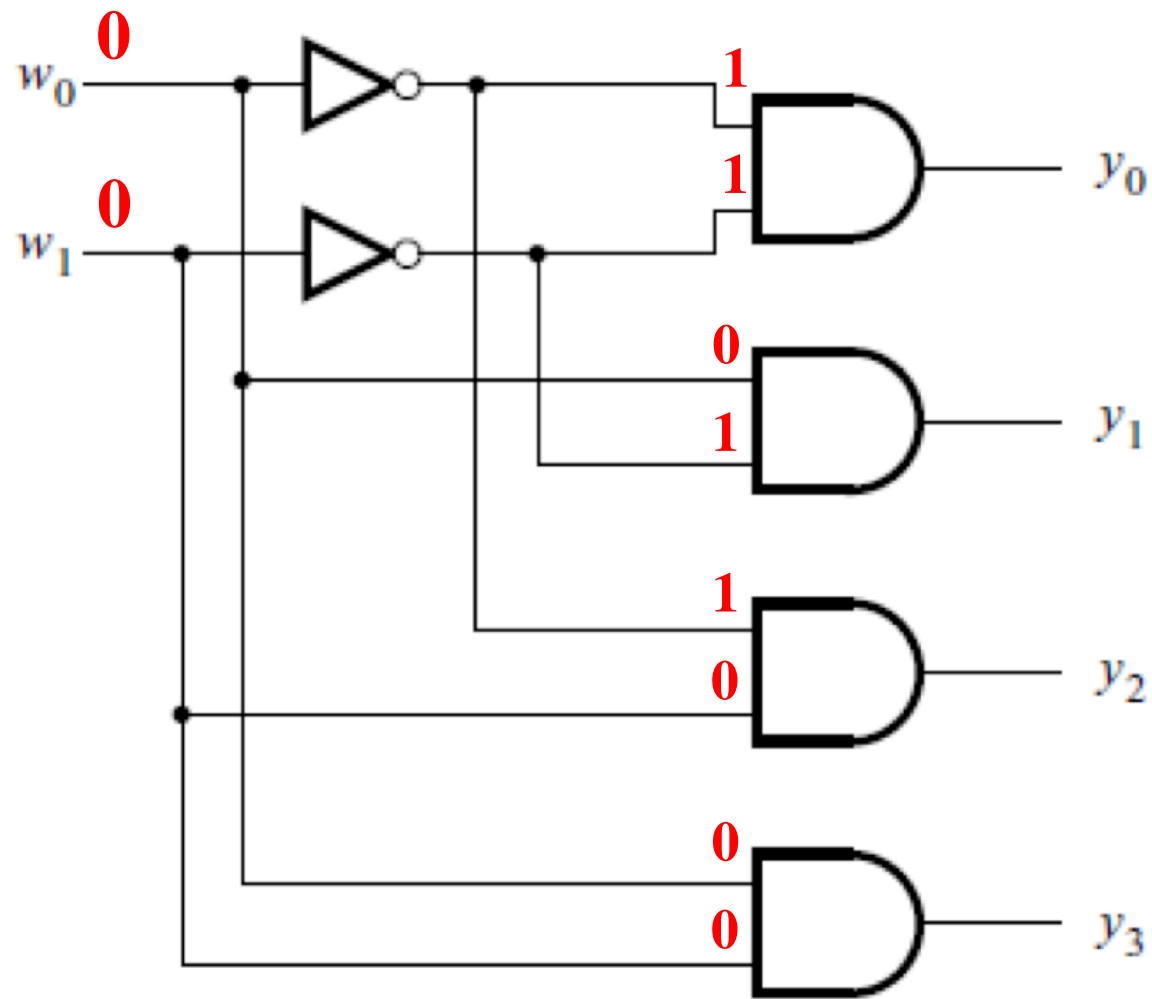


# The Logic Circuit for a 2-to-4 Decoder



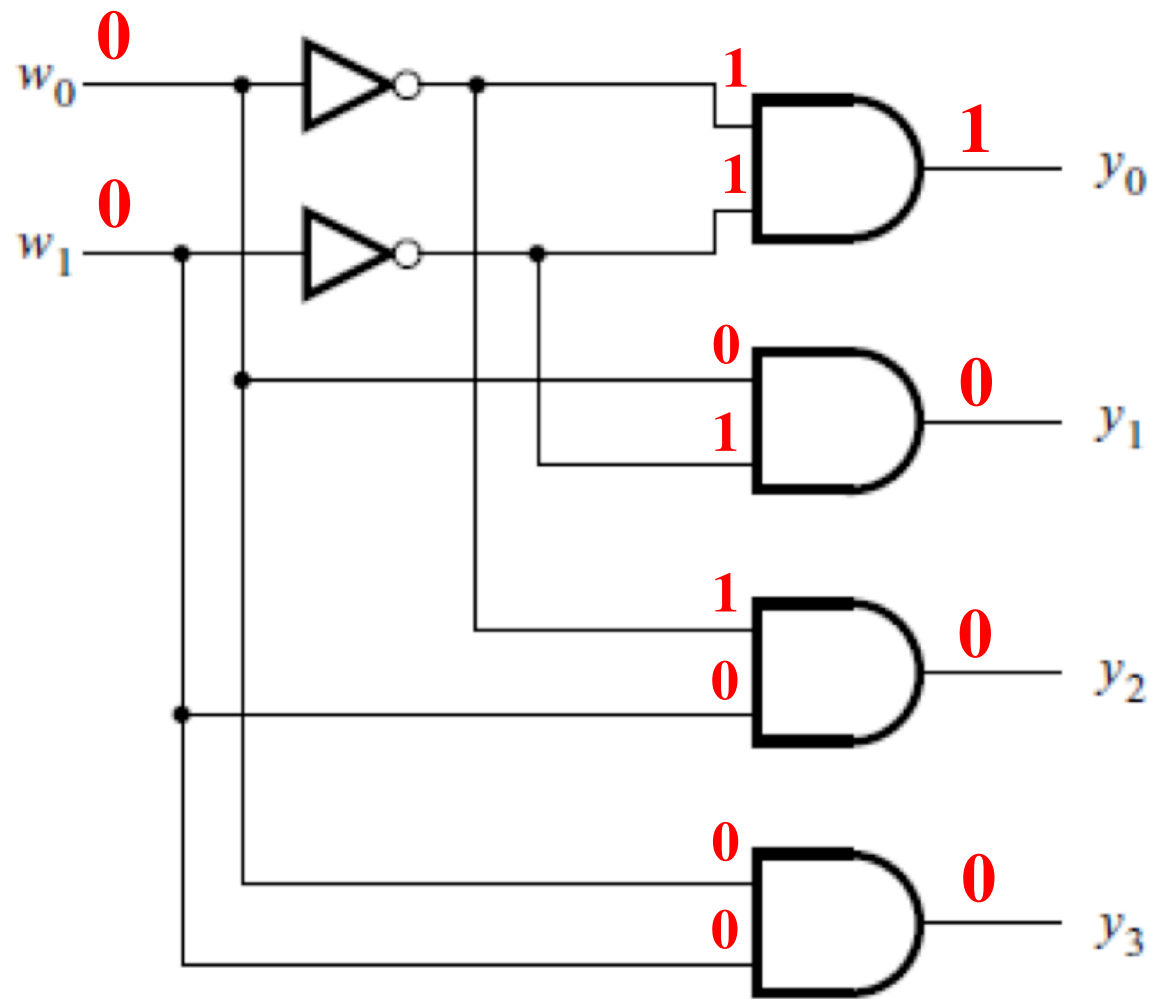
[ Figure 4.13c from the textbook ]

# The Logic Circuit for a 2-to-4 Decoder



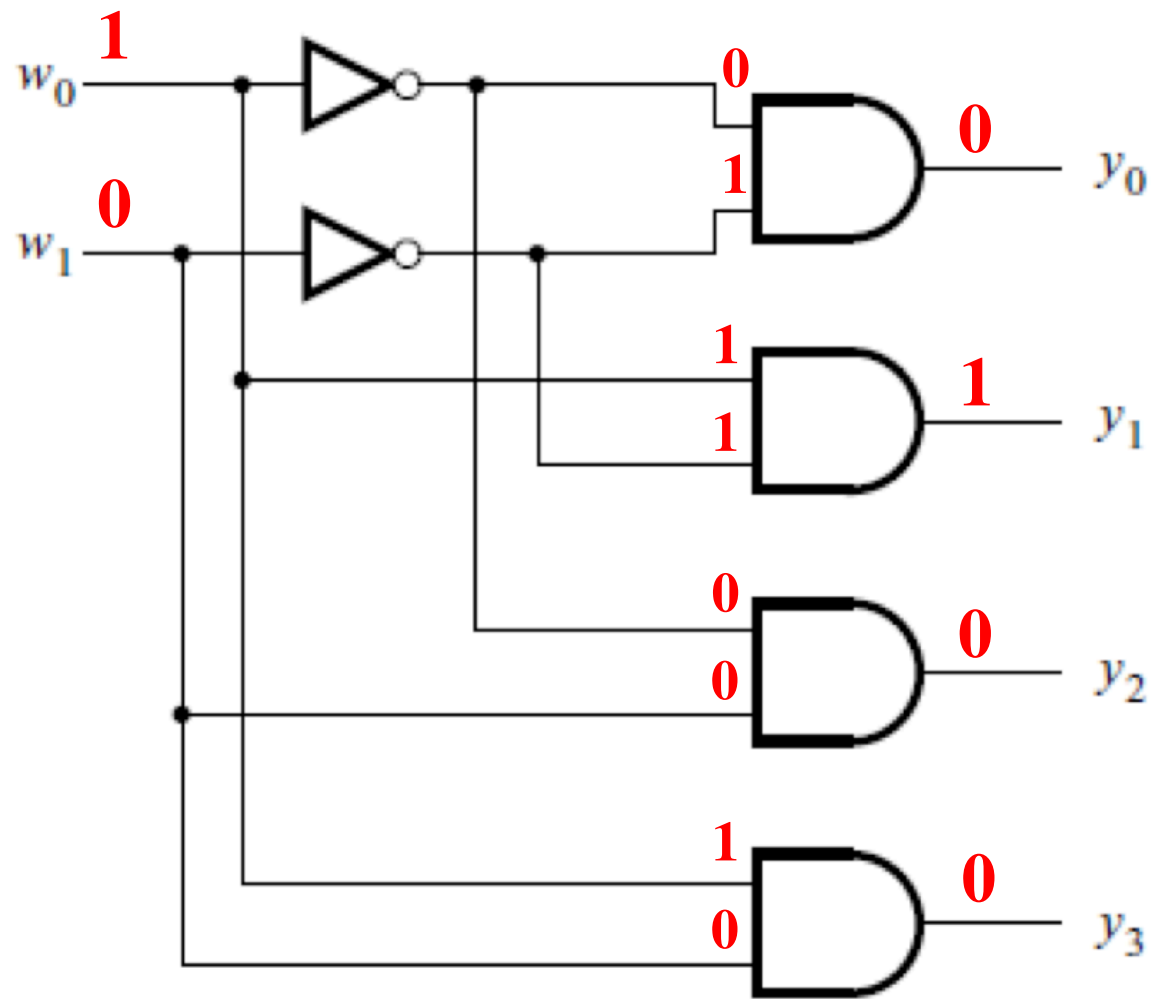
[ Figure 4.13c from the textbook ]

# The Logic Circuit for a 2-to-4 Decoder



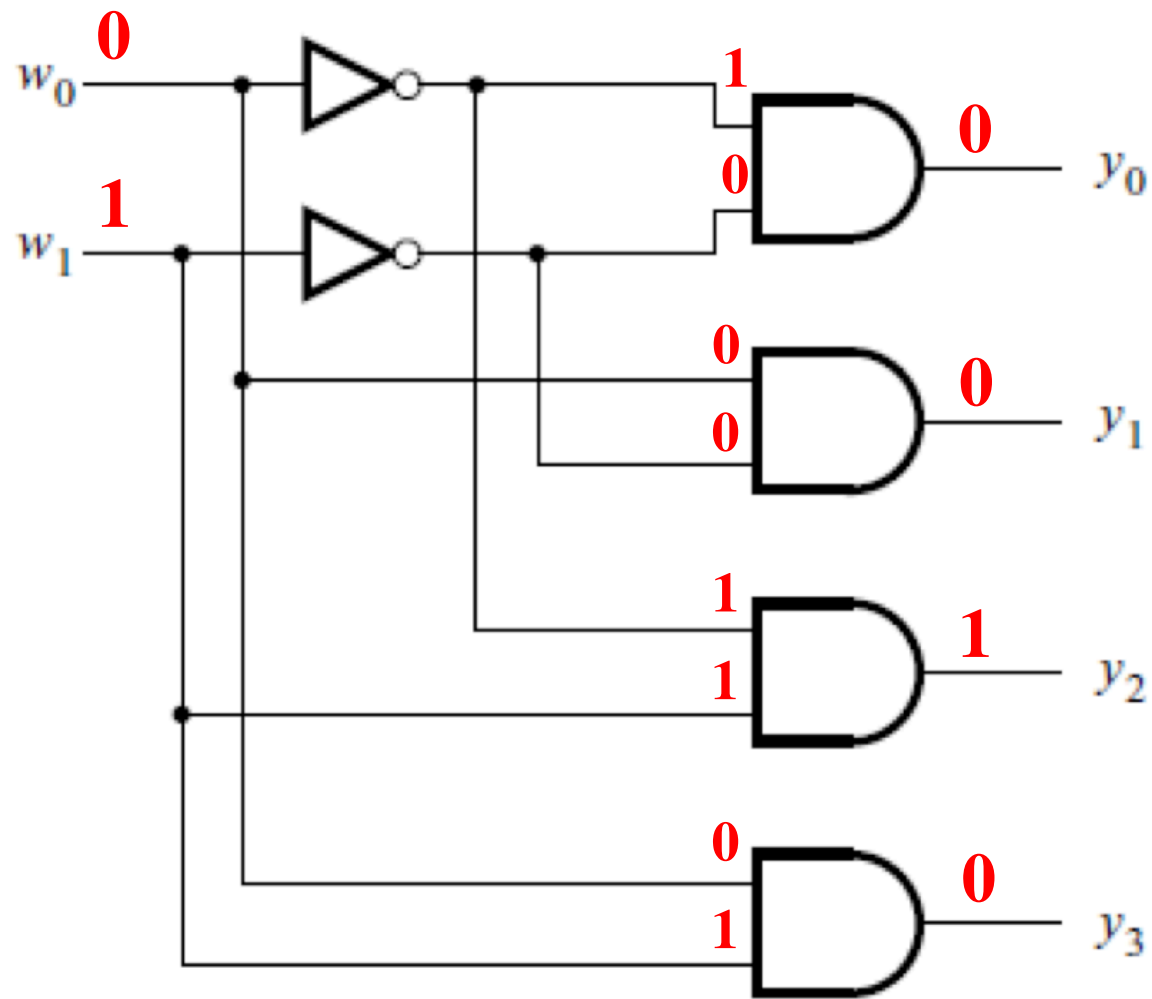
[ Figure 4.13c from the textbook ]

# The Logic Circuit for a 2-to-4 Decoder



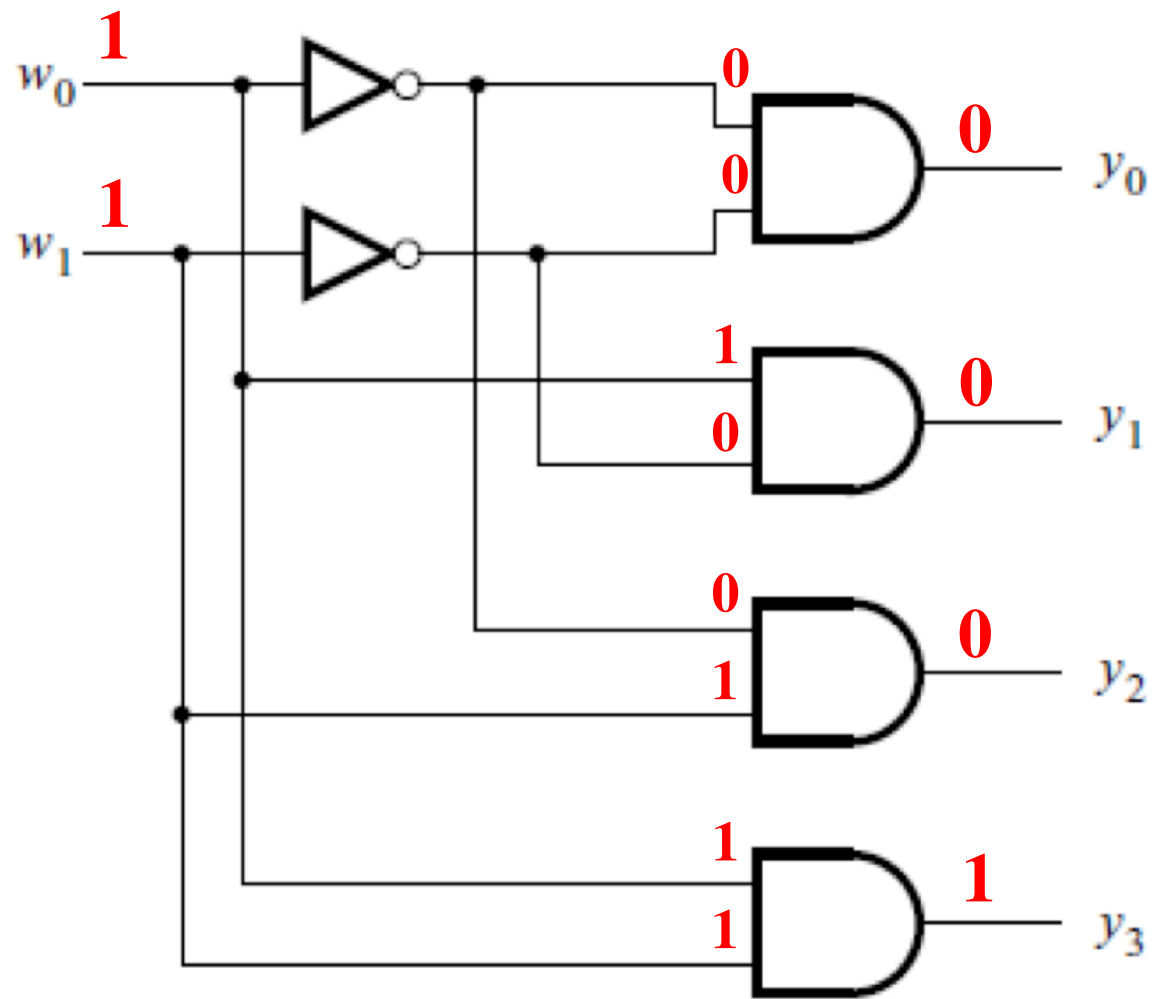
[ Figure 4.13c from the textbook ]

# The Logic Circuit for a 2-to-4 Decoder



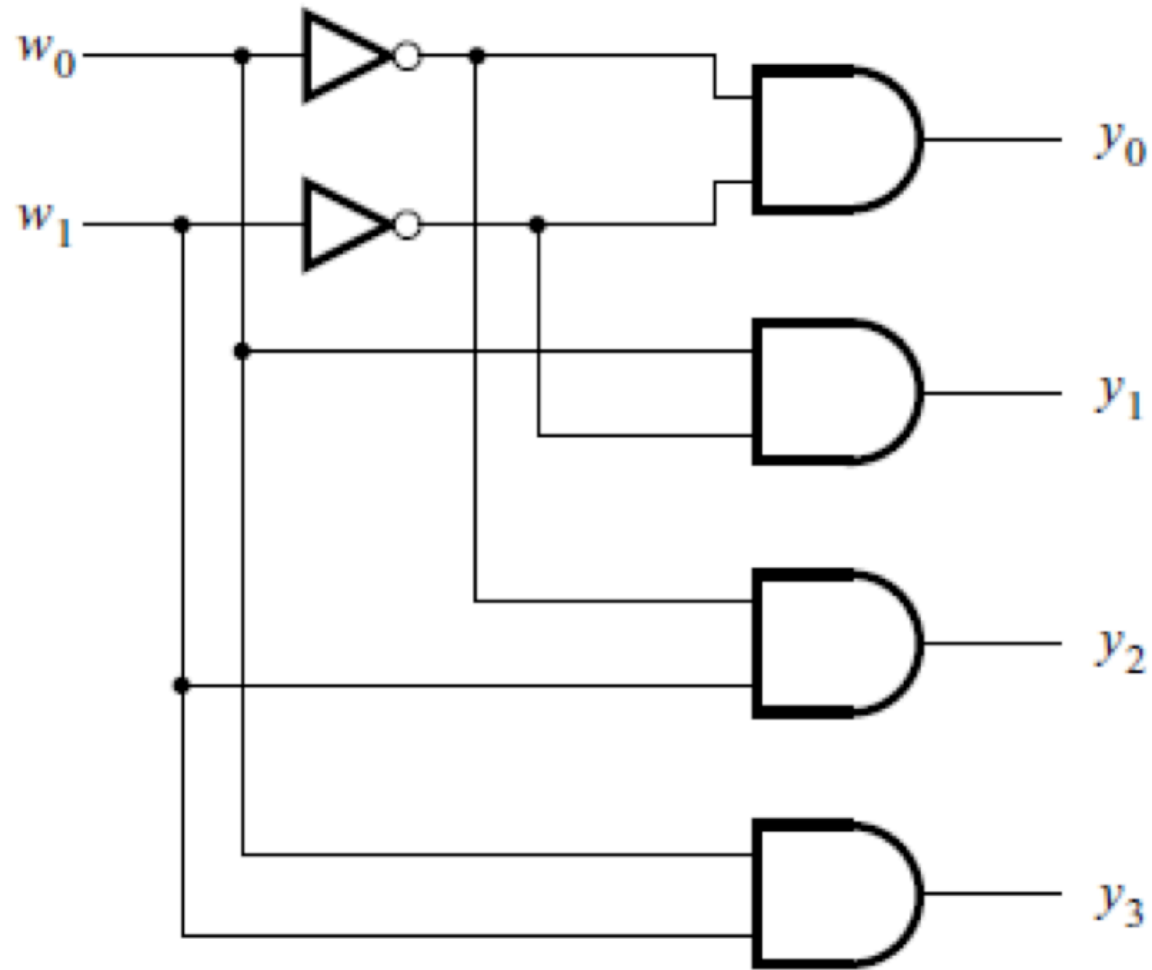
[ Figure 4.13c from the textbook ]

# The Logic Circuit for a 2-to-4 Decoder



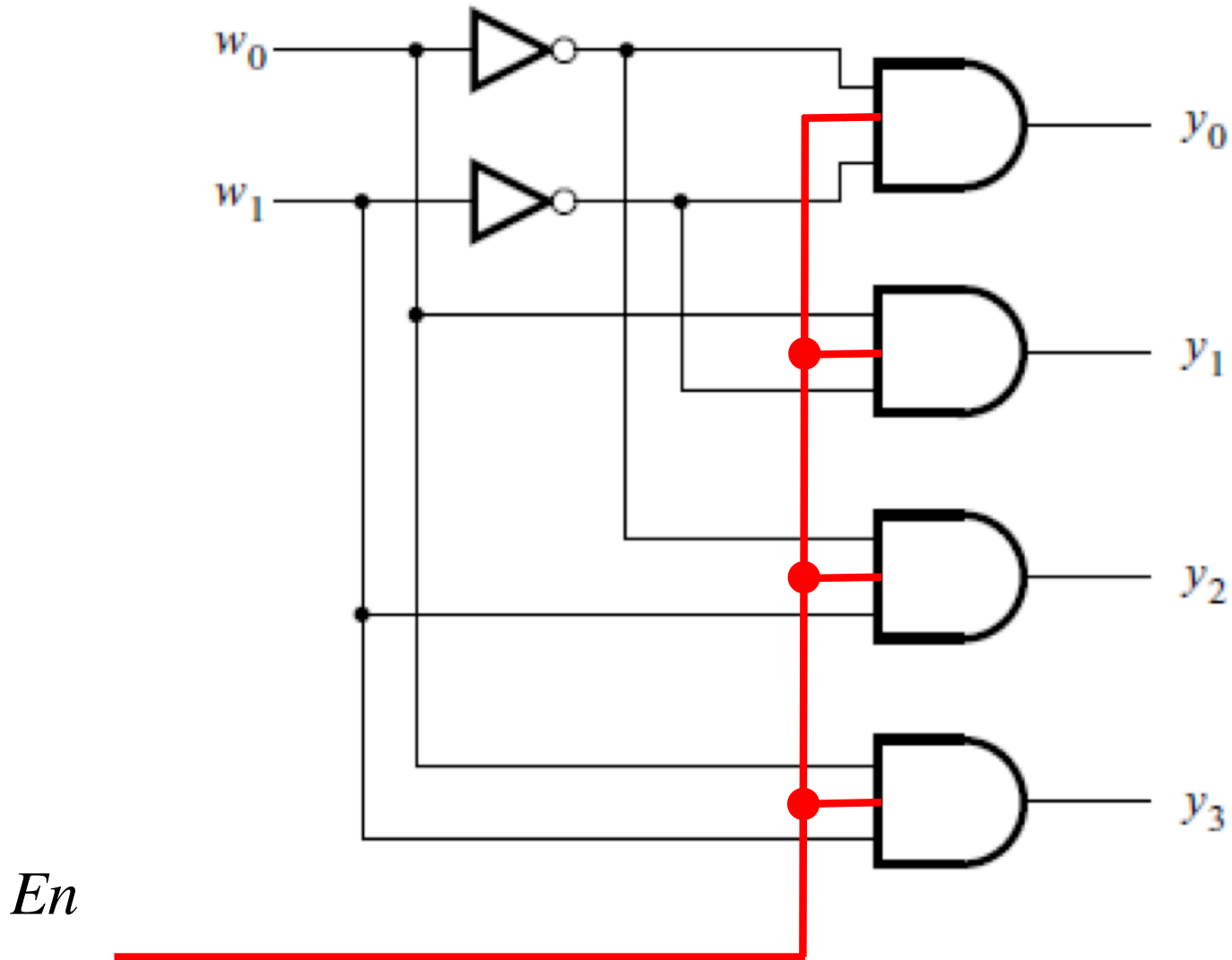
[ Figure 4.13c from the textbook ]

# Adding an Enable Input



[ Figure 4.13c from the textbook ]

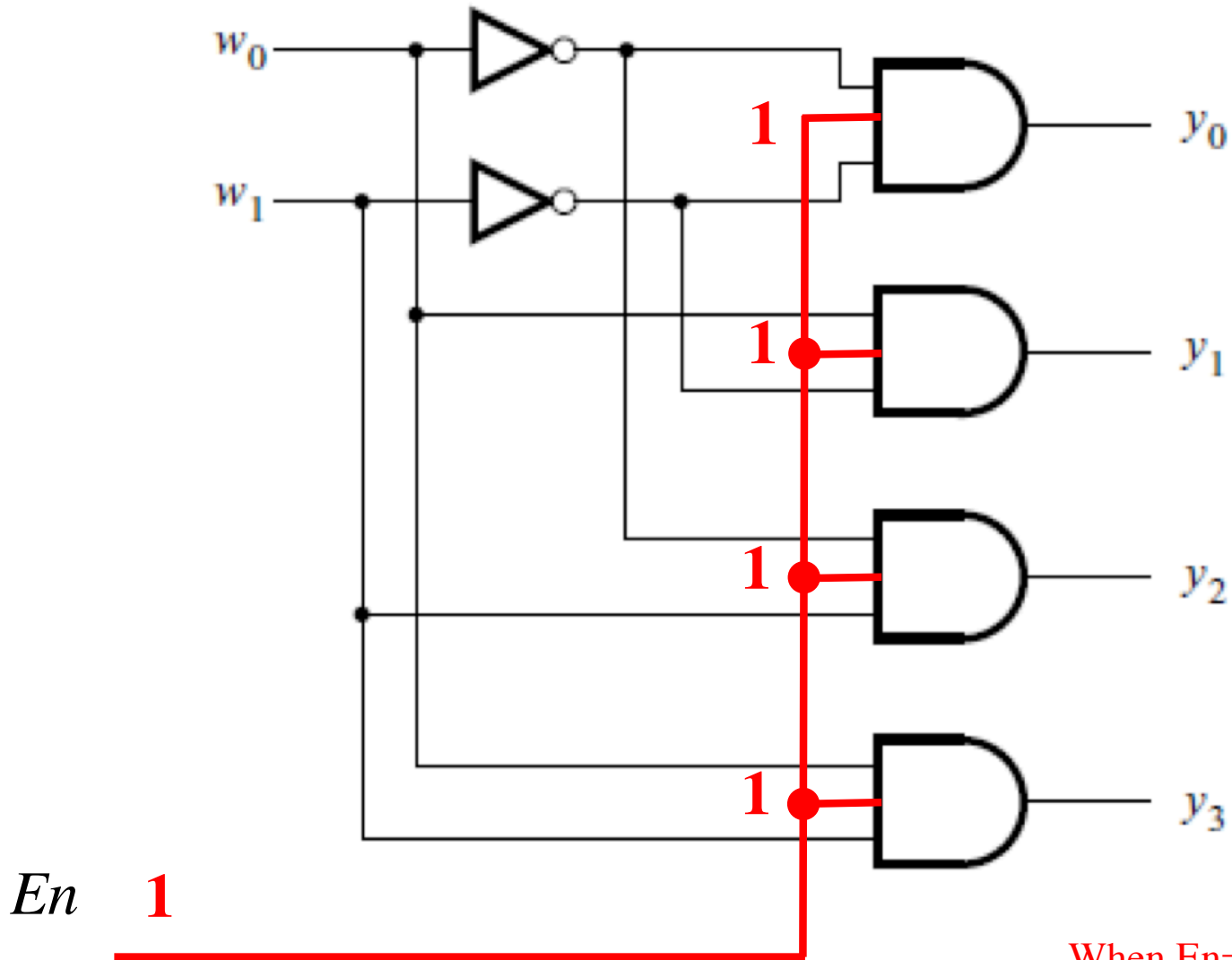
# Adding an Enable Input



[ Figure 4.14c from the textbook ]

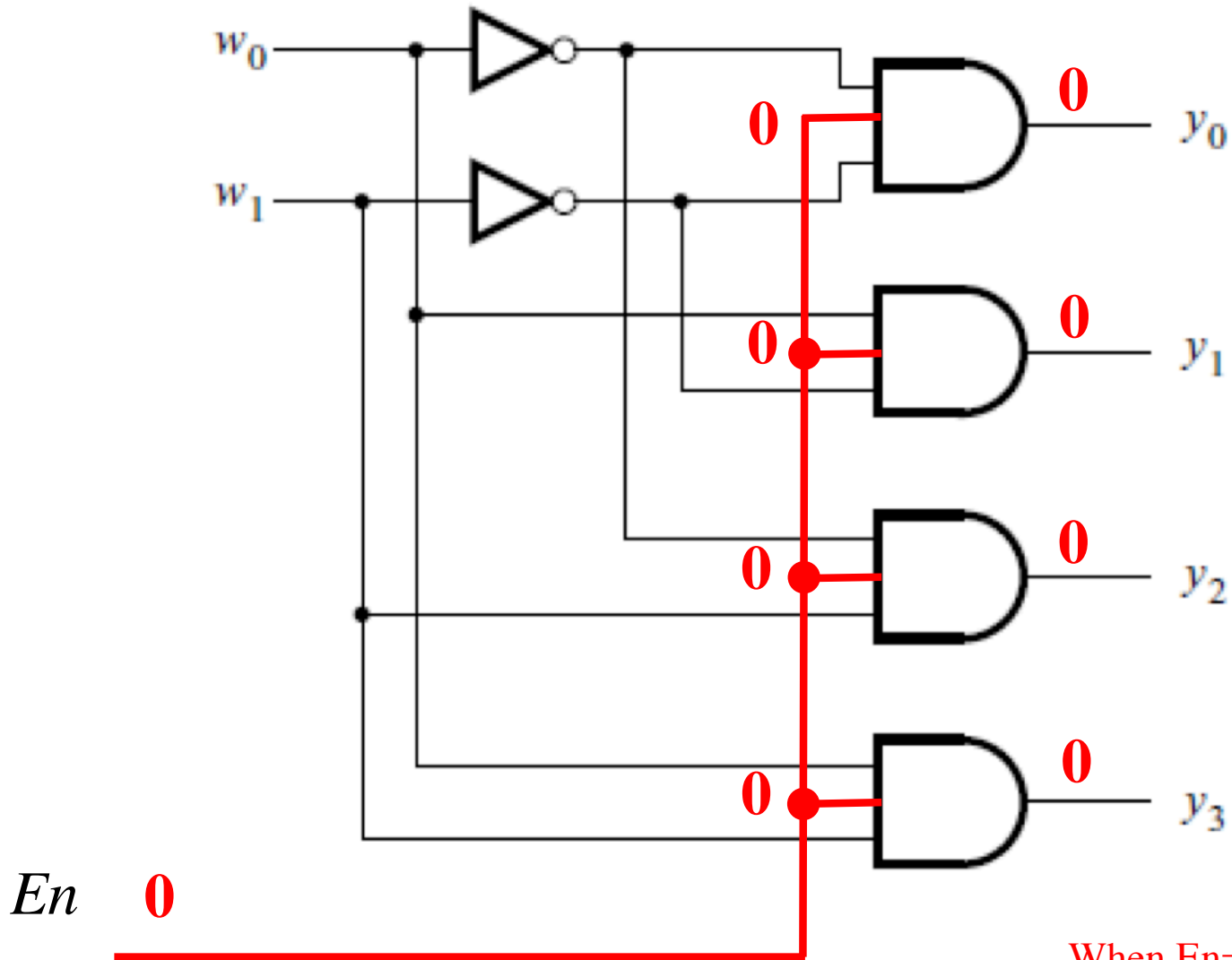


# Adding an Enable Input



When  $En=1$  this circuit behaves like the one without enable.

# Adding an Enable Input

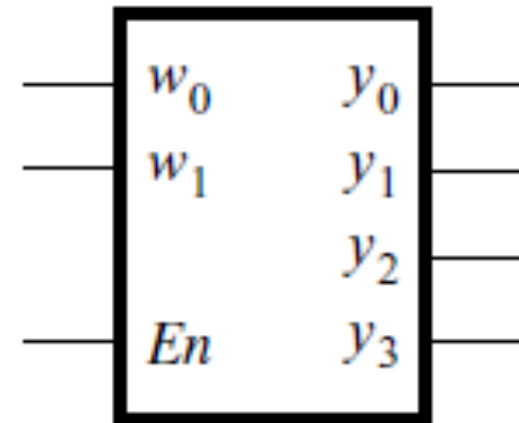


When  $En=0$  all outputs are set to 0 and the decoder is disabled.

# Truth Table and Graphical Symbol for a 2-to-4 Decoder with an Enable Input

$En$	$w_1$	$w_0$	$y_0$	$y_1$	$y_2$	$y_3$
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
0	x	x	0	0	0	0

(a) Truth table

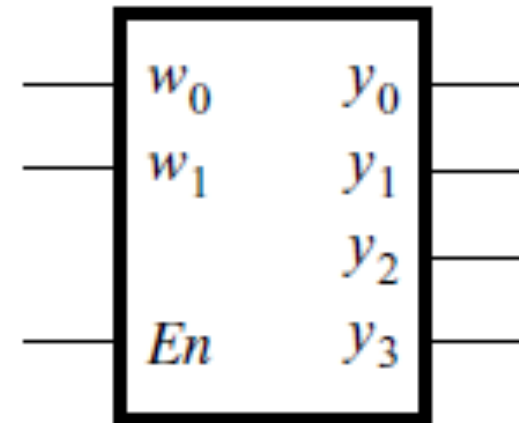


(b) Graphical symbol

# Truth Table and Graphical Symbol for a 2-to-4 Decoder with an Enable Input

$En$	$w_1$	$w_0$	$y_0$	$y_1$	$y_2$	$y_3$
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
0	x	x	0	0	0	0

(a) Truth table

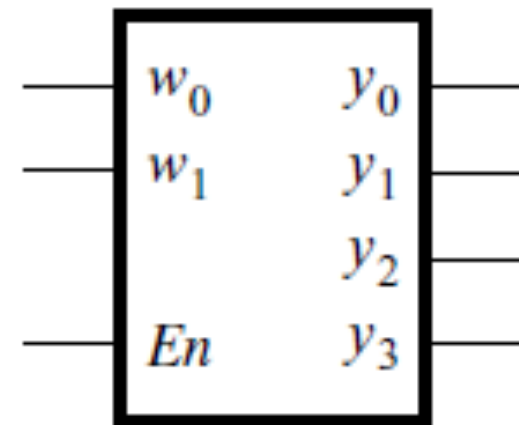


(b) Graphical symbol

x indicates that it does not matter what the value of this variable is for this row of the truth table

# Truth Table and Graphical Symbol for a 2-to-4 Decoder with an Enable Input

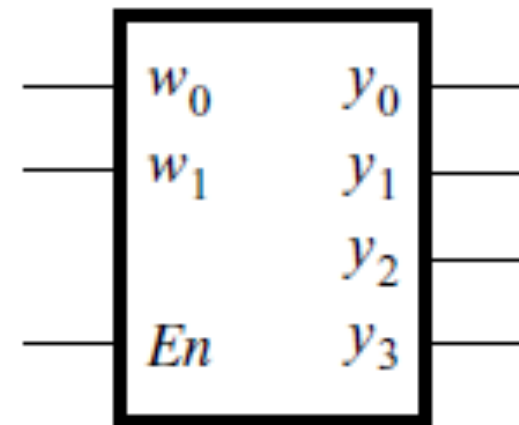
$En$	$w_1$	$w_0$	$y_0$	$y_1$	$y_2$	$y_3$
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0



(b) Graphical symbol

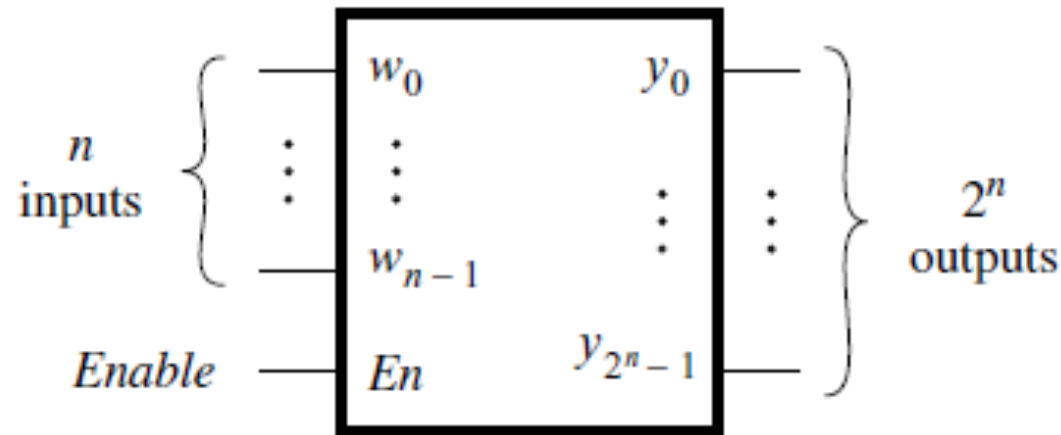
# Truth Table and Graphical Symbol for a 2-to-4 Decoder with an Enable Input

$En$	$w_1$	$w_0$	$y_0$	$y_1$	$y_2$	$y_3$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	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



(b) Graphical symbol

# Graphical Symbol for a Binary $n$ -to- $2^n$ Decoder with an Enable Input



(d) An  $n$ -to- $2^n$  decoder

A binary decoder with  $n$  inputs has  $2^n$  outputs

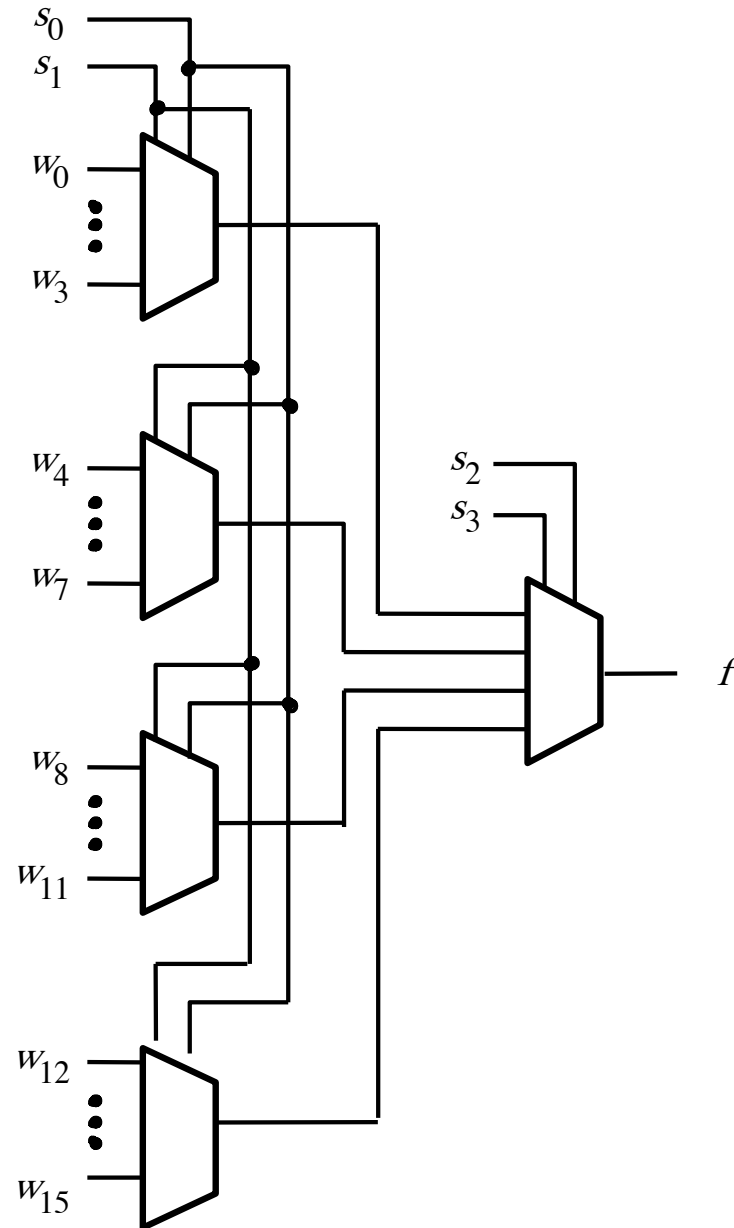
The outputs of an enabled binary decoder are “one-hot” encoded, meaning that only a single bit is set to 1, i.e., it is *hot*.

# How can we build larger decoders?

- 3-to-8 ?
- 4-to-16?
- 5-to-??

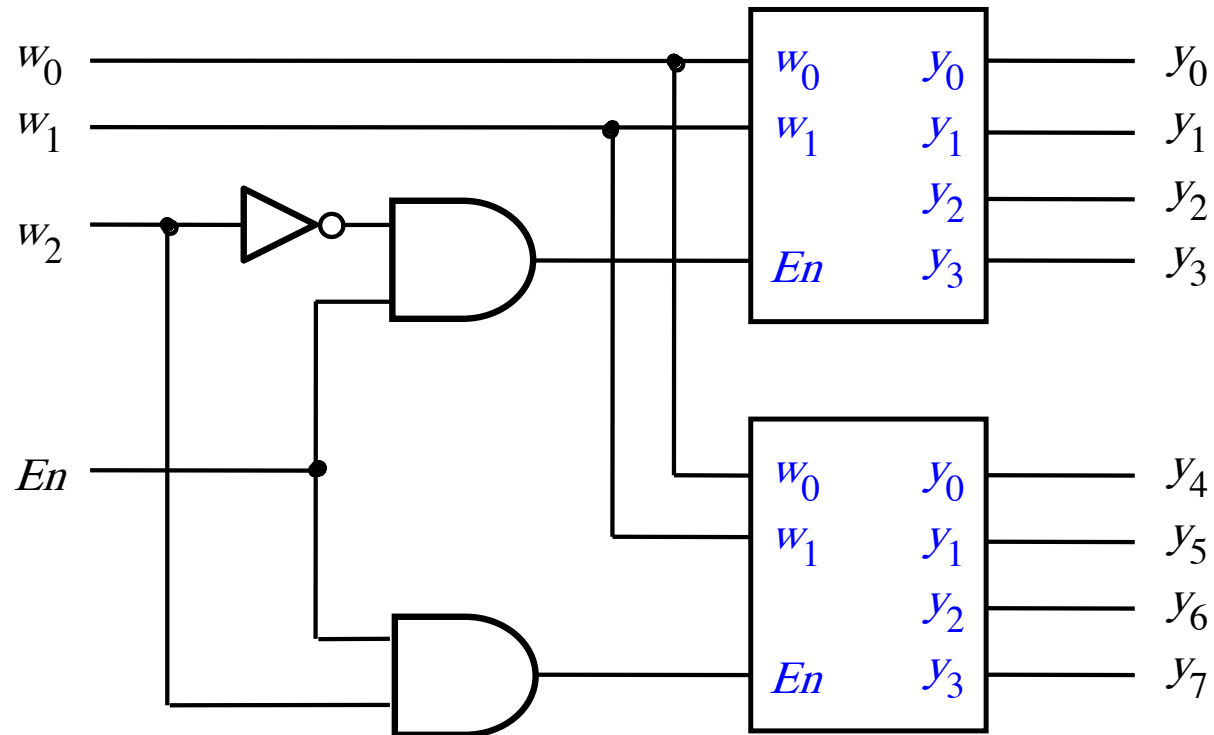


# Hint: How did we build a 16-1 Multiplexer

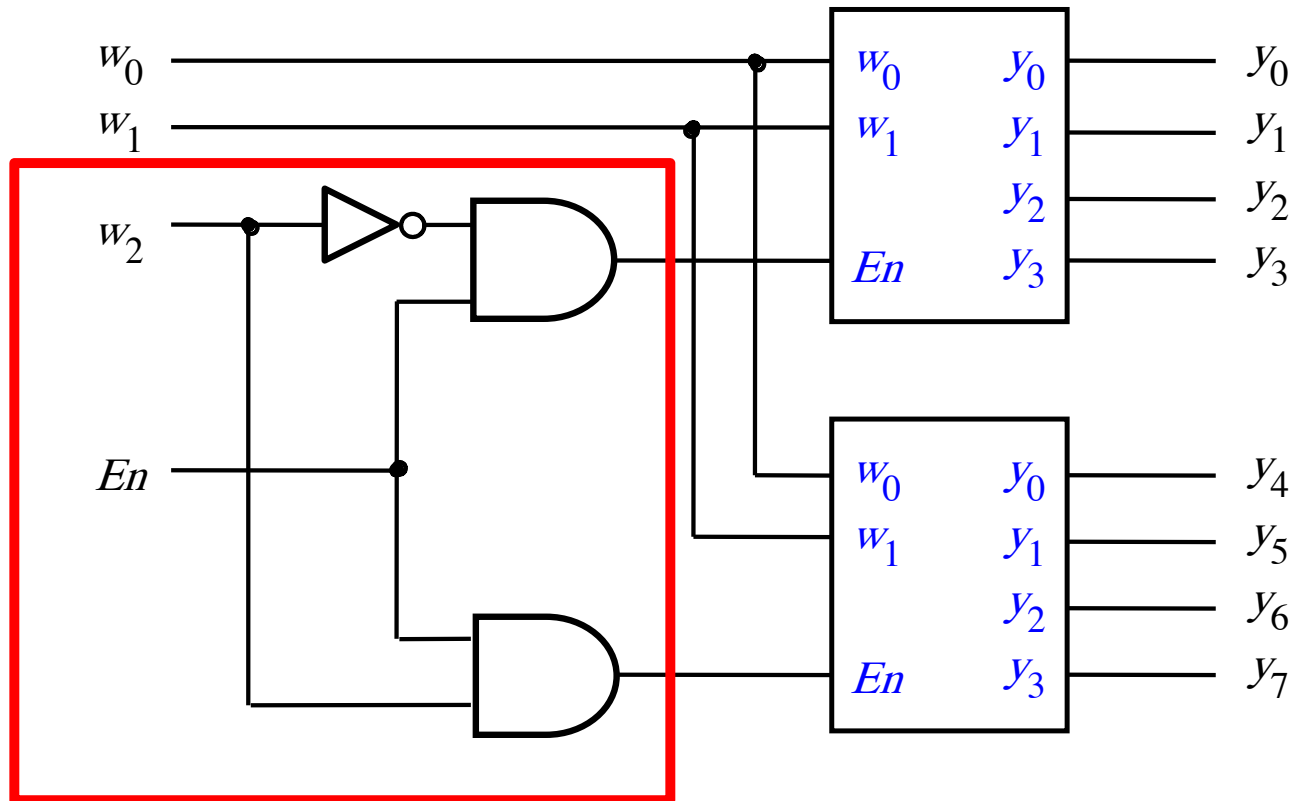


[ Figure 4.4 from the textbook ]

# A 3-to-8 decoder using two 2-to-4 decoders

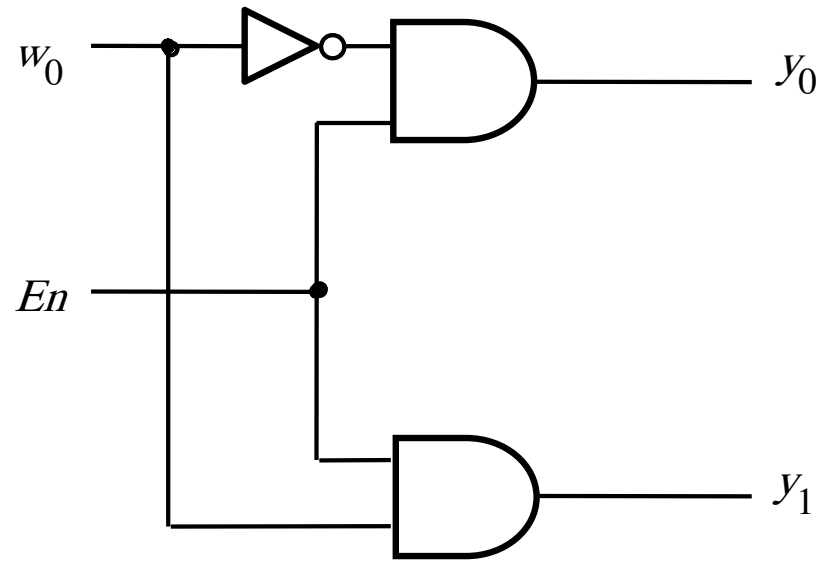


# A 3-to-8 decoder using two 2-to-4 decoders

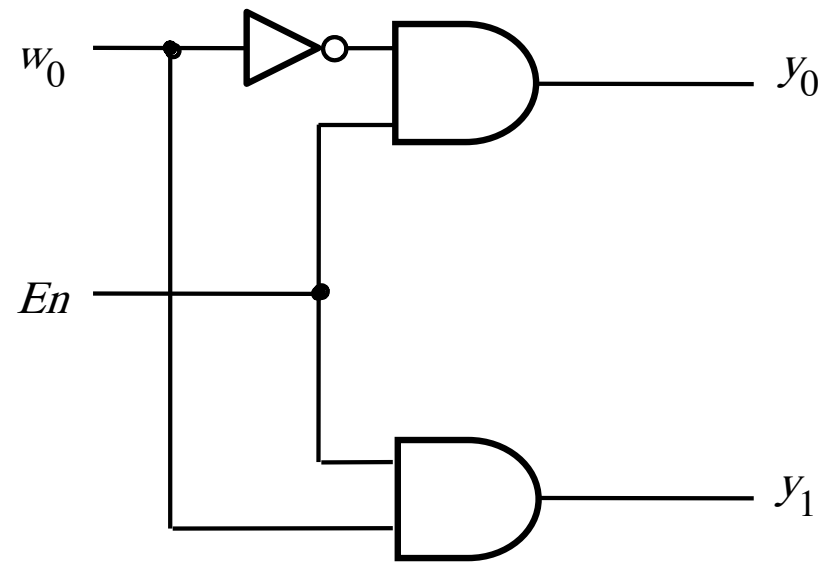


What is this?

# What is this?

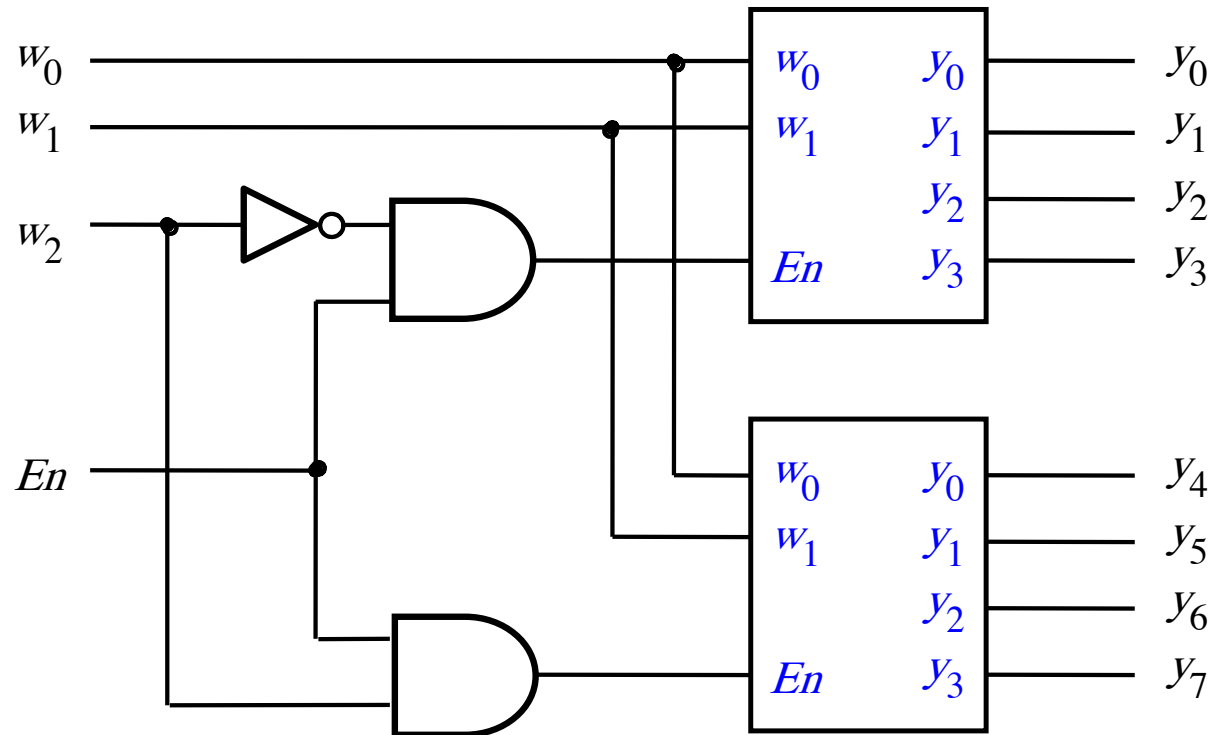


# What is this?

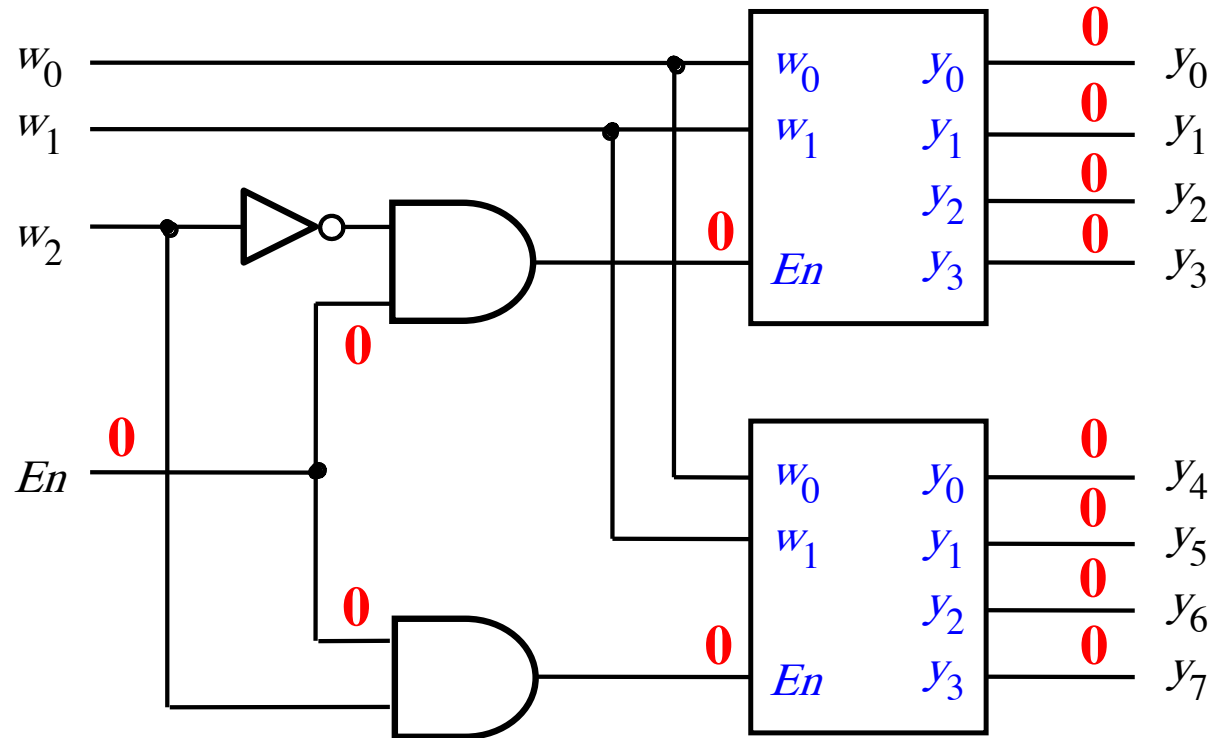


This is a 1-to-2 decoder with an enable input.

# A 3-to-8 decoder using two 2-to-4 decoders

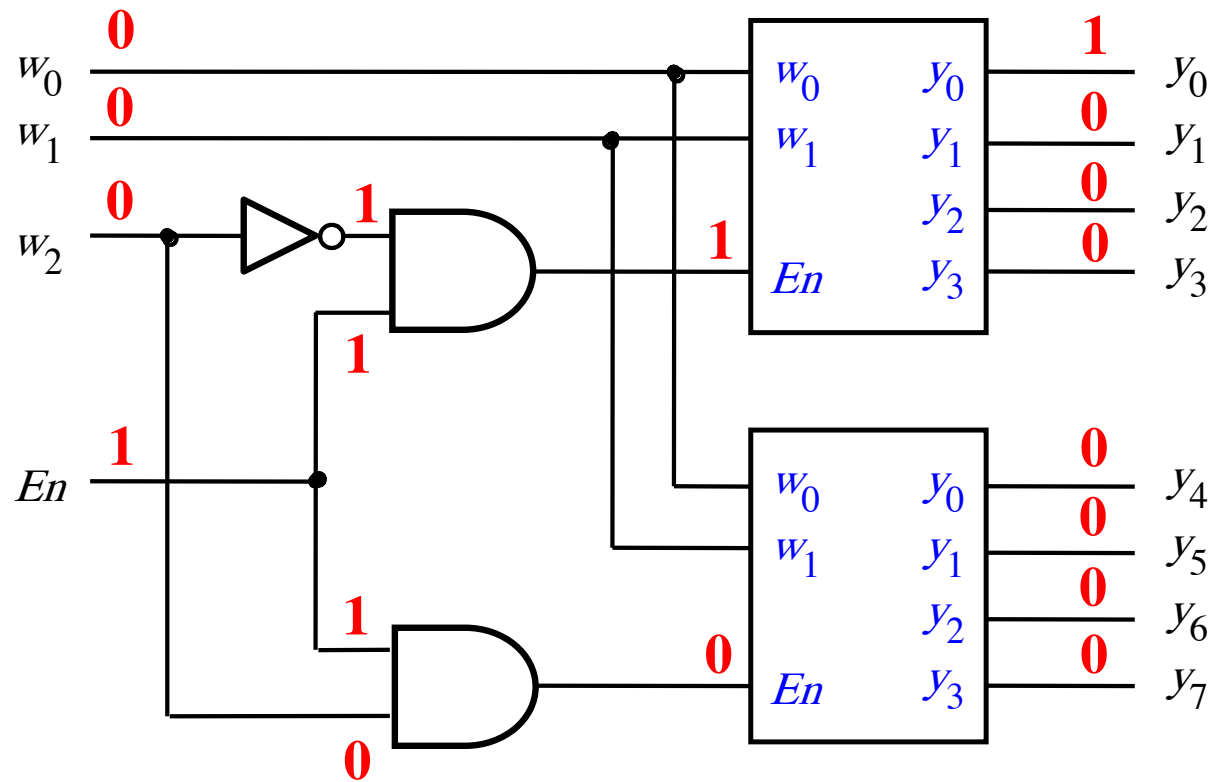


# A 3-to-8 decoder using two 2-to-4 decoders



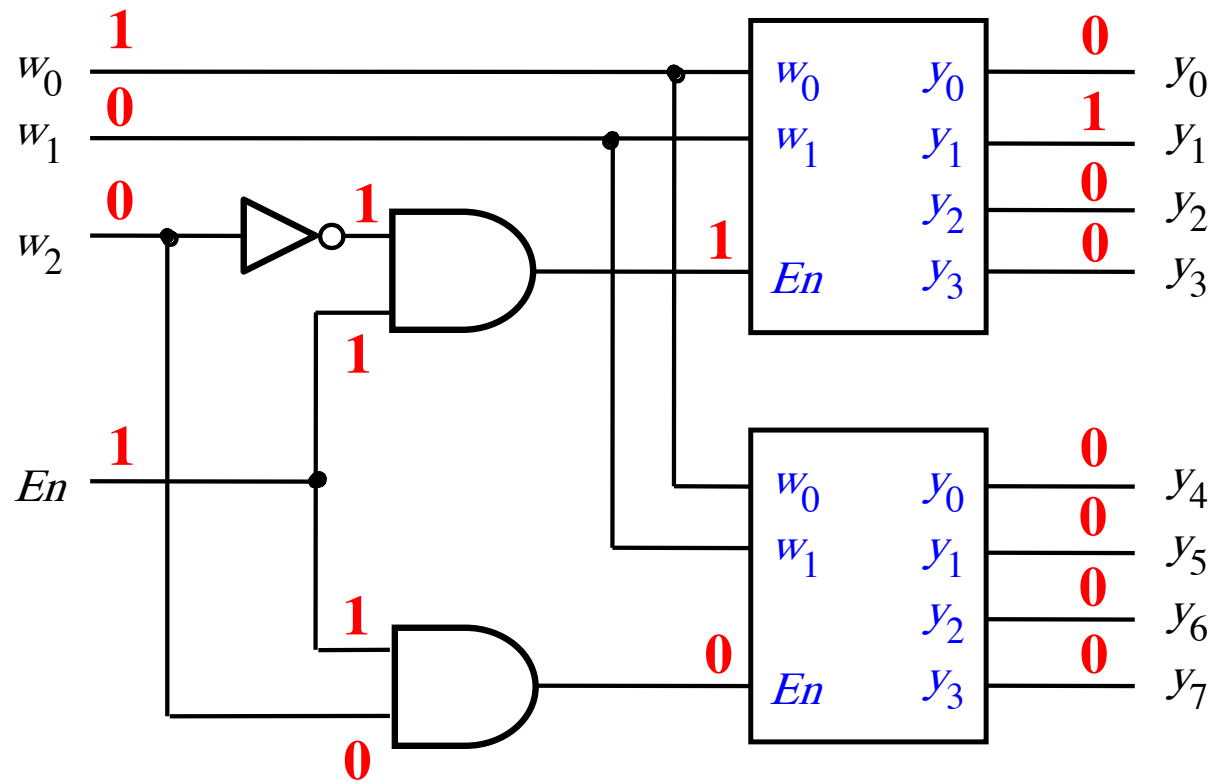
[ Figure 4.15 from the textbook ]

# A 3-to-8 decoder using two 2-to-4 decoders

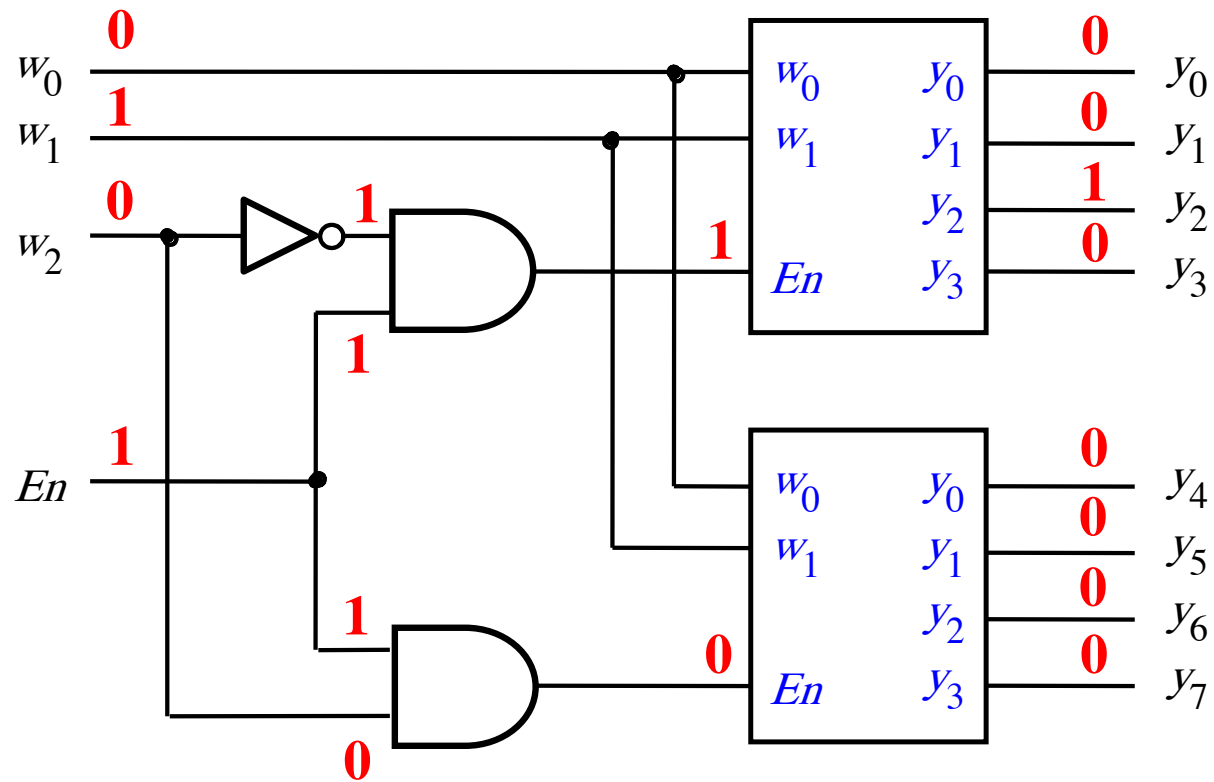




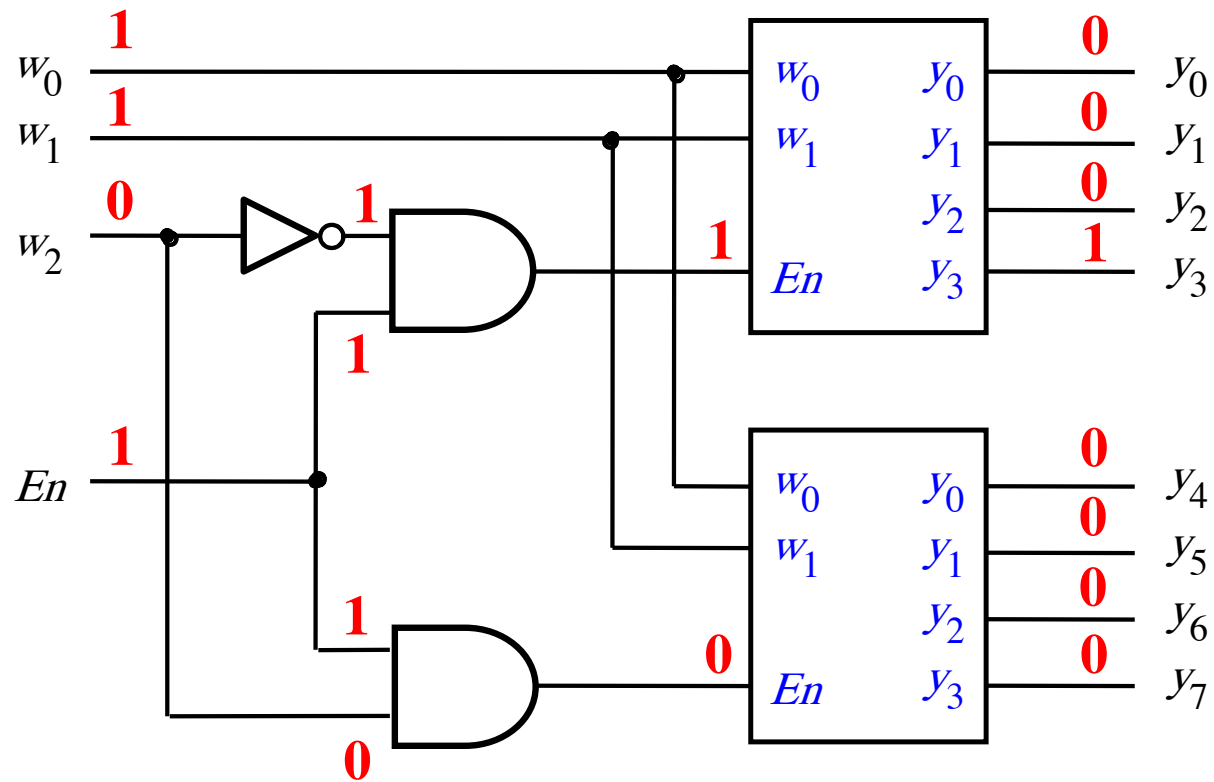
# A 3-to-8 decoder using two 2-to-4 decoders



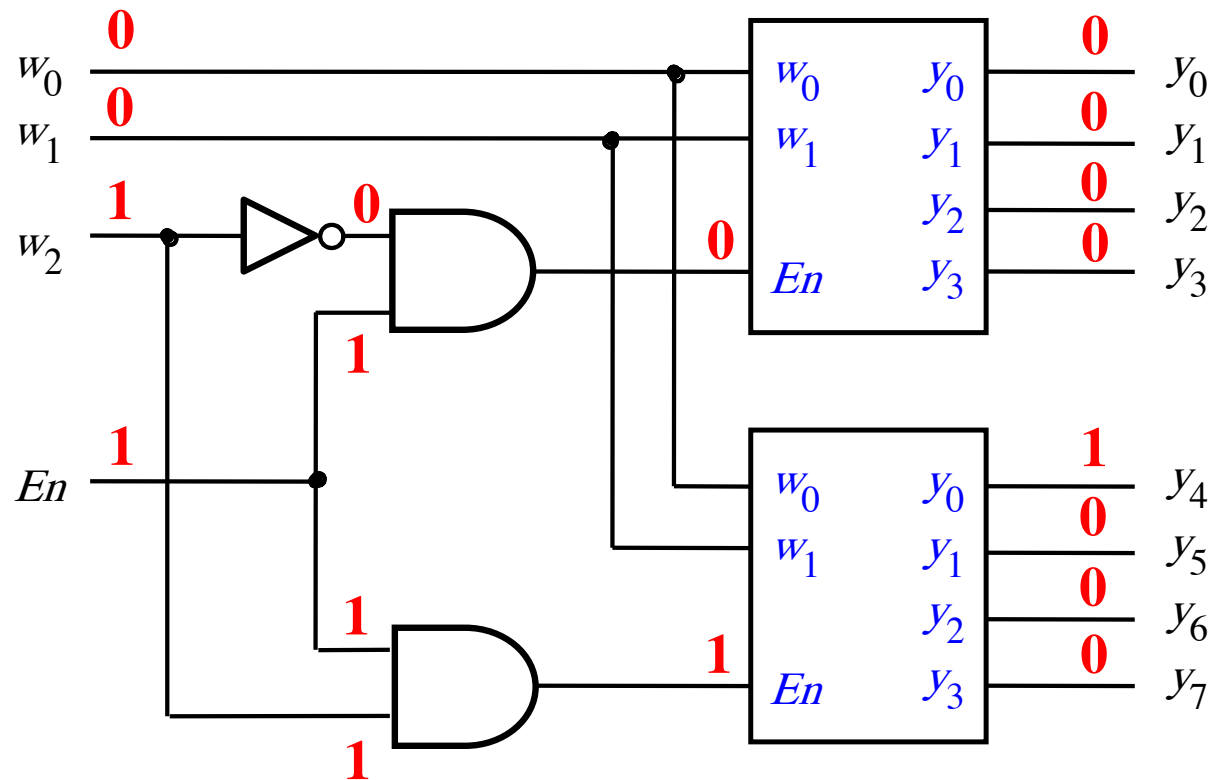
# A 3-to-8 decoder using two 2-to-4 decoders



# A 3-to-8 decoder using two 2-to-4 decoders

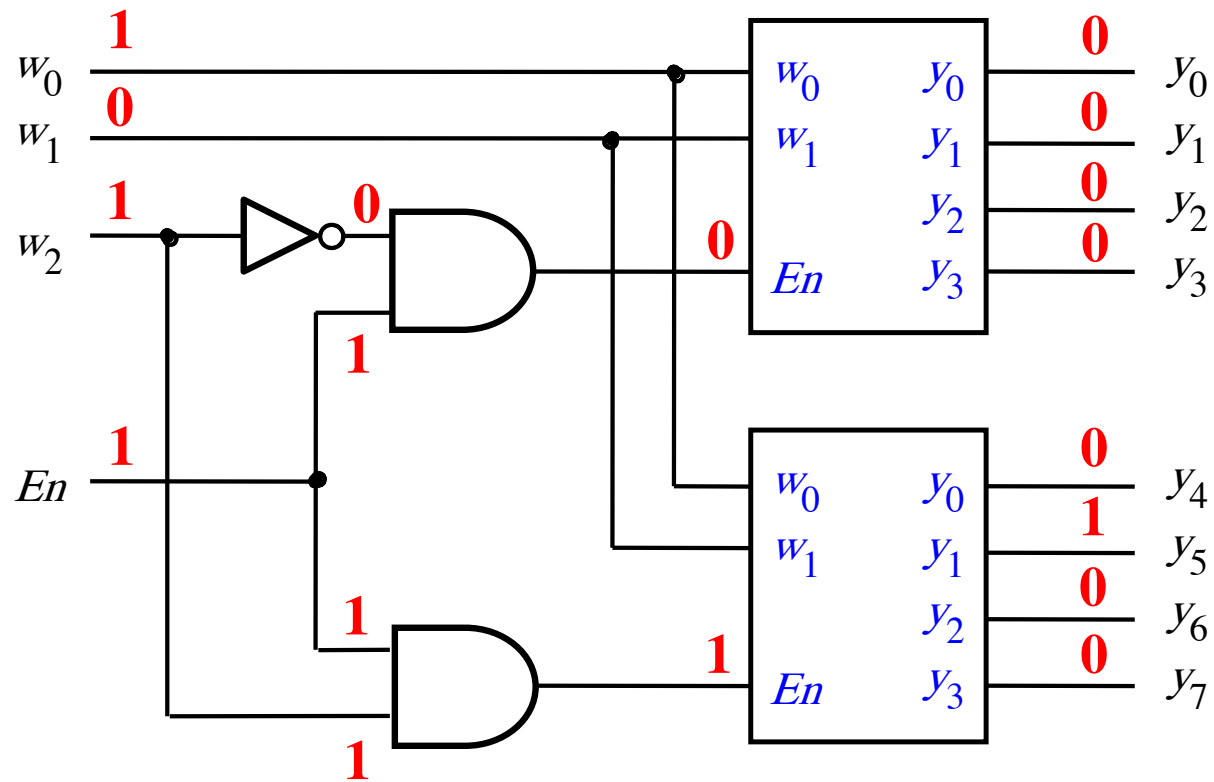


# A 3-to-8 decoder using two 2-to-4 decoders

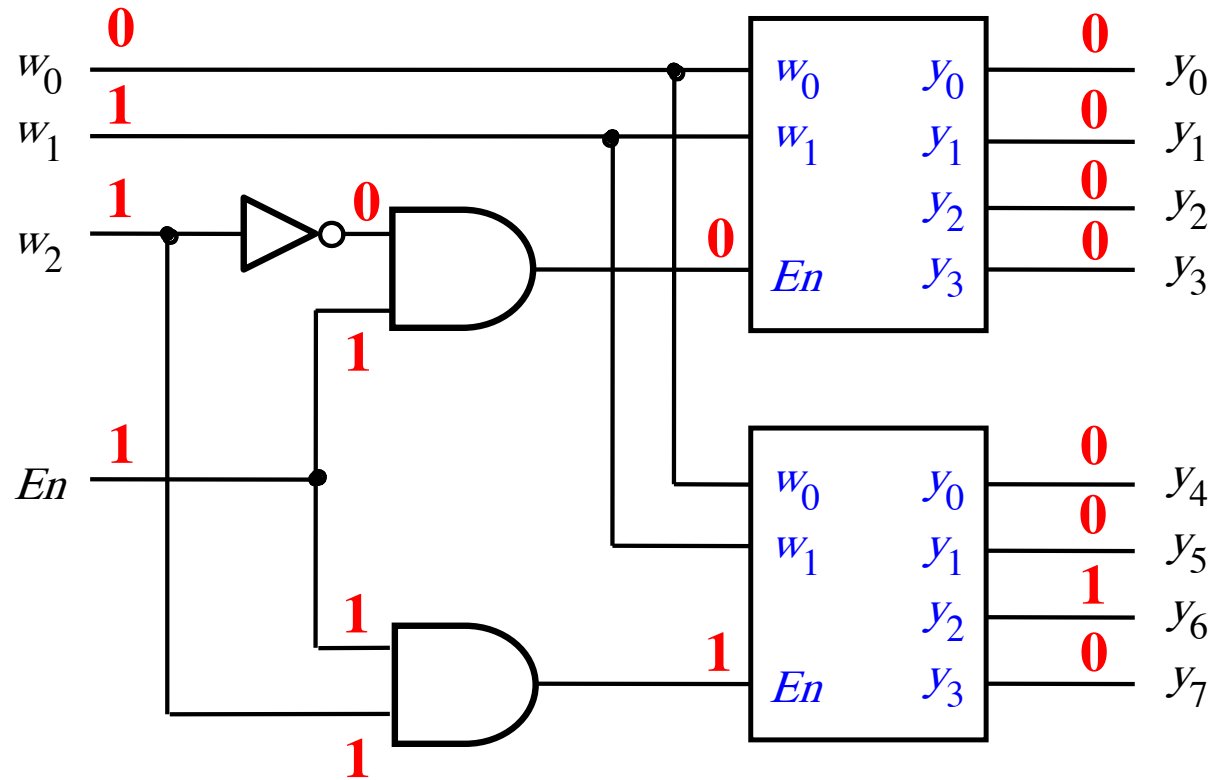


[ Figure 4.15 from the textbook ]

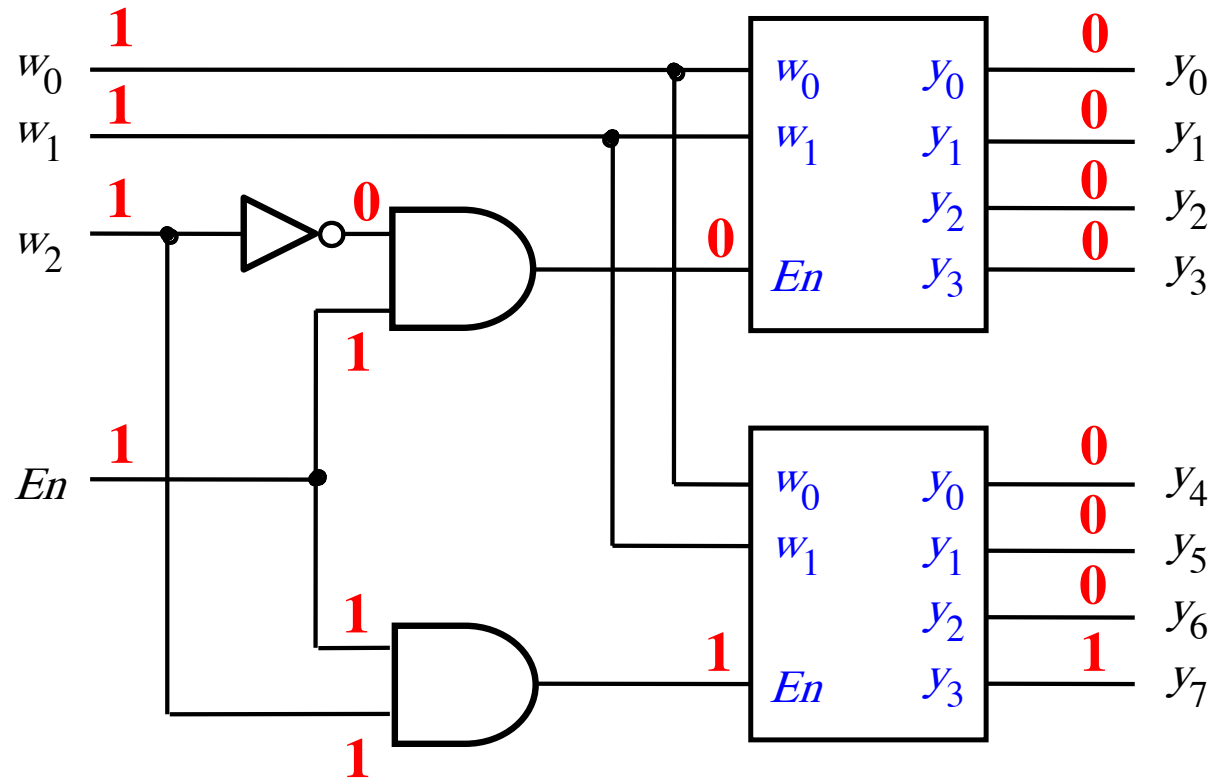
# A 3-to-8 decoder using two 2-to-4 decoders



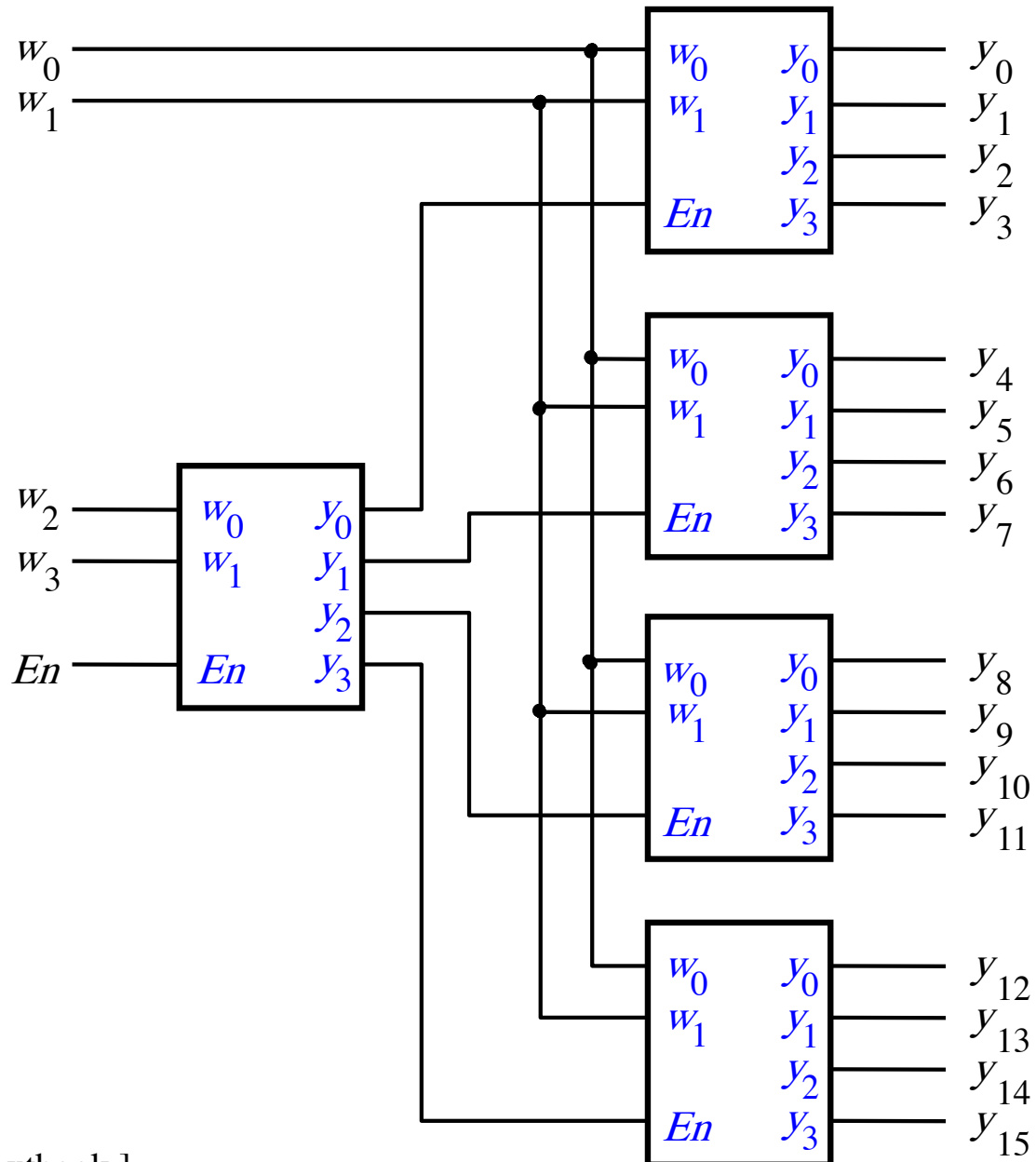
# A 3-to-8 decoder using two 2-to-4 decoders



# A 3-to-8 decoder using two 2-to-4 decoders



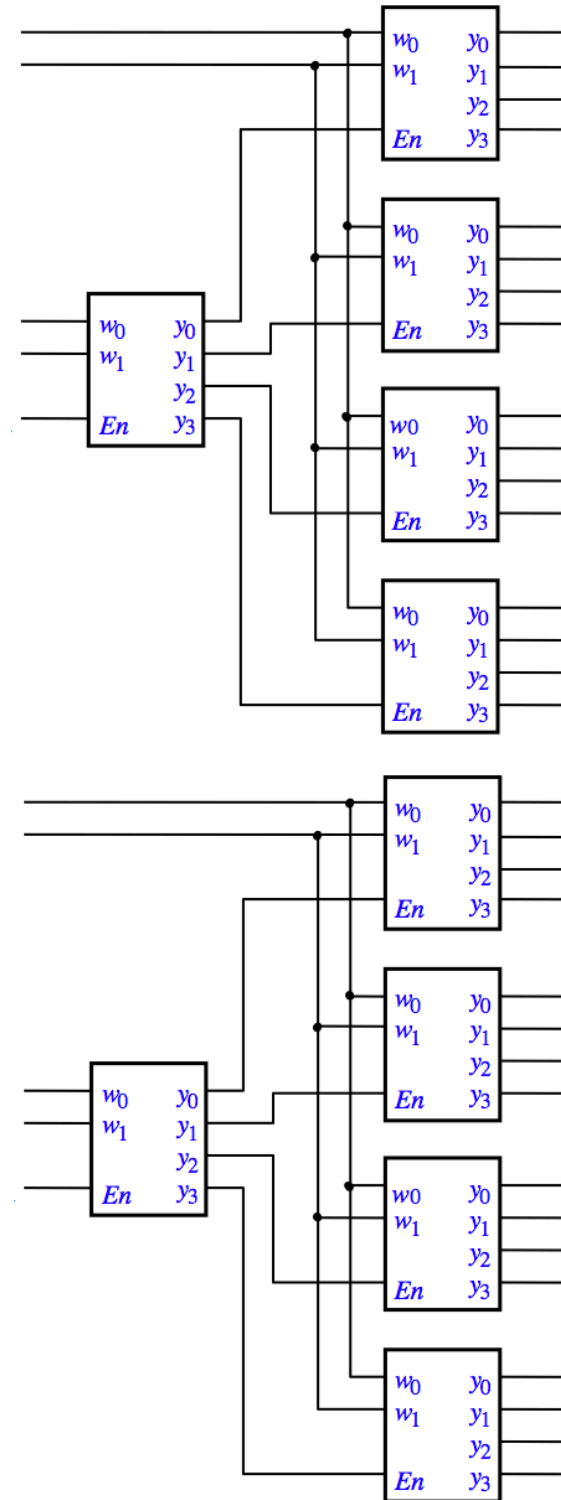
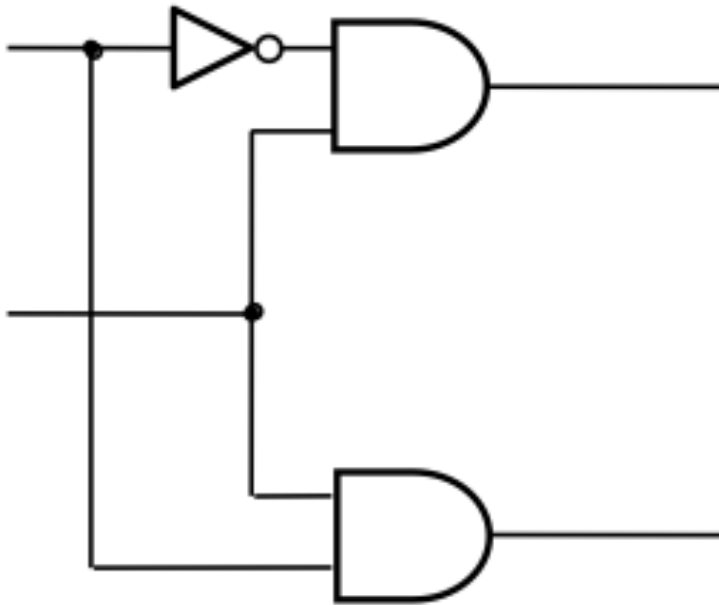
# 4-to-16 decoder built using a decoder tree



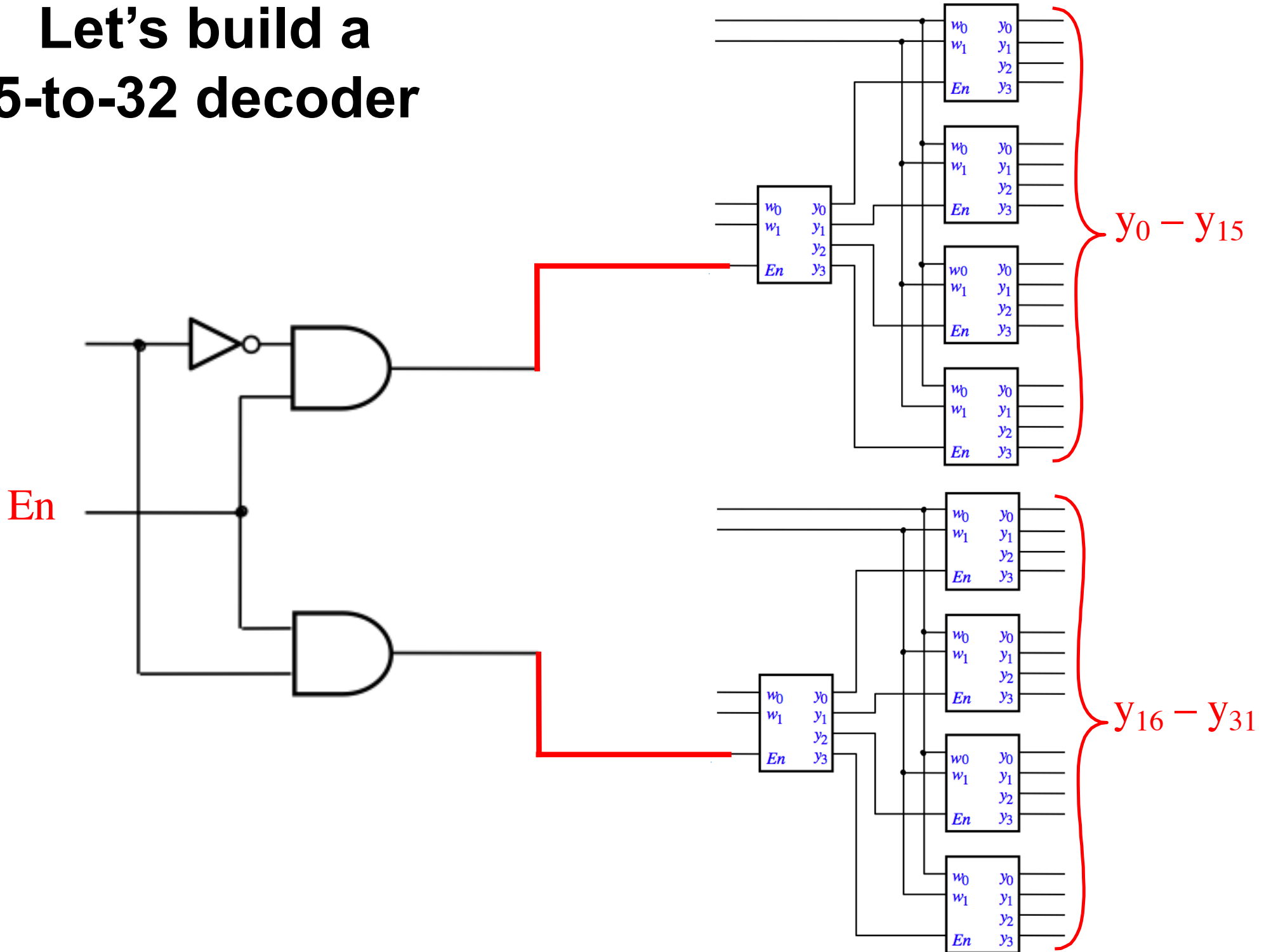
[ Figure 4.16 from the textbook ]



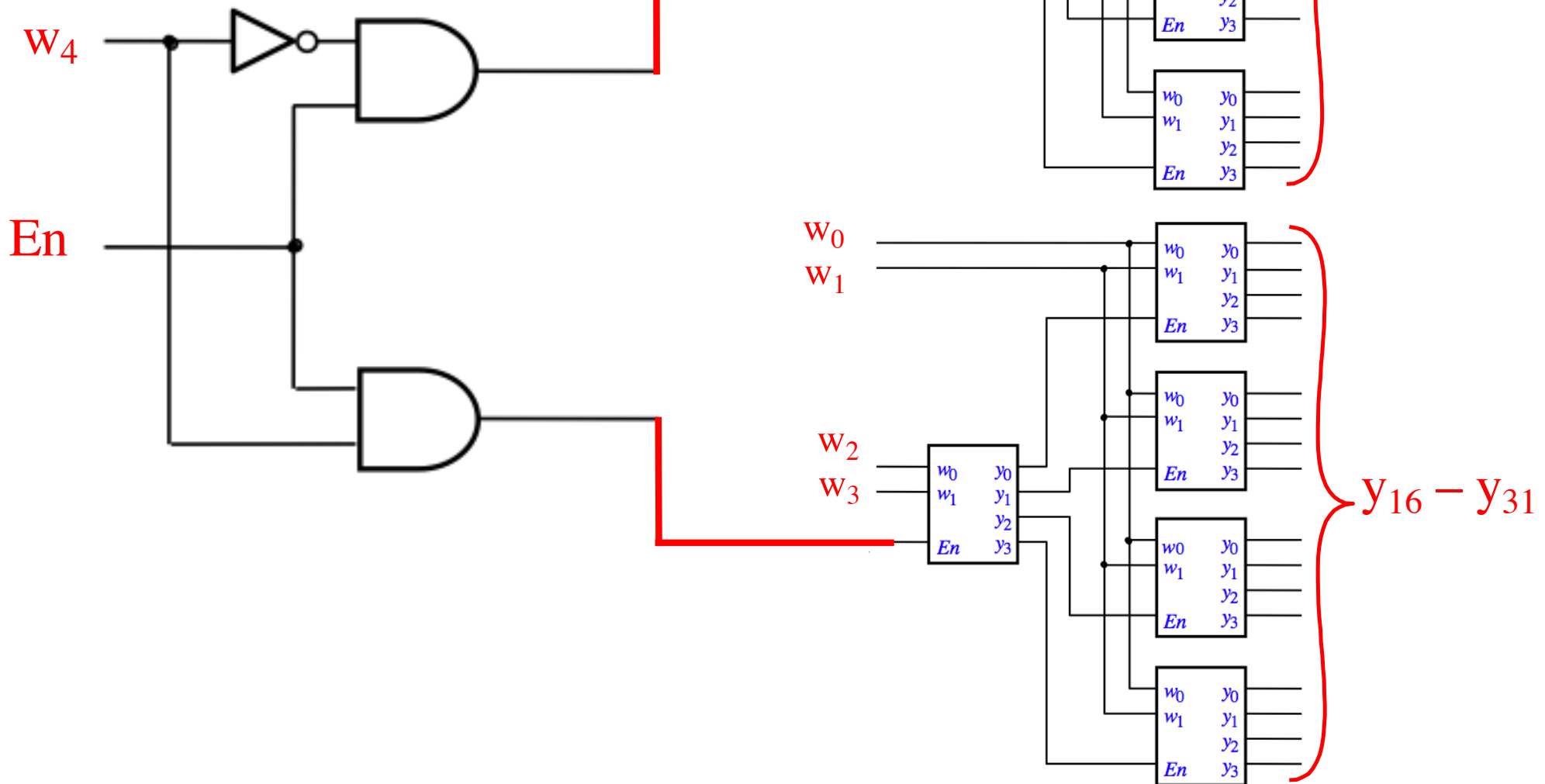
# Let's build a 5-to-32 decoder



# Let's build a 5-to-32 decoder



# Let's build a 5-to-32 decoder

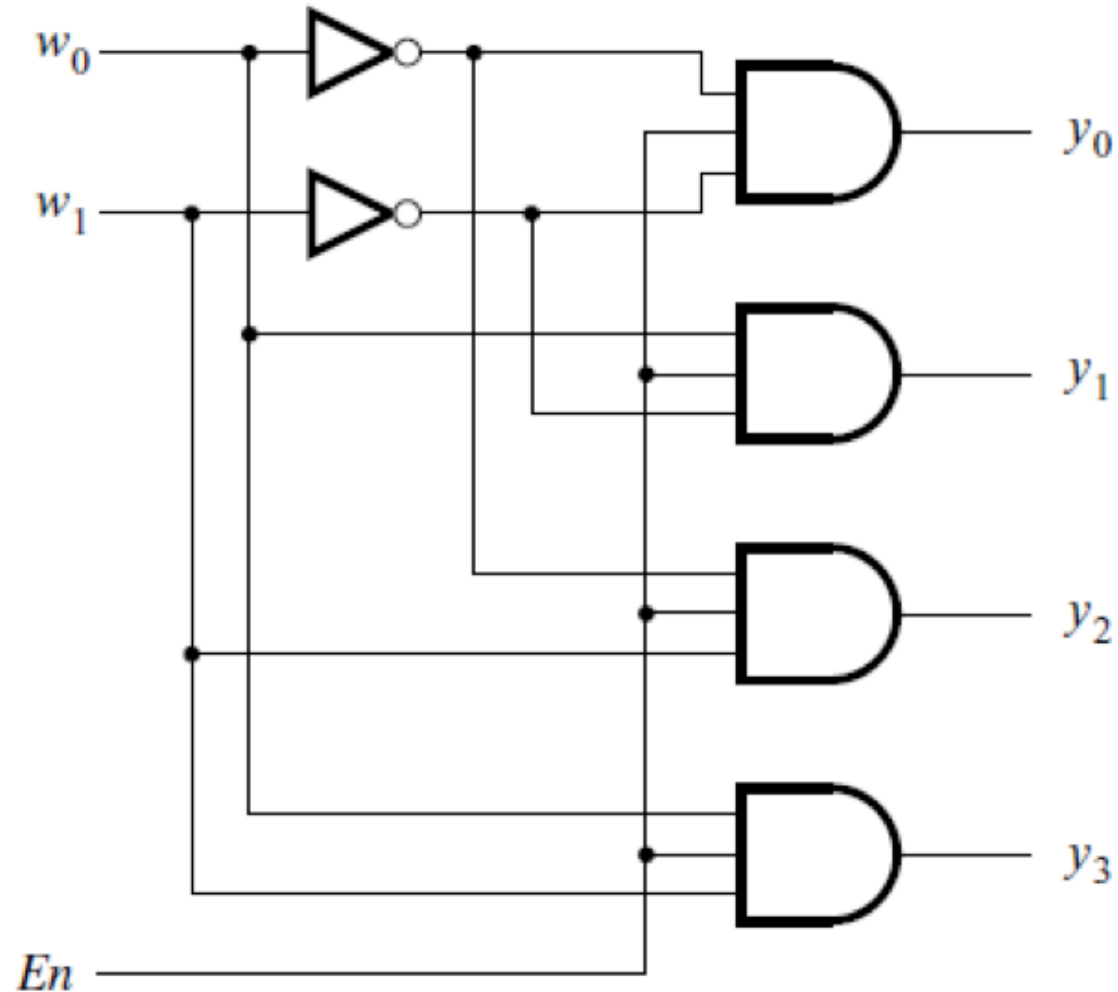


# Demultiplexers

# 1-to-4 Demultiplexer (Definition)

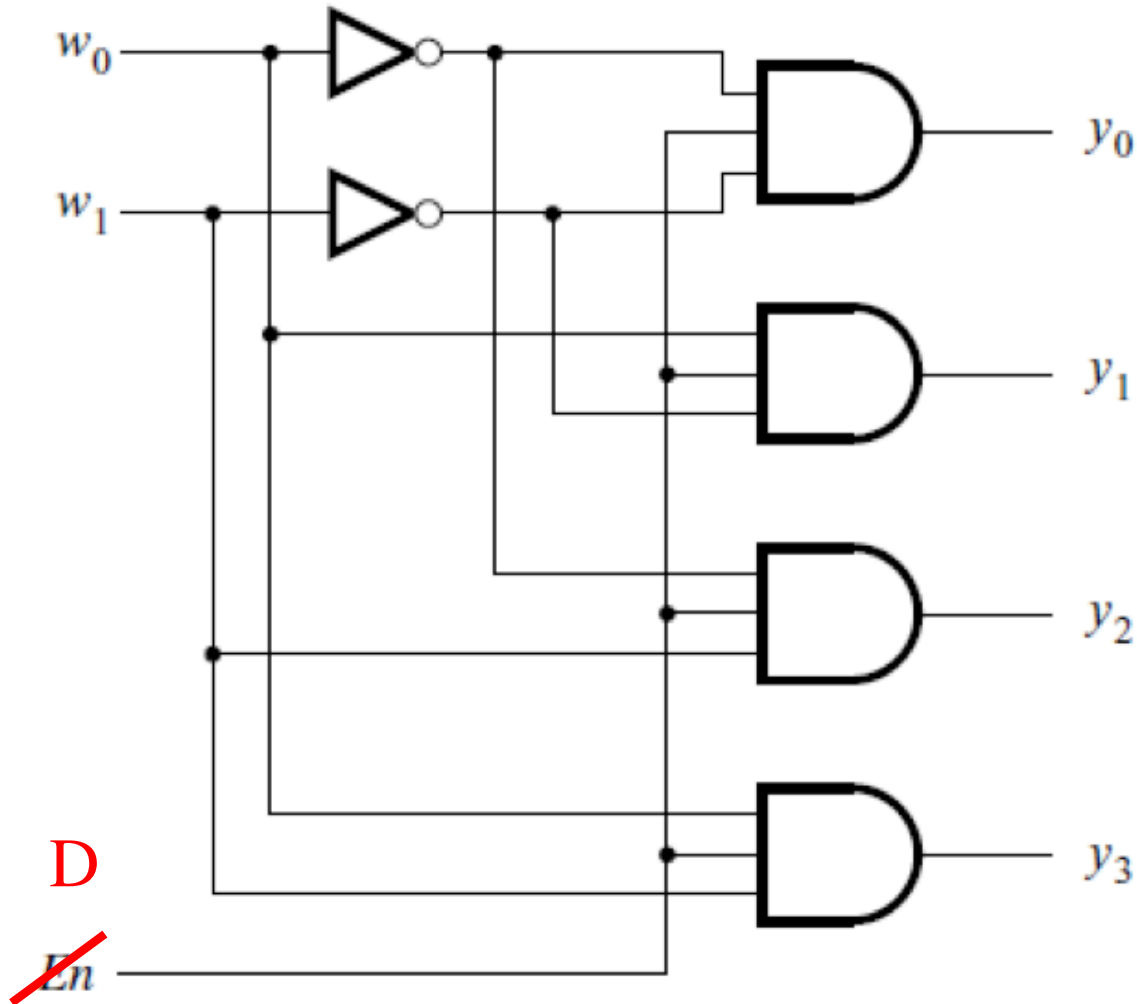
- Has one data input line:  $D$
- Has two output select lines:  $w_1$  and  $w_0$
- Has four outputs:  $y_0$ ,  $y_1$ ,  $y_2$ , and  $y_3$
- If  $w_1=0$  and  $w_0=0$ , then the output  $y_0$  is set to  $D$
- If  $w_1=0$  and  $w_0=1$ , then the output  $y_1$  is set to  $D$
- If  $w_1=1$  and  $w_0=0$ , then the output  $y_2$  is set to  $D$
- If  $w_1=1$  and  $w_0=1$ , then the output  $y_3$  is set to  $D$
- Only one output is set to  $D$ . All others are set to 0.

# A 1-to-4 demultiplexer built with a 2-to-4 decoder



# A 1-to-4 demultiplexer built with a 2-to-4 decoder

output  
select  
lines



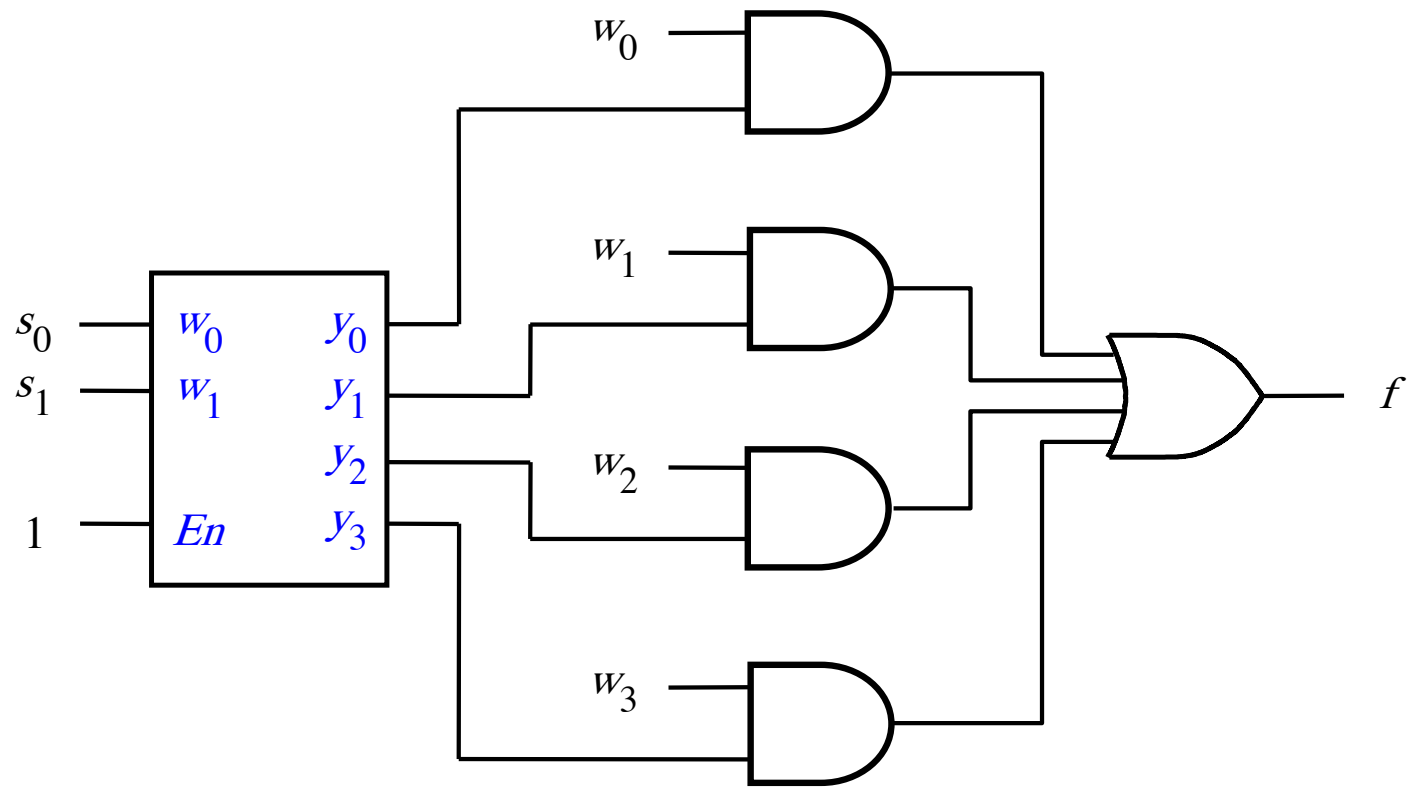
the  
four  
output  
lines

data  
input  
line

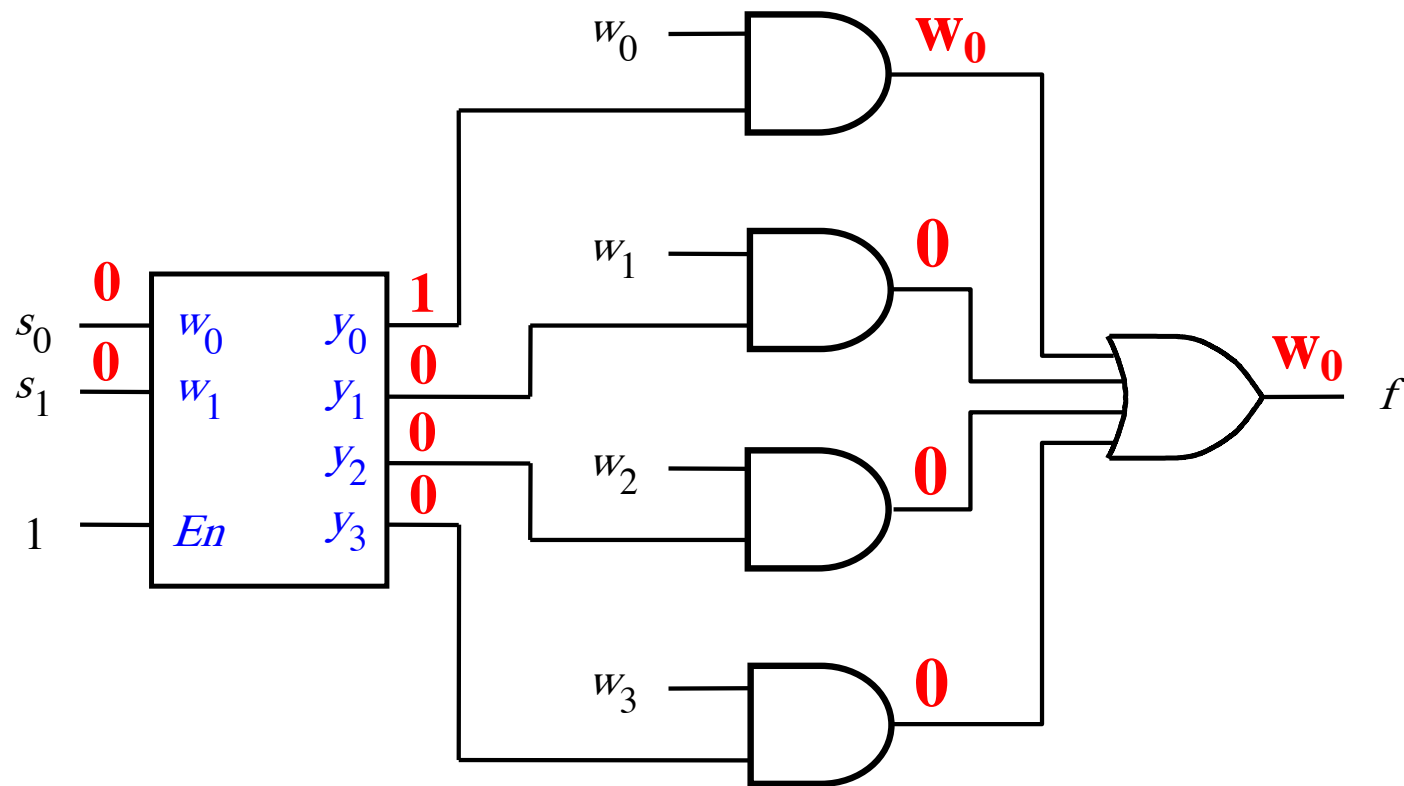
# **Multiplexers (Implemented with Decoders)**



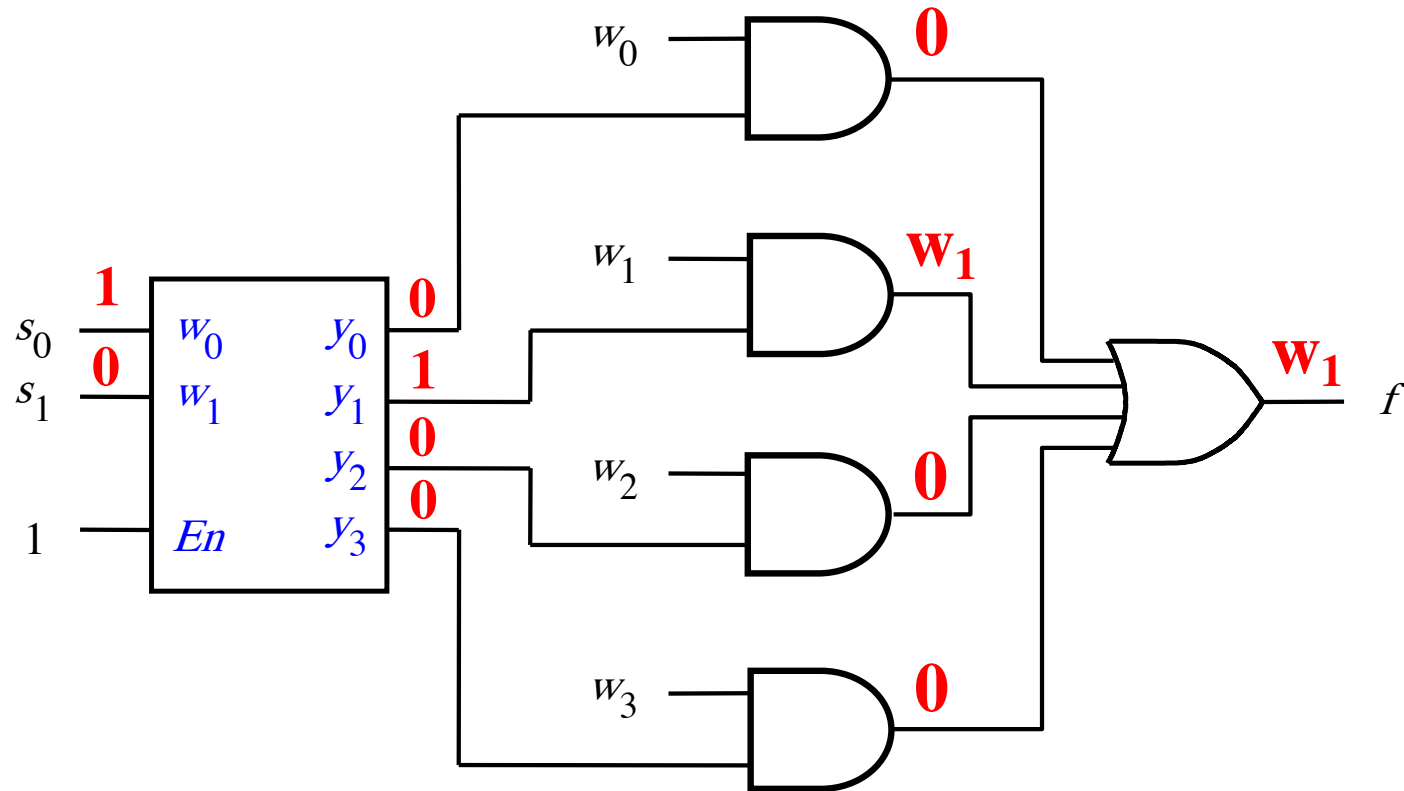
# A 4-to-1 multiplexer built using a 2-to-4 decoder



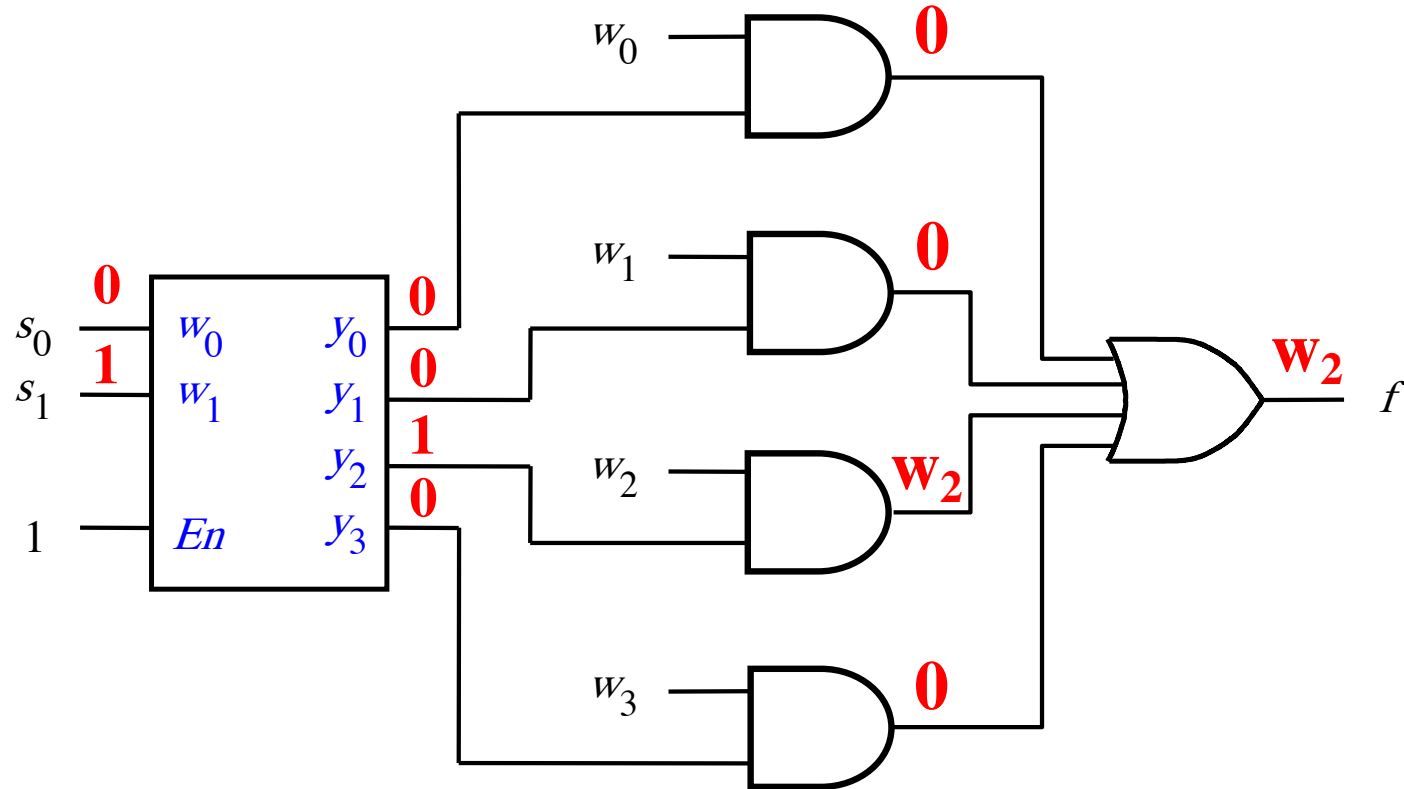
# A 4-to-1 multiplexer built using a 2-to-4 decoder



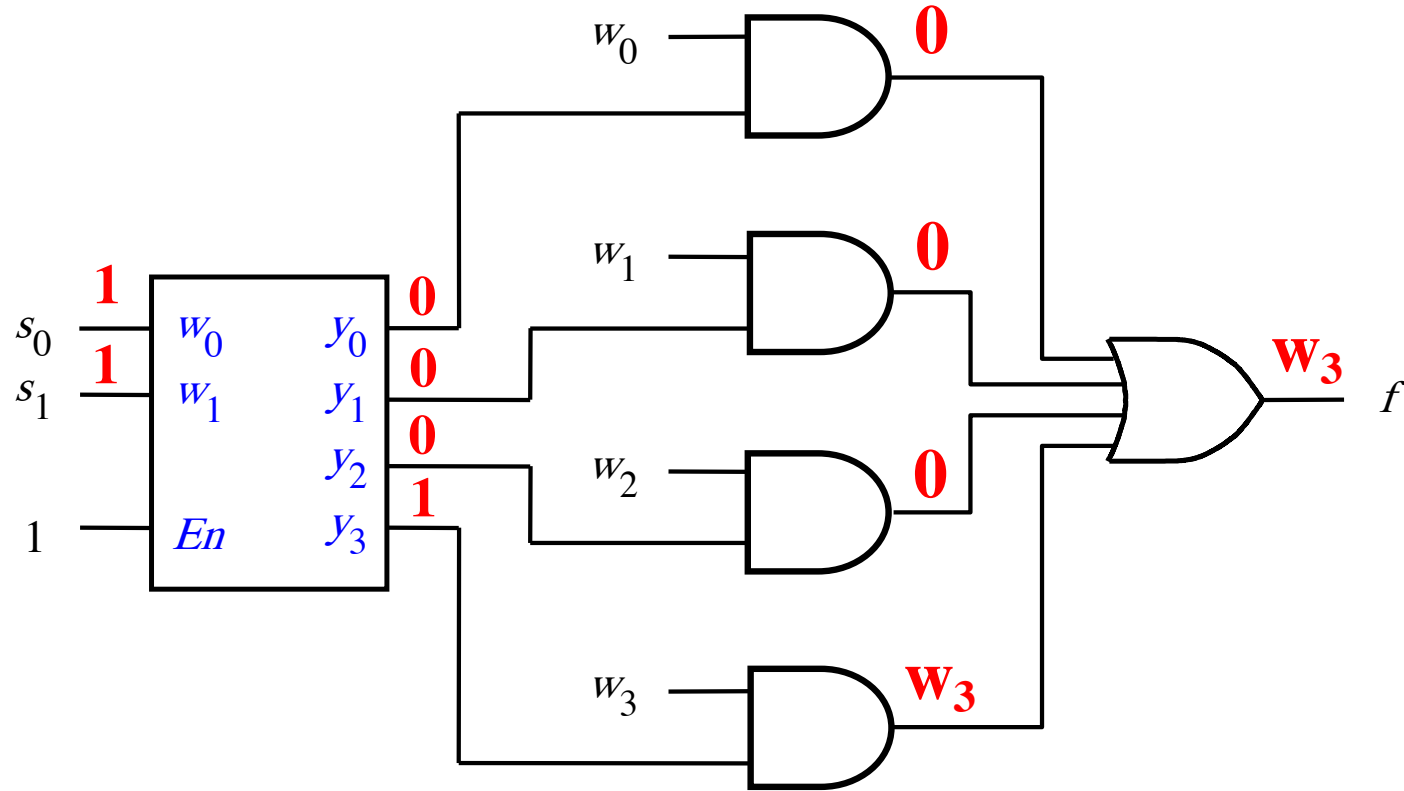
# A 4-to-1 multiplexer built using a 2-to-4 decoder



# A 4-to-1 multiplexer built using a 2-to-4 decoder

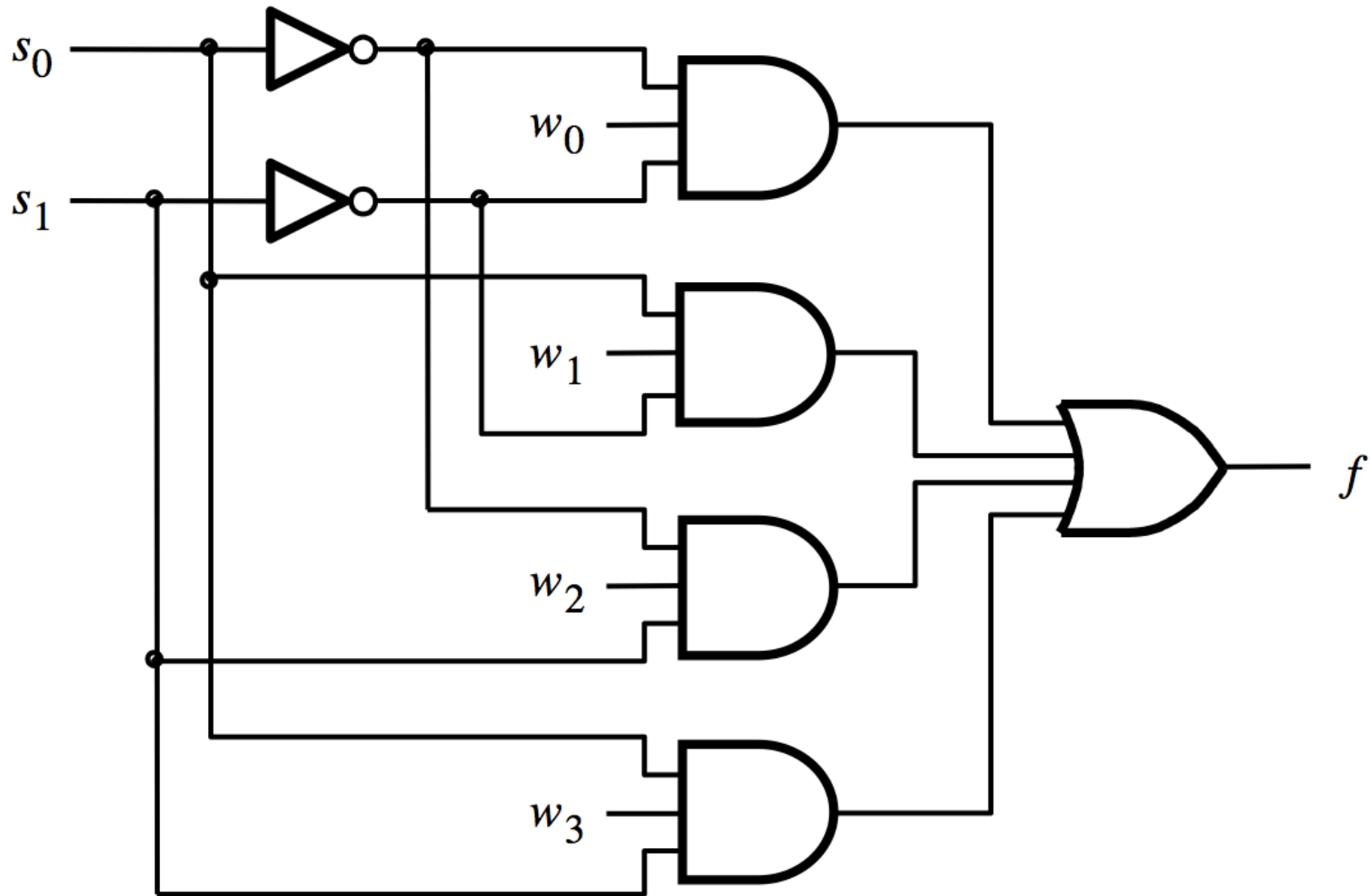


# A 4-to-1 multiplexer built using a 2-to-4 decoder



[ Figure 4.17 from the textbook ]

# 4-to-1 Multiplexer (SOP circuit)



$$f = \overline{s_1} \overline{s_0} w_0 + \overline{s_1} s_0 w_1 + s_1 \overline{s_0} w_2 + s_1 s_0 w_3$$

# Encoders

**Encoders**  
**(there are several types)**



# Binary Encoders

# 4-to-2 Binary Encoder (Definition)

- Has four inputs:  $w_3$  ,  $w_2$  ,  $w_1$  , and  $w_0$
- Has two outputs:  $y_1$  and  $y_0$
- Only one input is set to 1 (“one-hot” encoded). All others are set to 0.
- If  $w_0=1$  then  $y_1=0$  and  $y_0=0$
- If  $w_1=1$  then  $y_1=0$  and  $y_0=1$
- If  $w_2=1$  then  $y_1=1$  and  $y_0=0$
- If  $w_3=1$  then  $y_1=1$  and  $y_0=1$

# Truth table for a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

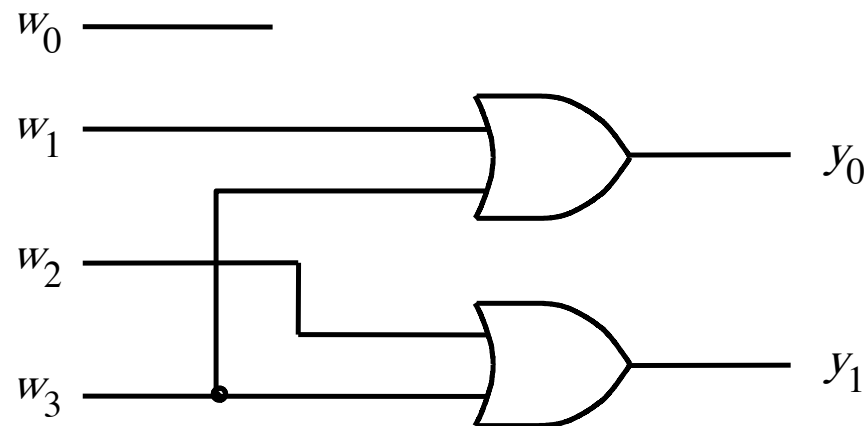
# Truth table for a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

The inputs are “one-hot” encoded

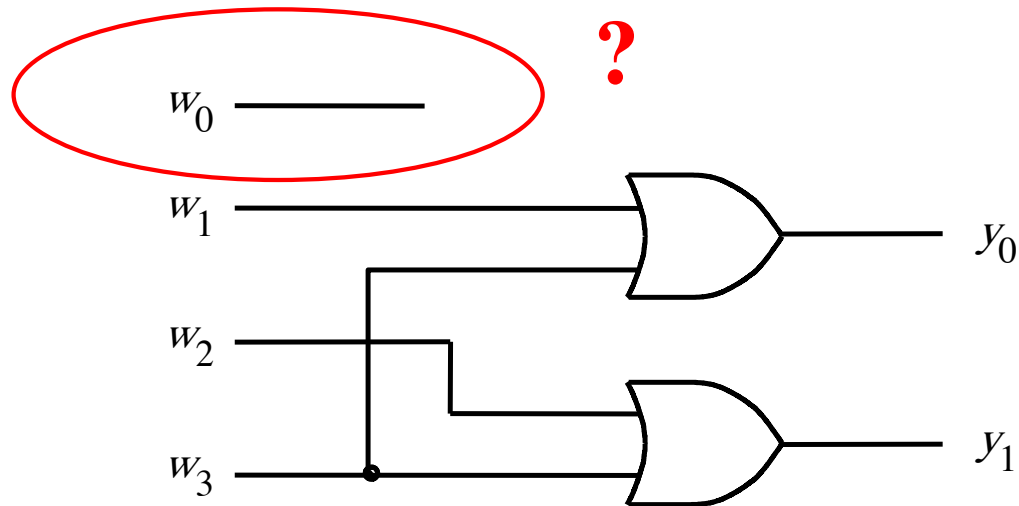
# Circuit for a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



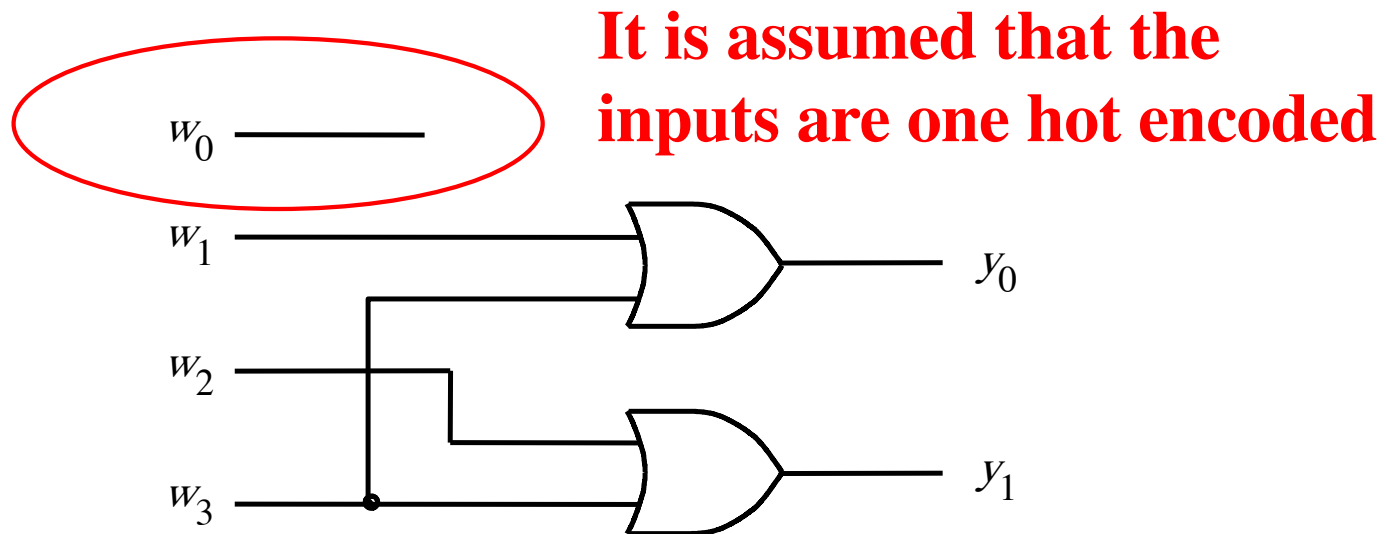
# Circuit for a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



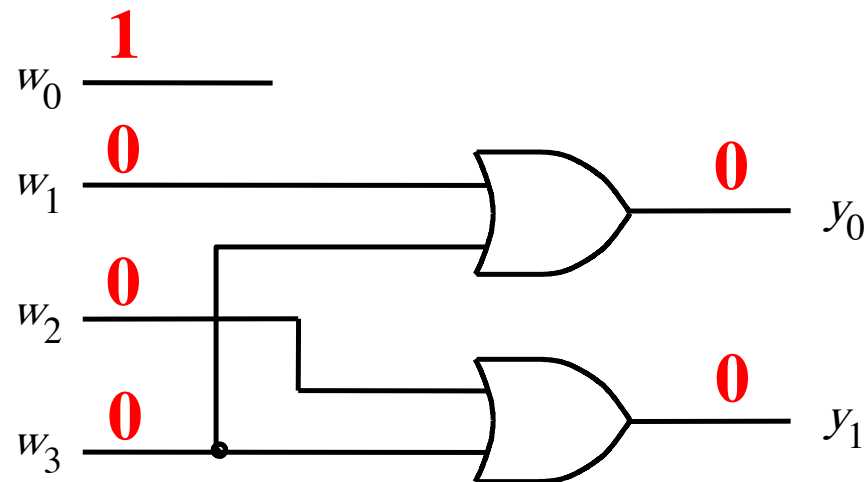
# Circuit for a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



# Circuit for a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

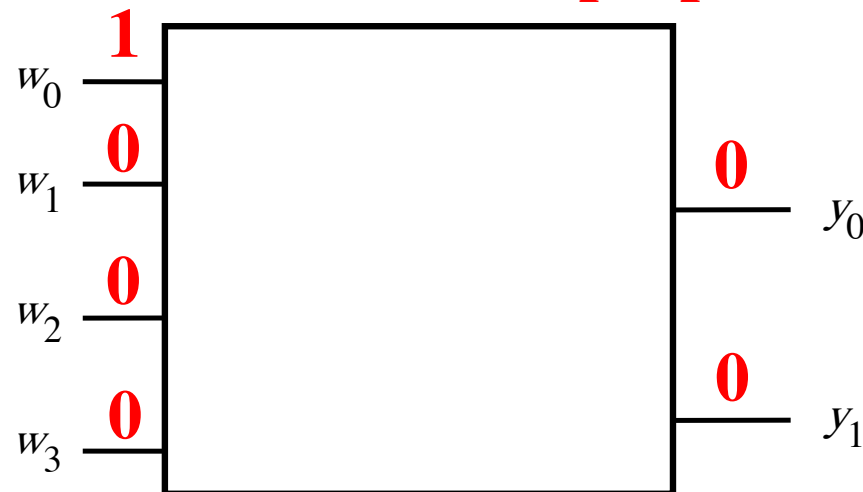




# Circuit for a 4-to-2 binary encoder

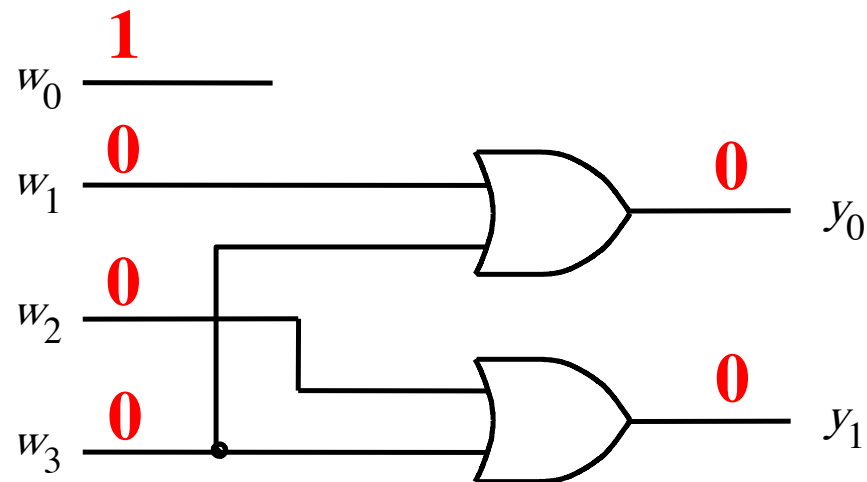
$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

As this level of abstraction we need that  $w_0$  input for this to be a proper 4-to-2 binary encoder.



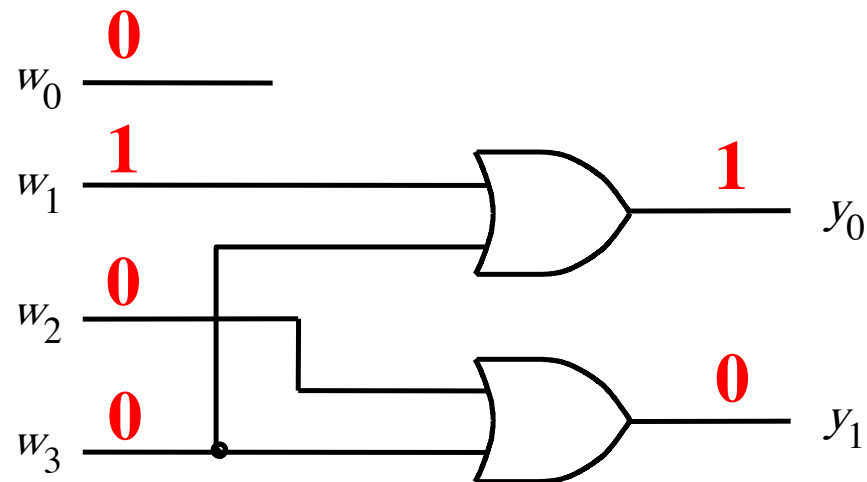
# Circuit for a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



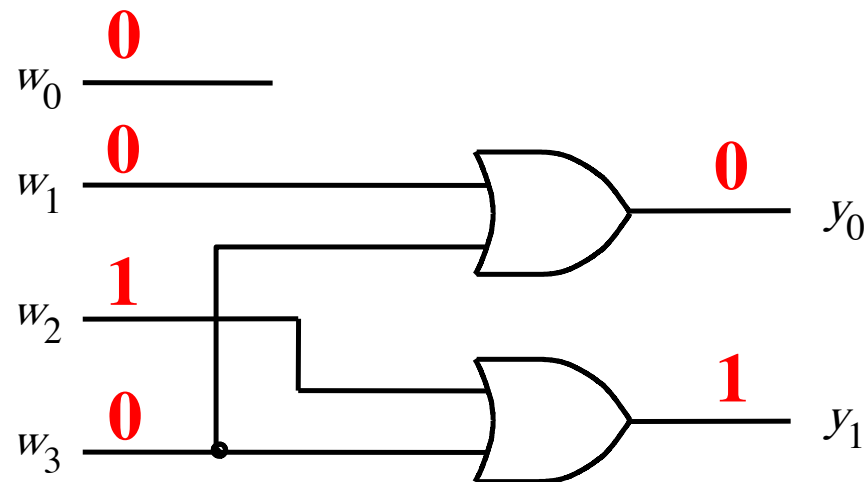
# Circuit for a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



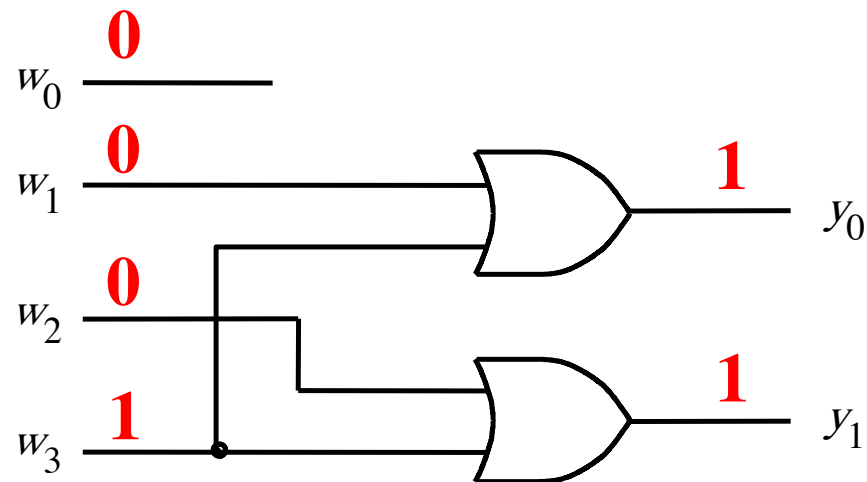
# Circuit for a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



# Circuit for a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



# Expressions for 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	0		
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1		
0	1	0	0	1	0
0	1	0	1		
0	1	1	0		
0	1	1	1		
1	0	0	0	1	1
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

# Expressions for 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	0	d	d
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	d	d
0	1	0	0	1	0
0	1	0	1	d	d
0	1	1	0	d	d
0	1	1	1	d	d
1	0	0	0	1	1
1	0	0	1	d	d
1	0	1	0	d	d
1	0	1	1	d	d
1	1	0	0	d	d
1	1	0	1	d	d
1	1	1	0	d	d
1	1	1	1	d	d

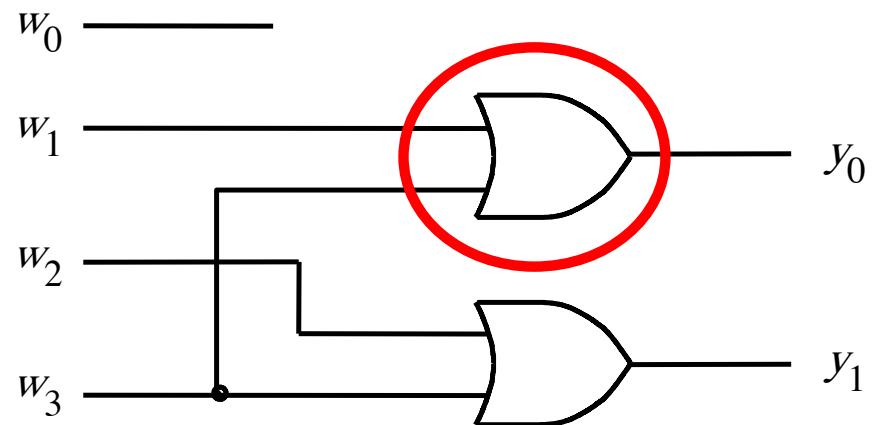
$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

# Expressions for 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	0	d	d
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	d	d
0	1	0	0	1	0
0	1	0	1	d	d
0	1	1	0	d	d
0	1	1	1	d	d
1	0	0	0	1	1
1	0	0	1	d	d
1	0	1	0	d	d
1	0	1	1	d	d
1	1	0	0	d	d
1	1	0	1	d	d
1	1	1	0	d	d
1	1	1	1	d	d

$w_3 w_2$	$w_1 w_0$	00	01	11	10
00	00	d	0	d	1
01	00	0	d	d	d
11	00	d	d	d	d
10	00	1	d	d	d

$$y_0 = (w_1 + w_3)$$



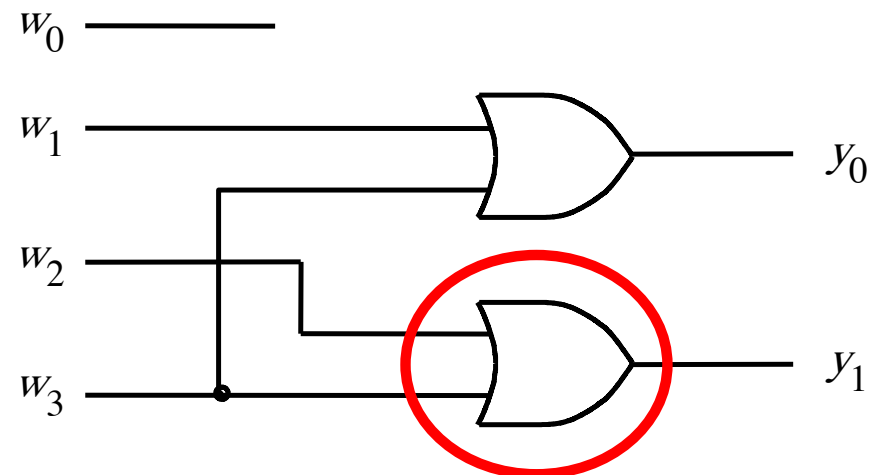


# Expressions for 4-to-2 binary encoder

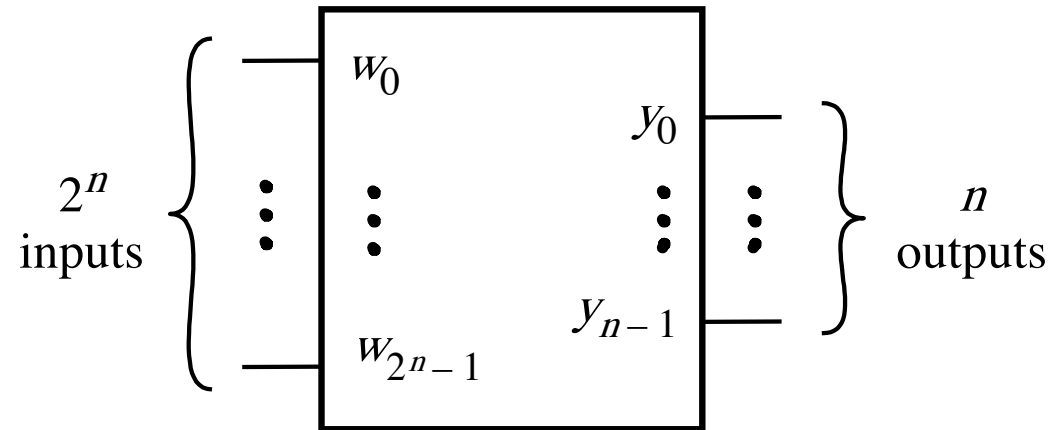
$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$
0	0	0	0	d	d
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	d	d
0	1	0	0	1	0
0	1	0	1	d	d
0	1	1	0	d	d
0	1	1	1	d	d
1	0	0	0	1	1
1	0	0	1	d	d
1	0	1	0	d	d
1	0	1	1	d	d
1	1	0	0	d	d
1	1	0	1	d	d
1	1	1	0	d	d
1	1	1	1	d	d

$w_3 \ w_2$	$w_1 \ w_0$	00	01	11	10
00	00	d	1	d	1
01	00	0	d	d	d
11	00	d	d	d	d
10	00	0	d	d	d

$$y_1 = (w_3 + w_2)$$



# The Most General Case: $2^n$ -to- $n$ binary encoder



# Priority Encoders

# Truth table for a 4-to-2 priority encoder (abbreviated version)

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$	$z$
0	0	0	0	d	d	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

# Truth table for a 4-to-2 priority encoder (abbreviated version)

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$	$z$
0	0	0	0	d	d	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

# Truth table for a 4-to-2 priority encoder

	$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$	$z$
0 0 0 0	0	0	0	0	d	d	0
0 0 0 1	0	0	0	1	0	0	1
0 0 1 x	0	0	1	0	0	1	1
	0	0	1	1	0	1	1
0 1 x x	0	1	0	0	1	0	1
	0	1	0	1	1	0	1
	0	1	1	0	1	0	1
	0	1	1	1	1	0	1
1 x x x	1	0	0	0	1	1	1
	1	0	0	1	1	1	1
	1	0	1	0	1	1	1
	1	0	1	1	1	1	1
	1	1	0	0	1	1	1
	1	1	0	1	1	1	1
	1	1	1	0	1	1	1
	1	1	1	1	1	1	1

# Expressions for 4-to-2 priority encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$	$z$
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

		$w_3 w_2$			
$w_1 w_0$		00	01	11	10
	00	d	1	1	1
01	0	1	1	1	
11	0	1	1	1	
10	0	1	1	1	

$$y_1 = w_3 + w_2$$

# Expressions for 4-to-2 priority encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$	$z$
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

$w_3 w_2$	$w_1 w_0$	00	01	11	10
00	00	d	0	1	1
01	00	0	0	1	1
11	00	1	0	1	1
10	00	1	0	1	1

$$y_0 = w_3 + w_1 \overline{w_2}$$



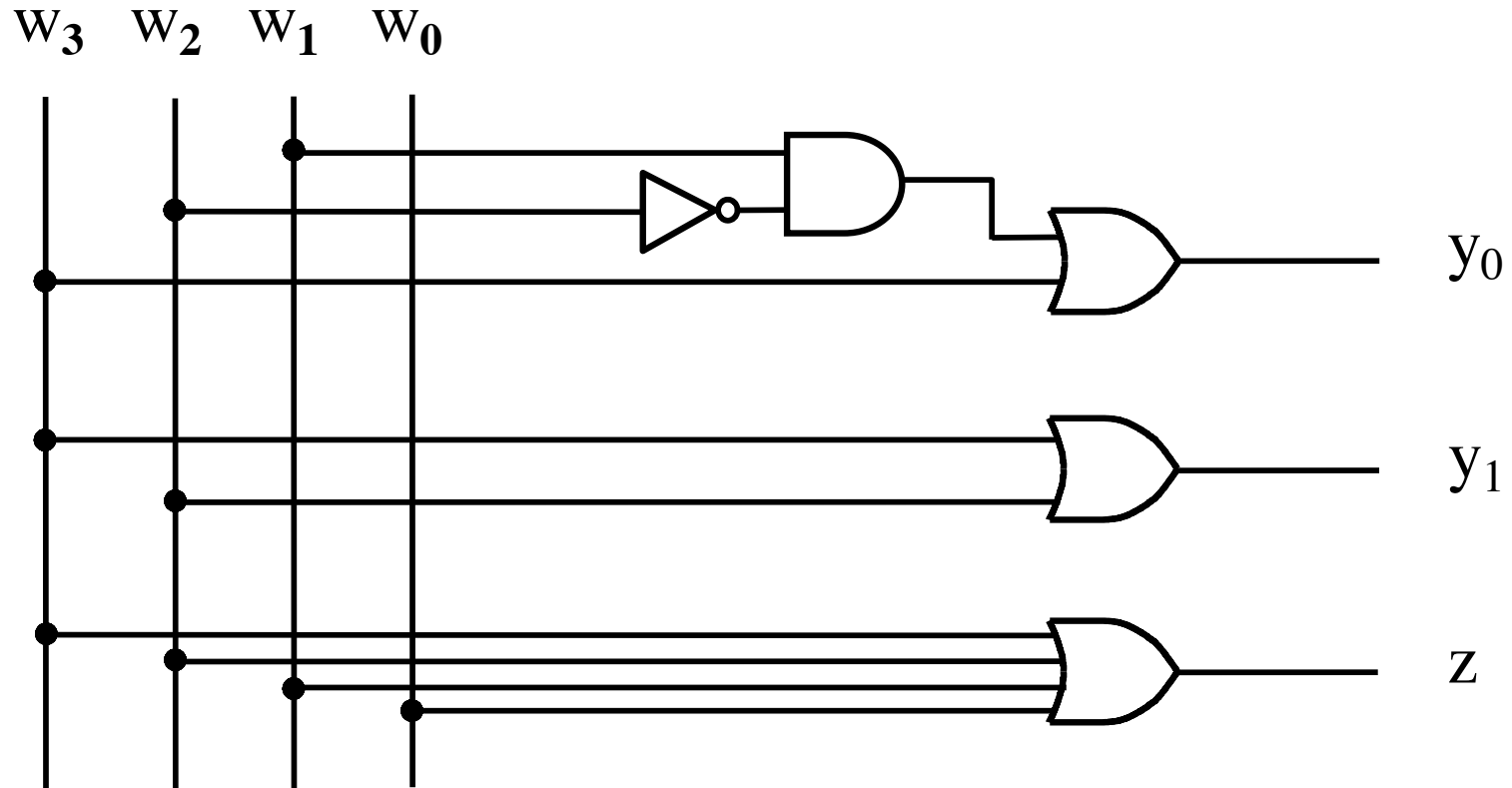
# Expressions for 4-to-2 priority encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$	$z$
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

		$w_3 w_2$			
		00	01	11	10
$w_1 w_0$	00	0	1	1	1
	01	1	1	1	1
	11	1	1	1	1
	10	1	1	1	1

$$Z = w_3 + w_2 + w_1 + w_0$$

# Circuit for the 4-to-2 priority encoder



# The textbook derives a different circuit for the 4-to-2 priority encoder using a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$	$z$
0	0	0	0	d	d	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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

$$i_2 = \bar{w}_3 w_2$$

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

$$z = i_0 + i_1 + i_2 + i_3$$

# The textbook derives a different circuit for the 4-to-2 priority encoder using a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$	$z$
0	0	0	0	d	d	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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

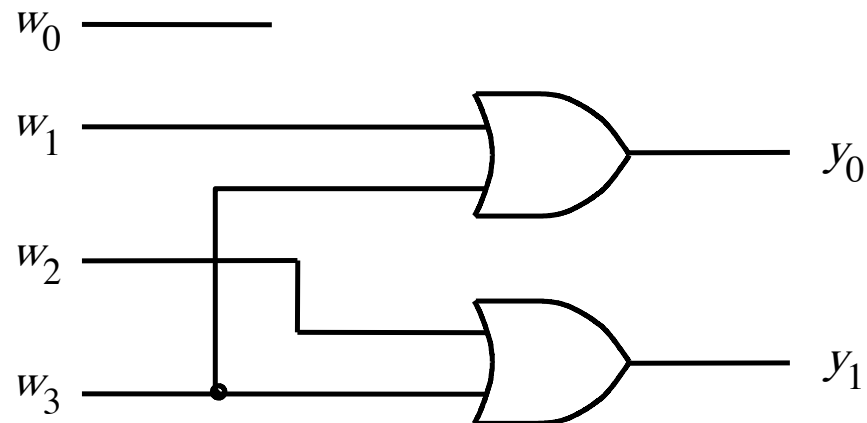
$$i_2 = \bar{w}_3 w_2$$

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

$$z = i_0 + i_1 + i_2 + i_3$$



# The textbook derives a different circuit for the 4-to-2 priority encoder using a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$	$z$
0	0	0	0	d	d	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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

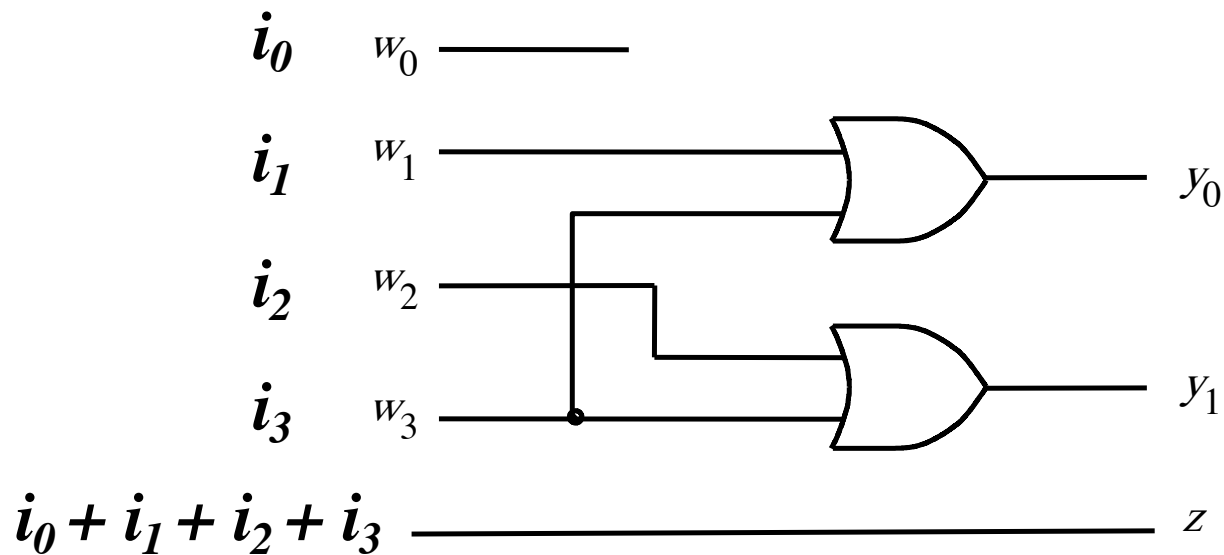
$$i_2 = \bar{w}_3 w_2$$

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

$$z = i_0 + i_1 + i_2 + i_3$$



# The textbook derives a different circuit for the 4-to-2 priority encoder using a 4-to-2 binary encoder

$w_3$	$w_2$	$w_1$	$w_0$	$y_1$	$y_0$	$z$
0	0	0	0	d	d	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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

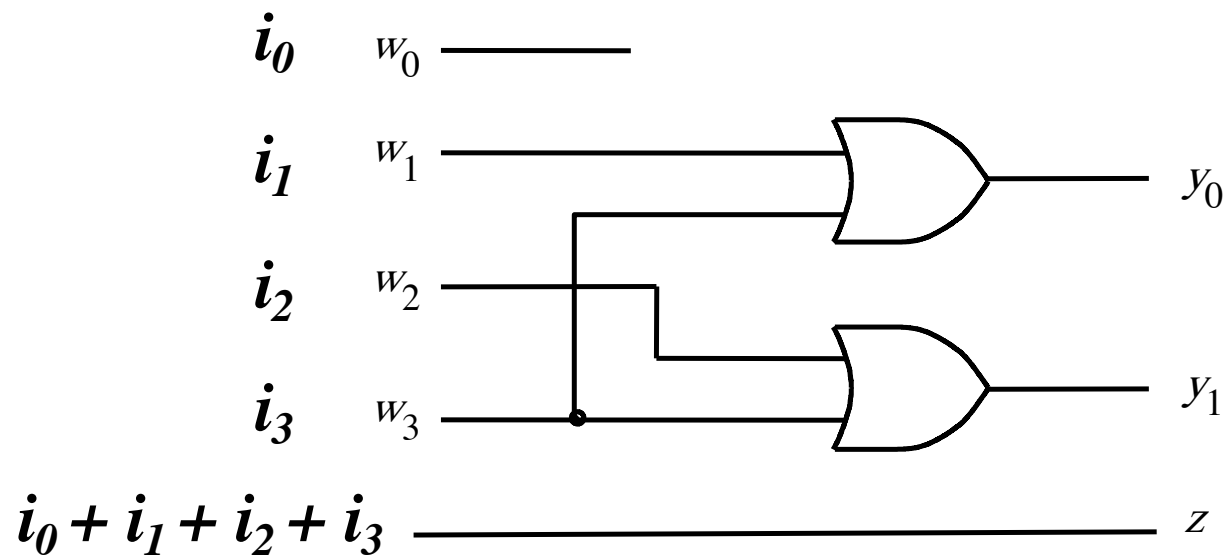
$$i_2 = \bar{w}_3 w_2$$

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

$$z = i_0 + i_1 + i_2 + i_3$$



Try to prove that this is equivalent to the circuit that was derived above.

**Questions?**

**THE END**