

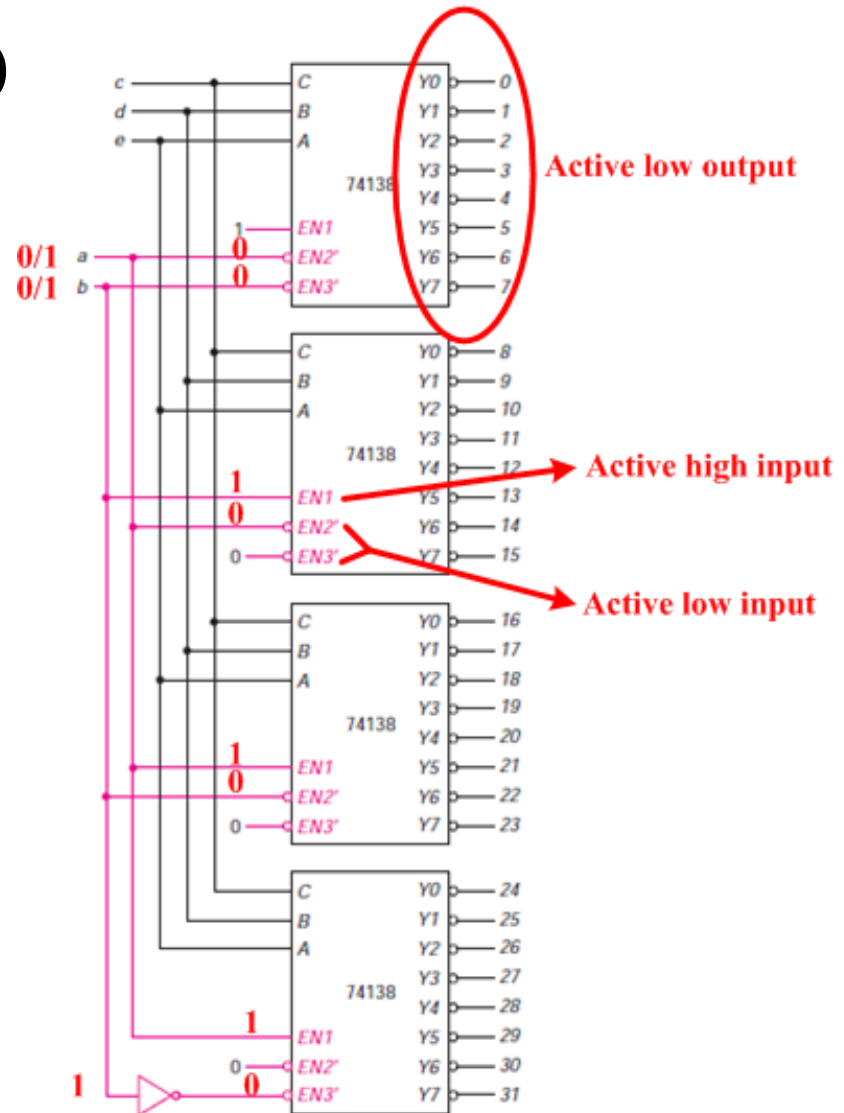
Chapter 5 Designing Combinational Systems (II)

Binary Decoders (1/11)

- **Example 5.1.** We have available 74138 decoders and wish to select one of 32 devices. Design the circuit by using four such decoders
- One of these would select one of the first eight addressed devices; another would select one of the next eight; and so forth
- Let the addresses were given by bits a, b, c, d, e , then c, d, e would be the inputs to each of the four decoders, and a, b would be used to enable the appropriate one
- The first decoder would be enabled when $a=b=0$, the second when $a=0$ and $b=1$, the third when $a=1$ and $b=0$, and the fourth when $a=b=1$

Binary Decoders (2/11)

- **Example 5.1. (Cont'd)**

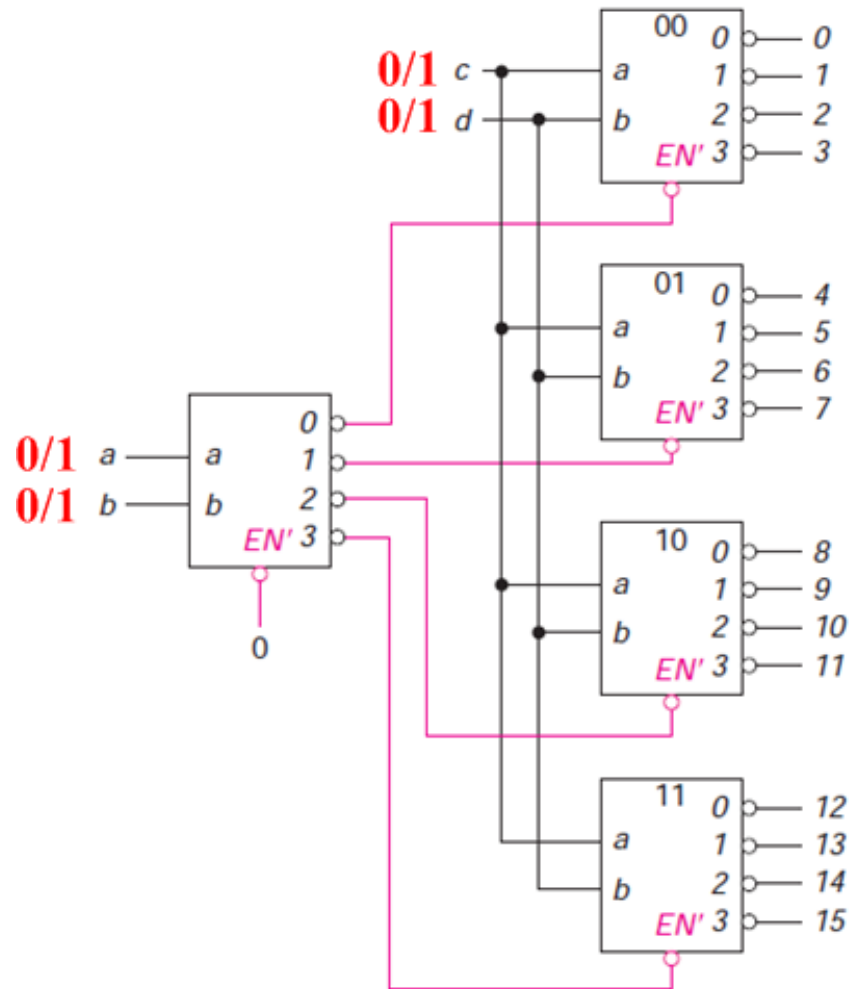


Binary Decoders (3/11)

- **Example 5.2.** An extra decoder can be used to enable other decoders. If we had two-input, four-output active low decoders and needed to select one of 16 devices. Design the circuit
- The 16 devices are divided into four groups, which are 0-3, 4-7, 8-11, and 12-15. The first two (highest order) inputs are used to choose the four groups, while the other two (lowest order) inputs are used to choose among the four devices in that group

Binary Decoders (4/11)

- Example 5.2. (Cont'd)



Binary Decoders (5/11)

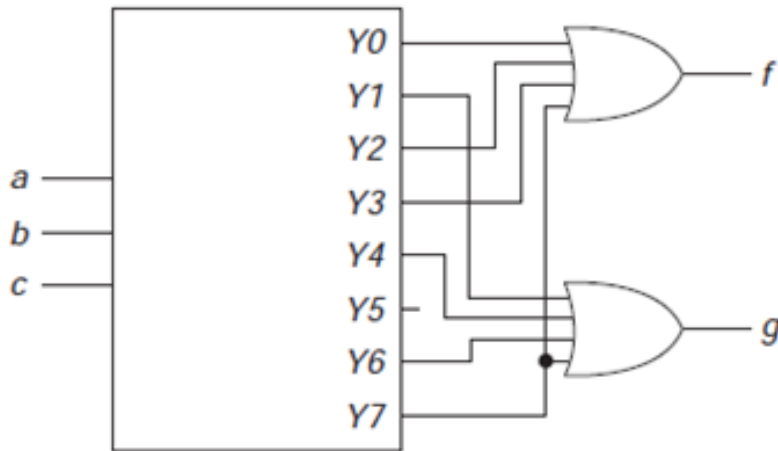
- Another application of decoders is the implementation of logic functions. Each active high output of a decoder corresponds to a minterm of that function. We need an OR gate connected to the appropriate outputs
- With an active low output decoder, the OR gate is replaced by a NAND
- With more than one output functions of the same set of inputs, we still only need one decoder, but one OR or NAND for each output function

Binary Decoders (6/11)

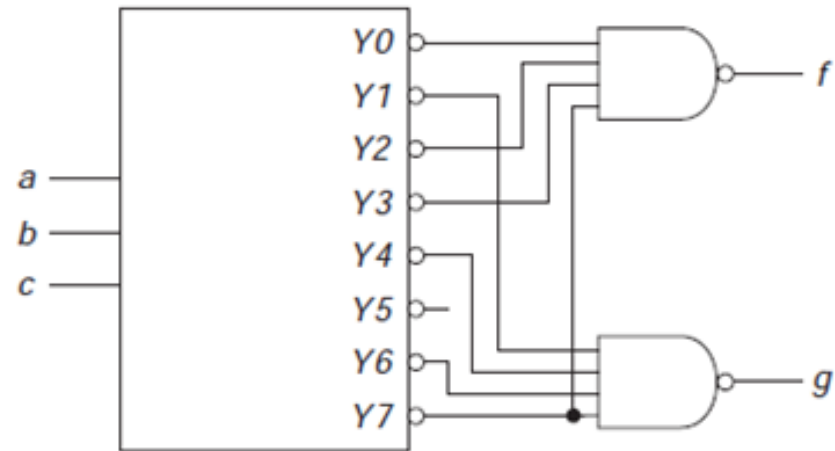
- **Example 5.3.** Implement the following two functions using an active-high output decoder and an active-low output decoder, respectively. where

$$f(a,b,c) = \sum m(0,2,3,7)$$

$$g(a,b,c) = \sum m(1,4,6,7)$$



Active-high output decoder



Active-low output decoder

Binary Decoders (7/11)

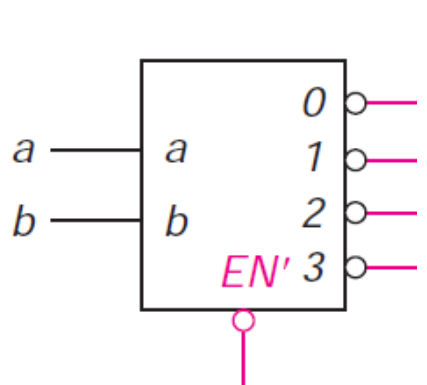
- **Example 5.4.** Implement the following functions using only the decoders shown below and three NAND gates

$$f(a,b,c,d) = \sum m(0,2,3,4,7,8,10,11,15)$$

$$g(a,b,c,d) = \sum m(0,1,2,5,8,9,10,12,13,15)$$

$$h(a,b,c,d) = \sum m(2,4,6,8,10,12,14,15).$$

- Truth table for the decoder. It is an active-low output decoder



EN'	a	b	0	1	2	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

Binary Decoders (8/11)

- **Example 5.4. (Cont'd)** We would need a 9-input NAND for f , a 10-input gate for g , and a 8-input gate for h .

$c d \backslash a b$	00	01	11	10
00	1	1		1
01				
11	1	1	1	1
10	1			1

f

$c d \backslash a b$	00	01	11	10
00	1		1	1
01	1	1	1	1
11			1	
10	1			1

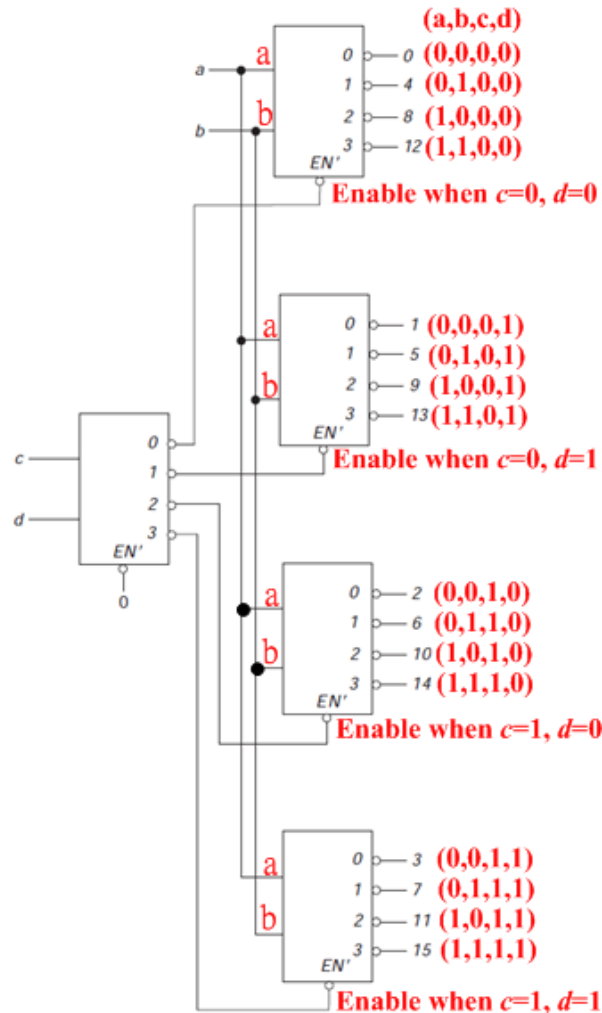
g

$c d \backslash a b$	00	01	11	10
00		1	1	1
01				
11			1	
10	1	1	1	1

h

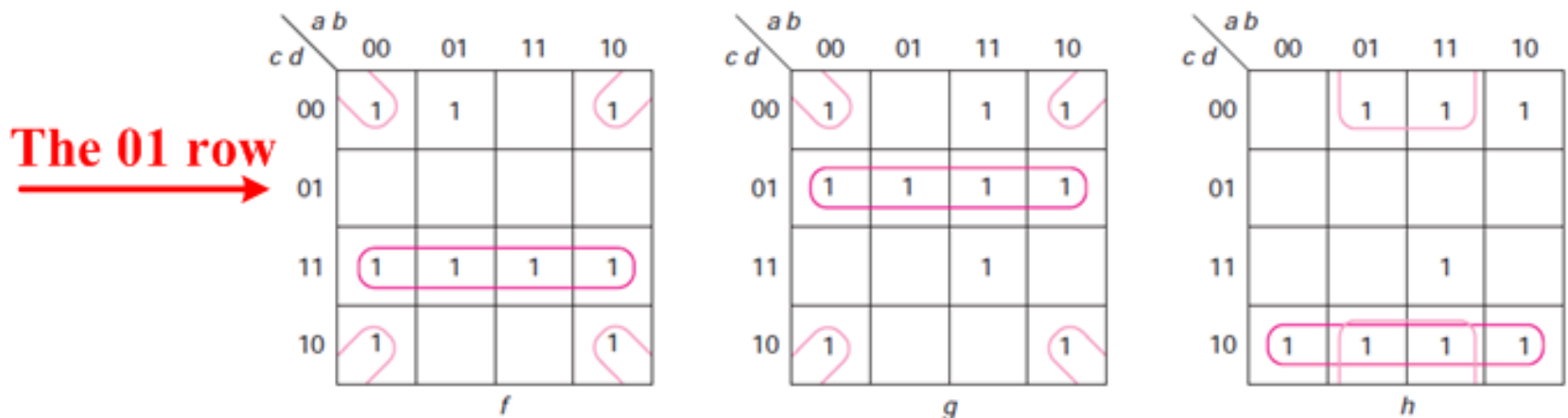
Binary Decoders (9/11)

- Example 5.4. (Cont'd)



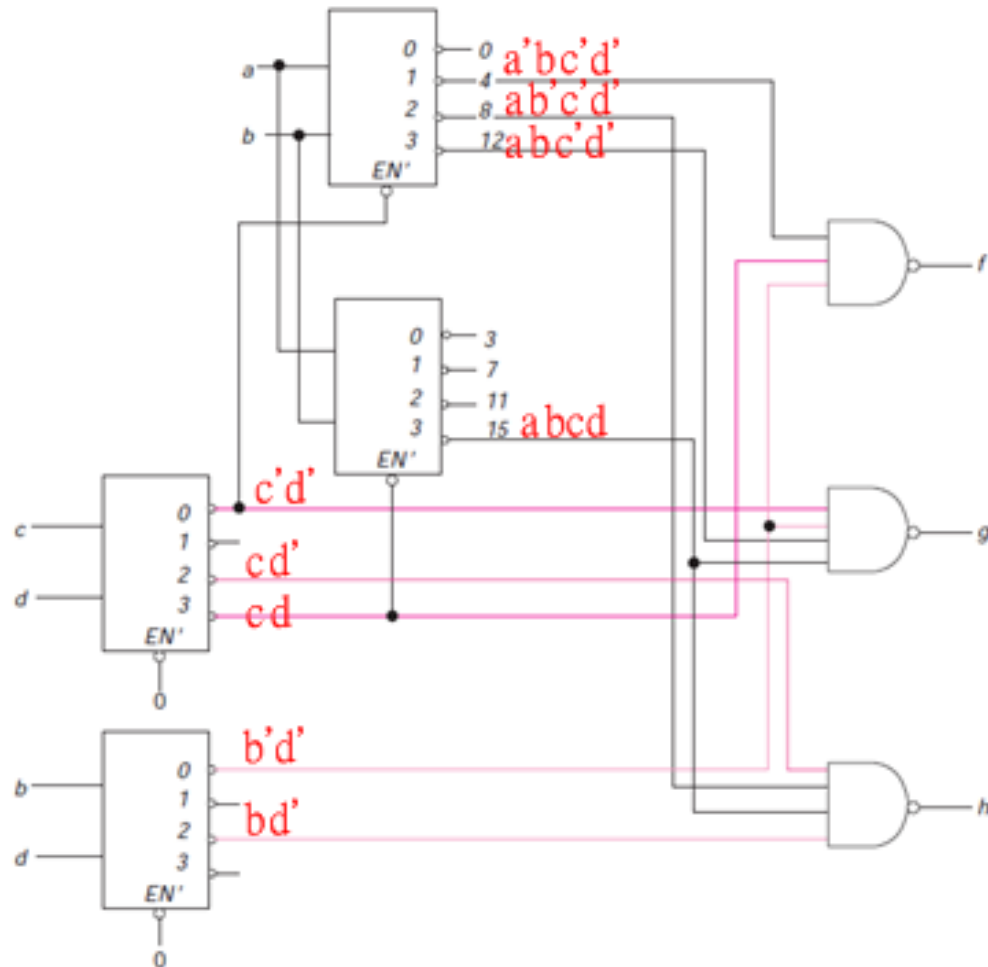
Binary Decoders (10/11)

- **Example 5.4. (Cont'd)** The 01 row of the maps either have no 1's or all 1's. This implies one decoder can be removed.
- The pink terms correspond to cd , $c'd$, and cd' , the 3, 1, and 2 outputs, respectively, from a decoder with inputs c and d
- The tan outputs correspond to $b'd'$ and bd' , the 0 and 2 outputs from a decoder with input b and d



Binary Decoders (11/11)

- The remaining 1's are covered as before



Encoders and Priority Encoders (1/4)

- A binary encoder is the inverse of a binary decoder
- If we assume there exactly only one input (of A_0, A_1, A_2, A_3) is 1, then the truth table of Table 5.4 describes the behavior of the encoder

Table 5.4 A four-line encoder.

A_0	A_1	A_2	A_3	Z_0	Z_1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

- Z_0 and Z_1 can be represented as

$$Z_0 = A_2 + A_3$$

$$Z_1 = A_1 + A_3$$

Encoders and Priority Encoders (2/4)

- If another output, N , indicates that no input is active, then

$$N = A_0' A_1' A_2' A_3' = (A_0 + A_1 + A_2 + A_3)'$$

- If more than one input can occur at the same time, then some priority must be established.
- The priorities are normally arranged in descending (or ascending) order with the highest priority given to the largest (smallest) input number

Encoders and Priority Encoders (3/4)

- Assume device 7 has the highest priority, the truth table is shown in Table 5.5.
- The output NR indicates that there are no requests

Table 5.5 A priority encoder.

A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7	Z_0	Z_1	Z_2	NR
0	0	0	0	0	0	0	0	X	X	X	1
X	X	X	X	X	X	X	1	1	1	1	0
X	X	X	X	X	X	1	0	1	1	0	0
X	X	X	X	X	1	0	0	1	0	1	0
X	X	X	X	1	0	0	0	1	0	0	0
X	X	X	1	0	0	0	0	0	1	1	0
X	X	1	0	0	0	0	0	0	1	0	0
X	1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0

Lowest priority

Highest priority

Encoders and Priority Encoders (4/4)

- The 74147 is a commercial BCD (binary coded decimal) encoder, taking nine active low input lines and encoding them into four active low outputs
- **Table 5.6** The 74147 priority encoder.

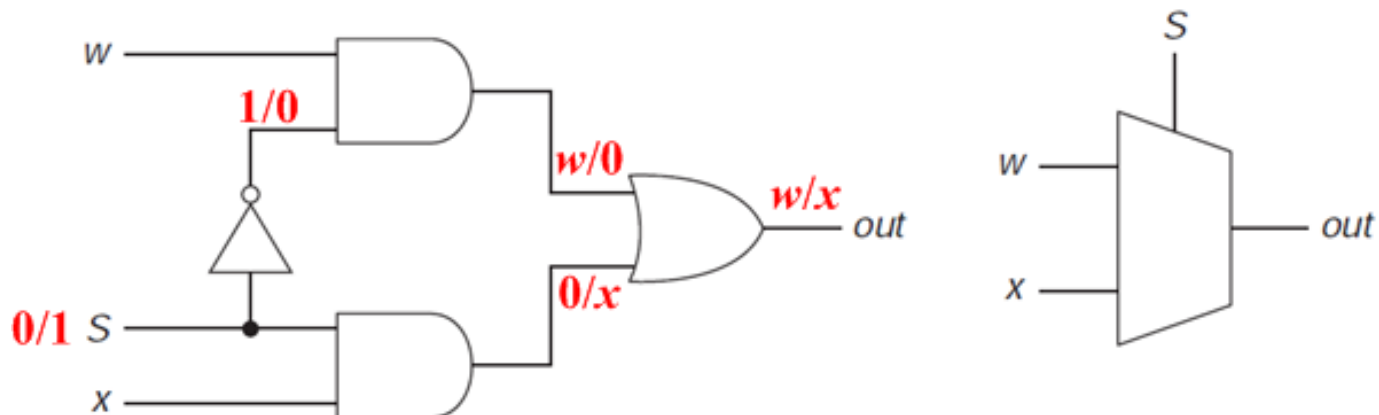
$1'$	$2'$	$3'$	$4'$	$5'$	$6'$	$7'$	$8'$	$9'$	D'	C'	B'	A'
1	1	1	1	1	1	1	1	1	1	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	0	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0

Multiplexers and Demultiplexers (1/8)

- A multiplexer, often referred to as a mux, is basically a switch that passes one of its data inputs through to the output, as a function of a set of selected inputs
- The output *out* is determined by *S*, where

$$out = \begin{cases} w, & S = 0 \\ x, & S = 1 \end{cases}$$

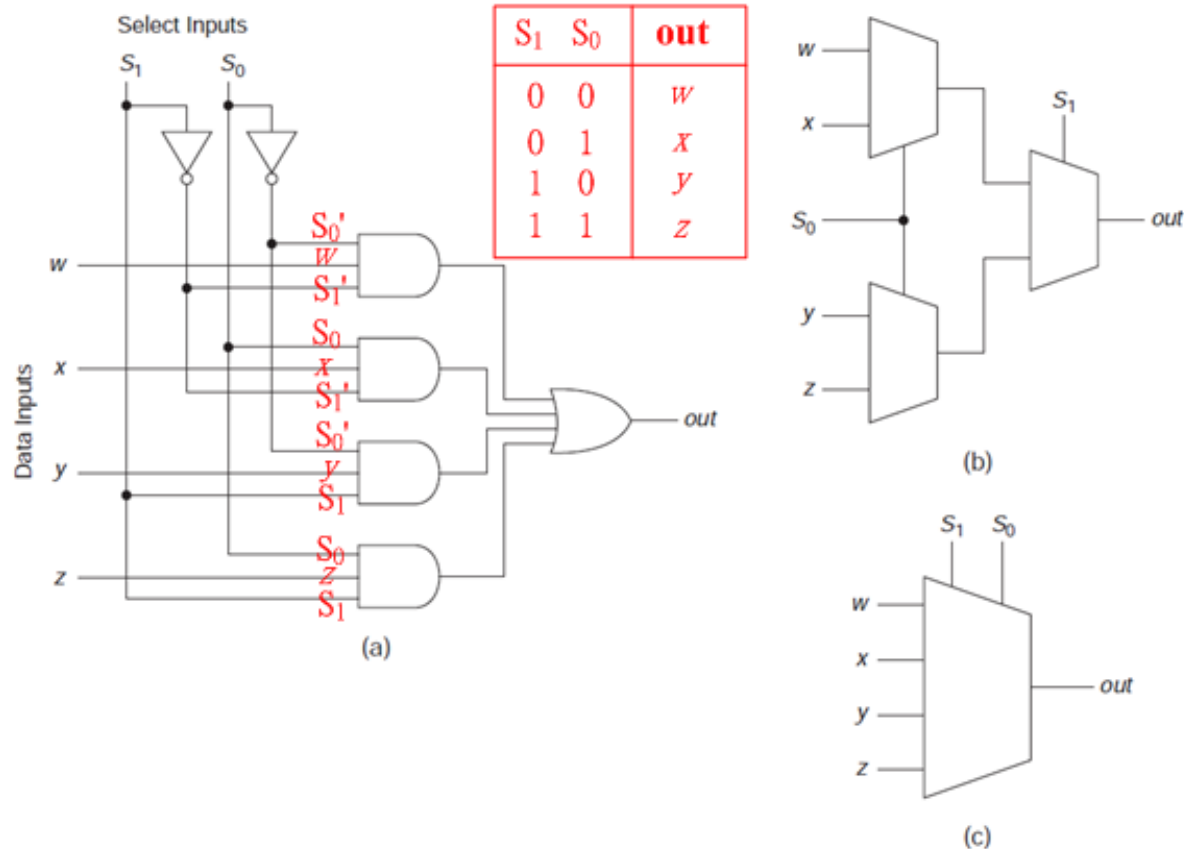
Figure 5.11 Two-way multiplexer.



Multiplexers and Demultiplexers (2/8)

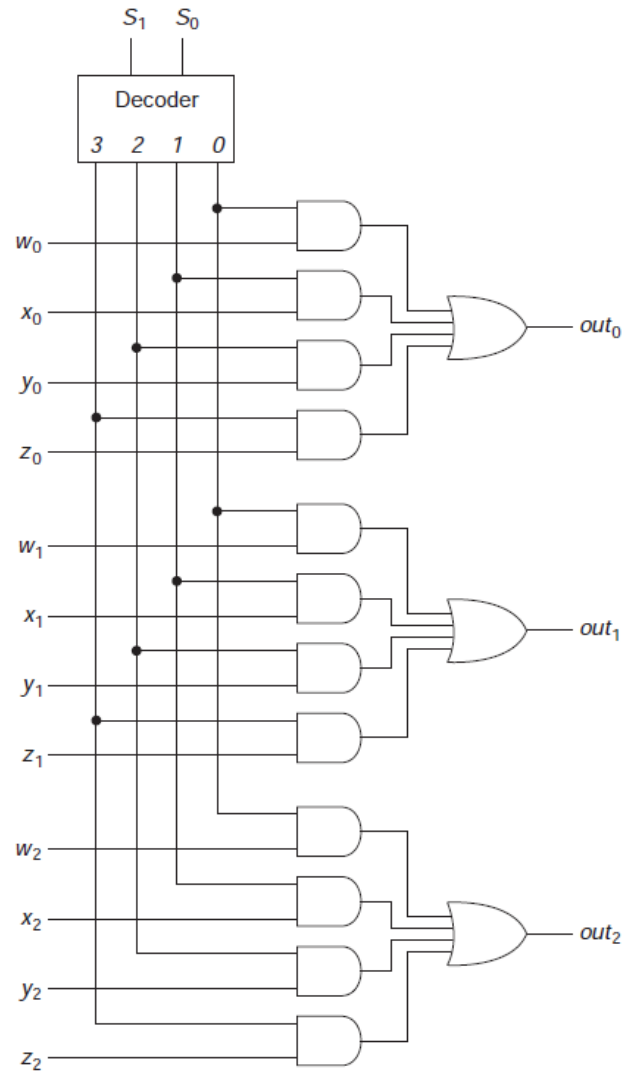
- A four-way multiplexer can be implemented with AND and OR gates, as shown in Figure 5.12

Figure 5.12 (a) A four-way multiplexer. (b) From two-way multiplexers. (c) Logic symbol.



Multiplexers and Demultiplexers (3/8)

Figure 5.13 A multibit multiplexer.



Multiplexers and Demultiplexers (4/8)

- **Example 5.5.** Implement a three-variable function $f(a,b,c)$ using an eight-way multiplexer. The three variables go to the control inputs. The truth table for the function is then connected to the data inputs. The function $f(a,b,c)$ is defined as follows

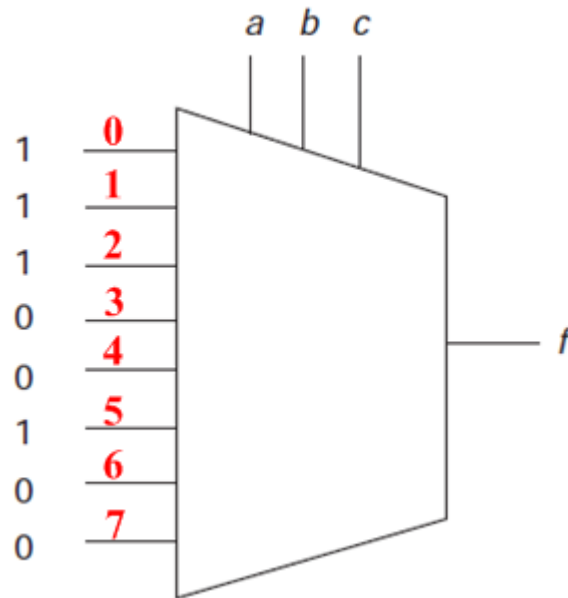
$$f(a,b,c) = \sum m(0, 1, 2, 5).$$

- The truth table is

	<i>a</i>	<i>b</i>	<i>c</i>	<i>f</i>
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Multiplexers and Demultiplexers (5/8)

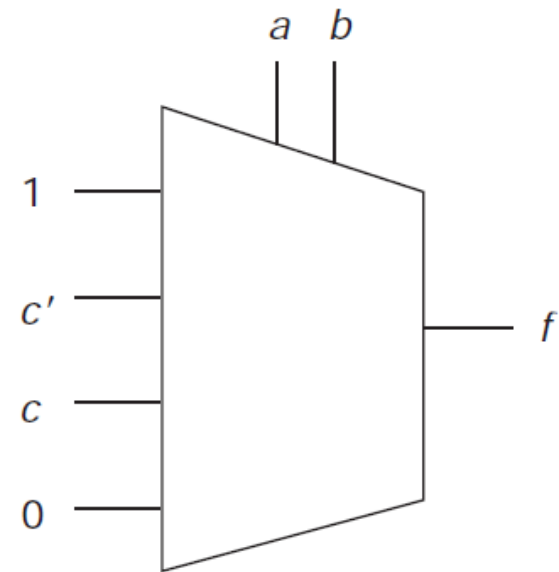
- **Example 5.5. (Cont'd)** With a mux, it can be implemented by



Multiplexers and Demultiplexers (6/8)

- **Example 5.5. (Cont'd)** We can also use a four-way multiplexer to implement it. The truth table and the implementation areas follows is

$a b$	f		f
	$c = 0$	$c = 1$	
00	1	1	1
01	1	0	c'
10	0	1	c
11	0	0	0

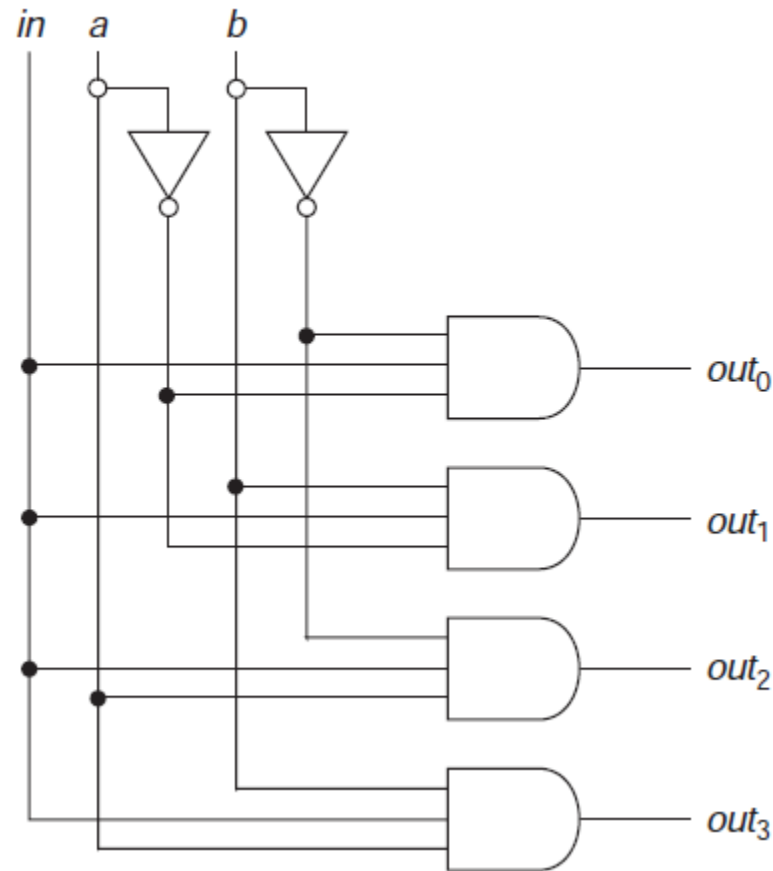


Multiplexers and Demultiplexers (7/8)

- A *demultiplexer* (*demux*) is the inverse of a mux. It routes a signal from one place to one of many
- There are some commercially available multiplexer packages, *e.g.*, 74151, 74153, and 74157
- Fig. 5.14 shows one bit of a four-way demux, where a and b select which way the signal, in , is directed. The circuit is the same as for a four-way decoder with the signal in replacing EN

Multiplexers and Demultiplexers (8/8)

Figure 5.14 A four-way demux.

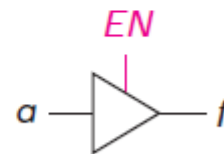


Three-Stage Gates (1/4)

- In a three-state gate, there is an enable input, shown on the side of the gate. If that input is active (it could be active high or active low), the gate behaves as usual. If the control input is inactive, the output behaves as if it is not connected (as an open circuit, high impedance). The output is typically represented by a Z

Figure 5.15 A three-state buffer.

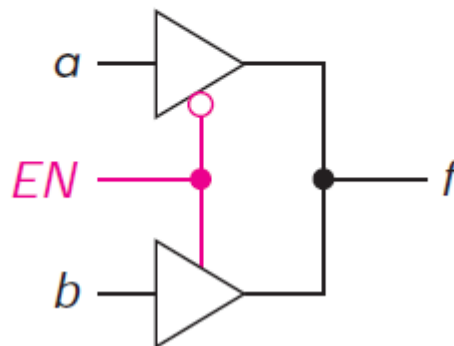
EN	a	f
0	0	Z
0	1	Z
1	0	0
1	1	1



Three-Stage Gates (2/4)

- Fig. 5.16 is a two-way multiplexer. The enable is the control input, determining whether $f=a$ ($EN=0$) or $f=b$ ($EN=1$). The three-state gate is often used for signals that travel between systems

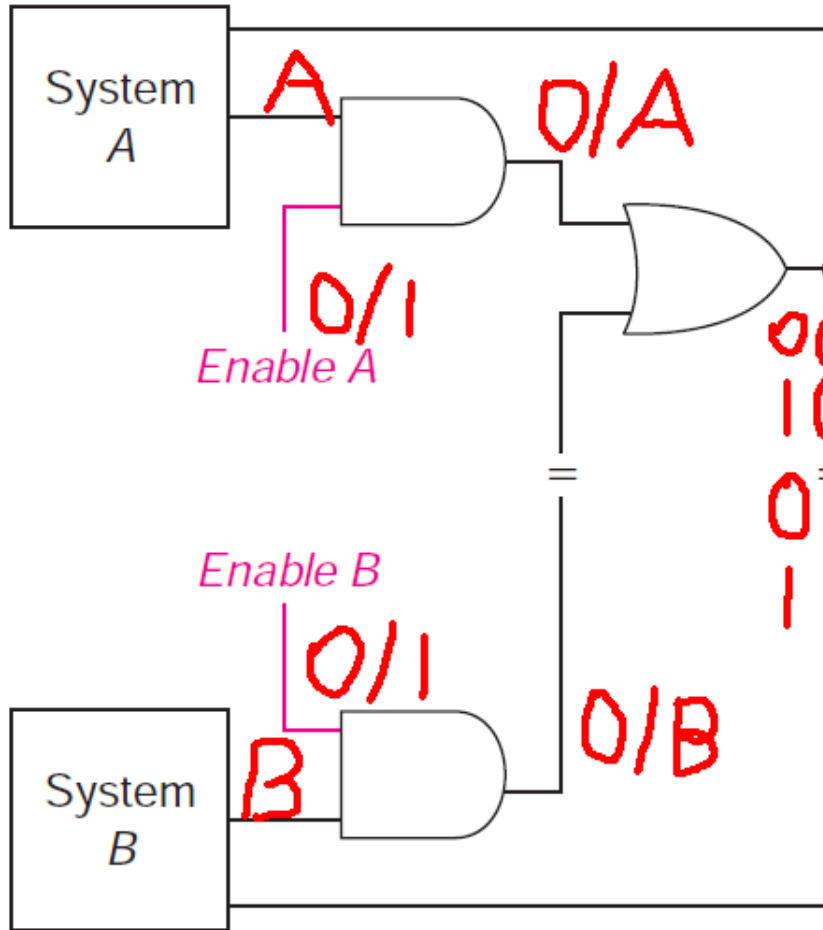
Figure 5.16 A multiplexer using three-state gates.



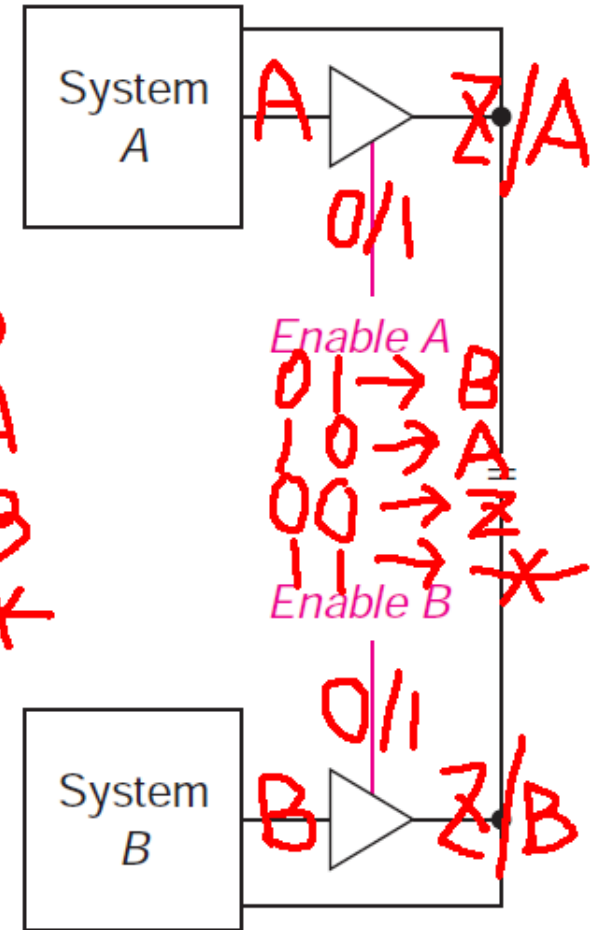
Three-Stage Gates (3/4)

- A *bus* is a set of lines over which data are transferred. Sometimes, that data may travel in either direction between devices located physically at a distance. A bus allows one bit to travel at a time instant
- Example 5.6. The following circuits show two implementations of a one-bit bus – one using AND/OR gates and the other using three-state gates

Three-Stage Gates (4/4)



(a) Using AND/OR Gates



(b) Using Three-State Gates