

chapter two

Combinational logic circuit

- A combinational circuits one where the output at any time depends only on the present combination of inputs at that point of time with total disregard to the past state of the inputs.
- The logic gate is the most basic building block of combinational logic.
- The logical function performed by combinational circuit is fully defined by a set of Boolean expressions.

Half adder

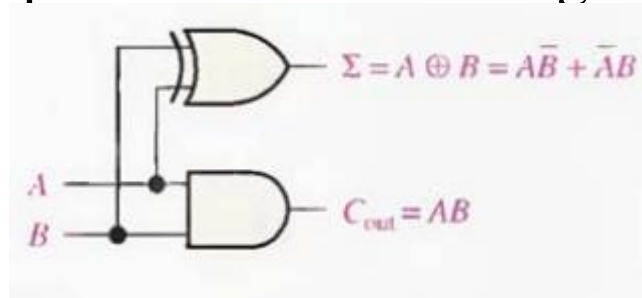
- Half adder accept two binary digit on its input and produce two binary digit on its output, a sum and carry.
- The truth table for half adder is as follows

A	B	C_{out}	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Σ = sum
 C_{out} = output carry
A and B = input variables (operands)

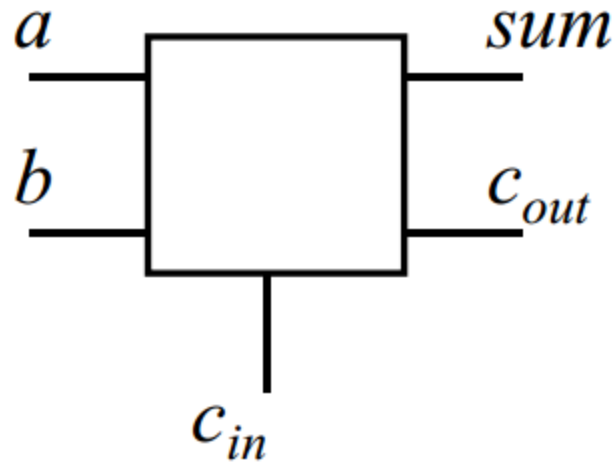
❑ From the truth table

- ✓ the sum output is 1 ,only if the input A and B are not equal.so the sum can be expressed as EXCLUSIVE OR of input variable.
- ✓ The output carry produced with AND gate with input A and B.



Full Adder

- Adds together two, single bit binary numbers a and b (note: with a carry input).



- Has the following truth table:
- So $sum = c_{in} \oplus a \oplus b$ and

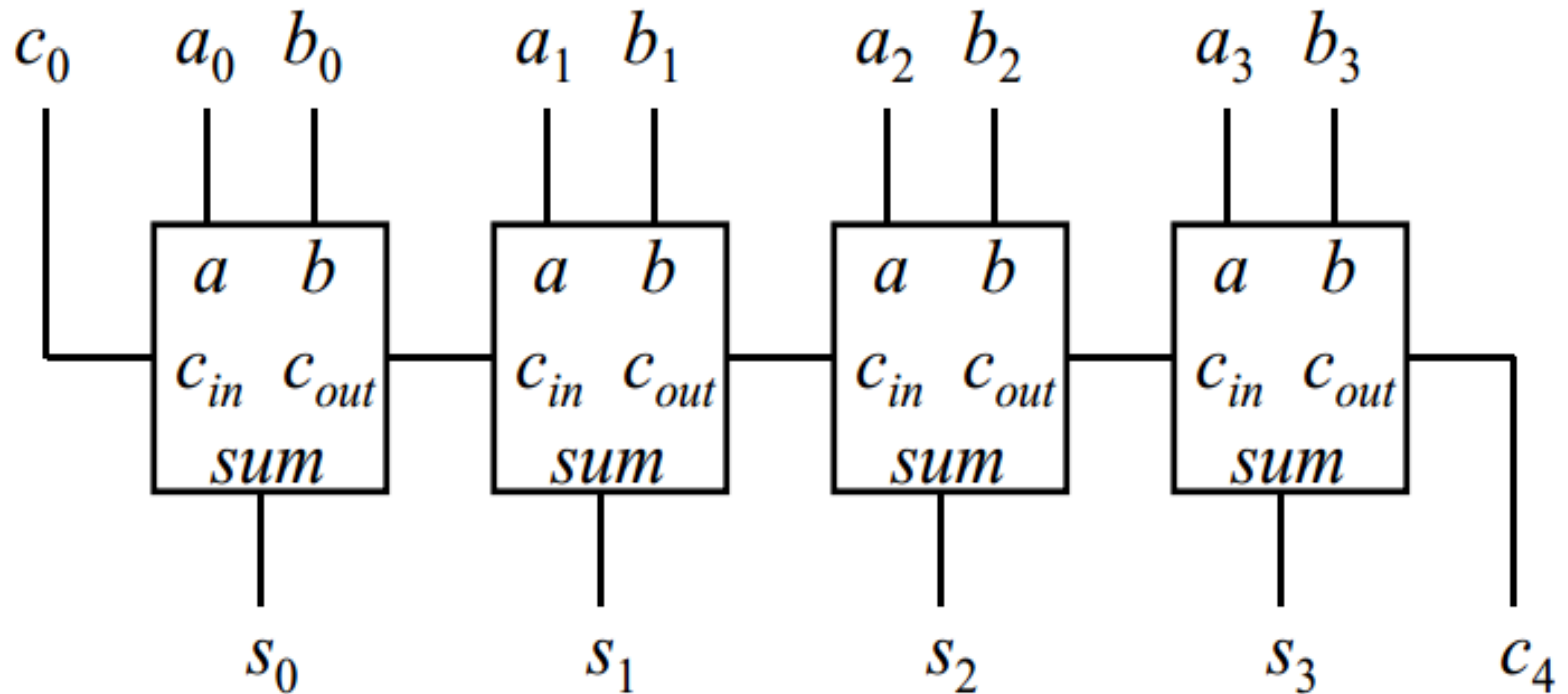
$$c_{out} = b.a + c_{in}.(b + a)$$

c_{in}	a	b	c_{out}	sum
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

Ripple Carry Adder

- We have seen how we can implement a logic to add two, one bit binary numbers (inc. carry-in).
- However, in general we need to add together two, n bit binary numbers.
- One possible solution is known as the Ripple Carry Adder
 - This is simply n , full adders cascaded together.
 - Example, 4 bit adder

Ripple Carry Adder



- Note: If we complement a and set c_0 to one we have implemented $s = b - a$

Decoder

- decoder is combinational logic circuit that detects the presence of specified combination of bits(code) on its inputs and indicates the presence of that code by a specified output level.
- Decoder has n input lines to handle n bits and from one tooutput lines to indicate the presence of one or more n -bit combinations.

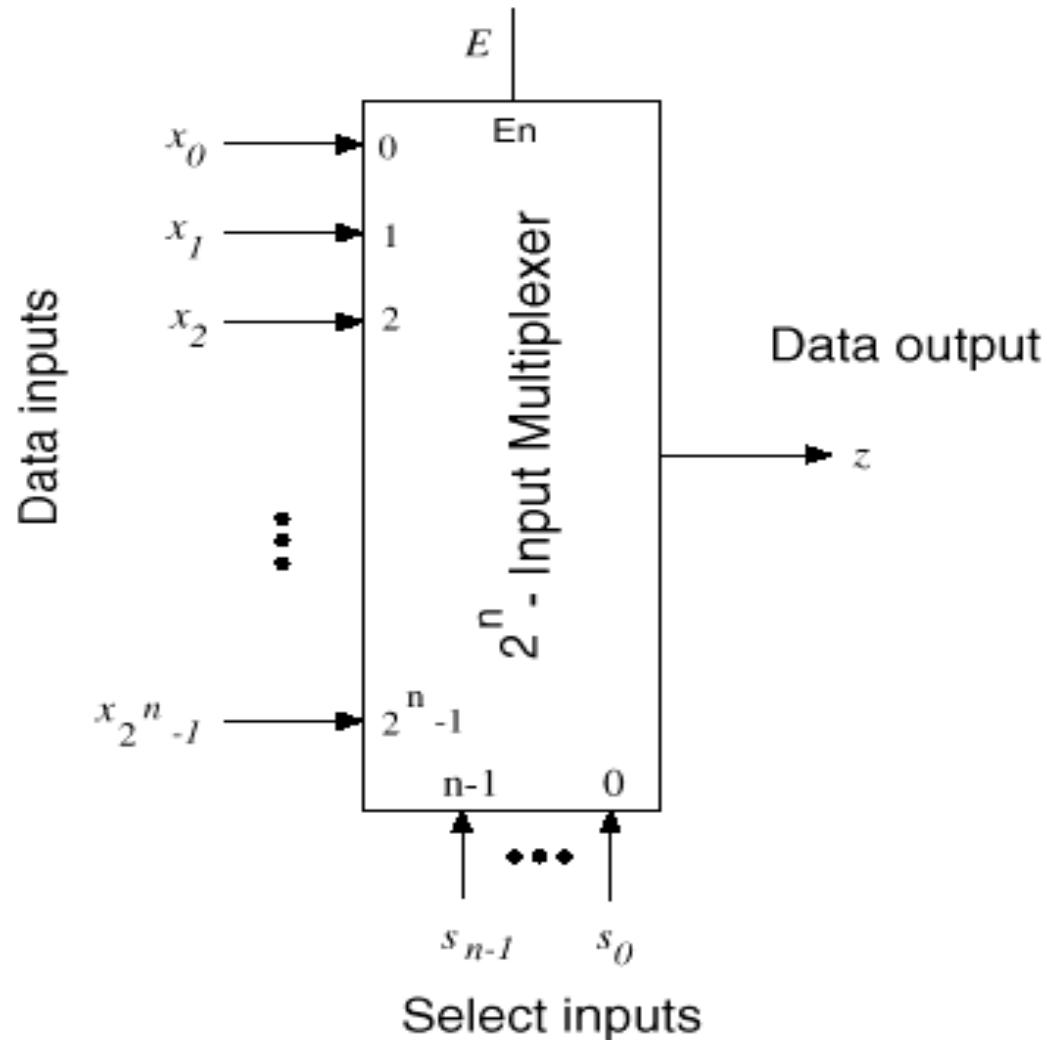
Overview of Encoder and Decoder

- MUX Gate
- Rudimentary functions
- Binary Decoders
 - Expansion
 - Circuit implementation
- Binary Encoders
- Priority Encoders

Multiplexer

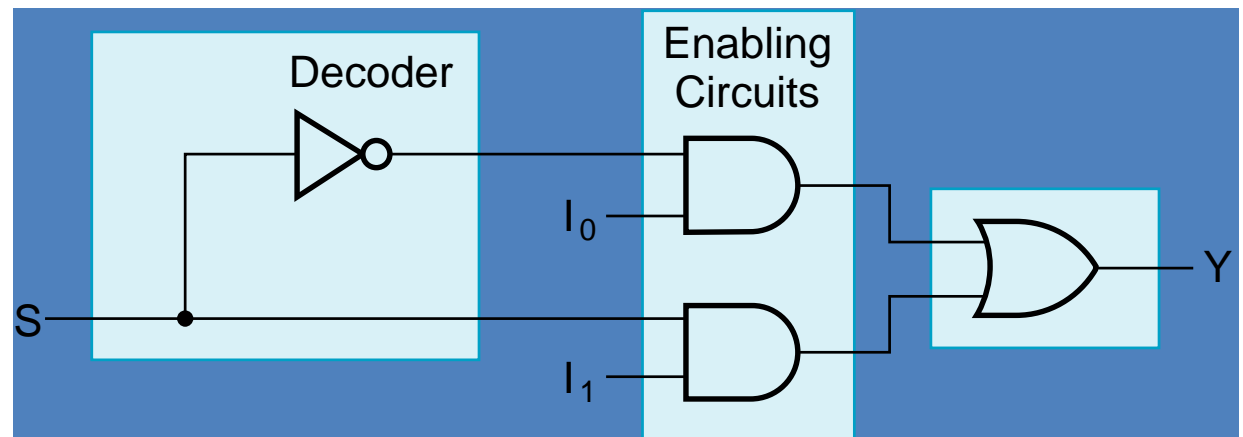
- “Selects” binary information from one of many input lines and directs it to a single output line.
- Also known as the “selector” circuit,
- Selection is controlled by a particular set of input lines whose number depends on the number of the data input lines.
- For a 2^n -to-1 multiplexer, there are 2^n data input lines and n selection lines whose bit combination determines which input is selected.

Multiplexer (cont.)

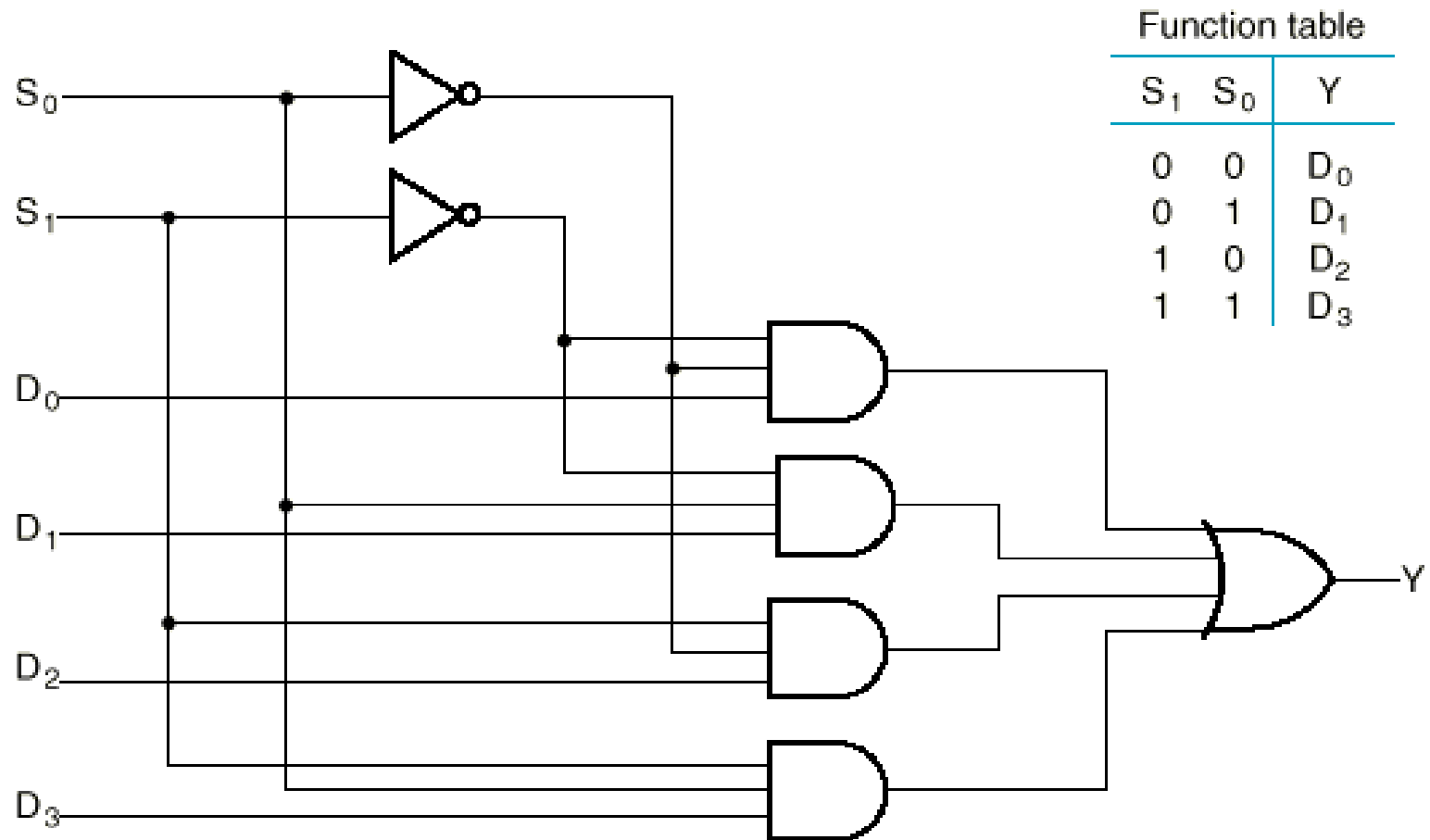


2-to-1-Line Multiplexer

- Since $2 = 2^1$, $n = 1$ number of select line.
- The single selection variable S has two values:
 - $S = 0$ selects input I_0
 - $S = 1$ selects input I_1
- The equation:
$$Y = S' I_0 + S I_1$$
- The circuit:

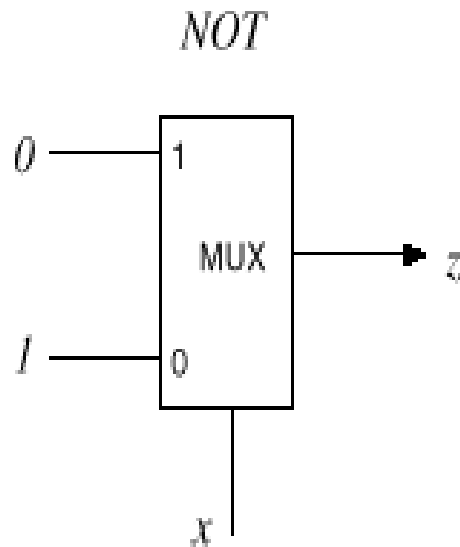
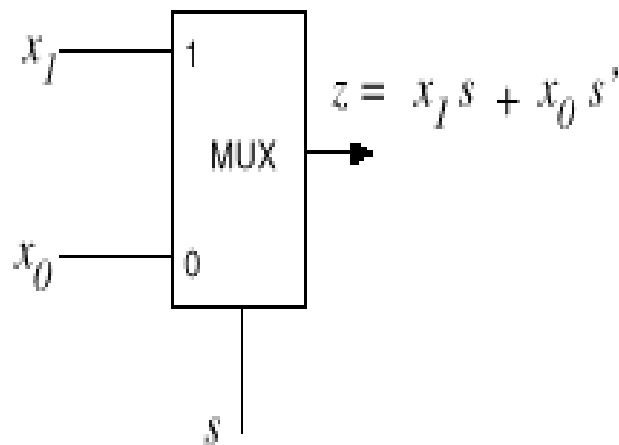


Example: 4-to-1 MUX using Cell Library Based Design

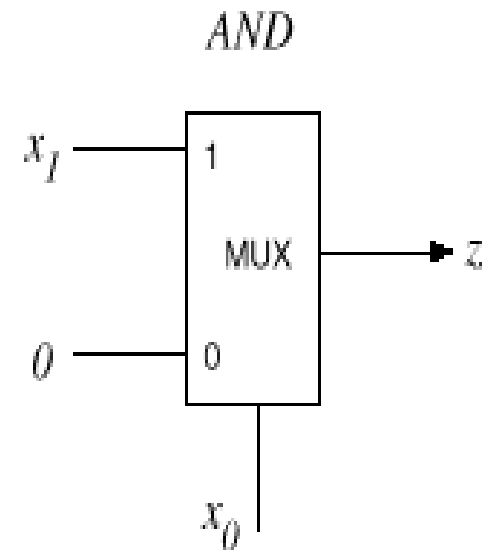


MUX as a Universal Gate

- We can construct AND and NOT gates using 2-to-1 MUXs. Thus, 2-to-1 MUX is a universal gate.



$$z = 0x + 1x' = x'$$



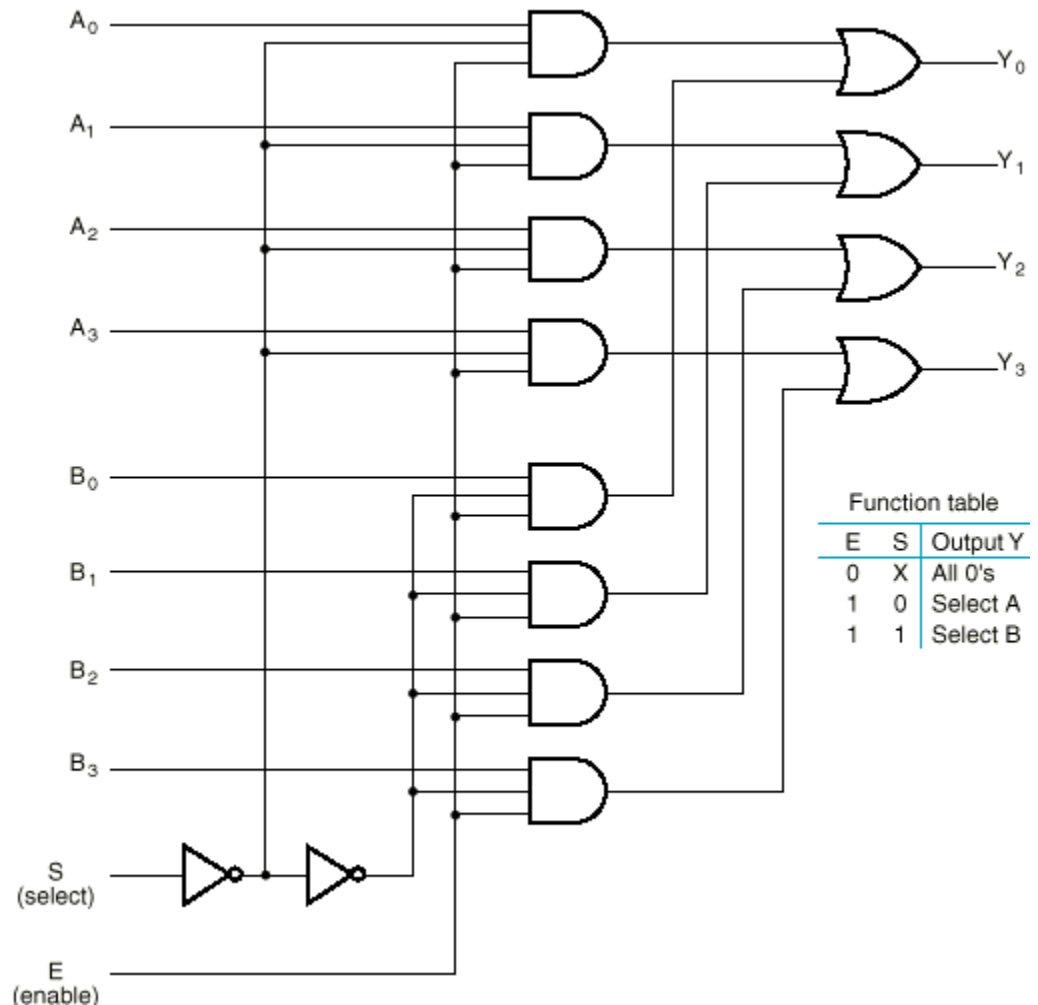
$$z = x_1 x_0 + 0x_0' = x_1 x_0$$

Multiple Bit Selection

- Until now, we have examined single-bit data selected by a MUX. What if we want to select m-bit data/words?
 - Combine MUX blocks in parallel with common select and enable signals
- Example: Construct a logic circuit that selects between 2 sets of 4-bit inputs (see next slide for solution).

Example: Quad 2-to-1 MUX

- Uses four 4-to-1 MUXs with common select (S) and enable (E).
- Select line chooses between A_i 's and B_i 's. The selected four-wire digital signal is sent to the Y_i 's
- Enable line turns MUX on and off (E=1 is on).



Implementing Boolean functions with Multiplexers

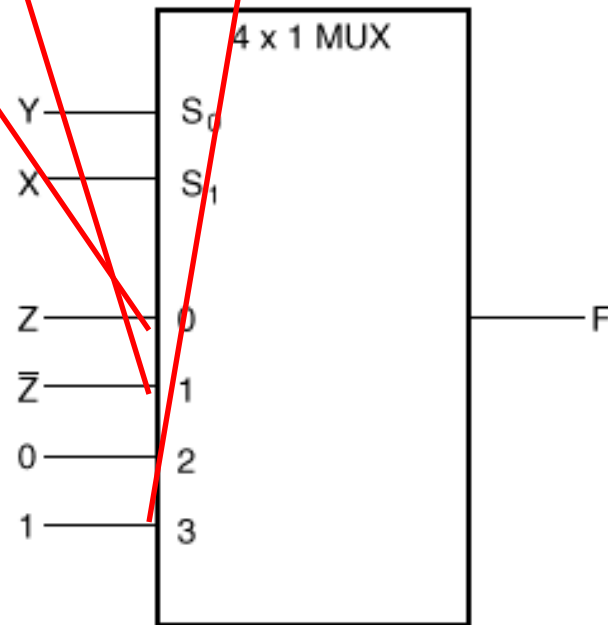
- Any Boolean function of n variables can be implemented using a 2^{n-1} -to-1 multiplexer. A MUX is basically a decoder with outputs ORed together, hence this isn't surprising.
- The SELECT signals generate the minterms of the function.
- The data inputs identify which minterms are to be combined with an OR.

Example

- $F(X,Y,Z) = X'Y'Z + X'YZ' + XYZ' + XYZ = \Sigma m(1,2,6,7)$
- There are $n=3$ inputs, thus we need a **2^2 -to-1 MUX**
- **The first $n-1$ ($=2$) inputs serve as the selection lines**

X	Y	Z	F	
0	0	0	0	$F = Z$
0	0	1	1	
0	1	0	1	$F = \bar{Z}$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(a) Truth table



(b) Multiplexer implementation

Efficient Method for implementing Boolean functions

- For an n -variable function (*e.g.*, $f(A,B,C,D)$):
 - Need a 2^{n-1} line MUX with $n-1$ select lines.
 - Enumerate function as a truth table with consistent ordering of variables (*e.g.*, A,B,C,D)
 - Attach the most significant $n-1$ variables to the $n-1$ select lines (*e.g.*, A,B,C)
 - Examine pairs of adjacent rows (only the least significant variable differs, *e.g.*, $D=0$ and $D=1$).
 - Determine whether the function output for the $(A,B,C,0)$ and $(A,B,C,1)$ combination is $(0,0)$, $(0,1)$, $(1,0)$, or $(1,1)$.
 - Attach 0 , D , D' , or 1 to the data input corresponding to (A,B,C) respectively.

The Other Example

- Consider $F(A,B,C) = \sum m(1,3,5,6)$. We can implement this function using a 4-to-1 MUX as follows.
- The index is ABC. Apply A and B to the S_1 and S_0 selection inputs of the MUX (A is most sig, S_1 is most sig.)
- Enumerate function in a truth table.

MUX Example (cont.)

When $A=B=0$, $F=C$

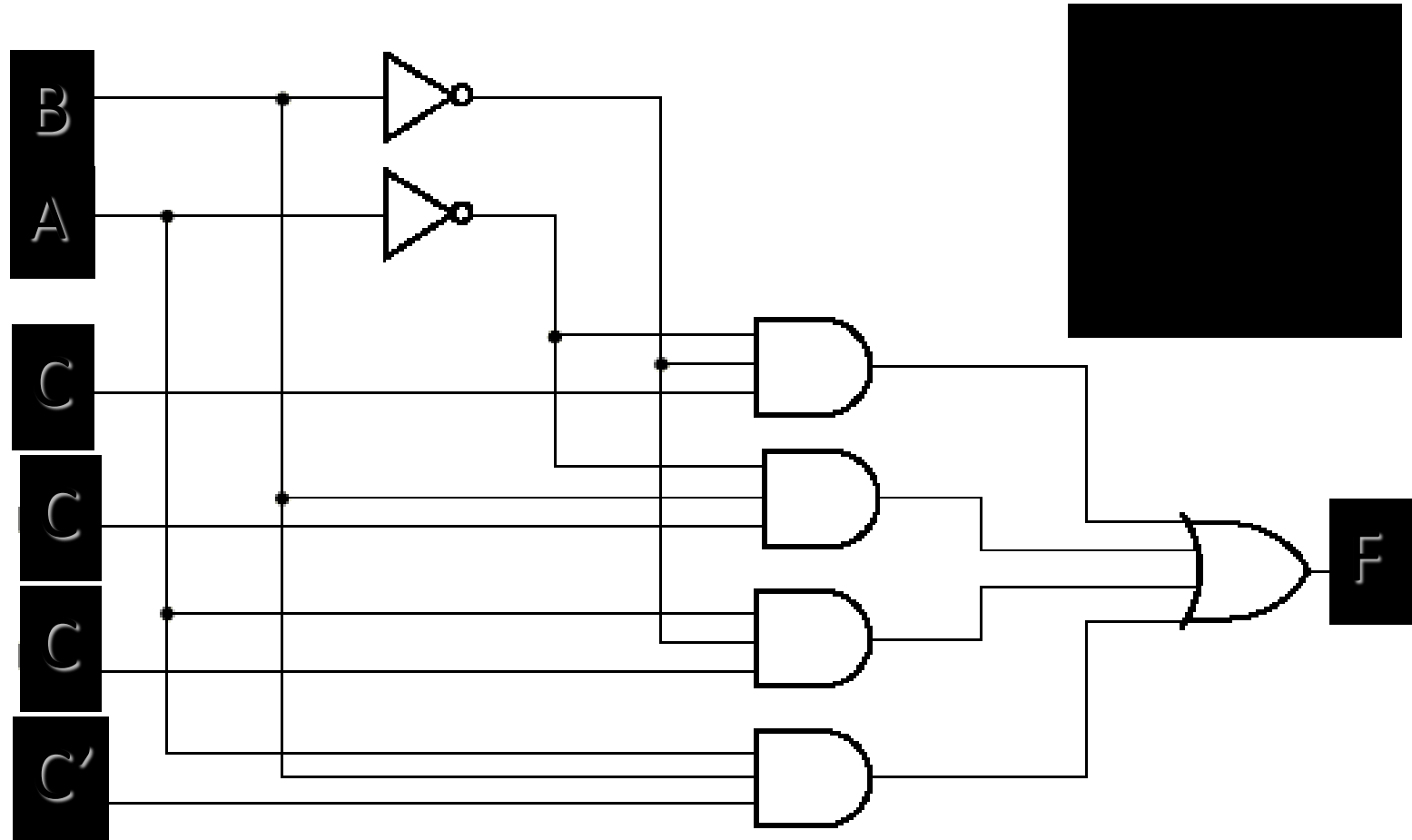
When $A=0$, $B=1$, $F=C$

When $A=1$, $B=0$, $F=C$

When $A=B=1$, $F=C'$

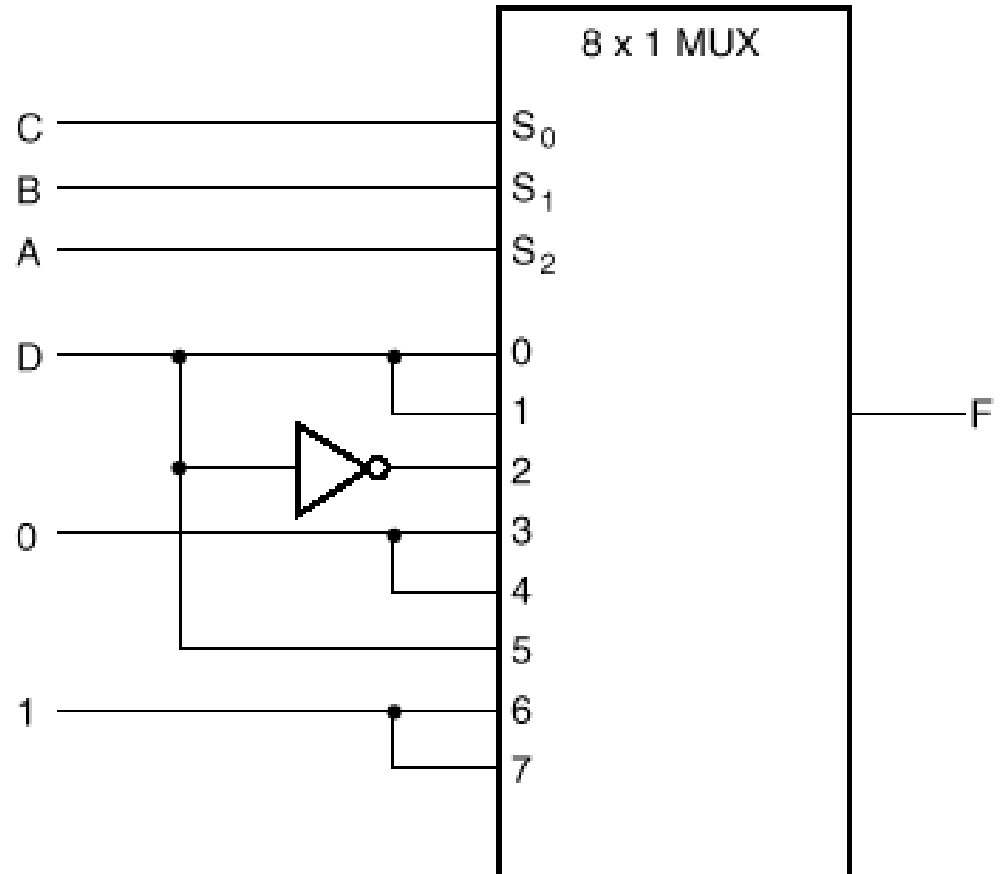
A	B	C		F
0	0	0		0
0	0	1		1
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		1
1	1	1		0

MUX implementation of $F(A,B,C) = \sum m(1,3,5,6)$



A larger Example

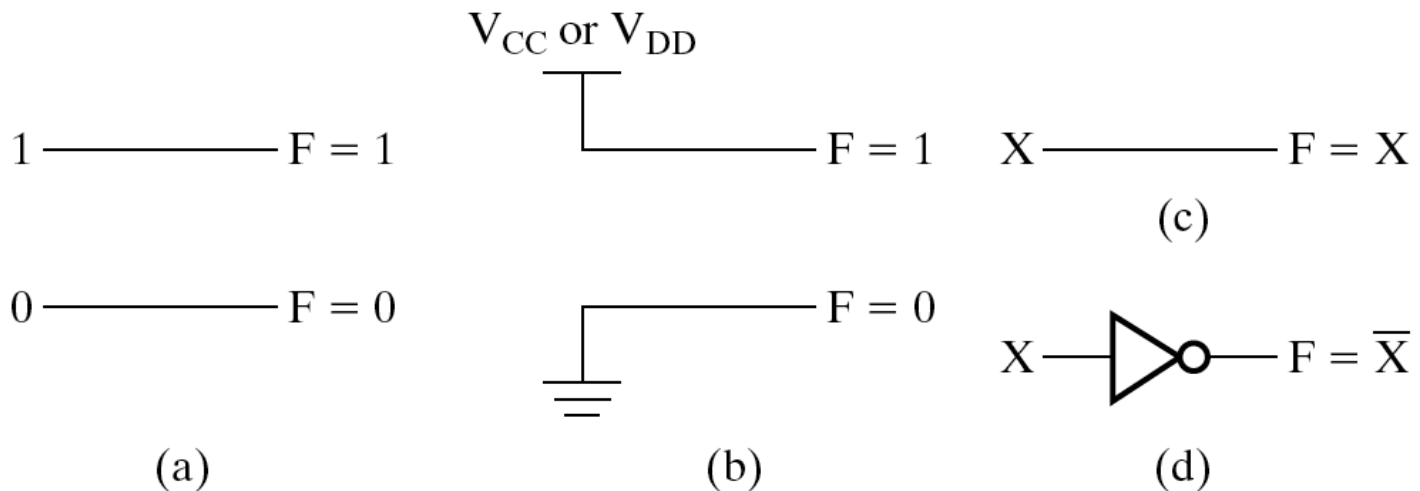
A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = \bar{D}$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



Rudimentary Functions

□ **TABLE 4-1**
Functions of One Variable

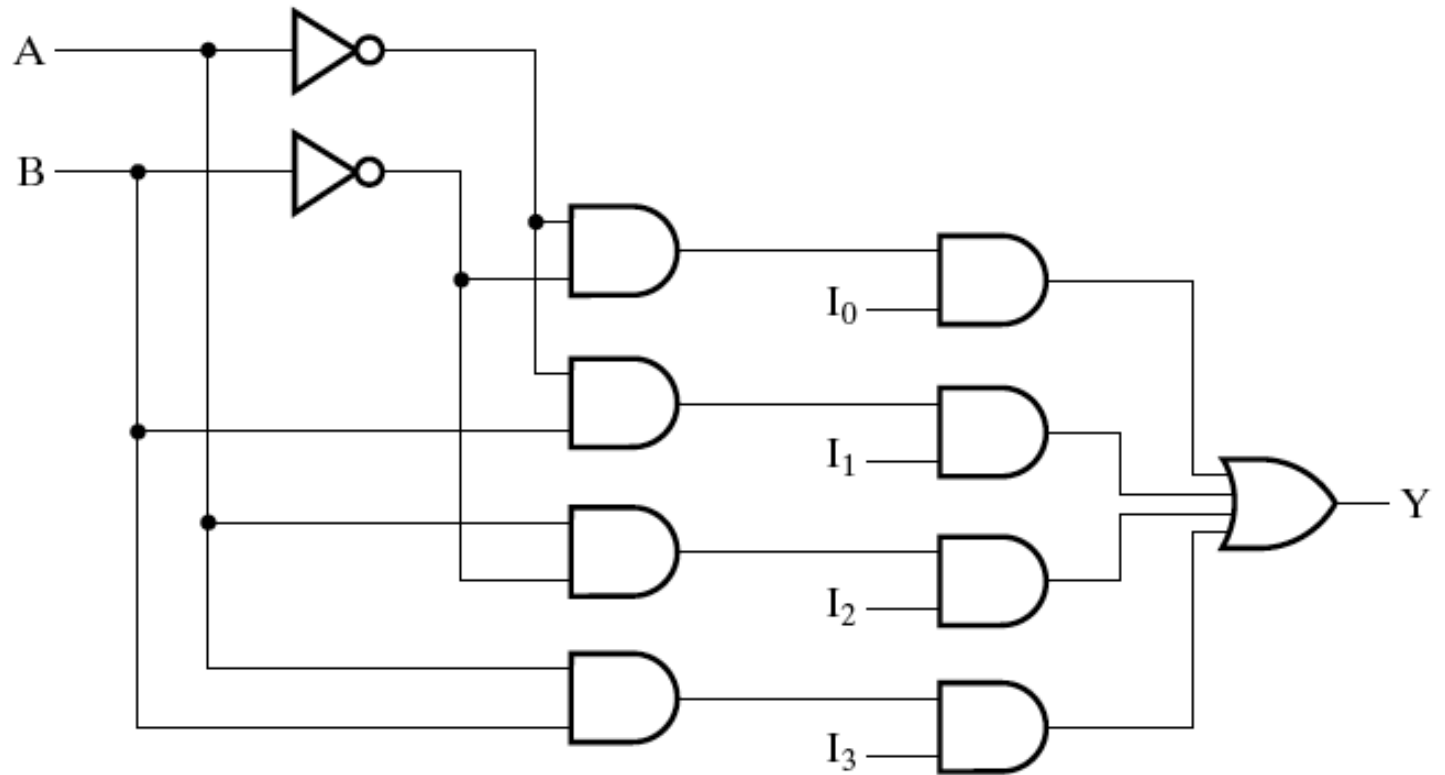
X	F = 0	F = X	F = \bar{X}	F = 1
0	0	0	1	1
1	0	1	0	1



Selection

A	B	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

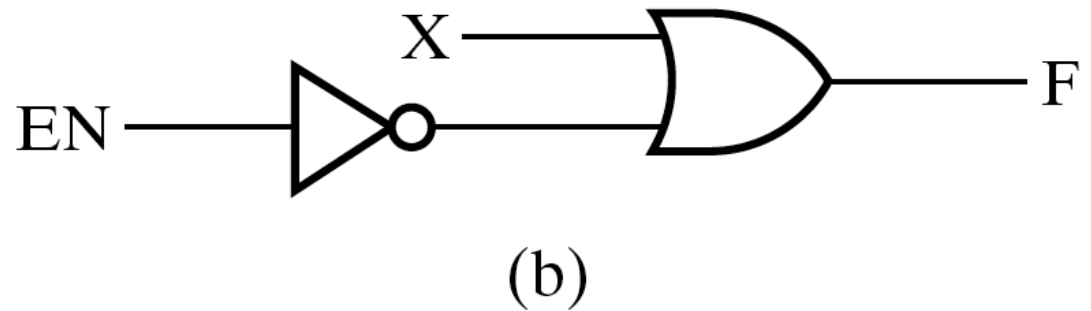
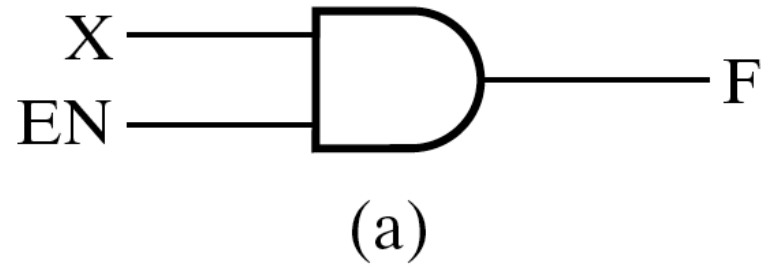
(a)



(b)

Enabling

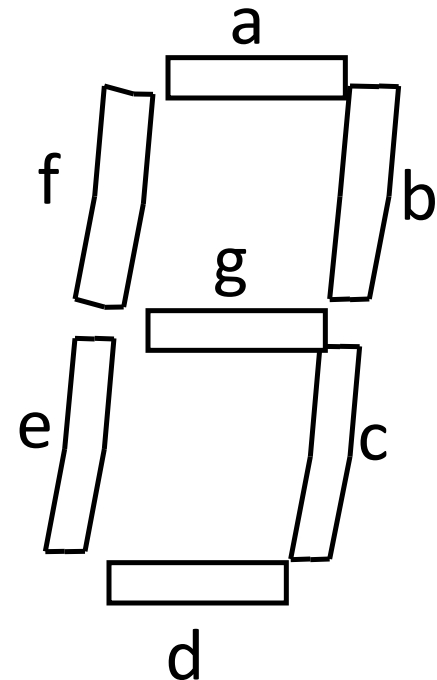
- “gating” ?



The Other Code Converter

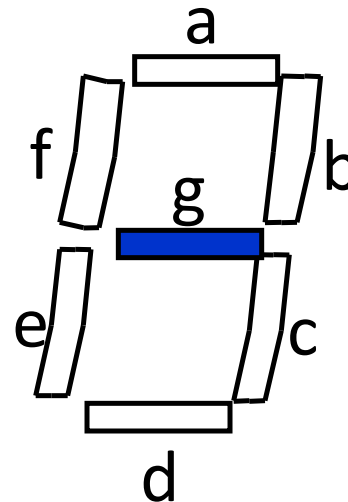
BCD-to-Seven-Segment Converter

- Seven-segment display:
 - 7 LEDs (light emitting diodes), each one controlled by an input
 - 1 means “on”, 0 means “off”
 - Display digit “3”?
 - Set a, b, c, d, g to 1
 - Set e, f to 0



BCD-to-Seven-Segment Converter

- Input is a 4-bit BCD code \rightarrow 4 inputs (w, x, y, z).
- Output is a 7-bit code (a,b,c,d,e,f,g) that allows for the decimal equivalent to be displayed.
- Example:
 - Input: 0000_{BCD}
 - Output: 1111110
(a=b=c=d=e=f=1, g=0)



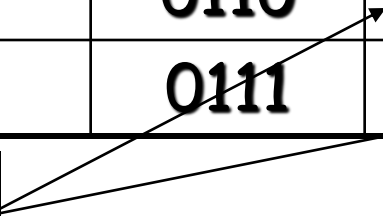
BCD-to-Seven-Segment (cont.)

Truth Table

Digit	wxyz	abcdefg
0	0000	111110
1	0001	0110000
2	0010	1101101
3	0011	1111001
4	0100	0110011
5	0101	1011011
6	0110	X011111
7	0111	11100X0

Digit	wxyz	abcdefg
8	1000	1111111
9	1001	111X011
	1010	XXXXXXXXX
	1011	XXXXXXXXX
	1100	XXXXXXXXX
	1101	XXXXXXXXX
	1110	XXXXXXXXX
	1111	XXXXXXXXX

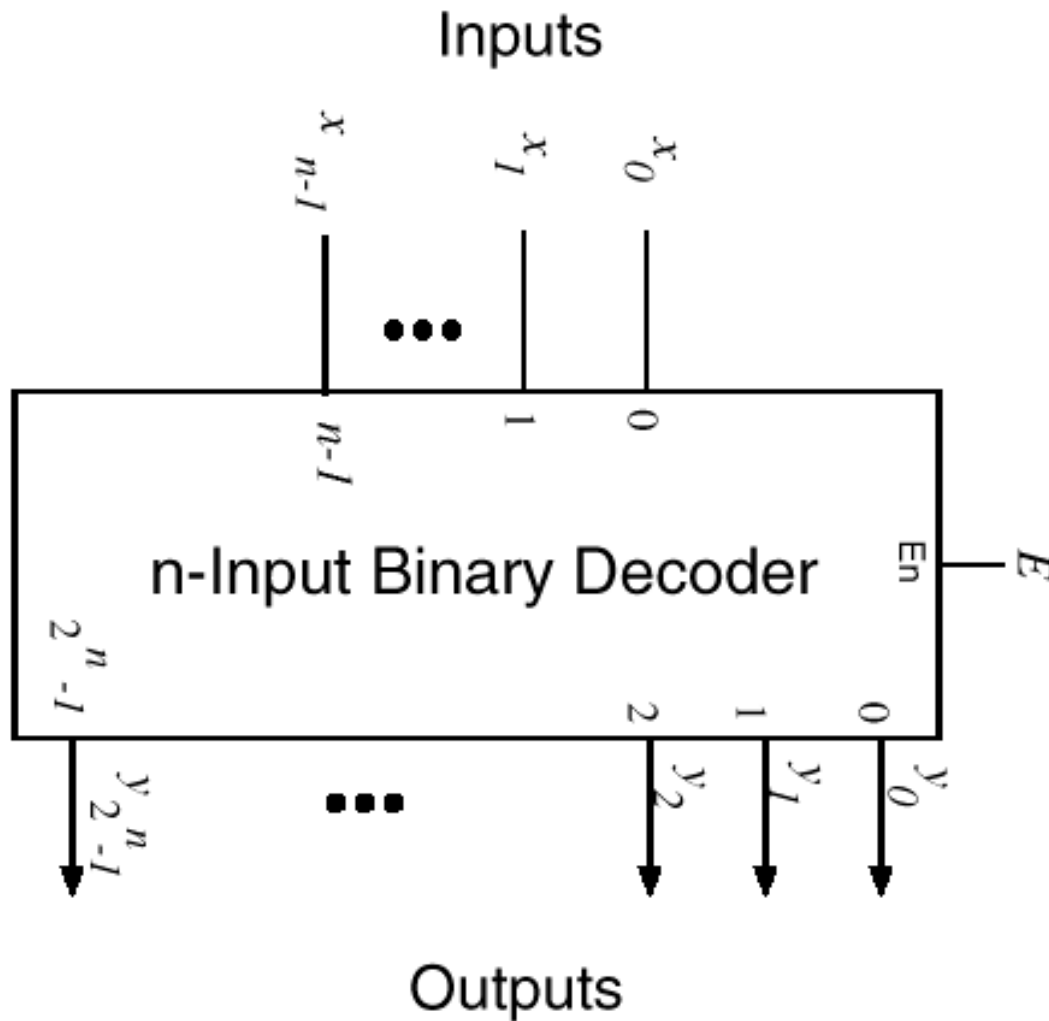
??



Decoders

- A combinational circuit that converts binary information from n coded inputs to a maximum 2^n coded outputs
→ n -to- 2^n decoder
- n -to- m decoder, $m \leq 2^n$
- Examples: BCD-to-7-segment decoder, where $n=4$ and $m=10$

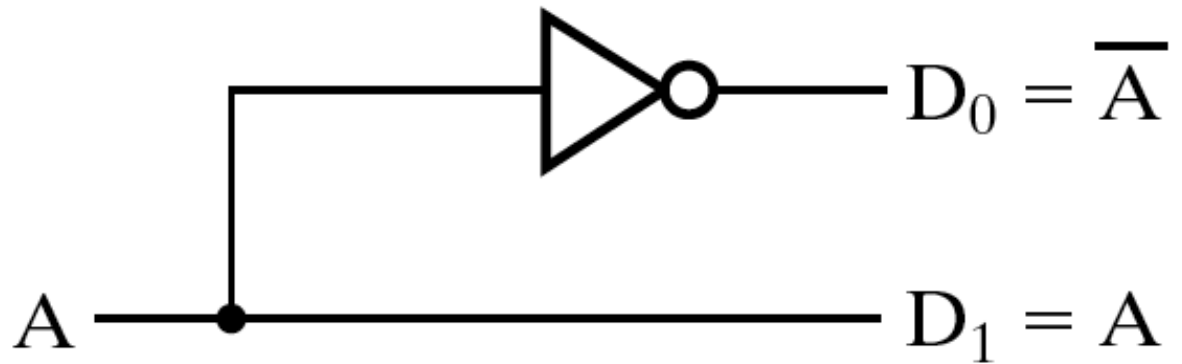
Decoders (cont.)



1-2 Decoder

A	D₀	D₁
0	1	0
1	0	1

(a)

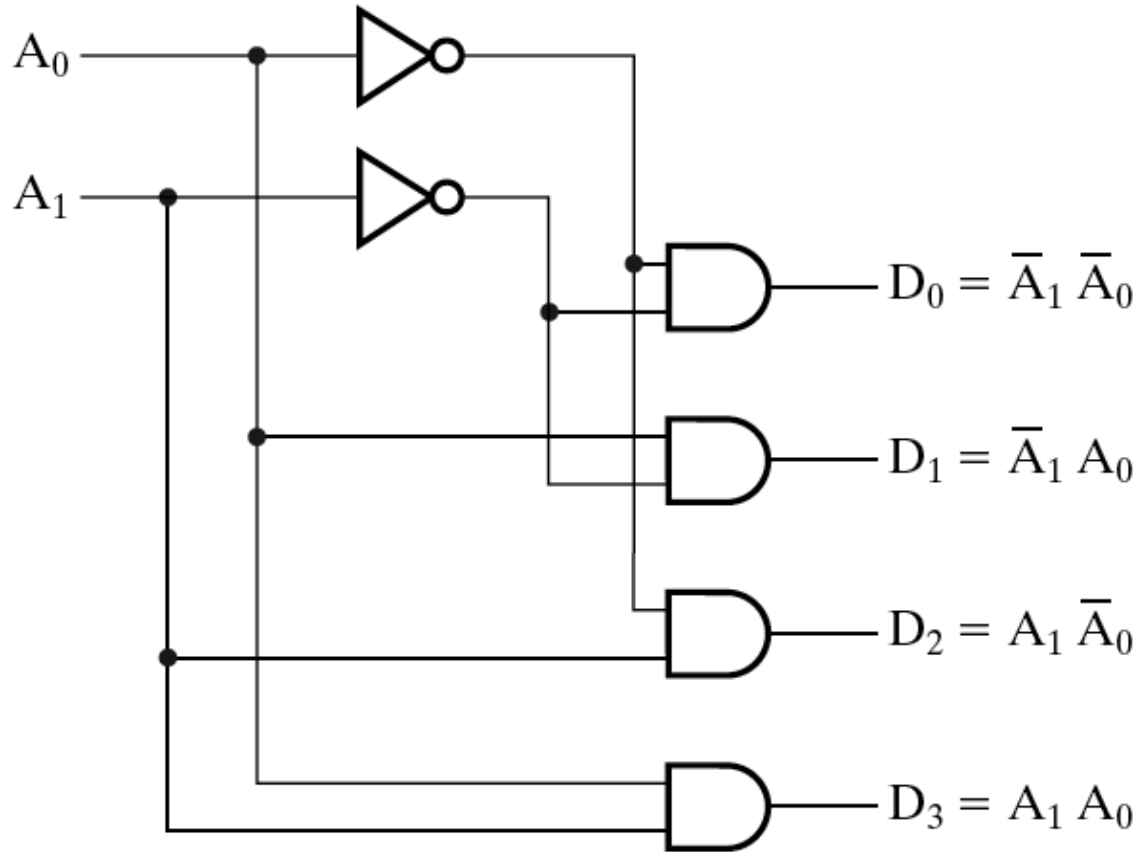


(b)

2-to-4 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)

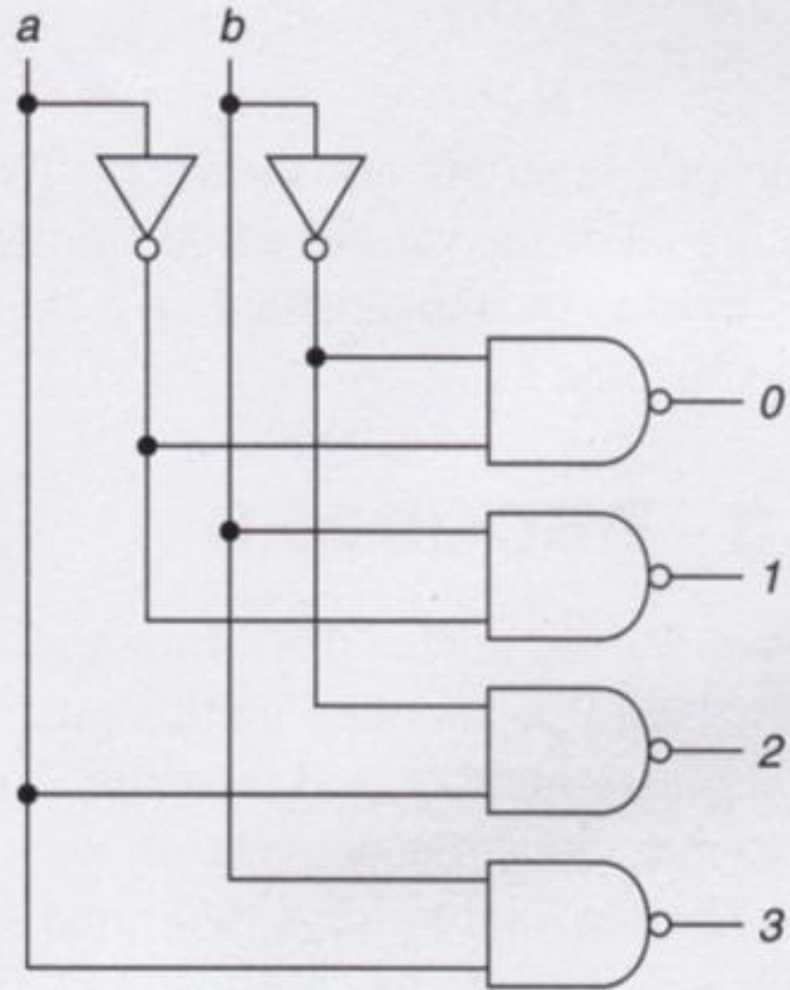


(b)

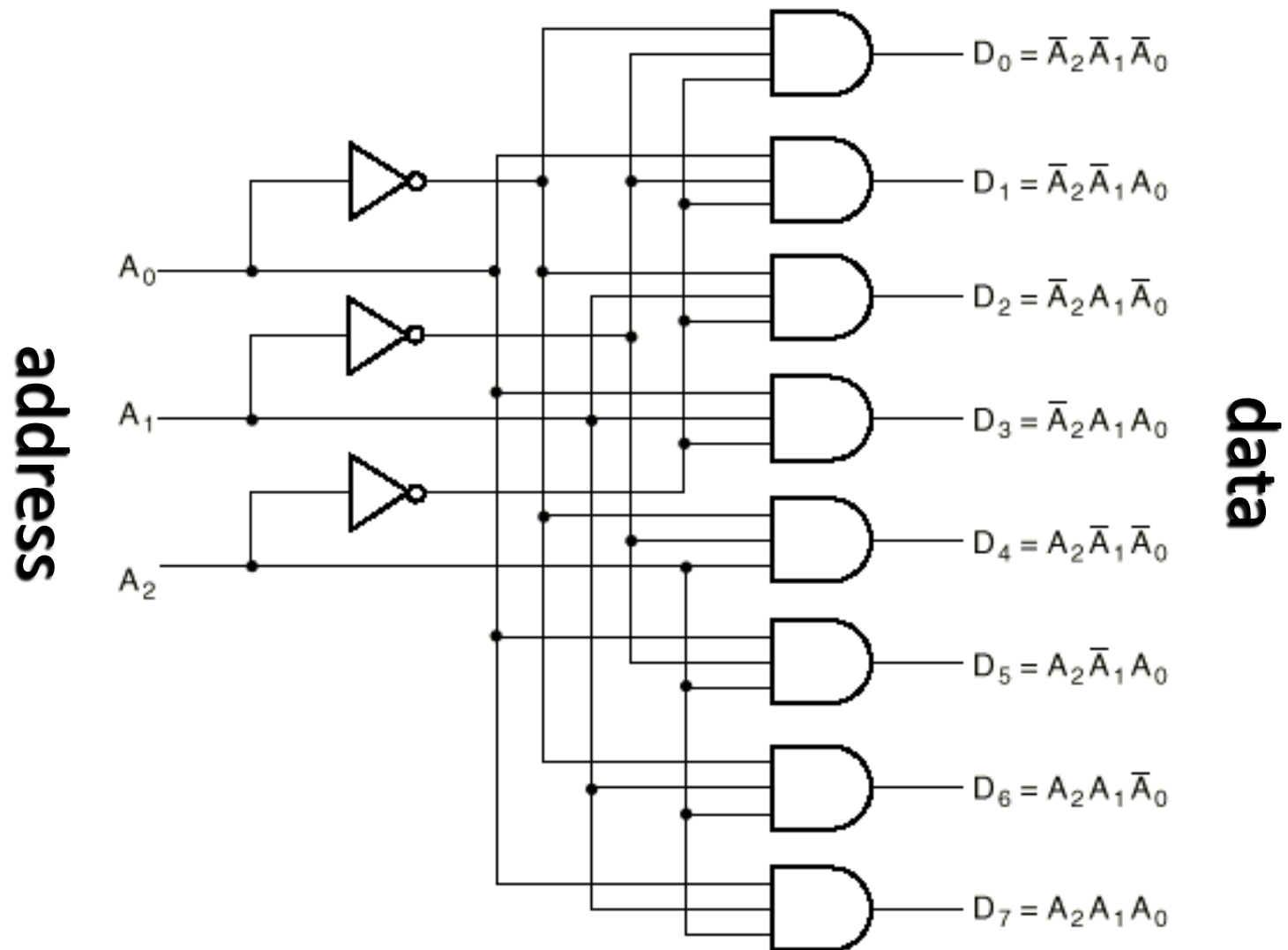
2-to-4 Active Low Decoder

Figure 4.5 An active low decoder.

<i>a</i>	<i>b</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0



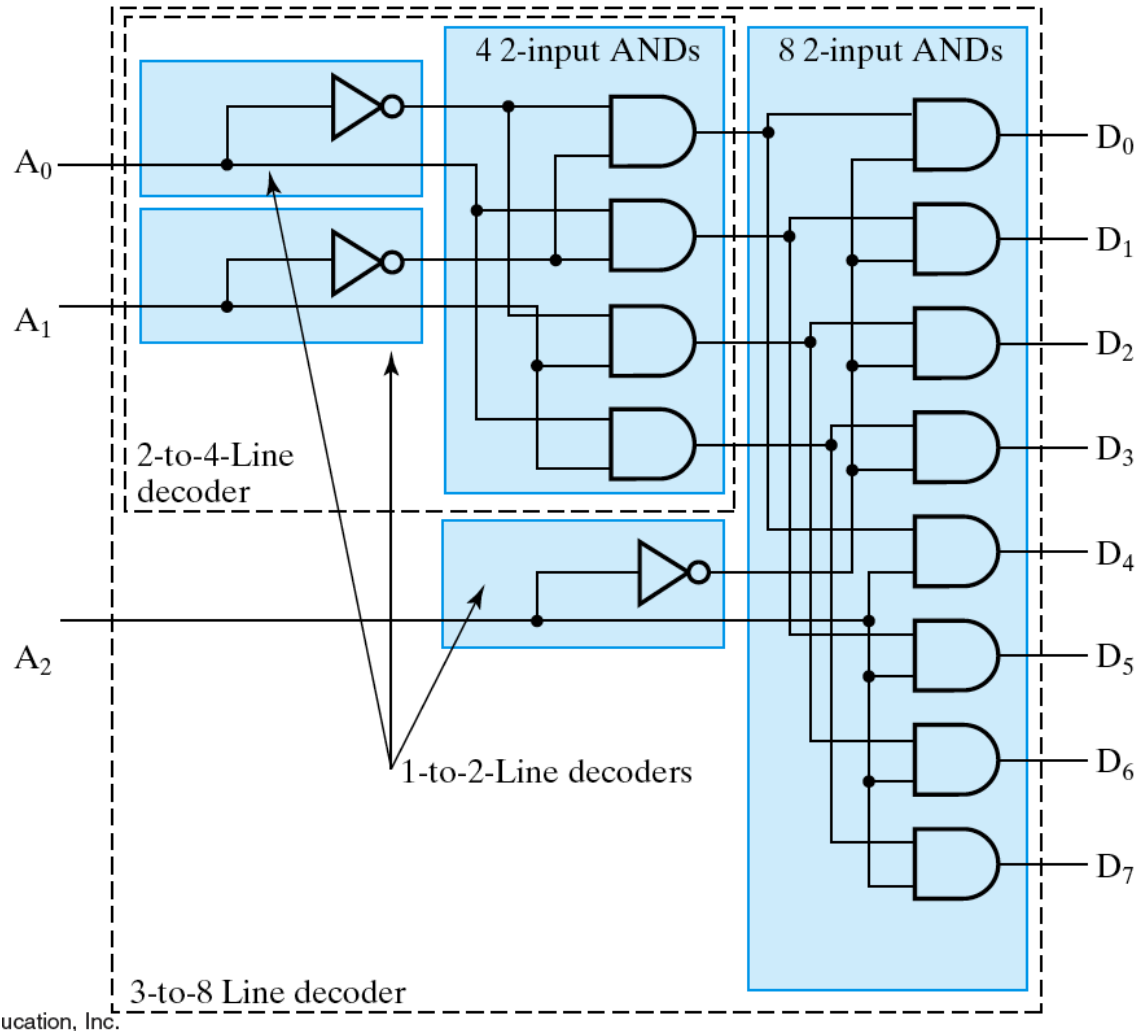
3-to-8 Decoder



3-to-8 Decoder (cont.)

- Three inputs, A_0 , A_1 , A_2 , are decoded into eight outputs, D_0 through D_7
- Each output D_i represents one of the minterms of the 3 input variables.
- $D_i = 1$ when the binary number $A_2A_1A_0 = i$
- Shorthand: $D_i = m_i$
- The output variables are mutually exclusive; exactly one output has the value 1 at any time, and the other seven are 0.

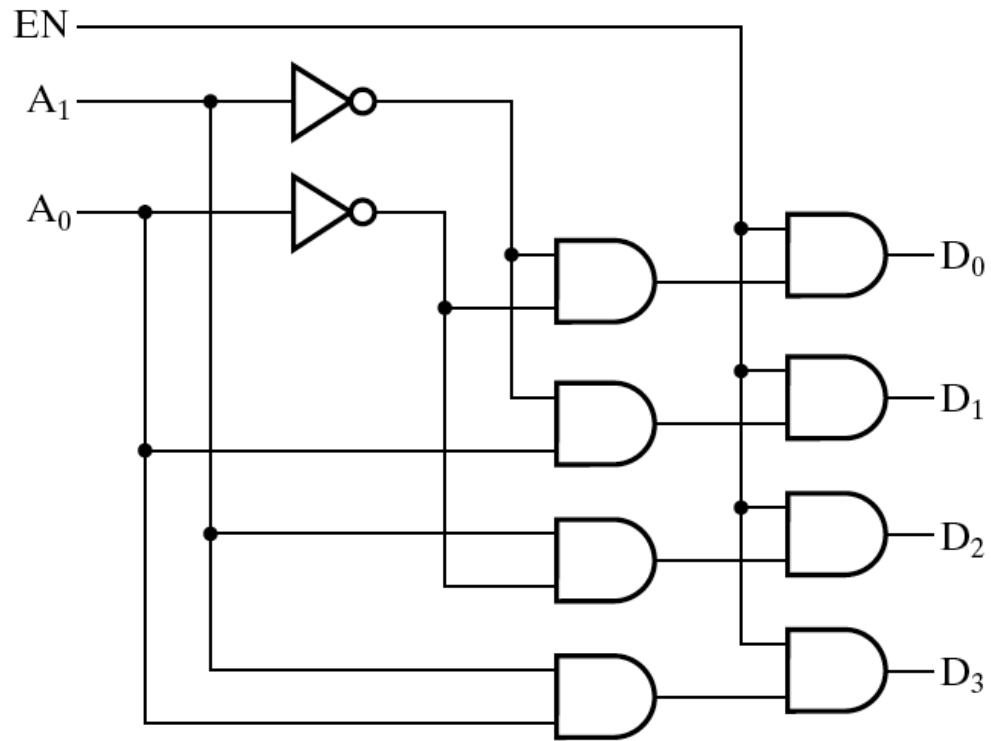
Decoder Expansion



Decoder with enable

EN	A ₁	A ₀	D ₀	D ₁	D ₂	D ₃
0	X	X	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

(a)

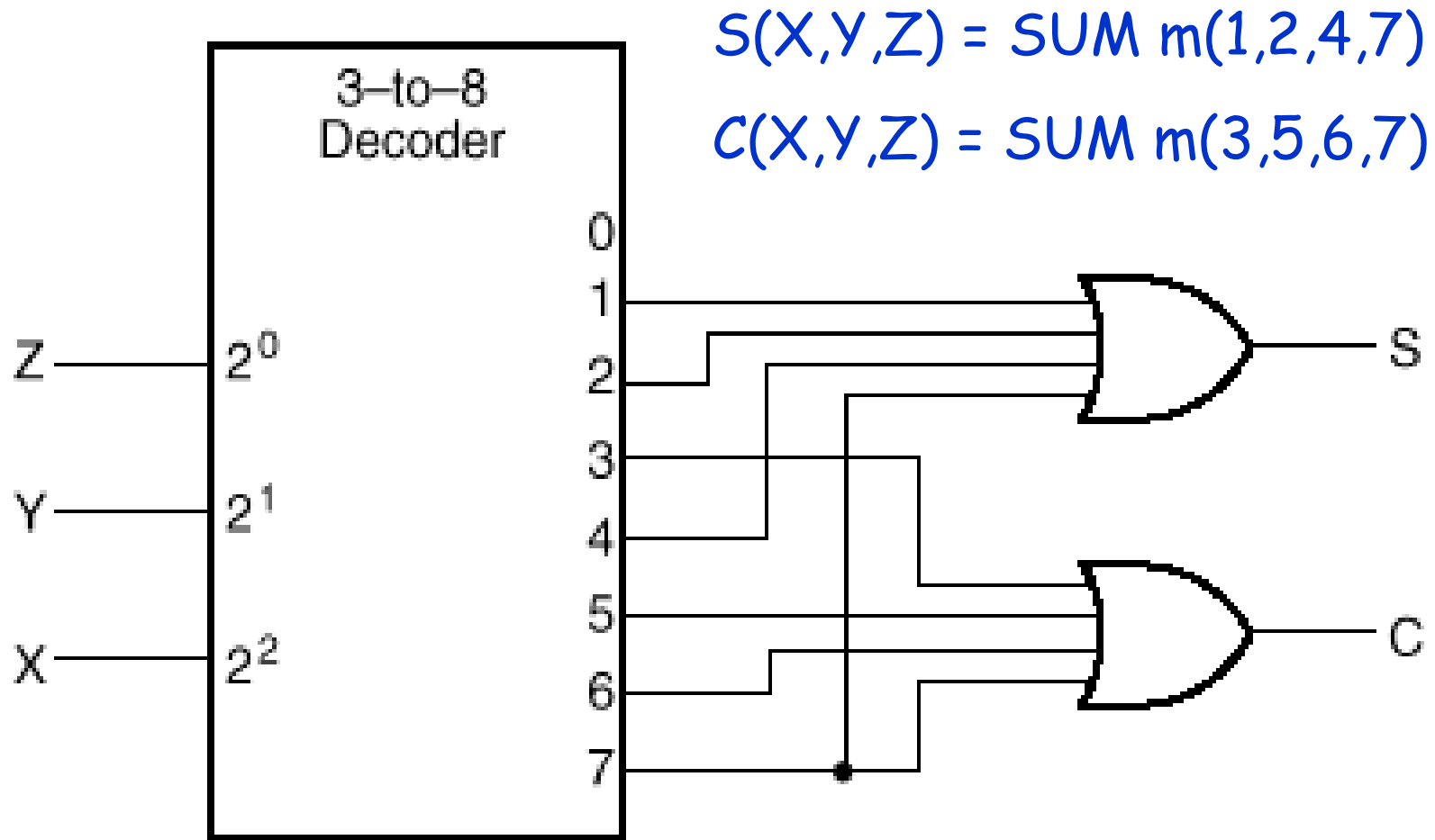


(b)

Implementing Boolean functions using decoders

- Any combinational circuit can be constructed using decoders and OR gates! Why?
- Here is an example:
Implement a full adder circuit with a decoder and two OR gates.
- Recall full adder equations, and let X, Y, and Z be the inputs:
 - $S(X,Y,Z) = X+Y+Z = \Sigma m(1,2,4,7)$
 - $C(X,Y,Z) = \Sigma m(3, 5, 6, 7)$.
- Since there are 3 inputs and a total of 8 minterms, we need a 3-to-8 decoder.

Implementing a Binary Adder Using a Decoder



Encoders

- An encoder is a digital circuit that performs the inverse operation of a decoder. An encoder has 2^n input lines and n output lines.
- The output lines generate the binary equivalent to the input line whose value is 1.

Encoders (cont.)

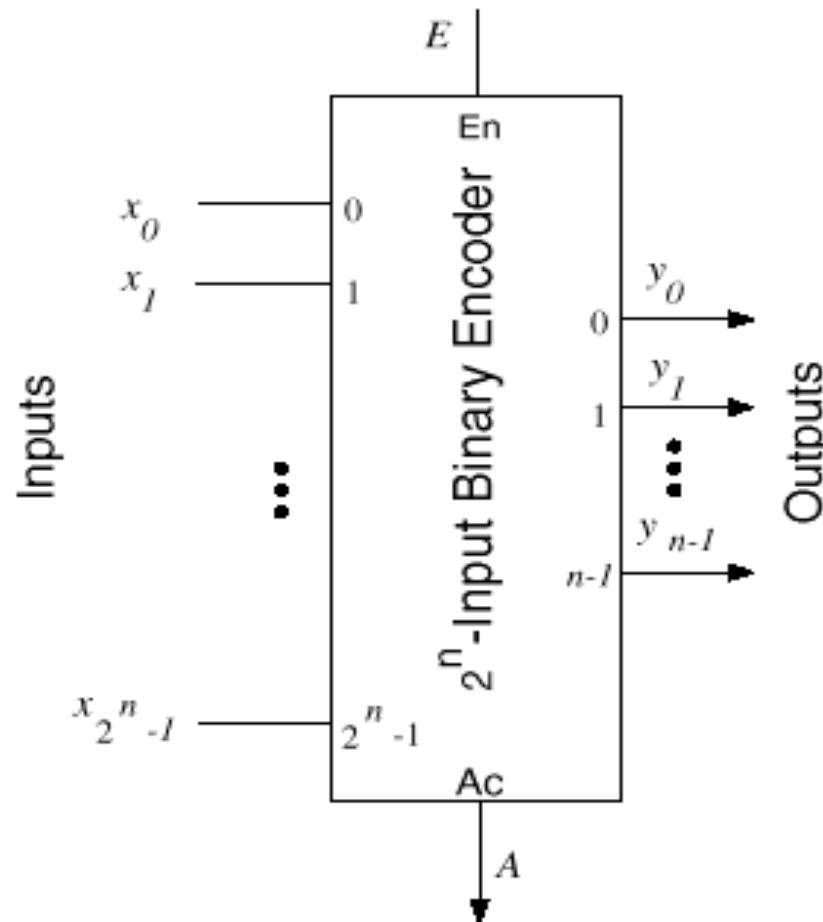


Figure 9.12: 2^n -input binary encoder.

Encoder Example

- Example: 8-to-3 binary encoder (octal-to-binary)

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$A_0 = D_1 + D_3 + D_5 + D_7$$

$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

Encoder Example (cont.)

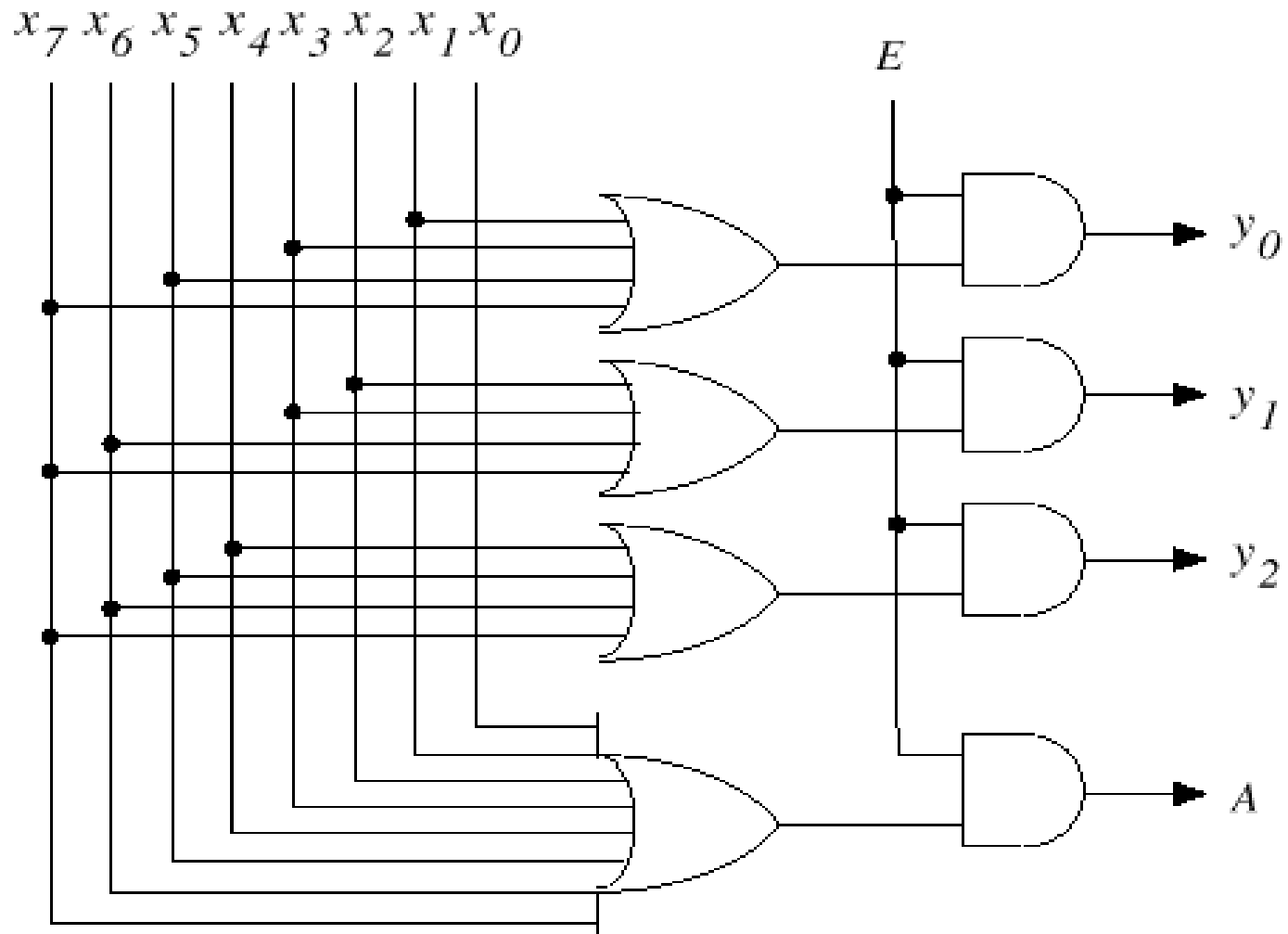


Figure 9.13: Implementation of an 8-input binary encoder.

Encoder Design Issues

- There are two ambiguities associated with the design of a simple encoder:
 1. Only one input can be active at any given time. If two inputs are active simultaneously, the output produces an undefined combination (for example, if D_3 and D_6 are 1 simultaneously, the output of the encoder will be 111).
 2. An output with all 0's can be generated when all the inputs are 0's, or when D_0 is equal to 1.

Priority Encoders

- Solves the ambiguities mentioned above.
- Multiple asserted inputs are allowed; one has priority over all others.
- Separate indication of no asserted inputs.

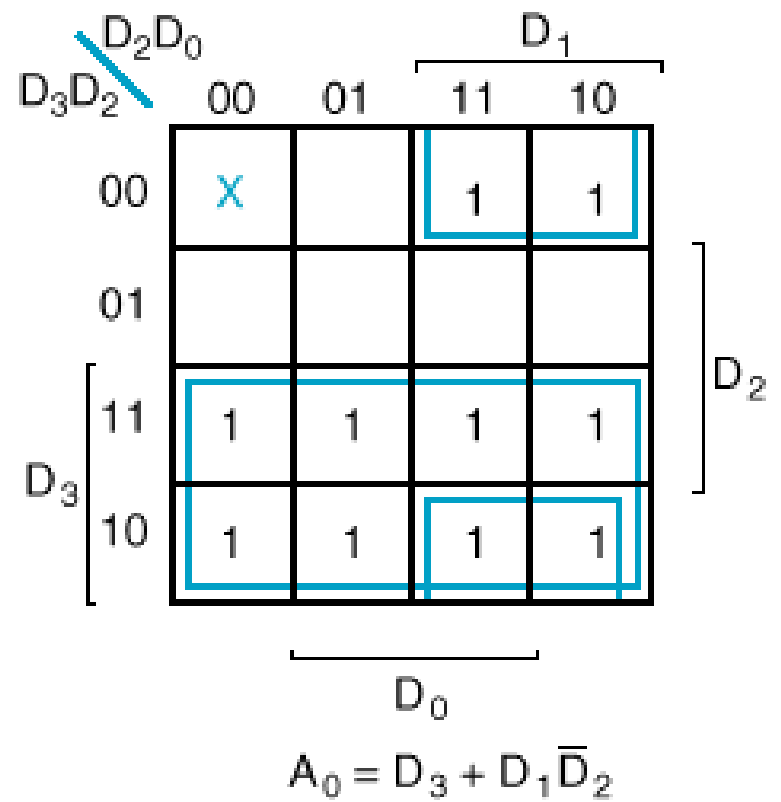
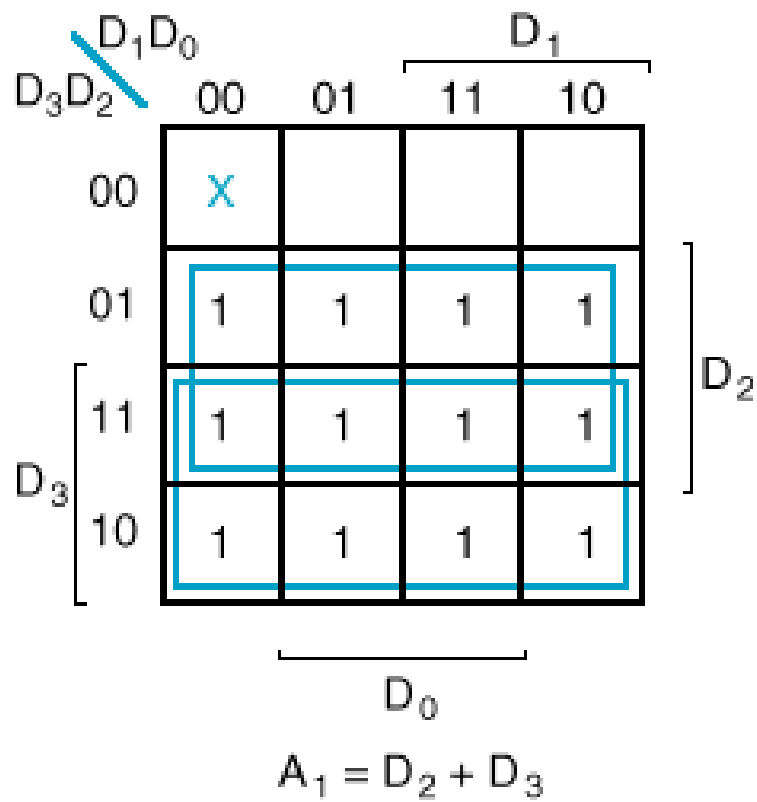
Example: 4-to-2 Priority Encoder Truth Table

Inputs				Outputs		
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	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

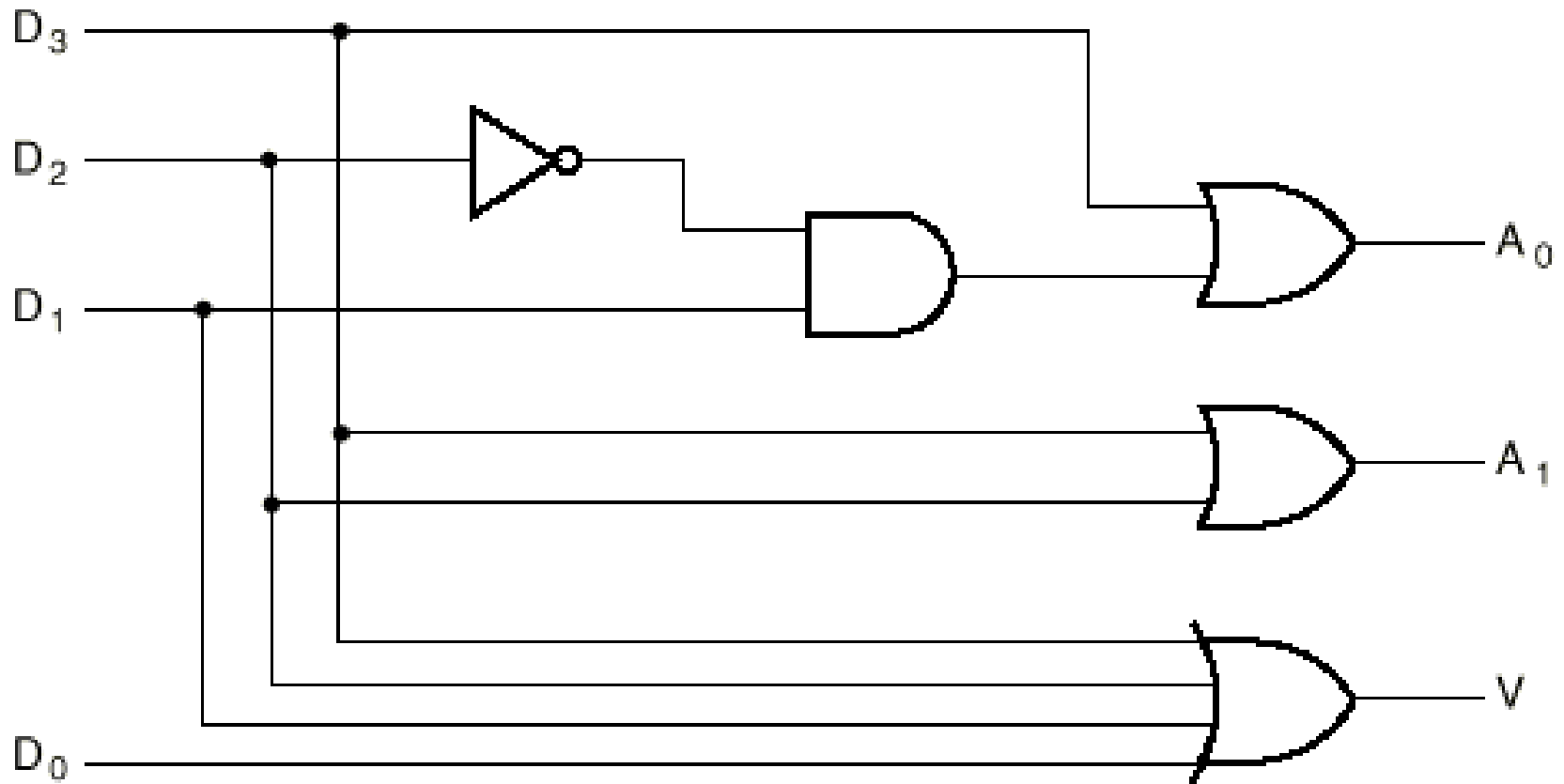
4-to-2 Priority Encoder (cont.)

- The operation of the priority encoder is such that:
- If two or more inputs are equal to 1 at the same time, the input in the highest-numbered position will take precedence.
- A ***valid output indicator***, designated by V , is set to 1 only when one or more inputs are equal to 1. $V = D_3 + D_2 + D_1 + D_0$ by inspection.

Example: 4-to-2 Priority Encoder K-Maps



Example: 4-to-2 Priority Encoder Logic Diagram



8-to-3 Priority Encoder

Table 4.6 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

Uses of priority encoders (cont.)

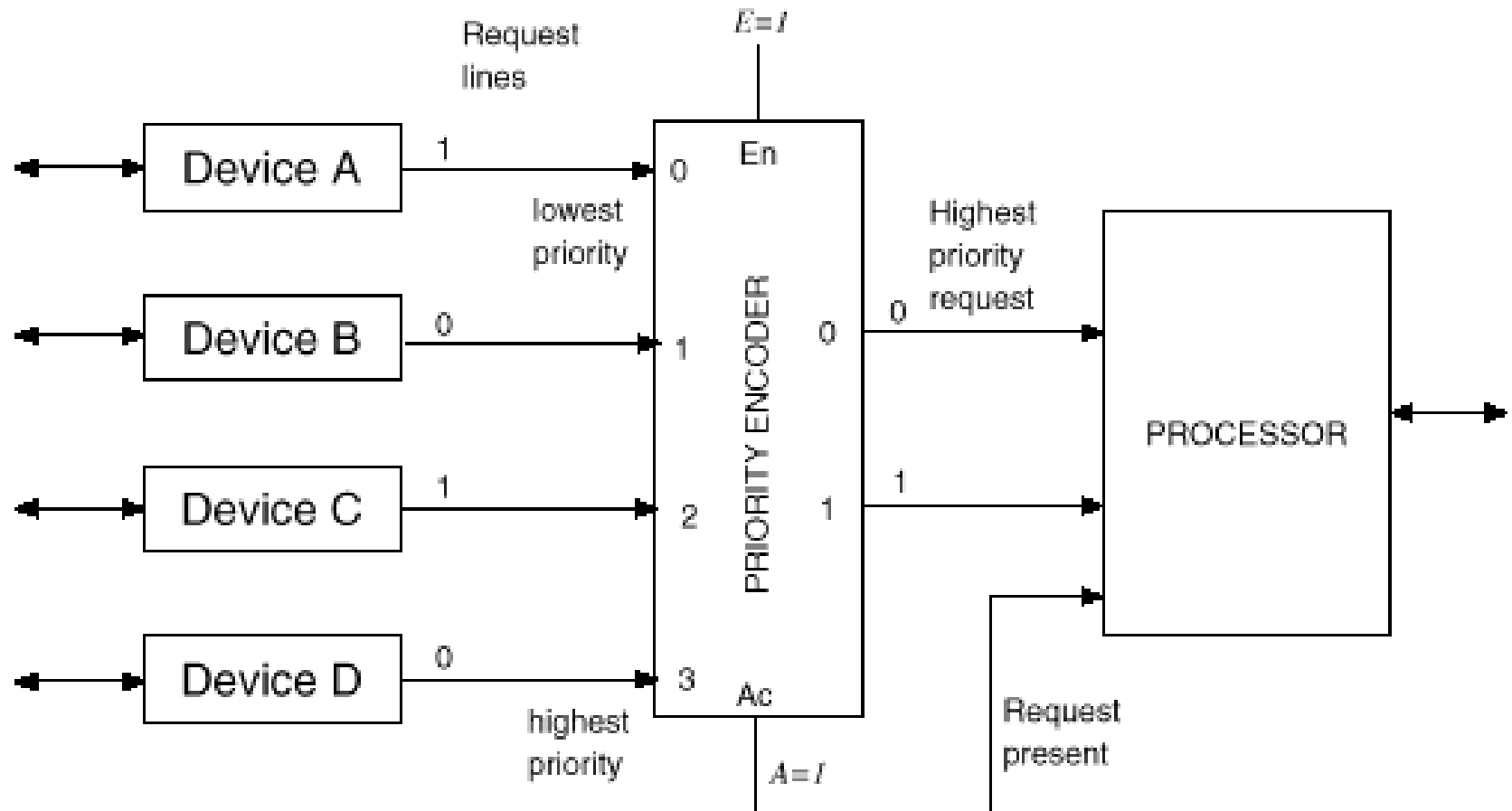


Figure 9.17: Resolving interrupt requests using a priority encoder.

Link Between Multiplexer and Decoder

- Note the regions of the multiplexer
 - 1-to-2-line Decoder
 - 2 Enabling circuits
 - 2-input OR gate
- In general, for an 2^n -to-1-line multiplexer:
 - n -to- 2^n -line decoder
 - 2^n AND gates

