CDA 3200 Digital Systems

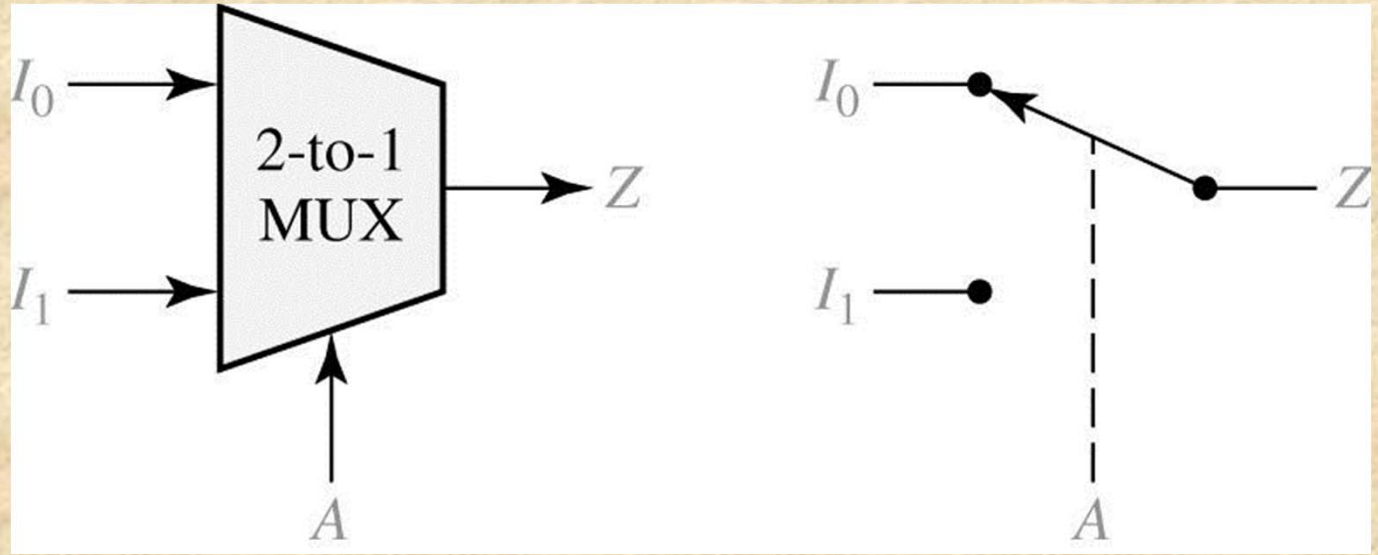
Instructor: Dr. Janusz Zalewski
Developed by: Dr. Dahai Guo
Spring 2012

Outline

- Multiplexers
- Three-State Buffers
- Decoders and Encoders
- Read-Only Memories
- Programmable Logic Devices

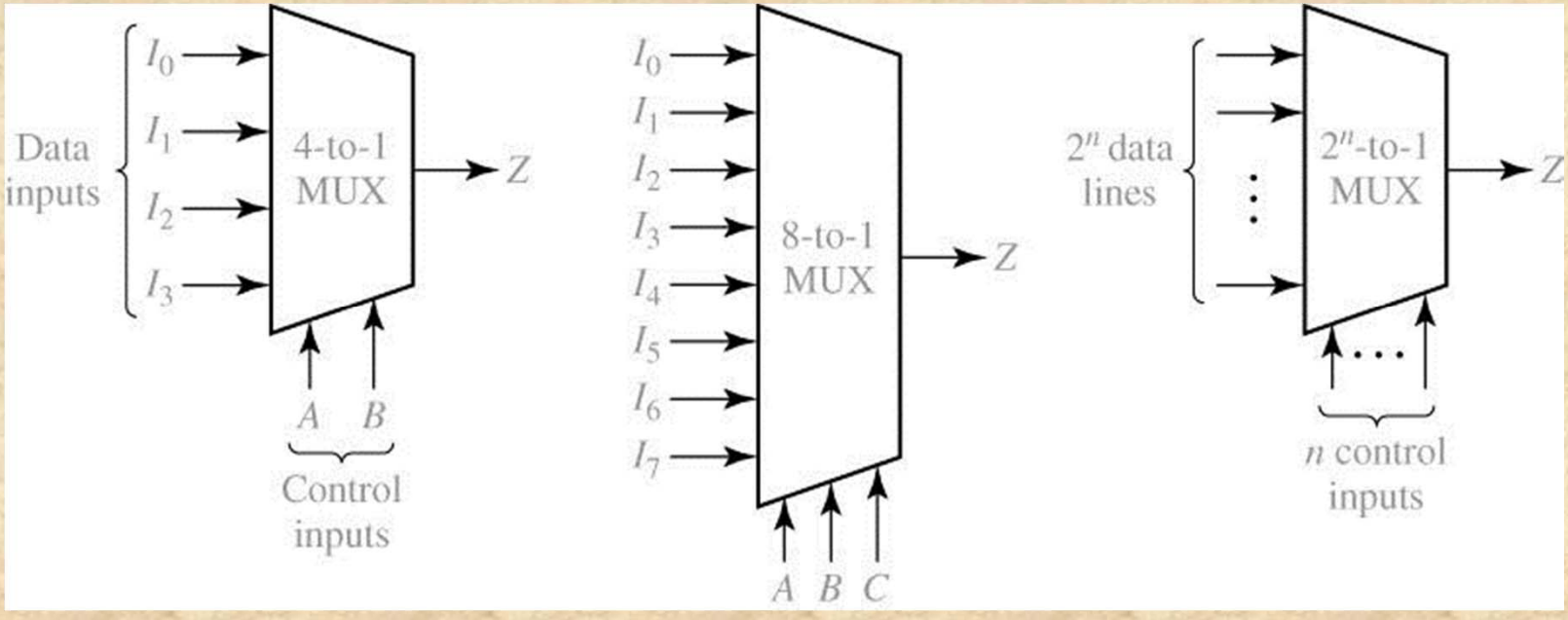
Multiplexers (1/4)

- A multiplexer (MUX) has a group of data inputs and a group of control inputs.
- The control inputs are used to select one of the data inputs.



A=0→Z=I_0
A=1→Z=I_1

Multiplexers (2/4)



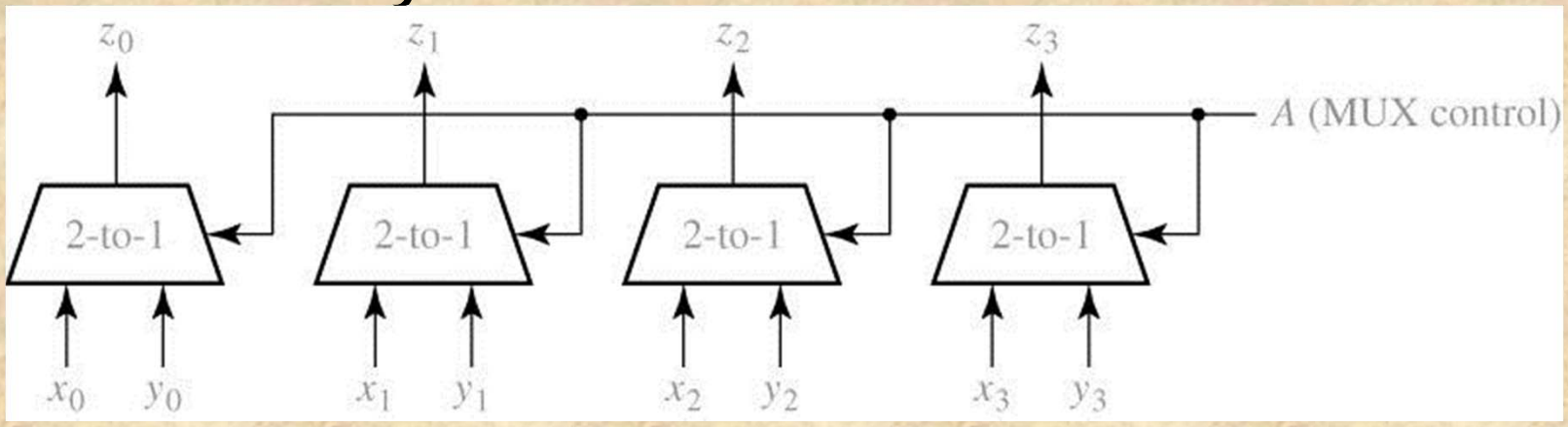
- $Z_{4\text{-to-}1}=A'B'I_0+A'BI_1+AB'I_2+ABI_3$
- $Z_{8\text{-to-}1}=A'B'C'I_0+A'B'CI_1+A'BC'I_2+A'BCI_3+AB'C'I_4+AB'CI_5+ABC'I_6+ABCI_7$

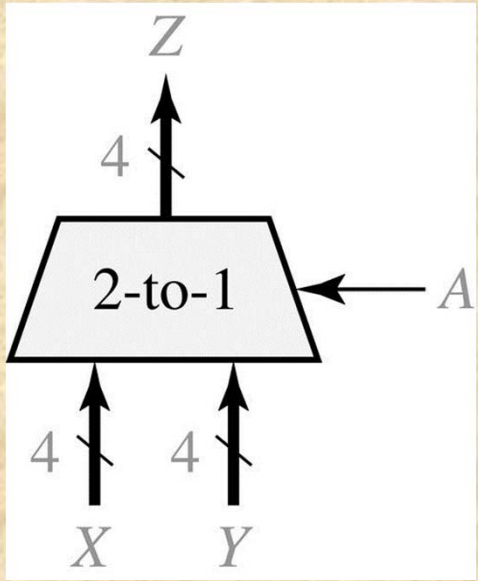
Multiplexers (3/4)

- n control inputs can select 2^n inputs.
- The logic expression for an n-control-input MUX is $Z = \sum_{k=0}^{2^n-1} m_k I_k$, where m_k is the minterm whose decimal representation is k

Multiplexers (4/4)

- Several multiplexers can be used to select from binary numbers.

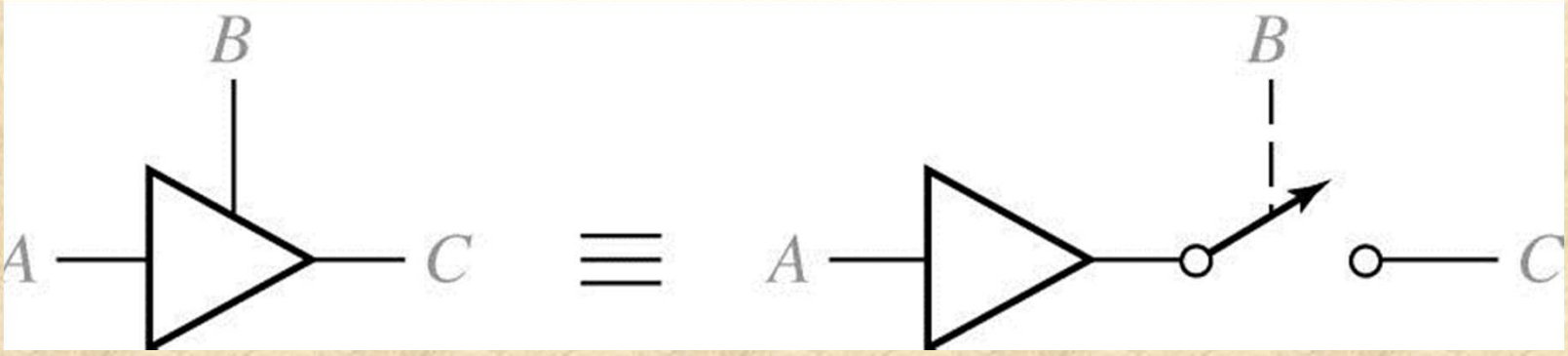




Bus Inputs/Outputs

Three-State Buffers (1/5)

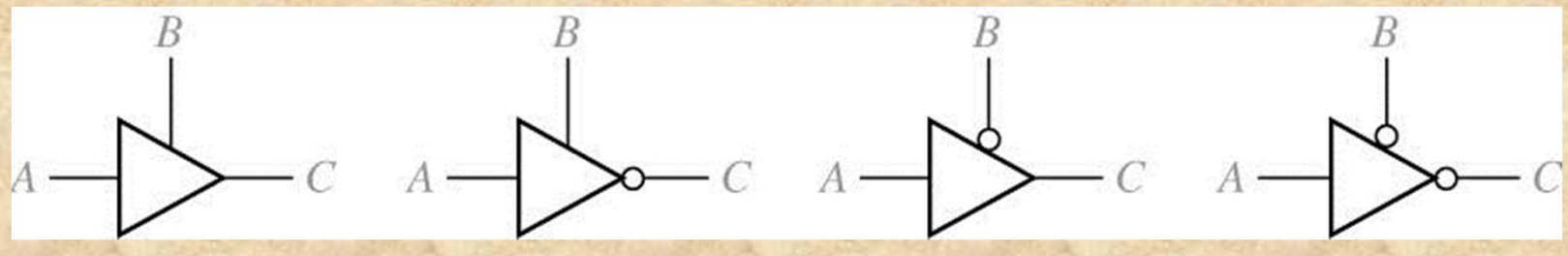
- A three-state buffer has an enable input



B	A	C
0	0	Z
0	1	Z
1	0	0
1	1	1

The symbol Z represent the buffer is in the high-impedance (Hi-Z) state.

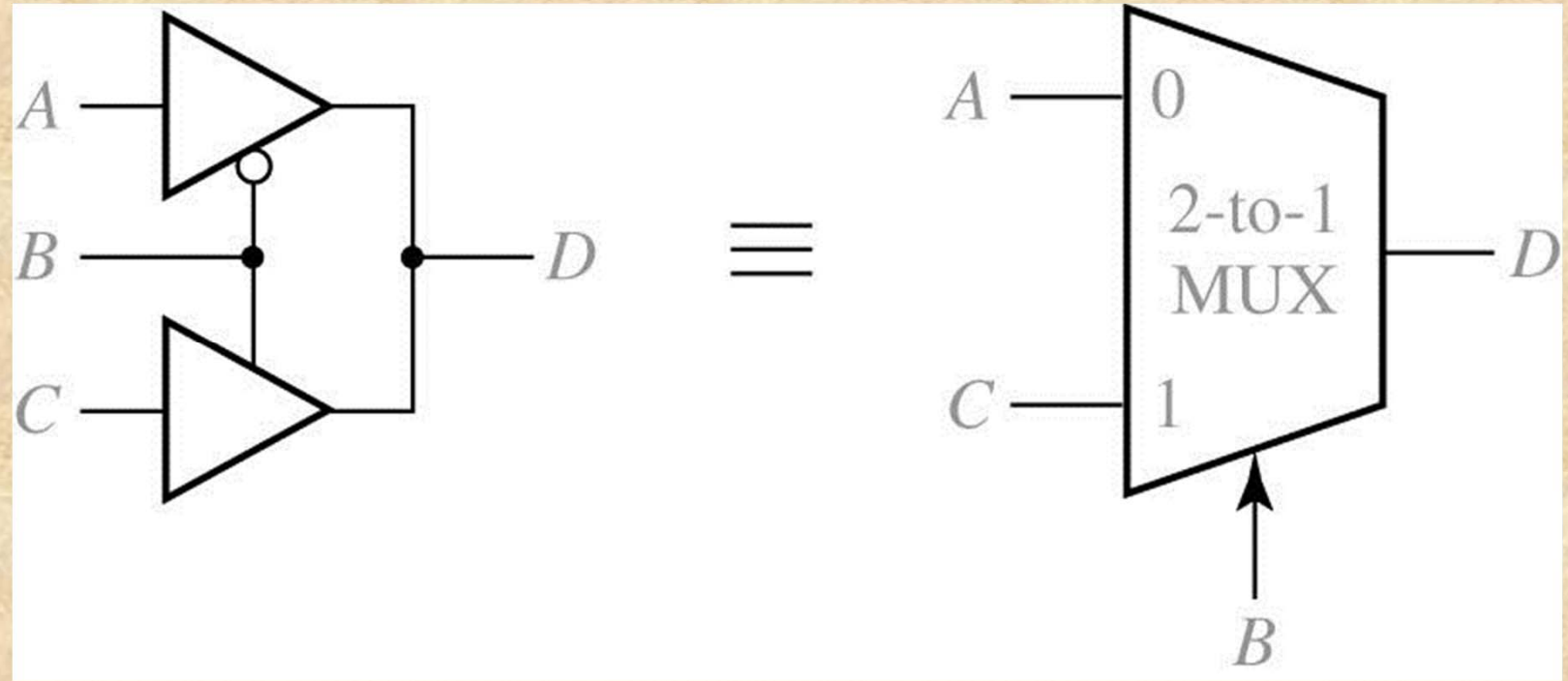
Three-State Buffers (2/5)



B	A	C	B	A	C	B	A	C	B	A	C
0	0	Z	0	0	Z	0	0	0	0	0	1
0	1	Z	0	1	Z	0	1	1	0	1	0
1	0	0	1	0	1	1	0	Z	1	0	Z
1	1	1	1	1	0	1	1	Z	1	1	Z

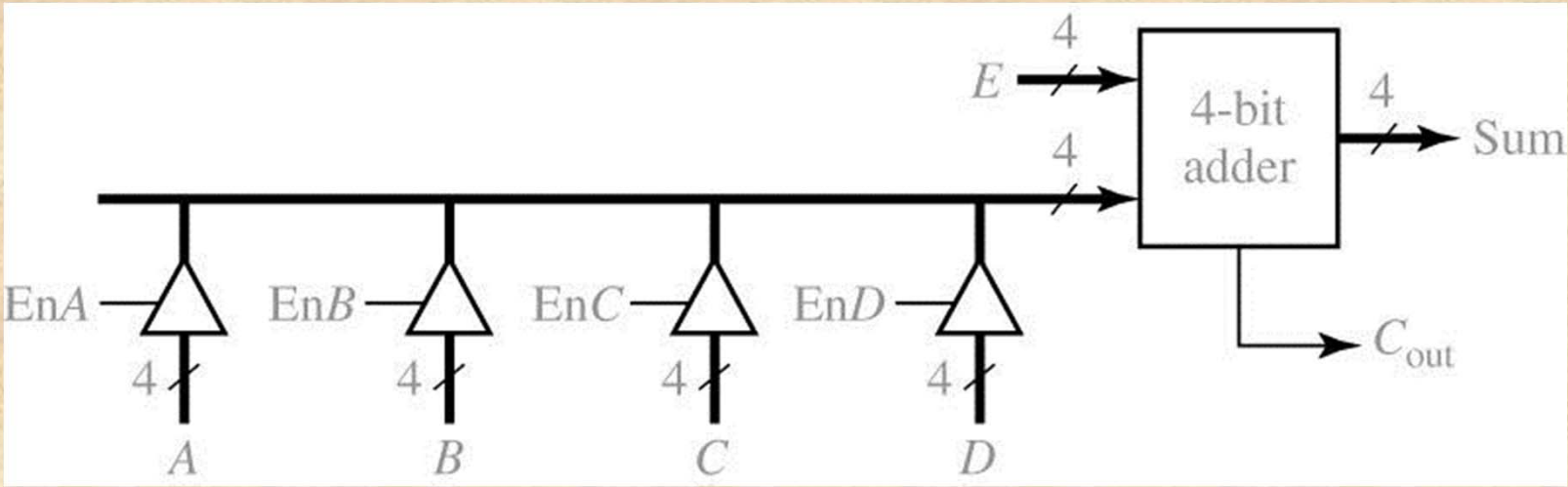
Three-State Buffers (3/5)

- How to use three-state buffers to realize a 2-to-1 multiplexer?



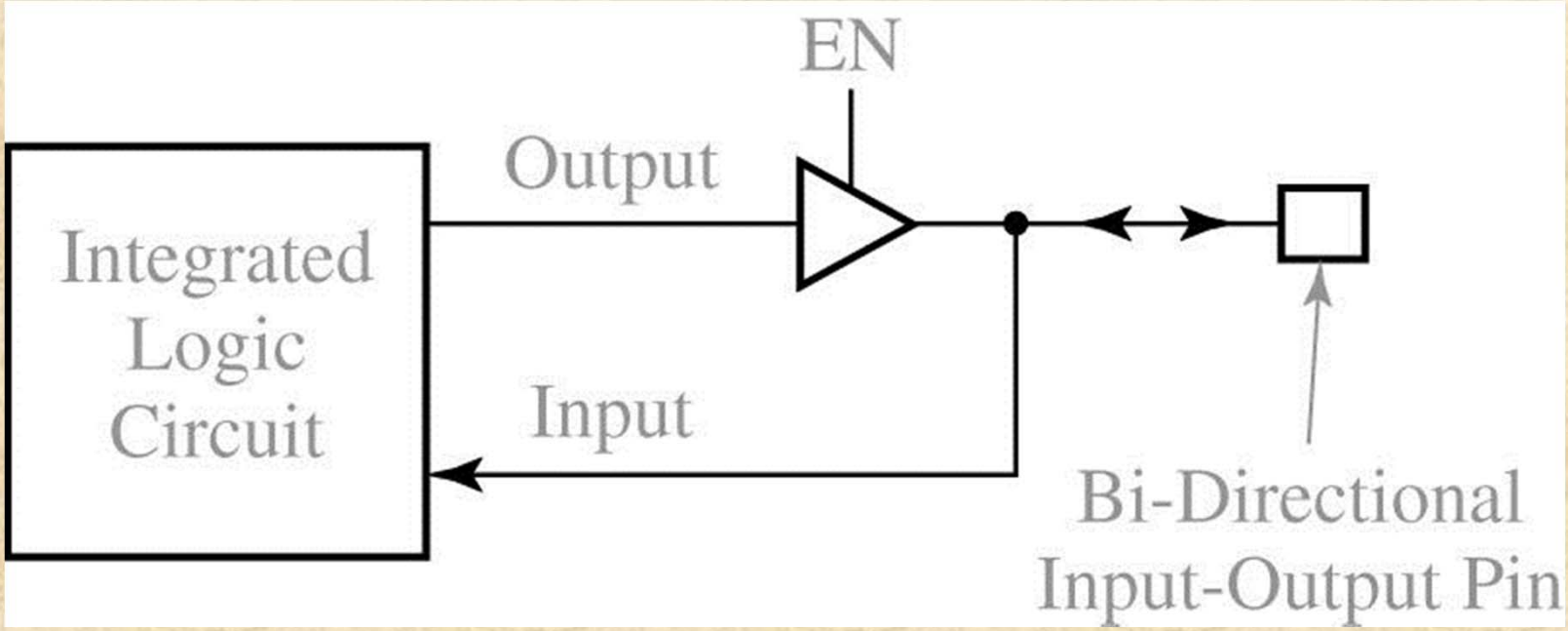
Three-State Buffers (4/5)

- 4-bit adder with four sources for one operand.



Three-State Buffers (5/5)

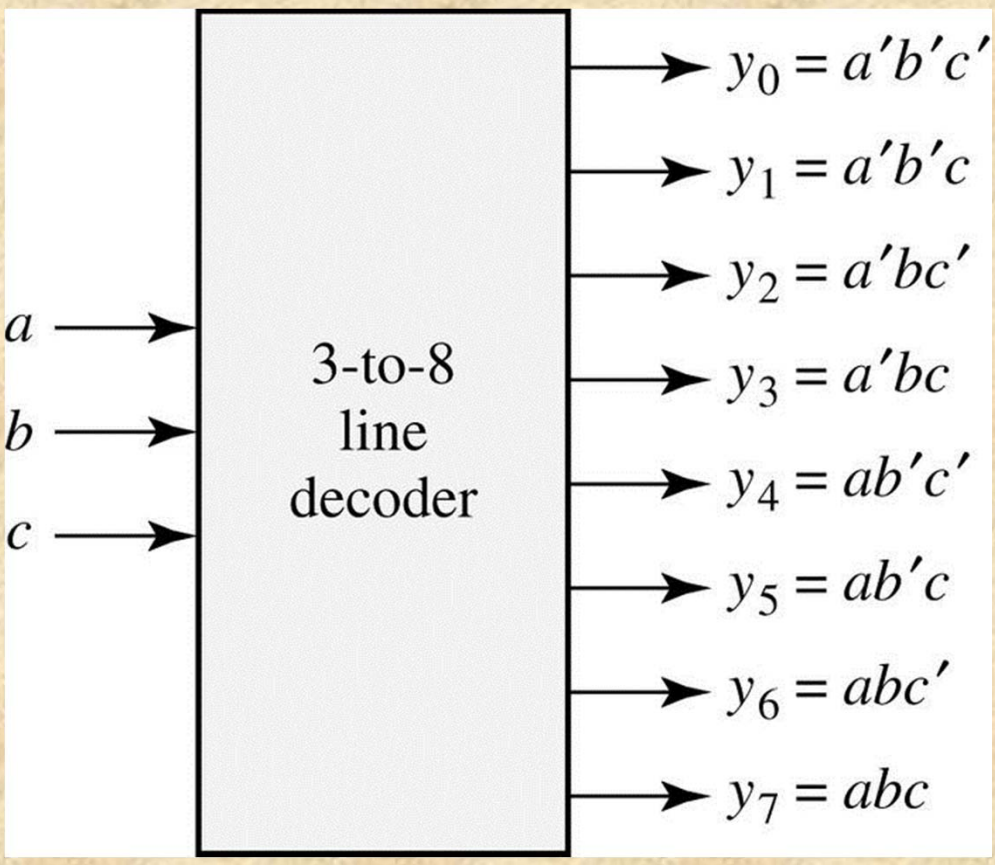
- Bi-Directional input/output pin.



EN=1: Output
EN=0: Input

Decoders and Encoders (1/3)

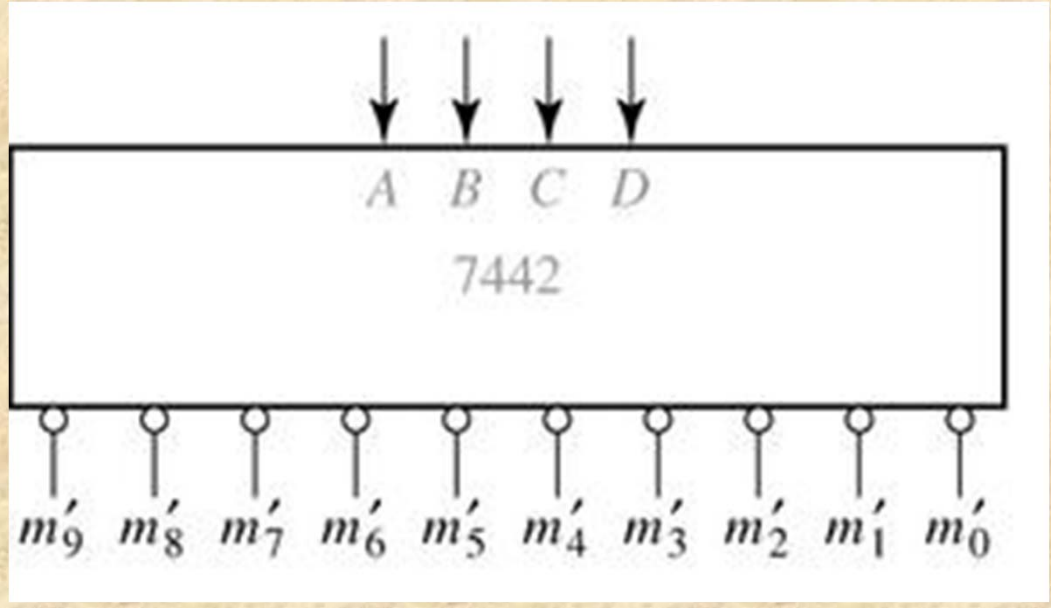
- Example: a 3-to-8 decoder.



Only the output whose subscript is equal to the binary number: abc is 1.

Decoders and Encoders (2/3)

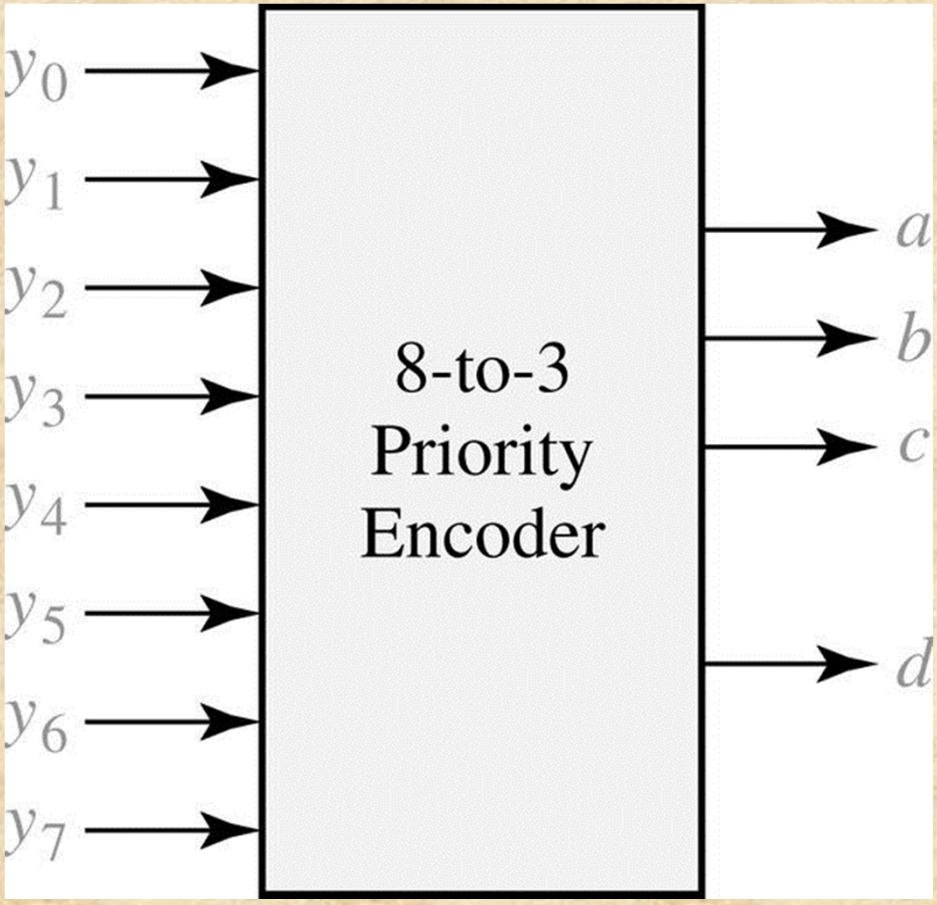
- Example 2: a 4-to-10 decoder



Only the output, whose subscript is equal to the binary number: ABCD, is 0.

Decoders and Encoders (3/3)

- Example 3: 8-to-3 priority encoder



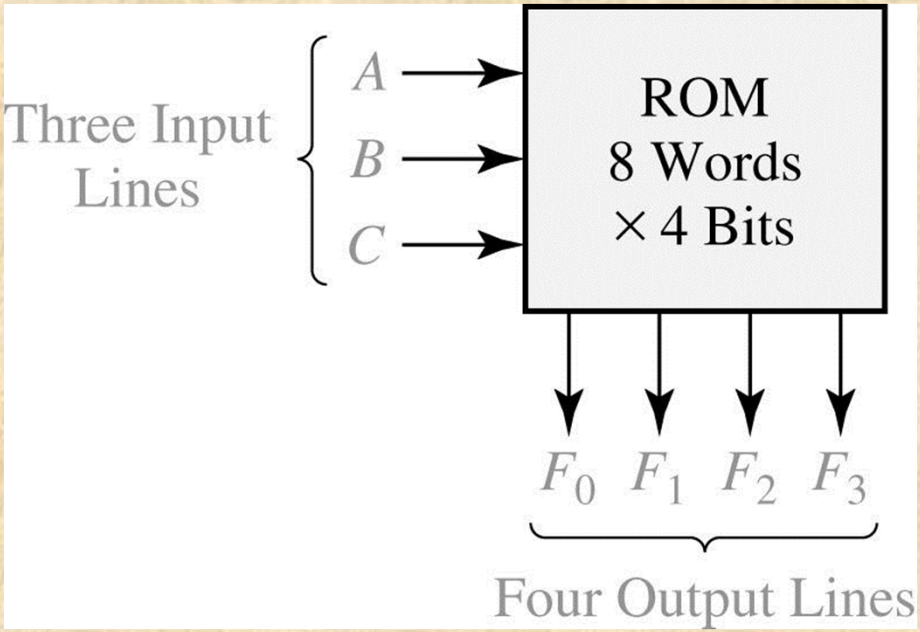
If none of y0-y7 is 1, d is 0; otherwise, the binary number abc indicates the highest numbered input whose value is 1.

Read-Only Memories (1/6)

- ROM: read-only memory
  - Interconnected gates to store an array of binary data
  - Inputs serve as address
  - Stored data cannot be changed under normal operating conditions

Read-Only Memories (2/6)

- Example 1:



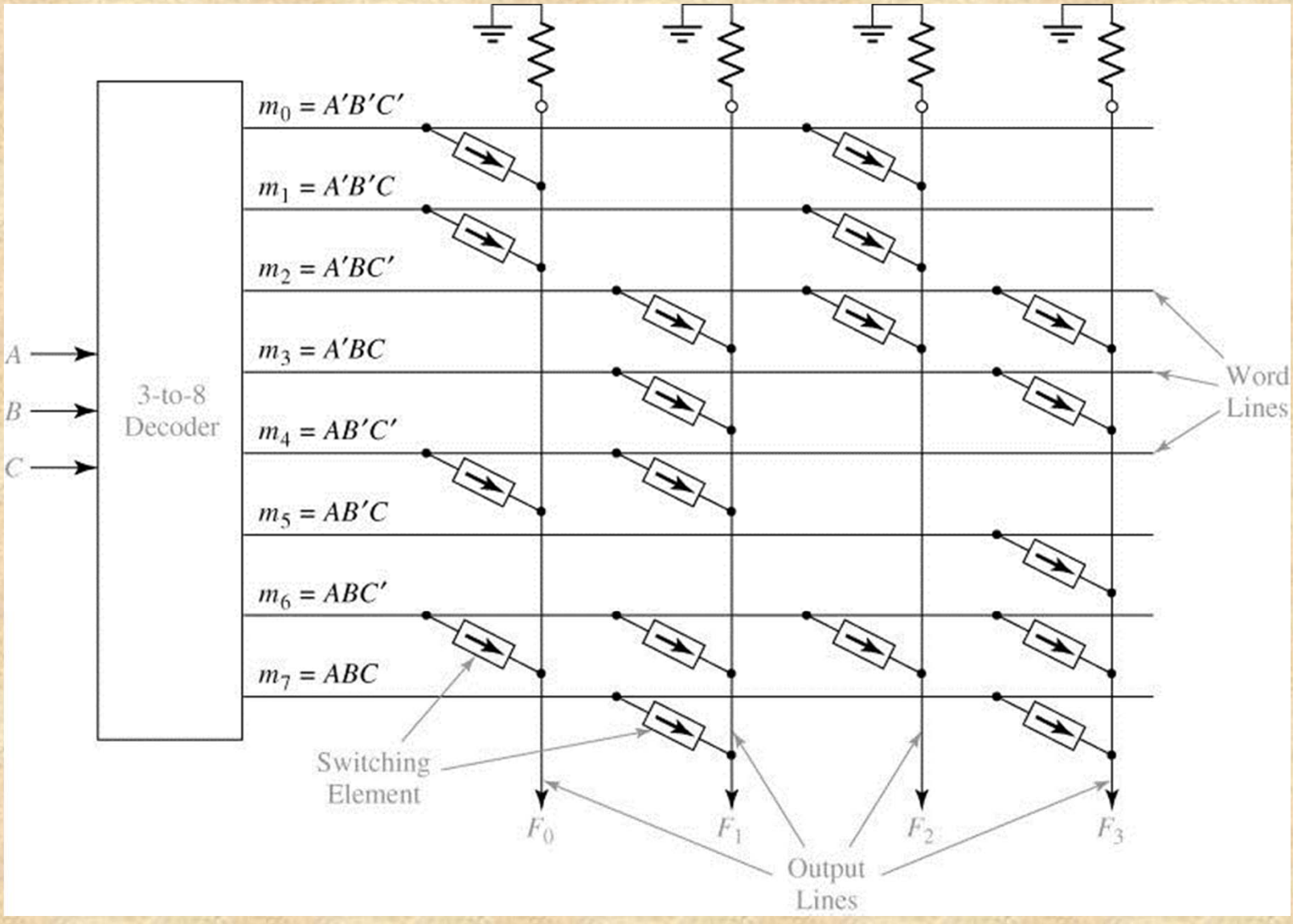
A	B	C	F_0	F_1	F_2	F_3
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1

Read-Only Memories (3/6)

- F0=sum[m(0,1,4,6)]=A’B’+AC’
- F1=sum[m(2,3,4,6,7)]=B+AC’
- F2=sum[m(0,1,2,6)]=A’B’+BC’
- F3=sum[m(2,3,5,6,7)]=AC+B
- Then, we can realize these expressions using an AND-OR network.

Read-Only Memories (4/6)

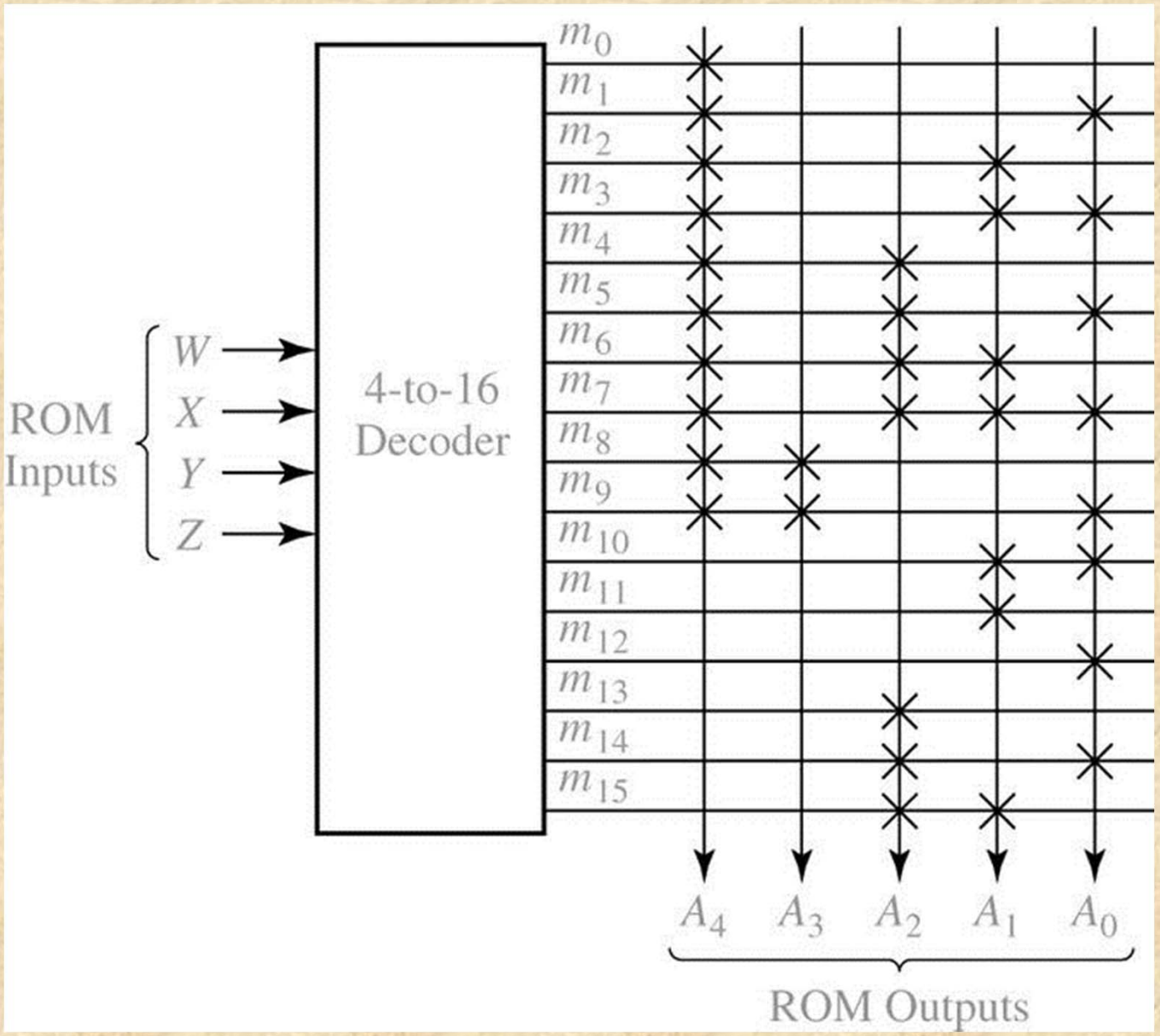
- Another way to realize is to use a 3-to-8 decoder



Read-Only Memories (5/6)

- Example 2:
  - Convert hexadecimal digits into 7-bit ASCII codes.
    1. Decide the inputs and outputs
    2. Develop a truth table
    3. Realize the circuit
       1. Minterm expansion → simplification → circuit
       2. Minterm expansion → decoder → circuit

Read-Only Memories (6/6)



An X indicates that the switching element is present and connected, and no X indicates that the corresponding element is absent or not connected.

Programmable Logic Devices (1/12)

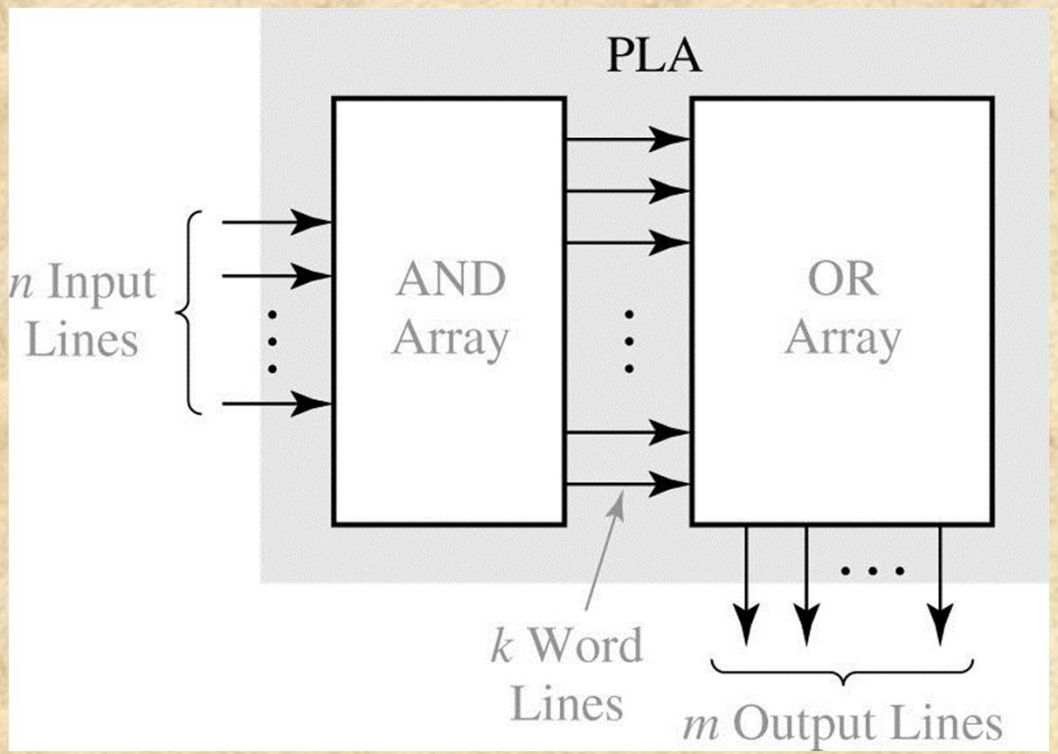
- ROMs use decoders and switching elements to OR minterms.
- But it is only a good choice when the number of inputs is small. Why?
- If we have many inputs, the required decoder will have to be very large.

Programmable Logic Devices (2/12)

- Programmable Logic Devices (PLD) can be used when the number of inputs is large.
  - Programmable Logic Arrays (PLA)
  - Programmable Array Logics (PAL)

Programmable Logic Devices (3/12)

- Programmable logic arrays (PLA)

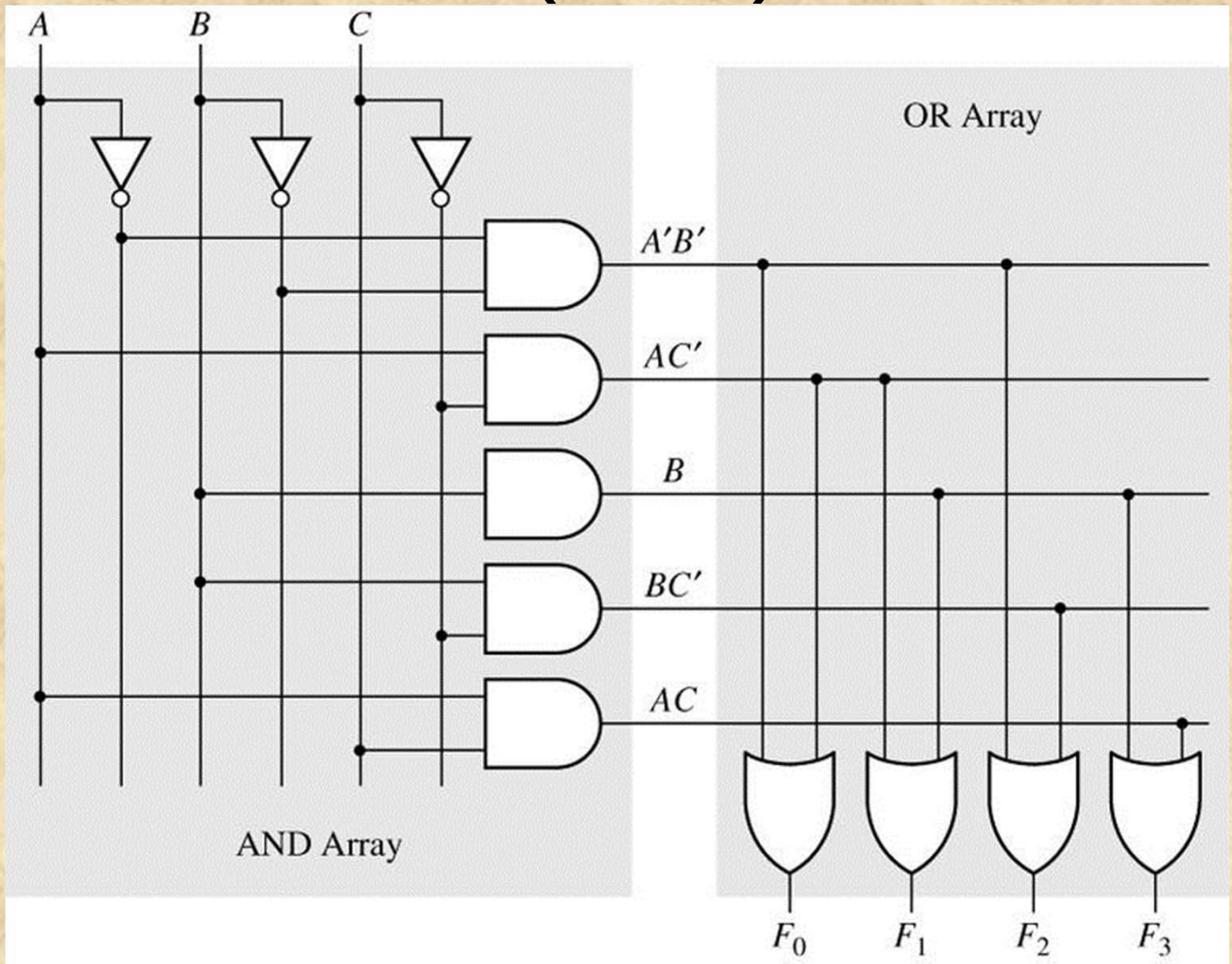


k does not have to be 2^n

Programmable Logic Devices (3/12)

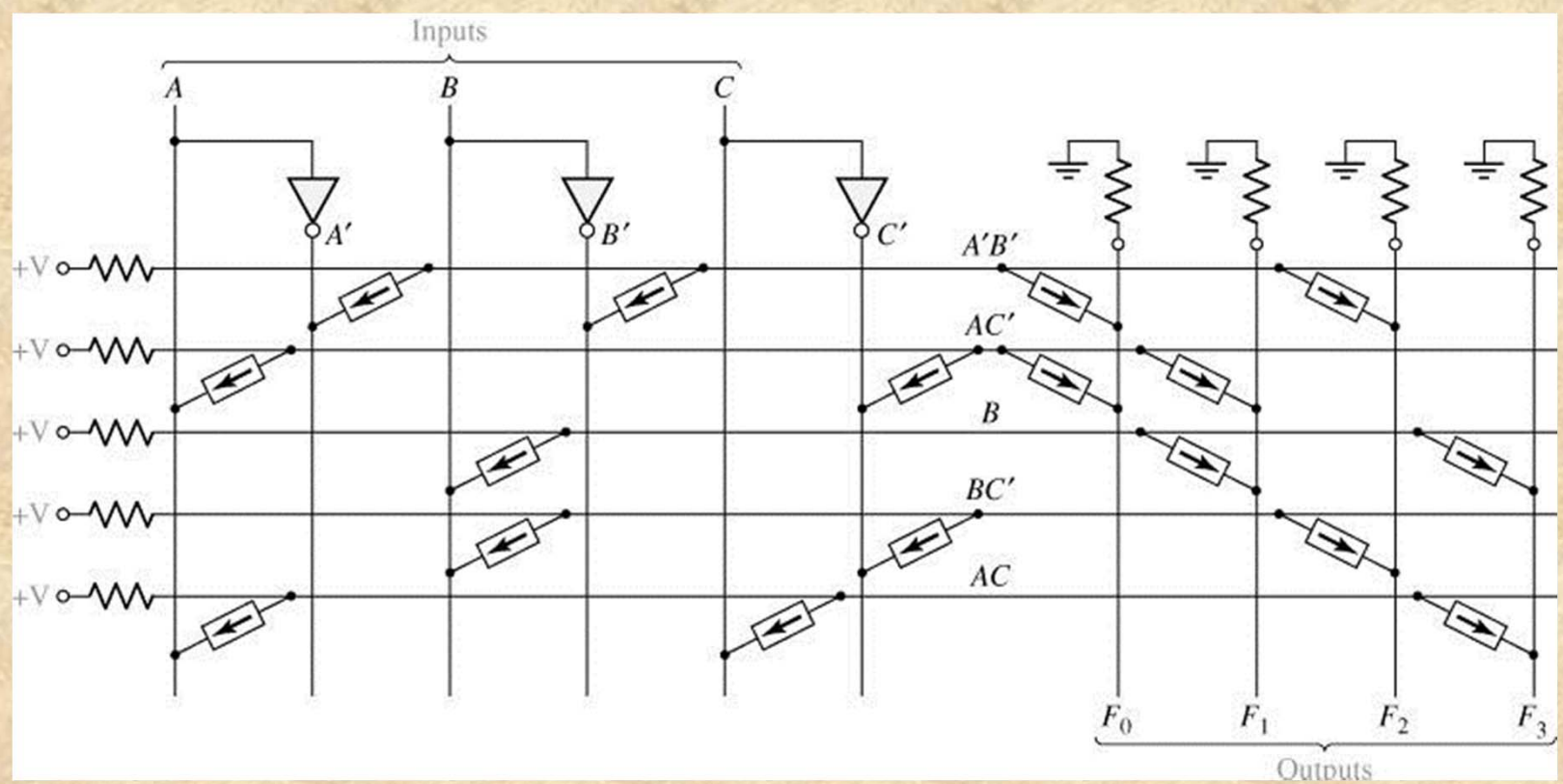
- Example 1:
  - F_0=A'B'+AC'
  - F_1=AC'+B
  - F_2=A'B'+BC'
  - F_3=B+AC

Programmable Logic Devices (4/12)



Programmable Logic Devices (5/12)

The implementation using PLA



Programmable Logic Devices (6/12)

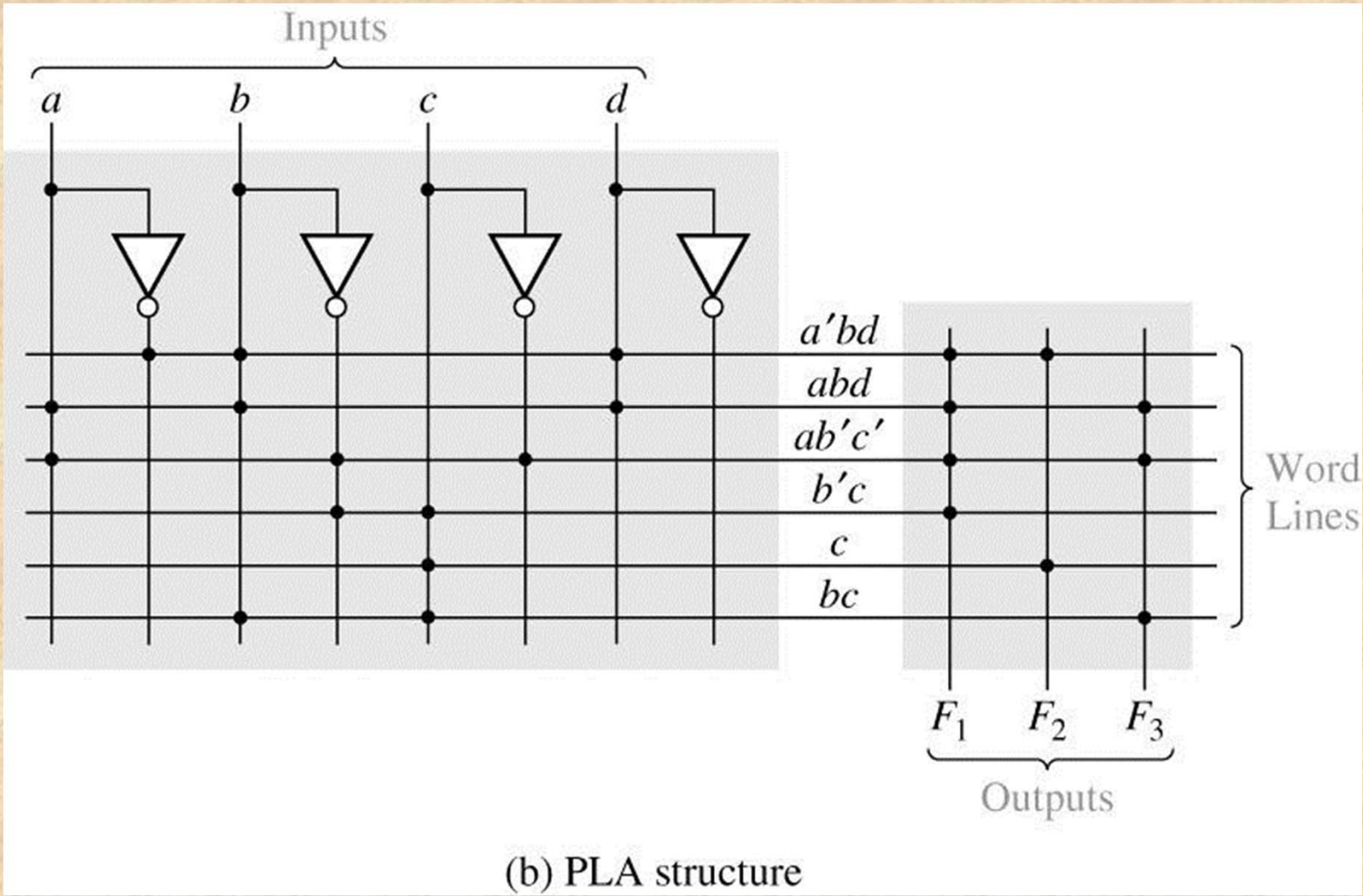
- Example 2:
  - f_1=a'bd+abd+ab'c'+b'c
  - f_2=c+a'bd
  - f_3=bc+ab'c'+abd

Programmable Logic Devices (7/12)

- Truth table used by PLA
- Note: the number of rows is not 2^n, where n is the number of inputs.

a b c d	f_1 f_2 f_3
0 1 - 1	1 1 0
1 1 - 1	1 0 1
1 0 0 -	1 0 1
- 0 1 -	1 0 0
- - 1 -	0 1 0
- 1 1 -	0 0 1

Programmable Logic Devices (7/12) (cont)



(b) PLA structure

Programmable Logic Devices (8/12)

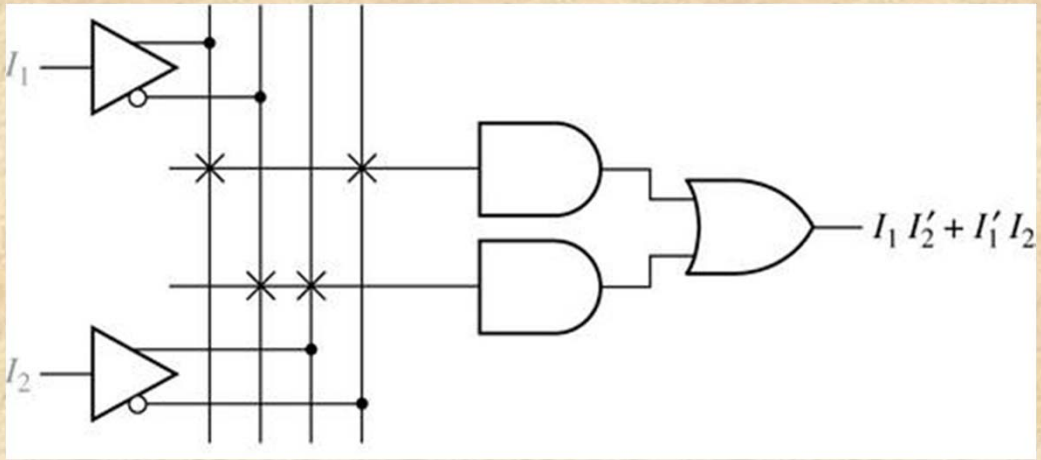
- Programmable Logic Devices (PLD) can be used when the number of inputs is large.
  - Programmable Logic Arrays (PLA)
  - Programmable Array Logics (PAL)

Programmable Logic Devices (9/12)

- PALs are similar to PLAs, but the OR array is fixed and the AND array is programmable.

Programmable Logic Devices (10/12)

- Example 3:
  - $F=I_1I_2'+I_1'I_2$



Programmable Logic Devices (11/12)

- Example 4:
  - Sum=X'Y'C_{in}+X'YC_{in}'+XY'C_{in}'+XYC_{in}
  - C_{out}=XC_{in}+YC_{in}+XY
  - If the PAL has two OR gates both of which have four inputs, one of the inputs for C_{out} has the input of 0, why?

Programmable Logic Devices (12/12)