Digital Systems Design and Laboratory

[9. Multiplexers, Decoders, and Programmable Logic Devices]

Chung-Wei Lin

cwlin@csie.ntu.edu.tw

CSIE Department

National Taiwan University

Spring 2019

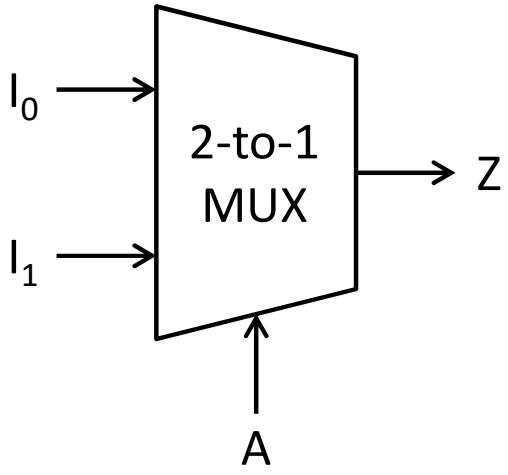
We would like to thank Prof. Hui-Ru Jiang (NTU) for contributing the lecture material

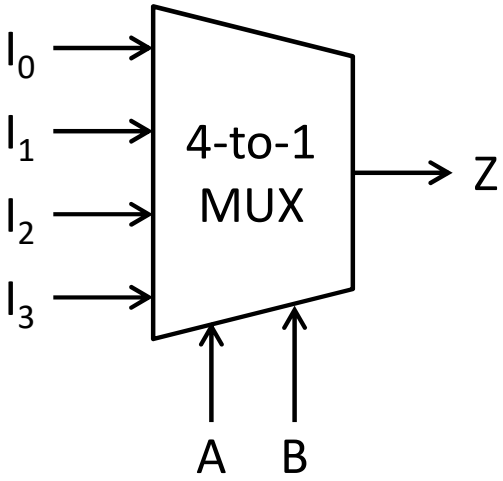
Outline

- **Multiplexers**
- Three-State Buffers
- Decoders and Encoders
- Read-Only Memories
- Programmable Logic Devices
- Complex Programmable Logic Devices
- Field-Programmable Gate Arrays

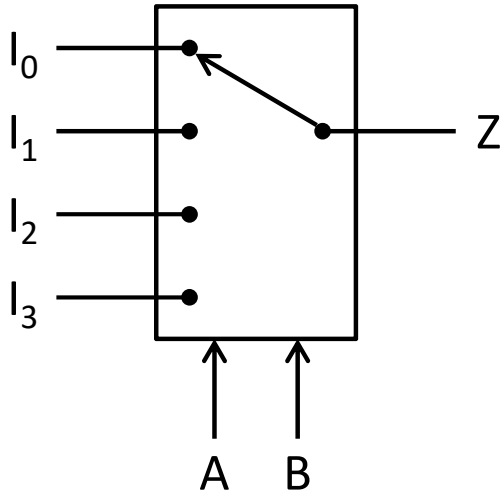
Multiplexers (1/3)

- A multiplexer (data selector, MUX)
  - Use the control inputs (A and/or B) to select one of the data inputs (I_X)
    - One combination of control inputs corresponds to one data input
  - Connects it to the output
- Symbol, truth table, and switch



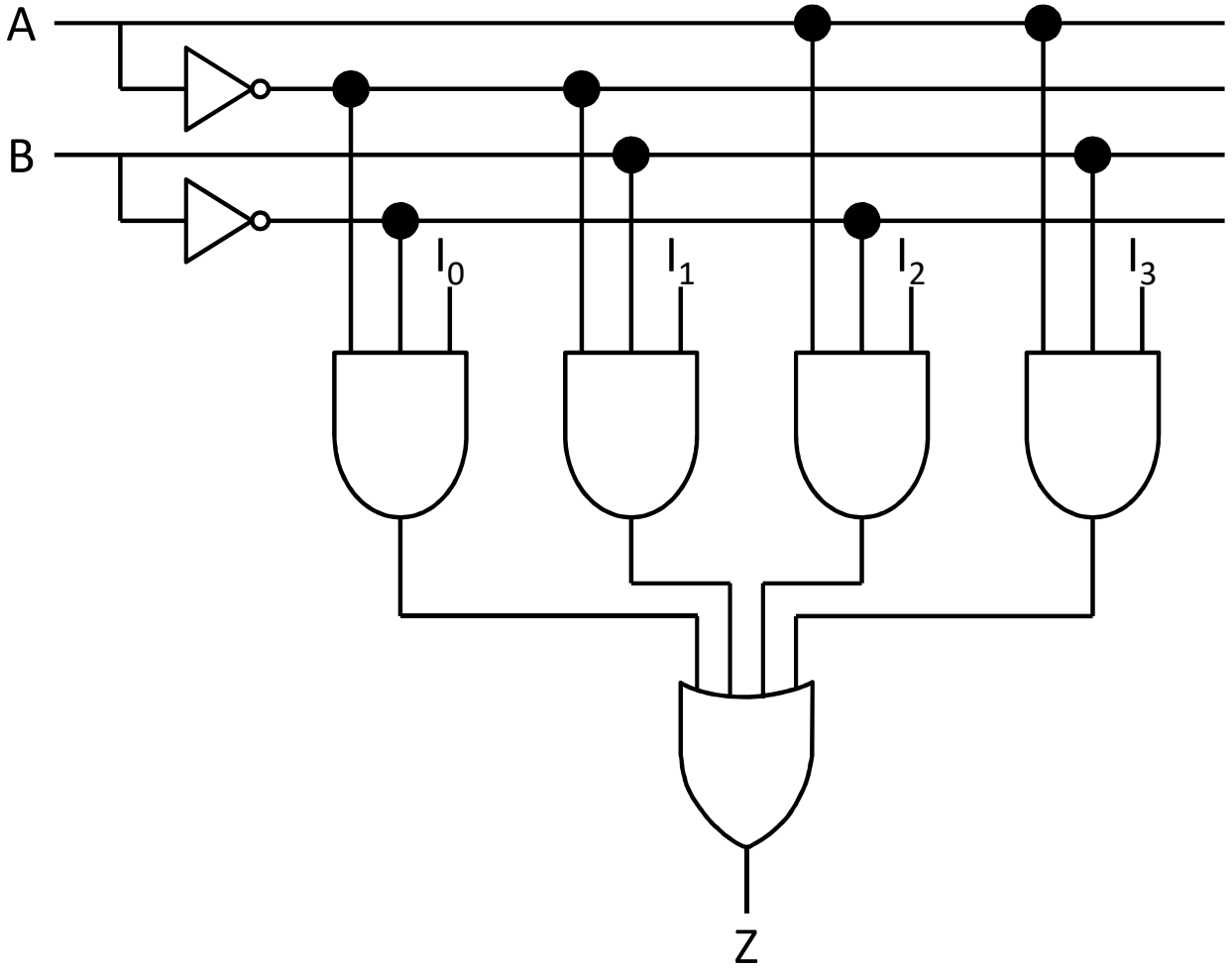


A	B	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Multiplexers (2/3)

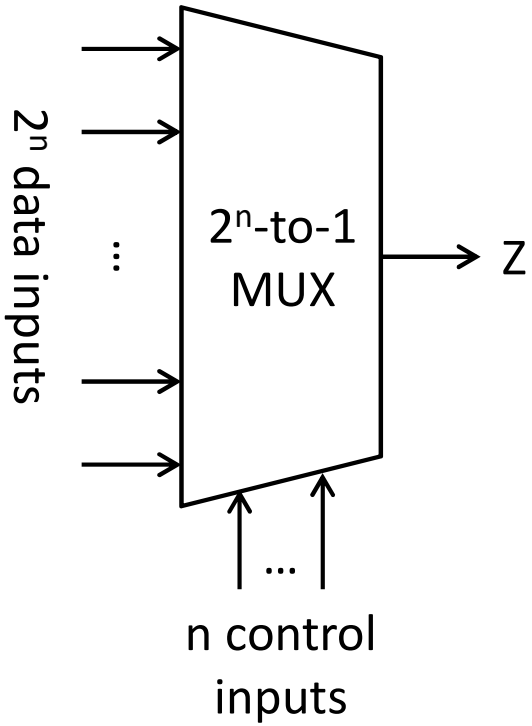
- Logic equation of 4-to-1 MUX
  - $Z = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3 = m_0I_0 + m_1I_1 + m_2I_2 + m_3I_3$



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

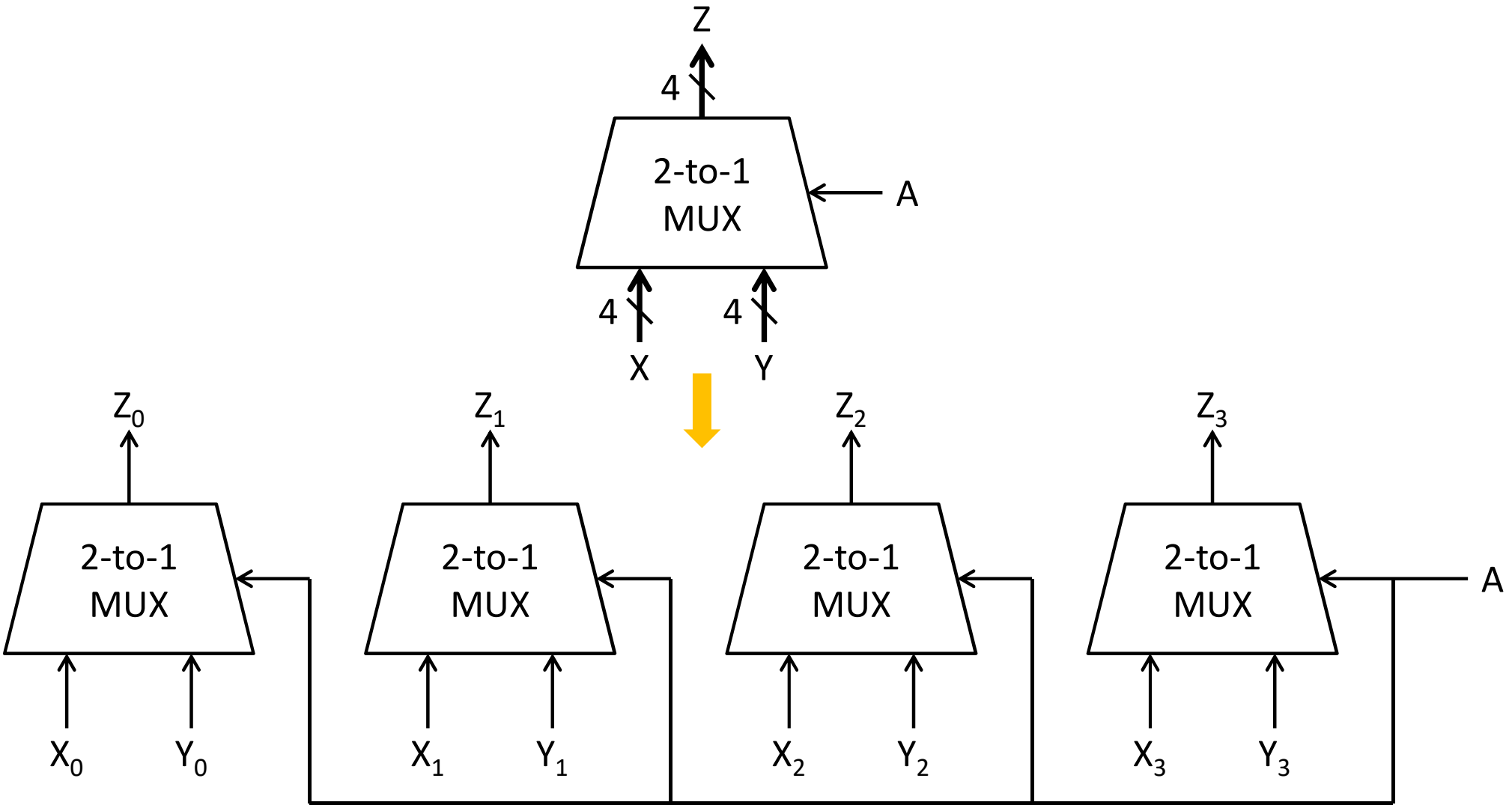
Multiplexers (3/3)

- The general equation for n control inputs
  - $Z = m_0 I_0 + m_1 I_1 + \ldots + m_i I_i + \ldots + m_{2^n-1} I_{2^n-1} = \sum_{k=0\ldots2^n-1} m_k I_k$



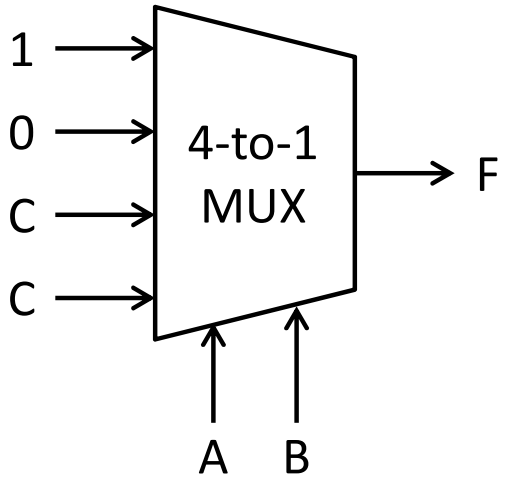
Application: Quad Multiplexer

- Select one of two 4-bit data words



Application: Combinational Logic

- Realize a 3-variable function by a 4-to-1 MUX
  - F(A,B,C) = A'B' + AC = 1 • A'B' + 0 • A'B + C • AB' + C • AB



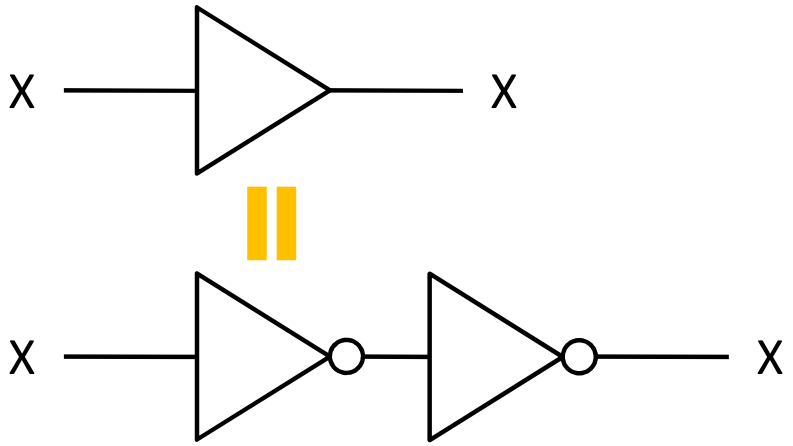
A	B	F
0	0	1
0	1	0
1	0	C
1	1	C

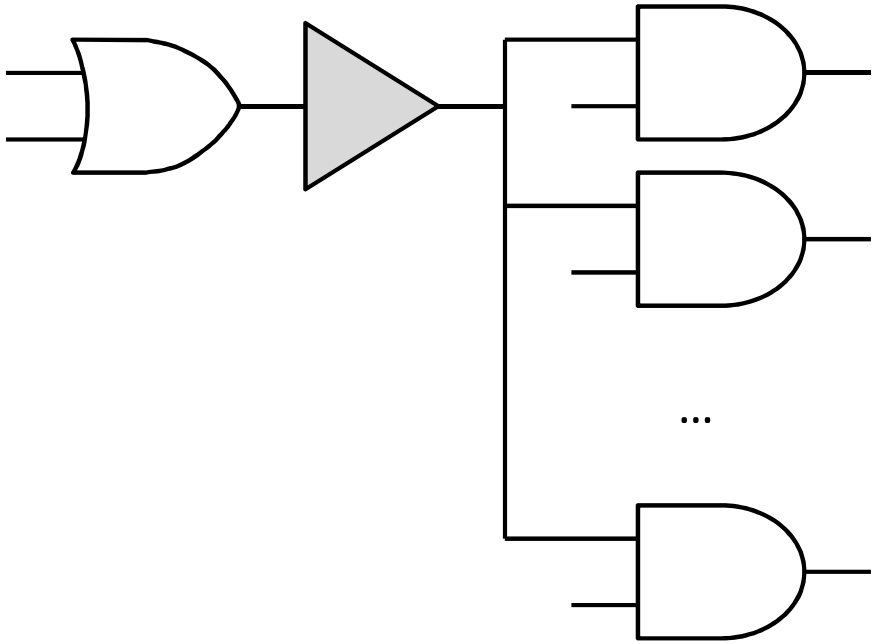
Outline

- ❑ Multiplexers
- ❑ **Three-State Buffers**
- ❑ Decoders and Encoders
- ❑ Read-Only Memories
- ❑ Programmable Logic Devices
- ❑ Complex Programmable Logic Devices
- ❑ Field-Programmable Gate Arrays

Buffers

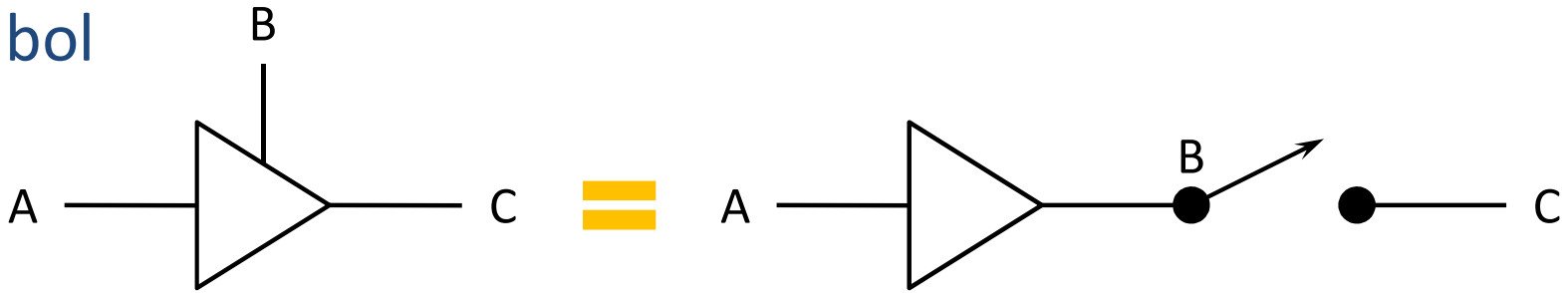
- Buffers: increase the **driving capability** of a gate output
- Symbol



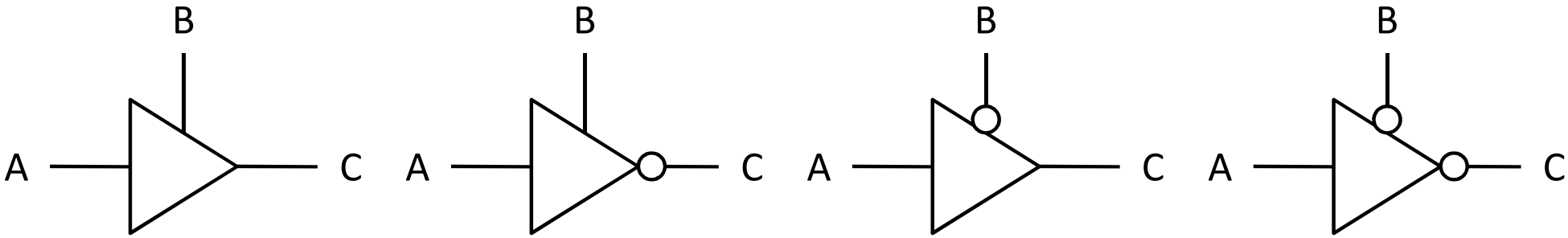


Three-State Buffers

- Three-state buffers: permits gate outputs to be connected together
- Symbol



- 4 variants (Z = high-impedance)



B	C
0	Z
1	A

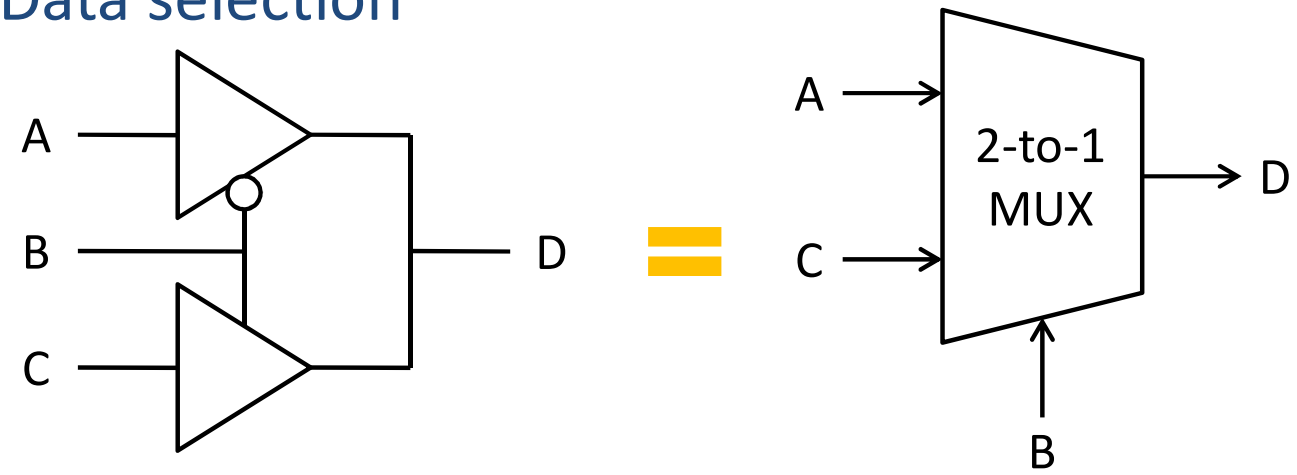
B	C
0	Z
1	A'

B	C
0	A
1	Z

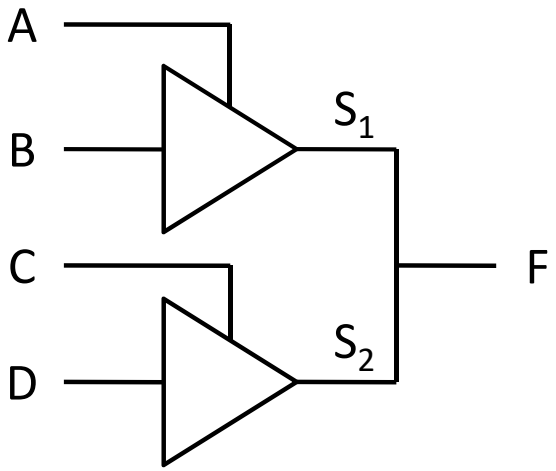
B	C
0	A'
1	Z

Applications (1/2)

- Data selection



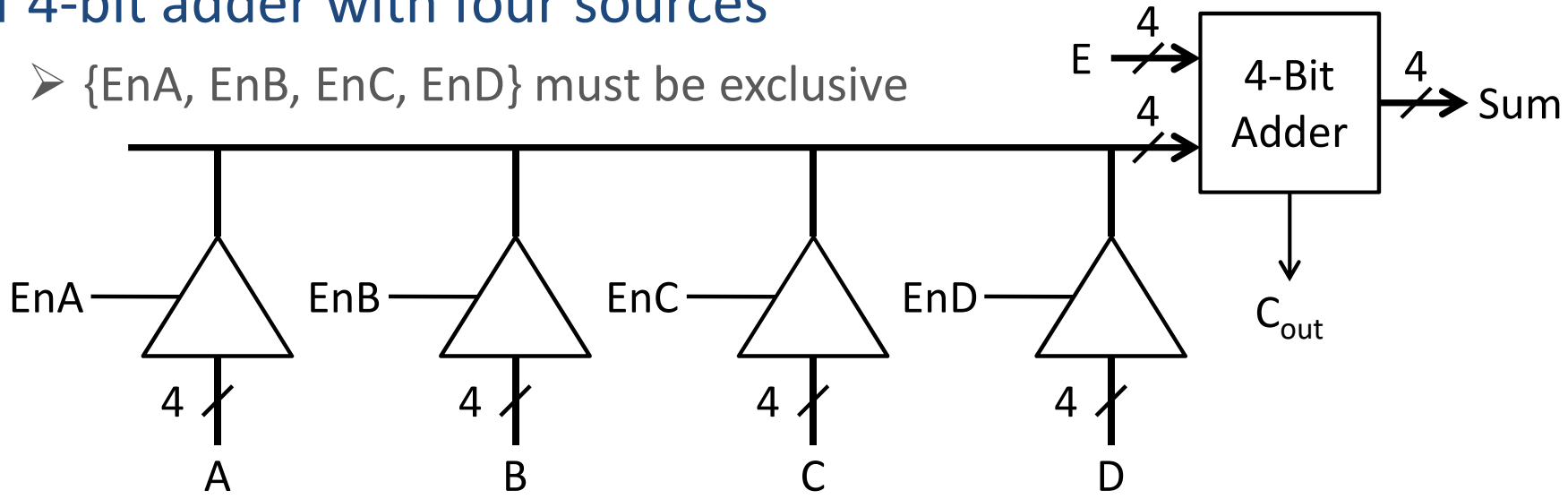
- Circuit with 2 three-state buffers (X = unknown)



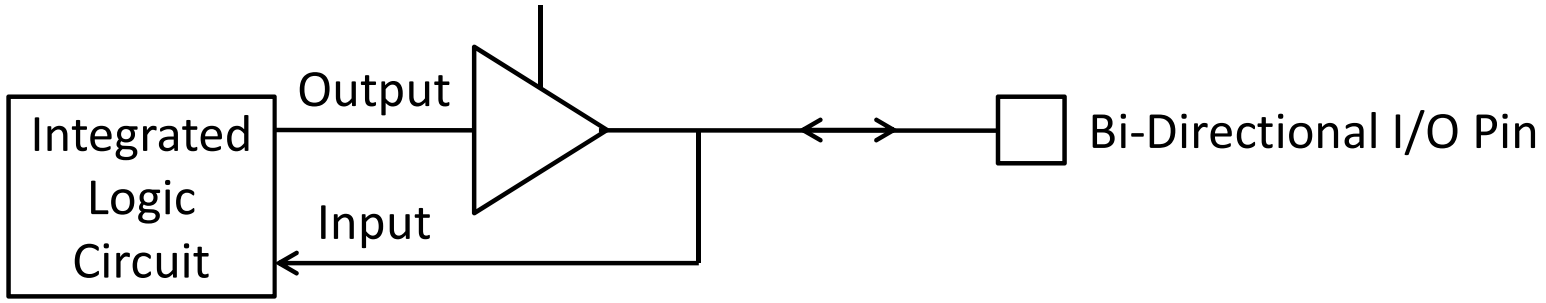
F	X	0	1	Z
X	X	X	X	X
0	X	0	X	0
1	X	X	1	1
Z	X	0	1	Z

Applications (2/2)

- 4-bit adder with four sources
  - {EnA, EnB, EnC, EnD} must be exclusive



- Bi-directional I/O pin
  - The same pin can be used as an input pin and as an output pin, but not both at the same time

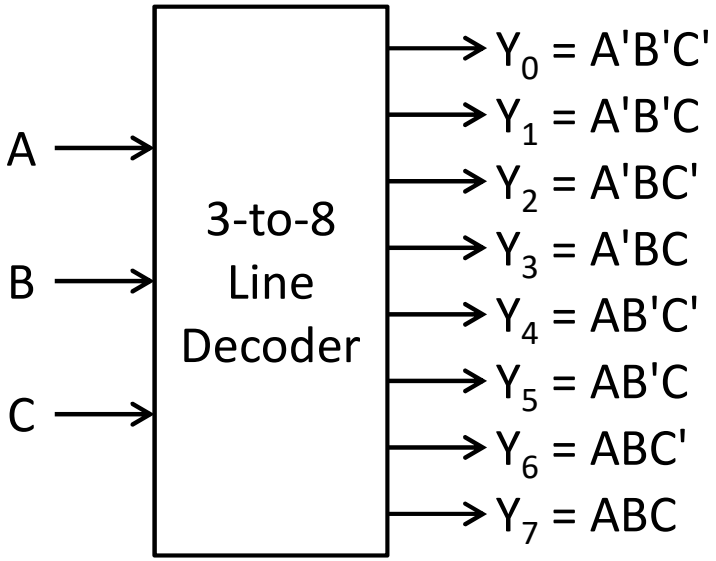


Outline

- ❑ Multiplexers
- ❑ Three-State Buffers
- ❑ **Decoders and Encoders**
- ❑ Read-Only Memories
- ❑ Programmable Logic Devices
- ❑ Complex Programmable Logic Devices
- ❑ Field-Programmable Gate Arrays

Decoders

- Decoder: generates all of the minterms of input variables
  - **Exactly one output line corresponds to one input combination**

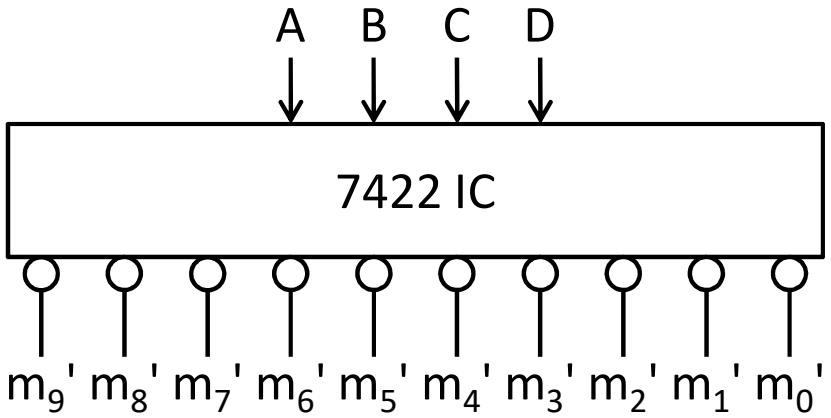


	A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
m_0	0	0	0	1	0	0	0	0	0	0	0
m_1	0	0	1	0	1	0	0	0	0	0	0
m_2	0	1	0	0	0	1	0	0	0	0	0
m_3	0	1	1	0	0	0	1	0	0	0	0
m_4	1	0	0	0	0	0	0	1	0	0	0
m_5	1	0	1	0	0	0	0	0	1	0	0
m_6	1	1	0	0	0	0	0	0	0	1	0
m_7	1	1	1	0	0	0	0	0	0	0	1

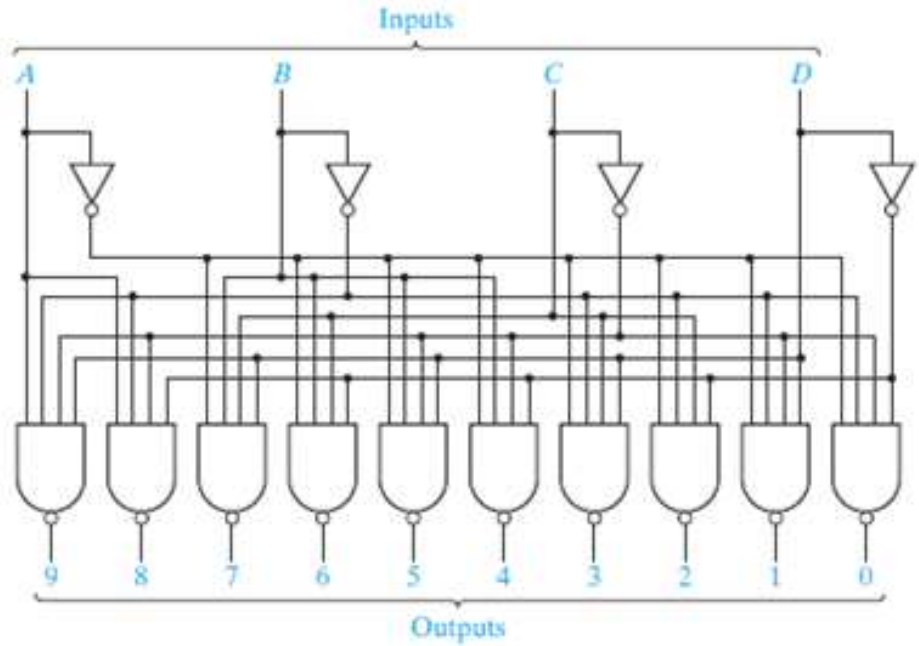
- n-to-2^n decoder
  - Non-inverted outputs
    - $y_i = m_i$, i = 0 to 2^n-1
  - Inverted outputs
    - $y_i = m_i' = M_i$, i = 0 to 2^n-1

Application: 4-to-10 Line Decoder

- Block diagram



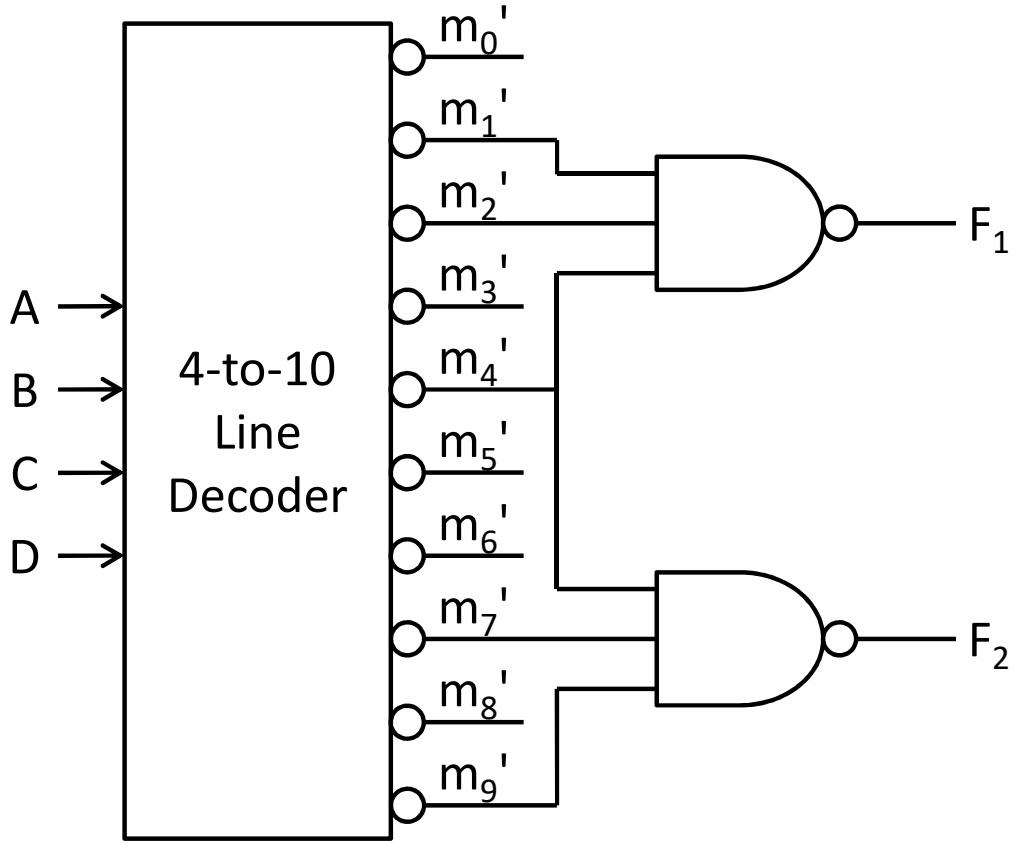
- Logic diagram



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

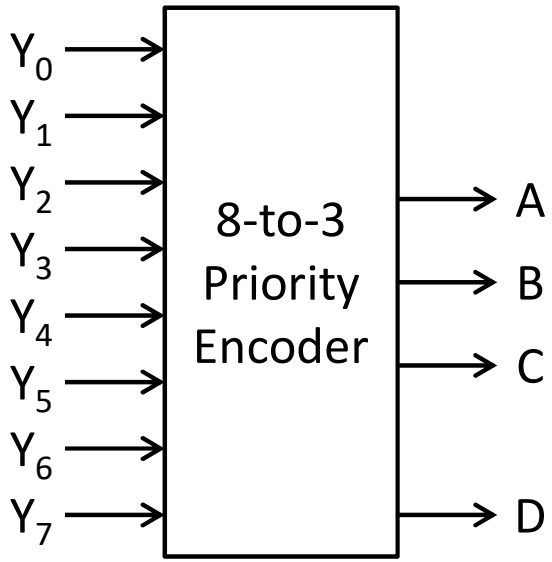
Application: n-Variable Function

- Exactly one output line corresponds to one minterm
  - Realize n-variable functions by ORing selected minterm outputs from a decoder
  - Examples: $F_1 = \sum m(1,2,4)$, $F_2 = \sum m(4,7,9)$



Encoders

- Encoder: performs the inverse function of a decoder
  - If $Y_i = 1$, ABC outputs represent a binary number equal to i
  - If more than one input can be 1 at a time, use a priority scheme



X: Don't Care

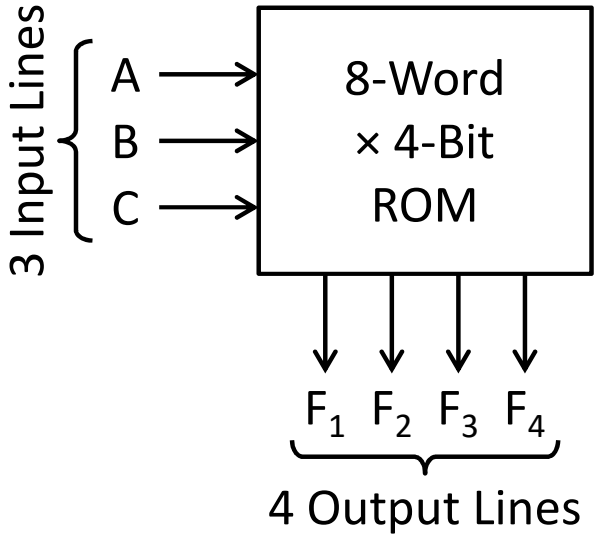
	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	A	B	C	D
	0	0	0	0	0	0	0	0	0	0	0	0
m_0	1	0	0	0	0	0	0	0	0	0	0	1
m_1	X	1	0	0	0	0	0	0	0	0	1	1
m_2	X	X	1	0	0	0	0	0	0	1	0	1
m_3	X	X	X	1	0	0	0	0	0	1	1	1
m_4	X	X	X	X	1	0	0	0	1	0	0	1
m_5	X	X	X	X	X	1	0	0	1	0	1	1
m_6	X	X	X	X	X	X	1	0	1	1	0	1
m_7	X	X	X	X	X	X	X	1	1	1	1	1

Outline

- ❑ Multiplexers
- ❑ Three-State Buffers
- ❑ Decoders and Encoders
- ❑ **Read-Only Memories**
- ❑ Programmable Logic Devices
- ❑ Complex Programmable Logic Devices
- ❑ Field-Programmable Gate Arrays

Read-Only Memories (1/3)

- Read-Only Memory (ROM): stores an array of binary data
  - Stored data cannot be changed
  - e.g., 8-word × 4-bit ROM: each word is 4-bit, total 8 words
    - Input (ABC): 3-bit lines index 2^3 values (0--7 addresses)
    - Output ($F_0F_1F_2F_3$): each one is called a word

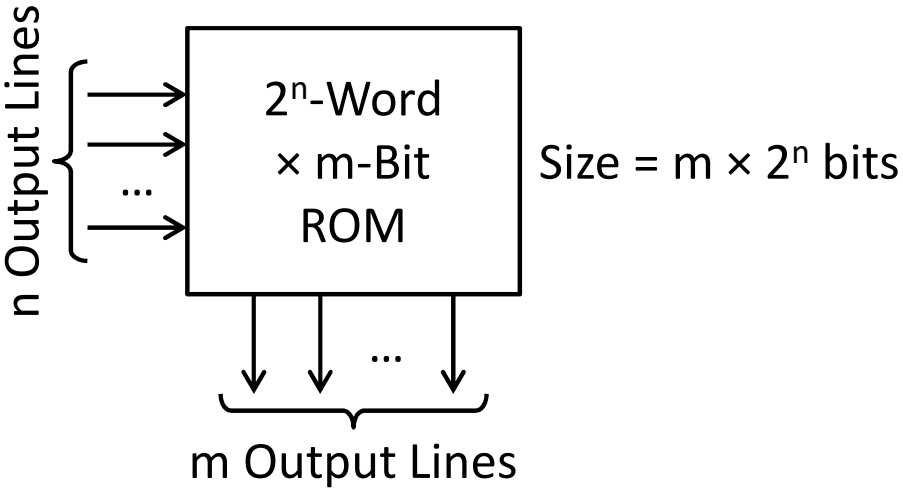


Size = 4 × 8 bits

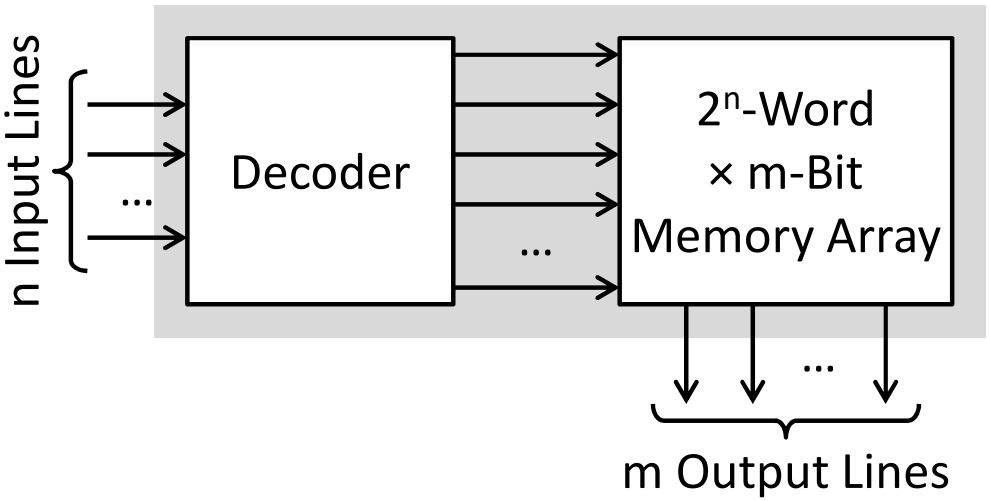
A	B	C	F_0	F_1	F_2	F_3
0	0	0	**1**	**0**	**1**	**0**
0	0	1	**1**	**0**	**1**	**0**
0	1	0	**0**	**1**	**1**	**1**
0	1	1	**0**	**1**	**0**	**1**
1	0	0	**1**	**1**	**0**	**0**
1	0	1	**0**	**0**	**0**	**1**
1	1	0	**1**	**1**	**1**	**1**
1	1	1	**0**	**1**	**0**	**1**

Read-Only Memories (2/3)

- Generalized form: 2^n-word × m-bit ROM (n inputs / m outputs)

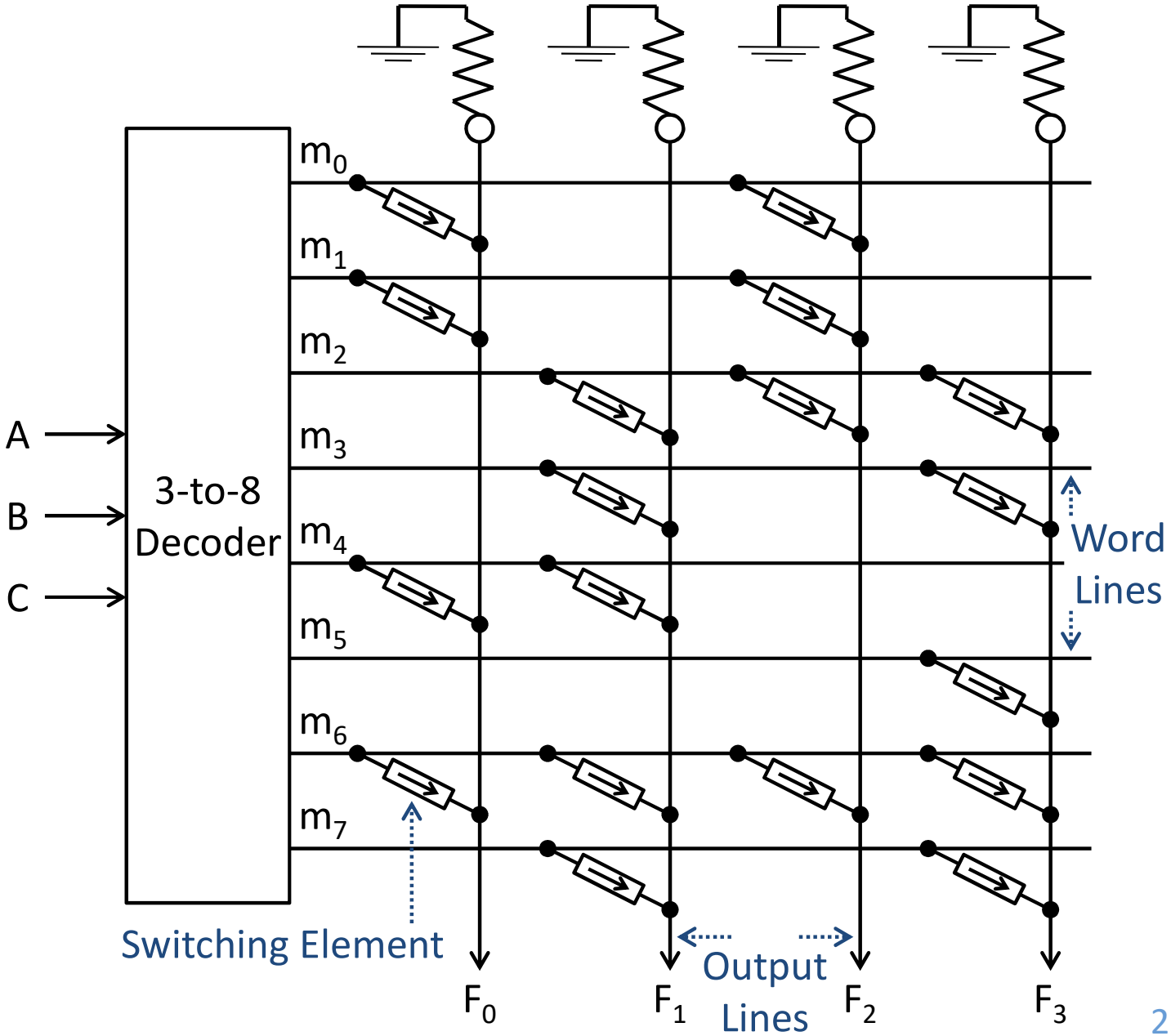


- Basic ROM structure: a decoder + memory array



Read-Only Memories (3/3)

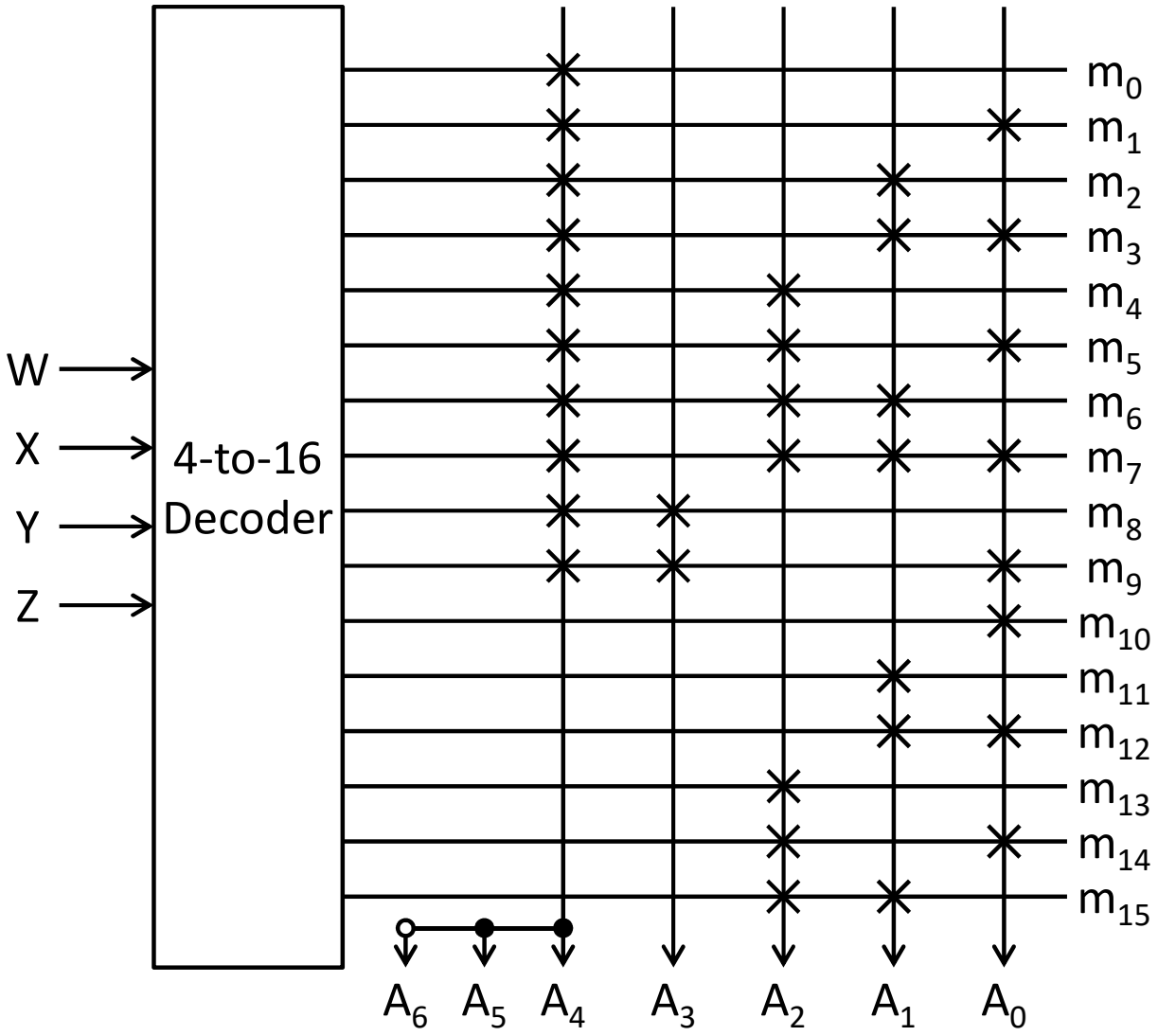
A	B	C	F_0	F_1	F_2	F_3
0	0	0	**1**	**0**	**1**	**0**
0	0	1	**1**	**0**	**1**	**0**
0	1	0	**0**	**1**	**1**	**1**
0	1	1	**0**	**1**	**0**	**1**
1	0	0	**1**	**1**	**0**	**0**
1	0	1	**0**	**0**	**0**	**1**
1	1	0	**1**	**1**	**1**	**1**
1	1	1	**0**	**1**	**0**	**1**



Application: Code Converter

- Hexadecimal-to-ASCII code converter

Hex	$A_6\ A_5\ A_4\ A_3\ A_2\ A_1\ A_0$
0	0 1 1 0 0 0 0
1	0 1 1 0 0 0 1
2	0 1 1 0 0 1 0
3	0 1 1 0 0 1 1
4	0 1 1 0 1 0 0
5	0 1 1 0 1 0 1
6	0 1 1 0 1 1 0
7	0 1 1 0 1 1 1
8	0 1 1 1 0 0 0
9	0 1 1 1 0 0 1
A	1 0 0 0 0 0 1
B	1 0 0 0 0 1 0
C	1 0 0 0 0 1 1
D	1 0 0 0 1 0 0
E	1 0 0 0 1 0 1
F	1 0 0 0 1 1 0



Common Types of ROMs

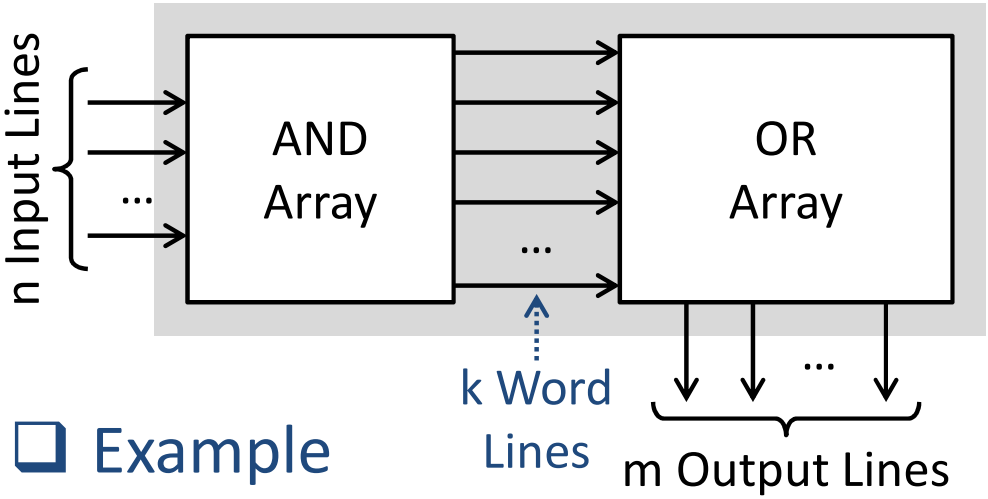
- Mask-programmable ROMs
  - Use mask to program
    - Include/omit switching elements
- Programmable ROMs (PROMs)
  - Program once
- Erasable programmable read-only memory (EPROM)
- Electrically erasable programmable ROMs (EEPROMs)
  - (Can reprogram 100 to 1000 times)
  - Flash memory → solid-state drive

Outline

- ❑ Multiplexers
- ❑ Three-State Buffers
- ❑ Decoders and Encoders
- ❑ Read-Only Memories
- ❑ **Programmable Logic Devices**
- ❑ Complex Programmable Logic Devices
- ❑ Field-Programmable Gate Arrays

Programmable Logic Arrays (1/4)

- Programmable Logic Array (PLA): 2-level SOP implementation
  - AND plane generates product terms
  - OR plane sums the product terms

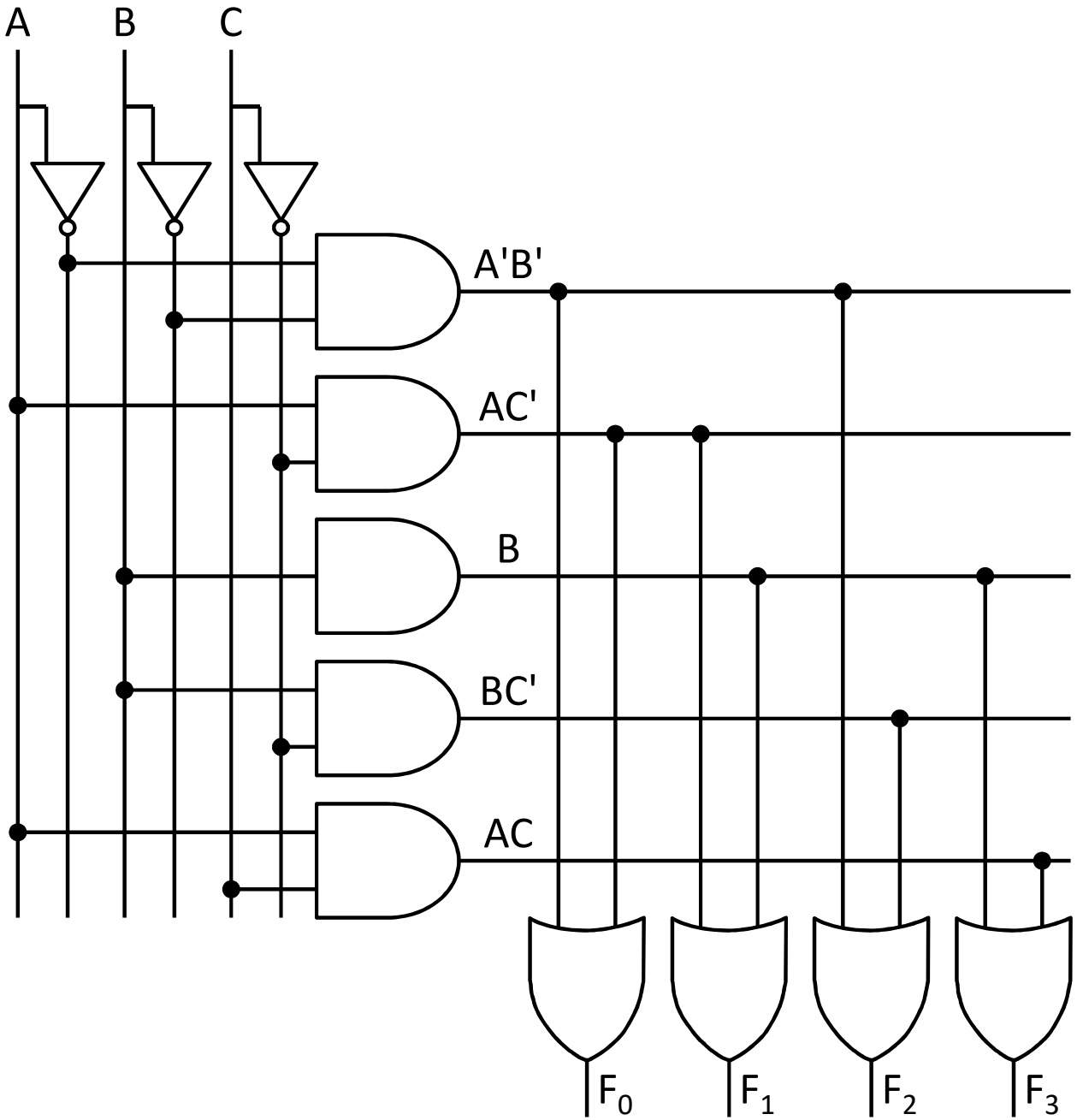


- Example
  - $F_1 = A'B' + AC'$
  - $F_2 = AC' + B$
  - $F_3 = A'B' + BC'$
  - $F_4 = B + AC$

	A	B	C	F_1	F_2	F_3	F_4
A'B'	0	0	–	1	0	1	0
AC'	1	–	0	1	1	0	0
B	–	1	–	0	1	0	1
BC'	–	1	0	0	0	1	0
AC	1	–	1	0	0	0	1

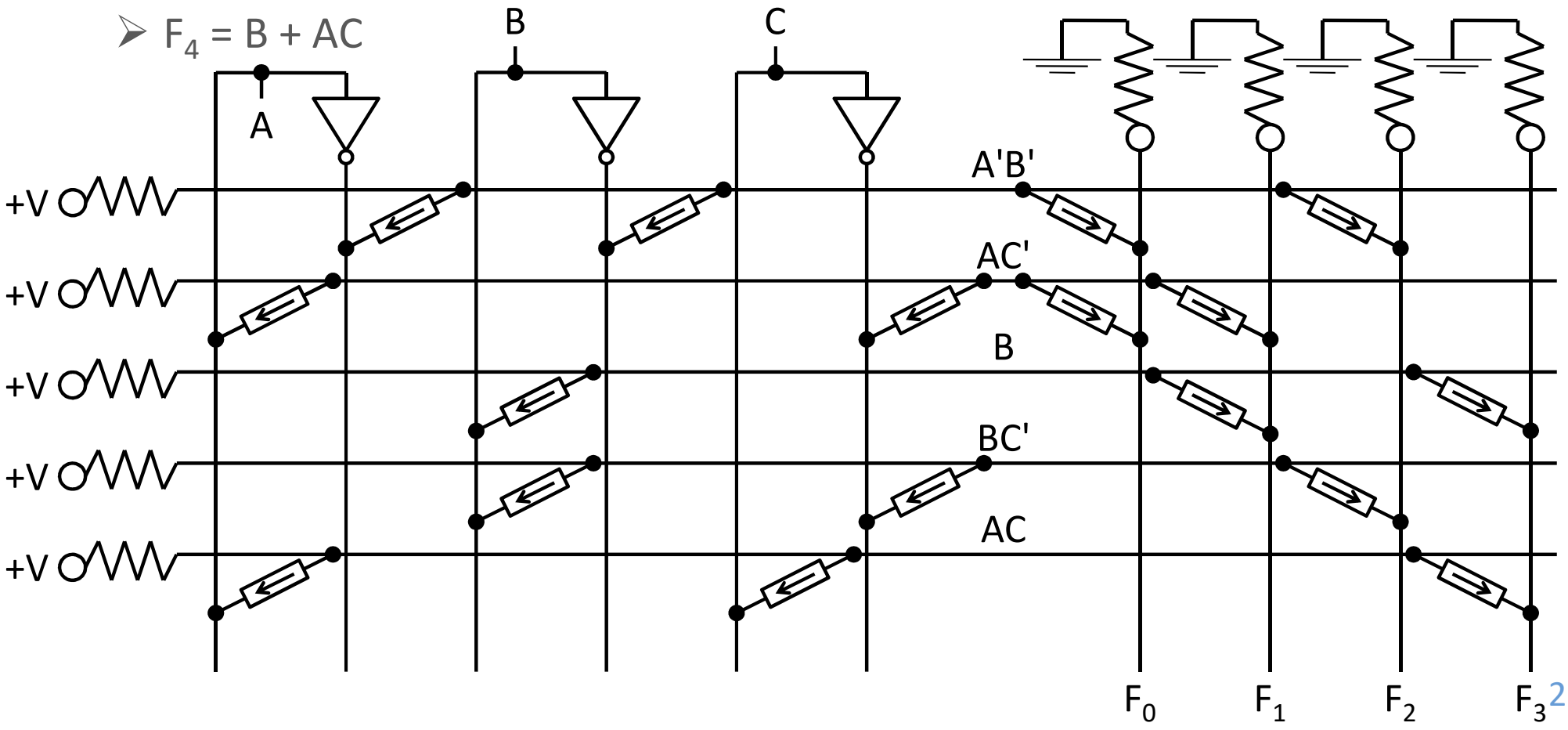
Programmable Logic Arrays (2/4)

- Example
  - $F_1 = A'B' + AC'$
  - $F_2 = AC' + B$
  - $F_3 = A'B' + BC'$
  - $F_4 = B + AC$



Programmable Logic Arrays (3/4)

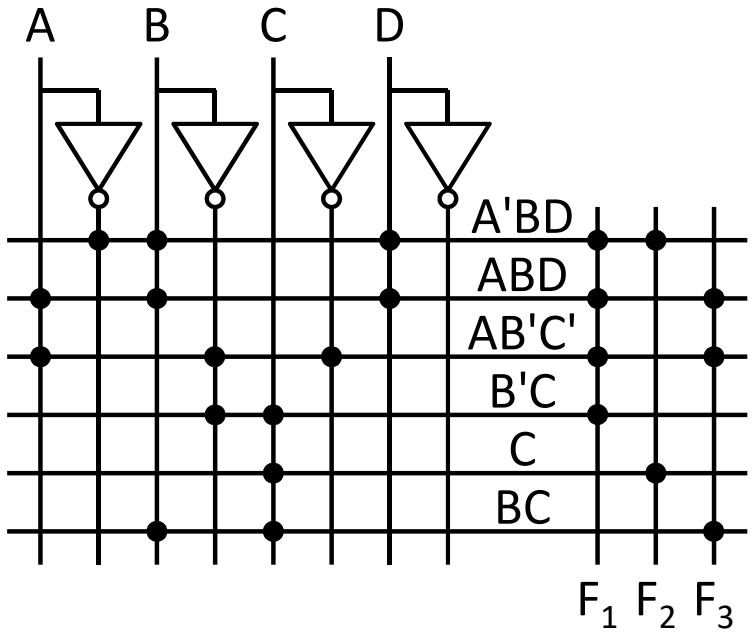
- Example
  - $F_1 = A'B' + AC'$
  - $F_2 = AC' + B$
  - $F_3 = A'B' + BC'$
  - $F_4 = B + AC$



Programmable Logic Arrays (4/4)

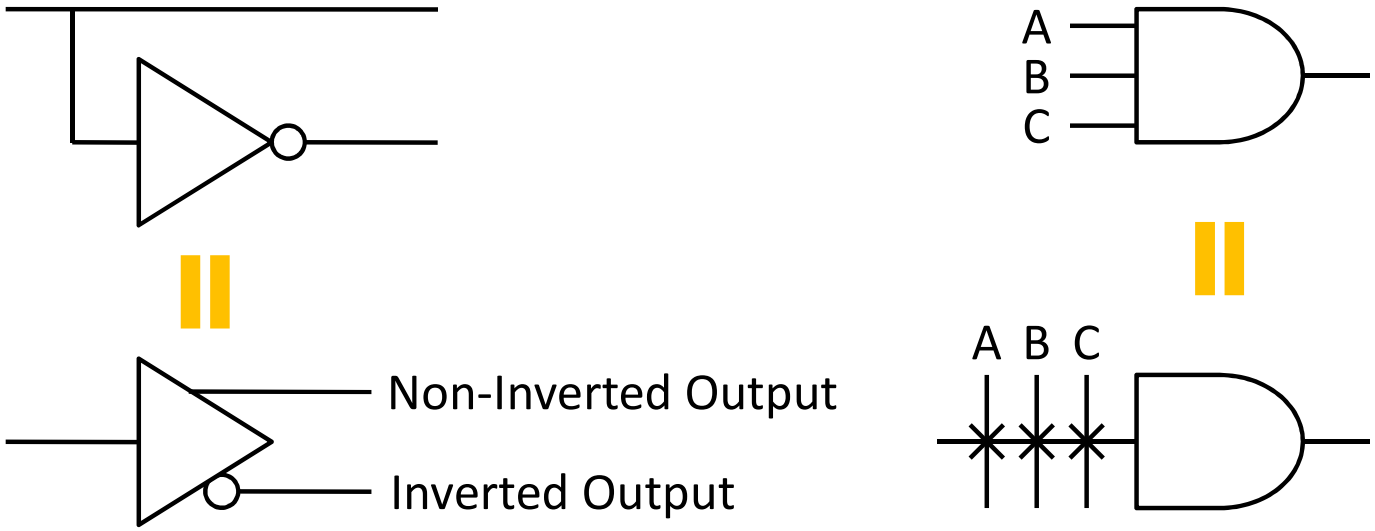
- Example
  - $F_1(A,B,C,D) = \sum m(2,3,5,7,8,9,10,11,13,15)$
  - $F_2(A,B,C,D) = \sum m(2,3,5,6,7,10,11,14,15)$
  - $F_3(A,B,C,D) = \sum m(6,7,8,9,13,14,15)$
- Minimize using K-map
  - F_1 = A'BD + ABD + AB'C' +B'C
  - F_2 = C + A'BD
  - F_3 = BC + AB'C' + ABD

	A	B	C	D	F_1	F_2	F_3
A'BD	0	1	–	1	1	1	0
ABD	1	1	–	1	1	0	1
AB'C'	1	0	0	–	1	0	1
B'C	–	0	1	–	1	0	0
C	–	–	1	–	0	1	1
BC	–	1	1	–	0	0	1



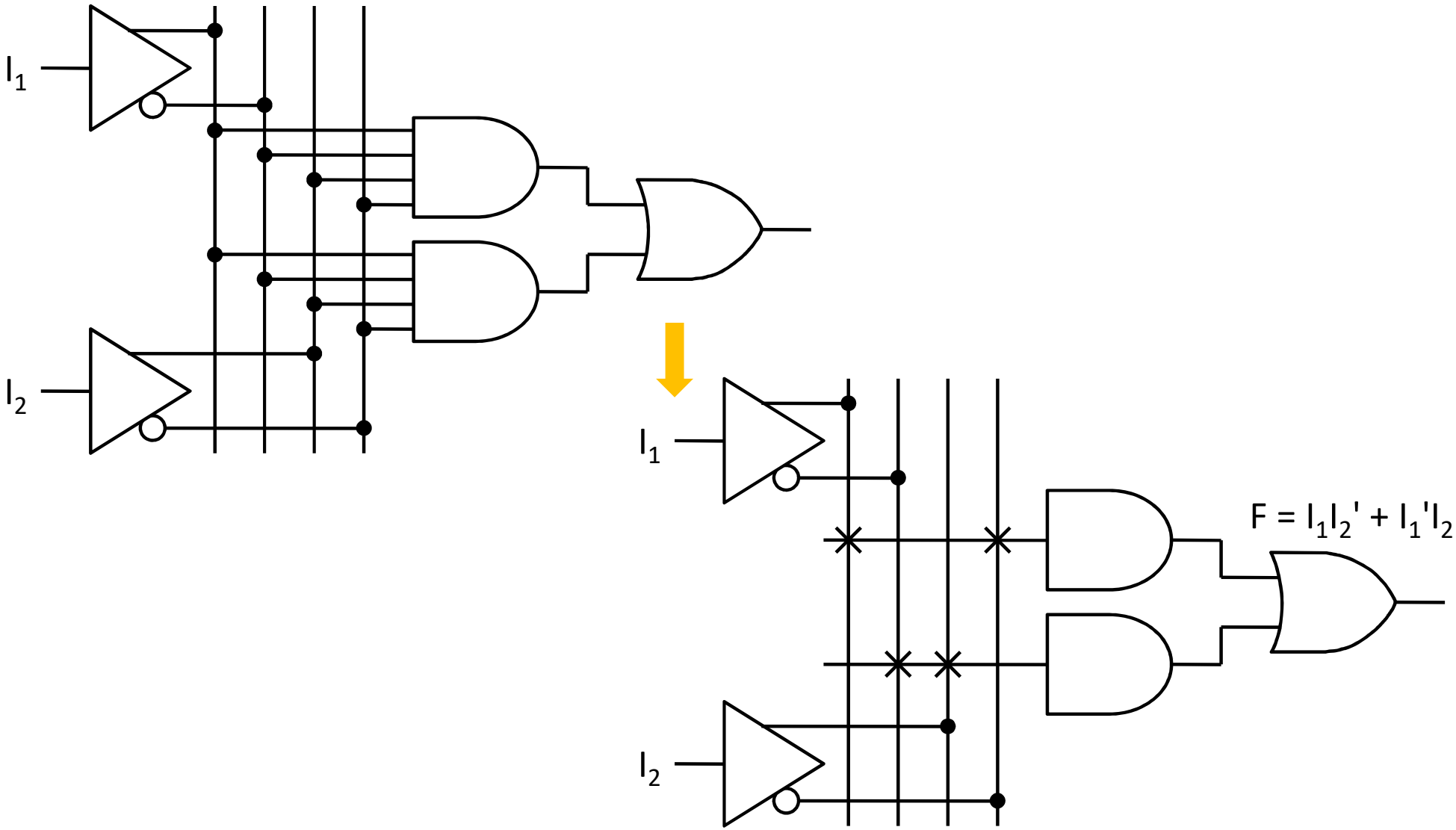
Programmable Array Logic (1/2)

- Programmable Array Logic (PAL): a special case of PLA
  - AND array: programmable
  - OR array: fixed
- Symbol



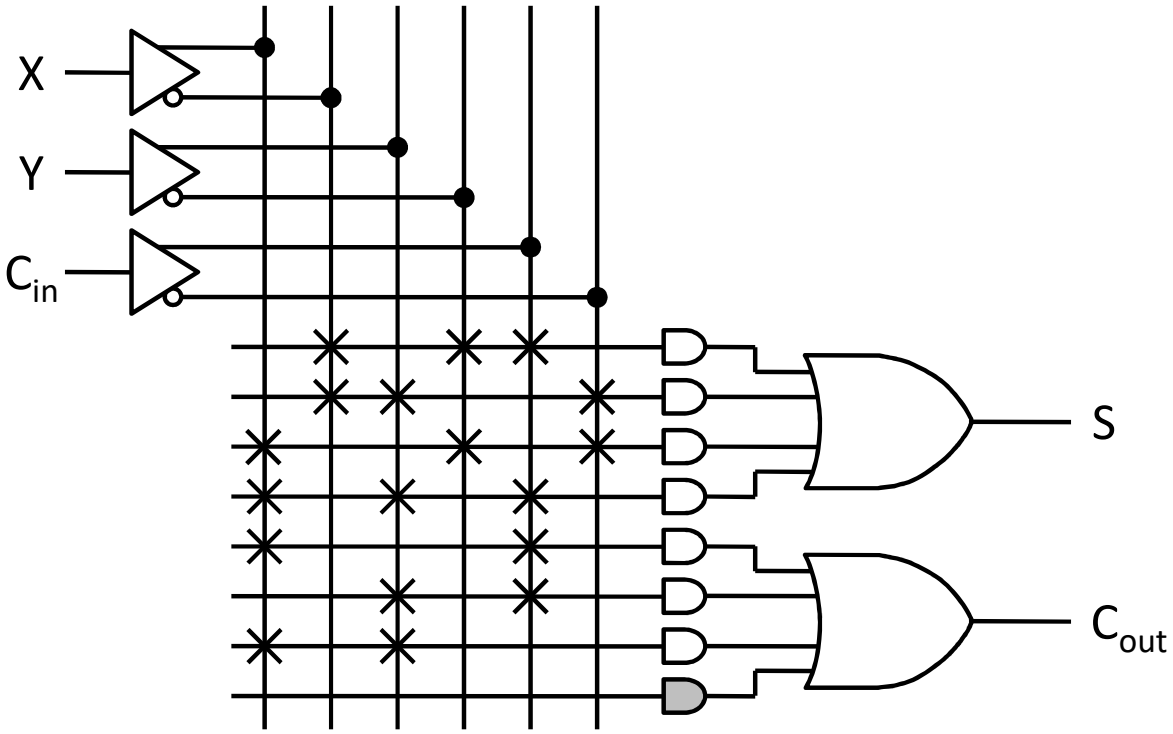
Programmable Array Logic (2/2)

- Fixed OR array



Application Full Adder

- $S = X'Y'C_{in} + X'YC_{in}' + XY'C_{in}' + XYC_{in}$
- $C_{out} = XY + XC_{in} + YC_{in}$



X	Y	C_{in}	C_{out}	S
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

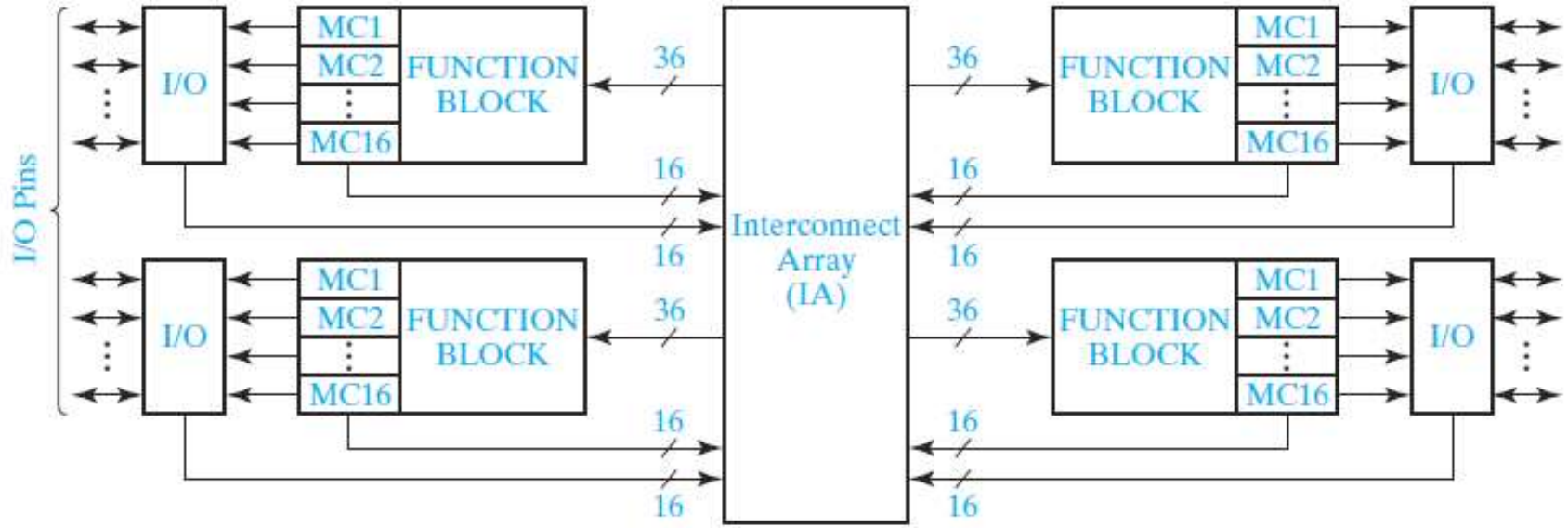
Outline

- ❑ Multiplexers
- ❑ Three-State Buffers
- ❑ Decoders and Encoders
- ❑ Read-Only Memories
- ❑ Programmable Logic Devices
- ❑ **Complex Programmable Logic Devices**
- ❑ Field-Programmable Gate Arrays

Complex Programmable Logic Devices

- Complex Programmable Logic Device (CPLD): integrates and interconnects many PALs and PLAs on a single chip
  - Tools will program for you
  - Example: Xilinx XCR3064XL

FIGURE 9-34 Architecture of Xilinx XCR3064XL CPLD (Figure based on figures and text owned by Xilinx, Inc., Courtesy of Xilinx, Inc. © Xilinx, Inc. 1999–2003. All rights reserved.)

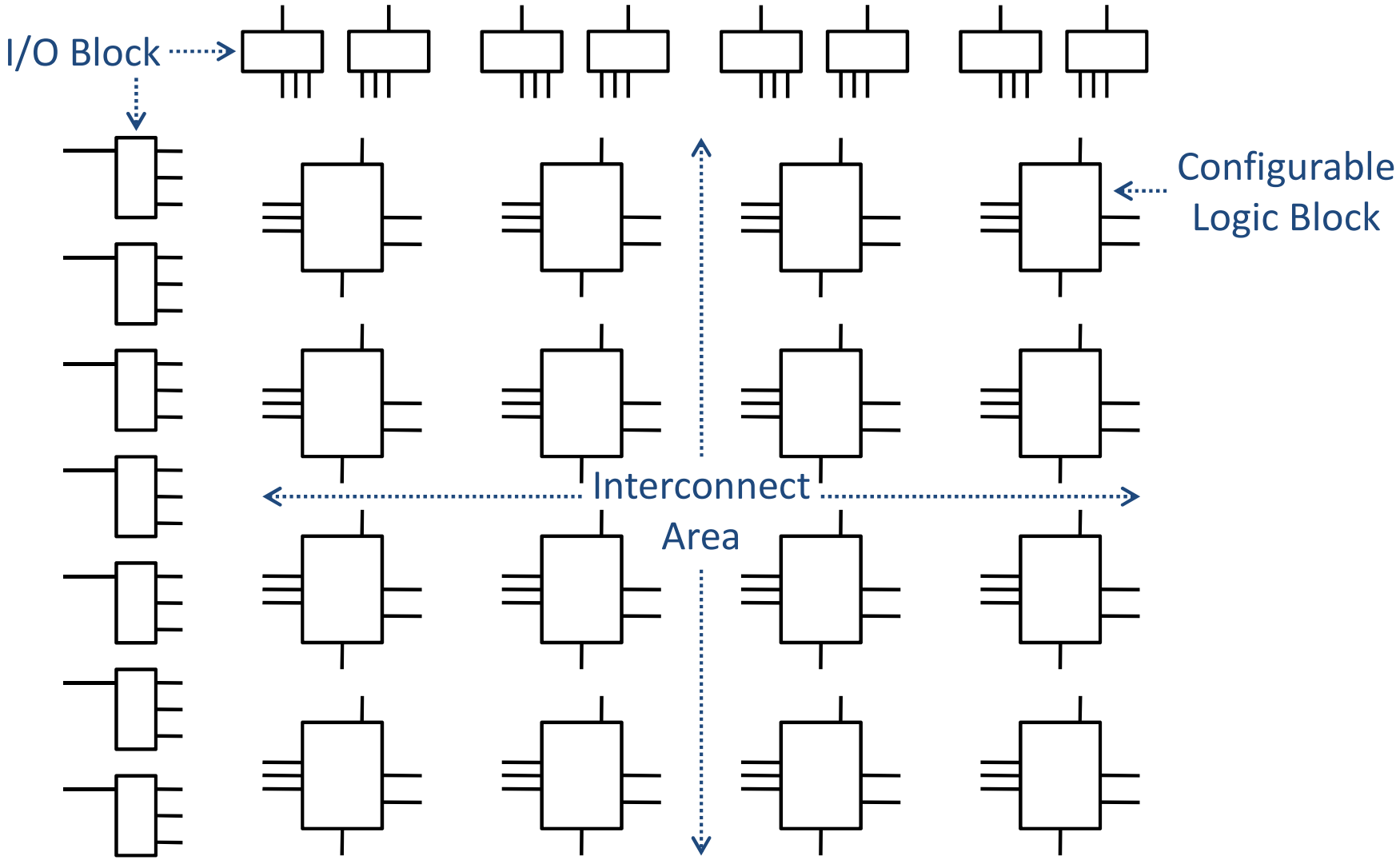


Outline

- ❑ Multiplexers
- ❑ Three-State Buffers
- ❑ Decoders and Encoders
- ❑ Read-Only Memories
- ❑ Programmable Logic Devices
- ❑ Complex Programmable Logic Devices
- ❑ **Field-Programmable Gate Arrays**

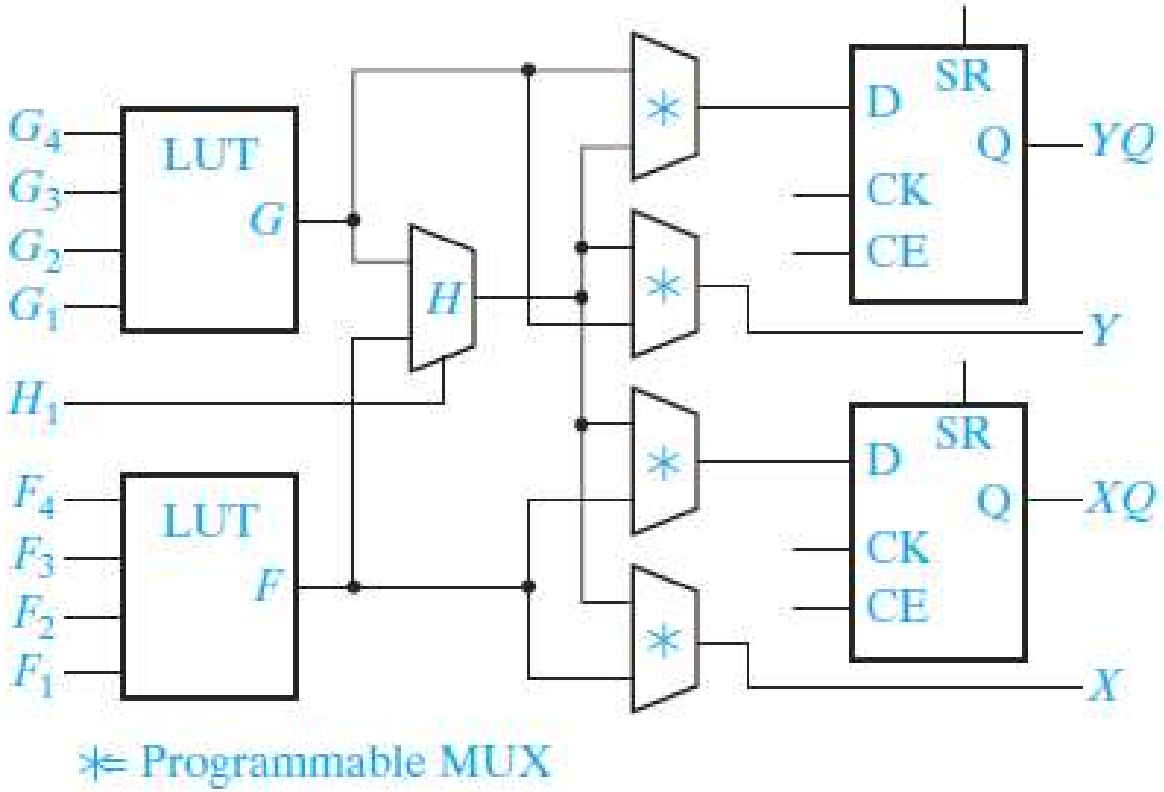
Field Programmable Gate Arrays

- Field Programmable Gate Arrays (FPGA): an array of identical logic cells + programmable interconnections



Simplified Configurable Logic Block

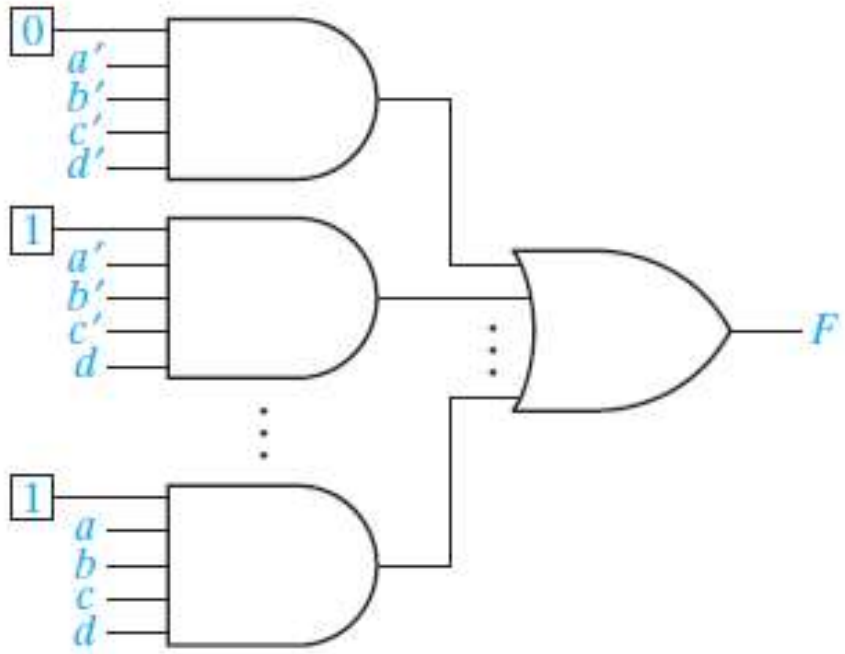
FIGURE 9-37
Simplified Configurable Logic Block (CLB)



Implementation of a Lookup Table (LUT)

FIGURE 9-38
Implementation of a Lookup Table (LUT)

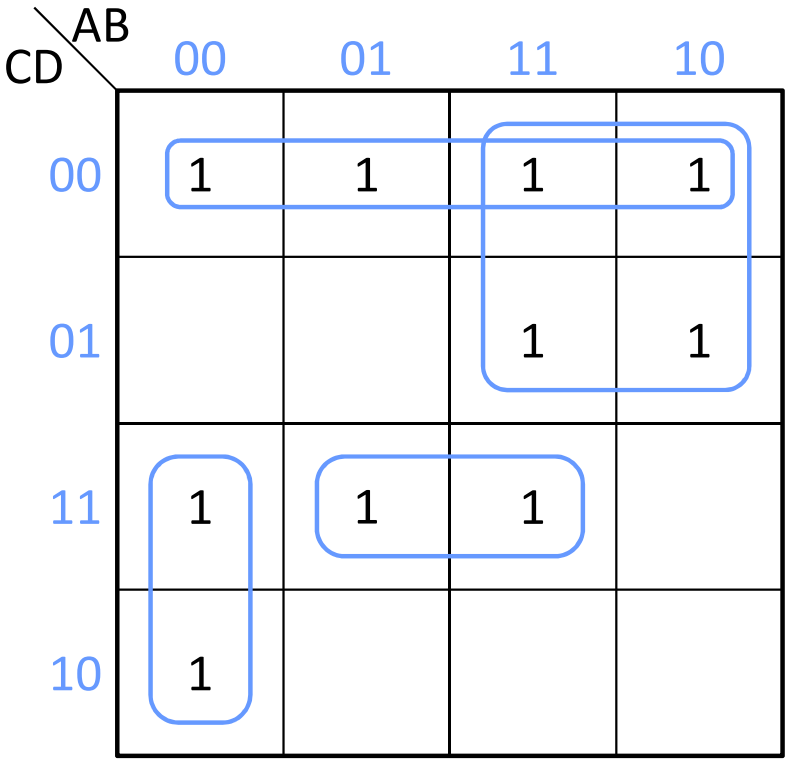
a	b	c	d	F
0	0	0	0	0
0	0	0	1	1
⋮				⋮
1	1	1	1	1



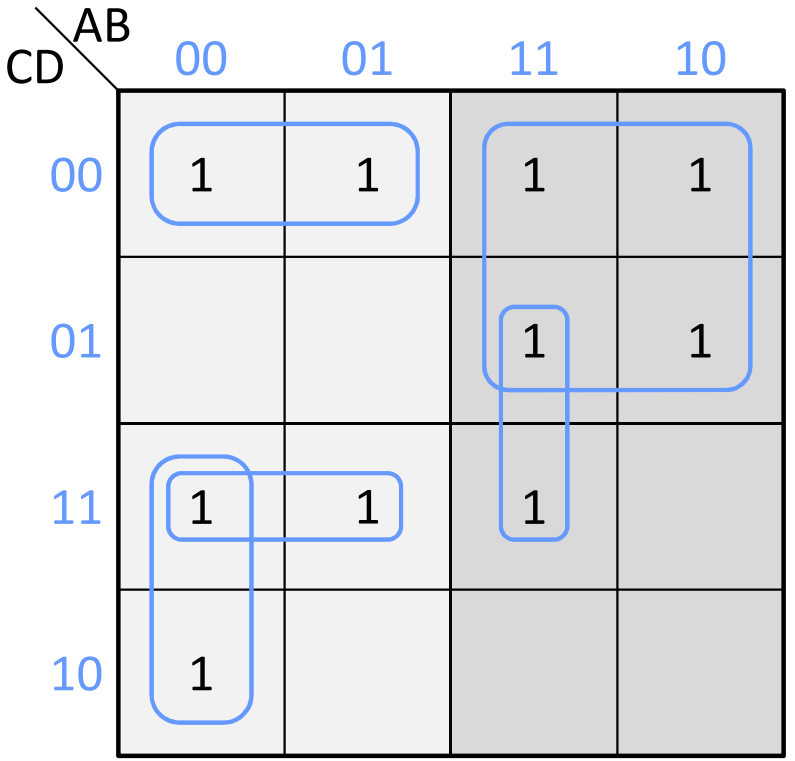
Shannon's Expansion

Shannon's expansion theorem

- $f(x_1,x_2,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_n)$
  $= x'_i \bullet f(x_1,x_2,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n) + x_i \bullet f(x_1,x_2,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n)$



$F = C'D' + AC' + A'B'C + BCD$



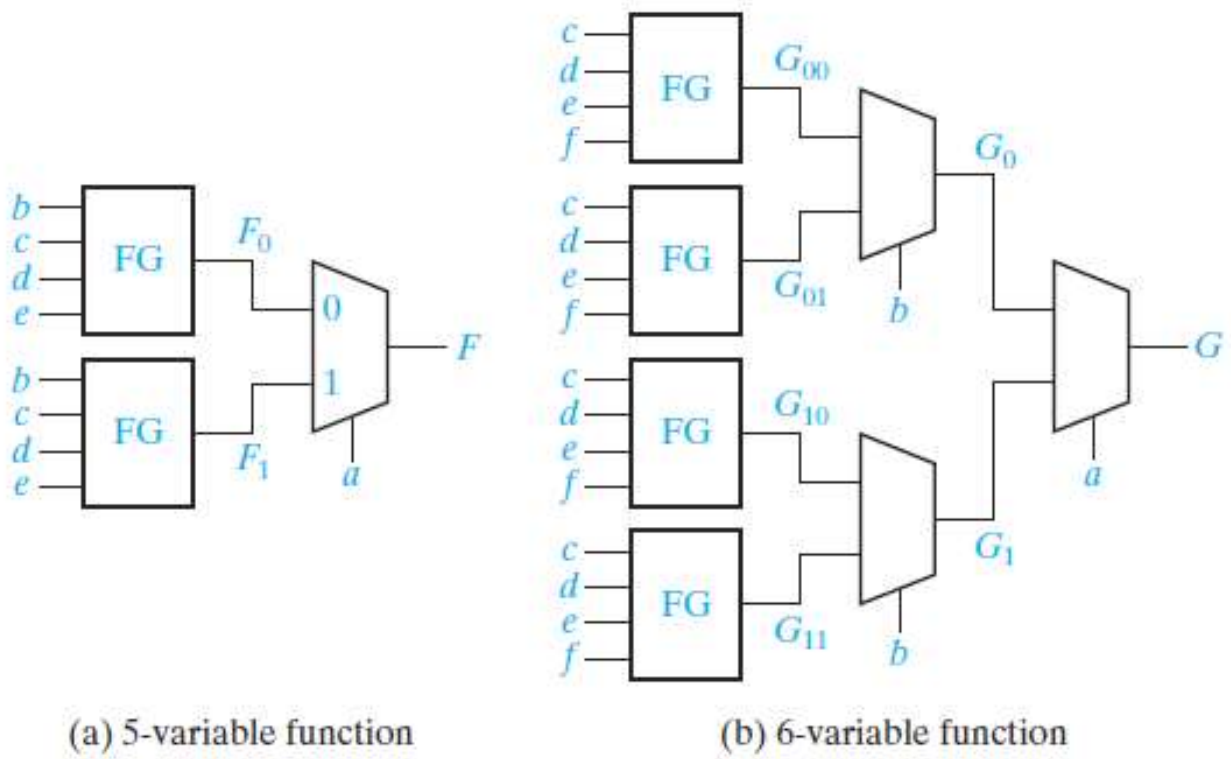
$F_0 = C'D' + CD + B'C \quad F_1 = C' + BD$

$F = A' \bullet F_0 + A \bullet F_1$

Realization with Function Generators

- Realize a 5-variable function with 4-variable FGs
  - F(A,B,C,D,E) = A' • F(0,B,C,D,E) + A • F(1,B,C,D,E) = A' • F_0 + A • F_1
  - Two 4-variable function generators + one 2-to-1 MUX (controlled by A)
- How about 6-variable function?

FIGURE 9-40
Realization of 5- and 6-Variable Functions with Function Generators



(a) 5-variable function (b) 6-variable function

Q&A