CS2022: 數位系統設計

Memory and Programmable Logic

Outline

- Introduction
- Random-Access Memory
- Memory Decoding
- Error Detection and Correction
- Read-Only Memory
- Programmable Logic Array
- Programmable Array Logic
- Sequential Programmable Devices

Introduction

- **Memory**
  - **Information storage**
  - **A collection of cells store binary information**
- **RAM – Random-Access Memory**
  - **Read operation**
  - **Write operation**
- **ROM – Read-Only Memory**
  - **Read operation only**
  - **A programmable logic device**

Programmable Logic Device

- **Programmable Logic Device (PLD)**
  - **ROM – read only memory**
  - **PLA – programmable logic array**
  - **PAL – programmable array logic**
  - **FPGA – field-programmable gate array**
    - **programmable logic blocks**
    - **programmable interconnects**

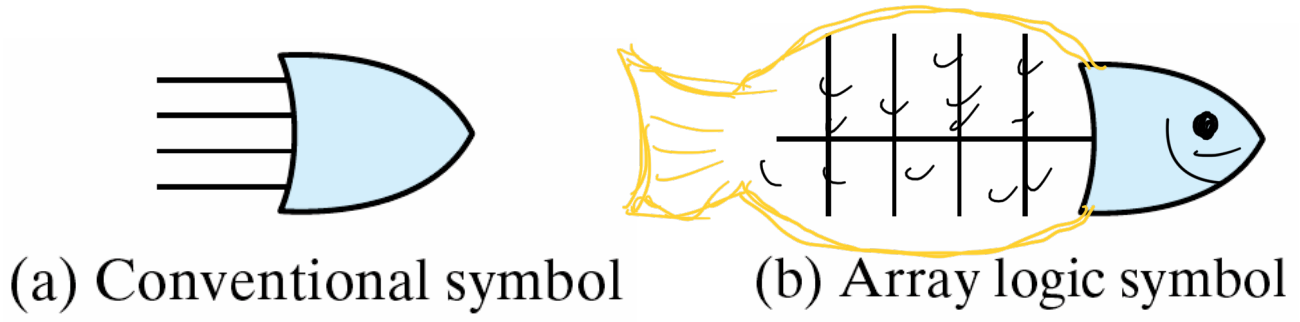


Fig. 1 Conventional and array logic diagrams for OR gates

Random-Access Memory

- **A memory unit**
  - **Stores binary information in groups of bits (words)**
  - **8 bits (1 byte), 2 bytes, 4 bytes**
- **Block diagram**

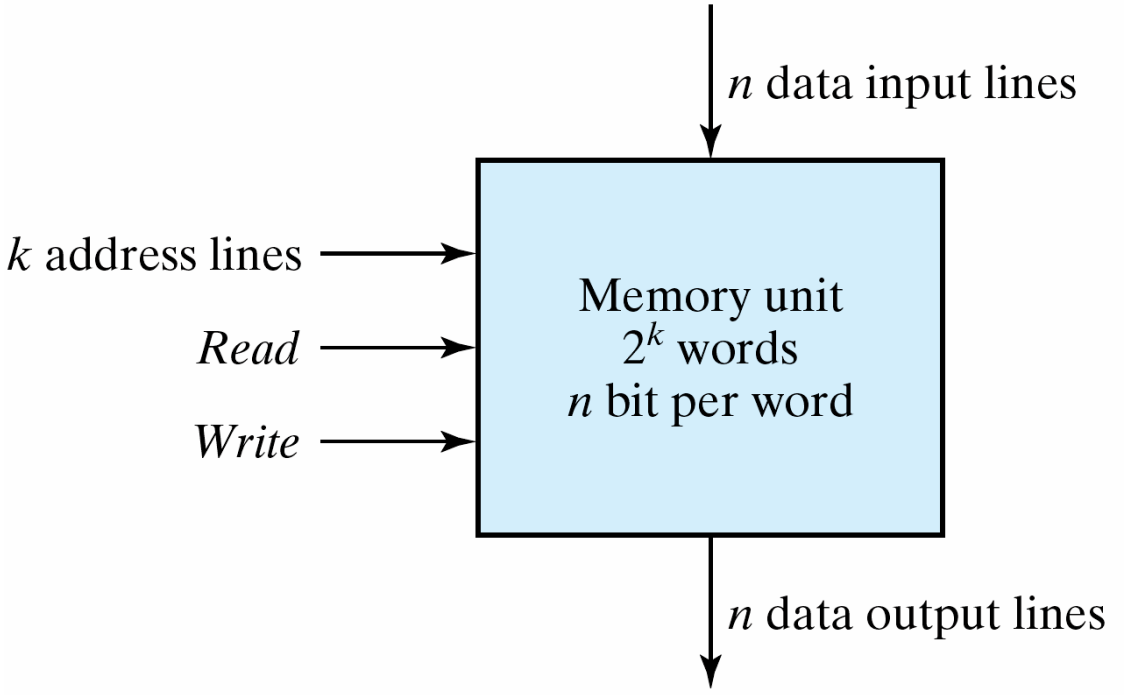


Fig. 2 Block diagrams of a memory unit

Example: 1024 × 16 Memory

Memory address		Memory content
Binary	Decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

Fig. 3 Contents of a 1024 × 16 memory

Write and Read Operations

- **Write operation**
  - Apply the binary address to the address lines
  - Apply the data bits to the data input lines
  - Activate the *write* input
- **Read operation**
  - Apply the binary address to the address lines
  - Activate the *read* input

Table 7.1
Control Inputs to Memory Chip

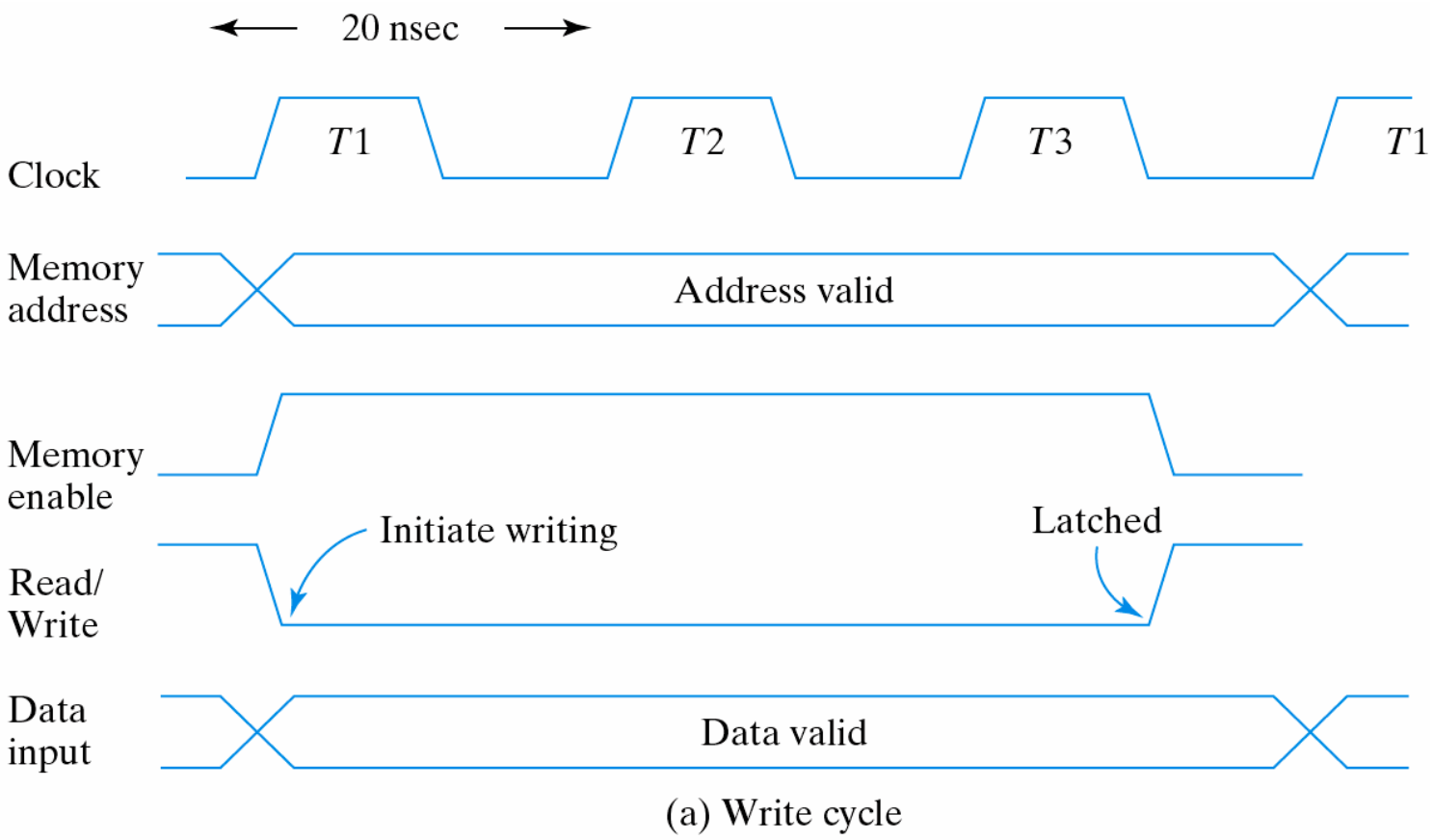
Memory Enable	Read/Write	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

Timing Waveforms

- The operation of the memory unit is controlled by an external device
- The memory access time
  - the time required to select a word and read it
- The memory cycle time
  - the time required to complete a write operation
- Read and write operations must be synchronized with a clock
  - Usually, CPU clock cycle time < memory access/cycle time
  - Multiple CPU clock cycles for a memory operation

Memory Write Cycle

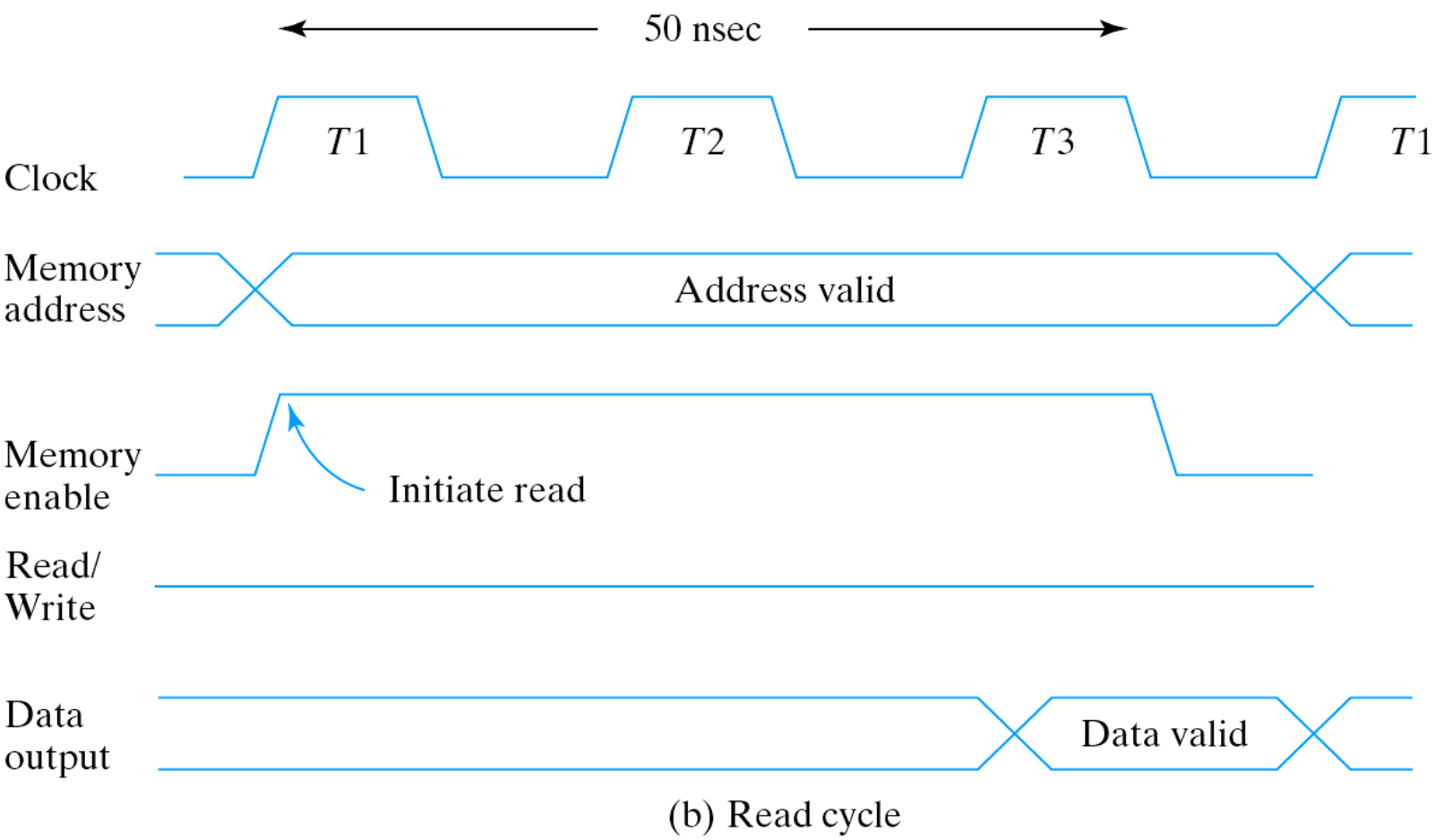
- **Assume that CPU clock: 50 MHz (i.e., cycle time = 20 ns)**
- **Memory access/cycle time < 50 ns**



(a) Write cycle

Fig. 4 Memory cycle timing waveforms

Memory Read Cycle



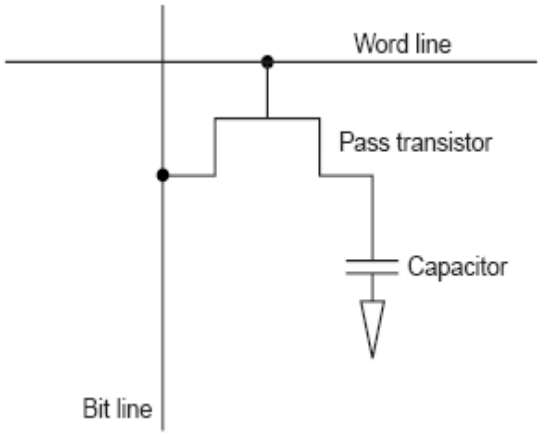
(b) Read cycle

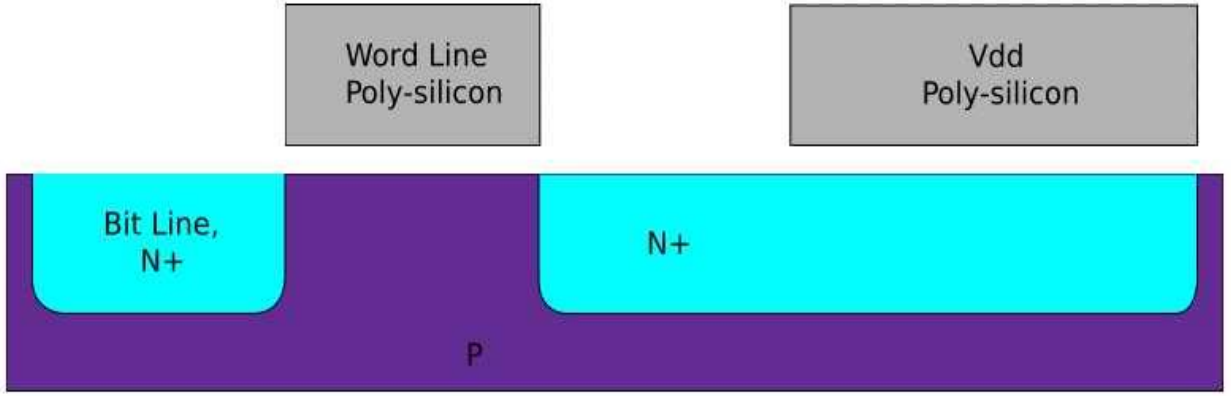
Fig. 4 Memory cycle timing waveforms

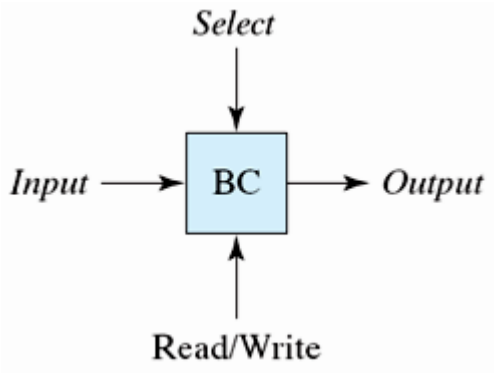
Types of Memories (1/2)

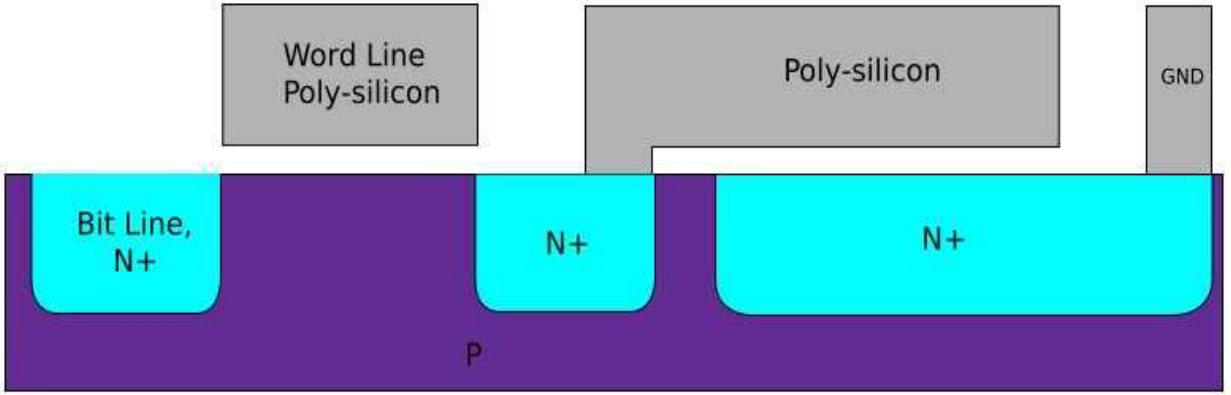
- **Random-access memory – RAM**
  - Access time is the same regardless the data location
  - Cp.: hard disk, CD-ROM, DVD-ROM, tape (sequential access)
- **Static memory – SRAM**
  - Information is stored in latches
  - Remains valid as long as power is applied
  - Short read/write cycle
- **Dynamic memory – DRAM**
  - Information are stored in the form of charges on capacitors
  - The stored charge tends to discharge with time
  - Need to be refreshed (read and write back)
  - Reduced power consumption
  - Larger memory capacity

DRAM Cell



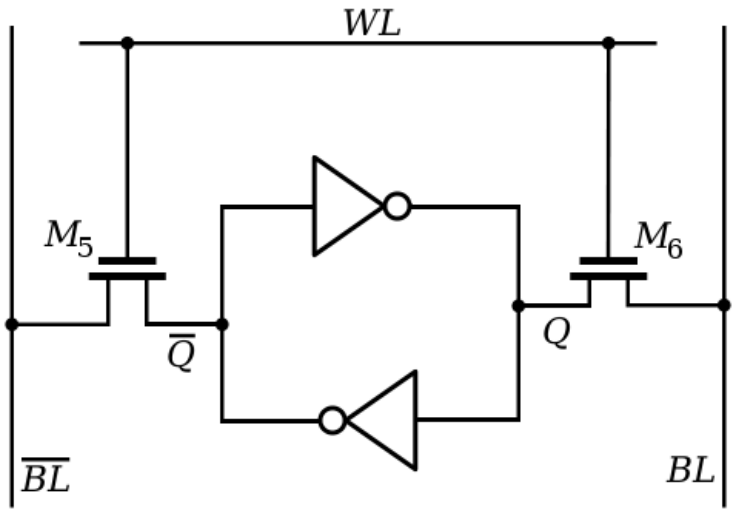


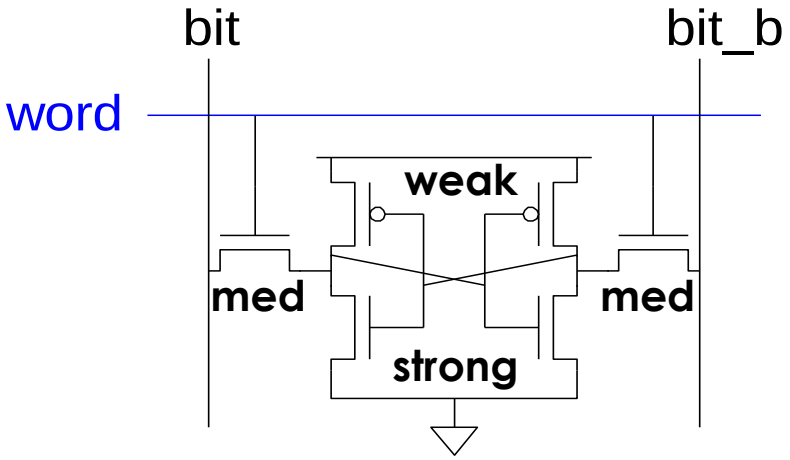


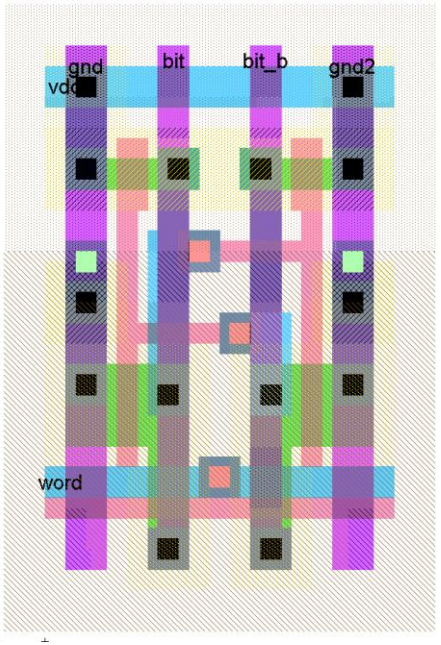


Source: Wikipedia

SRAM Cell


WL
M_5
M_6
$\overline{Q}$
Q
$\overline{BL}$
BL


bit
bit_b
word
weak
med
med
strong


gnd
vdd
bit
bit_b
gnd2
word

Types of Memories (2/2)

- **Volatile**
  - Lose stored information when power is turned off
  - SRAM, DRAM
- **Non-volatile**
  - Retains its stored information after the removal of power
  - ROM
  - EPROM, EEPROM
  - Flash memory

Memory Decoding

- **A memory unit**
  - **The storage components**
  - **The decoding circuits to select the memory word**

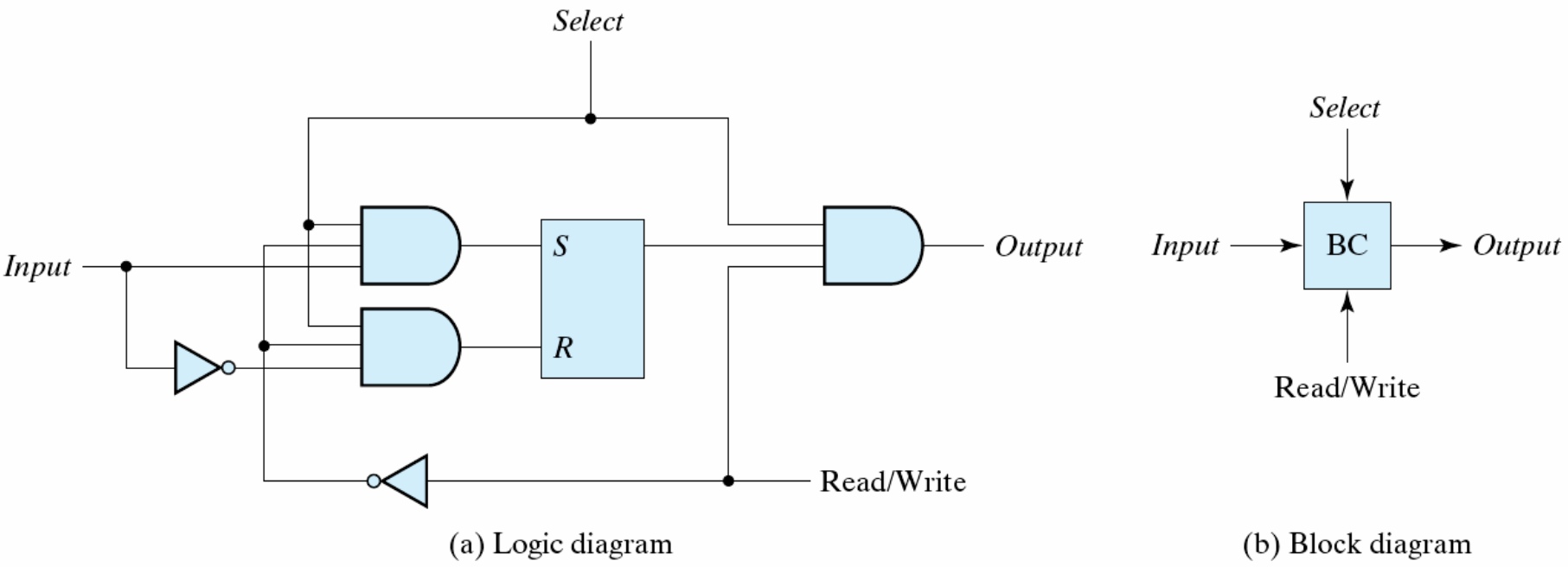


Fig. 5 Memory cell (equivalent logic)

Internal Construction

- **A RAM of *m* words and *n* bits per word**
  - ***m**n binary storage cells**
  - **Decoding circuits to select individual words**
    - » ***k*-to-2^k decoder**

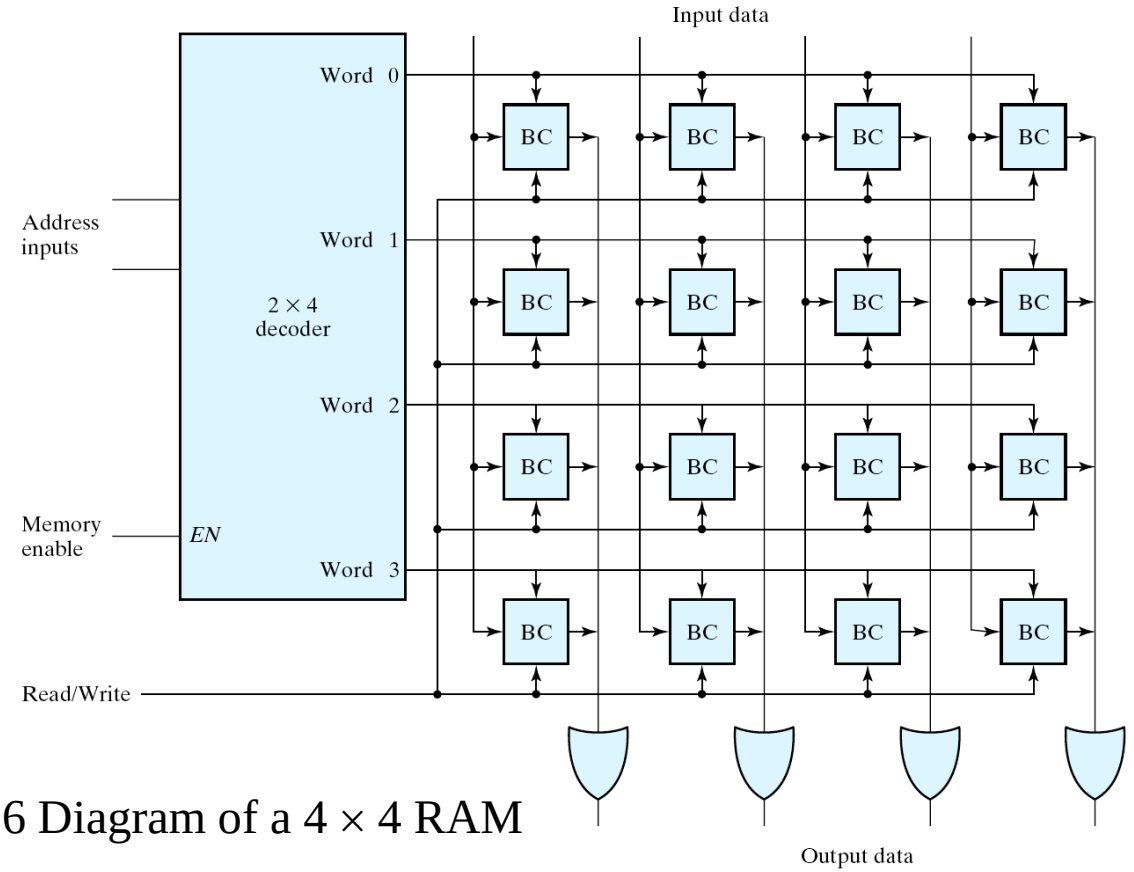


Fig. 6 Diagram of a 4 × 4 RAM

Coincident Decoding (1/2)

- A two-dimensional selection scheme
  - Reduce the complexity of the decoding circuits

- A 10-to-1024 decoder
  - 1024 AND gates with 10 inputs per gates
  - Two 5-to-32 decoders
    - 2 * (32 AND gates with 5 inputs per gates)
  - Reduce the circuit complexity and the cycle time

Coincident Decoding (2/2)

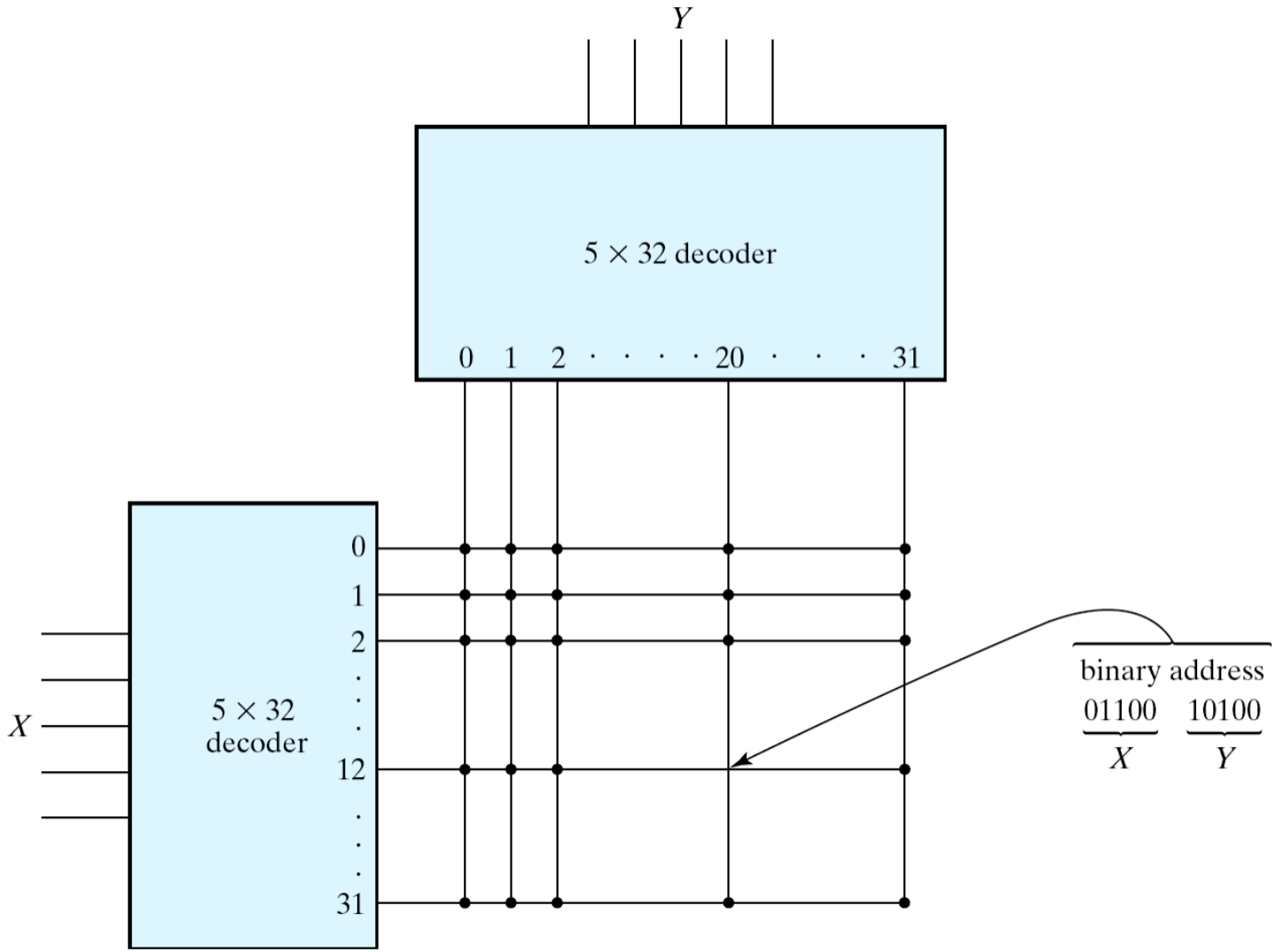


Fig. 7 Two-dimensional decoding structure for a 1K-word memory

Address Multiplexing (1/2)

- To reduce the number of pins in the IC package
  - consider a 64K×1 DRAM
    - 16-bit address lines
  - Multiplex the address lines in one set of address input pins
    - RAS: 8-bit row address strobe
    - CAS: 8-bit column address strobe

Address Multiplexing (2/2)

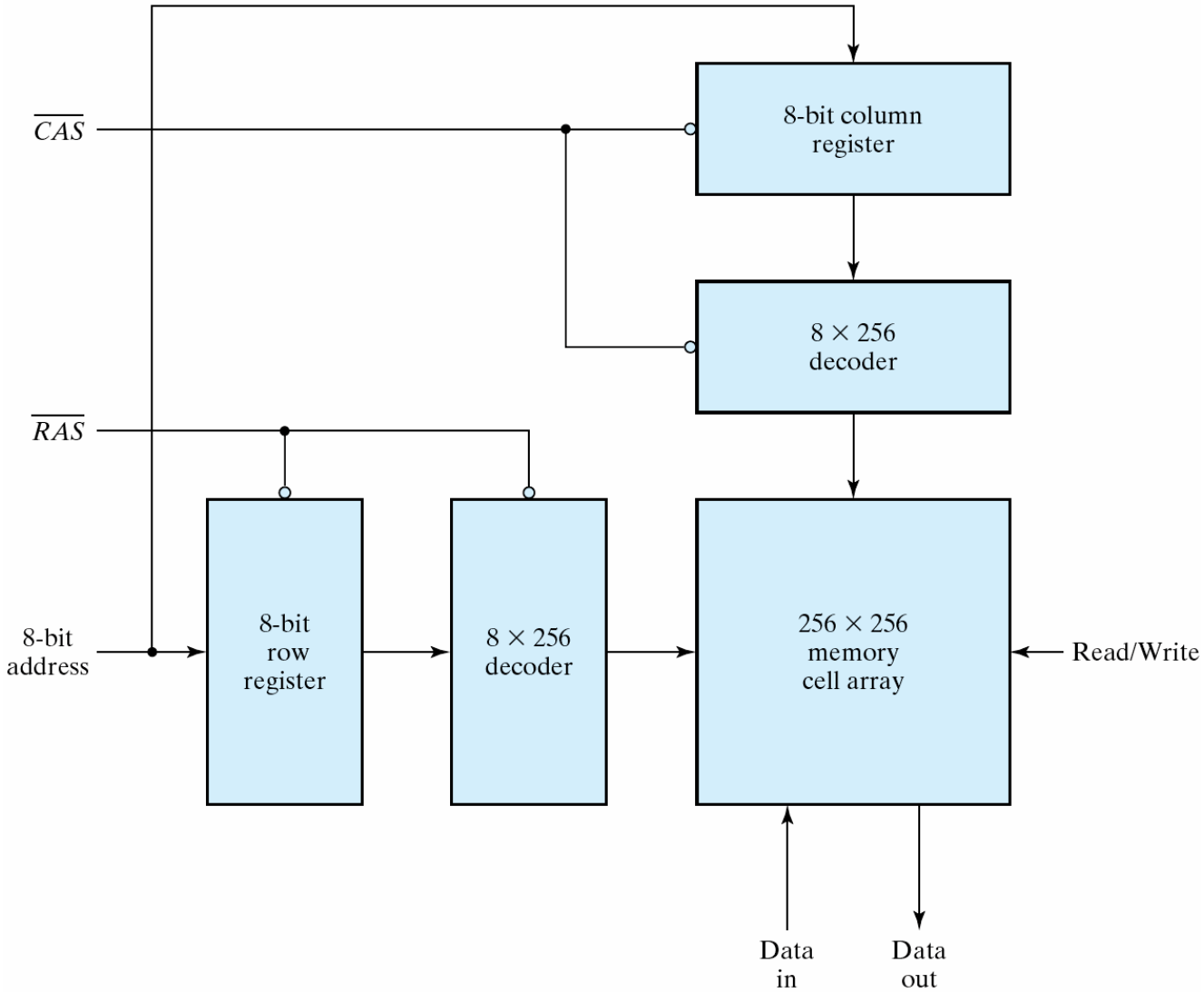


Fig. 8 Address multiplexing for a 64K DRAM

Read-Only Memory

- **Store permanent binary information**
- **2^k x n ROM**
  - ***k* address input lines**
  - **Enable input(s)**
  - **Three-state outputs**

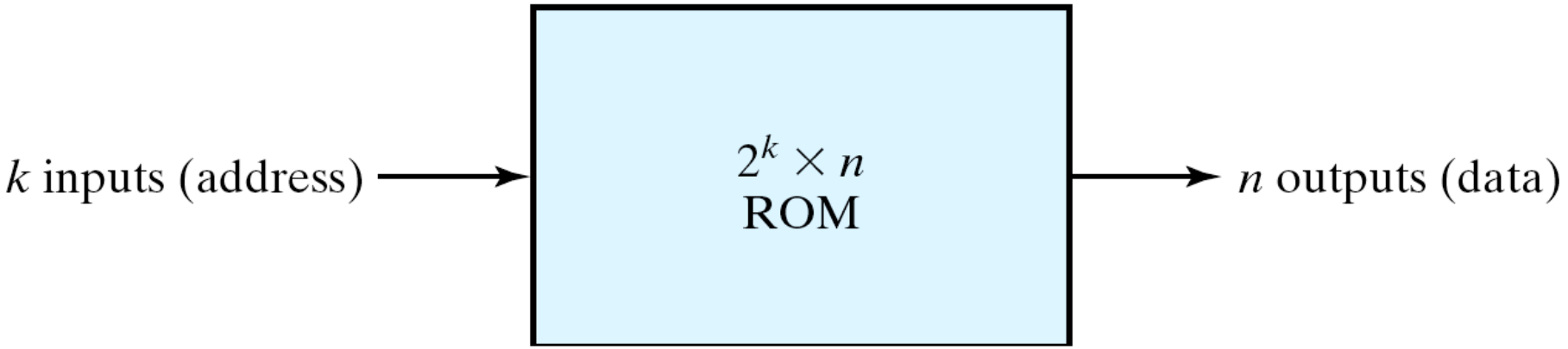


Fig. 9 ROM block diagram

Internal Logic of a 32×8 ROM

- 5-to-32 decoder
- 8 OR gates
  - » Each has 32 inputs
- 32x8 internal programmable connections

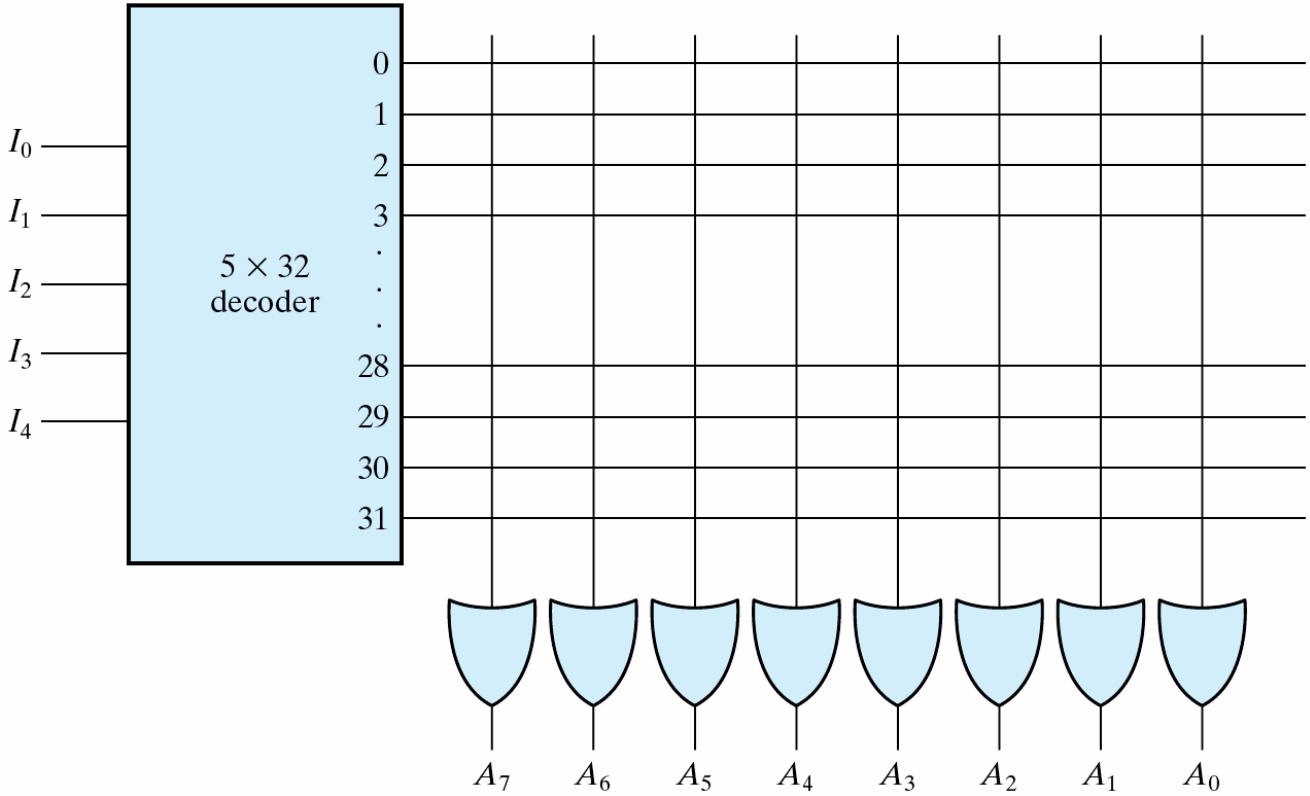


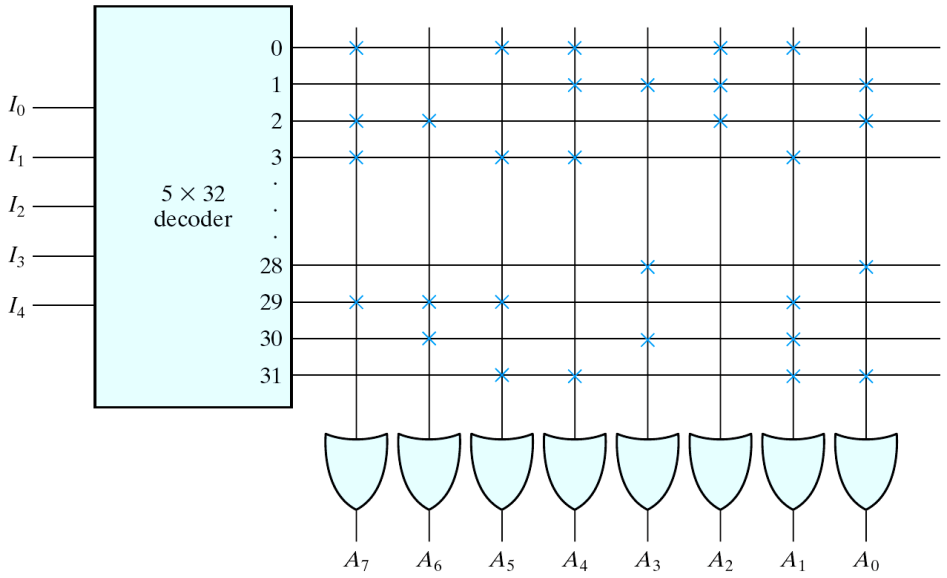
Fig. 10 Internal logic of a 32 × 8 ROM

Programming a ROM

- **Programmable interconnections**
  - **Crosspoint switch**
    - » **Each "fuse" can be blown by applying a high voltage pulse**
  - **Close (two lines are connected)**
  - **Open (two lines are disconnected)**

Table 7.3
ROM Truth Table (Partial)

Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮						⋮				
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1



Combinational Circuit Implementation

- **ROM: a decoder + OR gates**
  - Boolean function = sum of minterms
  - For an *n*-input, *m*-output combinational circuit

    $\Rightarrow 2^n \times m$ ROM
- **Design procedure:**
  1. Determine the size of ROM
  2. Obtain the programming truth table of the ROM
  3. The truth table = the fuse pattern

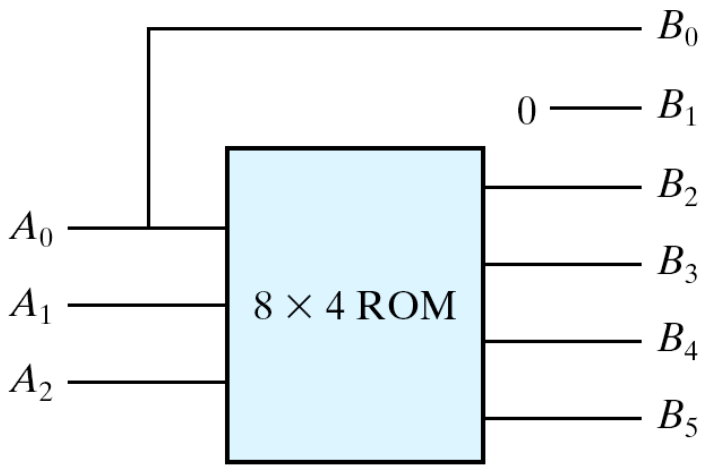
Three-Bit Binary Square (1/2)

Table 7.4
Truth Table for Circuit of Example 7.1

Inputs			Outputs						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

- 3 inputs, 6 outputs
- $B_1 = 0$
- $B_0 = A_0$
- Use 8x4 ROM

Three-Bit Binary Square (2/2)

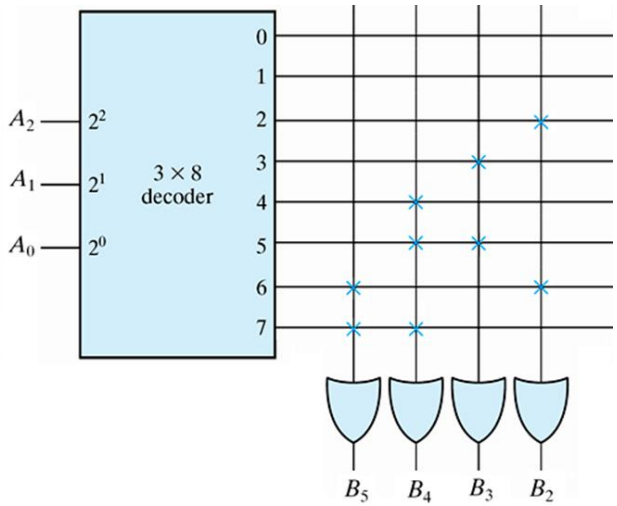


(a) Block diagram

A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

(b) ROM truth table

Fig. 12 ROM implementation of Example 1



Types of ROMs

- **Types of ROM**
  - **Mask programming ROM**
    - **IC manufacturers, economical only if large quantities**
  - **PROM: Programmable ROM**
    - **Fuses, one-time program**
  - **EPROM: erasable PROM**
    - **Floating gate**
    - **Ultraviolet light erasable**
  - **EEPROM: electrically erasable PROM**
    - **E^2PROM**
  - **Flash ROM**
    - **Widespread applications in recent years**
    - **Limited times of write operations ~ 10^5 erase cycles**

Combinational PLDs

- **Three major types of combinational PLD**
  - **AND-OR array**

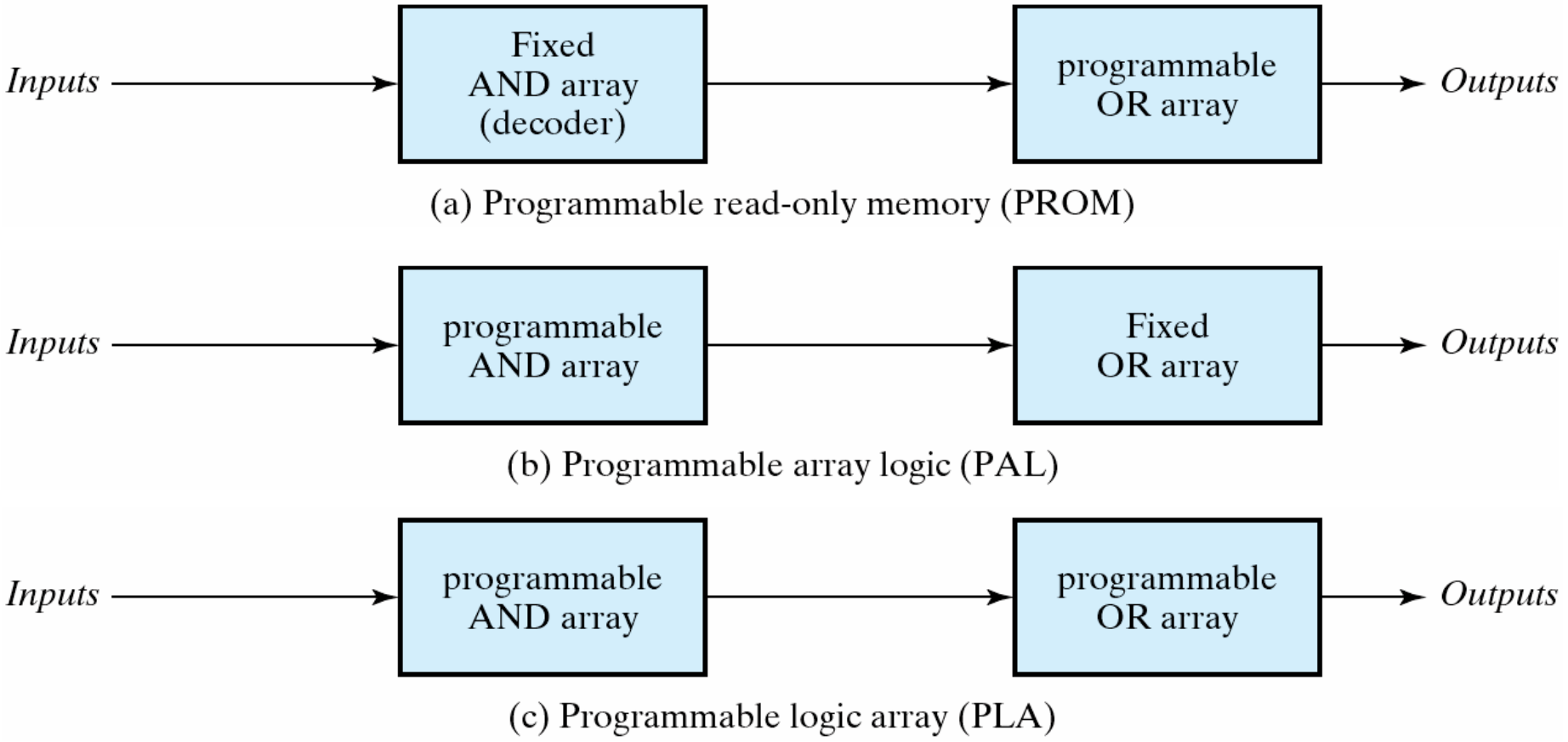


Fig. 13 Basic configuration of three PLDs

Programmable Logic Array

- **Programmable Logic Array – PLA**
  - **An array of programmable AND gates**
    - **Capable of generating any product term**
  - **An array of programmable OR gates**
    - **Capable of generating sum of the products**
  - **XOR gate at the output**
    - **Capable of generating positive and negative phases**
  - **More flexible than ROM**
  - **Use less circuits than ROM**
    - **Generate required product terms**

PLA Implementation (1/2)

$$F_1 = AB' + AC + A'BC'$$

$$F_2 = (AC + BC)'$$

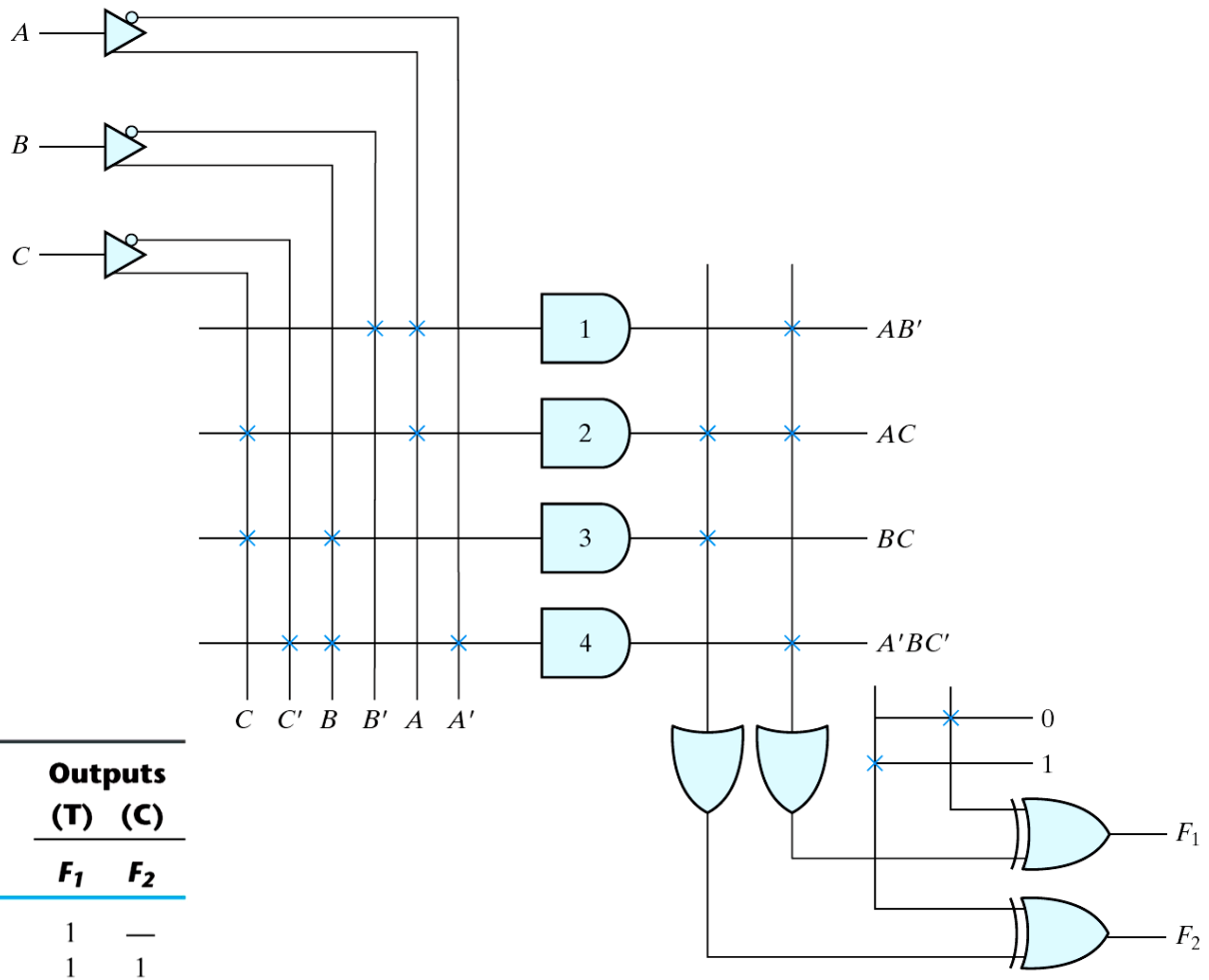


Table 7.5
PLA Programming Table

		Inputs			Outputs (T)	(C)
	Product Term	***A***	***B***	***C***	***F1***	***F2***
AB'	1	1	0	—	1	—
AC	2	1	—	1	1	1
BC	3	—	1	1	—	1
$A'BC'$	4	0	1	0	1	—

Note: See text for meanings of dashes.

PLA Implementation (2/2)

- **The size of a PLA**
  - **The number of inputs**
  - **The number of product terms (AND gates)**
  - **The number of outputs (OR gates)**
- **PLA design issues**
  - **Reduce the number of distinct product terms**
  - **The number of literals in a product is not important**

Examples 2 (1/2)

- $F_1(A, B, C) = \Sigma (0, 1, 2, 4)$
- $F_2(A, B, C) = \Sigma (0, 5, 6, 7)$
  - Both the true value and the complement of the function should be simplified to check

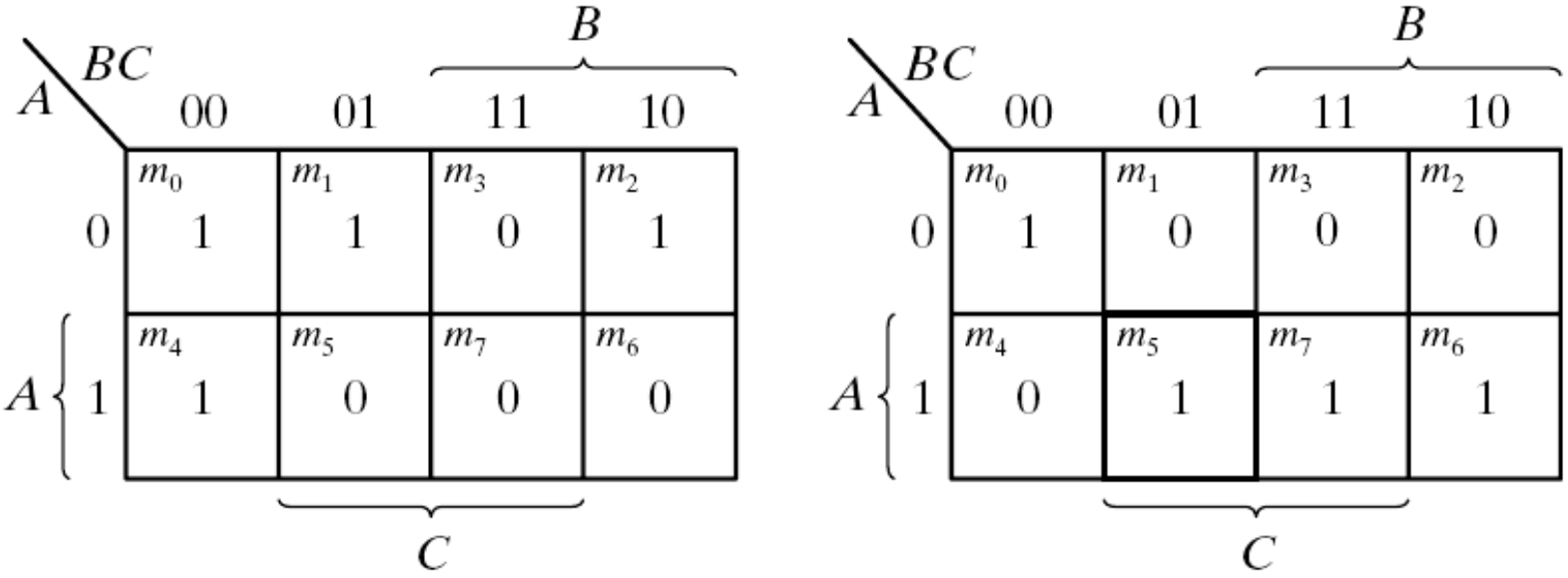
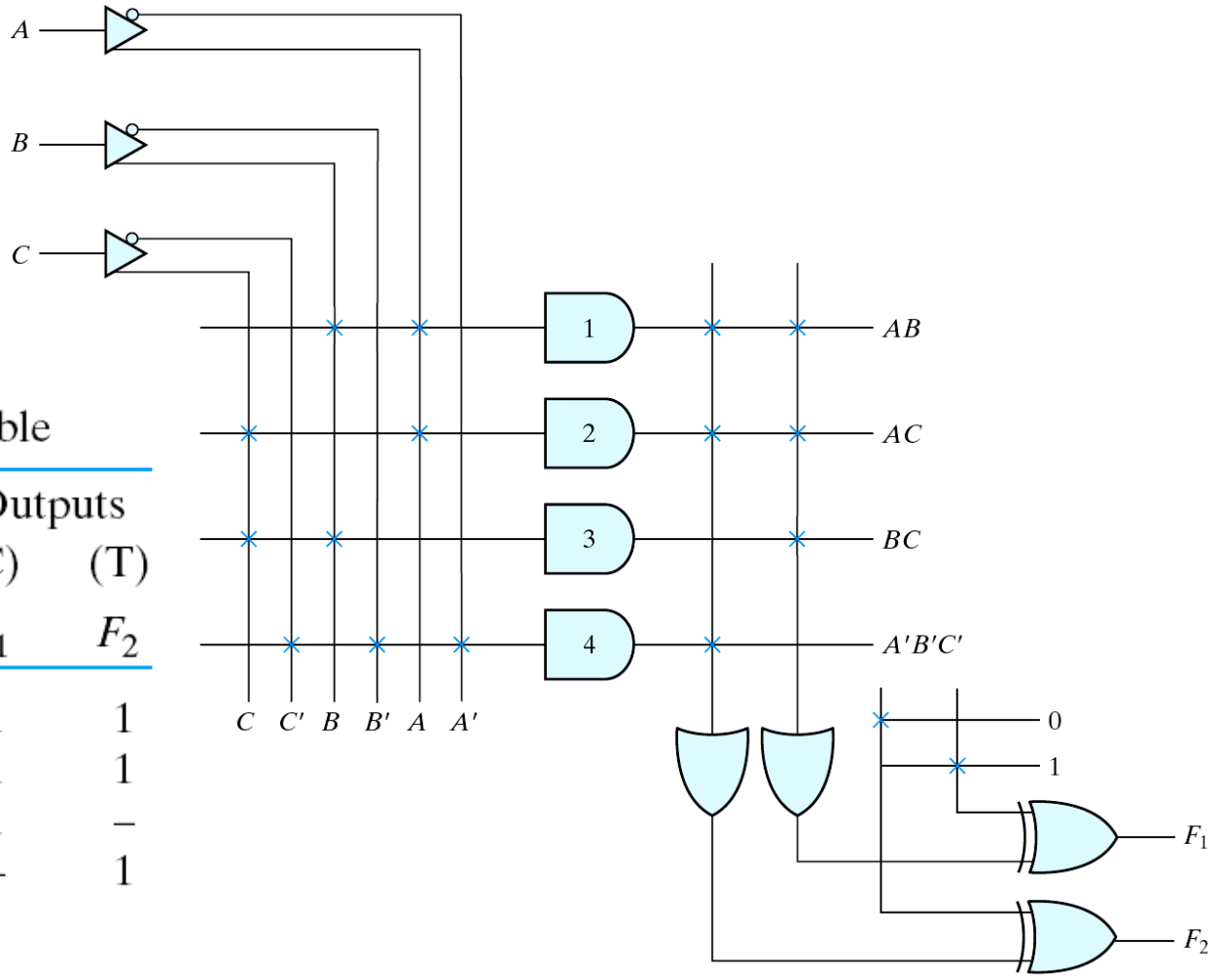


Fig. 15 Solution to Example 2

Examples 2 (2/2)

- $F_1 = (AB + AC + BC)'$
- $F_2 = AB + AC + A'B'C'$



PLA programming table

	Product term	Inputs A	B	C	Outputs (C) F_1	(T) F_2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
$A'B'C'$	4	0	0	0	–	1

Sequential Programmable Devices

- **Sequential programmable logic device**
  - **SPLD**
  - **PLD + filp-flops**

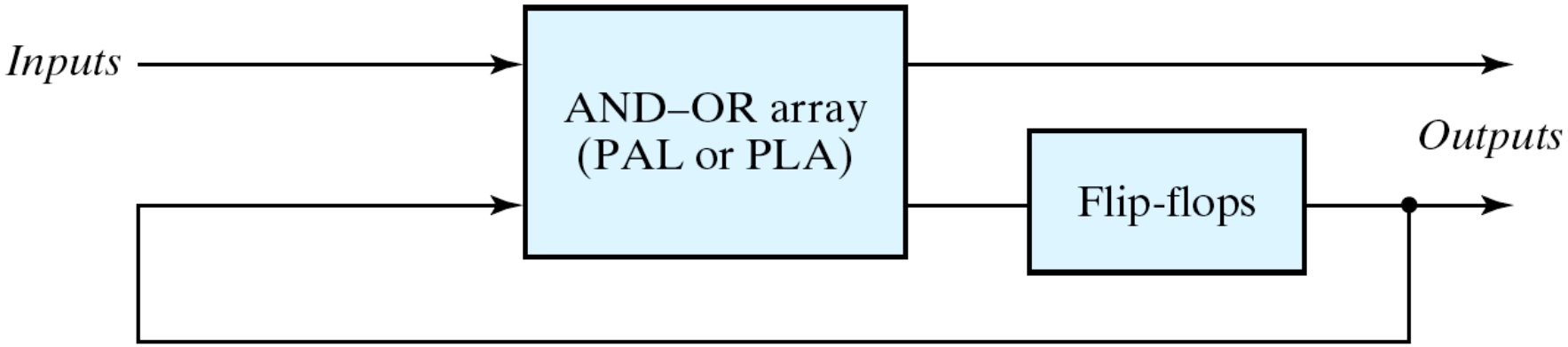


Fig. 18 Sequential programmable logic device

Complex PLD

- **Complex PLD – CPLD**
  - Put a lot of PLDS on a chip
  - Add wires between them whose connections can be programmed
  - Use fuse/EEPROM technology

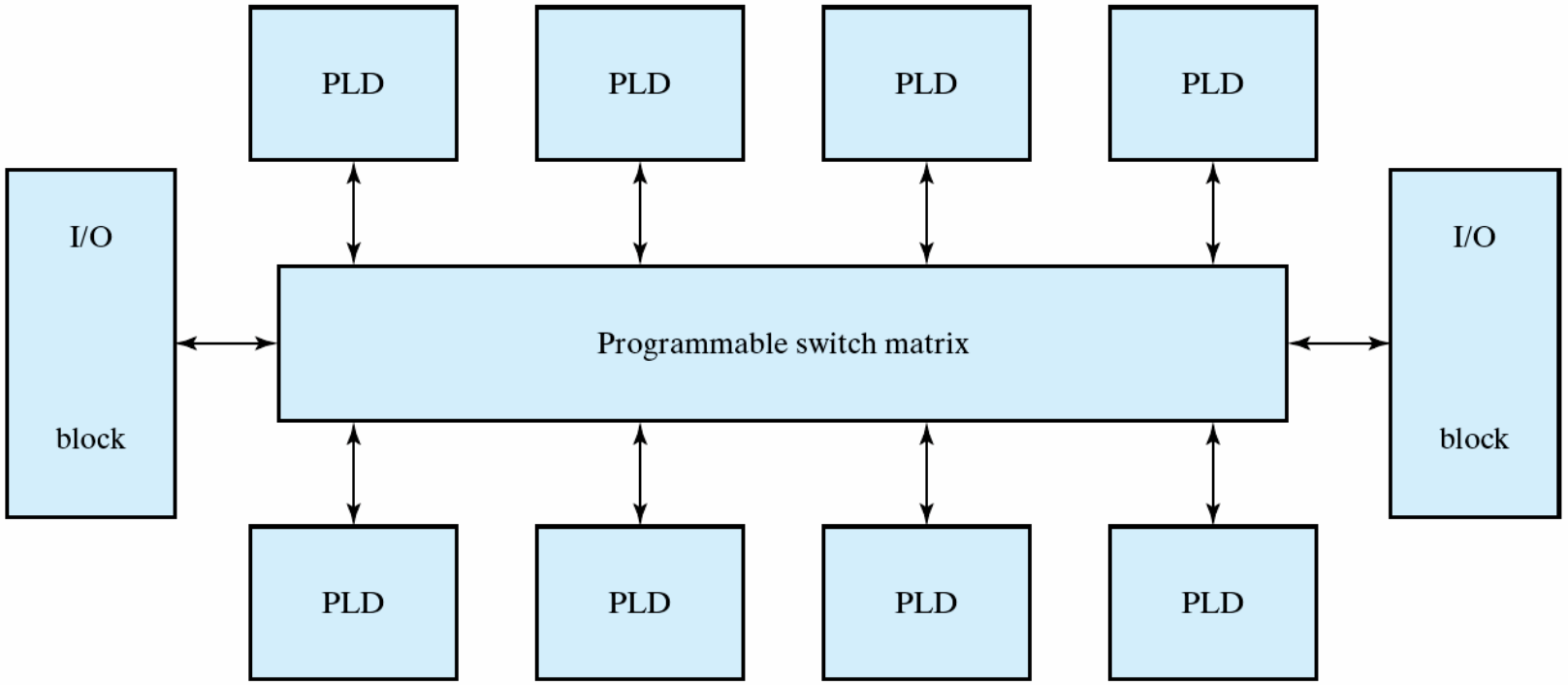
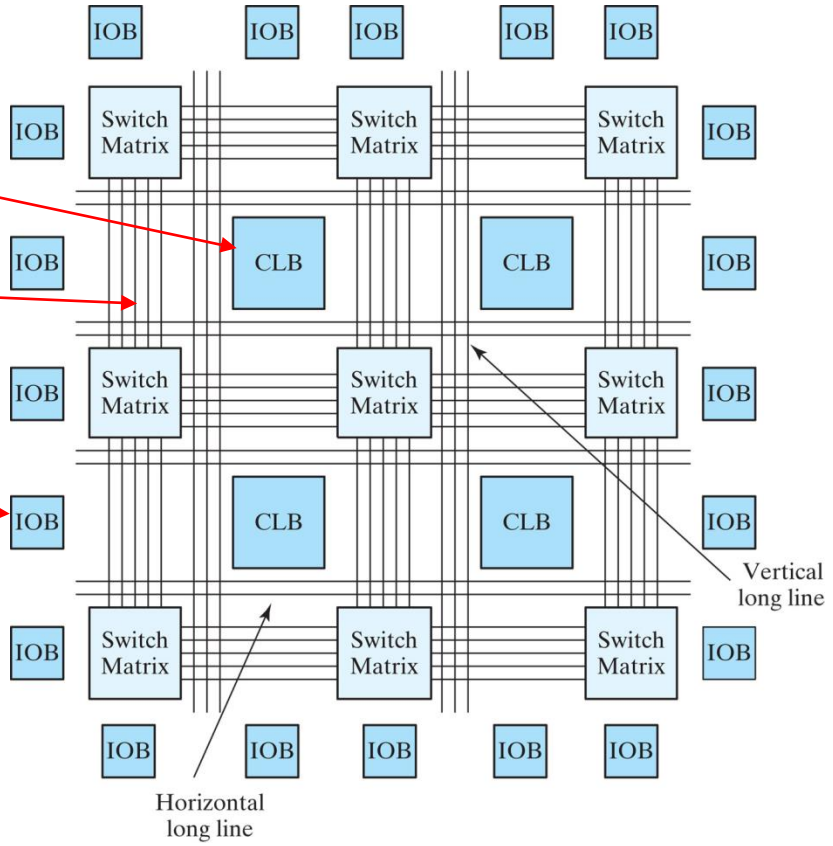


Fig. 20 General CPLD configuration

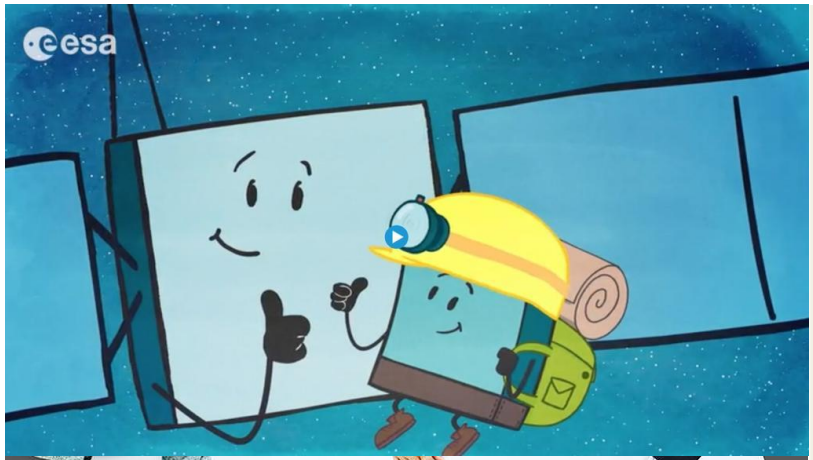
Field-Programmable Gate Arrays

- **Logic blocks**
  - To implement combinational and sequential logic
- **Interconnect**
  - Wires to connect inputs and outputs to logic blocks
- **I/O blocks**
  - Special logic blocks at periphery of device for external connections
- **Key questions**
  - How to make logic blocks programmable?
  - How to connect the wires?

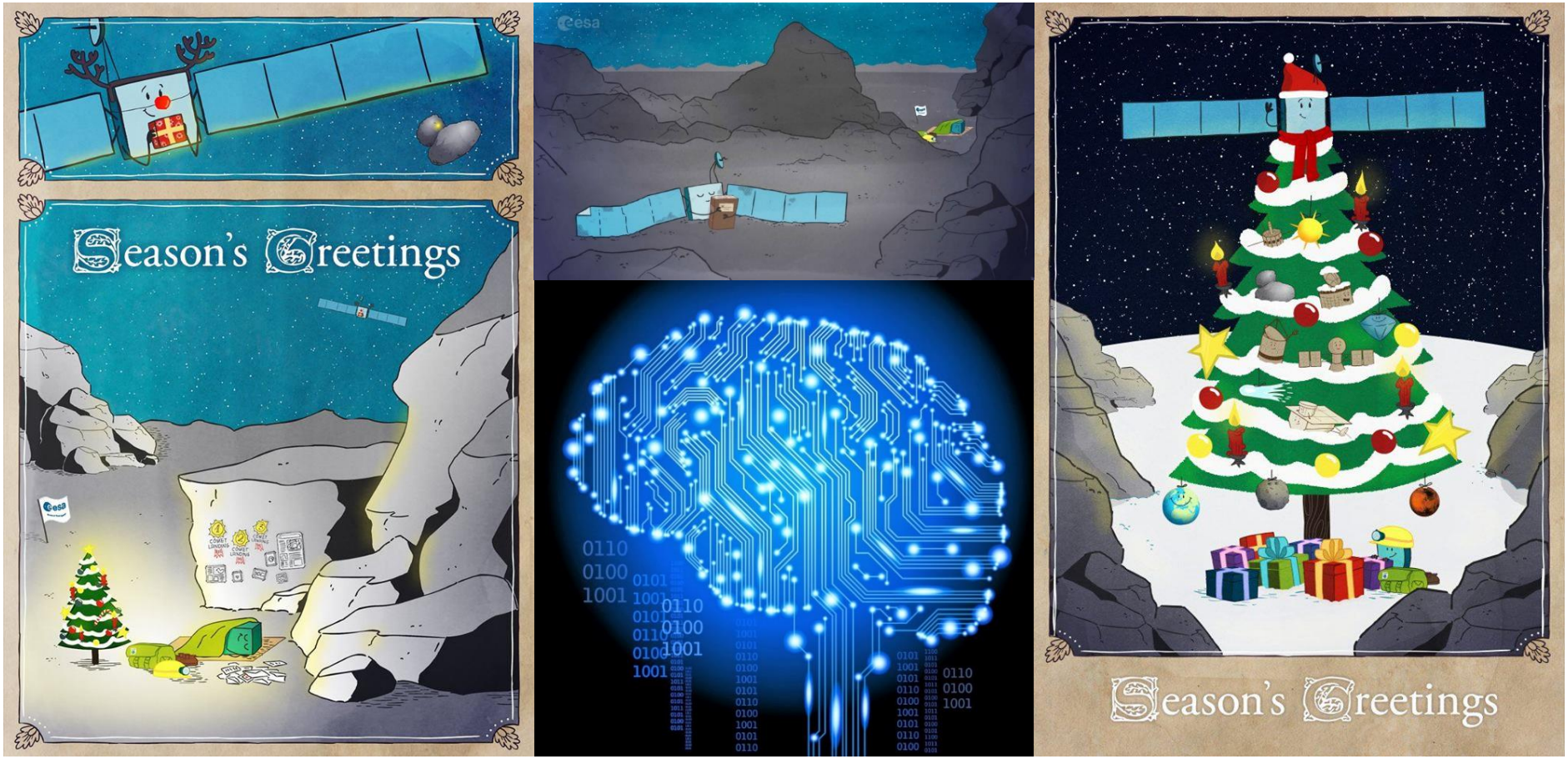


Digital Systems

- **Finite primitive elements create almost infinite possibilities!**
  - **Enabling technology for almost EVERYTHING we take for granted today!**



Farewell & Happy New Year



羅賽塔和菲萊的彗星大冒險
https://www.youtube.com/playlist?list=PLzYYnhQlXmVGDAJ9Dmn7V_alS5VBpZvMp

We thank you for pushing the frontiers of knowledge for the advancement of humanity. ☺