Memory Basics and Programmable Logics

Hsi-Pin Ma 馬席彬

http://lms.nthu.edu.tw/course/35472

Department of Electrical Engineering

National Tsing Hua University

Outline

- Random-Access Memory
- Memory Decoding
- Read-Only Memory
- Programmable Logic Array
- Programmable Array Logic
- Sequential Programmable Devices

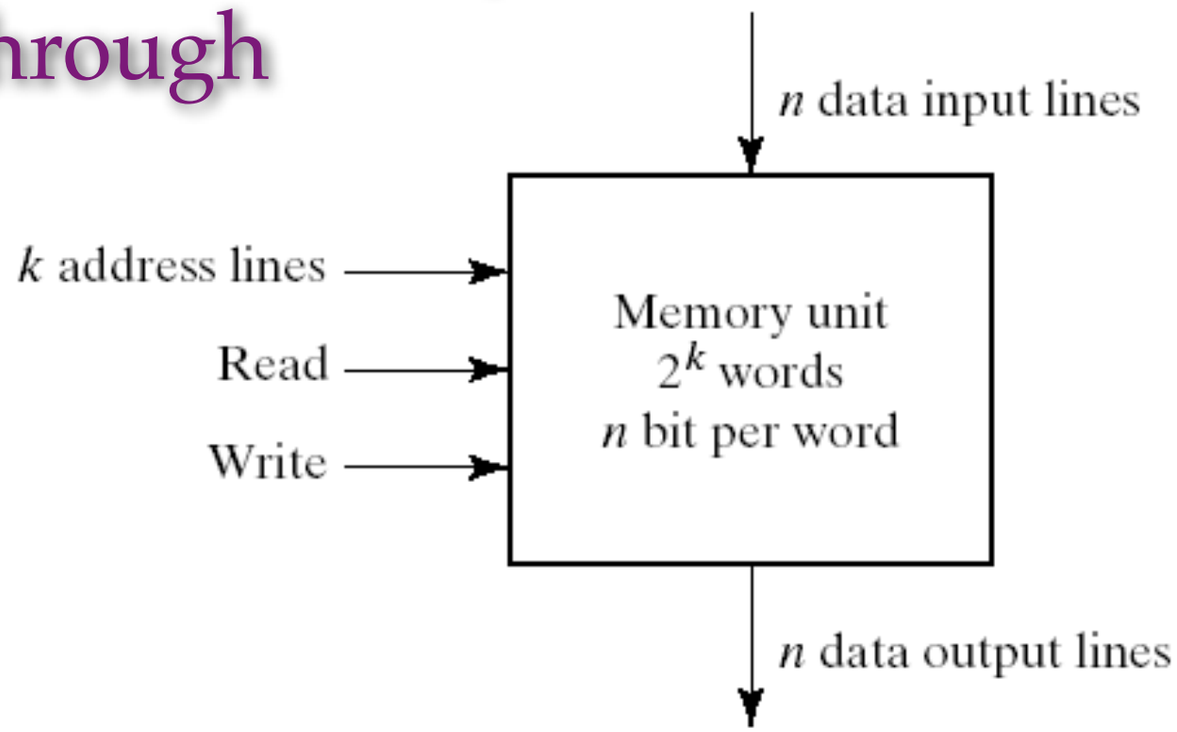
Memory Unit

- A collection of storage cells together with associated circuits needed to transfer information in and out of storage.
- RAM: Random-Access Memory
  - Volatile (memory units that lose stored information when power is turned off)
  - To accept new information for storage to be available later for use
  - read/write operation
- ROM: Read-Only Memory
  - Nonvolatile
  - The information inside can not be altered by writing
  - Programmable devices are specified by some hardware procedure

Random-Access Memory

Random-Access Memory

- Characteristics
  - The time it takes to transfer data to or from any desired location is always ***the same***.
  - The size of the RAM is 2^k x n bits. It has k address lines, n input data lines and n output data lines.
  - For a commodity RAM, 16<=k<=30, n=1, 4, 8, 16, 32 or 64.
- The communication between a memory and its environment is achieved through
  - Data input lines,
  - Data output lines,
  - Address selection lines
  - Control lines (read and write)



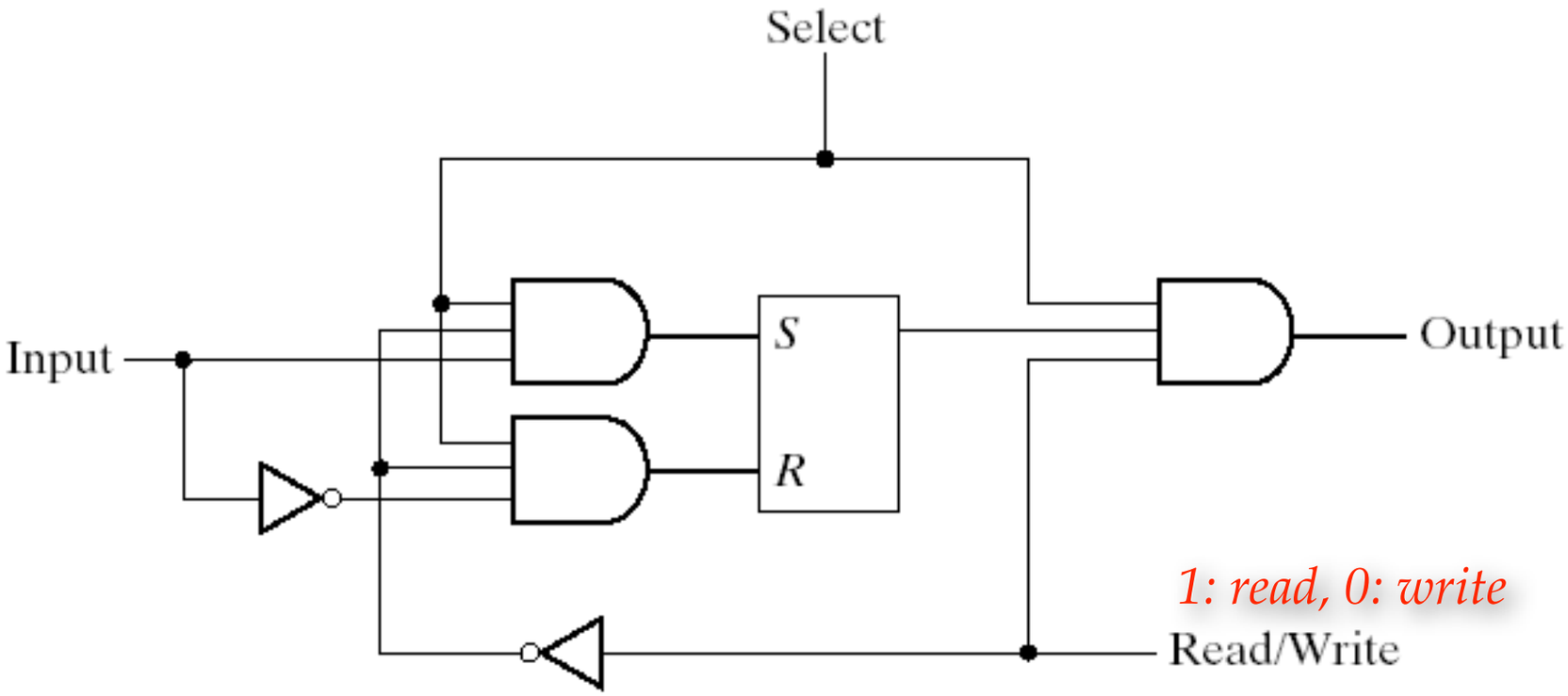
Memory Content Array

- A memory with *k*-bit address has 2^k data (depth)
  - (depth)x(width) memory
    - eg. 1K x 16 memory (1024x16 memory)

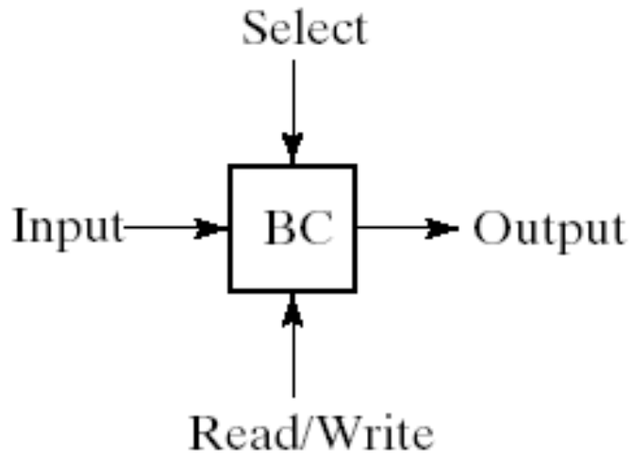
Memory address			Memory contest
Binary	decimal		
0000000000	0		1011010101011101
0000000001	1		1010101110001001
0000000010	2		0000110101000110
⋮	⋮		⋮
1111111101	1021		1001110100010100
1111111110	1022	2^k-2	0000110100011110
1111111111	1023	2^k-1	1101111000100101
k-bit address			*n bits*

Memory Cell

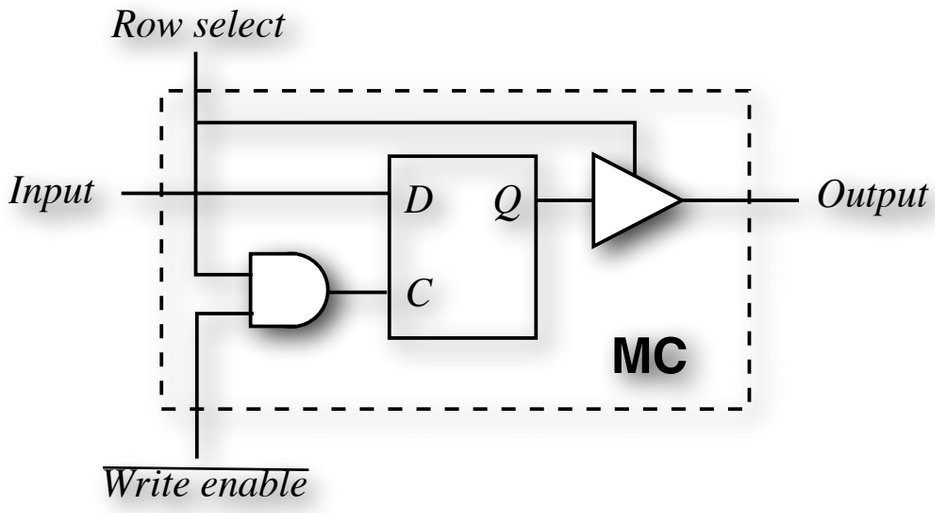
- A memory cell virtual model (*b*inary storage *c*ell)



(a) Logic diagram



(b) Block diagram



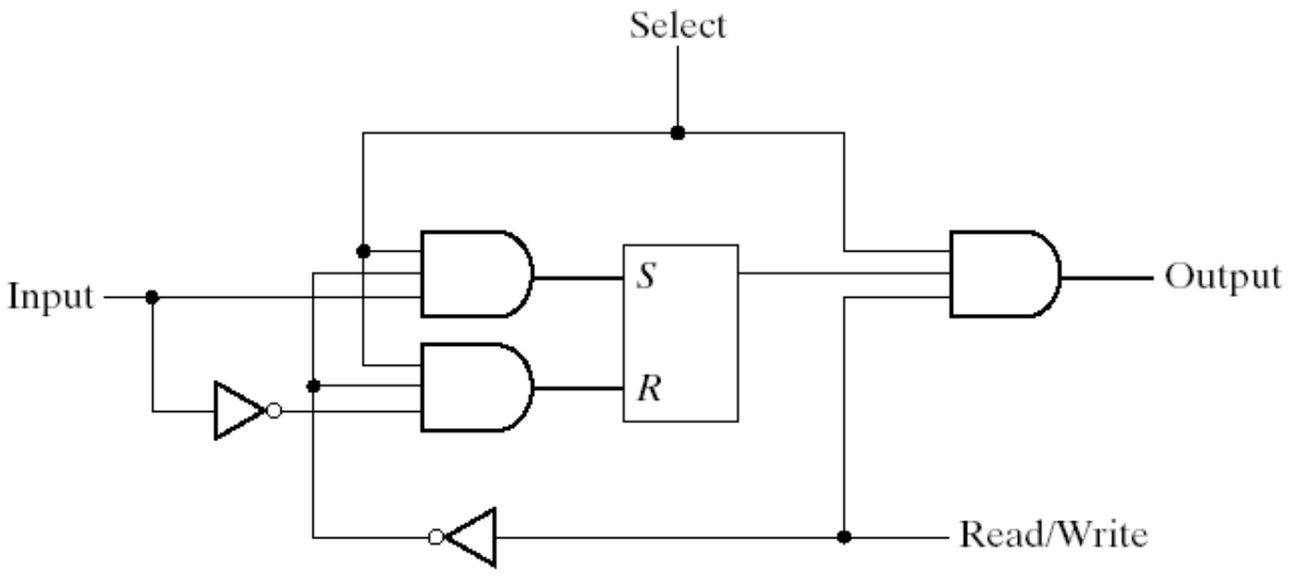
Static RAM vs. Dynamic RAM

- A memory cell (MC) can be considered as a clocked D latch with an AND gate and an output driver
  - For a ***static RAM (SRAM),*** MC is constructed by 6 transistors, using cross-coupled inverters to serve as a latch, and implementing the input AND gate and the output driver with one transistor each.
  - For a ***dynamic RAM (DRAM),*** MC is constructed by only 1 transistor
    - The latch is implemented by a capacitor.
    - It needs to be refreshed periodically.
    - It has high density (therefore low cost)

Write and Read Operation

- Write operation
  - Apply the binary address to the address lines
  - Apply the data bits to the data input lines
  - Activate the ***write*** input (0)
- Read operation
  - Apply the binary address to the address lines
  - Activate the ***read*** input (1)

Chip select CS	Read/$\overline{\text{Write}}$ R/$\overline{\text{W}}$	Memory operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

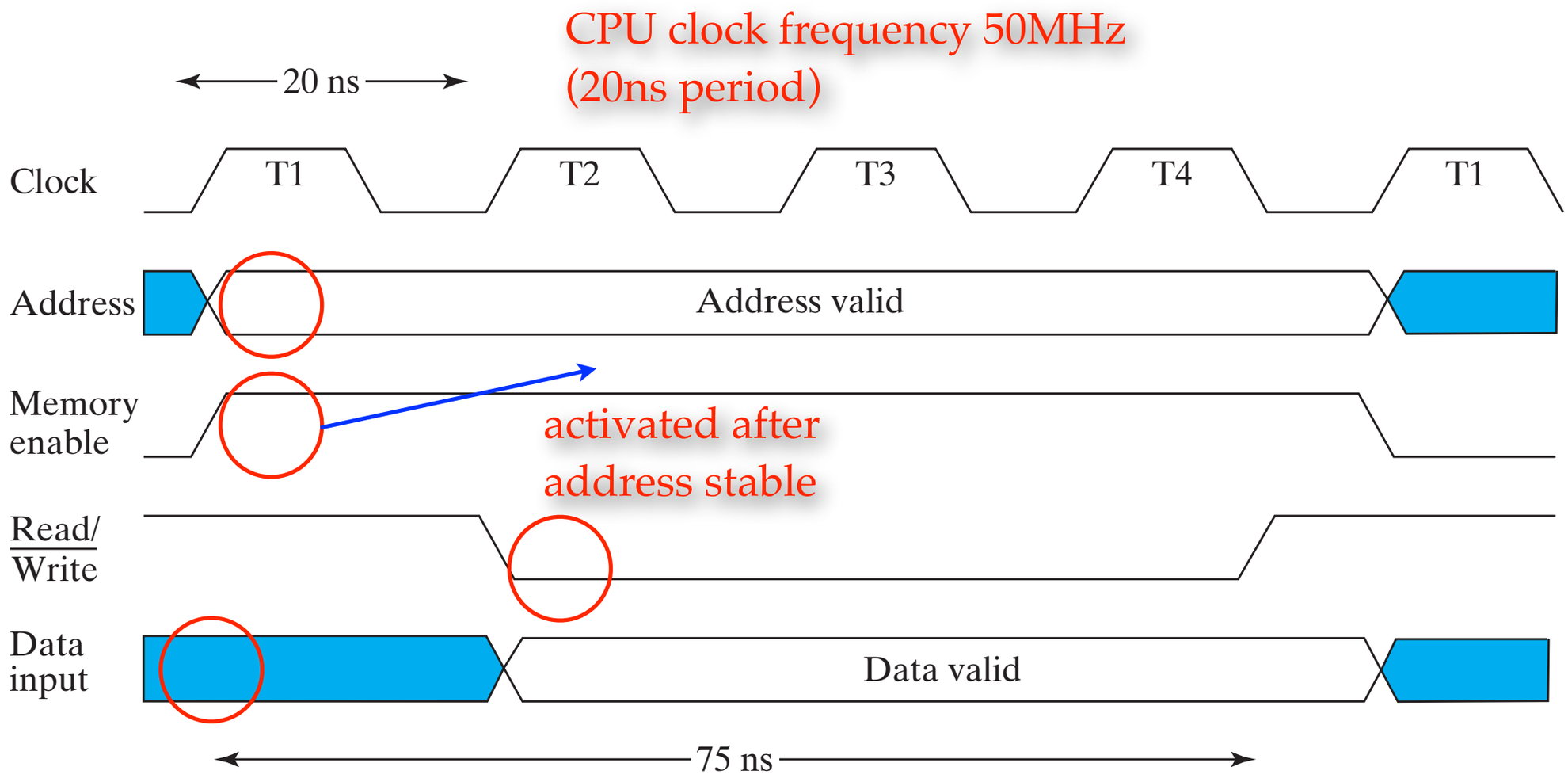


Timing Waveforms (1/3)

- The operation of the memory unit is controlled by an external device such as a CPU
- The ***access time*** is the time required to select a word and read it
- The ***write cycle time*** is the time required to complete a write operation
- Read and write operations must be controlled by CPU and be synchronized with an external clock.

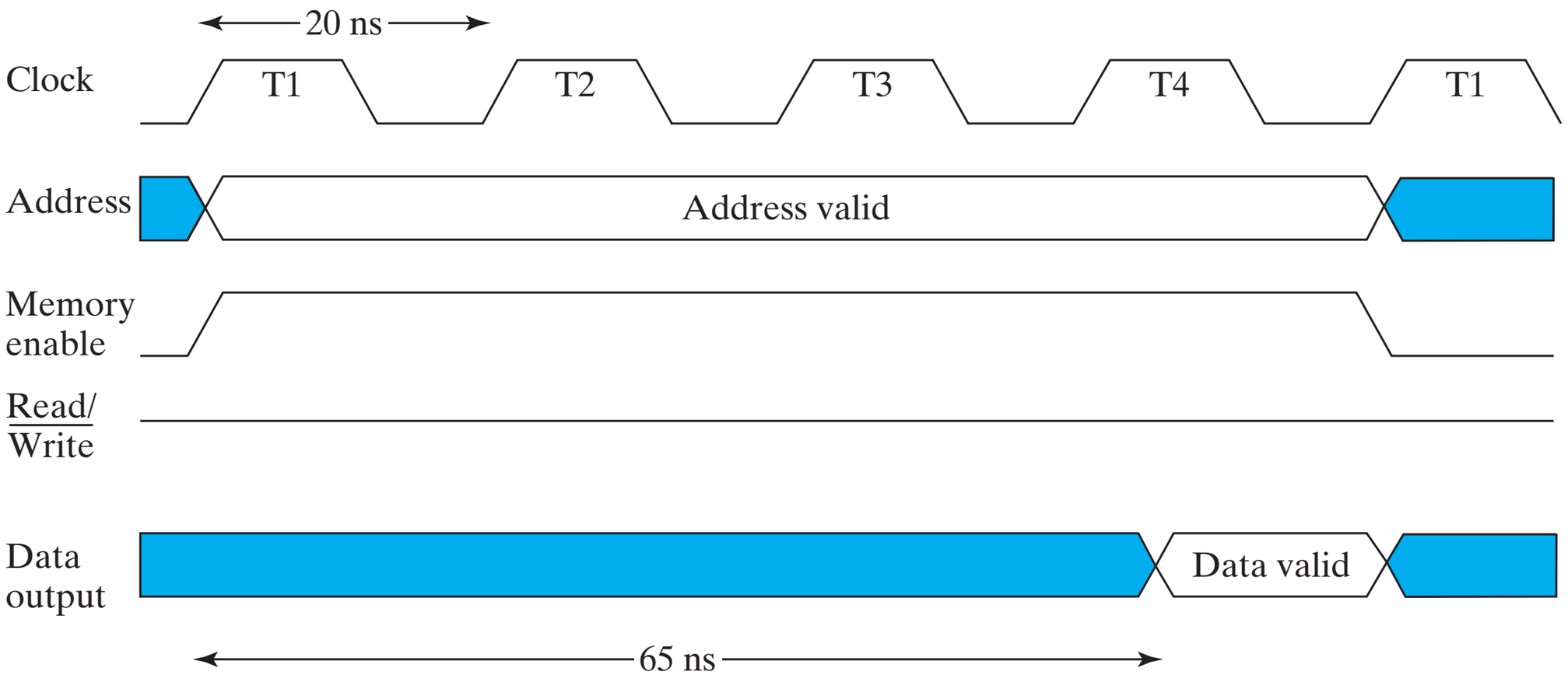
Timing Waveforms (2/3)

- A write cycle of *write cycle time* 75 ns



Timing Waveforms (3/3)

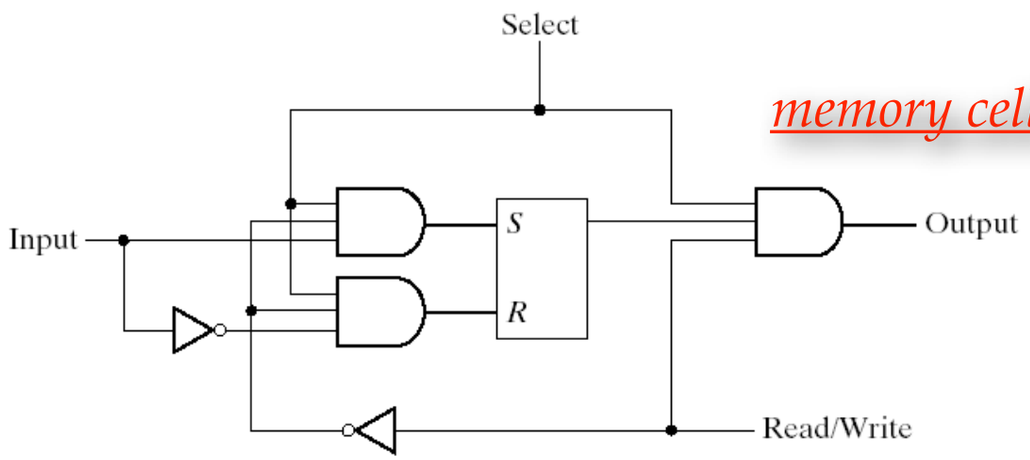
- A read cycle of *access time* 65 ns



Memory Decoding

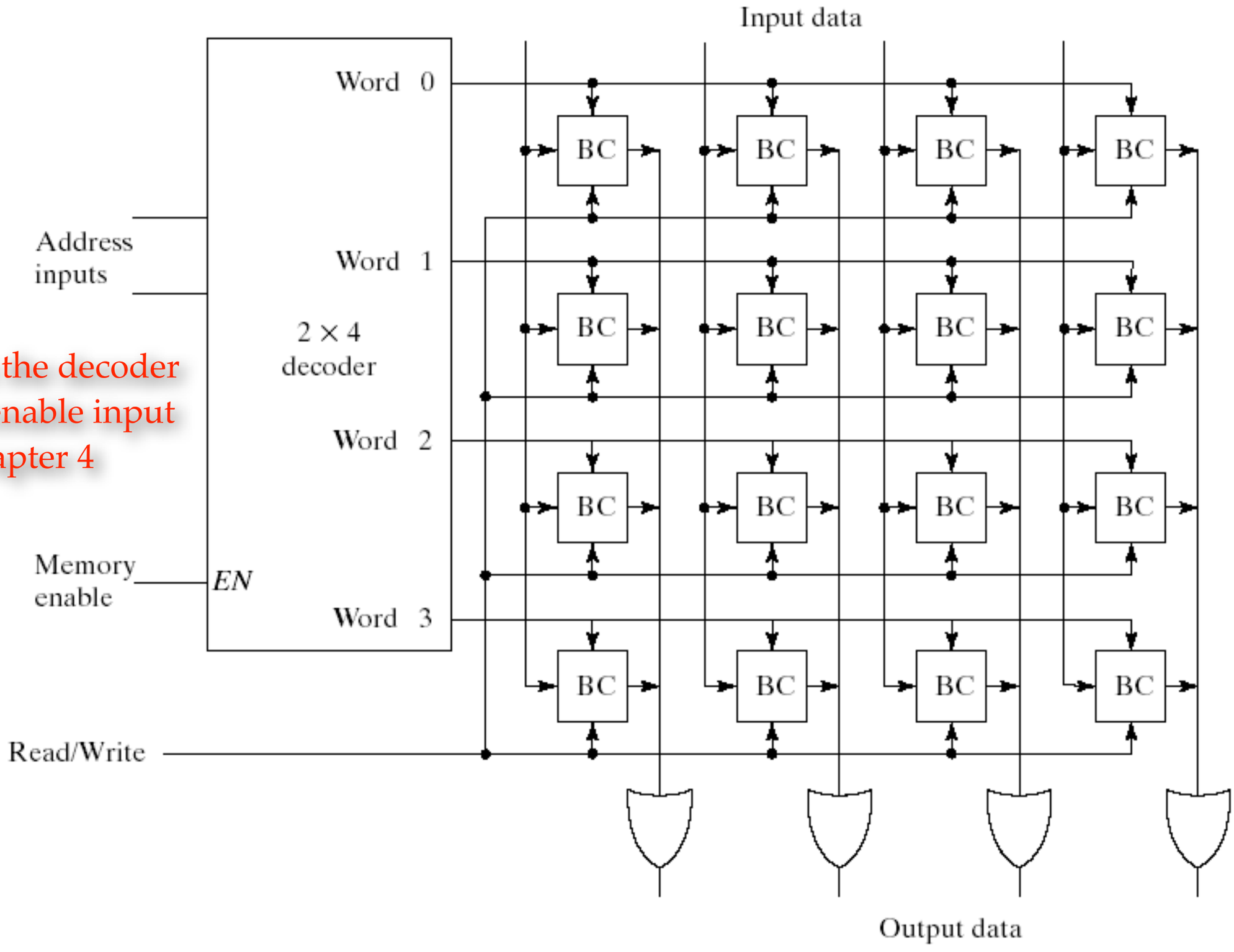
Memory Unit and Internal Construction

- A memory unit has two parts
  - The storage components (memory cell)
  - The decoding circuits to select the memory word
- A RAM of *m* (2^k) words and *n* bits per word
  - *m*n* binary storage cells
  - Decoding circuits to select individual words
    - k-to-2^k decoder: address input to word line
    - Read/Write control
    - Input data/Output data



memory cell (BC)

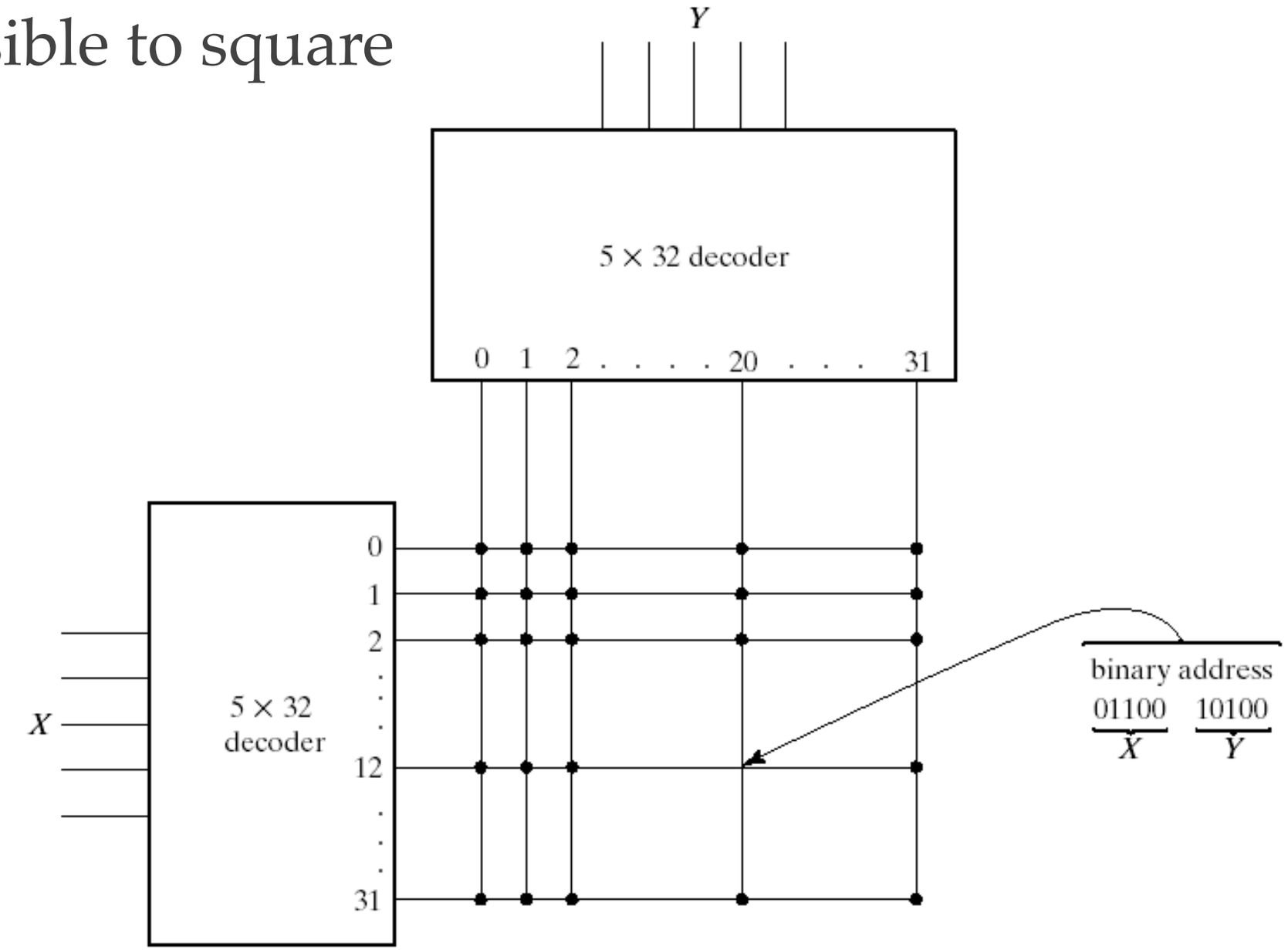
4x4 RAM



check the decoder with enable input in Chapter 4

Coincident Decoding (1/2)

- A *two-dimensional selection* scheme
  - To reduce the complexity of the decoding circuits
  - To arrange the memory cells in an array that is close as possible to square



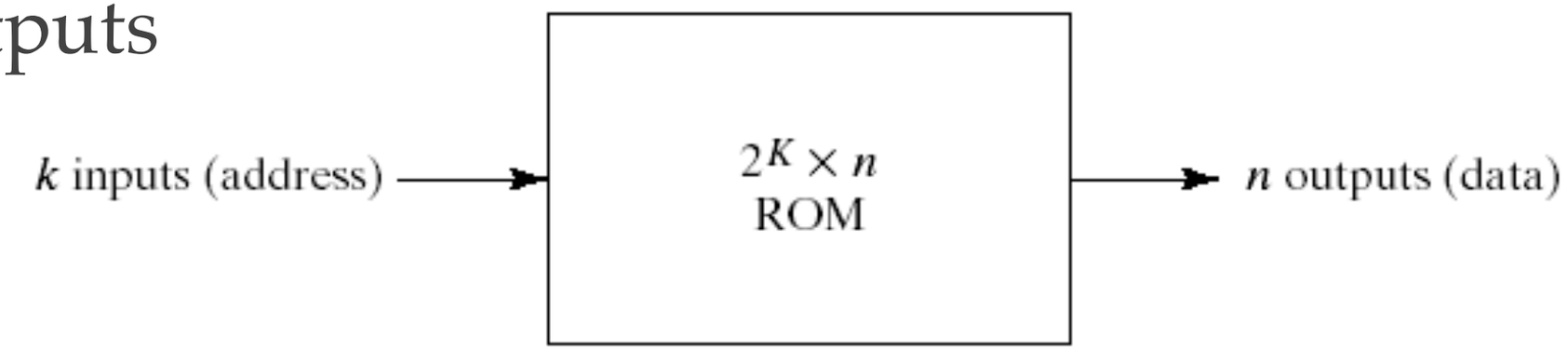
Coincident Decoding (2/2)

- Example: 1k-word memory
  - A 10-to-1024 decoder
    - 1024 AND gates with 10 inputs per gate
  - Two 5-to-32 decoders
    - 2*(32 AND gates with 5 inputs per gate)
    - Each word in the memory is selected by the coincidence between 1 of 32 rows and 1 of 32 columns for a total 1024 words
    - Two dimensional decoding structure reduces the ***circuit complexity*** and the ***cycle time*** of the memory

Read-Only Memory

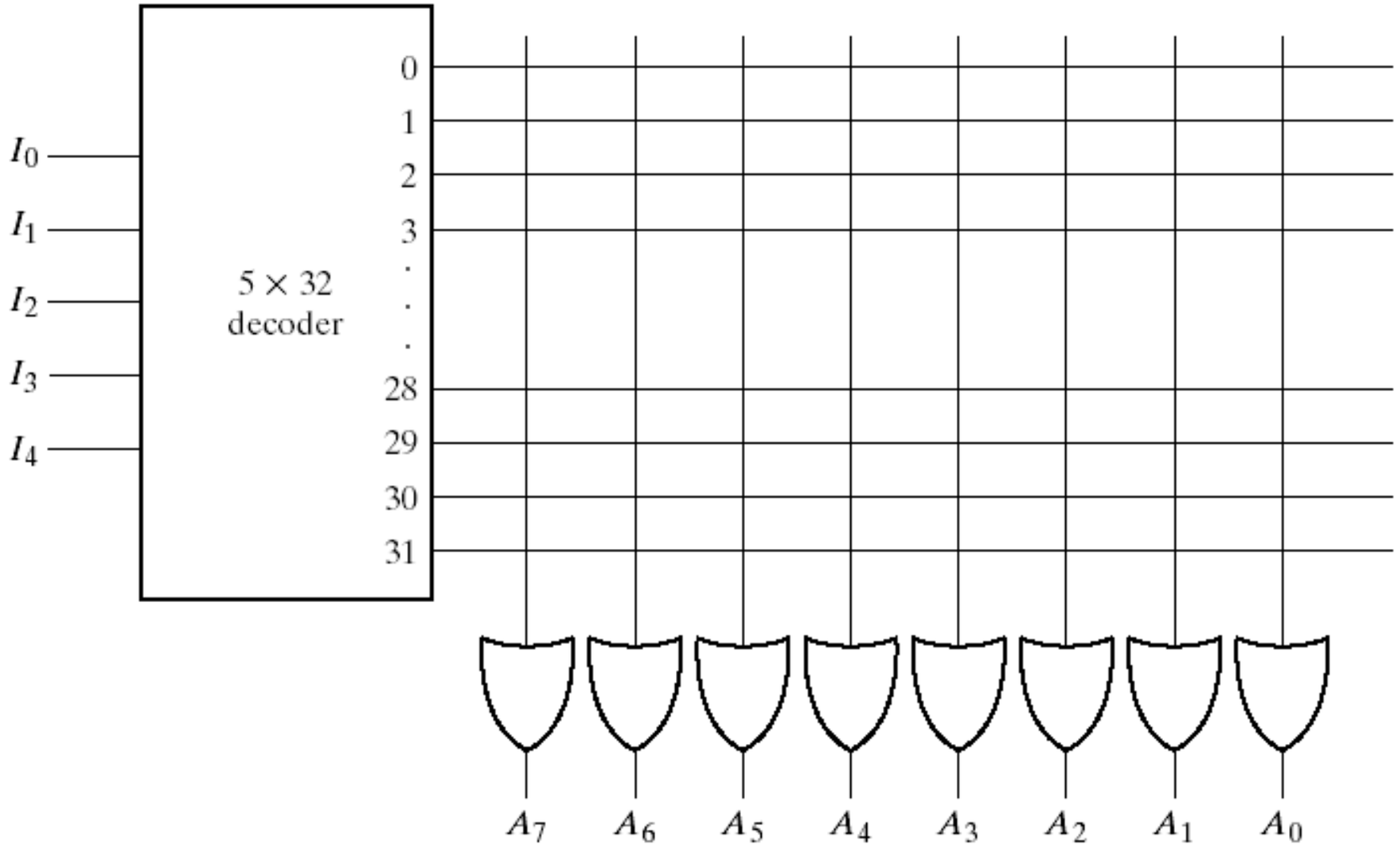
Read-Only Memory (ROM)

- ROM stores permanent binary information
- Once the pattern is established in the ROM, it stays within the unit, no matter power is on or off
- 2^k x n ROM
  - k address input lines
  - enable input(s)
  - three-state outputs



Example

- 32 x 8 ROM (2^5 x 8)
  - 5-to-32 decoder (k=5)
  - 32(2^5) outputs of the decoder are connected to each of the eight OR gates, which have 32(2^5) inputs
  - 32 x 8 internal programmable connections



ROM Programming (1/2)

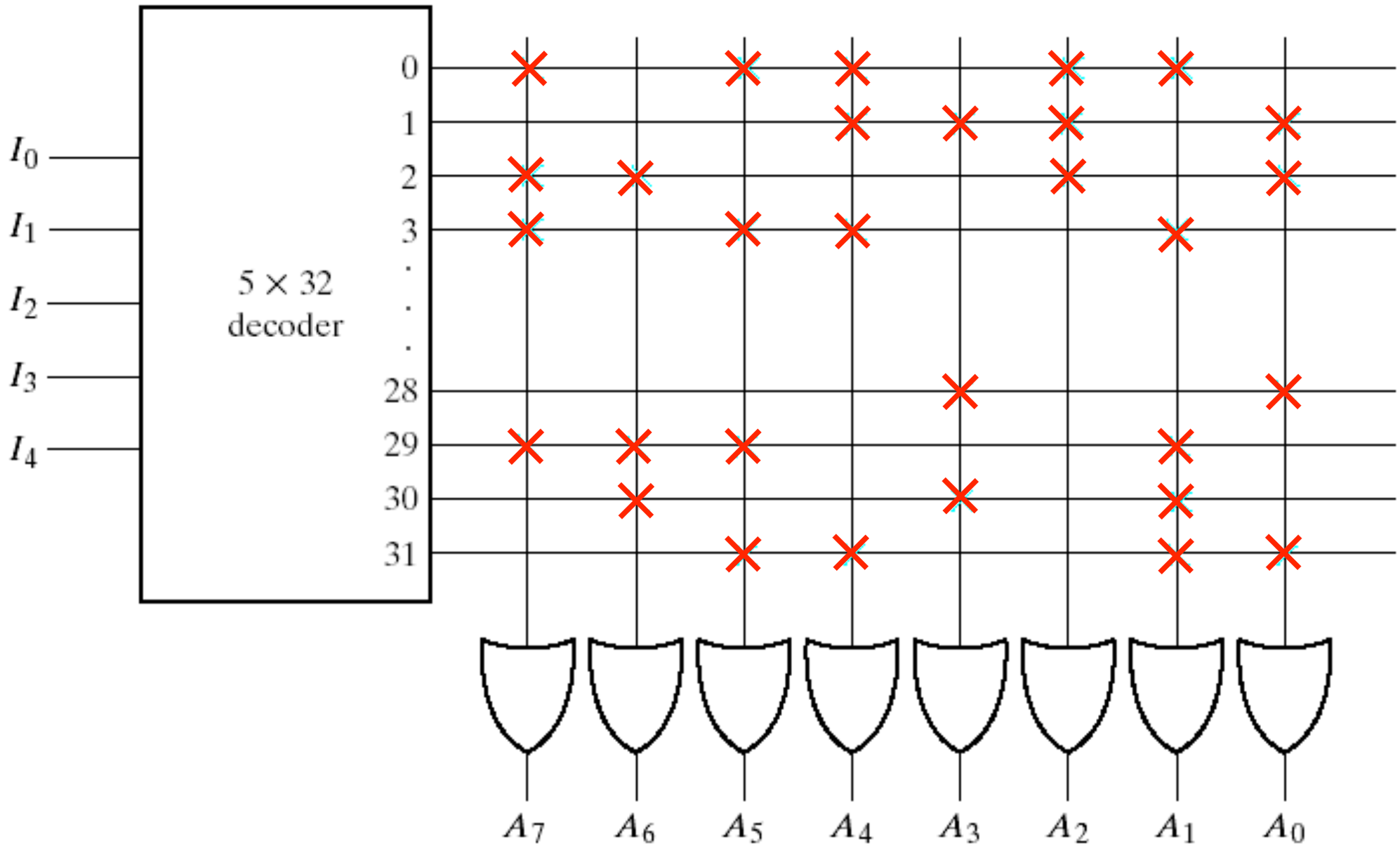
- Programmable interconnections
- close [1]: connected to high voltage
- open [0]: ground left
- A fuse that can be blown by applying a high voltage pulse

ROM Truth Table (Partial)

Inputs					Outputs							
I4	I3	I2	I1	I0	A7	A6	A5	A4	A3	A2	A1	A0
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

ROM Programming (2/2)

- ROM programming according to ROM table using fusing



Combinational Circuit Implementation

- The internal operation of ROM can be interpreted in two ways
  - A memory contains a fixed pattern of ***stored words***
  - A unit that implements a combinational circuit as ***sum of minterms*** in Section 4-9.
- ROM: a decoder + OR gates
  - a Boolean function = sum of minterms
  - Ex. $A_7(I_4, I_3, I_2, I_1, I_0) = \sum(0, 2, 3, \ldots, 29)$ (Fig.7.11)
  - For an n-input, m-output combinational circuit
    - 2^n x m ROM

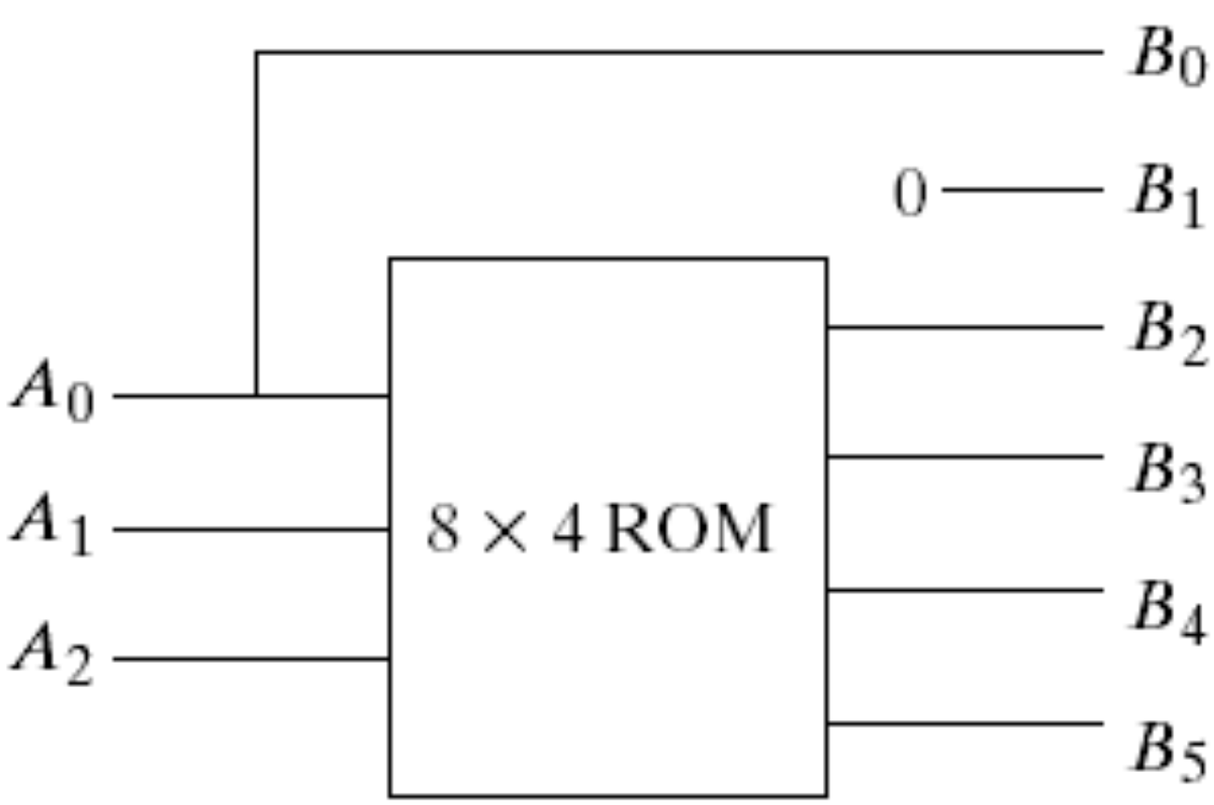
Example: ROM Implementation (1/2)

- A combinational circuit has 3 inputs, 6 outputs and the truth table, construct the circuit

Inputs			Outputs						
A_1	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

$B_0=A_0$, $B_1=0$, leave only 8x4 ROM

Example: ROM Implementation (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

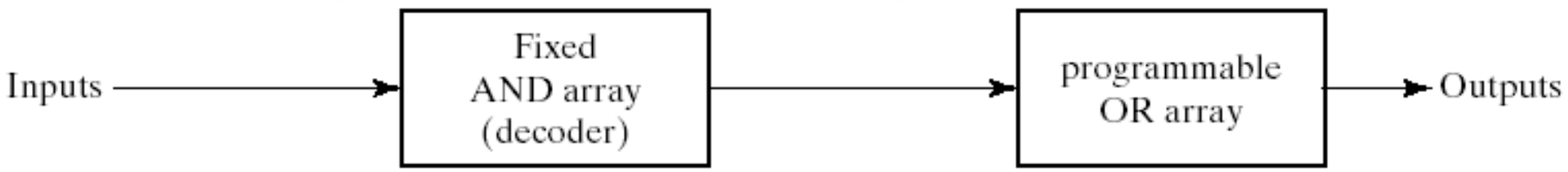
(b) ROM truth table

Types of ROM

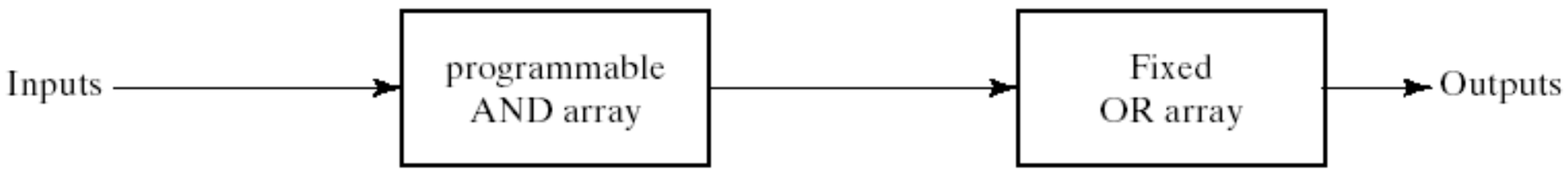
- Mask programmable ROM
  - Data is programmed by IC manufacturers
  - Economical if a large quantity of the same ROM
- PROM: Programmable ROM
  - Fuses in the ROM are blown by high-voltage pulse
  - Universal programmer can program PROM on time
- EPROM: erasable PROM
  - Floating gate
  - The data in the ROM can be erased by UV light
- EEPROM: electrically erasable PROM
  - Longer time is needed to write
  - Flash ROM
  - Limited times of write operations

Combinational PLDs

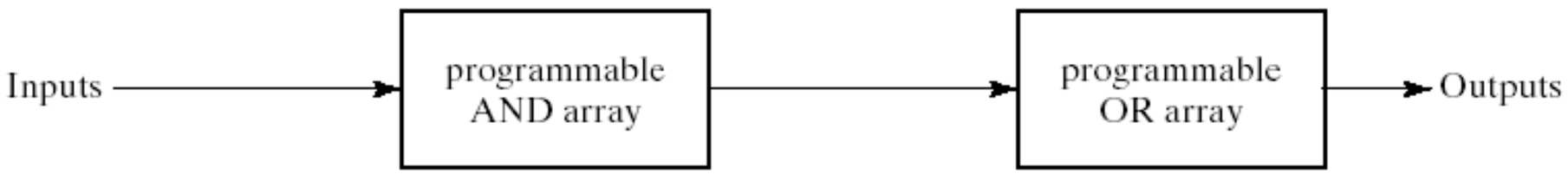
- PLD: Programmable logic device
- Programmable two-level logic
  - sum of products
  - an AND array and an OR array



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL)



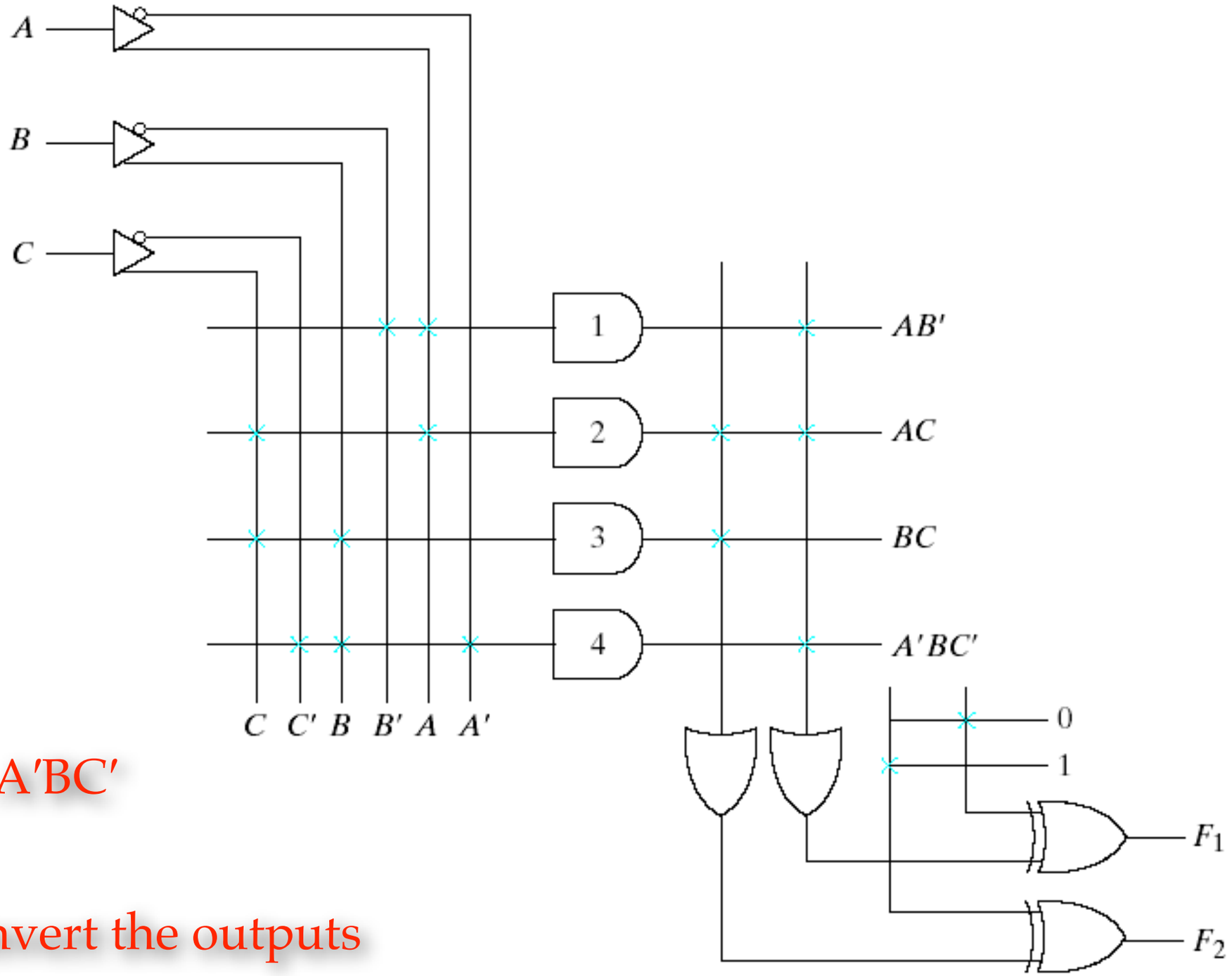
(c) Programmable logic array (PLA)

Programmable Logic Array

Programmable Logic Array (PLA)

- PLA has two main parts
  - An array of programmable AND gates
    - Can generate any product terms of the inputs
  - An array of programmable OR gates
    - Can generate the sums of the products
- PLA is more flexible than PROM
  - PLA uses any combination of products while PROM uses sum of minterms
- PLA uses less circuits than PROM
  - Only the needed product terms are generated in PLA while all minterms must be generated in PROM

Example (1/2)



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

$F_2 = (AC + BC)'$

XOR gates can invert the outputs

Example (2/2)

- Specify the fuse map
  - 1st col: list the product terms numerically
  - 2nd col: specify the requires paths between inputs and AND gates
  - 3rd col: specify the required paths between AND gates and OR gates
  - (T)(C) stand for true of complement for programming XOR

PLA Programming Table

		Inputs			Outputs (T)	Outputs (C)
Product Term		A	B	C	F_1	F_2
AB′	1	1	0	–	1	–
AC	2	1	–	1	1	1
BC	3	–	1	1	–	1
A′BC′	4	0	1	0	1	–

Characteristics of PLA

- The size of a PLA is determined by
  - the number of inputs
  - the number of product terms (AND gates)
  - the number of outputs (OR gates)
- When implementing with a PLA
  - Must reduce the number of distinct product terms before implementation
  - The number of terms in a product is not important

Example 7.2 (1/2)

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

A \ BC	00	01	11	10
0	1	1	0	1
1	1	0	0	0

$$F_1 = A'B' + A'C' + B'C'$$

$$F_1 = (AB + AC + BC)'$$

A \ BC	00	01	11	10
0	1	0	0	0
1	0	1	1	1

$$F_2 = AB + AC + A'B'C'$$

$$F_2 = (A'C + A'B + AB'C')'$$

Example 7.2 (2/2)

$F_1 = (AB + AC + BC)'$

$F_2 = AB + AC + A'B'C'$

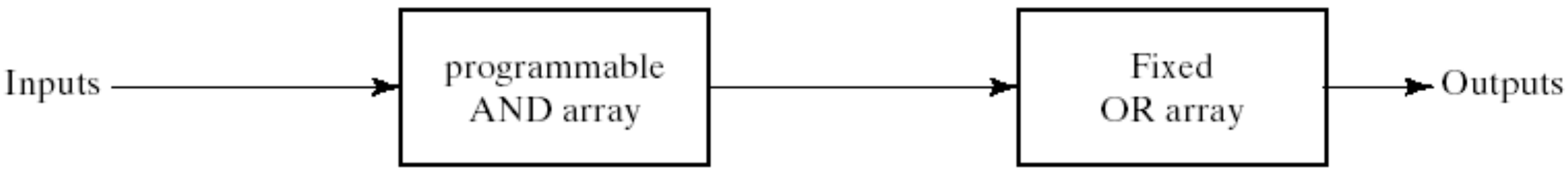
PLA programming table

	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

Programmable Array Logic

Programmable Array Logic (PAL)

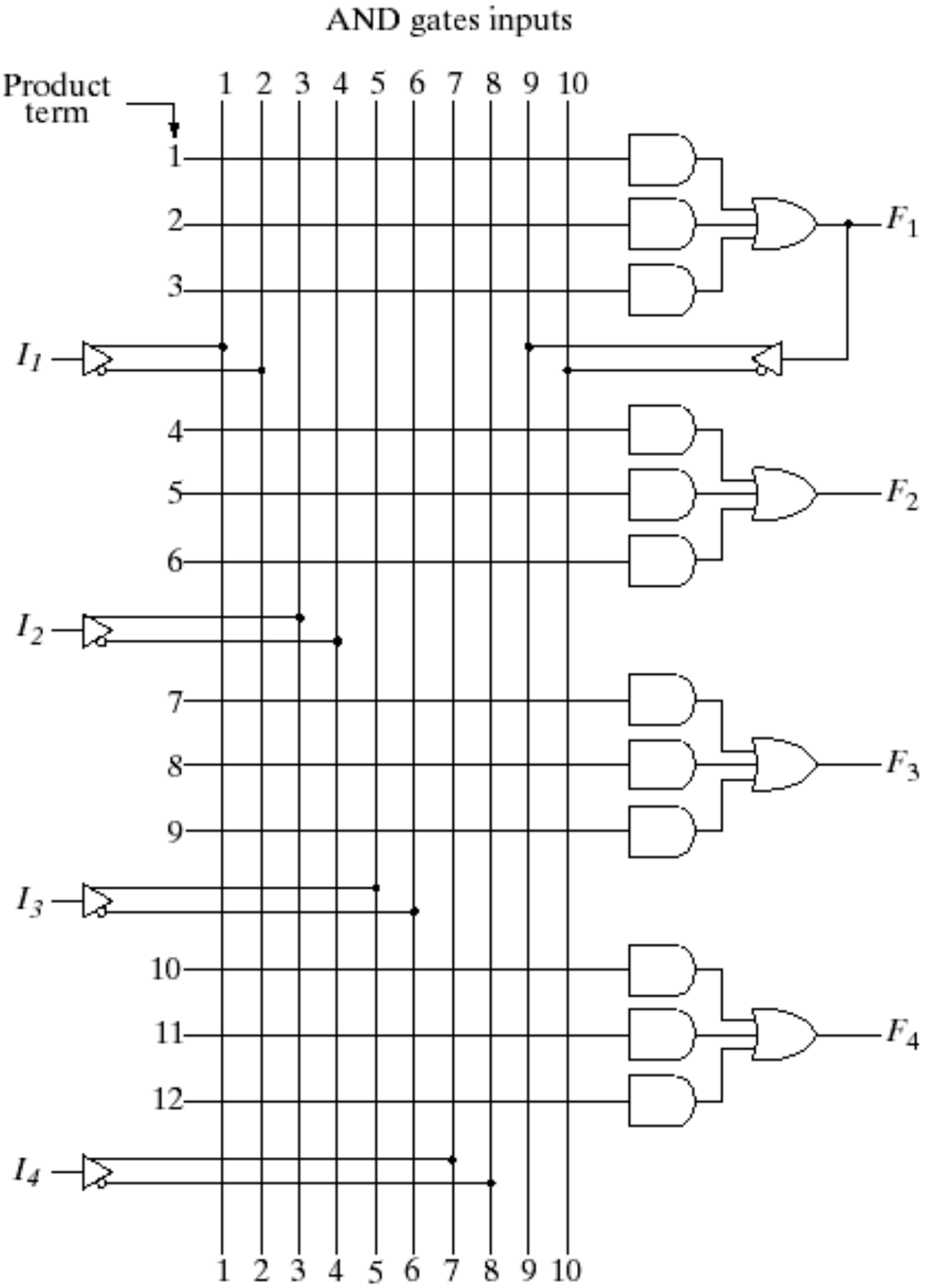
- A programmable AND array and a fixed OR array
  - The PAL is easier to program, but is not as flexible as the PLA



(b) Programmable array logic (PAL)

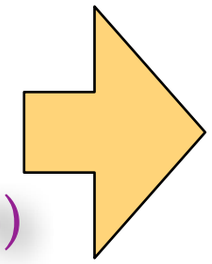
PAL Structure

- Example
  - product term cannot be shared



Example (1/2)

- w(A,B,C,D) = Σ(2,12,13)
- x(A,B,C,D) = Σ(7,8,9,10,11,12,13,14)
- y(A,B,C,D) = Σ(0,2,3,4,5,6,7,8,10,11,15)
- z(A,B,C,D) = Σ(1,2,8,12,13)

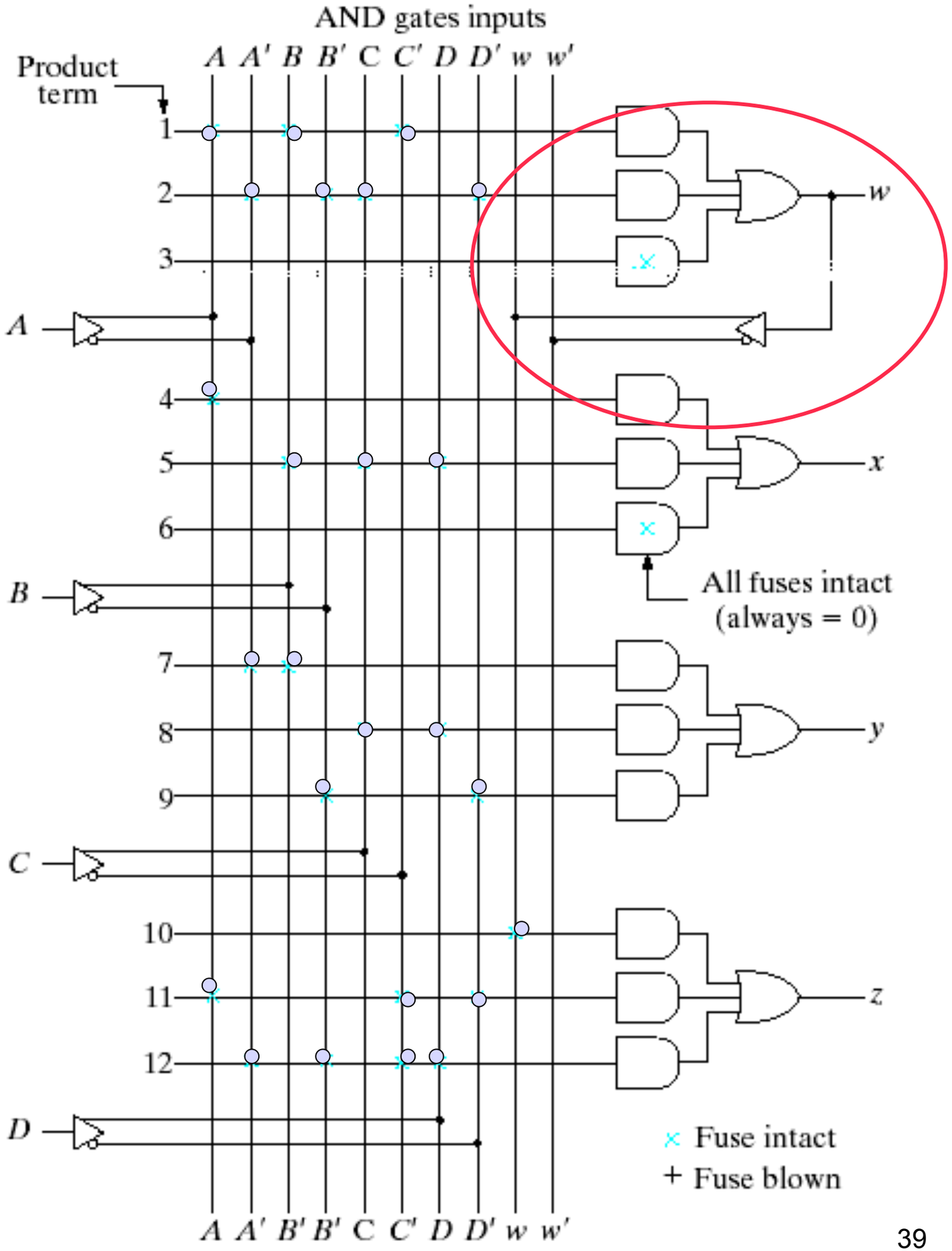


- *w* = ABC′ + A′B′CD′
- *x* = A + BCD
- *y* = A′B + CD + B′D′
- *z* = ABC′ + A′B′CD′ + AC′D′ + A′B′C′D
  = *w* + AC′D′ + A′B′C′D

Product Term	AND Inputs A	B	C	D	W	Outputs
1	1	1	0	–	–	$w = ABC'$
2	0	0	1	0	–	$+ A'B'CD'$
3	–	–	–	–	–	
4	1	–	–	–	–	$x = A$
5	–	1	1	1	–	$+ BCD$
6	–	–	–	–	–	
7	0	1	–	–	–	$y = A'B$
8	–	–	1	1	–	$+ CD$
9	–	0	–	0	–	$+ B'D'$
10	–	–	–	–	1	$z = w$
11	1	–	0	0	–	$+ AC'D'$
12	0	0	0	1	–	$+ A'B'C'D$

Example (2/2)

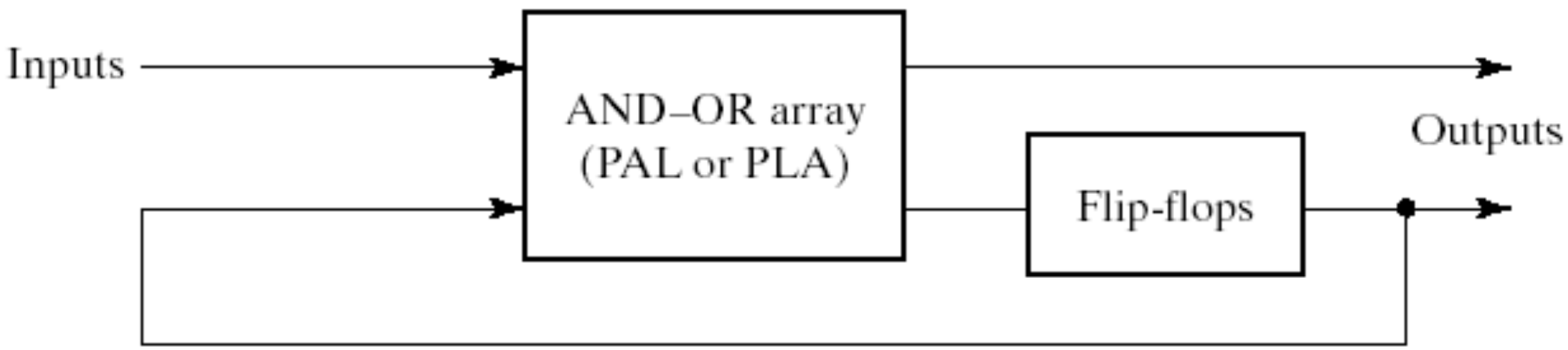
- Programmed PLA
  - $w = ABC' + A'B'CD'$
  - $x = A + BCD$
  - $y = A'B + CD + B'D'$
  - $z = w + AC'D' + A'B'C'D$



Sequential Programmable Devices

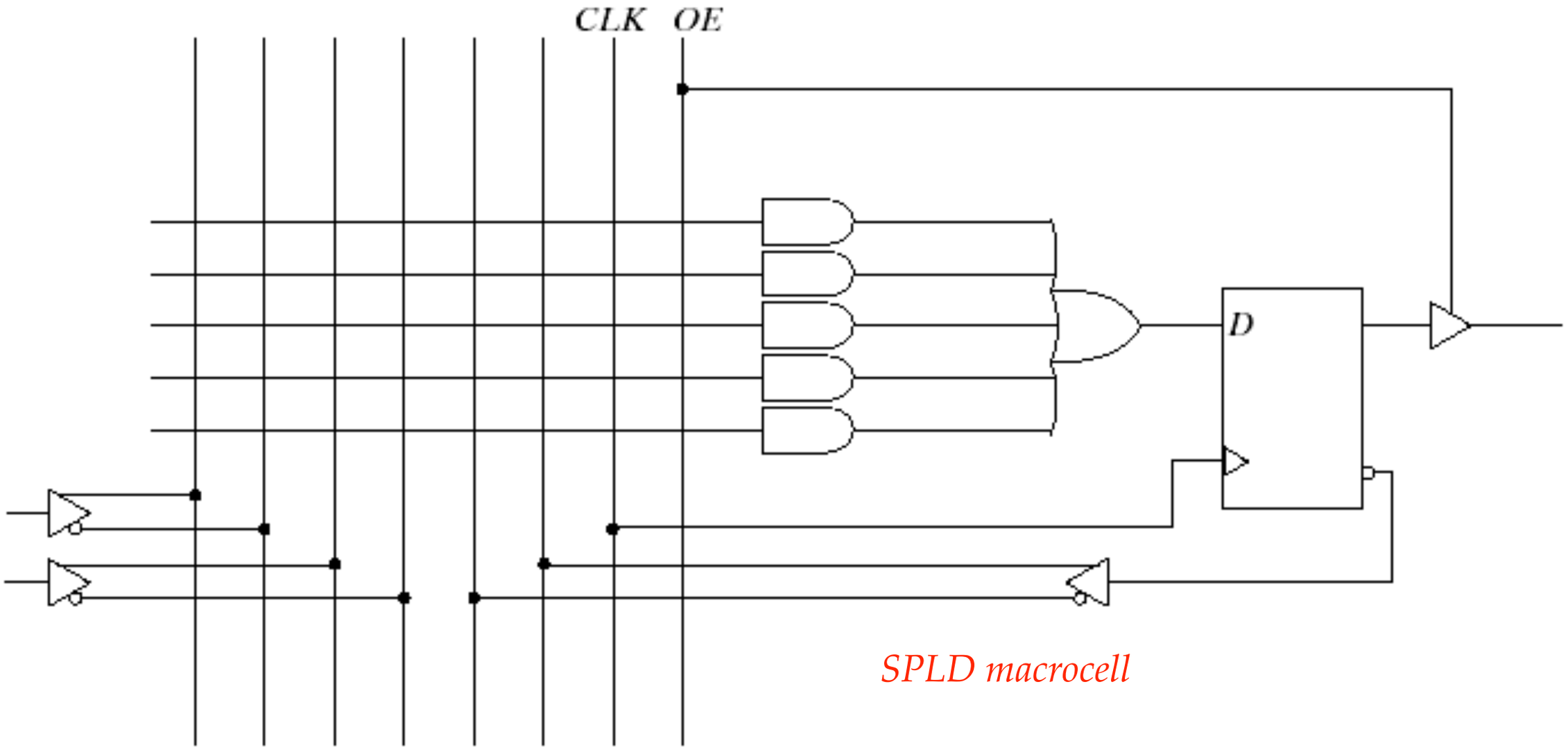
Sequential Programmable Devices

- Sequential programmable devices include both programmable gates and flip-flop
- Three major types os sequential programmable devices
  - SPLD: Sequential (or simple) PLD
  - CPLD: Complex PLD
  - FPGA: Field Programmable Gate Array



SPLD

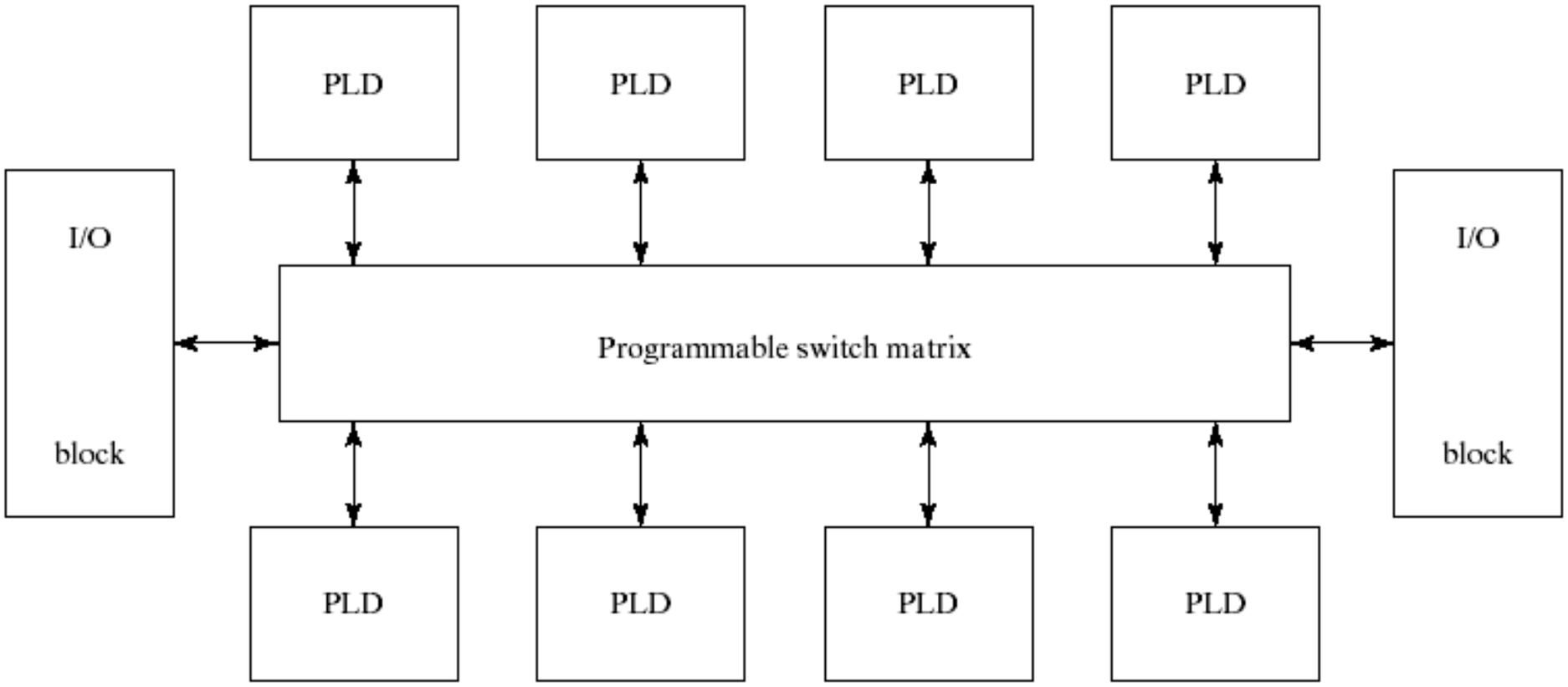
- A PAL or PLA is modified by including a number of flip-flops



SPLD macrocell

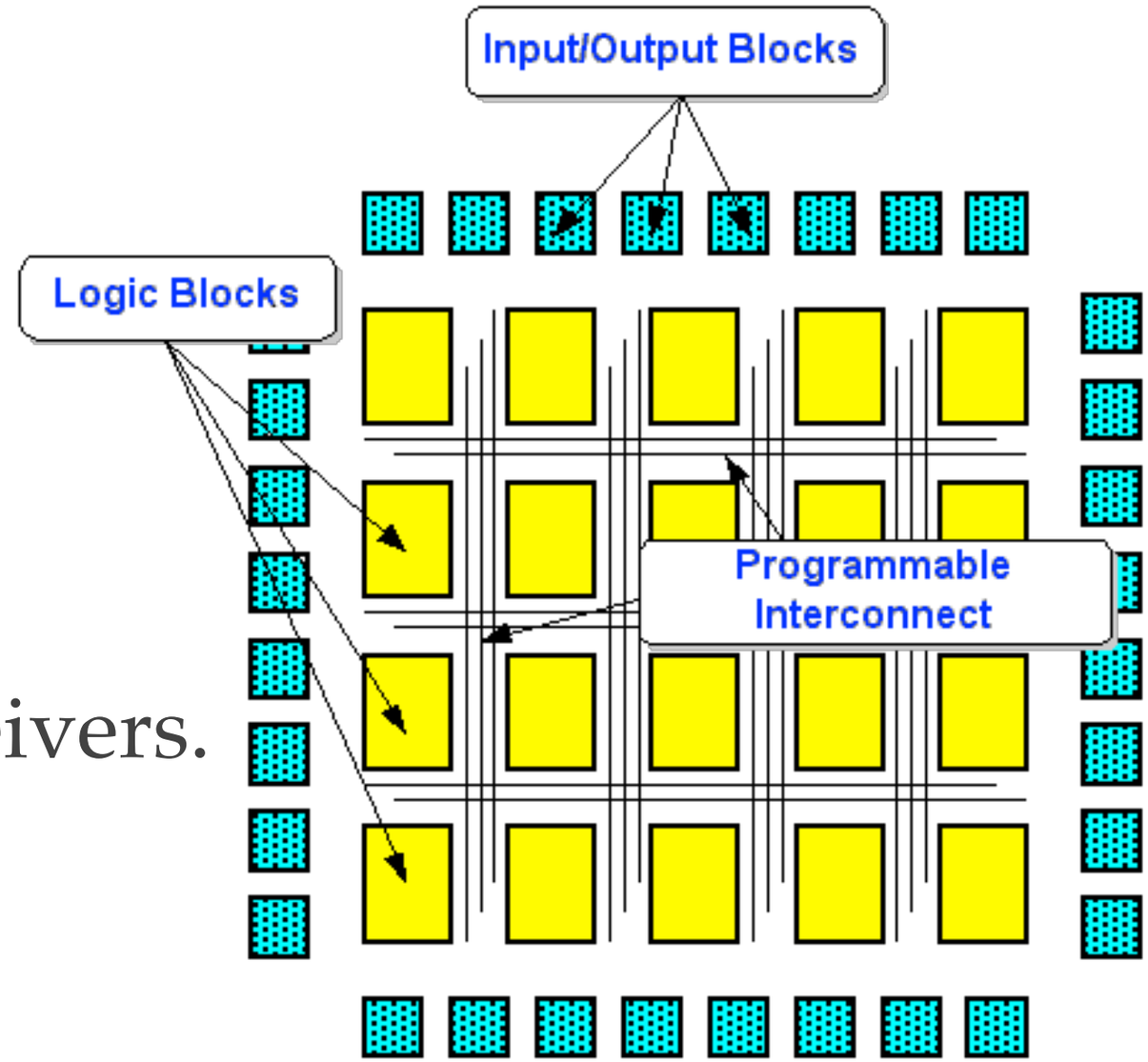
CPLD

- Complex PLD
  - Put a lot of PLDs on a chip
  - Adds wires between PLDs whose connections can be programmed
  - Use fuse/EEPROM technology

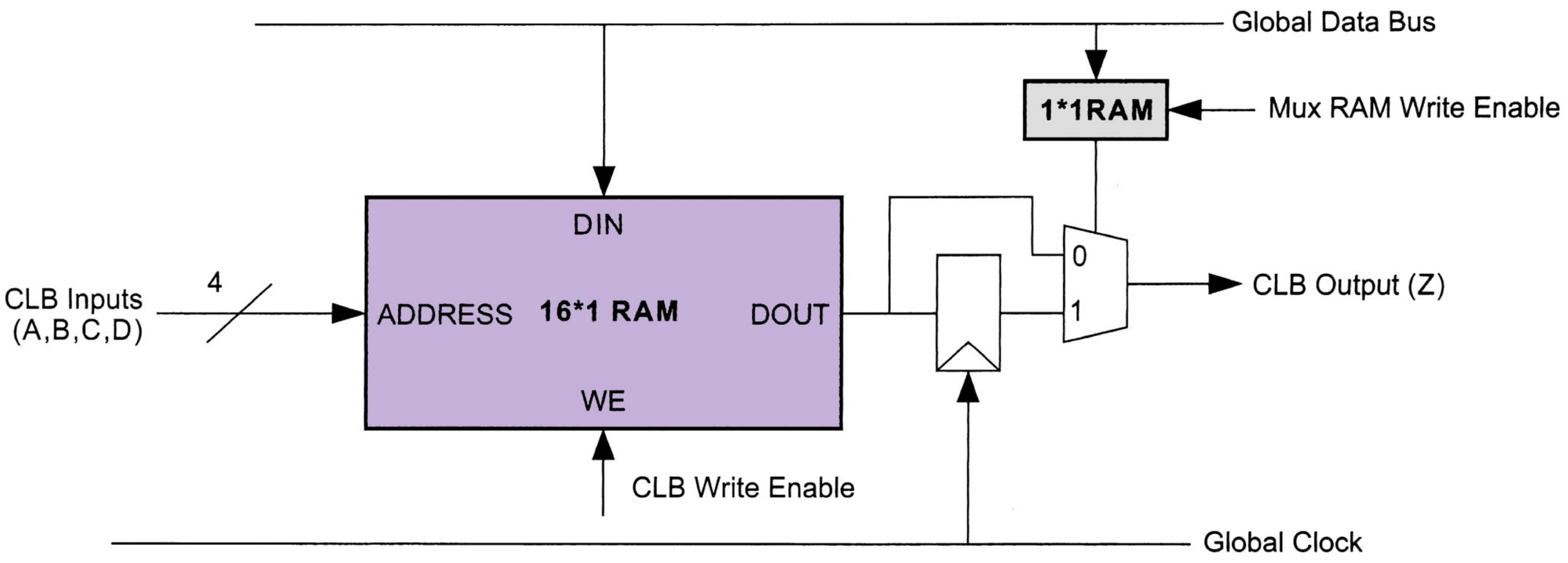


FPGA

- What is in the array? All sorts of stuff…
  - I/O Cells.
  - Logic Cells.
  - Memories.
  - Microprocessors.
  - Clock Management.
  - High Speed I/O Transceivers.
  - Programmable routing.



Simple FPGA Logic Cell (1/2)

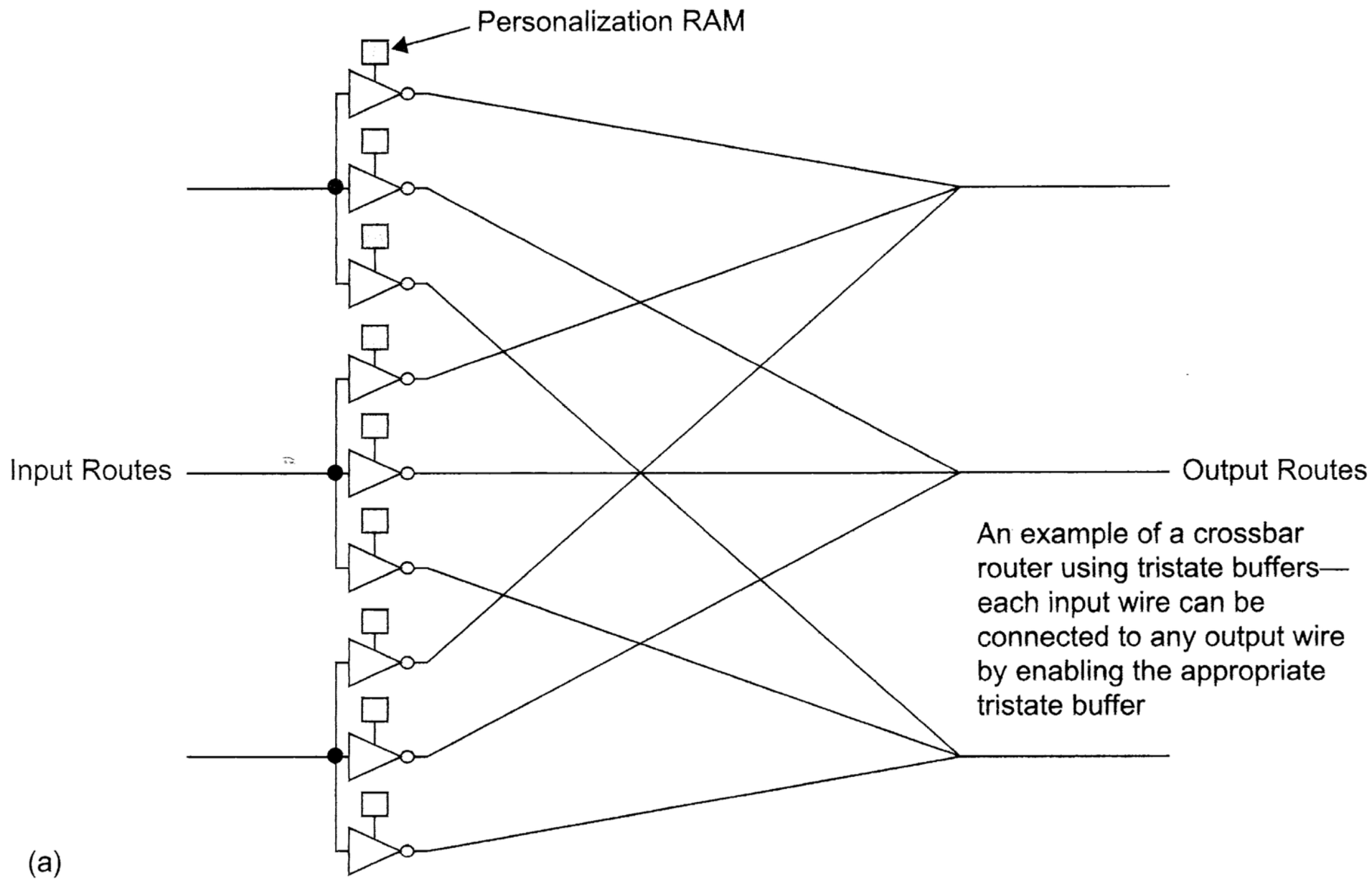


Simple FPGA Logic Cell (2/2)

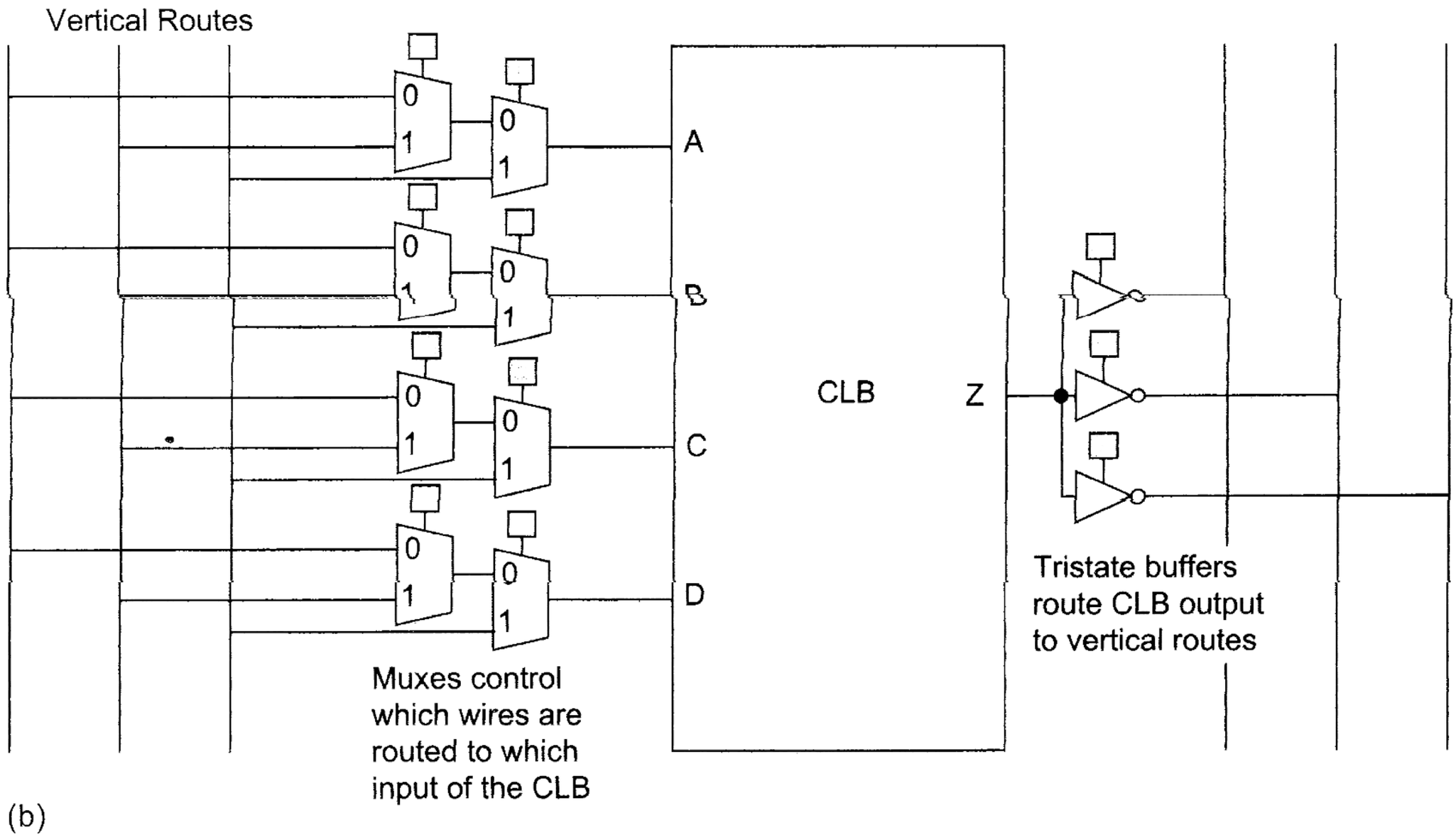
Table 8.2 RAM CLB functions

Address	*ABCD*	*A • B • C • D*	~A	SUM(A,B,C)
0	0000	0	1	0
1	1000	0	0	1
2	0100	0	1	1
3	1100	0	0	0
4	0010	0	1	1
5	1010	0	0	0
6	0110	0	1	0
7	1110	0	0	1
8	0001	0	1	0
9	1001	0	0	1
10	0101	0	1	1
11	1101	0	0	0
12	0011	0	1	1
13	1011	0	0	0
14	0111	0	1	0
15	1111	1	0	1

Simple FPGA Routing Cell (1/2)



Simple FPGA Routing Cell (2/2)



(b)

Routed FPGA