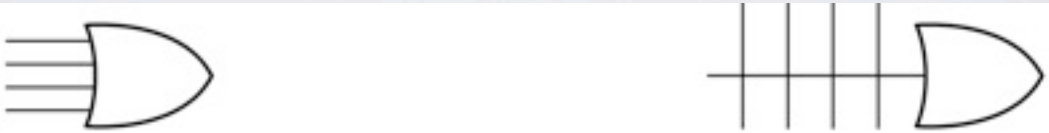
PROGRAMMABLE LOGIC AND MEMORIES

INEL 4205 - Logic Circuits - Spring 2012



(a) Conventional symbol

(b) Array logic symbol

Fig. 7-1 Conventional and Array Logic Diagrams for OR Gate

- Two types of memory
  - ROM: read-only memory: EPROM or FLASH / EEPROM
  - RAM: read/write memory: SRAM (static) or DRAM (dynamic).
- Memory device address inputs
  - select a location on (inside) the memory.
  - number from A_0 to A_{N-1} (2^N = # cells)

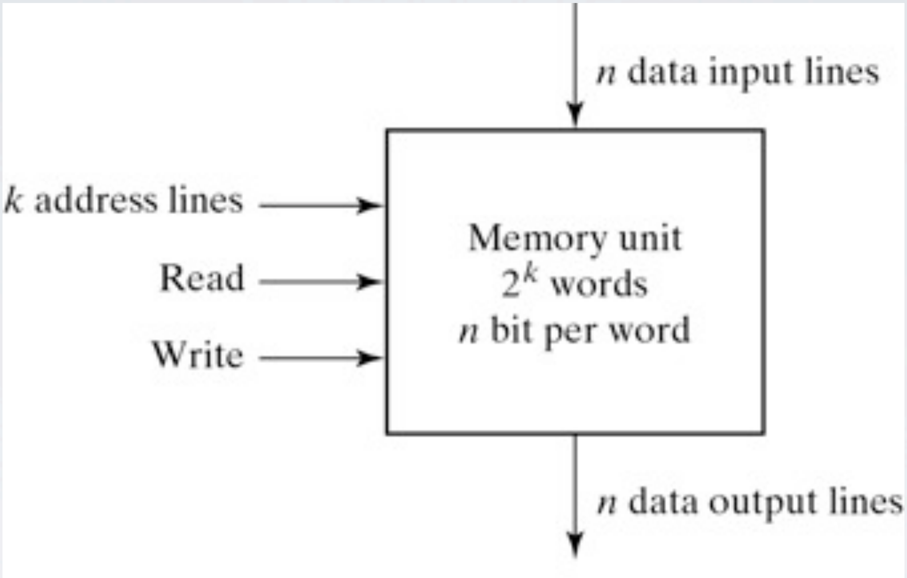
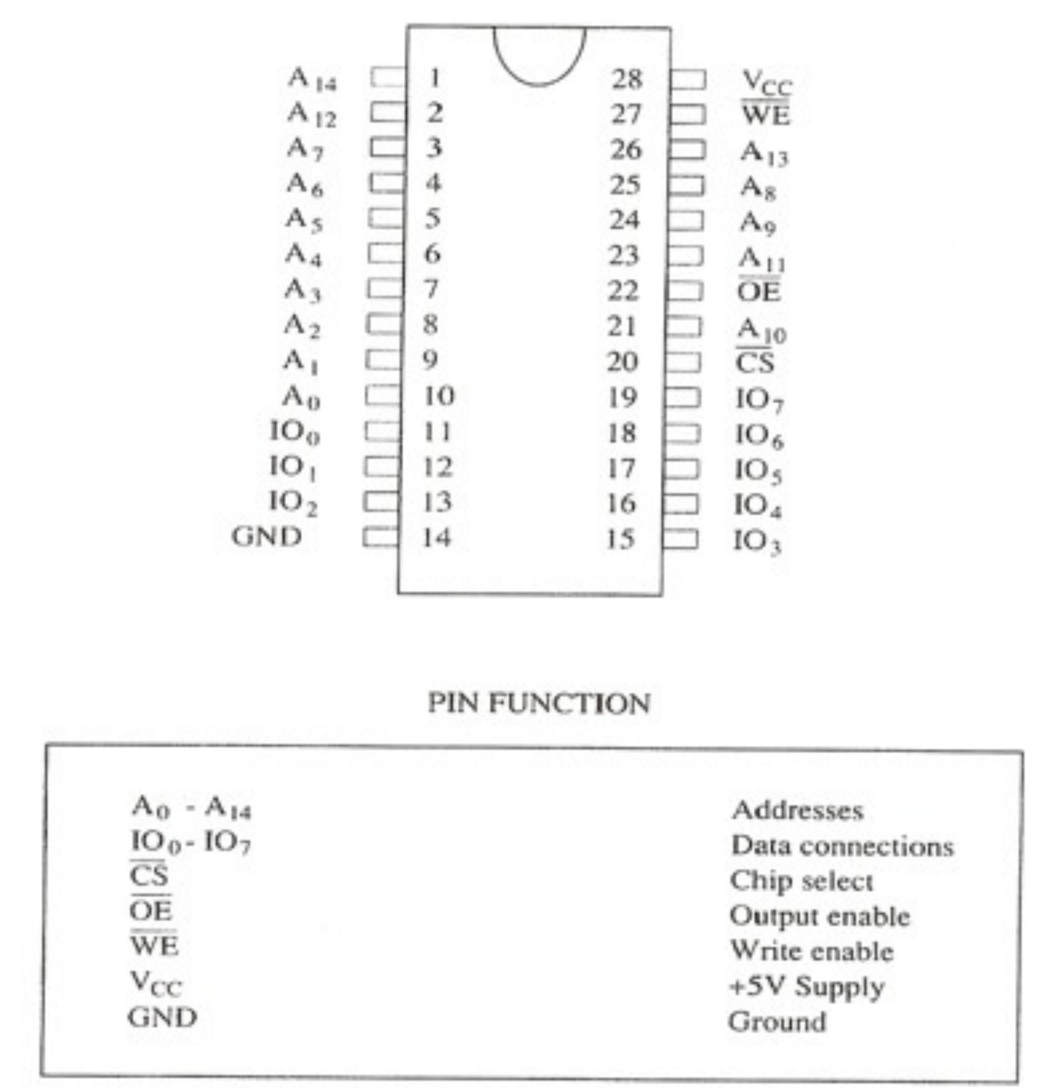


Fig. 7-2 Block Diagram of a Memory Unit

- Data pins
  - 8-bit in width: D0 to D7
  - eight devices form a 64-bit wide memory
  - D0: least significant bit
- Control inputs
  - Output enable (#OE): cause a read
  - Write enable (#WE): causes a write
  - Chip select (#CS)

TMS4016 2K x 8 SRAM Pinout

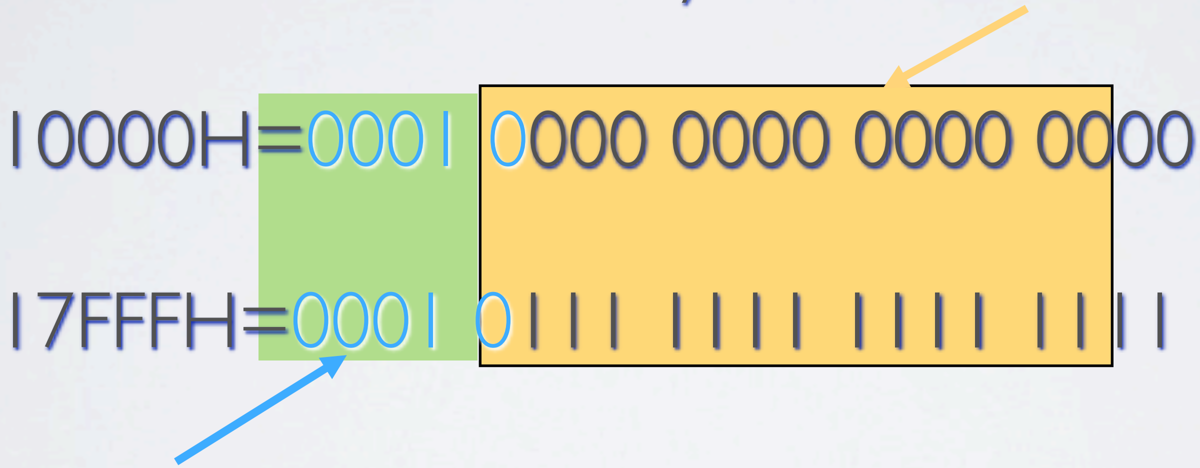


PIN FUNCTION

A_0 - A_{14}	Addresses
IO_0- IO_7	Data connections
$\overline{CS}$	Chip select
$\overline{OE}$	Output enable
$\overline{WE}$	Write enable
V_{CC}	+5V Supply
GND	Ground

- Location 10000H-17FFFH appear as follows in binary.

- Bits A0-A14 => internally decoded

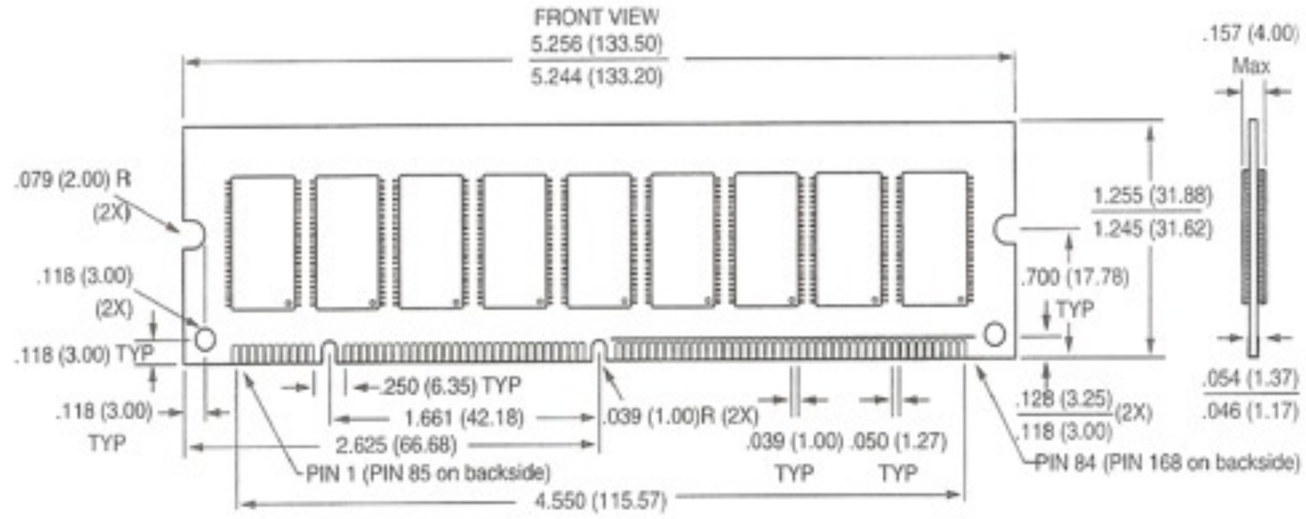


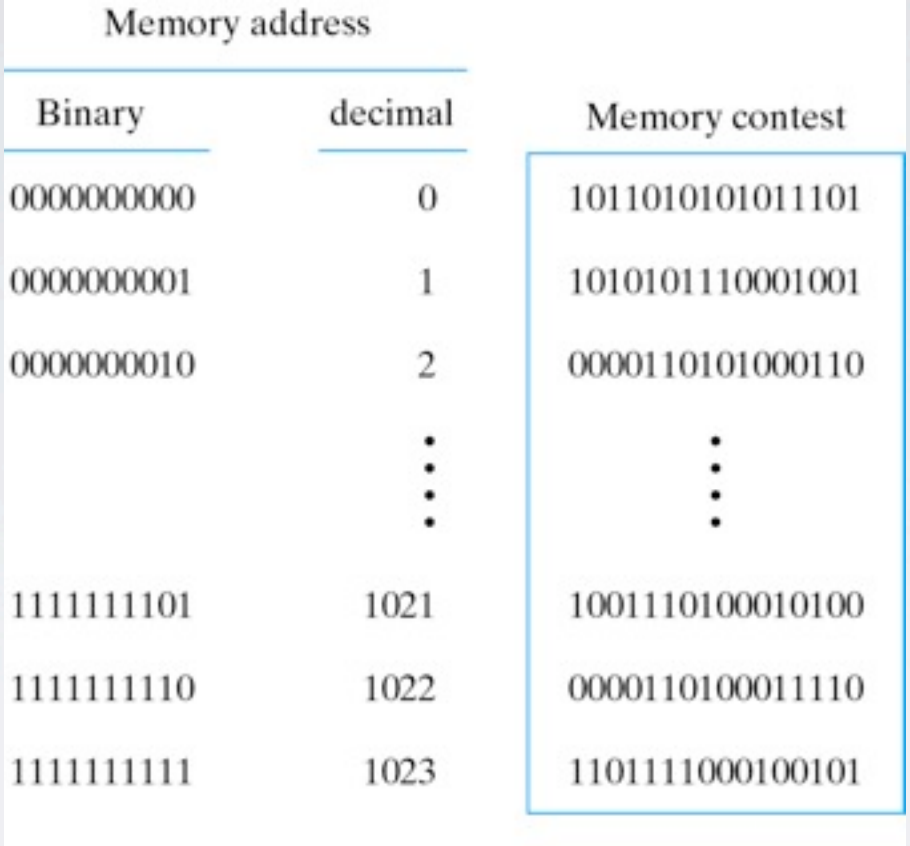
10000H=0001 0000 0000 0000 0000

17FFFH=0001 0111 1111 1111 1111

A19-A15 externally decoded to uniquely select the chip

168-pin *Dual In-Line Memory Module* (DIMM)





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

Fig. 7-3 Content of a 1024 × 16 Memory

STEPS TO WRITE INTO RAM

- Apply address to address lines
- Apply data to data input lines
- Activate *write* input & enable chip

For reads: do 1 and 3 using *read* input

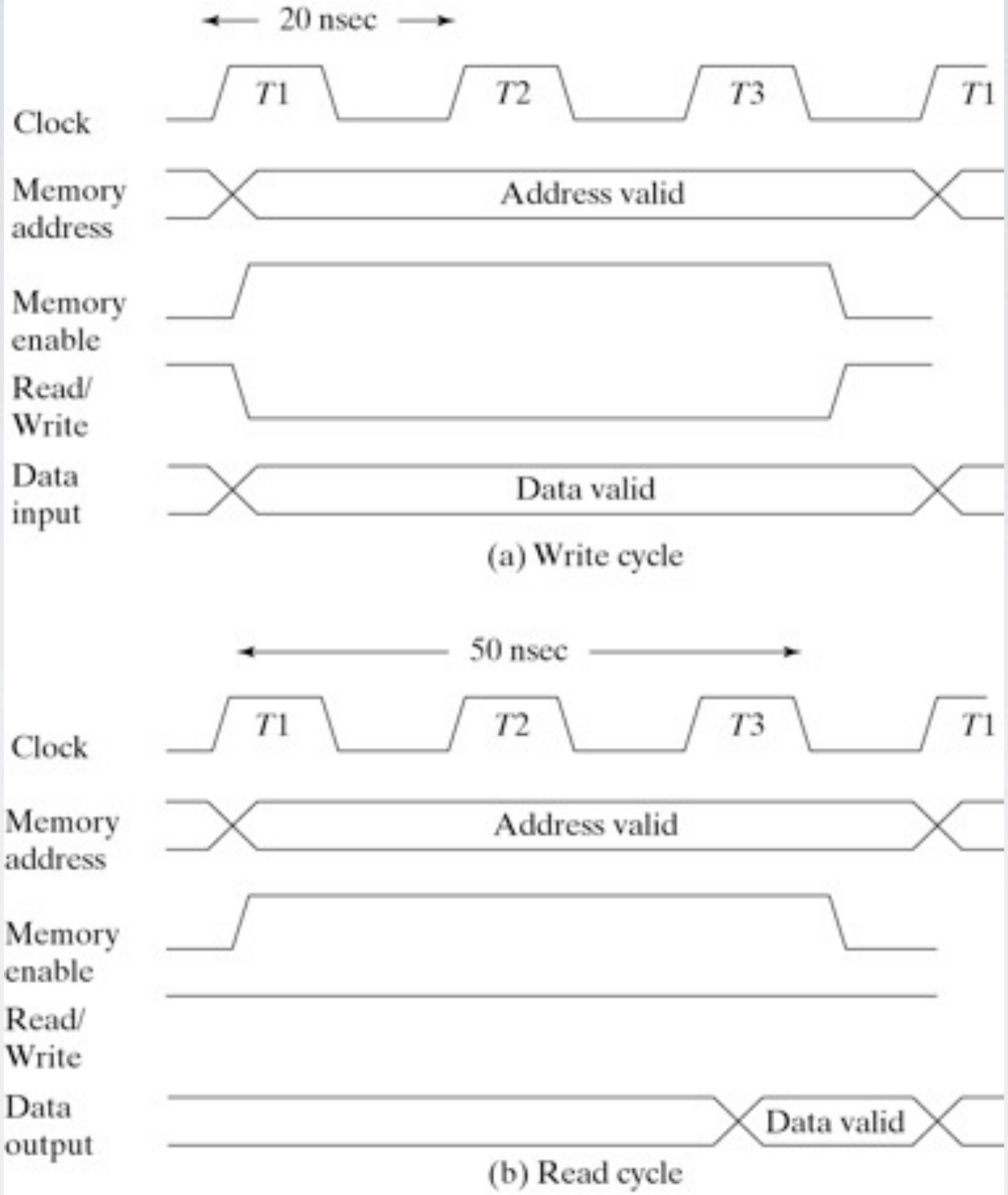
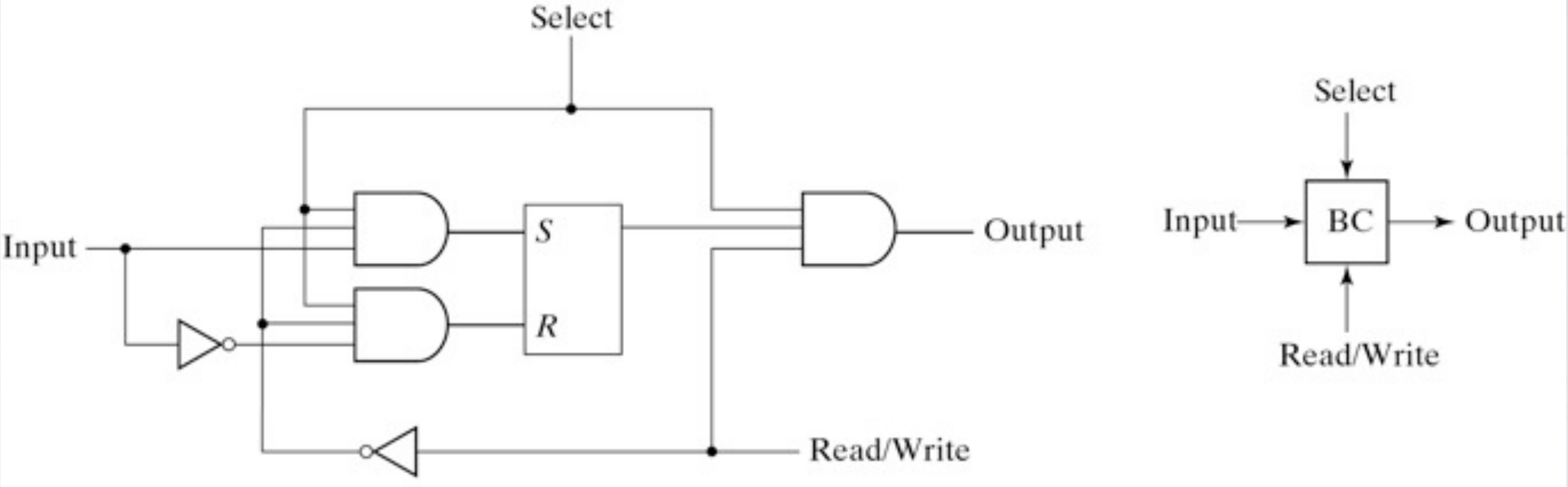


Fig. 7-4 Memory Cycle Timing Waveforms



(a) Logic diagram

(b) Block diagram

Fig. 7-5 Memory Cell

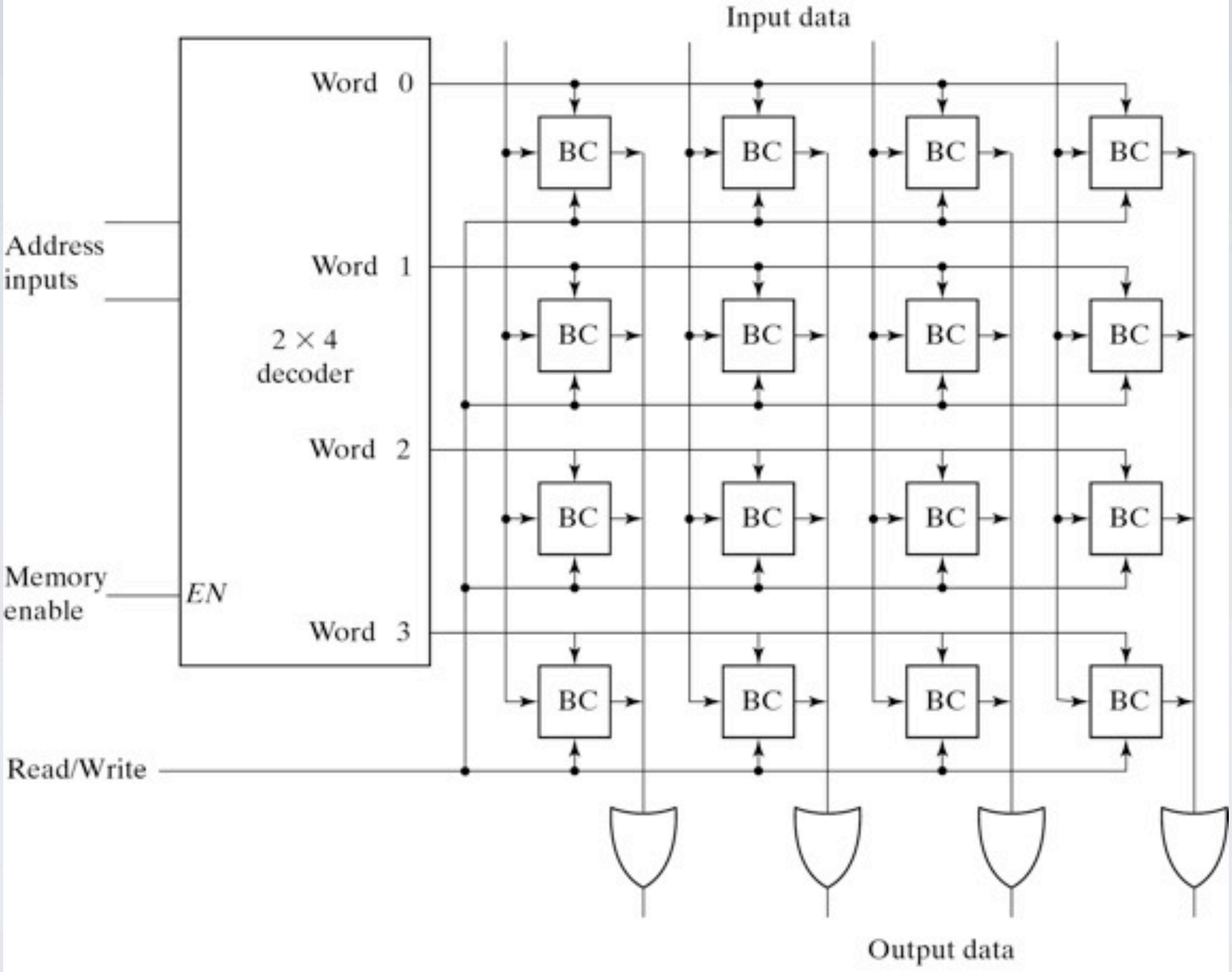


Fig. 7-6 Diagram of a 4 × 4 RAM

ROW/COLUMN DECODING

- 1K-word memory requires 10 address bits and a 10×1024 decoder
- The decoding can also be done with two 5×32 decoder, one for the *row* and one for the column. The cell connected to the row-column intersection is selected.

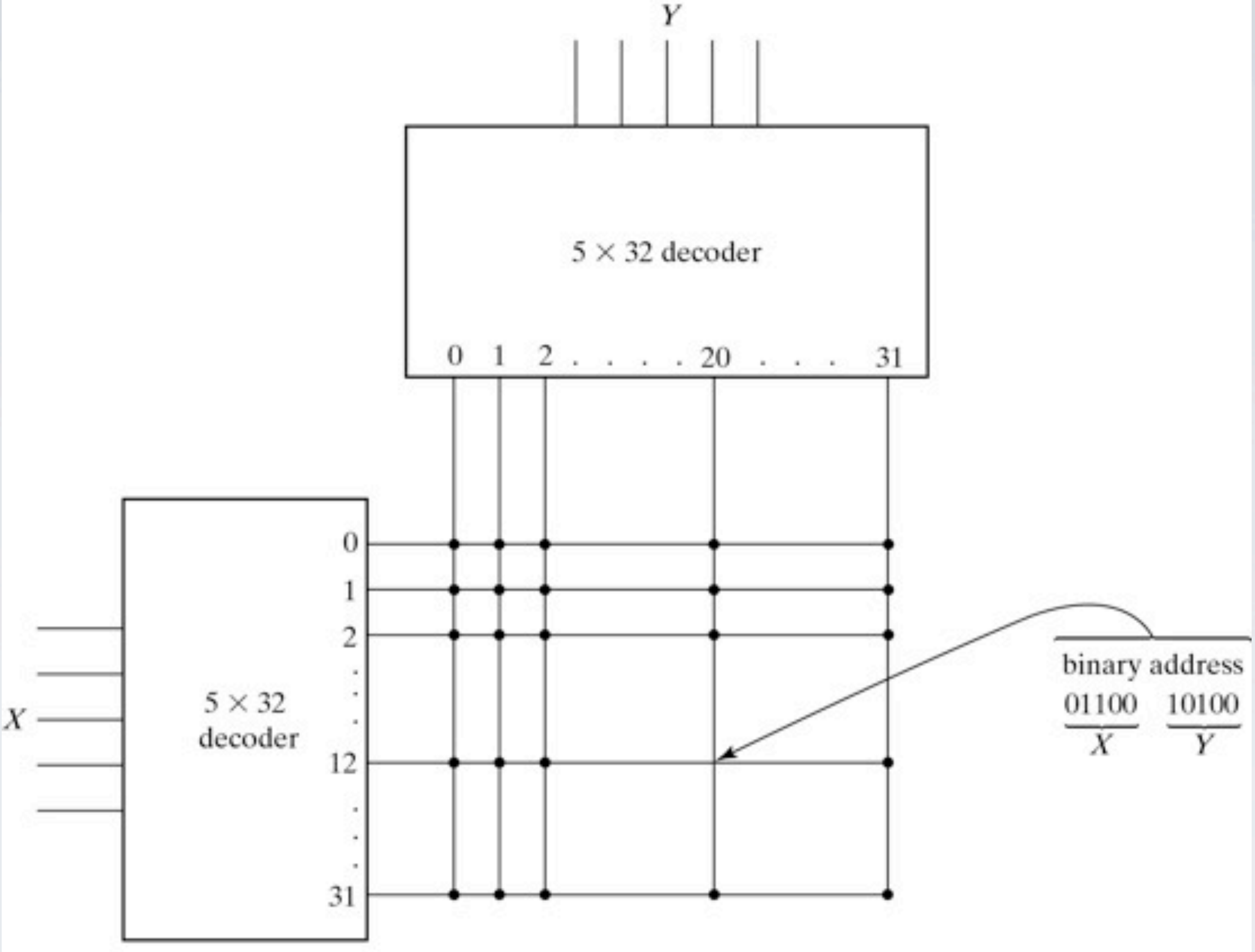


Fig. 7-7 Two-Dimensional Decoding Structure for a 1K-Word Memory

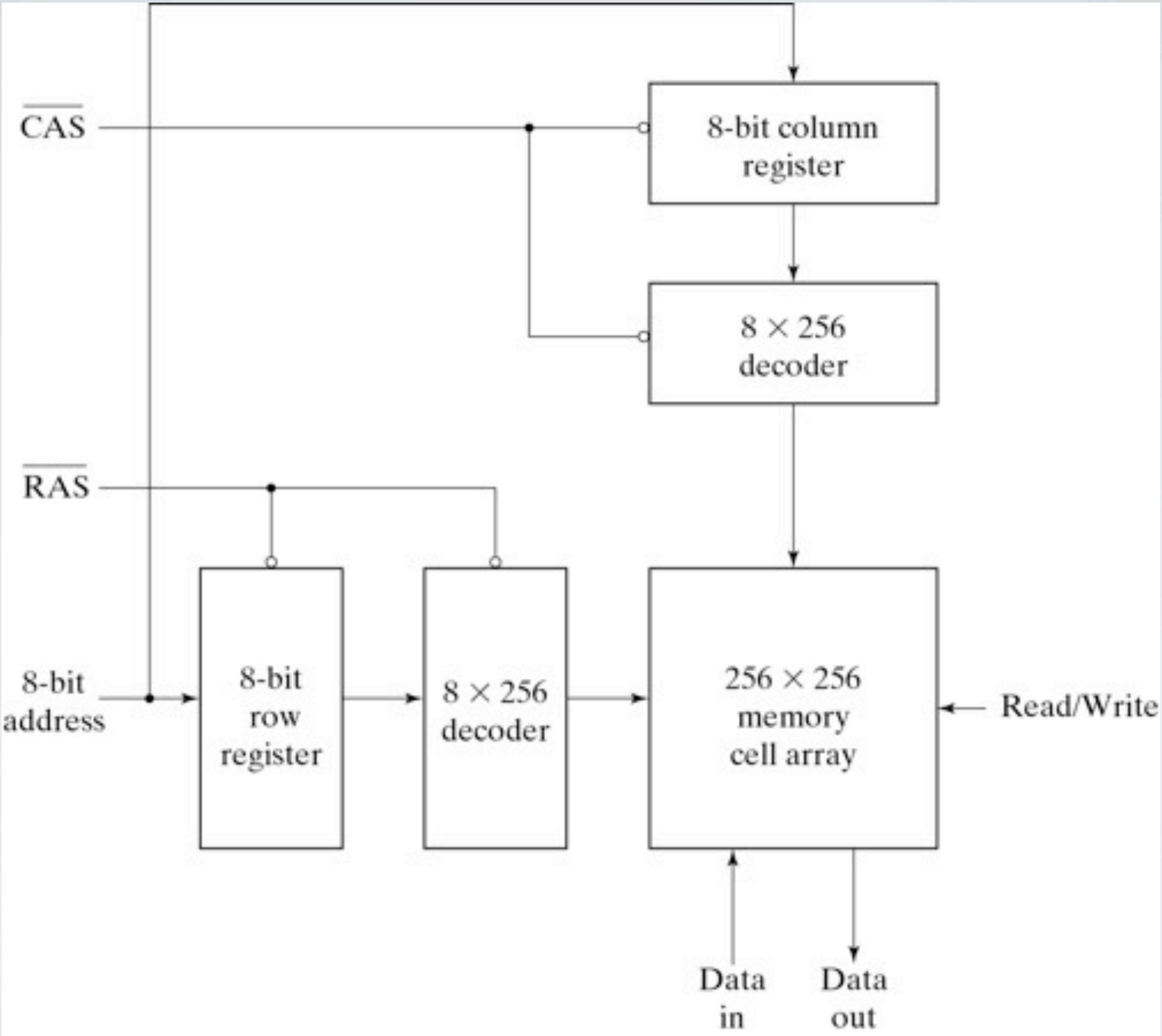


Fig. 7-8 Address Multiplexing for a 64K DRAM

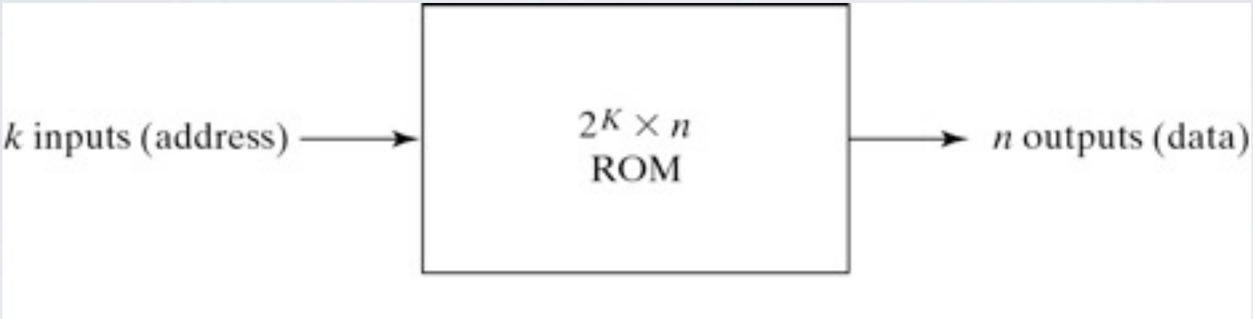


Fig. 7-9 ROM Block Diagram

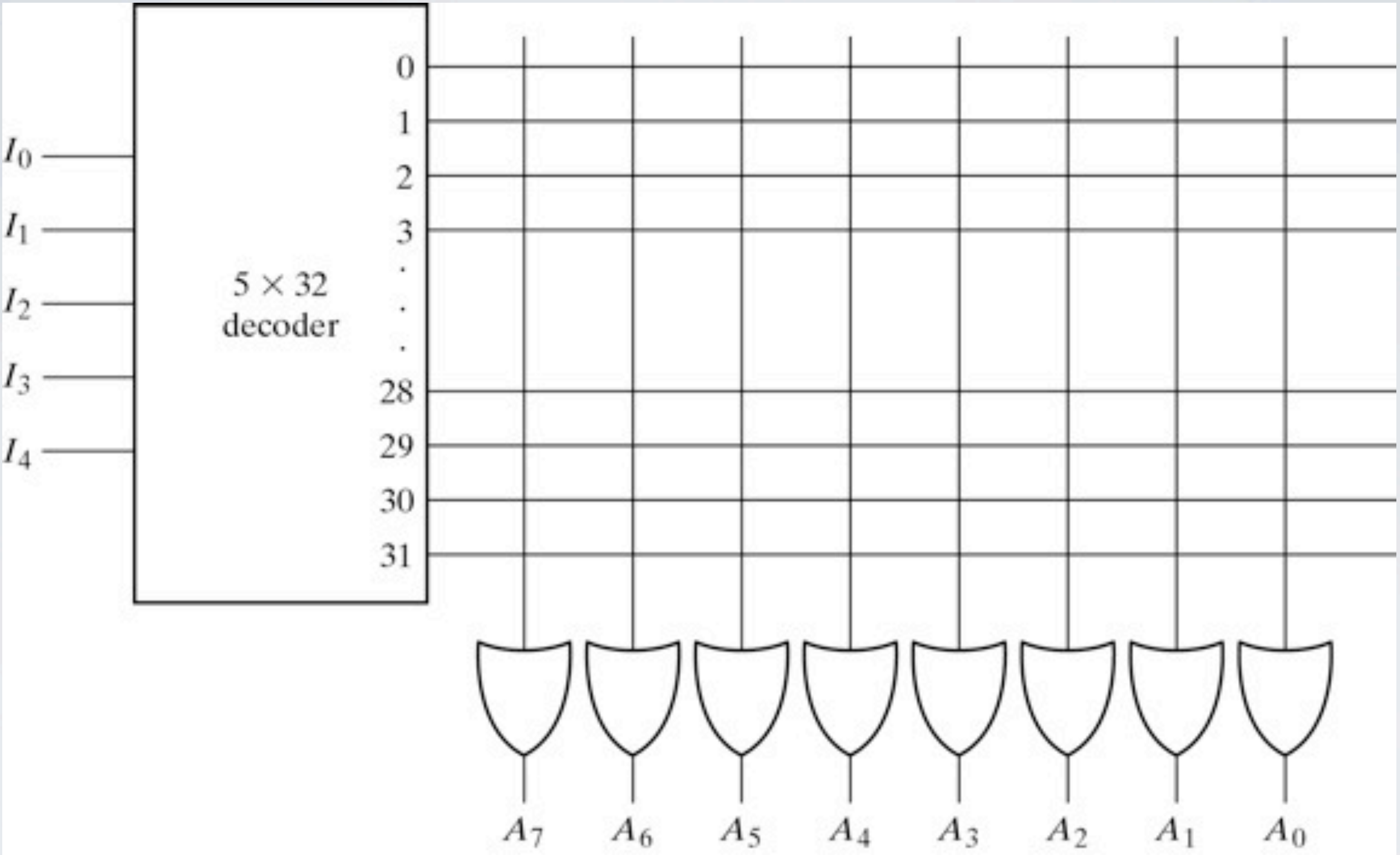


Fig. 7-10 Internal Logic of a 32 × 8 ROM

Used as programable logic, a PROM stores the truth table for N functions of M inputs. N = number of bits in each cell.

M = number of address bits; there are 2^M memory locations in the PROM.

"x" indicates a connection, and a "1" in the truth table

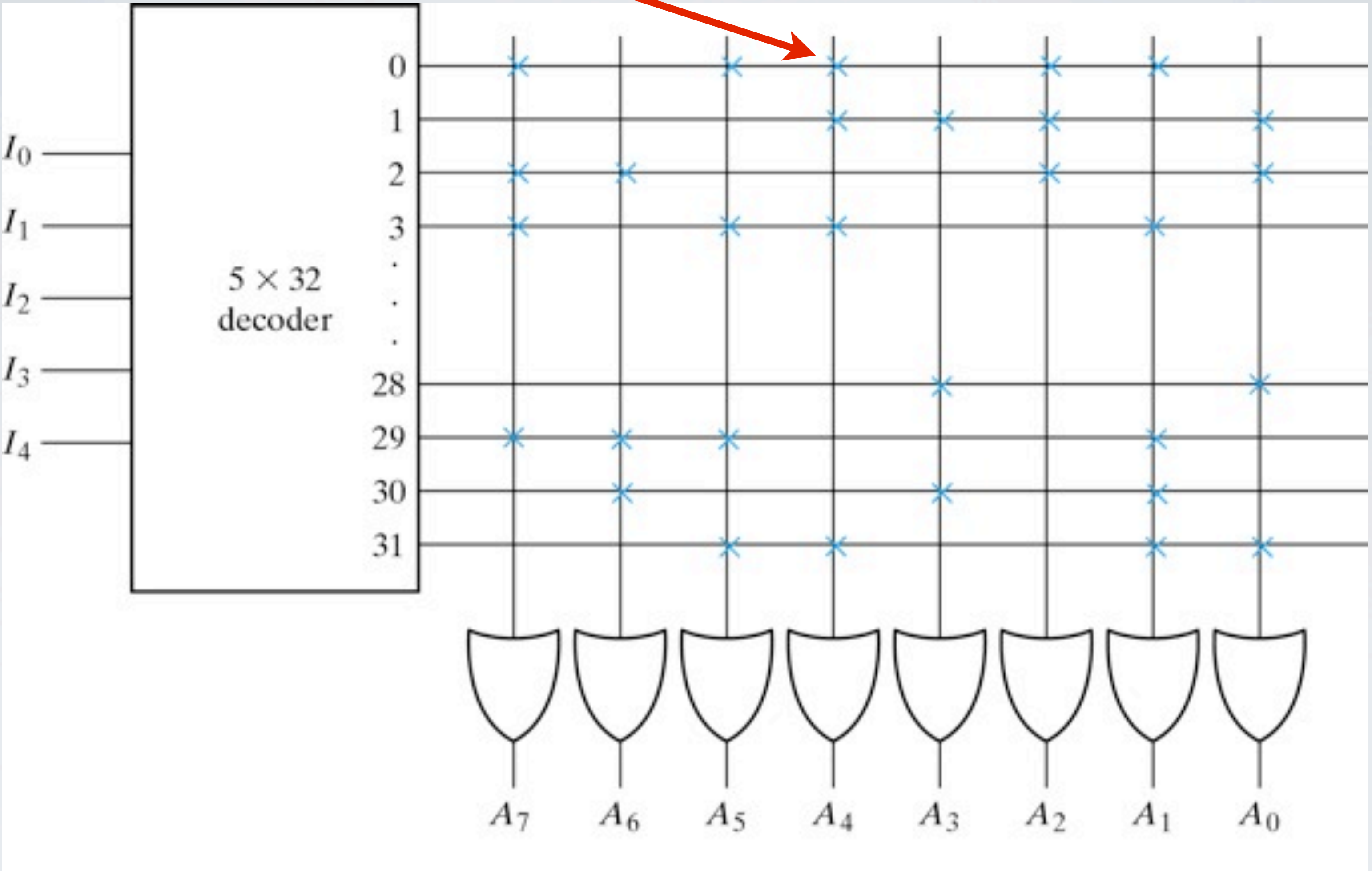


Fig. 7-11 Programming the ROM According to Table 7-3

Address 00000: cell contents is 10110110

Design a combinatorial circuit using a ROM. The circuit accepts a 3-bit input number and outputs a binary number equal to the square of the input.

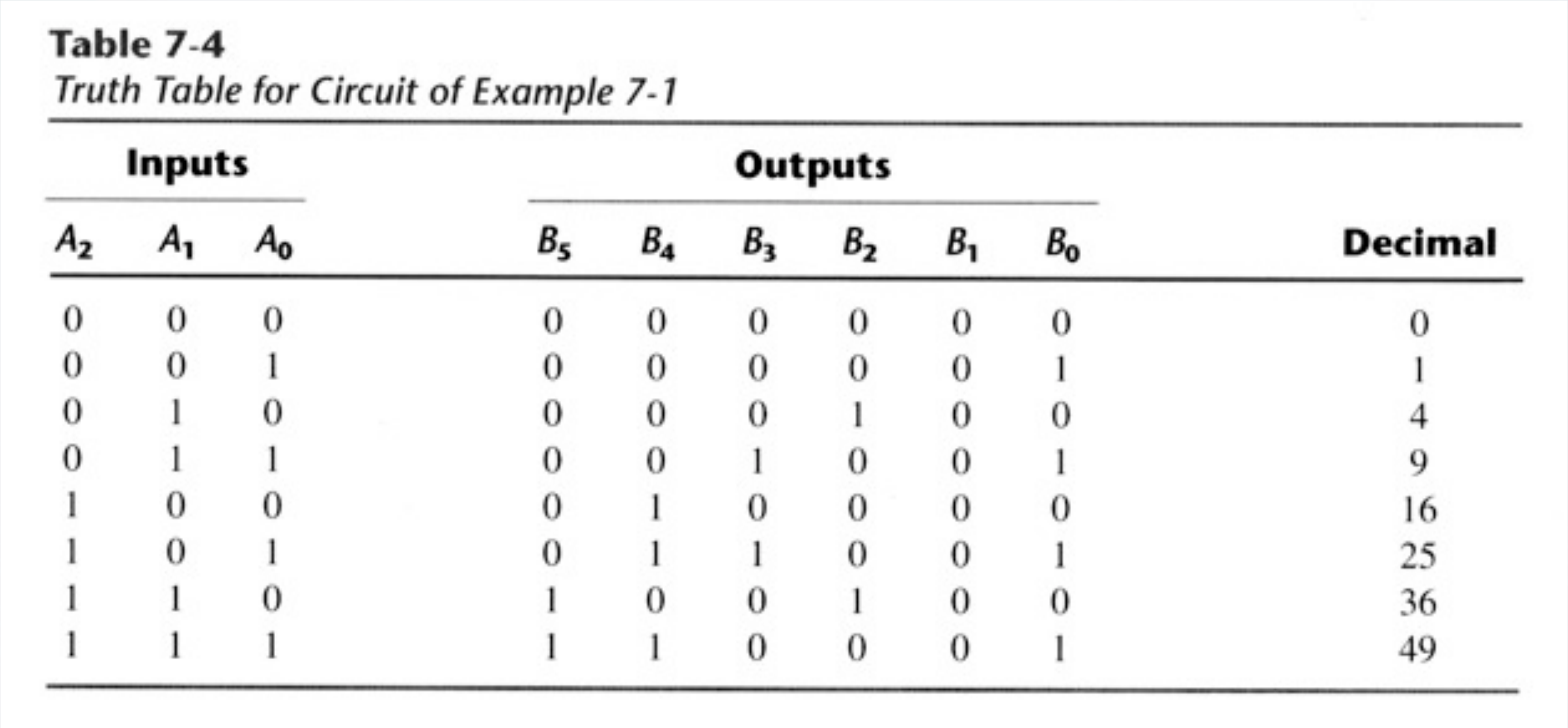
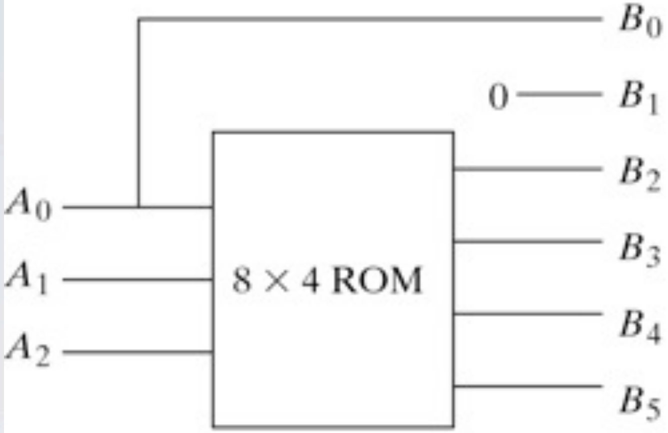


Table 7-4
Truth Table for Circuit of Example 7-1

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

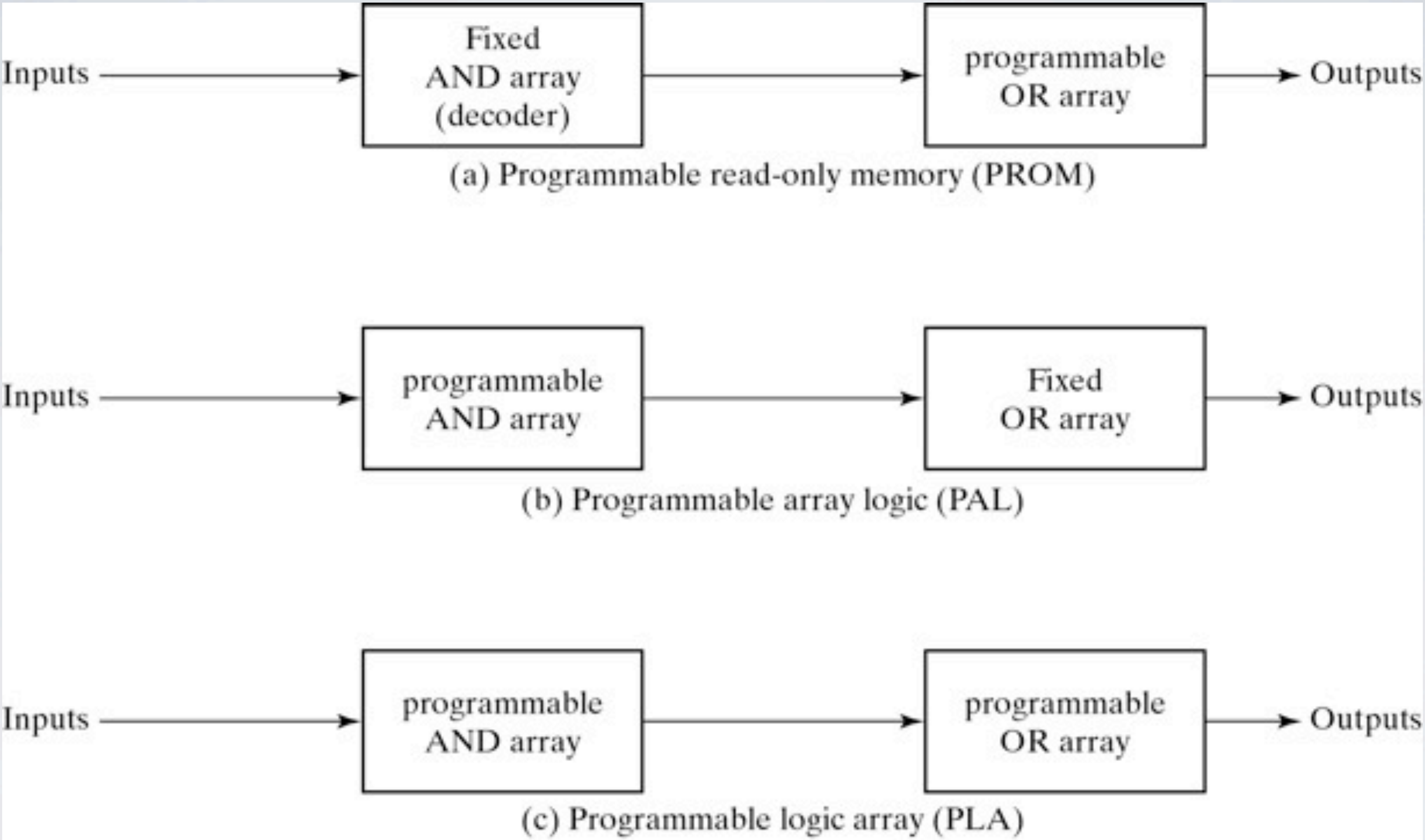


(a) Block diagram

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. 7-12 ROM Implementation of Example 7-1



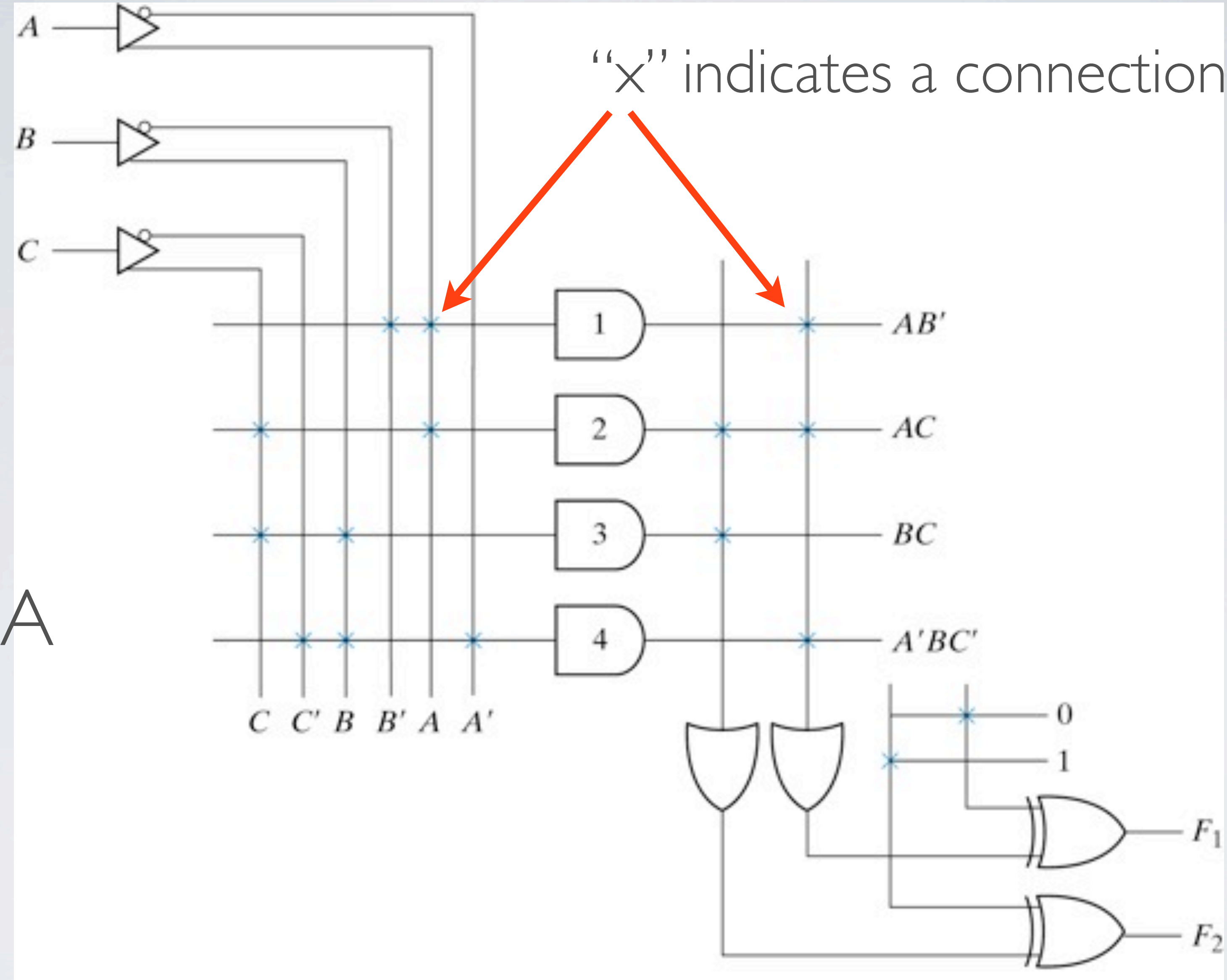
(a) Programmable read-only memory (PROM)

(b) Programmable array logic (PAL)

(c) Programmable logic array (PLA)

Fig. 7-13 Basic Configuration of Three PLDs

PLA



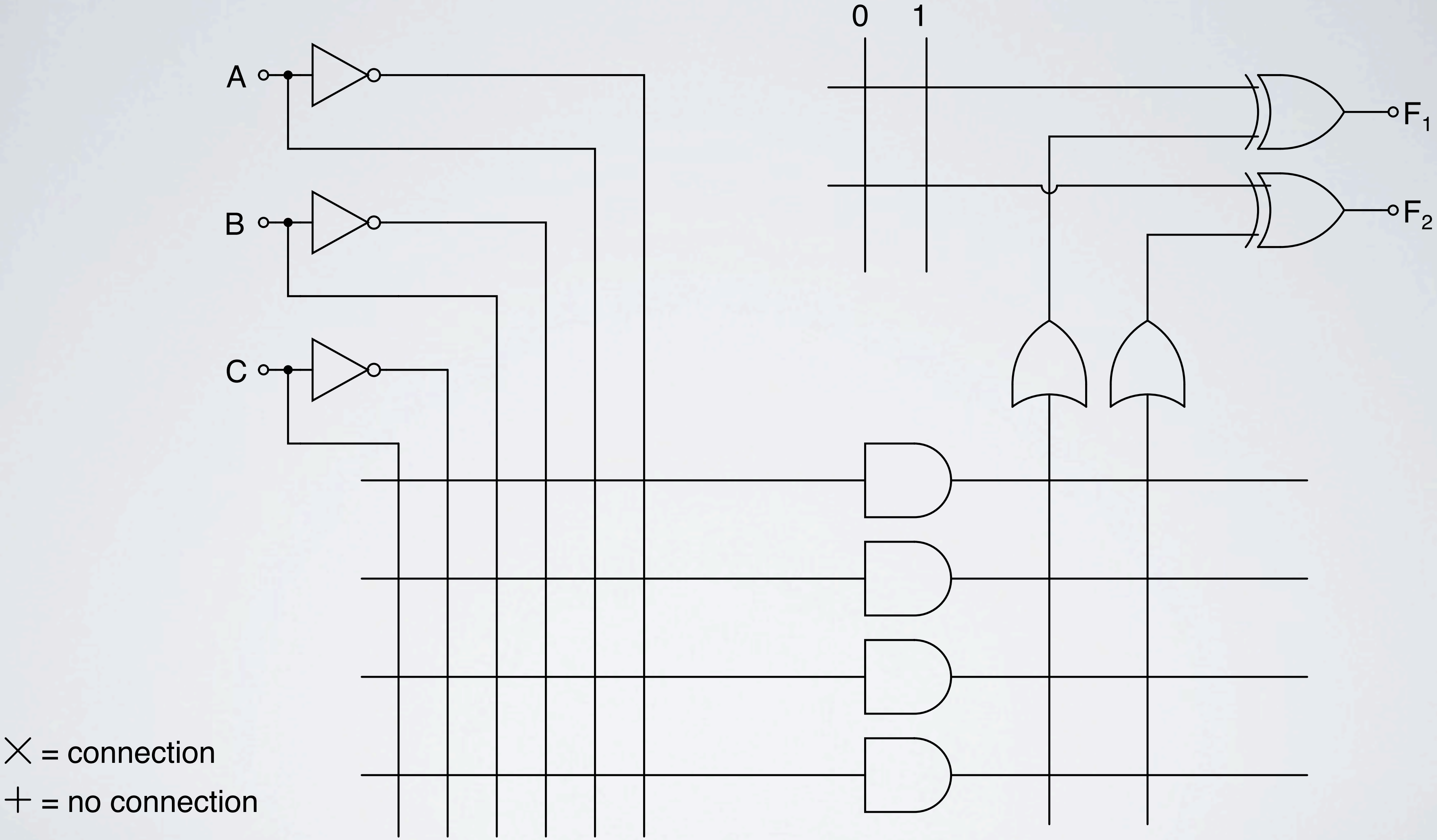
“x” indicates a connection

Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

PLA has limited # of ANDs: designer can implement F’ (instead of F) to minimize the # of distinct product terms, then complement F’ with the XOR to produce F

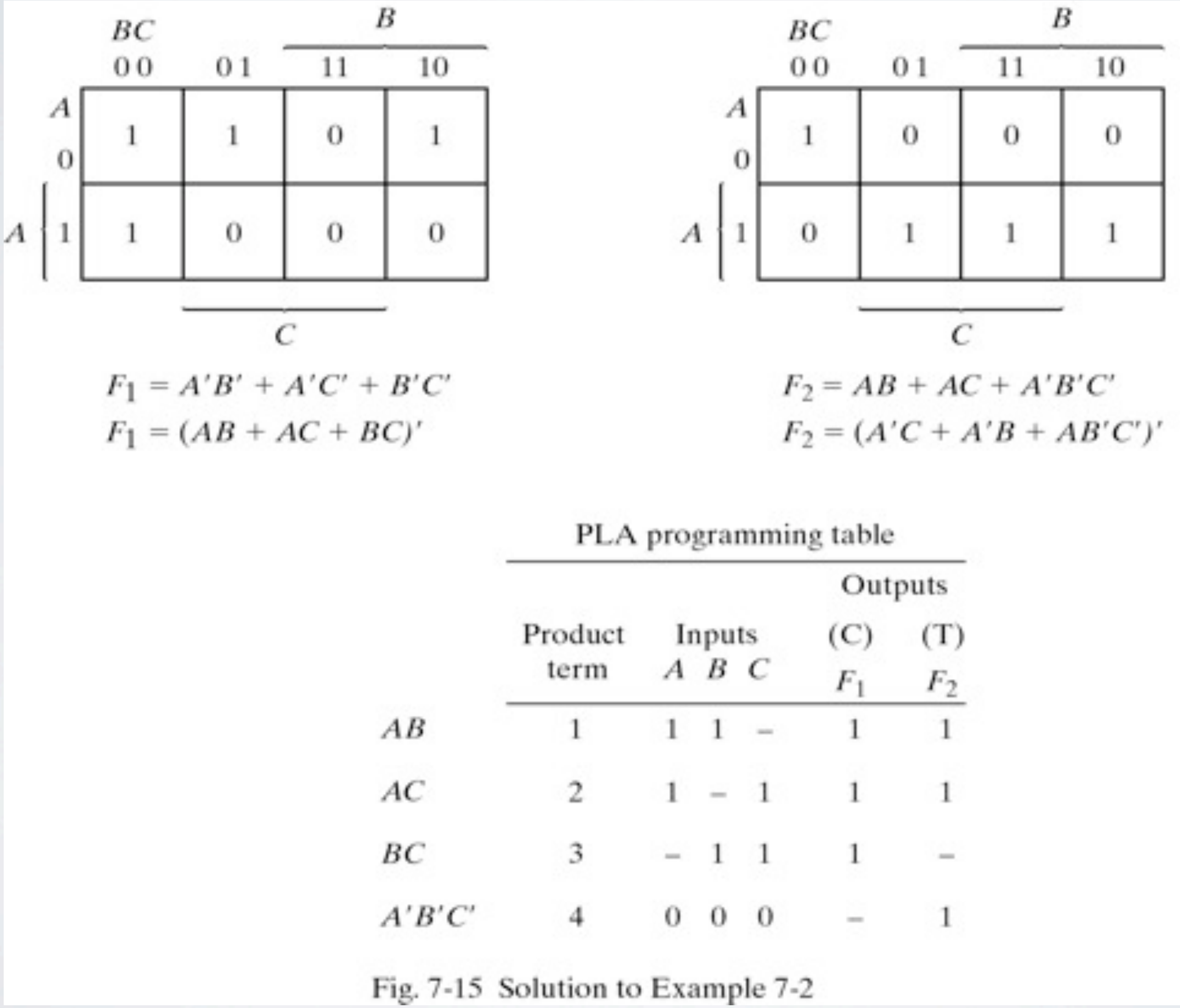
Example: Implement the following boolean functions in a PLA:

$$F_1(A,B,C) = \sum(0,1,2,4)$$
$$F_2(A,B,C) = \sum(0,5,6,7)$$



× = connection

+ = no connection



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

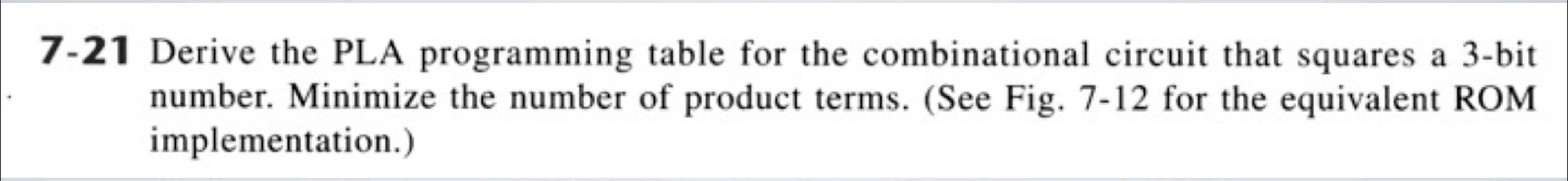
$F_2 = AB + AC + A'B'C'$

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

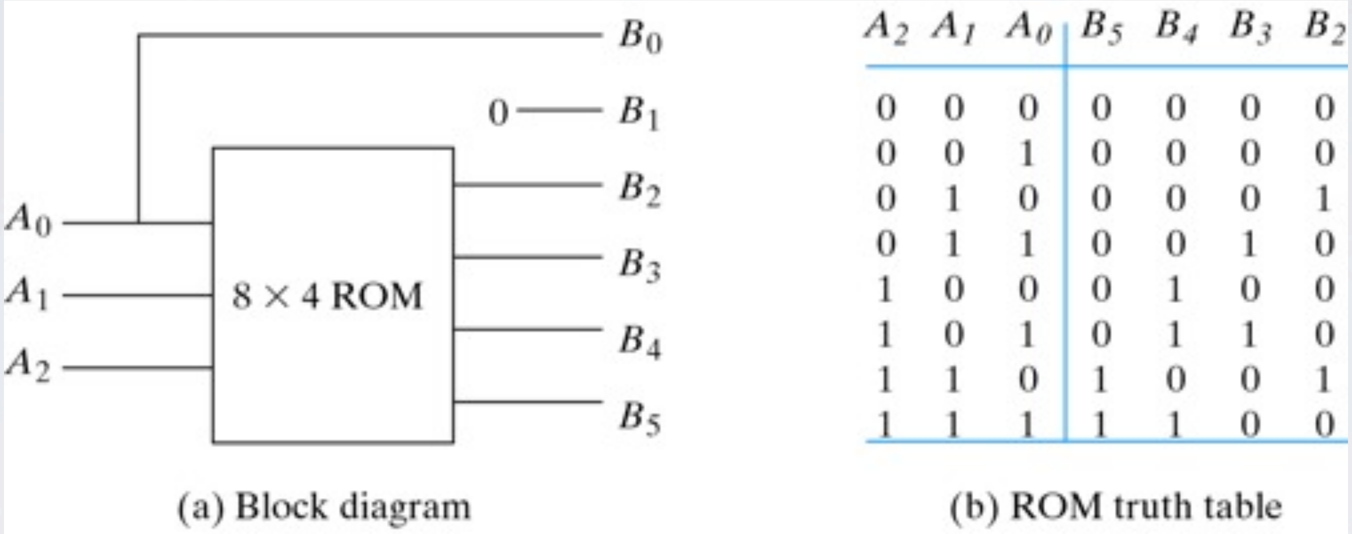
PLA programming table

	Product term	Inputs			Outputs	
					(C)	(T)
		A	B	C	F_1	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

Fig. 7-15 Solution to Example 7-2



7-21 Derive the PLA programming table for the combinational circuit that squares a 3-bit number. Minimize the number of product terms. (See Fig. 7-12 for the equivalent ROM implementation.)

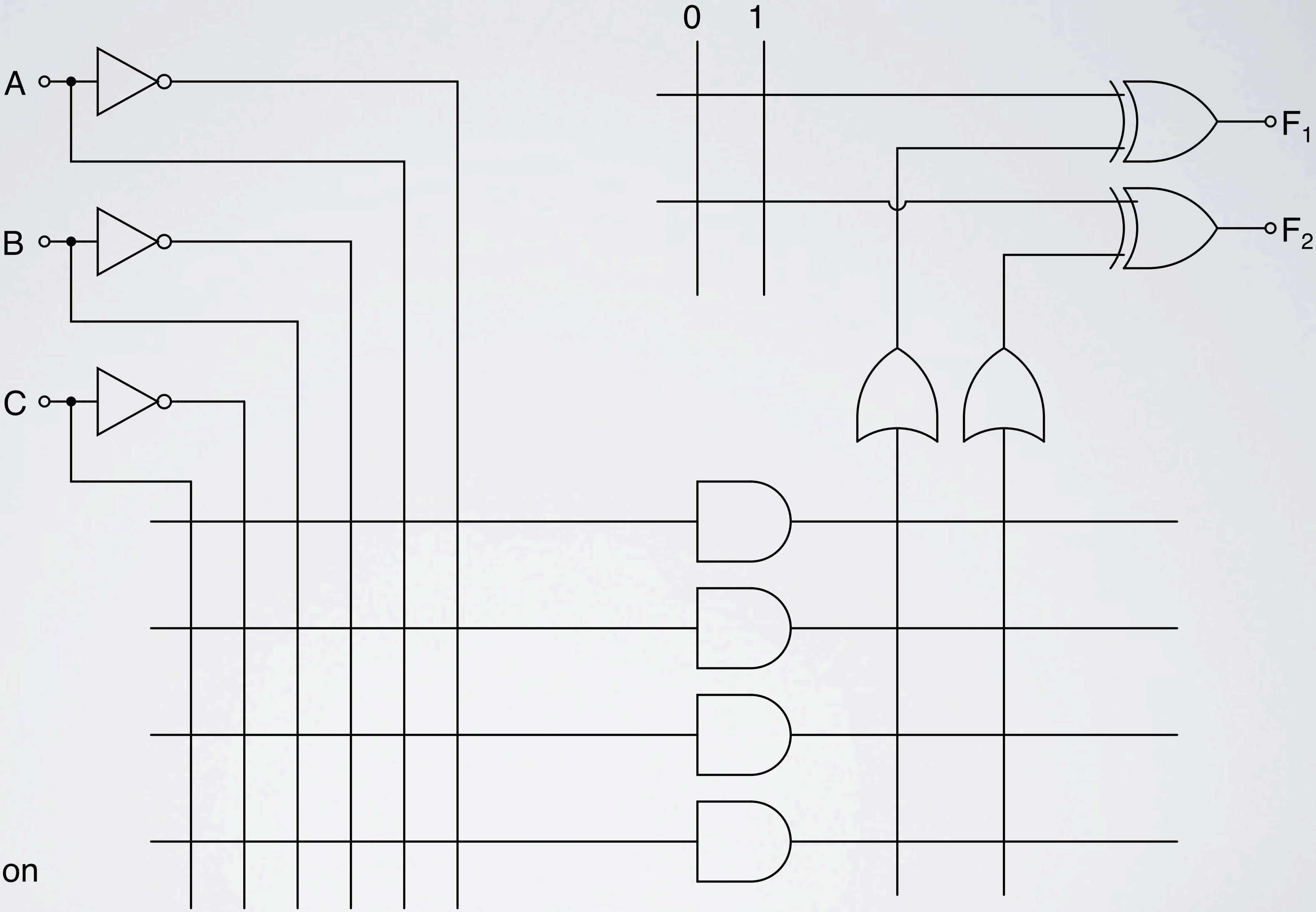


(a) Block diagram

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

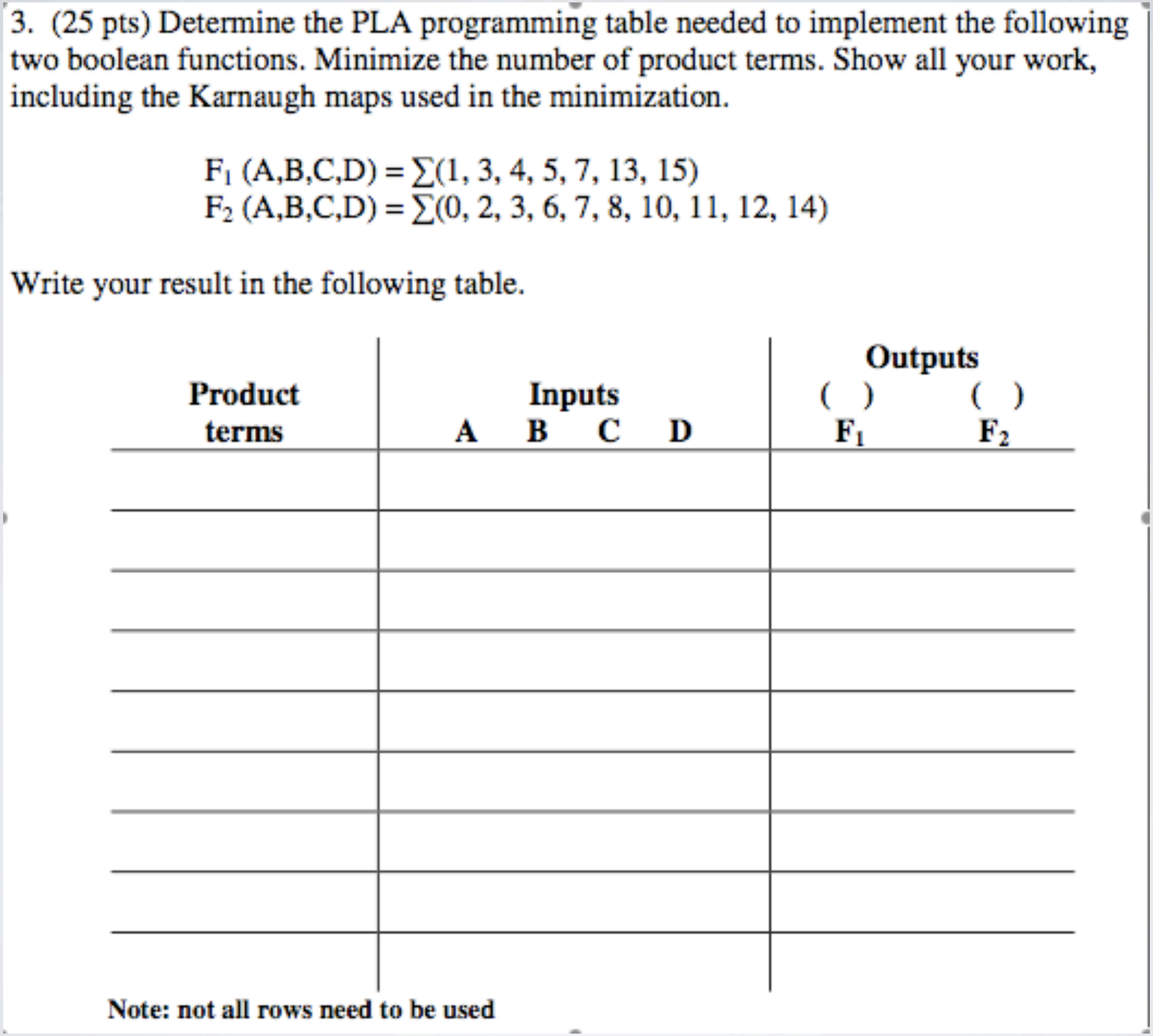
(b) ROM truth table

Fig. 7-12 ROM Implementation of Example 7-1



✕ = connection

+ = no connection



3. (25 pts) Determine the PLA programming table needed to implement the following two boolean functions. Minimize the number of product terms. Show all your work, including the Karnaugh maps used in the minimization.

$$F_1\ (A,B,C,D) = \sum(1, 3, 4, 5, 7, 13, 15)$$
$$F_2\ (A,B,C,D) = \sum(0, 2, 3, 6, 7, 8, 10, 11, 12, 14)$$

Write your result in the following table.

Product terms	Inputs				Outputs	
	A	B	C	D	( ) F_1	( ) F_2

Note: not all rows need to be used

PAL

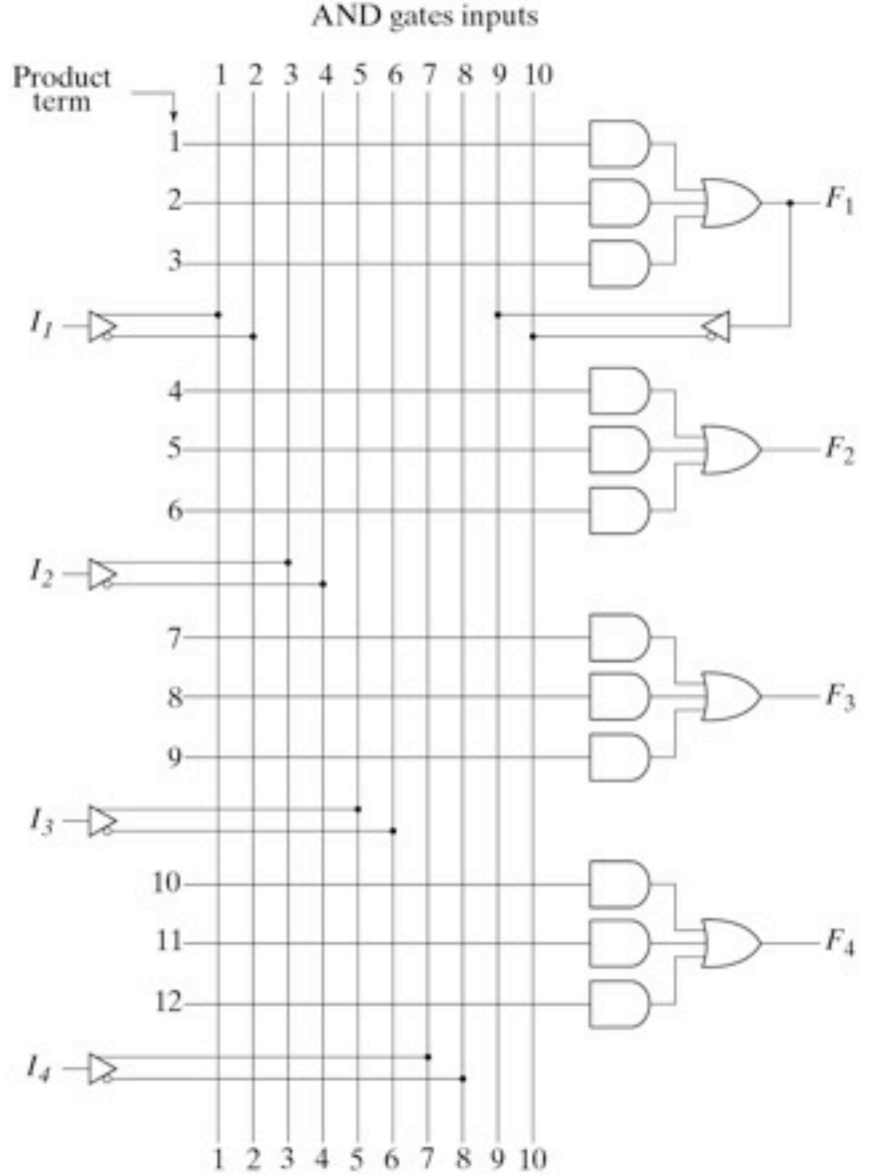
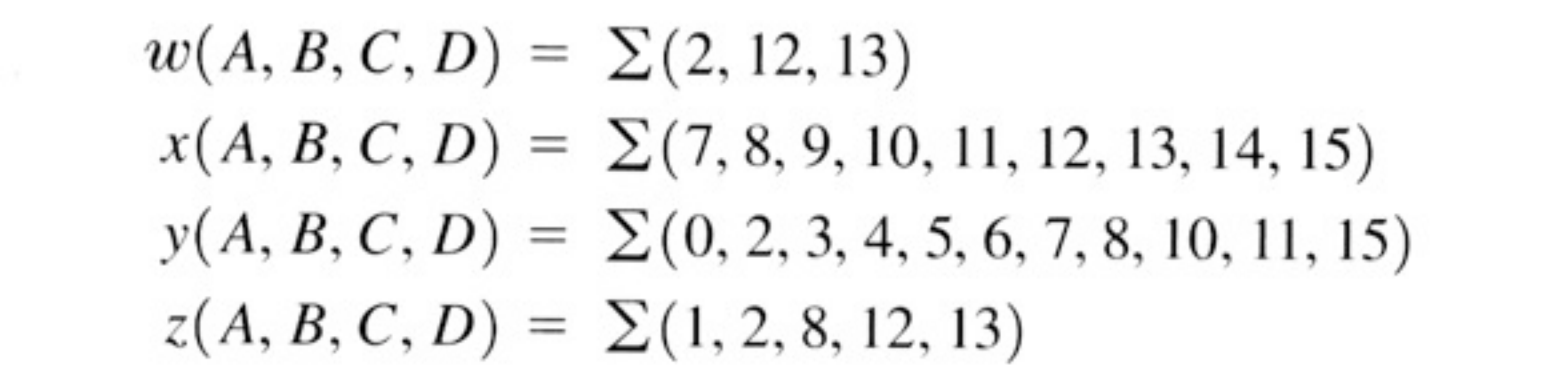


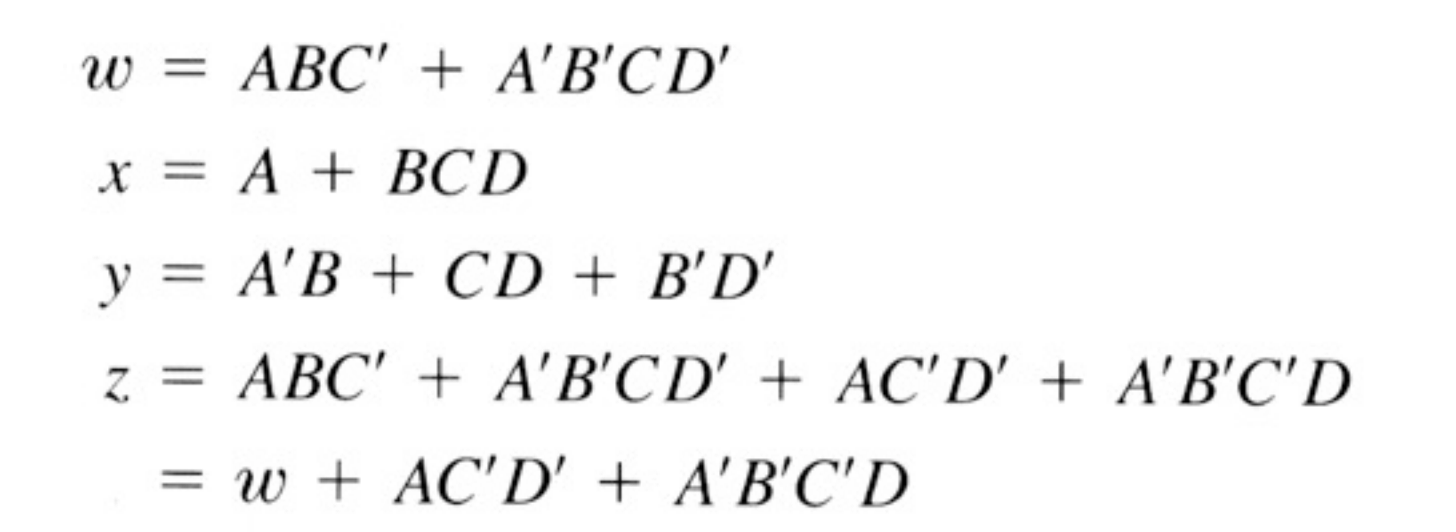
Fig. 7-16 PAL with Four Inputs, Four Outputs, and Three-Wide AND-OR Structure

PAL: Only inputs to AND gates can be programmed but one term (F_1) can be re-used in other functions



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

Manipulate expressions so that a common term is identified. Assign common term to F_1.



$$\begin{aligned} w &= ABC' + A'B'CD' \\ x &= A + BCD \\ y &= A'B + CD + B'D' \\ z &= ABC' + A'B'CD' + AC'D' + A'B'C'D \\ &= w + AC'D' + A'B'C'D \end{aligned}$$

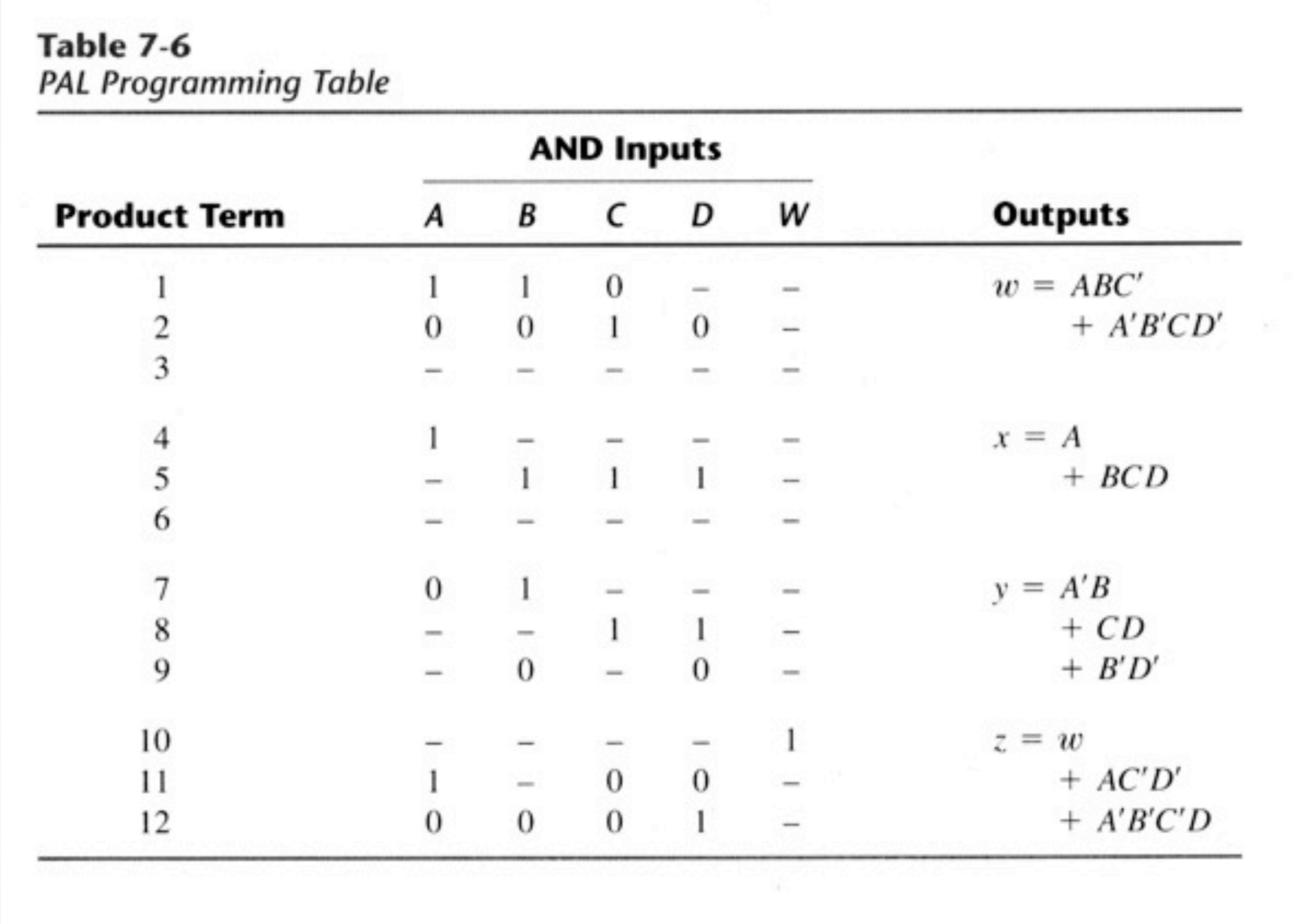


Table 7-6
PAL Programming Table

	AND Inputs					
Product Term	*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	–	$+ A'B'C'D$

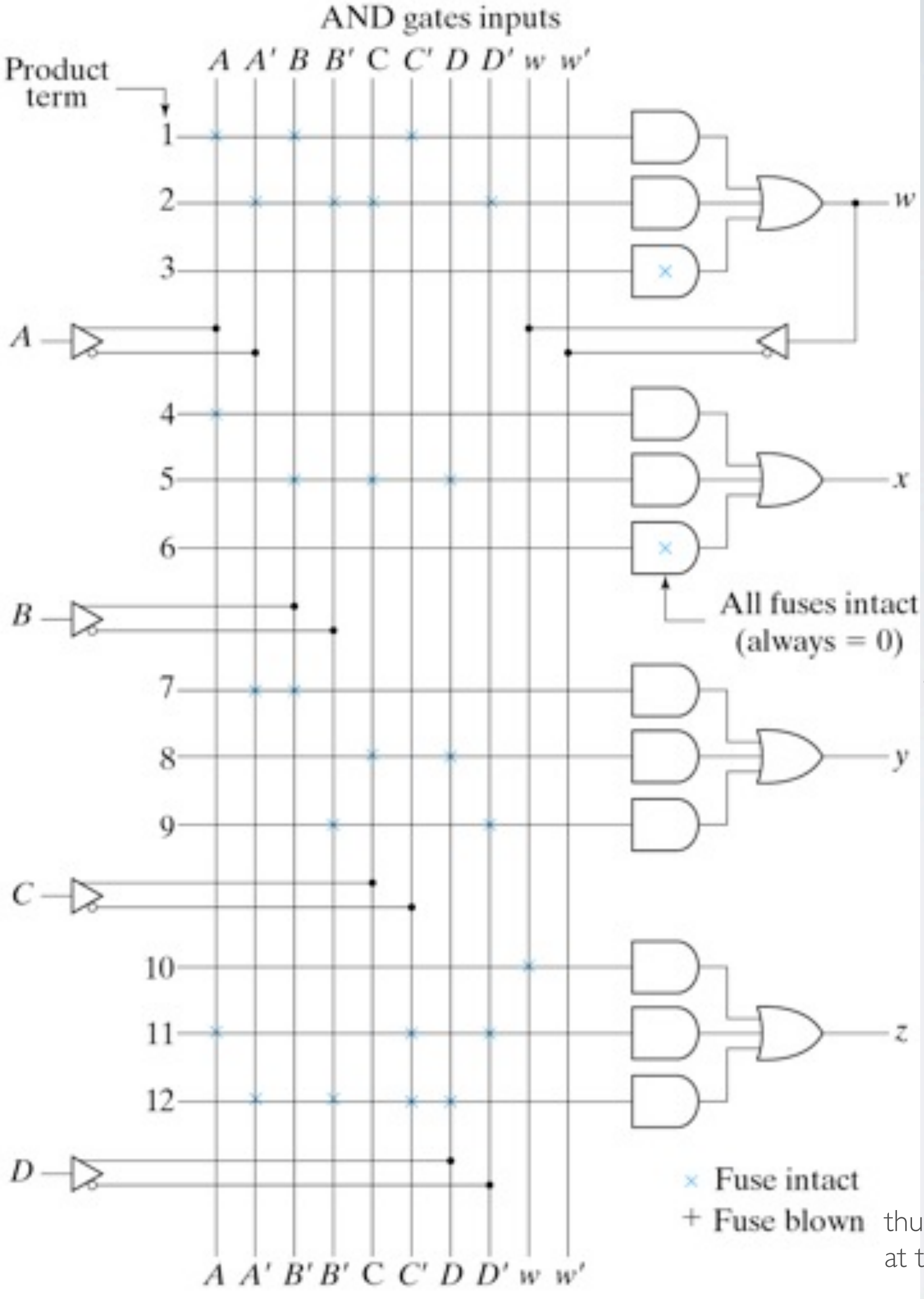


Fig. 7-17 Fuse Map for PAL as Specified in Table 7-6

Inputs are I AND I'
and thus produces a 0 always

thus there is no connection
at these points

- PAL practice: problem 7-24

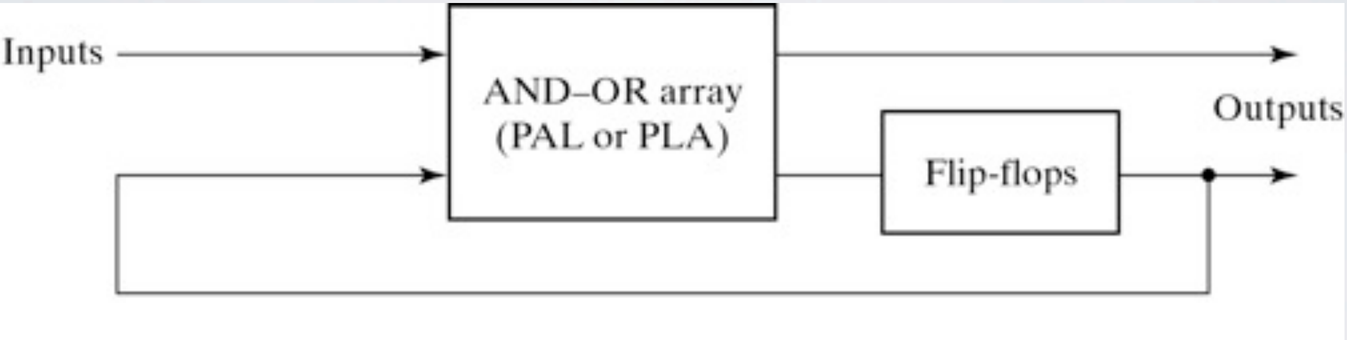


Fig. 7-18 Sequential Programmable Logic Device

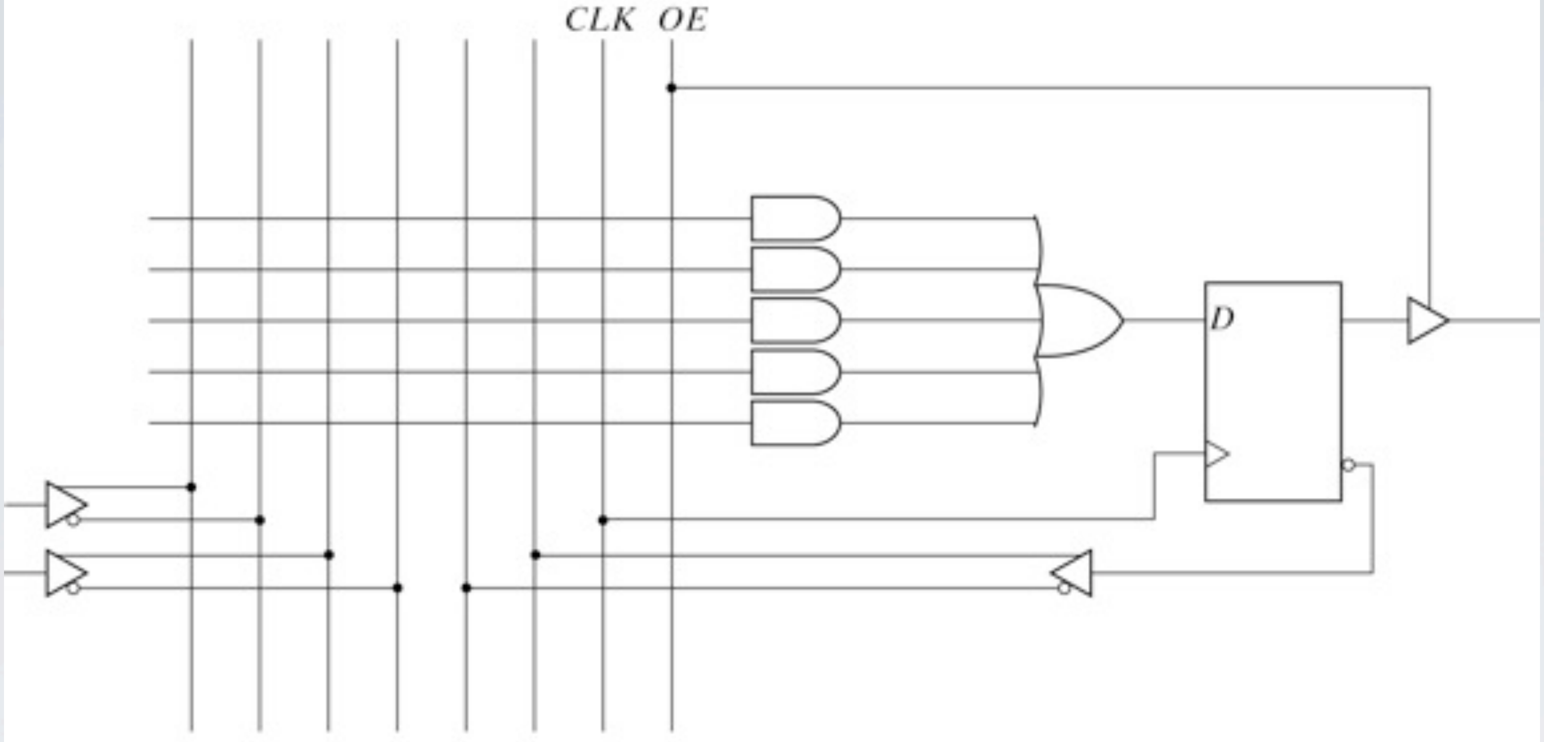


Fig. 7-19 Basic Macrocell Logic

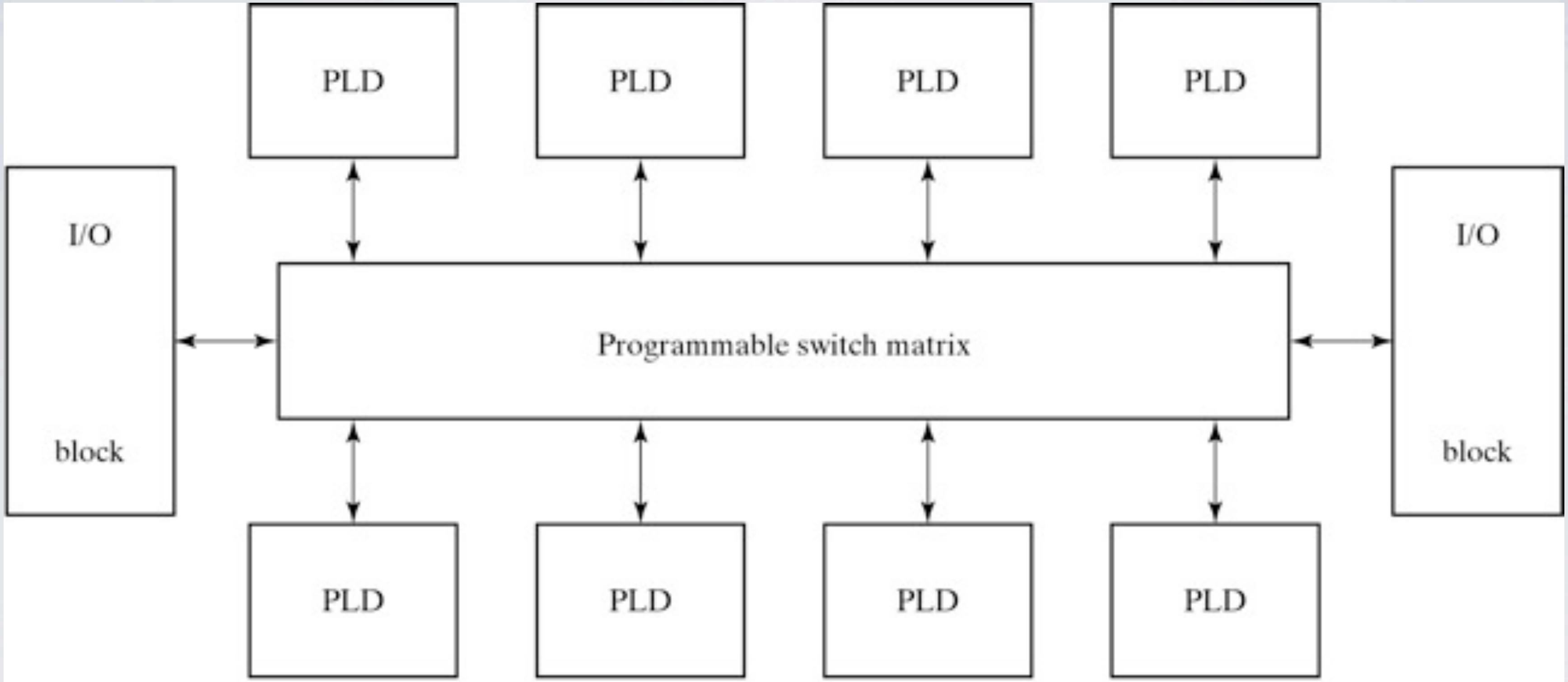
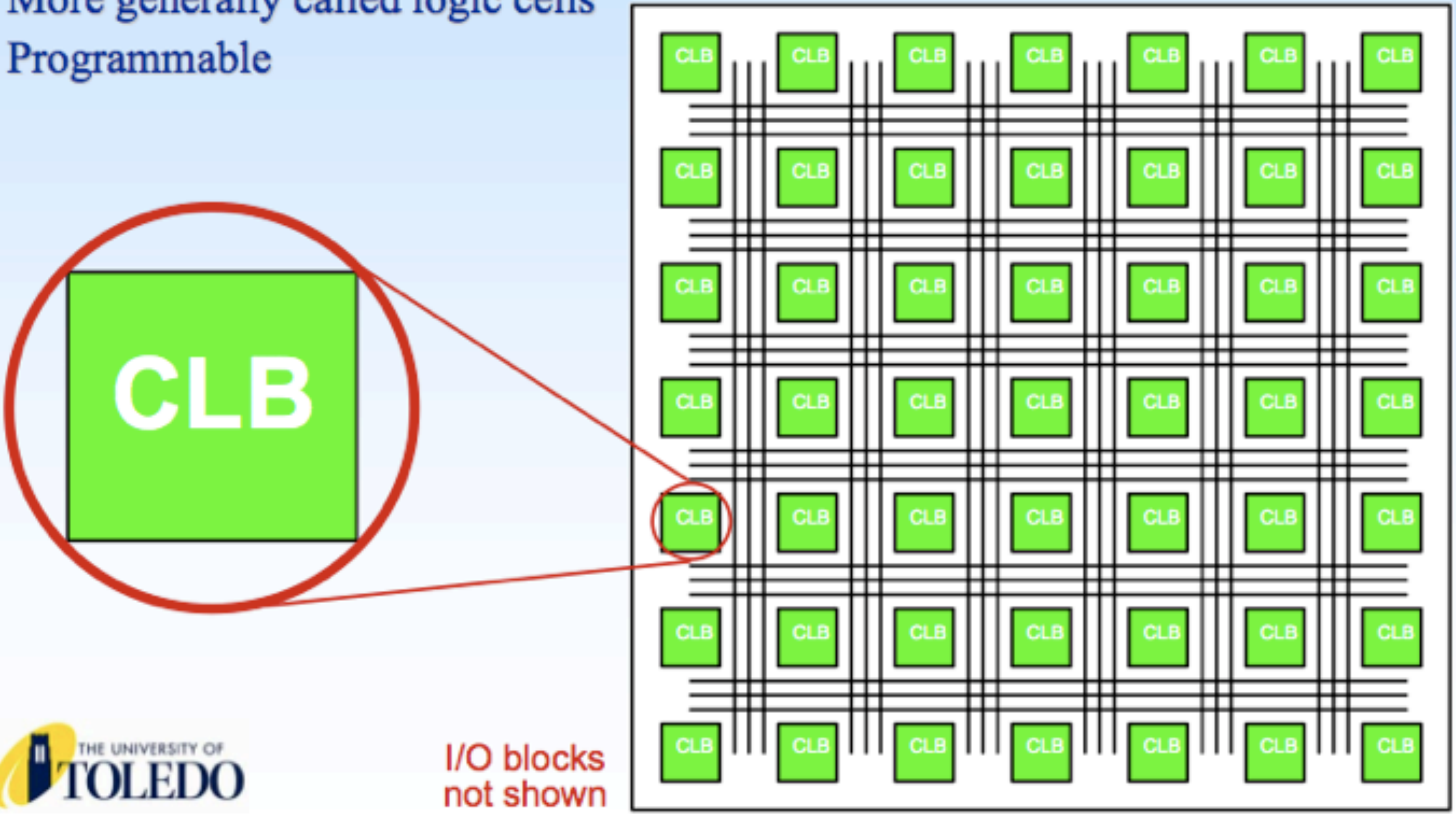


Fig. 7-20 General CPLD Configuration

Xilinx FPGAs are based on Configurable Logic Blocks (CLBs)

More generally called logic cells

Programmable



I/O blocks not shown

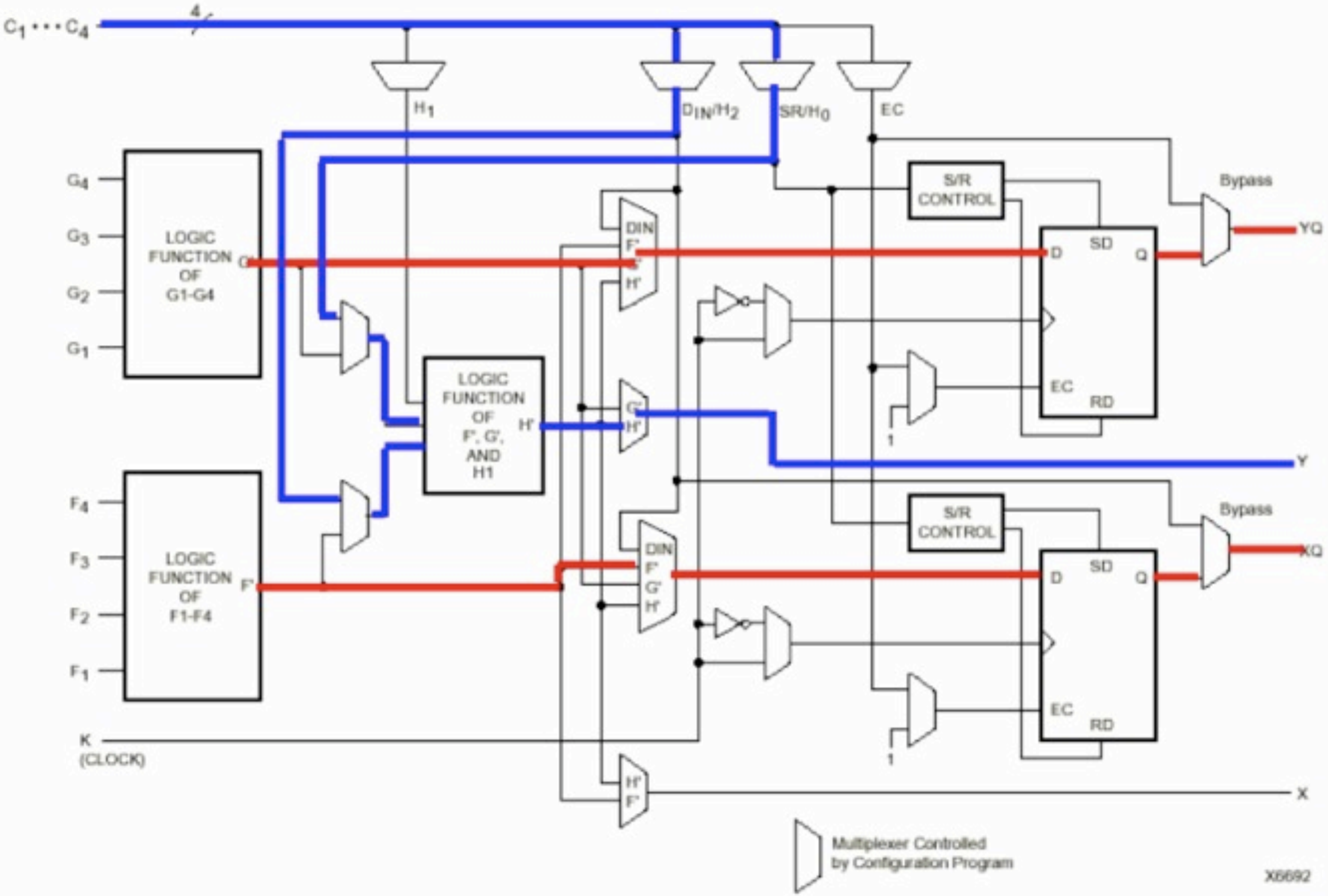
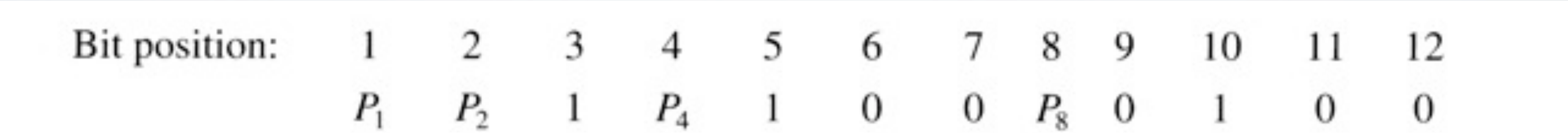
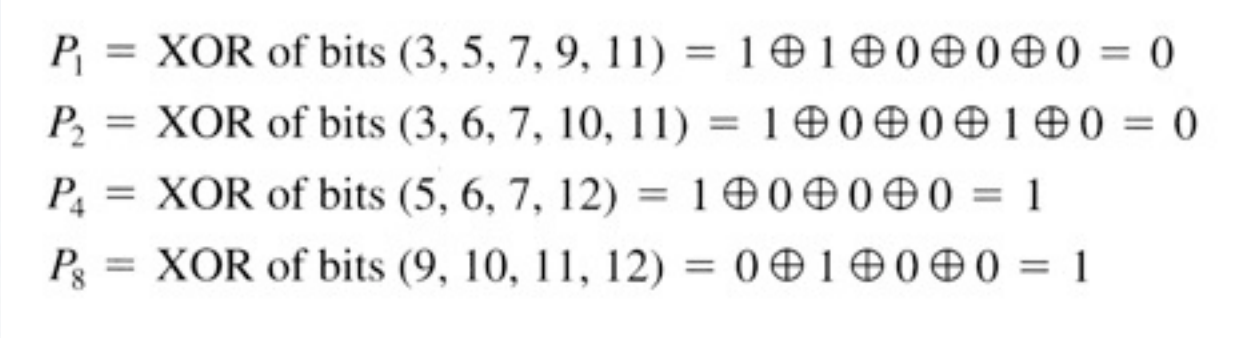


Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

HAMMING CODE: ERROR DETECTION AND CORRECTION



Bit position:	1	2	3	4	5	6	7	8	9	10	11	12
	P_1	P_2	1	P_4	1	0	0	P_8	0	1	0	0

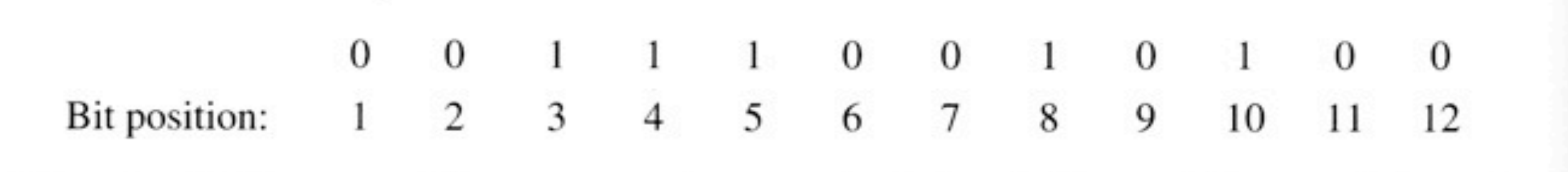


$$P_1 = \text{XOR of bits } (3, 5, 7, 9, 11) = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$$

$$P_2 = \text{XOR of bits } (3, 6, 7, 10, 11) = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$$

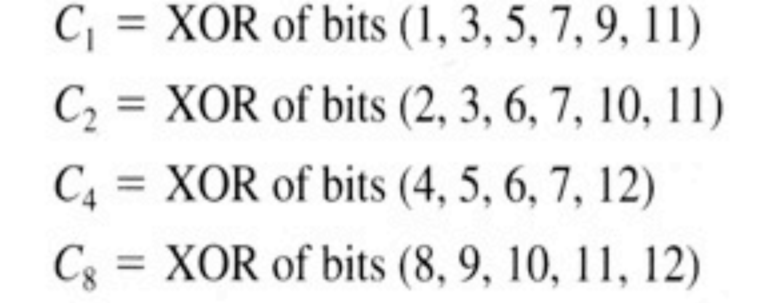
$$P_4 = \text{XOR of bits } (5, 6, 7, 12) = 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$P_8 = \text{XOR of bits } (9, 10, 11, 12) = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$



	0	0	1	1	1	0	0	1	0	1	0	0
Bit position:	1	2	3	4	5	6	7	8	9	10	11	12

- Compute correction bits

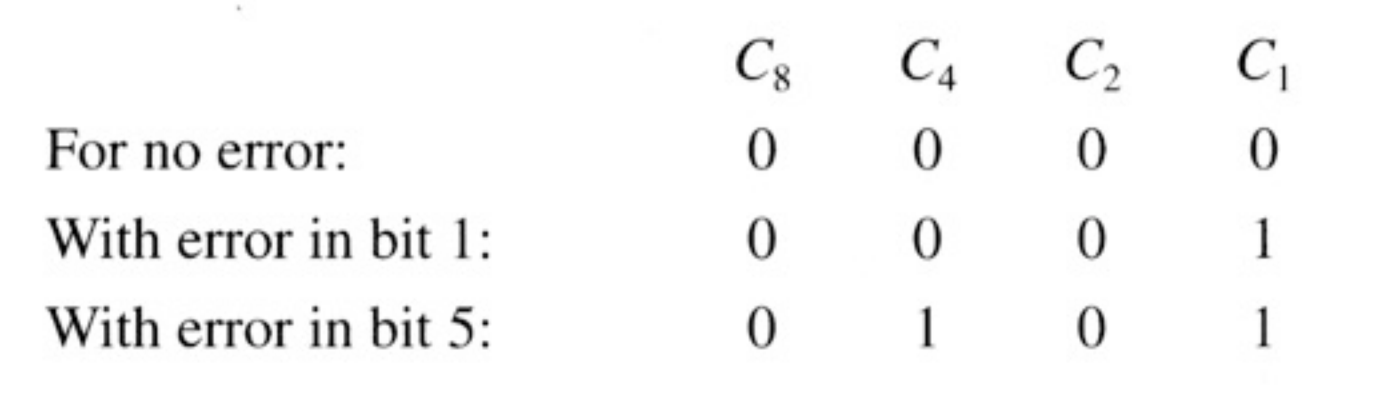


$$C_1 = \text{XOR of bits } (1, 3, 5, 7, 9, 11)$$

$$C_2 = \text{XOR of bits } (2, 3, 6, 7, 10, 11)$$

$$C_4 = \text{XOR of bits } (4, 5, 6, 7, 12)$$

$$C_8 = \text{XOR of bits } (8, 9, 10, 11, 12)$$



	C_8	C_4	C_2	C_1
For no error:	0	0	0	0
With error in bit 1:	0	0	0	1
With error in bit 5:	0	1	0	1

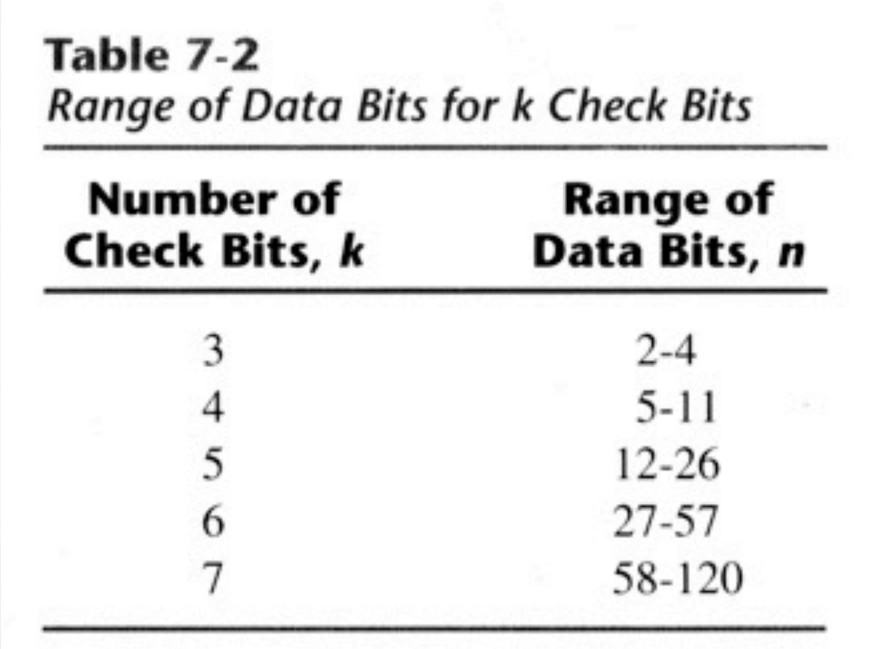
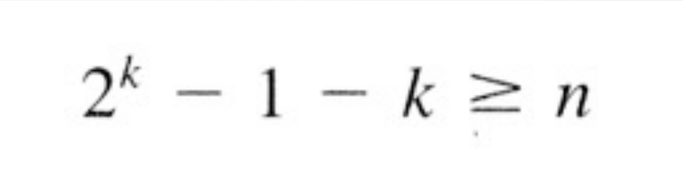


Table 7-2
Range of Data Bits for k Check Bits

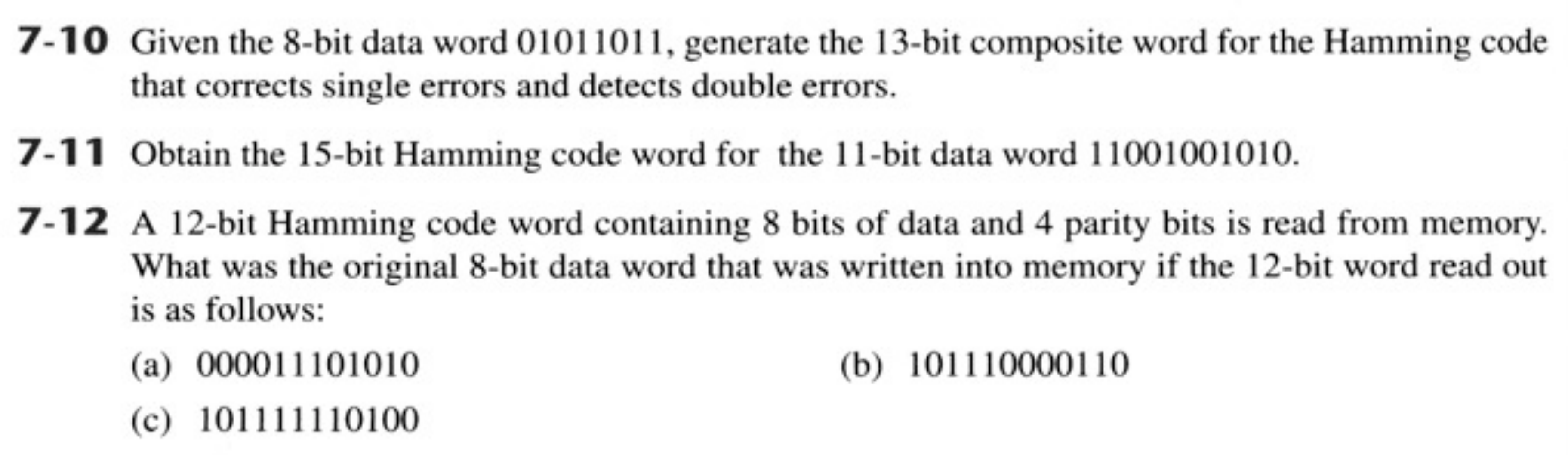
Number of Check Bits, *k*	Range of Data Bits, *n*
3	2-4
4	5-11
5	12-26
6	27-57
7	58-120



$$2^k - 1 - k \geq n$$

SINGLE-ERROR CORRECTION,

- To detect a double-error, add an aditional parity bit *P = XOR (all other bits)*
- 12-bit example: P_{13}=XOR(1...12)
- If
  - C=0 & P=0: no error
  - C≠0 & P=1: single error at bit indicated by C
  - C≠0 & P=0: double error detected
  - C=0 & P=1: error in P_{13}



7-10 Given the 8-bit data word 01011011, generate the 13-bit composite word for the Hamming code that corrects single errors and detects double errors.

7-11 Obtain the 15-bit Hamming code word for the 11-bit data word 11001001010.

7-12 A 12-bit Hamming code word containing 8 bits of data and 4 parity bits is read from memory. What was the original 8-bit data word that was written into memory if the 12-bit word read out is as follows:

(a) 000011101010 (b) 101110000110

(c) 101111110100

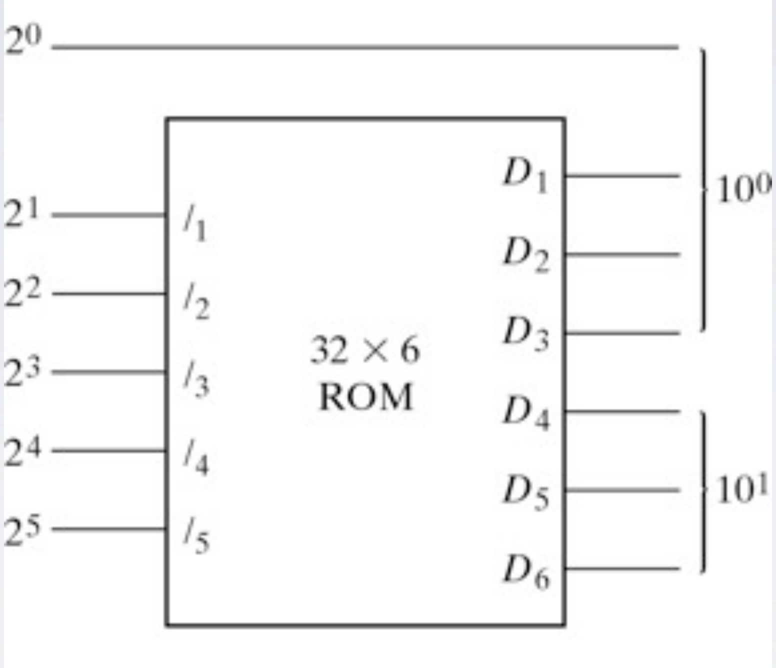


Fig. P7-17