EE1212-

Electronic System I

Memories

Capt Geeth Karunarathne

Read-Only Memory

A block diagram of a ROM is shown below. It consists of k address inputs and n data outputs.

The number of words in a ROM is determined from the fact that k address input lines are needed to specify 2^k words.

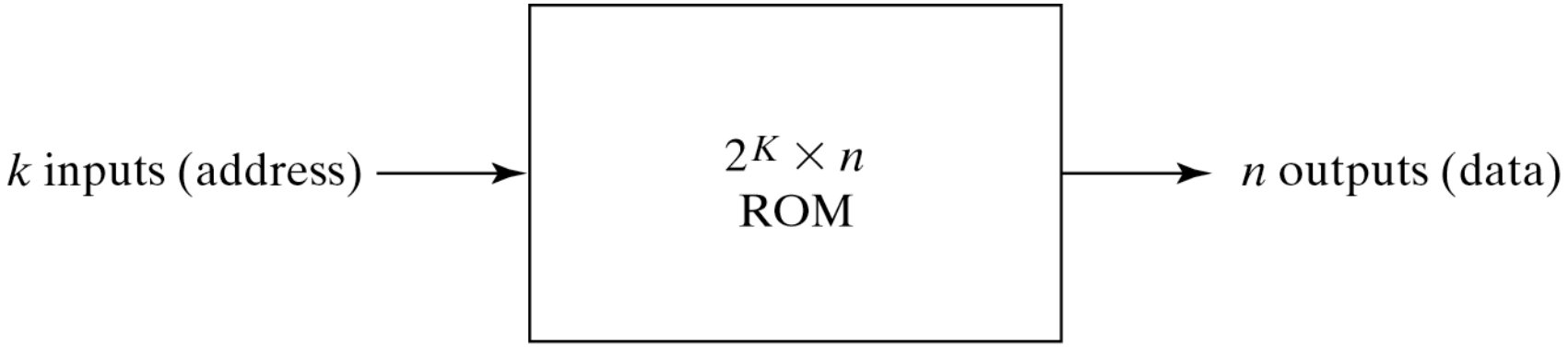


Fig. 7-9 ROM Block Diagram

Construction of ROM

Each output of the decoder represents a memory address.

Each OR gate must be considered as having 32 inputs.

A 2^k X n ROM will have an internal k X 2^k decoder and n OR gates.

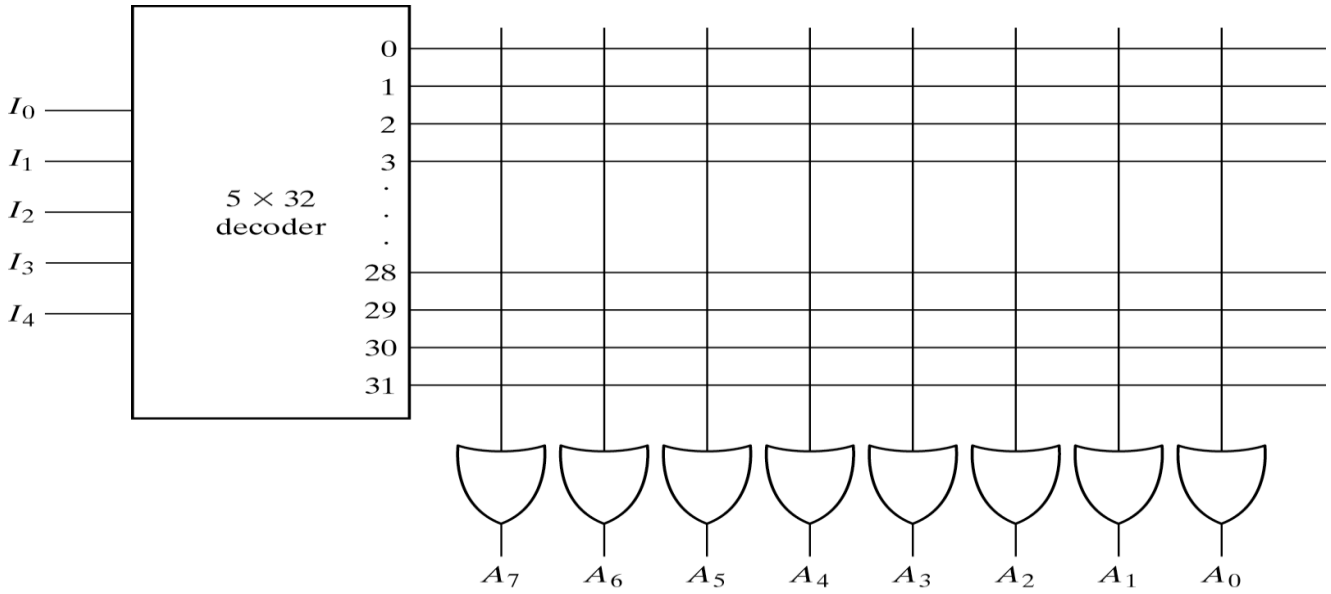


Fig. 7-10 Internal Logic of a 32 × 8 ROM

Truth table of ROM

A programmable connection between to lines is logically equivalent to a switch that can be altered to either be close or open.

Intersection between two lines is sometimes called a cross-point.

Table 7-3
ROM Truth Table (Partial)

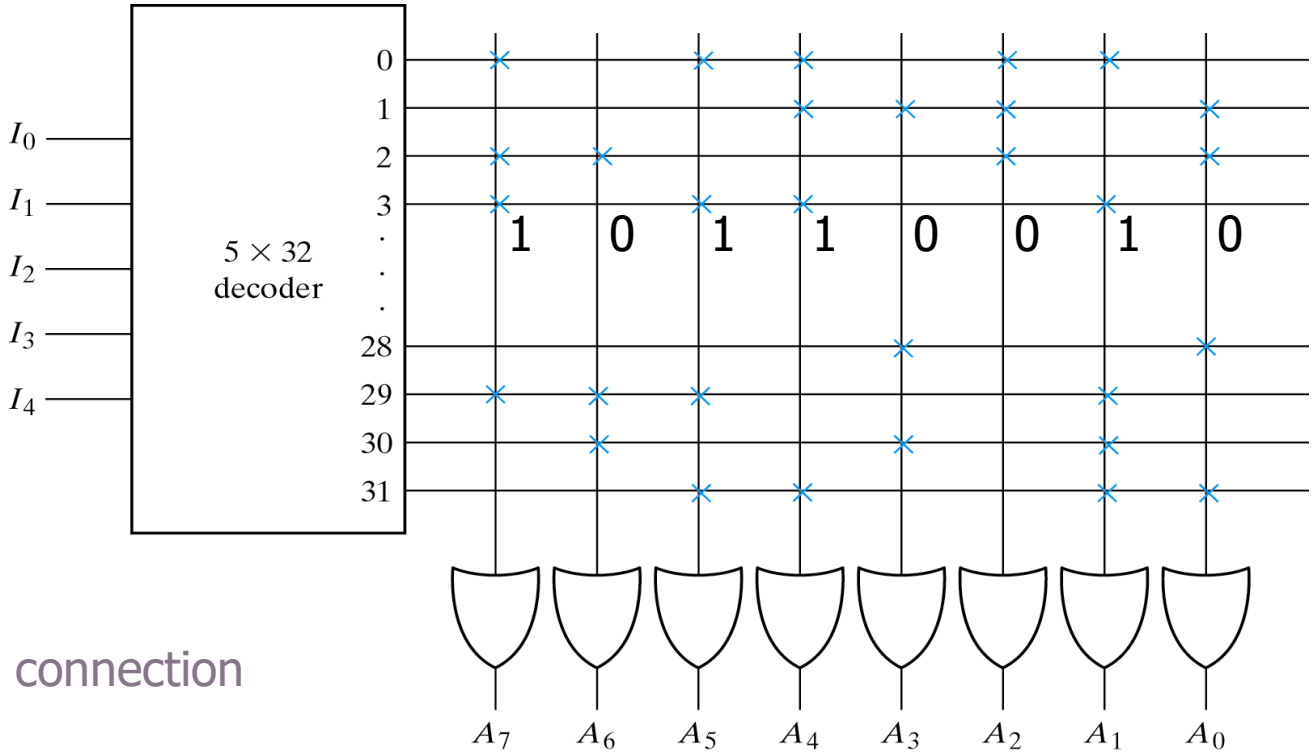
Inputs					Outputs							
I4	I3	I2	I1	I0	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮					⋮					
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

Programming the ROM

0 → no connection

1 → connection

Address 3 = 10110010 is permanent storage using fuse link



X : means connection

Combinational circuit implementation

The internal operation of a ROM can be interpreted in two way: First, a memory unit that contains a fixed pattern of stored words. Second, implements a combinational circuit.

Figure may be considered as a combinational circuit with eight outputs, each being a function of the five input variables.

$$A_7(I_4, I_3, I_2, I_1, I_0) = \Sigma(0,2,3\ldots,29)$$

Sum of minterms

In Table 7-3, output A_7

Example

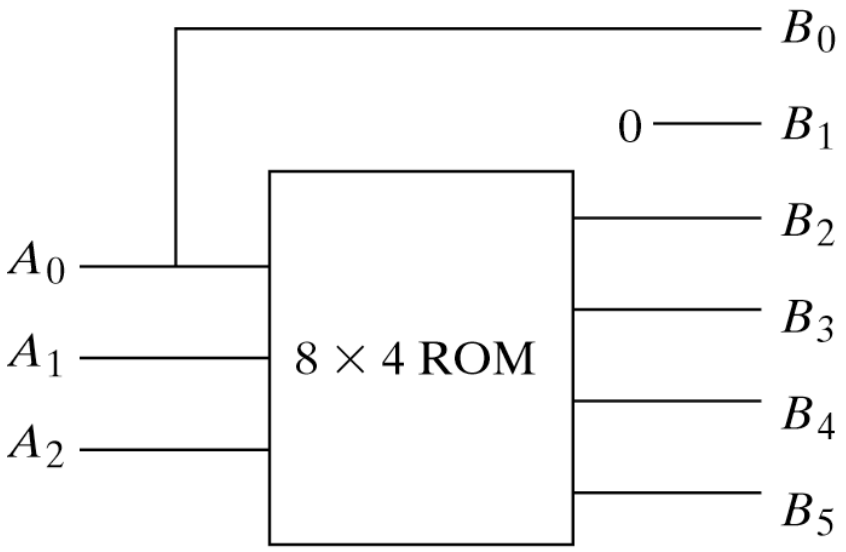
Design a combinational circuit using a ROM. The circuit accepts a 3-bit number and generates an output binary number equal to the square of the input number.

Derive truth table first

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

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

Example



(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

Types of ROMs

The required paths in a ROM may be programmed in four different ways.

1. Mask programming: fabrication process
2. Read-only memory or PROM: blown fuse /fuse intact
3. Erasable PROM or EPROM: placed under a special ultraviolet light for a given period of time will erase the pattern in ROM.
4. Electrically-erasable PROM(EEPROM): erased with an electrical signal instead of ultraviolet light.

Combinational PLDs

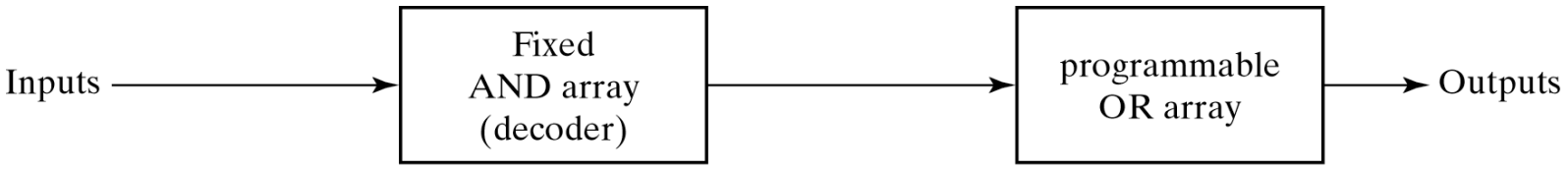
A combinational PLD is an integrated circuit with programmable gates divided into an AND array and an OR array to provide an AND-OR sum of product implementation

PROM: fixed AND array constructed as a decoder and programmable OR array.

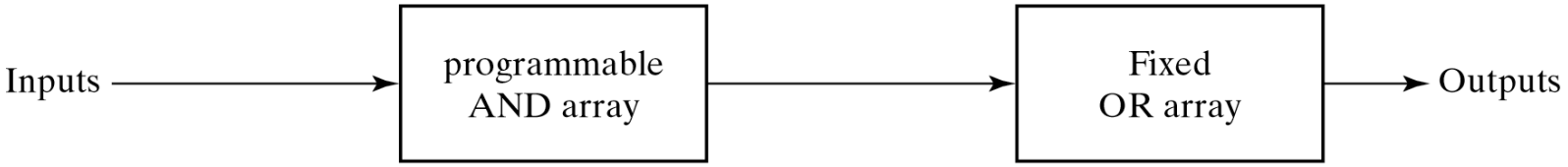
PAL: programmable AND array and fixed OR array.

PLA: both the AND and OR arrays can be programmed.

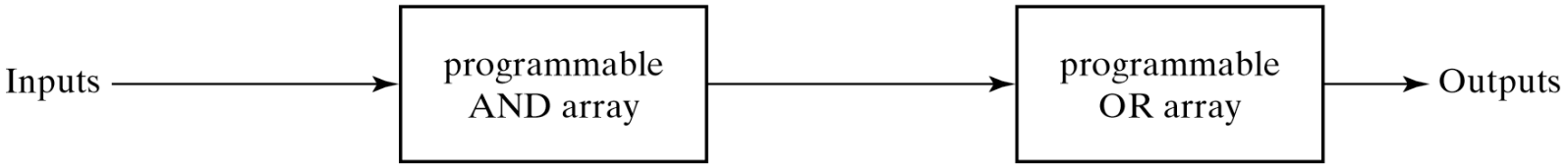
Combinational PLDs



(a) Programmable read-only memory (PROM)



(b) Programmable array logic (PAL)



(c) Programmable logic array (PLA)

Programmable Logic Array

The decoder in PROM is replaced by an array of AND gates that can be programmed to generate any product term of the input variables.

The product terms are then connected to OR gates to provide the sum of products for the required Boolean functions.

The output is inverted when the XOR input is connected to 1 (since $x \oplus 1 = x'$). The output doesn't change and connect to 0 (since $x \oplus 0 = x$).

PLA

F1 = AB'+AC+A'BC'

F2 = (AC+BC)'

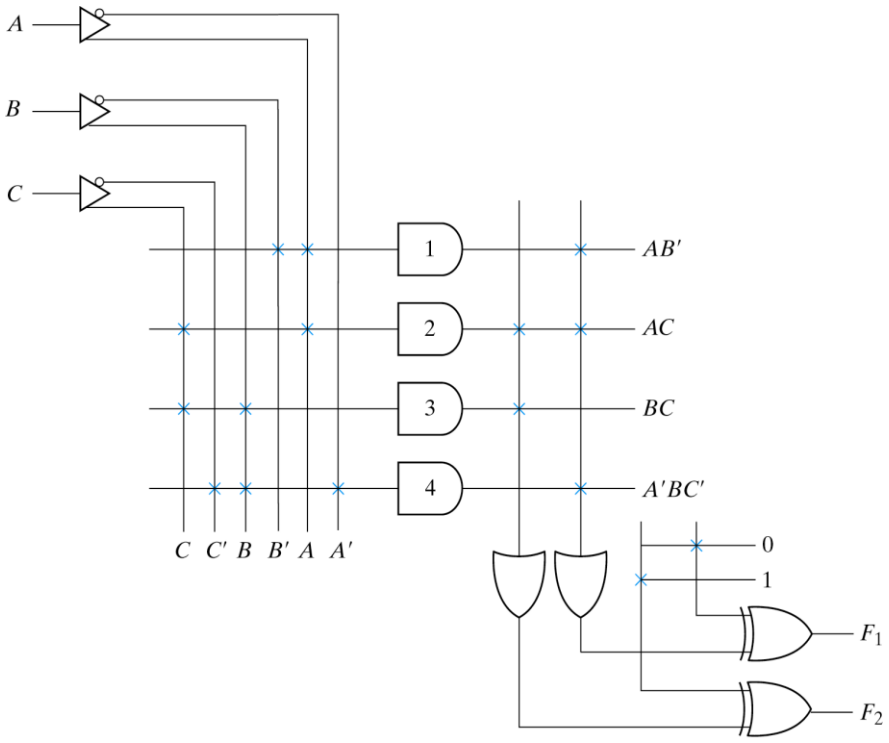


Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

Table 7-5
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	–

Programming Table

1. First: lists the product terms numerically
2. Second: specifies the required paths between inputs and AND gates
3. Third: specifies the paths between the AND and OR gates
4. For each output variable, we may have a T(ture) or C(complement) for programming the XOR gate

Simplification of PLA

Careful investigation must be undertaken in order to reduce the number of distinct product terms, PLA has a finite number of AND gates.

Both the true and complement of each function should be simplified to see which one can be expressed with fewer product terms and which one provides product terms that are common to other functions.

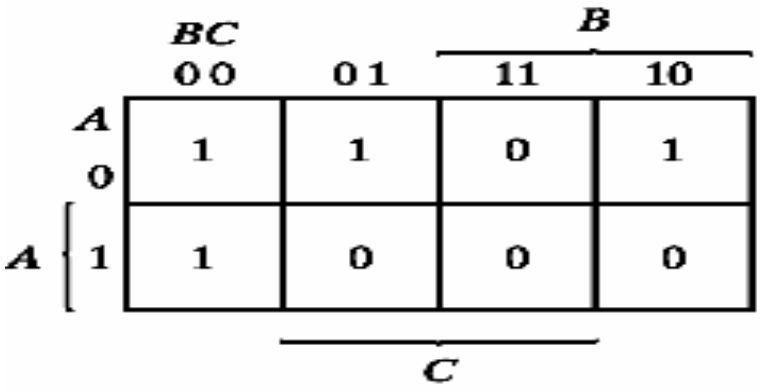
Example

Implement the following two Boolean functions with a PLA:

$$F_1(A, B, C) = \sum(0, 1, 2, 4)$$

$$F_2(A, B, C) = \sum(0, 5, 6, 7)$$

The two functions are simplified in the maps of Fig.7-15

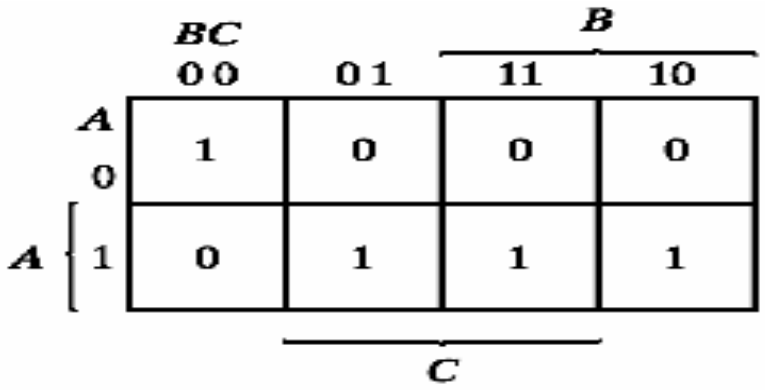


$F_1 = A'B' + A'C' + B'C'$

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

1 elements

0 elements



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

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

PLA table by simplifying the function

Both the true and complement of the functions are simplified in sum of products.

We can find the same terms from the group terms of the functions of F_1, F_1', F_2 and F_2' which will make the minimum terms.

F1 = (AB + AC + BC)'

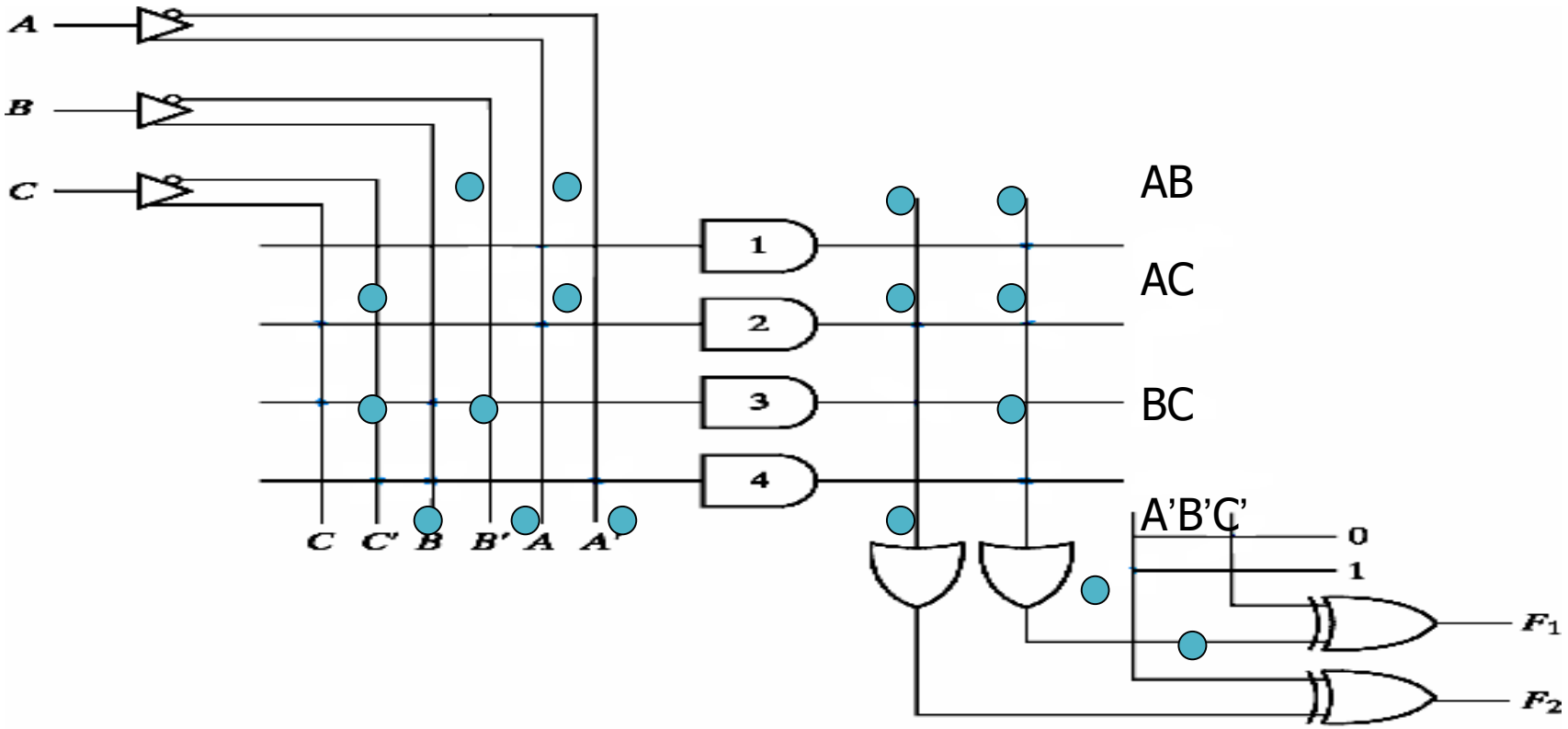
F2 = AB + AC + A'B'C'

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

Fig. 7-15 Solution to Example 7-2

PLA implementation



Programmable Array Logic

The PAL is a programmable logic device with a fixed OR array and a programmable AND array.

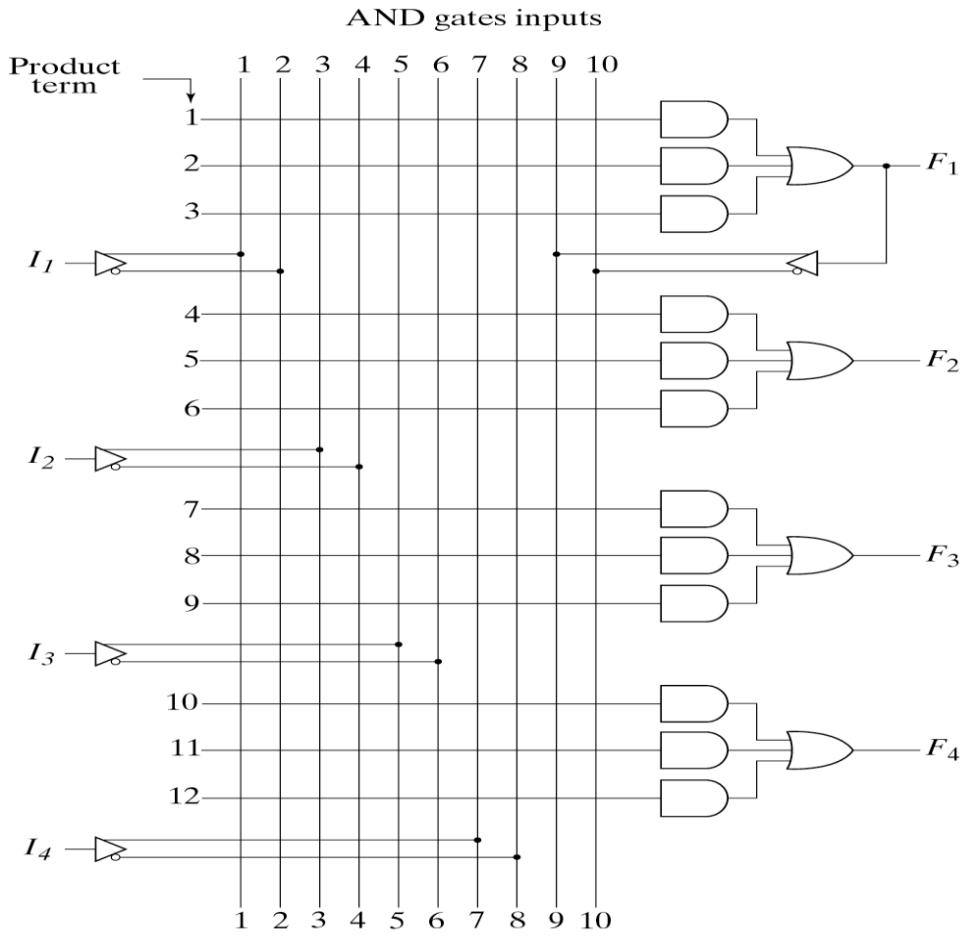


Fig. 7-16 PAL with Four Inputs, Four Outputs, and Three-Wide AND-OR Structure

PAL

When designing with a PAL, the Boolean functions must be simplified to fit into each section.

Unlike the PLA, a product term cannot be shared among two or more OR gates. Therefore, each function can be simplified by itself without regard to common product terms.

The output terminals are sometimes driven by three-state buffers or inverters.

Example

$w(A, B, C, D) = \sum(2, 12, 13)$

$x(A, B, C, D) = \sum(7, 8, 9, 10, 11, 12, 13, 14, 15)$

$y(A, B, C, D) = \sum(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)$

$z(A, B, C, D) = \sum(1, 2, 8, 12, 13)$

Simplifying the four functions as following Boolean functions:

$w = ABC' + A'B'CD'$

$x = A + BCD$

$y = A'B + CD + B'D'$

$z = ABC' + A'B'CD' + AC'D' + A'B'C'D = w + AC'D' + A'B'C'D$

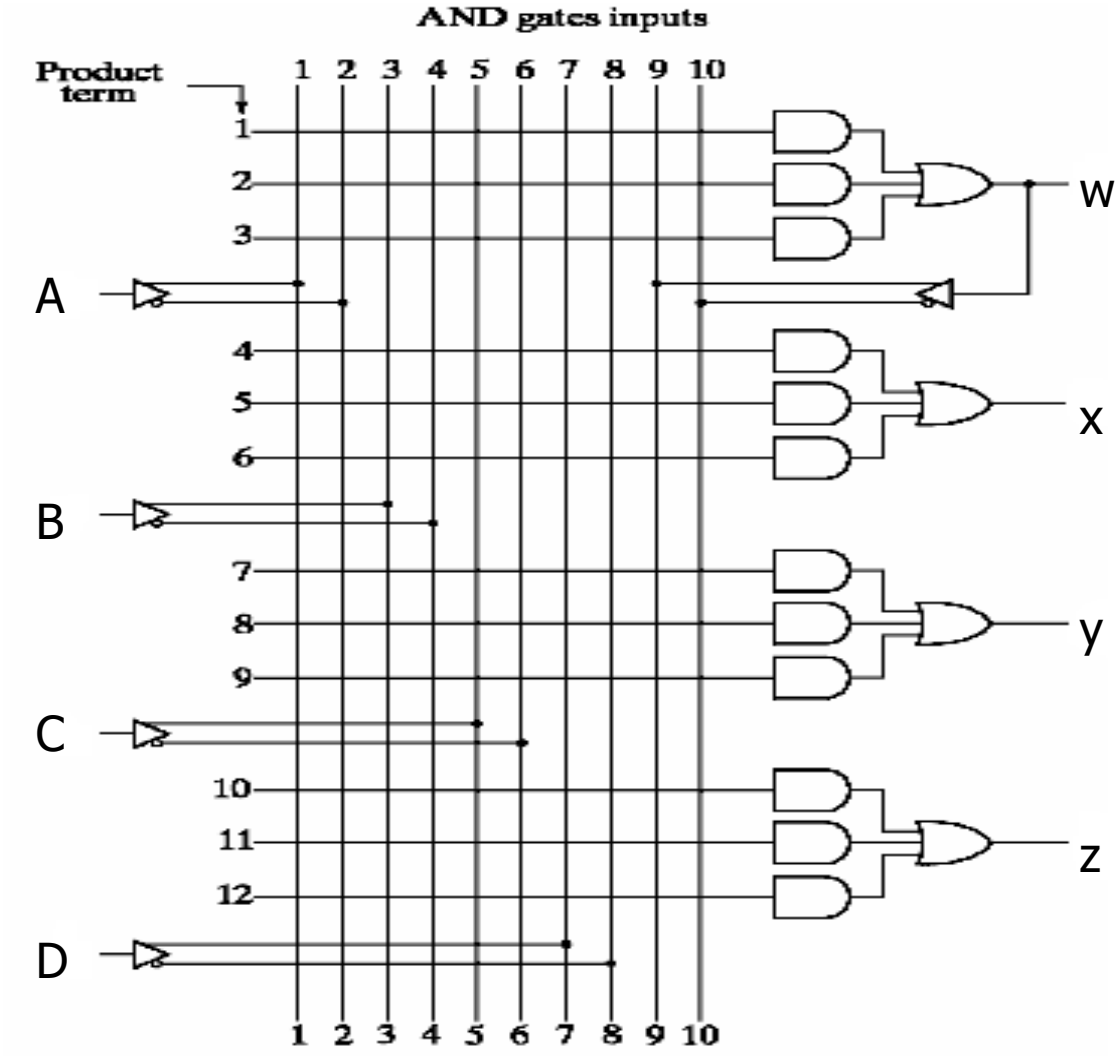
PAL Table

z has four product terms, and we can replace by w with two product terms, this will reduce the number of terms for z from four to three.

Table 7-6
PAL Programming Table

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	–	$+ A'B'C'D$

PAL implementation



Fuse map for example

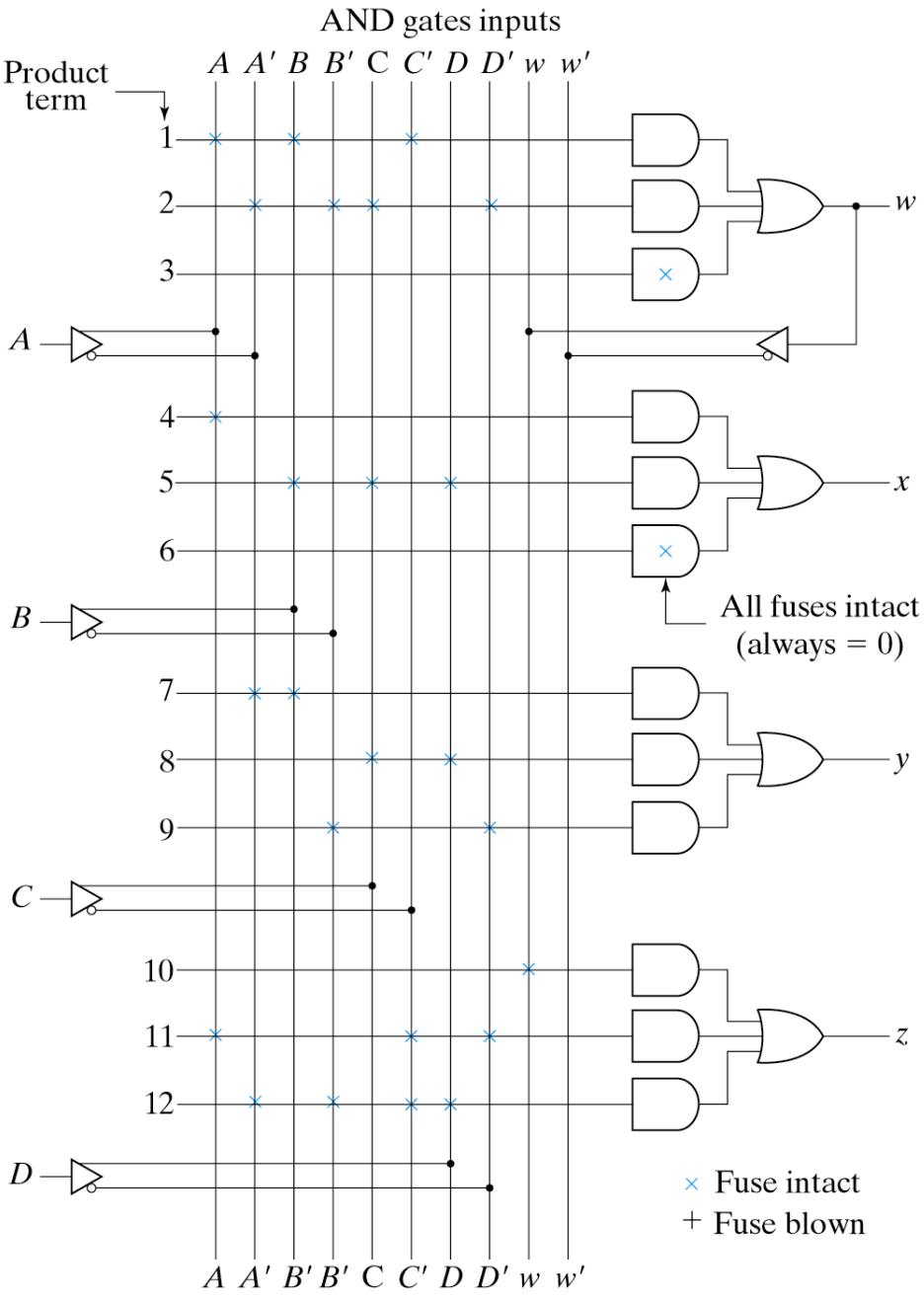


Fig. 7-17 Fuse Map for PAL as Specified in Table 7-6

Sequential Programmable Devices

Sequential programmable devices include both gates and flip-flops.

There are several types of sequential programmable devices, but the internal logic of these devices is too complex

Sequential Programmable Devices

1. Sequential (or simple) Programmable Logic Device (SPLD)
2. Complex Programmable Logic Device (CPLD)
3. Field Programmable Gate Array (FPGA)

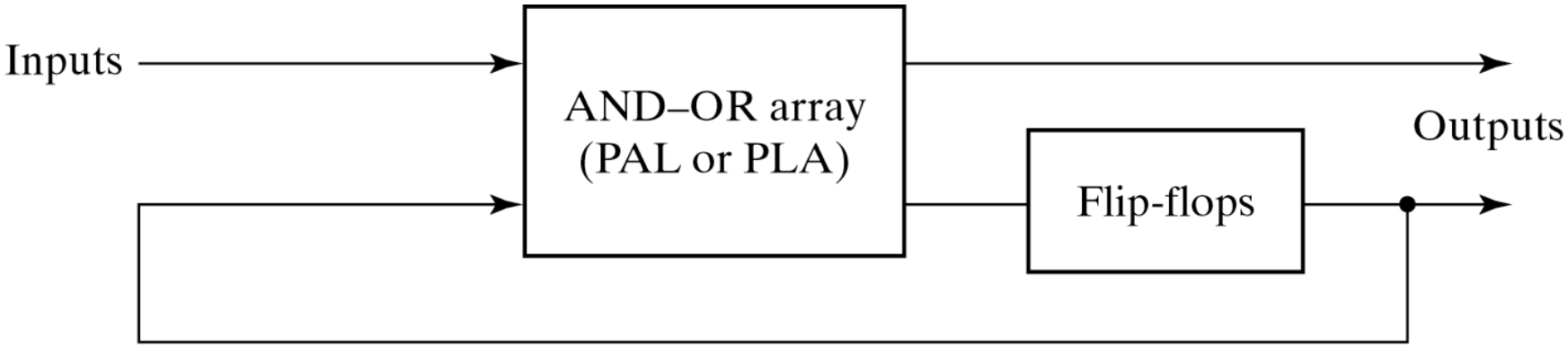


Fig. 7-18 Sequential Programmable Logic Device