Memory and Programmable Logic

Random-Access Memory

- A memory unit stores binary information in groups of bits called words.

  1 byte = 8 bits

1 word = 2 bytes

2 The communication between a memory and its environment is achieved through data input and output lines, address selection lines, and control lines that specify the direction of transfer.

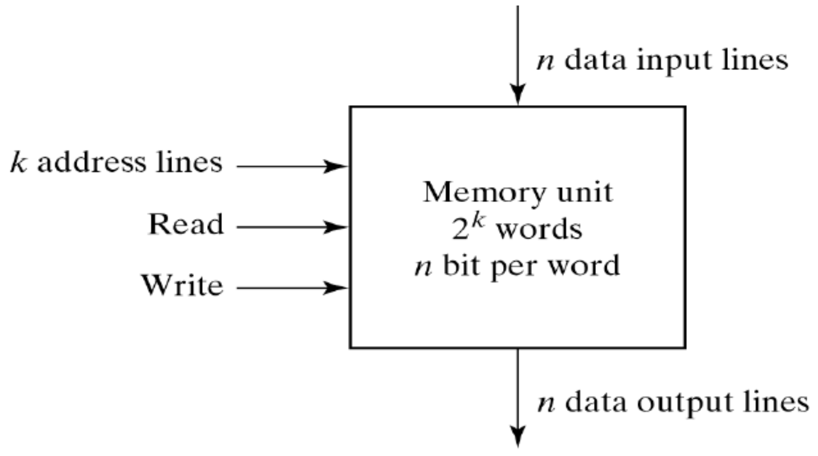


Fig. 7-2 Block Diagram of a Memory Unit

Content of a memory:

- Each word in memory is assigned an identification number, called an address, starting from 0 up to 2^k-1, where k is the number of address lines.
- The number of words in a memory with one of the letters K=2^{10}, M=2^{20}, or G=2^{30}.

  64K = 2^{16} 2M = 2^{21}

  4G = 2^{32}

Write and Read operations :

- Transferring a new word to be stored into memory:

1. Apply the binary address of the desired word to the address lines.
2. Apply the data bits that must be stored in memory to the data input lines.
3. Activate the write input.

- Transferring a stored word out of memory:

1. Apply the binary address of the desired word to the address lines.
2. Activate the read input.

- Commercial memory sometimes provide the two control inputs for reading and writing in a somewhat different configuration in table 7-1.

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

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 $\times$ 16 Memory

Types of memories:

- In random-access memory, the word locations may be thought of as being separated in space, with each word occupying one particular location.
- In sequential-access memory, the information stored in some medium is not immediately accessible, but is available only certain intervals of time. A magnetic disk or tape unit is of this type.

There are two basic types of RAM :

- (i) Dynamic Ram
- (ii) Static RAM

- Dynamic RAM : loses its stored information in a very short time (for milli sec.) even when power supply is on. D-RAM's are cheaper & lowe

Types of ROMs:

ROM : Read only memory: Its non volatile memory, ie, the information stored in it, is not lost even if the power supply goes off. It's used for the permanent storage of information. It also posses random access property. Information can not be written into a ROM by the users/programmers. In other words the contents of ROMs are decided by the manufactures.

The following types of ROMs an listed below :

(i) PROM : It's programmable ROM. Its contents are decided by the user. The user can store permanent programs, data etc in a PROM. The data is fed into it using a PROM programs.

(ii) EPROM : An EPROM is an erasable PROM. The stored data in EPROM's can be erased by exposing it to UV light for about 20 min. It's not easy to erase it because the EPROM IC has to be removed from the computer and exposed to UV light. The entire data is erased and not selected portions by the user. EPROM's are cheap and reliable.

(iii) EEPROM (Electrically Erasable PROM) : The chip can be erased & reprogrammed on the board easily byte by byte. It can be erased with in a few milliseconds. There is a limit on the number of times the EEPROM's can be reprogrammed, i.e.; usually around 10,000 times.

- 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.
- 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.

READ ONLY MEMORY

A ROM is essentially a memory device in which permanent binary information is stored. The binary information must be specified by the designer and is then embedded in the unit to form the required interconnection pattern. Once the pattern is established it stays within the unit even when power is turned off and on again.

A block diagram of ROM is shown in the Figure 1. It consists of k inputs and n outputs. The inputs provide the address for the memory and the outputs give the data bits of the stored word which is selected by the address. The number of words in a ROM is determined from the fact that k address input lines are needed to specify 2^k words.

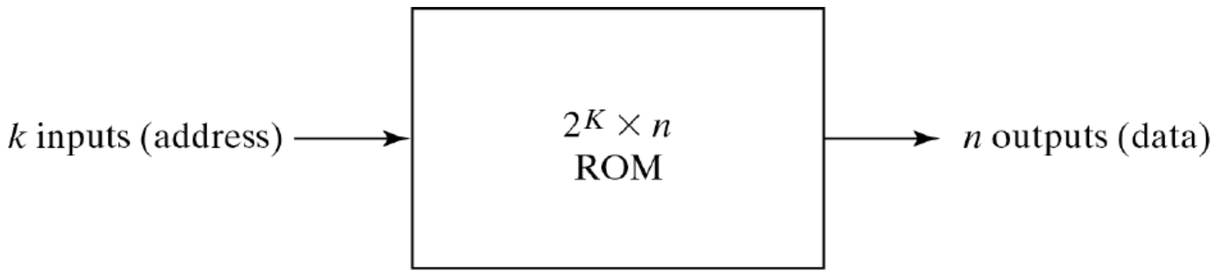


Fig. 7-9 ROM Block Diagram

ROM does not have data inputs because it does not have a write operation.

Consider for example a 32 x 8 ROM. The unit consists of 32 words of 8 bits each. There are five input lines that form the binary numbers from 0 through 31 for the address. The Figure 2 shows the internal logic construction of the ROM. The five inputs are decoded into 32 distinct outputs by means of a 5 x 32 decoder. Each output of the decoder represents a memory address. The 32 outputs of the decoder are connected to each of the eight OR gates.
The diagram shows the array logic convention used in complex circuits . Each OR gate must be considered as having 32 inputs. Each output of the decoder is connected to one of the inputs of each OR gate. Since each OR gate has 32 input connections and there are 8 OR gates, the ROM contains 32 x 8 = 256 internal connections.

In general, a 2^k x n ROM will have an internal k x 2^k decoder and n OR gates. Each OR gate has 2^k inputs, which are connected to each of the outputs of the decoder.

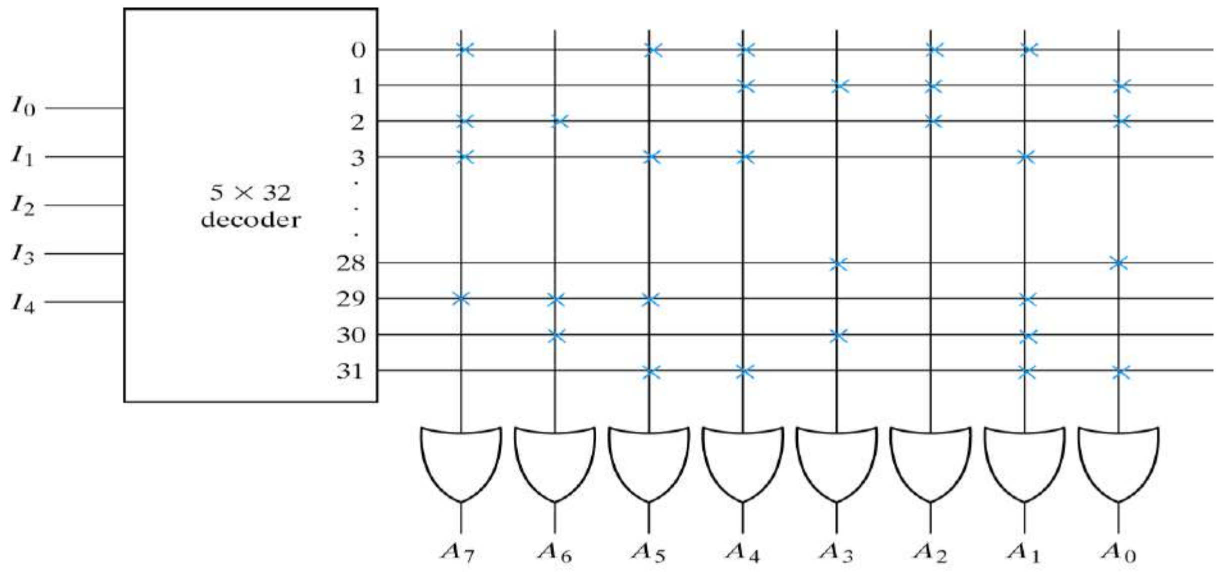
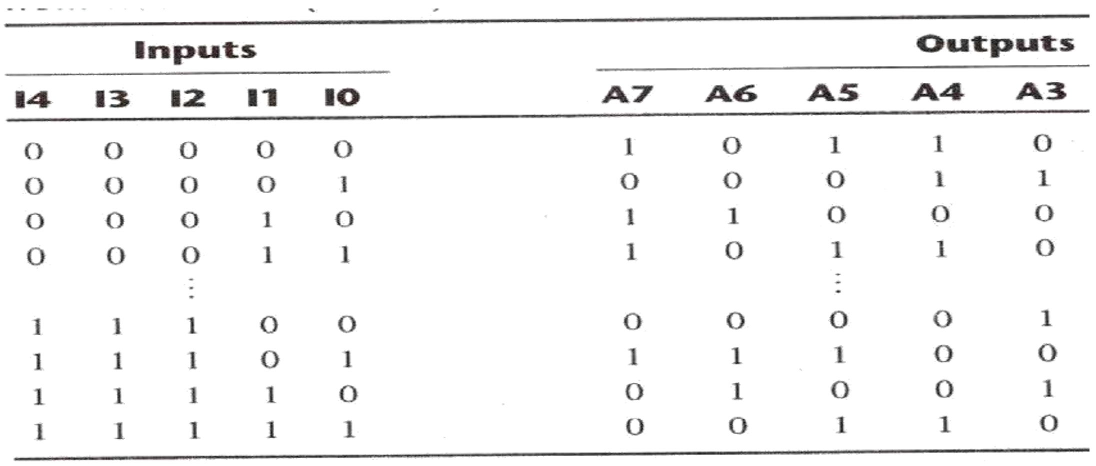


Fig. 7-11 Programming the ROM According to Table 7-3



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

Every 0 listed in the truth table specifies a no connection and every 1 listed specifies a path that is obtained by a connection. The four 0's in the word are programmed by blowing the fuse links between output 3 of the decoder and the inputs of the OR gates associated with outputs A6 , A3, A2 and A0. The four 1's in the word are marked in the diagram with a X to denote a connection in place of a dot used for permanent connection in logic diagrams.

When the input of the ROM is 00011, all the outputs of the decoder are 0 except for output 3, which is at logic 1. The signal equivalent to logic 1 at decoder output 3 propagates through the connections to the OR gate outputs of A7 , A5, A4 and A1.The other four outputs remain at 0. The result is that the stored word 10110010 is applied to the eight data outputs.

COMBINATIONAL PLDs

The PROM is a combinational programmable logic device (PLD). 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. There are three major types of combinational PLDs and they differ in the placement of the programmable connections in the AND-OR array. The Figure 3 shows the configuration of three PLDs.

The PROM has a fixed AND array constructed as a decoder and programmable OR array. The programmable OR gates implement the Boolean functions in sum of minterms.

The programmable array logic (PAL) has a programmable AND array and a fixed OR array. The AND gates are programmed to provide the product terms for the Boolean functions which are logically summed in each OR gate. The most flexible PLD is the programmable logic array (PLA) where both AND and OR arrays can be programmed. The product terms in the AND array may be shared by any OR gate to provide the required sum of products implementation.

Fig 3 Basic configuration of three PLDs

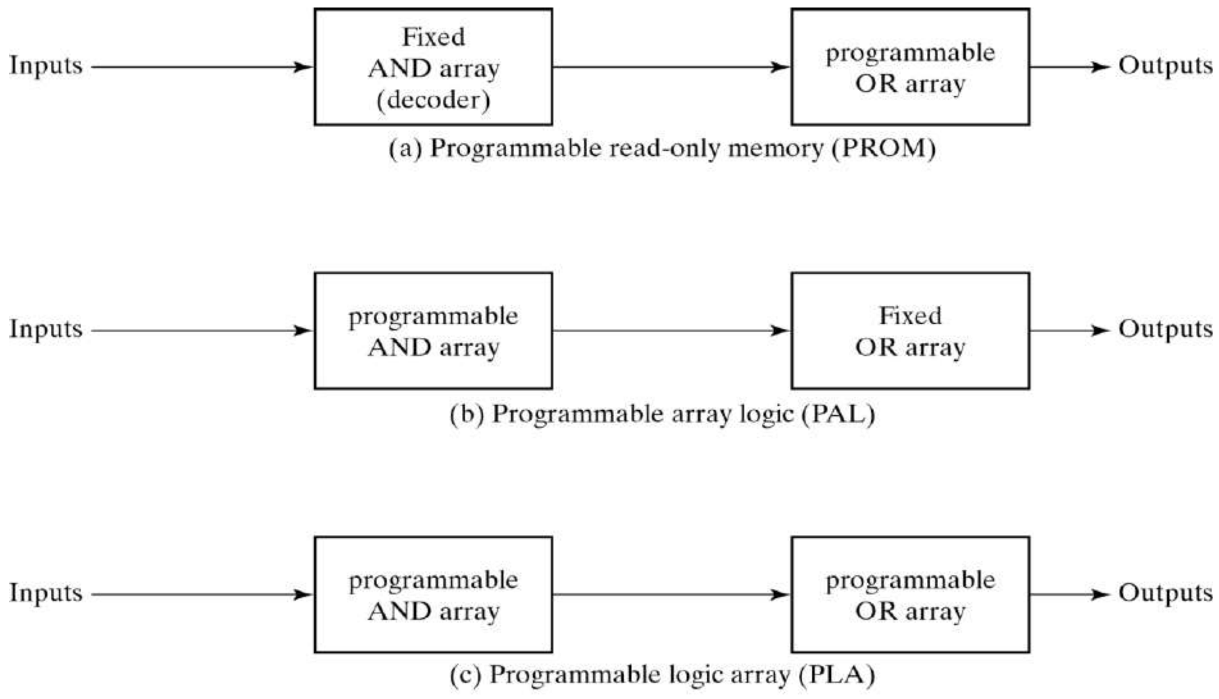


Fig. 7-13 Basic Configuration of Three PLDs

Types of ROMs:

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

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

<u>Example of ROM :</u>

- **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

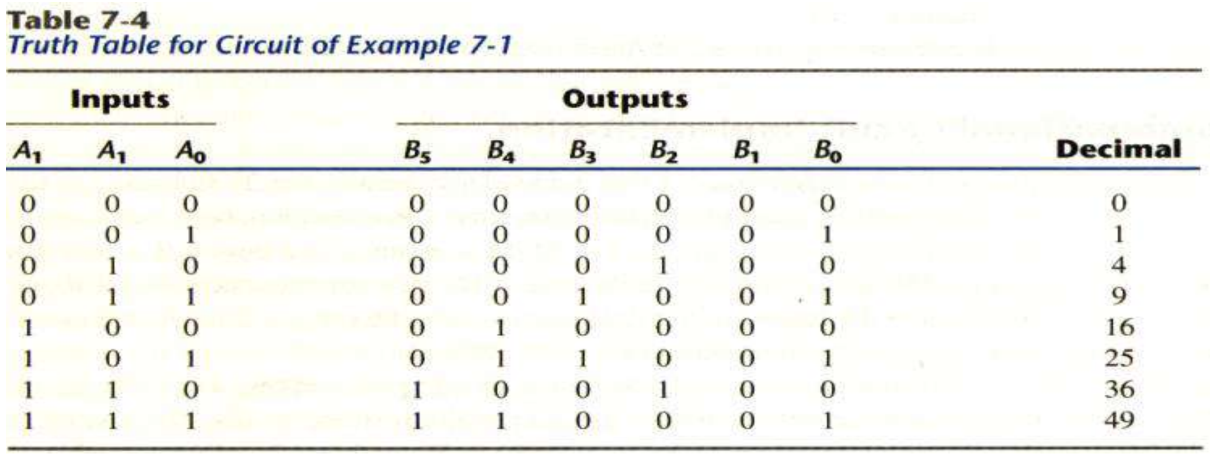
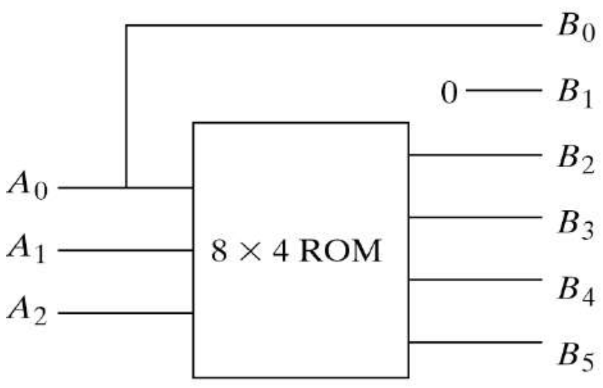
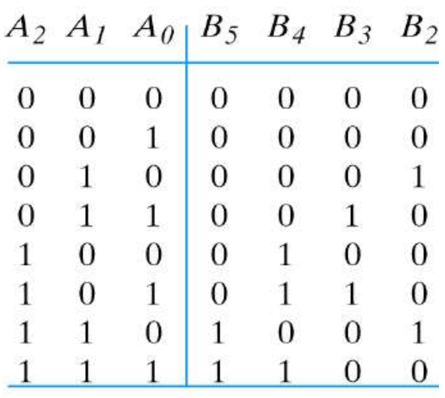


Table 7-4
Truth Table for Circuit of Example 7-1

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



(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

Programmable Logic Array(PLA):

The PLA is similar to PROM in concept except that PLA does not provide full decoding of the variable and does not generate all the minterms . The decoder 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.

- Fig.7-14, 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⊕1 = x'). The output doesn't change and connect to 0 (since x⊕0 = x).
    - F1 = AB'+AC+A'BC'
    - F2 = (AC+BC)'

The diagram uses the array logic graphic symbols for complex circuits. Each input goes through a buffer and an inverter shown in the diagram with a composite graphic symbol which has both the true and complement outputs. Each input and its complement are connected to the inputs of each AND gate as indicated by the intersections between the vertical and horizontal lines. The outputs of the AND gates are connected to the inputs of each OR gate.

- The output of the OR gates goes to an XOR gate where the other input can be programmed to receive a signal equal to either logic 1 or 0. The output is inverted when the XOR input is connected to 1. The output does not change when the XOR input is connected to 0 .

- The product terms generated in each AND gate are listed along the output of the gate in the diagram. The product term is determined from the inputs whose crosspoints are connected and marked with a X . The output of an OR gate gives the logic sum of the selected product terms. The output may be complemented or left in its true form depending on the connection for one of the XOR gate inputs.
- The programming table that specifies the PLA of Fig.4 is listed in Table 2. The PLA programming table consists of three sections .The first section lists the product terms numerically. The second section specifies the required paths between inputs and AND gates. The third section specifies the paths between AND and OR gates.

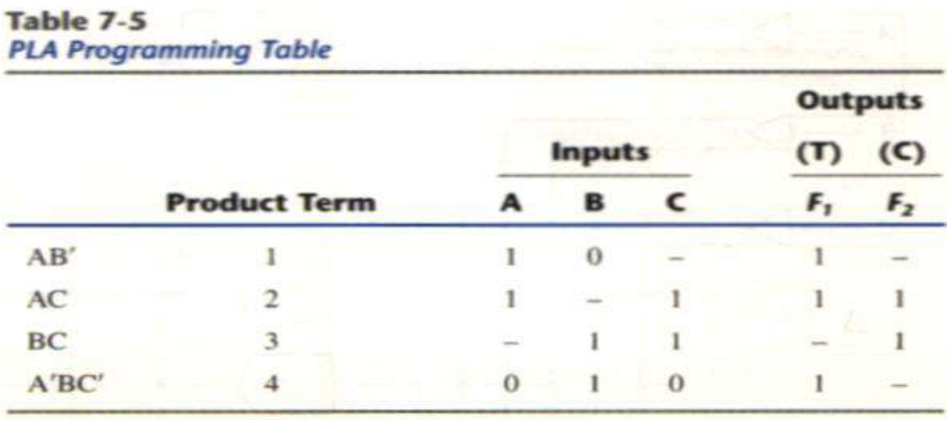


Table 7-5
PLA Programming Table

		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	–

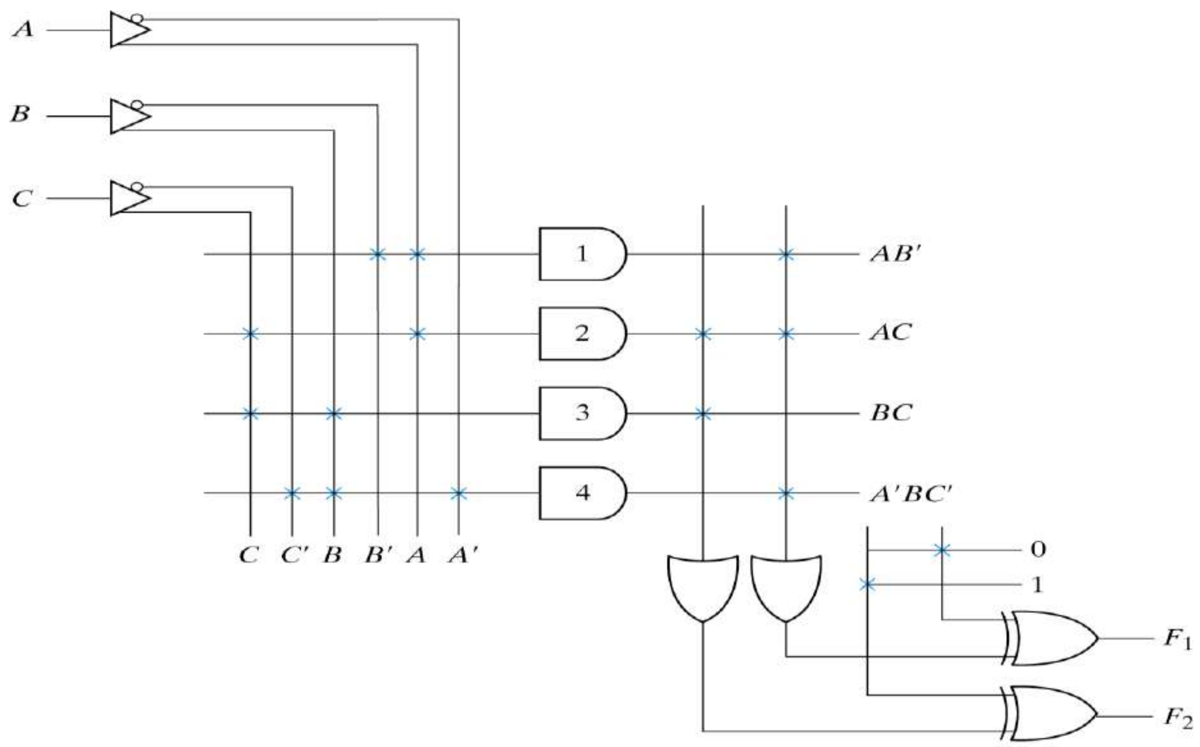


Fig. 7-14 PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

A T output dictates that the other input of the corresponding XOR gate be connected to 0,and a C specifies a connection to 1.The size of a PLA is specified by the number of inputs ,the number of product terms and the number of outputs. When designing a digital system with a PLA there is no need to show the internal connections of the units. All that is needed is a PLA programming table from which the PLA can be programmed to supply the required logic.

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.

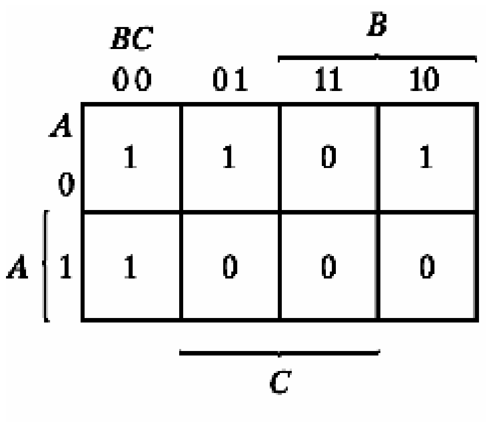
Example 7-2:

Implement the following two Boolean functions with a PLA:

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

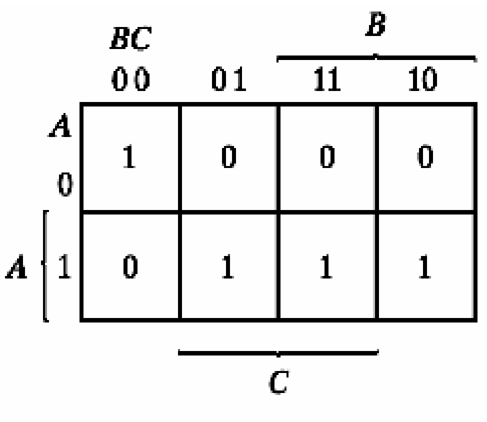
$$F_2(A, B, C) = \sum(0, 5, 6, 7)$$

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



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

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



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

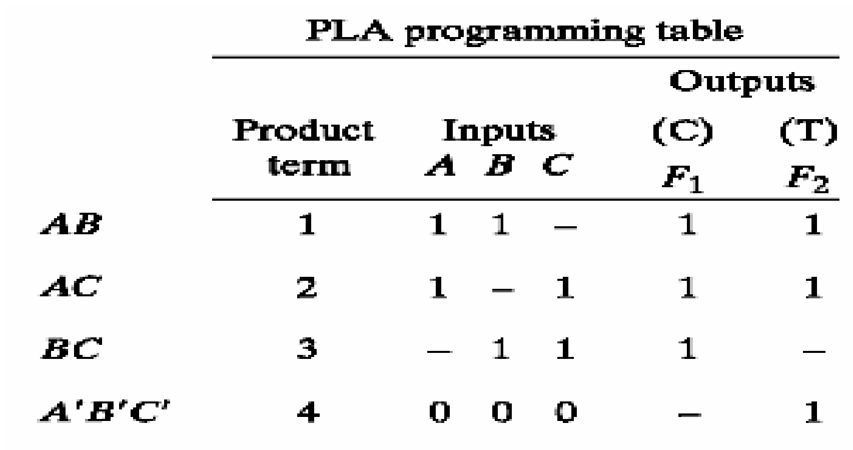
$$F_2 = (A'C + A'B + AB'C')'$$

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.

$$F1 = (AB + AC + BC)'$$

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

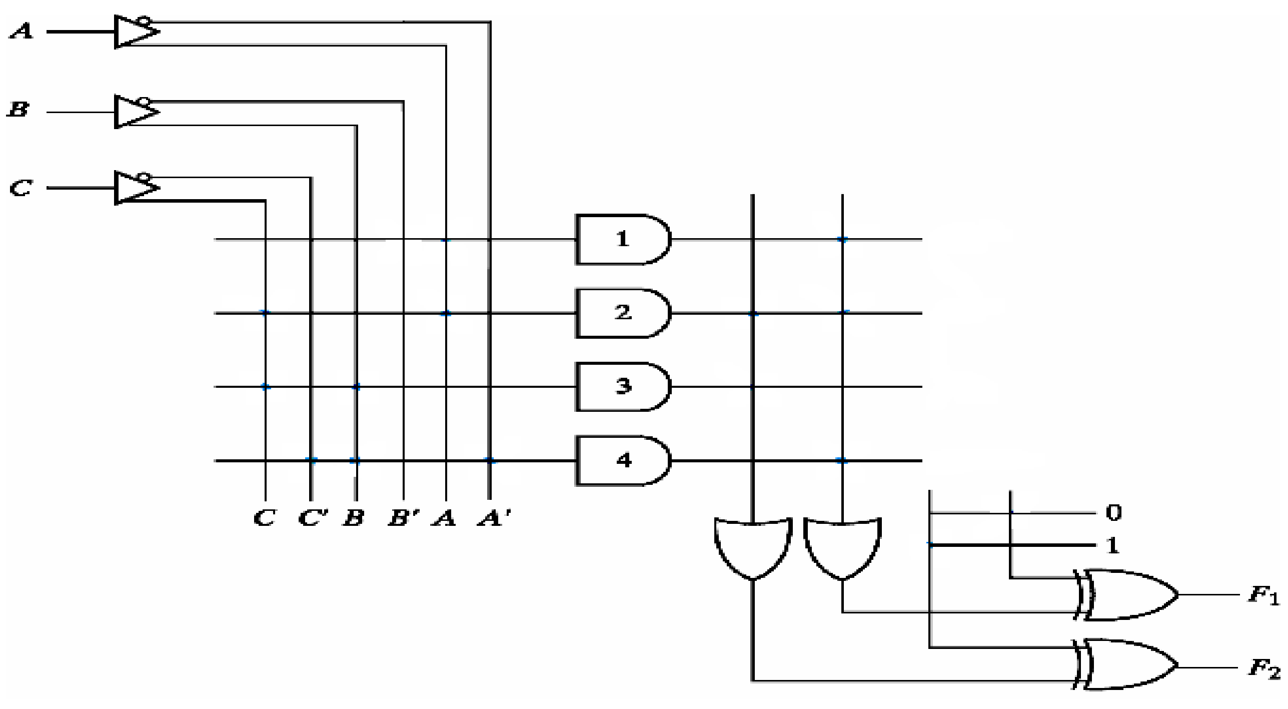


PLA programming table

	Product term	Inputs			Outputs (C)	Outputs (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

PLA implementation:



Note that output F1 is the true output even though a c is marked over it in the table. This is because F1 is generated with an AND-OR circuit and is available at the output of the OR gate. The XOR gate complements the function to produce the true F1 output.

The combinational circuit used in this example is too simple for implementing with a PLA.

PROGRAMMABLE ARRAY LOGIC (PAL)

The programmable array logic (PAL) is a programmable logic device with a fixed OR array and a programmable AND array. Because only the AND gates are programmable the PAL is easier to program, but is not as flexible as the PLA.
Figure 5 shows the logic configuration of a typical PAL.

It has four inputs and four outputs. Each input has a buffer inverter gate and each output is generated by a fixed OR gate. There are four sections in the unit, each being composed of a three wide AND-OR array.

Each AND gate has 10 programmable input connections. This is shown in the diagram by 10 vertical lines intersecting each horizontal line. The horizontal line symbolizes the multiple input configuration of the AND gate. One of the outputs is connected to a buffer inverter gate and then fed back into two inputs of the AND gates.

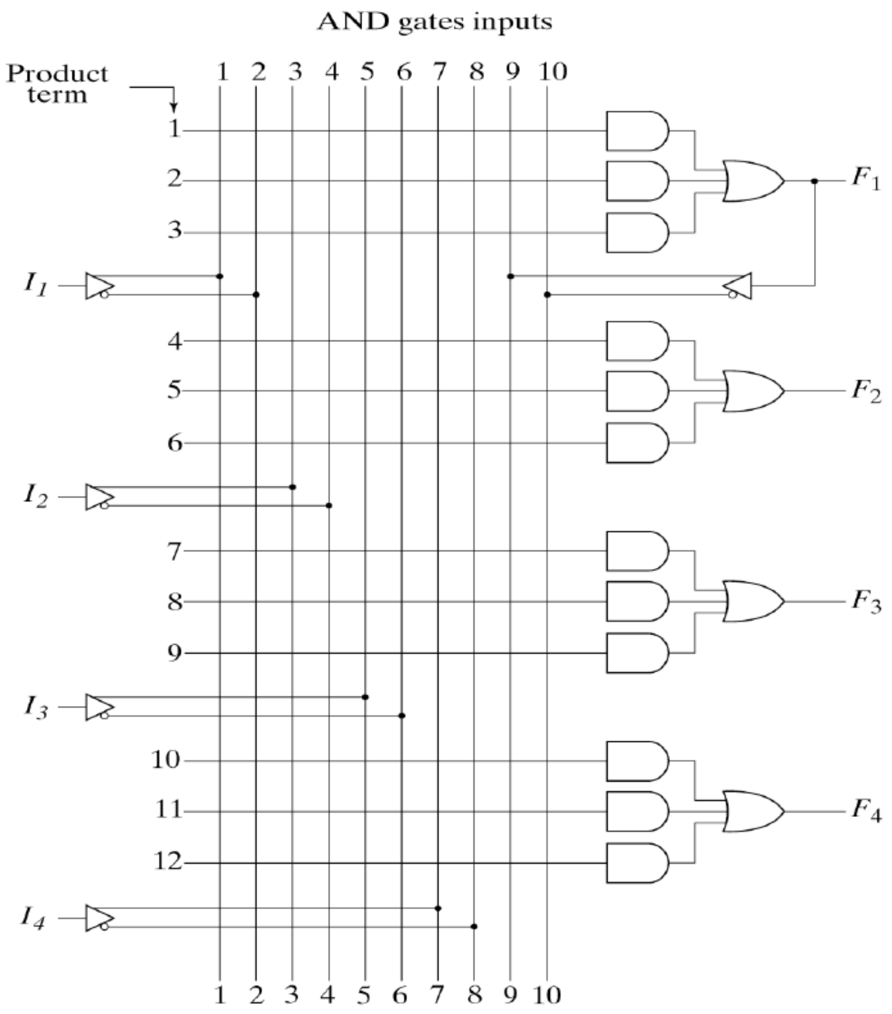


Fig. 7-16 PAL with Four Inputs, Four Outputs, and Three-Wide AND-OR Structure

- 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.

PAL Example :

w(A, B, C, D) = ∑(2, 12, 13)

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

y(A, B, C, D) = ∑(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15) z(A, B, C, D) = ∑(1, 2, 8, 12, 13)

Simplifying the four functions as following Boolean functions:

w = ABC’ + A’B’CD’

x = A + BCD

w = A’B + CD + B’D’

w = 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.
- The table is divided into four sections with three product terms. The first two sections need only two product terms to implement the Boolean function. The last section for output z needs four product terms. Using the output from w, we can reduce the function to three terms.

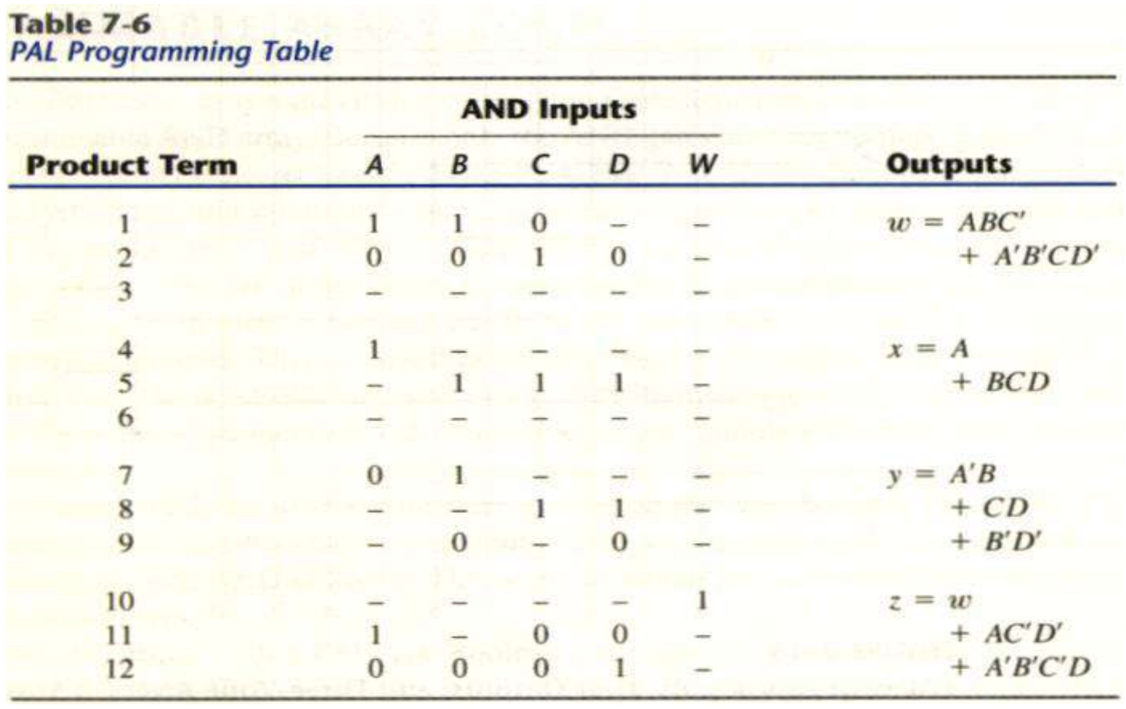
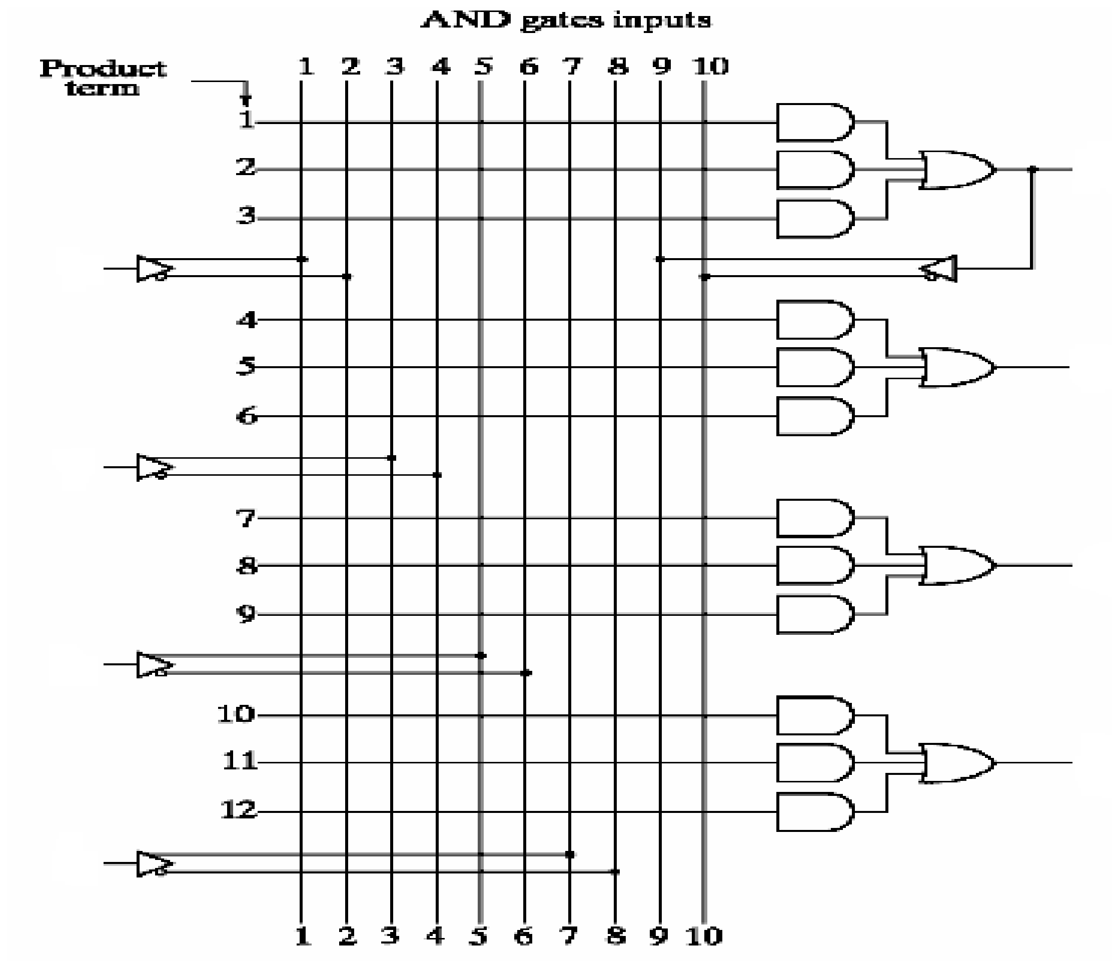


Table 7-6
PAL Programming Table

	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	–	$+ A'B'C'D$

PAL implementation :



Fuse map for example :

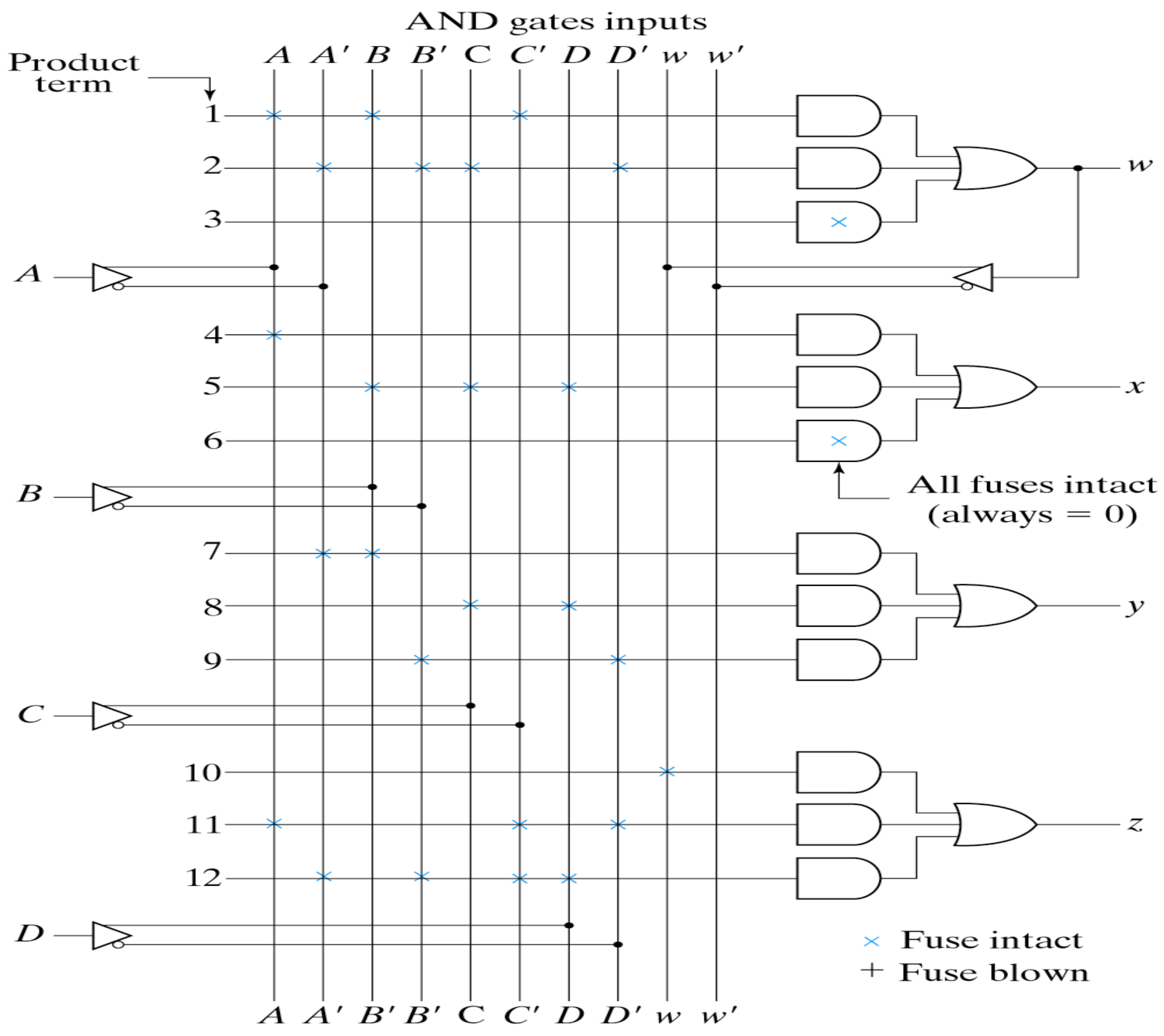
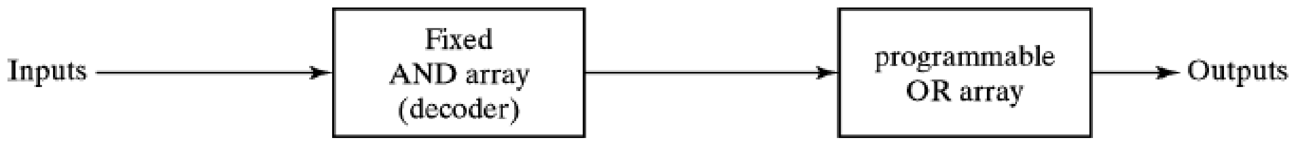


Fig. 7-17 Fuse Map for PAL as Specified in Table 7-6

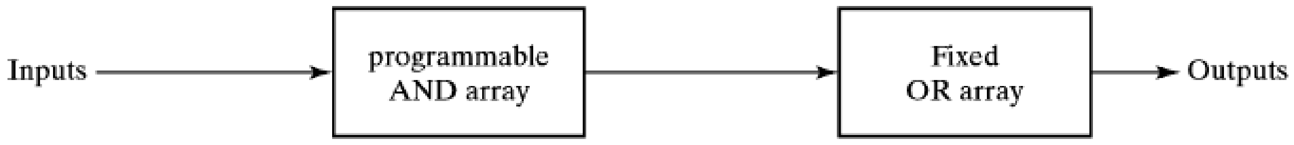
For each 1 or 0 in the table, we mark the corresponding intersection in the diagram with the symbol for an intact fuse. For each dash we mark he diagram with blown fuses in both the true and complement inputs. If the AND gate is not used, we leave all its input fuses intact.

Since the corresponding input receives both the true and complement of each input variable we have AA'=O and the output of the AND gate is always O.

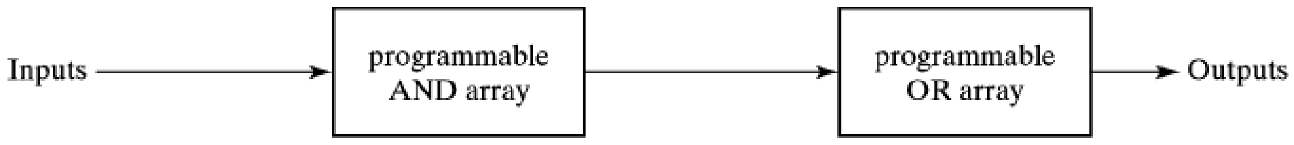
Configurations of Three PLDs



(a) Programmable read-only memory (PROM)



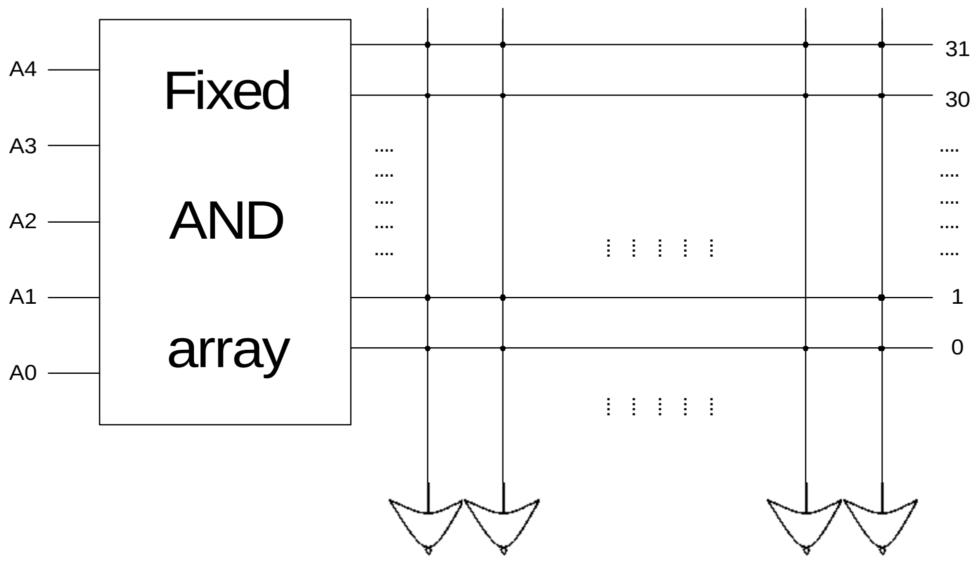
(b) Programmable array logic (PAL)



(c) Programmable logic array (PLA)

PROM

Programmable OR array



Blow fuses on all non required connections

PROM Design Example: BCD to Gray Code Converter

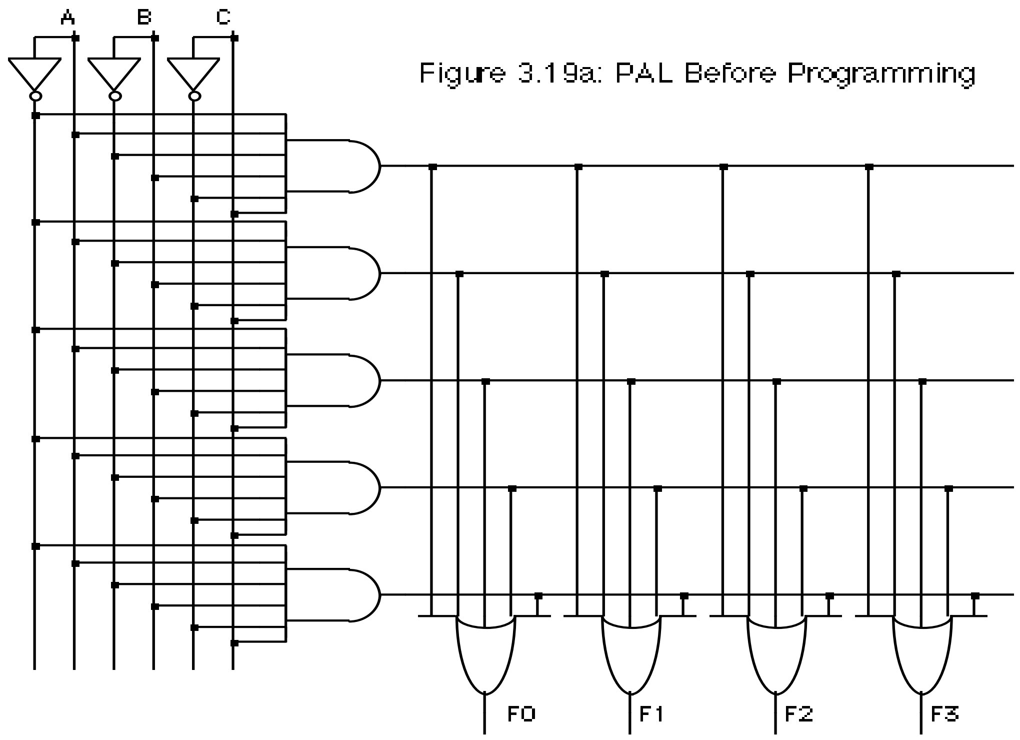
Connect address [A3 – A0] to inputs [A, B, C, D] & data o/ps to [W, X, Y, Z]

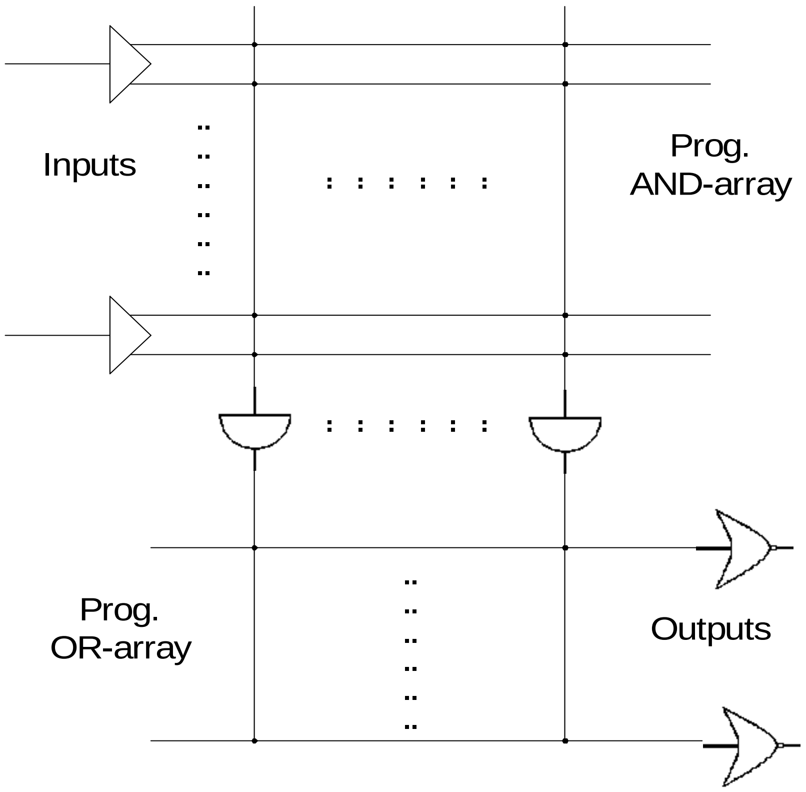
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

Without PROM, this decoder would need several ICs. PROM is well suited to those apps where every possible input combination (AND term) is needed, eg. Code converters, data storage.

Drawbacks:
Each new i/p needs no. of Memory cells to be doubled
No of i/ps & o/ps fixed
As memory size increases, so do cost, power, delay.
PLA or PAL may be more flexible & cost effective.

PLA – Programmable AND and OR arrays





Only subset of possible AND terms provided. Each AND gate has connections to each i/p. By leaving appropriate fuse intact, any i/p or its complement can be present in any AND term.

PLA Eg

Equations

F0 = A + B' C'
F1 = A C' + A B
F2 = B' C' + A B
F3 = B' C + A

Programming

Product term	Inputs A	B	C	Outputs F_0	F_1	F_2	F_3
A B	1	1	-	0	①	①	0
$\overline{B}$ C	-	0	1	0	0	0	1
A $\overline{C}$	1	-	0	0	1	0	0
$\overline{B}$ $\overline{C}$	-	0	0	①	0	①	0
A	1	-	-	①	0	0	①

Reuse of terms

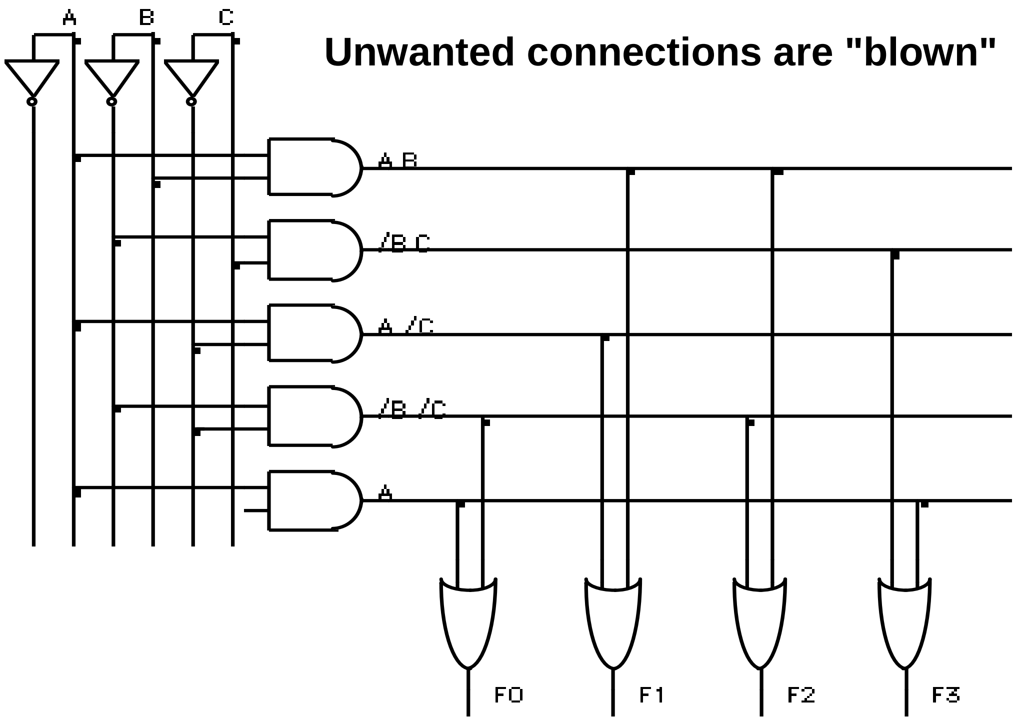
Input Side:

1 = asserted in term
0 = negated in term
- = does not participate

Output Side:

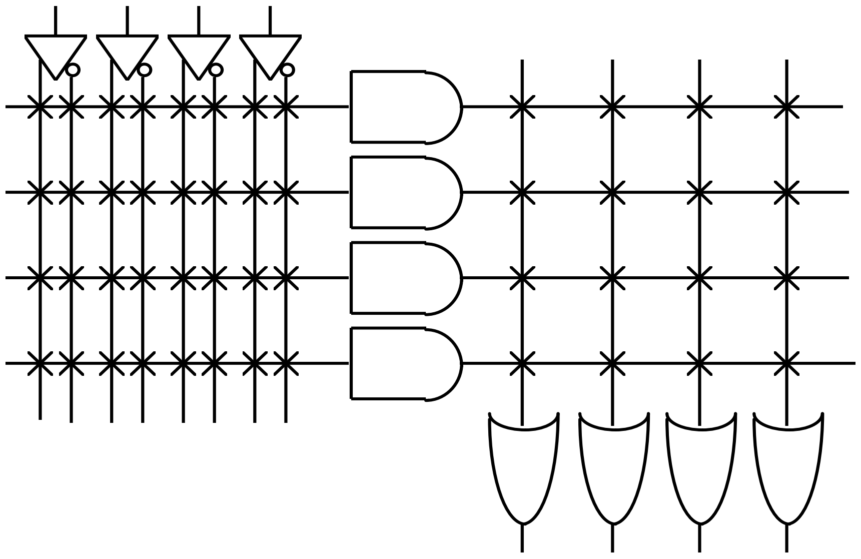
1 = term connected to output
0 = no connection to output

Unwanted connections are "blown"



Alternative representation for PLAs

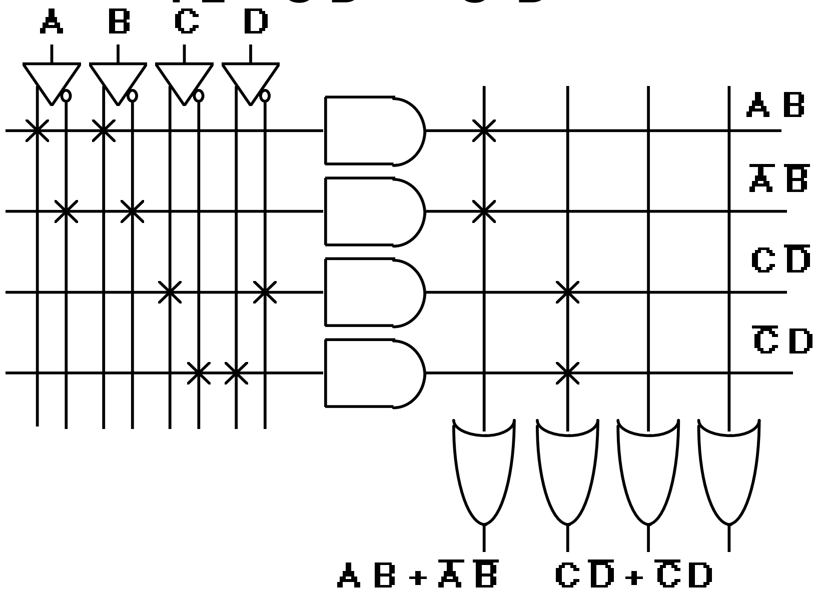
Short-hand notation so we don't have to draw all the wires!



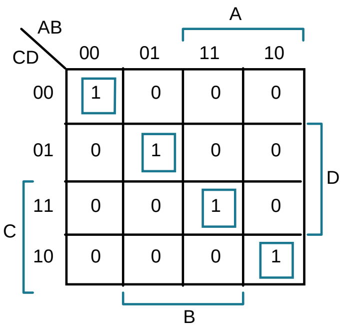
Notation for implementing

F0 = A B + A' B'

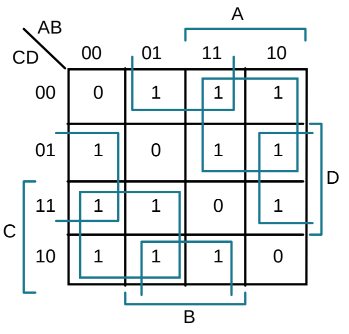
F1 = C D' + C' D


A
B
C
D
A B
$\overline{A}\,\overline{B}$
$C\,\overline{D}$
$\overline{C}\,D$
$A\,B + \overline{A}\,\overline{B}$
$C\,\overline{D} + \overline{C}\,D$

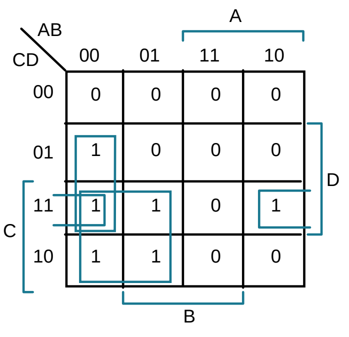
Another PLA Example: Magnitude Comparator



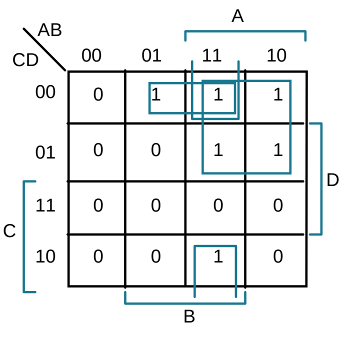
K-map for EQ



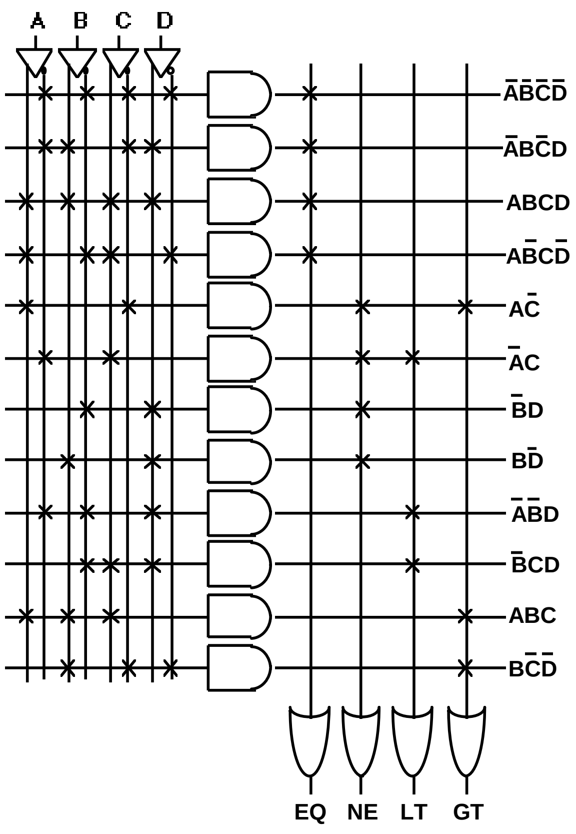
K-map for NE



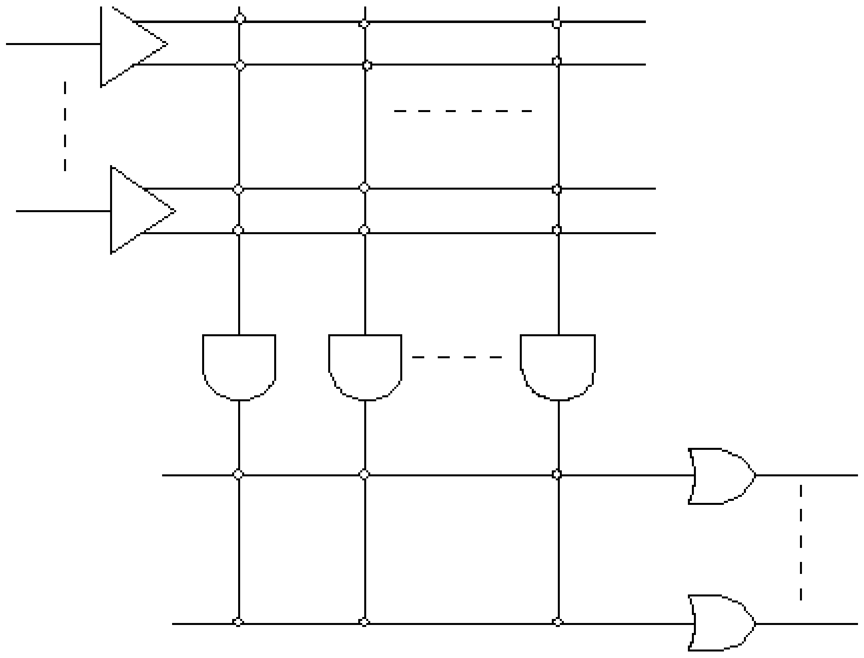
K-map for LT



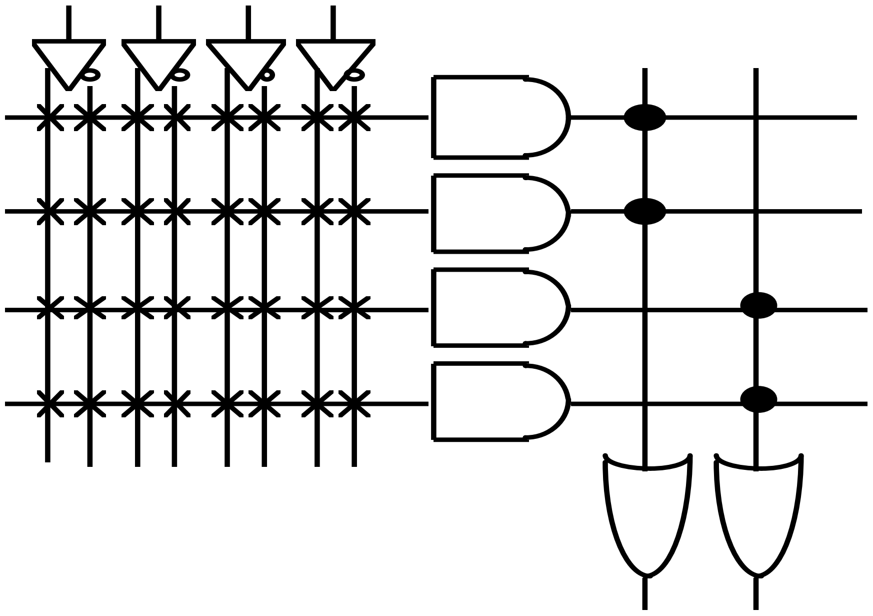
K-map for GT



PAL has programmable AND-array, but fixed OR-array.



A given column of the OR array has access to only a subset of the possible product terms



PALs simpler to understand and use than PLAs and have performance advantages: a fuse array has high capacitance which causes delay.

Reducing no. of AND terms reduces cost and power consumption.

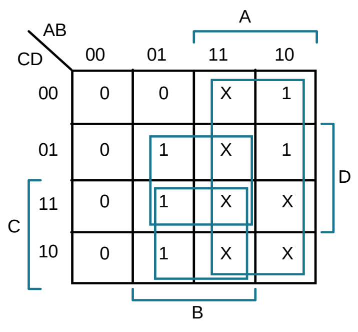
First PALs suffered from restrictions of fixed i/p and o/p nos. And fixed o/p polarity.

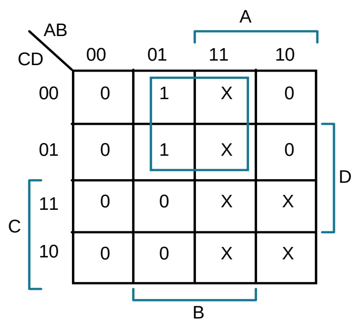
PAL Design Example: BCD to Gray Code Converter

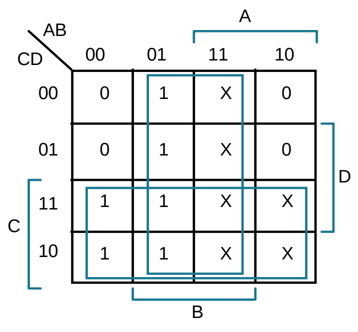
TruthTable

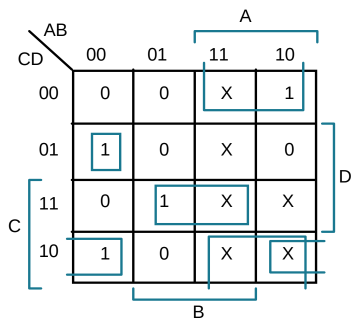
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X

K-maps


K-map for W


K-map for X


K-map for Y

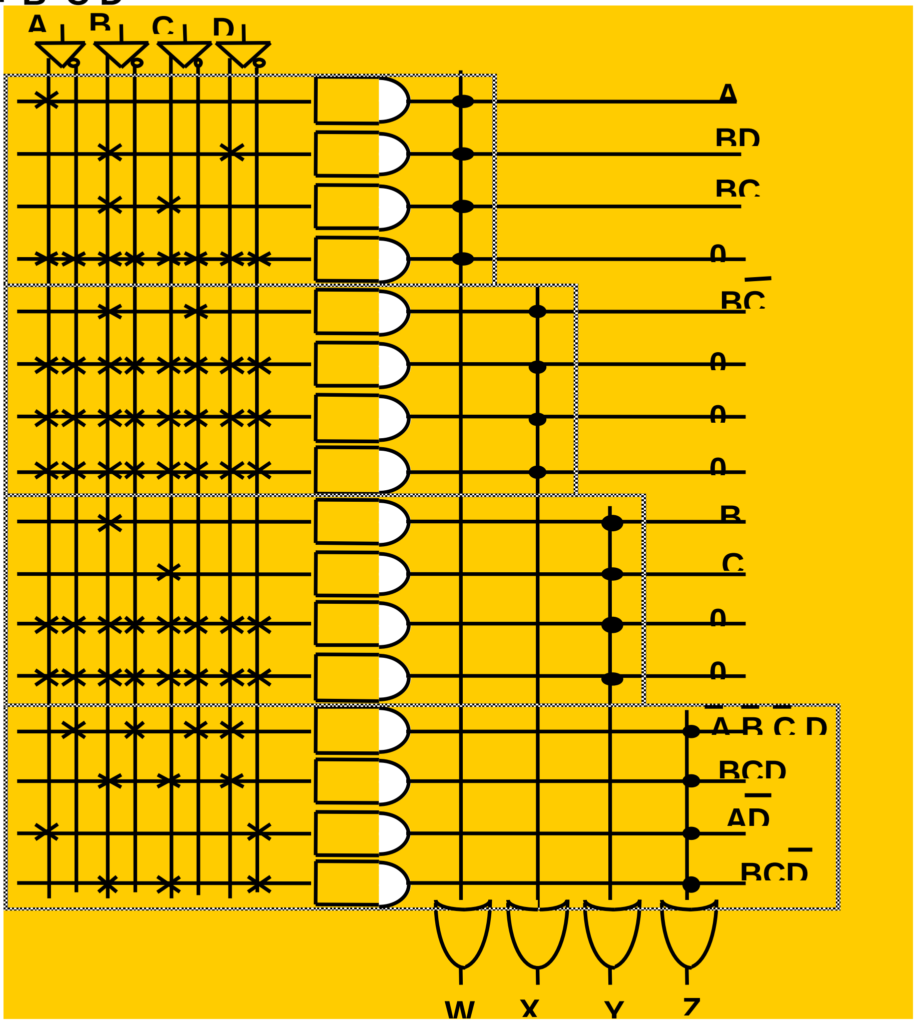

K-map for Z

Minimized Functions:

W = A + B D + B C
X = B C'
Y = B + C
Z = A'B'C'D + B C D + A D' + B' C D'

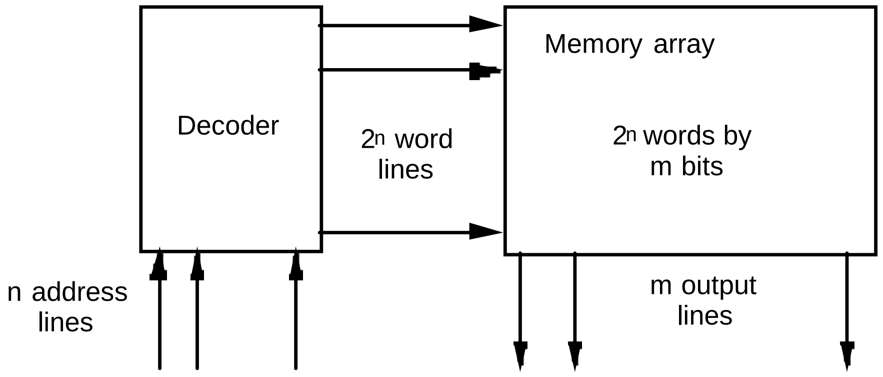
Programmed PAL:

4 product terms per each OR gate



PROM

Not unlike a PLA structure with a fully decoded AND array!



ROM vs. PLA:

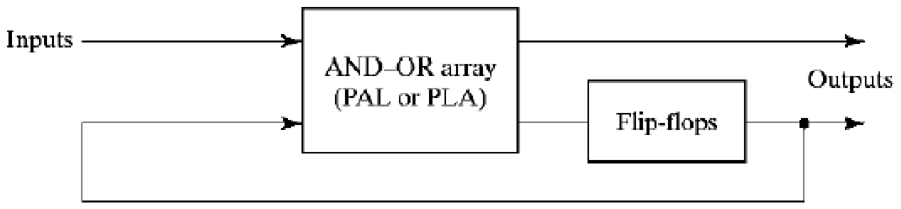
ROM approach advantageous when
(1) design time is short (no need to minimize output functions)
(2) most input combinations are needed (e.g., code converters)
(3) little sharing of product terms among output functions

ROM problem: size doubles for each additional input, can't use don't cares

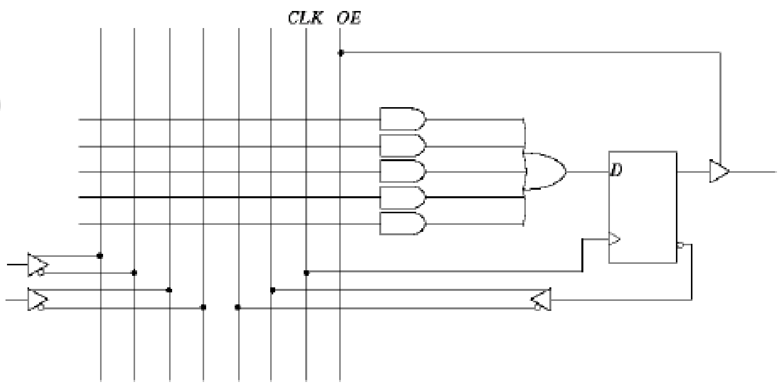
PLA approach advantangeous when
(1) design tool like espresso is available
(2) there are relatively few unique minterm combinations
(3) many minterms are shared among the output functions

PAL problem: constrained fan-ins on OR planes

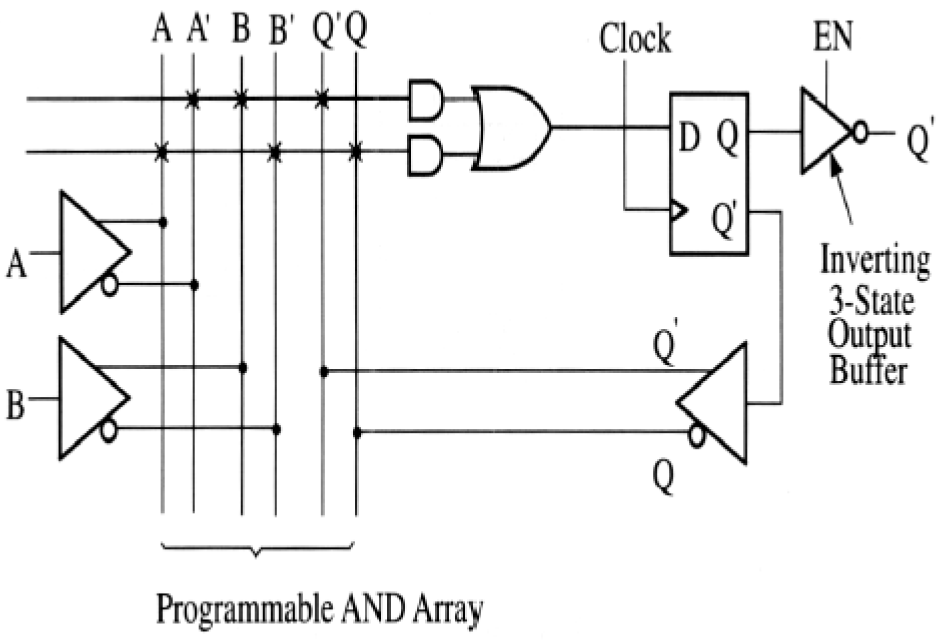
The most simple sequential PLD = PLA (PAL) + Flip-Flops



The mostly used configuration for SPLD is constructed with 8 to 10 macrocells as shown right

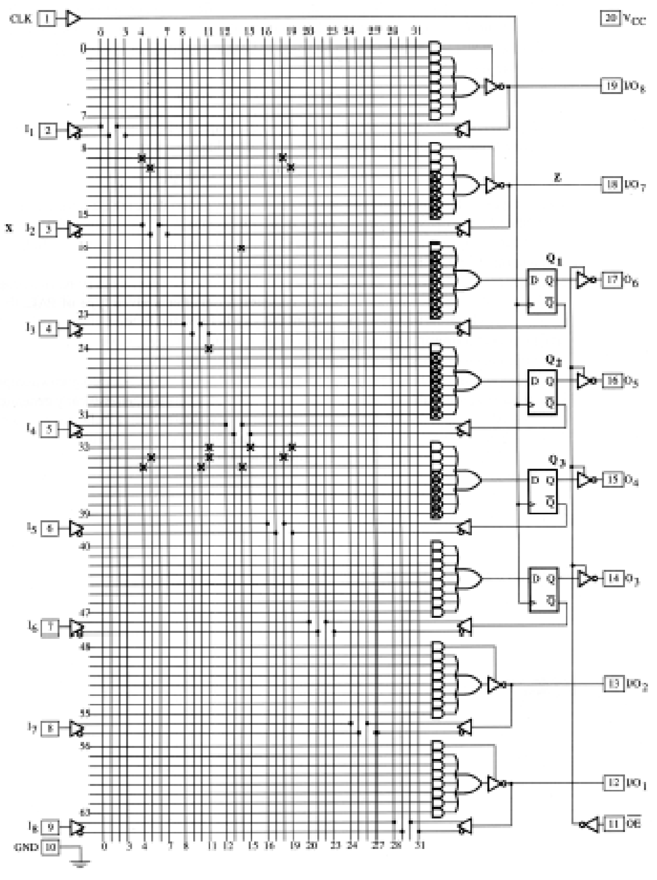


Registered (Sequential) PAL:



$$Q^{+}=D=A'BQ'+AB'Q$$

16R4 PAL:



AND gate is logic 1 when there are no connections to it

$$Z'=XQ_3'+X'Q_3$$

$$D_3=Q_1Q_2Q_3+X'Q_1Q_3'+XQ_1'Q_2'$$