Digital Logic and Circuit
Paper Code: CS-102

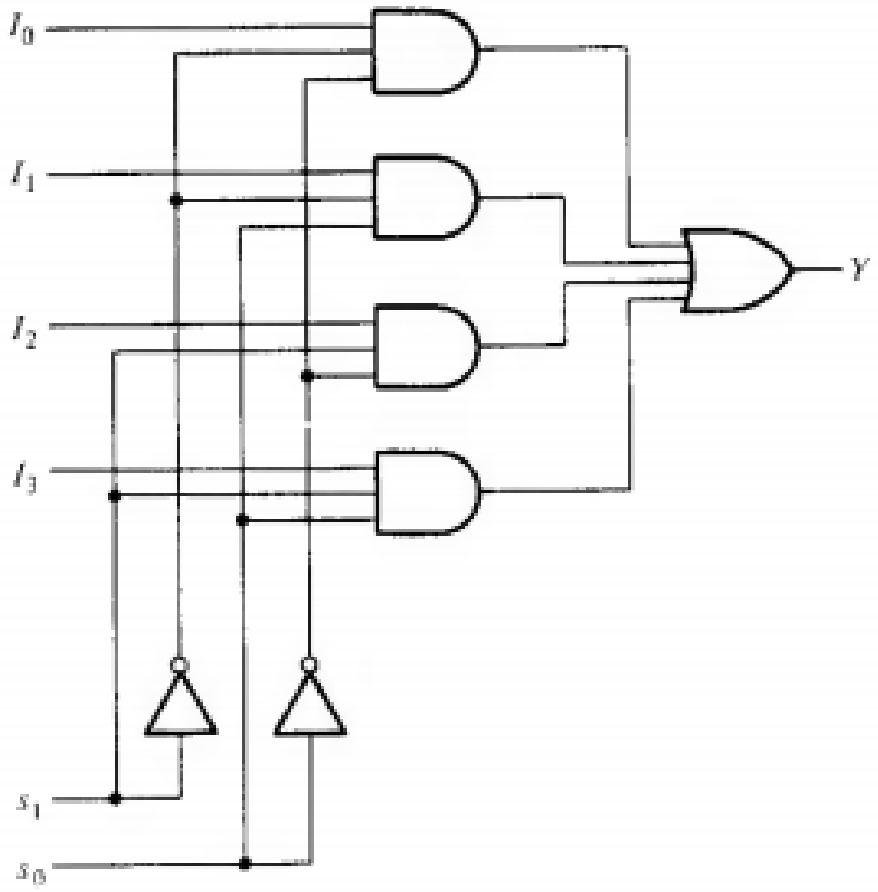
Outline

Combinational circuit

- **Multiplexers & de-multiplexers**
  - **Boolean function implementation using MUX.**
- Combinational Logic Implementation

Multiplexer

- A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.
- The selection of a particular input line is controlled by a set of selection lines.
- Normally, there are 2" input lines and n selection lines whose bit combinations determine which input is selected.



(a) Logic diagram

Boolean function implementation using MUX.

- A decoder can be used to implement a Boolean function by employing an external OR gate.
- A multiplexer is essentially a decoder with the OR gate already available.
- The minterms out of the decoder to be chosen can be controlled with the input lines.
- The minterms to be included with the function being implemented are chosen by making their corresponding input lines equal to 1 ; those minterms not included in the function are disabled by making their input lines equal to 0.
- This gives a method for implementing any Boolean function of n variables with a 2"-to-l multiplexer.

- If we have a Boolen function of n + 1 variables, we take n of these variables and connect them to the selection lines of a multiplexer.

- The remaining single variable of the function is used for the inputs of the multiplexer.

- If A is this single variable, the inputs of the multiplexer are chosen to be either A or A’ or 1 or 0. By judicious use of these four values for the inputs and by connecting the other variables to the selection lines, one can implement any Boolean function with a multiplexer.

- In this way, it is possible to generate any function of n + 1 variables with a 2”-to- 1 multiplexer

Steps for Boolean function implementation using MUX

First, express the function in its sum of minterms form.

Assume that the ordered sequence of variables chosen for the minterms is ABCD . . . ,

where A is the leftmost variable in the ordered sequence of n variables and BCD . . . are the remaining n - 1 variables.

Connect the n - 1 variables to the selection lines of the multiplexer, with B connected to the high-order selection line, C to the next lower selection line, and so on down to the last variable, which is connected to the lowest-order selection line s0 .

Consider now the single variable A. Since this variable is in the highest-order position in the sequence of variables, it will be complemented in minterms 0 to $(2^n/2) - 1$, which comprise the first half in the list of minterms.

The second half of the minterms will have their A variable uncomplemented. For a three-variable function, A, B, C, we have eight minterms. Variable A is complemented in minterms 0 to 3 and uncomplemented in minterms 4 to 7.

- List the inputs of the multiplexer and under them list all the minterms in two rows.
- The first row lists all those minterms where A is complemented, and the second row all the minterms with A uncomplemented, as shown in Fig. 5- 18(c). Circle all the minterms of the function and inspect each column separately.
- If the two minterms in a column are not circled, apply 0 to the corresponding multiplexer input.
- If the two minterms are circled, apply 1 to the corresponding multiplexer input.
- If the bottom minterm is circled and the top is not circled, apply A to the corresponding multiplexer input.
- If the top minterm is circled and the bottom is not circled, apply A' to the corresponding multiplexer input.

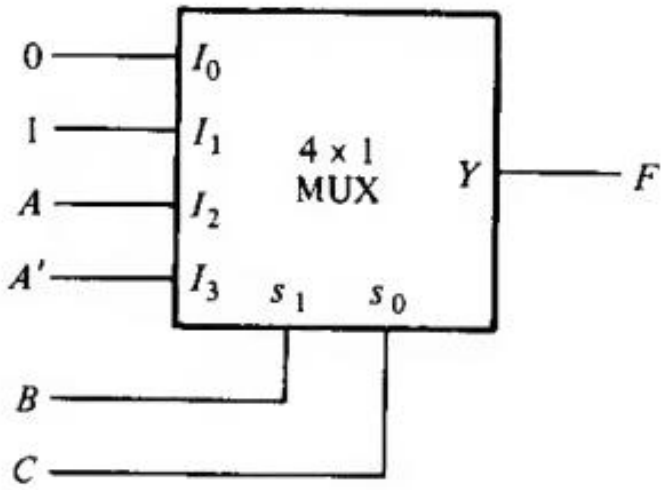
Example Implement Boolean function $F(A, B, C) = \sum(1, 3, 5, 6)$

The function can be implemented with a 4-to-1 multiplexer.

Two of the variables, B and C, are applied to the selection lines in that order, i.e. , B is connected to s1 and C to ,s0 .

The inputs of the multiplexer are 0, 1, A, and A ‘

Note: for 3 variables, we have 2 selection lines so (2^2:1 MUX) 4:1 MUX can be used



(a) Multiplexer implementation

Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

(b) Truth table

	I_0	I_1	I_2	I_3
A'	0	①	2	③
A	4	⑤	⑥	7
	0	1	A	A'

(c) Implementation table

Inputs to multiplexer are decided using Implementation table

Implement Boolean function
$F(A, B, C, D) = \sum(1, 3, 4, 11, 12, 13, 14, 15)$ using 8:1 and 16:1 MUX

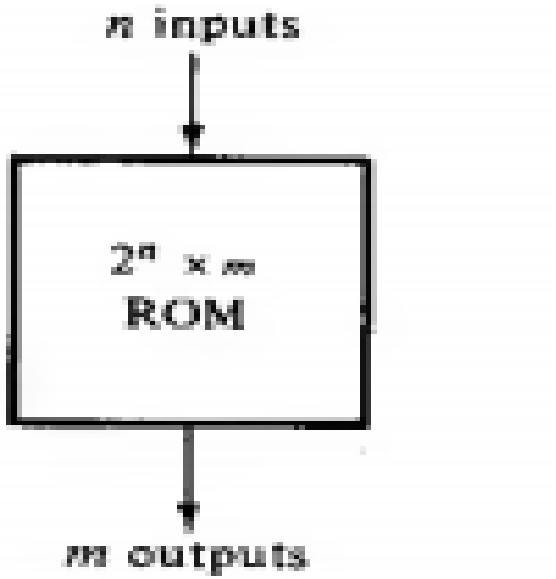
Read Only Memory (ROM)

- A ROM is essentially a memory (or storage) 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.
- ROMs come with special internal electronic fuses that can be “programmed” for a specific configuration.
- Once the pattern is established, it stays within the unit even when power is turned off and on again.

- A ROM is a device that includes both the decoder and the OR gates within a single IC package.
- Each bit combination of the input variables is called an address.

- Each bit combination that comes out of the output lines is called a word.
- The number of bits per word is equal to the number of output lines, m.
- An address is essentially a binary number that denotes one of the minterms of n variables.
- The number of distinct addresses possible with n input variables is 2".
- An output word can be selected by a unique address, and since there are 2" distinct addresses in a ROM, there are 2" distinct words that are said to be stored in the unit.
- The word available on the output lines at any given time depends on the address value applied to the input lines.
- A ROM is characterized by the number of words 2" and the number of bits per word m.

Block diagram of ROM



Consider a 32 x 8 ROM

The unit consists of 32 words of 8 bits each. This means that there are eight output lines and that there are 32, distinct words stored in the unit, each of which may be applied to the output lines.

The particular word selected that is presently available on the output lines is determined from the five input lines.

There are only five inputs in a 32 X 8 ROM because $2^5 = 32$, and with five variables, we can specify 32 addresses or minterms.

For each address input, there is a unique selected word.

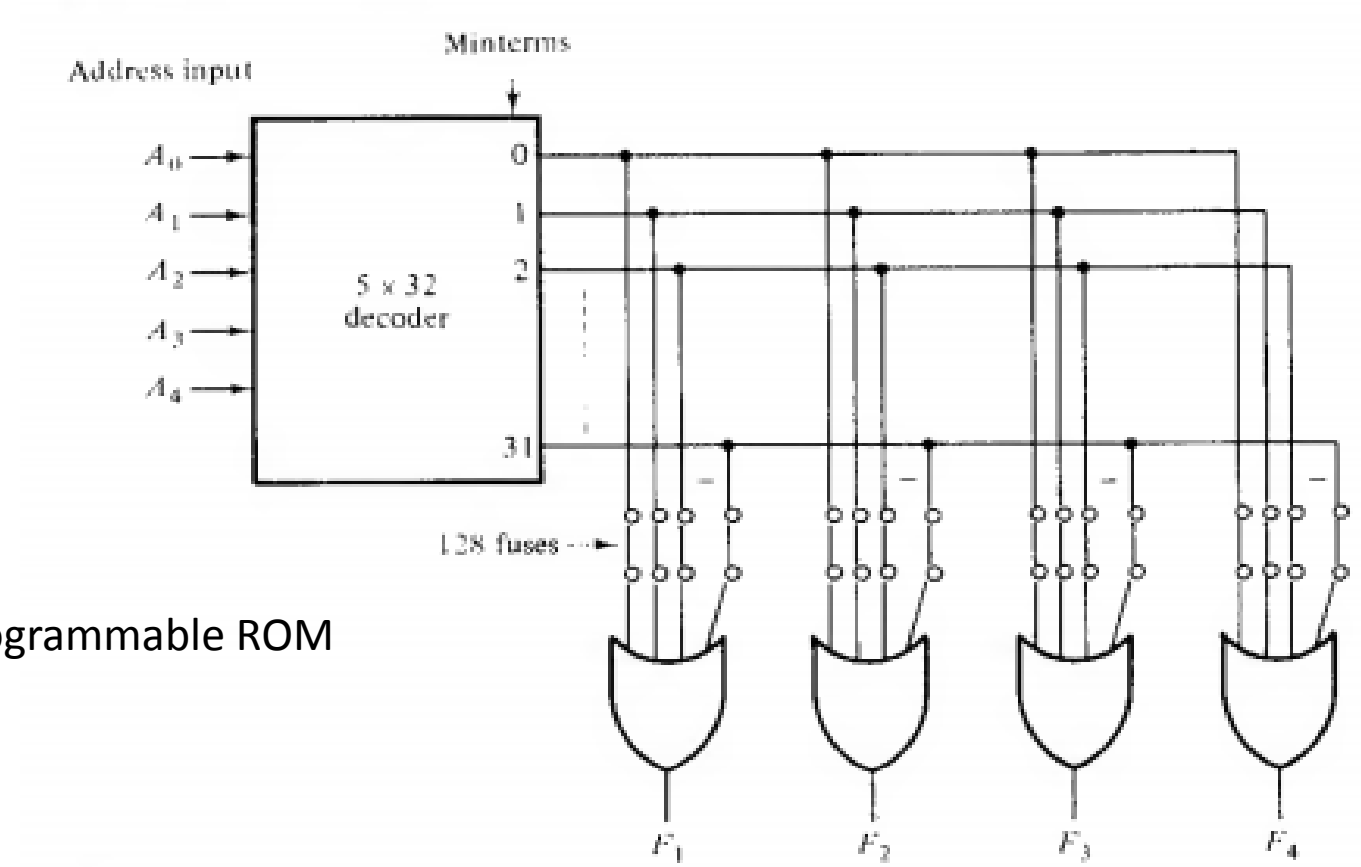
Thus, if the input address is 00000, word number 0 is selected and it appears on the output lines. If the input address is 11111, word number 31 is selected and applied to the output lines. In between, there are 30 other addresses that can select the other 30 words.

- The number of addressed words in a ROM is determined from the fact that n input lines are needed to specify 2" words.

- A ROM is sometimes specified by the total number of bits it contains, which is 2" x m.
  - For example, a 2048-bit ROM may be organized as 512 words of 4 bits each.
- This means that the unit has four output lines and nine input lines to specify $2^9 = 512$ words.

- The total number of bits stored in the unit is 512 x 4 = 2048.

Logic construction of a 32 x 4 ROM



Programmable ROM

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.

The first step is to derive the truth table for the combinational circuit.

In most cases, this is all that is needed. In some cases, we can fit a smaller truth table for the ROM by using certain properties in the truth table of the combinational circuit.

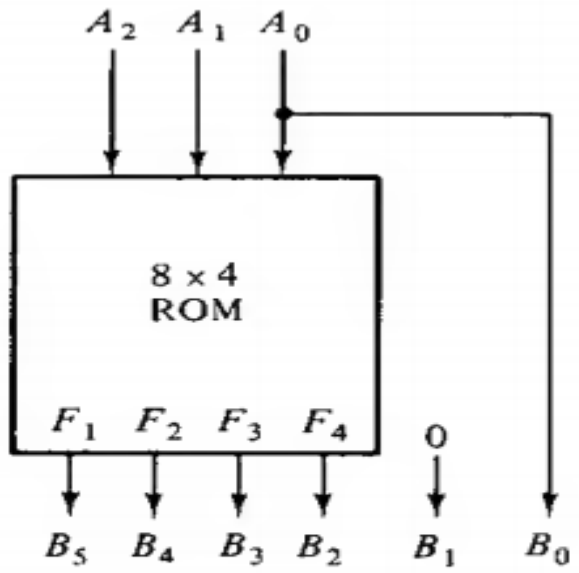
Truth Table for Circuit

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

Solution Cont..

- Three inputs and six outputs are needed to accommodate all possible numbers.
- We note that output B_0 is always equal to input A_0; so there is no need to generate Bo with a ROM since it is equal to an input variable.
  Moreover, output B_1 is always 0, so this output is always known. We actually need to generate only four outputs with the ROM; the other two are easily obtained.
- The minimum-size ROM needed must have three inputs and four outputs. Three inputs specify eight words, so the ROM size must be 8 x 4.

Solution Cont..



(a) Block diagram

A_2	A_1	A_0	F_1	F_2	F_3	F_4
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

ROM implementation

PROGRAMMABLE LOGIC ARRAY (PLA)

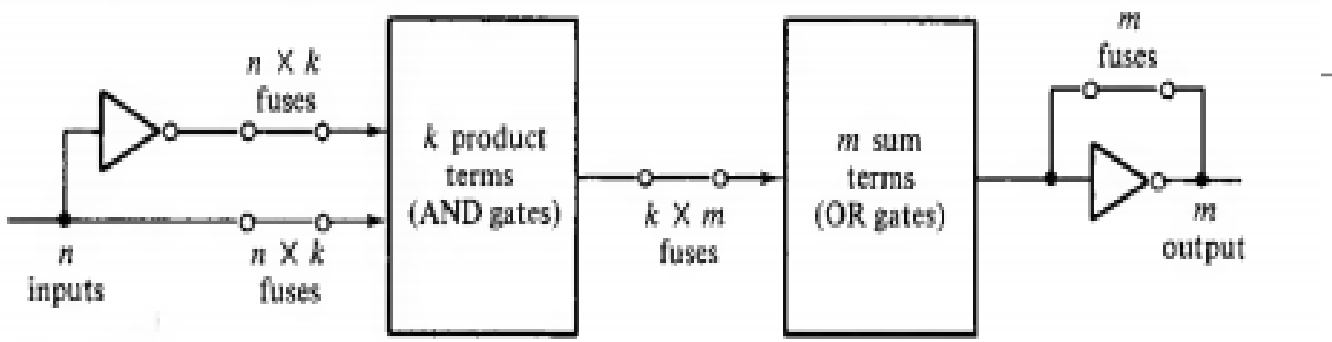
- A PLA is similar to a ROM in concept; however, the PLA does not provide full decoding of the variables and does not generate all the minterms as in the ROM.

- In the PLA, the decoder is replaced by a group of AND gates, each of which can be programmed to generate a product term of the input variables.

- The AND and OR gates inside the PLA are initially fabricated with fuses among them.

- The specific Boolean functions are implemented in sum of products form by blowing appropriate fuses and leaving the desired connections.

Block diagram of PLA



- It consists of n inputs, m outputs, k product terms, and m sum terms.
- The product terms constitute a group of **k** AND gates and the sum terms constitute a group of **m** OR gates.
- Fuses are inserted between all n inputs and their complement values to each of the AND gates.
- Fuses are also provided between the outputs of the AND gates and the inputs of the OR gates.
- Another set of fuses in the output inverters allows the output function to be generated either in the AND-OR form or in the AND-OR-INVERT form.
- With the inverter fuse in place, the inverter is bypassed, giving an AND-OR implementation.
- With the fuse blown, the inverter becomes part of the circuit and the function is implemented in the AND-OR-INVERT form.

- The use of a PLA must be considered for combinational circuits that have a large number of inputs and outputs.

- It is superior to a ROM for circuits that have a large number of don’t-care conditions.

Consider the truth table of the combinational circuit.

A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

(a) Truth table

- Although a ROM implements a combinational circuit in its sum of minterms form, a PLA implements the functions in their sum of products form.

- Each product term in the expression requires an AND gate.

- Since the number of AND gates in a PLA is finite, it is necessary to simplify the function to a minimum number of product terms in order to minimize the number of AND gates used.

Example:

The simplified functions in sum of products are obtained using k-Map

F_1=AB'+AC

F_2=AC+BC

There are three distinct product terms in thi combinational circuit: AB', AC, and BC. The circuit has three inputs and two outputs

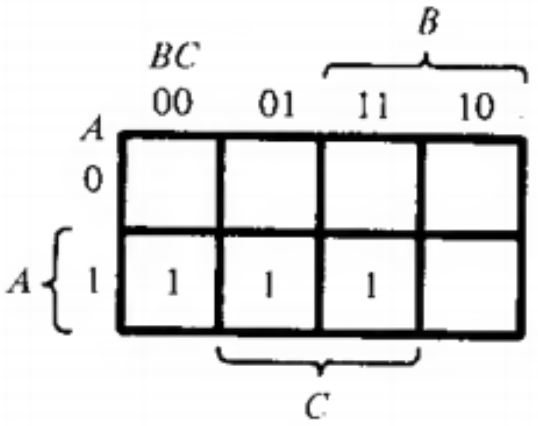
No. of input buffer=no. of input variable

No. of AND gate=no. of product term

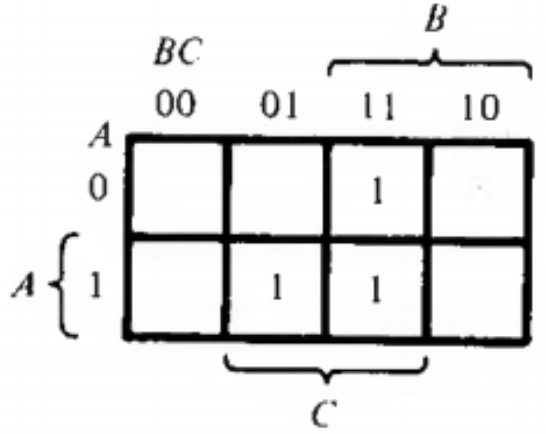
No. of OR gate=No. of Output function

A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

(a) Truth table



$F_1 = AB' + AC$



$F_2 = AC + BC$

(b) Map simplification

- For each product term, the inputs are marked with 1 , 0, or - (dash).
- If a variable in the product term appears in its normal form (unprimed), the corresponding input variable is marked with a 1.

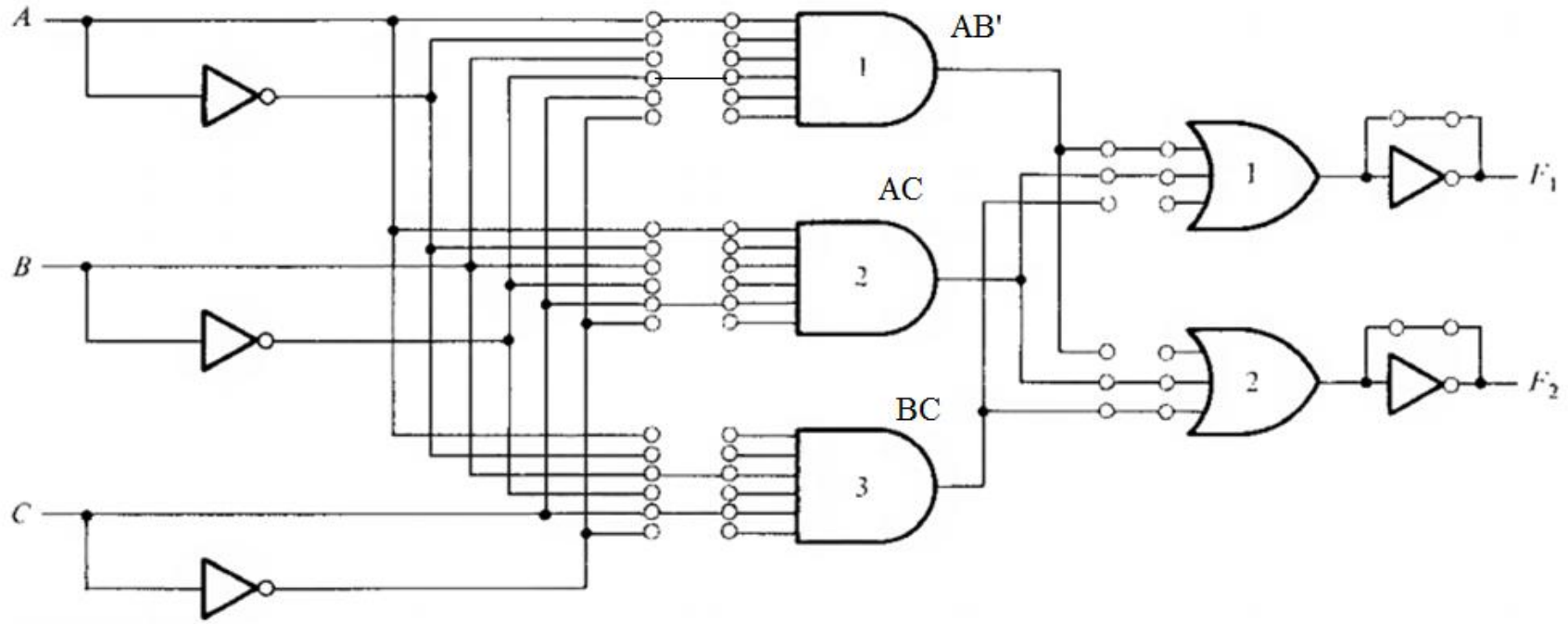
- If it appears complemented (primed), the corresponding input variable is marked with a 0.
- If the variable is absent in the product term, it is marked with a dash.

	Product term	Inputs A	B	C	Outputs F_1	F_2	
AB'	1	1	0	–	1	–	
AC	2	1	–	1	1	1	
BC	3	–	1	1	–	1	
					T	T	T/C

(c) PLA program table

- so F1 is marked with 1's for product terms 1 and 2 and with a dash for product term 3.
  Each product term that has a 1 in the output column requires a path from the corresponding AND gate to the output OR gate.
- Those marked with a dash specify no connection.
- Finally, a T (true) output dictates that the fuse across the output inverter remains intact, and a C (complement) specifies that the corresponding fuse be blown.

PLA with three Inputs, three product terms, and two outputs;



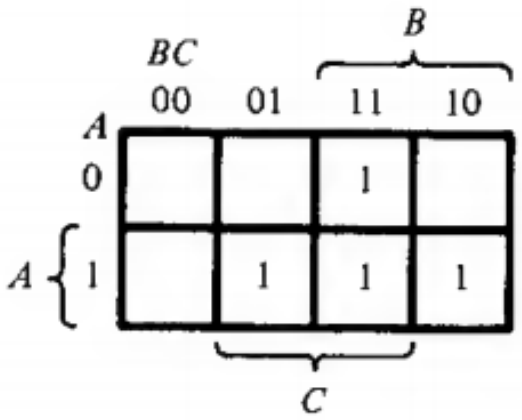
The example to be presented demonstrates how a PLA is programmed.

Bear in mind when going through the example that such a simple circuit will not require a PLA because it can be implemented more economically with SSI gates.

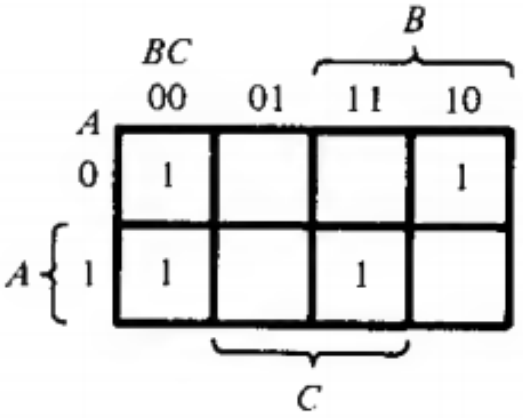
A combinational circuit is defined by the functions
$F_1(A, B, C) = \sum(3, 5, 6, 7)$
$F_2(A, B, C) = \sum(0, 2, 4, 7)$,
Implement the circuit with a PLA having three inputs, four product terms, and two outputs.

Both the true values and the complements of the functions are simplified.

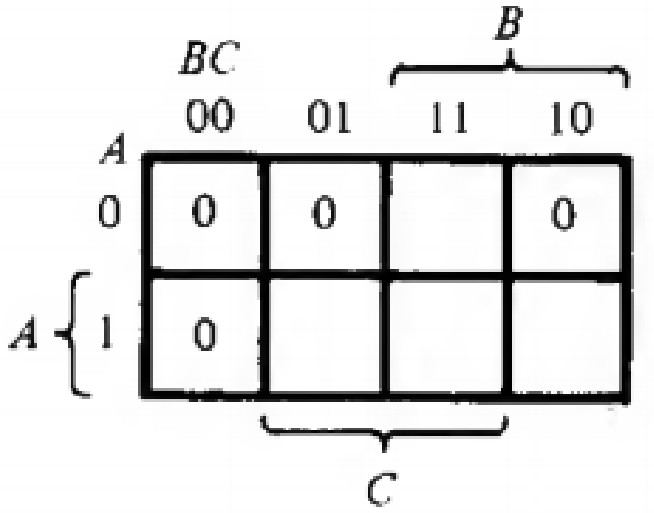
The combinations that give a minimum number of product terms are



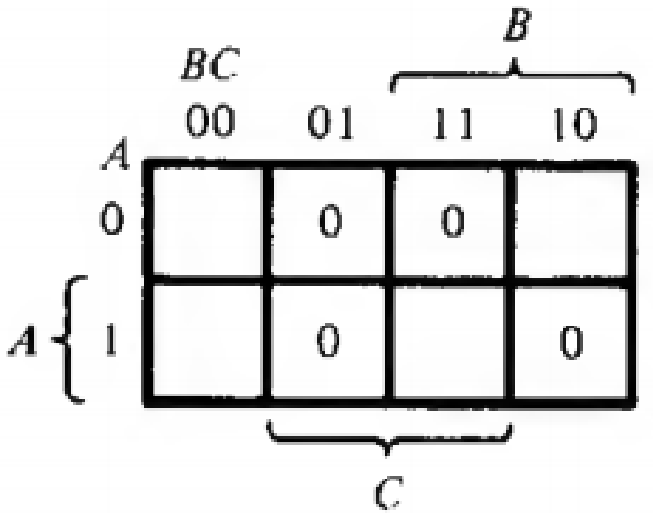
$F_1 = AC + AB + BC$



$F_2 = B'C' + A'C' + ABC$



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



$$F_2' = B'C + A'C + ABC$$

PLA program table

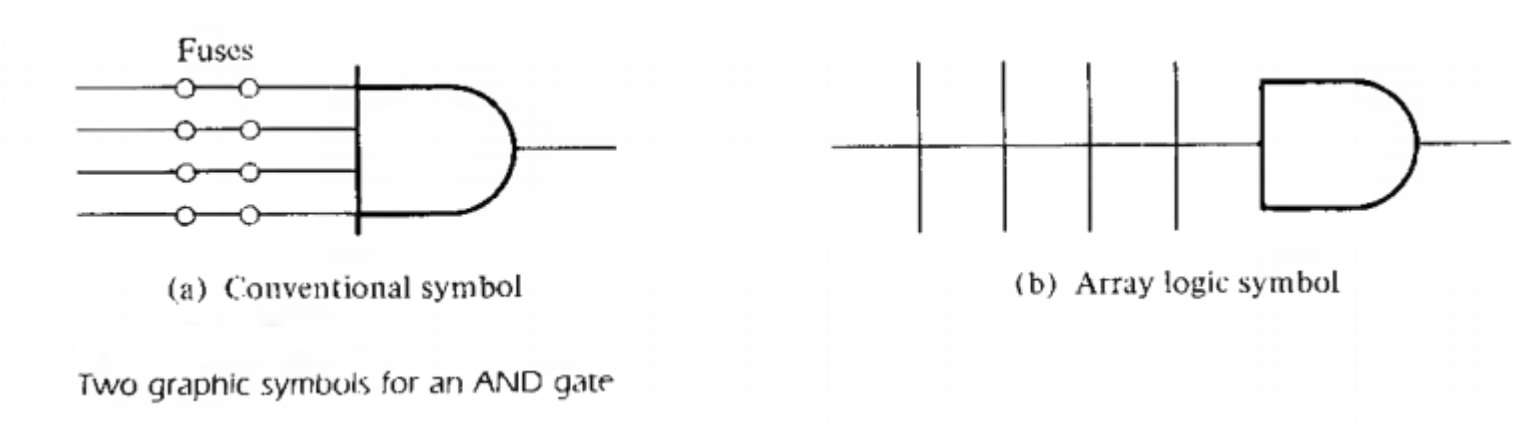
	Product term	Inputs A	Inputs B	Inputs C	Outputs F_1	Outputs F_2
$B'C'$	1	–	0	0	1	1
$A'C'$	2	0	–	0	1	1
$A'B'$	3	0	0	–	1	–
ABC	4	1	1	1	–	1
					C	T

T/C

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.



Two graphic symbols for an AND gate

Example: Consider the following Boolean functions given in sum of minterms:

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

Simplifying the four functions to a minimum number of terms results in the following Boolean functions:

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

Note that the function for z has four product terms. The logical sum of two of these terms is equal to w. By using w, it is possible to reduce the number of terms for z from four to three.

The PAL programming table is similar to the one used for the PLA except that only the inputs of the AND gates need to be programmed.

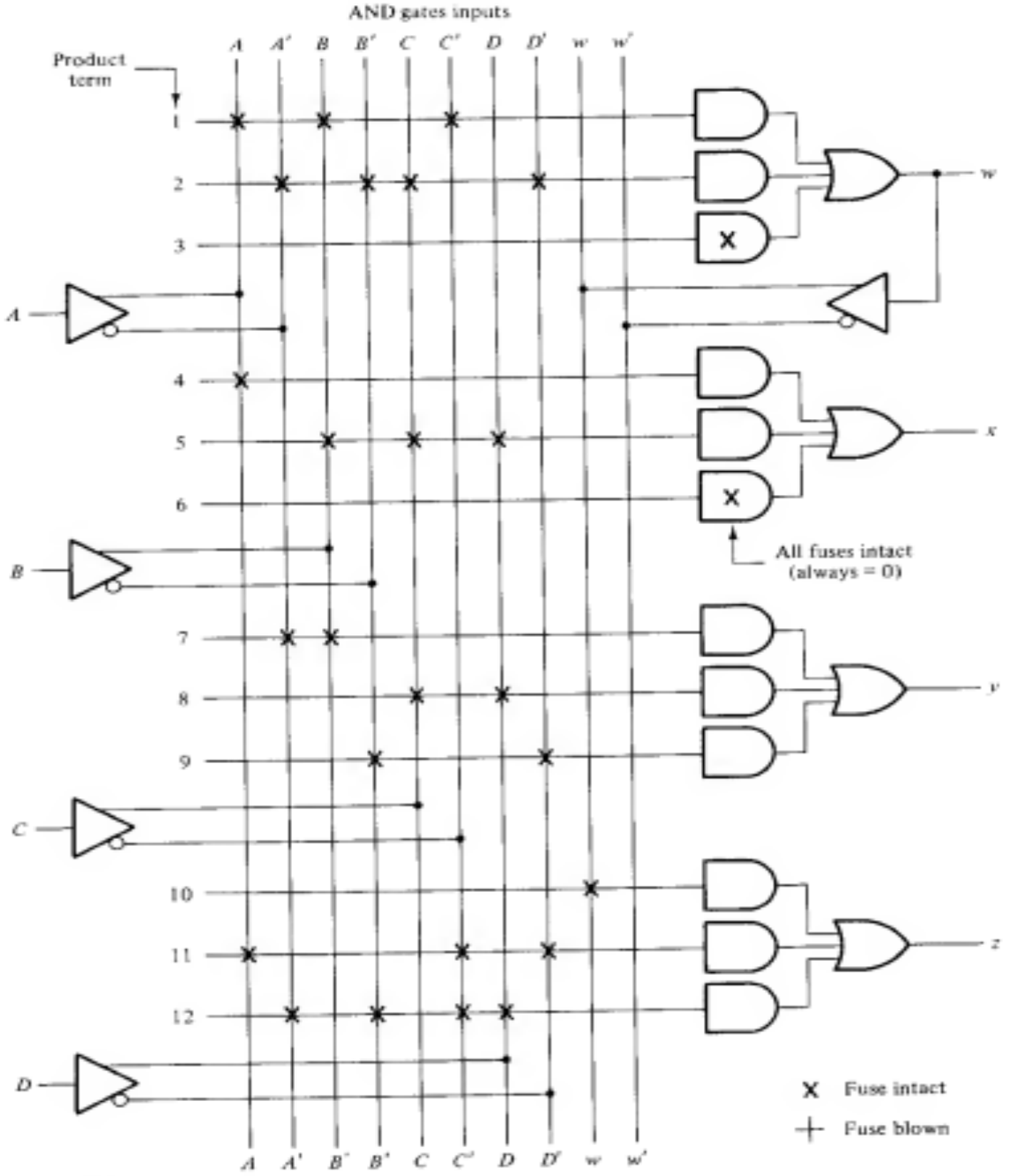
PAL Programming Table

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	–	$+ A'B'C'D$

Input buffer=no. of input varaible

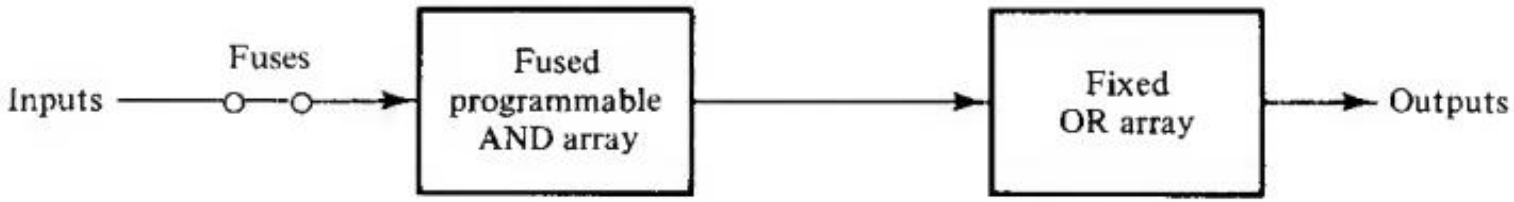
No. of AND gate= Highest number of product term present in any Boolean expression

No. of OR gate=number of output.

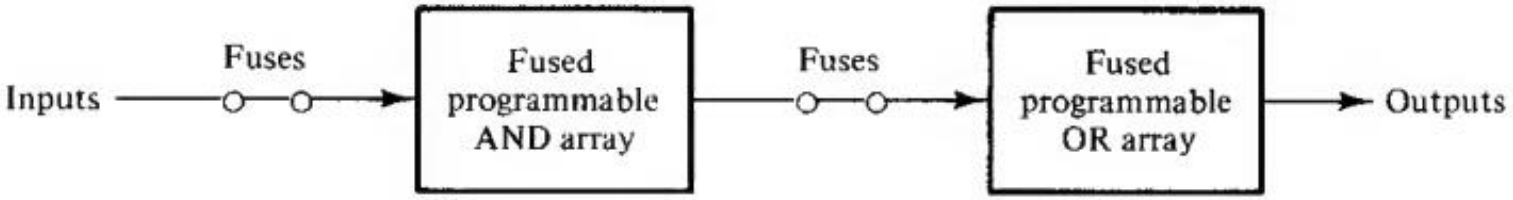


- PAL has four inputs and four out puts. Each input has a buffer and an inverter gate. Note that the two gates are shown with one composite graphic symbol with normal and complement outputs.
- There are four sections in the unit, each being composed of a threewide AND-OR array.
- This is the term used to indicate that there are three programmable AND gates in each section and one fixed OR gate. Each AND gate has 10 fused programmable inputs.
- 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 the inputs of the AND gates through fuses.

Basic Configuration of PAL and PLA



(b) Programmable array logic (PAL)



(c) Programmable logic array (PLA)

Suggested Reading

- M. Morris Mano, Digital Logic and Computer Design, PHI.

Thank you