❖ Introduction

In general there are two broad categories of logic devices, namely fixed logic devices and programmable logic devices. Whereas a fixed logic device such as a logic gate or a multiplexer or a flip-flop performs a given logic function that is known at the time of device manufacture, a programmable logic device can be configured by the user to perform a large variety of logic functions.

Fixed Logic Devices	Programmable Logic Devices
the time required from design to the final stage when the manufactured device is actually available is long	PLD-based design requires much less time from design cycle to production run.
In the case of fixed logic devices, the process of design validation followed by incorporation of changes, which leads to an enhanced cost of the initial prototype device.	In the case of PLDs, inexpensive software tools can be used for quick validation of designs.
These devices cannot be changed when the user want	PLDs offer to the users much more flexibility during the design cycle.
Fixed logic devices have an edge for large-volume applications as they can be mass produced more economically. They are also the preferred choice in applications requiring the highest performance level.	

There are many types of programmable logic device, distinguishable from one another in terms of architecture, logic capacity, programmability and certain other specific features.

Types of Programmable Logic Devices

SPLDs (Simple Programmable Logic Devices)

- ROM (Read-Only Memory)
- PLA (Programmable Logic Array)
- PAL (Programmable Array Logic)
- GAL (Generic Array Logic)
- CPLD (Complex Programmable Logic Device)
- FPGA (Field-Programmable Gate Array)

The first three varieties are quite similar to each other:

- They all have an input connection matrix, which connects the inputs of the device to an array of AND-gates.
- They all have an output connection matrix, which connect the outputs of the AND-gates to the inputs of OR-gates which drive the outputs of the device.

These devices can be programed by blowing fuses

❖ ***Programmable ROMs***

PROM (Programmable Read Only Memory) and EPROM (Erasable Programmable Read Only Memory) can be considered to be predecessors to PLDs. The architecture of a programmable ROM allows the user to hardware-implement an arbitrary combinational function of a given number of inputs. When used as a memory device, n inputs of the ROM (called address lines in this case) and m outputs (called data lines) can be used to store 2^n m-bit words. When used as a PLD, it can be used to implement m different combinational functions, with each

function being a chosen function of n variables. Any conceivable n-variable Boolean function can be made to appear at any of the m output lines. A generalized ROM device with n inputs and m outputs has 2^n hard-wired AND gates at the input and m programmable OR gates at the output. Each AND gate has n inputs, and each OR gate has 2^n inputs. Thus, each OR gate can be used to generate any conceivable Boolean function of n variables, and this generalized ROM can be used to produce m arbitrary n-variable Boolean functions . Figure 1 shows the internal architecture of a PROM having four input lines, a hard-wired array of 16 AND gates and a programmable array of four OR gates. A cross $(\times)$ indicates an intact (or unprogrammed) fusible link or interconnection, and a dot $(\bullet)$ indicates a hard-wired interconnection. PROMs, EPROMs and EEPROMs (Electrically Erasable Programmable Read Only Memory) can be programmed using standard PROM programmers. One of the major disadvantages of PROMs is their inefficient use of logic capacity. It is not economical to use PROMs for all those applications. Other disadvantages include relatively higher power consumption and an inability to provide safe covers for asynchronous logic transitions. They are usually much slower than the dedicated logic circuits. Also, they cannot be used to implement sequential logic owing to the absence of flip-flops. A read only memory (ROM) is essentially a memory device that can be used to store a certain fixed set of binary information. As outlined earlier, these devices have certain inherent links that can be made or broken depending upon the type of fusible link to store any user-specified binary information in the device. While, in the case of a conventional fusible link, relevant interconnections are broken to program the device, in the case of an antifuse the relevant interconnections are made to do the same job. This

is illustrated in Fig.2. Figure 2(a) shows the internal logic diagram of a 4 $\times 2$ PROM. The figure shows an unprogrammed PROM. Figures 2 (b) and (c) respectively show the use of a fuse and an antifuse to produce output-1 = AB. Note that in the case of a fuse an unprogrammed interconnection is a 'make' connection, whereas in the case of an antifuse it is a 'break' connection. Once a given pattern is formed, it remains as such even if power is turned off and on. In the case of PROMs, the user can erase the data already stored on the ROM chip and load it with fresh data. A PROM in general has n input lines and m output lines and is designated as a $2^n \times m$ PROM. Looking at the internal architecture of a PROM device, it is a combinational circuit with the AND gates wired as a decoder and having OR gates equal to the number of outputs. A PROM with five input lines and four output lines, for instance, would have the equivalent of a 5×32 decoder at the input that would generate 32 possible terms or product terms. Each of these four OR gates would be a 32-input gate fed from 32 outputs of the decoder through fusible links. Figure 3 shows the internal architecture of a 32×4 PROM. We can see that the input side is hardwired to produce all possible 32 product terms corresponding to five variables. All 32 product terms or minterms are available at the inputs of each of the OR gates through programmable interconnections. This allows the users to have four different five-variable Boolean functions of their choice. Very complex combinational functions can be generated with PROMs by suitably making or breaking these links. To sum up, for implementing an n-input or n-variable, m-output combinational circuit, one would need a $2^n \times m$ PROM. As an illustration, a PROM can be used to implement the following Boolean function with two outputs given by the equations

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

$$F_2(A,B,C) = \sum 1,4,7$$

Implementation of this Boolean function would require an 8×2 PROM.

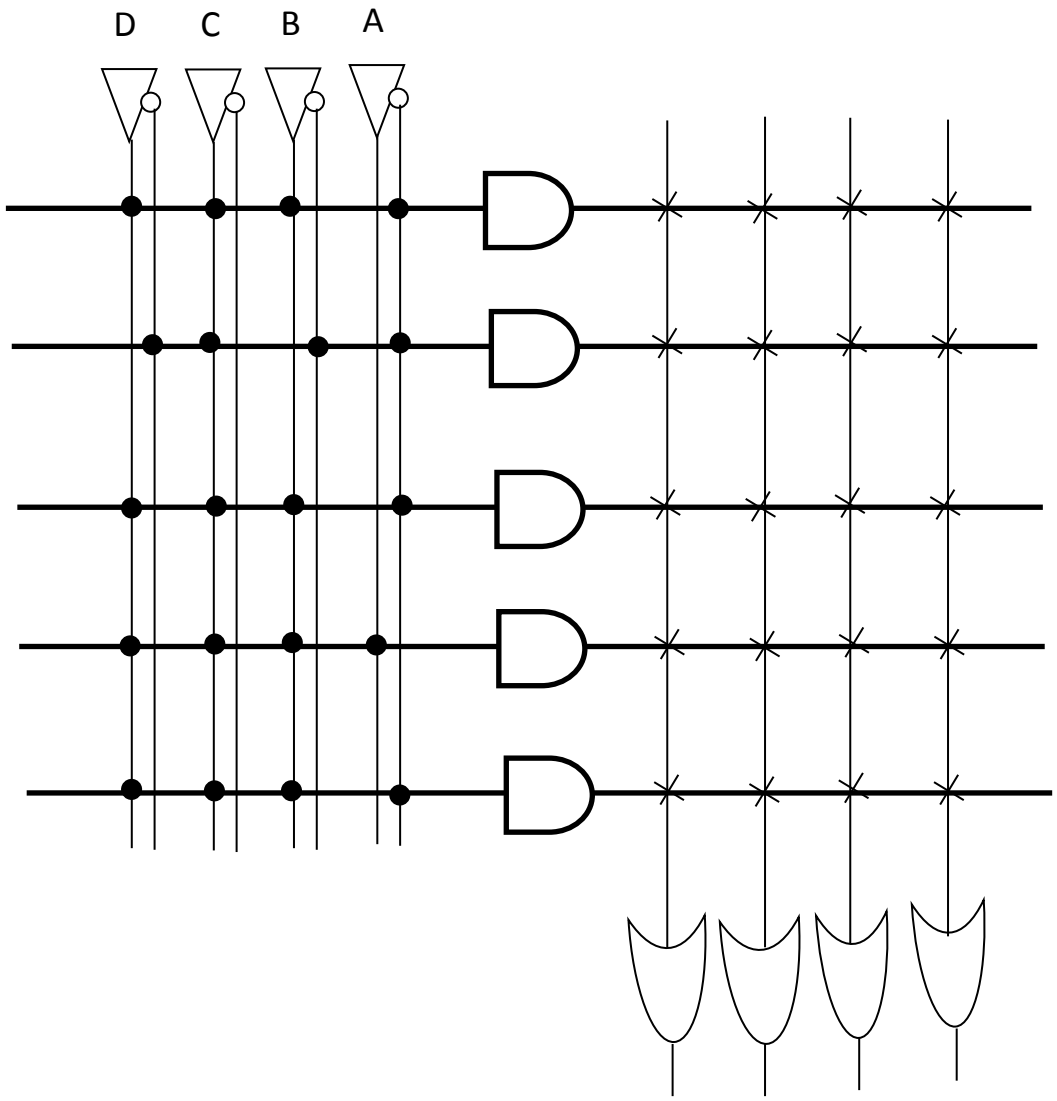

Figure 1 the internal architecture of a PROM

The internal logic diagram of the PROM in this case, after it is programmed, would be as shown in Fig.4. Note that, in the programmed PROM of Fig.4, an unprogrammed interconnection indicated by a cross ($\times$) is a 'make' connection. It may be mentioned here that in practice a PROM would not be used to implement as simple a Boolean function as that illustrated above. The purpose here is to indicate to readers how a

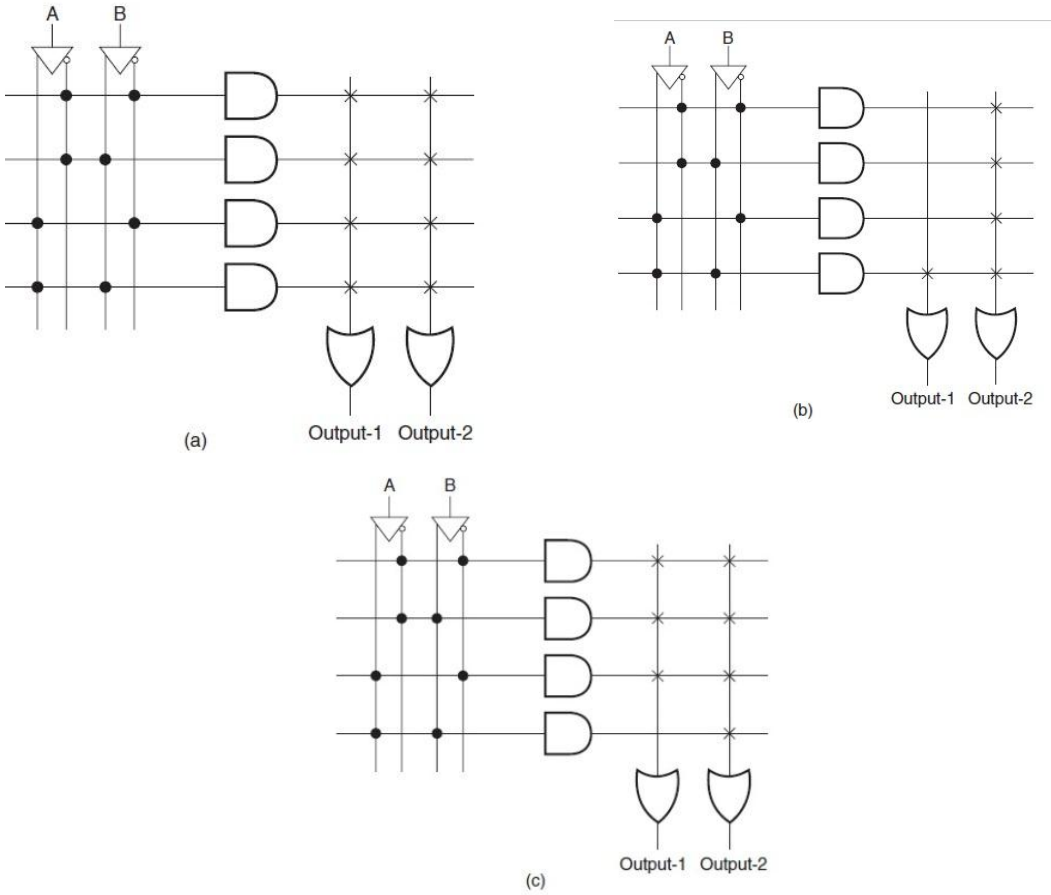

Fig. 2

PROM implements a Boolean function. In actual practice, PROMs would be used only in the case of very complex Boolean functions. Another noteworthy point is that, when it comes to implementing Boolean functions with PROMs, it is not economical to use PROM for those Boolean functions that have a large number of 'don't care' conditions. In the case of a PROM, each 'don't care' condition would have either all 0s or all 1s. Other programmable logic devices such as a PLA or PAL are more suitable in such situations.

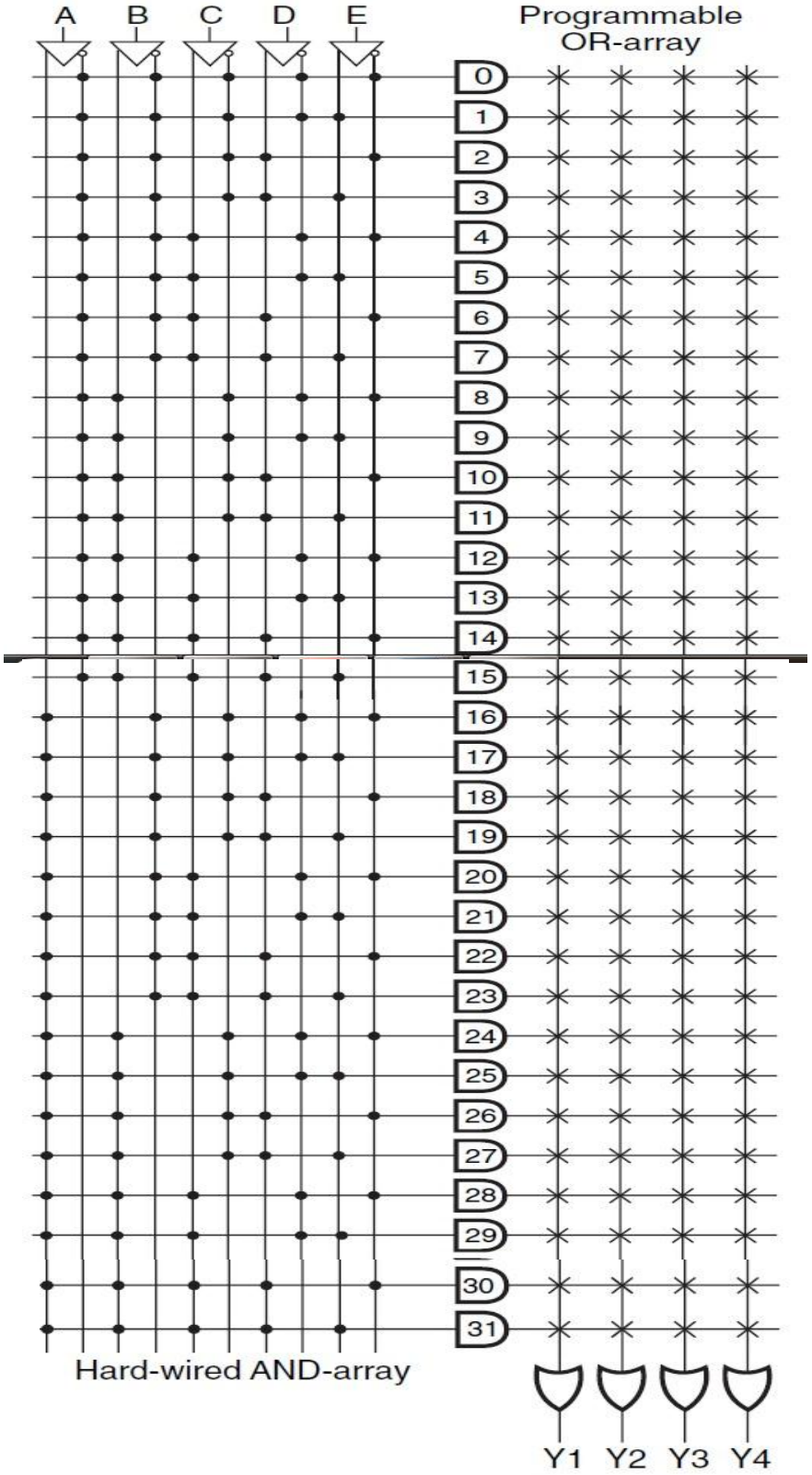
A
B
C
D
E
Programmable
OR-array
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
Hard-wired AND-array
Y1
Y2
Y3
Y4

Fig.3

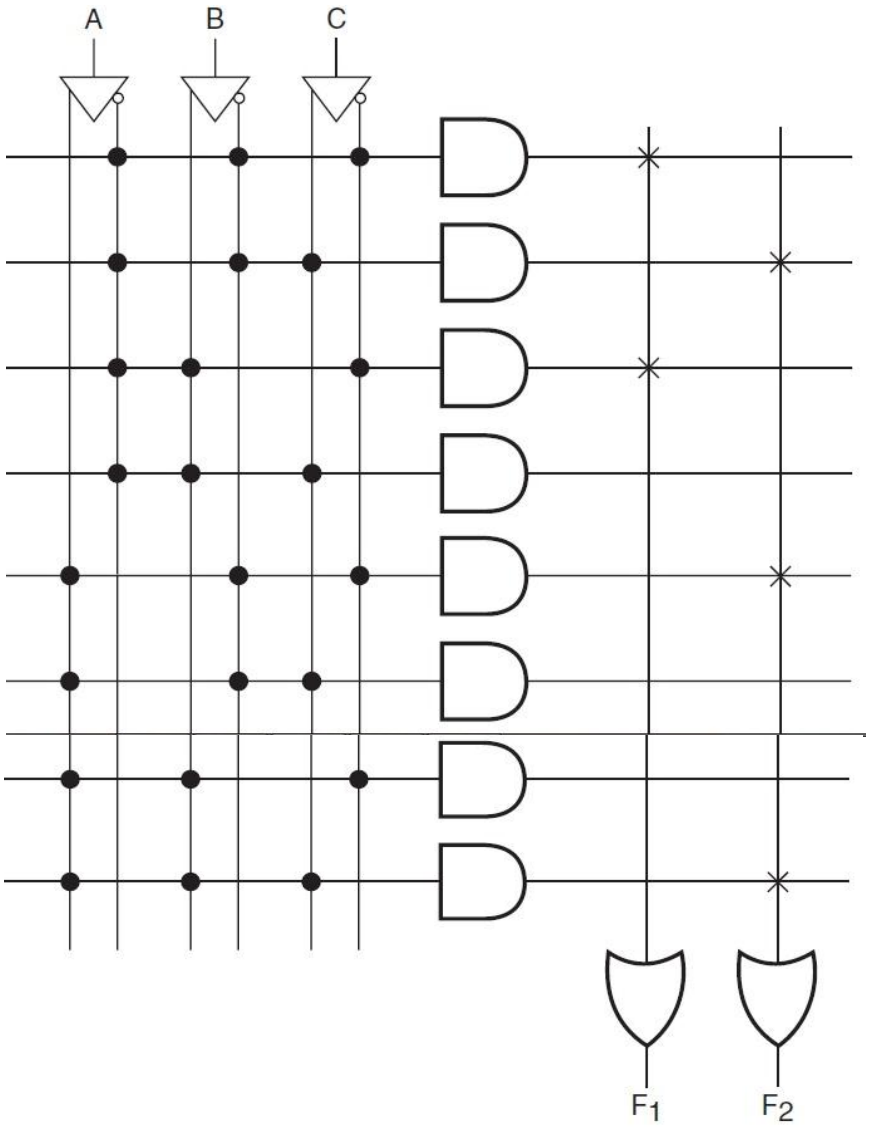

Fig.4

Ex.1/Determine the size of the PROM required for implementing the following logic circuits:

(a) a binary multiplier that multiplies two four-bit numbers;

(b) a dual 8-to-1 multiplexer with common selection inputs;

(c) a single-digit BCD adder/ subtractor with a control input for selection of operation.

Sol./ (a) The number of inputs required here would be eight. The result of multiplication would be in eight

bits. Therefore, the size of the PROM= $2^8 \times 8 = 256 \times 8$.

(b) The number of inputs $= 8 + 8 + 3 = 19$ (the number of selection inputs= 3). The number of outputs

$= 2$. Therefore, the size of the PROM $= 2^{19} \times 2 = 512K \times 2$.

(c) The number of inputs= $4 + 4 + 1$ (carry-in)+1 (control input)=10.

The number of outputs= 4 (sum or subtraction output bits)+1 (carry or borrow bit)= 5. The size

of the PROM= $2^{10} \times 5 = 1024 \times 5 = 1K \times 5$.

❖ ***Programmable Logic Array***

A programmable logic array (PLA) enables logic functions expressed in sum-of-products form to be implemented directly. It is similar in concept to a PROM. However, unlike a PROM, the PLA does not provide full decoding of the input variables and does not generate all possible minterms. While a PROM has a fixed AND gate array at the input and a programmable OR gate array at the output, a PLA device has a programmable AND gate array at the input and a programmable OR gate array at the output. In a PLA device, each of the product terms of the given Boolean function is generated by an AND gate which can be programmed to form the AND of any subset of inputs or their complements. The product terms so produced can be summed up in an array of programmable OR gates. Figure 5 shows the internal architecture of a PLA device with four input lines, eight product lines and four output lines. That is, the programmable AND gate array has eight

AND gates. Each of the AND gates here has eight inputs, corresponding to four input variables and their complements. The input to each of the AND gates can be programmed to be any of the possible 16 combinations of four input variables and their complements. Four OR gates at the output can generate four different Boolean functions, each having a maximum of eight minterms out of 16 minterms possible with four variables. The logic diagram depicts the unprogrammed state of the device. The internal architecture shown in Fig.5 can also be represented by the schematic form of Fig.6. PLAs usually have inverters at the output of OR gates to enable them to implement a given Boolean function in either AND-OR or AND-OR-INVERT form. Figure7 shows a generalized block schematic representation of a PLA device having n inputs, m outputs and k product terms, with n, m and k respectively representing the number of input variables, the number of OR gates and the number of AND gates. The number of inputs to each OR gate and each AND gate are k and 2n respectively. A PLA is specified in terms of the number of inputs, the number of product terms and the number of outputs. As is clear from the description given in the preceding paragraph, the PLA would have a total of 2Kn+Km programmable interconnections. A ROM with the same number of input and output lines would have $2^n \times$ m programmable interconnections.

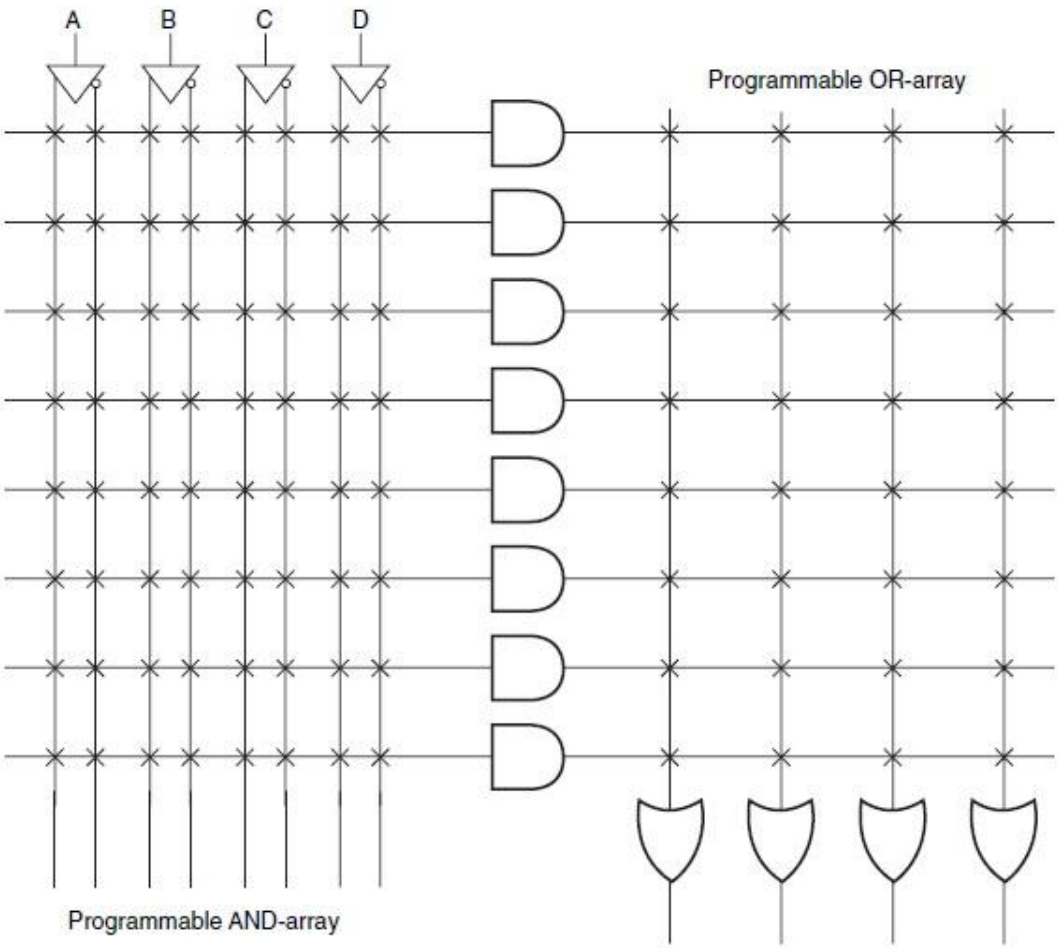

Fig. 5

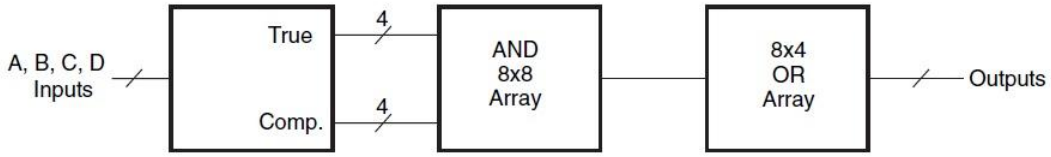

Fig. 6

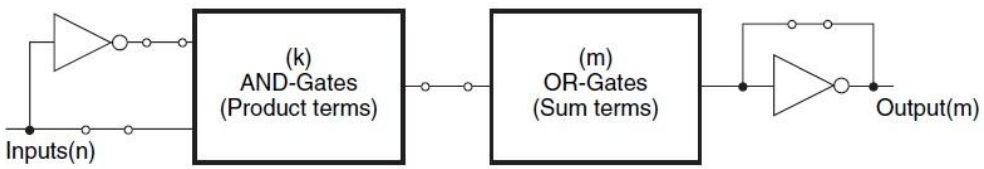

Fig. 7

Ex.2/ Show the logic arrangement of both a PROM and a PLA required to implement a binary full adder.

Sol: The Boolean expressions for sum S and carry-out Co can be written as follows:

$S = \sum(1, 2, 4, 7)$

$Co = \sum(3, 5, 6, 7)$

Figure 8 shows the implementation with an 8×2 PROM while figure 9 shows the representation of full adder using PLA

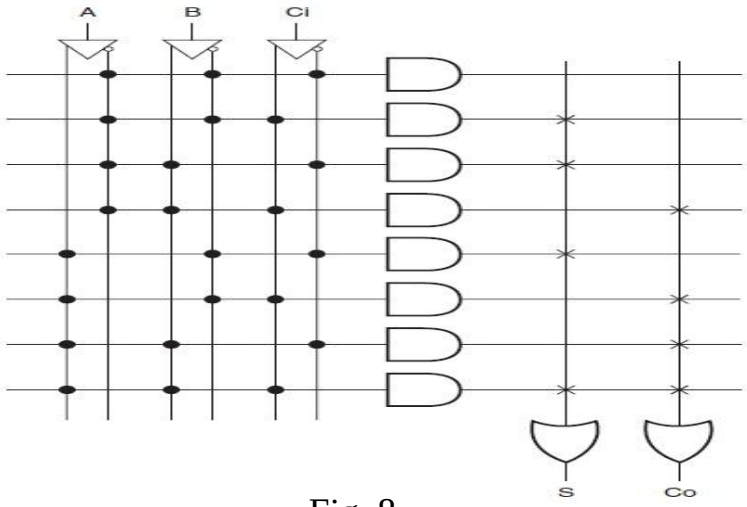

Fig. 8

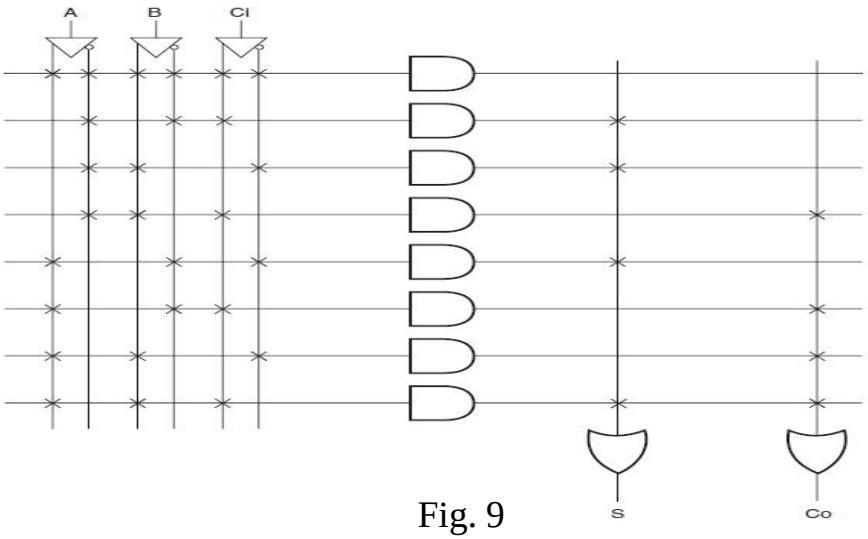

Fig. 9

Ex.3/ Design a logic cct. that find the square root of a four bit input, if the input has not the out is logic zero and produce a carry.

Sol:

A	B	C	D	y_1	y_2	C_o
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	0	1
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	0	1
1	0	0	1	1	1	0
1	0	1	0	0	0	1
1	0	1	1	0	0	1
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	0	1

The Boolean expressions of the outputs are

$$y_1 = A\,\bar{B}CD + A\,\bar{B}\,\bar{C}D$$

$$y_2 = ABC\,\bar{D} + A\,\bar{B}\,\bar{C}D$$

$$C_O = \bar{A}\bar{B}C\,\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\,\bar{D} + A\bar{B}C\,\bar{D} + A\bar{B}CD + AB\bar{C}\,\bar{D} + AB\bar{C}\,D + ABC\,\bar{D} + ABCD$$

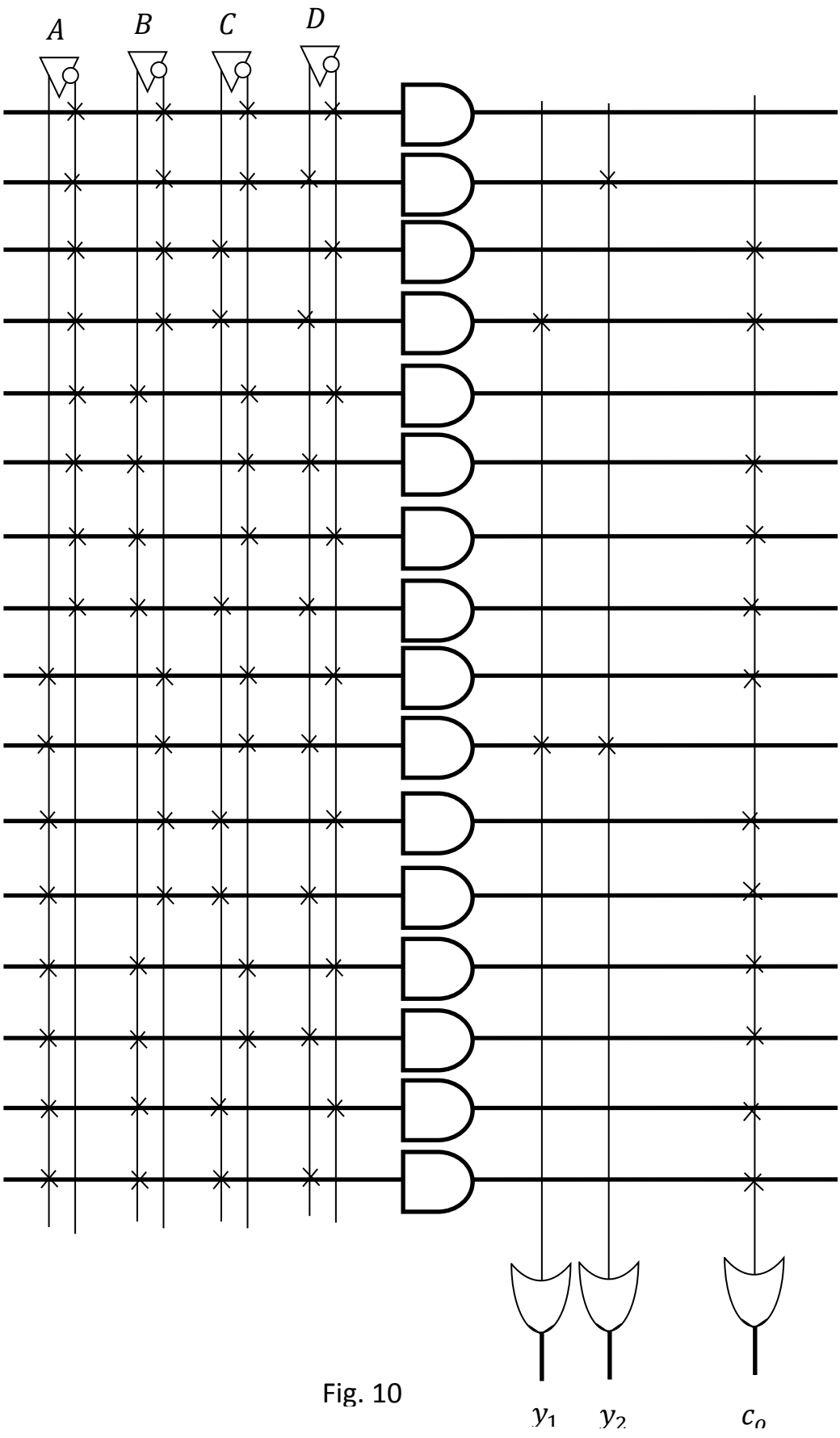

Fig. 10

❖ Programmable Logic Array

Programmable array logic (PAL) architecture has a programmable AND array at the input and a fixed OR array at the output. The OR array is fixed and the AND outputs are equally divided between available OR gates. Figure 11 shows the internal architecture of a PAL device that has four input lines, an array of eight AND gates at the input and two OR gates at the output, to introduce readers to the arrangement of various building blocks inside a PAL device and allow them a comparison between different programmable logic devices.

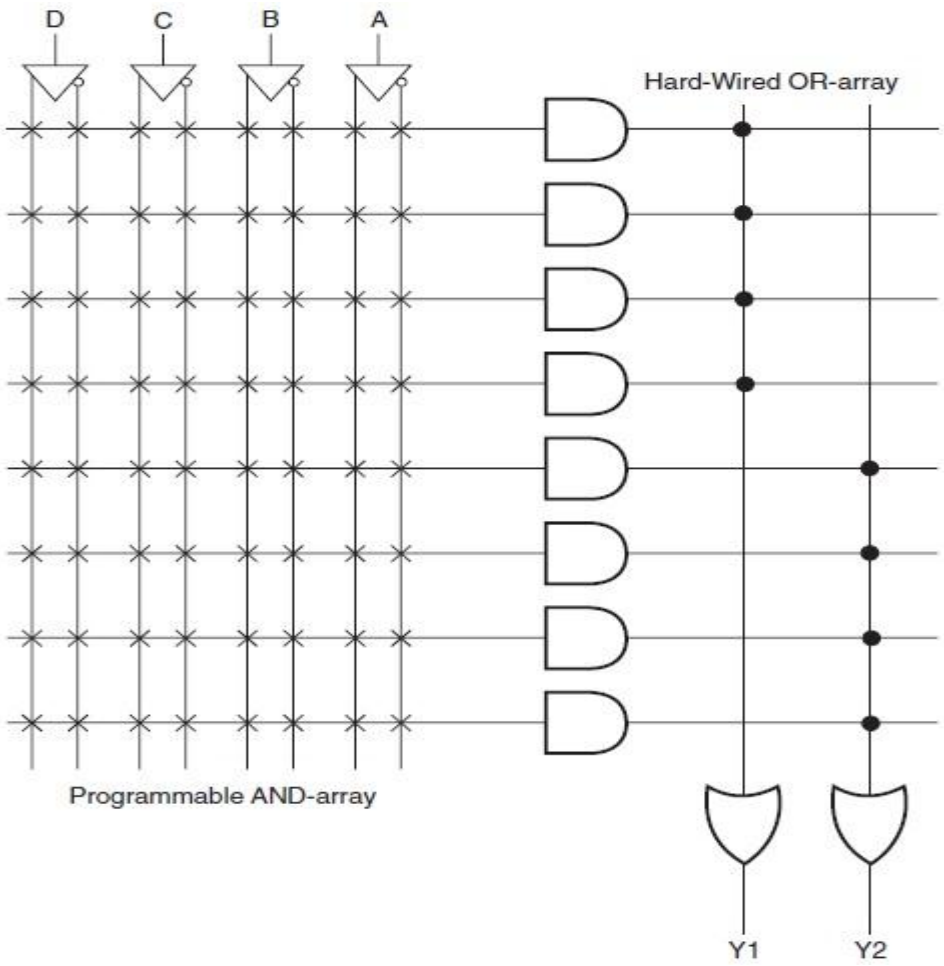

Fig. 11

Figure 12 shows the block schematic representation of the generalized architecture of a PAL device.

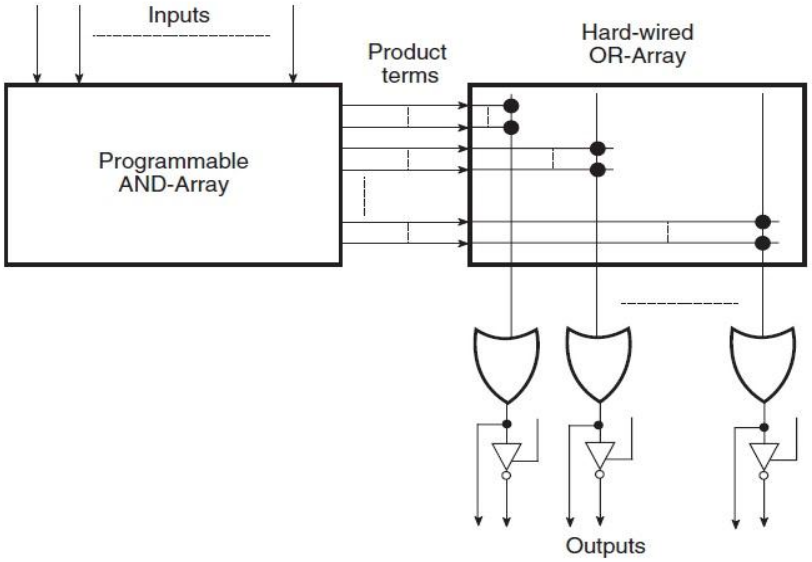

Fig. 12

Ex.3/ The T.T. below is the function table of a converter. Starting with the Boolean expressions for the four outputs (P, Q, R, S), minimize them using Karnaugh maps and then hardware-implement this converter with a suitable PLD with PAL architecture.

A	B	C	D	P	Q	R	S
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

Sol: From the T.T. the expressions of the outputs variables are

$$P = \overline{A}.B.\overline{C}.D + \overline{A}.B.C.\overline{D} + \overline{A}.B.C.D + A.\overline{B}.\overline{C}.\overline{D} + A.\overline{B}.\overline{C}.D$$

$$Q = \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.B.\overline{C}.D$$

$$R = \overline{A}.\overline{B}.C.\overline{D} + \overline{A}.\overline{B}.C.D + \overline{A}.B.\overline{C}.\overline{D} + \overline{A}.B.\overline{C}.D + \overline{A}.B.C.\overline{D} + \overline{A}.B.C.D$$

$$S = \overline{A}.\overline{B}.\overline{C}.D + \overline{A}.\overline{B}.C.\overline{D} + \overline{A}.B.C.D + A.\overline{B}.\overline{C}.\overline{D}$$

These expressions can be simplified using K.M. to be written as

$$P = B.D + B.C + A$$

$$Q = B.\overline{C}$$

$$R = B + C$$

$$S = \overline{A}.\overline{B}.\overline{C}.D + B.C.D + A.\overline{D} + \overline{B}.C.\overline{D}$$

The PAL logic cct. is given in figure 13

❖ Generic Array Logic

A generic array logic (GAL) device is similar to a PAL device and was invented by Lattice Semiconductor. It differs from a PAL device in that the programmable AND array of a GAL device can be erased and reprogrammed. Also, it has reprogrammable output logic. This feature makes it particularly attractive at the device prototyping stage, as any bugs in the logic can be corrected by reprogramming. A similar device called PEEL (Programmable Electrically Erasable Logic) was introduced by the International CMOS Technology.

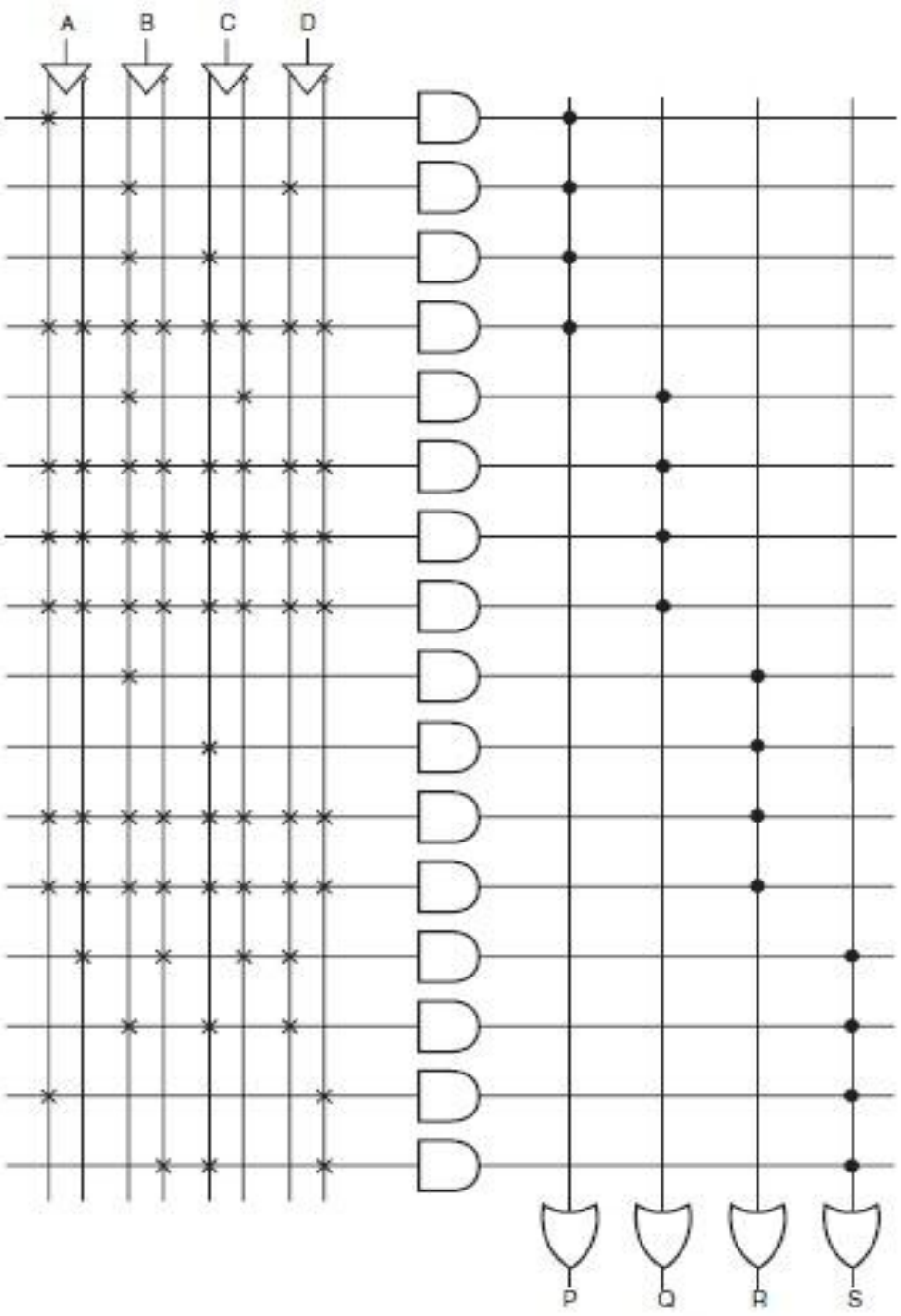

Fig 13