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.

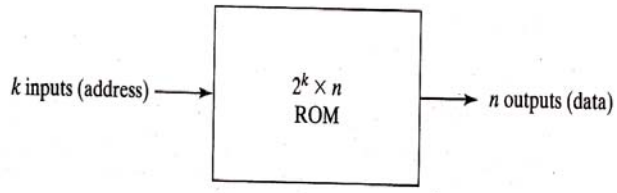


Fig 1 ROM Block Diagram

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.

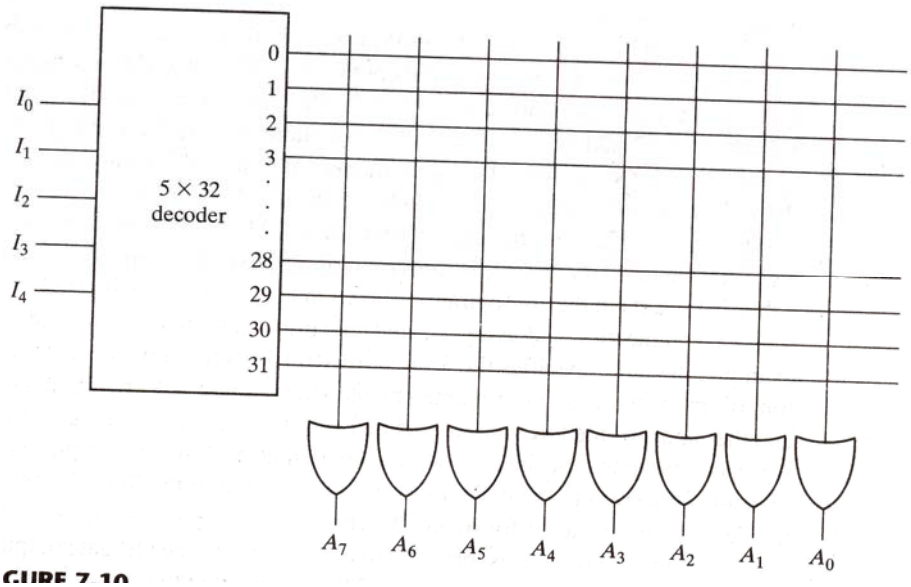


Fig 2 Internal Logic of a 32 X 8 ROM

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.

The 256 intersections of the above figure are programmable. A programmable connection

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

Table 1 ROM Truth Table

Between the lines is logically equivalent to a switch that can be altered to either be close or open .The programmable intersection between two lines is sometimes called cross point.

The internal binary storage of a ROM is specified by a truth table that shows the word content in each address. The Table 1 shows the five inputs under which are listed all 32 addresses. At each address, there is stored a word of 8 bits, which is listed under the outputs columns .The table shows only the first four and the last four words in the ROM. The complete table must include the list of all 32 words.

The hardware procedure that programs the ROM results in blowing fuse links according to a given truth table. The programming of ROM according to the truth table is given in Table 1 , results in the configuration shown in Figure 2.

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 A_6 , A_3, A_2 and A_0. 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 A_7 , A_5, A_4 and A_1.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.

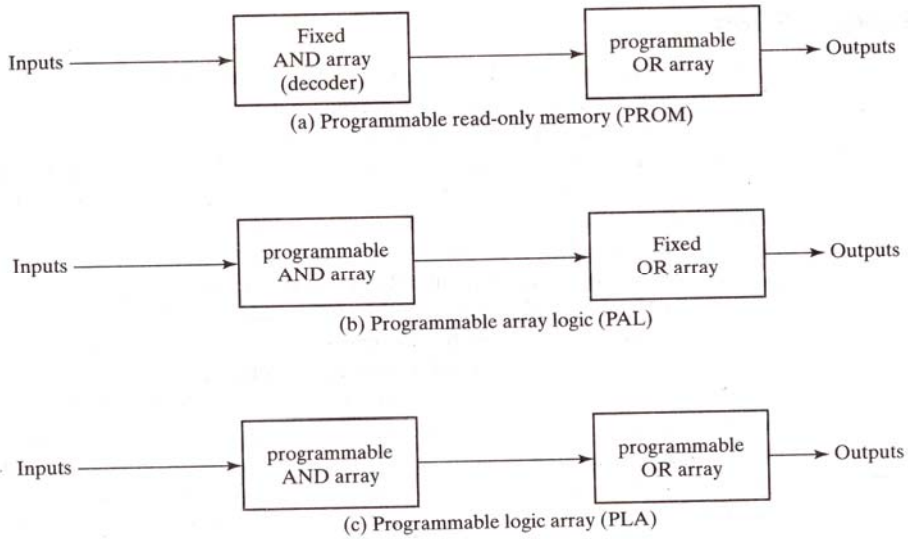
There are three major types of combinational PLDs and they differ in the placement of the programmable connections in the AND-OR array. The Figure 3 shows the configuration of three PLDs.

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

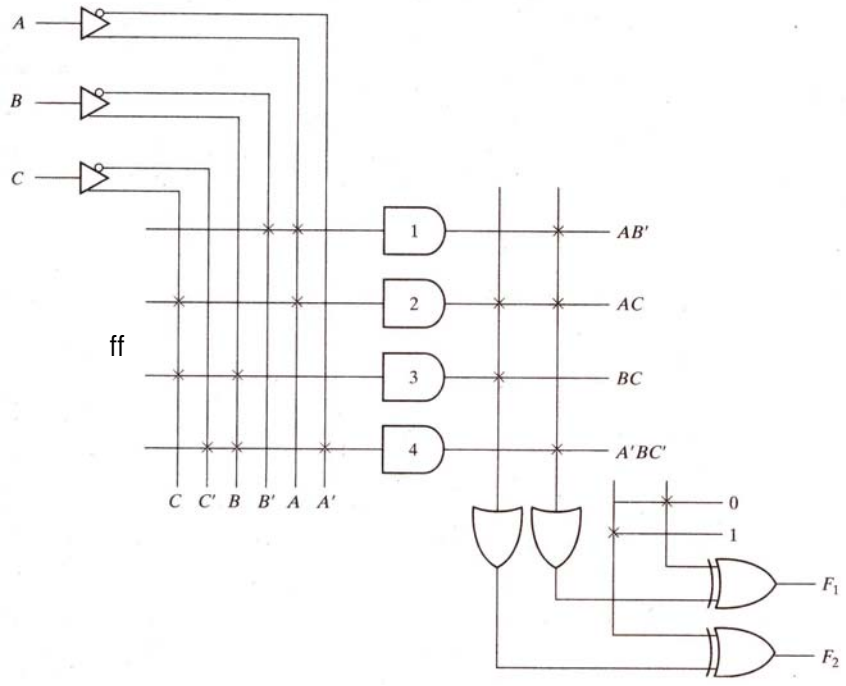
flexible PLD is the programmable logic array (PLA) where both AND and OR arrays can be programmed. The product terms in the AND array may be shared by any OR gate to provide the required sum of products implementation.

Fig 3 Basic configuration of three PLDs.



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. The internal logic of a PLA with three inputs and two outputs is shown in Figure 4.



ff

Fig 4 PLA with 3 inputs ,4 product Terms and 2 outputs

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 particular Boolean functions implemented in the PLA of Fig 4 are

$$F_1 = AB' + AC + A'BC'$$

$$F2 = (AC + BC)'$$

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.

For each output variable, we may have a T (for True) or C (for complement) for programming the XOR gate. The product terms listed on the left are not part of the table,they are included for reference only. For each product term the inputs are arked with 1,0 or --. I f a variable in the product term appears in its true form, the corresponding input variable is marked with a 1. If it appears complemented the corresponding input variable is marked with a 0. If the variable is absent in the product term it is marked with a dash.The paths between the inputs and the AND gates are specified under the column heading inputs in the programming table. A 1 in the input column specifies a connection from the input variable to the AND gate. A 0 in the input column specifies a connection from the complement of the variable to input of the AND gate. A dash specifies a blown fuse in both the input variable and its complement. The paths between the AND gates are specified under the column heading outputs. The output variables are marked with 1's for those product terms that are included in the function. Each product term that has a 1 in the output column requires a path from the output of the AND gate to the input of the OR gate behaves like a 0.

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	–

Table 2 PLA programming Table

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.

The PLA may be mask programmable or Field programmable with mask programming. The customer submits a PLA program table to the manufacturer .This table is used by the vendor to produce a custom made PLA.

A second type of PLA available is called a field programmable logic array or FPLA. The FPLA can be programmed by the user by means of a commercial hardware programmer unit.

REALIZATION OF A BOOLEAN FUNCTION USING PLA.

Implement the following two Boolean functions with a PLA

F1 (A,B,C) = Σ (0,1,2,4)

F2 (A,B,C) = Σ (0,5,6,7)

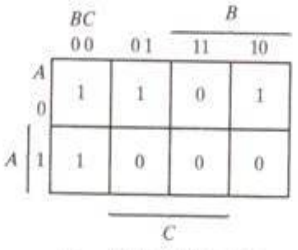
The two functions are simplified as shown in maps. Both the true and complement of the functions are simplified in sum of products.

The combination that gives a minimum number of product terms is minimum number of product terms is

F1 = (AB+AC+BC)'

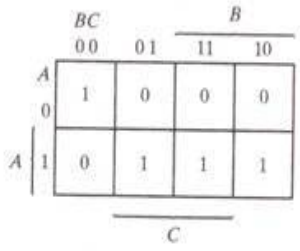
F2 = AB+AC+A'B'C'

This gives four district product terms AB, AC, BC and A'B'C'. The PLA programming table for this combination is shown in the Table 3.



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

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



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

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

PLA 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

Table 3 PLA programming Table

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.

Commercial PAL devices contain more gates than the one shown in Figure 4 .

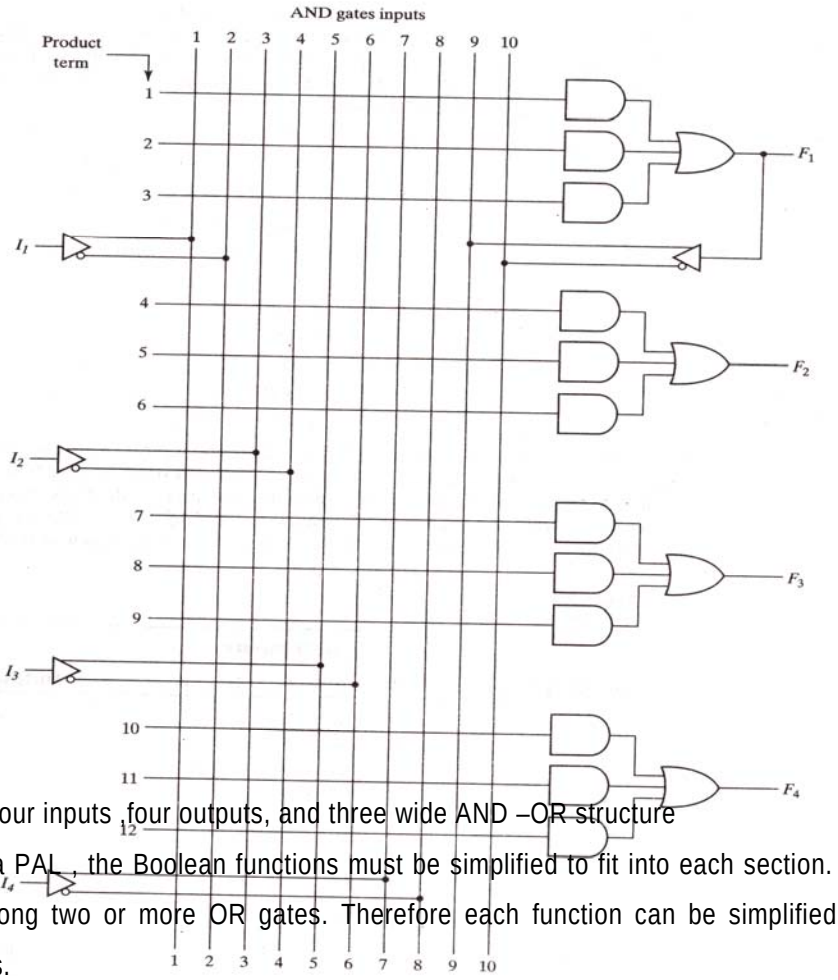


Fig 5 PAL with Four inputs ,four outputs, and three wide AND –OR structure

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

REALIZATION OF A COMBINATIONAL CIRCUIT USING PAL

Consider the following Boolean functions given in sum of minterms.

W (A,B,C,D) = Σ (2,12,13)

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

Y (A,B,C,D) = Σ (0,2,3,4,5,6,7,8,10,11,15)

Z (A,B,C,D) = Σ (1,2,8,12,13)

Simplify the four functions to a minimum number of terms result 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=W+AC'D'+A'B'C'D.

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 shown in Table 4

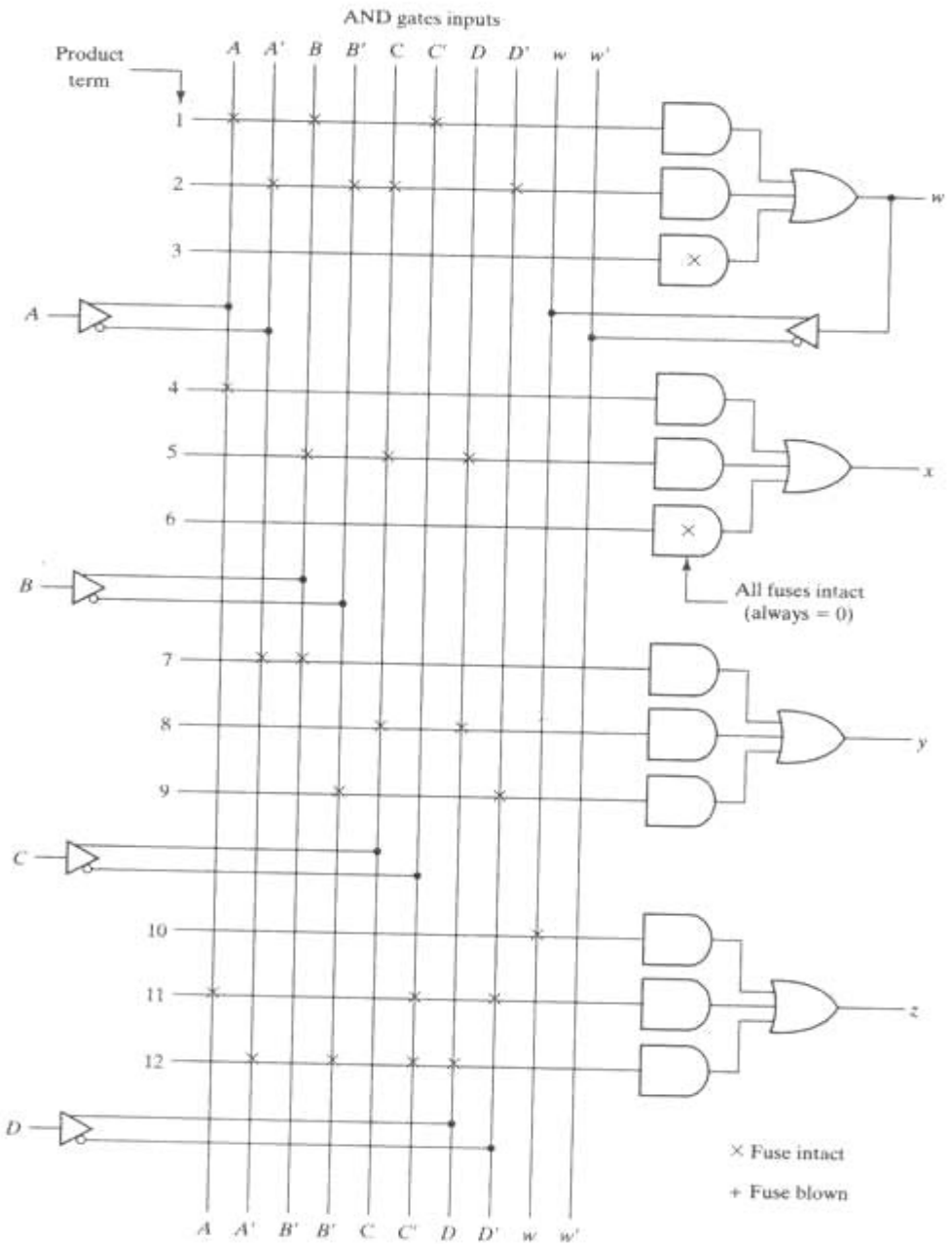
	AND Inputs					
Product Term	A	B	C	D	W	Outputs
1	1	1	0	–	–	$w = ABC'$
2	0	0	1	0	–	$+ A'B'CD'$
3	–	–	–	–	–	
4	1	–	–	–	–	$x = A$
5	–	1	1	1	–	$+ BCD$
6	–	–	–	–	–	
7	0	1	–	–	–	$y = A'B$
8	–	–	1	1	–	$+ CD$
9	–	0	–	0	–	$+ B'D'$
10	–	–	–	–	1	$z = w$
11	1	–	0	0	–	$+ AC'D'$
12	0	0	0	1	–	$+ A'B'C'D$

Table 4 PAL Programming Table

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.

The fuse map for the PAL as specified in the programming table is shown in fig 6

Fig. 6 Fuse map for Pal



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.

SEQUENTIAL PROGRAMMABLE DEVICES

Digital systems are designed using flip flops and gates. Since the combinational PLD consists of only gates, it is necessary to include external flip flops when they are used in the design. Sequential programmable devices include both gates and flip flops.

The device can be programmed to perform a variety of sequential circuit functions. There are several types of sequential programmable devices available commercially and each device has vendor-specific, variant within each type.

There are three major types

1. Sequential programmable logic device (SPLD)
2. Complex programmable logic device (CPLD)
3. Field Programmable gate array (FPGA)

The sequential PLD is referred to as a simple PLD to differentiate it from the complex PLD. SPLD includes flip flops within the integrated circuit chip in addition to the AND-OR array. The sequential circuit is shown in figure 7.

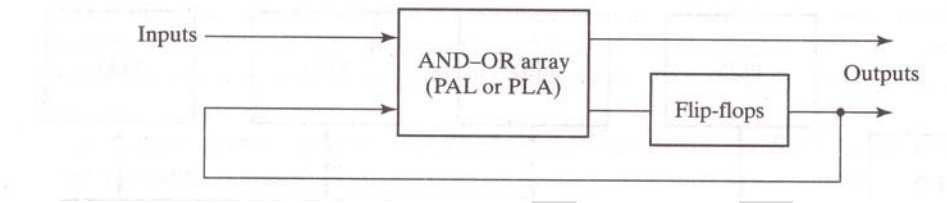


Fig.7 Sequential Programmable Logic Device

A PAL or PLA is modified by including a number of flip flops connected to form a register. The circuit outputs can be taken from the OR gates or from the outputs of the flip flops.

Additional programmable connections are available to include the flip flop outputs in the product terms formed with the AND array. The flip flops may be of the D or the JK type.

[The first programmable device developed to support sequential circuit implementation is the field programmable logic sequencer (FPLS). A typical FPLS is organized around a PLA with several outputs driving flip flops. The flip flops are flexible in that they can be programmed to operate as either JK or D type. The FPLS did not succeed commercially.] –not needed.

The configuration mostly used for SPLD is the combinational PAL together with D flip flops. A PAL that includes flip flops is referred to as a registered PAL to signify that the device contains flip flops in addition to the ANDOR array. Each section of

an SPLD is called a macrocell. A macrocell is a circuit that contains a sum of products combinational logic function and an optional flip flop.

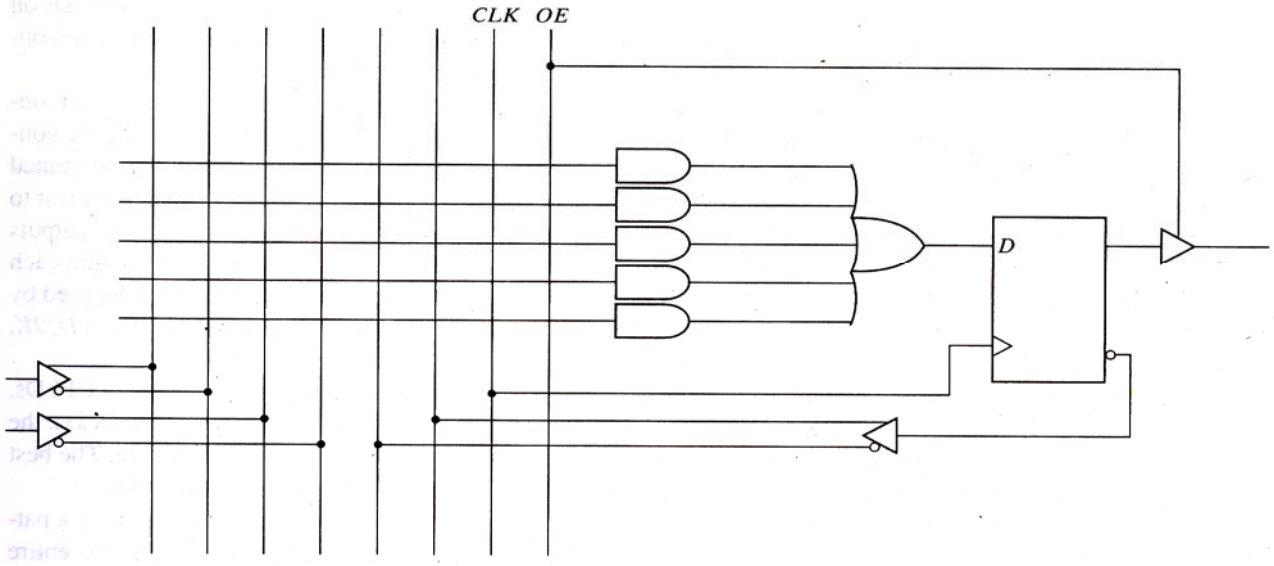


Fig.8 Basic Macrocell Logic

The figure 8 shows the logic of a basic macrocell. The AND-OR array is the same as in the combinational PAL. The output is driven by an edge triggered D flip flop. The flip flop is connected to a common clock input and changes state on a clock edge. The output of the flip flop is connected to a three state buffer controlled by an output enable signal marked in the diagram as OE. The output of the flip flop is fed back into one of the inputs of the programmable AND gates to provide the present state condition for the sequential circuit. A typical SPLD has from 8 to 10 macrocells within one IC packager. All the flip flops are connected to the common UK input and all three state buffers are controlled by the EO input.

The design of a digital system using PLD often requires the connection of several devices to produce the complete specification. For this type of application, it is more economical to use a complex programmable logic device.

A CPLD is a collection of individual PLDs on a single IC. A programmable interconnection structure allows the PLDs to be connected to each other in the same way that can be done with individual PLDs. Figure 9

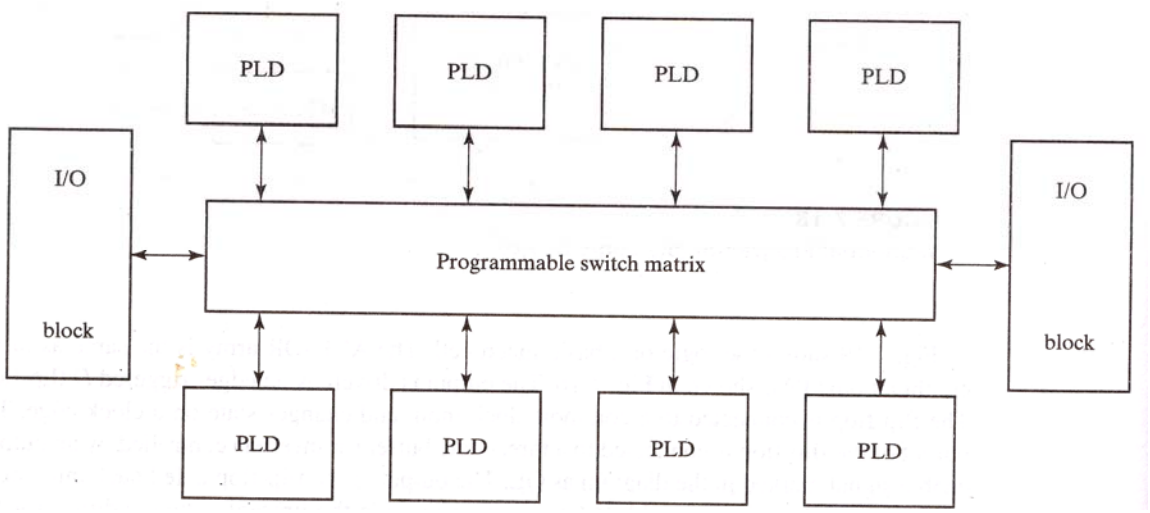


Fig.9 General CPLD Configuration

The figure shows a general configuration of a CPLD. It consists of multiple PLDs interconnected through a programmable switch matrix. The input/output blocks provide the connections to the IC pins. Each I/O pin is driven by a three state buffer and can be programmed to act as input or output. The switch matrix receives inputs from the I/O block and directs it to the individual macrocells. The selected outputs from microcells are sent to the outputs as needed. Each PLD typically contains from 8 to 16 macrocells. The macrocells within each PLD are usually fully connected. If a macrocell has unused product terms they can be used by other nearby macrocells.

- An ASIC is an Application Specific Integrated Circuit.

Examples of ICs that are not ASICs include : standard parts such as memory chips sold as a commodity item - ROMs, DRAM, SRAM, microprocessor, TTL or TTL equivalent ICs at SSI, MSI and LSI levels.

Examples of ICs that are ASICs include: a chip for a toy bear that talks; a chip for a satellite a chip designed to handle to interface between memory and a microprocessor for a workstation CPU.

If you can find it in a data book, then it is probably not an ASIC. Two ICs that might or might not be considered ASICs are a controller chip for a PC and a chip for a modem. Both these examples are specific to an application but are sold to many different system vendors. ASICs such as these are sometimes called application specific standard products (ASSP/S)

TYPE OF ASICs

ICs are made on a thin circular silicon wafer with each wafer holding hundreds of die. The transistors and wiring are made from many layers built on top of on another. Each successive mask layer has a pattern that is defined using a mask similar to a glass photographic slide. The first half dozen or so layers define the transistors. The last half dozen are so layers define the metal wires between the transistors (the interconnect).

A full custom IC includes some logic cells that are customized and all mask layers that are customized (eg) Microporcessor.

Designers spend many hours squeezing the most out of every last square micron of microprocessor chip space by hand. Full custom ICs are the most expensive to manufacture and to design. The manufacturing lead time is typically eight weeks for a full custom IC. These ICs are of ten intended for a specific application so they are called Full Custom ASICs.

In semicustom ASICs, all the logic cells are predesigned and some of the mask layers are customized. Using predesigned cells from a cell library makes the job of the designers much easier.

There are two types of semicustom ASICs.

(1) Standard cell based ASICs.
(2) Gate array based ASICs.

In programmable ASICs all the logic cells are predesigned and none of the mask layers are customized. There are two types of programmable ASICs: the programmable logic device and the newest member of the ASIC family the field prommable gate array.

FULL CUSTOM ASICs

In a full custom ASIC an engineer designs some or all of the logic cells, circuits or layout specifically for one ASIC. This means the designer abandons the approach of using protested and precharacterized cells for all or part of that design. It makes sense to take this approach only if there are no suitable existing cell libraries available, because existing cell libraries are not fast enough or the logic cells are not small enough or consume too much power.

Bipolar technology has been used for precision analog functions. There are some fundamental reasons for this. In all integrated circuits. The matching of component characteristics between chips is very poor, while the matching of characteristics between components on the same chip is excellent.

Suppose we have transistors T1,T2 and T3 on an analog/digital ASIC. The three transistors are all the same size and are constructed in an identical fashion. Transistors T1 and T2 are located adjacent to each other and have the same orientation. Transistor T3 is the same size as T1 and T2 but is located on the other side of the chip from T1 and T2 and has a different orientation.

If we were to make measurement of the characteristics of transistors T1, T2 and T3 we would find the following.

- T1 will have virtually identical characteristics to T2 on the same IC. The transistors are said to be matched.
- Transistors on ICs from different wafers in the same wafer lot will not match very well.
- Transistors on ICs from different wafer lots will match very poorly.

For many analog designs the close matching of transistors is crucial to circuit operation. For these circuit designs pairs of transistors are used, located adjacent to each other. Device physics dictates that a pair of bipolar transistors will always match more precisely than CMOS transistors.

The use of CMOS technology for analog functions is increasing.

There are two reasons for this.

1. CMOS is now by far the most widely available IC Technology.
2. Increased levels of integration require mixing analog and digital functions on the same IC. This has forced designers to find ways to use CMOS technology to implement analog functions.

STANDARD CELL BASED ASICs

A cell based ASIC shortly called as CBIC (sea bick) uses predesigned logic cells like AND gates, OR gates known as standard cells.

A standard cell areas in a CBIC are built of rows of standard cells like a wall built of bricks. The standard cell areas may be used in combination with larger predesigned cells, perhaps microcontrollers or even microprocessor known as megacells.

The ASIC designer defines only the placement of the standard cells and the interconnect in a CBIC. The standard cells can be placed anywhere on the silicon which means that all the mask layers of a CBIC are customized and are unique to a particular customer.

Advantages Of CBICs:

1. Designers save time, money.
2. Reduce risk by using a predesigned, pretested and precharacterized standard cell library.

Disadvantages

1. E1 Expense of designing or buying the standard cell library.
2. Time needed to fabricate all layers of the ASIC.

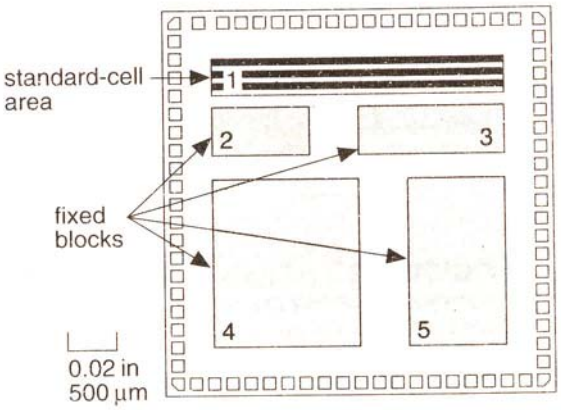
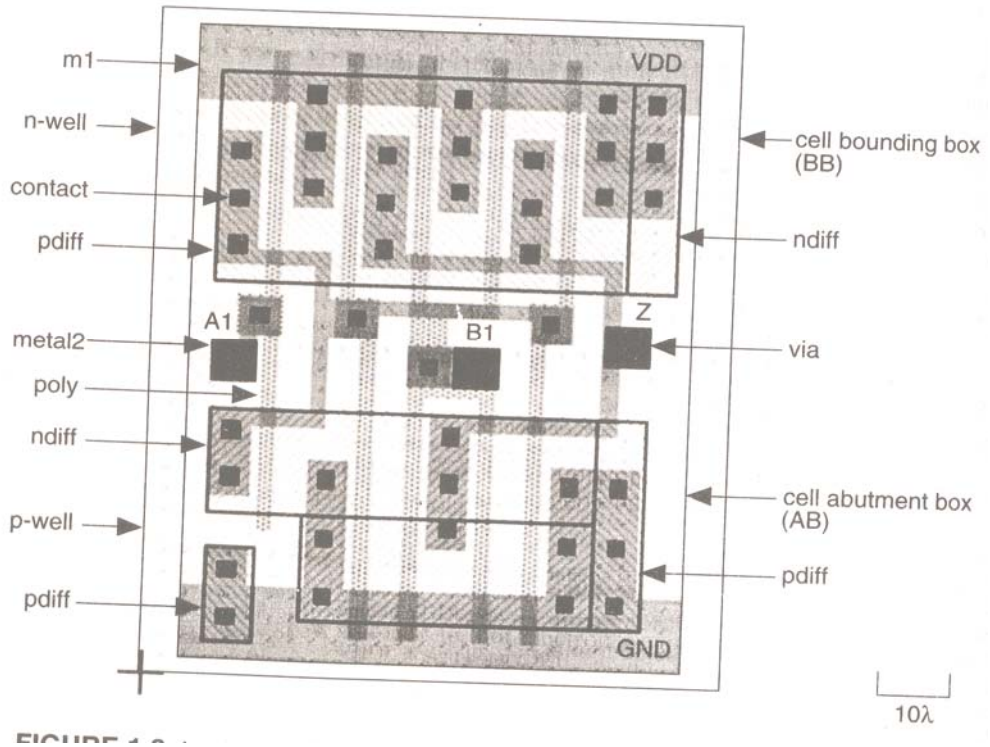


Fig.10 Cell Based ASIC

Figure 10 shows a CBIC. The important features of this type of ASIC are as follows.

- All mask layers are customized
- Custom blocks can be embedded
- Manufacturing lead time is about eight weeks.

Standard cells are designed to fit together like bricks in a wall. The figure 11 shows an example of a simple standard cell.



Power and ground buses run horizontally on metal lines inside the cells.

Fig.11 Layout of a Standard Cell

Standard cell design allows the automation of the process of assembling an ASIC. Groups of standard cells fit horizontally together to form rows. The rows stack vertically to form flexible rectangular blocks. Then a flexible block built from several rows of standard cells can be connected to other standard cell blocks or other full custom logic blocks.

Both cell based and gate array ASICs use predefined cells but there is a difference we can change the transistor sizes in a standard cell to optimize speed and performance but the device sizes in a gate array are fixed. The trade off between area and performance is made at the library level for a standard cell ASIC.

Modern CMOS ASICs use two, three or more levels of metal for interconnect. This allows wires to cross over different layers in the same way that we use copper traces on different layers on pcb. In a two level metal CMOS technology connections to the standard cell inputs and outputs are usually made using the second level of metal.

A connection that needs to cross over a row of standard cells uses a feed through. The term feed through can refer either to the piece of metal that is used to pass a signal through a cell or to a space in a cell waiting to be used as a feed through.

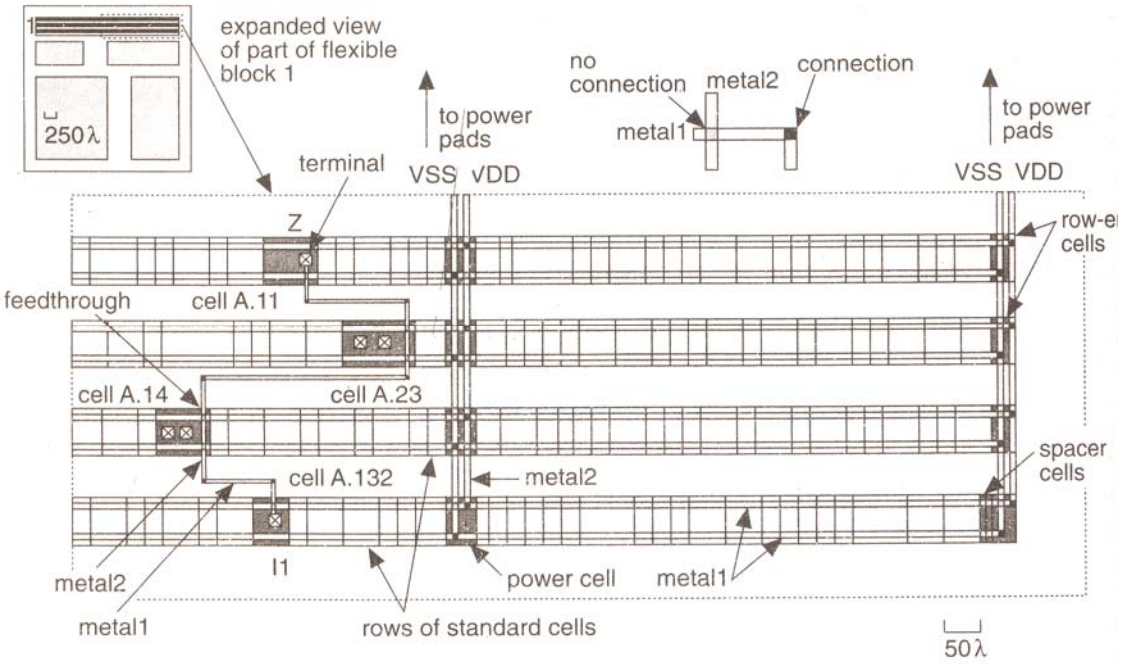


Fig.12 ASIC with Three Layers

Figure 12 shows two feed through one in cell A.14 and one in cell A.23.

In both two levels and three levels metal technology, the power buses inside the standard cells normally use the lowest layer of metal. The width of each row of standard cells is adjusted so that they may be aligned using spacer cells. The power buses or rails are then connected to additional vertical power rails using row end cells at the aligned ends of each standard cell block. If the rows of standard cells are long, then vertical power rails can also he run in metal 2 through the cell rows using special power cells that just connect to VDD and GND.

GATE ARRAY BASED ASICs

In a gate array or gate array based ASIC the transistors are predefined on the silicon wafer. The predefined pattern of transistors on a gate array is the base array and the smallest element that is replicated to make the base array is the base cell. Only the top few layers of metal, which define the interconnect between transistors are designed by the designer using custom masks.

The designer chooses from a gate array library of predesigned and precharacterized logic cells. The logic cells in a gate array library are often called macros.

There are different types of gate array based ASICs.

- Channeled gate arrays
- Channelless gate arrays
- Structured gate arrays.

The channeled gate array was the first to be developed but the channelless gate array architecture is now more widely wed.

CHANNELED GATE ARRAY.

The figure 13 shows a channeled gate array. The important features of this type of MGA are

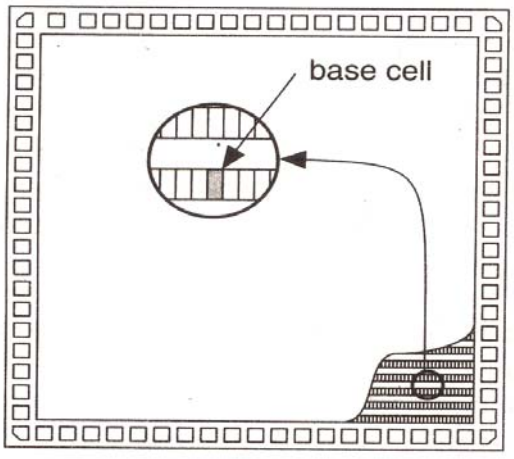


Fig.13 Channeled Gate Array Die

- Only the interconnect is customized
- The interconnect uses predefined spaces between rows of base cells
- Manufacturing lead time is between two days and two weeks.

A channeled gate array is similar to CBIC both use rows of cells separated by channels used for interconnect. One difference is that the space for interconnect between rows of cells are fixed in height in a channeled gate array, whereas the space between rows of cells may be adjusted in a CBIC.

CHANNELLESS GATE ARRAY

Fig 14 shows a channelless gate array. The important features of this type of MGA are as follows:-

- Only some mask layers are customized.
- Manufacturing lead time is between two days and two weeks.

The main difference between a channelless gate array and channeled gate array is that there no predefined areas set aside for routing between cells on a channelless gate array.

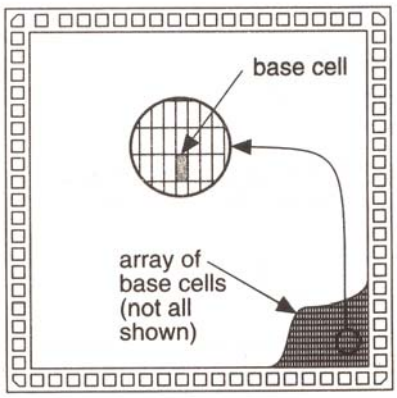


Fig.14 Channelless Gate Array

The logic densisty-the amount of logic that can be implemented in a give silicon area is higher for channelless gate arrays than for channeled gate arrays.

STRUCTURED GATE ARRAY.

An embedded gate array or structured gate array combines some of the feature of CBICs and MGAs. One of the disadvantage of the MGA is the fixed gate array base cell. This makes the implementation of memory very difficult and inefficient.

Figure 15 shows an embedded gate array. The important features of this type of MGA are the following.

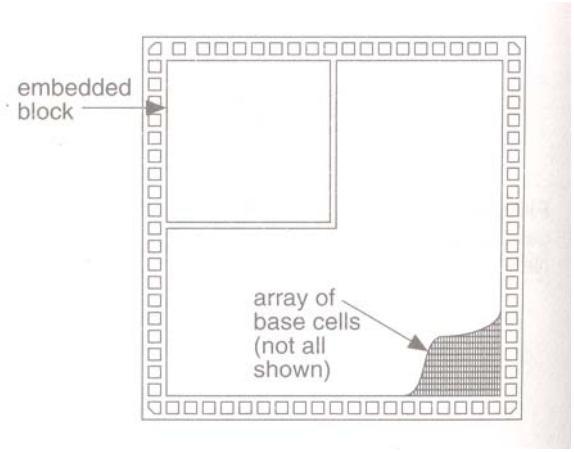


Fig.15 Embedded Gate Array

- Only the interconnect is customized
- Custom blocks can be embedded
- Manufacturing lead time is between two days and two weeks.

An embedded gate array gives the improved area efficiency and increased performance of a CBIC but with the lower cost and faster turn around of an MGA.

PROGRAMMABLE LOGIC DEVICES.

Programmable logic devices (PLDS) are standard ICs that are available in standard configurations from a catalog of parts and are sold in very high volume to many different customers. However, PLDs may be configured or programmed to create a part customized to a specific application and so they also belong to the family of ASICs.

Figure 16 shows a PLD and the following important features that all PLDs have in common.

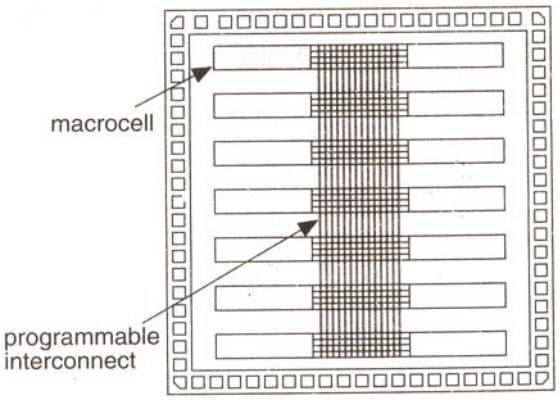


Fig.16 Programmable Logic Device

- ❖ No customized mask layers or logic cells
- ❖ Fast design turnaround
- ❖ A single large block of programmable interconnect

- ❖ A matrix of logic macrocells that usually consist of programmable array logic followed by a flip flop a latch.

The simplest type of programmable IC is a read only memory (ROM). The most common types of ROM use a metal fuse that can be blown permanently which is called a programmable ROM or PROM. An electrically programmable ROM or EPROM, uses programmable MOS transistors whose characteristics are attuned by applying a high voltage. The information can be erased in an EPROM either by using another high voltage (an electrically erasable PROM or EEPROM) or by exposing the device to ultraviolet light (UV- erasable PROM or UVPROM).

There is another type of ROM a mask programmable ROM.

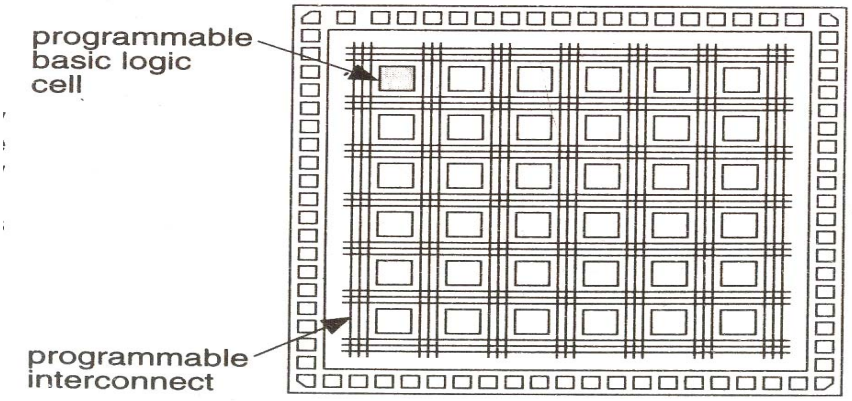
The same programmable technologies used to make ROMs can be applied to more flexible logic structures. By using programmable devices in a large array of AND gates and an array of OR gates. We create a family of flexible and programmable logic devices called logic arrays. There are two types of logic arrays called programmable logic array (PLA) and programmable Array Logic (PAL).

FIELD PROGRAMMABLE GATE ARRAYS

Fig.17 Field Programmable Gate Array

A step above the PLD in complexity is the field programmable gate array (FPGA). There is very little difference between an FPGA and a PLD – an FPGA is usually just larger and more complex than a PLD. In fact, some companies that manufacture programmable ASICs call their products FPGAs and some call them complex PLDs.

Fig 17 illustrates the essential characteristics of an FPGA.



- None of the mask layers are customized.
- A method for programming the basic logic cells and the interconnect.
- The core is regular array of programmable basic logic cells that can implement combinational as well as sequential logic flip flop.
- A matrix of programmable interconnect surrounds the basic logic cells.
- Programmable I/o cells surround the core.
- Design turnaround in few hours.

DESIGN FLOW.

The figure 18 shows the sequence of steps to design an ASIC which is called as design flow. The steps are listed below with a brief description of the function of each step

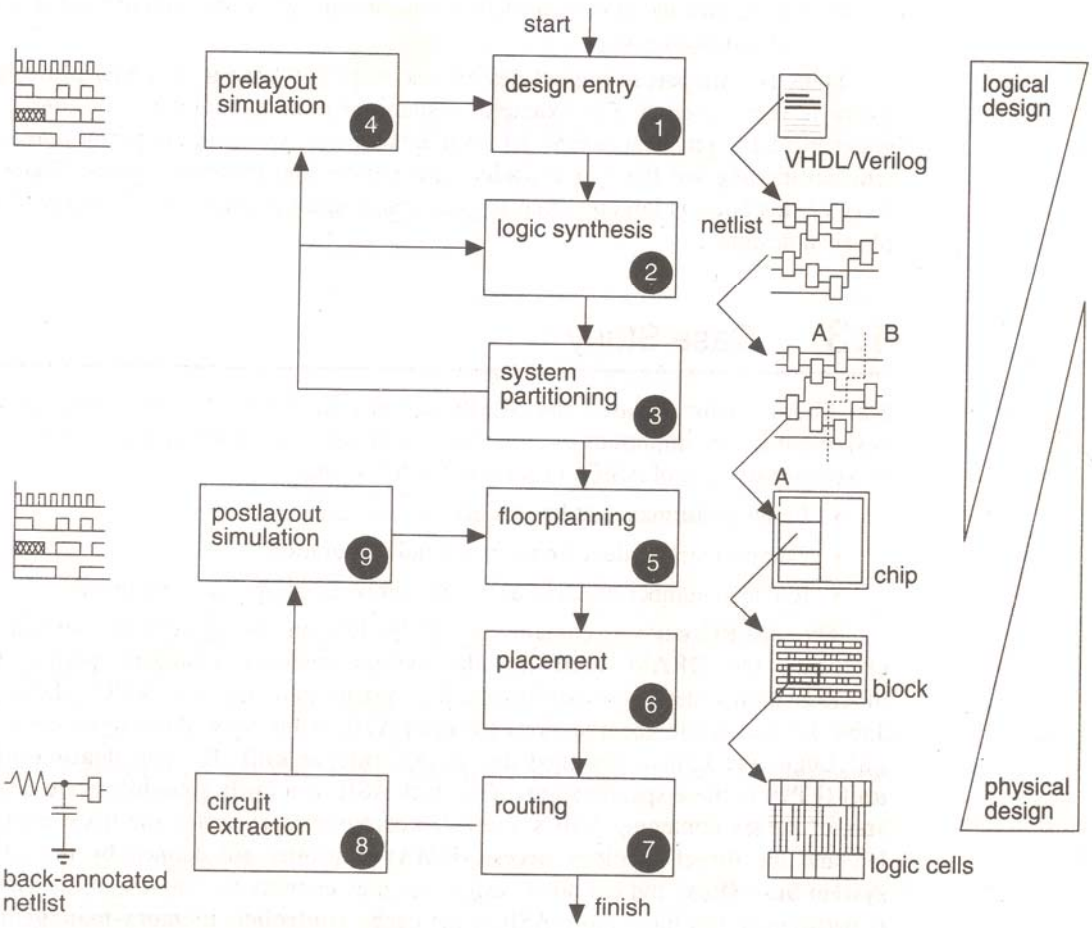


Fig.18 ASIC Design Flow

1) Design entry: Enter the design into an ASIC design system either using a hardware description language (HDL) or schematic entry.
2) Logic synthesis: Use an HDL (VHDL or verilog) and a logic synthesis tool to produce a netlist.
3) System partitioning – Divide a large system into ASIC sized pieces
4) Prelayout simulation – Check to see if the design functions correctly.

5) Floor Planning – Arrange the blocks of the netlist on the chip.
6) Placement – Decide the locations of cells in a block.
7) Routing – Make the connections between cells and blocks.
8) Extraction – Determine the resistance and capacitance of the interconnect
9) Postlayout simulation – Check to see the design still works with the added loads of the interconnect. Steps 1-4 are part of logical design, and steps 5-9 are part of physical design.