UNIT V IMPLEMENTATION STRATEGIES

Full custom and Semi custom design, Standard cell design and cell libraries, FPGA building block architectures, FPGA interconnect routing procedures.

5.1. Full Custom Design:

A number of techniques can be used to design standard cells or larger circuit blocks at the mask level. The oldest and most traditional technique is termed custom mask layout, in which a designer sits in front of a graphics display running an interactive editor and pieces designs together at the geometry level one rectangle at a time. This work is sometimes called polygon pushing. A variation of custom mask design is called symbolic layout. Rather than dealing with rectangles and polygons on various mask levels, the primitives are transistors, contacts, wires, and ports (points of connection). These primitives can also be manipulated by a graphics editor. Some systems allow for a "design rule free" placement of symbolic entities. The actual placement occurs after a spacing process that compacts each primitive as close to its neighbor as possible according to the design rules of the process in use. By using a symbolic layout system, layout topologies can be transported from process to process without a huge amount of effort.

In these times of cell-based design, digital CMOS ICs use custom mask design only for the highest of volume parts such as microprocessor data paths. However, analog and RF designs, cell libraries, memories, and I/O cells still frequently use custom design. Custom design is also worthwhile pedagogically because it completes the link from transistors to systems.

From time to time, we have mentioned software generators as a method of generating physical layout. This kind of idea has been around for a long time and was often referred to as silicon compilation. Complete microprocessors were typical of layouts that were generated. A "correct by construction" method was used to build the layouts hierarchically. In other words, only the mask description was generated, with perhaps a high-level instruction level simulator being the behavioral model. Generators are the most common method used today for library generation.

With modern design flows, many different "views" of a design are required to integrate with the regular path through the design system. For instance, in addition to the behavioral model, a timing view would be needed for timing verification, a logic view might be required for simulation, and a circuit view for layout versus schematic or netlist comparisons would be needed. Software generators can be used to provide all of these views automatically.

Modern versions of the venerable "silicon compiler" can be built in a structured hierarchical manner to generate memories, register files, and other special-purpose structures that can benefit from a customized layout. One of the most straightforward approaches is to write custom placement routines that in essence "hand place" certain standard cells within the row structure of a standard cell design. For instance, you may prefer a certain adder design and have a data path layout for the adder. An algorithm can be written to place the cells on the standard cell grid. In addition, a linked algorithm can be written to generate a gate netlist in an HDL. In this way, both the physical and structural design are captured. The behavior can be represented by an HDL function or module call. Such custom placement can shorten wire lengths and thus improve speed and power.

Custom-designed microprocessors routinely exceed 2 GHz in nanometer processes, while synthesized ASICs typically operate closer to 200–350 MHz. made a fascinating study of the differences between design methods that account for this gap. He identified micro architecture, sequencing overhead, circuit families, logic design, cell design, layout, and design margining as the major differences. Since that study, CAD tools have improved, especially in the integration of synthesis and placement. Custom designs have become more conservative and now use static CMOS circuits and cell libraries similar to their ASIC cousins. Nevertheless, a wide gap still exists.

In a follow up study, examines the gap between ASIC and custom design for power dissipation. Major factors for ASICs consuming more power than custom designs include micro architecture, clock gating, logic style, logic design, technology mapping, cell and wire sizing, voltage scaling, floor planning, process technology, and process variation. The study concludes that synthesizable designs typically consume 3–7× more power 7ustom designs but that better tools and cell libraries can close this gap to 2.6×.

5.2.Semicustom Design Flow:

So far we have defined the components that make up the cell-based design methodology. Fig 8.16 details the traditional sequence of steps to design a semi custom circuit. The steps of what we call the design flow are enumerated in the figure, with a brief description of each.

1.**Design Capture** enters the design into the ASIC design system.A variety of methods can be used to do so ,including schematics and block diagrams; hardware description languages (HDLs) such as VHDL,verilog and more recently, Cderivatives such as system C; behavioral description languages followed by high level synthesis; and imported intellectual property modules.

2.**Logic Synthesis** tools translate modules described using an HDL language into a netlist. Netlists of reused or generated macros can then be inserted to form the complete netlist of the design.

3.**Prelayout Simulation and Verification**: The design is checked for correctness .Performance analysis is performed based on estimated parasitics and layout parameters. If the design is found to be nonfunctional, extra iterations over the design capture or the logic synthesis are necessary.

4.**Floor Planning:** Based on estimated module sizes, the overall outlay of the chip is created. The global power and clock distribution networks also are conceived at that time.

5.**Placement**:The precise positioning of the cells is decided.

6.**Routing**:The interconnections between the cells and blocks are wired.

7.**Extraction**:Amodel of the chip is generated from the actual physical layout ,including the precise device sizes, devices parasitics, and the capacitance and resistance of the wires

8.**Postlayout Simulation and Verification**: The functionality and performance of the chip is verified in the presence of the layout parasitics. If the design is found to be lacking, iterations on the floor planning, placement, and routing might be necessary. Very often this might not solve the problem, and another round of the structural design phase might be necessary

9.**Tape Out**: Once the design is found to be meeting all design goals and functions, a binary file is generated containing all the information needed for mask generation. This file is then sent out to the ASIC vendor or boundary. This important moment in the life of a chip is called tape out.

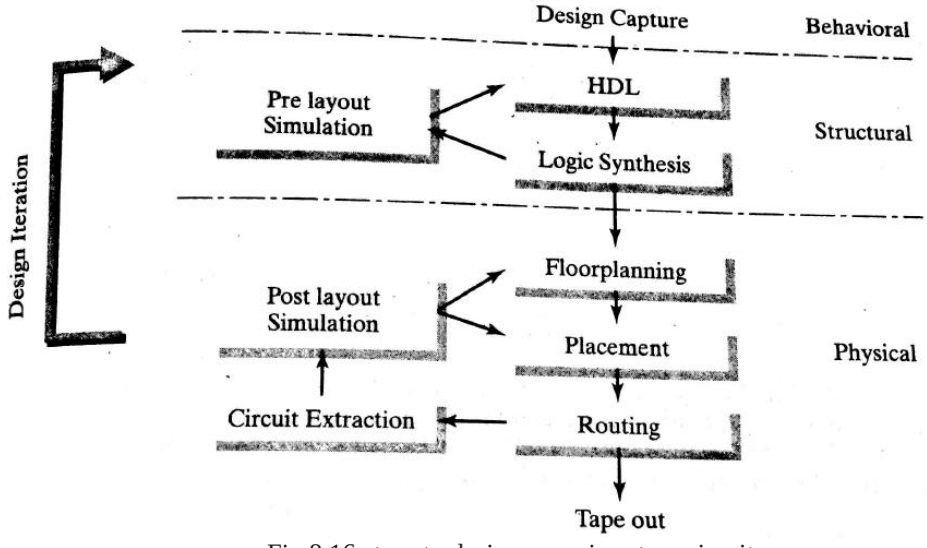


Fig.8.16:steps to design a semi custom circuit

5.3 Cell-Based Design

Cell-based design uses a standard cell library as the basic building blocks of a chip. The cells are placed in appropriate positions, then their interconnections are routed. Cell-based design can deliver smaller, faster, and lower-power chips than FPGAs but has high NRE costs to produce the custom mask set. Therefore, it is only economical for high volume parts or when the performance commands a lucrative sales price. As compared to fullcustom design, cell-based design offers much higher productivity because it uses predesigned cells with layouts. Foundries and library vendors supply cells with a wide range of functionality. These include the following:

- Small-scale integration (SSI) logic (NAND, NOR, XOR, AOI, OAI, inverters, buffers, registers)
- Memories (RAM, ROM, CAM, register files)
- System level modules such as processors, protocol processors, serial interfaces, and bus interfaces
- Possibility of mixed-signal and RF modules

Whereas Medium Scale Integration (MSI) functions such as adders, multipliers, and parity blocks used to be supplied as cells, synthesis engines commonly construct these from base-level Small Scale Integration (SSI) gates in current design systems.

5.3.1.Standared cell Design:

A design is captured using the standard cells available in a library via schematic or HDL .The layout is then normally automatically placed and routed by CAD software. For SSI and MSI blocks, the layout style is usually identifiable as rows of constant or near constant height

blocks separated by rows of routing. As the complete layout is being done, optimization of the height of routing channel may be completed by good placement. Most manufacturers have extended the SSI/MSI standard-cell technique to the design of data paths and other gigher level functions such as microprocessors and their peripherals. Another fundamental component of a standard cell system is a selection of memories. Often these are available as set of parameterizable modules based on word width, number of words ,and number of read and write ports.

Compared to gate arrays,standard-cell designs provide a density advantage at the cost of increased prototype costs and possibly increased design complexity. However where manufacturers have implemented sizable circuit blocks, the productivity of a standard cell approach might in fact be better than that of a gate array because the function does not need to be designed. Fig.shows the example of small standared cell.

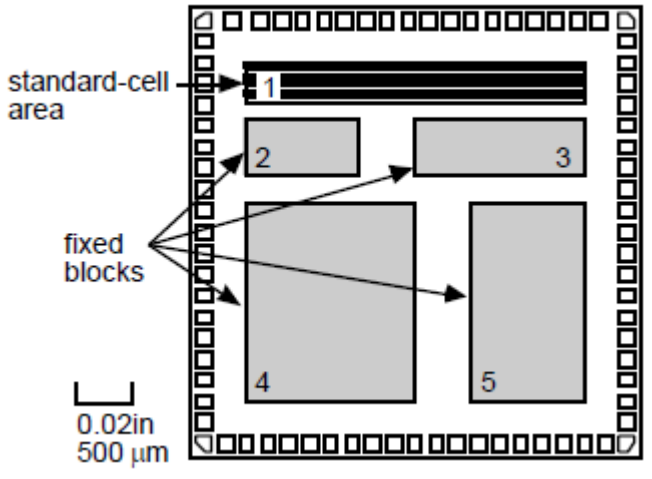


Fig: Standard cell Design

5.3.2.A typical standard cell library:

A typical standard cell library is shown in Table 14.3. A 1x (normal power) cell commonly is defined to use the widest transistors that fit within the vertical pitch of the standard cell. 2x and larger (high power) cells use wider transistors to deliver more current. They must fold the transistors to fit within the cell; this comes at the expense of increased cell width. Gates are often available in low power versions as well. These cells use minimum-width transistors to reduce capacitance. Low-power cells tend to be slow because of the wire capacitance they must drive. Although they do not save area, they do reduce power consumption on noncritical paths. Sophisticated libraries also generate memories of assorted sizes from a graphical user interface. The generators yield not only the physical layout but also a complete data sheet indicating access times, cycle times, and power dissipation.

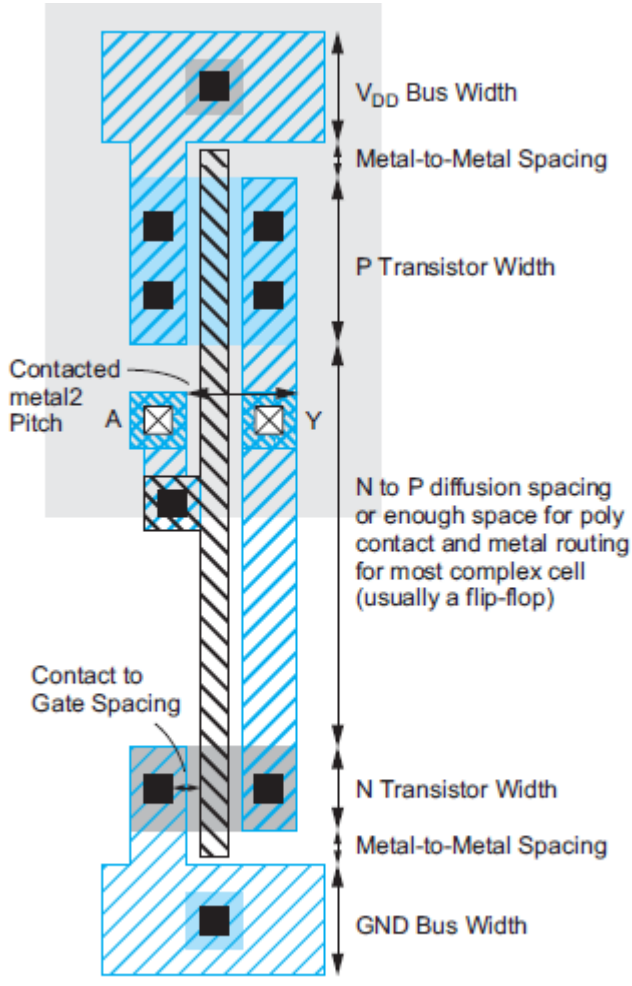


FIGURE 14.18 Typical standard cell layout with some of the constraints

In the event that a standard cell library may not be available for a process, it is worthwhile to review some of the approaches to standard cell design. Usually, standard cells are a fixed height with power and ground routed respectively at the top and bottom of the cells, as shown on the inside front cover. This allows the cells to be abutted end to end and to have the supply rails connect. A single row of nMOS transistors adjacent to GND (ground) and a single row of pMOS transistors adjacent to VDD (power) are normally used. The polysilicon gate is connected from nMOS transistor to pMOS transistor and, in the case of multiplexers and registers, the polysilicon connection has to be crossed between vertically coincident nMOS and pMOS transistors. Decisions about the sizes of transistors have to be made. Following this decision, the cells are almost completely defined by the process design rules. Figure 14.18 illustrates this point. The height of the cell is defined by the sum of the nMOS and pMOS transistor widths, the separation on n and p regions, the spacing to VDD and GND busses, and the width of these busses. The horizontal pitch is defined by the poly-tometal2 contacted pitch, as shown in the figure. It is relatively easy to construct a software program to automatically

generate cells like the one shown in Figure 14.18. Cell delay is characterized through simulation to good agreement with silicon. Fabrication of such cells to prove performance is rarely required. Options to standard cells include routing the clock with the power and ground busses and routing multiple supply voltages to each cell. The latter technique is sometimes used to reduce power by connecting gates that are not in the critical path to a lower than normal supply voltage. Recall that the power drops with the square of the supply voltage.

TABLE 14.3 Typical standard cell library

Gate Type	Variations	Options
Inverter/Buffer/ Tristate Buffers		Wide range of power options, 1x, 2x, 4x, 8x, 16x, 32x, 64x minimum size inverter
NAND/AND	2–8 inputs	High, normal, low power
NOR/OR	2–8 inputs	High, normal, low power
XOR/XNOR		High, normal, low power
AOI/OAI	21, 22	High, normal, low power
Multiplexers	Inverting/noninverting	High, normal, low power
Adder/Half Adder		High, normal, low power
Latches		High, normal, low power
Flip-Flops	D, with and without synch/asynch set and reset, scan	High, normal, low power
I/O Pads	Input, output, tristate, bidirectional, boundary scan, slew rate limited, crystal oscillator	Various drive levels (1–16 mA) and logic levels

5.4.FPGA building block architectures

By the early 1980's **large scale integrated circuits (LSI)** formed the back bone of most of the logic circuits in major systems. Microprocessors, bus/IO controllers, system timers etc were implemented using integrated circuit fabrication technology. Random "glue logic" or interconnects were still required to help connect the large integrated circuits in order to:

1. Generate global control signals (for resets etc.)
2. Data signals from one subsystem to another sub system.

Systems typically consisted of few large scale integrated components and large number of SSI (small scale integrated circuit) and MSI (medium scale integrated circuit) components. Initial attempt to solve this problem led to development of **Custom ICs** which were to replace the large amount of interconnect. This reduced system complexity and manufacturing cost, and improved performance. However, custom ICs have their own disadvantages. They are relatively very expensive to develop, and delay introduced for product to market (time to market) because of increased design time. There are two kinds of costs involved in development of custom ICs

1. Cost of development and design

2. Cost of manufacture

Therefore the custom IC approach was only viable for products with very high volume, and which were not time to market sensitive. FPGAs were introduced as an alternative to custom ICs for implementing entire system on one chip and to provide flexibility of reporogramability to the user. Introduction of FPGAs resulted in improvement of density relative to discrete SSI/MSI components (within around 10x of custom ICs). Another advantage

of FPGAs over Custom ICs is that with the help of computer aided design (CAD) tools circuits could be implemented in a short amount of time (no physical layout process, no mask making, no IC manufacturing)

An **FPGA** is a device that contains a matrix of reconfigurable gate array logic circuitry. When a FPGA is configured, the internal circuitry is connected in a way that creates a hardware implementation of the software application. Unlike processors, FPGAs use dedicated hardware for processing logic and do not have an operating system. FPGAs are truly parallel in nature so different processing operations do not have to compete for the same resources. As a result, the performance of one part of the application is not affected when additional processing is added. Also, multiple control loops can run on a single FPGA device at different rates. FPGA-based control systems can enforce critical interlock logic and can be designed to prevent I/O forcing by an operator. However, unlike hard-wired printed circuit board (PCB) designs which have fixed hardware resources, FPGA-based systems can literally rewire their internal circuitry to allow reconfiguration after the control system is deployed to the field. FPGA devices deliver the performance and reliability of dedicated hardware circuitry.

A single FPGA can replace thousands of discrete components by incorporating millions of logic gates in a single integrated circuit (IC) chip. The internal resources of an FPGA chip consist of a matrix of configurable logic blocks (CLBs) surrounded by a periphery of I/O blocks shown in Fig. 20.1. Signals are routed within the FPGA matrix by programmable interconnect switches and wire routes.

A further class of programmable device is the programmable (or reprogrammable) gate array. These may be further categorized into ad-hoc and structured array.

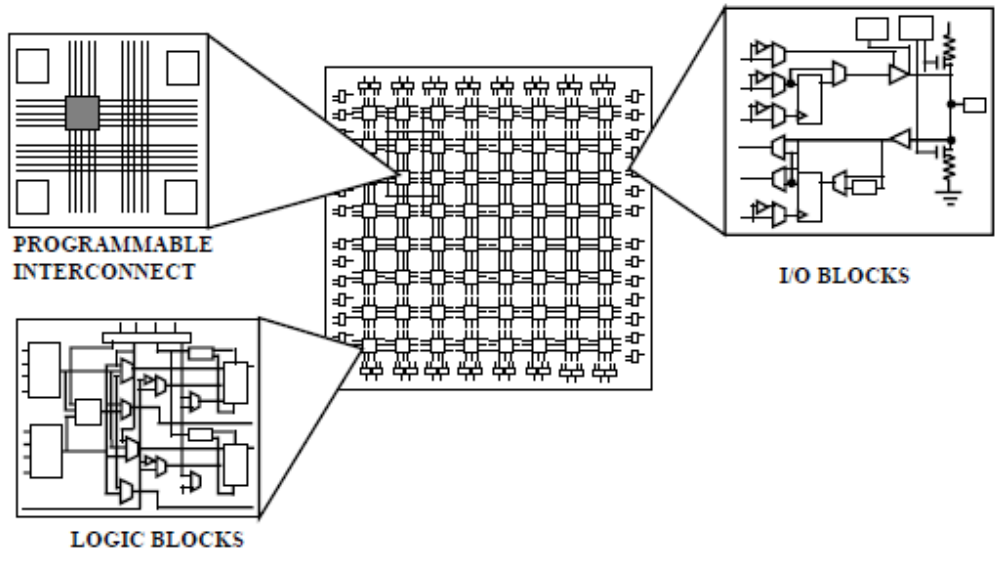


Fig. 20.1 Internal Structure of FPGA

In an FPGA logic blocks are implemented using multiple level low fan-in gates, which gives it a more compact design compared to an implementation with two-level AND-OR logic. FPGA provides its user a way to configure:

1. The intersection between the logic blocks and
2. The function of each logic block.

Logic block of an FPGA can be configured in such a way that it can provide functionality as simple as that of transistor or as complex as that of a microprocessor. It can used to implement different combinations of combinational and sequential logic functions. Logic blocks of an FPGA can be implemented by any of the following:

1. Transistor pairs
2. combinational gates like basic NAND gates or XOR gates
3. n-input Lookup tables
4. Multiplexers
5. Wide fan-in And-OR structure.

Routing in FPGAs consists of wire segments of varying lengths which can be interconnected via electrically programmable switches. Density of logic block used in an FPGA depends on length and number of wire segments used for routing. Number of segments used for interconnection typically is a tradeoff between density of logic blocks used and amount of area used up for routing. Simplified version of FPGA internal architecture with routing is shown in Fig. 20.2.

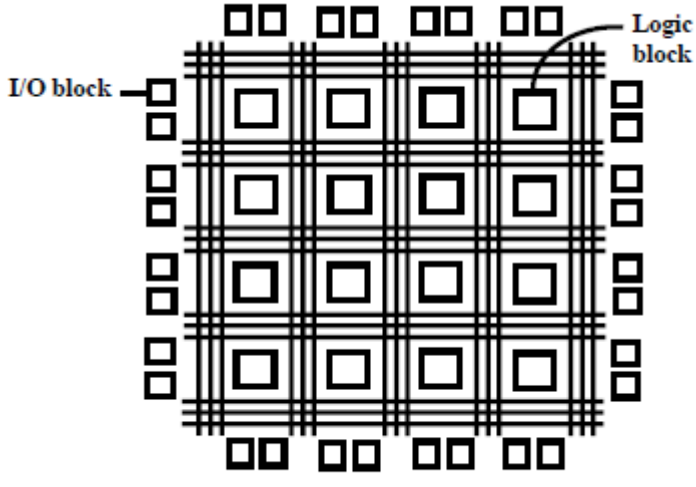


Fig. 20.2 Simplified Internal Structure of FPGA

5.4.1.The XILINX Programmable Gate Array:

An array of configurable logic block (CLBs) is embedded within a set of horizontal and vertical channels that contain routing that can be personalized to interconnect CLBs. The configuration of the interconnect is achieved by turning on n-channel pass transistors. The state that determines a given interconnect pattern is held in static RAM cells distributed across the chip close to the controlled elements. The CLBs and routing channels are surrounded by a set of programmable I/Os.

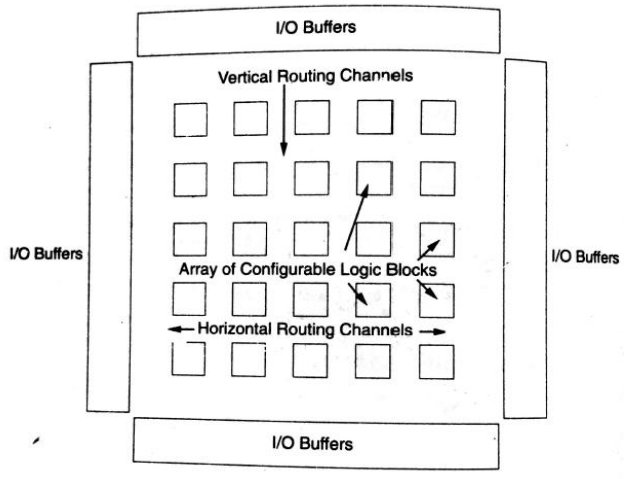


Fig.6.16:XILINX FPGA architecture

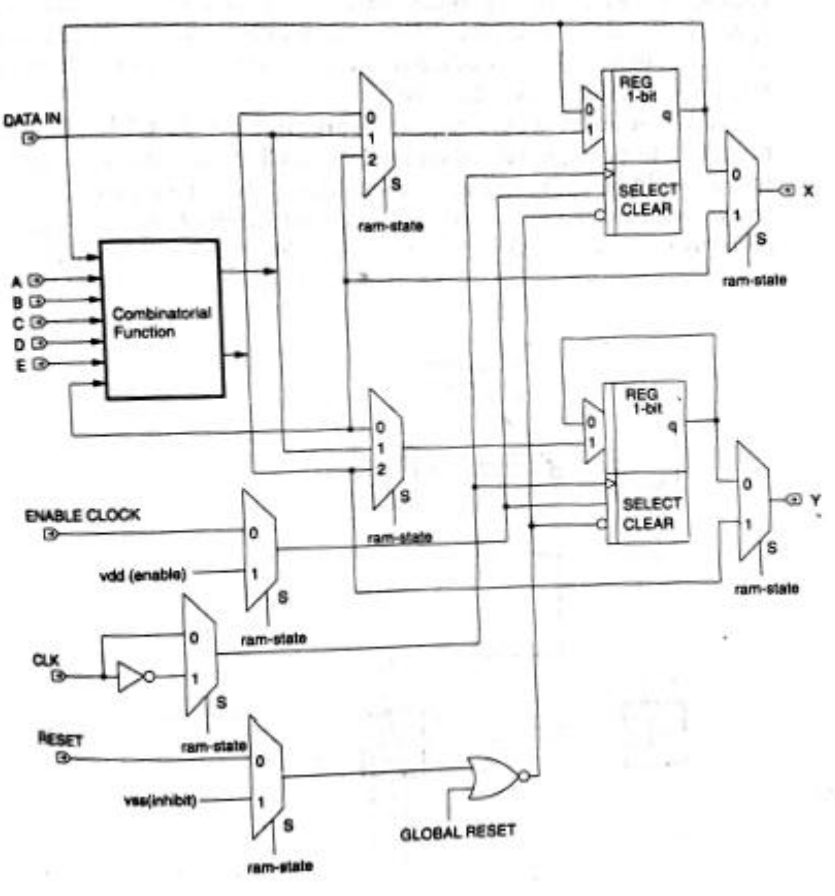
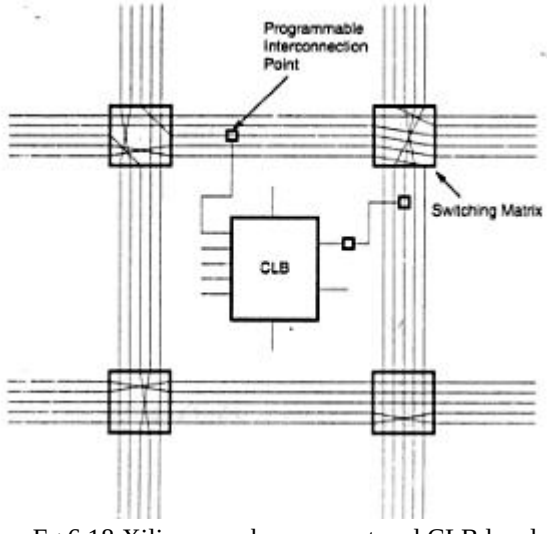


Fig.6.17:Xilinx configurable logic block

In detail the structure of a CLB is shown in fig.6.17.It consists of two registers, A number of muxes, and a combinational function unit. The latter can generate two functions of four variables, any function of five variables, or a selection between two functions of four variables. The function bit and each mux is controlled by a number of RAM state bits.

Fig 6.18 shows a typical CLB surrounded by switching matrices. The switching matrices perform crossbar switching of the global interconnect, Which runs both vertically and horizontally. programmable interconnect points or PIPs interconnect the global routing to CLBs. Both PIPs and the switching matrices are implemented as n-channel pass gates controlled by 1-bit RAM cells. Extra special long distance interconnect is used to route important timing signals with low skew.



Fg.6.18:Xilinx crossbar connect and CLB localmconnect

5.4.2.Algotronix:

This architecture contains 1024 identical logic cells arranged in a 32-by32 matrix. At the boundary of the chip.128 programmable I/O pins allow cascading the chips in even larger arrays. The cell interconnect is shown in fig.6.19Each cell is connected to the East, South, West, North neighbor. In addition two global-interconnect signals connect to each cell. These are used to supply a low-skew signal to all cells for clocking. Each cell also receives row select lines and bit lines that are used to program RAM bits within the logic cells that dynamically customize the logic cell.

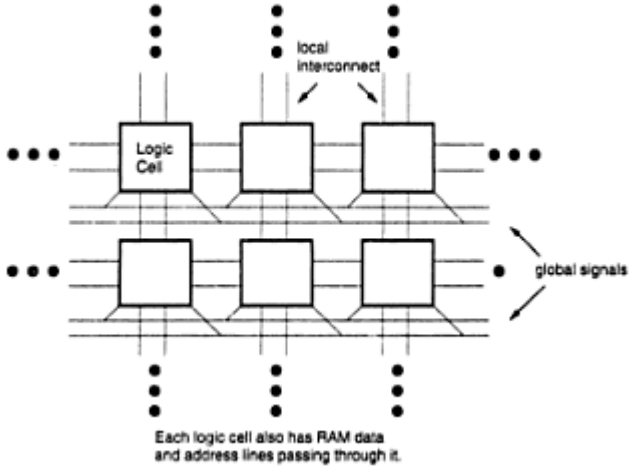


Fig.6.19Algotronix FPGA chip architecture

The cell design is shown in fig.6.20.It consists of four "through" multiplexers to route single-bit signals entering from the north, South, East, West. In addition two multiplexers route a selection of signals to a function unit. These signals include the signals entering on the orthogonal edges of the cell ,two global "clock" signals and the function block for feedback situations(latches).The muxes are controlled by small 5 transistor static RAM cells.

The I/O pads are very interesting. The trick is to use only one pin for I/O into and out of the array but have the communicating chips automatically deal with the pins that are outputs. The pads achieve this by using a ternary logic scheme to since when two outputs are driving each other via a contention circuit. This is used with an XOR gate, as shown in fig.6.21,to deduce the correct input value.

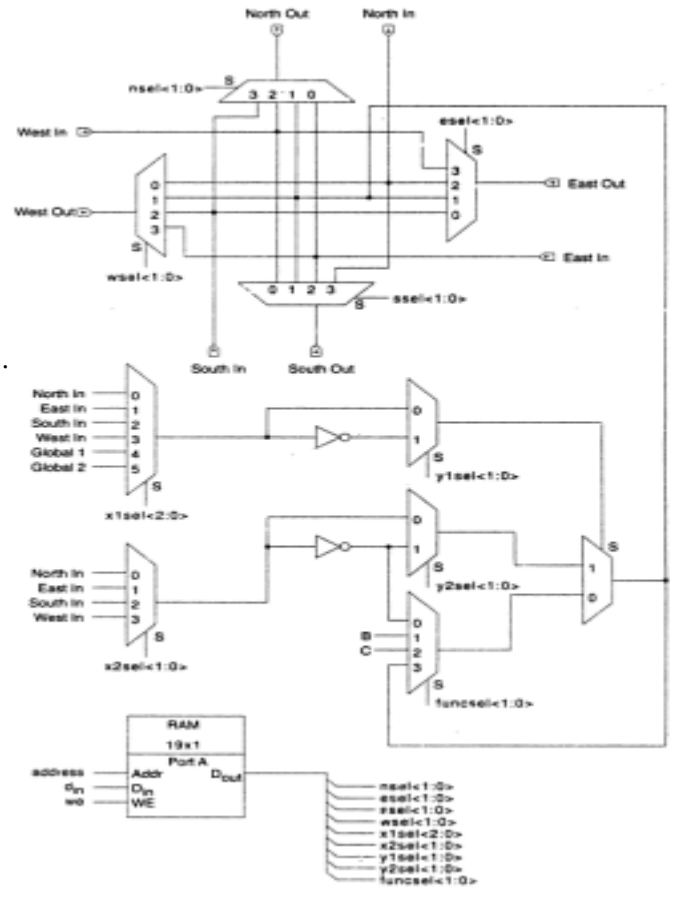


Fig.6.20:Algotronix cell design

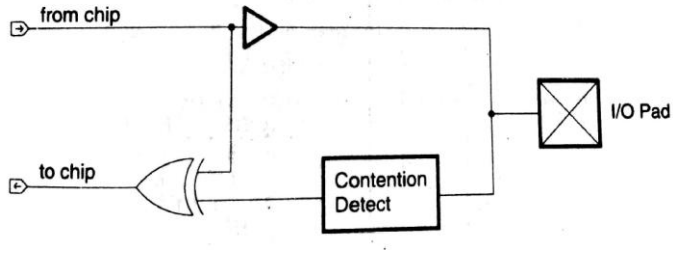


Fig.6.21:Algotronix I/O circuit

5.4.3.Concurrent Logic:

The CLi6000 series is another example of a regular array style FPGA. Current design have between 1000 and 3136 cells, with prospects of up to 10000 cells per chip in the next new years. As an example the CLi6005 consists of a 7-by-7 array of superblocks. Each superblock

has an array of 8-by-8 logic cells. Each logic cell connects to the four nearest neighbors and to a local and express bus (fig:6.22).The cell structure is shown in fig.6.23 compared with the algotronix cell it has considerably more functionality within a cell. A resettable register ,XOR, and an AND gate are included. Thus, for instance ,a single counter bit can be implemented in a single cell.

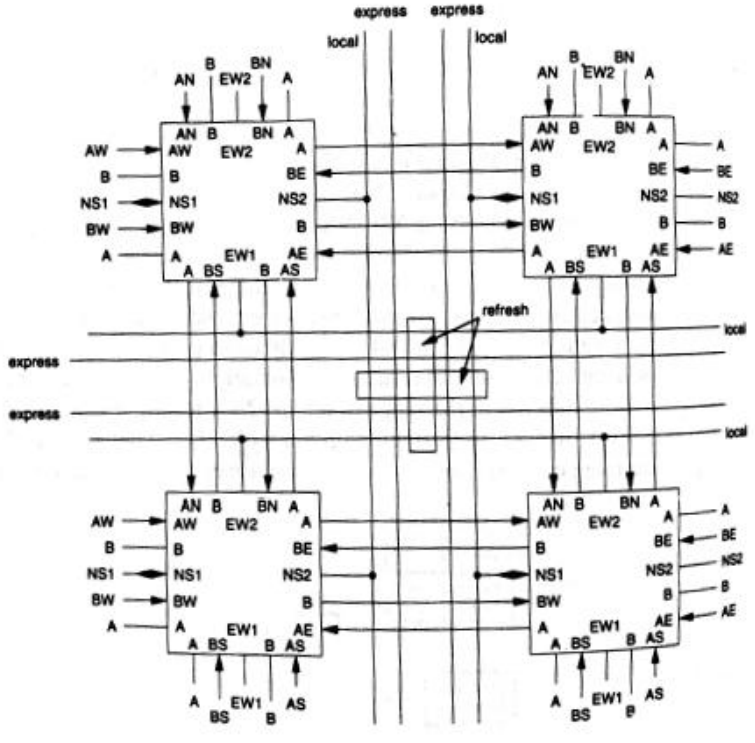


Fig.6.22.Concurrent logic array details

5.5.FPGA interconnect routing procedures:

FPGA has different types of programmable interconnect. The structure and complexity of interconnect is determined by programming technology and architecture of basic logic cell. The raw material used to build interconnect is aluminium-based metallization .Programmable ASIC comes with two layers three layers or more layers of matal interconnect.

5.5.1.Actel ACT:

Fig 5.28 shows the interconnect architecture of actel ACT family and is similar to a channeled gate array. The channel routing uses dedicated rectangular areas of fixed size within the chip called wiring channels. The horizontal channels run across the chip in the horizontal direction. In vertical direction, vertical channels run over the top of the basic logic cells or logic

modules. Within the horizontal or vertical channels wires run horizontally or vertically ,respectively ,within tracks. Each track holds one wire. Capacity of fixed wire channel is equal to the number of tracks it contains.

In a channeled gate array the designer decides the location and length of the interconnect within a channel. In a FPGA the interconnect is fixed at th time of manufacture. To provide interconnect programming, Actel divides the fixed interconnect wires within each channel into various lengths of wire segments. The segmented channel routing is called as a variation on channel routing. Antifuses join the wire segments. The designer then programs the interconnections by antifuses and making connections between wire segments. The unwanted connections are left unprogrammed.

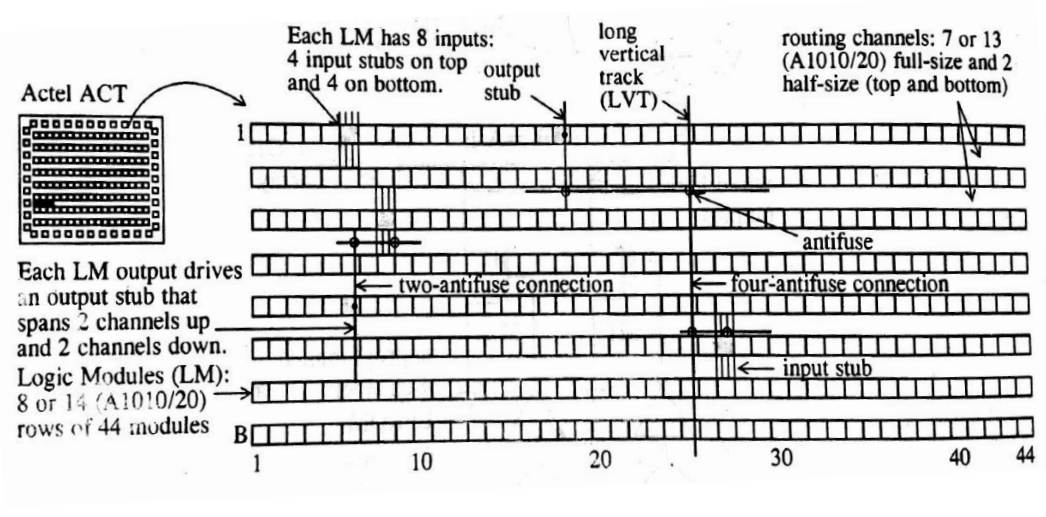


Fig 5.28 the interconnect architecture of actel

5.5.1.1.ACT 1 interconnect:

ACT 1 routing resources interconnection architecture uses 22 horizontal tracks per channel for signal routing with three tracks dedicated to VDD, GND ,and the global clock. This makes a total of 25 tracks per channel. Horizontal segments vary in length from four columns of logic modules to the entire row of modules called long lines.

Four logic module inputs are available to the channel below the logic module and four inputs to the channel above the logic module. Eight vertical tracks per logic module are available for inputs. This is the input stub. Single logic module and across the two channels below the module. This is the output stub. Output uses four vertical tracks per module. One vertical track per column is a long vertical track that spans the entire height of the chip. Fig 5.29 shows the ACT 1horizontal and vertical channel architecture.

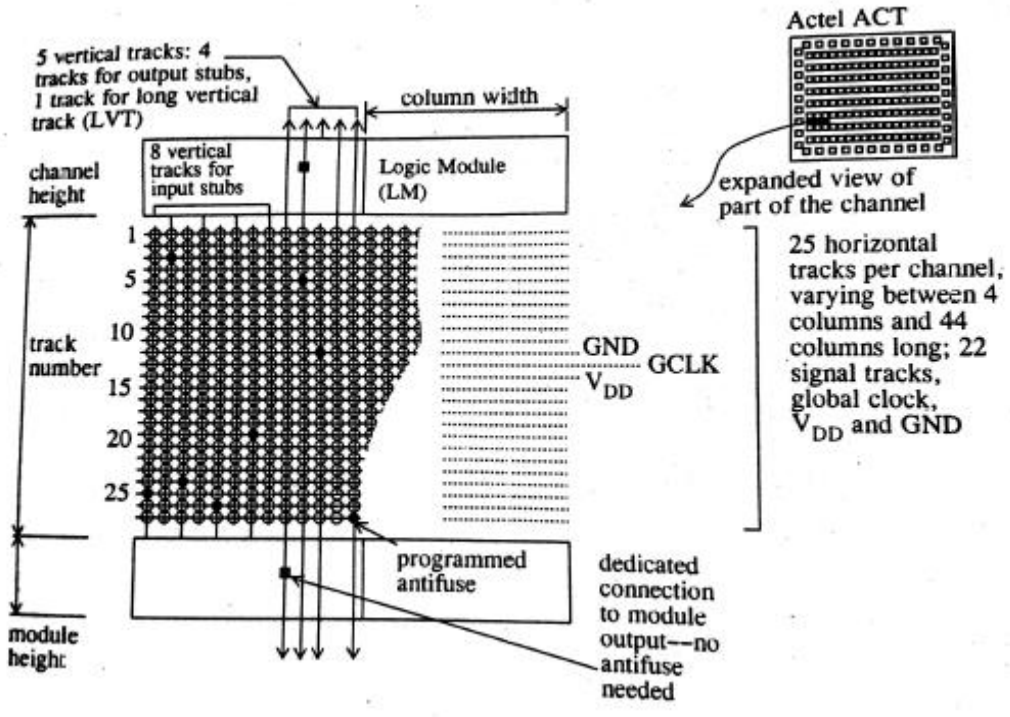


Fig 5.29 ACT 1 horizontal and vertical channel architecture

5.5.1.2 ACT 2 and ACT 3 interconnect:

ACT 1 architecture uses two antifuses for routing nearby modules, three antifuse to join horizontal segments, and four antifuses to use a horizontal or vertical long track. The ACT 2 and ACT 3 architectures use increased interconnect resources. This reduces the number of connections that need more than two antifuses. Delay is also reduced by decreasing the population of antifuses in the channels and by decreasing the antifuses resistance of certain critical antifuses. Channel density is the absolute minimum number of tracks needed in a channel to make a given set of connections. Software to route connections using channeled routing is so efficient.

ACT 2 devices have 36 horizontal tracks per channel rather than the 22 horizontal tracks available in the ACT 1 architecture. Horizontal track segments in an ACT 3 devices range

From a module pair to the full channel length. Vertical tracks are ,input with a two channel span for one up and one down, outrput with a four channel span with two up and two down, and long(LVT).Four LVTs arte shared by each column pair. The ACT 2/3 logic modules can accept five inputs ,rather than four inputs for the ACT 1 modules. Thus ACT 2/3 logic

modules need an extra two vertical tracks per channel. The number of tracks per column increases from 13 to 15 in the ACT 2/3 architecture.

The greatest challenge facing the Actel FPGA architects is the resistance of the polysilicon diffusion antifuses. The nominal antifuse resistance in the ACT 1-2 1-2µm process is approximately 500Ω t0 700Ω.The high resistance severly limits the number of antifuses in a connection. ACT 2/3 assign a special antifuse to each output making a direct connection to an LVT. This reduces the number of antifuse to three .This antifuse is blown at higher current. The antifuse resistance in ACT 3 using 0.8 µm process is 200 µm.

5.5.2 Xilinx LCA

Figure 5.30 shows the hierarchical Xilinx LCA interconnect architecture and has the following features.

1.Vertical lines and horizontal lines run between CLBs.
2.General purpose interconnects joins switch boxes or magic boxes or switching matrices.
3.Long lines run across the entire chip to form internal buses and the three state buffers are next to each CLB
4.Direct connections bypass the switch matrices and directly connect adjacent CLBs.
5.Programmable interconnection points(PIPs) are programmable pass transistors that connect the CLS inputs and outputs to the routing network
6.Bidirectional (BIDI) interconnects buffers restore the logic level and logic strength on long interconnects paths.

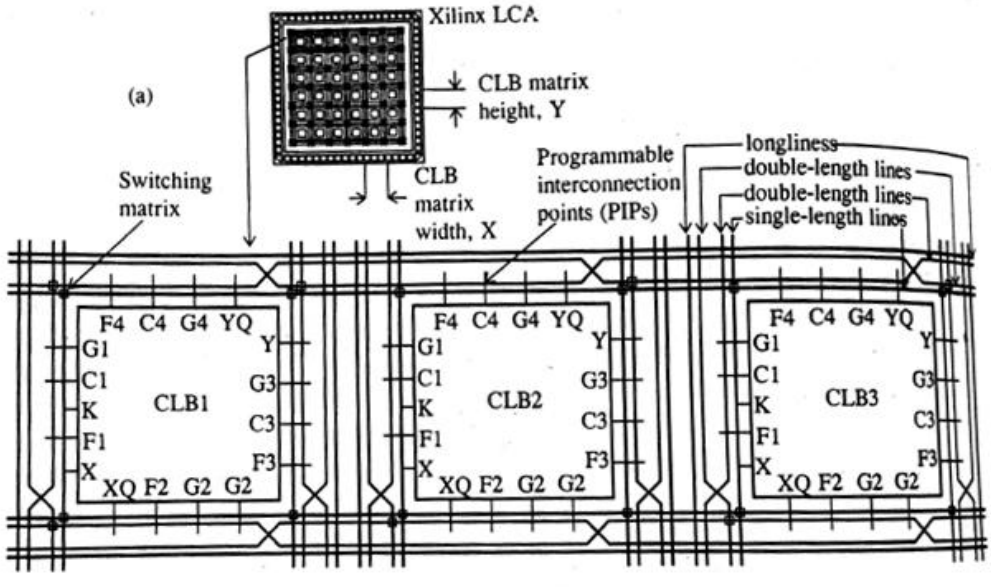


Figure 5.30 hierarchical Xilinx LCA interconnect architecture

5.5.3 Xilinx EPLD:

Xilinx EPLD family uses an interconnect bus known as universal inter connection module(UIM) to distribute signals within the FPGA. Figure 5.31 shows the Xilinx EPLD which uses a programmable AND array with constant delay from any input to any output. This has the following

1.CG is the fixed gate capacitance of the EPROM device
2.CD is the fixed drain parasitic capacitance of the EPROM device
3.CB is the variable horizontal bus(bit line)capacitance
4.CS is the variable bus(word line) capacitance

Figure 5.31 shows the UIM with 21 output connections to each FB.The UIM has 4*2 arrays of eight FBs and has 168 output connections.Most of the nine I/O cells attached to each FB have two input connections to the UIM,one from a chip input and one feedback from the macrocell output.

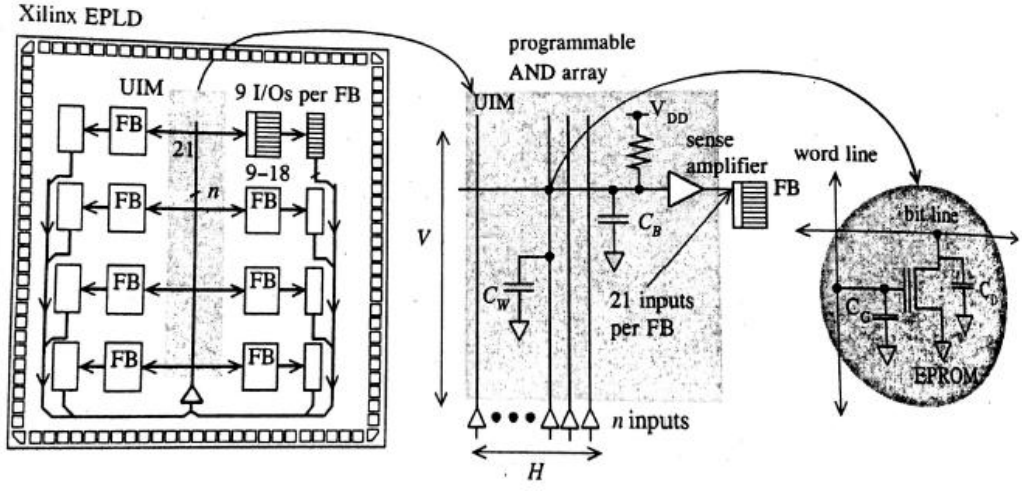


Figure 5.31 Xilinx EPLD

5.5.4.Altra max 5000 and 7000

Altra MAX 5000 and 7000 devices use a programmable interconnect array(PIA) as shown in figure 5.32.PIA is a cross point switch for logic signals traveling between LABs.

Advantages:
1.Uses a fixed number of connections
2.Fixed routing delay
3.Simple and improved speed in placement and routing software.

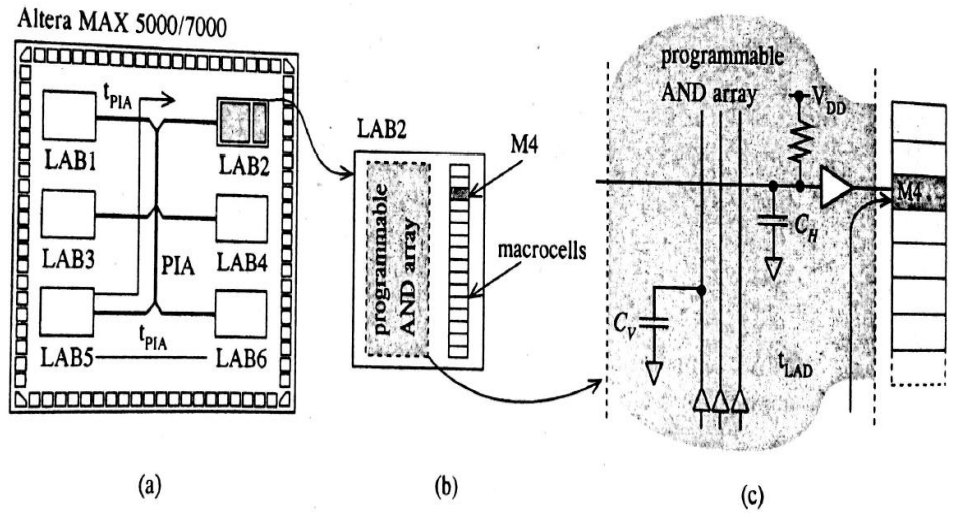


Fig.5.32 Altera Max interconnect

5.5.5.Altera Max 9000

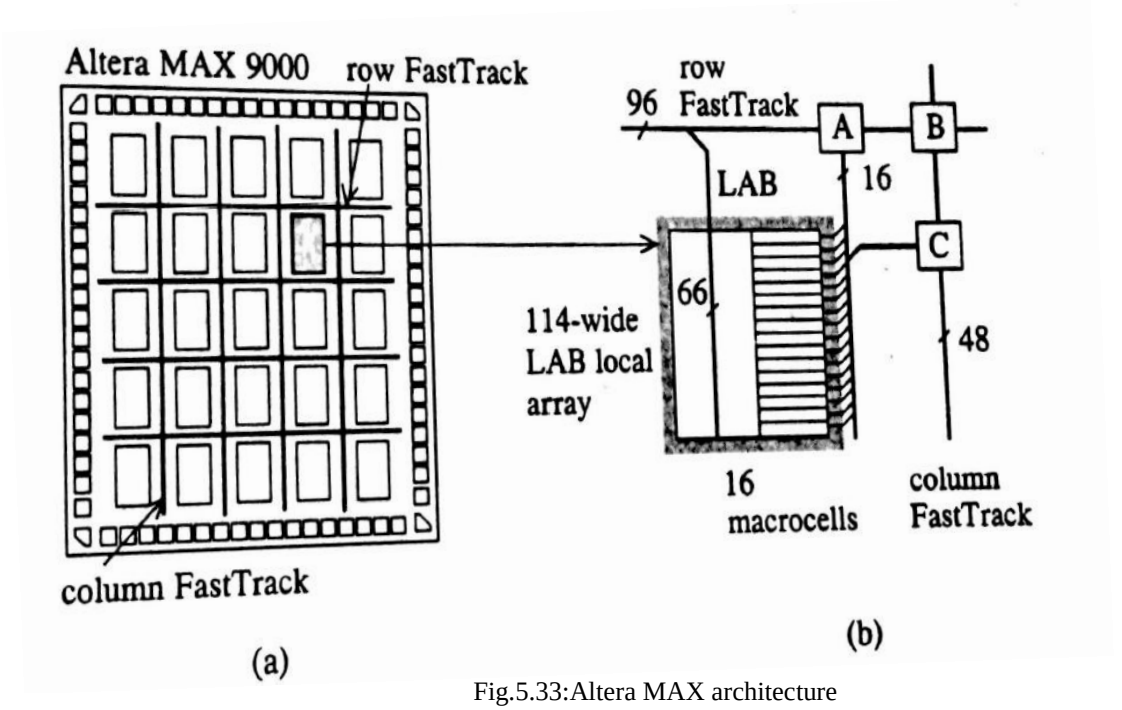


Fig.5.33:Altera MAX architecture

Figure 5.33 shows the altera MAX 9000 interconnect architecture. The size of the MAX 9000 LAB arrays varies between 4*5 (rows * columns) for the EPM9320 and 7*5 for the EPM9560.MAX 9000 is an extremely coarse grained architecture. This uses complex PLDs.

The boxes A,B and C represent the interconnection between the fast track buses and the 16 macrocells in each LAB.

1.Box A connects a macrocell to one row channel
2.Box B connects three column channels to two row channels
3.Box C connects a macrocell to three column channels

5.5.6.Altera FLEX:

Figure 5.34 shows the interconnect used in the Altera FLEX family of complex PLDs. Altera refers to the FLEX interconnect and MAX 9000 interconnect by the same name, FastTrack. These two are different because the granularity of the logic cell arrays in different. The FLEX architecture is of finer grain than the MAX arrays because of the difference in programming technology. The FLEX horizontal interconnect is much denser than the vertical interconnect. FLEX has an aspect ratio for interconnect of over 10.1.This imbalance is partly due to the aspect ratio of the die, the array and the aspect ratio of the basic logic cell,the LAB.

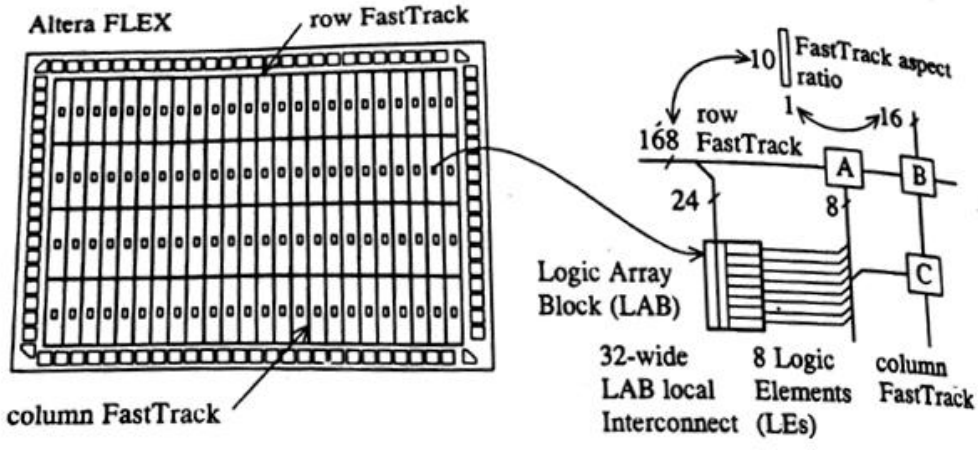


Fig.5.34:Altera FLEX interconnect