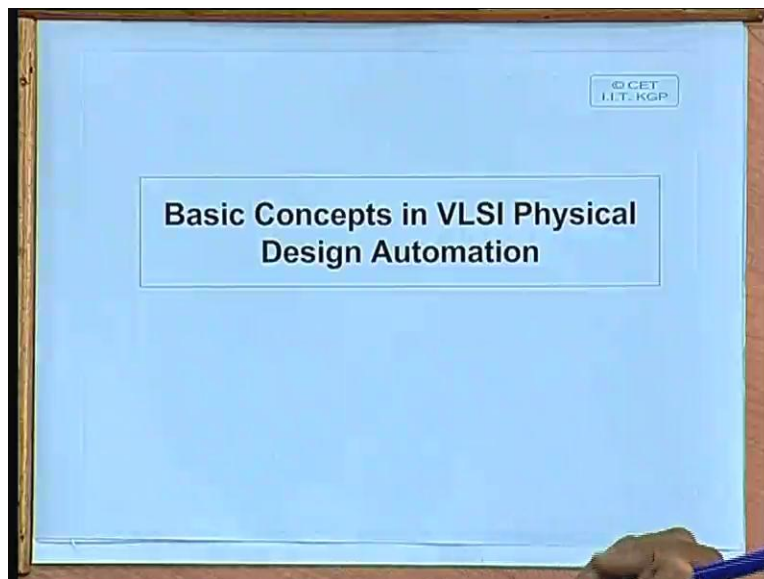


**Electronic Design Automation**  
**Prof. Indranil Sengupta**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture No #15**  
**Synthesis: Part I**

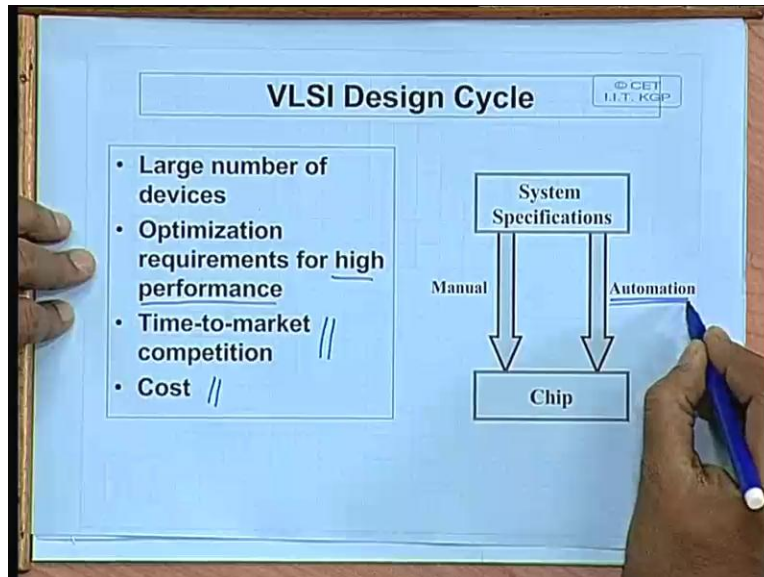
So from this class we would be starting our discussion on the back end design of VLSI systems. Means, in the asic design flow if you recall we had mention earlier that there a number of different steps we have to go through. Some of them of them you can categorize as the front end design some of them you can categorize as back end design just to recall. We would very fast go through the different steps and we will also have a very quick review of the VLSI design styles in order that you can appreciate or understand some of the algorithms that we would be discussing subsequently under this okay. So our broad topic of discussion today is um some basic concepts in VLSI physical design automation fine.

(Refer Slide Time: 01:57)



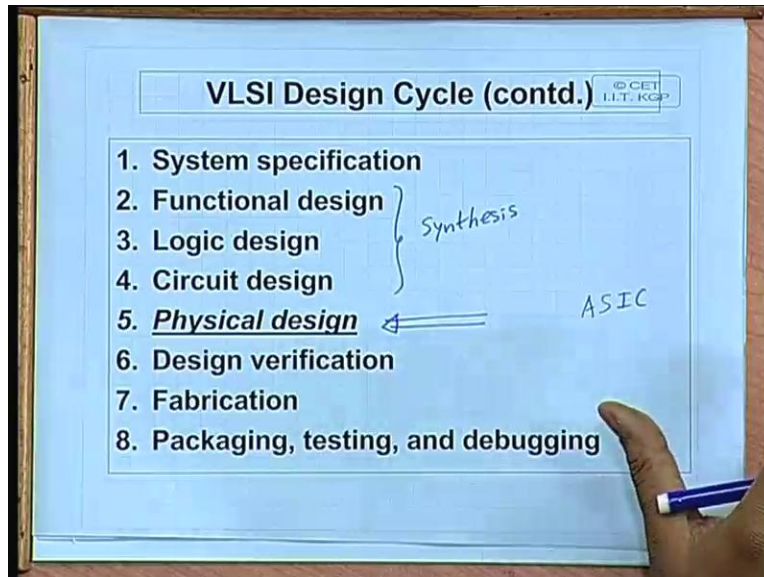
So we start by again looking at the VLSI design cycle.

(Refer Slide Time: 02:04)



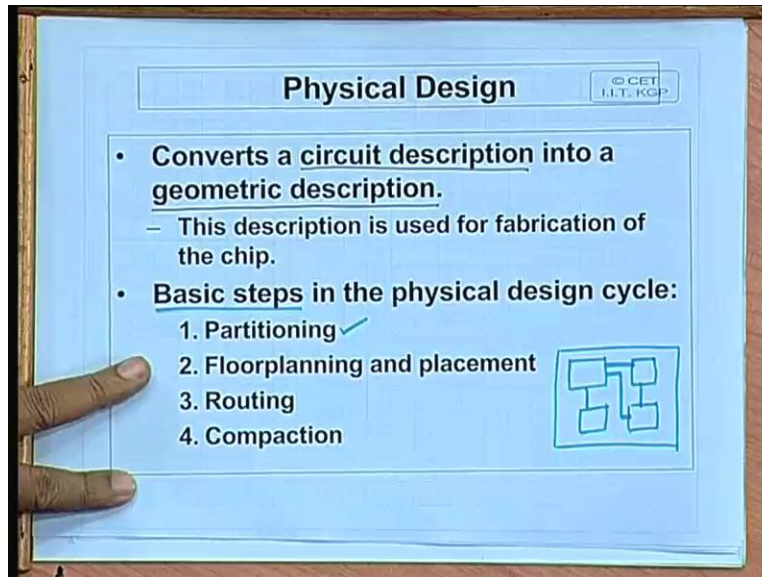
Now just to recall in a in a typical VLSI design today we have a very large number of devices. And and obviously to obtain reasonable amount of performance we need optimization at every phases of the design. Moreover we need to do it fast because there is computation from vendors there is time to market some kind of a constraint and cost of design is of course an issue. See earlier starting from our system specification um we could also carry out a manual or a semi-automated process in order to arrive at the chip. But today with the kind of chip sizes we talk about complete automation tools are mandatory. So just we have to rely very heavily on automation tools in order to start from the system specification down to the chip level. Now in the VLSI design cycle I had already mentioned earlier the very beginning.

(Refer Slide Time: 03:02)



That there are a number of different steps we need to go through starting from the system specification functional design, logic design, circuit design, they fall under the purview of the synthesis. Then you will have to carry out physical design to arrive at the layout possibly if the target is an ASIC design verification fabrication and finally packaging and testing. So there are a number of steps in the VLSI design cycle. So now we would be concentrating on physical design which follows the steps of synthesis. After synthesis is complete and we have obtained a circuit description possibly in the form of a netlist. So starting from there we will have to move through several steps in order to get a much lower level description of the design which I had said if our target is an application specific IC that can be a layout okay. So these are the different steps which have to be gone through and we are most interested right now in the step of physical design.

(Refer Slide Time: 04:19)



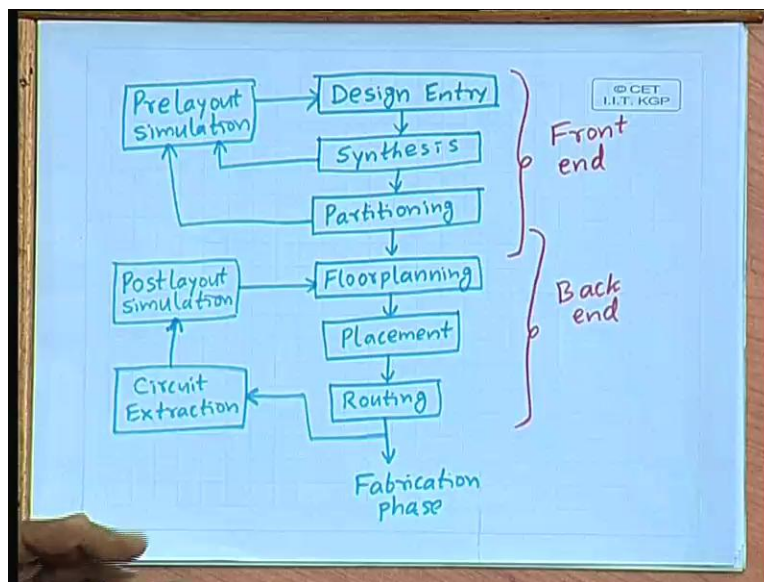
So broadly in the physical design phase we have a circuit description as the input this can be a netlist. And from there we typically go towards a layout description and this step of transformation is typically carried out through an intermediate description which is geometric in nature. Geometric in nature why because now we think about the silicon floor we think about the blocks or the components we need to put in the silicon floor. We need to think about the exact geometry of the interconnecting paths and so on so at this level we talk about rectangles and straight lines right. So ultimately our objective is to put them on the available area in a suitable way so as to optimize the performance based on several criteria we would be seeing. Now the different steps or the main step these are the basic steps involved in this process are ofcourse if it is a big design you have to do some partitioning floor planning.

And placement just like when you are building a house you have to make a plan at the beginning. First you plan roughly where you will put what then you refine your specification with respect to the netlist and you can carry out the actual placement then we have to interconnect them routing. And finally you will find that after everything is over still you can do some kind of compaction to reduce the total area. So this compaction is a post processing step you can say so after everything is done you typically do compaction. Now I am sure yes. [Student Noise Time:

06:23] Routing after floor planning yes [Student Noise Time: 06:27]. Okay we will see later that that means when you are doing floor planning or placement we have to have a very rough or gross estimate regarding the area we have to leave aside for interconnection.

But that is an estimate unless we carry out the actual interconnection we cannot know the exact amount of that will be actually required. [Student Noise Time: 06:49] Yes [Student Noise Time: 06:51] No there is no alternative loop as such because floor planning and placement are some kind of a rough placement route um after routing the geometry will be fixed. Placement says that I have to put this block to the right of this block this block to the bottom of this block there will be some space in between for routing. But when this routing step is carried out you will find out exactly how many tracks of wires you have to leave aside between two blocks so the exact geometry will be finalized only after routing. This floor planning are placement at tentative okay so after routing it gets finalized fine okay.

(Refer Slide Time: 07:39)



So I am showing you the over steps once more. So the first step is design entry design entry is typically through a hardware description language you know you specify the behavior. Next step is of course synthesis we have talked about several aspects of synthesis already this may involve

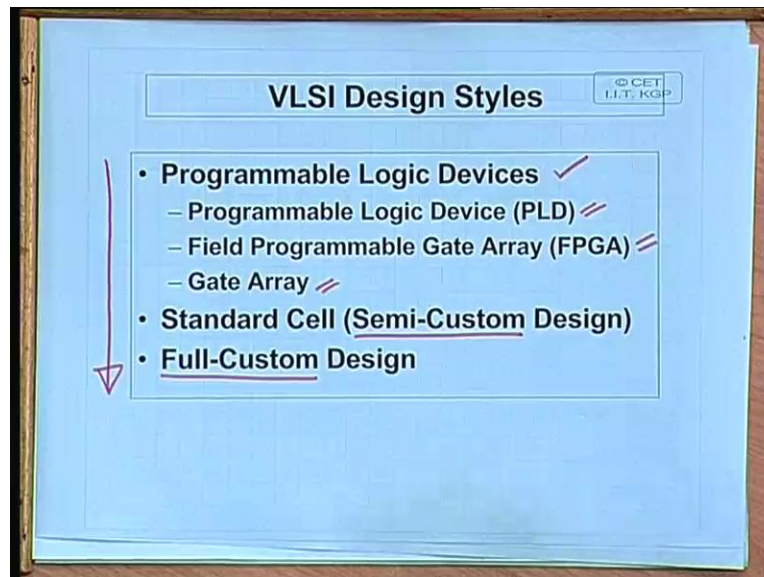
high level synthesis logic synthesis everything. So typically during or after synthesis you go for partitioning so usually we partition either the c d f g or the netlist depending on how we are using it. So after floor planning um after partitioning comes floor planning. Floor planning is the relative ordering of the different sub blocks then with some additional information and refinements we get a more accurate picture which is placement.

And after placement we carry out routing when the geometry becomes finalized, well in this flow I am not showing compaction but compaction is implied. So from the output of routing you can move on to the fabrication phase. So I am not going into detail of this because we are not that much interested in the fabrication phase. So after the routing phase the layout is ready so there are some feedback paths also like there is a step called pre layout simulation. This is typically used to verify the higher level design. So either from the output of the partitioning phase or from the output of the synthesis phase you go back do a simulation and from there you may refine your design and again come back. Similarly there is a step called post layout simulation.

So post layout simulation typically this is done after a process called circuit extraction is carried out. So what this circuit extraction step does it takes the output of the routing phase which is a layout and from the layout it tries to extract back the circuit and after extracting back the circuit it gives it to a simulator. Now this simulator is more accurate so here the basic components are transistors resistors capacitors that way so this can carry out simulation in a much more accurate way. And if it does not satisfies the timing or the power specification you may go back to floor planning and make some changes there or you may even go back to placement and routing phase also but I am shown it going back to one phase only and in this overall phase um we also mentioned in the beginning.

So of two partitioning you can say this broadly is called the front end cad design tools. And there is a small overlap here and the remaining part these is in this are called the back end design tools. Now what I have um had mentioned in order to understand the back end steps some of them atleast it is important to have some ideas regarding the target what you want to fabricate. There are some VLSI design styles that people follow and depending on the style the algorithm may be working differently. So we first look at the typical VLSI design styles which are used in practice.

(Refer Slide Time: 12:18)



Broadly speaking there are three different styles one is prog using programmable devices other is called semi custom design and the third one is called full custom design. Now in terms of the complexity. Complexity goes up in this direction in full custom design means say you are trying to design almost everything from scratch. Semi-custom design means you are trying to use or utilize some kind of standard cells which are available in library and you pick and select them and put on the floor of silicon and interconnect them. But in contrast programmable logic devices are something which are already pre fabricated and are available in the market. So you get them and you try to map your design on to them they are programmable in that sense that means you can configure them in a way so that any designs can mapped on to them. So there are user programmable or user configurable devices like pld's or fpga's. Here is also something called gate arrays which also can be configure. But you will have to go back to the manufacturer for that users cannot do that right.

So this is roughly the spectrum the different kinds of design stats people will use them you will see that depending on which style we use the requirement of placement and routing everything will change. [Student Noise Time: 13:45] PLD and FPGA these are some devices which you can buy from the market and using a programmer in a laboratory you can map a design on to it so it

can be done by yourself. But in contrast for gate array you will have to go back to the chip manufacturer tell them your specification they will give back give you back the finished product. But that is not as complex as compared to standard cell or full custom designs this we will see means what are the differences. So we start with the programmable devices for which this fpga's are the most common today. Okay, so we particularly talk of about fpga. I mean one particular family of FPGA we will have a brief look at.

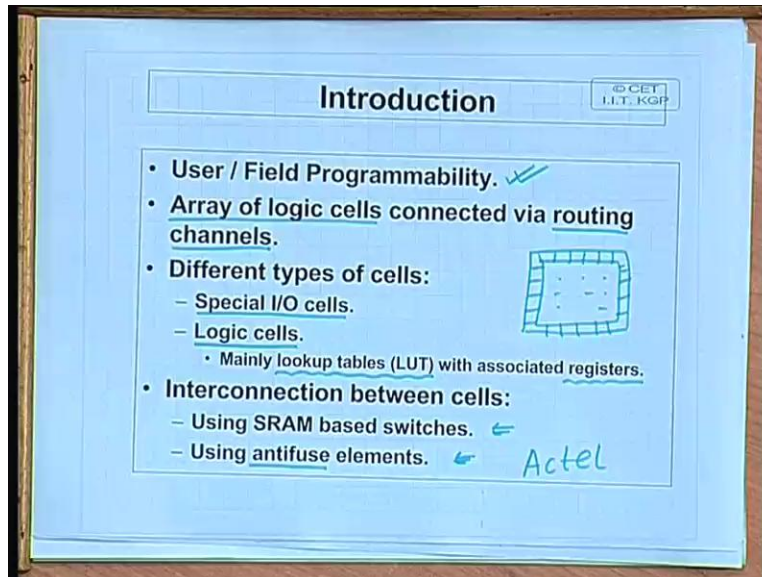
(Refer Slide Time: 14:35)



Now as the name implies field programmable gate array means that you can program it on field that means your lab.



(Refer Slide Time: 14:45)



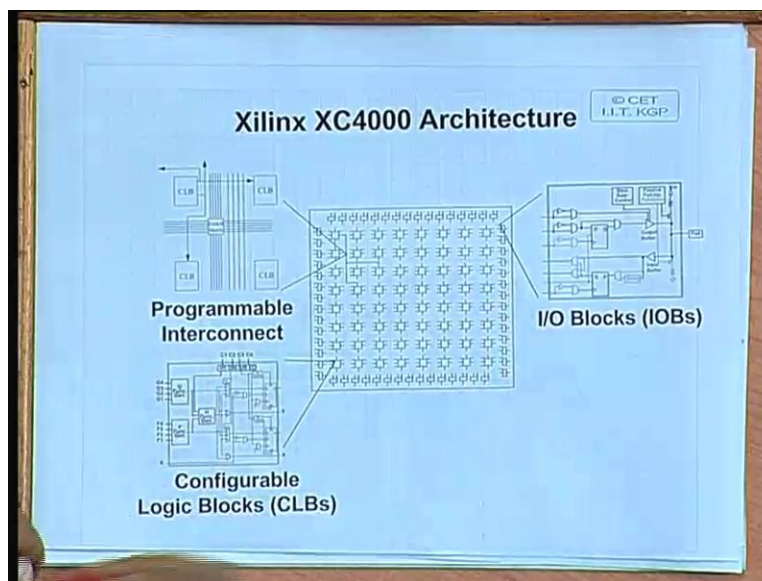
So this is the biggest advantage of fpga's that it supports user or field programmability. Roughly speaking an FPGA consist of an array of logic cells. Now the cells can be quite complex it may not be as simple as a gate. It can be a fairly complex function block it can also contain flip flops they are typically connected through routing channels and the functionality of this logic cells and this routing channels both are programmable. Since you can program both any netlist you have you can map it on to on to them talking about the logic cells. Now inside a FPGA chip there are broadly two kinds of cells. There are special input output cells which typically reside on the periphery of the chip they are special logic cells on the periphery which typically control the input output points the testability clocks etcetera these kinds of things are there.

And of course they are the regular logic cells which are all inside logic cells can be implemented using many ways. But most of um the modern fpga's today they use some sort of a look up table based designs. We will see what this means and there can also be a limited number of flip flops inside the cells and this cells can be interconnected. Well actually here again two different technologies are used one is one uses switches which are controlled by data stored in static RAMs. Now externally if you load the rams the interconnection can change. So this kind of a solution can be reused it is not a one time use you can again change the content of the ram and

interconnection changes. But in contrast there are some chips which can be programmed only once it is based on something called anti fuse technology.

See normally what is a fuse? Whenever you pass a high current there is a disconnection. But anti fuse is different normally there is a disconnection. If you try to pass or means if you apply a high potential difference that means a conducting path gets established. So, ACTEL. ACTEL is a company which used anti fuse elements in some of its FPGA's family. But, now again they have also started using sram. Now xilinx is one of um the popular FPGA manufacturer which you uses an architecture which is based on look up table and srams which is completely which can be reprogrammable you can use it over and over again. So I am trying to give you a very big very brief outline.

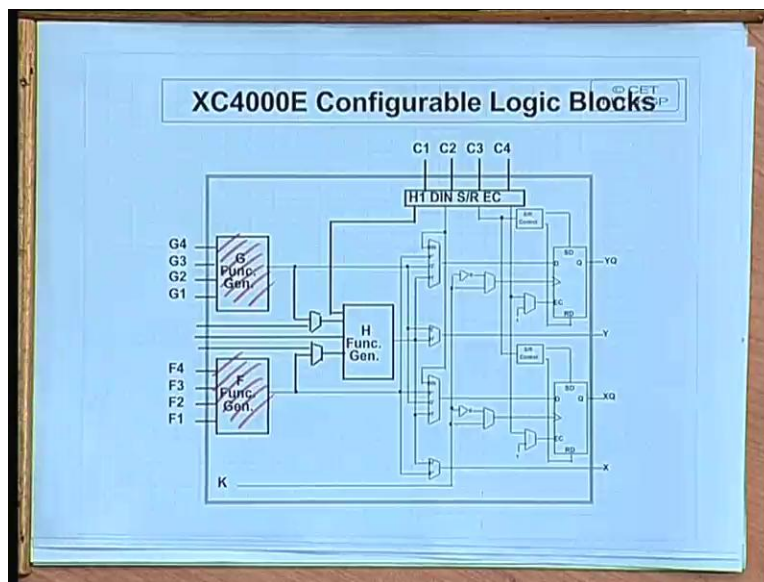
(Refer Slide Time: 17:53)



This picture is very small you cannot possibly see it very well but what I want to see is you should tell you are that. Say this is the floor of the FPGA there are some I/O cells on the periphery and inside there is an array of the logic cells the logic cells have a particular architecture we will talking about this architecture. So this architecture is capable of realizing some logic functions. The I/O cells also have some programmability you can say that whether it

will be an input port output port whether it will be stopped a clock etcetera. And interconnection among this line is again through some programmable interconnect this also we will see very shortly. So this is how the architecture looks like there are some I/O cells there are some logic cells which are called configurable logic blocks. And there are some programmable interconnects through which you can interconnect them. Now the configurable logic blocks constitute heart of the system.

(Refer Slide Time: 19:06)



This shows a very rough block diagram of the CLB's. Now each configurable logic block looks like this there are some combination logic modules there are some multiplexers for switching and there are there are two flip flops. Now each of this CLB's have a functionality like this and you can see that there are two blocks like this logic blocks like this with four inputs and one output. In fact each CLB can realize 2 4 variable functions any four variable functions and it can also realize some of the five variable function not all. Now the way they realize the four variable functions is very simple I will tell you how.

(Refer Slide Time: 20:00)

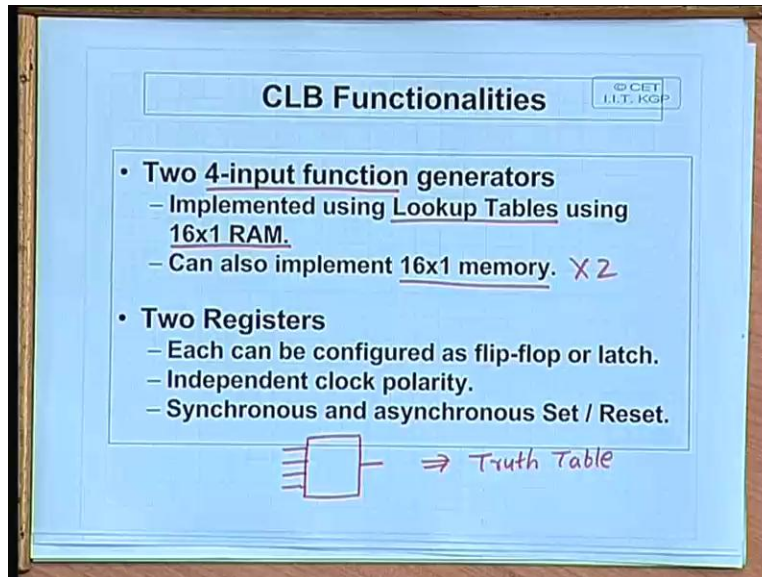
**CLB Functionalities** © CET  
I.I.T. KGP

- **Two 4-input function generators**
  - Implemented using Lookup Tables using 16x1 RAM.
  - Can also implement 16x1 memory.
- **Two Registers**
  - Each can be configured as flip-flop or latch.
  - Independent clock polarity.
  - Synchronous and asynchronous Set / Reset.

⇒ Truth Table

They do it using a technology called look up tables. But essentially they implement it by using a small ram array. See you try to understand it like this. Suppose you have a combinational circuit with four inputs and one output. Now if you construct the truth table of this circuit then output column what will be the output column output column will consist of sixteen one's. Now if you can store the output column in a ram, then if you can have architecture like this.

(Refer Slide Time: 20:50)

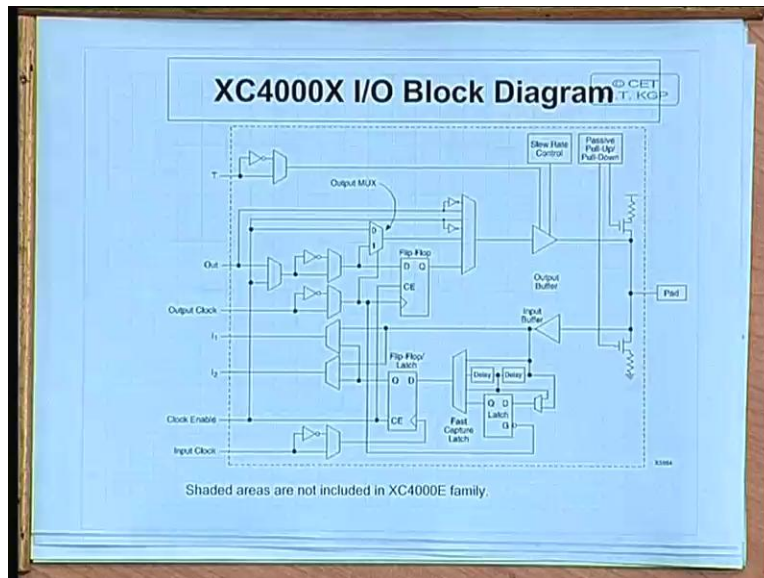


Like say you have a 16 by one ram and you have a 4 to 16 address decoder okay. So the inputs are fed here this decoder will select the memory location and this is your output. So in this way you can um you can realize any combinational function. Now of course this decoder means ram has to be fast in order to have to in order to have a reasonable performance. In fact in FPGA the rams are also designed using semiconductor technology which is fast not using the conventional memory technology using say capacitive elements for storage. So CLB's can implement for input functions in fact two of them have said that two out two such blocks and in some applications if you also need some memories. So the CLB can also be used to implement memories.

So, each of them can have 2 such 16 bit memories right and inside the CLB. There also two flip flops using which you can also realize some can say some sequential logic. There is some combinational part there also flip flops in the output of it you can basically partition with design in suitable way. So that you can just have some sequential logic also and there also fully programmable you can have it leading edge or falling edge trigger you can have it in the latch mode also level trigger it. Set reset can be synchronous with the clock or it can be asynchronous

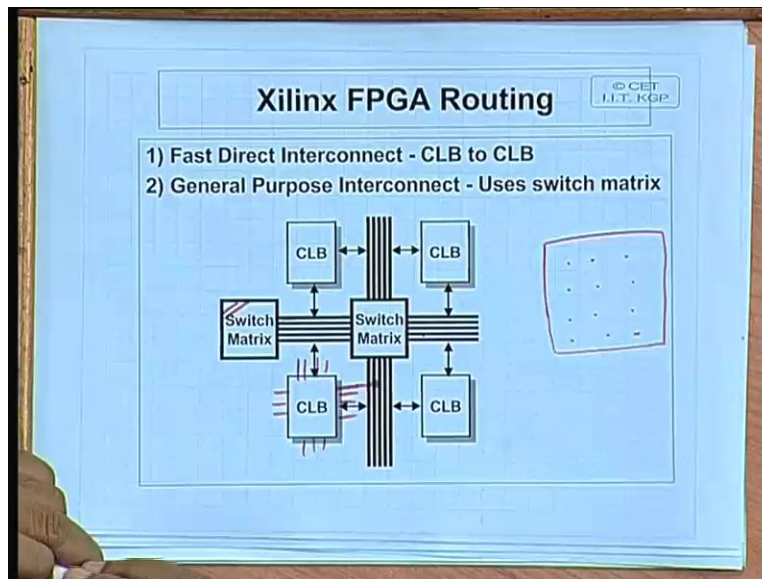
also so it has complete flexibility you can have anything you want. Just I am just showing this slide.

(Refer Slide Time: 22:48)



This is the diagram of the I/O cells, I/O cells, as you can see are also quite complex consist of a number of multiplexers drivers flip flops. So it depends whether you want them buffered bidirectional clock what kind of interface you require depending on that you have a general purpose interface block which again can be programmed to have the required or desired functionality. So I am not going into the detail of this.

(Refer Slide Time: 23:18)

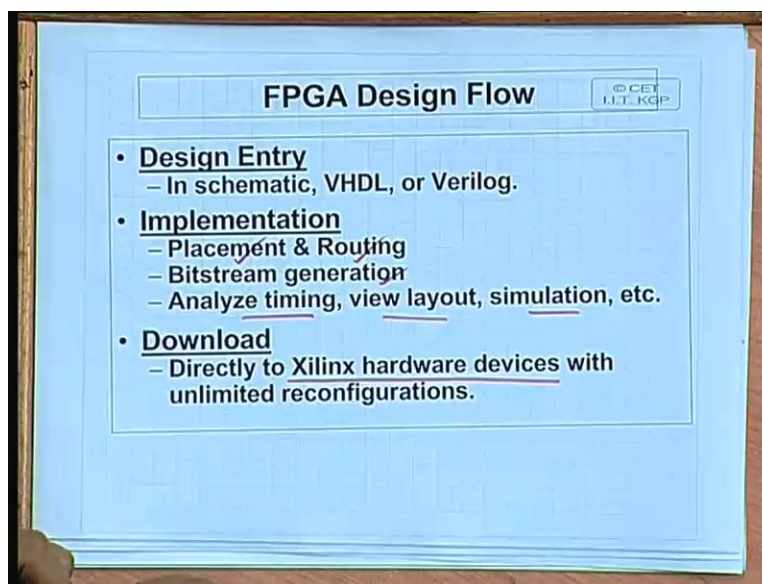


This FPGA routing is interesting. See on the floor of the chip I told you that the combinational logic block CLB's they are organized as an array. Now in between the array elements there are some switches matrices there are some fixed number of wires which are running between the switch matrices and also between the CLB's like this. Now a CLB has interconnecting lines on this side as well as on this side in fact has lines on all four sides. So you can connect a signal and of a CLB to one of these lines out here. This is programmed this can be programmed also or you can program this switch matrix also. So that say for example this line can be interconnected to this output line or can be interconnected to this output line as your need arises. So you can program this switch to specify which input has to be connected to which output so this switch matrix is also based on a static ram kind of a technology where inside there is a memory. So you basically fill up that memory with some bit pattern depending on that bit pattern the configuration of the interconnections um will be fixed up.

So as you can understand that they are fairly complex things but they are done in that way okay. So if your target is FPGA you see if your target is FPGA then many of steps in the design process are not needed floor planning placement are needed in the sense that which portion of your circuit will be mapped to CLB's; individual CLB's. You will have to partition the netlist so

that each partition can be mapped into a CLB. Routing is needed though it is programmable you will have to place this sub blocks into CLB such that the routing resources which are available they are sufficient to complete the interconnection. Because in FPGA what happens quite frequently is that you have a design you map it and find out the router is unable to complete the routing. Then you will have to reassign the placement and do something else so that you will again go back and re do.

(Refer Slide Time: 25:50)

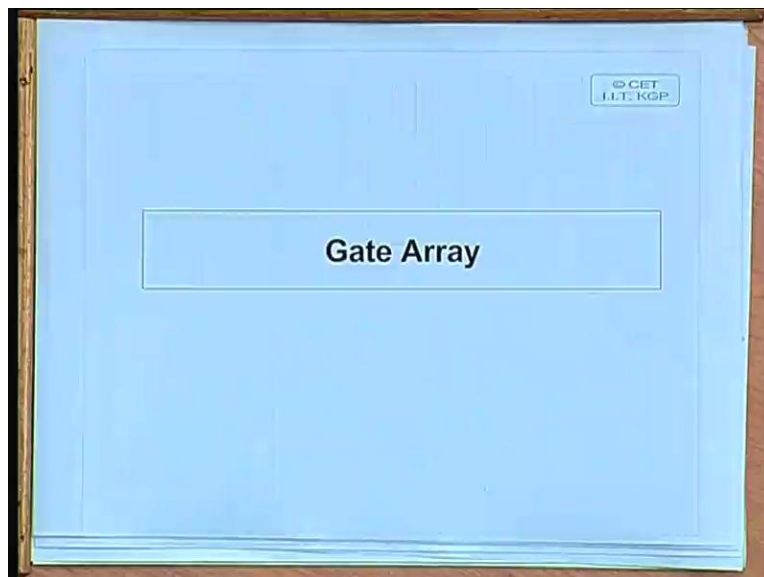


So FPGA design flow of course the first thing is design entry you can do as in schematic or in some description language. In case of implementation I told you placement and routing are required it is not that they are not required they are required because a cad tool will have to know that which part of net netlist it will be mapping to which CLB. And the neighboring net netlist should be mapped to neighboring CLB's otherwise the interconnecting lines will be very long delay will also be more. Bit stream generation is required in terms of the programmability whatever placement and routing you do ultimately in a programmable FPGA it will boil down to the bit patterns that will be required for the corresponding programming. So ultimately you will have to generate a long bit stream which will be downloaded on to the chip and which will be programming it according to your need okay. And there are some tools using which you can



analyze timing cd layout how the layout looks like you can simulate and finally after the bit stream is generated you can download it directly on to the xilinx hardware devices. Xilinx supported unlimited number of reconfiguration because they are based on static rams. [Student Noise Time: 27:12] High level synthesis need to be done everything has to be done everything has to be done. Synthesis has to be done, partitioning has to be done. But the target of partition is different they are not based on a technology library as such they are based on what kind of functionality the FPGA CLB's will support right fine. So the next level up if you can say complexity is something called gate arrays.

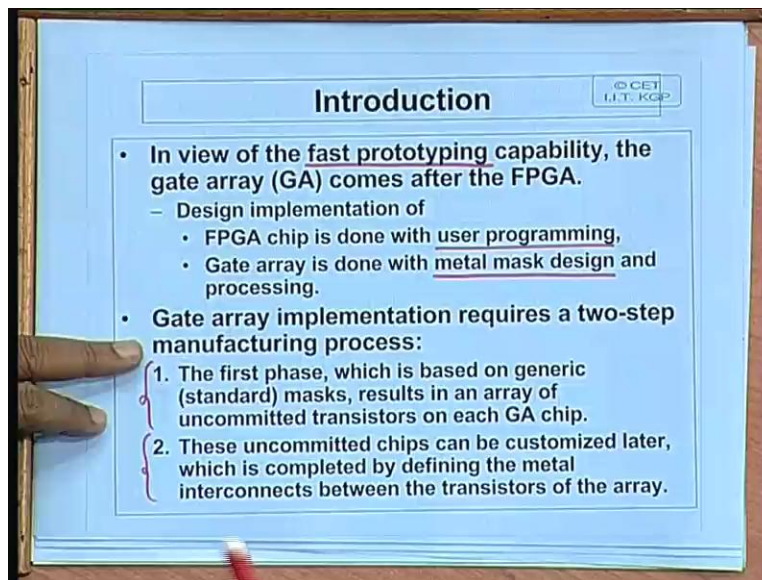
(Refer Slide Time: 27:47)



See FPGA provides complete you can say flexible to the user in the case that you can just buy an FPGA chip you have a developed system you straight away download the bit pattern you get a working chip ready. But in order but in doing that um if you look into the architecture of the chip there are a lot of redundant things which you had to put in. Switches lots of memories all around now mean in a practical design otherwise you would have not used like that. So in terms of performance wise your FPGA performance may be poor of course today FPGA chips which are available they work with a reasonable degree of performance. But still the frequencies are of the order of 25 to 50 megahertz or so not more than that but one step of the ladder in terms of

performance also complexity is something called gate array. It is not field programmable simply gate array so let us try to understand what this is.

(Refer Slide Time: 28:52)

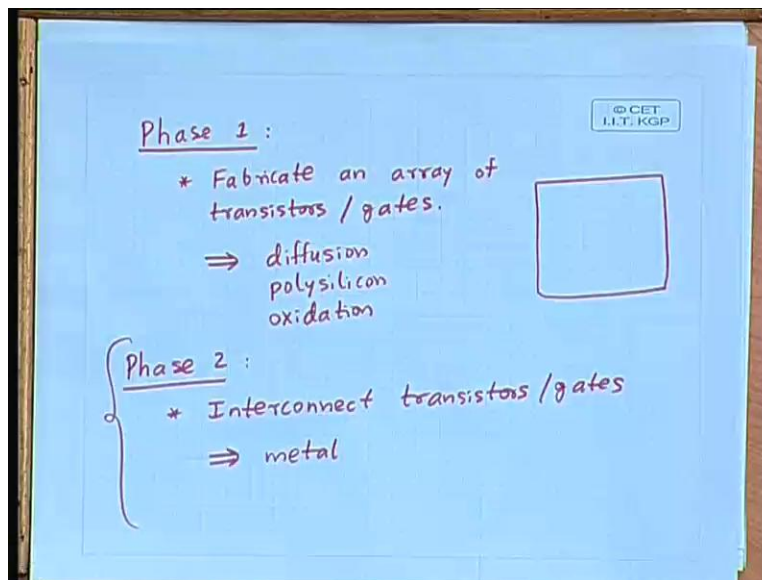


Gate array comes after FPGA in terms of this speed of prototyping. Gate array is also fast but in FPGA it can be in hours in gate array it can be weeks right. The difference is that design implementation of FPGA it is done using user programming through some proper cad tool and development system. But in a gate array you have to go back to the manufacturer there is some kind of mask design and fabrication step that need to be done depending upon the user request. But in that case what is the difference from a standard asic design, design there are differences the cost becomes less.

Gate array implementation requires two steps of manufacturing the first step is independent of your design. The second step is specific to your design what this means is that the cost of the first phase can be distributed amongst a number of users. Now in an asic the entire thing is your responsibility and the total cost you will have to bear but in a gate array the first part of the implementation that will give you some kind of generic gate array that can be customized for any

kind of designs the second step is that customization whatever you exactly require that is done in the second step. So the basic concept is like this in the first phase.

(Refer Slide Time: 30:45)

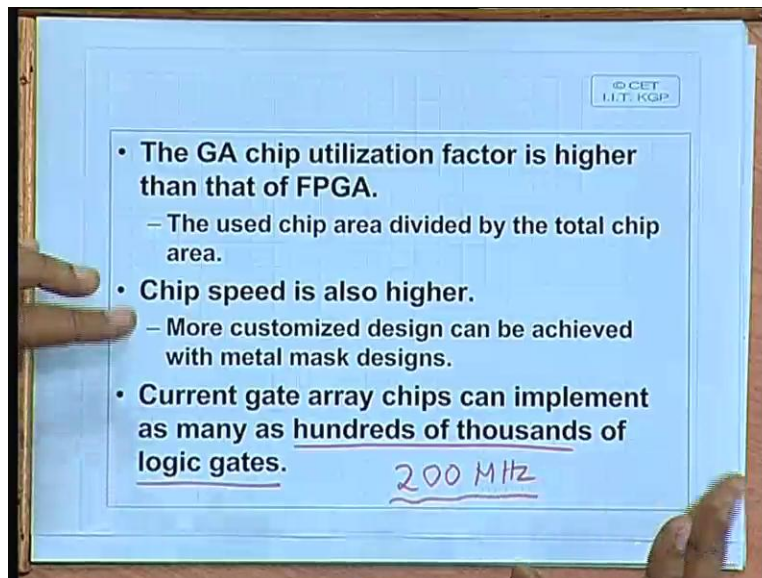


In the phase one you basically fabricate an array of transistors or gates depending on the kind of gate array you are using. Gates may be simple to input NAND gates. But on the surface of the chip you fabricate millions of such transistors or gates right. So in order to do that, you have to go through all these steps of oxidation poly silicon diffusion poly silicon. So diffusion layers poly silicon layers oxidation layers all these steps are already done. All these steps of fabrication are already done and you manufacture thousands of such wafers which contains millions of transistors or gates on them. Phase two says you interconnect the transistors or the gates. This is again done through some fabrication process through metallization. The transistors are being fabricated but without any metal layer.

Now you are starting to put on the metal layers now the metal layers are depending on the user requirements. So actually for a particular design you are basically paying for phase two the cost of phase one is very small since they are manufactured on a mass scale right. So this is the advantage of gate array that you can have or actually the manufacturer will provide you with a

very low cost uncommitted array of transistors depending on your design the manufacturer will interconnect them for you. And as I said since we have to go back to manufacturer fare for the metallization, so the turn around time can be several weeks but not more than that and the cost will also reasonably small. So now, means in terms of the performance and the utilization of the floor area.

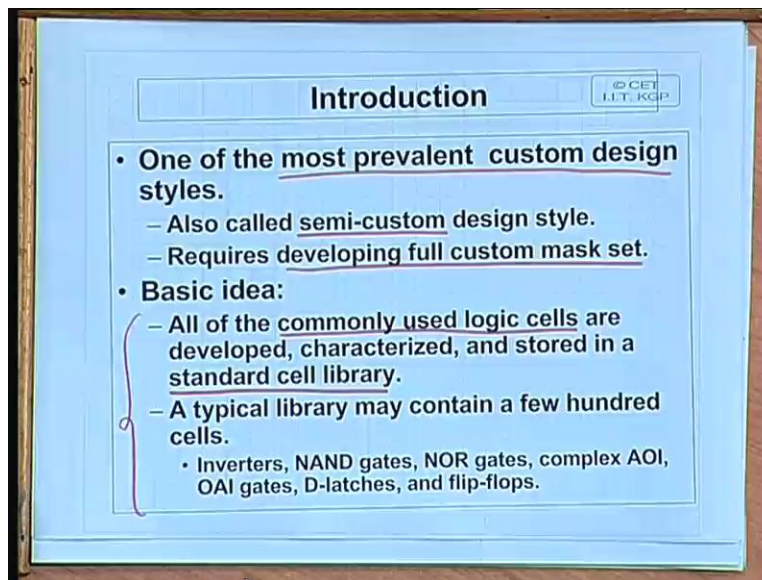
(Refer Slide Time: 33:30)



So obviously for a gate array the utility the chip utilization will be much higher than of FPGA. Because here on the chip we are not putting an in any useless thing in order to achieve the programmability memories multiplexer all those things are not there. They are only useful transistors and we are trying to utilize as many numbers of those transistors as we can in our design. Chips speed is also higher because all those useless things are not there so with the present day gate array chips you can have designs which can contain of hundreds of thousands of logic gates and they can work with reasonable speed say around 200 megahertz or so. These are the typical kind of performance you can achieve out of a gate array based design right. But if you want even higher speed even higher performance then you will have to carry out each step of fabrication depending upon your requirement. Do not just start with an array of gates and try to

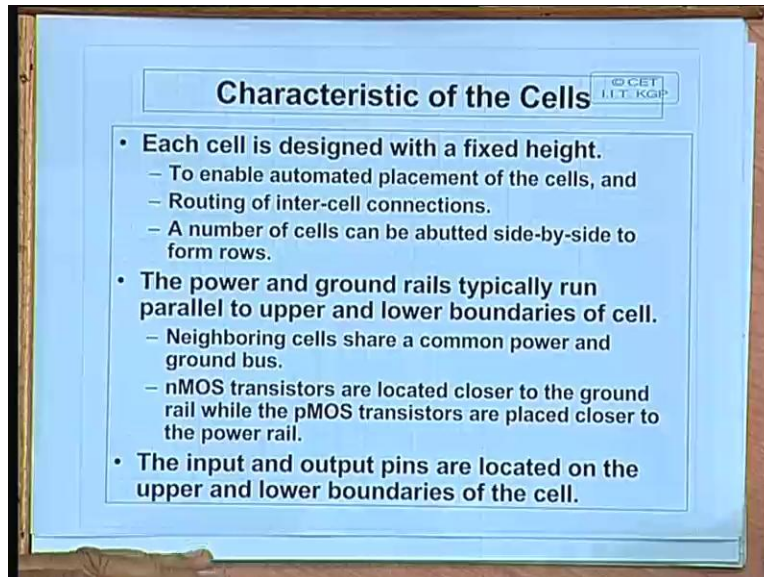
interconnect them to make a flip flop for example. So the next step of the ladder is something called standard cell based design this is also called semi-custom.

(Refer Slide Time: 34:50)



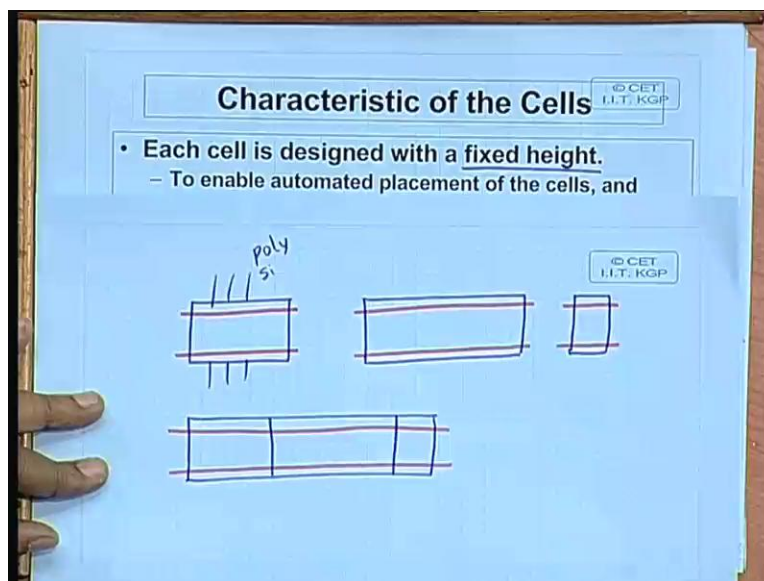
In fact this is the most commonly used design style today this is called semi custom because some steps which are done from scratch are done means actually quite exhaustively. But there are some steps where you simply pick up something from an existing library you are not redesigning everything something you are taking as a existing basic building block and you are trying to connect them together to form an actual design. But these chips require the entire set of masked set to be framed and fabricated so with respect to the fair facility you will have to pay for the entire thing. The chips will be manufactured entirely depending upon your design requirements basic idea is very simple it says that you have something called a standard cell library. And this standard cell library contains all the commonly used logic cell starting from simple gates to registers may be small ALU's multiplexers everything. Now a typical library can contain a few hundreds cells. Now in case of standard cell design the basic you can say difference as compared to any other design arises out of the way this standard cell library is created. The cells which are present in the library they have a very specific geometry or topology they are like this.

(Refer Slide Time: 36:34)



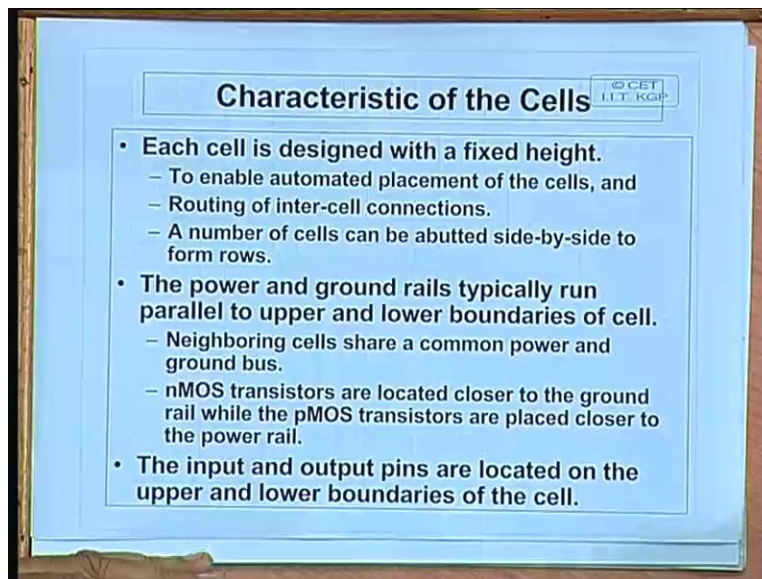
The heights of the cells are all identical they have fixed height. But widths may vary depending upon the complexity.

(Refer Slide Time: 36:50)



The reason is that it says that, if you have two cells suppose this is one cell I am representing as a rectangle. This is a layout there is another cell which may be longer there is another cell which may be thinner. So if all of them have the same height then I can pack them together place them side by side. So I can have a compact layout like this. But the first cell is like this the second cell is like this third cell is like this. This is possible if since the heights are the same well and since the heights are the same we can also do a few things like for example the power supply lines  $v_{dd}$  and ground. I can say that in all cells the  $v_{dd}$  and the ground line must be located out here so that when you put them together they will exactly touch and you have the continuous  $v_{dd}$  and power rails right. And the input output pins of this chips they are available on both sides like this and they are typically available on poly silicon layer or on a second level of metal if it is available fine.

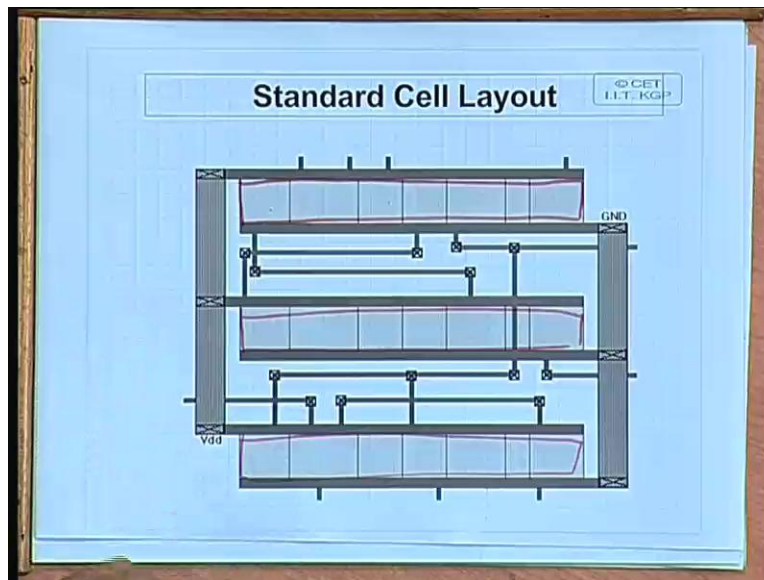
(Refer Slide Time: 38:11)



So this I have just mentioned the power and the ground rails typically run parallel to the upper and the lower boundary so that the neighboring cells can share that bus. And if it is CMOS there also you can follow some convention that in a standard cell the upper half will contain will consist of the p transistors the lower half will contain the n transistors. The reason is that when you put them altogether the top half will contain all p transistors. They can be fabricated with

respect to a common value. Similarly the n transistors can be fabricated with respect to a common value instead of isolated values all spread. And as I said the input and output pins are located on the upper and lower boundaries okay.

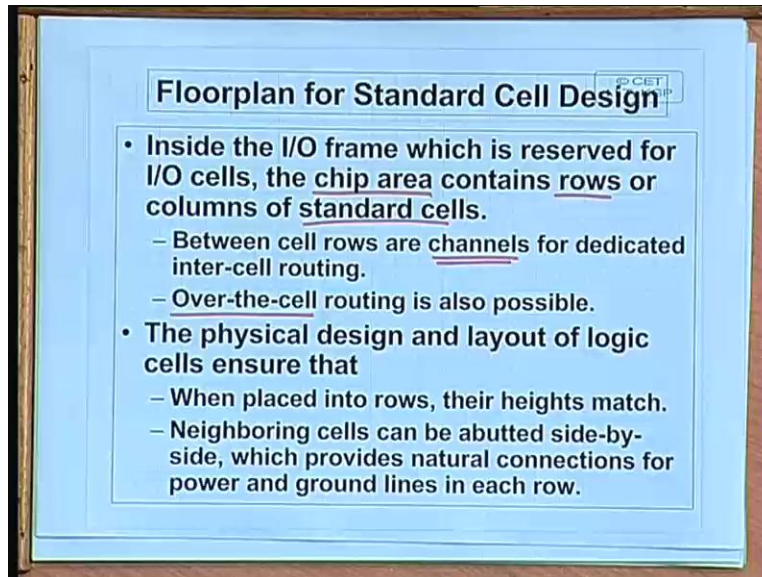
(Refer Slide Time: 39:00)



So a standard cell layout may look like this I have just given a very simple example here. These are the rows of the standard cells. So I have shown three rows 1 2 and 3. So as you can see in this three rows there are a number of cells which have been put together and in a standard cell layout the cells are put in rows and the space between the rows are kept for interconnection or routing. So as you can see there is some free space through which some interconnections had been carried out this I had a typical picture. And the v dd and the ground they can be fed from two different sides the v dd can be feeding the v dd lines and ground can be feeding the ground lines. So distribution of power also becomes very regular in this kind of a design okay.

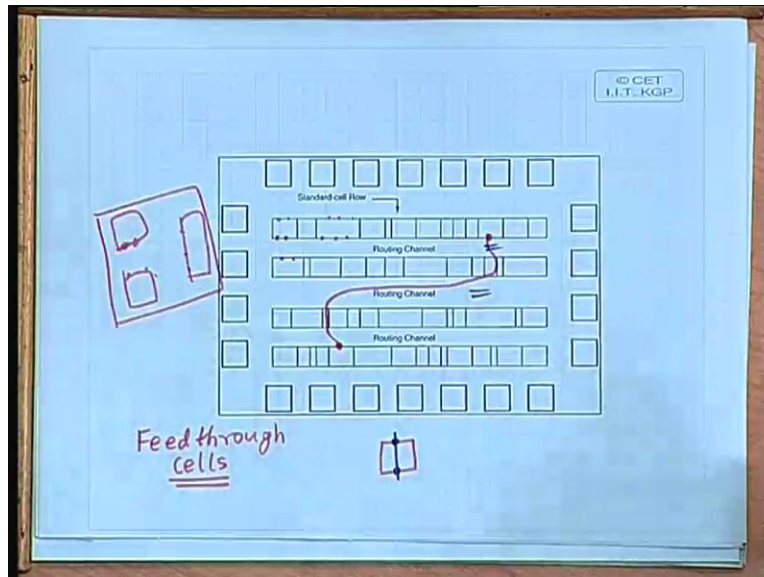


(Refer Slide Time: 40:10)



So as I said that with respect to the floor plan for a standard cell design the total chip area will contain rows of standard cells or columns whatever way you look at if you rotate it will become columns. Now between the cells rows you have the so called channels as I have shown which are dedicated for routing. And with some modern routing technology you can also have over the cell routing where means the place where you have this cells fabricated the wires can also go over the cell for the routing. So if it is on a separate metal layer you can even take those metal layers over the cell for interconnection purposes that save the total area for routing that we will see later. This is called the over the cell routing. And well since all the cells are of same height then when placed in rows they are put in absolutely regularly with the same height the neighboring cells can touch each other. Well one thing you now try to understand that your total layout will look like this.

(Refer Slide Time: 41:24)



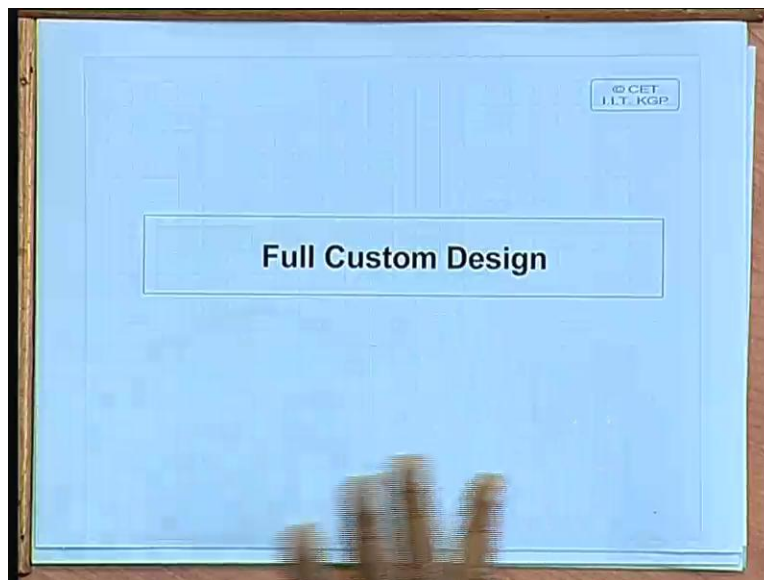
Okay several rows with I/O pins on the boundaries. Now with respect to the backend cad tools what does this placement and routing involve. Now placement will involve you have a partition sub block or a cell of your netlist you will have to map it to one of the rows. Because, now your technology library will consist of those components which are available in the standard cell library. So you will have to map each of them into one of the rows. Naturally the cells which are strongly connected they must be placed closer together right. Secondly the problem of routing becomes very simple here. Now here you can concentrate on only one kind of routing that is called channel routing there is a channel with some pins on top and bottom you will have to interconnect them through available tracks.

But if you think of a very arbitrary scenario where you had a chip with some component here some component here with pins arbitrary, arbitrarily located on all sides. So the problem of interconnecting them is much more complex. But here you have a very regular pattern the pins are located regularly on the top and bottom boundaries of the cells okay. But sometimes you may need to interconnect for example a point from here to a point from here it is not possible to place everything on the same roof if they are interconnected. So somehow we will have to take the

interconnection like this okay. Now in order to achieve this there are some standard so called feed through cells which are also used.

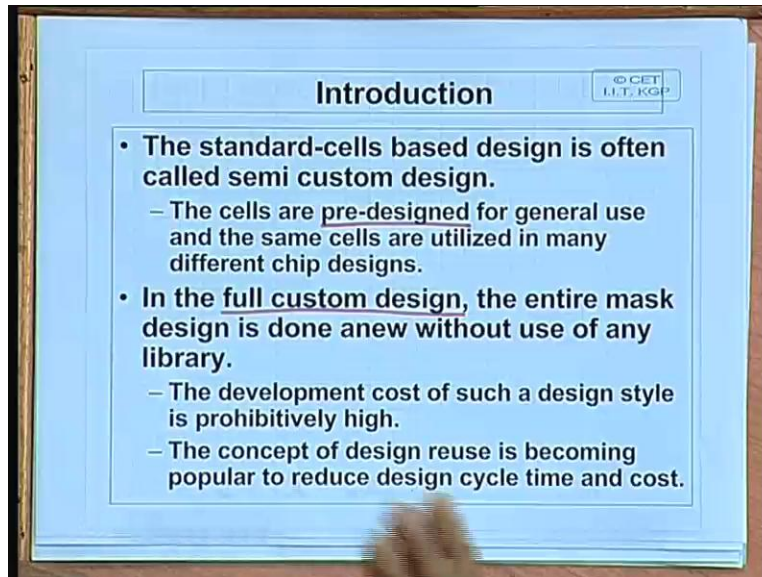
Feed through cells are very simple. It is a standard cell which does not implement any useful function just it has an interconnecting line connecting a point out here and a point out here through which you can take a netlist from one channel to another channel. Just in order to interconnect across channels. So your placement algorithm will give you an idea to the router that how many such feed through cells you require in each track. Because the placement tool will know that how many nets will be crossing across channels right fine? And finally a very brief a brief look at full custom design.

(Refer Slide Time: 44:18)



Full custom design it is entirely up to user there is nothing fixed or constraint which you have to follow.

(Refer Slide Time: 44:28)



The standard cell design is based on cell switch are pre designed with some specific geometry well in the full custom design it is not necessary that everything you have to design from the scratch. Because today people seldom design everything from scratch something may be there in library also. But they can come in all sort of shapes and sizes. There is no restriction that they have to be of the same height or anything. So in general full custom design says that you can design everything from scratch if you want to but in practice you use some design which you had already created earlier you try to reuse your design. So design reuse is an important concept in full custom design but the designs may be existing in any arbitrary shape and sizes as I had mentioned. So one area where full custom design is used quite popularly is in the design of a memory cell.

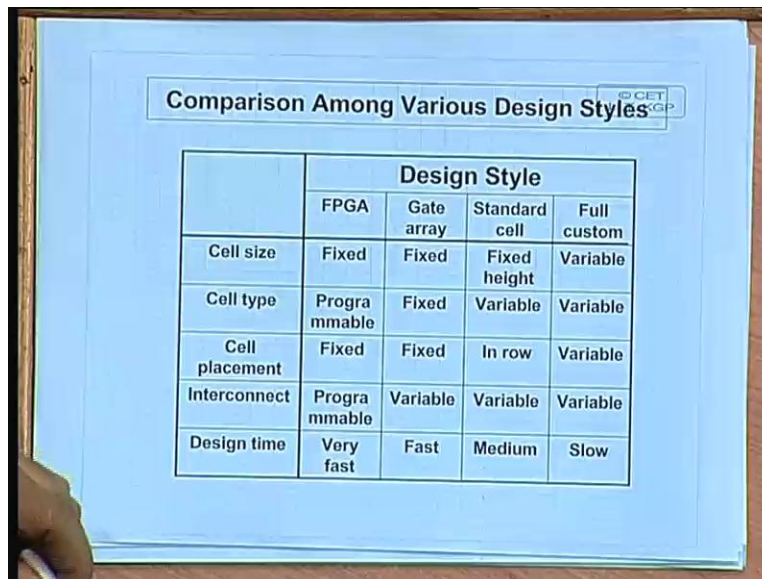
(Refer Slide Time: 45:32)

**Contd.** © CET I.I.T. KGP

- **The most rigorous full custom design can be the design of a memory cell.**
  - Static or dynamic.
  - Since the same layout design is replicated, there would not be any alternative to high density memory chip design.
- **For logic chip design, a good compromise can be achieved by using a combination of different design styles on the same chip.**
  - Standard cells, data-path cells and PLAs.

See memory cell layout is very regular. But you want to design it in such a way that it is very highly optimized. So in that respect this is one candidate for full custom design and for standard logic chip design we typically try to do a compromise between some standard cells data path cells like ALU's multipliers and PLA's for implementing some means random combination logic. So we typically mix them out together and our chip looks quite regular but the blocks are of different sizes and shapes.

(Refer Slide Time: 46:15)



	Design Style			
	FPGA	Gate array	Standard cell	Full custom
Cell size	Fixed	Fixed	Fixed height	Variable
Cell type	Programmable	Fixed	Variable	Variable
Cell placement	Fixed	Fixed	In row	Variable
Interconnect	Programmable	Variable	Variable	Variable
Design time	Very fast	Fast	Medium	Slow

So just finally to compare between the difference design styles. So with respect to the cell size for FPGA and gate array, they are fixed but for others they are variable for standard cell they have fixed height but width may vary. Cell type is programmable for FPGA fixed for gate array. But for these they are variable cell placement FPGA gate array they are fixed the geometry of these cells are fixed. But you can map them differently. In standard cell you can put them in rows full custom anywhere interconnects FPGA programmable but for these you can do anything. Design time FPGA is very fast gate array is medium full custom is the slowest.

So looking at the spectrum you can really look at what you want you need speed you need fast design turnaround time. Typically when in you are doing a big design you first do a prototyping on FPGA and if it works according to your need then you can go for a full asic design typically using standard size. So from our next class we would be looking at the different steps of the backend design process and we will be looking at specific algorithms their restrictions where they are applicable and in which design style they are applicable. And we would be continuing our discussion on back end design subsequently. Thank you.