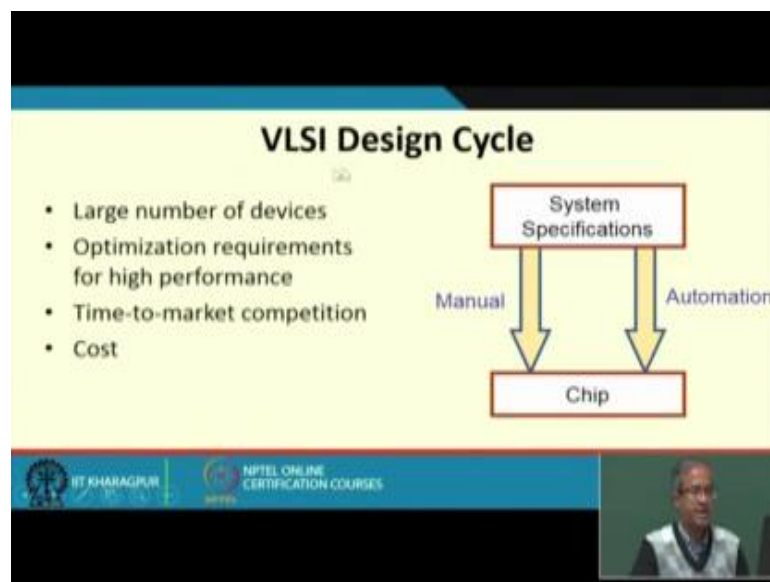


VLSI Physical Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 03
VLSI Design Styles (Part 1)

So, now shall be talking about some of the VLSI design styles, which would be useful in understanding the various physical design techniques and algorithm that we shall be discussing in the modules that we will be following. So, in the first part of the VLSI design styles here we shall be talking about some of the so called design styles and how they impact the design issues.

(Refer Slide Time: 00:54)



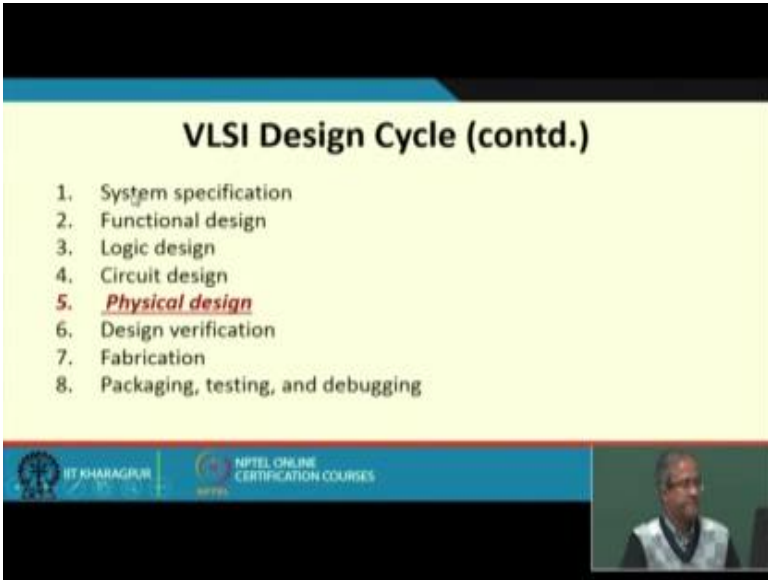
Now, let us first try to understand the VLSI design cycles; this I had emphasize during the first lecture also that with the advancements in semi conductor technologies, today we are having to handle very large number of devices, which can go up to and beyond a billion transistors in a device.

Secondly there are many applications which demand high performance and also low power because many of the applications today they operate on battery they are mobile handle devices like mobile phones. So, performance and power consumption have two very important requirements and of course, time to market competition is very high. So,

today no manufactures can effort to spend a lot of time in creating and optimizing the designs. So, this will have a direct impact on the cost.

So, in contrast to the earlier days, where starting from a specification to the final fabrication of the chip, we could have followed a manual approach today things are changed. Now because of the sheer size of the systems, and the complexity of the processes involved this automation is mandatory this is not a choice anymore right. So, this is one thing we have to very clearly understand, that is why we are looking at the VLSI design flow a physical design and things like that.

(Refer Slide Time: 02:51)



The slide displays the VLSI Design Cycle (contd.) with eight numbered steps. Step 5, 'Physical design', is highlighted in red. The slide also features logos for IIT Kharagpur and NPTEL Online Certification Courses, and a small video inset of a speaker in the bottom right corner.

VLSI Design Cycle (contd.)

1. System specification
2. Functional design
3. Logic design
4. Circuit design
5. **Physical design**
6. Design verification
7. Fabrication
8. Packaging, testing, and debugging

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, another look at VLSI design cycle. So, here we have mention the steps slightly differently. So, we say that we start with system specifications, we go through functional design which is something acting to register transfer level design as I had mentioned, the designs which are consisting of registers, failures, multiplexers etcetera. So, this is at the highest level of the design, then that design can be refined into logic level design; logic level design where we talk about the gates and the flip flops in our netlist. So, those RTL level components are decomposed into gate level netlist, say a multiplexer is realized using gates a flip flop is realized using gates and so on.

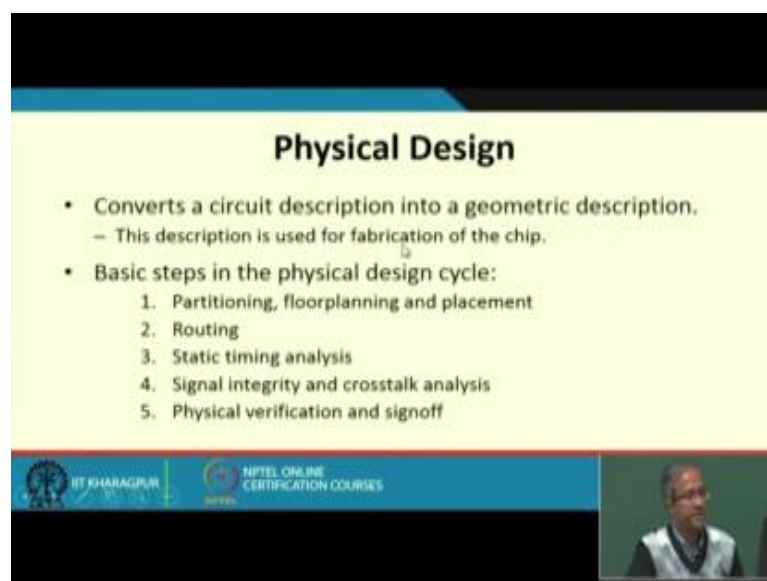
So, then at the next level we have circuit design; circuit design means this is the transistor level design. Now every gate or flip flop are now realized using say typically cmos transistors at this level. So, now, we have very large number of NMOS and PMOS

transistors and their interconnections through way they are connecting together then netlist of transistors. Now once we have this netlist of transistors, the next sequence of step which we have merged and shown as a single macro step is the physical design; this physical design is the main topic of discussion in this course that we are looking at. So, this physical design itself will consist of a number of different steps, which takes a netlist as the input and we will go through a number intermediate steps and create a final version which is ready for fabrication.

A final specification of your system or of your design, which can now be directly send to the fabrication facility for the final fabrication. So, that is the so called purposes of the physical design that macro step that I showed here and of course, there are some other this issues that need to be looked at, before you go for fabrication because fabrication is an expensive step. So, if something goes wrong there then you have to pay a lot. So, you have to go through various cycles of design verifications, layout verifications design sign off before you can actually go for fabrication.

After your chips of are fabricated, you have to package them in the die, you have to properly test them and if something is not working correctly you have to debug them why? This circuits or chips are not working correctly. So, was there any mistake or arrange the design, was there any error during the fabrication and so on. So, you need to diagnose the cause of the failures there. So, this is the so called VLSI design cycle.

(Refer Slide Time: 06:21)



Physical Design

- Converts a circuit description into a geometric description.
 - This description is used for fabrication of the chip.
- Basic steps in the physical design cycle:
 1. Partitioning, floorplanning and placement
 2. Routing
 3. Static timing analysis
 4. Signal integrity and crosstalk analysis
 5. Physical verification and signoff

IT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, coming back to this macro step of physical design, let us see in detail that what are the steps that are involved here in and the topics that we shall be discussing as part of this course. So, I am repeating physical design essentially is a sequence or set of steps, that converts a circuit netlist into the final layout, which is set of rectangles suggest set sometimes these are referred to as geometric descriptions. So, this geometric description is can be sent directly to the fabrication facility to manufacture the chips.

Typically this geometric description is created in the form of a standard file like GDS 2; we shall be talking about these things later. So, means how this files look like, and what they contain. And in the physical design cycle, so as part of this course in the next few course modules, we shall be discussing the issues like partitioning floor planning placement, routing. So, how the interconnection among the modules or the blocks or the circuits nodes are carried out; static timing analysis is very important the circuits or the netlist that we have created. So, whether it is able to satisfy our timing requirements.

Suppose we want to create a design that is suppose to work at one giga hertz clock, and you find after designing the circuits netlist, that we can only go up to 800 mega hertz. So, there as to be some tuning, some changes that need to be done in the design so that we can jack up the clock frequency for the, so the static timing analysis is step which helps us in identifying the critical parts, identifying the timing delays, and other things so as to take this decision very consciously and intelligently.

Then of course, signal integrity and crosstalk is a very important issue in VLSI design, because of the sheared size of the features; they are so narrow, they are so close together. So, when there are some signal changes going on in one line, if there is a line which is very closely running to it, there will be some coupling on the other line so some crosstalk issues can come into the picture. So, signal integrity crosstalk modeling and to ensure that the final layout that I have created will not adversely suffer from these when the chip gets fabricated, these are very important.

So, signal integrity and crosstalk analysis are something that we shall also be talking about and finally, various OS to physically to verify the layout before we go for signoff. Signoff means we are saying that we are happy with everything we have done; now we are giving the thing for fabrication that is this step of signoff. So, signoff is a very important thing because if anything goes wrong after signoff, that as a very huge cost

implication; because fabrication itself is a very expensive thing and if you do the fabrication and later on find that there are something wrong with you design of the layout then you adjust wasting your money right.

(Refer Slide Time: 10:05)

The slide is titled "Various Design Styles" and lists three main categories:

- Programmable Logic Devices
 - Field Programmable Gate Array (FPGA)
 - Gate Array
- Standard Cell (Semi-Custom Design)
- Full-Custom Design

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom, along with a small video inset of a speaker.

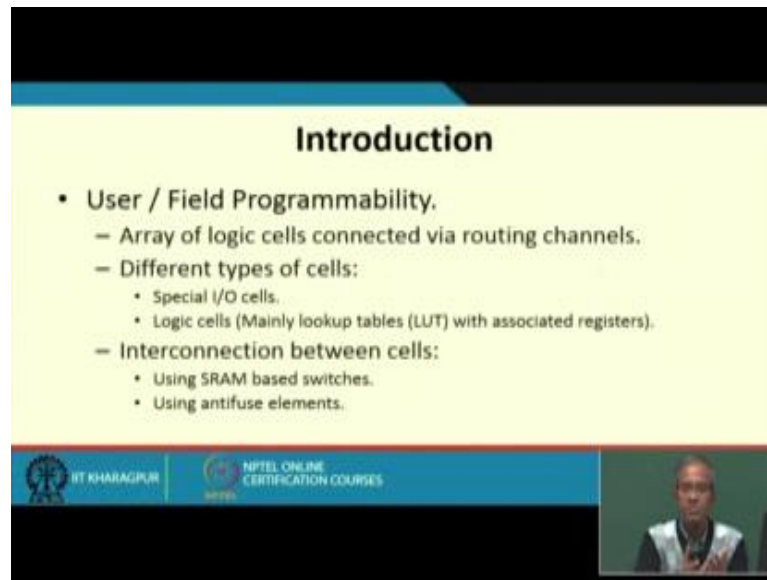
So, now shall talk about some so called design styles. So, what are the design styles? Design styles are various broad ways in which you can create our VLSI chip, very broadly speaking today people create their designs on chips using technologies like field programmable gate array, gate array, standard cell which is also called semi custom design or full custom design. Let us look at these different design styles separately, so that you can have an idea regarding the merits and demerits of this approaches and why and when should one use one approach I means over the other. So, let us see the differences.

So, the choice of selecting a particular design style depends on a number of issues of course, hardware cost. See the FPGA field programmable gate array at the cheapest, but these circuit delays are possibly the longest there. So, hardware cost, circuit delay and also the total time required for you to create your design and map it to the hardware these are all important. There are some applications or some scenarios where we need to minimize the time, but you can of course, compromise on the other two.

There are some applications where delay is very important, you want to minimize the delay, but may be you can give some more time for your design. So, this is basically a

tradeoff among these design parameters which are often conflicting. So, means experience designers can do a proper tradeoff between these parameters. Now let us look at the basic concept or idea behind field programmable gate arrays in the next few slides.

(Refer Slide Time: 12:29)



Introduction

- **User / Field Programmability.**
 - Array of logic cells connected via routing channels.
 - Different types of cells:
 - Special I/O cells.
 - Logic cells (Mainly lookup tables (LUT) with associated registers).
 - Interconnection between cells:
 - Using SRAM based switches.
 - Using antifuse elements.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

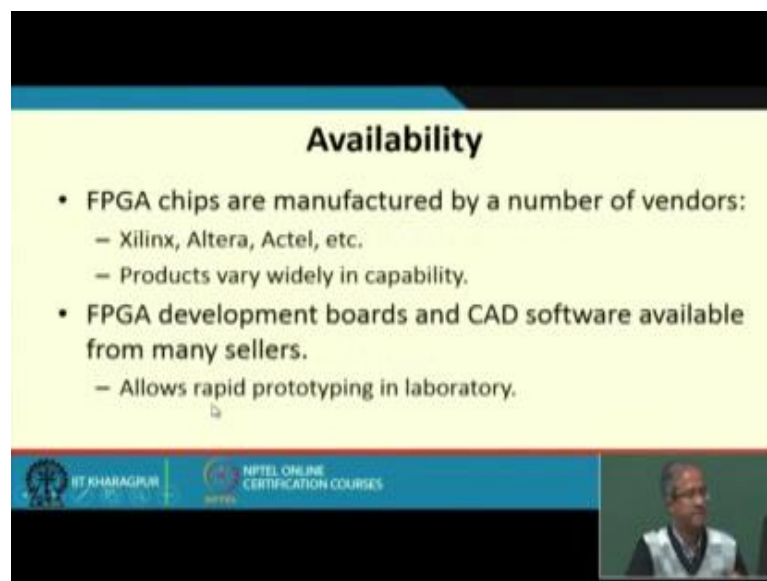
Well; field programmable gate array as the name implies field programmable means, the device is programmable in field. Now as a designer which is my field, my field is my laboratory. So, a field programmable gate array or an FPGA in short it is called I should be able to program it in my lab, I should not have to go through some expensive fabrication facility to get my chip fabricated there, but rather I can buy a chip of the self FPGA component, bring it to my lab and I can do with the necessary programming there and I get my desired circuit on that FPGA chip; this is how FPGA basically works and what it means.

So, how does it achieve the programmability? Well I am not going into the very detail because it is slightly out of this scope of this particular course, but I shall be talking about the salient points or the features. So, an FPGA consist of an array of logic cells, this array means a large number there can be thousands of such logic cells, which are connected via routing or interconnection channels which are also programmable, which means the way this logic cells are connected this also can be altered as per your requirements ok.

So, you have the logic cells, you can map some of your gates or design into the logic cells, then you can interconnect them as per your wish these are all programmable. So, there are a different kinds of cells in FPGA, there are special input outputs cells which allow you to take this signals to the pins. Some will be input some will be output with different requirements, some will be input only output only by direction and some tries it control stroked there are so many requirements. So, you can have this special I O cells, and you can have the logic cells. So, we shall see the later that this logic cells are typically implemented using something called lookup tables or LUT's, and this is routing channels that I said that are also programmable, they are created using static ram or using anti fuse elements I will mention then later.

These are two different technologies which some of the FPGA manufactures they use differently in the designs.

(Refer Slide Time: 15:28)



The slide is titled "Availability" and contains the following text:

- FPGA chips are manufactured by a number of vendors:
 - Xilinx, Altera, Actel, etc.
 - Products vary widely in capability.
- FPGA development boards and CAD software available from many sellers.
 - Allows rapid prototyping in laboratory.

The slide also features logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES at the bottom, and a small video inset of a speaker in the bottom right corner.

The FPGA chips are that available from a number of vendors like Xilinx, Altera, Actel there are many others. They manufacture the FPGA chips, this products vary widely in capability like when I say widely in capability I mean that there are products which are meant for first time use or the beginners use, they can be used in the labs in classes for training purposes they are really cheap; there are a few thousand rupees of course, may be, but there are other very sophisticated FPGA chips which can run at clock frequencies of the order of giga hertz, they can provide very high throughput and they can be used

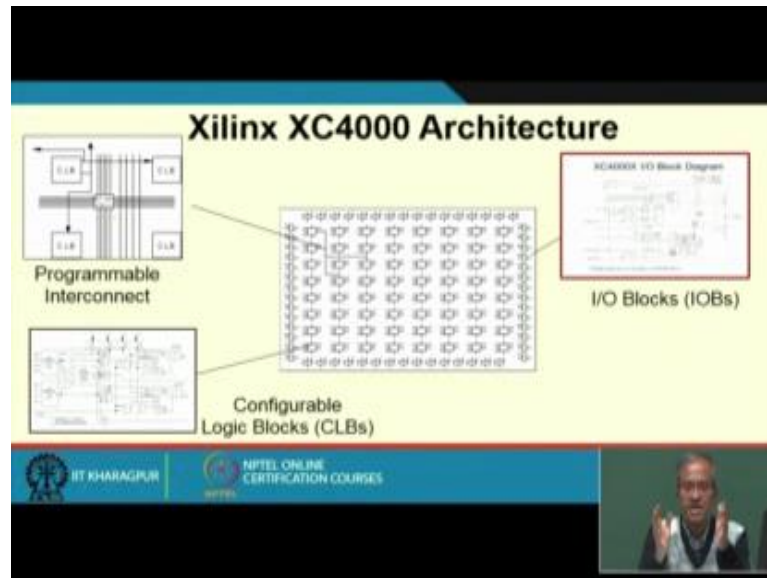
for really cutting edge applications where some you can say I mean otherwise you would have used some chip, which is created using application specific manner which are called assets; means I can give an example say the missiles that are manufactured.

So, I give an example same is you must have heard of this scud missiles, which are used in the world fair with (Refer Time: 16:47). So, these scud missiles have some FPGA's means built into it as part of the control system. So, what those FPGA's do, because of their programmability, so when the missiles is in its part depending on the environment, depending on the location of the target, the targets can also move with a. So, it can change its program internally, it can there is a mechanism in which the FPGA configuration can be altered dynamically so that this scud can hit the target very accurately.

So, this was one application where FPGA's have been used in a real time scenario with some very stringent timing constraints fine and other thing is that this FPGA chips are all right, but FPGA development boards are also available from many companies and sellers, and along with that you have the computed design tool in which you have the complete synthesis and layout and routing tools incorporated in built there, so that you can write your specification in say age dealer very log, and from that you can directly map your design on to your FPGA board or FPGA chip, and you have your design ready in hardware.

It is a very easy task, and this whole thing can take as some minutes or maximum an hour may be. So, it is not that time consume. So, this allows very rapid prototyping in the laboratory. So, there are many applications where the turnaround time is very important say FPGA's give you a great benefit in that respect.

(Refer Slide Time: 18:39)

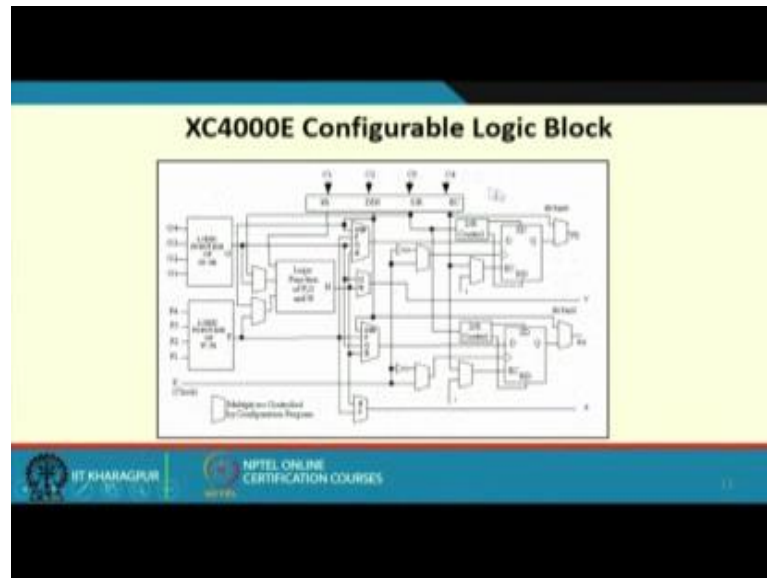


Now, let us look at a very quick look at to one of the FPGA's families, which is not very new of course,, but I have taken it as a representative, just in order to just explain some of the main features of FPGA chips.

Since FPGA looks like this in this central block frequency, there are I O blocks means along the boundaries. Because input output pins will be there along the boundaries, and along with every pin there is one input output block and this input output block can be programmed as per the requirement of that particular signal input or output that pin. Similarly there as some logic cells typically arranged in array fashion in between, these are called configurable logic blocks, these are logic blocks which can be programmed which can be modified in terms of the functionality and of course, the interconnection between this logic blocks that is also programmable.

So, you have so called programmable interconnects. Say FPGA will consist of this CLB's, the I O blocks and programmable interconnect. So, externally from outside you are able to program this CLB's, you are able to program their interconnections, you are able to program the I O blocks. So, given any design by a process which is called technology mapping, you can map parts of your design to the CLB's and interconnect them in a proper way so that whatever was your design intend that gets implemented on the FPGA in hardware fine.

(Refer Slide Time: 20:47)



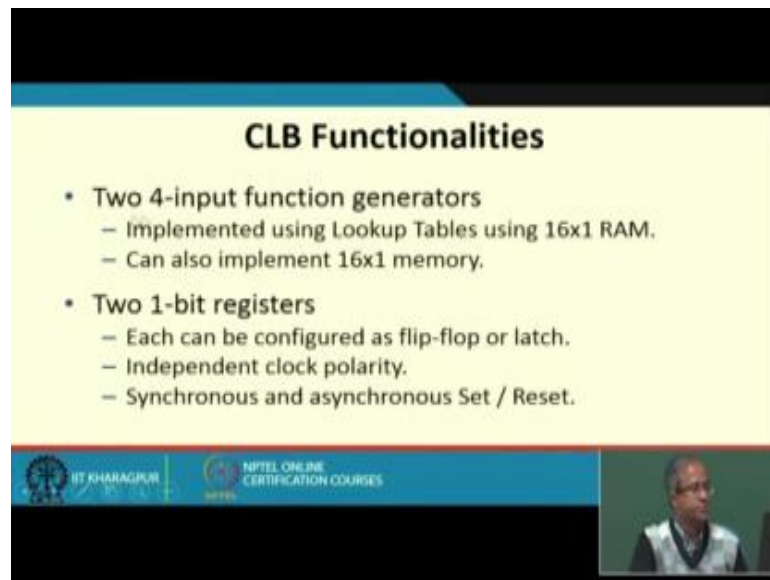
Let us look at this CLB's very quick look; see this CLB as you can see there are a lot of blocks inside, but without going into the detail what I want to do tell you is that the two most important blocks are here, these are two blocks which implement 4 input logic functions. So, I shall just tell you how these are implemented shortly. So, these two blocks can implement two 4 variable logic functions and the outputs can be available on this 2 output lines x and y there are parts. When the addition there are two flip flops using which you can also use the CLB as storage cell. So, you can either compute some logic, you can also use it as a memory two bit memory or you can also use it as some small finite set machine where some logic function along with some flip flops are there; and the other multiplexers and other blocks that you are seeing inside those are used for programming.

So, from outside you can feed some signals, so that this interconnection can be altered or programmed exactly how you want. So, I shall tell you how some of these at least.

(Refer Slide Time: 22:24)

CLB Functionalities

- Two 4-input function generators
 - Implemented using Lookup Tables using 16x1 RAM.
 - Can also implement 16x1 memory.
- Two 1-bit registers
 - Each can be configured as flip-flop or latch.
 - Independent clock polarity.
 - Synchronous and asynchronous Set / Reset.

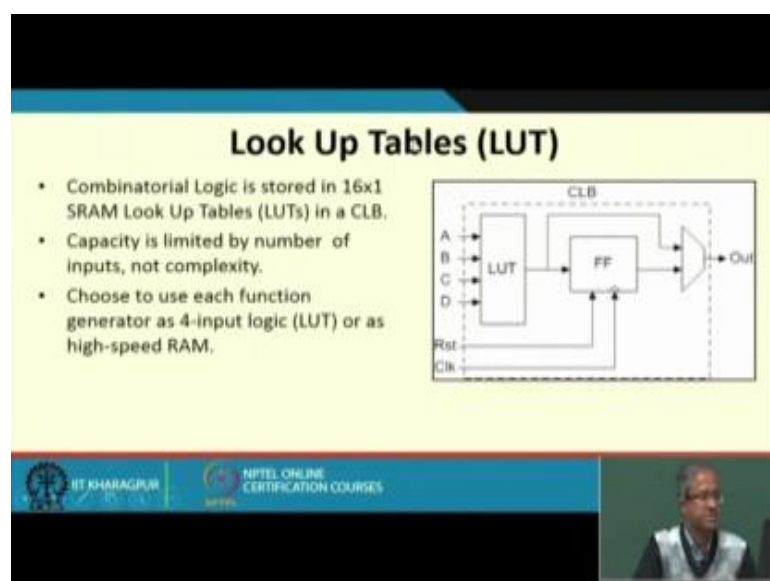
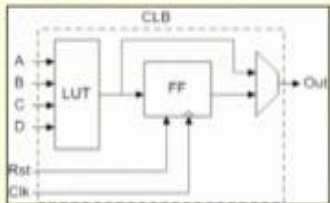


So, as I mentioned the basic features of this CLB's to implement two 4 input functions in these 2 blocks, these 4 input functions are implemented using look up tables using 16 by 1 ram. So, here I shall be explain this a little later, this can also be used to implement 16 bit memory. So, this we shall see a little later. There are 2 flip flops as I said there are two flip flops one here and one here, this flip flops can be configured as a block flip flop or a latch level sensitive latch. So, you can use it to be triggered in the leading edge or the falling edge of the clock, set reset can be either synchronous or asynchronous; so it is entirely programmable, right.

(Refer Slide Time: 23:26)

Look Up Tables (LUT)

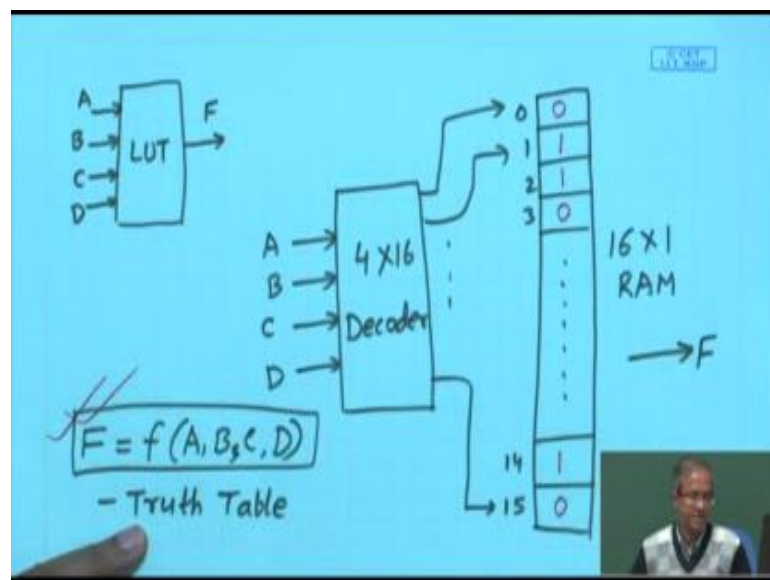
- Combinatorial Logic is stored in 16x1 SRAM Look Up Tables (LUTs) in a CLB.
- Capacity is limited by number of inputs, not complexity.
- Choose to use each function generator as 4-input logic (LUT) or as high-speed RAM.



Now, coming to look up tables so I have just mentioned in the previous slide, that each of these 4 bit lookup table can implement 4 Input logic function or can be used as a 16 by 1 memory ram let us see how. So, CLB this is the simplified diagram of CLB, there are lot of other things as I showed in the earlier diagram, but those are mainly used for programming, but essentially a CLB consist of a 4 input lookup table, a flip flop which can be by passed if required and an output multiplexers if flip flop reset a clock.

So, let us see how this LUT works. So, essentially the LUT works like this. So, as I said an LUT will be having 4 inputs A B C and D and it will be having an output let us say F, but internally what is there?

(Refer Slide Time: 24:17)



So, internally we have something like this that is one block which is a 4 to 16 decoder. So, this decoder will take this A B C D as the inputs, and there is a 16 by 1 memory unit ram. So, there are 16 cells. So, starting with address 0 1 2 up to 15, this decoder will be having 16 outputs and it will be selecting one of the cells, and that whatever cell you are selecting that will be obtained as the output this will be your F.

So, essentially this LUT works like this, you have a decoder depending on A B C D also you will be selecting one of the memory cells. So, each will be equal to one bit, and that one bit will be available in the output of the memory that you take as F right. Now you see this LUT is being implemented as a ram. So, what are the advantages? The advantage is that externally if you can load the ram with some bit pattern, you can change the

function F. Like say I want to implement some function of A B C D so what I do? So, I draw the truth table; so for all possible 16 combinations of A B C and D I write down what is my expected F.

Let us say my expected F is for 0 0 0 0 0 output should be 0, 0 0 0 0 one output should be 1, for 2 it should be 1, for 3 It should be 0 like this, let us say this is 1 this is 0 and so on. So, whatever is the output column on the truth table I load that same bit pattern in this ram. So, once I do it I have implemented this function right. So, you understand this is how this programmability of this CLB or LUT works, and LUT is implemented using a ram and by suitably loading any bit pattern in the ram I can implement any truth table.

So, any function with up to 4 variables can be implemented using a LUT right. Now in addition I can also use it as a 16 by 1 ram, just I am not using as a function I am using A B C D as the address inputs. So, with this I can read some data, with this I can also write some data. So, I can also use this same LUT structure as a random access memory with 16 locations each with one bit. See there are many designs where you required to have some memory sub system memory units. So, this kind of LUT's can be used to advantage to create those small memories as well fine.

(Refer Slide Time: 28:50)

LUT Mapping: An Example

- A function: $f = A'B + B'CD$
- The mapping process:
 - Create the truth table of the 4-variable function.
 - Load the output column into the SRAM corresponding to the LUT.
 - Apply the function inputs to the LUT inputs.
- Any 4-variable function can be realized.
 - Netlist to LUT mapping is an interesting design tradeoff.

IT KHARAGPUR NIPTEL ONLINE CERTIFICATION COURSES

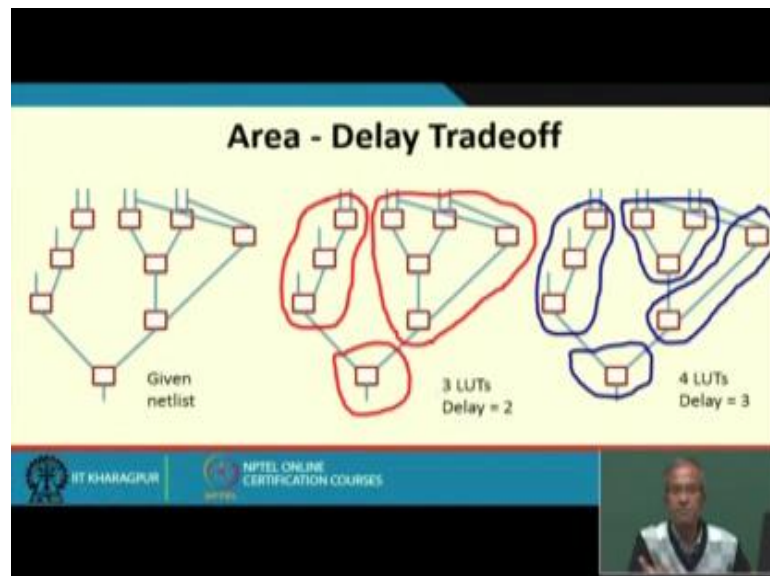
So, as I said, here I am repeating here. So, if you are given any arbitrary function of 4 variables let say this. So, the mapping of this function to LUT's is very simple, you create the truth table of this function, load the output column of the truth table, into this

static ram, corresponding to the LUT and once you have done that your LUT implements this function nothing else to be done. So, the point to note is that any 4 variable function can be realized, netlist to LUT mapping is very interesting. So, let me try to explain briefly; you see normally you forget FPGA think of the normal gates.

Suppose I want to implement a function, and I have certain type of gates which I can implement using my technology like for example, in CMOS I can create nand, nor, not this kind of gates; that and or these are not easy to implement, so I should always try to map my original function description, into a circuit which consist of only nand and nor gate. Let us see this process is called technology mapping. So, I know what I can do or what I can realize. So, my original specification I try to map using those functional blocks or cells, this is called technology mapping, but for FPGA the entire concept of technology mapping as become different.

Now, any sub functions with 4 variables I can map into the LUT. I do not think about specific gates or functions, I do not think about a nand gate or gate nand gate or nor gate I think of any 4 variable function if I can extract I can map it into one LUT. So, the mapping becomes very simple.

(Refer Slide Time: 31:02)



So, let me take an example, this will also illustrate area delay tradeoff; let us say I am showing a given netlist these small rectangles can represent some gates, I do not care

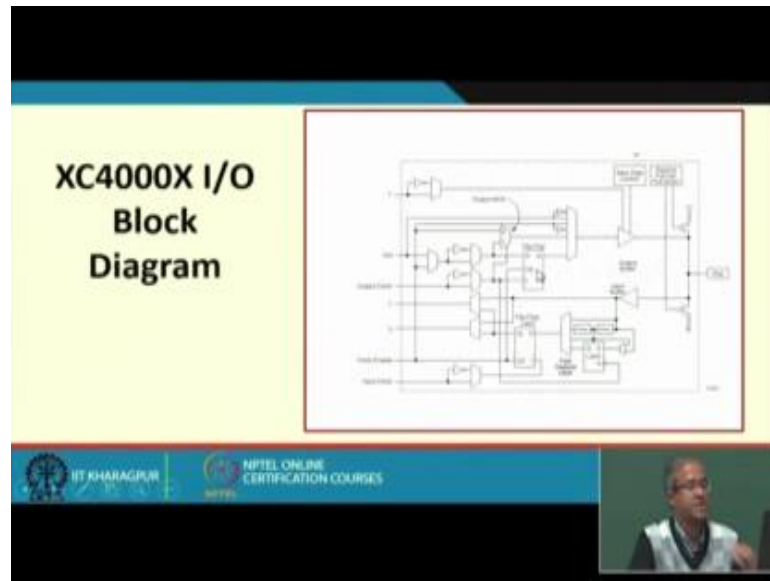
what kind of gates, this can be and gate, or gat, this can be exclusive or gate any complex kind of gates also right this is my original netlist which I want to implement.

Let say there can be alternatives. So, when I map this gates to say FPGA the LUT's, let say one possibility may be these three gates I can map into one LUT, why? Because you see in this set, the total number of inputs are 4, and there is a single output. So, I can map it Into LUT. Similarly look at this there are 4 inputs and 1 output. So, I can map it into 1 LUT, and the remaining is this 2 inputs and 1 output this also you can map into LUT. So, I need 3 LUT's with LUT delay of 2, because the output of these 2 LUT's are going as input to this LUT. So, delay will be 2 units.

But let say my mapping is slightly different, let say I again map these three into 1 LUT; let say I map these 3 into 1 LUT, this is also for inputs one output and let us say I map these 2 as a third LUT, and this as a forth LUT. So obviously, this will not be a very good mapping as compared to this. So, here as you can see I need 4 LUT's delay will be 3, because this first LUT is generating output which is fed to the second LUT, which is generating an output which is fed to the third LUT. So, delay is also more. So, as you can see the mapping is entirely different given netlist represented by a graph, I have to partition this graph into a way in which every sub graph will have up to 4 inputs and 1 output and of course, the depth or the delay as to be minimized.

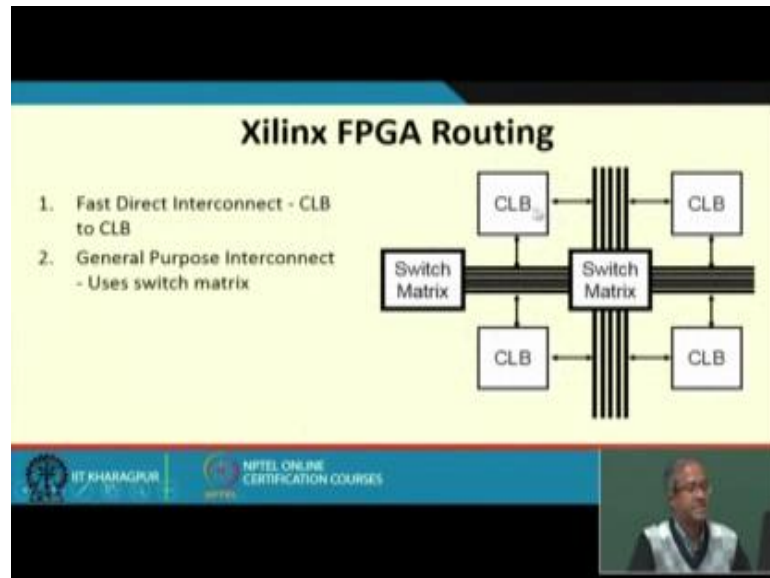
So, in the modern FPGA chips the number of lines of the LUT's scan they have increased. So, means you can have up to 5 by 6 also.

(Refer Slide Time: 33:31)



So, now, talking about the I O block diagram, so I am not going to detail; so I O block diagram looks like this, this is the input pin there are lot of circuits with flip latched is some output semi latched there some flip flops here, there are price state controllers there are lot of things inside which are programmable.

(Refer Slide Time: 33:52)

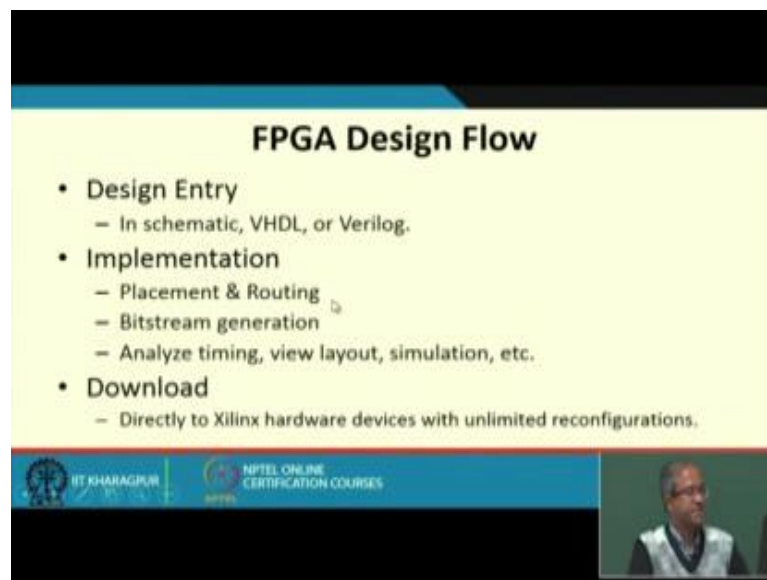


Then for FPGA routing again this CLB's are configuration logic block, they have some interconnection lines in between and there some switch matrices which are

programmable at the junctions. So, again from externally I can program the switch matrices, some of this columns can be connected to some of the rows.

So, by this some of the CLB pins can be connected to some other CLB pin. So, I can make some interconnections like this; and this switch matrix are typically implemented using again s ram by loading some bits in a memory I can switch on or switch off the connections, but there is another technology called anti fuse by passing a high current I can either disconnect a line or connect a line connect a connection. So, some FPGA manufactures also use anti fuse technology.

(Refer Slide Time: 34:50)



The image shows a slide titled "FPGA Design Flow" with a yellow background. The slide lists three main steps: Design Entry, Implementation, and Download. Design Entry includes schematic, VHDL, or Verilog. Implementation includes Placement & Routing, Bitstream generation, and Analyze timing, view layout, simulation, etc. Download includes Directly to Xilinx hardware devices with unlimited reconfigurations. The slide also features logos for IIT Kharagpur and NPTEL Online Certification Courses, and a small video inset of a speaker in the bottom right corner.

FPGA Design Flow

- Design Entry
 - In schematic, VHDL, or Verilog.
- Implementation
 - Placement & Routing
 - Bitstream generation
 - Analyze timing, view layout, simulation, etc.
- Download
 - Directly to Xilinx hardware devices with unlimited reconfigurations.

So, talking about the FPGA design flow, the software that is provided with the FPGA development boards they allow us to design entry using either schematic VHDL or Verilog, the entire placement and routing is carried out automatically, the bit stream to program all the blocks that is also generated automatically and that can be downloaded on the board. So, you can also analyze number of things I mean simulation and number of reconfiguration for many FPGA systems are usually unlimited you can do it many number of times.

So, FPGA provides them with a big advantage in terms of the configurability and ease of use. So, the next lecture we shall be looking at some other design styles, which have all (Refer Time: 35:42).

Thank you.