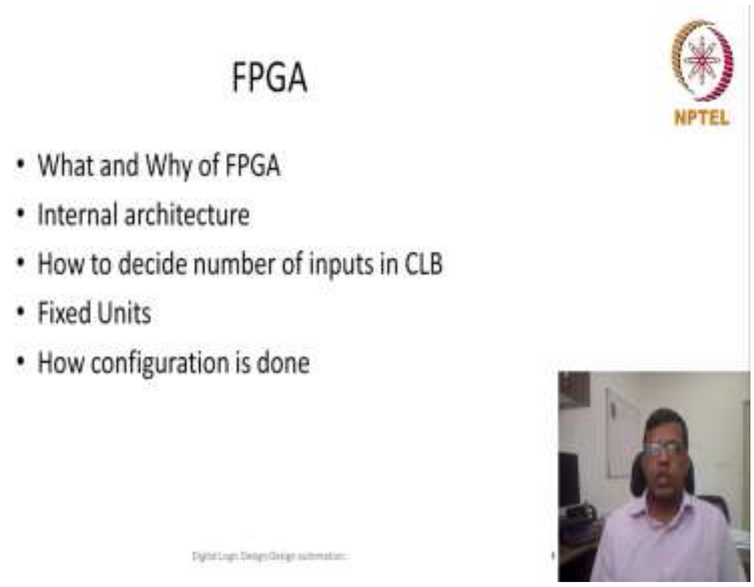
Digital System Design
Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology Ropar
Lecture 65
FPGA

Hello everybody, today we are going to start the last module of this course. And in this module we are going to touch upon a couple of important points which are related to design automation internal architecture of FPGA, how they are used and also some of the pointers for advanced topics like how bigger designs could be designed and what would be the utilities or what would be the future directions for this particular course. So, in today's lecture we are going to focus on FPGA.

(Refer Slide Time: 1:02)



FPGA

- What and Why of FPGA
- Internal architecture
- How to decide number of inputs in CLB
- Fixed Units
- How configuration is done

So, the question which we are going to address is why what is FPGA, why it is used and what is the internal architecture, what are the various design tradeoff if we would like to design an FPGA. And also how configuration is done how, so basically about different tradeoffs like fixed unit versus programmable units, fixed interconnection versus programmable interconnection.

(Refer Slide Time: 1:33)

FPGA

- **F**ield **P**rogrammable **G**ate **A**rray
- Fine grain reconfigurable/programmable logic
- Programmable hardware
  - Programmable logic: Configurable Logic block
  - Programmable interconnect

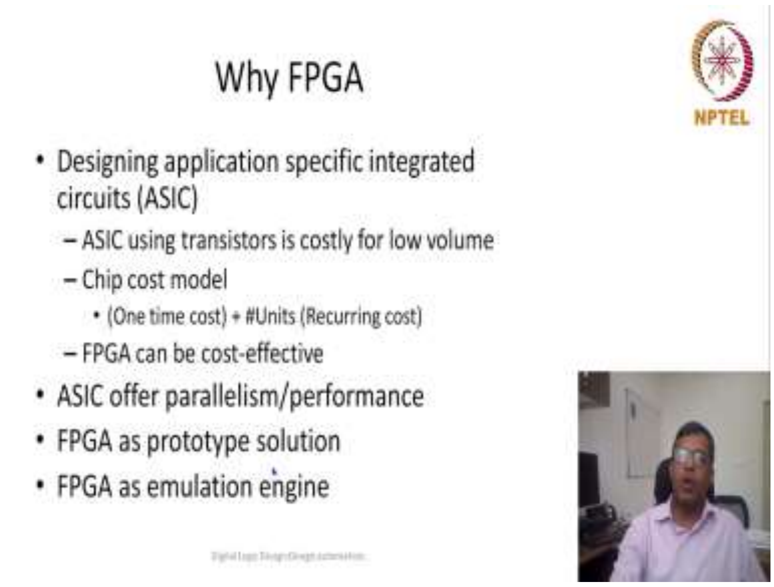
So, the first question we would like to understand is what is FPGA. FPGA stands for Field Programmable Gate Array. Field is usually an electric field and programmable means, this is a gate array which can be programmed using electric field or using electricity. So, gate array which could be programmed or basically array of logic which can be programmed as any other logic is field programmable gate array or FPGA.

Now, this was an early definition of FPGA like the acronym FPGA is used when it was initially designed as a gate array, but slowly these gate arrays has become quite a sophisticated programmable circuits, but still we call them FPGA. In general, we can call these FPGAs are fine grain reconfigurable or fine grain programmable logic. So, this is also in opposite to coarse grain programmable logic or coarse grained reconfigurable logic.

So, fine grain because each of the unit, basic unit is very small and it could be very small also with respect to number of logic it can implement, number of gates it can implement. So, here most of the time the smallest unit which can be programmed is a 5 input gate or 4 input gate or a 3 input gate. And that is why it is called fine grain reconfigurable logic or fine grain programmable logic.

Now, since it is programmable, so it should have two components programmable logic as well as programmable interconnects. So, both of these things we have discussed in brief in one or more of the previous modules, but let us for the sake of completeness, let us revise it here also to understand what is programmable logic and what is programmable interconnect. Here in this lecture, we are using configurable logic block also as a replacement or basically to mean the same thing as programmable logic it is configurable logic block or CLB is being used as a programmable logic in this lecture.

(Refer Slide Time: 4:10)



Why FPGA

- Designing application specific integrated circuits (ASIC)
  - ASIC using transistors is costly for low volume
  - Chip cost model
    - (One time cost) + #Units (Recurring cost)
  - FPGA can be cost-effective
- ASIC offer parallelism/performance
- FPGA as prototype solution
- FPGA as emulation engine

So, what is a programmable logic that we have seen earlier that, yeah, so before answering this programmable thing, let us also understand why this FPGAs are used. So, FPGAs form one of the implementation method or implementation way of implementation to implement ASIC. What is ASIC? Application Specific Integrated circuit. Now, we have certain applications, for example, in our last module last couple of lectures, we have taken an application as a traffic light controller or bubble sort or GST computer.

So, any of such application, which could be algorithmic in nature or control based application, so these application we would like to implement a specific integrated circuit or one hardware for such an application. So, if we would like to implement specific hardware, application specific hardware for that particular application, we call it ASIC. Now, let us say we would like to design an ASIC, then and the finally, we as we discussed during the courses or during the course, that finally it will boil down to implement using gates and gates would be implemented using transistors.

So, this standard procedure of implementing digital logic using gates in transistors could be quite an costly option. If number of units we would like to manufacture is low or we call it low volume also. Basically number of units are less than it could be quite costly. Why? Because generally, this chip manufacturing or hardware manufacturing can be modeled as non-recurring cost into, sorry plus, non-recurring cost means a one-time cost, a non-recurring cost plus number of units into recurring costs.

So, if number of recurring costs, sorry, number of units have less or the volume is less, then this one-time cost being very high cost, could be, this could make this particular chip or this particular hardware a very costly hardware. So, to reduce the hardware cost, the number of units has to be increased.

So, you would ask this question now that what we see these days that every time a new hardware is there, new mobile phones are there, it, the cost is decreasing day by day, but here we are saying it looks counter intuitive that ASIC using transistors or basically if we are designing a hardware digital hardware, then it could be a costly affair.

Since, it would be costly only if the volumes are less, because this one-time cost or design designing the mass or setting up the design that that particular cost is very, very high or very, very large. So, it runs into millions and 100s of millions or sometimes 1000s of million, but the factor which can reduce the overall cost is the number of units, the number of units are also in 100, 1000s or a million, then the cost of per transistor is very, very less and it is reducing since years.

So, with every technology, the number of, the costs of transistor get reduced, but the overall chip cost can be low only the volume is less. So, what if we have low volume and we still would like to have an ASIC solution, then we can have FPGA as a cost effective solution. So, FPGA will have, will be a cost effective solution because it is already manufactured chip and we can use this chip to, this chip is programmable, so that any ASIC can be configured on the same chip.

So, that is the most like that is the value proposition for FPGA that for it can be used to program any application. And because they are manufactured in large volumes, so the cost of one FPGA chip could be much lesser than the cost of an ASIC. So, now, why, there is still one more question that why ASIC has to be designed and implemented? So, usually the applications like any algorithmic application like you said sorting, searching or we usually write a C program and these C programs or high level language programs are run on a on a processor, processor is also a hardware.

So, that means these applications can use if they can run on a processor hardware called processor, then what is the requirement of designing our own hardware for one particular application? So, this itself is a is a big question to answer. So, the reason we would like to have a specific hardware for one particular application is that application requires very high performance, because processor hardware is generic. So, the whenever any application would run on that generic hardware, then it will be slow in nature.

So, if we would like to have very high performance or basically we would like to execute the same task in a very short time, then we would like to have application specific integrated circuit for these applications. So, one popular example in these days is these machine learning applications. So, the machine learning applications if we try to run on laptop processor even for a simple inference of image it may take a couple of seconds.

So, on the other hand, if we try to design a specific hardware for the same application, it could be done in a fraction of milliseconds. So, that is why this particular thing would be required wherever high performance is required or some specific objective like low power is required, then they are usually targeted or we would like to design a ASIC for a particular application.

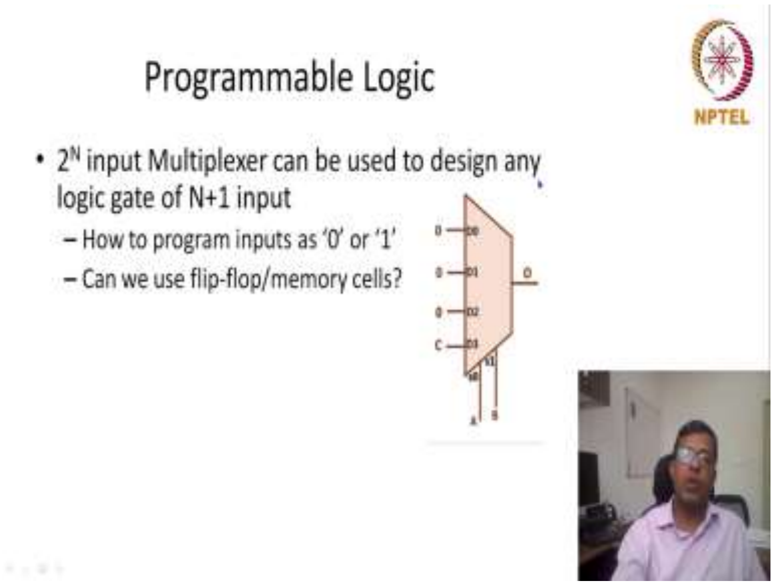
Now, so, again we are saying that, because, if we would like to design an ASIC and these ASICs are in low volume, so the requirement is less, so then we can use FPGA as a cost effective solution. Now, because of this particular requirement FPGAs are also used for prototyping solution. Prototyping means that we would like to first see whether our idea works or not.

So, whenever you would like to come up with a new hardware design, so rather than going for our direct transistor based design, we many people would like to go for an FPGA solution, so that they at least can prove to let us say funding agency that yes this idea works and that way because the volume is less, so you would like to have only probably a couple of units to show that this idea works.

And then similarly, if you would like to launch a product which has, which you would like to just see whether that particular product would be able to survive its purpose or not, then FPGAs could be used. Similarly, these FPGAs are also sometimes used as a emulation engine, emulation engine means that you would like to, you finally would like to design an ASIC, but FPGA can serve as a emulation engine, so that we can quickly simulate, quickly emulate that or that final design and then we can use this FPGA for debugging or for performance checks etcetera.

So, these are the various use cases why FPGAs are used. So, you will see, you will observe that for a simple logic FPGA would require probably 8x to 10, 10 times more area it would be slower also, but because they could be cost effective solution that is why FPGAs are predominantly used in industry. Primly they are called as prototyping engine because they can serve as the requirement to prototype a particular design.

(Refer Slide Time: 13:36)



Programmable Logic

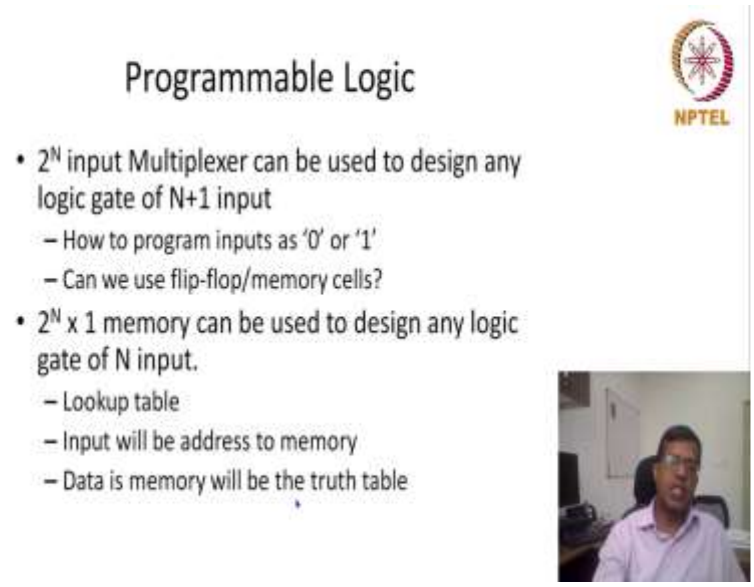
- 2^N input Multiplexer can be used to design any logic gate of N+1 input
  - How to program inputs as '0' or '1'
  - Can we use flip-flop/memory cells?

Now, after understanding what why FPGA would be used, let us see that what is programmable logic and programmable interconnect. So, these things we have already discussed in previous lectures. So, let us quickly glance through it that we have seen that 2 is to power N input multiplexer can be used to design any logic gate with N plus 1 inputs. So, for example, this is my 3 to, 4 is to 1 mask and my input is A, B and C.

So, you can see that this particular design where 4 inputs this is 0, this is 0, this is 0 and the third input is bounded to C. So, this design essentially works as, you can see this particular design works as a three input AND gate. So, similarly, you can see that any for, so these inputs could be like 0 and 1 and sometimes we can have the this also as a third input.

So, but when we are trying to design these we are trying to use these multiplexers in program, as a programmable logic, then the combination which we use is that the input to all these multiplexer inputs are only 0 and 1, the input signals are not used in, as input to this mask. So, if you would like to use these marks as a flip flop or a memory cell then it would be difficult.

(Refer Slide Time: 15:19)



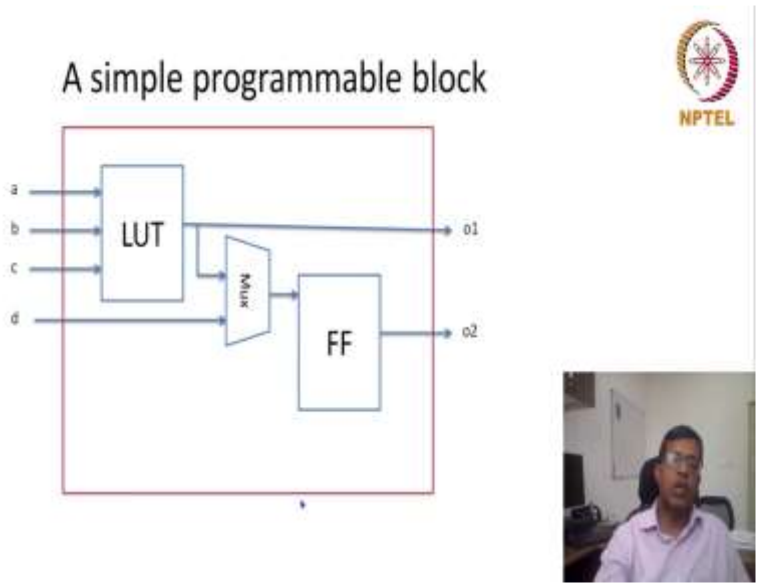
Programmable Logic

- 2^N input Multiplexer can be used to design any logic gate of N+1 input
  - How to program inputs as '0' or '1'
  - Can we use flip-flop/memory cells?
- 2^N x 1 memory can be used to design any logic gate of N input.
  - Lookup table
  - Input will be address to memory
  - Data is memory will be the truth table

So, what is typically done that we will have a specific flip flop or a memory cell assigned to this. So, finally, how this 0 and 1 would be given, so 2 is to power N into 1 memory can be used to give the N input, to give the, to make this configurable. So, this is also called lookup table, this memory of 2 is to power N into 1 is also called lookup table.

Now, the address to this memory is essentially the input to our, our logic. So, let us say here the example which we had taken A, B and C. So, A and B would be the input and the memory content 0 0 0 1 etcetera, would be the content of the lookup table. So, now the data of these memories essentially the truth table of that particular function.

(Refer Slide Time: 16:24)



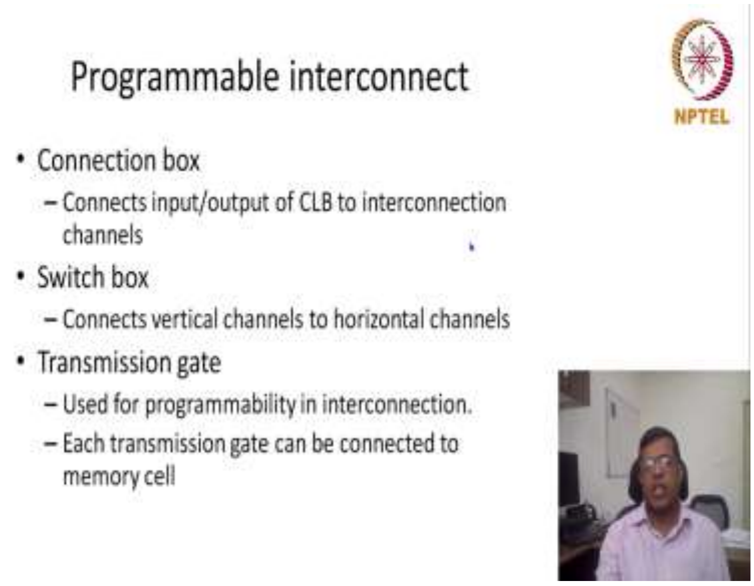
A simple programmable block

So, now because sometimes we would like to have output which is flip flop based another time we can, we will require output which is non-flip flop based. So, typical architecture of programmable block is you will have an LUT lookup table and this LUT essentially means that we would here the input to LUTs are 3. So, that means internally there would be 8 into 1 memory.

So, there are 8 memory cells and all these 8 memory cells are the lookup table, sorry, the truth table for these 3 variable function and so with if we have this 8 into 1 memory, we can program this LUT as any combinational logic with three inputs. So, there are two outputs to this programmable block; one as the unlatched output, the other one would be the latched output or output with a with a flip flop. Now, because here this is given as a mask input.

Now, there could be, this could be used as a simple mask, this if we gave another input d. So, this d input can be used as a flip flop or it could be this flip flop can take input from the output of the LUT. So, this way you we can, this particular programmable logic can either implement any three input combinational logic or it can implement one d flip flop or it can implement both one d flip flop as well as one 3 input non-latched output. So, this way one simple programmable block could be implemented as a different type of logic.

(Refer Slide Time: 18:27)



Programmable interconnect

- Connection box
  - Connects input/output of CLB to interconnection channels
- Switch box
  - Connects vertical channels to horizontal channels
- Transmission gate
  - Used for programmability in interconnection.
  - Each transmission gate can be connected to memory cell

Now, the second part is programmable interconnect. Now the programmable interconnect are usually divided. So, this this terminology is usually, this terminology is borrowed from Xilinx, but other programmable devices, other programmable FPGAs will have a similar terminology. So, here we are trying to, in Xilinx they have two different types of interconnect; one is called connection box another is called switchbox.

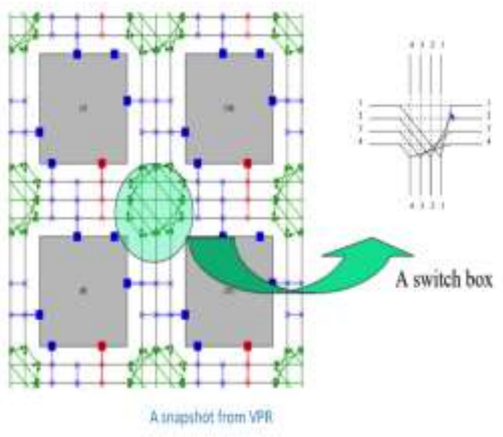
So, the connection box are simpler one which takes which connects input output of CLB's to interconnect channels, while switch box connects vertical channels and horizontal channels. So, how this thing will become programmable, how interconnect would become programmable, it with the help of switch box and connection box and how do we say that this particular wire should be connected to the another wire?

So, for that the, there could be different implementation, one of the popular implementation so one implementation is fused gates. So, where you fused the wires, so that if that wire is fused then it will get connected, one wire will get connected to the other way. The other popular option is transmission gate. So, transmission gates can be used for programmability of interconnect. So if the gate input of the transmission gate is 1, then the two wires will get connected, if the gate input is 0, these two wires will not get connected.

So, the gate would be connected to some memory cells so that now using the that memory cell, this transmission gate would be defining that which particular wire would be connecting to which another wire. So let us try to understand with a with a diagram.

(Refer Slide Time: 20:18)

Interconnection view



So, let us say this, the gray logic is a logic tile or a configuration logic block. And these blue ones, blue squares are actually input and output ports. Now here we have taken a design where input output could be on all the sides of my biologic tile or CLBs. Now, we see there is a two requirement things, so whether let us say to connect different wires, so there are some horizontal wires and there are some vertical wires.

So, if I would like to connect to my input output to these horizontal or vertical wires, I would still require a certain sort of connection. So, these blue ones are the connection boxes, where input output is connected to either horizontal wire or vertical. So, these crosses means that we are in this connection has to be made. So, for example, this input, whether we would like to connect to a vertical channel 1, vertical channel 2 or vertical channel 3, so that would be our programmable interconnect.

Now, this particular connection is simpler in nature, we are calling it connection box. So, where input would be connected to some of the channels. So, on the other hand, a switch box would be connecting more number of wires, there would be some horizontal wires, there would be some vertical wires, it would like to connect to these horizontal lines as well as vertical wire.

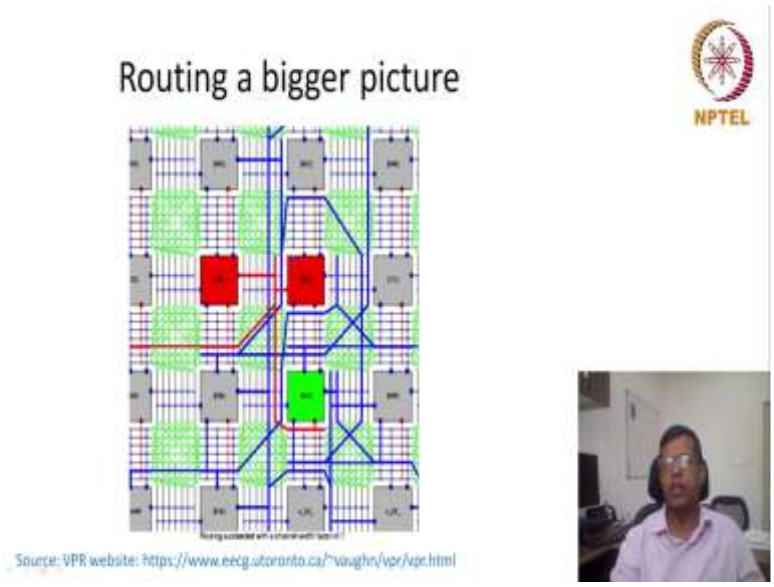
So, the idea here is that this one of the input let us say this input should be able to connect to any of the 3 plus 3 plus 3 any of the 12 inputs 3, sorry, 9 inputs, this input should be able to connect to any of the 9 inputs. Similarly, all the inputs are able to connect to all other 9 inputs. Now, one thing to notice here that if we are using switch boxes, it does not matter whether input is connected to output or output is connected to input.

So, for example, this is a input connection and the red ones are the output connection. So, this input connection can also be connected to the output connection or some of the other wires. So, which means that when we use, when you use transmission gate, they are essentially, they essentially become wired. So, the concept of direction is not there. So, it is not that the voltage can only travel from this point to this point.

So, reverse is also true, if this is connected, so this can also give signal to this. And similarly, basically here connection could be both ways. So, this diagram has been taken from a tool called VPR Versatile Place and Route. So, this is a is open source tool, which has been designed to study or to explore how input output connections could be done in an FPGA. So, a typical switch box is connected like this, for example, this wire could be connected to wire number 1 of this, wire number 1 of this.

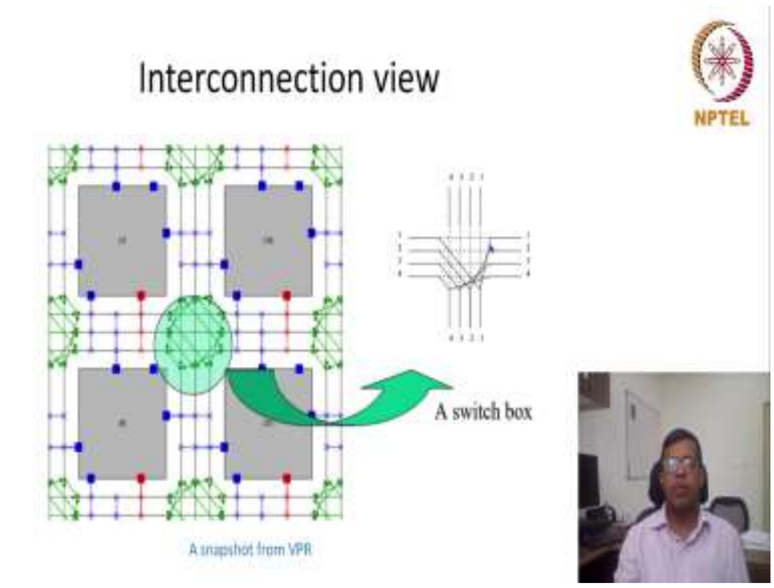
So similarly, all the connections can be made. So all the possibilities are there that each wire can be connected to any other device. So, you can see the switch box could be quite sophisticated in nature and there are still, there is some more tradeoffs to this.

(Refer Slide Time: 24:16)



Routing a bigger picture

Source: VPR website: https://www.eecg.utoronto.ca/~vaughn/vpr/vpr.html



Interconnection view

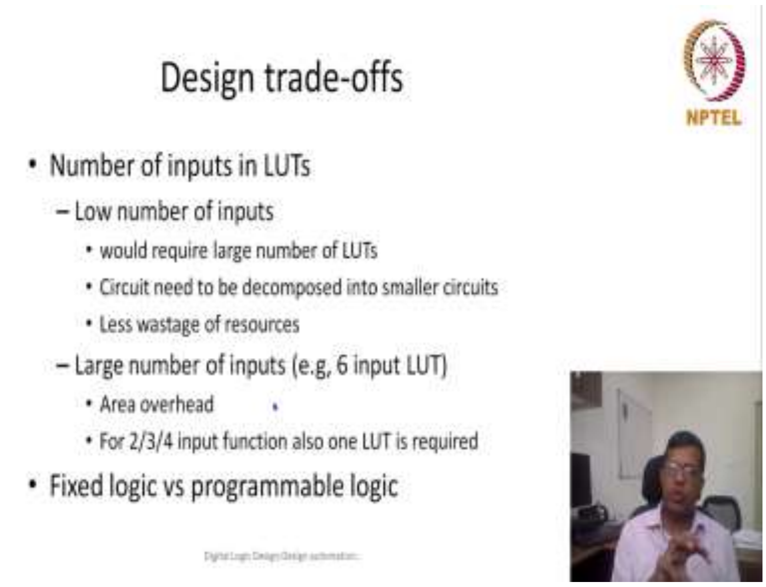
A snapshot from VPR

So let us say, yeah, this gives more bigger picture. Now here, the number of channels are also quite large. So here are 6, 7, channels 7 channels are there. 1 2 3 3 plus 3, 7, so seven horizontal channels, are there 7 vertical channels out there and the switch box is connecting all these horizontal channels to vertical channels.

So yeah, so more specifically, here you see that rather than connecting all of them to all of them, it is connecting channel number 1 to channel number 1 here. Yeah, similarly, the same thing we are seeing here also. So, let us say we would like to connect some logic from one place to other place. So, let us say this red connection is first connecting it to, this using the switch box it is connecting to one of the vertical wire and this vertical wire is connecting using the switch box to one of the horizontal wire.

And because this wire can also connect to this another vertical wire, so this way, this input this output is going to input of this particular LUT or this particular logic tile or CLB whatever you can call. So, this because of this configurable connections, any of the logic tile or CLB can connect to any other logic tile of CLB, sometimes we will feel that the number of channels, number of wires which are required could be huge or could be, the number of channels required or number of wires, number of input and output in connection switch boxes could be quite large, but that is the requirement for programmability.

(Refer Slide Time: 26:16)



Design trade-offs

- Number of inputs in LUTs
  - Low number of inputs
    - would require large number of LUTs
    - Circuit need to be decomposed into smaller circuits
    - Less wastage of resources
  - Large number of inputs (e.g, 6 input LUT)
    - Area overhead
    - For 2/3/4 input function also one LUT is required
- Fixed logic vs programmable logic

Now, if we would like to design our own FPGA or whenever somebody would like to design an FPGA, there are various issues which he has to think over. The first thing he will, the first design question would be that how many number of inputs in LUT should be there. So, should it be low in numbers, so for example, should it be 3 input LUT, 3 input LUT means that inside there would be 8 memory cells and or it should be 4 input LUT or 5 input LUT.

So, you see if the number of inputs in each LUT is less, let us say 3 or 4. So, what we would require is we would require a large number of these LUTs. So, but the issue or the difficult part is that any large circuit need to be decomposed into smaller circuits. So, let us say if we have to design we have only 4 input LUTs and we would like to design 4 is to 1 mask. So, 4 is to 1 mask would require 6 inputs.

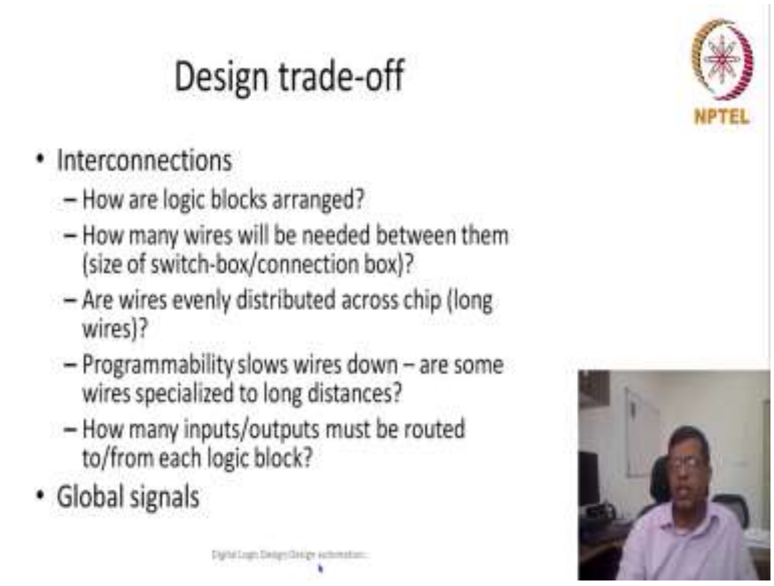
So, then we will have to decompose this 4, 6 inputs into a logic which would require all the 4 inputs the number of LUTs required would be quite large, but the resources would be a heavily utilized. So, on the other hand, if the number of inputs are large, for example, let us say 6 input LUT or 7 input LUTs, what would happen in that case? Any function which require only 2 inputs or 3 inputs, most of the memory content would not be required, many of the inputs will not be required. So that way it would be sheer wastage of hardware resources.

So, this is one thing, the other thing is that whenever we are designing using FPGA or we see closely the implementation of these configurable logic blocks or logic tiles, then we see the for implementing even a very simple gate like AND gate or an OR gate you require a complete logic type or a complete configurable logic block. So, this logic block, how can we improve the utilization the improvement of utilization can be done if we can have some fixed logic.

So, rather than using a programmable logic, if we use some of the logic as a fixed logic, then it could be quite efficient, but then it loses its own objective that hardware has to be programmable. Still to meet some performance goals, we would have, we would try to have at least some logic which is quite popular.

So, for example, if we are designing an ASIC and we try to see different application where this FPGA could be used. We see that addition is one circuit which is quite popular, which would be used in in almost all the applications. So why not to design an fixed logic for fixed logic for some components, which is quite popular or is to be used in in most of the circuits. So there, so that means there could be some programmable part, there could be some fixed logic. What logic should be used as a fixed logic is always a question or is a tradeoff.

(Refer Slide Time: 30:01)



Design trade-off

- Interconnections
  - How are logic blocks arranged?
  - How many wires will be needed between them (size of switch-box/connection box)?
  - Are wires evenly distributed across chip (long wires)?
  - Programmability slows wires down – are some wires specialized to long distances?
  - How many inputs/outputs must be routed to/from each logic block?
- Global signals

So, similarly for interconnections, the choices are even more difficult even more varied. So, for example, how logic blocks should be arranged, so whether they should be arranged in a

2D fashion or whether they should be arranged in a single line, how many logic block in each of the row or a column and how many wires should be there. So, we in some one of the previous slide we talked about horizontal channels and what vertical wires; horizontal wires and vertical wires, how many horizontal wires should be there?

How many vertical wires should be there? That also determine what is the size of switch box or connection box we would like to have. So, you we see that some of the wires are our local connections. So, basically you would like to connect the output of one of the logic block to the input or the other logic block while some of the connections are large fan of the nature, that one of the output would like to go to multiple of the inputs of different logic block, while some different kinds of wires would be global in nature or semi-global in nature where one other circuit is going to all different kind of logic blocks.

And sometime the connections between two blocks could be far. So, basically they are, the wire which you would like to connect from one particular unit to other unit, the other units are spaced quite far, some of the units are spaced quite near. So, these are all different kinds of issues so which also give a question whether the wire should be distributed uniformly or non-uniformly?

So, that or could it be possible that some of the wires could be long wires or some of the wires could be short wires or all these connections, all these become the design questions that how should we design these internal wires interconnections? Now programmability is also an interesting question here.

So, whenever two wires get connected using a past transistor or this past logic, then it always slow down the buyer. So, basically whenever you are giving 1 and using a past transistor logic, you are trying to pass this on to the other end. Because this past transistor, internal structure of the past transistor, it would the wire is much slower then the non-programmable wire or a direct wire.

So, that is why we will always think that if there are some connections which are more frequent, can we directly connect them or can we make certain connection as non-programmable, we provide that yes this non-programmability would always give the, give benefit in terms of latency or wire delays. So, this programmability and choosing between which wire should be programmable, which wire should be non-programmable is a good question.

Now, some of the global logic for example, a clock and reset has to be travel across all the logic blocks. So, there may be some non-programmable wires for those purposes also. So, how specially input and output need to be organized. So, in our previous for example diagrams we have seen that on all the four ends of logic tile or logic block there were inputs and outputs should it be there all the way or how they should be connected? These are all design questions for interconnection.

So, yeah global signals we have already talked about that there are some global signals where special wiring would be required, which will usually be non-programmable in nature. Now, after all these designs then we can, so that also gives us the clue that there would be different FPGAs for different kind of applications or different requirement, based on the requirement different interconnects could be designed, based on the requirement different kind of logic block could be designed.