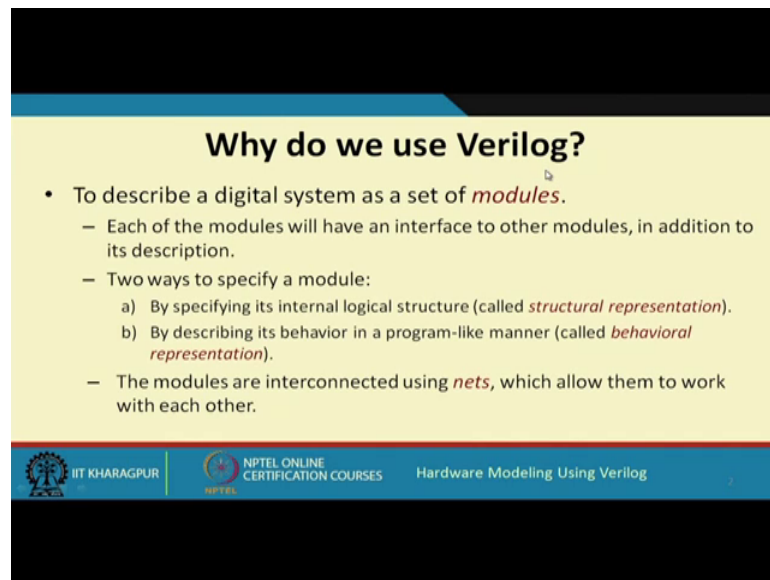


Hardware Modeling using Verilog
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 03
Getting Started With Verilog

Welcome back. In this third lecture, we shall be getting the first feel about verilog: how to use it, and how to simulate designs using verilog. As you can see that the title of this lecture is getting started with Verilog. So, let us see.

(Refer Slide Time: 00:46)



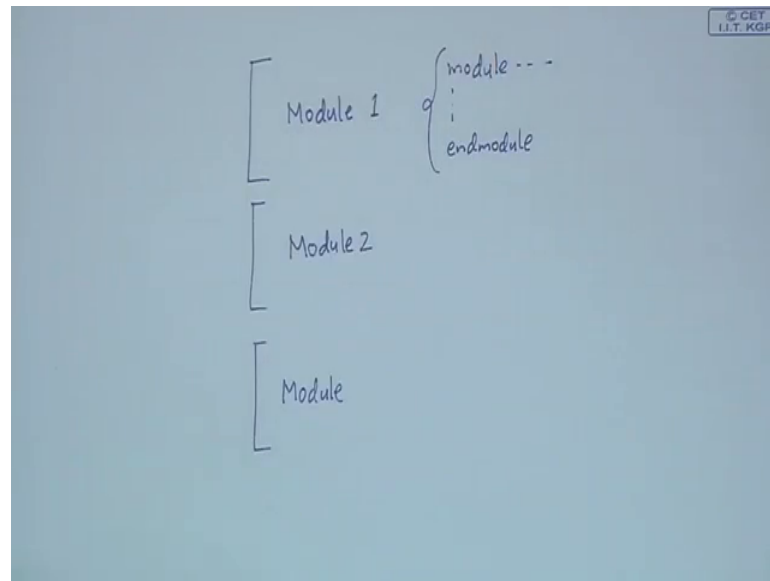
Why do we use Verilog?

- To describe a digital system as a set of *modules*.
 - Each of the modules will have an interface to other modules, in addition to its description.
 - Two ways to specify a module:
 - a) By specifying its internal logical structure (called *structural representation*).
 - b) By describing its behavior in a program-like manner (called *behavioral representation*).
 - The modules are interconnected using *nets*, which allow them to work with each other.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog

So, in our last lecture we have already seen some example; verilog programs. So, I have just mentioned, the concept of verilog modules. So, any description of a digital system that we are trying to create description, this has to be in terms of a set of modules. So, whenever you write a design description in a verilog; so, it will be looking as follows.

(Refer Slide Time: 01:22)



There will be several modules; let us say there are three modules. Now, each of these modules will be starting with the module keyword and it will end with the end module. The same thing will be for the other modules also. So, this all the modules taken together will define your complete specification of the hardware; whatever you are trying to arrive at.

So, if you want to design some system or a circuit in terms of multiple modules; like in the earlier example, we saw in the last class where we talked about a 4-bit triple carry adder. So, we saw we had multiple modules; carry modules, some module; we use those modules to create a full adder module add. Then we used four copies of the add module to create a ripple carry adder module and so, on. So, you can instantiate a module from other modules. So, there is no concept of calling one module from another; calling means instantiation in terms of hardware. So, if you call for example, from module 1; if you call module 2, there will be a copy of module 2, which will be inserted in module 1.

Because you have to remember that ultimately we are designing some hardware. So, whenever invoking something one module is invoking another module; that is something called instantiation. It means create a copy, if I caller invoke two times there will be two copies created fine. So, these modules when you describe them; they will as you see, they have some interfaces means in terms of the parameters. So, the way we specify the parameter values; the variables they will specify how these modules will be

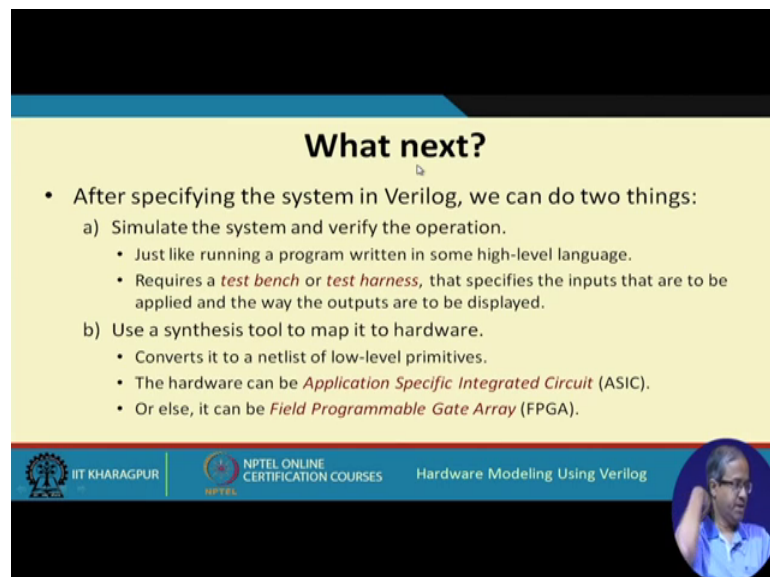
interconnected. Now, we have also seen that we can specify a module in two different ways; you saw some examples in the last lectures.

So, we can either specify its internal logical structure; which is called structural representation. This we had already seen for the some and carry modules of the full adder and also the full adder module and also the 4-bit triple carry adder module. Those are all examples of structural description, where we had some modules; we also specified how the modules are interconnected; this is the so called structure representation.

Secondly, we can also specify a module by describing its behaviour. So, you recall the earlier example we took; where the carry and some expressed in the form of Boolean expression; that was a behavioural representation. So, when we talk about structural representation; we also talk about the inter connections of the modules, they are inter connected by something called nets. Nets are just like wires the output of one module is connected to the input of another module.

So, we shall see; later what are the different kinds of nets, different kinds of wires that are there in verilog. We shall see this in detail later all right. So, what next?


(Refer Slide Time: 05:26)



What next?

- After specifying the system in Verilog, we can do two things:
 - a) Simulate the system and verify the operation.
 - Just like running a program written in some high-level language.
 - Requires a *test bench* or *test harness*, that specifies the inputs that are to be applied and the way the outputs are to be displayed.
 - b) Use a synthesis tool to map it to hardware.
 - Converts it to a netlist of low-level primitives.
 - The hardware can be *Application Specific Integrated Circuit* (ASIC).
 - Or else, it can be *Field Programmable Gate Array* (FPGA).

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog



See we have specified the system in verilog; let us say. So, we have already specified our design in verilog; now we can do one of two things, first is we can simulate the verilog code and we can verify it whether it is working correctly or not. This is just like running

a program which you do in a high level language like C. So, whenever we write a program; we immediately run the program, give some input data and check whether the outcome is coming correctly or not; this is how you verify.

So, here simulation means exactly a very similar thing. So, we simulate or we say that you run this verilog description; with some input data that I have given and you tell me what the output is coming. So, this simulation is just like running a program as I had said; written in some high level language. This requires us to specify the input and how we do it? We do it by writing something called test bench or test harness.

So, what is the test bench? Test bench is another verilog module; which will actually be generating the inputs that will be applied to the module that I want to test. So, here we will see; that how we can write this test benches. Test bench or test harness; specifies the inputs that have to be applied to my system which I have written in verilog and I want to also see the outputs. This is the one thing; I may want to do after I write this specification.

Now, you see as part of this course most of you will be doing exactly this. So here you are learning how to write or means how to specify some designs in verilog. You will be specifying some design, you will be simulating and you will be trying to verify whether your design is working correctly or not; this is through simulation. But as an alternative what you may do? You may use some kind of a synthesis tool to map it to a piece of hardware.

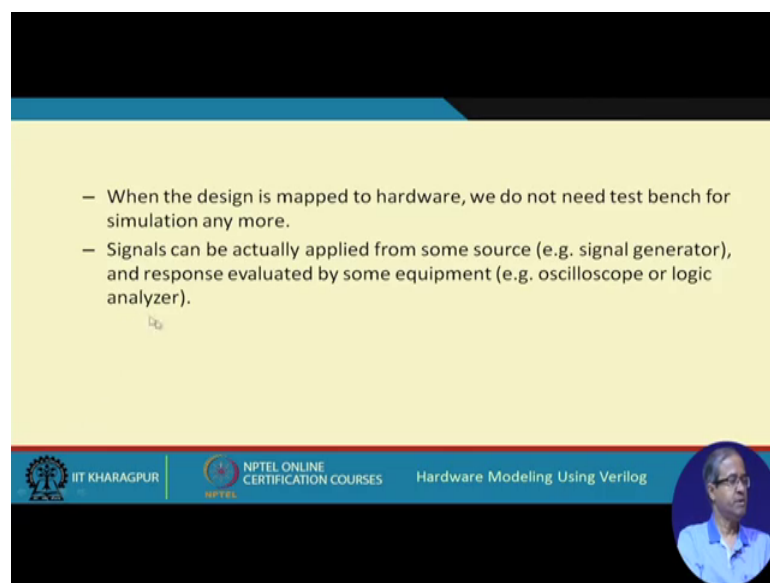
So, what is synthesis tool? The synthesis tool will be specifying your; it will take your specification which you have written in verilog and it will be translating into ultimate hardware design; that will be your final hardware, that what is called synthesis. So, the synthesis tool will convert your specification into a netlist of low level primitives. Now, when you say low level primitives; again there are two alternatives available with you.

Some of you may say that well I want to design VLSI chip; which means I want to design something called as an Application Specific Integrated Circuit or ASIC. So, here I will have to go through all the steps of the VLSI design cycles as I mentioned earlier. After doing that; I will have to send the layout specification that I have finally, there are some particular formats for the GDS 2; I will have to send it to a fabrication facility and they will fabricate the chip and they will send the chip back to me.

This is one way you can utilize your design; what you want to do with it or the other one is that most of us do the second one; that your target hardware can be Field Programmable Gate Array or FPGA. So, what is the Field Programmable Gate Array? and FPGA is something which is programmable.

So, you can use it in your laboratory. So, you can create a design in verilog; you can use a synthesis tool, it will be translating into some form which you can download on the FPGA chip by giving a command. Suppose you have designed a multiplier, you download that in the FPGA chip; your FPGA chip becomes a multiplier; that is your hardware. So, FPGA is nothing, but a programmable hardware you can make it behave in whatever the way you want by downloading an appropriate programming data on it. So, these are the two ways you can utilize this synthesis tool.

(Refer Slide Time: 10:27)




– When the design is mapped to hardware, we do not need test bench for simulation any more.

– Signals can be actually applied from some source (e.g. signal generator), and response evaluated by some equipment (e.g. oscilloscope or logic analyzer).

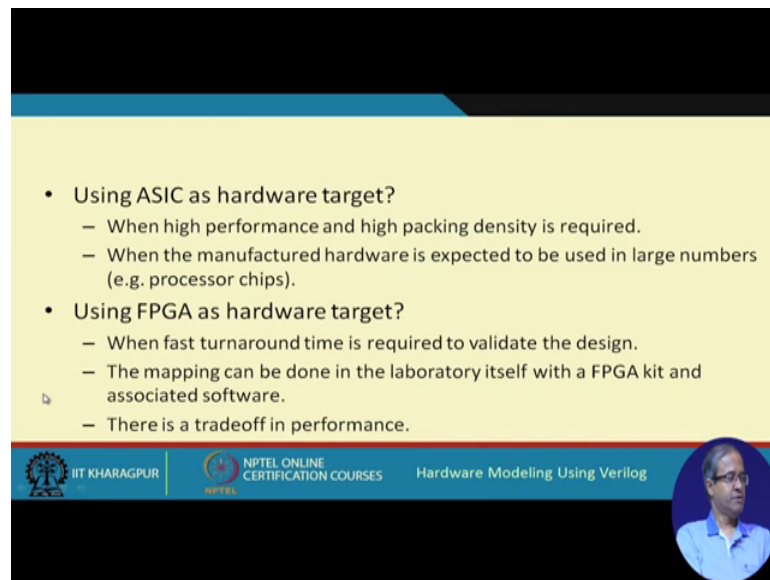
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog

NPTEL



So, when the design is mapped to the hardware; we do not need the test benches for simulation anymore. Because you already have the hardware; so, we can actually apply the voltage signals and see the output. We do not need any test bench and simulation to do that; we have the actual hardware in this. So, here the signals can be actually applied from let us say some signal generator and the outputs maybe actually observed on let us say an oscilloscope or some logic analyzer.

(Refer Slide Time: 11:02)



The slide contains the following text:

- Using ASIC as hardware target?
 - When high performance and high packing density is required.
 - When the manufactured hardware is expected to be used in large numbers (e.g. processor chips).
- Using FPGA as hardware target?
 - When fast turnaround time is required to validate the design.
 - The mapping can be done in the laboratory itself with a FPGA kit and associated software.
 - There is a tradeoff in performance.

At the bottom of the slide, there are logos for IIT KHARAGPUR, NPTEL ONLINE CERTIFICATION COURSES, and the text 'Hardware Modeling Using Verilog'. A small circular portrait of a man is visible in the bottom right corner of the slide area.

So, when do we use this ASIC as a hardware target? Now, we use ASIC as a hardware target; when we need high performance and high packing density. You see this application specific interior circuit is very often optimized. So, whenever you use this ASIC, you are generating a hardware; which will be very much optimized in the sense that its area will be less, its speed will be higher and it will possibly meeting all your delay and power related constraints.

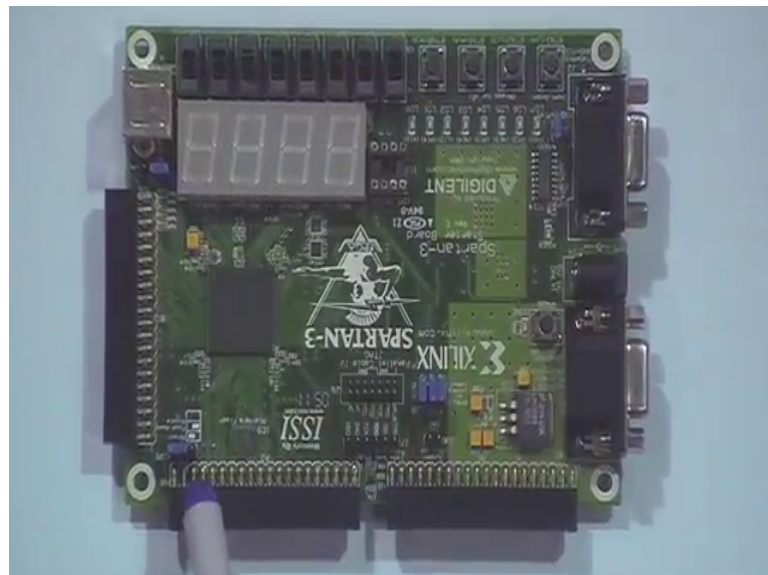
But there is a price to be paid; when you design an ASIC that two things, it is a very long driven processes. It will take you months to design the chip send it for fabrication getting the chip back and secondly, it is very expensive. So, the fabrication facility will be charging you a lot; many many lakhs of rupees; let us say to get a chip fabricated, so it is very expensive.

So, there has to be some kind of cost effectiveness in that; so you can possibly afford to use that only when manufactured hardware is expected to be used in large numbers. So, one classical example is a processor chips; so, you think of the Intel processors; they manufacture the chips in millions all around the world. So, they come up with the design which is best possible in terms of its speed and performance. So, they will have to go for in ASIC designs, but when I created a design for my class, my laboratory; I cannot afford to spend so, much of money. So, I will be doing it in my lab and I will be using a Field Programming Gate Array or FPGA.

So, use FPGA as the target when fast turnaround time is required; fast turnaround time is required means once you written a code in verilog; it can be a few minutes; you do a synthesis, you download it on FPGA; your hardware is ready. So, this you can do in your lab itself, so what you need? You need something called FPGA board or FPGA kit and some associated software.

Trade off in performance means well because you are using a programmable hardware, your speed may not be as good as ASIC; means your area will also be bigger. Well means it will not be that optimized; well here I am showing you one example of an FPGA kit.

(Refer Slide Time: 14:02)



This is an FPGA kit you see; this is a board where there is small chip you can see here; this is your FPGA, well I can just keep it here you can see it from the top. This is your FPGA; this square thing you can see, this is your FPGA and you see there are other components in this; there are some switches, you can apply some inputs to this switches; there are some seven segment displays, there are some small LED's there are some push button switches.

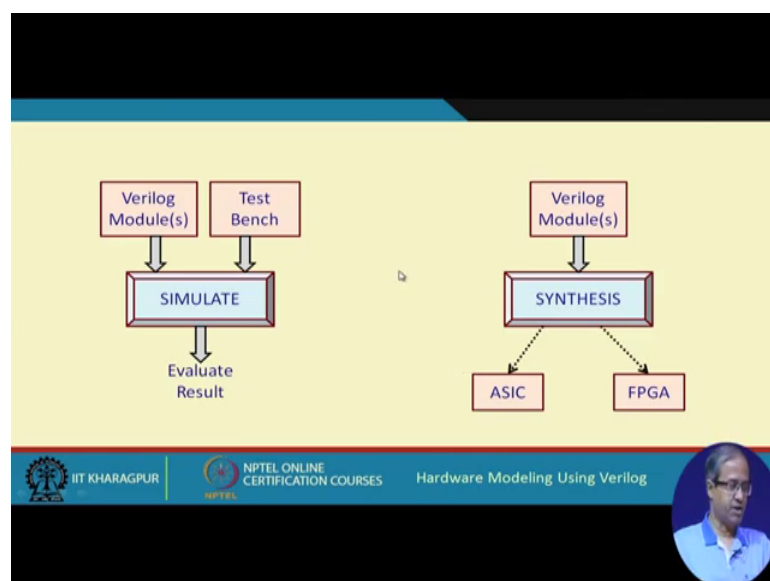
So, when you download some design on this FPGA. So, what will happen is; that means, mean you can actually apply some inputs through this switches; through this push buttons and the output you can observe on the LED's or the seven segment displays and see whether functionally it is correct or not. But if it is a more complex design, you possibly need to connect some external device like a logic analyzer or an oscilloscope.

So, you see that on this board; there are some other components are also there. Like here there are some serial ports, there are two serial ports out here on this side there some h connectors; you can connect some h connector through which you can connect to a processor. and here there is some connectors here; you can see, these connectors are connected to a PC or a laptop or a desktop; from there the FPGA data will be downloaded; through this pins.

So, this is a very cheap cost effective device; this device for example, costs only a few thousands of rupees. So, in your lab you can have a device like this and if you have a device like this; you can quickly come up with the design, put it on a FPGA and test it and see whether it is running or working correctly or not. Now, most of you or many of you possibly have already used FPGA or will be using FPGA at some point in time.\

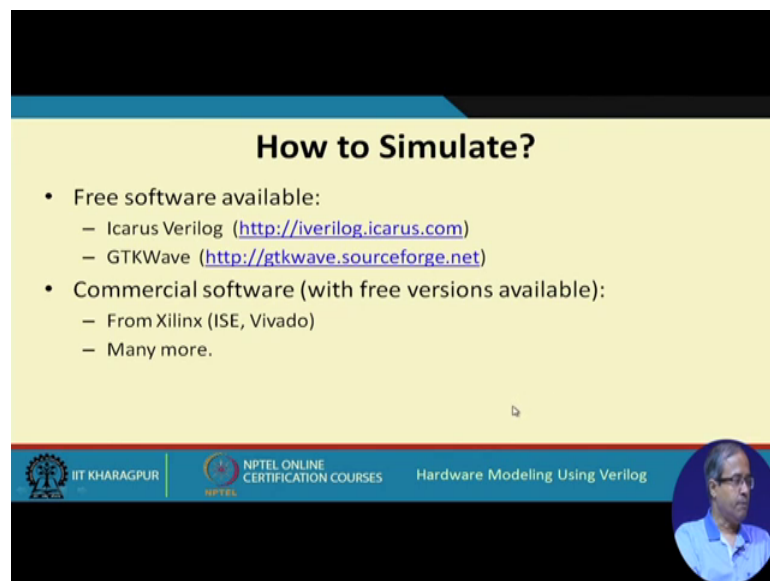
But, as I told you as part of this course; I shall not be teaching you how to use FPGA's or how to convert my verilog description through synthesis and burn into FPGAs. Rather in this course; I shall be concentrating on explaining how this verilog language is and what are the different ways to create a design or specify a design; based on certain constraints and requirements and we shall be mostly telling you and urging you to use simulation to verify that; whatever you have learnt you have written a code whether that code is working correctly or not. Because simulation is much easier and simpler to run and check, I shall show you an example also.

(Refer Slide Time: 17:24)



So, pictorially these are the two alternatives; so, if you decide to go for simulation, then you have the verilog modules and you have a test bench module, you combine them, simulate and we evaluate the result; whether the result is coming as per your expectation or not. and if your target is synthesis; then you only need verilog modules, you do not need test bench. and depending on your requirement you either map it to an ASIC or you can map it to an FPGA; as I said.

(Refer Slide Time: 18:08)



The slide is titled "How to Simulate?" and is presented on a yellow background. It lists two categories of software:

- Free software available:
 - Icarus Verilog (<http://iverilog.icarus.com>)
 - GTKWave (<http://gtkwave.sourceforge.net>)
- Commercial software (with free versions available):
 - From Xilinx (ISE, Vivado)
 - Many more.

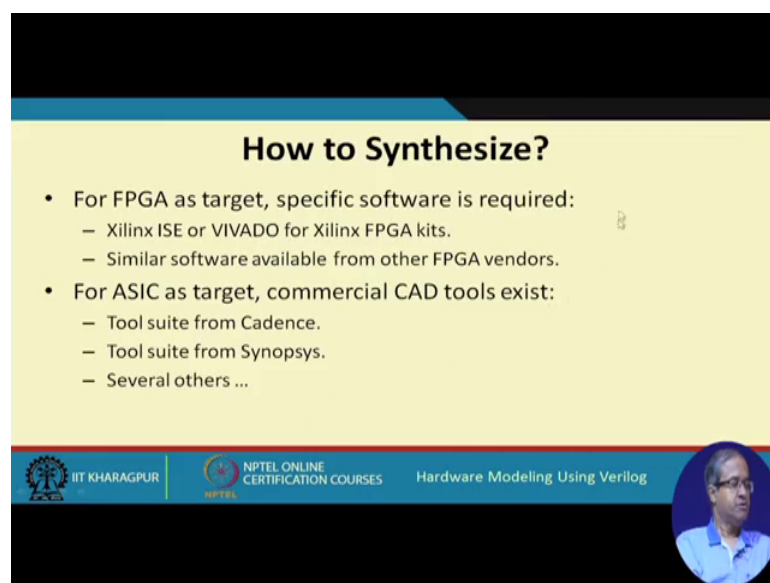
The slide footer includes the IIT Kharagpur logo, NPTEL Online Certification Courses logo, and the text "Hardware Modeling Using Verilog". A small circular inset image of a man in a blue shirt is visible in the bottom right corner of the slide.

Now, the question is how to simulate? Now, here I am showing you two links; there will be tutorials which will be uploaded also on these site, you can basically look at them; that how to download and install them. See here, I strongly recommend you to download this Icarus verilog or Iverilog in short; which is available on this link iverilog dot icarus dot com. This is a free software and this verilog simulator; it runs on almost all platforms; windows, various versions of Linux, Mac everywhere.

So, it is more or less platform independent you can run it anywhere you want and means after simulation if you want to see the results in a graphical way; is in wave form; there is another tool you can also download this is also free; GTK wave, you can download it from this site. So, again GTK wave is available on all platforms, but if you want to use some commercial software's; many of the commercial software's have the free version with limited capabilities available.

Like the FPGA manufactures Xilinx is one of the leading FPGA manufactures. They have the software tools called ISE, Vivado and a few others and there are many others vendors also. So, using any of these tools, you can carry out or do simulation, but in the examples that I shall be showing and illustrating. So, I shall be illustrating with iverilog only; so, if you want to reproduce my examples. So, it is always good, it is always recommended that you install iverilog on your machine, on your computer and you can also run the verilog codes there and so means, you can verify whatever I am saying whether it is matching with your simulation or not alright.



(Refer Slide Time: 20:20)



How to Synthesize?

- For FPGA as target, specific software is required:
 - Xilinx ISE or VIVADO for Xilinx FPGA kits.
 - Similar software available from other FPGA vendors.
- For ASIC as target, commercial CAD tools exist:
 - Tool suite from Cadence.
 - Tool suite from Synopsys.
 - Several others ...

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog



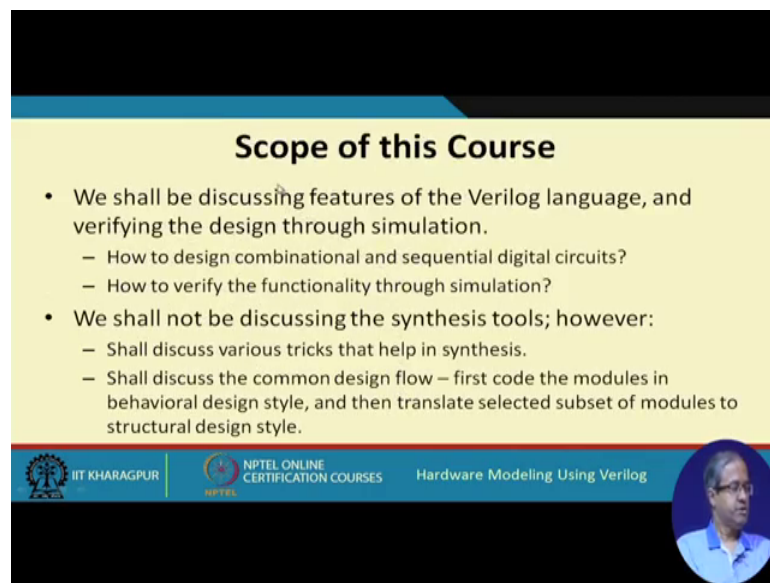
So, if you want to do synthesize; synthesize again if you use FPGA as target. So, you need specific software; like the FPGA board that I have shown that was an board which is manufactured by Xilinx; the FPGA chip was a Xilinx Spartan 3; FPGA chip. So, for Xilinx; you need to use either a Xilinx ISE software or a Xilinx Vivado software for those kits. But for other FPGA vendors Altera, Actel; there many several others; there are similar software available from the other vendors also.

So, for FPGA there is one constraint; if you use a particular type of a FPGA kit, you have to use this software that is provided by that particular means FPGA chip manufacturer; for Xilinx, you have to use only the Xilinx software. But when you want to go for ASIC, you can use some commercial CAD tools; which are more generic in that sense.

That means, you are not stuck with the particular rule; there are tools from Cadence, there are tools for Synopsis and several other companies are also there. These are commercial tools; much more professional, much more elaborate and of course, much more expensive as well.

So, you cannot afford to buy for personal use. So, only for large institutions and laboratories; you can possibly afford to buy a license.

(Refer Slide Time: 22:07)



The slide is titled "Scope of this Course" and is presented on a yellow background with a blue header and footer. The main content is a list of bullet points. The footer contains logos for IIT Kharagpur, NPTEL Online Certification Courses, and the course title "Hardware Modeling Using Verilog". A small circular portrait of a man is visible in the bottom right corner of the slide.

Scope of this Course

- We shall be discussing features of the Verilog language, and verifying the design through simulation.
 - How to design combinational and sequential digital circuits?
 - How to verify the functionality through simulation?
- We shall not be discussing the synthesis tools; however:
 - Shall discuss various tricks that help in synthesis.
 - Shall discuss the common design flow – first code the modules in behavioral design style, and then translate selected subset of modules to structural design style.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog

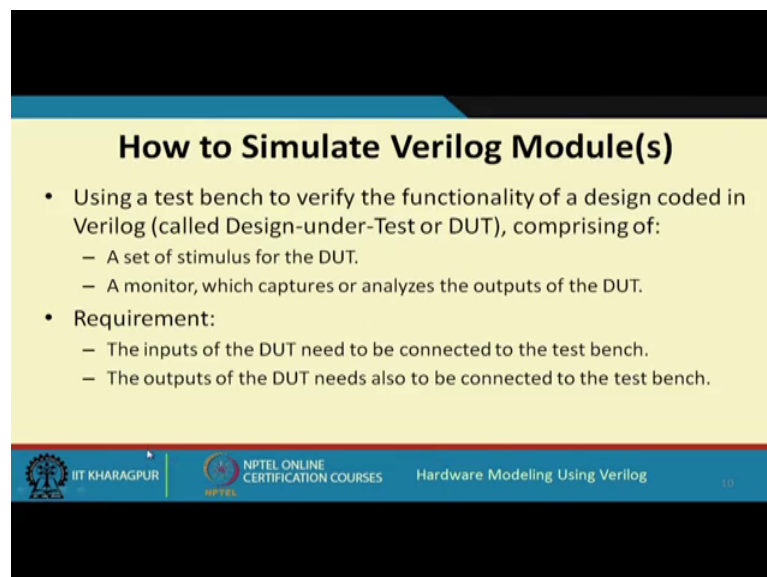
So, as I had said that; in this course, we shall be discussing various features of verilog. How to write verilog modules? So, how to design combinational and sequential modules, circuits? and we shall also tell you that how to verify functionality through simulation, but we shall not be discussing synthesis as part of this course; because it is slightly beyond this scope. But, we shall be discussing few things which can help you in the synthesis; for those of you, who are interested in actually carrying out synthesis; maybe now or maybe later.

So, we shall be discussing some tricks, some techniques in verilog coding; which will help you in generating; you can say good design, the final hardware that you are mapping to, that will be a more efficient hardware, more efficient design. There are various tricks so we shall be seeing some of them. and also we shall be looking at the typical design flow which is followed; because you see through the examples you may have got an impression that writing a behavioral code is easier, writing a structural code

is more complex. So, what is typically done is there for a complex design. First we code all the modules in behavioral fashion. Then one by one we place those behavioral specifications by structural specification. So, not all of them so, a good subset of the modules are finally, translated into structures specifications, but the remaining one say for example, the control you need the control module that is often left in the behavioral form only they will not be converted into structures ok.

So, there is a trade off that has to be decided by the designer. So, if I have 100 modules, how many of them I want to convert it into the structural fashion, because you see once I convert it to a structural fashion. So, as a designer, I will have much better control about this structure of my design and also the performance. So, I can guarantee that my design will be good and it will perform according to specifications. Now if I give everything to this synthesis tool, ultimately it is a translator program, very complex program. So, often it generates a circuit which is correct, but in terms of efficiency it is seldom it sometime happens in the, it is not so good. So, it is it always helps if the designer gives some inputs to the tools so that the tool will understand that what to do and how to do right fine.

(Refer Slide Time: 25:21)



How to Simulate Verilog Module(s)

- Using a test bench to verify the functionality of a design coded in Verilog (called Design-under-Test or DUT), comprising of:
 - A set of stimulus for the DUT.
 - A monitor, which captures or analyzes the outputs of the DUT.
- Requirement:
 - The inputs of the DUT need to be connected to the test bench.
 - The outputs of the DUT needs also to be connected to the test bench.

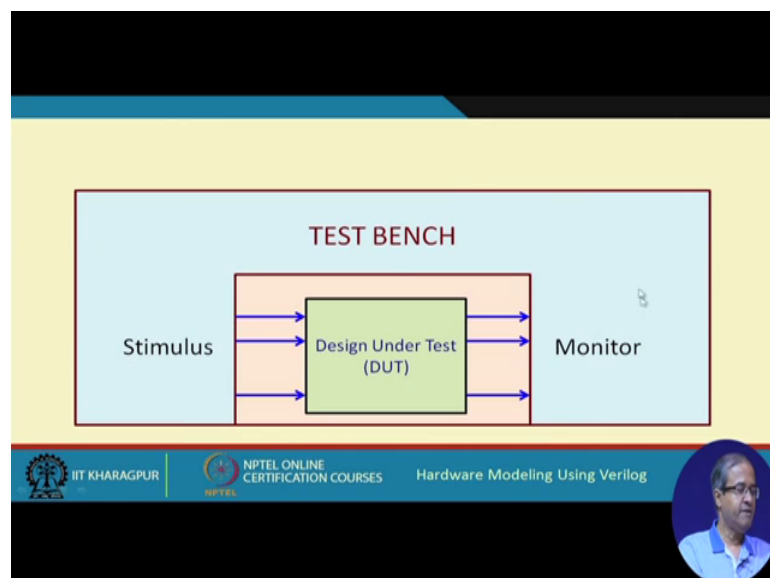
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog | 10

Now, the question is how to simulate verilog module? Let us just explain this with the help of there was small example. So, as at said for simulation we need a test bench. So, let us say we are using a test bench to verify the functionality of a design a module that module you are calling are referred to as Design-Under-Test or DUT. So, we have a

design under test, we want to simulate it and verify that this is working correctly or not. So, in order to do that what the test bench will be doing. So, it will first be: applying a set of inputs which are called stimulus. So, a set of stimulus will be applied to the DUT and the output of the DUT will be monitored. So, the test bench will also be capturing or analyzing the DUT outputs, right.

So, this is the requirement, the test bench has to need the input of the DUT. and also the outputs of the DUT need to be connected to the test, bench for doing this. So, pictorially I can visualize like this, as if this whole thing is the test bench.

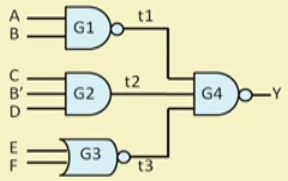
(Refer Slide Time: 26:39)



Now within the test bench, I have embedded my design under test. So, test bench has one part called stimulus; which is applying some inputs. There is another part called monitor, which is see observing and monitoring the outputs, right.


(Refer Slide Time: 27:05)

An Example



```
module example (A,B,C,D,E,F,Y);
  input A,B,C,D,E,F;
  output Y;
  wire t1, t2, t3, Y;
  nand #1 G1 (t1,A,B);
  and #2 G2 (t2,C,~B,D);
  nor #1 G3 (t3,E,F);
  nand #1 G4 (Y,t1,t2,t3);
endmodule
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog



Now, let us take an example. A small example, There is a small combination circuits consisting a four gates: 2 nand gate, 1 and gate and 1 nor gate. This is a simple structural description of this circuit using the primitive gates which are already available in the verilog library: nand, and, nor and nand.

So, all the signals A, B, C, D, E, F and Y these are specified. Out of the A, B, C, D, E, F are the inputs; Y is the output, and this intermediate lines these are given some names t 1, t 2, t 3 and this Y you can declare as Y may not is mandatory. Now just ignore this numbers for the timing this nand you see first nand, G 1 I am giving a name G 1, t 1, A, B; t 1 is the output A, B; G 2 and t 2, C this is not C not B, D C, B not D this is G 2; nor t 3, E, F; t 3 nor E, F; G 4 Y is the output G 4: t 1, t 2, t 3; and here for the sake of simulation purpose, I can specify the delay of the gates. Like I say that the delay of the nand gate is 1 time unit, nor is 1, nand is 1, but and is 2. Let us say I have defined like this.

So, this is the way to specify, this kind of a structural netlist. Now one thing; there are 2 nand gates right. So, instead of writing this nand twice, so here we can also combine the declaration I can write nand once, both are delay 1, and I can write G1 t 1, A, B comma G 4 this and only a semicolon after that. So, I can combine several gates of the same time type by writing it only once and combining them fine. So, this is the module we are written.

(Refer Slide Time: 29:18)

```
module testbench;
  reg A,B,C,D,E,F; wire Y;
  example DUT(A,B,C,D,E,F,Y);

  initial
  begin
    $monitor ($time," A=%b, B=%b, C=%b,
              D=%b, E=%b, F=%b, Y=%b",
              A,B,C,D,E,F,Y);
    #5 A=1; B=0; C=0; D=1; E=0; F=0;
    #5 A=0; B=0; C=1; D=1; E=0; F=0;
    #5 A=1; C=0;
    #5 F=1;
    #5 $finish;
  end
endmodule
```

```
example.v
module example
  (A,B,C,D,E,F,Y);
  wire t1, t2, t3, Y;
  nand #1 G1 (t1,A,B);
  and #2 G2 (t2,C,~B,D);
  nor #1 G3 (t3,E,F);
  nand #1 G4 (Y,t1,t2,t3);
endmodule
```

example-test.v

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog

Now, we are trying to write a test bench. What we are doing in the test bench? This is how the test bench will look like. This I am writing as the module test bench. This A, B, C, D, E since example this is my example right module example. So, I am instantiating this here A, B, C, D, E, F are the inputs and Y is the output right.

So, I have instantiated the example I am calling it DUT. So, A, B, C, D, E, F, Y all these are there. Now just the ignore this line for the timing I will come back. This is my test bench. When in test bench we often use the key word initial. Initial means there is a block of statements, this block will be executed only once begin end right. So, what does this block contain? Monitor dollar monitor, monitor is something like printf in C. So, here I am saying that you monitor the variables and you print certain things. what are the things to print? Dollar time means this simulation time comma within quote A equal to percentage B means bit or binary, B equal to C equal to just the syntax is very similar to C. C, D, E, F comma then the actual name of the variables.

So, we are printing these variables. Now what monitor means that whenever anyone of these variables changes, then you print. Monitor is an intelligent print statements. So, it will not print every time, it will print only when at least one of the variables change fine. So, then what I say then hash 5 means these are time. It says after time 5, apply these inputs. Then after again another time 5 apply these inputs. Then again time 5 A equal 1, C equal to it means the other inputs are not changing. Only you change A and you

change C, but B is still 0, D is still 1, E is 0, F is 0. And again after time 5: you set F equal to 1 and again our time dollar finish means you finish this simulation. This is a very simple test bench and this is the meaning. Let us say the name of the file is example test dot v, and the original description of the module that you wrote the name of example dot v.

So, this example was instantiated here right. This is how within a test bench, I can instantiate the module and apply the inputs. Now one thing, that inside the test bench initial block. So, whatever variables appear on the left hand side here A, B, C, D, E, F they have to be declared of type reg, reg is the register type variable that is, why we have declared these are type reg. and Y is the other variable, which is not on the left hand side as simply declared as wire; wire y right. Now this simulation results if we simulate this design, then you can actually see that when the gates are changing the delays. So, I leave it as an exercise for you that you can see.


(Refer Slide Time: 33:01)

• Simulation results:

```
0_ A=x, B=x, C=x, D=x, E=x, F=x, Y=x
5 A=1, B=0, C=0, D=1, E=0, F=0, Y=x
8 A=1, B=0, C=0, D=1, E=0, F=0, Y=1
10 A=0, B=0, C=1, D=1, E=0, F=0, Y=1
13 A=0, B=0, C=1, D=1, E=0, F=0, Y=0
15 A=1, B=0, C=0, D=1, E=0, F=0, Y=0
18 A=1, B=0, C=0, D=1, E=0, F=0, Y=1
20 A=1, B=0, C=0, D=1, E=0, F=1, Y=1
```

Command in iVerilog:
a) iverilog -o mysim example.v example-test.v
b) vvp mysim

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | Hardware Modeling Using Verilog



That your simulation result will come like this, at time 0 all the variables are undefined x means undefined. At time 5, you see at time 5 you say time 5 I am applying this: A 1, B 0, C 0, D 1, E 0, F 0. So, I have applied this, but still Y is undefined, see after time 3 your Y has changed; why because you see this circuit this is a delay of 2 plus 1 3 this and gate has a delay of 2 plus 1, after delay of 3 this Y will change right. After time 8 y will

change, similarly this is the time. So, here you can just check, similarly that the times will be printed and the values will change.

So, this will be your simulation output and now the question is given these two files example dot v and example test dot v. So, how do I get this output well its very simple if you just a second if you just install i verilog and if you just give a command like this iverilog minus o just like in C compilation you can give an output file name minus o this is the output file name mysim then all the verilog files example dot v example test dot v. Then you give another command v v p mysim. So, if you run the if you run this v v p this output will be obtained right very simple.

So, this you can verify. Now, here I have made a little modification in the test bench. I have just added these 2 lines: dump files and dump vares rest is same rest all are identical. Dump file means I am saying that well in the monitor is given it is printing all right, but I will also dump all these changes in a file called example dot vcd, v c d stands for value change dump and dump vars 0 dot 0 comma test bench means test bench is the name of this module. So, all the variables in this modules and all the variables that are included inside they will all be dumped right now just after this if you run it again that v v p. So, in this file example dot v c d will be created.

So, you just give a command like g t k wave example dot vcd if you install g t k wave also just give this. So, what you will get, you will get something like this. This is the window of g t k wave. So, in this left corner you will see the name of your test bench file DUT you can see all the name of the wire. What I have done? Is that I have just pulled this these values to this window one by one and this wave forms are immediately starting to appear. So, you can see this is the time on top 7 second, 7, 14, 21, 0. So, you can see that exactly the time at time 5 this is changing. A has become 1. So, at time 7, this y has become 1; this exactly how the simulation output is coming. you can see it in the form of time for timing diagram. So, you can see this visually in the form of a wave form.

So, I think with this we come to the end of this lecture. So, in this lecture we just saw, how we can actually write a verilog module, how we can write a verilog test bench, how we can simulate them, how we can see the output, and how also we can visualize the timing diagram. Now these experiments I shall not be showing on the time in my next

lectures. I expect that you will be installing verilog and g t k wave in your machines, and you will be actually running this codes yourself and seeing them working.

Thank you.