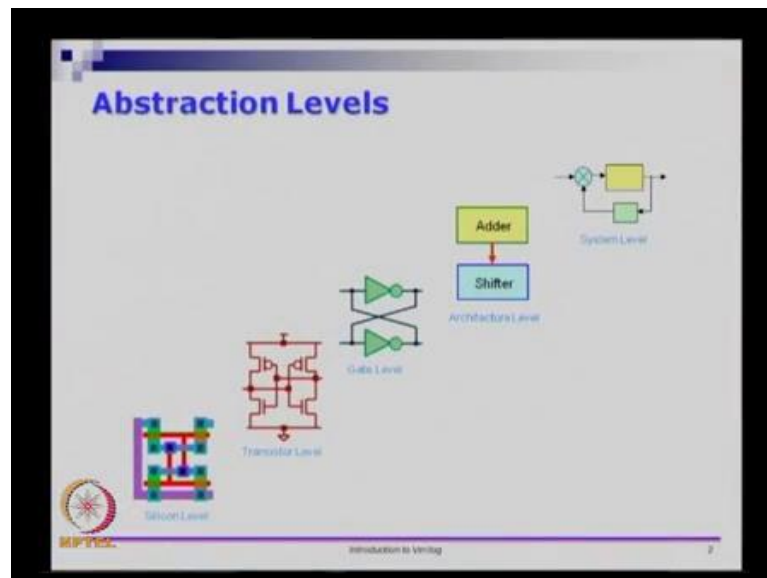


**Digital Circuits and Systems**  
**Prof. Shankar Balachandran**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Bombay**  
**And**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Chennai**

**Module – 06**  
**Introduction to Verilog**

We are in module 6, this brings us to the last module for week 1. In this module, I am going to give you a very quick introduction to verilog. So, when we design circuits we actually do it at different levels.

(Refer Slide Time: 00:30)

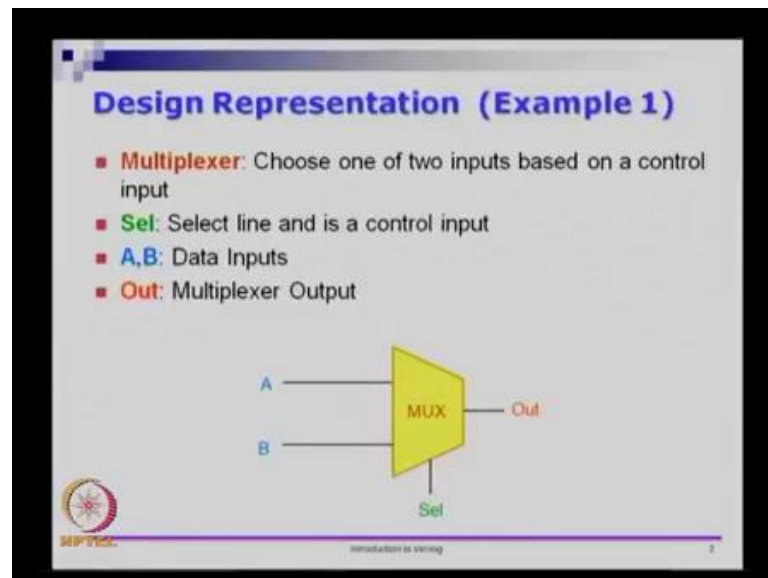


So, I already mentioned this in the first module, there we talked about different levels of abstraction. So, you can look at chip from the silicon level or from the transistor level. So, we know that these are things that you... So, this is usually taught in an advance course, when you are in a third year, you may get an elective. A transistor level design you might have done some of it, probably you will do some more of this if you are in the second year, you will probably learn some of this in the third year. We are looking at these levels right now.

So, these three are the levels at which we will be dealing with in this course. So, there are different levels of abstraction in which you can look at basic problems, you can go and look at it from the silicon level this is for chip designers, who are at the closest level to

silicon or the actual chip. So, this is usually done by circuit designers, this is done by logic designers or a system architects.

(Refer Slide Time: 01:30)



So, verilog is essentially a language which can be useful for doing that. So, to motivate that, I want to take a small example and show you how to do this in Verilog. So, for this we are going to use a digital circuit which is called a multiplexer. So, a multiplexer is the term that is used when for designing something in which you have multiple inputs and you have only one output.

So, imagine for example, telephone exchanges, so there are several telephone lines that connect through the exchange and what they do actually is when there are multiple connections that are connected, so it is not that you need that connection all the time. So, if you are making a phone call, you do not need this phone line to be allotted to you all the time. Instead what they do is, they do what is called multiplexing of the several inputs that are coming in, they will pick, which one to pass from input to the output.

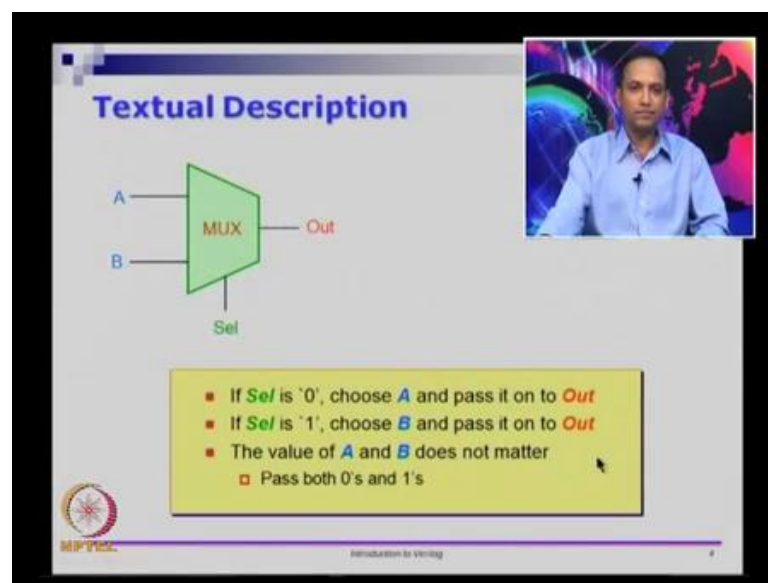
So, let us take this picture for example, so this is the symbolic notation for what is called a multiplexer and this multiplexer is a simple multiplexer, it has two inputs A and B it has one output. We want either A or B to be present at the output dependent on the value of the select line. So, if the select line will control whether A connects to output or B connects to output, so this is called a multiplexer.

So, at this point you do not know the logical description for the circuit, we just know that there are two inputs A and B and this is actually standard symbol for this combinational

block called multiplexer and based on input A and B and the value of select, it chooses one of these to be passed out. So, we will call this select line or Sel as a control input and the data inputs are A and B, so this is a symbolic representation of the circuit.

So, as an abstraction level this is a symbolic representation, so it is useful to see these pictures. Because, as humans we understand pictures slightly easily, but this is not something that a computer can work on, it is not something that you can manipulate as a symbol. By the way, this is the standard symbol for a multiplexer, just like for AND gate and OR gate it should be standard symbols, this is a standard symbols for a multiplexer.

(Refer Slide Time: 03:43)



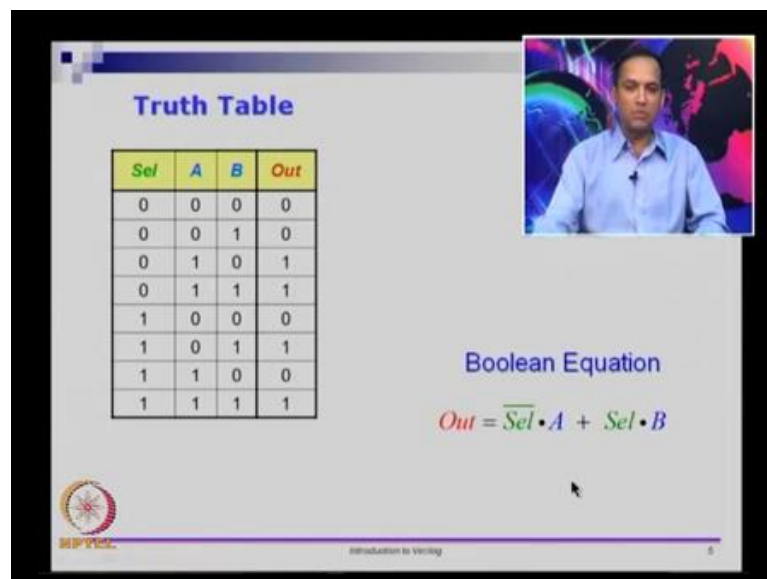
So, now I can write the description for this in English, we can say that if select is 0, so select is just a one line. So, it is a control input, it has it can take two values, either it can take 0 or it can take 1. If select is 0, then choose A and pass it to output, if select is 1 choose B and pass it to output. So, this is an English description of this whatever design problem I gave you.

So, I want one of A, B to be passed on to output at a time and this is an English description of them. So, at this point if you notice, the description says if select is 0 pass A to output, we do not care whether A is 0 or 1, we want to pass whatever is the current value of A we want to pass it to output. And similarly, whenever select is 1 it does not matter whether B is 0 or 1 we want that value to be passed to the output.

So, which means the multiplexer passes both 0s and 1s, so this is an English text description of the same thing. So, this is a schematic or a symbol description of the

problem that I gave you, this is an English description of the problem that I gave you.

(Refer Slide Time: 04:51)



**Truth Table**

| Sel | A | B | Out |
|-----|---|---|-----|
| 0   | 0 | 0 | 0   |
| 0   | 0 | 1 | 0   |
| 0   | 1 | 0 | 1   |
| 0   | 1 | 1 | 1   |
| 1   | 0 | 0 | 0   |
| 1   | 0 | 1 | 1   |
| 1   | 1 | 0 | 0   |
| 1   | 1 | 1 | 1   |

**Boolean Equation**

$$Out = \overline{Sel} \cdot A + Sel \cdot B$$

NPTEL

Introduction to Verilog

5

Now, we can go and think about this in a truth table format. So, let us go and see what are the different inputs we have, we have select as an input A as an input, B as an input. So, there are 3 inputs, if there are 3 inputs there are 8 combinations that are possible. So, we have those 8 combinations listed here and for these 8 combinations, let us see how we get the output. So, we said if select is 0, A should appear in the output.

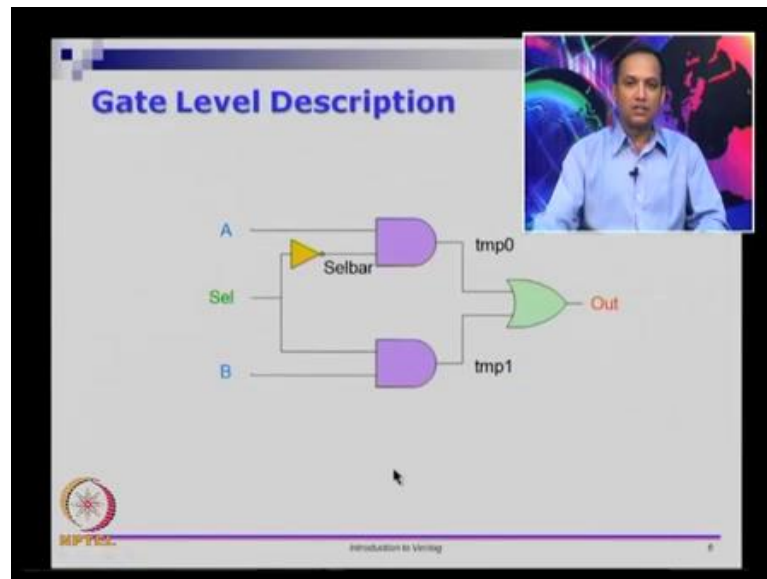
So, let us look at the first 4 rows, the first 4 rows has select equals 0 and if you notice this part of the column, this first 4 rows of this column or the copy of the first 4 rows of A. So, what that means is when select is 0, we have copied A to output, which is what we wanted. Similarly, if select is 1 we want a copy of B in the output, so you can notice that in the lower half of the truth table, select is 1 and if you notice this part of this column is copied on to output. So, we have two parts, the upper part is for select equals 0 and the lower part is for select equals 1.

So, this is the truth table way of describing this, we can write the same problem down as a truth table. So, we can write the symbol, you can write it in English or even if you write the truth table, there all the same. Finally, you can also go and write it as a Boolean equation. So, we said when select is 0 we want A and when select is 1 we want B, you can write it in Boolean expression like this.

So, Sel complement means, if Sel is 0, Sel complement is 1, so when select is 0 u and a when select is 1 and B. So, let us start with select Sel being 1, if Sel is 1 this term

becomes 0 and B is pass to the output. If Sel is 0 then this term is 0 and this part becomes 1, so we have 1 and A which is out. So, this is a Boolean expression that you can directly write, if select is 0 you can write it as select bar. So, if I asking this question if x is 0, then you can check x bar, if x bar is 1 then x is 0. So, it is a same thing here, if select is 0 pass on A, if select is 1 pass on B. So, this is a Boolean equation for the same design problem that I gave you.

(Refer Slide Time: 07:35)



So, this is the gate level description, so if you take the expression here and if you convert that to your circuit which is this is what you would have. So, select you send it through an inverter you get select bar, and it with A you get term 0, let us say I just given a name here call term 0, then you have select and it with B that gives you temp 1. So, temp 0 or temp 1 gives us the output.

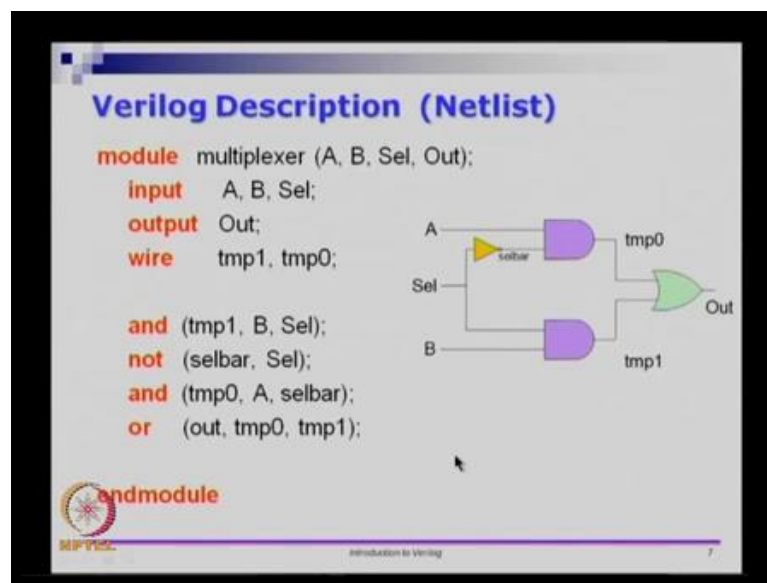
So, if you want to do this, so once you design it as circuit like this on paper, you should be able to go and do this using a breadboard also. So, in your labs probably you would of seen this already or so this online course does not have a lab associated with it directly. So, I cannot make you do physical, electrical connections and so on in this course, but probably you have this access in your colleges itself, I suggest that you go to the labs and try this out.

So, you pick that components for inverter, the AND gates and the OR gates and you can physically lay out in your breadboard or wherever. So, you will require wires to connect to these gates and you will need wires to connect from these gates to this one and so on.

So, if you have to physically do it you need this physical wires, so you may take this an external input, you want to connected to the input of gate B again to input of the gate, the output of these gates need wires to be connected to this and so on.

So, as a picture we can draw this picture easily, but to design this actual circuit that will work, we need the components and we need the wires. So, this line here is actually a wire, everywhere you see lines there are actually wires in the real world. So, this is another description for the same design.

(Refer Slide Time: 09:24)



Finally, I want to show you how to do this same thing in Verilog. So, if you look at it as a circuit, it takes three inputs A, B, and Sel and it has one output out. So, this is actually working Verilog code, this Verilog as a language you have working Verilog code. So, what you do in the language like Verilog is you specify, what are all the inputs and what are all the outputs.

So, in this case if you think of this as a black box circuit that is given to you, it would take 3 inputs and it will give 1 output. So, you will see physically 4 pins that are hanging out of this black box that is what you have there. You have 4 pins that are hanging out of the black box and we designate which are all inputs and which are all outputs. So, as a human being if you look at the picture, generally we associate left side to be the inputs and right side to be output and so on.

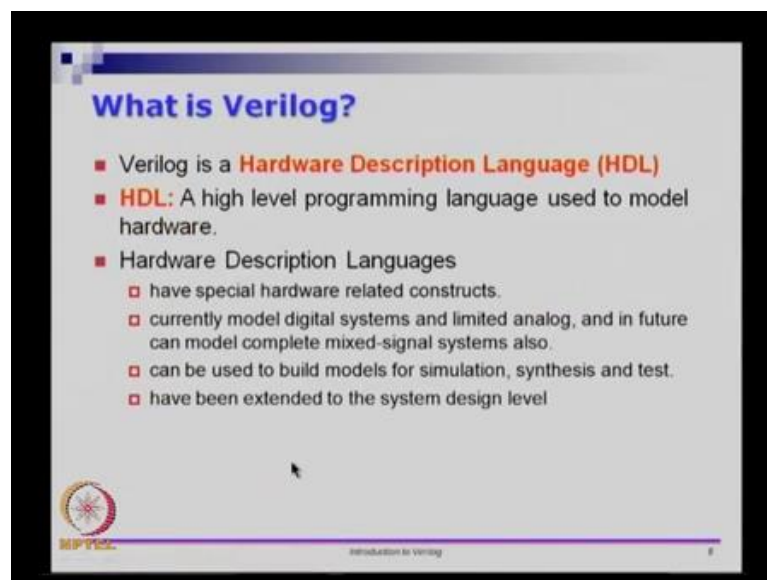
So, if you want right this for a computer to understand we need to explicitly say which are inputs and which are outputs that is what we have here. So, input A, B and select

output out, we also need to these 2 wires temporary wires, temp 1 and temp 0 and we need another wire called select bar. So, we need another wire call select bar here, now let see how to get each of this whole circuit represented.

So, we have B and select you and that together B and select you and it together, you will get the description for temp 1. So, AND B and select that gives you temp 1, so the meaning of this line we will get into the details later. So, this is the output and these two are the inputs for this line, so AND B and select to give temp 1, complement select to give select bar AND A and select bar to give temp 0 or temp 0 and temp 1 to give out, so that is the description of this whole thing.

So, what we have here is we have, so there should be another wire here called select bar, wires select bar must be there. Once, we have that this code essentially says how to connect the wires together, so if you do not have the circuit if I give you just the logical description, you should be able to come up with the circuit based on just the logical description. We will going to these details in will going to mush more details later about this, but I want to you to is just see how Verilog code looks like. So, this is a just a sample for how Verilog looks like.

(Refer Slide Time: 12:07)



So, what is Verilog? Veriolog is Hardware Description Language also called HDLs. So, HDL is a high level programming language and it is useful for modeling hardware. So, if you have been, if you are in 2nd year you probably have learnt one or two programming languages, you probably know C or C plus plus or java or something like that. So, these



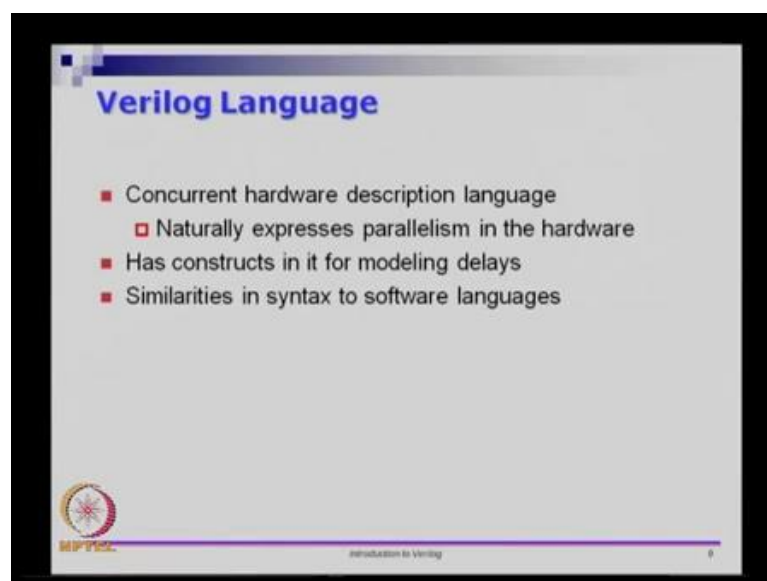
languages are not conducive for designing hardware. So, we need hardware description languages that understand what hardware designers want, not what software designers can do.

So, it is a high level language is use to model hardware and HDL language is have special hardware related constructs. So, the model digital systems really well and the do fare amount of analog systems also now a days. But, in the future you will probably have both analog and digital systems, we should be able to describe it in a text based programming language be able to realize circuits out of that.

You can actually take the models that you build in Verilog and do what is called simulation, you can give them test inputs and see what is expected out of the circuit, without actually getting the real components and connecting them and so on, you can write Verilog code, you can give it all the test inputs and so on do that design, verify the design first and then go and do the hardware for it. So, you can do simulation, you can do what is called synthesis and test also.

So, given a Verilog description there are tools which can take that and actually give you a gate level description of how it supposed to be. And this language has been extended into system level also it is called system Verilog. So, we will not teach system Verilog in this course, I will teach basic Verilog for this course and maybe in the other courses that you learn later you may of end up learning system Verilog or maybe Verilog itself in more detail in a future course.

(Refer Slide Time: 14:05)





So, just remember that Verilog is a hardware description language and there are lots of things that are in hardware description language are said different. First thing is in a hardware description language, you have to model what is called concurrency. So, if you have various gates in some sense, all these gates are actually firing simultaneously. So, if you write a C program, line 1 is executed before line 2 of the program, line 2 is executed before the line 3 of the program and so on. There is an inherent sequential nature to the way in which C and C plus plus programs.

However, in language is like Verilog they are inherently concurrent in nature they are very different the way they are written and the way they are conceived in fact. They also have constructs for modeling delays that are there in the real world. So, any physical component has what is called an inertial delay. So, if you have you cannot switch wire from some 0 to 1 instantaneously. So, it takes it is own time to switch, let us say imagine a capacitor which is charging, the charge does not happen immediately, so charging takes some time.

So, we can model such delays using Verilog, the syntax that you will see is fairly simple and you may even see similarity to other languages that you may already know. However, will try and keep this course self contain, so I am not going to ask you to go and run somewhere else and learn some other language and come back, we will try and learn Verilog within the frame work of this course itself.

So, all we have, so far is just a teaser for what Verilog looks like, clearly I will go into these details of how to write Verilog code for circuits. Once you know how to design circuits, then we can talk about how to write things in Verilog. So, this brings be to the end of the weeks lectures, we have modules 1 through 6. So, this week was slightly longer than what would be usual, so I think we probably have 2 and half hours of video, as opposed to 2 hours that will see from next week onwards.

So, there is lots of interactive material and so on some of these you do not have to memorize anything you just know and appreciate that there is history of computing and then there are lots of facts and details that we have. So, one thing I would like you to do is, take the homework exercise really seriously and you should try and do these excise on your own. So, only when you do these things on your own, you will really understand the material.

So, it is always tempting to go and search on Google or talk to your friends and so on.

So, one simple and serious advice that I will give you is go and do this exercise on your own, you will benefit a lot more by doing that then by borrowing your knowledge from others. So, I am hoping that your able to enjoy this week's lectures and if you have anything please get back to us on the forums. So, slowly I expect the forums to be very active in terms of discussions and so on.

So, thank you very much and I will see you next week.