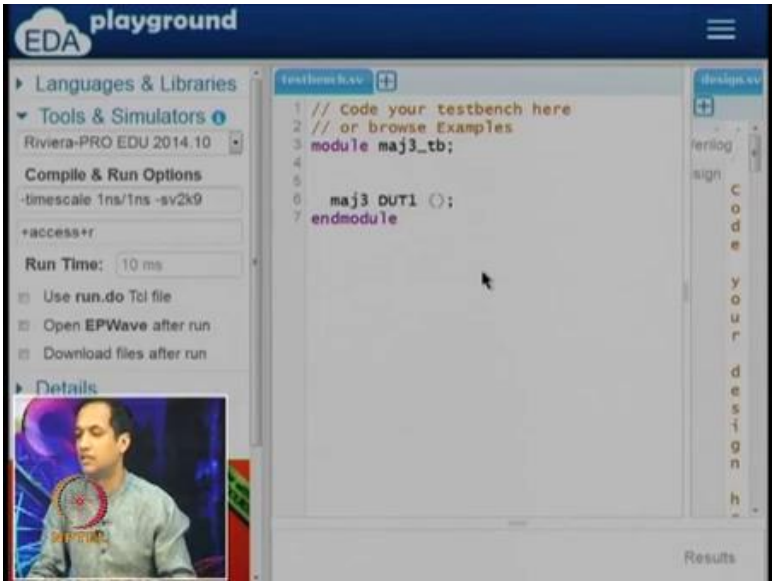
Digital Circuits and Systems
Prof. Shankar Balachandran
Department of Electrical Engineering
Indian Institute of Technology, Bombay
And
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 17
Demonstration of Verilog

Welcome to this module, in this module I am going to show a demonstration of verilog. What I am going to show is, write piece of code and show how a simulated works. So, I am going to use this website called EDA play ground dot com to demonstrate verilog.

(Refer Slide Time: 00:36)



So, let me type that EDA play ground dot com. So, it loads with two windows, so you will see on the top by default it loads with something called test bench dot S V and something called design dot S V. So, I am going to retain the names as it is, so S V actually stands for system verilog, but I will write verilog code here as of now. So, I am not going to write system verilog code, I will write verilog code.

So, what you have to do in this is; on the right side - you can type verilog code, on the left side - you write the test bench for it which is also in verilog. So, initially I am going to write the design part, then I will come to writing the test bench part. So, I will take it full screen. So, what do you want? We want a function called majority 3, so we want a

module called majority 3, which should take which should give 1 output called out and 3 inputs namely A, B and C.

So, I put that description in and let us say I put end module, so at this point, I know that this is the black boxes, which has four ports. At this point, all it has is, it is a black box called m a j 3 which has four ports. Now, I will specify, what are all inputs and what are all outputs. So, at this point, what I have is, I have a black box and this black back has 3 inputs called A, B and C and 1 output called out. So, these ports this order in which the ports are given, you should be careful about it.

So, I have given output as the first one and 3 inputs, later when we use this, we should note that the first one is output and the next three at the 3 inputs A, B and C respectively. So, I have this. So, so far there is no logical description, we only have a black box description of it. Now, I am going to write the internal descriptions. So, for the internal descriptions I already said, we need three wires and three gates. So, I will put three wires A B, B C and C A and I said A B is the AND of A and B and B C is the AND of B and C, A C is the AND of A and C.

So, we have the 3 AND gates and 1 OR gate, which is supposed to drive the output which will take A B, A C and B C as inputs. So, this is the description for the majority 3 functions and this is something that I have already shown, so I have all these things. So, you can see that this website, the module, the keyword module, input and output and so on are all shown as a purple colored thing and all the variables and other things are shown as black.

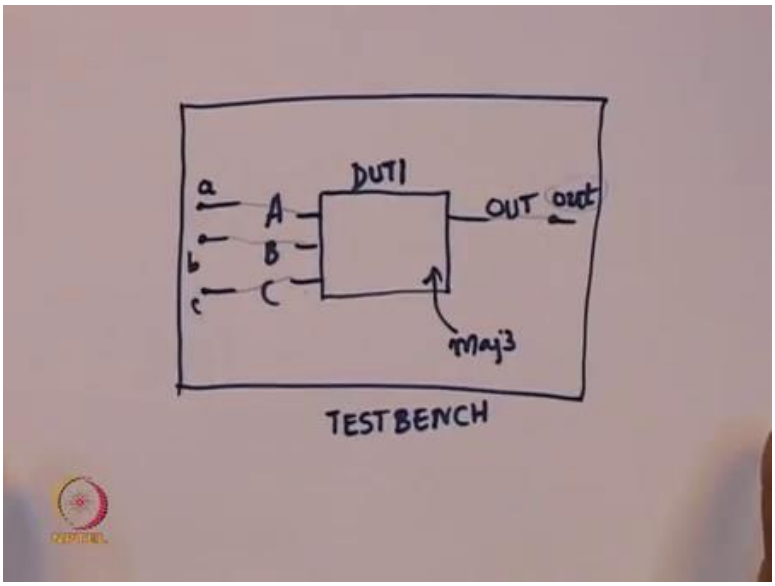
There is you can also command in verilog using slash, slash and anywhere, where you see a command that showing up as brown colored line. So, this is the design description. So, what we have is, we have the design under test, but we need mechanisms by which we need, we should do inputs and we should be able to test the outputs. Let us see, how to write a test bench that is on the other panel.

So, on the top, you can read this, code your test bench here or browse examples. So, at this point, I am not going to browse the examples, you can go and see the examples by clicking below here, we can see examples here, but I am going to write the test bench. So, the first thing I am going to do is, I am going to write the test bench, I am going to call it majority 3 underscore t b and I know that test benches do not take inputs.

So, test benches do not take inputs and they produce outputs. So, test bench is a black

box which is completely self containing. The only thing we need is that we need a design under test. So, the way we are going to do this is, we have going to instantiate the design under test. So, what is our design called our design was called majority 3 and I am going to do this as a declaration, majority 3 DUT 1 and to this, I am going to the give inputs and do the connections. So, I am going to show you picture first, let us see a picture.

(Refer Slide Time: 05:10)



So, let us see this picture here, what we have is, I am going to have a design under test. This is going to be the design under test and let us say this box is format test bench. Let us say, you by this chip from a shop and let us say this bigger box here is supposed to be the bread board and this is the chip that you bought. You should be able to place that chip on the bread board.

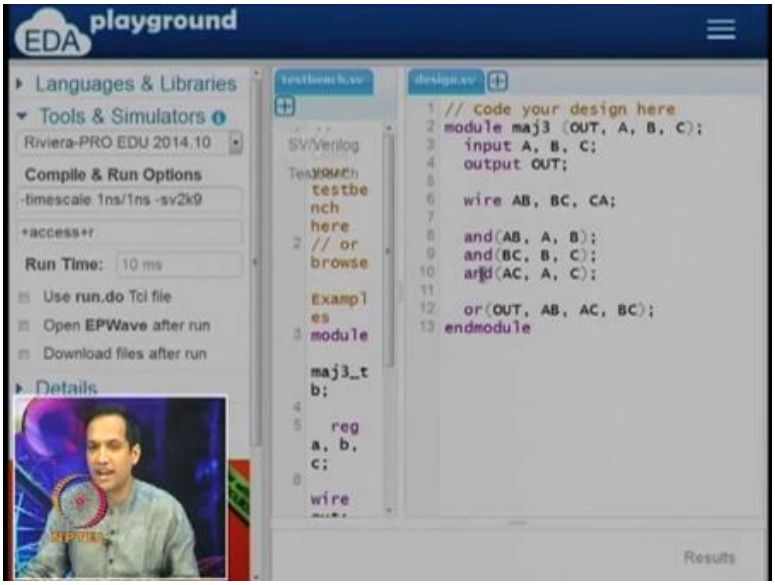
So, there will be three pins that will be hanging out which are input pins and there will one pin which will hang out, which is called the output. So, I am not showing power and other things and this one, I am going to call it DUT 1. This is the design under test 1, we know that this chip is actually a majority 3 chip. So, given 3 inputs, it will find out the majority of the 3 inputs, that is what you have in the line there.

But, what you can see is there are three lines A, B and C. So, there are three pins hanging as A, B and C and there is one pin hanging which is out. What we need is a mechanism by which we can give inputs to A B and C and have a mechanism by which we can see what the output is. So, to do that what I am go to do is, I am going to declare three local things called small a, small b and small c and I am going to declare something called out.

So, these four are going to be inside the test bench. So, see that this is not part of the design under test, they are part of the test bench. I am going to have small a, small b small c and small case out in the test bench and what I need eventually is, I want lower case a to be connected to A. Let us say, lower case b to connected to B, lower case c to connected to C and out to be connected to out.

So, you can think of these as wires that are going into VDD and this as a line that is connected to the oscilloscope. You can think of this out as in oscilloscope probe and a, b, c as something that is going to your power supply and you need three wires A, B, C; and one wire called out to take care of this. So, let us see how to do that.

(Refer Slide Time: 07:21)



So, to do that what are I am going to do is, I need three wires a, b and c, these are remember wires that are local to the test bench. So, these are the wires that you have to connect to the power supply and what we want. And I need a wire called out, so the one thing in verilog is, verilog is case sensitive, which means small case a is different from capital letter A and so on, we need to remember that.

And even though these are a, b, c are wires, I am going to temporally name them regs, we will see the distinction between regs and wires later. I know that it should be called regs down, I will put it right away and I had this instantiation. What is the instantiation do? I currently have a circuit called majority 3, I am going to call it DUT 1 under this test bench and I am going to do this connection.

So, I know that the first line is output and the other three lines are inputs in the majority

function. So, you go and see the description here, if you go and see the description, out is the first pin A, B and C is the other three pins. So, you want the same thing here, by accident, you do not want to connect an output to a voltage supply and an input to be connected to oscilloscope and so on.

So, what I have done is, I have declared local variables here, which are supposed to be connecting to VVD or ground and what not and I have maintain the order in which the design is expecting the inputs and outputs. So, if you have programmed in C or C plus plus, this is like the parameters that you have passing to functions, this is not a function call, remember that it is not a function call DUT 1. So, you have to remove your C and C plus plus ideas for a while.

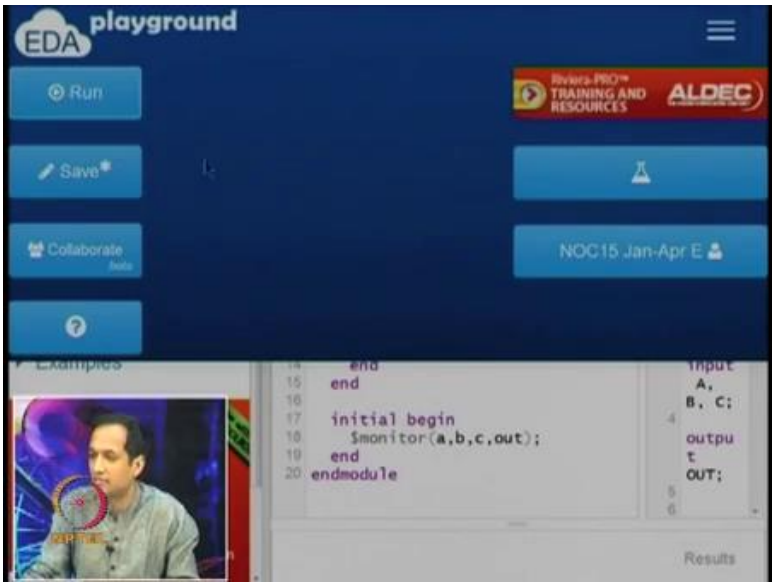
So, think of fresh, do not compare everything to C and C plus plus and ask questions about, if I do this, is this a function call and so on. So, assume that you do not know any programming language and you are starting fresh, this m a j 3 DUT 1 is just an instantiation for DUT 1. In fact, if you go and look at the design, there are three instantiations for AND gate.

By instantiation I mean, getting a physical component and placing them and wiring them. So, we had 3 AND gates here, the same thing here, whenever you see something like this m a j 3 stands for the circuit that we designed, it is not primitive remember. So, we design this, this is our circuit and the name we are going to assign is DUT 1 and I showed this wiring later. Small case a should be connected to capital A, small case b should be connected to B and so on; that will happen because that is the order in which the design is expecting the outputs.

So, we have those, we still do not have a mechanism by which we want to give different kinds of combinations to a, and to be able to c, a, b, c and to be able to see, what the output is. So, if we do that I am going to doing something else. So, I am going to write what is called an initial block. So, an initial block starts with the keyword initial and then you have to put statements in it within begin and end.

So, this is like open and close parenthesis, you have initial begin and end and within that I am going to write a piece of code which will enumerate various values of a, b and c. So, this is a very simple and direct way of writing test benches. If I go and look at inputs a, b and c, they can take the value 0 0 0 to 1 1 1 or if you think of it as an integer, you can go from 0 to 7.

(Refer Slide Time: 11:00)



So, what I am going to do is, I am going to write a for loop, you can see that the syntax is similar to C, expect that it has begin and end and so on. So, I am going to write a for loop in which I have declared a variable called i which is an integer, it will take the value from 0 to 7. So, if you take the integer 0 to 7 and I am going to do something which is very peculiar to few languages. So, I am going to do something like this.

So, I put a, b, c within parenthesis, I put set brackets and I said it is equal to i. So, the interpretation of that is, you take i which is an integer and it is going to take a values from 0 to 7. So, 0 to 7 can be represented using 3 bits. So, i you can assume that it is going to have 3 bits and when you say set bracket a, b, c equals i, it will take each bit of i and assign to the corresponding bits in a, b, c.

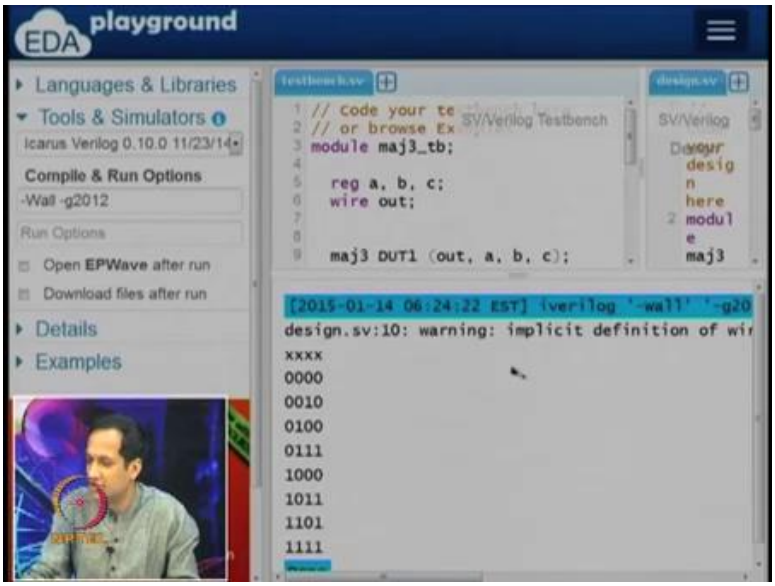
So, let us assume that i has three bits, i 2, i 1 and i naught, i 2 is the most significant and i naught is the least significant, then i 2 will be assigned to a, i of 1 will be assigned to b and i of 0 will be assigned to c. So, with this you are able to now see that a, b, c can actually take the inputs. So, remember, if you go back to the circuit, we need to be able to connect the pins a, b and c to power supply, we have done something now.

What we have done is, we have now able to take different values of a b and c. with because i is changing, I will start with 0 0 0, then it will go to 0 0 1 and this will go to 0 0 2 and so on. And what I want to do is, I also want to do this every 5 time units or source. So, I will given 1 input, after 5 time units I am going to give another input and so on. So, do not ask me, why it is 5, I just picked it arbitrarily, I am putting some value there 5.

So, this takes care of the fact that I am able to change the values of a, b, c, this takes care of all combinations of a, b, c. I still need a mechanism by which I should be able to observe the values. So, I am again going to open another block called another initial block, so there is already one initial block and there is initial block, in this initial block I am going to write this one.

So, call a function called monitor and I am going to say monitor that values of a, b, c and output. This is what I am going to do, so we have a mechanism by which we have instantiated this circuit that we wanted and we have the statement where we can actually take the inputs, all possible combinations of inputs and we have another statement here, another statement block here, which will monitor the outputs.

(Refer Slide Time: 14:00)



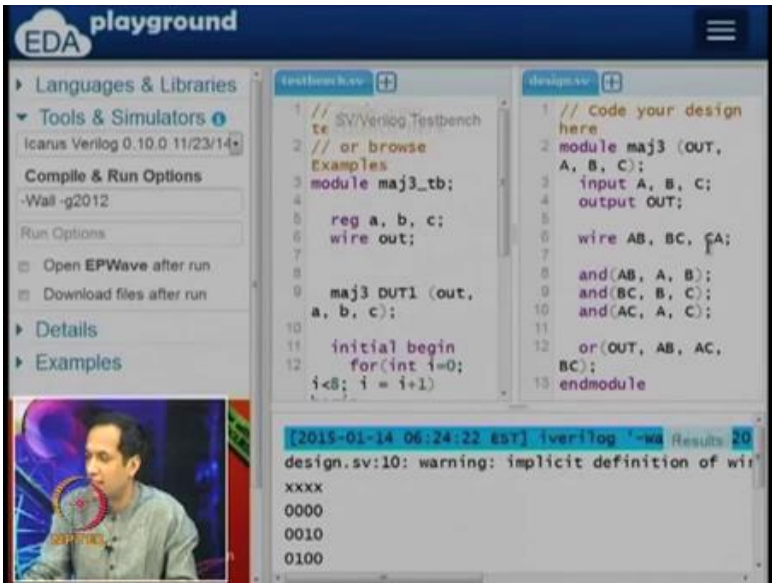
So, now, I am going to show sequence of steps with which we can compile, run, simulate and so on. So, in the process, if you discover any errors or bugs, we will fix those things also. So, the design is on the right side called design dot S V, on the left side, I have test bench dot S V and I am going to now simulate it. So, to simulate the first thing I am going to do is, I am going to pick this particular compiler called I verilog.

So, I verilog or Icarus verilog, I am going to pick this version 0.10.0, I am picking that version and I am going to run. The way to run that is you go to the thing on the top, click that and press run. So, if you press run, it goes into another mode, where it changes to red and in the bottom, you will see a bunch of output coming, so this is actually in the bottom of the window. So, there is something there and there is some warning, it says

implicit definition of wire A C, there is some warning there and it seems to be printing something on the screen.

So, let us first look at the warning, so you should not ignore any warnings that are provided, let us go and see, what the warning is. So, it says implicit declaration of wire A C, let us go and look at it. So, I am using the wire called A C here, but I called this wire C A. So, A C is not even declared. So, this is like in any programming language all variables should be declared, I am going to change that first, so I am changing it.

(Refer Slide Time: 15:47)



Now, I go and run it once more and it is not giving me any warning, it is starting with something strange x x x x. And then there seems to be some, it looks like some combination, something is happening, some big stream is coming out, let us see what that is. So, this is coming out, because of our test bench, our test bench is giving the inputs, but we also wrote another piece of code which set monitor a, b, c and output.

So, what you have seeing is each line, you are seeing some combination of a, b, c that we tried out and the output that we got for it, so let us see this. So, if a is 0, b is 0, c is 0 that will happen when i is 0, a, b, c will get 0 0 0. So, in some sense, this was given as inputs to the majority 3 circuit and the output from that comes which is called OUT in capital letters comes back as OUT for within the test bench and that is getting printed out.

So, this input a, b, c was connected to the inputs capital A, B, C in the majority 3 circuit, there is some evaluation there, whatever the output is that comes as capital OUT in that and that comes back as small cases out here and that gets printed out. You can see that

the output is see, whenever there are 2 0's, you will see a 0, if there are 2 or more 1's, you will see a 1, you can see that here.

So, in this case, there are 2 0's and a 1, the output is 0, in this case, there are 2 1's and is 0 the output is 1 and so on. So, this is actually doing our majority function. So, the nice thing about that is this equivalent to your print statements. You would do this in your C and C plus plus program, you will go and print, and see what the values are and so on, I have done the same thing here.

So, another thing that is very useful for a digital circuit design is it is to be able to see the waveforms. Sometimes the text is one way of looking at it, but picture is also a very useful thing to have and EDA play ground has a mechanism by which we can see the waveforms, so you see the left hand side, there is open EP wave after run. So, I did not click on that earlier, but you can click on it and it will also open up the wave form, so to be able to do that you need a few more things.

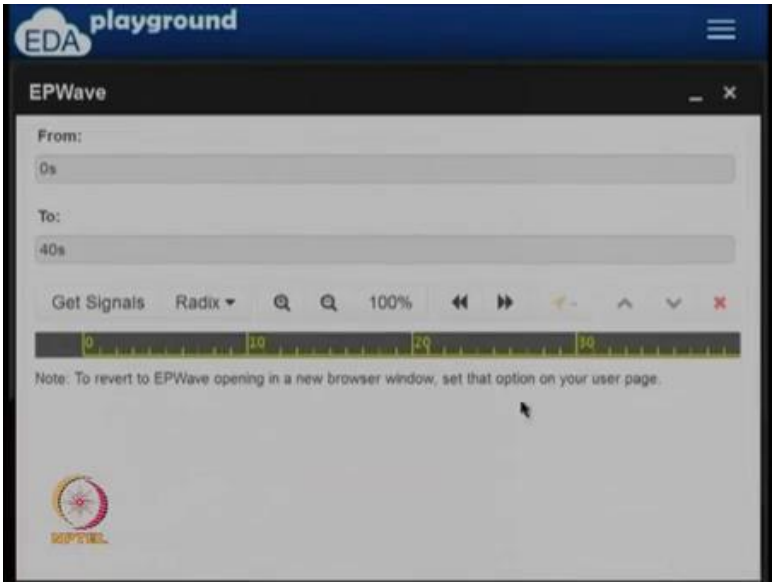
So, in this initial block, I am going to add two more lines. So, one line is going to say, I want a waveform file and the waveform file is going to be called dump dot VCD. So, this VCD stands for value changed dump. So, do not worry about it right now, we are saying that all the values that are there in the circuits should be dump to the file called dump dot VCD and this dollar dump wars essentially says dumps all the variables.

So, I am going to now again run it. So, when I run this again it look likes something is happening. And now, you see a new line, VCD in port dump file dump dot VCD open for output and then we see all these things, it comes to done. So, you did not show the way form, because I did not click on this open EP wave after run. I forgot that, let me now do that, go back and run it.

(Refer Slide Time: 19:10)

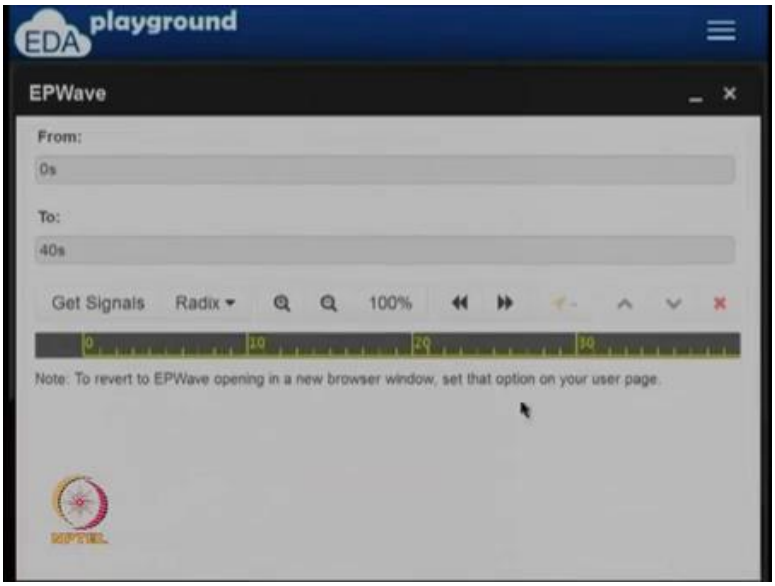
Now, will you see that, a new window opens up here, it says, use get signals button to add more signals with the wave form view. So, pay attention to all the messages that you get, do not ignore the messages. So, it is says, use get signals button.

(Refer Slide Time: 19:21)



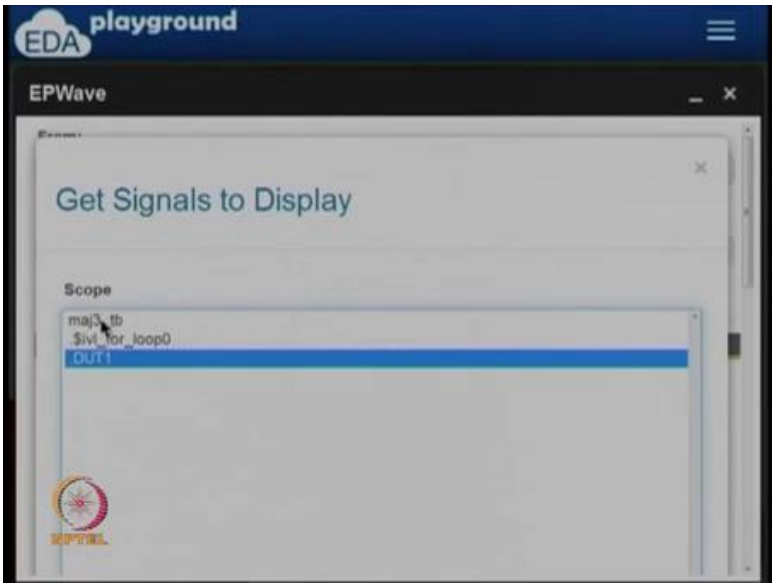
So, let me see, where they get signals button is, there is the get signals button here. So, I am going to the click on that, it is shows various things here. So, if you scrolled down you can see that currently, there is no signal; that is watch for viewing. So, we cannot see any wave form yet.

(Refer Slide Time: 19:28)



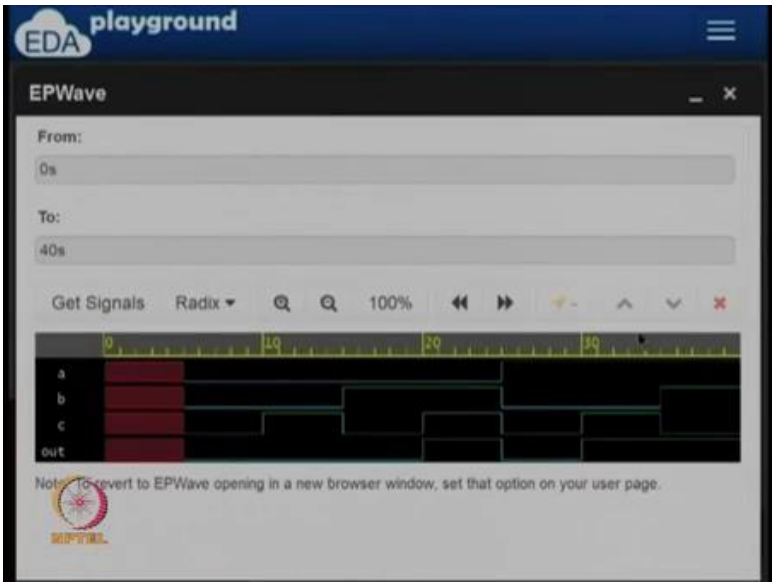
So, I am going to click on majority 3 underscore t b, majority 3 underscore t b has these three wires a, b, c and I have wire called out. Similarly, if you clicked on DUT 1, you will see it is internals wires, a, a b, a c, b, b c, c out, and so on, it is listed in alphabetical order.

(Refer Slide Time: 19:58)



So, what I want to see is, whether the wave forms for the test as seen by the test bench, test bench is where we actually see the output, final output will be that as bench. So, I am going to pick all the signals in the test bench and say append selected, actually something change in the back end and I will say close.

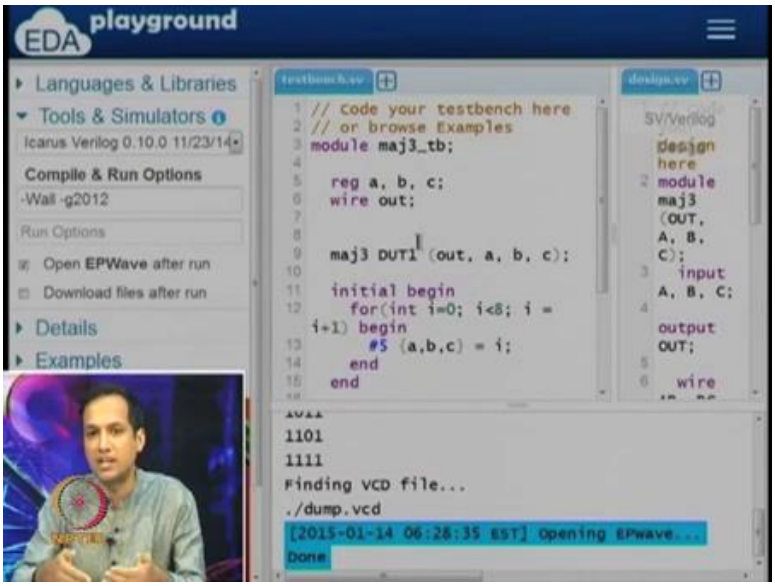
(Refer Slide Time: 20:19)



So, now what we have is, we have mechanism by which. So, at this, I brought EP wave to the four ground. So, what we see is, we see the three signals a, b and c, which are given as inputs, the output at the end and you can notice. So, let me pick this time, a time let say 30, a time t equal to 30, I can see that a is 1, b is 0 and c is 1, the output is 1. A time t equal to 20, a is 0, b is 1 and move it to this line, a is 0, b is 1 and c is 1, the output is 1 and so 1.

So, you can that in the wave form viewers, times in the x axis and the y axis, you see various signals and for every signal, you can see the time frame. So, what are the values of difference signals at difference points of time and you can zoom in, zoom out and so on. So, you want to zoom in and see something or you can zoom out, in this case the whole wave form fits here, there is no more zooming out, we can all of this design fit into this particular line, so it is done here. Once you are done with the wave form in this case, I know that it is correct, I will close it, it brings me back to this window here, where I saw the design on the left side. And the design on the left side, design on the right side and test bench on the left side.

(Refer Slide Time: 21:45)



So, the few things that I did not talk about are what is this initial begin and end, what this magic and what is this hash 5, suppose to do. So, this hash 5 is essentially saying do this after every 5 times units. So, take a, b, c current value of i assign it, keep quiet for 5 units, then go back and iterated one more time after 5 time units. And then take the current value of i and assign it to a, b, c; keep quiet for 5 minutes, then come back and to do this and so on.

So, 5 time units not necessarily minutes, the one another thing that needs explanation is why this starting with x x x x, we did not print x x x x, I did not take anything, we will talk about that in the later class. One final thing that I want to you note down is, currently if you click on this, if you see that save has a star, which means the design is not save. So, to be able to save, you click on that save and in this case, I was already logged in, otherwise, it will prompt you for logging in either with your Google account or with a face book account.

I suggest that you log in with you Google account that is attach to your course. So, later what we will do is, we will start seeing assignments and so on, on this plat form. So, it gives you a nice way by which you can describe that design, we can see the wave forms and so on and this gives you a lot of practice that you can do. So, eventually I intent to use is plat form to show that demonstrations for the course itself.

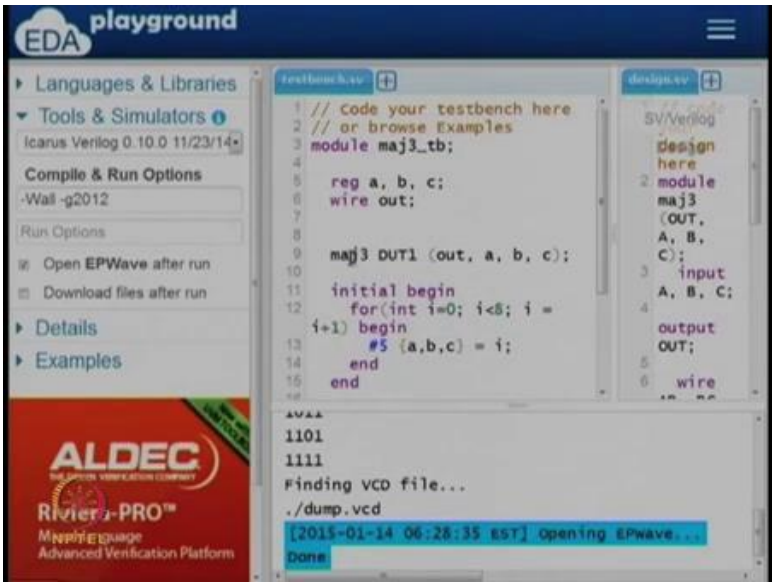
So, it is good to be able to login to this plat form with a Google account do not forget that later at some point of time we may even do evolution and so on, using the platform

itself. So, this brings me to the end of this week's lectures. So, this week was quite a bit of things, we saw quite a few things, we started with combinational blobs, we saw en coders, de coders and multiplexes in the lot of detail.

We also saw basically what a de multiplexes is, I want to you think about, where the difference things are useful, the differences between them and where each one is useful. Also remember that, by changing a few things you can go from one circuit to the other, the other thing that we did wish we look that the basic set up begin verilog, more importantly I exposed due to this platform.

So, one of the things that you can do yourself is, you can actually go and write a design which will do a 3 input XOR, go and try and do this yourself. Write a design for 3 input XOR and modify that test bench accordingly. So, it is already taking care of 3 inputs iterating through them and so on, all in need to do is change the circuit that you want here.

(Refer Slide Time: 24:31)



So, you want to whatever you call here, let us say module XOR 3, if you call that here, you have to put XOR 3 here, rest of it can remain this same. So, if you feel, it will little more confident, I will suggest that you go and come out with your own problems and write your own test benches and so on. From next week on words, what we will do is we will provide more of these examples and sometimes, I will give you the test benches which is you can test to you design. Sometimes, I will give you the designs and ask you to write the test bench is for it.

And later, when we move towards the course, towards the middle of the course and later, we will see that you will have to write both of those may be right. So, that we get enough practice in writing verilog. So, this is actually brings me to the end of week 3 and I hope that, this week's lectures are understandable and you are able to enjoy the lectures.

So, thank you and I will see you on next Monday.