

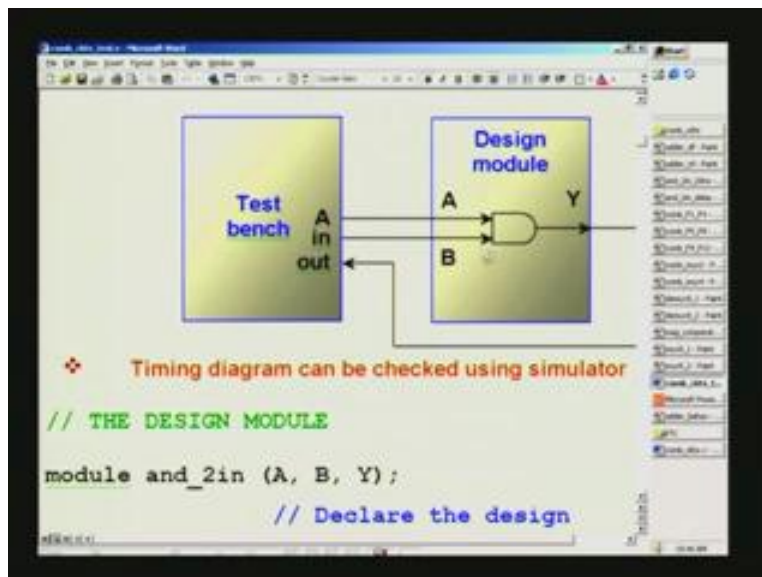
**Digital VLSI System Design**  
**Prof. Dr. S. Ramachandran**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Madras**

**Lecture - 24**

**Simulation of Combinational Circuits**

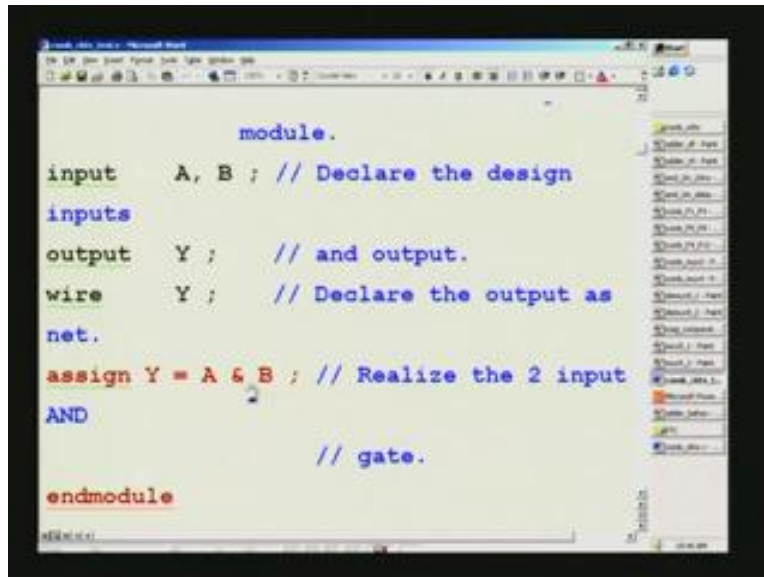
We saw earlier, realization of simple design using two input AND gate. We have also seen the set test bench with different nomenclatures here. For A input, it is same as the design A input whereas in is actually connected to the B of the design. Similarly, Y of the design that is the output is connected to out here, as far as the test bench. We have also seen the design module as such.

(Refer Slide Time: 2:06)



The core statement is assigned statement with A and B AND-ed and assigned to Y.

(Refer Slide Time: 2:40)

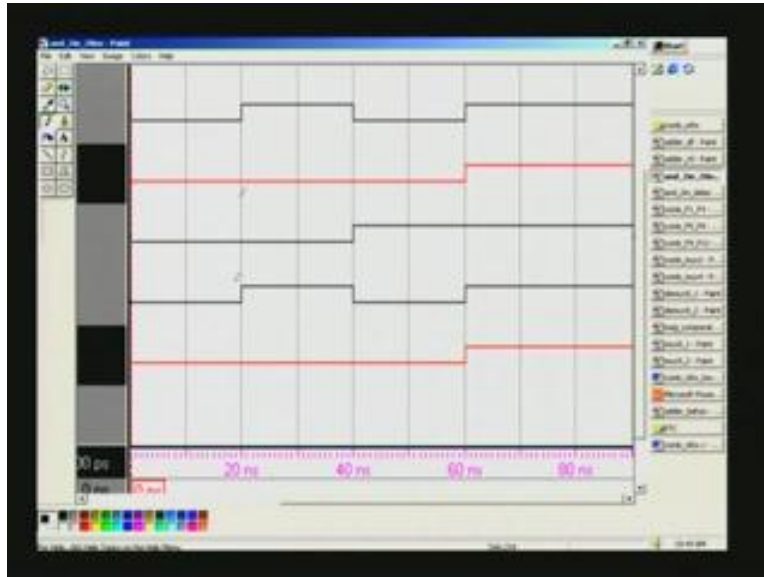


```
module.  
input  A, B ; // Declare the design  
inputs  
output Y ; // and output.  
wire  Y ; // Declare the output as  
net.  
assign Y = A & B ; // Realize the 2 input  
AND  
// gate.  
endmodule
```

We have also seen after invoking the modelsim and the waveform are quite difficult, so we have done it on paint although you will not see live as such. It is an actual capture and we will see right now in this. This is basically from the wave which we have seen after invoking modelsim and we will also invoke it once again. It has been pasted in paint and that is why you can see readabilities far enhanced now. So, you see that A and in are the two, I hope you can read this on the TV monitor. There are the two inputs of the test bench and out is the final output and this corresponds to A and B inputs of design and Y output corresponds to the out. Here on the right, you see the actual waveform. Just remember A and in our first 2 are the inputs and last one is the output what you see here.

You can also see at the bottom, time in nano seconds. Remember that, we had 20 nano seconds displaced at every point of time. For example in this, we have seen the test bench here; we have seen that, after 0, it is 20, 40 and so on. So what you have here is precisely the same here 20, 40 and so on and actually the second division is 20.

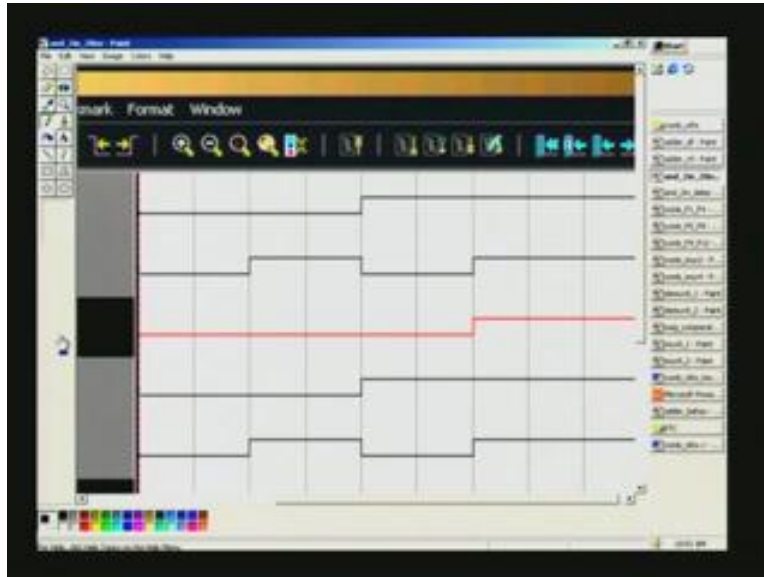
(Refer Slide Time: 4:59)



Let us go to the different inputs. See in the test bench, the two inputs are A and in. This is 0 level and this is 1 level here. Again for in input, this is 0 up to 20 nano seconds and for the next 20 nano seconds, it is 1. So the combination that you have as per the truth table that we have seen earlier for this AND test bench is 0 0. Corresponding to 0 0, this AND gate so you expect 0 as the output and that is precisely what we have here. This is 0 level. You also see here the exact state and since we cannot see it live, if you click here, this also will change. But here it will not change because we have captured earlier and it is rather static and go dynamically right here.

If I click here, I will be getting the same input that you see here. For example, it will be 0 here and 0 here, so the moment I click here, these two will transformed as 0 0 and so is the case anywhere. It will reflect the point at which you have clicked whatever be the logic level at that point of time, it would reflect there. You can see 0 0 combination, the next is 0 1 here and corresponding output is still 0 because it is an AND gate. AND is simple multiplication like thing in a binary. Next case, you have 1 0 for that A and in and for that also we will give you only 0.

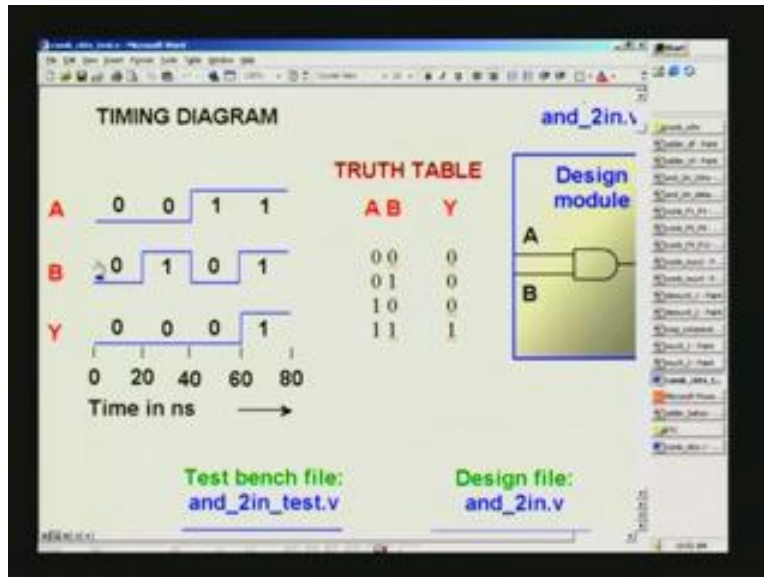
(Refer Slide Time: 6:48)



So this is the AND gate, this is the output and right for all the three combinations you have only 0 here and it goes to 1 when the last combination 1 1 is encountered. That is what is here and another thing that you would notice is this waveform. You do not have to remember, you can physically see this is nothing but A and the same waveform appears here also. This corresponds to the design; whereas, this corresponds to the test bench. I hope it is clear that whatever connection we have made at the test bench and design is precisely the same and that is what you get here.

What you have seen is A and input are corresponding inputs in designs A and B. So, B must be the same waveform as in here and similarly the out must correspond to Y output here of the design. These two must correspond, so that is this and this. So, it is precisely same here and you can see it here so right here and these are all the two outputs. This is nothing but what you have seen here in test bench and this is what you have got right now.

(Refer Slide Time: 8:07)



The timing diagram that you have seen is precisely the same, so A and B exactly in the same order. Let us move on to the next, one more thing we have mentioned in this is the test bench applying at different points of time.

(Refer Slide Time: 8:49)

```
time
// 20 ns,
#20 A = 1; in = 0; //40 ns, and
#20 A = 1; in = 1; //60 ns.

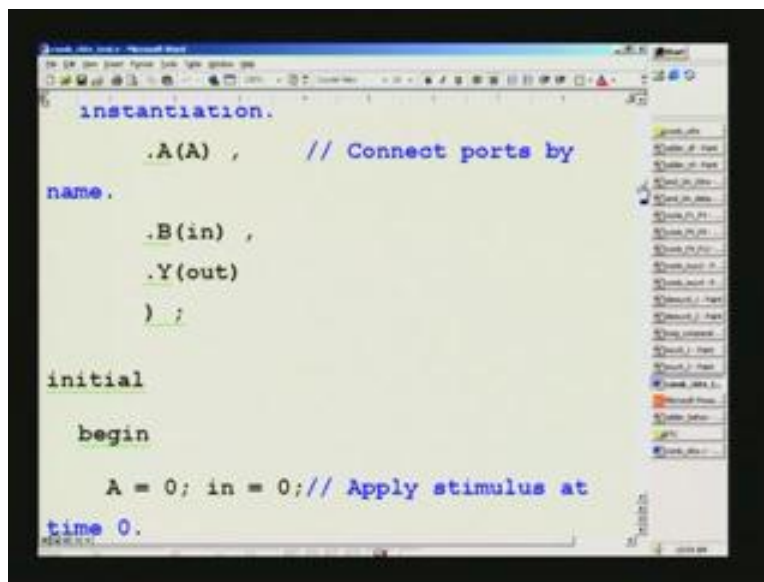
#40
test // Run long enough to
// all possibilities,

$stop ; // and stop.
$finish ; // Terminate simulation.
end
```

We have seen that every test bench gets the very same waveform as you have in design. Only then, you will get a feel that what you have really connected and really appears on the top level. Test bench is always the top level.

At least for once, you should verify that it is actually reflecting your design wherever you have connected. Once you have done, there is no need for you to do it every time. Just to get a feel that is make you feel for it, I mentioned that. So we have seen that, at every 20 units of time, you see all are precisely 20 units of time. Earlier, a question was asked whether we can have cumulative. It was cumulative as far as the times are concerned instead of 20 etc., you had 20, 30 actual time reflection and using non blocking statement as such that is less or equal to instead of this, you use less or equal to here and then put this 20 before 0.

(Refer Slide Time 9:14)



```
instantiation.  
    .A(A) ,    // Connect ports by  
name.  
    .B(in) ,  
    .Y(out)  
    );  
  
initial  
  
begin  
  
    A = 0; in = 0; // Apply stimulus at  
time 0.
```

At that point of time, this will be assigned there and if you had given 21 here, let us say, then at 21 nano seconds only, this will be assigned. We have also seen earlier, what will happen if there was a time scale here?

(Refer Slide Time 9:50)

```
begin

    A = 0; in = 0; // Apply stimulus at
time 0.
#20 A = 0; in = 1; // Change inputs at
time
                // 20 ns,
#20 A = 1; in = 0; //40 ns, and
#20 A = 1; in = 1; //60 ns.

#40
                // Run long enough to
test
```

We saw that 1 nano second is the time base and you have marking in steps of say 20 major markings as such.

(Refer Slide Time 10:01)

```
`include "comb_ckts.v"

// This is the design file.

`timescale 1ns/100ps // Time in ns,
                    // with a resolution
                    // of
                    // 0.1 ns.

module comb_ckts_test ;

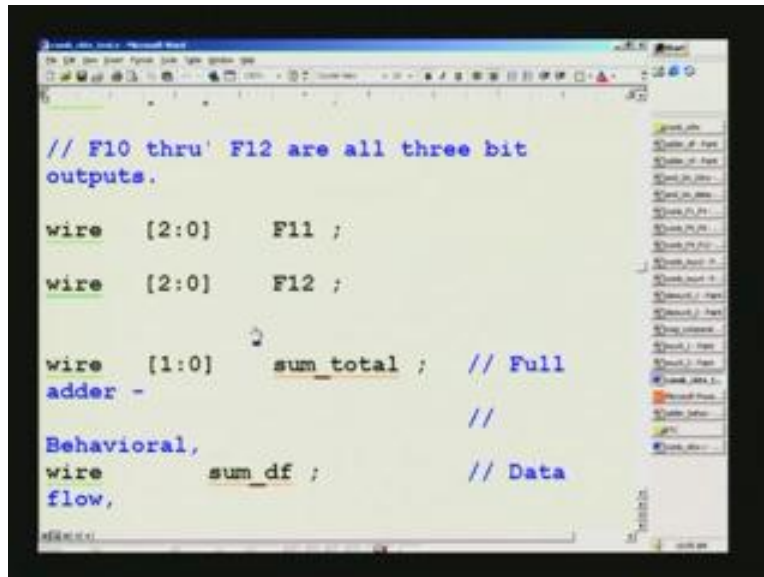
// Declare the test module.

reg    A ; // Inputs.
```

20 into 1 nano second is 20 nano second and what about this? This is 0.1 nano second. You can even specify here instead of 20 you can specify say 20.3 or something like that.



(Refer Slide Time 10:20)



```
// F10 thru' F12 are all three bit
outputs.

wire [2:0] F11 ;

wire [2:0] F12 ;

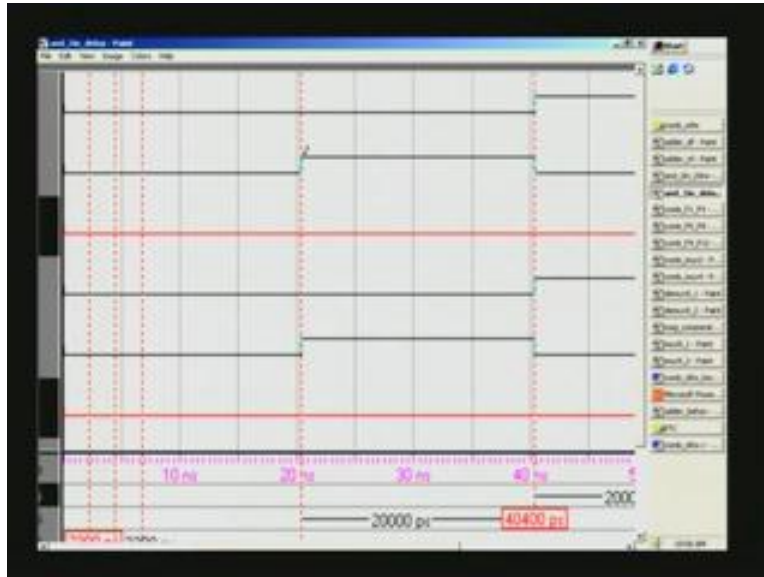
wire [1:0] sum_total ; // Full
adder - //
Behavioral,
wire sum_df ; // Data
flow,
```

You can have in steps of 0.1. So, what will happen if you specify 20.35? What is your guess as to what should it be? I will answer that; it will round off to the next higher value. So, 20.35 will be rounded off to 20.4 and that is precise. If you had commented statements which have only this 20.35 etc... specified, then what will happen and run the code in simulator. You will get the waveforms like this. Notice that at the bottom, different timings are mentioned. It is precisely the same as far as the function is concerned.

It is exactly the same except for slight difference here. You notice that, this transition is happening. See the output, this is the output. Input transition is happening at 20.35. It has been set in the original file.



(Refer Slide Time 11:18)



What will happen here? Let us see what it reports. See, here are 20400. It is reporting it as 20.4 nano second. So 20.35 have been rounded off to 20.4, that is, what it means. Another thing that you see here is that you can set as many cursors that you want here.

For example, you want to measure the time interval between 1 point of time; for example from this to this next transition, how much time it has taken? What you can do is, just make a cursor which we will be seeing in the modelsim although the visibility will be poor, we can **highlight on that**. You can click the cursor here and then say add the cursor then this dotted cursor will be put on that point of time and you can see once again. The next transition also is staggered because the very first thing was 20.35 and there after it was maintained as 20 nano second. It is a relative thing. Automatically everything will be displaced by that 20.35. Actually in this system, we will interpret as 20.4 and mind you all these timings; do not appear in synthesis because in the design you are not supposed to give any timing as such. This is exclusively for simulation.

Otherwise, the functional aspects are all precisely the same throughout. You can see one more here, the actual cursor is right here and once again there is marking here. It marks the interval between this cursor and this cursor. That is how you can find out the rise time, fall time of wave and so on, if there is ringing but it will reflect only after the back

annotation and not this stage. Having done this one, we will invoke the modelsim and then go on to the combinational circuits design.

We invoke the modelsim here and let us create a project. So, this is the project location. You have to give a project name, let us say this is the Combinational Circuit. We will give it as comb circuits and there it must be in work directory. What it says is, a project by the same name is already exists, so it is asking whether it should overwrite and just answer yes. There is one thing called library here and it must be in the work directory before you start compiling or loading the design and so on. Let us see where we are.

Here, we have opened modelsim and using that open project. The combinational circuits project is already there and we are right there, you can just give pwd here in the command mode and then make sure that we are in d user ROM d VLSI design etc.

This is the directory that we have all our files and if you want to view all the files in this particular directory, just type directory in this prompt here, modelsim prompt. It will list all this. It may be difficult for you to read. We have combinational\_circuits.v and then combinational\_circuits\_test.v. This is what we have already seen earlier and we will be seeing again and this is what we are going to simulate. Before we simulate, we have to do the compilation that we have already seen in the flowchart. Let us compile that and designs then compile. You see here combinational circuits test.

Since, we have included the design, there is no need for you to mention specifically all the lower modules. It is enough if you just compile the test bench as such. Double click here, it will automatically start the compilation which you can see on this window and I will just double click here on this file. It has compiled the combinational circuits that are a lower module first that is a design here and then followed by the test bench here. Once you are done, there are no errors and later on we will deliberately create errors and see how the compiler functions. Right now, we will have a smooth flow over our simulation. Once you have finished with compilation, you are done with it.

You dismiss off that window and then next step is to load the design. What you have compiled will have to be loaded. It opens another window with load design, we have seen

it opens this load design window and the module of interest is combinational circuits\_test that is the test bench. Double click here or you can just say single click and then say add here, then say load. Let us see if there are any errors, again it will point out even during loading time. What the compiler is unable to track the errors? This loading may do it and later on we will see by injecting deliberately.

We are ready for viewing the waveform. You use view and then signals. Click on the signals here and then it opens the signals window. In this, you see view and with view what you have is wave. Then, go on to the last option called signals in design and when you click on that, the waveform is opened. It will not require signals, so you can dismiss off that window. Now, this is the waveform window and right now it is blank. If you run this one, you notice that you have all this listed. Unfortunately, you will not be in a position to view that here. That is the reason why we are copying from this into paint and later on displaying. This will be using the same technique even here. Just listen to my words then you can make a guess as to what you are looking at.

Right here you see A B C, these are the select pins. If you remember in the combinational circuits we had used for mux selection and on 8 input mux, we have used. So I0 to I7, these are the signals and then we had a magnitude comparator comparing two numbers. These are  $n_1$  and  $n_2$  for that. One more small design was that which computed the sum and then if greater or less, more or less, combination when compared to a preset value is encountered, it sets the some output which will be there listed elsewhere.

Similarly, we have also seen so many outputs such as F1, F2 and so on, they are all listed here. Let us see here, in this column right now, it says x x because the simulator is not run as such and in order to run we will just see what commands are there. We have some lens here, you can zoom in and before that, let us get the waveform then we can do all this. You restart if you want to restart the simulation and straight away we will go for run all. What will happen if you give run all?

I had clicked on run all, you see a wave form pattern here and it has also opened your combinational circuits\_test where the test bench.v. You can see that, it has stopped at the last statement called the Stop. I am afraid that you are not supposed to read this once

again. Anyway, we have the actual bloated up combinational circuits file here, we will see that. Now that we do not need this dot v file we will remove that window. You see waveform here; we can improve this view by pressing this. This is the thing you see here that is third, fourth one. This is for zoom full. The entire vision is had here as far as that functionality is concerned and you see so many activities all around here. These are all corresponding to I0 through I7 and so on. In fact, activities are found everywhere and still it is small.

What you can do is you can just increase this and keep increasing using that plus or if you say minus, you can decrease it. All this will be very difficult for you to find. We will open a particular functionality as such and then describe that. There are more controls here, you can create cursor here. You can just click anywhere, a cursor appears and this is at 80 nano seconds. These are all graduated in 20, 40, as you have seen in one of the AND gate realization earlier. It is precisely same here and you can go from, let us say this we had selected one of the signals called I2 and wherever there is a transition occurring at I2, we can go there by pressing this arrow here.

See for example, it has gone to a negative transition of that I2 here and one more time you click on the left arrow, it goes to the positive transition. Like that you can go for any other signal. Suppose I have more than one or two signals, you can just say or let us see, we will use the other arrow in this time. So, it has gone to negative of transition of the I2 and with this it has gone to negative transition of I3. It may be that we will have to find out why it did not go to this.

So we will have to experiment with it and get a feel of the tool as such. This basic thing is that you can just utilize for your benefit going from one transition to another. Suppose, you want a cursor to be added, if you say add cursor, it has added a dotted line here. I hope you are able to view; it is possible in the paint shop image. That is the difficulty here and the cursor has moved here.

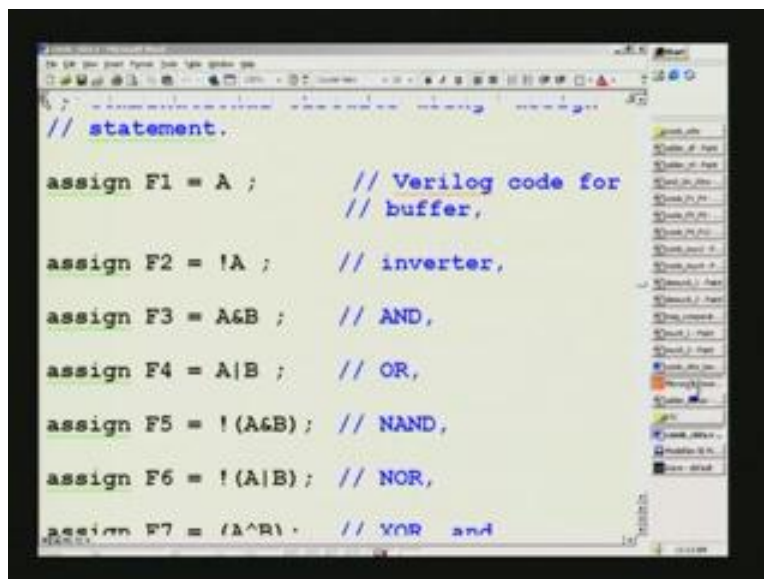
Suppose, I want to put, let us see what happens? If you move this one, that dotted line cursor has moved to the previous position. So, you have to make some alteration to it, may be we have not understood rightly the tool or there is a bug in the tool, it does not

freeze at that point. You may have to make a couple of attempts. Let us see here, we make it here, it has made cursor right at that point. Let us transfer it here, so there is no longer any cursor but earlier thing also has been marked. And if you want to remove a cursor, let us say this; so, this cursor has been removed.

Last in first out, something like that, you can experiment around that. This is not a very important thing. You can just play around and find out. Let us go to the functionality as such. Major things we have already covered and more will be covered little later or in the subsequent lectures.

Let us see the first thing that we have and we will go to this which is combinational circuits that we have. We have seen that so many inputs are listed here. We will straight away go to actual functionality where we have put the assign statements etc. The first assign statement is here and we have also a, it is much simpler to view this Power Point.

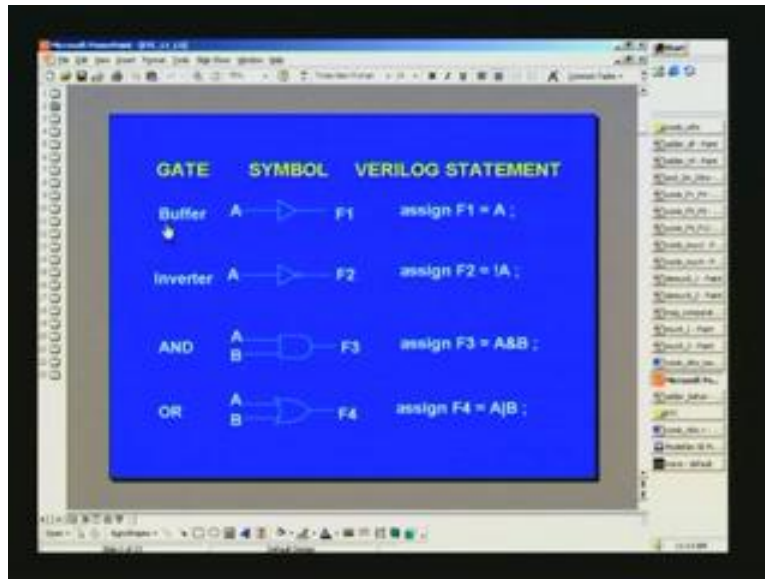
(Refer Slide Time: 24:38)



```
// statement.  
  
assign F1 = A ;      // Verilog code for  
                    // buffer,  
  
assign F2 = !A ;    // inverter,  
  
assign F3 = A&B ;   // AND,  
  
assign F4 = A|B ;   // OR,  
  
assign F5 = !(A&B) ; // NAND,  
  
assign F6 = !(A|B) ; // NOR,  
  
assign F7 = (A^B) ; // XOR and
```

We have seen buffer, inverter and OR, so this F1 through F4, we have this functionality. Let us first view this waveform and that particular thing is being opened right now.

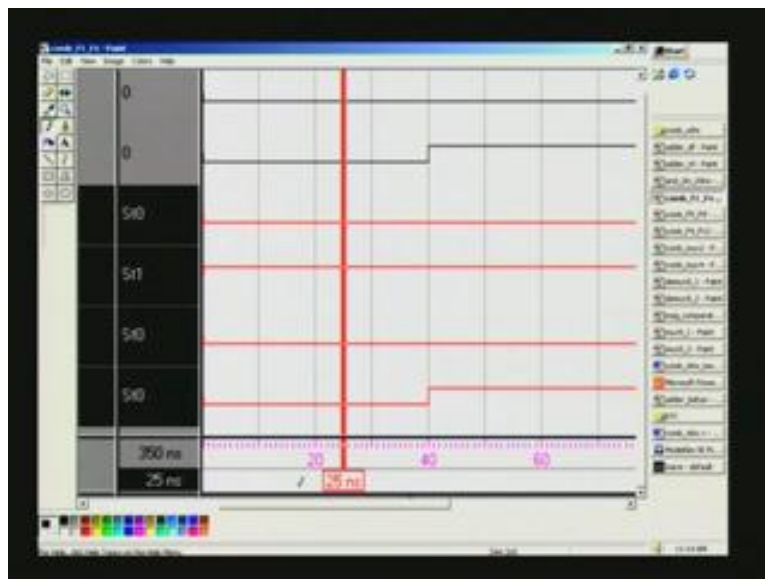
(Refer Slide Time: 24:42)



GATE	SYMBOL	VERILOG STATEMENT
Buffer	A → F1	assign F1 = A ;
Inverter	A → F2	assign F2 = !A ;
AND	A, B → F3	assign F3 = A & B ;
OR	A, B → F4	assign F4 = A   B ;

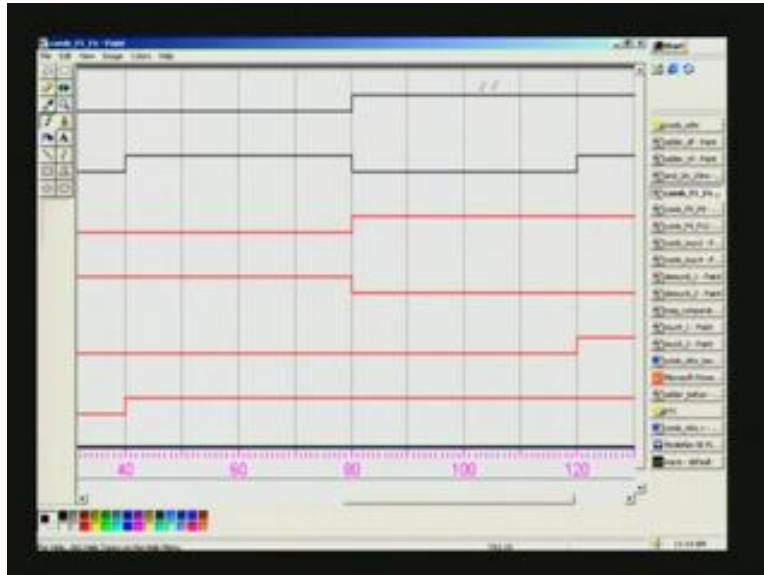
You see here A and B and F1, F2, F3, F4, remember that F1 is nothing more than a buffer. So, buffer of A signal; whatever is the A signal, it should get in F1 and F2 is just an inverted signal of A. Let us just remember these two and see the third column of F1 and F2 is the fourth one. The first two are A and B inputs, so you just remember and once again the time limit is here. And if you remember, we have given 125 nano seconds at one point of time. That is what is marked there, but it is not of much concern right now.

(Refer Slide Time 25:30)



It just shows the cursor can be placed anywhere. So this is the cursor that you had seen and going to the 2 inputs, what it does is, this is A and it is in fact 0 all through and the corresponding output which is buffered is also 0.

(Refer Slide Time: 26:08)



It just follows the input here because this is the output for the buffer. So, whatever is the input, we have precise this same here and next is inverter, this till the end is same and next is F2 inverter as we have seen here in the Power Point.

The next one is the inverter, A is inverted and assigned to F2 and next two also we will just remember AND and OR of A B. Go to this once again; see all marked by black are outputs here. So, A and B only are the two inputs. You can very easily remember and where we make the transition, we can find out from here also. See this is 0, here, this is 0 and now what we are looking for is the second output which is inversion of the A. What is happening here? You can recognize from this, this is high, that is here because it is low here. This is the inversion of that signal. It is high here, that is what you want. When the input goes high, it goes low because it is an inverter.

You have thus checked the functionality of inverter and next we will check A and B so that is the third one. F3 is the AND-ing of the two inputs A, B. A, B are the top ones here. You see 0 0 condition here and for all 0 0 condition you naturally expect 0 here. And for



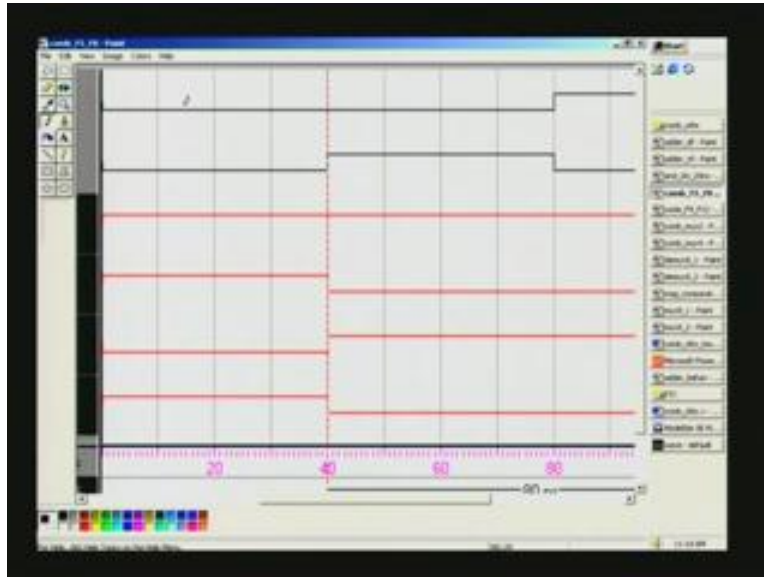
0 1 or 1 0, that also will be 0 that is why it is continuing at low here right up to this point. 1, 0 also is 0 here and now up to this, it is same. 1 0, this is 1 0 here. Here it is 1 1. So at that precise point of time, it goes high here. This is AND gate here and similarly OR gate. OR gate is F4 and add with A and B. Again 0 plus 0 is 0 here. 0 plus 1 is 1 here and it continues to be 1 because 0 1 or 1 0 or 1 1, they are all basically 1, so it is continuing to be 1. We have seen the functionality of first four outputs; similarly we can see the second one. The second one is NAND, NOR, EXCLUSIVE or EXCLUSIVE NOR. This is F5.

(Refer Slide Time: 29:04)

GATE	SYMBOL	VERILOG STATEMENT
NAND		<code>assign F5 = ~(A&amp;B) ;</code>
NOR		<code>assign F6 = ~(A B) ;</code>
XOR		<code>assign F7 = (A^B) ;</code>
XNOR		<code>assign F8 = ~(A^B) ;</code>

F8 - NAND, NOR EXCLUSIVE NOR, EXCLUSIVE OR with A B as inputs. So we will just see where we are, so A B once again is in F5 to F8. These are NAND, NOR, EXCLUSIVE OR, EXCLUSIVE NOR and see in this case the first output is this NAND, so this is 0.

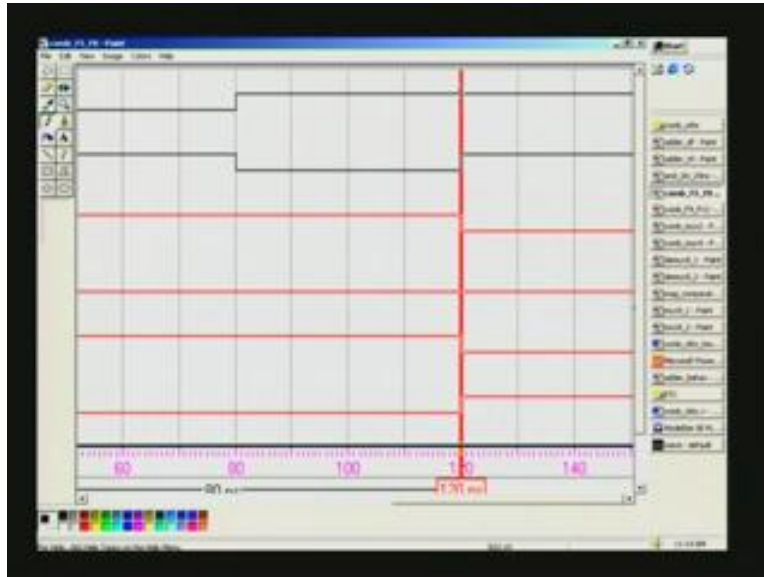
(Refer Slide Time: 29:31)



0 into anything is 0 and NAND means NOT, AND; so that means not 0, so there should be an inversion. This is 1 here and here also 0 1 so again 0, not 0 is 1, so here 1 0, so it will be 1. 1 will continue upto that 1, then it becomes 0 because both are 1, so 1 into 1 is 0. Then not 0 is 1, so you have verified NAND here. This is NOR, F6 is NOR is this waveform here and you can just see here.

0 0, not 0 is 1 and then 0 1 is 1, not 1 is 0 and that will be the same case here, all through for the rest because r is basic. r is 1 for all other combination and we have verified NAND and NOR.

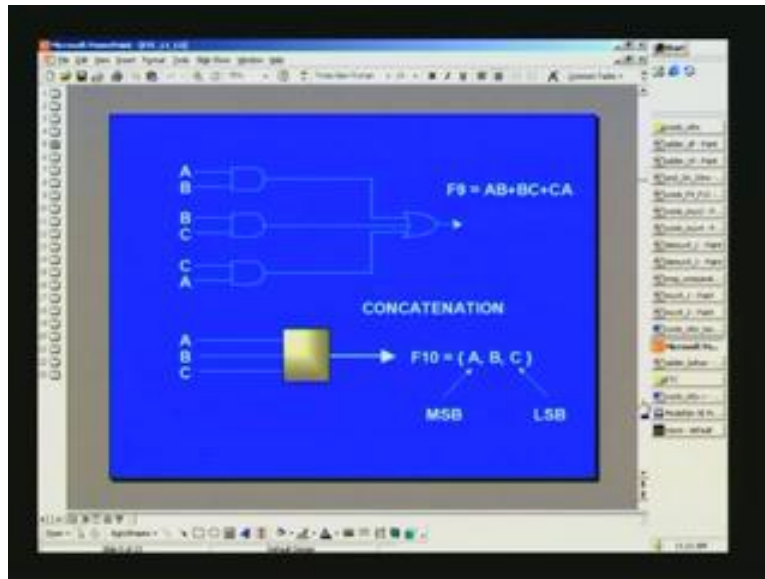
(Refer Slide Time: 30:32)



We will verify the EXCLUSIVE OR as F7 and F8 are EXCLUSIVE OR and here it is a 0 0 combination as the input here and you have 0 for EXCLUSIVE OR and EXCLUSIVE NOR is inversion of that. We can just see together and the next is 0 plus 1 here in this region and that is 0 and then 1. So, 0 plus 1 produces 1 there and not 1 is 0 here. Here also 1 0 also produces 1, the same as OR gate except for the last combination. That is why, you have 1 plus 0 is 1, and that is what is 1 here and this is exclusive NOR.

For this case, it is 1 1. In 1 1 case, EXCLUSIVE OR is 1 plus 1, 0. Actually the sum is first bit is 0, so we are looking at only the sum not the carry and that is what is here. It is 0. 1 plus 1 is 0 in EXCLUSIVE OR. Inversion of that one is not EXCLUSIVE NOR which is 1 here. We have checked up to F8 here. So, we will remove this window. The next one we have seen, we always statement in this thing.

(Refer Slide Time: 32:09)



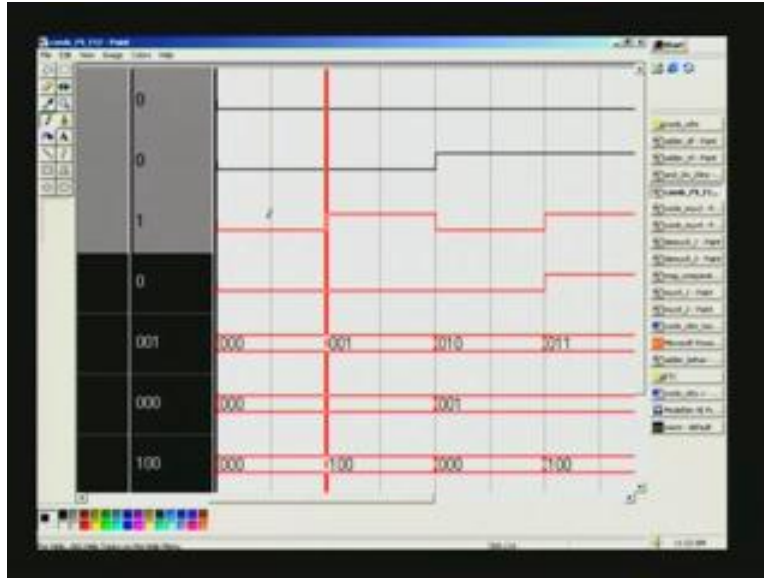
We have made majority logic as such, AB plus BC plus CA and the three inputs here and comparison also. A B C put in the same order, so we will see both here. This is F9. First we will see F9 to F12. Once again A B C are the inputs here and what it does you can see the assign statements also. That is what we have already seen here, always statement is here and this is what we are going to see here.

(Refer Slide Time: 32:09)

```
F9 = (A&B) | (B&C) | (C&A) ;  
  
// Realize AB+BC+CA.  
  
F10 = {A, B, C} ;  
  
// Concatenate A, B and C to get 3 bit  
result.  
F11 = F10 >> 1 ;  
  
// Right shift by one bit.
```

AB plus BC plus CA, then we get A B C.

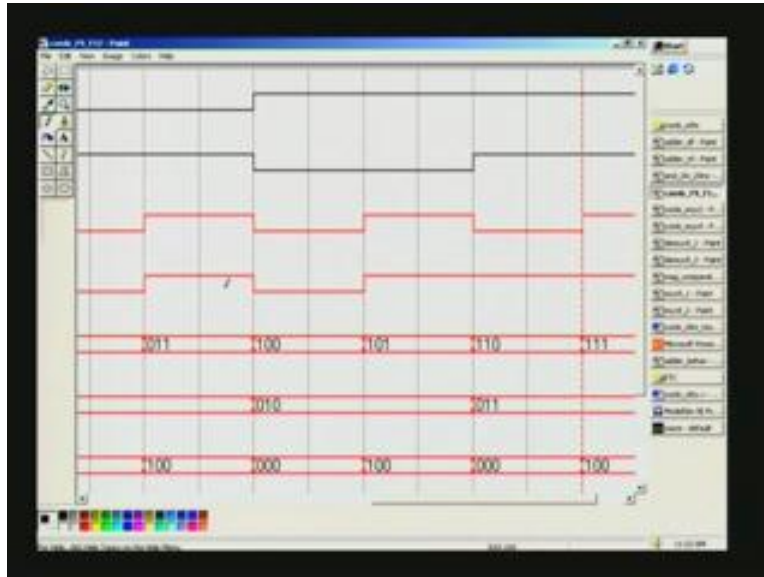
(Refer Slide Time: 33:04)



F9 is nothing but AB plus BC plus CA, so first three are A B C. Just have a look 0 0 0. All of them are 0. It is going to be in binary progression 0 0 1 then 0 1 0 and so on in a binary progression. That is what you have here first output. What is happening is AB plus BC plus CA, note that it is only majority logic. Two or more number of entries in this A B C will produce one output, for example here this triple 0 here and then 0 0 1 and here 1 0 0. All these should necessarily produce only 0 because maximum number of 1's are one only.

Beyond this, here you have 0 1 1 so two 1's are encountered, so it goes high that is what majority logic is about.

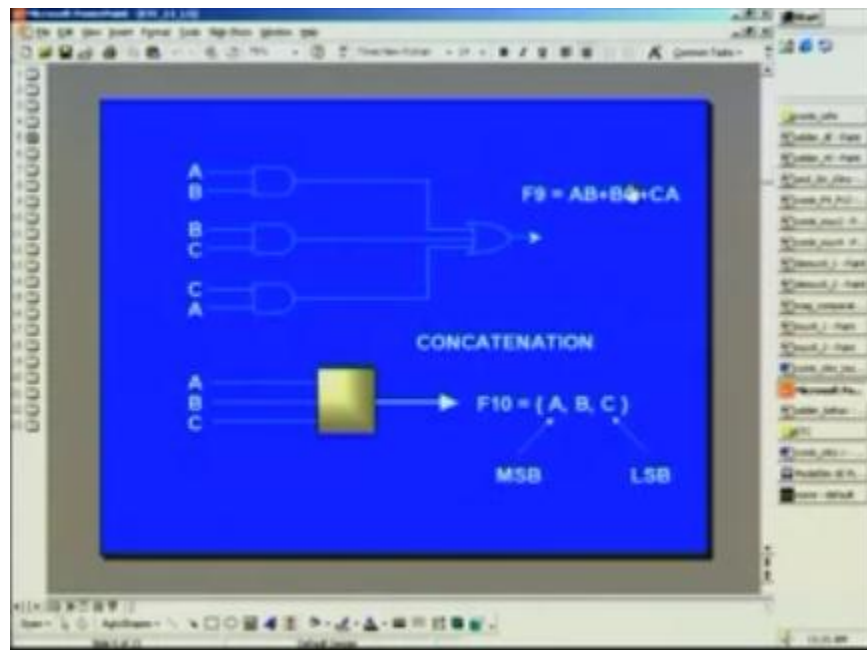
(Refer Slide Time: 33:57)



Here it is 1 0 0, so only one 1 entry is there, so it goes to 0 here and again here it is 1 0 1. So two 1's are there, it is going high and beyond this it will be high because it is either two or even three, for example 1,1 so it is 1 1 0. It continues to be 1 and here it is 1 1 and 1. So, two or more number of 1's will produce 1 output. Thus, we have verified the functionality of this majority logic and the second one was F9, next one is F10. What is F10? It a concatenation we have seen of A B C, right?

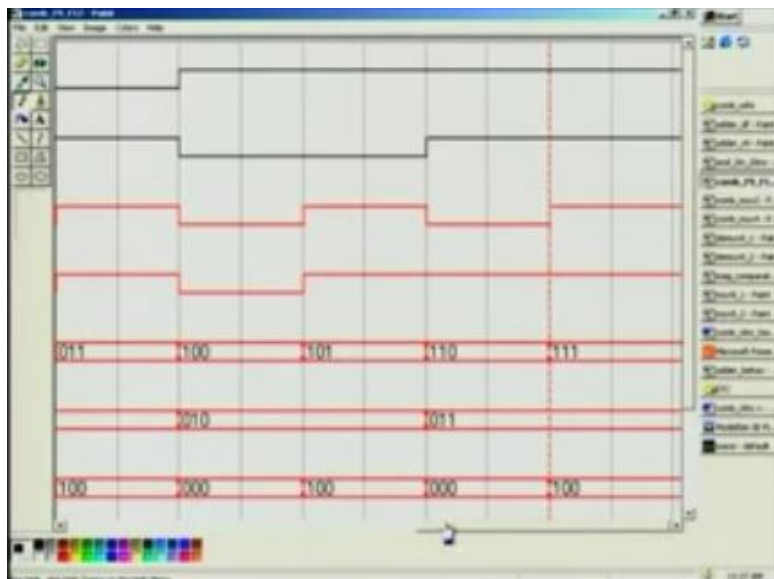
A B C is precisely this. It is in arithmetic progression we can say in binary progression like 0 0 0, 0 0 1 and so on. You should get precisely the same progression right here if you view this output. That is what you see here 0 0 0, 0 0 1, 0 1 0 because it is concatenation. Note that the first 0 on the left is A that is what we had here.

(Refer Slide Time: 32:09)



This is A and LSB is C that is what we had here and that is what you see here and it is all in just arithmetic progression so 0 1 1, 1 0 0, 1 0 1, 1 1 0, finally 1 1 1 as that is the end. Next there are two more outputs. What is F11? We will see here.

(Refer Slide Time: 35:48)



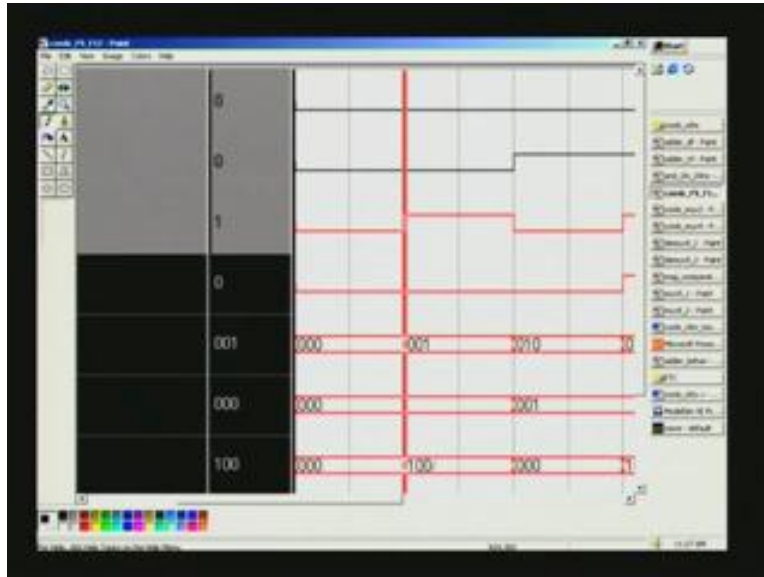


F10 and F11 are there. F10 is the input, this is the right shift register and it shifts by 1 bit; there is also a left bit by 2 bits here and we have seen earlier this source is F10 for both the cases whereas the output is F11 and F12. F11 happens to be that A B C is basically what we have here and A being the MSB here. When you shift right by 1 bit, what will happen for F11? We have to see. Similarly for F12, but in this case, 2 bits shift. Let us see that once again so that is F11 is right shift by 1 bit, so what should be shifted is actually F10 which is being shifted and that is the input we have seen.

Let us see what happens here, if you right shift here this 0 will get lost and vacated bit 0 will always be filled so it continues to be 0. That is why the result is 0 here and again the input is now in this case 0 0 1 and if you right shift here, what will happen is this 1 gets dropped and 0 fills the MSB. If you continue to have only three 0s that is why it is all same now you see the difference, 0 1 0, if you right shift by 1 bit, this 0 will get dropped. What? 0 will get filled on the MSB. What you get is 0 0 1 that is what you have here and you can straight away explain in same manner all these. (Refer Slide Time: 33:57)

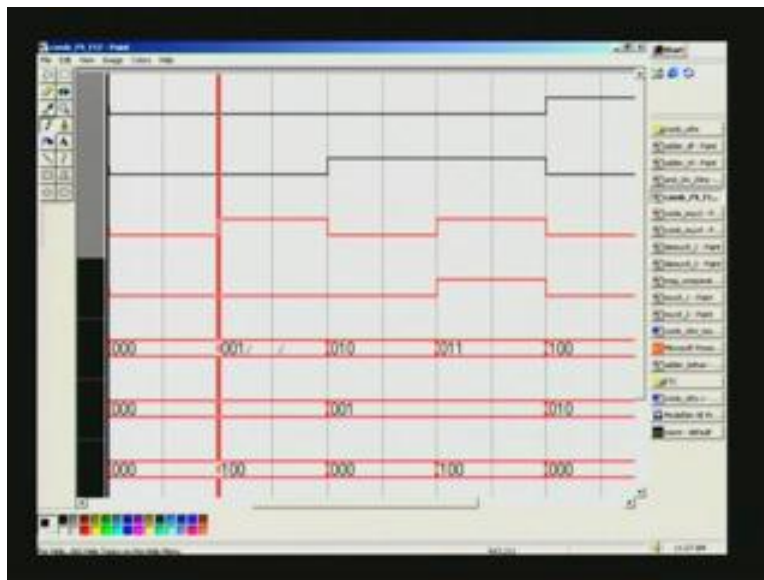
1 0 0 when you shift here, this 1 0 shifts here. Similarly this 1 0 continues to be here, so this 1 will go to the last bit and MSB again filled here. Now the last one is 2 bits left shift here, so we started with F10. When you shift left, it appears to be same here; 2 bits now we have to shift. Let us see this, see if you shift 2 bits, this 1 will go to this position first bit for first shift and for second shift, it will go here and bits vacated will be filled by 0's. So, it will be 1 0 0.

(Refer Slide Time: 38:13)



So, you can inspect every other combination, for example this one.

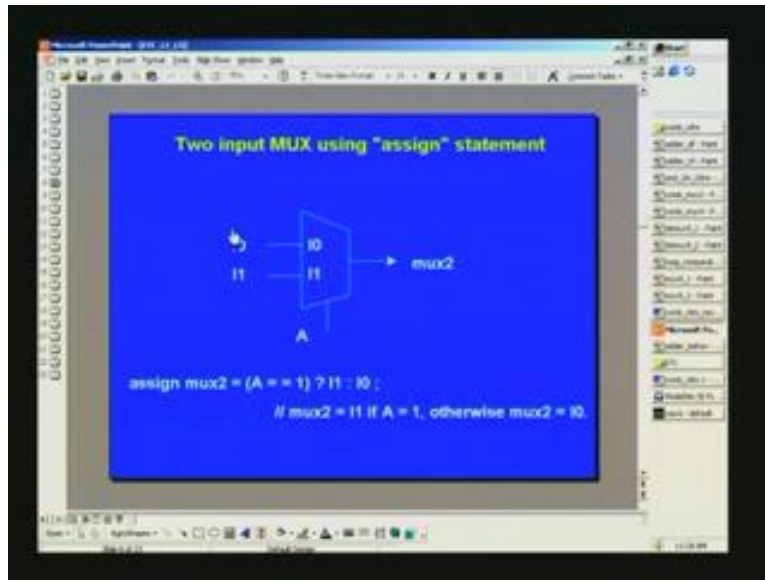
(Refer Slide Time: 38:19)



There is a 2 bit shift, so 1 is lost. It becomes 0 here and again this 0 1 will get dropped and only this 1 will survive here and so on. So, you can very easily. Finally, you can see this 1 1, 2 bits, this 1, two 1's get dropped and what remains is only 1 and 2 bits vacated are 0's here. This will complete up to F1 through F12 of the combinational circuit and

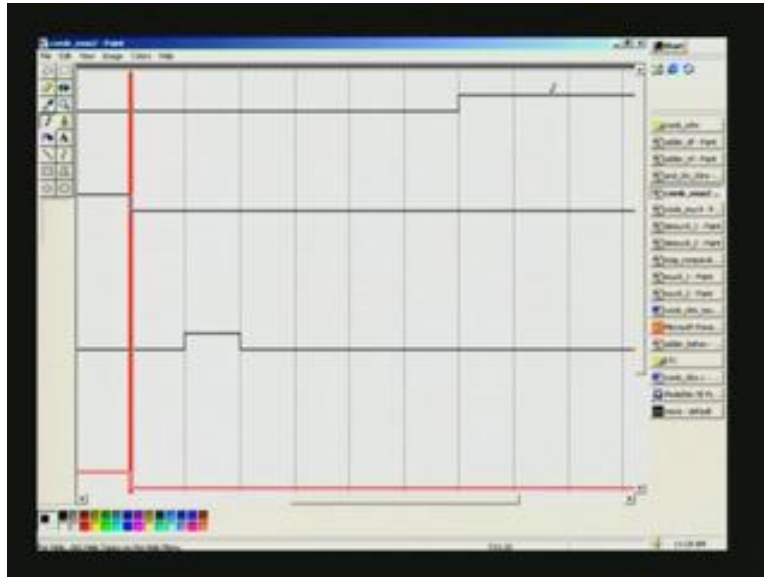
next we have seen a mux there, two input mux here and that is what we are going to show here.

(Refer Slide Time: 39:04)



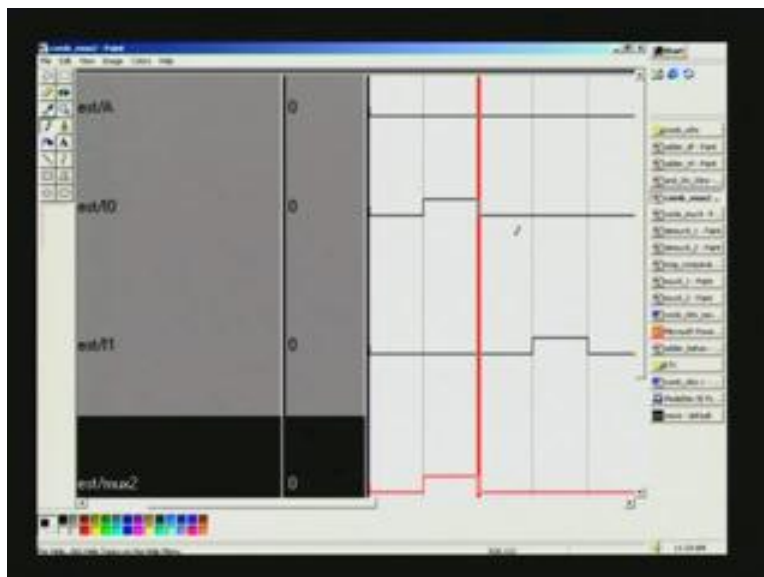
In a mux what we have two inputs, I0 and I1, and there is a select control A and mux2 is the final output. The assign statement which did this operation was when A equal to 1 then select I1 and then output it to mux2 as such. Otherwise, you take I0 as mux2 and this is precisely what we have here in this final output as such. Here A is the select control I0 I1 and mux2 is the output that we have already seen just now and we will remember A is I0 is the 1 which will be assigned if A equal to 0 that is what is here you can see.

(Refer Slide Time: 39:49)



This is 0. All through it is 0 here, as far as A input is concerned which is select input and here is I0 input as such. This is the input that we are simulating through test bench which we have already seen here. Every 10 nano second, we made it to 1 and then made it 0 and so on in the test bench. I0 and I1, it is having a look; I0 is applied here and I1 is applied here.

(Refer Slide Time: 40:23)



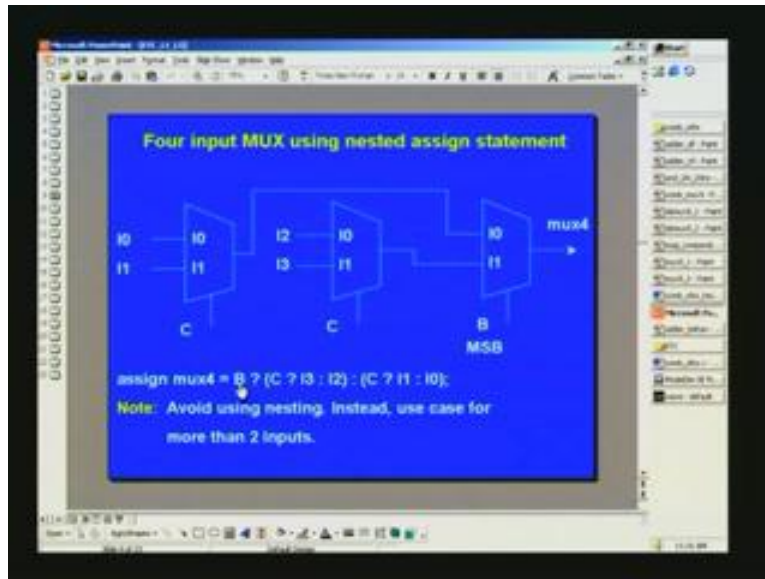
The output must reflect either I0 or I1, depending upon the select control A and here this A is 0 in this position. That means, we are going to select I0 and not the I1. I1 will be selected only if A is 1. Whatever is the input waveform here, this must appear at the output and that is precisely what we have here right at the bottom.

Beyond this, it is continuing to be 0 as far as A is concerned, therefore this pulse being low here, beyond this, it continues to be low here. Now, one unfortunate thing here is I should have made this 1 here, but if I make it here for other examples, you cannot see it clearly. This being a simpler case, I thought I will sacrifice here and show you that waveform in the next subsequent things.

What will happen here since A is 0 all through what you see at the output is I0 only again because I0 is 0 here, you see only 0 and this waveform is not coming that's what we have explained just now. This waveform will be taken into account, beyond this but you cannot see it unless the pulse was here. I could have written the pulse here or made the other way and this you will be seeing in the next example. The next example was to configure mux4 that is this (Refer Slide Time: 39:04)

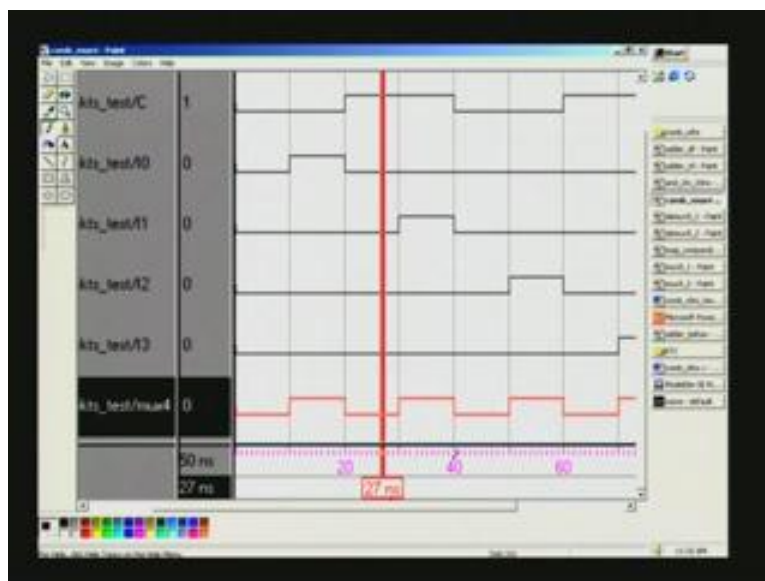
Here I0 through I3 are all the inputs now. BC is the select pins here and mux4 is the final output. B is the MSB here and we will see the waveform corresponding to mux2 and see that all the patterns are intact for this. We are trying to use, no matter what outputs are created, we are trying to use the very same inputs A B C repeatedly we use right because in the actual application design the all these designs will be interconnected.

(Refer Slide Time: 42:12)



Some of them will be used for some other purpose and very rarely few outputs will be energized and not utilized inside. This combination circuits are Sequential Circuits. Although, we have put together all different simpler modules, they are not the total project on such. It is **piece meal** thing. But it gives an appearance as if it is a total project.

(Refer Slide Time: 43:33)



A total application project is not much different from this, so it is only to illustrate and demonstrate actual simulation that we have put it together here. Let us see this mux4. Whether it works and in one shot, we can find out if BC is the input here. It is once again in binary progression, 0 0 here up to this is 20 nano second. All waveforms are marked by 20 40 60. Here, it is 0 0 here then 0 1 1 0 1 1. 4 combinations only are there because BC is only 4 combinations of 2 bits means 2 to the power of 2.

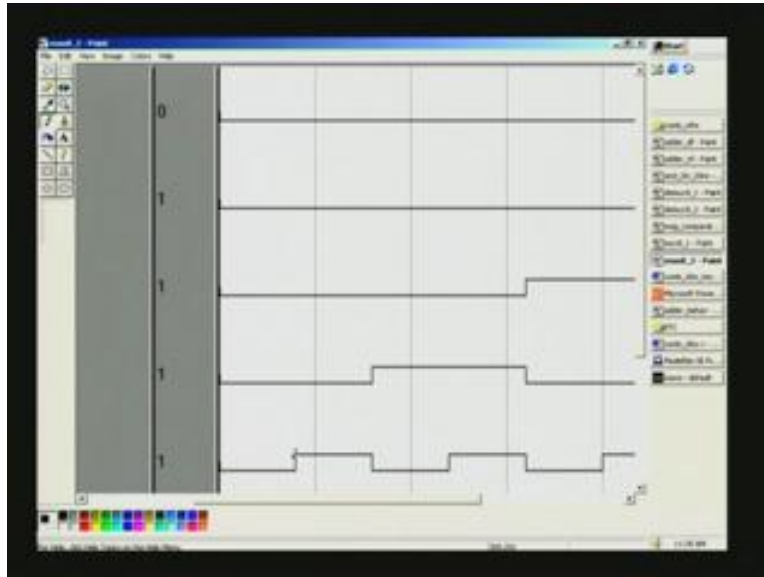
Now, let us say for 0 0, if this is select pin for the four input mux.. Naturally, there are four inputs are I0 through I3 here and I0 is the very first thing; it corresponds to BC being 0 0. For 0 0 input, I0 must be selected that means to say this mux4 output should be very same as I0. You can precisely see the very same waveform here and no other waveforms are there in any other inputs. You can very easily infer that this mux4 is clearly due to the input I0 here that is precisely what you have here.

In the next, when BC changes to 0 1, what you have is I1 appearing here. You can just see here and mind it happens right at this stroke of change, C is changing here, right at that point of time, it switches to the second input. What you see right from here till the end is precisely the I1 signal here that is what you see here and so is the case for I2 here which appears so you get the output corresponding to that. Finally for the last one, this is I3 as such and this is corresponding to B and C being 1 1, you can see this is 1 and this is 1.

So we have seen two input mux and four input mux and we will see eight input mux also. Now, we will see mux so in eight input mux how many controls totally will you need how many controls? This being a large waveform, it has been split into two here, and unfortunately this A B C appears only in the second waveform. But it is not a difficult thing to remember because we go only by the binary progression here.

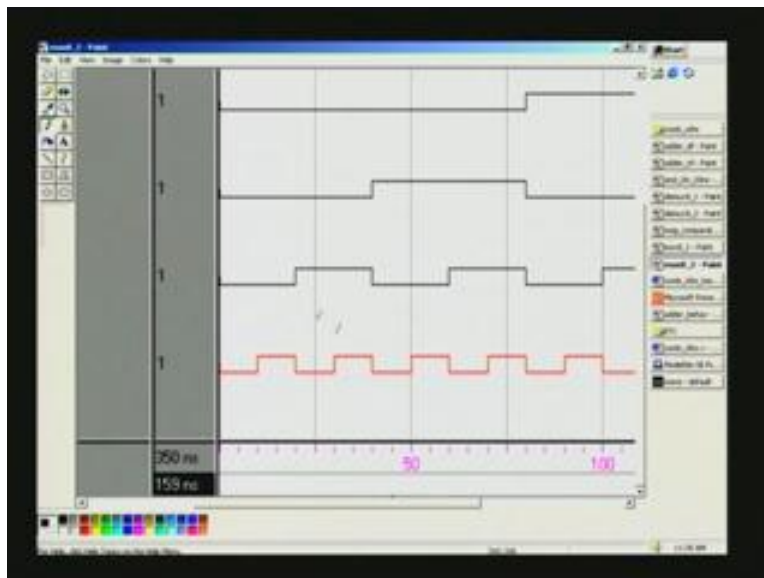


(Refer Slide Time: 45:58)



The only thing is to remember the pulse has been created at 10 nano seconds and interval is 20 nano seconds.

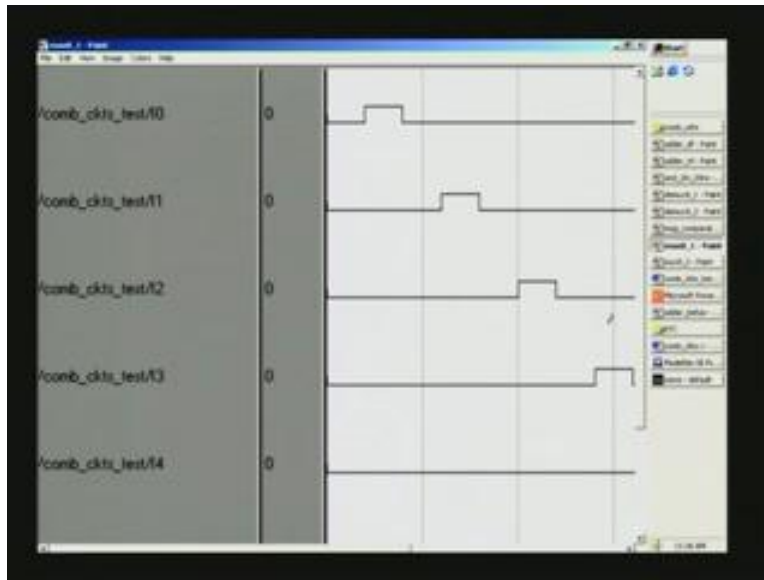
(Refer Slide Time: 46:05)



Now, the time base has been changed in order to accommodate all the combinations possible and that is why the time base is different. What matters is the actual waveform

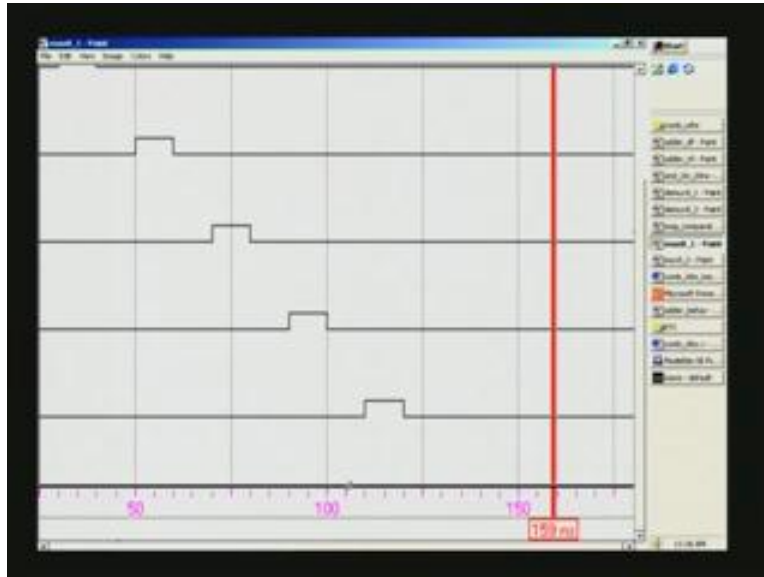
whether it is appearing. In fact we can see the waves here, this is mux8 here depending upon A B C. If it is 0 0 condition, that is here.

(Refer Slide Time: 46:44)



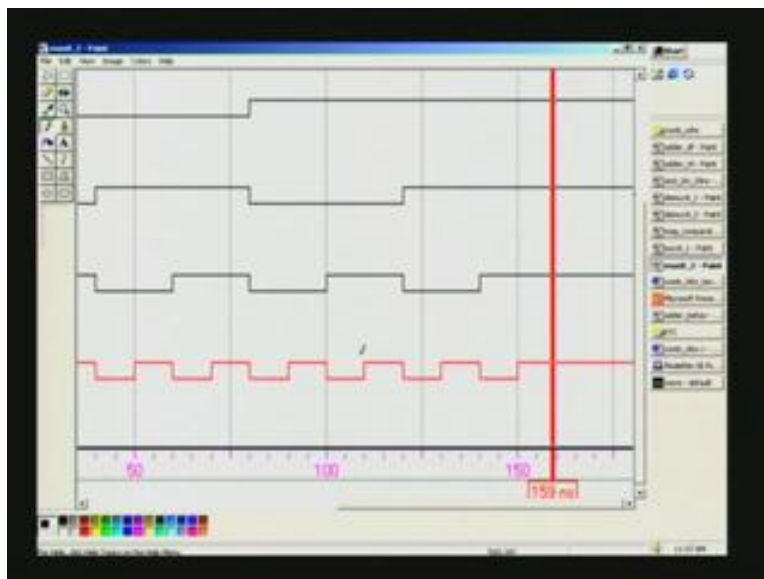
You should get the mux8 input from I0. Let us see whether I0 has pulse there and you will not get it here because it is there in the previous one that is in this. I0 is pulsed here and I1 is pulsed here. It is all subsequent. You do not have to go back to this file again. You can see every time it happens. And I4 up to I5, it is all sequential up to here, in the same order it goes. We will just see in the next file and here also I6 where does it goes. This is completed in this waveform.

(Refer Slide Time: 47:15)



Beyond 100 nano seconds just remember this is corresponding to I5. In this you have I6, it must be somewhere. It is coming here right and that is precisely what you have here at the output here see mux8. This was the very first pulse of I0, then I1, so that is precisely what you are having here right. A B C are the selector inputs and you can see for example, these are all A B C 1 1 0 here. So, 1 1 0 should have given 120 or something. We add this I5 to that.

(Refer Slide Time: 48:05)



This must be I6 which belongs to this precise input condition. Let us see whether there is any right. That is the one, this is I6. This was I5 and I7 is the last thing; this is corresponding to this in the test bench. I would have forgotten to make it low, so it remains high corresponding to this A B C are all 1.

Thank you.

(Refer Slide Time 49:05)

