

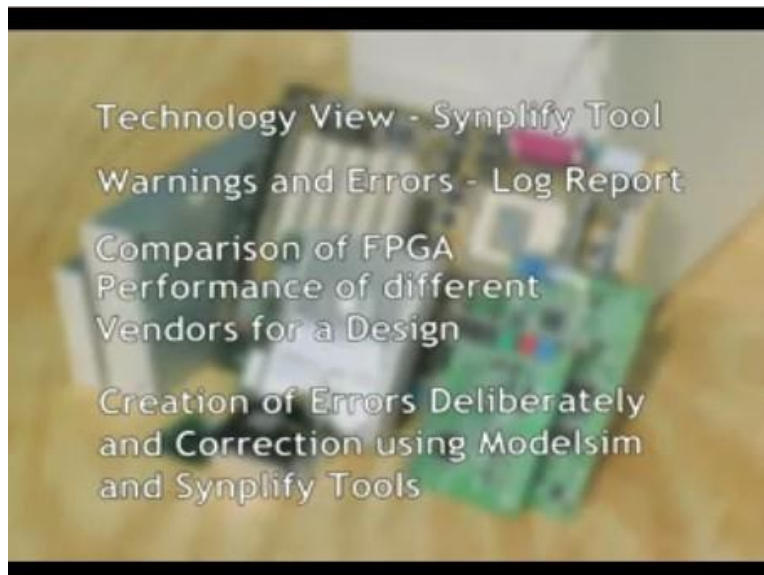
**Digital VLSI System Design**  
**Prof. Dr. S. Ramachandran**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Madras**

**Lecture - 33**

**Synopsis Full and Parallel Cases**

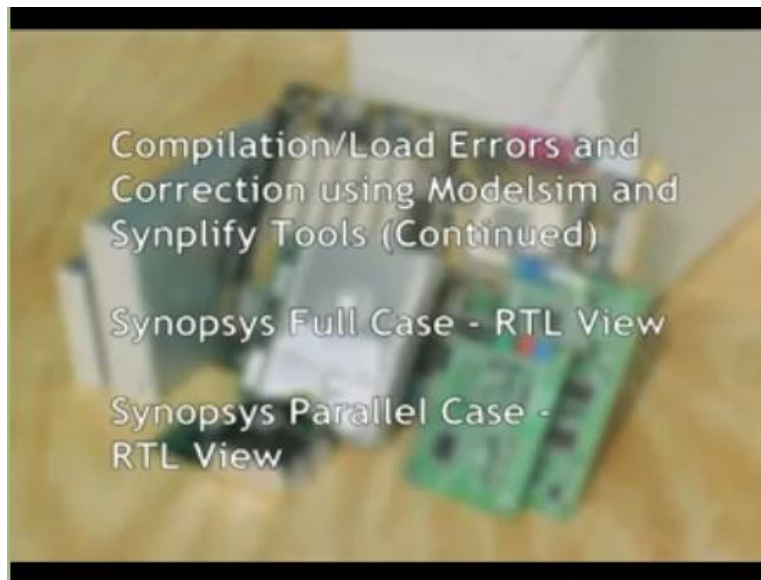
Lecture 32: Technology view using synplify tool.

(Refer Slide Time: 01:08)



Contents of Lecture 33:

(Refer Slide Time: 01:42)

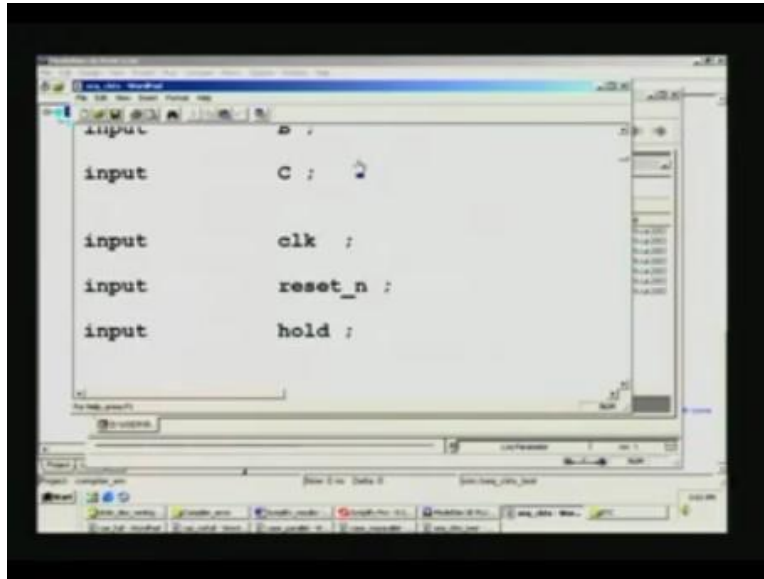


(Refer Slide Time: 02:07)



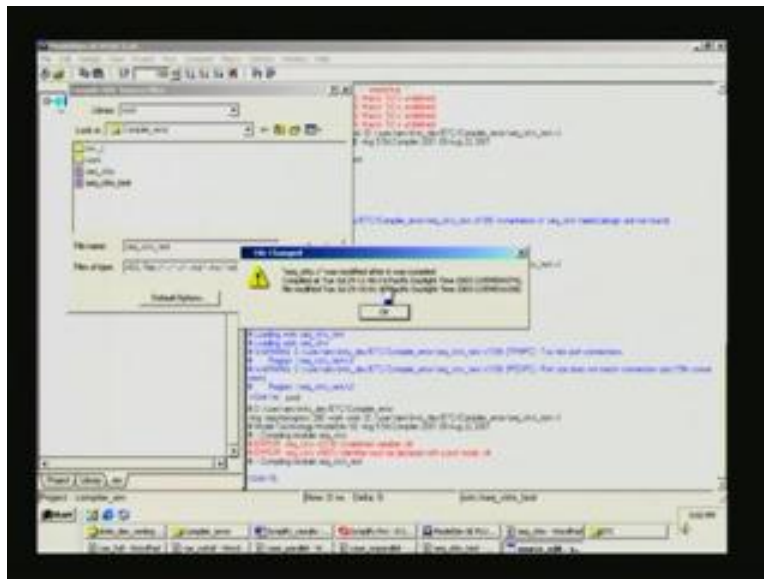
We created deliberate errors in the last class to see how synplify tool and modelsim tool behaves. The way it reports and how it guides us to correct it.

(Refer Slide Time: 02:26)



We will continue with some more examples of that kind. Let us say this is the source file, sequential circuit. Let us make an error here. Let us change the name of the signal. I will just change it as clock; let us observe how it reports.

(Refer Slide Time: 03:02)

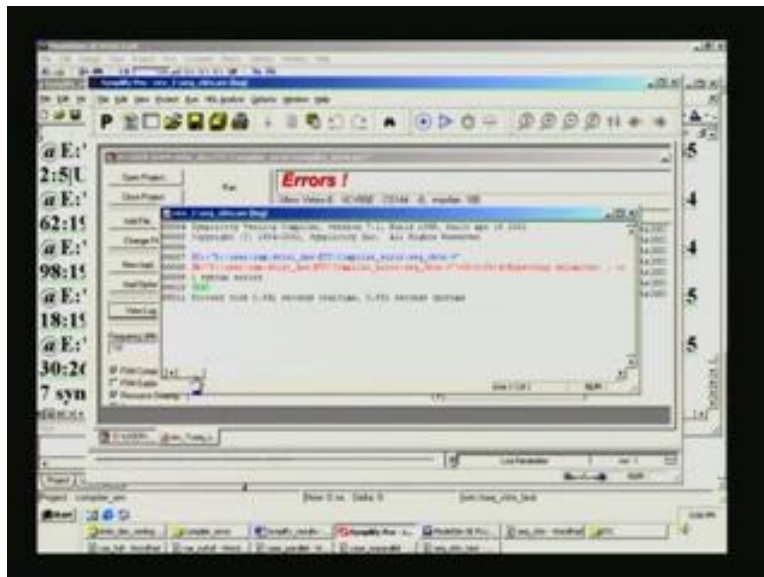


We will start with the modelsim in order to do the compilation we have to design compile take the test bench not to compile. Let us see it has reported some error. We change the name of clock it promptly says that undefined variable clock is encountered here. This is just a warning saying

that I have changed the source file. You can just ignore that. It opened the source file is pointing precisely to the place where CLK is used because this is an undefined variable. I have change the name right at the input as C L O C K. That is to say that CLK is undefined. We had to declare input this follows from that is what the errors are. There is one more error here this is line number 214. This is 214 this is 601. One more error is reported here 601 towards the end module it is reporting. Again it is the same CLK because towards the end of modulates in the habit of saying that once again the same thing here we will revert back to that old one.

We will create one more error of a different kind. For example, we have been using semicolon all along let us remove one semicolon. Let us see what happens. After input reset we have removed the semicolon here, compile once again. Again it reports error here, what it says is it is pointing to input hold which is the next in line with a reset here. We have removed semicolon here but it is pointing to this line say in that error is encountered here because just before processing this line this one was absent it cannot distinguish between one statement another. That is how it points out always to the next one, just remember that. Most often all beginners forget semi-colons are very usually encounter errors.

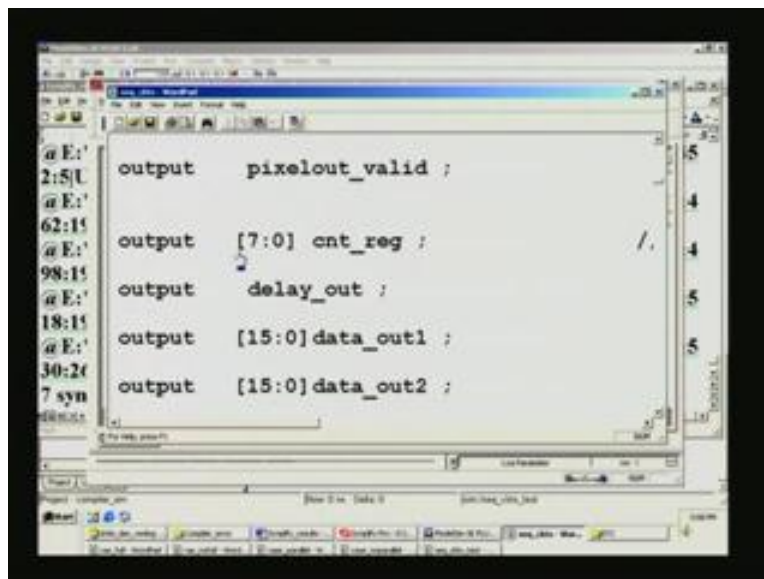
(Refer Slide Time: 06:34)



We have not seen the corresponding simplified thing. Before that we will see here. So it is reporting errors here and if view the log file you will see that it has done nothing beyond that

because very first shot it has encountered the error therefore no further report is seen within this log file, which usually is a very long file. I can see that error here. This is at e is reported and what it says is expecting delimited comma or semi colon. If you double-click that it opens that source file and once again it is pointing to the very same thing that modelsim had pointed. This is input reset underscore n and this is hold. Note that this semi colon is missing here and in both the cases it is precisely the same and earlier error would not create for the synthesis tool, we will do that as per. We will restore this semicolon here clock we had oc introduced that. We will run this again. Once again it is reporting error. Let us see where it is it is once again same except that this time it is reporting differently, signal clock is missing from port list that is what it says here. If I double-click it goes precisely to the same statement input clock, it is highlighting here. So, simplified tool is marginally better in reporting compile errors.

(Refer Slide Time: 09:05)

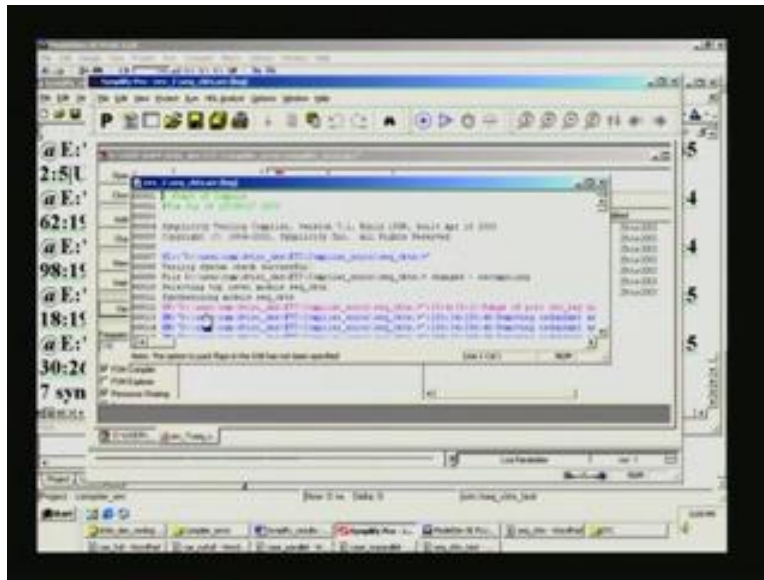


```
@ E: 2:5U output pixelout_valid ;
@ E: 62:15 output [7:0] cnt_reg ;
@ E: 98:15 output delay_out ;
@ E: 18:15 output [15:0] data_out1 ;
@ E: 30:26 output [15:0] data_out2 ;
7 syn
```

We will create one or two more errors. For example, far we have been dealing with only is single bit precision signals. We can have a multi bit precision signal. To see whether any changes are there, for example this counter reg is 7 through 0. Let us remove this and see what happens. Just as an indicator that we have created the error here we are just putting there. Let us see this synplify run that, it is reporting error, sequence signal clock is missing from port list. Is that the error that we had? That is not the one. Let us find out why it is that. It is because we have not

corrected here. This error you should deal only with one error at a time, so going back to synplify pro, let us run again.

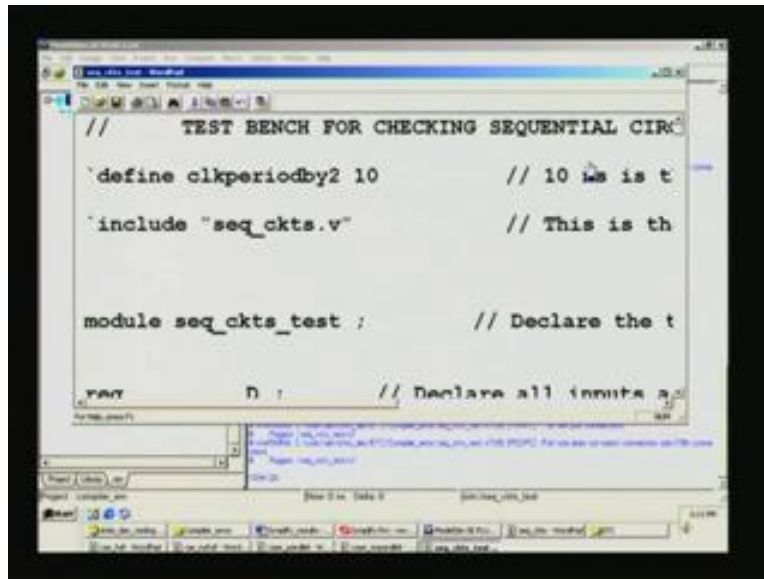
(Refer Slide Time: 10:52)



It has not reported any error this time but warnings are reported. Let us see what the warning is. This is the warning here and the warning says range of port counter reg in port declaration and body are different. At the beginning what we did is, for the count reg should have been 7 through 0. So we removed that one but elsewhere we do not remove. For example the same thing is declared as counter reg. Let us find out where it is (11:50). You can just find out from here that is one. It should have been a change here also. Earlier we changed; it is miss match that is what its reporting right. And if you do this same thing in modelsim, what will happen is, it does not report any error. Perhaps it will report at the time of loading. It is reporting warning here. What it says is at two few port connections that is what it implied and it is also reporting this, I think we have already seen earlier. All along we have been looking into the design file. So let us have a look at the test bench we will create one or two errors there also get a feel of it.



(Refer Slide Time: 13:08)



```
// TEST BENCH FOR CHECKING SEQUENTIAL CIRCUIT
`define clkperiodby2 10 // 10 ns is t
`include "seq_ckts.v" // This is th

module seq_ckts_test ; // Declare the t

VAR D : // Declare all inputs a

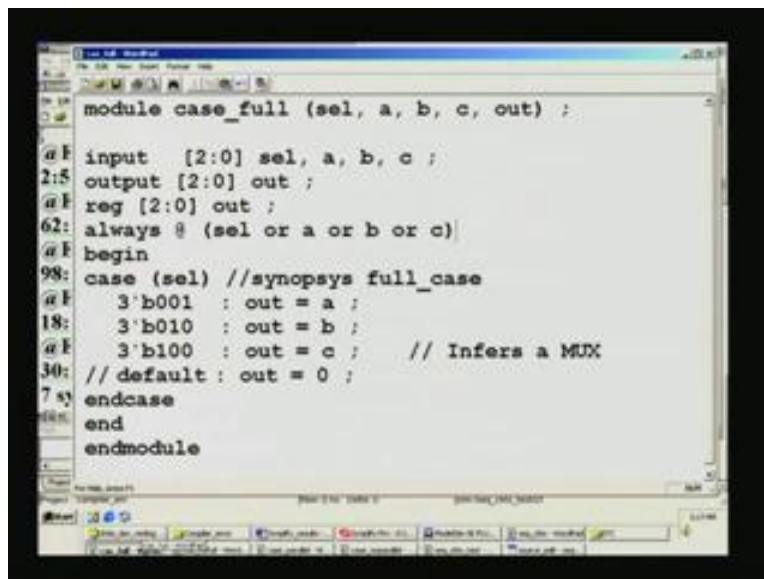
// Error: illegal reference to net 'D' at line 153
```

Test bench is here and so we will not repeat what we have already seen, for example, define etcetera module. I was saying that inputs or outputs are differently declared here it is reg scrap wire normally. Let us declare this as wire find out. So modelsim is reporting error here illegal reference to net, double-click the source will open. We change for D it is supposed to be reg but we deliberately made it to wire it is promptly reporting here. All of them are at various points of reference at 4 places it has been called. The error says it gives the path name of the file up to this and it gives the line number here and this is line number 153. It is reporting there because D has been referenced. It has been forced to one here because it is illegally declared as wire instead of reg.

It is promptly reporting what is says after the line number, it is illegal reference to net, then colon D. D is the signal underscore n. One thing should be very clear. Before we wind up, this error creation, so we will consider one last example let us go to the very end and you see here. Another point is once you have done all this, it is good practice for you to copy relevant details form the synplify results, place route results into the same source file as a ready reckoned for later use. You can very easily know how many get counted and has taken this particular design. You can just make a copy that is what as shown here along with that you can see all that. Many people after writing long codes, they invariable forget one n module. Let us simulate that also. I just commented on it this equivalent to know let us see what it does fix reporting error once

again here near module syntax error. It does not really throw much light. Guess on that let us see the synplify. It would not work for synplify thing because you are on the test bench; we have made the change only the test bench. I think with this confident any error that comes across tackle all by yourself. So the best thing would be to copy the codes I which is working namely if this codes are available to you we can make a copy into a folder then experiment that a file rather than temper with the original source code. We will move on to RTL coding guidelines. We have covered synopsis full case, n synopsis parallel case etcetera. We will take off those files independently see how optimization really works. Let us have a look first source code for the full case.

(Refer Slide Time: 18:38)



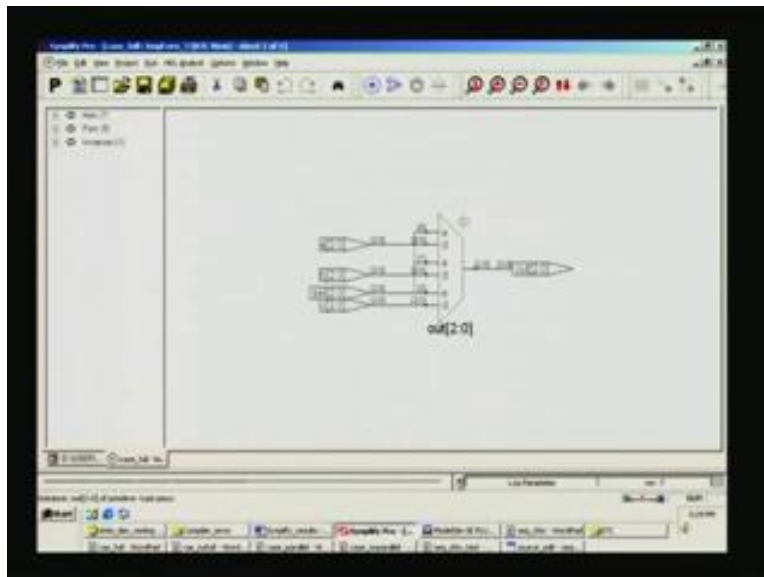
```
module case_full (sel, a, b, c, out) ;
@F input  [2:0] sel, a, b, c ;
2:5 output [2:0] out ;
@F reg [2:0] out ;
62: always @ (sel or a or b or c)
@F begin
98: case (sel) //synopsys full_case
@F 3'b001 : out = a ;
18: 3'b010 : out = b ;
@F 3'b100 : out = c ; // Infers a MUX
30: // default : out = 0 ;
7 sy endcase
end
endmodule
```

This is called the full case here and the program is quite simple and what we have here is three inputs ABC one select pin is also there. It is basically case implementation. The synopsis is the special thing which was mentioned earlier while dealing with RTL coding guidelines that is after comment use synopsis then place then full under underscore case. This is a reserved word although it is looking like a comment. So you cannot put comment which resembles this or instead of this you cannot put a parallel case also two are reserved. What is the specific advantage in having this? It will be having a look what this code supposed to do is simply assign A B or C to out depending upon the select pins for example puts 0 0 1. We have seen this for one hot machine. So in one hot machine there will be only one entry one you can see 0 0 1 only one



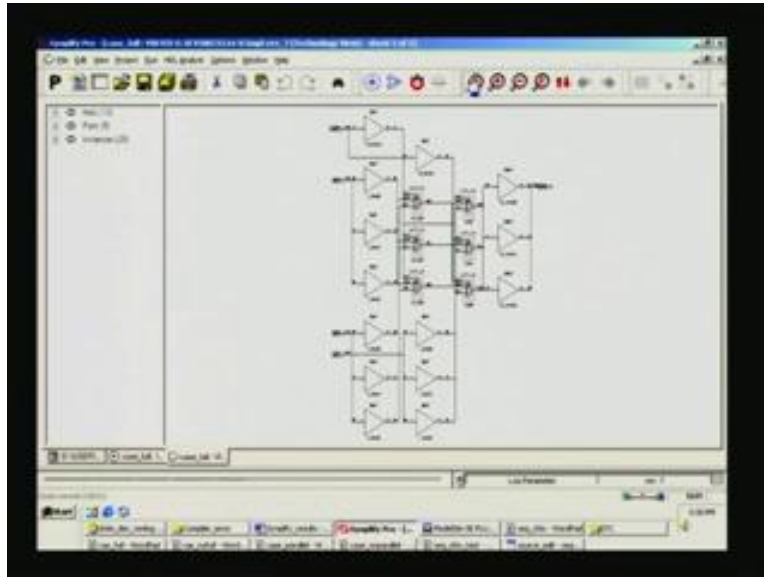
entry one here one here. In this case, default is not used we will see it later on. The whole thing infers a MUX; this also we have seen it is also applicable for one hot machine. Let us run this code that is all the programs for this. We are not going to invoke modelsim for this only this synplify. The modelsims, I leave it an exercise for you to do. We need to create a new project let us say will not see the synplify results for this full case. This has been done prepared already for you for all the different cases. So right now we are in full case let us view.

(Refer Slide Time: 21:17)



RTL view this one and you notice that there are only MUX'S inside here as mention before D stands for data, E for enable what you have here is A B C two through 0 bit. That is 3 bits each the select which is also 3 bit. In fact the select distributor all through as 0 bit here, then 1 bit here, 2 bit here. In fact they are all inside basically two input MUX'S which we can see by technology view.

(Refer Slide Time: 21:53)



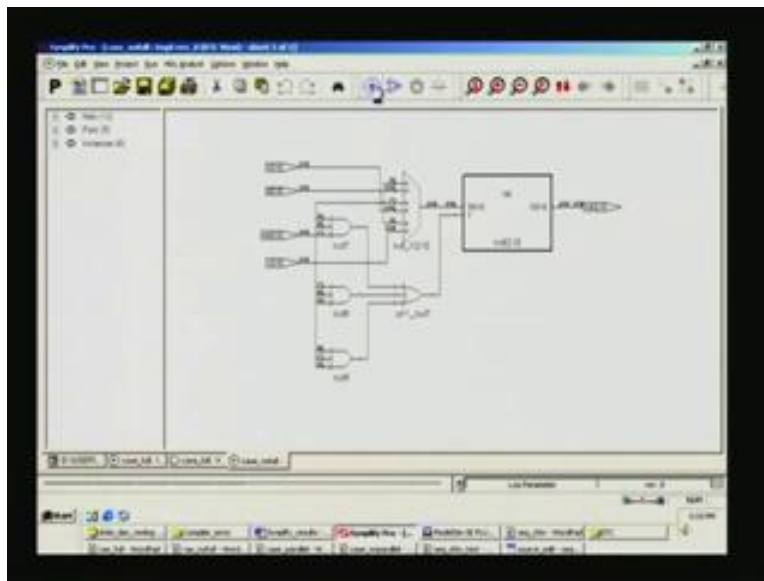
This is what it is. For example, let us say we zoom it here we can see I buff select two 0 use here all inputs appear on the left here through I buff it goes to different MUX here basically they are all 3 inputs LUTs use finally there are outputs buffers here. When you zoom it down you can see it very clearly. Basically there are 6 LUTs functionally there are 2 input MUX as you can see here output buffers are this then input buffers here.

(Refer Slide Time: 23:22)

```
module case_nofull (sel, a, b, c, out) ;
input [2:0] sel, a, b, c ;
output [2:0] out ;
reg [2:0] out ;
always @ (sel or a or b or c)
begin
case (sel)
3'b001 : out = a ;
3'b010 : out = b ;
3'b100 : out = c ; // Infers a latch
// default : out = 0 ;
endcase
end
endmodule
```

I compare with another case, for example, this is the result of what we have been seeing is full, which uses synopsis full case there is exactly this same thing we removed that synopsis full case call it the snow full case. Let us see otherwise the code is exactly the same only difference is that we have not we have removed the synopsis full case from here. In this case we also mention earlier that it infers some latch, let us run this in synplify find out whether it is so. We will just minimize this, this is the view this also will minimize. We will go to no full case here that is here the corresponding.

(Refer Slide Time: 24:28)



RTL view is this. What we had earlier actually this one now this is for synopsis full case; whereas, for no full case you see that in addition to this extra gates have appeared here. 3 plus 4 gates plus one latch is also there. Latch is detrimental for making real hard work. We have pointed out earlier during the guidelines. In fact that is what it is see here synthesis tool does not remove this as such but it gives warning. I hope that is a warning message corresponding to that. But all of sudden I can just see just by giving that synopsis full you sales so many gates including the latch. That is a quite substantial saving. If you can view once again it is same 2 plus 2 4 then 6 here one extra come over here this count input buffers output buffers may essentially remain same. Only this has made a difference plus you see a latch here. All LD are all latches three additional latches have been created. Only if you look into the technology view you get the real story behind that. So far we have seen case full case no full pointed out that we can

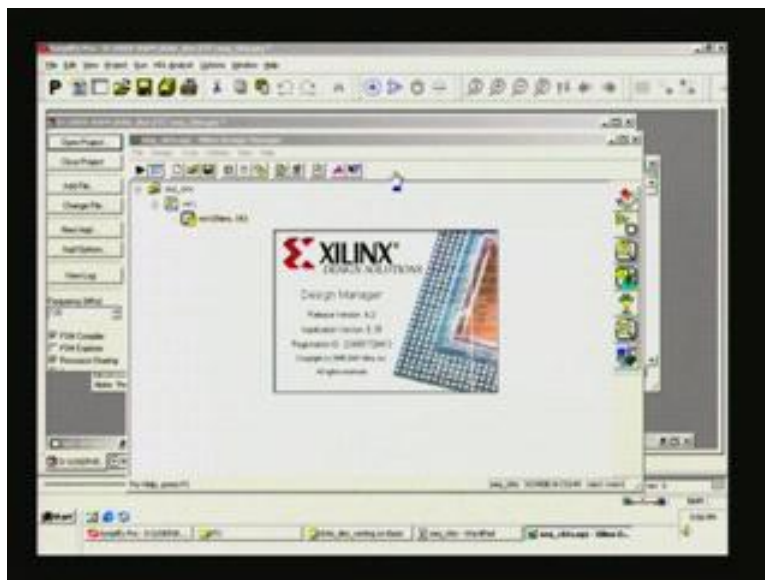
affect substantial savings just by that directive. Similarly, you can have a parallel case no parallel case the parallel cases once again similar to that full case this be used in a slightly different sense here. For example select a, b or c any of this combinations, I mean inputs if it is encountered then force to a particular value. It was different the last full case here it is slightly different. This is once again here notice the difference. Here it was select earlier but know it can be even a b or c. If it selects equal to a then it will come to this point similarly for c if select c then it will execute this. In this case this is called synopsis parallel case you notice that once again these directives important that should be no comment of this nature here parallel case means no priority. We will go to this parallel case this is parallel case. All this have been run kept ready for you so that we are merely viewing that.

This a parallel case what all it means is when you look at the code, select is equal to a then out is one so on with reference to select the actual inputs is here if you see this is this is c this is b. This is a, all the three bits similarly this is select a c are selected and compared here that is why equal to symbol straight to away they out the set. The output that is governed by the code that you have already written that is here. This is the not playing there is the case for select equal to b select equal to a so primarily this. This is the technology view once again you see all the inputs have I buffer then outputs o buffer then you see LUTs. Let us have look at the no parallel case. The no parallel case code is exactly same expect that this synopsis parallel case have been removed. That was what we had here synopsis parallel case in this case only that is removed otherwise it is same. Now let us look at this. This is the parallel case for which RTL view is this. Earlier case it was only three equal to other now it has created additional gates, three gates are extra here.

Earlier it was only parallel case. You can remove from here. It was this now in addition to that three more gates; that is the difference. Once again this technology view will reflect exactly same thing except for details here which you can see. As usual you can go to that to get all these signal details here. You can come up again here also you can see signal details by removing this assumption. It is better to use this and then go into that. What we have seen is no parallel case here we see that here also mean extra gates are put in. So I will leave it as an exercise for you. Suppose you remove this comment run. Notice the differences as for as the mapping is concern, whether it takes more gates so on. You can uncomment this run it also for full case. We covered

modelsim for simulation synplify for synthesis. We have also seen in synplify that it produces an output .tdf file with the same design name that you have specified. It should be input for the next stage which is called a place route will be taking only a Xilinx place. I am not demonstrating this total design flow here we have the synplify window opened. We have earlier used for sequential circuits one of the revisions is here which you saw in RTL view of sequential circuits. How do we invoke the Xilinx place and route learn about the tool. We have options here click on that options then you have Xilinx once again within Xilinx move on to this. You have to start the design manager, start floor planner then start isc project navigator. In Xilinx place and route design managers do not seem to become the recent revisions only project navigator is available. I am going to show you on the design manager because you can get more feel then the project navigator. It should not be difficult for you get design manager to project navigator. What are you need is couple of floors of a familiarity.

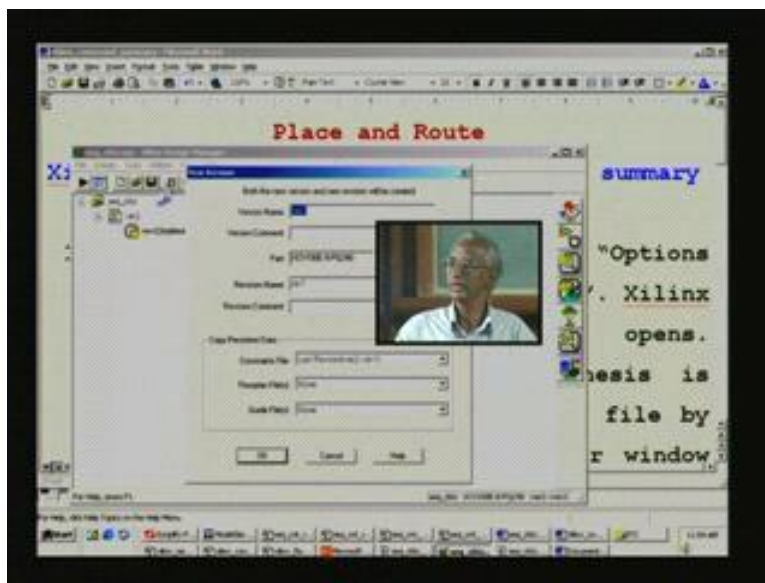
(Refer Slide Time: 33:54)



We will start the design manager by clicking on this. You see at the new window of power point this is the Xilinx place and route. Now what you see here is we have the input for this xyling place and route is dot edf file of sequential circuits. Since we invoke from synplify, it had automatically open the relevant design file. It has also created a version here version 1 and revision 1. As mentioned, to avoid over writing this tools normally they do created different versions revisions. That is how you have here. Now, next step here is it the space and route

already knows that the input files that is dot sequential circuits dot edf, this will be taken as input file and processed further. You have to do the implementation. Click on an arrow here. It also says implement here otherwise from file also I think you can have or design can have implement there also you can have you can start the place and route by clicking on this arrow button here. Prior to this we will create one new version because revision 1 has been already implemented. So we will start a new version click on design then you see here new version. Let us go for new version, just click on this.

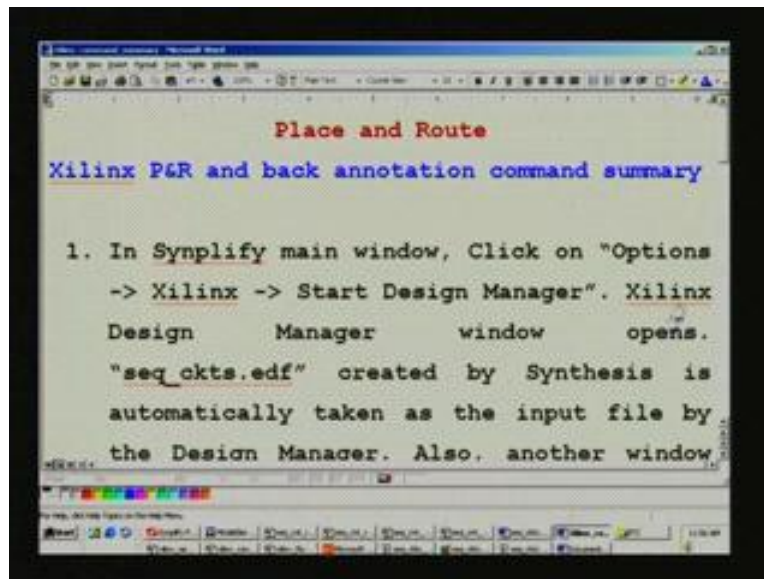
(Refer Slide Time: 35:43)



New version window pops up. Earlier it was version 2, now it is version 3. The device that you have selected right here, will have a zoomed picture of this. We will have look at that. This is a new version window here. You see the device that has been mapped here it is 50. We can change even at this moment. Input files for this a space and route is going to be a sequential circuit dot edf file which we created in the synthesis, in fact you had come directly from synthesis. Therefore you don't have to specifically mention the input file name it automatically takes dot edf generate a synthesis. As the input file here although showing revision 1 here I think it is revision three. Now notice that we have constraints file. There are pin concerns which have also been seen in future. To start with we will see only the constraints file. You have floor plan guide files etcetera for time being ignore all this. Before this will just have look

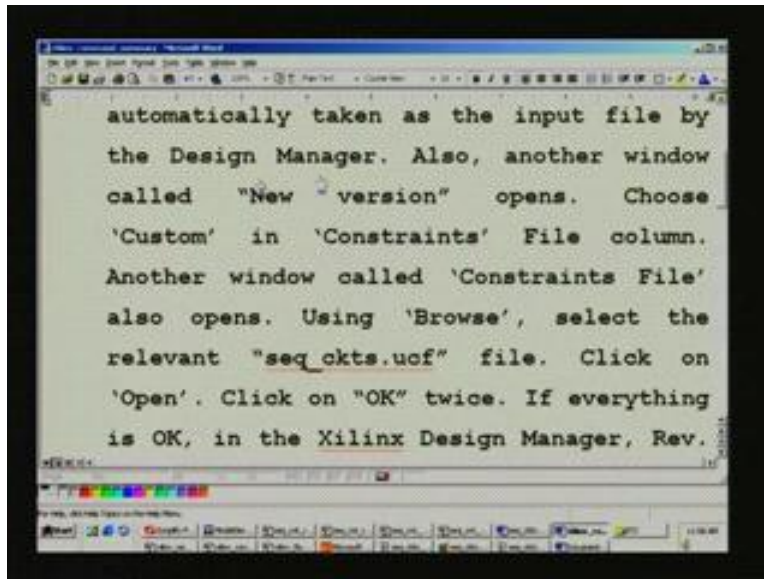


(Refer Slide Time: 37:38)



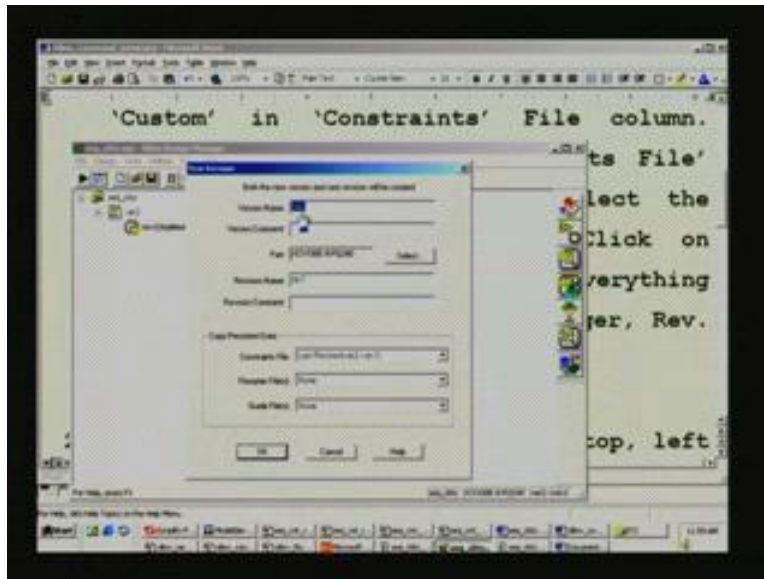
At this summary, this is the place and route we are talking about and specific Xilinx because you have to be very specific about the vendor being since it has selected target PGR as Xilinx 50 units 50 standing for the 50000 gates. After place and route back annotation only here actual get delay etcetera will come in to picture. Subsequent this you have to take black annotation file which also a dot v file. That will have to be taken in to the modelsim simulator. Run the simulation and satisfy yourself regarding functionality is concerned. There you get actual get delays reflector. I just read out this particular first step you need to take. Simplify main window click on options Xilinx start design manager what you have done earlier, in order to invoke the design manager, Xilinx design manager window opens. Sequential underscore circuits that edf created by synthesis automatically taken as the input file by the design manager that is we have already discussed.

(Refer Slide Time: 39:03)



Another window called new version opens. We opened it rather force fully here. Choose custom in constraints file. You have to select custom. Another window called constraints file also opens. Using browse select the relevant sequence circuits that is dot ucf file, ucf is a user constraints file. If you look at this file either you create it or place and route also can create it. I have already this file made we will have view of this file. After this click on open then you may have to close two windows. Therefore you have to say okay twice. If everything is okay, Xilinx design probably revision version three now, may be revision one. Very first time you had to do it will report in this fashion, new ok will displayed. It is slightly different from what has been mentioned here.

(Refer Slide Time: 40:33)

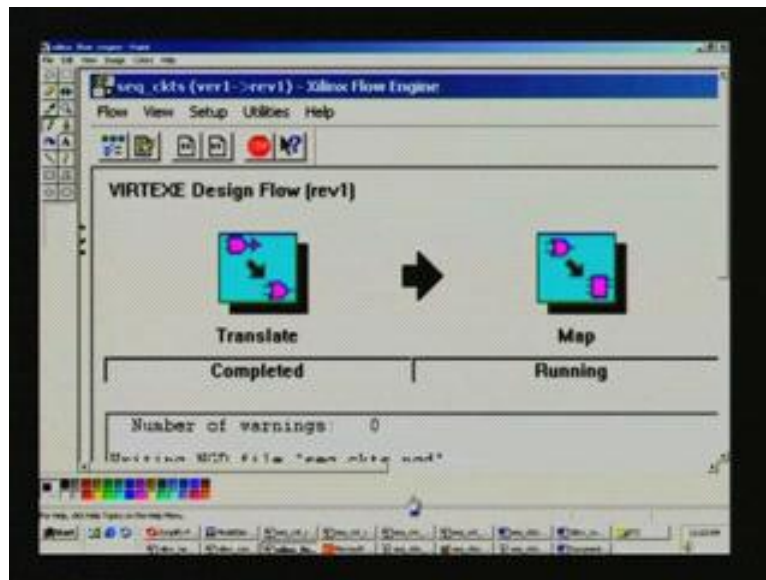


Let us go to the actual tool. This is what you have seen here it says version three. Regarding constraints file what it is says is version two is being taken here. If you click on custom it will ask for actual file. You can use browse here then go to that particular folder where you have already located ucf file click on it and say ok. If you do that it will cancel because already been taken earlier. That is what you have to do for constraints file. Here also there is zoom revision available here. We will have a constraints file, you had just seen another window popped up. Mention the file name ucf in fact if you do not mention, the system will do it for you. What all you have to do is just use this window go to the required folder where u have placed ucf file.

Click on that it will list all the files especially ucf files. Click on the relevant file if you click it will appear here now say open. Once you say open it will appear here then you can say okay. This will go to the Xilinx main window and start running. This is version three here it happens to be hundred e change the device here. Instead of 50e earlier it is 100e, this constraints file we have seen is previous version which is exactly the same sequential circuit ucf file. Once you satisfied with all this just say okay, version three, revision 1 it is reported here. The input files this is sequential circuit dot edf generated by synthesis tool. We will click on this to start. You see new open window up here, some activities going on you cannot read this file any way we will zoomed version of report files later on. First one is called translation which translates the input files that dot edf file then once to format compatible with its tool actually matches different

primitive cells etcetera including device. Within the device all the primitive cells which we already seen, after that unfortunately it went too fast.

(Refer Slide Time: 44:09)

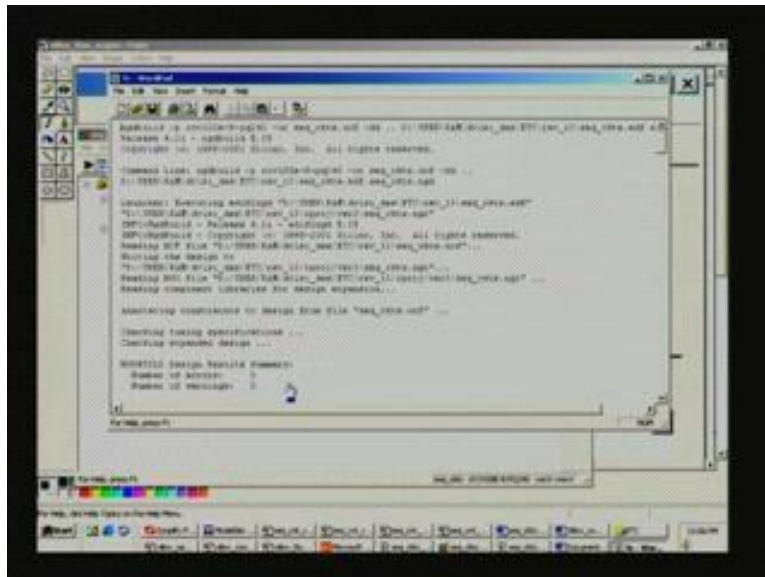


We have a zoom rotation here. Once mapping is completed this will also darken here. Also mentioned there it is already over completed just like this. This is laborious process place and route. What you have done is select is the device, then within device all the primitive cells it as the place inside. Actually placing happens only here. All the primitive cells are strategic points considering what is best in terms of frequency operations that you may get. It always has an eye on the frequency operations once it is placed. All this inputs are provided by the dot edf file which we have already seen. Based on those inputs it will start routing from 1 pin to another if it is external pin. In fact to the inner pins I mean for the functional modules.

Primitive cells also have their own IOs. There are no actual pins at the functionality within. All those things will be taken care. It will do total routing here. This is a very time consuming process. Similarly when you run the synthesis tool also takes quite substantial time. If your design file is very large, the case here in fact while placing routing will take much more time even than synthesis tool. I have personal experience for over three million gates designed. Once the place and route is completed this will also darken then final configuration will be done and overall configuration will go back to the Xilinx here. We have lots of information available say

they are reports file here as well as log file. Whatever process reported, remember it had reported on line even the place and route going on the real time it was continuing to the report. All those reports will be compiled and made available here. We can have look at a view log file.

(Refer Slide Time: 47:24)



This is the log file. I will just a copy a portion in to the document file so that you can give it from this they have an inner processing. Let us not go to the details of this. It has taken XCV 100 as a device that you have already mapped in the synthesis, also gave a user constraints file. This is the option here for that. Do not worry about all this comments unless you give it from dos. Here what is essential is what they mean. Basically, it has a constraints file which was given. Next is edf file that was reporting here, it was revision thirteen earlier in etcetera folder. We created this sequential circuit edf file.

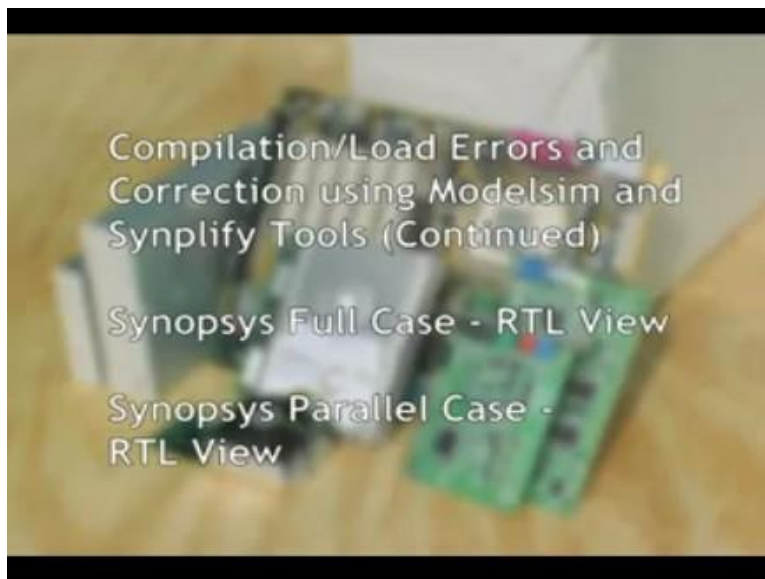
It has also created another file called ngd that is what is happening here. Let us not worry about these inner details which will be required for the further processing place. I think it is same command here again very same information is available at more than one place. It has also read from libraries regarding all primitive cells etcetera. Finally, what it says is that it is not a final thing starting on the report file. The first phase is over found no errors neither any warnings here. Then it is creating a file called sequential circuits ngd log file with this extension many other files will be created. The next step is to map the actual device and another command here, this

can also be used in a dos command mode and executed here. It takes previously generated files here this is once again constraint file. Writing into map, creating another file here, some delay information, any packing information all this. One more file is written here. Even during this stage. It reports number of errors warnings and number of slices. Slice could be this particular thing is 100000 gates so number of slices that you can have is 1200. I know what is slice is and how big it is. A collection of many LUTs will form slice.

It reports in terms of slice and number of flip flops required out of a maximum the percentage of utilities mentioned here. You have 4 input LUTs that also only 73 is for this particular design sequential circuits out of 2400. This is four inputs perhaps other numbers inputs are reported here, some iobs, iob buffers, flip flops, clocks are reported here. Finally, we can talk in terms of total gates count, only Xilinx has this provision. There is JTAG, compiles with us slandered for IO connections. That also will take more gates. Further things will be continued in the next class. Thank You.

Summary of lecture 33:

(Refer Slide Time: 54:01)





(Refer Slide Time: 54:23)



Next lecture: Xilinx place and route tool.

(Refer Slide Time: 55:10)

