Digital VLSI System Design

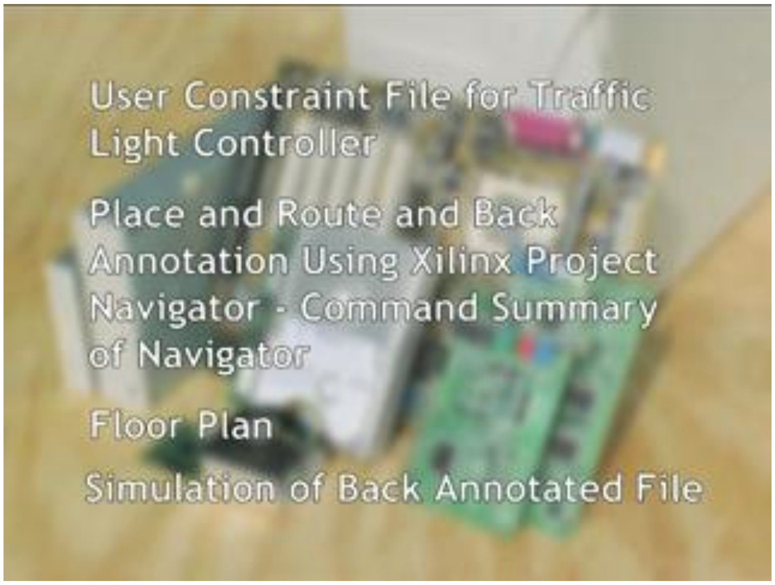
Dr. S. Ramachandran
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture 50
Advanced Features of Xilinx Project Navigator

(Refer Slide Time: 01:50)

Contents of Lecture 50

(Refer Slide Time: 02:16)



User Constraint File for Traffic Light Controller

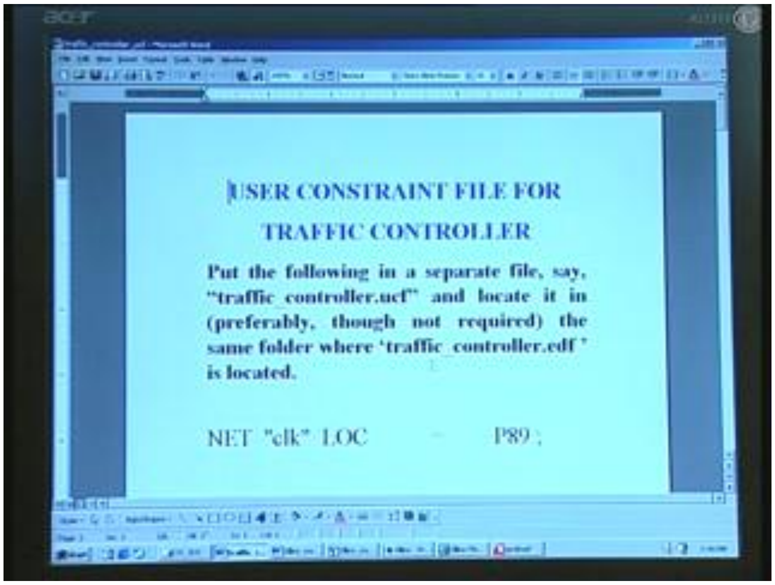
Place and Route and Back Annotation Using Xilinx Project Navigator - Command Summary of Navigator

Floor Plan

Simulation of Back Annotated File

The last time we met, you had witnessed a demo on a traffic light controller, which was a left-flowing traffic.

(Refer Slide Time: 02:30)



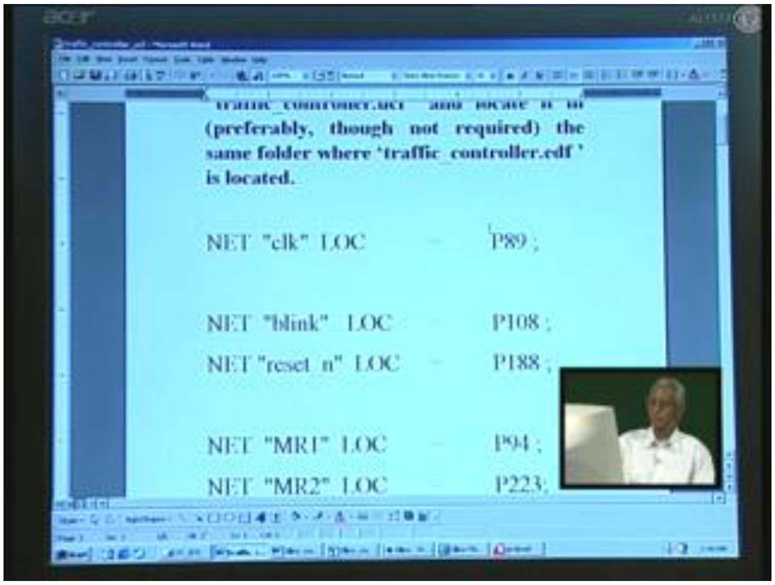
USER CONSTRAINT FILE FOR TRAFFIC CONTROLLER

Put the following in a separate file, say, "traffic_controller.ucf" and locate it in (preferably, though not required) the same folder where 'traffic_controller.edf ' is located.

```
NET "clk" LOC        =        P89 ;
```

We did not see the user constraint file for the same. What is shown here is precisely the same. You need to put that in a separate file, let us say, traffic_controller.ucf in order to do this and locate it in the same folder where the traffic_controller.edf is located. You have generated this edf file by using the Synplify synthesis tool. There is no compulsion for you to locate it in the same folder although it is preferable just for the sake of locating it readily and nothing else.

(Refer Slide Time: 03:07)

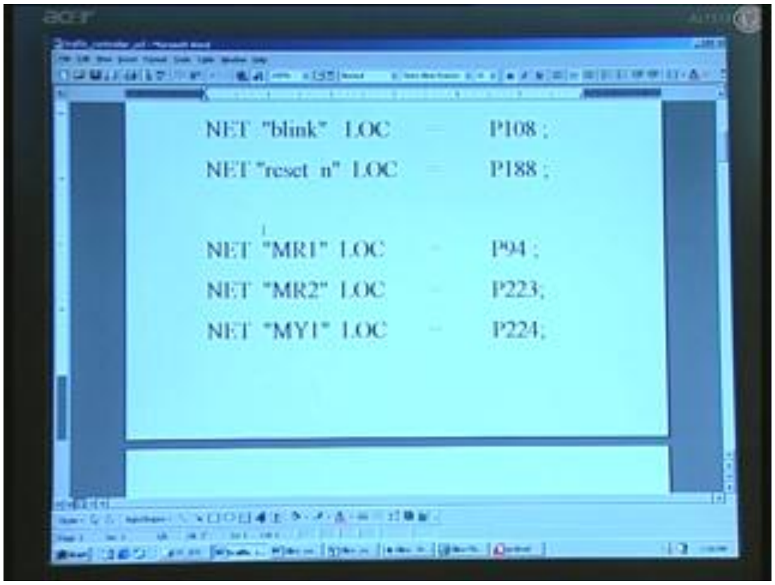


traffic_controller.ucf' and locate it in (preferably, though not required) the same folder where 'traffic_controller.edf ' is located.

```
NET "clk" LOC      =      P89 ;


NET "blink"  LOC   =      P108 ;
NET "reset_n" LOC  =      P188 ;


NET "MR1" LOC      =      P94 ;
NET "MR2" LOC      =      P223;
```

These are all the pins you will have to list. For example, you will have to declare it as NET. You can use any of the standard editors, say, WordPad or Notepad. This is the pin that you have to mention. It is to signal to the tool that you want to use this clock signal as pin 89 because we are now concerned with the hardware connection on the FPGA. This user constraint file abbreviated as .ucf is precisely to tell the tool that such and such a pin is to be taken as a particular signal. For example, three input signals are clock, blink and reset_n, which we have already seen quite a number of times.

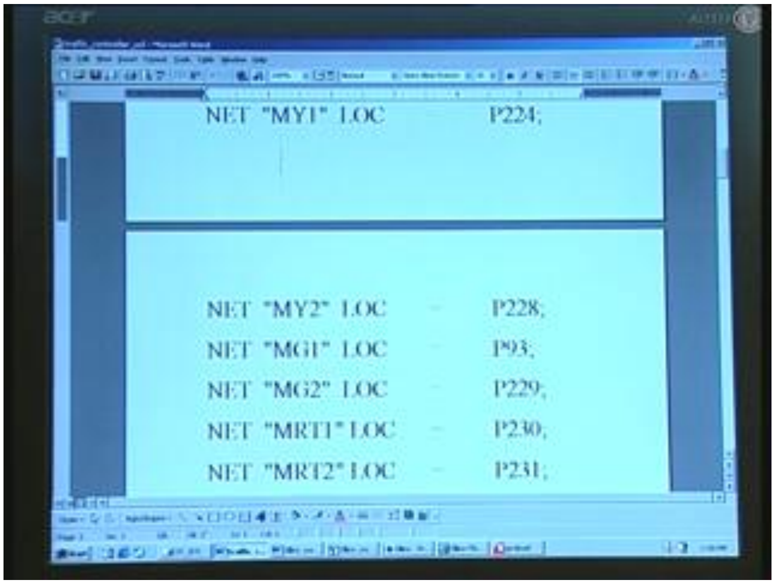
(Refer Slide Time: 03:57)



```
NET "blink"  LOC    =    P108 ;
NET "reset n" LOC   =    P188 ;

NET "MR1" LOC       =    P94 ;
NET "MR2" LOC       =    P223;
NET "MY1" LOC       =    P224;
```

We had sixteen lamps or LEDs. This is the nomenclature we adopted. M stands for main and S for side. It is a four-road junction meeting point and R stands for red. There are two roads in the main, M stands for main, so main red 1. Similarly, main road 2 indicates the very same main road. You also have a yellow light for the same. Just keep track of the number of outputs that you will see.

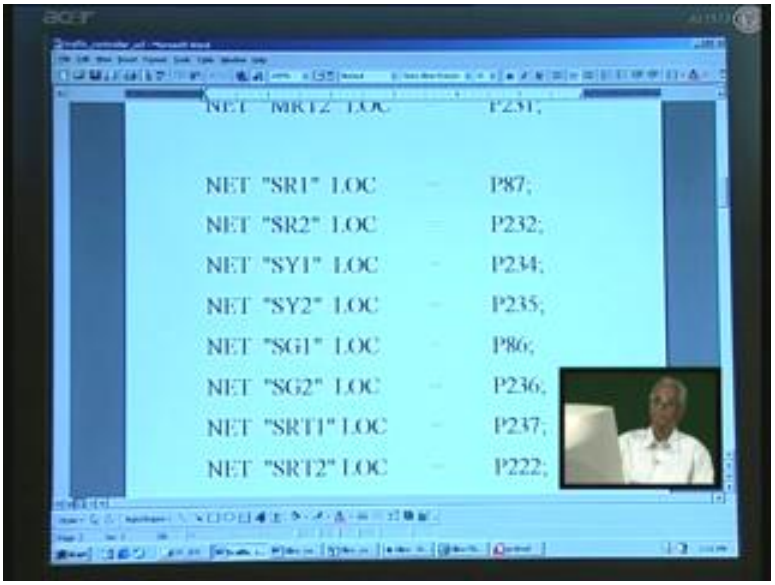
(Refer Slide Time: 04:31)



```
NET "MY1" LOC            P224;

NET "MY2" LOC       =    P228;
NET "MG1" LOC       =    P93;
NET "MG2" LOC       =    P229;
NET "MRT1" LOC      =    P230;
NET "MRT2" LOC      =    P231;
```

The actual pins are listed on the right. We have MY1, main yellow 1 and 2, followed by green and right, each of which is for 1 and 2.

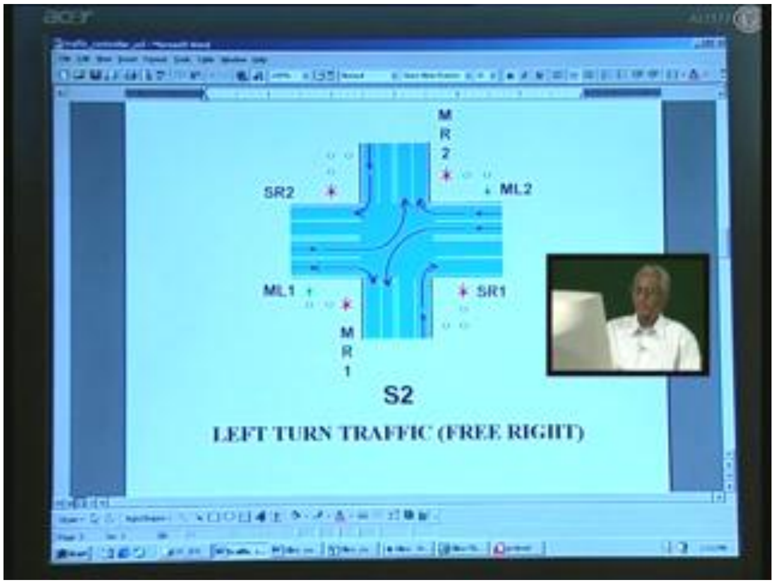
(Refer Slide Time: 04:45)



```
NET "MRT2" LOC    =    P231;

NET "SR1" LOC     =    P87;
NET "SR2" LOC     =    P232;
NET "SY1" LOC     =    P234;
NET "SY2" LOC     =    P235;
NET "SG1" LOC     =    P86;
NET "SG2" LOC     =    P236;
NET "SRT1" LOC    =    P237;
NET "SRT2" LOC    =    P222;
```

You also have a side road and precisely the same definitions follow for this side road as well. You see side road red 1 and 2 and then yellow, green and so also for the right. I think this completes the sixteen outputs for the traffic light controller.

(Refer Slide Time: 05:00)



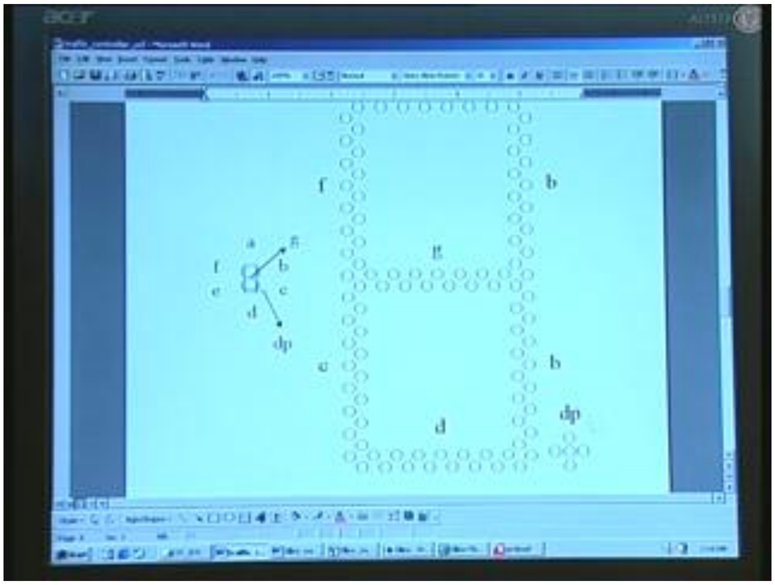
S2

LEFT TURN TRAFFIC (FREE RIGHT)

What we have seen so far is the left-side traffic and that is in the Asian countries. In European and United States, you have right-flowing traffic. What is a depicted is right-flowing traffic right here. It is two-lane traffic here, although generally, in expressways, etc., you have three ways and in freeways in the US, you have six ways for each of the two directions. There will be a median. Where there is no real concrete structure on the median, you will have a double yellow line. Each of the lanes will be separated by a white line. Notice that there is one more white part – this is normally for parking. There will be another white line closely here and that will be the cycle lane. Normally, when you pull on to the sides, you will have to see that it is after the cycle lane and right on the curb.

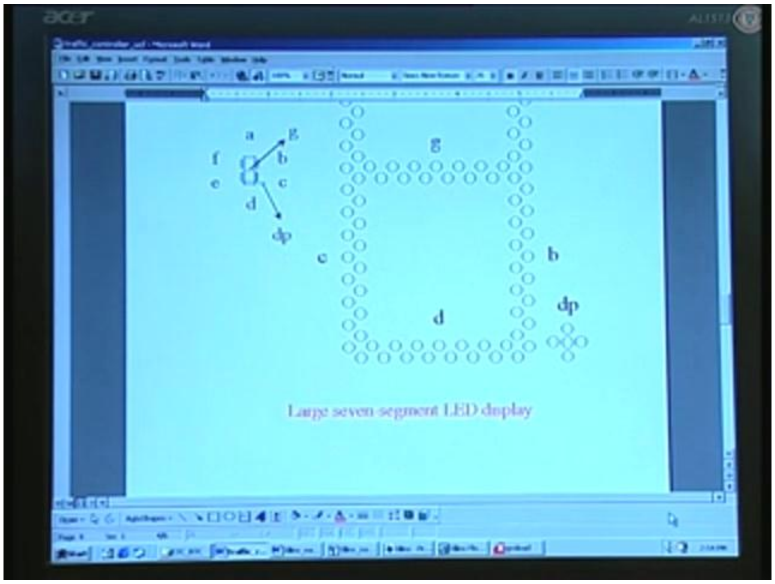
The right side of your wheel will have to be within 18 inches from the curb. I have assumed a free right turn and therefore, you can see a free right turn in all directions. What is depicted here is the left-flowing traffic, left turn. Naturally, this is the right flowing traffic. Earlier to this, it was straight-going traffic. If you want to go straight, you should not use this lane because this is generally meant for left-flowing traffic. Of course, in Indian roads, you do not really comply with all these rules and regulations but in US and other places, they strictly adhere to these. The unwritten law is that they should take to this left-most lane. Otherwise, they will have problems. If you get here, you may get stranded. This track or lane is for free right as well as for the straight-flow traffic and so when straight traffic resumes, it will go straight here. We have seen earlier left-flowing traffic. What is shown here is right-flowing traffic for the same sequence. One such sequence was S2 for left-flowing traffic. The very same sequence is maintained here for right-flowing traffic.

(Refer Slide Time: 07:34)



Of late, you would have noticed that there are timers displayed with big displays on the roads. Each display is actually comprised of 15 LEDs or so for each of the segments. For example, you are already used to this. This is a seven-segment displays a through f, g, with the right decimal point. I have put the same thing here so that we can form a bigger display. Each of the segments will have some 15 LEDs in the placement shown here.

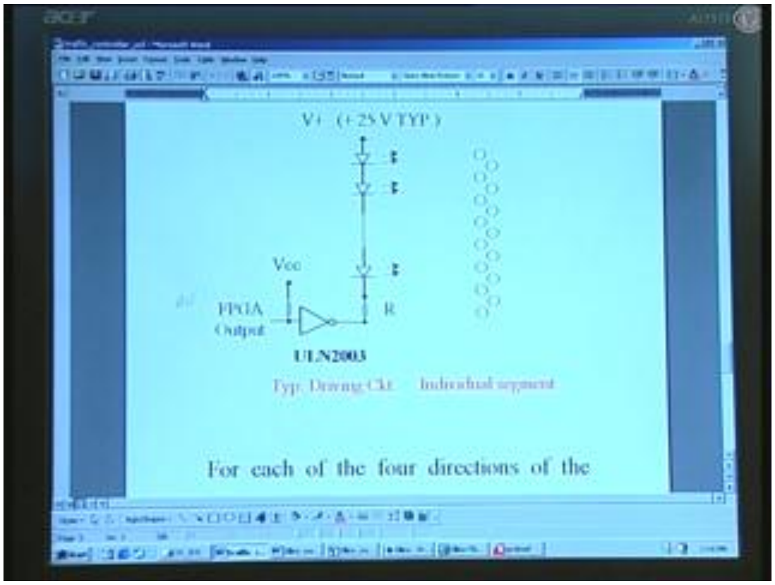
(Refer Slide Time: 08:12)



Large seven-segment LED display

There is also a decimal point here. The flexibility that you have is you can put it in any other size you want. All you have to do is add more number of LEDs.

(Refer Slide Time: 08:26)



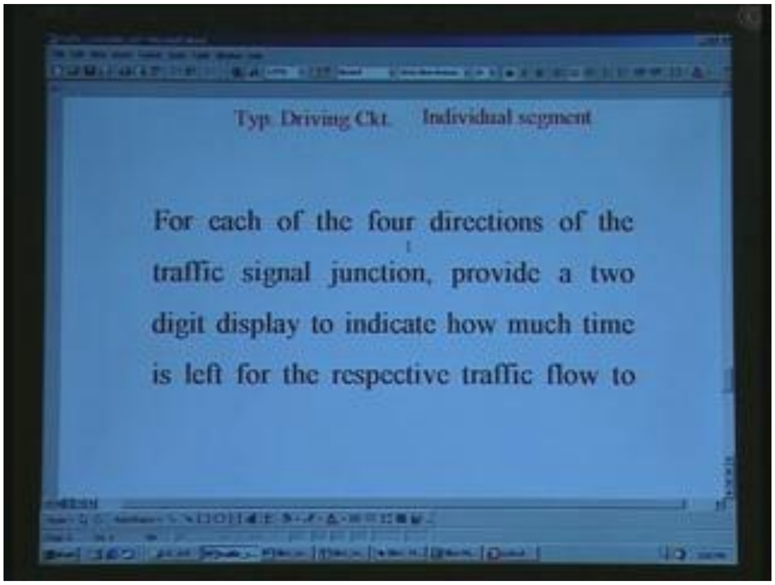
For each of the four directions of the

The interface circuit that you need will be as shown here. For example, you have what is called a ULN2003. This is the device available. It is actually a Darlington pair. The input comes from the FPGA output, say one of the lights that you have already seen, main red, etc. They are all single-bit outputs and they are all connected here. This is just a typical representative circuit for one of the traffic lights. Each of this one, if you go for a larger display, for one segment, you need a group of LEDs.

They are all cascaded and shown here as a series connection. Note that when an LED conducts, it will take approximately 1.5 Volts. If you put fifteen of them, naturally it calls for, I think, 22.5 Volts. If you have a supply of 25 Volts, the difference of around 2 Volts would be dropped across this resistor and this resistor would turn out to be around 200 Ohms. An individual segment has been shown here. This is the exploded display that you may like to configure (Refer Slide Time: 09:46), the individual segment of which we have already seen here.

The circuitry is quite simple. This 2003 can go right up to 50 Volts – you can connect 50 Volts here. If you wish, you can have some more LEDs here. You have to select the resistance in such a fashion that around 10 million, which is the safe current rating for the LED, passes through this. Suppose you want to drive a solenoid or a relay instead of an LED, you can go right up to 500 million. You can use the same circuit for some other purpose such as driving a solid-state relay, but you require a different setup, which we may see in future.

(Refer Slide Time: 10:40)

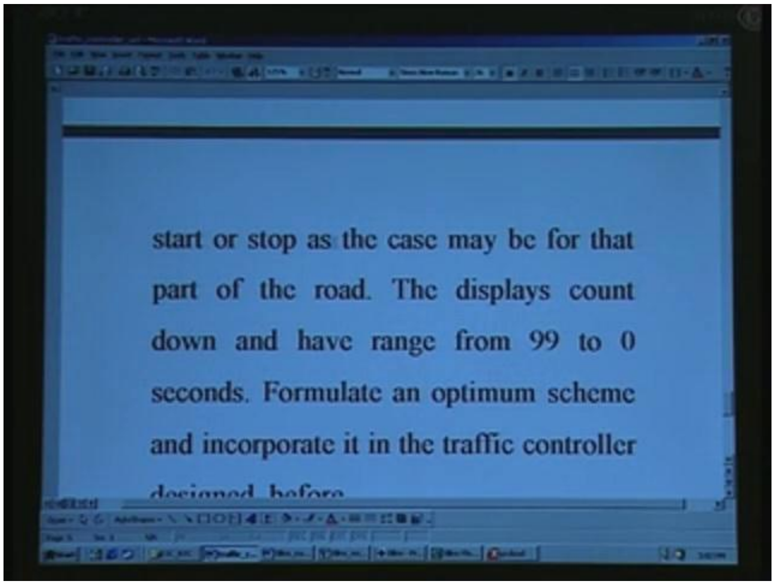


Typ. Driving Ckt.   Individual segment

For each of the four directions of the traffic signal junction, provide a two digit display to indicate how much time is left for the respective traffic flow to

This is the assignment we are actually seeing for you in order to include a two-digit display for the display of timer. I will read out the assignment first and then explain the same. For each of the four directions…. We have a four-road meeting point or junction point – that is what we mean by four. We had said main road 1 and 2 and side road 1. All these together would form four directions. From each of the directions, as a road user, you will be facing the traffic lights and alongside that, you have a two-digit display. That display must be big enough so that it is visible from a long distance.

(Refer Slide Time: 11:26)



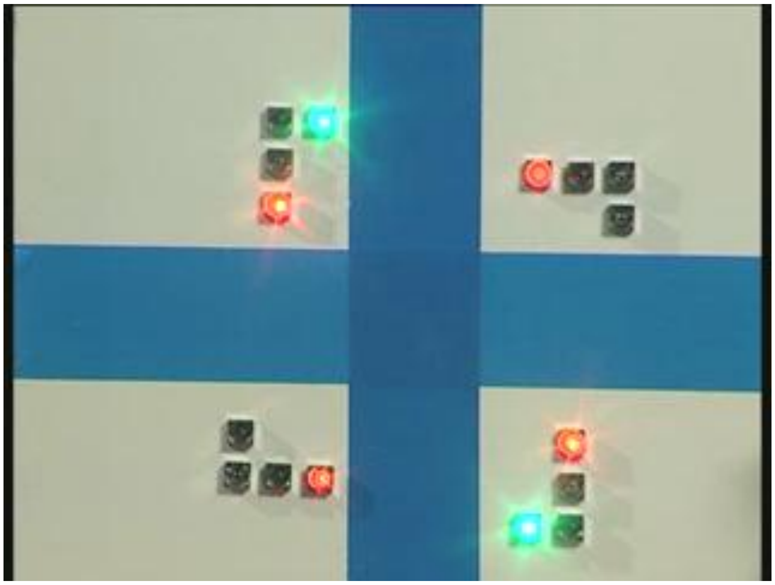
start or stop as the case may be for that part of the road. The displays count down and have range from 99 to 0 seconds. Formulate an optimum scheme and incorporate it in the traffic controller designed before

You have to provide a two-digit display to indicate how much time is left for the respective traffic flow to start or stop, as the case may be for that part of the road. I will explain this once again after showing the traffic light that we have seen before. The displays count down and have a range from 99 to 0 seconds. Formulate an optimum scheme and incorporate it in the traffic controller we have designed before.

Let us have a look at the signal we have already seen (Refer Slide Time: 12:04). In this case, what holds good for this holds equally good for the left-flowing traffic. Earlier, we have given this as an assignment to you, if you remember, asking you to recode what we have done earlier for left-flowing traffic. I hope you have not done that because there is no need for you to do it. The reason is that this is nothing but a mirror reflection of left-flowing traffic. You did not really have to do any coding at all. Absolutely no recoding is necessary, not even a single character needs to be changed; precisely the very same code will work on this for the simple reason that this is nothing other than a mirror reflection.

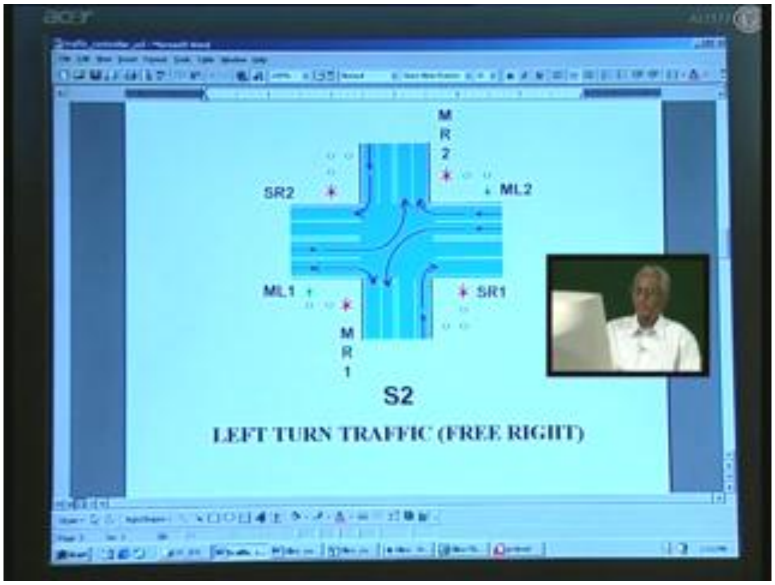
In fact, if you go to the other side of this 2D, what you will actually see is only left-flowing traffic. What all it amounts to is instead of putting these outputs here, it was right here (Refer Slide Time: 13:06). That is precisely what you will see if you go to the other side of this figure and that happens to be just a mirror reflection. Therefore, you do not need to have any recoding. This particular sequence is actually shown here on this board.

(Refer Slide Time: 13:27)



You can just have a look at this and this will be clear. What is shown is precisely the same left-flowing traffic that you have here.

(Refer Slide Time: 13:39)



S2

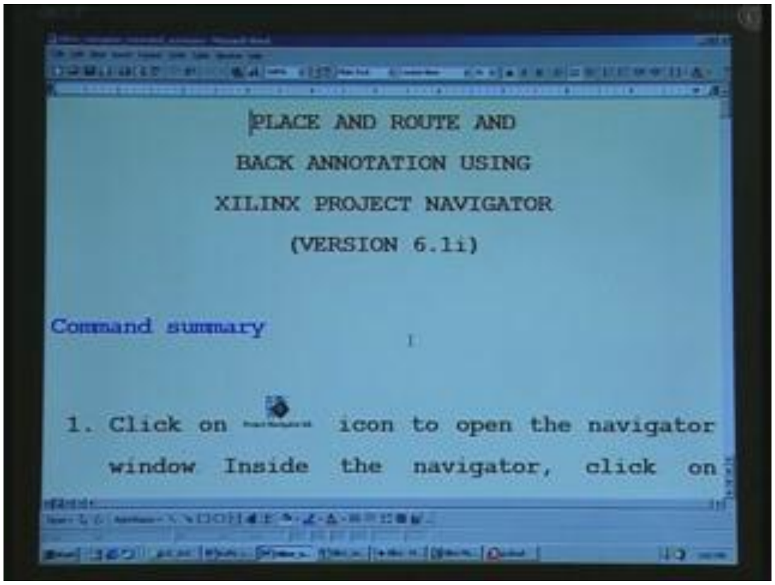
LEFT TURN TRAFFIC (FREE RIGHT)

Coming back to the next assignment that we have been discussing earlier, that is to provide a two-digit display for the timing, for each of the set here, you will put a two-digit display facing the traffic. If it is left-flowing traffic, this group will come there; otherwise, there is no difference. Now, let us see what the implications are. Earlier, there was straight-flowing traffic here and it was going this way. We have said that straight-flowing traffic is timed for 45 seconds and another 5 seconds for the yellow transition. That would take you to 50 seconds. For left-flowing traffic, even from the main road to the side road, we have given a timing of 20 seconds plus 5 seconds for the yellow and so in total, you have 50 plus 25, 75 seconds.

You will have to wait if you are on the side road. You cannot avail the thing because what is allowed first is straight-flow traffic, followed by the left-flow traffic for the main road. All other things will follow suit except for the side road – the main flow will be 25 seconds and the turn will also be 25 seconds. You have to take all this into consideration and then accordingly design the timer and enable them when to start and when to stop. For instance, let us say the main flow traffic has just commenced. If you look here, the two digits will be initially set to 50 seconds because we have 45 seconds plus 5 seconds transition, which will go straight here. After this, the same counter will be reset to 25 and allow this traffic flow. This is once again inclusive of 5 seconds for the yellow light.

Having done that, the total is 75 seconds. There would be another timer running all along and that would be running from 75 to 0, whereas here it would be first 50 seconds, followed by 25 seconds and then going over to 0. Like this, you have to take care for each of the directions. I hope you understand. Is it clear to all of you, this assignment? This completes one of the applications, the traffic light controller. We will go to the next application. In the meanwhile, I have seen that there is some change of software as far as Xilinx is concerned.

(Refer Slide Time: 16:16)



PLACE AND ROUTE AND

BACK ANNOTATION USING

XILINX PROJECT NAVIGATOR

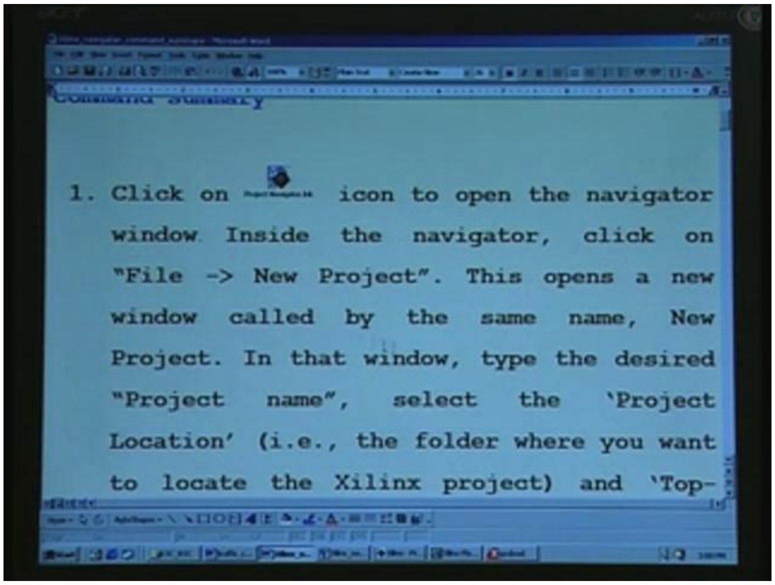
(VERSION 6.1i)

Command summary

1. Click on icon to open the navigator window Inside the navigator, click on

We had earlier used a design manager for place and route as well as for back annotation. What we will now see is the navigator used for the same purpose. Once again, I have given the command summary here. This is place and route and back annotation using Xilinx Project Navigator. This version is 6.1i and what is loaded on this machine is 6.2i. I do not see much difference between the two versions and I hope these are the latest ones. This is the command summary that we have here. I will read it out and then go on to the actual tool.

(Refer Slide Time: 16:55)



Command Summary

1. Click on icon to open the navigator window Inside the navigator, click on "File -> New Project". This opens a new window called by the same name, New Project. In that window, type the desired "Project name", select the 'Project Location' (i.e., the folder where you want to locate the Xilinx project) and 'Top-

If you have any doubt, please stop me at any point of time. We will have a navigator here.

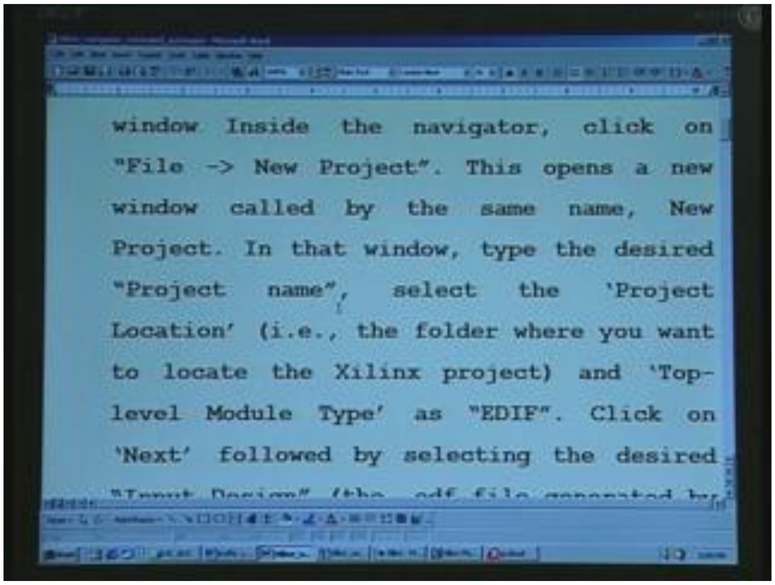
(Refer Slide Time: 17:04)

You can see the icon here and the icon here is here, I hope you are able to see that. This is the project navigator and the same icon is shown right here.

Just click on this one in order to invoke this window, to open the navigator window. I will just read partially and then go over to the actual tool. Inside the navigator, click on File and then New Project. Any starting you make will have to be as a new project. If it is for the second time, you can open the existing file. Here, it happens to be a new project. When we really go into the tool, we may not have to invoke this and it may straightaway enter the already created project.

I have verified and I have also asked other engineers to verify and so I do not think any problem will arise even if I bypass some of the steps. Anyway, I will read out and explain this. This opens a new window called by the same name, whatever is the file name that you have given for the new project. In that window, type the desired project name and select the project location – where you want to locate the project will be given over here.

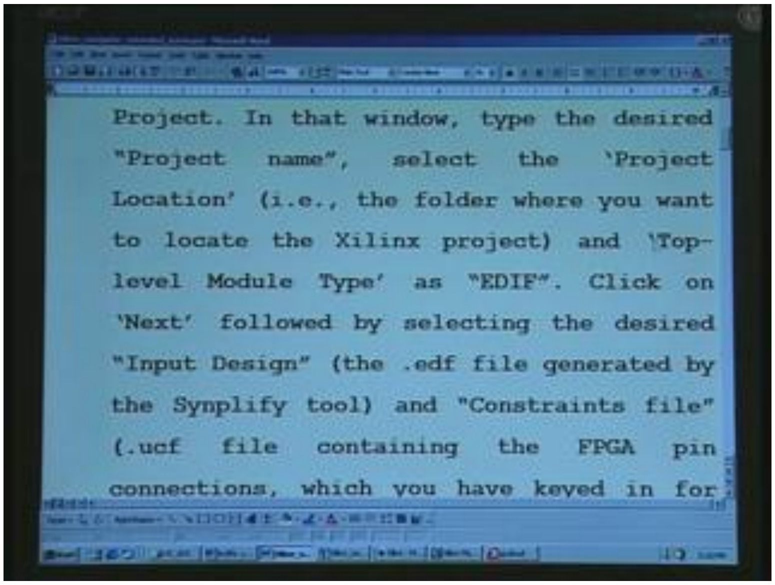
(Refer Slide Time: 18:32)



window Inside the navigator, click on "File -> New Project". This opens a new window called by the same name, New Project. In that window, type the desired "Project name", select the 'Project Location' (i.e., the folder where you want to locate the Xilinx project) and 'Top-level Module Type' as "EDIF". Click on 'Next' followed by selecting the desired "Input Design" (the .edf file generated by

Usually, this will be a place where you have already put all your source codes and Synplify outputs such as edif file or user constraint file, which we will be requiring for this particular navigator. You may house all this in a particular folder. Right in that folder, you can create a new project going by the same name. For example, if it is a real time-clock or traffic light controller, you name it as such. Select the project location, that is, the folder in which you want to locate the Xilinx project.

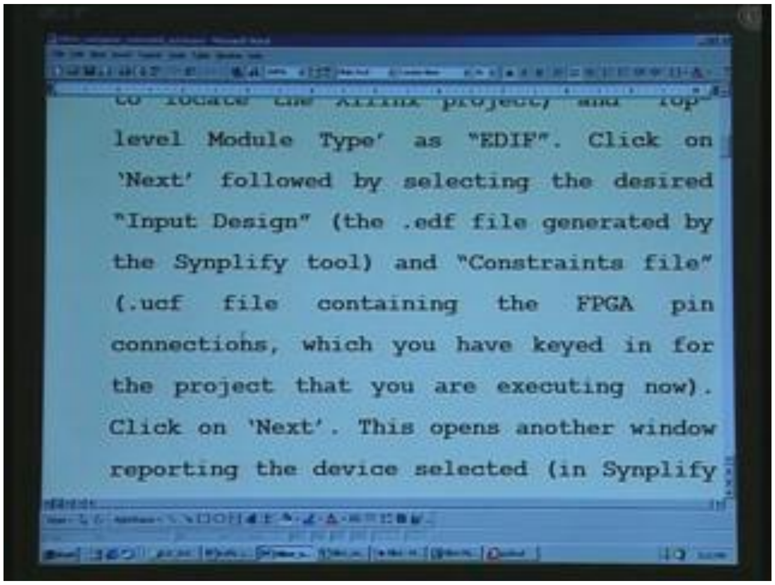
(Refer Slide Time: 19:07)



Project. In that window, type the desired "Project name", select the 'Project Location' (i.e., the folder where you want to locate the Xilinx project) and 'Top-level Module Type' as "EDIF". Click on 'Next' followed by selecting the desired "Input Design" (the .edf file generated by the Synplify tool) and "Constraints file" (.ucf file containing the FPGA pin connections, which you have keyed in for

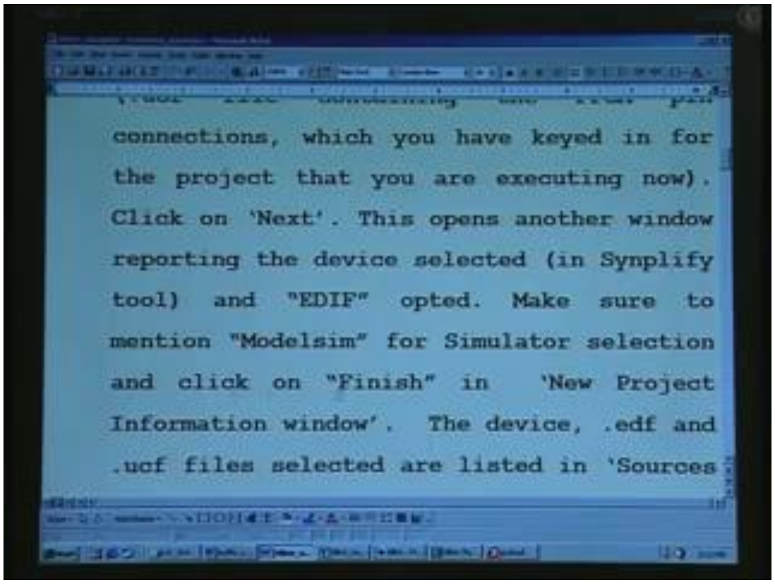
There is another subclass in that menu called top-level module type. You have to choose it as EDIF. It will become clear to you when we look into the actual navigator window that I will be opening shortly. Click on Next followed by selecting the desired input design. Mind you, we are right now doing the place and route. This implies that we have already run the synthesis tool, which is the Synplify tool that we have already used earlier and using which we created the .edf file. If traffic_controller.v is your design, you will get traffic_controller.edf as the Synplify output. That will be the input for this design manager or the navigator tool, which we are explaining right now. That will take in .edf file.

(Refer Slide Time: 20:16)


to locate the Xilinx project) and 'Top level Module Type' as "EDIF". Click on 'Next' followed by selecting the desired "Input Design" (the .edf file generated by the Synplify tool) and "Constraints file" (.ucf file containing the FPGA pin connections, which you have keyed in for the project that you are executing now). Click on 'Next'. This opens another window reporting the device selected (in Synplify

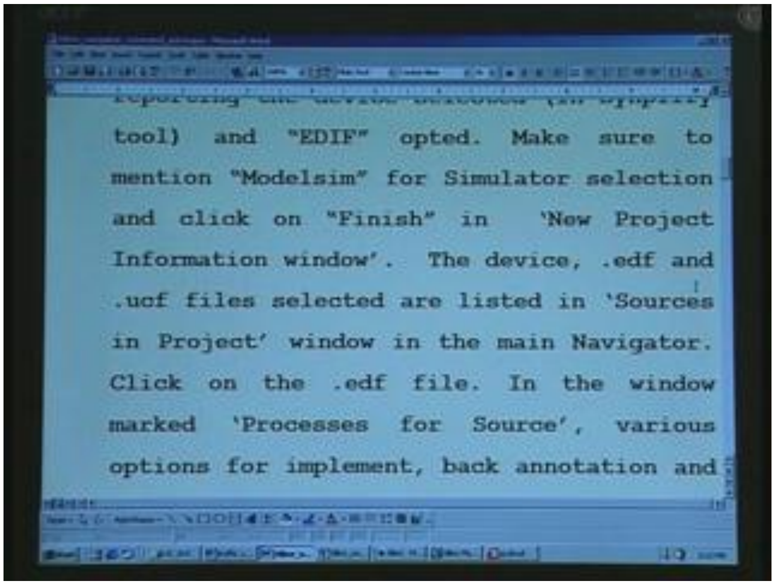
You also need the constraints file. We have seen an example of traffic controller constraint file earlier with .ucf extension. Obviously, the file containing the FPGA pin connections contains the pin connections here that you have keyed in for the project that you are executing now. Right now, you are executing a particular project, for which the .ucf file will have to be taken.

(Refer Slide Time: 20:40)


(.ucf file containing the FPGA pin connections, which you have keyed in for the project that you are executing now). Click on 'Next'. This opens another window reporting the device selected (in Synplify tool) and "EDIF" opted. Make sure to mention "Modelsim" for Simulator selection and click on "Finish" in 'New Project Information window'. The device, .edf and .ucf files selected are listed in 'Sources

Then, click on Next. This opens another window reporting the device selected in Synplify tool and this is what we have selected in Synplify. In this tool, it will automatically know what device you have mapped using the Synplify tool. That will be automatically taken care of, which will be apparent when we go further. This opens another window reporting the device selected in the Synplify tool, of course, and EDIF opted. You have opted for EDIF in this place and route tool. Make sure to mention Modelsim for simulator selection and click on Finish in the New Project Information window.

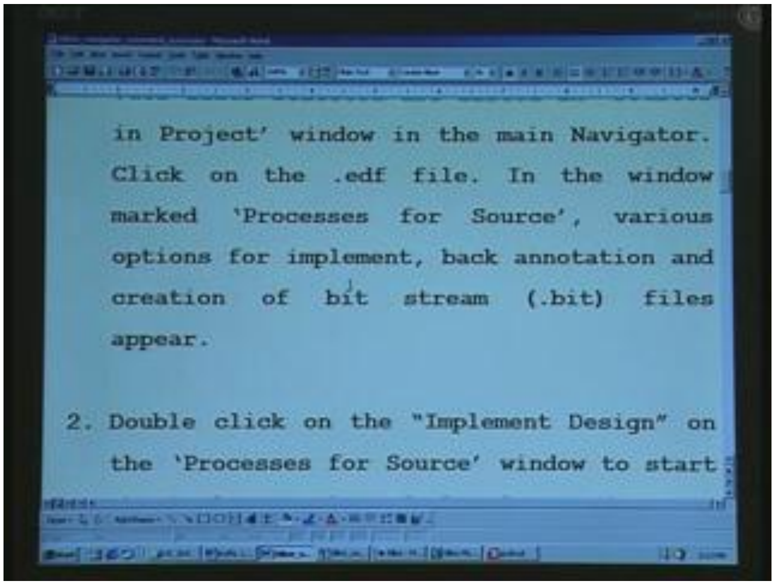
(Refer Slide Time: 21:24)



reporting the device selected (in Synplify tool) and "EDIF" opted. Make sure to mention "Modelsim" for Simulator selection and click on "Finish" in 'New Project Information window'. The device, .edf and .ucf files selected are listed in 'Sources in Project' window in the main Navigator. Click on the .edf file. In the window marked 'Processes for Source', various options for implement, back annotation and

The device .edf and .ucf files selected are listed in the Sources in Project window in the main Navigator. When we open the navigator, we will see all these windows. We will come back to this step after seeing the window.

(Refer Slide Time: 21:47)

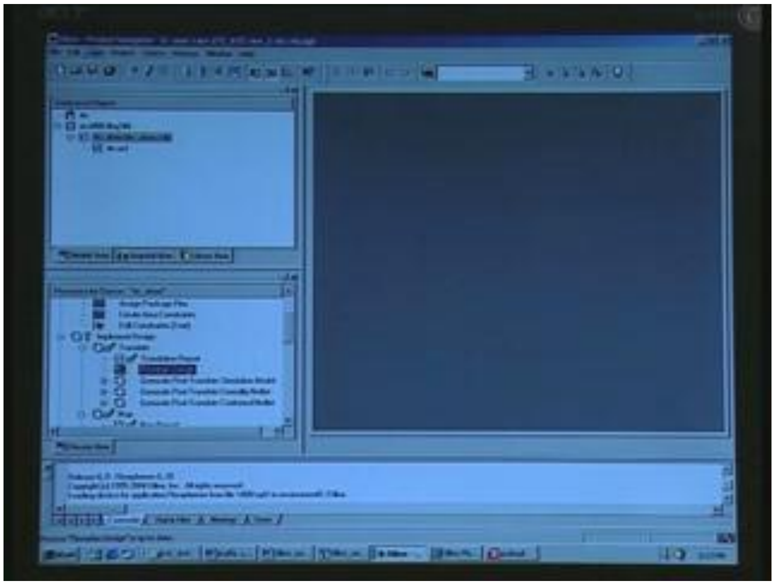


in Project' window in the main Navigator. Click on the .edf file. In the window marked 'Processes for Source', various options for implement, back annotation and creation of bit stream (.bit) files appear.

2. Double click on the "Implement Design" on the 'Processes for Source' window to start

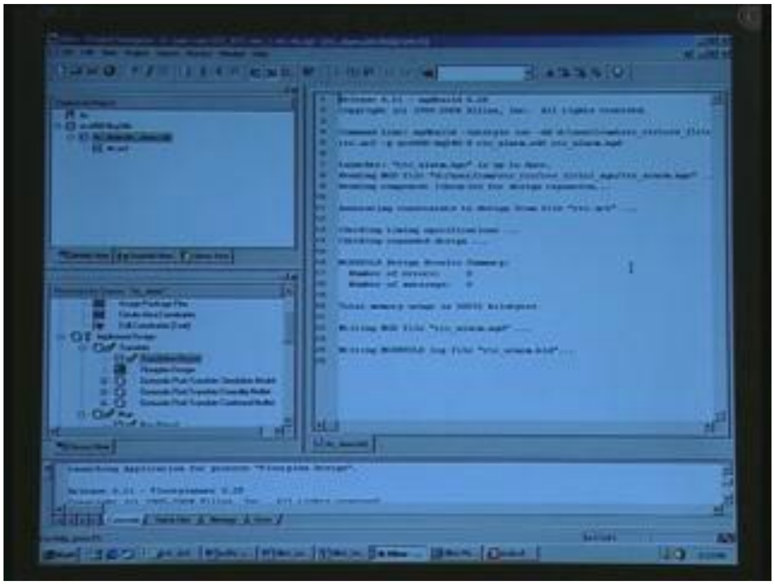
Click on the .edf file. In the window marked Processes for Source, various options for implement, back annotation and creation of bit stream (that is .bit files) appear. With this first point, we will go on to the actual navigator.

(Refer Slide Time: 22:00)



This is the main project navigator, which is obtained after clicking on the navigator icon. I am afraid it will not be readable to those of you who see the TV monitor. I am going to show an exploded view of this. Before that, I think you can make out at least some patch of white light here, white here and one more window here. These are all basically windows.

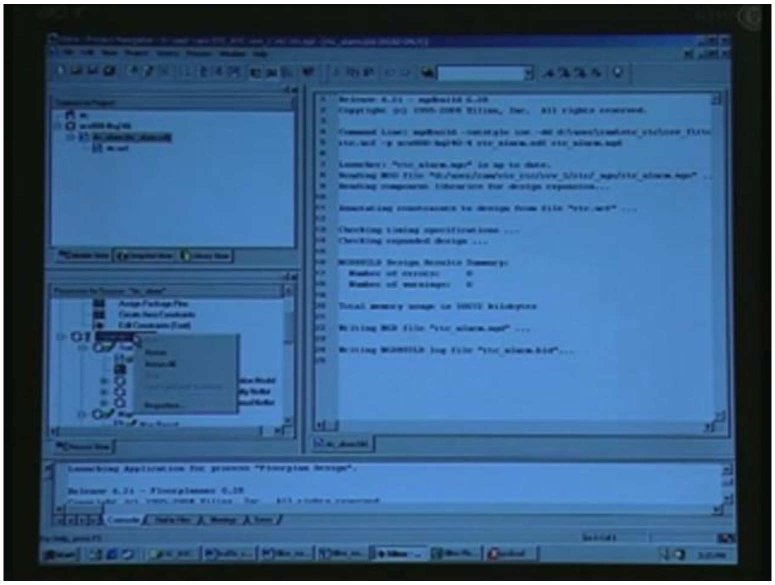
(Refer Slide Time: 22:30)



If I double click here, you will get another window energized. You will see in total four types of windows. This window is known as Sources in Project and it is here that you will get the edf file and ucf file. You will have to key in here. You can even change these files if you want to go for some other project. Another window is Processes for Source and here what is listed is Implement Design, Map, Place and Route. You will see wide varieties of place and route. We will see one after another. Finally, to generate a bit file, we need Generate Programming File. I will show you a zoomed version in a minute now.

You can go from one part of the tool to another. You also have what is known as floor design. You can see the actual layout of what you have put into the FPGA. There is a window here. When you start implementing, what all you have to do is go to a particular thing and double-click. For example, if I click on this implement design, if I double-click, it will run on this. There is a tick mark here (Refer Slide Time: 23:50). I wonder whether you can notice the tick mark on the monitor. If you do, then you will see a question mark here if I rerun the thing.

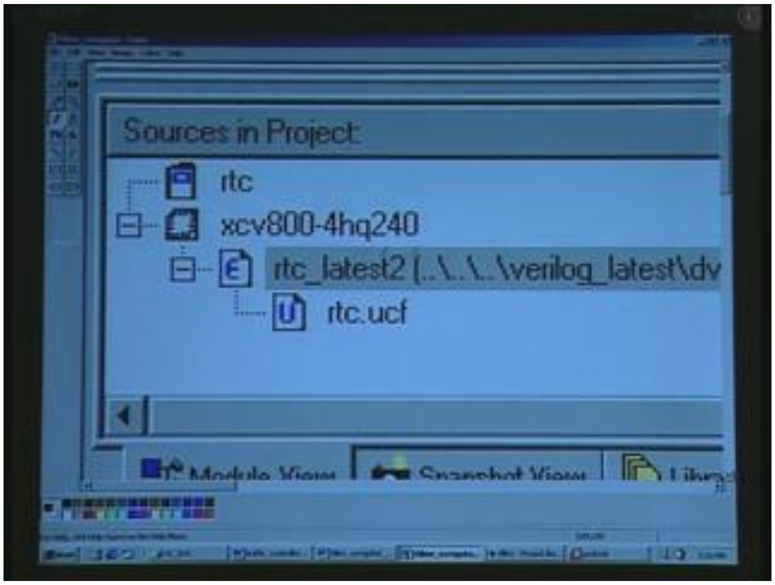
(Refer Slide Time: 24:03)



```
Release 4.2i - ngdbuild [illegible]
Copyright (c) 1995-2000 Xilinx, Inc.  All rights reserved.

Command Line: ngdbuild [illegible]
rtc.ucf -p [illegible] rtc_alarm.ngd rtc_alarm.ngd

Launcher: "rtc_alarm.ngo" is up to date.
Reading NGO file [illegible] ...
Reading component libraries for design expansion...

Annotating constraints to design from file "rtc.ucf" ...

Checking timing specifications ...
Checking expanded design ...

NGDBUILD Design Results Summary:
  Number of errors:     0
  Number of warnings:   0

Total memory usage is [illegible] kilobytes

Writing NGD file "rtc_alarm.ngd" ...

Writing NGDBUILD log file "rtc_alarm.bld"...
```

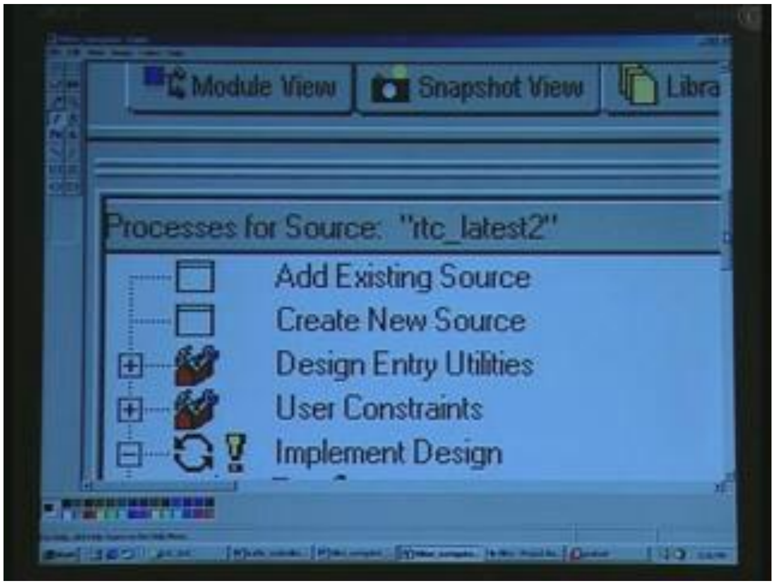
For example, I can click on the right and you have seen another small window opening. There is a Rerun and Rerun All. If I press Rerun now, it will start running. Now you can see a question mark here. It will keep on running and it will keep reporting on this window. Now let us have the zoomed window.

(Refer Slide Time: 24:35)



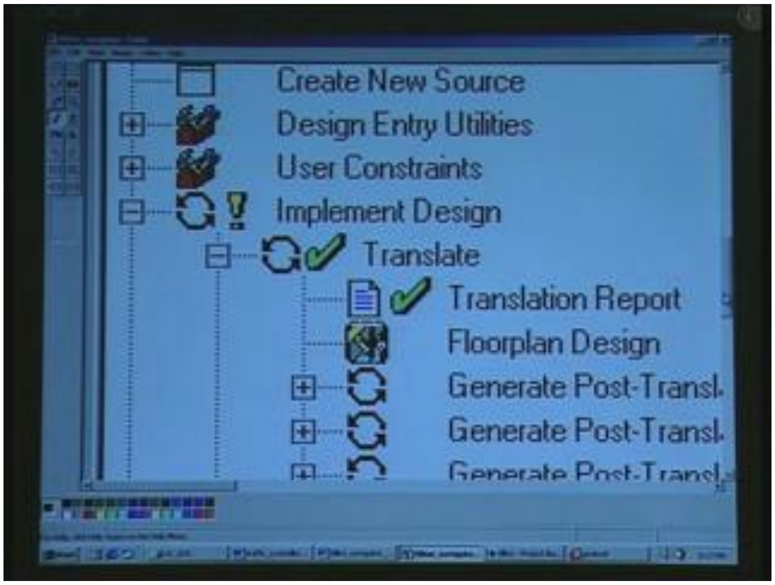
What you see is the zoomed window here and as I mentioned, this is the Sources in Project window. For example, this is what we are going to cover as our next design application, a real-time clock abbreviated as rtc. There are several versions. Do not worry about the actual name here. This is the actual edf file. Unfortunately, because it is large, we cannot see the whole thing here. Know that it is just .edf extension. You also need a constraint file, which is rtc.ucf. This is what is presented in the first Sources in Project window.

(Refer Slide Time: 25:18)



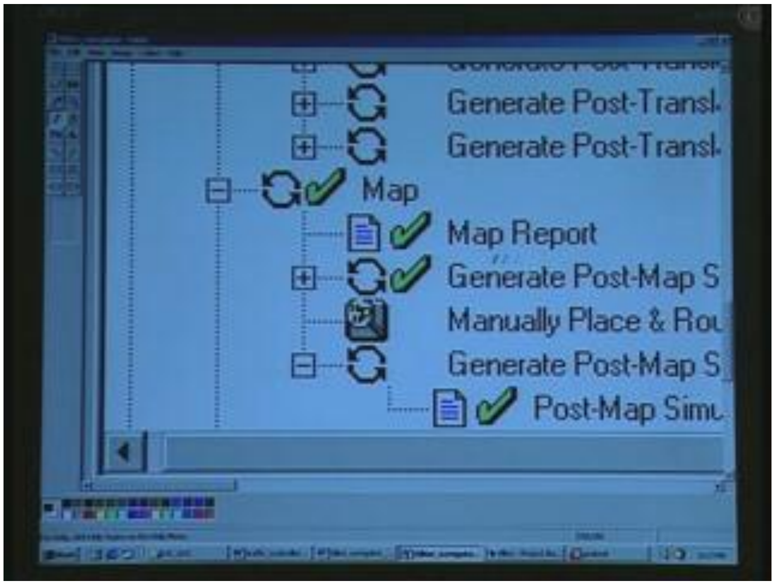
If you go down, you will see the Processes for Source. Once again, the very same project name is indicated here. You can see it only partially. Implement design is what we have already seen and I have explained to you.

(Refer Slide Time: 25:35)



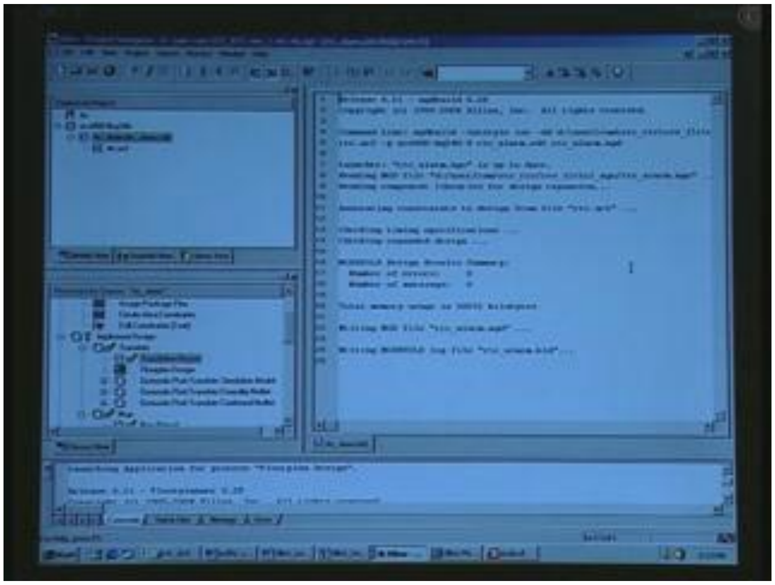
You can see the tick marks very clearly here. You have Translate and then you can do a translation report. If you want to see the actual FPGA layout, you can double-click on this Floorplan Design in order to invoke the same.

(Refer Slide Time: 25:55)



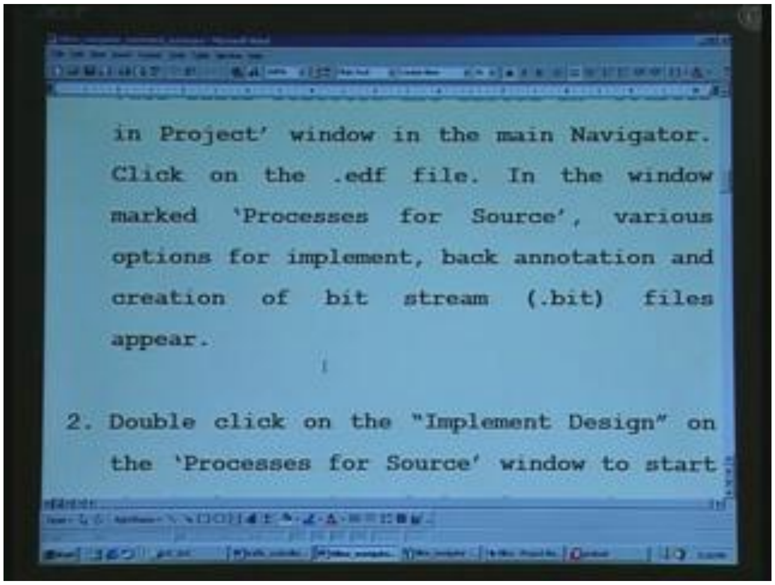
The device that you have mapped and what you need is what is called mapping. After mapping is done, you have a map report as well. These are all the various map reports available. There are many features here. I cannot take them here because this is only copied in Paint. I can only read out here.

(Refer Slide Time: 26:22)



This is the actual real window of the navigator. I can drag this here. Earlier, you have seen the zoomed version and I could not drag because it is just a snapshot there. What you see here, that the last one is the Generate Programming File. I have already explained this. This covers all the points of point one, which we have already seen here.

(Refer Slide Time: 26: 55)

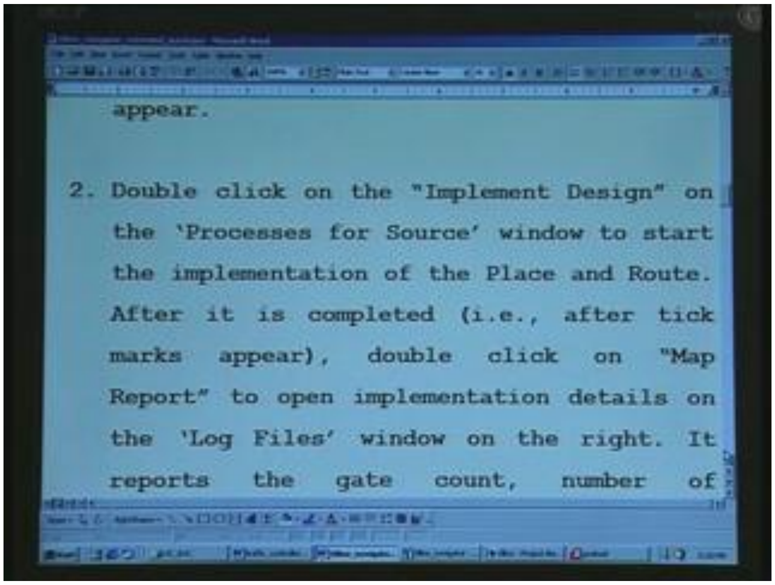


in Project' window in the main Navigator. Click on the .edf file. In the window marked 'Processes for Source', various options for implement, back annotation and creation of bit stream (.bit) files appear.

2. Double click on the "Implement Design" on the 'Processes for Source' window to start

We will go on to point two. This is what we have seen. In Processes for Source, various options for implement, back annotation and creation of bit stream files appear. We have already seen this.

(Refer Slide Time: 27:04)



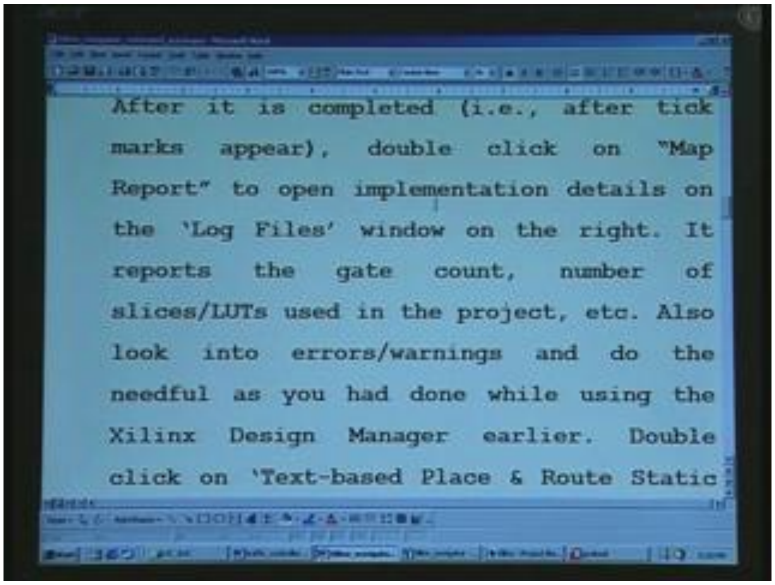
appear.

2. Double click on the "Implement Design" on the 'Processes for Source' window to start the implementation of the Place and Route. After it is completed (i.e., after tick marks appear), double click on "Map Report" to open implementation details on the 'Log Files' window on the right. It reports the gate count, number of

The second point is that we need to start the implementation. If you want to implement design, what you do is just double-click on the Implement Design on the Processes for

Source window to start the implementation of the Place and Route. This is the second window that we have already seen (the zoomed version as well). After if it is completed, that is after tick marks appear, double-click on Map Report to open implementation details on the Log Files window on the right.

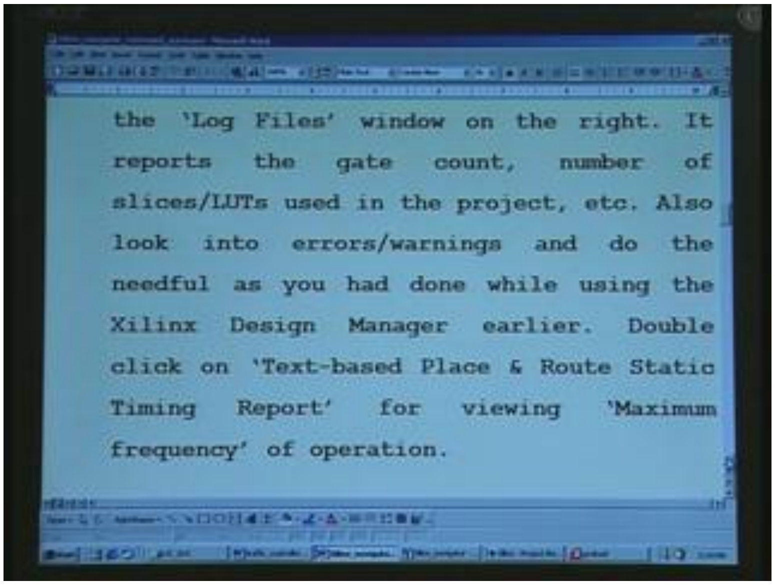
(Refer Slide Time: 27:35)



After it is completed (i.e., after tick marks appear), double click on "Map Report" to open implementation details on the 'Log Files' window on the right. It reports the gate count, number of slices/LUTs used in the project, etc. Also look into errors/warnings and do the needful as you had done while using the Xilinx Design Manager earlier. Double click on 'Text-based Place & Route Static

I will read it fully and then go on to the window to show the same. It reports the gate count, number of slices or LUTs used in the project, etc., for this particular project. Also look into errors or warnings and do the needful as you had done while using the Xilinx Design Manager earlier. We are already used to correcting these and you will have to make a number of trials till you get a working code on the FPGA.

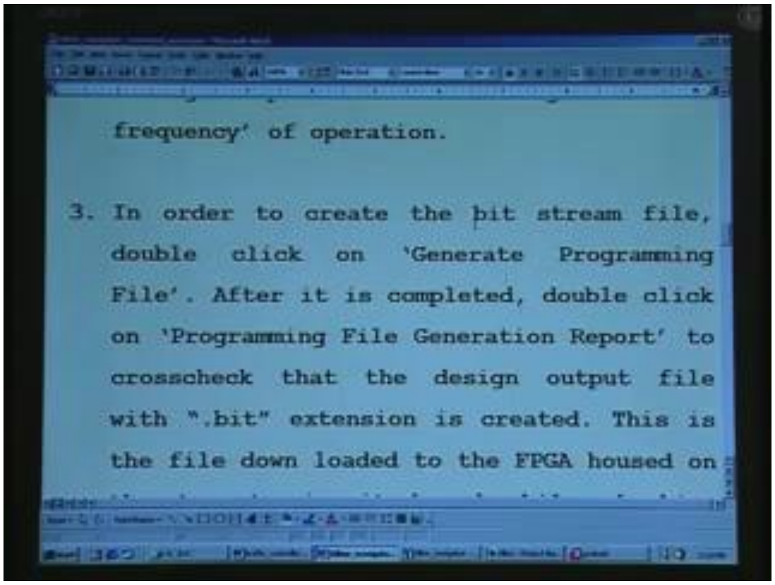
(Refer Slide Time: 28:07)



the 'Log Files' window on the right. It reports the gate count, number of slices/LUTs used in the project, etc. Also look into errors/warnings and do the needful as you had done while using the Xilinx Design Manager earlier. Double click on 'Text-based Place & Route Static Timing Report' for viewing 'Maximum frequency' of operation.

You have to double click on Text-based Place & Route Static Timing Report for viewing the maximum frequency of operation. If you are interested in finding out what is the maximum frequency of operation is, you can work on the real hardware. You will have to double-click on Text-based Place & Route Static Timing Report.

(Refer Slide Time: 28:30)

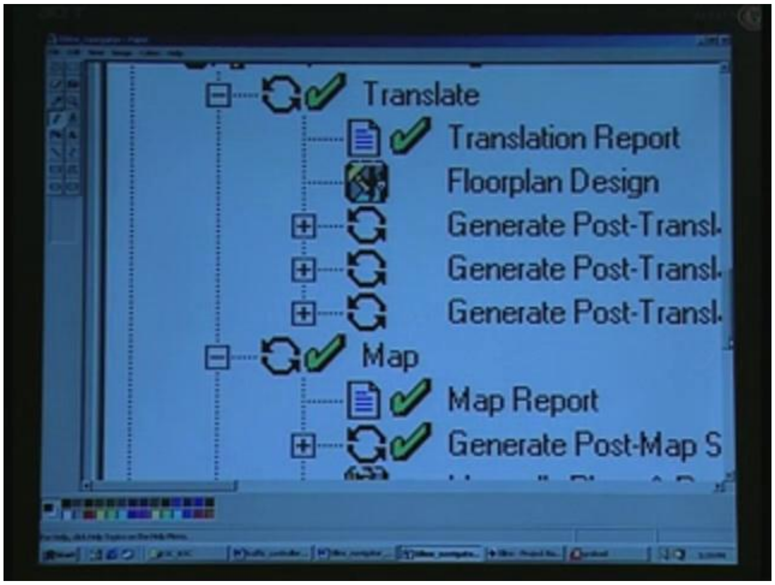


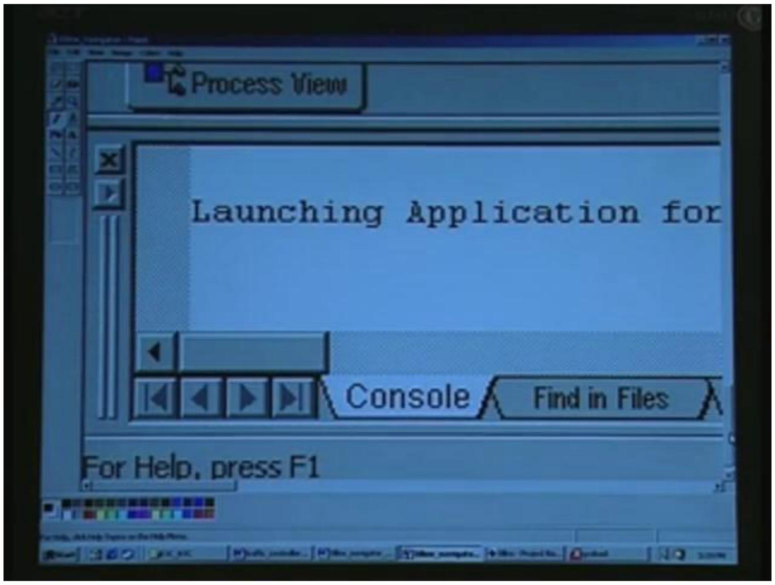
frequency' of operation.

3. In order to create the bit stream file, double click on 'Generate Programming File'. After it is completed, double click on 'Programming File Generation Report' to crosscheck that the design output file with ".bit" extension is created. This is the file down loaded to the FPGA housed on

The third one is bit generation. We will have a look once again here.

(Refer Slide Time: 28:37)



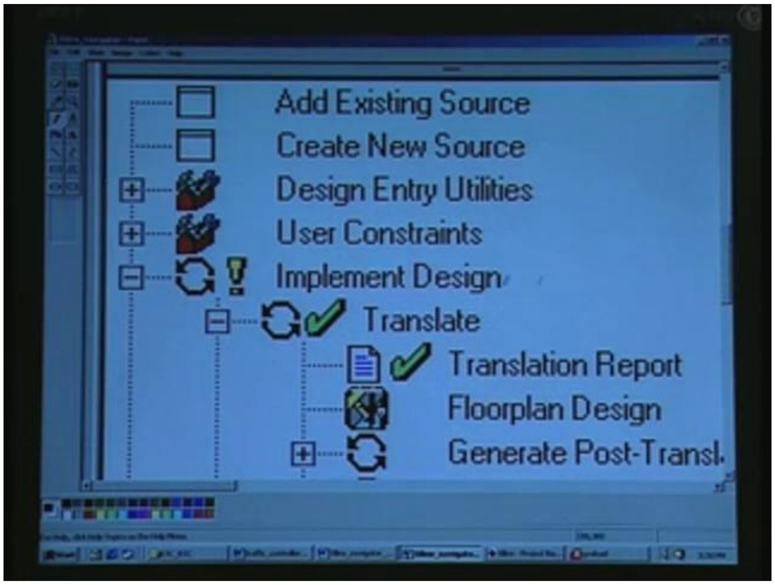


```
Launching Application for
```

For Help, press F1

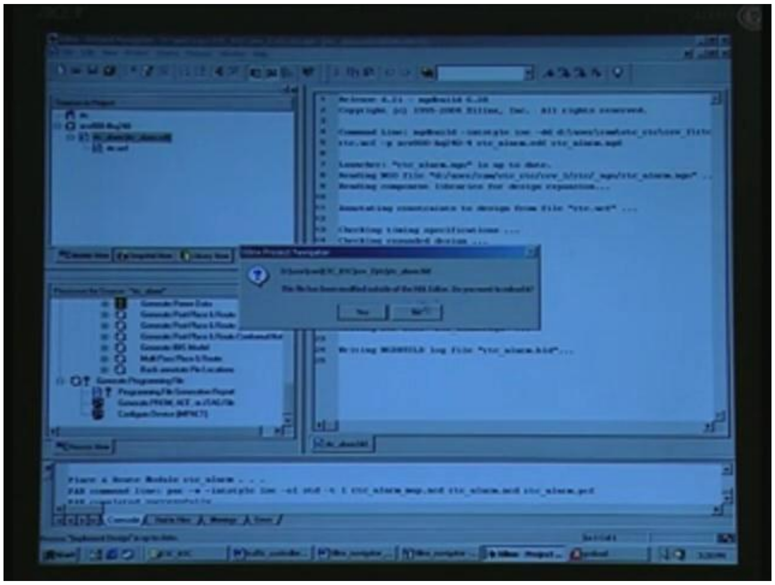
This is the zoomed version. What I said is there is a window down here, which keeps on reporting when you double-click Implement Design.

(Refer Slide Time: 28:49)



If you double-click on this, that will run. Let us have a look at the actual.

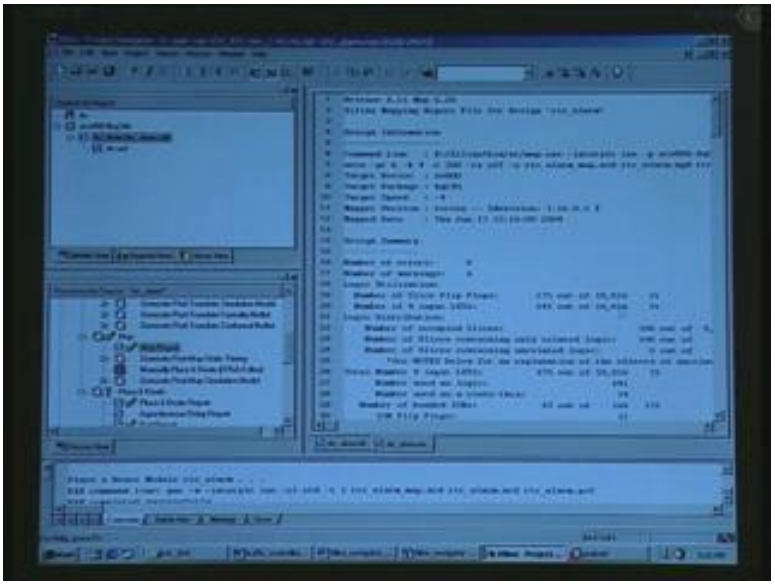
(Refer Slide Time: 28:58)



```
Copyright (c) 1995-2004 Xilinx, Inc.  All rights reserved.

Annotating constraints to design from file "rtc.ucf" ...

Checking timing specifications ...
Checking expanded design ...

Writing NGDBUILD log file "rtc_alarm.bld"...
```

Sometimes, it will ask whether to reload or not. I just said No for the time being.

(Refer Slide Time: 29:05)

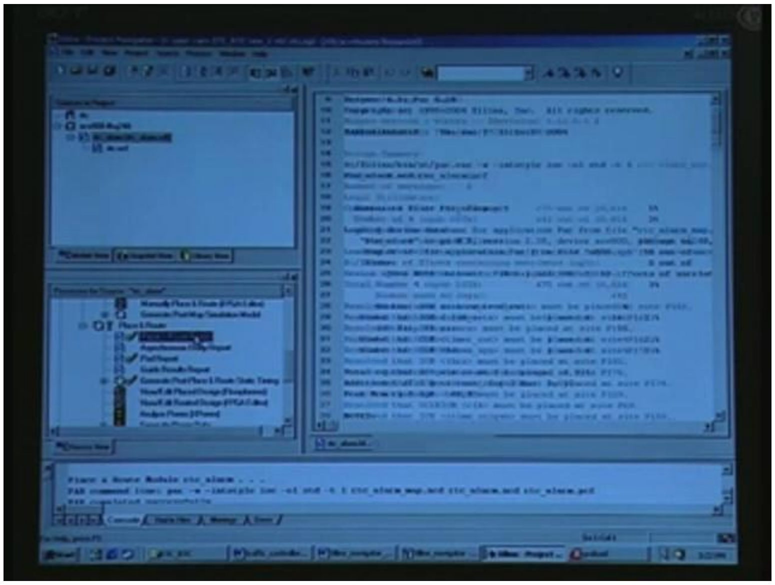


Let us see what we have here. This Implement Design, if you remember, I had given Rerun and it has completed. Earlier, when I gave Rerun, there was a question mark implying that it was running. Now having completed, it would give a tick mark. For example, I want to see the translation report.

What you do is just double-click on the translation report and you get the report on the right. It simply says number of errors is 0 and number of warnings is 0. This is for the next project that are going to take up. This is a real-time clock. rtc_alarm is the design file. If you want further details, just go to Map here and then double-click on Map Report. Out comes this. We have already read these points. It reports the number of slices. For example, this rtc requires just 275 slices of flip-flops out of 18,000 on the device. The device that we have used earlier was also the same device 800 because that happens to be the board that we have. Similarly, four input LUTs that have been consumed are just 641 out of 18,000.

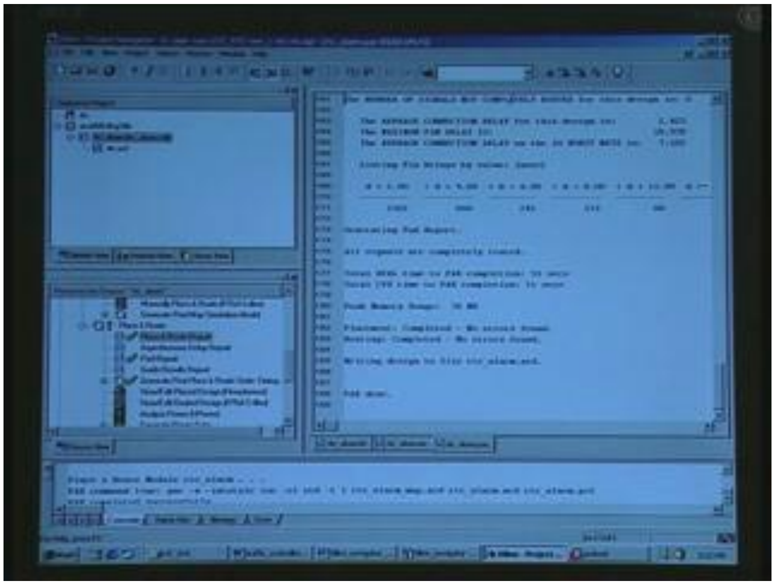
Let us not go into the details of this, which we are going to cover when we take up the next application. It is sufficient to say that it merely reports what the gate count is. For example, here it reports gate count 7,000 and odd, JTAG IOBs and part of the gate count you can take. All are reported only in this Map Report here.

(Refer Slide Time: 31:01)



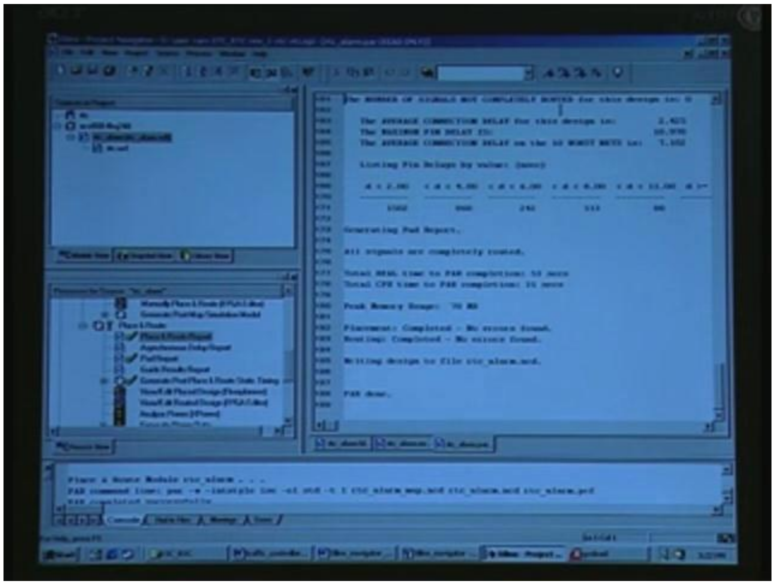
There is also a place and route report and it gives in more detail, for example, which pin was connected to which point and what IOBs are used. It gives all pin listing and also what clocks are used, the number of slice used and so on.

(Refer Slide Time: 31:23)



It also gives the timing summary here. Let us not go into the details right now. We will see this when we come to that design.

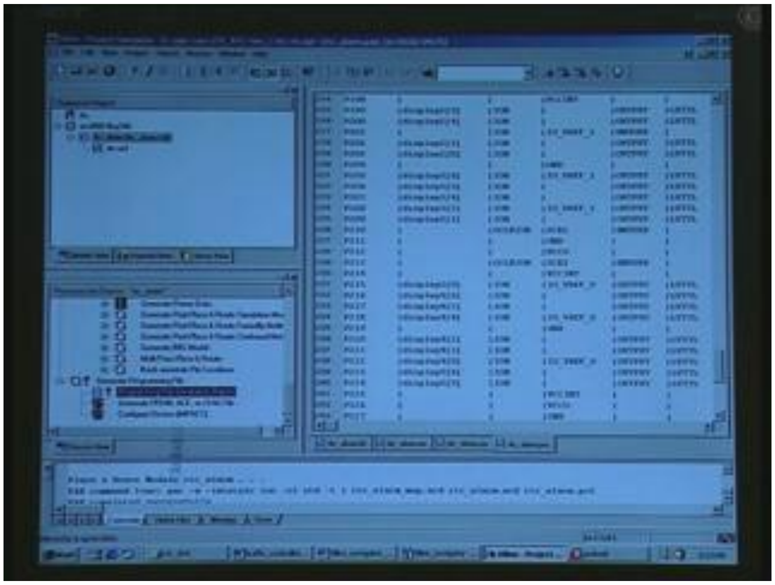
(Refer Slide Time: 31:32)



```
The NUMBER OF SIGNALS NOT COMPLETELY ROUTED for this design is: 0

   The AVERAGE CONNECTION DELAY for this design is:        [illegible]
   The MAXIMUM PIN DELAY IS:                               [illegible]
   The AVERAGE CONNECTION DELAY on the 10 WORST NETS is:   [illegible]

   Listing Pin Delays by value: (nsec)

Generating Pad Report.

All signals are completely routed.

Total REAL time to PAR completion: [illegible] secs
Total CPU time to PAR completion: [illegible] secs

Peak Memory Usage: [illegible] MB

Placement: Completed - No errors found.
Routing: Completed - No errors found.

Writing design to file rtc_alarm.ncd.

PAR done.
```

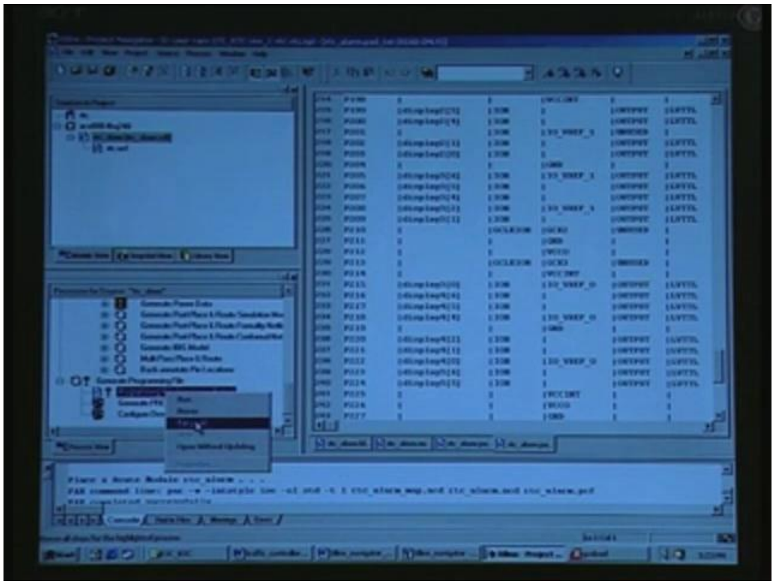
This is basically the timing details. Finally, it says there are no errors in place and route. That is what is here. We have seen implementation, then mapping, then place and route and you also have a paired report separately.

(Refer Slide Time: 31:47)



That will list all the IO pins that have been connected. That is what you see on the right. It may be difficult for you to read and that is why I am explaining. When you actually start using the tool, you will understand this clearly. Anyway, the command summary gives the exact directions, so you will have no difficulty in using the tool. Some post-place route static timing are also available there. We have the Generate Programming File here. This is also a question mark.

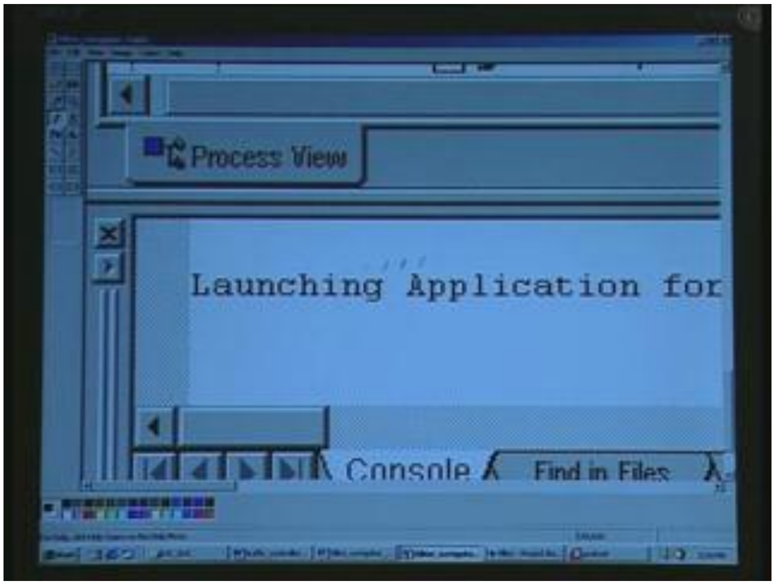
(Refer Slide Time: 32:30)



```
Place & Route Module ctc_alarm . . .
PAR command line: par -w -intstyle ise -ol std -t 1 ctc_alarm_map.ncd ctc_alarm.ncd ctc_alarm.pcf
PAR completed successfully
```

For example, if I want to run this, I just rerun there and let it run. When it runs, you see that some activity is taking place. The last window is to report what activity is going on. Right now, you can see that 50 percent of the job is completed as far as generating the bit file is concerned. What we will do is go on to the zoomed part again to see whether we have left out anything. This Sources in Project is the first window that we have seen, which will give you the edf and ucf files. This is the implementation file called Processes for Source and then, you have Implement, Mapping and Place and Route and then Generate Programming.

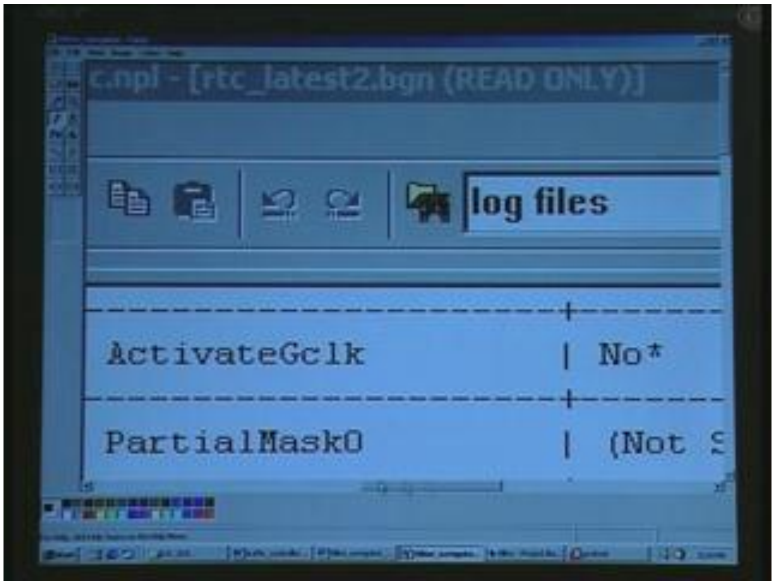
(Refer Slide Time: 33:11)



```
Launching Application for
```

This is to report the actual activity that is going on and we cannot drag this.

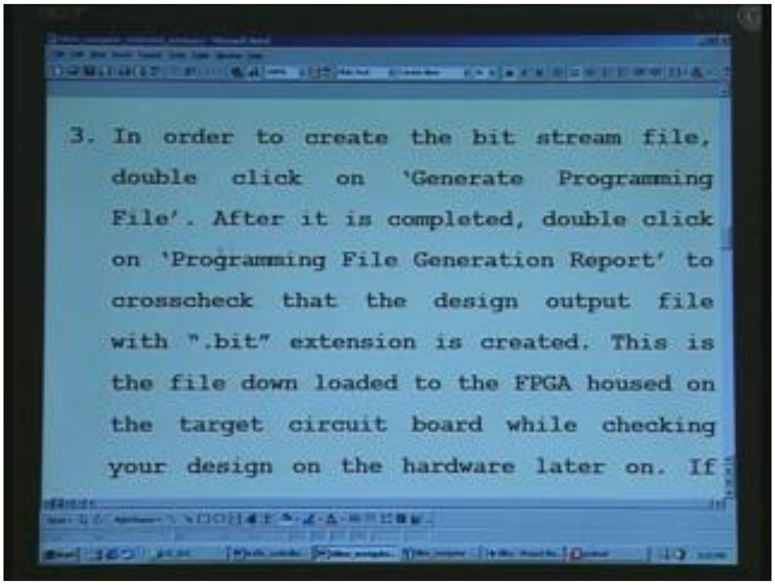
(Refer Slide Time: 33:26)


c.npl - [rtc_latest2.bgn (READ ONLY)]

```
-----------------------+-------
ActivateGclk           | No*
-----------------------+-------
PartialMask0           |  (Not S
```

Going to this part on the top, you see the report window that you have already seen. This is the report window and all the activities will be reported here. That is the meaning of saying log files here. We will go back to the command summary once again.

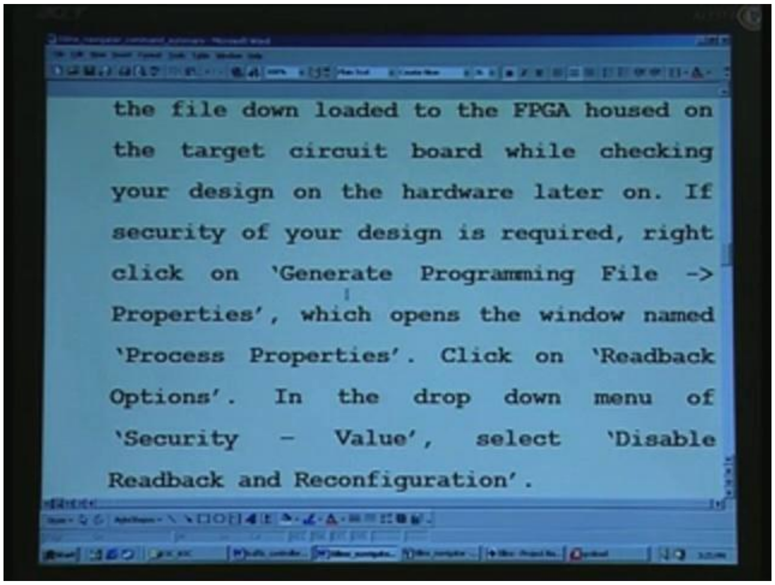
(Refer Slide Time: 33:47)



3. In order to create the bit stream file, double click on 'Generate Programming File'. After it is completed, double click on 'Programming File Generation Report' to crosscheck that the design output file with ".bit" extension is created. This is the file down loaded to the FPGA housed on the target circuit board while checking your design on the hardware later on. If

We have completed two points. We will see the third point. In order to create the bit stream file, double-click on Generate Programming File. After it is completed, double-click on Programming File Generation Report to crosscheck that the design output file .bit extension is created. That is over earlier and so a tick mark is got. In this file if you see towards the end, I will just read out – it may be difficult for you to read it yourself. What all it says is saving bit stream in rtc_alarm.bit. This implies that the bit stream generation is complete – that is what it reports towards the end. Coming back to the command summary, we were here. This is the file downloaded to the FPGA housed on the target circuit board while checking your design on the hardware later on.

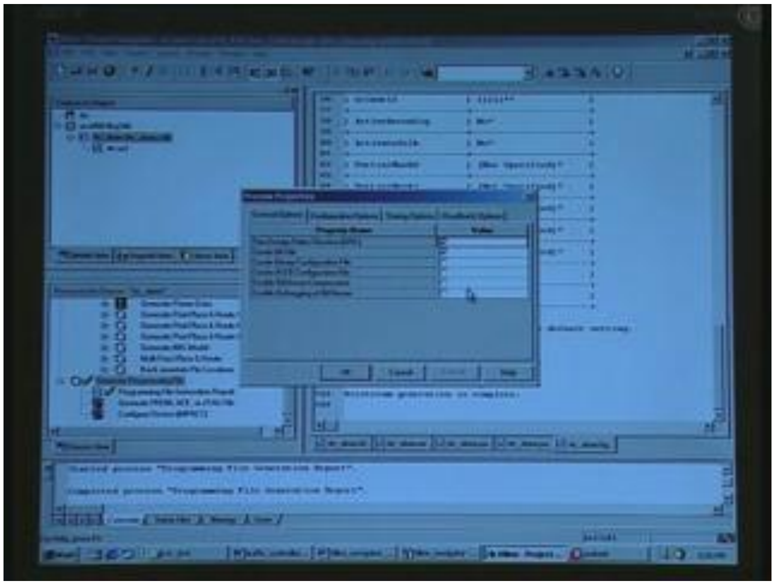
(Refer Slide Time: 34:42)



the file down loaded to the FPGA housed on the target circuit board while checking your design on the hardware later on. If security of your design is required, right click on 'Generate Programming File -> Properties', which opens the window named 'Process Properties'. Click on 'Readback Options'. In the drop down menu of 'Security – Value', select 'Disable Readback and Reconfiguration'.

If you want security of your design, you have to do the following procedure. If security of your design is required, right-click on Generate Programming File in Properties, which open the window named Process Properties. Click on Readback Options and in the drop-down menu of Security – Value, select Disable Readback and Reconfiguration.

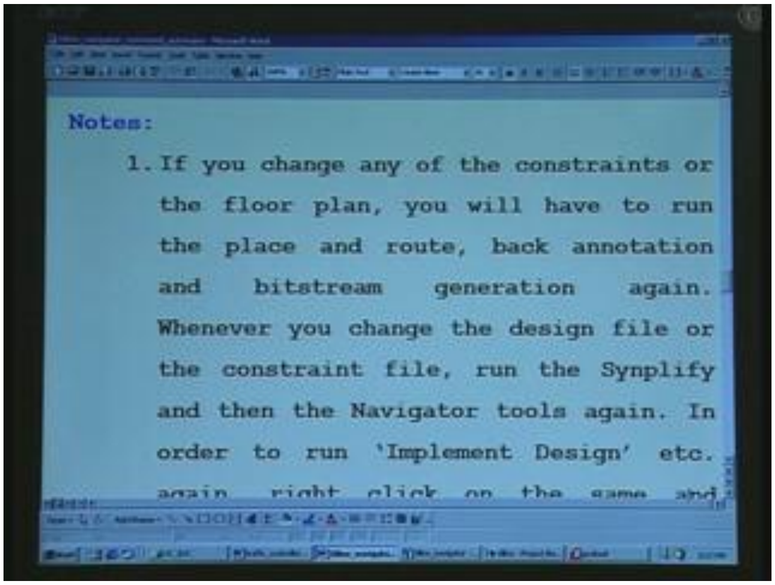
(Refer Slide Time: 35:04)



That is the in the project navigator. You have it here. If you right-click, you have Properties. The machine is rather slow, we will have to put up with that. What you have here is Property Name and Value. This is the window that opened and you see here enable Readback like this. If I click on this arrow, you have Disable Readback and reconfiguration. This is the option that I have mentioned in the text right now. You click on this and then say Ok, that is what is meant there. Again, there is a question mark because you have changed the same.

Coming back to this command summary, that is what we have seen here. Once you do this, the bit files cannot be read by other people. This will prevent piracy or stealing of your designs if you happen to be an IP core developer. This is a very vital thing for most of the companies.

(Refer Slide Time: 36:22)

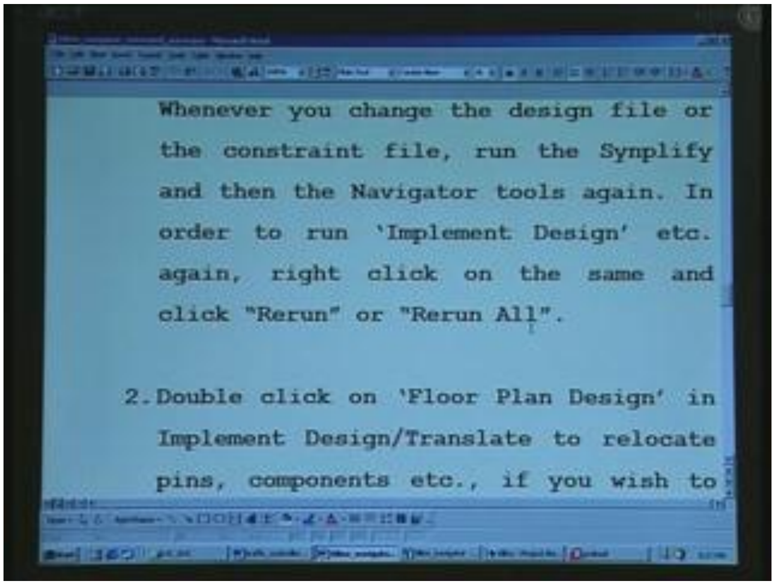


Notes:

1. If you change any of the constraints or the floor plan, you will have to run the place and route, back annotation and bitstream generation again. Whenever you change the design file or the constraint file, run the Synplify and then the Navigator tools again. In order to run 'Implement Design' etc. again right click on the same and

We have some notes, two notes here. If you change any of the constraints or the floor plan, you will have to run the place and route, back annotation and bit stream generation again. Whenever you change the design file or the constraint file, run Synplify and then the Navigator tools again.

(Refer Slide Time: 36:37)

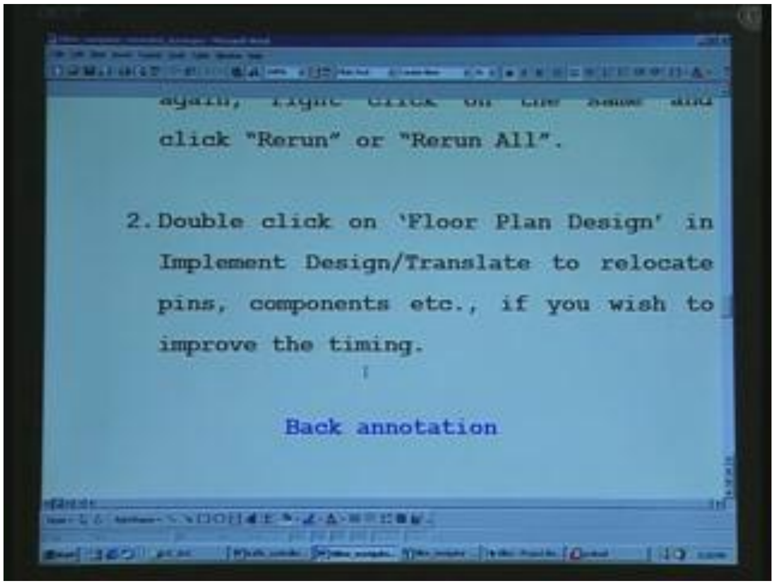


Whenever you change the design file or the constraint file, run the Synplify and then the Navigator tools again. In order to run 'Implement Design' etc. again, right click on the same and click "Rerun" or "Rerun All".

2. Double click on 'Floor Plan Design' in Implement Design/Translate to relocate pins, components etc., if you wish to

In order to run Implement Design etc., again right-click on the same and click Rerun or Rerun All. This is what we have already done.

(Refer Slide Time: 36:43)



again, right click on the same and click "Rerun" or "Rerun All".

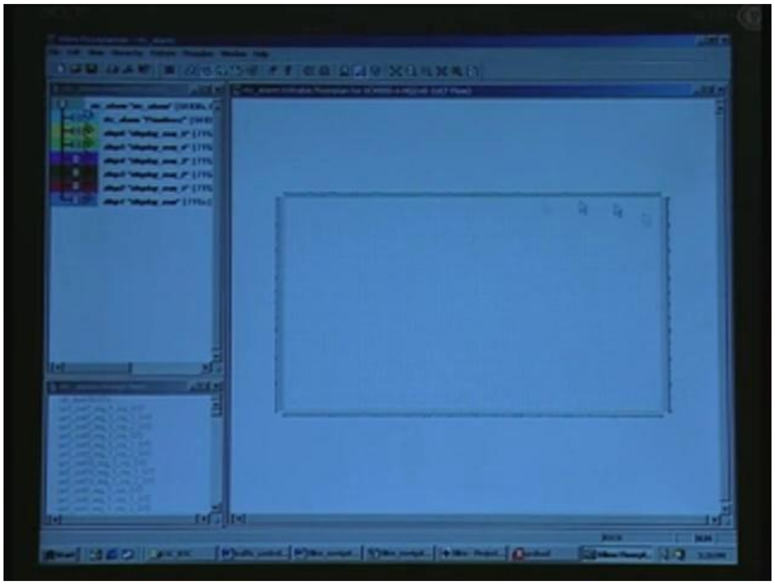
2. Double click on 'Floor Plan Design' in Implement Design/Translate to relocate pins, components etc., if you wish to improve the timing.

Back annotation

Double-click on Floor Plan Design in Implement Design or Translate to relocate pins, components etc., if you wish to improve the timing. We will have a glance at this as well.

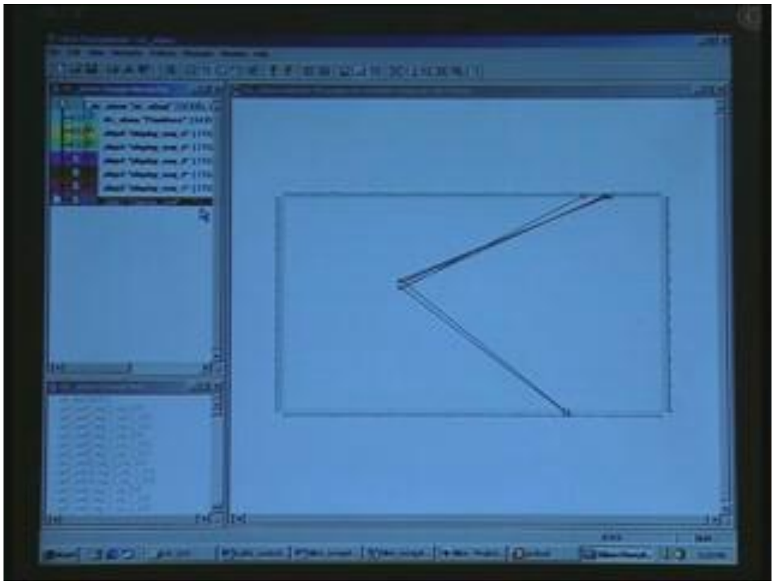
For example, at the implementation here, you see a floor plan design. If I double-click this, a floor plan will be opened here.

(Refer Slide Time: 37:13)



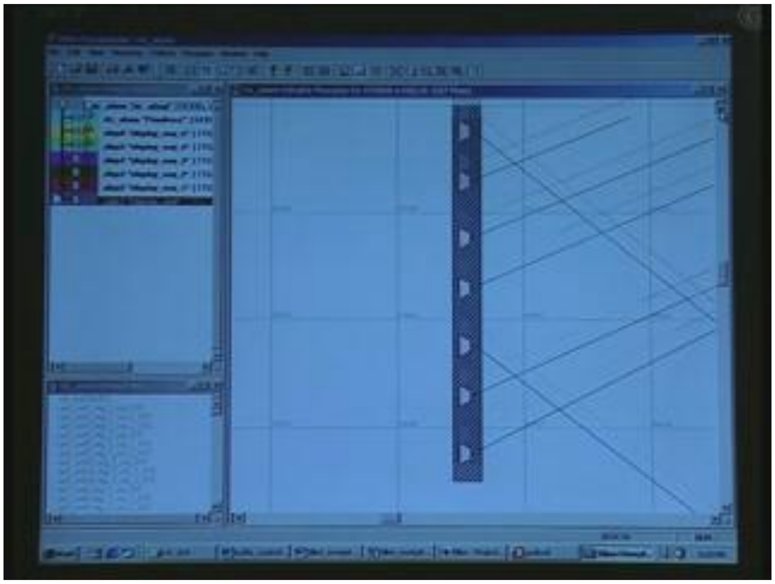
What you see here is the actual FPGA. For example, this is for the real-time clock that I mentioned. Suppose there are some displays on that, which you have already seen in the IO card that we have used earlier, which is right here. Suppose this is display 1.

(Refer Slide Time: 37:34)



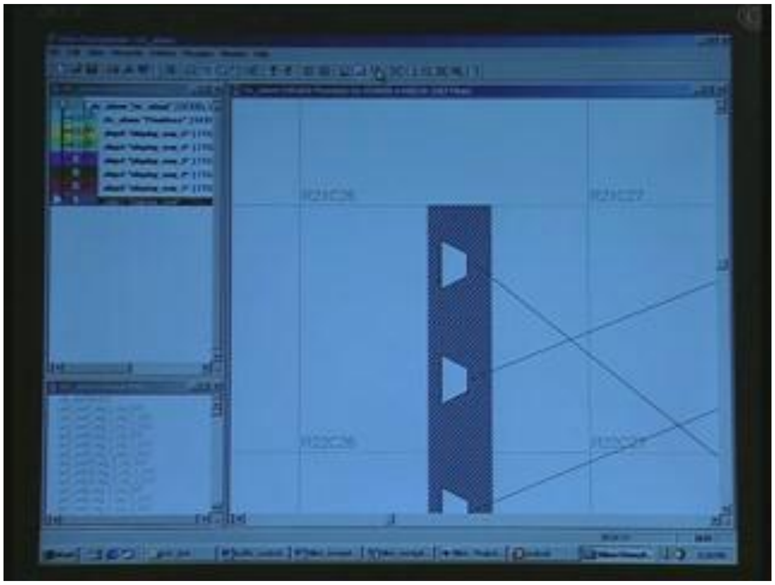
Suppose you want to get this here, what you can do is just click on the mouse and then hold it without releasing and get it into this area. Suppose you put it here, that part of the module that you have written as display code is portrayed here. The periphery depicts the actual pins. You can zoom on this. I will show this. For example, I want to zoom here. What I do is I just click on the right. You see a Zoom To box. I will just read out for you. I can just mark the window here and it will zoom there.

(Refer Slide Time: 38:08)



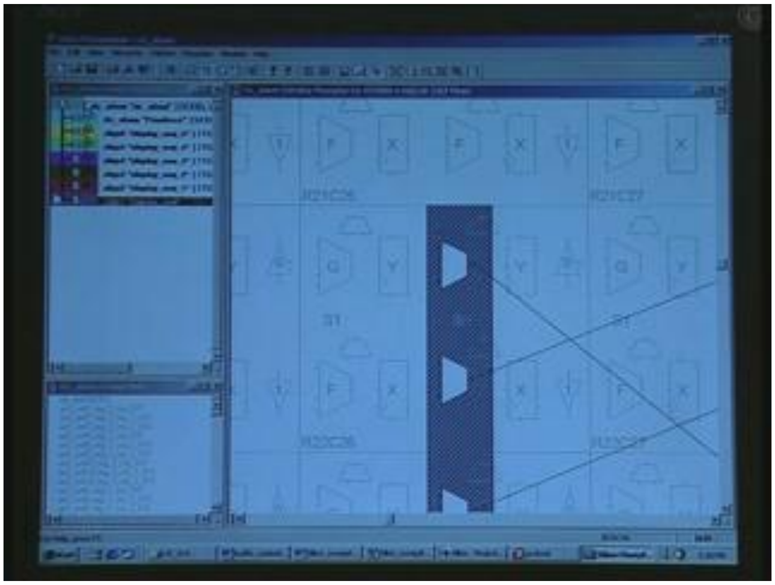
You have a plus and minus. You can use this zoom as well. You will just see that zooming happen now. You can see some faint character also appearing there. I am not sure whether this is a slice [38:26]. It is nothing but a collection of some MUX, etc. When I zoom, it will be clear to you.

(Refer Slide Time: 38:38)



This is the highest zoom that you have. Unfortunately, you cannot see much here. There is one more symbol here shown as a gate. If you click on this, let us see whether you get that.

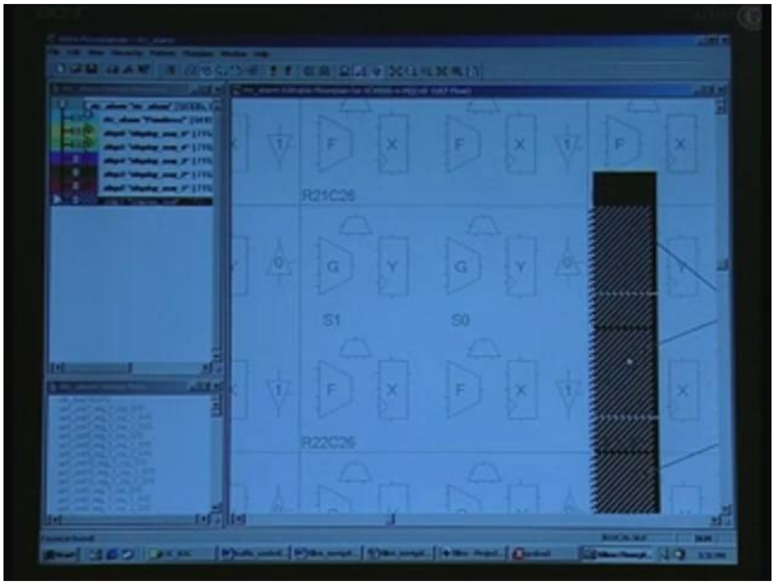
(Refer Slide Time: 38:53)



You see faint marks here. I am not sure whether you can see on the TV monitor there. You have a four-input MUX here and there are four of them – two are already here.

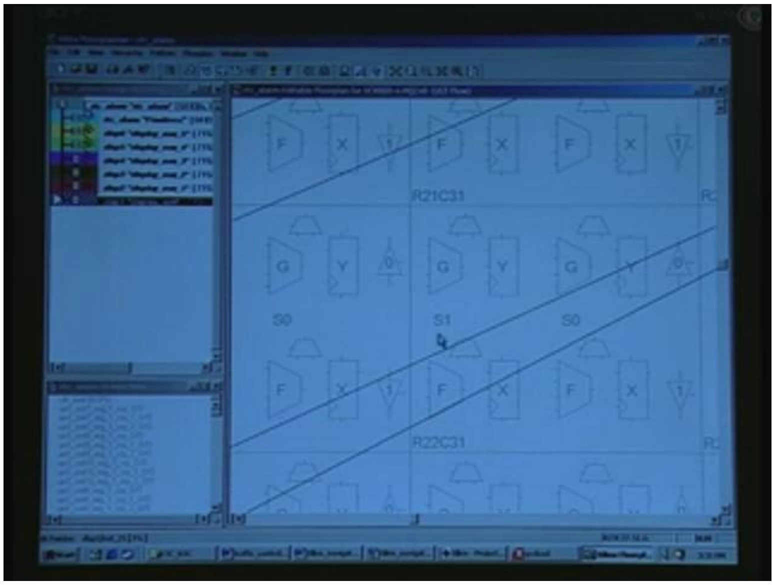
There are four flip-flops. Q, Q bar may be there, then Preset, clock input – all are available. In addition, four two-input MUX are also there. You can call this as one block of logic. I do not really know what the Xilinx terminology is.

(Refer Slide Time: 39:33)



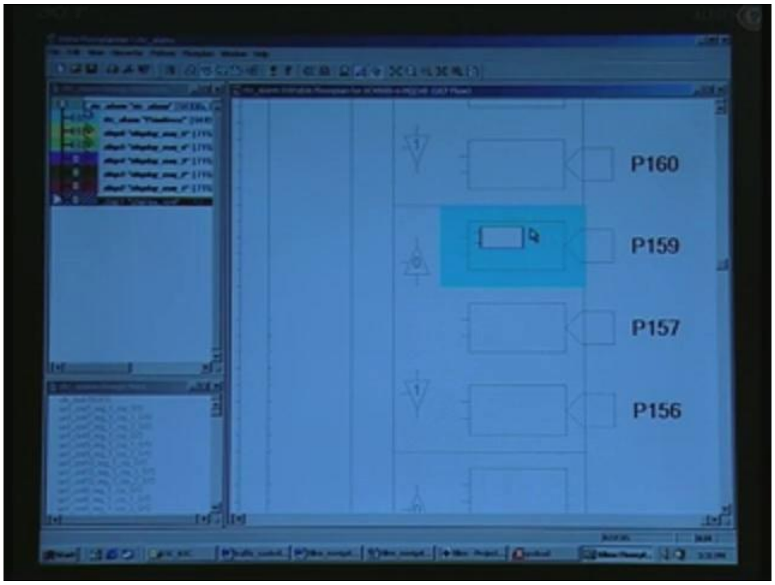
Suppose you want to relocate this from here to somewhere else, let us say here, you are free to do so. You have the flexibility of changing, that is, vetoing the Xilinx place and route – what is has already done. You can overrule that and place it elsewhere. Similarly, you can go on to a particular pin.

(Refer Slide Time: 39:49)



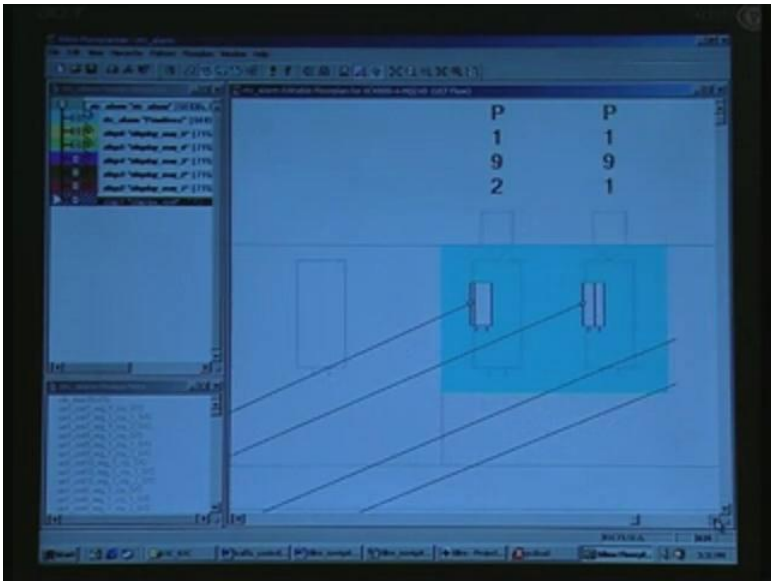
You see that the lines here are going precisely to the pins.

(Refer Slide Time: 39:57)



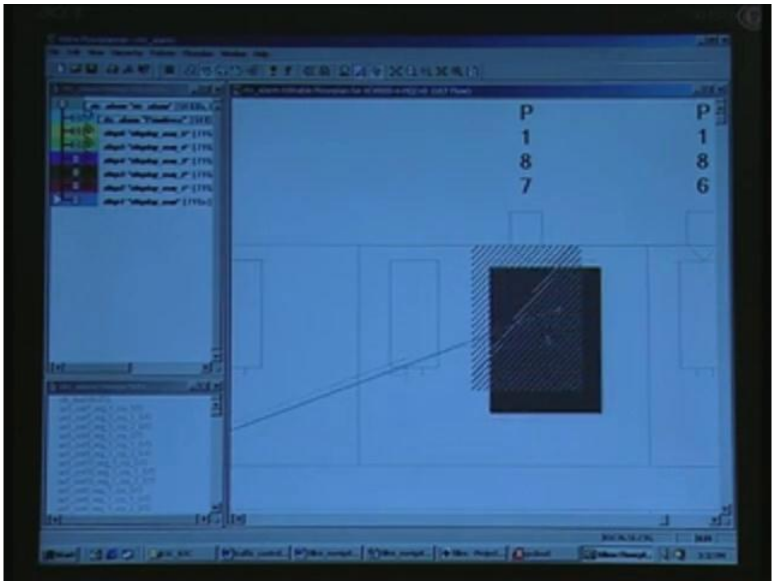
For example, one of the pins may be somewhere here. Let us see. There is a pin here but unfortunately, it is not connected to that. I hope we will be in a position to see one of the pins.

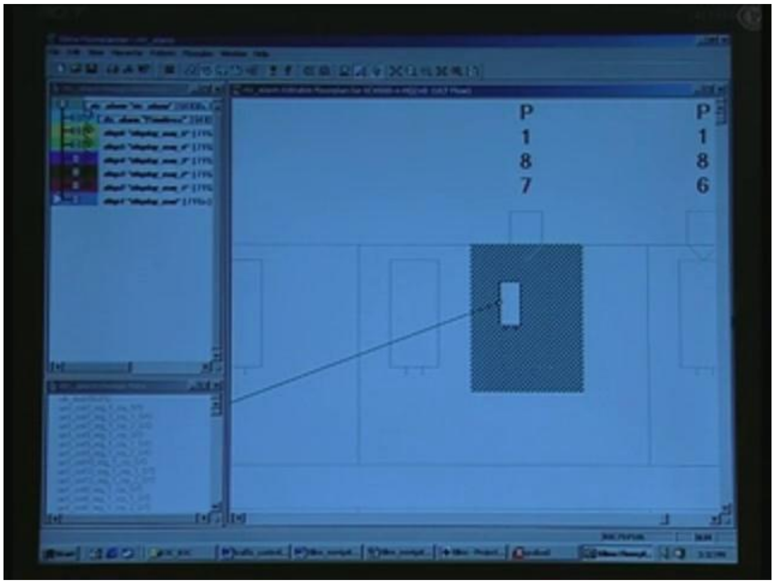
(Refer Slide Time: 40:13)



You can see here. Suppose I want to relocate these pins. I am not happy with the timing that I have got and I want to improve on the timing. What you do is you can just take this, pick it up and then locate it somewhere here. You can go to the pin where you want to locate.

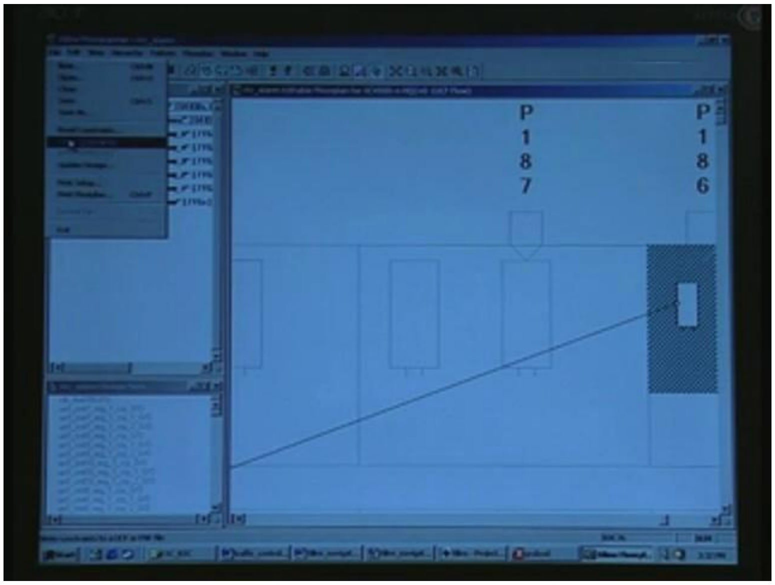
(Refer Slide Time: 40:36)





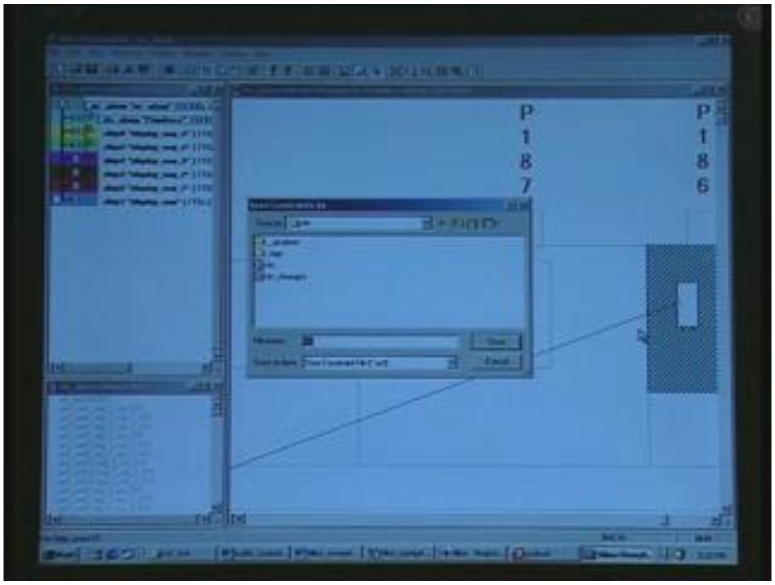
For example, I want to put it here. I just release it there. If I am not happy, I can change my mind n number of times and relocate anywhere.

(Refer Slide Time: 40:47)



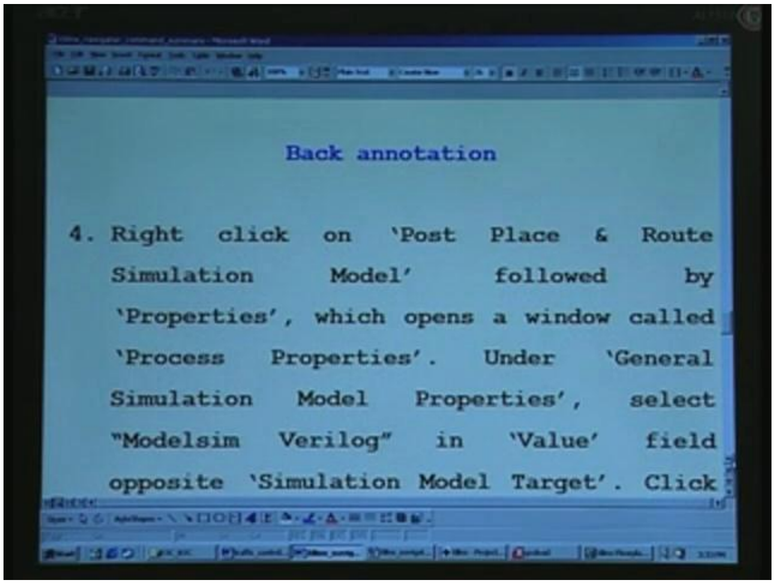
Under File, you have what is called the right constraints file.

(Refer Slide Time: 40:58)



If you click on the constraints file, you can save the very constraint file. These are all relocated pins and once again, you have to run the implement. After that, the new changes that you have made in the form of relocating either the components or the pin will be shown here. This is the floor plan. This will be done only at the fag end of your project.

(Refer Slide Time: 41:30)

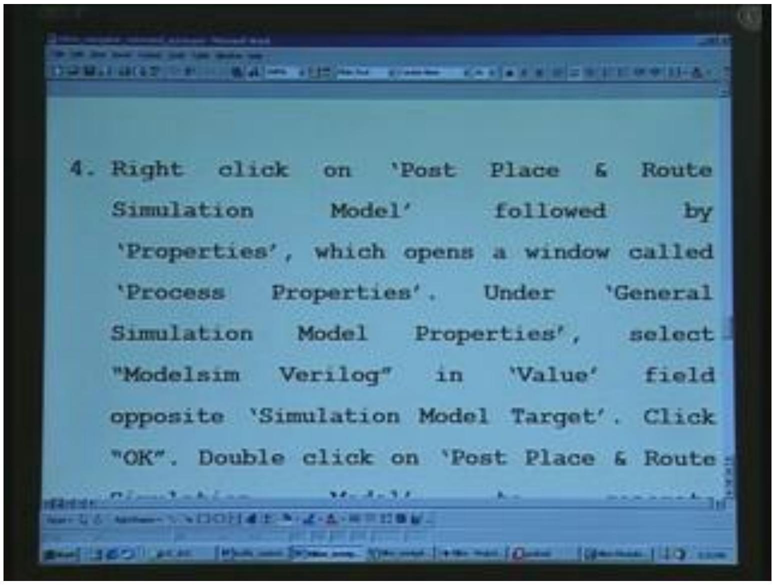


Back annotation

4. Right click on 'Post Place & Route Simulation Model' followed by 'Properties', which opens a window called 'Process Properties'. Under 'General Simulation Model Properties', select "Modelsim Verilog" in 'Value' field opposite 'Simulation Model Target'. Click

Coming back to the command summary here, we have already seen how to back annotate using the Xilinx design manager earlier. Now we are going to see the Navigator.

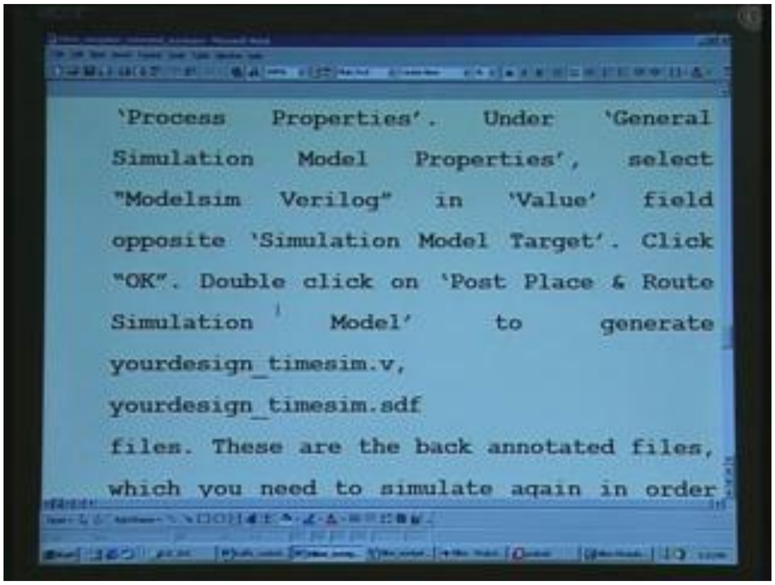
(Refer Slide Time: 41:37)



4. Right click on ‘Post Place & Route Simulation Model’ followed by ‘Properties’, which opens a window called ‘Process Properties’. Under ‘General Simulation Model Properties’, select "Modelsim Verilog" in ‘Value’ field opposite ‘Simulation Model Target’. Click "OK". Double click on ‘Post Place & Route

This is the back annotation. Right-click on Post Place and Route Simulation Model, followed by Properties, which opens a window called Process Properties, which we have already seen. Under General Simulation Model Properties, select Modelsim Verilog. In that, part of the menu will be this Modelsim selection and you will have to select this particular thing in Value field opposite Simulation Model Target.

(Refer Slide Time: 42:04)

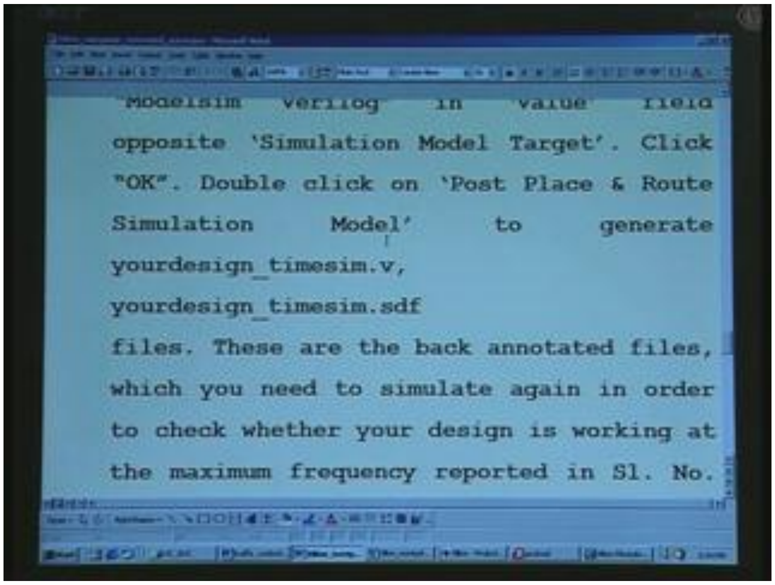


'Process Properties'. Under 'General Simulation Model Properties', select "Modelsim Verilog" in 'Value' field opposite 'Simulation Model Target'. Click "OK". Double click on 'Post Place & Route Simulation Model' to generate yourdesign_timesim.v, yourdesign_timesim.sdf files. These are the back annotated files, which you need to simulate again in order

Click Ok and double-click on Post Place and Route Simulation Model to generate…. Suppose you have an rtc as your design – I have just mentioned your design. You put your real design here. What we are talking about is to back annotate. Back annotation is the process by which gate delays are reflected. That is the true reflection of your final FPGA hardware that is going to be. Do not be misled by the simulation running at Gigahertz. In future, we will have a look at that as well.

After you do this, it creates the back annotated file. The Xilinx Place And Route Navigator tool that you are seeing now is what is going to generate this after you run this. Just now, we have just covered this. It creates two files. One is the _timesim. If your design is rtc, you put rtc_timesim.v and rtc_timesim.sdf is the one that contains delay information. Standard delay format is the expansion of this and .v is the back annotated file. If you open this file, you will see that all primitives are used. Primitives refers to what has been put in the FPGA such as MUX, flip-flops and so on.

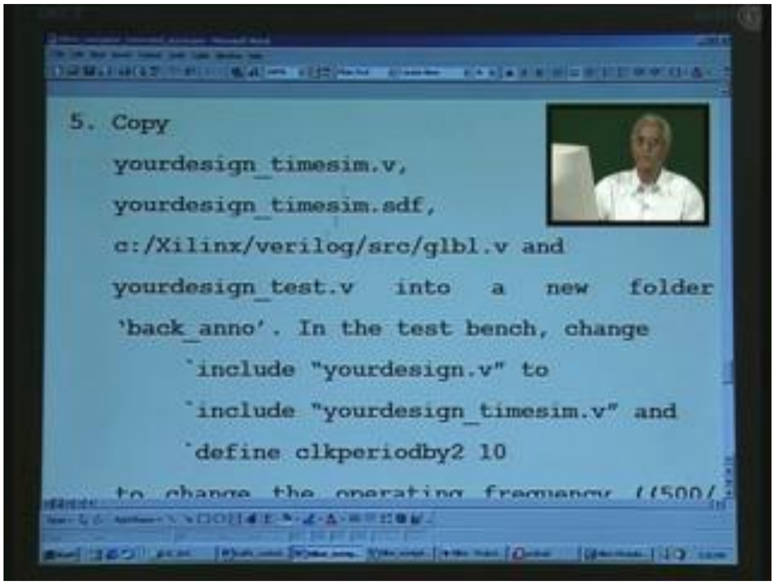
(Refer Slide Time: 43:29)



'Modelsim Verilog' in 'Value' field opposite 'Simulation Model Target'. Click "OK". Double click on 'Post Place & Route Simulation Model' to generate yourdesign_timesim.v, yourdesign_timesim.sdf files. These are the back annotated files, which you need to simulate again in order to check whether your design is working at the maximum frequency reported in Sl. No.

These are back annotated files, which you need to simulate again in order to check whether your design is working at the maximum frequency reported in serial number 2 or not. We have already got that maximum frequency reported in one of the files, serial number 2 – we have already seen that.

(Refer Slide Time: 43:43)



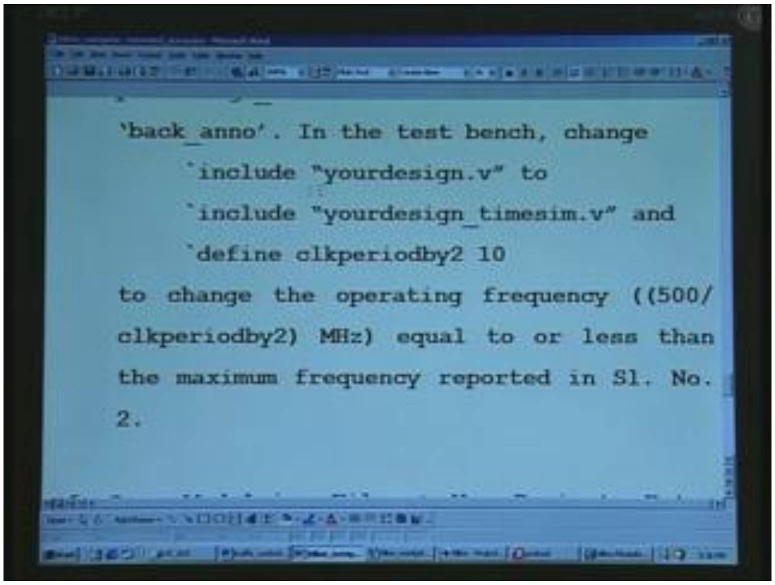
5. Copy yourdesign_timesim.v, yourdesign_timesim.sdf, c:/Xilinx/verilog/src/glbl.v and yourdesign_test.v into a new folder 'back_anno'. In the test bench, change

```
`include "yourdesign.v" to
`include "yourdesign_timesim.v" and
`define clkperiodby2 10
```

to change the operating frequency ((500/

Although it is not necessary to copy it in a different folder, it is a good practice to do so because you will not confuse it with your design file, and you especially need to modify the test bench because now it has to reflect the back annotated file instead of your actual design. For example, if it is rtc, this is what this Xilinx tool has created. It is a good practice to have it in a different folder and so you put these two back annotated files that you have just seen earlier. A library file is also to be put. This is available in this path: c, Xilinx, Verilog, source and then global. They are all global variables. This is mandatory because this contains the information of all the primitives such as MUX, etc., which are being used in the actual FPGA device. Also, copy your design_test (this is the test bench we have already seen) into a new folder back_anno. Just give some appropriate name as I have done here.

(Refer Slide Time: 44:52)



'back_anno'. In the test bench, change

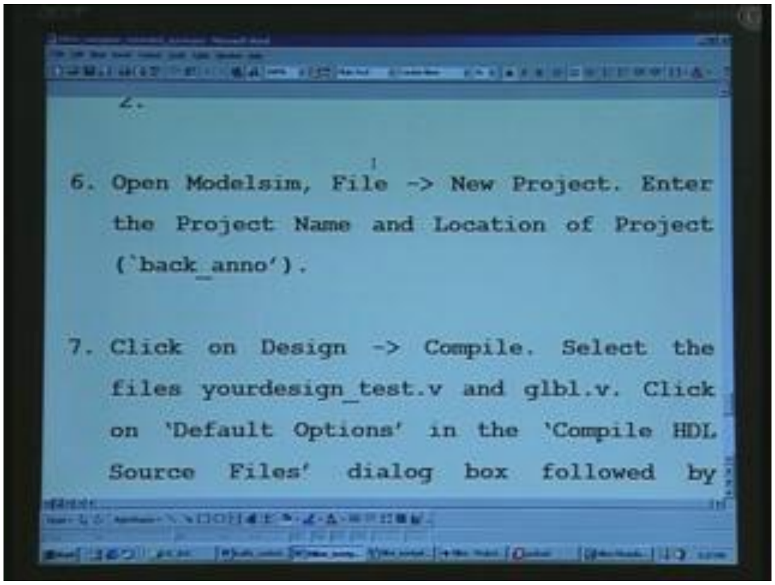
    `include "yourdesign.v" to
    `include "yourdesign_timesim.v" and
    `define clkperiodby2 10

to change the operating frequency ((500/clkperiodby2) MHz) equal to or less than the maximum frequency reported in Sl. No. 2.

In the test bench, you have to make the following changes. This is the statement for including your design. For example, rtc.v is actually a design here and this was the original file in your test bench. Now, you have to change this as your design_timesim.v. You are replacing this file by this back annotated file. You also have a clock there. It was defined as clckperiodby2 and this 10 nanoseconds implies is 20 nanosecond frequency. As a ready reckoner, I have given a formula here: 500 divided by clckperiodby2 will straightaway give you the value in MHz. For example, if you want for 10, you get 50

MHz here. Equal to or less than the maximum frequency reported in serial number 2 which we have already seen. See that the value of the frequency that you select is such that the operation frequency is less than what is reported as the maximum frequency.

(Refer Slide Time: 45:59)



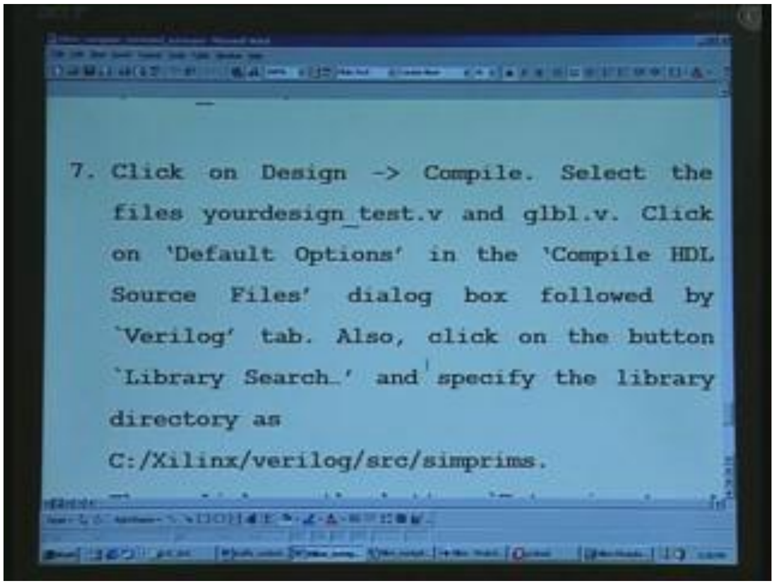
2.

6. Open Modelsim, File -> New Project. Enter the Project Name and Location of Project (`back_anno').

7. Click on Design -> Compile. Select the files yourdesign_test.v and glbl.v. Click on 'Default Options' in the 'Compile HDL Source Files' dialog box followed by

Then, as we have seen earlier while using the design manager and now using the very same thing, we are doing using the Navigator, you need to invoke the Modelsim in order to load your back annotated file, which will actually reflect the gate delays. Without this, I once again maintain that your project is not complete unless you go through the entire exercise of back annotation. Open Modelsim, File, New Project. Enter the project name and location of project as back_annotate.

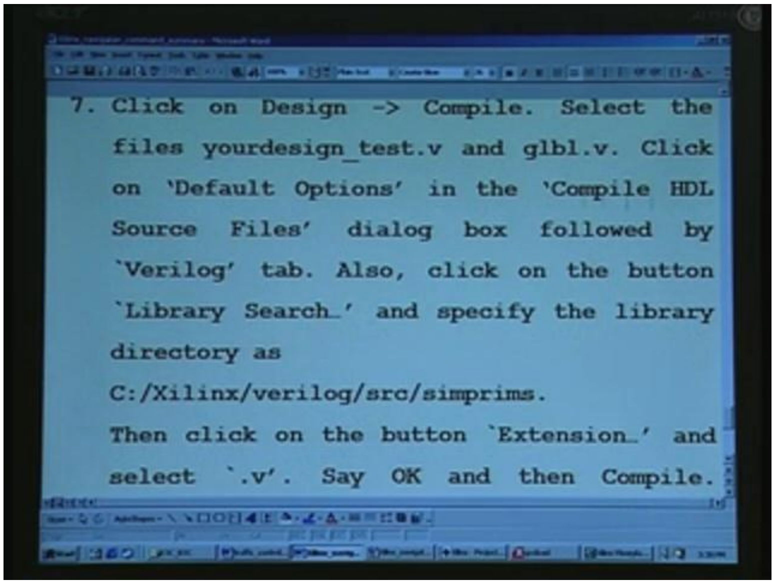
(Refer Slide Time: 46:31)



7. Click on Design -> Compile. Select the files yourdesign_test.v and glbl.v. Click on ‘Default Options’ in the ‘Compile HDL Source Files’ dialog box followed by `Verilog’ tab. Also, click on the button `Library Search…’ and specify the library directory as C:/Xilinx/verilog/src/simprims.

Click on Design, Compile. We have already seen this n number of times earlier. Select the files yourdesign_test.v, then glbl.v. I will not show the Modelsim again. I am just going to just read out before winding this up. Click on Default Options in the Compile HDL Source Files. Everything is given verbatim. You will have no difficulty in following the same sequence when you handle the tool.

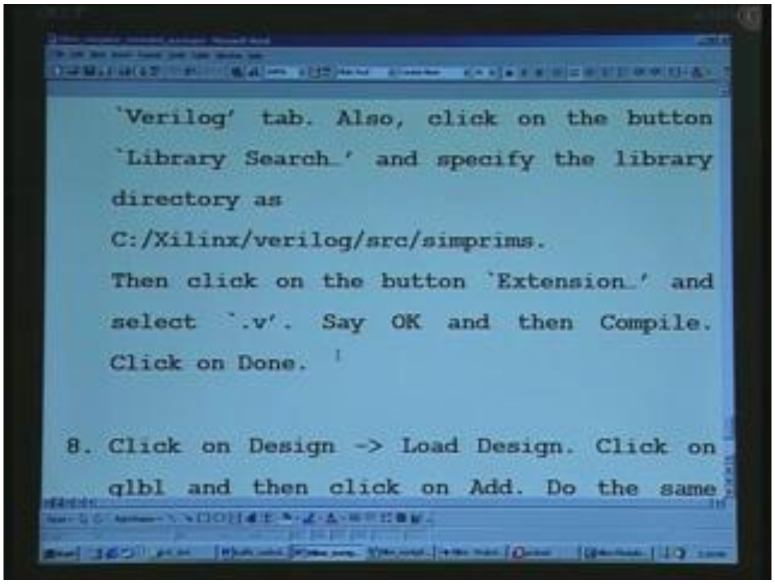
(Refer Slide Time: 46:58)



7. Click on Design -> Compile. Select the files yourdesign_test.v and glbl.v. Click on ‘Default Options’ in the ‘Compile HDL Source Files’ dialog box followed by `Verilog’ tab. Also, click on the button `Library Search…’ and specify the library directory as C:/Xilinx/verilog/src/simprims. Then click on the button `Extension…’ and select `.v’. Say OK and then Compile.

Compile HDL Source Files dialog box followed by Verilog tab. When you do this, also click on the button Library Search and specify the library directory as C:/Xilinx/verilog/src/simprims. In fact, in the earlier command summary, whatever you have used for back annotation will also work as it is. This is only an alternative I am giving here. These are all menu-driven, so you have only to click the mouse – keep on clicking whereas earlier, we used to key the actual commands.

(Refer Slide Time: 47:35)

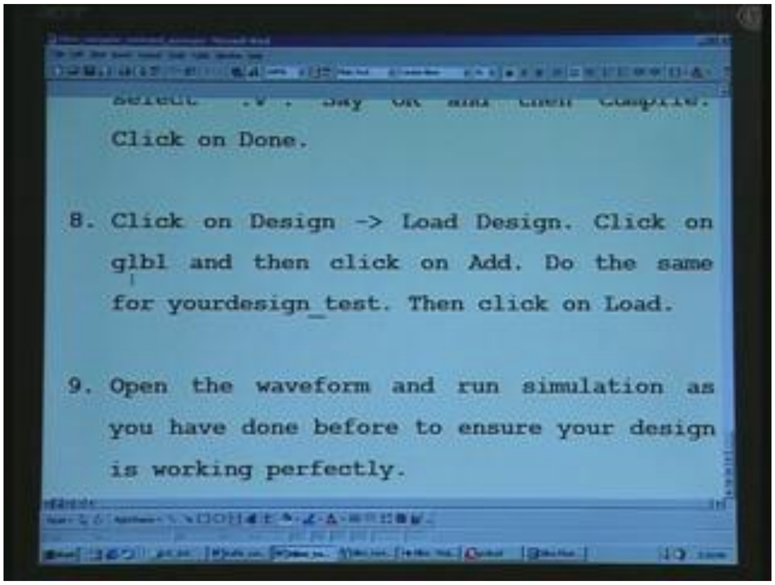


`Verilog' tab. Also, click on the button `Library Search…' and specify the library directory as C:/Xilinx/verilog/src/simprims.
Then click on the button `Extension…' and select `.v'. Say OK and then Compile.
Click on Done.

8. Click on Design -> Load Design. Click on glbl and then click on Add. Do the same

This is an alternative approach. If you do not want to use this technique, you can use the previous ones. Specify directory as this: Xilinx/verilog/src/simprims. Then click on the button extension and select .v. Say OK and then Compile. Click on Done. This compilation is over, taking into consideration the back annotated file.

(Refer Slide Time: 48:00)



Select .v . Say OK and then Compile. Click on Done.

8. Click on Design -> Load Design. Click on glbl and then click on Add. Do the same for yourdesign_test. Then click on Load.

9. Open the waveform and run simulation as you have done before to ensure your design is working perfectly.

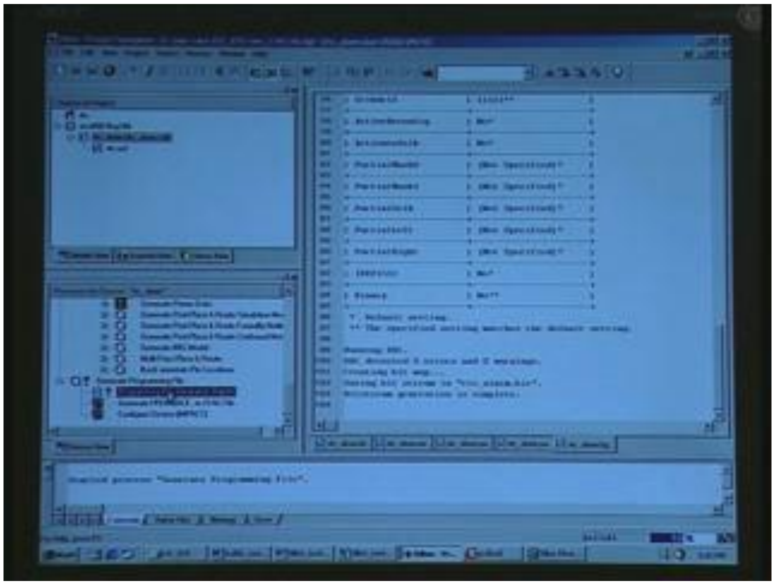
The next step is automatically to load your design, which is done by clicking on Design, Load Design. Click on glbl. Here, follow it very carefully. If you violate this sequence, it will report an error. Click on glbl, it will list in the same folder, then click on add. After this, do not press the load immediately. Be patient and do the second thing also – do the same for yourdesign_test. This is the test bench, right?

First is that glbl, followed by then say add, then your design. In the executing window at the bottom, you will have both the files located in the same opened window. Only then, click on load. If you just click after this, promptly you will get an error, so take care here. There is no other difficulty other than this here. If you follow this sequence, there will be no difficulty.

Then open the waveform and run simulation as you have done before, to ensure that your design is working perfectly. We have seen the command summary for this Navigator, which is actually a replacement for design managers. I think this is what you have to handle in the latest software. If you happen to procure new software, you have to adopt this technique. It is worth it, it will not take much time – you can very easily learn in an hour. This is the projected view, which we have already seen (Refer Slide Time: 49:40).

If you have any specific questions on what we have covered, you are free to ask. Do you have any questions?

(Refer Slide Time: 49:55)

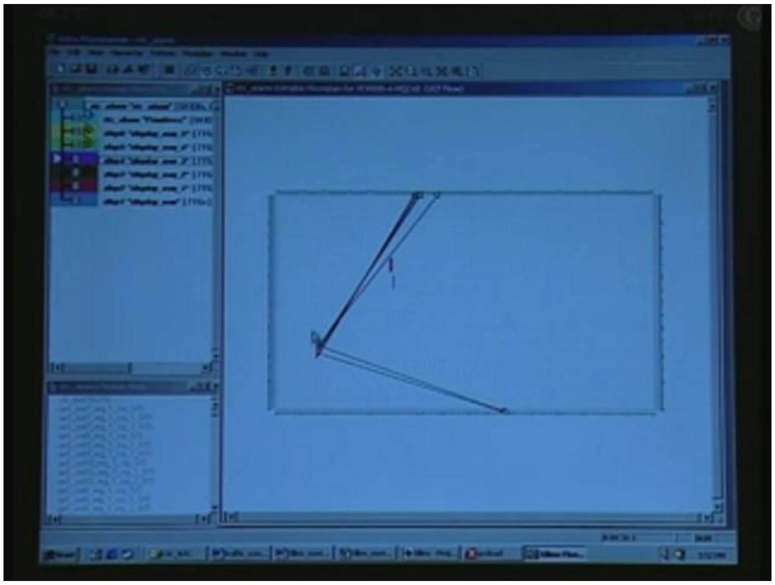


I think we have seen all the report files listed here and also the Generate File, although a question mark is put here. We have created a .bit file. We have opened the report file and then seen this. If you want, you can rerun, for we have another minute or so. You can just watch what is happening at this window. It has already stopped, so I think we will have to rerun some of this. You have a question?

Question (by Student) : The question is this. As a designer, when should I move around the components or move around the MUXs? After every design, should I check out whether I should do some manually?

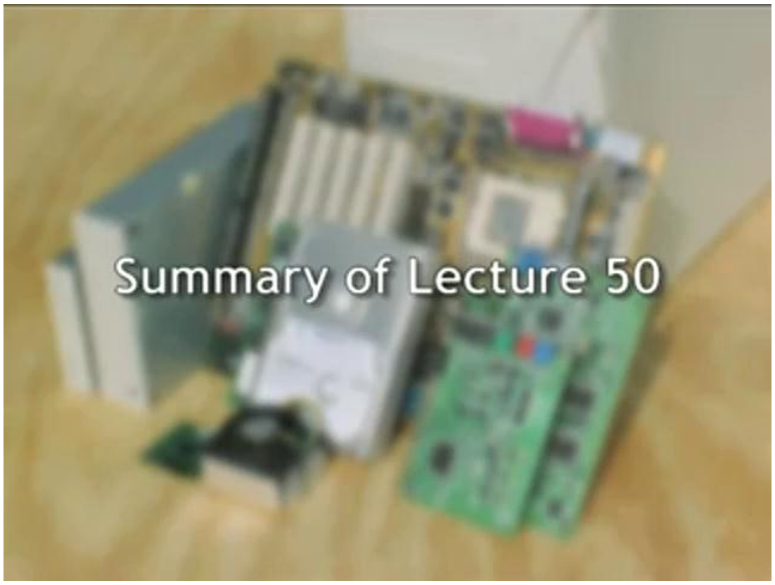
If you recollect what I have spoken several lectures before, while explaining the entire design flow, any design is an iterative process. You have to run through all the tools, start coding, run through all the tools, repeat this n number of times till you are totally satisfied both in the simulation point as well as the synthesis as well as the place and route. In spite of running all through, the place and route tool will not put it very efficiently. After all, no tool can equal human intelligence. It is a human who can make the right judgment. A human is vitally interested in a particular project solely from performance point of view – how efficiently a design is working, how fast the design is working. If you are interested in making a design that will work at the optimum speed, that is the highest possible speed, naturally it calls for intervention from the human. The designer will have to interact and interactively relocate so that the end goal of jacking up the speed of the system is actually done. From this point of view, we are trying to relocate not just to play around with the tool. It is not to play but out of the dire necessity that we want to enhance the speed of operation.

(Refer Slide Time: 52:35)

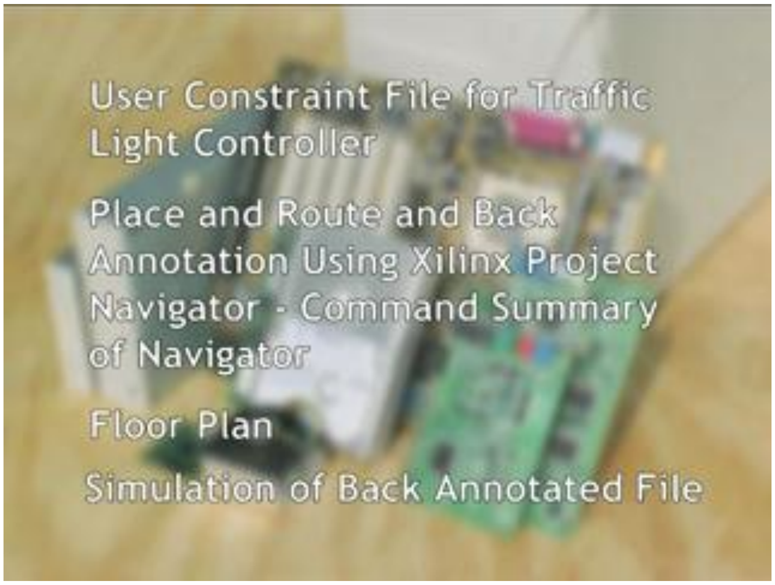


For example, you can see different modules are distributed over here. The pins are connected over a long distance here. This would naturally be an impediment for the high speed of operations. You have to relocate these pins as close as possible so that you may enhance the speed. This is the reason why we want to relocate both the modules as well as pins. Thank you.

(Refer Slide Time: 53:03)



Summary of Lecture 50



User Constraint File for Traffic Light Controller

Place and Route and Back Annotation Using Xilinx Project Navigator - Command Summary of Navigator

Floor Plan

Simulation of Back Annotated File