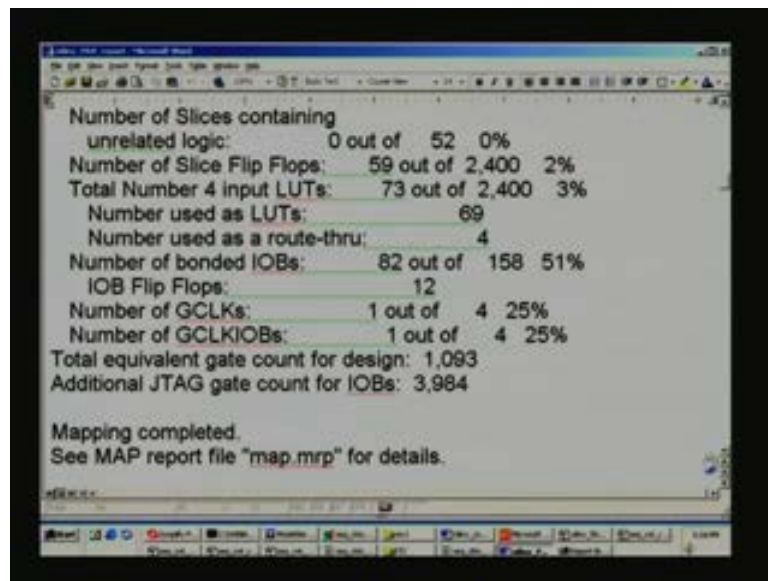


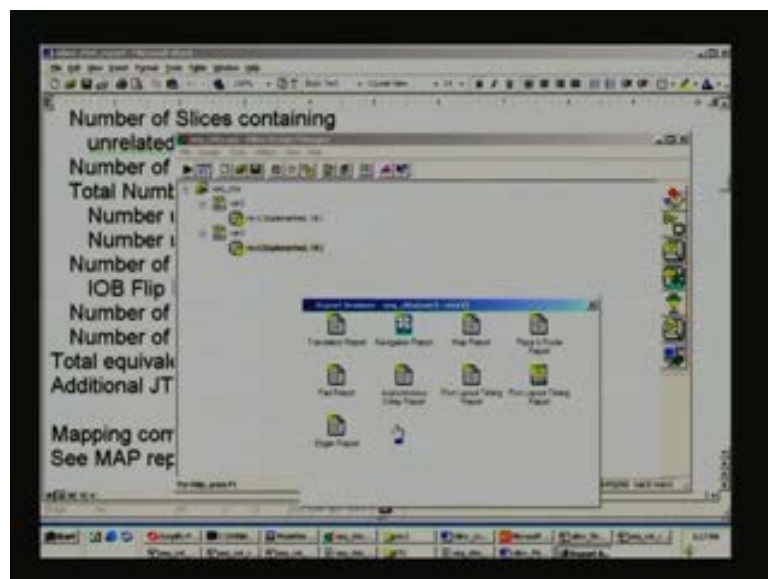
Digital VLSI System Design
Prof. Dr. S. Ramachandran
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture No. # 34
Xilinx Place and Route Tool

(Refer Slide Time: 02:27)

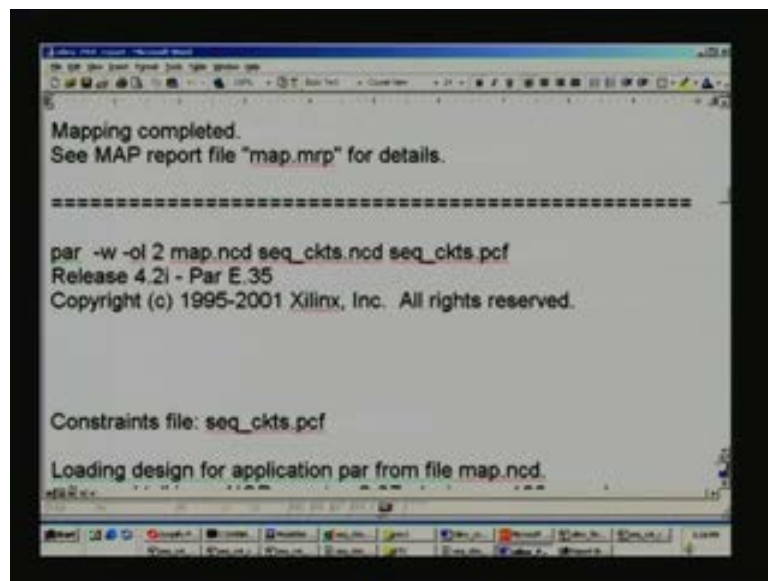


(Refer Slide Time: 02:33)



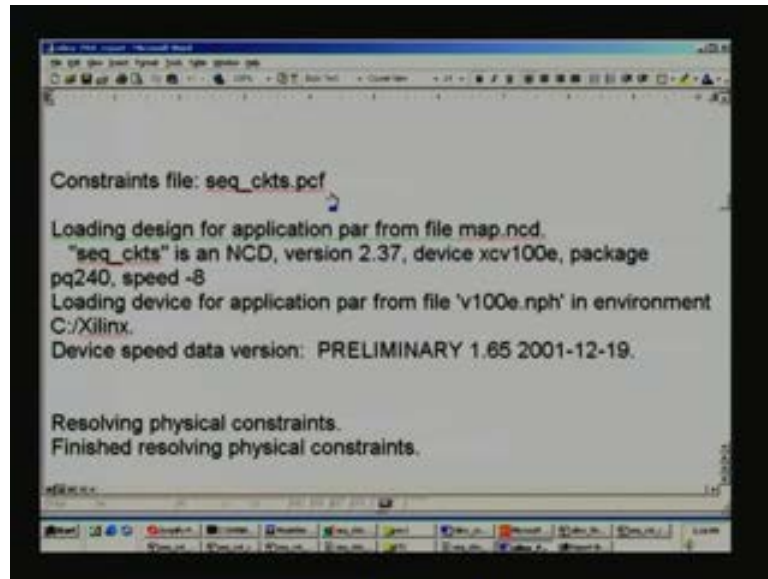
We were looking at the log file of Xilinx place and route and we will continue with same now. **and** This is the Xilinx's window which we had a look. We were in version 3 and division 1 and we had run and it has successfully completed the place and route and we were inspecting the log report. From here, we can see, you can base a what is called utilities and 1 report browser there, if you click on this, you can get the report even after you closed after running replace and route earlier.

(Refer Slide Time: 03:50)

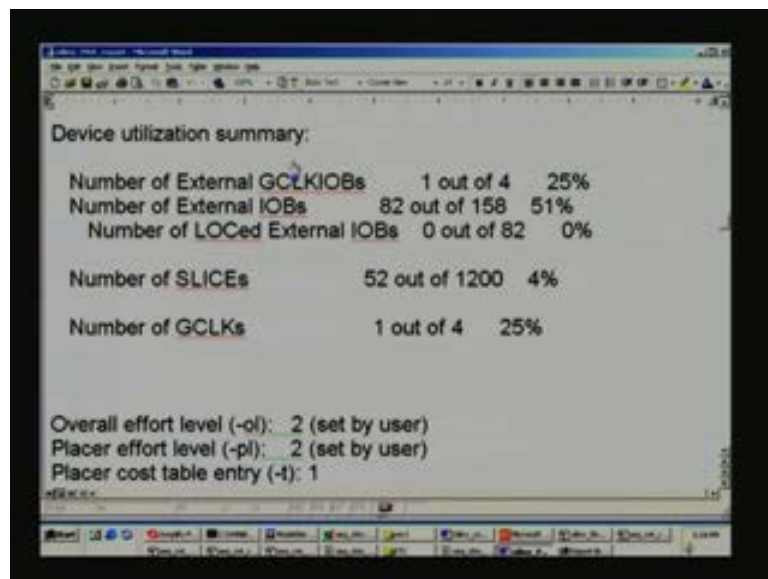


You can click one after another and inspect all this report or we have already copied this log file here and we were just mentioning that total equal and get for the design is just 1093 gates. Whereas, some additional gates are require for JTAG compatible IOBs and that is about 4000 gates that you need. For larger design, this will be much larger when compare to this and will just continue with this report. So, we already mention that what is being reported to at this point of time is that the device is being mapped, and for that, a constraint files or also created.

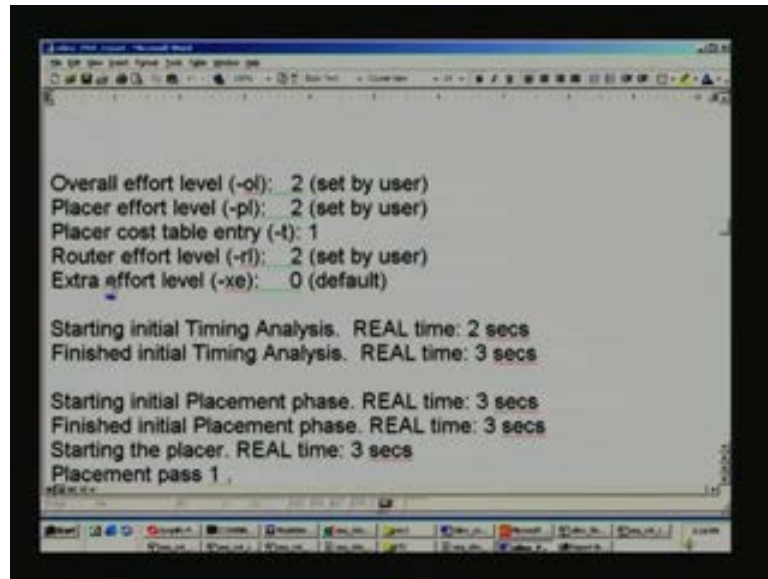
(Refer Slide Time: 04:10)



(Refer Slide Time: 04:21)



(Refer Slide Time: 04:33)



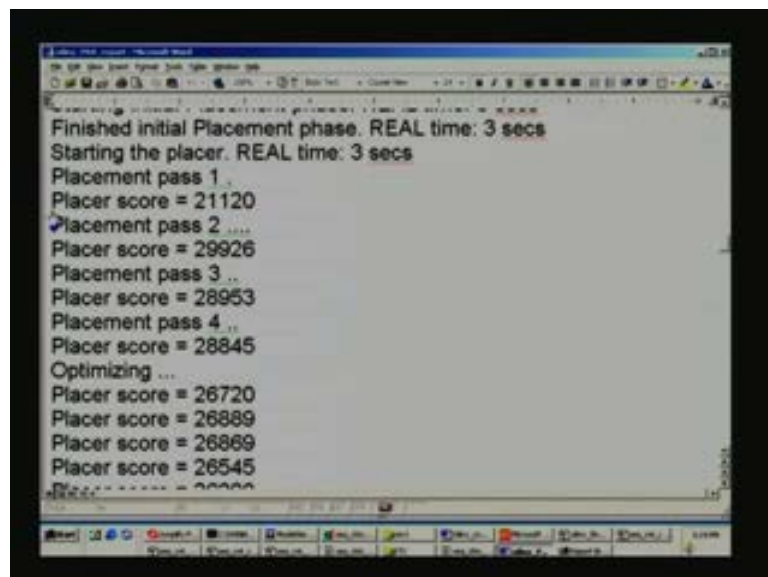
```
Overall effort level (-of): 2 (set by user)
Placer effort level (-pl): 2 (set by user)
Placer cost table entry (-t): 1
Router effort level (-rf): 2 (set by user)
Extra effort level (-xe): 0 (default)

Starting initial Timing Analysis. REAL time: 2 secs
Finished initial Timing Analysis. REAL time: 3 secs

Starting initial Placement phase. REAL time: 3 secs
Finished initial Placement phase. REAL time: 3 secs
Starting the placer. REAL time: 3 secs
Placement pass 1 .
```

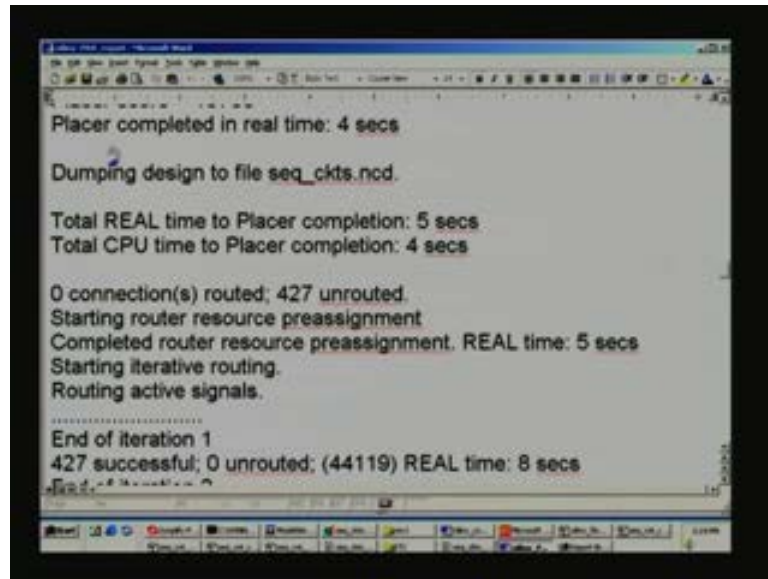
These the constraint file - pcf file - created by one of the phases of place and route. Here, once again it reports nearly the number of clocks and number of slices; we have already seen what slices are. In order to place and route it all iterative process and requires a constraint amount of time as we had mention earlier, and this are all, I mean this will keep going in a cyclic fashion. So, as we see here.

(Refer Slide Time: 04:56)



```
Finished initial Placement phase. REAL time: 3 secs
Starting the placer. REAL time: 3 secs
Placement pass 1 .
Placer score = 21120
Placement pass 2 ....
Placer score = 29926
Placement pass 3 ..
Placer score = 28953
Placement pass 4 ..
Placer score = 28845
Optimizing ...
Placer score = 26720
Placer score = 26889
Placer score = 26869
Placer score = 26545
Placer score = 26545
```

(Refer Slide Time: 05:16)

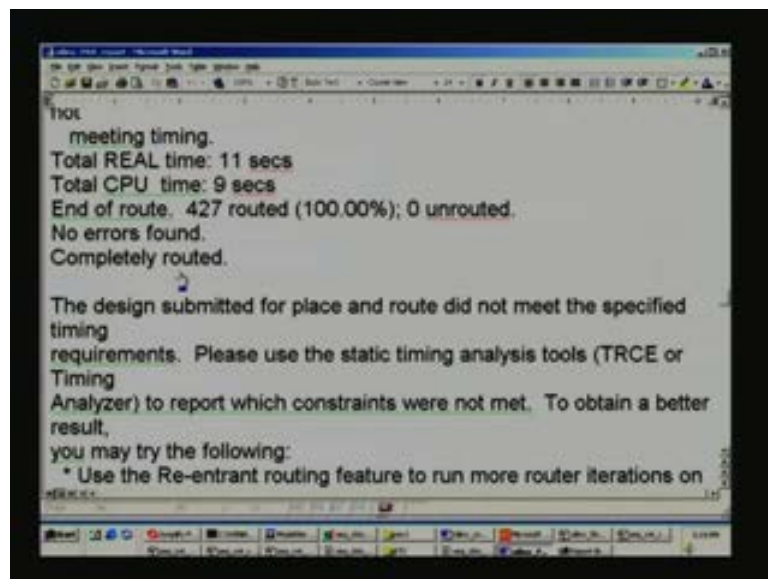


```
Placer completed in real time: 4 secs
Dumping design to file seq_ckts.ncd.
Total REAL time to Placer completion: 5 secs
Total CPU time to Placer completion: 4 secs

0 connection(s) routed; 427 unrouted.
Starting router resource preassignment
Completed router resource preassignment. REAL time: 5 secs
Starting iterative routing.
Routing active signals.

.....
End of iteration 1
427 successful; 0 unrouted; (44119) REAL time: 8 secs
```

(Refer Slide Time: 05:41)

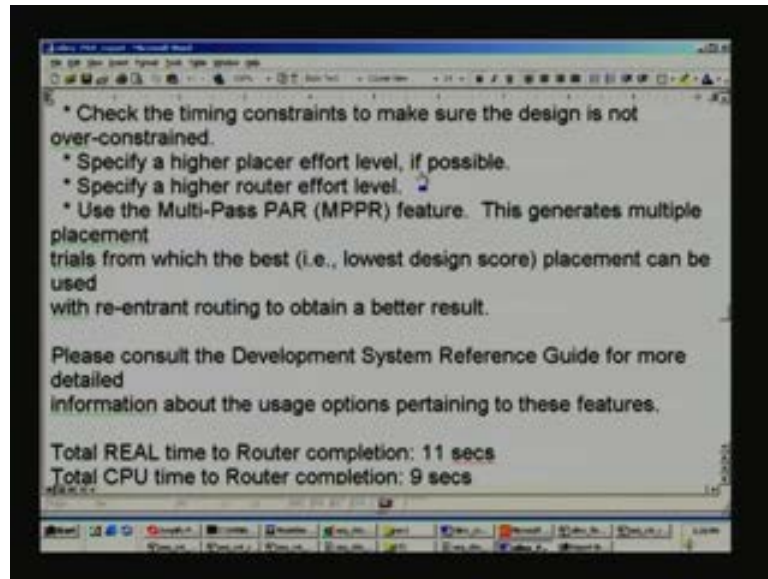


```
not
meeting timing.
Total REAL time: 11 secs
Total CPU time: 9 secs
End of route. 427 routed (100.00%); 0 unrouted.
No errors found.
Completely routed.

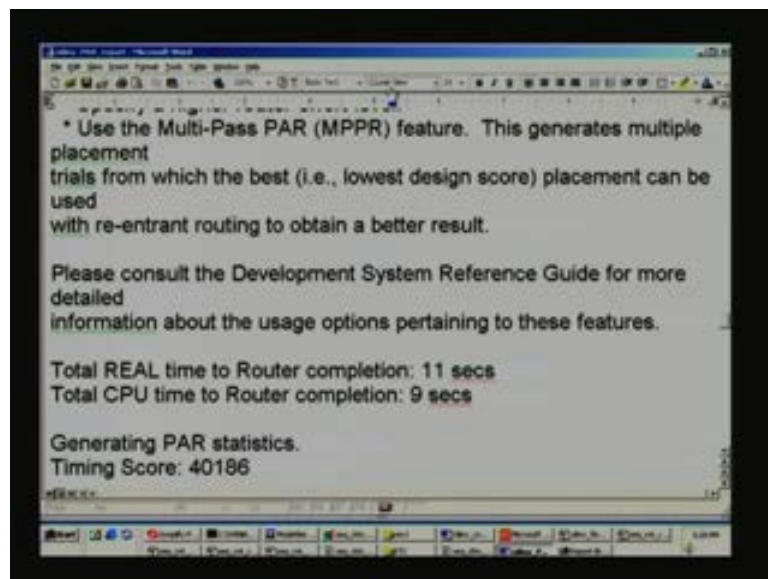
The design submitted for place and route did not meet the specified
timing requirements. Please use the static timing analysis tools (TRCE or
Timing Analyzer) to report which constraints were not met. To obtain a better
result,
you may try the following:
* Use the Re-entrant routing feature to run more router iterations on
```

We can see here, the placer had tried already some 21,000 times. These are the first pass and second pass more and so on. Huge iteration, even for a small design, just the sequential seconds, and you can see further, and once this is only the placing that has happened and once the placing is done, so, file is also created here called dot ncd which will be requiring for back annotation, etcetera, later on. It says there are no errors and it was also doing place and route and completely routed it says here.

(Refer Slide Time: 05:58)

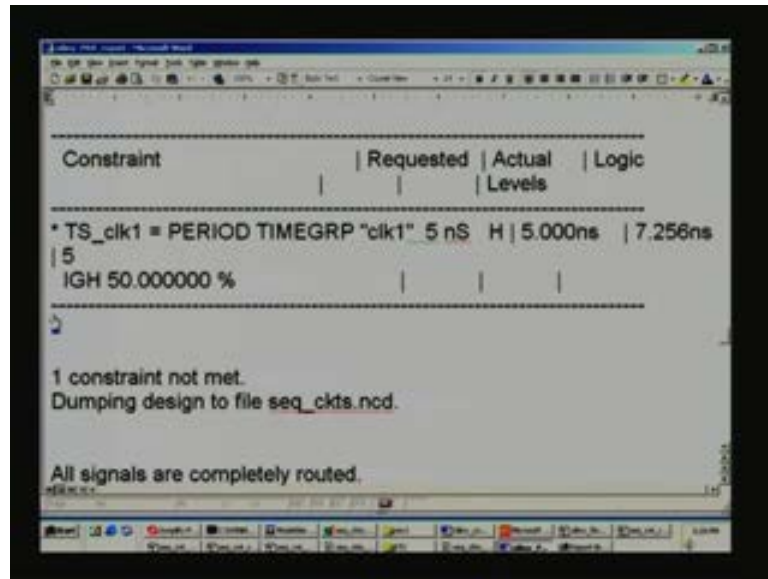


(Refer Slide Time: 06:16)



You can even specify higher level of effort level from one of the menus in the place and route and higher router effort level and placer effort. So, that also can be change as per your requirement. So, more you do the iteration, more, I mean, better results will get in times of frequency of operation. So, that is the advantage in over raiding the default.

(Refer Slide Time: 06:35)



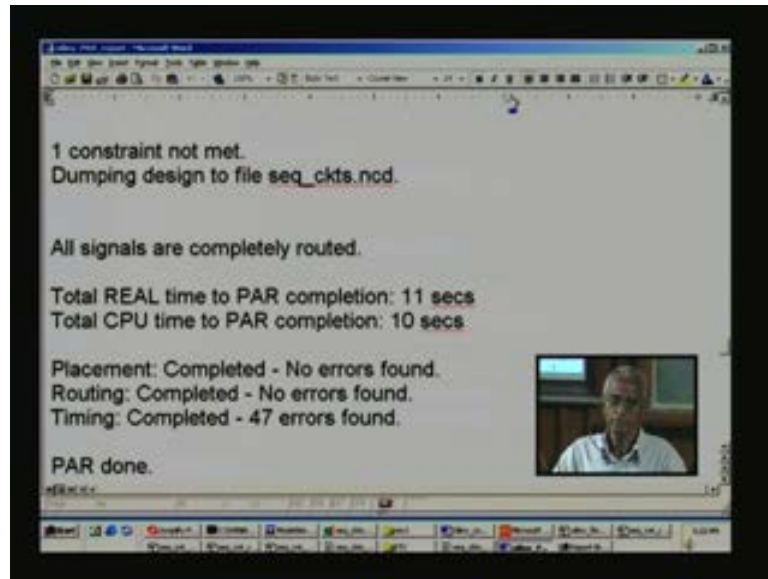
Constraint	Requested	Actual	Logic Levels
*TS_clk1 = PERIOD TIMEGRP "clk1" 5 nS H	5.000ns	7.256ns	5
IGH 50.000000 %			

1 constraint not met.
Dumping design to file seq_ckts.ncd.

All signals are completely routed.

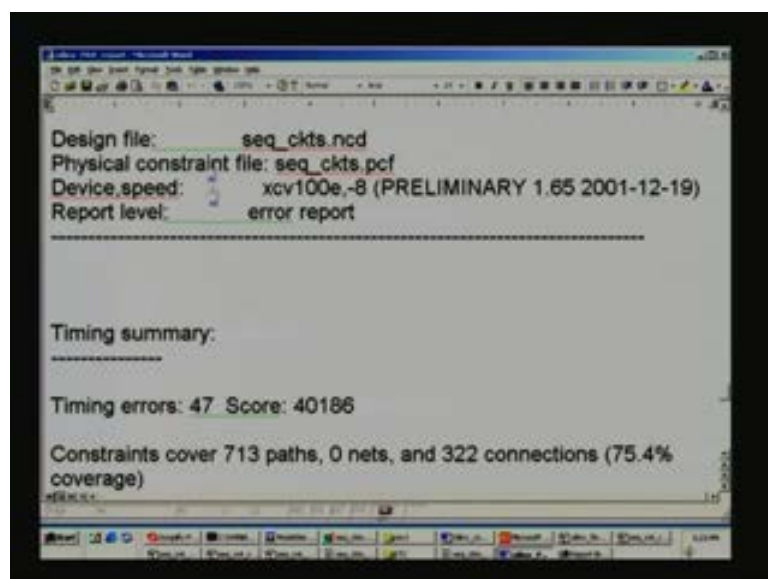
Here, it gives the timing report. We have remember that we had already seeing similar timing report in synthesis, during synthesis, and it was also pointed out that what was obtained there is only refer estimate, only Xilinx place and route is more realistic. It is close at to the real picture. We said is closer because unless you map it, put your FPGA right on the pcb and run at your decide frequency and observe whether they are really functioning as per your functionality that you have program, and that is the final deciding a point for you. Even here, I mean in real practice, people have reported even a bettering of the report, the Xilinx place and route gives. For example, it may be 10 to 25 percent a better performance also we can get based on real experience.

(Refer Slide Time: 07:51)

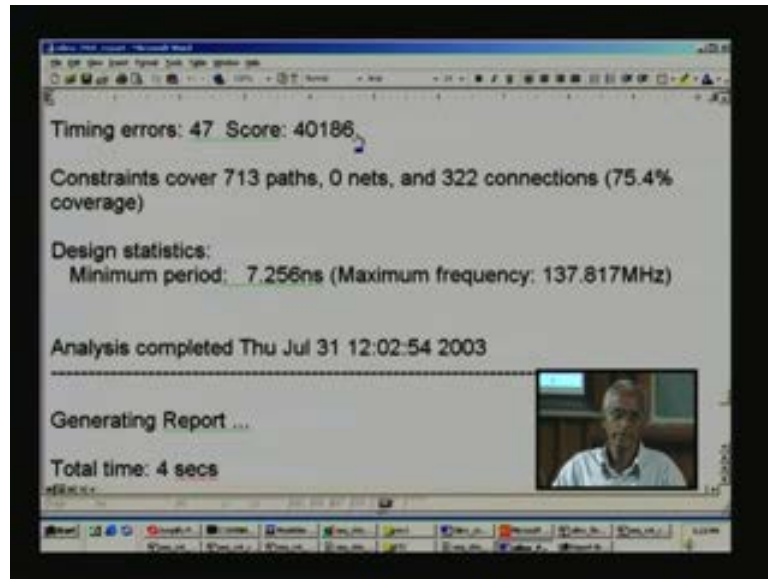


So, what they give is a guaranteed speed of operation here. Normally, as I mentioned, if they design is a quite huge, what synthesis will report will be much higher frequency operation then they place and route. So, and once the place routing are completed without errors, it will report here, and some timing error if saying because we had specified I think 100 megahertz and it could not meet 100 megahertz. So, here also timing error will naturally come, but this is not a handicapped. You can, once again the tool will report how much frequency operation you could get and so on. So, based on that we can see.

(Refer Slide Time: 08:51)

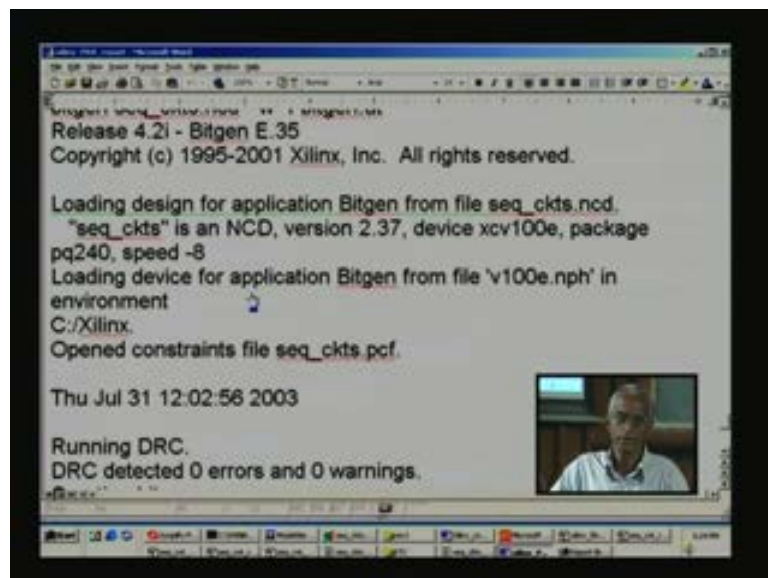


(Refer Slide Time: 08:57)



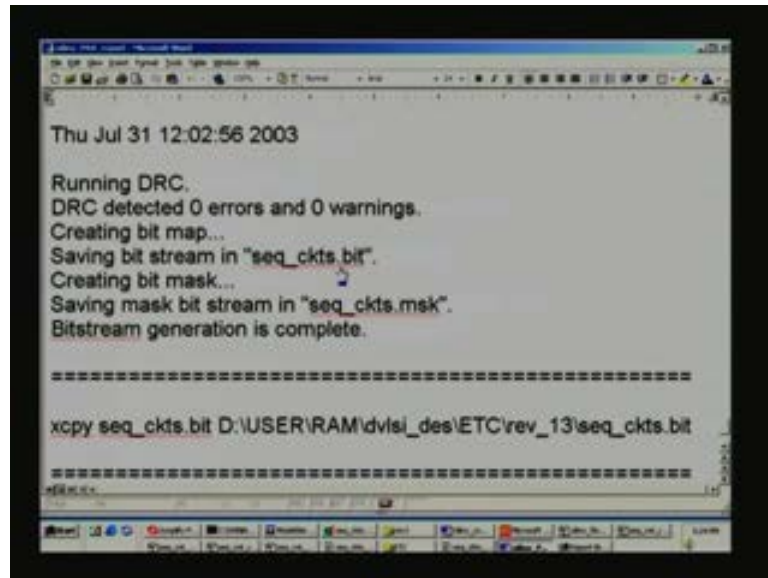
These are all the intermediate files, right now we need not be consider about it. Here comes the frequency evaporation. We see it surprisingly hire here. As I mentioned for smaller design, my observation is synthesis will be very conservative, **for**, but real designs are all big, so, it will be the reverse. So, normally synthesis will report, say 100 megahertz, but place and route can give only some 80 megahertz; so, that is the usual scenario. Whereas, here, it appears to be exactly reverse and I think we started with 1 16 megahertz and synthesis.

(Refer Slide Time: 09:49)



And here, for final use in FPGA, you need a separate file called bit file, and it is also known as a bit stream. So, that comes as an extinction of bit, dot bit will be the extinction use, and it is a Bitgen, it is a loading the design for this generation.

(Refer Slide Time: 10:16)



```
Thu Jul 31 12:02:56 2003

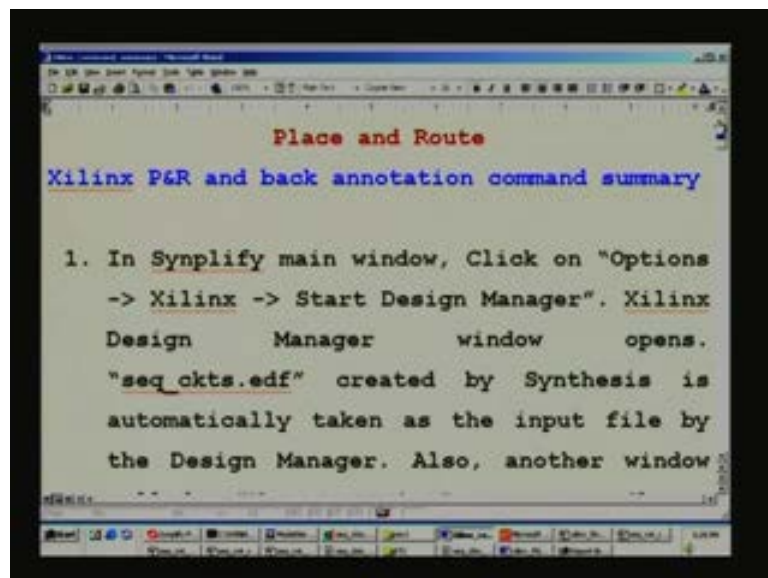
Running DRC.
DRC detected 0 errors and 0 warnings.
Creating bit map...
Saving bit stream in "seq_ckts.bit".
Creating bit mask...
Saving mask bit stream in "seq_ckts.msk".
Bitstream generation is complete.

=====

xcopy seq_ckts.bit D:\USER\RAM\dvlsi_des\ETC\rev_13\seq_ckts.bit

=====
```

(Refer Slide Time: 10:57)



```
Place and Route

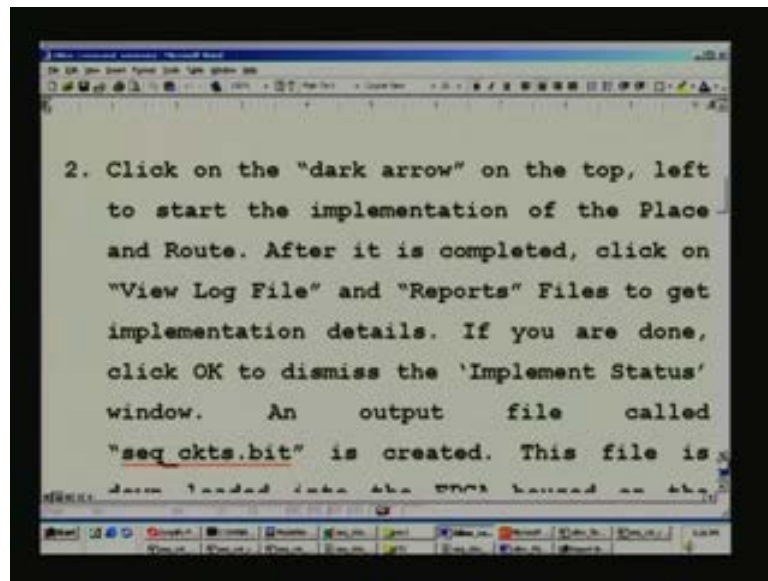
Xilinx P&R and back annotation command summary

1. In Synplify main window, Click on "Options
-> Xilinx -> Start Design Manager". Xilinx
Design Manager window opens.
"seq_ckts.edf" created by Synthesis is
automatically taken as the input file by
the Design Manager. Also, another window
```

It also run for this called a DRC this is a Design Rule Check. So, if there are no errors are warnings, it reports solve that, and here it is creating this bitmap as I mention, and this is the output file, because our design happens with sequential circuits, it nearly takes and appends this dot bit, and rights all this bit file here, creating bit. So, that is all to it for

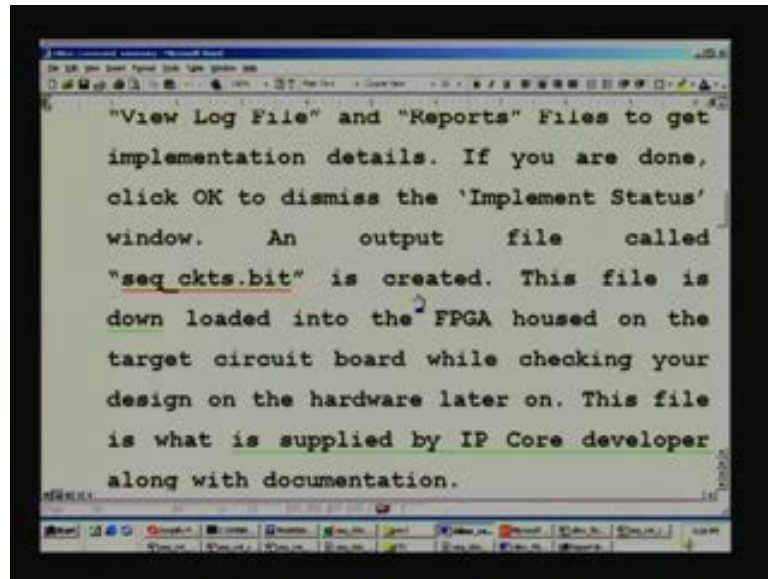
the log file. We will just go through this comments summary which we have already used. To your, just the second step, first step was to invoke the simplify and then in gone to the design manager - that is a place and route - and we did not see the second step at that time, and we are just mentioning how to start the implementation place and route, and perhaps, we did not read this, so, I will just read this out.

(Refer Slide Time: 11:09)



Click on the dark arrow on the top, left to start the implementation of the place and route. After it is completed, click on view log file and reports files to get implementation details. We have seen the view log files so far, and reports file, I should browser. You will literally get exactly the same thing expect for some more detail, such as what pins, what signal are mapped on to what pins, and I will also be reported that, and some timing report all that we will be getting. So, we will see those reports as well, files to get implementation details.

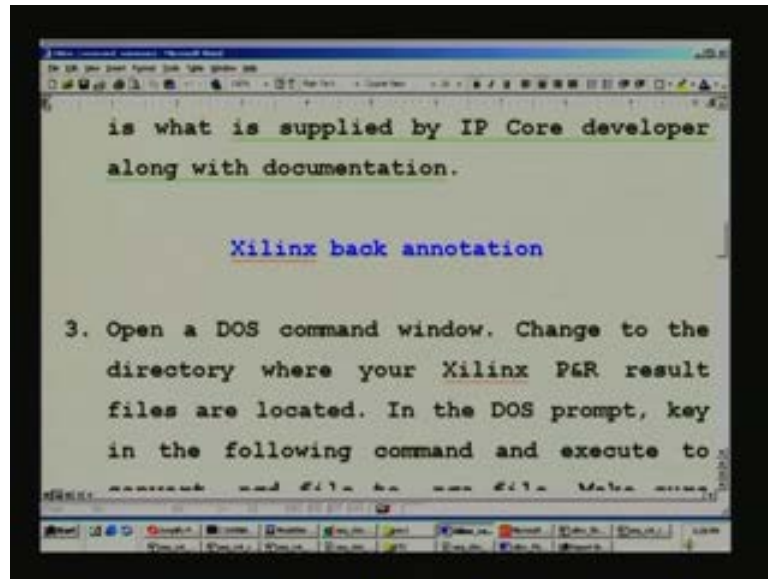
(Refer Slide Time: 12:06)



If you are done, click to dismiss the implement status window. An output file called sequential bits; this is what we have seen. So, circuits dot bit is created, and this file is downloaded into the FPGA house on the target circuit board, while checking your design on the hardware, later on. So, this will happen only at the later stages of your design cycle. So, when the pcb is ready, only then you put on you are FPGA populate bare pcb with different components including FPGA, and then, it will be ready for testing. At that point of time, you can download this bit file into the FPGA, and normally, from the development system you can you have provision to connect straight to the FPGA.

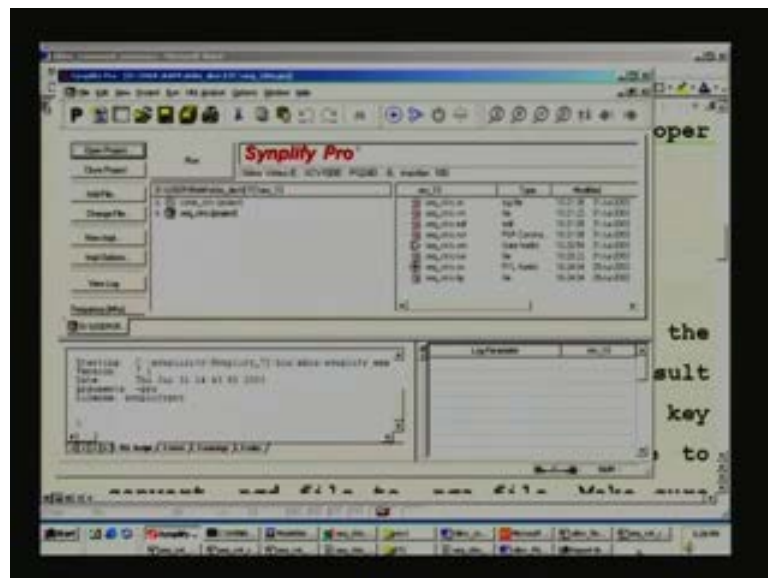
You need a special interface for that some serial interface or even a parallel interface. This file is downloaded into the FPGA housed on the target circuit board while checking your design on the hardware later on. This file is what is supplied by IP core developer along with documentation. See, suppose you happen to be working for an IP core company, so, you as a designer will had to prepare a document full document as to how the user can use it, and then also you should give this dot bit file. Normally, this IP core people will not give the source file, because they want to keep it to themselves and unless the contract demands the otherwise.

(Refer Slide Time: 13:36)

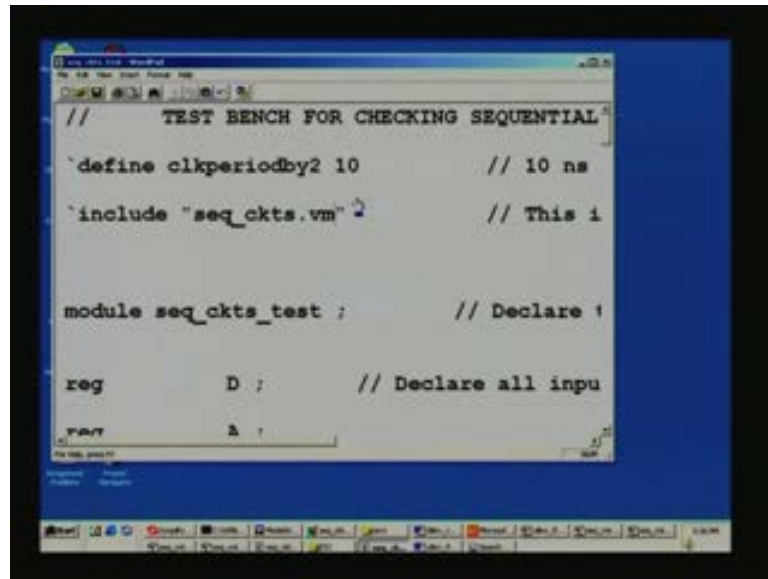


And next step we will go on to Xilinx back annotation, and before this some of you express the desire to see an optimized verilog files, that is dot vm file created during synthesis or simply simplify. So, will see that, we will take up that first, and then come back to this.

(Refer Slide Time: 13:58)



(Refer Slide Time: 14:43)

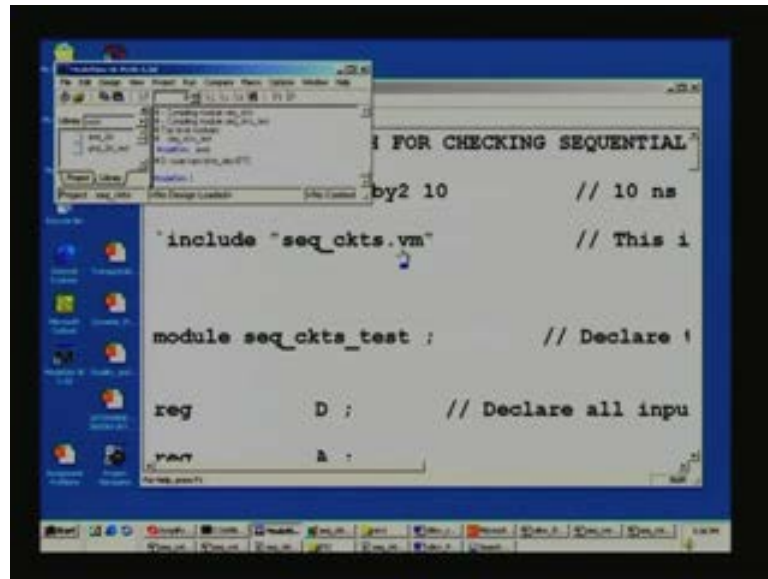
A screenshot of a text editor window displaying Verilog code for a test bench. The code is as follows:

```
// TEST BENCH FOR CHECKING SEQUENTIAL
`define clkperiodby2 10 // 10 ns
`include "seq_ckts.v" // This i
module seq_ckts_test ; // Declare
reg D ; // Declare all input
PART A ;
```

The window has a blue title bar and a standard Windows taskbar at the bottom.

This is the main simplify window and use this sequential circuits test bench, and in order to, I mean that is of little no difference between the original. We need to invoke the model same in order to see whether this optimized file, dot vm file, of course, here, makes, this is only a test bench. So, what is different is the optimize file is only the designed file, but the test bench includes that particular file. So, you had to have this. This is not really the correct way to do. Actually, should, what we should do is we should go to the designed file, sequential circuits, and then add ((No voice from 15:35 to 15:38)) for time being we will retain that, let us see. So, will had to, what I am saying is in the test bench, you go and include, earlier it was dot v file, designed file. Now, what, optimized file instead of the designed file must be used. so this is what, it is here.

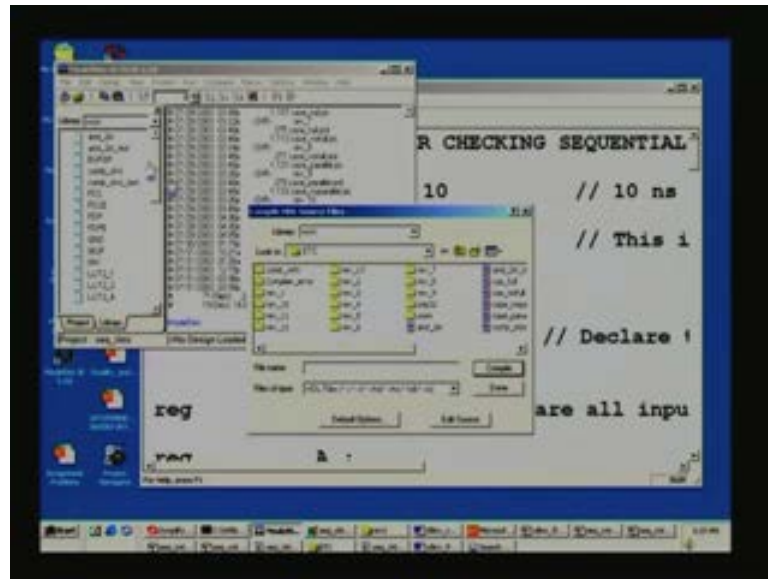
(Refer Slide Time: 16:33)



```
FOR CHECKING SEQUENTIAL
by2 10 // 10 ns
include "seq_ckts.v" // This i
module seq_ckts_test ; // Declare
reg D ; // Declare all input
A :
```

So, the, in the test bench, we have to include this dot vm, as I mention, this is a change from design file which was just dot v to these a dot vm which was created during synthesis. This is the optimized file, and we can even have a look at this file if you wish later on. Right now, let us open the modalism, its already open I think, it is opened. We are in the directory, let us use pwd to reveal what directory, d used run, d vlsi, etcetera, etc, it is not etcetera. So, under this directory, I mean this dot vm has been created in some other directory area, but I have already copied that into this directory, and in fact, by directory we can find out whether it is, so, no, it's already here, it's here, it's a huge file some 42000 and odd. So, now, what we need to do is if you compile this once again, so, in in this test bench we have replaced it with dot vm file, so, it will take this as design file now and then compile.

(Refer Slide Time: 17:30)

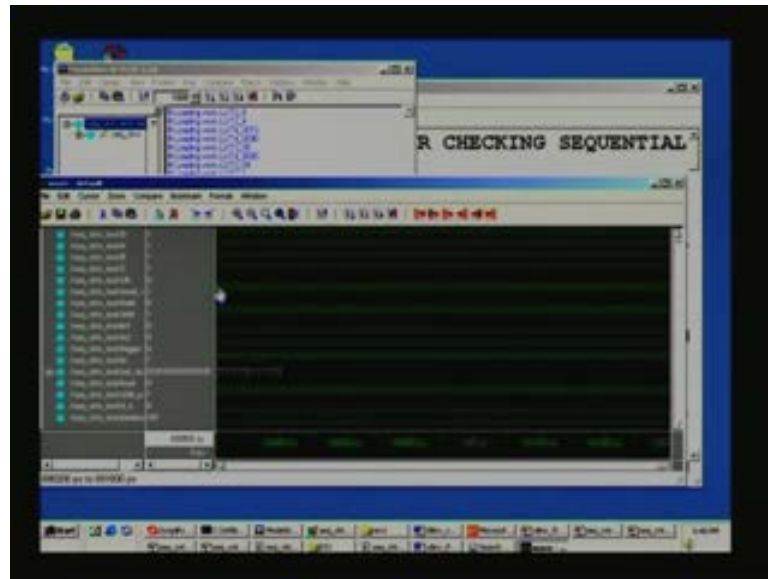


So, let us see what it says, not to compile, design, compile, window opens, its in etc window, and here, we need all files at the bottom, then only we can view this, and in fact, we do not need that also, because it can be verilog itself. So, in which clear, after all what we need is only sequences require test and the test bench. So, we will just double click on this, and what happens here, in this window, when you double click here, you see, give a big list, and I read it for you. So, you are already familiar with the synthesis having created lut 4 lut 2 and fdc fdp, then once again lut d flip of for a fdp, etcetera, and so many lut's, then macs, buffer, input buffer, output buffer, it has taken all this and compiled.

So, that is because we have mention in the test bench that vm. So, earlier, if it were dot v do not have reported all this. So, that means to say it has right now taken your dot vm which was optimize file by synthesis are the source file and next step is to load this design file.

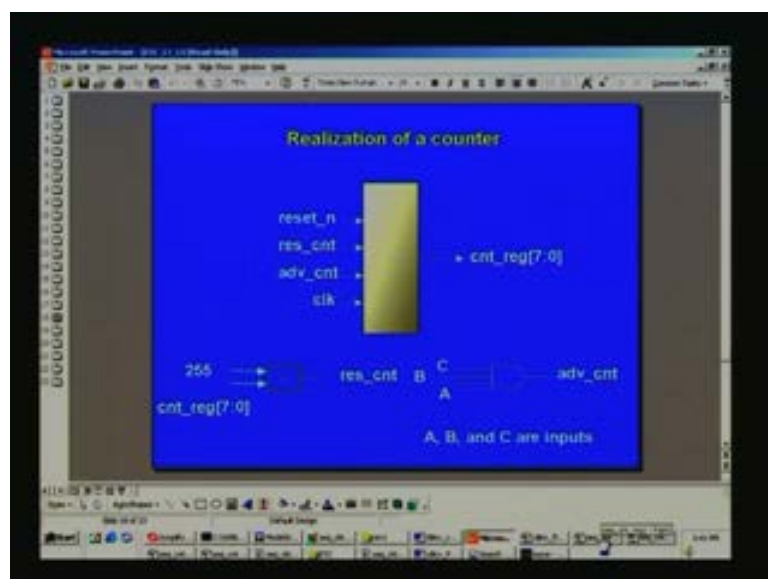
We do not need the compile here, so we will remove that and this is the load file and once again same buffer all that primitives, its reporting here. I think towards the end you have sequential and has got test. So, let us load this file.

(Refer Slide Time:20:12)



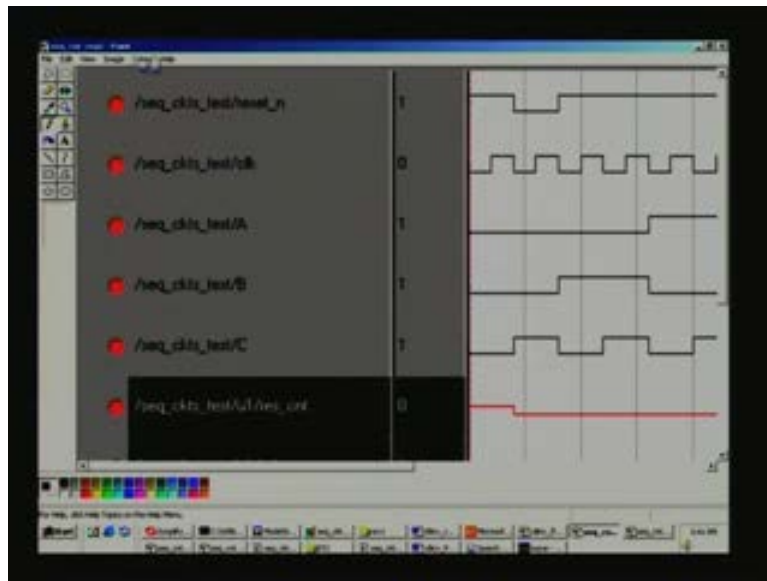
So, while loading also it has displayed all that primitive cells here and there are no errors so, we can conveniently see the way from now. So, in order to see that, click view, then signals, then again view, then wave, then final last option signals in design. So, it has opened the wave file; it do not require the signals anymore. If you click on this, it will run. So, it has run and swaps and as usual source file is opened at a swap at terminated, so, you do not require this. So, this is the wave from that we have, and in order to analyze, will take just one example, and in a similar fashion, you can analyze other functionalities.

(Refer Slide Time: 20:25)

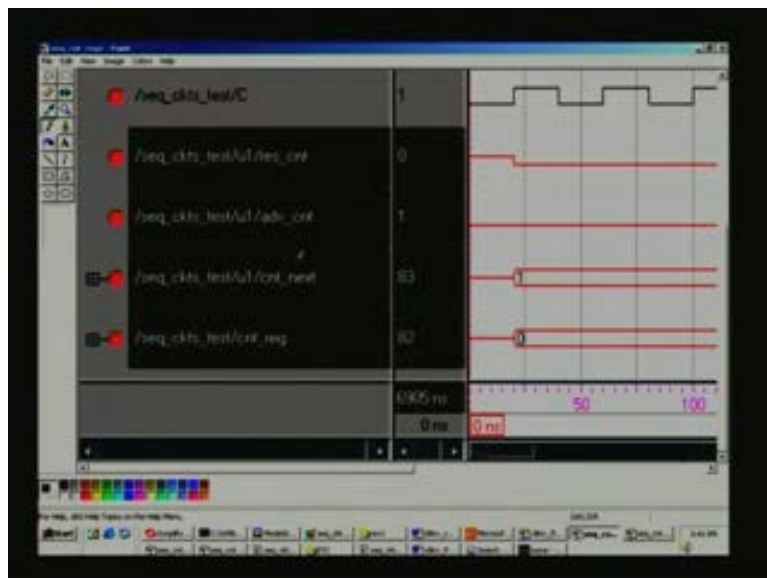


So, you remember we had this is sequence circuit, we had let us say a count, we had a counter, this is simple counter which advances right from 0 through 255 and revolves around that, and it can be reset for certain conditions when counter equal to 255 and can be advance if A B C are each one. So, that is a simple counter. So, will just have a, I have already preserve the wave from and has we had done before in paint. So, we will just have a look at that.

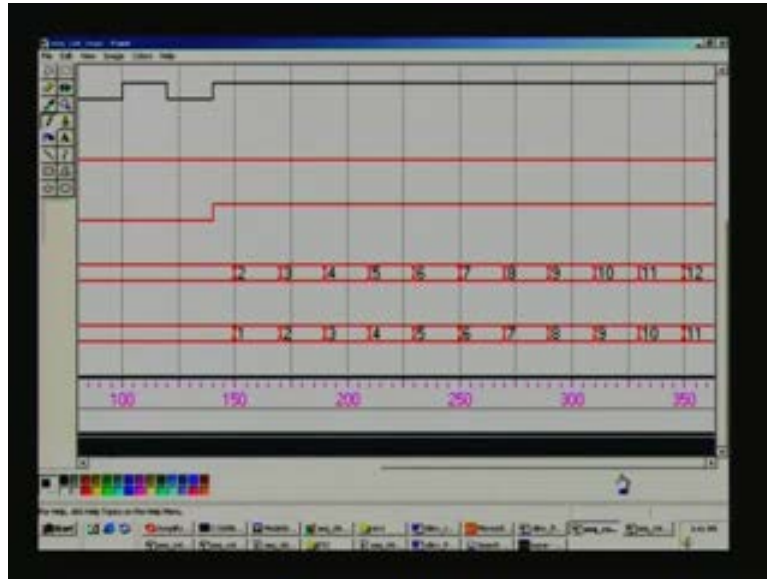
(Refer Slide Time: 21:02)



(Refer Slide Time: 21:21)

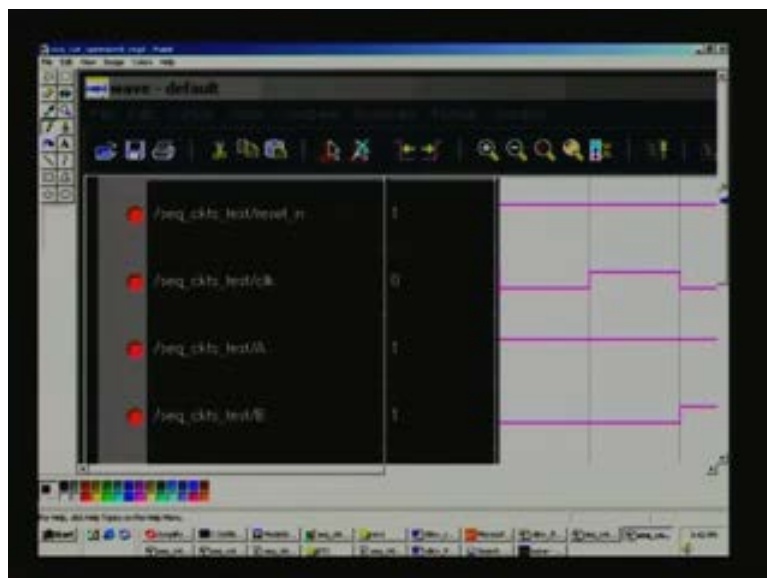


(Refer Slide Time: 21:37)

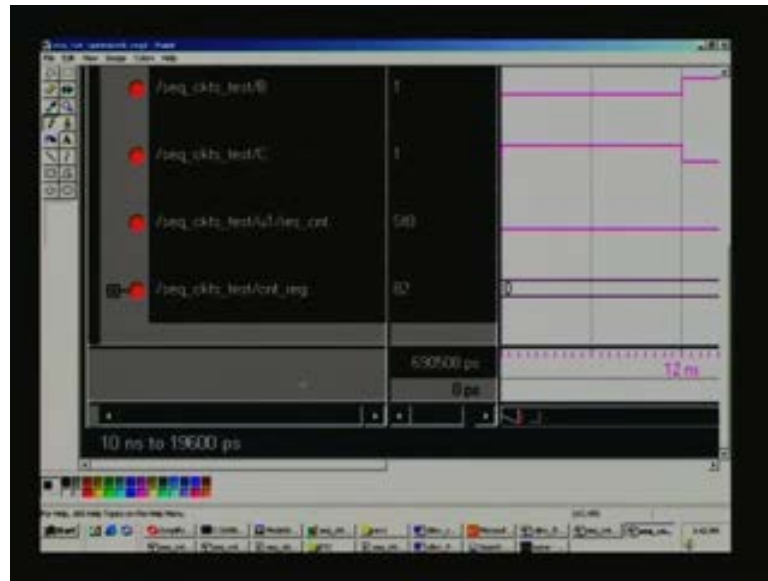


So, this is the wave from which you already seen and analyze earlier. So, this is nothing but starting of the counter, for example, this is reset clock A B C are the inputs, then a reset counter here, then advance counter, then counter next in which only a next value that indicates in advance, and this is the actual counter register. Notice that it is 0 here this is forty i have already seen and if advance sink form 0 1 2 3 and so on and the this is for advance being A B C being 1 here and this corresponds that, this we have already seen. Now, what we need to do is we have already run the simulation capture the wave form for corresponding to the optimized file.

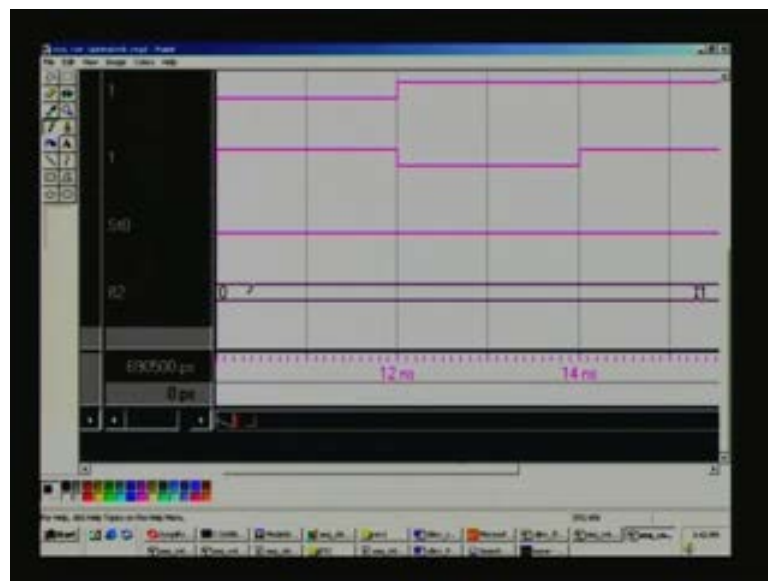
(Refer Slide Time: 22:06)



(Refer Slide Time: 22:18)



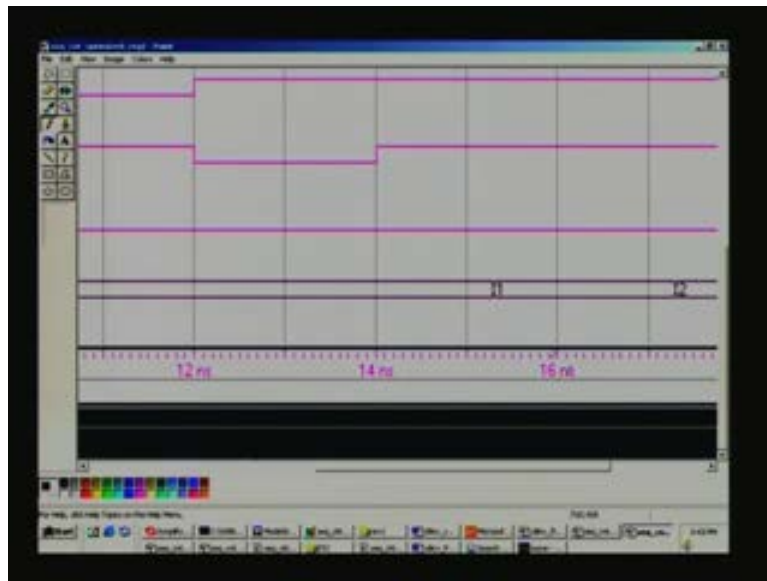
(Refer Slide Time: 22:34)



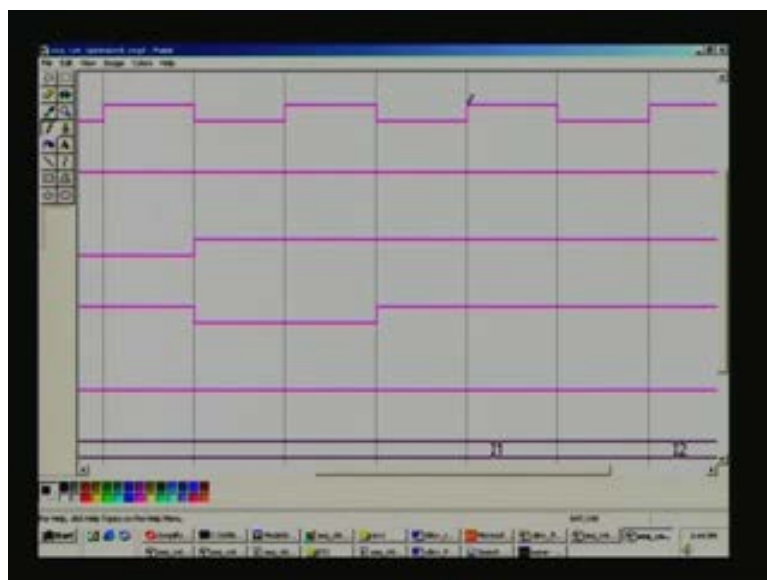
So, let us see what it is and this is the optimized one. So, I have name the test sequential count same rage one I have preserved and it is optimized, and this a timing, time base may be different because I had done it a fresh, so, that may be different thing. Form that, this was here, graduated like this, and here, it is closer, it is not closer, I mean wave from is closed that means zoomed in, and you can see again this is the counter final count value and A B C are the inputs here. You can just see all other wave form represent the same reset 10 clock A B C here and you would notice one thing here. So, in, we have a reset counter has before, but there is no advance counter.

So, I think in optimization it might have removed that signal, because the same signal might be appearing elsewhere with some other name because we had conglomeration of different circuits, so, that may be the reason. So, I could not find that signal advance count. So, that is what I was saying, while optimizing, duplicate will not be created. So, it will lift that one and use, I mean it will be map just one time and then reuse at different points of time, and here, you can see again the counting advance sink here.

(Refer Slide Time: 23:56)

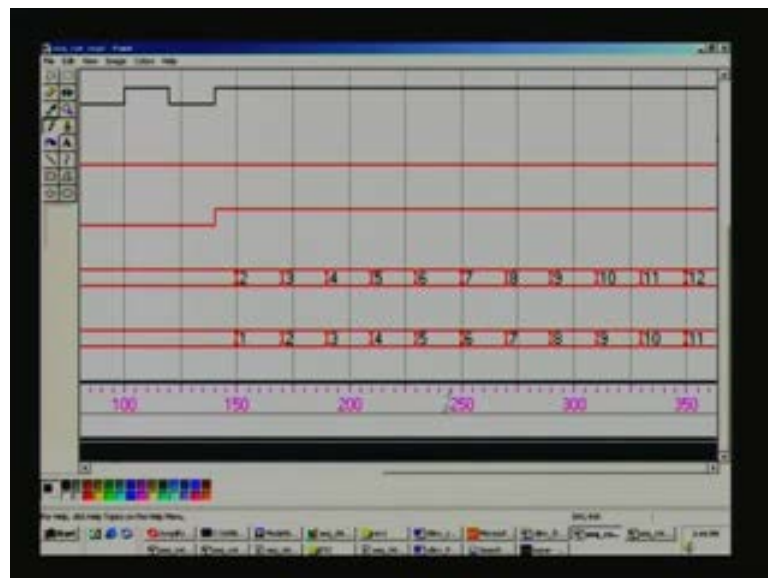


(Refer Slide Time: 24:02)

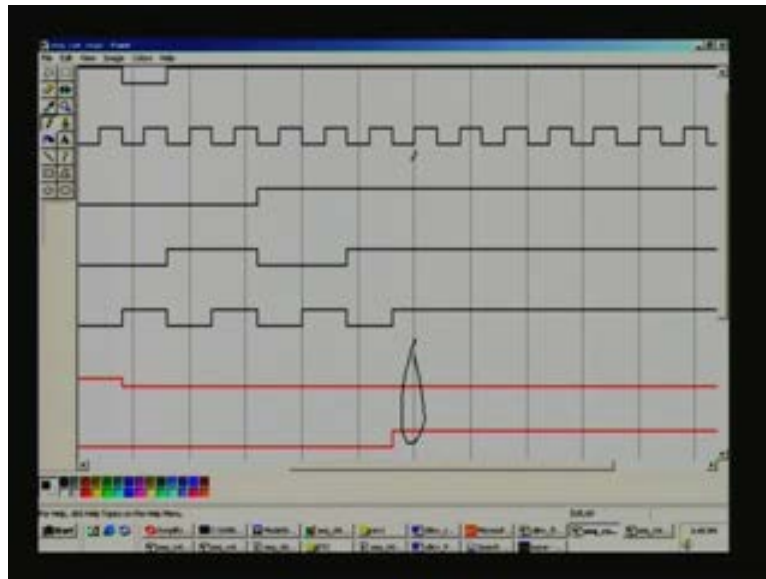


Now, let us see what the, is there a clock. I think forget in the clock, or no, its there. So, the clock is this wave from here and very last one is the counter here, and unfortunately we cannot see both, can you? If the difficulty you can. So, now notice this here, see, this positive edge of the clock, it is not happening, this counting; it is only after shift it has happened. That means to say, here, after all this synthesis is aware of the delays, gate delays, because it has snapped on to the actual primitives that Xilinx place and route is ultimately going to use. So, whatever place and routes details supplied by Xilinx have been incorporated. So, it may not be in depth, may be interconnection delays are not fully accounted for and that is the reason why we need to go for Xilinx place and route.

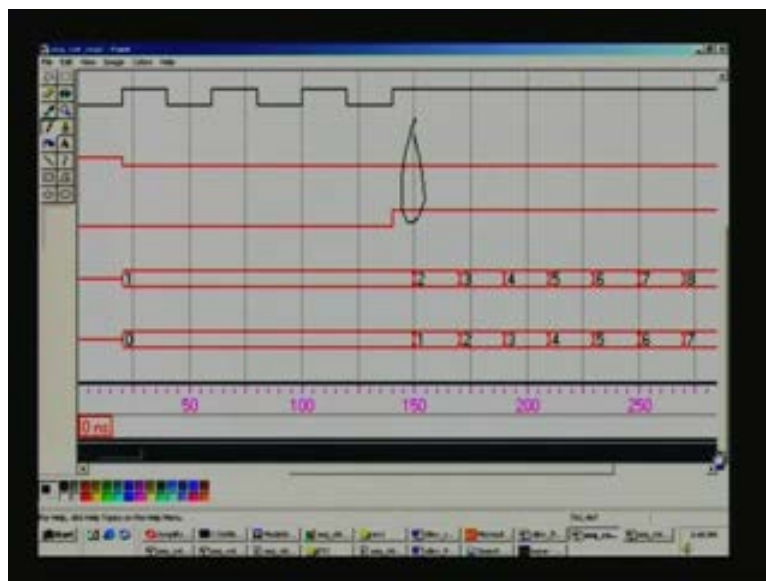
(Refer Slide Time: 24:50)



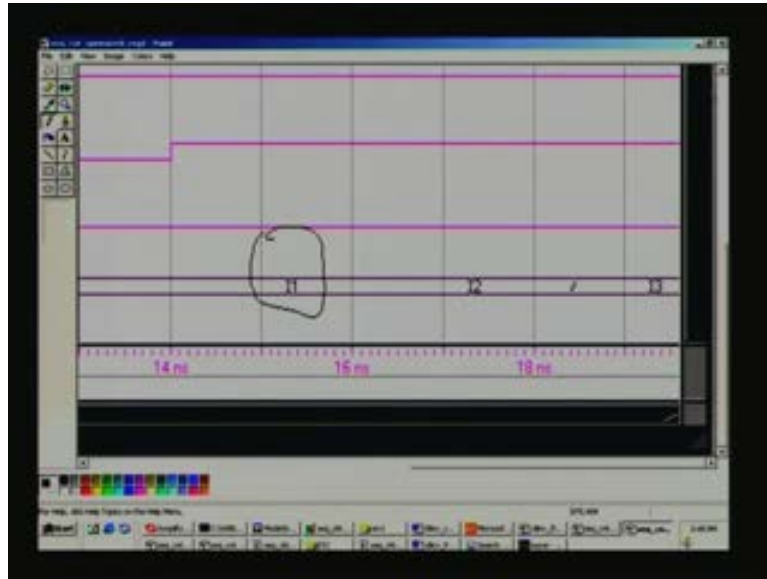
(Refer Slide Time: 25:04)



(Refer Slide Time: 25:18)

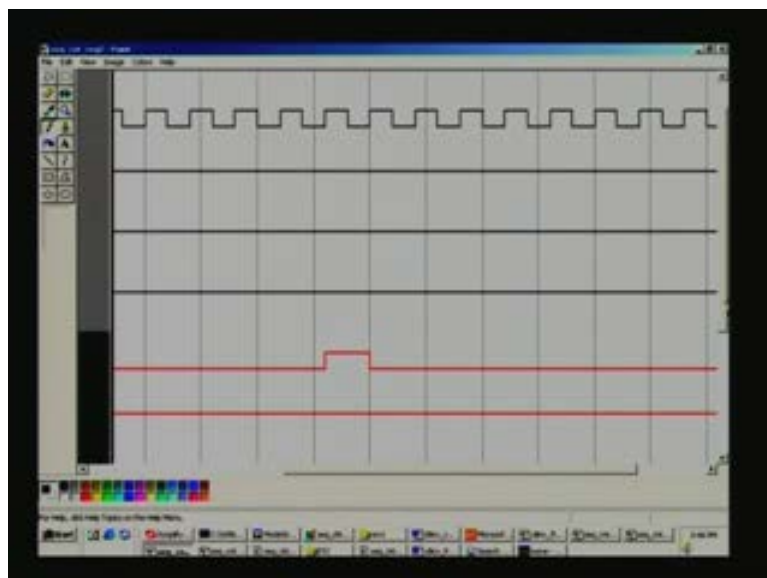


(Refer Slide Time: 25:33)

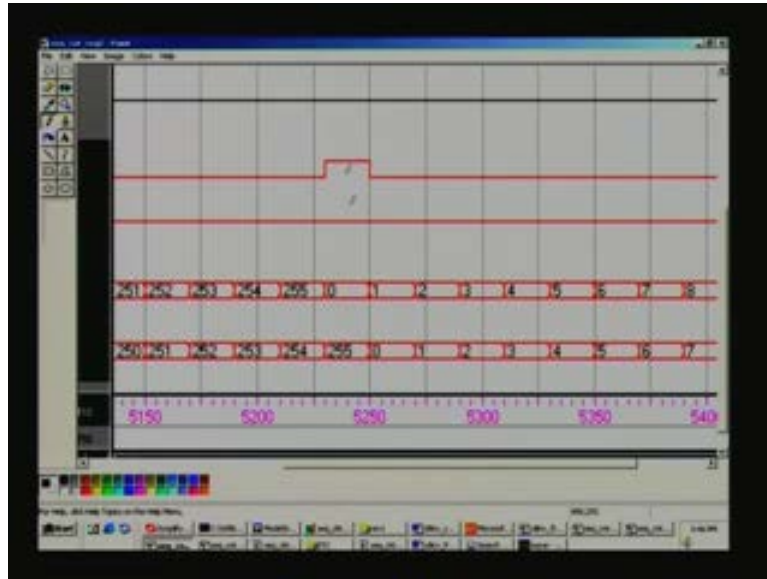


So, you have some set of delay element of also coming to picture that why you are able to see here. Let us see, compare this with the previous one. So, for example, so, let us get to the clock. The clock is here, not to, it may be difficult for you to remember. Let us take this here; some were here, just draw line here. This is the positive edge of the clock. You can see, it is exactly align with the same clock, say any count change is exactly align with the positive edge of the clock here, whereas, they dot vm has delay here that is what we have seen here, is it clear? And the counting as usual progresses in this session. Similarly, towards the end of the count, we can have a look and for both this.

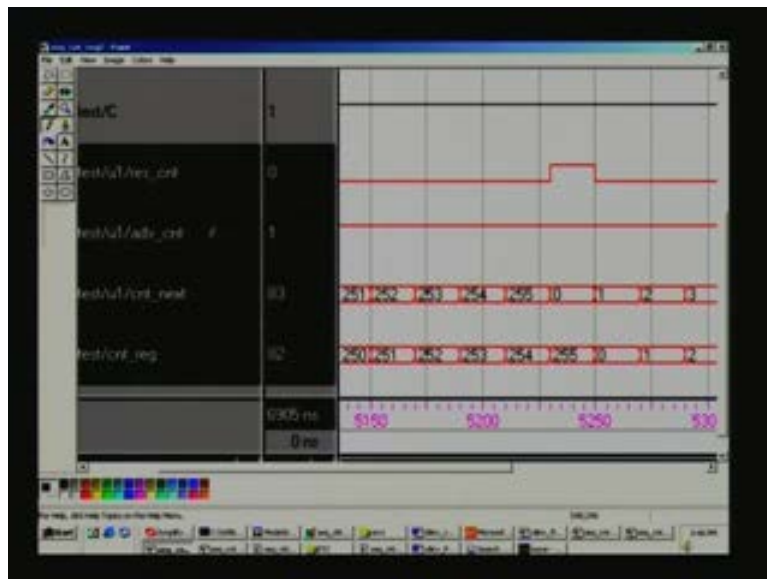
(Refer Slide Time: 25:55)



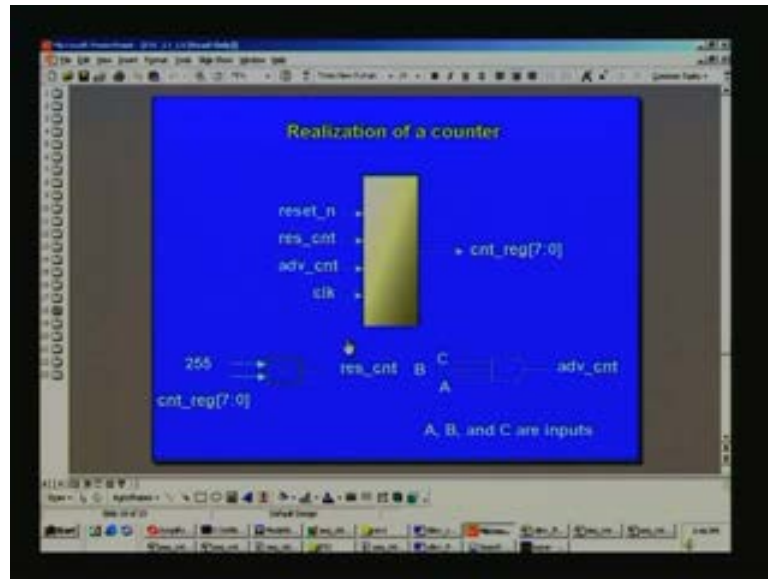
(Refer Slide Time: 26:02)



(Refer Slide Time: 26:08)

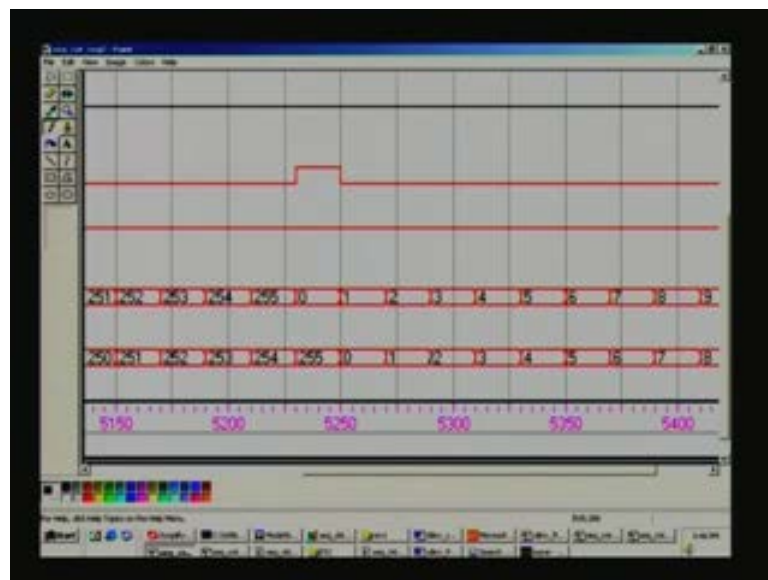


(Refer Slide Time: 26:39)

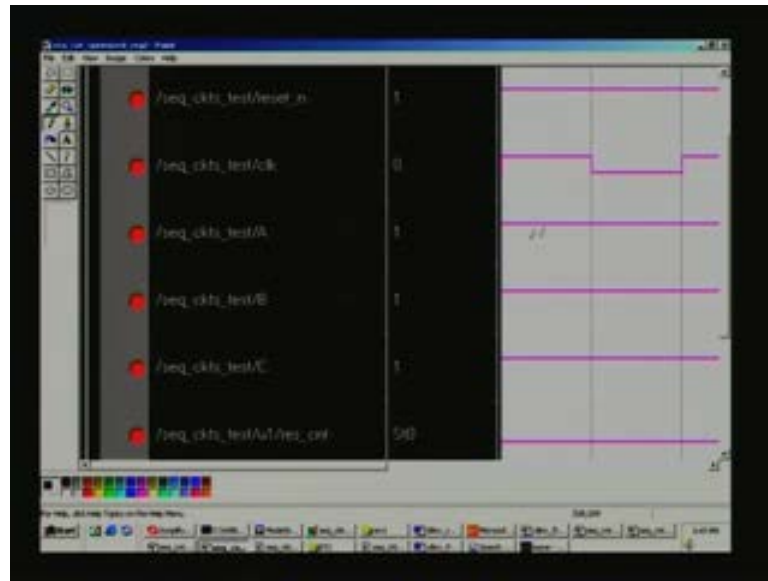


See this is the original one, and this is towards the end. Once again you can see at this edge, and when the count is 255, so, this goes high, so, that is a reset count is acting here. Here, you see this advance count, but in optimize thing you are unable to see because it has been optimize, it has been removed. The actual signal itself is not removed, but it may bit preserved under some other name not as this name. So, you see here, right at 255, when the counter is 255, so, it gives one reset output, that is what we have for resetting, is not it? So, as per this, you should rest when the 2 matches.

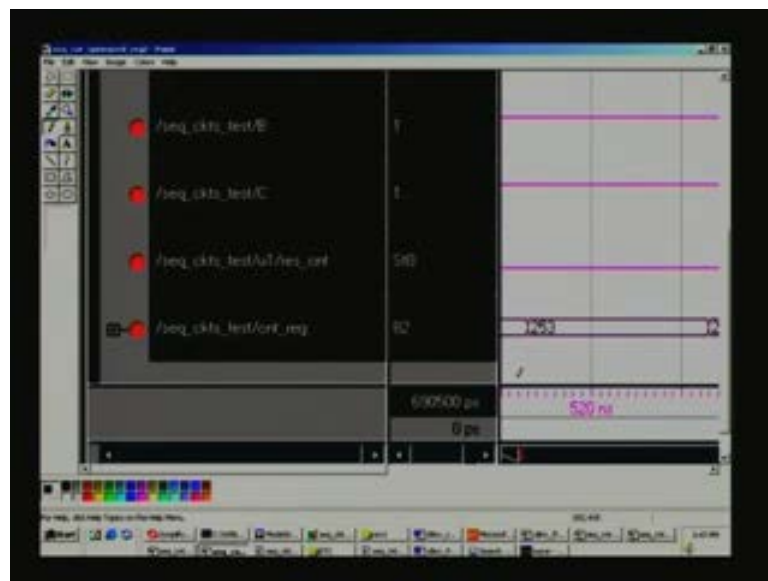
(Refer Slide Time: 26:49)



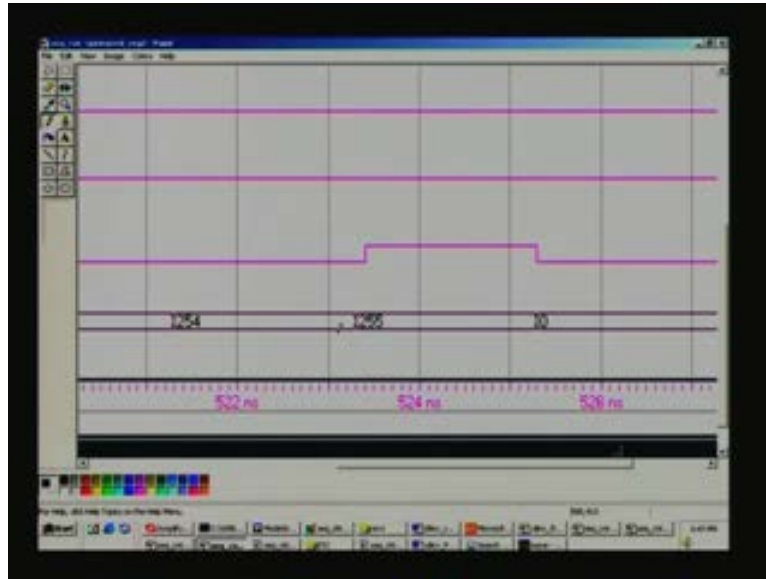
(Refer Slide Time: 26:59)



(Refer Slide Time: 27:07)



(Refer Slide Time: 26:27)



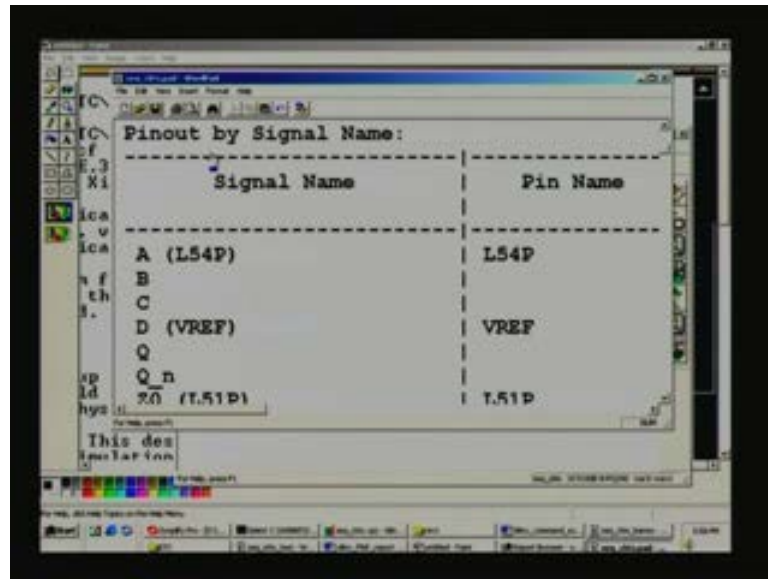
So, coming to this, it's advancing, this is already seen earlier, and again it repeats around the counter and optimizes on corresponding to that is this, the clock is here as before and all A B C are all one here, and so, the reset count is advancing, that is the condition for advancing. From that, we make out, all though at this advance count is removed in this, and last one is the count register. I am early shown the last few counts, and once again you can see this reset counter is active, when 255 is on, and notice the delay here also. When 255 is occurred, only after a slight delay here, this is happening because this is being compact. Similarly, here, when you go on to the clock, rising at this here, and to 255 itself is maintaining the same phase shift here. So, this tells some in this fashion, you can analyze any other circuitry that we have seen before or create an own design and analyze.

We are looking in to the reports browser earlier appear to going for some diversion into the synthesis dot vm file analysis that is after optimization. So, we will resume from the place and route report browser. So, you have what is called utilities and you have a report browser, if you click on this, so, you will have list of different report files. For example, you want know; let us say we want to have place and route report.

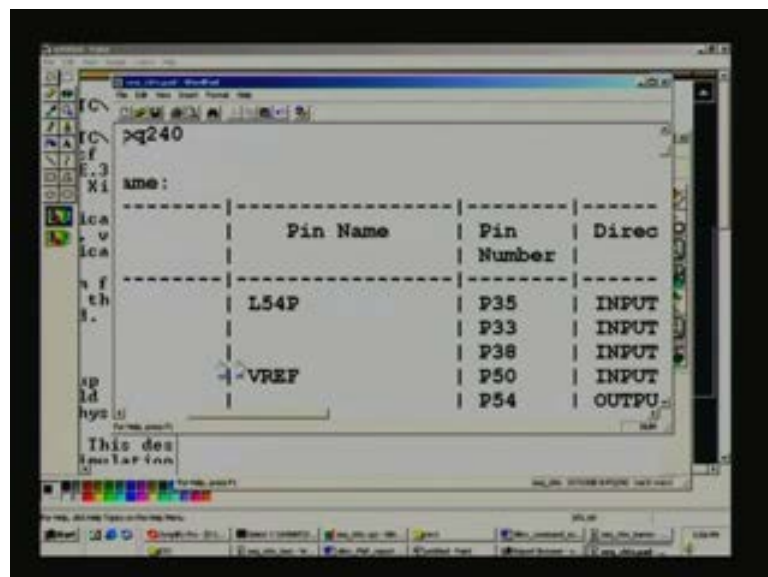
So, this is the report. So, and then, if you want just mapping alone, you can click on this map report. Then you have what is called translation report, navigation report, and then pad report, IO pins, etcetera will be reported here. Some then delay report is available;

then post layout timing report, and like this two other, in fact, then one more you have called Bitgen report, and all these things are clubbed into one, we can say that log file which is already inspected. I would rather leave it has an excise for we to go into the depth of all these. Most of them will can way moral like same importation that you have got from the log file.

(Refer Slide Time: 29:46)

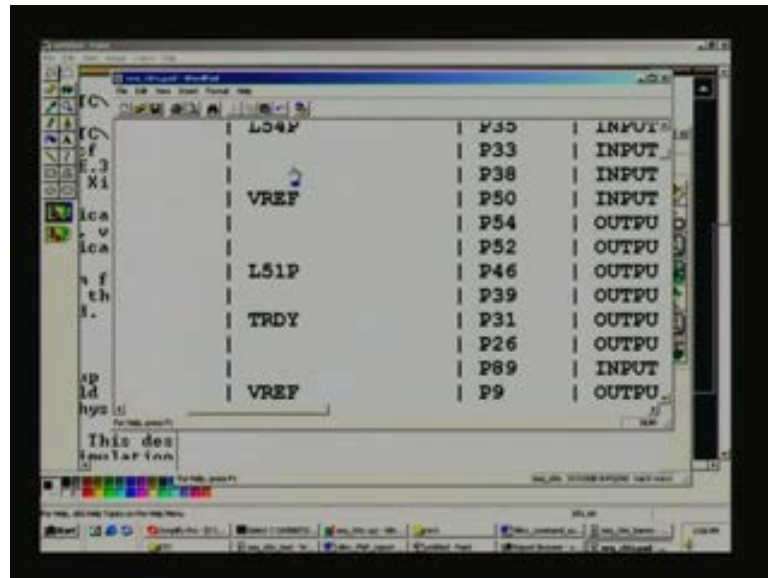


(Refer Slide Time: 30:07)

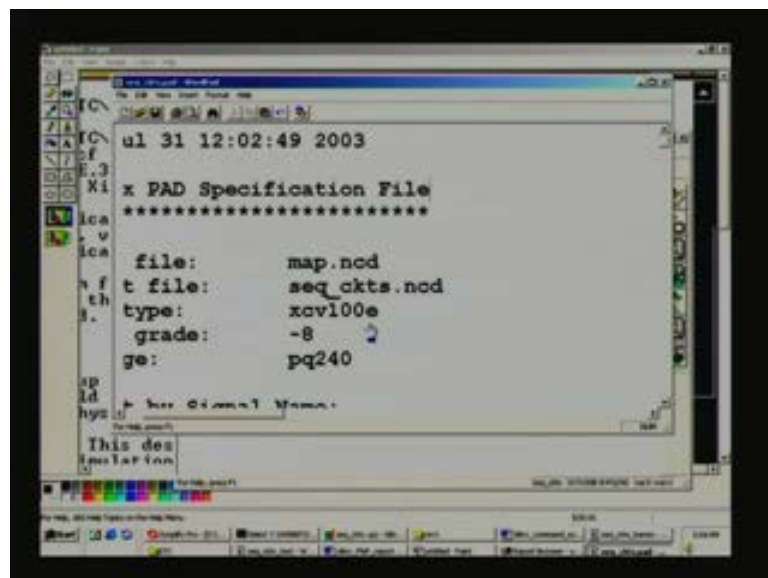


Let us see one of this. See, for example, this a pad report, so, this is the pad report, and IO pins will be reported here. The signal we had used in our design are A B C D, then flip flop, and then state machine, those outputs, a counter and so on.

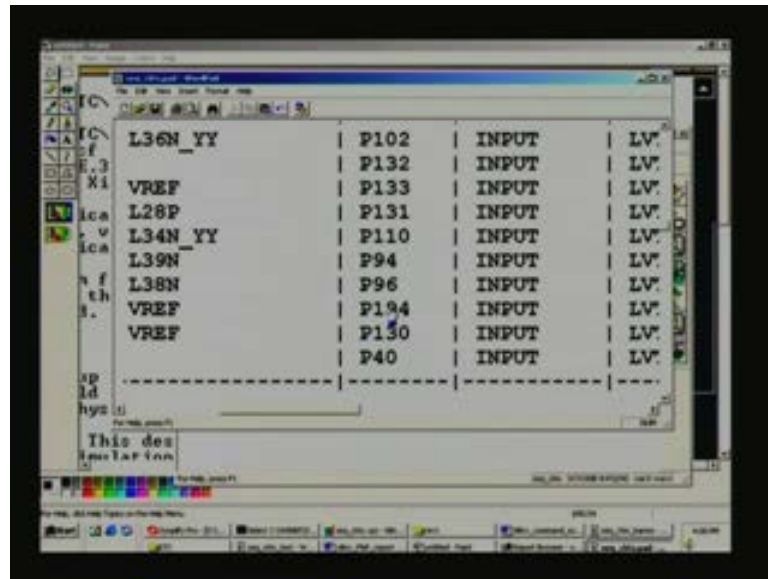
(Refer Slide Time: 30:12)



(Refer Slide Time: 30:42)

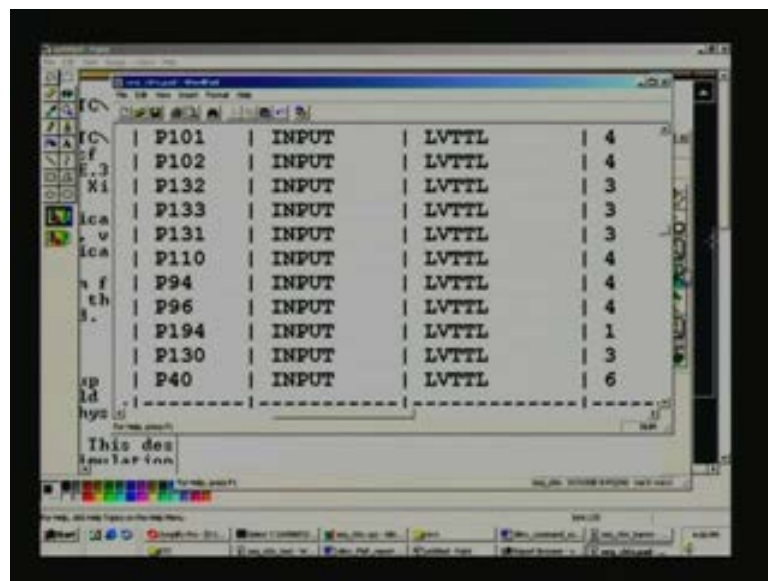


(Refer Slide Time: 031:17)



L36N_YY	P102	INPUT	LV.
	P132	INPUT	LV.
VREF	P133	INPUT	LV.
L28P	P131	INPUT	LV.
L34N_YY	P110	INPUT	LV.
L39N	P94	INPUT	LV.
L38N	P96	INPUT	LV.
VREF	P194	INPUT	LV.
VREF	P130	INPUT	LV.
	P40	INPUT	LV.

(Refer Slide Time: 31:21)



P101	INPUT	LVTTL	4
P102	INPUT	LVTTL	4
P132	INPUT	LVTTL	3
P133	INPUT	LVTTL	3
P131	INPUT	LVTTL	3
P110	INPUT	LVTTL	4
P94	INPUT	LVTTL	4
P96	INPUT	LVTTL	4
P194	INPUT	LVTTL	1
P130	INPUT	LVTTL	3
P40	INPUT	LVTTL	6

So, all these, these are all the pin numbers for that; pin number is here; pin name is also given here. In this, you may have so many other pins along to the particular FPGA and for programming, some reference, and some ready, and so on. There are so many pins there even bcc ground all that will be there **because** its full listing of all the pins for potening to that particular package, but we have used earlier; it is a device you remember was 100 speed 8 and p q 2 40 a package. So, this all are the pins that you have already seen here. So, pin 35, **may be the,** may be some rows, because its only 240 pin probably it is a nearly is some dip type for, I do not remember really, and you can see some 237

pin number coming. So, it is a 240 pin package, so, naturally you should not see anything greater than 240 here, and each of this signals weather it is an input or not weather it is TTL compatible, or let us see.

(Refer Slide Time: 31:40)

Bank #	Drive (mA)	Slew Rate	Pullup / Pulldown	IOB D
	12*	SLOW*	NONE**	NONE
	12*	SLOW*	NONE**	NONE
	12*	SLOW*	NONE**	NONE
	12*	SLOW*	NONE**	NONE
	12	SLOW	NONE**	***

This is not clear to me, and how much load it can take? Drive, so, drive means so much load can be connecter up to this much, I think may be active load could be, and weather, I mean what is the speed, weather it is slow or not, when is there a pull up here, and so on. Then what is the voltage, any other constraint that we have here turns on.

(Refer Slide Time: 32:14)

```

Release 4.2i - Par E.35
Copyright (c) 1995-2001 Xilinx, Inc. All
Thu Jul 31 12:02:49 2003

Xilinx PAD Specification File
*****

Input file:      map.ncd
Output file:     seq_ckts.ncd
Part type:       xcvt100e
Speed grade:    -R
  
```


So, we can see voltage. This is pad report. Similarly, there are other reports you can have look at this. You have a question?

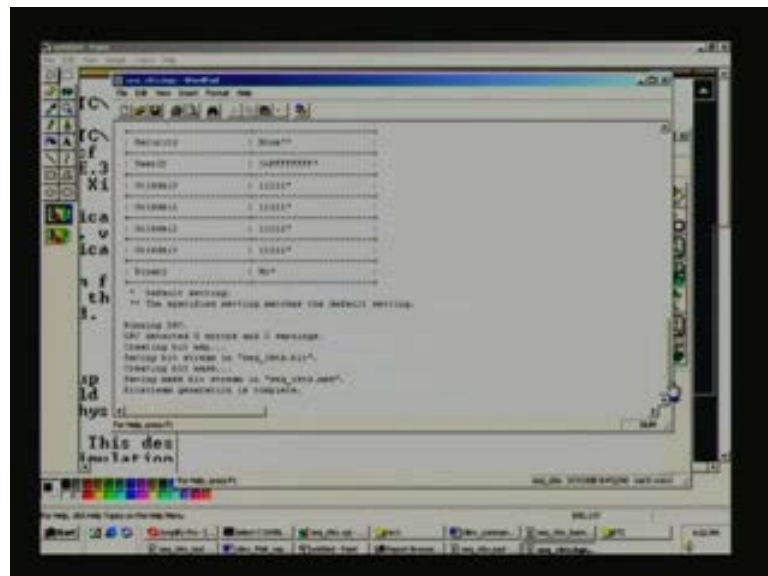
Should the level is TTL compatible, is not it?

Yes

Can you have other compatible level also?

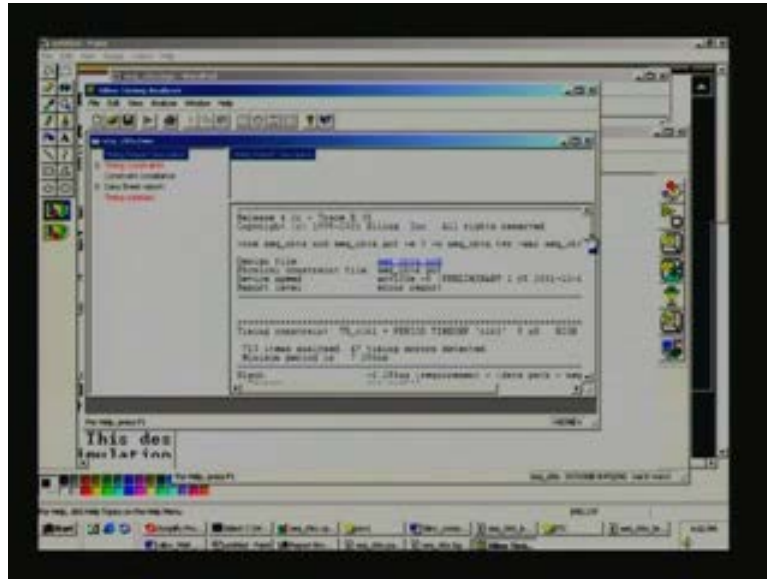
Mouse, **in fact**, this pj as are basically mouse base version, so, they have what are called TTL tolerant voltages; at the chip itself may be working at 1.5volts or 3.3 volts, and so on.

(Refer Slide Time: 33:23)

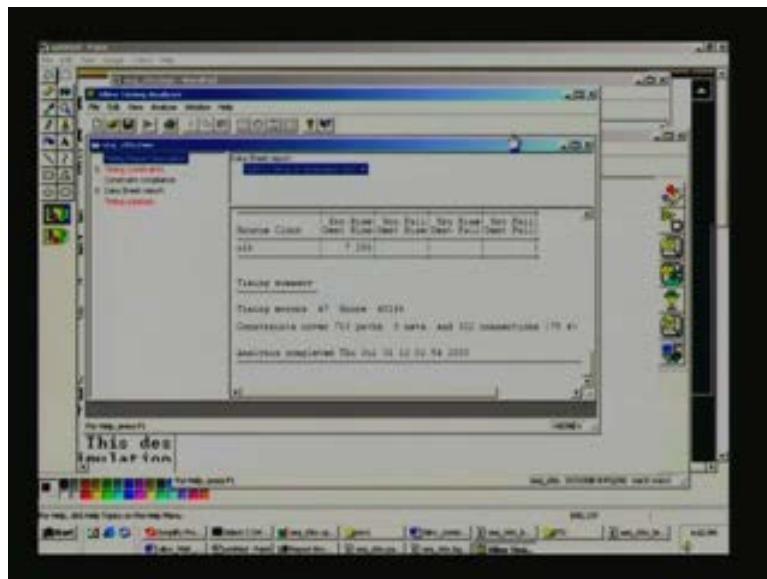


So, no matter what supply voltage is and they will have compatible IO, so that any other TTL or any other device c mos or mos can be connected. So, coming back to this design manager here, so, as I mentioned you can use this utilities then click on report browser, then you get, you can open one file after another and find out what they are. If you are interested say Bitgen here and it may have look weather it gives any new information; nothing much is there. So, except that, it reports saving bids stream in sequences circuit bit here.

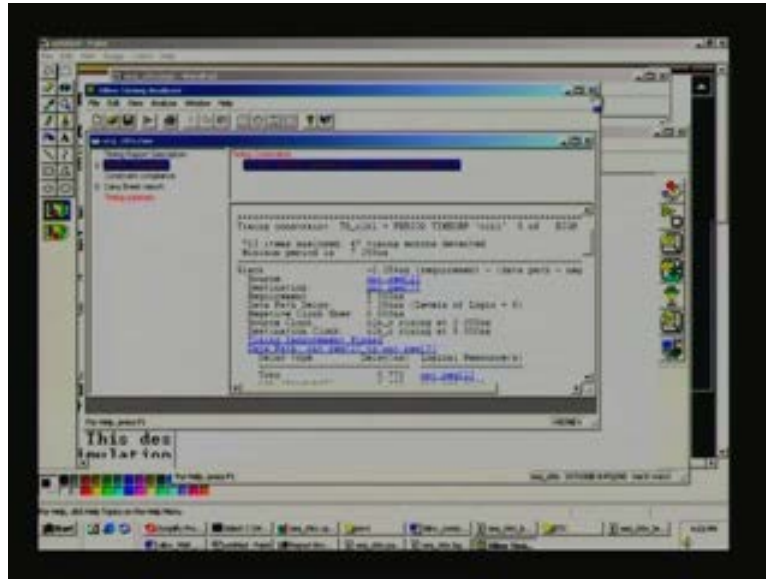
(Refer Slide Time: 33:43)



(Refer Slide Time: 34:08)



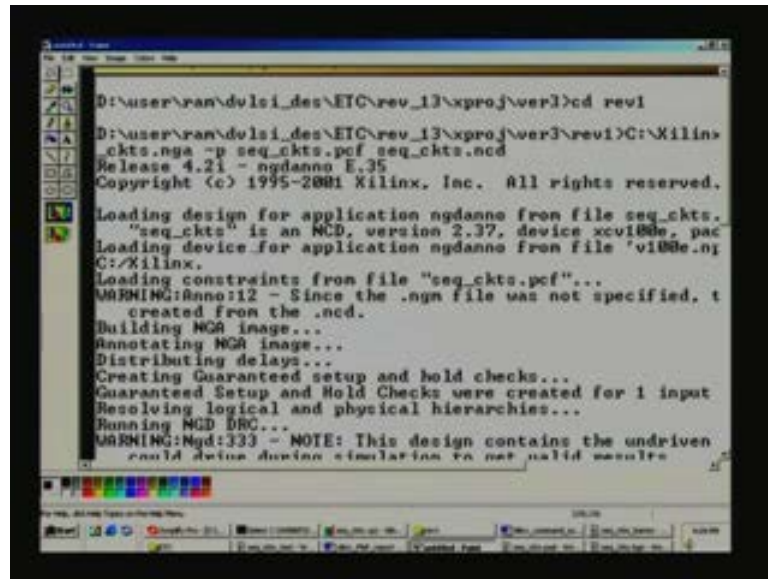
(Refer Slide Time: 34:20)



So, finally will see place and route map are all almost same has log file. Here, so, this is timing file. So, naturally all timing are reported in this, in fact, log file contains almost all of them here; same clock timing its reporting. So, you can have a look all by using, it is not a problem, it also called timing analyzer. You can click various for various constraints, etcetera, and you get corresponding delays and so on.

So, next what will constraint is, the back annotation. So, we need to back annotated because, so for what we have done is only routed placed map on to particular device, then placed and then routed it. We need to get information for timing, it say, for example, you on get delays, so, you need to have this what is called back annotation and dot v file created finally. So, for this you have couple of comments which we can execute in dos mod, and not in the design manager, but state away from the dos mod you can have.

(Refer Slide Time: 35:20)

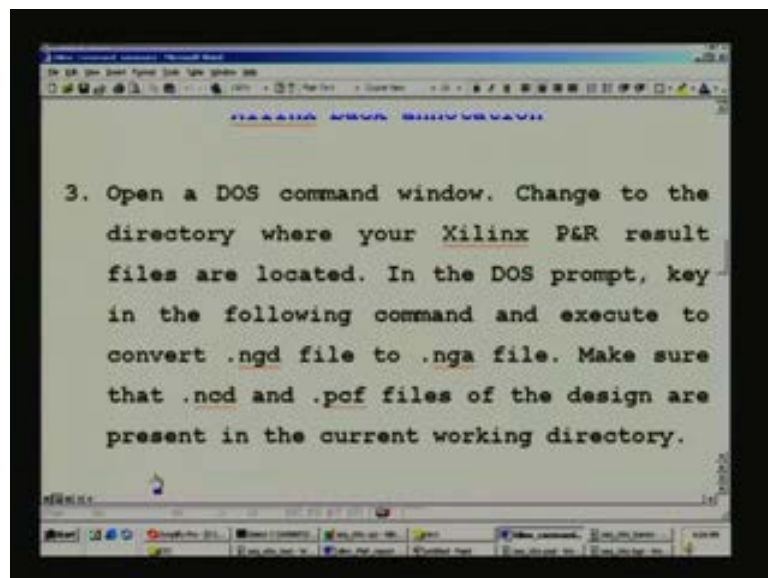


```
D:\user\ram\dulsi_des\ETC\rev_13\proj\ver1>cd rev1
D:\user\ram\dulsi_des\ETC\rev_13\proj\ver1>C:\Xilinx>
_ckts.nga -p seq_ckts.pcf seq_ckts.ncd
Release 4.2i - ngdanno E.35
Copyright (c) 1995-2001 Xilinx, Inc. All rights reserved.

Loading design for application ngdanno from file seq_ckts.
"seq_ckts" is an NCD, version 2.37, device xcvi80e, pac
Loading device for application ngdanno from file 'v180e.ng
C:\Xilinx.
Loading constraints from file "seq_ckts.pcf"...
WARNING:Anno:12 - Since the .ngn file was not specified, t
created from the .ncd.
Building NGA image...
Annotating NGA image...
Distributing delays...
Creating Guaranteed setup and hold checks...
Guaranteed Setup and Hold Checks were created for 1 input
Resolving logical and physical hierarchies...
Running NCD DRC...
WARNING:Ncd:333 - NOTE: This design contains the undriven
could detect during simulation to get valid results
```

This is little time consuming. There are comments here, I will have made into, I mean I have incorporated in the comment summary and I will read out this one. You have a question?

(Refer Slide Time: 35:33)



3. Open a DOS command window. Change to the directory where your Xilinx P&R result files are located. In the DOS prompt, key in the following command and execute to convert .ngd file to .nga file. Make sure that .ncd and .pcf files of the design are present in the current working directory.

See, if a frequency, target frequency is achieved, I did not bother what initially get frequency is at all get delays are, is not it?

So, if my target is achieved, I just did not bother about it, in that case.

In general, that is, the case, suppose you want to improvise, suppose, you have done and deliver the product, they say that it is malfunctioning at particular time, then you need to analyze. So, how will know that, had you analyze before, you would have got in to such a mess later on. So, it is a good practice; as a design engineering, we should go through the entire cycle, do not leave anything is a chance.

Ya

If they target frequency is less 100 megahertz

Yes

It is a good practice to see that it work at straightly higher frequency or if a achieve 100 megahertz it is enough?

As I mention before this a tricky thing because, sometimes it is observed whatever Xilinx as reported as maxim frequency of operation, normally, people have experience 10 to 25 percent better performance, but the some other have also experience the other way; so, one cannot say best thing is to deal case by case. So, for every design, we will have to go through that, but how can we get that one unless you have gone through the entire cycle. If you bypass at just because you want to make a quick but there it, so, I do not really click, you have to go systematically step by step, so as to not to prolong your agony later on, it is better if you sort it out right at the inception stage. In fact, well, what will coding I have mention that technological dependence you may give some gates even assuming synthesis tool keeps in intact.

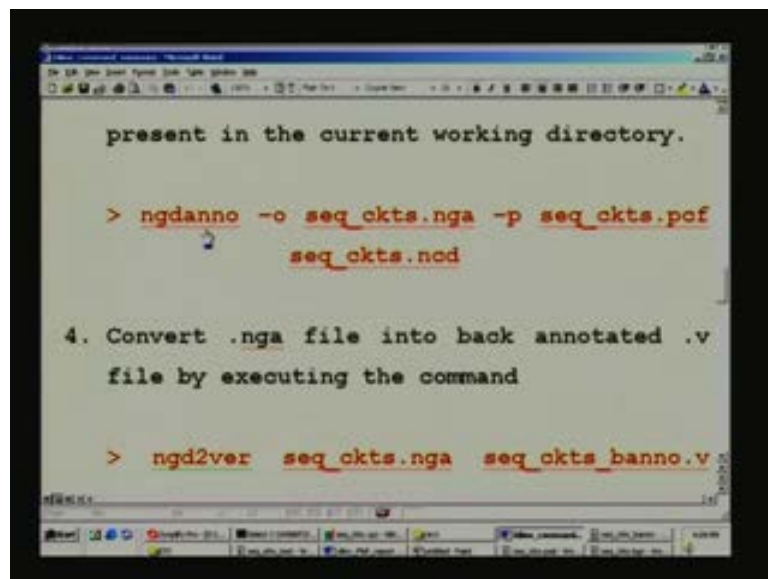
So, even then, you have to go through all that and find out whether the delay, total delay is really up to the mark. There may be some critical paths which you may take it light now and later on that may blow out in to a big problem.

So, it is always better to do it. After all, we are going to do only once in life time of for the product, so, why do not you do it completely. Hardly it is going to be couple of days, not more and will continue with the will sort rather the Xilinx back annotation and the purpose of back annotation to incorporate the gate delays and then go on to the simulator and see actually the gate delays shown as a way form there. In the timing diagram, you

can see the real delays, and we have already seen in the synthesis tool in the wave form analysis prior to this that slight timing is effected in that even after optimization.

So, in place and route it must be more so. So, I will just read out this one. Open a dos command window. You need to open a separate dos window because we want to execute in dos command, and it is just a couple of, mean commands, only you need to run, but it is time consuming, therefore, I have, whatever is the logging of that window I have just preserved it here and we will go through this as step by step. After going through the step by step, I have actually run it just few minutes back and captured on paint and will see that version. Here it is says open a dos command window. Change to the directory where your Xilinx place and router results files are located. Obviously, you have to going to that directory because from elsewhere you had to give the entire path. So, it is a good practice to just change your directory to the place where your P and R results are.

(Refer Slide Time: 41:07)



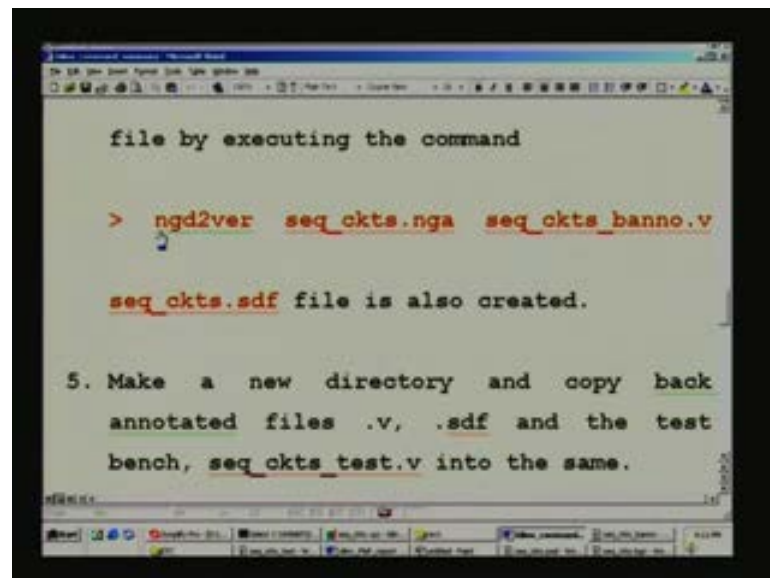
```
present in the current working directory.  
  
> ngdanno -o seq_ckt.s.nga -p seq_ckt.s.pcf  
          seq_ckt.s.ncd  
  
4. Convert .nga file into back annotated .v  
file by executing the command  
  
> ngd2ver seq_ckt.s.nga seq_ckt.s_banno.v
```

In the dos prompt, key in the in the following command and execute to convert dot ngd file to dot nga file. So, this are all intermediate files generated. As I mention there are just two comments, one of which is already p ping here, and the purpose is to have two poses and get the back annotated dot v file. So, first phase is this here and I will just read this. Make sure that dot ncd and dot pcf files of the design are present in the current working directory.

See, well, running the Xilinx place and route, we had already got this files in that. So, I thing it may be one of this here. If you have a look, this is that revulsion one window and you can see that dot ngd here and that is what we have one here. So, ngd file to nga file we need to convert. So, this is being done by this commend. So, make sure that dot ncd and dot pcf, this is a constraint file, that was also created, it is fundamentally the input is uca file, you remember, we had used as an input here and **are** the pca files are the design or present in the current working directory. So, make sure about that. Very first command for this is ngd anno, so, anno is annotation. So, it is first phase. So, **I did not remember the expansion for this one**. It is just it, you get the back annotated file with is commend.

As I mention there are only two commands here, and here, the options are, I mean you had to identify output files that is how identify here. What do you want to create here is an dot nga file, that is why, that was what is the indicated earlier, and some minus p optional so **seq_ckts**, I thing it is for indicating it is pca file. pca pcf constraint file created by Xilinx place and router and that need to be given along with dot ncd file. The output is nga file and if you run this command and I will show after both our covered.

(Refer Slide Time: 42:17)

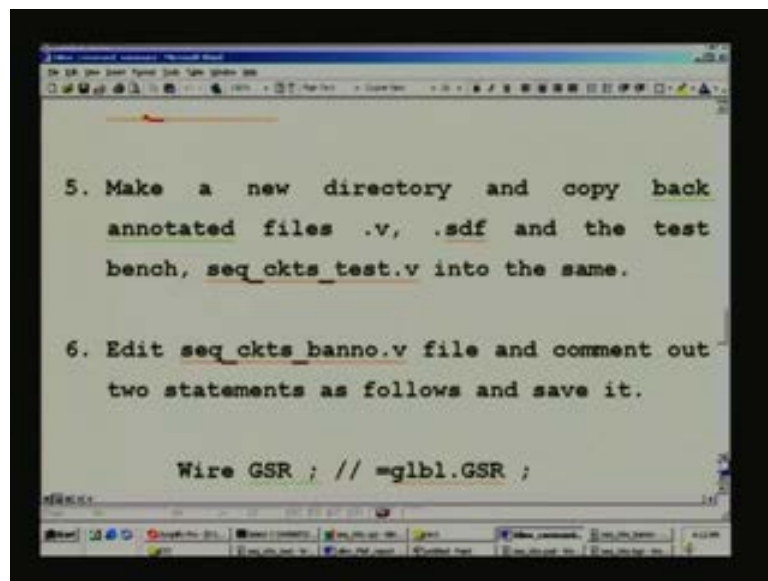


```
file by executing the command  
  
> ngd2ver seq_ckts.nga seq_ckts_banno.v  
seq_ckts.sdf file is also created.  
  
5. Make a new directory and copy back annotated files .v, .sdf and the test bench, seq_ckts_test.v into the same.
```

Then next step is, second phase is, you take this nga file which was created here and then convert in to back annotated dot v file by executing the following command. Here, you have ngd to verilog, ver sense for verilog, so, you want to create a back annotated file

which is nothing other than dot v file, so also a verilog file. That file I have named it like this just to discriminate bit in that origin design and the back annotation. In case you are putting all this in the same directory, suppose you have forgotten this, it will lower right that origin file. So, have a make it habit to indicate, if this is to long, you can say b a r whatever, and then it sorts from nga file which was created then the previous step and that it takes as input and create simply this here.

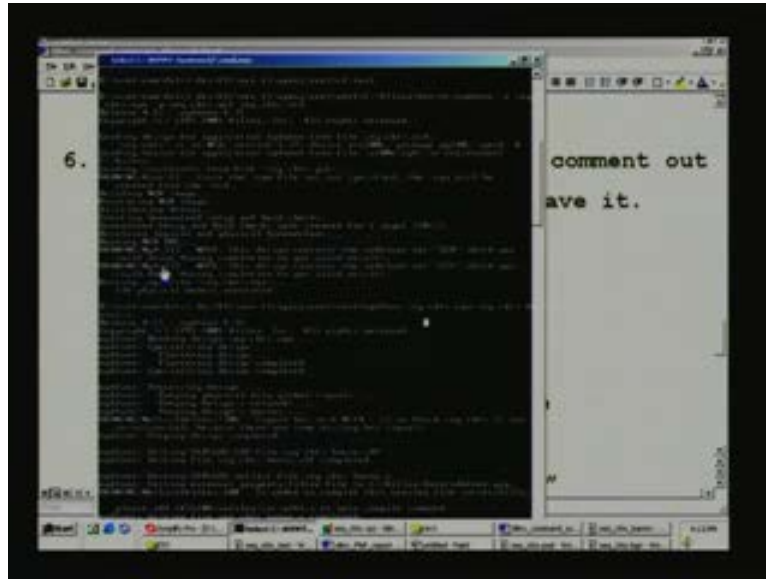
(Refer Slide Time: 43:26)



So, it is just into two phases we should say and we should also make sure, sorry, sequences circuit that sdf file is also created are in this process. This sdf file is the one which as a delay information is called standard delay format. So, this also created here and that is also important. The next step is make a new directory and copy back annotated files. We have just dot v file dot sdf file created on the and also the test bench.

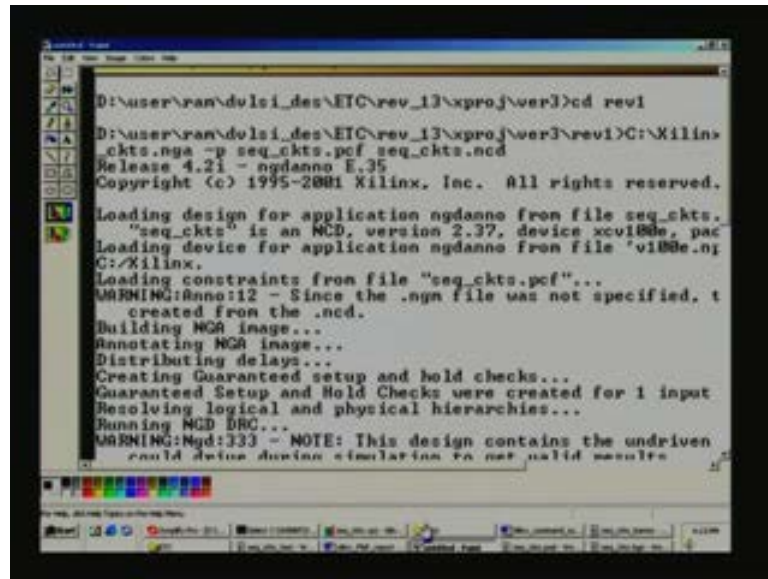
So, if you had these three files, we can go to the next step; let us see what the next step is. Here, I will read it again: make a new directory and copy back annotated files dot v, and dot sdf, and the test bench sequence circuits and score test dot v into the same.

(Refer Slide Time: 44:06)



So, then, before going to **this here, will, for to the** next step, will just have a look at the window. So, in fact, this was what is the actual window which had run the this two, but you can read here this on is the ngd anno minus o that is the very first command, and this, after it has finish all this, it has given some warning also which will going to about some gsr and gts, will going in to that shortly and after that, it has completed this. The second command was given here, that is ngd to where that is a verilog, from nga to the back annotated file. Here, once again its go through, this is a time consuming operation here, it may take some 5 minutes or more, being a small file, but even that is quite a subtraction time here, as for a presentation is concerned. Here, it reports that verilog stf file sequence back annotated dot stf file is created here, that sort it.

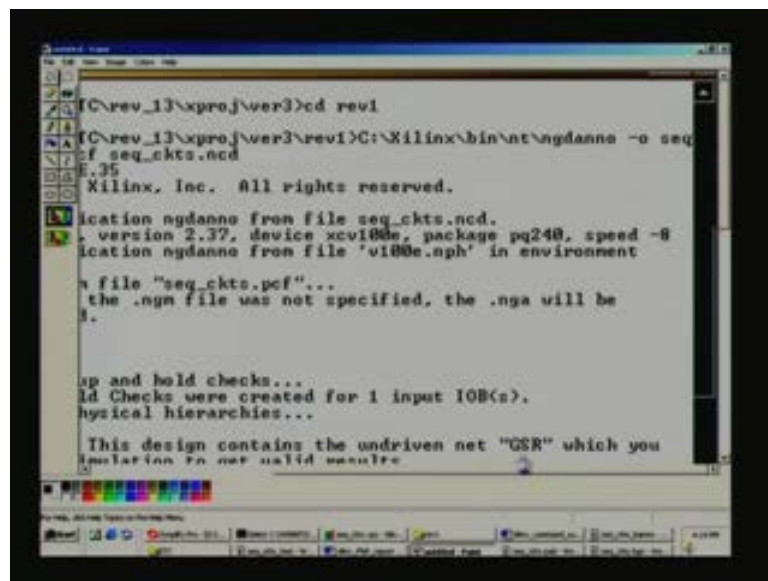
(Refer Slide Time: 45:09)



```
D:\user\ram\dulsi_des\ETC\rev_13\proj\ver3>cd rev1
D:\user\ram\dulsi_des\ETC\rev_13\proj\ver3\rev1>C:\Xilinx>
_ckts.nga -p seq_ckts.pcf seq_ckts.ncd
Release 4.2i - ngdanno E.35
Copyright (c) 1995-2001 Xilinx, Inc. All rights reserved.

Loading design for application ngdanno from file seq_ckts.
"seq_ckts" is an NCD, version 2.37, device xcvi80e, pac
Loading device for application ngdanno from file 'v180e.nj
C:\Xilinx.
Loading constraints from file "seq_ckts.pcf"...
WARNING:ngd:133 - Since the .nga file was not specified, t
created from the .ncd.
Building NGA image...
Annotating NGA image...
Distributing delays...
Creating Guaranteed setup and hold checks...
Guaranteed Setup and Hold Checks were created for 1 input
Resolving logical and physical hierarchies...
Running NCD DRC...
WARNING:ngd:133 - NOTE: This design contains the undriven
could detect during simulation to get valid results
```

(Refer Slide Time: 45:18)

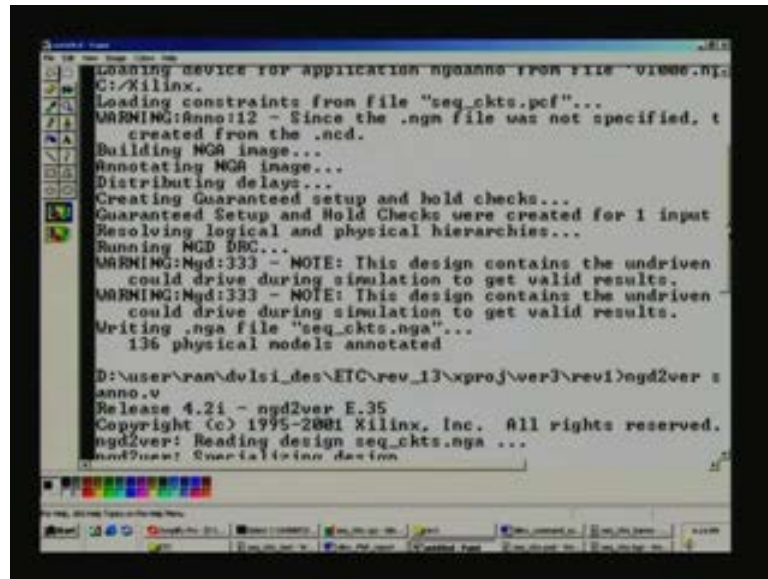


```
D:\user\ram\dulsi_des\ETC\rev_13\proj\ver3>cd rev1
D:\user\ram\dulsi_des\ETC\rev_13\proj\ver3\rev1>C:\Xilinx\bin\nt>ngdanno -o seq
_ckts.ncd
Release 4.2i
Xilinx, Inc. All rights reserved.

Loading design for application ngdanno from file seq_ckts.ncd.
"seq_ckts" is an NCD, version 2.37, device xcvi80e, package pq248, speed -8
Loading device for application ngdanno from file 'v180e.nph' in environment
Loading constraints from file "seq_ckts.pcf"...
WARNING:ngd:133 - Since the .nga file was not specified, the .nga will be
created from the .ncd.
Building NGA image...
Annotating NGA image...
Distributing delays...
Creating Guaranteed setup and hold checks...
Guaranteed Setup and Hold Checks were created for 1 input IOB(s).
Resolving logical and physical hierarchies...
Running NCD DRC...
WARNING:ngd:133 - NOTE: This design contains the undriven net "GSR" which you
should check during simulation to get valid results
```

I have, I think this is the one. The very same thing copied here and that is what you see here. The very first command was: so, this ngd, anno, minus o, sequence, all that, we already seen. Say that nga then minus p option pcf ncd, this is the command we had given. So, web items here; it is a basically the same copy, and you can see that pcf file is being loaded here, and what device, all that is reported here. It is saying annotating nga image and so on and drc check is also run and here is a warning as I mention here, some, here we have be very careful; it is not big thing gsr and gts will just comment it out later on.

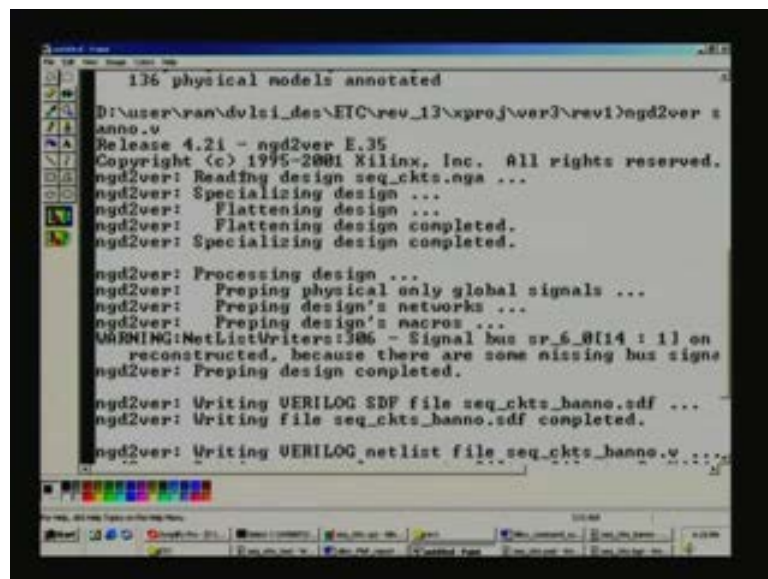
(Refer Slide Time: 45:44)



```
Loading device for application nganno from file "vrome.ngi:
C:/Xilinx.
Loading constraints from file "seq_ckt.pcf"...
WARNING:Anno:12 - Since the .ngm file was not specified, t
created from the .ncd.
Building NGA image...
Annotating NGA image...
Distributing delays...
Creating Guaranteed setup and hold checks...
Guaranteed Setup and Hold Checks were created for 1 input
Resolving logical and physical hierarchies...
Running NCD DRC...
WARNING:Ngd:333 - NOTE: This design contains the undriven
could drive during simulation to get valid results.
WARNING:Ngd:333 - NOTE: This design contains the undriven
could drive during simulation to get valid results.
Writing .nga file "seq_ckt.nga"...
136 physical models annotated

D:\user\ran\dvlsi_des\ETC\rev_13\proj\ver3\rev1\ngd2ver s
anno.v
Release 4.2i - ngd2ver E.35
Copyright (c) 1995-2001 Xilinx, Inc. All rights reserved.
ngd2ver: Reading design seq_ckt.nga ...
ngd2ver: Specializing design
```

(Refer Slide Time: 46:18)



```
136 physical models annotated

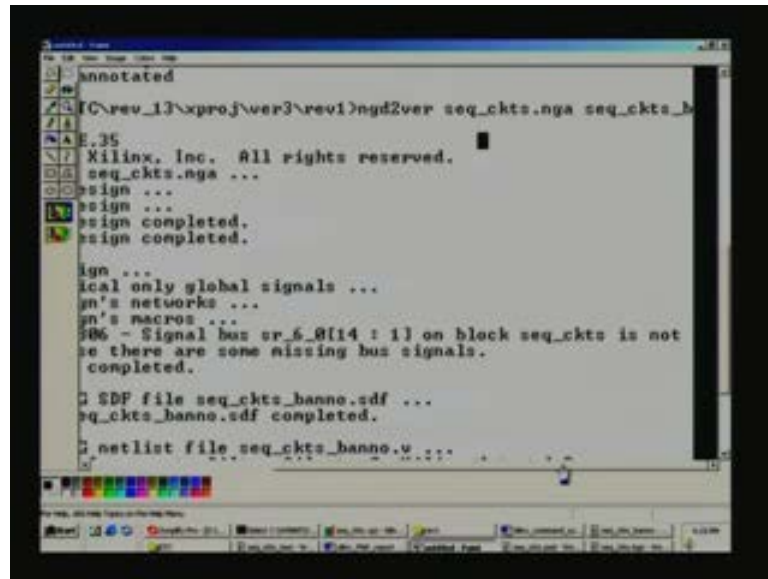
D:\user\ran\dvlsi_des\ETC\rev_13\proj\ver3\rev1\ngd2ver s
anno.v
Release 4.2i - ngd2ver E.35
Copyright (c) 1995-2001 Xilinx, Inc. All rights reserved.
ngd2ver: Reading design seq_ckt.nga ...
ngd2ver: Specializing design ...
ngd2ver: Flattening design ...
ngd2ver: Flattening design completed.
ngd2ver: Specializing design completed.

ngd2ver: Processing design ...
ngd2ver: Preping physical only global signals ...
ngd2ver: Preping design's networks ...
ngd2ver: Preping design's macros ...
WARNING:NetListWriters:306 - Signal bus sr_6_0[14 : 1] on
reconstructed, because there are some missing bus signa
ngd2ver: Preping design completed.

ngd2ver: Writing VERILOG SDF file seq_ckt_banno.sdf ...
ngd2ver: Writing file seq_ckt_banno.sdf completed.

ngd2ver: Writing VERILOG netlist file seq_ckt_banno.v ...
```

(Refer Slide Time: 46:32)



```
annotated
[C:\rev_13\proj\ver3\rev1\ngd2\ver seq_ckts.nga seq_ckts_b
E.35
Milinx, Inc. All rights reserved.
seq_ckts.nga ...
esign ...
esign ...
esign completed.
esign completed.

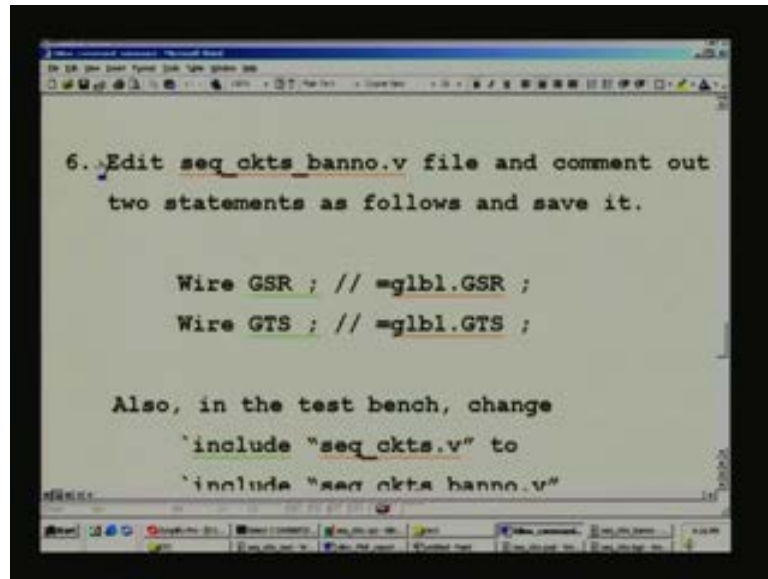
ign ...
ical only global signals ...
gn's networks ...
gn's macros ...
[B6 - Signal bus sr_6_B14 : 1] on block seq_ckts is not
e there are some missing bus signals.
completed.

SDF file seq_ckts_banno.sdf ...
eq_ckts_banno.sdf completed.

netlist file seq_ckts_banno.v...
.
```

The next step, height itself, nearly I warning there, and second command is here, ngd where two sequence circuit dot nga, then this is the back annotated file here. So, it is reading the file, then specialize flattening, specializing design completed, all that its report here. Once again some warning is there, so, we had to look into warning, should not just take it light. It says, on block sequences circuits, some of them are missing or something its saying. Re-constraint because there are some missing bus signal here. So, any way we are going to take this one into the model simulator and verify the functionality. So, we do not had to read in-between lines. So, finally on sdf file is created here, writing into that, its says and that is completed its says, and then a net list, verilog net list, that is dot v, this is the file which we need is created here. Once again some warning is given here, and finally, its say some its completed. Its only reminding of some procedure; so, this warring is not a serious thing.

(Refer Slide Time: 47:32)

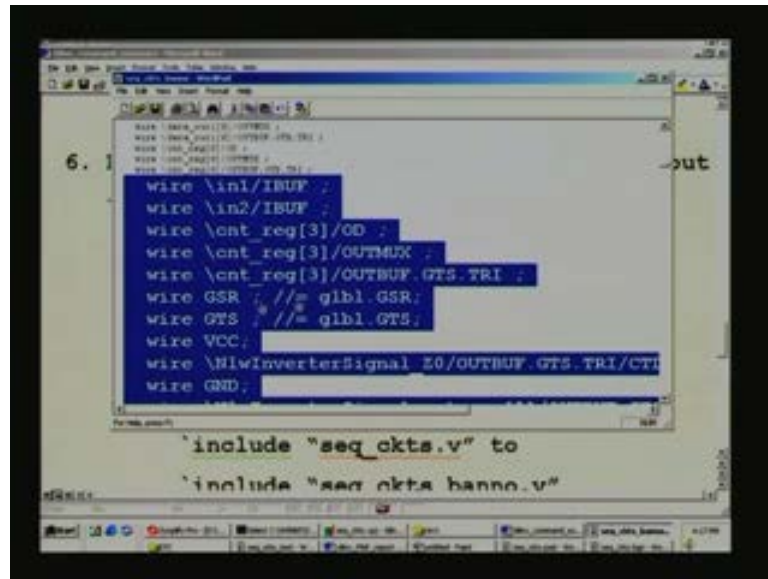


```
6. Edit seq_ckt_banno.v file and comment out  
two statements as follows and save it.  
  
Wire GSR ; // =glbl.GSR ;  
Wire GTS ; // =glbl.GTS ;  
  
Also, in the test bench, change  
`include "seq_ckt.v" to  
`include "seq_ckt_banno.v"
```

So, now, what will do is, will go back to the summary and continue from where we left. Here, I readied out: edit sequential circuit back annotated dot v file and comment out two statements on as follows and save it.

So, we need to identify to this one that was the last warning we saw there, that is what earlier also it have given. So, what we need to do is, they last one he suggest another alternative, so, I have taken some other alternative. So, in this alternative, what I am doing is, I am going to the back annotated file, editing that one, and nearly comment it here, and terminating this, after this, originally this was not there. So, this was equated here; so, only two statement.

(Refer Slide Time: 49:12)

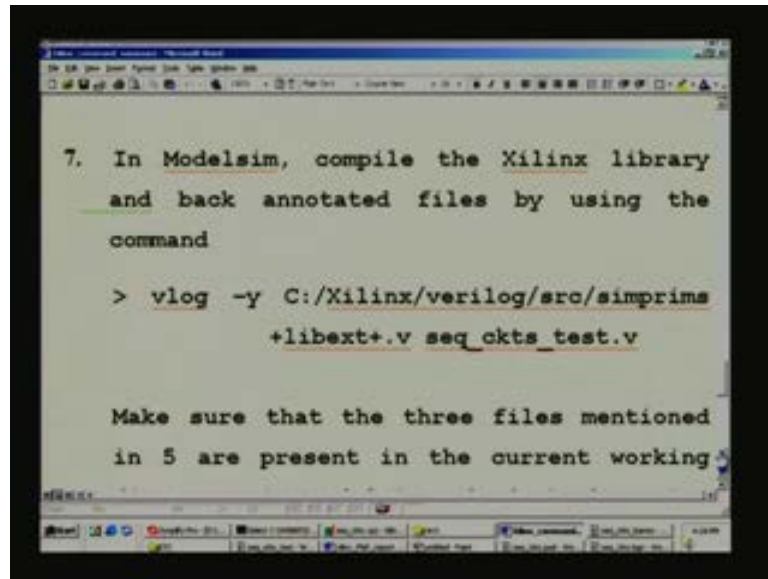


```
6. 1
wire \in1/IBUF ;
wire \in2/IBUF ;
wire \cnt_reg[3]/OD ;
wire \cnt_reg[3]/OUTMUX ;
wire \cnt_reg[3]/OUTBUF.GTS.TRI ;
wire GSR ; // glbl GSR;
wire GTS ; // glbl GTS;
wire VCC;
wire \MwInverterSignal_20/OUTBUF.GTS.TRI/CTI
wire GND;

`include "seq_ckt.v" to
`include "seq_ckt_hanno.v"
```

In order to do this one, you can use find and locate that particular file, it is here. So, you can use find, and then, I just miss, it was already there, I think I do not to say. This is back annotated file, once again you can see lut four, etcetera, all the primitives are there, time scale, all that are here. If you want you just say find - wire gsr - it will find it again, that g 1. So, if you just want this here, so, you can just, see, these are the two statements which I have already commented. Only after this step, we can go for the next step. A next step is, in addition to this, you had to go to the test bench, and then change, we had use dot v m for that earlier, you remember for that is synthesis, instead of dot vm, it should be this back annotated file. So, it is exactly same thing we are going to do here, and again after this run the model same, that is the next step here.

(Refer Slide Time: 49:48)



```
7. In Modelsim, compile the Xilinx library
and back annotated files by using the
command

> vlog -y C:/Xilinx/verilog/src/simprims
+libext+.v seq_ckts_test.v

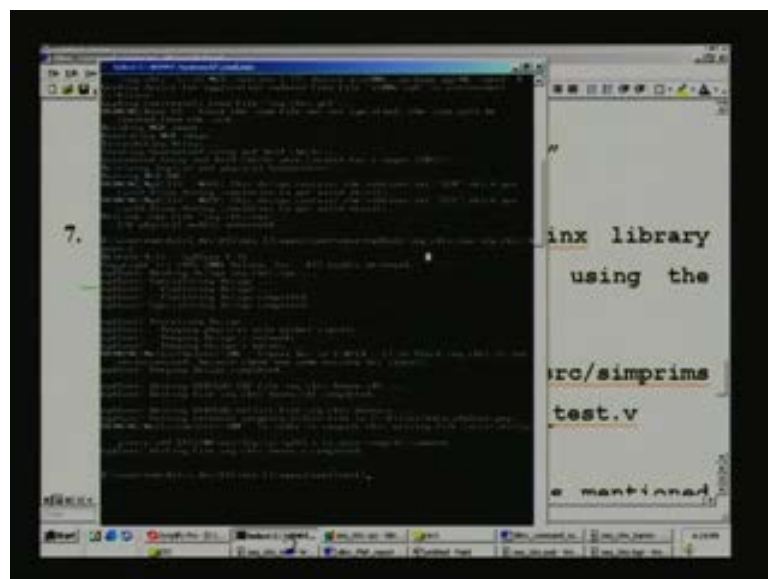
Make sure that the three files mentioned
in 5 are present in the current working
```

So, we need little more time for this, and that will cover. It is a hardly another three more statement we need to go for, and if you have any question, you can ask, because I am going to stop at this point of time, because this is going to be involved, so, will see in the next lecture.

Have any question?

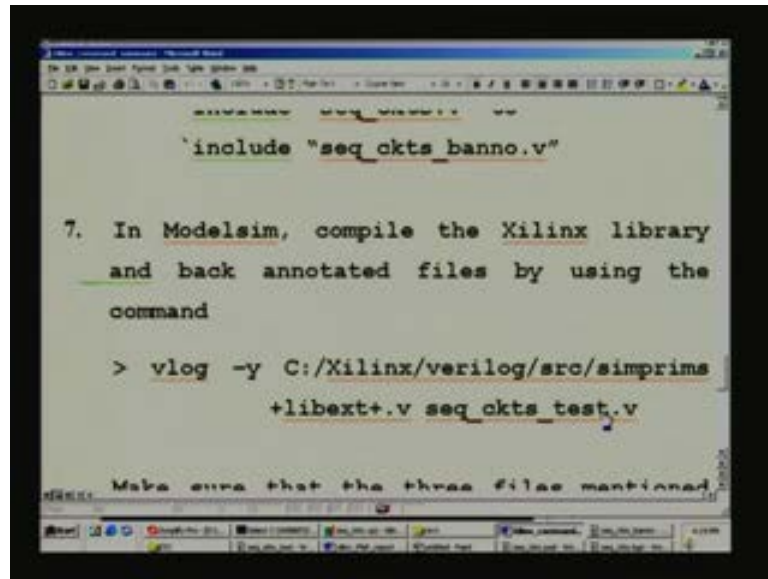
Why did you do that commenting out?

(Refer Slide Time: 50:20)



```
7. In Modelsim, compile the Xilinx library
using the
C:/Xilinx/verilog/src/simprims
test.v
files mentioned
```

(Refer Slide Time: 50: 59)



```
include "seq_ckt.v"
`include "seq_ckt_banno.v"

7. In Modelsim, compile the Xilinx library
and back annotated files by using the
command

> vlog -y C:/Xilinx/verilog/src/simprims
+libext+.v seq_ckt_test.v

Make sure that the three files mentioned
```

That is because it has as for some comments here, I will find out what that comment is. What it says is - please add Xilinx very log source glbl dot v to your compile comment, because this are all reserved words, some library is being will be invoke. So, unless that you specifically mention it one go other and pick it up. So, these are all mandatory requirements. So, either you can do the Xilinx way suggested or this way also. I preferred that way because it was easier, just commenting out. Otherwise, I had key in this much, **at the time of**. So, next step here is, to take this back annotated file and going to the modalism and once again view the way from, see, whether the delay has been corporate or net. We will continue in the next class. Thank you.