Digital VLSI System Design
Prof. Dr. S. Ramachandran
Department of Electrical Engineering
Indian Institute of Technology Madras
Lecture. - 32
Technology View Using Symplify Tool

(Refer Slide Time: 01:00)

Lecture 32

Technology View using Synplify Tool

Previous lecture: Synplify tool- Schematic Circuit Diagram view.

(Refer Slide Time: 01:13)

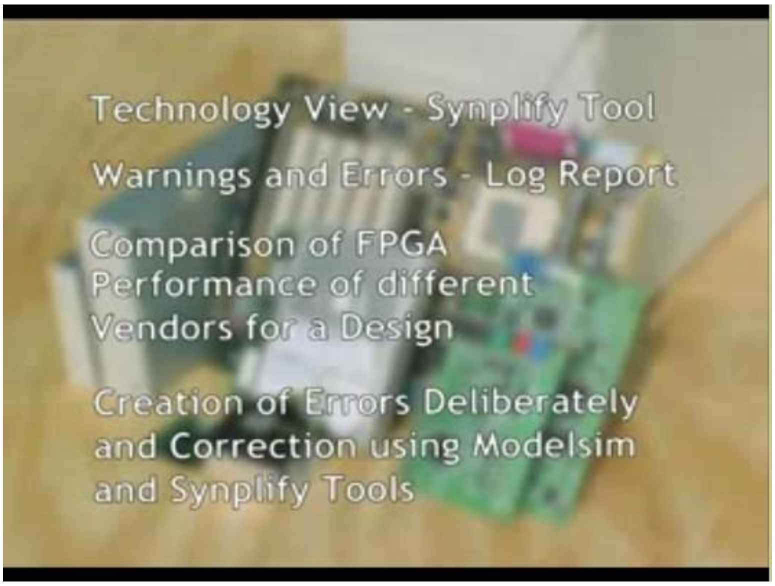
Analysis of the Report File generated by Synplify Tool

Commands Continued - Optimized Verilog File using Synplify Tool

Viewing Verilog Code as RTL Schematic Circuit Diagrams

Contents of Lecture 32:

(Refer Slide Time: 01:42)



Technology View - Synplify Tool

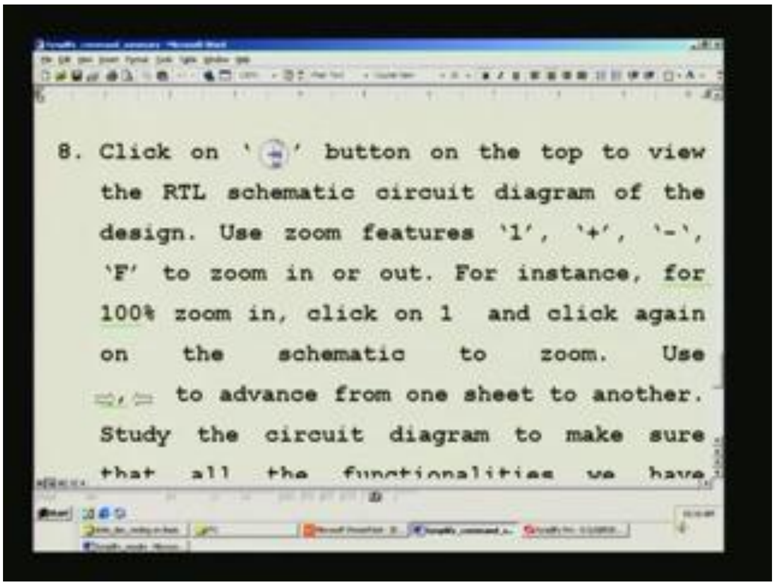
Warnings and Errors - Log Report

Comparison of FPGA Performance of different Vendors for a Design

Creation of Errors Deliberately and Correction using Modelsim and Synplify Tools

We were looking at simplified tool in order to synthesis our design. We had a look at schematics known as RTL view. We will be looking at technology view.

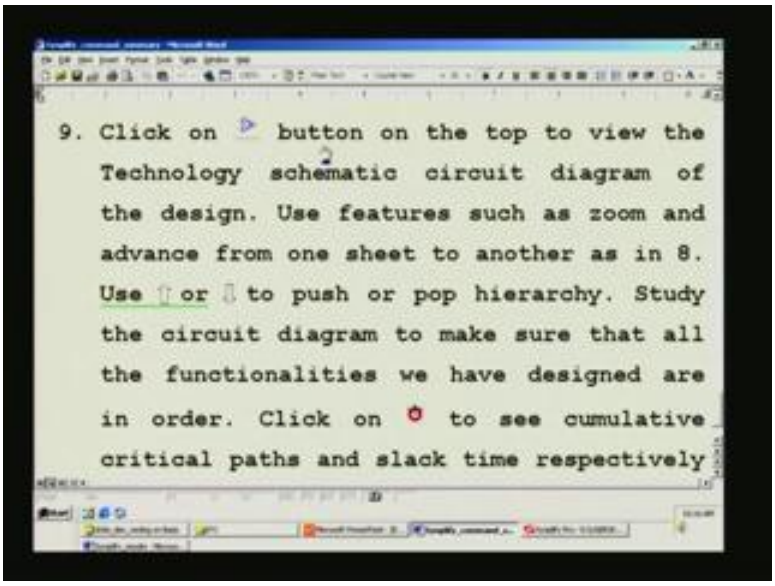
(Refer Slide Time: 02:30)



8. Click on '' button on the top to view the RTL schematic circuit diagram of the design. Use zoom features '1', '+', '-', 'F' to zoom in or out. For instance, for 100% zoom in, click on 1 and click again on the schematic to zoom. Use ⇨, ⇦ to advance from one sheet to another. Study the circuit diagram to make sure that all the functionalities we have

Technology view as you see here, this is for RTL view.

(Refer Slide Time: 02:35)

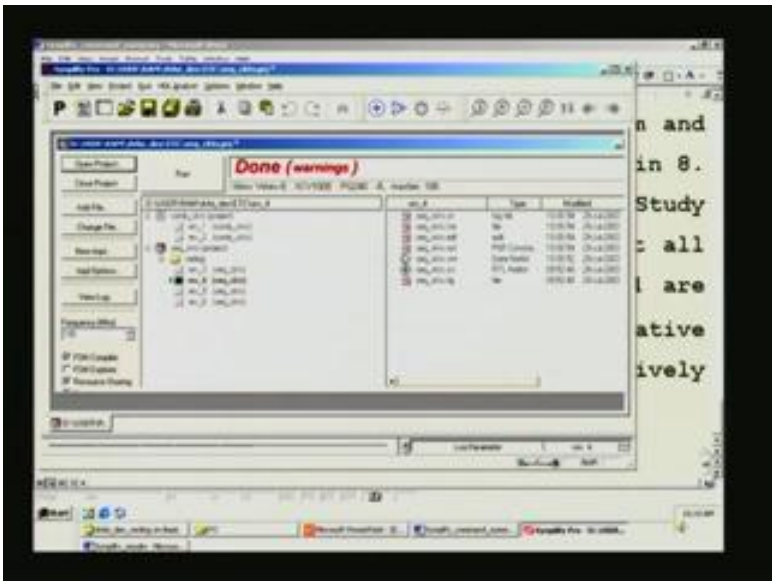


9. Click on ▷ button on the top to view the Technology schematic circuit diagram of the design. Use features such as zoom and advance from one sheet to another as in 8. Use ⇧ or ⇩ to push or pop hierarchy. Study the circuit diagram to make sure that all the functionalities we have designed are in order. Click on ◉ to see cumulative critical paths and slack time respectively

The RTL view is invoked by a plus symbol on the menu. Now we will be invoking another symbol. First I will read out what we need and then will go into the synplify window and learn more about it. Click on this button on the top to view the technology schematic circuit diagram of the design. Use features such as zoom and advance from one sheet to another as in 8. We already saw how to advance from one sheet to another that is with the horizontal arrows. In zoom also we have seen one plus minus and so on. You can use up arrows and down arrow to push or pop hierarchy. This will become clearly when you look in to the actual systematic. You can go down the level and have a peep into it. It may not be very much down maybe just one layer or exactly the same thing that you see on this systematic on the top. As usual you have to study the circuit diagram to make sure that all the functionalities we have designed are in order.

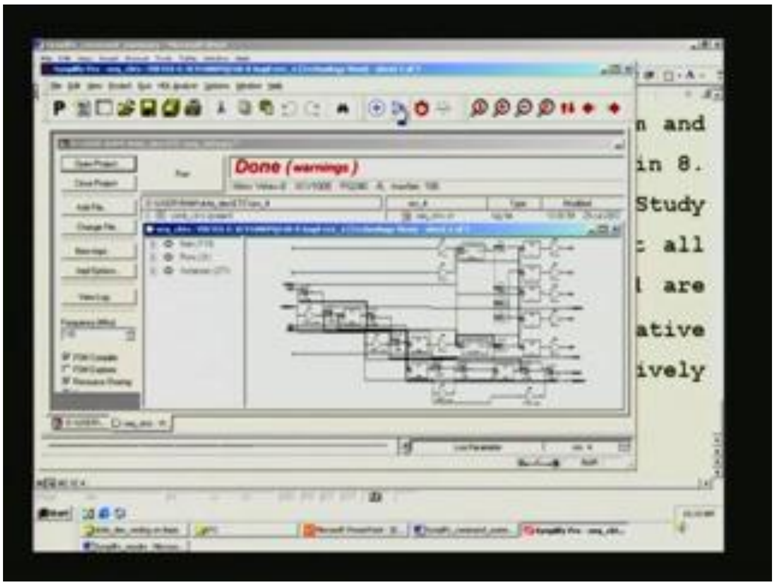
Also I mentioned this need not be done for every design and it is only for the beginner to ascertain that it is doing the logic optimization. From that point of view at least once in your life time, I recommend that you go laborious through this and there after you do not have to go unless you have some specific doubt and in a particular circuitry. You can also use a clock symbol and to see a cumulative critical paths and slack time respectively on the circuit. In fact we have already seen in the view log file by opening which we could see the critical paths. You could trace all the entire path and associated delays with gates, the propagation delays and so on. How much we specified and what finally it turned out to be and if there is a difference that is called this slack time. If it is positive you have more cushions, if negative it has not met the constraints that you have asked for. Naturally, it calls for either a redesign using heavier pipe lining or bring down your actual frequency operation to what is being reported. Click on this symbol to see cumulative critical paths and slack time on the circuit respectively. Now we will open the synplify window, to run exactly the same thing.

(Refer Slide Time: 05:22)



We have here in addition to the plus symbol, you have technology view here I hope you can see on the monitor.

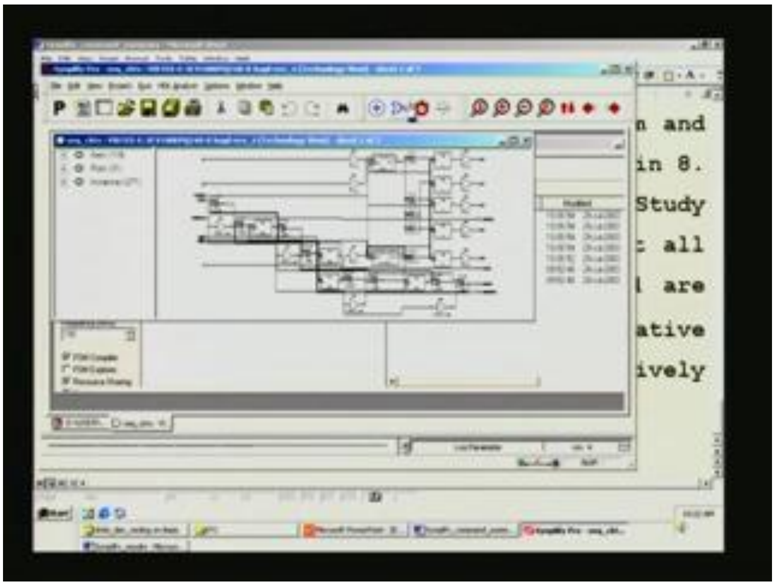
(Refer Slide Time: 05:35)



What you see is the technology view and I forgot to tell one more thing, before you go for this, I have changed the design file from the combinational circuits to sequential circuits. No matter what your design file is, it is exactly the same that you have to follow that what you have done for combinational circuits. It is absolutely the same thing irrespective of the design so that is the beauty of synthesis tool. You do not have to go much specific about your design for time being, unless you are analyzing a portion of circuitry and here I have already made 2 3 revisions. I can expedite the explanation so that I may not repeat same thing over and over again. I have added means, same file is being used for all the revisions. This is for sequential circuits and if I click on revisions 3, xcv 50 e and packages 144 pin with a dash 8 speed. If it is revision 4 I have xcv 100 e, this vertex higher speed is possible and dash 8th is highest speed. I hope you will have a look into the results then we will know whether it is so. The packages 240 pins and it is called pq instead of hq, I think we used earlier and do not worry about fan out etc. To what it refers to is not clear from this and it is railing series and we have one more here.

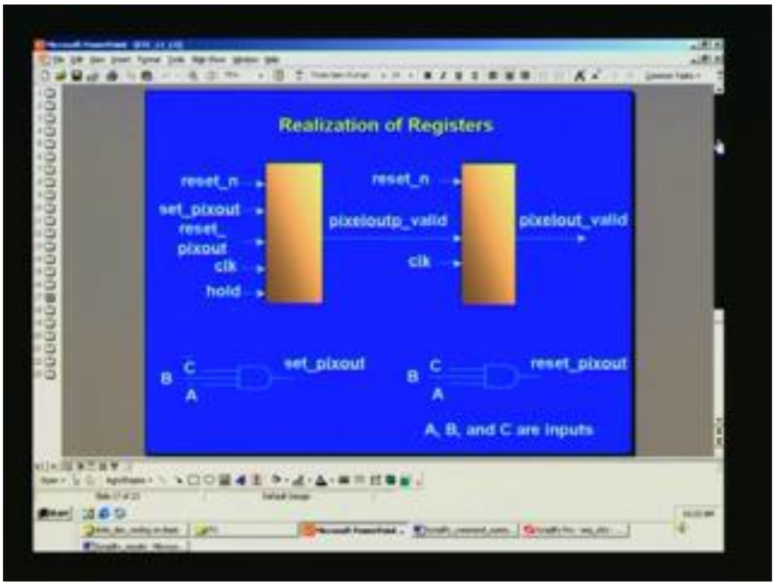
This is xcv 100 e pq 240 dash 6. If you have a look at the new implementation this is speed xcv 100 is mapped here, I mean, selected here on this and if you just click on speed you will see 3 speeds dash 8, dash 7 and dash 6. I had deliberately chosen two of this speed because we can make a comparison as to how much speed difference you can get. That is the motive there. In addition to this we would also like to compare with another vendor more or less same capacity. I have done for another leading supplier for pgs. That is Altera, one of the popular series in Altera is flex 10k series. The device selected is EPF 10k 100 A R C 240 dash 1. In Altera, I think smaller the number higher the speed. You experiment all though I have put only 1. We will be having a look. I have already run all this that is why you have it here. You will straightway see the results. We do not have to run the same thing, because we have already seen all that in combination circuits. First we will have a look at this technology view before you proceed with this. We will go for XCV 100e pq 240 dash 8. This is the highest speed available in 100,000 gates capacity. It can accommodate approximately 100,000 gates plus ram as well. It is a 240 pin package this is highest speed. Let us give the technology view.

(Refer Slide Time: 09:12)



You have here in technology view is, 7 sheets for the sequential circuits.

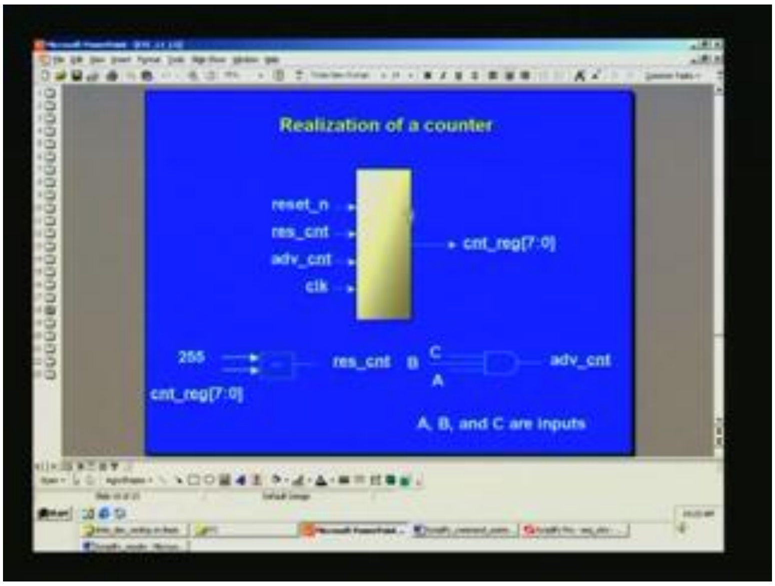
(Refer Slide Time: 09:25)


Realization of Registers

A, B, and C are inputs

In sequential circuits, what you have is, quickly go through what they are. This is sequential circuits and what you have done is registers and I am not going to the details I am just asking you to recollect what you have done earlier.

(Refer Slide Time: 09:37)

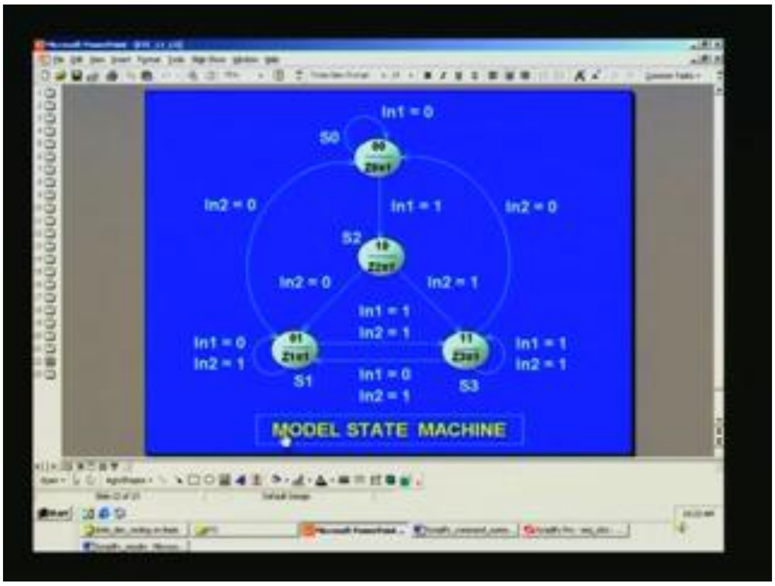


Realization of a counter

A, B, and C are inputs

Then a counter, mono-shot, retractable mono-shot followed by shift registers for 16 bits and finally a parallel to serial converter.

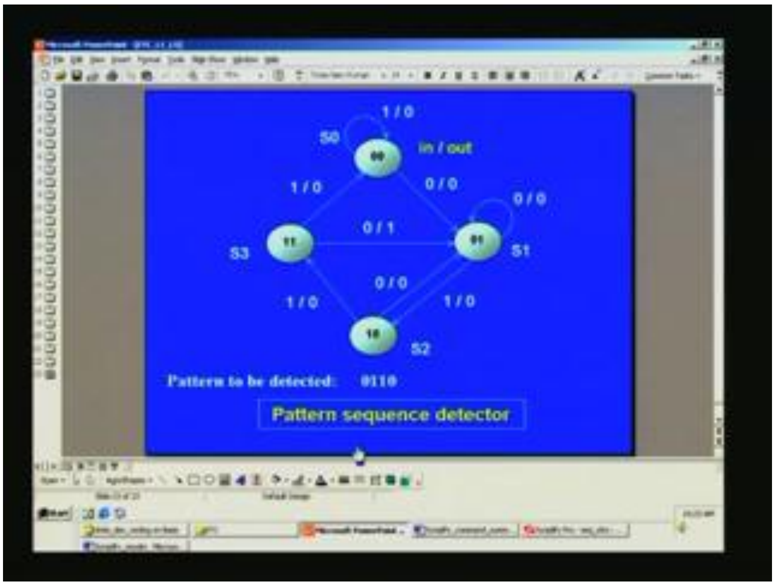
(Refer Slide Time: 09:57)



MODEL STATE MACHINE

I think this is the final state machine.

(Refer Slide Time: 10:02)

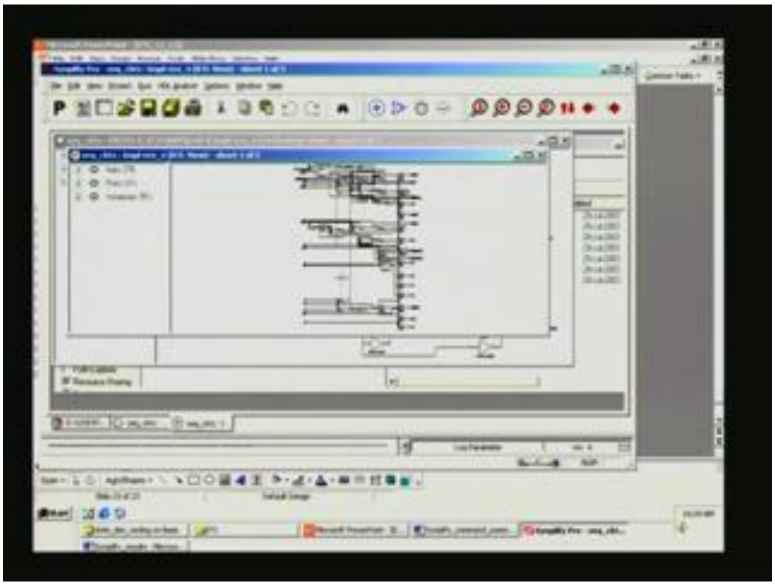


Pattern to be detected: 0110

Pattern sequence detector

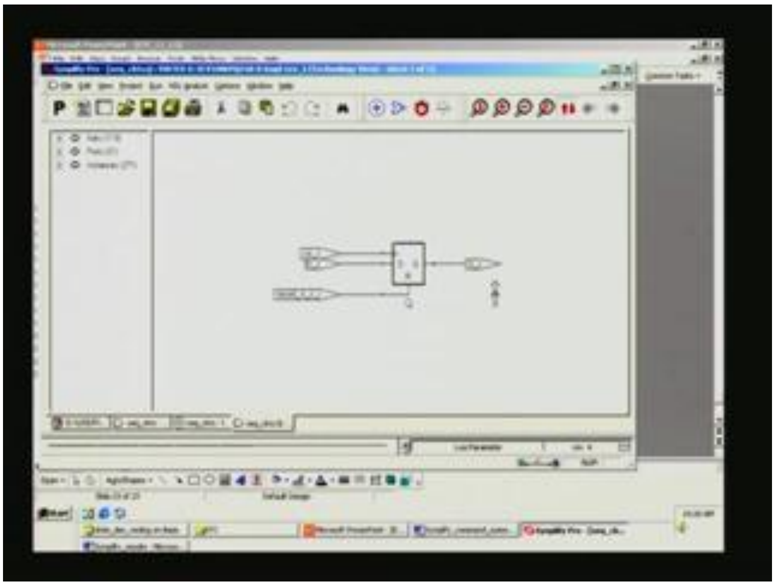
One more is that, there is a pattern sequence detector also.

(Refer Slide Time: 10:35)



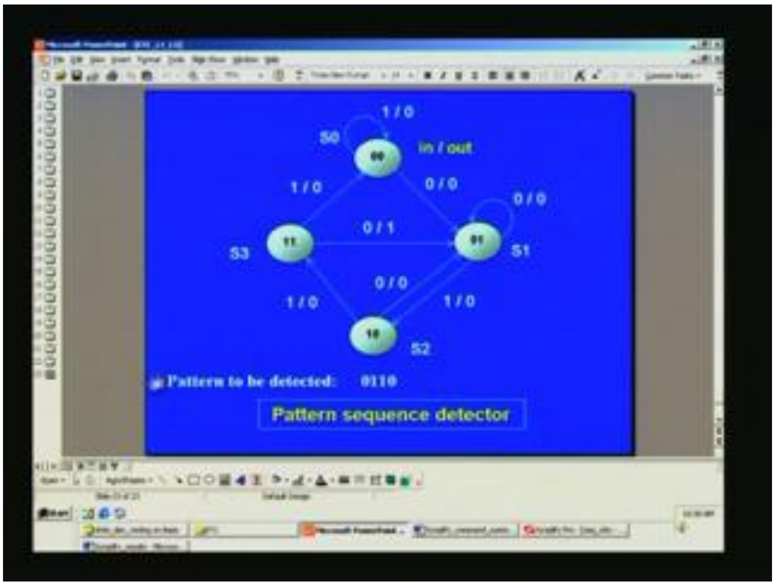
All this put together is sequential circuits. This is the circuitry for that. It runs in to 7 schematics. As I mentioned, to weight through systematic it is a very difficult process even after optimization. At least you must do for one of that as I recommended earlier. This is the technology view; whereas, in atrial view it is only two sheets. For sequential circuits if I blow it up, this is what you get here. Some counters are there. You just look at the outputs and then find out. So this is for mono-shot and shift registers here. Some D flip-flop also we have realized. It is quite cumbersome to go through this circuitry obviously. Imagine had you starter with systematic diagram, what will be the state? I mean in contrast with that we have seen that Verilog coding is much simpler so you have only to write few assign statements are always statements and so on. Finally, when it is translated into circuit it becomes very complex. As we have seen here in the technology view we have some 7 sheets. As mentioned before you can go from one sheet for another just by clicking this arrow. Right now it is on 2nd, then 3rd, 4th and so on. 7 seem to be much simpler. Even that is quite complicated, you can see buffers and deep flip-flops which we could not see in combination circuits, you can see here. Another thing that you have to learn is there is one up down arrow, if you click on this, you can go inside this block. Let us see if more details are available.

(Refer Slide Time: 12:34)



What all it means is it is just a deep flip-flop with a reset pin. It puts all the symbols for the signals that is being connected to this, which will not be apparent if you go up. If you just click on this, it will take you back from where you came. Here it may be difficult as I mentioned you can disable this, put the cursor and find out, what this signal is. It is so small that you cannot read. It is a D input for this. It is all same, no matter what circuit it is. It is going to be either combinational circuits or flip-flops. All this you have already seen in log file. You have an I buff which is D input. Nothing more other than all these primitive cells which are already explained 4 input LUT, You can just see 1 2 3 4 inputs are there. Actual logic realizes using the 4 input true tables. It is what you see here with inverter. From this you cannot have much information except u go deep into the circuitry and find out what it has to tell you. Before you wind up there is another interesting thing which will show you the critical path. You can see here it is exactly same thing, there is some more available here. You can take any two numbers 1.8 or 996. One will be the cumulative delay which is probably this 990 second, all in nanoseconds or in picoseconds. We have already seen in the table earlier in the log file. This one is the slack time. There may be plus minus, if you see somewhere else and if you see negative sign somewhere you can make it out. Unfortunately no negative sign. I am almost sure, first one is the actual delay at this point although put here it means this delay is at this point, not at input but at the output. That will be cumulative. You can trace the entire path and find out where more time is being taken, thereby try to optimize by inserting a register by way of pipelining. This would complete the schematic view.

(Refer Slide Time: 16:11)

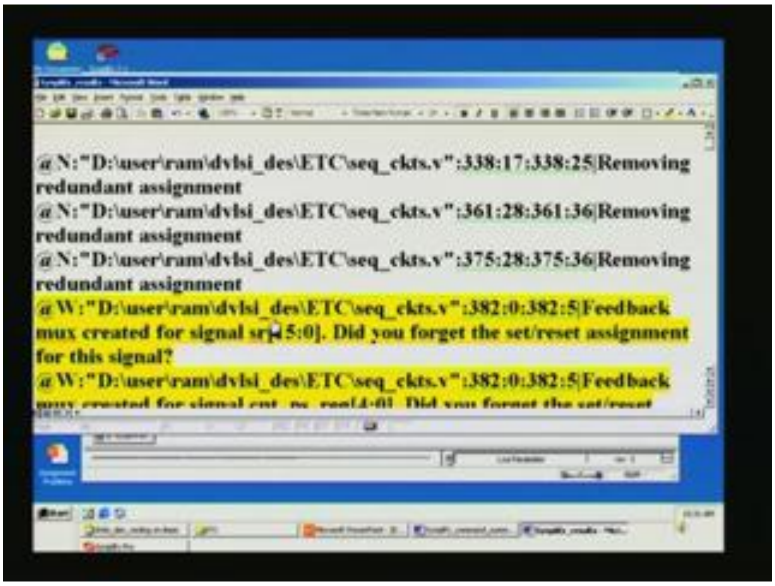


Pattern to be detected: 0110

Pattern sequence detector

Let us see what we have further. We do not require the details because our intention is only to do synthesis and not analysis which we have already done. You can create dot vm file then input it to the simulator tool and find out whether optimization has not taken any wanted functionality. That is how you ensure fastest way rather than go through the circuit diagrams, although cursory glance can help. In this window next we will dismiss this, you do not require, and it is asking for save and unfortunately I have closed the window. Any way you do not require this because I have put in separate file.

(Refer Slide Time: 17:40)

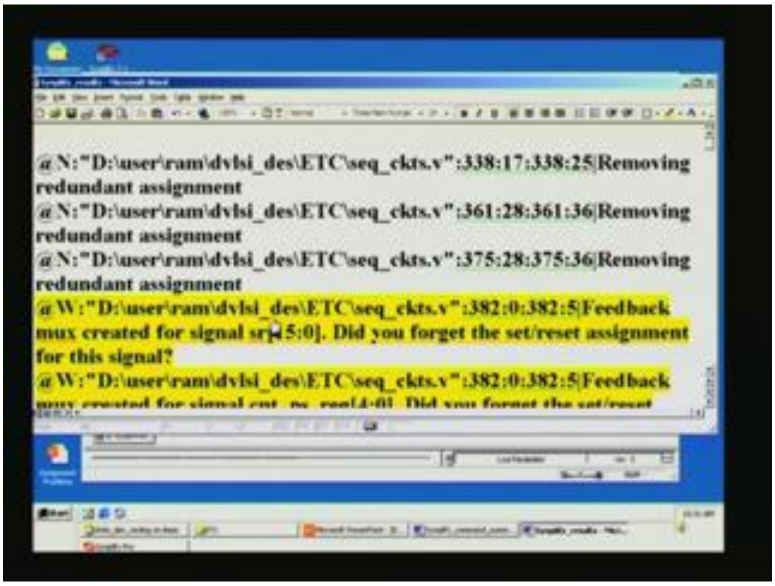


```
@N:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":338:17:338:25|Removing
redundant assignment
@N:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":361:28:361:36|Removing
redundant assignment
@N:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":375:28:375:36|Removing
redundant assignment
@W:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":382:0:382:5|Feedback
mux created for signal sr[15:0]. Did you forget the set/reset assignment
for this signal?
@W:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":382:0:382:5|Feedback
mux created for signal cnt_ps_reg[4:0]. Did you forget the set/reset
```

We have seen one device that is xcv 100 dash 8. In log file if you see it will report. I have just pasted here and I told you that I should hit the warning set, the log file reports. For this sequential circuit there are two warnings given here. One is it implies the sequential circuits taken as the design. Once again, I want to emphasise that no test bench must be given but only you have to give your top design. It says reports feedback must be created for signal 15 to 0. Did you forget the set reset assignment for the signal? It is just a warning because it does not find, let see why it is reporting.

The warning that we have here, this did you forget, this set reset assignment for the signal. Similarly for one more signal it gives the very same warning. It says feedback must be created for signal count PS reg. Once again it is asking same question. I have copied that portion of the Verilog code here. It says I have forgotten the reset portion for that particular signal because it does not find those two variables. This is the reset part of the block the whole block will be if nested, that is the statement we had seen. When the system is reset then we need to reset only these as I had said. It is reminding me, whether I forgot those two signals.

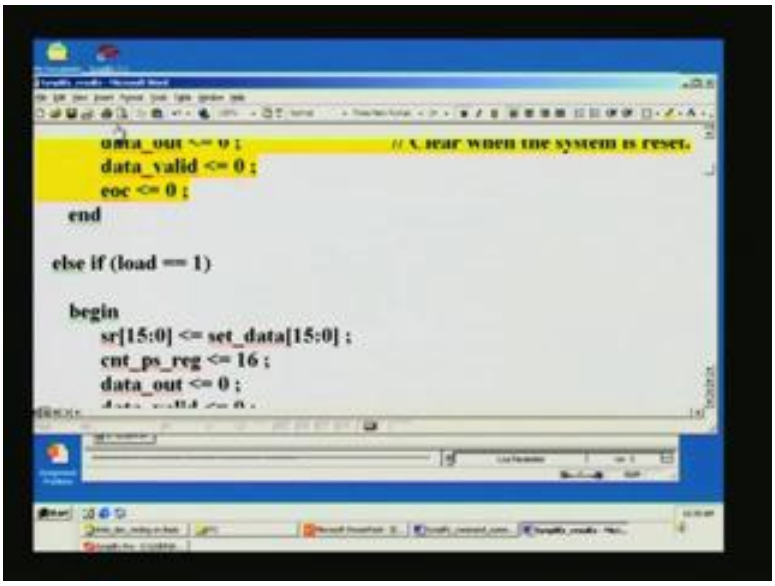
(Refer Slide Time: 20:06)



```
@N:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":338:17:338:25|Removing
redundant assignment
@N:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":361:28:361:36|Removing
redundant assignment
@N:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":375:28:375:36|Removing
redundant assignment
@W:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":382:0:382:5|Feedback
mux created for signal sr[15:0]. Did you forget the set/reset assignment
for this signal?
@W:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":382:0:382:5|Feedback
mux created for signal cnt_ps_reg[4:0]. Did you forget the set/reset
```

Naturally this is very alert state and it is reporting but you have the right to veto it, need not bother about warning if it does not make any sense or it needs to be ignored. Let us see here. What does it say here? Did you forget the set reset assignment for the signal? What is the signal? S R 15, here count P S_ reg.

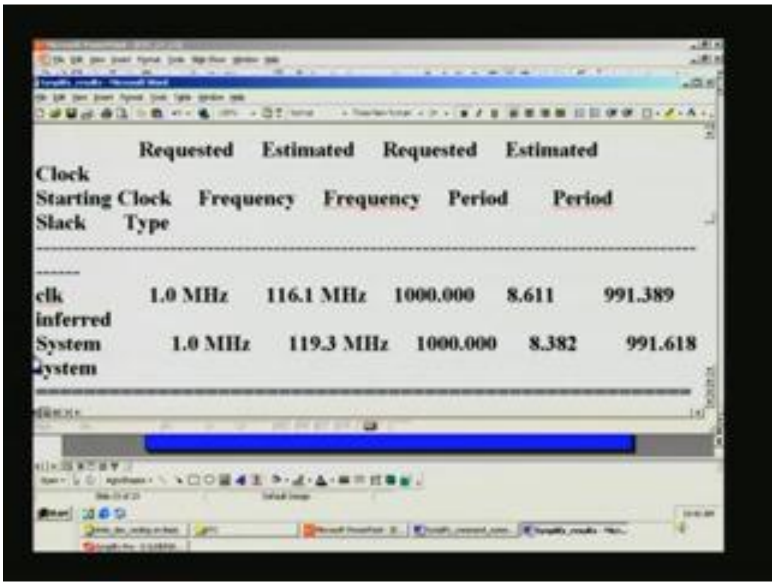
(Refer Slide Time: 20:32)



```
        data_out <= 0 ;                         // Clear when the system is reset.
        data_valid <= 0 ;
        eoc <= 0 ;
    end

  else if (load == 1)

    begin
        sr[15:0] <= set_data[15:0] ;
        cnt_ps_reg <= 16 ;
        data_out <= 0 ;
        data_valid <= 0 ;
```

Obviously the signals are not appearing here. Let us see whether it is appearing elsewhere. For example, for this count PS actually, if you see this power point here I do not think it is available in this so it does not matter because we have already seen earlier. All you need to do is, the point is very clear. In reset these two signals are not taken in to account. Why it was not taken into account? It is nothing but a shift register here. We are setting data from input. We were setting here, since it is being loaded because of load condition. We do not need to initialize when you reset the system because we need to load the things first. I think it is pattern sequences detect. How many bits have to be sent we put it in account, remember this pattern sequence is director. What you see here finally is precisely this but unfortunately you cannot see anything from the state diagram. So, both of them are loaded therefore need not be reset. Although synthesis tool has reported we need not brush it up. That is what I am trying to say the two warnings. Just because it has reported warning we should ponder over and find out whether we have really missed something in our design. It may be serious design flaw. There you always have a look at it, give due consideration and make a design whether to ignore the warnings or not. It will not prohibit you from going further but if there are errors you cannot go further. These are the other portion of the same code. We will see performance summary from the log file for the device XCV 100 EPQ 240 dash 8 which we saw earlier. Let us look at the speed it has offered.

(Refer Slide Time: 23:49)

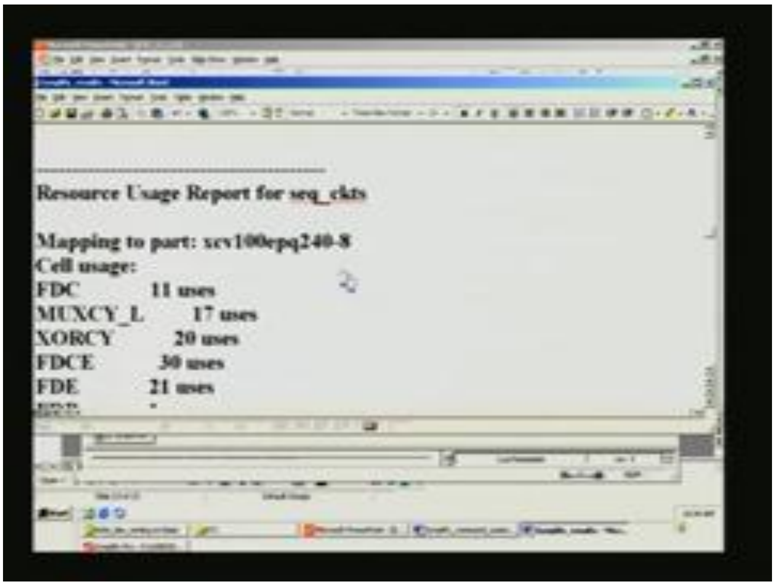


```
               Requested    Estimated    Requested    Estimated
Clock
Starting Clock    Frequency    Frequency    Period       Period
Slack      Type
--------------------------------------------------------------------------
------
clk            1.0 MHz      116.1 MHz    1000.000     8.611        991.389
inferred
System           1.0 MHz      119.3 MHz    1000.000     8.382        991.618
system
==========================================================================
```

This is not of concern, it is not correct way of reporting so you can just ignore. There is no clock and still reporting this. We had only one clock in sequencial circuits I think that is the clock, so what is relevant is this clock. The requested frequency is… what is the frequency here. I think I forget to put the clock frequency there so by default it was 0 there. In the synplify window we are supposed to put 100. I made an entry of 100 only subsequently. Earlier had forgotten it was just 0. Although it was 0, which conveys no meaning. So put it as

1 megahertz but that is not handicap here because whatever is actual thing achieved only will be reported here that is what counts. In fact you would have normally asked for 100 megahertz surprisingly it has given 116 megahertz for this sequential circuit whereas we are around 86 megahertz for combinational circuits. We cannot say which is superior, it all depends upon the circumstances and ignore this here and these are all nothing but estimated time and so what matters is this.

(Refer Slide Time: 25:23)



```
-------------------------------------------
Resource Usage Report for seq_ckts

Mapping to part: xcv100epq240-8
Cell usage:
FDC          11 uses
MUXCY_L        17 uses
XORCY        20 uses
FDCE         30 uses
FDE          21 uses
```

The actual usage of different cells, primitive cells are listed here. D flip-flop is used 11 times. 2 input MUXCY 17, exclusive OR gates and other types of flip-flops. Flip-flops come under various flavours either enable or reset and so on. That is why different flip-flop have been listed here. When you want to know the gate count of your design you will have to look and take everything in to account including the number of look up tables and I buffer and output buffer and so on. You can see the number of LUTs taken is 77. This does not mean 2 inputs or 4 inputs, there are an assortment once again. XCV 100 e this vertex serious, dash 8 speed. Now we will have look at very same results for another device in same type. The only difference is dash 6. Series is same 100000 pack and 100000 gates; number of pins are also same, except that speed is different. Remember earlier it reported 160 megahertz that is for 248.

Now it has climbed down to 100 megahertz. That is the difference between dash 8 dash 6. Based on this you can make a selection. There will be only little choice 6 7 8 are available in that particular type. This will be probably the same FDC 11 17 20 or something it is reporting

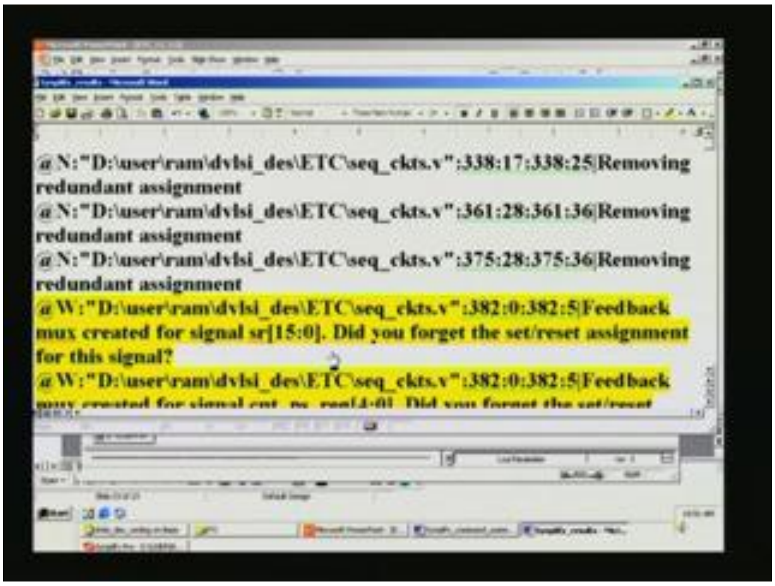
11 17 20 30 exactly same as this. You can have a look at LUT 77 here. It has consumed only 3%. With this we have a made comparison in same series but 2 different speeds. We want to compare this one with Altera whose capacity is more or less same so this is 100 ARC Dash 2. This case you got a lower speed, I may not be fully justified in making 1 2 1 comparison without really seeing the actual speed the vendor has claimed. The best way is to look into the vendor specification and see what their claim is and make a comparison and then choose the right device. I think I had taken the high speed available in this category.

Once again you can see 240 pin package same PQ is not available in this. Whatever possible combination I have selected. This is another series of 10k popular series in Altera just like vertex. 100 again both have more or less same meaning. They also have ram etc it is over above 100000 gates. I think I am fairly justified in making comparison between 2 vendors on equitable platform. Here you can see LUTs and total, if it is 100,000 gates total number of LUTs that you can have is 4092 out of which only 2 % has been consumed. Here in the other case it was 77 or something and here it has taken little more.

Likewise some more logic cells also have been used. There is number of units some register bits are also used here. Some they call EABs I do not remember embedded blocks or something I am not sure. I have looked the vendor specification, some cells how many logic and arithmetic cells consumed so on. It will be very difficult to make a one to one comparison between 1 vendor to another. It is not that plane or simple, technology they have adopted might be slightly different. So it is fair to jump to conclusion that this is superior to other one and so on. Both are in fact in the market so that approves that both are going to be there and both are popular. Now you can ask if you have any question. If I met the target frequency then the design has been successful, isn’t it? Apart from the logic worked out properly, if frequency target has been met, my design is complete isn’t it? The provided functionality is also ok. Assuming functionality is also ok. If there is any other parameter I need to watch out for. In fact you have to take it through the next tool place route, these are all only as I mentioned rough estimate.

Naturally Xilinx place route would not have conveyed to this synplify people, all their proprietary information. That is specific for their own product. Delay reporter in synthesis tool may not be exact, it may not refract the interconnection delays etc. In addition to this primitive cell delays reflecting the actual thing based upon their own technology, their experience so naturally you cannot get it on third party vendor like synplify.

(Refer Slide Time: 32:13)

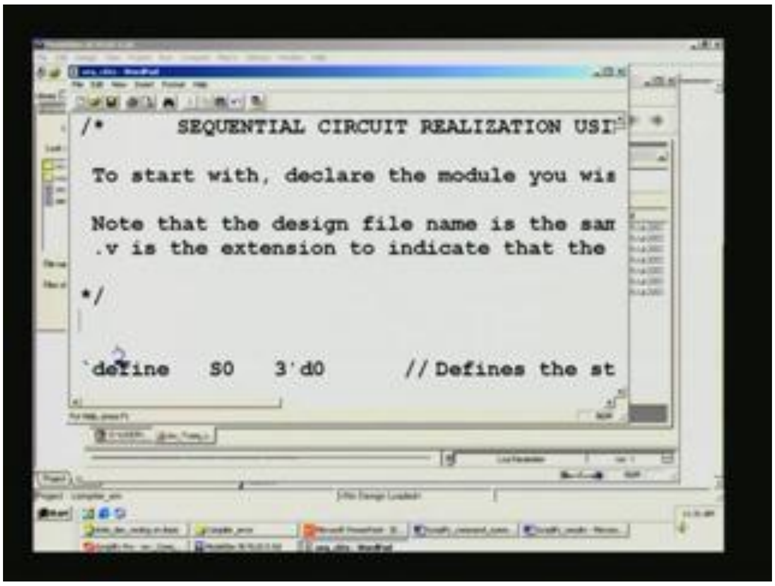


```
@N:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":338:17:338:25|Removing
redundant assignment
@N:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":361:28:361:36|Removing
redundant assignment
@N:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":375:28:375:36|Removing
redundant assignment
@W:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":382:0:382:5|Feedback
mux created for signal sr[15:0]. Did you forget the set/reset assignment
for this signal?
@W:"D:\user\ram\dvlsi_des\ETC\seq_ckts.v":382:0:382:5|Feedback
mux created for signal cnt_ns_reg[4:0]. Did you forget the set/reset
```

This gives us an edf file as an output which you can input to the space route in order to carry out the space route which you will be seeing later on. This is as far as synthesis tool is concerned. We will go on for creating deliberately some errors both in synplify as well as in modelsim and try to learn from that experience. It will not be an exhaustive treatment as such what is normally committed we will have a look, based on that you can do it all by yourself. The only thing you need to remember is create an error in already working file for example let us say this codes are available to you on cd. I am not sure whether you will be getting the codes but in case you have, the best thing is to take disc make a copy in any of your folders. When you want to experiment, you make another folder and put another copy of this source file where you want to experiment.

Whatever you do with that particular duplicate copy, if something goes wrong so you can always copy from the original file and revert back to where you were, if you get lost. In the course of making errors you may over indulge and keep on giving many errors and get drowned ultimately. If you get drowned so easiest way to come is copy back the working code and start from where you left. So you stick on to making one error at a time, learn about that error then rectify that error then create a fresh error. In this fashion, this very same file which is a duplicate copy which you can destroy, it should not matter really.

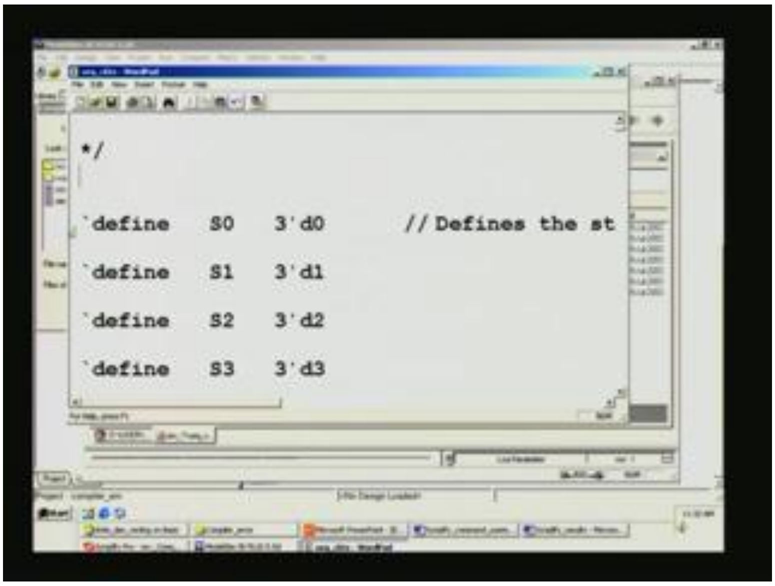
(Refer Slide Time: 34:24)



```
/*      SEQUENTIAL CIRCUIT REALIZATION USI

 To start with, declare the module you wis

 Note that the design file name is the sam
 .v is the extension to indicate that the

*/


`define   S0   3'd0      // Defines the st
```

With that it will be a quick way of learning things. We will deliberate commit some errors during compilation; let us see whether it will report any errors. Even during loading a design file, errors may be reported. We will first try out one after another. As I said before we will create only 1 error at a time even that will make a copy of design files in some temporary directory then make these experiments. This is like embarking on dangerous expeditions as such. So if you do some grave mistakes, entire file will have to be thrown. You be on your guard so preserve your actual file which is really working elsewhere and make the experiment in temporary register. You can start designate it as a complier error folder, copy design file in that then make the experiment. So to start with while experimenting, we will see both the platforms such as synthesis using synplify as well as modelsim for simulator.

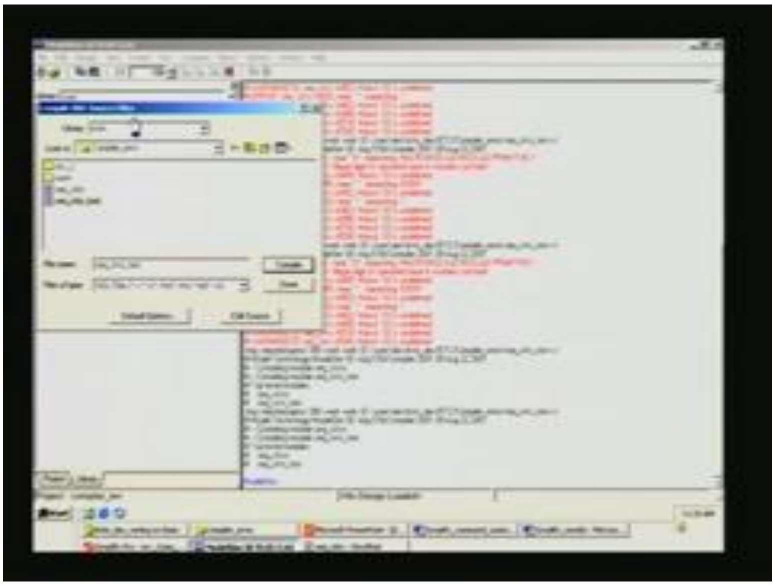
(Refer Slide Time: 35:47)



```
*/

`define   S0   3'd0      // Defines the st
`define   S1   3'd1
`define   S2   3'd2
`define   S3   3'd3
```

Let us take one error at a time. The error that I am going to commit is what I have observed over the years not only myself committing these errors but also other engineers or students committing similar types. One such error is my very first mistake when I started a learning verilog several years ago. Instead of this I put this. So let me commit this error. Let us see what it does. Just remember you have made an error here, just save the file only then it gets reflected there.

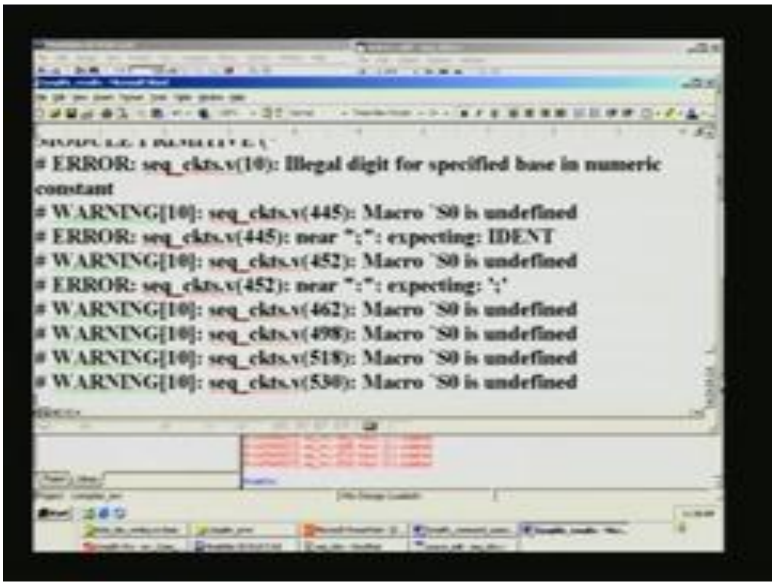
(Refer Slide Time: 36:39)



Let us see this modelsim is opened here. As I mentioned it is in compiler error folder. This was in that ATC. The actual files were safe in the ATC folder; whereas, I have copied only those two here in compiler error. That is what you see here. Another specific point I need to mention is, I will be running test bench here because I have included the design file in that test bench. When I compile this automatically it takes this design as well and compiles. That is the difference I am going to make between running the modelsim as well as synplify. In synplify test bench has no role to play. I will straight away running the sequential circuits there and therefore the error I am going to commit is only on these sequential circuits. I think I will agree upon that. I have already made the error as I mentioned here. I will double click this compile and then let us see what error it reports. I will just copy this modelsim. In modelsim let us see what it is reporting. Here it says illegal digit for specified base in numeric constant, it means this is not plane from here. Let us go to the modelsim once again. You do not require this for time being or just go to this, remove that. This is the error so if I double click it might open the source code. Let us see whether it does, yes its opening. It is pointing to the define statement which we made the error. This is one here that is precisely the error that we made.

If I click on next one I think it does not report illegal digit for specified base so it is only a consequence of that. It does not say anything further I think it is pointing to the same. I clicked on the next, it says warning and macro s0 is undefined. It is pointing to where that s0 was used in the source code. This is only as a result of defined so earlier what we had is this macro S0 is undefined. One more error also it is pointing out. Let us find out what it is. There is one more that is warning and error point to the same. There is one more warning and followed by again same S0 state, because that define pertains to S0. That is why very first statement was not recognized everywhere same occurrence is reported as error or warning. You can see here also its pointing to same. Again some more warnings here, always same S0 is being pointed out. You can see different lines, it reports which line number, its 530 here, same S0 is appearing here.

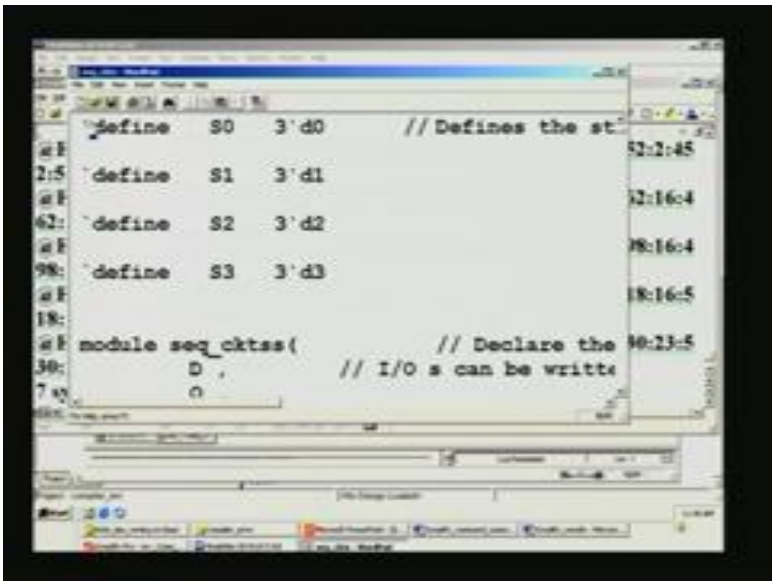
(Refer Slide Time: 41:18)



```
# ERROR: seq_ckts.v(10): Illegal digit for specified base in numeric
constant
# WARNING[10]: seq_ckts.v(445): Macro `S0 is undefined
# ERROR: seq_ckts.v(445): near ":": expecting: IDENT
# WARNING[10]: seq_ckts.v(452): Macro `S0 is undefined
# ERROR: seq_ckts.v(452): near ":": expecting: ';'
# WARNING[10]: seq_ckts.v(462): Macro `S0 is undefined
# WARNING[10]: seq_ckts.v(498): Macro `S0 is undefined
# WARNING[10]: seq_ckts.v(518): Macro `S0 is undefined
# WARNING[10]: seq_ckts.v(530): Macro `S0 is undefined
```

Let us remove this error that is what you see zoomed here. Before removing let us see what the synplify reports. In synplify also we are in compiler error and we will run here. Remember we are going to run the design file and not the test bench, so that is difference here. Let us see whether errors are encountered. You see view log and find out what the errors are. For examples here I will copy this portion into this here. So this is the modelsim error and synplify error is here. E stands for error, you can see compiler error unknown macro S0 exactly same thing has been reported there. I mistook one symbol for the other because both resembled, in fact have personally spent sleepless nights on this, so be on your guard. It appears so simple to trivial thing but actual point is you may waste lots of time while debugging. We will create another type of error normally made by designers. Let us see how it is done.

(Refer Slide Time: 43:18)

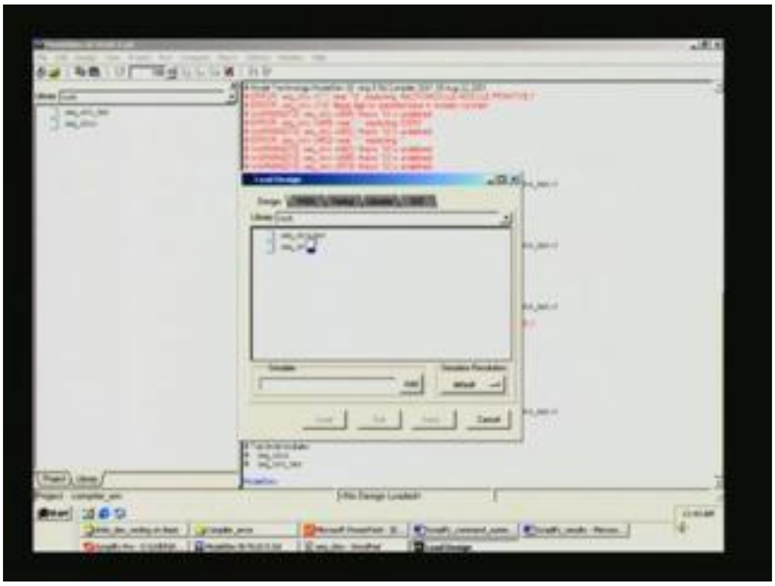


```
`define   S0   3'd0       // Defines the st
`define   S1   3'd1
`define   S2   3'd2
`define   S3   3'd3

module seq_cktss(            // Declare the
          D ,         // I/O s can be writte
          Q
```

Before that, let restore this to the original one, that was the symbol. So this is there at the point where tilda is on the extreme left of the keyboard and next thing is I have already made a mistake here. Module name, I have misspelled sequential circuits by a mistake I put to. Many people have done this also. Especially those who are not strong in English off course here I do not need; those who do not take care and they make a mistake like this. Let us see what sort of error it gives. So we have rectified this. There are no other errors, so we are right here on this module name. First let us see the synplify, we do not need a report, it says errors. Let us see what it says here. If you double click here it shows the same defined with previous mark although we had corrected, because I think this file was open at the background because I had another file here. It did not get reflected there, although I made the change here. Now the errors are disappeared.

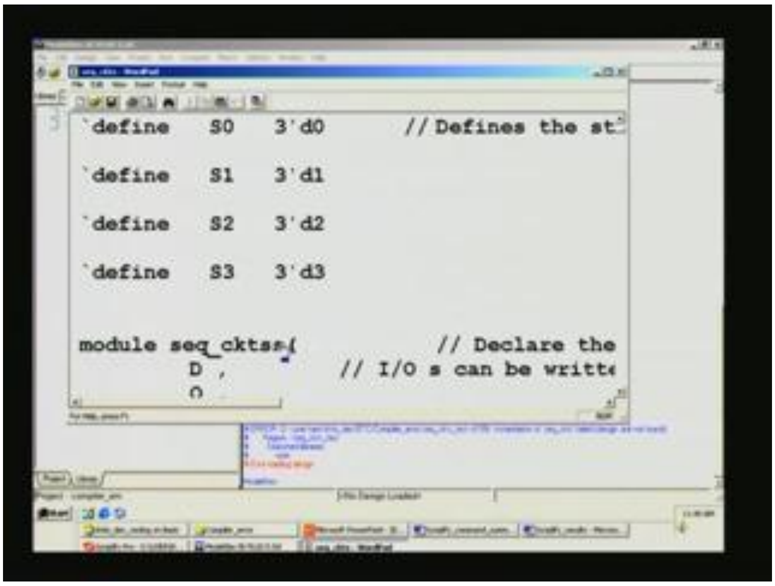
Same source file was opened in this editor, so when synplify runs it looks for only its own source file. If you had made any changes outside it does not get reflected; you see there are no errors here. But we had committed an error there. Let us go to modelsim and find out. In modelsim if I say compile this is the test bench which we are going to compile. Let us see what it does. I have compiled a modelsim, fortunately or unfortunately error reported here. This is a dangerous situation, you cannot take it for granted. If you take it for granted, later on you will be surprised to see lots of functionalities are lost or the entire thing is lost. Let us see whether it reports somewhere else. Once it is compiled successfully, design should be loaded.

(Refer Slide Time: 46:42)



Let us see whether this is okay or not. We had to load this same test bench only, so will load this. Let us see what happens here, the error is caught here. So the thing which went unnoticed during compilation was caught while loading the design. Are you clear how deliberately you can create errors then try to solve it. You can just click on this, sometimes it does not open the source code because it is a load file, it does not open this source code. So I will just read it out for you. Instantaneous of sequential circuit failed because test bench makes it as sequential circuits with only 1 S whereas within the design I deliberately change it to 2 S. Naturally, this being test bench because what we are loading is test bench so here it is okay. It is unable to find that a particular design because we have deliberately changed. You can see that during compilation you will get errors, even if it is free of errors do not take it for granted. Only at the time of loading design you may find more. We will see 1 or 2 more before we wind up.

(Refer Slide Time: 48:07)



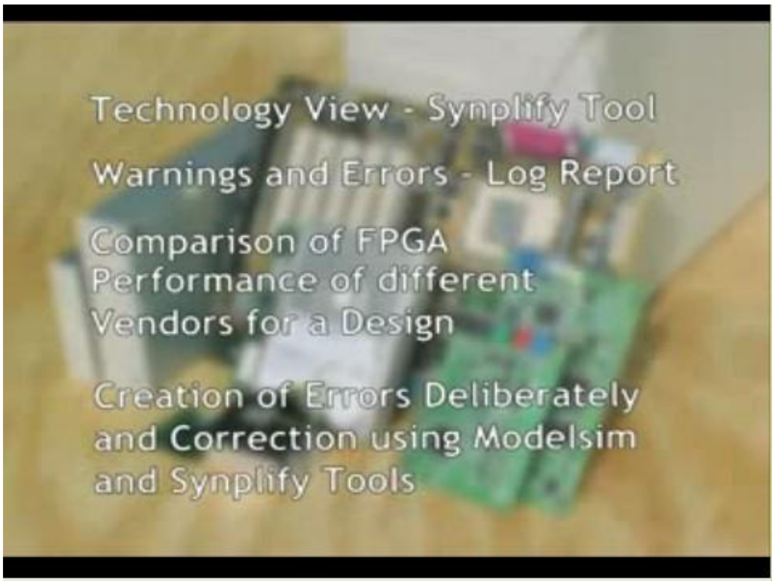
```
`define   S0   3'd0       // Defines the st
`define   S1   3'd1
`define   S2   3'd2
`define   S3   3'd3

module seq_cktss(              // Declare the
        D ,              // I/O s can be writt
        Q
```

This we can remove and we have removed the pervious error too. For example I said you should not use comma. I have removed comma, that error was also there but it went unreported. You should be on the guard probably some functionality got affected, you have to look in to the systematic diagrams and find out whether the functionality is or otherwise you take this dot vm file once again generated and look in to the simulator to make sure functionality is intact. I leave this as an exercise to you. Just remove the comma for normal functioning. If you inject a comma then of course I have commented it out that means to say I have removed the comma its equivalent to that. So what I will do is I will put a comma then we will see what it is; in fact there are no errors here.

Let us see whether during load, error is reported. It says 2 few port connections. If you click on this normally the source file must be opened if it is a relevant message. Two few port connections you will have to find out why there are two ports because we have to put a comma there it was expecting more ports to be listed. I think it is reporting 2 few port connections here and ports size does not match connection size. It does not open the 15th connection. Let see the source file if you take this one, if you just count 2 4 6 8 10 11 12 13 14, it does not throw light. So you will have to investigate it separately which would be a time consuming affair. So you have to cautiously go. The best way is, remember to put comma for all the ports except for the last. We will continue with creation of errors and experimenting the same in our next lecture. Thank you.

Summary of Lecture 32:

(Refer Slide Time: 51:24)



Technology View - Synplify Tool

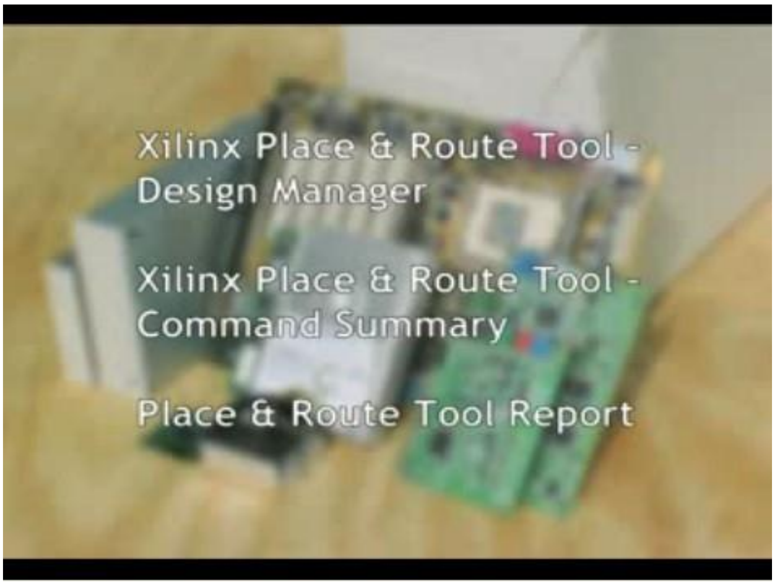
Warnings and Errors - Log Report

Comparison of FPGA Performance of different Vendors for a Design

Creation of Errors Deliberately and Correction using Modelsim and Synplify Tools

Next lecture: Synopsys full and parallel cases.

(Refer Slide Time: 51:24)



Xilinx Place & Route Tool - Design Manager

Xilinx Place & Route Tool - Command Summary

Place & Route Tool Report