Digital Systems Design with PLDs and FPGAs

Kuruvilla Varghese
Department of Electronic Systems Engineering
Indian Institute of Science – Bangalore

Lecture-44
Case study on FPGA Board

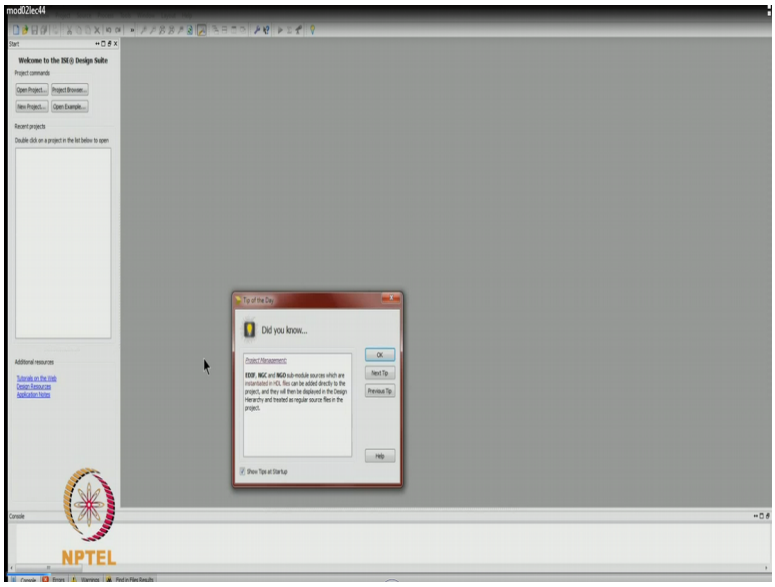
So, welcome to this lecture on digital system design with PLDs and FPGA. The last lecture we have seen a case study what we have seen is that we have seen the design of an 8 bit unsigned integer multiplier. I said you can extend the design so we have seen the algorithm, we have modified the basic algorithm, we have seen resources, we have seen the datapath design at the top level.

Then each blog we have expanded and we are also parallely seen the VHDL code for it. We have design the state diagram. Then we have put together everything together in a VHDL code and some kind of you know concise way of writing the VHDL code and all that today. And that has you know kind of winding up everything in the course you can extend that and today I want to show you the FPGA tool and the same exercise on FPGA kit or FPGA board on a spot and 6 board.

But at the beginning I will show you the Xilinx Is or ISC integrated know system environment where we can do everything you know we can do synthesis place and out simulation the programming of the FPGA, timing simulation, timing analysis all that. I want to straight away go to the tool and show you some simple example 1 combinational and 1 sequential then we will go to the real design wherein I will just I will not simulate it.

Because it is combusum to simulate it. I just show you the program on the board. So, let us go to this particular tool, so when you take when you install this program on the computer.

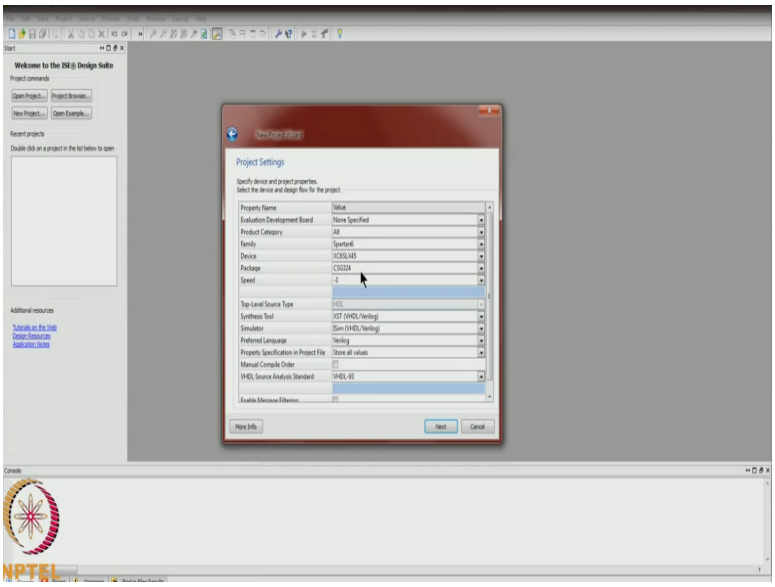
(Refer Slide Time: 02:32)



And you will have some you know the Xilinx ISC designs suite 14.7 click on it, it is available in the in the menu also I will minimize it a bit ok, os that I can see it little bit full screen ok I will make it resize it, so that it is visible ok. So this is the tool and the latest tool from the Xilinx is called the reward which is only used for the 7 series, what at 7 series devices using some older devices we will use this particular Xilinx is and see this is area where all the file and the related process will be shown.

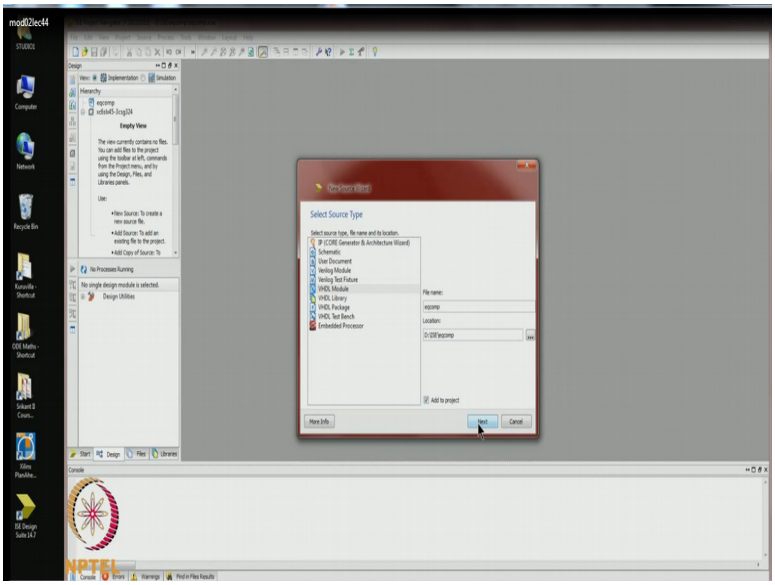
This is the area where all the open you know files, waveforms, statistics everything will be shown and this area is a control where all the messages you know the process messages whereas everything comes okay. So first thing is to start a project okay every problem is associated with the project. So we will open a project a new project. So let us you choose a directly to put that and I will choose some predefined directly.

(Refer Slide Time: 04:13)



So let us take an very simple example the equality comparator we have developed. So let us college EQ I have this was the first text size we have looked at the VHDL so this is the name this is the location and all that and you say next and here a very important is to choose the device family device name and the package that you to choose very properly and this is a synthesis tool stimulator, what is a Preferred language and all that.

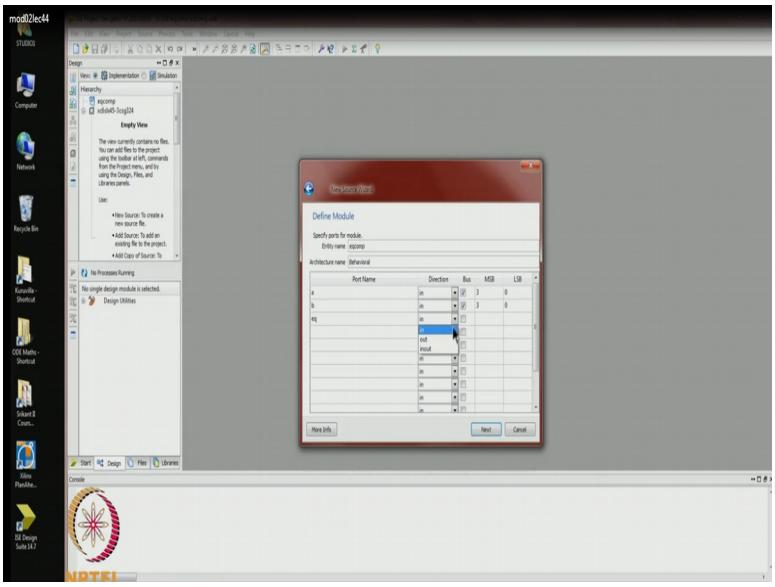
(Refer Slide Time: 04:46)



And all that can be chosen you say next and you say finish now you can see that this is the project area where all the all the files will be shown now depending on what it to see what all you can do the processes will be shown here ok so first thing to create a VHDL program and there are two ways of doing it if you already have it you can right click and add source or you can create one say let us create one.

Now let say that a simple 1 let us create, say new source and it opens up a menu for you and there are different files you can create like test bench, VHDL design, Verilog design and all that. So let us take the VHDL module give a name and this is the directory, give a name for the project, let us call EQ com again ok and say that is the file name of the VHDL ok. We say had to project should be next.

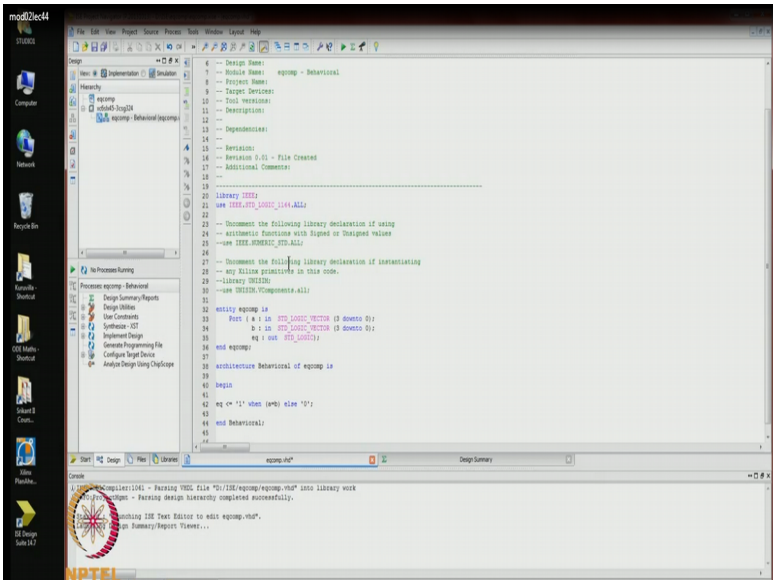
(Refer Slide Time: 05:54)



And now this gives a template instead of creating your file from you know from the beginning you can created for the template, so it is already giving the NDT in a eqcom architecture name of behavioral and it allows you to kind of generator ports. So let us called this port as A ok and that is a bus which is you know if you remember this a 3 down to 0 ok it is 3 down 0.

Similarly we create a b which is input mode and this is also a 3 down 0 ok and so we have an output which is call let us call eq which is you select it is an out and it is a single bit, so do not do anything that is it.

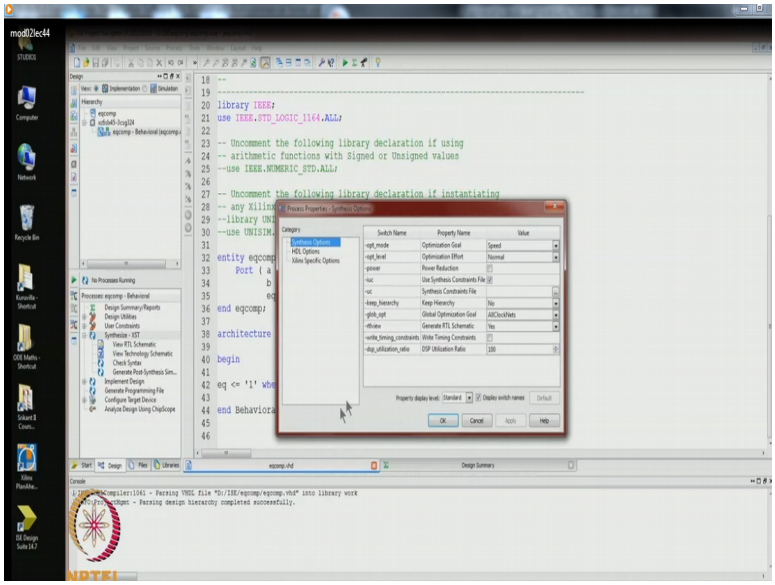
(Refer Slide Time: 06:49)



And see now you see that it has created for you a template call library and use declaration the entity with all the ports, a b is for which factors input port equal is up, architecture is written here with the begin, it is like here the code eq get shift 1 when a=b ok else 0 ok. So that is how ok that is the code ok, that is what is written here ok and now maybe I hope you can see you soon then if this is visible ok.

So I hope you can see that now so this is you see the entity architecture we say eq which is the output port get 1 when a=b maybe you can give some space for read it. Now you say ok to say, now you select that the file now you see we have two options 1 is implementation one is simulation. When you do implementation it will show the synthesis place and route all that you know the programming and all that.

(Refer Slide Time: 08:39)

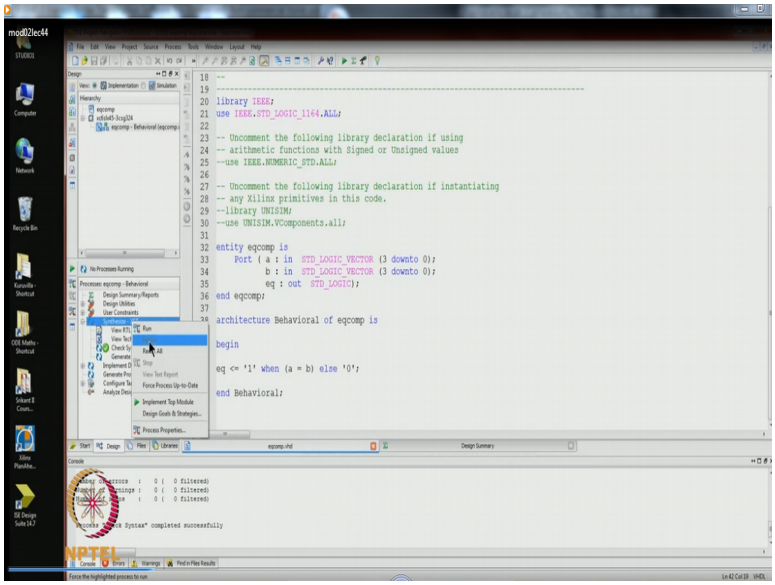


So here you see the moment I have selected that you know there are various you know you can you can synthesise you can implement and all that. Let us click on the synthesize it then there are lot of properties which we can go through there are some synthesis option, HTL option, Xilinx specific options, there are lot of options you have we can you can choose it weather you optimise for area, optimise for Speed or you can kind of Saturn in forensic automatically with the muxes are to be infer.

All that can be chosen to the first thing to do is that you have to check the syntax ok. So let us introduce an error saying like this I will write like this that is equal and I will see that and I say checks you can see that already it is lagging an error which say that a Syntax error near equal and I just click on the light and shows what is the place where it is error, so I just delete that and save it. I automatically checks and it say that it is the passing is completed.

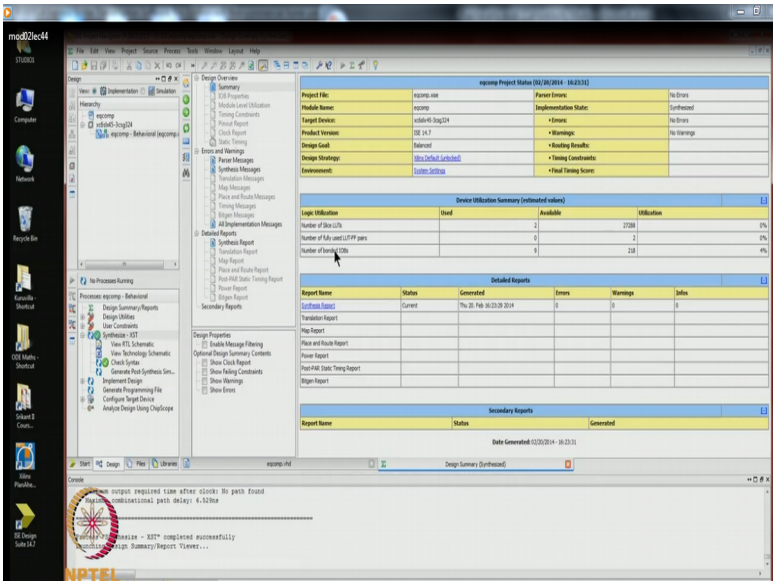
So if you want to double check you can say syntax use a run so it takes a while to do that and it will show lot of messages ok. Now that after checking the syntax when it is completed it is + sign you can you run for synthesise, so it is synthesise the following code ok. That is what it is going to do and while it is doing it will show messages here it is already showing a delay of 6.55 nanoseconds.

(Refer Slide Time: 10:41)



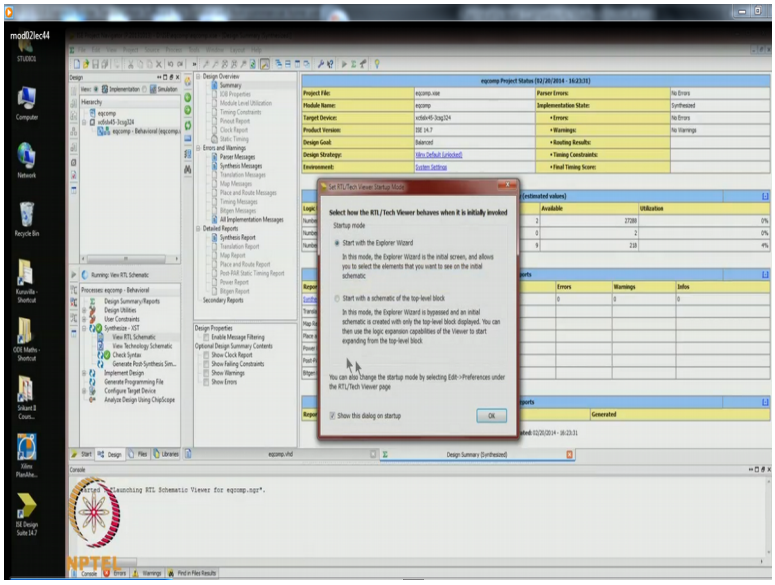
While synthesizing itself it knows that what are the resources used in FPGA, so it estimate the delay and insecure ok. So here we can do is that you can say in the in the process if you click this Sigma it shows the design summary okay.

(Refer Slide Time: 10:51)



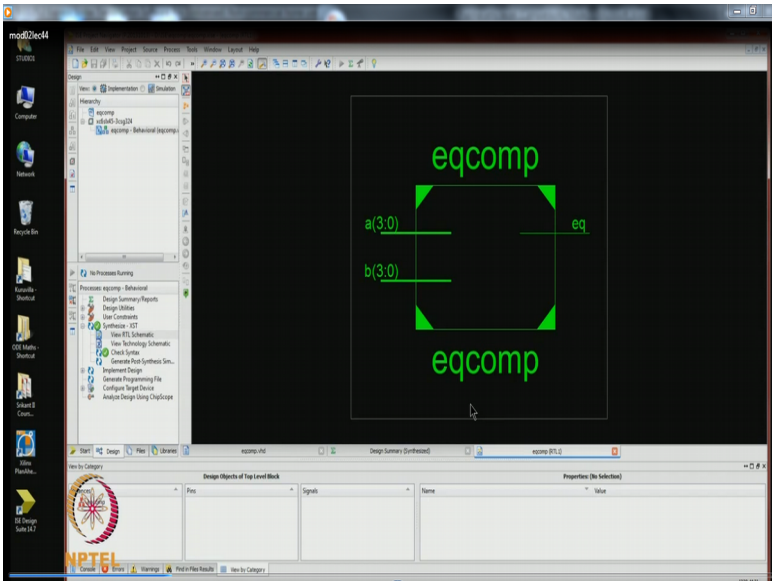
It shows that number of lookup table use are two and the number of IO pins is used or 9 because we have a 4 bit equality comparator will be 2 look up tables used we will see that and the number of pins because there are four two 4 input inputs and 1 1 bit output that makes a 9 and 1 way of seeing what is synthesized is viewing this RTL schematic.

(Refer Slide Time: 11:18)



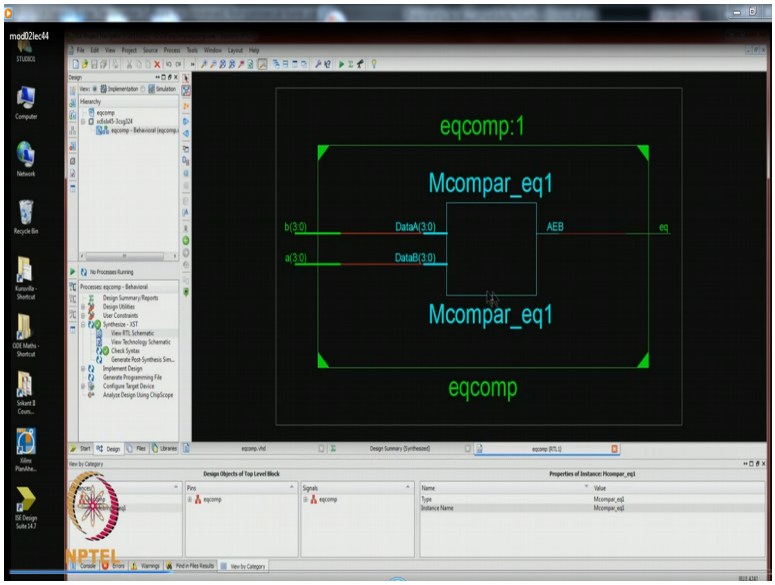
So if you can you say start with top level block you can say ok.

(Refer Slide Time: 11:25)



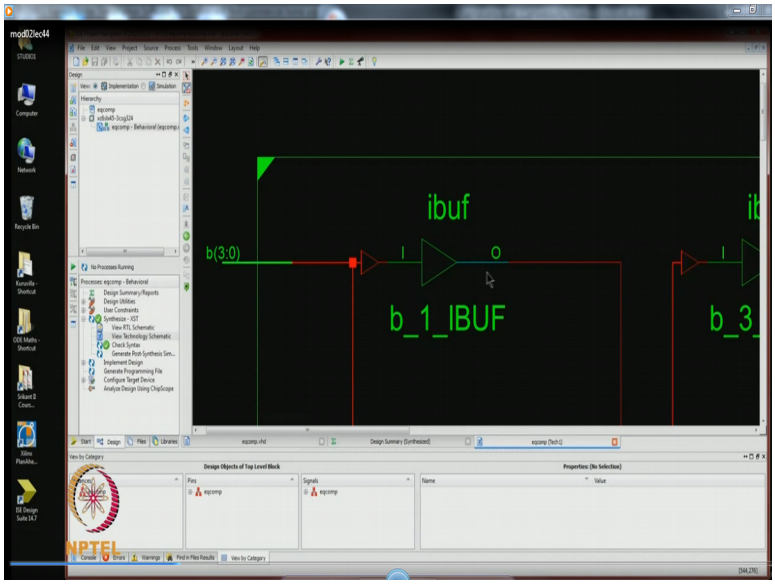
And that is a schematic of what is infer.

(Refer Slide Time: 11:29)



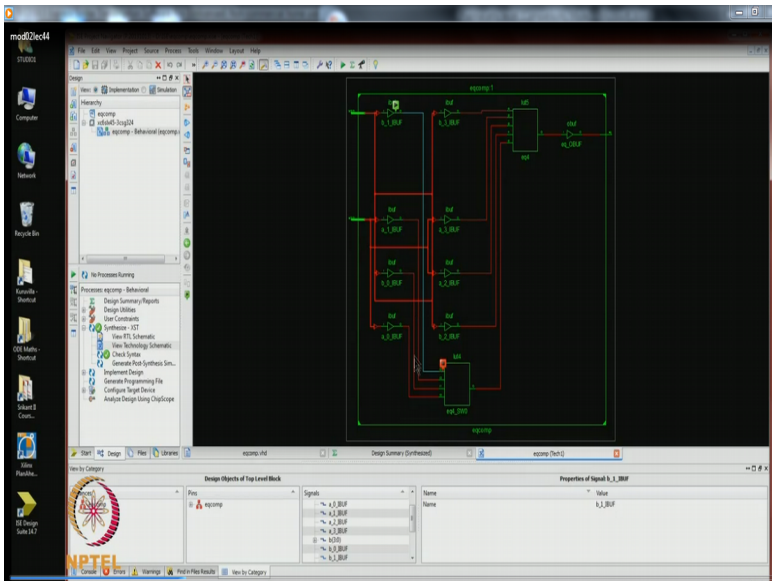
And you can just so it is detected as a comparator so it is showing say a 3 down to 0, b 3 down 0 is an equal operator and equal comes out because it is a nice so it is it is showing as blocked ok. So another thing is that you can check the technology schematic where did you show how it is implemented within FPAG ok.

(Refer Slide Time: 12:00)



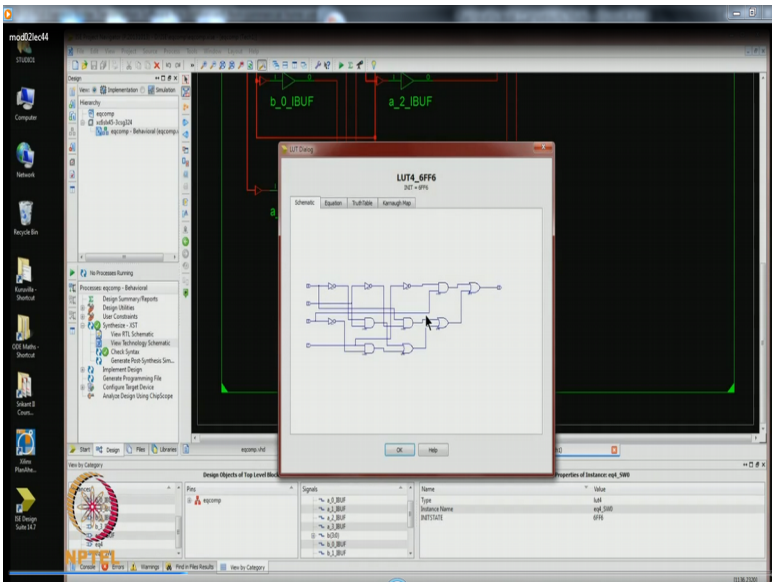
So you go here and you say zoom to the thing and you can see that are these all are input buffers like a this is signal like that see what is a signal there.

(Refer Slide Time: 12:18)



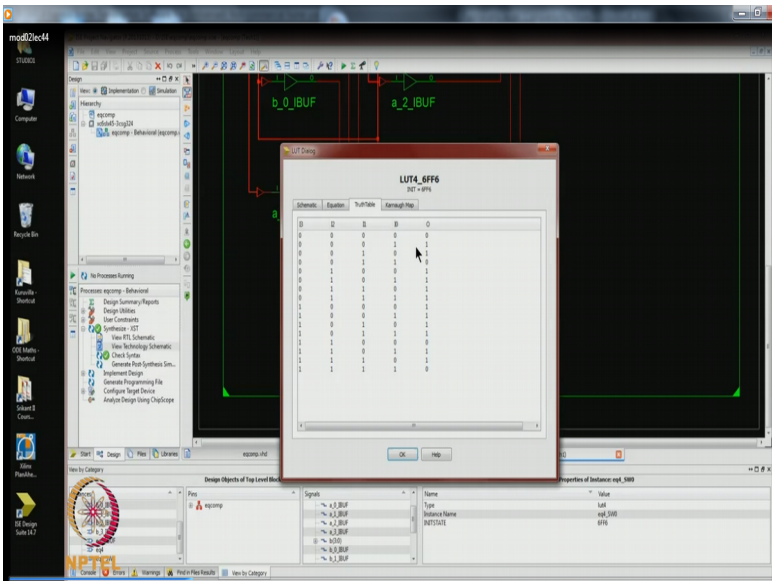
That is a and this is be so this is b, so there are this is a buffered and given to a lookup table b is buffered and given to a lookup table, we can zoom in on that you know you can just.

(Refer Slide Time: 12:35)



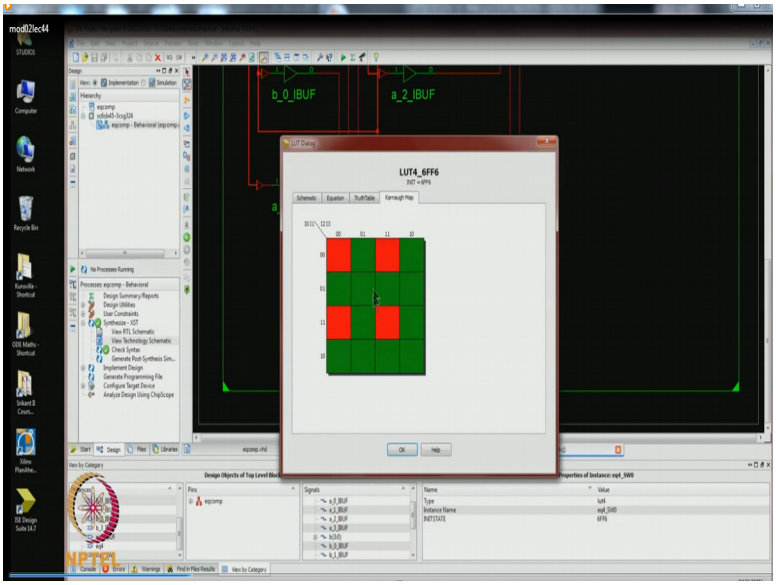
So this is the lookup table and click on that and you see you can see the internal the secretary and if you see the equation which is comparison.

(Refer Slide Time: 12:53)



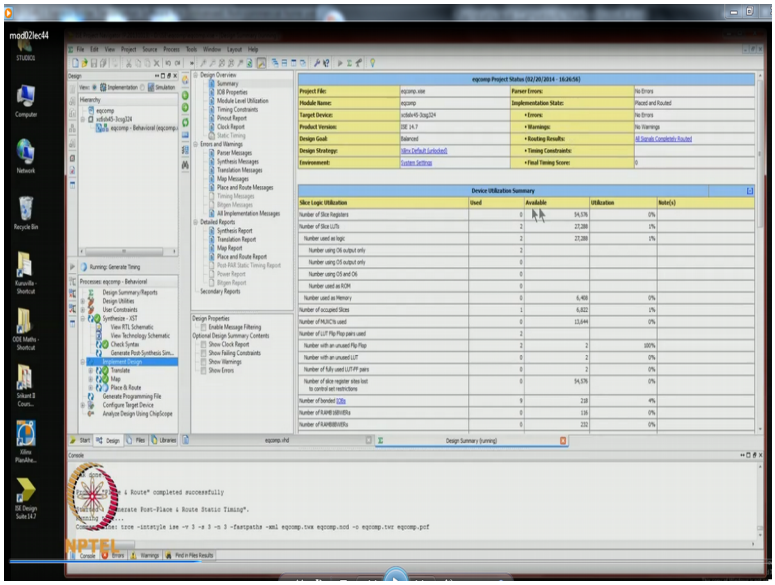
And you can see the truth table say when it is you see it is 1 when it is you know when we can analyse that depends a kind of map of the optimization.

(Refer Slide Time: 13:01)



All the detail can be seen it's very useful or learning exercise also so won't have done that you know you have done this in this week and then you can go for implement the design so you just say Run do something all played map and the place and route ok. So it takes a while lot of lot of things are done and that messages come here in the console and mind you this utilisation everything is updated in a quite lot.

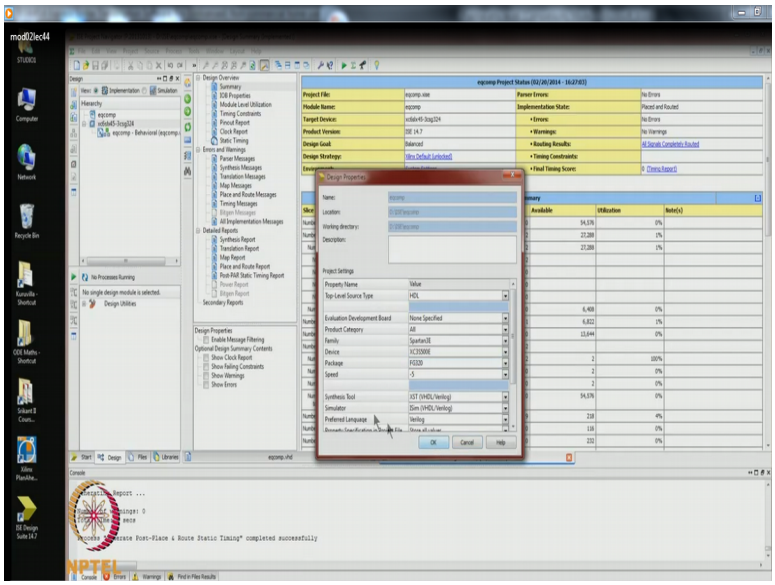
(Refer Slide Time: 13:49)



So you can watch that what is going on here. So it is done now all that is done any can see very elaborate statistics or written here, so there is nothing much because the symbol design a register 0, there we are not using any flip flop look up to you please to all that you know there is nothing much other than that is available here, lot of resources not like PLL, DSP blogs are nothing is used.

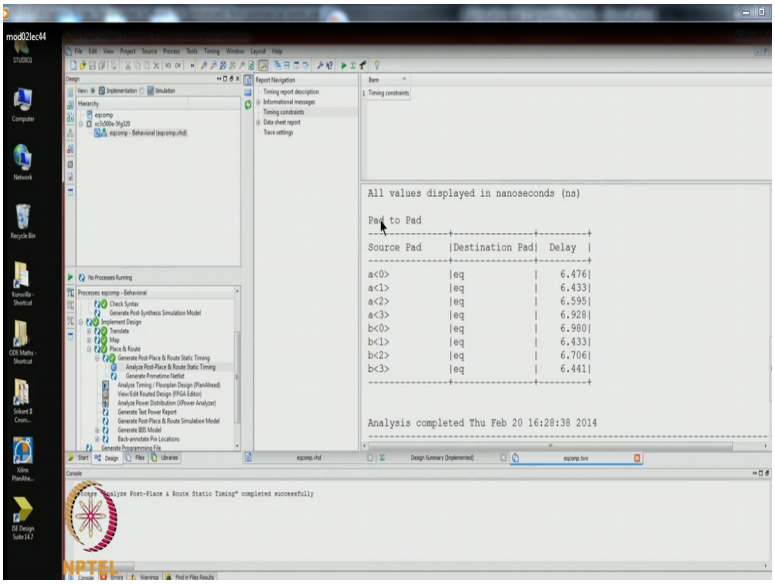
We have very simple example and I have chosen a ok this is a Spartan 6 device which I have closed and maybe I can change it to maybe Spartan 3 ok that because I will change it in free let me see Spartan 3e because that is similar to the Virtex we discussed in the class.

(Refer Slide Time: 14:56)



So that we get some when we see inside we will have some familiarity with the lookup table and all that. So I have to say now the Spartan 3e which is similar to the Virtex we have learnt in the class, so early what is default was Spartan 6 I will change that, so that we can when we see the resources, when we see look into the router design we need compare, your comfortable ok.

(Refer Slide Time: 15:56)

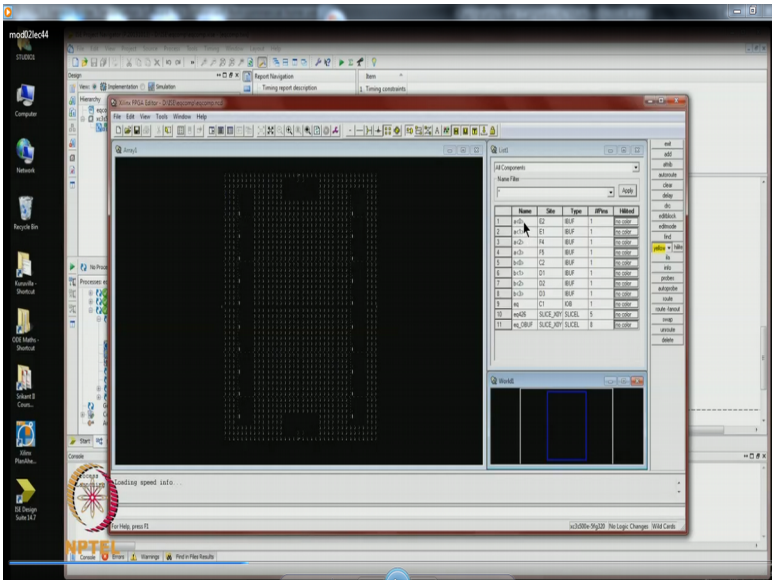


That is done the first thing to do if we can open this place and route and let us go and see the static timing analysis ok. This shows the analyse port space, route static timing. So click on it

and you can see this one ok maybe I will magnified so you can see that it is reporting the delay say source path it IAS a pad to pad is a combination circuits so input and output path so a02 equal it is 6.476 nanosecond similarly a1 a2 a3 b0, b1, b2 b3 all the parts are chosen.

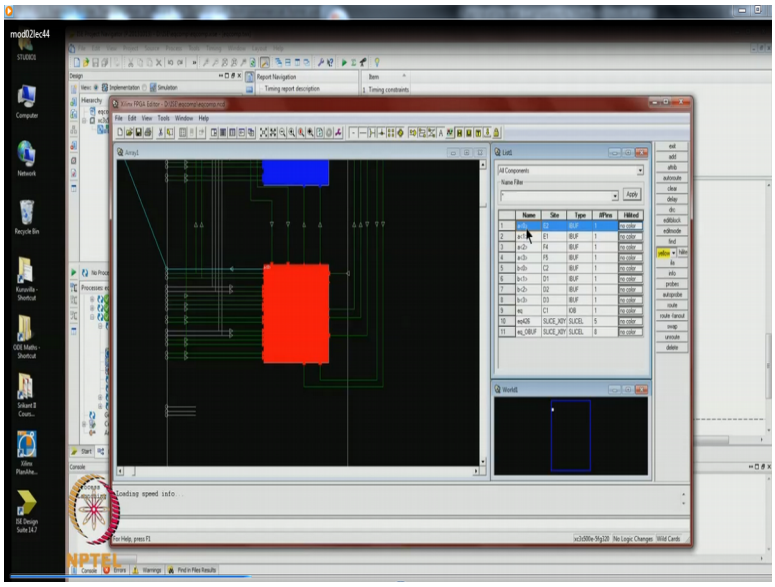
And shown maximum delay reported here is a 6.98 nanoseconds which is somewhat similar to what is reported during the synthesis. So that is quite correct and other useful thing you can do is that you can look at the router design within the FPGA, so you click this particular menu view edit router design.

(Refer Slide Time: 16:46)



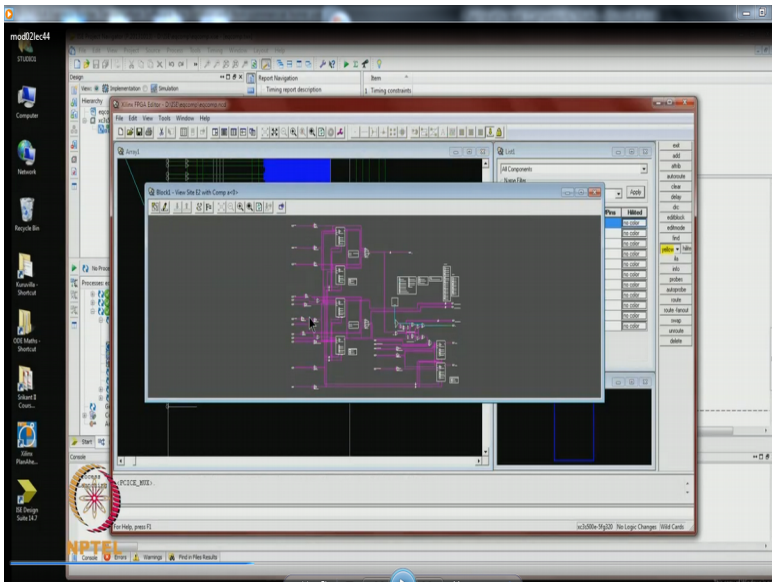
So if you click that ok this window show the whole FPGA ok the whole FPGA with the I/O pins around all the slices and the DSP blogs the memory everything with no special resource, so you see some gaps and all that ok.

(Refer Slide Time: 17:09)



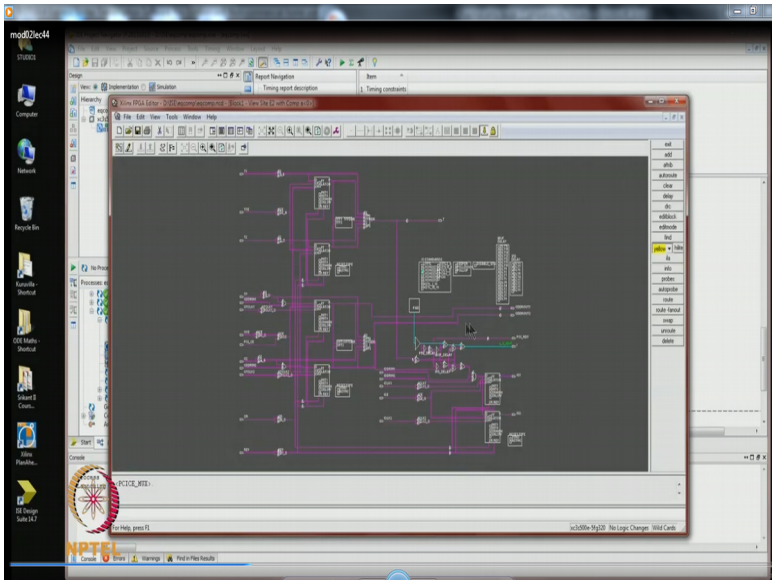
And it shows you know where are the signal suppose I click on that it shows ah where is that I/O pad is coming ok.

(Refer Slide Time: 17:16)



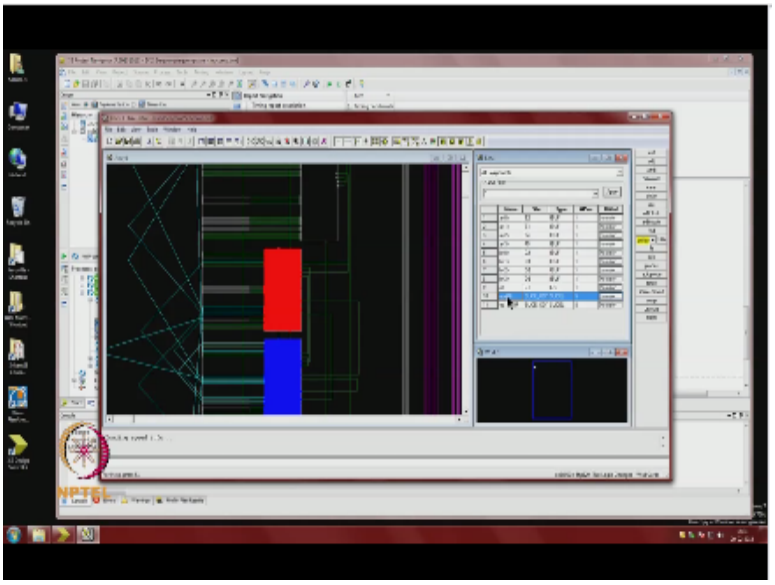
So you click on that that is the picture of the I/O block of this particular device.

(Refer Slide Time: 17:18)



And which is ok, So that shows I/O block ok, and one can zoom there is zoom, so this one shows that I/O block ok, this is the pad, these are buffers and all that, you know input you can input, so input is coming. Now sorry I will open that again ok let us look at this particular, so there are 2 slices you select let me click on that.

(Refer Slide Time: 17:55)



So you can see that this is the table use that say this particular slice is used because we need 2 look up and this is a slice used so you click on that and now you can see that that particular picture we have seen the detail view of the slice. So you see here, so this is the 1 lookup table which is used you can see that is magnified. So can you see b3 and a3, a2 and b2 are going.

So this is a comparison of a2 b2 a3 b3 and you know combining them to this is this implements say a3*orb3 a2*or a3 a2*orb3 and a2*orb2, so that is the logic implemented here

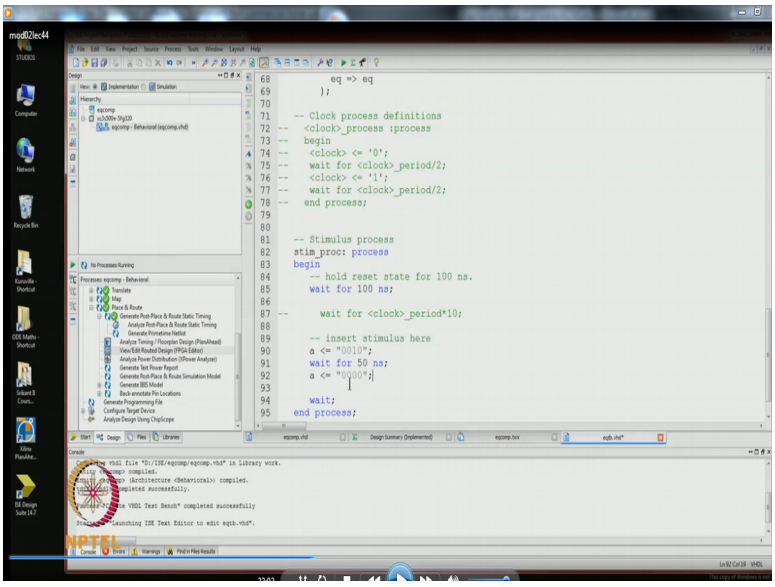
in this particular thing. Similarly can you know this shows you know the various the architecture of the slice and if you remember if you can remember these are the 2 look up tables, 2 flip flops.

This is a carry chain all that can be kind of you know looked at and you can clear your doubt is very useful is not only designing you can see very detailed a picture of what is going on within FPGA. So let us now do a simulation ok now for simulation you can write create a test bench as have shown earlier like maybe can say a new soul ok and you can select a VHDL test bench ok and you say give some names let us call it eqtd test bench will show which is the entity to be assigned.

So it is showing if you come because it is only one entity there are multiple NDT architecture after whichever you are which of what you are trying to simulate mostly the top level component next the finish ok, what good thing is that again this tool will create a template code for you that I will magnified it ok so that is the library function, this is a empty NDT, this is the equality comparator instantiate these are the declaration.

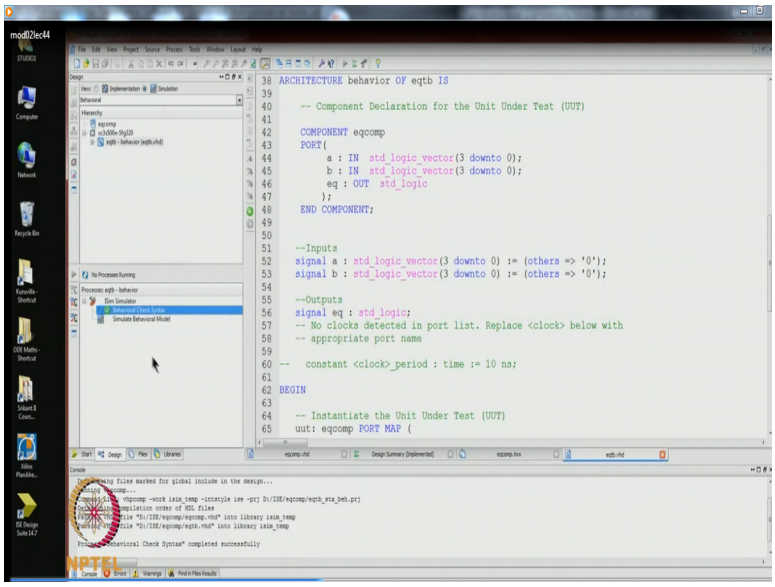
This is the kind of infatuation in also clock which we will come and its which we do not require, so this is a clock process again this is not required. So we will come and its ok and there is a weight 4 clock period that also we will come and it and here we can insert the stimulus ok. So I will just show a very simple thing a gets a 0 0 1 0 ok now both are 0, 0 at the beginning.

(Refer Slide Time: 22:21)



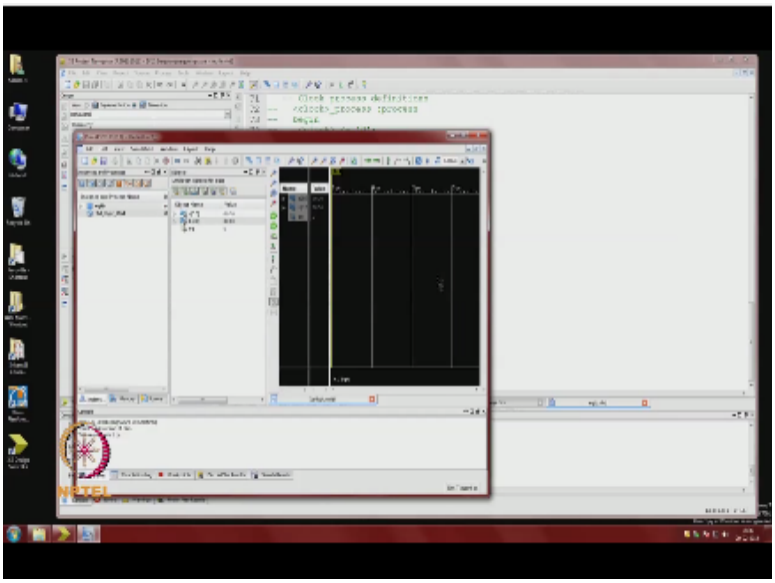
And I will say wait for sometime ok I will say wait for 15 nanosecond and I will change may be a back to 0 again so that you get quality ok and so on ok you can give delay sorry that is not come what happened ok fine ok just save it. Now the test bench is ready now what we do that you have to select the simulation, select this test bench and we will see this is the simulator I symptoms click open.

(Refer Slide Time: 22:25)



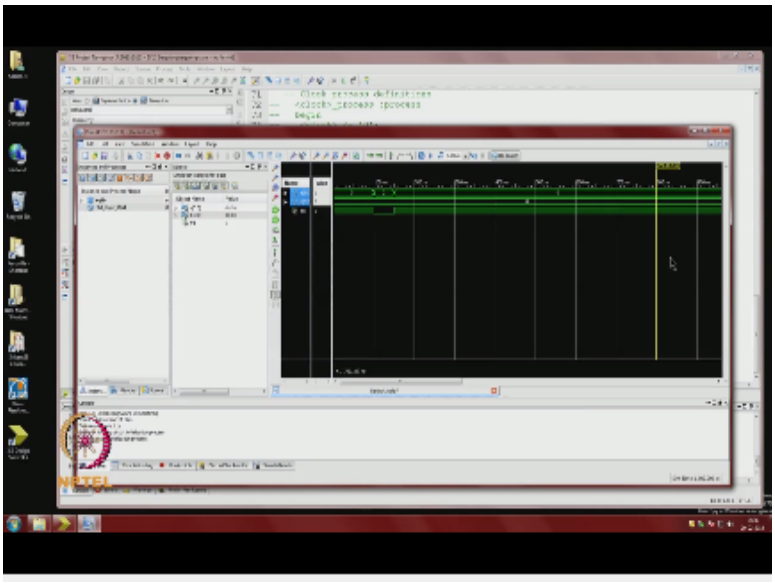
Do a Syntax check now you can right click on this might be able you can see that a lot of property is it in the default it runs only for 1000 nanoseconds you can extend it to 1, but our thing it is ok and also most simulation will have a whole state to 100 nanosecond to stimulate the FPGA programing time, so you assign only the signals input signals after a 100 nanosecond delay.

(Refer Slide Time: 23:07)



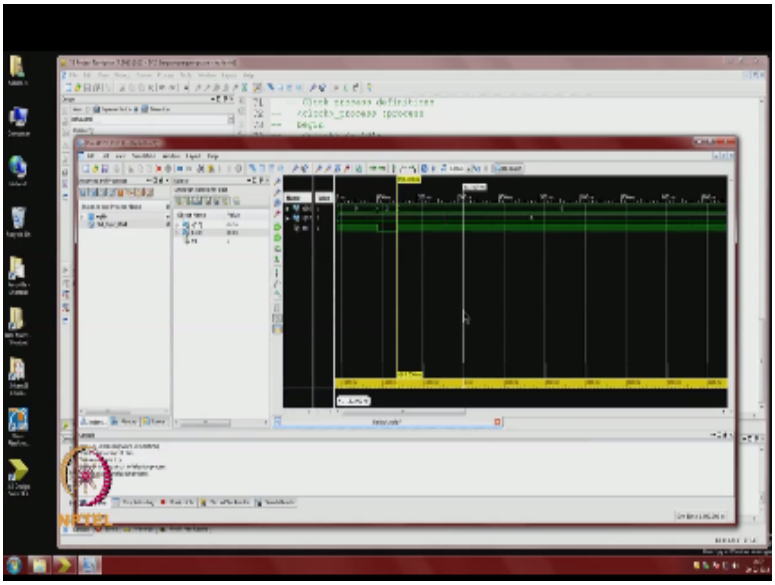
So now you clock similar behaviour model so it shows a waveform window, so that you clock open and you can zoom to this thing and you can see this you can select the radic as unsigned signal, so usually both were 0, 0 and you can zoom, so ok let us zoom in, so here both inputs are 0, so the equality output was 1, that we made something for 2 either 1 of the input is 2, then the equality goes low.

(Refer Slide Time: 23:43)



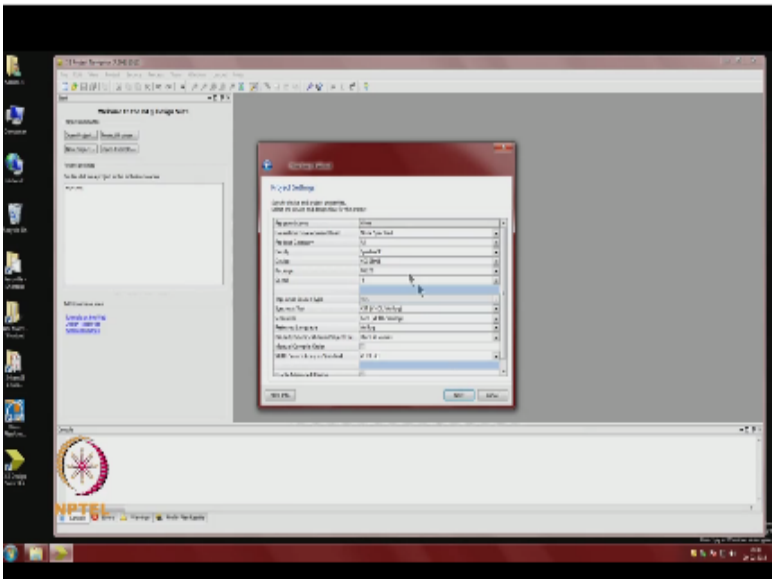
Then again we made it so it goes up ok and this is a marker which shows timing suppose you want to latch this on to the to this edge you click on this latches you want to insert another latches word insert another marker, say you say marker and add marker. So you get another marker.

(Refer Slide Time: 24:01)



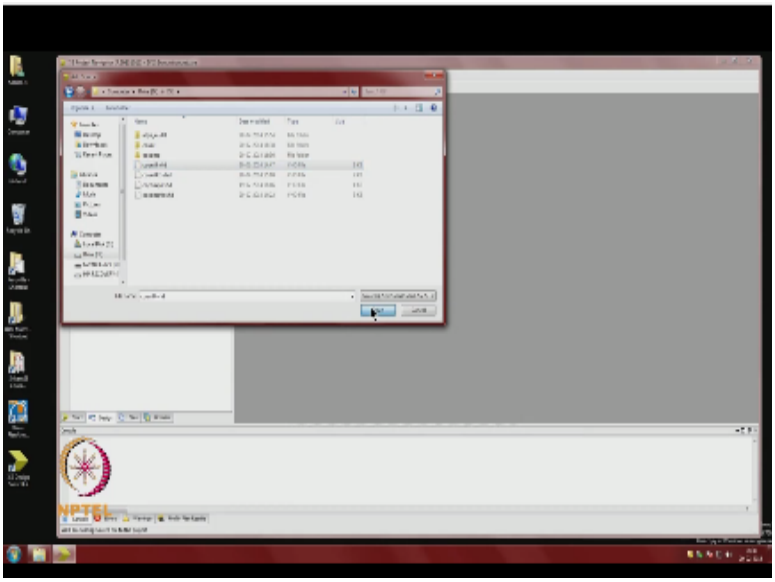
And you can move that any number and it shows you can measure the time in between and you can choose lot of things I think you can play with it I have that is what is this symbol combination circuit I want to stop here. So let me close this and I will show you a sequential circuits and simple counter which again we have discussed in the class what we have done was a 8 bit counter with asynchronous load ok.

(Refer Slide Time: 24:48)



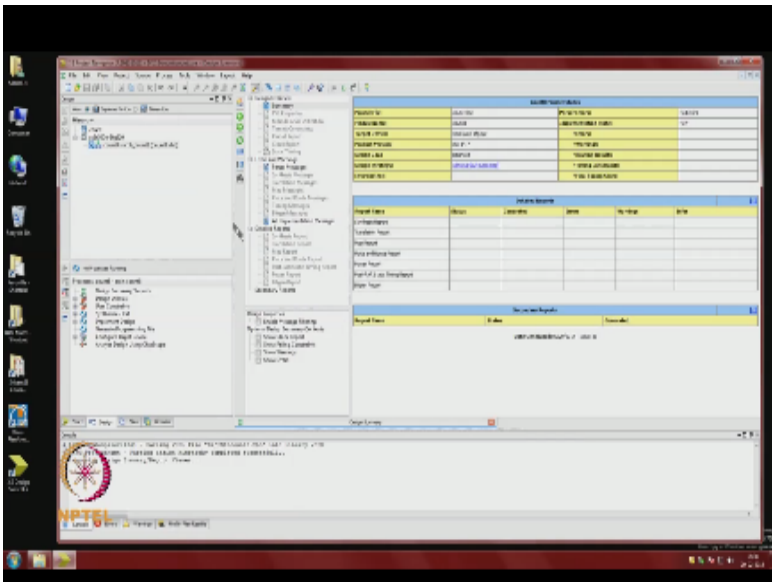
When the load signal is high it gets loaded with some values. I just want to bring that thing and show you that ok. So let me close this project, and I will open a new project ok try, again I will call account ok same thing Spartan 3e remembers that will be in your profile and we say finish what I am going to do is I am not going to create a file I just had the source.

(Refer Slide Time: 25:14)



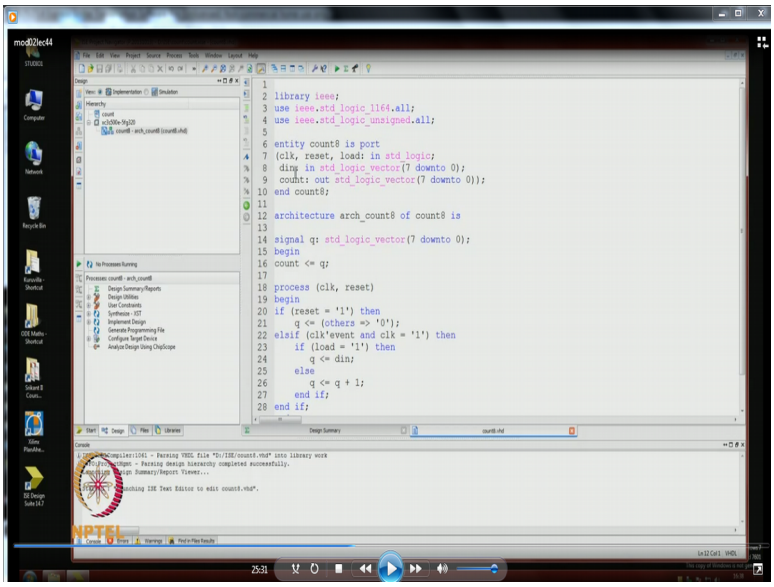
So I will add the source which I have with ready so say which is count 8 which I open.

(Refer Slide Time: 25:16)



So this is the file you know while magnify it ok.

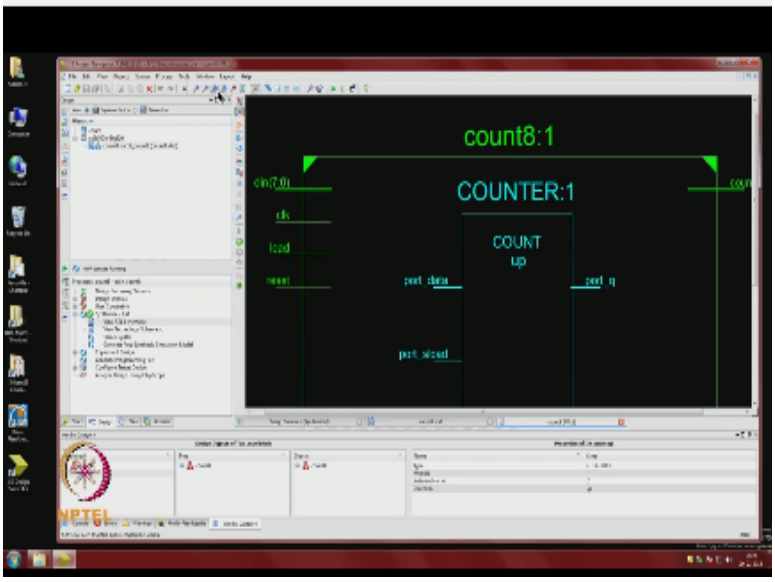
(Refer Slide Time: 25:21)



So this is the library 11, package 1164 and unsigned this is a input clock reset load is input, d is in data input which is 8 bit call this output which is 8 bit. So we have a q because here to use q is q+1 call is assign with q. This is a process where counter is there, if reset is 1 q is 0, as clock one clock is 1, then we introduce a synchronous load, if load is 1 q just din as q gets q+1 ok I have shown you everything this particular thing how it synthesize.

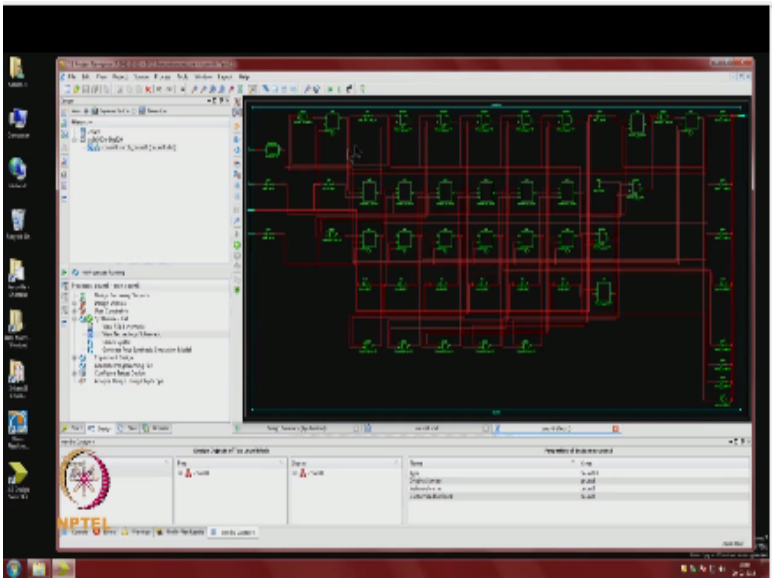
You refer back to the earlier lectures, you get a picture ok, so I just want to show you synthesise so I want to synthesize is so I run that, so that shows a quite a bit now this will have you see a sequential timing. So register to register delay to show all that.

(Refer Slide Time: 26:36)



Let us look at the technologies atrial schematic and ok now this is identified as a counter so it is not answering for the counter maybe we can look at the technology schematic.

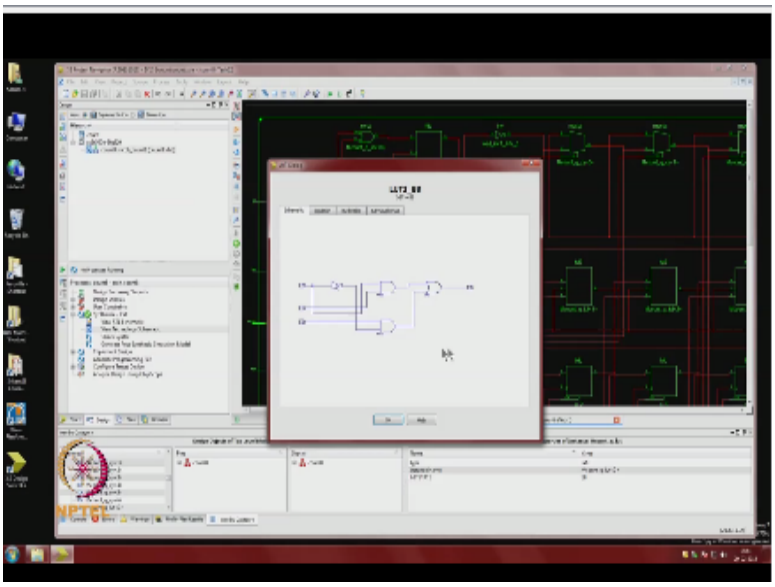
(Refer Slide Time: 26:49)



So that shows tell the whole scenario so there are lot of lookup table because you know that we have learnt FPGA we have said that and 8 bit counter will occupy it look up the carry things and this is a lookup table, if you look at let us magnify that ok, so this is a look up table for this is for D0, so this is the lookup table for Q0 and this is a flip flop for the 0^{th} bit and also you will have this is a xor combine with the carry out side xor.

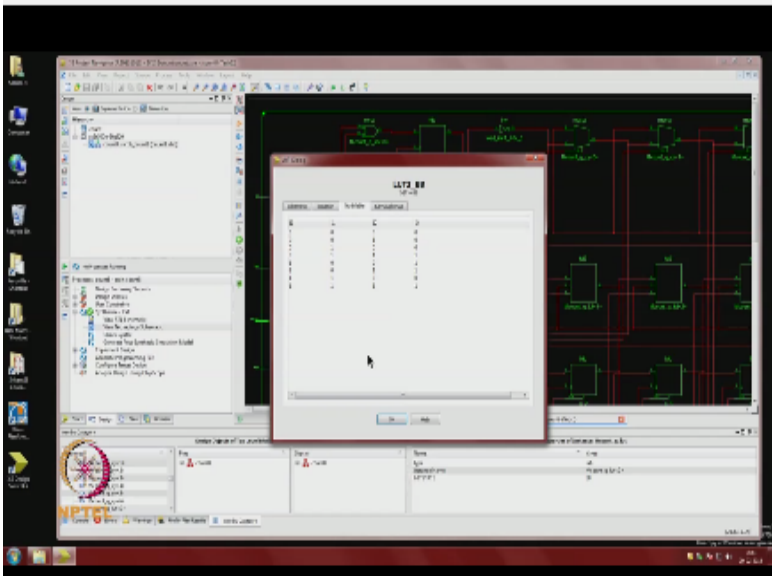
And if you look at you will see the carry chain mux and all that ok here ok. So there will be this is for fifth bit, sixth bit and if you can find it over, this is for 0 bit.

(Refer Slide Time: 27:52)



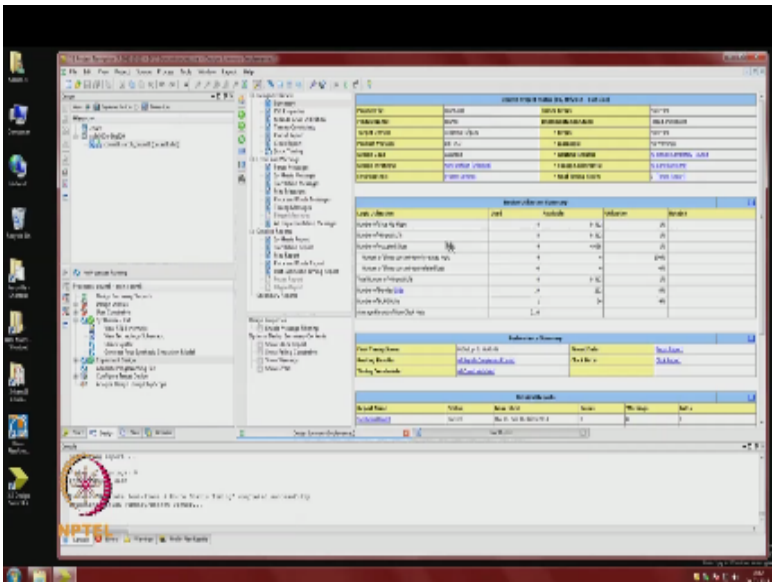
So all that is shown here and if you click on this look up table with suitable with that addition ok.

(Refer Slide Time: 28:00)



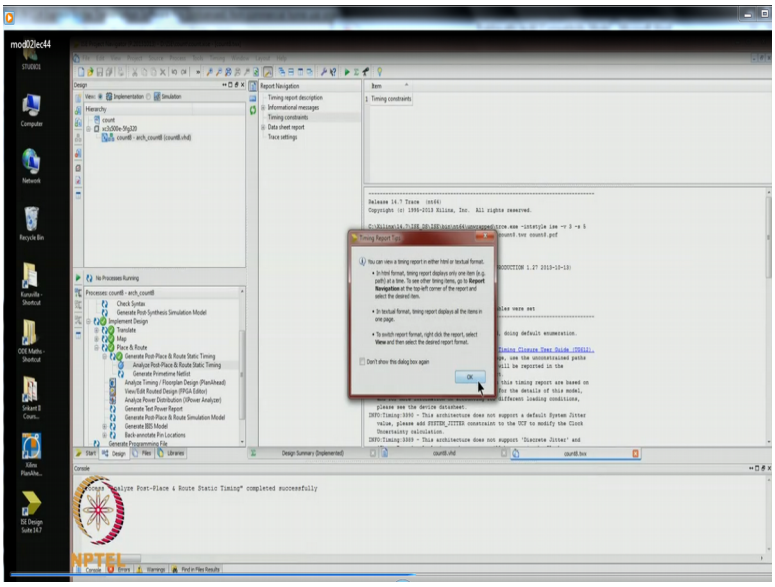
This is shown here, so you can analyse that. So I will not spend time with it, now we can implement it ok, so it now it is doing place and route initially does some process call map and translate map and now it is doing the static timing analysis and you can see the summary.

(Refer Slide Time: 28:46)



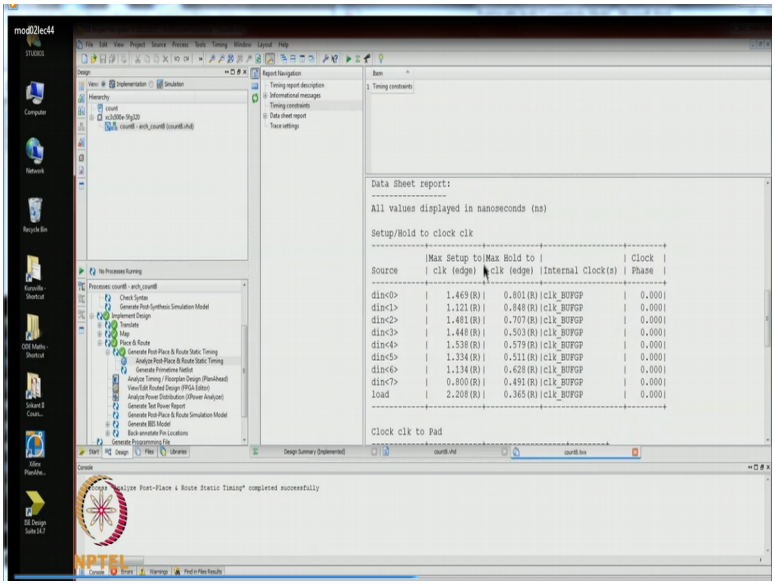
And you see there are number of flip flops use there 8 ok and number of lookup tables use are 8 and you know that there will be carry chain use, so 8 lookup table, 8 flip flop and it occupies 4 slices and what I want to show is a static timing analysis, so let us look at the static timing analysis ok.

(Refer Slide Time: 29:12)



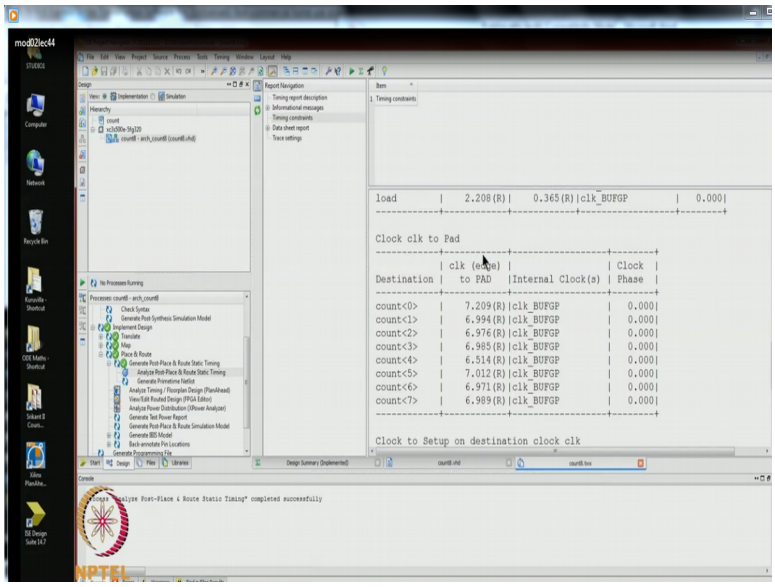
Now here you see there are various different things which I think is all important, so first thing is called setup and hold to the clock ok.

(Refer Slide Time: 29:25)



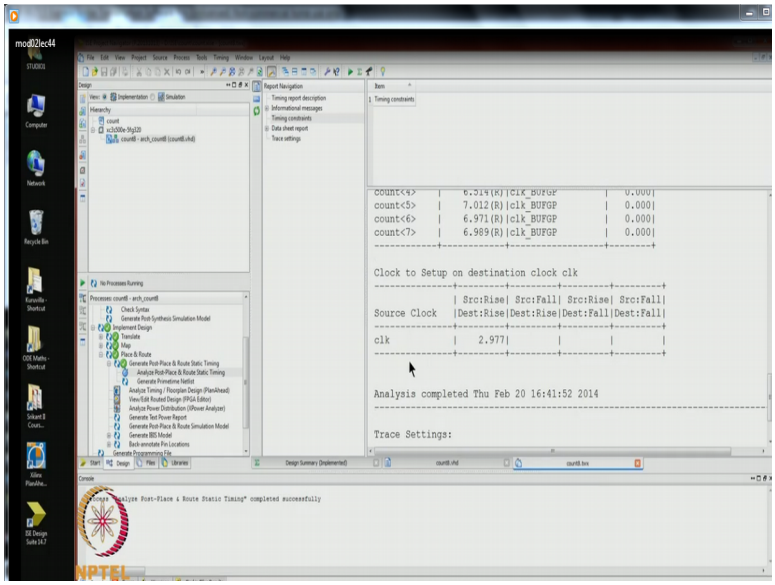
So this shows the data and the load signal, it shows what is the setup and hold time of each of the external input up to the flip flop because data is going through some kind of combinational logic to the flip flop and it shows that the data in 0 as a setup time with respect to the pad 1.469 second and the whole time is 0.8 nanosecond.

(Refer Slide Time: 29:56)



Similarly there is a clock to pad which we are discuss this shows that given the counter output, it is like equal to tcq, but all the way up to up in because the counter going to a pin from the clock edge what is the delay all the way up to the pin, not just at the output flop so that's a count 0 is 7.2 nanosecond.

(Refer Slide Time: 30:23)



Similarly this is a setup time of the input all the way with respect to the pin ok and this shows o'clock to output with respect to the output pin ok and the lastly what it shows is that it is just not to setup that is nothing but a register to register delay, so this show the minimum clock period okay, the minimum clock period is 2.97 nanosecond. So basically our counter will walk with 3 nanosecond clock period.

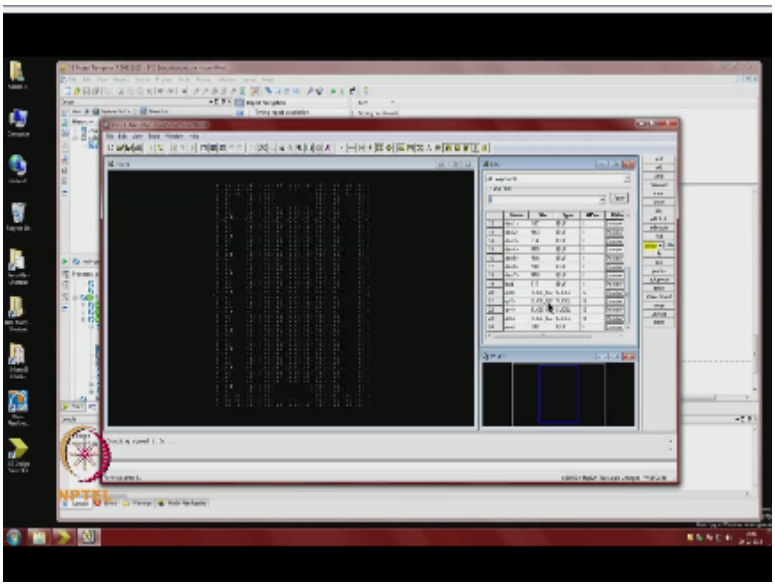
But you see the since it is placed within a FPGA for the count out to come on then I have 7 nanosecond ok. So even if you simulator 3 nanosecond though the counter internally works properly, the input will not read in 3 nanoseconds output. So it will be forced to kind of simulated at 7 nanoseconds and run the design at 7 nanoseconds or 8 nanoseconds. Similarly setup time is not a big problem because we taking 1.5 nanosecond.

That can be set up in the lord as 2 nanosecond, so still it is within the 3 clock period but that could be some the all that combination delay is accommodated here. So this is something which have to realise though the clock frequency can high because of the delay to the pin, you will be forced to choose the slowest of this as a clock period when you stimulate it, but now that does not show the your designs performance.

Because many times you are designing an IP where the outputs are not taken to a FPGA pin that is going to a nearby another circuit ok or in an for advance use this can be or these things pins can be assigned very close to the location where the logic is place all that is possible. So in when you do the static timing analysis of sequential you will have the setup time with respect to the pin, it is call setup to clock and clock to pad tcq with respect to the clock.

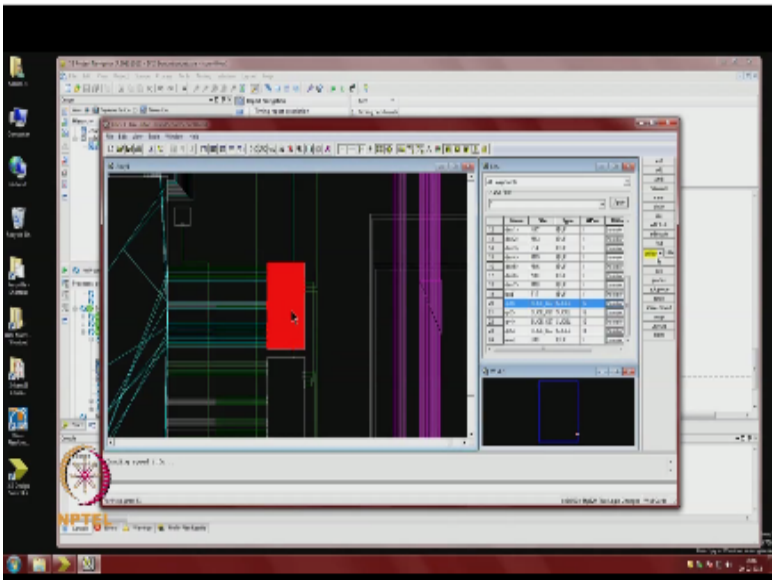
And register to registered delay and I am not trying to that I have shown you the simulation so I am not going to simulate that because we have to complete that our case study on this particular tool, so let us look at maybe you can you know you can look at the edited outer design to which you know say here you see you come down.

(Refer Slide Time: 33:09)



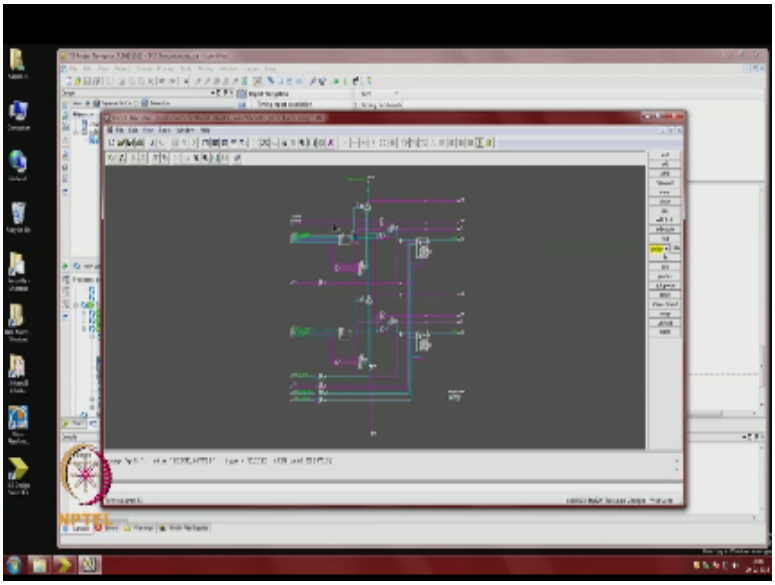
And you pick up a **a** slice for the q0 then the slice come.

(Refer Slide Time: 33:12)



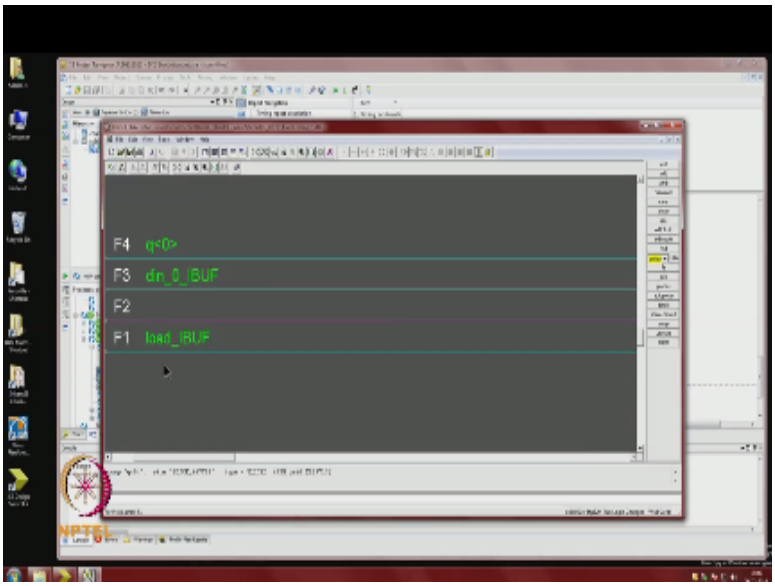
And you can see this is it.

(Refer Slide Time: 33:17)



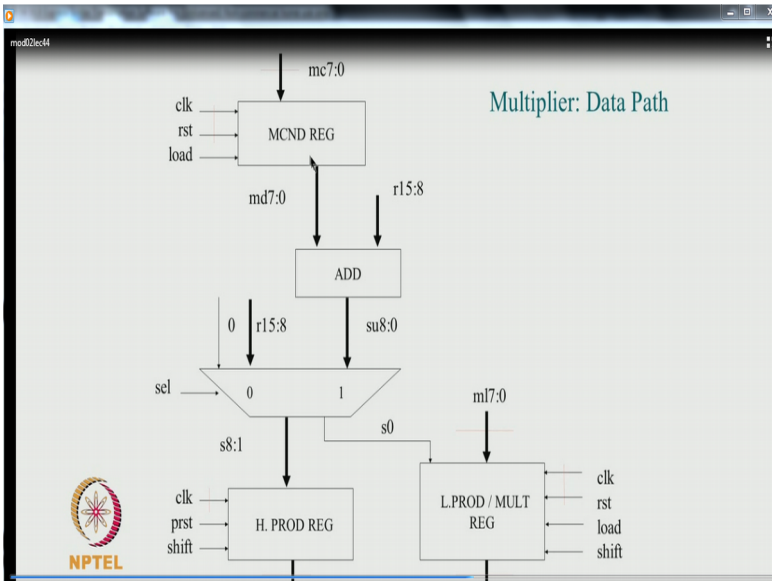
And you can you can very clearly see the q0 coming here the q0 data in load which we have discussed how a single look up table can accommodate a data in q0 and load.

(Refer Slide Time: 33:34)



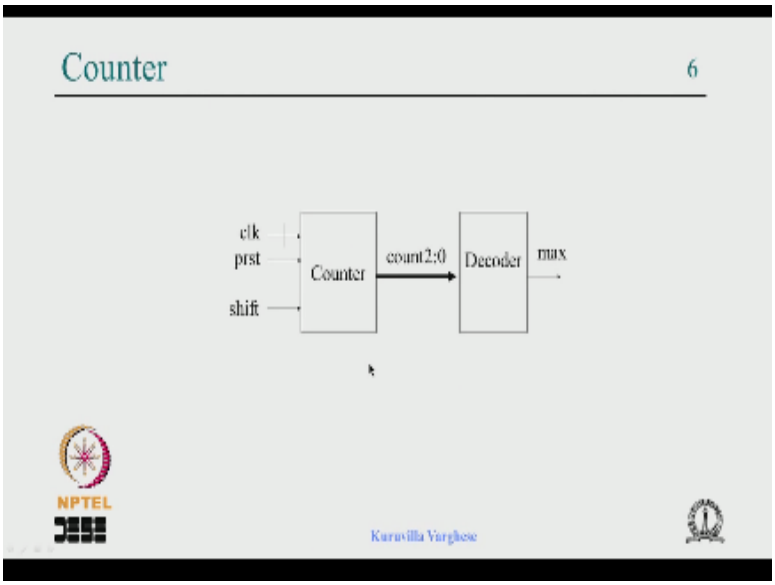
You refer back to the clock that is the lookup table and that is coming with XOR gate and this is a carry chain wires maybe else more bit to be able to see that, so this is a carry input and this is the carry mux which is going to the next stage here to look up table ok. So this can be very well analyse whatever I have shown in the earlier lectures that you can find out and verify what I have shown is correct or your own ideas are correct.

(Refer Slide Time: 34:16)



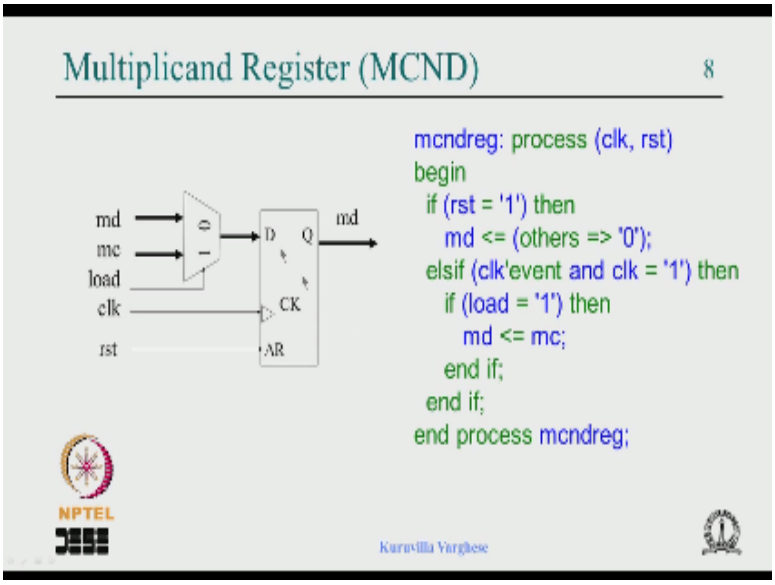
So at this point I will close this project and I will come back to the slides you have the keys study we have done briefly to remind you so what we have done it was a multiplier and we had implemented you know the data path multiplicand result adder, mux.

(Refer Slide Time: 34:35)



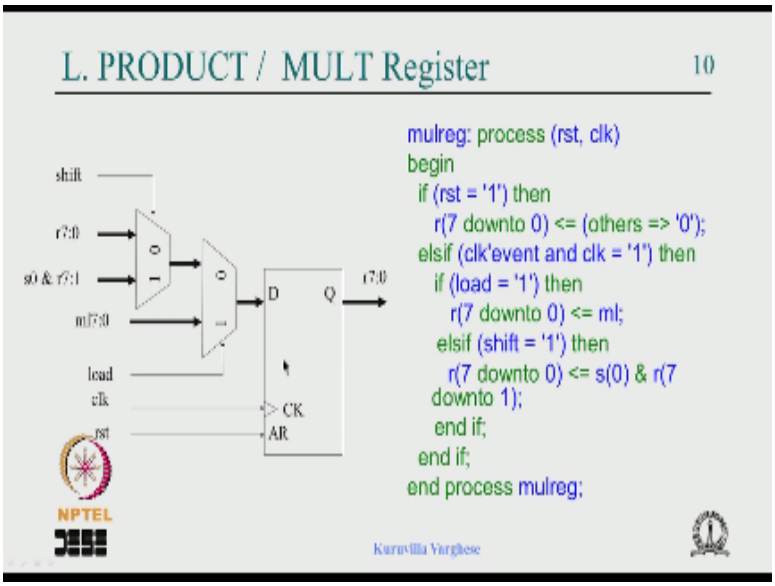
Then a counter to keep track of the count controller.

(Refer Slide Time: 34:39)



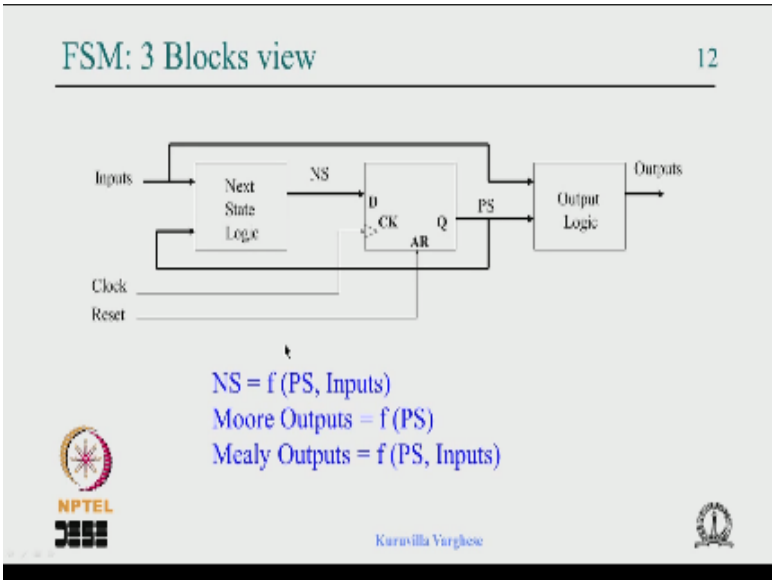
Then we have expanded multiplicand register.

(Refer Slide Time: 34:42)



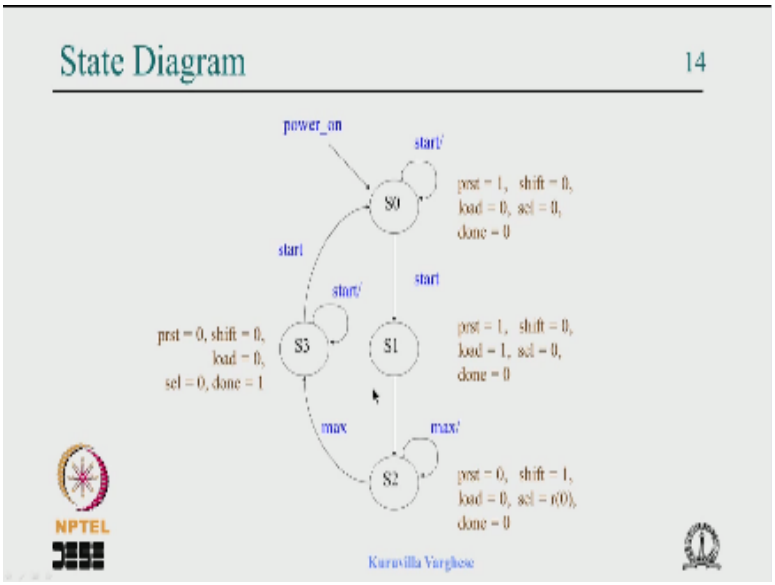
Multiple register, counter, the FSM.

(Refer Slide Time: 34:45)



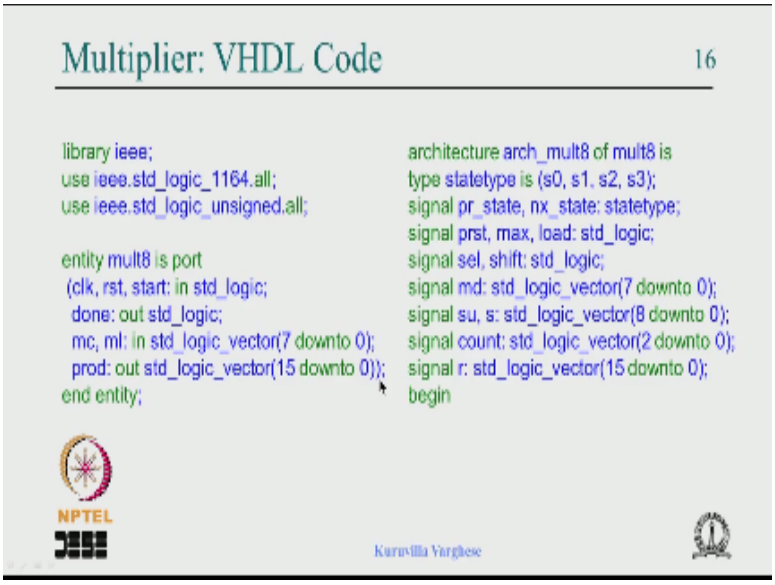
And you have seen the state diagram.

(Refer Slide Time: 34:49)



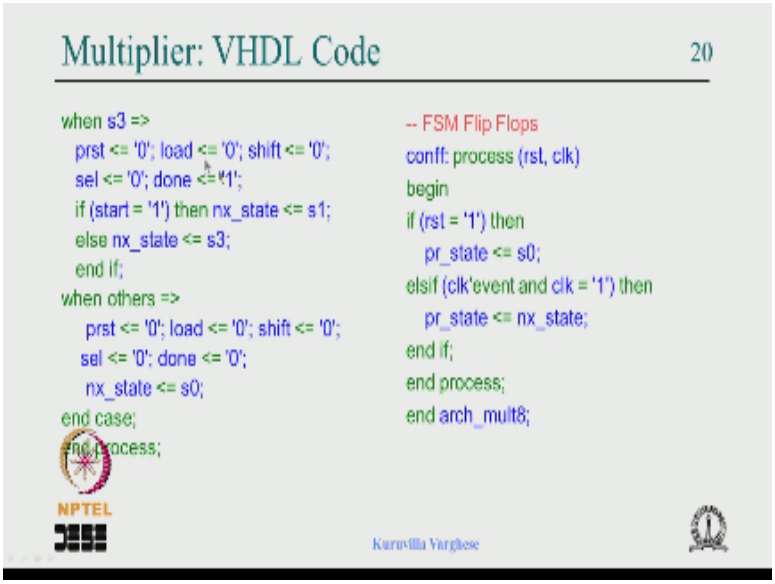
And this was a code you know this was the big code.

(Refer Slide Time: 34:49)



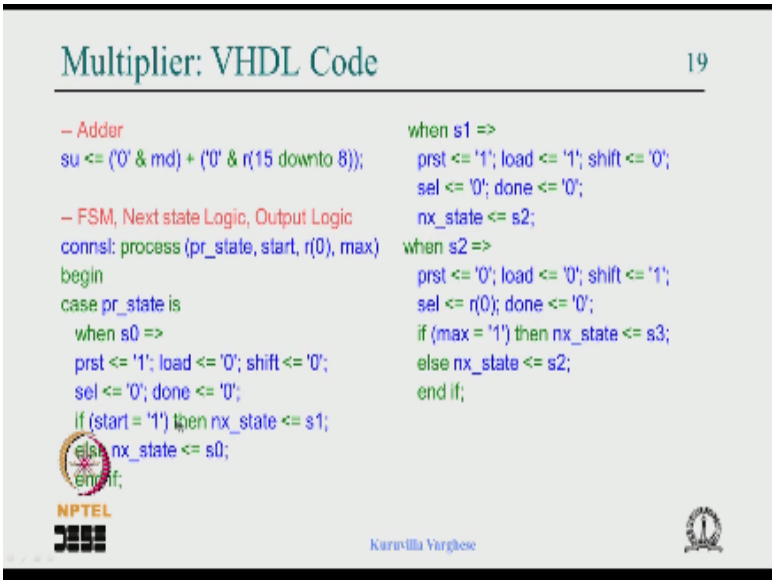
And the multiplicand and multiplier.

(Refer Slide Time: 34:56)



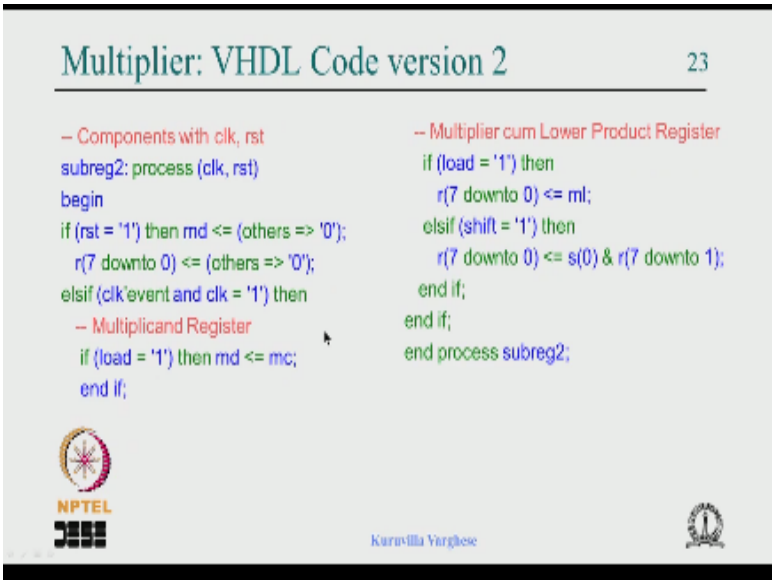
The multiplex higher product register counter.

(Refer Slide Time: 35:04)



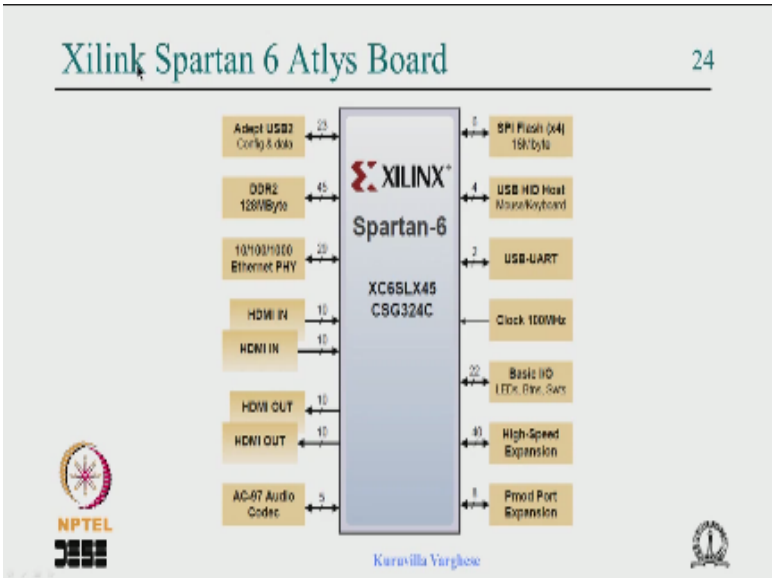
Decoder then the next state logic and output logical have some adder and FSM flip flop, so that was the code ok. Now what I am going to do that I am going to program this in a knife PG ok now this is a state diagram ok.

(Refer Slide Time: 35:22)



Now the particular FPGA board I am going to use is this particular Xilinx ok.

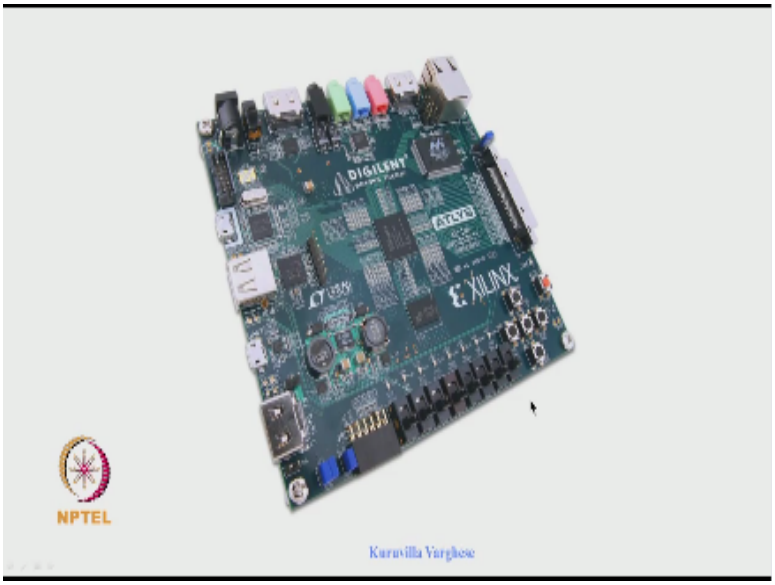
(Refer Slide Time: 35:29)



Sorry for this mistake it is Xilink Spartan 6 at least board and this board as a Spartan 6 FPGA which is 6XLX45 which package is CST 324 pin you know the board grade chip scale the grade that means the there is concentrate squares beneath the chip with the pin and in a lot of things in a very powerful board you have a clock we require something clock. We you have some LEDs connected to some pins.

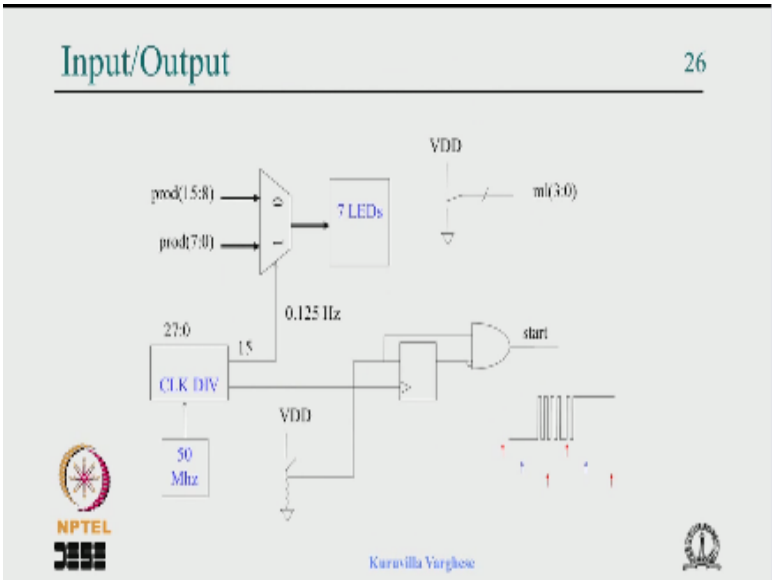
And we have some buttons, so that we can give some reset start, there was slight switches all that and there is Ethernet memory to USB HDMI audio connect and all that, but essentially we are going to use clock source because we are going to do something with clock and because the state machine need the clock and then the LED is to display the multiplier output button to give a start at the multiplier. We have discussed that what is the start come from the external thing.

(Refer Slide Time: 36:44)



So that is how the FPGA board looks this is where the FPGA is, this is a FPGA and you see the lot of port, this ODI port, USBs this Ethernet port and this is a flash memory and maybe these are the HDMI port. This is a program port, these are the sites which is very can give 1 and 0s and these are suppose not LEDs, 8 LEDs ok and these are the buttons and there is a reset button and 5 buttons like cross where can give a push button ok.

(Refer Slide Time: 37:29)



Now we have to show the result of the multiplier so I am going to give them the multiplicand fixed ok. So within the program I am going to assign hard assign the multiplicand and within the program apart 4 base multiplier is assign hard, but the lower bits I am taking from the slide switches ok. So I can choose to lower bid as a up to say 000 all the four 0 to all the four 1s ok. 00 to f index okay can be changed.

And see the result because we are implementing we have implemented 8 x8 multiplication with 16 bit. So these are sorry these are once again there are 8 LEDs ok not the 7 LED, 8 LEDs, what we're going to do that we are going to multiplex the 16 bit on this LED with a 0.125 hertz that means say once in 4 seconds it will show the most significant byte then the re-significant byte okay.

So we are using there is a inbuilt clock oscillator which is 50 megahertz we will implement a clock divider ok and that will generate 0.1 to 5 hertz and that is used to multiplex this LEDs and this 50 megahertz can be used for the state machine of the controller and one other thing I am going to do is that I want to give us start signal ok. So I can use a push button there is a push button on the board say these buttons ok.

Now the problem is a push button is that when you press it does not make a clean contact ok it just goes and bounce you know like that is the pictures shown initially it is 0, when you press it kind of goes up and down and make a 1 ok. Now we need to de-bounce ok we need to remove this glitches otherwise restart many times ok. So what we do is that we will sample, so what we are trying to do is sample with clock period which is which are the period which is higher than this bus ok.

So we have to have some sense how long this bus for the most good switch off bit locker say around 20 millisecond like that ok, even less so sometime it can very bad switches big switches can I know that but we assume that it is it is around 20 millisecond, so we sample at 2 points and if there is a 0 and 1 clean 1 then we will you know make it a start. So that is what is done by this is a flip flop, you see here the input is going.

And we are giving at 20 millisecond clock and I am typing one of the queues of top clock divider with appropriate you know waveform and it need be very correct because this will kind is a mod counter after sometime it is going to become 0. So the waveform will be little bit restore, but does not matter, not a very very serious thing, so what we are going to do if we are shifting that to the output.

And the earlier one comes here the current one is here, so we are looking if it is 0 and that is 1 then we get a pulse here ok which is of the duration of the clock period because when it is 0 and 1 this is happening with every kind of so that you get up a pulse here you say start and

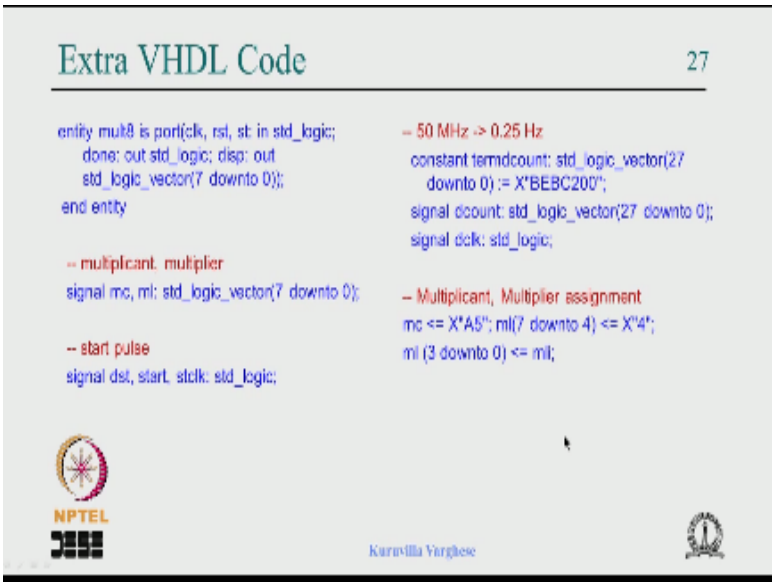
this pulses is kind of little big of the 20 millisecond duration, if you want make a narrow we can resynchronization this, but this is this pulse is of 20 millisecond duration that could be a problem.

Because if it is very long our multiplier will completed and restart again because the multiplier was very fast and their sample, saying things can happen suppose 1 come here and one come here, so if you like it is both are 0, if it is this is 0, this is anyways 0, this is 1 we got 1 transition it may happen that is very clean we sample here 0 and 1 then clean thing maybe that you know we sample it here.

If it is 0 then this will catch 1, if it is 1 the previous 1 would have 0, so that will catch it. Anyway it works this declutching work recently but only thing is that is duration is 20 millisecond maybe can put on other flip flop with clock to get narrow pulse, that possible I am not found that, I have done that little change to the state diagram. I know that the start going to be long.

So what I did that when we come to the finals state I remain there till the start go slow and come back here. So that you care just to demonstrate but real design you better put one more flip flop here with the same fashion same circuit which stock by 50 megahertz and we will on the transition you will get a **a** single cause of 20 nanosecond duration that should be enough for our controller.

(Refer Slide Time: 43:18)



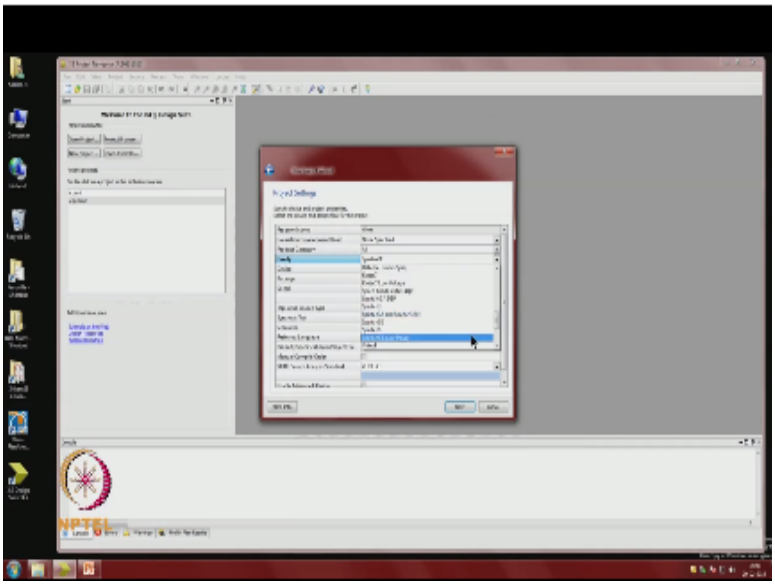
Because the controller also work with 50 megahertz clock and that is the additional the code required for multiplexing so I have some start signal start signal do not think like this this particular clock I am go naming as ST clock and you see this is a hard assignment MC is assigned A5 the ML most significant 4 bit are assign 4 and this is the ML 3 down to 0 which is coming from the input pin.

And this is a constant which is a representing 50 megahertz or 50 megahertz and 4 and to generate point 1-5 and you can see how that count works with the divider upon the reset the that huge 27 nanosecond make 0 up on the clock it is incremented but if it reaches this particular number it is reset and make a square wave make the d clock which is 0 at the terminal count it is kind of toggle not the d clock.

So you get a divide by 2 out of the you know that is the counter and this is LED multiplexing we are using this Declock when it is 1 or 7 down to 0 goes to the display and else or 15 down 0 goes to the display and this is a dcount 15 is this for the debouncing you see that the debouncing here, reset is 1 that is dst 0 up on the clock dst get that so that is representing this flip flop.

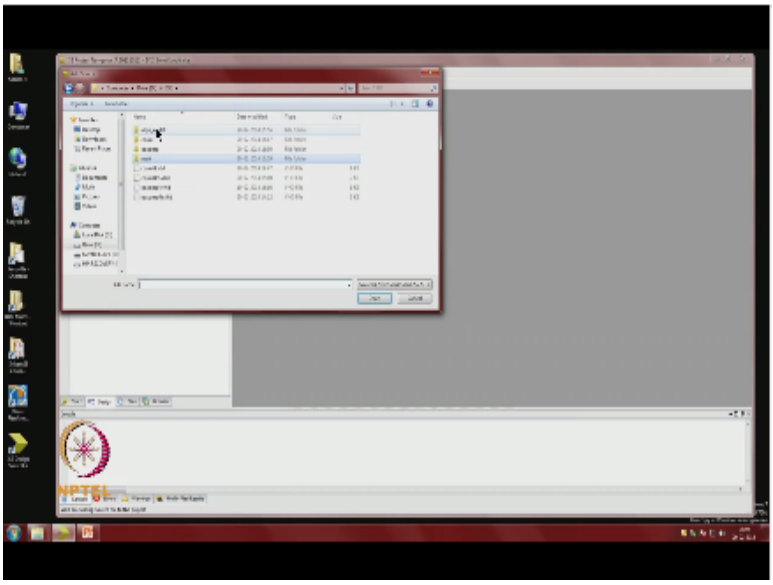
And this AND gate is written here, say start is nothing but ST and non DS ok and that start goes to the state machine and I have told you about the state machine chain, so let us go to the to the program ok and let us move it and let us come here and open this thing I will open a new project let us call it as multiplication mult.

(Refer Slide Time: 45:35)



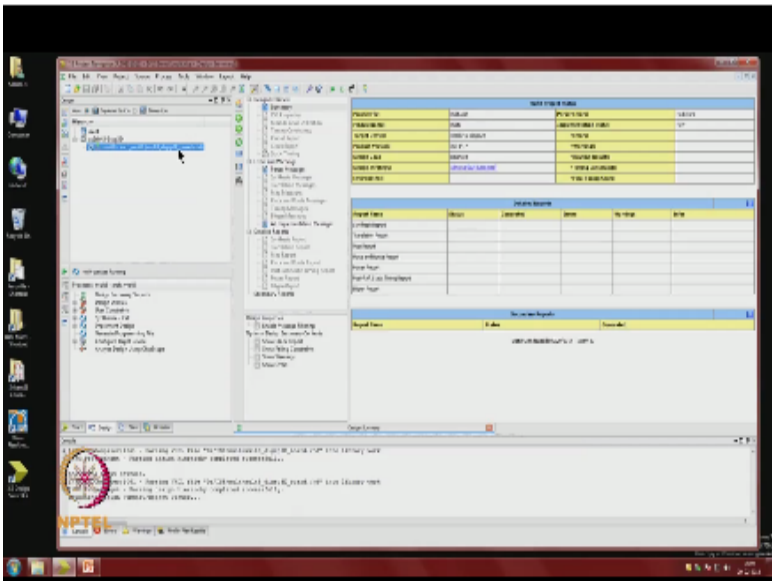
Now it is very important that you select the correct device as it is on the board which is Spartans6, the correct packages be chosen then only to work because the pin number change depending on the package it XC6SLX45, now this is a package CST324, I will say next finish.

(Refer Slide Time: 46:06)



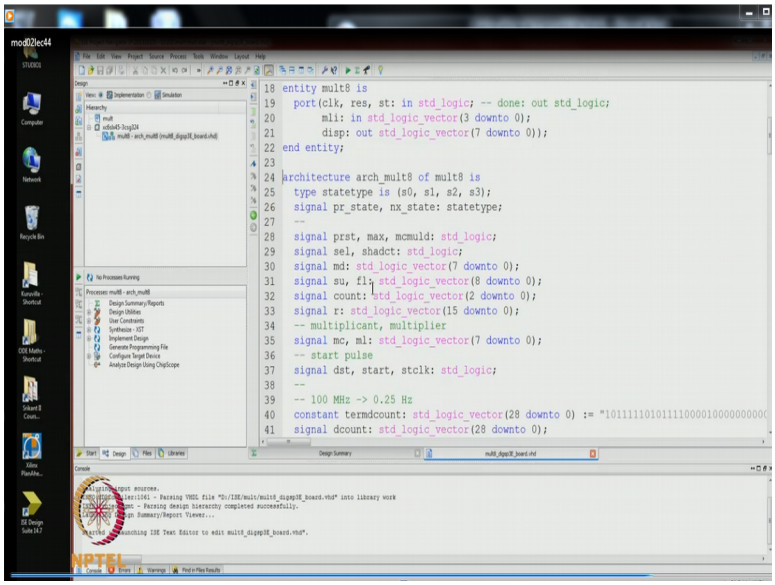
Now I will add the source file let me add it up I will pick it up from ok I will cancel it and I will copy, add copy of the source, I will go here that is the VHD file open and done ok.

(Refer Slide Time: 46:23)



So that is a huge file is quite a bit huge.

(Refer Slide Time: 46:27)

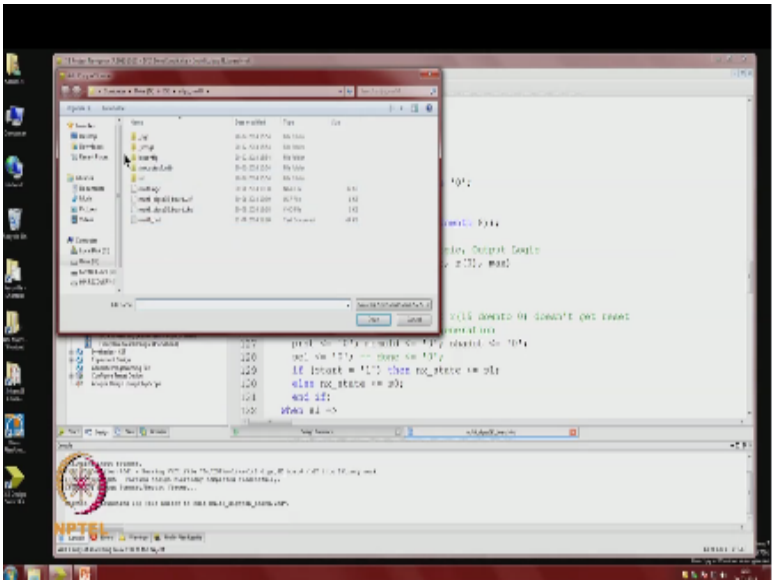


So you can see the library package multiplier, then the architecture with signal and this is the constant for the counter and this is the various multiplicand register, multiplier register, multiplexer higher product register, counter then decoder. This is a counter for multiplier, adder, this is the next state logic and output logic of FSM and this is the FMS flip flops. This is a multiplexing counter ED multiplexing, this is a LED multiplexing.

And this is the start pulse generator okay whole thing is there. Now very important thing now is that when we do this kind of thing you have to assign the pin because we are going to now you see put this program into a FPGA, this FBGA some of the pins are connect one of pin is connected the clock, some pins are connected to LED. So your signal should be map correctly to this and there is a way of doing it graphically.

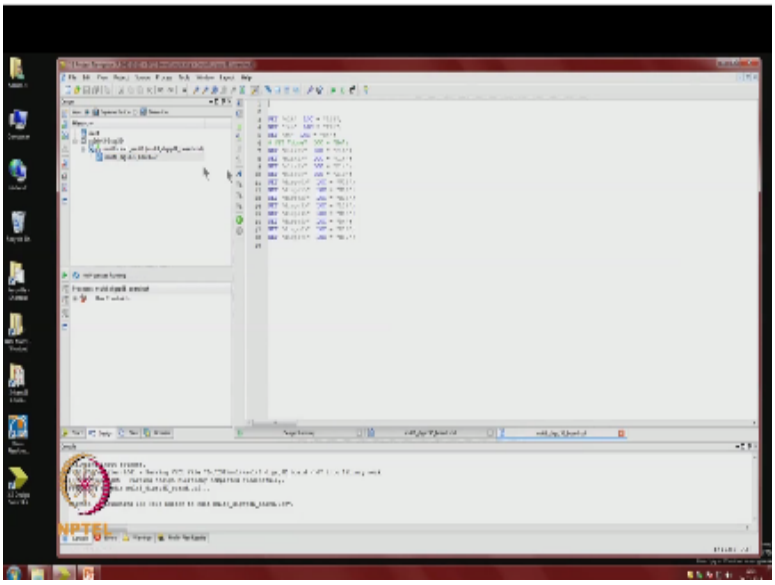
But there is also a way of doing it to the by a file called UCF file, so that file in a you can say you can go to use and you can say I/O pin planning re-synthesis you can click it and you can assign it manually and create that UCF file.

(Refer Slide Time: 48:13)



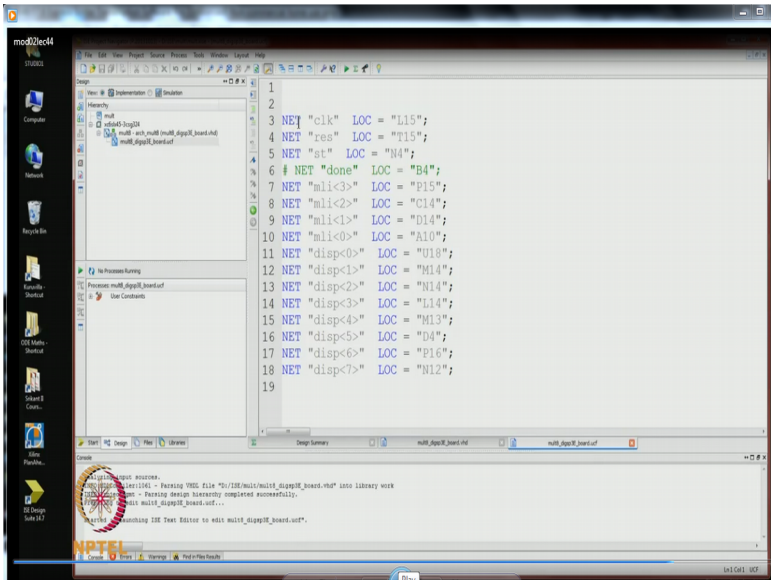
But for convenience I will I have prepared that use your files I will pick it up from what is pre prepared ok, I will copy that UCF file here.

(Refer Slide Time: 48:26)



So if you see the UCF file this is the syntax and that if you see this UCF file this is the syntax and may be you see that it shows the net clock location is L15 that means the signal clock for us is connected to the pin L15 because on the board this particular clock oscillator is connected with L15, so we are tapping the clock, similarly you see this place map is something, this is what LED are connected.

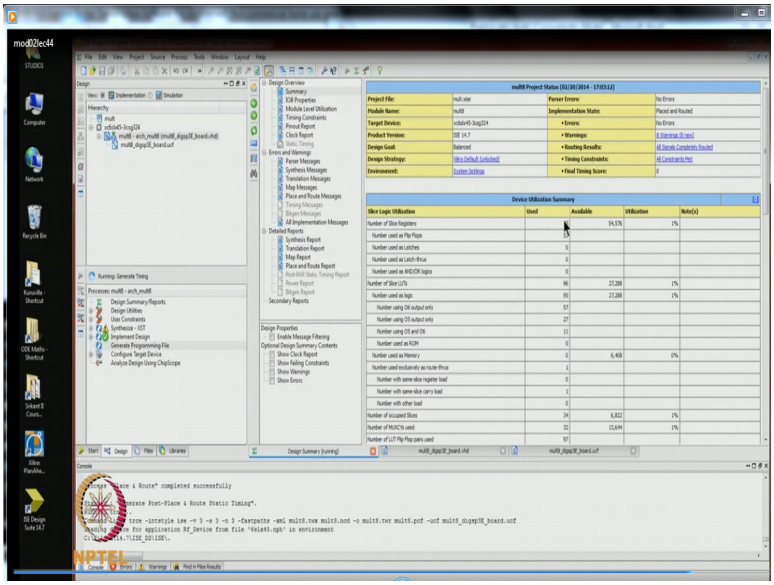
(Refer Slide Time: 48:56)



Similarly the multiplier input is connected to some pin that was slight switches program and the start coming from a button, reset coming from a button, so this is very important. Now I am not going to simulate I am just right away going to generate the programming files, so it will do synthesis place and route programming I am generator programming file, everything is done.

It might give you some warning but it does not matter because I have done some kind of assignment it is made of the most optimum, so that is doing the place and route the translate synthesise, translate map, place and route all that ok.

(Refer Slide Time: 49:54)

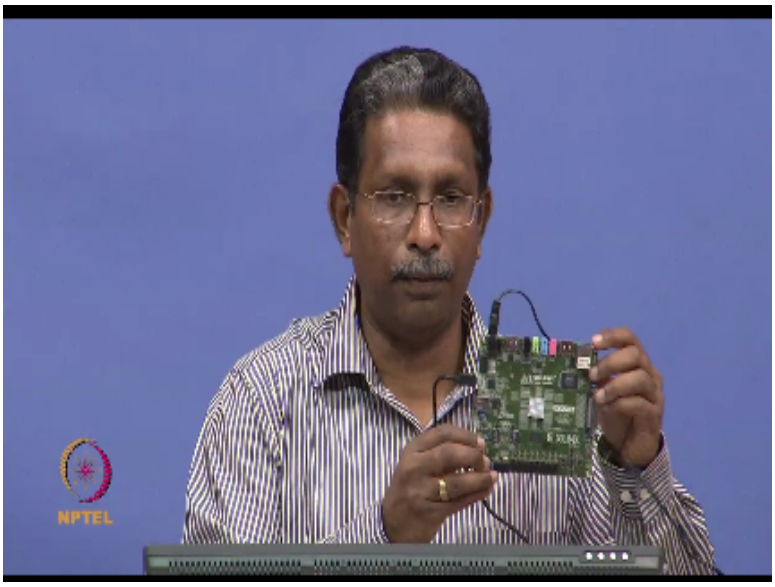


And it will show the design summary it is using the 53 registers ok and the lookup table is 96 because it is a quite a big design and we are using lot of counters you know counter huge

counter to generate that de-bound demultiplexing signal, the counter, 8 bit counters, lot of register, so it is 53 registers and 96 lookup table using. So it is done complete the static analysis, but I would not go through the other part like all that let us write a program that.

So before that I just want to show you what I have assigned the input say come here you see the multiplier, multiplicand is assigned A5 x number, multiplier higher been assign 4 ok. Now I want to show you the board.

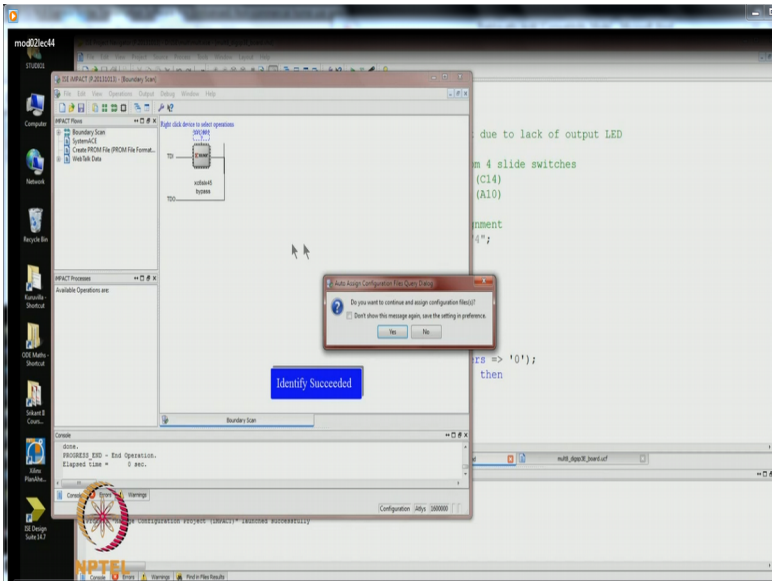
(Refer Slide Time: 50:52)



So this is the particular board ok, so this is what y. Picture I have shown, so this is a FPGA with heat sink, this is a power supply , these are the slide switch, so I will keep this 4 bit at 0, these are the LEDs ok. Now what I am going to do I am going to program it, so let us go back to the tool, ok, so let us come back to the software so that how do you program is at you click this and open that thing and you say configure target device and you say manage configuration project you click on it.

And make sure that the board is switched on, you connect the power the programming is done through a USB cable ok and make sure this is powered on, now had to the tool ok and you see you click on the boundary scan ok it comes here. Now you say right click and say initialise chain, so it is detected the FPGA, might ask whether you want to assign a configuration file you say yes and you open up the correct bit file in the project you click on it, you say open, last can other question for flash from say no, say ok.

(Refer Slide Time: 52:19)



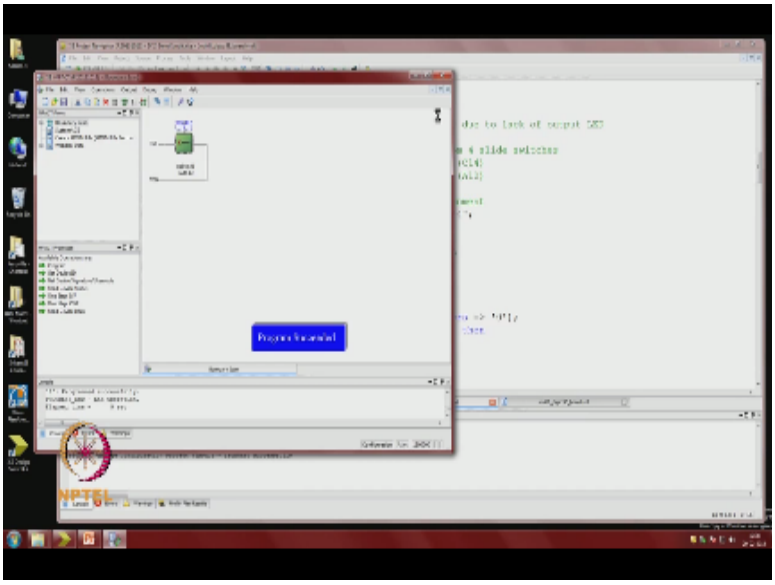
Now right click and you say program and you can see that it is programming the board, now let us look at this particles programmer you see that, when that happens you can see there is an LED which is a green LED which is called it done that LED is blowing ok this particular is blowing. Now so now what I am going to do is here I am going to reset to it, I am going to give a start ok.

So now you can see the that these LEDs are not lighted ok, so I give a stop, pause now you can see that LEDs are blinking which is A5*40 actually a real result is 2940, so maybe this so small you cannot see from the camera but then it just shows the 29 and 40 and I can change the values say A5 and 4F and I give a start signal for you can see this LEDs which showing 32 and ED 32 is less lights and D is more like more lights now.

So that is how you use the FPGA board by programming a lot more can be done. I just want to show a quick kind of run through the tool. So we have seen today assemble combination like thing we have seen the synthesis, we have seen RTL schematic, technologies schematic, static timing analysis, inside the lookup table and all that you know all the details which is in counter design, a sequential design against static timing analysis.

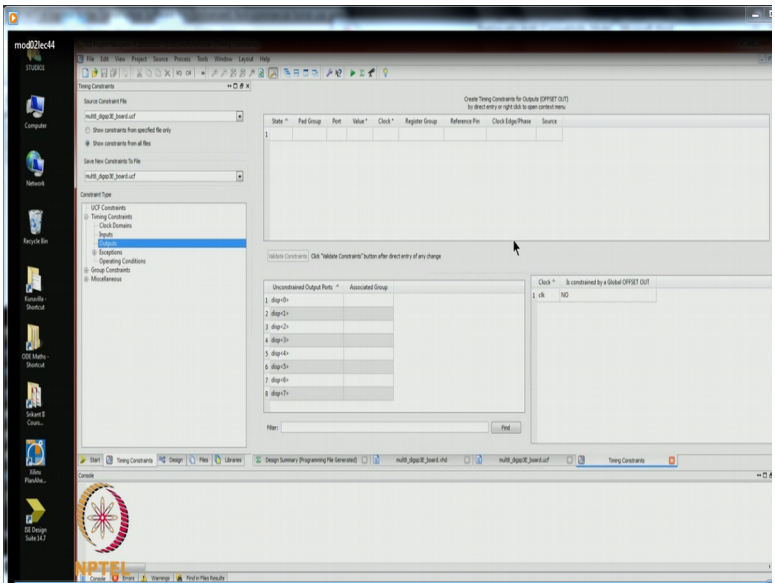
The whole case study we have with an additional some addition we have put the result on to the LED ok. One thing remember is do a pin assignment and you can do something all you have to confirm maybe one thing which is remaining may be come back to the tool you can do something call the timing constraints ok that I will quickly I will find out.

(Refer Slide Time: 54:36)



Then you can play with it, so it is possible that you can specify the 5 timing, so here you see the create timing constraints and you can see that you can specify the clock period say the clock domain, various clock, the input, output.

(Refer Slide Time: 54:56)



What is the setup time required, what is a clock to output and you can group some pins and you can specify the setup time clock you know the period all that, say quite elaborate and when you do that this will be converted to the entries in the UCF file of time constraint, so we can achieve something performance you know you can say that your design should fine 5 nanoseconds, 10 nanoseconds, or part of the design should work with 5 nanoseconds.

Another part by say 7 nanosecond, all possibilities are there, that you can play with it but there is mode to do the tool I suggest you go to tutorial and get acquainted with that. So with

this I am kind of completing the course, I have covered in of the digital design very detailed, how to design top down and controller design issue, then the VHDL for synthesis and FPGA in detail, how given a code with maps to the RTL code, how that really gets implemented with FPGA.

And we wind up with case study showing at the tool showing it on FPGA board. So I hope that has been a good learning experience you can build on this I know I normally cover quite a lot in my course in the in the university I teach, but I do not have time to cover all that and this was committed at the beginning of the contract. So I stuck to the syllabus, so please study well go through the material, go through reference book.

Workout lot of example in the line I have suggested, then you can master it, you think you try to understand and then you can build on this basic which I have given, I hope whoever was watching this watches course will become an expert VLSI design, so I wish you all the best and thank you.