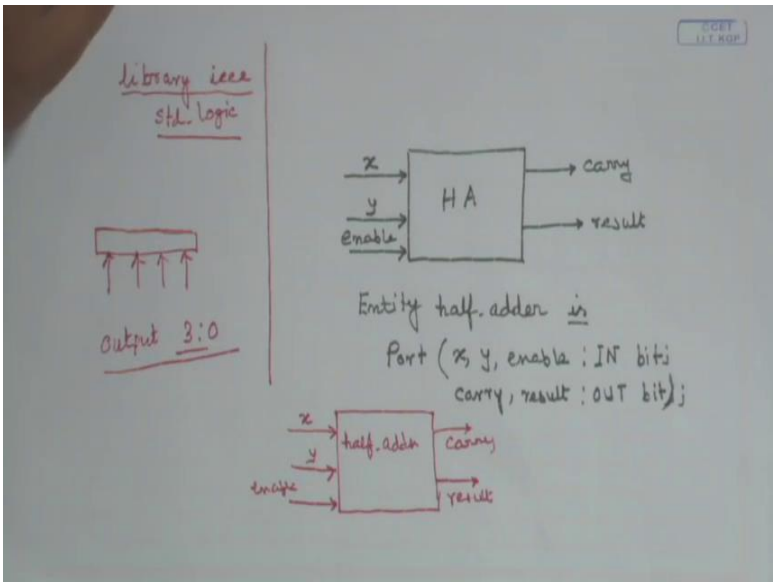
**Embedded Systems Design**
**Prof. Anupam Basu**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Kharagpur**

**Lecture - 08**
**Behavior Synthesis on FPGA using VHDL**

In the last class, we were discussing about the VHDL description of entities.

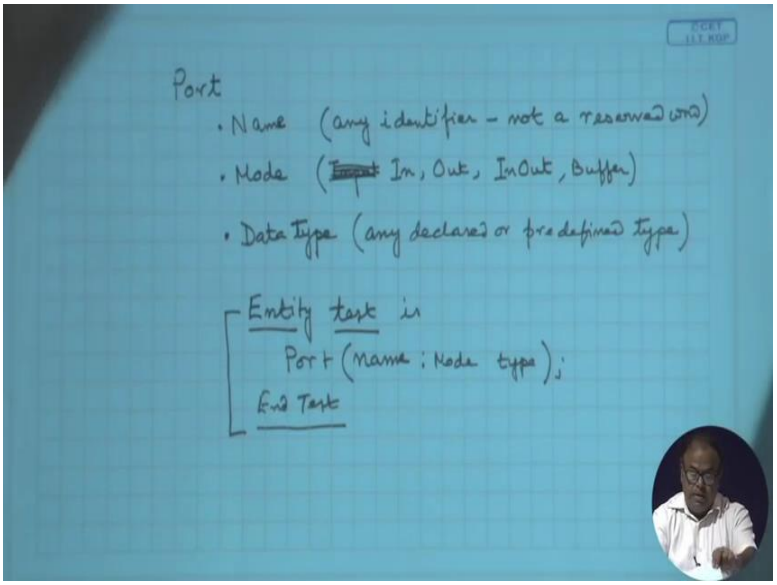
(Refer Slide Time: 00:25)




So, we described the half adder and which has got some inputs and outputs and we had shown that we can write that in VHDL as entity some name; name of the entity is and then describe it with some ports. So, that is complete description of the entity as a black box at the top most level. So, again we reconstructed, if we had suppressed this and we for purely from this if you had reconstructed the schemata of the entity that would be the same as it was being shown here.

Now, also we described that we include actually in VHDL some libraries like the ieee library and the standard logic library.

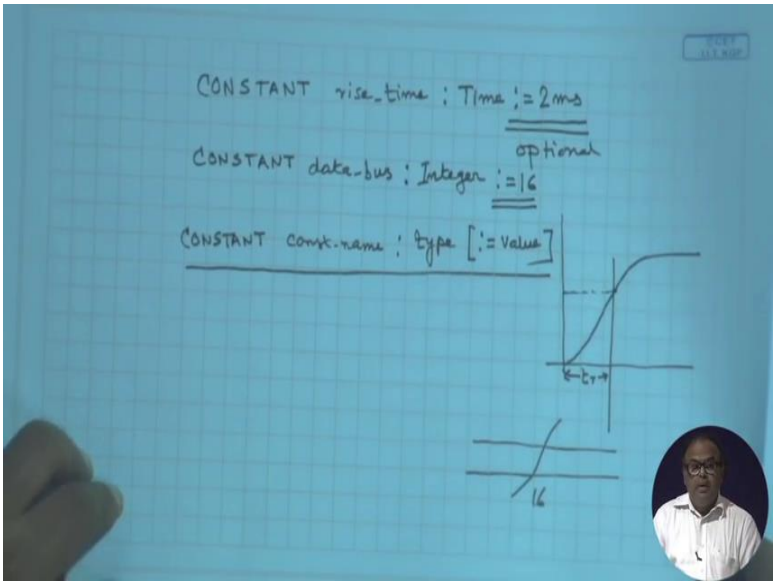
(Refer Slide Time: 01:43)




Now, as you can see here the port description can be formally stated as say a port; any entity in order to interact with external world will require the ports and the port will have 3 parts, one is the name of the port that we have to give name of the port; any identifier which can be any identifier, but not a reserved word. The other thing could be the mode will have to be the mode that is whether it is an input or output. It can be typically, it can be input or actually we write it as in, out, it can be inout, which is bidirectional or it can be a buffer also that it will store something for a while.

And the third component is a data type; the type of data any declared which can be any declared or predefined type. Here if you just take back this, we can see that we have got the declaration port and so I can declare an entity as again; a different entity like entity suggest some entity test is this is the name with the ports. Now port can be whatever port you give say name, then colon, the mode, in or out and then the type; that means, the data type, semicolon and here should be end of the declaration end test. So, these are complete here also here also I should have given; I should have written n half adder. So, that is how we declare the ports, fine.

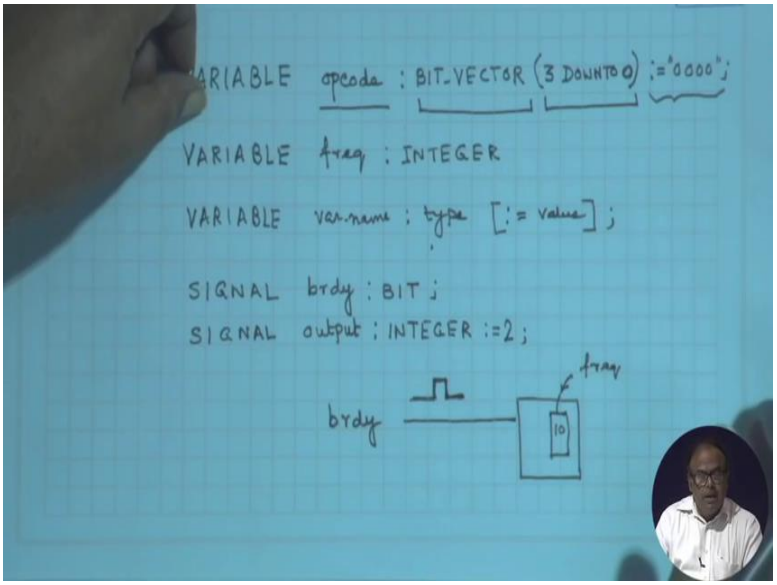
(Refer Slide Time: 05:27)




The next component that we need is need to know is some constant, we can use the reserved word constant and we can give some constant name. Say for example, rise time; rise time of a signal, you must be knowing that whenever a signal is said to 1, it takes some time to become 1 and 67 approximately 67 percent of the value, the time it takes to come up to this level is known as the rise time. This is the rise time or t r. So, a constant rise time and then I put a colon and the type is time. So, it is the type. So, here and then I can this is optional, I can initialize it to say 2 millisecond. Now this is optional, this part is optional, alright. Similarly I can declare the data bus. So, how do I do that? I can write constant I did define a data bus data bus.

Now, this data bus is of type integer and I can initialize it to 16 second, again this is optional, what does it mean? That means, whenever I write a data bus; that means, I am mentioning a 16 bit data bus, alright. So, the width of the data bus is mentioned as 16. So, this is 16 alright. In general, a constant will have the reserve word constant then the constant name whatever name you want to put over there, colon the type that is the data type or and then this part is optional that you can initialize it, just like the c language you can optionally initialize it. So, there is a generic declaration of a constant similarly we can is alright. So, similarly we can do the same thing with variables.

(Refer Slide Time: 08:37)




If I want to use some variables in my VHDL, I can declare variable then some variable name say I gave opcode is a variable name and its type is now here, you see I say bit vector, this is a type that is predefined in VHDL bit vector. Now here is a vector therefore, I write 3 then down to 0 3 down to 0, I can initialize it again optionally with say 0, 0, 0, 0, if 1, alright. So, what is there in this? So, I am here is a variable name and here is the variable type since it is a vector I have to give the dimension of that vector and if I want, I can like to initialize it.

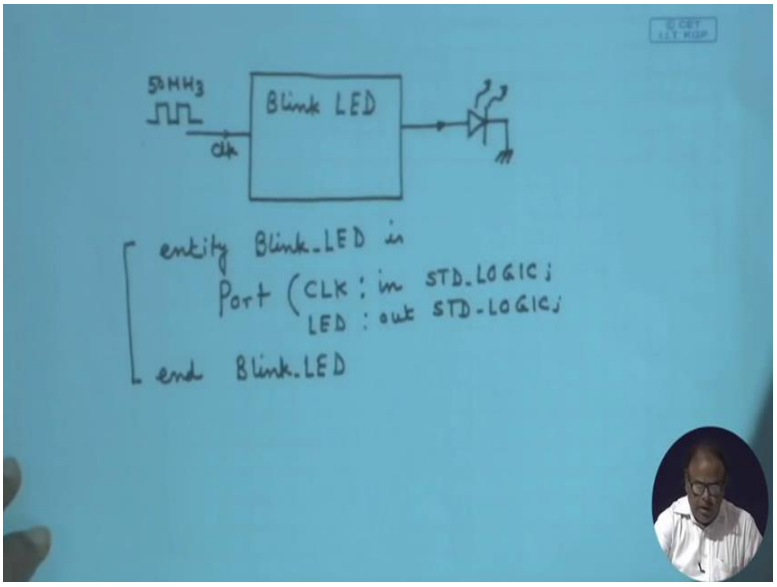
Similarly, let us define another 1; say this is a simpler variable frequency. So, I give the name f r e q to frequency and it is of type integer and I decide that I will not initialize, it is also perfectly alright. So, essentially again a variable declaration is of the form this keyword variable followed by variable name colon type and then if I want, I can optionally initialize it to some value. Now for type you need that if it is a bit vector, I have to give the width of the vector similar to variable also we have got another interesting thing in VHDL that is known as signal what are the key word is signal and say I have got a signal bus ready b r d y is a bus reducing and that is a bit.

I give another example signal output is an integer which can be initialized it 2. So, again just like this there is a signal, signal name, type and some optional initialization. Now

this signal and variable, they look very much the same, but what is the essential difference between signal and variable? A variable just as in a programming language refers to a memory location or a register that where it is stored. So, variable is stored where as the signal is actually representing a piece of wire say a wire is carrying a signal say this is the bus ready b r d y signal. So, this signal will come here will be sensed and when it goes when it comes to this low it is no longer available on this way, on the other hand if I had on the signal support some programming was done and there was some variable frequency say this is the frequency which was set to some value 10 then that remains even after I complete loading of this variable just as it remains.

Now, whenever we are describing some hardware we will need not only the variables, but also we will need some wires which will carry information and those can be very well represented using signals we will see some applications of this later, Now given this there are a lot more to come lot more may come in VHDL description, but even before that let us start developing one simple system since we have done FPGs, let us take a simple system which you want to develop and through a FPGs are you also want to describe it using VHDL.

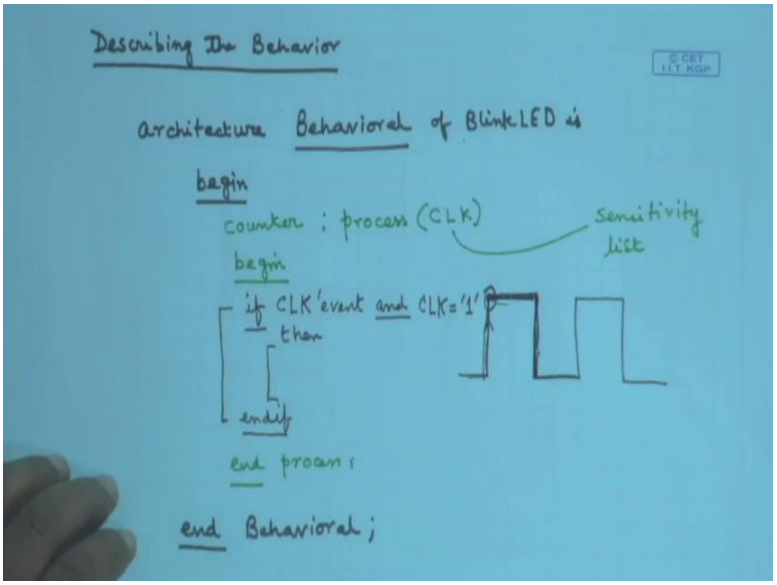
(Refer Slide Time: 14:48)



Here you can see the system that I want to implement conceptually is this a system which will blink an led and the input of the system is a 50 mega hertz clock and I want that this led blinks on for 1 second and then goes off for 1 second, remember the clock is 50 mega hertz. So, in 1 second it is 55 million times, it is coming. I have to glow this led once every second and then off for another second that is my objective and I want to develop this simple system. Now how do I describe this simple system? This is the overall entity. So, I start with describing it as entity blink led I have put this underscore. So, that the variable name is proper I just got ports, what are the ports clock CLK is the clock which is of type int and standard logic clock.

Now, you recall that when I talked about VHDL then we had mentioned that we are including libraries ieee library and the standard logic therefore, we can say and we declared any port we had the mode as well as the data type. So, here I have said that n is a mode and standard logic is the data type similarly led is out this point is my led one is out and the type is static. So, again with this description of the entity int blink led. So, my entity declaration entity description is complete now that just tells me this outer shape that has it will have one input port another output port nothing beyond. Now what I need to do is I have to describe the behavior of this what I was telling you in words that it will take this clock and every for every one second it will glow on and then it will glow off and that thing have to describe now.

(Refer Slide Time: 18:20)




For describing the behavior, VHDL allows us to describe the behavior in this form now what are we specifying we are specifying the architecture of the data bus architecture behavioral of this just a name of blink led is begin and behavior. So, within begin and end behavioral. So, within this part I am describing the behavior now in order to implement to describe its behavior that for every 5 million clocks it will glow once and then it will be off I need a counter therefore, I would describe a counter and what is the count? You can say a counter can be implemented by hardware a counter can be implemented by software, but whatever it is a counter essentially implements a process. So, a counter is a process of this again another keyword of each and every process has got a sensitivity list this is sensitivity list.

Here in this list there is only one element that this process gets sensitive is sensitized by this signal clock when the clock comes then this process will act on the clock is the event to which this process is sensitive then I have this begin and end now what do I want to have? I want to have something like this that the clock is coming like this and as the clock goes up I want to do something and then the clock goes down and again the clock goes up and the clock goes down. I want that the counter works at this rising edge of the clock therefore, within this process you see I am now here was the entity complete sorry there was the entire behavioral description within that I am describing the process now

this process is within this begin and end and it is sensitized with respect to clock in what ways it sensitized let us write that down if I write it down.

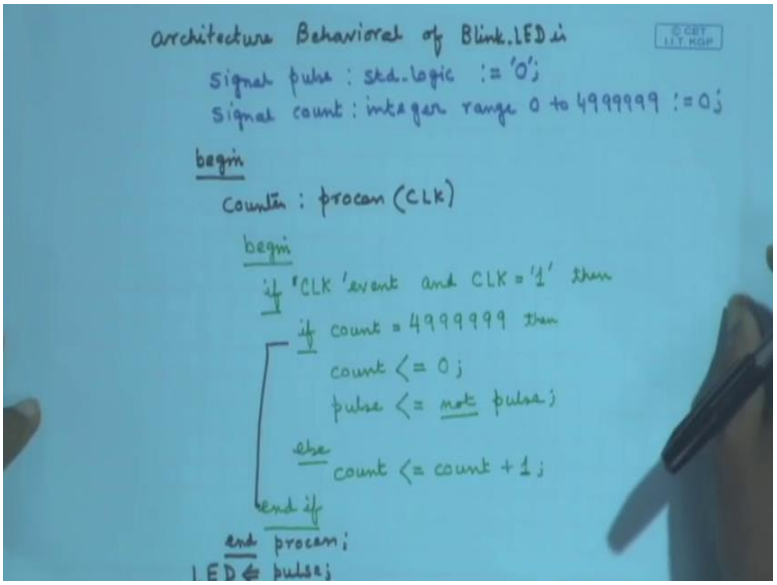
If clock we write it in this way on clock event and CLK is 1 then something end if this is a path if the clock event occurs and the clock is 1; that means, this will remain true clock event has occurred and the clock is one up to this part it will remain at this point what is happening it is again a clock event, but the clock is 0 immediately therefore, whatever I write here then whatever I write here will be executed only up to this event has occurred at the rising it and this condition is hold is true, event is occurring also here clock age event this means the clock is event. So, the clock age event is occurring here as well as here, but this will see later that this is a guard condition, this is the condition that clock is one is no longer holding here. What is happening here is- clock event here clock event and clock is 0 that is Ls part if at all.

Student: (Refer Time: 23:50).

No, but it will start to get into this at this because it is instantaneous. So, I have got this part will execute, this is true up to this part, this condition is true, but it will get into this immediately at this point. This part will be ending, we will build up this. So, there is only one part of it right this part it is not a loop we are not looping here we are just getting in here and executing clock is one process is say this process a process will have different parts of the body; I am right now describing the process the process can do this thing or that thing now this part will be executed on the rising it will be initiated in the rising edge of the clock and will be working as long as this is 1.

It is initiated at this point as you can see and we will as soon as this comes out then my Ls part will be breaking we will write this again. So, this is how we gradually I am trying to build up the VHDL description.

(Refer Slide Time: 25:39)




Now, forget about whatever is written here, I will just compare with whatever I had written in the earlier slide here and we will continue from there. So, what we had written was this black part architecture behavioral of blink led is then begin and behavior then begin and behavioral this whole part was there all those things were not there. This I will introduce gradually then this was there the description of a counter process and in begin what we did is on clock event and clock I will be doing something, what I will do? I did not say, now you can imagine and say that when what am I going to do in a counter whenever it happens one then I will count increment; the count and I will not increment the count on this event, I will again increment the count on this event. So, every time this condition is satisfied, I will up count.

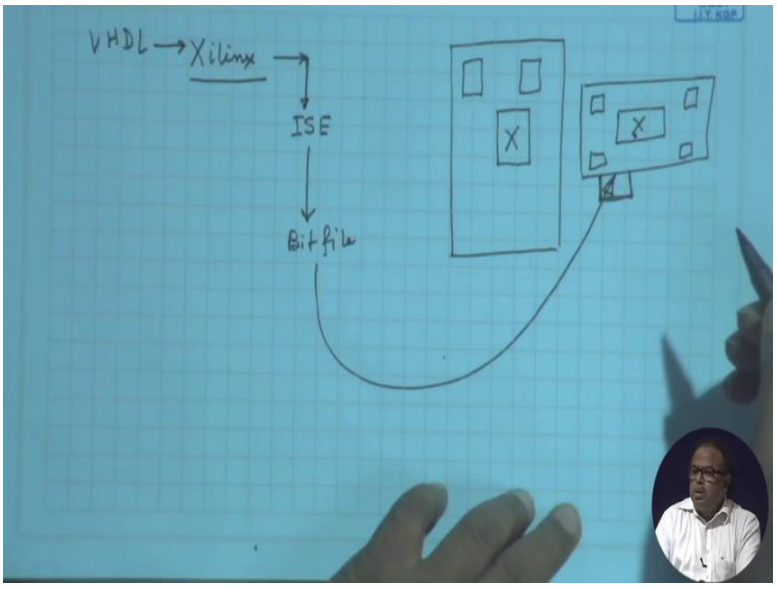
If count is as I am counting up here I have the counter I have taken a variable count and this is not a variable by the way it could be a variable, but here we have done which is the signal which has got an integer range up to this now this count is being initialized to 0 as soon it is counting up if the count is above this 50000 then the count will be made 0 and I will be sending a pulse. Therefore I will right now, what is happening is I have declared pulse to be a single because I have to send that signal to the output of the I mean led output so, but the led output is not known to me right. So, that I declare a signal pulse which is standard logic and initialize to 0. So, right now the pulse is 0 now count is 0 as

soon count is this is this condition is initially not satisfied. So, I am not generating the pulse I am simply coming here on the clock event I am incrementing the count and going up on clock event again when I will again come here enter it on clock event and I will go on counting up.

When the count becomes 0 to 49 9, this 5 million, so after 5 million counts, I will make the count to be 0 is it with all of you here the count will you again made 0 and pulse which was. So, 0 will not now be made will be toggled not of pulse; that means, I am getting a one at the output and this is the process of count the counter process is counting up to 55 million 5 million clocks then it is sending the pulse and resetting the count to 0 and at that time it is coming out of this and then that pulse is being sent to led.

Now, this led where is this led declared led is been declared as a port in the entity description right because this clock has also been described in the entity description that is the port. So, this led goes there I mean the pulse goes to the led and that is the end of my behavior alright. So, this is how I write a very simple VHDL code now once I write this is VHDL code what am I supposed to do after I write this VHDL code I will be simply putting that.

(Refer Slide Time: 30:52)

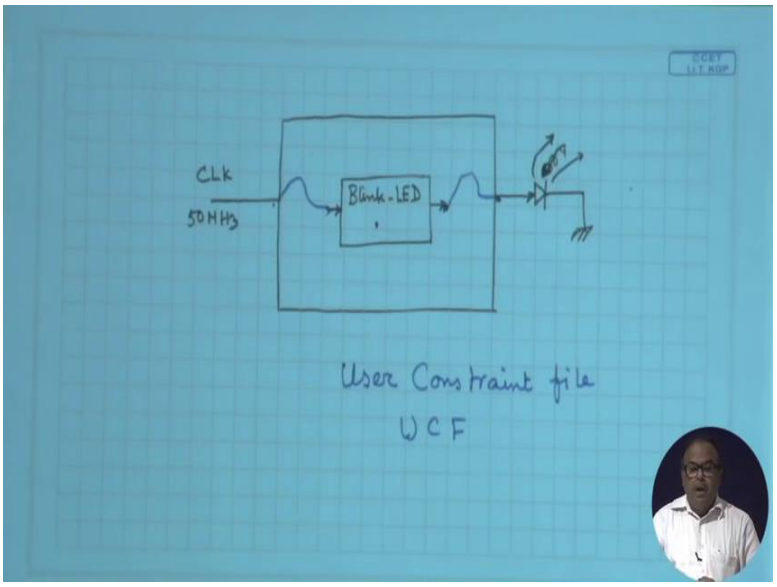


I have got there are tools say for example, I am using Xilinx and Xilinx has got a particular variant of Xilinx maybe and for every variant on just for Xilinx there can be different boards right which has all those boards will have Xilinx here, but the other peripherals may differ.

Some other board see the mercury board if you take it will also have the same Xilinx, but maybe the other peripherals will be there. So, depending on what peripherals I have I will all have to instantiate the Xilinx programming tool? So, that Xilinx programming tool will have some separate tutorials on this tutorial lectures on this and there are say one environment is the ISC using which is in the last class we are saying that we compile it for this particular FPGA kit that I am using and this VHDL code will be converted to the bit file.

So, I have got the VHDL or the VLR description and that bit file is loaded into this there are each of them have got the USB port or j tag port using which I can upload this program over here and this becomes on. So, that is the step we carry out we first write it in some high level land high level means synthesizable high level language which is VHDL Verilog is also used. So, using that I compile it I describe the behavior and then put it in.

(Refer Slide Time: 33:04)

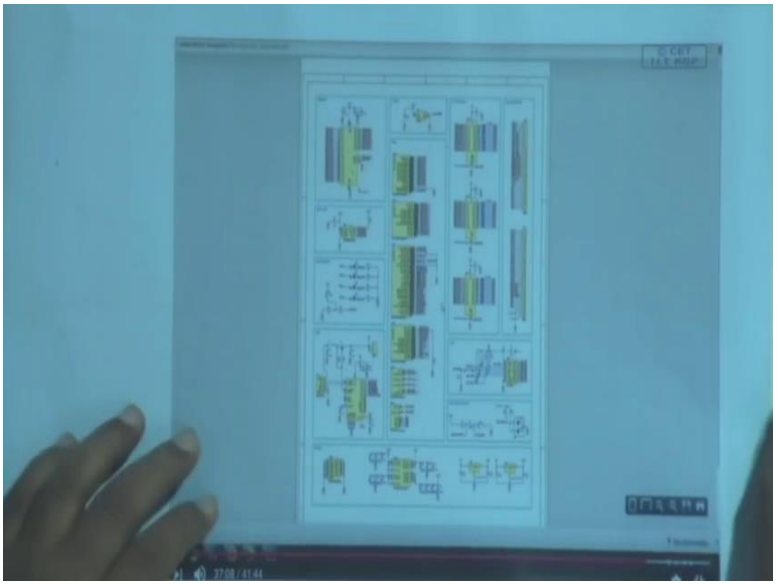


Now after I put this, whatever I have done now is something like this that here is my FPGA board and I have got my Xilinx here and after I loaded it I got a blink led has been loaded here now this blink led has got a clock and an output and I have got the led is actually connected here. So, led is rise.

So, we write like this alright and the clock is coming here clock is a 50 mega hertz clock my program has populated this part now it is my task to establish the interconnection between this point this one does not know that this particular input will have to be connected to this or whatever output that the pulse is going giving will come here that led that led in my program was nothing but a port to what port to the blink led entity this led this led is a port and port to the blink led entity therefore, this led is this point because the, but I have to establish the correspondence between this point in this for that. In order to do that we need to have I was telling them the other earlier class user constraint file.

Now this is user constraint file allows the this the tools that I was using the tool that was using will allow me give me an interface by which I can specify using a drop box and all those specify the different interconnections, now once I specify that what I am actually doing is I am establishing a correspondence between the schema schematic what is a schematic this will be a little clumsy maybe.

(Refer Slide Time: 35:50)



Because say a typical schematic is shown here where you can see this is FPGA among this there are different components and nothing is readable to you now here are the boards. So, I need to amplify, but if you are carefully you can see some led's here right now these led's are connected now if I amplify this if I then I can show a particular path how much visible is that is better that here you see I have got the led one 0 one 2 different led's are there, there are and here if you concentrate you can see here there is a interconnect point where this is a led 0 if I am figuring led 0 then in that UCF file I have to establish a correspondence because this led 0 is connected to this line. So, I have to establish the output of my Xilinx to whatever the number is this one that I can do just simply selecting on that.

That is why hands on is very much useful, but once I do this, then my interconnections are established. So, what I have now he is not only the blink led entity, but along with that I have establish the connection to these ports. I have established the connection to these ports and (Refer Time: 37:30). So, with this we conclude our first phase of discussion on FPGAs this will be followed later by a couple of tutorial lectures on FPGA programming where a particular tool will be followed, but it is not mandatory that you will always get that tool handled.

So, accordingly whenever you are given the responsibility of developing something on any FPGA you will have to adapt to that particular tool and you have to understand it.

Thank you.