Digital System Design

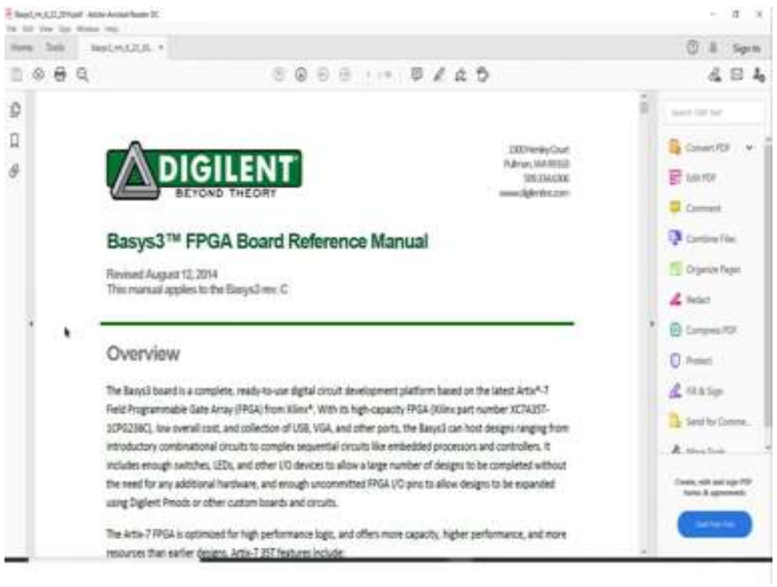
Professor Neeraj Goel
Department of Computer Science Engineering
Indian Institute of Technology, Ropar
FPGA - Design flow

Hello again, so now what we would like to do in this lecture is we would like to demonstrate onto an FPGA device how a simple circuit will work. The example, we are taking is a counter which would be displayed onto a FPGA board.

(Refer Slide Time: 0:34)

So, the FPGA board we are taking is a basis board, so you can see here in the basis board there are switches here and there are some LEDs and you can see there are seven segment displays, there are some buttons here and this is the FPGA chip.

(Refer Slide Time: 1:02)



1300 Henley Court
Pullman, WA 99163
509.334.6306
www.digilentinc.com

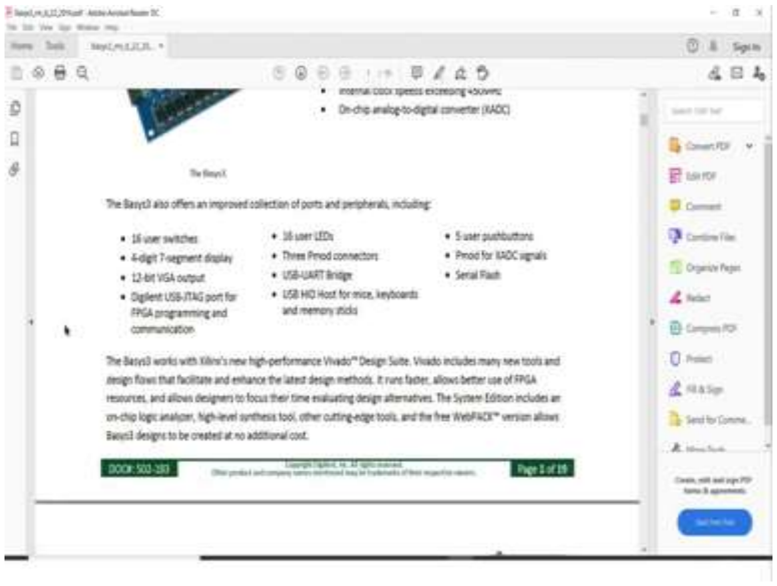
Basys3™ FPGA Board Reference Manual

Revised August 12, 2014
This manual applies to the Basys3 rev. C

Overview

The Basys3 board is a complete, ready-to-use digital circuit development platform based on the latest Artix®-7 Field Programmable Gate Array (FPGA) from Xilinx®. With its high-capacity FPGA (Xilinx part number XC7A35T-1CPG236C), low overall cost, and collection of USB, VGA, and other ports, the Basys3 can host designs ranging from introductory combinational circuits to complex sequential circuits like embedded processors and controllers. It includes enough switches, LEDs, and other I/O devices to allow a large number of designs to be completed without the need for any additional hardware, and enough uncommitted FPGA I/O pins to allow designs to be expanded using Digilent Pmods or other custom boards and circuits.

The Artix-7 FPGA is optimized for high performance logic, and offers more capacity, higher performance, and more resources than earlier designs. Artix-7 35T features include:



- Internal clock speeds exceeding 450MHz
- On-chip analog-to-digital converter (XADC)

The Basys3.

The Basys3 also offers an improved collection of ports and peripherals, including:

- 16 user switches
- 4-digit 7-segment display
- 12-bit VGA output
- Digilent USB-JTAG port for FPGA programming and communication
- 16 user LEDs
- Three Pmod connectors
- USB-UART Bridge
- USB HID Host for mice, keyboards and memory sticks
- 5 user pushbuttons
- Pmod for XADC signals
- Serial Flash

The Basys3 works with Xilinx's new high-performance Vivado™ Design Suite. Vivado includes many new tools and design flows that facilitate and enhance the latest design methods. It runs faster, allows better use of FPGA resources, and allows designers to focus their time evaluating design alternatives. The System Edition includes an on-chip logic analyzer, high-level synthesis tool, other cutting-edge tools, and the free WebPACK™ version allows Basys3 designs to be created at no additional cost.

DOC#: 502-183 Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners. Page 1 of 19

So, to see it onto the yeah, this is the reference manual of the basis three board so it comes from Digilent and it is a good FPGA board for practicing basic design for teaching purpose.

(Refer Slide Time: 1:16)

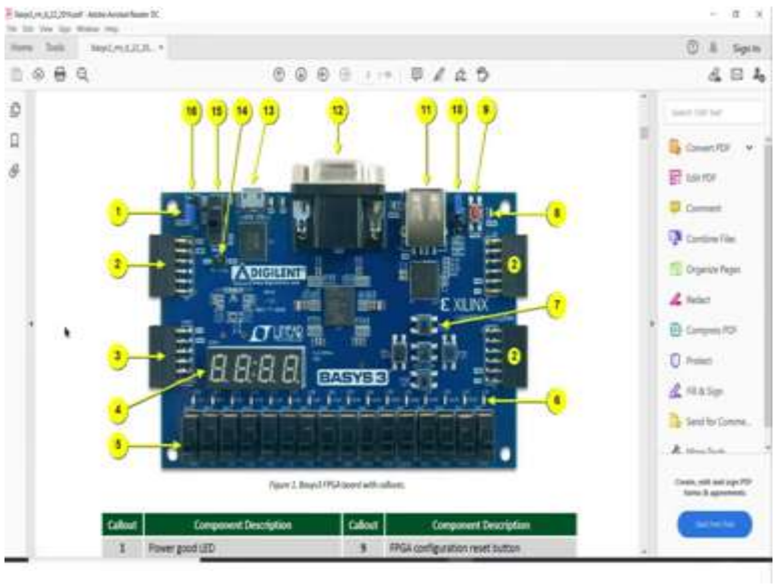


Figure 2. Basys3 FPGA board with callouts.

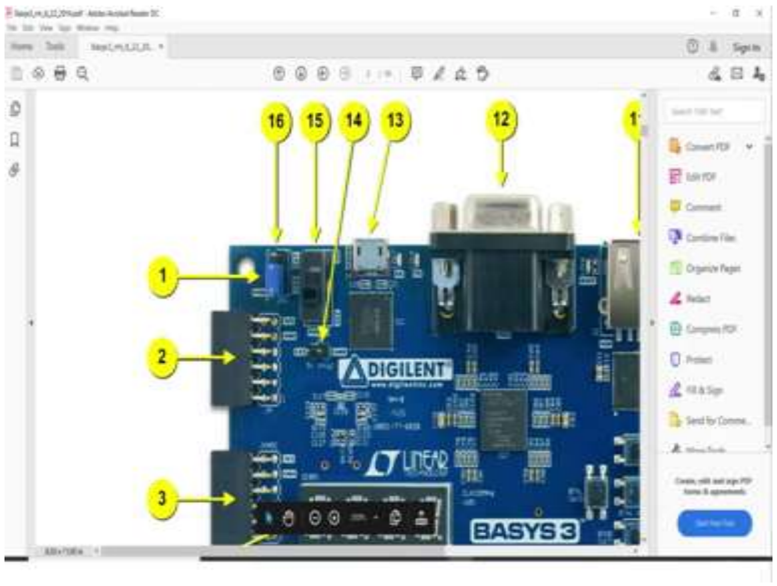
Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button

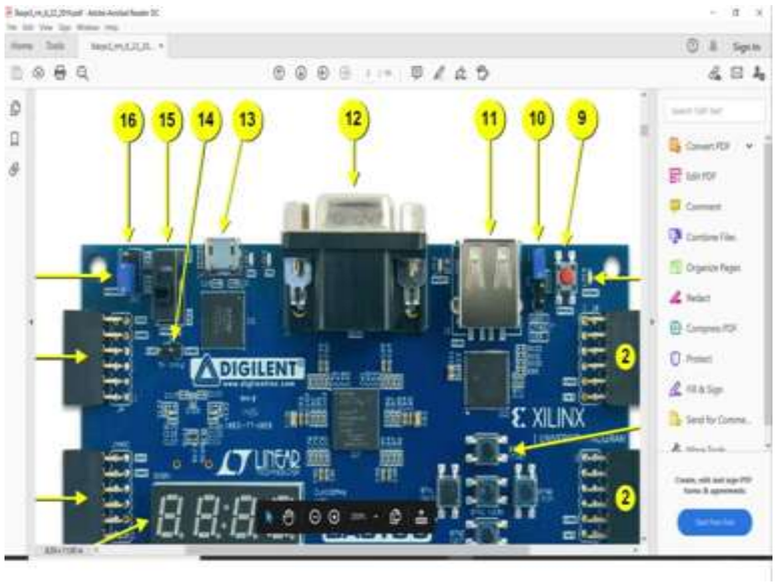
So, this is the block diagram here. So, what you will see multiple things, so these are the switches which I have shown you and these are the LEDs and these are the LEDs and you see these are the seven segment displays which can be used and the input can be either taken using these switches or using these buttons.

Now, you also see there are several inputs like we have USB input here, there is a Ethernet port here and this is a VGA port so because of this VGA port and Ethernet port and USB connections this effectively this because of this it can effectively work like a computer also, so the output can be displayed onto a monitor and the keyboard input or a mouse input can be taken using this USB connection.

And these are the general input output connections which can be connected to any of these sensors or actuators, etc. So, this particular part help us to program whether it is programmed using just I will give a more.

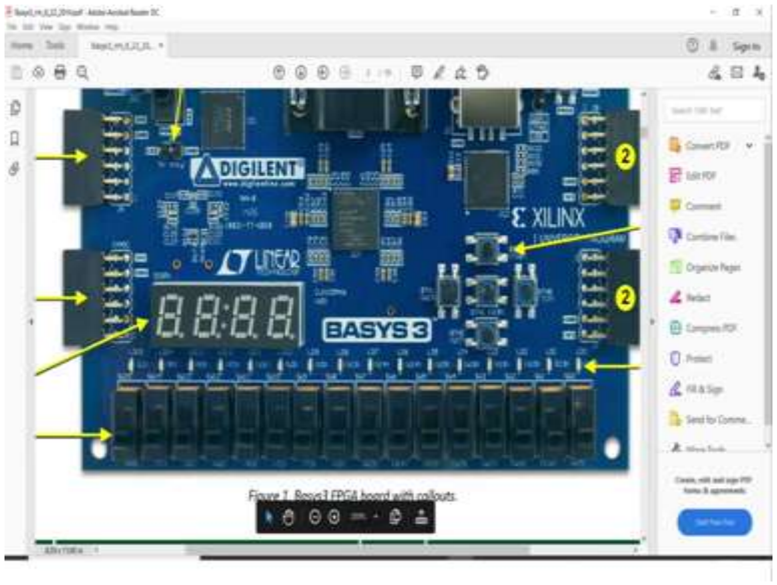
(Refer Slide Time: 2:32)





So here, it this particular plug it helps us to give whether the power is using external or using USB. And similarly, the connections whether yeah, so there are three ways to program this board whether Q SPI, J Tag or USB so currently it is we have put a connection so that it can be programmed using USB. So, in our board we have made this connection using USB so that we can program using the USB, so this is the USB cable which can help us in connecting.

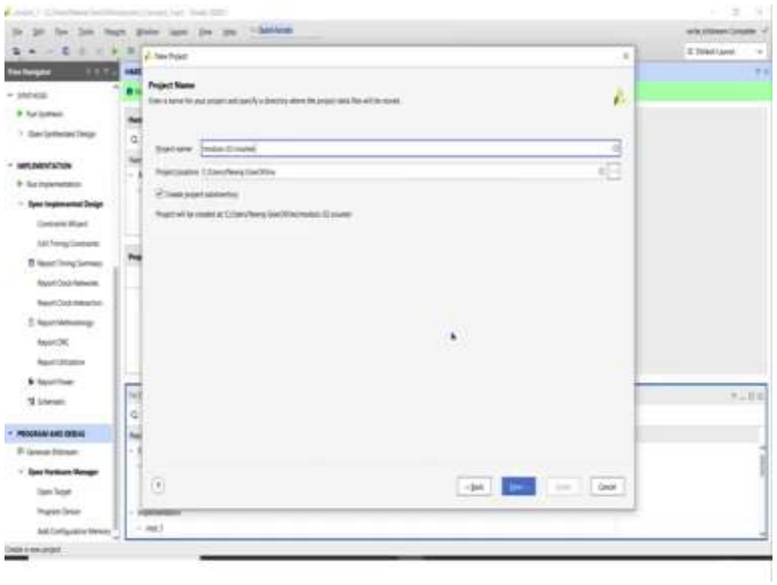
(Refer Slide Time: 3:19)



The other thing which I would like to point out here is the numbers which are written here, so for example this is the LED 0 and here E19 is written and for LED0 u16, so this is the u16 number is written LED 1 u19 is written. So, what is this u19, u16, et cetera so this is our FPGA chip, on the FPGA chip there would be various pins, these pins numbers are written here that means there would be several general purpose pins these.

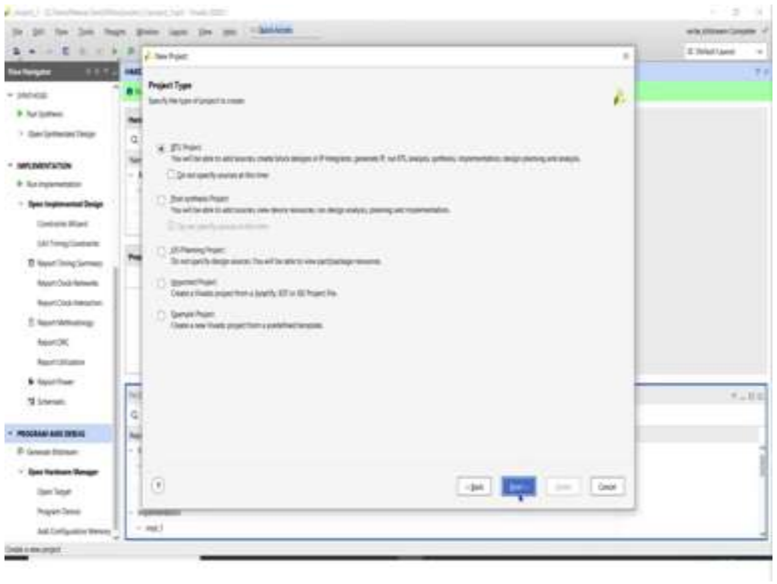
So, the pins of FPGAs are connected to let us say E19, E19 is a pin number on to this artrex FPGA chip and that E19 pin is connected to this LED. So, after looking at this board schematic now let us go back into design and see how the design process will work.

(Refer Slide Time: 4:34)


New Project

Project Name

Project name: modulo 32 counter


New Project

Project Type

RTL Project

Post-synthesis Project

I/O Planning Project

Imported Project

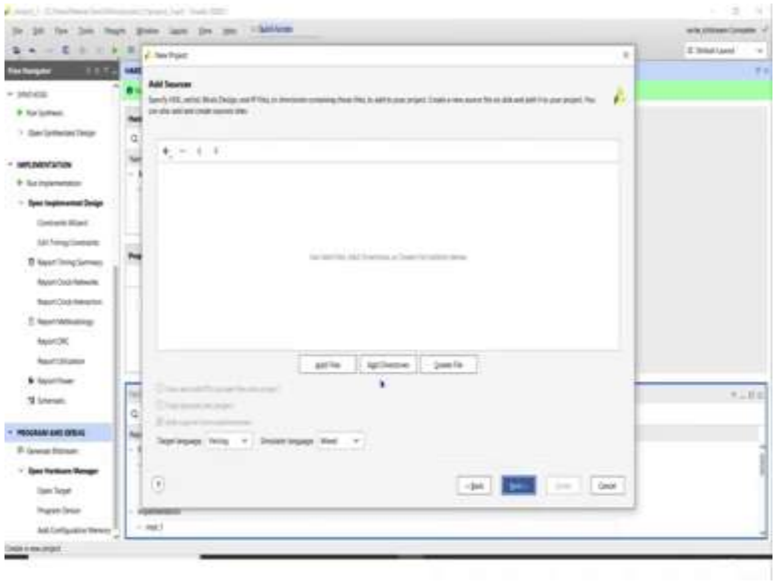
Example Project

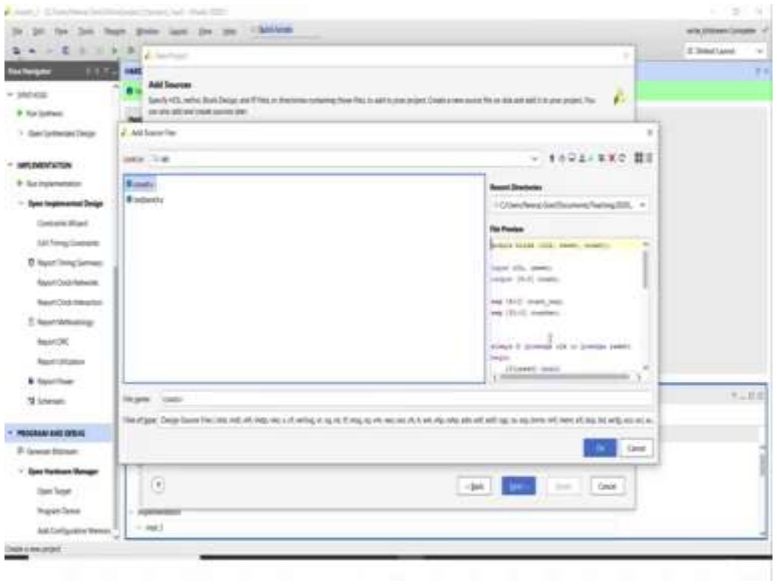
So, we will use because this board is coming from Xilinx, so we have to use Xilinxs software which is called Xilinxs Vivado. So this Xilinxs Vivado is actually a free software which can be used so here we are creating a new project and when we create a new project so the let us say we because we would like to design a counter, so let us say we want to design a counter.

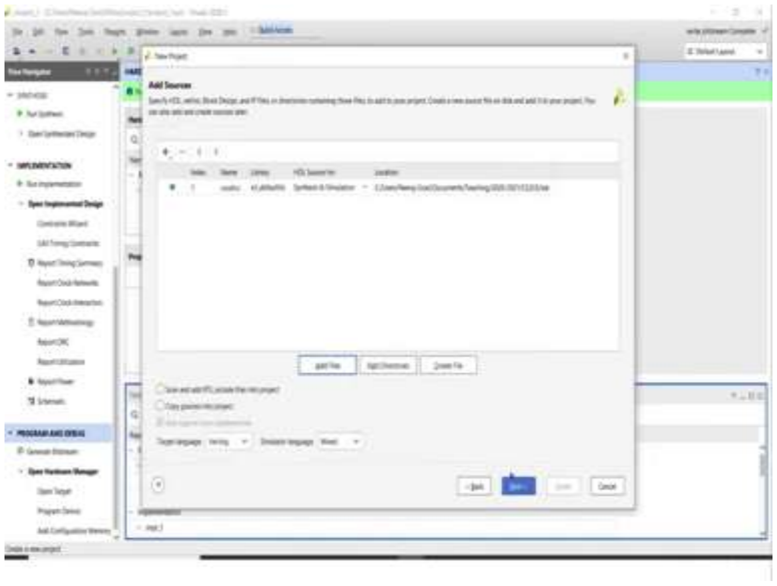
So, this is actually 15 modulo 32 counter. So, let us design a modulo 32 counter and then it will ask us that whether it is a RTL project, post synthesis, IO planning, imported project or example project. So, I will tell you, quickly tell you the difference. So, RTL project is now since we are designing our own project and this will come under RTL project category in post synthesis that means we have already done design synthesis then after synthesis we can use that synthesis and we can directly start from there.

Imported project means that you have done project using some other software like simplify XXT or ISC and then we have to, we want to do this. Example projects are there certain templates which has been predefined in this software. So, if we would like to start with the example project then we can start. So, this one is a is a RTL project.

(Refer Slide Time: 6:05)






New Project

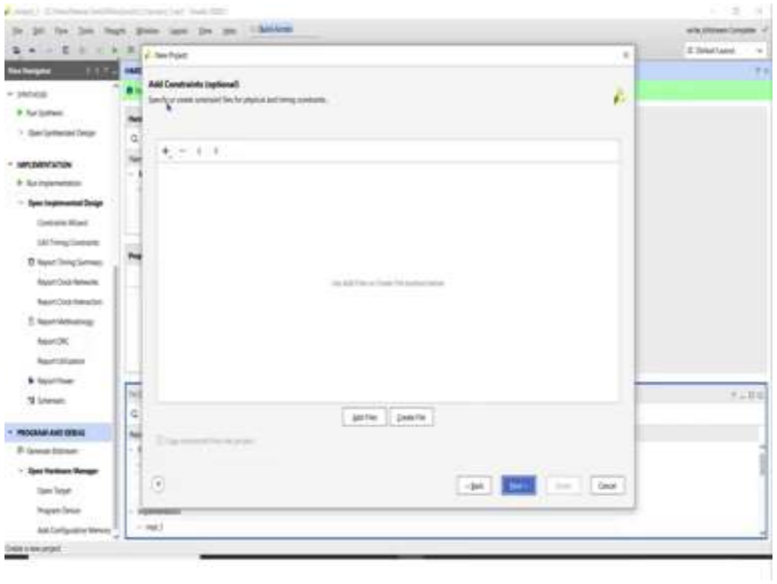
Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Index	Name	Library	HDL Source for	Location
1	count.v	xil_defaultlib	Synthesis & Simulation	[illegible]

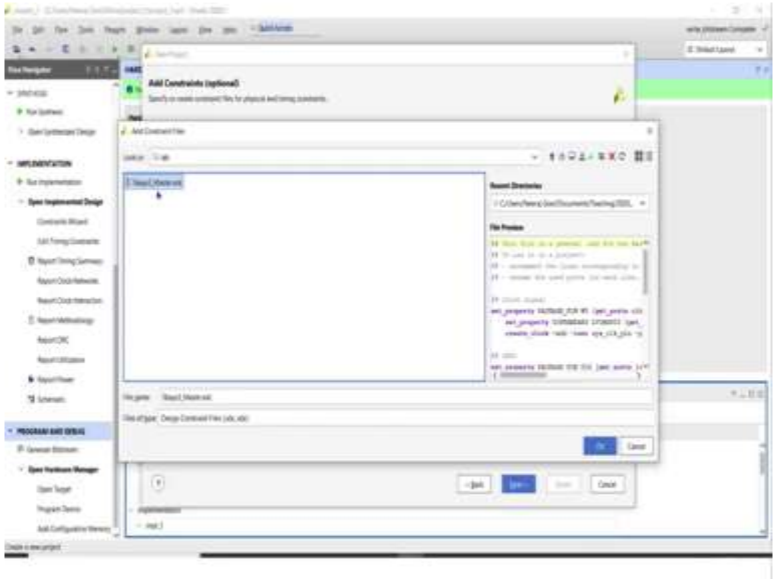
So, in the RTL project now we have to specify what is the sources. So, the sources I have already created into very low, so there is a counter of here I have to, so there I already created this count dot v, so that we will import and then we will go to the next step.

(Refer Slide Time: 6:36)


New Project

Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.



Add Constraints (optional)

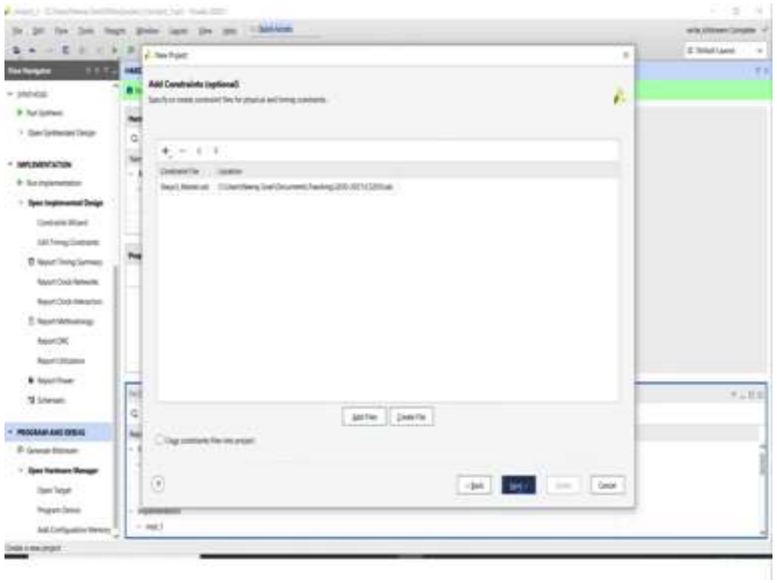
Specify or create constraint files for physical and timing constraints.

Add Constraint File

File Preview

File name: Basys3_Master.xdc

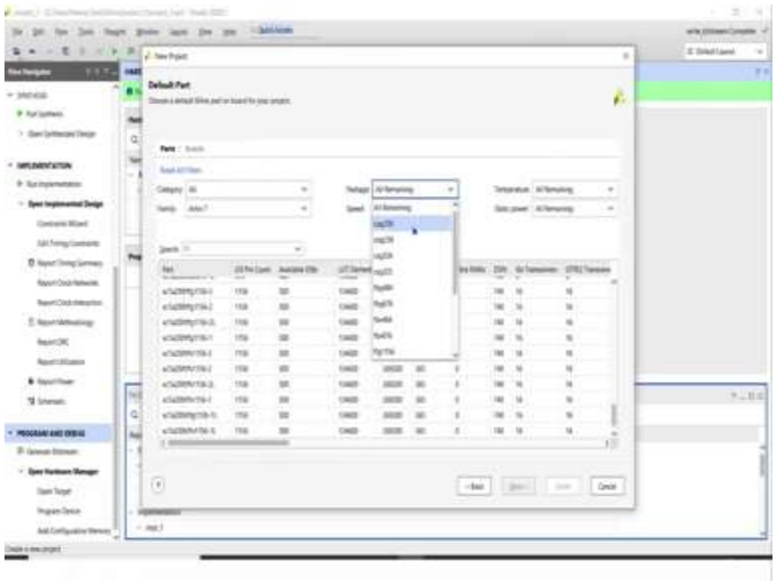
Files of type: Design Constraint Files (xdc, sdc)



Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.

Constraint File	Location
Basys3_Master.xdc	[illegible]


New Project

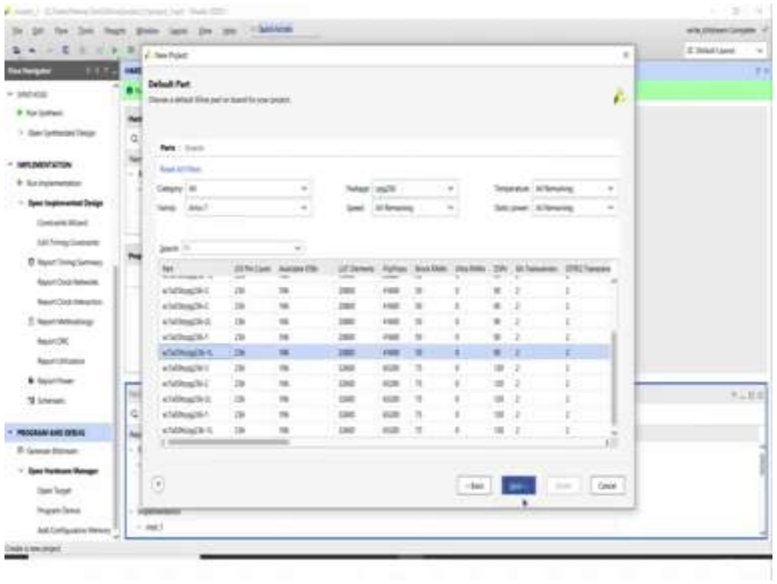
Default Part

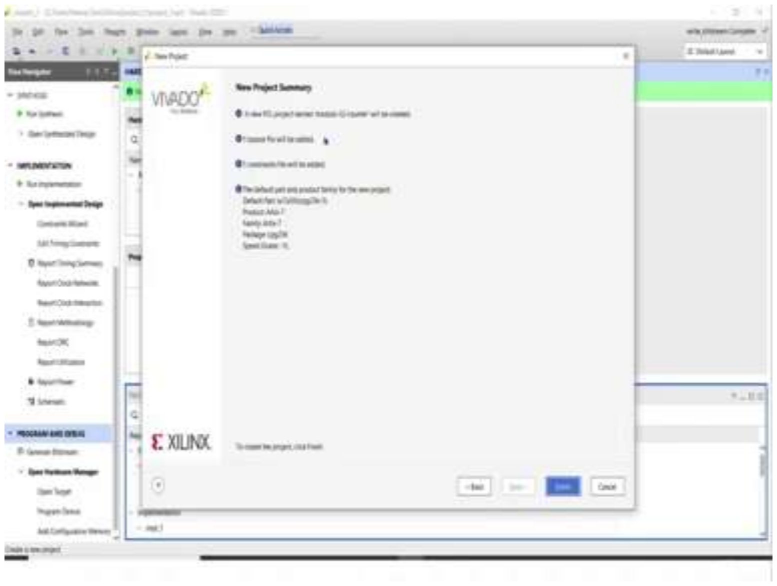
Yes, so in the next step we have to add constraint file, so what is this constraint file? I will come to this little, in a little while, so this constraint file will tell that which input output is connected to which particular pin of the FPGA. So, I have already created this constraint file, let us add this also into the design and then the next step, the next step is finding which particular part or which particular FPGA board.

So, now how do we find out that, so if we see this FPGA board here it is written that which particular is the part number it says Xilinxs, Artix 7, and xc7a35t, so this part number we will choose and the how to choose this part number so first thing we know that it artix part so we say the family is artix7.

And then we further select the package. So, if we see from the specification the package is 236 cpg 236 so that give us the limited options, so from the limited option the one which is ours one is xc7a35 ti cpg. So, we selected this part.

(Refer Slide Time: 8:14)




New Project Summary

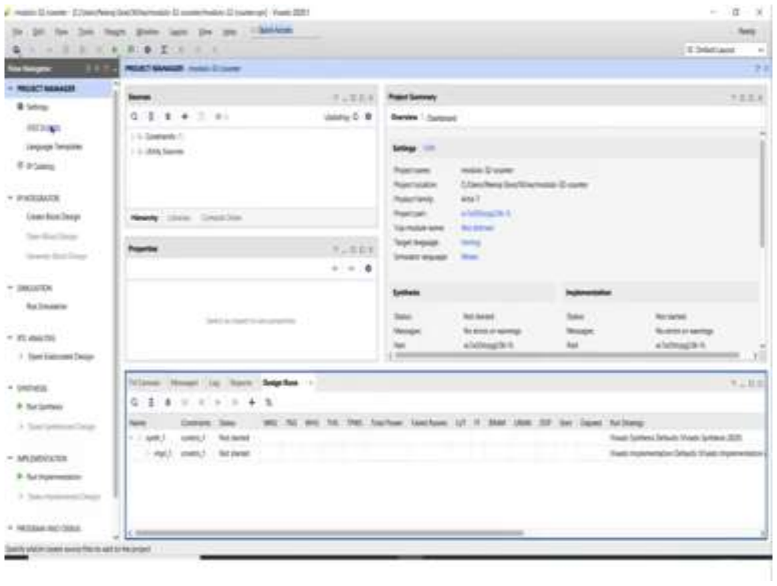
A new RTL project named [illegible] will be created.

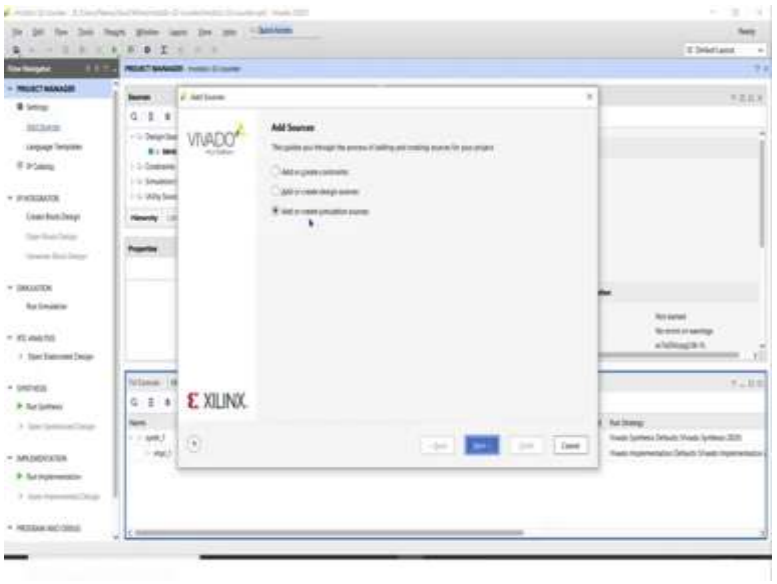
1 source file will be added.

1 constraints file will be added.

The default part and product family for the new project:
Default Part: xc7a35tcpg236-1L
Product: Artix-7
Family: Artix-7
Package: cpg236
Speed Grade: -1L

To create the project, click Finish





Add Sources

This guides you through the process of adding and creating sources for your project

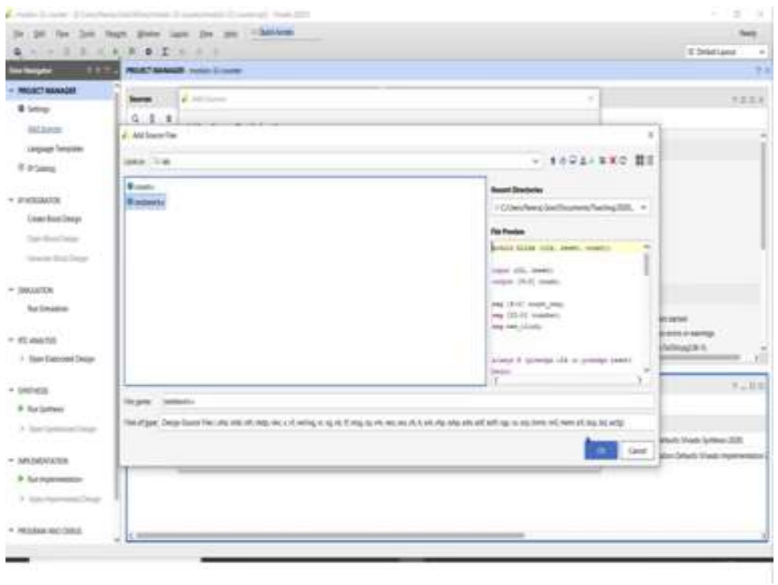
Add or create constraints

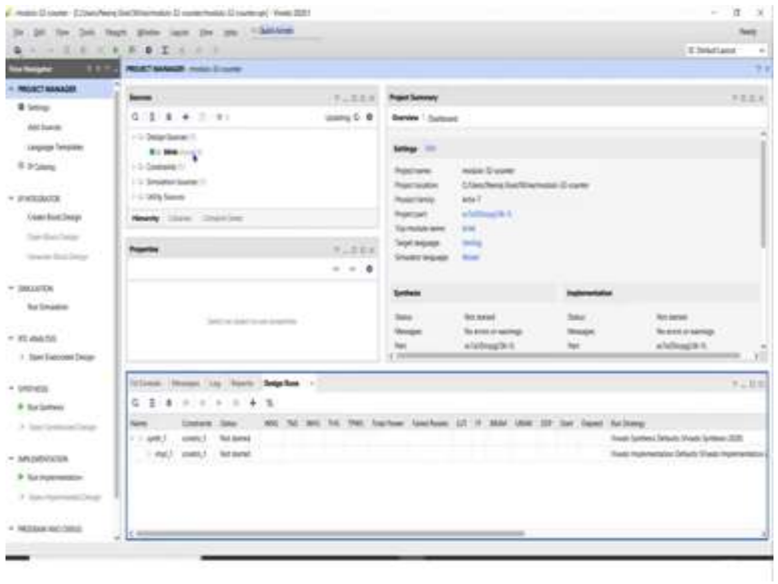
Add or create design sources

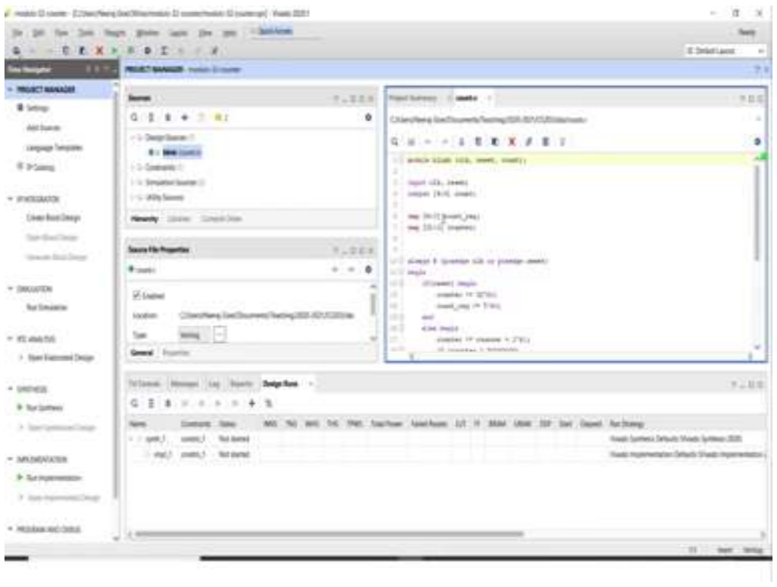
[✓ Add or create simulation sources]

And now we can finish creating this project. Now, we need to add our testbench dot v also so let us add one more source which is called simulation sources, so in the simulation sources we have added another file which is testbench dot v so this is we have done.

(Refer Slide Time: 8:41)







```
module blink (clk, reset, count);

input clk, reset;
output [4:0] count;

reg [4:0] count_reg;
reg [31:0] counter;


always @ (posedge clk or posedge reset)
begin
    if(reset) begin
        counter <= 32'b0;
        count_reg <= 5'b0;
    end
    else begin
        counter <= counter + 1'b1;
```

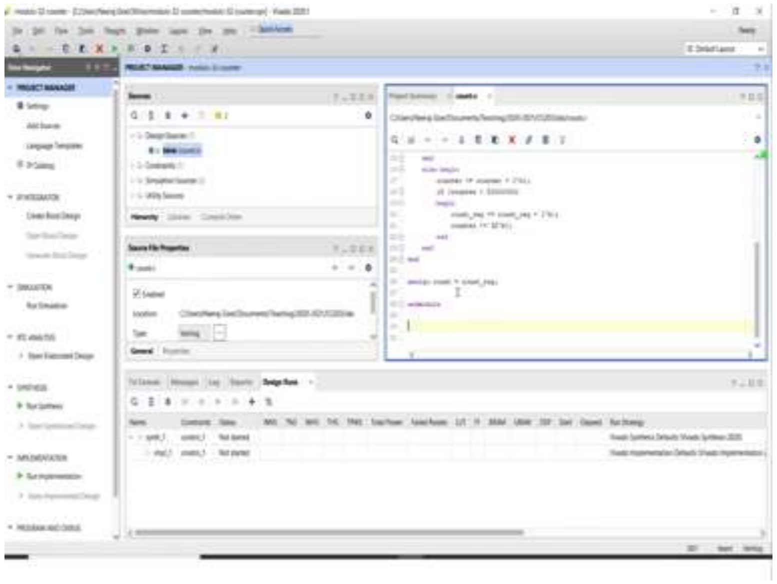
Now, let us also see what is this very log file, so that we can, this the module name is called blink and it has three inputs clock, reset and count, the input is clock and reset, output is 5 bit count so that is a 32 bit counter, so sorry 32, modulo 32 counter. And now we see the count we have created two more registers count reg as well as a 32 bit counter.

Now, you see this design here we are saying at always at every edge of the clock and reset we are making counter equal to 0 and counter is being incremented at every clock cycle, so if reset is not there which will not be there so reset is not there, counter is incremented every every clock cycle.

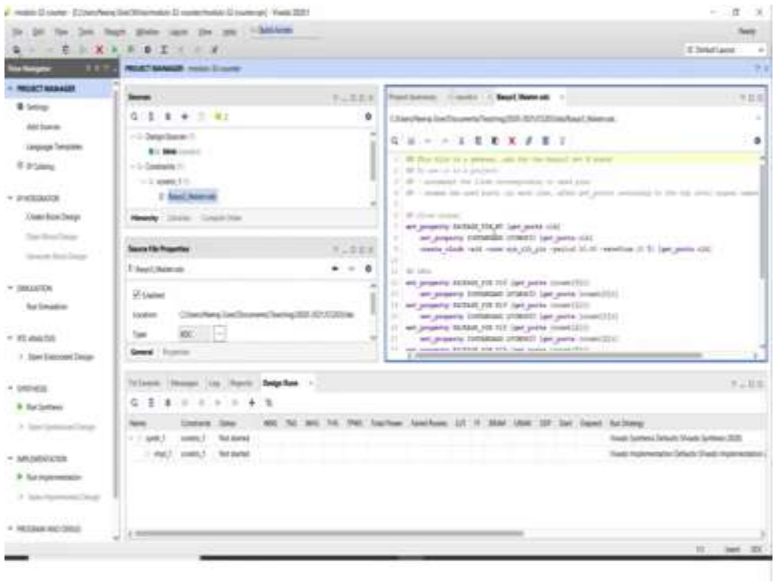
So, now because our clock is going to be very very fast, so typical value of the clock is 100 megahertz, so 100 megahertz is very fast so if that will run will not be able to see what is happening, so we have to slow down that. So, if counter value increases beyond certain limit then we are saying the counter has to become 0 and the final count rank which we would like to give as output would be incremented by 1. So, this incremented by one would happen only when the counter is more than this large value.

Now, then we are assigning count which is our output to count reg, so count reg is assigned to counter, so this is a very simple design suddenly at RTA level and now let us also look at the constraint file.

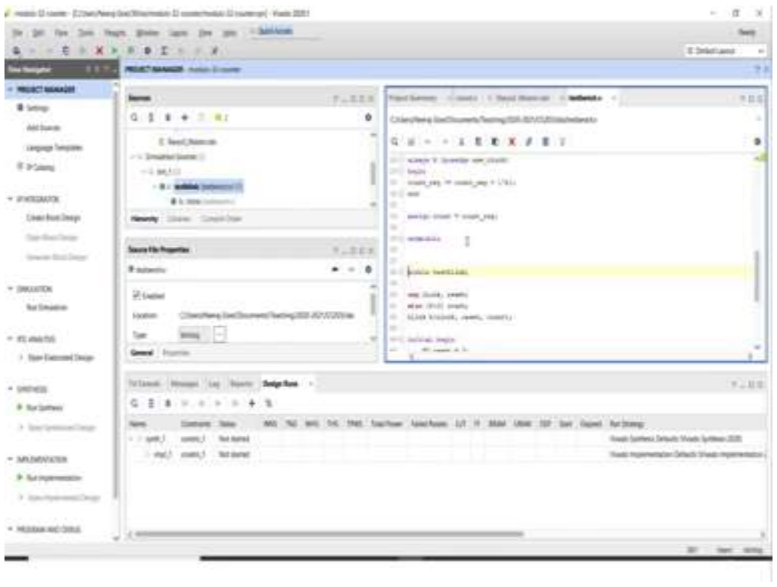
(Refer Slide Time: 10:31)



```
        end
        else begin
            counter <= counter + 1'b1;
            if (counter > 50000000)
            begin
                count_reg <= count_reg + 1'b1;
                counter <= 32'b0;
            end
        end
end

assign count = count_reg;

endmodule
```



```
## This file is a general .xdc for the Basys3 rev B board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project

## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
    set_property IOSTANDARD LVCMOS33 [get_ports clk]
    create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## LEDs
set_property PACKAGE_PIN U16 [get_ports {count[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {count[0]}]
set_property PACKAGE_PIN E19 [get_ports {count[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {count[1]}]
set_property PACKAGE_PIN U19 [get_ports {count[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {count[2]}]
```



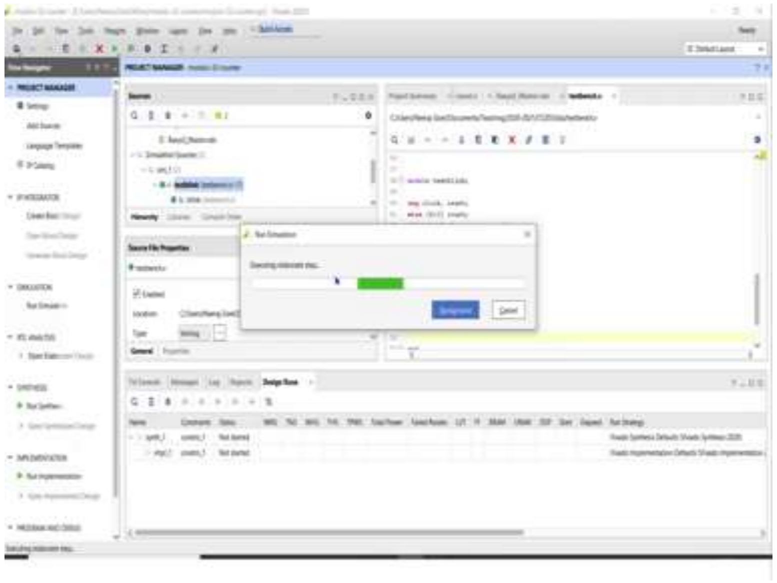
So, in the constraint file if we see the constraint file is specifying that my clock has to be connected to w5 pin of the package and here we are saying that the clock is actually having a period of 10 nanoseconds so that means 100 megahertz clock and we are saying count 0 pin is to be connected with u16 and count 1 pin is to be connected with E19, so on so forth, so basically this is LED 0, LED 1, LED 2, LED 3.

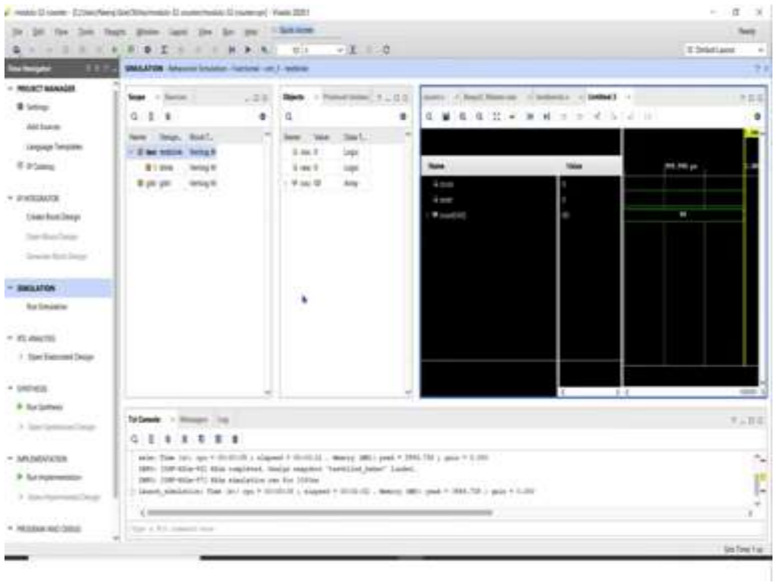
Reset is connected to a central button so this so that whenever we would like to reset we can press the central button. So, this is how we are binding our input and output to the input output which is specified on the FPGA port. In the test bench, the test bench has, in the test bench we are simply giving the initial basically initially we are seeing reset equal to 0, then clock equal to 0, and then forever basically we are saying clock equal to not clock.

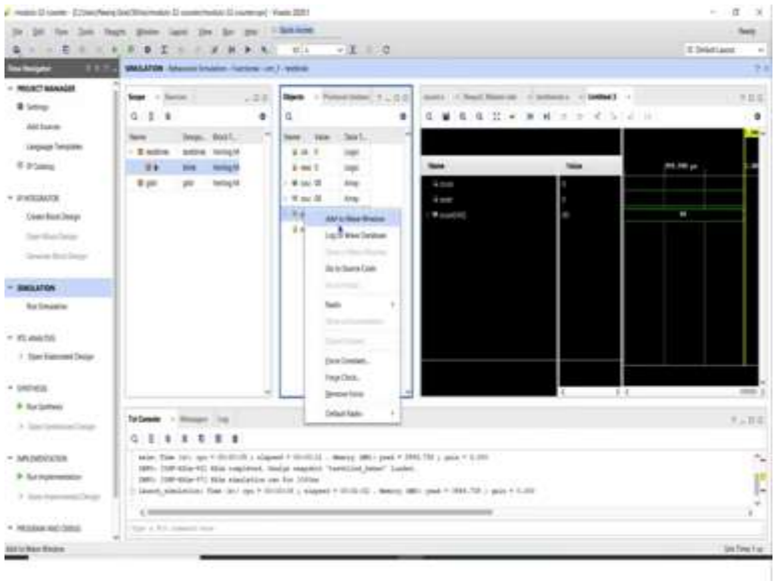
So, this clock it equal to not clock would keep on happening but a reset would happen only in the initial stage and after that reset will remain 0. So, once it is done we have to first see, we have to run the simulation to see whether things are correct or not, so let us, the whole process is in the same fashion so basically first is the designing of the in the left pane we can see initially we have to create the project and after that we have to the second step is simulation so let us do the simulation part and we can do only behavior simulation that means functional simulation.

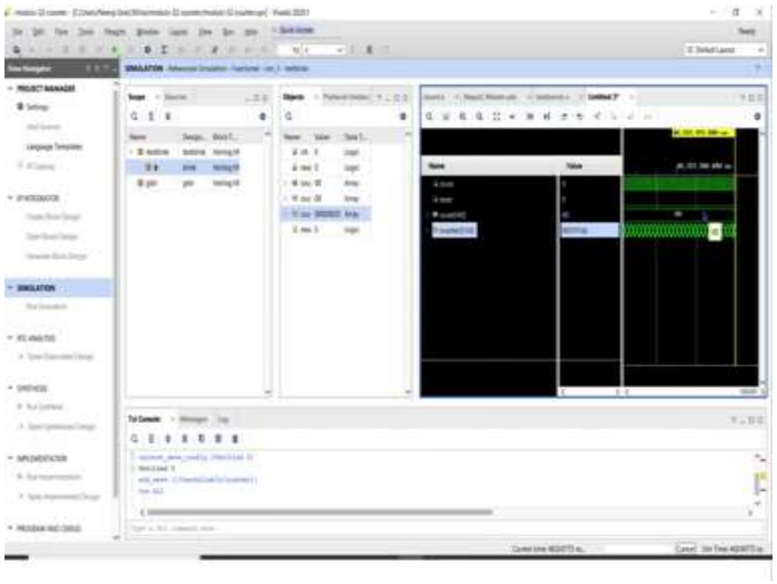
So, this functional simulation would help us to check whether our circuit is correct or not. So, during the simulation process you can also see that it is doing execution, it is doing elaboration and so after doing elaboration step then it will start the simulation phase.

(Refer Slide Time: 13:11)









So, now you see this panel automatically coming which is saying test and then this is the b block, in the b block we have a counter array, so this is our test block b and there is a counter so let us also add this to our you remember that there are two things one is count which is a five bit counter the another is a 32 bit counter so we add this also to a wave window.

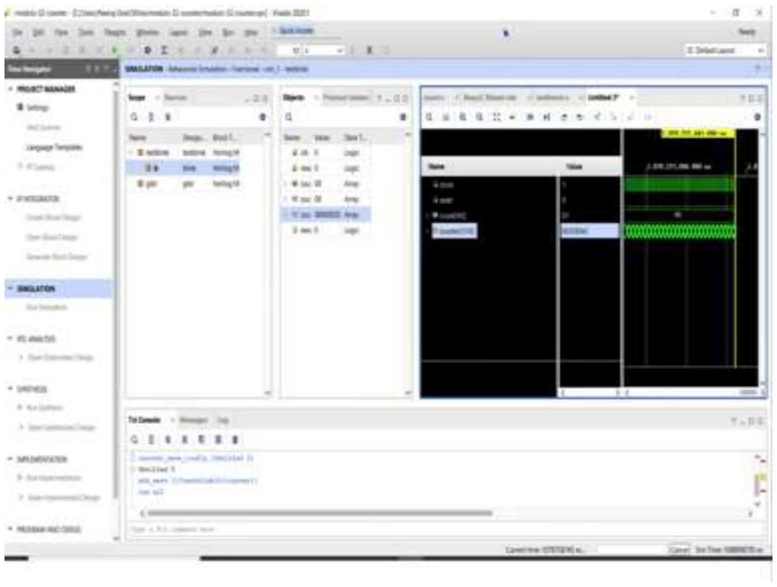
So, if we see if we try to run now we have to run it for a while then only we can see something which is happening, so during this minus using minus or plus we can see that how values are being changed. So, this clock is changing every time and similarly this value of count is also being updated every time.

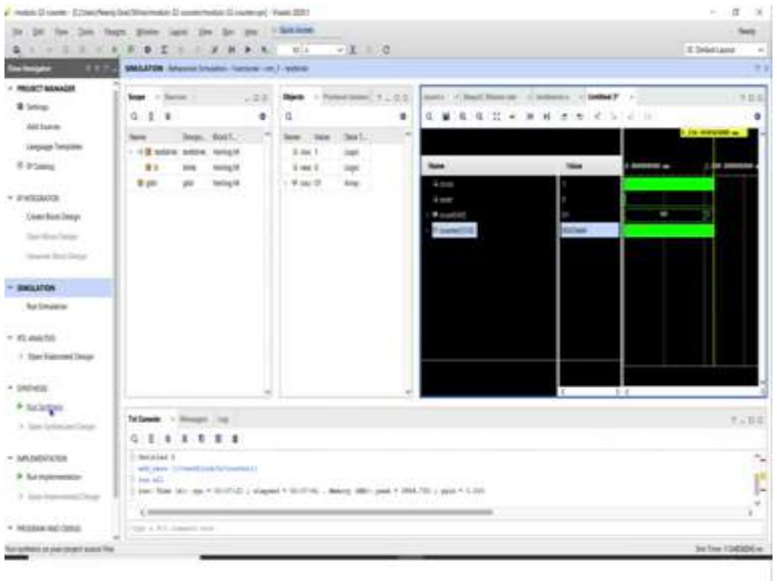
So, now I am running it for 10 seconds because the value of count will increase at every clock period but the value of count which is a 5 bit count can only change after that the counter value is large enough that it can change the value of count. So, will not run the complete simulation but at least we will reach to certain stage so that we can see whether our count is increasing or not.

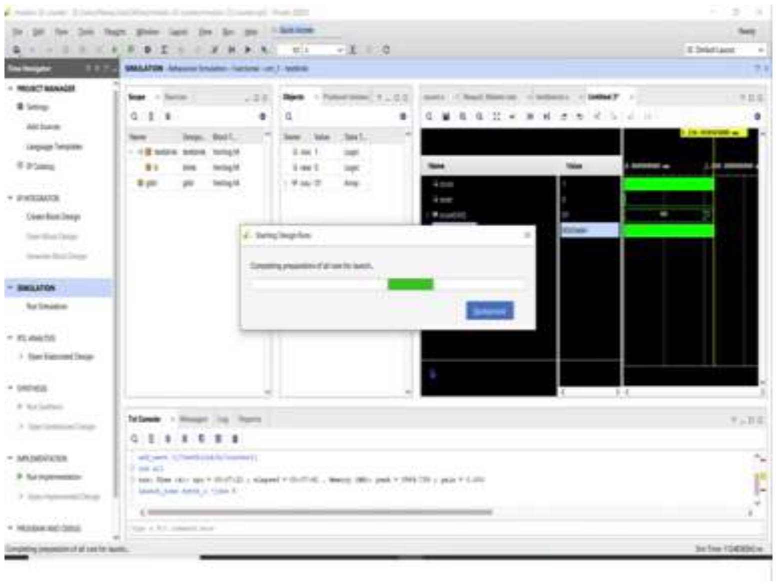
So, here you see that every clock my counter is increasing so it is changing we can, if we zoom on to this window then we can clearly see whether it is increasing by 1 or not but after some time you will see that this 0, 0 will become 0, 1 that means my count is increasing and that also signifies that my functional simulation is correct.

So, let us wait for a while, you see after 1 million, after 1 million nanoseconds this has changed from 0 to 1, so we can pause it for now and if you want to see in detail then we can see. By clicking on this pause button the simulation will be paused after, yes, so there you can see the simulation and now if we try to do a zoom out then we can see that so the it changed exactly at 1 million nanoseconds, so 1 million nanosecond it change from 0, 0 to 0, 1. So, similarly after every million sorry 1 billion nanosecond it will keep on changing, so the idea here is that at every one second it would change to a new value.

(Refer Slide Time: 16:37)

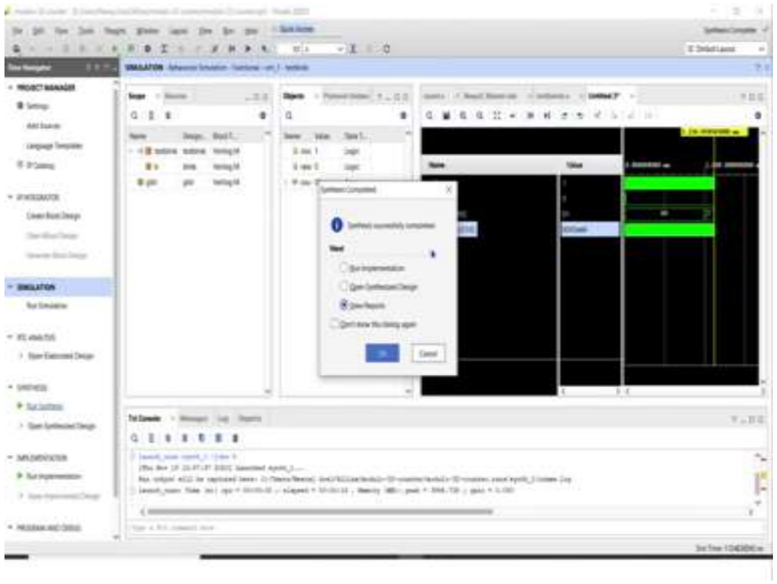


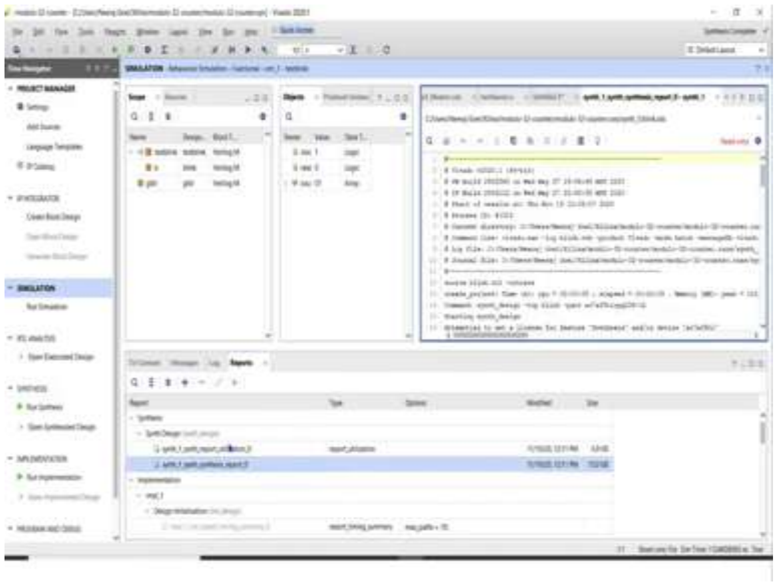


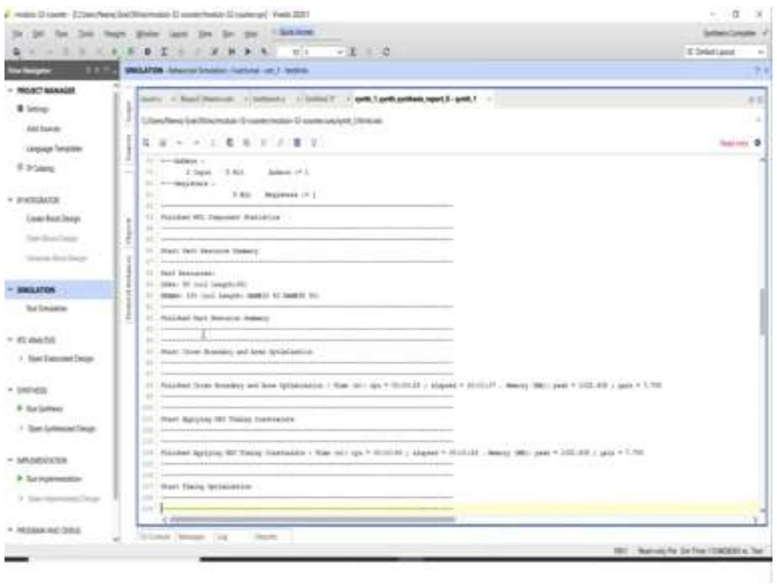


Now, after our simulation is correct then the next step is we have to go through the synthesis process. So, we click on the run synthesis step and then we will see that how synthesis will go on. After synthesis after synthesis is completed we see this window that now it is suggesting that either we would like to run implementation, open synthesized design or view report. So, let us go ahead and look at the reports. So, we can see from there that what has been done after the synthesis. So, this is the synthesis report.

(Refer Slide Time: 17:38)







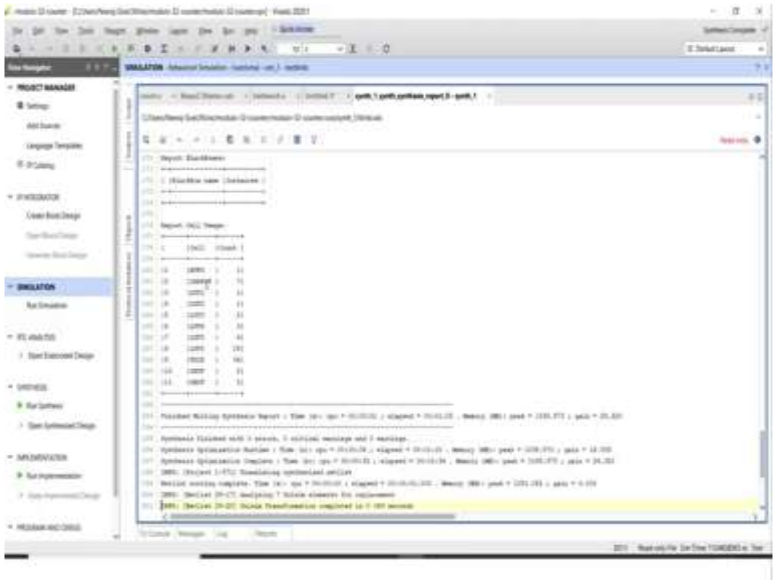
So, if we see the two reports here, one is synthesis report, another is report of utilization. So, the synthesis report tell us various information, it says that how much time it took and it also tell us various kind of analysis like how many, it says that in the RTL component there were two, there were adder of five bit and there are register which is also five bit registers, total number of DSPs and VRAMs are the total number of VRAMs and DSP which were there but looks like nothing was utilized.

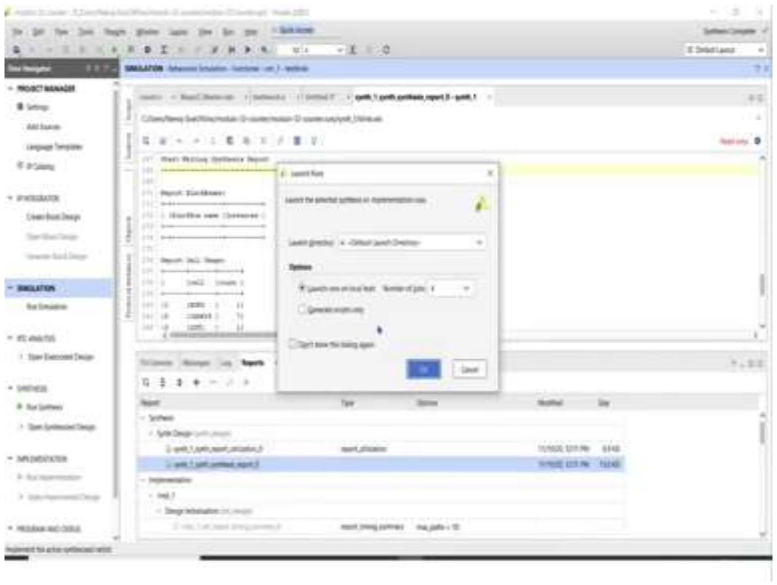
But this report will tell us that how much is the utilization, potential utilization factors. In the end it also tell us about the what would be the potential or estimated resource utilization, buffer is 1, carry units four is 7, so there are 7 carry four units are used, this carry four are you remember that we have in each slice there four bit carry which can be utilized. So, this is the cell which has automatically inferred by the synthesis tool and it has used 7 of them.

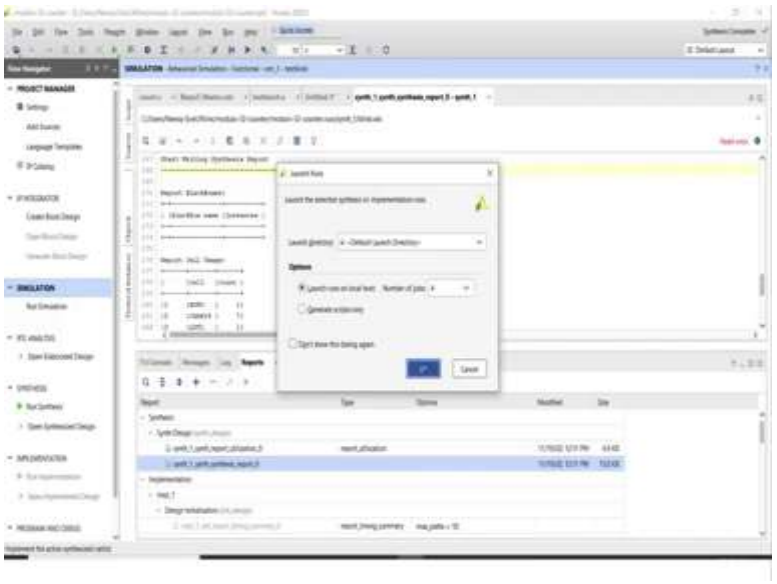
LUT with single input is used once, LUT with two inputs, LUT with three, inputs four inputs and five inputs so basically although there were only five input LUTs but they have been we can utilize into different formats so that is why different so that our utilization could be complete, so you see the six input LUTs are also 29 which has been used.

These are the flip flops which are also 34 in number and input buffer so there are two inputs and five outputs so that is why ibuff and obof is used. So, this give us the overall estimate how much time, what is the timing analysis and also the resource utilization. So, after looking at these reports then we can run the implementation, this is the second part. So, as I suggested earlier in the earlier part of this lecture we can also go for post synthesis simulation but let us skip that step for now and we go for a implementation part.

(Refer Slide Time: 20:08)

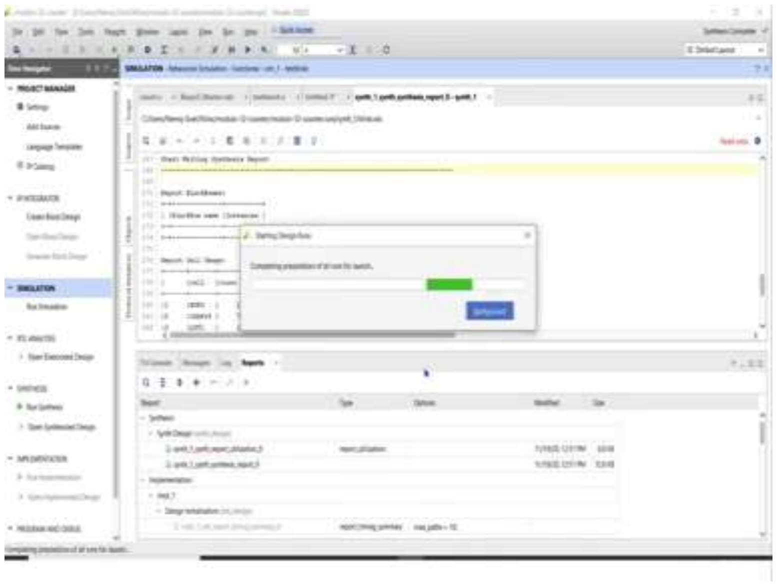






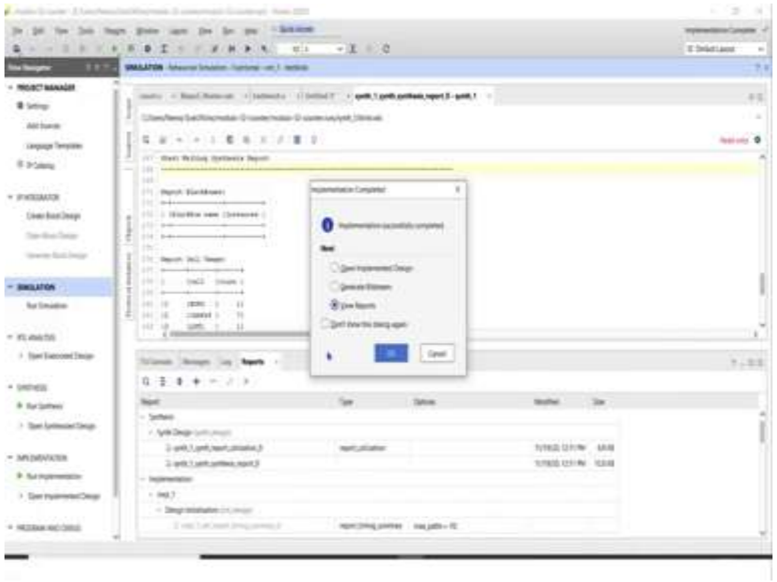
So, the second part is the implementation when we run the implementation then it is asking how many number of jobs, so this help us to do things in parallel so we are trying to use four processors on the computer and then we run it.

(Refer Slide Time: 20:32)



Now, the implementation process is complete so after implementation process we can go ahead and look at the reports.

(Refer Slide Time: 20:43)



Implementation Completed

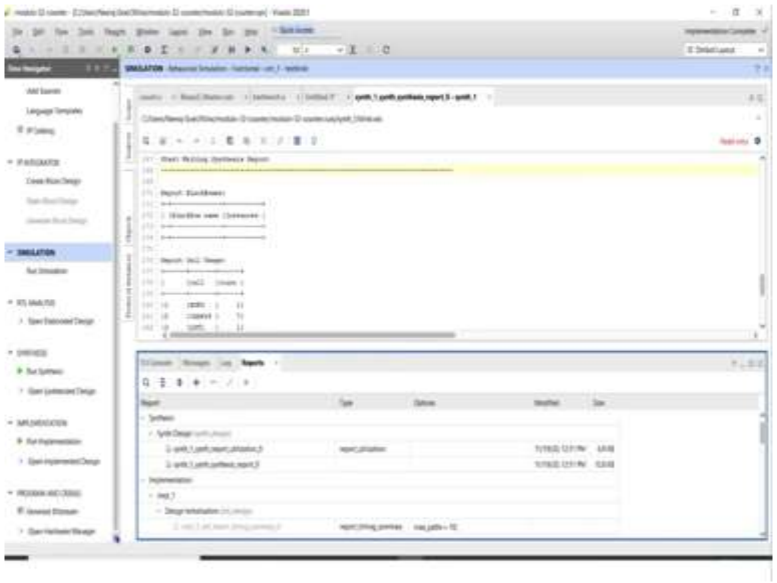
Implementation successfully completed.

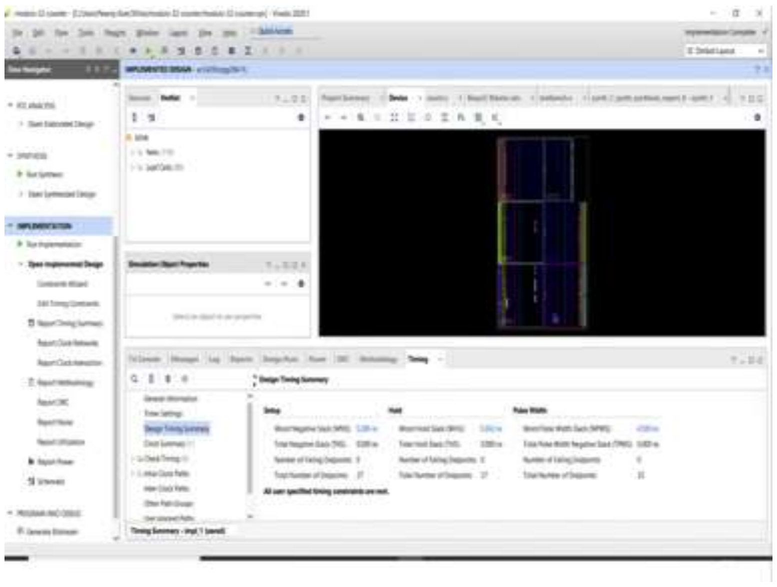
Next

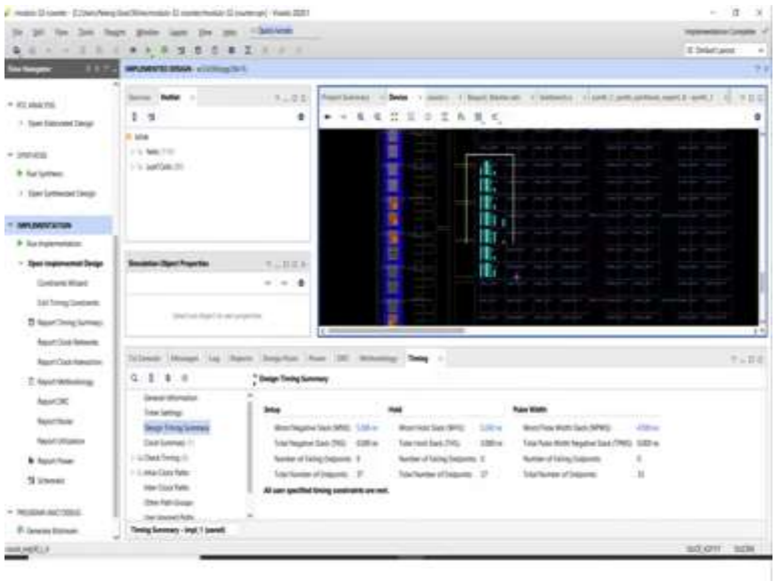
( ) Open Implemented Design
( ) Generate Bitstream
(●) View Reports

☐ Don't show this dialog again

OK Cancel





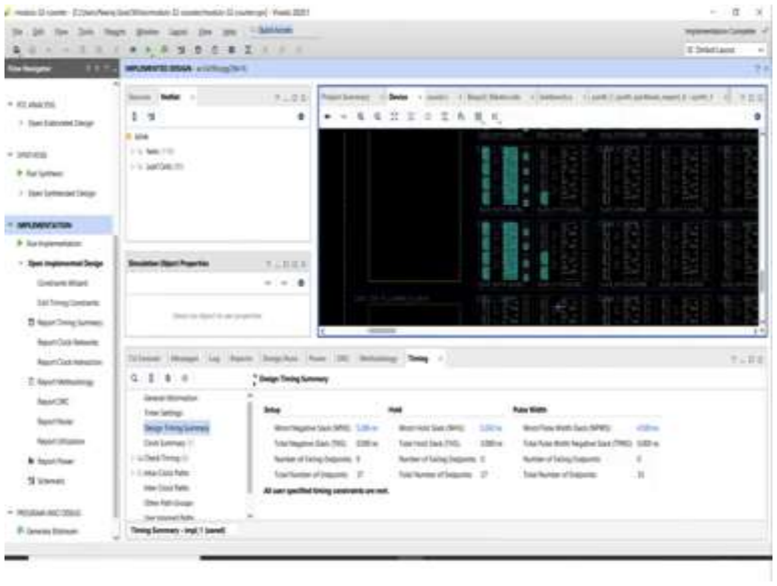


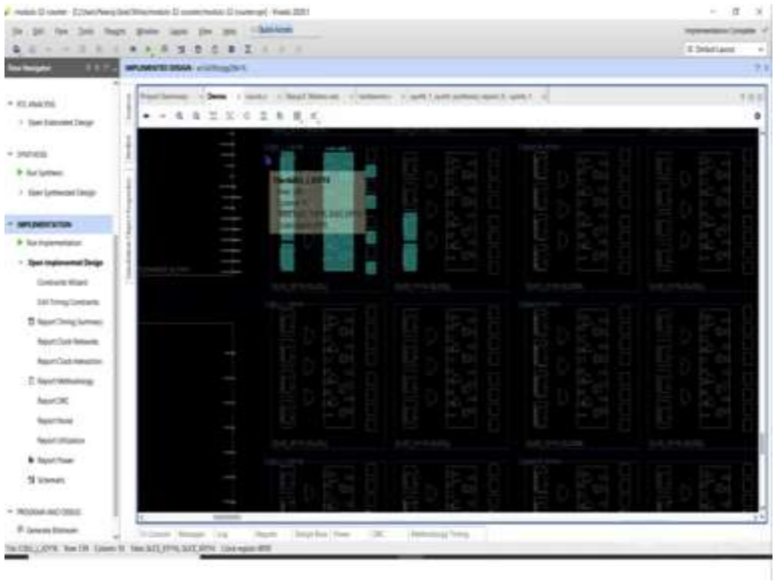
So, if you see here the type of reports we have generated is the, you can see in the left panel that there are reports where reports for timing, reports of clock network, report of methodology, report of utilization, so let us and we can also look at the implemented design. So, if we open the implemented design then we can also look at some of the interesting aspects of the FPGA.

So, this is the overall layout of our FPGA that it has the FPGA has been divided into four different physical parts and then let us look at we see that this is after placement and routing so this is the after implementation we can see there is some logic here, so we can zoom this part and then look at this area.

So, if we further zoom in then we can see that this is clock reason and yeah and so by browsing we can see that the multiple clock registers, so these clock registers are helping in their multiple clock registers so that clock could be buffered, the major part of the design is on to this part. So, if you see let us look at even further, so we can further make a zoom in onto this part.

(Refer Slide Time: 23:18)





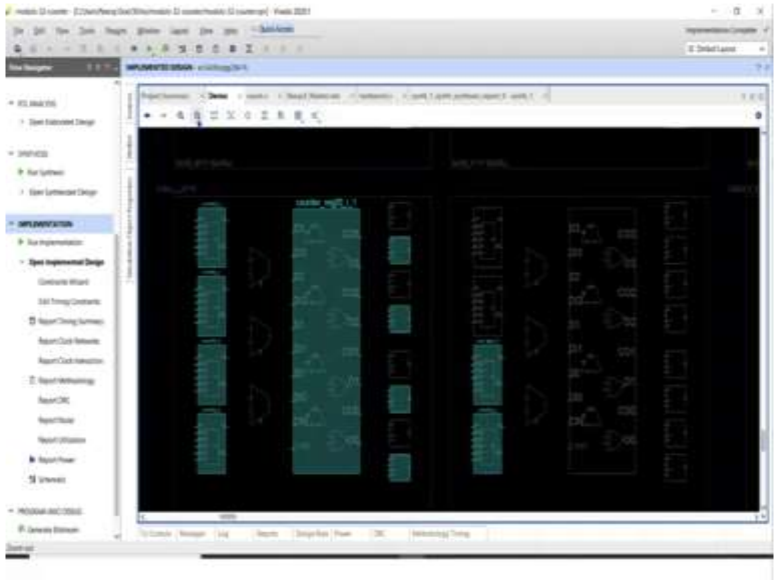
So, you see this is one CLB, we can further zoom it see what is there inside so internal structure is being shown here. Now, it is showing that this is the counter reg so the register which is a 32-bit counter so that 32-bit counter is also being incremented every time so that means it would require you can see there is a this part of the slice, so this is one slice, in this one slice there are the four LUTs so all the six input LUTs are used and further this is the counterpart, so basically the addition which is happening, so this is the carry chain so you can see the carry chain which is going from here to here so this carry 4 chain module has been used and four of the registers has been used.

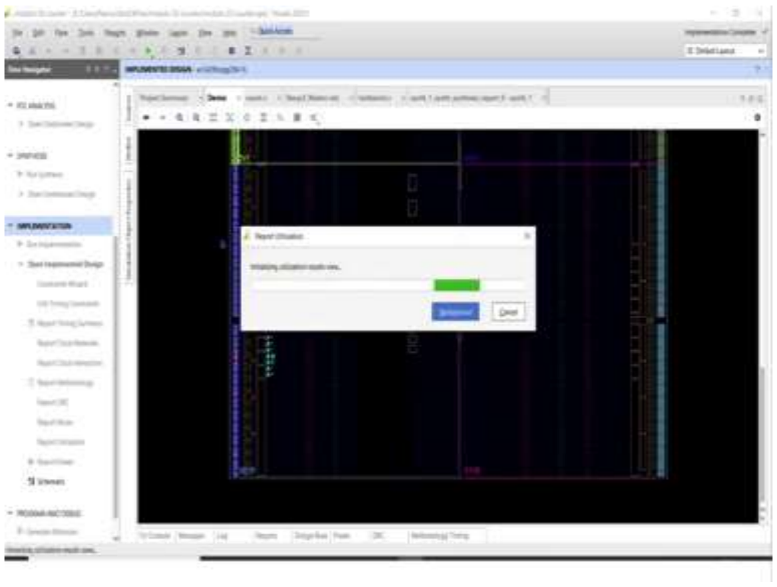
And similarly, this is another part of the counter which is being used here. So, you can see the utilization of the resources interconnect is not shown but at least the physical placement is shown and another thing to notice is that this is the all the parts all the basically the all the logic is placed nearby so that this would have minimum wire delay and also the if we see the input and output so these are the u19 v19 these are the pins where the pins are physically placed.

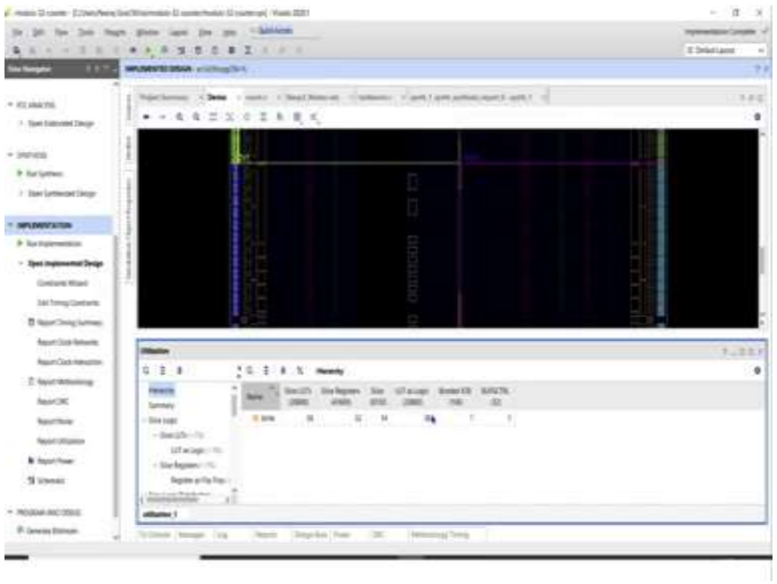
So, because pins are here that is why all the logic has been placed on the nearest CLBs which were there near these bits. So, this physical view or the physical implemented design can give us an idea that how which part of the FPGA has been used and which particular logic has been used.

So, if you would like to explore it further you can install this Vivado software and you need not to have FPGA board but still you can explore how different units are there and where they are placed and all those all that kind of analysis has can be done.

(Refer Slide Time: 25:48)





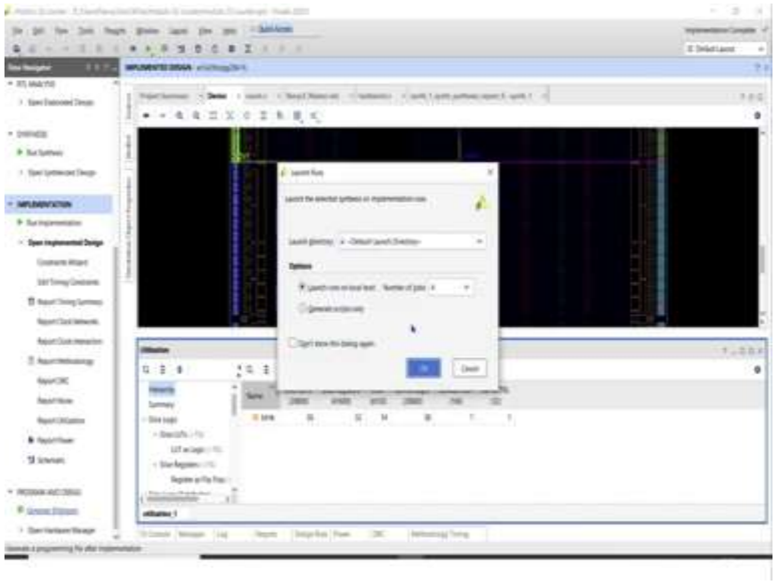


So, the in detail we can see all other reports also for example design dual check, noise, utilization, power so here it clearly says that out of 20000 LUTs some 38 LUT has been used, out of 41 registers 32 registers has been used and total number of 14 slices has been used and LUT as a logic is 38 so sometime LUT also can be used as a memory so all of those things will come in the utilization report.

So, this is how we can look at different reports and analyze what how the circuit has been implemented onto the FPGA. The last step of the process is to generate the bit stream, now we will generate the bit stream.

So, now the implemented design clearly say that which particular CLB which particular slice and how they are configured which logic has been configured so all those things all this information is there. Now, given the which particular device is there so based on that the bits stream is generated. So, let us wait for some time till our bit stream is generated.

(Refer Slide Time: 27:09)



Launch Runs

Launch the selected synthesis or implementation runs.

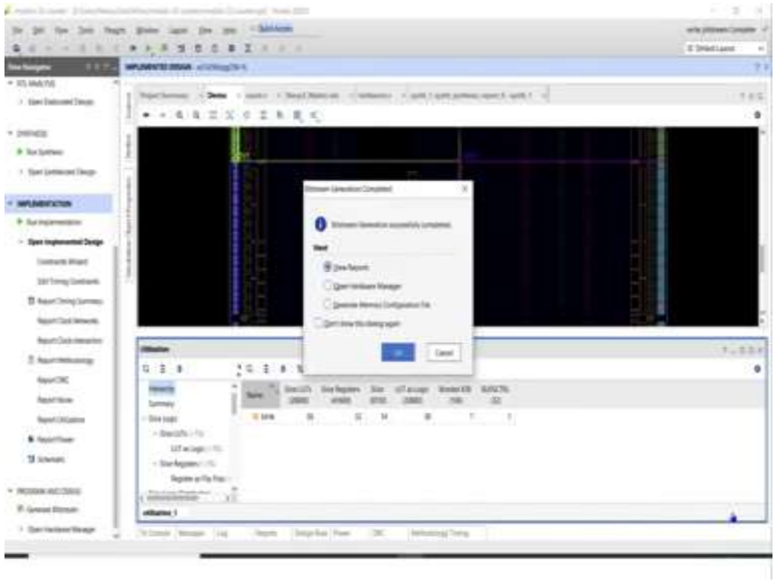
Launch directory: <Default Launch Directory>

Options

[✓ Launch runs on local host] Number of jobs: 4

Generate scripts only

Don't show this dialog again



Bitstream Generation Completed

Bitstream Generation successfully completed.

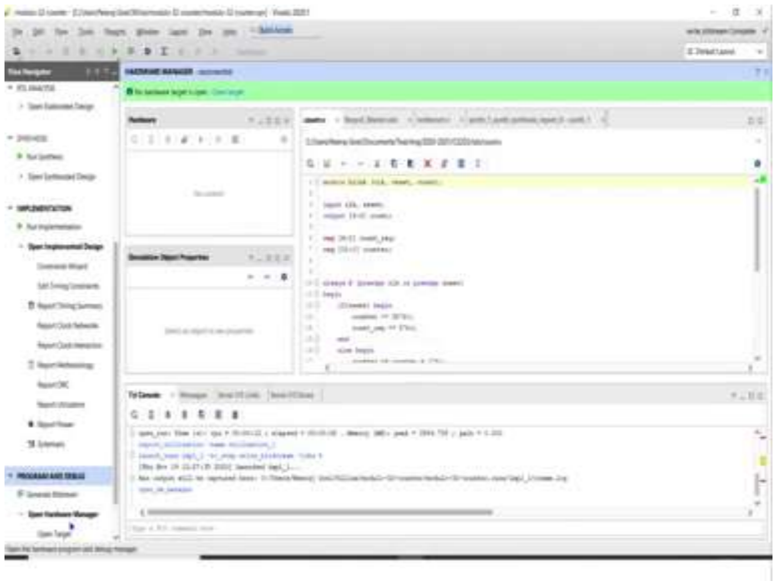
Next

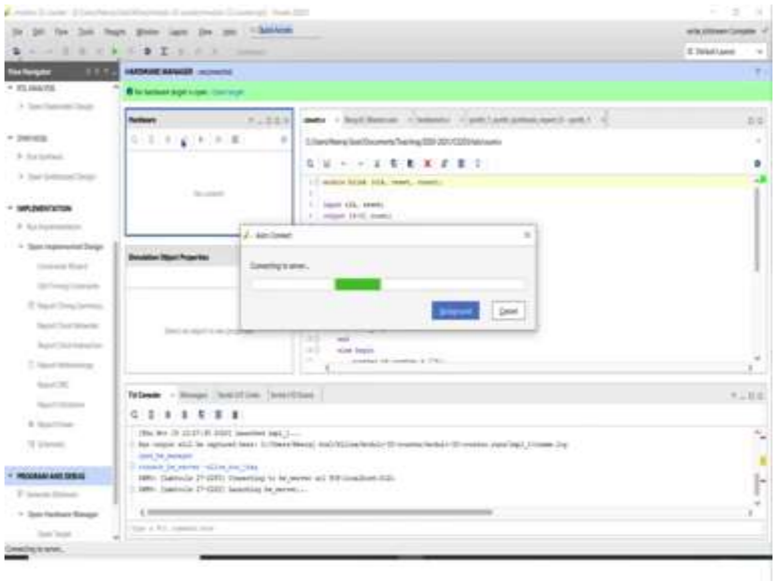
[✓ View Reports]

Open Hardware Manager

Generate Memory Configuration File

Don't show this dialog again



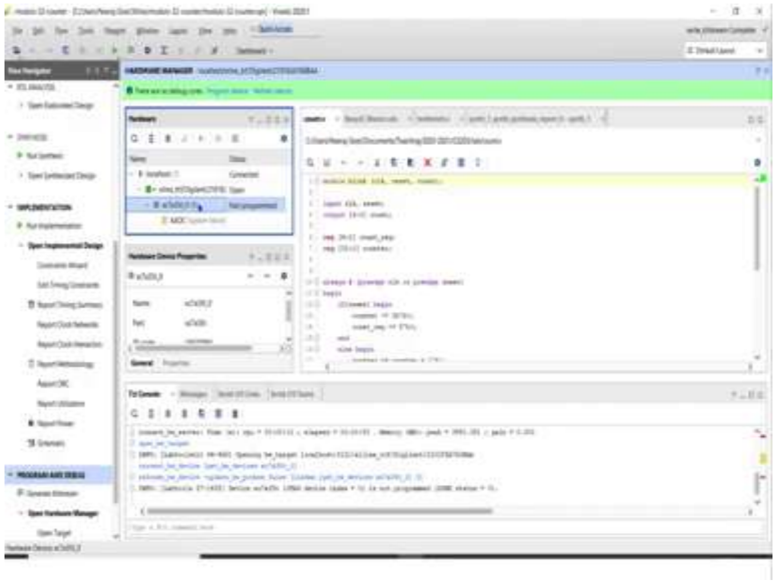


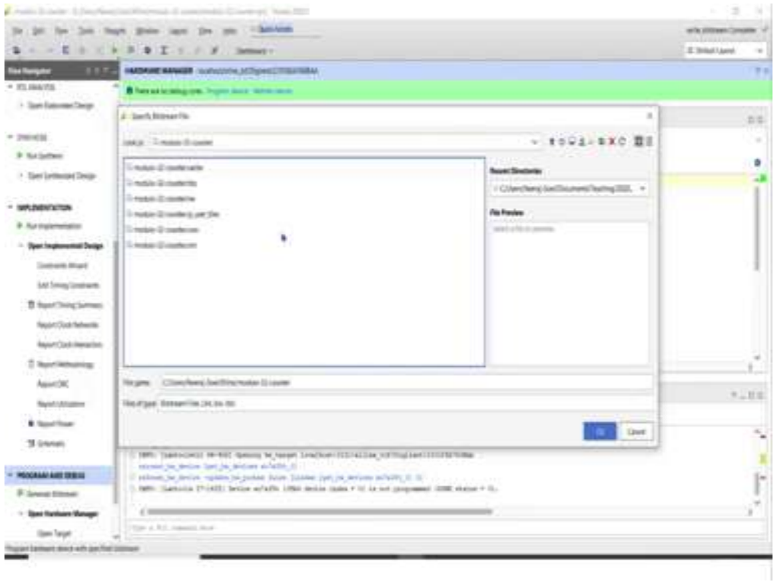
Now, bit stream has been generated so after bit stream has been generated the next step is we have to download the bit stream onto the hardware using the hardware manager. So, we have to, now we can open the hardware manager so that we can see where we can install that bit stream onto this.

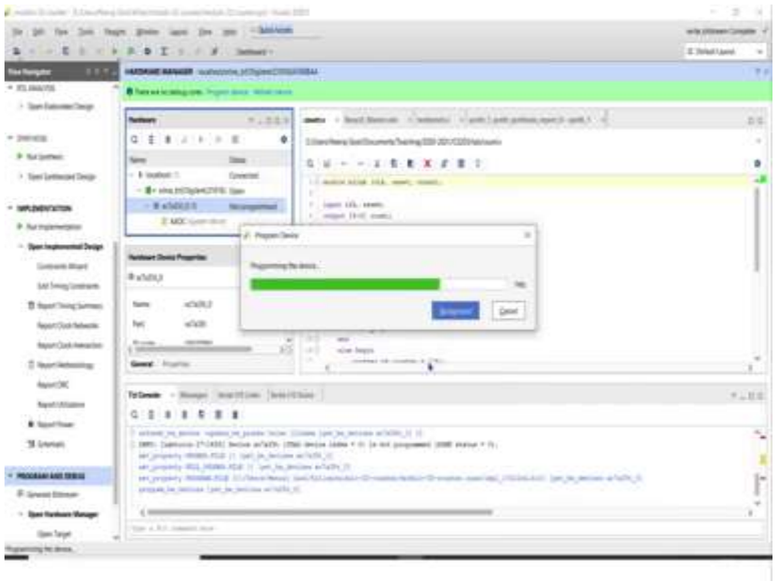
So, hardware manager can also be opened here so now this is the hardware manager which is here now we see here there is a auto connect so basically if there is a there is a FPGA unit which has been plugged in it will try to identify whether there has been any FPGA which is connected so it would be able to explore and see which particular FPGA is there.

So, after this exploration you see that xc, xc7a35 this particular we can see our basis board which is present here. Now, if we right click on to this and then we can say program device, so using the program device we have to now first of all tell where is the bit stream.

(Refer Slide Time: 28:43)

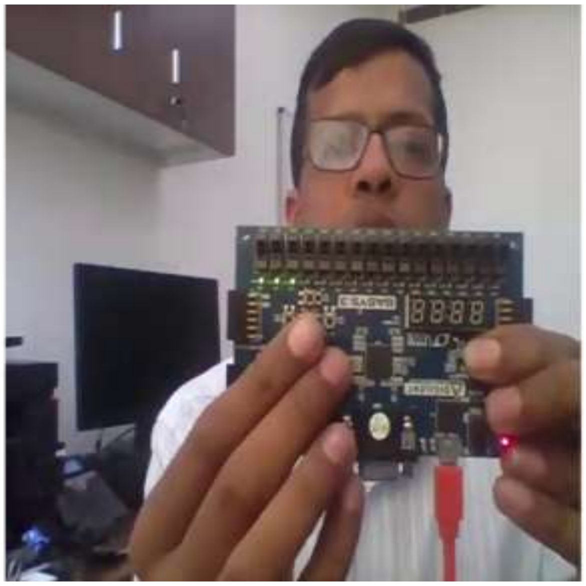






So, the bit stream path need to be given, the path would be there in the project directory, in the project directory we have to see in the project dot hardware and then there is a hardware dot 1 and then we can see sorry it is not there, it would be there in dot runs and here it is a implementation, in the implementation we can see a bit file so this bit file we can use say ok and we can say program.

(Refer Slide Time: 29:35)



So, during the program phase it would be downloaded onto the hardware and I am putting this in front of you so you can see that these are blinking in a fashion so if I press this button then it will again reset and it will start from the 0, 1, 2, 3, 4 so again if I press a reset button it will again start from the 0.

So, this way we see that the although our module is sitting on the FPGA but it is connected to these LEDs and also the central pin which can help us to reset. So, this is how the whole process will work and this demonstration can be helpful to design or to see that we can explore more, so you can, you would have a question that if I do not have a FPGA, FPGA hardware how can I do experiment?

So, that that question is correct so FPGA hardware is also a bit costly but this software which is coming from vendor like Xilinx or Altera so those software are usually free and we can at least go to the stage where post layout or basically after placement and routing that kind of analysis can be done and but actual testing can be done only on the board that will always remain the fact.

So, if you get access to these ports then that is wonderful but there would be some schemes or there would be we hope that some time there would be a we would be able to access some of the FPGAs on cloud and then we would be able to remotely access them so that would also become one of the possibility to do experiment on FPGAs.

So, with this I would like to close and I would like to I would expect that you would also do at least some experiments so that you can go through this whole process and can explore that how much is the resource utilization, how much is, how you can do static timing analysis, et cetera, with this yeah, thank you very much.