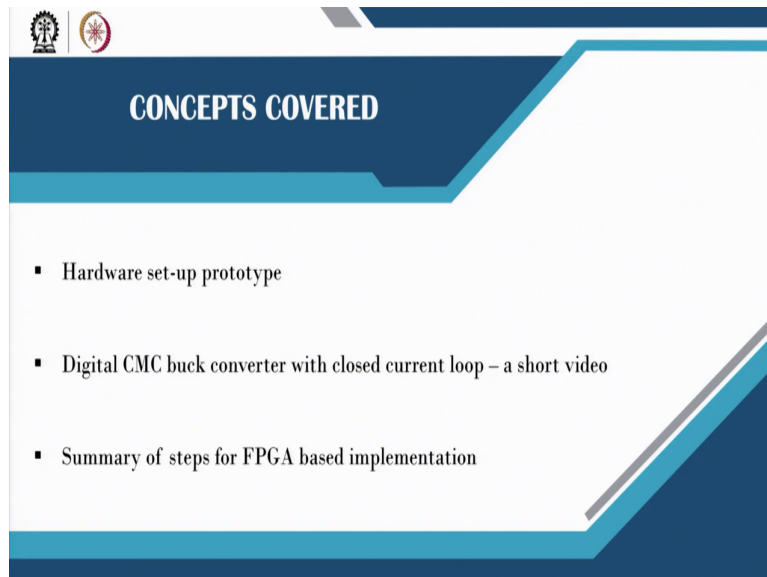


**Digital Control in Switched Mode Power Converters and FPGA-based Prototyping**  
**Prof. Santanu Kapat**  
**Department of Electrical Engineering**  
**Indian Institute of Technology, Kharagpur**

**Module - 06**  
**Digital Control Implementation and FPGA-based Prototyping**  
**Lecture - 57**  
**Step-by-Step Guidelines for Digital Control Implementation using FPGA**

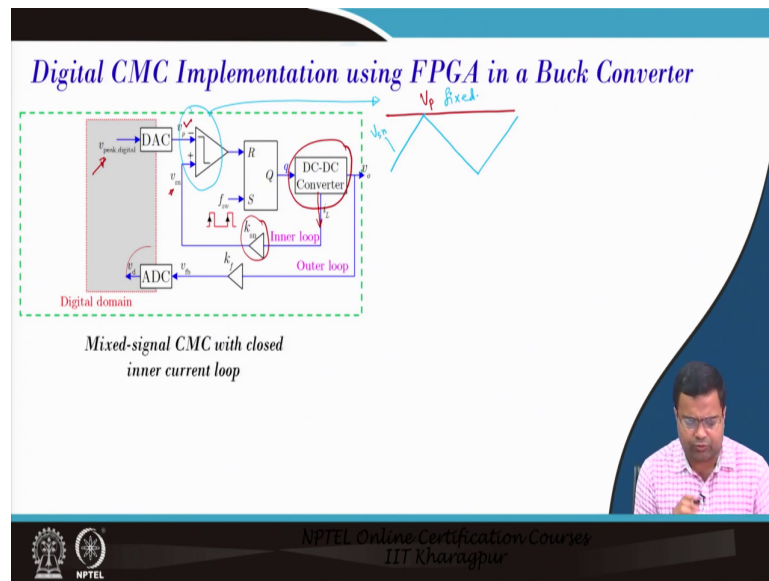
Welcome to this lecture we are going to talk about Step-by-Step Guidelines for Digital Control Implementation using FPGA.

(Refer Slide Time: 00:33)



So, here we will first talk about our hardware setup prototype, then a digital current mode control buck converter with a closed current loop a short video will be presented, and the summary of the steps for FPGA-based implementation.

(Refer Slide Time: 00:47)



So, the digital control current mode control implementation using FPGA in a buck converter. So, here if you recall lecture number 15 where we talked about mixed signal current mode control, and in this case, we are talking here about a DC-DC converter, and in this particular, you are using a buck converter.

So, any DC-DC converter can be used and we are using current inductor current and then this inductor current I mean we are using a sensor there will be a sensing gain and this will convert into sense voltage. We can sense the output voltage through an A-to-D converter, but initially, we are using a fixed current difference. So, that is the peak voltage that is going out of the digital loop and passed through a D to A converter and this is generating a peak current voltage reference.

That means we are talking about this peak voltage, equivalent voltage and this will be compared with the sense voltage; that means, this is our V sense which is analogous to our inductor current. That means, this is a sense voltage corresponding to the inductor current; so, this is what is expected from these two signals here, which I am showing here.

So, here we are using a fixed current difference; that means, V P is fixed this is fixed; that means, this is the inner current loop closed, but the outer voltage loop is still open. Because we just want to show some implementation demonstration in the subsequent lecture this week; so, here is mixed signal current mode control with a closed loop.

(Refer Slide Time: 02:30)

### Digital CMC Implementation using FPGA in a Buck Converter

Mixed-signal CMC with closed inner current loop

Complete closed-loop test set-up for this online course

NPTEL Online Certification Courses  
IIT Kharagpur

Next, this is a setup and we have discussed that this setup can be arranged for buck mode as well as the boost mode. So, this we are not now we are using like a buck converter and this is our signal conditioning board and in the bottom of this will be FPGA.

(Refer Slide Time: 02:55)

### Demonstration of Buck & Boost Converters using FPGA kit

Buck or Boost Converter

Signal conditioning board

Xilinx FPGA kit

Mixed-signal CMC with closed inner current loop

only digital signals

NPTEL Online Certification Courses  
IIT Kharagpur

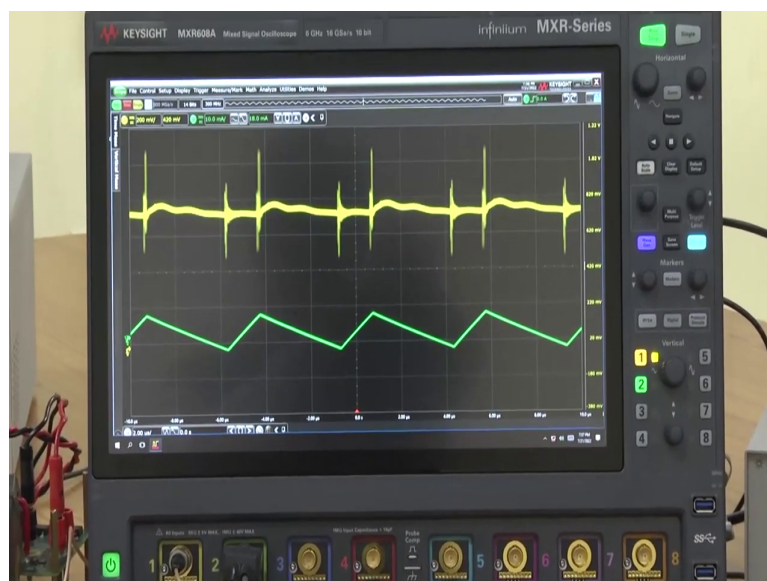
So, now we are going to implement mixed signal current mode control; so, we have discussed that we are initially using fixed current difference. That means,  $V_p$  and that is compared with our sense inductor current like this; so, this is our sense  $V_{sn}$  sense inductor current. And in this case, this DC-DC converter which is here is in this board and it has the current sensor

and everything in this board, voltage sensor then up, and also we are providing gate signal from this particular board; so, the gate signal and the gate drive is also in the PCB.

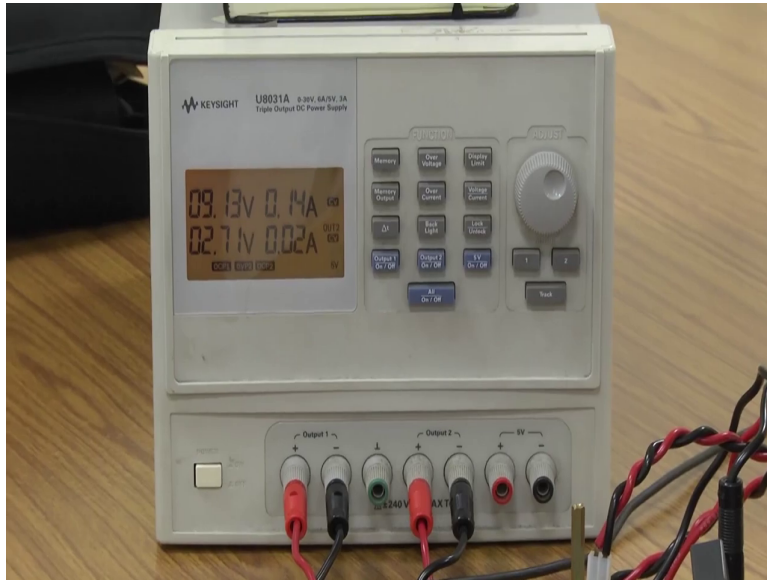
Next, we are talking about ADC DAC these all are in this board these all are in this signal conditioning board. And then this digital platform where we are implementing this logic is in the FPGA ok so; that means, we are using the digital control algorithm. So, this platform only deals with the digital signal; only digital signals; only digital signals; which means, the output of the ADC will go to the FPGA.

Then from the FPGA, we will generate the digital value which will go to the DAC; that means, D to A converter and the D to A converter is sitting in this mixed signal board. So, we are using FPGA as the digital control platform and we are using a Xilinx FPGA. So, now, we are going to see a short video where we will show the full implementation; which means, the life-you-know implementation of the mixed signal current mode control with the inner current loop closed.

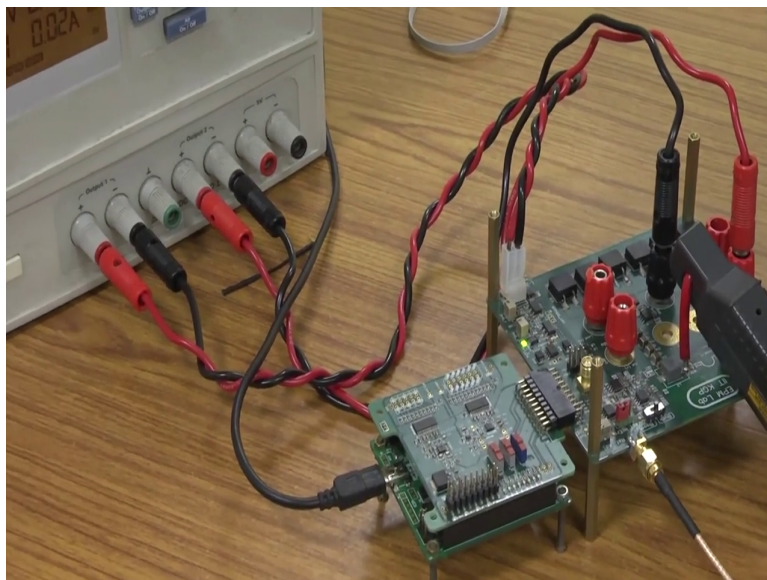
(Refer Slide Time: 04:50)



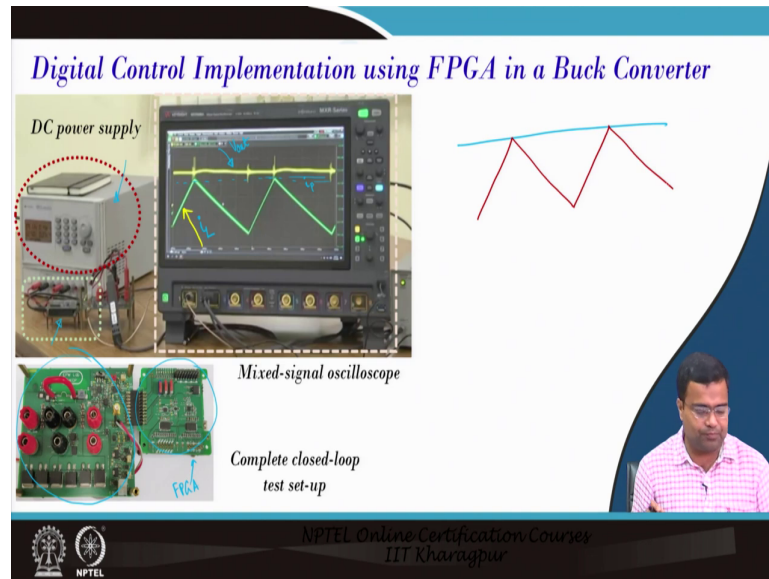
(Refer Slide Time: 05:31)



(Refer Slide Time: 05:44)



(Refer Slide Time: 06:24)



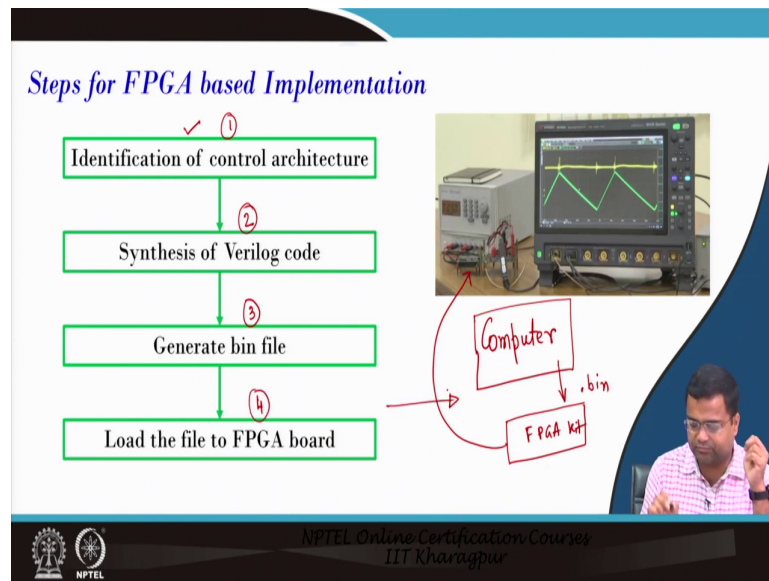
So, now we have just seen we have seen the live demonstration of the mixed signal current mode control. And you can see that we are using an oscilloscope to display the inductor current waveform here and we are using a current probe this is the current probe we are using ok.

So, we are using a current probe to display the inductor current and this current probe result is shown here this is a current probe result. Now sorry; so, this is our current probe a current probe result inductor current and we are also displaying the output voltage this is the output voltage. So, this is my  $V_{out}$  and this is my actual inductor current because you are using a current probe; so, this is my actual inductor current  $I_L$  ok.

So, you can see the periodic waveform, and here the current reference is fixed. So, we call it a  $V_{peak}$  or the peak current since we are talking about the current here directly. So, it is better to use in this case as if like an  $I_P$  in the analogous sense it is like there should be an  $I_P$  which will limit the current ok.

And this is the power supply and this is our actual kit which consists of this power stage board signal conditioning board at the bottom, there will be an FPGA for the complete setup. Here, we are using a fixed current reference and the inductor current is just touching this, but in the subsequent lecture, we will do a closed loop.

(Refer Slide Time: 08:17)

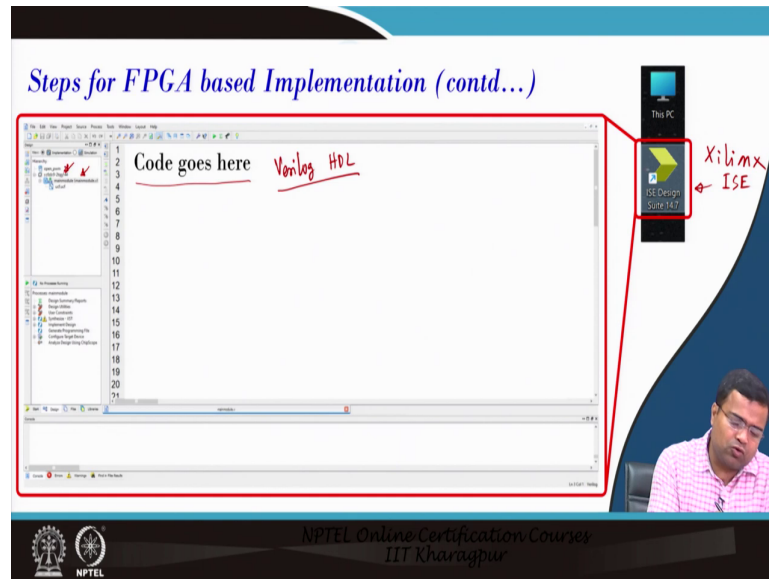


So, to implement what is missing here is the computer, because we need a computer to generate the code beam file and that beam file will be. So, the computer will generate the dot bin file and this will be loaded to the actual FPGA and this FPGA kit is here. So, then once you dump it computer is nowhere in the picture; so, the real controller implementation or the real-time FPGA controls the whole DC-DC converter.

So, to go to that computer how to generate that file beam file; so, that you can dump it into FPGA? So, first, we have to identify the control logic; that means, whether you are here we are talking about the peak current mode control with a constant current difference.

Though we are not going to present the whole code because we have not yet started the Verilog programming and that will be discussing the subsequent lecture. The second thing we have to synthesize is that code is the first step, third step we have to generate the beam file and we have to load it into the FPGA that we have discussed here ok.

(Refer Slide Time: 09:38)



So, here you know we will be discussing in detail this Verilog coding when you go to the actual FPGA class that will be coming in the next week. So, in the FPGA we have to install the Xilinx ISE again this is just for Xilinx I would say ISE and it is not mandatory I would say to implement this or to install this software. But at least you should have some visual understanding of how you implement digital control.

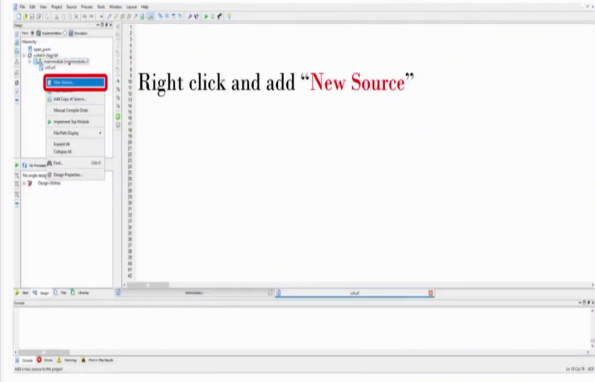
So, we have to create a project we will discuss this then under this project you have to write a dot V file; that means, you have to write a Verilog code and we will discuss in detail Verilog HDL we can also use V HDL, but I am just talking about Verilog HDL.

So; that means, your code whatever you want to implement should come and I am not showing the code because unless we discuss the code it may not be a good idea. But I am just saying that whenever you go to the next week's Verilog code; we will first show you how to create a project, then how to write a Verilog code and add it.




(Refer Slide Time: 10:57)

*Steps for FPGA based Implementation (contd...)*




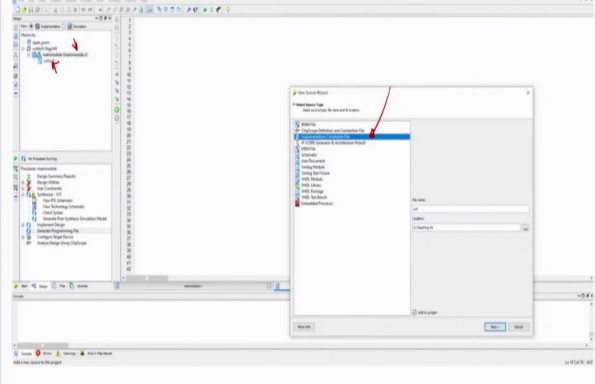
Right click and add "New Source"




NPTEL Online Certification Courses  
IIT Kharagpur

(Refer Slide Time: 11:03)

*Steps for FPGA based Implementation (contd...)*



gate out



NPTEL Online Certification Courses  
IIT Kharagpur

In addition to that; that means, we need to add a new source file into the project, and here since you are going to implement it, we have to use an implementation constant file because you know in FPGA there are a lot of digital pins because a lot of digital pins. So, each pin has its address; that means, every pin should have its address otherwise it will not be unique you cannot identify it.

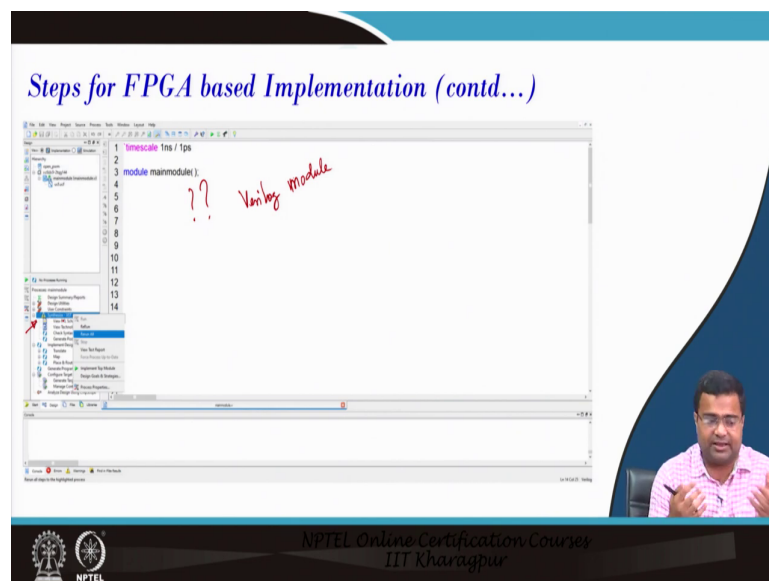
So, with this UCF file user constant file is the implementation file the job of this file is to indicate. I am because whenever a vendor you know maybe like a Xilinx commercializes this

FPGA or any other third party makes the kit. They make sure that because Xilinx FPGA if you go to the FPGA kit that we are showing here. We are talking about this FPGA made by Xilinx, but this whole kit is made by Numato lab associated with Xilinx.

So, you need to know the address of this pin, not the FPGA pin, because this pin is the address that we have to define, but eventually, some of these pins will be connected to the FPGA pin. So, naturally, the address of this pin will ultimately get reflected into the address of this pin, but it depends on how they are connected, how the routing is used, and whether this pin is connected to this pin or not that will be given.

So; that means, you need the UCF file to highlight that what is the pin configuration what is their address and when you go to the actual implementation we will show it.

(Refer Slide Time: 12:38)



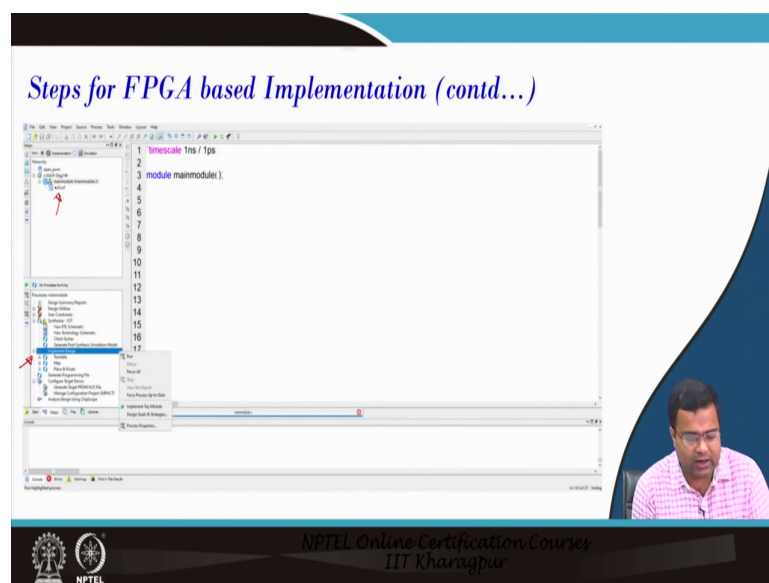
Once you create the UCF file properly that means, you want to send the gate signal, the gate signal, through the gate signal and you want to generate that gate signal and if you want it to be available let us say this PIN 1. So, the PIN one will be some address; so, you have to make signal out; that means, through FPGA you have to make signal out so; that means, you have to give an out pin and this gate signal go to that particular pin.

So, we will discuss this, but today we are just discussing the steps; so, there is a UCF file then assuming that you have already written a Verilog code and you also have an

implementation file. Now, you are ready to import your code run it, synthesize, generate the beam file, and load it to the FPGA; so, it is again not complete.

So, you have to write a Verilog module for the control implementation that we are going to do. Once you tested fine then once you are ready you have to go for run rerun all; that means, the synthesis step. So, this is a step synthesis, and this step will generate all the RTL everything; that means, then we will discuss RTL and how to synthesize it in the next weeks in detail.

(Refer Slide Time: 14:06)



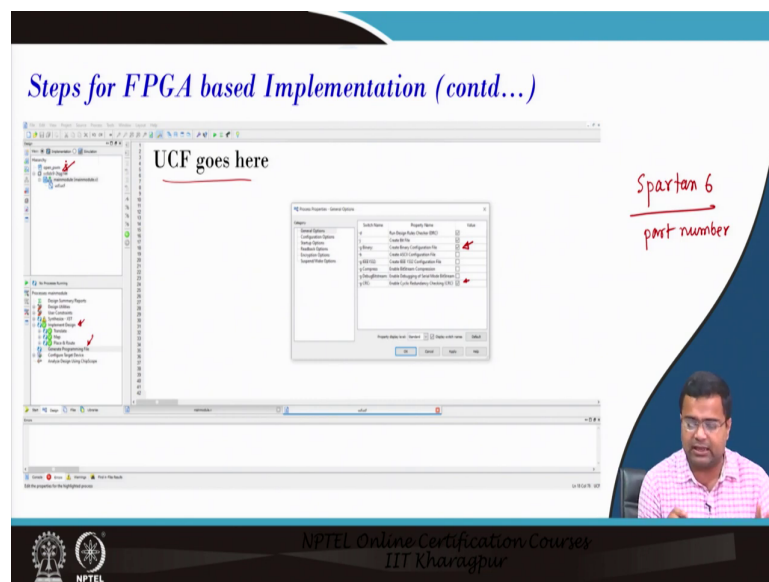
Then once your synthesis is done; so, till synthesis you do not need to bother about the user constant file. Because we are not going to implement you can even check the simulation whether my algorithm is working fine, whether are you getting the right algorithm that let us say I want to implement an AND gate or OR gate or any other full adder 4-bit adder; so, you have to check that. Once you are satisfied synthesis is complete then you go to the next stage which is the implementation stage, and this stage requires that you have properly defined the user constant file UCF file.



Then you have to route place and route how you connect because you have to consider the actual input-output pin as well as the internal routing. So, then once it is complete your implementation is over now you have to generate the program file. So, once you generate the program file so; that means if you zoom in on this part to generate the program file implementation you have to see the process property.

That means, in the process property you have to go general option where you have to choose the beam file; that means, here you can see run design rule checker create bit file and binary file also; so, these two are the default.

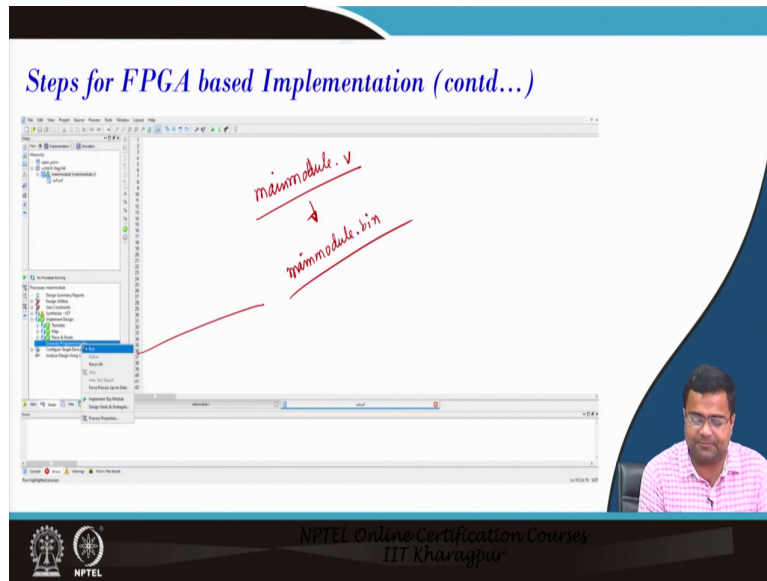
(Refer Slide Time: 17:08)



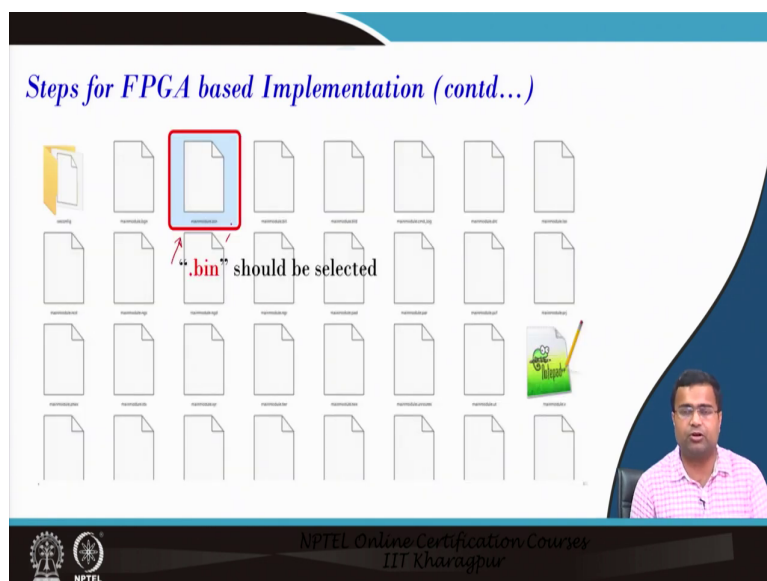
So, you have to also select the beam file and then you have to also enable this CRC check and this is your UCF file linking. So; that means, your implementation is done you have generated the beam file. So, this beam file is the file that will be dumped into FPGA and it will create automatically all these routing. And in this software the computer, has a library that has similar library to the actual FPGA because every FPGA Xilinx we are talking about spartan 6 OKs.

So, under spartan 6 there are N numbers of how many variants are there; so, which variant, what is the part number of the IC, IC part number? So, everything is important when you generate this program because we will discuss otherwise it will give a wrong result; so, you have to specify all this then it will generate the beam file.

(Refer Slide Time: 18:17)



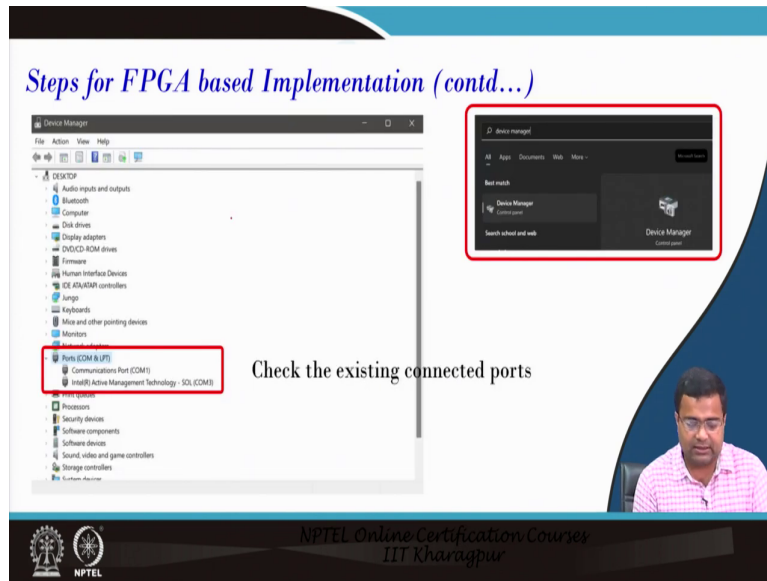
(Refer Slide Time: 18:19)



And once the beam file is generated, we have discussed; so, once you generate the programming file then you run it then it will generate a beam file. So, you have to select the beam file which is the beam file; that means, the program main is the main module. So, we have a main module sorry main module dot V file then it will generate the corresponding main module dot beam file after this stage and you have to search this main model dot beam file.

(Refer Slide Time: 18:57)

*Steps for FPGA based Implementation (contd...)*



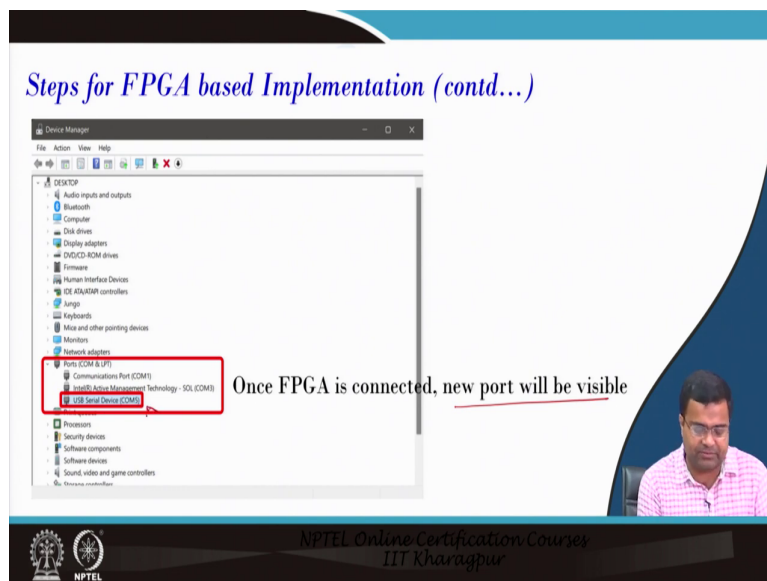
Check the existing connected ports

NPTEL Online Certification Courses  
IIT Kharagpur

Then once you search for it you also have to check the ports, because you have to this will go to this way you have to connect a USB cable from your computer to that USB C type cable which will go to your FPGA ok FPGA also has a USB port. So, then before you send this out you have to make sure that your port address is defined properly. So; that means, you go to a device manager in your computer check the port, and check the existing connected port.

(Refer Slide Time: 19:33)

*Steps for FPGA based Implementation (contd...)*

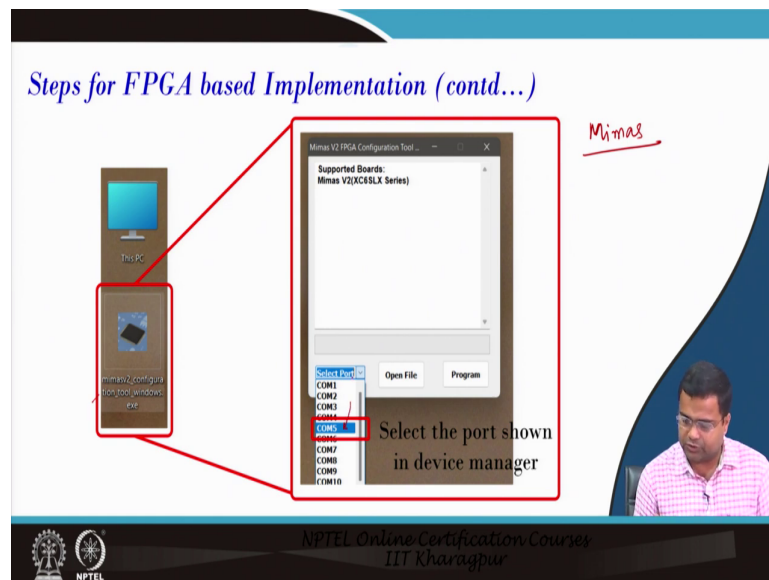


Once FPGA is connected, new port will be visible

NPTEL Online Certification Courses  
IIT Kharagpur

Once FPGA is connected; that means, if you connect to the FPGA then a new port will be visible. That means, your new port USB serial; so, it is a COM5; that means, that is a port is COM5; so; that means, your port is selected.

(Refer Slide Time: 19:44)



Now, you have to select you to have to give another software Mimas you can see; so, this is an app this app will interface your computer and the FPGA kit Numato FPGA kit. In the Mimas software since you have connected FPGA in the common COM port 5 which you have checked through the device manager you have to select in the Mimas software that I have selected COM 5 from the select port; so, this must be consistent.



(Refer Slide Time: 20:23)

*Steps for FPGA based Implementation (contd...)*

Name	Date modified	Type	Size
mainmodule.bgn	7/19/2022 9:51 AM	BGN File	7 KB
mainmodule.bin	7/19/2022 9:51 AM	BIN File	333 KB
mainmodule.bit	7/19/2022 9:51 AM	BIT File	333 KB

Click "Open File" and select the ".bin" file generated

NPTEL Online Certification Course  
IIT Kharagpur

Once it is done then you have to open the file; that means, you have to dump this code into FPGA. So, you click the open file here, select the port correspondingly then click on the open file. Then in the open file, it will ask for the directory where you kept your main module dot beam select that and select this particular file.

(Refer Slide Time: 20:48)

*Steps for FPGA based Implementation (contd...)*

Click "Program" to dump the Verilog code to FPGA

NPTEL Online Certification Course  
IIT Kharagpur

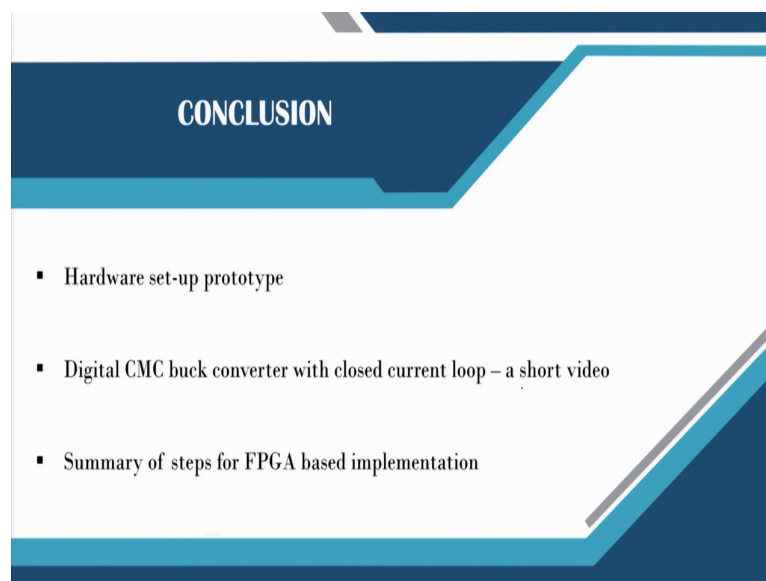
Once you select; that means, the selected file selector will be written like this you have to make sure that you have selected the right beam file then your program. Once you program, the program will have this green line to show the progress it is it will show that programming

is done and it is rebooting FPGA; so, then FPGA is dumped into FPGA. That means if you go to the FPGA you know if you go back, I will show you that this particular K.

So, once you dump then there will be a LED light; so, before you dump the LED light will glow once you dump it will turn off; that means, your FPGA is ready now you can start the operation. That means we have discussed that your program is dumped into the FPGA beam file is selected, the COM port is selected, and then the main module is selected, then you have programmed it and programming successfully installed, and then it will your FPGA run.

So, whatever we have shown in the clip video clip it is after this programming. Now, FPGA is ready it will generate the digital output and then that will convert to the analog voltage it will compare with the sensed inductor current and in that way, it will implement the current mode control with only the inner closed inner loop closed.

(Refer Slide Time: 22:14)



So, in summary, we have discussed the hardware setup prototype, and we have discussed the digital current mode control buck converter with a closed current loop. And we have also demonstrated a short video and we have summarized the steps for FPGA implementation. So, in the subsequent lecture this week we will show the actual hardware you know case studies where we want to show actual current mode control.

And how to use a mixed signal oscilloscope to check and validate your operation closed-loop operation of the DC-DC converter. Then if there is any stability issue or instability issue,

what is their power spectrum, and what does it look like? So, all these things we will be discussing in the remaining lecture this week that is for today.

Thank you very much.