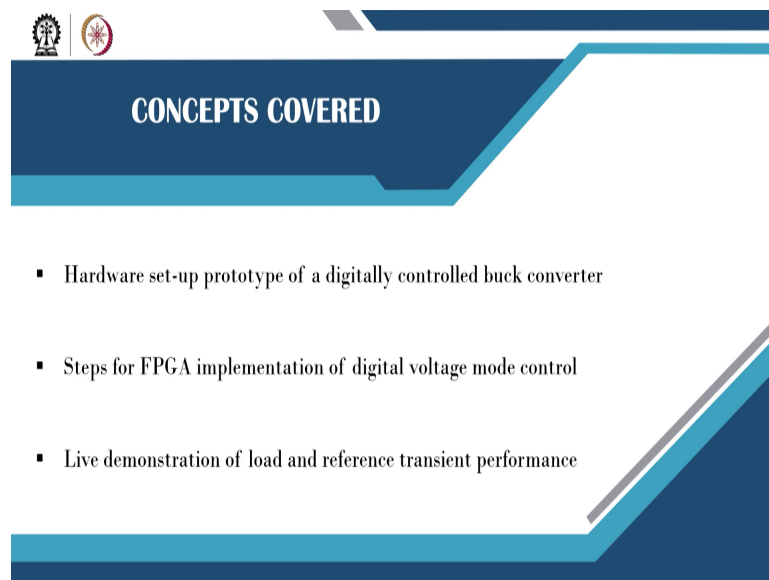


Digital Control in Switched Mode Power Converters and FPGA-based Prototyping
Dr. Santanu Kapat
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

Module - 10
Steps for FPGA Prototyping of Digital Voltage Mode and Current Mode Control
Lecture - 91
Steps for FPGA Implementation of Digital Voltage Mode Control

Welcome to this lecture we are going to show Steps for FPGA Implementation of Digital Voltage Mode Control in a buck converter.

(Refer Slide Time: 00:33)



The slide features a dark blue background with a light blue geometric shape on the left side. At the top left, there are two small circular logos. The title 'CONCEPTS COVERED' is centered in white text. Below the title, there are three bullet points, each preceded by a small square icon.

- Hardware set-up prototype of a digitally controlled buck converter
- Steps for FPGA implementation of digital voltage mode control
- Live demonstration of load and reference transient performance

So, here we will say the hardware setup prototype of a digitally controlled buck converter. Step for FPGA implementation of digital voltage mode control and the live demonstration of load and reference transient performance.

(Refer Slide Time: 00:45)

FPGA-based Digital Voltage Mode Control Implementation

Test set-up of the prototype

Digital voltage mode control

NPTEL Online Certification Courses
IIT Kharagpur

So, if you go for FPGA-based Digital Voltage Mode Control Implementation. We have discussed this in sufficient detail from lecture number 71 to 74 and then these are you know PCB picture prototypes that we are going to consider.

(Refer Slide Time: 01:02)

Demonstration of Digital VMC Buck Converter using FPGA kit

Buck Converter

Signal conditioning board

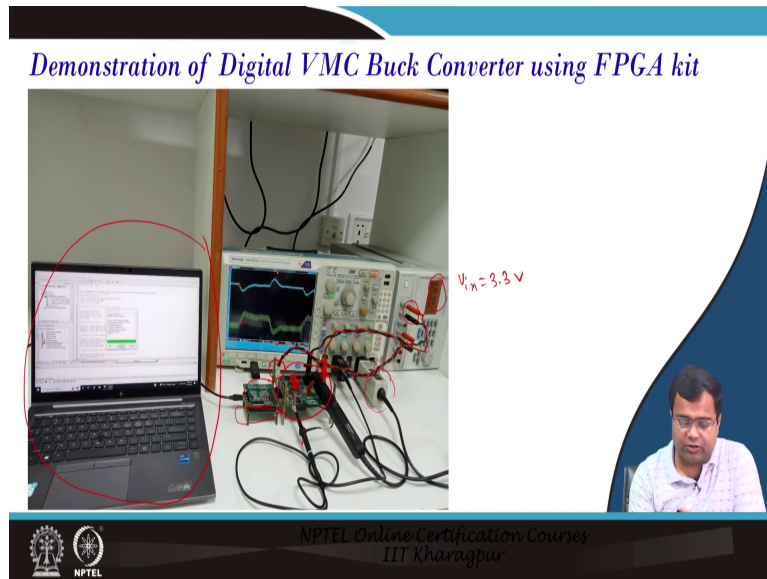
Xilinx FPGA kit

Digital voltage mode control

NPTEL Online Certification Courses
IIT Kharagpur

And in this particular demonstration, we are going to consider digital voltage mode control in a continuous conduction mode or a synchronous buck converter and this PCB is for the buck converter then the signal conditioning circuit and the Xilinx FPGA that we are using the FPGA prototype.

(Refer Slide Time: 01:20)

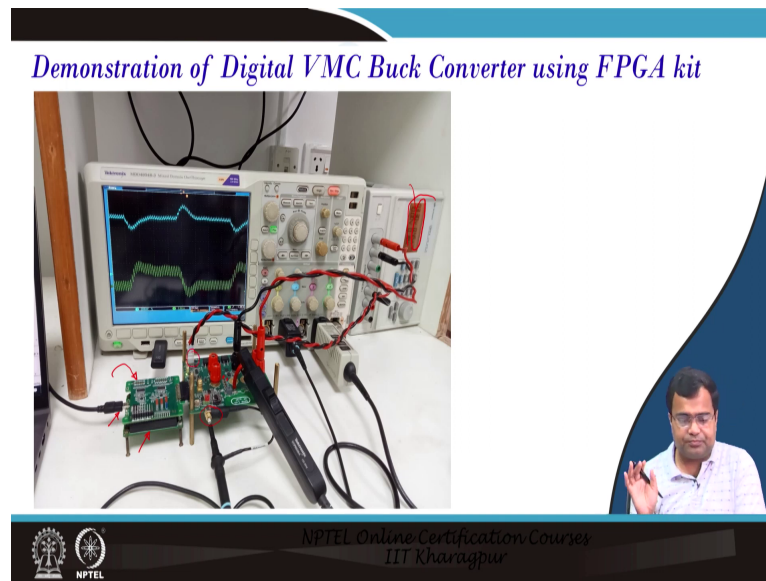


And this is the picture of this you know our setup where you know we are programming through a computer and this computer that a USB cable is connected and the top part of here is a signal conditioning board and you can see the bottom part which is FPGA.

So, the signal conditioning board just plugs into the FPGA ok. And this is our power stage and here we are this probe is used to measure the output voltage and this current probe is used to measure the inductor current. And this is the power supply you can see it is set V_{in} to be 3.3 volt. We are applying ok.

There are 2 power supplies sorry 2 supply one is for the buffer like you know op-amp supply and all this that is for supplying op amp a DC DAC and all and we have used a voltage regulator in the PCB also. And this particular one is used to use as the input to the DC-DC converter which is these two points, ok. And this is your current probe that thing this is the oscilloscope.

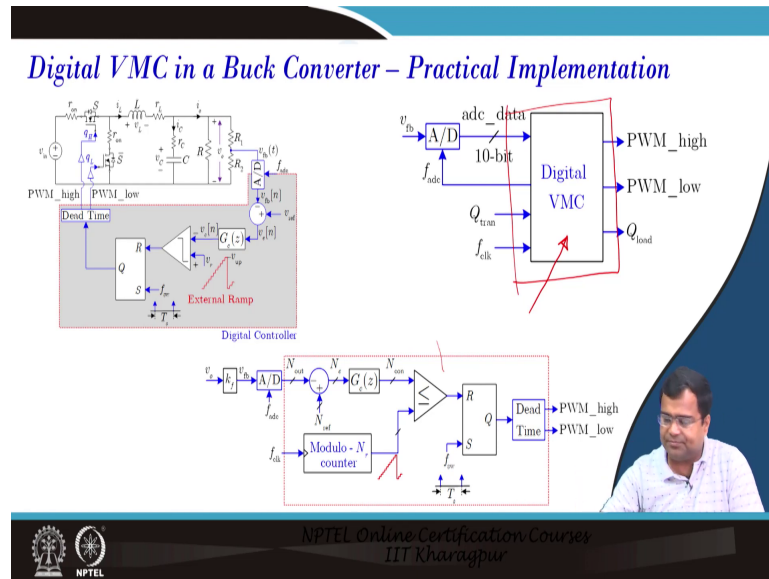
(Refer Slide Time: 02:33)



So, here if we take a closer look so, again you can see the bottom one is the FPGA1 and the top one is the signal conditioning board and this is the USB cable that we are going to consider and this is our power stage prototype. And this pin we are taking the output voltage. We can get the output much cleaner if we can tape tap directly from the capacitor output capacitor.

But just to make it flexible for the demonstration we are using this one. And this is a current probe that has 120 megahertz current probe and you can see the input voltage is the first channel is the input voltage, which is 3.3 volt and this one is 10 volt for supplying to the op-amp ads and so on.

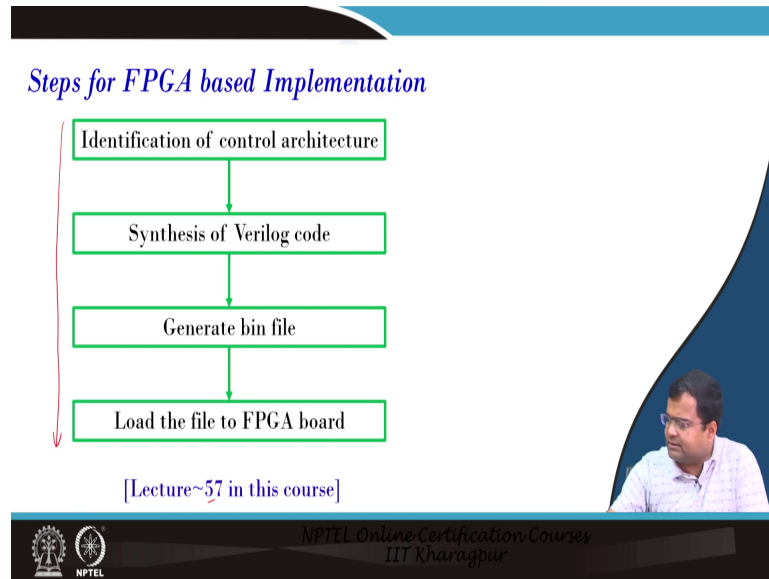
(Refer Slide Time: 03:19)



So, this digital control implementation is what we have kept; that means, I just want to show you this board we have ADC and DAC though for voltage mode control we do not need DAC and these data are getting interfaced with the FPGA through this i o plug-in pin. So, this side also we have and they are going to the FPGA is. So; that means, this i o pin ultimately whatever is going on is nothing, but this interface and FPGA inside FPGA were implemented in digital voltage mode control ok.

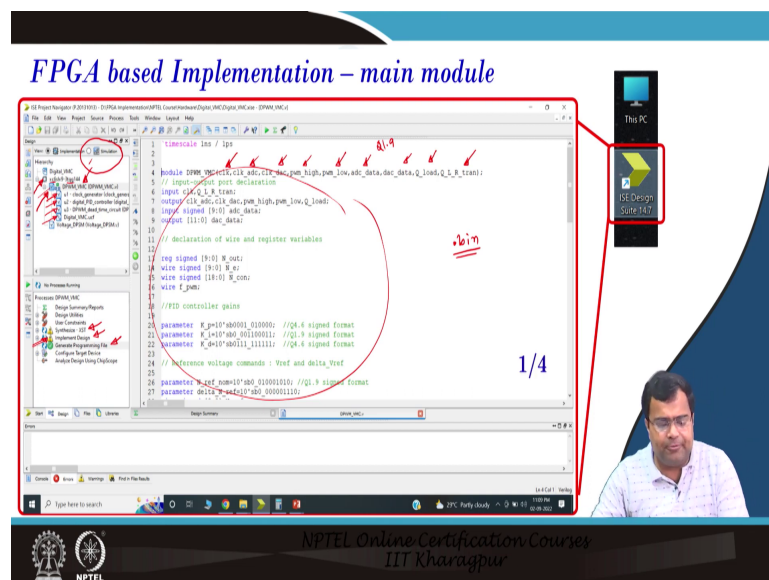
So, this is the realization where we are using all the inside logic and we have demonstrated all the Verilog codes and etcetera.

(Refer Slide Time: 04:04)



And we discussed in lecture number 57 what is the step for plugging in this you know this computer and then how to dump the code. How to compile the code? How do synthesize then how generate the programming file and how do dump it into the actual FPGA?

(Refer Slide Time: 04:27)



So, these steps are discussed in lecture 57, here we are only recollecting with our code. So, it is a digital voltage mode control. So, you can see this is the main module which should be kept at the top module. So, this main module is called the sub-module which is a clock generator, then a digital PID controller DPWM plus dead time circuit, and there is a user

constant file because this file will link with the actual io pin with their address. After all, we are taking the clock data 100 megahertz clock through this pin.

We are sending the ADC clock from here we are sending the DAC clock though DAC is not required. We are sending the high side gate signal it is going to the driver, we are sending the low side gate signal. We are capturing the ADC data, which is you know 2s complement, Q1 dot 9. We are sending that data, in this case, we are sending 0 data because we are not using that, but since the design is made we are just making this the in our module.

But we can drop it. This is used for the load transient; that means, the Q load is 1, which means it will give the load step-up transient Q load 0 means load step-down transient. And this is to select the transient there will be an external switch if you set it to one position it will be a load if you set it to another position it will be a reference transient. And all this code we have discussed in a lecture I think 72 and 73. So, we are not going to discuss again here what we are going to show the top module.

So, initially before going to implement we have shown some case studies of how to simulate. Because in lecture number 80, I remember that I think lecture number 80, I think it is not 80 6, 70. I think in lecture 70 we discussed how to simulate you know a PWDM block with a gate signal and dead time. So, then we have to go to the simulation mode. And we can check all the timing diagrams.

Once each block is simulated and tested then you are ready to go for the implementation part. Once you go to implementation you have to make sure which block is your top module. And then if you select the top module these are the step the synthesis step it will go then the implementation of the design and the synthesis will generate rtl ok. And the implementation how this rtl, because Verilog is a platform independent; means, the same code can be used in a synopsis and another digital tool synthesizes. Design one digital ic.

But when it goes to FPGA. So, this step requires this implementation stage is critical it will take that i c and route that FPGA pin, because FPGA we have discussed lookup table, configuration logic block then programmable i o pin. So, all this interconnect of the implementation will be done in this implementation digitally and we have to be very careful about this what FPGA we are using.

So, we are using this particular part number of the FPGA of this package this must be consistent otherwise it will give you the wrong result because this will link it will take the library of the Xilinx ISE and it will correspondingly take the library of the relevant ic and then it will make i o interconnection of this. Once it is done then it will generate the program file and we have discussed that we want to generate a dot bin file binary file and this will be loaded in using another know software that we have discussed in lecture 57.

(Refer Slide Time: 08:11)

FPGA based Implementation – main module

```
23 // reference voltage commands - typef and delta_typef
24
25 parameter M_ref_name="100 (10000000) //Q1.9 signed format
26 parameter M_ref="1000000000 //Q1.9 signed format
27 parameter M_ref_16="1000000000 //Q1.9 signed format
28 wire signed [15:0] M_ref;
29 reg signed [15:0] M_ref_temp;
30
31 // Output voltage from ADC and generate error voltage
32
33 always@(posedge f_psm) begin // Capturing output voltage samples
34   M_ref_load_data[15:0] := M_ref;
35 end
36
37 assign M_ref_ref_0_out;
38
39 //Clock generation circuit
40 clock_generator cl(f_clkclk0, f_adc_clockclk_adc0, f_dac_clockclk_dac0, f_ref_f_psm);
41
42 // digital PID controller
43 digital_PID_controller u2(f_psm(f_psm), M_ref_16, 1, reset0, M_con0_con0, K_p0(p0), K_i0(i0), K_d0(d0));
44
45 //SPWM and Deadtime
46 SPWM_dead_time_circuit u1(clkclk0, f_pwm(f_psm), con0_out0_con0, q_0(pwm_high), q_1(pwm_low));
47
48 // creating transient events
49
50
```

2/4

NPTEL Online Certification Courses
IIT Kharagpur

(Refer Slide Time: 08:16)

FPGA based Implementation – main module

```
47 // clearing transient events
48
49
50 parameter M_tran=0;
51 reg [31:0] counter;
52 reg Q_tran;
53 wire Q_tran_type_1_1st;
54
55 assign Q_tran_type_0_1_1st; // 0 for load tran, 1 for ref tran
56
57 always @(posedge f_psm) begin
58   if (counter==0) Q_tran := 0;
59   Q_tran := counter;
60   M_ref_temp_0_ref_0;
61   counter := counter + 1;
62 end
63
64 Q_tran := counter == M_tran;
65 M_ref_temp_0_ref_0;
66 counter := counter;
67 end
68
69 wire begin;
70 M_ref_temp_0_ref_0;
71 counter := counter + 1;
72 end
73
74 end
75
```

3/4

NPTEL Online Certification Courses
IIT Kharagpur

(Refer Slide Time: 08:30)

FPGA based Implementation – main module

```
61 counter<=counter+1;
62 end
63 else if (counter==M) start begin
64   Q_tick<=0;
65   M_ref_tick<=M_ref_tick_ref_num;
66   counter<=0;
67 end
68 else begin
69   Q_tick<=1;
70   M_ref_tick<=M_ref_tick_ref_num;
71   counter<=counter+1;
72 end
73 end
74 assign Q_tick<=Q_tick_type ? 0 : Q_tick;
75 assign M_ref<=M_ref_tick_type ? M_ref_tick_ref_num;
76 // mode detect
77
78 module DPWM_DPWM;
79   input PWM_Q_load;
80   output reg PWM_Q_load;
81   output reg PWM_PWM;
82   output reg PWM_PWM;
83   output reg PWM_PWM;
84   output reg PWM_PWM;
85   output reg PWM_PWM;
86   output reg PWM_PWM;
87 endmodule
```

NPTEL Online Certification Courses
IIT Kharagpur

So, this is the first step if we show the whole code one by one we are changing only this file because there is a 4 you know these blocks. So, transient creation. These are called clock generator digital PID, DPWM, and finally, this is the end of the 4th one it's the main module. So, it will be 4 by 4.

(Refer Slide Time: 08:30)

FPGA based Implementation – clock generation

```
1 timescale 1ns / 1ps
2
3 module clock_generator(f_clk, f_adc_clock, f_dac_clock, f_sw);
4
5   input f_clk;
6   output reg f_adc_clock, f_dac_clock, f_sw;
7   parameter M_adc=10;
8   parameter M_dac=10;
9   parameter M_sw=10;
10  reg [3:0] counter, counter2;
11
12  initial begin
13    counter=0;
14    counter2=0;
15  end
16
17  always@posedge f_clk begin // Switching frequency clock
18    if (counter==0) begin
19      f_sw<=1;
20      counter<=counter+1;
21    end
22  end
23
24  else if (counter==M_adc) begin
25    f_adc_clock<=1;
26    counter<=0;
27  end
28
29  else if (counter==M_dac) begin
30    f_dac_clock<=1;
31    counter<=0;
32  end
33
34  else if (counter==M_sw) begin
35    f_sw<=0;
36    counter<=0;
37  end
38
39  end
```

NPTEL Online Certification Courses
IIT Kharagpur

Next, if you go to a separate clock generator. You see this implementation will not come because it is not the top module, but we have discussed the clock generation block.

So, I am not going to discuss detail. This block we have presented in lecture 73 even we have presented in I think seventy-seven also for current mode control.

(Refer Slide Time: 08:55)

FPGA based Implementation – clock generation

```
27 end
28 wire begin
29 f_clk0;
30 counter1<counter1+1;
31 end
32 end
33
34 always@posedge f_clk0 begin // ADC and DAC clock
35 if (counter1<0) begin
36 f_clk0<0;
37 f_clk1<0;
38 counter1<counter1+1;
39 end
40 else if (counter1==M_adc) begin
41 f_clk0<0;
42 f_clk1<0;
43 counter1<0;
44 end
45 end
46 wire begin
47 f_clk0<0;
48 counter2<counter2+1;
49 end
50 end
51
52 endmodule
53
54
```

2/2

NPTEL Online Certification Courses
IIT Kharagpur

So, these things we have discussed, but it is just showing the screenshot of the FPGA coding.

(Refer Slide Time: 09:00)

FPGA based Implementation – digital PID controller

```
1 timescale 1ns / rps
2
3 module digital_pid_controller(f_pwm_w_err,M,Kp,Ki,Kd,f_reset);
4   input signed [18:0] M_err,M_pwm_w_err;
5   input f_pwm_i_reset;
6   output signed [18:0] w_con;
7
8   wire signed [18:0] M_prop_temp,M_int_temp;
9   reg signed [18:0] M_err_temp;
10  wire signed [18:0] M_prop_w_err,M_int_w_err;
11  reg signed [18:0] M_int_w_err,M_err_temp;
12  reg signed [18:0] M_err_prev;
13
14  parameter M_int_max=18'b111111111111111111;
15
16  assign M_prop_temp = M_err_w_err;
17  assign M_int_temp = M_err_w_err;
18  always@posedge f_pwm0 begin
19    M_err_temp = M_err_w_err;
20    M_err_prev = M_err_w_err;
21  end
22
23  assign M_prop = (M_prop_temp[18:0]); // in Q4.15
24  assign M_int_temp = (M_int_temp[18:0]); // in Q4.18
25  assign M_err = (M_err_temp[18:0]); // in Q4.15
26
27  always@posedge f_pwm0 begin
```

1/2

NPTEL Online Certification Courses
IIT Kharagpur

Then we are going to talk about digital PID controllers. This block is also discussed in our presentation which was probably 72 and 73 we have discussed. 73 was a digital PID controller.

(Refer Slide Time: 09:14)

FPGA based Implementation – digital PID controller

```
21 end
22
23 assign M_prop = (K_prop*temp[18:0]); // in Q4.15
24 assign M_int_temp2 = (K_int_temp[18:0]); // in Q1.18
25 assign M_der = (K_der_temp[18:0]); // in Q4.15
26
27 always@posedge f_pwm begin
28   M_int_temp2<=M_int_temp;
29   M_int_temp3<=M_int_temp2;
30 end
31
32 assign M_int_inst=(M_int_temp4[18],M_int_temp4[18],M_int_temp4[18],M_int_temp4[18:0]);
33
34 always@posedge f_pwm begin
35   if (!count)
36     M_int<=0;
37   else if (M_int_inst[M_int_max])
38     M_int<=M_int_max;
39   else if (M_int_inst[M_int_min])
40     M_int<=M_int_min;
41 end
42
43 assign M_conM_propM_intM_der;
44
45
46 endmodule
47
48
```

2/2

NPTEL Online Certification Courses
IIT Kharagpur

(Refer Slide Time: 09:18)

FPGA based Implementation – DPWM & deadtime

```
1 timescale 1ns / 1ps
2
3 module DPWM_dead_time_circuit[clk_f_pwm_cnt_out_Q_M_Q_Lv];
4   input clk_f_pwm;
5   input signed [18:0] cont_out; // Q4.15
6   output Q_M_Q_Lv;
7
8   reg Q_pwm;
9   wire set;
10  reg Q_pwm;
11  reg [18:0] shiftw;
12  reg pwm_delay;
13  wire signed [18:0] controller_output;
14  reg signed [18:0] counter //Q4.8
15
16 initial begin
17   counter=0;
18 end
19
20 assign controller_output=cont_out[18:7];
21
22 always@posedge clk begin
23   if (counter==0) begin
24     Q_pwm=1;
25     counter=counter+1;
26   end
27   else if (counter<controller_output) begin
28     Q_pwm=1;
29     counter=counter+1;
30   end
31 end
32
```

1/2

NPTEL Online Certification Courses
IIT Kharagpur

So, we discuss it in detail, we are not going to discuss it again. This is again a sub-module of the main module, then the DPWM dead type this block we have discussed in the context in lecture number 70 where we have also simulated; that means if you have a pulse width modulator and if you compare with the fixed reference, that will generate a duty ratio command and that can be used to generate your high side and low side gate signal with suitable dead time. That also we have discussed.

(Refer Slide Time: 09:54)

FPGA based Implementation – DPWM & deadtime

```
28 Q_pwm=1;
29 counter=counter+1;
30 end
31 else if (counter==499) begin
32   Q_pwm=0;
33   counter=0;
34 end
35 else begin
36   Q_pwm=0;
37   counter=counter+1;
38 end
39 end
40
41 always@(posedge clk)
42 begin
43   #10ns Q_pwm;
44   #10ns Q_pwm;
45   #10ns Q_pwm;
46   #10ns Q_pwm;
47   #10ns Q_pwm;
48   #10ns Q_pwm;
49 end
50
51 assign Q_pwm = Q_pwm;
52 assign Q_pwm = Q_pwm;
53
54 endmodule
```

2/2

NPTEL Online Certification Courses
IIT Kharagpur

So, there is a redundant block we are not using digital PSM. When we go to the multi-mode we will be using this block, but here dead time we have discussed different codes of the dead time. We have discussed this in detail in lecture number 72 the Verilog code. So, this part is over.

(Refer Slide Time: 10:02)

FPGA based Implementation – UCF file

```
1 set "DPWM" UCF = R40A;
2 set "DPWM" UCF = R41;
3 set "DPWM" UCF = R42;
4 set "DPWM" UCF = R43;
5 set "DPWM" UCF = R44;
6 set "DPWM" UCF = R45;
7 set "DPWM" UCF = R46;
8 set "DPWM" UCF = R47;
9 set "DPWM" UCF = R48;
10 set "DPWM" UCF = R49;
11 set "DPWM" UCF = R50;
12 set "DPWM" UCF = R51;
13 set "DPWM" UCF = R52;
14 set "DPWM" UCF = R53;
15 set "DPWM" UCF = R54;
16 set "DPWM" UCF = R55;
17 set "DPWM" UCF = R56;
18 set "DPWM" UCF = R57;
19 set "DPWM" UCF = R58;
20 set "DPWM" UCF = R59;
21 set "DPWM" UCF = R60;
22 set "DPWM" UCF = R61;
23 set "DPWM" UCF = R62;
24 set "DPWM" UCF = R63;
25 set "DPWM" UCF = R64;
26 set "DPWM" UCF = R65;
27 set "DPWM" UCF = R66;
28 set "DPWM" UCF = R67;
29 set "DPWM" UCF = R68;
30 set "DPWM" UCF = R69;
31 set "DPWM" UCF = R70;
32 set "DPWM" UCF = R71;
33 set "DPWM" UCF = R72;
34 set "DPWM" UCF = R73;
35 set "DPWM" UCF = R74;
36 set "DPWM" UCF = R75;
37 set "DPWM" UCF = R76;
38 set "DPWM" UCF = R77;
39 set "DPWM" UCF = R78;
40 set "DPWM" UCF = R79;
41 set "DPWM" UCF = R80;
42 set "DPWM" UCF = R81;
43 set "DPWM" UCF = R82;
44 set "DPWM" UCF = R83;
45 set "DPWM" UCF = R84;
46 set "DPWM" UCF = R85;
47 set "DPWM" UCF = R86;
48 set "DPWM" UCF = R87;
49 set "DPWM" UCF = R88;
50 set "DPWM" UCF = R89;
51 set "DPWM" UCF = R90;
52 set "DPWM" UCF = R91;
53 set "DPWM" UCF = R92;
54 set "DPWM" UCF = R93;
55 set "DPWM" UCF = R94;
56 set "DPWM" UCF = R95;
57 set "DPWM" UCF = R96;
58 set "DPWM" UCF = R97;
59 set "DPWM" UCF = R98;
60 set "DPWM" UCF = R99;
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, the most important part of the implementation is the user constant file and we have discussed in lecture number 57 how to right-click on this file and add this implementation ok.

(Refer Slide Time: 10:18)

FPGA based Implementation – Programming file

```
1 // Verilog code for FPGA implementation
2
3 module DPMAC(ctrl_clk_adc, ctrl_dec_pwm_high_pwm_low_adc_data, dac_Q_load_Q_r_tran);
4 // input-output post-declaration
5 // input: ctrl_clk_adc;
6 // output: ctrl_clk_adc; ctrl_dec_pwm_high_pwm_low_adc_data;
7 // output: dac_Q_load_Q_r_tran;
8 // output: ctrl_clk_adc; ctrl_dec_pwm_high_pwm_low_adc_data;
9 // output: dac_Q_load_Q_r_tran;
10 // declaration of wires and registers variables
11
12 reg signed [19:0] W_out;
13 wire signed [19:0] W_in;
14 wire signed [19:0] W_out;
15 wire signed [19:0] W_in;
16 wire [1:0] W_out;
17
18 //PID controller gain
19 parameter K_p=10'd0001; //Q1.4 signed format
20 parameter K_i=10'd00; //Q1.9 signed format
21 parameter K_d=10'd01111111; //Q1.4 signed format
22 // Reference voltage commands : Vref and delta_Vref
23 parameter W_ref=10'd00010000; //Q1.9 signed format
24 parameter delta_W_ref=10'd00000110;
25
26 endmodule
```

NPTEL Online Certification Courses
IIT Kharagpur

Then once this generating program generation is over, what are we going to do we have to load to the target device, and that part we will be showing in the live demonstration. How is it doing and finally, what is happening in the actual circuit?

(Refer Slide Time: 10:37)

Live Demonstration of FPGA based Digitally VMC Buck Converter

NPTEL Online Certification Courses
IIT Kharagpur

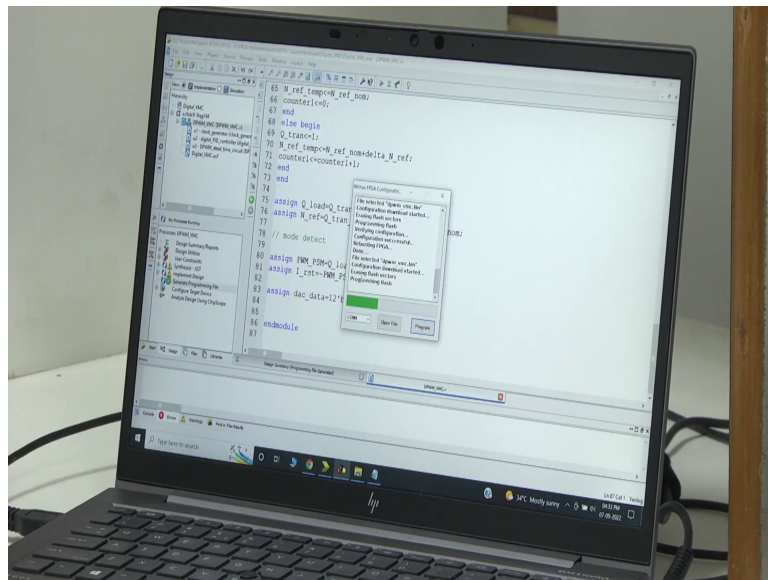
And this is the picture of the live demo, but now we are going to consider all the steps that we have discussed in the live demonstration.

So, now we are going to demonstrate the digital voltage mode control experiment which we have already presented in class. So, here we have already discussed in lecture number 91 about the steps for know synthesizing we have demonstrated all the Verilog code then we have told what are the step you know first you have to synthesize and then implement the design.

So, as if we have all the modules here and you say file dead time digital PID controller clock generator and the main module. So, we are first generating the program file we will rerun it. And this is the Xilinx ISE software tool and we have already discussed what device we are using everything in detail. Now, you see one by one the stage is executing like synthesis is over then now it is going to implement.

So, the synthesis will generate RTL logic then implementation will try to map into the target FPGA device, the SDL code. That is over now once it is generated it is generating the dot bin file, which will be dumped into the FPGA. So, now, that stage is coming. So, now, we are going to. Once, this process once it is over now we are going to this Mimas FPGA configuration tool we have already discussed the steps and we are taking the file dot bin file.

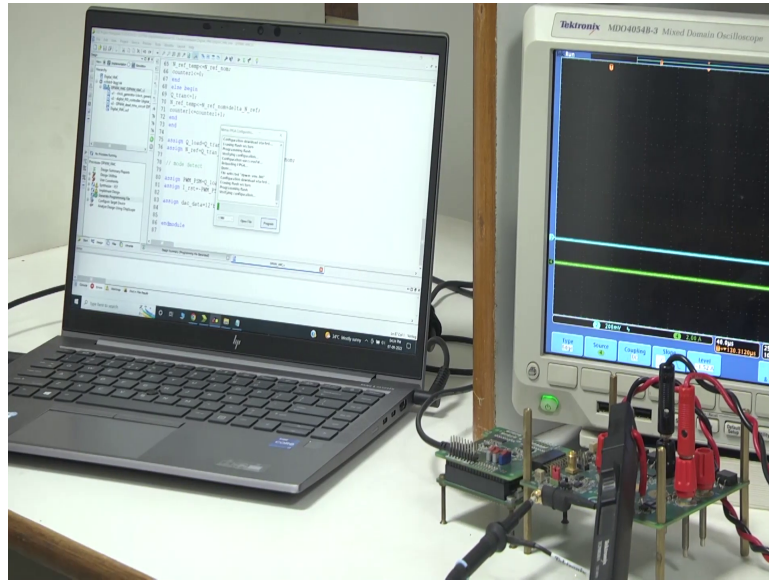
(Refer Slide Time: 12:14)



So, this is the voltage mode control file you can see the dot bin file. Once you select it will show that it is selected dot bin file now we are going to program. So; that means, the well selected file is DPWM underscore VMC dot bin now it is programming the bin file is getting

programmed into this FPGA device. So, it is dumping. Once it is dumped the code is dumped then it will verify once.

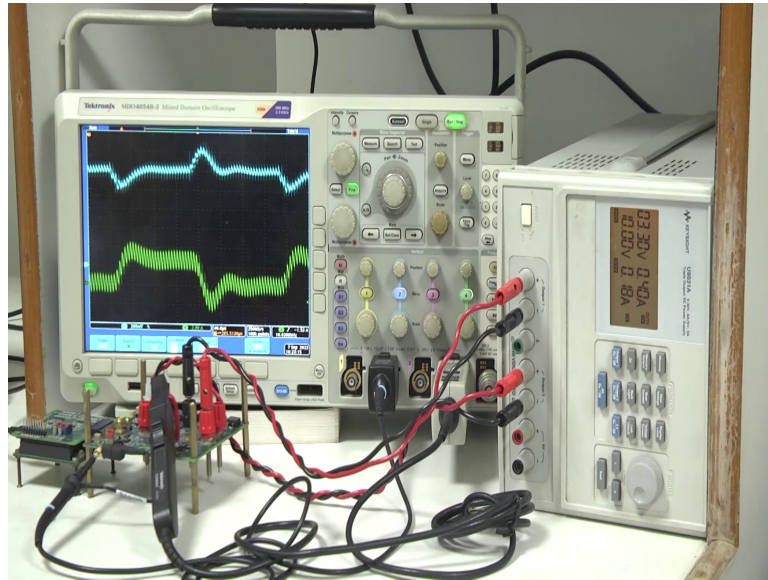
(Refer Slide Time: 12:45)



And then you know there is a red light at the bottom. So, it is not visible from here, because it is in between two boards. The top board is our signal conditioning board, the bottom is FPGA.

So, if it is FPGA you know red led light which will display when it is dumping is over. The led light will stop now it is showing the program dumping process is going on. So, now, it will verify the process; which means, whether it has been correctly dumped or not. So, that is the final stage. Once it is over then we will move to we will turn on the power supply.

(Refer Slide Time: 13:22)



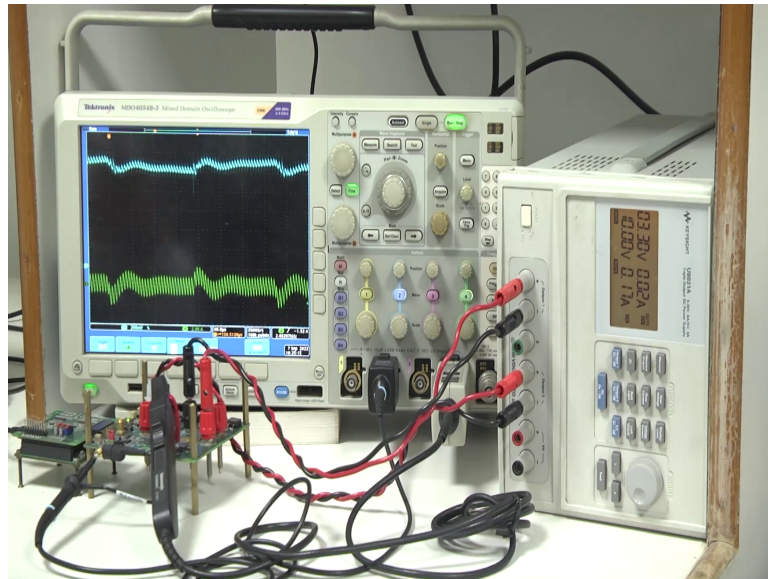
Now, we are going to turn on the supply. So, first, we will turn on the second supply that is the powering for the signal conditioning circuit, then buffer circuit IDC and all these, because we have ldo in the board which will generate derive all the required power supply from this 10 volt. Now, we are turning on the main power supply which is the input voltage of the buck converter and as we have discussed we are taking 3.3 volt as the input, but you can also increase the input voltage if there is no problem.

Now, we are going to first set based on the trigger. So, it is I am just showing here the load transient response. You know the load transient response. We have not yet designed the compensator properly. We have chosen some value, but for the given value this is the transient performance and in the 11th week we are going to design first we need to validate how to realistically model our MATLAB model to match closely match with the experimental result whether they closed or there is a deviating we will also show the validation process.

And there followed by that we will show the design step, but here imagine that we have somehow designed the controller and the controller values are already displayed you know we have shown them in class. Now, with that controller value, we are making a load transient where the resistance is initially it was connected you can see 13.5 ohm resistance now this RSW which is 0.33 ohm that is connected in parallel here. As a result, you can see a load step-up transient.

So, if we know to stop it this is the step-up transient where the load step is happening here this is the undershoot current overshoot and then there is an overshoot due to the load steps down transient. Now, we also have and we have discussed we have shown this you know we are going to show the results in more detail in the subsequent lecture. We also have a provision for changing to reference transient.

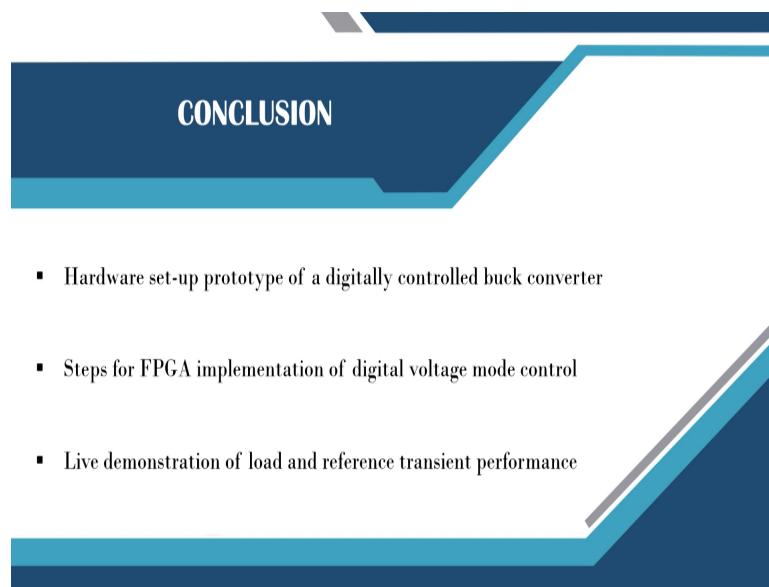
(Refer Slide Time: 15:24)



So, there is a switch and we have discussed there is a called switch called for, which will select the transient type. So, when it is 0 that will take the reference transient when it is 1 or vice versa, sorry it is 0 means load transient, and 1 means reference transient. So, here we are showing the reference transient. So, first, it is stepping down; that means, it was 1.1 volt you can see 0.2 volt, 1, 2, 3, 4, 5. So, this is the 1 volt. This is 1.1 volt, then it is going down to 1 volt then again it is going to 1.1 volt.

So, there is a periodic reference state transient which is happening from 1 volt to 1.1 volt and back to 1 volt and so on and this is the response. So, now, we have shown how to you know to run this voltage mode control now if we increase the input voltage accordingly you know you can check the waveform and how it is getting affected and we can design for various input voltage conditions and we will discuss the design process in the subsequent lecture.

(Refer Slide Time: 16:32)



CONCLUSION

- Hardware set-up prototype of a digitally controlled buck converter
- Steps for FPGA implementation of digital voltage mode control
- Live demonstration of load and reference transient performance

Now, we will move to the class. So, now, we have just concluded our live demonstration and we want to summarize that we have discussed the hardware setup prototype digitally controlled converter. We have shown steps for FPGA implementation of digital voltage mode control and we have also considered a live demonstration of load and reference transient performance that is it for today.

Thank you very much.