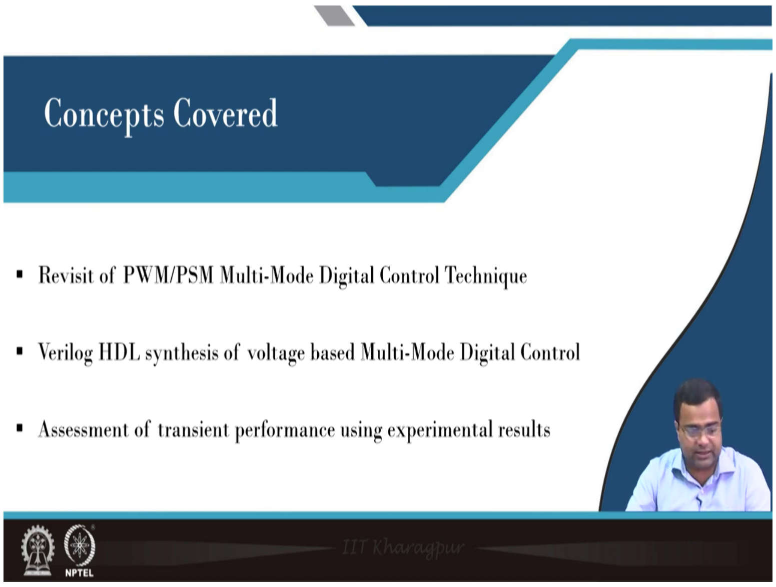
Digital Control in Switched Mode Power Converters and FPGA-based Prototyping

Dr. Santanu Kapat
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur

Module - 12
Hardware Case Studies of Advanced Digital Control Techniques and Course Summary
Lecture - 114
Verilog HDL-based FPGA Prototyping of PWM/PSM Multi-Mode Digital Control

Welcome back. This is the continuation of the previous lecture. In this lecture, we are going to prototype and we are going to show how Verilog HDL programming and FPGA prototyping with hardware experiment of PWM PSM my multi-digital control.

(Refer Slide Time: 00:40)



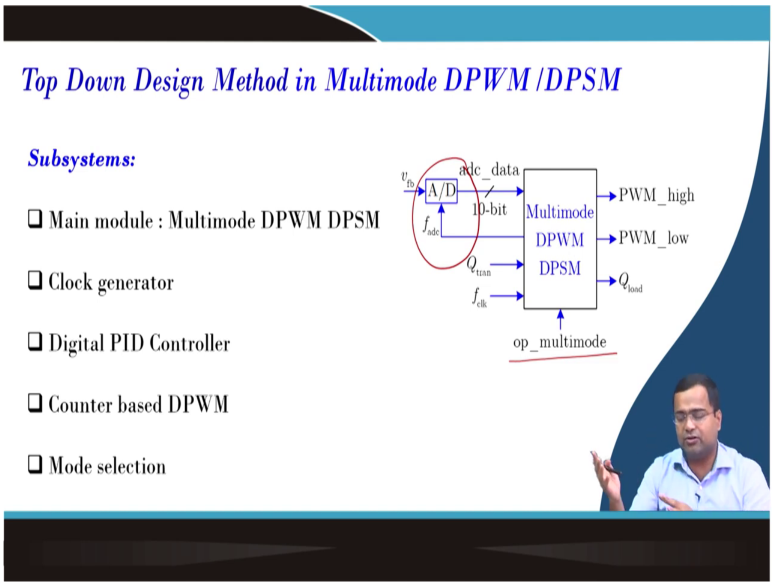
Concepts Covered

- Revisit of PWM/PSM Multi-Mode Digital Control Technique
- Verilog HDL synthesis of voltage based Multi-Mode Digital Control
- Assessment of transient performance using experimental results

IIT Kharagpur

So, here we are going to revisit our PWM PSM multi-mode digital control technique, then Verilog HDL synthesis of voltage-based multi-mode digital control, and then assessment of transient performance using experimental results.

(Refer Slide Time: 00:54)



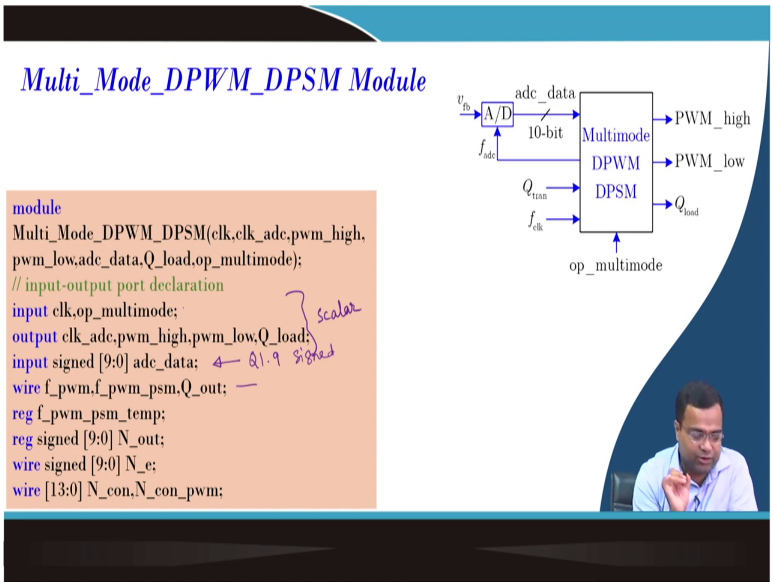
Top Down Design Method in Multimode DPWM /DPSM

Subsystems:

- ❑ Main module : Multimode DPWM DPSM
- ❑ Clock generator
- ❑ Digital PID Controller
- ❑ Counter based DPWM
- ❑ Mode selection

So, if you recall our last lecture I mean the previous lecture where we discussed modules you know in the context of digital multi-mode digital PWM PSM control. So, here you know we are taking the A to D converter; that means, the voltage loop and we are generating the gate signals and all this discussion and this is the option for mode selection. If it is 0, it will throughout take the DPWM. If it is 1, then at high load we are setting PWM, and at light load we are,e going to pulse skipping modulation.

(Refer Slide Time: 01:28)



Multi_Mode_DPWM_DPSM Module

```
module
Multi_Mode_DPWM_DPSM(clk,clk_adc,pwm_high,
pwm_low,adc_data,Q_load,op_multimode);
// input-output port declaration
input clk,op_multimode;                        } scalar
output clk_adc,pwm_high,pwm_low,Q_load;        }
input signed [9:0] adc_data;  ← Q1.9 signed
wire f_pwm,f_pwm_psm,Q_out;  —
reg f_pwm_psm_temp;
reg signed [9:0] N_out;
wire signed [9:0] N_e;
wire [13:0] N_con,N_con_pwm;
```

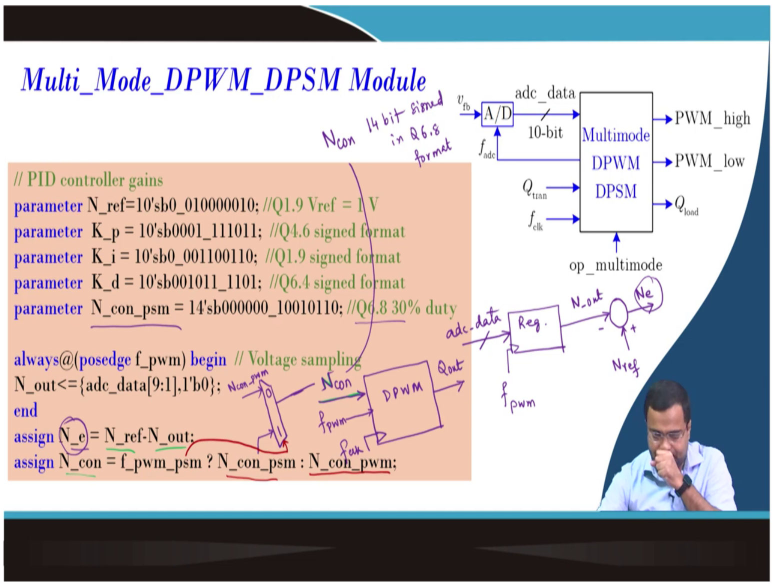
Now, in this architecture, we are building the main module. So, the main module if you see this is the main module and this main module name is here, this is the name of the main module. Now what we are going to see we want to map all the input-output pins; that means, if you see the ADC data. So, this is our ADC data ok.

So, if we take once more ADC data. Next, we want to show what is the adc clock. So, this is the ADC clock ok, then what is the main clock? So, the main clock is here, the main clock is here ok. Then Q transient, we are not giving this option here. We are setting that because this was set for load and reference transient, but since we are taking only load transient.

So, this option for this particular case we are not considering then what else? Then we are talking about this Q load the option mode option. So, this is the option mode option then PWM high is ok. Let me go into the other part PWM high we can take from here to here then PWM low. So, this will go from here to here then ADC data we have already shown Q load this is the Q load ok.

So, that means, now we have mapped all the input outputs which are you know external interface of this main module and which are going to the FPGA different pins ok. Now we have to define this as the scalar input and scalar output. So, these are all scalar and this is a vector input Q 1 dot 9 signed format and this is we do not we are not using DAC. So, these are wires all this definition.

(Refer Slide Time: 03:39)



Multi_Mode_DPWM_DPSM Module

```
// PID controller gains
parameter N_ref=10'sb0_010000010; //Q1.9 Vref = 1 V
parameter  K_p = 10'sb0001_111011;  //Q4.6 signed format
parameter  K_i = 10'sb0_001100110;  //Q1.9 signed format
parameter  K_d = 10'sb001011_1101;  //Q6.4 signed format
parameter  N_con_psm = 14'sb000000_10010110; //Q6.8 30% duty

always@(posedge f_pwm) begin  // Voltage sampling
N_out<={adc_data[9:1],1'b0};
end
assign N_e = N_ref-N_out;
assign N_con = f_pwm_psm ? N_con_psm : N_con_pwm;
```

Now, what we are going to do, here we are taking the output voltage and we have discussed whatever ADC data is coming we have discussed multiple times. So, this is our ADC data, this is the register and we have a clock that is our switching frequency clock, or where we are calling as a PWM clock you can say.

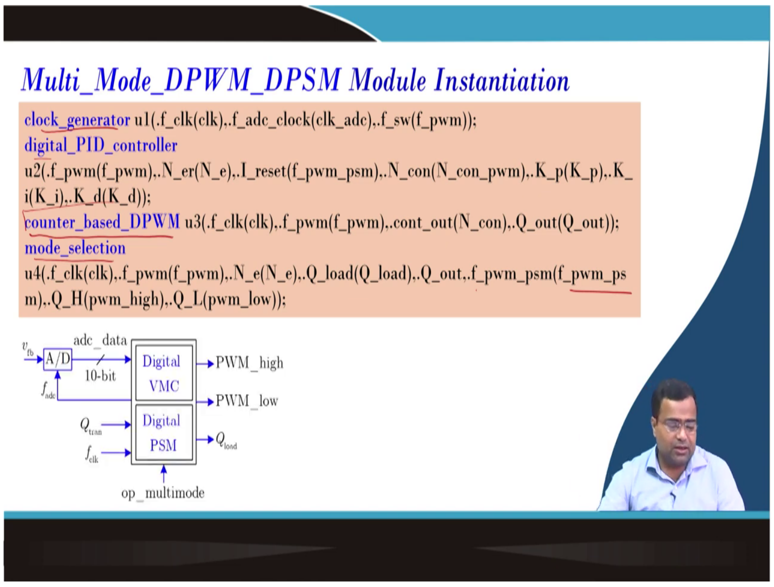
This is the PWM clock, this is you can say f pwm f pwm, and the data which is going out is my you know n out; that means, the output and this data were subtracted from the reference command which is N ref and it is generating the error N e which is here. So, this is the error, this we are discussing and this is what we are talking about this reference minus out.

Now, the controller output means we are going to use DPWM and we will discuss it. So, this structure that means, what we are going to do if we consider this N con ok. So, let me take a different color which is our N con, this is going to our DPWM block our DPWM block. So, in the case of PWM this N con will be the output of the controller. In case of pulse skipping during charge pulse, we are giving a fixed duty ratio ok and we are we need a high-frequency clock and this DPWM also we are taking PWM clock.

Because we want to reset this counter based on the edge of this clock and the output of this is coming to be Q out that we will discuss and this N con for this picture it is clear that it is the output of a mux 0 and 1 where this 1 is the N con psm N con PWM; that means, the output of the pwm controller and this is the N con psm ok and what is the select line.

The select line here if you use a different color. So, the select line here is this one; that means, if the select line is 0 it will take PWM. If the select line is 1, it will take this N psm. So, this is going to be digital.

(Refer Slide Time: 06:22)



Multi_Mode_DPWM_DPSM Module Instantiation

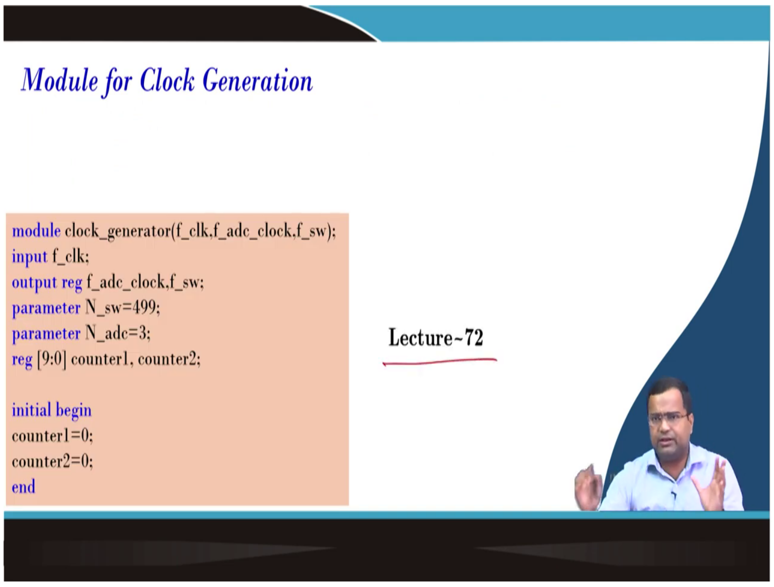
```
clock_generator u1(.f_clk(clk),.f_adc_clock(clk_adc),.f_sw(f_pwm));
digital_PID_controller
u2(.f_pwm(f_pwm),.N_er(N_e),.I_reset(f_pwm_psm),.N_con(N_con_pwm),.K_p(K_p),.K_
i(K_i),.K_d(K_d));
counter_based_DPWM u3(.f_clk(clk),.f_pwm(f_pwm),.cont_out(N_con),.Q_out(Q_out));
mode_selection
u4(.f_clk(clk),.f_pwm(f_pwm),.N_e(N_e),.Q_load(Q_load),.Q_out,.f_pwm_psm(f_pwm_ps
m),.Q_H(pwm_high),.Q_L(pwm_low));
```

Now, we are instantiating the clock generator circuit digital PID controller counter-based DPWM and the mode selection. So, I have just shown the overview of this counter base DPWM, but the mode selection will decide what output should be taken. Is it the output of the because it will generate finally, we are generating this PWM PSM flag signal from a transient detection circuit?

But in an actual commercial product whether to select PWM or psm that depends on some transient detection and some load estimation. There is a Nan number in the algorithm, but here we are making it simple just for understanding that we are assuming the load transient comment we are sending from our digital control platform by turning on and off the switch.

The switch resistance and we know that transient. So, whenever it is going to load transient high load it is directly moving into PWM. When it is going to light load then we are essentially operating the converter under PWM for a few cycles and then moving to psm and we will explain why we are doing that.

(Refer Slide Time: 07:34)



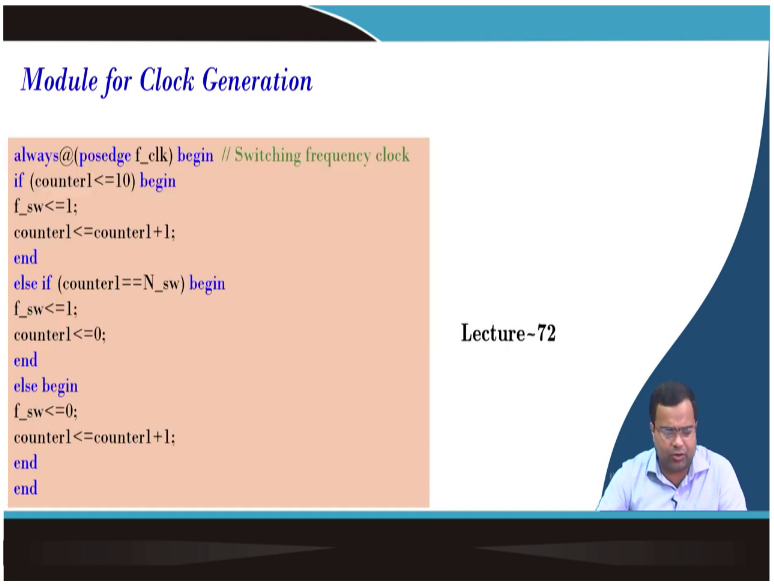
Module for Clock Generation

```
module clock_generator(f_clk,f_adc_clock,f_sw);
input f_clk;
output reg f_adc_clock,f_sw;
parameter N_sw=499;
parameter N_adc=3;
reg [9:0] counter1, counter2;

initial begin
counter1=0;
counter2=0;
end
```

Lecture-72

(Refer Slide Time: 07:40)

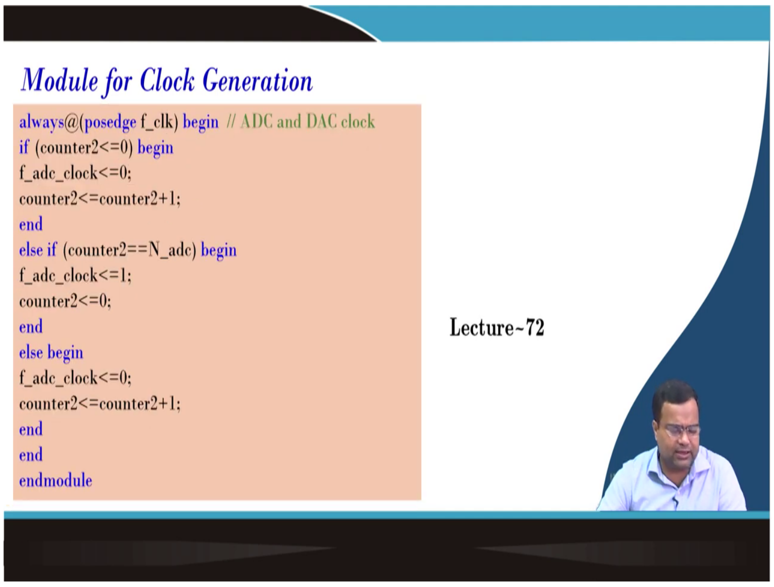


Module for Clock Generation

```
always@(posedge f_clk) begin  // Switching frequency clock
if (counter1<=10) begin
f_sw<=1;
counter1<=counter1+1;
end
else if (counter1==N_sw) begin
f_sw<=1;
counter1<=0;
end
else begin
f_sw<=0;
counter1<=counter1+1;
end
end
```

Lecture-72

(Refer Slide Time: 07:43)



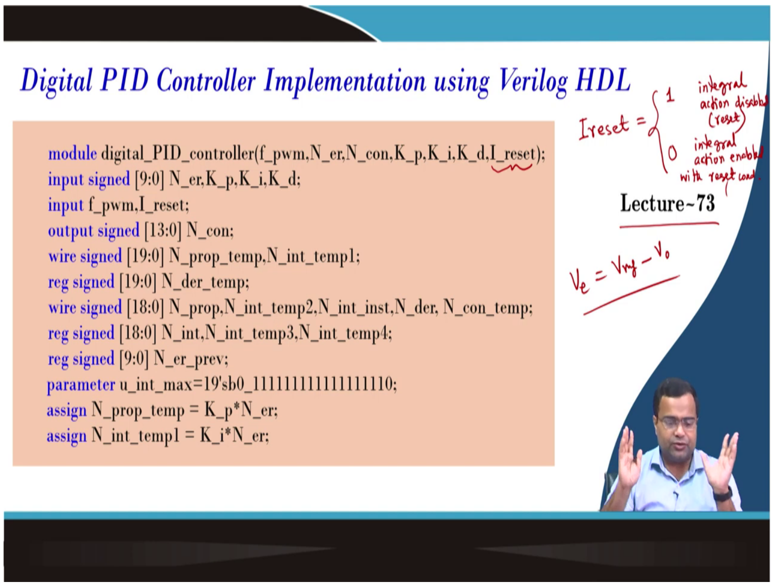
Module for Clock Generation

```
always@(posedge f_clk) begin // ADC and DAC clock
if (counter2<=0) begin
f_adc_clock<=0;
counter2<=counter2+1;
end
else if (counter2==N_adc) begin
f_adc_clock<=1;
counter2<=0;
end
else begin
f_adc_clock<=0;
counter2<=counter2+1;
end
end
endmodule
```

Lecture~72

So, this is the overall diagram and then the module for clock generation that we have discussed in lecture number 72 in detail. So, I am not going to repeat this block, these are all standard and we have already discussed them. The only thing we are not generating a DAC because we do not need a DAC clock. Otherwise, whenever we need a current mode control-based architecture where we need to use DAC we are using essentially the identical clock of ADC, but it is flexible and it is it can be programmed.

(Refer Slide Time: 08:03)



Digital PID Controller Implementation using Verilog HDL

```
module digital_PID_controller(f_pwm,N_er,N_con,K_p,K_i,K_d,I_reset);
input signed [9:0] N_er,K_p,K_i,K_d;
input f_pwm,I_reset;
output signed [13:0] N_con;
wire signed [19:0] N_prop_temp,N_int_temp1;
reg signed [19:0] N_der_temp;
wire signed [18:0] N_prop,N_int_temp2,N_int_inst,N_der, N_con_temp;
reg signed [18:0] N_int,N_int_temp3,N_int_temp4;
reg signed [9:0] N_er_prev;
parameter u_int_max=19'sb0_111111111111111110;
assign N_prop_temp = K_p*N_er;
assign N_int_temp1 = K_i*N_er;
```

$$\text{Ireset} = \begin{cases} 1 & \text{integral action disabled (reset)} \\ 0 & \text{integral action enabled with reset cond.} \end{cases}$$

Lecture~73

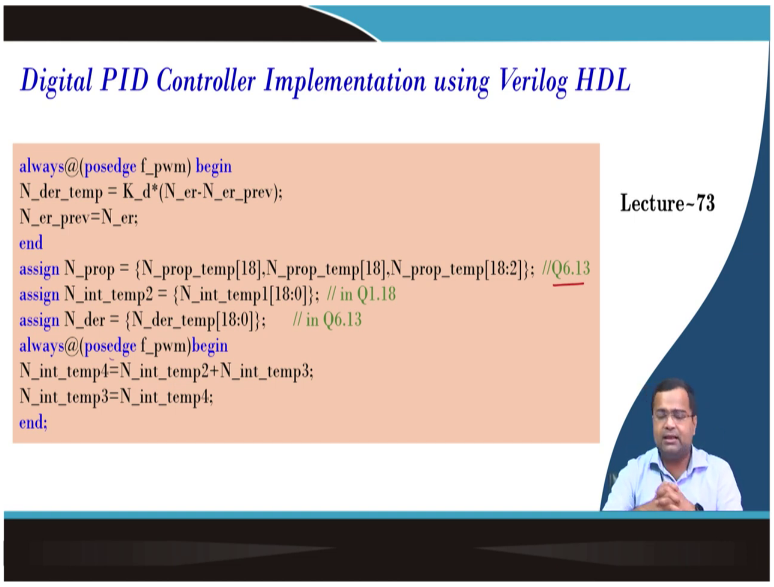
$V_e = V_{ref} - V_o$

Now, the digital PID controller, this also we have discussed in lecture number 73 where we have developed the Verilog HDL code for the digital PID controller. Everything else is the same except we are considering an additional reset circuit and this reset logic is that if this reset is 1, then I will say integral action disable, and if it is 0 that integral action enabled with reset condition; that means, integral action disable means it is reset it is set to 0. Wherever it is starting integral action then the initial value will be 0 and then it will keep on adding.

And this will also have some problems, particularly when you make for you know psm to PWM I will discuss that. But during the pulse skipping operation why we are resetting the integral action because the PWM integral action will accumulate all the errors, but under psm, since the average output voltage is always above the reference voltage because as per the pulse skipping logic if the output voltage just falls below V ref then the charge pulse start and then the voltage goes above the V ref and then it starts discharging.

So, all the time the output voltage is more or less above V ref and if we enable the reset integral action then the error will be negative all the time because it is the error voltage will we know the error is what it is V ref minus V 0. Since V 0 will always be above V ref. So, this negative quantity will be accumulated and this can cause some wind-up problems you know also during the transition the controller output will be very much negative. So, you want to avoid that.

(Refer Slide Time: 10:19)



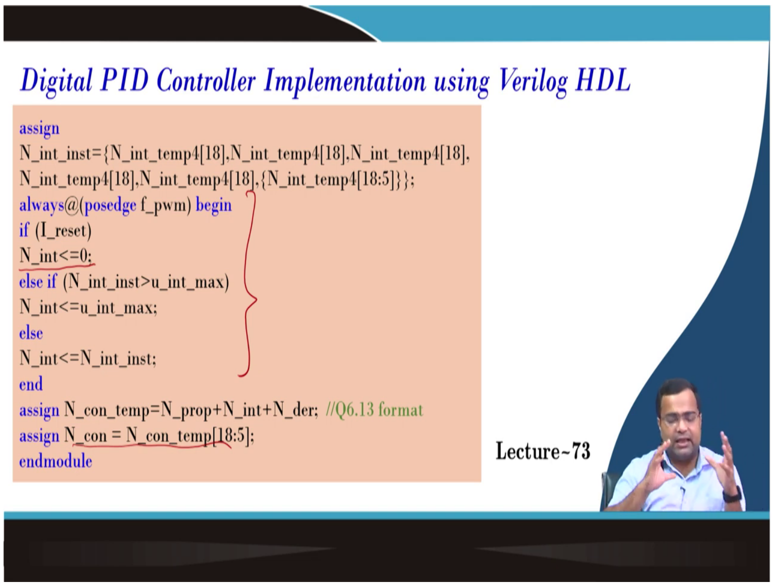
Digital PID Controller Implementation using Verilog HDL

```
always@(posedge f_pwm) begin
N_der_temp = K_d*(N_er-N_er_prev);
N_er_prev=N_er;
end
assign N_prop = {N_prop_temp[18],N_prop_temp[18],N_prop_temp[18:2]};  //Q6.13
assign N_int_temp2 = {N_int_temp1[18:0]};  // in Q1.18
assign N_der = {N_der_temp[18:0]};      // in Q6.13
always@(posedge f_pwm)begin
N_int_temp4=N_int_temp2+N_int_temp3;
N_int_temp3=N_int_temp4;
end;
```

Lecture~73

Other than that this PID controls ler everything we discussed in lecture number 73 and you know we have used here Q6 dot hereere we have used Q4 dot 15. But again if we understand all these Q formattings properly then there is no problem.

(Refer Slide Time: 10:36)



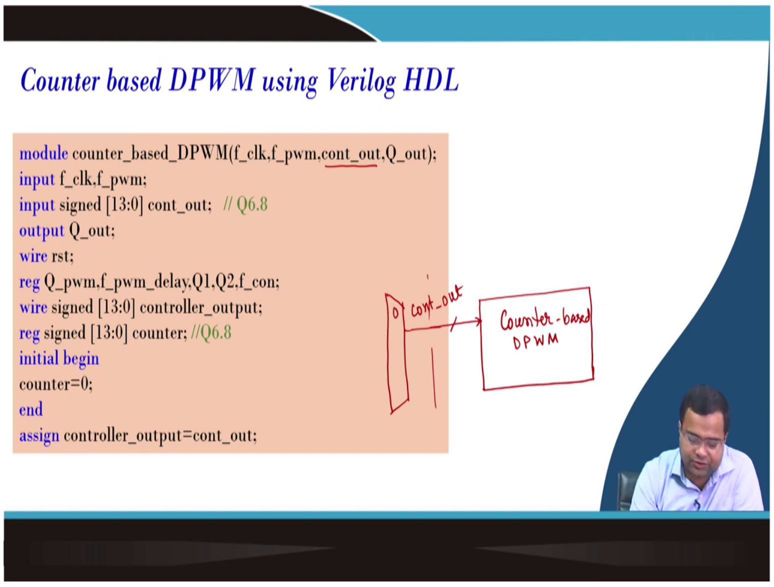
Digital PID Controller Implementation using Verilog HDL

```
assign
N_int_inst={N_int_temp4[18],N_int_temp4[18],N_int_temp4[18],
N_int_temp4[18],N_int_temp4[18],{N_int_temp4[18:5]}};
always@(posedge f_pwm) begin
if (I_reset)
N_int<=0;
else if (N_int_inst>u_int_max)
N_int<=u_int_max;
else
N_int<=N_int_inst;
end
assign N_con_temp=N_prop+N_int+N_der; //Q6.13 format
assign N_con = N_con_temp[18:5];
endmodule
```

Lecture-73

So, this you know the resizing the data, and then finally, we are generating the controller and here you will find the output of the integral which is this one. If you see this block the integral controller output will be 0 if the I reset is enabled otherwise it will take the regular integral action and that is the; that is what I have discussed. So, when the initiate will be the then integral will be disabdisabledit will be set to 0 the output of the integral.

(Refer Slide Time: 11:08)



Counter based DPWM using Verilog HDL

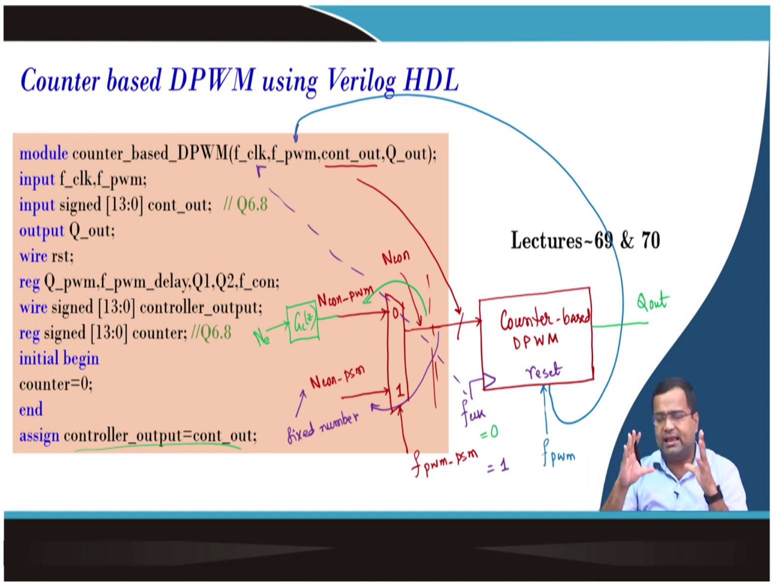
```
module counter_based_DPWM(f_clk,f_pwm,cont_out,Q_out);
input f_clk,f_pwm;
input signed [13:0] cont_out;  // Q6.8
output Q_out;
wire rst;
reg Q_pwm,f_pwm_delay,Q1,Q2,f_con;
wire signed [13:0] controller_output;
reg signed [13:0] counter; //Q6.8
initial begin
counter=0;
end
assign controller_output=cont_out;
```

Now, we are also using a counter-based DPWM because I have shown some overall diagrams and we have discussed this counter-based DPWM we have discussed in lecture number 69 and 70.

But here; that means, the counter-based DPWM will take a count output; that means, this logic simply the slight difference here is that in this logic I will say it is a counter-based DPWM input to this block here as per this it is like controller output. But this control output can be either this is just a definition local variable and if you go for the module instantiation block you will find that this count PWM.

The local variable is the N con; that means, this particular controller output is bigger in the broad sense yeah. So, in the broad sense this control output it is a local variable inside this block, but it is this thing 0 and so inside this module. So, inside this module, I would say its name is this.

(Refer Slide Time: 12:37)



Counter based DPWM using Verilog HDL

```
module counter_based_DPWM(f_clk,f_pwm,cont_out,Q_out);
input f_clk,f_pwm;
input signed [13:0] cont_out;  // Q6.8
output Q_out;
wire rst;
reg Q_pwm,f_pwm_delay,Q1,Q2,f_con;
wire signed [13:0] controller_output;
reg signed [13:0] counter; //Q6.8
initial begin
counter=0;
end
assign controller_output=cont_out;
```

Lectures~69 & 70

But, we are connecting locally. So, this name is N con; that means, this is just an interface. That means, the same data outside name is N con inside that is in the main module and which is interface or wire connected inside name is control output and what is this?

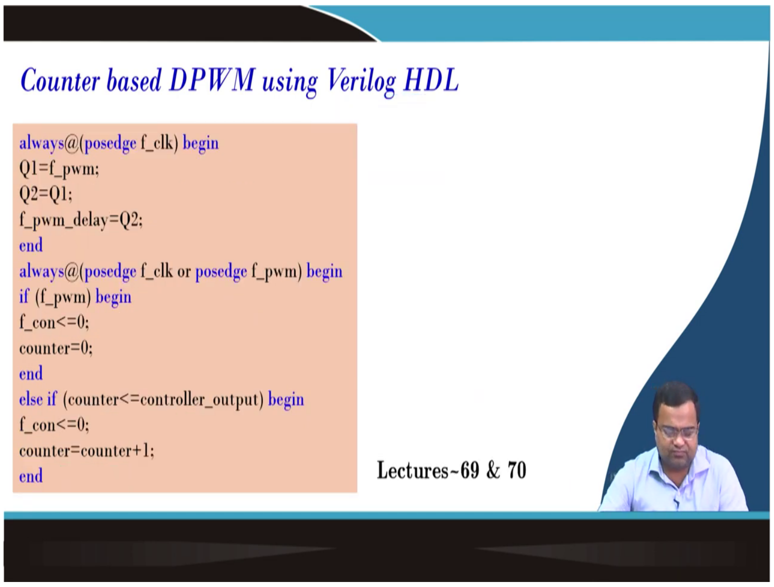
So, this is your N con pwm and this is your N con psm and there is a flag which is your f pwm psm; that means if the flag is 0; that means, if it is 0 then this gets connected to this and this is coming out from G c Z the control output and input is our error voltage; that means, the controller output will be connected if it is operating under pwm. But if it is 1; that means, if it is 1; that means, it will get connected here. If it is 0 if it is 1 and in that case it is psm and in psm, we are applying a fixed duty ratio.

So, it will be a fixed number, it is a fixed number in this ok and that we have parameterized at the very beginning if you go to the main module before that. So, this is the N con psm and this is what you have to match; that means, it is a total of 14-bit data. The N con is a controller output which is that this data is a 14-bit signed in Q6 dot 8 format. So, accordingly, N psm is, set and this is set to have a 30 percent duty ratio; that means, during on time we are giving a 30 percent duty ratio during the charge pulse ok.

Now, this counter output. So, then the counter also requires incrementing requires we are using a PWM clock edge because it is taking the PWM clock edge sorry counter is incremented with the high-frequency clock which is our f clock which is the high-frequency clock. So, this is the clock that is used and it also uses a reset counter is reset for that we are using this clock. This clock we are using as a reset; that means, reset is our f pwm and this is my output of this that is the output ok.

So, here it is taking an assignment and we have discussed the basic operation of counter DPWM counter-based DPWM in lecture numbers 69 and 70.

(Refer Slide Time: 16:04)

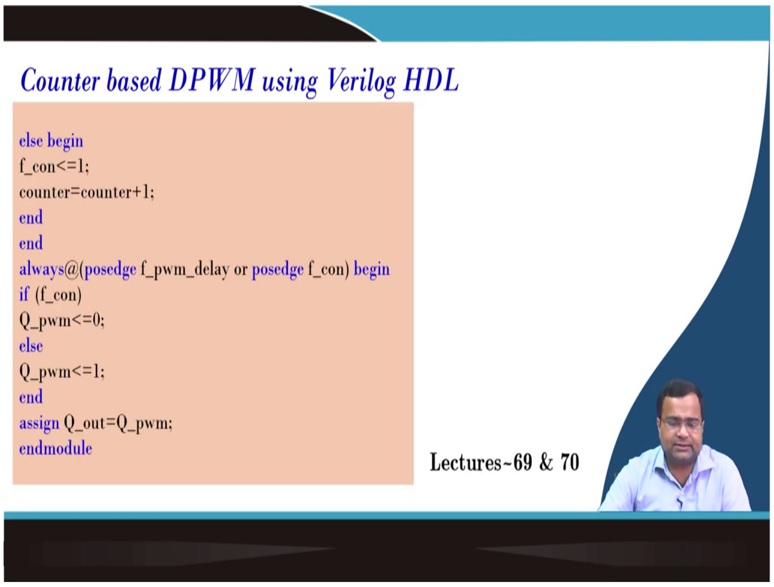


Counter based DPWM using Verilog HDL

```
always@(posedge f_clk) begin
Q1=f_pwm;
Q2=Q1;
f_pwm_delay=Q2;
end
always@(posedge f_clk or posedge f_pwm) begin
if (f_pwm) begin
f_con<=0;
counter=0;
end
else if (counter<=controller_output) begin
f_con<=0;
counter=counter+1;
end
```

Lectures-69 & 70

(Refer Slide Time: 16:06)

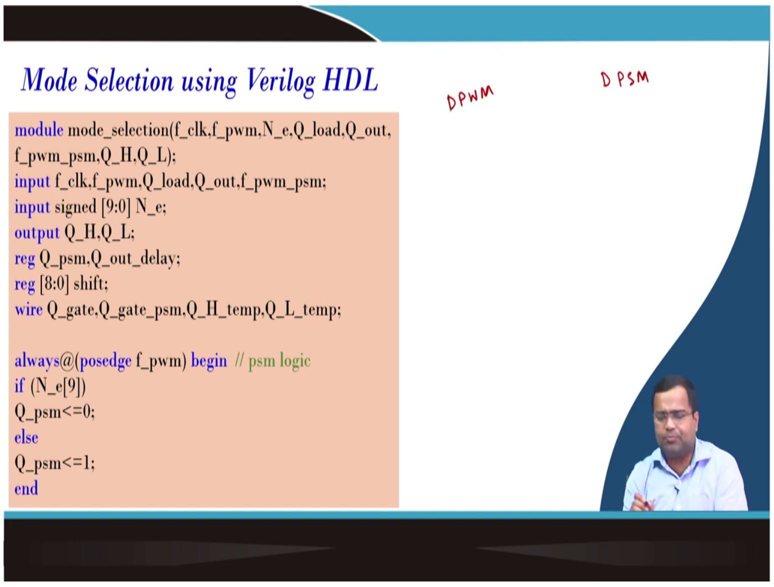


Counter based DPWM using Verilog HDL

```
else begin
f_con<=1;
counter=counter+1;
end
end
always@(posedge f_pwm_delay or posedge f_con) begin
if (f_con)
Q_pwm<=0;
else
Q_pwm<=1;
end
assign Q_out=Q_pwm;
endmodule
```

Lectures-69 & 70

(Refer Slide Time: 16:11)



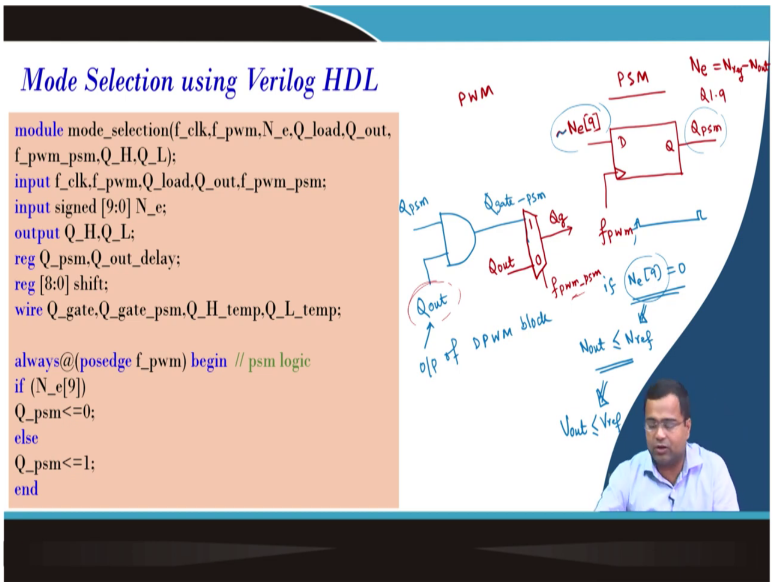
Mode Selection using Verilog HDL

```
module mode_selection(f_clk,f_pwm,N_e,Q_load,Q_out,
f_pwm_psm,Q_H,Q_L);
input f_clk,f_pwm,Q_load,Q_out,f_pwm_psm;
input signed [9:0] N_e;
output Q_H,Q_L;
reg Q_psm,Q_out_delay;
reg [8:0] shift;
wire Q_gate,Q_gate_psm,Q_H_temp,Q_L_temp;

always@(posedge f_pwm) begin  // psm logic
if (N_e[9])
Q_psm<=0;
else
Q_psm<=1;
end
```

So, I am not going to repeat this it is already discussed in lectures 69 and 70. Now the next part which is one of the very crucial parts is the mode selection. So, how are you going to select mode? First of all, there are two modes one is a DPWM mode another is a DPSM; which means, digital psm.

(Refer Slide Time: 16:38)



Mode Selection using Verilog HDL

```
module mode_selection(f_clk,f_pwm,N_e,Q_load,Q_out,
f_pwm_psm,Q_H,Q_L);
input f_clk,f_pwm,Q_load,Q_out,f_pwm_psm;
input signed [9:0] N_e;
output Q_H,Q_L;
reg Q_psm,Q_out_delay;
reg [8:0] shift;
wire Q_gate,Q_gate_psm,Q_H_temp,Q_L_temp;

always@(posedge f_pwm) begin  // psm logic
if (N_e[9])
Q_psm<=0;
else
Q_psm<=1;
end
```

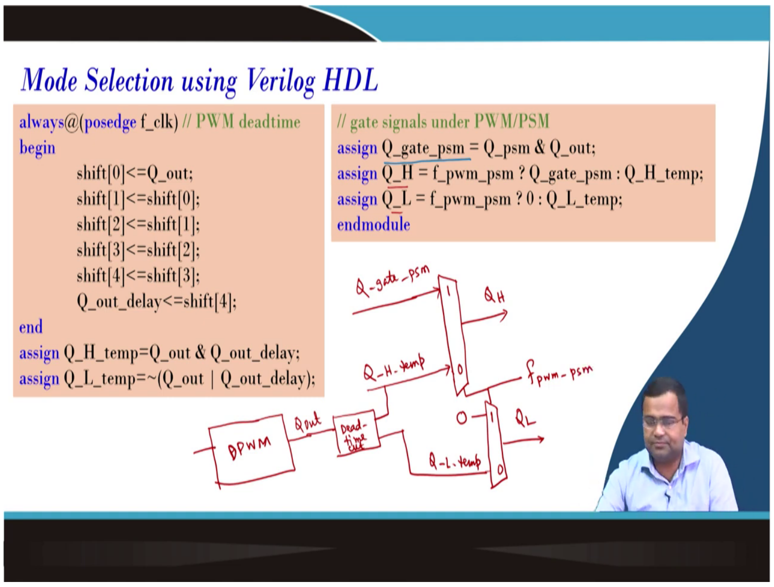
So, whether you call it digital PSM or PSM maybe we can use a generic name because it is a PWM we are implementing digitally, but this is a PSM. So, PWM we are defining under PSM what is there. So, PSM what it will do it will take I would say error; that means, we have the error voltage which is N ref minus N out. Now it is a Q1 dot 9 sign format and so the tenth bit is the MSB ok I would say it has a total of 10 bits the MSB indicates the sign bit.

So, if we take N e 9 if this is high then what does it mean? That means as if we are settling it like this, if this bit is 0 then at the edge of the PWM clock this counter which is your D flip flop is your Q psm; that means, what does it mean? At the edge of this clock that means if this is low; that means e 9 is equal to 0 what does it mean?

This implies or I would say 0 means your N output N out is less than equal to N ref and if it is 0 then its invert logic will be 1 and then Q psm will be 1 which means if the output voltage this implies that output voltage is less than V ref. So, at the edge of this clock if it says output voltage is less than equal to V ref then the cycle will be charged; that means, Q psm will be 1.

So, you need to charge the inductor; that means, the switch should be turned on and the duration of the on-time will be decided by the width of the fixed width which is set by the N con psm. But if this guy is 1 then this invert will be 0 then Q psm will be 0 then the actual gate signal which means, you can say, so there will be another logic.

(Refer Slide Time: 19:20)



Mode Selection using Verilog HDL

```
always@(posedge f_clk) // PWM deadtime
begin
        shift[0]<=Q_out;
        shift[1]<=shift[0];
        shift[2]<=shift[1];
        shift[3]<=shift[2];
        shift[4]<=shift[3];
        Q_out_delay<=shift[4];
end
assign Q_H_temp=Q_out & Q_out_delay;
assign Q_L_temp=~(Q_out | Q_out_delay);
```

```
// gate signals under PWM/PSM
assign Q_gate_psm = Q_psm & Q_out;
assign Q_H = f_pwm_psm ? Q_gate_psm : Q_H_temp;
assign Q_L = f_pwm_psm ? 0 : Q_L_temp;
endmodule
```

I mean if you go to the next one that is you know here the Q gate psm Q gate psm that is your Q psm multiplied by the Q; that means, your Q what? The output of the DPWM is so we are setting Q out which is the output of the psm, output this is the output of dpwm block ok and this we are setting as Q gate psm. That means, this will be the gate signal main gate signal during psm, but we have the choice of whether to let you know the psm or we need to select the PWM; that means, now in addition to that.

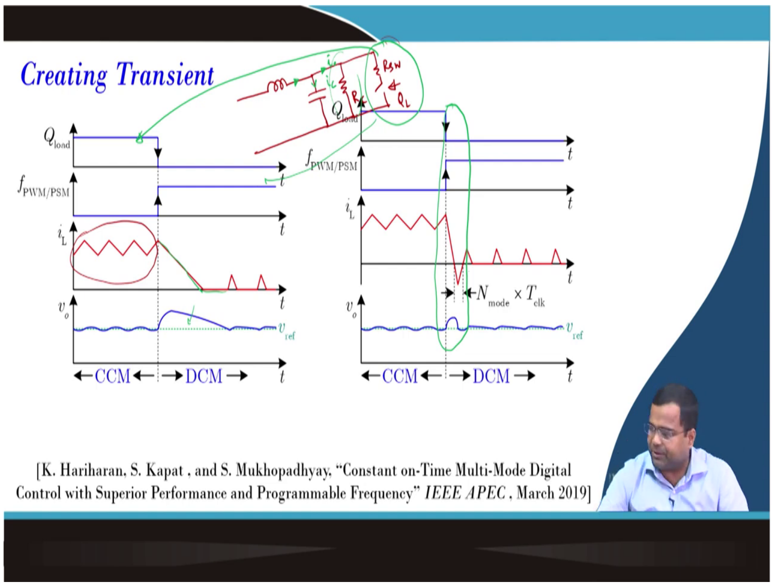
so, this is 1 this is 2 and it will simply take Q out and which will be your actual Q gate I will say and this is your f pwm psm the mode logic. That means, if it is set to 0 it will take Q out which is the output of the DPWM. In that case, the input to the DPWM will be the output of the controller N con PWM, but if this is high then it will set the psm logic and in that case, the DPWM will output will be ended with Q psm.

Q psm will decide whether the cycle will be charged or skipped. If it is a charge it is 1 for the whole cycle if it is 2 means it is skipped and during that time the Q output which is the PWM output will be decided by the input of the DPWM which is the N con psm there is a fixed quantity ok. So, this is a mode selection logic and accordingly, the Q high that we have discussed and Q low.

So, now we have one signal is that Q gate psm that we have discussed and that is muxing with that is 1 0 then Q I would say here if you see gate Q H temp because you remember for DPWM this is the DPWM, sorry DPWM an output of the DPWM will also have a dead time circuit. Dead time circuit will have this Q high and there will be another Q low Q L temp because under PWM will operate in a synchronous configuration and this will be going to Q high side gate signal based on the select line.

Now, there will be another option again 1 0 this will be simply 0. This will go to Q low, this will go to Q low this will go to the gate drive circuit and for this all this flag here it is f PWM psm that is it. This is the logic that we have described.

(Refer Slide Time: 23:16)



Creating Transient

[K. Hariharan, S. Kapat , and S. Mukhopadhyay, "Constant on-Time Multi-Mode Digital Control with Superior Performance and Programmable Frequency" *IEEE APEC*, March 2019]

So, now we are creating a transient. So, suppose when it was in high load we know Q load is high then we are operating in PWM. Suppose it goes to PSM then we are simply setting it to PSM; that means when it goes to light load. But you see whenever it goes to light load then if you consider the inductor then you know this capacitor suppose your resistance was there which is a continuous resistance which is a large value and there is a switch resistance and this is your Q L switch.

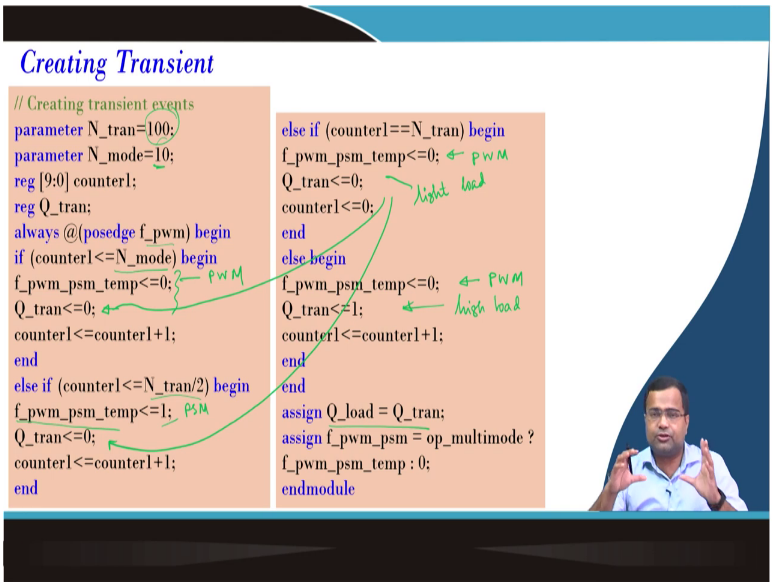
Now when this Q L initially was connected, so when it was connected that means, you can see this condition link with this. So, it was connected, but this time it is disconnected. Now, since the total load current will decrease because this low resistance path is removed, the inductor current was high.

So, it has to mean, you have more current coming in from the inductor than the current going out of the capacitor. So, that means the effective capacitor current will become positive large positive because suddenly you have removed the load and this will cause an overshoot but inductor. So, this overshoot can be I mean to get back the regulation point you have to extract the excess charge from the capacitor.

But, if you purely operate in DCM the inductor will hit 0 and it will remain 0 because there is no other path than this high resistance path it will take a longer duration based on the amount of light load current and this will cause a large recovery time or settling time when you are going for step down transient. So, to overcome that actually, you can operate PWM for a small time that time the current will go negative because it is a synchronous converter and it will extract the charge out of the capacitor.

In that way, you can quickly come back to a steady state and then you can again move back to the DCM operation with pulse skipping. So, you can save the light load efficiency and this is a standard practice we have also discussed this aspect you know for constant on-time context in this paper.

(Refer Slide Time: 25:38)



Creating Transient

```
// Creating transient events
parameter N_tran=100;
parameter N_mode=10;
reg [9:0] counter1;
reg Q_tran;
always @(posedge f_pwm) begin
if (counter1<=N_mode) begin
f_pwm_psm_temp<=0;    PWM
Q_tran<=0;
counter1<=counter1+1;
end
else if (counter1<=N_tran/2) begin
f_pwm_psm_temp<=1;    PSM
Q_tran<=0;
counter1<=counter1+1;
end
else if (counter1==N_tran) begin
f_pwm_psm_temp<=0;    PWM
Q_tran<=0;            light load
counter1<=0;
end
else begin
f_pwm_psm_temp<=0;    PWM
Q_tran<=1;            high load
counter1<=counter1+1;
end
end
assign Q_load = Q_tran;
assign f_pwm_psm = op_multimode ?
f_pwm_psm_temp : 0;
endmodule
```

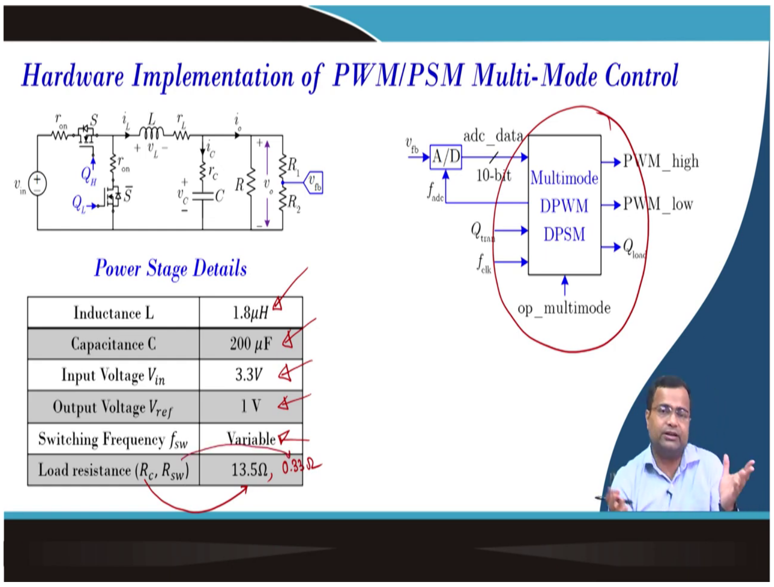
Now, this is a transient to a k t. So, you can see here there is something called N mode in the T clock. So, this is the number of the clock that we are turning on PWM operation even after the load step-down transient to extract the current and that will decide how much and how many.

So, this is number 10 10 number of the cycle we are operating in PWM because we are not doing any large signal-based control which could take 2 to 3 cycles to come back or 1 cycle, but here we are making you know small signal operation that is why we are allowing for a certain time and then we enable. So, this is what I told you if the load current goes high; that means, we are deciding the number of. So, a total of 100 cycles are given 50 cycle load high and 50 cycle load low this is based on this and we are setting the load to be transient.

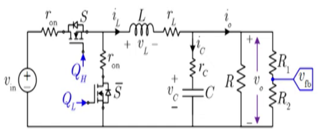
But, with this signal, we are allowing for some more time to turn on the PWM operation. So, this mode is supposed to be because your load is 0; that means, it is a light load, but we are continuing for ten more cycles in PWM and then we come back to this mode to psm. So, this is like a psm this is PWM and then of course, this is PWM because it is under height load. I will when they are equal this is PWM and this is your high load; that means, yeah this is your pwm and this is your high load.

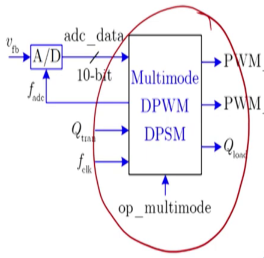
But, this is your light load. This is also light load and this is also light load. So, for a light load in some cases, we are using PWM ok.

(Refer Slide Time: 27:30)



Hardware Implementation of PWM/PSM Multi-Mode Control





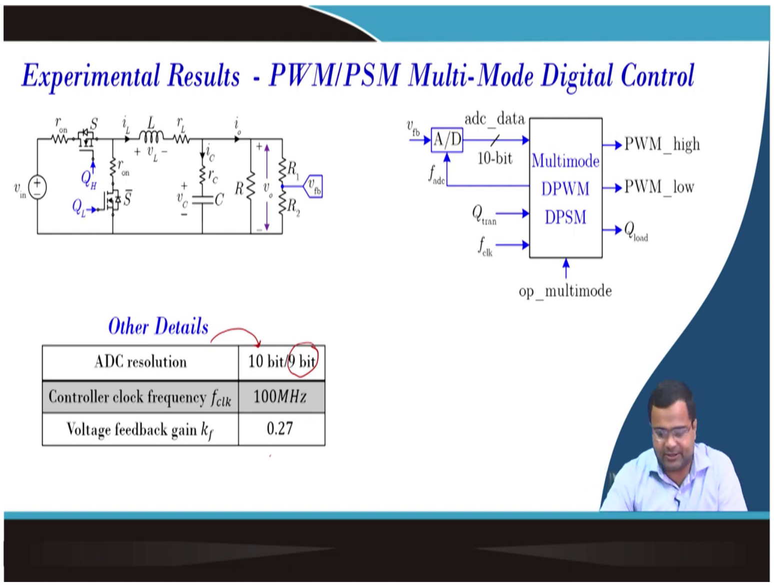
Power Stage Details

Inductance L	$1.8\mu H$
Capacitance C	$200\ \mu F$
Input Voltage V_{in}	$3.3V$
Output Voltage V_{ref}	1 V
Switching Frequency f_{sw}	Variable
Load resistance (R_c, R_{sw})	13.5Ω, $0.33\ \Omega$

Now, we want to show the hardware implementation. So, this is a buck converter that we have demonstrated multiple times in this course now we have written the Verilog code and we already know how to synthesize the Verilog code and plug into or dump it to FPGA and we have followed the same step. We have considered our hardware prototype of a 1.8 microhenry inductor 200 microwatt capacitor.

We are taking the test condition at 3.3 input voltage output voltage is 1-volt switching frequency under PWM is fixed 200 kilohertz, but under pulse keeping it will vary because depending on the number of skip cycles. The load resistance the fixed value is this and there is one more which is 0.33 ohm that is under high load. So, when this switch resistance will be connected it will be a high load when it is disconnected it is under light load.

(Refer Slide Time: 28:20)



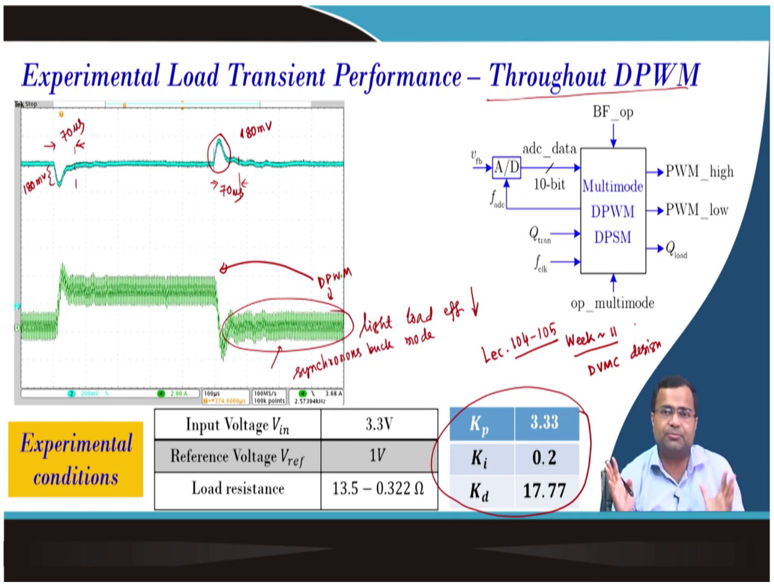
Experimental Results - PWM/PSM Multi-Mode Digital Control

Other Details

ADC resolution	10 bit/9 bit
Controller clock frequency f_{clk}	$100MHz$
Voltage feedback gain k_f	0.27

So, the additional we are using an ADC resolution of 9 bit even though it is a 10 bit ADC we are discarding 1 MSB to avoid any limit cycle oscillation, the feedback gain is 0.2, and the clock frequency is 100 megahertz.

(Refer Slide Time: 28:32)



Experimental Load Transient Performance – Throughout DPWM

Experimental conditions

Input Voltage V_{in}	3.3V
Reference Voltage V_{ref}	1V
Load resistance	13.5 – 0.322 Ω

K_p	3.33
K_i	0.2
K_d	17.77

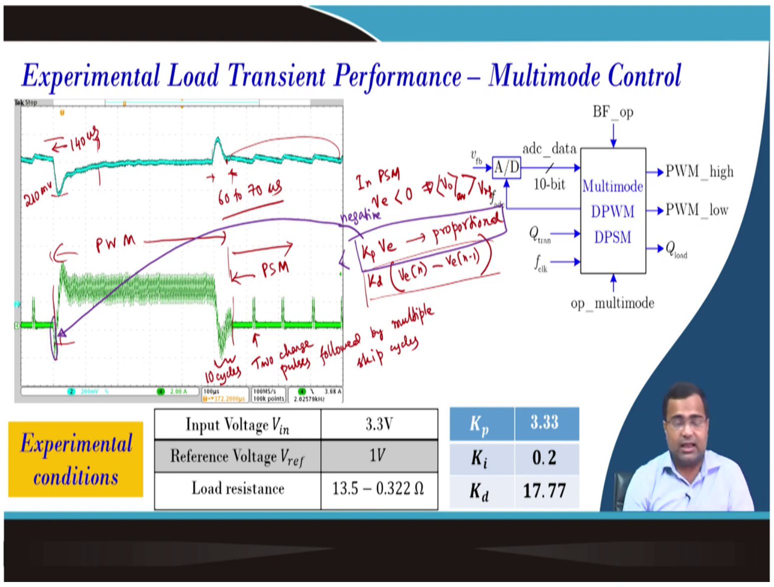
Now, we are showing experimental results. So, test condition we are using the same K p K i K d which we discussed in the week I think 11 in the design. Do you know where we have designed the digital voltage mode control design I think it could be lecture number I think it is 1024105 or something like that.

So, here we have discussed this, or maybe 106 10 2345 yeah. So, the same value for 3.3. So, this is the step-up transient and you can see the undershoot here is roughly around 8180 millivolt and the overshoot is also around 180 millivolt because it is a 200 millivolt scale, and this transient time you can see where this the scale is 100 microsecond.

So, it is around 70 microsecond that we found, and here also if you take this transient time, we are getting around 70 microsecond. Now, this is when you are operating throughout DPWM and you can see this is under a very light load, but under light load, we are still operating in synchronous configuration right synchronous buck mode as a result your losses would be very high because the switching frequency is very high under PWM.

So, this is your DPWM for both of these cases. So, here the light load efficiency will be penalized. It will degrade because the switching frequency is high driver loss will be high and you are unnecessarily burning power.

(Refer Slide Time: 30:25)



Experimental Load Transient Performance – Multimode Control

Experimental conditions

Input Voltage V_{in}	3.3V
Reference Voltage V_{ref}	1V
Load resistance	13.5 – 0.322 Ω

K_p	3.33
K_i	0.2
K_d	17.77

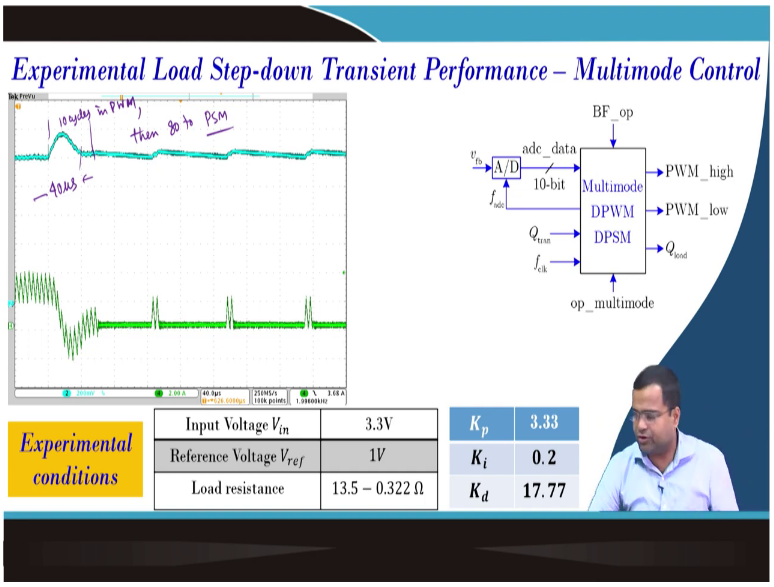
Now, we are enabling multi-mode. So, you see here from here you are PSM start and you can see there is a large ripple and here there are two charge pulses followed by multiple skip cycles ok PSM mode. So, this operating region is PSM mode and this mode to this mode is your PWM mode ok.

So, high power is fine, but now you see we have told this we have taken 10 cycles. So, we have operated under PWM just to you know you know to improve this recovery time because this time is also around 60 to 70 microseconds; that means, the step-down transient remains the same as earlier. But this has a problem. Now this undershoot is coming like a 200 milli volt why and the recovery time is also large. It is you know it is coming to be if you take from here to here it is a 100-microsecond scale.

So, it is like 140 microseconds which are almost double, but why because I have discussed in PSM PSM your error voltage is generally negative because your output voltage average value is generally greater than the average value V ref. So, as a result integral, we are disabled the integral. Even though we disable the integral the K p into V e which is the proportional gain at the time of this mode transition it is negative. Derivative gain is also negative because K d into V e n V e n minus 1 this is also negative.

This condition when you turn it on is 0 because the error is settled. So, this will primarily drive during the mode transition, this will primarily drive and this quantity is negative. I would say this is during this time this quantity is negative as a result it is going down and the switch is turned off for some time in the beginning and that taking down the current.

(Refer Slide Time: 33:16)



Experimental Load Step-down Transient Performance – Multimode Control

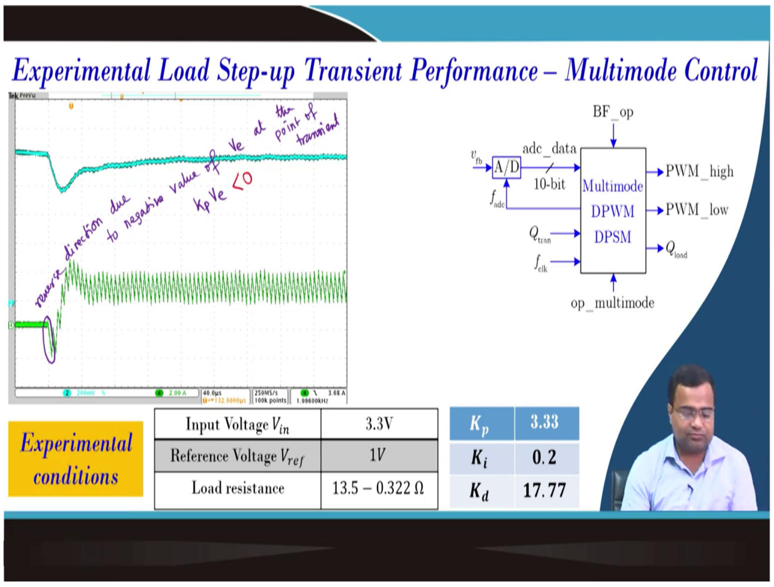
Experimental conditions

Input Voltage V_{in}	3.3V
Reference Voltage V_{ref}	1V
Load resistance	13.5 – 0.322 Ω

K_p	3.33
K_i	0.2
K_d	17.77

So, to overcome that multi-mode you zoom the step-down transient. So, we have retained we got just like a 40 micro semicrosecond is far I mean very good almost we can retain the PWM operation and we have continued from here to here we have continued 10 cycles nearly 10 cycles in PWM then go to PSM and this logics actually product to product varies there are different typology, but I am just showing conceptual understanding.

(Refer Slide Time: 33:52)



Experimental Load Step-up Transient Performance – Multimode Control

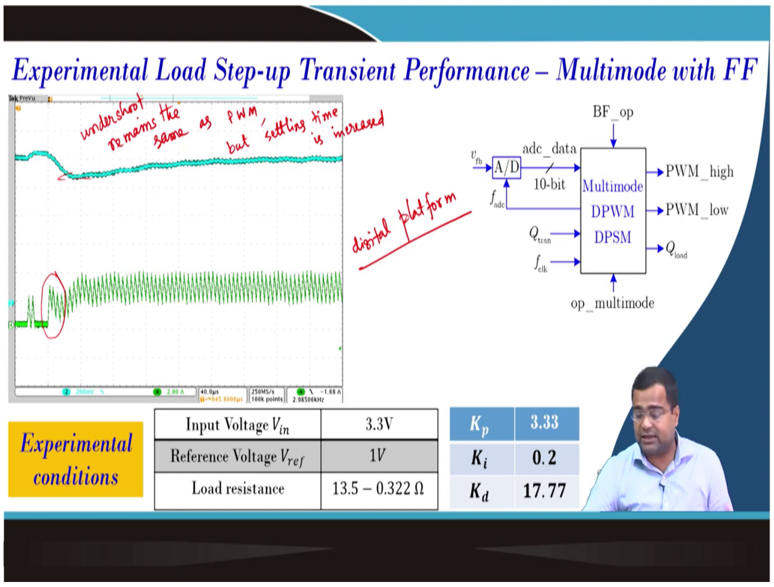
Experimental conditions

Input Voltage V_{in}	3.3V
Reference Voltage V_{ref}	$1V$
Load resistance	13.5 – 0.322 Ω

K_p	3.33
K_i	0.2
K_d	17.77

Now, the multi-mode, if you step up this, is the problem I have explained because initially instead of going up it is going down. So, reverse direction due to negative effect from minus sorry K p. I will not say negative effect due to the negative value of error voltage at the point of transient ok and as a result, this term becomes. So, this term becomes negative.

(Refer Slide Time: 34:53)



Experimental Load Step-up Transient Performance – Multimode with FF

Experimental conditions

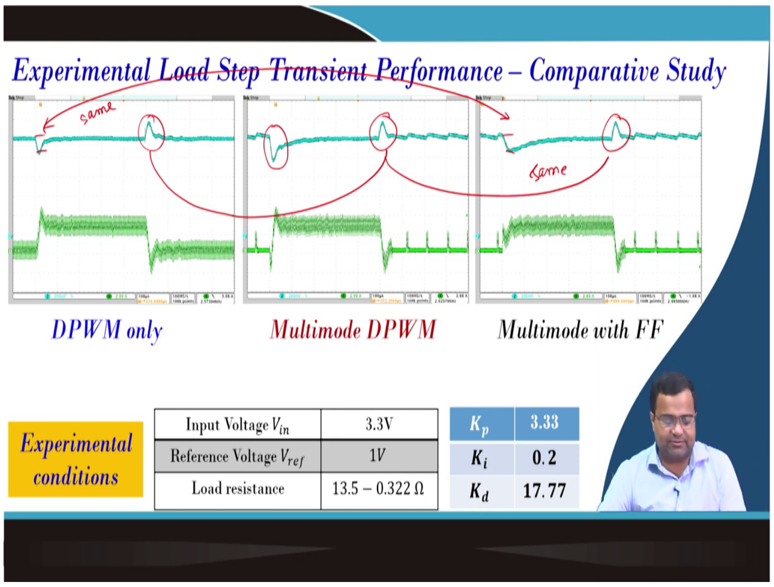
Input Voltage V_{in}	3.3V
Reference Voltage V_{ref}	$1V$
Load resistance	13.5 – 0.322 Ω

K_p	3.33
K_i	0.2
K_d	17.77

So, to overcome that we added a feed-forward term; that means, we have added some offset term for fuse I mean the at this time of transient and this is continued since there is an integral action it will take over. So, undershoot now this undershoot sorry undershoot remains the same as PWM. But settling time has increased is increased because the integral will take slow action to reach, but we can make the output voltage undershoot can be reduced.

So, that is one of the most important parts. That means, this feed-forward term is an offset term and that is the digital platform it will allow. So, any transient you can detect and if you can add an offset term to quickly ramp up that is possible.

(Refer Slide Time: 36:07)



Experimental Load Step Transient Performance – Comparative Study

DPWM only *Multimode DPWM* *Multimode with FF*

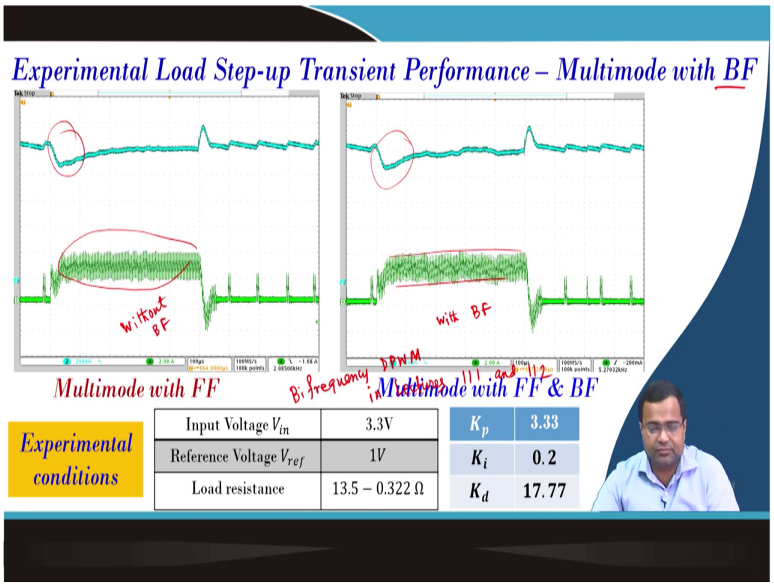
Experimental conditions

Input Voltage V_{in}	3.3V
Reference Voltage V_{ref}	1V
Load resistance	13.5 – 0.322 Ω

K_p	3.33
K_i	0.2
K_d	17.77

Now, we are showing the comparative study this is purely DPWM, this is with a multi-mode where we suffered this operation and you see this undershoot and this undershoot are the same. This undershoots these also are same. So, you can retain it and we can now achieve very high efficiency also.

(Refer Slide Time: 36:35)



Experimental Load Step-up Transient Performance – Multimode with BF

Experimental conditions

Input Voltage V_{in}	3.3V
Reference Voltage V_{ref}	1V
Load resistance	13.5 – 0.322 Ω

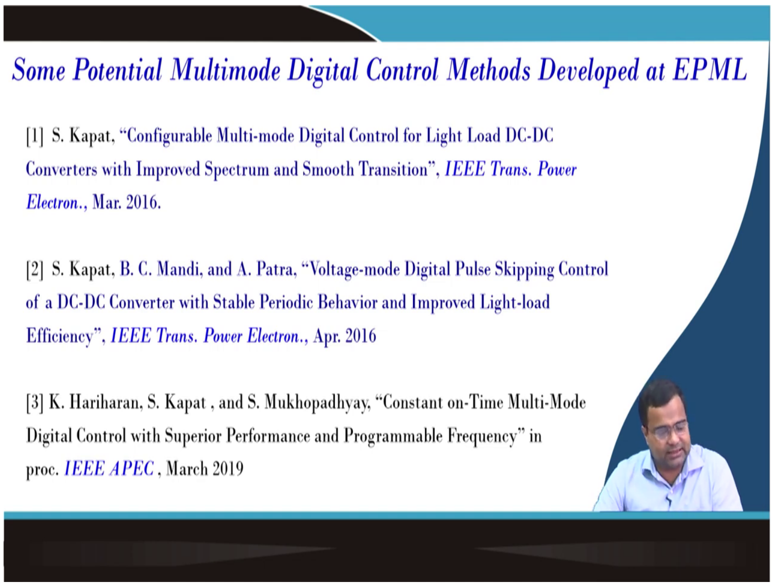
K_p	3.33
K_i	0.2
K_d	17.77

And experimental transient with now because it is a multi-mode we are now enabling the bi-frequency operation. So, bi frequency operation that bi frequency operation bi frequency DPWM in we have discussed in lectures 111 and 112. So, if you go there is an almost insignificant impact and the transient is without bi frequency and this is with bi frequency.

So, you can reduce the spectral peak and you see the transient response there is no change in the output voltage transient. So, in that way, we can enable it because we have discussed the need for multi-mode in lecture number 90 you know lecture number 100 where we have identified that in high power it is essential to have spectral spreading for mi I reduction that can be achieved.

We want to achieve the fast transient that also we have achieved we want to reduce the undershoot overshoot that is also achieved. So, now this multi-mode control you can play with multiple features and make it super fast with a highly efficient converter.

(Refer Slide Time: 37:57)



Some Potential Multimode Digital Control Methods Developed at EPML

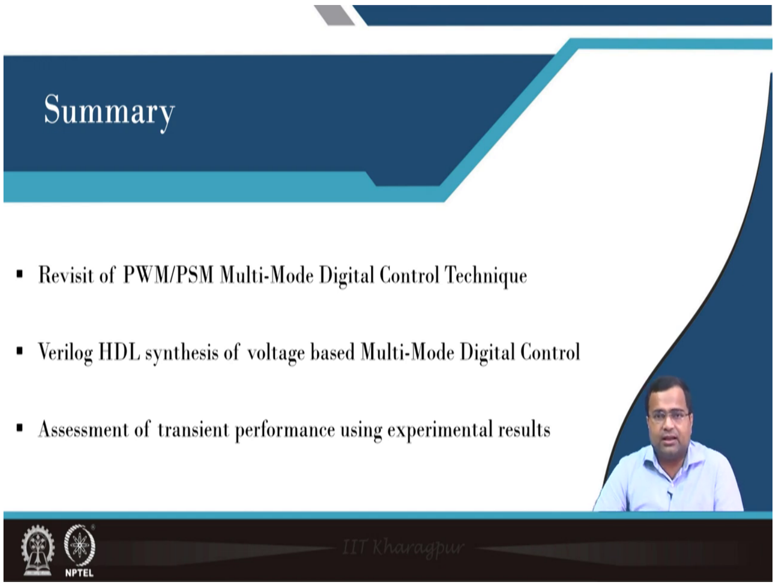
[1] S. Kapat, "Configurable Multi-mode Digital Control for Light Load DC-DC Converters with Improved Spectrum and Smooth Transition", *IEEE Trans. Power Electron.*, Mar. 2016.

[2] S. Kapat, B. C. Mandi, and A. Patra, "Voltage-mode Digital Pulse Skipping Control of a DC-DC Converter with Stable Periodic Behavior and Improved Light-load Efficiency", *IEEE Trans. Power Electron.*, Apr. 2016

[3] K. Hariharan, S. Kapat , and S. Mukhopadhyay, "Constant on-Time Multi-Mode Digital Control with Superior Performance and Programmable Frequency" in proc. *IEEE APEC* , March 2019

And in this you know we have some work which is in the context of multi-mode that can be useful you know for pulse skipping control basically and you know this can be also multi-mode constant on-time pulse skipping and a combination of PWM. So, all these things are discussed here.

(Refer Slide Time: 38:17)



Summary

- Revisit of PWM/PSM Multi-Mode Digital Control Technique
- Verilog HDL synthesis of voltage based Multi-Mode Digital Control
- Assessment of transient performance using experimental results

IIT Kharagpur
NPTEL

So, in summary, we have discussed we have revisited our PWM PSM multimode digital control, Verilog HDL synthesis of voltage-based multi-mode digital control. We have also made some assessments of transient performance using experimental results. Now we will go in the next lecture another multi-mode control which is a peak current based you know constant on-time adaptive multi-mode control which is interfaced with the PWM fixed frequency peak current mode control that is it for today.

Thank you very much.