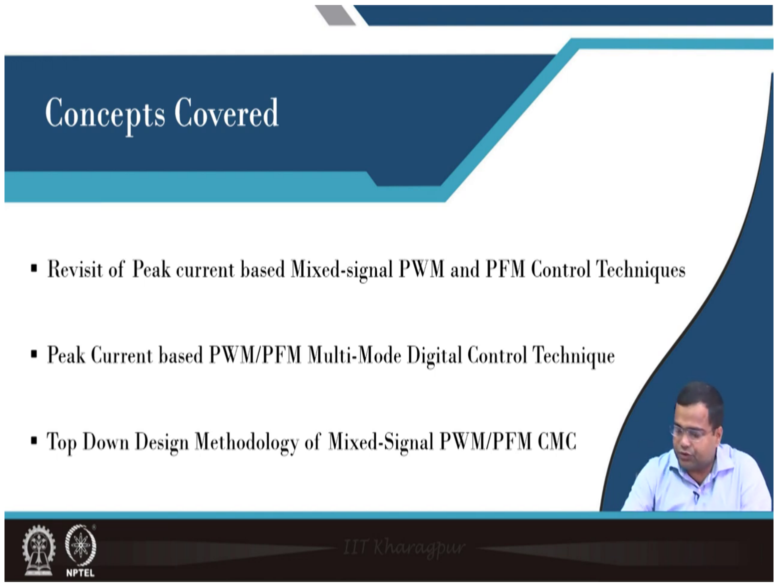
Digital Control in Switched Mode Power Converters and FPGA-based Prototyping

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

Module - 11
Hardware Case Studies of Advanced Digital Control Techniques and Course Summary
Lecture - 115
FPGA Prototyping of Peak Current-based PWM/ PFM Multi-Mode Digital Control - I

Welcome to this lecture we are going to consider another Multi-Mode Control Architecture which is a Peak Current based Mix Signal PWM PFM Control.

(Refer Slide Time: 00:35)



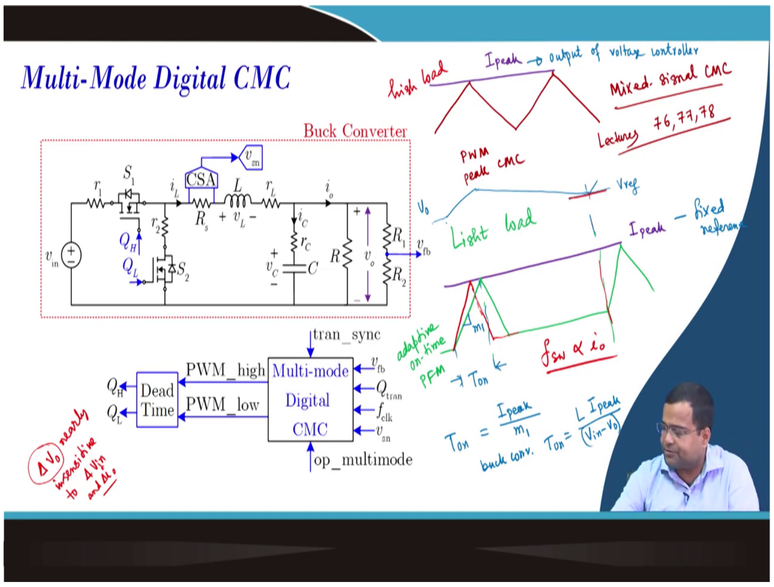
Concepts Covered

- Revisit of Peak current based Mixed-signal PWM and PFM Control Techniques
- Peak Current based PWM/PFM Multi-Mode Digital Control Technique
- Top Down Design Methodology of Mixed-Signal PWM/PFM CMC

IIT Kharagpur

In the first part of this lecture, we will revisit the peak current-based multi-mixed signal current mode control PFM peak current base multi-mode control. Then we will do top-down design methodology and Verilog implementation and we will continue this lecture in the second part also.

(Refer Slide Time: 00:52)



So, here is multi-mode digital current mode control and what is our objective. So, if you recall there are two things. If we talk about our mixed-signal current mode control; that means, if we consider our peak current reference. So, this is our peak current reference now we know in mixed signal current mode control this current will look like this.

So, this is under PWM and this is the peak current mode control. And we have discussed how to implement we have discussed that mixed signal current mode control which is a peak current mode control.

We have discussed in lecture numbers I think 70 5, 6, 7 no 76, 77, 76, 77, and 78 these lectures we have discussed ok. So, we have discussed in lecture number we have discussed. Now another thing we have discussed so we want to consider is a high load, is under high load.

Now we want to consider under light load we want to maintain this same peak current base approach. Then what we can do? We can again take a peak current and we want to now make this peak current base under DCM which we call an adaptive on-time PFM where the on-time is getting generated.

So, this is on time, what is the on time if this is the slope? So, T on was equal to I P versus M 1, and for a buck converter for buck converter T on equal to L I peak you know what is this V in by V in minus V 0; V 0. Here the beautiful thing is that if the input voltage and output voltage are the same and the peak current is fixed. This is a fixed reference and this is the output of the controller output of voltage controller ok.

So, now here if everything is fixed and how does this trigger this one? So; that means, during this time if there is an output voltage it will go up it will go down like this. So, this is the output voltage; that means, whenever the output voltage crosses V ref; that means, if you use a different color V ref then it will again turn on on time and this on time is not a timer based on time.

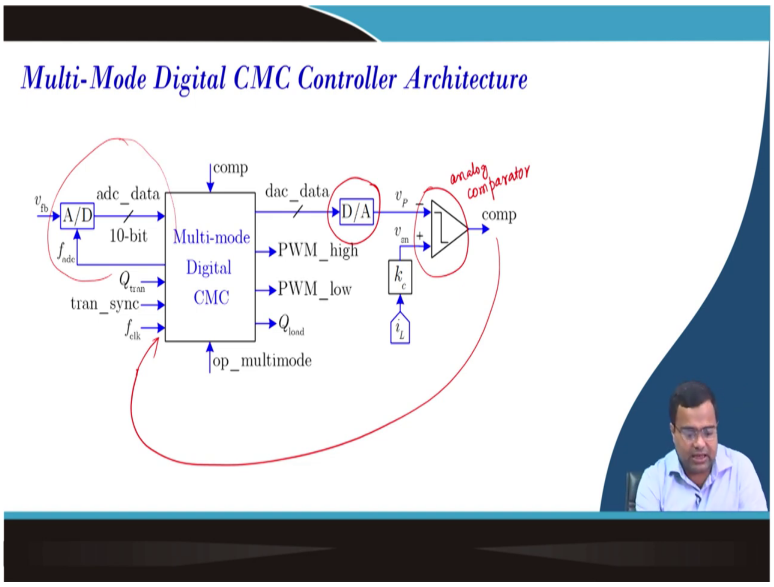
So, it is coming from the inner current loop where the switch will be turned on and the inductor current will rise and it will compare with the peak current in the analog domain. Because this will be using DAC we will discuss and then that will generate the on-time. So, the good thing about this here is if the input voltage increases. Then what will happen?

This will go fast then it will come down here; that means, T on will decrease if the input voltage increases and so; that means, your inject charge injection will remain the same. But the switch on time will be different and we know if the on-time varies then what happens if the T on increases?

So, it can be shown that output voltage ripple in this case will be insensitive nearly insensitive to V delta V in variation and load current variation delta load. That means, under light load if the load current changes or input voltage changes the output voltage ripple will be more or less the same. Because typically in constant on time we know that the output voltage ripple is almost independent of load current, but it is a strong function of input voltage.

But if we can use an adaptive on-time control by this peak current base approach. Then we can also make for a given input-output voltage it will retain that switching frequency is linearly varying with I 0 all these features will be retained. In addition to that output voltage ripple will be also insensitive to input voltage variation and that makes this connection very popular for many commercial products.

(Refer Slide Time: 06:30)

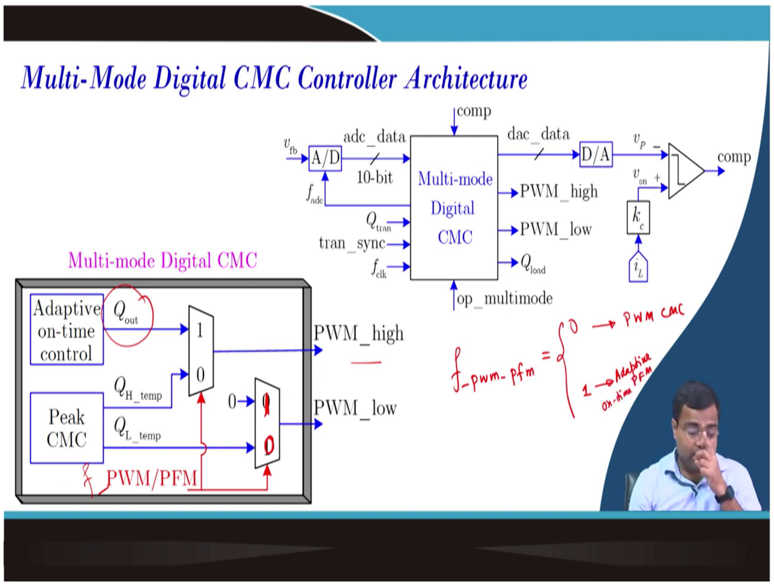


Multi-Mode Digital CMC Controller Architecture

Now, we want to discuss; that means, we know for any current base architecture for digital we need a DAC and DAC will generate the current reference, and that current reference is the current analog comparator. This is our analog comparator, analog comparator and this comparator output is coming here.

So, this comparator output is going here inside this block and there is a transient synchronization that I will discuss and it has an ADC. So, we can have a pure analog kind of comparator voltage comparator. But since we have this digital platform that is why you are using virtually you want to replicate like an analog compactor and this block we have to synthesize.

(Refer Slide Time: 07:20)

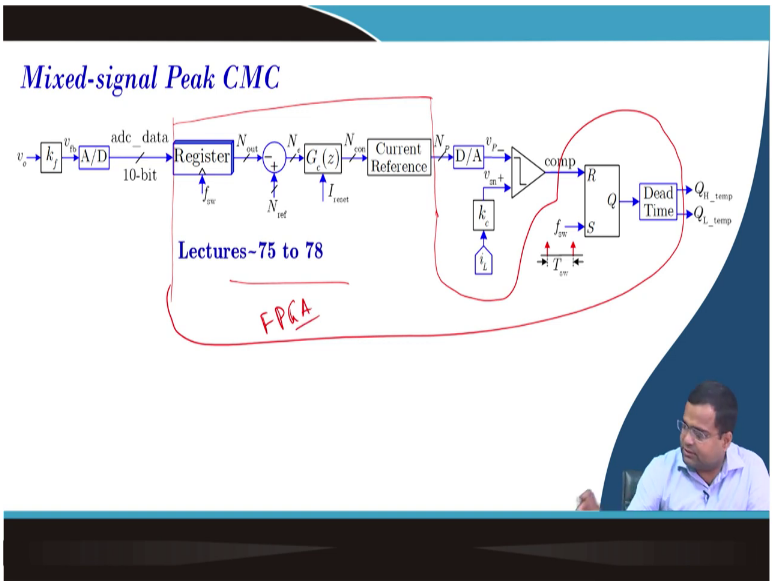


Now, this is the overall block and if we recall the earlier lecture I mean lecture number 114. We wanted to integrate that PWM and there we did PWM PSM here PWM PFM. So, the methodology of selecting the high-side and the low-side gate signal is the same; which means, it can take directly the adaptive on-time output.

If the select line is high PWM PFM and in that case, the high side gate signal will directly get connected to the output of the adaptive one time and the low side gate signal is 0 this should be 1, this should be 0, there is a typo here. But when this is 0; that means, this is; that means, what is my f PWM pfm?

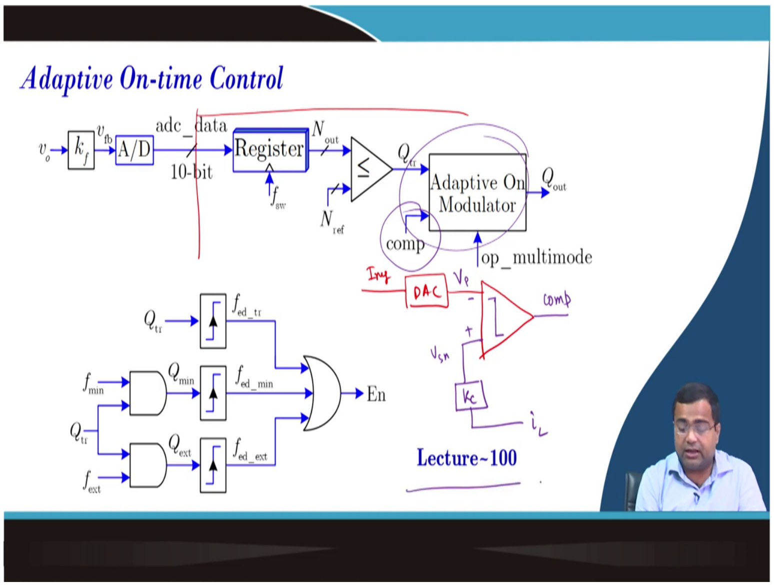
This is equal to 0 then it is for PWM operation, PWM current mode control. If it is 1, then it is not I would say it implies that is adaptive on time PFM ok, and during that time for this mode low side gate signal will be set to 0.

(Refer Slide Time: 08:47)



Now, we have discussed the mixed signal peak current mode control in lectures 75 to 78 in sufficient detail that how to use a register then this part will be your inside the FPGA. Then this part will also come and this whole part will be inside the FPGA that we have discussed.

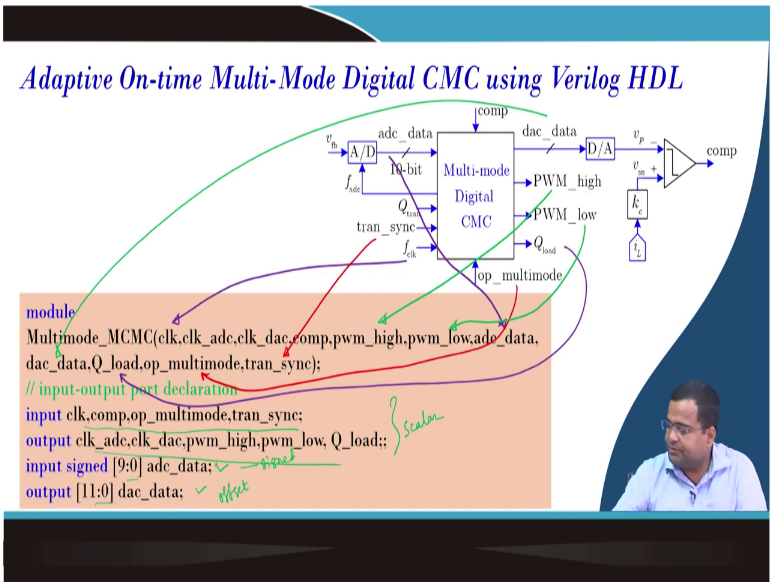
(Refer Slide Time: 09:10)



We have also discussed adaptive on time where the same thing this data to this adaptive on time modulator and we after that adaptive on time this is the comparator output; that means, here also which is not shown here there will be DAC. So, it is like an I ref current reference it is going to consider compared with the I mean if you go to the previous the comparator has mixed signal current mode control minus plus.

So, this is V p, this is V p minus and this is V s n sense voltage and there is a current sensor the current loop gain i L and this is your comparator and this we are calling comp and this comp is going inside. And this edge detection circuit how to generate this edge detection circuit for enabling this adaptive on time we have discussed in lecture number 100 in detail.

(Refer Slide Time: 10:27)



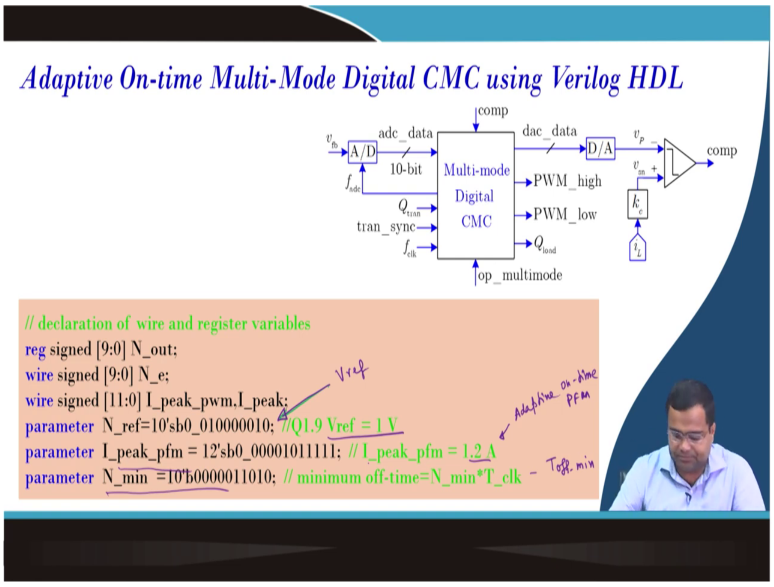
Adaptive On-time Multi-Mode Digital CMC using Verilog HDL

```
module
Multimode_MCMC(clk,clk_adc,clk_dac,comp,pwm_high,pwm_low,adc_data,
dac_data,Q_load,op_multimode,tran_sync);
// input-output port declaration
input clk,comp,op_multimode,tran_sync;
output clk_adc,clk_dac,pwm_high,pwm_low, Q_load;;
input signed [9:0] adc_data;
output [11:0] dac_data;
```

But now we want to develop this multi-mode control. So, this is the main module again I want to link; that means, this is your ADC data; that means, let us have ADC data. So, this ADC data is coming here ok. Then f the clock is here ok Q then we are generating inside, so this is you are here. Then we are also giving another option for this multimode trans sync we are giving here. We will see what is this is an external switch you know then the high side gate signal which is here.

So, this high side gate signal is here then the low side gate signal is here ADC data we have discussed DAC data here and this transient. So, we have discussed this is the interface and we can define which is the input and then what are the output these are all scalar. This is the vector input vector, output this is signed this is offset binary this is twelve bit this is ten bit ok.

(Refer Slide Time: 12:10)

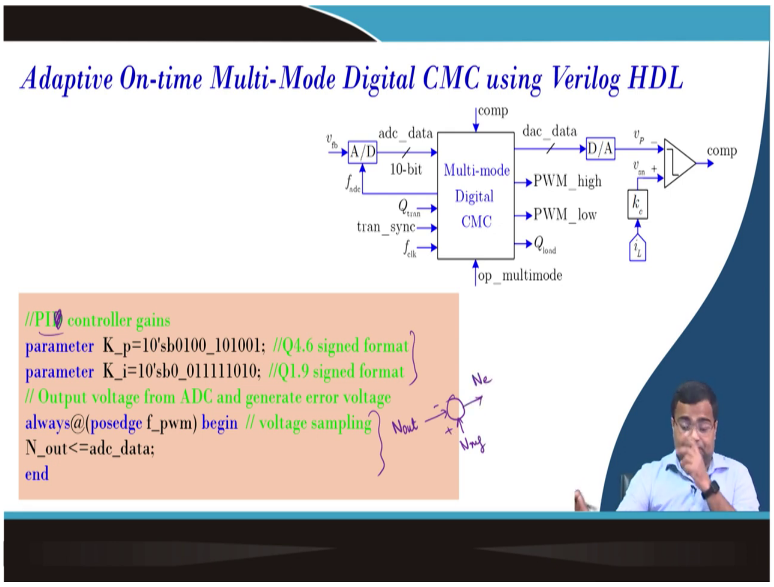


Adaptive On-time Multi-Mode Digital CMC using Verilog HDL

```
// declaration of wire and register variables
reg signed [9:0] N_out;
wire signed [9:0] N_e;
wire signed [11:0] I_peak_pwm,I_peak;
parameter  N_ref=10'sb0_010000010; //Q1.9 Vref = 1 V
parameter  I_peak_pfm = 12'sb0_00001011111; // I_peak_pfm = 1.2 A
parameter  N_min  =10'b0000011010; // minimum off-time=N_min*T_clk
```

Now, we are defining this I ref this is linked with the V ref reference voltage. So, this is linked with the V ref equal to 1 parameter peak current limit. We are setting the pfm 1 for adaptive on-time pfm and this is the minimum this corresponds to the cut-off minimum because any on-time modulator should have a minimum of time.

(Refer Slide Time: 12:50)

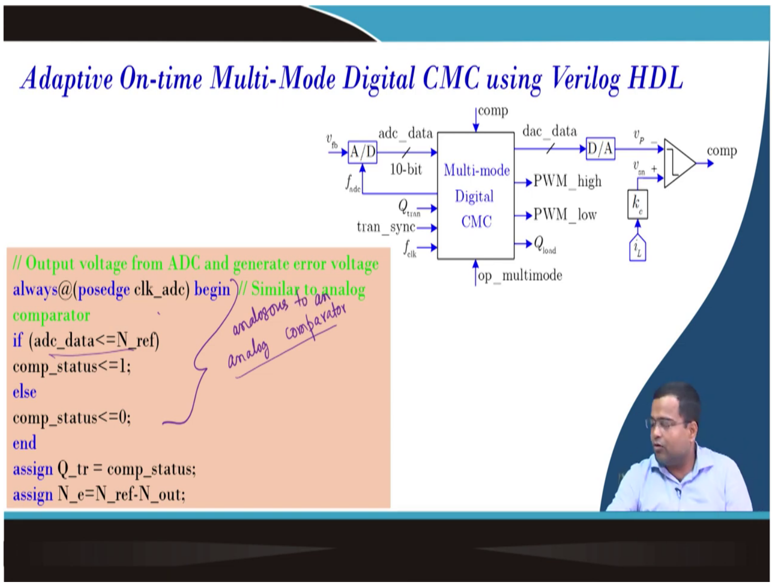


Adaptive On-time Multi-Mode Digital CMC using Verilog HDL

```
//PID controller gains
parameter  K_p=10'sb0100_101001;  //Q4.6 signed format
parameter  K_i=10'sb0_011111010;  //Q1.9 signed format
// Output voltage from ADC and generate error voltage
always@(posedge f_pwm) begin  // voltage sampling
N_out<=adc_data;
end
```

Then we need only the PI controller, not the PID controller PI controller gain. Because current mode control uses a PI controller this is for voltage sampling. Because we are sampling this voltage for the fixed frequency control where this will generate I N out and this n out will be subtracted from your N ref to generate an N error this error will go to the PI controller.

(Refer Slide Time: 13:23)



Adaptive On-time Multi-Mode Digital CMC using Verilog HDL

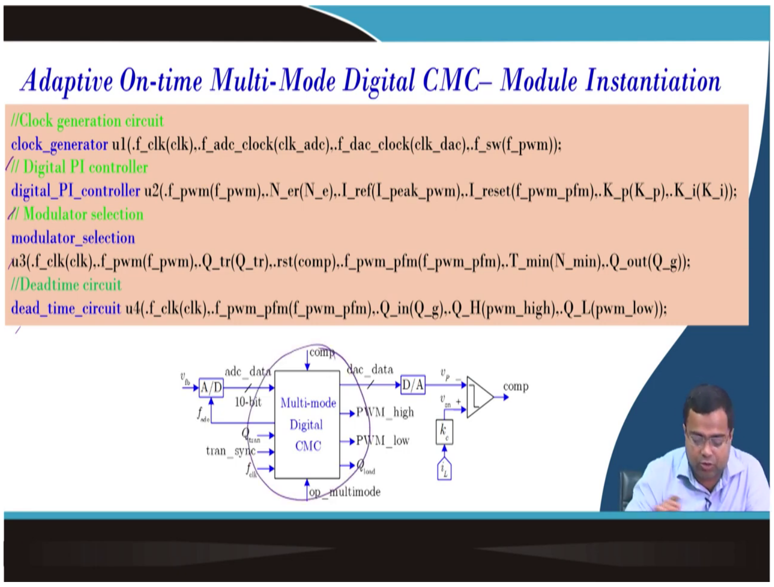
```
// Output voltage from ADC and generate error voltage
always@(posedge clk_adc) begin // Similar to analog
comparator
if (adc_data<=N_ref)
comp_status<=1;
else
comp_status<=0;
end
assign Q_tr = comp_status;
assign N_e=N_ref-N_out;
```

analogous to an analog comparator

Here we want to mimic this particular part like an analog comparator. Because we do not have a pure analog comparator here we can use it. But if you do not have then we are using the MSB of the ADC and this is running at 40, 25 megahertz clock, MSB of the ADC decides whether the output voltage is greater than V ref or less than V ref ok.

That; means, we are getting the data from ADC like only the voltage output, but then we are comparing with NL; N ref. So, whether it is a greater than or less than. So, it is analogous to the analog comparator.

(Refer Slide Time: 14:08)

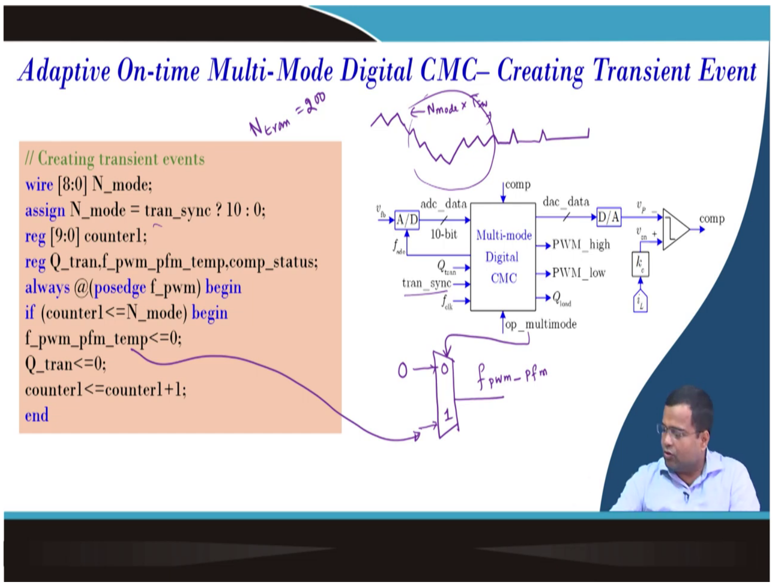


Adaptive On-time Multi-Mode Digital CMC– Module Instantiation

```
//Clock generation circuit
clock_generator u1(.f_clk(clk),.f_adc_clock(clk_adc),.f_dac_clock(clk_dac),.f_sw(f_pwm));
// Digital PI controller
digital_PI_controller u2(.f_pwm(f_pwm),.N_er(N_e),.I_ref(I_peak_pwm),.I_reset(f_pwm_pfm),.K_p(K_p),.K_i(K_i));
// Modulator selection
modulator_selection
u3(.f_clk(clk),.f_pwm(f_pwm),.Q_tr(Q_tr),.rst(comp),.f_pwm_pfm(f_pwm_pfm),.T_min(N_min),.Q_out(Q_g));
//Deadtime circuit
dead_time_circuit u4(.f_clk(clk),.f_pwm_pfm(f_pwm_pfm),.Q_in(Q_g),.Q_H(pwm_high),.Q_L(pwm_low));
```

Now, we are instantiating the clock generator about the digital PI controller modulator selection dead time circuit. So, it consists of this four sub-block inside.

(Refer Slide Time: 14:23)



Adaptive On-time Multi-Mode Digital CMC– Creating Transient Event

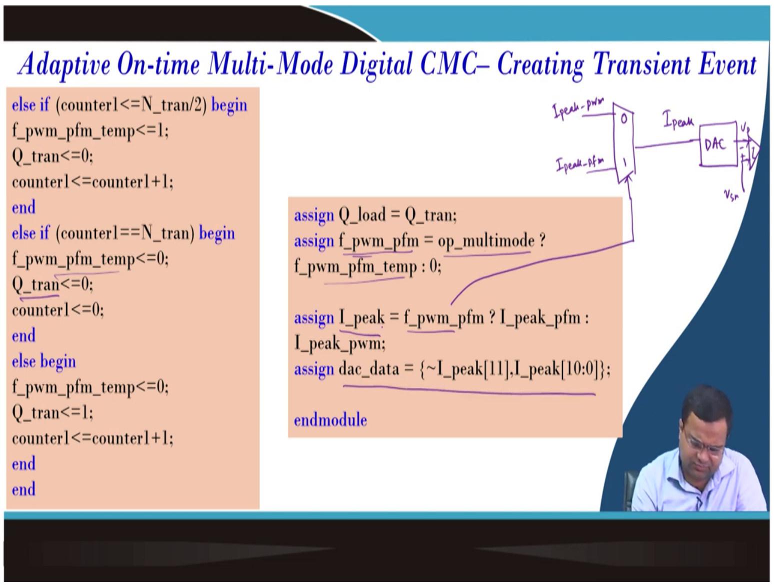
```
// Creating transient events
wire [8:0] N_mode;
assign N_mode = tran_sync ? 10 : 0;
reg [9:0] counter1;
reg Q_tran,f_pwm_pfm_temp,comp_status;
always @(posedge f_pwm) begin
if (counter1<=N_mode) begin
f_pwm_pfm_temp<=0;
Q_tran<=0;
counter1<=counter1+1;
end
```

So, first apart from these four blocks we also have a transient event and this transient event is the same. So, now, here we have a synchronization; that means, whether we want to make the N mode. If you remember during step down transient the current was like this it is coming down this was like this and then goes into or because we are not doing large signal control.

So, I think so coming down like this then it goes into. So, this is the time we call it as this duration is N mode into T s w. So, for these many numbers of switching cycles, we are retaining the PWM operation or we can make it 0 that is why the option. So, the trans sync is whether you keep this or you remove this ok.

Otherwise and it is also generating the same way we have discussed your f PWM; p FM. We will have two options 0 or 1, 0 means it is 0. If the option multimode is 0 it will simply make a PWM operation or it will take this block which is generated from here which is changing and this is created with the multi-mode ok.

(Refer Slide Time: 15:58)



Adaptive On-time Multi-Mode Digital CMC– Creating Transient Event

```
else if (counter1<=N_tran/2) begin
f_pwm_pfm_temp<=1;
Q_tran<=0;
counter1<=counter1+1;
end
else if (counter1==N_tran) begin
f_pwm_pfm_temp<=0;
Q_tran<=0;
counter1<=0;
end
else begin
f_pwm_pfm_temp<=0;
Q_tran<=1;
counter1<=counter1+1;
end
end
```

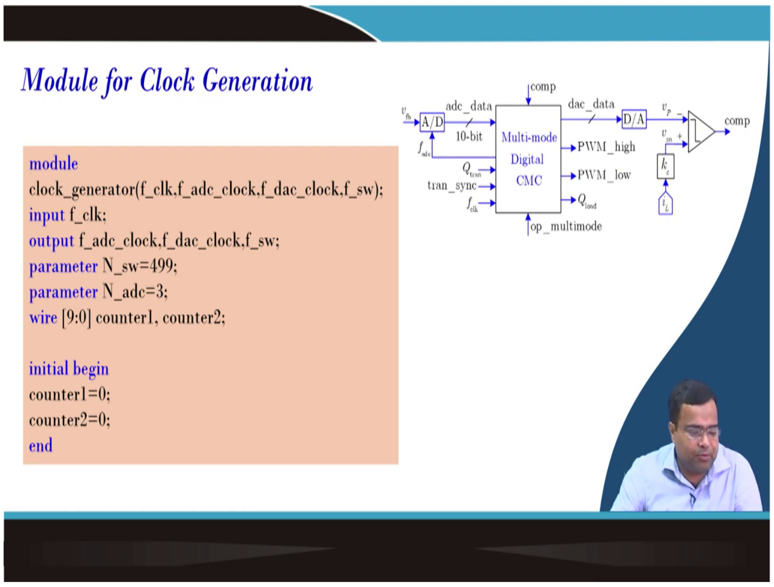
```
assign Q_load = Q_tran;
assign f_pwm_pfm = op_multimode ?
f_pwm_pfm_temp : 0;

assign I_peak = f_pwm_pfm ? I_peak_pfm :
I_peak_pwm;
assign dac_data = {~I_peak[11],I_peak[10:0]};

endmodule
```

So, this is just what I have discussed. Now the load transient here is the Q transient we are making total here. So, I am setting the value transient value to how much so maybe it is there somewhere in several cycles. So, I think it is roughly 100 cycles; that means, we are setting the number of cycle number of the transient to maybe 100 cycles or it can be 200 cycles, 200 cycles, 100 cycles with high load, or 100 cycles with low load.

So, these things we have discussed in detail, and the multi-mode option that I have discussed whether this flag will be set to 0. If the multi-mode is not selected or otherwise it will take how it is selected then the peak reference depends upon multi-mode; that means, we are creating a peak current reference.

I peak and this I peak will eventually go to the DAC and this DAC will have an analog comparator V sm V p minus plus. So, this I peak can be I peak PFM or I peak PWM this is 0, this is 1 and this select line is your this particular line this will decide. If it is 0 it will take sorry this is not the line. So, I will say the select line is here ok and this is required for converting 2 s complement to offset binary.

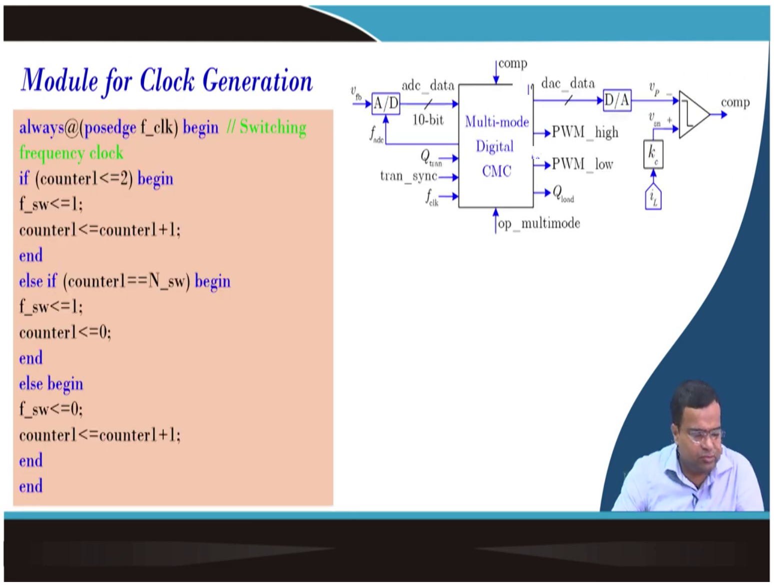
(Refer Slide Time: 18:05)



Module for Clock Generation

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

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

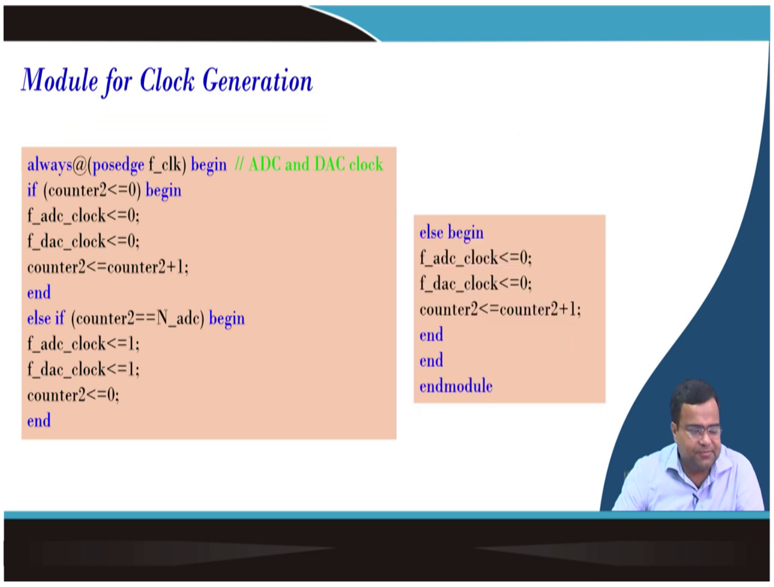
(Refer Slide Time: 18:08)



Module for Clock Generation

```
always@(posedge f_clk) begin // Switching
frequency clock
if (counter1<=2) 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
```

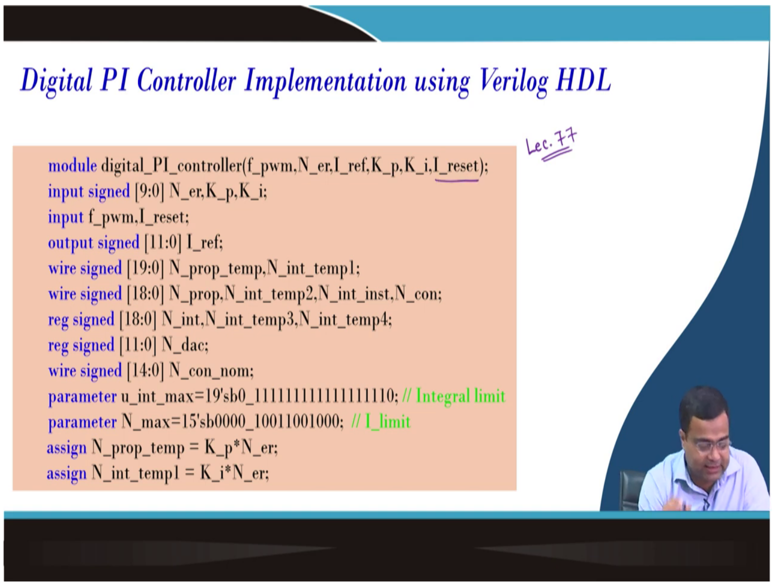
(Refer Slide Time: 18:10)



Module for Clock Generation

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

(Refer Slide Time: 18:13)



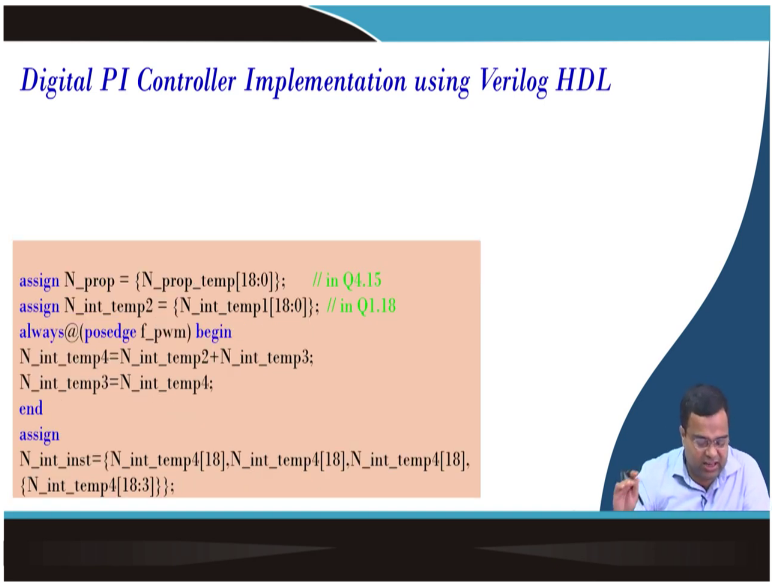
Digital PI Controller Implementation using Verilog HDL

Lec. 77

```
module digital_PI_controller(f_pwm,N_er,I_ref,K_p,K_i,I_reset);
input signed [9:0] N_er,K_p,K_i;
input f_pwm,I_reset;
output signed [11:0] I_ref;
wire signed [19:0] N_prop_temp,N_int_temp1;
wire signed [18:0] N_prop,N_int_temp2,N_int_inst,N_con;
reg signed [18:0] N_int,N_int_temp3,N_int_temp4;
reg signed [11:0] N_dac;
wire signed [14:0] N_con_nom;
parameter u_int_max=19'sb0_111111111111111110; // Integral limit
parameter N_max=15'sb0000_10011001000;  // I_limit
assign N_prop_temp = K_p*N_er;
assign N_int_temp1 = K_i*N_er;
```

Now, a module for clock generation is these things we have discussed. So, we are not going to repeat how to digital PI controller. I think this we have discussed I think it is lecture number. I believe it was 77, In lecture 77 we discussed it in detail. We have only added the reset component which we have discussed; that means if we do not use this PI controller or means PWM control. Then we will make sure this reset if it is high the integral output is 0.

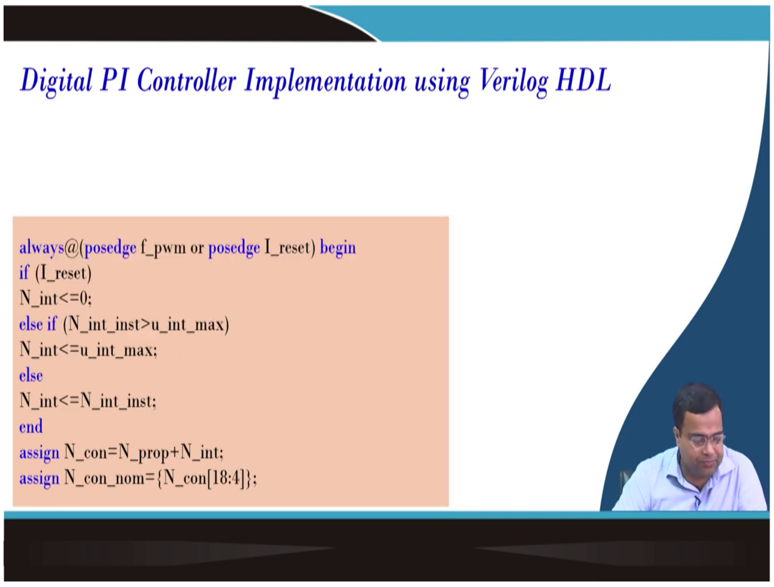
(Refer Slide Time: 18:28)



Digital PI Controller Implementation using Verilog HDL

```
assign N_prop = {N_prop_temp[18:0]};      // in Q4.15
assign N_int_temp2 = {N_int_temp1[18:0]}; // in Q1.18
always@(posedge f_pwm) begin
N_int_temp4=N_int_temp2+N_int_temp3;
N_int_temp3=N_int_temp4;
end
assign
N_int_inst={N_int_temp4[18],N_int_temp4[18],N_int_temp4[18],
{N_int_temp4[18:3]}};
```

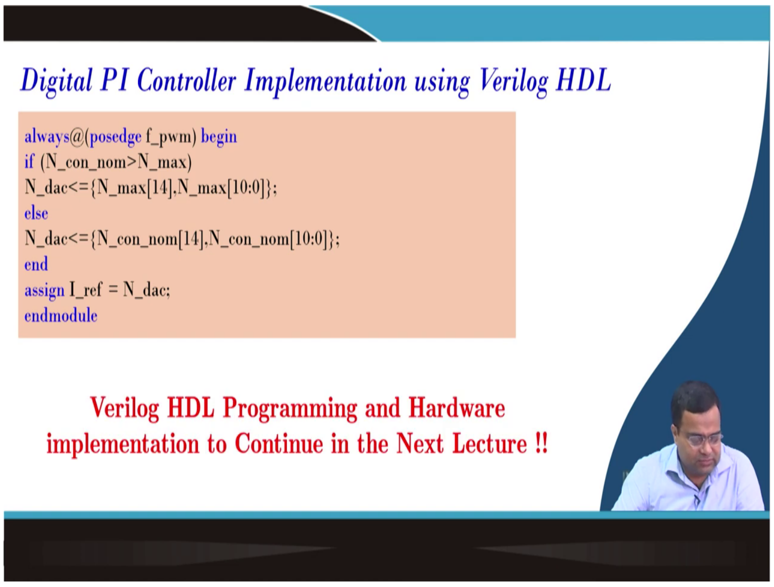
(Refer Slide Time: 18:33)



Digital PI Controller Implementation using Verilog HDL

```
always@(posedge f_pwm or posedge I_reset) 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=N_prop+N_int;
assign N_con_nom={N_con[18:4]};
```

(Refer Slide Time: 18:44)



Digital PI Controller Implementation using Verilog HDL

```
always@(posedge f_pwm) begin
if (N_con_nom>N_max)
N_dac<={N_max[14],N_max[10:0]};
else
N_dac<={N_con_nom[14],N_con_nom[10:0]};
end
assign I_ref = N_dac;
endmodule
```

Verilog HDL Programming and Hardware implementation to Continue in the Next Lecture !!

So; that means, we are making an integral reset when the PWM control is not activated that is the PI controller. Now we are going to continue this Verilog code for the other block in the next lecture that is it for today.

Thank you very much.