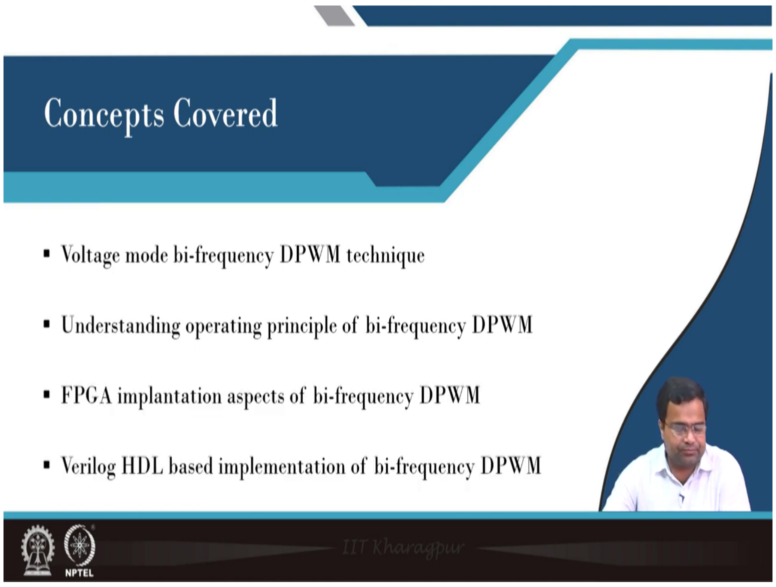
Digital Control in Switched Mode Power Converters and FPGA-based Prototyping

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

Module - 12
Implementation of Multimode Digital Control and Course Summary
Lecture - 111
Implementing Bi-frequency Spread Spectrum in Digital VMC using Verilog HDL

Welcome. In this lecture, we are going to consider this as perhaps the last week we are going to consider the Bi Frequency Spread Spectrum Technique in the Digital Voltage Mode Control and how to implement using Verilog HDL.

(Refer Slide Time: 00:39)



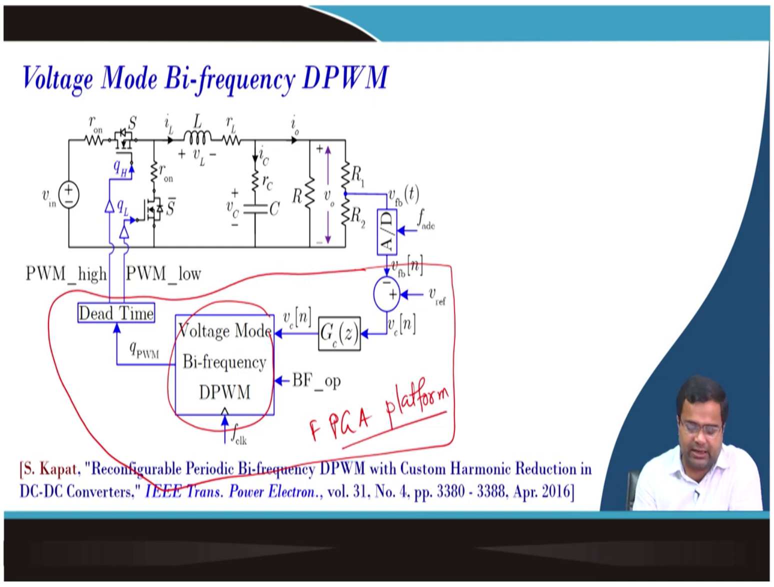
Concepts Covered

- Voltage mode bi-frequency DPWM technique
- Understanding operating principle of bi-frequency DPWM
- FPGA implantation aspects of bi-frequency DPWM
- Verilog HDL based implementation of bi-frequency DPWM

IIT Kharagpur
NPTEL

So, here we will first talk about the voltage mode bi-frequency DPWM technique and we need to understand the operating principle of bi-frequency DPWM and some FPGA implication aspects finally, Verilog HDL-based implementation.

(Refer Slide Time: 00:55)



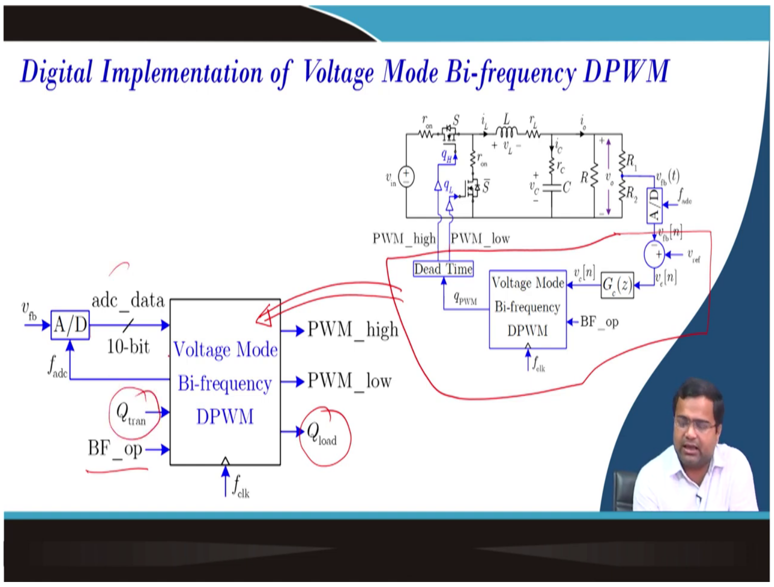
Voltage Mode Bi-frequency DPWM

[S. Kapat, "Reconfigurable Periodic Bi-frequency DPWM with Custom Harmonic Reduction in DC-DC Converters," *IEEE Trans. Power Electron.*, vol. 31, No. 4, pp. 3380 - 3388, Apr. 2016]

So, if we consider bi-frequency DPWM you know we have already considered a voltage mode control. You know if we take from this block to this block. We have already implemented using FPGA. So, this is what we are making implementing using the FPGA platform, but one can use a microcontroller also no problem, but this is under a digital platform. So, it has a specific modulator which is a voltage mode bi-frequency DPWM.

So, how does it work? And actually, we have discussed in detail about this technique in this research paper. Where if you take first we want to design this block?

(Refer Slide Time: 01:31)

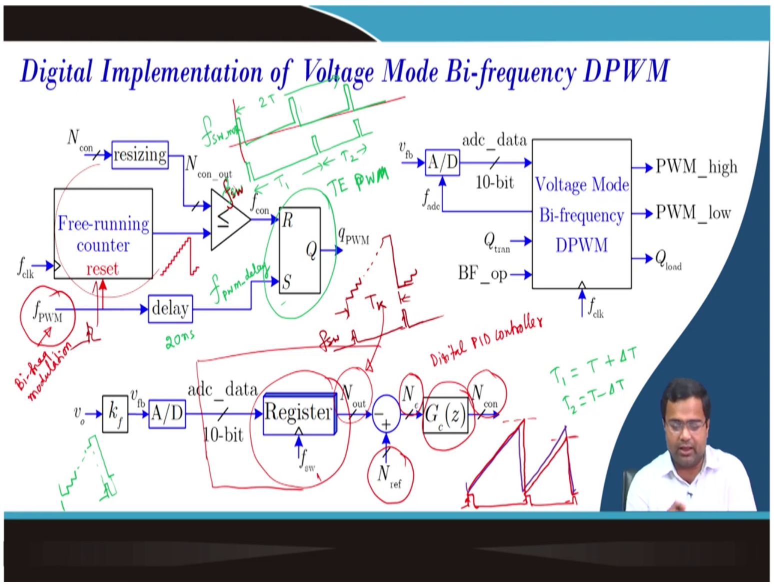


Digital Implementation of Voltage Mode Bi-frequency DPWM

This is where our implementation block is. And we know that actually, this whole block is coming from the ADC data I think to the dead time. This is the block that we are going to implement, where it is going gate signal is going. So, now, we have an additional option from outside that whether you want bi-frequency or not.

If you stop it will become a normal DPWM and otherwise we have all discussed the load transient we can make, the transient type whether load or reference transient ADC data is coming here we are not considering it because it is voltage mode control.

(Refer Slide Time: 02:20)



So, how does it operate? Here if you look at this there is a free-running counter ok. So, first of all, we know about this ADC data it is going inside the FPGA and then we are using a register and this is the switching frequency clock and with sync, with the clock we are taking the ADC data and that is our output voltage.

So, this is related to your digital number related to the output voltage. And this is the reference command then we are generating the error voltage and this is going inside your digital. So, here we are using a digital PID controller because it is a voltage mode control PID controller.

The output of the controller is the N con which is the controller output. So, N con we have to resize and this is the counter free running counter and this is the block which is responsible for DPWM. So; that means if you consider a traditional I would say nominal; that means if you say this clock. So, this is our f fsw nominal.

Now, we may consider another, which will be like this. It will be slightly higher than it will be smaller. So; that means as if we are using T1 and T2 and originally it was like twice T. So; that means, we are taking T1 to be T plus delta T and T2 to be T minus delta T. Now, it is not necessary that we need to change in alternative cycle. Maybe we can consider T plus delta T for some subsequent cycle followed by T minus delta T in another subsequent series of cycles, I mean several cycles. So, that will decide the actual fsw clock.

So, the actual fsw clock is a bi-frequency clock and this clock is edge whenever the clock's positive edge comes this will reset the counter. So; that means, it is a free-running counter. So, this counter will be a free-running counter. So, it will reset. So; that means, in this period whether it is a T1 T k i will say whether T1 and T2 will be decided by our this switching frequency clock fsw clock.

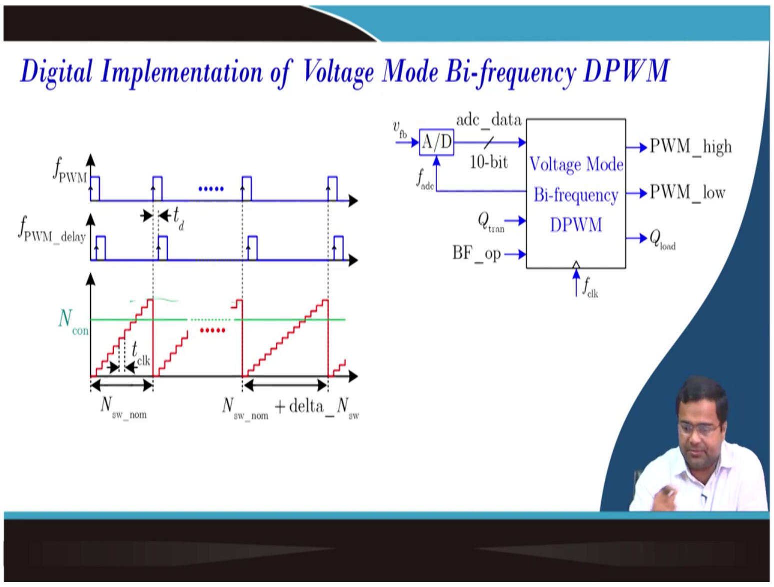
So, this fsw clock if we; that means if we create a bi-frequency clock; means, frequency hopping using this clock by varying time period. This will simply automatically generate the sawtooth waveform which has the same slope, but varying amplitude; that means, nominally the sawtooth will be like this in nominal condition, but under bi-frequency modulation, it will look something like this. It will sorry I think it should. So, if we consider the nominal case.

So, in nominal case it will be like this it will reset like this, but what bi frequency it will do it will continue, it will reset, then it will continue, and it might reset here. So; that means, twice the T period may be the same. So, we are essentially modulating, in this case, the peak value of you know the sawtooth waveform, but the slope is the same. So, this results in variation in the time period and that whole thing is done by this fsw clock. So, which is varied.

So; that means, the synthesis of this block is of prime concern and this will come from a bi-frequency modulation block ok. Now, in digital circuits whenever you know suppose if we consider you know let us say this case suppose if we apply this kind of counter and if we go like this. So, we want to suppose if the reset pulse comes here; means, if the reset pulse; that means, whenever it sees the frequency pulse comes when it is equal the action takes place in the next cycle.

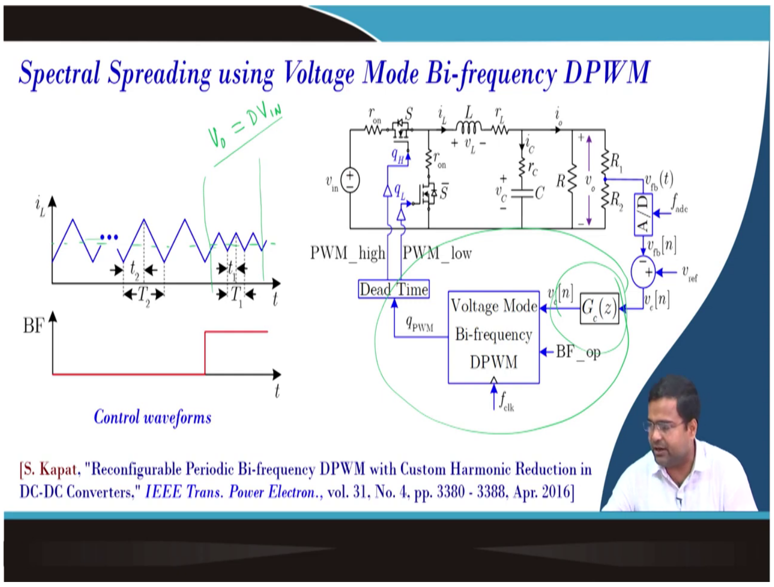
So, to avoid any false triggering we are pitting a delay and this delay presently we are giving 20 nanosecond delay, which can be 10 nanosecond. So, this is the fpwm delay that will be set, because this is regular this is what is our trailing edge modulation PWM, PWM modulator trailing edge PWM modulation ok.

(Refer Slide Time: 07:43)


Digital Implementation of Voltage Mode Bi-frequency DPWM

So, this is all about this principle. How does it work? I have told you it is a nominal case this is the same, but now we can periodically vary and this is what we are going to discuss in the Verilog code.

(Refer Slide Time: 07:53)


Spectral Spreading using Voltage Mode Bi-frequency DPWM

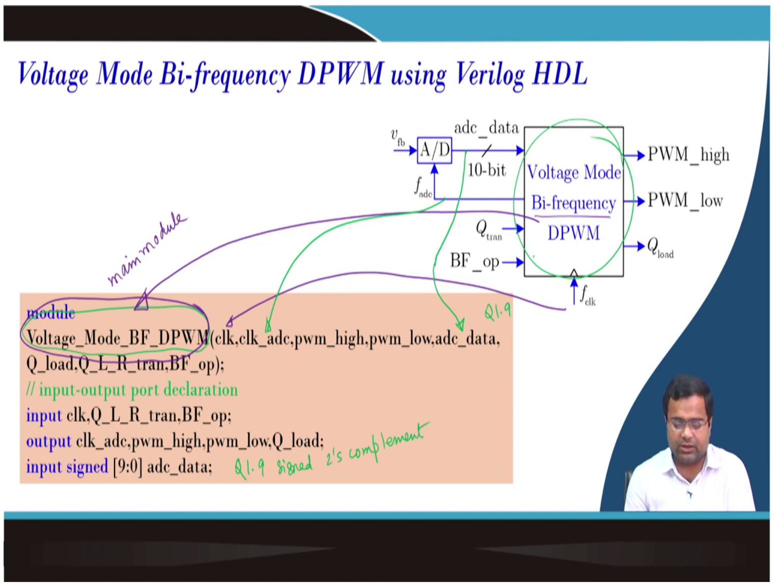
[S. Kapat, "Reconfigurable Periodic Bi-frequency DPWM with Custom Harmonic Reduction in DC-DC Converters," *IEEE Trans. Power Electron.*, vol. 31, No. 4, pp. 3380 - 3388, Apr. 2016]

So; that means, what we are doing if we vary the time period the loop is closed all the time we are not turning off the loop. So, the loop is always closed. Let us say we have if we considered this average current of a buck converter if the load does not change then we know that the average current under a steady state should be average inductance should be equal to the load current. So, under bi frequency, if you let us say 20 cycles if we use 1 time period another 20 cycles another time period.

But the average current must be the same and we know if we take the ideal buck converter V0 to be equal to D into Vin; that means if we change the switching frequency or switching period not too drastically then this approximation is valid. So, if we change the time period essentially duty ratio remains the same. So, your duty ratio will be automatically adjusted. So, the average current will remain the same that is the load current if I do not consider the transient effect.

So; that means, this paper we have discussed in detail; that means, if we close the loop all the time and make the modification in the time period digitally then we can retain more or less closely the transient response using without and with bi-frequency operation at the same time we are going to achieve the spectral spreading with minimum effect. That is what we are going to show in this lecture as well as the next lecture using experimental results.

(Refer Slide Time: 09:30)



Voltage Mode Bi-frequency DPWM using Verilog HDL

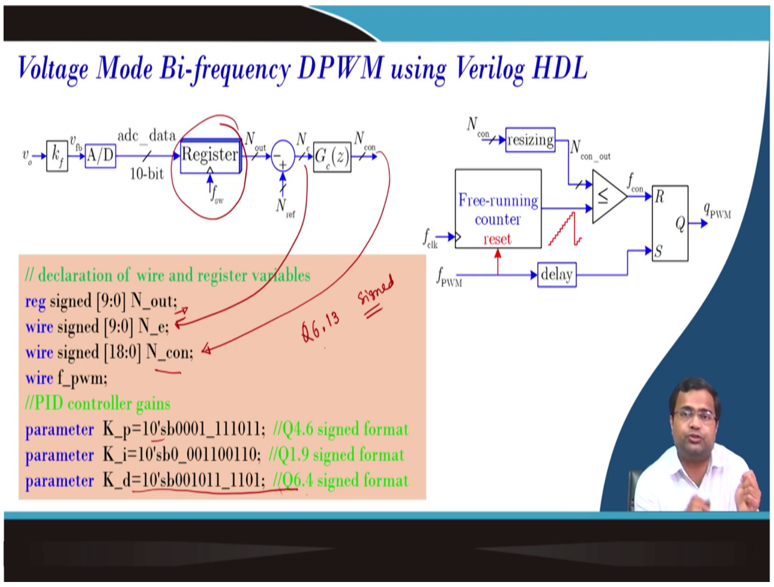
```
module
Voltage_Mode_BF_DPWM(clk,clk_adc,pwm_high,pwm_low,adc_data,
Q_load,Q_L_R_tran,BF_op);
// input-output port declaration
input clk,Q_L_R_tran,BF_op;
output clk_adc,pwm_high,pwm_low,Q_load;
input signed [9:0] adc_data;   Q1.9 signed 2's complement
```

So, how to implement this block? So, we are writing in the main module. So, this main module name is you can see voltage mode is the name of the main module. So, let me use a different color. So, this is the name of the main module. This is consistent with this name, right? So, maybe this terminology you can see is the name of the main module, the main module. So, what is getting interfaced with the main module? Because we are not using any DAC. So, we have a clock which is this clock ok. Now you can map the clock of the ADC that you can map this is the ADC clock.

So, this is a map then the data coming from the ADC this data is nothing but data. That is the 10-bit Q1 dot 9 format. We can define that as Q1 dot 9 format signed data. It is in 2s complement ok. Sorry 2s complement signed data is already written 2s complement data. Next, we, want to generate PWM low and this is PWM high ok. Then the Q load is here, then Q transient this part is this. And finally, we are talking about this bi frequency option this is a bi-frequency option.

So, this gives us all the mapping it is communicating with the external devices. So, all these are defined here.

(Refer Slide Time: 11:28)



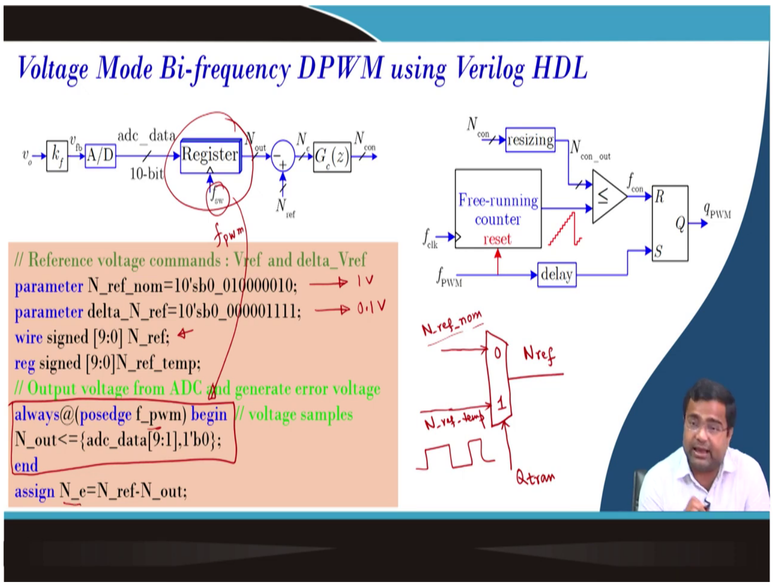
Voltage Mode Bi-frequency DPWM using Verilog HDL

```
// declaration of  wire and register variables
reg signed [9:0] N_out;
wire signed [9:0] N_e;
wire signed [18:0] N_con;
wire f_pwm;
//PID controller gains
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
```

Next, we are going for the next step. So, we are defining the register which is our output voltage and we know a register the output will be the clock synchronized. The error voltage here which is coming out should be signed, because it is a subtraction and we will show it is simply a wire connection N con is the controller output that we are taking 19 bits and I will show what is the format right now it is difficult to say.

I think it should be Q6 dot 13 formats; that means, 6 bit is the integer side it is the signed in 2s complement. I think. So, I will we will check, because you see the proportional gain in 4 dot 6 integral gain 1 dot 9, and derivative gain Q dot 6, 6 dot 4. So, it should be 6, because that is the highest. So, the largest integer bit is defined by this one as well as the error voltage which is 1 dot 9.

(Refer Slide Time: 12:34)



Voltage Mode Bi-frequency DPWM using Verilog HDL

```
// Reference voltage commands : Vref and delta_Vref
parameter N_ref_nom=10'sb0_010000010;
parameter delta_N_ref=10'sb0_000001111;
wire signed [9:0] N_ref;
reg signed [9:0]N_ref_temp;
// Output voltage from ADC and generate error voltage
always@(posedge f_pwm) begin // voltage samples
N_out<={adc_data[9:1],1'b0};
end
assign N_e=N_ref-N_out;
```

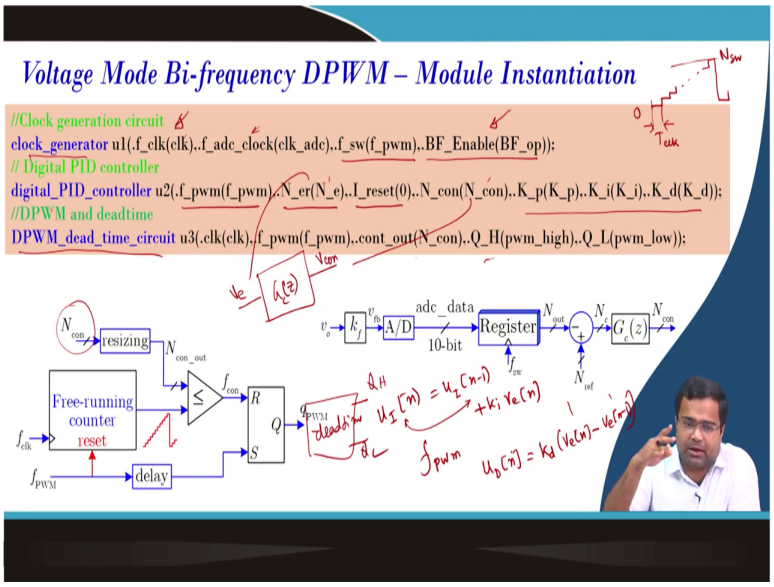
So, now the reference command we are using a nominal reference voltage to this corresponds to 1 volt. And we can also make a reference transient which corresponds to 0.1 volt in the chain. And we are taking the reference voltage which is a wire connection. And we have discussed that reference is coming from what? two things this is our N ref it is coming from where it is coming our it can be N ref nominal; that means, N ref nominal or it can be N ref temp.

This is coming from a resistor because this can change if we want to make a transient effect, because it is added with delta I ref or it is simply a nominal voltage. So, this can be changed; that means if you want to take the transient effect; that means, we can make a reference transient by giving a suitable transient; which means, your type of transient which is like a Qlr. If you go to the last block so, this is the transient type whether you want a low transient or reference transient. So; that means, this is your Q tran.

In this name particular module, this is this signal this is this Qlr this Qlr term either load. So, if it is 0 it is load transient; that means, I would say if it is 0 it is load transient, and if it is 1 it is a reference transient. So; that means, it is 1 and 0 if it is 0 loads transient that case it will take the nominal reference voltage because we are not making simultaneous load and reference transient. If it is one it will take the reference transient.

Similarly, for the load case also we can consider the load transient or we will take a fixed load and make a reference transient. Here this block captures the output voltage; that means, this is this particular register this is coming here and with respect to the PWM clock which is fsw. So, it is fpwm here and then we are making the error voltage.

(Refer Slide Time: 14:53)



Voltage Mode Bi-frequency DPWM – Module Instantiation

```
//Clock generation circuit
clock_generator u1(.f_clk(clk),.f_adc_clock(clk_adc),.f_sw(f_pwm),.BF_Enable(BF_op));
// Digital PID controller
digital_PID_controller u2(.f_pwm(f_pwm),.N_er(N_e),.I_reset(0),.N_con(N_con),.K_p(K_p),.K_i(K_i),.K_d(K_d));
//DPWM and deadtime
DPWM_dead_time_circuit u3(.clk(clk),.f_pwm(f_pwm),.cont_out(N_con),.Q_H(pwm_high),.Q_L(pwm_low));
```

Then this error now we have 3 circuits is the clock generator this is the most important part. Because it also has a bi-frequency enable the option. Earlier when we talk about digital voltage mode control we did not consider any bi-frequency then it was giving a fixed frequency PWM where we set the fixed value of the counter because it will counter base DPWM. So; that means, it will generate a fix and if we set the upper limit we use now it is starting from 0 to n s w and then it is resetting.

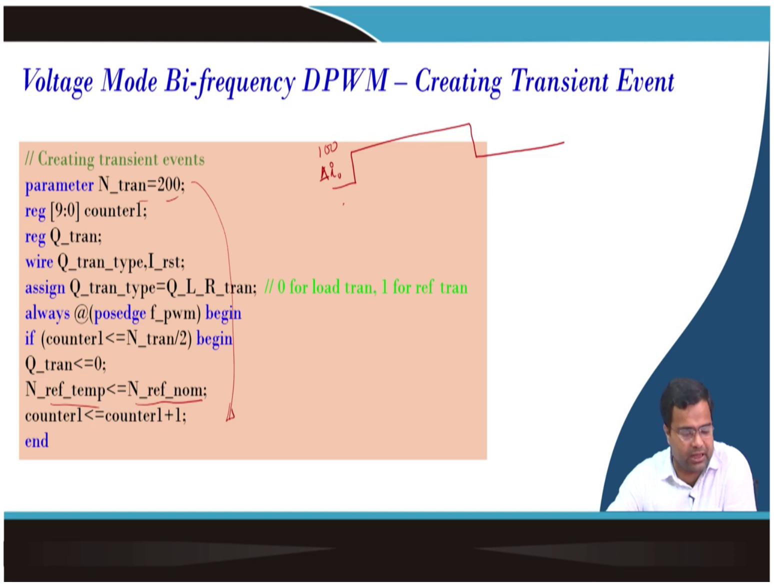
So, this was generated and we have used a high-frequency clock of time period T clock this corresponds to your f clock which is the high-frequency 100 megahertz clock. So, that is used and f ADC clock then f pwm clock which is coming out, and then enables the option. So, if this enables 1 then we are enabling bi frequency if it is 0, it will simply be dpwm and will go inside it has a digital PID controller where it takes the input f PWM because we have an integral action and derivative action.

Because integral we know u I of n is equal to u I of n minus 1 plus k i into Vn. So, this previous and the present value you know the transfer of register from current to previous or previous to current this requires a clock and this is in the edge of f pwm. Similarly for the derivative control, we have what; that means, we have k d into V e n minus V e n minus 1. This also requires a clock to generate the previous and the current. So, that is why we are using p w m clock. The error signal is the input to the PID controller. That is your PID controller that is your error, it is like an error and then this is your controller.

So, in the digital number, it is an N e and an N con. This corresponds to this and this corresponds to this. We are keeping I reset here we are reset is not enabled if the reset is enabled then the integral action will be disabled or the initial value will be 0. If it is one it will be always like integral action is N action then we are importing the parameter k p ki kd which we set outside and in this case, we have to consider I will I we will take whenever we will take the experimental case study we will say what is the value that we have to consider ok.

The next block is the PWM and dead time. So, this block we know that this is the pwm block. This is the block that will generate your Q PWM and it will also have a dead time after that it will have a dead time. That will QH and QL this is QH and QL and here the input to this is the controller output that will be compared with the sawtooth right and the sawtooth slope is the same. But its peak value is changing if there is a bi-frequency operation.

(Refer Slide Time: 18:07)

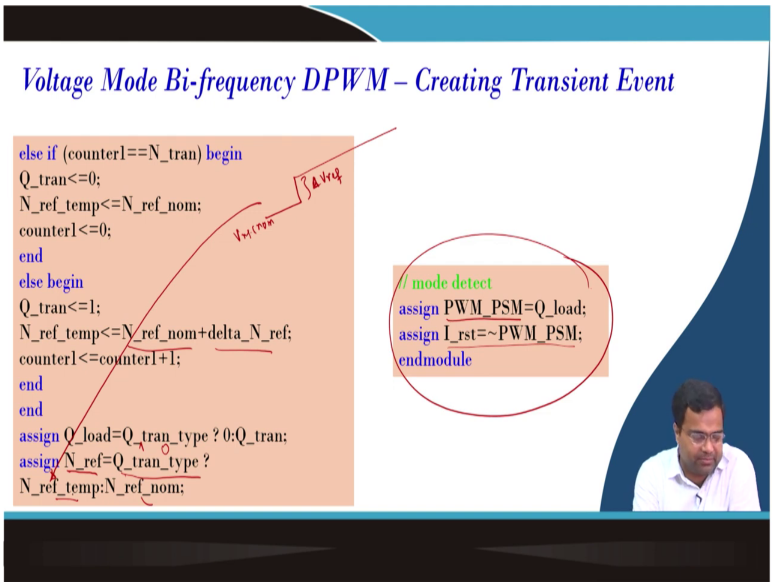


Voltage Mode Bi-frequency DPWM – Creating Transient Event

```
// Creating transient events
parameter N_tran=200;
reg [9:0] counter1;
reg Q_tran;
wire Q_tran_type,I_rst;
assign Q_tran_type=Q_L_R_tran;  // 0 for load tran, 1 for ref  tran
always @(posedge f_pwm) begin
if (counter1<=N_tran/2) begin
Q_tran<=0;
N_ref_temp<=N_ref_nom;
counter1<=counter1+1;
end
```

So, now we are making a transient event. I think this thing we have discussed in lecture number 74, and 73 in voltage mode is to create a load or reference transient. So, we are using 200; that means, first hundred cycles it will be one load current than another load current. So; that means like a load step adding and discarding. It is adding and discarding and continuous load is always connected.

(Refer Slide Time: 18:42)



Voltage Mode Bi-frequency DPWM – Creating Transient Event

```
else if (counter1==N_tran) begin
Q_tran<=0;
N_ref_temp<=N_ref_nom;
counter1<=0;
end
else begin
Q_tran<=1;
N_ref_temp<=N_ref_nom+delta_N_ref;
counter1<=counter1+1;
end
end
assign Q_load=Q_tran_type ? 0:Q_tran;
assign N_ref=Q_tran_type ?
N_ref_temp:N_ref_nom;
```

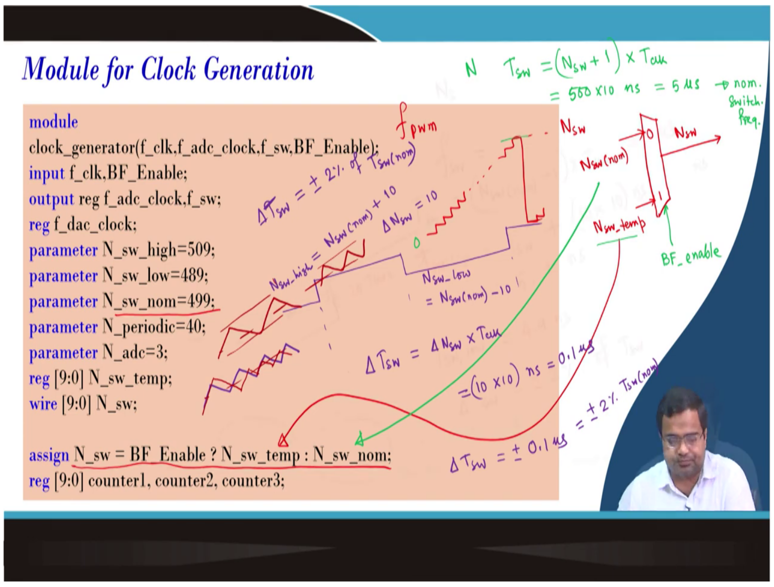
```
// mode detect
assign PWM_PSM=Q_load;
assign I_rst=~PWM_PSM;
endmodule
```

So, that is why it is created and we can all it is also created a reference one nominal value for the half cycle and the remaining half cycle nominal plus delta; that means, we are also changing V ref to this. So, this is your V ref nominal and this is our delta V ref ok. So, V ref is also changing if it is possible.

And you see here we are assigning the reference voltage to be Q ref tran; that means, if this is 0 then it will take the nominal value if it is 1 it will take this which creates a transient effect; that means, this guy that overall is this signal; that means, temp; that means, it will keep on changing after 100 cycles.

But if we take load transient which is transient type 0. It will take the nominal value ok. So, then assign we can make PWM psm, but we are not making multimode at this point. So; that means, we are setting 0 and reset we can make. So, this will be used for mode when you go for multi-mode right now we are not using it. So, Q load we are making load transient, and here we are making load transient we are directly taking from this you know this signal; that means, transient type Q tran type.

(Refer Slide Time: 20:20)



Module for Clock Generation

```
module
clock_generator(f_clk,f_adc_clock,f_sw,BF_Enable);
input f_clk,BF_Enable;
output reg f_adc_clock,f_sw;
reg f_dac_clock;
parameter N_sw_high=509;
parameter N_sw_low=489;
parameter N_sw_nom=499;
parameter N_periodic=40;
parameter N_adc=3;
reg [9:0] N_sw_temp;
wire [9:0] N_sw;

assign N_sw = BF_Enable ? N_sw_temp : N_sw_nom;
reg [9:0] counter1, counter2, counter3;
```

f_{pwm}
N
$T_{sw} = (N_{sw}+1) \times T_{clk}$
$= 500 \times 10$ ns $= 5\,\mu s \to$ nom. switch. freq.
$\Delta T_{sw} = \pm 2\%$ of $T_{sw(nom)}$
$N_{sw\_high} = N_{sw(nom)} + 10$
$\Delta N_{sw} = 10$
0
$N_{sw\_low} = N_{sw(nom)} - 10$
N_{sw}
$N_{sw(nom)}$
N_{sw_temp}
BF_enable
0
1
N_{sw}
$\Delta T_{sw} = \Delta N_{sw} \times T_{clk}$
$= (10 \times 10)$ ns $= 0.1\,\mu s$
$\Delta T_{sw} = \pm 0.1\,\mu s = \pm 2\%\ T_{sw(nom)}$

Next, we are going to show the module for clock generation. So, this is one of the most important blocks, because where we are going to consider the bi-frequency operation. Earlier in this block, we straight away took the 499 now this is the nominal value. So, let us consider you know we want to generate f. So; that means, we want to generate an f p wm clock. So, how to generate?

So, let us say we have a counter and this counter will reach up to N sw then it will again reset and it will now. Now, this N sw is coming from if you see this assigned statement it is a mux combination; that means, it has a mux combination 0 and 1. If it is 0 then it is taking N sw nominal and this mux output is N sw or it is taking N sw temp. So, N sw temp is this signal and N sw nominal is this signal ok.

So; that means, what is the select line? The select line here is bi frequency enable; that means, if the bi frequency enables is 0 it will not do any bi frequency operation dpwm it will be a nominal value (Refer Time: 22.05). Now, what is the nominal value, and what is our time period? T sw it will be N sw plus, because this is starting from 0 to up to N sw. N sw plus 1 into T clock ok. And in this case what it is coming; means, is the value is coming total of 500 in 10 nanosecond which is equal to 5 microsecond right?

So, that is our is our nominal switching frequency, but in this case, we have taken the other 2 values; that means, this N sw temp is generated from a high low kind of pulses; which means, it can be changed. So, where this is like N sw high and this is N sw low. What is N sw high?

N sw high is N sw nominal plus 10 you can see because it is changing it is 419 10 and this is N sw. What is this? This is equal to N sw nominal minus 10.

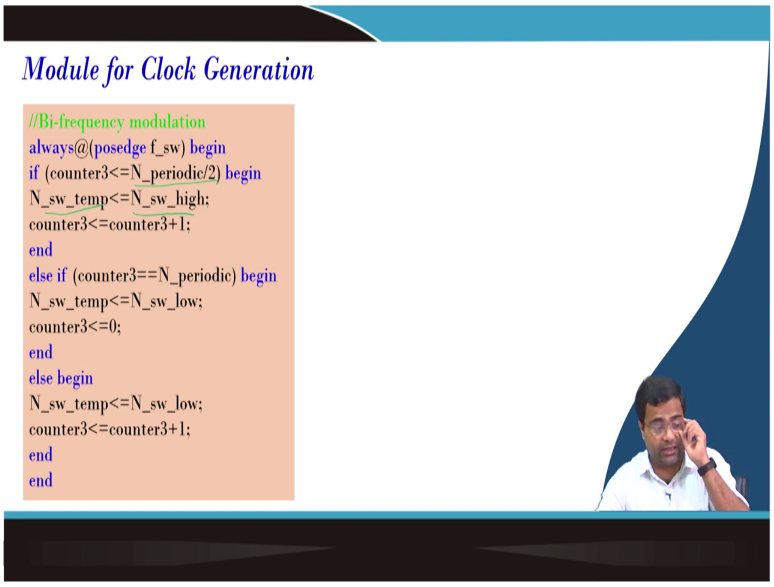
So, we have a delta variation; that means as if there is a delta N sw which is 10; that means, what is our delta T sw? Delta T sw will be equal to delta N sw into T clock and T clock is 10 nanoseconds. So, it will be 10 in 10 nanosecond. So, it is like a 0.1 microsecond. So, what is the net change? In; that means, 0.1 microsecond; that means, we have delta T s that can be positive. So, the positive value is the positive value, or it is the magnitude.

So, I would say. So, delta T s can vary it is basically plus minus 0.1 microsecond and if we take the time period of five microsecond it is nothing but so; that means, delta T sw it is plus minus 2 percent of T sw nominal. T sw nominal is; that means, it is plus minus 2 percent of T sw nominal. So, you can change it user can change it 4 percent, or 5 percent we have just taken 2 percent. This is just a value just to show that the bi-frequency of the spectral spreading is happening ok.

If you take too large then it may cause some ripple impact. We want to because if we take the nominal inductor current without any bi frequency this will be the case, but we are talking about a scenario where the period can be large ok. And then we are also talking about a scenario where the time period can be small also. So, like this. So, we are talking about; that means, their e is a talarlarge period for a few-cycle followed by there is a small time period for a few cycles.

So, we want to make sure that peak to peak effect is not significant because otherwise, the overall RMS current can increase. So, we want to reduce that effect that is why this delta variation we have taken 2 plus minus 2 percent. In this case, it is user-dependent on how much it is acceptable ok.

(Refer Slide Time: 26:27)

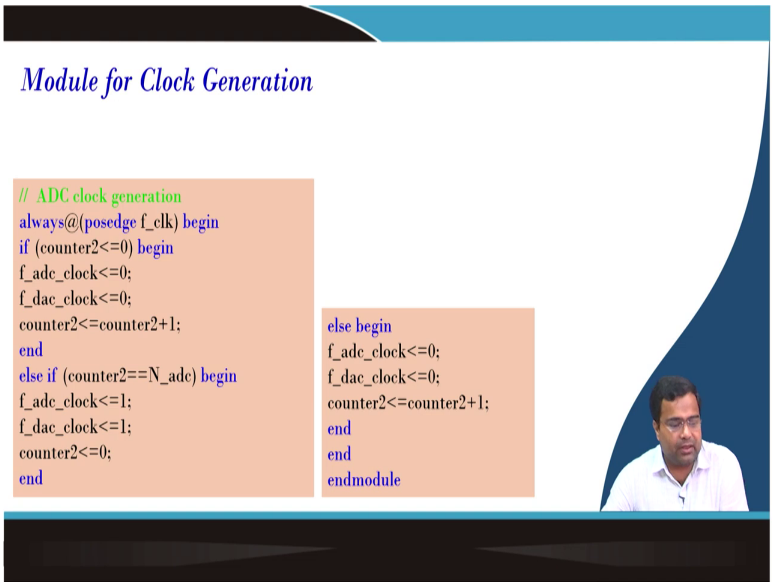


Module for Clock Generation

```
//Bi-frequency modulation
always@(posedge f_sw) begin
if (counter3<=N_periodic/2) begin
N_sw_temp<=N_sw_high;
counter3<=counter3+1;
end
else if (counter3==N_periodic) begin
N_sw_temp<=N_sw_low;
counter3<=0;
end
else begin
N_sw_temp<=N_sw_low;
counter3<=counter3+1;
end
end
```

So, this will keep on changing and this is what is created 50 percent time sorry the 3rd clock is up to half of this you see it will take N SW high the remaining half will take N SW low.

(Refer Slide Time: 26:37)



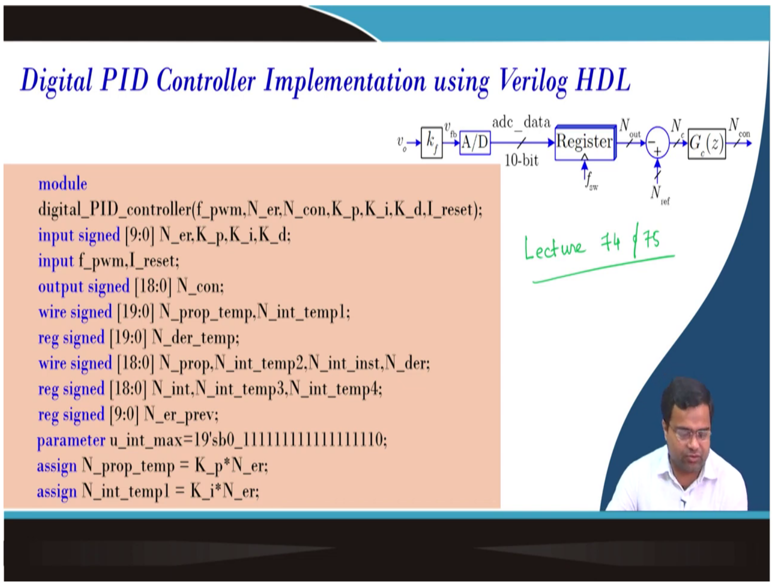
Module for Clock Generation

```
// ADC clock generation
always@(posedge f_clk) begin
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
```

So, it will just be for changing output and the clock generation I think we have discussed how to generate the ADC clock we are just taking a 20 megahertz clock.

(Refer Slide Time: 26:45)

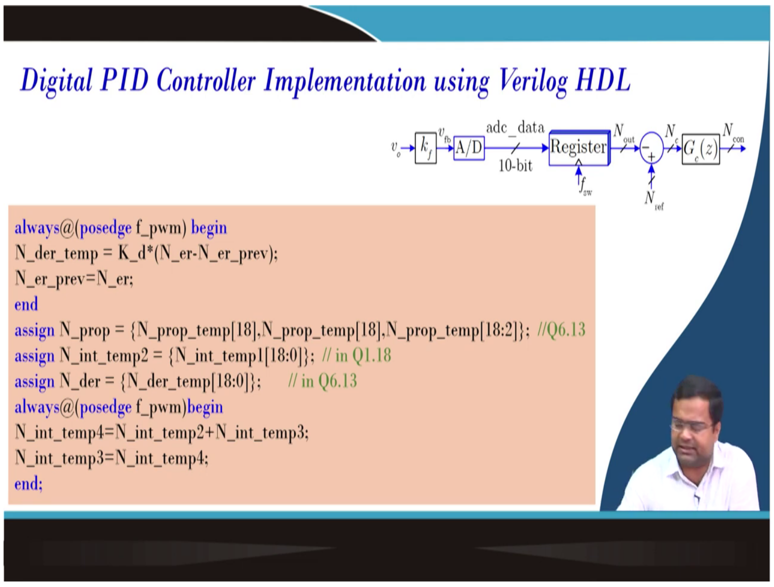


Digital PID Controller Implementation using Verilog HDL

```
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 [18:0] 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;
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_11111111111111110;
assign N_prop_temp = K_p*N_er;
assign N_int_temp1 = K_i*N_er;
```

And the digital PID controller we have discussed; that means, earlier in sufficient detail. So, I am not going to because if you go to I think lecture number I believe it will be 74 as well as 75 we have discussed the detailed implementation of digital PID controller.

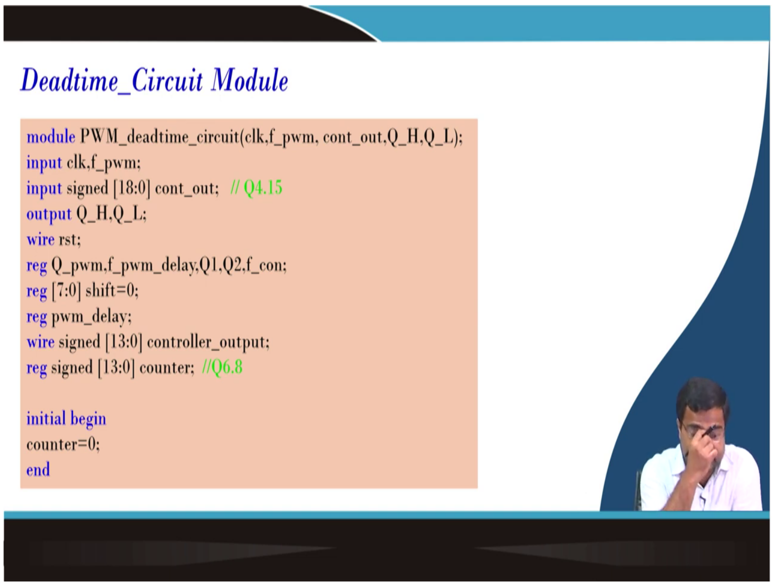
(Refer Slide Time: 27:07)



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;
```

(Refer Slide Time: 27:13)

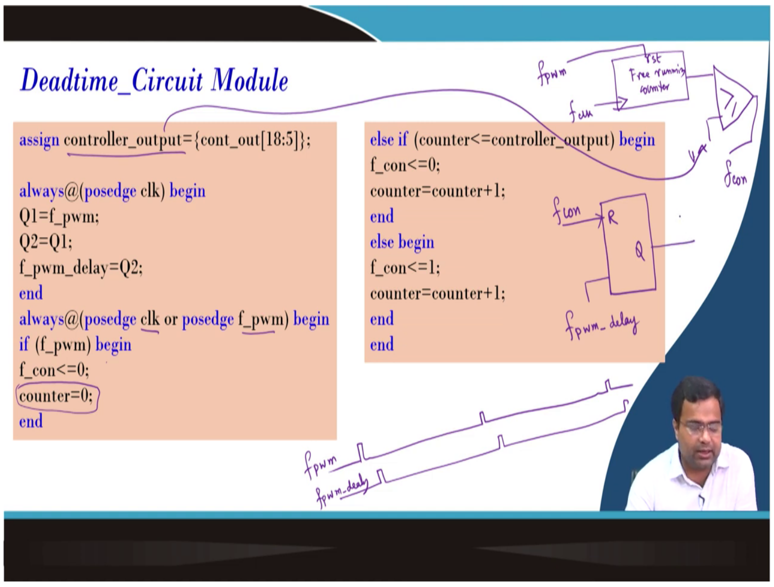


Deadtime_Circuit Module

```
module PWM_deadtime_circuit(clk,f_pwm, cont_out,Q_H,Q_L);
input clk,f_pwm;
input signed [18:0] cont_out;  // Q4.15
output Q_H,Q_L;
wire rst;
reg Q_pwm,f_pwm_delay,Q1,Q2,f_con;
reg [7:0] shift=0;
reg pwm_delay;
wire signed [13:0] controller_output;
reg signed [13:0] counter; //Q6.8

initial begin
counter=0;
end
```

(Refer Slide Time: 27:25)



Deadtime_Circuit Module

```
assign controller_output={cont_out[18:5]};

always@(posedge clk) begin
Q1=f_pwm;
Q2=Q1;
f_pwm_delay=Q2;
end
always@(posedge 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
else begin
f_con<=1;
counter=counter+1;
end
end
```

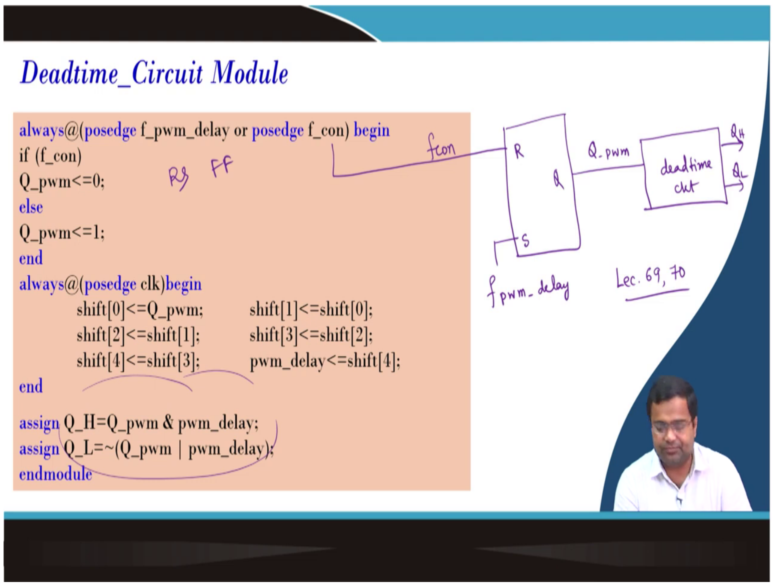
fpwm
rst
Free running counter
fclk
fcon
fcon
R
Q
fpwm_delay
fpwm
fpwm_delay

So, we are not going to implement the show again. So, how to resize the data all these things are discussed. So, this is the output of the digital PID controller and the dead time circuit is simply what we are doing it is a DPWM plus dead time. It has both DPWM plus dead time. What it does do? The controller output will resize then it will take always the posedge. So, this is 1 delayed clock, because we have discussed that we have I mean let me use a different color.

We have let us say fpwm and we are generating a delayed signal this is our fpwm delay and we have used fpwm to make a free running counter this is our f clock and we have a reset of this counter and this reset it is active high it is using PWM signal. And this counter output and we have to take the same Q format consistent Q format it is compared with you can say it is a digital comparator it is our V con resize; that means, the controller output.

So, this will be coming here. Controller output if the controller output is that mean if this guy is greater than equal to this; that means, if the sawtooth crosses the controller output then we will take this as the signal. So, what is the name of this signal; that means, the controller output; that means if you go inside sorry. So, this is the block yeah you see at the clock edge or the fpwm if the clock edge is high it will reset the counter. So, this is the counter and this will generate something called f con whenever it crosses the trigger.

(Refer Slide Time: 29:51)



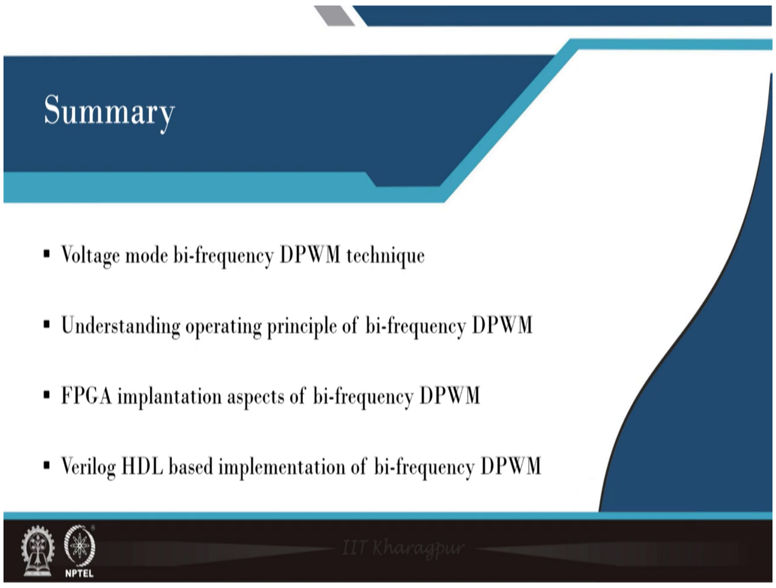
Deadtime_Circuit Module

```
always@(posedge f_pwm_delay or posedge f_con) begin
if (f_con)
Q_pwm<=0;
else
Q_pwm<=1;
end
always@(posedge clk)begin
        shift[0]<=Q_pwm;        shift[1]<=shift[0];
        shift[2]<=shift[1];     shift[3]<=shift[2];
        shift[4]<=shift[3];     pwm_delay<=shift[4];
end
assign Q_H=Q_pwm & pwm_delay;
assign Q_L=~(Q_pwm | pwm_delay);
endmodule
```

Now, we have a set-reset pulse. So, a reset is coming from f con and the set is coming from fpwm delay and this is the output Q we are we are calling it as you know if you go to that this is your RS flip flop, RS flip flop, and; that means, it 1 input is f pwm delay this is our set and what is the reset? It is the f con.

And this is Q and we call it a Q pwm this Q pwm is going to the dead time circuit and this is generating QH and QL and we have already discussed this dead time circuit generation in lecture number I think 60 I think we have discussed in 69, 70 of this course detail how to implement in Verilog with simulation. So, all these things we have discussed.

(Refer Slide Time: 30:50)



Summary

- Voltage mode bi-frequency DPWM technique
- Understanding operating principle of bi-frequency DPWM
- FPGA implantation aspects of bi-frequency DPWM
- Verilog HDL based implementation of bi-frequency DPWM

IIT Kharagpur
NPTEL

So, in summary, we have discussed the voltage mode bi-frequency DPWM technique we understood the basic operation of bi-frequency DPWM. And we have discussed the FPGA implementation aspect and we have discussed also Verilog HDL-based implementation of bi-frequency DPWM. So, in the next lecture, we are going to show an experimental case study using bi-frequency DPWM and we want to compare performance as well as spectral power spectral density with and without bi-frequency DPWM that is it for today.

Thank you very much.