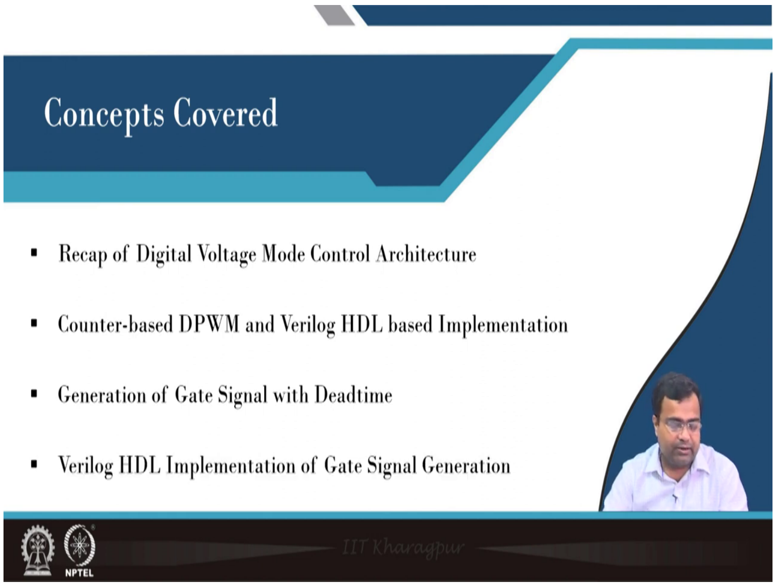
Digital Control in Switched Mode Power Converters and FPGA-based Prototyping

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

Module - 07
Introduction to Verilog and Simulation Using Xilinx Webpack
Lecture - 69
Counter-based DPWM with Deadtime and Verilog HDL Programming

Welcome. In this lecture, we are going to talk about Counter-based DPWM and Deadtime and Verilog HDL Programming.

(Refer Slide Time: 00:32)

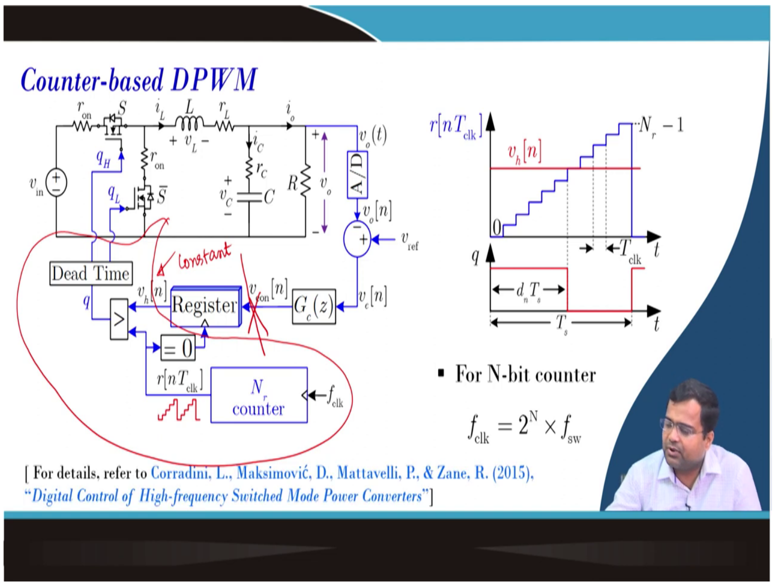


Concepts Covered

- Recap of Digital Voltage Mode Control Architecture
- Counter-based DPWM and Verilog HDL based Implementation
- Generation of Gate Signal with Deadtime
- Verilog HDL Implementation of Gate Signal Generation

So, in this lecture, we are going to talk recapitulate our digital voltage mode control architecture then counter base DPWM and Verilog HDL-based implementation then generates gate signal and also the dead time, and finally, how to implement it in Verilog HDL.

(Refer Slide Time: 00:51)



Counter-based DPWM

- For N-bit counter

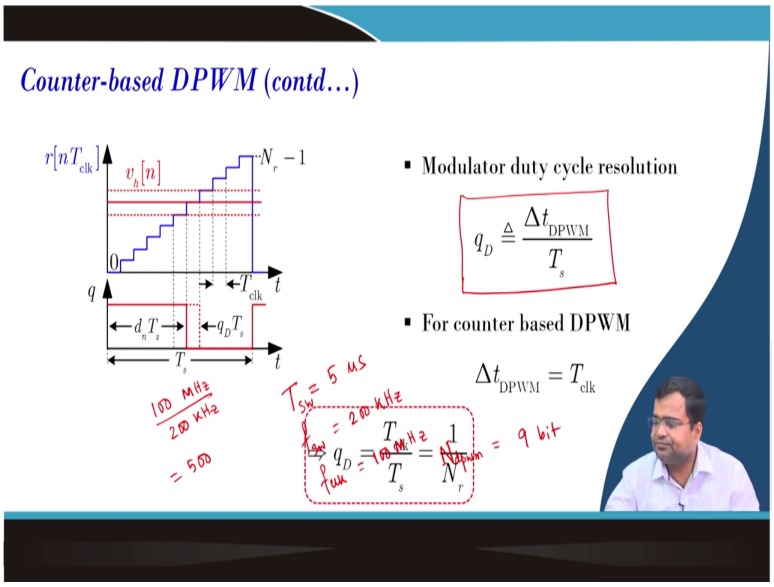
$$f_{clk} = 2^N \times f_{sw}$$

[For details, refer to Corradini, L., Maksimović, D., Mattavelli, P., & Zane, R. (2015), "*Digital Control of High-frequency Switched Mode Power Converters*"]

So, now we are going to discuss the counter-based DPWM and we have discussed this architecture I think week 2 where we are talking about a different type of DPWM architecture, and in this particular lecture, we are not going to close the outer loop. We only know if you discard this path; that means if you discard this path we are setting a constant value here because our objective is to get generate the duty ratio which is with a fixed frequency clock.

Because we have to we want to implement this particular part which will generate this duty ratio and we are calling a DPWM and dead time circuit we know about the resolution effect; which means, what is the minimum resolution and we have discussed how to overcome that you know if you want because if you want a resolution of 9-bit 10-bit DPWM or 9 DPWM then if the switching frequency is 1 megahertz and if you need a 10-bit DPWM you need 1 mega 1 gigahertz clock which is not acceptable. So, there is an alternative architecture that we have discussed.

(Refer Slide Time: 02:04)



Counter-based DPWM (contd…)

- Modulator duty cycle resolution

$$\boxed{q_D \triangleq \frac{\Delta t_{\text{DPWM}}}{T_s}}$$

- For counter based DPWM

$$\Delta t_{\text{DPWM}} = T_{\text{clk}}$$

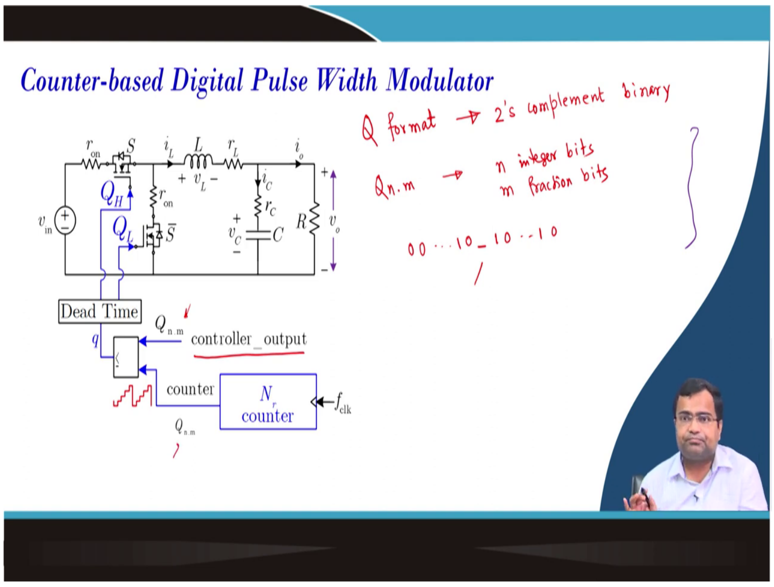
$$\Rightarrow q_D = \frac{T_{\text{clk}}}{T_s} = \frac{1}{N_r}$$

$r[nT_{\text{clk}}]$
$v_h[n]$
$N_r - 1$
0
q
T_{clk}
t
$d_n T_s$
$q_D T_s$
T_s
t
$\frac{100 \text{ MHz}}{200 \text{ kHz}}$
$= 500$
$T_{sw} = 5 \ \mu s$
$f_{sw} = 200 \text{ kHz}$
$f_{clk} = 100 \text{ MHz}$
$N_{dpwm} = 9 \text{ bit}$

So, now, the similar context is because we are just recapitulating what we have discussed so; that means, to get certain DPWN, we need to play with the architecture of DPWM. In this case, we are only taking out the DPWM counter base and we have our desired switching frequency for this stitching key we are setting T s to be 5 microseconds; that means, you know T s or you can say T sw switching frequency is 200 kilohertz and we have available controller clock which is 100 megahertz.

So, we need an account for the DPWM I can say dpwm I mean if you take the number; that means, the ratio will be 100 megahertz by 200 kilohertz and this will be equal to 500. So, we need a DPWM of 9 bits which can count from 0 to 499 and we are familiar with resolution ok.

(Refer Slide Time: 03:11)

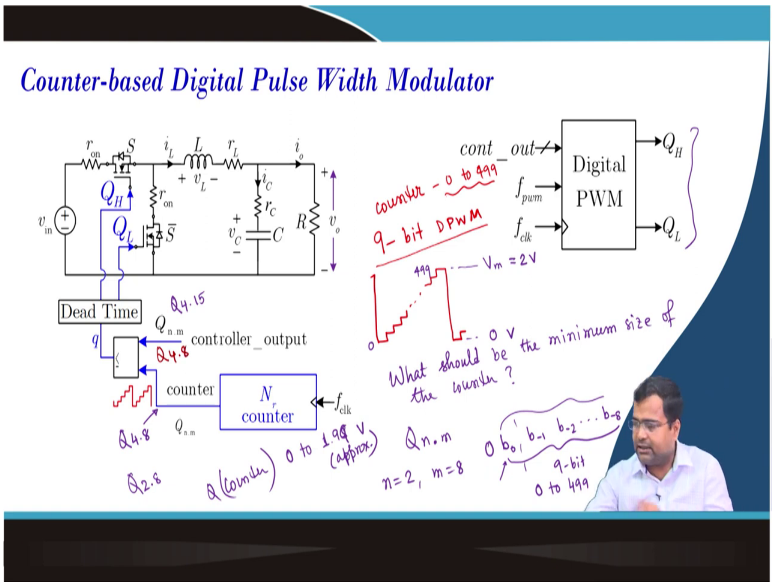


So, now in this particular architecture as I discussed that I have discussed that we want a fixed control output; that means, we are not closing the loop and we are talking about the sin Q format and we have discussed that Q format concept in the previous lecture this is to represent 2's complement binary number.

And when you say Q n dot m; that means, we want to get n as if like an integer bit we are talking about n integer bit and m fraction bit. There is no physical fraction because is a notional concept; that means if you have 0 0 sum 1 0 underscore sum 1 0 1. So, like this. So, this is a notional concept that Verilog ignores, but it is just to represent the n number of bits on the left side and the m number of bits on the right side that we have discussed.

And we have also discussed comparing two numbers because in the digital circuit comparison of two number means, it is simply subtraction and we can simply check the MSB that is enough. So, for any subtraction, we need to ensure the format of this control output and the counter must match ok.

(Refer Slide Time: 05:05)



Now, the next question. So, this block we have to design now what is the next question? Our next question is that we want to know if it is a 9-bit DPWM ok. And 9-bit DPWM is a requirement. So, the counter will count from 0 to because this is the counter will count from. So, the counter will vary from 0 to 499. 0 to 499 but this is a binary number which is a real-life voltage equivalent that is the most important because we are talking about a saw tooth waveform. So, like this ok.

Now, this for the counter is just a number, but in real life, this corresponds to some voltage because if you pass this signal from d to a converter it should appear like analog voltage. So, you want this V m to be let us say 2 volts and this to be 0 volts then the counter is counting; that means, for the counter, it is 0 and it is 499. So, this is from the counterpoint of view. So, what should be the Q format? The question is what should be should be the minimum size of the counter?

So, first of all, we need a positive 2 volt and the Q format we know because it has to take the 9-bit. So, we need. So, for the dot side, what will be m? That means we are talking about n dot m. First of all, we only need a positive value; that means, we must have a 0 MSB to represent the positive number so; that means, it should be 0 on the MSB side then the rest of the bit can be utilized to accommodate the number.

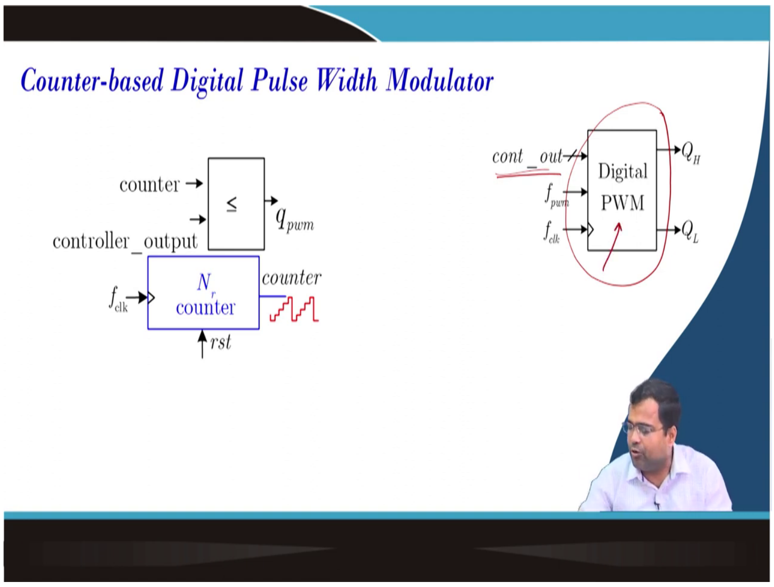
So, the rest of the bit let us say it can be there will be; that means, if it is 9 bit the other bit will be b 0 let us say then b minus 1 b minus 2 like that b minus 1. So, since this, if you take this total as 9 bit let us say the total is 9 bit. So, this 9 bit if you vary it's just a number it's like an unsigned number. So, this can vary from 0 to 499 and when it is set to 499 it is approximately equal to 1 volt because the integer bit can be 1 maximum and all close to 1.

That means if we set n equal to 2 and m equal to 8 then this counter and if you make the 0 leading with 0 and use this 9-bit DPWM then it can take from; that means, your counter if you say the quantized voltage level can vary from 0 to nearly 1.95 or something it can be approximately volt. So, it is approximately; that means, we want 0 to close to 2 volt. So, for that our minimum bit size should be Q 2.8.

But remember we need to match this bit size so, but this bit side we saw if we said because we will be discussing the control output this will be 4 bit let us say it is Q 4 dot 15 then how do you match? So, to match this counter must have 4 in the integer bit, and where we can simply make two additional MSB by sign extension. So, this should be a minimum of 4.8 to be comparable.

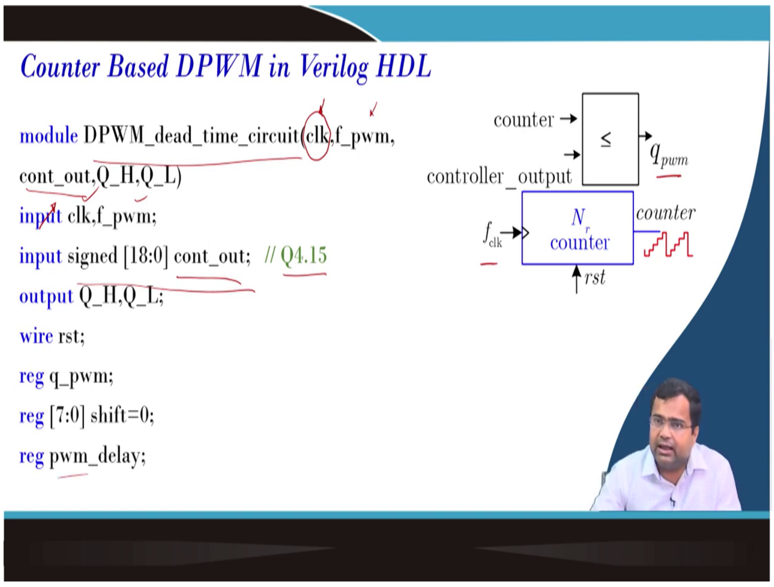
So, you need a resizing and this should you can discard the bit so, that you can make this bit be Q 4 dot 8 where the 7 extra bits are disconnected and that we are going to discuss.

(Refer Slide Time: 09:51)


Counter-based Digital Pulse Width Modulator

So, this is what we are discussing that counter control output where we are making the module.

(Refer Slide Time: 09:55)



Counter Based DPWM in Verilog HDL

```
module DPWM_dead_time_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;
reg [7:0] shift=0;
reg pwm_delay;
```

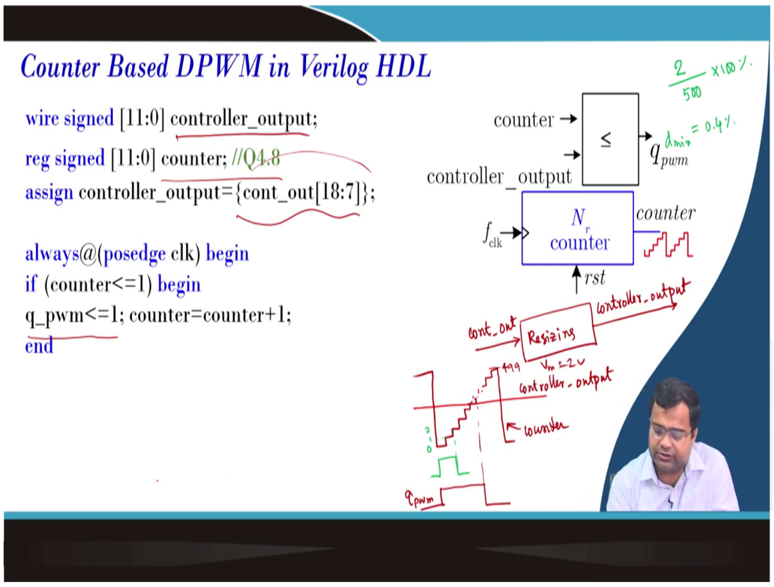
So, it is a DPWM module main module where we are talking about a high-frequency clock this clock PWM clock at this point is not needed because you are not synchronizing.

But when you go to the closed-loop control it requires, but you need to sync with this PWM next to the counter output. So, control output because in the main block if you go back to the main block as if this block is a submodule of the main voltage mode control module that will be discussed in next week’s lectures where this submodule only takes control output and it will generate the high low gate signal.

So, in this case, the control output when you go inside. So, this controller output we have assumed because it is a part of the digital control and that will discuss it is given in the 4 dot 15 format, it is there is no choice because it’s coming from the control output and the switching frequency clock we may synchronize, but in this case, we are just giving a fixed number and we want to ensure that switching frequency overall switching frequency will be 200 kilohertz.

So, this clock is very very important that is the input QPWM is input, but we are we may or may not use it and the controller output is a vector data is a significant number input and then Q H K L is the scalar output ok. So, there is an internal variable. So, QPWM is the output of this block which will be used to generate the dead time QH and QL. So, this will be the base duty ratio clock signal, and that will be used to generate a known dead time circuit and actual high side low side gate signal. And we need some PWM delay because we need to pass for delayed work that we will discuss.

(Refer Slide Time: 11:52)



Counter Based DPWM in Verilog HDL

```
wire signed [11:0] controller_output;
reg signed [11:0] counter; //Q4.8
assign controller_output={cont_out[18:7]};

always@(posedge clk) begin
if (counter<=1) begin
q_pwm<=1; counter=counter+1;
end
```

So, the control output we want to match; means, we are talking about resizing. Now initially if you go back you know this is the initial block input which is a submodule block inside this block if you go inside this block we are assigning a variable which is controller output. So, the con underscores out which was the output of the main block we are resizing to get that we are calling as a controller output that is the reason wire variable.

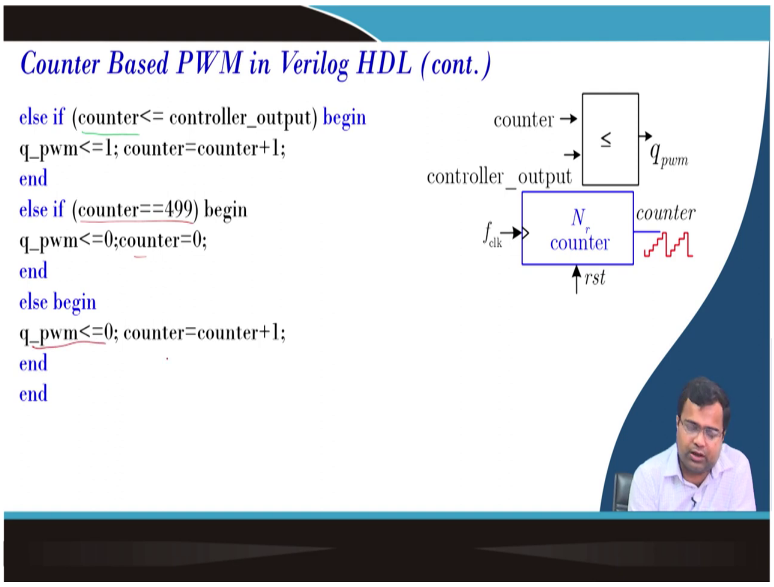
And we are taking a counter and we discuss the counter has to be 4.8 because the 4-bit is the requirement coming from the cont output that MSB and 8-bit is mandatory for the counter. After all, you want 0 to 2 volt ok and we are resizing it as a concatenation operation. Now at this point, we can also define a value because we are talking about just a fixed number ok we can simply take the value because, in the next lecture, we are going to simulate this.

So, always you know the clock edge; that means, at the positive edge of the clock. So, the counter will start incrementing and we are taking for the counter from 0 and 1 value, we are taking q PWM 1; that means, the for 2; that means, we are talking about the counter like that. So, for the two values, if we consider this as 0, this is 1, and this is 2.

So, up to 1, the action will be taken in the next cycle; that means, we are setting a minimum duty ratio as per this command which is 2 is the number minimum and the total number is

500 into 100 that percent, and if you do that you will get 0.4 percent duty ratio that is my d minimum. We may set the minimum value or we may increase the minimum value does not matter but we have to make sure that you know whatever we want whether we want a minimum duty ratio or linearity can be 0. So, the counter is incrementing.

(Refer Slide Time: 14:23)



Counter Based PWM in Verilog HDL (cont.)

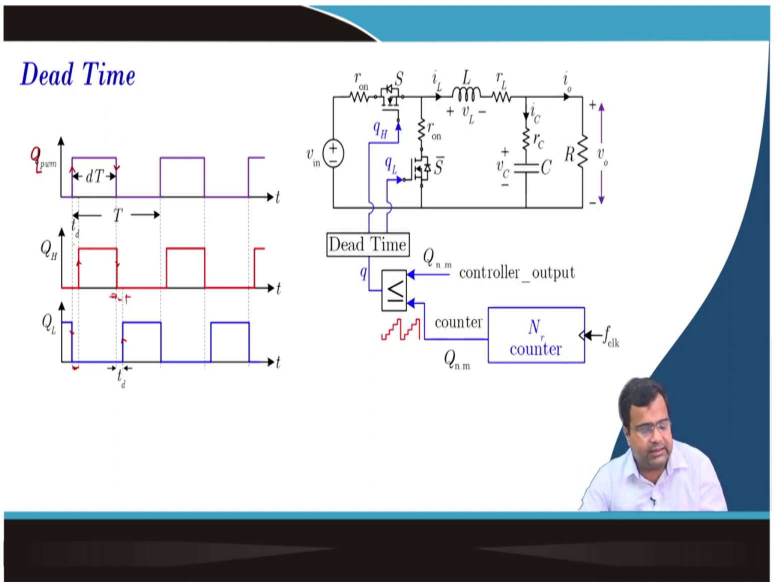
```
else if (counter<= controller_output) begin
q_pwm<=1; counter=counter+1;
end
else if (counter==499) begin
q_pwm<=0;counter=0;
end
else begin
q_pwm<=0; counter=counter+1;
end
end
```

Then next the counter as long as the counter output is lower than the control output because the next will be we have a controller and this is the control output and if we take this signal ok then. So, this we have to take. So, this will be your q this is your q p w m.

So, whenever the controller output; that means, this is my controller output as long as the and this is my counter count output as well as the count output is below the control output the PWM signal will be high when it crosses it will be low and this is exactly what is mean for. So, when it is up to this point it is high when it is equal to 499 which is the reset condition. So, this is 499 and this corresponds to 2 volt.

Then roughly 2 volt approximately I would say approximately 2 volt then it will be set to 0 counter will be set and else; that means, when it goes above the QPWM output is 0 and the counter will upon increment.

(Refer Slide Time: 15:46)

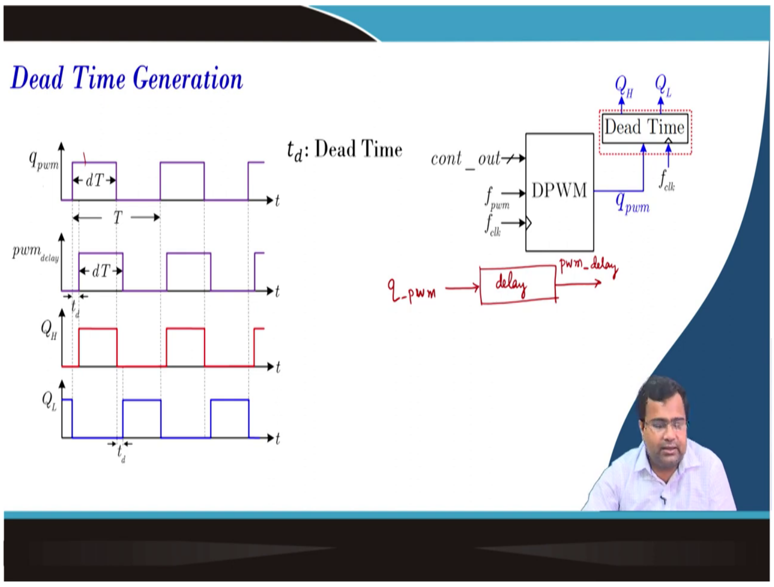


So, this is the Verilog of code and the dead type. Now we got this q PWM which is this q PWM which has a duty ratio dT, but we cannot send in the real circuit because there has to be a dead time, and we must avoid any overlap period we have to provide an overlap period of the rise time and the fault time of the high side and low side switch that is why we have to provide the dead time and this dead time if you look at this whenever the controller the PWM output goes high then you have to first turn off the low side switch which was on because the switch was off now we have to turn on the switch.

So, that is why we have to first turn off the low side switch its a break and make then you have to provide a delay and then you turn on the high side switch and when you get the falling edge of the PWM signal then you first turn off the high side switch then provide a delay then you turn on the low side switch. So, this is called you need a dead time circuit, so this time is called dead time.

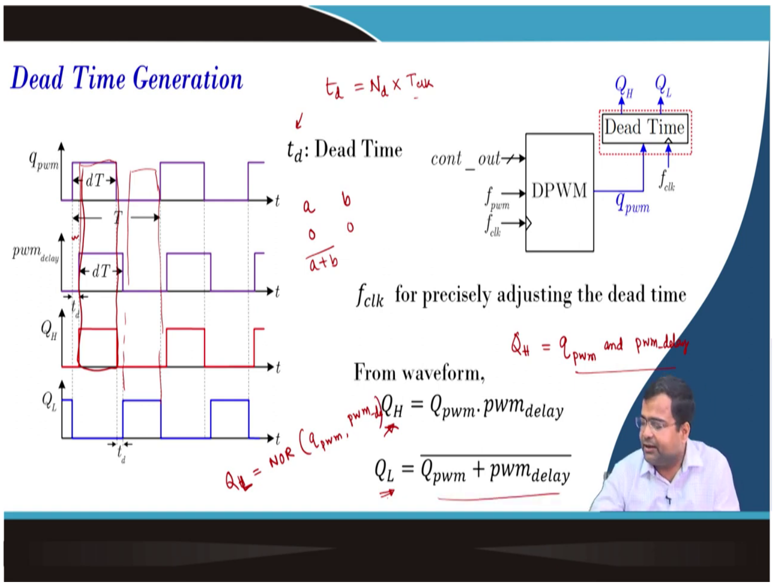
So, during this time you are avoiding you know turning on and off the simultaneous two switches. So, that means, but this time should not be too large because it will otherwise actually turn on the body diode and that may lead to reverse recovery and other losses. So, you have to be very careful and if it is too small there can be a condition both the switch can be on and that will be high index current ok.

(Refer Slide Time: 17:18)



So, how to generate this dead time circuit? If you go we are delaying this PWM signal; that means if i pass this q PWM through a delay block. So, delay block and this delay should be customizable then we are talking about PWM delay. So, that signal is named and you can see it is delayed. Now if you look at this waveform how do you synthesize it? So, QH and QL how it is generated? td that is the dead time ok.

(Refer Slide Time: 17:52)



Dead Time Generation
$t_d = N_d \times T_{clk}$
t_d: Dead Time
a b
0 0
a+b
f_{clk} for precisely adjusting the dead time
$Q_H = q_{pwm}$ and pwm_delay
From waveform,
$Q_H = Q_{pwm} \cdot pwm_{delay}$
$Q_L = \overline{Q_{pwm} + pwm_{delay}}$
Q_L = NOR (q_{pwm}, pwm_d)
cont_out
DPWM
f_{pwm}
f_{clk}
q_{pwm}
Dead Time

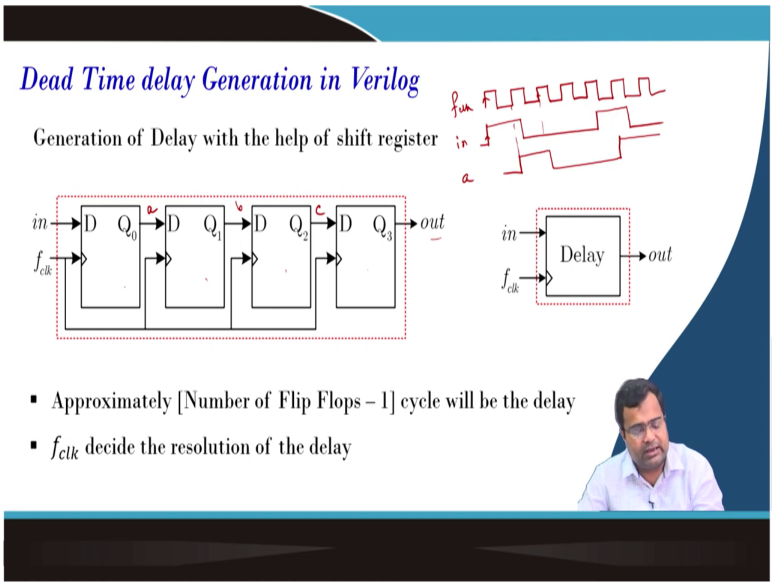
So, let us say you know. So, for clock frequency, we can use the clock frequencies which means the resolution of this td this td will be some N number of delay into the T clock time period of the high-frequency clock. So, it's an integer multiple because you cannot get a better resolution than the T clock it is defined by the T clock. So, if your PWM switching I mean the controller's clock is slower then you cannot have precise control over the dead time.

So, you need to have a high-resolution clock. So, now, this dead time from the waveform what is QH you can see? It is; that means, I am talking about the QH high side gate signal we will be q PWM and PWM delay; that means, q PWM. So, you take QH it is the end between these two signals because if you take this region and to this region, they are simply ending operation when both are on ok. So, this is simply what about QL?

So, QL if you take; that means, this period and this period when both the switch should be off; that means when both are off then only; that means if let us say we have a and b if both are off then only it will be high then what will be this? This is to be OR a plus b and NOT of that.

So, it's frequently NOR. So, NOR of this. So, QL is the NOR. So, QH is the end of two signals and QL is the; that means, QH the is the AND gate; that means, it is AND that we have written and QL is the NOR operation of what q PWM comma PWM delay and this is exactly is written here.

(Refer Slide Time: 20:17)



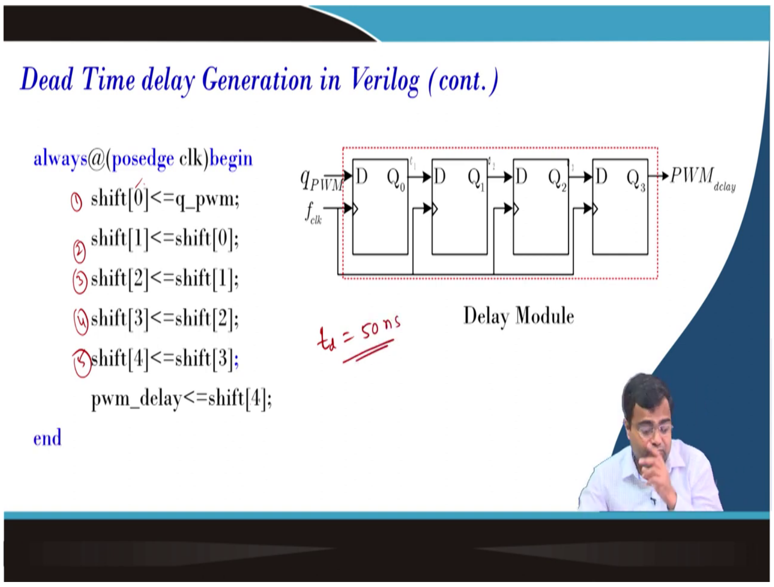
Dead Time delay Generation in Verilog

Generation of Delay with the help of shift register

- Approximately [Number of Flip Flops – 1] cycle will be the delay
- f_{clk} decide the resolution of the delay

That means if you write this Verilog code delay generation. So, the delay has to use a shift register; that means, there is a clock whenever this edge clock comes this signal is propagated to this site, and whatever was there will propagate this way. So, suppose the clock edge comes.

(Refer Slide Time: 20:38)

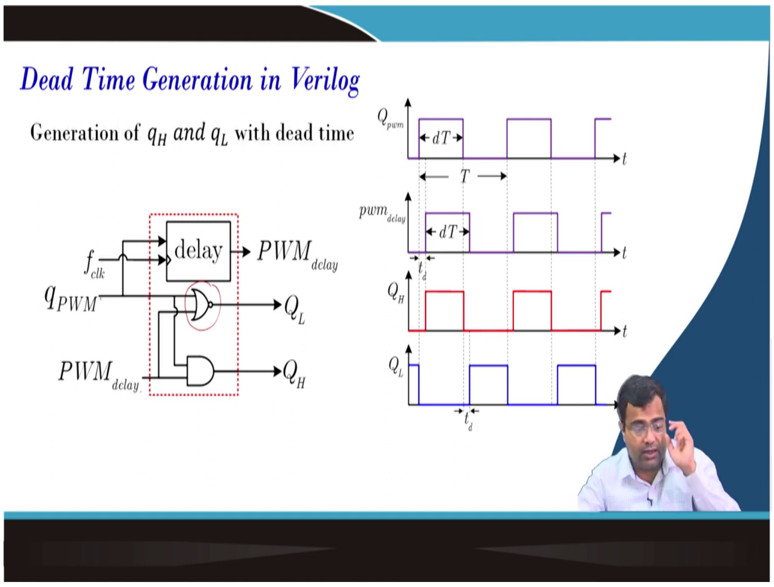


Dead Time delay Generation in Verilog (cont.)

```
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
```

So, if you take several if you take the waveform.

(Refer Slide Time: 20:40)



Dead Time Generation in Verilog

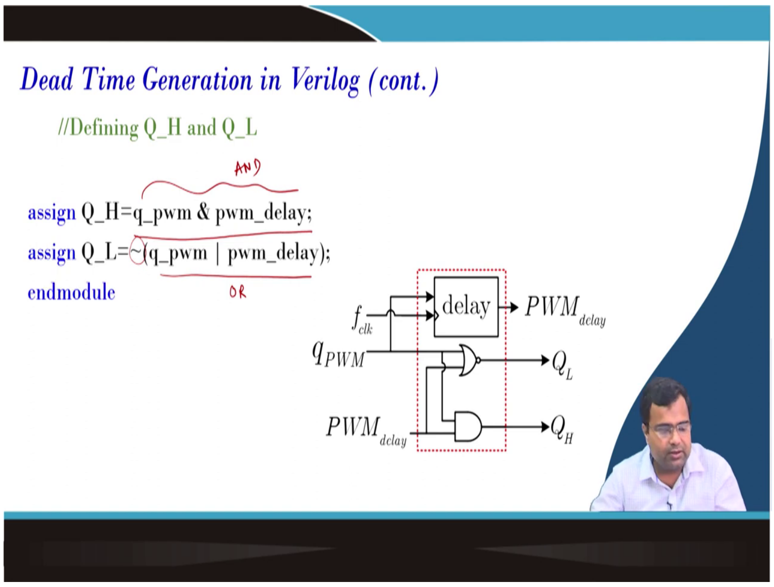
Generation of q_H and q_L with dead time

You know let us say we take this waveform. So, suppose we have this clock frequency ok. Suppose we have this clock frequency. So, this is the f clock suppose your input clock is like that let us say it was high like that then what will be this signal let us say it is a b c what will be a? a will be delayed by 1 a; that means, this edge comes. So, this will be our this clock it will be delayed by 1 unit what will b? It will be delayed by another unit. So, that way the out will be delayed by 1, 2, 3, 4 four unit delay you need 4 shift registers ok.

So, several flip flops or d flip flops here, in this case, will be the total delay will be several flip flops minus 1 because if you take. So, one will be delayed the next will be delayed. So, we are using how many from shift 0 to shift 4; that means, 5 unit delay we are using 1 2. So, there are there will be 1 shift register 2 3 4 5. So, the delay will be 5 shift register delay.

So; that means, we are taking the tau d to be 50 nanosecond delay ok and this can be; that means, we have discussed that if you have this q P W M then we just generate this delay, and what is QH? We know that QH is the AND operation of the q PWM and the delay signal and the QL is the NOR operation of assign. So, you can simply write a code to do that

(Refer Slide Time: 22:54)



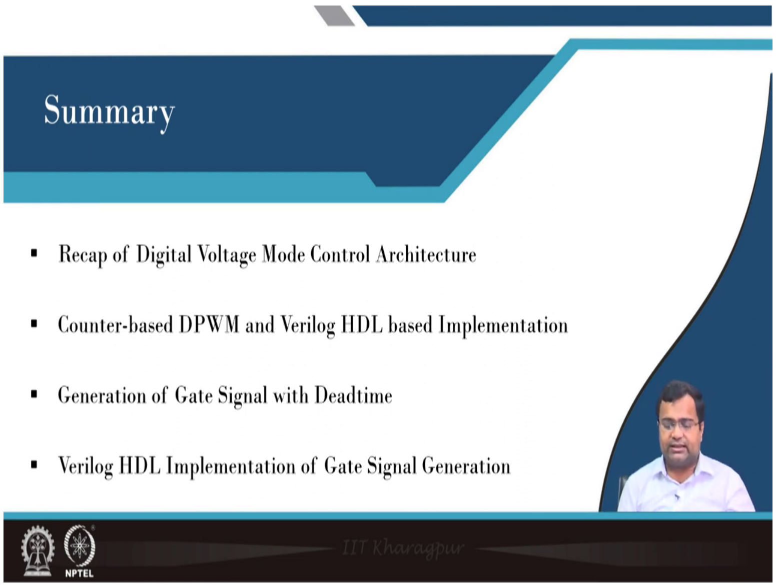
Dead Time Generation in Verilog (cont.)

```
    //Defining Q_H and Q_L

assign Q_H=q_pwm & pwm_delay;
assign Q_L=~(q_pwm | pwm_delay);
endmodule
```

So, in one line you can write; that means, assign Q H this is in data flow modeling where which represents the AND gate. And this is a not of OR operation. So, this is the OR operation and this is the AND operation ok. So, AND operation or operation and this is the invert; that means, you are complementing this.

(Refer Slide Time: 23:25)



Summary

- Recap of Digital Voltage Mode Control Architecture
- Counter-based DPWM and Verilog HDL based Implementation
- Generation of Gate Signal with Deadtime
- Verilog HDL Implementation of Gate Signal Generation

NPTEL
IIT Kharagpur

So, this is now the end of this module. So, in summary, we have discussed we have recapitulated the digital voltage mode control talked about counter base DPWM Verilog HDL implementation discussed the generation of gate signal with dead time and also discussed what is the Verilog HDL implementation of gate signal generation.

So, in the next lecture, we are going to show the live simulation of you know the part of that we want to check the behavioral simulation of this DPWM and the gate signal that is it for today.

Thank you very much.