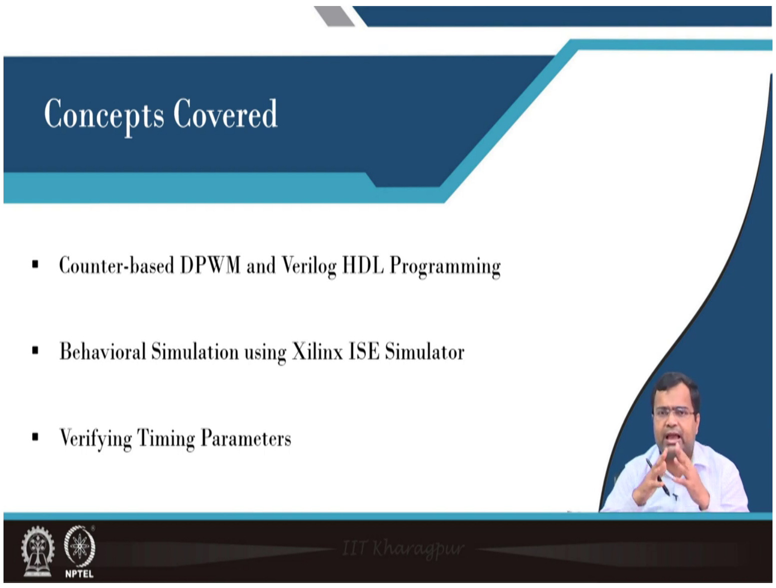
Digital Control in Switched Mode Power Converters and FPGA-based Prototyping

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

Module - 07
Introduction to Verilog and Simulation Using Xilinx Webpack
Lecture - 70
Simulating Counter-based DPWM with Deadtime using Xilinx ISE Simulator

Welcome to this lecture we are going to discuss the behavioural simulation of counter base DPWM with dead time using the Xilinx ISE simulator.

(Refer Slide Time: 00:38)



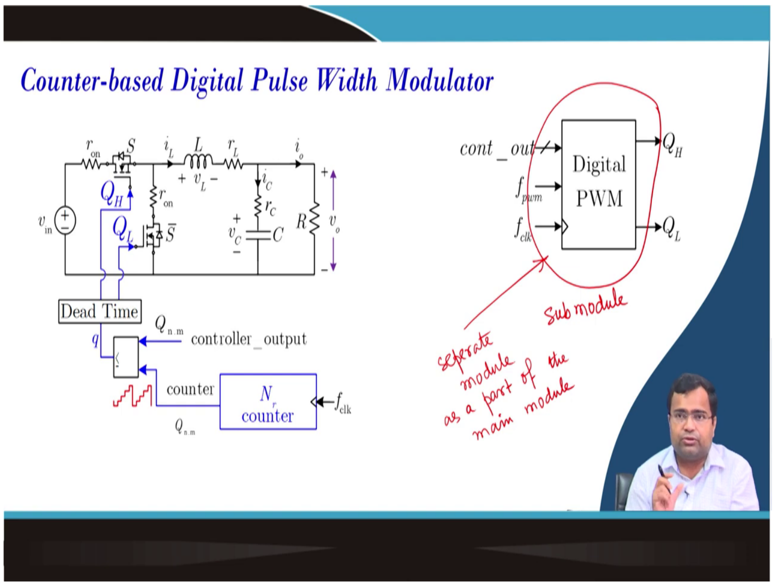
Concepts Covered

- Counter-based DPWM and Verilog HDL Programming
- Behavioral Simulation using Xilinx ISE Simulator
- Verifying Timing Parameters

IIT Kharagpur
NPTEL

So, this is a continuation of the previous lecture where we discussed in detail you know the counter-based DPWM and the Verilog HDL programming. Then we are now going to do a behavioural simulation and verify the timing parameter. In fact, in lecture number 66 we introduced a live simulation of how to do Verilog simulation sorry Xilinx ISE simulator, how to use Xilinx ISE simulator, and how to check the timing parameter ok. So, those things we have discussed for a 4-bit ripple carry adder; but now we are going to check for this counter-based DPWM with dead time.

(Refer Slide Time: 01:11)

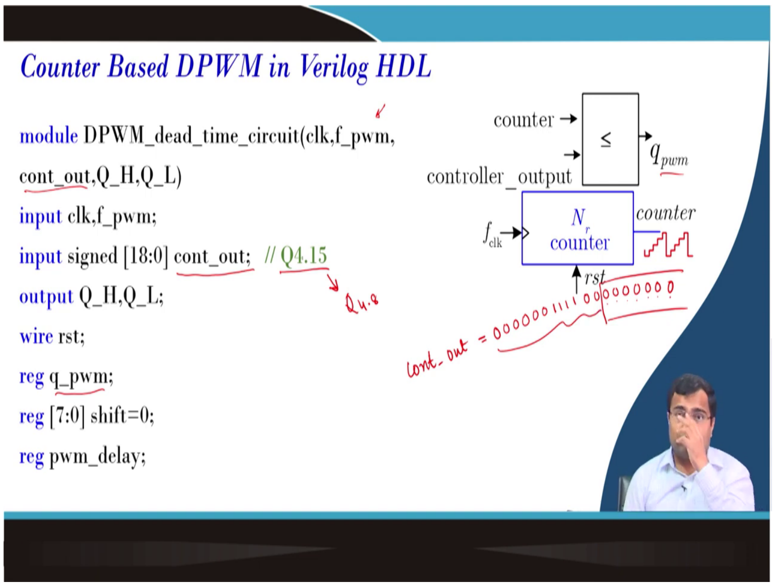


Counter-based Digital Pulse Width Modulator

So, here again, we are not going to spend time because these things have been discussed in the previous lecture in detail. So, we have to make this block; so, our objective is to make this block and this block is a submodule basically or basically, I will say this is a module in Verilog it is a separate module when it is a submodule it is a separate module as a part of the main module.

And the main module and all these things will discuss in the subsequent lecture in next week's lecture. But today we want to check how the DPWM work and we have over the timing there.

(Refer Slide Time: 02:09)



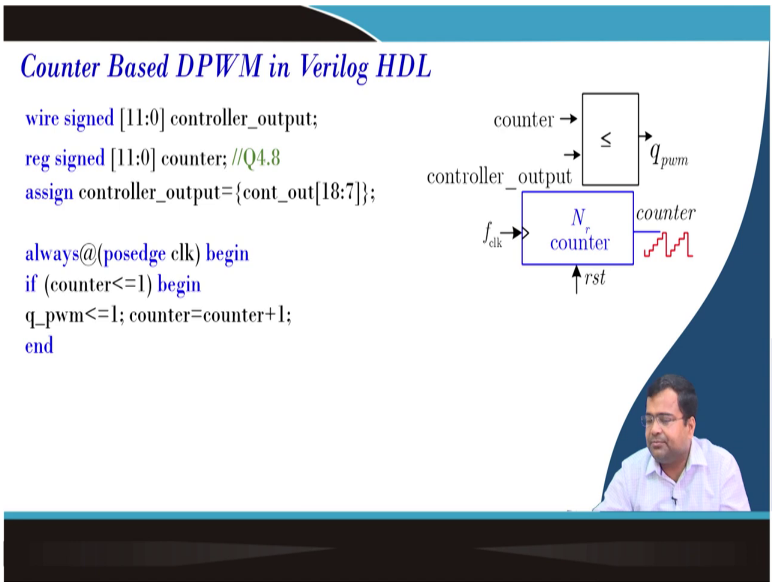
Counter Based DPWM in Verilog HDL

```
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, again circuit we have explained in the previous lecture Verilog code I am not going to that. The only thing is the output of the control output that will define our own because we are not using any controller closed loop. So, it is user-defined for the time being to check. Again, this PWM switching clock we are not utilizing here, because we are not synchronizing.

Because it is just to check whether we can generate the duty ratio and use that PWM gate signal Q PWM that we will have defined. We have we are generating a Q PWM signal and out after getting Q PWM how do the Q H and Q L look like, that is the main objective in this particular lecture ok

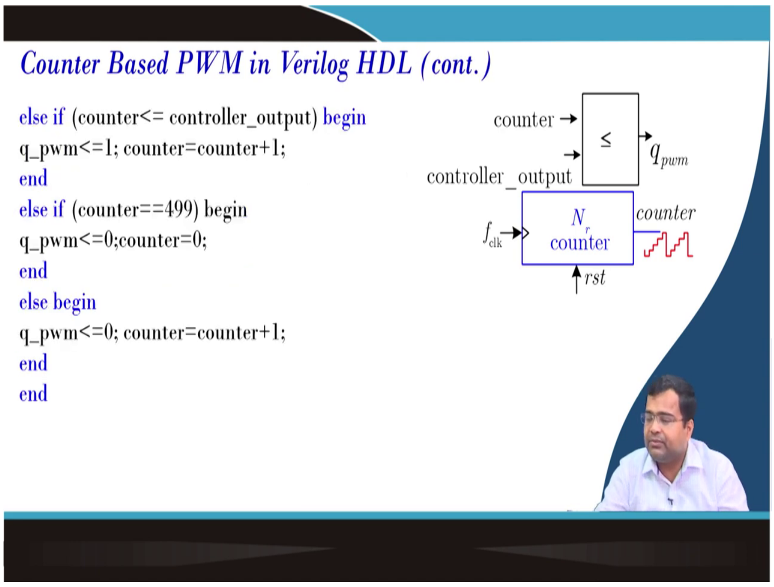
(Refer Slide Time: 03:02)



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
```

(Refer Slide Time: 03:05)

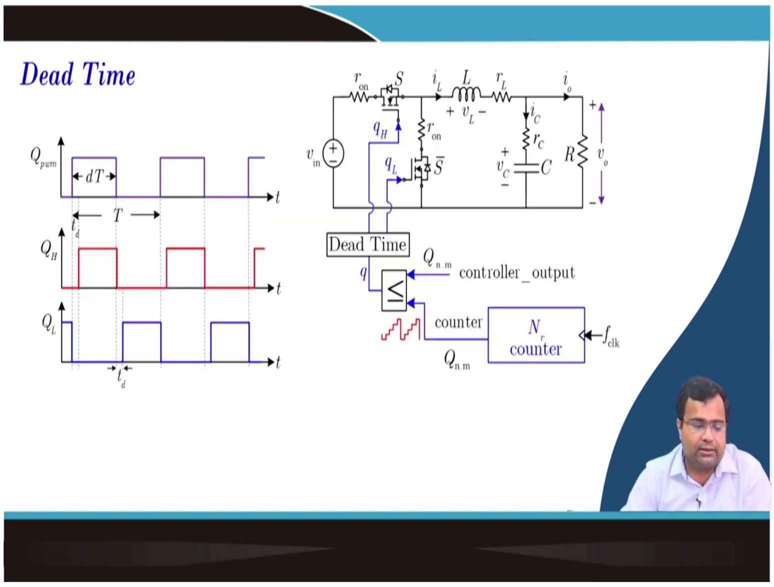


Counter Based PWM in Verilog HDL (cont.)

```
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
```

So, there are delay units we have discussed all this detail counter DPWM and now for this case study and the dead time circuit also we have discussed it.

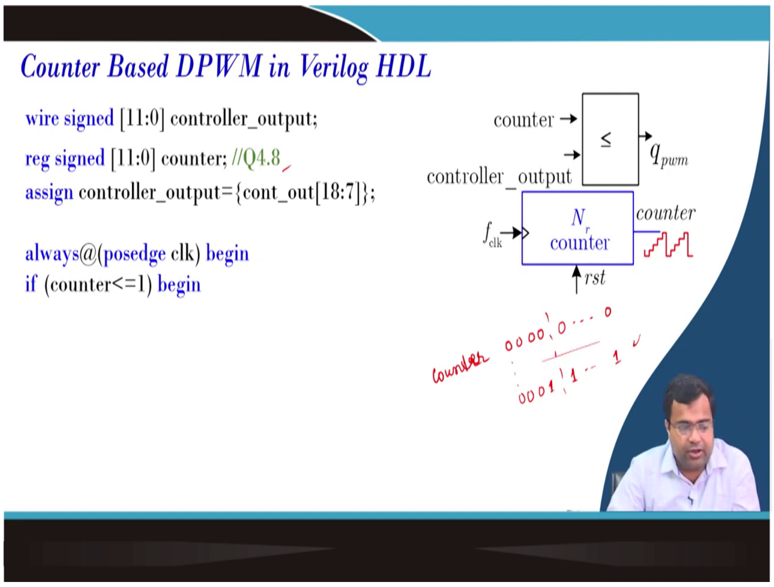
(Refer Slide Time: 03:09)



Dead Time

So, what we have discussed now in this particular lecture, let us say the counter we are using in which format if you go back to the counter. So, it is counter is used 4 dot 8 formats, but counter varies from what? We want the counter to vary from 0 to 499; that means, it will go to the first bit of the integer and the remaining 3-bit like 0.

(Refer Slide Time: 03:45)



Counter Based DPWM in Verilog HDL

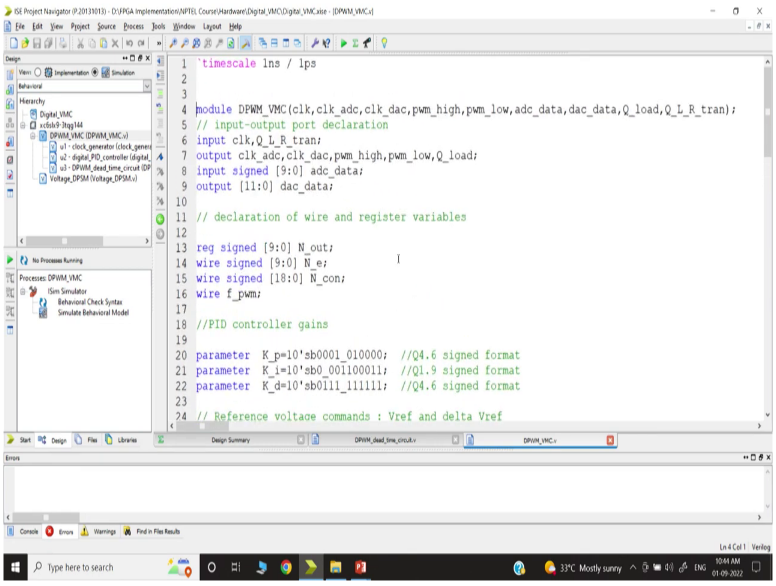
```
wire signed [11:0] controller_output;
reg signed [11:0] counter; //Q4.8
assign controller_output={cont_out[18:7]};

always@(posedge clk) begin
if (counter<=1) begin
```

That means the counter will vary from 00000 dots dot dot 0 and it can reach up to 00011 may be a few internal bits we have to check which corresponds to 499. But it cannot go beyond that; so, whatever is the corresponding number for 499? So, we have to choose this output which should be in between; that means because the input to this block if you go to this particular block is a cont output.

So, the cont output originally is a 19-bit number 4 dot 15; that means, we are considering this cont output for testing purposes in the binary number what are you going to take it is in 4 dot 19? So, we are going to take the first 6 digits just for sake of 6 this is 0, then the next 4 digits 1; so, 10 digits then we still have 9 another 9 digits all 0 5 6 7 8 9.

So, this is just a number we are taking may be less and this will give you what will be the duty ratio. And we have the resizing; that means when you resize we are discarding because we have to convert this into Q 4 dot 8; that means, we are discarding 7-bit. So, 1 2 3 4 5 6 7; so, anyway these bits are discarded; so, we are taking eventually when it will be resized this data will look like this ok. So; that means, we are taking we have discussed the cont output and how much we are going to take: so, we will take this and let us go to our simulation.

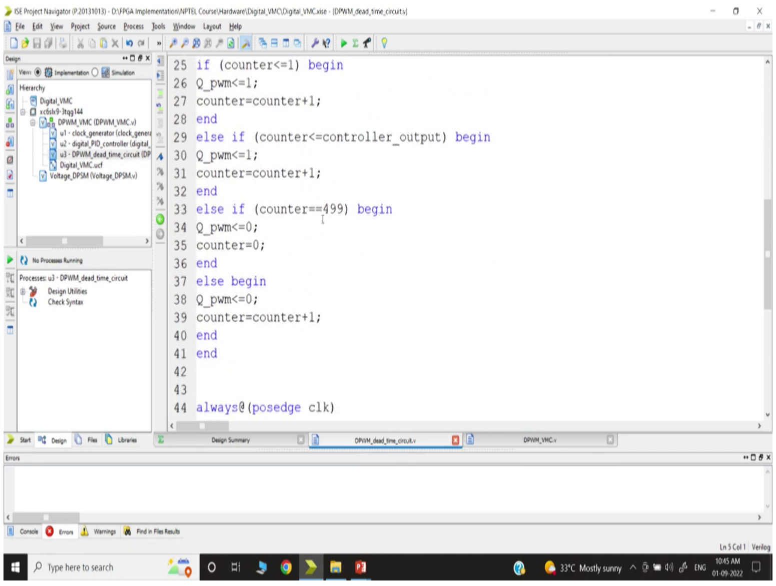
(Refer Slide Time: 05:41)



```
`timescale 1ns / 1ps


module DPWM_VMC(clk,clk_adc,clk_dac,pwm_high,pwm_low,adc_data,dac_data,Q_load,Q_L_R_tran);
// input-output port declaration
input clk,Q_L_R_tran;
output clk_adc,clk_dac,pwm_high,pwm_low,Q_load;
input signed [9:0] adc_data;
output [11:0] dac_data;

// 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_010000;  //Q4.6 signed format
parameter  K_i=10'sb0_001100011;  //Q1.9 signed format
parameter  K_d=10'sb0111_111111;  //Q4.6 signed format

// Reference voltage commands : Vref and delta Vref
```

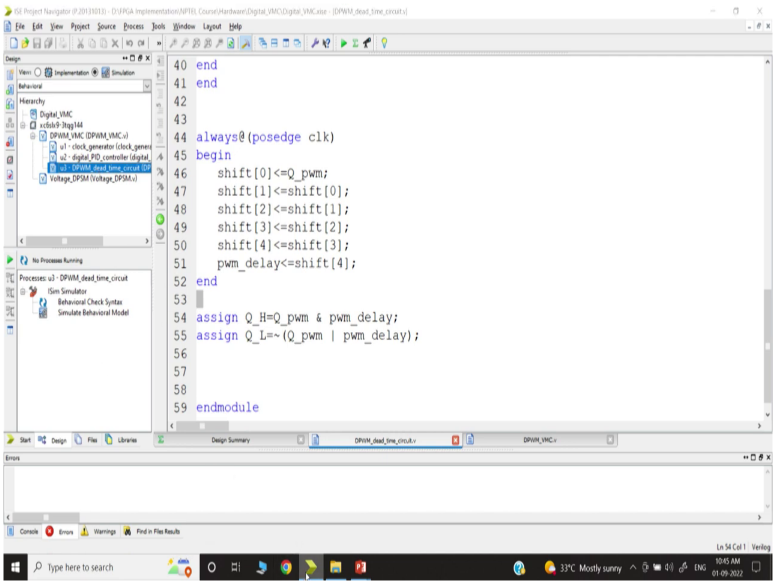
So, here I am showing just part of this lecture because this is the overall voltage mode control which we are going to discuss in the subsequent lecture ok. And here we are going to say you know we are going to program and we want to implement in the hardware using FPGA.

(Refer Slide Time: 06:10)



```
if (counter<=1) begin
Q_pwm<=1;
counter=counter+1;
end
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


always@(posedge clk)
```

(Refer Slide Time: 06:16)



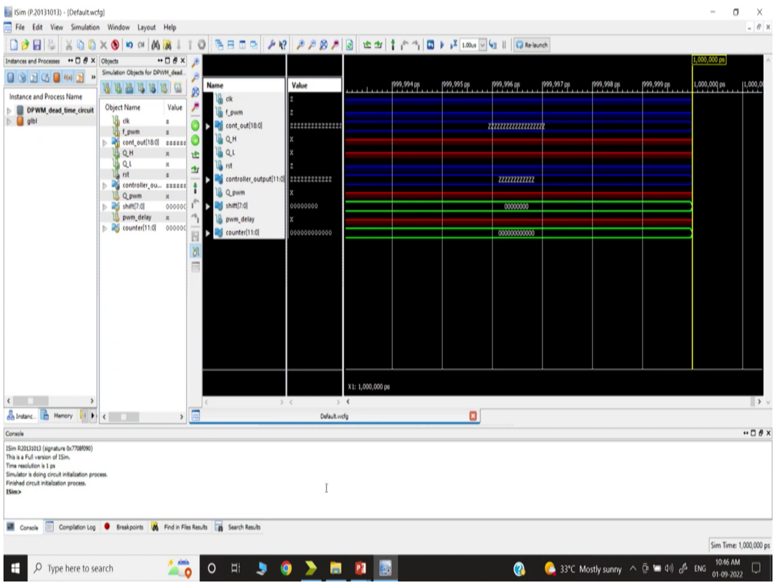
```
end
end


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
```

ISE Project Navigator (P.20131013) - D:\FPGA Implementation\NPTEL Course\Hardware\Digital_VMC\Digital_VMC.xise - [DPWM_dead_time_circuit.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
Digital_VMC
xc6slx9-3tqg144
DPWM_VMC (DPWM_VMC.v)
u1 - clock_generator (clock_genera
u2 - digital_PID_controller (digital_
u3 - DPWM_dead_time_circuit (DP
Voltage_DPSM (Voltage_DPSM.v)
No Processes Running
Processes: u3 - DPWM_dead_time_circuit
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start Design Files Libraries
Design Summary
DPWM_dead_time_circuit.v
DPWM_VMC.v
Errors
Console Errors Warnings Find in Files Results
Ln 54 Col 1 Verilog

But today we are only showing one part which consists of DPWM and dead time circuit that we have explained. So, the only concern about this circuit and this Verilog code we have already explained in the previous lecture ok; so, it is a commented signal; so, as we discussed in previous lecture. Now, what we are going to do? We want to do behavioural simulation; so, if we select to go to the simulation mode we have to select.

And in the behavioural simulation sorry we have to first run this because we are doing the behavioural simulation it will check whether the code is fine or not, and then we have to click on that. So, we must make sure that we are talking about this block, only the module that we are going to test in the timing diagram using simulator Xilinx ISE simulator. So, this is the Xilinx ISE simulator then simulates the behavioural model; so, now we are opening this behavioural model.

(Refer Slide Time: 07:04)

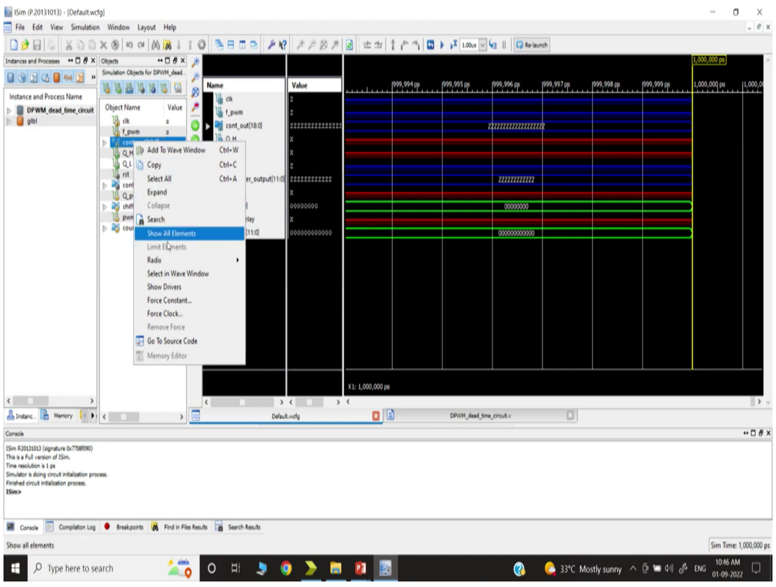


```
ISim R20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Now, we should remember in the simulation mode which was not in the hardware we have to initialize the counter. It is not requiring hardware because the hardware does not care about the initialization, because it is the counter in a physical FPGA device which will always be 0 we cannot.

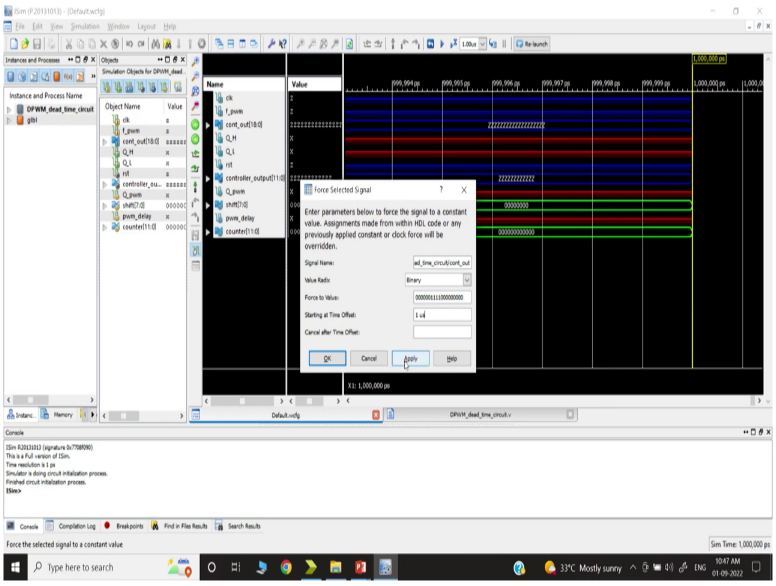
But in the simulation mode we have to define 0 otherwise you know it will always be set like a high impedance mode. So, that is why this line will only be applicable for simulation, but you can keep it for the hardware it will be ignored. So, then we have to set; so, we discussed what are we going to do.

(Refer Slide Time: 07:51)



```
ISim R20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

(Refer Slide Time: 07:53)



Force Selected Signal

Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

Signal Name: ad_time_circuit/cont_out

Value Radix: Binary

Force to Value: 0000001111000000000

Starting at Time Offset: 1 us

Cancel after Time Offset:

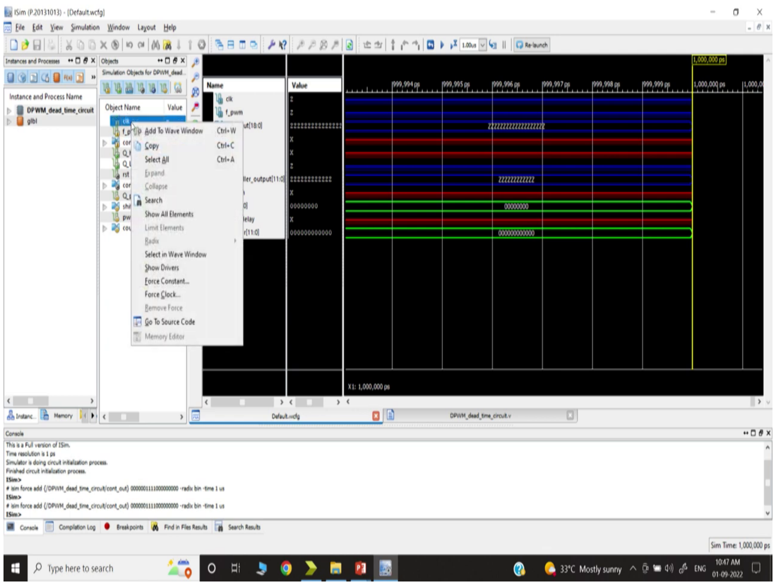
```
ISim R20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Force the selected signal to a constant value

So, we have to select the right force constant and it is in a binary number. So, what is the value that we are going to discuss; so, if you remember we are going to talk about this number; that means, the first 6 digits are 0 then 4 digits are 1 all are 0 ok; so, let us go back. So, the first 6 digit is 0, 1 2 3 4 5 6, then 1 2 3 4, then all 9 bit 0 1 2 3 4 5 6 7 8 9 ok. So, it will start from the beginning or you can mention some starting time; so, let us say we are starting from you know maybe after one microsecond.

Before that, it was you know it I mean may I mean we have to check what value it will take by default, but we are forcing this value at one microsecond; so, we will apply it here.

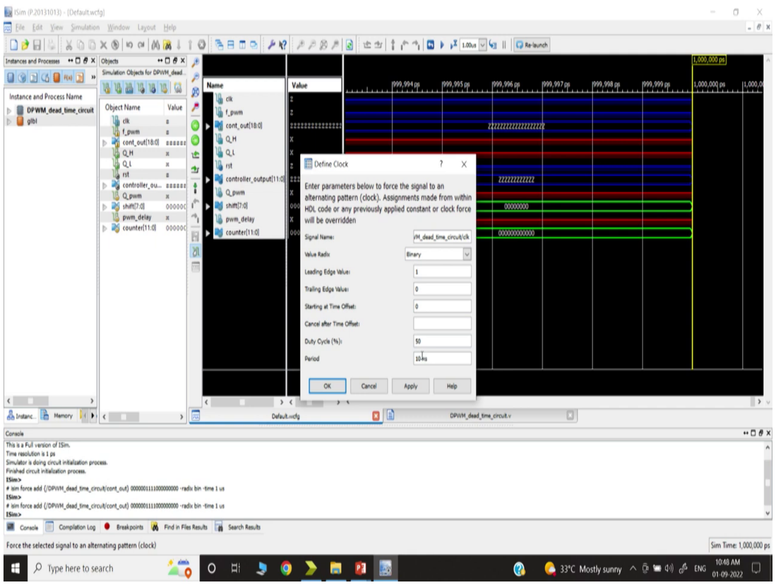
(Refer Slide Time: 08:48)



```
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000001111000000000 -radix bin -time 1 us
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000001111000000000 -radix bin -time 1 us
ISim>
```

Next, the Q the clock is very important; so, sorry we should not make constant, we should use this as a force clock.

(Refer Slide Time: 08:58)



Define Clock

Enter parameters below to force the signal to an alternating pattern (clock). Assignments made from within HDL code or any previously applied constant or clock force will be overridden

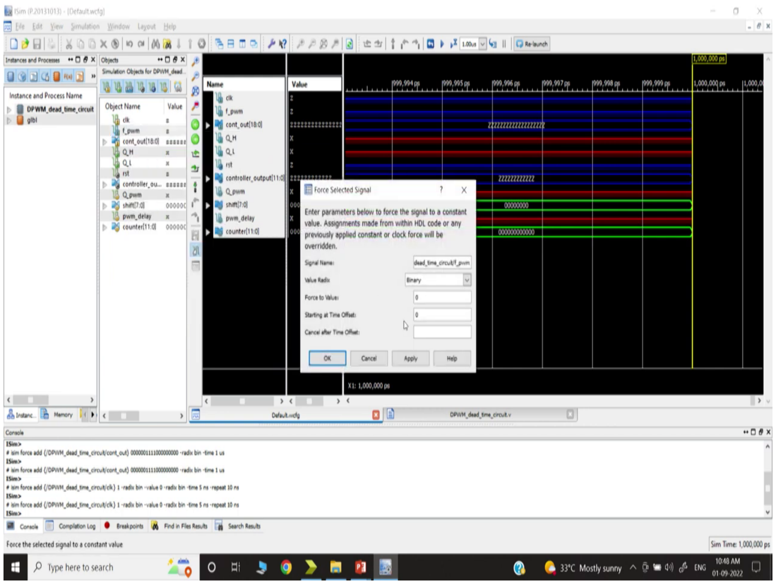
Signal Name: /M_dead_time_circuit/clk
Value Radix: Binary
Leading Edge Value: 1
Trailing Edge Value: 0
Starting at Time Offset: 0
Cancel after Time Offset:
Duty Cycle (%): 50
Period: 10 ns

```
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000001111000000000 -radix bin -time 1 us
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000001111000000000 -radix bin -time 1 us
ISim>
```

Force the selected signal to an alternating pattern (clock)

So, the leading edge is 1 the trailing edge is 0 and duty cycle is 5 per cent and the period is 10 nano second.

(Refer Slide Time: 09:16)



Force Selected Signal

Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

Signal Name: dead_time_circuit/f_pwm
Value Radix: Binary
Force to Value: 0
Starting at Time Offset: 0
Cancel after Time Offset:

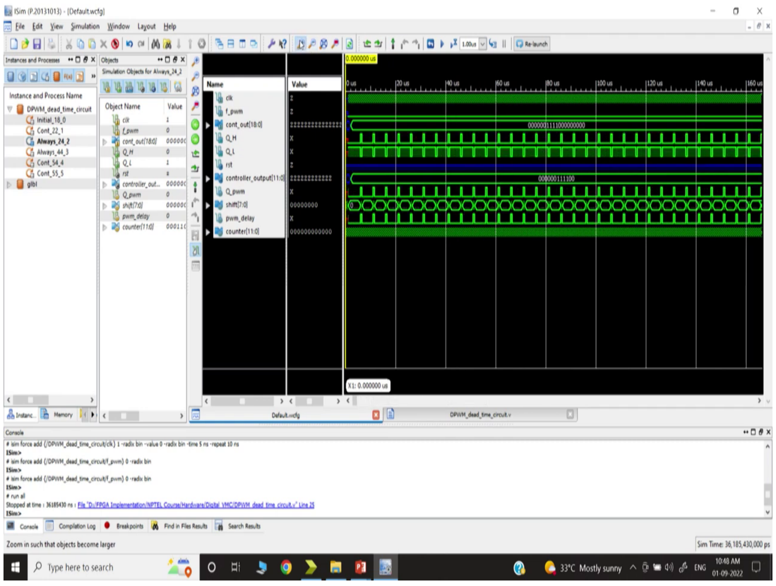
```
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000001111000000000 -radix bin -time 1 us
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000001111000000000 -radix bin -time 1 us
ISim>
# isim force add {/DPWM_dead_time_circuit/clk} 1 -radix bin -value 0 -radix bin -time 5 ns -repeat 10 ns
ISim>
# isim force add {/DPWM_dead_time_circuit/clk} 1 -radix bin -value 0 -radix bin -time 5 ns -repeat 10 ns
ISim>
```

Force the selected signal to a constant value

Sim Time: 1,000,000 ps

So, this is what we need, f w f we are not using; so, we can simply set force constant we can just make it 0; so, we are not using.

(Refer Slide Time: 09:23)



```
# isim force add {/DPWM_dead_time_circuit/clk} 1 -radix bin -value 0 -radix bin -time 5 ns -repeat 10 ns
ISim>
# isim force add {/DPWM_dead_time_circuit/f_pwm} 0 -radix bin
ISim>
# isim force add {/DPWM_dead_time_circuit/f_pwm} 0 -radix bin
ISim>
# run all
Stopped at time : 36185430 ns : File "D:/FPGA Implementation/NPTEL Course/Hardware/Digital VMC/DPWM_dead_time_circuit.v" Line 25
ISim>
```

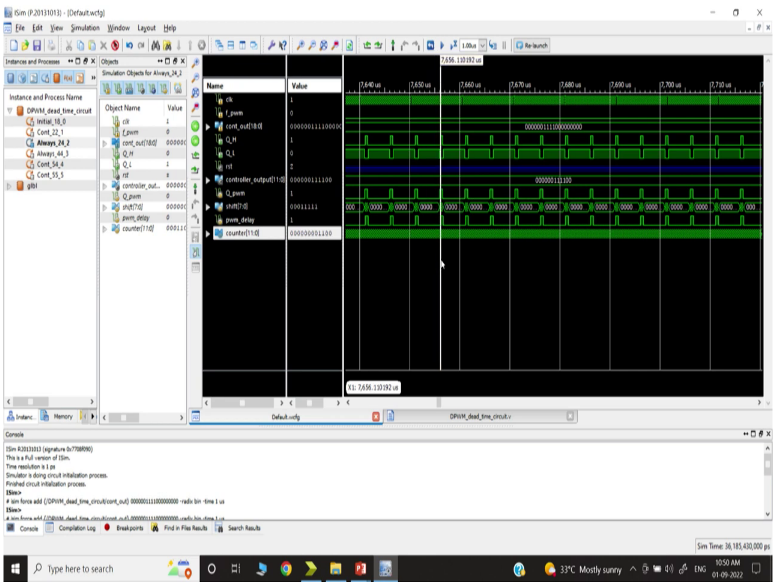
Zoom in such that objects become larger

Sim Time: 36,185,430,000 ps

Now, let us do simulation; so, let us simulate and check; so, we will stop here simulation ok. Now, let us go to the very beginning of this process; so, what does it look like? You see we have defined the value of this control output at this point because I think we are I think somewhere around 100 1000 nanosecond is a 1 microsecond. So, this is where we are going to set, what was the value that we have chosen if you set a resolution.

So, just a minute controller output we have chosen what? I think we have taken 1 microsecond yeah. So, or it could be 2 micro second because you are setting the value I think it was 2 microsecond and it is coming 2000 nanosecond; that means, 2 microsecond yes. So, after 2 microsecond the values have been stored in the controller output and this is where we set.

(Refer Slide Time: 10:46)



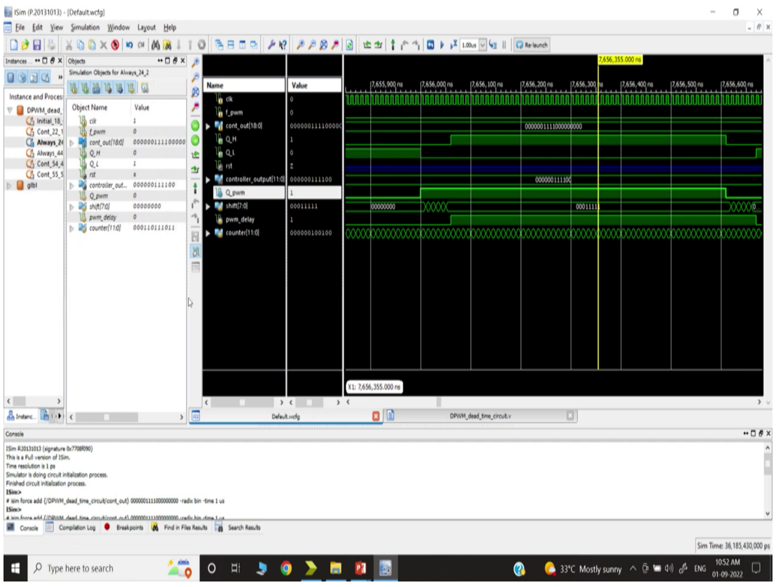
Name	Value
clk	1
f_pwm	0
cont_out[18:0]	000000111100000
Q_H	1
Q_L	0
rst	Z
controller_output[11:0]	000000111100
Q_pwm	1
shift[7:0]	00011111
pwm_delay	1
counter[11:0]	000000001100

```
ISim R20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000001111000000000 -radix bin -time 1 us
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000001111000000000 -radix bin -time 1 us
```

And if you check if you come close, I just want to show that we have set what value? We have set 0 0 0 for all 6 first 6 bit 0 4 bit 1 then the remaining all 9 bit 0. And if you see the control output it will resize; that means, the last 7 bits it has discarded the other bit. So, this is now cube 4 dot 8 format, this was cube 4 dot fifteen formats. And what about the counter? The counter is incrementing; that means, if you go you can see the counter is incrementing this counter is incrementing and it will reset whenever it will reach the 499 value.

So, we will go to that point ok let us go to the PWM signal; that means, let us go to the PWM signal; so, So, let us go to the PWM signal; so, let us choose some here value. So, now, you can see Q PWM is here; that means, let us go here we choose some point here and let us zoom, you see this is a point when the controller counter is getting reset. That means, if you just go inside, I think PWM is high if you check the Q PWM where is the Q PWM yeah; so, if I take it here and now try to take; so, the counter is reset just before that; that means, here.

(Refer Slide Time: 12:28)



Name	Value
clk	0
f_pwm	0
cont_out[18:0]	000000111100000
Q_H	1
Q_L	0
rst	Z
controller_output[11:0]	000000111100
Q_pwm	1
shift[7:0]	00011111
pwm_delay	1
counter[11:0]	000000100100

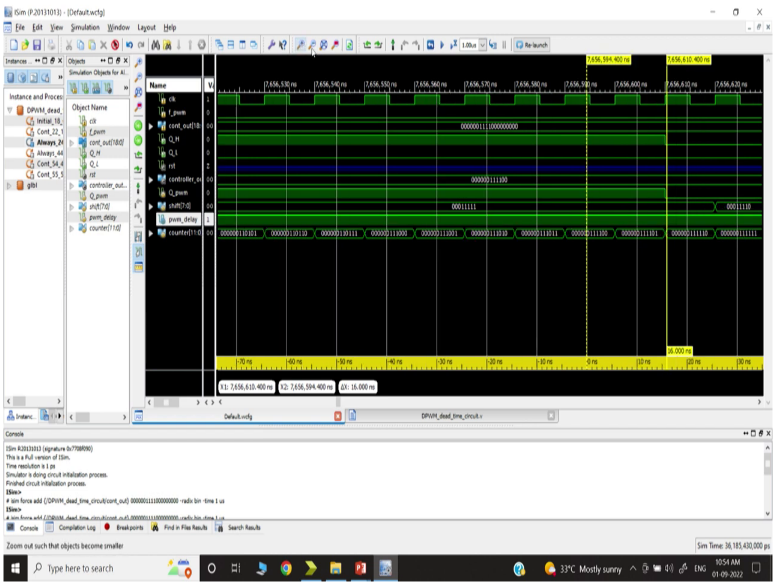
```
ISim R20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000001111000000000 -radix bin -time 1 us
ISim>
```

So, the counter is reset and next cycle it will take action next cycle; so, Q PWM becomes high because at this point the counter reaches 499 value. So, this corresponds to 499 then it becomes 0 and this 0 becomes 0 it is reset and then the action is getting taking place Q PWM is going high the next cycle this is a one-cycle delay. Once it goes high Q PWM; so, you can see the width is getting set I mean whatever width we are getting we are setting it is coming like this ok; so, let us know take this yeah.

So, you see first that when Q PWM goes high then the Q PWM delay is this; that means, there is a delay, let us go back very closely between this point you can see that is delaying, how many cycles it is delaying? So, this is appearing here; so, 1 2 3 4 I mean where Q PWM delay it is here; so, in between how many; so, this is 1 2 here. So, in between it is; so, you can say 1 2 3 4 5 on 6 cycles it is coming. So, there is a 6-cycle delay, because I think we are using 0 to 5 years; so, that delay is coming; so, this is a delayed signal.

The next part we want to see here; so, the next part is that when Q PWM goes high and Q PWM delays there is a delay; so, you can see the high side gate and low side gate signal getting off. So, this is the low side gate signal that is turning off here ok; so, you just take this as the low side gate signal is turning off here and this is what exactly we want the low side gate to be turning on. And the high side is getting high whenever the delay is coming; now, you can take this cursor to this side to see what is happening.

(Refer Slide Time: 15:01)

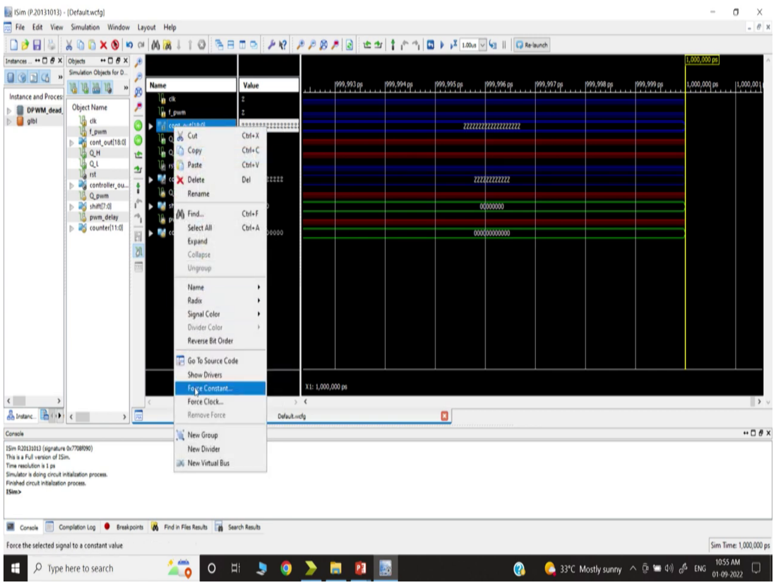


```
ISim R20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000001111000000000 -radix bin -time 1 us
ISim>
```

So, you can take it again whenever the low side signal the Q PWM is going low, then first it is turning off the high side switch and after some time the Q L is on. So; that means, we can generate the dead time circuit and that will be used in the actual hardware; that means, you know if you go back to that.

So, now, you can change this value; that means, we can take another extra one; so, let us go back. So, if you want to again you know you can simulate; that means, if you want to simulate once more maybe you can close this then go back to this we are again doing this simulation.

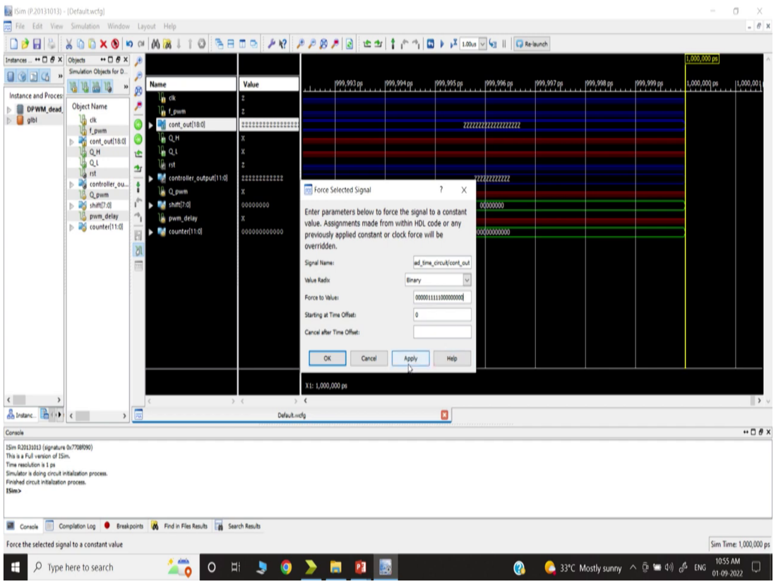
(Refer Slide Time: 15:53)



```
ISim R20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Force the selected signal to a constant value

(Refer Slide Time: 16:02)



Force Selected Signal

Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

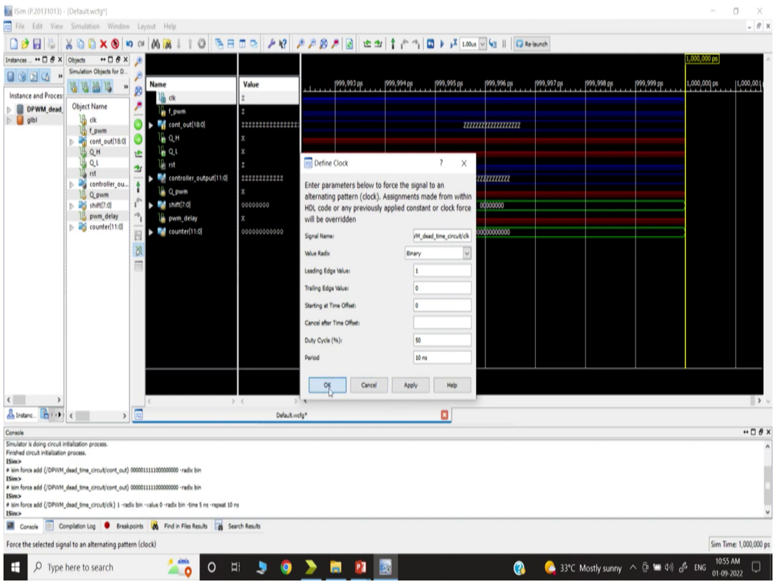
Signal Name: ad_time_circuit/cont_out
Value Radix: Binary
Force to Value: 0000011111000000000
Starting at Time Offset: 0
Cancel after Time Offset:

```
ISim R20131013 (signature 0x7708f090)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

Force the selected signal to a constant value

And now; so, we can change the value; so, let us say now we can fix the value force constant. Maybe we can use 5 0 1 2 3 4 5 then 5 1 1 2 3 4 5, then all 9 0 one 2 3 4 5 6 7 8 9 apply.

(Refer Slide Time: 16:22)



Define Clock

Enter parameters below to force the signal to an alternating pattern (clock). Assignments made from within HDL code or any previously applied constant or clock force will be overridden

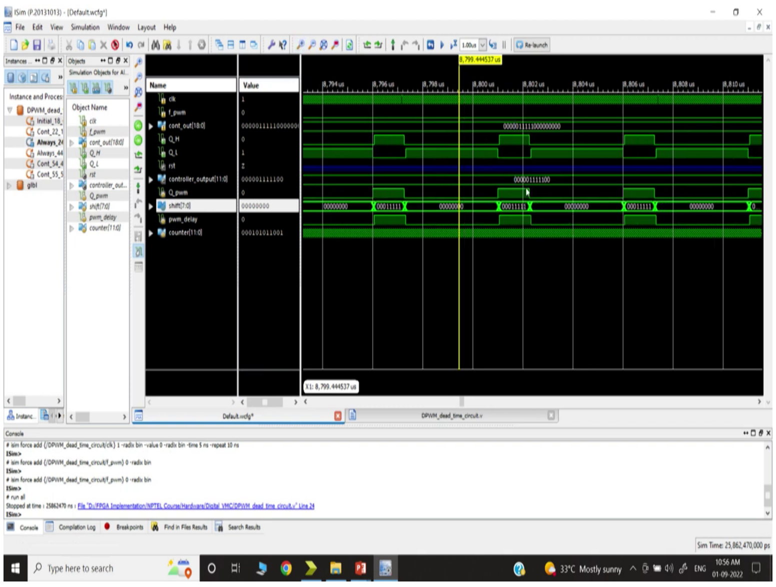
Signal Name: VM_dead_time_circuit/clk
Value Radix: Binary
Leading Edge Value: 1
Trailing Edge Value: 0
Starting at Time Offset: 0
Cancel after Time Offset:
Duty Cycle (%): 50
Period: 10 ns

```
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000011111000000000 -radix bin
ISim>
# isim force add {/DPWM_dead_time_circuit/cont_out} 0000011111000000000 -radix bin
ISim>
# isim force add {/DPWM_dead_time_circuit/clk} 1 -radix bin -value 0 -radix bin -time 5 ns -repeat 10 ns
ISim>
```

Force the selected signal to an alternating pattern (clock)

And then you set the Q L to be force clock 1 0 10 nanosecond and Q PWM we are setting to be force constant to be 0.

(Refer Slide Time: 16:50)

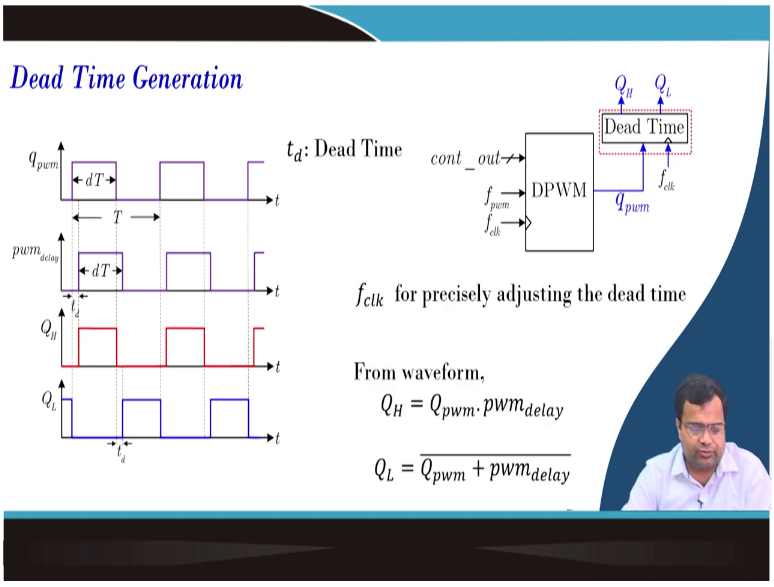


Name	Value
clk	1
f_pwm	0
cont_out[18:0]	000001111100000000
Q_H	0
Q_L	1
rst	Z
controller_output[11:0]	000001111100
Q_pwm	0
shift[7:0]	00000000
pwm_delay	0
counter[11:0]	000101011001

```
# isim force add {/DPWM_dead_time_circuit/clk} 1 -radix bin -value 0 -radix bin -time 5 ns -repeat 10 ns
ISim>
# isim force add {/DPWM_dead_time_circuit/f_pwm} 0 -radix bin
ISim>
# isim force add {/DPWM_dead_time_circuit/f_pwm} 0 -radix bin
ISim>
# run all
Stopped at time : 25862470 ns : File "D:/FPGA Implementation/NPTEL Course/Hardware/Digital VMC/DPWM_dead_time_circuit.v" Line 24
ISim>
```

Now, if we simulate; so, let us go back; so, you can take any arbitrary location here because you want to check the simulation you can check yeah. Now, you can see the PWM duty ratio has increased; now, you can make it you can further increase it. So, in that way, we can adjust the duty ratio and in the closed-loop control, this PWM duty ratio will be coming from the control output.

(Refer Slide Time: 17:30)



Dead Time Generation

t_d: Dead Time

f_{clk} for precisely adjusting the dead time

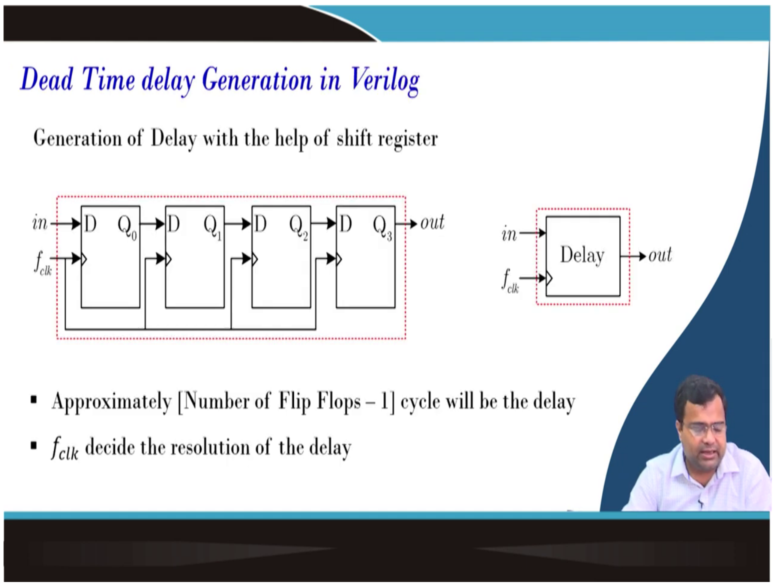
From waveform,

$$Q_H = Q_{pwm} \cdot pwm_{delay}$$

$$Q_L = \overline{Q_{pwm} + pwm_{delay}}$$

So, in summary, we have discussed this control output at the counter and we have successfully validated our dead time circuit. And we have discussed this code and we have discussed how to check, because this is the first step for doing Verilog slowly we are going to implement voltage mode and current mode control. And I will show you know some more light demonstrations of this digital voltage mode control, but the code will be discussed the overall code in the subsequent class.

(Refer Slide Time: 17:53)

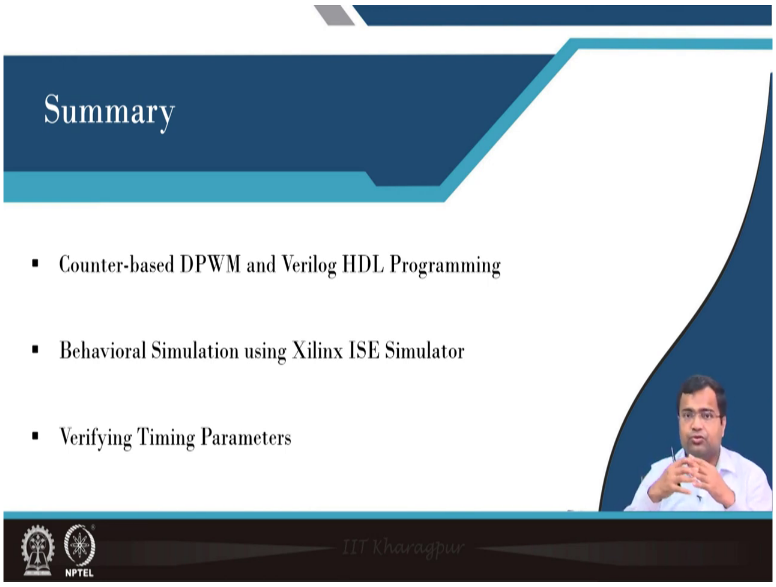


Dead Time delay Generation in Verilog

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

(Refer Slide Time: 18:01)



Summary

- Counter-based DPWM and Verilog HDL Programming
- Behavioral Simulation using Xilinx ISE Simulator
- Verifying Timing Parameters

So, all these things we have discussed dead time circuit you know we have discussed; so, how to generate this delay block these things are also discussed. So, we have; so, in summary, we have discussed the counter base DPWM Verilog HDL coding, and we have successfully tested and verified the timing parameter.

Now, we are ready to move to the next step which will be coming in the next week how to implement digital voltage mode and digital current mode control, and how to implement PID controller PI controller. And how to successfully test it and dump it into the FPGA, how to get the hardware result, and how to implement all this control strategy, that is it for today.

Thank you very much.