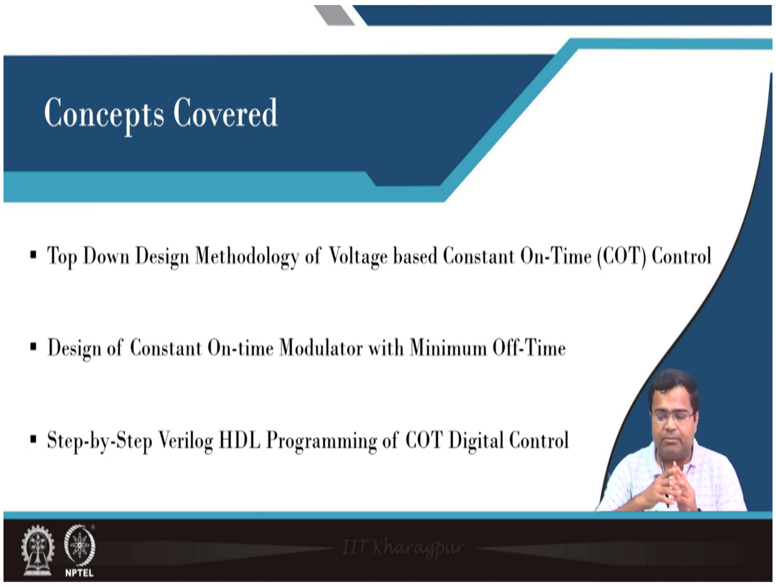
Digital Control in Switched Mode Power Converters and FPGA-based Prototyping

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

Module - 10
Steps for FPGA Prototyping of Digital Voltage Mode and Current Mode Control
Lecture - 96
Verilog HDL Implementation of Voltage-based Constant On-Time Control

Welcome. In this lecture, we are going to talk about Verilog HDL Implementation of Voltage-Based Constant On-Time Control. And we will also show some experimental results.

(Refer Slide Time: 00:36)



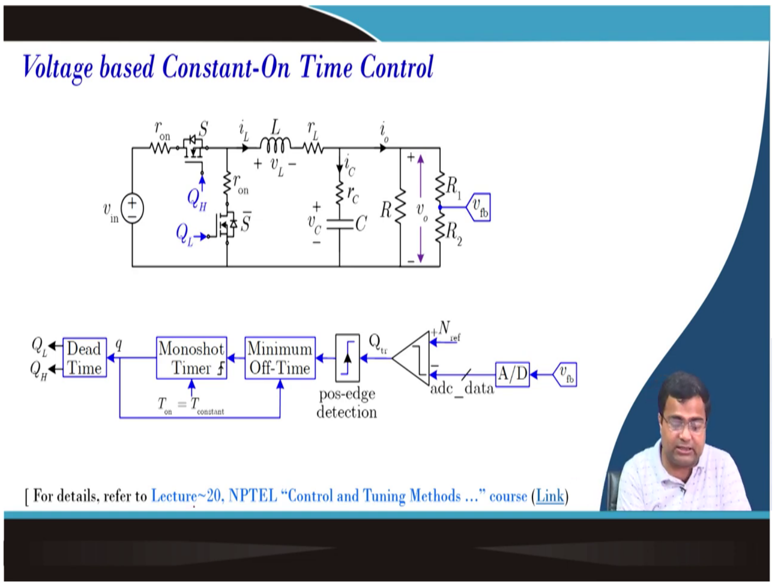
Concepts Covered

- Top Down Design Methodology of Voltage based Constant On-Time (COT) Control
- Design of Constant On-time Modulator with Minimum Off-Time
- Step-by-Step Verilog HDL Programming of COT Digital Control

IIT Kharagpur
NPTEL

So, first, we will talk about top-down we will revisit our top-down design methodology which we have discussed in the previous lecture. Then we will design a constantly on-time modulator with minimum off time and then we will show step-by-step Verilog HDL implementation of constant on-time digital control.

(Refer Slide Time: 00:56)



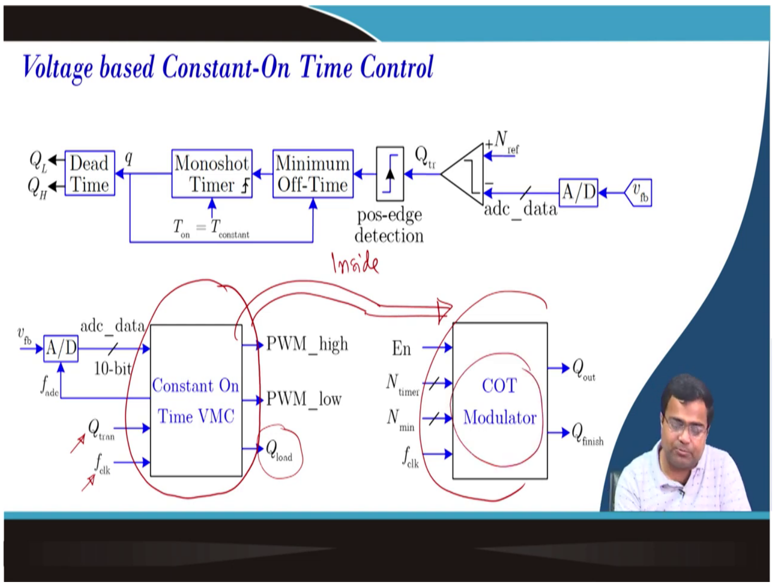
Voltage based Constant-On Time Control

r_{on} S i_L L r_L i_o
v_L
v_{in} Q_H r_{on} Q_L $\bar{S}$
i_C r_C v_C C R v_o R_1 R_2 v_{fb}
Q_L Q_H Dead Time q
Monoshot Timer
Minimum Off-Time
Q_{tr}
N_{ref}
A/D
v_{fb}
adc_data
pos-edge detection
$T_{on} = T_{constant}$

[For details, refer to Lecture~20, NPTEL "Control and Tuning Methods ..." course (Link)

So, first, we will go we will recollect this block we have already discussed in the previous lecture. Then and we discussed the basic concern on the time concept in lecture 20 in our earlier NPTEL course.

(Refer Slide Time: 01:10)



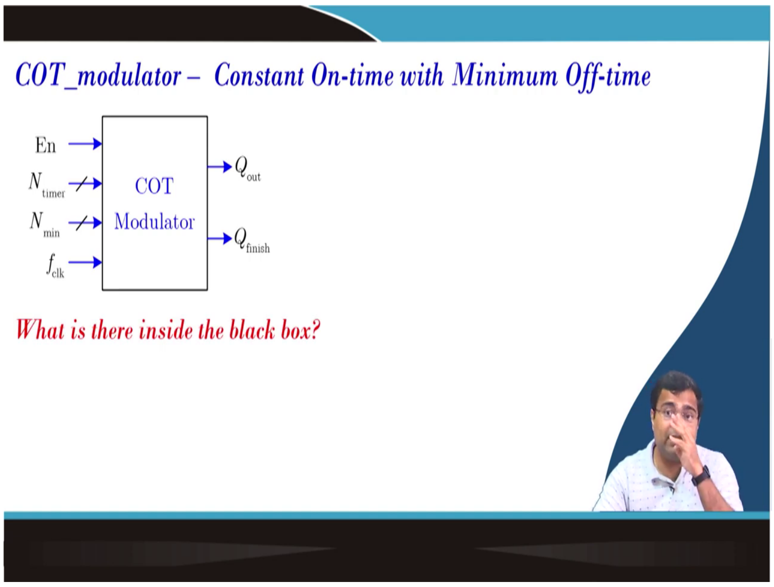
Voltage based Constant-On Time Control

Now, we want to implement using Verilog HDL this block. And we want to synthesize and we want to prototype using FPG. So, in this block, it will accept the data from ADC and it will also send the ADC clock this Q transient is optional we are not using it. The external clock is the clock which is you know high-frequency clock 100 megahertz.

We high side and gate side low side gate signals we are generating from this and the Q load which is for load transient, but here we are not making load transient, because we want to operate in discontinuous conduction mode here. So, this is the block that we are going to now this block inside this block we need to consider this modulator.

This is the heart of constant time where we need to have a constant on-time modulator. So, this block is inside this block.

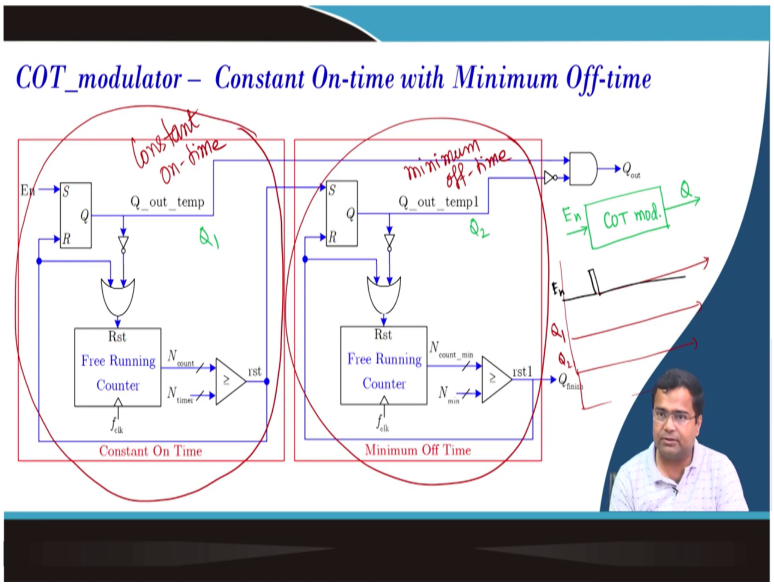
(Refer Slide Time: 02:09)



COT_modulator – Constant On-time with Minimum Off-time

What is there inside the black box?

(Refer Slide Time: 02:17)



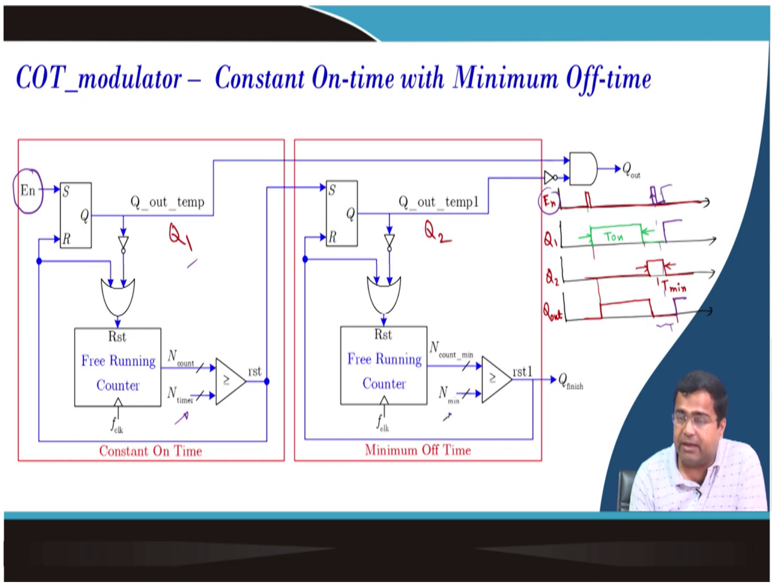
COT_modulator – Constant On-time with Minimum Off-time

Now, so, if we talk purely about the constant on-time modulator with minimum off time we have discussed, what is there inside the box? We have discussed that this particular block in the previous lecture we have discussed is for constant on time.

Constant on time and this block will represent minimum off time. And more precisely if we consider that overall constant on time modulator I would say COT modulator, if this is your enable clock and this is your overall Q. So, we are dividing this part to be Q1, this part to be Q2.

So, what do we want for Q1? So, whenever our enable pulse to comes; that means, this is our enable pulse to come to enable pulse to come. What do you want that? First, we want to show how does my Q1 look like. What does my Q2 look like? And finally, how does my ok. So, I will use a different place here. Maybe I will erase it so, ok. So, let me draw here.

(Refer Slide Time: 03:50)


COT_modulator – Constant On-time with Minimum Off-time

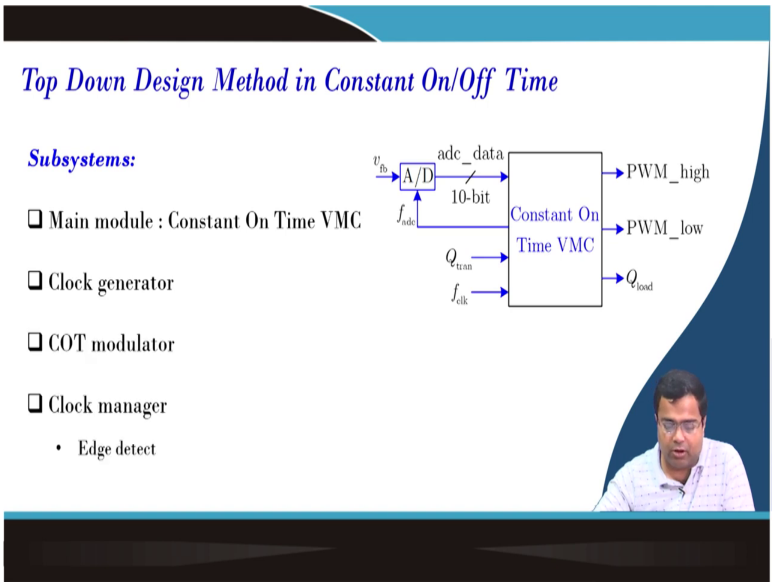
So, let me draw here. Let us say this block, this block, this block and we can divide this into 2 pieces. This block, this block. Now, first, we will have an enabled clock. So, enable. So, it is coming high here ok. Now, we want to write here to be Q1, here it is Q2 and this is Qout. So, we want to draw Q1, we want to draw Q2 and we want to draw Qout.

So, in this case first, whenever this pulse comes this will go high, and how long it will be enabled? It will be during this Ton time. Then whenever the Ton finish is counted this is the time Q2 will be high and this duration is what this duration is Minimum time is the minimum off time. And what will be our Qout? The Qout will go high low and low.

So, I am just saying, this point to this point it will release; that means, suppose we have another enable signal coming in. So, this will only respond after this time; that means, during this time this will not respond. So, if another enable comes this will not respond until the minimum off time is elapsed. So, it will only respond after this; that means, this Q can go high only here. That will l enable.

S ito, we need to make sure this enables signals should be placed here rather than here. So, you should not enable it. So; that means, we need to generate this enable signal through a proper clocking. So, we will be using a clock manager. Q1 will be generated for the counter. So, it is customizable you can set the timer minimum time is also customizable which is why if you look at this particular block we are talking about.

(Refer Slide Time: 06:16)

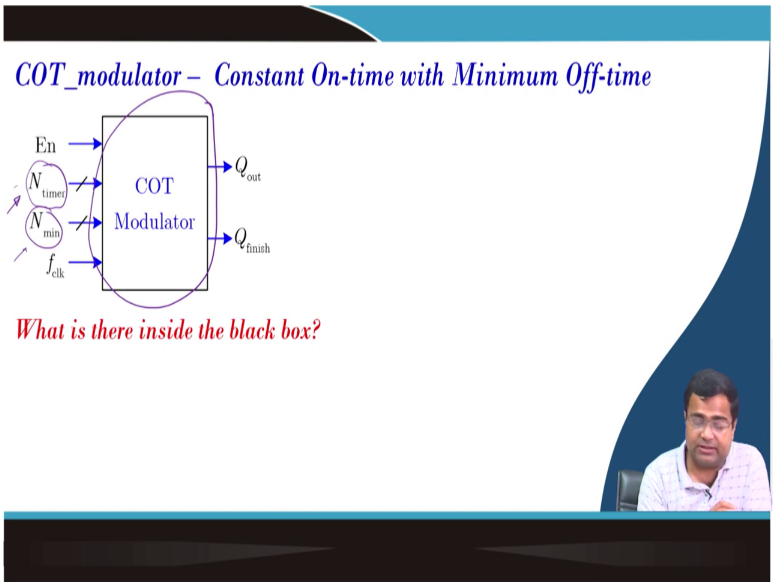


Top Down Design Method in Constant On/Off Time

Subsystems:

- ❑ Main module : Constant On Time VMC
- ❑ Clock generator
- ❑ COT modulator
- ❑ Clock manager
  - Edge detect

(Refer Slide Time: 06:20)



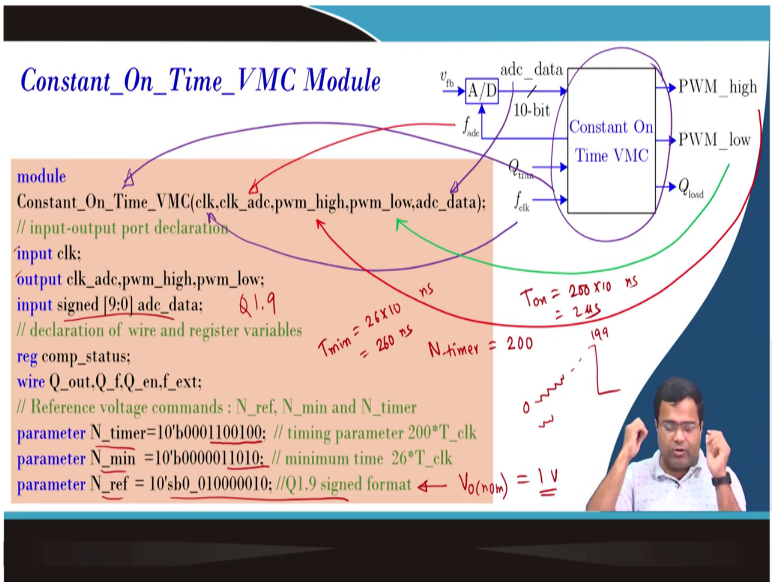
COT_modulator – Constant On-time with Minimum Off-time

What is there inside the black box?

So, if you go back to our previous thing these things are the synthesis we want to synthesize this modulator where we can vary, because if you go for adaptive on time. So, the on-time can be varied by simply varying this number. Similarly, we can vary the minimum one time. So, these things are possible by the adaptation of this ok.

We want we already discussed the main module, clock generator, COT modulator, and clock manager. So, the clock manager's job is to generate the enable signal which is very crucial ok. And that also requires an edge circuit.

(Refer Slide Time: 06:55)



Constant_On_Time_VMC Module

```
module
Constant_On_Time_VMC(clk,clk_adc,pwm_high,pwm_low,adc_data);
// input-output port declaration
input clk;
output clk_adc,pwm_high,pwm_low;
input signed [9:0] adc_data;
// declaration of  wire and register variables
reg comp_status;
wire Q_out,Q_f,Q_en,f_ext;
// Reference voltage commands : N_ref, N_min and N_timer
parameter N_timer=10'b0001100100;  // timing parameter 200*T_clk
parameter N_min  =10'b0000011010;  // minimum time  26*T_clk
parameter N_ref  = 10'sb0_010000010; //Q1.9 signed format
```

So, first, we will go for a constant on-time modulator. So, this is the basic block. So, this is the overall control. So, we are talking about this control logic and this is the main module.

What is considered here? We are talking about this clock? We need a clock for the ADC which is sitting here the clock ADC is here. Then we high it is coming from here. Then we have this PWM low which is here ok. And ADC data which is coming from here. Adc data is coming it is here. So, these are the input-output port and we have defined here that we are not using Q load we set it to 0. So; that means, you know we are not sending anything, we are not sending anything from here.

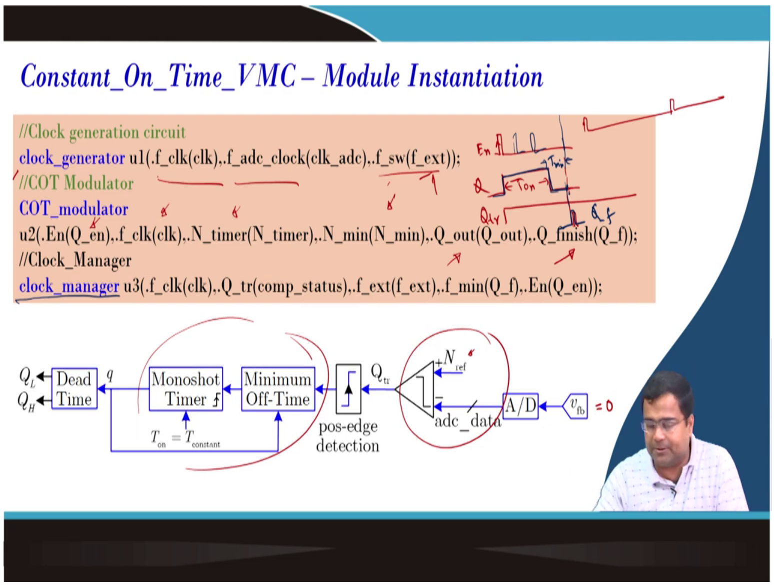
So; that means, if you do not send anything it will be set at a 0 position and we know about the defining how to define this quantity; that means, you know input, output vector input and we are taking signed data it is Q1 dot 9 format we have discussed earlier. Now, we want to parameterize; that means, N timer, N min, and N ref.,

N ref we have discussed earlier this corresponds to V0 nominal to be 1 volt. That is the nominal voltage we used earlier. Now, if you take the N timer if you take this binary number N timer here is nothing but in decimal, it will be 200. So; that means, 200 times we are creating on time.

Because we have discussed that it will take 200 such cycles from 0 to 199 it will go right like this. And the minimum time we are taking is 26 cycles; that means, it will go up to 26. So, that we will have a 26; that means, the T minimum will be how much? 26 into 10 nanosecond, sorry.

26 into 10 that is nanosecond which is nothing but 260 nanosecond. And what is 10 constant on time? It is 200 into 10 nanosecond which is 2 microsecond, but you can reduce it, because 2 microsecond may be large. So, these quantities are adjustable.

(Refer Slide Time: 09:32)



Constant_On_Time_VMC – Module Instantiation

```
//Clock generation circuit
clock_generator u1(.f_clk(clk),.f_adc_clock(clk_adc),.f_sw(f_ext));
//COT Modulator
COT_modulator
u2(.En(Q_en),.f_clk(clk),.N_timer(N_timer),.N_min(N_min),.Q_out(Q_out),.Q_finish(Q_f));
//Clock_Manager
clock_manager u3(.f_clk(clk),.Q_tr(comp_status),.f_ext(f_ext),.f_min(Q_f),.En(Q_en));
```

Now, we are instantiating the clock generator circuit which is generated by taking the clock high-frequency clock generating ADC clock.

And we are generating an external clock. Why? I will tell you. So, this external clock lets us say if our let us say we are talking about the voltage. So, we are talking about this Q trigger Qtr. Suppose you start the circuit when the output voltage was 0. So, this guy was 0 and then this Nerf circuit was it was its value.

Now, since you are starting the circuit. So, voltage; means, first this Qtr will be high, at the beginning of the circuit when we start then your timer will be Q will be high for this on time and we know there is a minimum off time that goes high. Now, this Qtr ly high, because from the very beginning as long as the output voltage is below the reference voltage. It will remain high unless it goes above.

Since it is high. So, even after this time elapsed and the minimum off time also elapsed. There is no edge trigger signal. So; that means, this enable signal requires an edge trigger to turn on this monoshot timer. So; that means if there is no edge trigger signal it cannot turn on the timer and the voltage will collapse.

So, we need to create this thing. That is why we are giving an external signal which will I will discuss with I will discuss in the subsequent slide, and how this enable signal should be generated using an external clock. That means, external clock there is a fixed frequency clock that will check whether you know the Q trigger is high, if it is high there is no edge then this will also generate an edge.

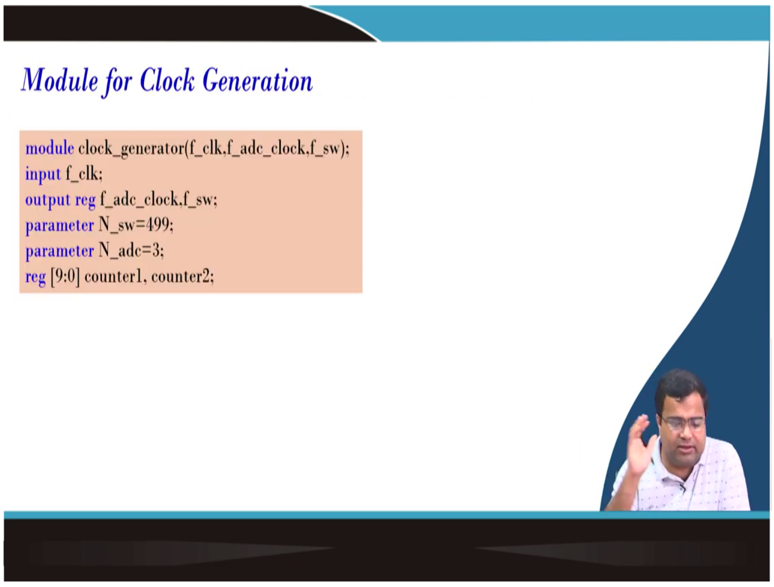
That is like that, that is something. Similarly, we suppose we want to synchronize your constant on-time modulator with an external clock. Then this external clock can be used and we are not going to discuss the synchronization effect, but it is possible, but for the time being, let us say there is a clock. Now, we have a cot modulator where input is the enable signal it has a high-frequency clock that will generate the timer on time and off time.

It will take the data as an on-time comment, the minimum off-time comment will generate the output Qout and that Qout will be the output and it will generate the Q finish. When how does the Qfinish is generated? Whenever the on time is elapsed and minimum off time is gone. This is my T off, T min then this will generate a flax signal; that means, this will generate a flax signal, and we are talking about Q finish.

So, Qfinish is not simply after finishing on time, it is after finishing on time followed by the minimum time then it will say that you can now turn on the timer; that means, the turn the on monoshot timer is relaxed or it is released. So, you can now turn it on again, but until this minimum off time is not complete, it will not enable any further change.

Similarly, during this process, if multiple enable signals come it will not respond, because it will remain it is a monoshot timer it goes to the one state and then comes back after the on-time then minimum off time then only it will entertain any further enable signal. So, that is why we need to generate a clock manager. So, that which actually can manage this enable signal.

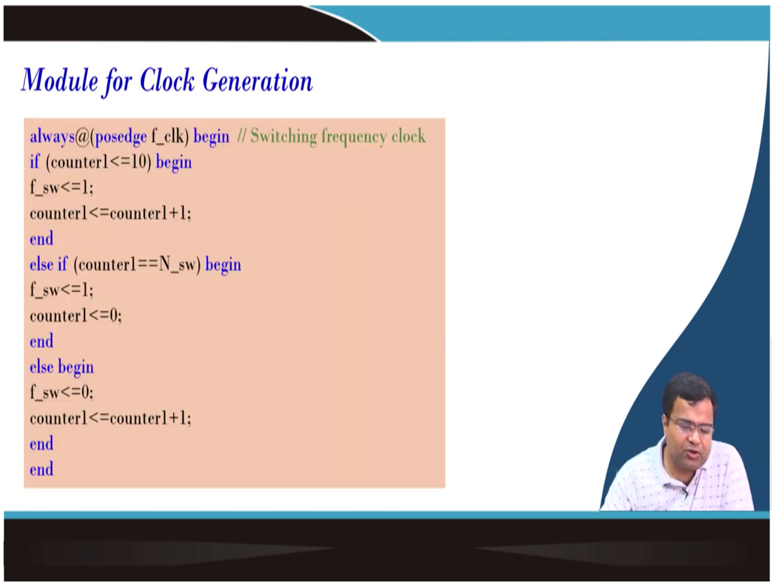
(Refer Slide Time: 13:25)



Module for Clock Generation

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

(Refer Slide Time: 13:33)

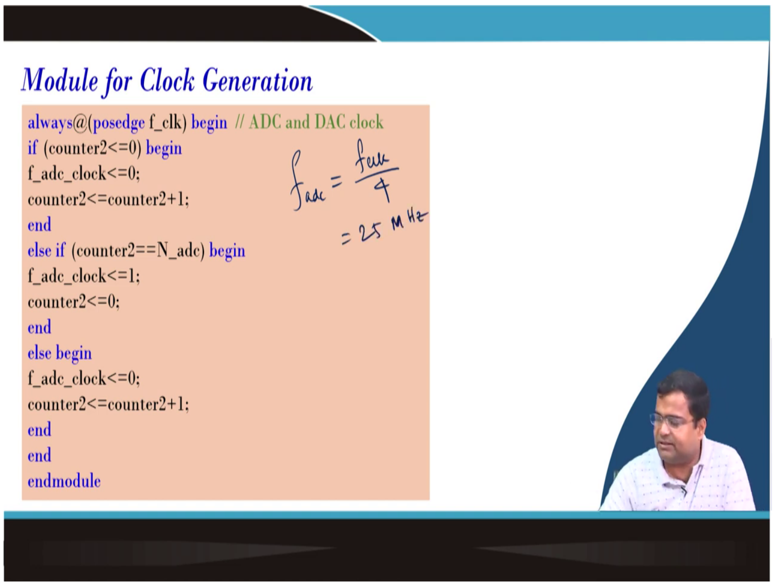


Module for Clock Generation

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

So, regarding the clock generator, I think we have discussed that this block was the same block we have used which we have used for voltage mode and current mode.

(Refer Slide Time: 13:42)



Module for Clock Generation

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

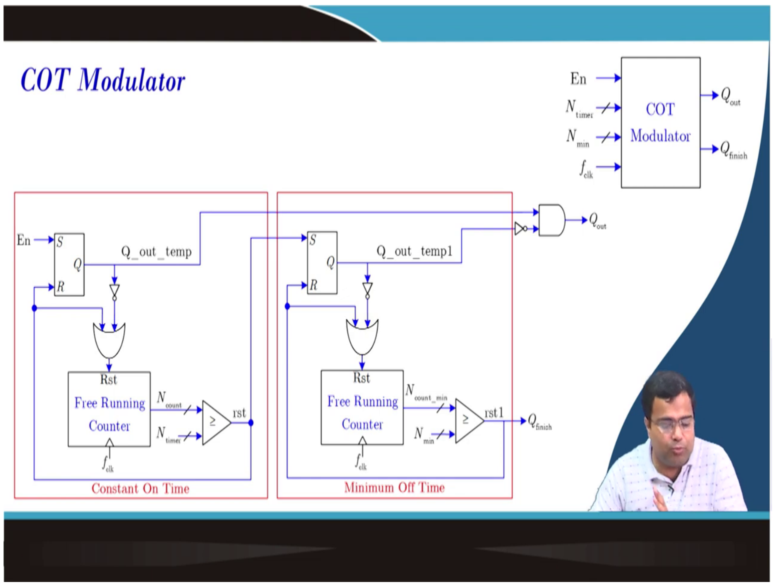
$$f_{adc} = \frac{f_{clk}}{4}$$
$$= 25 \text{ MHz}$$

So, we are using you know we are using this we want to generate a switching frequency clock which is it will count up to 499 OKs. And this will go up to now 0 to 499.

For ADC we are using just three clocks; that means, we want to generate an f ADC which is an f clock divided by 4 that is it; that means, which is 25 megahertz.

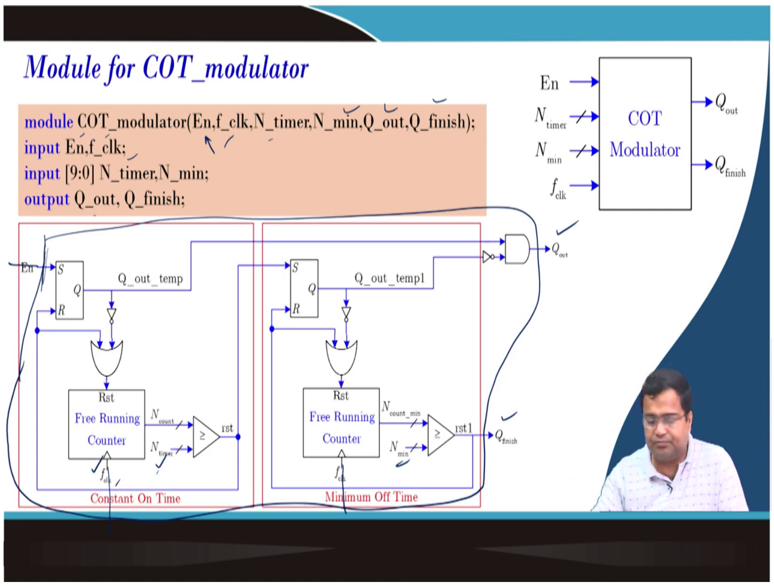
(Refer Slide Time: 14:04)

COT Modulator



So, we have discussed earlier next the COT modulator. This block we have presented and this block also we have presented. So, how does it work?

(Refer Slide Time: 14:12)



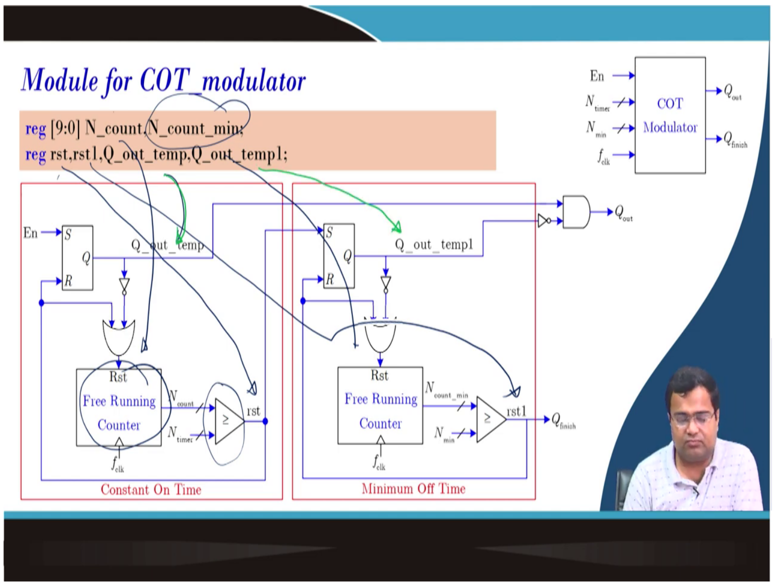
Module for COT_modulator

```
module COT_modulator(En,f_clk,N_timer,N_min,Q_out,Q_finish);
input En,f_clk;
input [9:0] N_timer,N_min;
output Q_out, Q_finish;
```

So, the COT modulator first the COT modulator is a submodule or it is a module that is instantiated within the main module. It requires an enable signal this is the boundary of this, requires a timer signal, and requires a minimum value Qout is the actual out, and Qfinish is the finish. So; that means, you can imagine that this enables this Qout this modulator will show as it will appear like this block and yeah.

So, it will take the data clock from outside that is the clock. It will also need the timer command, and minimum on-time command all these commands are given. Now, the input is enabled signal then the f clock which is the high-frequency clock here this clock N timer we have discussed and the output is a Qout and Qfinish ok.

(Refer Slide Time: 15:10)



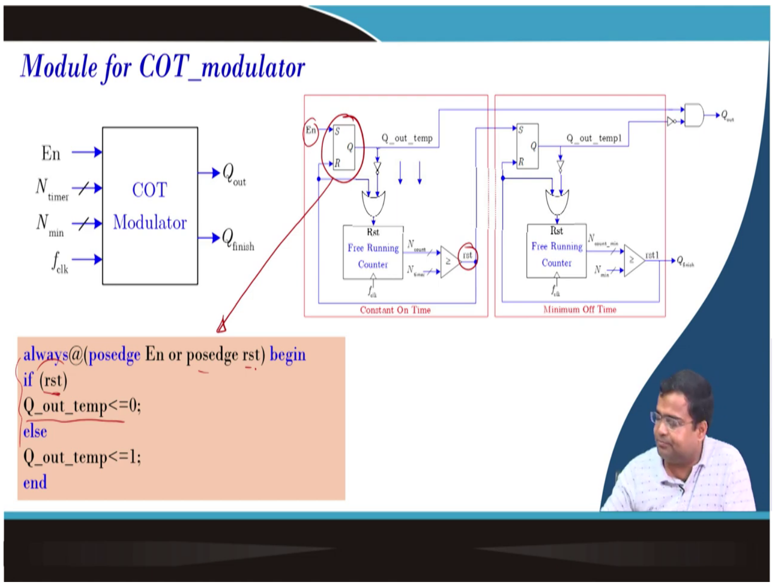
Module for COT_modulator

```
reg [9:0] N_count,N_count_min;
reg rst,rst1,Q_out_temp,Q_out_temp1;
```

Now, how are you going to generate? So, N count and N minimum count; means, we need 2 counters one is this. So, this counter is here. Another is this counter is here. And both are taking 10-bit counter. It is an unsigned counter and we have defined first which is this signal here and rst1 which is this signal here these 2 are the resistor that will generate the whenever the change comparators take place.

And there are some Qtemp you can see this is the signal. So, let me use a different color. This is the signal and this Qtemp is this signal ok. So, here we want to erase this ok.

(Refer Slide Time: 16:04)



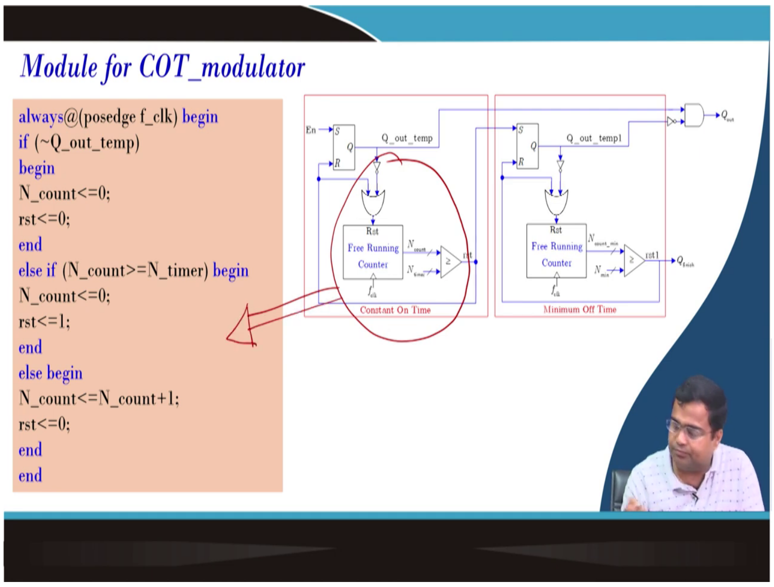
Module for COT_modulator

```
always@(posedge En or posedge rst) begin
if (rst)
Q_out_temp<=0;
else
Q_out_temp<=1;
end
```

Now, the next is always at the end. So, this first we are implementing which logic? We are implementing this logic. So, this logic is here. Always at the posedge of enable or the posedge of this reset.

Reset then it goes inside it check if the reset is high the Q temp is set to 0 because this is the rs flip flop. Otherwise, it will be 1 because this comment will be executed only when either the enable high positive edge comes or the positive sorry reset comes. And if positive edge reset comes; that means, if it is high it will be 0 and if the enable reset comes if the reset is low it will be. So, it is a reset-dominated rs flip-flop.

(Refer Slide Time: 16:50)



Module for COT_modulator

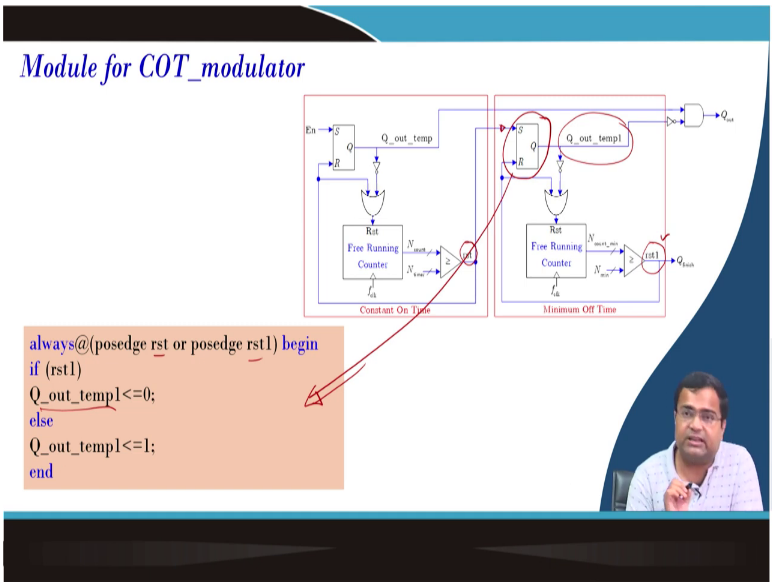
```
always@(posedge f_clk) begin
if (~Q_out_temp)
begin
N_count<=0;
rst<=0;
end
else if (N_count>=N_timer) begin
N_count<=0;
rst<=1;
end
else begin
N_count<=N_count+1;
rst<=0;
end
end
```

Next, we want to implement this counter. So, this counter will check at every clock edge. First, it will check whether this output is high or low because we want to implement this logic. So, it will take the Q output not the gate you can see not the gate. If the Q output, not gate is high; that means, the Q output, not gate is low; that means, it goes low then this will be high and in that case, the counter is simply 0.

That means we are not going to count when this goes low, when this goes high then this signal will be low and this reset will be released in that case the counter will start counting. Now, the counter will start counting and reset is 0 when reset will be high when the counter will try to exceed the timer and this is the comment that means when the counter goes above or equal to the timer then the count will counter will be reset and it will generate the reset to be high that is active high of the reset signal.

So, this whole logic is to use this block. I mean I would say this whole block is implemented by this Verilog code.

(Refer Slide Time: 18:08)

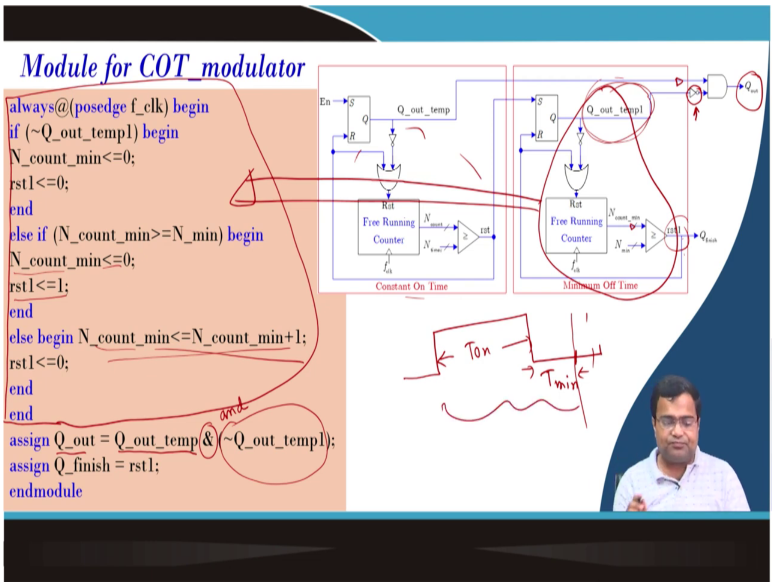


Module for COT_modulator

```
always@(posedge rst or posedge rst1) begin
if (rst1)
Q_out_temp1<=0;
else
Q_out_temp1<=1;
end
```

Next again we are implementing this block. This block is here. So, it takes the posedge off. Now, for the next stage, the reset rst of these posedges is like a enable signal and rst 1 of this will be the posedge it's like a reset signal. So, again if this signal goes high then this output will be 0 and if this is not high and this edge comes positively it will be set to high.

(Refer Slide Time: 18:40)



Module for COT_modulator

```
always@(posedge f_clk) begin
if (~Q_out_temp1) begin
N_count_min<=0;
rst1<=0;
end
else if (N_count_min>=N_min) begin
N_count_min<=0;
rst1<=1;
end
else begin N_count_min<=N_count_min+1;
rst1<=0;
end
end
assign Q_out = Q_out_temp & (~Q_out_temp1);
assign Q_finish = rst1;
endmodule
```

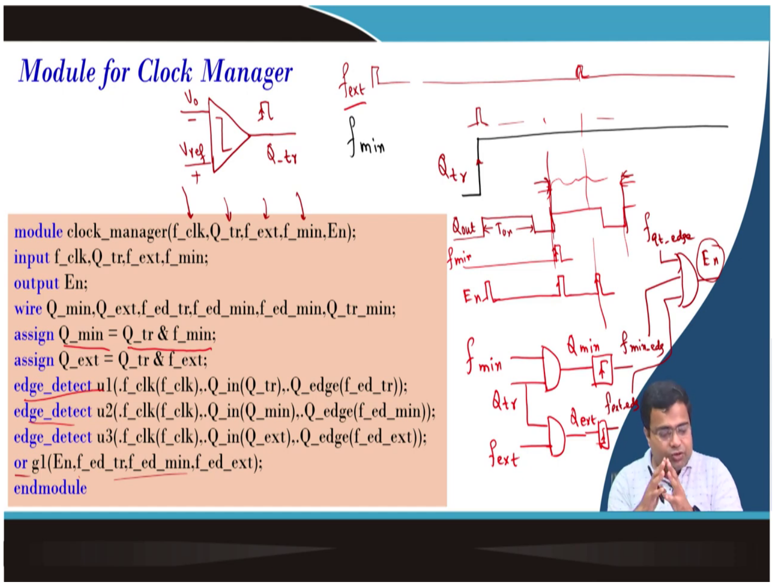
Next, again the same we want to implement this logic. Sorry, we want to implement this logic which is this particular logic. And this is nothing, but this. So, it is the same as this block where this counter will be 1 or 0.

So, this is what, but now up to this point I will say this is up to this point this always block. Where again it ends in the clock edge, if the temp out is 0 then it is disabled, the counter will not be counting if reset is also 0, but if it is 1 then this will be reset if disabled then the counter will start counting ok. And this counter name is the minimum time and the reset is 0.

Once this counter exceeds the minimum value then the counter will be reset and reset 1 will be activated. This will be and this will implement this logic. Now last what is Qout? Qout assign where variable it is the Qout this signal you can check anded to it, not of this. So, anded this is the and in the data flow modeling and operation and this is the invert of this logic which is nothing, but this.

And what is Q finish? It is reset to 1 because this will allow any further action in the monoshot timer. After all, you have completed your timer followed by minimum off time; that means, this is your Ton and here let us say you have completed your Tmin. Then the counter is allowed for further you know you can trigger it further if you want, but till this point, you cannot do anything.

(Refer Slide Time: 20:33)



Module for Clock Manager

```
module clock_manager(f_clk,Q_tr,f_ext,f_min,En);
input f_clk,Q_tr,f_ext,f_min;
output En;
wire Q_min,Q_ext,f_ed_tr,f_ed_min,f_ed_min,Q_tr_min;
assign Q_min = Q_tr & f_min;
assign Q_ext = Q_tr & f_ext;
edge_detect u1(.f_clk(f_clk),.Q_in(Q_tr),.Q_edge(f_ed_tr));
edge_detect u2(.f_clk(f_clk),.Q_in(Q_min),.Q_edge(f_ed_min));
edge_detect u3(.f_clk(f_clk),.Q_in(Q_ext),.Q_edge(f_ed_ext));
or g1(En,f_ed_tr,f_ed_min,f_ed_ext);
endmodule
```

So, next is the important part of how to generate this enable signal. So, this is where it is coming from? So, the enable signal generation requires a clock signal high-frequency clock Qtr, which is the comparator output because we have taken the voltage comparator; that means, we as if you can assume this is the output voltage and the reference voltage.

In a very simple sense, whenever the output voltage goes above the reference voltage or the reference voltage is above; that means the output voltage falls below the reference voltage. This will generate a trigger pulse we call a Q trigger. So, this may not be triggered as long as the output voltage is below reference all the time Qtr is high.

So, this is the Qtr, Q external I told them that we will check the status periodically. So, this is my external signal which is a fixed frequency signal. So, it has a dual purpose, in this case, we are using it to check the status, but in the future, you can use this clock to synchronize your time period of the monoshot timer, but we are not going to discuss that part.

And f min; that means, what is f min? It will check whenever you know this minimum of time elapsed it will generate another trigger pulse; that means, it will generate a trigger pulse when this is this we call it an f min. What is the purpose? It will check whenever the off time is elapsed it will check whether this Qtr; that means, the output of the comparator is still high if it is high and if there is no edge then this guy will create an edge.

So, that the monoshot timer will be again activated. So; that means, the monoshot timer will be only activated subsequently only after the minimum on-time, and off time. So, that is why this f min is generated f min it will be enabled when you're on time is elapsed followed by minimum off time is elapsed then this will be generated and we will check the status of the Qtr is high; that means, what I am saying if the Qtr is high for a long time.

So, the first time this is your Qtr for example. So, the first time I think the mono shot timer will be enable that will disable for minimum off time. Now, this is your I am saying the Qout. So, this is your Ton, but after that Ton. Now your f min will generate, f min. It will check since it is high it will create a pulse again; that means, you turn on the timer again because you need to turn it on if the voltage that is already below.

But suppose the counter is not turned on then this signal will check at this stage. So, at this stage again this signal will come and since Qtr is high it will again generate an enable signal. So, this is an enable signal, it is enabled here it will further enable, but here as if again the counter goes high it goes for this minimum and all, but during this time if multiple enable signal comes it does not matter.

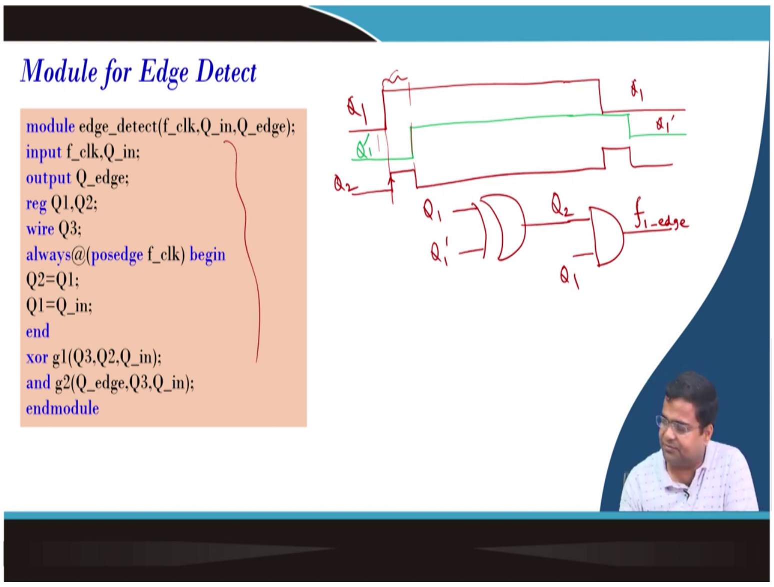
Once it is enabled, because it will not respond to any external enable. As long as this is already operational. So, that is why it does not matter. So, this frequency clock is kept to make sure the voltage should not collapse; that means, we should turn on the enable timer. So, that is why you see I have generated the Q min which is the combination of Qtr and so; that means, I have added this f min anded with Qtr, and Qtr is also anded with f external.

So, this I am calling a Q min and this I am calling a Q external and now I am using an edge detection circuit. So, you can see that there is an edge detection circuit and these edges are finally, or to get and since these 2 edges only take this we also need to consider another edge. What is that edge? The edge of the Qtr itself.

So, 1 is the Qtr edge itself, because we also need to identify the edge of this guy as well. So; that means, there will be an edge detection circuit if any edges come. So, one edge is a Q min edge sorry I will say f min edge f external edge, another I want to say f Qtr edge. So, all 3 edges will be detected and these 3 edges will be this will be coming these are all or.

So, finally, all or all these edges and that is my final enable signal; that means, enable signal edge can is a combination of the edge of the Qtr, the edge of the f min, and the edge of the external, but all this edge will make sense in the timer only when you are allowed within this duration if there are multiple edges come it will not respond.

(Refer Slide Time: 26:09)



Module for Edge Detect

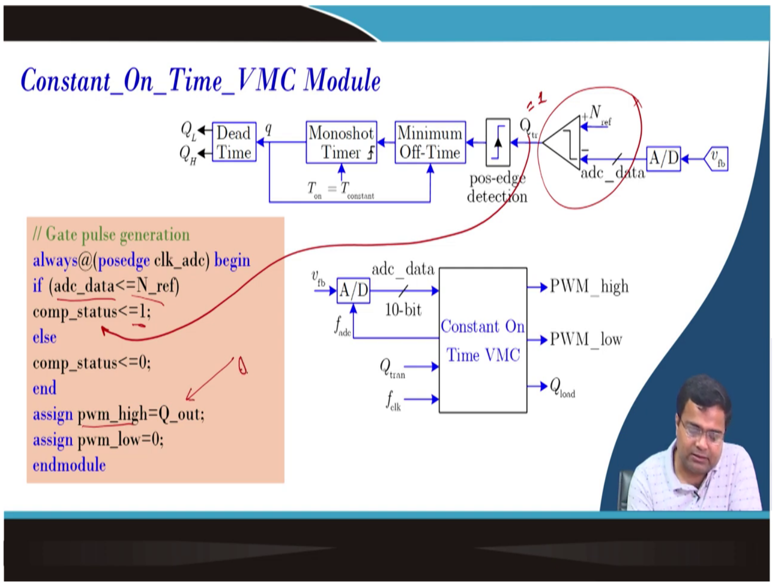
```
module edge_detect(f_clk,Q_in,Q_edge);
input f_clk,Q_in;
output Q_edge;
reg Q1,Q2;
wire Q3;
always@(posedge f_clk) begin
Q2=Q1;
Q1=Q_in;
end
xor g1(Q3,Q2,Q_in);
and g2(Q_edge,Q3,Q_in);
endmodule
```

So, this completes the implementation now if you go to edge detection what does it do? Suppose you take any signal we want to detect the edge, let us say this is my Q1 signal, what we will do? I will delay this signal. This is my Q1 dash then what I will do? I want to take the positive edge. So, let us say this signal goes high here. So, naturally, this signal will go high-low here.

So, if we use an edge detection circuit this will be your Q1. So, Q1 dash and this is Q1. So, and Q1 dash if you pass through a xor gate then if this is your Q let us say 2. So, the Q2 will be high during this time Q2 xor gate, but we want to detect it. So, depending upon how much the delay you are giving that width will depend now we want to detect a positive edge.

So, that is why after this you have to and with Q1, because this time the Q1 is high. So, it will detect and this is my actual positive edge I will say Q f 1 edge. So, the edge of this Q1 signal can be detected by this logic and this is exactly what we did.

(Refer Slide Time: 27:45)



Constant_On_Time_VMC Module

```
// Gate pulse generation
always@(posedge clk_adc) begin
if (adc_data<=N_ref)
comp_status<=1;
else
comp_status<=0;
end
assign pwm_high=Q_out;
assign pwm_low=0;
endmodule
```

Next, we have this modulator. Now, we want to generate this particular implementation logic.

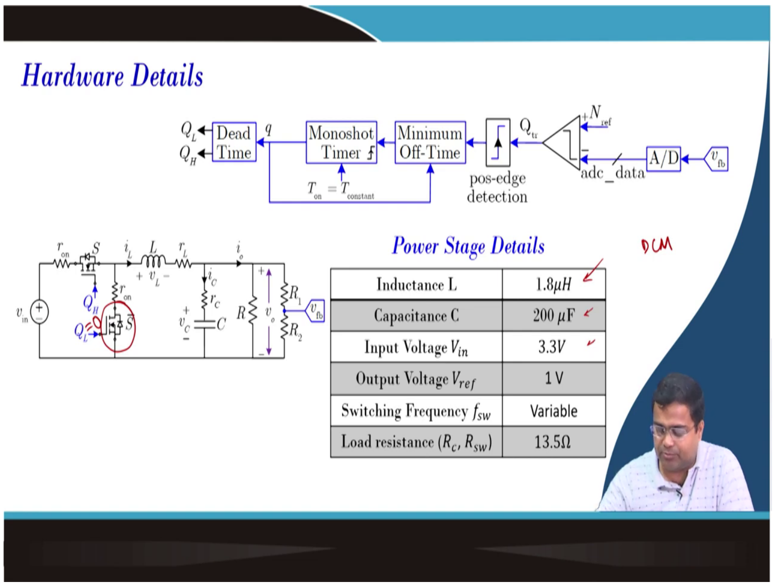
So, we will take, it because we want to go close to the analog comparator, but since we have a limitation in the clock we are using a clock. So, we will check each ADC clock data, data is updating whether the ADC data is smaller than N ref, if it is smaller; that means, the output voltage is smaller than the reference voltage then the status of the comparator goes low high; that means, this Qtr will be high this is my (Refer Time: 28:16).

So, this Qtr is nothing, but my comparator status. If it is above then it will be 0. So, now, since we are using you know only the not synchronous buck the conventional buck this output of the gate will simply go to the high side gate will be the high side gate and this will be the low side gate that will be set to 0.

And what is Q out? Now, after all these discussions you have to go to the module instantiation block. Where instantiated the module, yes. So, the Q out is the actual output of this counter. Monoshot timer which is a combination of on-time plus minimum off time and then what is that is Qout what is the enable it is the output of the clock manager. I told you that you have two or three edges one is the edge of the Qtr another is the edge of the f f min another is the edge of the f external.

So, all these combinations will go to the enable time. So, this will complete the implementation of this block that we have discussed ok.

(Refer Slide Time: 29:31)



Hardware Details

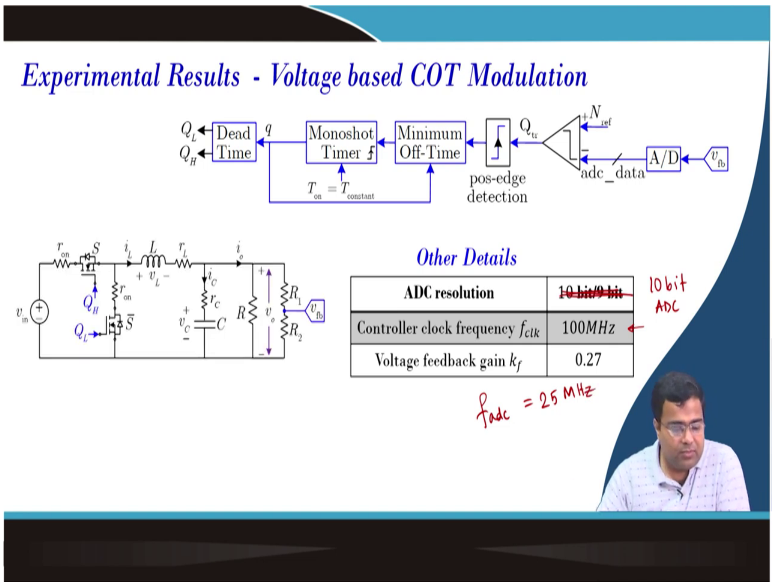
Power Stage Details

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

Now, we want to show the harder details. So, we are using the same converter that we have discussed 200 microfarad cap, 1.8 microhenry inductor, 3.3 input voltage, and 1-volt output the switching frequencies is varying because it is a constant on-time control.

And we are operating in DCM, Discontinuous Conduction Mode because we are replacing where QL is set to 0. So, the (Refer Time: 29.53) will come ok. And load resistance we are setting 13.5.

(Refer Slide Time: 30:04)



Experimental Results - Voltage based COT Modulation

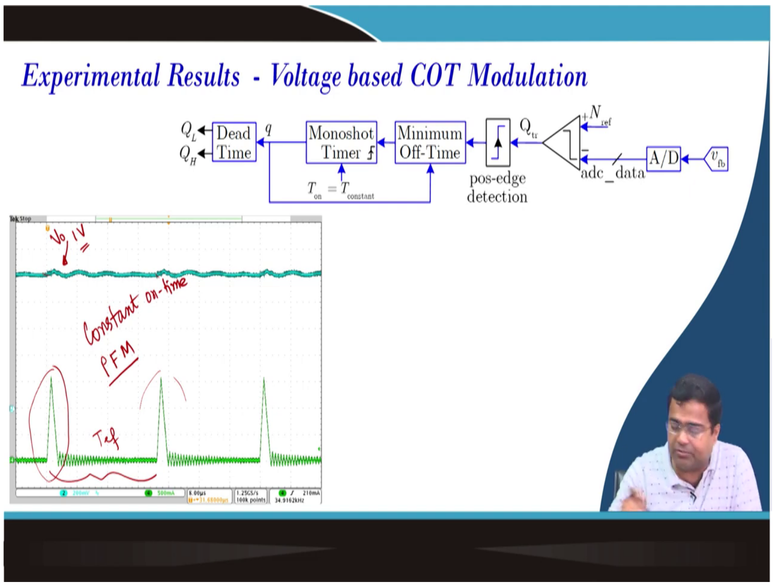
Other Details

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

$f_{adc} = 25$ MHz

Now, with this, we want to go for. So, we are using ADC. So, it is a 10-bit resolution, we are using 10-bit ADC and we are using a controller clock of 100 megahertz. But what is the ADC clock? We are using 25 megahertz ok.

(Refer Slide Time: 30:25)



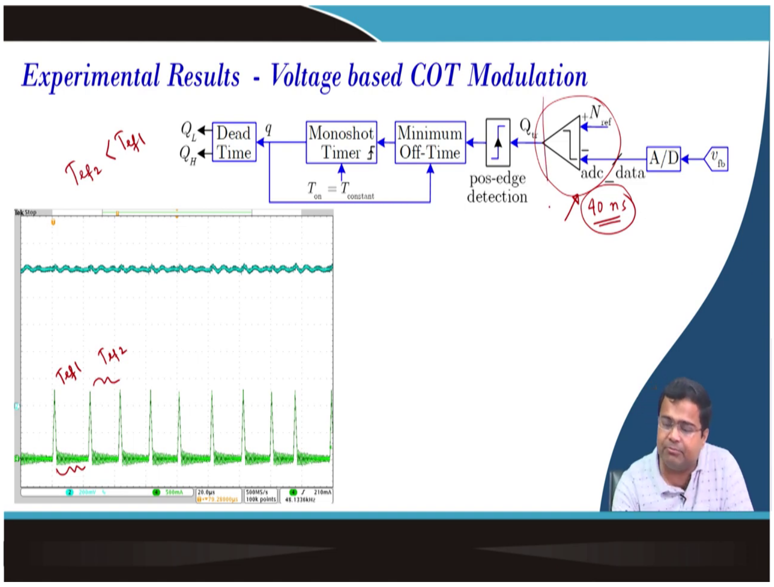
Experimental Results - Voltage based COT Modulation

Now, this is the experimental result. Where you can see this channel is the output voltage. So, his output voltage is 1 volt is a reference voltage this is my V0. You see when the output voltage falls below the reference voltage then this on-time is triggered and one of the on-time elapsed will have a minimum off time, but that is not coming here.

Then the voltage goes above and the voltage remains above again it comes below then again on time is triggered. So, this is your time period which is the effective time period and it is it is called pulse frequency modulation, because constant on time is a pulse frequency where the frequency varies with your know operating condition PFM.

So, it is a COT PFM constant on time, because there are many PFM techniques one is the constant on-time PFM, we can have a hysteresis PFM. So, there are different burst mode control is a different thing.

(Refer Slide Time: 31:33)



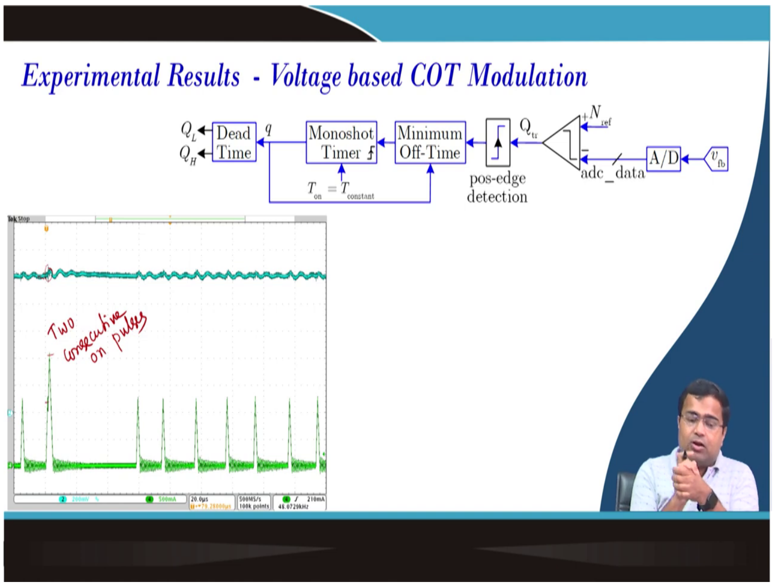
Experimental Results - Voltage based COT Modulation

But here. So, this is one of the waveforms. The next waveform I want to show is the operation, but you see this time period they are supposed to be the same and this time periods are slight.

So, if you take T effective 1, T effective 2. So, T effective 2 seems to be smaller than T effective 1. So, it slightly varies, why? Because we are not taking an analog comparator. So, that we do not have an infinite resolution of time. The time has a finite resolution because it is taken the data is taken after f t 40 nanosecond and during this 40 nanosecond, your output voltage actual output voltage might have crossed.

But we are not taking action, because the data is not available. As a result of this time quantization you are getting some jitter in the time period in the T effective, which will be avoided if you simply use an analog comparator.

(Refer Slide Time: 32:28)

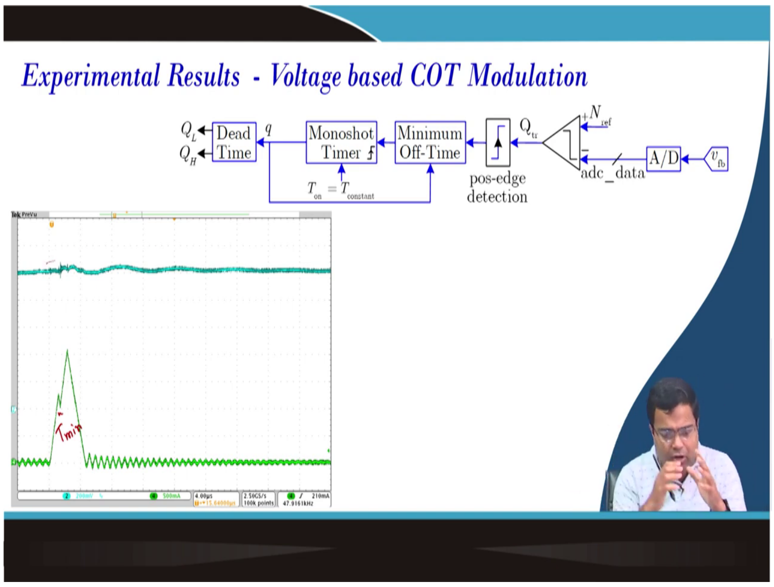


Experimental Results - Voltage based COT Modulation

Another thing, because of this time quantization it may. So, this happens when the on-time is low or the input voltage is low. Even if we apply a on time and your voltage is supposed to rise.

But there will be always some spike in the output voltage or there will be some output voltage min because it has gone down significantly down 1 on time and may not be enough to immediately take it up. So, then it is undergoing 2 consecutive pulses which is why it is double, but we also discussed we are putting a minimum off time. So, we need to zoom this part and see whether it is happening or not.

(Refer Slide Time: 33:23)

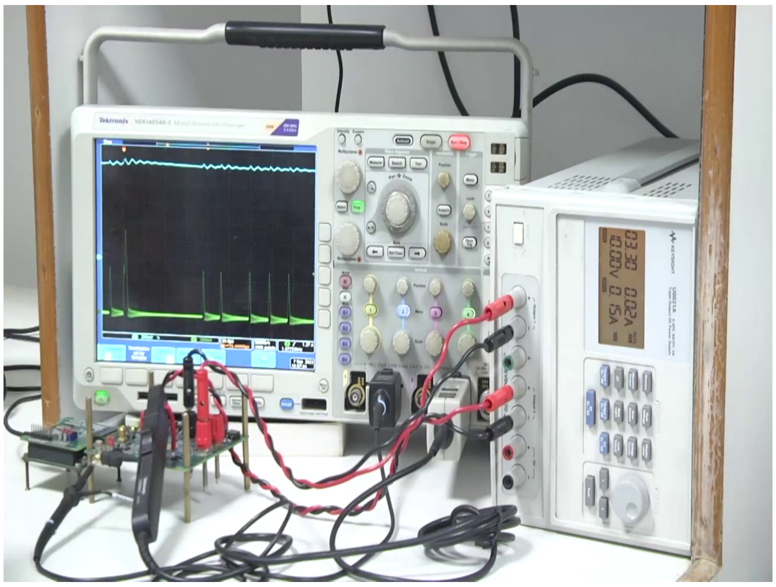


Yes, you can see there is a small portion of the time that we call T minimum, during this time the switch is turned off. So, whatever algorithm we have discussed is working, but this problem is coming due to the time quantization, but what can be done if we increase the input voltage for the fixed on time, your current will increase, because on time is the fixed input voltage increasing means the slope will increase and you may not find this consecutive on time.

But otherwise, this will increase the ripple, because of the continued on time the ripple may increase, but all these problems may be resolved if you use an analog comparator. Now, we are going to show the live demonstration of voltage-based constant on-time control which is typically used for light load discontinuous conduction mode where you know we are trying to improve the efficiency. So, it is also knowns known as pulse frequency modulation.

As where we have discussed that the switching frequency will linearly vary with you know the load current. So, in this we have already run the code we have synthesized and then we have generated the program file and we have loaded the FPGA file dump into the FPGA, I am not showing going to show this step because this is already we have done and we have shown this step in multiple lectures.

(Refer Slide Time: 34:52)

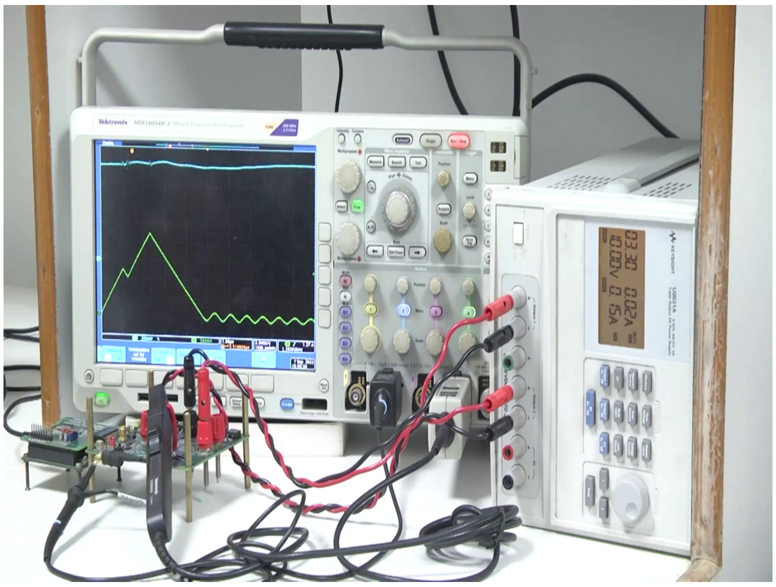


Now, we want to start; that means, first the supply. So, we want to start with let us say 3.3 volt which is the sort of nominal voltage that we want to design nominal voltage. So, this is 3.3 volt. So, if you see this scope first of all you will find 2 things here. If we do a single shot.

So, there are in some cases we are getting pulses which 2 consecutive pulses because you know this depends on the logic, because what is the logic here? When the output voltage goes below Vref then we trigger a monoshot timer and then it gives a min on time fix on time we have discussed this in this lecture what is the duration of on time?

But, now since the input voltage is at 3.3 volt actually if you zoom this portion. You know maybe we can continue. So, if we take this position here I just want to show the zoom version.

(Refer Slide Time: 35:57)



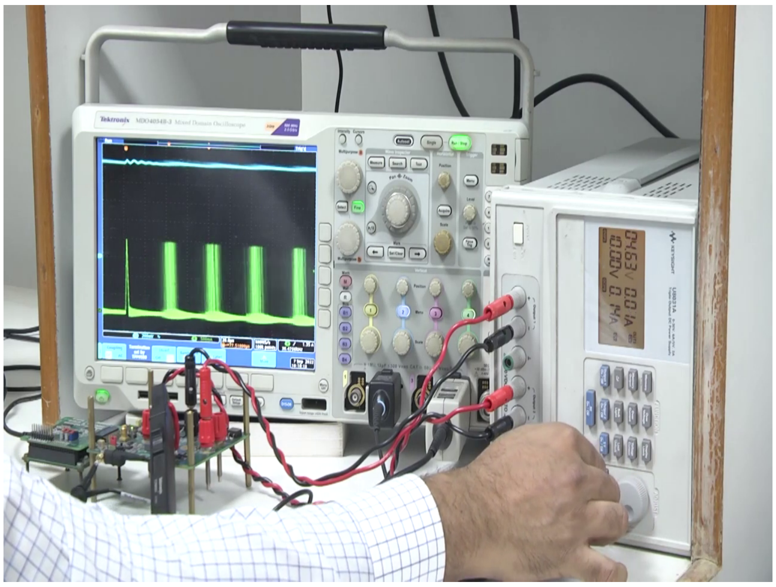
So, I want to show the zoom version of this, what is happening here you can. So, can see there are 2 consecutive pulses and in between, there is a minimum of time, because we have discussed; that means if we take this waveform.

So, first, there is an on time, and what is on time? It is a 2 microsecond on time is 1 microsecond. So, on time 1 microsecond, but after this one time, it seem that the output voltage was not even above the reference voltage. So, it again hit the lower limit and we have discussed we are not taking an analog comparator. We are taking an ADC which is running at 25 megahertz clock.

So, naturally, there is a time quantization effect; which means, the sampling effect; that means, we are capturing the sample which shows that it is below reference. That is why it is triggered it is triggering another you know on pulses. And in between we have given a minimum off time which is I think 0.1, 100 nanosecond, or maybe 200 nanosecond, but I mean if you zoom this further we may figure out what is the minimum duration.

So, the minimum duration seems to be you know around 260 nanosecond. That is the minimum duration ok, but the first time your switch turns on. Now since there are two on the duty on pulses, as a result, it has injected a large number of energy huge energy, as a result, you will find if you run again there is a sufficient gap between the next pulses ok. So, as a result, you will find this, but if we do not select this suppose you select this.

(Refer to Slide Time: 37:47)



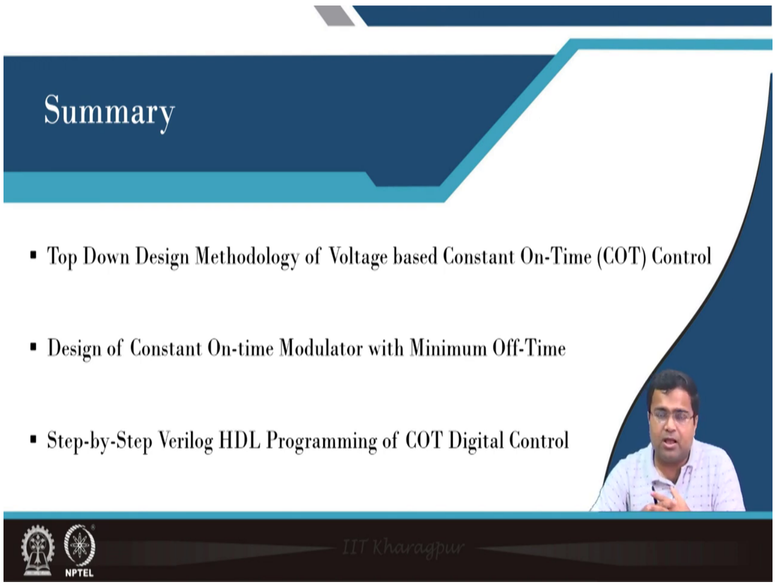
Then you will find the regular operation, where it should show PFM; that means, it should show a fixed frequency and it should repeat, but we have discussed that since there is a time sampling time which is 40 nanoseconds. So, in time resolution the voltage there is a mismatch between the because your effective time period is not the integral multiple of 40 nanosecond.

As a result, there is always some quantization effect and there is a (Refer Time: 38.14) effect that is coming here, but this large spike that is coming there actually is still there. Now, if we increase now if we increase this input voltage; that means, it is 3.3 volt we are increasing. You see when the pulse is sufficiently large the occurrence that high pulses occurrence of such high pulses is now going down.

I mean these pulses are not coming and you are getting more or less PFM operation, but what is happening is the peak current has increased, because you’re on-time is fixed and the input voltage is increasing. As a result, the slope of the current is increasing and as a result, the current ripple is increasing.

So, it may increase the voltage ripple also. So, that is one of the problems with on time if we use a timer based on time. And when you move on I will demonstrate in another lecture adaptive on time where this can be taken care of, but at this, in this lecture, I am going to show that you are getting a pulse frequency operation under light load condition.

(Refer Slide Time: 39:22)



Summary

- Top Down Design Methodology of Voltage based Constant On-Time (COT) Control
- Design of Constant On-time Modulator with Minimum Off-Time
- Step-by-Step Verilog HDL Programming of COT Digital Control

IIT Kharagpur
NPTEL

So, in summary, we have discussed the top-down design methodology of voltage base constant on-time control. We have discussed this design of a constant on-time modulator with minimum off time. And we have discussed step-by-step Verilog HDL programming of constant on-time digital control and we have also shown some experimental case studies.

So, in the subsequent lecture, we want to consider an experimental case study or DCM under DCM with constant time with PSM Pulse Skipping Modulation as well as PWM. So, these things will be compared in the subsequent lecture. That is it for today.

Thank you very much.