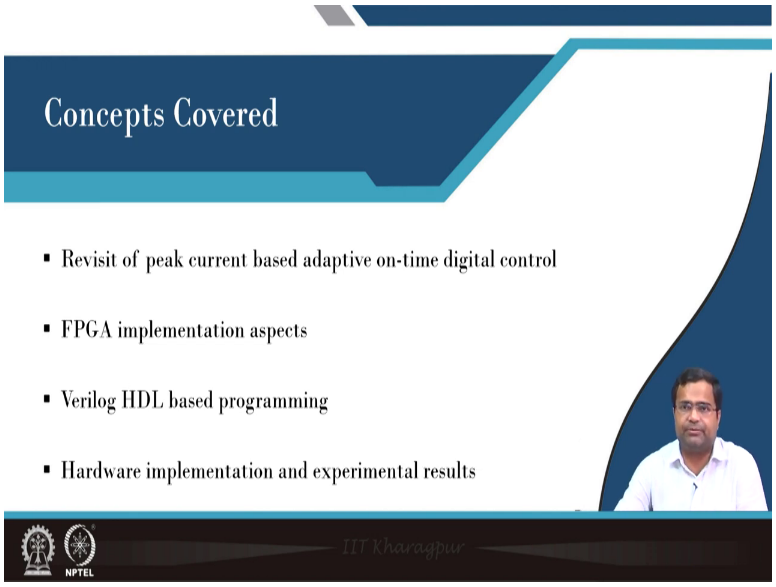
Digital Control in Switched Mode Power Converters and FPGA-based Prototyping

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 - 100
Adaptive On-Time Digital Control in DCM with Verilog HDL Implementation

Welcome. In this lecture, we are going to talk about Adaptive On-Time Digital Control for Discontinuous Conduction Mode and we want to show Verilog HDL Based Implementation.

(Refer Slide Time: 00:37)

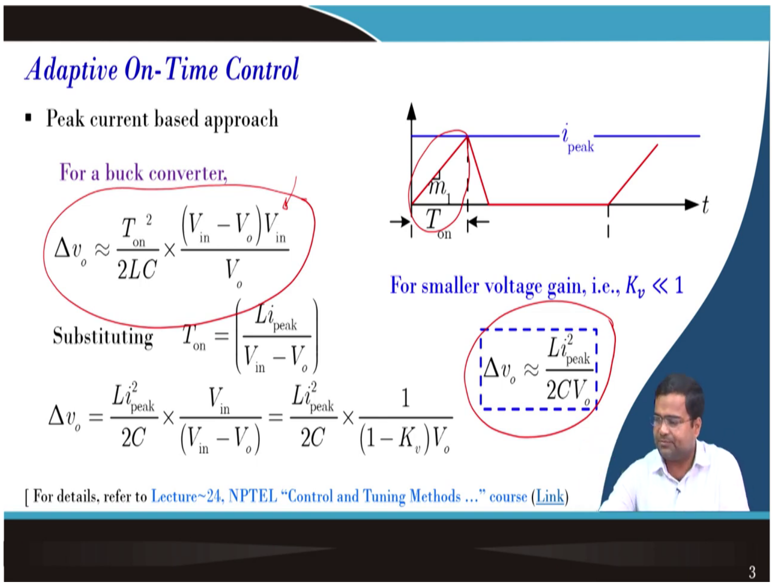


Concepts Covered

- Revisit of peak current based adaptive on-time digital control
- FPGA implementation aspects
- Verilog HDL based programming
- Hardware implementation and experimental results

So, we will first talk about we want to revisit the peak current based adaptive on time control. Then FPGA implementation aspect, Verilog HDL-based programming, and finally, hardware implementation and experimental results.

(Refer Slide Time: 00:50)



Adaptive On-Time Control

- Peak current based approach

For a buck converter,

$$\Delta v_o \approx \frac{T_{on}^{\ 2}}{2LC} \times \frac{(V_{in} - V_o)V_{in}}{V_o}$$

Substituting $T_{on} = \left(\frac{Li_{peak}}{V_{in} - V_o}\right)$

$$\Delta v_o = \frac{Li_{peak}^2}{2C} \times \frac{V_{in}}{(V_{in} - V_o)} = \frac{Li_{peak}^2}{2C} \times \frac{1}{(1 - K_v)V_o}$$

For smaller voltage gain, i.e., $K_v \ll 1$

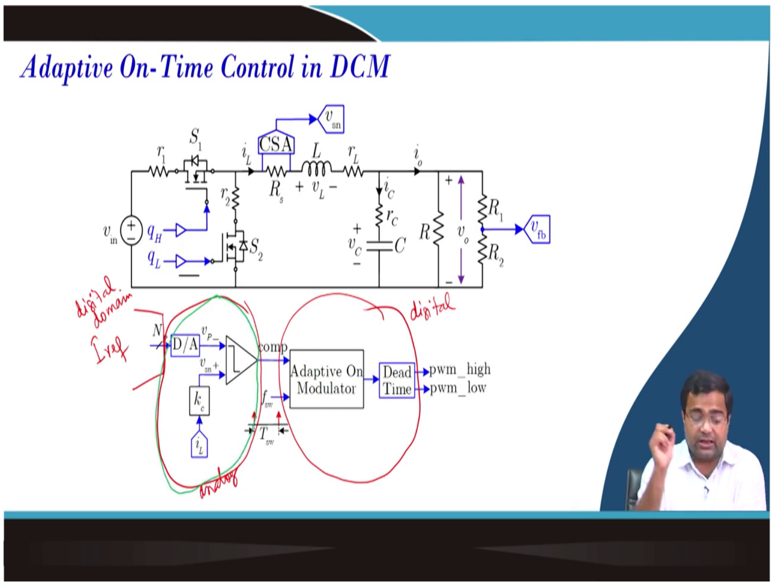
$$\boxed{\Delta v_o \approx \frac{Li_{peak}^2}{2CV_o}}$$

[For details, refer to Lecture~24, NPTEL "Control and Tuning Methods ..." course (Link)

So, if we take the adaptive on time this thing we have discussed in the previous lecture and here we want to retain the same feature of constant on time where you know the ripple voltage will be independent of load current, but it is still a function of input voltage and we discussed in the last lecture also. If the input voltage increases ripple will violate ripple will increase. So, we want to make sure the ripple will be more or less insensitive to input voltage also.

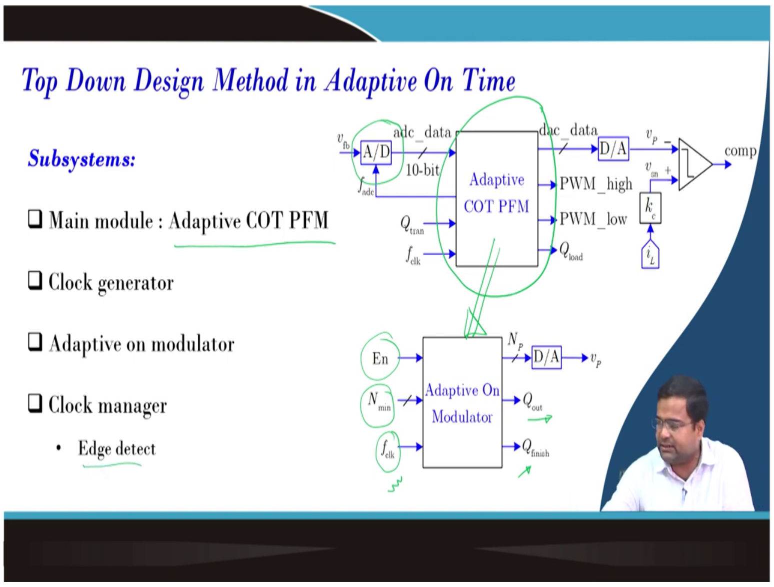
So, if you replace here we are not giving any timing constant from a timer rather we are using a peak current-based constant on time. And if you replace and we have already discussed in lecture number 24 in detail. It can be shown that the output voltage ripple will be independent of the input voltage and it is already independent of the load current. So, this will retain all the good features of constant on time at the same time it can be made (Refer Time: 01:41) of ripple can be made insensitive to the input voltage.

(Refer Slide Time: 01:45)



And now we are going to talk about adaptive on-time control, and how to implement using FPGA. So, here what we are talking about in this part is the current reference. So, I will say this is the current reference; it is coming from the digital domain. And this is your deck this part is in analog this part is in analog. Again this part is digital. So, just to demarcate the analog part we will use a different color for this part in analog ok.

(Refer Slide Time: 02:33)


Top Down Design Method in Adaptive On Time

Subsystems:

- Main module : Adaptive COT PFM
- Clock generator
- Adaptive on modulator
- Clock manager
  - Edge detect

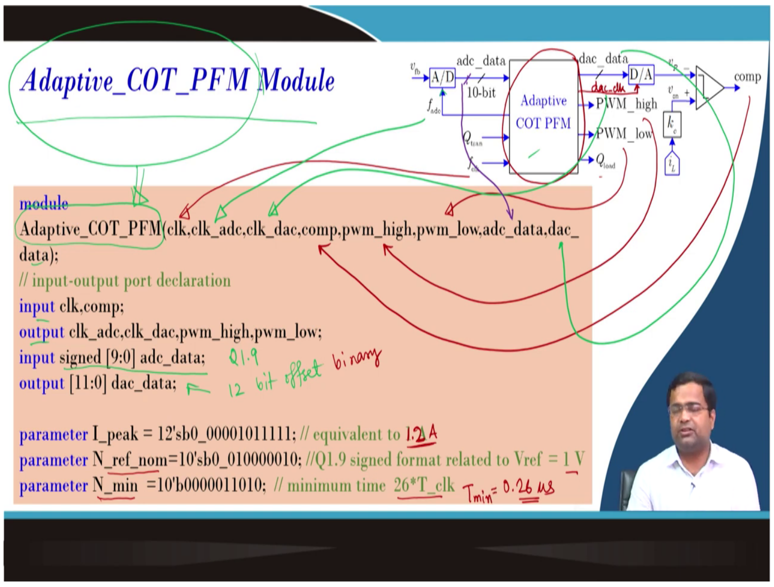
Now, that is why we want to implement this logic and we have discussed that we need A to D converter for voltage comparison. In fact, in the commercial product, we can simply use an analog comparator, with no issue; that means, we want the output voltage how it is compared with the reference voltage and when the output voltage falls below Vref then we will get Qtr.

So, in that case, you may not need an A-to-D converter. And we will be discussing this in the last lecture. Some of this architecture can have purely analog implementation and the other blocks' timing circuit digital data can be processed digitally. And that way you can reduce the cost. Here, inside this block, there will be a minimum on-time adaptation modulator that requires an enable block.

Minimum on-time minimum off time sorry then this clock will be used to count that the L all timing calculation will be done based on this high-frequency clock. Then it will generate a Quote and we will show how Qout can be mapped to a high pulse in DCM it can be done straight away sent no dead time circuit is needed. And then Qfinish is the time when the on-time adaptation; that means, on-time is elapsed followed by the minimum off time elapse.

Then it is again it can accept another trigger pulse on time if it is required. So, it consists of a main module which is called Adaptive COT PFM then a clock generator circuit adaptive on time modulator, and a clock manager. So, how does it work? And the clock manager inside there is another module is an edge detection circuit.

(Refer Slide Time: 04:14)



Adaptive_COT_PFM Module

```
module
Adaptive_COT_PFM(clk,clk_adc,clk_dac,comp,pwm_high,pwm_low,adc_data,dac_
data);
// input-output port declaration
input clk,comp;
output clk_adc,clk_dac,pwm_high,pwm_low;
input signed [9:0] adc_data;
output [11:0] dac_data;

parameter I_peak = 12'sb0_00001011111; // equivalent to 1.2 A
parameter N_ref_nom=10'sb0_010000010; //Q1.9 signed format related to Vref = 1 V
parameter N_min  =10'b0000011010;  // minimum time  26*T_clk
```

So, if you go inside. So, Adaptive COT PFM is the main module. The name of the module and you can see the module name is coming here.

So; that means, this is the module name ok. So, this is the module name of any module that is the interface of this module that has to communicate with real-world data. And we need to figure this out because this can only accept digital IO data. So, naturally, there is an ADC. So, first of all, this is a high-frequency clock which is here then Qload which is going out here we are not giving any load transient. So, we are setting we are disconnecting Qload because we are not making any load transient.

But, when you go to the multi-mode we will show load transient. Then here Qlow this is your PWM Qlow then Qhigh is here PWM high and the analog comparator is here. You can see this is a comparator walk then you also have clock data. That will be here the f DAC clock. So, which will be coming to this DAC clock is here clock dac ok. Then there is a clock ADC which is here clock ADC is here then ADC data. So, where is the ADC data if we take another?

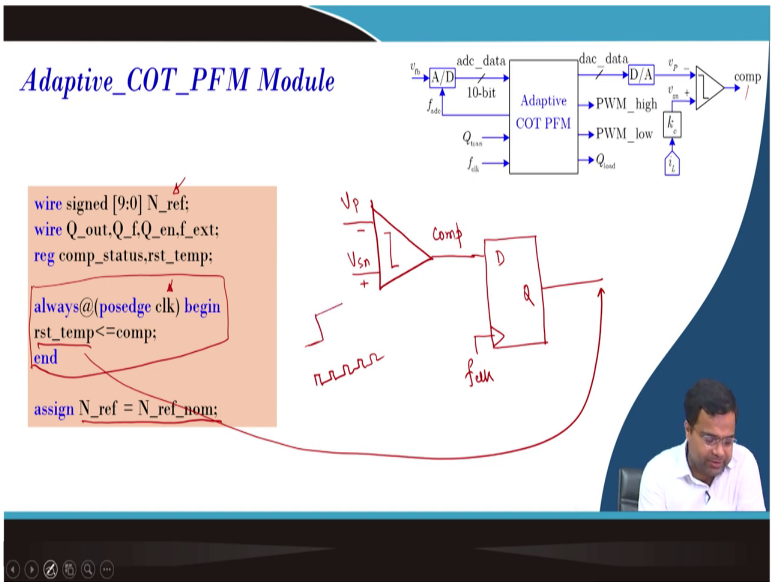
So, this is the data vector data this is coming here ADC data and the DAC data which is here you can see this data is here sorry DAC data. So, maybe I can use green color. So, this is the DAC data. The Hub part is going here and here it is a continuation. So; that means, we have to interface all these real-world signals which are why it is the main module. And we have also discussed whether it requires an ucf file user constant file or the implementation file and I am not going to show, it because we have already discussed the procedure of running the ucf file and so on.

We want to synthesize the internal module which will be a prototype using FPGA. So, we have to declare which are the input-output and from this picture, it is clear from the direction it is clear which is going out which is coming out that is the output which is going in that is an in, and input also we have a vector data that is an ADC data signed and the output though this is in Q1 dot 9 format.

And this is the 12-bit, 12-bit offset binary. We have discussed already our ADC is offset by our DAC and accepts offset binary data. Then sorry this is offset binary then here we are setting a peak current reference which is equivalent to 2, but we later found it is 1.2 ampere which is roughly around 1.2 ampere yeah. Closely and it can vary just because there are some other delays and all. So, the target is 1.2 ampere then we have a reference nominal voltage which is set to 1 volt.

But if we and we also set a timer that will generate the minimum off time; that means, what is the minimum off time? Because it is a constant on-time moderator. So, it also has a minimum off time. And here we are setting 26 into 10 nanosecond 260 nanosecond or it is 0.26 microsecond that is your Toff, Tmin which is the minimum off time for this constant on time.

(Refer Slide Time: 08:10)



Adaptive_COT_PFM Module

```
wire signed [9:0] N_ref;
wire Q_out,Q_f,Q_en,f_ext;
reg comp_status,rst_temp;

always@(posedge clk) begin
rst_temp<=comp;
end

assign N_ref = N_ref_nom;
```

Now, we are defining. Here the reference voltage we are using a muxing because you can make a load transient, but here we are simply taking the reference voltage to be the nominal value ok. This one is your research comparator; that means, you see that at the clock edge, yeah.

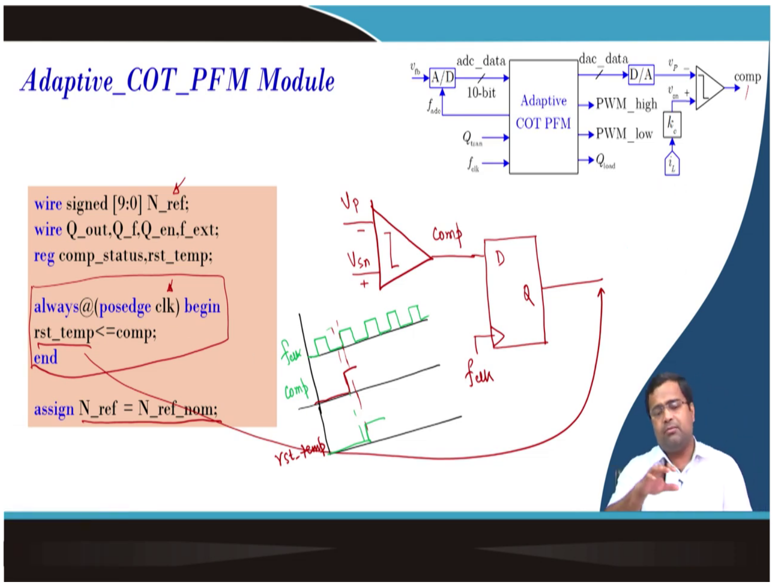
So, this is another issue, because we face some difficulty here. If we take the direct comparator output if you go to the main module this comparator output was used in multiple places inside this block. So, it was getting loaded; that means if you take that signal and this is an IO pin it may not have sufficient current driving capability.

And if we use that pin to take derive multiple circuits using that pin then this pin may not have sufficient current to drive, because if you use just comparator 1 or 2 blocks to trigger that is fine, but this comp will be used multiple times inside. So, there are 2 possibilities we can put a buffer circuit. Additional buffer circuit or here we have used a clock synchronized comparator; that means if we have an analog comparator we are actually that analog comparator.

Let us say this is comparing with peak current with Vsense and you can see it is minus and this is an analog comparator. This will be our comp, now this comp we are using like a D flip flop and this is our clock and this is our high-frequency clock and this is Q; that means, the output will be comp Q output will be comp which is the reset temp.

So, here we are using reset temp this will be reset temp only when the clock edge comes; that means, it will go inside this block whenever the clock has come otherwise it will not take, and since the frequency is very high; that means, this is a very high-frequency clock this clock. So, if the comp changes let us say the comp changes somewhere here ok.

(Refer Slide Time: 10:29)



Adaptive_COT_PFM Module

```
wire signed [9:0] N_ref;
wire Q_out,Q_f,Q_en,f_ext;
reg comp_status,rst_temp;

always@(posedge clk) begin
rst_temp<=comp;
end

assign N_ref = N_ref_nom;
```

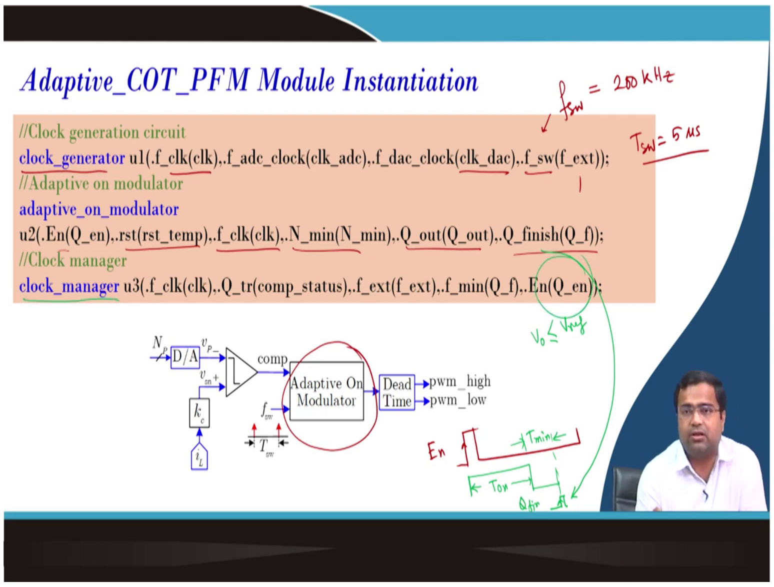
So, let me draw, how does it matter? So, suppose I draw a timing diagram like this. This is the timing diagram. So, let us say we have this high-frequency clock. So, this is our high-frequency clock. So, this is our f CLK and this is our comp which is a comparator output. Let us say comparator output goes high sorry somewhere here it goes high somewhere here. So, we cannot turn it on immediately.

So, we have to wait for this clock edge to come; that means, whatever you are using rst comp; that means, reset comp or basically clock say sorry rst temp the name here whatever signal we arising here this one; that means, rst temp some name is given. If you draw this particular clock; that means, if you continue. It will be turned on when it will be turned on it will turn on here. So; that means, there is a delay and this delay maximum can be up to Tclock because there is a time period.

So, this is a comparator operation, but the clock synchronizes. So, due to this delay, because it is a clock-synchronized comparator I would say it is a clock-synchronized comparator. So, due to this delay sometime your peak current can exceed the reference voltage because you are not turning off the switch rather due to this clock synchronization operation you are taking action after some time.

So, this may be a possibility. So, it may exceed the actual constant on time, but there is n number of ways to tackle this and you know to solve this buffering problem, but I am not going to discuss this because it depends on the user.

(Refer Slide Time: 12:14)



Adaptive_COT_PFM Module Instantiation

```
//Clock generation circuit
clock_generator u1(.f_clk(clk),.f_adc_clock(clk_adc),.f_dac_clock(clk_dac),.f_sw(f_ext));
//Adaptive on modulator
adaptive_on_modulator
u2(.En(Q_en),.rst(rst_temp),.f_clk(clk),.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));
```

Then we are using we are recalling calling one the clock generator block and this we have discussed though we are not using the DAC clock, it is still there this is the highest resolution clock and this was the switching frequency clock that we have used for PWM. And this clock the frequency we are talking like you know like a 200 kilohertz.

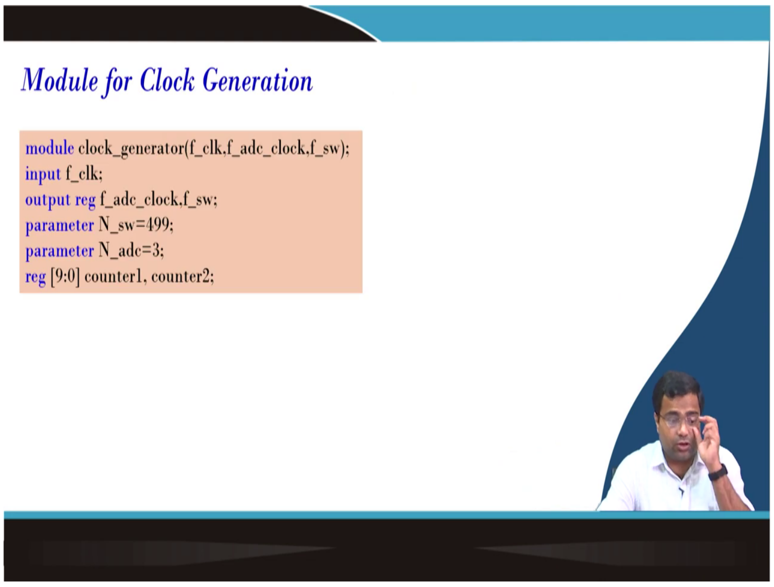
That means after every; that means, what is the time period it is 5 microsecond; that means, after 5 microsecond this edge will come and it will check whether your Qtr is high or low if the Qtr is high and mono short timer is still not activated, it will simply turn on ok. So, that is why the external signal. Adaptive on-time modulators require an enable signal because we have discussed adaptive on-time inside it requires to enable and it also requires a reset that is a comparator output which is a clock synchronization in this case.

And the high-frequency clock is used to generate the minimum off time using this timer and then the Qout which is the output this output consists of the on-time followed by the minimum off time and then this Qfinish will be enabled; that means if an edge comes which is let us say I would say the if the enabled signal goes high here. If an edge comes then what will happen your constant on-time will be activated ok.

So, this is our Ton which will be coming from the current loop and there will be minimum off time; that means, this is the time which will be the Minimum time it will not be on even though any pulse comes because it has a minimum off time then it will check here. So, here you are our Qfinish; that means, whatever clock we are talking about it is this clock.

So, whenever this minimum off time that timing is elapsed then it will generate this, and because this is to check whether your Qtr is still high; that means, whether your output voltage is still below the Vref then it may trigger another on time. And then this is the clock manager who will generate this enable clock and we are going to discuss.

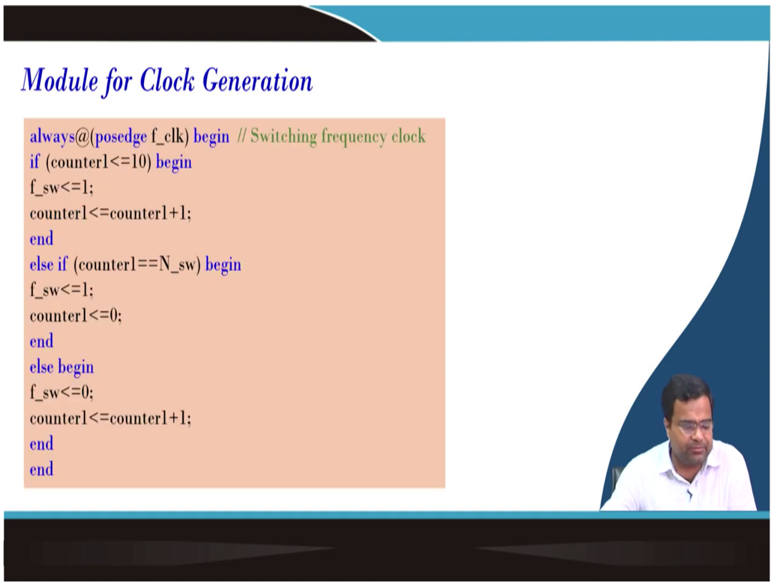
(Refer Slide Time: 14:35)



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;
```

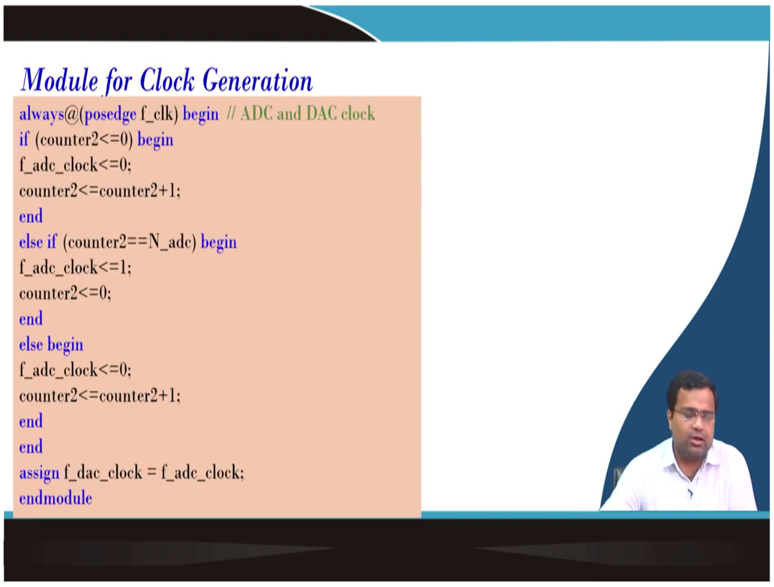
(Refer Slide Time: 14:42)



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
```

(Refer Slide Time: 14:44)

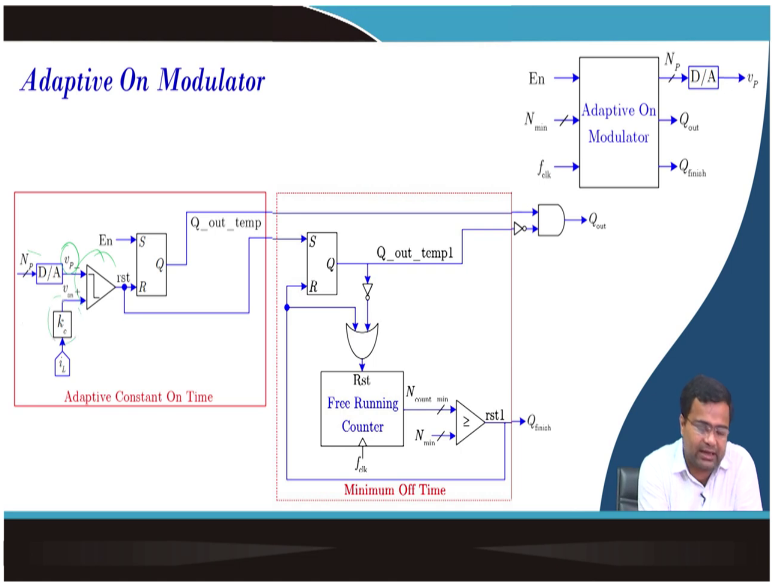


Module for Clock Generation

```
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
assign f_dac_clock = f_adc_clock;
endmodule
```

So, the clock generation block we have discussed multiple times. So, I am not going to discuss this to generate a switching frequency clock. As well as the ADC clock and we have discussed lecture numbers 74, and 75 in the context of voltage mode control the same module is used.

(Refer Slide Time: 14:51)

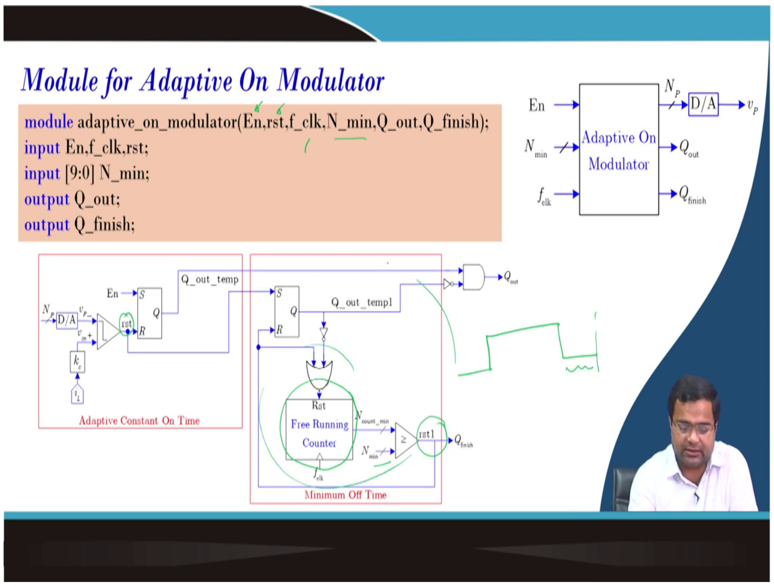


Adaptive On Modulator

Now, adaptive one-time modulator. Here we have already discussed this in the previous lecture; that means, we have this peak current reference here which is coming from the DAC. Analog current reference is an analog comparator and the sense current.

Then this will create a reset pulse and this is coming inside; that means, this comparator output is there and this will be one of the; it is reset pulses to turn off the monoshot timer.

(Refer Slide Time: 15:24)



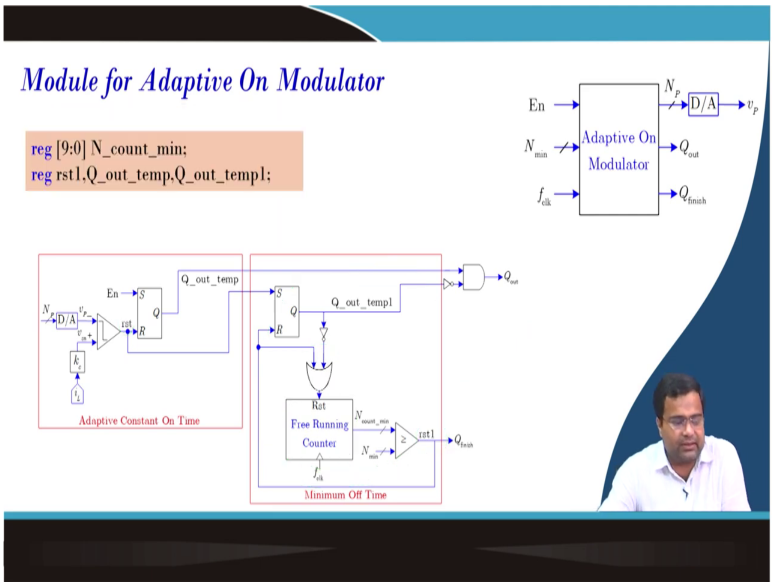
Module for Adaptive On Modulator

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

And how is the module? So, this module accepts the enable pulse which is the trigger pulse reset which is nothing, but this reset is a comparator output, but here we are using a clock synchronize comparator then f clock is needed to generate this for the free running counter high-frequency clock.

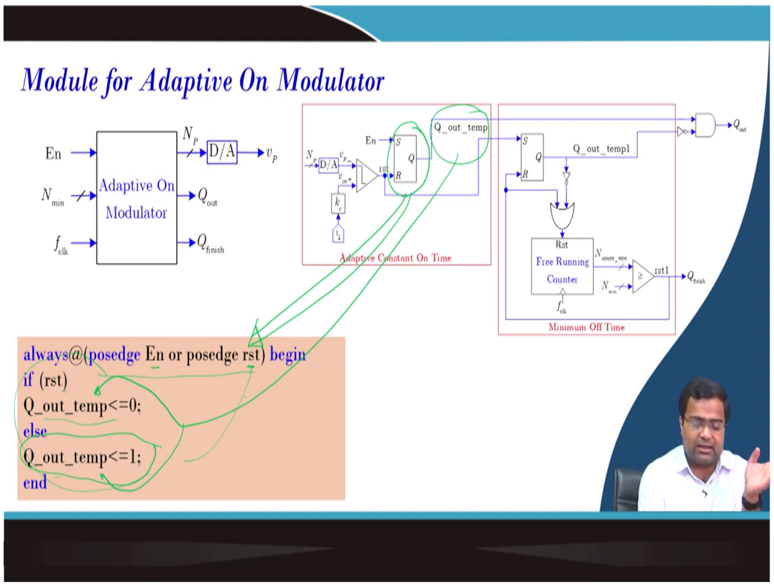
And this is a minimum quantity it will generate the reset pulse which will enable it because we discussed that if the onsite timer is on then it will be minimum on time. So, this duration to generate this is the circuit the whole circuit.

(Refer Slide Time: 15:59)


Module for Adaptive On Modulator

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

(Refer Slide Time: 16:01)

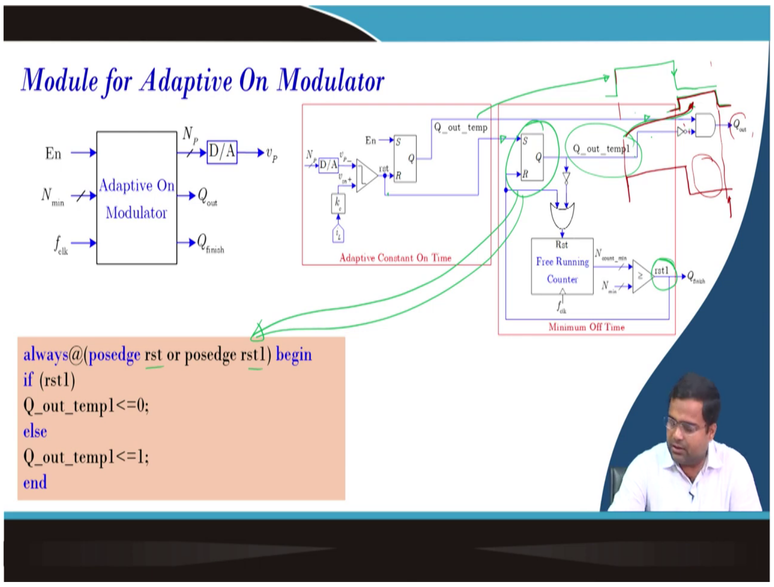

Module for Adaptive On Modulator

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

This thing we have discussed in detail then this resistor then if you this is our this particular block. If you see this block is this always a block? So, it will check at every edge of enable as well as the reset, whichever comes this line will be executed this whole portion will be executed. So, it will first check when the reset is high then the output will be this output will be 0. So, this output will be here and these are the 2 output right, but if the reset is low, it can only enter if the enable edge come or the reset edge come.

If the reset edge comes, it will see reset high it will be 0. If the enabling edge comes and that time if the reset is 0 then it will take 1. So, that is why if the reset is 0 and if the enabling edge comes multiple times this will not this will remain at 1. So, it will not change the status. As long as the reset comes and this is a reset-dominated pulse because if the reset comes then nothing can happen it has to reset ok.

(Refer Slide Time: 17:01)


Module for Adaptive On Modulator

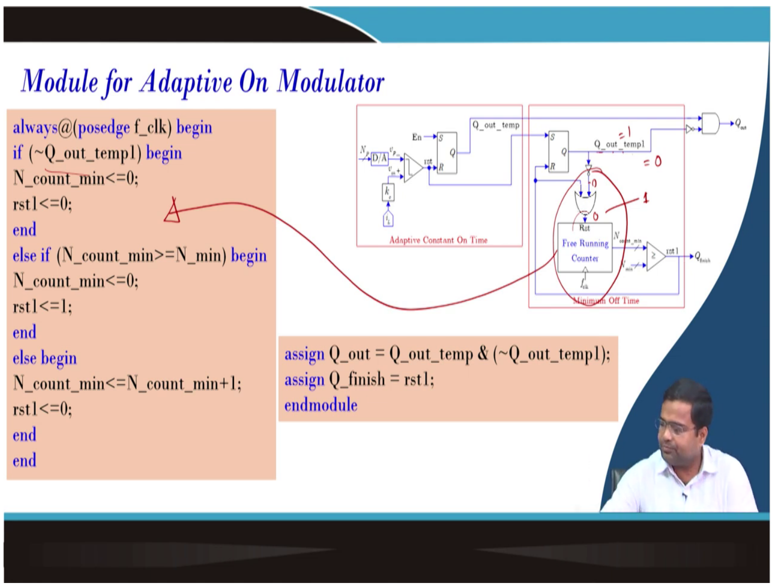
```
always@(posedge rst or posedge rst1) begin
if (rst1)
Q_out_temp1<=0;
else
Q_out_temp1<=1;
end
```

Now, this is the next one which will be this one. So, this is here. You see it has two inputs pulses reset and reset 1. This is the reset of this pulse and this is reset 1 for this pulse. Now, again, in this case, the reset one is dominated, if reset 1 is high this output will be 0. If this is this will be 1. And we are making a lot operation of this. So, that making sure whenever this guy is 0 then will be 1. So, this will pass that on when this guy is like sorry when this guy is 1 this will be 0.

So, it will make because there is a knot operation here. Just a minute reset ok. So, this is the combination of Q1 and Q2 yes, because when this is already high then this will not start this will start at this edge. So; that means, this is our this particular Q temp let us say, and what will be this? This will be high for this time this is this pulse ok. So, maybe I can use a different color this is this pulse this is a minimum off time.

So, during this time when this is high then this will be low and this will be whatever the signal is the N. So, it will become low so; that means, the net output will be if we take the net output this will high this will be low and again now this is released you can turn it on again if it is required, but till this time it cannot turn on there is a minimum off time ok.

(Refer Slide Time: 18:52)



Module for Adaptive On 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
```

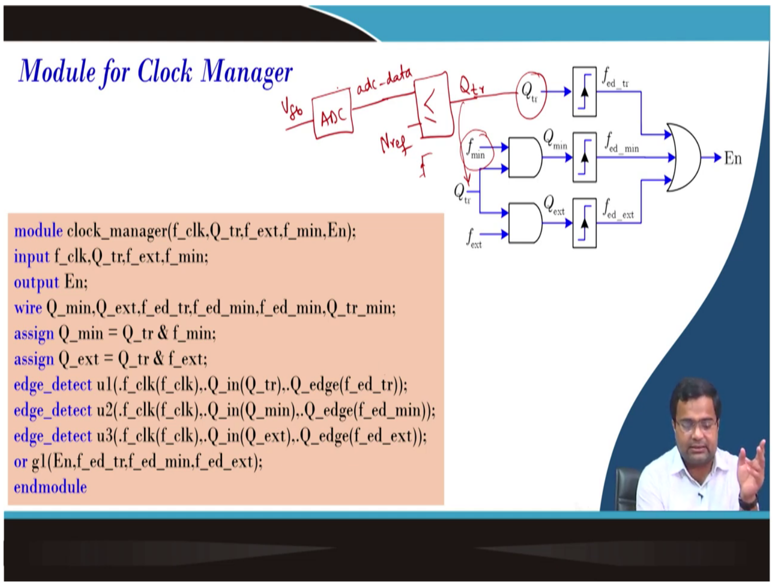
```
assign Q_out = Q_out_temp & (~Q_out_temp1);
assign Q_finish = rst1;
endmodule
```

If this logic is enabled and disabled edge clock edge will check.

So, this is this particular block it is implemented here ok. So, you can say whenever this is this is low the reset; that means if this is high then only the counter will be incremented if it is low then the counter. So, if this is high then this will be low the reset will be deactivated if this is high the reset will be deactivated, because it is you know active high reset signal. So, if it is high then this will be 0.

So, this will be 0 if it is 1 then this will be 0. So, reset is disabled and it will start counting it will count, but when this is 0, let us say this is 0. In that case, this will be 1, and then reset will be activated and as a result, you know the circuit will be disabled completely ok. So, this is the and this Qfinish is coming when your off time minimum off time is over.

(Refer Slide Time: 20:03)



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
```

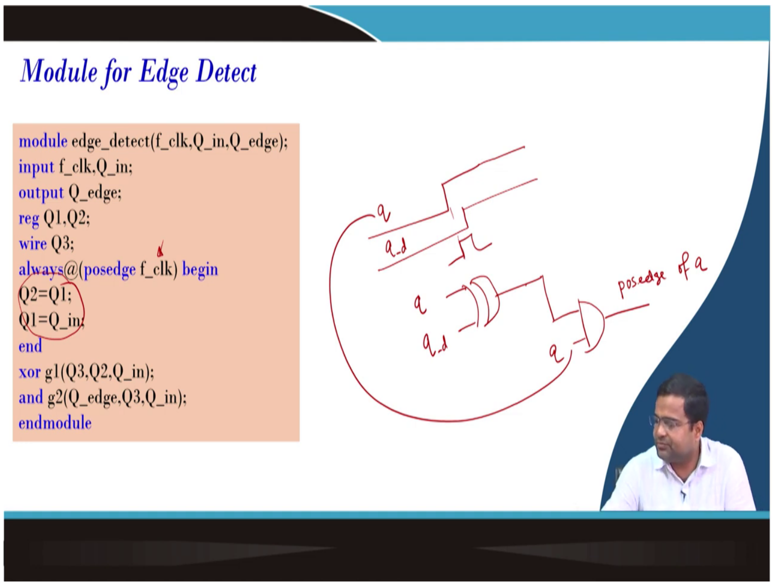
The clock manager will simply take Qtr. What is Qtr? We have discussed that we have ADC, which is your feedback voltage and this is your ADC data, this is compared with our Nerf, and if it is less than equal to high. So, this is your Qtr and this Qtr is coming from here and this signal is used here also multiple times, and this is all.

If Qtr is high then whenever Qtr becomes high that is required to turn on the monoshot timer or if it is high, but no edge is coming then it will check the minimum on f minimum will come that will come from the edge detection circuit and you see the minimum because we will pass the Qtr detect that this.

So, Qmin, what is Qmin? It is a combination of Qtr and fmin and how to generate f min that we have discussed in the if you go to this block. So, fmin will be the min when your module instantiated, yeah. So, fmin if you see fmin it is the Qf; that means when this monoshot timer on time is over and off time is minimum is over then this of will be enabled. So; that means, this fmin corresponds to the flag which shows that minimum on time is elapsed. And if that time Qtr is high it will generate the enabled trigger.

So, the enabled trigger is the combination of all these trigger pulses and it can come multiple times when the switch is already on it will not change the status, but even if this pulse comes during minimum off time it cannot turn on the counter ok.

(Refer Slide Time: 21:54)



Module for Edge Detect

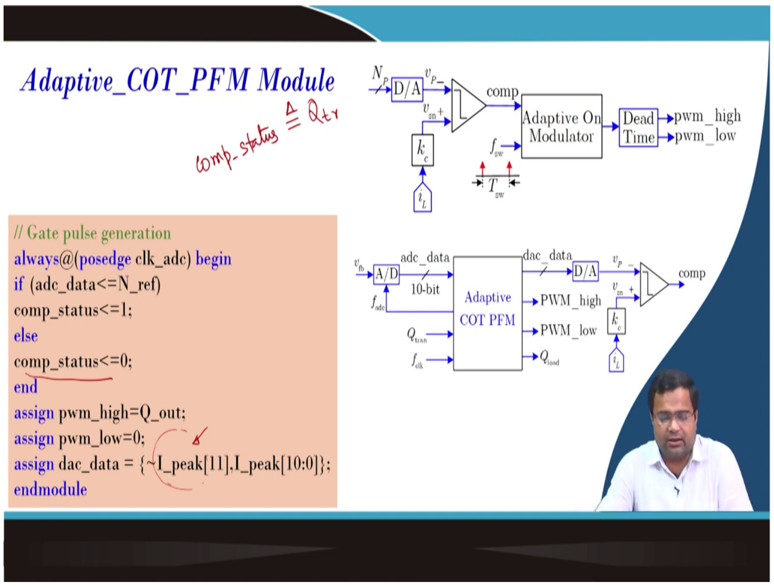
```
module edge_detect(f_clk,Q_in,Q_edge);
input f_clk,Q_in;
output Q_edge;
reg Q1,Q2;
wire Q3;
always@(posedge f_clk) begin
Q2=Q1;
Q1=Q_in;
end
xor g1(Q3,Q2,Q_in);
and g2(Q_edge,Q3,Q_in);
endmodule
```

So, the module of edge detection is simple. Edge detection is what we are doing whatever signal we are getting you to know we are talking about the positive edge direction.

So, we are delayed by another signal. So, this is yours let us say one particular signal q this is q delayed and if we do xor; that means, q and q delay sorry if we do q and q delay if we do xor of this then this will be high and that is ended with this particular signal. That is your q and this is your positive edge pos edge of q ok. So, this is a circuit and here we are making 2 unit delay because there is a blocking statement 2-unit delay.

Because it depends on the f clock which is a high-frequency clock of 10 nanoseconds. So, you are making a 20-nanosecond delay.

(Refer Slide Time: 22:54)



Adaptive_COT_PFM Module

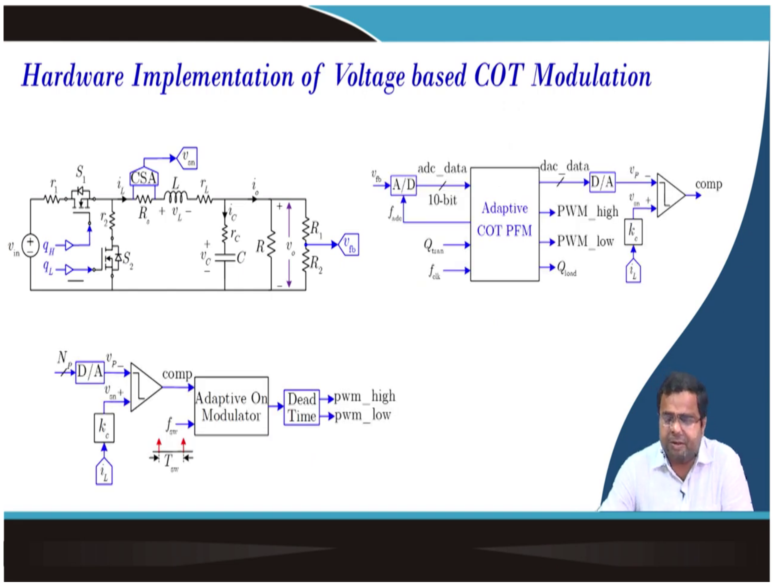
comp_status $\triangleq$ Q_{tr}

```
// 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;
assign dac_data = {~I_peak[11],I_peak[10:0]};
endmodule
```

Then adaptive again this is the logic I told you this will generate comparator status which is our Qtr; that means, we name here comparator status that is our equivalent to what we are using Qtr ok, because if you go back to the main module where we are instantiated you see the Qtr is nothing, but comparator status ok.

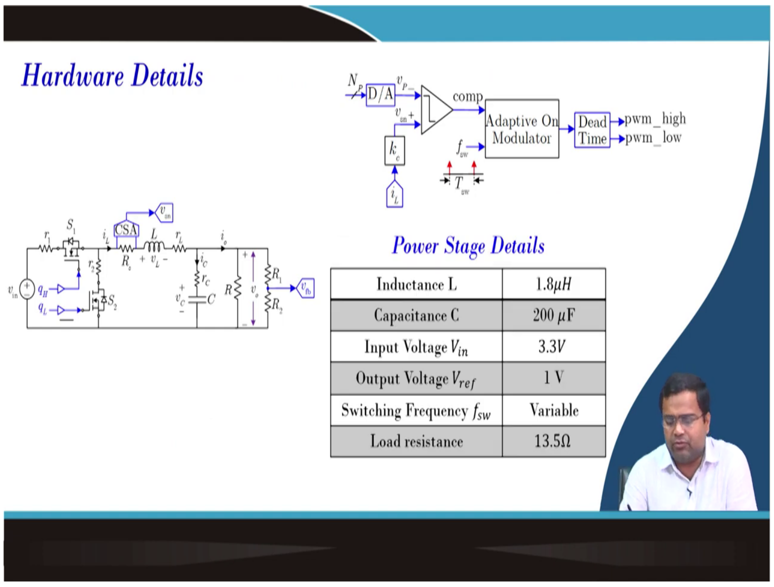
So; that means, we have discussed in detail DAC since it accepts the offset binary we have defined I peak in 2s complement we have to simply invert the MSB and that will convert any 2 s complement into an offset binary. So, that is going to the DAC.

(Refer Slide Time: 23:45)



Hardware Implementation of Voltage based COT Modulation

(Refer Slide Time: 23:48)



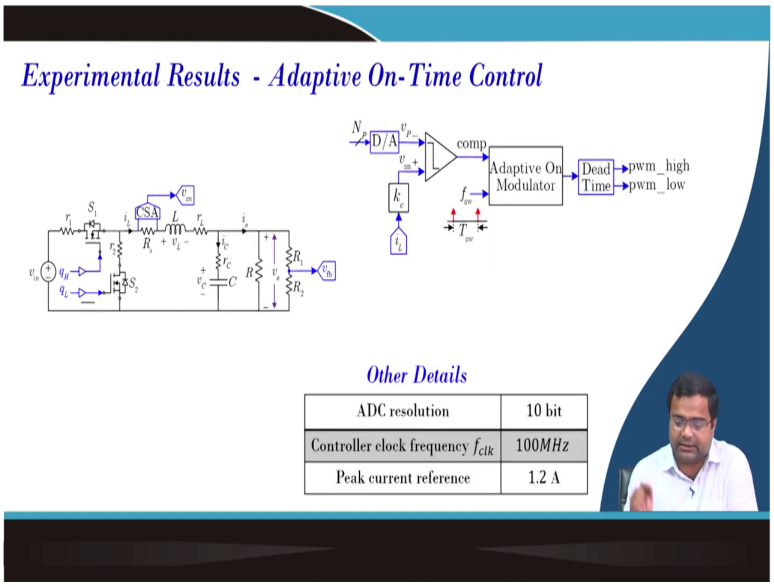
Hardware Details

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	13.5Ω

And this is a hardware implementation detail. So, you are using our buck converter in DCM mode and we have already discussed that our power stage inductor-capacitor input voltage can change output voltage also we can change. And since this is adaptive on time the switching frequency is varying. And we are using a low load condition the load resistance is high 13.5 ohm.

(Refer Slide Time: 24:06)



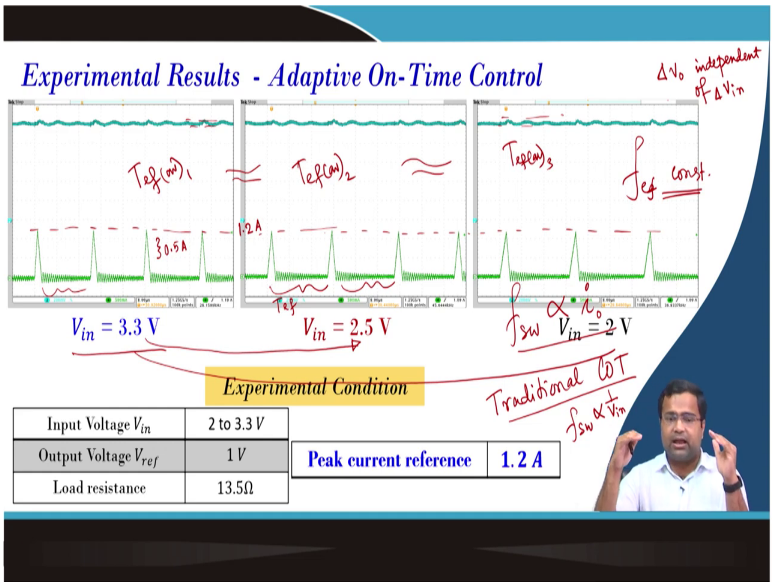
Experimental Results - Adaptive On-Time Control

Other Details

ADC resolution	10 bit
Controller clock frequency f_{clk}	$100MHz$
Peak current reference	1.2 A

And this is the adaptive on-time experimental result other detail we are using 10-bit ADC, the controller clock is 100 megahertz and we are using a current reference of nearly 1.2 ampere again this can slightly vary because the peak can slightly be higher because of the delay.

(Refer Slide Time: 24:24)



Experimental Results - Adaptive On-Time Control

Experimental Condition

Input Voltage V_{in}	2 to 3.3 V
Output Voltage V_{ref}	1 V
Load resistance	13.5Ω

Peak current reference	1.2 A

And experimental condition if we use we are varying input voltage and what do we want to expect? We know that output voltage ripple or will be more or less I would say you know it is independent of input voltage variation delta Vi will say I will not say it is the variation in the input voltage. So, this is the condition where you are operating at a higher input voltage and you can see the current division is 0.5 ampere. So, this division is 0.5 ampere.

So, we are getting here the peak current reference of around roughly 1.2 ampere ok. And you can see the voltage ripple here because it is in the dc couple mode. Now we have reduced the input voltage from 3.3 to 2.5 and we will find the peak current reference remains the same more or less the same at 1.2 as a result the output voltage ripple also can be shown to be the same because you can see the ripple voltage has no as such effect.

Another interesting fact is that we know that the time period effective time period between 2 consecutive pulses is the frequency it is related to the frequency of pfm, but here it will not be perfectly stable, because we are not using an analog comparator. We are using an ad which is we are using a 20 megahertz clock; that means, there is a 40 nanosecond every 40 nanosecond the data will be available.

As a result, we have a time resolution that is 40 nanosecond. In the case of analog, we have infinite time resolution. So, this can cause some deviation with T effective in the subsequent cycle and which we have discussed, but on average if you see T effective for both cases since the load current is the same the frequency does not depend on the input voltage in this case; that means, your switching frequency will be linearly proportional to load current.

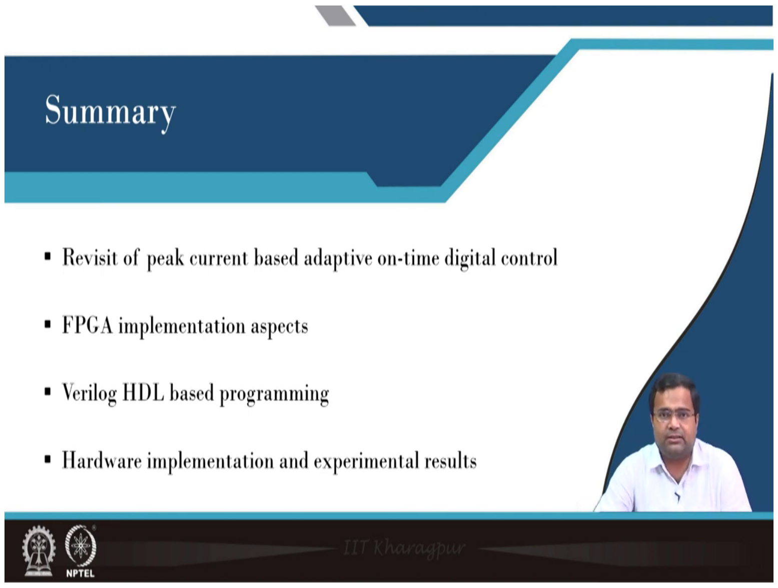
Here it is almost independent of input voltage because we are using adaptive on-time. The time period will be more or less maintained. And if you go to the other; that means, 2 volts; that means, we are decreasing from 3.3 to 2 we are maintaining the same peak current and the voltage ripple is also sort of maintained, but the time period time period. So, in all cases, if your T is effective I will say average value if you take because there is a resolution due to the 40 nanosecond difference.

And if you take T effective average case 1, case 2 anti effective average I would say case 3 all will be more or less you know equal. And that is the beauty; that means, your effective switching frequency I would say effective switching frequency is almost constant in the average sense. So, it is insensitive to input voltage variation. Generally what happens if you remember the formulation on time? If you keep the input-output voltage constant then the frequency varies with the load current, but if the load current is fixed frequency will be fixed.

But there we saw if the input voltage increases then naturally the because on time is fixed. The ripple will increase and as a result time period will increase and the frequency will decrease; that means, in traditional, I will say traditional constant on time the frequency will be inversely proportional to the input voltage; that means if the input voltage increases the frequency will decrease.

Because, the time period will increase, but here in this adaptive on time it is more or less constant. So, it retains good features which is why many commercial products use this peak current-based adaptive on-time control.

(Refer Slide Time: 28:28)



Summary

- Revisit of peak current based adaptive on-time digital control
- FPGA implementation aspects
- Verilog HDL based programming
- Hardware implementation and experimental results

IIT Kharagpur
NPTEL

So, in summary, we have discussed peak current-based architecture in the FPGA implementation aspect. We have also discussed Verilog HDL programming of this adaptive on-time digital control and we have shown hardware implementation as well as experimental results that are it for today.

Thank you very much.