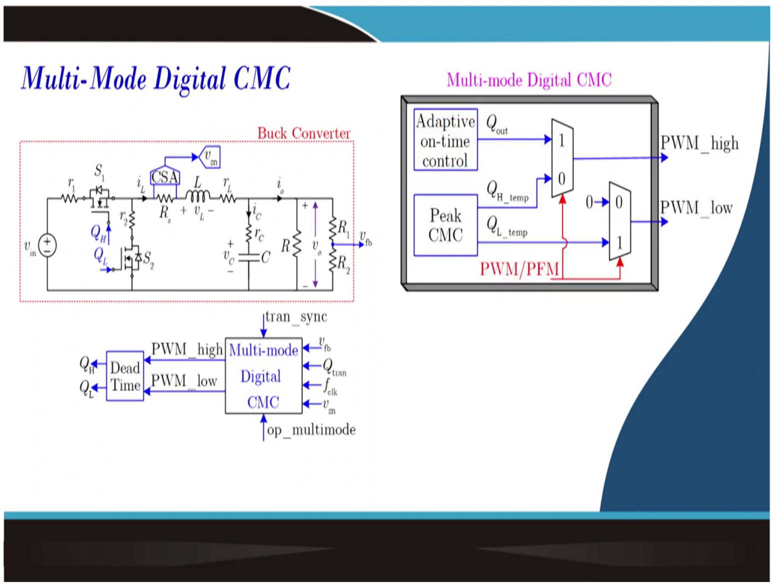
Digital Control in Switched Mode Power Converters and FPGA-based Prototyping

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

Module - 12
Hardware Case Studies of Advanced Digital Control Techniques and Course Summary
Lecture - 116
FPGA Prototyping of Peak Current-based PWM/PFM Multi-Mode Digital Control - II

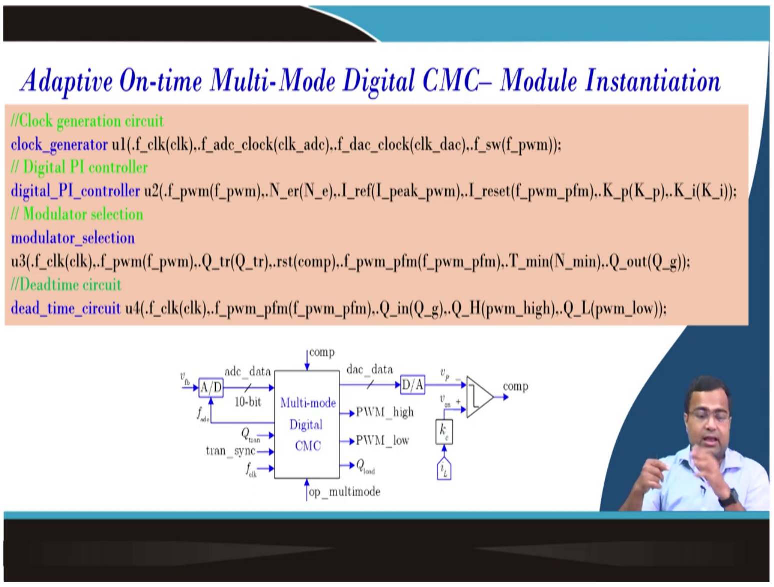
(Refer Slide Time: 00:28)



Multi-Mode Digital CMC

Welcome back this is a continuation of the previous lecture and here we will start with the Multi-Mode Digital Current Mode Control we have already discussed digital Multi-Mode Control Consisting of PWM and PFM.

(Refer Slide Time: 00:37)

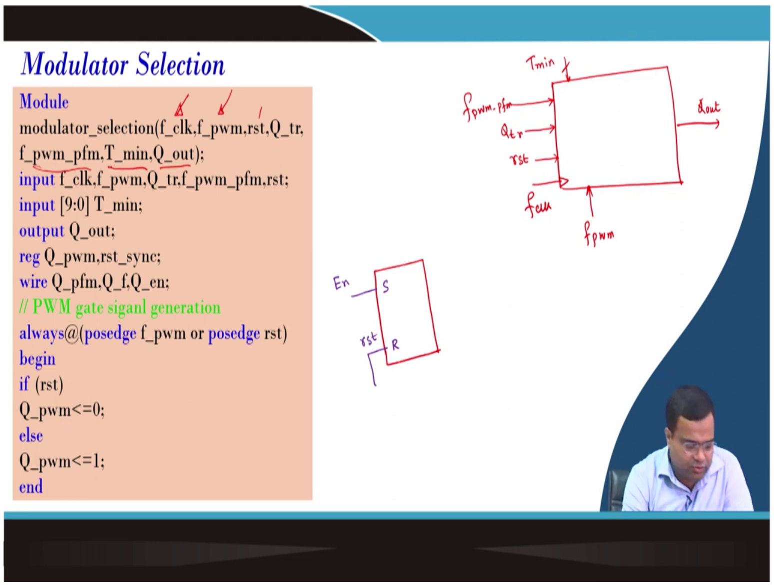


Adaptive On-time Multi-Mode Digital CMC– Module Instantiation

```
//Clock generation circuit
clock_generator u1(.f_clk(clk),.f_adc_clock(clk_adc),.f_dac_clock(clk_dac),.f_sw(f_pwm));
// Digital PI controller
digital_PI_controller u2(.f_pwm(f_pwm),.N_er(N_e),.I_ref(I_peak_pwm),.I_reset(f_pwm_pfm),.K_p(K_p),.K_i(K_i));
// Modulator selection
modulator_selection
u3(.f_clk(clk),.f_pwm(f_pwm),.Q_tr(Q_tr),.rst(comp),.f_pwm_pfm(f_pwm_pfm),.T_min(N_min),.Q_out(Q_g));
//Deadtime circuit
dead_time_circuit u4(.f_clk(clk),.f_pwm_pfm(f_pwm_pfm),.Q_in(Q_g),.Q_H(pwm_high),.Q_L(pwm_low));
```

And we have also discussed the module instantiation ok. Now we are going to talk about what is the modulator selection ok.

(Refer Slide Time: 00:48)



Modulator Selection

```
Module
modulator_selection(f_clk,f_pwm,rst,Q_tr,
f_pwm_pfm,T_min,Q_out);
input f_clk,f_pwm,Q_tr,f_pwm_pfm,rst;
input [9:0] T_min;
output Q_out;
reg Q_pwm,rst_sync;
wire Q_pfm,Q_f,Q_en;
// PWM gate siganl generation
always@(posedge f_pwm or posedge rst)
begin
if (rst)
Q_pwm<=0;
else
Q_pwm<=1;
end
```

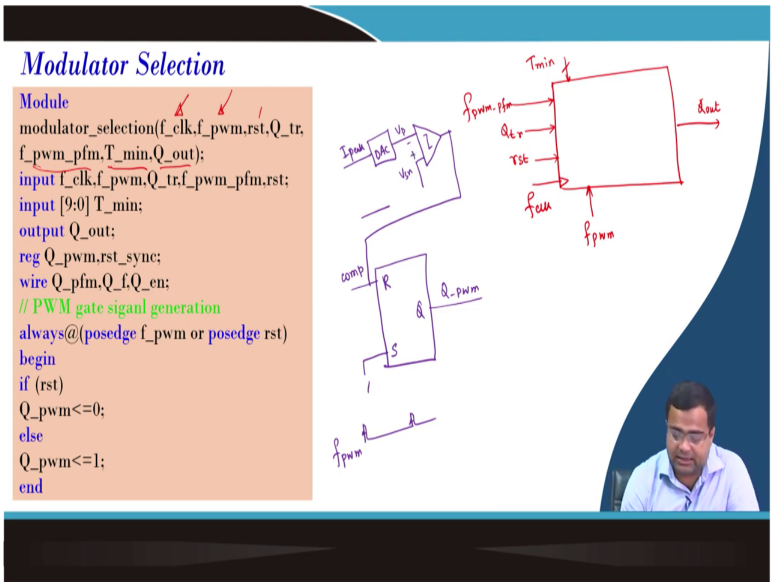
So, first, the modulator selection takes the clock’s highest frequency clock input because this is to generate the timing circuit inside the modulator that we are going to discuss. It also takes the PWM signal because if you remember in adaptive on time which we have discussed in lecture number 19.

This PWM clock was used that 5 microsecond time period was like an external clock that is coming periodically. It will check whether your comparator output is high, but on time is still not activated or your off time is elapsed. So, all this activity checks because then it may initiate another constant on-time action ok.

Then it also requires a reset then Q tr and this is a flag signal which will decide whether it is a constant on time or sorry whether it is a PWM or PFM. This is a minimum time because every constant on time should have a minimum off time. This is a command and the output of this block is Q. That means I would say this block has modulator selection it has this f clock high-frequency clock. Then it also takes f pwm then it also has a reset clock then it takes Q tr that is coming from the voltage comparator.

It has a select line f p wm pfm which controller then it also has the minimum off time that is a vector quantity and its output is Q out ok. Now the first thing you remember you know this circuit we have this block if you remember this is the set and sorry hm. So, this is set reset what is set is coming an enabling signal and we will discuss how to generate this enabled signal and this is a reset pulse. Typically it should come from; that means, I am telling that in p w m it is ok. So, there are two functions here one feature this modulator also has p wm right.

(Refer Slide Time: 03:32)

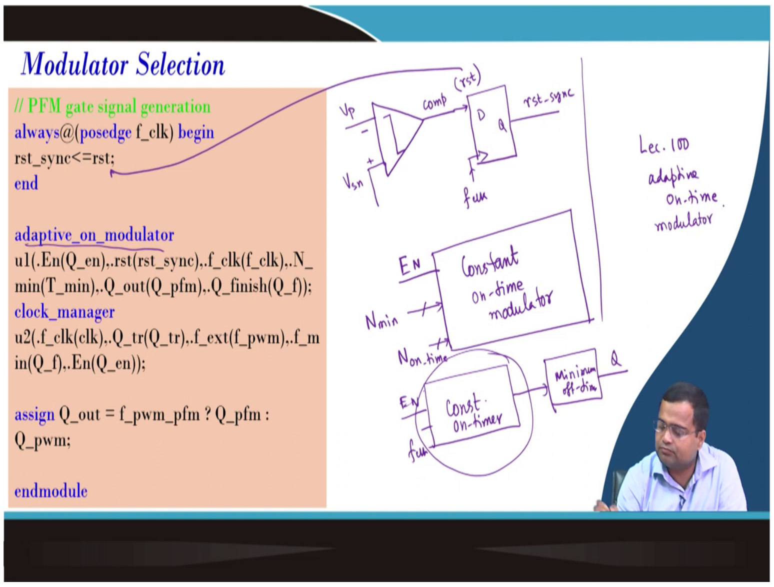


Modulator Selection

```
Module
modulator_selection(f_clk,f_pwm,rst,Q_tr,
f_pwm_pfm,T_min,Q_out);
input f_clk,f_pwm,Q_tr,f_pwm_pfm,rst;
input [9:0] T_min;
output Q_out;
reg Q_pwm,rst_sync;
wire Q_pfm,Q_f,Q_en;
// PWM gate siganl generation
always@(posedge f_pwm or posedge rst)
begin
if (rst)
Q_pwm<=0;
else
Q_pwm<=1;
end
```

So, under p wm, it will take a reset set pulse. Set it will have a switching frequency clock which is f p wm at this edge this will turn on this will Q and we call it a Q p wm and this is coming from the comparator. And what is the comparator? We have that DAC, this is your Ip this is V p in voltage sense minus this is V sn. So, this is your comparator output ok directly coming. So, this will generate the PWM clock.

(Refer Slide Time: 04:22)



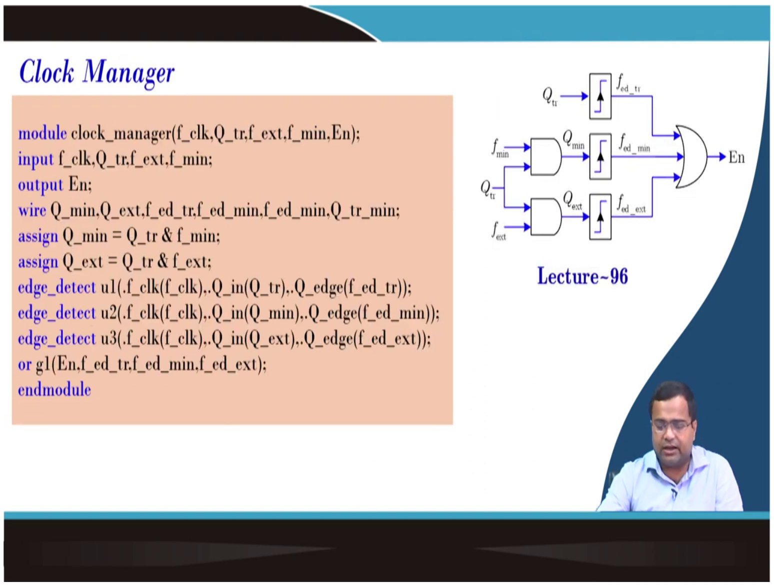
Modulator Selection

```
// PFM gate signal generation
always@(posedge f_clk) begin
rst_sync<=rst;
end

adaptive_on_modulator
u1(.En(Q_en),.rst(rst_sync),.f_clk(f_clk),.N_
min(T_min),.Q_out(Q_pfm),.Q_finish(Q_f));
clock_manager
u2(.f_clk(clk),.Q_tr(Q_tr),.f_ext(f_pwm),.f_m
in(Q_f),.En(Q_en));

assign Q_out = f_pwm_pfm ? Q_pfm :
Q_pwm;

endmodule
```

Next for the constant adaptive on-time modulator, we have discussed adaptive on-time. That means, now for adaptive on-time we are using clock synchronization. If you remember that the actual comparator output which is V peak minus and V sn plus. This is the comp this if you give it to the adaptive on time and we saw lecture number 100 that it has to drive multiple signals inside the modulator and it was getting loaded.

So, to avoid this loading effect we made it clock synchronized; that means, this is the input. It can be a d flip flop you can say a d flip flop and this is a high-frequency clock and this is what we call it you know here that this pulse we are internally defined as a reset and here we call a reset sync. This is an internal clock named reset that is why we are calling as a reset pulse it is equal to reset sync when that clock edge comes this clock edge comes.

(Refer Slide Time: 05:50)



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
```

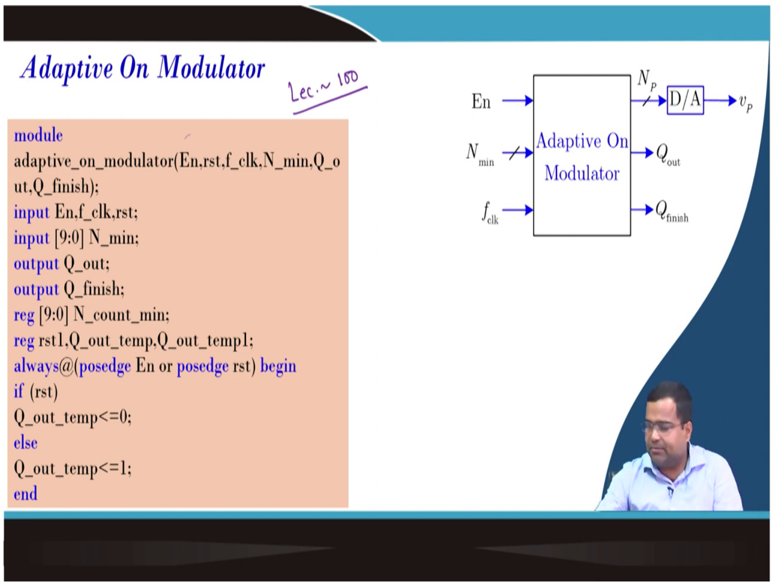
Lecture~96

Now, we are entering into adaptive on-time modulators and any adaptive on-time modulator we have discussed in lecture number 100 that you know if you go lecture number 100. What was needed? Sorry lecture number 96 also we discussed that a constant on-time modulator requires an enabled pulse right?

So, enable and high-frequency timer circuit like an N I would say on; on time N min time this is a regular constant on time. But what was the case and it consists of two; that means, the first block will be monoshot constant on a timer. This we have discussed in lecture number 96 in detail followed by minimum off time and this will generate your Q.

So, this will require enabling signal and this clock this f clock high-frequency clock. But in the case of adaptive on time this part is coming from the adaptive; that means, your current loop ok. So, that is why the adaptive on-time modulator which we have discussed in lecture number 100 adaptive on time. So, I am not going to discuss again in detail the adaptive on-time modulator ok.

(Refer Slide Time: 07:45)

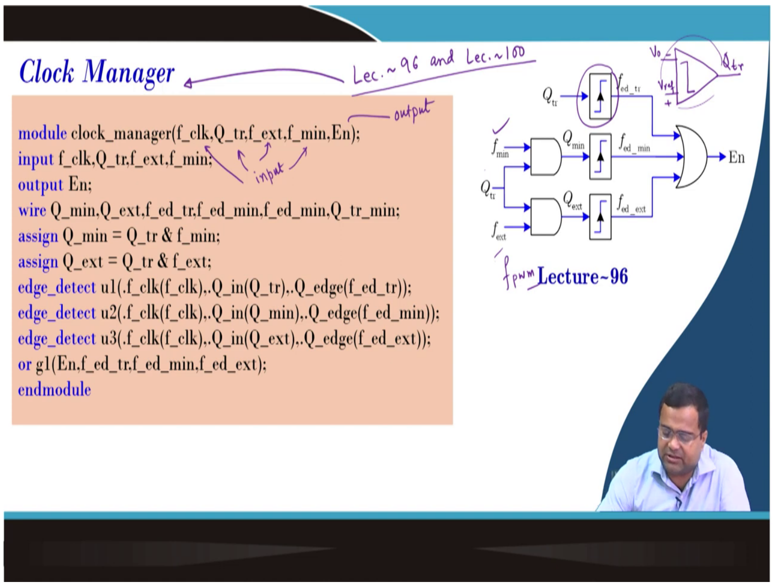


Adaptive On Modulator

```
module
adaptive_on_modulator(En,rst,f_clk,N_min,Q_o
ut,Q_finish);
input En,f_clk,rst;
input [9:0] N_min;
output Q_out;
output Q_finish;
reg [9:0] N_count_min;
reg rst1,Q_out_temp,Q_out_temp1;
always@(posedge En or posedge rst) begin
if (rst)
Q_out_temp<=0;
else
Q_out_temp<=1;
end
```

So, the adaptive on-time modulator we discussed in lecture number 100. So, one can refer to how to generate everything else is the same. Here adaptive on time we are using they are also using a peak current-based approach. So, everything is the same as the details are there in lecture number 100. I am not going to repeat the output of the adaptive.

(Refer Slide Time: 08:08)



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
```

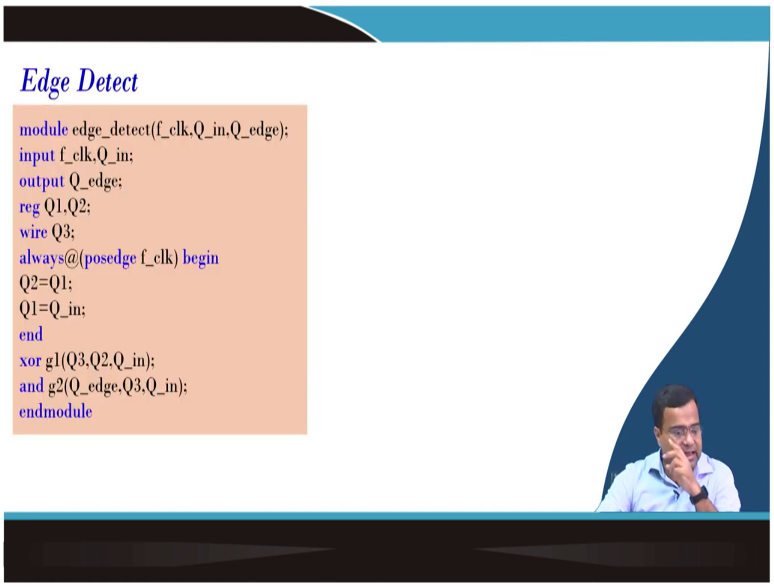
And we know for any adaptive on the timer again in lecture number I think both 96 and lecture number 100. We have discussed that any constant on-time modulator or adaptive on-time modulator requires a clock manager circuit that will generate the enabled clock. It ultimately generates this to be output and that will be the input to the adaptive on-time modulator or you can say constantly on-time modulator and the Q t r is the input.

So, this also has a input external clock this clock these are the input and we know that this generates a trigger. It will generate a positive edge trigger for the Q to which is the output of the voltage comparator. That means if the if you take output voltage and V ref if the output voltage falls below; that means, reference; that means if this is greater than this is your Q tr this is for the analog comparator.

But since we do not have an analog comparator we are using we are using the ADC 20 megahertz clock ADC. Virtually like an analog comparator by particularly comparing the output of the ADC with the reference voltage, this can be replaced by an analog comparator.

So, detecting this edge then we are generating the minimum clock; that means, when the on-time is over then it will have a minimum off time after that there will be a flag and it will check whether Q t r is still high or external clock come whether Q tr is high and this external clock we are talking about f pwm clock. Sometimes this constant time you have to synchronize with the external clock also we have discussed. So, this edge trigger we have discussed in lecture 96.

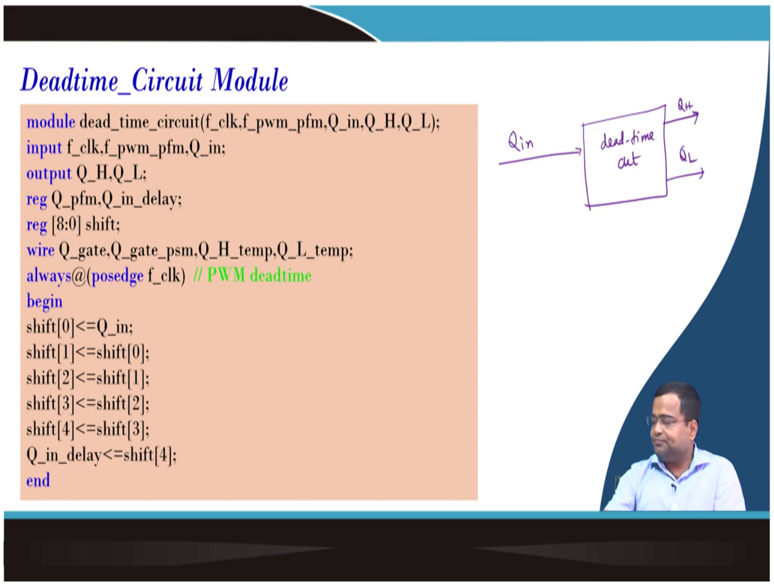
(Refer Slide Time: 10:07)



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
```

(Refer Slide Time: 10:11)

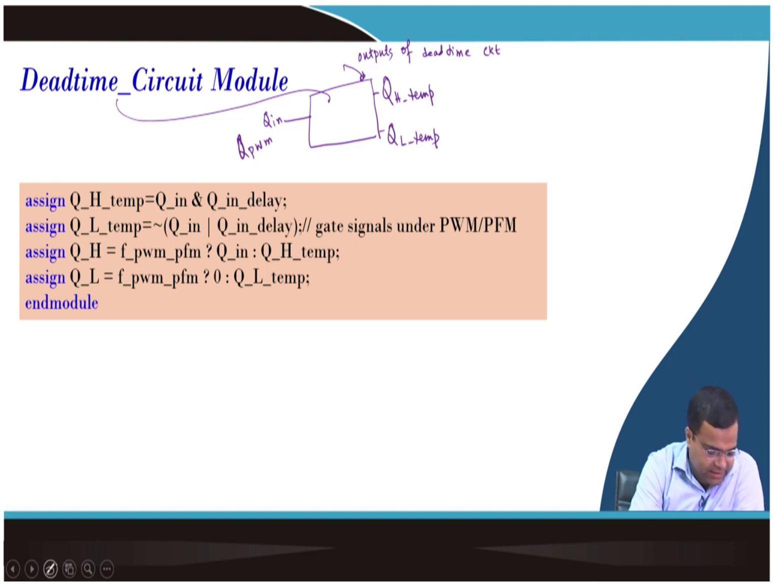


Deadtime_Circuit Module

```
module dead_time_circuit(f_clk,f_pwm_pfm,Q_in,Q_H,Q_L);
input f_clk,f_pwm_pfm,Q_in;
output Q_H,Q_L;
reg Q_pfm,Q_in_delay;
reg [8:0] shift;
wire Q_gate,Q_gate_psm,Q_H_temp,Q_L_temp;
always@(posedge f_clk)  // PWM deadtime
begin
shift[0]<=Q_in;
shift[1]<=shift[0];
shift[2]<=shift[1];
shift[3]<=shift[2];
shift[4]<=shift[3];
Q_in_delay<=shift[4];
end
```

Edge detection circuit we have discussed also in lecture number 96 and the dead time circuit here we have an option. So, for the PWM clock dead time circuit is just simply you have a Q in, then this dead time circuit generates two signals Q H, and Q L this is the local variable.

(Refer Slide Time: 10:36)

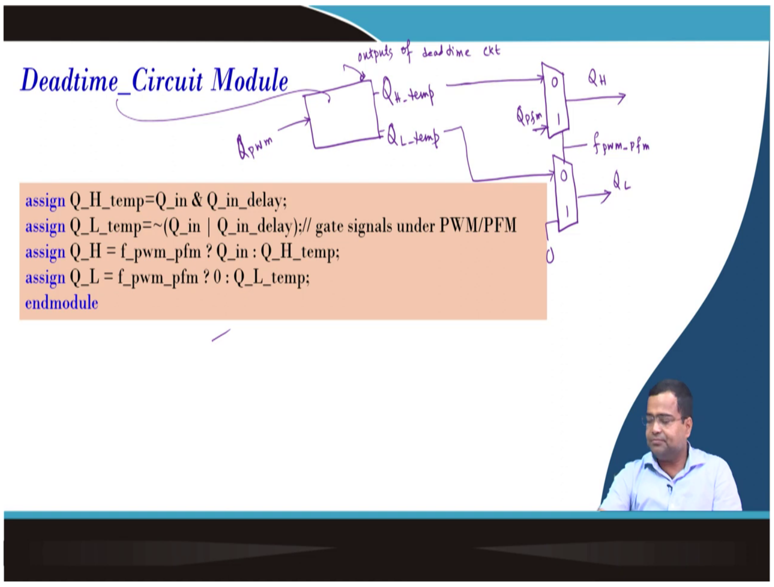


Deadtime_Circuit Module

```
assign Q_H_temp=Q_in & Q_in_delay;
assign Q_L_temp=~(Q_in | Q_in_delay);// gate signals under PWM/PFM
assign Q_H = f_pwm_pfm ? Q_in : Q_H_temp;
assign Q_L = f_pwm_pfm ? 0 : Q_L_temp;
endmodule
```

And finally what is the actual signal going into this? You see Q H temp. This is the output of the dead time circuit, circuit outputs this is one another Q low temp the high side, and the low side, and this was the dead time circuit which was generating dead time circuit. What was the input? It was Q in and we have defindefined Q pwm, it is the Q pwm with the input of ok.

(Refer Slide Time: 11:40)

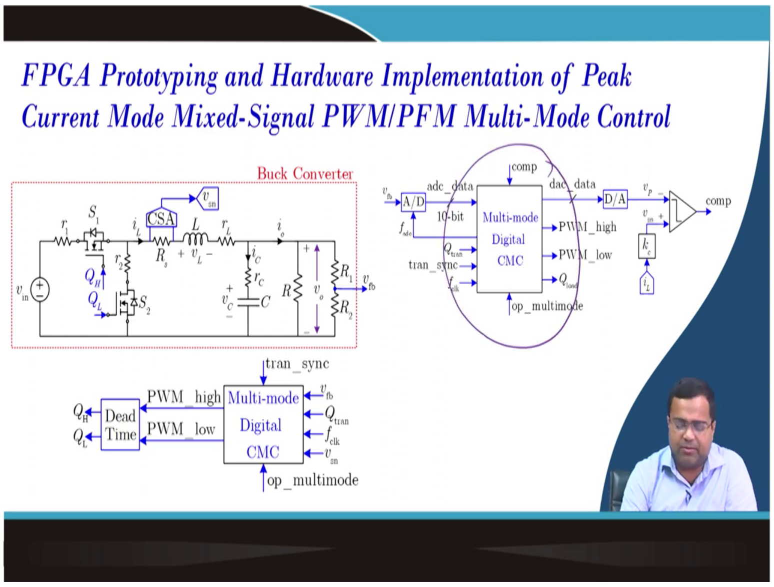


Deadtime_Circuit Module

```
assign Q_H_temp=Q_in & Q_in_delay;
assign Q_L_temp=~(Q_in | Q_in_delay);// gate signals under PWM/PFM
assign Q_H = f_pwm_pfm ? Q_in : Q_H_temp;
assign Q_L = f_pwm_pfm ? 0 : Q_L_temp;
endmodule
```

Now, when the PWM will come again it will take two parts Q pfm 0, 1. This will be your Q high side gate signal and this is again an f pwm; pfm this will come as this 0, 1 and this will be simply 0. This will go to the Q low gate signal and this is the logic.

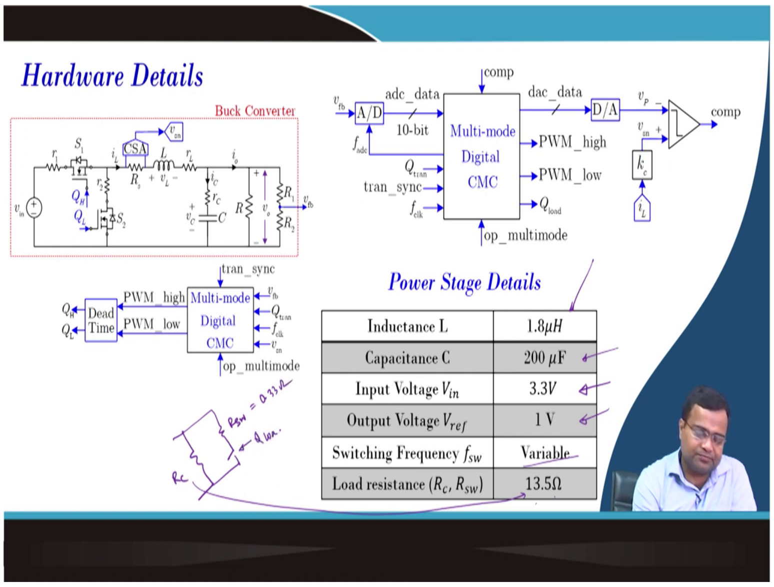
(Refer Slide Time: 12:28)



FPGA Prototyping and Hardware Implementation of Peak Current Mode Mixed-Signal PWM/PFM Multi-Mode Control

So, now we are going to say FPGA prototyping and hardware implementation of peak current-based mode mixed signal PWM PFM multi-mode control. So, this is the block that we have to synthesize in Verilog we have discussed.

(Refer Slide Time: 12:42)



Hardware Details

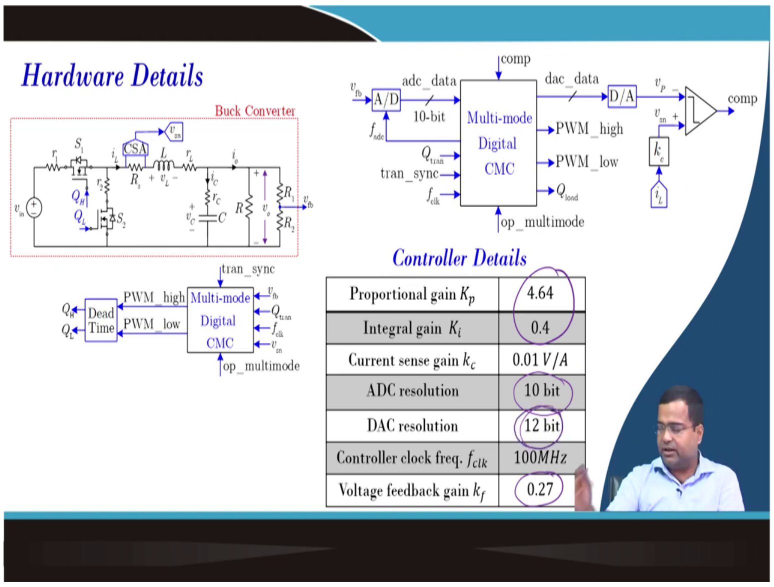
Power Stage Details

Inductance L	$1.8\mu H$
Capacitance C	200 μF
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 hardware result and we are using our hardware prototype 1.8 microhenry inductor; 200 micro watt capacitor are we are going to test at 3.3 volt input. But you know the users can use different specifications and we are using 1 volt output switching frequency is varying for pfm.

But it will be fixed for PWM which is 200-kilo hertz and load resistance was changing from 13.5. Because we know there are two load resistance R c this R c is this and there is another load resistance R s w which is 0.333 ohm and this is whenever the Q load is high ok.

(Refer Slide Time: 13:25)



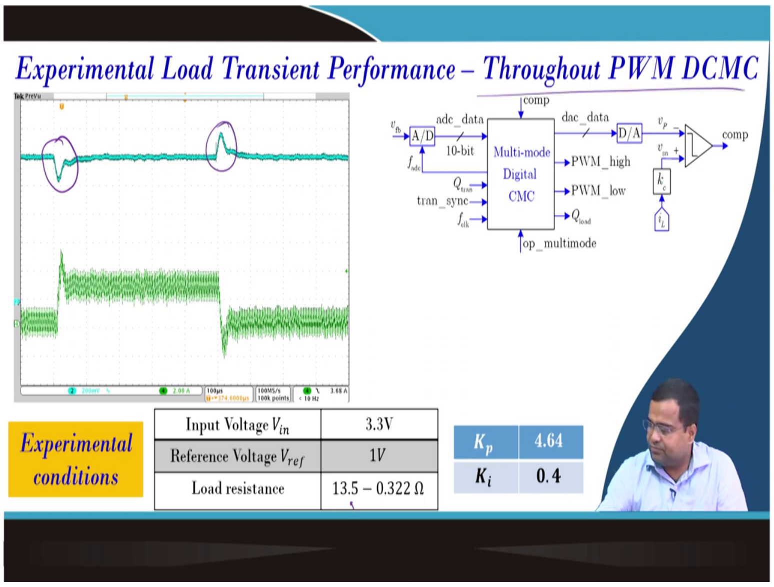
Hardware Details

Controller Details

Proportional gain K_p	4.64
Integral gain K_i	0.4
Current sense gain k_c	$0.01\ V/A$
ADC resolution	10 bit
DAC resolution	12 bit
Controller clock freq. f_{clk}	$100MHz$
Voltage feedback gain k_f	0.27

When we did controller gain we are using the PI controller gain same that we used in lecture numbers I think 107, 108, and current sense again all are known voltage feedback gains this much ADC resolution is 12-bit; DAC resolution sorry ADC resolution is 10 bit and DAC resolution 12 bit with offset binary we have discussed.

(Refer Slide Time: 13:50)



Experimental Load Transient Performance – Throughout PWM DCMC

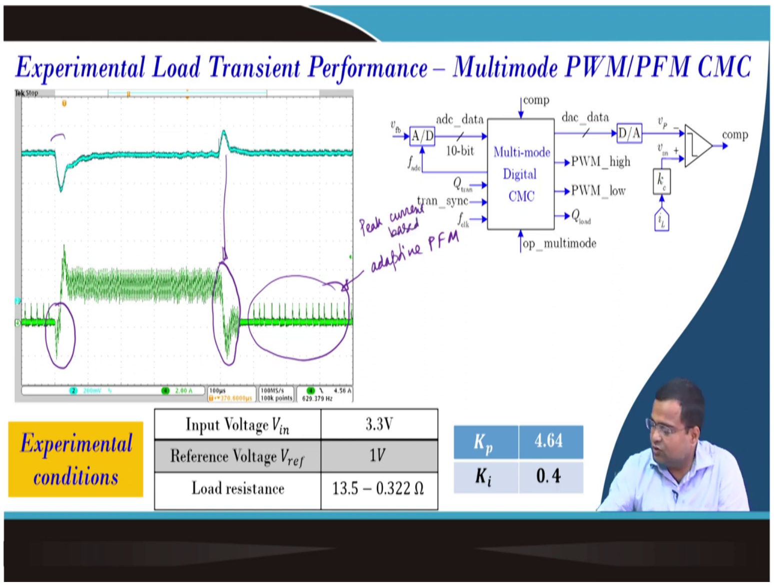
Experimental conditions

Input Voltage V_{in}	3.3V
Reference Voltage V_{ref}	1V
Load resistance	13.5 – 0.322 Ω

K_p	4.64
K_i	0.4

Now, we are going to show an experimental case study at 3.3 volt input; 1 volt output. We are changing the load resistance from 13.5 which was the continuous load and we are changing to 0.322 when both the resistance are connected and it shows the load transient performance. This is the step up transient step down transient when it is throughout PWM digital current mode control.

(Refer Slide Time: 14:13)



Experimental Load Transient Performance – Multimode PWM/PFM CMC

Experimental conditions

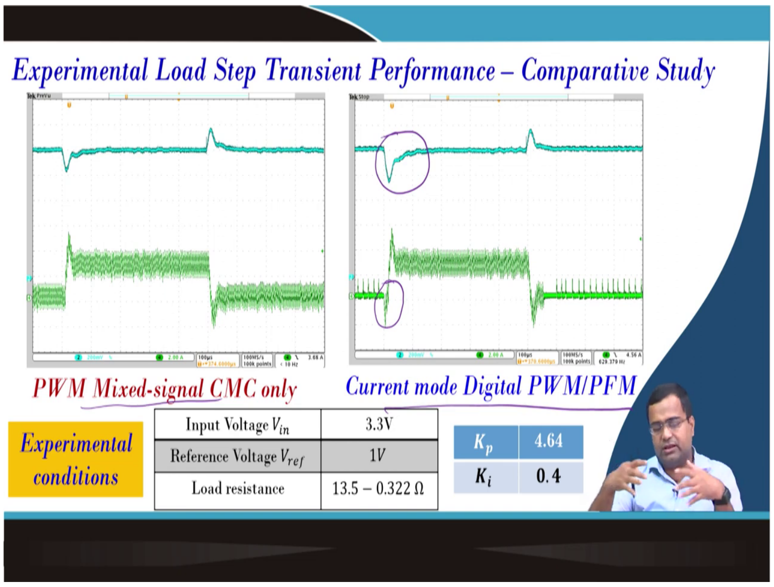
Input Voltage V_{in}	3.3V
Reference Voltage V_{ref}	1V
Load resistance	13.5 – 0.322 Ω

K_p	4.64
K_i	0.4

Now, we are enabling multi-mode control. So, when you say multi mode you see there is an additional thing because integral we are starting with 0 resettings and that time as we discussed in I think lecture number 114 the error will be because, in PFM generally, the output voltage is above the reference voltage. And as a result, your error voltage will be negative and this is going negative direction.

Another thing during step down transient we are enabling this p wm operation for a certain time that is how we can reduce the settling improve the settling time and we can retain the same settling time for PWM. And you can see during light load it is in adaptive PFM; adaptive PFM; Peak Current Based ok.

(Refer Slide Time: 15:16)



Experimental Load Step Transient Performance – Comparative Study

PWM Mixed-signal CMC only

Current mode Digital PWM/PFM

Experimental conditions

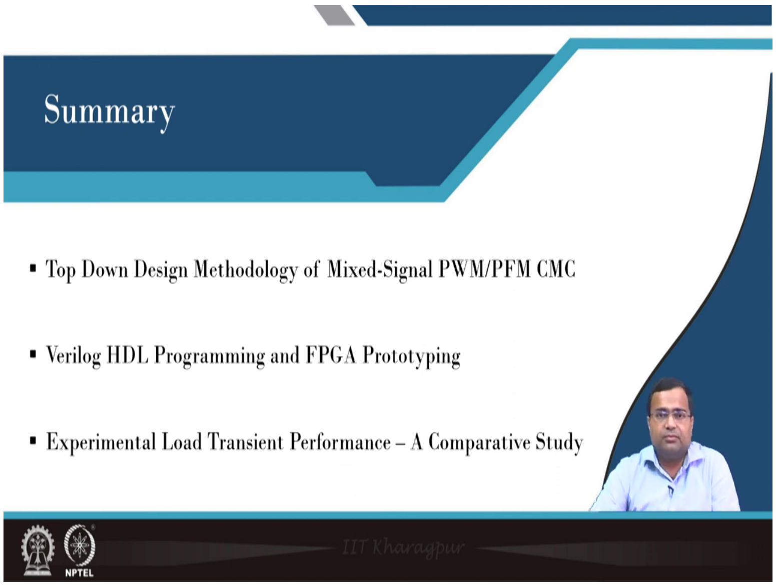
Input Voltage V_{in}	3.3V
Reference Voltage V_{ref}	$1V$
Load resistance	13.5 – 0.322 Ω

K_p	4.64
K_i	0.4

And this is the transient performance comparison this is throughout PWM and this is a multi-mode control. So, there will be a reduction because of this negative error at the time of transition when we are making an integral reset. But there are research papers and commercial products which try to improve this mode transition. These are penalties due to the mode transition when you are doing the integral reset.

So, there is N number of techniques to detect transient and take action by a non-linear adding non-linear term. So, we are not going to discuss this here this may be a research topic or you know the participant can do further exploration.

(Refer Slide Time: 15:57)



Summary

- Top Down Design Methodology of Mixed-Signal PWM/PFM CMC
- Verilog HDL Programming and FPGA Prototyping
- Experimental Load Transient Performance – A Comparative Study

NPTEL
IIT Kharagpur

So, in summary, we have discussed the top-down design methodology of mixed-signal peak current-based PWM PFM control. We have synthesized this control technique using Verilog programming and we have shown experimental load transient performance along with a comparative study throughout PWM and a multi-mode control that is it for today.

Thank you very much.