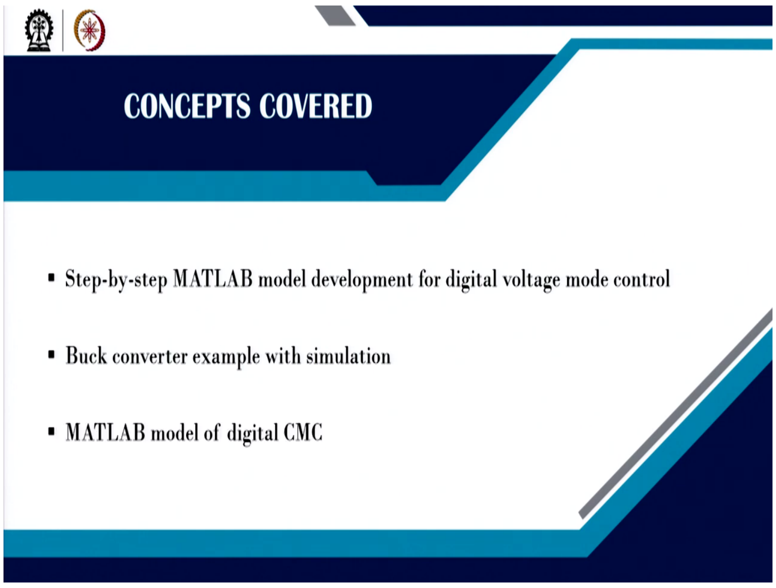
Digital Control in Switched Mode Power Converters and FPGA-based Prototyping

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

Module - 03
MATLAB Custom Model Development under Digital Control
Lecture - 23
MATLAB Model Development for Fixed Frequency Digital Control

Welcome back. In this lecture, we are going to talk about MATLAB Model Development for Fixed Frequency Digital Control. This lecture is the continuation of the previous lecture. Here we will extend whatever we learned about the step-by-step MATLAB model development for digital voltage mode control, but at a very basic level.

(Refer Slide Time: 00:31)

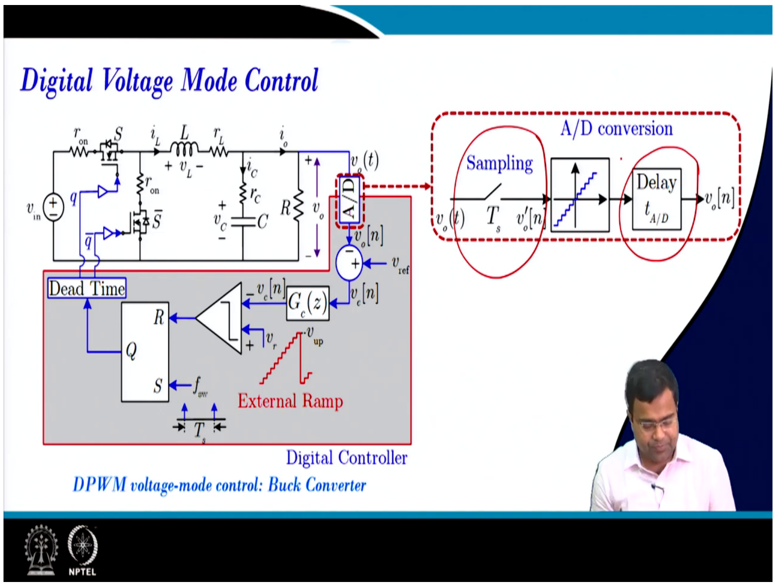


CONCEPTS COVERED

- Step-by-step MATLAB model development for digital voltage mode control
- Buck converter example with simulation
- MATLAB model of digital CMC

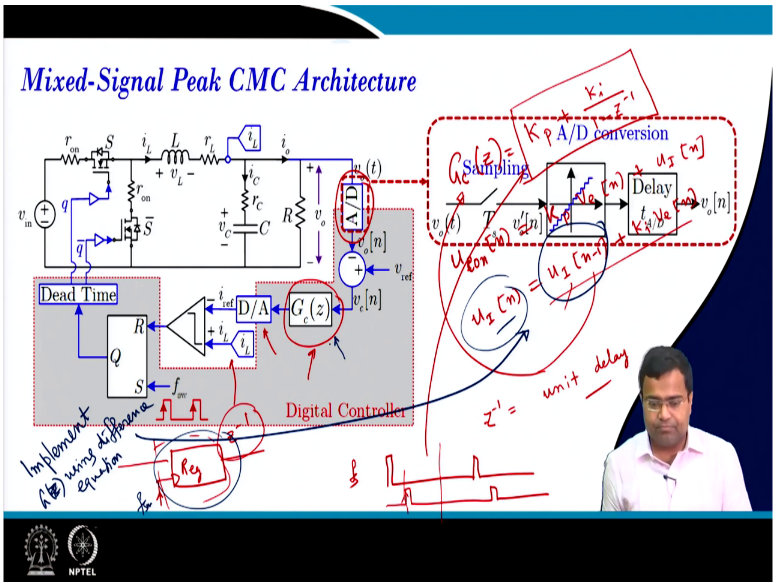
Then, we have shown a buck converter example with simulation. We will just show up once again. And, then we will show here the MATLAB model for digital current mode control.

(Refer Slide Time: 00:52)


Digital Voltage Mode Control
A/D conversion
Sampling
Delay
Dead Time
External Ramp
Digital Controller
DPWM voltage-mode control: Buck Converter

So, here in digital voltage mode control in the previous lecture, we talked, that we talked about A to D converter. And, we have already discussed how to implement this sampling block using enable and disable blocks, where we can customize the sampling point as well as the sampling rate. We have also discussed how to introduce a delay in that signal, basically at what point should we sample ok.

(Refer Slide Time: 01:20)

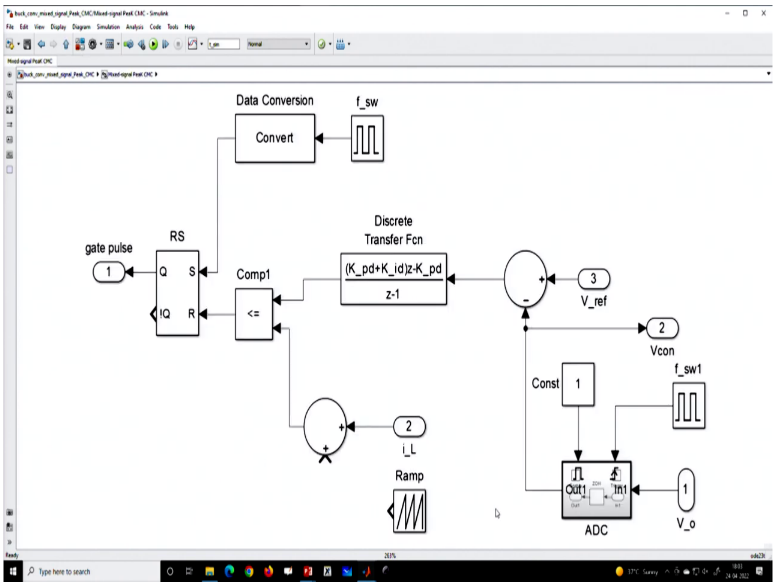

Mixed-Signal Peak CMC Architecture
A/D conversion
Sampling
Delay
Dead Time
D/A
Digital Controller
Implement G(z) using difference equation
unit delay

Next, we will talk about mixed signal current mode control because we have discussed in mixed signal current mode control your current loop is analog and the voltage loop is digital.

So, we need to implement digital. So, we need A to D converter and since we are not using a digital platform number; so, it is a MATLAB version which is why we do not need a separate D-to-A converter in the MATLAB simulation.

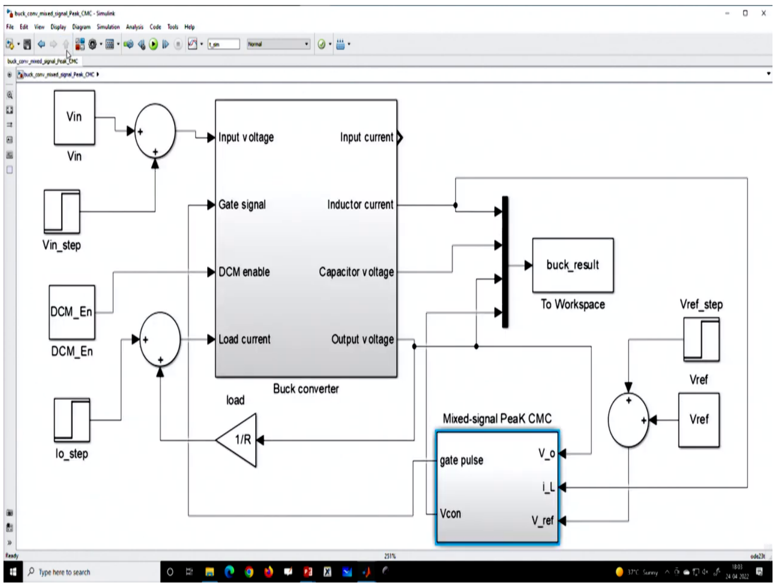
But, you know one can try out the model of a D to-A converter because D to A converter model is like a zero-order hold effect and that is already incorporated.

(Refer Slide Time: 01:53)

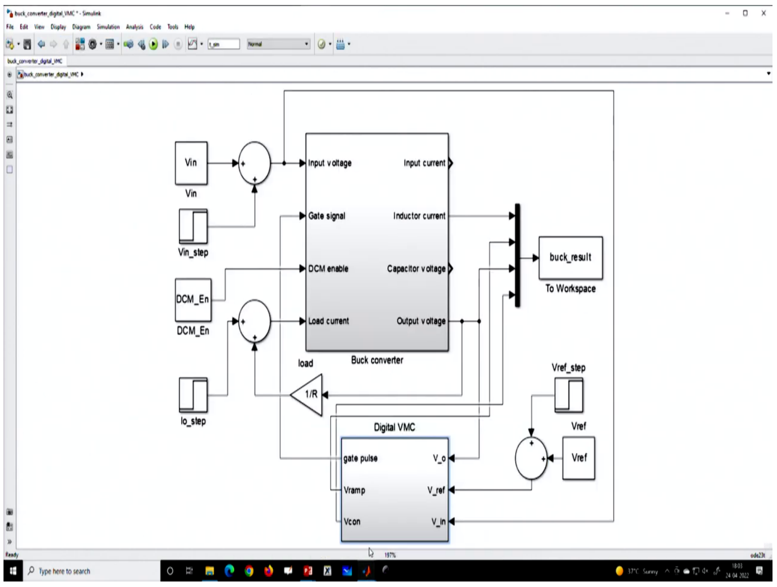


So, let us go to the MATLAB model of the digital current mode control.

(Refer Slide Time: 01:58)

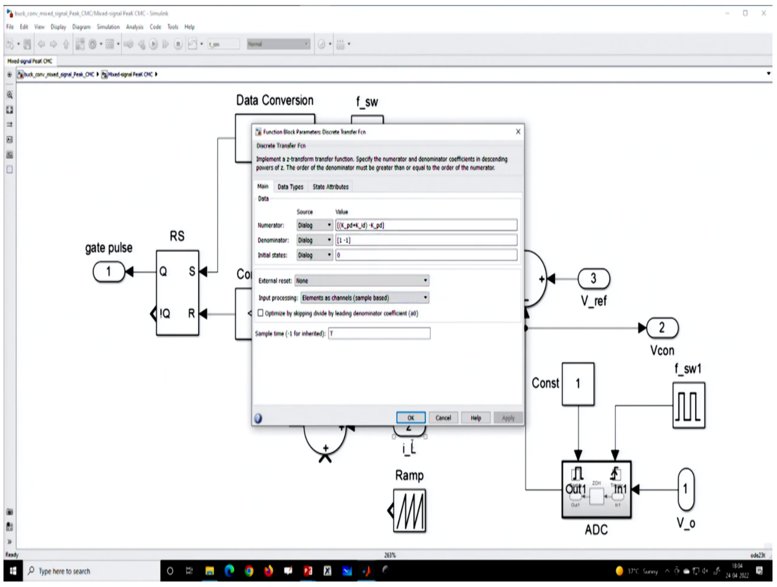


(Refer Slide Time: 02:02)



So, here you know we have already discussed digital voltage mode control in the previous lecture. So, I am not going inside this block, but here we are talking about digital current mode control. So, everything else is the same, only this controller block is different. If you go inside again there is a controller. I have used the same controller, but I will not use derivative action.

(Refer Slide Time: 02:27)



Function Block Parameters: Discrete Transfer Fcn

Discrete Transfer Fcn

Implement a z-transform transfer function. Specify the numerator and denominator coefficients in descending powers of z. The order of the denominator must be greater than or equal to the order of the numerator.

Main | Data Types | State Attributes

Data

	Source	Value
Numerator:	Dialog	[(K_pd+K_id) -K_pd]
Denominator:	Dialog	[1 -1]
Initial states:	Dialog	0

External reset: None

Input processing: Elements as channels (sample based)

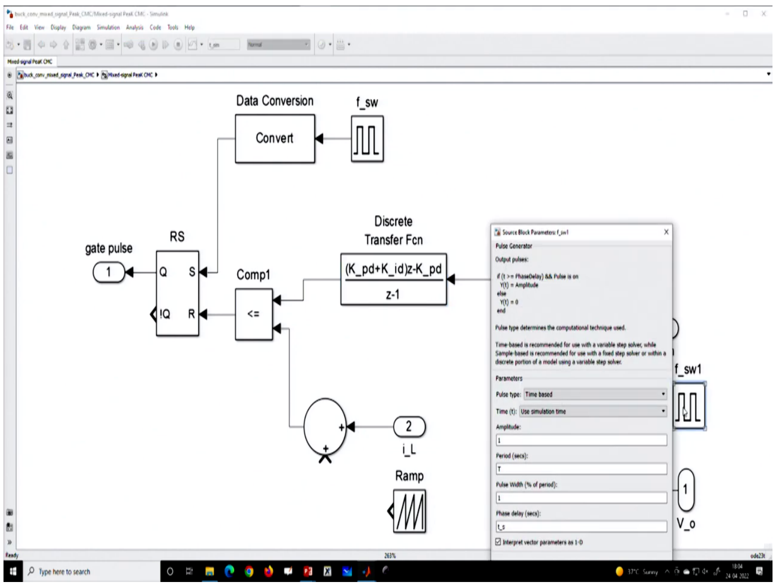
☐ Optimize by skipping divide by leading denominator coefficient (a0)

Sample time (-1 for inherited): T

So, I will put derivative action to be 0 and you see you know z is equal to 1, because if only PI controller we will see. So, it is only; so, pd sorry it is not derivative, it is a p and I because

I am talking PD means discrete time proportional gain and K id means discrete-time integral gain. So, this is the representation of a discrete-time PI controller ok. And, then you know this is the reference voltage and this is the A to D converter that we have discussed.

(Refer Slide Time: 02:56)



Source Block Parameters: f_sw1

Pulse Generator

Output pulses:

```
if (t >= PhaseDelay) && Pulse is on
  Y(t) = Amplitude
else
  Y(t) = 0
end
```

Pulse type determines the computational technique used.

Time-based is recommended for use with a variable step solver, while Sample-based is recommended for use with a fixed step solver or within a discrete portion of a model using a variable step solver.

Parameters

Pulse type: Time based

Time (t): Use simulation time

Amplitude: 1

Period (secs): T

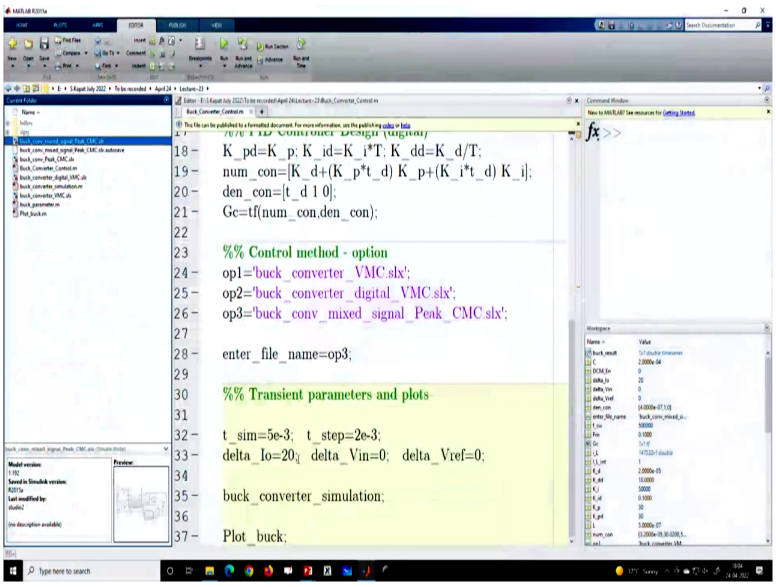
Pulse Width (% of period): 1

Phase delay (secs): t_s

☑ Interpret vector parameters as 1-D

And, we can customize the sampling point here. So, this is a sampling delay inserted, this is an inductor current. So, this block is common. This whole block, this block is common, all these blocks are common analog domain; analog current mode control, only this block is different here right? So, this discrete-time transfer function and this block. So, that means this is a basic diagram where we are just inserting the digital block ok.

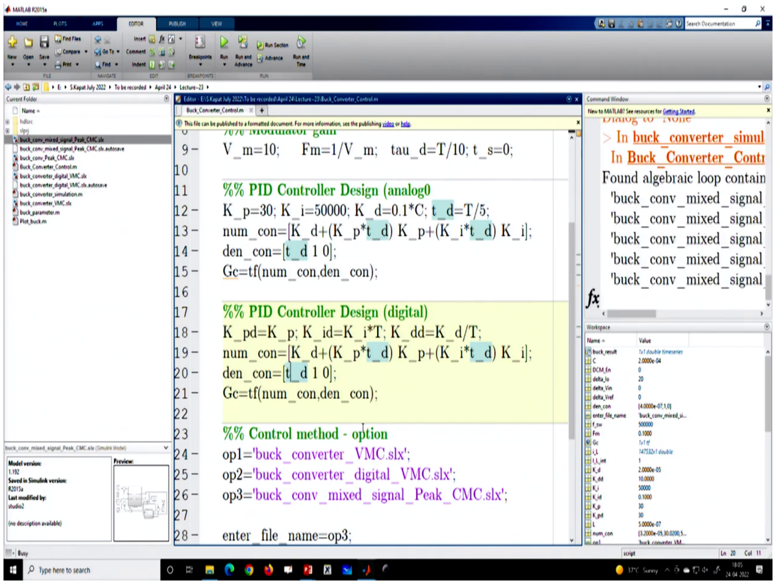
(Refer Slide Time: 03:23)



```
K_pd=K_p; K_id=K_i*T; K_dd=K_d/T;
num_con=[K_d+(K_p*t_d) K_p+(K_i*t_d) K_i];
den_con=[t_d 1 0];
Gc=tf(num_con,den_con);

%% Control method - option
op1='buck_converter_VMC.slx';
op2='buck_converter_digital_VMC.slx';
op3='buck_conv_mixed_signal_Peak_CMC.slx';

enter_file_name=op3;

%% Transient parameters and plots

t_sim=5e-3;   t_step=2e-3;
delta_Io=20;   delta_Vin=0;   delta_Vref=0;

buck_converter_simulation;

Plot_buck;
```

Now, let us go and run the simulation. So, here we are talking about the mixed signal peak current mode control.

(Refer Slide Time: 03:33)

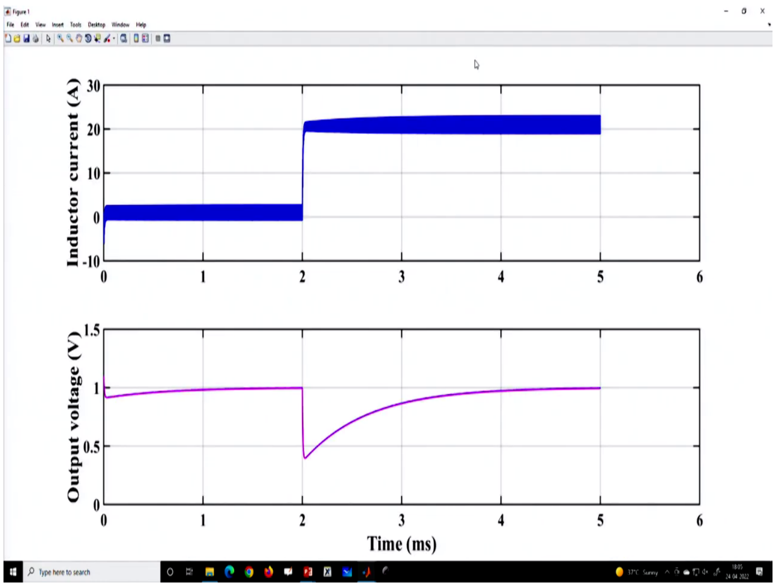


```
V_m=10;     Fm=1/V_m;   tau_d=T/10; t_s=0;

%% PID Controller Design (analog0
K_p=30; K_i=50000; K_d=0.1*C; t_d=T/5;
num_con=[K_d+(K_p*t_d) K_p+(K_i*t_d) K_i];
den_con=[t_d 1 0];
Gc=tf(num_con,den_con);

%% PID Controller Design (digital)
K_pd=K_p; K_id=K_i*T; K_dd=K_d/T;
num_con=[K_d+(K_p*t_d) K_p+(K_i*t_d) K_i];
den_con=[t_d 1 0];
Gc=tf(num_con,den_con);

%% Control method - option
op1='buck_converter_VMC.slx';
op2='buck_converter_digital_VMC.slx';
op3='buck_conv_mixed_signal_Peak_CMC.slx';

enter_file_name=op3;
```

```
> In buck_converter_simul
  In Buck_Converter_Contr
Found algebraic loop contain
 'buck_conv_mixed_signal
 'buck_conv_mixed_signal
 'buck_conv_mixed_signal
 'buck_conv_mixed_signal
 'buck_conv_mixed_signal
```

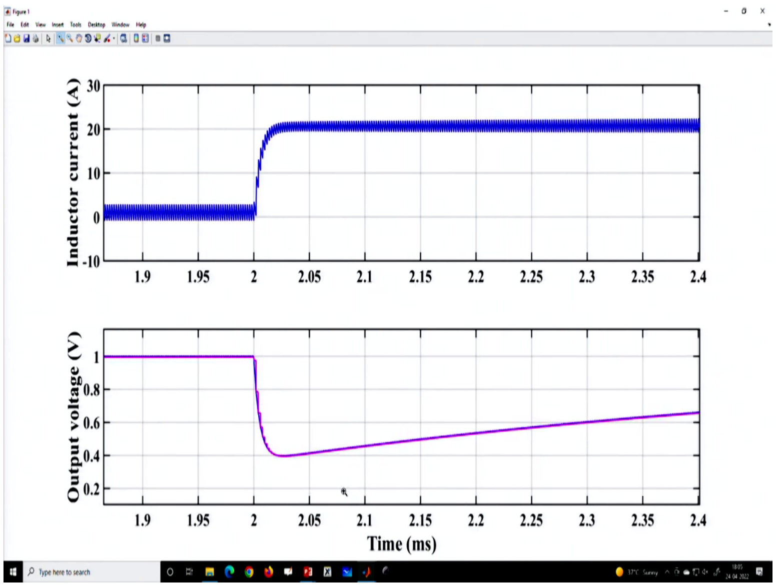
And, if we want to increase the gain, let us say we are using a little bit higher gain.

(Refer Slide Time: 03:39)

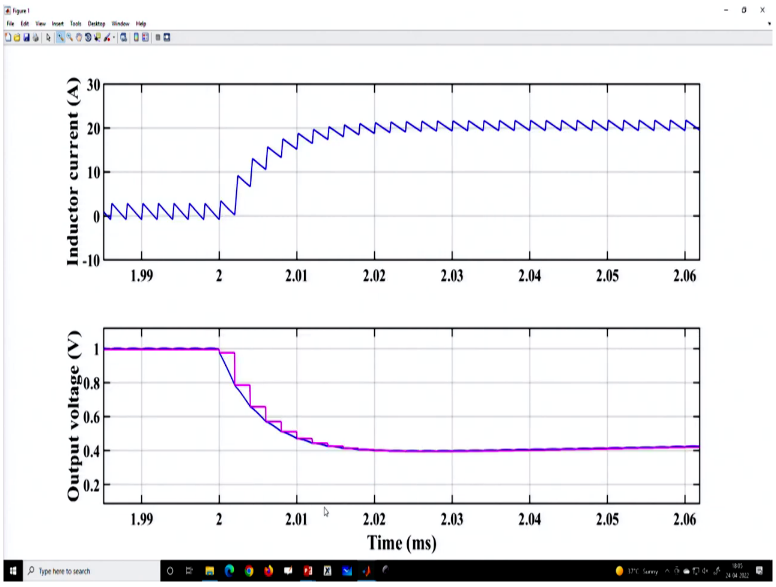


And, we will see what the response looks like. So, this is a mixed signal current mode control.

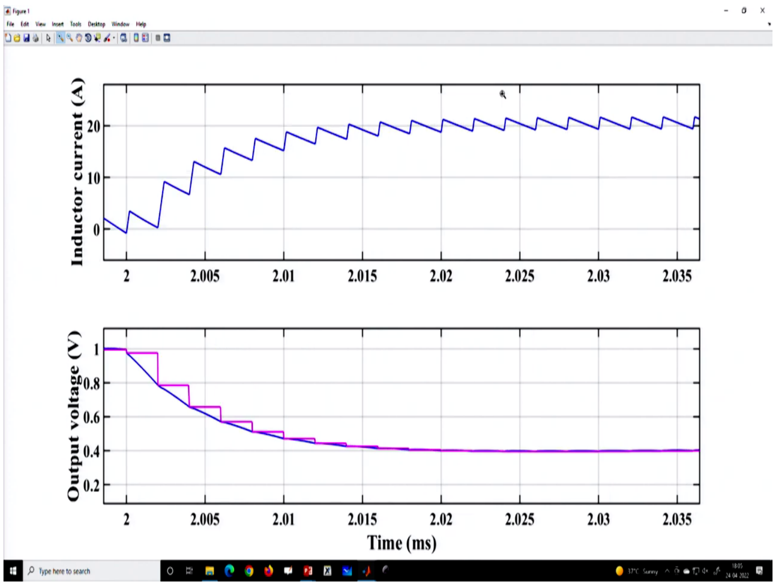
(Refer Slide Time: 03:46)



(Refer Slide Time: 03:48)

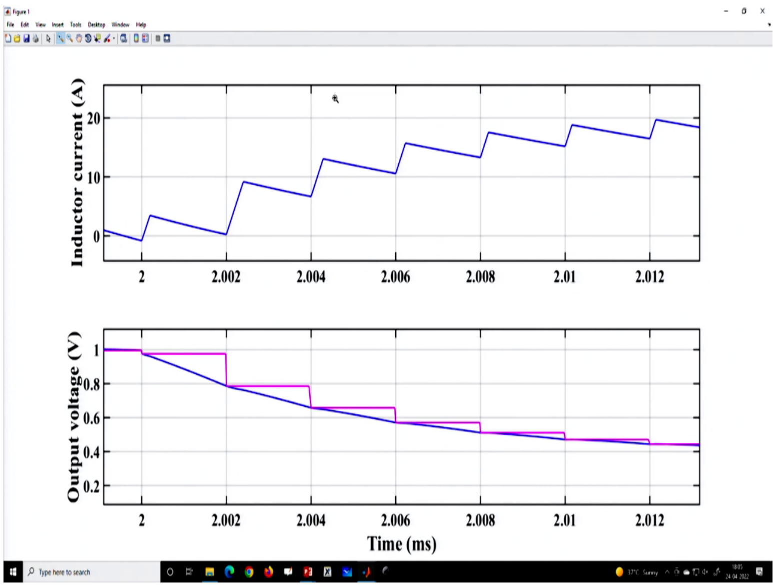


(Refer Slide Time: 03:50)



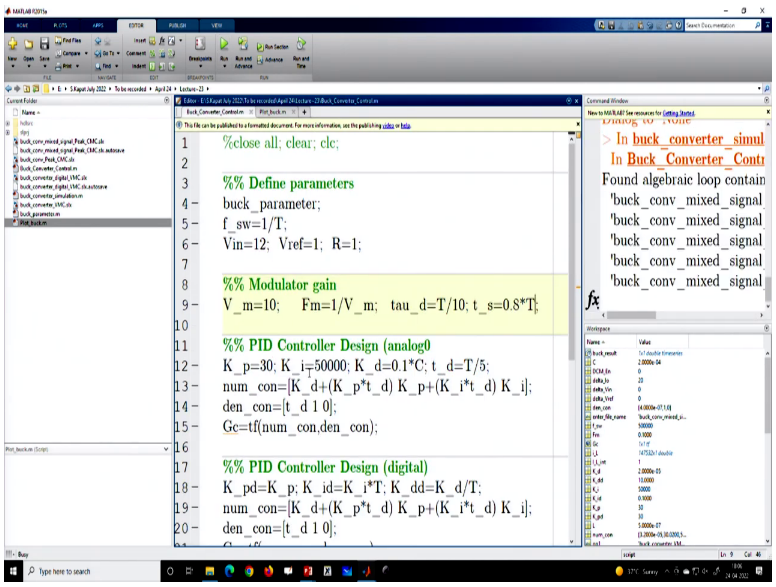
Where we are using, this is we are sampling once per cycle.

(Refer Slide Time: 03:55)



And, there is no delay here. You see there is sampling exactly at the point of switching ok. But, now we want to sample a little bit earlier. So, how to incorporate it? So, let us go back and hear it we have t s.

(Refer Slide Time: 04:15)

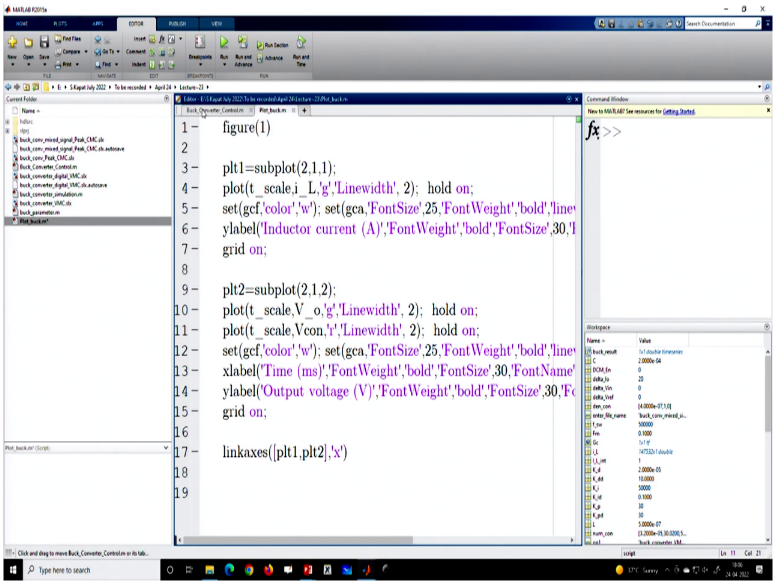


```
%close all; clear; clc;

%% Define parameters
buck_parameter;
f_sw=1/T;
Vin=12;  Vref=1;  R=1;

%% Modulator gain
V_m=10;     Fm=1/V_m;   tau_d=T/10; t_s=0.8*T;

%% PID Controller Design (analog0
K_p=30; K_i=50000; K_d=0.1*C; t_d=T/5;
num_con=[K_d+(K_p*t_d) K_p+(K_i*t_d) K_i];
den_con=[t_d 1 0];
Gc=tf(num_con,den_con);

%% PID Controller Design (digital)
K_pd=K_p; K_id=K_i*T; K_dd=K_d/T;
num_con=[K_d+(K_p*t_d) K_p+(K_i*t_d) K_i];
den_con=[t_d 1 0];
```

```
> In buck_converter_simul
  In Buck_Converter_Contr
Found algebraic loop contain
  'buck_conv_mixed_signal
  'buck_conv_mixed_signal
  'buck_conv_mixed_signal
  'buck_conv_mixed_signal
  'buck_conv_mixed_signal
```

So, in our code let us go back. So, we will hold the previous result and put t s equal to 0 points, let us say 8 times T.

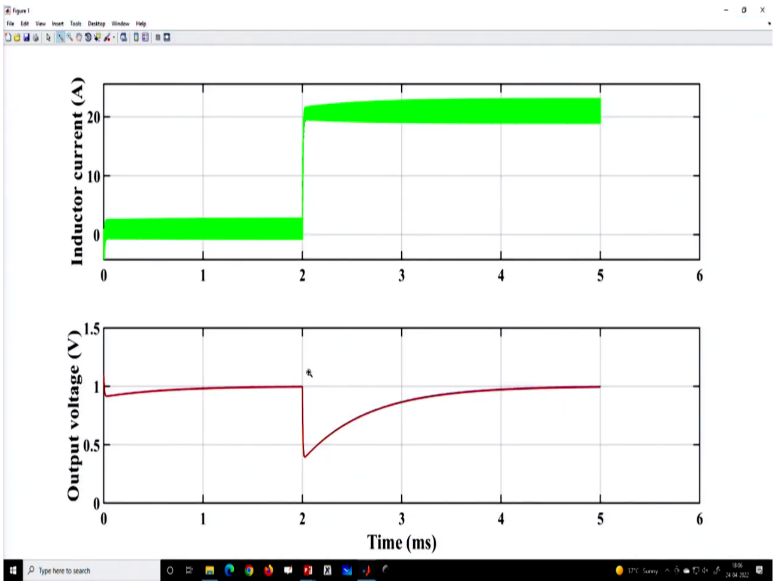
(Refer Slide Time: 04:26)



```
figure(1)

plt1=subplot(2,1,1);
plot(t_scale,i_L,'g','Linewidth', 2);  hold on;
set(gcf,'color','w'); set(gca,'FontSize',25,'FontWeight','bold','line
ylabel('Inductor current (A)','FontWeight','bold','FontSize',30,'
grid on;

plt2=subplot(2,1,2);
plot(t_scale,V_o,'g','Linewidth', 2);  hold on;
plot(t_scale,Vcon,'r','Linewidth', 2);  hold on;
set(gcf,'color','w'); set(gca,'FontSize',25,'FontWeight','bold','line
xlabel('Time (ms)','FontWeight','bold','FontSize',30,'FontName'
ylabel('Output voltage (V)','FontWeight','bold','FontSize',30,'F
grid on;

linkaxes([plt1,plt2],'x')
```

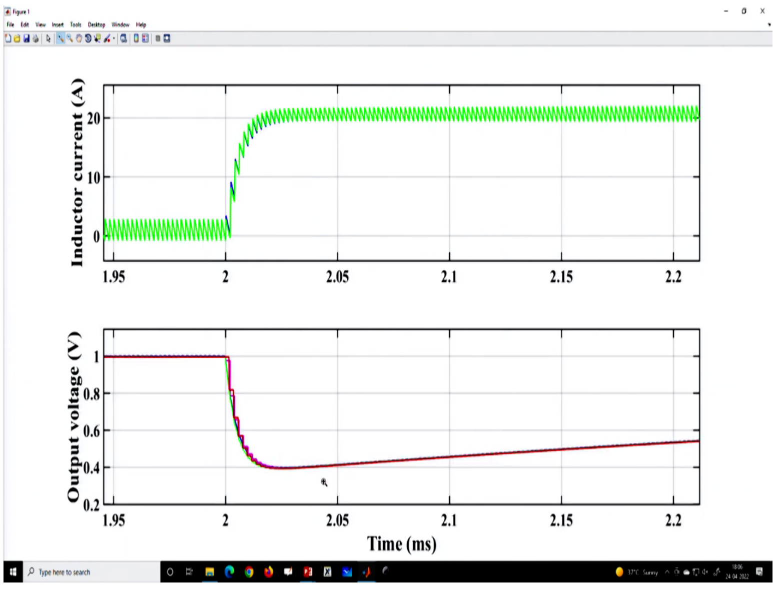
And, in the plot command, we want to change the color. So, we will use green color and red one and let us run. So, this is with the delayed version ok.

(Refer Slide Time: 04:41)



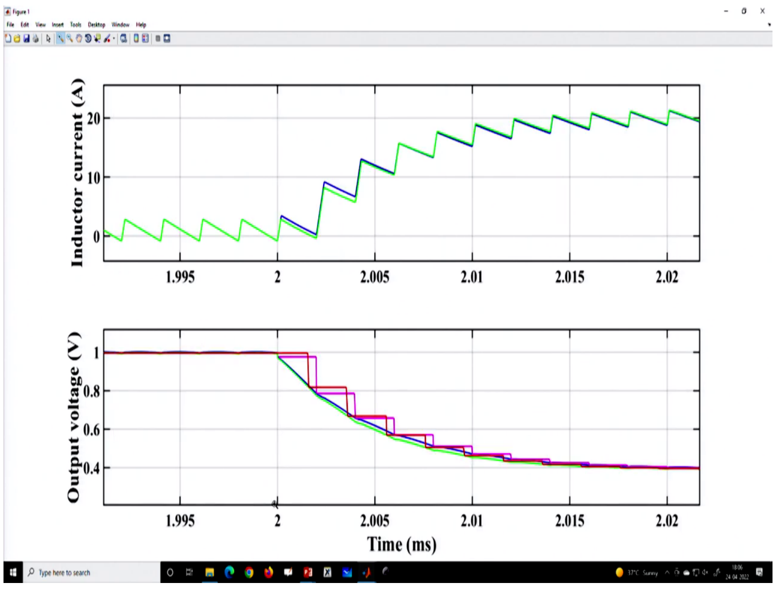
So, we will see the response is not fundamentally different.

(Refer Slide Time: 04:46)



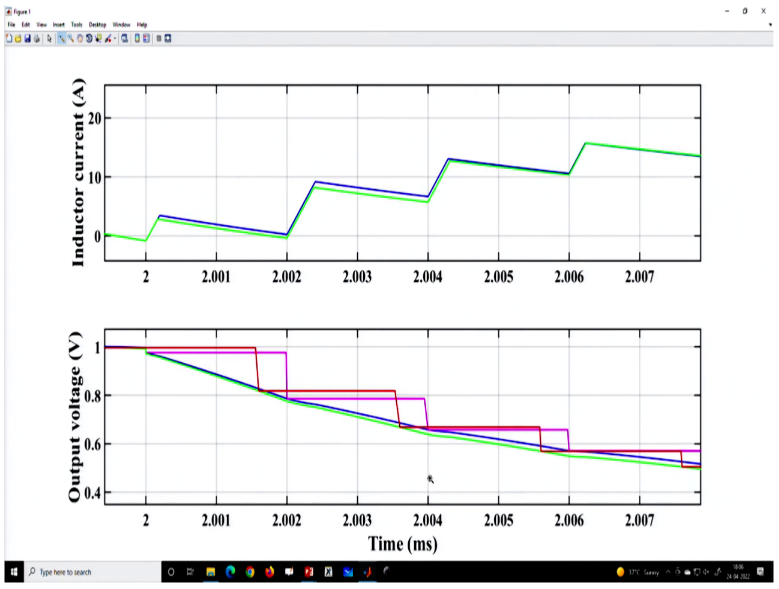
Because it is already kind of a sluggish response.

(Refer Slide Time: 04:48)

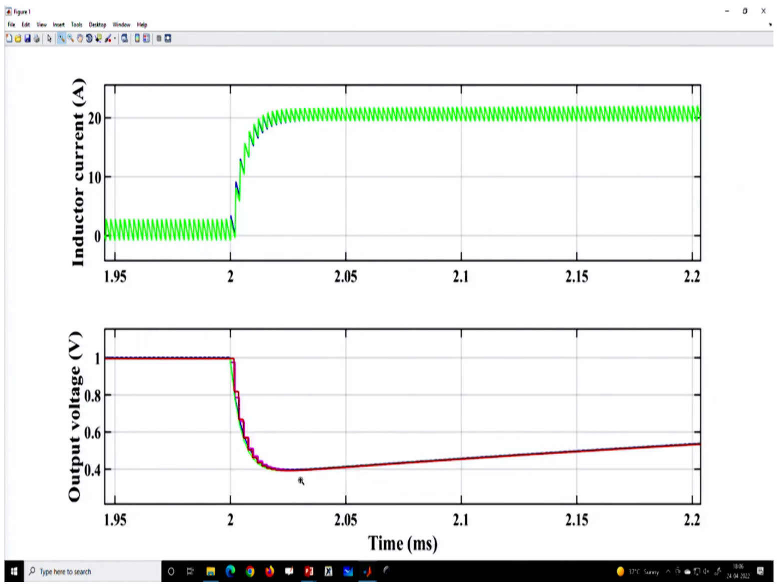


But, the sampling point got is different.

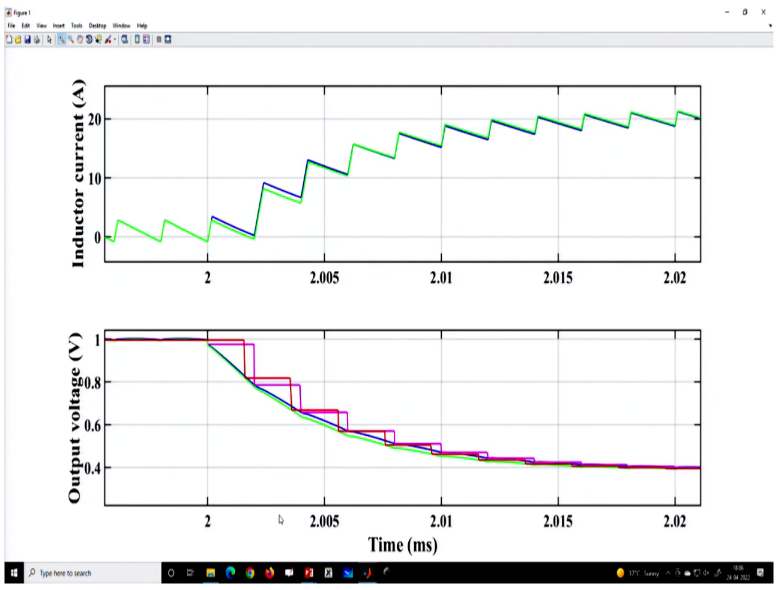
(Refer Slide Time: 04:52)



(Refer Slide Time: 04:59)

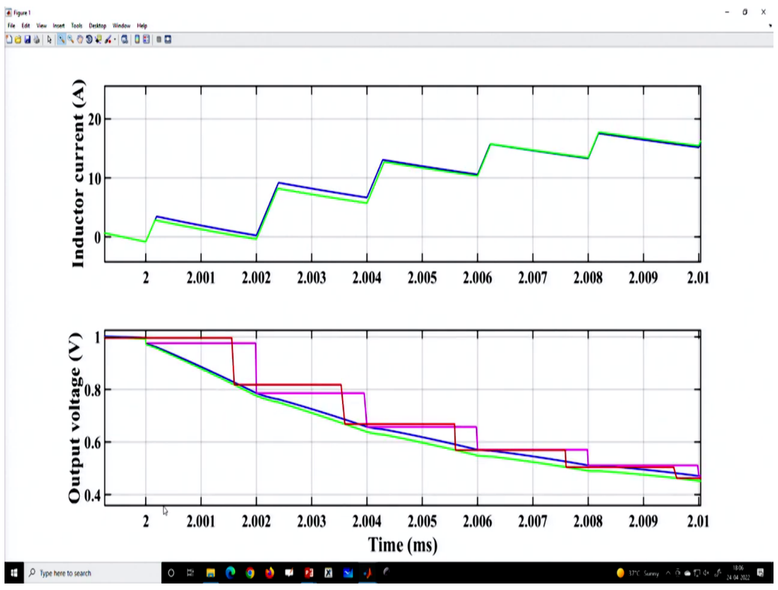


(Refer Slide Time: 05:01)



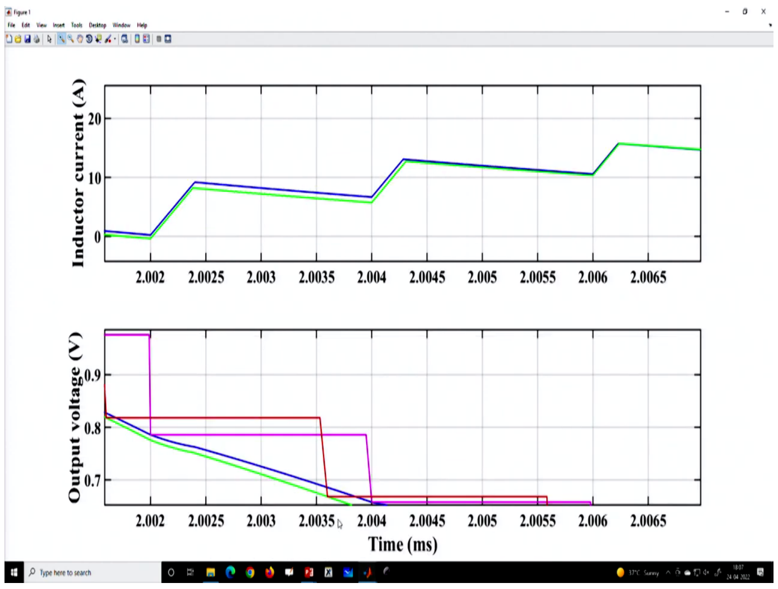
So, if you see the green waveform and, if you take these two particular update mechanisms you see the magenta one which we have drawn earlier; that means if we take only a few cycles.

(Refer Slide Time: 05:09)



You see the magenta one was getting upgraded at the switching point which was the earlier case. But, now the red one where we have introduced a sampling delay, and that got changed. So, we are getting we are sampling before the switch turns on. So, we have some around 0.1 t time and that can be considered for ADC conversion time and computational time. So, you can keep on changing.

(Refer Slide Time: 05:39)



So, the bottom line is this. By this block, we can actually; that means, by using this block we can customize a sampling point. And, this will also enable you to go to for constant on-time control or constant off-time control. The event-based sampling, the event can be created by this clock and that will sample the output voltage.

So, in summary in current mode control; means, we are talking about the analog current loop, but right now we are using this compensator as a transfer function right. But, imagine what will happen if your; suppose you take a PI controller; what does it look like? So, the PI controller you know is generally written like this, the output of the controller n is k p into V error voltage plus u I n.

And what is u I n? It is nothing but u I n minus 1 plus integral gain into error n. So, this can be simple if you take the transfer function here, you can get this. How to get this? You will get k p plus k i 1 minus z inverse ok and this block you will get. But, right now we are simply plugging this transfer function. But, what will happen if the z inverse; what is z inverse? z inverse is a unit delay, right?

So, what is the unit delay? That means it is defined with respect to some sampling clock. Now, if the sampling itself, if you want to customize the sampling edge; that means, you know we talked about we want to change the sampling point. Suppose, this is my sampling point, I want to shift to it here; I want to shift it here which is one possibility. Then, simply plug in this may not work, because if you go inside, if you go inside this block; it will ask for sampling time as if the whole computation happens at that edge.

But, we do not want, we want the computation to happen once the data of the ADC is ready. Since the ADC clock is shifted; so, naturally the conversion time will be there and your data will be ready after some time. So, you need to start computing after that. So, this transfer function will not allow customizing the point of computation or the point where two computations start. So, this cannot be done using just plug in this transfer function.

So, in summary, if you want to customize that I want to start computing once this sampling clock is the same as the ADC and the data is ready, then I will start computing. So, then you have to realize this. So, this u I n will be a register; that means you will have this kind of a register.

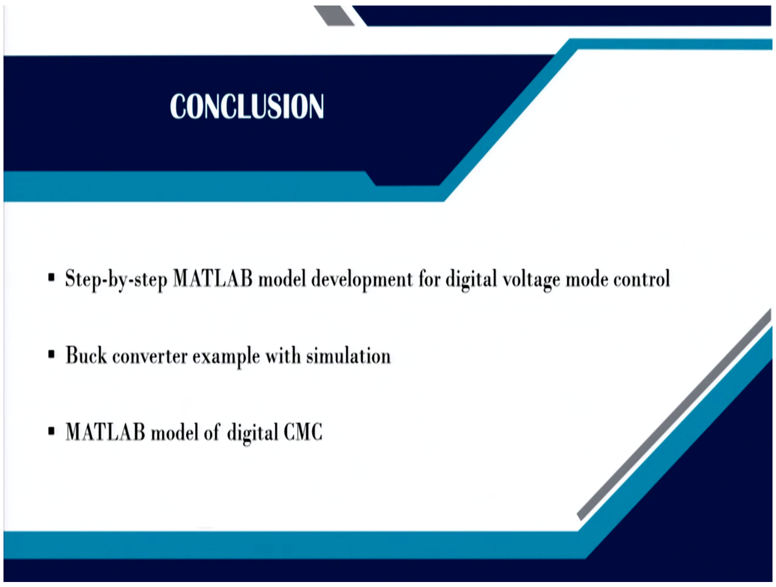
There will be a clock and the register will store and pass data. And, this register will behave like a z inverse block. But, whether this clock; means, we want to shift this register based on the edge of the clock, not by this transfer function. Another point we want to even change is the sampling rate, because if you go for event base sampling, the sampling clock the edge will differ; may be transient their edge-to-edge will be different.

Then, under a steady state, if there is any small perturbation, there will be a difference in the event base. So, for such cases the z inverse is not very straightforward, it cannot be implemented like this. So, there is no fixed sampling rate. So, we need to go for something called difference equation-based implementation; that means we need to go for difference equation.

That means, we need to implement; means, finally, the thing we need to implement this transfer function using a difference equation. How it is done? So, one of the examples I have shown here ok. So, if you use this differential equation and then this will be a delayed version of this; which means these two can be if this is your u n and u n minus 1; this register can be used to delay this block. And, at what point will delay with respect to which clock can be synced with this sampling clock here?

So, ultimately whatever we are implementing to digital control straight away using this block is not a good idea. So, in the next lecture we will implement, we will go further down and implement this controller using a difference equation.

(Refer Slide Time: 11:04)



CONCLUSION

- Step-by-step MATLAB model development for digital voltage mode control
- Buck converter example with simulation
- MATLAB model of digital CMC

So, in summary, we have discussed some aspects of the digital voltage mode control MATLAB model that was also described in the previous lecture. We have shown a buck converter example and in this lecture, we also talked about digital current mode control implementation.

And, we have realized that the direct transfer function, importing the direct transfer function is not a viable solution, particularly when we are considering the customized sampling edge as well as the varying sampling plot. So, we need to go for a different equation-based controller implementation. So, in the next lecture, we will talk about that aspect. So, for today that is it for today.

Thank you very much.