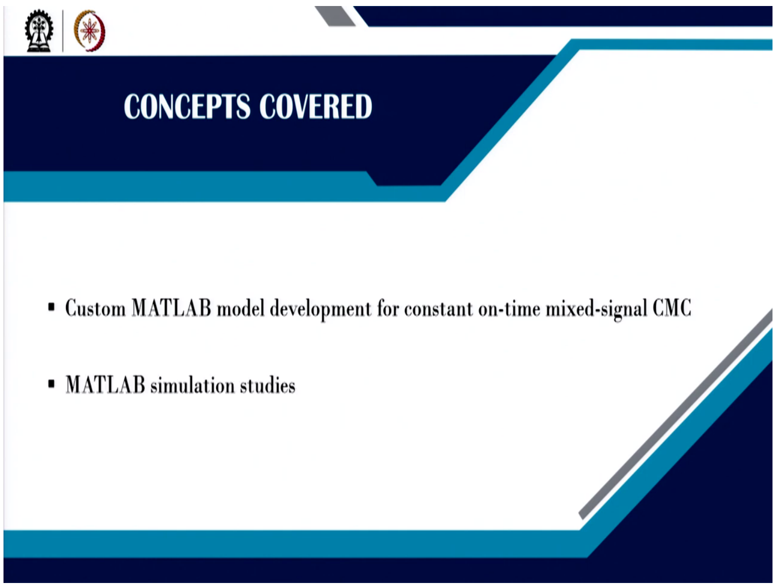
Digital Control in Switched Mode Power Converters and FPGA-based Prototyping

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

Module - 03
MATLAB Custom Model Development under Digital Control
Lecture - 28
MATLAB Model Development for Constant-On Time Control

Welcome, today we are going to talk about digital current mode control using constant-on-time modulation.

(Refer Slide Time: 00:34)



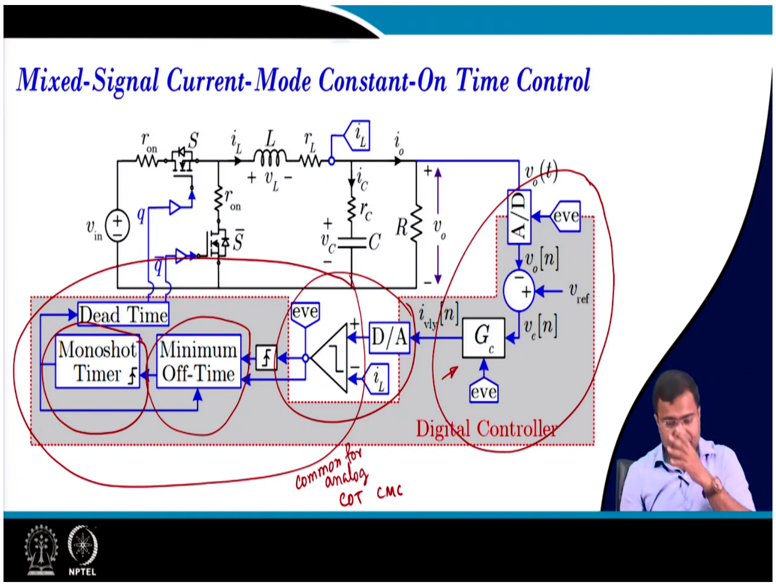
CONCEPTS COVERED

- Custom MATLAB model development for constant on-time mixed-signal CMC
- MATLAB simulation studies

And this lecture is slightly different from the previous lecture because, till the previous lecture, we talked about fixed frequency control. Now, we are going to talk about variable frequency control. In the last week, we have learned different architectures like constant on-time digital current mode control, and mixed-signal current mode control, then we also learn constant off-time mixed-signal current mode control.

So, we are extending the architecture that we have learned in the last week, we want to develop a MATLAB simulation model and we want to consider a few case studies.

(Refer Slide Time: 01:07)


Mixed-Signal Current-Mode Constant-On Time Control
Dead Time
Monoshot Timer
Minimum Off-Time
D/A
A/D
eve
Digital Controller
common for analog COT CMC

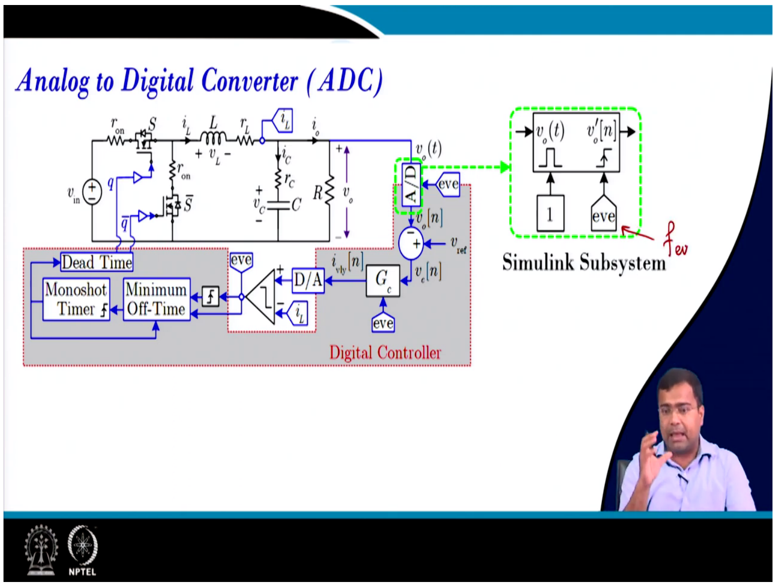
So, first, if you talk about mixed signal constant current mode constant on-time control. So, this process is the same, but the way of sampling is different; that means, here we will be using the event-based sampling that we have discussed; that means, we will be talking about event-based sampling, not uniform sampling because it will lead to instability and then we need to consider; that means if we consider the analog because in you know; in that architecture we have discussed the monoshot timer is used for constant on-time modulation.

In addition to that we need to consider a minimum of time because for practical realization this is something mandatory for any commercial product that we use and here the current loop is in analog; that means, we need a D to A converter.

So, after the controller output we have to use a D-to-A converter and then the analog comparator is there and the output of the comparator is used as an event trigger clock and then the output of the comparator. So, up to this point even if you consider this point it is the same as the constant-on-time current mode control that we have discussed in our previous NPTEL course and that we have also explained in the architecture exploration.

So, this part if you consider any analog; that means, you know if you consider any analog you know constant on-time current mode control that we have discussed is the same, it is common for this block is common, but only this digital block A to D converter and the DAC, these are the new thing and that also requires sampling.

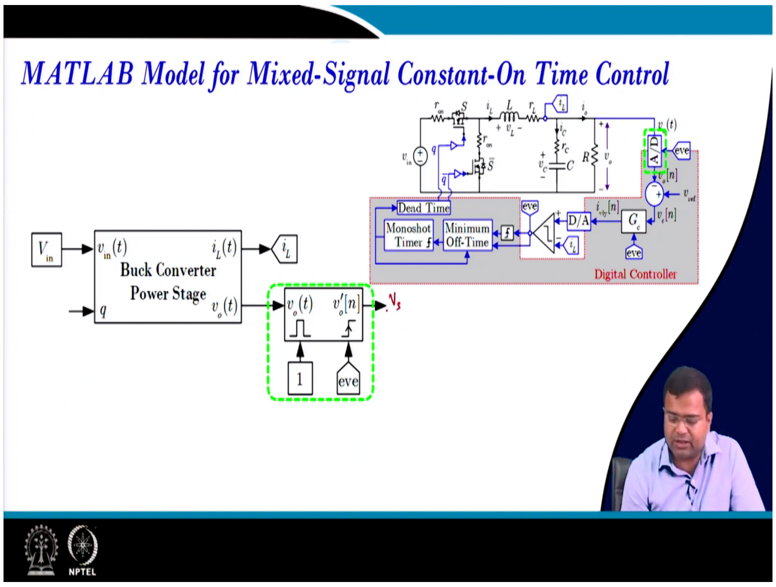
(Refer Slide Time: 02:56)



Analog to Digital Converter (ADC)

So that means, first analog to digital converter and we have discussed that we have to use an event trigger ADC and this even signal we need; that means, we need an event clock to sample this ADC because it has to be sampled non-uniformly.

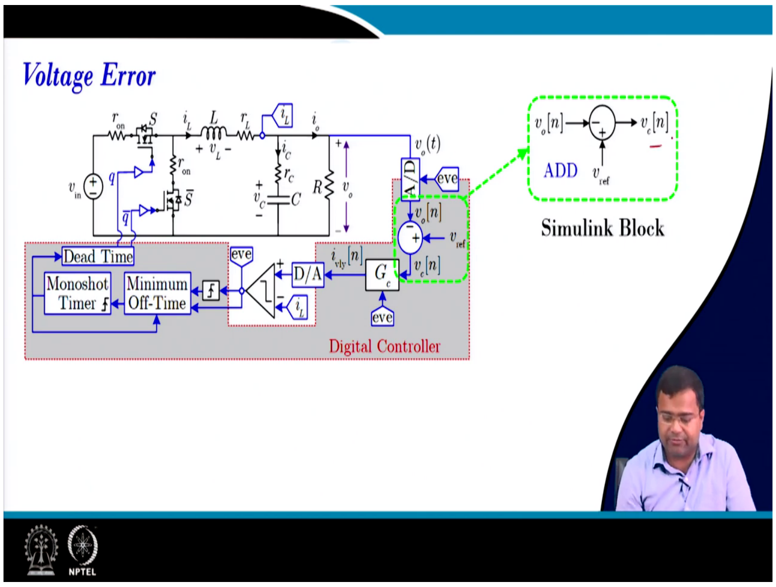
(Refer Slide Time: 03:11)



MATLAB Model for Mixed-Signal Constant-On Time Control

So, that means, we will use the first and event trigger A to D converter to sample the output voltage. So, this is your sample output voltage, you have to use a sample output voltage.

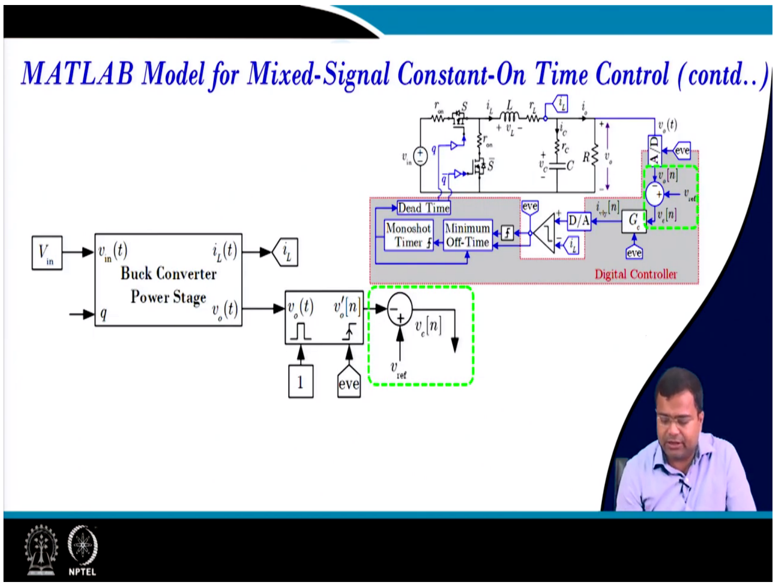
(Refer Slide Time: 03:24)



Voltage Error

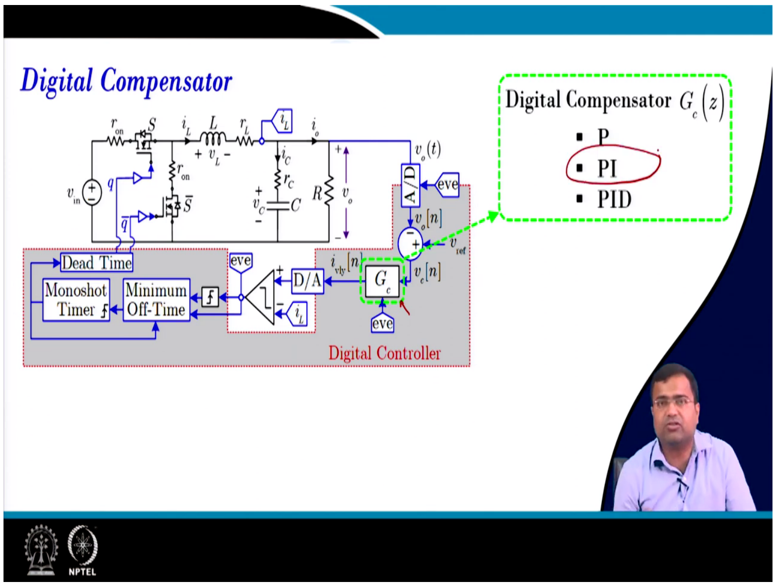
Then after that sample output voltage will be compared with the reference voltage will generate the error voltage, then this error voltage will go to the next compensator the controller which can be P, PI, or PID.

(Refer Slide Time: 03:32)



MATLAB Model for Mixed-Signal Constant-On Time Control (contd..)

(Refer Slide Time: 03:35)



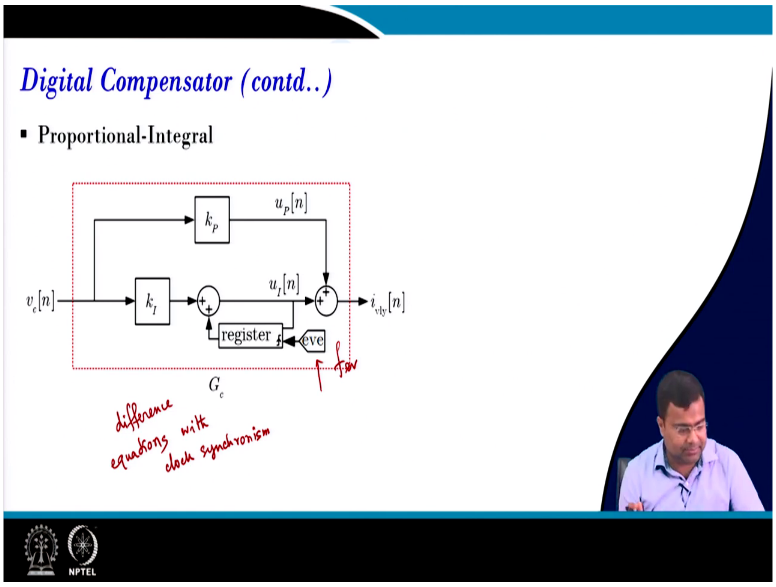
Digital Compensator

Digital Compensator $G_c(z)$

- P
- PI
- PID

But since it is a current mode control we are typically considered the PI controller.

(Refer Slide Time: 03:47)



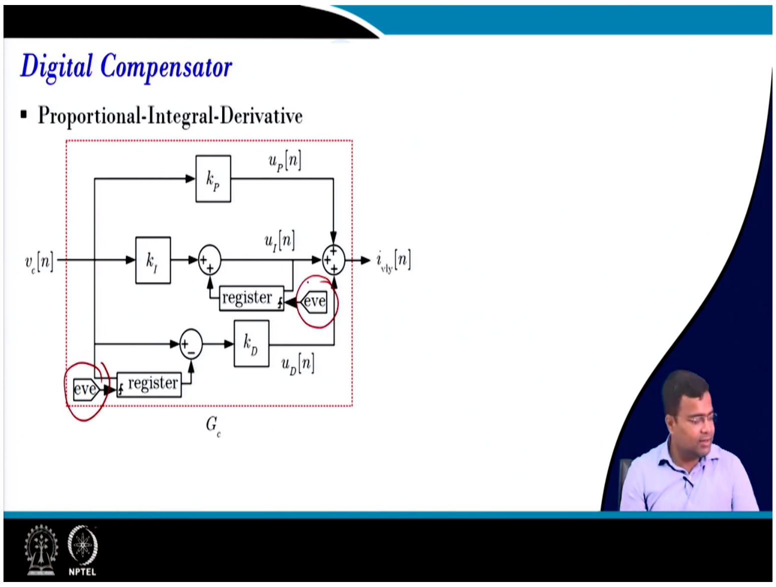
Digital Compensator (contd..)

- Proportional-Integral

So, now in this PI controller again we have to realize using this block is the same as whatever we have discussed for the previous two classes, for full digital as well as mixed signal as well as voltage mode control. We have used using difference equation right difference equations and difference equations with clock synchronism.

The only difference in the earlier case we used a fixed-frequency clock. Now, we will be using an event clock here which is the event clock, but the block diagram realization is the same.

(Refer Slide Time: 04:34)

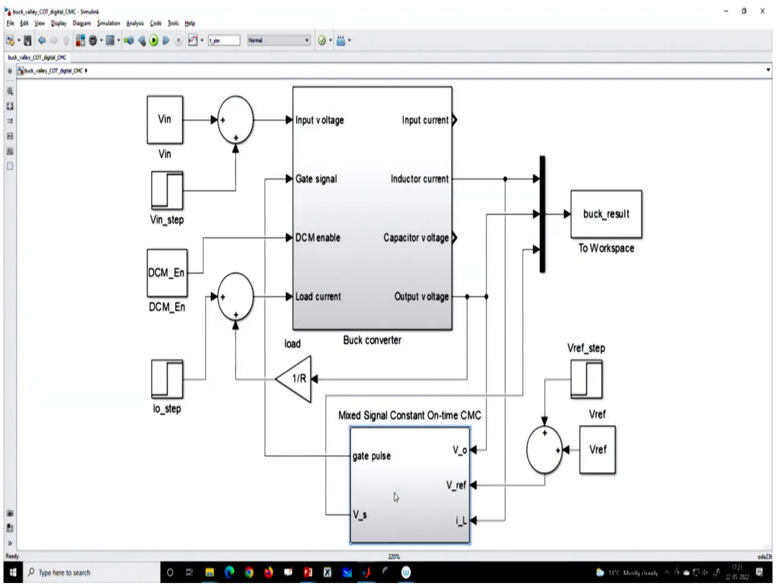


Digital Compensator

- Proportional-Integral-Derivative

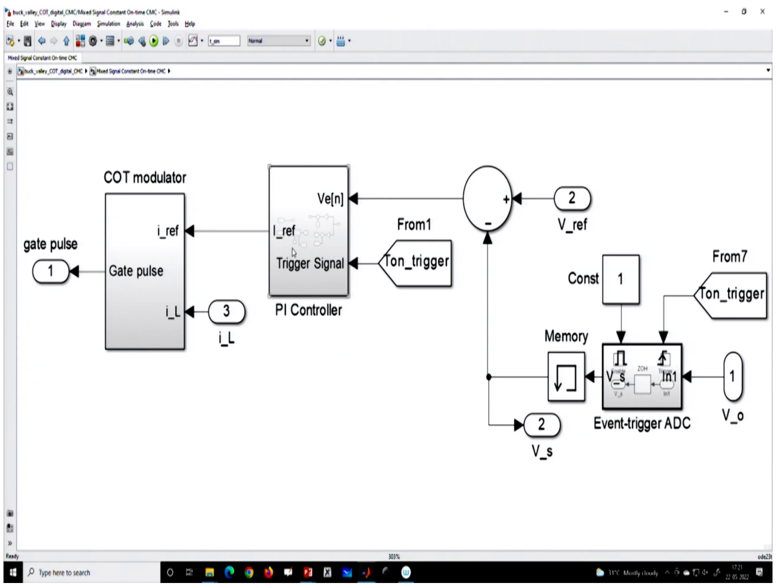
That means if we go to you know this is you can if you use a PID controller, the same thing only this clock is different. So, this is the vital point here. So, now, we are coming to the MATLAB block.

(Refer Slide Time: 04:46)

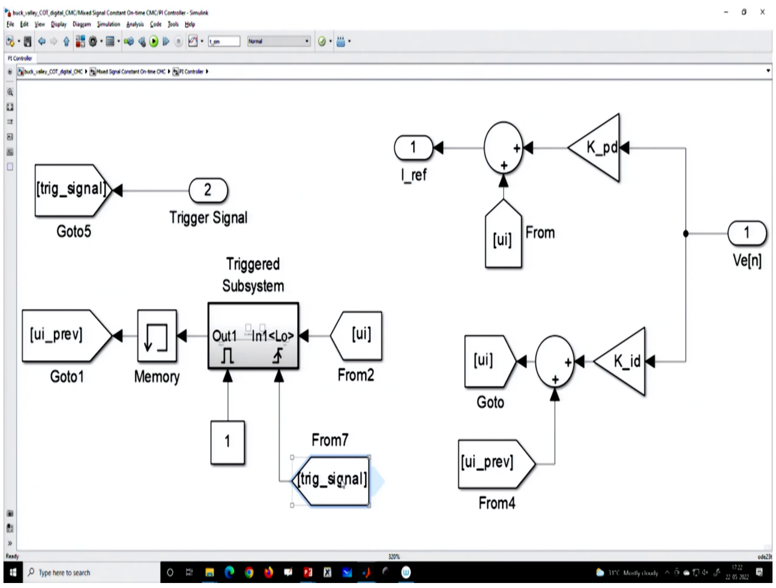


So, in this block, we want to implement. So, everything else is the same you have a power stage block which is common and is a controller.

(Refer Slide Time: 04:57)



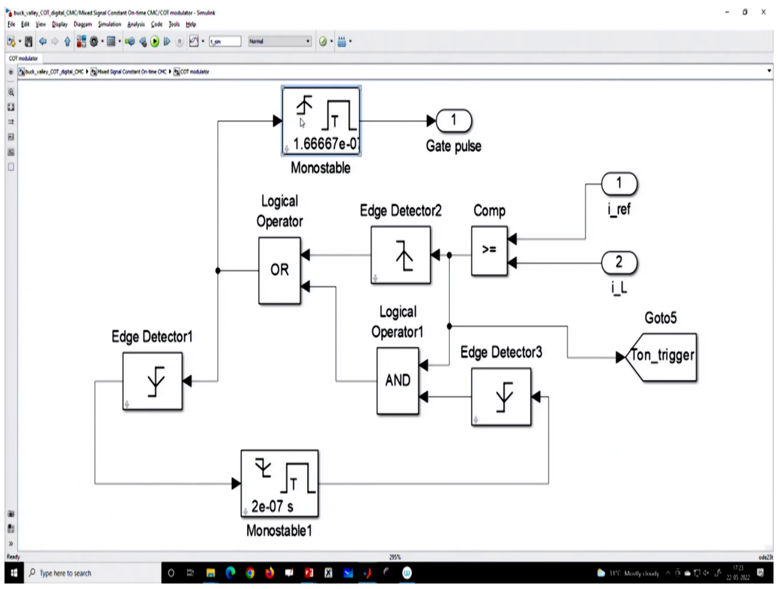
(Refer Slide Time: 05:00)



So, here if you see the PI controller the realization is exactly what we have realized for the last two classes; that means, we have realized in the same way the mixed signal current mode control as well as the full digital current mode control or voltage controller was PI controller and we have used this clock synchronized difference equation but the difference is the trigger signal.

There we have used a uniform clock maybe with a customized time timing that we can delay the clock, but there is a customized clock that has a fixed frequency that we have used, but here you are using an event-based clock. How is it generated?

(Refer Slide Time: 05:38)

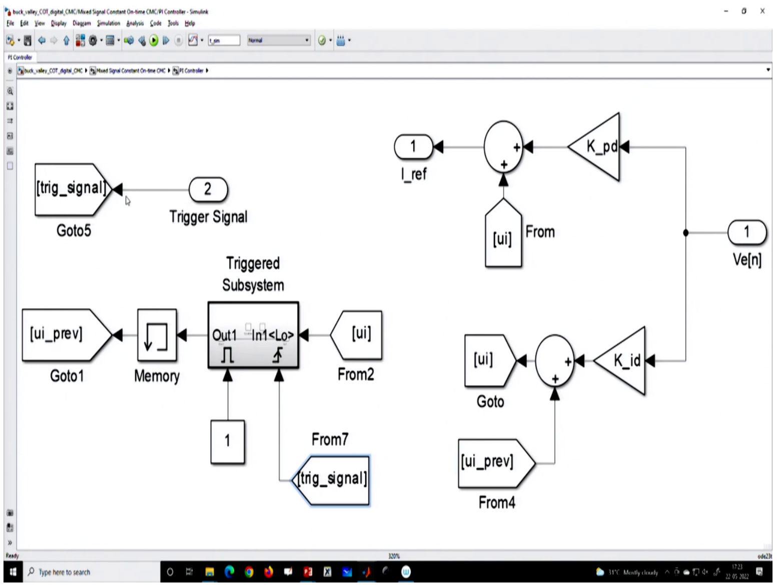


So, if you go inside this the modulator will come constant on time modulator. So, based on where; that means when the inductor current hit the reference current.

That is the trigger point; that means, whenever the inductor current; that means, reaches the valley point that is the trigger point; inductor current reaches the valley point there you trigger and that trigger is used to update you know to sample the ADC update the controller, but we can also shift the time because once this edge comes then it will trigger it will enable the mono-shot timer. Since the mono-shot timer's overall duration is fixed and is defined by the designer.

Then we can generate this trigger signal anytime within that on time because that is in our hand, but off time is not in our hand because off time is governed by the comparator, but whenever here we are taking just whenever the mono-shot timer is enabled, we are generating the trigger pulse, but we can slightly delay this trigger pulse may be in between the on time, in between the mono-shot timer; that means when the mono-shot timer is enabled it is enabled for on time. So, we can take half the on-time and generate the trigger signal. So, it is customizable.

(Refer Slide Time: 07:03)

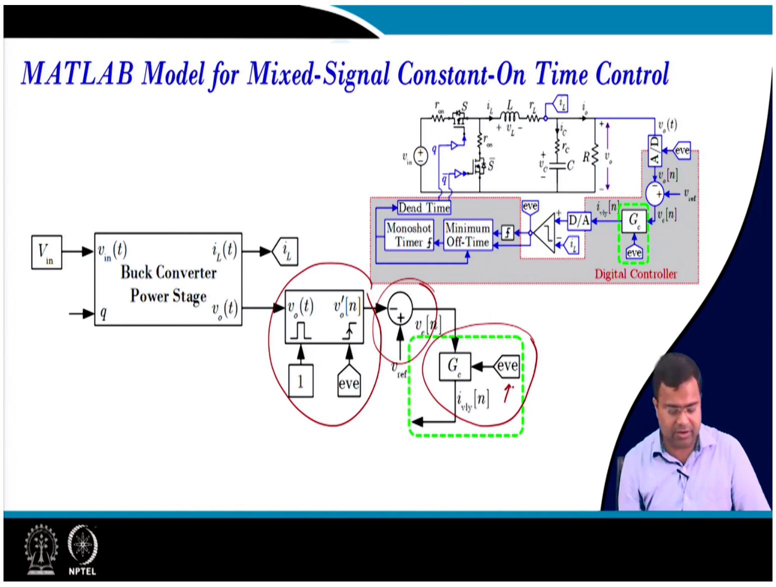


Next, once we generate that trigger pulse is used to generate this PI controller. To enable this PI controller and the same trigger pulse is used for the A to D converter, voltage loop for the controller computation here because we are not considering for the time being any delay, but one can use delay, but here we want to make I want to first give you you know the concept, how to implement rather than going too much into advanced detail.

And this block is common, this block even if you take an analog constant on-time control this simple like you know it takes the inductor current, it is in the analog domain and reference voltage reference current which is coming as if it is like a D to A converter. In Simulink since it is a floating point data, but in actual realization, you need to consider D to A converter and that means, both i_ref and i_l are analog.

So, you can use an analog comparator, and then it has an inbuilt mono-shot timer and this is used for minimum off time; that means, every constant on-time control has to have a minimum off-time. Now, if you go back to our realization; that means, we have discussed how to design because we are using current mode control. So, we will not use the PID controller, but I am showing the generic realization of the PID controller and we are using this PI controller. So, this is a MATLAB block diagram.

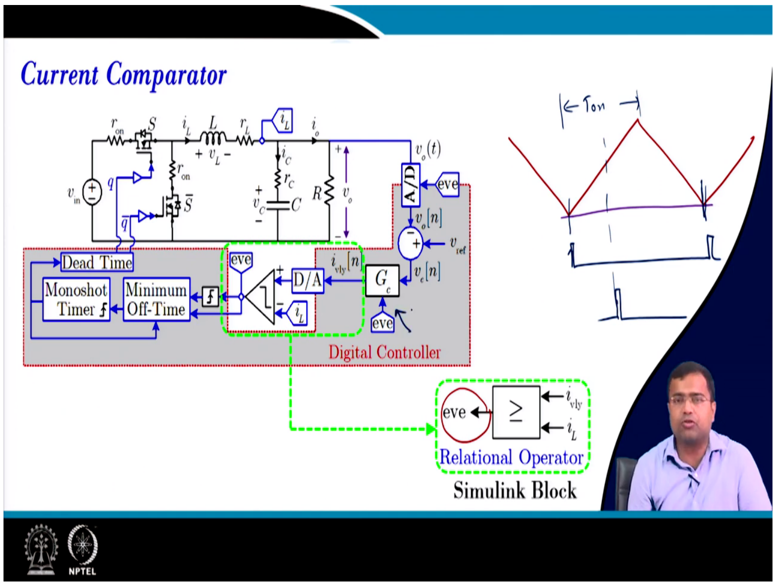
(Refer Slide Time: 08:23)



MATLAB Model for Mixed-Signal Constant-On Time Control

Once you have this event trigger ADC, then you have an error voltage, then you can realize this controller, and this controller is updated by the event clock.

(Refer Slide Time: 08:35)



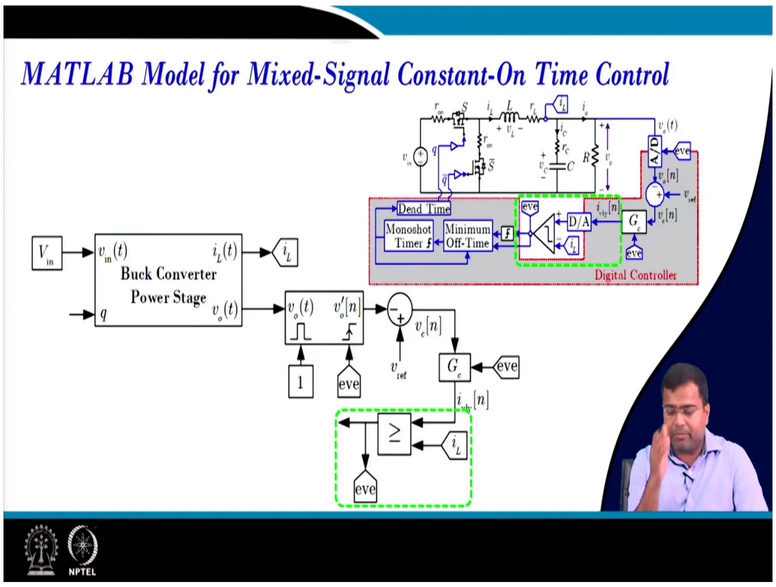
Current Comparator

The current comparator, since it is in analog. So, it is simply just a comparator in MATLAB, and this comparator outputs whenever the inductor current; that means, if we draw the inductor current, this is the inductor current and if we draw the waveform; that means if we draw this valley current waveform let us say this is valley current.

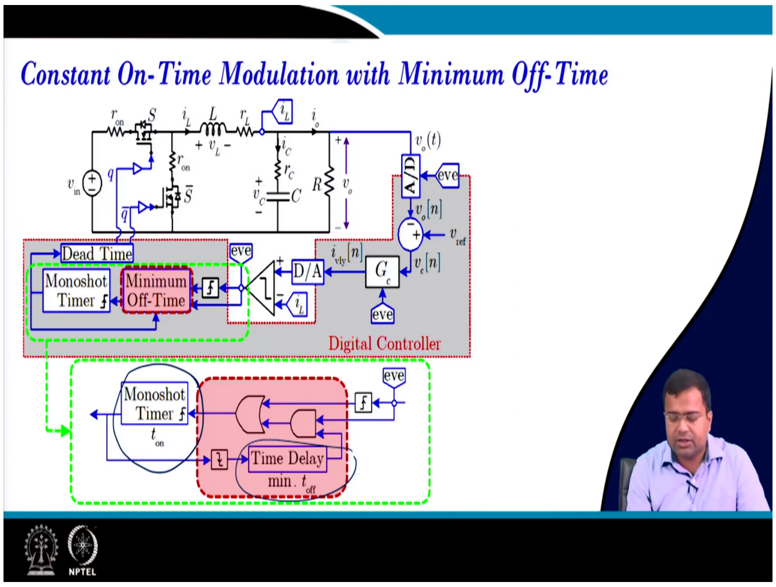
So, whenever this hit this current as if here we are generating this event clock; as if here we are generating the event clock. So, I think we should update the diagram, sorry yeah it should be like this. It will be compared after that it may be updated. So, we should use this and if there is any change let us say. So, this is just a valley point and whenever it touches the valley point we are presently taking here as the edge clock edge.

But, since the on time is fixed this is the total on the time it is fixed, you can generate a clock here, no issue because this is in our hands. So, that means, we can customize this clock and ultimately that clock will be used for your controller computation as well as the ADC conversion ok.

(Refer Slide Time: 10:05)



(Refer Slide Time: 10:07)

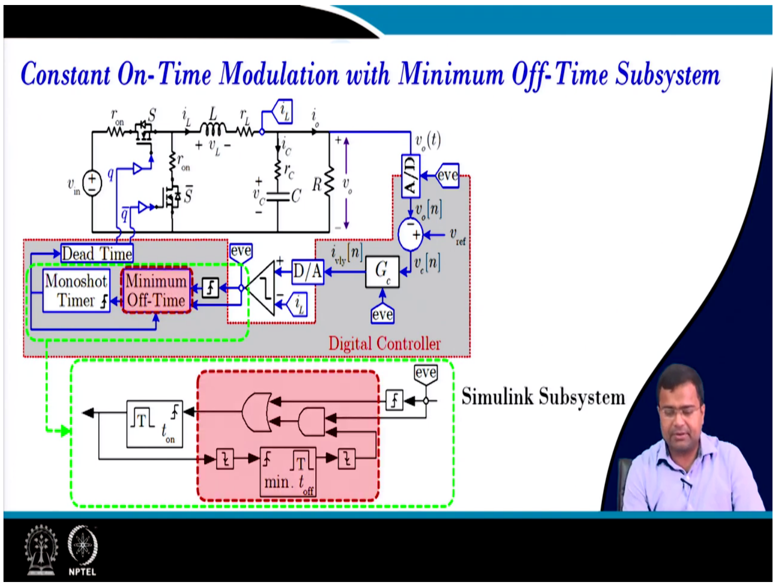


Constant On-Time Modulation with Minimum Off-Time

So, the MATLAB generation that comparator then goes to the mono-shot timer. So, that means, after the comparator output it goes to the mono-shot timer, and the mono-shot timer will be loaded on time and it needs to have a minimum of time.

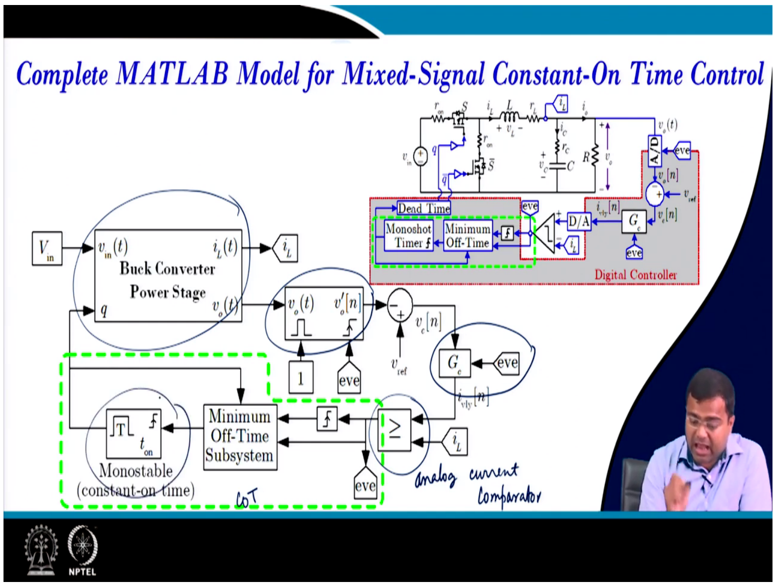
That means this is the minimum off-time realization and this we have discussed. You know this implementation in our last week's architecture exploration. So, here in MATLAB, we are going to implement this.

(Refer Slide Time: 10:34)



Constant On-Time Modulation with Minimum Off-Time Subsystem

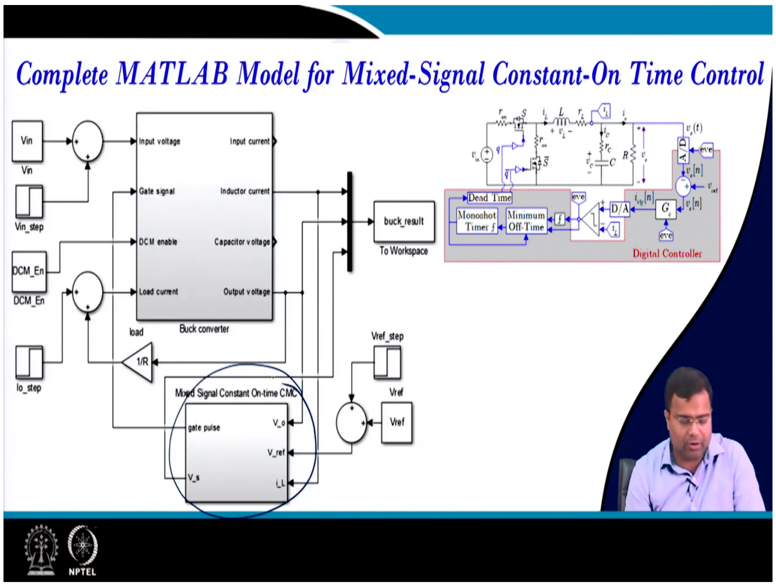
(Refer Slide Time: 10:37)


Complete MATLAB Model for Mixed-Signal Constant-On Time Control
Buck Converter Power Stage
Minimum Off-Time Subsystem
Monostable (constant-on time)
Dead Time
Monoshot Timer
Minimum Off-Time
Digital Controller
analog current comparator
CoT

That means this is a subsystem and we are going to complete the MATLAB implementation we are going to do; that means, the first thing our power stage then we have an event trigger ADC, then we have an event trigger controller, then we have this regular comparator analog current comparator this is analog current comparator.

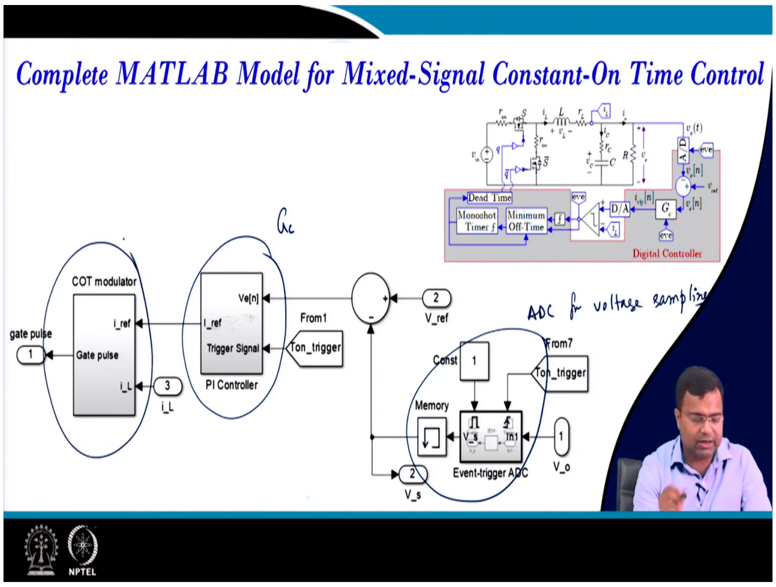
I would say it is a voltage comparator, but sense current then the output of the comparator goes to this is our constant on time; constant on time modulator which consists of the monostable mono-shot timer as well as a minimum off time.

(Refer Slide Time: 11:23)


Complete MATLAB Model for Mixed-Signal Constant-On Time Control

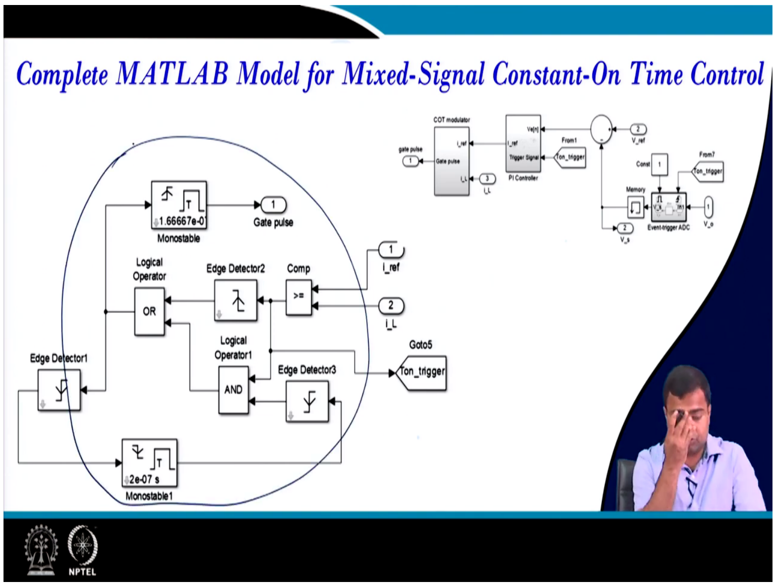
So, this is the total complete MATLAB block diagram of the constant on-time mixed signal current mode control.

(Refer Slide Time: 11:28)



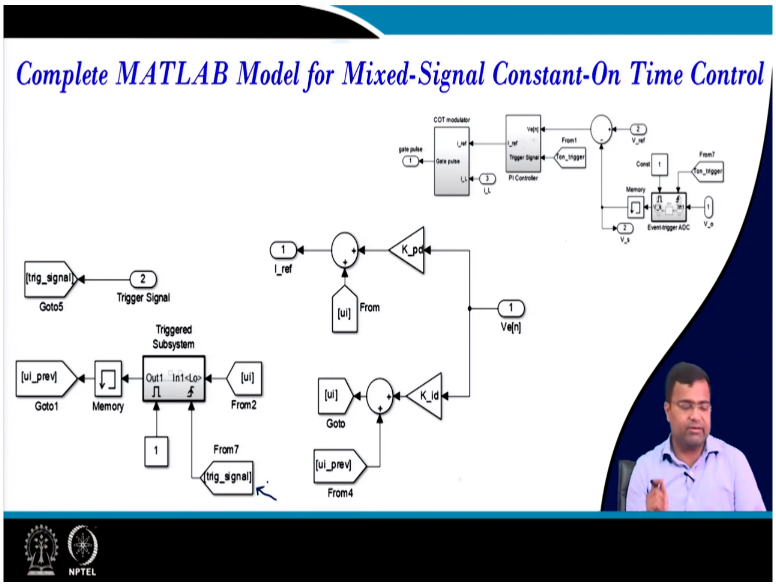
Here if you go inside this block; that means, I am talking about if you go inside this block, first is the event trigger A to D converter. So, this is the ADC for voltage sampling then this is for the G c controller, again it is the event trigger voltage comparator and this is the constant on-time modulator.

(Refer Slide Time: 11:59)



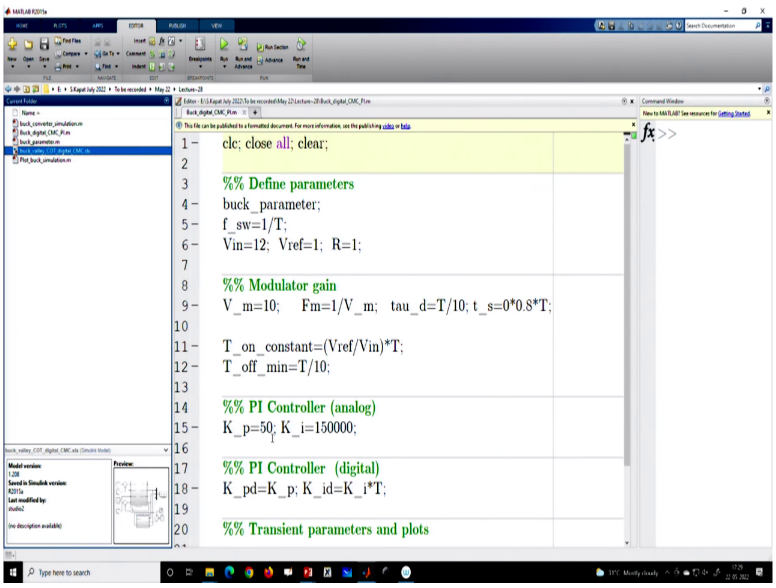
So, this is a constant on-time modulator and, this part is what we have discussed in our previous NPTEL course and this is we have already discussed in the last two classes that how to realize the controller PI controller using difference equations and using our customized clock. So, here the clock is the event-driven clock.

(Refer Slide Time: 12:07)


Complete MATLAB Model for Mixed-Signal Constant-On Time Control

So, if you go to the MATLAB now this is a complete realization and this is the if you go to this block.

(Refer Slide Time: 12:31)

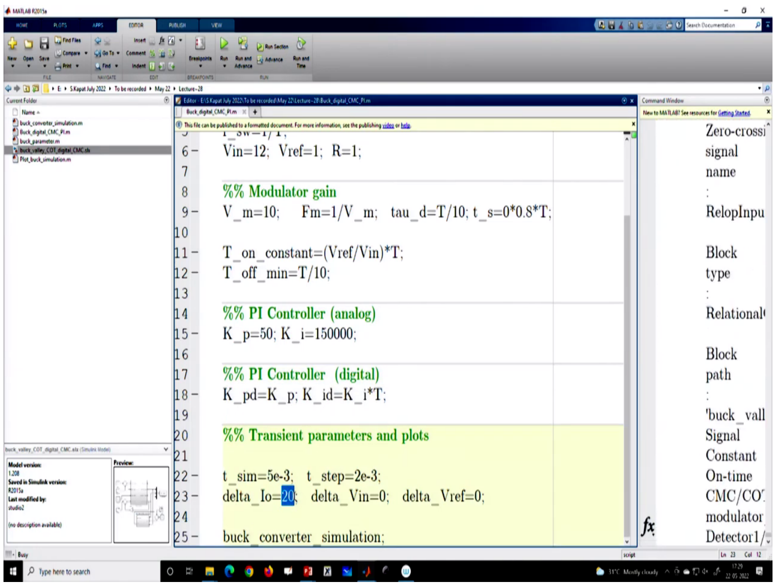

Editor - E:\S.Kapat July 2022\To be recorded\May 22\Lecture-28\Buck_digital_CMC_PI.m

```
clc; close all; clear;

%% 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*0.8*T;

T_on_constant=(Vref/Vin)*T;
T_off_min=T/10;

%% PI Controller (analog)
K_p=50; K_i=150000;

%% PI Controller  (digital)
K_pd=K_p; K_id=K_i*T;

%% Transient parameters and plots
```

So, here we are using constant on time. So, you can customize the on-time, and minimum off time you can provide as you wish and you can use a PI controller, this is the analog PI controller, and corresponding digital values are given we have already shown the total block diagram the PI block diagram this is a modulator block diagram.

And, then these are the event triggers A to D block. Now, we want to show a simulation case study.

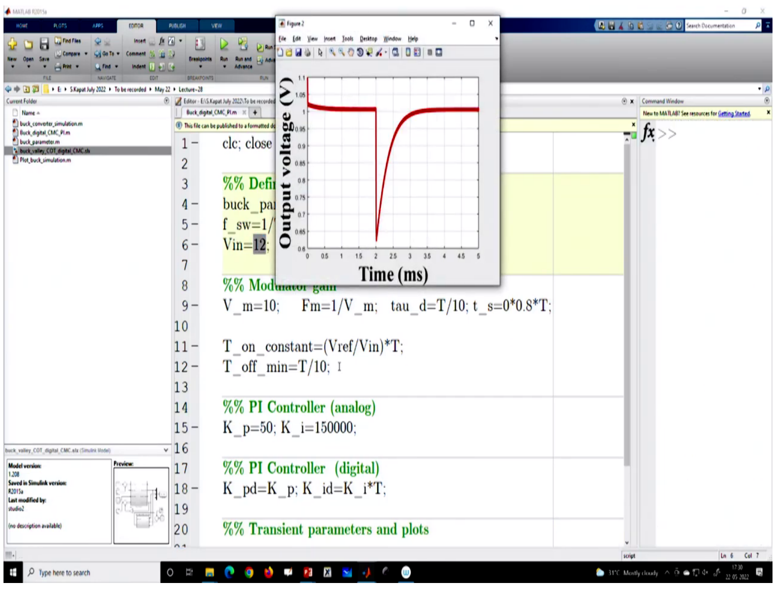
(Refer Slide Time: 13:01)



```
Vin=12;  Vref=1;  R=1;

%% Modulator gain
V_m=10;     Fm=1/V_m;   tau_d=T/10; t_s=0*0.8*T;

T_on_constant=(Vref/Vin)*T;
T_off_min=T/10;

%% PI Controller (analog)
K_p=50; K_i=150000;

%% PI Controller  (digital)
K_pd=K_p; K_id=K_i*T;

%% Transient parameters and plots

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

buck_converter_simulation;
```

```
Zero-crossi
signal
name
:
RelopInpu

Block
type
:
Relational

Block
path
:
'buck_vall
Signal
Constant
On-time
CMC/CO
modulator
Detector1/
```

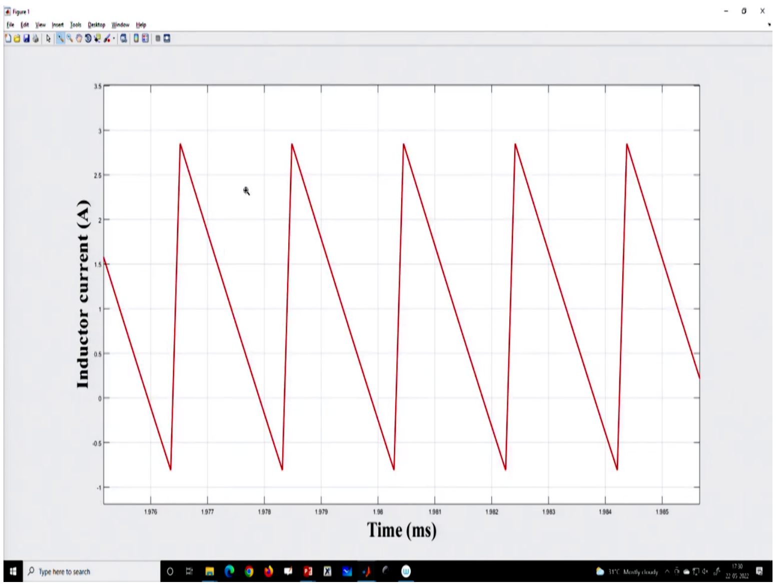
(Refer Slide Time: 13:13)



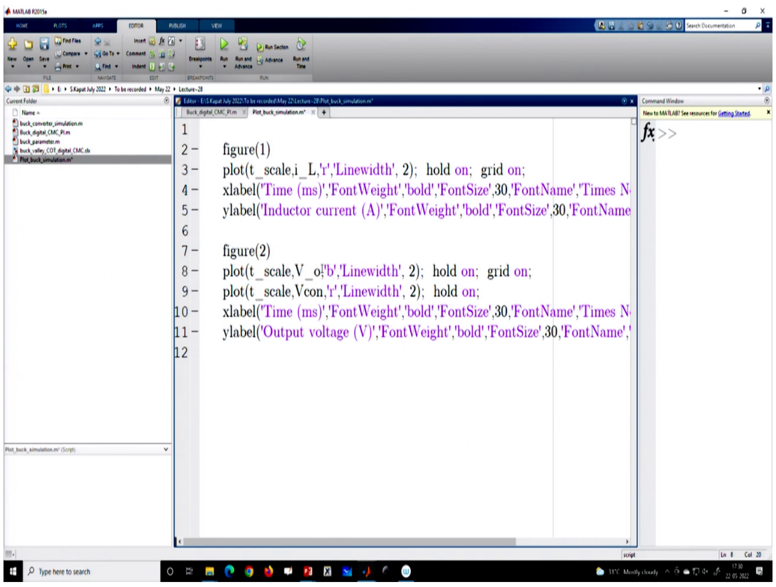
```
clc; close
 
%% Defi
buck_pa
f_sw=1/
Vin=12;

%% Modulator gain
V_m=10;     Fm=1/V_m;   tau_d=T/10; t_s=0*0.8*T;

T_on_constant=(Vref/Vin)*T;
T_off_min=T/10;

%% PI Controller (analog)
K_p=50; K_i=150000;

%% PI Controller  (digital)
K_pd=K_p; K_id=K_i*T;

%% Transient parameters and plots
```

So, I am just simulating you know it is for load transient response; that means, I have applied a 20-ampere load step-up transient, the input voltage is 12 volts and the output is 1 volt and initially it is started with 1 ampere.

(Refer Slide Time: 13:20)



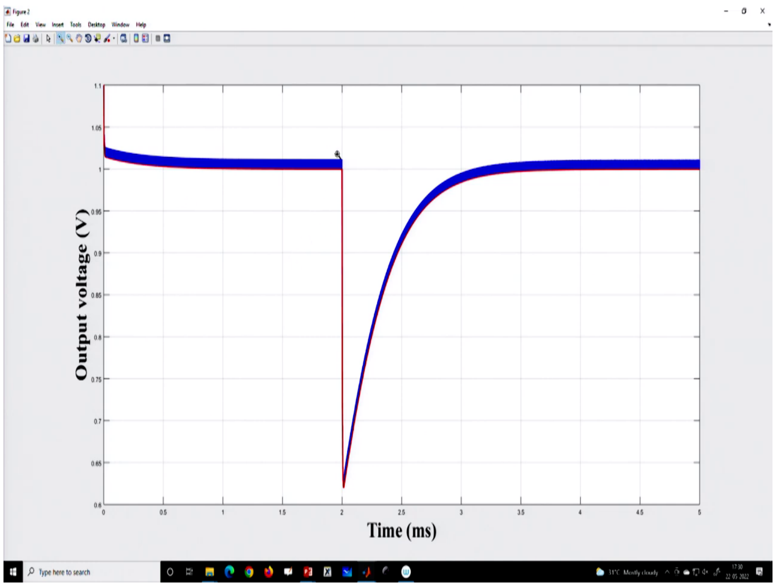
(Refer Slide Time: 13:35)



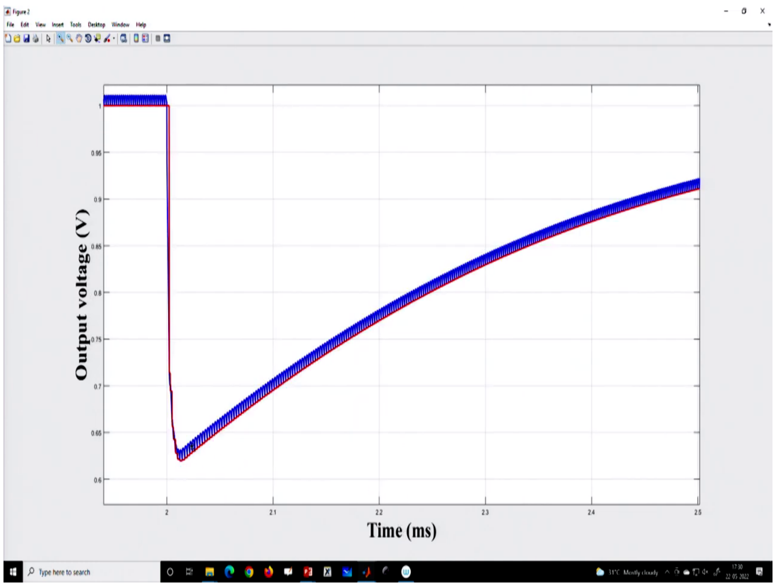
```
figure(1)
plot(t_scale,i_L,'r','Linewidth', 2);  hold on;  grid on;
xlabel('Time (ms)','FontWeight','bold','FontSize',30,'FontName','Times N
ylabel('Inductor current (A)','FontWeight','bold','FontSize',30,'FontName

figure(2)
plot(t_scale,V_o,'b','Linewidth', 2);  hold on;  grid on;
plot(t_scale,Vcon,'r','Linewidth', 2);  hold on;
xlabel('Time (ms)','FontWeight','bold','FontSize',30,'FontName','Times N
ylabel('Output voltage (V)','FontWeight','bold','FontSize',30,'FontName','
```

Now, if we see that, this is our constant on-time modulator ok.

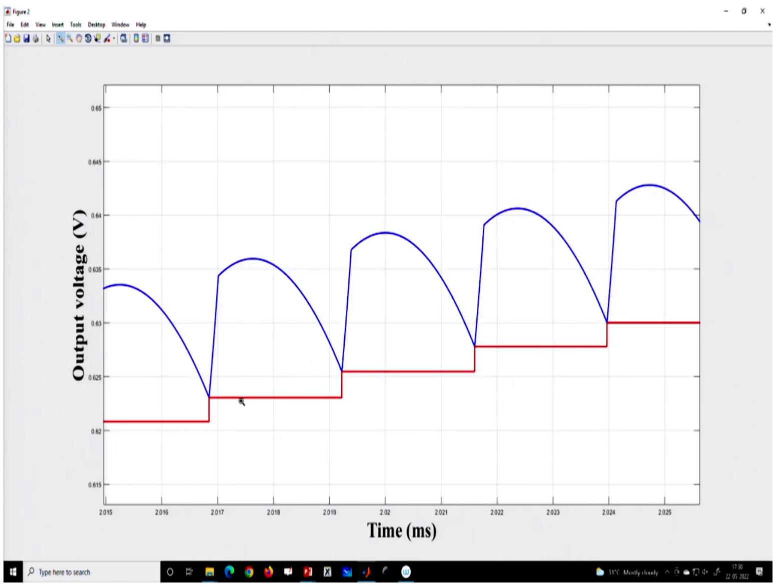
(Refer Slide Time: 13:48)



(Refer Slide Time: 13:52)

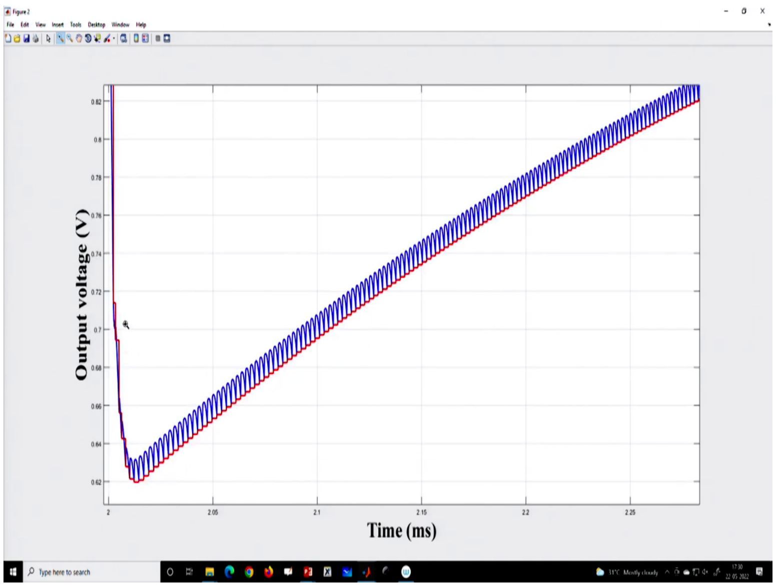


(Refer Slide Time: 13:55)

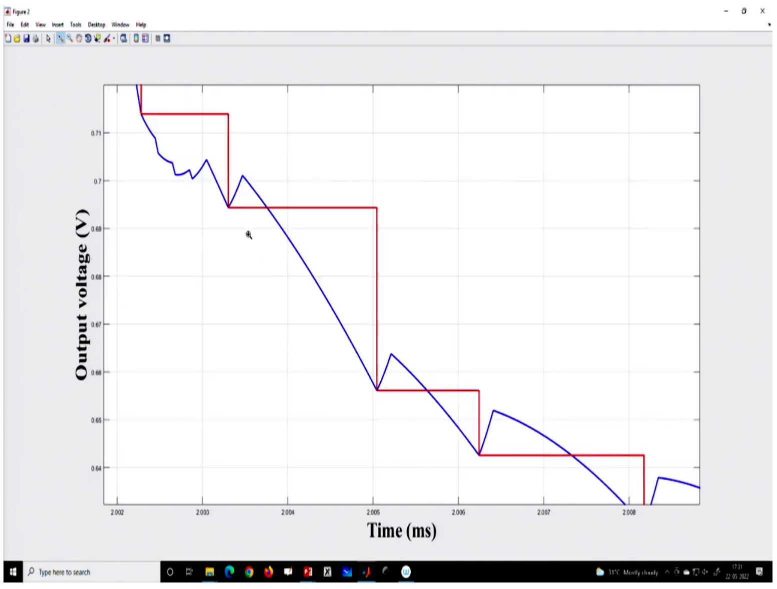


Inductor current and we want to show also let us say if we go to the plot comment, we want to enable the sample also; that means, let us this is blue color yeah. So, I am showing, here what I am doing here; if you go here whenever the switch is turned off; that means, here the inductor current hit then you take the sample ok. Even there can be variation in the time period.

(Refer Slide Time: 14:07)

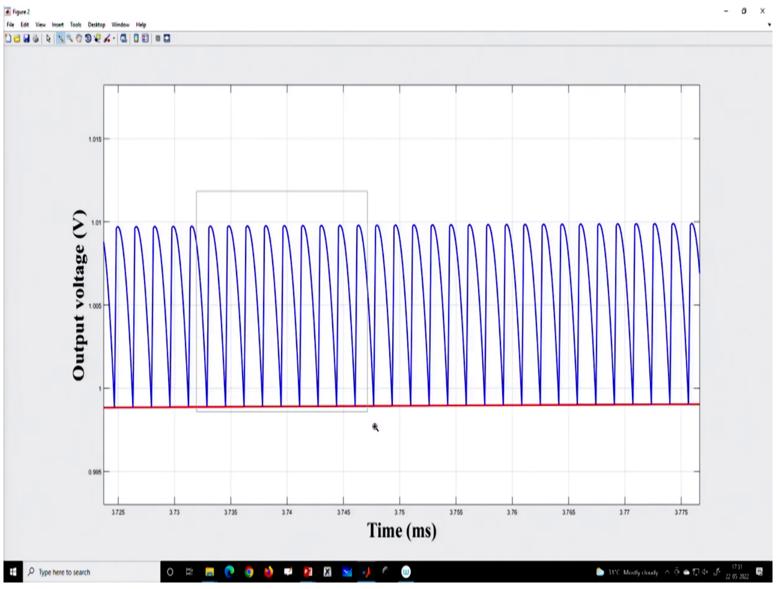


(Refer Slide Time: 14:11)



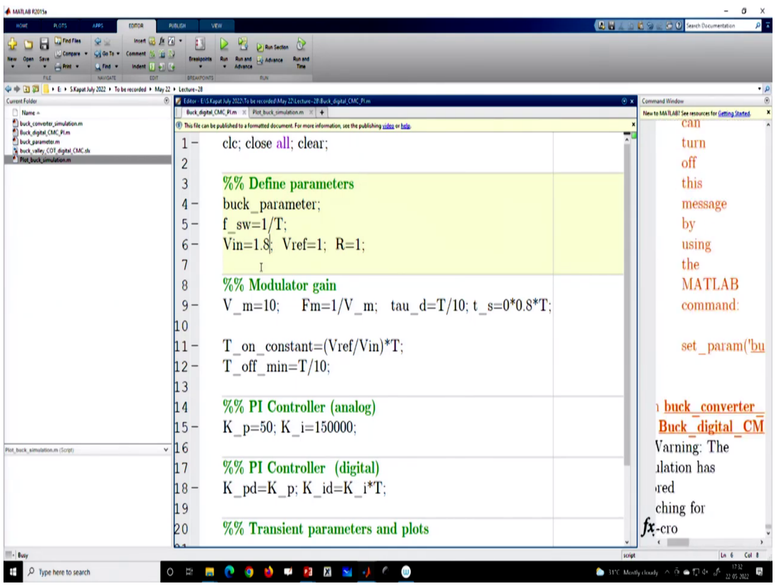
Because if you take particularly this case you see here the time is varying actually because on time is fixed, but the off time is varying, on time is fixed. So, as a result, the sampling time is also varying because it is a variable time control and we are using event-based sampling. So, the sampling is happening with respect to the event. So, naturally, the sampling rate is also changing when there is a transient.

(Refer Slide Time: 14:35)



But under steady state, they get more or less fixed and the switching frequency also gets locked if you want to regulate the switching frequency then we have to update the timing parameter, particularly the constant on time. So that you can achieve the desired switching frequency. So and whenever we will go for; that means, we can easily realize the constant on-time control ok. Now, I want to show an interesting case study.

(Refer Slide Time: 15:02)



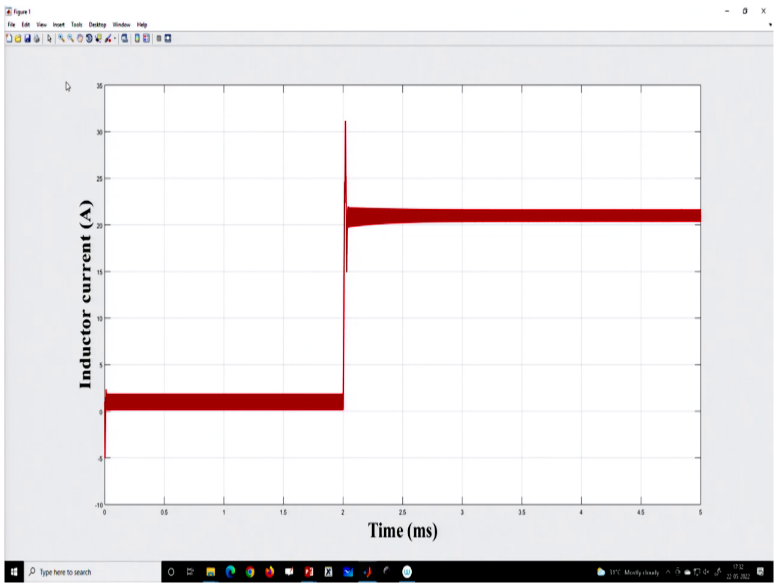
```
clc; close all; clear;

%% Define parameters
buck_parameter;
f_sw=1/T;
Vin=1.8;  Vref=1;  R=1;

%% Modulator gain
V_m=10;    Fm=1/V_m;   tau_d=T/10; t_s=0*0.8*T;

T_on_constant=(Vref/Vin)*T;
T_off_min=T/10;

%% PI Controller (analog)
K_p=50; K_i=150000;

%% PI Controller  (digital)
K_pd=K_p; K_id=K_i*T;

%% Transient parameters and plots
```

```
can
turn
off
this
message
by
using
the
MATLAB
command:

set_param('bu

buck_converter_
Buck_digital_CM
Varning: The
ılation has
ıred
ching for
-cro
```

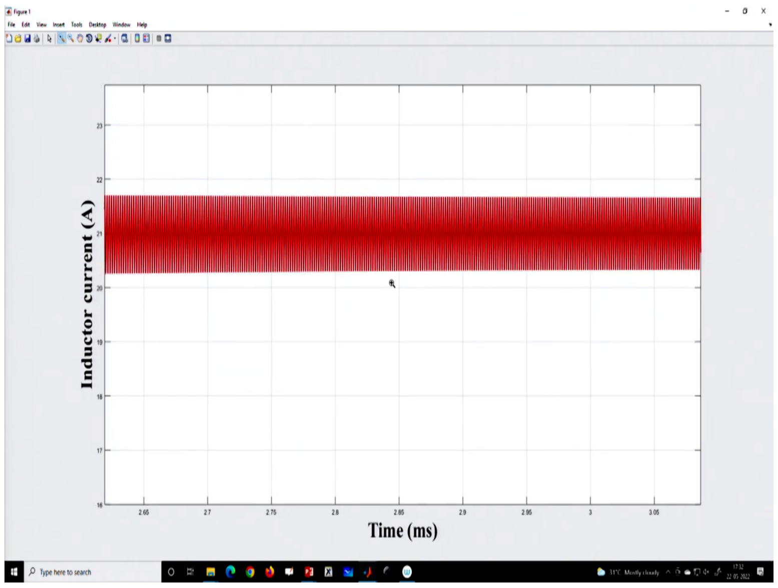
That means since it is a constant on-time control it is a 1-volt output so; that means, it should be inherently stable. So, we are talking about 2-volt input; that means, let us say 1.8-volt input because it is less than it is more than 50 percent duty ratio ok.

But in fact, constant on time means the valley current mode control. So, valley current mode control in fixed frequency is unstable for a duty ratio less than point five. So, it is perfectly stable.

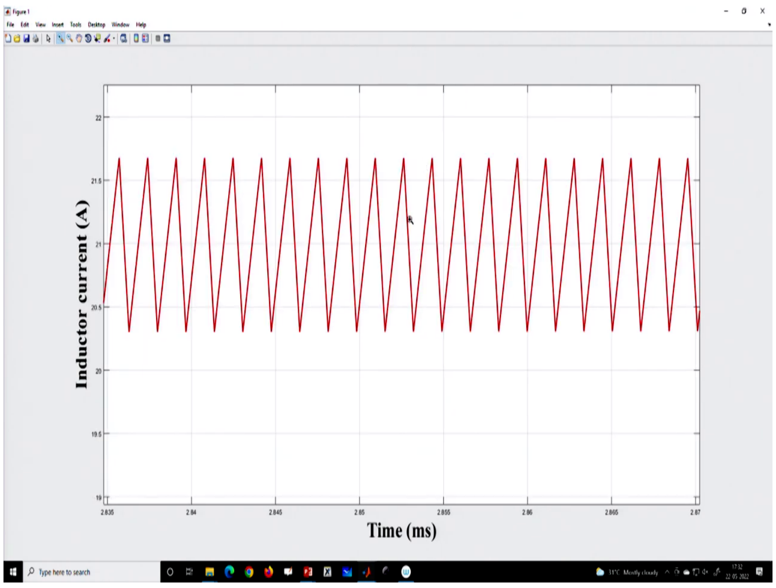
(Refer Slide Time: 15:33)



(Refer Slide Time: 15:39)

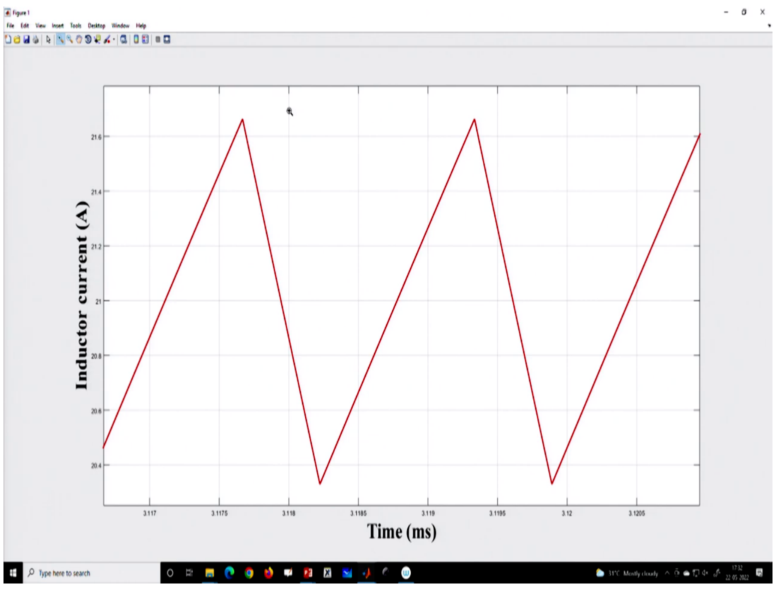


(Refer Slide Time: 15:40)



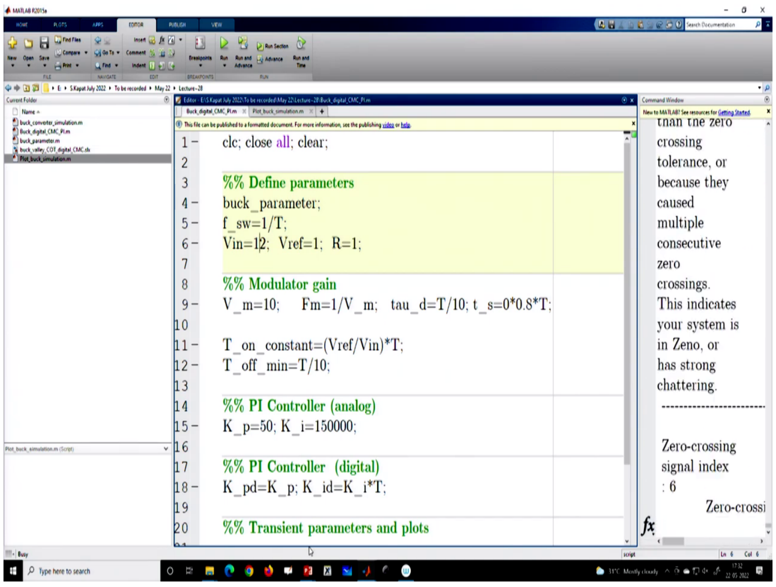
And, you can see even with the high duty ratio also the current loop is perfectly stable. So, there is no problem with the current loop, it is perfectly stable.

(Refer Slide Time: 15:47)



Because that is the beauty of this constant on time off time control there is no problem with the stability of the current loop high duty ratio and earlier we used the low duty ratio.

(Refer Slide Time: 15:52)

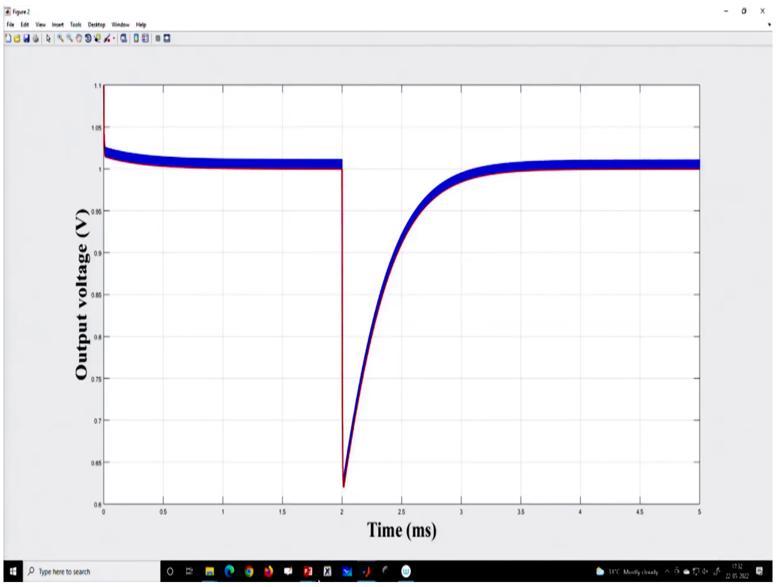


```
clc; close all; clear;

%% 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*0.8*T;

T_on_constant=(Vref/Vin)*T;
T_off_min=T/10;

%% PI Controller (analog)
K_p=50; K_i=150000;

%% PI Controller  (digital)
K_pd=K_p; K_id=K_i*T;

%% Transient parameters and plots
```

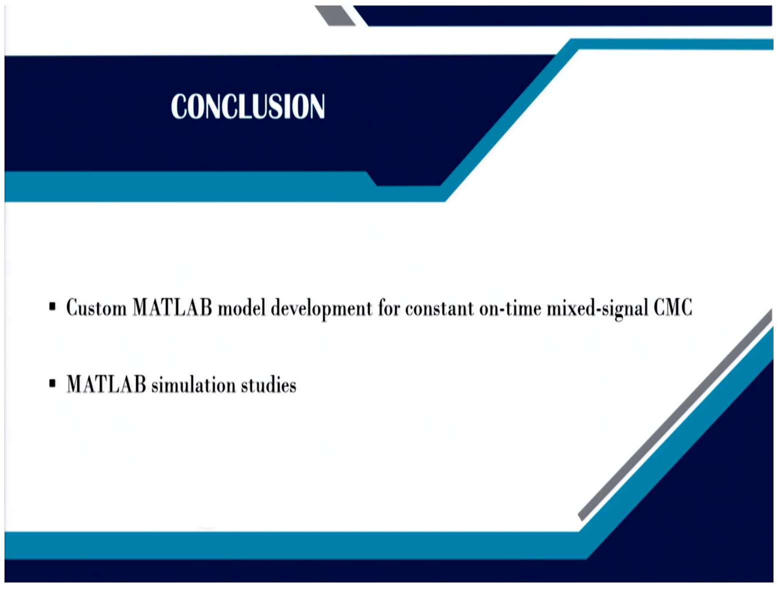
```
than the zero
crossing
tolerance, or
because they
caused
multiple
consecutive
zero
crossings.
This indicates
your system is
in Zeno, or
has strong
chattering.
------------------------

 Zero-crossing
 signal index
 : 6

          Zero-crossi
```

So, if you consider a low duty ratio. So, that means, for why duty range this is perfectly stable there is no problem ok. So, here I am just showing a very high and very low duty ratio. So, you can use it for yeah. So, perfectly stable.

(Refer Slide Time: 16:11)



(Refer Slide Time: 16:15)



CONCLUSION

- Custom MATLAB model development for constant on-time mixed-signal CMC
- MATLAB simulation studies

So, let us go back to our so that means, in summary, we have discussed custom MATLAB model development for constant on-time mixed signal current mode control, and we have also discussed a few simulation case studies. So, we will be considering constant off-time mixed signal current mode control in the next lecture. So, that is it for today.

Thank you very much.