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

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

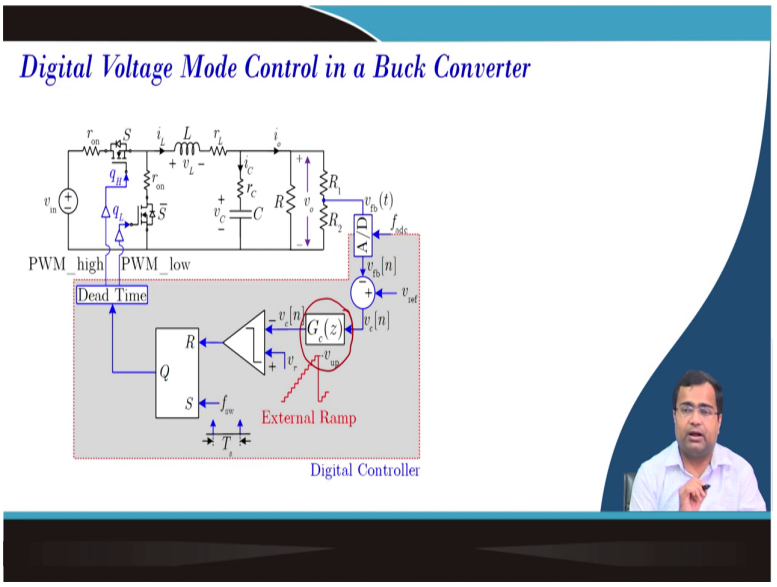
Module - 08
Digital Controller Implementation using Fixed-Point Arithmetic and Verilog HDL
Lecture - 73
Digital PID Control Implementation using Verilog HDL Programming

Welcome to this lecture we are going to talk about Digital PID Control Implementation using Verilog HDL Programming.

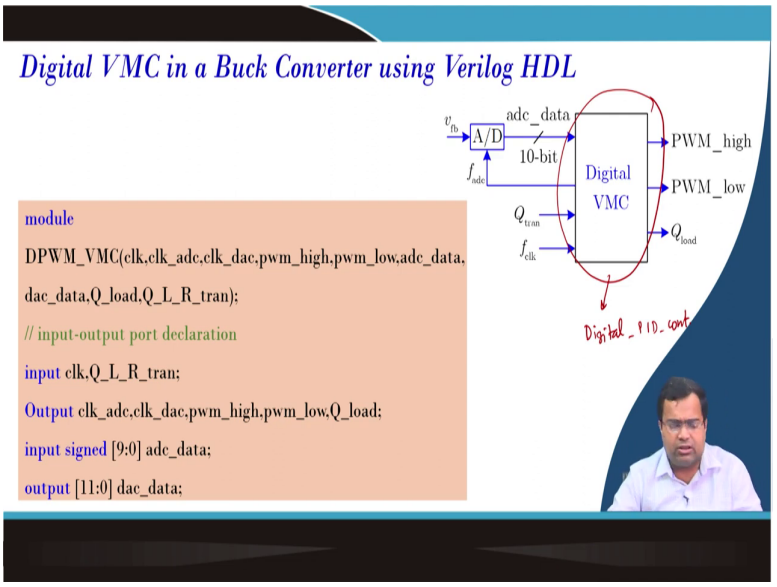
(Refer Slide Time: 00:31)

This is the continuation of the previous lecture where here we want to emphasize especially on the PID controller part. So, we will take PID control implementation and Verilog coding.

(Refer Slide Time: 00:44)



(Refer Slide Time: 00:52)

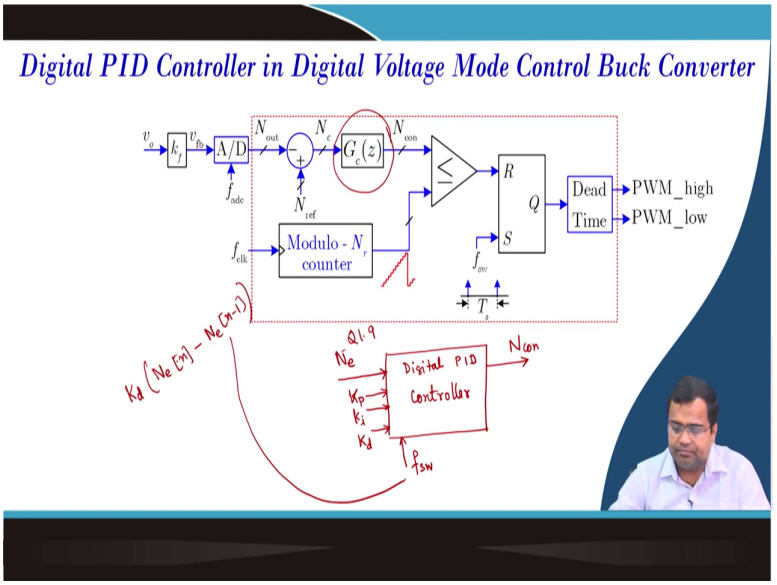


First again we will recall that in our digital voltage mode control where we are considering only this particular part; that means if we take the overall block that we want to design for this digital controller we know there are some modules and we are only emphasizing the digital PID controller. So, this part we are going to highlight.

So, in this part, the main module again is a main module digital you know main module and we just want to highlight our I would say the voltage PID controller part. So, the main

module again is calling DPWM DAC clock all these interfaces we have discussed in the previous lecture.

(Refer Slide Time: 01:31)



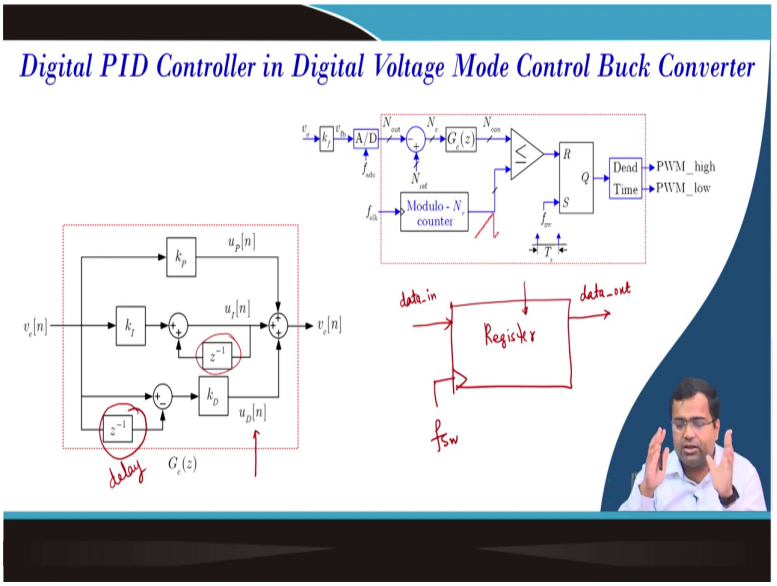
Now, we want to emphasize this particular block, what is the input to this block? So, if we consider this digital PID controller I am talking about; that means, a PID ok. So, it is a digital PID controller what is the input? So, one input is your error voltage which is N e and it is in the format of Q 1 dot 9.

What is the output of this controller? It is the N con that is the controller out and we will decide what is the size of this controller output, but what else do we need? So, we need all the parameters K p we need K i we need K d, and what else do we need? We also need f s w for this block because when you calculate if you remember that you know derivative control, for example, K d we know that N e minus.

So, if you take the n-th cycle minus N e n minus 1; that means, there has to be a delay so; which means, this delay has to be generated concerning the switching frequency clock that is why we need a switching frequency clock for the synchronization also for the integral control we need the additional block like a previous value we have to add is an incremental integral control implementation. So, for that kind of summation, we need to synchronize with the switching frequency clock so, that is one of the inputs.

We can provide other input like a flag; that means, if there is saturation in the PID controller then the flag can be high. So, you can always add an additional thing, but these are mandatory requirements.

(Refer Slide Time: 03:40)

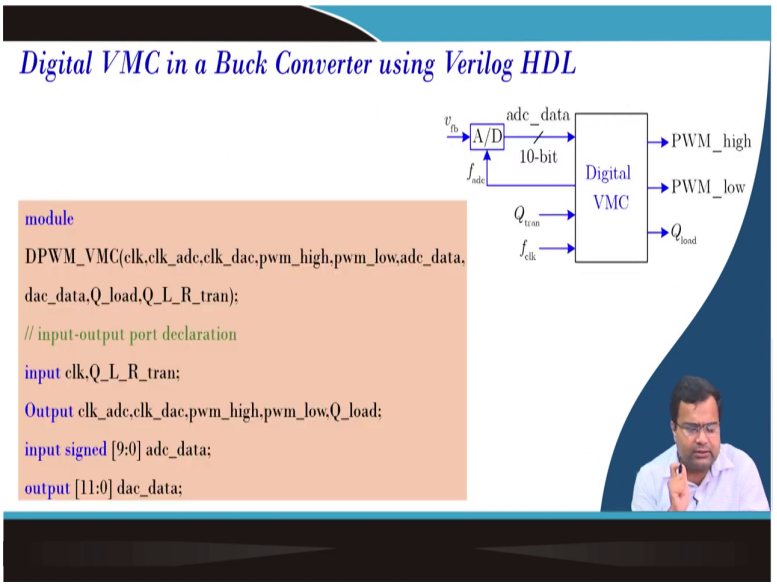


This is the overall PID controller simple implementation we did it in MATLAB, when you say this block the delay right? So, this delay again this delay which should be with respect to the clock we have discussed this delay as a register and this register will be synchronized with the switching frequency clock. And this register will have data in again this is vector data and you will have data out the size of the input and output vector are the same and the data size will define the size of the register which is an 8-bit register, 6-bit register, 10-bit register whatever.

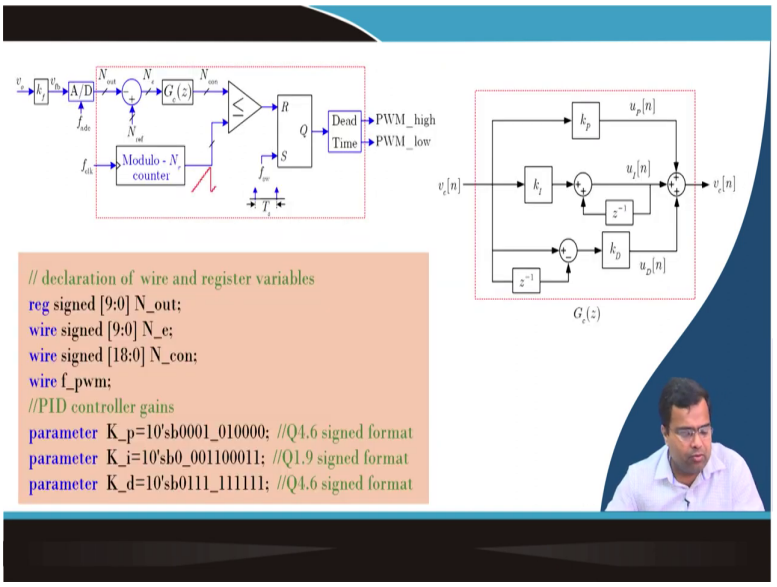
So, the same thing we have we require for this register and we have discussed in MATLAB how to build the custom model using difference equation rather than using plugging in any z transform or any transfer function that also we have discussed customized.

And in the subsequent maybe towards the end, we want to see if we have a realistic MATLAB model and then an actual experiment and then how close they are there, they may not be closed, but how far I mean what is the deviation, how much they can capture the transient behavior. So, with we want to discuss this aspect as well.

(Refer Slide Time: 05:10)

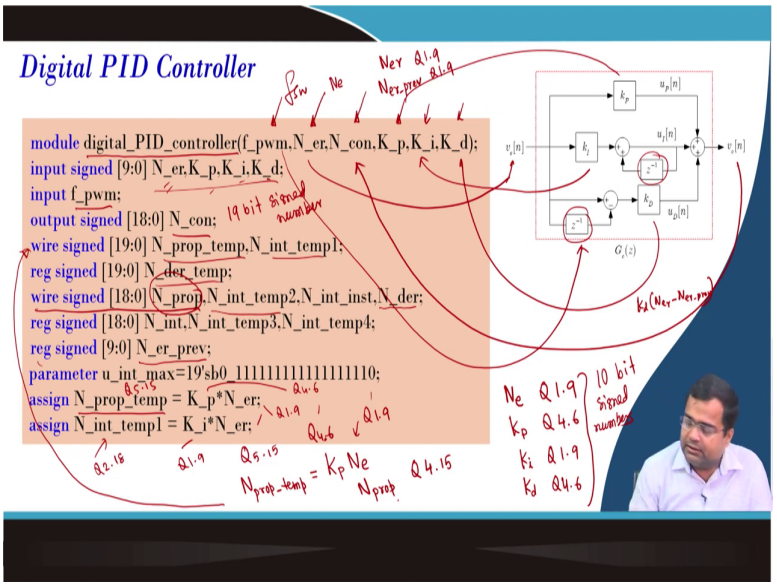


(Refer Slide Time: 05:15)



So, here is the main module again it is the calling that we have discussed, but now where to sign parameters have been defined. So, this up to this point is a main module declaration we require because you need to see the values of the parameter that is the part of the main module the N out and K p K d ok.

(Refer Slide Time: 05:36)



Now, this module was called if you recall in the previous lecture we have instantiated the module name digital PID controller, now we are talking about that particular module. What is the input to the module? We have discussed that input to the module is a switching frequency clock. This is the error signal local error signal we have discussed, this is the controller output and these are the parameters K p K i K d.

And we discussed that these are the bare minimum requirement; that means, we need error input, and in this case, this is the digital representation of this. We need this parameter which is this, this parameter which is this, and this parameter which is this right and we need this output which is this right and we need all these registers to be synchronized with respect to this clock. So, this clock will be used to generate this delay it has to synchronize.

Next, we have to define what is input, and what is the output. We know that the input is a PWM switching clock and there is a scalar, but there is vector input like error voltage K p K i K d, and all we are writing in 10 digit number, but their Q format is different. So, for error voltage, the Q format is Q 1 dot 9, for proportional gain we took Q 4 dot 6, for integral gain we took Q 1 dot 9 and for the derivative gain we are taking 4 dot 6, which means all numbers are 10-bit number 10 bit signed numbers they are represented by the same vector size, but they are Q formats are different.

And we have discussed that if you multiply and add some different formats of numbers we need to synchronize, we need to scale, and normalize them otherwise you cannot add. So,

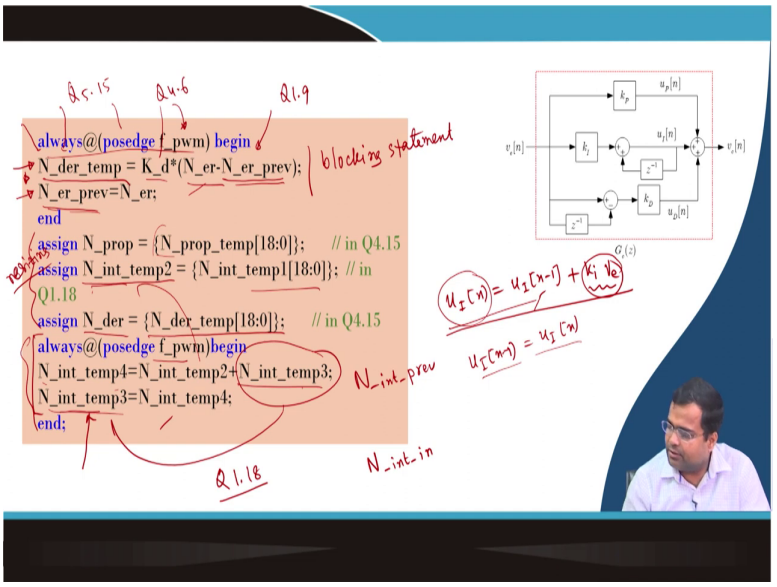
their Q format must match. Now, the output signal is the control output and you can see it is a 19-bit number.

So, it is a 19-bit signed number, but we are here to define what the sign is as a signed number. So, we are internally variable some temporary value ok. So, we can check and see whenever we multiply let's say we multiply K p into N e K p is Q 4 dot 6 then this is Q 1 dot 9 and we call this to be N proportional temporary and that is why you can see their bit size is 19 because their size will be Q 5 dot 15.

But what is the problem, each of these is the sign bit and if you multiply by to sign bit number then resultant there will be 2 sign bits; that means, one you can discuss. So, this will be normalized to N proportional which will be in the Q 4 dot 15 format. So, it is a 19-bit number and that is defined here all this 19; that means, this is the resize data of the proportional term and we will talk about resize rate of the integral and the resize data of the derivative will come to that point.

So, then these are all internal declarations we also need a previous value of error voltage because error voltage we know N e r. So, this is the error; that means, we have N e r which is an error that is Q 1 dot 9 and we also need N e r previous which is also Q 1 dot 9 because for derivative control we know that. So, the derivative will be K d into N e r minus N e r previous. So, we need this precious and that is why we have declared wire sorry register because we will make just a shift operation and that we will discuss.

(Refer Slide Time: 10:17)



Next, we are computing the derivative action because we have to create a previous value and that is happening with respect to the switching frequency clock, all these previous updates will happen concerning the switching frequency clock because the datasdatacaptured output data are captured with respect to the switching frequency clock. So, with that clock we have and we are calculating the error at that clock only and we are calculating the previous error with that clock which is why the always block is there.

So, always derivative term which is K d into N e r minus N previous, what is N previous? After that, here we are using a blocking statement, you can use a non-blocking statement there is no issue, but I just want to avoid so; that means, this line will be executed fast, and then this line so; that means, in the first line it will take the previous value which was stored in the previous cycle.

Once this computation is over then the instruction will come to this and then it will take the previous will be storing the present value at this point of time then this process continues when the next cycle will come N e r will be updated with the new value, previous will be storing the previous N e r value and the action continue and these things we know. So, this is clear.

But now this one since we are multiplying K d to be Q 4 dot 6 and this whole operation is Q 1 dot 9. So, then the resultant here will be Q 5 dot 15 and we are resizing the proportional control. What are residing? We are simply discarding the 19th bit; that means, this is another MSB discarding because 2 sign bit 2 MSBs sign bit. So, we are discarding the MSB.

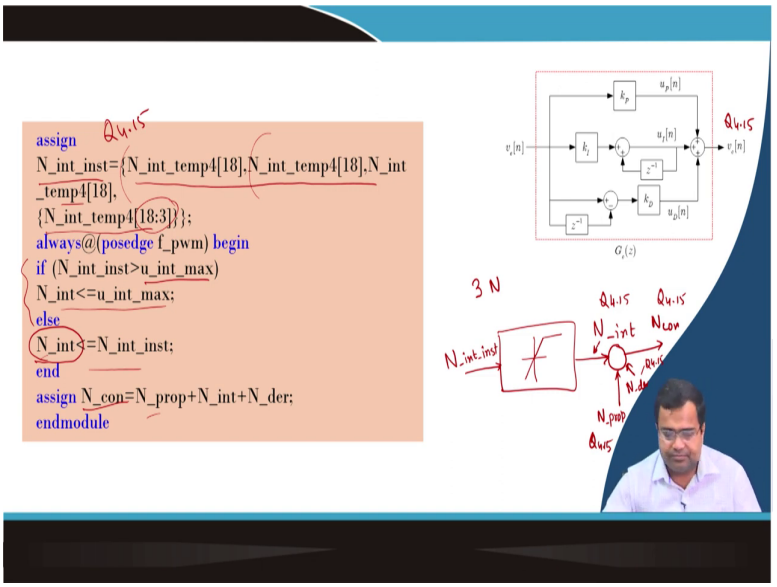
Similarly, we are resizing this bit also and we are also resizing the derivative by discarding the MSB. So, these are all things that are resizing I will say resizing by discarding the redundant sign MSB bit ok. Then here we are doing integral control. So, if we go back to the integral control you will see yeah. So, this proportion we have discussed will be Q 5 dot 15 because this guy is what Q 4 dot 6 and this guy is Q 1 dot 9.

What about this? Q 1 dot 9. So, what will be this? It will be Q 2 dot 18. So, we need to resize; that means, we need to resize because we know that u I of n will be u I of n minus 1 plus K i into v error. So, we are writing in me because here it is N e and that we did in the previous. So, this is the operation K i into N e which N e r is the error voltage and we are resizing this because we need to resize.

Then, now again integral has to be computed because you need a previous value. So, first, a temporary variable and this one is already there and this is another temporary variable. So, you can think of a previous value. So, this will be N integral previous; that means, you can think of this as like N integral previous. So, this is loaded with the current value; that means, whatever we will do to calculate in the same cycle after this line is executed we will take u n minus 1 as equal to u I n, and then the cycle continues.

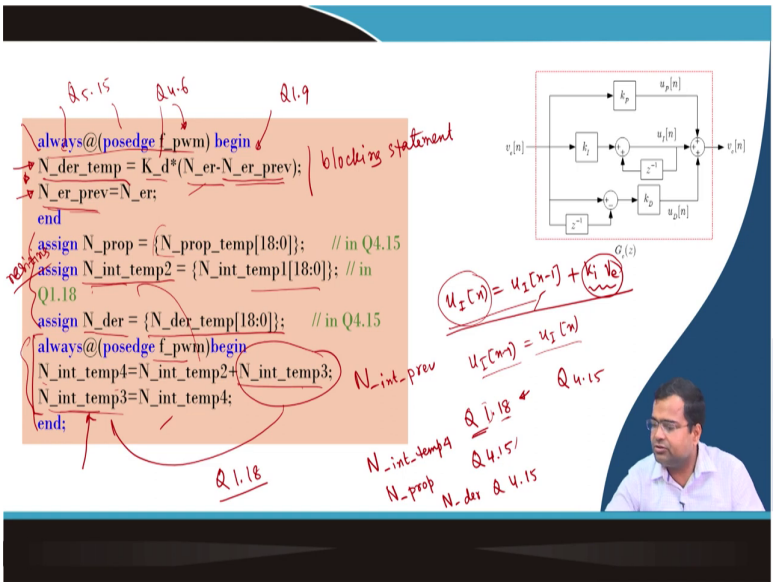
So, these values will be now stored as a previous They will be used in the next cycle operation and this is exactly happening inside this block.

(Refer Slide Time: 14:38)



Then once you have done then the actual instantaneous integral value we are you can see what is the size of this data, we have made Q 1 dot because if we remember that this is 2 dot 18 we have resized it to Q 1 dot 18 this is 2 large data.

(Refer Slide Time: 15:24)



So, we have integrated into the stance you know I would say instantaneous value or this stem ok may be. So, here are all internal variables named 4 this is in the format of Q 1 dot 18 and we have N proportional in the format of Q 4 dot because after resizing 15, we have N derivative which is in the form of Q 1 dot 15. So, we cannot add them.

What do we have to do? First, this data we have to pair with that we can make sign bit extension; that means, we have to extend this sign bit 3 times because we need to make it 4 we need to convert this into 4 dot 15, and then we can discard 3 LSB because it is up to 18. So, you can discard it because otherwise we cannot resize or we can pair it with 0. So, there is no point can discard it. So, either you can discard this or you can resize, if you resize adding another three zeros size will go up. So, it is a designer's choice.

So, you are padding with 3 sign bits. It could have been 3 you know N this sign bit, but for some reason actually, the Verilog was giving an error. So, I have manually made 3 sign bit a concatenation operation and here I have discarded 3 LSB so, that this number now becomes Q 4 dot 15.

Now, we are making it a saturation limit there is a high chance the integral gain will saturate because it is a sign number. So, if this number exceeds some upper limit which again is defined as the upper limit is defined as some number. So, I can set some number it is your choice, I am not going to the full upper limit of the dynamic range leaving one space down

that is the upper limit if the integral gain reaches that then we will simply set the maximum value otherwise it will take the current value.

You can also set the lower limit as well no problem once this is the actual integral gain coming from the saturation block; that means, we are putting an upper saturation limit lower as if we are not putting, but we can put it and this is my N integral before that it was N integral instantiation flow.

Once it is generated now we are adding it. So, this is this particular value N proportional, we are adding N derivative and this is my N con, since this number is 4 dot 15 this number all are 4 dot 15, 4 dot 15, I am also taking 4 dot 15, but there is a possibility that there can be overflow. So, that can be taken care of, but otherwise, it is a 4 dot 15, it is a 19-bit number which is why what is going out is 4 dot 15.

So, here you know again you can take another. So, it makes a little bit complex, but it is unlikely that all will add up and exceed, but there may be possible if your gain is high, otherwise, they will come under this 4 dot 15.

(Refer Slide Time: 19:00)

So, in summary, we have discussed digital PID control implementation and we have also discussed Verilog HDL coding for digital PID control implementation. And we will discuss the light demonstration of this PID controller that transient response in the subsequent lecture and that is it for today.

Thank you very much.