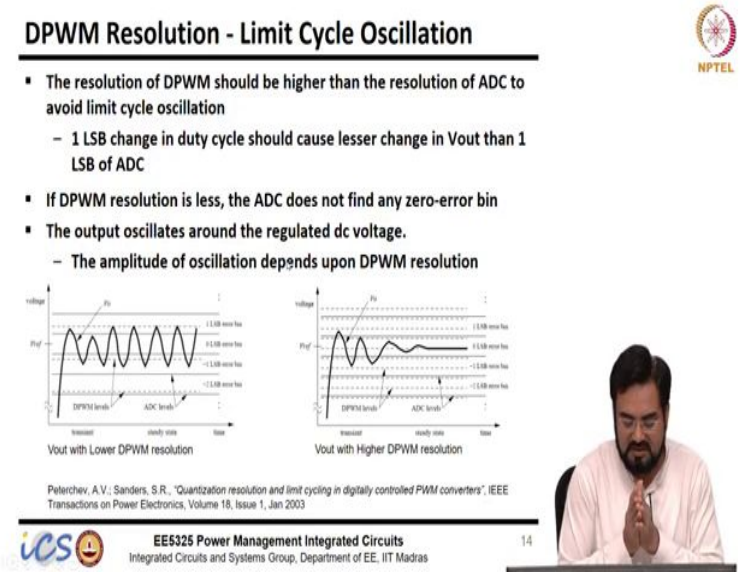
Power Management Integrated Circuits
Dr. Qadeer Ahmad Khan
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 93
Digital Pulse-Width Modulator Architectures, Adaptive Compensation

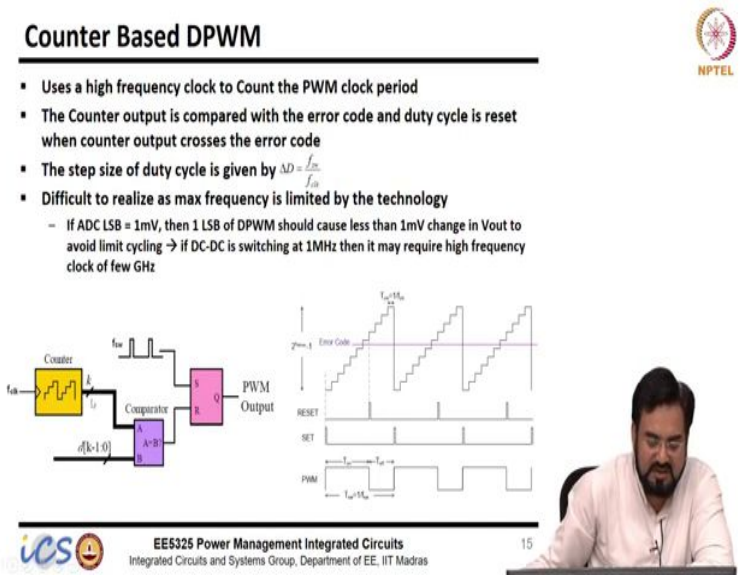


Then the second thing we require is DPWM which is digital PWM. So, after the compensator, it is digital PID then output control voltage and it is fed to DPWM and we have to convert it to PWM. So, the resolution of DPWM should be higher than the resolution of ADC to avoid limit cycle oscillation. We talked about the limit cycle in digital LDOs. The resolution of the current DAC or resistive DAC we are using to regulate output should be more than ADC and DPWM is nothing but a DAC.

It's converting digital code into PWM which is time. So, if DPWM resolution is less than ADC, then we will get limit cycle oscillations. The output will never settle and keep bouncing between ± 1 LSB and we will get the kind of behavior as shown in the left plot in the above image.

But if the resolution of DPWM is higher. When we say the resolution of DPWM is higher, delta step in voltage due to change in LSB of ADC should be higher than the change in voltage when we change the LSB of DPWM. In ADC, we know that we do not require it for a full range of voltage and design it with less number of bits. If we have to cater to a wide

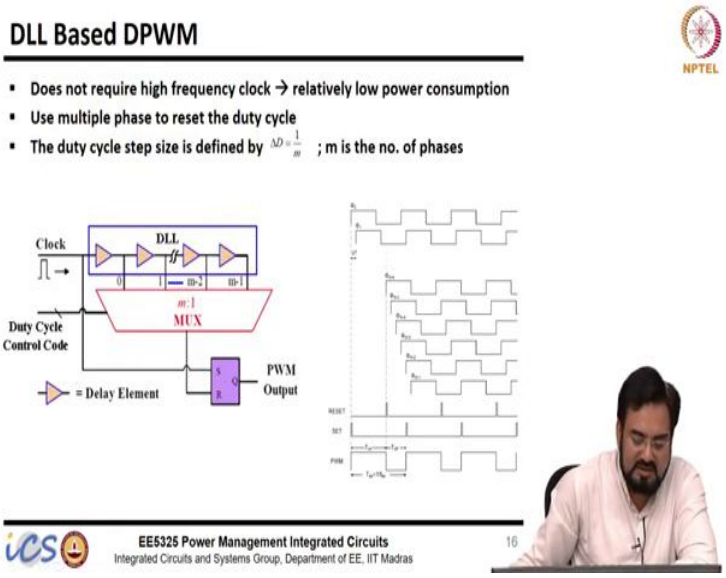
range of V_0/V_{in} ratio then the duty cycle may vary from 10 percent to 90 percent. We are looking at 80 percent range here, then the number of bits which we will require in the DPWM would be much higher compared to ADCs even if the step size remains the same.



The brute force way of implementing DPWM is counter based DPWM. So, let us say we are operating a 1MHz clock and we need 10-bit resolution which can give us a 0 to 100 percent duty cycle. So, 10 bit is 1024.

We can count with the high frequency clock. Let us say we have a 1GHz clock. So, 1GHz clock will count 1000 in for 1 megaHertz 1 cycle and for 10 percent it will count 100, for 1 percent it will count 10. We can get a high resolution DPWM by just feeding a high frequency clock and counting up to the duty cycle we want. The circuit diagram and waveforms are shown in the above image.

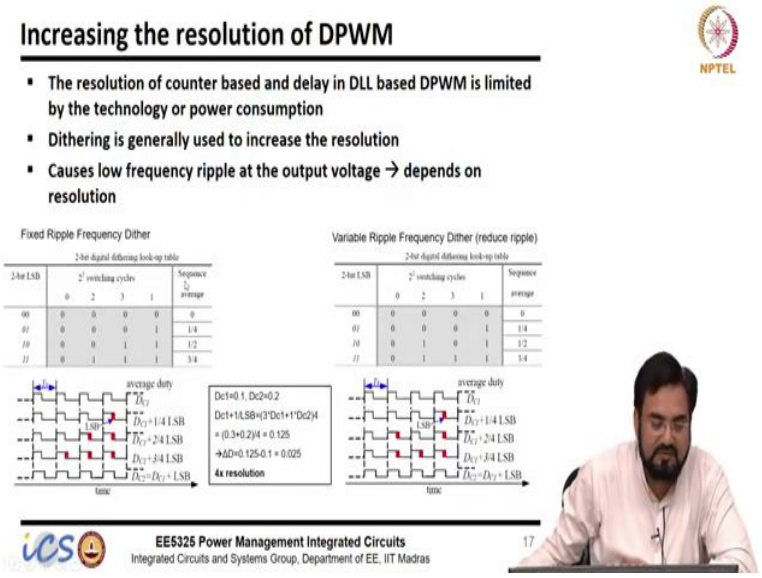
The control code which we are generating from PID will control the count. So, how many counts we require based on that, it will reset the duty cycle and we can vary the duty cycle. But the problem here is that the oscillator for a 1GHz clock is not simple to design and it is going to take a lot of power.



So, what we do? We use DLL based. So, in the DLL based, you can still feed the same 1 megaHertz clock and have let us say 10-bit resolution and can have 1000 taps.

You can think about you have a 1 megaHertz clock i.e. 1 microsecond delay which is 1000 nanoseconds and you have 1000 delay cells each giving 1 nanosecond delay. You have 1000 phases spaced by 1 nanosecond and depending on the code at which phase you want to reset the DPWM you will get similar behavior.

The code will decide which phase to pass to reset and set is always appearing at the positive edge of your 1 megahertz clock. This is how it will be looking like each phase will be delayed based on your resolution. But then again you require 1000 delay cells here and you cannot match each delay cell. So, 1 nanosecond one might be like a half a nanosecond or something. So, matching is a problem. You may get random delays actually here instead of equally spaced these delay cells.



So, one way could be to limit the resolution. So, instead of 1000, I will use only 100, the same thing you can do here also. So, instead of 1 gigaHertz, let us say we use 100 megaHertz. So, your resolution now is 1 percent only.

But 1 percent is basically a large step, it may cause a transient in the output when you change the duty cycle. Let us say you go from 50 to 51 percent duty cycle, you will see a large transient in the output which may be unacceptable. So, then what do you do actually, you dither and dither we already talked about, if you remember the pulse skip mode and even in the buck boost mode we are talking with these number of buck and these number of boost or these number of 100 percent duty cycle and 90 percent duty cycle the D_{max} you are dithering between the same thing you can do here also. If you remember sub-harmonic oscillation in the current mode, what happens? It basically alternately changes the duty cycle, but on an average.

So, let us say 25, 75, and on an average 50. So, let us say I have a 1 percent resolution here. So, between 10 percent and 11 percent, I can dither. So, one cycle of 10 percent, another cycle of 11 percent will give me 10.5 on an average, if you want more resolution then 1 cycle of this, 2 cycles of the other. So, 1 is to n or n is to 1 kind of code you choose and you can keep dithering.

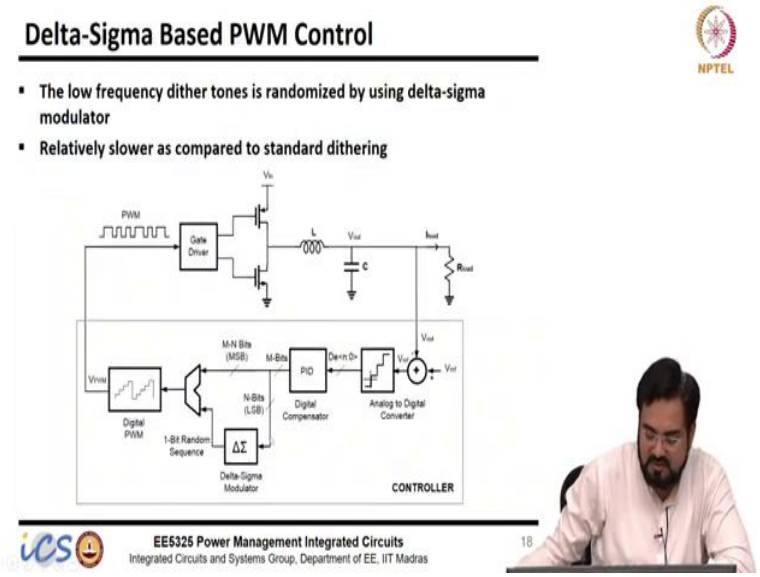
So, dithering could be like two types; one is the fixed ripple frequency dither other is variable ripple frequency. So, when I say fix ripple, so let us say I have a 1 cycle of 10 percent and 1

cycle of 11 percent. So, I am getting 10.5 percent, but now 1 cycle of 10 percent and 2 cycles of 11 percent. So, what are you getting here?

Slightly more than 10.5. So, now, 11 percent is 2 cycles and 10 percent is only 1 cycle, but your dithering frequency has changed now. So, we will have two tones here; one tone will be coming at PWM frequency and the other tone will be coming at dithering frequency. When I say dithering frequency, this is 1 duty cycle and these 3 cycles let us say different duty cycles.

So, on average, you will get each cycle may have the same frequency, but if you look at another tone this is your dithering frequency which means 1 cycle of this and 3 cycles of this. So, it will be looking one-fourth of your PWM correct. So, because the total is 4 cycles of this, but the same thing I can achieve by doing 1 is to 1. So, what if I do let us say 2 is to 2. If I do 2 is to 2 and 1 is to 3 the number of cycles will remain the same. So, your dithering frequency remains fixed, but depending upon what is the level of dither, that is the frequency you will require you may require 10 is to 10 actually or 20 is to 20. So, that dithering frequency will be much lower. So, your output ripple will be large at that particular frequency which is coming due to dithering. So, that is why you always prefer this variable frequency, where if I require let us say the same number of cycles instead of choosing let us say 10 is to 10, I will always choose 1 is to 1.

So, I will keep 1 fixed to 1 other increasing and in the other case you can choose 1 is to N or N is to 1. So, that is the better way, so that 1 is always 1 cycle other is changing instead of using the same number of cycles in both, then you will have a low-frequency tone here and your ripple will be large.



So, another option could be delta-sigma based, this is nothing, but more like dither. So, how do you dither it?

Let us say, I have an M bit code. So, the N bit is going to the delta-sigma. So, let us say I have a 10 bit. So, 3 bit I can feed to my DPWM and 7 bit I can feed to delta sigma. So, what does delta sigma do? It will automatically generate a code that will dither between this and so 3 bits mean you have 8 levels ok. So, from 8 levels you have to generate let us say 1000 different values.

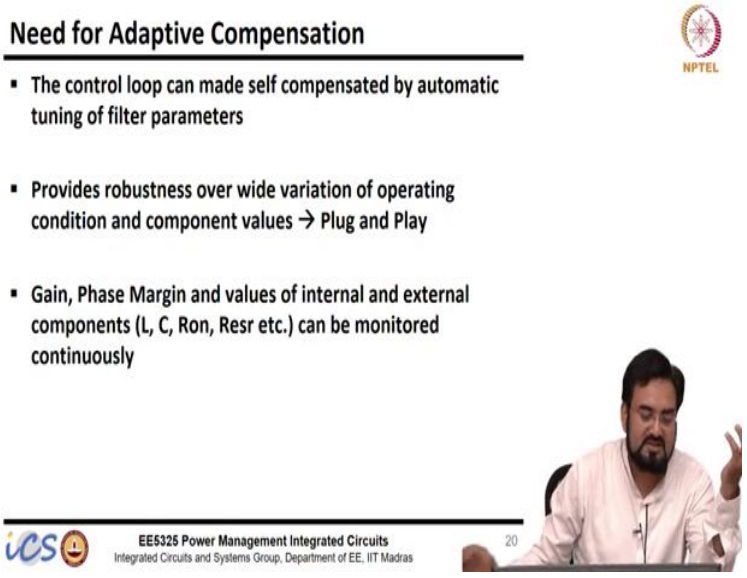
So, what that delta sigma will do from that 7 bit? It will choose which one to use and how many cycles of that. So, it is doing nothing but selecting between which will be 1 and which will be N that is it. And those N could be values out of this 7 bit. 7 bit is 128, basically 0 to 127 you will get.

So, from each level now, you can think that this is doing a course DPWM and the fine control is coming from delta sigma. So, I have sliced let us say in 8 levels, my DPWM, I mean, 8 levels is actually very low, but you can think about the same case where we were talking about 100 megaHertz clock or 7 bit. So, 7 bit is also usually on the higher side.

So, let us say I have a 4 bit; 4 bit is 16 code total. So, let us say I have 16 levels I have sliced into 16 levels and those 16 levels between the 2 adjacent levels, I can create one 7 bit

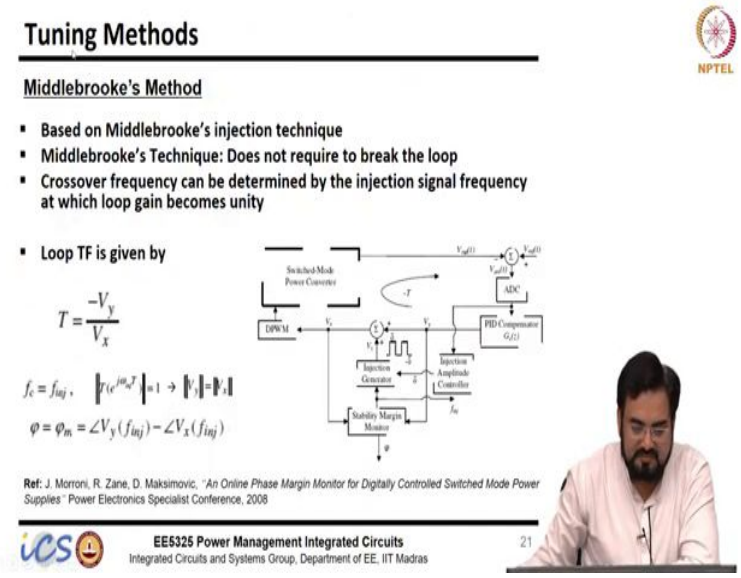
combination out of that correct and that will be for each level. So, you will be getting a 127 multiplied by 16, so that high resolution you can get.

And if you want a total of 10 bit then 4 bit versus 6 bit you can do. So, you can do a 6 bit in the delta sigma. So, it is up to you like you can split into 5 is to 5, 5 bit here and 5 bit there or 3 bit here and 7 bit here.



Adaptive compensation. So, we talked about the advantages of your digital control because it can adapt to any changes in the L and C or any other parameters which may change dynamically or due to process or even voltage or temperature. So, another advantage of this adaptive compensation is plug and play actually. Plug and play mean you have given a converter to a customer without bothering about what L and C are going to put.

So, you can choose any L and C and your control loop will adapt to that L and C and place your zeros and poles, and again everything will be set inside. So, which means you need to know the value of L and C for that. I mean, in order to compensate what all we needed? We mostly needed the L and C. Once you know your resonance after that you need to look at what gain you have to set based on uncompensated loop gain and based on that you decide what gain you need to choose, k_i or $k_{p,}$ and to get UGB at your required desired value.



When I say tuning methods, it is basically the adaptive compensation and you tune your parameters based on your changes in L and C or any other parameters. There are multiple tuning methods.

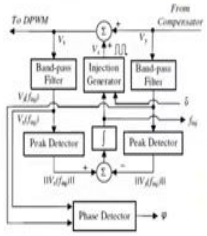
In the Middlebrooke method, we look at the loop gain without breaking the loop. So, one way is to look at the phase response and when we open the loop, we look at a ugb and what is the phase margin at that, but if you look in the closed loop what will happen? So, that will be unity gain correct. And then whatever phase you get there it will tell you the phase margin.

So, you tune the frequency and at what frequency it becomes a unity gain you look at the phase delay in the two and you can get both phase margin and your UGB. So, that is what you do in the closed loop and you can basically and once you have your phase margin and you know the UGB.

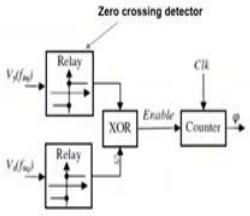
Ultimately what I want is to measure L and C. The reason I want to measure L and C, I want to fix my phase margin and UGB, but if you can directly measure the phase margin and UGB, then I do not need to know L and C. So, that is what it does actually, but the problem here is you need to inject a signal. So, on the fly you cannot do, your output may get disturbed. So, the signal which is injected here will appear in the output. So, it may cause a disturbance in the output.

Middlebrooke's Method (Contd.)

- The injected signal is variable frequency square wave therefore requires narrow BPF to get rid of higher harmonics
- Peak Detector is used to determine the amplitude of Vx and Vy and compared
- Phase Margin is determined as the phase difference between two signal



Detection of crossover frequency

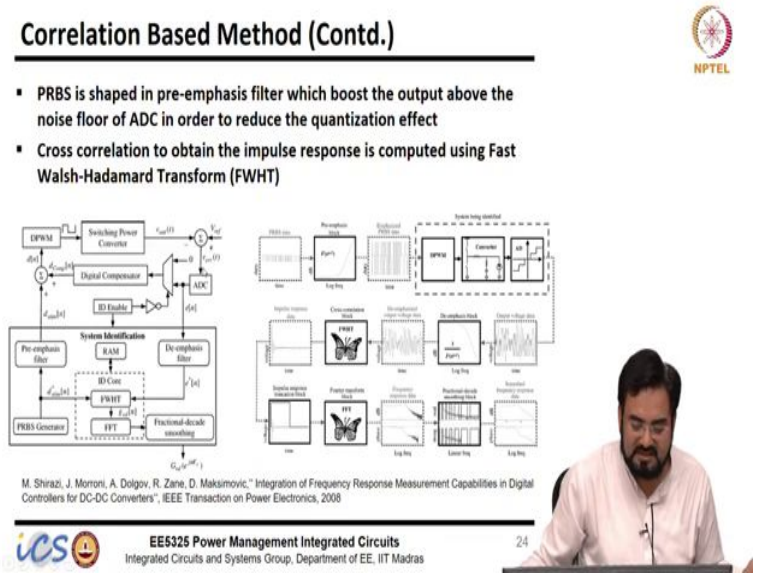


Detection of Phase Margin

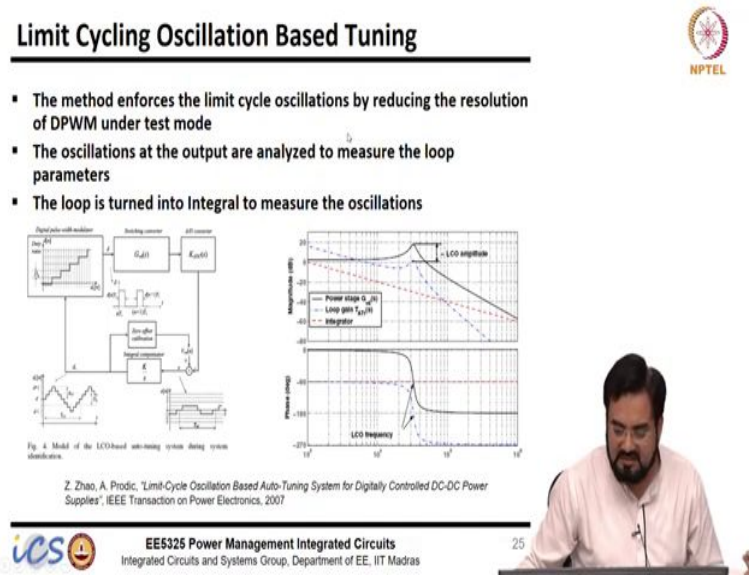
This is not a very clean method, another method is again the same thing, but done in a whole different way. So, yeah these two are actually the same, this is just showing you how you inject the signal and then you need to have a band pass filter, it is very complicated because you need to inject a sinusoidal wave at one particular frequency only. If you have multiple frequencies then it is very hard to know where UGB is because UGB will appear only at one frequency. So, you need to have a very narrow band signal. So, that is why you need to use a band pass filter here.

Correlation Based Method

- Uses PRBS generator to determine the impulse response and transfer function is determined by DFT
- PRBS is shaped in pre-emphasis filter which boost the output above the noise floor of ADC in order to reduce the quantization effect
- Cross correlation to obtain the impulse response is computed using Fast Walsh-Hadamard Transform (FWHT)

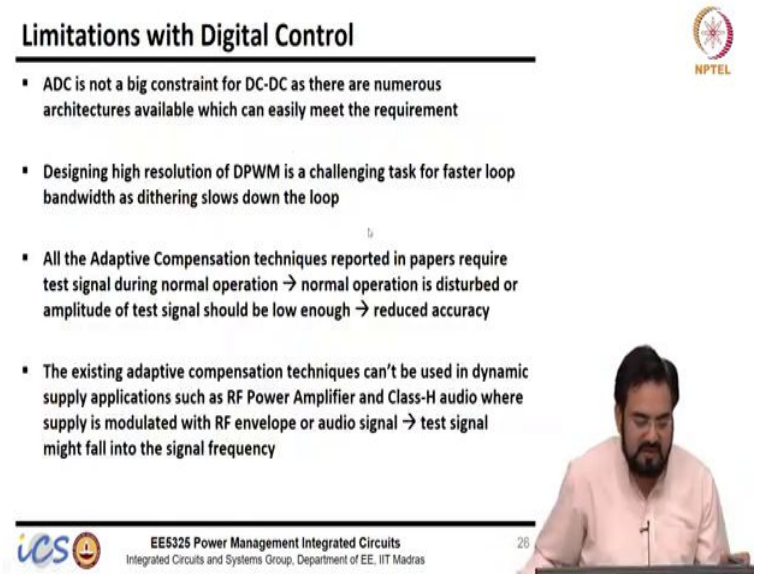


So, another method is correlation based, and this is very complicated. It requires a whole DSP to implement this. So, they are looking at the FFT here, which means the frequency response you are looking at. So, it has its own like all digital here and in this paper, they have implemented on FPGA. So, why would I use a whole FPGA to build a DC DC converter, it does not make any sense.

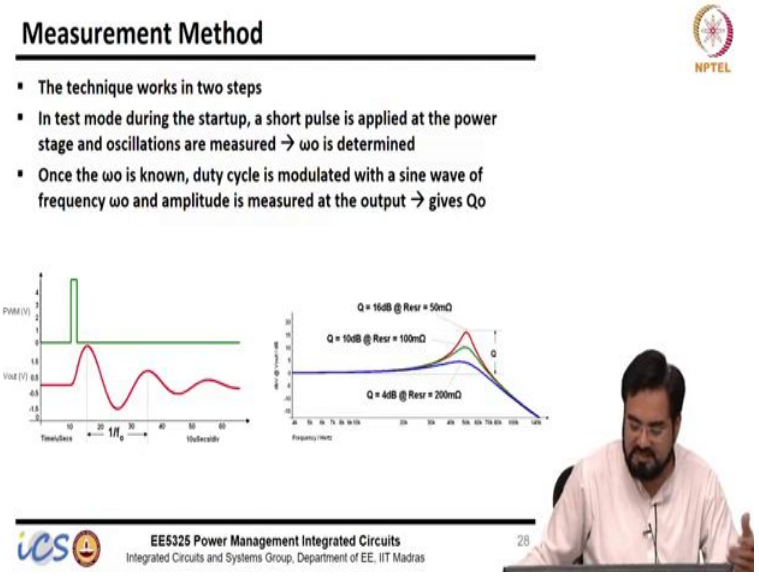


So, this is another method which is limit cycling oscillation based. So, what they do actually if you look at this uncompensated response. This is uncompensated and they literally force it in unstable, but they want to make sure that when it is unstable the amplitude is not that large. So, they force into a limit cycle which means you keep on increasing your UGB or gain. Let

us say I am tuning the gain. So, whenever it crosses ω_0 it will start oscillating at ω_0 frequency. So, if you have enough gain here at ω_0 then it will always oscillate at ω_0 and that frequency will tell you where the ω_0 is or the resonance frequency. So, once I know the resonance frequency I can tune, but again the problem is I cannot do it dynamically because it will again cause a disturbance in the output.



These are the limitations of the existing techniques.



So, simplified tuning techniques could be like I can give a step signal or impulse at the input, and in the open-loop let us say, I give a step L and C, what should I expect? The ringing and

the ringing frequency will tell me ω_0. So, this is the simplest way we can implement it. So, can I measure the Q also here if let us say I want to measure the Q.

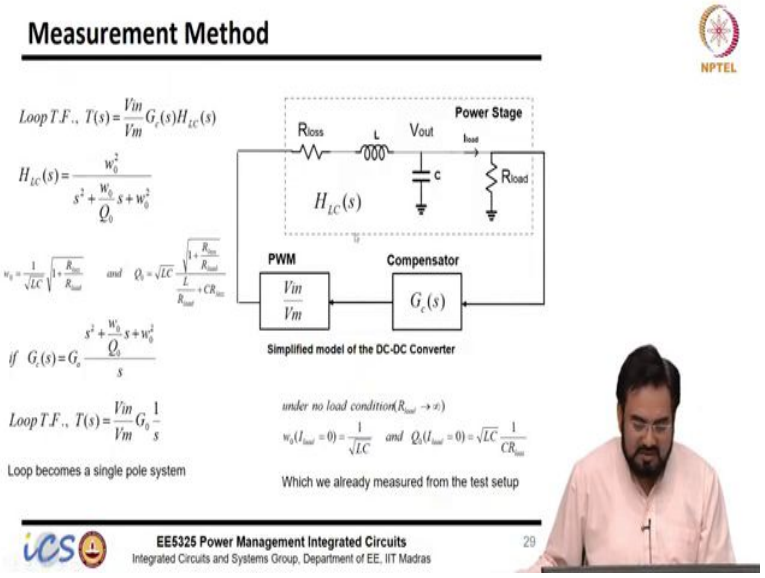
So far what we have seen we can say that ADC is not a big constraint for DC DC converter especially, if you are looking for a smaller range. So, you do not need a 10-bit ADC with a very high resolution. I mean you can achieve that same resolution by limiting the range of the ADC because you are only basically catering to the change in the output during transient plus minus 50 to 100 millivolt. One thing you have to consider is that reference is fixed. If you are changing the reference then the common mode is changing. So, then you may require a wide range of ADC. So, if your feedback voltage remains fixed for all the output voltages if you are changing your output voltage only with the feedback resistor. Then you can limit the range of your ADC, but if you are changing the output voltage with the reference then the ADC input voltage is changing then you may require a very wide range.

So, designing high resolution DPWM is a challenging task for faster loop bandwidth as dithering slows down the loop. So, we talked about this if you want to increase the resolution you dither between two duty cycles. So, which means, you are slowing down the loop. If let us say my dither level is much higher and you are doing a 1 is to n like some 1 is to 10. So, you have to wait for that many cycles to settle your output. You cannot change the code before that. So, you are basically slowing down the output or the loop. So, because output will take a long time to settle.

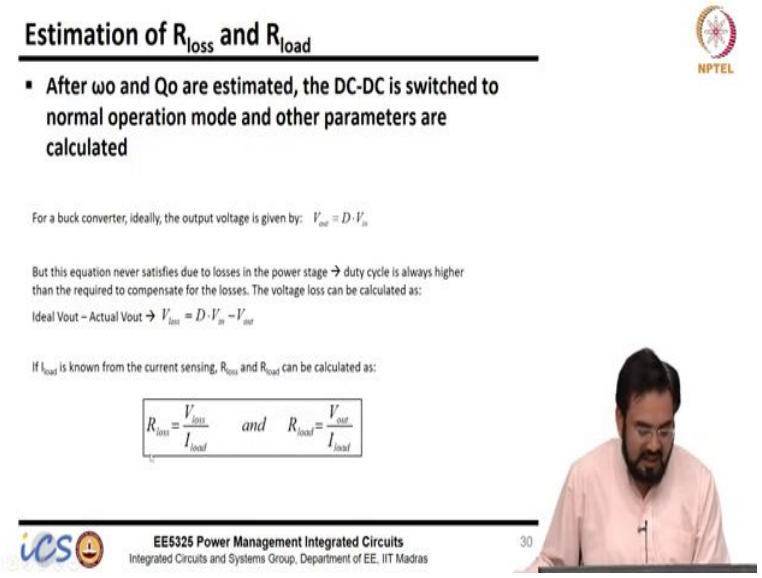
As you keep increasing the order of your dithering, you have to limit your bandwidth. You do not have any option and if you use a very high order dithering with the faster loop then it may become unstable. And also we saw that there are different techniques for adaptive compensation, but most of them look very complicated or they cannot be used in normal operation because you will get disturbance in the output.

The existing adaptive compensation techniques cannot be used in dynamic supply applications such as RF power amplifiers and class H audio where supply is modulated with RF envelope or audio signal because test signal might fall into the signal frequency range. So, you have a test signal, I mean most of them were using test signals. So, obviously, it should be more than the output ripple, if it is less than the output ripple then you cannot detect it.

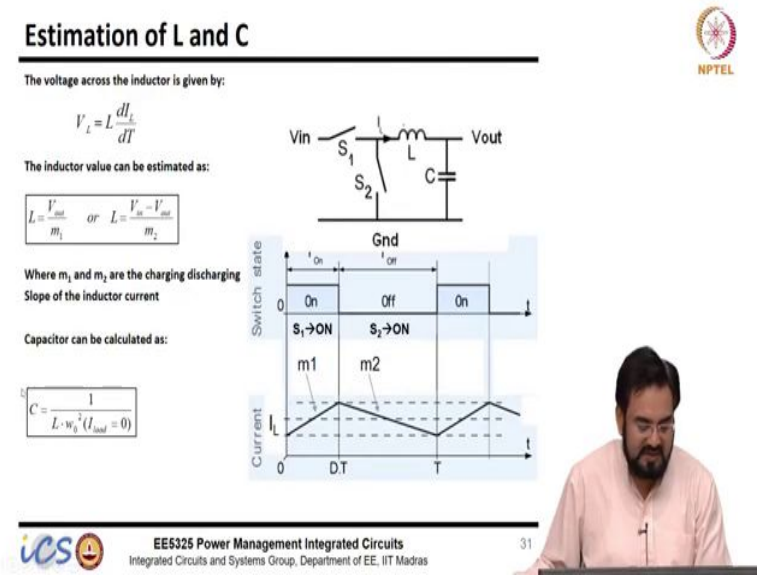
This means you are introducing more noise in the output and if it falls within your signal range then it will corrupt everything, so you cannot use it. So, then we looked into simplified techniques where we are only using a step or impulse response and this can be achieved by using when you are starting your converter during that time you can do that before the output gets settled and you just look at the ringing frequency of the output that will give you the ω_0. Usually, Q may not be required, but let us say in case you want Q also you without measuring the amplitude of the output ringing you can simply look at the number of cycles it rings for and you can very well relate that to Q it may not be that accurate, but even if you are let us say within 20 25 percent, that is fine.



Then we know that H_{LC} is this. So, ω_0 can be related to this and Q_0 can be related to this, and then under no load condition that is what we look for in the worst case Q_0. So, most of the compensation we do for no load. So, this will become a square root L C over C R_{loss} and ω_0 is 1 over square root LC under no load condition.



R_{loss} can also be calculated, we know that V_{loss} is D times V_{in} minus V_{out}. So, if we can measure the average of your switching node and take the difference between V_{out}. That will be the loss in your inductor and if you look before the gate driver the PWM, then that will include the power fet loss also. So, you can separate out even inductor and power fet losses. So, that is not difficult to do and if you know I_{load} you can measure R_{load} also.



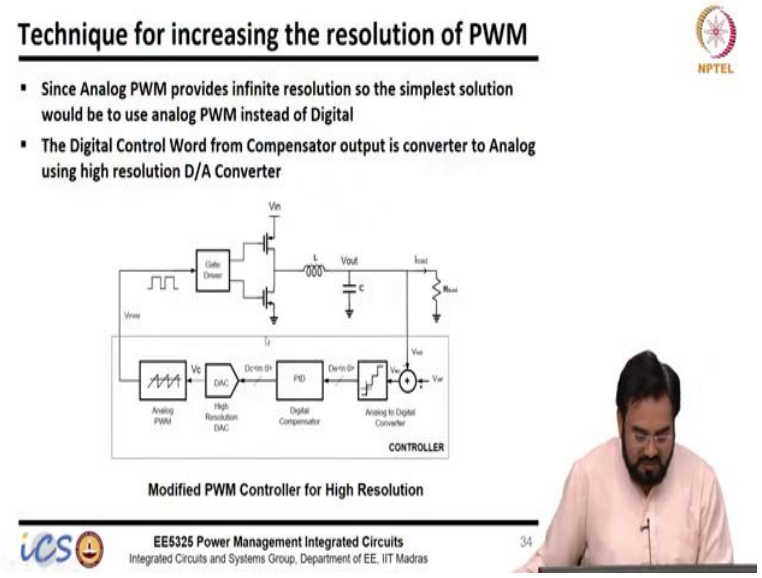
So, all the measurements can be done very easily. Let us say I have a current sensor and I can sample this ramp, then you can calculate your L also. I mean, we have ω_0 information, but if I want to calculate let us say L and C_0 separately then you can do that also.

I can calculate L by simply measuring the slope of this and that you can do a data sample here and another data sample here and take the delta and from there you know time and you can very well get the inductor value from here. L equal to V_{in} minus V_{out} over m_2 where m_2 is the slope of this falling or rising slope whatever you want to use you can use either of them. When you are using a current sense then depending upon what duty cycle you are operating your bandwidth requirement is very high.

If you are operating at a very low duty cycle then rising slope may be faster compared to the falling slope. If you are only measuring the inductor, then you should always look for which has a higher time which means, here you are operating a low duty cycle. So, you will get more time during t_{off}. So, you can measure this slope because both of them will give you the same information and we are interested in only L.

And once you have that you can calculate your output capacitor. So, one thing you have to remember this requires some maths and that is why everything is done digitally. So, you require ADC once you get the output from ADC you can do division, multiplication, etc. and get all the values in digital. Once you have those values you know how to tune your compensation parameters, ω_{z1}, ω_{z2}, and gain. So, you need some algorithm to run in the background which will keep doing this and you can do this in the background in the normal operation also. So, during a startup, you can measure ω_0.

So, place your ω_{z1} omega_z2 and you have started and you have made sure that now my converter is stable, but you cannot ensure that you have optimum bandwidth let us say one tenth and I want to achieve always one tenth bandwidth or the best bandwidth and phase margin across all the condition. So, any dynamic changes, let us say once your converter is running I mean due to your change in input and output voltages or due to change in any other parameters due to temperature, location of ω_0 may change or your gain may change. If your V_{in} is changing or V_{out} is changing. So, then you have to adjust your parameter accordingly. So, those things can be done on the fly in the background this will be running because you are already measuring your L. So, if there is any change in the L that can be taken care of.



As we know that designing a digital PWM is not easy especially if you are running at a very high frequency. Let us say I am running at 10MHz. So, it is not very simple, if I want ten bit kind of accuracy. An alternate way could be like you just use the analog ramp and you convert your digital code into an analog voltage by using high resolution DAC and doing a high resolution voltage DAC is easier compared to doing a DPWM.