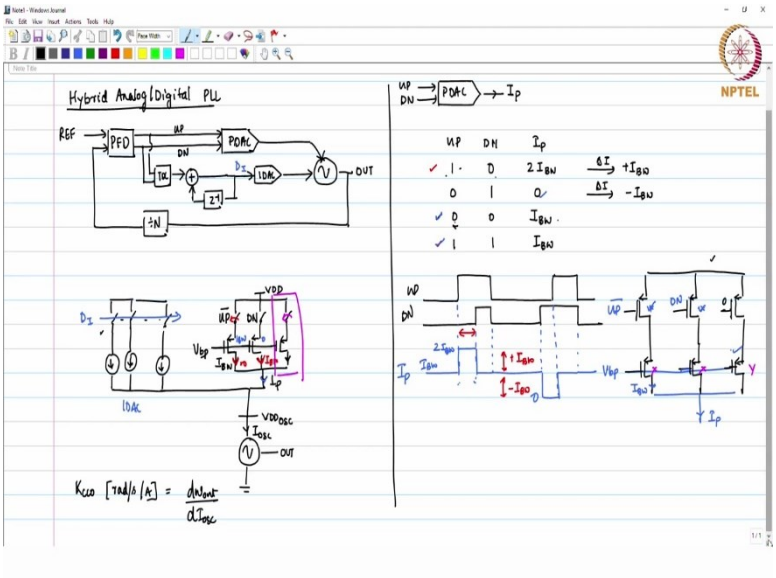
Phase-Locked Loops
Dr. Saurabh Saxena
Department of Electrical Engineering
Indian Institute of Technology Madras

Lecture − 64
Analog/Digital Hybrid PLL: Part II

Hello everyone. Welcome to this session. In the previous session, we looked at the digital PLL, the block diagram of the digital PLL and later we went to the hybrid PLL to undo the quantization noise of the TDC in the proportional path. So, let us begin with that and see how we are going to implement it.

(Refer Slide Time: 0:39)



So, this is hybrid PLL, analog/digital, you want to write it, you can write it like this analog/digital PLL. So, here, we had PFD, the implementation of the PFD we have seen and then you had this feedback path from here. The PFD output goes to a DAC which I call as proportional path DAC or PDAC. The output of the PDAC goes to the oscillator and this I have been using a current controlled oscillator, for example.

The output of PFD also goes to a TDC. Quite often this is a 1-bit TDC. The output of the TDC goes to a digital loop filter. You can have gain if you want and then this is $1 - z^{-1}$ implementation. The output of the digital loop filter goes to a DAC which is IDAC, current DAC in the integral path and this is what you have. Then the output of the VCO feeds back through a frequency divider.

So, here, what we have seen earlier is the implementation of the phase frequency detector, the TDC using delay elements or using a single D flip-flop, your current controlled oscillator also. So, this is your OUT. So, just think about it how we are going to control the current controlled ring oscillator using our proportional DAC such that there is no quantization noise. So, the oscillator which you have while discussing the oscillator here, the ring oscillator in our case.

Now, let us call this as VDD_{osc} and you have current flowing through the oscillator. It will have a certain frequency whether you control I_{osc} or you control VDD_{osc}, if either of these two parameters, control parameters I_{osc} or VDD_{osc} is same, the frequency will remain same in both the cases whether the control is VDD or current, the other will just follow. So, this particular I_{osc} can be controlled using a current DAC and that is what effectively your IDAC is doing.

So, you can have a current DAC let me just make it here with switches. So, in the previous session, we looked at that. I can have multiple current sources and the output of these current sources, that is actually fed to the oscillator. So, this is I_{osc}, I will just bring it down. So, this current source is fed to the oscillator like this. You are controlling the current. This control of the IDAC, this is your IDAC by the way and the digital bits which are coming D_I, they are going to control this.

Now, with respect to UP and DN signals, the way we can control these current sources that we have considered that you have these switches here, these three switches here and then all are biased with a certain bias voltage such that when the switch is closed, at that time the current in each branch is I_{BW}, if the switch is closed. Then this switch is controlled by your UP and DN signals.

So, during the off-state when the UP and DN signals both are low, at that particular time, you have only one switch closed. So, this is like you can say that there is a fixed current which is flowing here when you are having your UP and DN signals low. So, let me just first write it here what we want to achieve. Here with respect to your UP and DN signals, the current is not actually 0 or a large value, it is, so, you have this PDAC.

The output of the PDAC is like this. You have these UP and DN signals coming in. If I call the total current output which I will write this as I_P, the I_P current has the following format. So, you are having UP, DN and I_P, the current which comes out of this proportional DAC. When your UP is high and your DN is actually 0, this is one case or when your UP is 0 and DN is 1, these are the two cases and when this happens.

So, in this case, what will happen here is when UP is high, the total current which you would, which you are going to have is I_{BW}. When your DN is high, the current is 0 and when you have both as 0, at that time the current is I_{BW}. So, what you are going to have is when both the signals are 0, at that time the fixed amount of current I_{BW} flows through the proportional path. When your UP signal is high, you are having the phase error which is positive and you would like to increase the frequency, the current increases.

So, from here, if I just write the ΔI, you are having $+I_{BW}$ and when your DN signal is high, your ΔI is $-I_{BW}$ from the steady state. So, this is the logic which we need to implement here. So, let me just say when your UP is high and DN is zero, that time the total current should be 2I. So, I give $\overline{UP}$ signal, when UP is high, $\overline{UP}$ is low, at that time you have I_{BW} coming in.

And when you are having your UP is 0 and DN is high, at that time the current flowing through this particular logic is actually 0 only. So, if you look at it, you will have when DN is high, it is here. Now, in this case, just look at it that when UP is high and DN is also high, what happens in this format and this when UP is 0 and DN is 0. So, the thing which we have so far confirmed is when UP is high and DN is 0, UP is high, DN is 0, you get I_{BW} here and there is no current in this branch whereas when UP is 0, there is no current in this branch and there is no current also in this branch.

So, the total current is what you are getting is $+I_{BW}$. You are getting I_{BW} and the other one which you are getting is zero. In case when UP is 0 and DN both are 0, so if UP is 0 and DN is 0, then you will have I_{BW} current flowing through this and there is no current here because UP is 0, $\overline{UP}$ is 1, this value is 0. So, I_{BW} flows when UP is 0, DN is 0.

When UP is 1 and DN is 1, in that case, UP is 1 and DN is 1, this switch is open, this switch is closed, you have I_{BW} from here and you have 0 current from here, from this branch. So, let me just write it this way that you are having if UP is 1 and DN is 1, then you have I_{BW} current flowing in this branch and 0 from here.

So, in both these cases whether UP is 0 and DN is 0, you have I_{BW} current. When UP is 1 or DN is 1, then also you have I_{BW} current. When UP is 1 and DN is 0, then you have $2I_{BW}$ current total and when UP is 0 and DN is 1, then you have 0 current at the output.

Now the third switch which I have actually added here, this is for giving some amount of bias current in case you need. In case you need an extra current to bias the oscillator, you can always

have the same amount of current and this switch is always closed. So, this is an always close switch. So, just to understand this more clearly, I will now draw these switches controlled like this.

These are the switches and then the third one is something which you can always have as a fixed, always fixed bias current which is flowing through the oscillator and then you have the actual current sources. You can have different multiplication factor here. So, the size can be Y, here the size can be X in both the cases and they are controlled by your V_{bp}, this is $\overline{UP}$, this is DN. So, the total current which you are looking at here is I_P and when it is on, it is biased at such a voltage that the current which flows is I_{BW}.

So, now, just look at it for the case when your UP is 0 and UP goes high, then you have like this, just a waveform here to tell you how this I_P changes actually and similarly I will show you for the other case, this is one and then the other DN signal is like this for example, just an example here, this is UP and DN.

So, I am looking at I_P. So, when UP is 0 and DN is 0, that means you will have this transistor on, the current which flows, right now you remove this, just discard this extra current from the third branch, you only look at I_P, this is the control which you are having. So, $I_P = I_{BW}$. This is I_P and this value is I_{BW}.

When your UP goes high which means $\overline{UP}$ goes low, when $\overline{UP}$ goes low and DN still remains low, then you have extra current coming from this transistor, the total current will increase to $2I_{BW}$. When UP and DN both become 1, at that time these two transistors will exchange their roles and the current will still be I_{BW}.

It will remain I_{BW} like this. When your DN signal goes high first and UP still remains low, both these transistors are turned off and the current actually becomes 0 and then when your DN signal remains 1, UP signal goes high, this transistor turns on and the current becomes I_{BW} and it remains I_{BW}, the role only changes.

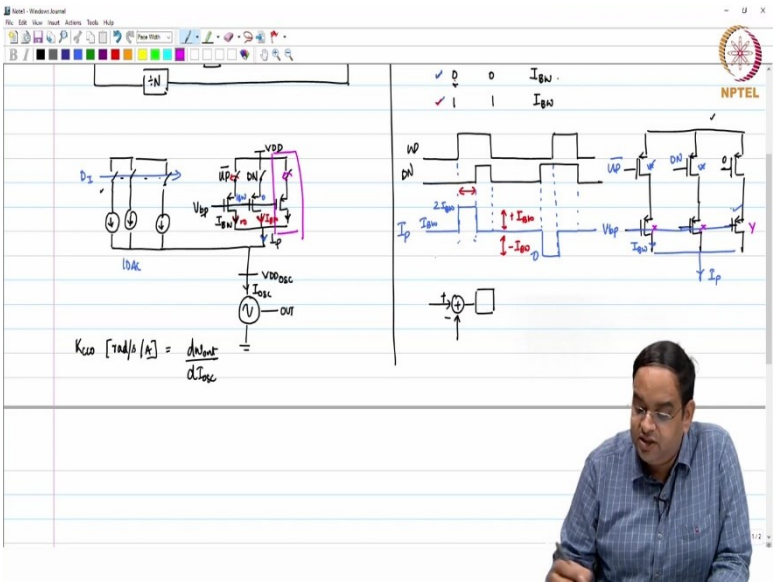
So, what you are seeing here is that from the bias point, you are seeing $+I_{BW}$ and here you are seeing $-I_{BW}$ and this phase error whatever phase error you have from the PFD, this phase error is converted to your I_{BW} for that time which is going to change your oscillator's frequency.

So, we also define K_{CCO}, the gain of the oscillator with respect to the current. The units are going to be you can say rad/s/A. It is the current gain, it is defined as,

$$K_{CCO} = \frac{d\omega_{out}}{dI_{osc}}$$

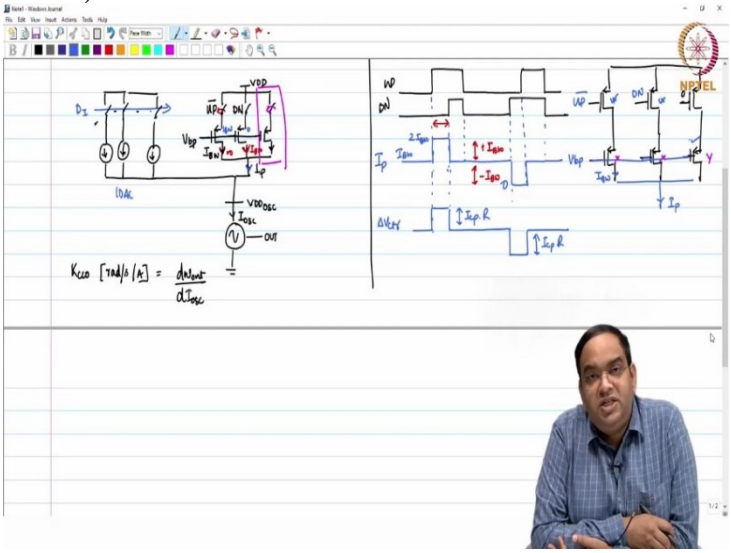
Now, IDAC is controlled in this manner because there are many bits. PDAC has the pulse width modulated signal. This is the best way to control the oscillator's current without having any quantization noise in the proportional path. So, given this, now you see that the control through the proportional path is going to be your I_{BW} times the phase error.

(Refer Slide Time: 16:43)



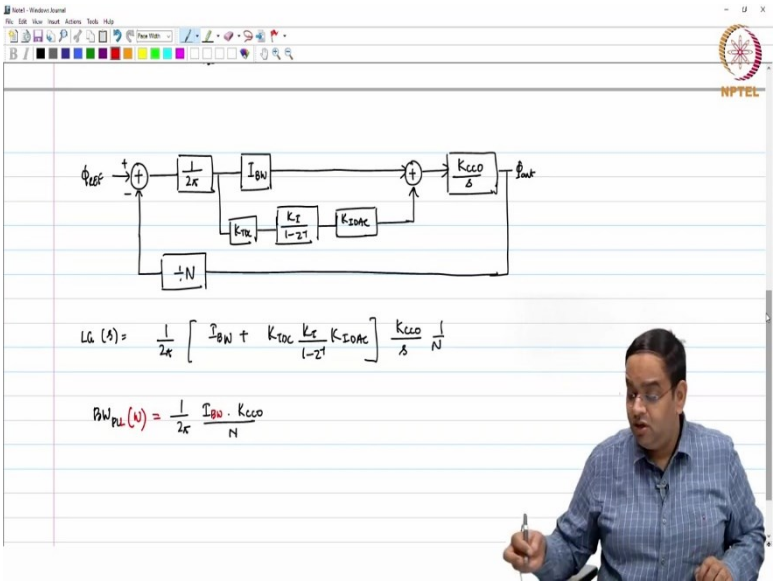
So, I will just draw the small signal model of the PFD now which has this phase error, + and - , the phase error you have if you look at the average current which flows out of this proportional DAC, that is going to be $\frac{I_{BW}}{2\pi}$ with respect to the phase error. This is the same thing which you had in case of your PFD output with $I_{CP}R$.

(Refer Slide Time: 17:26)



If you recall your voltage control ΔV_{ctrl}, let me just put it back here, your ΔV_{ctrl} because of the proportional path was in this case it would have been $I_{CP}R$ and this would have been 0 and this case would have been $-I_{CP}R$ whatever your charge-pump current is, that was the proportional path gain, this is the voltage and $I_{CP}RK_{VCO}$ gave you the change in the output frequency.

(Refer Slide Time: 18:07)



Now, similarly, what we are going to have here is the gain is here I_{BW} because that is the change by which the output current changes. So, we have this phase error, gain $\frac{1}{2\pi}$ and from here it changes, you have this I_{BW}, this goes to your VCO but the VCO has the other factor also coming in from the integral path.

So, here I will write this as $\frac{K_{CCO}}{s}$. The output of the PFD what you are looking at for the phase error detector, that particular PFD output goes to your TDC, so you have K_{TDC} here, then you have this transfer function which is $\frac{K_I}{1-z^{-1}}$, then you have IDAC where you will have the gain of the IDAC.

So, I will write that as your K_{IDAC} and all these things actually add up at the CCO's input. So, this comes here, this comes here and this is your output. Here, you have $\div N$. This is your φ_{REF} and this is your φ_{OUT}. So, in this particular case, what you see is, well, the loop gain is given by,

$$LG(s) = \frac{1}{2\pi}\left[I_{BW} + K_{TDC}\frac{K_I}{1-z^{-1}}K_{IDAC}\right]\frac{K_{CCO}}{s}\frac{1}{N}$$

This is the loop gain and from our discussion in the previous session, the bandwidth of this PLL is actually your proportional path gain. We worked it out earlier.

So, proportional path gain or you can say bandwidth of the PLL, this is hybrid PLL, analog/digital PLL, is your proportional path gain which is given by,

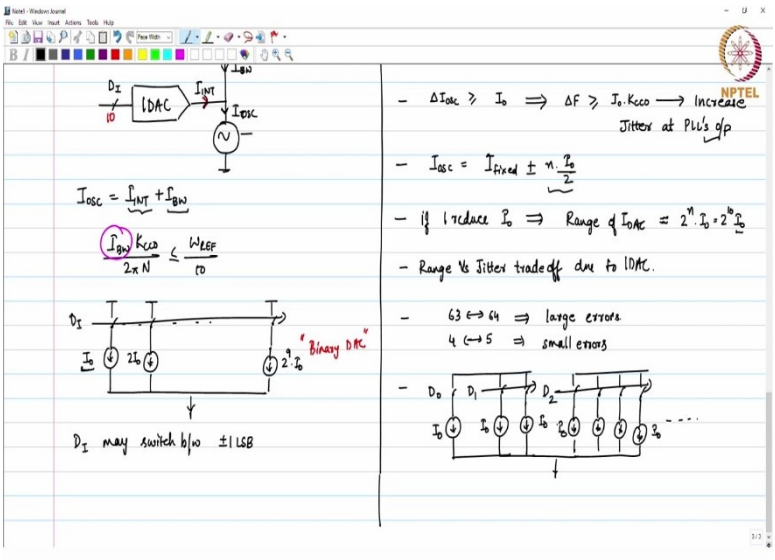
$$BW_{PLL}(\omega) = \frac{1}{2\pi}\frac{I_{BW}K_{CCO}}{N}$$

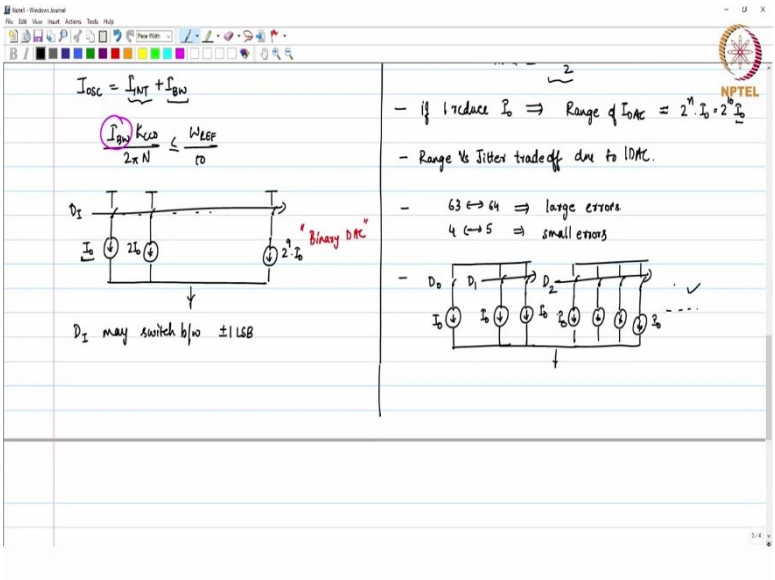
Now, you may realize why I used the term bandwidth initially because that actually turns out to be defining the bandwidth of the PLL.

So, if you want to increase the bandwidth of this PLL, then you have to increase this current source the PDAC gain by changing the current in the proportional DAC. You can work this out the whole loop gain and other things, what you will find is that the exact calculation will also give you bandwidth which is quite close.

So, now, given this hybrid PLL analog and digital PLL and the proportional and digital path control, let us look at how we are going to control through this IDAC because this IDAC normally has a problem of your resolution versus range trade-off. So, what is typically done, so, let us just if we just consider the IDAC path and the CCO.

(Refer Slide Time: 22:32)





So, we will only look at IDAC and the current controlled oscillator. So, what we have here is a large number of bits coming from the digital accumulator going to a current DAC, this current DAC directly controlling the current in the oscillator or you can say the frequency of the oscillator. So, here I will call this as I_{INT}. So, now converting from D_I to your IDAC, the thing is that the total current which is required for the oscillator to oscillate at a given frequency in our case.

So, we also see that there is an additional current which is flowing in is I_{BW} coming from the proportional path. So, total current this is I_{INT} and this is I_{total} or I_{osc}. So, we have,

$$I_{osc} = I_{INT} + I_{BW}$$

There will be a limitation on I_{BW} because the bandwidth which you can have is limiting your I_{BW}.

So, we all know that the bandwidth of the PLL here is,

$$\frac{I_{BW} K_{CCO}}{2\pi N} \leq \frac{\omega_{REF}}{10}$$

So, this is going to limit your I_{BW} and quite often it may not also always be possible to have this bandwidth as the optimized bandwidth. Your actual bandwidth may be even lesser than that. So, I_{BW} is limited.

So, for the oscillator to oscillate at a given frequency, a major portion of this total current comes from I_{INT}. Now, when the major current is coming from this I_{INT} path, the problem is that all

this current is based on the DAC which we are using right now. So, just take an example, if I have 10 bits at the input.

So, these 10 bits which you are having, you need to convert this using a 10-bit DAC, we saw this thing in the previous session that I need to have the current sources like this. We looked at this in the voltage DAC. Now, I am looking at this in the form of a current DAC. A minimum of 10 unit elements are required, 10 elements are required such that the current is I_0, $2I_0$ and similarly $2^9 I_0$ and then these switches are controlled by your D_I.

Now, the problem here is the following, in steady state, your TDC which you have seen before, this particular TDC will keep on having because if you want to use 1-bit TDC or any other, in the locked state, you will keep on flipping the TDC output between 1 or 0. Let us say even the last bit will keep on flipping 1 or 0 which will keep on changing your current or your D_I by a minimum of $\pm\frac{1}{2}$LSB.

So, in the locked state, D_I may switch between ± 1LSB or it can also have, in the steady state whatever value it has, it may switch between ± 1LSB or between 0 and LSB and that way your current is going to be switching between whatever 0 and I_0. So, you can say $-\frac{I_0}{2}$ to $+\frac{I_0}{2}$ and if the changes are a little more, then you will have maybe few LSBs $\pm$ few LSBs.

So, what happens is that in response to this minimum LSB size which you are having, this current actually changes if I write from the integral path point of view, we have,

$$\Delta I_{osc} \geq I_0$$

Every clock cycle, ΔI_{osc} is changing by I_0. So, you can say there is 0 or I_0 or $-I_0$ to $+I_0$, that is the minimum, it is always going to be greater than that which implies that,

$$\Delta F \geq I_0 . K_{CCO}$$

This is the change in the output frequency. So, if you have an oscillator where your frequency is changing randomly but every other cycle it is changing by $I_0 . K_{CCO}$, this is going to create deterministic jitter or you can say this will increase jitter at the output, at PLL's output. Why does it increase? Because steady state requires some fixed value of current but the current whatever that fixed value is, from that fixed value, the current is always changing like this.

So, I can write that in steady state,

$$I_{osc} = I_{fixed} \pm n.\frac{I_0}{2}$$

where $n = 1, 2, 4$ or so on. So, this is going to create the change in the frequency and if your frequency changes every other clock cycle, you will have the additional jitter at the output. The way to address this is that I go and keep on reducing my I_0. The way I have told you that I am having 10 bits but actually the bits which are coming from your integral path are many more because you are integrating it whatever bits you have.

If you are having 10 bits and you are implementing in this manner, that is the max which you can do. Now if I say I am going to reduce my I_0, if I reduce I_0, then what happens is the total range of the frequency variation by IDAC reduces. What is the total range? The range of IDAC is given by,

$$\text{Range of IDAC} = 2^n.I_0$$

In our case, it is $2^{10}.I_0$.

So, if I am going to reduce I_0 to reduce this jitter, then I am reducing the range of the frequency which can be controlled by IDAC. So, there is a range versus you can say resolution or range versus jitter trade-off due to IDAC, this you will see. There is one other problem. The other problem is implementing the way I have shown you this is like a binary DAC where each current source is actually having 2^n times the current value.

Now, this DAC itself suffers from a lot of non-linearity. So, it is like this when you are switching from 63 unit elements to 64 unit elements, all the LSBs have to be turned off and only one MSB turns on. So, this is the point where you will mostly see errors, big errors, then if you are just switching like from 13 to 14, if only one LSB switches, then the error is different though.

So, example is for the IDAC switching from 63 back and forth to 64 will give you large errors in this kind of DAC implementation whereas if you are switching from 4 to 5 where only one LSB switches, not all the current cells switch, that will give you small error. So, error actually depends on though it is only one LSB switching but it depends on which bits are being switched.

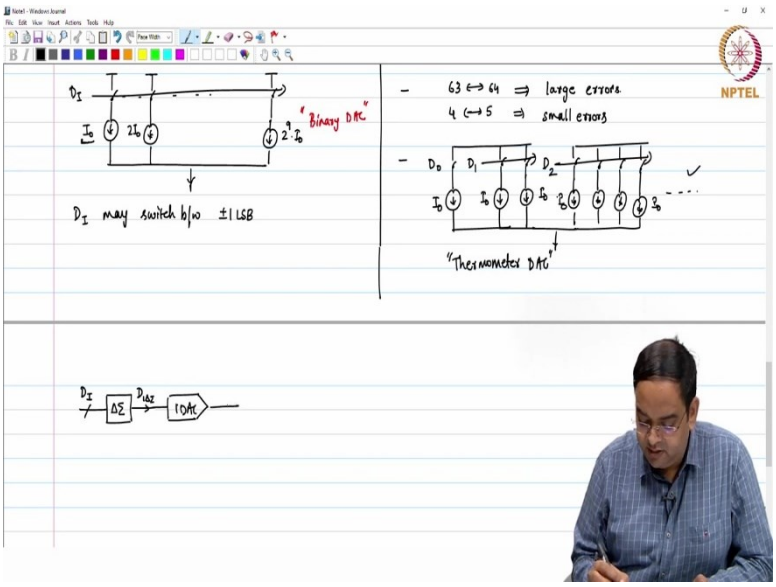
And in order to make sure that there are no errors or the errors are limited, well, you have to have a lot of matching requirements. Even then such problems will exist. So, what other thing is done to implement this DAC is to have better linearity that all these elements they are having the same unit element. So, if this is controlled by D_0, then you can have binary input, but the control is more of a thermometer one.

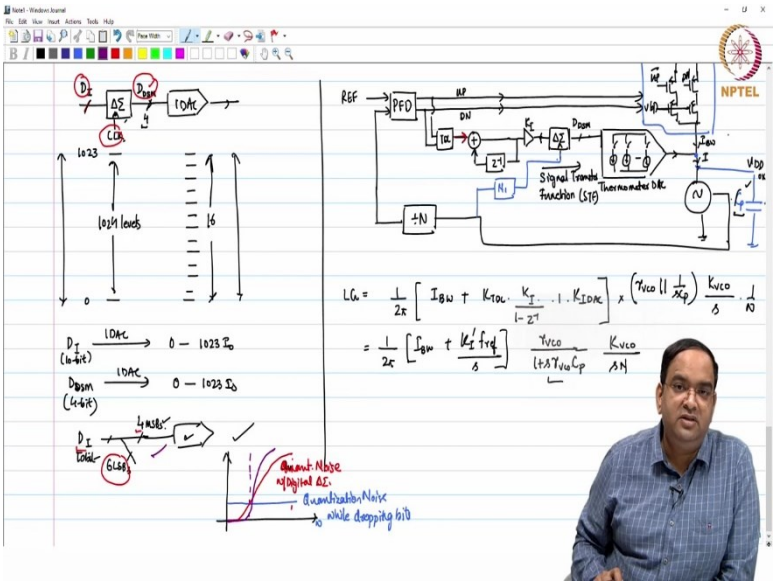
So, this is I_0, this is also I_0 and this is also I_0 and these two are controlled by D_1 bit. Similarly, you are going to have four current sources like this for D_2 control and all the current sources are like this. So, you have to spread it out and if you implement in this particular manner, then the matching between the current sources is good.

The errors while switching from one LSB to the other LSB, one value to the other value while changing only one LSB will be minimum but as you go ahead and do this, the number of unit elements which you need to have will just increase in the order of 2^n. So, you can see now for 10 bits, I need to have $2^{10} = 1024$ elements which is a problem.

So, looking at both of these things, what we do is the previous approach is a good approach in terms of minimizing the errors but it is quite cumbersome to implement and manage the matching between them.

(Refer Slide Time: 34:49)





So, what is done is that your D_I which is coming from the, this is a digital signal which is coming from the integrator is actually quantized using a digital delta sigma modulator. So, do not worry much about it. I will tell you the basic operation. So, you can say D_{DSM}. This is implemented using the thermometer DAC as you see here. This is by the way you can call this as a thermometer DAC. This is implemented using a thermometer DAC here and this will give you the current. If you want to convert into a voltage, you can convert it into a voltage.

Now, what is happening while going from D_I to D_{DSM}? So, going from D_I to D_{DSM}, what delta sigma does is, this is a digital delta sigma modulator which takes this whole like a 10 bit number and it may give you depending on what you want, a 4 bit number or a 5 bit number, a lower bit number.

So, specifically what we are saying is this. Just think it in this way that D_I range whatever these digital bits you need to think, you are having 1023 here, these are the levels for digital, I am writing their decimal equivalent. So, there are 1023 intervals here. You convert this whole unit to only let us say few intervals. If I am saying 4 bit, we are having only I need to have more, but we need to have only 15 intervals.

So, the range is maintained in both the cases, but here you have 1020 is just an easy number, 1024 levels, here you have only 16 levels for a 4 bit. It is effectively saying I want to match the total current. So, for example, if this D_I bit while going through an ordinary DAC would have given me, D_I digital control going through ordinary DAC would have given me a current range of $1023I_0$.

Even D_{DSM} going through an ordinary DAC should also give me $1023I_0$. It should, this range is maintained, just that is the important part, but this here is a 10 bit number and this here is a 4 bit number. You can think about it, the easiest way to do is that I take this 10 bit number D_I. So, I am having this 10 bit number coming in, I drop 6 LSBs and I just take 4 MSBs and I control the same IDAC.

So, if you have only 4 MSBs, then you need only 4 binary weighted current sources or maybe 16 current sources which are with the same unit element. So, this is one way of changing from a 10 bit number to a 4 bit number while maintaining the range. The other way is that you change using delta sigma. The difference between this method and delta sigma method is that when you are dropping the bits like this from the actual value, you are having a quantization error added to the signal because the actual signal is this but you are representing by only 4 MSBs. So, whatever is the rest, that appears to be, that is now a quantization error.

Similarly, when you are converting from D_I to D_{DSM}, you will also see the quantization and the only difference is that this quantization error when you are going to see at D_{DSM} output, this quantization error is noise-shaped. So, we will not go into the detail of how this noise shaping happens. You can very well read about it. The idea is that you convert a large bit number to a smaller bit number using delta sigma such that you shape the quantization noise while converting from the larger number of bits to a smaller number of bits.

And this delta sigma modulator actually operates at a certain clock. Depending on what the clock frequency is, you will see the noise shaping, but for now, you can say that this is a technique which is used to reduce the number of bits. Now, implementing a 4-bit thermometer DAC here is much easier than implementing a 10-bit thermometer DAC.

Now, given this that I can control the PLL in this manner, let us just put everything back and then talk about how we have actually solved the problems or what are the problems remaining. So here, now what we have done is this PDAC is something we know now, so, let me just do this. Your UP and DN signals, they are going to come to an oscillator and this is D_I.

So, this D_I is going through a digital delta sigma modulator, this is clocked. The output of this digital delta sigma modulator will mostly have lower number of bits D_{DSM}. This goes through a thermometer DAC like I will just give you some block representation of that. This thermometer DAC is like switches with currents. By the way, you can also convert this to a voltage and use that. So, this DAC now controls your oscillator.

Similarly, your UP and DN signals which are coming, you can control it in this manner that you get $\overline{UP}$ here and DN here and these two are control, you can have a fixed bias current if you require, this is a bias voltage V_{bp}, these two are also going to give the current here. So, this block is your proportional DAC and the other block is your integral DAC. So, these two signals come here. So, what we have done so far is we have simplified, or you can say now we know how to control our PLL using these current DACs. So, this is thermometer.

Now, the problem as you go on implementing and realizing a certain resolution, mostly you will not feel that much of an issue with the proportional DAC. But with IDAC implementation, the range versus resolution trade-off will always be there. So, you come up with some other ways, you can give extra current as required. That will be for you to read about and then you can actually create, you can have an extra DAC which feeds into this oscillator, that is perfectly fine.

Now, when we are going to model this PLL, there is one thing which we have not added so far that this node though we are controlling only the oscillator current, this node actually comes with a good amount of capacitor. It is not that this node is you have I and that it directly goes through the, I goes to the oscillator. So, let us call this as C_p. So, what happens is your loop gain transfer function now, let us say this is you are having whatever current gain you have, that you are adding here.

So, you will see a change in the loop gain. I am going to have,

$$LG = \frac{1}{2\pi}\left[I_{BW} + K_{TDC}.\frac{K_I}{1-z^{-1}}K_{IDAC}\right] \times \left(r_{VCO} || \frac{1}{sC_p}\right)\frac{K_{VCO}}{s}.\frac{1}{N}$$

So, from here to here, what we have is the signal transfer function or STF remains same. So, this gain from the input of the delta sigma to the output of the delta sigma for the actual signal is the same. So, that gain is 1 and then you have K_{IDAC} whatever gain we would like to implement, that is what you have. All this current, it does not directly flow through the oscillator. It sees this capacitor also.

So, the modeling requires that we take this I whatever the current goes to the oscillator and then we find out what is the output frequency. So, here we come up with this logic that in steady state when you have all the currents fixed, there is only the noise and the variations you have,

I can model the VCO using its equivalent resistance and this particular current goes through this $r_{VCO}C_p$, that is the pole.

So, you can see this current goes through this oscillator and the capacitor which is $r_{VCO}||\frac{1}{sC_p}$ to give you the control voltage whatever you have, VDD_{osc}, and then from that voltage to output frequency, we know it is $\frac{K_{VCO}}{s}$, this is $\frac{1}{N}$. So, to simplify it further, we have,

$$LG = \frac{1}{2\pi}\left[I_{BW} + \frac{K_I{}' f_{ref}}{s}\right]\frac{r_{VCO}}{1 + sr_{VCO}C_p}\frac{K_{VCO}}{sN}$$

So, now, you see we started with a digital PLL which was only second order. Because of this capacitor and the kind of control which we have, it becomes third order and you have a third order system, 3 poles and the pole is mostly dominated by C_p. So, now, I will bring back that we added the ripple capacitor earlier, ripple bypass capacitor in our regular analog charge-pump PLL. In this case, we did not add an extra bypass capacitor.

But the parasitic capacitor which is existing at the VDD of the oscillator, it provides you the third pole and it will help in reducing the ripple because of this proportional path or because of whatever residual ΔI you are going to have from IDAC, it is going to have the ripple there and it will help in having the ripple bypass.

Now, whether we want to go ahead with whatever the ripple bypass capacitor gives us or if we want to change this particular pole position, we can go ahead and add extra capacitor also. Now, this is the 3 pole, 1 zero system and you have to just keep it stable. So, you can go with the same process which we used initially that we want to have the maximally flat phase response at the unity gain frequency, we would like to have the positions of poles and zeros.

So, with that method, you can find the location of C_p or the value of C_p, the gain $K_I{}'$ and other values. So, this is a very basic hybrid analog and digital PLL with all the given controls. One other thing which we need to know here is that when you are doing the quantization from D_I to D_{DSM}, a lot depends on this frequency, clocking frequency.

If this clock frequency is higher, then your noise will be filtered off more, the quantization noise will be filtered more and just to tell you that to differentiate between this conversion from directly dropping 6 bits or doing through delta sigma, the difference is this that the quantization noise in directly dropping 6 bits, that quantization noise spectral density will be flat. This is

quantization noise while dropping bits with respect to ω. This is a noise spectral density, it is going to be flat whereas when you are doing the quantization using digital delta sigma, it is going to be something kind of this, it is called noise-shaped.

So, within a certain bandwidth, within a certain band, if that happens to be the bandwidth of the PLL, the noise is much lesser. So, this is your quantization noise with digital delta sigma. So, this kind of noise shaping actually depends on the clock frequency which you are using. So, quite often this also happens that you do not clock it at the reference frequency. You may take this clock and maybe divide by some factor N_1 or some value and use this clock to shape this noise better or something like this.

Those are the details which one can look into when one is designing the digital delta sigma modulator. If you are not using digital delta sigma, you can very well go ahead with this. But then you have this quantization noise, you have to have a lower bandwidth in the system. Thank you.