VLSI Data Conversion Circuits
Prof. Shanthi Pavan
Department of Electrical Engineering
Indian Institute of Technology, Madras

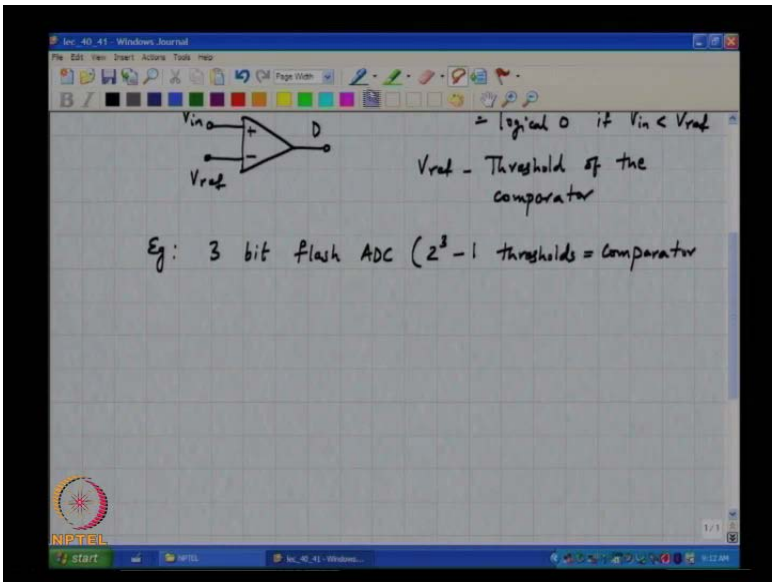
Lecture - 40
Flash ADC Design

Good morning, this is VLSI data conversion circuits lecture 40. So far we have seen how to design a continuous time delta sigma loop, that is how one finds a loop filter coefficients, how one scales them for proper swings, how one can go and realize integrators, which are basically the main parts of the loop filter. The next thing do is to look at the quantizer part, the quantizer as we all know consist of the A to D and the D to A converter, so today we will start looking at the A to D converter.

Since, at delta sigma loop is basically a feedback based structure, the kind of architecture that you would use for the A to D converter is the one which has the least amount of delay. So, there are several ways of realizing A to D conversion you all know that and some of them can be very hardware efficient, but then will take a long time and that make sense, because basically for hardware efficient you need to accomplish a task. So, you keep reusing the hardware again and again, which means that it will take a long time to accomplish the conversion.

On the other hand if you want to do everything very quickly, it means that you pay a price in terms of the extra hardware required. When you want to do things quickly what you would do, would be to work in parallel, and once you do things in parallel, it means that usually you waste a lot of resources in the sense that you are doing. You know redundant things with much of the hardware, what that is the price you pay for speed.

(Refer Slide Time: 02:13)



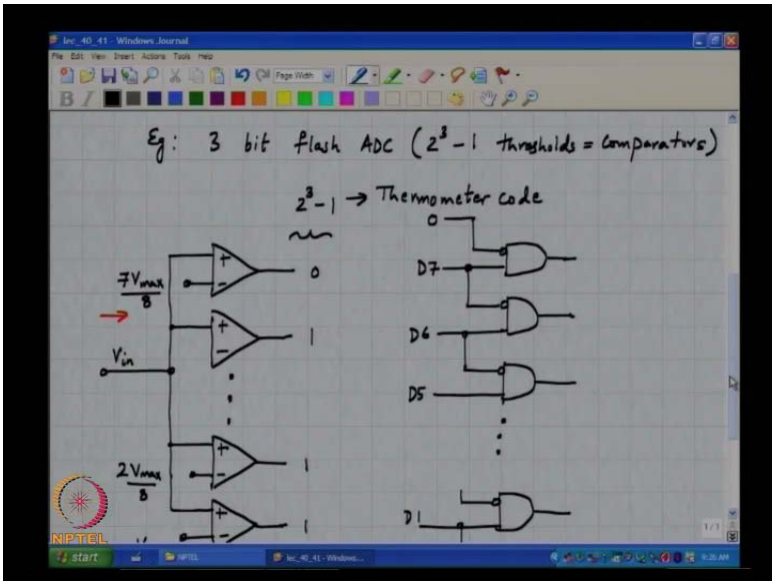
So, the considerations for the A D C in the sigma delta loop must be low latency, and thankfully this is not a too much of a problem, as we will see going forward. And the basic idea is to therefore, use the family of converters which work the fastest and these are called flash A to D converters. Flash A to D conversion is not just restricted to use in delta sigma loops, it happens to be the best choice in a sigma delta loops simply from the point of view is speed.

Flash converters are also used in other systems, where feedback is involved to close some kind of loop, and in such cases the latency of a converter has to be as small as possible, and this is what motivates the use of a flash converter. Now, the basic idea behind a flash converter is very straight forward, what is the A to D converter basically trying to do, it is trying to compare the analog input to a set of levels. The simplest of the most straight forward way of doing this would be to have a basic building block called a comparator, which in principle gives an output D.

Where, D is a logical 1 if v in is greater than v ref, and is a logical 0 if v in is less than v ref, now v ref is the, so called threshold of the comparator. Now, since the flash is basically trying to compare an analog input against a bunch of levels a straight forward implementation would be to say, I will have an array of comparator. For example, for a 3 bit flash one would want to classify the input into 8 bins, so how many thresholds would be needed.

You need 7 thresholds, so in general for an n bit flash you need 2 to the n minus 1 thresholds, which is equivalent to, so many comparators, and you have 8 of these guy's or 7 of these guy's.

(Refer Slide Time: 06:05)



The input is common to all these comparators and the references go from v max by 8 to 2 v max by 8 all the way to 7 v max by 8, this is v in. And you therefore get a code at the output, which is what you get at the output is basically a string of 1's followed by a string of 0's, at depending on the location of the input, and how many bits are here, I mean how many lines are here.

Student: 7.

Student: 7.

In general 7, which is 2 to the 3 minus 1. So, as you can see if the input lies say between the 6th and the 7th comparator, all the comparators below the input will read a 1, and all the comparators above the input will read a 0. So, if everything is working properly, then this code which is 2 to the n minus 1 bits long, consist of string of 1's followed by a string of 0's. And this is given a special name right depending on the level of the input the number of 1's keeps varying, just like the mercury in a thermometer and this is why this code is called a thermometer code.

Now, the thermometer code is a somewhat of a special code in the sense that, even though you have 2 to the n minus 1 bits, all possible sequences are not allowed only those special sequences which are all 1's flowed by all 0's are allowed. And you can see clearly the redundancy already present, because in principle if we know that this comparator is showing the 1, it means that the input is higher than 6 v max by 8, there is in principle no need for us to.

Student: ((Refer Time: 10:27)).

To look any further below. Let if say, if it is less than I mean is greater than 6 v max by 8 it automatically follows that it is greater than 5 v max by 8 and all the way up to v max by 8. However, there is no time for all that to do, we decide to do we compare it with all the thresholds anyway that is why you wasting hardware, so in principle the what really you care about is the transition between the.

Student: ((Refer Time: 10:59)).

The all 1's and the all 0's, and in principle all the comparators above this one, and all the comparators below this one are basically redundant, as far as simply figuring out what the input is you understand. So, that explains why a flash converter is fast, and also explains why it is you know within codes very wasteful of hardware, so it is naturally only used in applications, where a latency is paramount, and b where the hardware complexity is modest.

So, if you want to do a regular standard flash, you can see that every time you increase resolution by 1 bit the number of comparators actually doubles, so the hardware complexity goes exponentially as the number of bits, is this clear. So, typically 6 bits is a breakeven point for standalone flash converters, because there several difficulties associated with increasing the number of comparators beyond about 64, which what you will need for a 6 bit converter, somebody has to drive the flash.

And as we know, usually you would like to sample and hold the input, and then work off that sample that is what you are trying quantize you understand. So, what happens is that the circuitry which drives the array of comparators, has to now drive a larger and larger load, the moment the number of comparators keep increasing.

So, for standalone flash converters typically 6 bits is a breakeven point, beyond that you know people try and see either explore ways of avoiding the flash converter, or try simply I mean some innovative techniques should try. And keep the number of comparators low, while still giving the resolution you want may be 7 bits or 8 bits, but standard flashes are usually limited to about 6 bits, as you will find out if you just browse the literature.

Now, as in the particular case of a delta sigma loop, driving a big flash array cause a lot of power, because the loop filter is driving the array of comparators is not it. In a flash converter at a pinch you can say well you know I am only designing the comparator array, I mean I do not really care about whose driving the array. So, it is not my problem, so I have you know 6 bits, but in a delta sigma loop, I mean the same guy is probably designing the flash as well as the loop filter, which means that the array of comparators has to be driven by the loop filter, which is being designed by the same person.

So, you now doubly careful, because if you try and make life easy as far as the sigma delta loop is concerned by choosing a large resolution for the flash, then driving it becomes a problem. And causes if you drive a larger capacitance, you know for the same power it means that it will take a longer time, so is associated with not only the hardware complexity of a 6 converter, but also the delay and drive needed to drive such an animal. So, in sigma delta loops typically you know people have restricted themselves to 4 bits, again as you will find upon per using the literature.

So, 4 bits seems like reasonable tradeoff between the benefits offered by multi bit modulation, which are a increase in the maximum stable amplitude for a given noise transfer function. or you can make the more the resolution is quantizer. The larger you can make the out of band again of the noise transfer function thereby reducing the in band noise, to the if you use an NRZ type feedback DAC, the effect of jitter is greatly reduced if you use a multi bit modulator.

And as you keep increasing the resolution of the quantizer, on the other hand doing the quantizer itself becomes more and more difficult as you keep increasing the number of levels. So, reasonable pin point beyond which the benefits offered by multi bit quantization are overweighed by the hardware complexity involved in making the quantizer itself, restrict the number of bits to about 4. Now, the output of the flash needs

to be a binary code finally, between in this particular example 0 0 0 through 1 1 1, but what we have at the output of the comparator array is a 2 to the n minus 1 bit code, which is a thermometer code.

So, to figure out which bin the input lies in what we are interested in is figuring out the transition between the all 1, and the all 0. So, as again very straight forward simply, because in principle what you can do is to find the transition, so, this becomes a all. So, this becomes D 7, this is seventh decision of this seventh comparator D 6, D 5, etcetera, and all the way up to, and ((Refer Time: 18:33)), so this must always be a 1, so how many will be there now 8.

So, this is the digital hardware required to determine, whether I mean where the transition occurs between the all 0's, and the all 1's, as we can see it is very straightforward there’s nothing but, it the properties of this code are.

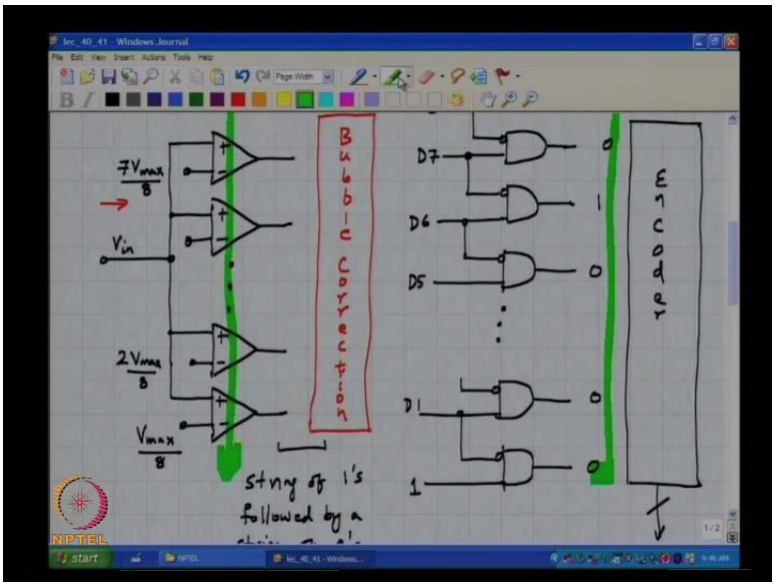
Student: ((Refer Time: 19:39)).

This is only of these 2 to the n wire is coming out, only.

Student: one of them.

One of them will be high all the others will be low, and this is often called even though it is 1 of 2 to the n, it is actually called the 1 of n code, so this is called the output of the thermometer I mean comparator array is the thermometer code. And the output of the transition detection is the in this particular example, you will get a 1, here and all l’s should be 0, this is the 1 of n code. Now, we need to find from this one of code you have to generate a binary word, which is 3 bits long, so what is the easiest way you think of generating the binary output, uses an encoder, so this goes into an encoder.

(Refer Slide Time: 21:11)



And the output of the encoder is a 3 bit word, is this clear, so in the early flash converters, what would happen would be that the comparator array would by itself be driven by the analog input not by a sampled and held input. And the comparators would implicitly have this sampling operation in there, in other words the comparators would not be something which compare the analog waveform.

You know with reference and then sample, they would be inherently the comparators would be what are called sampling comparators as opposed continuous time comparators. Where, the output at all times is supposed to be sigmum of input minus reference, you understand when you see the circuit you will appreciate this point a little better. So, the comparators used to have implicit sampling, and that mean that meant especially, when the number of comparators was large you would find that you would had to propagate the clock.

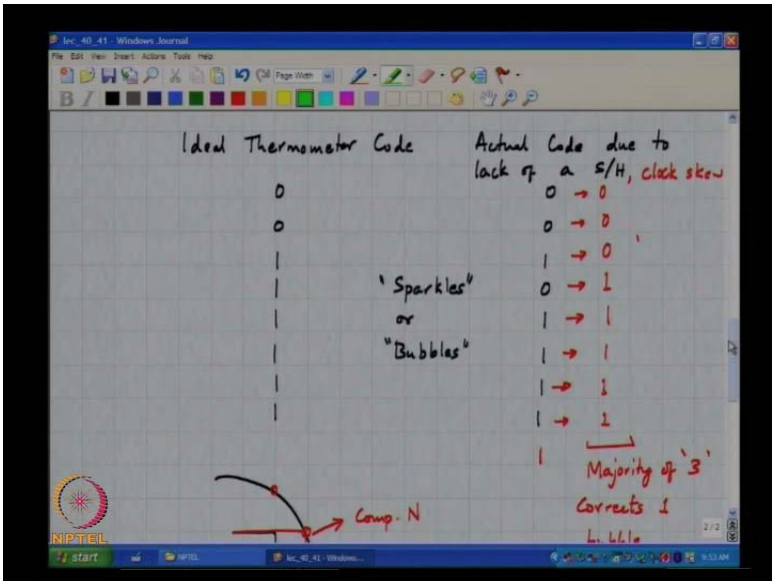
The sampling clock would be responsible; obviously, for the comparator sampling, and when you physically layout these transistors, because of the large number of comparators which would be laid out as an array. The clock which is responsible for sampling would have to travel from 1 end of the array to the other end of the array, so nominally all the comparators are supposed to be working of the same input. But, now if the clock is delayed across the array, then what happens is that in effect different comparators are making decisions on.

Student: ((Refer Time: 24:07)).

On inputs which are at different times. Granted that this difference in time is small, but in principle all these comparators are making, which are supposed to be making a decision on the same input, are in reality making decisions on inputs which are slightly displaced from each other in time. Now, this is not an issue, when the input is varying slowly, but remember the whole idea in having a flash used to be able to sample, really fast, which also means that the input signal is likely to be fast.

So, now, since the input is I mean different comparators actually digitizing slightly different inputs, one used to often end up with a situation. Where, the thermometer here would not be a pure thermometer, I mean ideally you are expecting a bunch of a string of 1's followed by a string of 0's.

(Refer Slide Time: 25:21)



In practice what 1 would observe would be ideal, let us say you were expecting something like this, actual code the lack of a sample and hold would perhaps look like this simply, because these comparators here are not looking at the same signal. You can always come up with a signal, which will give you this kind of thermometer output, simply because all the thermometers are not all the comparators are not digitizing the same input. And this is happening because of two things, one there is no sample and hold, if there was a sample and hold, what do you think will happen.

So, let us say this is the input and there is clock skew, so for instance this would be say the sampling edge at comparator one, and because of clock skew effects this would be perhaps the sampling edge of comparator n. So, the first comparator is actually digitizing this point the last comparator is making a decision on that point, and the comparators in the middle are actually, comparing something in the middle to their references.

So, since all these comparators are not working of the same input, there is a possibility this is actually a distinct possibility. Now, this is happening because a lack of a sample and hold, and b because of clock skew, clock skew is something which a fundamental, and you cannot avoid it because finally, all these comparators cannot be in the same place. They will be physically placed apart, which means that any clock that you run to these comparators will have a skew between the first and the last one, so how do you think having a sample and hold up fronts solve this problem.

Student: ((Refer Time: 28:41)) replace.

Student: Less than 1 clock cycle is same in the same.

The skew is definitely, you know at least your hoping that it will be much smaller than a clock cycle.

Student: We have sample and hold.

Student: Because, signal tend constant.

So, the problem as you can see is that, the signal is changing within this interval if you make it stay constant, the fact that you have clock skew, does not really matter. Of course, you have reduced amount of time to make a decision, but the slope is 0, therefore all the comparators are seeing the same input. So, having a sample and hold helps significantly in that way, so a lot of the earlier flashes, where there would not be a sample and hold upfront the performance would be for low speed inputs or low frequency inputs.

But, as a input frequency went on becoming higher and higher and got closer to Nyquist, you would see that the performance would degrade very rapidly, and the reason would be things like this. So, the moment you add a sample and hold up front that problem is

addressed, simply because once the sample is held it is flat with respect to time which means that all the current, I mean all the comparators see the same input.

Now, in the earlier days therefore, there would be situations like this, where since neighboring comparators are not really seeing the same input, you would get not a string of 1's followed by a string of 0's. But, ((Refer Time: 30:38)) fact like this. And these were called either sparkles or bubbles in the thermometer code, I mean clearly there is a problem, the problem is what.

Student: Due to clock cycles.

Student: After encoding we use a.

I mean in principle we do not know whether the input is here, so if we just use the standard hardware, then after doing the 1 of n encode you will get.

Student: 2 1.

2 1's and that will go into the encoder, and the encoder will give you the complete garbage, so this these bubbles or sparkles is a problem, while we do not know what that true input is at least one nothing we would like to make sure is that. We know that the input is in the neighborhood of this, so the final code that we get must not be widely of from this neighborhood, and that can be only ensured, when only one line of the encoder is.

Student: 2.

Is 1 if the encoder or I mean if the encoder input is a true 1 of encode, then only one row of that encoder is held high, which means that even if the true I mean decision is wrong where only off by a little bit may be may be an LSB or 2. Whereas, if 2 lines of the encoder are asserted high, then the output could be catastrophically off you understand, so the key point is to make sure that, only the output of the 1 of n code is truly 1 of n.

Which means, there at the input of the 1 of n coding logic, they must be a true thermometer only, then you will have only one transition, so what do you suggest we could do to fix the problem, what is the priority encoder, what do you do.

Student: It will take what is the highest value highest one.

What is highest one.

Student: Highest.

Student: Almost highest.

See one thing you must be aware of is that anything you do here, must be modula, one of the nice things about the flash. As you can see is that the entire logic all the way up to the encoder is.

Student: ((Refer Time: 33:48))

Is copy and paste of the same basic unit. Now, trying to do you know priority encoder, I suppose you could do it, but I mean is there anything else that you can think of.

Student: What a ((Refer Time: 34:16)).

No, what are you trying to do.

Student: Sir first 1 is error when you cannot get ((Refer Time: 34:23)).

Student: What we interested on every bits ((Refer Time: 34:31)) neighboring bits are even if the neighboring bits are 1 0's and input of the decoder as only one one.

Yes correct, so how do you ensure that.

Student: Giving that only the neighboring bits will be

Yeah.

Student: And then I mean add with the I mean ((Refer Time: 34:56)).

Student: The neighbors are more ((Refer Time: 35:12)).

So, basically the ideal is to say, now I am not the input to the transition detector, I am not going to give the actual thermometer code I get, because it could be an errors because of bubbles or sparkles. What I will do is, I will you know fix the thermometer code, so that it is a true thermometer, and one way of doing that is to say, the output of this particular comparator is not going to be just the output of this comparator itself, but I am going to look at my neighbours and give out a majority.

For example, if I do a majority of 3, majority cannot be of 2 you have to have an odd number of comparators, and if I did a majority of three; that means, that for this particular comparator, for instance the upper one is 1 the lower one is 1. So, I can say without any you know ambiguity, that this indeed a 1, similarly if I assume that the lowest 1 is always a 1, then this will give me a 1 and what will happen here, this will also give me a 1 this will also give me a.

Yes.

Student: One.

It I will be a 1, what happens here, majority of 3, where the 3 are the present comparator the 1 above and the 1 below, so it will give you a 1, what about this.

Student: 0.

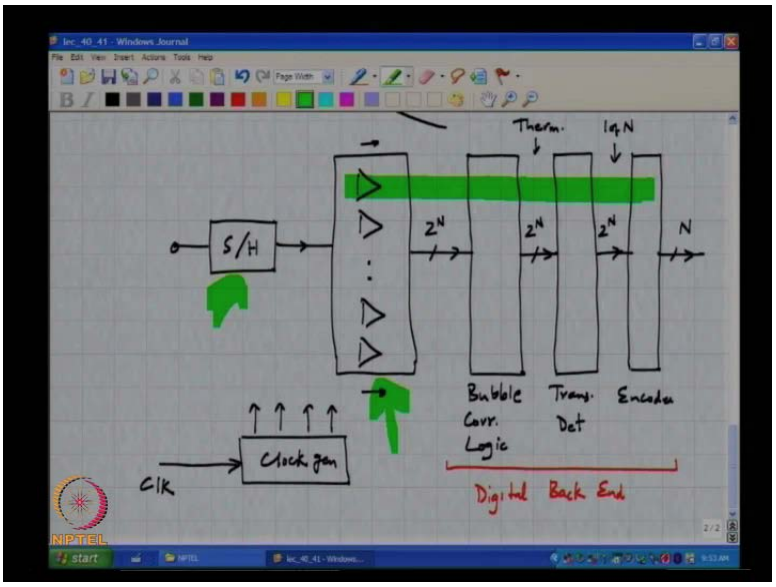
It will give you a 0, 0 and 0. So, this is what is called majority of 3, and this can correct 1 bubble, of course 1 can say if I do majority of 5, for instance I can correct 2 bubbles, but majority of 5 is a lot more messy then majority of 3. Majority of 3 means that physically 1 has to only talk to find the majority output, decision one has only has to talk to the decision above and the decision below, so routing signals becomes much easier. Now, if you do majority of 5, you have not only talk to one person below, you talk to the second person below as well as the second person above, which makes the whole thing very messy.

And your hoping that the probability of 2 bubbles is much smaller than the probability of one bubble, so what is there, here between the actual thermometer code and the one, that goes to the transition detector logic is basically, a majority detection logic whose job is to correct bubbles. So, this is basically what is involved in a flash converter, and therefore the latency of the flash converter consists of the amount of time the comparators take to make a decision plus some logic delay associated with the bubble correction.

The transition detect and the encoder, and your hoping that all this is done you known as soon as possible, at least architecturally there is no deliberate delay in the whole signal j.

So, the generic block diagram of flash converter will therefore, look like this, as we just discussed.

(Refer Slide Time: 40:10)



There will be a sample and hold upfront or a track and hold, where the job of the track and hold is to hold the analog input to a constant value, and advantage of this is that, now in spite of clock skew, which is going, which is bound to be there. All the comparators will see the same input, because the input is held, therefore it is derivative is 0. So, you have the comparator array, then it goes to the bubble correction logic, then it goes to the transition detect, so here you hoped to get a thermometer code here, you hoped to get a one of n code. And then you go to the encoder, and you get the n bits out, here it will be 2 to the n bits, 2 to the n bits, 2 to the n bits.

Student: ((Refer Time: 42:04)).

Now, you need to add the extra there is one here, and then one on top right, and all these have to be controlled by clocks, and for example, as we have already seen long ago for the sample and hold you need non overlapping clocks, and perhaps boosted 1's and so on. So, you cannot expect all, that all those clocks to be given to you from the outside, so all you will get is a single clock and from this single clock from a master clock, you have to generate all the other clock waveforms that you need, so this will go to various places on the chip.

Yes.

Student: sir, in this case we are using the sample and hold, so the bubble correction logic is ((refer time: 43:18)).

So, if you use the sample and hold strictly a bubble correction logic is not necessary, because the output is held, I mean a lot of I mean this is a you know, so but still people just use majority of 3 bubble detection, bubbles correction. Simply, because it gives you know it is like buying insurance, you hope that there is no bubble and, but you know it does not cost much too much anyway it is a, so you know you just have it. So, strictly that what is saying is strictly, if you have good sample and hold there should be no need to have bubble correction logic is this clear. So, this is the generic block diagram of a flash A to D converter, this is often what is called the digital backend.

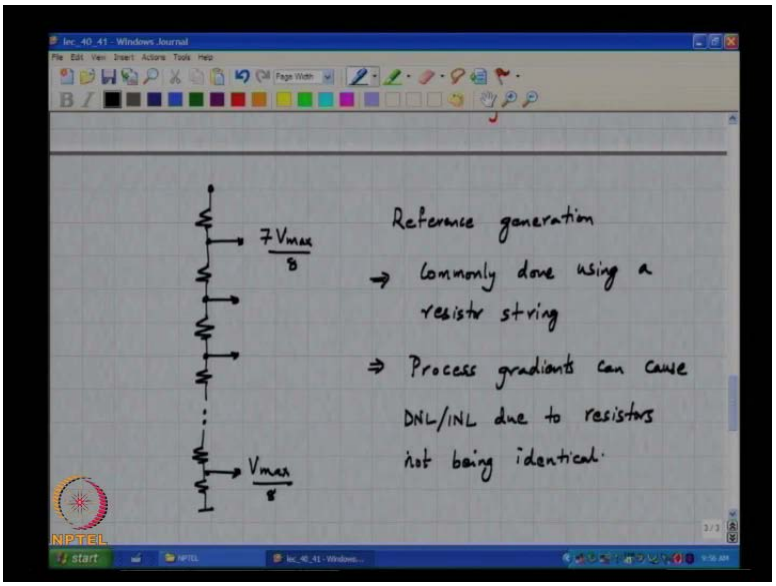
Student: ((Refer Time: 44:17)).

Another design of these are actually quite straight forward, I mean it is just logic design that is nothing much to it, and the key point as you noticed in the flash is speed. And therefore, you would like all these things to be as modular as possible, and the way we have done the logic it is definitely, so the design procedure would be to make the I mean you know design. One slice of this entire paths carefully, which is slice means comparator that portion involved with bubble correction transition detect and the encoder part carefully, then is copy and paste this 2 to the n minus 1 times.

So, you get you are able now get the entire flash A to D converter, once you done all that you must go and then design the sample and hold to be able to derive the input capacitance of the comparator array. The input is not truly capacitance as we will see going forward, it can have all sorts strain issues associated with kick back, and think like that, but this is how the design would proceed.

So, the key analog building blocks are; obviously, the sample and hold and the comparators, and before I proceed 1 thing I would like to point out is that the comparator array needs a bunch of references, can you think of a straight forward way of generating these resistances I mean of this references.

(Refer Slide Time: 46:41)



So, reference generation is commonly done, do not necessarily this way, but is commonly done using, so v max by 8, 7 v max by 8, so reference generation commonly done using a resistor string. So, as the resistors for instance very across the DAC please note that the flash array is physically big thing, so the resistors also had to be the resistor string has to go from one end of the array to the other.

So, if there is gradient of resistance along the DAC it the it means that these resistors are all not identical, and there is a gradient in these resistors, which will then boiled down to D N L and I N L in the converter, even if the comparators were ideal you understand. So, one thing must be aware of is that process gradients can cause D N L and I N L due to of course, a resistors are also miss matched statistically, and this means that you will get some small amount of D N L, I N L any way.

So, in the next class will take a look at how to design, I mean what one could do to design the comparator, we saw that the comparator is the key building block of the flash. So, we need to be able to figure out what all ways are there of taking a voltage comparing it to the reference, and making a digital decision a logic design has to whether the input is.

Student: ((Refer Time: 49:41)).

Smaller than the reference or larger than the reference.