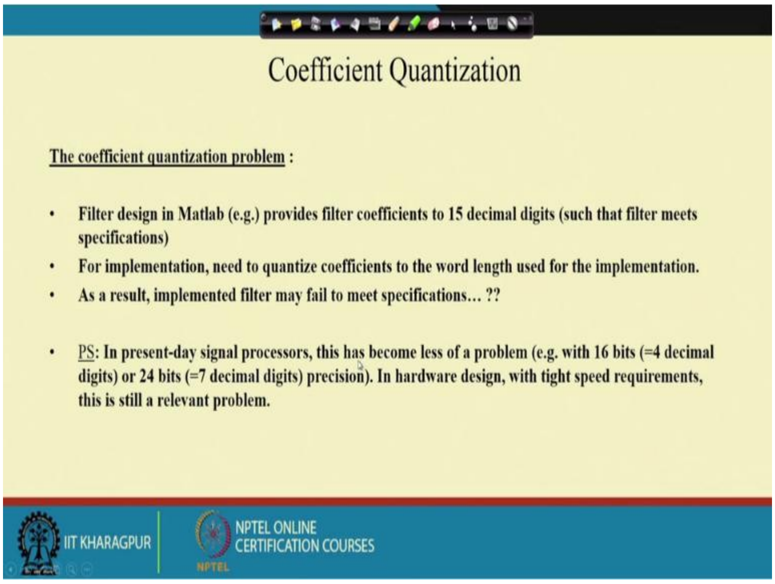
Architectural Design of Digital Integrated Circuits
Prof. Indranil Hatai
School of VLSI Technology
Indian Institute of Engineering Science and Technology, Shibpur, Howrah

Lecture – 40
Fixed Point Number Representation(Contd.)

So, welcome back to the course on Architectural Design of ICs. So, in the last class we have seen that Fixed Point Number Representation; that means, whenever we are implementing any system or any function or any algorithm in hardware, so, at the time we have to do it in fixed point domain. So, for that we have to convert from floating point to the fixed point, because of the conversion there is some error we basically gathers or we accumulate ok.

So, depending on the error the performance of the system that may be degraded ok. So, to for the degradation of the performance, what is the effect? So, we will take 1 example of any filter design and then we will see how basically, doing this or for this conversion of floating point to fixed point, how basically it effects the performance of the corresponding system design, so, that we will see in today's class ok.

(Refer Slide Time: 01:29)



Coefficient Quantization

The coefficient quantization problem :

- Filter design in Matlab (e.g.) provides filter coefficients to 15 decimal digits (such that filter meets specifications)
- For implementation, need to quantize coefficients to the word length used for the implementation.
- As a result, implemented filter may fail to meet specifications… ??

- PS: In present-day signal processors, this has become less of a problem (e.g. with 16 bits (=4 decimal digits) or 24 bits (=7 decimal digits) precision). In hardware design, with tight speed requirements, this is still a relevant problem.

So, suppose we have to what I said in the last class that, whenever we are doing this a filter implementation, so at that time, we have to that means, we have to bother about the corresponding of conversion in 2 area; that means, one is for the coefficient, another for the arithmetic operation, whatever we are doing for filter implementation ok. So that means, for the coefficient quantization, the quantization of the coefficient that may be creates the problem ok. Suppose in MATLAB, if I take the corresponding, what I said that the corresponding the coefficient values they are basically in the fractional domain as the maximum value that may not be greater than 1.

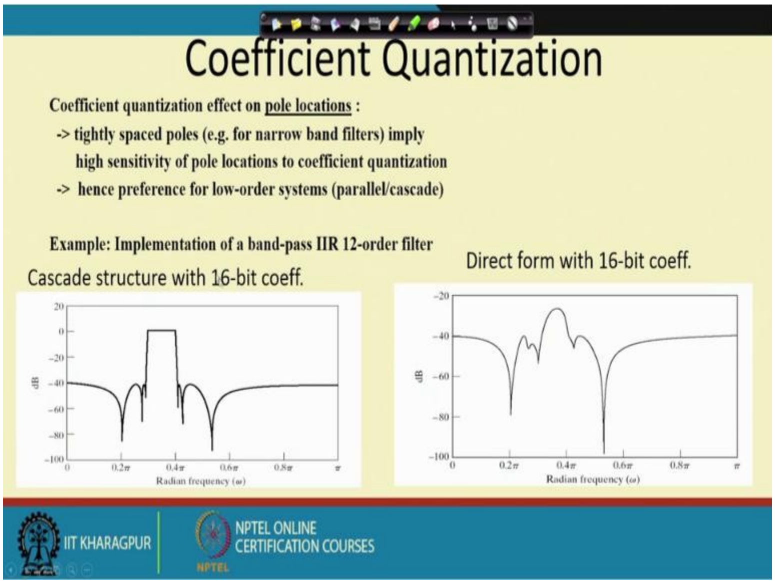
So that means, all the coefficient values they are represented in fractional bits only ok. So, what will be the exact value or sorry, what will be the exact word length to represent each of this coefficient that is basically very much critical ok. So, if I just consider that the filter coefficient that may be represented, if it is represented using 15 decimal digit or 15 bits then, this corresponding filters specification that it satisfy or that basically meets the specification ok. For implementation, this we need to quantize the coefficient to the word length use for the implementation ok.

So, for this quantization of this coefficient, what will be the that means, minimum number of this; that means, of word length of the coefficient to meet the proper constant on this specification of the filters that, we have to analyze using or that we can analyze using MATLAB ok.

So, in the present day signal digital signal processor this has become say less of a problem because, this 16 bits equals to 4 decimal digits or 24 bits that is equals to 7 decimal digit precision In hardware design, with the tight speed requirement, this is still a relevant problem. For this, what is basically meaning is that what will be that means, corresponding; so, 15 decimal digit means it is basically to the word length for that is basically 60 bit ok.

So to 60 bit means, in the multiple constant actually, we have taken the example of FIR filter at the time. So, we need the multiple constant, where we have to consider the multiplier bit length of 60 bit ok to meet the corresponding specification.

(Refer Slide Time: 04:53)



Coefficient Quantization

Coefficient quantization effect on pole locations :

-> tightly spaced poles (e.g. for narrow band filters) imply high sensitivity of pole locations to coefficient quantization

-> hence preference for low-order systems (parallel/cascade)

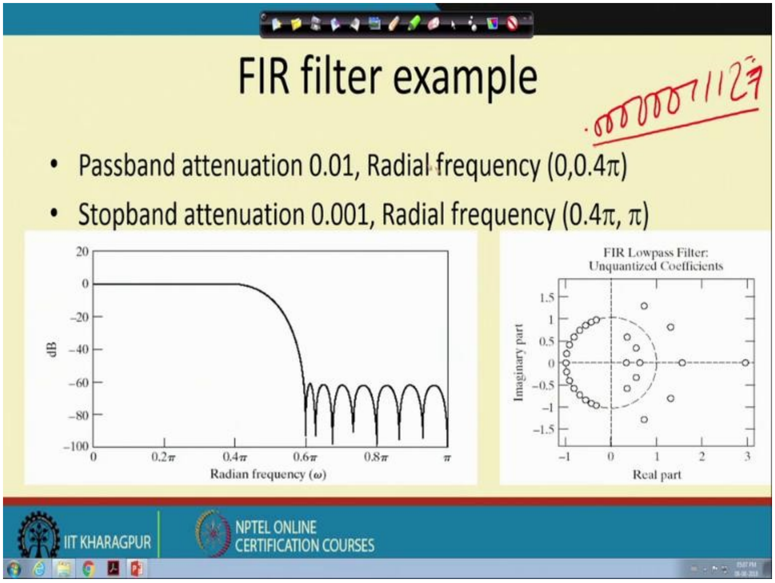
Example: Implementation of a band-pass IIR 12-order filter

Cascade structure with 16-bit coeff.

Direct form with 16-bit coeff.

So, 60 bit means it requires a lot of hardware right. So to reduce that what will be the; that means, the optimum word length that we will see.

(Refer Slide Time: 05:05)



FIR filter example

.00000071127

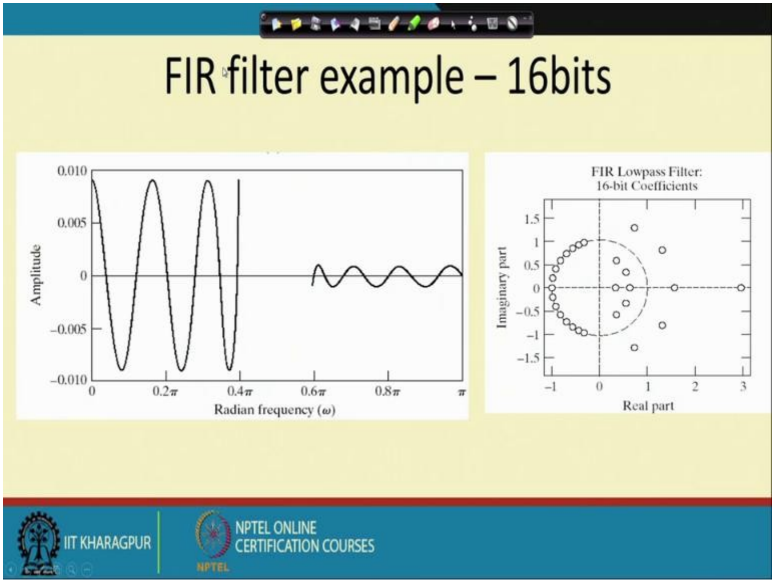
- Passband attenuation 0.01, Radial frequency (0,0.4π)
- Stopband attenuation 0.001, Radial frequency (0.4π, π)

Suppose we had that means, this not this one, suppose we have to implement one FIR filter, which has the specification of passband attenuation of 0.01 with the radial frequency of 0 to 0.4 pi and the stopband attenuation of 0.001 with radial frequency of 0.4 pi to pi ok.

So, if you just see this is the; that means, the corresponding the filter response ok, and this is the, this pole 0 plot, this is the pole 0 plot ok considering the unquantized coefficient. That means, where each of this pole and zero; that means, each of the coefficient has been considered in a on it is original value, original value means if the value is let us say, if the value is; that means, 0.0000000012 something like this. So, any of this value, if this is the real value of this coefficient if I consider then, this is the corresponding, that means, pole zero location which, I am getting this is the actual pole zero location of this particular FIR filter ok.

Now, the thing is that ok, if I take fixed point coefficient. So, at the time what will happen?

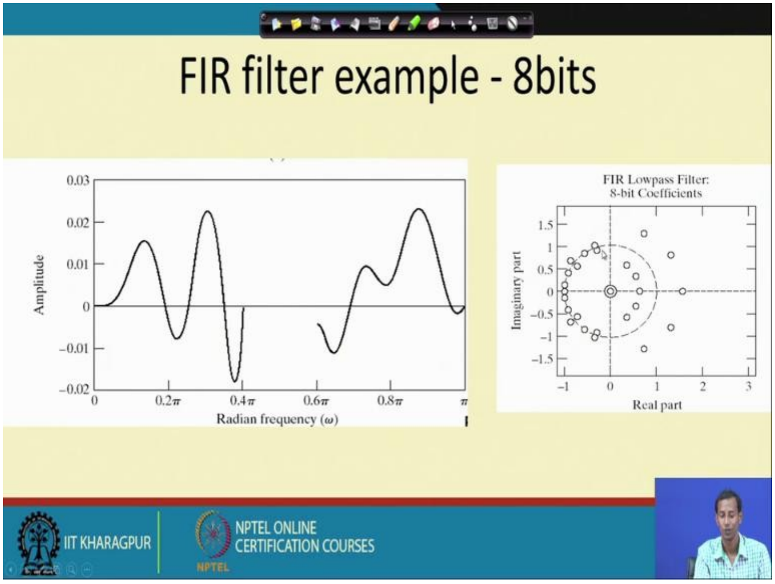
(Refer Slide Time: 06:43)



FIR filter example – 16bits

So, if the FIR filter, consider the coefficients of 16 bits. So, at the time the corresponding pole zero location will be something like this ok. So, how it is different? It is not that much different from this unquantized coefficient one; that means, it is more likely or if you just see it is more likely closer to the this unquantized one, the performance of this is not degraded if I considered the coefficients are each of 16 bits ok.

(Refer Slide Time: 07:27)



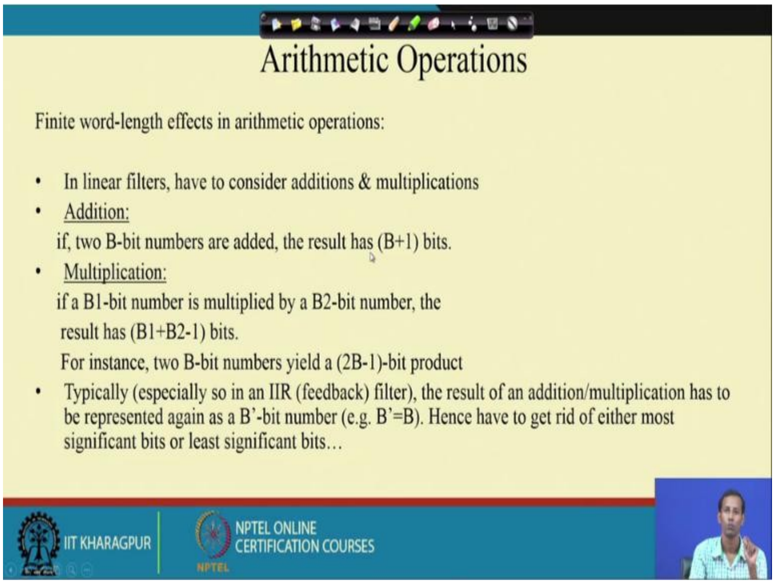
FIR filter example - 8bits

But, if you just considered that if you just considered this, coefficients of 8 bits. So, at the time you see these are basically deviating. So that means, if this is the original one ok, then this is the corresponding to the considering the fact that each of coefficients are of 16 bit, but if I consider each of 8 bit. So, at that time these basically, this zero location, they are basically deviating from the original one so; that means, the performance and it is actually, it is sometimes, it is basically going beyond of this unity circle so; that means, your filter whatever you have implemented in hardware, that may be unstable at that time.

So, it is not; that means, permissible or it is not advisable that, whatever hardware I have designed the filter hardware, if I have designed considering the fact that each of the coefficients are of 8 bit. So, at that time the filter response, may be or the performance of the filter that may be unstable at that time. So in some of the case, it may be unstable ok. So, that is not desirable. So, to avoid that we need to do this, what will be the; that means, corresponding this word length for this? So, that we have to consider ok.

So, what is the benefit here? The; that means, for if I consider each of the bits instead of the coefficients are of 8 bit; that means, the hardware for doing this, multiple constant multiplication that will be less at that time, if I consider 16 bit so at that time, I need more number of adder to do that multiple constant, but if I consider 8 bits. So at that time, I need lesser number of adder to; that means, to implement that, but I have to keep in mind that my performance is degraded that is not at all satisfying the overall, whatever is my desired; that means, specification or desired functionality of the filter, if that is not meeting. So at that time, this is not the case which I will consider ok. So, this is the fact which we have to keep; that means, analyze from the beginning; that means, from the implementation of the hardware itself ok.

(Refer Slide Time: 09:59)



Arithmetic Operations

Finite word-length effects in arithmetic operations:

- In linear filters, have to consider additions & multiplications
- Addition:
  if, two B-bit numbers are added, the result has (B+1) bits.
- Multiplication:
  if a B1-bit number is multiplied by a B2-bit number, the result has (B1+B2-1) bits.
  For instance, two B-bit numbers yield a (2B-1)-bit product
- Typically (especially so in an IIR (feedback) filter), the result of an addition/multiplication has to be represented again as a B'-bit number (e.g. B'=B). Hence have to get rid of either most significant bits or least significant bits…

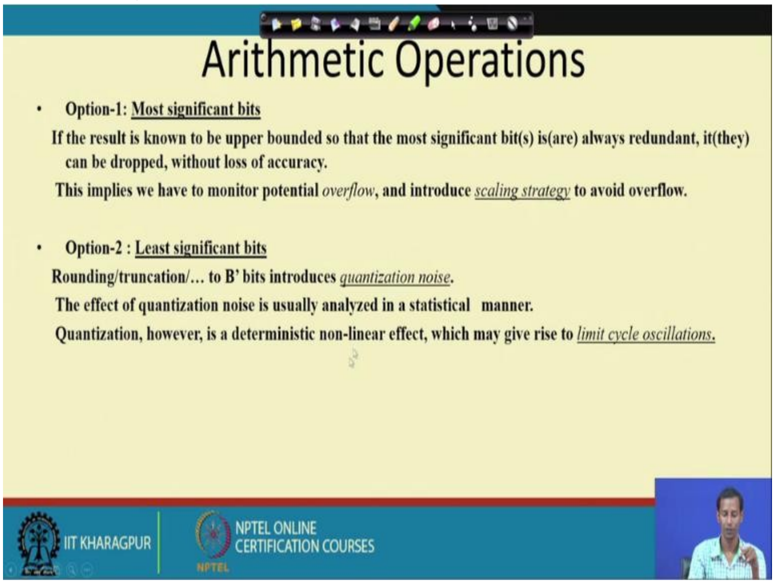
So then; that means the quantization that may happen or that is required for the arithmetic operation too. So, in the arithmetic operation for the addition operation, what I need? If 2 B bit numbers are added, then the results have to be of B plus 1 number of bits ok.

And then for the constant multiplication if a B 1 bit number is multiplied by B 2 number of bit. So, at the time the result has to be of B 1 plus B 2 minus 1 number of bits so; that means, for the multiplication the word length will be B 1 plus B 2 minus 1, for the this addition that minimum requirement of the that corresponding to hold the proper result; that means, to reduce the overflow or to avoid sorry, to avoid the overflow, we need to consider the corresponding bits for the addition B plus 1 for multiplication, that is B 1 plus B 2 minus 1 ok.

Then for typically or for this is for IIR filter oh sorry FIR filter for especially, for IIR filter the results of an addition or multiplication has to be represented again as a B number of bits. Hence to get rid of either most significant bits or less significant bits, means what? When in IIR filter, when there is feedback so; that means, for this arithmetic operation, whatever error I will gather. So, that will be run in a loop ok. So, run in a loop means as that particular point is basically affecting the overall performance of the filter.

So at the time, I have to be considered this fact or I have to be more causes, while consider in this corresponding word length for this multiplication as well as this addition operation ok. For the FIR filter, if it not that much it effects, but for IIR filter representation or IIR filter design, if this arithmetic operation truncation or the corresponding word length selection of the arithmetic operation, that basically means effects a lot ok.

(Refer Slide Time: 12:49)



Arithmetic Operations

- Option-1: Most significant bits

If the result is known to be upper bounded so that the most significant bit(s) is(are) always redundant, it(they) can be dropped, without loss of accuracy.

This implies we have to monitor potential *overflow*, and introduce *scaling strategy* to avoid overflow.

- Option-2 : Least significant bits

Rounding/truncation/… to B' bits introduces *quantization noise*.

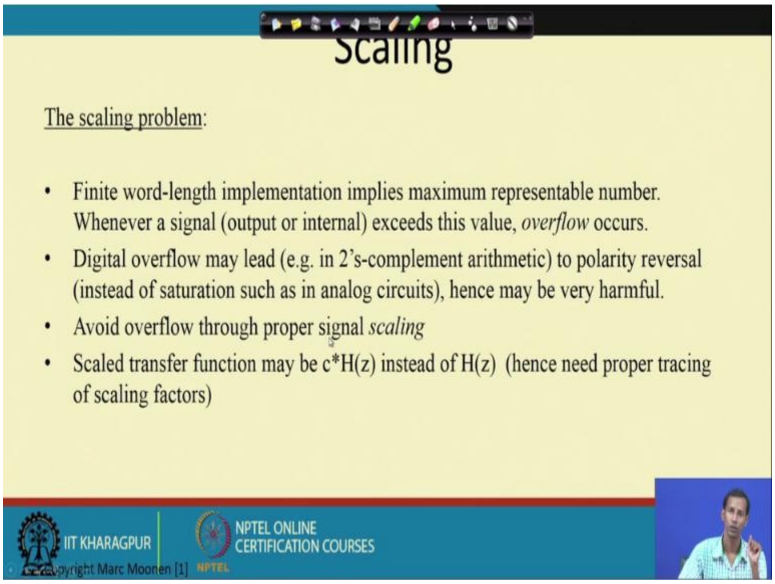
The effect of quantization noise is usually analyzed in a statistical manner.

Quantization, however, is a deterministic non-linear effect, which may give rise to *limit cycle oscillations.*

So in arithmetic operation so, we can have 2 option, one is this a for MSB, another one is for LSB ok. LSB, when I require? Basically, if I miss the MSB; that means, I can, what I said that if I suppose to hold the data or hold the result, if I represent that ok, I need only 9 bit; that means, if and if the results is more than 5 11. So at the time, the result will be represented for 5 12, the result will be 0 though the result is 5 12 at that time; that means, there will be 1 overflow, if I miss the MSB and if I miss the LSB; that means, there will be the truncation or this quantization error ok.

So; that means, quantization error means suppose, I am having this suppose, I am having 3. If the last bit, if I just missed so; that means, the data will now became 1; so that means, 2 is the error which, I am basically accumulating for just missing one of the LSB so; that means, in both the case for considering the fact of LSB and considering the fact of missing the MSB in both of the case for MSB, I am just doing, this overflow I am getting overflow and for truncation in the LSB, I am getting this quantization noise ok.

(Refer Slide Time: 14:29)



Scaling

The scaling problem:

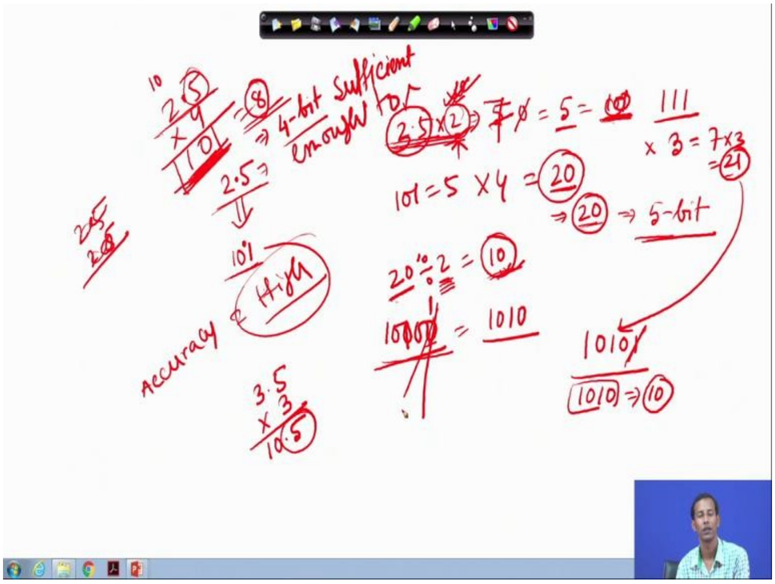
- Finite word-length implementation implies maximum representable number. Whenever a signal (output or internal) exceeds this value, *overflow* occurs.
- Digital overflow may lead (e.g. in 2's-complement arithmetic) to polarity reversal (instead of saturation such as in analog circuits), hence may be very harmful.
- Avoid overflow through proper signal *scaling*
- Scaled transfer function may be c*H(z) instead of H(z) (hence need proper tracing of scaling factors)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Copyright Marc Moonen [1]

So then, another point is that that is which is basically scaling ok. So, what is scaling? Scaling basically is needed for finite word length implementation implies maximum representable numbers, whenever a signal exceeds this value, this overflow occurs digital overflow may lead to polarity reversal, hence it may be very harmful to the circuit or to the hardware design, avoid overflow through proper signal scaling. Scaled transfer function may be as c is multiplied with the H z instead of H z, where H z is the corresponding transfer function.

(Refer Slide Time: 15:33)



So, what is the meaning of scaling then? Meaning of scaling, means suppose ok, let us considered one of the example at the time, it will be much more visible suppose, I have to do this.

Suppose, I have the number as 2.5 ok, so whenever I am having this 2.5 in fixed point, how it will be represented, if I just take the corresponding that numbers? So, the number will be represent as 1 0 and here this is 1. I am I am basically, considering this that this is the point is here, but how we convert one floating point to fixed point basically, we repeatedly we multiplied with 2. So, that is 5 means this portion is 0.

So, I will take the interior portion 5, which is 1 0 1. So here, you see the number basically represented as 1 0 1, but in actual, the point is basically at that time. So, I have just multiplied with 2; that means, the corresponding point is here, that virtually I am considering ok. So now, suppose I have to multiply this 2.5 along with this 2.5 ok. So, if I just have to let us considered 2.5 that is multiplied with 4. So, the result is 10 ok.

But here, actually the to the represent this 2.5 properly, if I just only take the integer portion. So at the time, it will be 1 0 ok. So, 1 0 means only 2. So at the time, I will get the result as 8 so; that means, 2 will be the error, which I will gather for this particular multiplication,, but if I do not require or if I do not need this particular error, which will be accumulated for truncation of this fractional part. So, I need to hold the original value. So that means, at the time 2.5 I need the value of 1 0 1 and it is multiplied with 4 means, I need the results will be 10 which I basically needs, but because of properly holding that this 2.5, I need in the hardware actually, I need the data to be represented as 1 0 1.

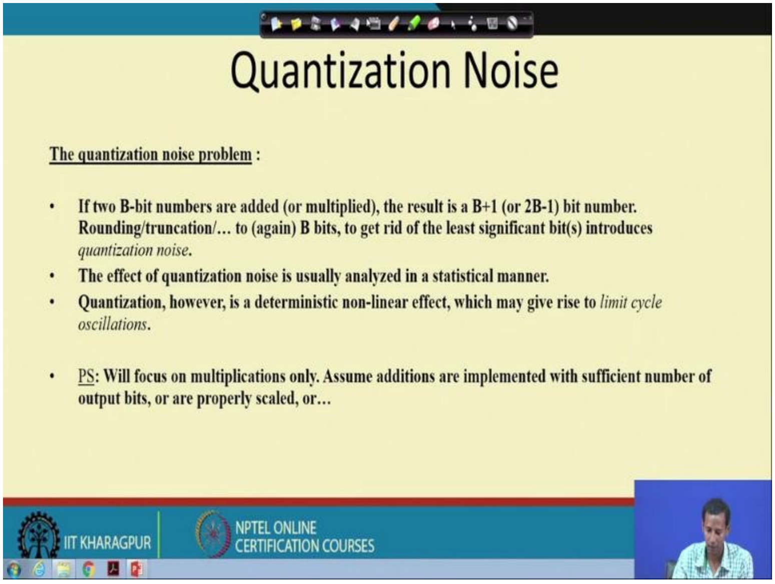
So now, 1 0 1; that means, which is 5, now at will be multiplied with 2 means the value is 20 now. Now, the thing is that to represent this 20, my original value is what? Is 10; that means, if I just let us considered that 4 bit is basically, sufficient enough sufficient enough to hold the corresponding values of 10, but in actual what I am doing? The result is 20. I am getting because of the scaling of this, what I am doing here? I am basically use the scale factor as 2 to hold the proper data of 2.5. So, the result is 20; that means, that corresponding result, I need to hold that is instead of 4 bit, I need 5 bit here, but how I will get the original values of 10?

So, initially what I did? I basically scaled up, this value by 1 bit. So now, again I have to scale down to get the proper result, I need to scale down the corresponding value. So, scale down the corresponding value means? Now, I have to divided by 2. So here, what I did? I have multiplied initially I have multiplied one of that value by 2. So here, the I have to divide by the same factor that is, I will get the result as 10; that means, here initial my scaling factor is 2 and the same factor here, I have divided to get the proper result. So here, you see the original result is 10.

So here also, I am getting the proper result which is 10. So, this is the; that means, the kind of techniques we used to get so; that means what? Here accuracy is basically high; that means I am not losing any error; that means, I am not doing any error while though, I am doing the operation. So, this is the scale factor or this is the scaling of the input variables, which basically happens in the arithmetic operation ok.

So, this is the scaling ok.

(Refer Slide Time: 21:21)



Quantization Noise

The quantization noise problem :

- If two B-bit numbers are added (or multiplied), the result is a B+1 (or 2B-1) bit number. Rounding/truncation/... to (again) B bits, to get rid of the least significant bit(s) introduces *quantization noise.*
- The effect of quantization noise is usually analyzed in a statistical manner.
- Quantization, however, is a deterministic non-linear effect, which may give rise to *limit cycle oscillations.*
- PS: Will focus on multiplications only. Assume additions are implemented with sufficient number of output bits, or are properly scaled, or...

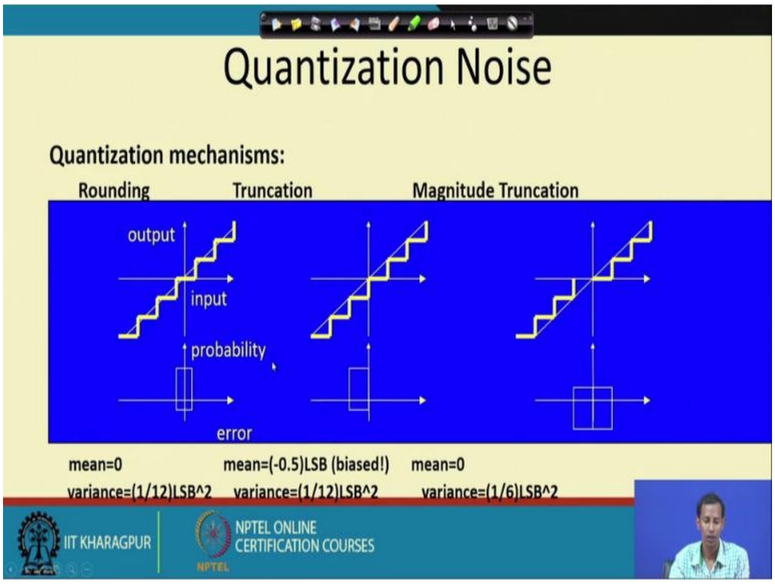
Then, this if you have two B bit numbers has to be added then that result what I said the result has to be this, B plus 1 bit number ok. And for this truncation and rounding or rounding or truncation actually by B bits, this quantization noise is basically introduced ok. So, in this particular example, I will show you, how we can basically get the; that means, error where I will get the error. So, whenever we are dividing by 2; that means I am doing what? So, 1 0 0 means?

So, 20 means, 10100. So, divide by 2 means I am just discarding the LSB. So, I am getting the results as 10, if this is not 0, if this value is 1, let say their result is. So here, if it is 3 3.5 means, this is 7. So, 7 means it will be 1 1 1. Now I have to multiplied, this value with let us considered 3. So 7 into 3, the results will be 21 right. So, 21 means the corresponding representation will be this right.

So here, if I just discard the LSB, what will be the results? That will be 10 right, but in actual, what will be the results? That is 3.5 multiplied with 3. So, this is 10.5; that means, because of this truncation of this LSB, I am basically gathering the error of 0.5 over here ok, but if I just consider at the time something like this. So, I will not get it; that means, for if I, while we are doing this scaling up. So, at the time if I get whatever operation, I am getting at that time, if I am getting the number as word and then I am doing the truncation.

So; obviously, I will accumulate the error, if this is the results is even and then we are doing the LSB means what at the 20, if this 0 has been discarded? That means, 0 discarded means, there is no chance of doing any or accumulating any error as this portion is 1 so; that means, I am accumulating the error ok. So, this position 1 means the number is basically even, sorry odd. So, odd number then doing truncation obvious surely your getting the; that means, your accumulating the quantization noise or the truncation error you have basically doing ok.

(Refer Slide Time: 25:09)



Quantization Noise

Quantization mechanisms:

Rounding Truncation Magnitude Truncation

mean=0 mean=(-0.5)LSB (biased!) mean=0

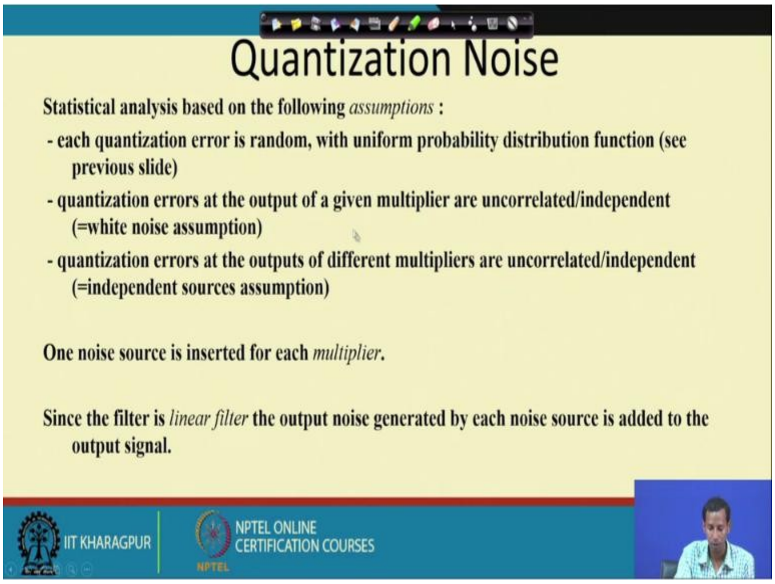
variance=(1/12)LSB^2 variance=(1/12)LSB^2 variance=(1/6)LSB^2

So, this is the example of this scaling and now this quantization that may be have deterministic non-linear effect, which may give rise to this limit cycle oscillation ok

So, these are the facts now, we will considered that what is rounding? What is truncation? And what is magnitude truncation? Ok, so; that means here, if you just see suppose rounding means if the, actually the original value is something like like this so; that means, this may now it may get or this value that may be get just lower or upper of this, but in truncation always the it will be whatever is the original; that means the value, it will be just below 1, what I why it is that? Because truncation means, what I am just discarding in rounding means, we are just getting the nearest value of it ok.

So, that is why truncation, in truncation it will be always the below of this particular; that means the particular line. In magnitude truncation, so based on that, we can take the decision. So, there is the corresponding, that error curve; that means, probability of error that may occur within the range. So, in this truncation the error is this. So, magnitude truncation the error may be this ok.

(Refer Slide Time: 26:37)



Quantization Noise

Statistical analysis based on the following *assumptions* :

- each quantization error is random, with uniform probability distribution function (see previous slide)
- quantization errors at the output of a given multiplier are uncorrelated/independent (=white noise assumption)
- quantization errors at the outputs of different multipliers are uncorrelated/independent (=independent sources assumption)

One noise source is inserted for each *multiplier*.

Since the filter is *linear filter* the output noise generated by each noise source is added to the output signal.

So then, this quantization noise, what I said that in the previous slide, this quantization effect that may be you can calculate. So here, there is a procedure of doing that that is this statistical analysis based assumption, you can do for this quantization noise analysis ok. So, in this statistical analysis based noise analysis; that means, this quantization noise analysis is quantization error is random with uniform probability distribution function ok.

So, if you just see the previous slides then a quantization error at the output of a given multiplier are uncorrelated and independent quantization error at the output of the different multipliers are uncorrelated an independent and one noises source is intersected for each of the multiplier, since the filter is linear, filter the output noise generated by each of the noise source is added to the output signal means what? Basically, whenever we will calculate the overall transfer function.

So at the time, each of this basically means what? Actually the point is that whatever analysis we have doing. So, that is based on the overall, this quantization of the considering, the fact of quantization in the coefficient as well as quantization in the arithmetic operation, whether that is of a multiplier or that is of adder for both ok. So, considering that fact, we are basically getting this particular curve or getting this particular graph ok. So, this is the; that means, this is the way of doing this.

So; that means, in fixed point number infiltration, what are the; that means, terms which are associated? That is overflow, if we are discard the MSB at the time, we will get overflow error; that means, we will get the magnitude of the red; that means, results in a wrong way, if you discard the LSB at the time, we will get quantization error and for that, I need to find out what will be the optimum word length? Keeping in the mind that, the performance and the cost both it would be optimal point for what of word length? It is the maximize maximizing in both the case. So, that is the optimum word length for that. So, while we are doing this floating point to fixed point conversion at that time, we have to keep remember of this things ok.

So, and more of the analysis, we can do using MATLAB or not MATLAB in SPW or in CO Centric. So, these are the tool set, where you can analyze your algorithms. What will be the effect for floating point to fixed point conversion? So if you need actually, this is not this is just to give the idea about, what is happening and how basically you can calculate? So, there is more on to this so; that means, you have to do a rigorous analysis, how we can do? So, whenever you are basically doing your project or you doing your work.

So at that time, how you have to do that? There is a method. So for that, you can go for this the help, which is available with this tool particular, tool set or any material which is available on the internet. So that, you can get or else what we can do, if you more interested, how to do that? Then please let me know via discussion forum. So, you just let me know by asking that ok, I need to more on this for my purpose for my project work or for my work purpose. So, any purpose if you just interested more to this is just the basic idea, but there is more of for each of this point, there is more if you want know more then, please let me know, I will try my level best to help on that particular regards by providing, any tutorials or any other documents if I am having it. So, surely I will try to share with you. So, this is for today.

Thank you for this today’s class.