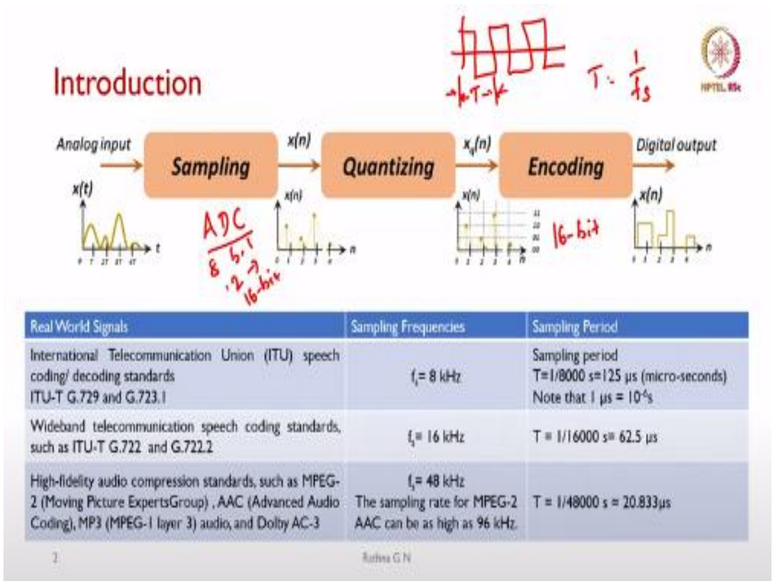
Real-Time Digital Signal Processing
Prof. Rathna G.N.
Department of Electrical Engineering
Indian Institute of Science, Bengaluru

Module No # 12
Lecture No # 56
M3u33-Summary of the Course

Welcome back to real time digital signal processing course so hope you have enjoyed the course today we will look at the summary of the course.

(Refer Slide Time: 00:33)



Real World Signals	Sampling Frequencies	Sampling Period
International Telecommunication Union (ITU) speech coding/ decoding standards ITU-T G.729 and G.723.1	f_s= 8 kHz	Sampling period T=1/8000 s=125 μs (micro-seconds) Note that 1 μs = 10^{-6}s
Wideband telecommunication speech coding standards, such as ITU-T G.722 and G.722.2	f_s= 16 kHz	T = 1/16000 s= 62.5 μs
High-fidelity audio compression standards, such as MPEG-2 (Moving Picture ExpertsGroup) , AAC (Advanced Audio Coding), MP3 (MPEG-1 layer 3) audio, and Dolby AC-3	f_s= 48 kHz The sampling rate for MPEG-2 AAC can be as high as 96 kHz.	T = 1/48000 s = 20.833μs

2 Rathna G N

We started the course with introduction so we know in real world all the input is in analog domain so that is what we have seen here is a time domain signal what we are taking it. And then we are going to do the sampling using our ADC basically, so ADC can be either 8 bit or 12 bit or depending on the latest 1 sigma delta ADC'S we can go up to 16 bit ADC conversion. Then we will be converting it into a digital signal $x(n)$ o

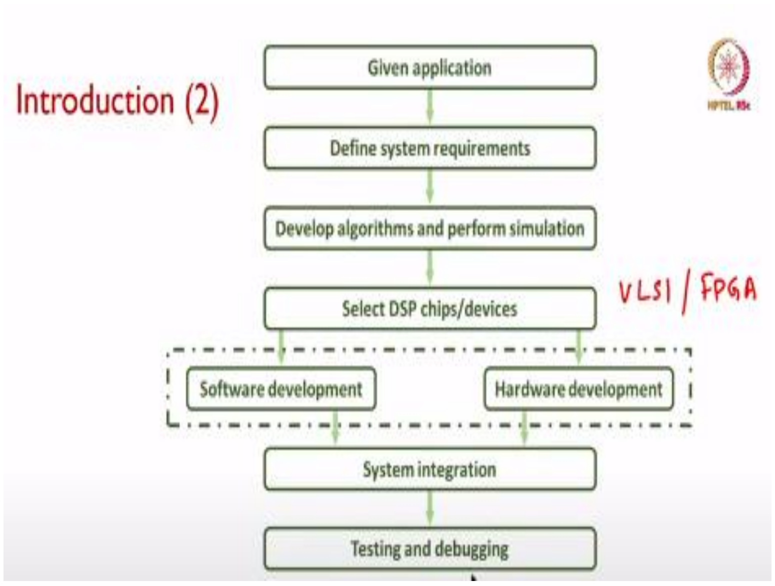
So then we are going to do the quantization because we know number of bits what we want to represent it. So in our case we have taken it as 16 bit basically in all our processor applications so we can convert it into 16 bit. And then do the quantization with respect to that and then we have to do encoding of this quantize signal into the number representation. So that is how we will get the digital output, but this constraint what we said was in the real time signal processing.

So if I take my clock as this way so we know that this is my time period what we are discussing about it or $t = 1/f_s$ is my sampling frequency. So the First different sampling frequencies what should be a real time constraint one has to look at it, so the bottom table what it gives is different applications what we have in the real world basically. One is the speech coding or decoding one can use the ITU standard basically G.729 or 723 and the, which requires a sampling frequency of 8 kilohertz.

So we must be able to complete both, decoding or encoding within this time period. So and the time second in that is what we give it as T = 125 microsecond. And in the case of wideband telecommunication speech coding which uses g.722 or 72.2, so we know that the sampling frequency is 16 kilohertz. So our f s has to be 16 kilohertz and we must be able to finish our computation time in 62.5 microseconds time period.

And when you go for the high fidelity audience compression like; mpeg2 or your AAC standard or MP3 standards or DOLLBY, so you know that they have operated 48 kilohertz. So within 20.833 micro second I must be able to finish my sampling quantization encoding and even the computation of my signals and even outputting which has to be carried out, so that is what our real time constraint what we have put it.

(Refer Slide Time: 03:46)

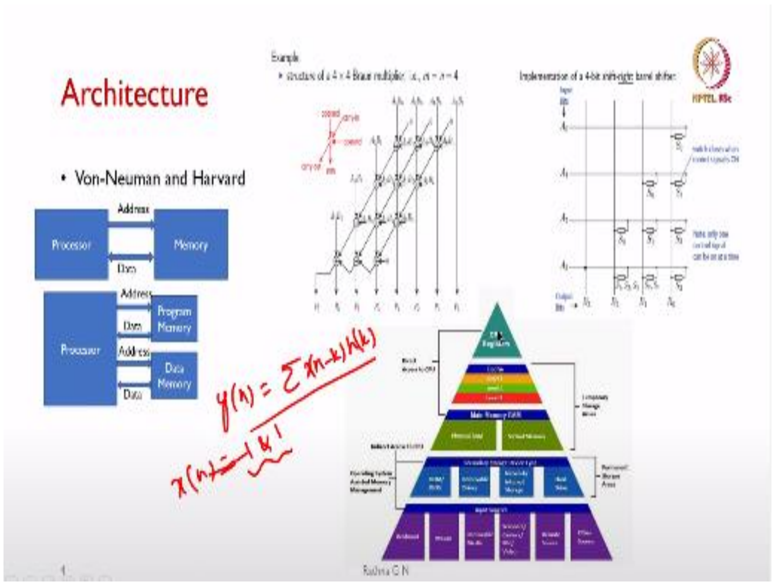


So the other way of looking at is it depends on application what we are choosing; so here only three what it is listed so we have umpteen number of applications. So we have to define our system requirements in this case and develop the algorithms and perform simulation. And then if we are using the DSP processor which DSP chip we have to select it we have to select it, or the devices. So if you are having your own design either you can do your design in VLSI or in FPGA.

So most of the application we want to try it on field programmable gate arrays and if we have a requirement of many numbers basically then we can go in for the VLSI design. So the other the two legs what we will be taking is software development is going to happen and then hardware development. So now it is it is going to be what we call it as hardware and software code designed together they have to go hand in hand so that both the things match simultaneously.

Otherwise all of us know that hardware multiple units come out and then we may not be accessing most of it because software is the must, most costlier to develop basically. So then usually we combine these to do the system integration and then check if it is going to satisfy our requirement or we may have to go back and select different algorithms for this application. Then once everything is finalized then we will be doing the testing and debugging. So once the product is ready you can give it in the market.

(Refer Slide Time: 05:43)



So different architectures what we went through in this course so the first one was all of us know Von-Neuman architecture so we have the processor and then memory is single memory. So both address and data resides in the same memory and then we will be using the address and database to access this. So for our DSP applications most of the applications we set that it is of this nature that is $y(n) = \sigma$. So $x(n-k)h(k)$ so this is our most of the applications what we said we need multiplication and then addition simultaneously.

And then we have to access two inputs at a time and then we may have to write one output. So for that reason so von-Neuman architecture is not sufficient so we said Harvard architecture was developed for this purpose. So what is this it has separate program memory and then data memory and even address and then data base for them is separate. So in spite of developing the thing so will be instruction fetch we can get it from the program memory but data still we have one data memory.

So if we keep on adding now the hardware architecture is going to increase. So what it was decided because most of the if we are talking about the filter then we know that filter is designed much earlier in real time we will be using only those filter coefficients to pass it through our input. So this one can be stored in our program memory and both and then input data is going to come into the data memory and then we can access them simultaneously.

And then now we can one more multiplication what we said was we will go with the Braun multiplier so which is a much faster. So we said that these are the two inputs for four bit what we drew are Braun multiplier. So in this case we are going to have an, and gate before the data comes into our Braun multiplier. So as you will be seeing that the longest part delay in this is going to be as you are seeing that from this path to the final output.

So to reduce this also we can have pipelining in between stages, so that we looked into pipeline architecture later on. So and we know that we need a shifting operations because we said that the input is going to be represented that is my x of n is the input which is going to be in between 1 and - 1 and then 1 this is the range what I will be considering it. So what we have to do is it has to be shifted on the go, so most of the shifting operation we can use the barrel shifter.

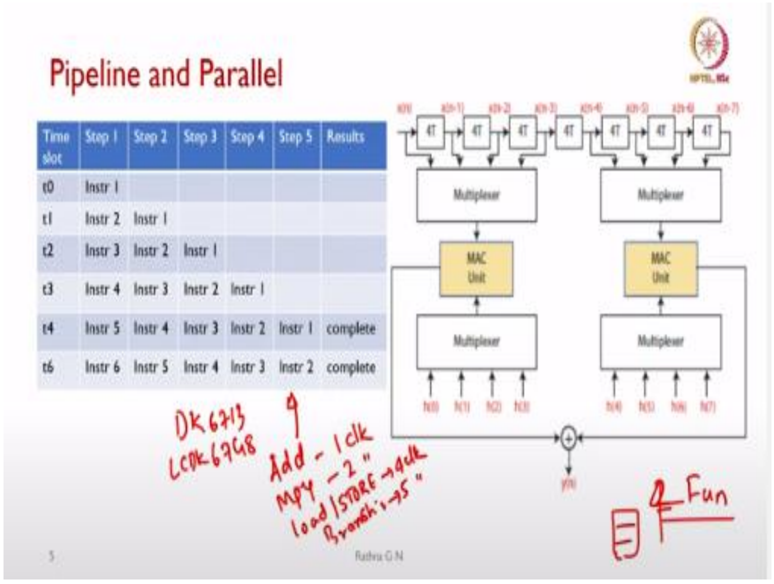
Either we can do a left shift or right shift even the output is exceeding certain limit then we have to do shifting of the thing before we feeding it to the next stages. So we can use the hardware barrel shifter, so which is going to do the shifting operation in 1 clock cycle. So if I want to shift 5 bits basically in normal cases in the sequential it will take 4, 5 clock cycles whereas using the barrel shifter I can finish it in 1 clock cycle, so that is what we saw the thing.

Next is coming to the memory architecture, so how we are going to have the memory because we as far as possible we want to have a latency of accessing our data writing into the memory, or reading from the memory, storing has to be minimized. But the architecture defines that we cannot have whatever our wish so based on the things of what we have is we will be having along with the CPU some of the registers, will be having it.

So that the fastest one as you can look at it that is what we discussed it and we will have some this thing l 1 and l 2 memory which are little away from the CPU but they are much closer so their access is minimized. And then we have larger memories down the lane, so this triangle gives you how the access time is going to be. So the least one you will be seeing that are keyboard, mouse, and other things here even some of the tape readers what we had it earlier.

And then we will be seeing that ROM and then our removal devices and other things will be much slower compared to this; this is how the memory hierarchy we went through in the class.

(Refer Slide Time: 11:12)



Pipeline and Parallel

Time slot	Step 1	Step 2	Step 3	Step 4	Step 5	Results
t0	Instr 1					
t1	Instr 2	Instr 1				
t2	Instr 3	Instr 2	Instr 1			
t3	Instr 4	Instr 3	Instr 2	Instr 1		
t4	Instr 5	Instr 4	Instr 3	Instr 2	Instr 1	complete
t6	Instr 6	Instr 5	Instr 4	Instr 3	Instr 2	complete

DK6713
LCDK6748

9
Add - 1clk
MPY - 2 "
load/STORE → 4clk
Branch → 5 "

5
Rathna G N

So then we said that for increasing the speed of our computation we can go with pipelining and parallel architecture. So our intention was to get within 1 clock cycle we have to get the, whatever the input clock our output should be available in the same clock cycle. So every clock cycle I want to have the output whatever my computation is going to be, so one of the way of looking at is we can have a instruction pipeline.

So most of the DSP processors have 5 stages of pipeline so that when the first stage is instruction one what it is happening. So later on when instruction one moves to step 2 basically so the next instruction can be fetched. So when instruction one is getting decoded in step 2. So in the next stage you are going to access your data basically that is instruction one moves to get the data in the third clock cycle. And then you will be passing the decoding of instruction two is happening and then you are trying to fetch instruction 3.

And then in step 4 and 5 I can have the execution, so most of the time it is step 5 is our execution can be from 1 clock cycle to multiple clock cycles it may take depending on the hardware what we are going to use it. So most of the DSP processor we know that our add is going to be in 1 clock cycle what I can fetch the I can complete my execution whereas my multiply MPY operation is going to take 2 clock cycles.

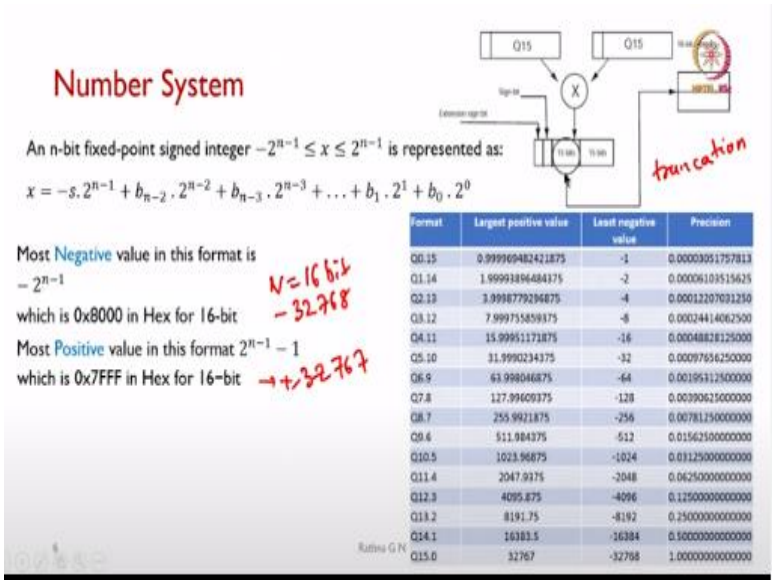
Whereas my, this thing what is it we have a load and then store architecture what we said it is going to take 4 clock cycles. And branches in all this DSP pre-processor 6, 7 what I have considered that is DSK or 6713 and then 6748 LCDK board from the TI they take branching of 5 clock cycles. So these are the computation that is execution times what we need it in this. So these are basically a load and clock load and store architecture what we call it so we know how much clock cycle each instruction is going to take.

Some of the instruction I will come in a while we may not know how much clock cycles then we called them as this thing complex instruction cycle basically you would have heard of risk and sisk. So DSP processor most of the time 80% it is going to use is a risk architecture and 20% for our circular buffering and then what we call it as FFT operation we have to bit reversal has to happen, they use sisk architecture for that.

And this is pipelining what we call it and then m the parallel operation so we said that our DSP processor has a, 2 functional units in them. So parallely I can do my operations that, is this has four functional blocks within each side there are two sides and then four functional blocks in each side. So if I operate them in parallel so 8 functions what I can operate in parallel and each is 32 bit. So we have a data width of 256 bits for this to get the complete instruction in one clock cycle.

So we said that when we do 2 this, thing sides of it we can do them in parallel. So how the delay is going to be here if it was taking for 8 clock cycles here you will be seeing that within 4 t what I can finish one side of it the other side. And then finally the output from both the side I have to add them up and give my y of n output. Although we say that 2 units are there I should be completing within half the clock cycle of the original one but you will be seeing that there will be some delay so we will not be getting exactly that computation done in half the clock cycle.

(Refer Slide Time: 16:24)



Q15
Q15
X
truncation

Number System

An n-bit fixed-point signed integer $-2^{n-1} \leq x \leq 2^{n-1}$ is represented as:

$x = -s.2^{n-1} + b_{n-2}.2^{n-2} + b_{n-3}.2^{n-3} + \ldots + b_1.2^1 + b_0.2^0$

Most Negative value in this format is -2^{n-1} which is 0x8000 in Hex for 16-bit

N=16 bit
-32768

Most Positive value in this format $2^{n-1} - 1$ which is 0x7FFF in Hex for 16-bit → +32767

Format	Largest positive value	Least negative value	Precision
Q0.15	0.999969482421875	-1	0.00003051757813
Q1.14	1.99993896484375	-2	0.00006103515625
Q2.13	3.9998779296875	-4	0.00012207031250
Q3.12	7.999755859375	-8	0.00024414062500
Q4.11	15.99951171875	-16	0.00048828125000
Q5.10	31.9990234375	-32	0.00097656250000
Q6.9	63.998046875	-64	0.00195312500000
Q7.8	127.99609375	-128	0.00390625000000
Q8.7	255.9921875	-256	0.00781250000000
Q9.6	511.984375	-512	0.01562500000000
Q10.5	1023.96875	-1024	0.03125000000000
Q11.4	2047.9375	-2048	0.06250000000000
Q12.3	4095.875	-4096	0.12500000000000
Q13.2	8191.75	-8192	0.25000000000000
Q14.1	16383.5	-16384	0.50000000000000
Q15.0	32767	-32768	1.00000000000000

So the next one was our number system what we discussed that is we said that we can represent floating point and then fixed point number system. So all of you know about the floating point so just I will put one of the slides with respect to that. So what was the fixed point signed integer representation we said the number can be represented between $-2^{n-1} \leq x \leq 2^{n-1}$.

And how it is going to represent it so we are will be considering that is $-s.\ 2^{n-1}$ and we are not giving any bit for our binary point representation. So we will be considering $b_{n-2}\ .\ 2^{n-2} + b_{n-3}\ .\ 2^{n-3} + \ldots + b_1\ .\ 2^1 + b_0\ .\ 2^0$. The most negative value what I can represent with this number system is -2^{n-1}. So in because we have considered n = 16 bit in our processor.

So it the most negative value what I can represent with representation is 0 x 800 in hex basically or we call it as - 32768 is the value in decimal what I can represent. And the most positive value in this format is $2^{n-1} - 1$ what I can have although we put - 1 to 1. So I would not be able to represent + 1 in this format so the maximum what I can represent is 0 x7 FFF in hex so which comes out to be + 32767 is the maximum value of what I can represent with this number system.

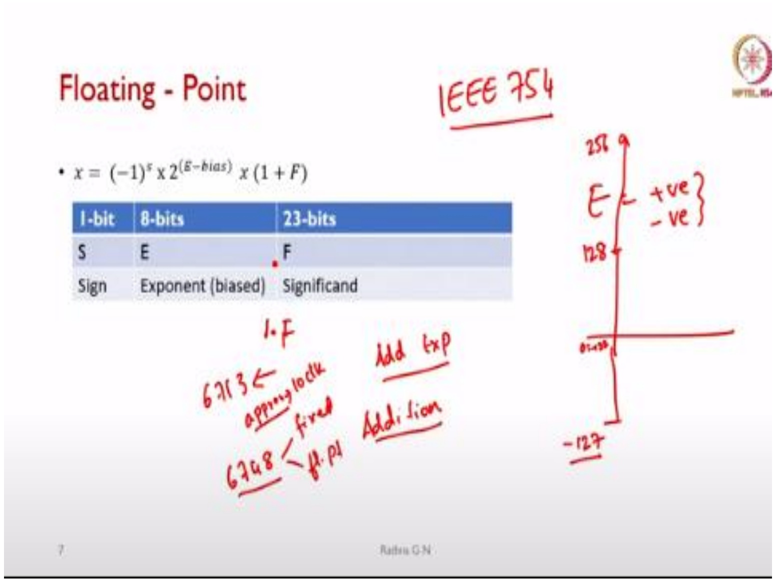
So you are seeing that a different format what we considered in our representation that is q15 format means that it is 0.15. And the largest value as you can see the thing and the least negative value is – 1, and the precision what I am going to get it with respect to this number system is the as it is given. So like that we can go last one is the integer representation that is a q 15.0 format or q 0 format what we call it so in that case the number in decimal what I will be representing is 32767 to – 32768.

So the precision is going to be 1 between the 2 bits that is LSB bit 10 the next bit that what I will have precision is 1. So as we know that we need high precision for most of our speech or audio processing so q15 format what will be using get in our application. And we know that the multiplication is going to have what is it 30 bits it represents because both are in q15 format both inputs that is 16 bit memory what I have taken the thing.

So when I multiply it so we will be getting 2 sign bits basically from both the thing depending on if both are positive these two will be positives. So if one is negative one is positive so it is going to be both are going to be negative. And then both are negative then will be having positive in the sign bit representation so we call this as a extension of the sign bit. And we will be discarding this and then we will be taking the 15 bits that is MSB bits from our output and will be storing it in the memory, this we call it as truncation when I am doing it.

So if I know that my most of the values are in this higher 16 bits then I will say that I can do the truncation. But sometimes what happens is a rounding may be required because I may have most of them are 1 in the LSB bits. Then what I consider my 14 bit if it is 1 then I can add 1 to this and then store the number in again back in the q15 format. So in that case I will be accounting for additional computation so which we have to pick otherwise the simplest one is doing the truncation; so proper care is taken so that the results are within this higher bits.

(Refer Slide Time: 21:37)



Floating - Point

- $x = (-1)^s \text{ x } 2^{(E-bias)} \text{ x } (1+F)$

1-bit	8-bits	23-bits
S	E	F
Sign	Exponent (biased)	Significand

So then we went on to compare with the floating point number system so we said that will be using the IEEE. Basically IEEE 754 format which is the standard one everybody uses so the 32-bit representation what it is shown here. So you will be having one sign bit and then exponents are represented with the 8 bits we said it is a biased what I will be representing them with, and then the fractional is 23 bits.

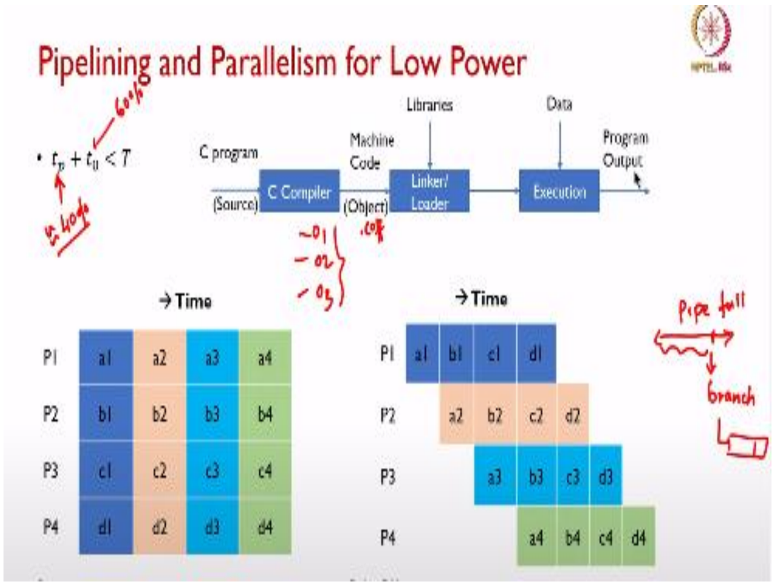
And then we know that there is a binary point here for our fractional representation which does not require any bits for it. And what is the representation it is $(-1)^s$ x $2^{(E-bias)}$ $x\,(1+F)$ because we assume that there is a one value before our point and then I will be representing this 23 bits for the fractional representation. So we call this is a bias so that is what happens to my negative numbers I can have exponent both positive then e can be both positive and then negative.

So how I will be representing it so that is how we have the bios what we have given the thing so what we have is this is – 127 to + 128 is the number what I can represent with 8 bits in sign format. So what we do is instead of this representation so we shift this one to 0 here which is equivalent to - 127 and this one will be shifting up to 256. So that is how we will be having exponent - buyers whatever x is 7 bit what I have given the thing, so which will be computing it.

And we went on to discuss about some of the problems both in a fixed point multiplication addition and then the floating point addition and multiplication how we have considered it. So most of the architecture as we know that multiplication is going to be multiply the fractional bit since floating point and add the exponent basically, or to get the result. So but whereas addition if you are doing the; addition of two numbers then you have to bring them into the same exponent and then add them up.

So what we say is in 6713 is a floating point processor so which takes approximately if it has to do maximum adjustment it will take 10 clock cycle to 2 addition, as you can see in the fixed point processors it takes only one or 2 clock cycles. So 1 clock cycle to do the addition and 2 clock cycle to do the multiplication. So whereas 6713 is a floating point process 6748 is a dual processor, that is I can do either fixed point operations or floating point operations, both can be incorporated in the thing if we are writing our code in assembly.

(Refer Slide Time: 25:13)



Pipelining and Parallelism for Low Power

- $t_p + t_0 < T$

	→ Time			
P1	a1	a2	a3	a4
P2	b1	b2	b3	b4
P3	c1	c2	c3	c4
P4	d1	d2	d3	d4

	→ Time						
P1	a1	b1	c1	d1			
P2		a2	b2	c2	d2		
P3			a3	b3	c3	d3	
P4				a4	b4	c4	d4

So coming to the thing we said that why we need pipelining and parallelism although I have we considered it for a low power applications basically. So we know that our typical computation time what we said was this is my t_p is my computation time and then t_0 is for whatever IO operations what I have it together it should be less than or equal to t. So most of the applications we say; t_p has to be approximately what we call it as a 40% only the time what I will be getting it.

So the rest of the things goes in getting the data in and then adjusting it and then sending the data out. So some cases if you are completely knowing about this operation we can make it as 50% and 50% time what is allocated for your computation as well as your IO operations. So then we said this is the parallelism if there are no dependencies then I can use the parallelism. And then if this thing we took an example of water pipe how the water is going to flow, so we will be seeing that the pipelining architecture is going to go in this way.

So what are the pipelining hazards basically so if the complete pipe is full, what we call it if it is full. And at the starting itself I get a branch basically, so if the code is not properly written if I have a branch lot of branching in between. So then what happens, whatever which has come into the pipe which has to be flushed. And then you at the branching place again you have to put your pipe and then you have to go with the instructions which are following it.

And if you it so happens that if again if it happens after one this thing you are going to branch it then you will be seeing that whatever you have designed for getting the speed you may not achieve it may work at a lesser speed than in the sequential code basically. So one has to take care such that the branches are avoided as far as possible or usually what we do in the assembly program is we do the delayed of it.

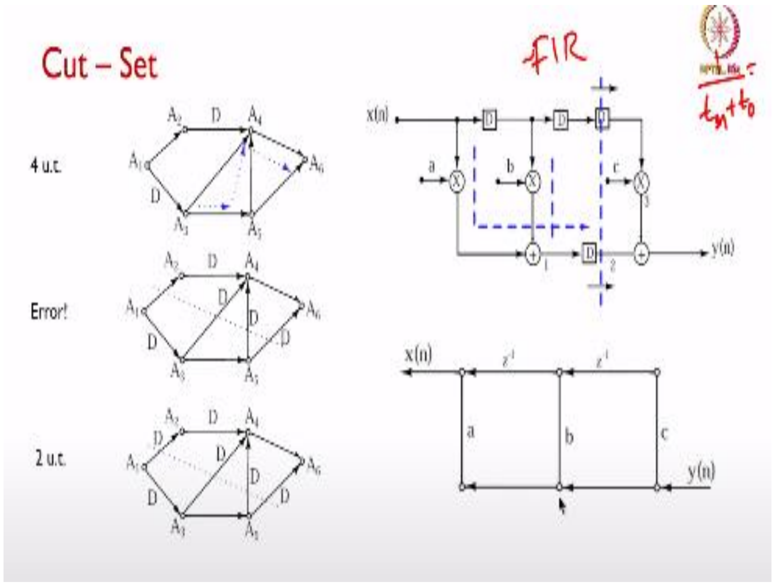
That is will have the branch here but we will be putting some of the instructions which can be operated because as I was mentioning it may take branching takes 5 clock cycles. Before that if there are no dependencies I can operate those clock cycles. So I will be saving those 5 clock cycles by the time the branch is going to happen so that my flushing of the pipe is going to be reduced.

So and then how we are going to write our code in basically so most of the thing we said we will have it c programming most of our applications what we discussed was both in MatLab and then in the DSP processor only we concentrated on the c programming we did not have time to do assembly programming those who are interested you can go through the textbooks and then see and then work on the assembly programming.

So we have the compiler there and we have the compiler options also for branching and other things how we can minimize optimization one can do it. Up to 3 optimizations you can have it so they give it as O1, O2, and then O3 basically so for different loop optimization we can incorporate and then use it. So we generate the machine code here in our case we said it is a COFF file what we are going to generate it for our boards.

And then we will be linking with the libraries, and then loading onto the board, and then we will be doing the execution, data input data is going to be given from codec if it is we are considering the real time. If it is not real time it can be stored in a memory and then it can be used. And then if we are working on the real time thing output goes out of the board or you can store it in the memory for further usage.

(Refer Slide Time: 30:30)



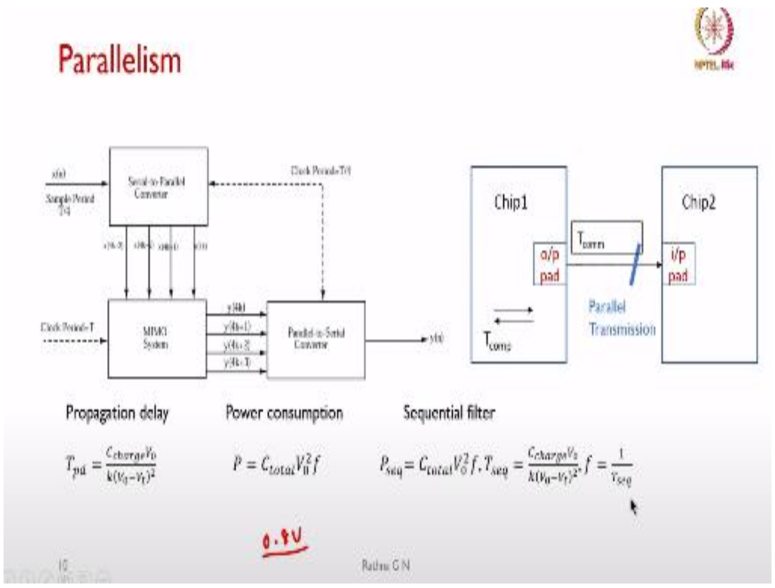
So then we said that how we can use this pipelining and then parallelism for the low power. So for that we discussed about little on the concept, how we are going to have the pipelining architecture developed from the existing one. So we said that the longest path delay is the time taken to compute our output. So in this case it is going to take four unit of time so what we are going to do is I am going to put a cut set so we said that it is only in the feed forward path what we had the cut sets.

And then as it shows there is an error so we have to put a delay line or a buffer what we are going to put it since you are cutting this line also, so you have to put a delay. So all the lines wherever you have cut the thing you will be having the delay. So that is how in the FIR filter this is what the example what we took it so which was taking 1 multiplication and then 2 clock cycles for our addition.

So; how we can reduce 1 clock cycle for multiplication and 1 clock cycle for addition by putting the feed forward cut set it here and introducing the delay. So what is the disadvantage of it is will be adding the hardware to this structure and the First output is going to be delayed by 1 clock cycle in this case and later on we will be getting every one by I will put it as T multiply + t_0 I will be getting my output from the structure.

So we said that can we avoid this cut set by doing modification to our structure itself we said that transpose the filter structure. So this is how we did the filter transpose structure that is input and output were reversed and even all the whatever you had the feed forward one you have reversed them. So by doing that so we call this is a broadcast structure so $x(n)$ is broadcast to all the legs basically and you will be having the delay in it and then you will be multiplying. In this case each this thing node you will be seeing the delay is 1 multiplication and 1 addition what you will be getting the output.

(Refer Slide Time: 32:47)



Parallelism

Propagation delay

$$T_{pd} = \frac{C_{charge}V_0}{k(V_0-V_t)^2}$$

Power consumption

$$P = C_{total}V_0^2 f$$

Sequential filter

$$P_{seq} = C_{total}V_0^2 f, T_{seq} = \frac{C_{charge}V_0}{k(V_0-V_t)^2}, f = \frac{1}{T_{seq}}$$

So then we went on to say why pipelining there are restriction to the thing because as I have already said because of the pipeline hazard so we may not have more than 3 to 4 cut set at the most. Most of the time we will have only 2 to 3 that is a maximum, not even the fourth one okay. So beyond that how we can want to have what is its speed or in the case of you are considering the computation time or from the power point of view both the pipeline or parallelism is going to reduce this the power consumption.

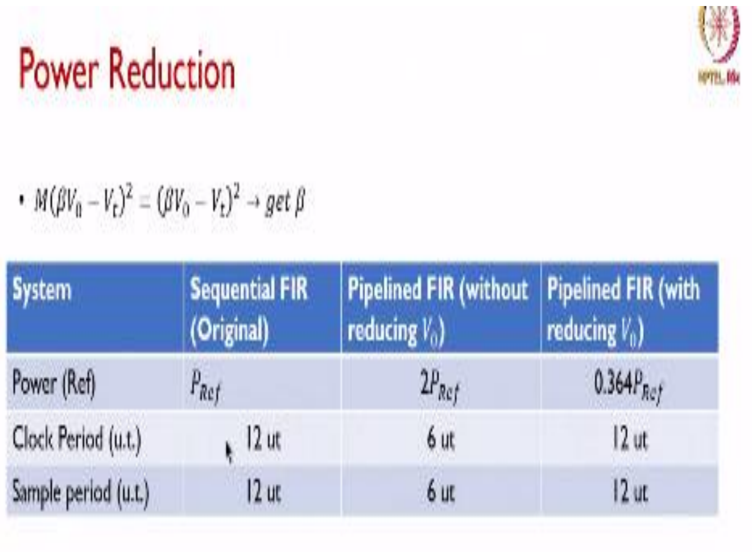
So you will be seeing that how we can convert from serial to parallel structure and then parallel to serial, so we said the critical path in this case is the communication. If communication between the two units becomes much more than the computation it is better to go for parallel architecture, so that is the restriction. So what is the power basically this is the propagation delay what we have put it.

So which is going to have the charge capacitors and our, the input voltage is v naught, and then this is directly proportional so we are putting the constant k in the, this thing here basically, and then we have v t is the threshold. Above which our transistors are going to switch on as we know that most of the TTL transistors you know that 0.8 volts is the cutoff point above that what you will be taking it as on and then below that is going to be off.

And then if they are operating at 5 volts so the power consumption is as you can see it is directionally proportional to the capacitors that are being used in the computation. And you will be seeing that it is the one which is proportional to square of your input voltage. So whether we can operate it sub optimal voltage because which is the dominant we will call it as the unit in our power consumption, so we can reduce our power quite considerably.

And it is directly proportional to frequency so if I want to operate at higher frequencies we will be going with parallelism or pipelining. But if I want to operate at the same frequency but at reduced power consumption then I can go with red optimal what is it input voltage what we can operate. So the equation what it was given for the sequential one is given by this equation that is $\frac{C_{charge}V_0}{k(V_0-V_t)^2}$ and then f is given by $1/T_{seq}$, that is the frequency of operation.

(Refer Slide Time: 36:06)



Power Reduction

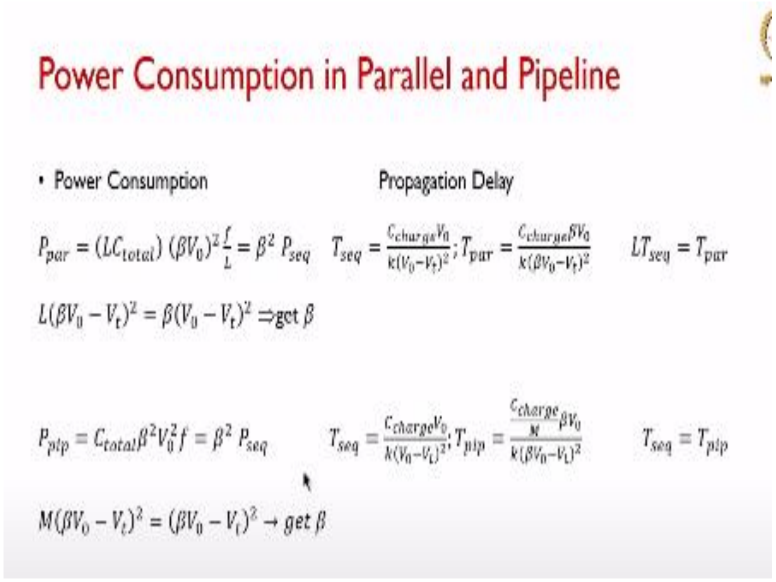
- $M(\beta V_0 - V_t)^2 = (\beta V_0 - V_t)^2 \rightarrow get\ \beta$

System	Sequential FIR (Original)	Pipelined FIR (without reducing V_0)	Pipelined FIR (with reducing V_0)
Power (Ref)	P_{Ref}	$2P_{Ref}$	$0.364P_{Ref}$
Clock Period (u.t.)	12 ut	6 ut	12 ut
Sample period (u.t.)	12 ut	6 ut	12 ut

So how we reduce the power by pipelining that is m stages of pipeline what we will be considering it by substituting in equations. So this is what we do the left hand side and right hand side will be putting it along with the sequential what we are considering are pipelined 1. And then we will be assuming beta is the value by which I can reduce my operating voltage. So we will be considering calculating this beta and then we went on to show that in the sequential 1 the, whatever our p reference takes 12 units of time whereas in the case of pipeline without reducing f_0, so we can operate it 2 times the P_{ref}.

And the clock period is going to be reduced to 6 unit of time. So if we use the reduced voltage that was reduced to 0.364 times the P_{ref} basically but I am operating at the same frequency as the original one.

(Refer Slide Time: 37:22)



Power Consumption in Parallel and Pipeline

- Power Consumption

Propagation Delay

$P_{par} = (LC_{total})\,(\beta V_0)^2 \frac{f}{L} = \beta^2\, P_{seq}$ $\quad T_{seq} = \frac{C_{charge}V_0}{k(V_0 - V_t)^2}; T_{par} = \frac{C_{charge}\beta V_0}{k(\beta V_0 - V_t)^2}$ $\quad LT_{seq} = T_{par}$

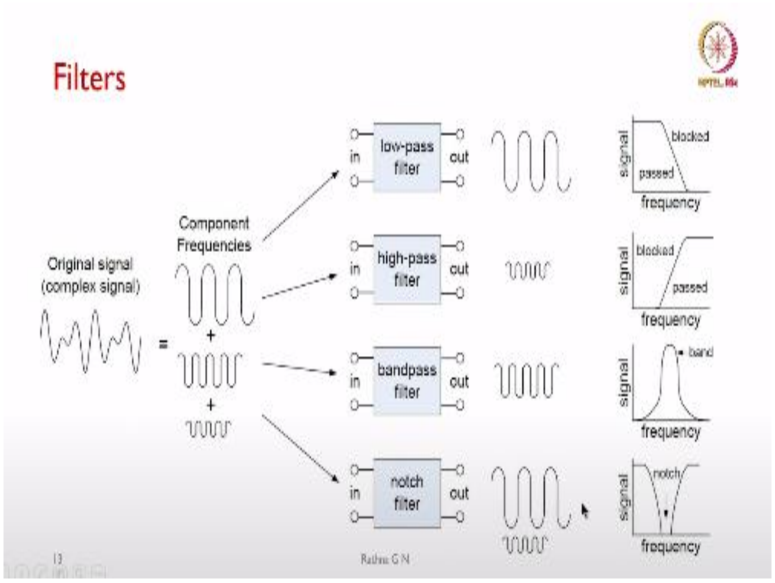
$L(\beta V_0 - V_t)^2 = \beta(V_0 - V_t)^2 \Rightarrow \text{get } \beta$

$P_{pip} = C_{total}\beta^2 V_0^2 f = \beta^2\, P_{seq}$ $\quad T_{seq} = \frac{C_{charge}V_0}{k(V_0 - V_t)^2}; T_{pip} = \frac{\frac{C_{charge}}{M}\beta V_0}{k(\beta V_0 - V_t)^2}$ $\quad T_{seq} = T_{pip}$

$M(\beta V_0 - V_t)^2 = (\beta V_0 - V_t)^2 \rightarrow get\ \beta$

So then we considered both the parallel and pipeline how we are going to consider the power consumption with respect to propagation delay we calculated so we have the l $T_{seq} = T_{par}$ what we will be having it. And then based on our pipeline as we can see that it is T_{seq} is equal to T_{pip} what will be calculating it. So the equations have been propagation delays have been given in this and will be solving l and then m values. And we can combine both what is it pipeline and parallelism to get l/m will be the operating voltage of our circuits.

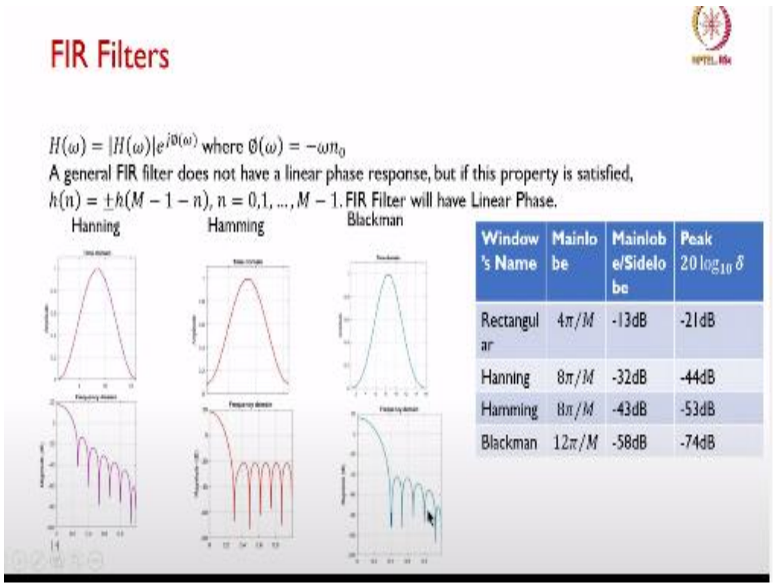
(Refer Slide Time: 38:10)



Filters

So later on we went on to define our filters so we said the common filter what you can see this is the original signal and then you have multiple components which got have added with it and either we can do low pass filtering to get one of them. And then or do the high pass filter to get the multiple of the thing, or I can select one of them to do the band pass filter. Or I can do the notch filters so that I will be eliminating these two and then I will be trying to get one of them, from the thing this is what we discussed.

(Refer Slide Time: 38:50)



FIR Filters

$H(\omega) = |H(\omega)|e^{j\emptyset(\omega)}$ where $\emptyset(\omega) = -\omega n_0$

A general FIR filter does not have a linear phase response, but if this property is satisfied,

$h(n) = \pm h(M-1-n), n = 0,1,\dots,M-1$. FIR Filter will have Linear Phase.

Window 's Name	Mainlobe	Mainlobe/Sidelobe	Peak $20\log_{10}\delta$
Rectangular	$4\pi/M$	-13dB	-21dB
Hanning	$8\pi/M$	-32dB	-44dB
Hamming	$8\pi/M$	-43dB	-53dB
Blackman	$12\pi/M$	-58dB	-74dB

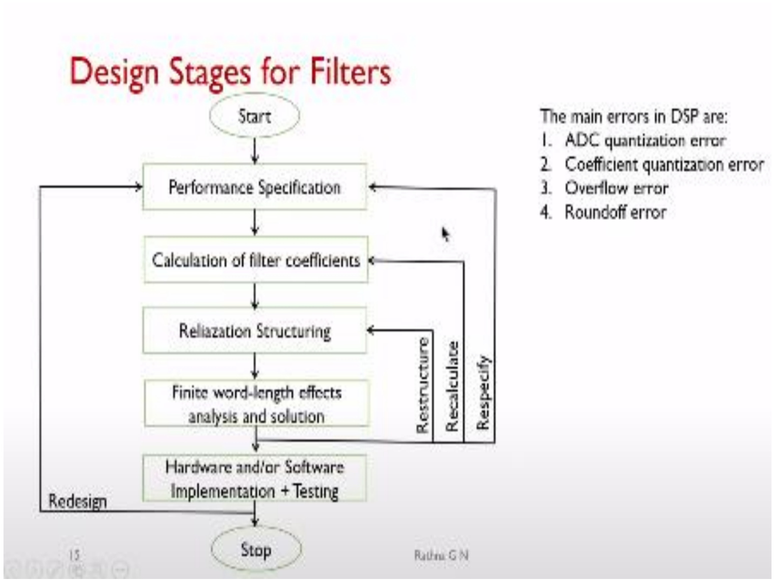
And then we went on to discuss about FIR filter how FIR filter can be used in the linear phase response. Only a normal FIR filter is not a linear phase how do we represent the magnitude and then a phase of the signal is given here that is a $|H(\omega)|e^{j\emptyset(\omega)}$. Where our $\emptyset(\omega) = -\omega n_0$ when we are going to achieve this linear phase is when the we consider the symmetric coefficients basically that is $h(n) = \pm h(M - 1 - n), n = 0,1, \ldots, M - 1$, M is the order of the filter.

So we discussed about how we can design our FIR filter in our digital domain using Hamming window, Hanning window, and then Blackman window with a different responses. Depending on what you want the peak that is - 20 log delta whatever the cutoff frequency what you want to have it whether you need it 21 db down the lane or - 44 db or up to 74 db what blackman can give.

And then we said that main lobe is going to be the smallest one for rectangle window and then as you will be seeing that Blackman has the next lowest one and then it will be same for both hamming and then hanning. So as one comments here energy neither can be created nor demolished basically what we say so how we are going to distribute the energies in the windowing thing.

So if you distribute more energy in what you will see that in the main lobe then the side lobes is going to be reduced. Basically you what you will have it or so the main lobe has what you say narrow basically then you will be seeing that peak side lobes what you will be having it and then you have to have one of them compromise on the thing.

(Refer Slide Time: 41:08)



Design Stages for Filters

The main errors in DSP are:
1. ADC quantization error
2. Coefficient quantization error
3. Overflow error
4. Roundoff error

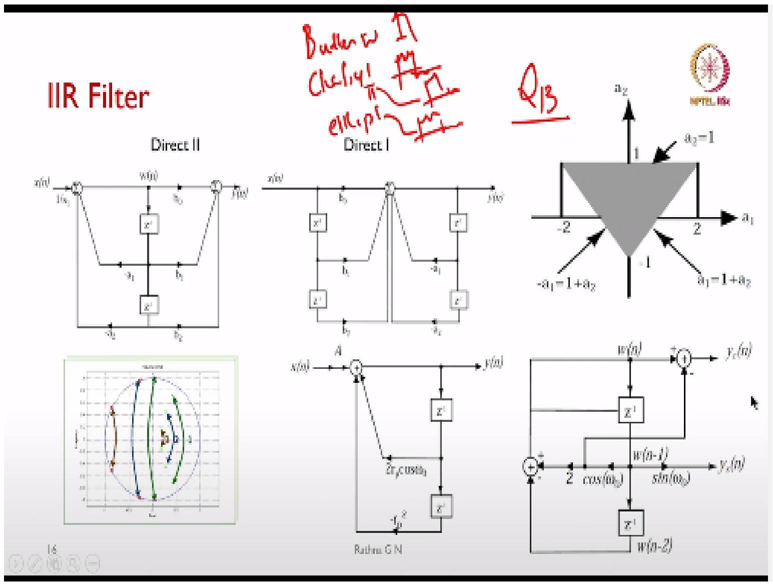
Start
Performance Specification
Calculation of filter coefficients
Reliazation Structuring
Finite word-length effects analysis and solution
Hardware and/or Software Implementation + Testing
Stop
Restructure
Recalculate
Respecify
Redesign
15
Rathna G N

So what are the design stages for our filters basically, so we are going to start we will be specifying some of the performance and then we will be doing the filter coefficients calculation using the MatLab what we said. Either it can be a FIR filter or IIR filter and then we have to define our realization structure what we wanted. So whether we want the linear phase for FIRr filter or IIR filter cascading section.

And then we will be looking at the finite word length effects analysis and solution and later on we will be putting them onto the hardware and our software implementation. And then will be going with the testing. So any of the stage if you are not satisfied the thing so you will be going back to change the either the structure or you can calculate your filter coefficients by doing little bit narrowing or widening the thing.

And then or you have to go back to specify your thing modify that itself so you will be doing all this redesign if none of them satisfy you may have to go back finally to this and then come down. And then we went on to say what are the errors that are creeping in our digital signal processing or using the digital signal process. So one is the ADC quantization error, and then we had the coefficient quantization error we looked at it, and then there is a overflow error from our signal processors, and then finally when we are representing at the output we will have the round off error.

(Refer Slide Time: 42:58)



So we set in IIR filter from the FIR filter we went on to design our IIR filter so we said direct one form and direct 2 form what are the advantages and disadvantages basically. So our storage is minimized in this and whereas here it is twice in this case and the problem with respect to these 2 is here if there are positive and negative values. So because you have only one summation unit here so it may get nullified.

Whereas in the direct form two cascade section so both negative also is getting added and then here it is the positive addition what it is going to happen. So, all of us know that multiplication is not going to cause overflow but addition can cause us overflow in this case. So this is one of the disadvantage what we have to bear with our direct 2 form. And we went on to say that our coefficient as in case of FIR filter we can represent it in q15 format but in the case of IIR filter it is not possible to represent it.

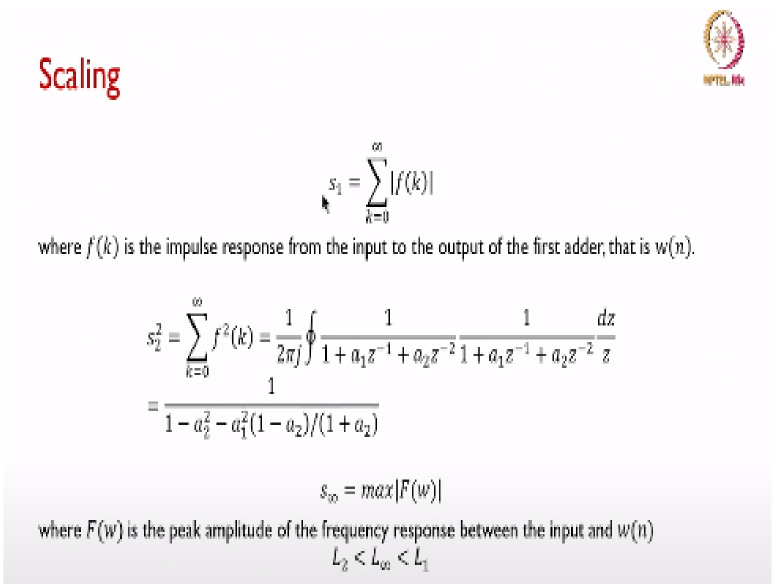
Because the maximum and then minimum value what we will be getting is - 2 to 2 so that is what we went on to show with this triangle basically. So we have to represent coefficients in q13 format for all our IIR filter applications. And we said that this how we are going to combine your most of the time we have the complex conjugate poles basically and 0's will also be occurring.

How we can combine them that is the which is from the 0 which is farthest in the pole positions which is going to be combined with the nearest 0's in the nearest to the 0 basically and then so on what you will be applying it. So that any disturbance in the thing it is going to be minimized so this has one has to work it has been over the years one has worked out. And we know that in IIR filter we will be designing our filter in using the analog domain and will be doing the transformation.

So we know that most of them are butter worth filter what will use it, or Chebyshev 1 and then 2 basically, and then the elliptic filter what we can design using in the unlock domain. So this has we know that it has a narrow steep of what we discussed it and here is this thing what Chebyshev 1 has a little ripple and then flat response. And then Chebyshev 2 is you will have a ripple in the pass band and then the flat response. And elliptic will have both ripple and pass band and then stop band.

And then we know that this is the lowest order filter what we can design and we went on to design a sine generator in the oscillatory mode. And the same oscillatory mode can be designed for the quadrature oscillator both you will be getting sine and then cosine output basically that was the application.

(Refer Slide Time: 46:42)



Scaling

$$s_1 = \sum_{k=0}^{\infty} |f(k)|$$

where $f(k)$ is the impulse response from the input to the output of the first adder, that is $w(n)$.

$$s_2^2 = \sum_{k=0}^{\infty} f^2(k) = \frac{1}{2\pi j}\oint \frac{1}{1+a_1 z^{-1}+a_2 z^{-2}} \frac{1}{1+a_1 z^{-1}+a_2 z^{-2}} \frac{dz}{z}$$

$$= \frac{1}{1-a_2^2-a_1^2(1-a_2)/(1+a_2)}$$

$$s_\infty = max|F(w)|$$

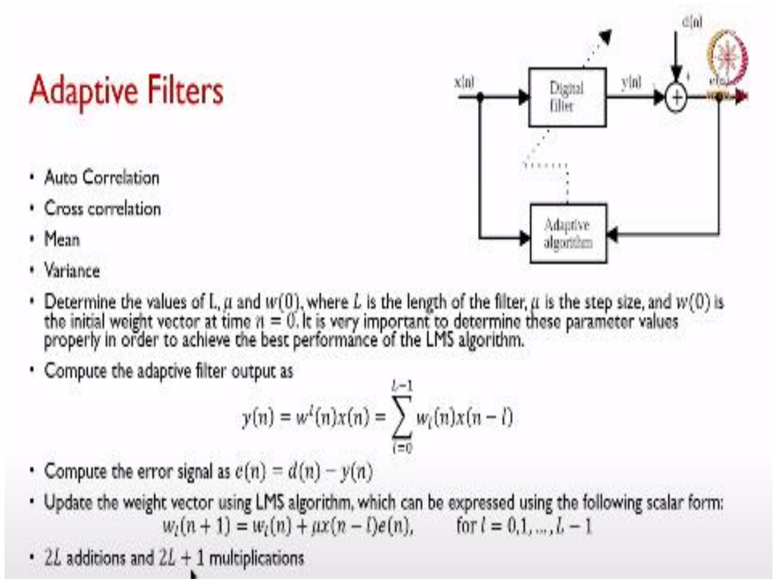
where $F(w)$ is the peak amplitude of the frequency response between the input and $w(n)$

$$L_2 < L_\infty < L_1$$

And then we went on to say that how the scaling is one of the important one in the case of IIR filter so either we can have l 1 norm or l 2 norm or l infinity norm. So we said that for hand calculation and other things we can use our pole positions a 1, a 2 coefficients to calculate our l 2 norm. So and then l infinity we have to calculate the fourier transform and then maximum of the value what we have to take it. And then went on to show that our l 2 norm is the minimum what we can calculate which is less than l infinity which is less than l 1.

So we say that $f(k)$ is the impulse response from the input or output of first adder so that is w of 1 what we take it that the magnitude of it and you have to do summation and then calculate your scaling factor in this.

(Refer Slide Time: 47:41)



x(n)
Digital filter
y(n)
d(n)
e(n)
Adaptive algorithm

Adaptive Filters

- Auto Correlation
- Cross correlation
- Mean
- Variance
- Determine the values of L, μ and $w(0)$, where L is the length of the filter, μ is the step size, and $w(0)$ is the initial weight vector at time $n = 0$. It is very important to determine these parameter values properly in order to achieve the best performance of the LMS algorithm.
- Compute the adaptive filter output as

$$y(n) = w^t(n)x(n) = \sum_{l=0}^{L-1} w_l(n)x(n-l)$$

- Compute the error signal as $e(n) = d(n) - y(n)$
- Update the weight vector using LMS algorithm, which can be expressed using the following scalar form:

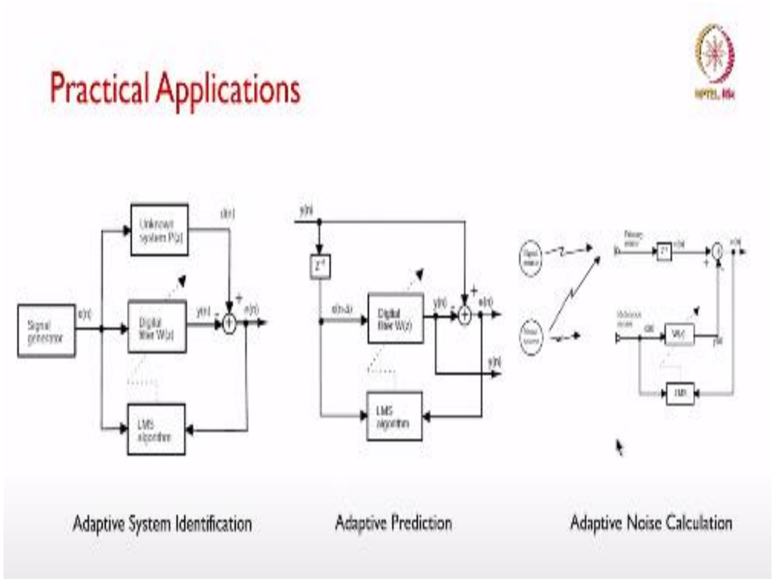
$$w_l(n+1) = w_l(n) + \mu x(n-l)e(n), \qquad \text{for } l = 0,1,\ldots,L-1$$

- $2L$ additions and $2L + 1$ multiplications

So later on we went on to discuss about adaptive filters I thought filters I will complete it and then go to my DFT and then FFT, and then DCT we can combine the thing. So we discussed about auto correlations, little bit on cross correlation, and we went on to see little bit on random variables how to calculate our mean and variance. And then we said that why do we need the adaptive filters so if we have the input and then the noise is known then you can design either FIR or IIR filters.

Where you do not know the noise you have to adapt it to the scenario so then in that case it is better to go with adaptive filters. So this is the adaptive algorithm so these are the digital filter which weights can be adjusted based on my error function. If I know the desired signal so my output is going to be subtracted and try to minimize on the error and then adjust my weights. These are the equations what we considered and we went on to show that it needs 2 l addition and then 2 l + 1 multiplications to do my adaptive filter.

(Refer Slide Time: 48:56)



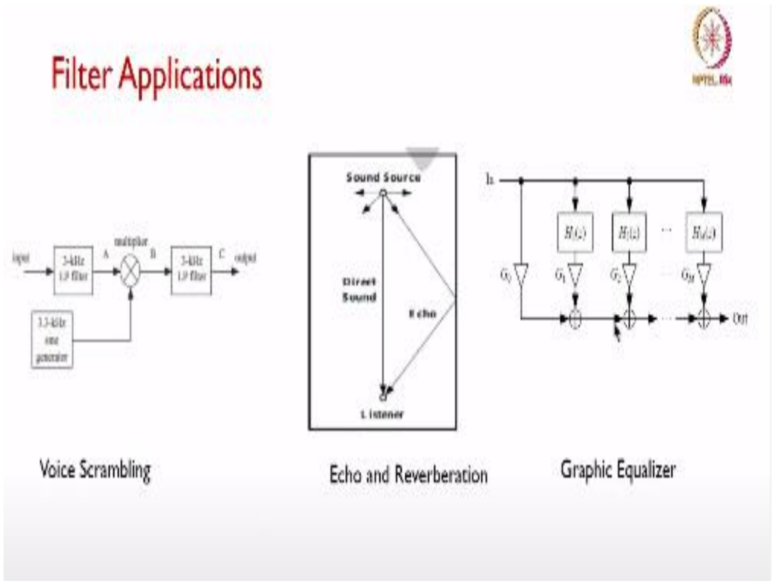
Practical Applications

So the practical applications few of it what we discussed only I have listed three here so that is one is the adaptive system identification. If I unknown system what I have with whether I can predict what is that system is using this structure basically and then computing my error function minimizing on it I will be trying to know what is this unknown system.

Later on we said that adaptive prediction what I can do it so by delaying my input itself whether I can know what kind of this thing prediction I can do my system is going to be that is what I am looking at. Next is one of the advantage of using noise cancellation in most of the speech if you are not in the closed environment where the noise is not creeping in then you will be seeing in the public address system and other things.

So you will be having 2 mic's or this is the signal what you are taking the source and noise is coming from here. So whether you can predict this noise what it can be and then feed it into the system and see whether this noise is going to be subtracted from wherever you have your speech plus noise which is going in as an input. So that I can adaptively cancel the noise most of the application as you can see that in the engine noise most of the time in the airplanes or in the cars which is going to be canceled out.

(Refer Slide Time: 50:37)



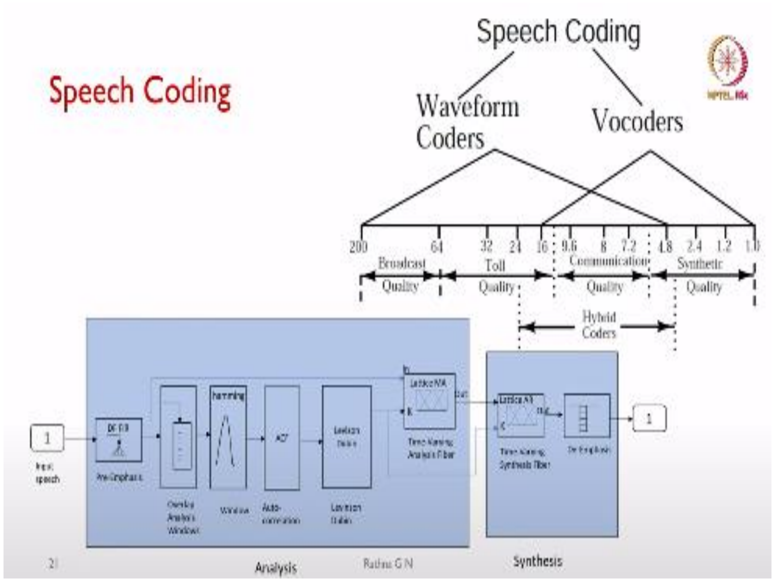
So the other filter applications what we see was in the scrambling so if you want to scramble your speech and then protect it or your audio or whatever may be the thing. So we took an example of 3 kilohertz input signal low pass filter, and then 3.3 kilohertz as my this thing sine generator what I have taken I will be adding them up at sorry I will be multiplying these two signals.

And use the low pass filter with 3 kilohertz this this can be replaced with your input or speech or audio if it is within the thing you have to take care of your filter design here, and then output is going to be a scrambled one. Same output I can feed it to the input and then do this operation I will be getting my clean speed. So that way you will be protecting your own one way of what you will call it as a safeguarding your voice basically.

The other one we said that echo we did the creation and then how we can cancel the echo which is getting created. So multiple of the, it can be a single echo or it can be multiple of them if it has got reflected we said it is a reverberation. So how we can create them how you can cancel it that is what some of the using our FIR and then IIR filters what we saw the example. Next one was using the graphic equalizer using different equalization techniques different filters it can be low pass band pass or high pass filters.

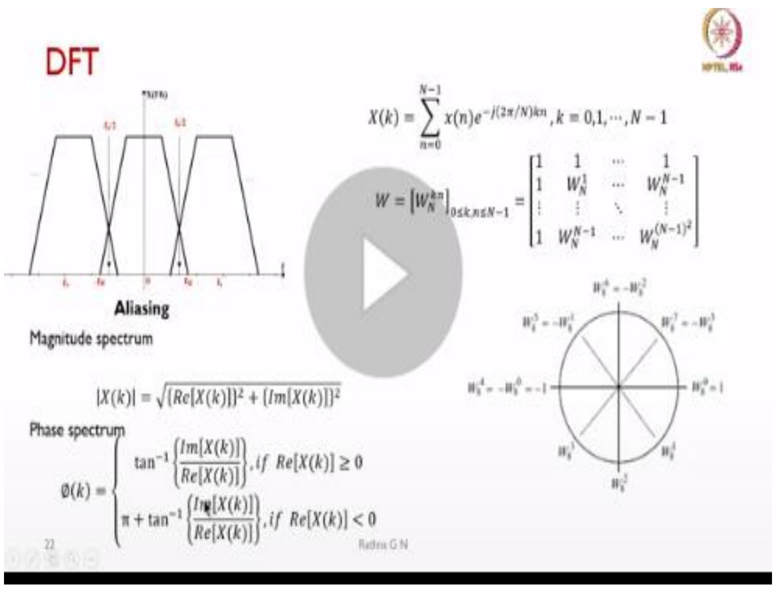
You can either allow your audio to pass through or only the instrument to pass through so most of the application you can use it in normal life.

(Refer Slide Time: 52:34)



So next we went on to discuss about the speech coding techniques whether we had the waveform coders or vocoders. And you can see that what kind of applications they use which are the bands what it is going to be used. And one of the example what we took was one of the students a project to show that LPC coding how it can be done both analysis and synthesis and then reconstruct your voice at the receiving end. Almost equivalent will not say same but at least it is going to match with whatever voice it was getting generated.

(Refer Slide Time: 53:12)



DFT

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j(2\pi/N)kn}, k = 0,1,\cdots,N-1$$

$$W = [W_N^{kn}]_{0\le k,n\le N-1} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N^1 & \cdots & W_N^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & \cdots & W_N^{(N-1)^2} \end{bmatrix}$$

Aliasing

Magnitude spectrum

$$|X(k)| = \sqrt{[Re[X(k)]]^2 + [Im[X(k)]]^2}$$

Phase spectrum

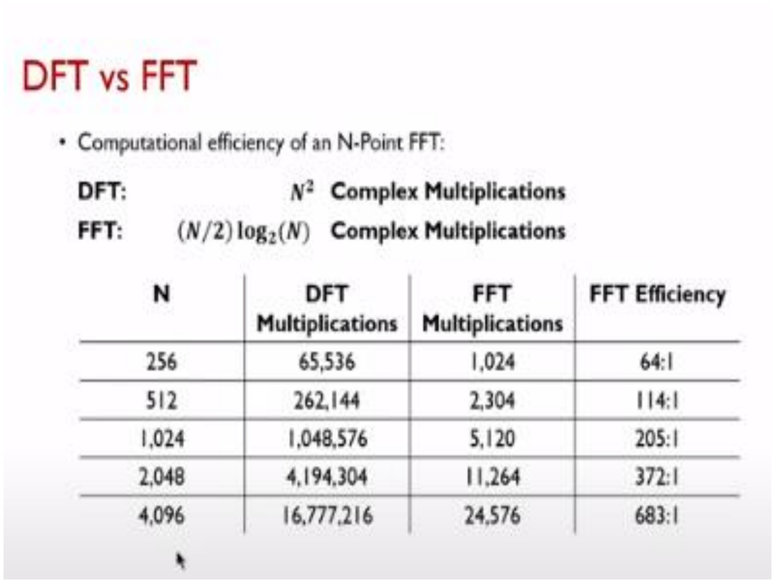
$$\emptyset(k) = \begin{cases} \tan^{-1}\left\{\frac{Im[X(k)]}{Re[X(k)]}\right\}, if\ Re[X(k)] \ge 0 \\ \pi + \tan^{-1}\left\{\frac{Im[X(k)]}{Re[X(k)]}\right\}, if\ Re[X(k)] < 0 \end{cases}$$

So later on now we went on to why we have to go for discrete Fourier transform. So in the time domain we know that some of the frequencies component that are present in our input signal may not be clearer better to go into the other domain other transformed domain to see that whether we can look at what is the thing happening to the signal. So as we said if the sampling frequency is not met so we will be seeing that there will be a overlap of the thing.

So whatever the frequencies so will be accounting for the aliasing, so we intended to have some frequency at the output but we are getting some other frequency. Then definitely we know that in the Fourier transform which frequency we are getting it how it has got aliased and then you can avoid it in the time domain. For that or your sampling frequency what you have to increase more than twice of that whatever Shannon a sampling theorem says.

So we know that the equation what we have is with respect to it is a complex equation that is e power - j both real and imaginary parts are there and we went on to say that our coefficients can be represented in this manner. And then how in the unit circle how your coefficients that is weight factors are represented is shown with that. Then we said we can calculate our application of our DFT was to find out the magnitude spectrum and then how the face is going to look at it. So we went on in the DSK board also to implement the same thing.

(Refer Slide Time: 55:03)



DFT vs FFT

- Computational efficiency of an N-Point FFT:

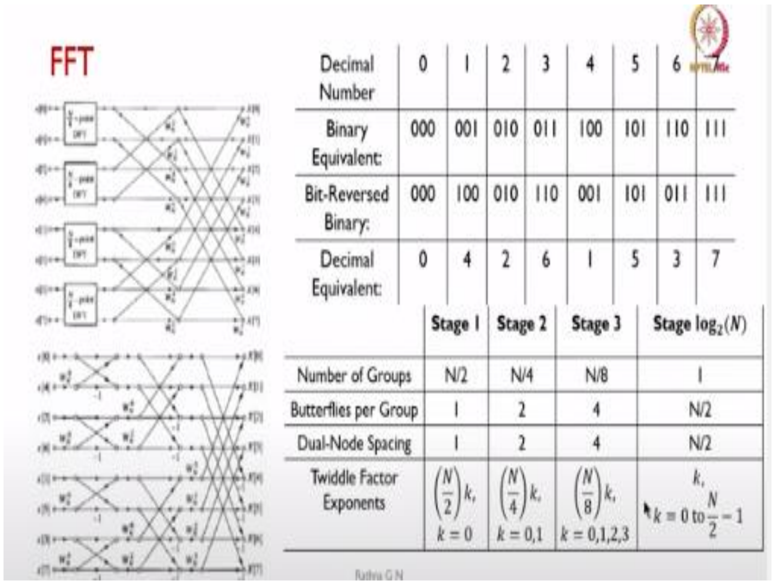
DFT: N^2 **Complex Multiplications**

FFT: $(N/2)\log_2(N)$ **Complex Multiplications**

N	DFT Multiplications	FFT Multiplications	FFT Efficiency
256	65,536	1,024	64:1
512	262,144	2,304	114:1
1,024	1,048,576	5,120	205:1
2,048	4,194,304	11,264	372:1
4,096	16,777,216	24,576	683:1

And some comparison we did the thing DFT if it is order of n squared is our complex multiplication which can be reduced to $(\frac{N}{2})\log_2 N$ complex multiplication in the case of FFT. And we have shown that how it is going to be the ratio for different order of the input basically.

(Refer Slide Time: 55:27)



FFT

Decimal Number	0	1	2	3	4	5	6	7
Binary Equivalent:	000	001	010	011	100	101	110	111
Bit-Reversed Binary:	000	100	010	110	001	101	011	111
Decimal Equivalent:	0	4	2	6	1	5	3	7

	Stage 1	Stage 2	Stage 3	Stage $\log_2(N)$
Number of Groups	N/2	N/4	N/8	1
Butterflies per Group	1	2	4	N/2
Dual-Node Spacing	1	2	4	N/2
Twiddle Factor Exponents	$\left(\frac{N}{2}\right)k,$ $k=0$	$\left(\frac{N}{4}\right)k,$ $k=0,1$	$\left(\frac{N}{8}\right)k,$ $k=0,1,2,3$	$k,$ $k=0 \text{ to } \frac{N}{2}-1$

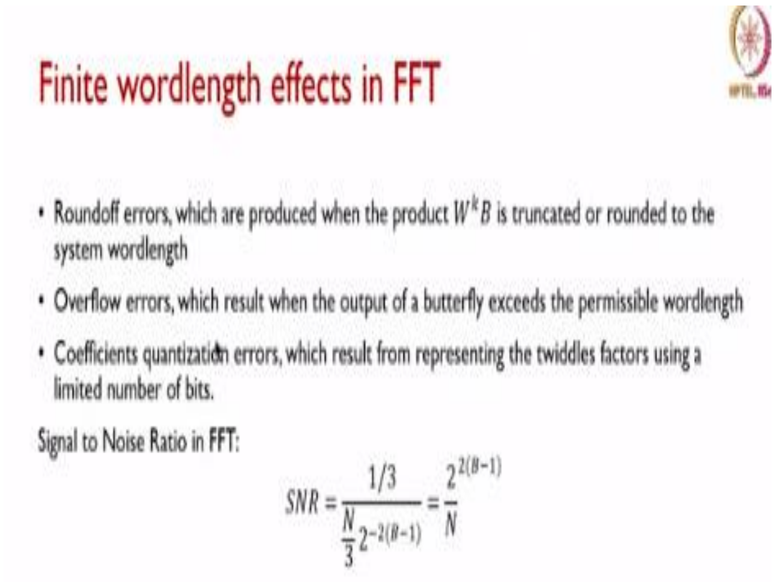
Rathna G N

And then we went on to say that our decimation in time and decimation in frequency to develop our butterfly structure to get our FFT as it is shown. And then we said that how our computation in hardware can be for the bit reversal. So if it is input is bit reversed output we said it is an bit reversed format what we are going to have it sorry in line output if input is bit reverse in line then output is going to be bit reversed.

So how we are going to access it as we discussed in the hardware we can do the reverse addition to calculate our bit reversal of the value. So that we need not have to spend on the algorithm if you go and look at the book you will be getting what complex way you have to do the reversal of it which will be taking lot of time. Whereas in the hardware we did the reverse addition so that we can immediately get our output.

And then we went on to show that so how many this thing butterflies or groups and how many twiddle factors are required at each stage because we have $\log_2 N$ stages. So we went on to say what are the twiddle factors that are going to be present in each stages.

(Refer Slide Time: 57:00)



Finite wordlength effects in FFT

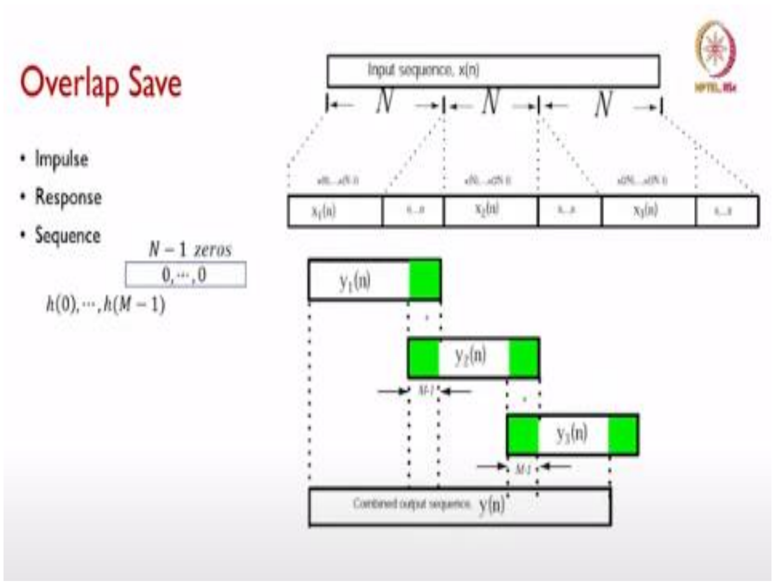
- Roundoff errors, which are produced when the product $W^k B$ is truncated or rounded to the system wordlength
- Overflow errors, which result when the output of a butterfly exceeds the permissible wordlength
- Coefficients quantization errors, which result from representing the twiddles factors using a limited number of bits.

Signal to Noise Ratio in FFT:

$$SNR = \frac{1/3}{\frac{N}{3}2^{-2(B-1)}} = \frac{2^{2(B-1)}}{N}$$

So we went on to discuss about finite word length effects in FFT also so we know that signal to noise ratio is given by this equation that is it is directly proportional to number of bits and inversely proportional to the order of your FFT what you are going to consider. And then these are the errors what we considered in FFT also.

(Refer Slide Time: 57:24)

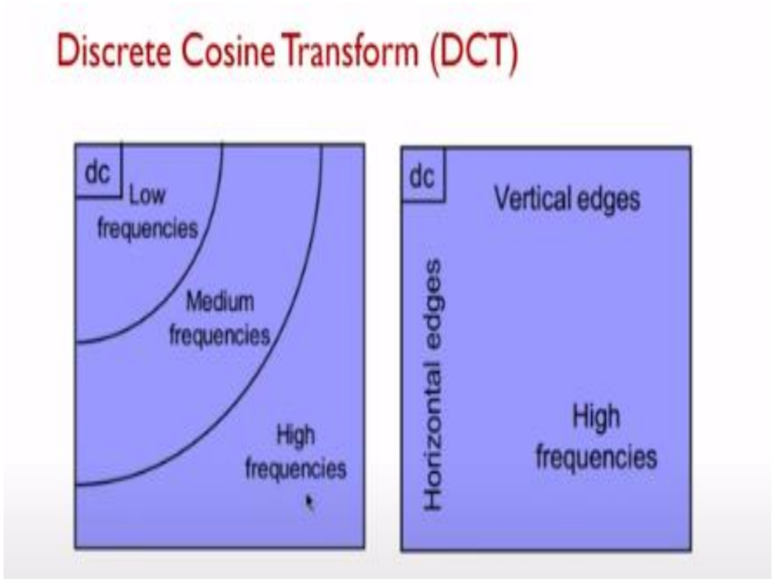


Overlap Save

- Impulse
- Response
- Sequence

And later on for a lengthy signal how we can work on using a overlap add and then overlap save method to implement the same using our FFT.

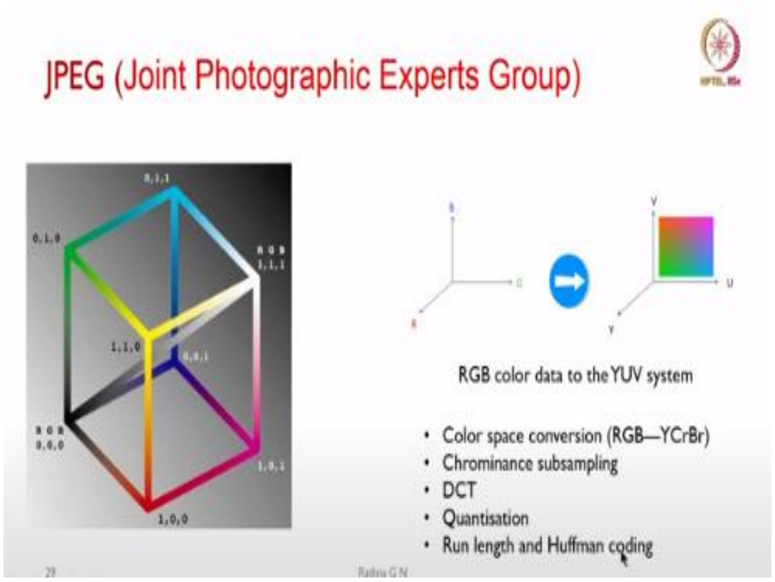
(Refer Slide Time: 57:39)



Discrete Cosine Transform (DCT)

So then we went on to discuss about discrete cosine transform that is little on image processing what we did the thing. So how the dc value is going to be maximum in case of discrete cosine transform. And then where are the low frequencies present and medium and then high frequencies. So because of our human visual system so we may not be able to view these high frequencies; so we can eliminate it and then go for the compression basically.

(Refer Slide Time: 58:10)



JPEG (Joint Photographic Experts Group)

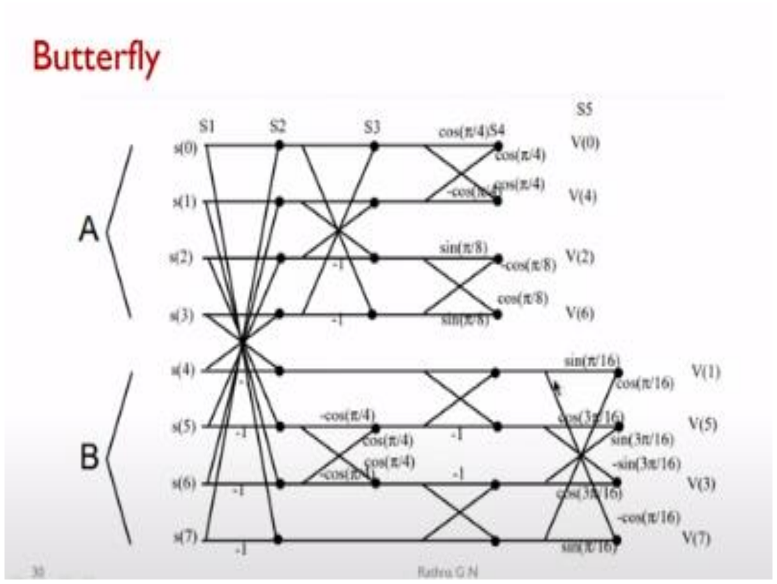
- Color space conversion (RGB—YCrBr)
- Chrominance subsampling
- DCT
- Quantisation
- Run length and Huffman coding

And then went on to say that our jpeg uses this DCT or compression technique to implement it so you will be seeing that different colors how it is represented in the RGB domain. And we know that it needs maximum word length and then number of bits are much more required in this case.

So; how we can go from one domain to the other here also in the YUV format. As you will be seeing that it is represented in only here and then illuminance what we will be having it in the y.

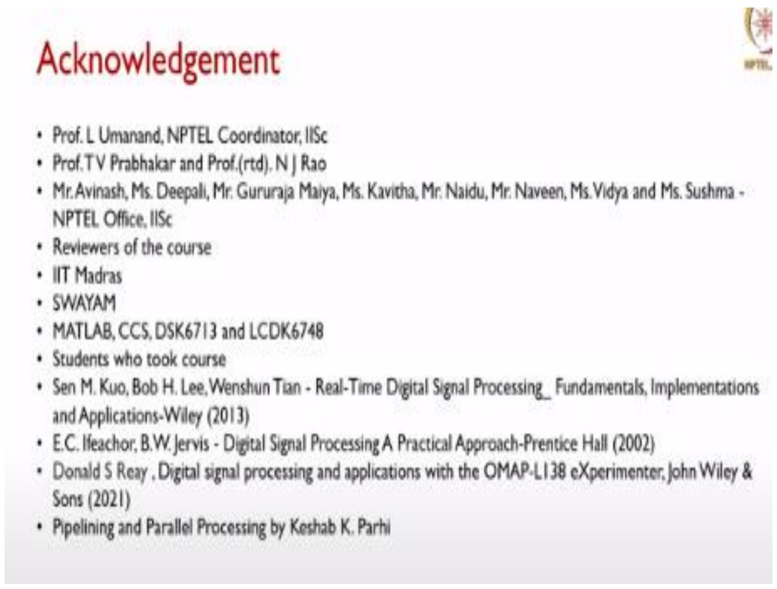
And then what we said was the color space conversion we will be doing it into that is illumination and then chrominance and then both CR and then BR what we will be taking it. And then we can do the sub sampling in the chrome and subspace basically and apply the discrete cosine transform later on go on to do the quantization. So we use different quantization matrix to see that how much information we can discard. Later on we went on to do in the DCT run length coding and in the inverse DCT it is going to use the Huffman coding to get back our original value.

(Refer Slide Time: 59:40)



Later on we went on to use butterfly structure to show that the order of multiplication for the 8 bit; 8 by 8 bit 1d or DCT needs 64 multiplications and 63 additions which we can be reduced using the butterfly structures to 13 multiplication and then 27 additions. So that is one of the technique what we used it to show this so you will be seeing cos and sine function what it is going to be used here.

(Refer Slide Time: 01:00:15)



Acknowledgement

- Prof. L Umanand, NPTEL Coordinator, IISc
- Prof. T V Prabhakar and Prof.(rtd). N J Rao
- Mr. Avinash, Ms. Deepali, Mr. Gururaja Maiya, Ms. Kavitha, Mr. Naidu, Mr. Naveen, Ms. Vidya and Ms. Sushma - NPTEL Office, IISc
- Reviewers of the course
- IIT Madras
- SWAYAM
- MATLAB, CCS, DSK6713 and LCDK6748
- Students who took course
- Sen M. Kuo, Bob H. Lee, Wenshun Tian - Real-Time Digital Signal Processing_ Fundamentals, Implementations and Applications-Wiley (2013)
- E.C. Ifeachor, B.W. Jervis - Digital Signal Processing A Practical Approach-Prentice Hall (2002)
- Donald S Reay , Digital signal processing and applications with the OMAP-L138 eXperimenter, John Wiley & Sons (2021)
- Pipelining and Parallel Processing by Keshab K. Parhi

So to wind up we have lot of people to be acknowledged few of them what I have listed here one is professor L Umanand NPTEL coordinator for allowing me to take up this course. And, then professor T V Prabhakar, who is the one who initiated me to give this NPTEL course. And then initial he looked into my all the course contents and other thing he said it is very good and then you can go head.

And then professor N J Rao was the mentor I should ask once I have given everything I was supposed to give my lectures, he sat with me and then said these are the things what you can do it so final tuning and everything that was done by professor N J Rao. And then I cannot forget to acknowledge all NPTEL office people that is Avinash, Deepali and then Guruja Maiya, Kavitha, Naidu, Navin, Vidya and Sushma.

If I have forgotten any other names these are the ones I could remember sorry I try to acknowledge everyone and bearing with me my lunch hour recording and other things they had to forego a little bit fine tune their lunches and other things. And then I am very grateful to my reviewers of the course because those are the ones they said your course content is too much you may not be able to finish it in 12 weeks.

So better to fine tune it and then it has to be fine-tuned and then it has taken the stage. And first of all I have to tell that IIT madras for giving me permission to do this course for NPTEL and Swayam government organization who is I think no doing all the relay and everything and putting all questions and everything. And whatever the support required from MatLab because they were the first one to contact me saying that what are the tool boxes you needed for the students to use it.

So definitely I have to acknowledge them for continuously in touch with them what are the things I need it, and then providing online MatLab for all of you to do the thing. Later on I took the permission of code composer studio although it is there online and those students who want to use their kits DSK 6713 in their colleges they can use that. And then I have used the LCDK 6748 board as I was mentioning both it is a fixed point and then a floating point board for all my real time applications.

And this is would not have happened to show all the exercises what I have shown with without my students who took DSP system design and real-time digital signal processing course with me so far. And then some of the books I have referred few of them what I have listed it is a list may little bit go more actually. The first one is Sen M Kuo real-time digital signal processing fundamentals implementation and applications from Wiley that is 2013 version what I am using it.

And Ifeachor for my fixed point implementation and scaling and other things what I have referred from this book. And the Donald Reay you would have seen some of the examples I was running from this book which uses both OMAP l138 and then for the LCDK 6748 is the subset of this experimental kit, from John Wiley which is the 2021 version. And one more what I have used pipelining and parallel processing chapter from Keshab K. Parhi on the VLSI digital signal processing book.

So if I have forgotten and first of all I have to thank you all for taking this course and then it is going well. And if I am a little bit delayed in answering your questions please bear with me thanks to one and all so happy listening and then working with respect to all lab experiments thank you.