UNIT - 4

Analysis of finite Word length Effect in fixed-point DSP Systems:-

1) Finite word length Effects in Digital filters:-

Introduction:- Fundamental operations in various Computational Procedures like Convolution spectral estimation etc., in DSP are Multiplication and Addition.

These operations are performed using sample of i/p sequence, sample of impulse response and co-efficients of differential equation governing system.

→ The registers are basic storage device in digital systems. The Maximum size of binary information (or) data that can be stored in register is called register word length.

Eg:- When register stores an 8-bit data then its word length is 8-bit.

For storing o/p data in register they have quantized and coded in binary.

where quantization & coding depends on register word length.

Eg:- If register word length is 8-bit we can generate $2^8 = 256$ binary codes, so we have 256 quantized levels.

The quantization and coding will introduce error in o/p data, bcoz the analog data has infinite precision but the digital equivalent has finite Precision.

→ While Performing Computation size of result may be exceeding the size of register used storing the result.

Eg:- If result of addition of two eight bit data may be 8 (or) 9 bits and result of multiplication of two 8-bit data may be 16-bits.

→ In case of register used to store the result 8-bit the result has to be truncated (or) rounded to accomodate in the register. This makes system is non-linear and leads to limit Cycle behaviour.

→ The effective of truncation & rounding can be represented in terms of an additive error signal, which is called round off noise.

(*) Finite word length effects in digital filters:-

1. Error due to quantization of o/p digital data by A/D Converter
2. " " " " " filter co-efficients
3. " " " rounding the Product in multiplication.
4. " " " overflow in addition.
5. Limit cycles.

2) Representation of Numbers in Digital systems:-

In a radix representation the no: can be represented by a summation relation as

Eg:- $178.25_{10} = (1\times10^{+2}) + (7\times10^{+1}) + (8\times10^{0}) + (2\times10^{-1}) + (5\times10^{-2})$

$= (d_2\times10^{+2}) + (d_1\times10^{1}) + (d_0\times10^{0}) + (d_1\times10^{-1}) + (d_2\times10^{-2})$

$= \sum_{i=-2}^{2} d_i \gamma^{-i}$ where $\gamma = 10$

Eg:- $111.11_2 = (1\times2^{2}) + (1\times2^{1}) + (1\times2^{0}) + (1\times2^{-1}) + (1\times2^{-2})$

$= (d_{-2}\times2^{2}) + (d_{-1}\times2^{1}) + (d_0\times2^{0}) + (d_1\times2^{-1}) + (d_2\times2^{-2})$

$= \sum_{i=-2}^{2} d_i \gamma^{-i}$; where $\gamma = 2$.

i.e generally any number be represented as

Number, $N = \sum_{i=-A}^{B} d_i \gamma^{-i}$ → (1)

A = no: of integer digits

B = no: of fraction digits

γ = radix (or) Base

d_i = i^{th} digit of the number.

where d_i ranges $0 \le d_i \le (\gamma-1)$

Eg:- If $\gamma = 2$; $d_i = 0$ (or) 1

where $\gamma = 10$; $d_i = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$.

→ The radix $r=2$ can be represented from binary number

Binary number $N = \sum_{i=-A}^{B} d_i 2^{-i}$ → (2)

In binary digit d_{-A} is called Most significant Bit (MSB) and binary digit d_B is called least significant bit (LSB) of binary number 'N'.

The binary point b/w the digits d_0 and d_1 does not exists physically in digital systems.

→ In various computation procedures of DSP we use fraction format bcoz mixed no:- (no:- with integer and fraction parts) are difficult to multiply and the number of bits representing an integer. Cannot be reduced by truncation (or) rounding. for fraction format of binary no:- is modified as.

Binary fraction no:- $N = \pm \sum_{i=1}^{B} d_i 2^{-i}$

(or)

Binary fraction number $N = \cdot \sum_{i=0}^{B} d_i 2^{-i}$ → (3)

where d_0 is used to represent the sign of number.

There are 2 - methods representing binary numbers.

(a) Fixed point representation
(b) Floating point "

(a) Fixed point representation :- 3 - Different formats

(i) Sign - magnitude format
(ii) Ones Complement "
(iii) Two's " "

In fixed point representation is only unique way of representing positive binary fraction no:-

Positive binary fraction number $N_f = 0.d_1 d_2 \ldots d_B$

$= (0 \times 2^0) + (d_1 \times 2^{-1}) + (d_2 \times 2^{-2}) + \ldots + (d_B \times 2^{-B})$

$= \sum_{i=0}^{B} d_i 2^{-i}$ where $d_0 = 0$ → (4)

where d_0 is set to zero to represent the positive sign.

(i) Sign magnitude format:-

The negative value of given no:- differ only in sign bit (i.e bit d$_8$)
The sign bit d_0 is zero for +ve and one for -ve number.

0.125 ×2 → 0.250 ×2 → 0.500 ×2 → 0.000
↓ .001

∴ Positive binary fraction no:- $N_p = (0 \times 2^0) + \sum_{i=1}^{B} d_i 2^{-i}$ → (5)

Negative " " " $N_N = (1 \times 2^0) + \sum_{i=1}^{B} d_i 2^{-i}$ → (6)

Eg:- $+0.125_{10} = 0.001_2$ (or) 0001_2 ; $-0.125_{10} = 1.001_2$ (or) 1001_2

Note:- The binary point b/w d_0 and d_1 is not mandatory becoz

(ii) One's Complement format:-

Here +ve no: is same as sign magnitude format and
-ve no is given by obtaining by bit by bit complement.

∴ Complement of digit d_i can be obtained by subtracting the digit from one.

i.e $d_i = \bar{d}_i = (1 - d_i)$ → (7)

∴ Negative binary fraction no: in one's complement $\left.\right\} N_{1c} = (1 \times 2^0) + \sum_{i=1}^{B} (1 - d_i) 2^{-i}$ → (8)

Eg:-
✓ $+0.125_{10} = 0.001_2$ (or) 0001 ;
✓ $-0.125_{10} = 1.110_2$ (or) 1110_2.

(iii) Two's Complement:-

The +ve no:- is same as sign magnitude format
-ve no:- is obtained by taking one's complement of positive representation and adding one to LSB.

∴ Negative binary fraction no: in two's complement $\left.\right\} N_{2c} = (1 \times 2^0) + \sum_{i=1}^{B} (1 - d_i) 2^{-i} + (1 \times 2^{-B})$ → (9)

Eg:-
✓ $+0.125_{10} = 0.001_2$ (or) 0001 ; ✓ $-0.125_{10} = 1.111_2$ (or) 1111_2.

Decimal Equivalent of 4-bit binary no:- in fixed Point representation:- (5)

Binary number in fixed Point representation:			Decimal Equivalent
Sign magnitude	one's Complement	two's Complement	
0.000	0.000	0.000	0
0.001	0.001	0.001	1/8
0.010	0.010	0.010	2/8
0.011	0.011	0.011	3/8
0.100	0.100	0.100	4/8
0.101	0.101	0.101	5/8
0.110	0.110	0.110	6/8
0.111	0.111	0.111	7/8
1.000	1.111	1.111	-0
1.001	1.110		-1/8
1.010	1.101	1.110	-2/8
1.011	1.100	1.101	-3/8
1.100	1.011	1.100	-4/8
1.101	1.010	1.011	-5/8
1.110	1.001	1.010	-6/8
1.111	1.000	1.001	-7/8
		1.000	-8/8

(b) Floating Point representation:-

Is employed to represent larger range of no:- in a given binary word size

Floating Point no:- $N_f = M \times 2^E$

M - mantissa ; E = Exponent

The value of M - ranges $0.5 \le M \le 1$ & M is binary func format

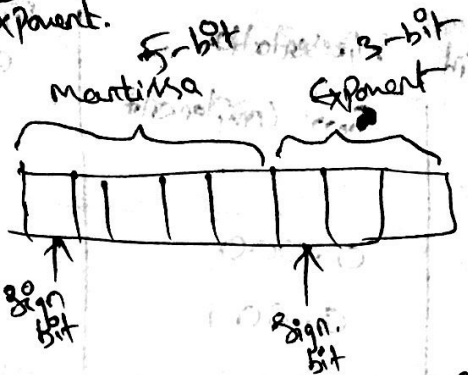
E is called Exponent and it is either Positive (or) negative integer.

LSB of mantissa/Exponent represents sign (either +ve/-ve).

↓ ↓

0 1

In floating point representation Explained by 5-bits mantissa and 3-bits Exponent.



$\Rightarrow M \times 2^{E}$

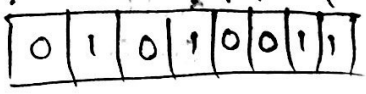
→ 8-bit.

the range of floating point represented is

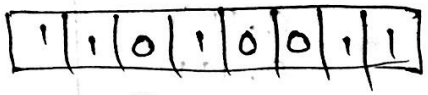
$\pm\left(2^{-4} \times 2^{-3}\right)$ to $\pm\left(2 - 2^{-4}\right) \times 2^{3}$ i.e $\left(\pm 7.8125 \times 10^{-3}\right.$ to $\left.\pm 15.5\right)$

Eg:- Binary format

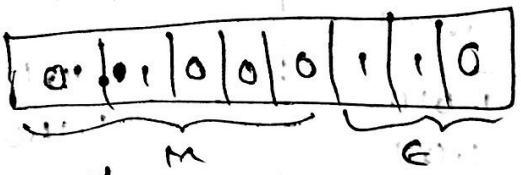
$+5_{10} = +101_2 = 0.1010 \times 2^{+3} = 0.1010 \times 2^{+11_2}$ (binary form)



$-5_{10} = -101_2 = 1.1010 \times 2^{+3} = 1.1010 \times 2^{+11_2}$



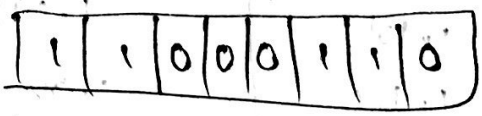
$+0.125_{10} = +0.001_2 = 0.0010 \times 2^{0} = 0.1000 \times 2^{-2} = 0.1000 \times 2^{-10_2}$



Complement

$-0.125_{10} = -0.001_2 = 1.0010 \times 2^{0} = 1.1000 \times 2^{-2}$ (binary)

$= 1.1000 \times 2^{-10_2}$



2) Types of Arithmatic in Digital systems:-

classified broad classes

Fixed Point Arithmatic ← → Floating point Arithmatic.

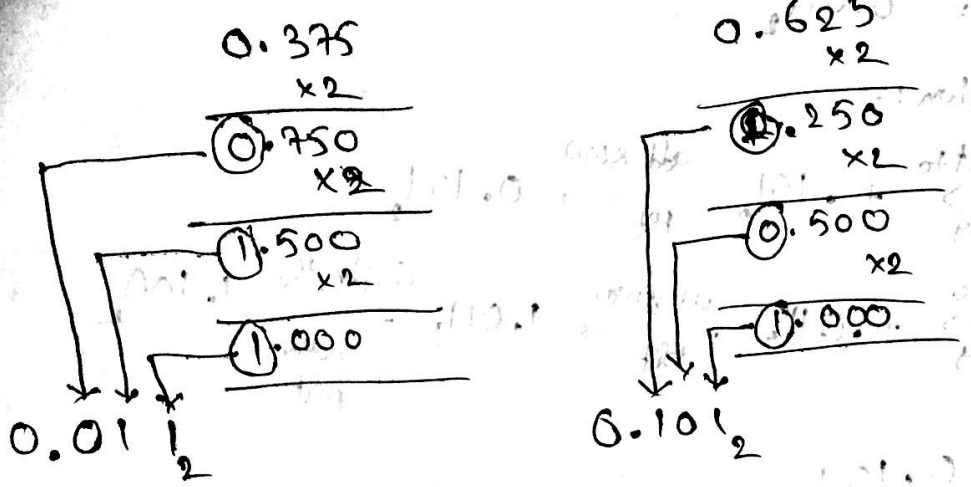
3-types

- Sign magnitude Arithmatic (avoided in Digital system due to difficulties in handling -ve no during addition)
- one's complement Arithmatic
- 2's Complement Arithmatic

1's Complement Addition:-

Ex:- Add +0.375 and −0.625 by 1's Complement Addition.

Step-1 Convert Decimal to binary



Step-2:- Representation is given as

$+0.375_{10} \xrightarrow{\text{Convert to binary}} +0.011_2 \xrightarrow{\text{add sign bit}} 0.011_2$

$-0.625_{10} \xrightarrow{\text{Convert to Binary}} -0.101_2 \xrightarrow{\text{add sign bit}} 1.101_2 \xrightarrow{\text{Complement Fraction Part}} 1.010_2$

Step-3 Addition

$(+0.375)_{10} = 0.011$

$+ (-0.625)_{10} = +1.010$

$0\ 1.101 \leftarrow$ sum

carry ←

∵ carry is zero the sum is −ve so the sum is converted to decimal as.

$1.101_2 \xrightarrow[\text{sign bit}]{\text{extract}} -0.101_2 \xrightarrow[\text{Fraction Part}]{\text{Complement}} -.010_2 \xrightarrow[\text{decimal}]{\text{Convert to}} -0.25_{10}$

Binary to decimal

$.010_2 = (0\times 2^{-1}) + (1\times 2^{-2}) + (0\times 2^{-3}) = 0.25_{10}$

$\therefore (+0.375_{10}) + (-0.625_{10}) = (-0.25_{10})$

Eg 2:- Add +0.625 and −0.375 by 1's Complement addition.

Step 1:- Decimal to binary we get

$+0.625 = 0.101_2$

$-0.375 = 0.011_2$

Step 2 representation:-

$+0.625_{10} \xrightarrow{\text{Convert to Binary}} +.101_2 \xrightarrow{\text{add sign bit}} 0.101_2$

$-0.375_{10} \xrightarrow{\text{Convert to Binary}} -.011_2 \xrightarrow{\text{add sign bit}} 1.011_2 \xrightarrow{\text{Complete the fraction part}} 1.100_2$

Step 3:- now

$$\begin{array}{r} 0.101 \\ +\ (1)\ 1.100 \\ \hline (1)\ 0.001 \leftarrow \text{sum} \\ \text{Carry} \leftarrow \quad \rightarrow 1 \text{ add Carry to LSB} \\ \hline 0.010 \leftarrow \text{final sum} \end{array}$$

∵ Carry is one, the sum is positive.

now final sum Converted to decimal.

$0.010 \xrightarrow{\text{Extract sign bit}} +.010 \xrightarrow{\text{Convert to decimal}} +0.25_{10}$

$\therefore (+0.625_{10}) + (-0.375_{10}) = (+0.25_{10})$ ✓.

Eg:-3 Add +0.375 and −0.625 by two Complement addition

Sol:- Representation as:-

$+0.375_{10} \xrightarrow{\text{Convert to Binary}} +0.011_2 \xrightarrow{\text{add sign bit}} 0.011_2$

$-0.625_{10} \xrightarrow{\text{Convert to Binary}} -.101_2 \xrightarrow{\text{add sign bit}} 1.101_2 \xrightarrow{\text{Complet fraction Part}} 1.010_2 \xrightarrow{\text{add one to LSB}} 1.011_2$

$$\begin{array}{r} 0.011 \\ 1.011 \\ \hline (0)\ 1.110 \leftarrow \text{sum} \\ \text{Carry} \end{array}$$

⇒ 0.011_2 / 1.010_2 / 0

ce Carry is zero the sum is '-ve'

convert to decimal

$1.110_2 \xrightarrow{\text{Extract sign bit}} -.110_2 \xrightarrow{\text{Complem the fraction part}} -.001_2 \xrightarrow{\text{add one to LSB}} -.010_2 \xrightarrow{\text{Conv to Binary}} -0.25_{10}$

$\therefore$ $+0.375_{10} \Rightarrow 0.011_2$

$-0.625_{10} \Rightarrow 1.011_2$

$1.110_2 \Rightarrow -0.25_{10}$.

Eg:- Add $+0.625_{10}$ and -0.375_{10} by two's Complement Addition.

The 2's Complement representation is

$+0.625_{10} \xrightarrow{\text{Convert to Binary}} +.101_2 \xrightarrow{\text{add sign bit}} 0.101_2$

$-0.375_{10} \xrightarrow{\text{Convert to Binary}} -.011_2 \xrightarrow{\text{add sign bit}} 1.011_2 \xrightarrow{\text{Complet the fraction part}} 1.100_2 \xrightarrow{\text{add one to LSB}} 1.101_2$

$\Rightarrow$ 0.101_2

$+1.101_2$

carry $\leftarrow$ (1) 0.010 $\leftarrow$ sum

$\therefore$ Carry is '1' the sum is Positive. then Carry is discarded in 2's Complement addition.

$\therefore$ $0.010_2 \xrightarrow{\text{Extract sign bit}} +.010_2 \xrightarrow{\text{Convert to Decimal}} +0.25_{10}$.

(i.e $0\times2^{-1}+0\times2^{-2}+0\times2^{-3}$ $=0.25_{10}$)

$\therefore$ $+0.625_{10} \Rightarrow 0.101_2$

$-0.375_{10} \Rightarrow 1.101_2$

$(+0.625_{10}) + (-0.375_{10}) \Rightarrow 0.010_2 \Rightarrow +0.25_{10}$

Floating Point addition:-

Addition can be performed only when Exponent of both no: are Equal.

i.e Exponent of smaller no:- is changed to Equal Exponent of bigger no:

Consider 10 - bit floating Point format with

7 - bit mantissa
3 - bit Exponent

Let LSB in mantissa and exponent be sign bit.

let: $+5_{10} = 101_2 = 0.101000 \times 2^{+3_{10}} = 0.101000 \times 2^{+11_2}$

$= 0.101000 \times 2^{011_2}$

$+0.25_{10} \Rightarrow$

0.25 ×2
0.50 ×2
1.00

.01

$+.01_2 = 0.100000 \times 2^{-1_{10}} =$

$\Rightarrow +0.25 \Rightarrow$ ~~$+.01_2 = 0.00$~~

$0.1000000 \times 2^{-01_2} = 0.1000000 \times 2^{10_2}$

When exponents are not Equal. so The small no:- is +0.25 and so it is unnormalized to make its Exponent Equal to that +5

$+0.25_{10} = 0.1000000 \times 2^{101_2} = 0.1000000 \times 2^{-1}$ unnormalize $= 0.000010 \times 2^{3}$

$= 0.000010 \times 2^{011_2}$

Now unnormalized mantissa of $+0.25_{10}$ is added to mantissa Exponent +5.

$+5_{10} \Rightarrow \quad 0.101000 \times 2^{011}$

$+0.25_{10} \Rightarrow \quad 0.000010 \times 2^{011}$

$0.101010 \times 2^{011} = 5.25_{10}$

$\Rightarrow 101.01_2 = 5.25_{10}$ //.

dy =

3) Truncation & Rounding :-

If finite word length of register and if have infinite number of data then how to fit it in to a register

If 8-bit Register ← and fit into register ← number is 10-bits } solved by using 2-methods → (1) Truncation, (2) Rounding

After using 2-methods what is effect on design parameters of DSP

If Eg:- 10.2015623 87542 → neglected
↳ displayed in calculator

Truncation

what is happened when neglected in number then we study the effects in truncation.

then this type of truncating number/ stop the number required is called truncation

Rounding :- Eg:- 10.201562

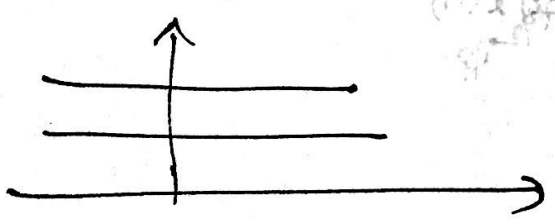
↳ 2-ways of Rounding If we keep '3' as it is then it is truncation.

Rule :- If the number is high above '5' then it will round off to the next digit to next higher one. 10.2015623 87542

Rounded to 3 | Rounded to 2 | Rounded to 1 | Rounded to 5 | Rounded to 4

Rounded for '4'

⇒ 10.2015624

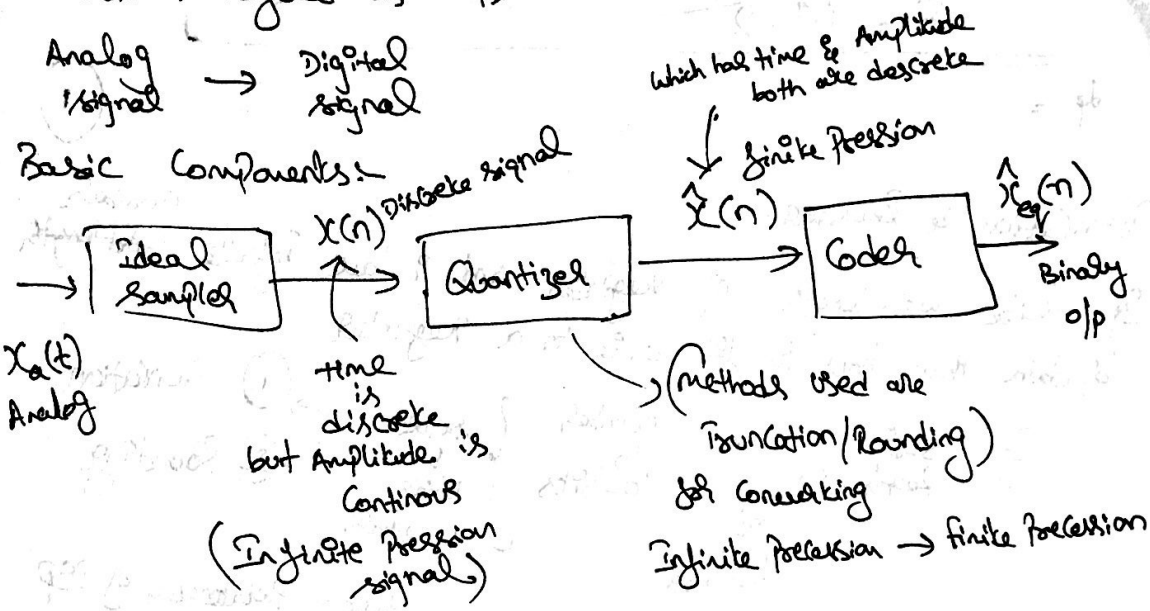


Quantization noise:-

This is refered as A/D Conversion noise

Analog signal → Digital signal

Basic Components:-



Let us Consider

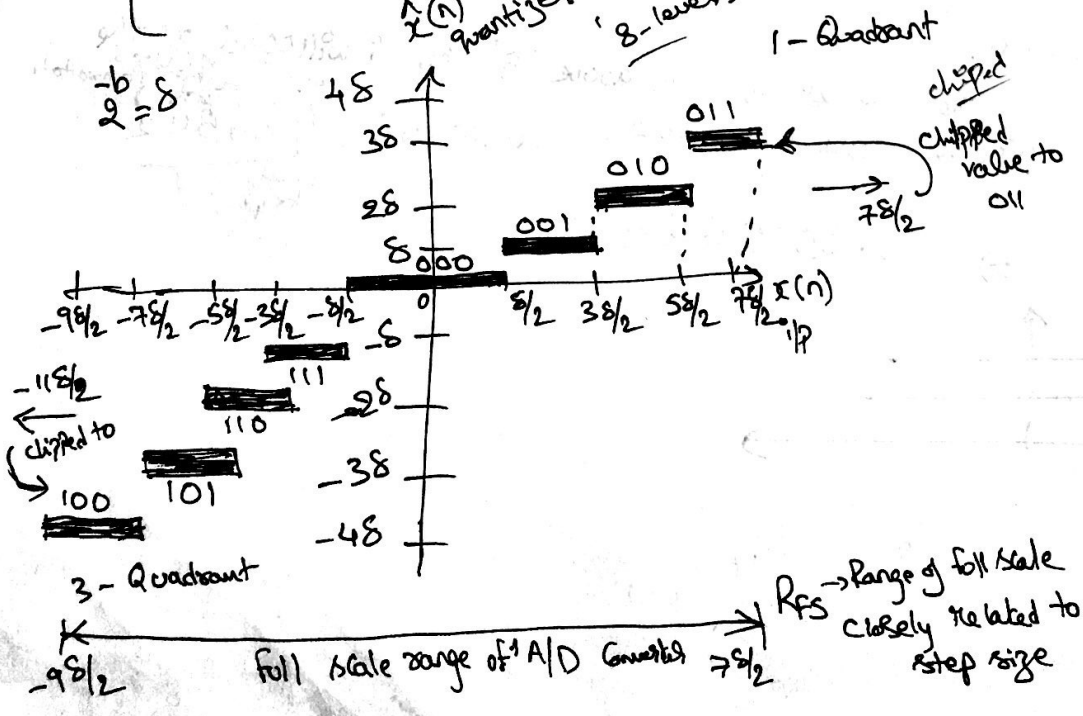
3-bit Quantizer (or) 3-bit Coder:-

q^3 = 8 levels i.e $b = 2$, $b+1 = 3$

true numbers:
- $0.0\,0$ ($2^{-1}\,2^{-2}$) Jump $1/4 = 0.25$
- $0.0\,1$ $1/2 = 0.5$ $(\frac{1}{4}+\frac{1}{4})$
- $0.1\,0$ $3/4 = 0.75$ $(3\,1/4 + 1/2)$
- $0.1\,1$

2's Complete -ve numbers:
- $1.0\,0 \Rightarrow 111$
- $1.0\,1 \Rightarrow 110$
- $1.1\,0 \Rightarrow 101$
- $1.1\,1 \Rightarrow 100$

If suppose o/p goes from $7\delta/2 \rightarrow 9\delta/2$ then all the o/ps are truncated. o/p.



using b+1 = 3-bits total no: of levels can be 8 but for bipolar signals

+ve no will be represented using magnitude

-ve " " " " " 2's Complement

if $b = 2$

$2^{-b} = 2^{-2} = 1/4$

if i/p is above $+7\delta/2$ then Quantizer will clipp the o/p to 0.11 (or) 011

$\Rightarrow 0.75\ (3/4) \Rightarrow 1 - 2^{-b} = 1 - 2^{-2} = (1 - \frac{1}{4}) = 3/4$.

if i/p is below $-9\delta/2$ then Quantizer will clip o/p to 100 (2's complete) → 011

From the graph to define truncation & Rounding

2's ↓ 1

-1.00

$2^0 \cdot \bar{2}^1 \cdot \bar{2}^2 =$

if i/p value is in b/w $\delta/2$ to $3\delta/2$ then the half value

(or) middle value then decision goes

if it is down then it is truncation (000)

if it is up then it is Rounding (001)

→ Quantization is a nonlinear Process.

$R_{fs} = 2^{b+1} \cdot \delta$ ← R_{fs} step size relation

final $\hat{x}_{eq} = \frac{2\hat{x}}{R_{fs}}$ i.e $-1 < \hat{x}_{eq} < 1$ ← Coder bound

← R_{fs} Coder relation

range is $-(2^{-b}+1)\,\delta/2 \le x_q(n\bar{1}) \le (2^{b+1}-1)\cdot\delta/2$

The noise is

$e(n) = \hat{x}(n) - x(n)$

(or)

$e(n) = Q(x(n)) - x(n)$

Rounding noise is bounded by

$-\frac{1}{2}(2^{-b} - 2^{-b_1}) \le E_R \le \frac{1}{2}(2^{-b} - 2^{-b_1})$

where b_1 – before rounding bits

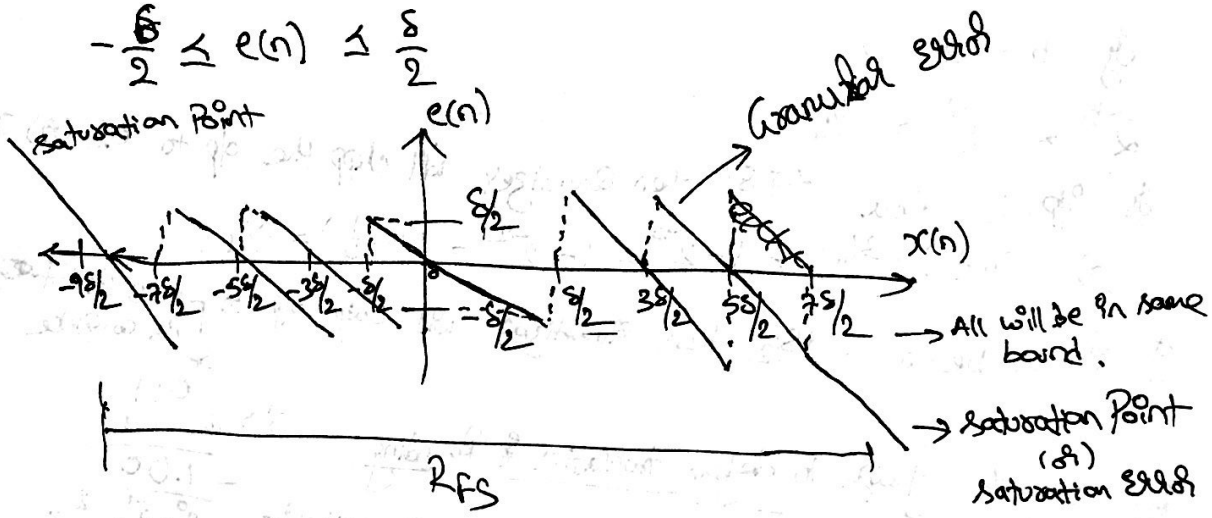
b – after " bits i.e $b_1 >> b$.

∴ The error bounded to

$$-\frac{1}{2}\,2^{-b} \le e(n) \le \frac{1}{2}\cdot 2^{-b}$$

where $2^{-b} = \delta$ so

$$-\frac{\delta}{2} \le e(n) \le \frac{\delta}{2}$$



If error is swinging b/w $-\delta/2$ to $\delta/2$ then it is granular error

If the error is going beyond the saturation point it is saturation error.

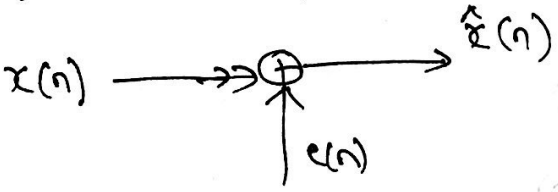
(*) **Statistical model of error :-**

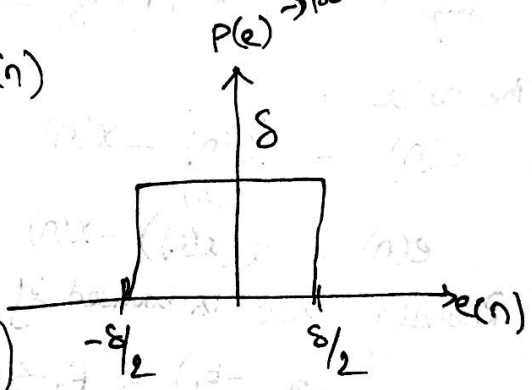
Consider error signal is random process

(1) e(n) → WSS white noise process
↑ white sense stationary process
& it is uniformly distributed over 1-sample period

(2) e(n) → are uncorrelated to i/p. (i.e error remains same)
(independent of each other)

(3) x(n) → stationary Random Process

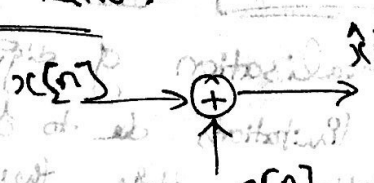




Mean of error $(m_e) = \dfrac{-\delta/2 + \delta/2}{2} = 0$

Variance $\sigma_e^2 = \dfrac{\left((+\delta/2) - (-\delta/2)\right)^2}{12} = \dfrac{\delta^2}{12}$ ⇒ Power of noise.

* Signal to Quantization Noise Ratio :-



→ Based on statistical model

We evaluate the effect of additive quantization noise $e[n]$ on o/p signal $x[n]$ by computing signal to noise Ratio ($SNR_{A/D}$) (or) signal to quantization noise ratio in dB defined by

A statistical model of A/D Quantizer

$$SNR_{A/D} = 10 \log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) \cdot dB \rightarrow (a)$$

σ_x^2 is o/p signal variance representing signal power

σ_e^2 is noise variance representing quantization noise power

In case of bipolar (b+1) bit A/D converter

$$\delta = 2^{-(b+1)} R_{fs} \text{ and hence}$$

$$\sigma_e^2 = \frac{2^{-2b}(R_{fs})^2}{48} \rightarrow (b)$$

Substituting Eq (b) in Eq (a) we get

$$SNR_{A/D} = 10 \log_{10}\left(\frac{48\sigma_x^2}{2^{-2b}(R_{fs})^2}\right) \rightarrow (c)$$

$$= 6.02b + 16.81 - 20 \log_{10}\left(\frac{R_{fs}}{\sigma_x}\right) \cdot dB \rightarrow (d)$$

∴ Eq (d) is used to determine the minimum A/D converter word length needed to meet a specified signal to quantization noise ratio

$$(SNR)_{A/D} = 6.02b + 16.81 - 20 \log_{10}(k) \rightarrow (e)$$

where $k = \frac{R_{fs}}{\sigma_x}$

5) Finite word length effect in IIR digital filters:-

The realisation of digital filters in hardware (or) software has some limitations due to finite word length of registers that are available to store these filter co-efficients.

∵ co-efficients stored in these registers are either truncated/Rounded off the system is realised using these co-efficients in not accurate the system may have different frequency response than the desired one.

There are 2- approaches for analysis and synthesis of digital filters with quantized co-efficients.

1st approach the digital filter is assumed to be an ideal one and represented by a transfer function in parallel with ideal filter.

2nd approach, Each filter is studied seperately and the quantized co-efficients can be optimised to minimise the maximum weighted difference b/w the ideal and actual frequency response.

let the transfer function of co-efficients quantised digital filter being realised

$$H(z) = \frac{\sum_{k=0}^{M} b_k \cdot z^{-k}}{\sum_{k=0}^{N} a_k \cdot z^{-k}} \rightarrow (1)$$

If $\bar{a}_k$ & $\bar{b}_k$ are filter co-efficients of unquantized filter & α_k and β_k are the error quantities then the filter co-efficients of quantised filter can be

$$a_k = \bar{a}_k + \alpha_k$$

$$b_k = \bar{b}_k + \beta_k$$

that error due to quantisation of filter co-efficients is

$$e(n) = y'(n) - y(n)$$

where $y(n)$ = o/p of ideal filter

$y'(n)$ = " " actual "

when 2-bit numbers are multiplied the product is 2B bit long This product must be rounded to B bits in all digital processing applications

the o/p of a finite word length multiplier can be expressed as

$$Q[\alpha_i x(n)] = \alpha_i x(n) + e(n) \longrightarrow (2)$$

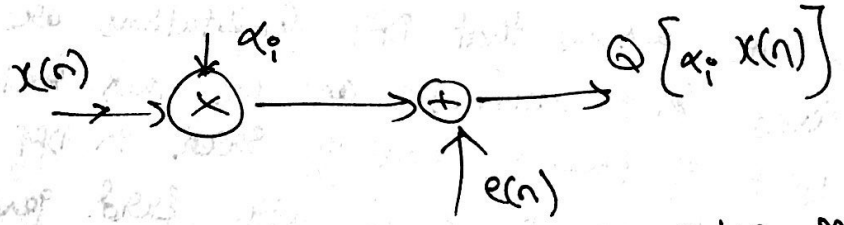


fig :- Product Quantisation noise model

where $\alpha_i x(n)$ is the product which is 2B bits long & e(n) is the error resulting from rounding the product to B-bits

The analysis of product round of noise is similar to A/D Converter noise

The avg Power (variance) is given by

$$\sigma_e^2 = \frac{2^{-2B}}{12} \longrightarrow (3)$$

6) Finite wordlength Effect in FFT algorithms:-

(or)

Roundoff Errors in FFT algorithms:-

Since FFT is often employed in a number of digital signal processing applications.

and it is interest to analyze the effect of finite word lengths in FFT wordlengths in FFT Computations.

The most critical error in the Computation is that due to arithmatic round off errors

As earlier sections we assume that DFT Computations are being carried out using fixed point arithmatic and we thus restrict our analysis to the effect of product round off error in DFT Computation via FFT algorithms and to compute them with error generated in DFT Computation.

→ If DFT computation requires K-complex multiplications, there are 4K sources of quantization errors in the computational structure

Assumptions:-

(a) All 4K errors are uncorrelated with each other and uncorrelated with ip sequence.

(b) The quantization errors are random variables uniformly distributed with a variance $\sigma_0^2 = \frac{2^{-2b}}{12}$ assuming a signed b-bit fractional fixed point arithmatic.

(a) Direct form DFT Computation:-

Recall that the N-Point DFT $X[k]$ of length N Complex sequence $x[n]$ is

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{nk}, \quad 0 \le k \le N-1 \rightarrow (a)$$

The Computation of a single DFT sample requires N-Complex multiplications and hence total no:- of real multiplications for the Computation of a single DFT sample is 4N.

The 4N Quantization error source. The variance of the error in Computation of one DFT sample is

$$\therefore \sigma_f^2 = 4N\sigma_0^2 = \frac{2^{-2b} N}{3} \rightarrow (b)$$

indicating o/p round off error is proportional to the DFT length.

now the i/p sequence $x[n]$ must be scaled to avoid overflow in Computation of $X[k]$

$$|X[k]| \le \sum_{n=0}^{N-1} |x(n)| \le N.$$

assuming that i/p range samples satisfy the dynamic range Constraint $|x[n]| \le 1$ to prevent overflow we need to ensure

$$|X[k]| < 1$$

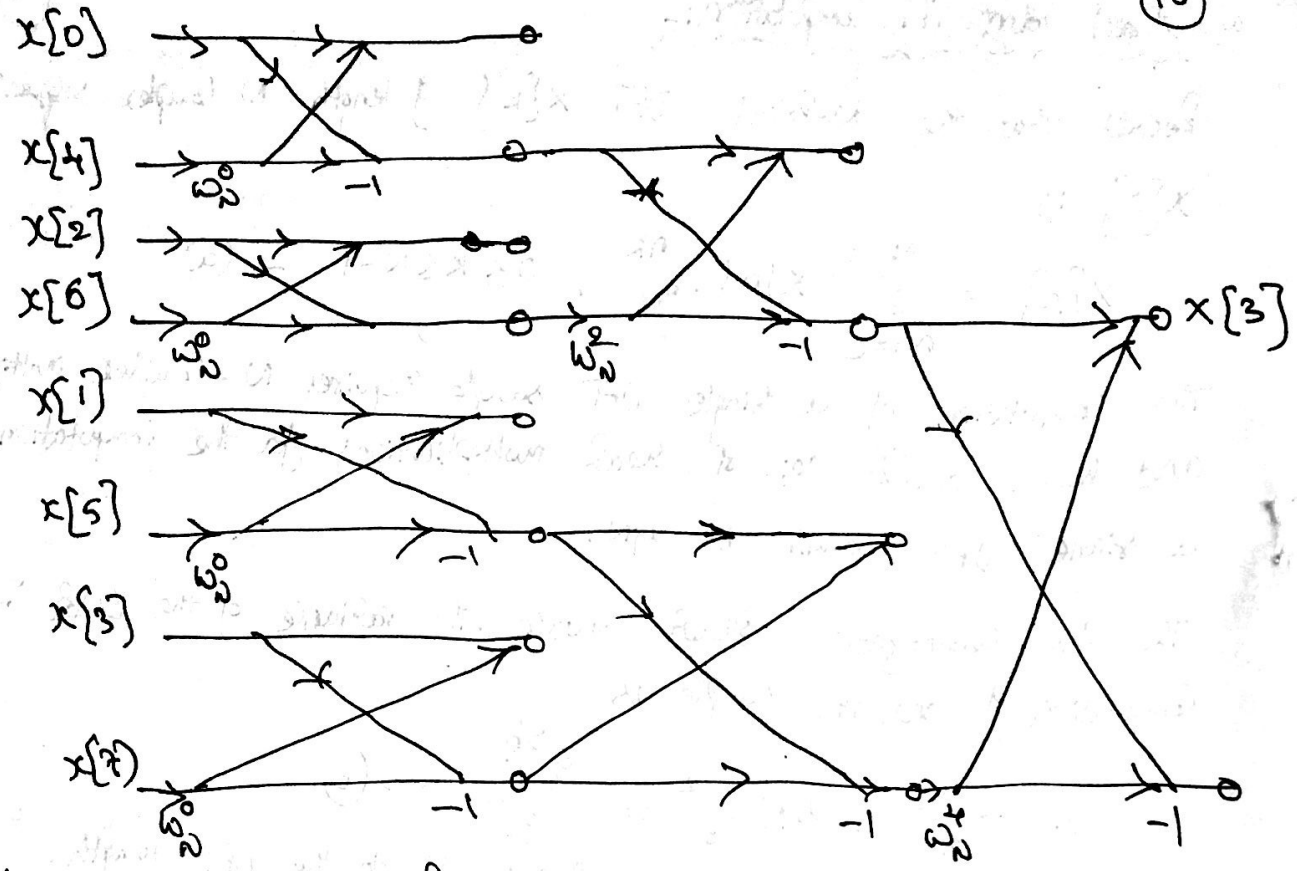
which can be guaranteed by dividing Each sample $x[n]$ in the i/p sequence by N.

The i/p signal Power is given by

$$\sigma_x^2 = \frac{(2/N)^2}{12} = \frac{1}{3N^2}$$

The Corresponding o/p signal Power is

$$\sigma_x^2 = N\sigma_x^2 = \frac{1}{3N}$$



As a result signal to noise ratio is

$$SNR = \frac{\sigma_x^2}{\sigma_y^2} = \frac{2^{2b}}{N^2}$$

∴ Above expression indicates SNR has been reduced by a factor of N^2 due to scaling and round off error.

(*) DFT Computation via FFT algorithm:-

FFT is used in large no:- of applications. Hence it is necessary to analyze the effect due to finite word length in ~~necessary to~~ FFT algorithms

Let σ_x^2 represents the variance o/p DFT Co-efficients $|X(k)|$

for N-point DFT 'σ' is given as

$$\sigma_x^2 = \frac{1}{3N} \longrightarrow (1)$$

for direct computation of DFT the variance of quantization error in multiplications is

$$\sigma_q^2 = \frac{N}{3} \cdot A^2 \longrightarrow (2)$$

Hence σ_q^2 is variance of quantization error and

Δ is step size

$$\text{where } \Delta = 2^{-b} \rightarrow (3)$$

where b is no: of bits represents level.

$\therefore$ Eq (2) becomes

$$\sigma_q^2 = \frac{N}{3} \cdot 2^{-2b} \rightarrow (4)$$

The signal to noise ratio of the o/p can be considered as measure of quantization errors.

The ratio of variance DFT co-efficients $\left(\sigma_x^2\right)$ to variance quantization error $\left(\sigma_q^2\right)$ is

$$\text{Signal to noise ratio direct computation of DFT} = \left(\frac{\sigma_x^2}{\sigma_q^2}\right)_{\text{Direct DFT}}$$

from Eq (1) & (2) we have

$$\left(\frac{\sigma_x^2}{\sigma_q^2}\right)_{\text{direct DFT}} = \frac{\frac{1}{3N}}{\frac{N}{3} \cdot 2^{-2b}} = \frac{2^{2b}}{N^2} \rightarrow (5)$$

when DFT is computed using FFT algorithms the variance of signal remains same.

$$\sigma_x^2 = \frac{1}{3N} \text{ from Eq (1)} \rightarrow (6)$$

But with algorithm the variance of the quantization error is

$$\sigma_q^2 = \frac{2}{3} \cdot 2^{-2b} \rightarrow (7)$$

Hence signal to noise ratio in FFT algorithm is

$$\left(\frac{\sigma_x^2}{\sigma_q^2}\right)_{FFT} = \frac{\frac{1}{3N}}{\frac{2}{3} \cdot 2^{-2b}} = \frac{2^{2b}}{2N} \text{ //.}$$