FIXED POINT AND FLOATING POINT REPRESENTATION

Finite Word length Effects:

- In the design of FIR Filters, The filter coefficients are determined by the system transfer functions. These filter co-efficient are quantized/truncated while implementing DSP System because of finite length registers.
- Only Finite numbers of bits are used to perform arithmetic operations. Typical word length is 16 bits, 24 bits, 32 bits etc.
- This finite word length introduces an error which can affect the performance of the DSP system.
- The main errors are
  1. Input quantization error
  2. Co-efficient quantization error
  3. Overflow & round off error (Product Quantization error)
- The effect of error introduced by a signal process depend upon number of factors including the
  1. Type of arithmetic
  2. Quality of input signal
  3. Type of algorithm implemented

1. Input quantization error

- The conversion of continuous-time input signal into digital value produces an error which is known as input quantization error.
- This error arises due to the representation of the input signal by a fixed number of digits in A/D conversion process.

2. Co-efficient quantization error

- The filter coefficients are compared to infinite precision. If they are quantized the frequency response of the resulting filter may differ from the desired frequency response.
  i.e poles of the desired filter may change leading to instability.

3. **Product Quantization error**

- It arises at the output of the multiplier
- When a 'b' bit data is multiplied with another 'b' bit coefficient the product ('2b' bits) should be stored in 'b' bits register. The multiplier Output must be rounded or truncated to 'b' bits. This known as overflow and round off error.

****

Types of number representation:

There are two common forms that are used to represent the numbers in a digital or any other digital hardware.

1. Fixed point representation
2. Floating point representation

*** Explain the various formulas of the fixed point representation of binary numbers.**

1. Fixed point representation

- In the fixed point arithmetic, the position of the binary point is fixed. The bit to the right represents the fractional part of the number and to those to the left represents the integer part.
- For example, the binary number 01.1100 has the value 1.75 in decimal.

  $(0*2^{1)} + (1*2^{0}) + (1*2^{-1}) + (1*2^{-2}) + (0*2^{-3}) = 1.75$

  In general, we can represent the fixed point number 'N' to any desired accuracy by the series

$$N = \sum_{i=n_i}^{n_2} C_i r^i$$

Where, r is called as radix.

- If r=10, the representation is known as decimal representation having numbers from 0 to 9. In this representation the number

$$30.285 = \sum_{i=-3}^{1_2} C_i 10^i$$

$$= (3*10^1) + (0*10^0) + (2*10^{-1}) + (8*10^{-2}) + (5*10^{-3})$$

- If r=2, the representation is known as binary representation with two numbers 0 to 1.
- For example, the binary number

$$110.010 = (1*2^2) + (1*2^1) + (0*2^0) + (0*2^{-1}) + (1*2^{-2}) + (0*2^{-3}) = 6.25$$

Examples:

Convert the decimal number 30.275 to binary form

2	30	
2	15	--0
2	7	--1
2	3	--1
	1	--1

0.55 * 2	➔1.10	➔1
0.10 * 2	➔0.20	➔0
0.20 * 2	➔0.40	➔0
0.40 * 2	➔0.80	➔0
0.80 * 2	➔1.60	➔1
0.60 * 2	➔1.20	➔1
0.20 * 2	➔0.40	➔0

$(30.275)_{10}$ = $(11110.01000110)_2$

0.275 * 2 ➔0.55 ➔0

In fixed point arithmetic =, the negative numbers are represented by 3 forms.

1. Sign-magnitude form
2. One's complement form
3. Two's complement form

1.1 Sign-magnitude form:

- Here an additional bit called sign bit is added as MSB.
  - If this bit is zero → It is a positive number
  - If this bit is one → It is a positive number
- For example
  - 1.75 is represented as 01.110000.
  - -1.75 is represented as 11.110000

1.2 One's complement form:

- Here the positive number is represented same as that in sign magnitude form.
- But the negative number is obtained by complementing all the bits of the positive number
- For eg: the decimal number -0.875 can be represented as
  - $(0.875)_{10}=(0.111000)_2$
  - $(-0.875)_{10}=(1.000111)_2$

0.111000
↓↓↓↓↓↓ (Complement each bit)
1.000111

1.3 Two's complement form:

- Here the positive numbers are represented as same in sign magnitude and one's complement form.
- The negative numbers are obtained by complementing all the bits of the positive number and adding one to the least significant bit

$(0.875)_{10}=(0.111000)_2$

$\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow$ (Complement each bit)

1.000111

$+ \quad 1$

1.001000

$(-0.875)_{10}=(1.001000)_2$

Examples:

- **Find the sign magnitude, 1's complement, 2's complement for the given numbers.**

1. $-\frac{7}{32}$

2. $-\frac{7}{8}$

3. $+\frac{7}{8}$

1. $-\frac{7}{32}$

$0.21875 * 2 \rightarrow 0.43750 \rightarrow 0$

$0.43750 * 2 \rightarrow 0.87500 \rightarrow 0$

$0.87500 * 2 \rightarrow 1.750000 \rightarrow 1$

$0.75 * 2 \rightarrow 1.50 \rightarrow 1$

$0.50 * 2 \rightarrow 1.00 \rightarrow 1$

$-\frac{7}{32}=(-0.21875)_{10}=(1.00111)_2$

Sign magnitude form = 1.00111

1's complement form = 1.11000

2's complement form = 1.11001

2. $-\frac{7}{8}$

$0.875 * 2 \rightarrow 1.75 \rightarrow 1$

$0.750 * 2 \rightarrow 1.500 \rightarrow 1$

$0.500 * 2 \rightarrow 1.000 \rightarrow 1$

$-\frac{7}{8}=(-0.875)_{10}=(0.111)_2$

Sign magnitude form = 0.111

1's complement form = 1.000

2's complement form = 1.001

3. $+\frac{7}{8}$

Sign magnitude form = 0.111

1's complement form = 0.111

2's complement form = 0.111

Addition of two fixed point numbers:

- **Add $(0.5)_{10} + (0.125)_{10}$**

$(0.5)_{10} = (0.100)_2$

$(0.125)_{10} = (0.001)_2$

$(0.101)_2 = (0.625)_{10}$

- Addition of two fixed point numbers causes an overflow.

For example

$(0.100)_2$

$(0.101)_2$
$(1.001)_2 = (-0.125)_{10}$ in sign magnitude form

Subtraction of two fixed point numbers:

- Subtraction of two numbers can be easily performed easily by using two's complement representation.
- **Subtract 0.25 from 0.5**

0.25 * 2 ➔0.50 ➔0 Sign magnitude form = $(0.010)_2$
0.50 * 2 ➔1.00 ➔1 1's complement form = $(1.101)_2$
0.00 * 2 ➔0.00 ➔0 2's complement form = $(1.110)_2$

$(0.5)_{10}$ = $(0.100)_2$
$-(0.25)_{10}$ = $(1.110)_2$ ➔Two's complement of -0.25
$(10.010)_2$

Here the carry is generated after the addition. Neglect the carry bit to get the result in decimal.

$(0.010)_2 = (0.25)_{10}$

- **Subtract 0.5 from 0.25**

0.5 * 2 ➔1.00 ➔1 Sign magnitude form = $(0.100)_2$
0.00 * 2 ➔0.00 ➔0 1's complement form = $(1.011)_2$
0.00 * 2 ➔0.00 ➔0 2's complement form = $(1.100)_2$

$(0.25)_{10}$ = $(0.010)_2$
$-(0.5)_{10}$ = $(1.100)_2$
$(1.110)_2$

Here the carry is not generated after the addition. So the result is negative.

Multiplication in fixed point arithmetic:

- Here the sign magnitude components are separated.
- The magnitudes of the numbers are multiplied. Then the sign of the product is determined and applied to the result.
- In the fixed point arithmetic, multiplication of two fractions results in a fraction.
- For multiplications with fractions, overflow can never occur.
- Eg:

$(0.1001)_2 * (0.0011)_2 = (0.00011011)_2$

2. Floating point representation

- Here, a number 'x' is represented by

$X=M.r^e$

Where, M → Mantissa which requires a sign bit for representing positive number and negative numbers.

R → base (or) radix

e → exponent which require an additional and it may be either positive or negative.

  - For eg, 278 can be represented in floating point representation.

$278 = \frac{278 \text{ X } 1000}{1000} = 0.278*10_3$

0.278 → Mantissa (M)
10 → base (or) radix (r)
3 → exponents (e)

- Similarly, to represent a binary floating point number

$X=M.2^e$ in which the fractional part of a number should fall (or) lie in the range of 1/2 to 1.

$5 = \frac{5 \text{ X } 8}{8}$ $0.625 \text{ X} 2^3$

Mantissa (M) = 0.625
Base (or) radix (r) = 2
Exponent (e) = 3

- Some decimal numbers and their floating point representations are given below:

4.5 → $0.5625 \text{ X } 2^3$ $=0.1001 \text{ X } 2^{011}$
1.5 → $0.75 \text{ X } 2^1$ $=0.1100 \text{ X } 2^{001}$
6.5 → $0.8125 \text{ X } 2^3$ $=0.1100 \text{ X } 2^{011}$
0.625 → $0.625 \text{ X } 2^0$ $=0.1010 \text{ X } 2^{000}$

- Negative floating point numbers are generally represented by considering the mantissa as a fixed point number. The sign of the floating point number is obtained from the first bit of mantissa.
- To represent floating point in multiplication

  Consider $X_1 = M_1 r^{e_1}$

  $X_2 = M_2 r^{e_2}$

  $X_1 X_2 = (M_1 * M_2) r^{(e_1 + e_2)}$

Example

Given $X_1 = 3.5*10^{-12}$, $X_2 = 4.75*10^{6}$. Find the product $X_1 X_2$

X=(3.5 X 4.75) $10^{(-12+6)}$

= $(16.625)10^{-6}$ → in decimal

In binary: $(1.5)_{10}$ X $(1.25)_{10}$ = $(2^1 0.75)$ X $(2^1 0.625)$

= 2^{001} X 0.1100 X 2^{001} X 0.1010

= 2^{010} X 0.01111

Addition and subtraction:

- Here the exponent of a smaller number is adjusted until it matches the exponent of a larger number.
- Then, the mantissa are added or subtracted
- The resulting representation is rescaled so that its mantissa lies in the range 0.5 to 1.
- Eg: **Add $(3.0)_{10}$ & $(0.125)_{10}$**

  $(3.0)_{10}$ = 2^{010} X 0.1100 = r^{e_1} X M_1

  $(0.125)_{10}$ = 2^{000} X 0.0010 = r^{e_2} X M_2

  Now adjust e_2 Such that $e_1 = e_2$

  $(0.125)_{10}$ = 2^{010} X 0.0000100

  Addition → 2^{010} (0.110000 + 0.0000100) → 2^{010} X 0.110010

  Subraction → 2^{010} X 1.001101

Compare floating point with fixed point arithmetic.

Sl.No	Fixed point arithmetic	Floating point arithmetic
1	Fast operation	Slow operation
2	Relatively economical	More expensive because of costlier hardware
3	Small dynamic range	Increased Dynamic range
4	Round off errors occurs only for addition	Round off errors can occur with addition and multiplication
5	Overflow occur in addition	Overflow does not arise
6	Used in small computers	Used in large general purpose computers.

****