ECE4270
Fundamentals of DSP
Lecture 19

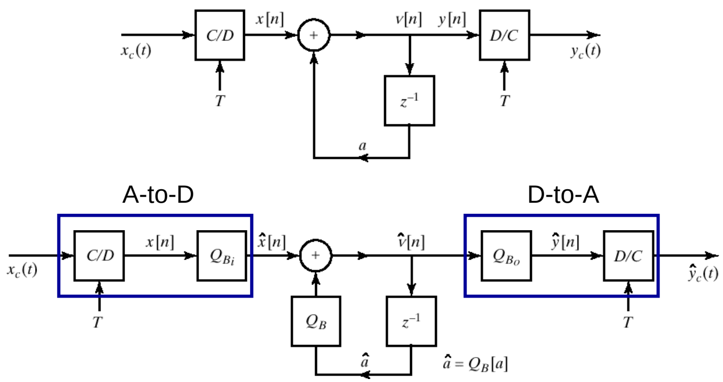
Fixed-Point Numbers and Arithmetic

School of Electrical and Computer Engineering
Center for Signal and Information Processing
Georgia Institute of Technology

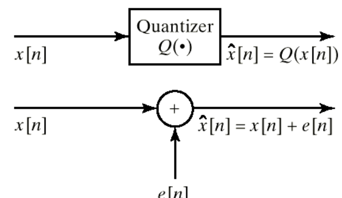
Overview of Lecture

- Quantization in LTI Implementation
- Two's-complement arithmetic
  - Integers and fractions
  - Scaling for fixed-point arithmetic
  - Quantizing filter coefficients
  - Addition & Multiplications

Quantization in LTI Implementation - I



Linear Noise Model



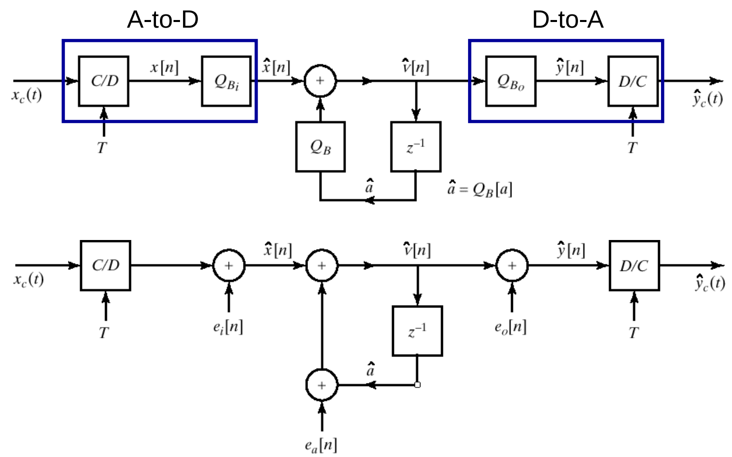
- Error is uncorrelated with the input.
- Error is uniformly distributed over the interval

$$-(\Delta/2) < e[n] \leq (\Delta/2).$$

- Error is stationary white noise, (i.e. flat spectrum)

$$P_e(\omega) = \sigma_e^2 = \frac{\Delta^2}{12}, \quad |\omega| \leq \pi$$

Quantization in LTI Implementation - II



FIR Filter Coefficients

Coefficient	Unquantized
$h[0] = h[27]$	1.359657×10^{-3}
$h[1] = h[26]$	-1.616993×10^{-3}
$h[2] = h[25]$	-7.738032×10^{-3}
$h[3] = h[24]$	-2.686841×10^{-3}
$h[4] = h[23]$	1.255246×10^{-2}
$h[5] = h[22]$	6.591530×10^{-3}
$h[6] = h[21]$	-2.217952×10^{-2}
$h[7] = h[20]$	-1.524663×10^{-2}
$h[8] = h[19]$	3.720668×10^{-2}
$h[9] = h[18]$	3.233332×10^{-2}
$h[10] = h[17]$	-6.537057×10^{-2}
$h[11] = h[16]$	-7.528754×10^{-2}
$h[12] = h[15]$	1.560970×10^{-1}
$h[13] = h[14]$	4.394094×10^{-1}

The condition for linear phase is

$$h[M-n] = \pm h[n]$$

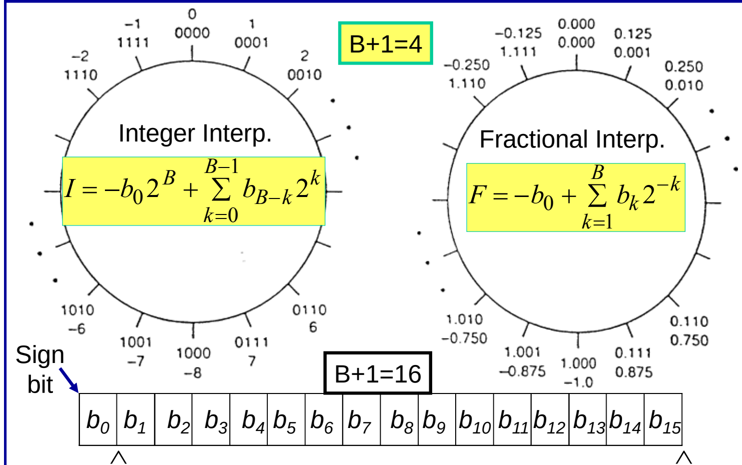
In this case,

$$\max\{h[k]\} \le 0.5$$

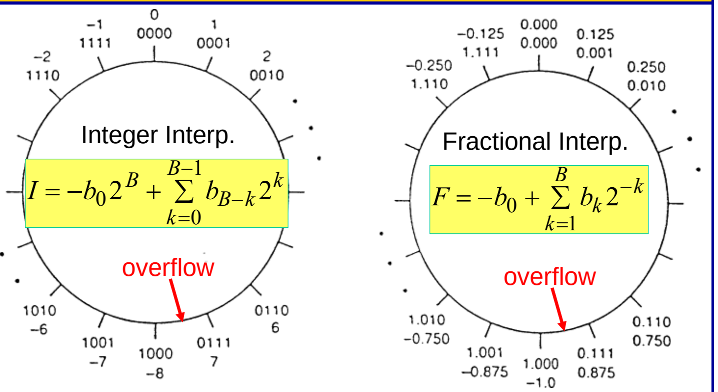
Fixed-Point Arithmetic in DSP Chips

- Numbers in fixed-point DSPs are represented as two's complement numbers. (16 bits in TI chips)
- Although the processor deals implicitly with signed two's complement integers, filter coefficients have fractional parts. Representation of these fractions must be built into the program.
- Proper scaling of the signals and coefficients is required to maintain precision while avoiding overflow. Therefore, the programmer must constantly worry about the following: scaling, quantization (roundoff noise), and overflow.

Two's Complement Numbers



Two's Complement Addition



Adding moves clockwise and subtracting moves counter clockwise around the circle. Overflow wraps around.

Fixed-Point Scaling - I

- 16-bit two's complement integers range in size from -32768_{10} to $+32767_{10}$.
- Q notation is a convenient way for a programmer to keep track of the binary point when representing fractional numbers by integers. Consider a fractional number a such that $-1 \le a < 1$, then we can represent a by a Q15 integer A such that $-32768 \le A \le 32767$. The relationship between a and A is simply

$$a = A \times 2^{-15} \quad \text{or} \quad A = a \times 2^{15}$$

Fixed-Point Scaling - II

- Consider a fractional number a such that $-1 \le a < 1$, then we can represent a by a Q15 integer A such that $-32768 \le A \le 32767$. The relationship between a and A is

$$a = A \times 2^{-15} \quad \text{or} \quad A = a \times 2^{15}$$

- For example

$$a = 0.75 \quad \Leftrightarrow \quad A = 24576_{10} \text{ Q15}$$

$$0_{\wedge}\underbrace{110000000000000}_{15 \text{ bits}}$$

Q15 means 15 bits to the right of the binary point.

Fixed-Point Scaling - III

- Consider a mixed number a such that $-4 \le a < 4$ Then we can represent a by a Q13 integer A such that $-32768 \le A \le 32767$. The relationship between a and A is

$$a = A \times 2^{-13} \quad \text{or} \quad A = a \times 2^{13}$$

- For example

$$a = 3.5 \quad \Leftrightarrow \quad A = 28672_{10} \text{ Q13}$$

$$011_{\wedge}\underbrace{1000000000000}_{13 \text{ bits}}$$

Fixed-Point Scaling - IV

- The smallest number that can be represented by a Q15 number is

$$\Delta = \frac{\text{range}}{\text{number of possibilities}} = \frac{2}{2^{16}} = \frac{1}{2^{15}}$$

- In general, the smallest number representable as a QB number will be

$$\Delta = \frac{1}{2^B}$$

- That is, in a QB number, the least significant bit (LSB) has value $\boxed{\Delta = \frac{1}{2^B}}$

Example of Quantizing Coefficients

```
% Multiply the coefficient by 2^15
  »a=-.001359657;  A=a*2^(15)
  A =
   -44.55324057600000
% Round (or truncate) the result
  »Ahat=round(A)
  Ahat =
     -45
% The equivalent quantized fraction is therefore
  »ahat=Ahat/2^(15)
  ahat =
    -0.00137329101562
```


Quantized FIR Filter Coefficients

Coefficient	Unquantized	Q15	Q12
$h[0] = h[27]$	1.359657×10^{-3}	45×2^{-15}	6×2^{-12}
$h[1] = h[26]$	-1.616993×10^{-3}	-53×2^{-15}	-7×2^{-12}
$h[2] = h[25]$	-7.738032×10^{-3}	-254×2^{-15}	-32×2^{-12}
$h[3] = h[24]$	-2.686841×10^{-3}	-88×2^{-15}	-11×2^{-12}
$h[4] = h[23]$	1.255246×10^{-2}	411×2^{-15}	51×2^{-12}
$h[5] = h[22]$	6.591530×10^{-3}	216×2^{-15}	27×2^{-12}
$h[6] = h[21]$	-2.217952×10^{-2}	-727×2^{-15}	-91×2^{-12}
$h[7] = h[20]$	-1.524663×10^{-2}	-500×2^{-15}	-62×2^{-12}
$h[8] = h[19]$	3.720668×10^{-2}	1219×2^{-15}	152×2^{-12}
$h[9] = h[18]$	3.233332×10^{-2}	1059×2^{-15}	132×2^{-12}
$h[10] = h[17]$	-6.537057×10^{-2}	-2142×2^{-15}	-268×2^{-12}
$h[11] = h[16]$	-7.528754×10^{-2}	-2467×2^{-15}	-308×2^{-12}
$h[12] = h[15]$	1.560970×10^{-1}	5115×2^{-15}	639×2^{-12}
$h[13] = h[14]$	4.394094×10^{-1}	14399×2^{-15}	1800×2^{-12}

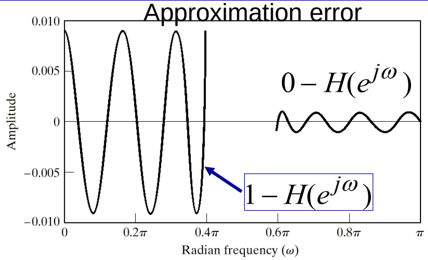
Quantized Filter Coefficients

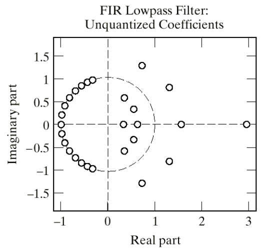
- For fixed-point implementation, the filter coefficients generally will be computed by a design algorithm that gives the filter coefficients as floating-point numbers. Therefore, they must be quantized to B+1 bits

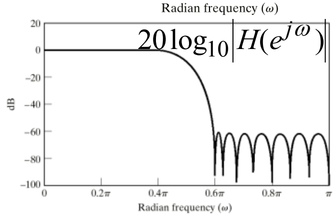
$$\hat{h}[n] = Q_B\{h[n]\} = h[n] + \Delta h[n]$$

$$\hat{H}(e^{j\omega}) = \sum_{n=0}^{M} (h[n] + \Delta h[n]) e^{-j\omega n}$$
$$= H(e^{j\omega}) + \sum_{n=0}^{M} \Delta h[n] e^{-j\omega n}$$

Unquantized FIR Filter Response (II)

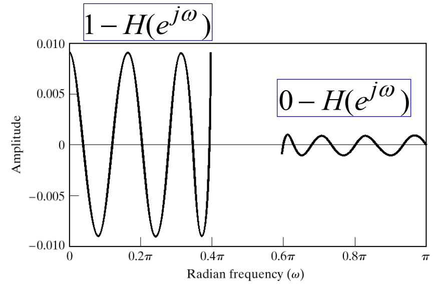


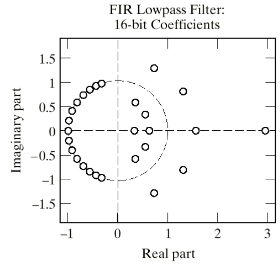




Note that zeros are in conjugate reciprocal groups (1,2, or 4). This is a basic property of FIR linear phase filters.

16-Bit Quantization of FIR Filter (III)

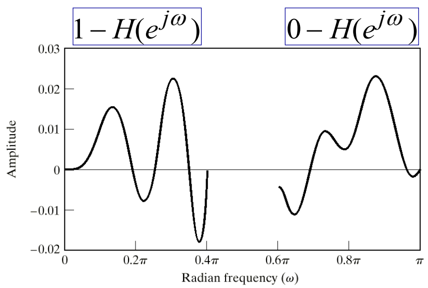


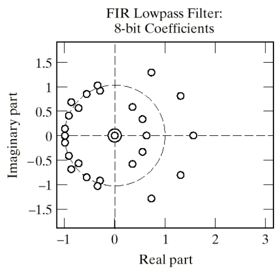


We preserve the linear phase property since

$$\boxed{\hat{h}[M-n] = \pm\hat{h}[n]}$$

8-Bit Quantization of FIR Filter (IV)





Zeros stay in reciprocal quads, but shift significantly due to the quantization.

2's Complement Numbers

- To change the sign of a 2's complement number, just complement all the bits, and add 1 to the least significant bit.

$$-6 = -(0110) = 1001 + 1 = 1010$$

- When accumulating 3 or more 2's complement numbers, the intermediate sums can overflow, but the final sum will be correct if it does not exceed the word length of the numbers.

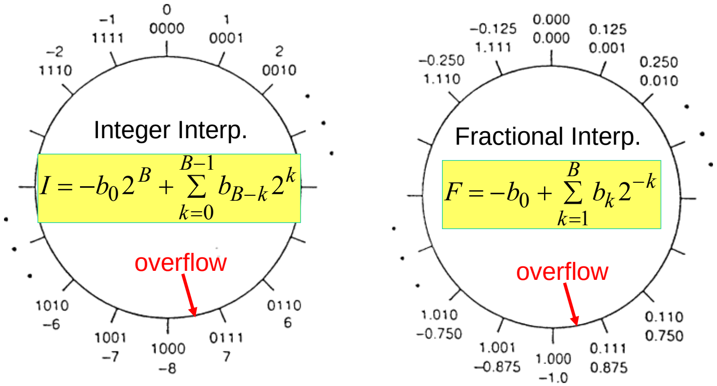
Final Sum correct

$$\underline{6+4} + (-6) = 10 + (-6) = 4$$

$$\underline{0110 + 0100} + 1010 = 1010 + 1010 = 0100$$

Partial sum overflows

Two's Complement Addition



Adding moves clockwise and subtracting moves counter clockwise around the circle. Overflow wraps around.

Adding

- When adding binary numbers, you must line up the binary points. This can be done by shifting one or the other of the numbers either left or right (multiply by power of 2).

lost in shift

$$
\begin{array}{rll}
00_\wedge 010000000000000000000000000000 & = 0.25 & Q30 \\
+\ 0001_\wedge 0010000000000000000000000000 & = 1.125 & Q28
\end{array}
$$

$$\Downarrow$$

$$
\begin{array}{rll}
0000_\wedge 0100000000000000000000000000 & = 0.25 & Q28 \\
+\ 0001_\wedge 0010000000000000000000000000 & = 1.125 & Q28 \\
\hline
0001_\wedge 0110000000000000000000000000 & = 1.375 & Q28
\end{array}
$$

Multiplication of 2's Comp. Numbers

- The product $A * X$ of two B+1 bit 2's complement numbers is a 2B+1 bit number. TI hardware gives a 2B+2 bit number with two sign bits. If you retain all bits, therefore, products cannot overflow.
- Let A and X be Q15 numbers. Then $A * X$ will be a Q30 number with the binary point moved right by one position. Example,

2 sign bits

$$
\begin{array}{rll}
0110000000000000 & = 0.75 & Q15 \\
*\ 0100000000000000 & = 0.5 & Q15 \\
\hline
\underbrace{00_\wedge 01100000000000}_{\text{16 most significant}}\ 0000000000000000 & = 0.375 & Q30
\end{array}
$$

Multiplication of 2's Comp. Numbers

- Let A and X be Q0 numbers (i.e., integers). Then $A * X$ also will be a Q0 number. Example,

$$
\begin{array}{rll}
0000000000010001 & = 17 & Q0 \\
*\ 1111111111111011 & = -5 & Q0 \\
\hline
1111111111111111\ \underbrace{1111111110101011}_{\text{16 least significant}} & = -85 & Q0
\end{array}
$$

- Let A and X be Q14 numbers. Then $A * X$ will be a Q28 number. Example,

$$
\begin{array}{rll}
0110000000000000 & = 1.5 & Q14 \\
*\ 0011000000000000 & = 0.75 & Q14 \\
\hline
0001_\wedge 0010000000000000000000000000 & = 1.125 & Q28
\end{array}
$$

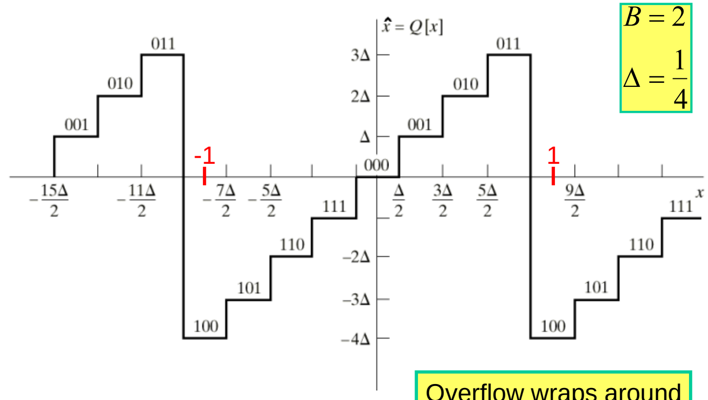
Overflow

- Overflow can occur in two ways:
  - – Accumulator overflows due to additions
  - – Taking 16 bits out of the wrong part of a product
- Overflow can be prevented by:
  - – More precision - use larger accumulator
  - – Saturation arithmetic - clip accumulator at largest value
  - – Shift products to the right to discard low-order bits. This results in loss of precision.

Two's Complement Quantizer

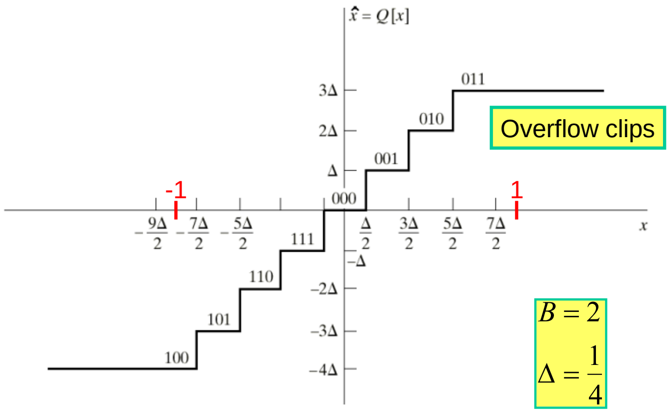
$B = 2$

$\Delta = \frac{1}{4}$



Overflow wraps around

Two's Complement Saturation



Overflow clips

$B = 2$

$\Delta = \frac{1}{4}$