EECS 598 VLSI for Wireless Comm. and Machine Learning

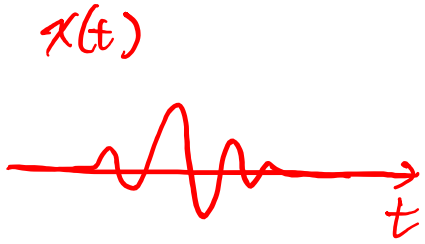
Discrete Signal, Quantization, Fixed Point

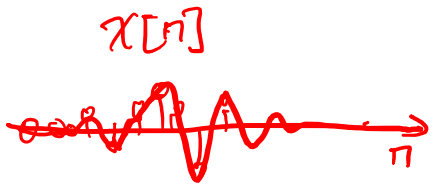
Prof. Hun-Seok Kim

hunseok@umich.edu

Continuous vs. Discrete Time Signal

- Digital signal processing systems deal with discrete signal
  - Notation: x(t) vs. x[n]
  - n could be time, pixel index, …
- Frequency domain representation (non-discrete/discrete Fourier transform)
  - Notation: X(f) vs. X[k]
  - k is frequency bin index



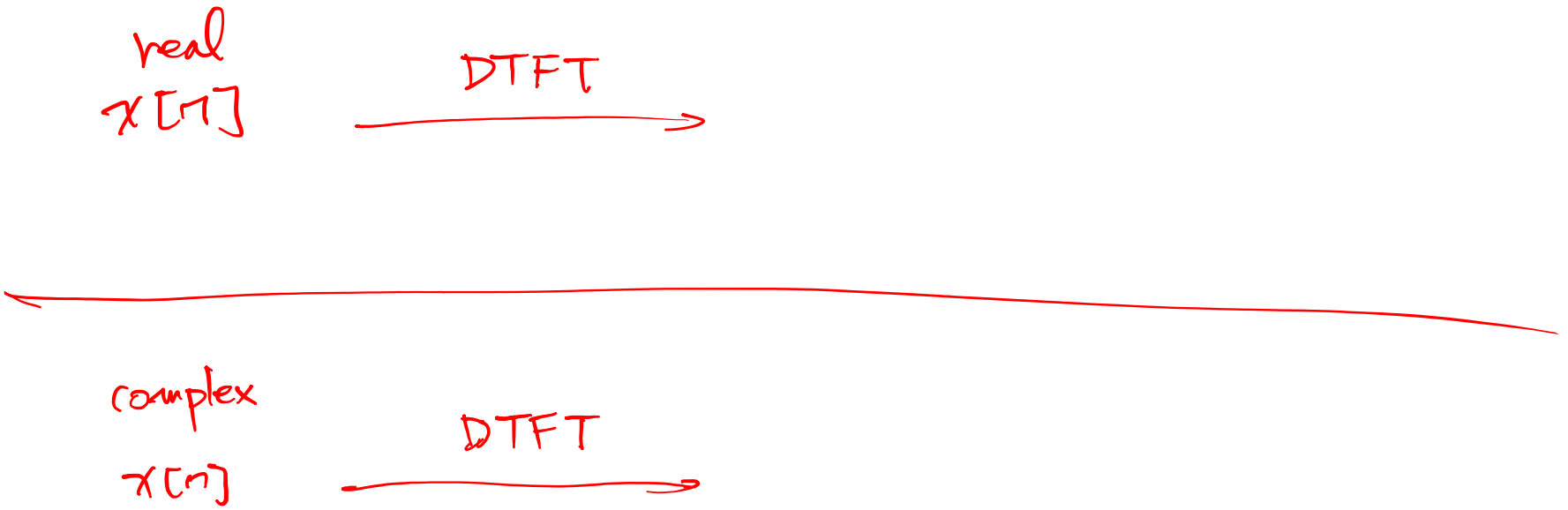


$$\text{DTFT}: \quad X(f) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j2\pi F n}$$

$$\text{DFT}: \quad X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi \frac{kn}{N}}$$

Real vs. Complex Signal

- Real signal is symmetric, complex signal is asymmetric in frequency domain
  - Real signals: audio, image
  - Complex signals: wireless communication, radar



Sampling and Reconstruction

$x(t) \xrightarrow{FT} X(F)$

$x[n] \xrightarrow{DTFT} X(f)$

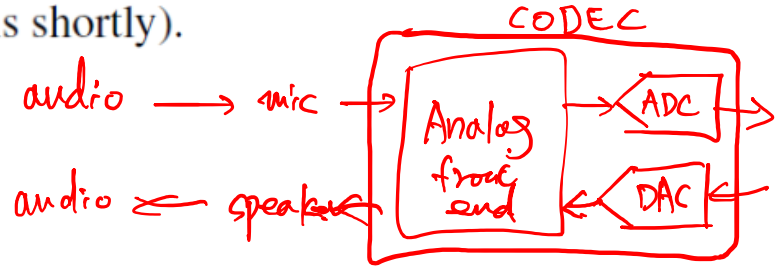
We continue to use the convention:
F denotes Herzian freq. and f denotes Digital freq.

Sampling is the part of the Analog-to-Digital Converter (ADC) that converts cts time signal $x(t)$ into discrete time signal $x[n] = x(nT_s)$. $F_s = 1/T_s$ is the *sampling rate* (samples/sec).

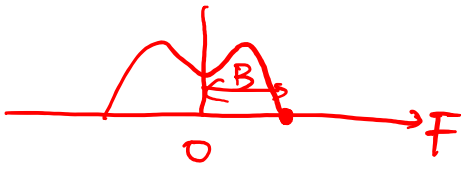
Reconstruction is the part of the Digital-to-Analog Converter (DAC) that converts discrete time signal $x[n] = x(nT_s)$ to cts time signal $x(t)$. $F_s = 1/T_s$ is the *conversion rate*.

Sampling and reconstruction are commonly combined into a ADC/DAC device called the CODEC (Coder-Decoder).

CODECs work well as long as $x(t)$ does not have significant energy at frequencies above $F_s/2$ Hz (more on this shortly).

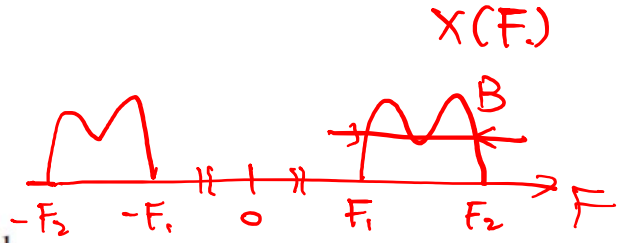


Sampling Band-limited Signals



Lowpass signal: Negligible energy ($X(F) = 0$) for all $|F| > B$. Single sided bandwidth is B Hz.

Sampling $x(t)$ at $F_s > 2B$ samples/sec allows us to exactly reconstruct. (Nyquist sampling theorem)



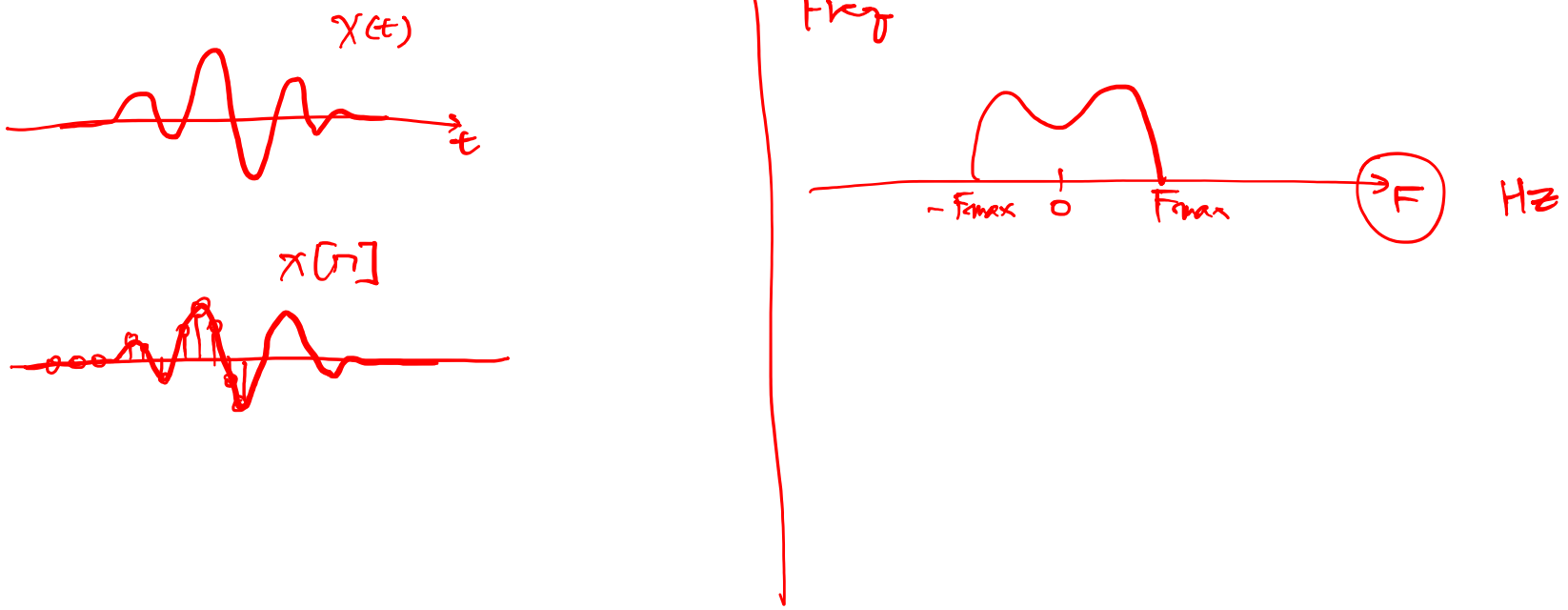
Bandpass signal: Negligible energy outside of a band, $B = F_2 - F_1$ not containing 0 Hz.

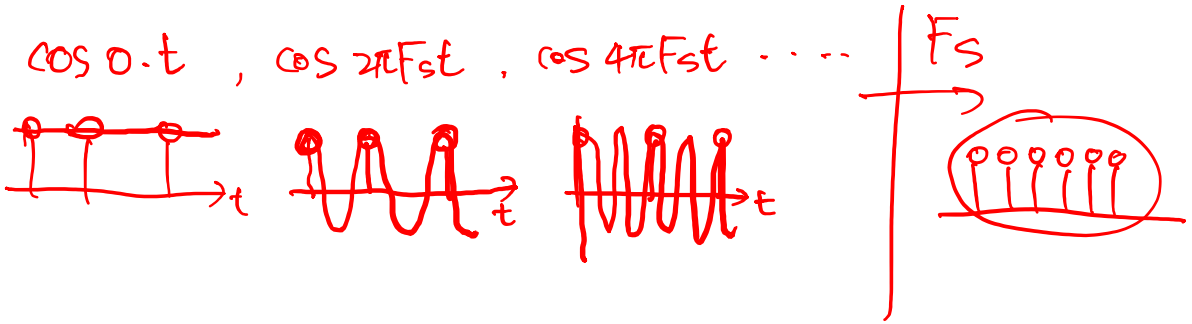
Sampling at $F_s > 2B$ allows us to exactly reconstruct. (Down-convert followed by Nyquist sampling, or using the Bandpass sampling theorem[a])

Note that for bandpass waveforms do not need $F_s > 2F_2$!

[a] see https://en.wikipedia.org/wiki/Undersampling

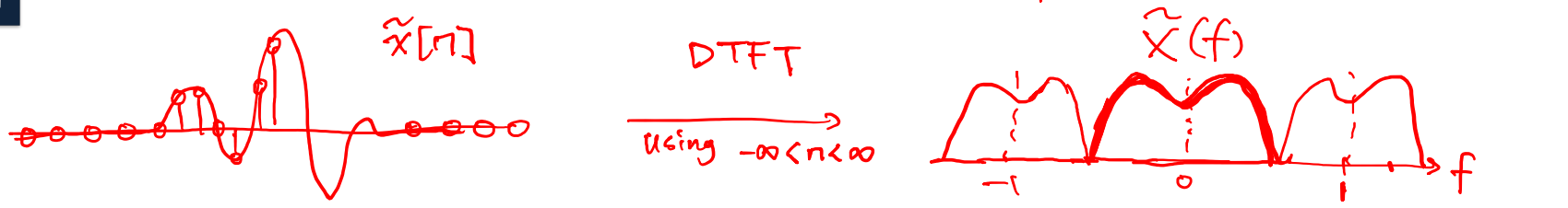
Sampling: Frequency domain view





DFT vs. DTFT

DFT : "sampled" version of DTFT



Comments on sampling

Given a real valued lowpass spectrum with bandwidth B the sample frequency equal to $2B$ is often called the *Nyquist* sample rate.

In practice one should sample at a rate of *at least* two or three times the Nyquist *rate*.

Common sample rates:

standard telephone system	8 kHz
wideband telecommunications	16 kHz
home music CDs	44.1 kHz
professional audio	48 kHz
DVD-Audio	192 kHz
instrumentation, RF, video	extremely fast

Sampling a bandpass signal $x(t)$ with frequency component limited to $[F_1, F_2]$, with $F_2 - F_1 = B$: The bandpass sampling theorem states that we only need to sample at frequency $F_s > 2B$, and not $> 2F_2$

Number Representation

- ADC and DAC have finite bit precision
- In many practical DSP applications, >16-bit precision is unnecessary
- Floating point computation is much more expensive compared to fixed point counterpart
  - Yes and No
- Many low power microcontrollers do not natively support floating point HW

Quantization

- DSP systems deal with ‘quantized’ signal
- Uniform vs. non-uniform quantization
  - Signal distribution determines quantization error statistics

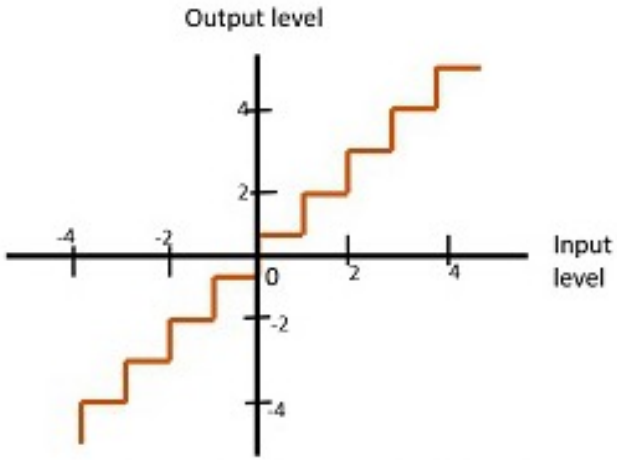


Fig 1 : Mid-Rise type Uniform Quantization

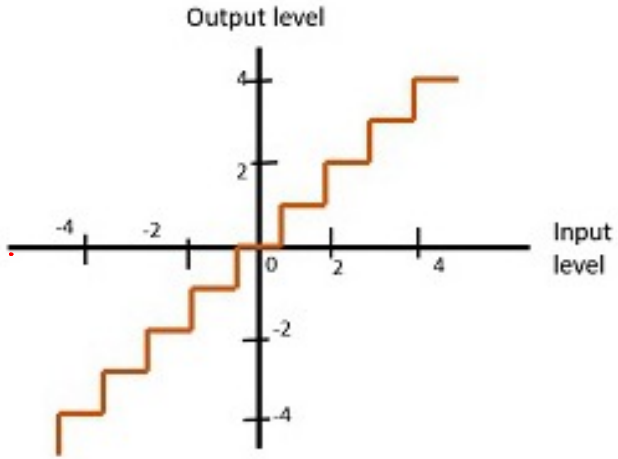
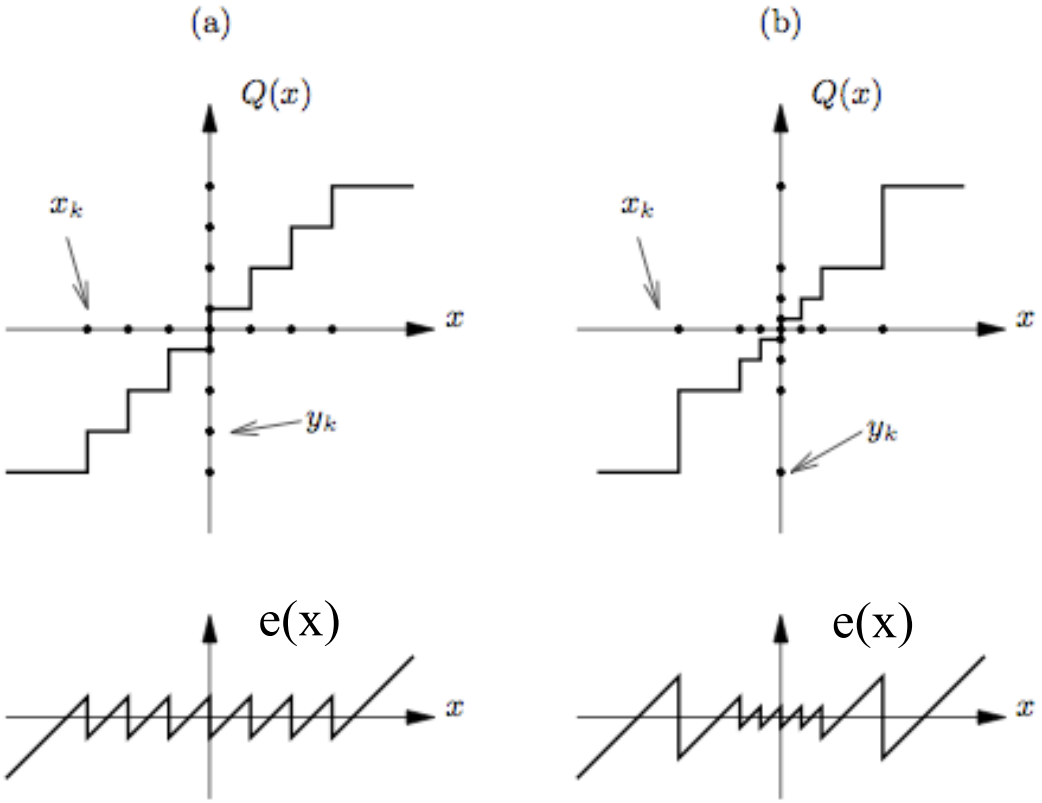


Fig 2 : Mid-Tread type Uniform Quantization

Quantization Error

$Q(x) = x + e(x)$

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



Uniform Quantization

- Signal range: $\pm V_{max}$
- # of quantization levels (bins) represented by b-bits: $N = 2^b$
- Uniform quantization step: $q = 2V_{max} / 2^b$
- Variance (power) of quantization error (noise) given input pdf p(x)

$$\sigma_e^2 = E\{(Q(x) - x)^2\} = \int_{-V_{max}}^{V_{max}} e(x)^2 p(x) dx$$

- If input signal is uniformly distributed, quantization error within a bin is also uniformly distributed: (-q/2, q/2]
- Variance of quantization error for <u>uniform input</u>

$$\sigma_e^2 = E\{(Q(x) - x)^2\} = \int_{-\frac{q}{2}}^{\frac{q}{2}} e^2 \frac{1}{q} de = \frac{q^2}{12}$$

Uniform Quantization

- Uniform quantization step: $q = 2V_{max} / 2^b$
- Variance (power) of quantization error

$$\sigma_e^2 = \frac{q^2}{12} = \frac{1}{3} 2^{-2b} V_{max}^2$$

- Signal power (uniform distribution)

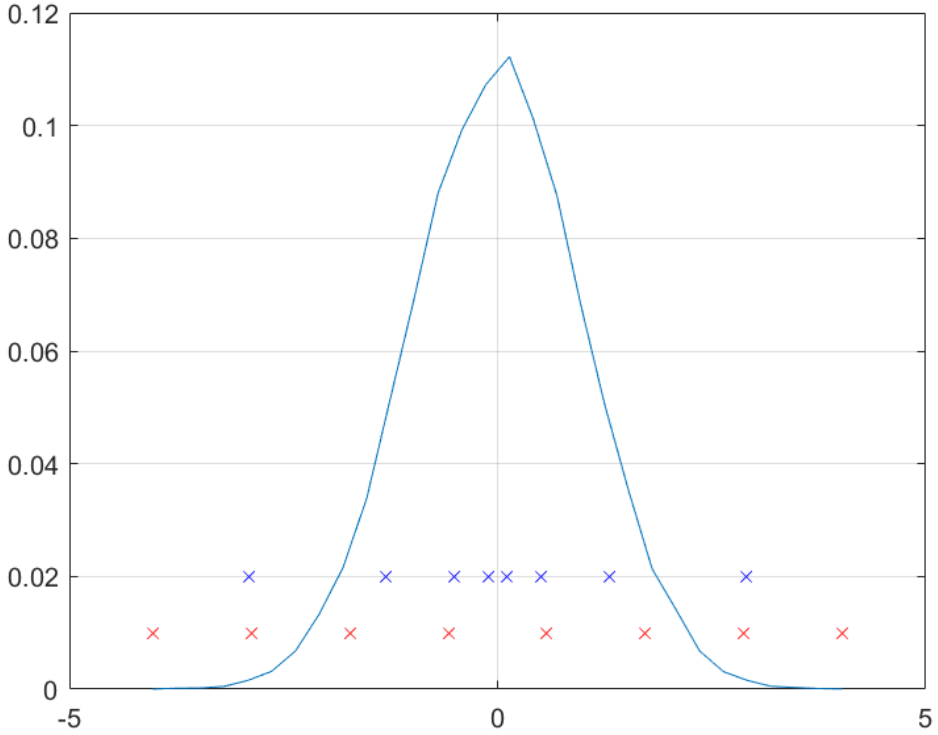
$$\sigma_s^2 = \int_{-V_{max}}^{V_{max}} x^2 p(x) dx = \int_{-V_{max}}^{V_{max}} x^2 \frac{1}{2V_{max}} dx = \frac{1}{3} V_{max}^2$$

- Signal to Quantization Noise Ratio (SQNR)
  - Signal power / quantization noise power $\sigma_s^2 / \sigma_e^2 = 2^{2b}$
  - In dB: $10 log_{10}(\sigma_s^2 / \sigma_e^2) = 10 log_{10}(2^{2b}) = 20b \; log_{10} 2 = 6.02b$
  - **6dB SQNR per bit**

Non-uniform Quantization

- Many signals do NOT have uniform distribution
- Exponential quantization

  Quantization center points: $Q(x) \in \{\pm c2^i\},\ i = \cdots, -2, -1, 0, 1, 2, \ldots$



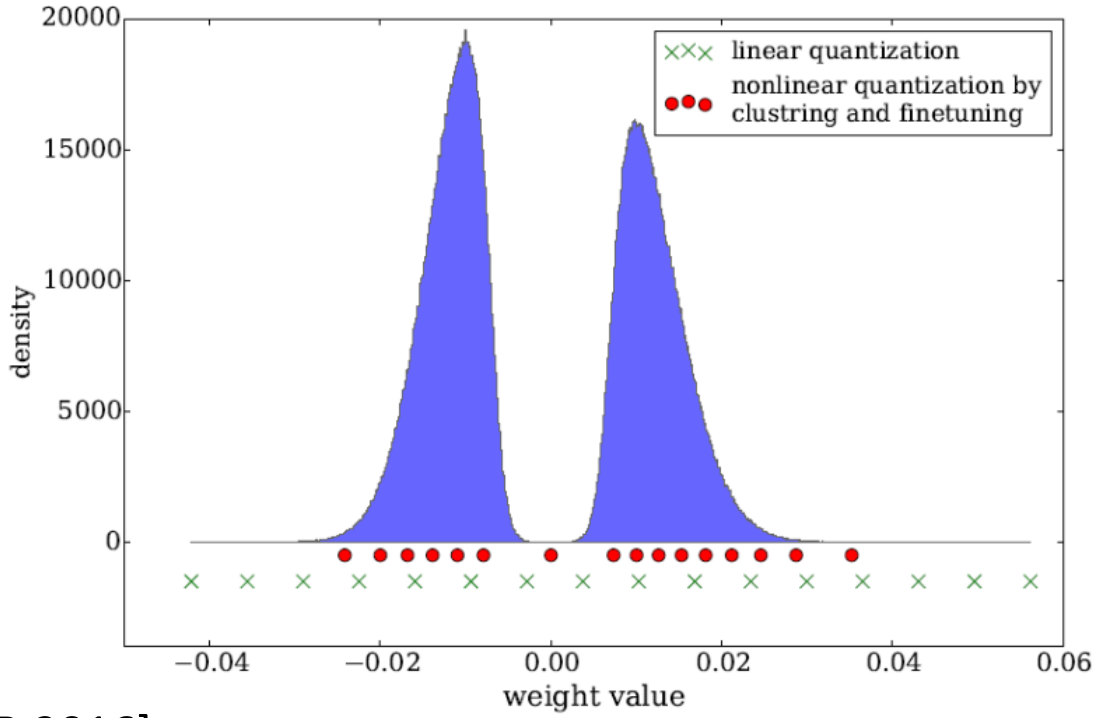
- MSE quantizer: $argmin_{Q(x)} \int_{-\infty}^{\infty} (Q(x) - x)^2 p(x) dx$
- How to find Q(x)?

How to determine quantization

- Factors to consider to determine a proper quantization scheme $Q(x)$
  - Input signal to noise ratio (SNR)
    - SQNR is desired to be (much) higher than input SNR
  - Peak-to-average power ratio (PAPR)
  - Input signal probability density function $p(x)$
    - Typical signals do not follow uniform distribution
  - Hardware complexity of quantization and arithmetic operations on quantized values
    - Uniform quantization
    - Non-uniform quantization
  - Quantization error variance $\int_{-\infty}^{\infty}(Q(x)-x)^2 p(x)dx$

Non-Uniform Quantization

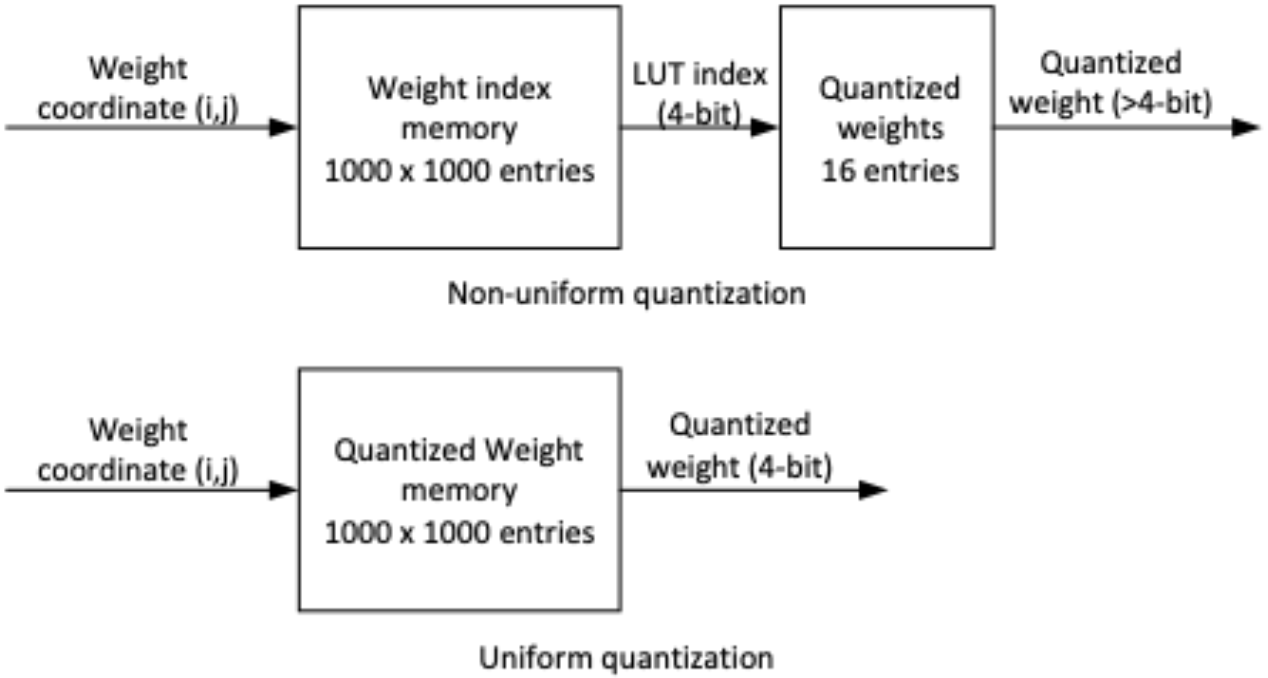
- Example: y = Wx where W dimension is 1000 x 1000
- How many bits needed to represent / store non-uniform quantized values for W?



[Han ICLR 2016]

LUT based non-uniform quantization

- Example: y = Wx where W dimension is 1000 x 1000
- How many bits needed to represent / store non-uniform quantized values for W?



Digital Number Representation: Fixed point vs. Floating point

- DSP deals with binary coded numbers
  - ADC output is binary coded
- How do we encode numbers in binary representations?
  - Fixed point (from ADC / to DAC)
  - Floating point
- How do we represent complex (with imaginary part) numbers in VLSI accelerators and processors?

Floating Point Representations

- Single precision (32-bit) floating point representation
- IEEE 754 standard binary representation: b[31:0]
  - b[31]: sign, s = -1 or 1
  - b[30:23]: stored exponent e, 0 <= e <= 255
  - b[22:0]: stored mantissa m, 0 <= m <= 2^{23}-1
- Range of single precision floating point: (-2^{128}, 2^{128})
- Finest resolution 2^{-149}
- Real number

	m = 0	m ≠ 0
e = 0	0	s x m x 2^{-149}
1<= e <= 254	s x (1.0 + m x 2^{-23}) x 2^{e-127}	
e = 255 (all ones)	s x infinity	NaN

Half-Precision Floating Point

- Half-precision 16-bit floating point (FP16)
  - Getting popular for DNN accelerators (available in some GPUs)
- IEEE 754 standard FP16 binary representation: b[15:0]
  - b[15]: sign, s = -1 or 1
  - b[14:10]: stored exponent e, 0 <= e <= 31
  - b[9:0]: stored mantissa 0 <= m <= 1023
- Range of single precision floating point: $(-2^{16}, 2^{16})$
- Finest resolution 2^{-24}
- Real number

	m = 0	m ≠ 0
e = 0	0	$s \times m \times 2^{-24}$
1<= e <= 30	$s \times (1.0 + m \times 2^{-10}) \times 2^{e-15}$	
e = 31 (all ones)	s x infinity	NaN

8-bit Floating Point (Minifloat)

- 8-bit floating point (FP8, Minifloat)
  - Getting popular for DNN accelerators (available in some neural processors)
- FP8 binary representation: b[7:0]
  - b[7]: sign, s = -1 or 1
  - b[6:3]: stored exponent e, 0 <= e <= 15
  - b[2:0]: stored mantissa 0 <= m <= 7
- Range of single precision floating point: $(-2^8, 2^8)$
- Finest resolution 2^{-9}
- Real number

	m = 0	m ≠ 0
e = 0	0	$s \times m \times 2^{-9}$
1<= e <= 14	$s \times (1.0 + m \times 2^{-3}) \times 2^{e-7}$	
e = 15 (all ones)	s x infinity	NaN

Fixed Point Representations

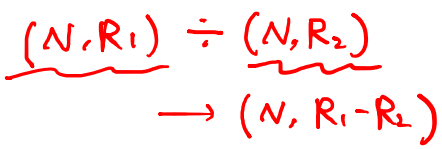
- Fixed point representation
  - Exponent is 'fixed' (no need to store it for each value): 2^{-R}
- Binary representation: b[N-1:0]
  - Stored mantissa: b[N-1:0] = m
  - m is a signed integer: $\mathbf{-2^{N-1} \leq m \leq 2^{N-1}-1}$
  - Real number = m x 2^{-R}
- Notation: (N,R) for N-bit, exponent is -R
  - 1.0 is represented by $m=2^R$
  - Example (16, 4): b[15:0] = m = -28 → real number?
    - Maximum? Minimum? Precision?
  - Example (8, 6): b[7:0] = m = 7 → real number?

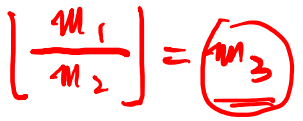
Floating Point vs. Fixed Point

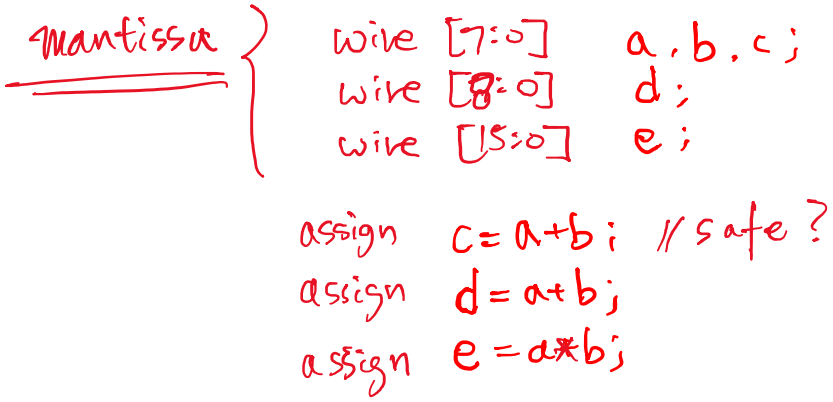
- What's floating / fixed?
- Fixed point can only represent integer values? (true/false)
- In DSP VLSI, fixed point is generally preferred (true/false)? Why?
- Recall uniform quantization SQNR is ~6b
  - How much SNR / SQNR needed in DSPs?
    - Audio, image
    - Wireless communication
    - Scientific data analysis

Fixed point math

- Operation with two numbers in different notations
  - Dynamic range growth after each operation
  - (N1,R) + (N2,R) → (max(N1, N2)+1,R)
  - (N,R) + (N,R) → (N+1,R)
    - Do not add two mantissa values when exponents are different!
    - Need to shift one value first to match exponent
  - (N,R) + (N,R) → (N,R): saturation necessary unless you know that the result won't overflow
  - (N1,R1) x (N2,R2) → (N1+N2, R1+R2)
  - (N1,R1) x (N2,R2) → (N3, R3)







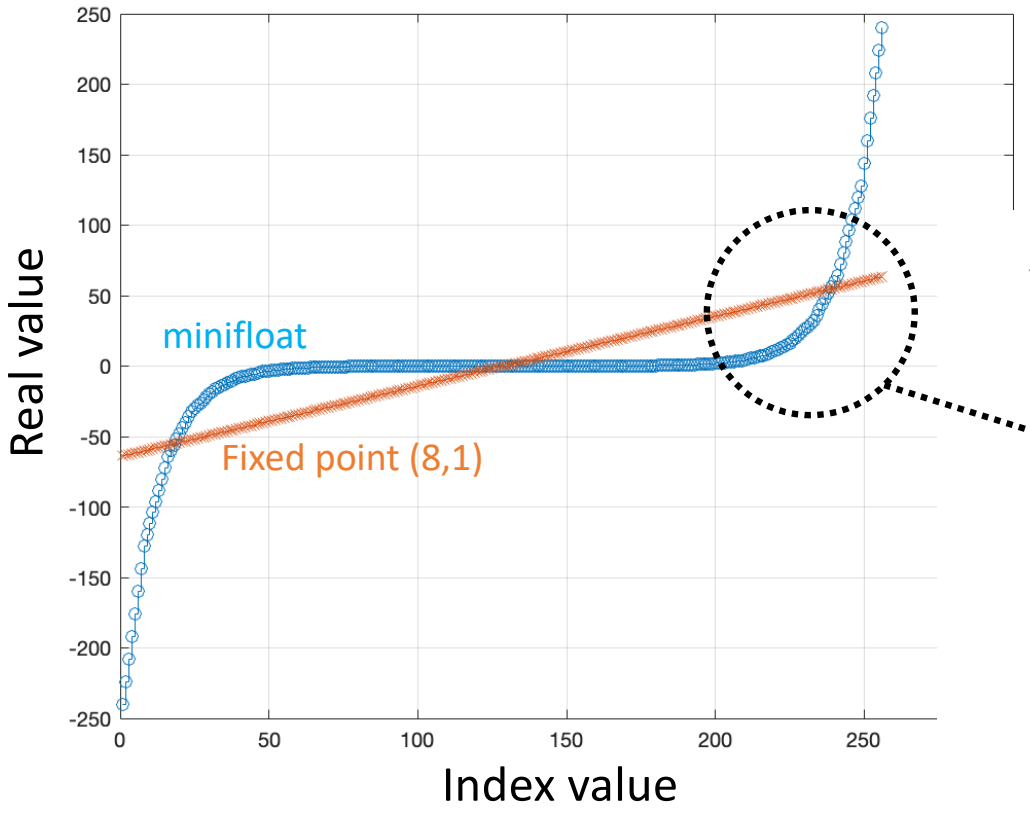
Fixed point math examples

- Example m = 36 in (8,4), m = -8 in (8,5)
  - +, -, x, / between these two numbers

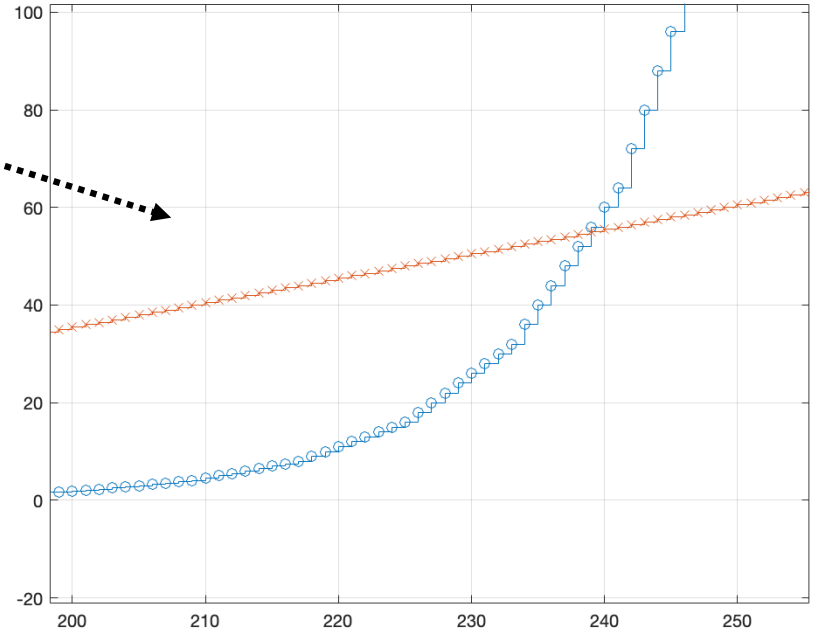
- (8,4) representation
  - Example: Real number 0.8125 x -0.4375 → Mantissa?

Floating point vs. Fixed point Tradeoff

- Minifloat (FP8) vs. fixed point (8,1)



- Both representations have 256 values
- Compared to (8,1) fixed point
  - Minifloat has wider dynamic range
  - Minifloat has finer resolution for smaller values but coarser resolution for larger values
  - Minifloat consumes more energy & circuit area for computing



Floating point vs. Fixed point Tradeoff

Floating point energy (pJ in 40nm process)		Fixed point energy (pJ in 40nm process)		Ratio
FP8 Mult	0.4	8bit Mult	0.2	**2**
FP16 Mult	1.1	16bit Mult	0.8	**1.37**
FP32 Mult	3.7	32bit Mult	3.1	**1.19**
FP8 Add	0.23	8bit Add	0.03	**7.6**
FP16 Add	0.4	16bit Add	0.05	**8**
FP32 Add	0.9	32bit Add	0.1	**9**

- Floating point multiplication consumes only 1.2 – 2x more energy than fixed point (why?)
  - Mantissa multiplication complexity vs. overhead in exponent handling
- Floating point add consumes much higher energy than fixed point add (why?)
  - Overhead in exponent handling
  - Floating point add energy is not negligible comparable to floating point mult
  - Fixed add is much cheaper than fixed point mult
  - In MAC (mult-and-accumulate), often a mult is followed by a wider bit-width add (e.g., 16-bit mult & 32-bit add)

Canonical Signed Digit Multiplication

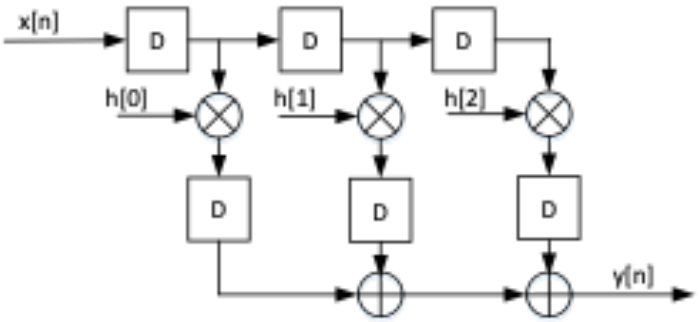
- Commonly used for a 'constant' multiplication
- CSD uses a ternary number system which uses 1, 0, −1 denoted by $1, 0, \bar{1}$
- Multiplication consists of minimal number of add/sub and shifts
- For example, the CSD representation of 30 is $100\bar{1}0$ = 32 - 2
  - X * 19 = ?
  - X * 23 = ?
- In a CSD number, two consecutive digits cannot be non-zero
- There are a number of algorithms to convert a binary number to CSD
- Modern Verilog HDL compliers convert constant numbers to CSD for multiplication automatically

Example: Direct Digital Synthesis (DDS) for Arbitrary Sine Wave Generator

- Q: How to you make a sine wave generator with an arbitrary (user programmable) frequency?

FIR Filter Example

- FIR: y[n] = x[n] * h[n] = $\sum_{k=0}^{N-1} x[n-k]h[k]$
- N=3-tap filter has h[0] = 0.27, h[1] = 0.46, h[2] = 0.31



- Design FIR in fixed point with below spec and compare the output with floating point
  - Example fixed point design
    - x[n] is quantized with (10,6)
    - h[n] is quantized with (5, 4)
    - Multiplication output is truncated to (10, 5)
    - Each sum output is truncated to (10, 4)