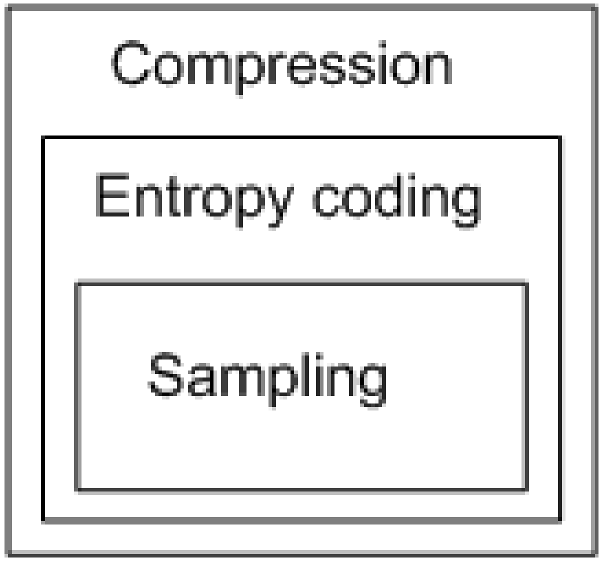
DSP Basics

- Digital data representation
  - Sampling theorem
  - Quantization and number systems
  - Quantization errors
- Entropy coding (Huffman coding)
- Spectrum analysis

Coding Strategies

- Compression coding – to minimize the required resources (downloading time, frequency bandwidth, memory space) for the application
- Entropy coding – to minimize the number of bits to represent **a sequence of symbols**
- Sampling and quantization – minimize the number of bits for **a sample** (symbol)



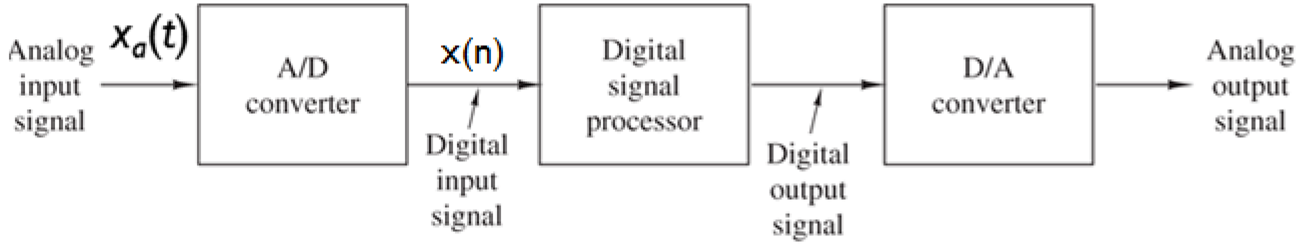
- Compression coding needs extensive signal processing to identify the information to be ignored for human application
- Coding for other purpose, i.e., reliability of transmission or disk writing/reading, is also needed.

Digital Signal Representation

- Most practical signals are analogue, and must be converted into a discrete representation before being processed
- Reconstructing the analogue signals from the processed samples is also necessary for the end user application
- Sampling is the first step of processing the signals for almost all applications, and these sample values are represented by either a fixed-point or a floating-point format
- Finite word length has effects on the resolution of sample representation and give the **quantization noise**.
- For audio compression, we would like to reduce the number of bits representing the samples and keep **the noise effects to be inaudible**.
  - For audio applications, high resolution is particularly important for small amplitude signals

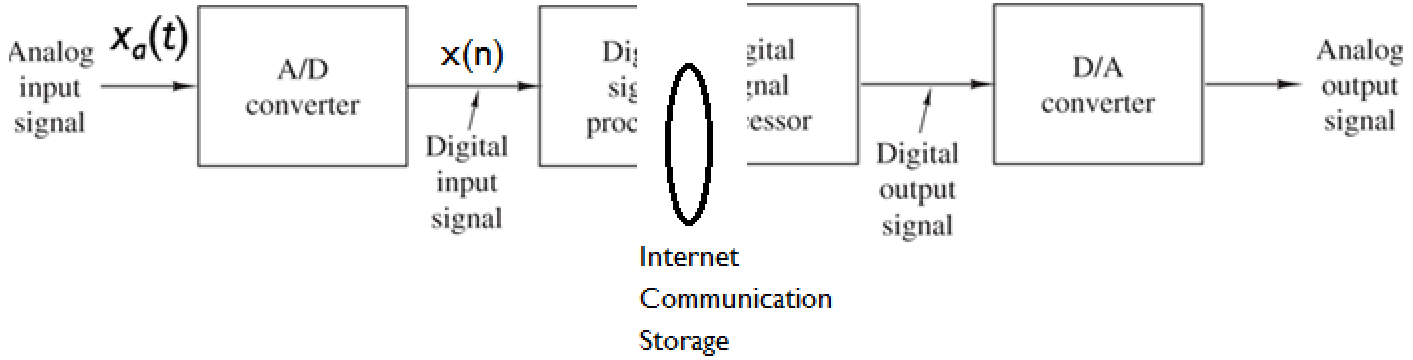
Digital Signal Representation

- Note the difference of the two general structures for practical applications below:



Block diagram of a digital signal processing system.

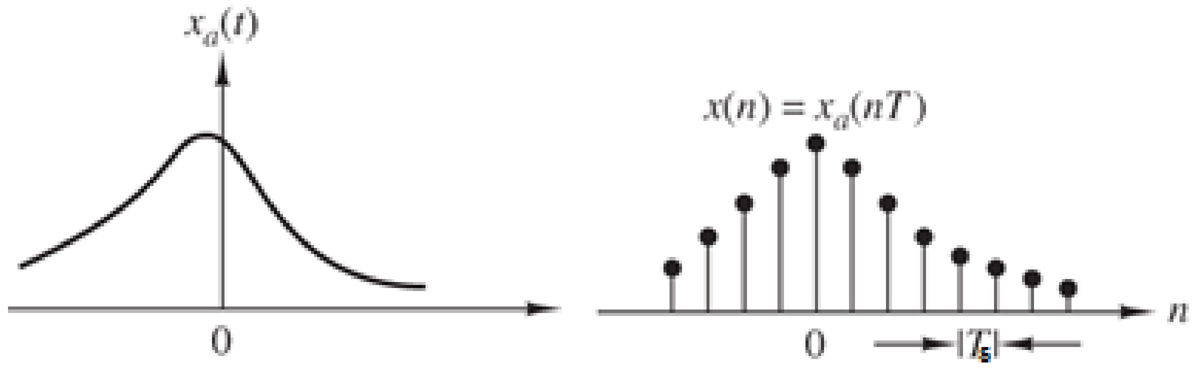
- For audio applications, the entire system may be at different physical locations. The channel is considered as another key element of the entire system.



Block diagram of a general audio system

Digital Signal Representation

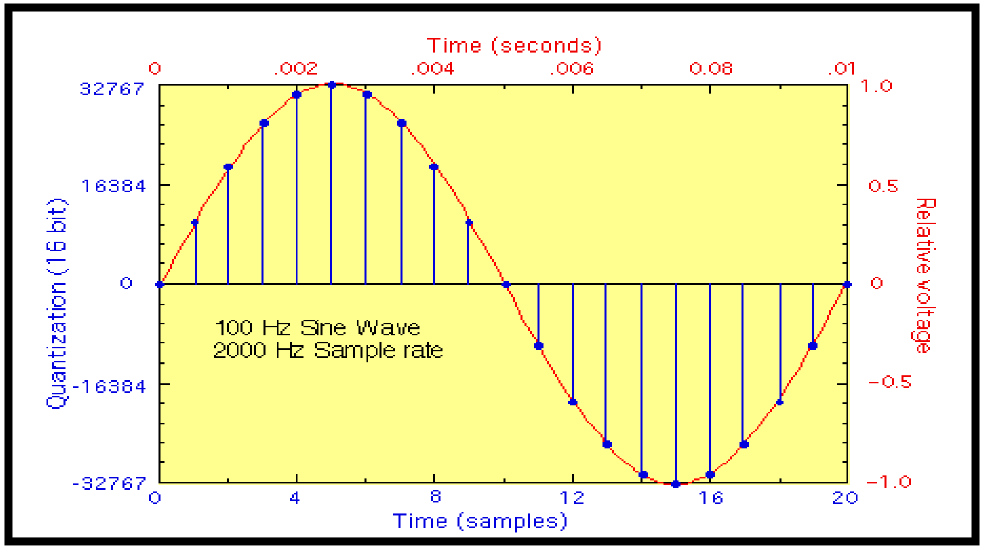
- The analogue input signal $x_a(t)$ is converted into digital one, $x(n)$.



- Two main operations are performed in the AD conversion are
  - Sampling: to get discrete samples from $x_a(t)$ (what is the suitable sampling frequency?)
  - Quantization: to represent these samples in a set of given values (how many bits are used to approximate the sample values?)

A simple PCM coder

- Analog signal is sampled at regular time intervals, T_s.
- The samples are then quantized into one of the digitized values (see below the linear PCM). The samples you received as input of the audio compression



- Simple, but does not take into account of ear model.
- Possible to use a nonlinear quantizer for coding of large and small amplitude signals with higher resolutions (non-linear PCM).

Digital Signal Representation

The samples can be expressed by

$$x(n) = \sum_{n=-\infty}^{\infty} x_a(nT_s)\delta(t-nT_s) = x_a(t)\sum_{n=-\infty}^{\infty}\delta(t-nT_s) \qquad (1.1)$$

where T_s is the sampling interval.

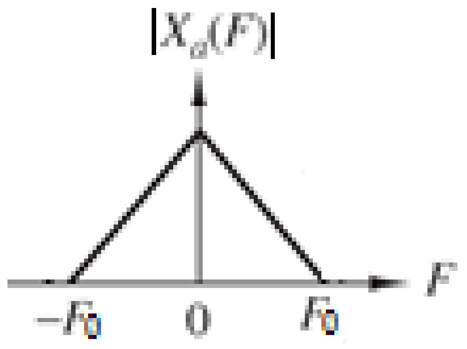
The main issue is to determine a suitable sampling freq. $f_s=1/T_s$.

- High sampling freq. leads too much computation or costs
- Aliasing effect is due to insufficient sampling frequency so that the recovered signal will not be *uniquely* related to the original input
- High sampling frequency provides better accuracy and high computational complexity, which should be compromised
- Note $x(n)$ has to be quantized before being processed, which will be discussed latter.

Digital Signal Representation - Sampling

Band-limited signal

- A signal is band-limited if there exists a finite frequency F_0 such that $X_a(f)$ is zero for $|f| > F_0$. The frequency F_0 is called the **signal bandwidth** in Hz.



Sampling Theorem

A band-limited signal, $x_a(t)$, with bandwidth F_0 can be reconstructed from its sample values $x(n) = x_a(nT_s)$ if the sampling frequency $F_s = 1/T_s$ is greater than twice the bandwidth F_0 of $x_a(t)$, i.e. $F_s > 2F_0$. Otherwise aliasing would result in $x(n)$.

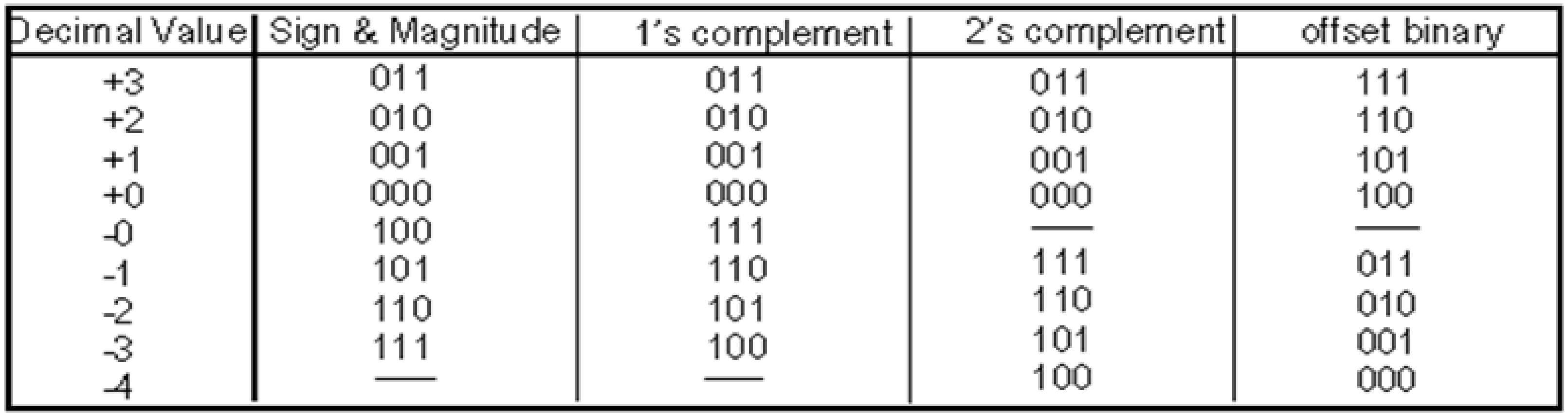
- The sampling frequency of $2F_0$ for an analog band-limited signal is also called the *Nyquist rate.*

NANYANG TECHNOLOGICAL UNIVERSITY

Quantization

Digital Signal Representation - Quantization

- Commonly used fixed point binary representations :
  - Sign and magnitude;
  - One’s complement;
  - Two’s complement;
  - Offset binary

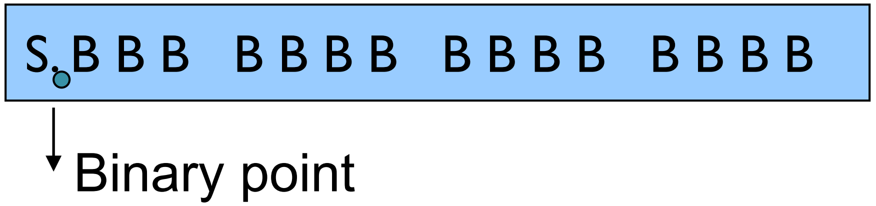


Decimal Value	Sign & Magnitude	1's complement	2's complement	offset binary
+3	011	011	011	111
+2	010	010	010	110
+1	001	001	001	101
+0	000	000	000	100
-0	100	111	—	—
-1	101	110	111	011
-2	110	101	110	010
-3	111	100	101	001
-4	—	—	100	000

- DSP devices commonly use the 2’s complement

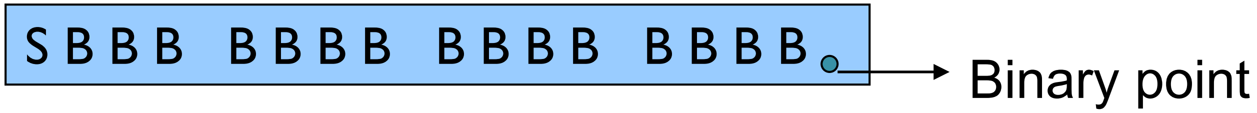
Digital Signal Representation - Quantization

- Fractional point number in *Qn* format, where *n* is the number of bits to the right of the binary point.
- Example : *Q15* (or 1.15) number format



S : sign bit
B : fractional bit

- Example : *Q0* (or integer) number format



- DSP supports both fractional and integer notations and is optimized for fractional Q15 notation

Quantization or Finite Word Length Effects

- **Finite Word Length Effects :**
  - The value to be represented: 0.000125481405920
  - Represented in Q15 (16-bit word length) = 0.0001220703125, Quantization error (word length effect) = 0.0000034111.
- **4 possible sources of quantization errors (finite word length effects) in DSP implementation:**
  - ADC quantization error - reduce SNR
    Remedy : can be improved by using longer word length in ADC
  - Arithmetic Overflow - When partial sum is larger than the maximum/smallest permissible value, leads to wrong result.

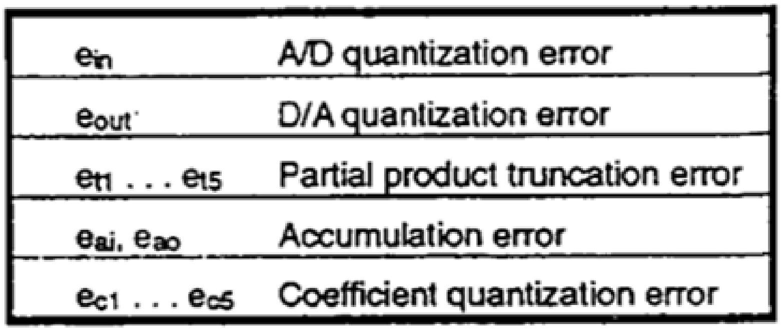
    Example:
    - In decimal representation: 0.6125 + 0.5 = 1.1125
    - or in binary sign and magnitude representation: 0.1011+0.1000 = 1.0011 which is equal to -0.1875. This is the effect of overflow.
  - Remedy : Scaling the values to be smaller to avoid overflow

Quantization or Finite Word Length Effects

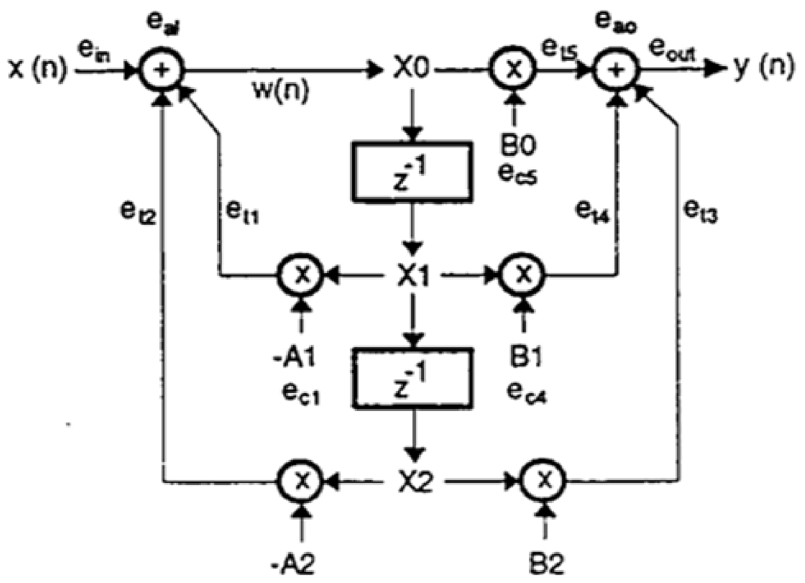
- **Coefficient quantization errors –** Representing filter coefficients with a limited number of bits, leads to modify the true frequency response.
  Remedy : Use more bits to reduce the error effects

- **Roundoff and truncation errors –** The product of two *N*-bit samples is 2*N*-bits. Truncate the least significant bits before storing the result of multiplication, which reduces SNR.
  Remedy : Use double-length summing of product.

- The figure shows all the possible errors due to word length effects.



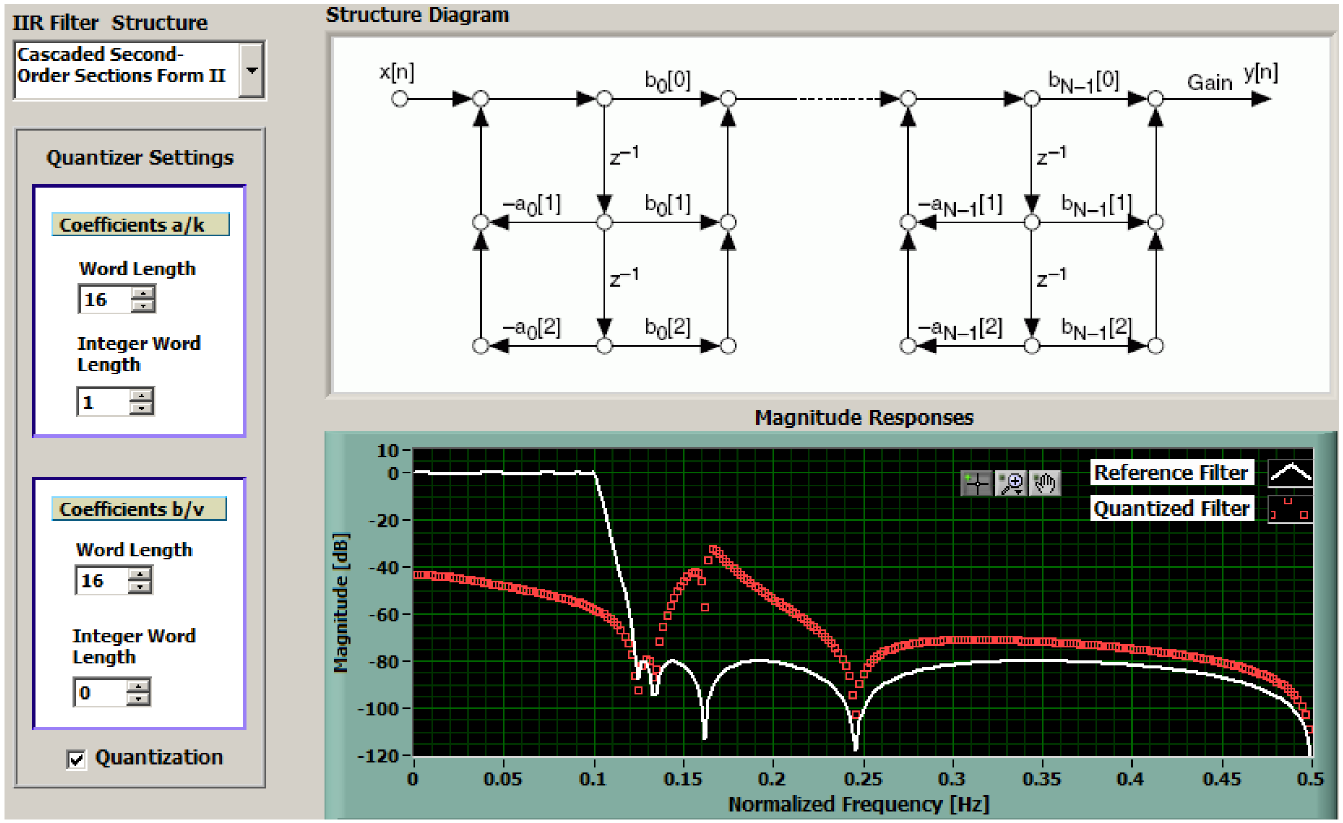
e_{in}	A/D quantization error
e_{out}	D/A quantization error
$e_{t1} \ldots e_{t5}$	Partial product truncation error
e_{ai}, e_{ao}	Accumulation error
$e_{c1} \ldots e_{c5}$	Coefficient quantization error



Quantization or Finite Word Length Effects

Example: Coefficient quantization errors

The red curves shows the frequency response of an IIR filter with 16-bit quantization.



IIR Filter Structure

Cascaded Second-Order Sections Form II

Quantizer Settings

Coefficients a/k

Word Length 16

Integer Word Length 1

Coefficients b/v

Word Length 16

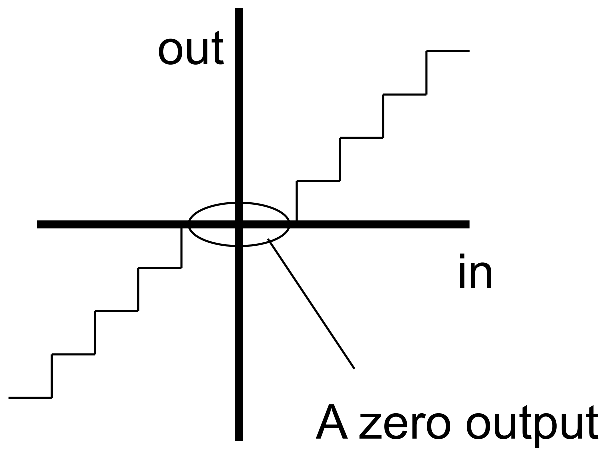
Integer Word Length 0

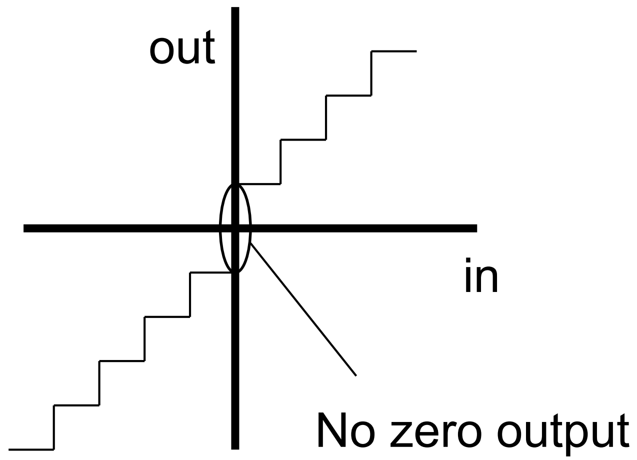
☑ Quantization

The number systems used in Audio Coding

- A uniform quantizer spans the positive and negative range of the input signal, in a symmetric manner (the same number of levels for positive and negative numbers).
- Two types of symmetric and uniform quantizers:

Midtread **Midrise**

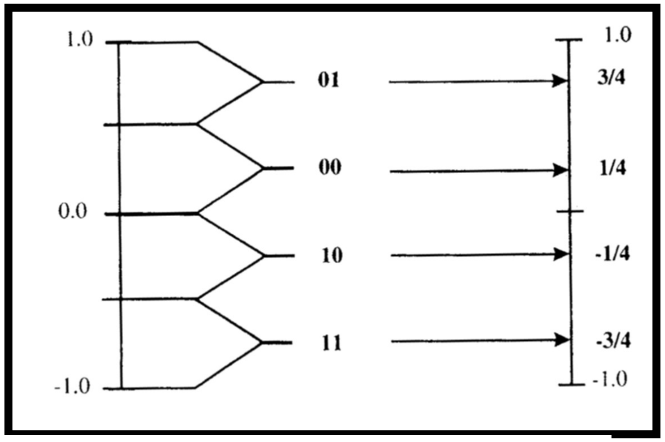


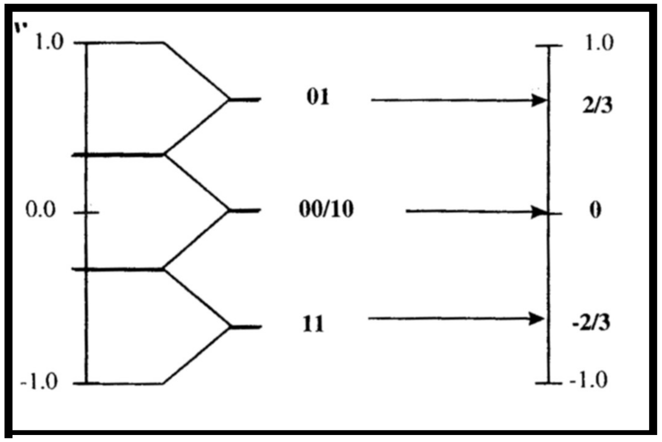


- With a number of bits, B, the numbers of values using a midtread and a midrise quantizer are 2^B-1 and 2^B, respectively

The number system used in Audio Coding

- Some observations:
  - In the **midrise** quantizer, the quantization step is $\Delta = 2 * x_{\max} / 2^B$
  - Number of steps in midrise is even.
  - In the **midtread** quantizer, the quantization step is $\Delta = 2 * x_{\max} / (2^B - 1)$
  - The number of steps in midtread is odd.





- 2-bit midrise quantizer :
  11 ➔ -0.75
  10 ➔ -0.25
  00 ➔ 0.25
  01 ➔ 0.75
  (No zero representation)

- 2-bit midtread quantizer :
  11 ➔ -2/3
  10 ➔ 0
  00 ➔ 0
  01 ➔ 2/3
  (10 or 00 is redundant)

Algorithms of Quantizing and Dequantizing

For Midrise

Quantizer: (ADC)

Code(number, B) = [s][code]

where s = 0 for number ≥ 0;

and s = 1 for number < 0

$$|\text{code}| = \begin{cases} 2^{B-1} - 1, & \text{when} |\text{number}| \geq 1 \\ \text{int}[2^{B-1}.|\text{number}|], & \text{elsewhere} \end{cases}$$

Dequantizer: (DAC)

Number(code; B) = sign*|number|

where sign = 1 if s = 0

and sign = -1 if s = 1

$$|\text{number}| = (|\text{code}| + 0.5) / 2^{B-1}$$

For Midtread

Quantizer: (ADC)

Code(number, B) = [s][code]

where s = 0 for number ≥ 0;

and s = 1 for number < 0

$$|\text{code}| = \begin{cases} 2^{B-1} - 1, & |\text{number}| \geq 1 \\ \text{int}\{[(2^B - 1).|\text{number}| + 1]/2\}, & \text{elsewhere} \end{cases}$$

Dequantizer: (DAC)

Number(code;B) = sign*|number|

where sign = 1 if s = 0

and sign = -1 if s = 1

$$|\text{number}| = 2.|\text{code}|/(2^B - 1)$$

NANYANG TECHNOLOGICAL UNIVERSITY

Examples

For Midrise

Quantize:

number = 0.6,

s=0, B=4

|code|=int[2^{4-1}x0.6] =int[4.8]=4→100

Then we have 0100.

Dequantizer:

Number(0100; 4) = sign*|number|

sign = 1 when s = 0

$$|\text{number}| = (4+0.5)/2^{4-1} = 0.5625$$

The output is 1x0.5625=0.5625

|Error| = |0.6 – 0.5625|=0.0375

When B = 6, we have 010011

Error =0.009375

For Midtread

Quantizer:

number = 0.6,

s=0, B=4

|code|={int[(2^4-1)x0.6+1]/2}

=int{15 x 0.6 +1)/2} =5→101

Then we have 0101

Dequantizer: (DAC)

Number(0101;4) = sign*|number|

sign = 1 when s = 0

$$|\text{number}| = 2 \times 5/15 = 0.666$$

The output is 1x0.666=0.666

|Error| = |0.6 – 0.666|=0.066

When B = 6, we have 010011

Error = 0.00317

Signal Representation - SNR

- A general signal quality measure is the **signal-to-noise ratio, (SNR),** which tells how much of the signal energy compared with the noise energy
- The theoretical SNR for a B-bit fixed point representation is 1.76+ 6.02*B dB

- **Example :** If B = 16 bits, SNR ~ 96 dB.
  - What is the SNR when using Q.15 format?
  - How much SNR improvement when using a 16-bit over an 8-bit ADC?
- Other ways to improve SNR include:
  - Maximize the input signal for the highest SNR
  - Filter noise above desired input frequency
  - Using oversampling, i.e., $F_s >> 2F_0$
- For audio compression, the key concept is to keep the quantization noise effects below our hearing threshold.

Floating Point Quantization

- The total number of bits is $R = R_s+R_m$, where R_s is number of bits for **scale factor** and R_m is the number of **mantissa bits**

To convert from a number into a scaled-mantissa floating point coded with R_s scale bits and R_m mantissa bits:

- Quantize the number as an R-bit code where $R = 2^{Rs} -1 + R_m$
- Count the number of leading zeros in |code|.
  - If the number of leading zeros is less than $2^{Rs} -1$, set the scale equal to $2^{Rs} -1$ minus the number of zeros
  - Otherwise, set the scale equal to zero

$$\begin{tikzcd}
{[s0000000abcd]} \arrow[r] & \begin{array}{l}\text{scale} = [000]\\ \text{mant} = [sabcd]\end{array} \arrow[r] & {[s0000000abcd]} \\
{[s0000001abcd} \arrow[r] & \begin{array}{l}\text{scale} = [001]\\ \text{mant} = [sabcd]\end{array} \arrow[r] & {[s0000001abcd]} \\
{[s000001abcde]} \arrow[r] & \begin{array}{l}\text{scale} = [010]\\ \text{mant} = [sabcd]\end{array} \arrow[r] & {[s000001abcd1]}
\end{tikzcd}$$

- If scale equals to zero, set the first mantissa bit equal to s and set the remaining R_m-1 bits equal to the bits following the $2^{Rs} -1$ leading zeros in |code|;
- Otherwise, set the first mantissa bit equal to s and set the remaining R_m-1 bits equal to the bits following the leading zeros (omitting the leading one).

Floating Point Quantization

To convert from a scaled-mantissa floating point code with R_s scale bits and R_m mantissa bits into a number:

- Create an $R=2^{R_s}-1+R_m$ bit code from the mantissa and scale factor where s is the first mantissa bit and |code|
  - has $2^{R_s}-1$ scale leading zeros
  - followed by the remaining R_m-1 mantissa bits if scale is zero;
  - otherwise followed by a one and then the remaining mantissa bits
  - followed by a one and as many trailing zeros as will fit if scale is greater than one

$$
\begin{array}{lcccl}
[\mathrm{s0000000abcd}] & \longrightarrow & \begin{matrix}\text{scale} = [000] \\ \text{mant} = [\mathrm{sabcd}]\end{matrix} & \longrightarrow & [\mathrm{s0000000abcd}] \\
[\mathrm{s0000001abcd} & \longrightarrow & \begin{matrix}\text{scale} = [001] \\ \text{mant} = [\mathrm{sabcd}]\end{matrix} & \longrightarrow & [\mathrm{s0000001abcd}] \\
[\mathrm{s000001abcde}] & \longrightarrow & \begin{matrix}\text{scale} = [010] \\ \text{mant} = [\mathrm{sabcd}]\end{matrix} & \longrightarrow & [\mathrm{s000001abcd1}]
\end{array}
$$

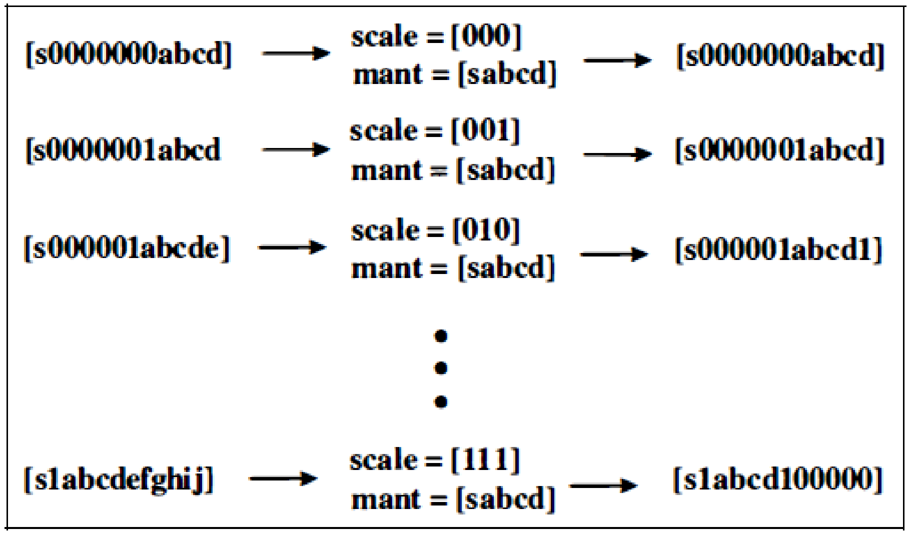
- Quantize the *R* bit code into a number.

NANYANG TECHNOLOGICAL UNIVERSITY

Floating Point Quantization

To convert from a scale-mantissa floating point code with R_s scale bits and R_m mantissa bits into a number:

Example (R_m = 5 and R_s =3)



Applying floating point quantization with Rs=3 scale bits, Rm=5 mantissa bits

- This eight bit total floating point quantizer reaches an accuracy at low signal levels comparable to a 12-bit uniform quantizer
- We first quantize our input signals using a 12-bit uniform quantizer.
- We use one of the five mantissa-bits to store the sign bit, and the remaining four mantissa-bits and the three scale-bits for storing the code.

Floating Point Quantization

- **Convention:** the scale factor is set equal to zero when there are the maximum allowed seven leading zeros
- We use this representation until the signal power reaches a level where the 12-bit quantization has less than seven leading zeros.
- In this case, we know that the first bit following those zeros is a one and does not need to be stored so we use the four mantissa-bits to store the four bits after the leading one.
- From this point all the way up to overload levels the quantization acts like a uniform quantizer with six bits corresponding to the sign bit, plus the leading one, plus the four other mantissa bits.
- As seen in the last line of the example, for the lower order bits beyond what we stored in the mantissa bits we split the difference and use a 1 followed by zeros when we recreate the 12-bit code.
- Once the 12-bit code is created we dequantize it back onto output signals as described in the previous section for uniform quantizers.

Block Floating Point

- Several floating point numbers are encoded using the same value of the scale factor to cut down on the number of scale bits used per number
- If the numbers are of different sizes, this method limits the ability of adjusting its bin sizes to the numbers being quantized.
- In this case, the signal SNR decreases for those values that are lower than the one used to set the exponent for the block.
- In general, to implement block floating point quantization, the scale factor for the group of numbers is set to the scale factor of the number with the largest absolute value.
- That single scale factor is then used to floating point quantize and dequantize all of the numbers in the group using floating point quantization.

- Both scale factor and block floating point are used in MPEG-1 coding.

Huffman Coding

Coding of Digital Symbols

- Quantization is to represent **one** particular sample (symbol) value.
- We now consider how to represent **a set of symbols** by using a minimum number of bits

- Among all available digital symbols, some are more frequently used and the others may be used rarely
- To transmit or store these symbols, we need to minimize the required channel bandwidth or storage capacity, and to maintain the required service quality, i.e., the quality of audios
- Entropy coding is used to remove the redundancy of the code by exploiting the **likelihood of the symbol occurrence**
- This is implemented by estimating the probability of each possible code, (as a descriptor for the source of the signals, for example, the probability of using a word in the dictionary).

Entropy Coding - Huffman Coding

- To obtain a code of **variable length** for reducing the overall number of bits to be transmitted/stored.
- The main idea is to use a shorter code length for frequently-occurring symbols and longer code length for seldom-occurring symbols.

Example: we may have

- 2-bit quantized signal that has symbols: [00],[01],[10],[11], with their respective probabilities of 70%,13%,10%, 7%.

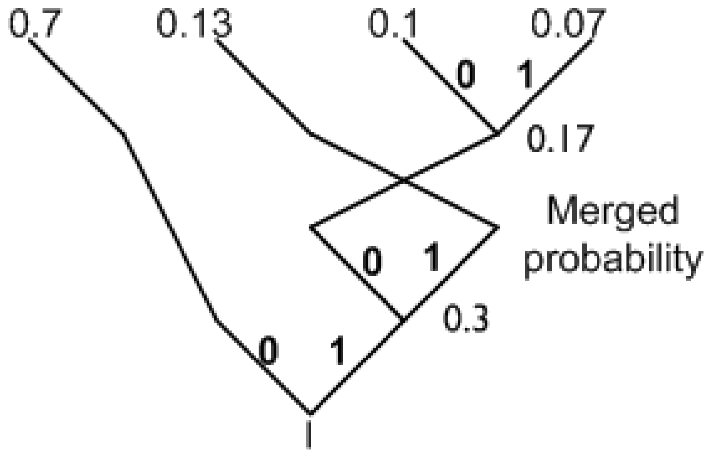
- Using Huffman coding to generate a variable-length code

Steps of Huffman Coding

1) Arrange all these symbols according to the descending order of their probabilities of occurrence
2) Combine the pair of symbols with two smallest probabilities as one whose probability is the sum of the probabilities of these two symbols.
3) Assign 0 and 1 to the two branches that are merged
4) Repeat steps 1), 2) and 3) until all the symbols have been dealt with
5) The code words are obtained by reading 0 (or 1) along the branch from the top to the root.

Example:

Probabilities:	0.7	0.13	0.1	0.07
Symbols:	00	01	10	11
Code words:	0	11	001	101



Note: a swap of the branches is performed to meet the condition in 1), i.e., the branches with the probabilities of 0.13 and 0.17 are swapped.

Performance of Huffman Coding

Measures of coding efficiency

- **Average code length** is defined by

$$L = \sum_{n=1}^{N} l_n p_n$$

where N is the number of symbols, l_n is the number of bits for the *n*th symbols and p_n is the probability of the *n*th symbol.

- **Entropy** is a function of the probabilities, P_n, of the code symbol being the *n*th code:

$$E = \sum_{n=1}^{N} p_n \log_2(1/p_n) = -\sum_{n=1}^{N} p_n \log_2(p_n)$$

- $\log_2(p_n)$ is the uncertainty of a symbol., for example, p_n is 1 or 0 means very certain (E =0); p_n is 0.5 means not very certain (E = 1).
- Shannon proved that entropy coding results in the lowest number of bits/symbol that any coder could produce.
- **Coding Efficiency** = E/*L*

Entropy Coding - Huffman Coding

Example:

p_n	0.7	0.13	0.1	0.07	Entropy
$p_n \log_2(1/p_n)$	0.3602	0.3826	0.3322	0.2685	1.3435

$$L = 0.7 \times 1 + 0.13 \times 2 + 0.1 \times 3 + 0.07 \times 3 = 1.47$$

Coding Efficiency = E/L =1.3435/1.47=0.9139

Summary:

- Entropy coding exploits the redundancy in the signal representation to provide new representation with fewer bits for the same information.
- Entropy is a measure of the information with a minimum number of bits needed to represent a given signal.

Frequency Domain Representation

Frequency Domain Representation

- Time domain signal representation may not be useful to reveal meaningful information
- Frequency domain signal representation is effective to show the characteristics of the signals

Demo Example, which shows

- The signal is a noise distorted single frequency sinusoidal wave
- Different frequencies are associated with different sounds
- Easily verify that it is much easier to reveal useful information in freq. domain

- Signal analysis in frequency domain has been extremely important for many applications.
- Among many available methods, discrete Fourier transform, (DFT) and discrete cosine transform (DCT) have been used in audio and image applications as well as other applications

Frequency Domain Representation

THE DISCRETE FOURIER TRANSFORM (DFT)

- The DFT takes an *N*-point sequence *x*(*n*) in the time domain to produce an *N*-point sequence, *X*(*k*), in the frequency domain. The *N*-point DFT of *x*(*n*) is defined by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \qquad 0 \le k < N$$

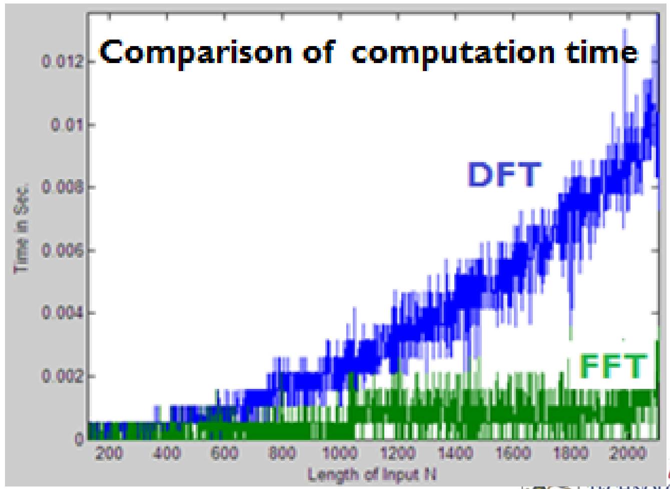
where $W_N = e^{-j2\pi/N}$.

- The inverse DFT to obtain *x*(*n*) from *X*(*k*) is

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-nk}, \qquad 0 \le n < N-1$$

- Because we compute the DFT of *N*-point only, a window of *N*-point has to be used to capture the segment of input data stream

- Fast Fourier transform (FFT) has the same definition as the DFT and performs the significantly faster computation.

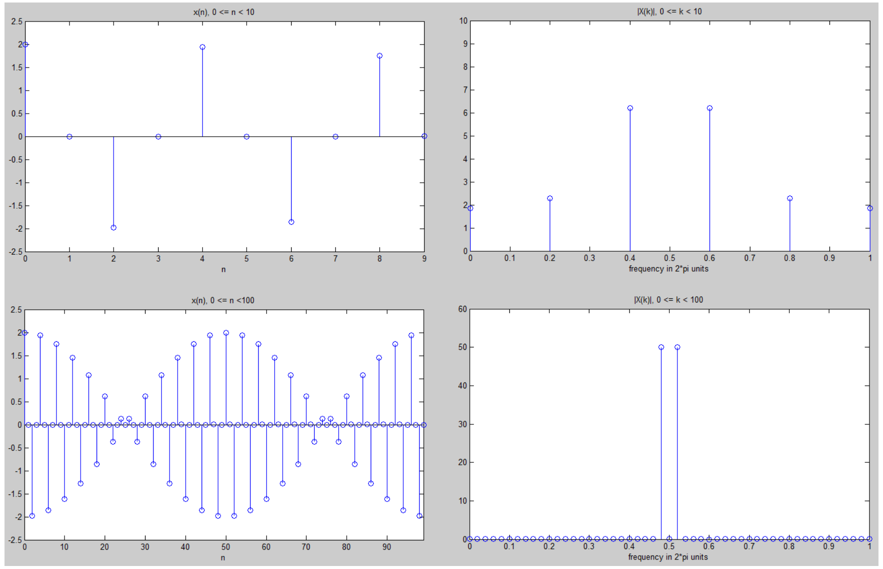


Frequency Domain Representation

Example: Consider $x(n) = [\cos(0.48\pi n) + \cos(0.52\pi n)]w(n)$, where $w(n) = 1$ for n=0, N-1, otherwise $w(n)$=0.

a) Determine and plot the DFT of $x(n)$ for N=10;

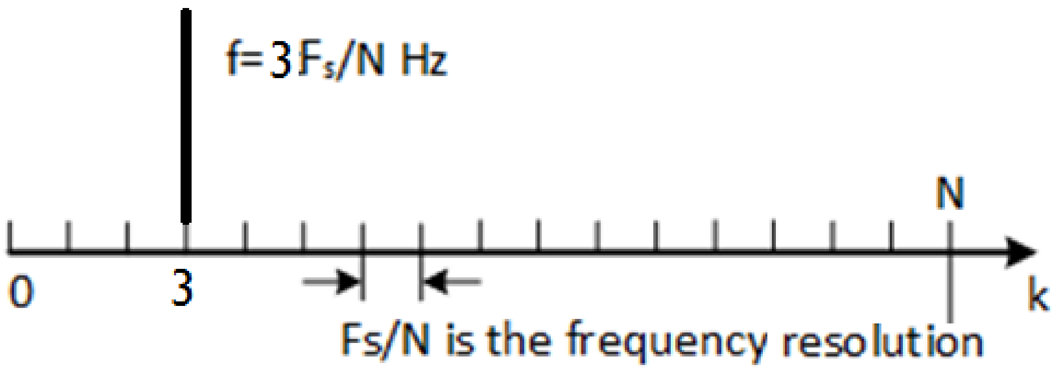
b) Determine and plot the DFT of $x(n)$ for N=100.



- Because N=10 is too small, the DFT of 10 points is not usable (poor resolution)
- Better resolution in the frequency domain is obtained only when enough data (e.g. large N) are collected. For example, we can accurately guess the frequencies (the peak location) in the signal when N = 100
- The obtained DFT is the DFT of the **product,** $[\cos(0.48\pi n) + \cos(0.52\pi n)]*w(n)$, instead of the DFT of $\cos(0.48\pi n) + \cos(0.52\pi n)$, where $w(n)$ is the window function.

Frequency Domain Representation

- The following understandings of the DFT outputs are most important.
- The frequency resolution is defined as $\Delta f = B_s/M$, where B_s is the signal bandwidth and M is the number of points used to represent the signal spectrum.
- In practice, we assume $B_s = F_s/2$ (Hz). and DFT uses $N/2$ values to represent B_s for real signal.
- Therefore, $\Delta f = (F_s/2)/(N/2) = F_s/N$, *(Hz)* see the figure below.
- The *k*th DFT value there is a component at the **frequency band** kF_s/N *Hz*, i.e., for k=3, it is $3F_s/N$ *Hz*.



Frequency Domain Representation

Example: A digital data sequence is obtained by a sampling frequency of 8 kHz. Calculate the frequency resolution provided by a DFT (or FFT) of 256 and 512 samples, respectively.

Solution: The **frequency resolution** is defined by $\Delta f = F_s/N$. Therefore the frequency resolution is

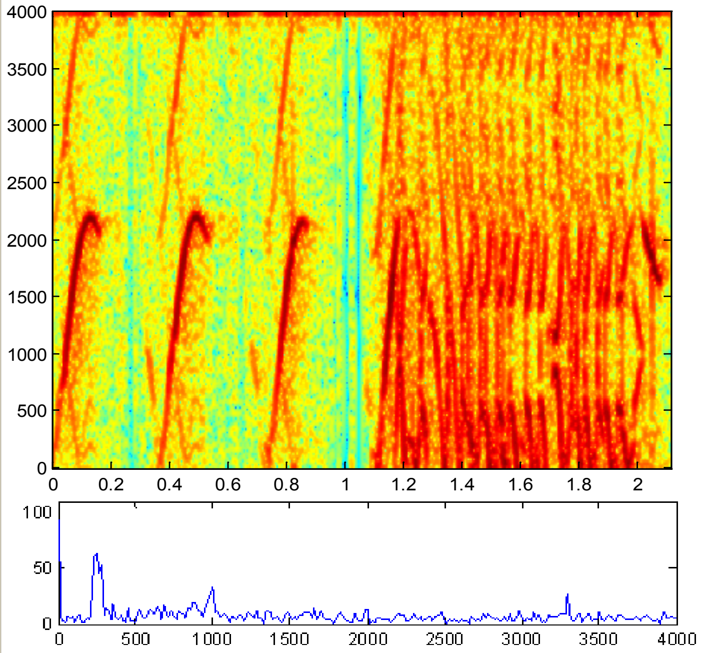
$$\Delta F = 8k/256 = 31.35 \text{ Hz} \quad \text{and} \quad 8k/512 = 15.625 \text{ Hz}$$

An FFT with a large N provides a better frequency resolution.

- In general, a large N for FFT requires a large processing delay, which deduces the temporal resolution
- The **temporal resolution** is defined by $\Delta T = 1/\Delta f = NT_s = N/F_{s,}$ where T_s and F_s are the sample interval and sampling frequency, respectively
- Another consequence for a large N is the increase of the computation complexity. For example, the FFT algorithm requires the number of multiplications and additions proportional to $N\log_2 N$.

Frequency Domain Representation

- The DFT is particularly useful for signals whose frequencies do not change with time.
- It only tells what the frequencies are in the signal.
- When the frequencies are changed with time, such as speech and audio, DFT does not tell when the frequencies are generated and when the frequencies disappears.



- The top figure is easily related to the sound content, i.e., the frequency characteristics changing with time.
- The DFT in the bottom figure does not provide good information on the sound contents.

Frequency Domain Representation

- Also a DFT processes only *N* samples, and cannot deal with continuous data stream
- An feasible method is to compute the consecutive segments of the input stream.
- These segments are obtained by using a window function. The frequencies are assumed to be unchanged within the window duration.
- The window shape should be adaptively selected according to the content of the input signal (more discussion in later weeks).
- This operation is known as the short-time Fourier transform (STFT), which is often used for spectrum analysis.
- In audio coding, we adaptively use different window lengths for spectrum analysis.

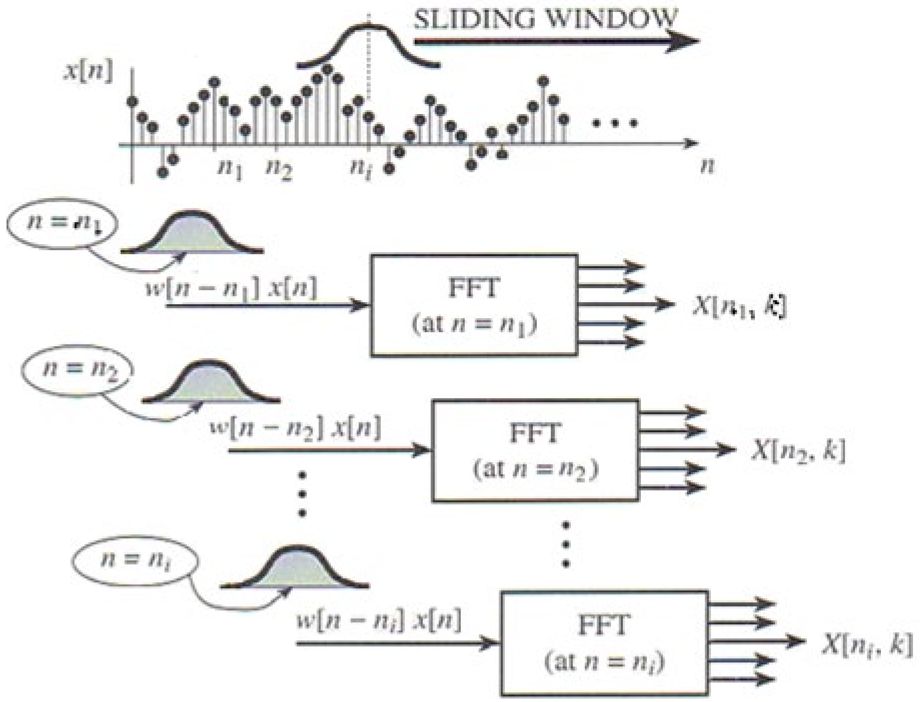
Frequency Domain Representation

- The spectrum located at n_0 is defined as

$$STFT(x, n_0, k) = \sum_{m=n_0-N+1}^{n_0} x[m]w[m-n_0]e^{-j(2\pi/N)km}$$

where $w[n]$ is an N-point window function to divide the input into N-point segments, n_0 is the current time index and k is the index of DFT.

- For each n_0, the STFT is an DFT operation.
- For better performance, an overlap between the adjacent segments is generally used.



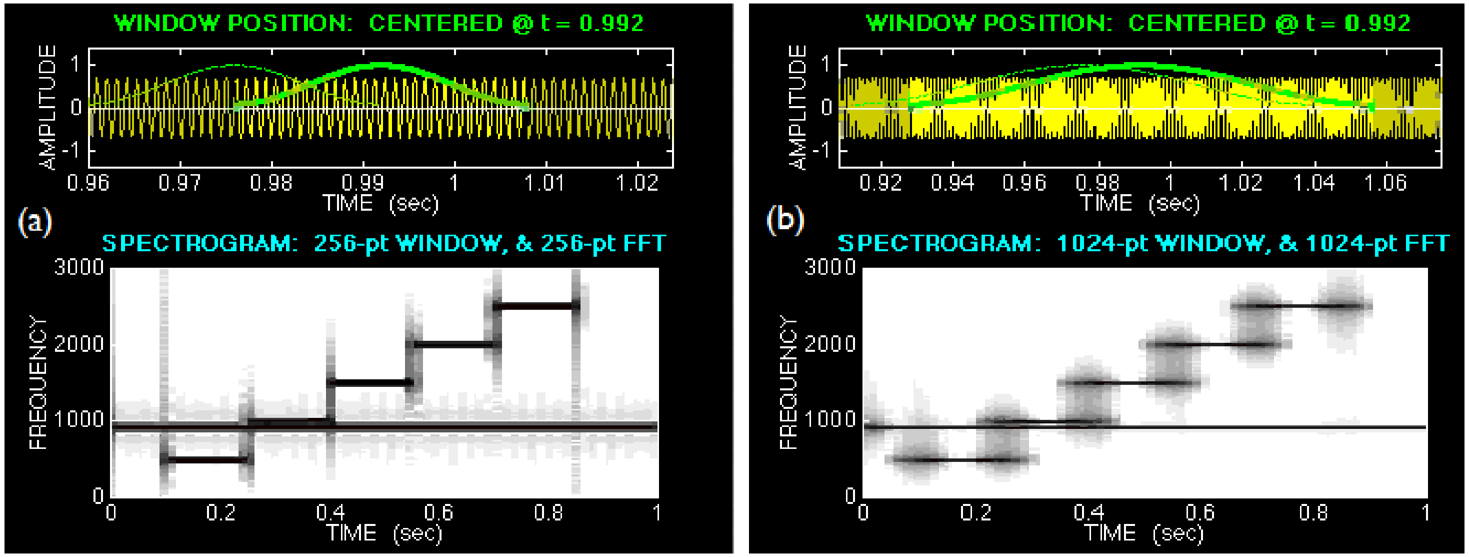
- The STFT is a sequence of N-point DFTs. The spectrogram of $x(n)$ is defined as $|STFT(x, n_0, k)|$ or $\log|STFT(x, n_0, k)|$, which is a function of n_0 and k.

Frequency Domain Representation

Effects of Windows

- We should clearly understand that the window function is not a part of signal. It is necessary only because we use it to capture a signal segment.
- The output of STFT is the spectrum of the product of the window function and the input signal segment, instead of the spectrum of the signal segment that we hope to have .
- A process to remove the effect of the window function will be introduced in latter lectures
- The length of the window function is also very much relevant to the frequency resolution and temporal resolution offered by the STFT.

Frequency Domain Representation



- The window lengths for (a) and (b) are 256 and 1024 points, respectively.
- (a) maintains reasonable resolutions in both time and frequency domains (horizontal and vertical directions), and reveals the true information.
- (b) uses a long window in the time domain (poor time resolution) and achieves a thinner line (high frequency resolution).
- (b) is not usable since the frequency components are overlapped, which is not truthfully reveal the signal information.
- The compromise between the resolutions in the time and frequency domains is made by carefully select window length, which will be further discussed latter.

Frequency Domain Representation

- We generally compute the SPL of signal frequencies with the equation

$$SPL_{DFT} = 96dB + 10\log_{10}\left[\frac{4}{N^2 < w >^2}\sum_{peaks} | X(k) |^2\right]$$

where $X(k)$ is the kth DFT value of input, $<w^2>$ is the window power factor, N is the length of the DFT.

- The range of k for the sum is from 1 to $N/2$ and others are ignored, due to the symmetric property of the DFT.
- For example, $<w^2>$ is the window energy which is 1 for rectangular window and 1/ 2 for hamming and sine windows.
- This SPL is used in calculating the effect of masking

Audio Applications

Current Practice of Audio Coding

- The basic requirement is to reduce the data, as much as possible, to be transmitted or stored without losing the quality **perceived** by human hearing system
- The main coding process is **lossy** to meet the above requirements by throwing away those audio information that is not audible.
- Although not possible to reproduce the actual uncompressed audio at the audio player, our ears still perceive the audio with very good sound quality.
- The term "CD quality" is often referred to compressed audio that sounds like true CD (44.1 kHz, 16 bits/samples), see comparisons below.
- **Example:** Compare Serenade-op.6_wave (17.2 Mb) with

  Serenade_op.6_mp3 (2.2 Mb)

The Aim of Digital Audio Coding

- ***What is a Digital Audio Coder?***

Digital Audio Coder

Audio source
- How to represent sound source?
- How to describe it?

Ear (perceptually identical to the input)
- How to model the ear processing acoustical stimuli?

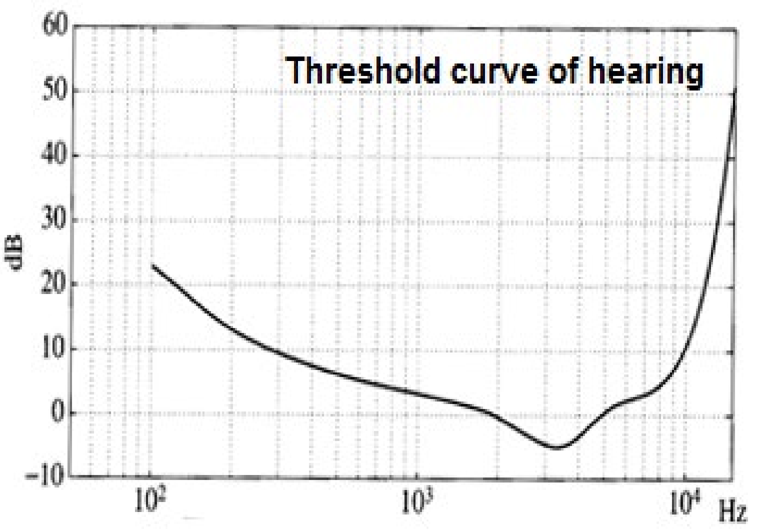
How to reduce amount of information (bits) needed to represent the signal by throwing away **perceptually** irrelevant information without sacrificing the sound quality?

Important Factors in Audio Coding

- **Fidelity:** perceptual quality is equivalent between output and input of the coder
- **Data rate:** down load throughput, storage and bandwidth of system
- **Complexity:** hardware and software costs in encoder/decoder
- **Coding delay:**
  - Due to all kinds of processes from the sources to the human ears.
  - The tolerance on the amount of delays depend on different applications of real time requirements
- **Robust:** able to work with different audio sources in various environments
- **Scalability:** able to deal with mono, stereo, multi-channel signals at different rates
- **Flexibility:** applicable to as many signal types as possible
- **Standard:** fatal for any commercial development

Human Hearing System

- Human hearing system responds the signals in a frequency range as shown in the Figure below

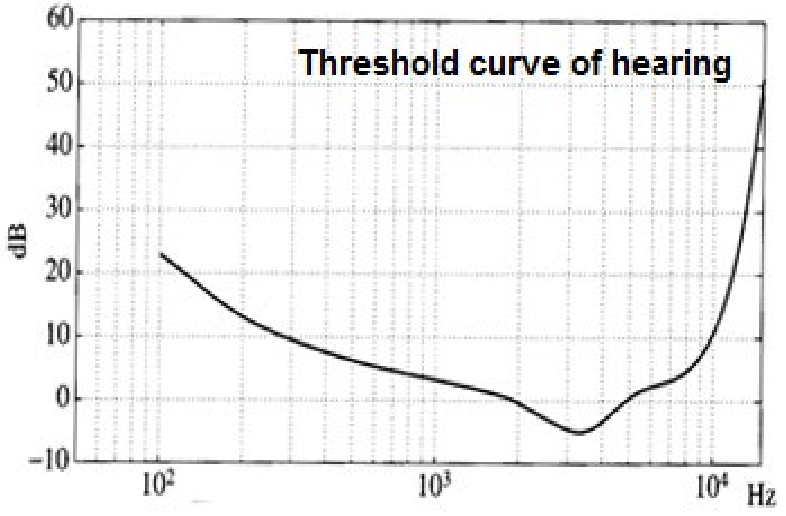


- The figure is in terms of loudness in the frequency range.
- Sound is not heard if the loudness level is below the curve
- The most sensitive frequency range is from 1 to 6 kHz
- Lower and higher frequency components require higher loudness level to be heard.

- Because the hearing systems respond differently to different frequency bands, we
  - divide the audio frequency range into different bands,
  - quantize each band separately with different number of bits
  - combine these coded signal bands for the restored signal

A more complex Audio Coder

- In the frequency domain, we may allow users to allocate a suitable no. of bits to different frequency bands.
  - Because human ear is more sensitive at around 2-5 kHz, more bits are allocated for signal components in this frequency range and fewer bits allocated for signal components in other frequencies.
- Further redundancy reduction can be performed by using:
  - Predicting the next few samples without using transmitted/stored samples (will be taught by Prof Lin)
  - Entropy coding for minimum transmitted /stored data
  - Perceptual masking to remove non audible data
  - The last two items will be considered.



Applications

- Consumer devices:
  - MP3 player
  - Mini Disc (Sony)
  - Game (Play Station)
  - DVD
  - Smart phone
- Streaming and distribution:
  - **Internet (communications, audio/video data distribution/braodcasting, online translation …)**
  - Audio-on-demand (set-top box)
  - Satellite/Terrestrial audio broadcast (DAB)
  - Digital TV with multi-channel surround
  - Digital film
- Audio is fast moving towards a **total digital technology** in *capture, storage, post production, pre- and post-processing, exchange and distribution*.

Data reduction by quality reduction

Name	No. of Samples Per sec	No of bits Per Sample	No. of Channels	Freq range (Hz)	SNR (dB)	PCM bit rate (kbps)
DVD	96000	24	6	48k	144	**13,824**
DAT	48000	16	2	24k	96	**1,536**
CD	44100	16	2	22k	96	**1,411**
FM	32000	12	2	16k	72	**768**
PC	22050	12	2	11k	48	**176**
Phone	8000	8	1	3.4k	48	**64**

- The above shows the parameters of a few common applications
- The audio quality is reduced as the data rate decreases
- The data rate is computed by

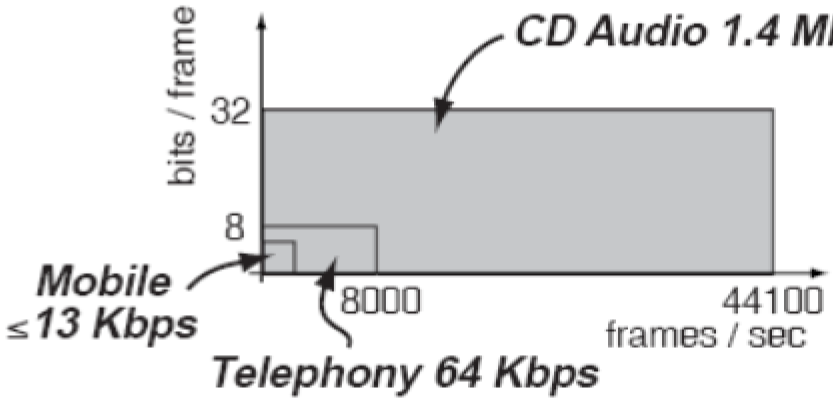
$$\mathbf{F_s\ (sample/sec) \times b\ (bit/sample) \times c\ (channel)}$$

- **How big is audio data? What is the bitrate?**
  - F_s frames/second (e.g. 8000 or 44100)

    x C samples/frame (e.g. 1 or 2 channels)

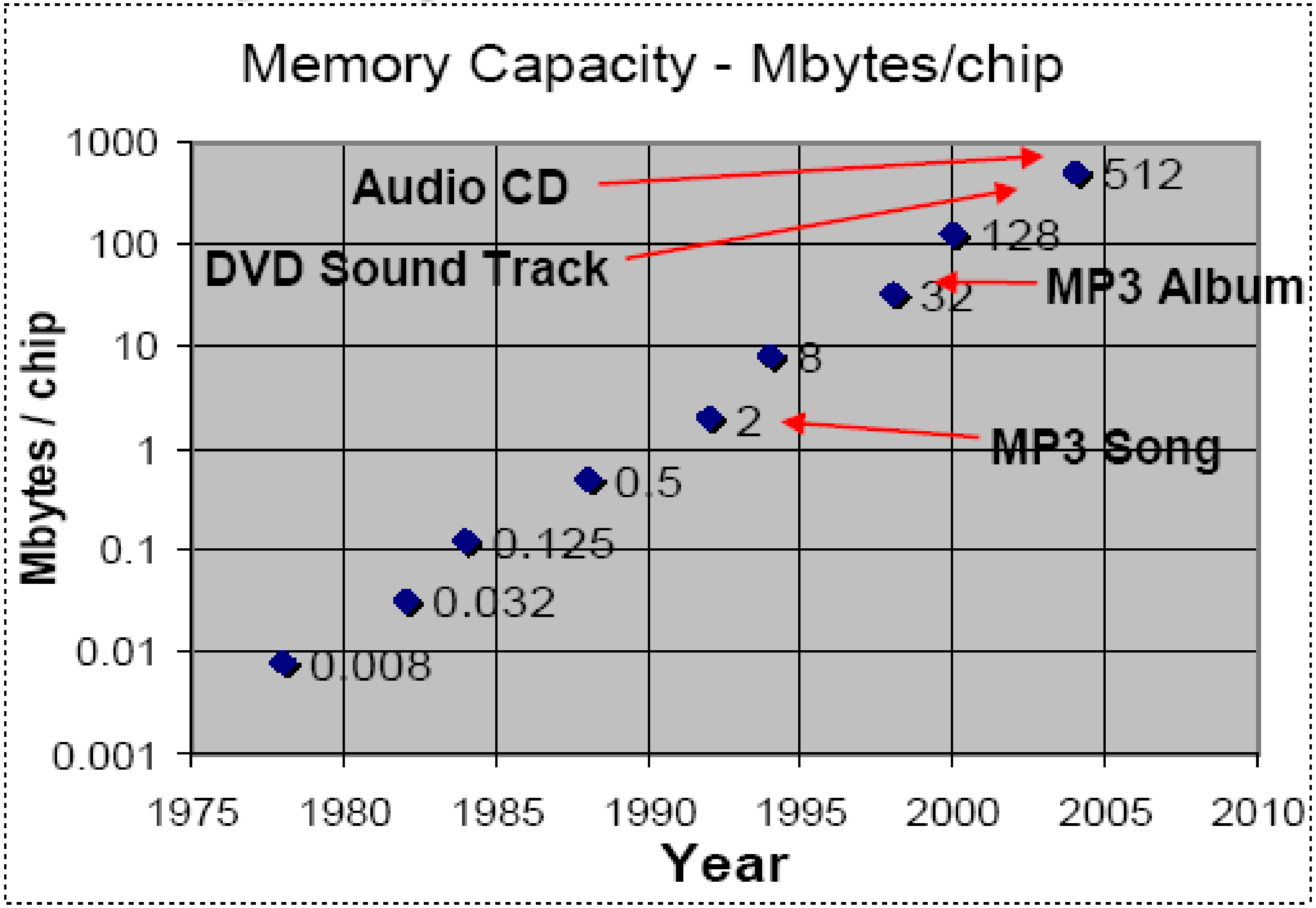
    x B bits/sample (e.g. 8 or 16)

  $\rightarrow$ $F_s \cdot C \cdot B$ bits/second (e.g. 64 Kbps or 1.4 Mbps)



- **How to reduce?**
  - lower sampling rate $\rightarrow$ less bandwidth (muffled)
  - lower channel count $\rightarrow$ no stereo image
  - lower sample size $\rightarrow$ quantization noise

Memory Required for Audio



- 32 x 8 Mbits = 128 kbits/sec * T sec ➔ T = 34 mins
- 2 x 8 Mbits = 128 kbit/sec * T ➔ T = 2 mins

Some Practical Questions

Example 1. How long (in minute) can a CD stereo sound tracks be saved onto a 1000 Mbyte in the hard disk drive? How much longer (in minute) can be saved if a compression algorithm with 4:1 ratio is used?

1000M * 8 /(60 * 44.1k * 16 * 2) = 88.57 mins

88.57 x 4 = 354.3 mins

Example 2. The bit rate of the CD is 2x44.1kx16 = 1.41Mbps. But CD needs a significant overhead for runlength-limited line code for synchronization, error correction of up to 49 bits for every 16-bit audio samples. What is the total stereo bit rate? What is the % overhead ?

Total bit rate = 44.1k*2*49 = 4.32 Mbps

% overhead = (49-16)/49 * 100% = 67%

The overhead requires even more downloading time.

How much compression is required?

Example 3. Network channels have a bit rate capacity of 2 Mbps. What is the compression ratio to deliver a stereo audio signal (DAT) sampling with 16 bit/s? (Assuming 2/3 of network data are used for error correction)

Compression ratio =

(48 kHz x 16 bit/s x 2 channels x 3) / 2Mbps = 4.608/2

- Can you listen on line this high quality audio? Why not?

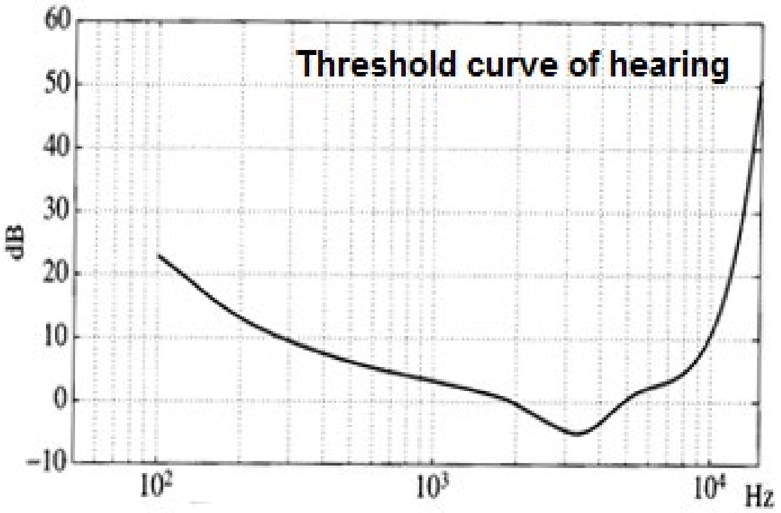
Similarly, for a speech signal, sampling at 8 kHz, with PCM of 12 bits can be reduced to 8 bits using nonuniform quantizer to fit in a bit rate of 64 kbps for PSTN.

The compression ratio is only **1.5:1**.

How to reduce data rate?

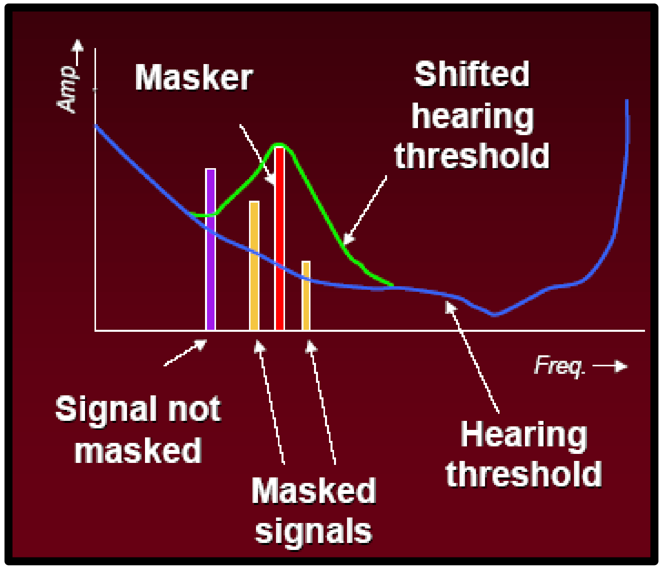
- Do not transmit or store anything that the ears cannot hear (irrelevant data that are perceptually insignificant)
- Allow some quantization noise that cannot be heard, (means that the samples can be represented by fewer bits).
- We need to understand psychoacoustics:
  - Range of human hearing loudness curve (do not send these components that are out of hearing range, also do not send these components that below the hearing threshold)
- Remove redundant information:
  - Sample at the right frequency
  - Remove correlated data
- Make use of masking phenomena: frequency masking and temporal masking

- Ear can accommodate a very wide dynamic range (DR) of about 140 dB. Equivalent to sound intensity of 10^{12} to 1 (0 dB SPL)
- Ear sensitivity is freq dependent. Max sensitivity occurs at 1 to 5kHz, where we hear signals several dB below 0 dB.
- Relative insensitive at low and high freq., Eg. 20 Hz tone would have to be approx 70 dB louder than 1 kHz tone to be barely audible.
- A perceptual coder compares the input signal to the minimum threshold, and discards signal that falls below the threshold, because the ear cannot hear this signal.

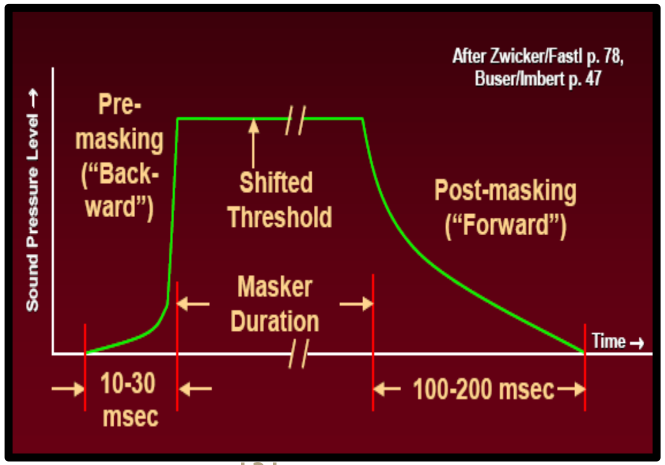


How to reduce data rate?

- **Masking:** masking is the process by which the threshold of hearing for one sound is raised by the presence of another sound.
- **Frequency masking** is due to the fact that the hearing system has a limited frequency resolution to distinguish different frequencies in the audio signal.



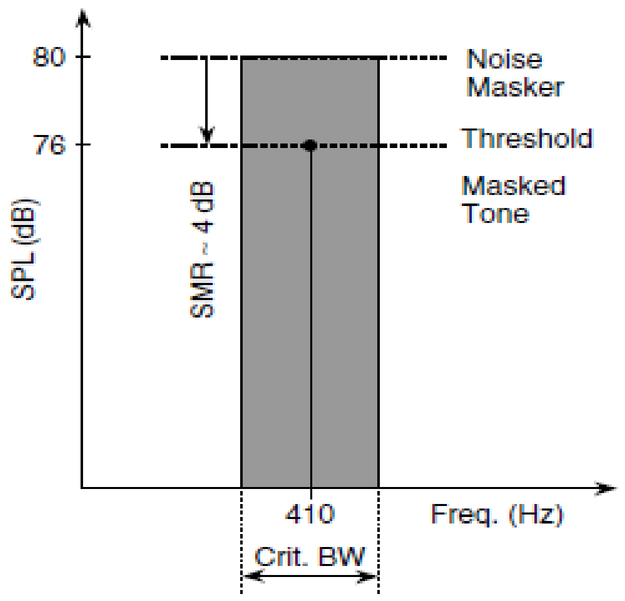
- **Temporal masking** is also used for compression.



Frequency Masking

- Masking refers to a process where one sound is rendered inaudible because of the presence of another sound.
- It is mainly because the presence of a strong noise or tone masker creates an excitation of sufficient strength on the basilar membrane at the critical band location to block effectively detection of a weaker signal.
- We shall mainly consider two types of simultaneous masking, namely *noise-masking-tone* (NMT), and *tone-masking-noise* (TMN).
- Tone-masking-tone (TMT) can be easily described.
- Noise-masking-noise (NMN) is more difficult to characterize.

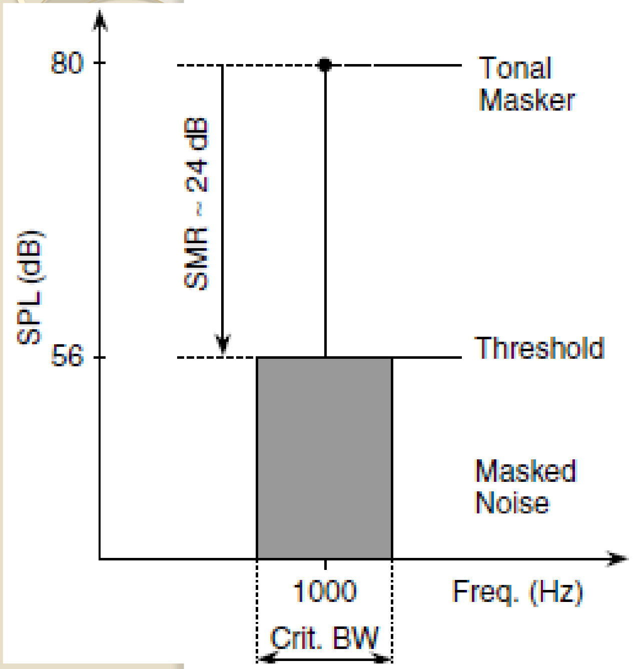
Noise masking tone



- A narrowband noise (e.g., having 1 Bark bandwidth) masks a tone within the same critical band,
- The intensity of the masked tone is below a predictable threshold given at the centre frequency of the masking noise.
- The threshold of detection is the smallest difference between the SPL of the masking noise and the SPL of the masked tone when the frequency of the masked tone is close to the masker's centre frequency.

- The figure shows that, a critical band noise masker cantered at 410 Hz with an intensity of 80 db SPL masks a 410 Hz tone, and the resulting signal to minimum masking ratio (SMR) at the threshold of detection is 4 dB.

Tone masking noise



- A pure tone at the centre of a CB masks narrow band noise, provided the noise spectrum is below a predictable threshold directly related to the strength of a centre frequency of the masking tone.
- The smallest difference between the intensity of the masking tone and the intensity of the masked noise occurs when the masker frequency is close to the centre frequency of the probe noise, i.e., between 21 and 28 dB.
- In the figure, a narrowband noise of one Bark bandwidth centred at 1 kHz is masked by a 1 kHz tone of intensity 80 dB SPL.

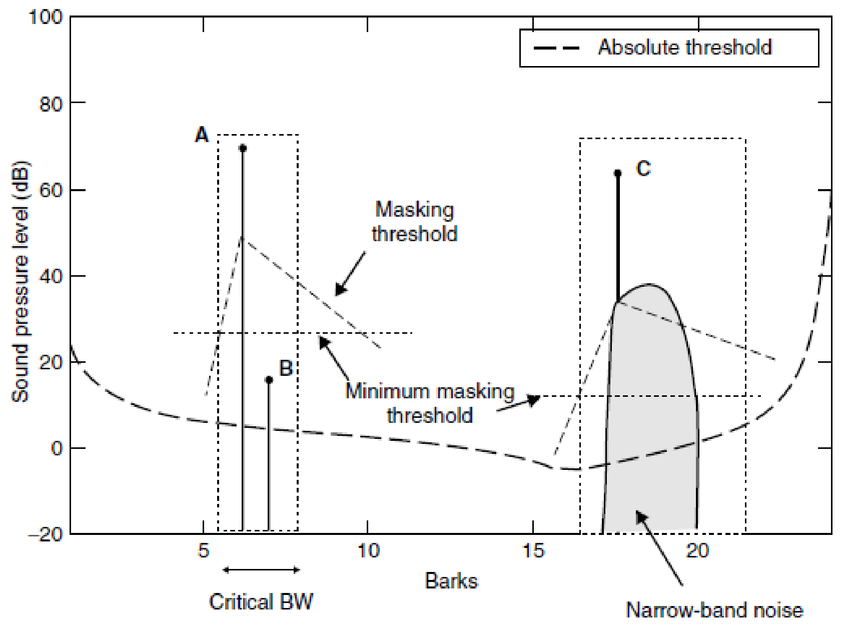
- The resulting SMR at the threshold of detection is 24 dB.
- Similar to NMT, the TMN masking power decreases for the narrow noise moving away from the masker.

Example

Consider the Figure below. State whether tone B would be masked by tone A and whether tone C would mask the narrow-band noise. Give reasons.

Solutions:

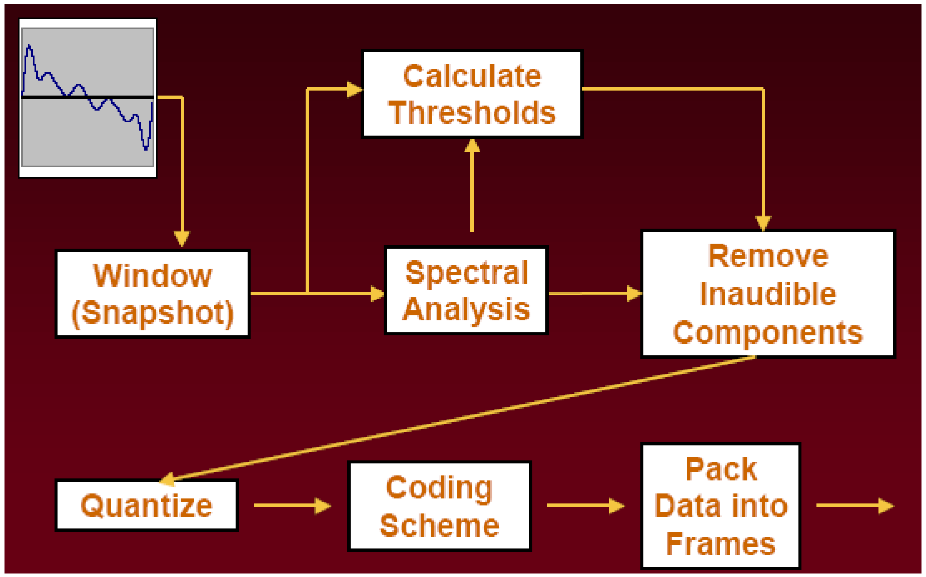
- Tone B is masked by tone A since B is near A and has a much smaller SPL
- It is likely the narrow band noise C is audible since it SPL is higher than the masking threshold.



- In general, the SMR is much larger for TMN than that for NMT since narrow band noise has more impact on our basilar membrane.

General Block Diagram for audio encoding

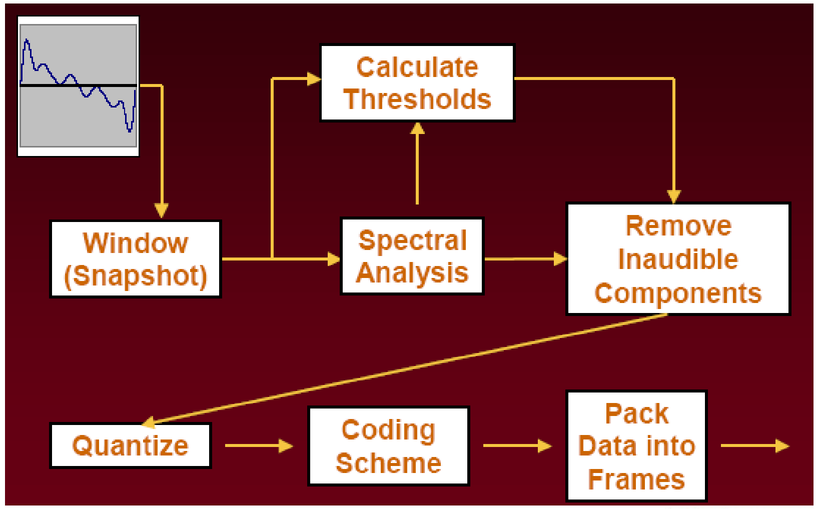
- We would like to throw away the vast majority of data and still end up with something that sounds very much like the original
- This is a general block diagram of an audio coder.
- The input data stream is segmented into successive data segments. Each segment will be analyzed by the spectral analyzer.
- At the same time, the masking threshold is also derived for the data segment.
- Based on the masking threshold information obtained, only some frequencies and data samples are selected for quantization. Some inaudible components will be removed.



General Block Diagram for Lossy Audio Encoding

- The survived data are quantized into some discrete values which are further coded by an efficient coding algorithms.
- Finally, the coded data are arranged into a defined format for transmission or storage.
- A reverse process is to be performed for recovering the original information from the coded data.
- We will show the details on the techniques used to compute the power spectrum, derive the masking threshold and bit assignment in the following lectures.

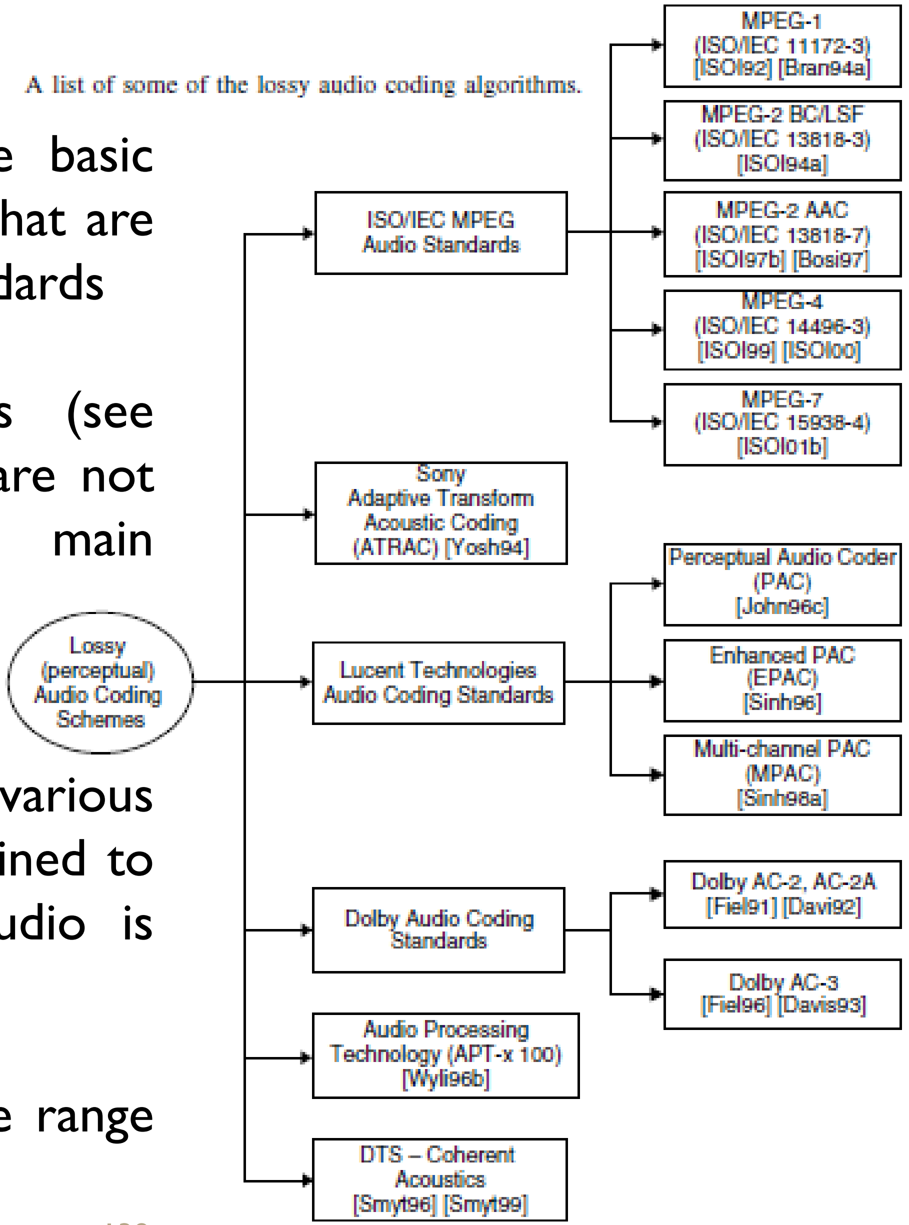
- Related signal processing techniques, such as sampling, quantization, Fourier transforms, filter banks, used for audio coding will be also examined.



Lossy Compression Algorithms

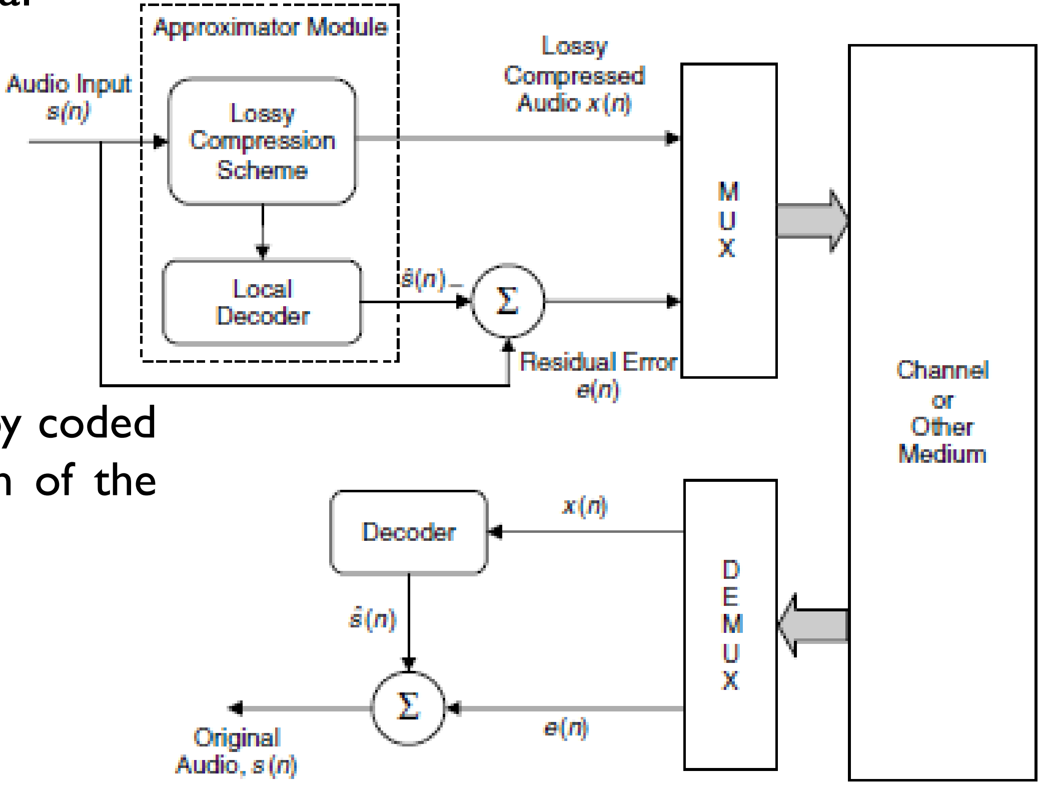
- We shall learn only some basic concepts for compression that are used to support MPEG standards
- Although other standards (see figure) are still used, they are not widely used for the main applications.
- Compatibilities among various standards have been maintained to ensure the compressed audio is playable.
- Compression ratio is in the range of 10:1-25:1

A list of some of the lossy audio coding algorithms.


Lossy (perceptual) Audio Coding Schemes
ISO/IEC MPEG Audio Standards
MPEG-1 (ISO/IEC 11172-3) [ISOI92] [Bran94a]
MPEG-2 BC/LSF (ISO/IEC 13818-3) [ISOI94a]
MPEG-2 AAC (ISO/IEC 13818-7) [ISOI97b] [Bosi97]
MPEG-4 (ISO/IEC 14496-3) [ISOI99] [ISOI00]
MPEG-7 (ISO/IEC 15938-4) [ISOI01b]
Sony Adaptive Transform Acoustic Coding (ATRAC) [Yosh94]
Lucent Technologies Audio Coding Standards
Perceptual Audio Coder (PAC) [John96c]
Enhanced PAC (EPAC) [Sinh96]
Multi-channel PAC (MPAC) [Sinh98a]
Dolby Audio Coding Standards
Dolby AC-2, AC-2A [Fiel91] [Davi92]
Dolby AC-3 [Fiel96] [Davis93]
Audio Processing Technology (APT-x 100) [Wyli96b]
DTS – Coherent Acoustics [Smyt96] [Smyt99]

Lossless Coding Algorithms

- The basic idea is to remove the statistical dependencies associated with the signal via prediction techniques.
- This provides a close approximation to the input audio and a low variance prediction residual



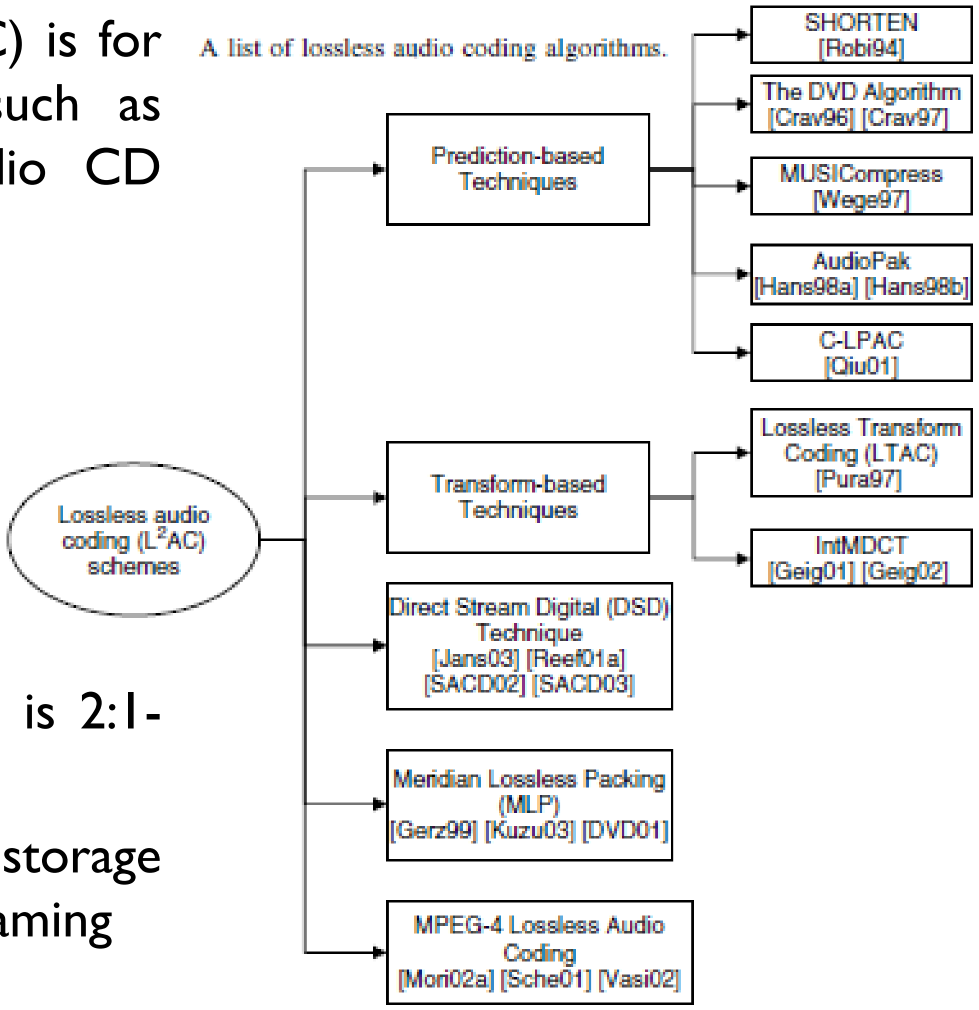
Lossless audio coding scheme based on the LAC method.

- The residual error is entropy coded and used for reconstruction of the original signal

Lossless Coding Algorithms

- Lossless audio coding (L^2AC) is for high-end storage format such as DVD audio and super-audio CD (SACD)

A list of lossless audio coding algorithms.


Lossless audio coding (L^2AC) schemes
Prediction-based Techniques
SHORTEN [Robi94]
The DVD Algorithm [Crav96] [Crav97]
MUSICompress [Wege97]
AudioPak [Hans98a] [Hans98b]
C-LPAC [Qiu01]
Transform-based Techniques
Lossless Transform Coding (LTAC) [Pura97]
IntMDCT [Geig01] [Geig02]
Direct Stream Digital (DSD) Technique [Jans03] [Reef01a] [SACD02] [SACD03]
Meridian Lossless Packing (MLP) [Gerz99] [Kuzu03] [DVD01]
MPEG-4 Lossless Audio Coding [Mori02a] [Sche01] [Vasi02]

- Compression ratio of L^2AC is 2:1-4:1.
- Not for real-time storage processing and Internet streaming

Sub-band coding

General concepts

- **Subband** and **transform** approaches
  - Sub-band approaches use a small number of sub-band filters (time domain processing)
  - Transform approaches use a large number of subbands (frequency domain processing)

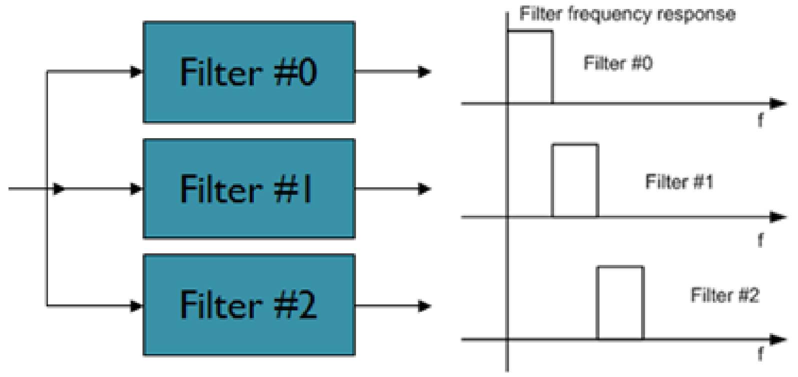


Figure 1(a) Time to frequency domain mapping based on sub-band filtering

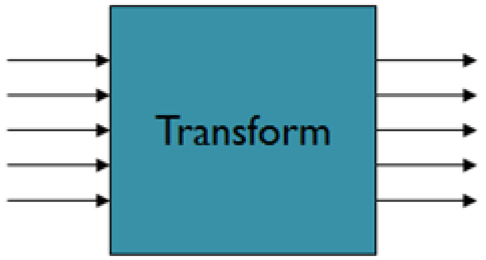
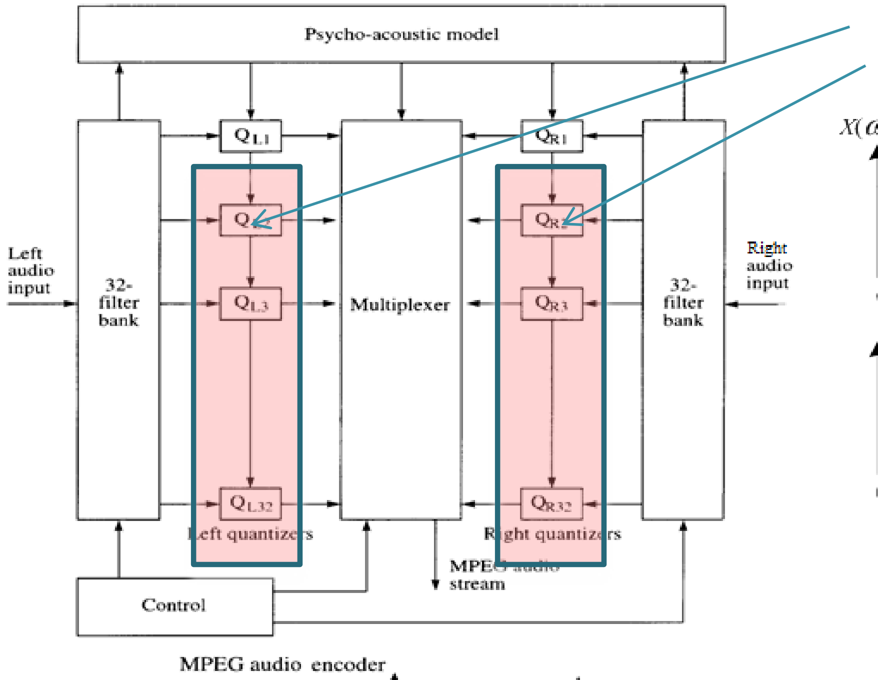


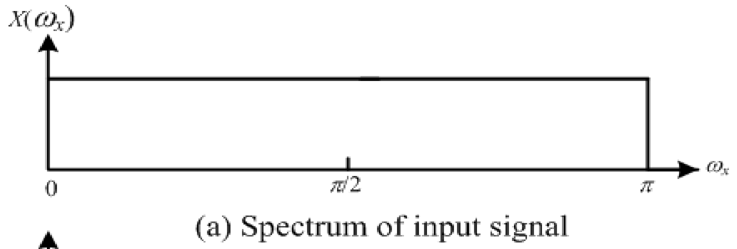
Figure 1(b) Time-to-frequency domain mapping baded on transforms.

General concepts

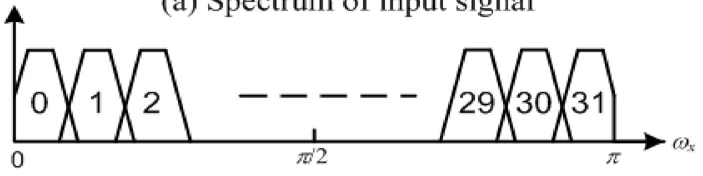
- The signal energy that belongs to a particular band of frequencies can be measured from these parallel outputs
- Because the sub-filters process the input signals whenever they are available, the sub-band filters provide a better time or temporal resolution.
- The transform approach uses FFT or MDCT to convert the time domain signal into many spectrum bins in the frequency domain.
- Because the transform based coders generally provide information of more subbands (or frequency bins), it may require higher computational costs
- It may also have longer processing delay due to the block-based processing style.

Subband Coding

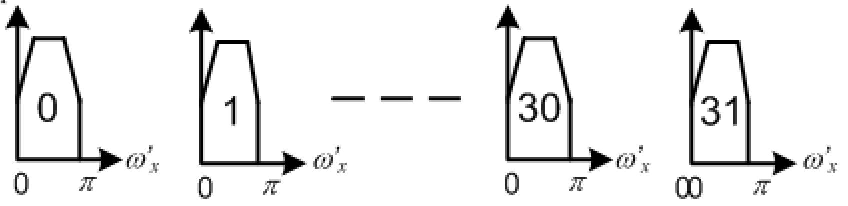




(a) Spectrum of input signal



(b) Filterbank frequency response



(c) Subband frequency response ($\omega'x = 2\pi f/(Fs/32) = 32\times\omega_x$)

- A general block diagram of MPEG audio coder is shown.
- It has left and right channels, each is divided into 32 subbands
- Each subband is quantized individually.

Subband Coding

In the Figure:

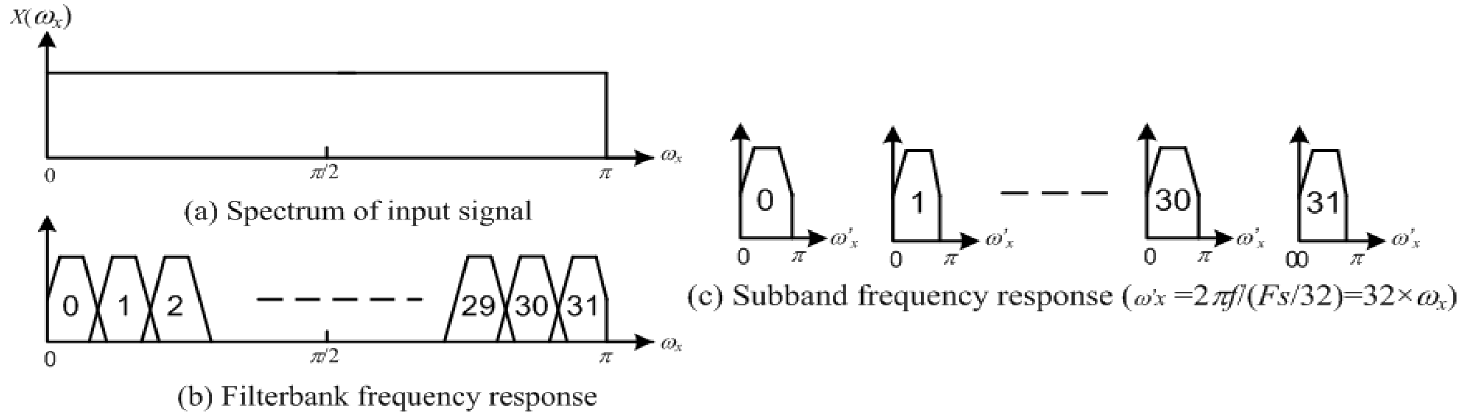
a) The spectrum of the input signal

b) The frequency response of a filter bank

c) Each sub-band is treated as a baseband signal.

- Because the bandwidth of each sub-band is 32 times smaller than the input signal bandwidth, the sampling frequency of the each sub-band should be also 32 times smaller of the input sampling frequency of the filter bank.

Sampling Rate Conversion

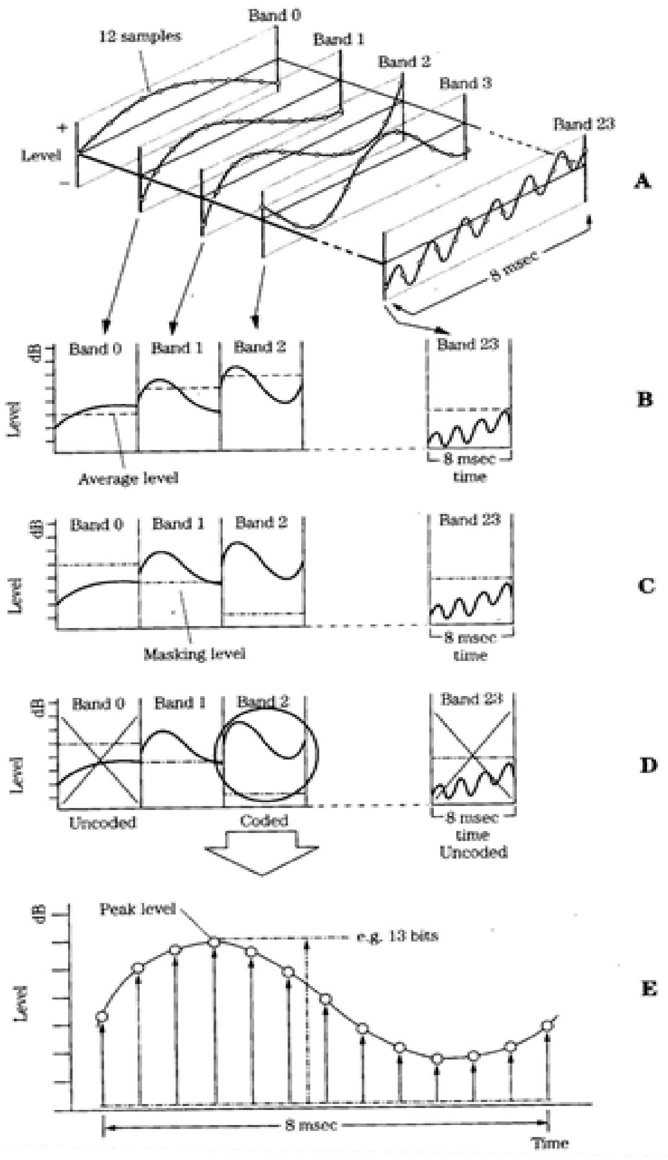
- 

(a) Spectrum of input signal
(b) Filterbank frequency response
(c) Subband frequency response ($\omega'x = 2\pi f/(Fs/32) = 32\times\omega_x$)

- Based on sampling theorem, the minimum bandwidth of each sub-band should be $F'_s = F_s/32$ without any signal distortion for recovery.
- Therefore, a sampling rate decrease by 32 times has to be performed on the filtered sub-band, which also reduces the work loads of quatizers.
- A sampling rate increase by 32 times has also to be performed when these sub-band are assembled at the decoder side.

- http://wiki.benchmarkmedia.com/wiki/index.php/Sample-rate

An Example of Subband Coder



A: 24-band sub-band filter

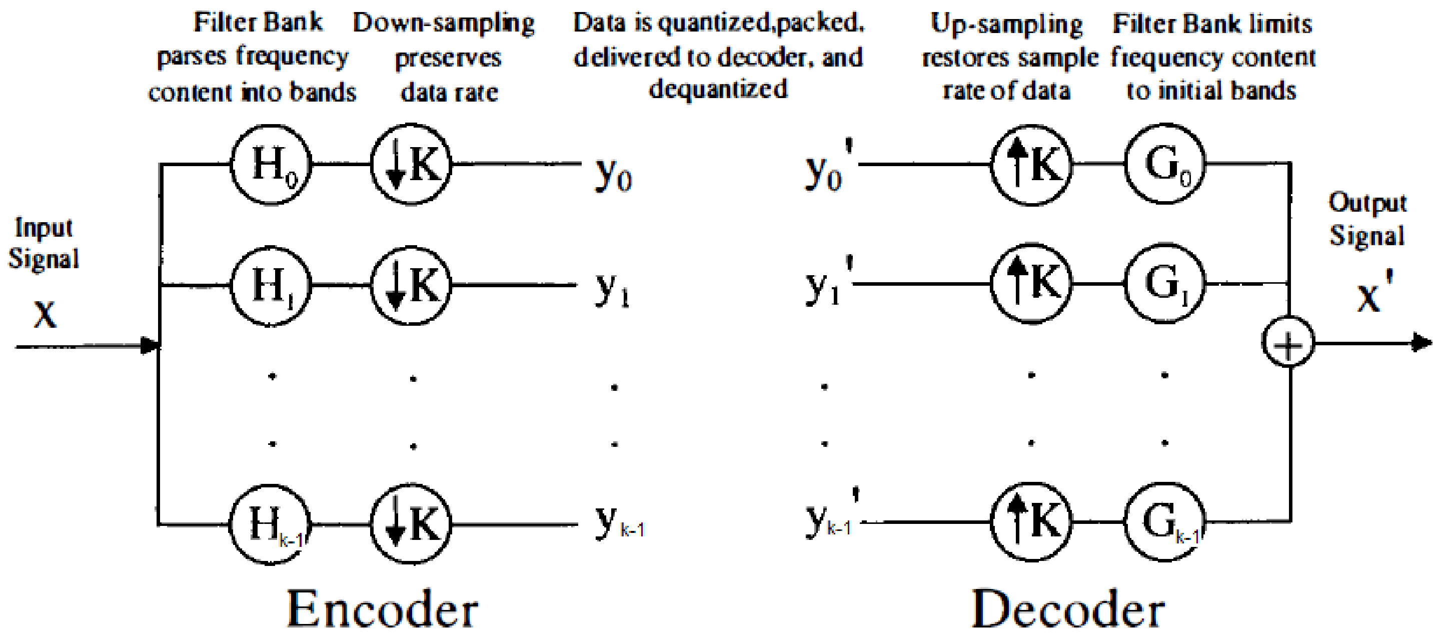
B: Calculate the average level in each subband

C: Calculate masking threshold in each sub-band

D: Subband below masking threshold are NOT coded

E: Bits allocated according to peak level above masking threshold.

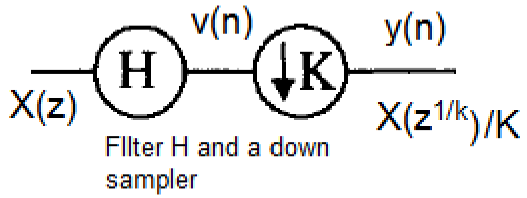
Details of Filter Bank


Filter Bank parses frequency content into bands
Down-sampling preserves data rate
Data is quantized,packed, delivered to decoder, and dequantized
Up-sampling restores sample rate of data
Filter Bank limits frequency content to initial bands
Input Signal
X
H_0
H_1
H_{k-1}
$\downarrow K$
y_0
y_1
y_{k-1}
y_0'
y_1'
y_{k-1}'
$\uparrow K$
G_0
G_1
G_{k-1}
Output Signal
X'
Encoder
Decoder

Overview of the time to frequency mapping process

- K sub-bands
- Down sampling and up sampling are used.
- Filters are needed before the down-sampler and down-sampler
- Hope the input is the same as the output, i.e., $x'(n) = Cx(n-D)$, known as **perfect reconstruction.**

Details of Filter Bank



(a) Decimator

- The decimator effectively decreases the sampling frequency by a factor of *K* without any distortion on the input signal
- The time domain operation of the down-sampler is

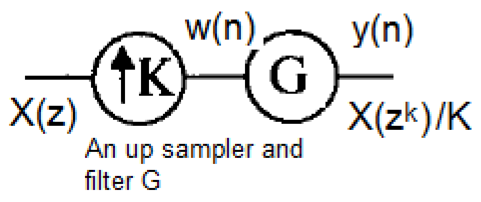
$$y(n) = v(Kn)$$

and $v(n) = x(n) \odot h(n)$, where $\odot$ is the linear convolution and $h(n)$ is the impulse response of the filter $H(z)$.

- If the filter is properly designed, the input output relation of the decimator is described by

$$Y(z) = X(z^{1/k})/K$$

Details of Filter Bank



(b) Interpolator

- The interpolator effectively increase its input sampling frequency by a factor of K without any distortion on the input signal
- The time domain operation of the up-sampler is

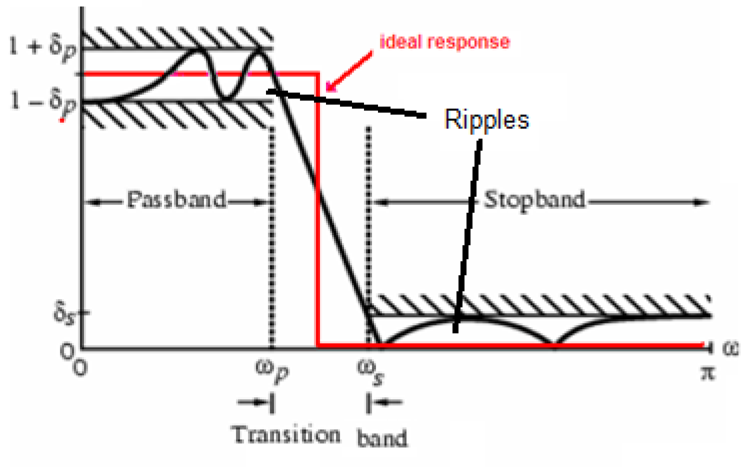
$$w(n) = x(m), \text{if } n = mK$$

otherwise y(n) =0.

- It is equivalently to insert K-1 zero samples between any pare of input samples.
- Then $y(n) = w(n) \odot g(n)$, where $\odot$ is the linear convolution and $g(n)$ is the impulse response of the filter G(z).
- If the filter is properly designed, the input-output relation of the interpolator is described by

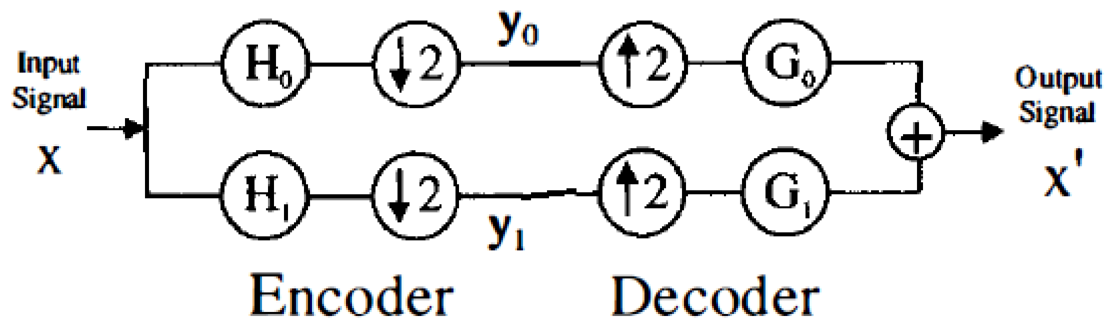
$$Y(z) = KX(z^k)$$

Details of Filter Bank



- Because it is impossible to achieve the ideal filter response, ripples in the passband and stopband, as well as the transitional band distort the input signal.
- It is desired to eliminate the distortions due to these phenomena.
- By collectively considering the filters used in both the encoder and decoder, these undesirable distortions can be eliminated.

Details of Filter Bank



Two-channel perfect reconstruction filter bank

- Assuming the outputs of the encoder, y_0 and y_1, are the inputs of the decoder, the output of the decoder is expressed by

$$X'(z) = Y_0(z^2)G_0(z) + Y_1(z^2)G_1(z) \qquad (1)$$

- The outputs of the encoder are expressed by

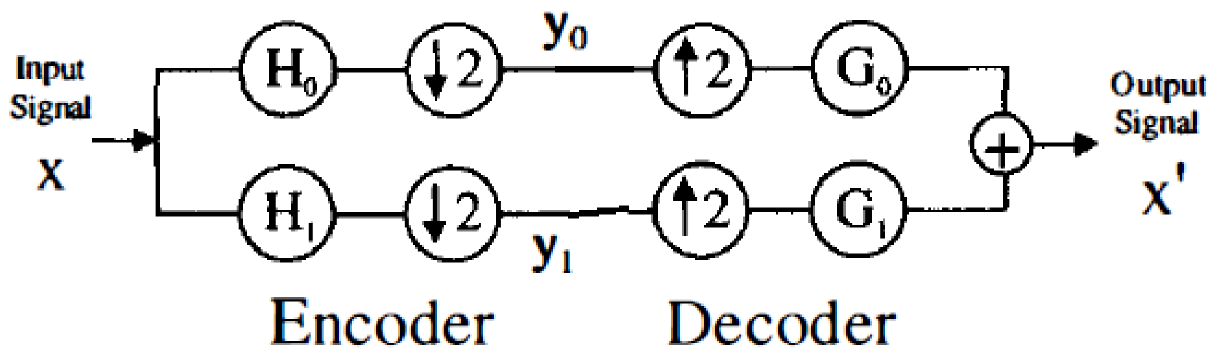
$$Y_i(z) = [H_i(z^{1/2})X(z^{1/2}) + H_i(-z^{1/2})X(-z^{1/2})]/2, \quad i= 0, 1 \qquad (2)$$

- By putting (2) into (1) for i = 0, 1, we finally have

$$X'(z) =\{[H_0(z)G_0(z)+H_1(z)G_1(z)]X(z) \qquad (3)$$
$$+[H_0(-z)G_0(z)+H_1(-z)G_1(z)]X(-z)\}$$

- The second term in (3) involving the aliasing, X(-z), from the filter transitional band and should be eliminated.

Details of Filter Bank



Two-channel perfect reconstruction filter bank

$$X'(z) = \{[H_0(z)G_0(z)+H_1(z)G_1(z)]X(z) + [H_0(-z)G_0(z)+H_1(-z)G_1(z)]X(-z)\} \quad (3)$$

- To make sure $H_0(-z)G_0(z)+H_1(-z)G_1(z)]=0$, we have

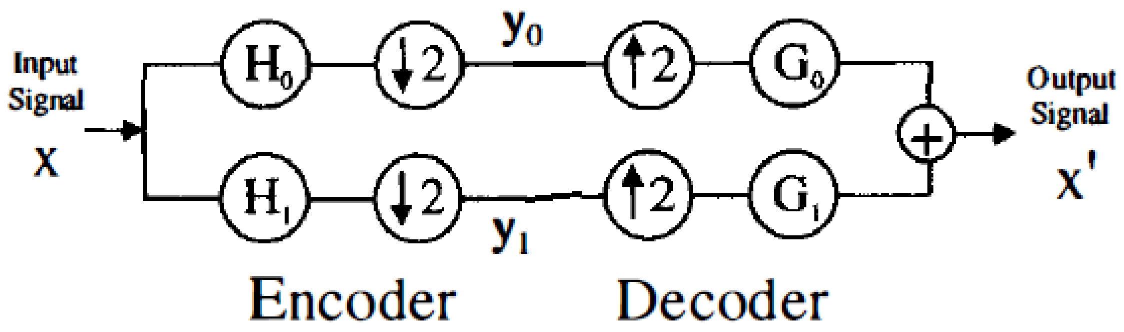
$$G_0(z) = -H_1(-z) \text{ and } G_1(z) = H_0(-z)$$

  which shows the necessary relationships among the filters used in both the analysis and synthesis sections.
- In time domain, these relations are

$$g_0[n] = -(-1)^n h_1[n] \qquad g_1[n] = (-1)^n h[n]$$

  where $g_0[n]$, $g_1[n]$, $h_0[n]$ and $h_1[n]$ are the impulse responses of these filters in the above figure.

Details of Filter Bank



Two-channel perfect reconstruction filter bank

- Equation (3) now becomes

$$X'(z) = [H_0(z)G_0(z)+H_1(z)G_1(z)]X(z) \qquad (3)$$

- To recover the original input signal X(z), equation (3) requires

$$H_0(z)G_0(z)+H_1(z)G_1(z) = \text{Constant} \qquad (4)$$

- Then equation (3) becomes

$$X'(z) = Cz^{-D}X(z) \qquad (5)$$

where *C* is a nonzero real constant and *D* is the a positive integer.

Details of Filter Bank

- Translating equation (5) into the time domain, we have the **perfect reconstruction** shown as

$$x'(n) = Cx(n-D). \qquad (6)$$

- In summary, we have the following results

$H_1(z) = -H_0(-z)$	$h_1(n) = -(-1)^n h_0(n)$
$G_0(z) = H_0(z)$	$g_0(n) = h_0(n)$
$G_1(z) = H_0(-z)$	$g_1(n) = (-1)^n h_0(n)$

- By putting the filter relationships above, equation (4) becomes

$$H_0(z)^2 - H_0(-z)^2 = Cz^{-D} \qquad (7)$$

  where C is non-zero and D is a positive integer.

- Therefore, we have $X'(z) = Cz^{-D}X(z)$, or equivalently $x'(n) = Cx(n-D)$,

NANYANG TECHNOLOGICAL UNIVERSITY

Example

- In the following, we shall select an audio input to observe the dynamic operations of this filter banks.

Demo of 4 channel filter bank

- We shall listen to the original signal and each of the four sub-band signals.
- Visual observation of the time domain waveforms of the original and the sub-band signals are also available.
- The top figure shows the original signal waveform in either the time domain or frequency domain.
- The middle left figure is the lowest sub-band signal, the middle right figure is the next higher frequency sub-band signal,
- The bottom left figure is the next higher frequency sub-band signal and the bottom right figure is the highest frequency sub-band signal.

- It will be seen that the amplitudes of the sub-band signals become significantly smaller as the sub-band frequency becomes higher.

Example

- This demo program demonstrates an input signal is divided into four sub-band by using a filter bank.
- This module allows
  - selecting input signal,
  - designing low pass filter and displaying filter responses in time and frequency domain
  - displaying the time domain wave forms and spectra of the separated sub-band signals
  - listen to the recovered signal, and the separated sub-band signals

Example

- We have the following conclusions from this experiment
  - The sub-band signal from the lowest frequency sub-band alone can be used as low quality audio signal.
  - When the higher frequency sub-band signal is also used, the quality of the recovered signal is improved.
  - When all the sub-band signals are used, the final synthesized signal sounds the same as the original one.
  - Some of the frequency components will be ignored because their magnitude are smaller than our minimum hearing threshold.
- Because the amplitudes of different sub-band signals are different, fewer bits are used for representing higher sub-band signals. This will be performed by coding process to be discussed soon.