Fast Fourier Transform (FFT)

Z. Jerry Shi
Department of Computer Science and Engineering
University of Connecticut

CSE4903: Microprocessor Laboratory

Fourier Transform

- Named after Jean Baptiste Joseph Fourier (1768-1830), a French mathematician and physicist
- The paper was presented in 1807, and published 15 years later
  - After Joseph Louis Lagrange died
- The idea: any continuous periodic signal could be represented as the sum of properly chosen sinusoidal waves

- Four categories:
  - Aperiodic-Continuous (Fourier Transform)
  - Periodic-Continuous (Fourier Series)
  - Aperiodic-Discrete (Discrete Time Fourier Transform)
  - Periodic-Discrete (Discrete Fourier Transform, i.e., DFT)

Example of four types of signals

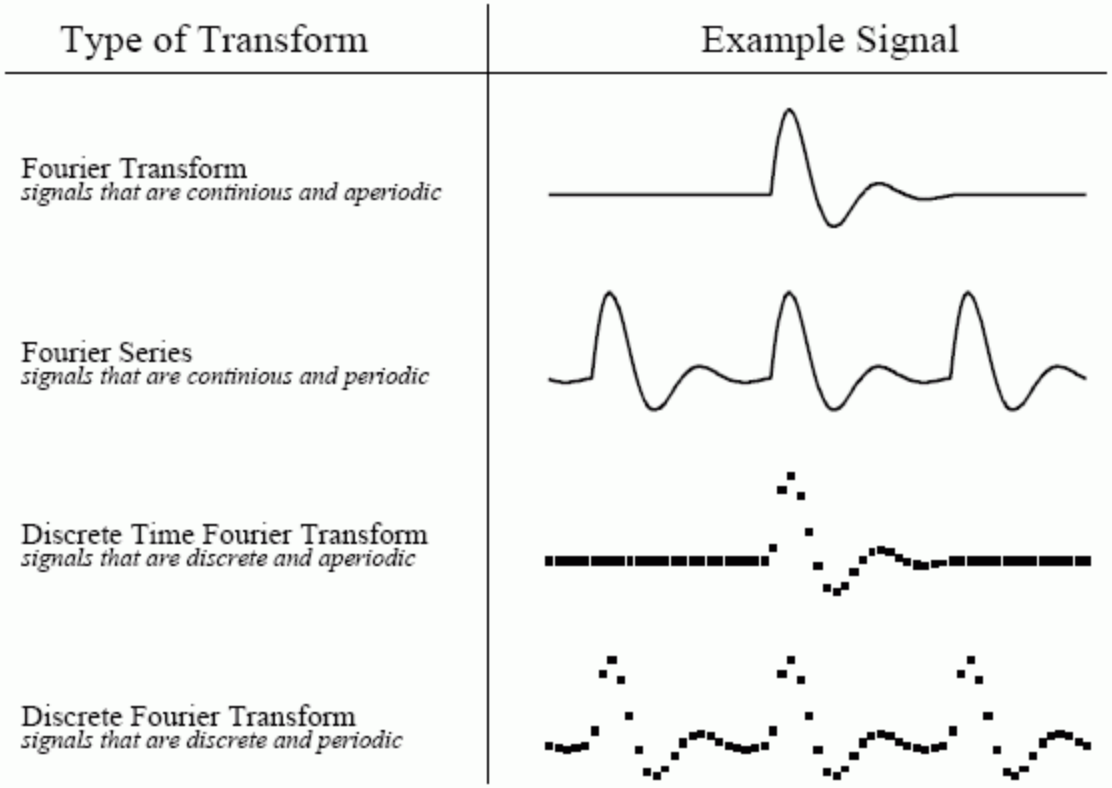
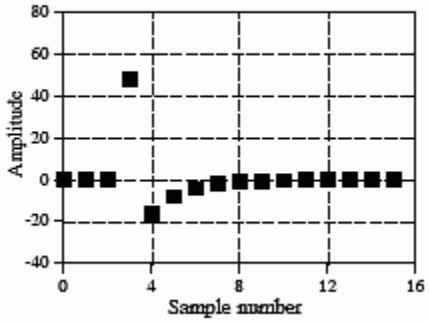


FIGURE 8-2
Illustration of the four Fourier transforms. A signal may be continuous or discrete, and it may be periodic or aperiodic. Together these define four possible combinations, each having its own version of the Fourier transform. The names are not well organized; simply memorize them.

Example of DFT

FIGURE 8-1a
(see facing page)



DECOMPOSE

SYNTHESIZE

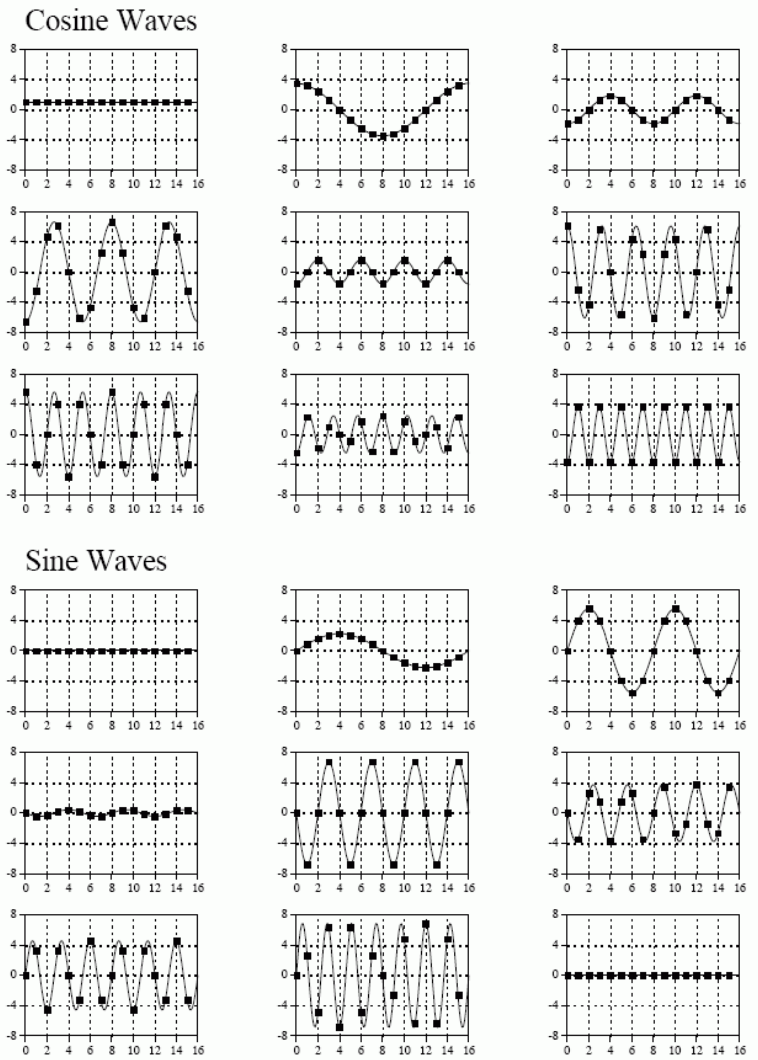


FIGURE 8-1b
Example of Fourier decomposition. A 16 point signal (opposite page) is decomposed into 9 cosine waves and 9 sine waves. The frequency of each sinusoid is fixed; only the amplitude is changed depending on the shape of the waveform being decomposed.

Basic idea on computers

Time domain: x [0..N-1]

Transform to

Frequency domain: R[0..N/2] and I[0..N/2]

N points in x

N/2 + 1 in R and I

R[*k*] reflects the magnitude of cosine wave that has *k* complete cycles

I[*k*] reflects the magnitude of sine wave that has *k* complete cycles

Previous example

Left column: cosine waves

Right column: sine waves

$i = 0 .. (N - 1)$

$c0[i] = E$ (DC offset)

$c1[i] = \cos(2\pi i/N)$

$c2[i] = \cos(4\pi i/N)$

…

$s0[i] = 0$

$s1[i] = \sin(2\pi i/N)$

$s2[i] = \sin(4\pi i/N)$

…

$s_{N/2}[i] = 0$

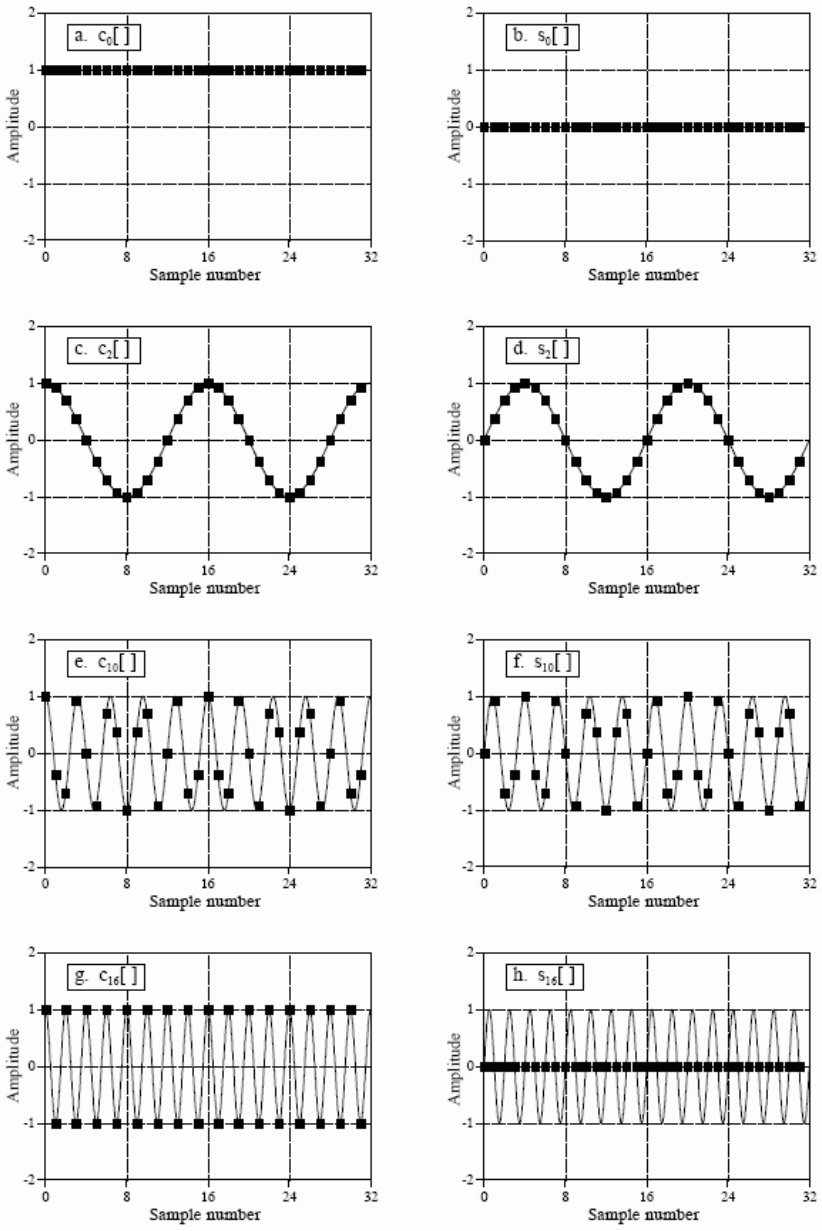


FIGURE 8-5
DFT basis functions. A 32 point DFT has 17 discrete cosine waves and 17 discrete sine waves for its basis functions. Eight of these are shown in this figure. These are discrete signals; the continuous lines are shown in these graphs only to help the reader's eye follow the waveforms.

From frequency domain to time domain

- $x[i]$ can be obtained as

$$x[i] = \sum_{k=0}^{N/2} \bar{R}[k]\cos(2\pi ki / N) + \sum_{k=0}^{N/2} \bar{I}[k]\sin(2\pi ki / N)$$

$$\bar{R}[k] = \frac{R[k]}{N/2}$$

$$\bar{I}[k] = -\frac{I[k]}{N/2}$$

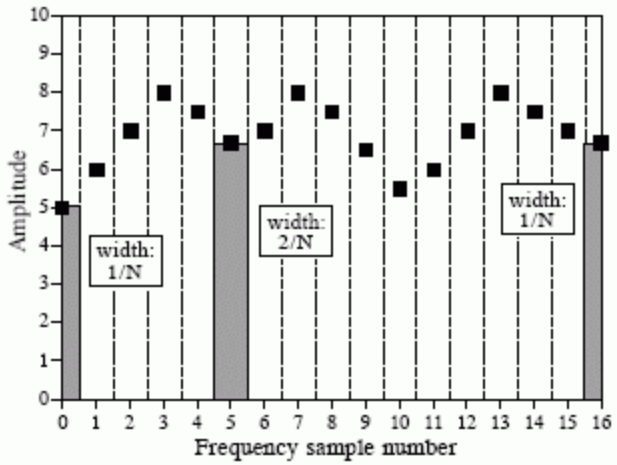
R[0] and R[N/2] have a special factor of N

Now, we have defined the inverse DFT

Why scaling?

- The frequency domain is defined as a spectral density
  - S*pectral density* describes how much signal (amplitude) is present *per unit of bandwidth*

FIGURE 8-7
The bandwidth of frequency domain samples. Each sample in the frequency domain can be thought of as being contained in a frequency band of width $2/N$, expressed as a fraction of the total bandwidth. An exception to this is the first and last samples, which have a bandwidth only one-half this wide, $1/N$.



DFT

- Solve a large equation system
- Correlation

$$\sum_{k=0}^{N} x[k]y[k]$$

$$R[k] = \sum_{i=0}^{N-1} x[i]\cos(2\pi ki / N)$$

$$I[k] = -\sum_{k=0}^{N-1} x[k]\sin(2\pi ki / N)$$

- Fast Fourier Transform (FFT)

Some background

$$e^{jx} = \cos(x) + j\sin(x)$$

$$e^{j0} = e^{j2\pi} = 1$$

$$e^{j\pi/2} = j$$

$$e^{j\pi} = -1$$

$$e^{j3\pi/2} = -j$$

$$e^{j(x+y)} = e^{jx}e^{jy}$$

$$e^{j(x+2\pi)} = e^{jx}$$

$$e^{j(x+\pi/2)} = je^{jx}$$

$$e^{j(x+3\pi/2)} = -je^{jx}$$

Fast Fourier Transform (FFT)

- J.W. Cooley and J.W. Tukey are given credit for bringing the FFT to the world in their paper: "An algorithm for the machine calculation of complex Fourier Series," *Mathematics Computation*, Vol. 19, 1965, pp 297-301.
  - Karl Friedrich Gauss (1777-1855) had used the method more than a century earlier

Definition

Given a sequence of N samples $f[n]$, where $n = 0 \ldots N-1$,
the DFT is defined as $F(k)$, where $k = 0 \ldots N-1$

$$F[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f[n] e^{-j 2\pi kn / N}$$

$$= \frac{1}{\sqrt{N}} \left(\sum_{n=0}^{N-1} f[i] \cos(2\pi kn / N) - \sum_{n=0}^{N-1} f[i] \sin(2\pi kn / N) \right)$$

$f[n]$ can be calculated from $F[k]$ using IDFT:

$$f[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F[k] e^{j 2\pi kn / N}$$

Very often, the scaling factor is ignored in FFT.

Decimation in Frequency (DIF) algorithm

$$FFT(k) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi kn/N}$$

$$= \sum_{n=0}^{N/2-1} f(n) e^{-j2\pi kn/N} + \sum_{n=N/2}^{N-1} f(n) e^{-j2\pi kn/N}$$

$$= \sum_{n=0}^{N/2-1} \left(f(n) + f(n+N/2) e^{-j\pi k} \right) e^{-j2\pi kn/N}$$

$$= \sum_{n=0}^{N/2-1} \left(f(n) + f(n+N/2)(-1)^k \right) e^{-j2\pi kn/N}$$

Divide and conquer

- For even $k = 2k'$:

$$FFT(2k', f) = \sum_{n=0}^{N/2-1} (f(n) + f(n+N/2)(-1)^{2k'}) e^{-j2\pi 2k'n/N}$$

$$= \sum_{n=0}^{N/2-1} (f(n) + f(n+N/2)) e^{-j2\pi k'n/(N/2)}$$

- For odd $k = 2k'+1$:

$$FFT(2k'+1, f) = \sum_{n=0}^{N/2-1} (f(n) - f(n+N/2)) e^{-j2\pi k'n/(N/2)}$$

Decimation In Time (DIT) Algorithm

$$FFT_N(k, f) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi kn/N}$$

$$= \sum_{n=0}^{N/2-1} f(2n) e^{-j2\pi k(2n)/N} + \sum_{n=0}^{N/2-1} f(2n+1) e^{-j2\pi k(2n+1)/N}$$

$$= \sum_{n=0}^{N/2-1} f(2n) e^{-j2\pi kn/(N/2)} + e^{-j2\pi k/N} \sum_{n=0}^{N/2-1} (f(2n+1)) e^{-j2\pi kn/(N/2)}$$

$$= FFT_{N/2}(k, fe) + e^{-j2\pi k/N} FFT_{N/2}(k, fo)$$

$$= FFT_{N/2}(k, fe) + T_N(k) FFT_{N/2}(k, fo)$$

where $fe(n) = f(2n), fo(n) = f(2n+1), T_N(k) = e^{-j2\pi k/N}$

However, only k = 0 .. N/2 – 1 are defined.

Divide and conquer

For $k = 0...N/2-1$:

$$FFT_N(k, f) = FFT_{N/2}(k, fe) + T_N(k) FFT_{N/2}(k, fo)$$

$$FFT_N(k + N/2, f)$$
$$= FFT_{N/2}(k + N/2, fe) + e^{-j2\pi(k+N/2)/N} FFT_{N/2}(k + N/2, fo)$$
$$= FFT_{N/2}(k, fe) - T_N(k) FFT_{N/2}(k, fo)$$

$$FFT_{N/2}(k + N/2, f) = FFT_{N/2}(k, f)$$

$$e^{-j2\pi(k+N/2)/N} = e^{-j\pi} e^{-j2\pi k/N} = -e^{-j2\pi k/N}$$

Example of 4-point DFT

$$\begin{bmatrix} F(0) \\ F(1) \\ F(2) \\ F(3) \end{bmatrix} = \begin{bmatrix} W_0 & W_0 & W_0 & W_0 \\ W_0 & W_1 & W_2 & W_3 \\ W_0 & W_2 & W_0 & W_2 \\ W_0 & W_3 & W_2 & W_1 \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix} = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -j & -1 & +j \\ +1 & -1 & +1 & -1 \\ +1 & +j & -1 & -j \end{bmatrix} \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ f(3) \end{bmatrix}$$

$$T_N(k) = W_k = e^{-j2\pi k/N}$$

$$\text{If } N = 4,$$

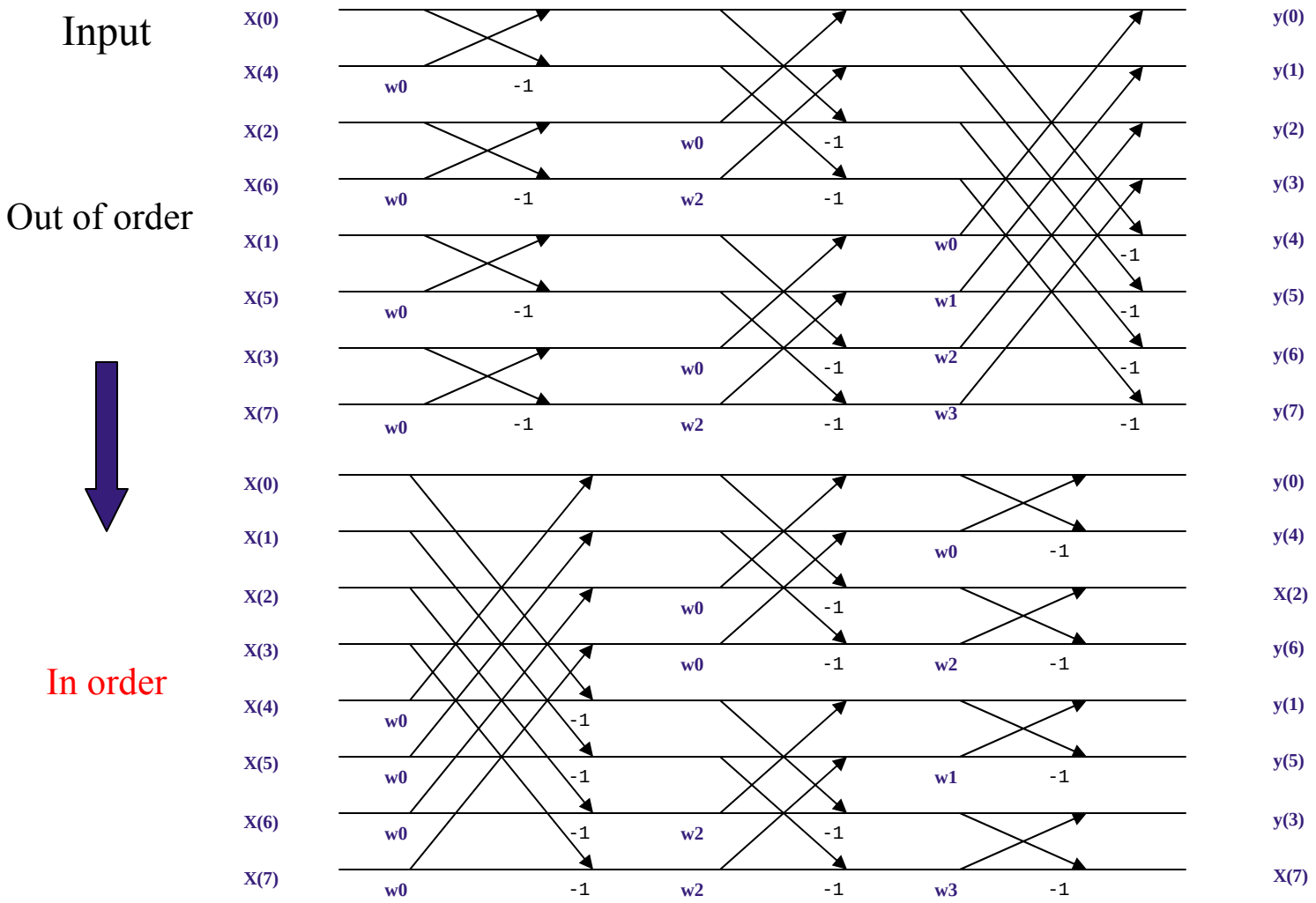
$$W_0 = 1$$

$$W_1 = -j$$

$$W_2 = -W_0 = -1$$

$$W_3 = -W_2 = j$$

Butterfly diagram of FFT



Inverse FFT

- Conjugate the twiddle factors
- Conjugate the input and output
- Use DFT symmetry

Also,

$$IFFT_N(n, FFT_N(k, f)) = Nf(n) \neq f(n)$$

Fixed Point Representation of Real Numbers

- Fixed point vs. floating point
  - The position of point is fixed in fixed point numbers
- Common on fixed-point processors

Consider n bits in Q[qi].[qf] format:

qi bits for integer part

$qf = (n - qi)$ bits for fraction

For example, Q3.5 is an 8-bit value with three integer bits and five fractional bits

Q31 Format

- Acutally it is Q0.31 for 32 bits
  - The MSB is the sign
  - The remaining 31 bits are fractional

- Basically you can consider it is an integer divided by 2^{31}

- So the range that can be represented by Q31 is $[-1, 1 - 1/2^{31}]$ or [– 1, .99999999953433871269226074218 75]

Arithmetic of Q31 in C

```
signed int a, b, result;
result = a+b;
result = a-b;

signed long int temp;
temp = (long int) a * (long int) b;
temp += (1 << 30); // rounding, many do not do it
result= temp >> Q; // move the point

// Overflow is not checked
// Scale after each FFT stage
```

Addition with saturation

```
#define MIN_32 0x80000000
#define MAX_32 0x7fffffff

int sat_add1 (int a, int b)
{  int c;
   c = a + b;
   if (((a ^ b) & MIN_32) == 0) { // a b have same sign
       if ((c ^ a) & MIN_32)  // c has different sign
              c = (a < 0) ? MIN_32 : MAX_32;
   }
   return c;
}
```

References:

- FFTW, http://www.fftw.org/
- FFT Demystified, http://www.engineeringproductivitytools.com/stuff/T0001/