The Fast Fourier Transform

1. FFT Basics

1.1 What is the FFT?

The Fast Fourier Transform (FFT) is simply a fast (computationally efficient) way to calculate the Discrete Fourier Transform (DFT).

1.2 How does the FFT work?

By making use of periodicities in the sines that are multiplied to do the transforms, the FFT greatly reduces the amount of calculation required. Here's a little overview.

Functionally, the FFT decomposes the set of data to be transformed into a series of smaller data sets to be transformed. Then, it decomposes *those* smaller sets into even *smaller* sets. At each stage of processing, the results of the previous stage are combined in special way. Finally, it calculates the DFT of each small data set. For example, an FFT of size 32 is broken into 2 FFTs of size 16, which are broken into 4 FFTs of size 8, which are broken into 8 FFTs of size 4, which are broken into 16 FFTs of size 2. Calculating a DFT of size 2 is trivial.

Here's a slightly more rigorous explanation: It turns out that it is possible to take the DFT of the first N/2 points and combine them in a special way with the DFT of the second N/2 points to produce a single N-point DFT. Each of these N/2-point DFTs can be calculated using smaller DFTs in the same way. One (radix-2) FFT begins, therefore, by calculating N/2 2-point DFTs. These are combined to form N/4 4-point DFTs. The next stage produces N/8 8-point DFTs, and so on, until a single N-point DFT is produced.

1.3 How efficient is the FFT?

The DFT takes N^2 operations for N points. Since at any stage the computation required to combine smaller DFTs into larger DFTs is proportional to N, and there are log2(N) stages (for radix 2), the total computation is proportional to N * log2(N). Therefore, the ratio between a DFT computation and an FFT computation for the same N is proportional to N / log2(n). In cases where N is small this ratio is not very significant, but when N becomes large, this ratio gets very large. (Every time you double N, the numerator doubles, but the denominator only increases by 1.)

1.6 Are FFTs limited to sizes that are powers of 2?

No. The most common and familiar FFTs are "radix 2". However, other radices are sometimes used, which are usually small numbers less than 10. For example, radix-4 is especially attractive because the "twiddle factors" are all 1, -1, j, or -j, which can be applied without any multiplications at all.

Also, "mixed radix" FFTs also can be done on "composite" sizes. In this case, you break a non-prime size down into its prime factors, and do an FFT whose stages use those factors. For example, an FFT of size 1000 might be done in six stages using radices of 2 and 5, since 1000 = 2 * 2 * 2 * 5 * 5 * 5. It might also be done in three stages using radix 10, since 1000 = 10 * 10 * 10.

1.7 Can FFTs be done on *prime* sizes?

Yes, although these are less efficient than single-radix or mixed-radix FFTs. It is almost always possible to avoid using prime sizes.

2. FFT Terminology

2.1 What is an FFT "radix"?

The "radix" is the size of an FFT decomposition. In the example above, the radix was 2. For single-radix FFTs, the transform size must be a power of the radix. In the example above, the size was 32, which is 2 to the 5th power.

2.2 What are "twiddle factors"?

"Twiddle factors" are the coefficients used to combine results from a previous stage to form inputs to the next stage.

2.3 What is an "in place" FFT?

An "in place" FFT is simply an FFT that is calculated entirely inside its original sample memory. In other words, calculating an "in place" FFT does not require additional buffer memory (as some FFTs do.)

2.4 What is "bit reversal"?

"Bit reversal" is just what it sounds like: reversing the bits in a binary word from left to right. Therefore the MSBs become LSBs and the LSBs become MSBs. But what does that have to do with FFTs? Well, the data ordering required by radix-2 FFTs turns out to be in "bit reversed" order, so bit-reversed indexes are used to combine FFT stages. It is possible (but *slow*) to calculate these bit-reversed indices in software; however, bit reversals are trivial when implemented in hardware. Therefore, almost all DSP processors include a hardware bit-reversal indexing capability (which is one of the things that distinguishes them from other microprocessors.)

2.5 What is "decimation-in-time" versus "decimation-in-frequency"?

FFTs can be decomposed using DFTs of even and odd points, which is called a Decimation-In-Time (DIT) FFT, or they can be decomposed using a first-half/second-half approach, which is called a "Decimation-In-Frequency" (DIF) FFT. Generally, the user does not need to worry which type is being used.

3. FFT Implementation

3.1 How do I implement an FFT?

Except as a learning exercise, you generally will never have to. Many good FFT implementations are available in C, Fortran and other languages, and microprocessor manufacturers generally provide free optimized FFT implementations in their processors' assembly code, Therefore, it is not so important to understand *how* the FFT really works, as it is to understand how to *use* it.

3.2 I'm not convinced. I want to really learn how the FFT works.

(Gosh you're difficult!) Well, virtually every DSP book on the planet covers the FFT in detail. However, if you want to read something online right now, see The Scientists and Engineer's Guide to DSP.

3.3 OK, I read that stuff. Let's cut to the chase. Where do I get an FFT implementation?

1. If you want an assembly language implementation, check out the web site of the manufacturer of your chosen DSP microprocessor. They generally provide highly optimized assembly implementations in their user's guides and application manuals, and also as part of the library of their C compilers.
2. Here are a couple of the best C implemenations:
    - MixFFT - This Mixed-radix FFT implemenation by Jens Joergen Nielsen is fast, flexible, and easy to use.
    - FFTW - The "Fastest Fourier Transform in the West".
3. There are several great FFT hotlists on the net. One of the best is jj's useful and ugly FFT page, and FFTW also has a very good FFT links page.