TeleBench™ Version 1.1

Benchmark Name: Fast Fourier Transform (FFT)

Highlights

- **Implements a decimation in time 256 fixed point 16 bit FFT using a Butterfly technique**
- **Allows for interleaved or non-interleaved data**
- **Typical FFT used in Telecommunications applications (e.g. a mobile/cellular phone)**
- **Multiple (3) data sets: sine, pulse, and high frequency**
- **FFT is benchmarked; Inverse FFT (iFFT) is included in the code for analysis purposes only**
- **Integer Math with complexity; fits inside small L1 caches.**

Application

The Fast Fourier transform benchmarks perform tests of a very fundamental algorithm that underlies a wide variety of signal processing applications. A Fourier transform performs a frequency analysis of a signal and therefore can be used for filtering frequency-dependent noise or interference of a transmission, for identifying the information content of a frequency-modulated signal, and many other purposes. A good general reference for Fourier transforms, algorithms for computing them, and some of their signal processing applications may be found in *The Digital Signal Processing Handbook*, Vijay K. Madisetti and Douglas B. Williams, Eds. (CRC Press, Boca Raton, FL, 1998). **The benchmark provides an indication of the potential performance of a microprocessor in a core task used in a wide variety of telecommunications applications.**

Benchmark Description

The FFT benchmarks apply to discrete data, which may be obtained for example from an analog-to-digital converter applied to a continuous signal. All benchmark FFTs use decimation in time and are performed on 256 16-bit complex points. All data are in fixed-point format, and therefore scaling must be performed, as needed, to prevent arithmetic overflow. Three different varieties of input data are used: a square pulse, a high-frequency test module, and a sine wave of a certain frequency. Benchmark scores for each variety are reported separately. The initial bit-reversal step is explicitly included.

A Fourier transform operates on the principle that a set of N stochastic input data points can be Fourier expanded in terms of N orthogonal exponentials of period N, taken as $\exp[-j(2\pi/N)kn]$. The n^{th} element of the data is expressed as a sum over k=0,…,N-1 of these trigonometric factors (known as twiddle factors), each multiplied by a frequency coefficient. In most cases the data are available and the frequency coefficients are desired. These are obtained by a similar expansion in terms of the data; this direction is known as the *forward* transform. The absolute squares of these coefficients specify the strength of each frequency in the variations of the data. By convention the input data are said to lie in the time domain, such that each data point comes at a fixed time interval from the preceding. The output coefficients then are said to lie in the frequency domain. An FFT algorithm will produce N such coefficients separated by a fixed frequency interval.

For example, if all data points had exactly the same value then the only non-zero frequency coefficient would be the one at zero frequency (the DC component), since no variation is present.

A fast Fourier transform takes advantage of the fact that the trigonometric factors repeat periodically. Therefore partial sums can be formed that can be reused many times, so an FFT will take an amount of time proportional to Nlog(N) instead of N^2, as brute-force computation of a Fourier transform to obtain N coefficients from N data points would require. The base to which the logarithm is taken depends on the algorithm employed. Very many algorithms are available, depending primarily on the number of points N. Many common applications rely on the fact that a single Fourier transform can be decomposed into two equal-sized transforms that are combined at the end. Proceeding in this manner to repeatedly subdivide the data in halves, one will arrive at

a set of simple two-component transforms that must be combined, provided the number N of data points is a power of 2. Each subdivision is a "stage;" there will be $\log_2(N)$ stages in the computation using this technique. This is the basis of the "radix-2" algorithm, the implementation used in the EEMBC Out-of-the-Box FFT algorithm.

A "bit-reversal" re-ordering step is required to complete an FFT because the output coefficients do not otherwise occur in increasing frequency order. This step may be performed on the input data ("decimation in time," DIT) or on the output frequencies ("decimation in frequency," DIF). All EEMBC benchmarks use DIT.

The execution speed of an FFT has had a revolutionary impact on the digital signal-processing industry. The FFT is a fundamental component of very many signal-processing applications.

This benchmark performs an FFT with three different assumptions on the shape of the input data. These shapes are a sine wave, a square pulse, and a high-frequency test module. The shape may or may not affect the timing and the accuracy of the output. All input data is 16-bit complex and the FFTs are performed on N=256 points.

The twiddle factors are supplied in the test harness. Input data and/or twiddle factors may have real and imaginary parts interleaved or sequential, as specified by C preprocessor parameters (i.e., defined at compile time). Whichever choice is made, both have to be treated the same. These choices do not affect timing. Default is for both to be interleaved. The bit-reversal indices are also pre-computed and supplied as part of the test harness.

An inverse Fourier transform is also possible (see separate datasheet). This computation is very similar except that it begins with N equally-spaced frequency coefficients and returns N equally-spaced time-domain data. A Fourier transform followed by its inverse should yield the original data, unchanged except for computation errors. The default for all EEMBC benchmarks is in the forward direction. The direction may be set by a preprocessor parameter.

Analysis of Computing Resources

The forward FFT benchmark performs integer math on 16-bit signed quantities (the time-domain input data and twiddle factors). Both data and twiddle factors are assumed to be complex and will therefore each require 256×2 16-bit locations in cache or memory; the output frequency coefficients will require the same amount.

The code size is small and fits ina small L1 instruction cache.

It is left to the user to run the benchmark through enough iterations to amortize the overhead associated with the test harness and initial cache misses. The default for this benchmark is 1000 iterations. ECL will double-check this with other values. Assumptions about data values would be imprudent, as ECL has its own private data sets.

Special Notes

1. Each of data files must be run to obtain an EEMBC Telemark™ score.