It is unlikely that many readers of this chapter will ever have to implement their own fast Fourier transform software, except as a learning exercise. The computation of the DFT, much like basic linear algebra or integration of ordinary differential equations, is so central to numerical computing and so well-established that robust, flexible, highly optimized libraries are widely available, for the most part as free/open-source software. And yet there are many other problems for which the algorithms are not so finalized, or for which algorithms are published but the implementations are unavailable or of poor quality. Whatever new problems one comes across, there is a good chance that the chasm between theory and efficient implementation will be just as large as it is for FFTs, unless computers become much simpler in the future. For readers who encounter such a problem, we hope that these lessons from FFTW will be useful:

• Generality and portability should almost always come first.
• The number of operations, up to a constant factor, is less important than the order of operations.
• Recursive algorithms with large base cases make optimization easier.
• Optimization, like any tedious task, is best automated.
• Code generation reconciles high-level programming with low-level performance.

We should also mention one final lesson that we haven't discussed in this chapter: you can't optimize in a vacuum, or you end up congratulating yourself for making a slow program slightly faster. We started the FFTW project after downloading a dozen FFT implementations, benchmarking them on a few machines, and noting how the winners varied between machines and between transform sizes. Throughout FFTW's development, we continued to benefit from repeated benchmarks against the dozens of high-quality FFT programs available online, without which we would have thought FFTW was “complete” long ago.
Acknowledgements

SGJ was supported in part by the Materials Research Science and Engineering Center program of the National Science Foundation under award DMR-9400334; MF was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract No. NBCH30390004. We are also grateful to Sidney Burrus for the opportunity to contribute this chapter, and for his continual encouragement—dating back to his first kind words in 1997 for the initial FFT efforts of two graduate students venturing outside their fields.

Contributor

- ContribEEBurrus