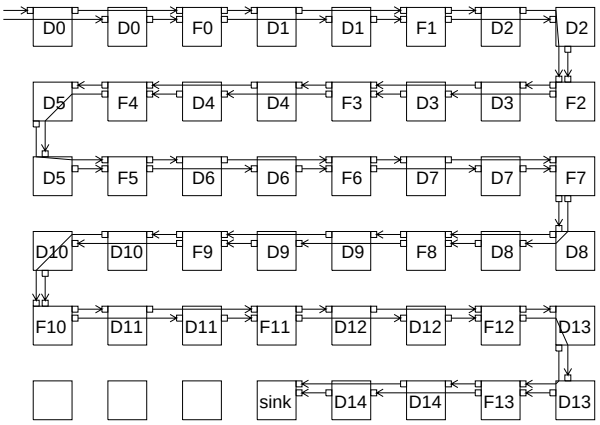

Figure 6: 16K-point pipelined phase-rotation FFT running at 40 Mbytes/sec (350 MFLOPS) on iWarp

performs 16 floating-point operations per iteration per stage (2 adds and 4 multiplies by each twiddle operator, and 4 adds by the kernel operator). But the standard formula for computing FFT MFLOPS is $5N \log N$ floating-point operations per N-point FFT [3], which implies 10 floating-point operations per iteration per stage. Therefore, in order to do fair comparisons with other FFT algorithms, we must compute the phase rotation performance using the standard of 10 floating-point operations per iteration per stage, even though the phase-rotation FFT is actually performing 16 floating-point operations per iteration per stage.

Since each node program executes its computation in at most 8 clocks, and since each clock is 50 nanoseconds, each stage of the iWarp phase-rotation FFT runs at a rate of

$$\frac{10 \text{ fp operations}}{8 \text{ clocks}} \times \frac{1 \text{ clock}}{50 \text{ nanoseconds}} \times \frac{1 \times 10^9 \text{ nanoseconds}}{1 \text{ second}} = 25 \text{ MFLOPS}$$

for a total performance over all of the $\log N$ stages of $25 \log N$ MFLOPS. For example, the 16K-point FFT in Figure 6 achieves a measured performance of $25 \times 14 = 350$ MFLOPS (single precision) on iWarp. As a point of comparison, a highly optimized 16K-point FFT has been measured at 237 MFLOPS (double precision) on a single-processor Cray Y-MP [3, p.114]. The numbers are not directly comparable because of the different floating-point precisions, but they do suggest that the absolute performance of the phase-rotation FFT on iWarp is quite good.

6. Concluding remarks

We have described an improved version of the Whelchel pipelined phase-rotation FFT, developed recipes for generating the appropriate twiddles and shuffle indices off-line and directly in terms of the parallel pipeline, outlined mapping approaches for the radix-2 case on the iWarp parallel computer, and presented measured performance results of an implementation on iWarp.

The improvement on the original phase-rotation FFT is significant in that it eliminates a potential pipeline bottleneck during the digit reversing step at the end. The twiddle and shuffle recipes should be helpful to the programmer who wants to implement the pipelined phase rotation FFT. The iWarp implementation

validates a simple and realistic approach for building scalable pipelined FFT's on a programmable parallel system. Further, the implementation demonstrates that, given a balanced parallel computer architecture with word-level access to the communication links, it is possible to build FFT's that run at the full link bandwidth of the links, even when the FFT's are relatively small.

Acknowledgements

We would like to thank Tom Warfel and LeeAnn Tzeng for their help with the iWarp implementation, and Doug Noll and Doug Smith for discussions that led to the more node-efficient mapping.

References

[1] BORKAR, S., COHN, R., COX, G., GLEASON, S., GROSS, T., KUNG, H. T., LAM, M., MOORE, B., PETERSON, C., PIEPER, J., RANKIN, L., TSENG, P. S., SUTTON, J., URBANSKI, J., AND WEBB, J. iWarp: An integrated solution to high-speed parallel computing. In *Supercomputing '88* (Nov. 1988), pp. 330–339.

[2] BORKAR, S., COHN, R., COX, G., GROSS, T., KUNG, H. T., LAM, M., LEVINE, M., MOORE, B., MOORE, W., PETERSON, C., SUSMAN, J., SUTTON, J., URBANSKI, J., AND WEBB, J. Supporting systolic and memory communication in iWarp. In *Proceedings of the 17th Annual International Symposium on Computer Architecture* (Seattle, WA, May 1990), pp. 70–81.

[3] CARLSON, D. Ultra-performance FFTs for the CRAY-2 and CRAY Y-MP supercomputers. *Journal of Supercomputing 6* (1992), 107–115.

[4] COOLEY, J., AND TUKEY, J. An algorithm for the machine computation of complex Fourier series. *Mathematics of Computation 19* (Apr. 1965), 297–301.

[5] CORINTHIOS, M. The design of a class of Fast Fourier Transform computers. *IEEE Transactions on Computers C-20* (June 1971), 617–623.

[6] MCCLELLAN, J., AND PURDY, R. Radar signal processing. In *Applications of Digital Signal Processing*, A. Oppenheim, Ed. Prentice-Hall, Englewood Cliffs, NJ, 1978.

[7] PEASE, M. An adaptation of the Fast Fourier Transform for parallel processing. *Journal of the Association for Computing Machinery 15* (1968), 252–264.

[8] VAN LOAN, C. *Computational Frameworks for the Fast Fourier Transform*. SIAM, Philadelphia, PA, 1992.

[9] WHELCHEL, J., O'MALLEY, J., RINARD, W., AND MCARTHUR, J. The systolic phase rotation FFT - a new algorithm and parallel processor architecture. In *Proceedings of ICASSP '90* (Apr. 1990), pp. 1021–1024.

[10] WITHERS, JR., L., AND WHELCHEL, J. The multidimensional phase-rotation FFT - a new parallel architecture. In *Proceedings of ICASSP '91* (May 1991), pp. 2889–2892.