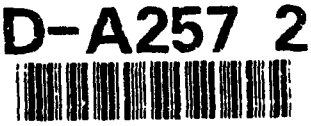

(1)

A Pipelined, High-Precision FFT Architecture

MTP 92B0000004
October 1992

Thomas M. Hopkinson
G. Michael Butler

MITRE
Bedford, Massachusetts

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1992	

4. TITLE AND SUBTITLE
A Pipelined, High-Precision FFT Architecture

5. FUNDING NUMBERS

6. AUTHOR(S)
Thomas M. Hopkinson
G. Michael Butler

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730-1420

8. PERFORMING ORGANIZATION REPORT NUMBER
MTP 92B0000004

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
same as above

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
same as above

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This paper presents a highly-integrated, high-precision FFT architecture. A 1.2 μm CMOS implementation of this architecture has yielded a 32-bit, 64K-point FFT that operates at a continuous 4-million-samples-per-second data rate. All FFT support functions, including coefficient generation and memory interfacing, are included on-chip.

14. SUBJECT TERMS
FFT architecture, CMOS implementation

15. NUMBER OF PAGES
15

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	unlimited

A Pipelined, High-Precision FFT Architecture

MTP 92B0000004
October 1992

Thomas M. Hopkinson
G. Michael Butler

Contract Sponsor ESC
Contract No. F19628-89-C-0001
Project No. 7170
Dept. D05C

MITRE
Bedford, Massachusetts

ABSTRACT

This paper presents a highly-integrated, high-precision FFT architecture. A 1.2μm CMOS implementation of this architecture has yielded a 32-bit, 64K-point FFT that operates at a continuous 4-million-samples-per-second data rate. All FFT support functions, including coefficient generation and memory interfacing, are included on-chip.

PREFACE

This paper was originally presented at the 35th Midwest Symposium on Circuits and Systems in Washington, DC, August 1992.

ACKNOWLEDGMENTS

The authors are indebted to Paul Capozza and Kevin Moore for their VLSI implementation of the radix-2 processor, and to Paul Capozza and Doug Kolb for the VLSI implementation of the radix-4 processor. Additionally, the authors would like to thank Bruce Johnson for his continuing encouragement and technical leadership. This work was supported by the United States Air Force's Electronic Systems Center under contract F19628-89-C-0001. The program office was the Surveillance and Photonics Directorate of Rome Laboratory (RL/OC).

TABLE OF CONTENTS

LIST OF FIGURES

SECTION 1

INTRODUCTION

The fast Fourier transform (FFT) class of algorithms [1] is widely used in communication and sensor signal processing. Several communication and sensor applications require very high precision (32-bit) real-time Fourier transforms of large (64K-point), complex data blocks. One such application is a high-frequency, spread spectrum communication system as described in [2]; radar systems designed to detect small cross-sectional targets are similarly demanding applications. Although the FFT algorithm is readily implemented with commercial digital signal processing (DSP) components, those components lack either the throughput or precision required for these applications. This paper presents an FFT processor architecture designed to satisfy the precision and throughput requirements of a class of applications. We begin with a review of the inadequacies of commercially available components. We then propose an alternative, highly-integrated architecture, provide design details on portions of the architecture, and conclude with a discussion of a radix-4 implementation of this architecture.

FFT processors are readily constructed from commercially available integrated circuits. There are, essentially, three approaches available: (1) use a programmable DSP component such as the TMS320, (2) use the commercially available "single-chip" FFT processors, or (3) construct an FFT processor from available arithmetic components such as ALUs.

Programmable DSP components, such as the TMS320, provide high-precision computation in a very flexible form. Their flexibility and performance have allowed these programmable components to subsume many DSP applications. However, their flexibility comes at the expense of throughput; the DSP chips are not well suited to real-time computation at modest or high throughput rates.

An alternative to programmable DSP components is commercial "single-chip" FFT processors. These components meet the throughput requirements of high-performance applications, but they lack the necessary precision. Many such components provide only 16 bits of precision, while a few others offer 24 bits. Secondly, these "single-chip" processors typically require a large number of supporting components; in particular, address generators and coefficient memories are not incorporated on-chip. Finally, processor throughput and FFT block size are tightly coupled in these processors—larger blocks are typically processed at lower throughput rates.

A third option is the construction of a high-precision FFT processor from commercially available "building blocks," such as high-performance ALUs. This approach provides both precision and performance, but the resulting system is large and inflexible. Using this approach, we constructed a 32-bit, 16K-point FFT in 1990. The processor required nearly 300 components and could not be readily extended to larger block sizes or throughput rates.

Application-specific integrated circuits (ASICs) provide the only feasible solution to this dilemma. ASIC integration allows flexible, high-precision, high-performance FFT processors to be realized. In this paper, an FFT architecture that differs significantly from commercially available "single-chip" FFT processors is presented. Unlike the commercial offerings, this architecture incorporates all FFT support functions—including coefficient and memory-address generation—on a single die. Additionally, our processor computes the transform to full 32-bit precision, significantly greater precision than currently available with commercial processors. Our current implementation operates with a continuous complex-data rate of 4 million samples-per-second (MSPS) and can be cascaded to provide up to 64K-point transforms. The architecture is easily extensible to a 20-MSPS processor with no compromise in precision or block length.

SECTION 2

PIPELINED FFT ARCHITECTURE

Recall that the discrete Fourier transform, $X[k]$, of a series of N samples $x[n]$ is given by

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi nk}{N}}.$$

As shown, the computation of a block of length N requires $O(N^2)$ arithmetic operations. A radix-r fast Fourier formulation of this calculation reduces this number to $O(N \log_2 N)$ arithmetic operations in $\log_r N$ stages. This computation is suggested in figure 1 for $r = 2$.

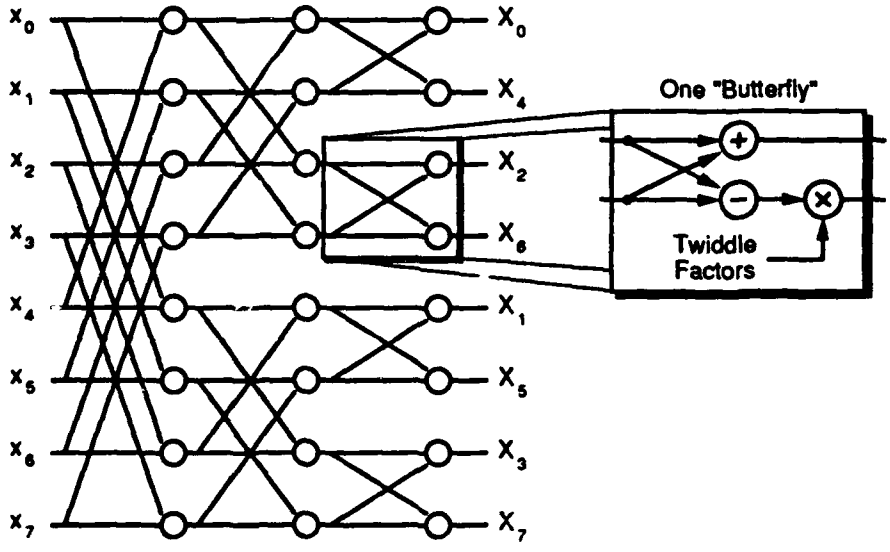

Figure 1. FFT Signal Flow Graph

Commercial "single-chip" FFT processors generally provide all $O(N \log_2 N)$ operations in a single component. This is advantageous in situations where N is relatively small and throughput rates are low. In this case, the computational elements, or "butterfly," can easily be time multiplexed. For larger values of N, this approach is viable only if a decrease in throughput can be tolerated (i.e., in a "single-chip" processor, throughput and block length are inversely proportional). This architecture requires a very fast butterfly, but compromises integration—all available silicon area is devoted to the butterfly, and none can be spared for support functions. In addition, the architecture's I/O bandwidth requirements scale with increasing N. Typically, this increased bandwidth is provided by additional signal pins on the processor chip and places stringent bandwidth requirements on the buffer memories as well. It is easily seen, then, that this "single-chip" architecture offers high performance at the expense of precision and integration—support functions such as coefficient generation and memory addressing must be provided off-chip.

An alternative approach is to partition the $O(N \log_2 N)$ arithmetic operations among $\log_r N$ processors for the radix-r FFT. This scheme reduces the computational and I/O requirements of the processor by a factor of $\log_r N$ and effectively decouples block length and throughput. At first, this approach seems unappealing since it requires $\log_r N$ processors, but this approach does provide a smaller overall system.

Reduced processing and I/O requirements allows greater architectural flexibility. To improve precision and integration, we chose to reduce the silicon area devoted to the butterfly's processing components. Reduction was accomplished by the application of digit-serial arithmetic [3] and iterative multiplication architectures [4] wherever possible. An additional area savings was realized by the adaptation of a distributed arithmetic multiplier architecture presented in [5]. The multiplier architecture will be presented in more detail shortly. These techniques enable a butterfly data path that provides high precision processing in a very modest area. Our radix-2 and radix-4 implementations of this architecture incorporate a 32-bit complex FFT butterfly and all support functions—including a 42-bit complex coefficient generator and a buffer memory interface—on a single die. The remainder of the paper will focus on our radix-4 implementation of this architecture.

SECTION 3

RADIX-4 PIPELINED FFT

The radix-4 FFT of an N-point data block consists of $\log_4 N$ stages of processing. At each stage, groups of four data samples are gather-read from the input memory buffer, operated on, and scatter-written to the output buffer. Each interstage memory serves as output buffer for one processor and input buffer to the next. Transforms of length $N = 4^n, N \leq 64K$, are constructed by cascading n identical stages, each stage consisting of a processor chip and one commercial static random-access memory (SRAM) as shown in figure 2. Note that the FFT chips implement a unidirectional pipeline—the double-buffered memories traditionally associated with the FFT computation are not required. The interstage memories allow the reordering of data as it progresses through the pipeline.

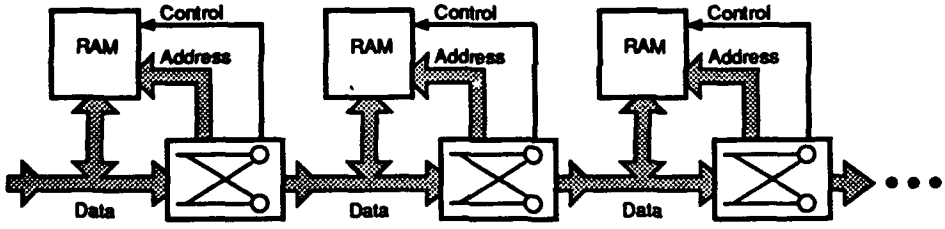

Figure 2. Pipelined FFT Approach

Two points should be noted about figure 2. First, since the FFT ICs are programmable, a single design can serve as any stage of a forward or inverse FFT. Second, since all support functions are included on-chip, no ancillary control or support devices are needed—there are no coefficient ROMS, no external address generators, and no memory controllers required. The result is a simple, regular FFT system implemented with precisely two IC types: a commercially available SRAM module and the custom FFT processor.

RADIX-4 BUTTERFLY

Figure 3 shows a block diagram of the radix-4 decimation-in-frequency [1] FFT IC. In addition to the butterfly's arithmetic components, figure 3 shows a coefficient generator, which calculates the root-of-unity "twiddle factors," and a buffer-memory controller. Input and output registers convert between an on-chip digit-serial data format and the word-parallel format used for chip-to-memory communication.

Prior to each calculation, a four-tuple, (a, b, c, d), is transferred from the buffer memory to the processor's input registers. Together, these values represent 256 bits of data and are transferred over a 32-bit input data bus in eight memory read cycles. In a forward FFT, the radix-4 butterfly computes a new vector, (a', b', c', d'), given by:

$$\begin{aligned} a' &= a + b + c + d, \\ b' &= (a - jb - c + jd)(e^{-j2\pi/N})^{nk}, \\ c' &= (a - b + c - d)(e^{-j2\pi/N})^{n2k}, \\ d' &= (a + jb - c - jd)(e^{-j2\pi/N})^{n3k}. \end{aligned}$$

The computation of the inverse FFT is the complex conjugate of the above.

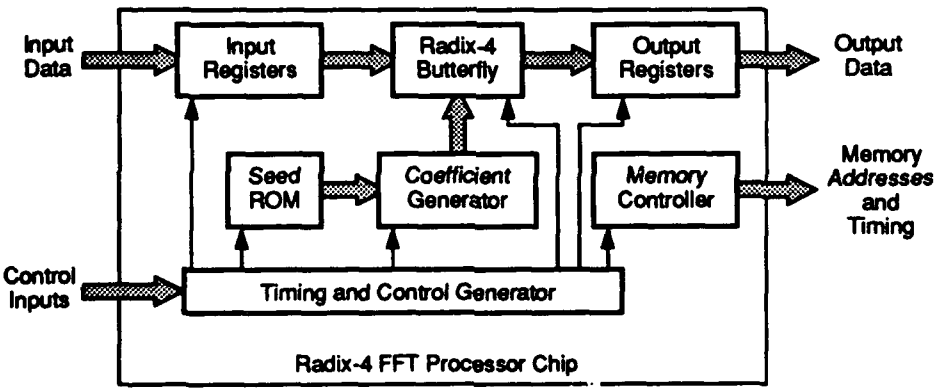

Figure 3. Block Diagram of Radix-4 FFT Processor

The input registers provide the synchronization of input data required for these computations. Each of the values a, b, c, and d is clocked out of the input registers as a stream of quaternary digits and fed to the appropriate adder/subtracter combination. Operating on two-bit digits represents a compromise between the area efficiency of bit-serial computation and the speed of parallel computation. The outputs of the digit-serial adders and subtracters, themselves quaternary streams, are buffered for time-division multiplexing through a single complex multiplier. The size of the high-precision complex multiplier prevented its replication and, therefore, mandated the use of time-division multiplexing. (Notice that a' requires no rotation, so the multiplier need only be multiplexed between the three remaining data items.) The buffered data is synchronized with an on-chip coefficient generator, and the pair of values, data and coefficient, is fed to the multiplier. The multiplier's outputs are captured and reformatted in the output registers. Reformatting is primarily a conversion from the on-chip digit-serial quaternary representation to the parallel inter-chip format. The resulting values are gated to the output data bus and written as eight 32-bit quantities to the inter-stage buffer memory.

Since the multiplier processes three sets of operands per butterfly, its performance determines the chip's overall throughput rate. This presents a significant design challenge. On the one hand, there is the need to minimize the multiplier's size to allow higher precision calculation and better integration, while on the other hand, the multiplier's size determines its throughput. The majority of our effort has been invested in refining the multiplier architecture.

MULTIPLIER ARCHITECTURE

Our multiplier is an adaptation of the distributed arithmetic architecture suggested by [5]. Rather than the naive complex multiplier configuration that requires four real multipliers and two real adders, the distributed arithmetic complex multiplier uses the equivalent of only two multipliers to compute a complex product. The multiplier, y, is converted into the values K and K' as follows:

$$\begin{aligned} K &= \mathrm{Re}(y) + \mathrm{Im}(y), \\ K' &= \mathrm{Re}(y) - \mathrm{Im}(y), \end{aligned}$$

where $\mathrm{Re}(y)$ and $\mathrm{Im}(y)$ are the real and imaginary parts of y respectively. The multiplicand is then encoded to control the accumulate and shift operations on $\pm K$ and $\pm K'$. This approach halves the number of real multipliers without impacting throughput. However, this technique alone does not provide adequate area savings.

Additional area savings is possible, but only with a reduction in multiplier throughput. As described in [4], the multiplication can be performed iteratively in a single, time-multiplexed multiplier row. This iterative multiplication architecture complements our quaternary number system nicely; the two provide a small-area multiplier with acceptable performance.

The iterative multiplier, along with the digit-serial adders and subtracters, enables the radix-4 butterfly to be implemented in a mere 19,000 transistors. By comparison, a butterfly containing a flash multiplier and parallel adders would require approximately 120,000 transistors. This savings allows us to address higher-level system issues.

COEFFICIENT GENERATOR

Traditionally, the FFT coefficients, or "twiddle factors," are stored in ROM and read by the FFT processor as needed. There are two primary disadvantages with this approach. First, it exacerbates the FFT's I/O bottleneck problem, and second, it increases the number of components required per stage. The latter is particularly objectionable when the FFT consists of several stages. On-chip coefficient storage is possible only for modest size or low-precision FFTs—the 32-bit complex coefficients for a 64K-point transform require 4Mb of ROM, so on-chip storage is impractical. Our solution is to compute the coefficients on-chip.

For the decimation in frequency (DIF) FFT algorithm, each of the three coefficient sequences is given by the power series of a complex "seed." The sequences, and therefore the seeds, are determined by position of the processor in the pipeline. Our on-chip coefficient generator consists of a small on-board ROM and a high-precision recursive multiplier. The ROM contains the seed values for each possible processor position. A simplified block diagram of the coefficient generator is shown in figure 4. In the radix-4 FFT, an element from each of three sequences is needed for each butterfly operation. The three sequences are generated in a single, high-precision multiplier that is time-multiplexed.

The disadvantage of on-chip coefficient calculation is the generation and propagation of error. Representation error of the initial seed proves to be the dominant source of error. Recall that the seed values are roots of unity, so any seed has unit length. If the initial seed has a magnitude error of ϵ, then the kth coefficient will have a magnitude of

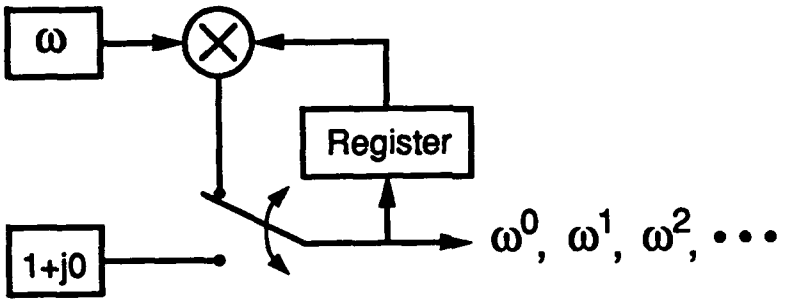

Figure 4. Simplified Block Diagram of Coefficient Generator

$$\begin{aligned} |(\omega)^k| &= (1+\epsilon)^k \\ &= 1 + k\epsilon + O(\epsilon^2) \\ &\approx 1 + k\epsilon. \end{aligned}$$

The initial representation error therefore grows with k. If the initial representation error were restricted to $\pm\frac{1}{2}LSB$, the least significant $\log_2 I$ bits would be corrupted after I iterations. To reduce the impact of propagated error, the coefficient generator performs all calculations to 42 bits. Experimental results agree with this simple model; the actual growth in the calculated error vector are shown in figure 5 for the first 64 terms of one coefficient series.

To minimize the impact of this error term, the heart of the coefficient generator is a 42-bit complex multiplier that is architecturally similar to that in the butterfly, but with three-bit digits. The 42-bit complex results are rounded to 32 bits before being used in the butterfly. The 42-bit precision of the multiplier represents a compromise between multiplier complexity and error magnitude. We note that when more than 2K multiplier iterations are needed, the low-order bits of coefficients in some stages may be corrupted by noise. Since, for the radix-4 processor, $I = N/4$, this can occur only for block lengths of 16K points or longer. Experiments indicate that this is not problematical.

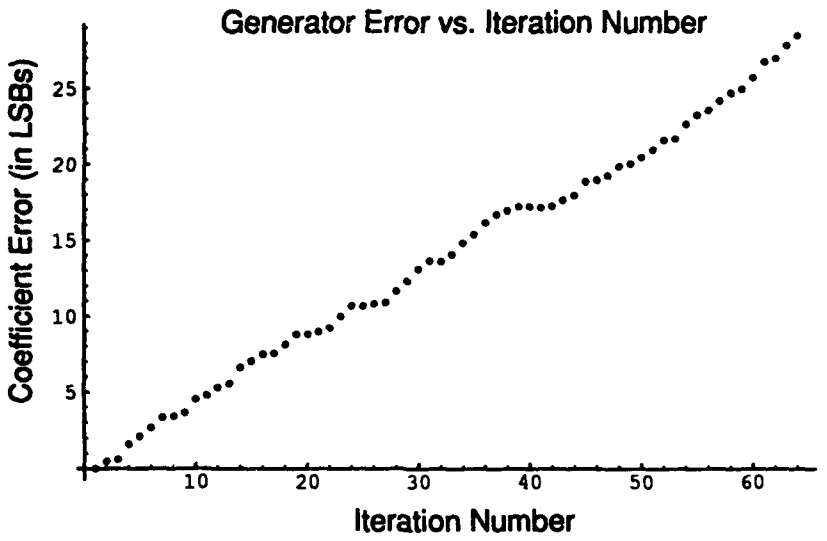

Figure 5. Coefficient Generator Error Growth

MEMORY CONTROLLER

Another integral part of an FFT is the permutation of data between processing stages. In traditional "single-chip" processors, this permutation is accomplished by combining results-in-place FFT algorithms with a double buffering scheme. In a unidirectional pipeline such as ours, data shuffling could be accomplished by double-buffering between each FFT stage, but results-in-place computation is not possible. However, a double-buffering approach would increases the processor's memory requirements; we regard this as unacceptable. An alternative is to shuffle, or permute, the data "on-the-fly," an option available only in the pipeline processor. The shuffling sequence required in the decimation-in-frequency algorithm is amenable to such an approach.

In our approach, data is written into the buffer memory by a sequence of addresses, A. It can then be read in a permuted order, $\rho(A)$, as required by the FFT algorithm. Note that the initial ordering, A, is insignificant so long as $\rho(A)$ can be generated. It is possible, therefore, to operate the interstage buffers efficiently by using read-modify-write memory access cycles. As the first block's data is read in order $\rho(A)$, the next block's data is written in that order. That block is then read by a new permutation of addresses,

$\rho(\rho(A))$, while the third block is simultaneously written in that order. The cycle length of the permutation is given by the smallest integer c for which the equality

$$\rho^c(A) = A$$

holds. The cycle length is an indication of the relative complexity of the address generator. For the simple case of bit reversal, $c = 2$. In the 64K-point radix-4 FFT some stages have a c as large as eight.

Generating this cycle of permutations is the primary function of the memory controller. Additionally, the controller provides the signal timing required to read the 256 bits (four 64-bit quantities) over a 32-bit input data bus. Of course, this includes the generation of *write enable* and *output enable* signals for the memories.

PROGRAMMABILITY

To simplify system design, the radix-4 FFT processor provides modest programmability. Limited programmability allows a single processor design to serve a wider application base, but a high degree of programmability would require off-chip support at power-up. Since our goal was to minimize component count in the final system, we have restricted the device's programmability by constraining all options to be strap or pin configurable. Using this approach, programmability is expensive—one pin per bit—but off-chip support is not required.

SECTION 4

CONCLUSION

We have defined and implemented a new architecture for moderate-rate continuous-throughput FFTs. The heart of this architecture is a monolithic radix-4 FFT processor implemented as a full-custom VLSI circuit. This IC contains all computational and support circuitry—excepting interstage memory—for a single stage of an FFT processor. The resulting IC is fully programmable—a single IC can perform the computations for any stage in a forward or inverse FFT—enabling large FFT processors to be constructed simply. We have implemented both a radix-2 and a radix-4 version of the architecture. Both implementations support 64-bit complex samples, a maximum block length of 64K, and a continuous, real-time throughput of up to 4 million samples per second. Note that these parameters are not limits of the architecture, but the limits of our *implementation* only. The fundamental architecture is applicable to FFTs of arbitrary input and block size. The complete radix-4 processor, which includes all ancillary support functions, requires fewer than 63,000 transistors and fits on a small 6.5×5.2 mm die when implemented in 1.2μm CMOS. Although throughput rates are moderate, butterfly I/O pin requirements have been minimized through the use of digit-serial techniques. This implementation of the processor is packaged in a 132-pin leadless chip carrier. It is suitable for a wide variety of applications that require processing of data at moderate throughput rates with a minimum of available system area and power.

LIST OF REFERENCES

1. Oppenheim, A. V. and R. W. Schafer, 1975, *Digital Signal Processing*, NJ: Prentice-Hall.

2. Perry, B. D., E. A. Palo, R. D. Haggarty, and E. L. Key, 1975, "Trade-off Considerations in the Use of Wideband HF Communications," In *Proceedings IEEE International Conference on Communications*, Volume 2, pages 0930–0940.

3. Hartley, R. and P. Corbett, June 1990, "Digit-Serial Processing Techniques," *IEEE Transactions on Circuits and Systems*, Vol. 37, No. 6, pp. 707–719.

4. Santoro, M. R. and M. A. Horowitz, April 1989, "SPIM: A Pipelined 64 × 64-bit Iterative Multiplier," *IEEE Journal of Solid-State Circuits*, Vol. 24, No. 2, pp. 487–493.

5. MacTaggart, I. R. and M. A. Jack, June 1984, "A Single Chip Radix-2 FFT Butterfly Architecture Using Parallel Data Distributed Arithmetic," *IEEE Journal of Solid-State Circuits*, Vol. 19, No. 3, pp. 368–373.