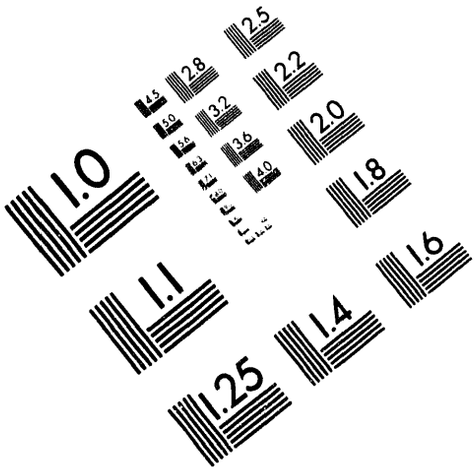


AIIM

Association for Information and Image Management

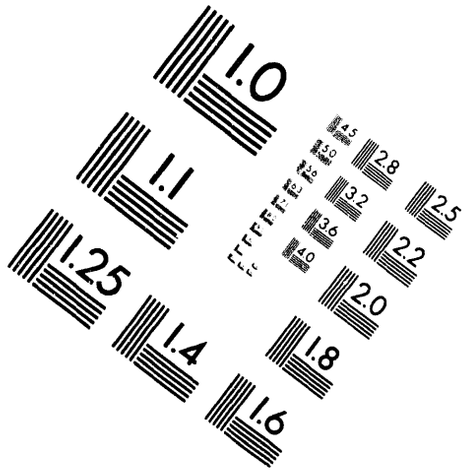
1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910

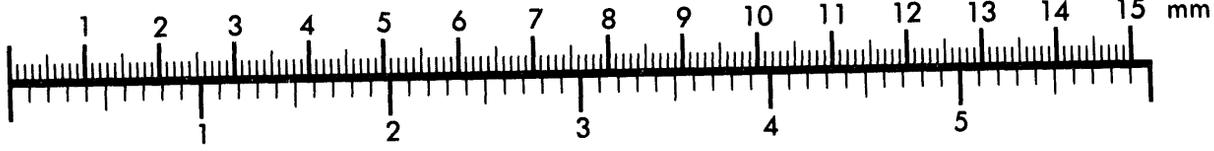
301/587-8202

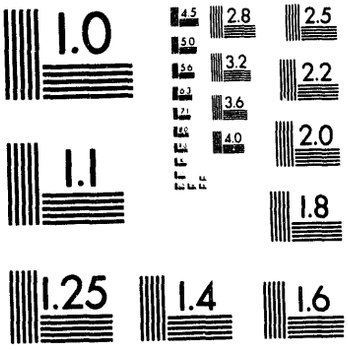
Centimeter

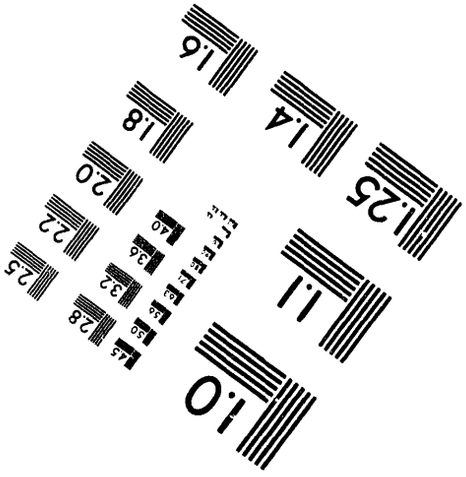
Inches

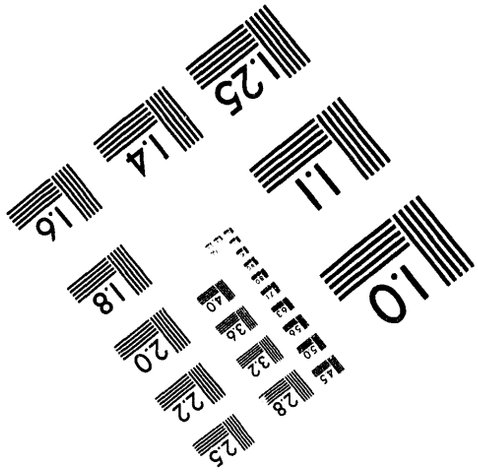
MANUFACTURED TO AIIM STANDARDS
BY APPLIED IMAGE, INC.











1 of 1

Conf-9406205-4

PARALLEL FAST FOURIER TRANSFORMS FOR NON POWER OF TWO DATA

Bart D. Semeraro
Mathematical Sciences Section
Engineering Physics and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831

"The submitted manuscript has been authored by a contractor of the U.S. Government under contract No. DE-AC05-84OR21400. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes."

* This research was sponsored by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with the Martin Marietta Energy Systems, Inc.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Parallel Fast Fourier Transforms for Non Power of Two Data

B. D. Semeraro
Oak Ridge National Laboratory
Mathematical Sciences Section
Oak Ridge, TN 37831-6367

Abstract

This report deals with parallel algorithms for computing discrete Fourier transforms of real sequences of length N not equal to a power of two. The method described is an extension of existing power of two transforms to sequences with N a product of small primes. In particular, this implementation requires $N = 2^p 3^q 5^r$. The communication required is the same as for a transform of length $N = 2^p$.

The algorithm presented is intended for use in the solution of partial differential equations, or in any situation in which a large number of forward and backward transforms must be performed and in which the Fourier Coefficients need not be ordered. This implementation is a one dimensional FFT but the techniques are applicable to multidimensional transforms as well. The algorithm has been implemented on a 128 node Intel Ipsc/860.

1 Introduction

The algorithm described in this report is an extension of the parallel algorithm given by Swarztrauber [2] and Walker [3] for computing power of two Fourier transforms on a hypercube. The algorithm for obtaining the FFT of a real sequence is based on the method by Cooley, Lewis and Welch [1] in which a post processing step is used to construct the Fourier coefficients of the real sequence from the coefficients of a complex half length sequence. This algorithm is unordered in that the Fourier coefficients are not in a natural order but in a hypercube or processor permuted order.

Fast algorithms for evaluating the Fourier transform of a sequence of N elements rely on N being factorable into a product of small prime numbers. In the power of two transform $N = 2^k$ and there are k stages to the algorithm. When data is distributed over several processors each stage may require some communication. In the case of a hypercube architecture, Swarztrauber and Walker demonstrated that power of two stages require only nearest neighbor communication. In the remainder of this paper we investigate communications induced by non power of two stages in the sequence and show a simple strategy for dealing with the associated difficulties. We begin with a discussion of issues related to the evaluation of Fourier transforms of real data in section (2). Section (3) outlines the FFT algorithm and introduces the matrix notation used in describing the communication requirements in the various stages. Section (4) describes the parallel algorithm and the extension for non power of two data.

2 Real Fourier Transform

The discrete Fourier transform can take several forms depending on the convention adopted for the sign of the exponent and when the normalization takes place. In this work the forward and backward Fourier transforms are,

$$F_j = \sum_{k=0}^{N-1} f_k e^{-ijk\frac{2\pi}{N}} \quad (j = 0 \ldots N-1), \qquad (1)$$

and,

$$f_k = \frac{1}{N} \sum_{j=0}^{N-1} F_j e^{ijk\frac{2\pi}{N}} \quad (k = 0 \ldots N-1), \qquad (2)$$

respectively. Here the f_k are assumed to be real and as a result the Fourier coefficients, F_j, will be conjugate symmetric, ($F_j = F^*_{n-j}$). We concern ourselves with the evaluation of equation (1).

The Cooley, Lewis and Welch algorithm for computing the Fourier transform of an N element real sequence begins with the transform of a half length complex sequence constructed from the real input data. Each element of the complex vector h is taken to be $h_j = f_{2j} + if_{2j+1}$, where $j = 0 \ldots M-1$, and $M = N/2$. The kth element of the Fourier transform

of this vector is,

$$\begin{aligned} H_k &= \sum_{j=0}^{M-1} h_j e^{-ijk\frac{2\pi}{M}} \\ &= \sum_{j=0}^{M-1} f_{2j} e^{-ijk\frac{2\pi}{M}} + i \sum_{j=0}^{M-1} f_{2j+1} e^{-ijk\frac{2\pi}{M}} \\ &= F_{e_k} + iF_{o_k}. \end{aligned}$$

where F_e and F_o are the Fourier transforms of the even and odd components of the original N element real sequence and $k = 0 \ldots M-1$. The kth element of the transform of the original sequence is,

$$\begin{aligned} F_k &= \sum_{j=0}^{N-1} f_j e^{-ijk\frac{2\pi}{N}} \\ &= \sum_{j=0}^{M-1} f_{2j} e^{-ijk\frac{2\pi}{M}} + e^{-ik\frac{2\pi}{N}} \sum_{j=0}^{M-1} f_{2j+1} e^{-ijk\frac{2\pi}{M}} \\ &= F_{e_k} + e^{-ik\frac{2\pi}{N}} F_{o_k}. \end{aligned}$$

where F_{e_k} and F_{o_k} are again the transforms of the even and odd components of the original sequence and $k = 0 \ldots N-1$. These components can be obtained from the components of the half length transform by taking into consideration that the transforms of the even and odd components of the original sequence are conjugate symmetric. Using this information it follows that $F_{e_k} = (H_k + H^*_{M-k})/2$ and $F_{o_k} = -i(H_k - H^*_{M-k})/2$. Substitution into the expression for the transform of the original sequence gives,

$$\begin{aligned} F_k &= \Big[(H_k + H^*_{M-k}) - \\ &\quad ie^{-ik\frac{2\pi}{N}} (H_k - H^*_{M-k})\Big]/2. \end{aligned} \tag{3}$$

This expression only needs to be evaluated for $k = 0, \ldots, M$ since the Fourier coefficients of the original sequence are conjugate symmetric. The positive frequency spectrum of the original sequence can be overwritten onto the original vector.

The algorithm for computing the Fourier transform of an n element real sequence f can be summarized as follows:

- Calculate the Fourier transform of an $M = N/2$ element complex sequence h with $h_j = f_{2j} + if_{2j+1}$, $j = 0, \ldots, M-1$. This step requires that the number of elements in the original sequence f is even. Notice that no reordering of the original sequence is necessary to form h.
- Calculate the Fourier coefficients of the original sequence by combining the coefficients of the half length complex sequence via equation (3).
- The original sequence is overwritten by the independent Fourier coefficients F_j. This is possible because only the positive half of the frequency spectrum needs to be stored and because F_0 and F_M are real. This allows F_M to be stored in the imaginary part of F_0.

Notice that this algorithm for computing the Fourier transform of a real sequence depends on the efficient calculation of the Fourier transform of a complex sequence. The next section describes fast methods for evaluating the Fourier transform of such a complex sequence.

3 Fast Fourier Transform

The calculation of a Fourier transform by direct application of equation (1) is equivalent to a dense matrix vector multiplication $F = Wf$, in which $w_{jk} = \omega^{jk}$ and $\omega = e^{-i\frac{2\pi}{N}}$. Evaluation of the Fourier transform by this method requires $0(N^2)$ operations for a sequence of length N. Fast methods for evaluation of the Fourier transform reduce the number of operations by taking advantage of special properties of the Fourier matrix W.

Consider the permutation of the linear system of equations $F = Wf$ in which the even and odd equations are grouped together. $F' = PF = (F_e, F_o)^T$, where P is the permutation matrix which performs the odd-even sorting of the equations and $F_e = F_{2k}$, $F_o = F_{2k+1}, k = 0, \ldots, M-1$. The rows of the Fourier matrix are also permuted by P giving $W' = PW$. The even and odd components of equation (1) can be written:

$$\begin{aligned} Fe_k &= \sum_{j=0}^{M-1} e^{-ijk\frac{2\pi}{M}} (f_j + f_{j+M}), \\ Fo_k &= \sum_{j=0}^{M-1} e^{-ijk\frac{2\pi}{M}} e^{-j\frac{\pi}{M}} (f_j - f_{j+M}), \end{aligned}$$

where $M = N/2$. The matrix form of this expression is:

$$\begin{pmatrix} Fe \\ Fo \end{pmatrix} = \begin{pmatrix} W_M & 0 \\ 0 & W_M \end{pmatrix} \begin{pmatrix} I & I \\ D_M & -D_M \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}.$$

The elements of the matrix W_M are $w_{jk} = \omega(M)^{jk}$ and $\omega(M) = e^{\frac{-i2\pi}{M}}$. This matrix is the M element

Fourier matrix. Multiplication of a vector by W_M is equivalent to performing an M element Fourier transform. The matrix D_M is diagonal with elements $d_j = e^{\frac{-ij\pi}{M}}$

The above expression represents the factorization of the matrix W' into a block diagonal matrix,

$$\begin{pmatrix} W_M & 0 \\ 0 & W_M \end{pmatrix},$$

and a power of two butterfly operator,

$$\begin{pmatrix} I & I \\ D_M & -D_M \end{pmatrix} = \begin{pmatrix} I & \\ & D_m \end{pmatrix} \begin{pmatrix} I & I \\ I & -I \end{pmatrix}.$$

Recursive application of this factorization process to the resulting linear system gives the power of two FFT. If the length of the original sequence is a power of two the number of operations required to evaluate the transform is reduced from $O(N^2)$ to $O(N \log_2 N)$.

Similar factorizations exist for powers other than two. These factorizations allow the treatment of sequences of length N for which N is not a power of two. Of particular interest are situations in which N is a product of small prime numbers. The power of three butterfly operator is,

$$\begin{pmatrix} I & & \\ & D_M & \\ & & D_M^2 \end{pmatrix} \begin{pmatrix} I & I & I \\ I & a^2 I & -aI \\ I & -aI & a^2 I \end{pmatrix}$$

where, $M = N/3$, $a = e^{-\frac{\pi}{3}}$, and the jth element of the diagonal matrix D_M is $e^{-ij\frac{\pi}{M}}$. The power of five butterfly operator is the product of the diagonal matrix,

$$\begin{pmatrix} I & & & & \\ & D_M & & & \\ & & D_M^2 & & \\ & & & D_M^3 & \\ & & & & D_M^4 \end{pmatrix}$$

and the butterfly matrix,

$$\begin{pmatrix} I & I & I & I & I \\ I & b^2 I & b^4 I & -bI & -b^3 I \\ I & b^4 I & -b^3 I & b^2 I & -bI \\ I & -bI & b^2 I & -b^3 I & b^4 I \\ I & -b^3 I & -bI & b^4 I & b^2 I \end{pmatrix}$$

where, $M = N/5$, $b = e^{-i\frac{\pi}{5}}$, and D_M is defined as above. These butterfly operators are the basic operations in the fast Fourier transform of a sequence of length $N = 2^p 3^q 5^r$. In the next section we explore the types of communication induced by the butterfly operators.

4 Parallel Transforms

In this section we examine the communications necessary to perform a fast Fourier transform of an N element complex vector f in parallel. It was shown in the previous section that a fast Fourier transform can be performed as a sequence of butterfly operations. In particular, if the length of a complex sequence is a power of two, $N = 2^p$, then the transform can be performed with p power of two butterfly operations. If N has factors other than two, then the fast transform will incorporate butterfly operations corresponding to those factors. These butterfly operations can occur in any order. For example, if $N = 18$, two power of three butterflies can be performed followed by a power of two. Or, a power of two can be performed between two power of three operations. Both sequences of events will yield a Fourier transform of the original data. However, the Fourier coefficients will be ordered differently in each case.

The primary aspect of the parallel transform is the communication required to perform the sequence of butterfly operations. For the hypercube the interprocessor communication required to perform a sequence of power of two butterfly operations is relatively simple. Swarztrauber [2] showed that $d+1$ parallel transmissions are required to perform a sequence of power of two butterflies on a d dimensional cube. In addition, each transmission is a nearest neighbor communication. The situation is more complex for a non-power of two butterfly operation.

We illustrate the communications involved by considering the example problem of transforming an N element vector on four processors. The initial data is naturally ordered and evenly distributed among the processors. Assume that $N = 2^p \cdot 3^q \cdot 5^r$ and that $p \geq 3$. The FFT of this data will then require $p \cdot q \cdot r$ stages consisting of p power of two butterfly operations, q power of three butterfly operations, and r power of five butterfly operations.

Applying the techniques of section 3, the matrix form of the FFT can be written

$$F = W B_2 B_1 f, \tag{4}$$

where B_1 and B_2 are the first stage and second stage butterfly operators respectively, F is the vector of Fourier coefficients in index reversed order, and W represents the product of the remaining butterfly operations.

The distribution of the elements of the input vector f across the four processors can be viewed as a

partitioning of the vector,

$$f = \begin{array}{c} \left.\begin{array}{c} f_0 \\ f_1 \end{array}\right\} \leftarrow P_0 \\ \left.\begin{array}{c} f_2 \\ f_3 \end{array}\right\} \leftarrow P_1 \\ \left.\begin{array}{c} f_4 \\ f_5 \end{array}\right\} \leftarrow P_2 \\ \left.\begin{array}{c} f_6 \\ f_7 \end{array}\right\} \leftarrow P_3 \end{array}.$$

where each f_j is a subvector of length $N/8$. The first two subvectors are located on processor P_0. The second two are located on processor P_1 and so on as indicated above.

If the first stage of the calculation is a power of two butterfly; then the conformal partitioning of the matrix B_1 is given by

$$\left(\begin{array}{cc|cc|cc} I & & I & & & \\ & I & & I & & \\ \hline & & I & & I & \\ & & & I & & I \\ \hline D_0 & & -D_0 & & & \\ & D_1 & & -D_1 & & \\ \hline & & D_2 & & -D_2 & \\ & & & D_3 & & -D_3 \end{array}\right).$$

Notice that the multiplication of f by the first block row of B_1 requires the addition of the subvectors $(f_0, f_1)^T$ to the subvectors $(f_4, f_5)^T$. These elements are on different processors and so some communication is required before the sums $f_0 + f_4$ and $f_1 + f_5$ can be evaluated. A reordering of the elements of f is required which will allow the computations involved in the multiplication of f by B_1 to be done locally. Since the data involved resides on only two processors, the required communication can be carried out as an exchange of data between the processors. For example, processor P_0 can exchange f_1 with f_4 on processor P_2. Swarztrauber [2] used a class of orderings called index-digit permutations to derive the reorderings required to do power of two butterfly operations.

An important aspect of Swarztrauber's algorithm can be illustrated by viewing the index-digit permutations as permutation matrices operating on equation (4). The feature we wish to illustrate and exploit is that for certain situations only a subset of the total number of butterfly stages require communication. In other words, a point in the algorithm is eventually reach at which the remaining computation is simply a local FFT of length N/p, where p is the number of processors. We illustrate this by completing the description of the Swarztrauber algorithm in the context of our example.

A permutation matrix P is obtained by exchanging rows of the identity matrix. Multiplication by P will result in the exchange of the corresponding elements in a vector. Premultiplication of a matrix by P will exchange rows and post multiplication by P will exchange columns of the original matrix. In addition $P^2 = I$. The permutation matrix corresponding to the reordering required to perform the first stage butterfly computation described above is

$$P_1 = \left(\begin{array}{c|c|c|c} I & & I & \\ \hline & I & & I \\ \hline I & & & \\ & & I & \\ \hline & I & & \\ & & & I \end{array}\right).$$

Multiplication of f by this matrix gives the following reordering

$$f' = \begin{array}{c} \left.\begin{array}{c} f_0 \\ f_4 \end{array}\right\} \leftarrow P_0 \\ \left.\begin{array}{c} f_2 \\ f_6 \end{array}\right\} \leftarrow P_1 \\ \left.\begin{array}{c} f_1 \\ f_5 \end{array}\right\} \leftarrow P_2 \\ \left.\begin{array}{c} f_3 \\ f_7 \end{array}\right\} \leftarrow P_3 \end{array}.$$

Pre and Post multiplication of B_1 by P_1 gives,

$$B_1' = \left(\begin{array}{cc|cc|cc|cc} I & I & & & & & & \\ D_0 & -D_0 & & & & & & \\ \hline & & I & I & & & & \\ & & D_2 & -D_2 & & & & \\ \hline & & & & I & I & & \\ & & & & D_1 & -D_1 & & \\ \hline & & & & & & I & I \\ & & & & & & D_3 & -D_3 \end{array}\right).$$

Referring to equation (4), the matrix form of the parallel evaluation of the first butterfly operation is,

$$\begin{aligned} F &= \hat{W} B_2 B_1 f \\ &= \hat{W} P_1^2 B_2 P_1^2 B_1 P_1^2 f \\ &= \hat{W} P_1 B_2' B_1' f' \\ &= \hat{W} P_1 B_2' f_a \end{aligned}$$

where f' and B_1' are defined above. Notice that B_1' is block diagonal and that the structure of each block is that of a power of two butterfly operation. The communication required to perform the butterfly operation can be thought of as being induced by a permutation matrix which transforms B_1 to block diagonal

form. The vector $f_a = B_1' f'$ is the result of the first butterfly operation reordered by application of P_1.

The second stage power of two butterfly matrix B_2 has the form

$$\left(\begin{array}{cc|cc|cc|cc} I & & I & & & & & \\ & I & & I & & & & \\ \hline D_0 & & -D_0 & & & & & \\ & D_1 & & -D_1 & & & & \\ \hline & & & & I & & I & \\ & & & & & I & & I \\ \hline & & & & D_0 & & -D_0 & \\ & & & & & D_1 & & -D_1 \end{array}\right).$$

The matrix B_2' has the same block structure as B_2 but differs in the location of some of the elements. B_2' is,

$$\left(\begin{array}{cc|cc|cc|cc} I & & I & & & & & \\ & I & & I & & & & \\ \hline D_0 & & -D_0 & & & & & \\ & D_0 & & -D_0 & & & & \\ \hline & & & & I & & I & \\ & & & & & I & & I \\ \hline & & & & D_1 & & -D_1 & \\ & & & & & D_1 & & -D_1 \end{array}\right).$$

This butterfly matrix is reduced to block diagonal form by application of a second permutation matrix P_2. This operation will induce communication in the reordering of f_a. The matrix form of this procedure is written as

$$\begin{aligned} F &= \hat{W} P_1 B_2' f_a \\ &= \hat{W} P_1 P_2^2 B_2' P_2^2 f_a \\ &= \hat{W} P_1 P_2 B_2'' f_a' \\ &= \hat{W} P_1 P_2 f_b, \end{aligned}$$

where $f_b = P_2 P_1 B_2 B_1 f$, is the result of the first two butterfly operations reordered by the application of P_1 followed by P_2.

The matrix $\hat{W}$ is

$$\left(\begin{array}{cc|cc|cc|cc} W_{11} & W_{12} & & & & & & \\ W_{21} & W_{22} & & & & & & \\ \hline & & W_{11} & W_{12} & & & & \\ & & W_{21} & W_{22} & & & & \\ \hline & & & & W_{11} & W_{12} & & \\ & & & & W_{21} & W_{22} & & \\ \hline & & & & & & W_{11} & W_{12} \\ & & & & & & W_{21} & W_{22} \end{array}\right).$$

This matrix is block diagonal with block size equal to the local number of elements of f on a processor. Multiplication by $\hat{W}$ requires no interprocessor communication. In addition this matrix retains this structure under the transformation $P_1 P_2 P_1 W P_1 P_2 P_1$. This allows us to write the matrix form of the Swarztrauber FFT as,

$$\begin{aligned} P_1 \cdot P_2 \cdot P_1 F &= \hat{W} P_1 B_2'' P_2 B_1' P_1 f \\ F' &= \hat{W} P_1 B_2'' P_2 B_1' P_1 f \end{aligned}$$

where, F' is the vector of Fourier coefficients in processor permuted order.

In this example the FFT has three stages. The first stage is an interprocessor communication induced by the multiplication of f by P_1 followed by a parallel butterfly computation. The second stage is an interprocessor communication induced by the multiplication of the result of the first stage by P_2 followed by the parallel second stage butterfly operation. The final stage is an interprocessor communication induced by the multiplication of the second stage result by P_1. This final reordering is followed by a set of local FFTs of length $N/4$. The length of the local FFTs is not restricted. It is in this final local stage that the non power of two butterfly operations are performed.

5 Summary

The algorithm for computing the FFT of an N element sequence in parallel on a hypercube architecture is extended to handle non power of two factors by restricting those factors to the local part of the FFT. The parallel stages still consist of power of two butterfly operations. On a d dimensional cube N must be divisible by 2^{d+1}. An important aspect of this algorithm is that the interprocessor communication is not effected by non power of two factors.

References

[1] P.A.W. Lewis J.W. Cooley and P.D. Welch. The fast fourier transform algorithm: programming considerations in the calculation of sine, cosine and laplace transforms. *Journal of Sound and Vibrations*, 12:315–337, 1970.

[2] Paul N. Swarztrauber. Multiprocessor ffts. *Parallel Computing*, 5:197–210, 1987.

[3] D. W. Walker. Portable programming within a message-passing model: The fft as an example. In *Proc. Third conference on Hypercube Concurrent Computers and Applications, G.C. Fox, ed.*, pages 1438–1450. ACM Press, 1988.

DATE
FILMED
10/14/94

END