Submitted to Int. Conf. Paris 88.

# COMPLEXITY ANALYSIS OF GENERALIZED FAST FOURIER TRANSFORMS IN A MULTIPROCESSOR ENVIRONMENT

TATYANA D. ROZINER, Boston University, Boston, USA
MARK G. KARPOVSKY, Boston University, Boston, USA
LAZAR A. TRACHTENBERG, Drexel University, Philadelphia, USA

MAILING ADDRESS:

Prof. Tatyana ROZINER
BOSTON UNIVERSITY
COLLEGE OF ENGINEERING, ECS DPT
44 CUMMINGTON STREET
BOSTON, MA 02215, USA

## ABSTRACT

The paper presents a method for an optimal implementation of General Discrete Fourier Transform (GDFT) algorithms over finite groups (Abelian and non-Abelian) in multiprocessor environment. Tradeoffs between hardware complexity/speed and computation time are investigated for different multiprocessor implementations with local non-shared memories (unibus, complete communication network). Formulas are presented for the number of arithmetic operations and for the number of interprocessor data transfers.



This work was supported by NSF under Grant No. ECS-8512748, Subcontract No.350021A.

## I. INTRODUCTION.

Fast Fourier Transforms (FFT) are well known to play an important role in many application fields as spectral analysis, digital filtering, image processing, video transmission, etc. The increasing requirements to speed in many real-time applications stimulated in the recent years the development of a number of new very fast FT algorithms as well as numerous investigations on comparative complexity of different FFT versions [2], [4],[12],[14]. A systematic approach to the problems of design of FFT and complexity evaluation was developed by Beth [2] who generalized the results formulated in [3,4,5] having considered the classical FDFT as a special case of GDFT (General Discrete Fourier Transform) based on the theory of representations of finite groups. .

As noted in [ 4 ; 13 ] and substantiated in [2], the decrease in the number of multiplications in a very fast FT algorithm involves inevitably an increase in the number of additions and/or preprocessing operations. From the viewpoint of further advance as to speed increase of FFT, and with high progress in VLSI technology, the multiprocessor highly parallel systems become an attractive alternative compared to uniprocessor architectures.

In a multiprocessor environment, a new factor arises that may strongly influence the efficiency of FFT algorithm performance. The structure of the algorithm involves numerous interprocessor data transfers which can, in case of inappropriate or too slow processor communication network, become a reason for a degradation of performance. Algorithm-independent upper bounds on complexity including the evaluation of the number of interprocessor transfers were determined in [14] for the classical FFT algorithm performance in terms of multiprocessor communication network design. From the viewpoint of group theory approach, the results of [14 ]apply to the case of the cyclic group structure only

with multiple processors performing the group algorithm.

The subject of the present paper is the generalization of the results mentioned above for the case of finite decomposable (possibly non-Abelian) group structure introduced on the input data set, for the problems of GDFT spectrum calculation by a multiprocessor SIMD system with non-shared memory. The investigation of tradeoffs between the number of operations (including data transfers) needed for GDFT over arbitrary finite groups ) and hardware complexity (multiple processors and different communication networks) is performed. It is shown for sample evaluations with different group structures that the use of non-Abelian groups (e.g. quaternions) may result in many cases in optimal (fastest) performance of GDFT.

## II. EVALUATION OF THE NUMBER OF OPERATIONS NEEDED FOR FFT OVER GROUP STRUCTURE $G = G_0 \times G_1 \times \ldots \times G_{m-1}$ USING MULTIPLE PROCESSORS.

Let $G$ be an arbitrary finite group of order $|G| = N$ ; let $V$ be the vector space of dimension $d$ over the field $C$ of complex numbers, and let $GL(V)$ be the group of all nonsingular $d \times d$ matrices with elements in $C$ . A representation of $G$ is a homomorphism $R: G \to GL(V)$, that is, for $x, y \in G$, $xy \in G$, $R(xy) = R(x) \cdot R(y)$. A representation $R$ is irreducible if there are no nontrivial subspaces of $V$ which are mapped to themselves by all matrices $R(x)$, $x \in G$. Every representation $R_w$ is equivalent to some unitary representation $R_u$ (i.e. to a representation with unitary matrix $R_u(x)$ ), that is, there exists a $Q \in GL(V)$ such that $R_u(x) = Q^{-1} R_w(x) Q$ for all $x \in G$ . Irreducible unitary representations for $G$ are orthogonal. This allows to define the direct and inverse General Discrete Fourier Transform (GDFT) over $G$ as follows.

If $f$ is a function $f : G \to C$, then

$$\hat{f}(w) = S_f(w) = \frac{d_w}{N} \sum_{x \in G} f(x) R_w(x^{-1})$$

$$f(x) = \sum_{R_w \in R(G)} Tr\left( S_f(w) \cdot R_w(x) \right)$$

where $x^{-1}$ is the inverse of $x$ in $G$.

Let $G, |G| = N$, be a direct product of groups $G_j$, $0 \le j \le m-1$, where $|G_j| = n_j$; $N = n_0 n_1 \cdot \ldots \cdot n_{m-1}$. Denote the elements of $G$ as $X_k \in G$, $0 \le k \le N-1$, and the elements of group $G_j$ as $x_j \in G_j$, $0 \le j \le m-1$. Let $R(G)$ be the set of all irreducible unitary representations of $G$ with elements $R_w \in R(G)$, and let $R(G_j)$ be the set of all irreducible unitary representations of $G_j$ with elements $R_{w_j} \in R(G_j)$, $0 \le j \le m-1$ [ ].

Following [8], note that:

1) If $X \in G$, then $X = (x_0, x_1, \ldots, x_{m-1})$, $x_j \in G_j$.

2) If $R_w \in R(G)$, then $R_w(X) = R_w(x_0, x_1, \ldots, x_{m-1}) = \bigotimes_{j=0}^{m-1} R_{w_j}(x_j)$, where $R_{w_j} \in R(G_j)$; symbol $\otimes$ denotes the Kronecker product of matrices.

Thus, for every $R_w \in R(G)$, we may denote $w = (w_0, w_1, \ldots, w_{m-1})$.

The direct Fourier transform over group $G$ may be defined as follows [8]:

$$\hat{f}(w) = \hat{f}(w_0, w_1, \ldots, w_{m-1}) = \frac{d_w}{N} \sum_{x_0} \sum_{x_1} \ldots \sum_{x_{m-1}} f(x_0, x_1, \ldots, x_{m-1}) \bigotimes_{j=0}^{m-1} R_{w_j}(x_j^{-1}) =$$

$$= \frac{d_w}{N} \sum_{x_{m-1}} \Big( \ldots \Big( \sum_{x_1} \Big( \sum_{x_0} \Big( f(x_0, x_1, \ldots, x_{m-1}) R_{w_0}(x_0^{-1}) \Big) \otimes R_{w_1}(x_1^{-1}) \Big) \otimes \ldots \Big) \otimes R_{w_{m-1}}(x_{m-1}^{-1}) \Big)$$

The calculation of spectrum $\hat{f}(w_0, w_1, \ldots, w_{m-1})$ is performed in $m$ steps (Step $j$ over group $G_j$, $0 \le j \le m-1$). Let

$$f_0(x_0, x_1, \ldots, x_{m-1}) = f(x_0, x_1, \ldots, x_{m-1})$$

Step 0:

$$f_1(w_0, x_1, \ldots, x_{m-1}) = \sum_{x_0} f_0(x_0, x_1, \ldots, x_{m-1}) R_{w_0}(x_0^{-1});$$

. . . . . .

Step $j$ : $f_{j+1}(w_0, w_1, \ldots w_j, x_{j+1}, \ldots, x_{m-1}) = \sum_{x_j} f_j(w_0, \ldots, w_{j-1}, x_j, \ldots, x_{m-1}) \otimes R_{w_j}(x_j^{-1})$

. . . . . .

Step $(m-1)$: $f_m(w_0, w_1, \ldots, w_{m-1}) = \frac{N}{d_w} \hat{f}(w_0, w_1, \ldots, w_{m-1}) =$

$$= \sum_{x_{m-1}} f_{m-1}(w_0, w_1, \ldots, w_{m-2}, x_{m-1}) \otimes R_{w_{m-1}}(x_{m-1}^{-1}) .$$

Let $X_i$ be the $i$-th element of $G$ : $X_i \in G$, $0 \le i \le N-1$ .It will be convenient to list the elements of $G$ in a certain order. If $x_j^i$ designates the $i$-th element of group $G_j$, $0 \le i \le n_j - 1$, $0 \le j \le m-1$, let the elements $X_0, X_1, \ldots, X_{N-1}$ of $G$ be ordered as follows.

Element $X_p = (x_0^{i_0}, x_1^{i_1}, \ldots, x_{m-1}^{i_{m-1}})$ precedes the element $X_q = (x_0^{j_0}, x_1^{j_1}, \ldots, x_{m-1}^{j_{m-1}})$ if $i_0 = j_0$, $i_1 = j_1, \ldots, i_{k-1} = j_{k-1}$, and $i_k < j_k$, for some $k$, $0 \le k \le m-1$.

In the following evaluations of the number of operations we do not take into account the operations needed for spectrum reshuffling and normalization.

Let $p$ denote the number of processors operating in parallel during the calculation of the generalized spectrum. Assume that each processor contains local memory, but there is no global shared memory. (See [14] for the examples of existing computers of this type). In each one of steps of spectrum calculation, the $N$ elements of intermediate spectrum are partitioned into disjoint subsets . In step $j$, $N/n_j$ subsets (each containing $n_j$ elements) are to be used as input data for butterfly algorithm of the constituent group $G_j$. In order to avoid idle processors in any step, $N/pn_j$ must be integer for $0 \le j \le m-1$.

Denote the number of operations needed to perform butterfly algorithm of $G_j$ on a single input data set (with one processor) as $L(G_j)$. (In general case, the algorithm is characterized by a pair of numbers - the number of additions/subtractions and the number of multiplications. We shall assume that $L(G_j)$ is the equivalent number of additions).

To perform the spectrum calculation over $G = G_0 \times G_1 \times \ldots \times G_{m-1}$ by a single processor, $L(G)$ operations are needed:

$$L(G) = \sum_{j=0}^{m-1} L(G_j) \prod_{\substack{i=0 \\ i \neq j}}^{m-1} n_i = N \sum_{j=0}^{m-1} \frac{L(G_j)}{n_j} \qquad (1)$$

To generalize (1) for the case of $p$ parallel processors, denote $L^{(p)}(G)$ the number of operations needed for spectrum calculation over $G$ with $p$ processors. Evidently,

$$L^{(p)}(G) = \frac{L(G)}{p} = \frac{N}{p} \sum_{j=0}^{m-1} \frac{L(G_j)}{n_j}$$

However, for the number of processors $p > 1$, the data transfers among the processors are needed (at least in one of the algorithm steps). Therefore the total time $T^{(p)}(G)$ needed for spectrum calculation over $G$ with $p$ processors is:

$$T^{(p)}(G) = L^{(p)}(G) \cdot t_a + M^{(p)}(G) \cdot t_c ,$$

where $t_a$ is the time needed for one addition/subtraction, $t_c$ is the time needed for one data transfer (communication time), and $M^{(p)}(G)$ is the total number of transfers needed for spectrum calculation over $G$ with $p$ processors. The value of $M^{(p)}(G)$ is not only a function of $p$ and of the group structure for a fixed $|G| = N$ ; it depends also on the type of the processor communication network.

To minimize $M^{(p)}(G)$, the communication network is needed where each processor can communicate directly with any other processor (if the

corresponding communication is required by the algorithm), and is able to receive or to transfer a unit of data during communication time . This type of connection network is of high cost for a large number of processors (though not always the complete network is needed). It seems more reasonable to assume another extreme case when all processors communicate via single bus ("unibus" connection). In the last case, during transfer time $t_c$ only one data unit can be transferred from one processor to another. This type of connection puts some restrictions on $t_c$ - the unibus must be fast enough, in order not to increase significantly the total calculation time.

Assume the following:

a) For any $j$ , $0 \le j \le m-1$, the $N/pn_j$ value is an integer (no idle processors in any step). The job load is distributed equally among $p$ processors.

b) The input data are entered into local memories of $p$ processors as follows: first $N/p$ elements of the input array are given to processor 1, next $N/p$ elements - to processor 2,etc.

c) For any group $G_j$ (that is, in each step of the algorithm) any set of input data containing $n_j$ elements is delivered to a single processor. It can be shown that the violation of the rule "single group set - single processor" results in a drastic increase in the number of interprocessor communications needed, and in many cases the number of operations is also growing.

Define two parameters: $p_1 = N/p$ - the number of elements of $G$ per 1 processor; $n_0 n_1 \ldots n_j$ - block size for group $G_j$ , $0 \le j \le m-1$.

Since before processing (before Step 0) $N/p = p_1$ successive elements of $G$ were delivered to the 1st processor,next $p_1$ successive elements - to the 2nd processor,etc.,the block sizes in successive steps of spectrum calculation allow to trace the distribution of intermediate

spectrum elements in block lines among different processors and to determine whether interprocessor data transfers are needed in a certain step. The following rule is valid: if block size $n_0 n_1 \cdot \ldots \cdot n_j \le p_1 = N/p$, no data transfers are needed in Step $j$ . If block size $n_0 n_1 \cdot \ldots \cdot n_j > p_1$ , transfers among $p$ processors must be performed. (Note that the term "transfers needed in Step $j$ " means transfers needed before processing of Step $j$ ).

It can be easily seen that transfers are never needed in Step 0 (block size $n_0$ ) if our previous assumption is valid that $N/pn_j$ is an integer for any $j$, $0 \le j \le m-1$ . In particular, $N/pn_0 \ge 1$ is an integer which means $n_0 < p_1$ (no transfers needed). For Step 0, there are $N/n_0$ "degenerate" blocks each containing only one line.

For the case when transfers are needed( $n_0 n_1 \cdot \ldots \cdot n_j > p_1 = N/p$ ), define parameter $\mathcal{K}_j$ , $1 \le j \le m-1$, as $\mathcal{K}_j = n_0 n_1 \cdot \ldots \cdot n_j / p_1$ ; $\mathcal{K}_j$ is the number of processors per block of group $G_j$.

Formula (1) may be generalized now for the case of $p$ processors. Total time $T^{(p)}(G)$ needed for spectrum calculation over $G = G_0 \times G_1 \times \ldots \times G_{m-1}$ is:

$$T^{(p)}(G) = L^{(p)}(G) t_a + M^{(p)}(G) \cdot t_c = \frac{N}{p}\left( t_a \sum_{j=0}^{m-1} \frac{L(G_j)}{n_j} + K_1 t_c \sum_{j=1}^{m-1} \varphi_j(p, n_0, \ldots, n_j)\left(1 - \frac{1}{\min(n_j, \mathcal{K}_j)}\right)\right) \quad (2)$$

where $K_1 = 2$, for complete communication network, and $K_1 = p$ , for the unibus connection of processors; $\varphi_j(p, n_0, n_1, \ldots, n_j) = 0$ for $n_0 n_1 \cdot \ldots \cdot n_j \le p_1 = N/p$, and $\varphi_j(p, n_0, n_1, \ldots, n_j) = 1$ otherwise.

The upper limit of the number of processors for this approach is:

$$p \le \frac{N}{\max\limits_{0 \le j \le m-1} n_j} , \quad N = |G|, \quad n_j = |G_j| .$$

Special case of Fast Walsh Transform [9] .

$G = C_2 \times C_2 \times \ldots \times C_2$ (m groups); $|G| = N = 2^m$; $n_j = 2$, $0 \le j \le m-1$; $L(G_j) = 2$ .

For the number of processors $p = 2^i$ , where $i \le m-1$, transfers are needed in steps with numbers $j \ge m-i$ .

For unibus connection ( $k_1 = p = 2^i$ ),

$$T^{(2^i)}(\underbrace{C_2 \times C_2 \times \ldots \times C_2}_{m \text{ groups}}) = m \cdot 2^{m-i} \cdot t_a + i \cdot 2^{m-1} \cdot t_c \qquad (3)$$

Special case of Fast Chrestenson Transform [9] .

$G = C_q \times C_q \times \ldots \times C_q$ (m groups)

$|G| = N = q^m$; $n_j = q$ ; $L(G_j) \le q(q-1)$ , $0 \le j \le m-1$

For the number of processors $p = q^i$ , where $i \le m-1$, interprocessor transfers are needed in steps with numbers $j \ge m-i$ .

$$T^{(q^i)}(\underbrace{C_q \times C_q \times \ldots \times C_q}_{m \text{ groups}}) \le m q^{m-i}(q-1)t_a + q^{m-1} \cdot i(q-1)t_c \qquad (4)$$

For $q = 2^l$ (FFT over groups $C_q$ ) , $L(C_q) = 2q(\log_2 q - 1) + 2$;

and (2) gives for $p = q^i = 2^{li}$ processors:

$$T^{(2^{li})}(C_{2^{li}} \times C_{2^{li}} \times \ldots \times C_{2^{li}}) =$$

$$= 2^{l(m-i-1)+1} \cdot m(2^{l(l-1)+1}) \cdot t_a + 2^{l(m-l)} \cdot i(2^l - 1) \cdot t_c \qquad (5)$$

III. COMPUTER VERIFICATION OF THE NUMBER OF TRANSFERS.

To verify the the number of interprocessor transfers in formula (2), a simulated multiprocessor spectrum calculation has been performed by uniprocessor computer. The simulation algorithm provided an independent way to count the number of transfers; computer experiments with numerous sample groups confirmed the correctness of (2) and consequently of the derived formulas (3) - (5).

With the use of introduced concept of block, a simple rule for the

ordering of constituent groups has been determined. To minimize the number of transfers for a fixed set of the constituent groups $G_j$, it is necessary to order the groups according to the rule:

$$|G_0| \leq |G_1| \leq |G_2| \leq \ldots \leq |G_{m-1}| .$$

Small groups in the initial steps of the algorithm produce small block sizes, and the transfers are required in the minimal number of steps.

For sample calculations and comparison, several groups with $N = 512$ were chosen. Table 1 gives the numbers of transfers for the groups for different number of processors. It may seem at the first sight that low-order group implementation of $G$ may result in the lowest number of transfers due to the lower degree of "data intermixing" in the successive steps of spectrum calculation algorithm. However, small-size group "design" of $G$ results in the larger number of algorithm steps. As it can be seen from Table 1, $C_2^9$ group seems to be the worst choice . Nevertheless, $C_2^9$ turns out to be a good implementation in many cases due to relatively low number of the arithmetic operations.

To compare the different groups of Table 1 as to the speed of spectrum calculation, the assumption $t_m = t_a$ has been made ( $t_m$ is the time needed for one real multiplication). The ratio $t_m/t_a$ varies within the limits from 1 (Cyber-205, Cray-1,CDC Star-100 [15]) to 20 - 30 (Zilog Z-8000 [2]) for different computers and types of operands. Since in the constituent group algorithms chosen for our estimations [1] all multiplications are the multiplications by real constants, the choice of $t_m = t_a$ seems to be reasonable. The ratio $\kappa = \frac{t_c}{t_a}$ varied from 0.02 (fast bus as the Nanobus of Encore Multimax) to $\kappa = 1 - 1.5$. The choice of the upper limit for depends on speedup values to be given below.

The use of $p$ processors decreases the number of the arithmetic operations for spectrum calculation to $L^{(p)}(G) = L(G)/p$. We shall use the term "ideal speedup" to denote speedup equal to $p$ for $p$ processors. The

ideal speedup can never be achieved because of the additional time needed for interprocessor data transfers. However, for a bus that is fast enough it is possible to obtain the speedup rather close to ideal.

We shall take the ratio $T^{(1)}/(pT^{(p)})$ as a measure of closeness of an actual speedup $T^{(1)}/T^{(p)}$ to the ideal speedup. Figures 1, 2 show the dependence of the relative speedup for different groups of the order 512 of the bus speed and the number of processors. Table 2 summarizes the results of calculation of the equivalent number of additions (including converted multiplications and data transfers). The two criteria - high relative speedup and low equivalent number of additions - are conflicting. With the increase in the number of processors, the number of operations tends to decrease as well as the relative speedup. The comparison of different implementations of $G$ with $N = 512$ reveals that the fastest performance in most cases is provided by $C_2^6 \times Q_2$ group. The group is the best one for all $p$ in case of the fast bus ($\kappa = 0.02$). With the decrease of bus speed, the $C_2^6 \times Q_2$ group is still the best for $p = 2,4,8,16$, while for the larger values of $p$ and $\kappa > 0.02$, the $C_2^3 \times Q_2^2$ group has the smallest number of operations. The groups next to the best are $C_2^9$ (for all $p$ in the fast bus case and for the lower range of $p$ with the decrease of bus speed), $C_2^3 \times Q_2^2$ (for $p = 32$ and $\kappa = 0.1$ as well as for $k=0.5$, 1 and $p = 8$, 16), and $Q_2^3$ (for $\kappa = 0.5$, 1 and $p = 32$, 64).

It can be assumed that fast performance is typical for the groups $C_2$ and $Q_2$ (quaternion group) used as constituent groups in the direct product for $G$. The group algorithms for $C_2$ and $Q_2$ do not require multiplications (except of trivial ones). It could be expected that $G = C_2 \times C_4 \times Q_2$ will also be characterized by a low number of operations since $C_4$ also has only trivial multiplications in its algorithm. The calculations showed that the performance of this group is very close to that of $Q_2^3$ group being about 5% slower in the worst cases.

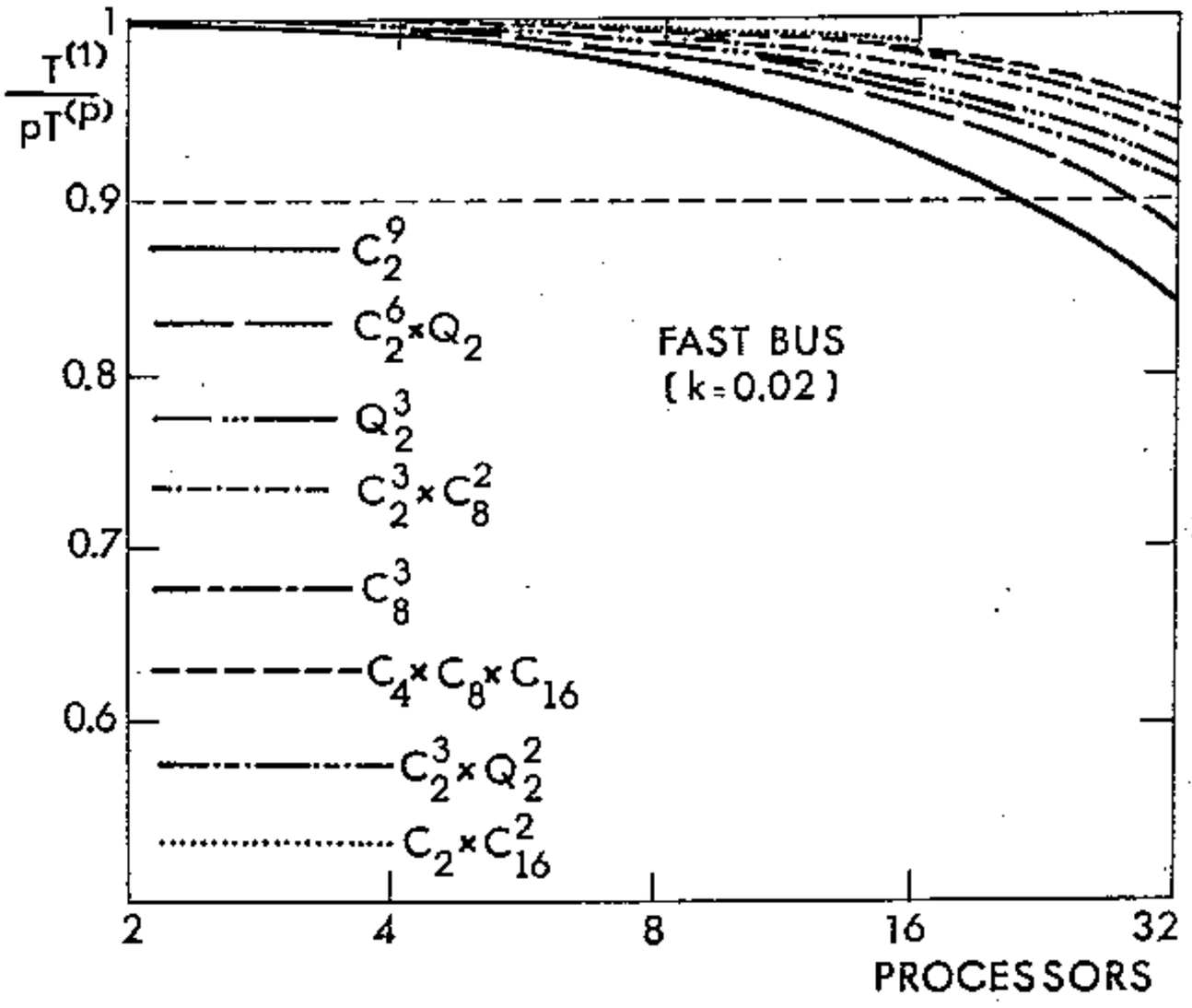


Fig. 1 . The relative speedup $T^{(1)}/pT^{(p)}$ for eight sample groups of the order 512 as a function of the number of processors (Fast bus with $K$ = 0.02).

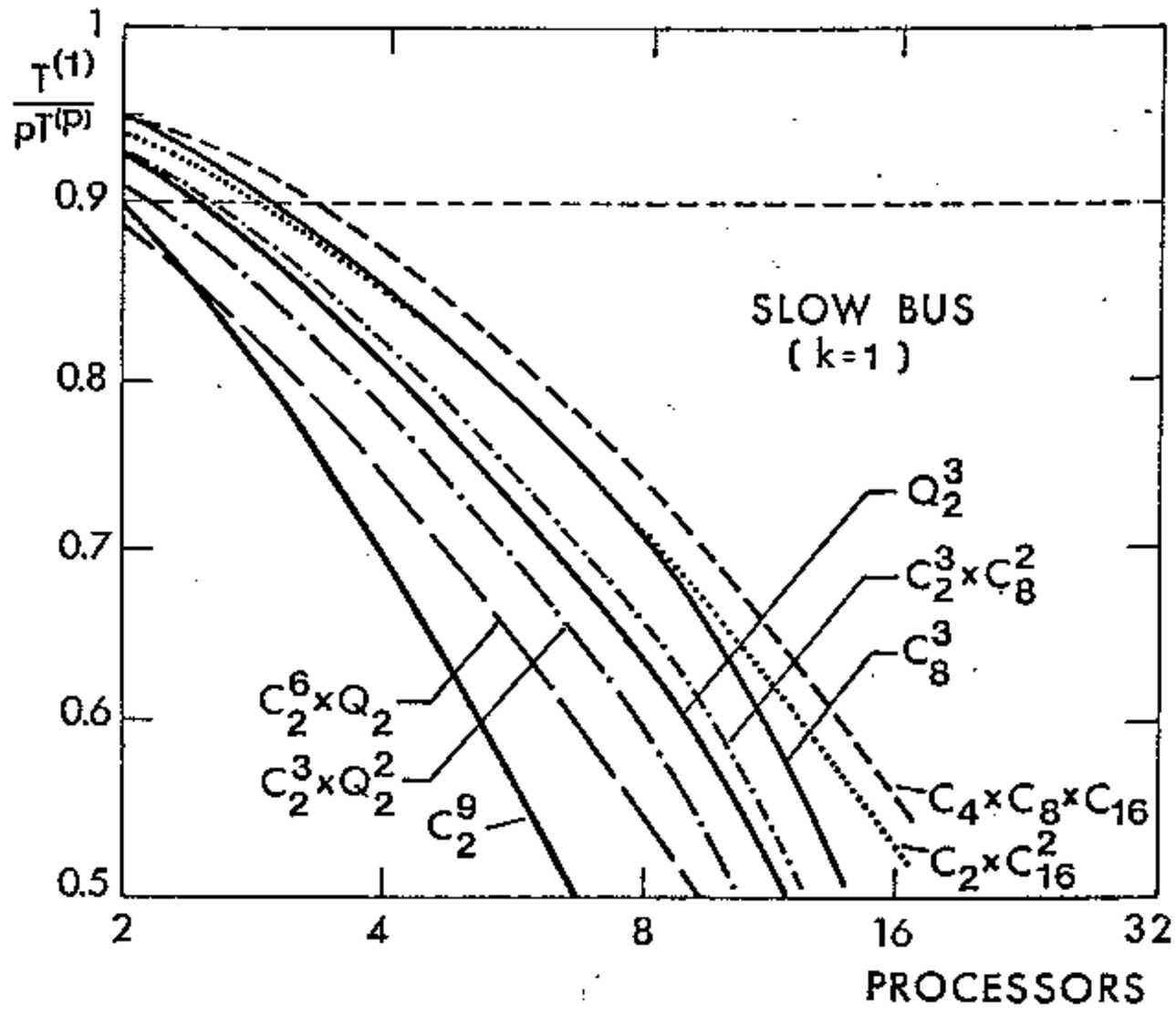


Fig. 2 . The relative speedup $T^{(1)}/pT^{(p)}$ for eight sample groups of the order 512 as a function of the number of processors (Slow bus with $K$ = 1).

## REFERENCES


1. D.F.Elliot and K.R.Rao, Fast Transforms: Algorithms,Analyses and Applications. New York: Academic Press, 1982.
2. T.Beth, Verfahren der schnellen Fourier-Transformationen. Leitfaden der angewandten Mathematik und Mechanik LAMM, Bd.61. B.G.Teubner, Stuttgart, 1984.
3. G.Apple and P.Wintz, "Calculation of Fourier transforms on finite Abelian groups", IEEE Trans. Inform. Theory, vol.IT-16, pp.233-236, 1970.
4. M.D.Atkinson, "The complexity of group algebra computations", Theor. Comp. Sci., vol.1, pp.205-209, 1977.
5. J.Neubuser, "Computing with groups and their character tables", Computing Suppl.4, pp.45-56, 1982.
6. L.Dornhoff, Group Representation Theory. New York: Marcel Dekker,1971.
7. E.A.Trachtenberg, "Construction of group transforms subject to several performance criteria", IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-33, pp.1521-1531, 1985.
8. M.G.Karpovsky, "Fast Fourier transforms on finite non-Abelian groups", IEEE Trans. Comput., vol.C-26, pp.1020-1030, 1977.
9. M.G.Karpovsky, Finite Orthogonal Series in the Design of Digital Devices, New York: Wiley, 1976.
10. M.G.Karpovsky and E.A.Trachtenberg, "Fourier transforms over finite groups for error detection and error correction in communication channels", Inform. Contr., vol.40, pp.335-358, 1979.
11. E.A.Trachtenberg and M.G.Karpovsky, "Filtering in a communication channel by Fourier transforms over finite groups", in Spectral Techniques and Fault Detection, M.Karpovsky, Ed., New York: Academic Press, 1985.
12. S.Winograd, "On computing the discrete Fourier transform", Math. Comput., vol.32, pp.175-199, 1978.
13. J.Morgenstern, "Note on a lower bound of the linear complexity of the fast Fourier transform", J. ACM, vol.20, pp.305-306, 1973.
14. D.B.Gannon and J. van Rosendale, "On the impact of communication complexity on the design of parallel numerical algorithms", IEEE Trans. Comp., vol.C-33, pp.1180-1194, 1984.
15. K.Hwang and F.A.Briggs, Computer Architecture and Parallel Processing, New York: McGraw-Hill, 1984.
16. Encore Multimax Technical Summary, Encore Corp., Marlboro, MA, 1986.


Table 1. The number of interprocessor transfers (unibus) for different constituent groups with $N=|G|=512$.

| p \ G | $C_2^9$ | $C_2^6 \times Q_2$ | $Q_2^3$ | $C_2 \times C_4^4$ | $C_4 \times C_8 \times C_{16}$ | $C_2 \times C_{16}^2$ | $C_{16} \times C_{32}$ | $C_2^3 \times Q_2^2$ |
|---|---|---|---|---|---|---|---|---|
| 2 | 256 | 256 | 256 | 256 | 256 | 256 | 256 | 256 |
| 4 | 512 | 384 | 384 | 384 | 384 | 384 | 384 | 384 |
| 8 | 768 | 448 | 448 | 640 | 448 | 448 | 448 | 448 |
| 16 | 1024 | 704 | 704 | 768 | 480 | 480 | 480 | 704 |
| 32 | 1280 | 960 | 832 | 1024 | 736 | 736 | – | 832 |
| 64 | 1536 | 1216 | 896 | 1152 | – | – | – | – |
| 128 | 1792 | – | – | – | – | – | – | – |
| 256 | 2048 | – | – | – | – | – | – | – |

Table 2. Equivalent number of additions ( $t_m = t_a$ ) for different groups of the order $N = 512$ as a function of bus speed parameter $k = t_c/t_a$ for different number of processors ( p ).

| p \ G | k = 0.02 (Fast bus) | | | | | | | | k = 0.1 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $C_2^9$ | $Q_2^3$ | $C_8^3$ | $C_4 \times C_8 \times C_{16}$ | $C_2^3 \times C_8^2$ | $C_2^3 \times Q_2^2$ | $C_2^6 \times Q_2$ | $C_2 \times C_{16}^2$ | $C_2^9$ | $Q_2^3$ | $C_8^3$ | $C_4 \times C_8 \times C_{16}$ | $C_2^3 \times C_8^2$ | $C_2^3 \times Q_2^2$ | $C_2^6 \times Q_2$ | $C_2 \times C_{16}^2$ |
| 2 | 2309 | 3205 | 4485 | 4997 | 3461 | 2693 | 2181 | 4293 | 2330 | 3226 | 4506 | 5018 | 3482 | 2714 | 2202 | 4314 |
| 4 | 1162 | 1608 | 2248 | 2504 | 1736 | 1352 | 1096 | 2152 | 1203 | 1638 | 2278 | 2534 | 1766 | 1382 | 1126 | 2182 |
| 8 | 591 | 809 | 1129 | 1257 | 873 | 681 | 553 | 1081 | 653 | 845 | 1165 | 1293 | 909 | 717 | 589 | 1117 |
| 16 | 309 | 414 | 574 | 634 | 446 | 350 | 286 | 546 | 390 | 470 | 630 | 672 | 502 | 406 | 342 | 584 |
| 32 | 170 | 217 | 297 | 327 | 233 | 185 | 155 | - | 272 | 283 | 363 | 386 | 299 | 251 | 232 | - |
| 64 | 103 | 118 | 158 | - | 126 | 102 | 92 | - | 226 | 190 | 230 | - | 198 | 174 | 190 | - |
| p \ G | k = 0.5 | | | | | | | | k = 1.0 (Slow bus) | | | | | | | |
| | $C_2^9$ | $Q_2^3$ | $C_8^3$ | $C_4 \times C_8 \times C_{16}$ | $C_2^3 \times C_8^2$ | $C_2^3 \times Q_2^2$ | $C_2^6 \times Q_2$ | $C_2 \times C_{16}^2$ | $C_2^9$ | $Q_2^3$ | $C_8^3$ | $C_4 \times C_8 \times C_{16}$ | $C_2^3 \times C_8^2$ | $C_2^3 \times Q_2^2$ | $C_2^6 \times Q_2$ | $C_2 \times C_{16}^2$ |
| 2 | 2432 | 3328 | 4608 | 5120 | 3584 | 2816 | 2304 | 4416 | 2560 | 3456 | 4736 | 5248 | 3712 | 2944 | 2432 | 4544 |
| 4 | 1408 | 1792 | 2432 | 2688 | 1920 | 1536 | 1280 | 2336 | 1664 | 1984 | 2624 | 2880 | 2112 | 1728 | 1472 | 2528 |
| 8 | 960 | 1024 | 1344 | 1472 | 1088 | 896 | 768 | 1296 | 1344 | 1248 | 1568 | 1696 | 1312 | 1120 | 992 | 1520 |
| 16 | 800 | 752 | 912 | 864 | 784 | 688 | 624 | 776 | 1312 | 1104 | 1264 | 1104 | 1136 | 1040 | 976 | 1016 |
| 32 | 784 | 616 | 696 | 680 | 632 | 584 | 616 | - | 1424 | 1032 | 1112 | 1048 | 1048 | 1000 | 1096 | - |
| 64 | 840 | 548 | 462 | - | 556 | 532 | 674 | - | 1608 | 996 | 1036 | - | 1004 | 980 | 1284 | - |