Fast Fourier Transforms Over Finite Groups by
Multiprocessor Systems

TATYANA D. ROZINER, MARK G. KARPOVSKY, SENIOR MEMBER, IEEE, AND LAZAR A. TRACHTENBERG, SENIOR MEMBER, IEEE

Abstract—This paper presents a method for an optimal implemen-
tation of General Discrete Fourier Transform (GDFT) algorithms over
finite groups (Abelian and non-Abelian) in a multiprocessor environ-
ment. Tradeoffs between hardware complexity/speed and computation
time are investigated for different multiprocessor implementations with
local nonshared memories (unibuses, complete communication networks).
Formulas are presented for the number of arithmetic operations, for
the number of interprocessor data transfers, and for the number of
communication links among the processors.

I. INTRODUCTION

F

AST Fourier Transforms (FFT) are well known to play
an important role in many application fields, such as
spectral analysis, digital filtering, image processing, video
transmission, etc. The increasing requirements of speed
in many real-time applications stimulated in recent years
the development of a number of new very fast FT algo-
rithms as well as numerous investigations on comparative
complexity of different FFT versions [2], [6], [18], [19],
[23]. A systematic approach to the problems of design of
FFT and complexity evaluation was developed by Beth
[2], [3] who generalized the results formulated in [1], [5]-
[7]. Of General Discrete Fourier Transform (GDFT) as a spe-
cial case of GDFT (General Discrete Fourier Transform)

based on the theory of representations of finite groups.
The powerful mathematical apparatus allowed Beth to ob-
tain complexity evaluations for different existing FT algo-
rithms as well as to design new versions of very fast
GDFT algorithms for uniprocessor systems in case of de-
composable group structure introduced on the input data
set. The methods of the theory of representations of finite
groups (Abelian and non-Abelian) were also used by other
researchers [9]-[14], [28], [29] for different applications
related to FFT.

As noted in [6], [19], and [20] 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 in-
crease of FFT, and with high progress in VLSI technol-
ogy, the multiprocessor highly parallel systems become
an attractive alternative compared to uniprocessor archi-
tectures. Many recent publications suggested a wide va-
riety of multiprocessor implementations for FFT, among
them numerous works on systolic arrays and array pro-
cessors with different interconnection networks (e.g.,
[17], [24]-[27], [29]-[32]).

In a multiprocessor environment, a new factor arises
that may strongly influence the efficiency of FFT algo-
rithm performance. The structure of the algorithm in-
volves numerous interprocessor data transfers which can,
in case of inappropriate or too slow processor communi-
cation network, become a reason for a degradation of per-
formance. Even in uniprocessor systems, the interregister
data transfers, loads, stores, and data copying operations
may take a considerable part of the total execution time
(up to 80 percent, in case of certain architectures, as
shown in [21]). The evaluations performed in [22] showed
that the minimum number of the interregister data trans-
fers in case of uniprocessor system is at least of the same
order as the number of arithmetic operations.

Algorithm-independent upper bounds on complexity in-
cluding the evaluation of the number of interprocessor
transfers were determined in [23] for the classical FFT
algorithm performance in terms of multiprocessor com-
munication network design. From the viewpoint of group
theory approach, the results of [23] 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 decom-
posable (possibly non-Abelian) group structure intro-
duced on the input data set, for the problems of GDFT
spectrum calculations by a multiprocessor SIMD system
with nonshared memory. The investigation of tradeoffs
between the number of operations (including data trans-

0096-3518/90/0200-0226$01.00 © 1990 IEEE

Manuscript received November 18, 1987; revised May 10, 1989. This
work was supported by the Office of Naval Research under Naval Research
Contract N00014-87-K-0755, and is part by the National Science Foun-
dation under Grant ECS-8512748.

T. D. Roziner and M. G. Karovsky are with the College of Engineer-
ing, Boston University, Boston, MA 02215.

L. A. Trachtenberg is with Drexel University, Philadelphia, PA 19104.
IEEE Log Number 8932777.
II. GDFT AS THE GENERALIZATION OF FFT

Discrete Fourier transforms considered in this work are the generalizations of the classical Fourier transforms (FT) in the following sense. For the classical 1-D (one-dimensional) DFT, the vector of input data \( f(0), f(1), \ldots \) \( f(N-1) \) is multiplied by the matrix whose elements are the powers of the \( N \)th root of unity, resulting in the vector of \( N \) spectrum values:

\[
\hat{f}(w) = \frac{1}{N} \sum_{n=0}^{N-1} f(n) W_N^{nw},
\]

where \( W_N = \exp(-2\pi j/N) \); \( j = \sqrt{-1} \); \( w = 0, 1, \ldots, N-1 \).

In terms of group theory approach, the matrix of FT corresponds to introducing the cyclic group structure \( C_N \) on the set of input data: the set \( \{0, 1, \ldots, N-1\} \) can be considered as a group, the group operation \( \ast \) is modulo-\( N \) addition (commutative operation). The set of group elements is closed with respect to group operation. Also, for each element \( a \) of \( C_N \), its inverse element is defined as \( a^{-1} \) such that \( a \ast a^{-1} = 0 = N \mod N \). Element 1 is the generator of \( C_N \); any other element can be obtained from 1 by repeated use of group operation.

Example 1: Group \( C_2 \) of the order 4 contains four elements: 0, 1, 2, 3. Group operation on \( \ast \) is modulo-4 addition. Evidently, for each pair of the group elements, the result of their modulo-4 addition is also the group element, e.g., 1 \( \ast 2 = 3 \), 2 \( \ast 3 = 2 \), etc. Inverse of \( 0 \) is \( 0 \), inverse of \( 1 \) is \( 1 \), inverse of \( 2 \) is \( 2 \), inverse of \( 3 \) is \( 3 \).

For the cyclic group \( C_N \), \( N \) distinct roots of unity \( 1, W_n, \ldots, W_{N-1} \) form the set of \( N \) distinct 1-D representations. A 1-D representation \( \mathcal{R} \) is the homomorphism \( R : C_N \rightarrow \mathbb{C} \) where \( \mathbb{C} \) is the field of complex numbers. For any two group elements \( a, b \in C_N \), \( R(a \ast b) = R(a) \ast R(b) \) (usual multiplication). Therefore, for a cyclic group, it is enough to define \( N \) distinct 1-D representations for group generator 1.

Example 2: For \( C_8 \), \( R_1(1) = 1 \) (trivial representation); \( R_1(1) = W_8 = j; R_1(1) = W_8^2 = -j; R_1(1) = W_8^3 = 1 \). To find \( R_1(2) \), the property of the homomorphism can be used: \( R_1(3) = R_1(1) \ast R_1(1) \ast R_1(1) = (-j)^3 = j \), etc.

<table>
<thead>
<tr>
<th>Group Element ( i )</th>
<th>( R_0(x) )</th>
<th>( R_1(x) )</th>
<th>( R_2(x) )</th>
<th>( R_3(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>j</td>
<td>-j</td>
<td>1</td>
<td>-j</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-j</td>
<td>1</td>
<td>-j</td>
</tr>
</tbody>
</table>

The table is the usual DFT matrix. To obtain FFT, the inverses of group elements in the leftmost column must be taken (row reordering).

which is the matrix of FFT (Fast Fourier Transform for 4 points: \([13, 27, 28]\). The spectrum \( f(w) \) is obtained by multiplication of vector of data by the matrix given by the table above:

\[
f(w) = \frac{1}{4} \sum \limits_{x=0}^{3} f(x) R_x(x^{-1}); \quad w = 0, 1, 2, 3.
\] (2.1)

The extension of this approach is to introduce more general group structure on the input data set. For a general group structure, its group operation may be noncommutative (non-Abelian groups). The theory of non-Abelian groups allows us to construct generalized transform (GDT) based on group representations that in the general case may be multidimensional (matrices with complex-valued elements). The GDT in this case can be also defined by a transform matrix; when the input data vector is multiplied by this matrix, the result is the vector of generalized spectrum elements. The GDT has properties analogous to DFT (linearity, scaling property, group convolution, Parseval’s theorem, etc. \([2, 12, 16]\). For GDT over certain non-Abelian groups, the transform matrix has the sparse and simple form; for example, non-Abelian quaternion group \( Q_8 \) (of the order 8, that is, a group of 8 elements) implies GDT matrix \( 8 \times 8 \) with 16 zero elements, and the nonzero elements are \( \pm 1, \pm j \) only, that is, no multiplication is needed (see Section III, Fig. 3).

Multidimensional DFT that is of interest in many applications is defined for \( m \) dimensions as \([1]\):

\[
f(w_0, w_1, \ldots, w_m-1) = \frac{1}{n_1 \cdot \ldots \cdot n_m} \sum \limits_{x_0=0}^{n_0-1} \ldots \sum \limits_{x_m=0}^{n_m-1} f(x_0, x_1, \ldots, x_m-1) \cdot W_w^{w_0x_0+n_1x_1+\ldots+n_mx_m},
\]

(2.2)

where \( W_w = \exp(-2\pi j/w) \); \( 0 \leq n_i \leq n_i-1 \); \( 0 \leq w_i \leq n_i-1 \); \( i = 0, 1, \ldots, m-1 \); the dimension \( i \) has \( n_i \) points.

In terms of group theory, multidimensional DFT is equivalent to the FT over the group \( G \) with the following structure:

\[
G = G_0 \times G_1 \times \ldots \times G_{m-1},
\]
where "**x**" denotes the direct product of cyclic (commutative) groups $G_0, G_1, \ldots, G_{m-1}$. The group theory approach allows us to obtain the matrix of $m$-dimensional FFT as the Kronecker product $(\otimes)$ of FFT matrices for the constituent groups.

**Example 3:** 3-D FFT (with 2 points in each dimension) can be described in terms of the group structure $G = C_2 \times C_2 \times C_2$. The FFT matrix is the direct product of three matrices $[I; I; I; I; I; I; I; I; I]$:  

$$
\begin{align*}
| & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
| & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
| & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
| & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
| & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
| & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
| & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
| & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1
\end{align*}
$$

(Fast Walsh Transform [1], [13].)

In this work, even more general multidimensional FFT (i.e., multidimensional DFT) is considered. The group structure $G = G_0 \times G_1 \times \cdots \times G_{m-1}$, introduced on the $N$-point data set $(N = n_0 \cdot n_1 \cdot \cdots \cdot n_{m-1})$, where $N$ is the order of group $G$, is that the number of elements of $G$ may include noncommutative groups as well as commutative (e.g., cyclic) ones. That implies that the matrix of DFT may be expressed in terms of smaller transform matrices (for commutative groups, in terms of the Kronecker product) [8], [12]. These generalized FFT's over finite non-Abelian groups have been widely used in different applications [9], [14], [15]. The rule of multiplication of the input data vector by such a matrix for fast DFT is equivalent to the factorization of the DFT matrix. As a consequence, the DFT algorithm can be performed in $m$ steps. Step number $i$ corresponds to the butterfly algorithm of the DFT for constituent group $G_i$ performed in parallel on a number of disjoint subsets of data (input data or the intermediate spectra). This creates a possibility of exploiting the parallelism of the algorithm in each step by use of a multiprocessor system. However, multiprocessor execution creates a problem of interprocessor data exchange. Both aspects of multidimensional DFT (parallelism and addition time needed for data transfers) are considered in the following sections. Estimation will be given for the execution time and for the number of data transfers as a function of group structure and of the number of processors.

**III. EVALUATION OF THE NUMBER OF OPERATIONS NEEDED FOR FFT OVER GROUP STRUCTURE $G = G_0 \times G_1 \times \cdots \times G_{m-1}$ 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 $\mathbb{C}$ of complex numbers, and let $GL(V)$ be the group of all nonsingular $d \times d$ matrices with elements in $\mathbb{C}$. A representation of $G$ is a homomorphism $R : G \rightarrow GL(V)$, that is, for $x, y \in G$, $x \cdot y \in G$, $R(x \cdot y) = R(x) \circ R(y)$ (where $\circ$ is the group operation). 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 $R_s$ is equivalent to some unitary representation $R_{\phi}$, i.e., to a representation with unitary matrix $R_{\phi}(x)$, so there exists a $Q \in GL(V)$ such that $R_s(x) = Q^* R_{\phi}(Q) Q$ for all $x \in G$. The set of all irreducible unitary representations for $G$ has the following orthogonality properties [7]. Let $R_{\phi}^{(s)}$ denote the $(s, t)$th element of matrix representation $R_{\phi}(x)$, and let $R_{n_t}(x)$ be of dimension $d_n$, then

$$
\frac{1}{|G|} \sum_{x \in G} R_{\phi}^{(s, t)}(x) R_{\phi}^{*(s', t')}(x) = \frac{1}{d_n} \delta_{n_t} \delta_{n'_t} \delta_{s_t} \delta_{s'_t} \tag{3.1}
$$

$$
\sum_{x \in G} d_n \text{Tr}(R_{\phi}(x)) = N \delta_{n_t} \tag{3.2}
$$

where $R(G)$ is the set of all irreducible unitary representations of $G$; $e$ is the identity element of $G$; $\delta$ is the Kronecker symbol; $\text{Tr}$ is the trace of matrix $A$, and $A$ denotes the adjoint matrix (transposed and complex conjugate) of $A$. The relations (3.1) and (3.2) allow us to define the direct and inverse General Discrete Fourier Transform (GDFT) over $G$ as follows. If $f \in \mathbb{F}(G) \rightarrow \ldots$, then

$$
f(w) = \sum_{x \in G} f(x) R_{\phi}(x^{-1}) \tag{3.3}
$$

where $x^{-1}$ is the inverse of $x$ in $G$ (cf. formula (2.1), Example 2 of Section II where $d_n = 1$).

Let $G, |G| = N$, be a direct product of groups $G_0, 0 \leq j \leq m - 1$, and the elements of group $G_0$, as $x_j$; $G_j, 0 \leq j \leq m - 1$. Let $R(G)$ be the set of all irreducible unitary representations of $G$ with elements $R_{\phi} \in R(G)$, and let $R(G_j)$ be the set of all irreducible unitary representations of $G_j$ with elements $R_{\phi} \in R(G_j), 0 \leq j \leq m - 1$ [12].

Following [12], note the following:

1) If $x \in G$, then $x = (x_0, x_1, \ldots, x_{m-1})$, $x_j \in G_j$ (the $x_j$ may be represented as $m$-tuple of elements of groups $G_j, 0 \leq j \leq m - 1$).

2) If $R_{\phi} \in R(G)$, then $R_{\phi}(X) = R_{\phi}(x_0, x_1, \ldots, x_{m-1}) = \otimes_{j=0}^{m-1} R_{\phi}(x_j), x_j \in G_j$; symbol $\otimes$ denotes the Kronecker product of matrices. Thus, for every $R_{\phi} \in R(G)$, we may denote $w = (w_0, w_1, \ldots, w_{m-1})$. 

}
The direct Fourier transform over group $G$ may be defined as follows (12):

$$
\hat{f}(w) = \frac{d}{N} \sum_{m} \sum_{x_{m-1}} f(x_{0}, x_{1}, \ldots, x_{m-1}) \otimes R_{m}(x_{m-1}^{-1})
$$

$$
= \frac{d}{N} \sum_{m} \left( \left( \sum_{x_{m-1}} \sum_{m} f(x_{0}, x_{1}, \ldots, x_{m-1}) \otimes R_{m}(x_{m-1}^{-1}) \right) \otimes R_{m-1}(x_{m-1}^{-1}) \right).
$$

(3.4)

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

Step 0:

$$
f_{0}(x_{0}, x_{1}, \ldots, x_{m-1}) = \hat{f}(x_{0}, x_{1}, \ldots, x_{m-1});
$$

$$
f_{j}(w_{0}, w_{1}, \ldots, w_{m-1}) = \sum_{x_{m-1}} f_{0}(x_{0}, x_{1}, \ldots, x_{m-1}) \otimes R_{m}(x_{m-1}^{-1}).
$$

Step 1:

$$
f_{1}(w_{0}, w_{1}, x_{2}, \ldots, x_{m-1}) = \sum_{x_{0}} f_{0}(w_{0}, x_{1}, \ldots, x_{m-1}) \otimes R_{0}(x_{0}^{-1}).
$$

Step $j$:

$$
f_{j+1}(w_{0}, w_{1}, \ldots, w_{j}, x_{j+1}, \ldots, x_{m-1}) = \sum_{x_{j}} f_{j}(w_{0}, w_{1}, \ldots, w_{j-1}, x_{j}, \ldots, x_{m-1}) \otimes R_{0}(x_{j}^{-1}).
$$

Step $(m - 1)$:

$$
f_{m}(w_{0}, w_{1}, \ldots, w_{m-1}) = \sum_{x_{m-1}} f_{m-1}(w_{0}, w_{1}, \ldots, w_{m-2}, x_{m-1}) \otimes R_{m-1}(x_{m-1}^{-1}).
$$

During Step $j$, the variables $w_{0}, w_{1}, \ldots, w_{j-1}, x_{j+1}, \ldots, x_{m-1}$ are fixed, and the summation is performed by variable $x_{j} \in G_{j}$.

$$
\hat{f}(w_{0}, w_{1}, \ldots, w_{m-1}) = \frac{d}{N} \sum_{m} f_{m}(w_{0}, w_{1}, \ldots, w_{m-1}).
$$

Let $X_{j}$ be the $i$th element of $G$: $X_{j} \in G$, $0 \leq i \leq N - 1$. It will be convenient to list the elements of $G$ in a certain order. If $x_{j}$ designates the $i$th element of group $G_{j}$, $0 \leq i \leq n_{j} - 1$, $0 \leq j \leq m - 1$, then the elements $X_{0}, X_{1}, \ldots, X_{N_{0} - 1}$ of $G$ are ordered as follows:

$$
X_{0} = (x_{0}^{0}, x_{0}^{1}, x_{0}^{2}, \ldots, x_{0}^{n_{0} - 1})
$$

$$
X_{1} = (x_{1}^{0}, x_{1}^{1}, x_{1}^{2}, \ldots, x_{1}^{n_{1} - 1})
$$

$$
X_{2} = (x_{2}^{0}, x_{2}^{1}, x_{2}^{2}, \ldots, x_{2}^{n_{2} - 1})
$$

$$
\vdots
$$

$$
X_{N_{0} - 1} = (x_{N_{0} - 1}^{0}, x_{N_{0} - 1}^{1}, x_{N_{0} - 1}^{2}, \ldots, x_{N_{0} - 1}^{n_{N_{0} - 1} - 1}).
$$

(3.5)

Element $X_{j} = (x_{j}^{0}, x_{j}^{1}, x_{j}^{2}, \ldots, x_{j}^{n_{j} - 1})$ precedes the element $X_{0} = (x_{0}^{0}, x_{0}^{1}, x_{0}^{2}, \ldots, x_{0}^{n_{0} - 1})$ if $l_{0} = j_{0}, l_{1} = j_{1}, \ldots, l_{k-1} = j_{k-1}$, and $l_{k} < j_{k}$ for some $k$, $0 \leq k \leq m - 1$.

Example 4: Let $G = G_{1} \times G_{2} \times G_{3}$; $n_{1} = n_{2} = n_{3} = 2, n_{4} = 6$; $N = n_{0} n_{1} n_{2} = 24$; $m = 3$ ($S_{3}$ is the permutation group of the order 6). Elements of $G$ are listed as follows (only superscripts of $x_{j}$ are given): group number $j$, $0 \leq j \leq 2$, is implicitly determined by the position of superscript in a 3-tuple corresponding to $X_{j} \in G$.

$$
\begin{align*}
X_{0} & = (0, 0, 0) \quad X_{4} = (0, 1, 0) \quad X_{8} = (0, 3, 1) \\
X_{1} & = (1, 0, 0) \quad X_{5} = (1, 3, 0) \\
X_{2} & = (0, 1, 0) \quad X_{6} = (0, 4, 0) \\
X_{3} & = (1, 1, 0) \quad X_{7} = (1, 4, 0) \\
X_{4} & = (0, 2, 0) \quad X_{8} = (0, 5, 0) \\
X_{5} & = (1, 2, 0) \quad X_{10} = (1, 5, 0) \\
\end{align*}
$$

(3.6)

The generalized Fourier transform over group $G = G_{0} \times G_{1} \times \cdots \times G_{m-1}$ is performed in $m$ steps as shown earlier. Step 0, $0 \leq j \leq m - 1$, involves the calculation of intermediate spectrum using the FFT algorithm specified by the nature of group $G_{j}$. To perform the spectrum calculation of step $j$, $N/n_{j}$ subsets of elements of the intermediate spectrum are to be taken as input data for the algorithm of group $G_{j}$. The structure of each one of $N/n_{j}$ subsets is as follows: $\{X_{0}, X_{1}, \ldots, X_{N_{0} - 1}\}$ ($X_{j}$ denotes an element of intermediate spectrum array), where

$$
\begin{align*}
X_{0} & = (x_{0}^{0}, x_{1}^{0}, \ldots, x_{j}^{0}, x_{j+1}^{0}, \ldots, x_{N_{0} - 1}^{0}) \\
X_{1} & = (x_{0}^{1}, x_{1}^{1}, \ldots, x_{j}^{1}, x_{j+1}^{1}, \ldots, x_{N_{0} - 1}^{1}) \\
X_{2} & = (x_{0}^{2}, x_{1}^{2}, \ldots, x_{j}^{2}, x_{j+1}^{2}, \ldots, x_{N_{0} - 1}^{2}) \\
\vdots
\end{align*}
$$

(3.7)
that is, each set \((3.7)\) must contain all elements of group \(G_i\) in \((j + 1)\)th position; elements \(x_0 \in G_0, x_1 \in G_1, \ldots, x_{j-1} \in G_{j-1}\) are constant for each set \((3.7)\) but different for different sets.

From the viewpoint of practical calculation, we can consider the formally defined procedure of spectrum evaluation \((3.4)\) as the \(m\)-step procedure that transforms the input array of \(N\) values into the output array of the same size. In the following consideration, we shall denote the elements of input, output, or any intermediate spectrum array by their numbers only, since the values of array elements are not important for our purpose (that is, to trace the data distribution and transfers in case of multiple processors). We shall assume also that for the butterfly algorithm of any constituent group, the numbers of input elements are retained for the outputs. For example, if \(X_i\) and \(X_j\) are entered in Step \(j\) into the algorithm of \(C_i\) group, the resulting outputs \(Y_i + X_i, Y_i - X_i\) will be considered as updated elements number \(i\) and \(k\), respectively, of the intermediate spectrum array of \(N\) elements. (For \(C_i, R_i(x) = (-1)^x; x, \in \{0, 1\}.)\)

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 \([23]\). In each one of \(m\) steps of spectrum calculation, the \(N\) elements of the intermediate spectrum are partitioned into disjoint subsets \((3.7)\). In step \(j\), \(N/\eta_j\) subsets (each containing \(\eta_j\) elements) are to be used as input data for butterfly algorithm of the constituent group \(G_i\). In order to avoid idle processors in any step, \(N/(pn)\) must be an integer for any \(j, 0 \leq j \leq m - 1\).

Denote the number of operations needed to perform the butterfly algorithm of \(G_i\) on a single input data set \((3.7)\) (with one processor) as \(L(G_i)\). (In the general case, the algorithm is not characterized by the pair of numbers—the number of additions/subtractions and the number of multiplications. We shall assume that \(L(G_i)\) is the equivalent number of additions.)

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

\[
L(G) = \sum_{j=0}^{m-1} \sum_{i=0}^{n-1} L(G_i) = N \sum_{j=0}^{m-1} \frac{L(G_i)}{n_j}. \tag{3.8}
\]

To generalize \((3.8)\) for the case of \(p\) parallel processors, denote \(L_{\text{pr}}(G)\) the number of operations needed for spectrum calculation over \(G\) with \(p\) processors. Evidently,

\[
L_{\text{pr}}(G) = \frac{L(G)}{p} = \frac{N}{p} \sum_{j=0}^{m-1} \frac{L(G_i)}{n_j}. \tag{3.8a}
\]

However, for \(p > 1\), the data transfers among the processors are needed (at least in one of the algorithm steps).

Therefore, the total time \(T_{\text{pr}}(G)\) needed for spectrum calculation over \(G\) with \(p\) processors is:

\[
T_{\text{pr}}(G) = L_{\text{pr}}(G) \cdot t_s + M_{\text{pr}}(G) \cdot t_c,
\]

where \(t_s\) is the time needed for one addition/subtraction, \(t_c\) is the time needed for one data transfer (communication time), and \(M_{\text{pr}}(G)\) is the total number of transfers needed for spectrum calculation over \(G\) with \(p\) processors.

The value of \(M_{\text{pr}}(G)\) 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 \(M_{\text{pr}}(G)\), the communication network is needed where each processor can communicate directly with any other processor (if the communication is required by the algorithm), and is able to receive or to transfer a unit of data during communication time \(t_c\). This type of network is of high cost for a large number of processors (although not always the complete network is needed). It seems more reasonable to assume another extreme case when all processors communicate via single bus ("unit-bus" connection). In the last case, during the 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 \(j, 0 \leq j \leq m - 1\), the \(N/(pn)\) 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 the operations is also growing.

Define two parameters: \(p = N/p — \) the number of elements of \(G\) per processor; \(n_0, n_1, \ldots, n_{m-1}\)—block size for group \(G_j, 0 \leq j \leq m - 1\).

The concept of block size will be helpful in evaluation of the number of interprocessor transfers. In consequence of \((3.5)\), the accepted ordering of the elements of \(G\) is as follows. For processing over group \(G_0\) (Step 0 of the algorithm), \(N/n_0\) sets of elements of \(G\) are to be entered into the algorithm of \(G_0\) as input data. Each set contains \(n_0\) elements, and they follow in succession (again, we list only the numbers of the elements):

\[
\{0, 1, 2, \ldots, n_0 - 1\} \quad (\text{Set 0})
\]

\[
\{n_0, n_0 + 1, n_0 + 2, \ldots, 2n_0 - 1\} \quad (\text{Set 1})
\]

\[
\{n_0, n_1, n_1 + 1, \ldots, n_0 + \cdots + n_{m-1} - 1 = N - 1\} \quad (\text{Set } (N/n_0) - 1).
\]
For processing over group $G_i$ (Step 1), $N/n_i$ sets of elements of the intermediate spectrum are to be entered into the butterfly algorithm of $G_i$; each set contains $n_i$ elements. The elements are chosen "skipping over $n_0 - 1".

\begin{equation}
\begin{align*}
\{0, n_0, 2n_0, \ldots, (n_1 - 1)n_0\} & \quad \text{(Set 0)} \\
\{1, n_0 + 1, 2n_0 + 1, \ldots, (n_1 - 1)n_0 + 1\} & \quad \text{(Set 1)} \\
\{2, n_0 + 2, 2n_0 + 2, \ldots, (n_1 - 1)n_0 + 2\} & \quad \text{(Set 2)} \\
\{n_0 - 1, 2n_0 - 1, 3n_0 - 1, \ldots, (n_1 - 1)n_0 + n_0 - 1 = n_0n_1 - 1\} & \quad \text{(Set $(n_0 - 1)$)}
\end{align*}
\end{equation}

The elements listed above form block 1 for group $G_i$; there are $N/n_0n_1$ blocks in total for $G_i$, each containing $n_0n_1$ elements. Similarly, for group $G_j$, $0 \leq j \leq m - 1$, there are $N/(n_0n_1 \cdots n_j)$ blocks each containing $n_0n_1 \cdots n_j$ elements. (Evidently, for $G_{m-1}$, there is a single block containing all $N$ elements of group $G_m$.) For the group $G_m$, elements in block lines are chosen "skipping over $n_0n_1 \cdots n_{m-2} - 1".

Example 5: Let $G = C_3 \times C_5 \times C_6$ (where $S_6$ is the permutation group of the order $6$); $N = |G| = 24, m = 3, n_0 = n_1 = 2, n_2 = 6$. Block structure for $3$ steps of the spectrum algorithm is as follows:

<table>
<thead>
<tr>
<th>Step 0</th>
<th>Step 1</th>
<th>Step 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>{0, 1} Block 1</td>
<td>{0, 2, 4, 6, 8, 10} Block 2</td>
<td>0, 12 Block 1</td>
</tr>
<tr>
<td>{2, 3} Block 2</td>
<td>{1, 3, 5, 7, 9, 11} Block 3</td>
<td>1, 13 Block 2</td>
</tr>
<tr>
<td>\ldots</td>
<td>{12, 14, 16, 18, 20, 22} Block 11</td>
<td>2, 14 Block 1</td>
</tr>
<tr>
<td>{20, 21} Block 11</td>
<td>{13, 15, 17, 19, 21, 23} Block 12</td>
<td>10, 22</td>
</tr>
<tr>
<td>{22, 23} Block 12</td>
<td>\ldots</td>
<td>11, 23</td>
</tr>
</tbody>
</table>

Since before processing (before Step 0) $N/p = p_i$, successive elements of $G$ were delivered to the 1st processor, next $p_i$ successive elements to the 2nd processor, etc., the block sizes in successive steps of the spectrum calculation allow us to trace the distribution of the intermediate spectrum elements in block lines among different processors and to determine whether the interprocessor data transfers are needed in a certain step. The following rule is valid: if the block size $n_0n_1 \cdots n_j \leq p_i = N/p$, no data transfers are needed in Step $j$. If the block size $n_0n_1 \cdots n_j > p_i$, the 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 the transfers are never needed in Step 0 (block size $n_0$) if our previous assumption is valid that $N/(p_m)$ is an integer for any $0 \leq j \leq m - 1$. In particular, $N/(p_m) \geq 1$ is an integer which means $n_0 \leq p_1$ (no transfers needed). For Step 0, there are $N/n_0$ trivial blocks each containing only one line.

For the case when transfers are needed ($n_0n_1 \cdots n_j > p_i = N/p$), define parameter $K_j$, $1 \leq j \leq m - 1$, as $K_j = n_0n_1 \cdots n_j/p_i = (n_0n_1 \cdots n_jp)/(N/K_j)$ is the number of processors per block of group $G_j$. By the definition, $K_j$ is an integer greater than 1. The number of lines (i.e., group $G_j$ sets) in a block of $G_j$ is

\begin{equation}
N/n_0n_1 \cdots n_{j-1} = \frac{N}{n_0n_1 \cdots n_{j-1}} = \frac{N}{n_0n_1 \cdots n_{j-1}} \cdot \frac{p_j}{p_j} = \frac{N}{p_j} \cdot K_j.
\end{equation}

Since $n_0n_1 \cdots n_{j-1}$ is an integer as well as $N/(p_j)$ (by the assumption made above), $K_j$ is also an integer. The relation (3.10) means also that it is always possible to distribute the lines of a block equally among $p$ processors (in case transfers are needed in Step $j$).

The number of transfers in Step $j$ of the algorithm depends upon the relation between $n_j$ and $K_j$. Consider two cases: $n_j > K_j$ and $n_j < K_j$.

Case $n_j > K_j$: Assume that Step 1 is the one where transfers are needed. Since the elements in any block column follow in succession, and since the $n_0n_1 \cdots n_j$ elements of a block belong to $K_j$ different processors, then the group order $n_j$ must be an integer multiple of $K_j$: $n_j = A_j K_j, A_j \geq 1$ integer.

Block size for group $G_j$ may be expressed as $n_0n_1 \cdots n_j = n_0n_1 \cdots n_{j-1} \cdot K_j$. Since $n_0n_1 \cdots n_j$ and $K_j$ are integers, the value of $A_j = n_j/K_j$ is also an integer.
Deliver first \( n_0 n_1 \cdots n_{j-1} / K_j \) lines of a block for \( G_j \) to the first one of the participating processors, the second \( n_0 n_1 \cdots n_{j-1} / K_j \) lines to the second processor, etc., then in each line of a block \( \lambda_j = n_j / K_j \) elements belong to the proper processor (that is, they are already located in its local memory), and the remaining \( n_j - n_j / K_j \) elements are to be transferred from other processors. The total number of transfers in all blocks of \( G_j \) is, in \( N/n_j \) lines of all blocks together is

\[
N \left( \frac{n_j - n_j / K_j}{n_j} \right) = N \left( 1 - \frac{1}{K_j} \right).
\]

Case \( n_j \leq K_j \): In this case, \( n_0 n_1 \cdots n_{j-1} \leq K_j n_0 n_1 \cdots n_{j-1} \), or \( n_0 n_1 \cdots n_{j-1} > n_0 n_1 \cdots n_{j-1} \) if \( K_j = 1 \). In other words, one block column contains more elements than the number of elements per processor. The block column size must be an integer multiple of \( p_1 \) (block column is distributed among \( N_j > 1 \) processors):

\[
n_0 n_1 \cdots n_{j-1} = N_j p_1.
\]

Block size

\[
n_0 n_1 \cdots n_{j-1} = \frac{n_0 n_1 \cdots n_{j-1}}{p_1} = \frac{N_j}{\frac{p_1}{n_j}} = n_j.
\]

Since \( p_1 \) and \( n_j \) are integers, \( N_j = 1 \) is also an integer. It follows by definition that \( N_j > 1 \):

\[
N_j = \frac{n_0 n_1 \cdots n_{j-1}}{p_1} = \frac{n_0 n_1 \cdots n_{j-1}}{p_1} = n_j.
\]

\( N_j = 1 \) for \( K_j = n_j \) which is the case we considered above.

In the last case, all elements of each block line belong to different processors. It is possible to distribute \( n_0 n_1 \cdots n_{j-1} \) lines of the block among \( K_j \) processors in such a way that only one element of each line belongs to the proper processor (that is, this element is already located in memory of the processor that the line is delivered to), and the remaining \( n_j - 1 \) elements are to be transferred from the other processors. The total number of transfers for all blocks is

\[
N \left( \frac{n_j - 1}{n_j} \right) = N \left( 1 - \frac{1}{n_j} \right).
\]

Both cases \( (n_j \geq K_j \) and \( n_j < K_j \) may be unified in one formula for the number of transfers among \( p \) processors in the beginning of Step \( j \):

\[
M^{p^j}(G_j) = N \left( 1 - \frac{1}{\min(n_j, K_j)} \right)
\]

(note that \( K_j \) by definition depends on the number of processors).

Formula (3.8) may be generalized now for the case of \( p \) processors. Total time \( T^{p^j}(G) \) needed for spectrum calculation over \( G = G_0 \times G_1 \times \cdots \times G_{n-1} \) is

\[
T^{p^j}(G) = L^{p^j}(G) t_j + M^{p^j}(G) t_j.
\]

\[
= N \left( \frac{\sum_{j=0}^{n-1} L(G_j)}{n_j} + k_j t_j \frac{\sum_{j=0}^{n-1} \phi_j}{p} \left( \frac{n_j - 1}{\min(n_j, K_j)} \right) \right)
\]

where \( N = |G| = n_0 n_1 \cdots n_{n-1} \); \( n_j = |G_j| \), \( 0 \leq j \leq m - 1 \); \( p \) is the number of processors; \( t_j \) is the time for one addition; \( t_i \) is the time for one data transfer; \( L(G) \) is the number of operations (converted to the equivalent number of additions) to calculate the spectrum over \( G_j \) (single butterfly algorithm with one processor); \( k_j = 2 \), for the complete communication network, and \( k_i = p \), for unibus connection of processors; \( p_j = N/p \) is the number of elements of \( G \) per processor; \( \phi_j(p, n_0, \ldots, n_j) \) is \( 0 \) for \( n_0 n_1 \cdots n_j \leq p_j = N/p \) and \( \phi_j(p, n_0, \ldots, n_j) = 1 \) otherwise; \( K_j \) is the number of processors per one block of \( G \); \( L^{p^j}(G) \) is the total number of operations for spectrum calculation over \( G \) (converted to equivalent number of additions), with \( p \) processors; \( M^{p^j}(G) \) is the total number of interprocessor transfers of data.

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

\[
p \leq N \left( \frac{1}{\min(n_j, K_j)} \right) = N / \max(n_j, K_j).
\]

Example 6: \( G = C_2 \times C_2 \times Q_2 \) by 4 processors; \( N = 32, m = 3, n_0 = n_1 = 2, n_2 = 8 \) (see Fig. 1). Number of additions/subtractions: \( 8 + 8 + 20 = 36 \). (A processor can perform \( C_2 \)-butterfly with a pair of input data at a time; for example, \( P_3 \) processes the pair \( (0, 1) \) and then the next pair \( (2, 3) \) in Step 0.) Number of data transfers: \( 24 = 32 \times (1 - 1/4) \). \( \phi_j = 0, \phi_j = 1 \).

Special Case of \( n_j = B_j p_j \), Integer \( B_j \geq 1 \) for \( \forall j \), \( 0 \leq j \leq m - 1 \).

In this case, formula (3.11) can be simplified. It can be shown that the data transfers are needed in the last step only:

\[
\phi_0 = \phi_1 = \cdots = \phi_{m-2} = 0, \quad \text{and} \quad \phi_{m-1} = 1.
\]

In the last step, \( K_{n-1} = p \), and \( n_{n-1} \geq p \), so that \( \min(n_{n-1}, K_{n-1}) = p \). Formula (3.11) is simplified as follows:

\[
T^{p^j}(G) = N \left( \frac{\sum_{j=0}^{n-1} L(G_j)}{n_j} + k_j t_j \left( \frac{1 - 1}{p} \right) \right).
\]

(3.12)
Special Case of Fast Walsh Transform [13]

\[ G = C_2 \times C_2 \times \cdots \times C_2 = C_2^m \text{ (m groups)}; \]
\[ |G| = N = 2^n; \quad n_j = 2, \quad L(G_j) = 2, \]
\[ 0 \leq j \leq m - 1. \]

For the number of processors \( p = 2^i \), where \( 0 \leq m - 1 \), transfers are needed in steps with numbers \( j \geq m - i \), since block size for Step \( j \) is

\[ n_{m-1} \cdots n_{m-i} = 2^j \cdot 2^i \cdot \frac{N}{2^i} = 2^{m-i} \cdot \frac{N}{2^i} = 2^{m-i} > 2^{m-i} \]

for \( j + 1 > m - i \), or \( j \geq m - i \).

For unibus connection (\( k_i = p = 2^i \))

\[ T^{2^i}(C_2^m) = m \cdot 2^{m-i} \cdot t_o + 1 \cdot 2^{m-i} \cdot t_c \quad (3.13) \]

Example 7: \( G = C_2 \times C_3 \times C_3 \times C_3 \) by 4 processors: 16 transfers (unibus), and 16 additions/subtractions (see Fig. 2). Data transfers are needed in Steps 2 and 3 (before the processing).

The number of transfers in (3.13) for Fast Walsh Transform is similar to the result of Gannon and Rosendale [23] (differing by a factor of 2 since the algorithm used in [23] includes spectrum reshuffling).

Special Case of Fast Chesterson Transform [13]

\[ G = C_q \times C_q \times \cdots \times C_q = C_q^m; \]
\[ |G| = N = q^n; \quad n_j = q; \quad L(G_j) = q(q - 1), \]
\[ 0 \leq j \leq m - 1. \]

For the number of processors \( p = q^i \), where \( i \leq m - 1 \), the data transfers are needed in steps with numbers \( j \geq m - i \).

\[ T^{q^i}(C_q^m) \leq mq^j(q - 1) \cdot t_o + q^{m-i} \cdot i(q - 1) \cdot t_c \quad (3.14) \]

(In 3.14), the relation \( L(C_q) \leq q(q - 1) \) was used that overestimates the number of operations needed for spectrum calculation.) In case of FFT over a group \( C_q \), \( q = 2^i \),

\[ L(C_q) = 2q \log_2 q + 2. \]

By (3.11) for \( p = 2^i = 2^m \), we obtain

\[ T^{2^i}(C_q^m) = 2^i(m - i + 1) - 1 \cdot m(2^i(l - 1) + 1) \cdot t_o + 2^i(m - i) \cdot i \cdot (2^i - 1) \cdot t_c \quad (3.15) \]

The Number of Communication Links Among \( p \) Processors

Up to now, the term **"complete communication network"* has been used, without entering into details of processors connection. It was assumed that the network allows communication in parallel among any processors that need to exchange data. In fact, our model of "complete" network does not always need to have all \( p(p - 1)/2 \) communication links among \( p \) processors; which links are to be present in the network is determined by the structure of group \( G = G_0 \times G_1 \times \cdots \times G_{m-1} \). The concept of block introduced in this section allows us to obtain the formula for the exact number of communication links for the arbitrary group structure \( G = G_0 \times G_1 \times \cdots \times G_{m-1} \), and for any \( p \leq (N/(\max_{0 \leq j < m-1} n_j)) \).

One case when the complete network is needed is rather evident: the case when \( p = n_{m-1} \). For \( j = m - 1 \), there
is a single block of $N$ elements, and $K_{n-1} = P$. By definition, $N_j = K_j/n_j$ (we recall that $K_j$ is the number of different processors per block, and $N_j$ is the number of different processors per one block column). If $p = n_{n-1}$, then $N_{n-1} = 1$, and all the processors in each block line are different. Evidently, the complete network is needed in this case. As to other cases, consider again two possibilities: $n_j \geq K_j$ and $n_j < K_j$ (Note that bidirectional communication links among all processors are assumed.)

Case $n_j \geq K_j (n_0n_1 \cdots n_{j-1} > p_j = N/p)$

It has been shown that in this case, $n_j = A_j K_j$, where $A_j \geq 1$ is the number of block columns with the elements located in the memory of one processor (there are $K_j$ processors in a block). The number of communication links is 

$$\psi_{j} (n_j, K_j) = \frac{2^{n_j/N_j}}{N_j} \left( \frac{N}{N_j} \right)^{n_j} \frac{1}{p_j}$$

where $n_j = n_0n_1 \cdots n_{j-1}$ is the number of blocks.

Case $n_j < K_j$

In this case, $N_j = n_0n_1 \cdots n_j/p_j = K_j/n_j; N_j > 1$ is an integer ($N_j$ is the number of different processors in one block column). A block column contains $n_0n_1 \cdots n_j$ elements; the number of different processors in one block line is $n_j = K_j/N_j$, and the number of links needed for the transfers among $n_j$ processors is $(N_j - 1)$.

This value is to be multiplied by $p_j/n_j$, to obtain the total number of needed communication links, since there are $K_j$ processors per one block. $K_j/n_j$ sets of different processors (each set consisting of $n_j$ processors), and $n_j$ is the number of blocks:

$$\psi_{j} (n_j, K_j) = 1, \quad \text{for } n_j \geq K_j, \quad \text{and } 0 \text{ otherwise}$$

$$\psi_{j} (n_j, K_j) = 1, \quad \text{for } n_j < K_j, \quad \text{and } 0 \text{ otherwise}$$

$$K_j = n_0n_1 \cdots n_{j-1} \frac{N}{N_j}$$

Special Case of Fast Walsh Transform [11], [13]

For $G = C_2^1 \times C_2^2 \times \cdots \times C_2^n = C_2^n$, the number of communication links does not depend on the order of group $G$. For any $|G| = N = 2^n$, the number of links is the function of the number of processors only. It is the minimal network for $p$ processors: all the other group structures involve the larger number of links for $p > 2$. Assume $p = 2^i; i$ as it has been shown earlier, for Fast Walsh transform, the interprocessor transfers are needed in steps with the numbers $j = m - i$ where $i = \log_2 p$.

$$K_j = 2^{-i}(n_{j-1} + 1); \quad j = m - i, \quad K_j \geq 2, \quad \text{and } n_j = 2 \leq K_j$$

$$\text{CL}^{(j)}(C_2^n) = \frac{(m - 1 - (m - i - 1)) (1/2)^{-i}}{2^i} \frac{1}{p \log_2 p}$$

(bidirectional links). The network for Fast Walsh Transform is the binary cube with $p$ vertices and $(1/2) p \log_2 p$ edges. Table I summarizes the numbers of communication links needed for different groups with $N = 64$ and $N = 512$. The minimal communication network is the binary cube (for $C_2^8$ and $C_2^4$ groups) for any value of $p$. For comparison, one of the table columns shows the number of links in the complete network for each $p$. It can be seen that for the smaller groups ($N = 64$), in most cases, the complete network is needed; the exceptions are $C_2^8$ (for all $p$) and $C_2^4$ (for $p = 8, 16, \text{i.e., for } p > n_{n-1}$). In the case of large groups (e.g., with $N = 512$), for $p > n_{n-1}$, some of the sample groups are next to the best (minimal) network of $C_2^n$ having smaller network than the complete one. However, these networks, although being incomplete, contain about twice as many links as the binary cube network of Fast Walsh Transform. Note that among the groups next to the best, there are those containing non-Abelian quaternion groups $Q_2$ or a factor of $Q_2$ combined with $C_2$ groups ($Q_2 \times C_2^1, C_2 \times Q_2$).

IV. COMPUTER VERIFICATION OF THE NUMBER OF TRANSFERS AND SAMPLE CALCULATIONS FOR DIFFERENT GROUPS

To verify the number of interprocessor transfers in formula (3.11), a simulated multiprocessor spectrum calculation has been performed by a 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 (3.11) and consequently of the derived formulas (3.12)-(3.15).

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 $G_i$, it is necessary to order the groups according to the rule:

$$|G_0| \leq |G_1| \leq \cdots \leq |G_{n-1}|.$$

Small groups in the initial steps of the algorithm produce small block sizes, and the transfers are required in the minimal number of steps. Table II shows the transfer numbers for a fixed sample set of seven constituent groups (six groups $C_2$ and one $C_4$ group; $N = 512$) taken in different order. For two processors, the number of transfers is the same for all versions of $G$. However, for $p > 2$, the group $C_2^2 \times C_4$ always has the lowest number of transfers, while the group $C_4 \times C_2^2$ is "the worst" having the greatest number of transfers for all $p > 2$.

For sample calculations and comparison, several groups with $N = 64$ and with $N = 512$ were chosen. Table III gives the numbers of transfers for eleven sample groups of order 512, for different numbers of processors. It may seem at the first sight that low-order group implementation of $G$ (when $n_1$ are small) 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 can be seen from Table III, $C_2^2$ group seems to be the worst choice having the largest number of data transfers. Nevertheless, $C_2^2$ 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 III with respect to the speed of spectrum calculation, the assumption $t_m = t_a$ has been made ($t_a$ is the time needed for one real multiplication). The ratio $t_m/t_a$ varies within the limits from 1 (Cyber-205, Gray-1, CDC Star-100 [33]) to $20 \div 30$ (Zilog 2-8000 [2]) for different computers and types of operands. Since in the constituent groups chosen for our estimations, all multiplications are real ([11], Ch. 5, cyclic groups) or trivial (the butterfly for the quaternion group shown in Fig. 3 [15]), the choice of $t_m = t_a$ seems to be reasonable. The ratio $k = t_i/t_a$ varied from 0.02 (fast bus as the Nanobus of Encore Multimax [34]) to $k = 1 + 1.5$. The choice of the upper limit for $k$
TABLE III
THE NUMBER OF INTERPROCESSOR TRANSFERS (UNLESS) FOR DIFFERENT
CONSORTIUM GROUPS WITH \(N = 512\)

<table>
<thead>
<tr>
<th>(p)</th>
<th>(c_{2}^{(1)})</th>
<th>(c_{2}^{(2)})</th>
<th>(c_{p}^{(1)})</th>
<th>(c_{p}^{(2)})</th>
<th>(c_{2p}^{(1)})</th>
<th>(c_{2p}^{(2)})</th>
<th>(c_{2p}^{(16)})</th>
<th>(c_{2p}^{(16)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>4</td>
<td>512</td>
<td>384</td>
<td>384</td>
<td>384</td>
<td>384</td>
<td>384</td>
<td>384</td>
<td>384</td>
</tr>
<tr>
<td>8</td>
<td>768</td>
<td>448</td>
<td>448</td>
<td>448</td>
<td>448</td>
<td>448</td>
<td>448</td>
<td>448</td>
</tr>
<tr>
<td>16</td>
<td>1024</td>
<td>704</td>
<td>704</td>
<td>704</td>
<td>704</td>
<td>704</td>
<td>704</td>
<td>704</td>
</tr>
<tr>
<td>32</td>
<td>1280</td>
<td>900</td>
<td>632</td>
<td>1024</td>
<td>726</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>2560</td>
<td>1216</td>
<td>636</td>
<td>2560</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3. Flow diagram ("bommerly") for the quaternion group \(G_2\).

The value of the ratio \(T^{(1)}(p^{TP})\) may be chosen as a criterion for the comparison of different groups for different \(p\) and \(k\). Other criteria may be the total system cost, the number of arithmetic operations, the accuracy of calculation, etc.

Fig. 4 shows the curves of relative speedup \(T^{(1)}(p^{TP})\) for sample groups \(C_2^2(N = 64)\) and \(C_2^2(N = 512)\) for different values of \(k\) (that is, for different speeds of the bus). The smaller group has lower values of speedup due to the relatively large number of transfers compared to the number of arithmetic operations. It can be seen that if the minimum acceptable speedup is chosen as \(0.9\) of the ideal speedup value (\(p = p\) for processors), then for bus speed parameter \(k \geq 1\) it is not worthwhile to have two processors instead of one for \(G = C_2^2\). Fast increase in the number of transfers results in a decrease of the speedup for \(k = 1\), \(p = 2\) to 0.86 of the "ideal" value only. For \(k = 1\), \(p = 4\) for the same group the relative speedup becomes only 0.6, that is, the absolute speedup is 2.4 only. For a fast bus \((k = 0.02)\), both groups \(C_2^2\) and \(C_2^2\) may have up to 16 processors (relative speedup values for \(p = 2, 4, 8, 16\) and \(k = 0.02\), are higher than or equal to \(0.9\)).

For another pair of sample groups \(G = C_2 \times C_2^4 (N = 64)\) and \(G = C_2 \times C_2 \times C_2^3 (N = 512)\), the same rule can be observed: the smaller group allows us to have the smaller number of processors for the same bus speed, to provide the relative speedup not lower than a given cutoff value (0.9 for our examples). It can be concluded that smaller groups are more sensitive to the decrease of the bus speed; for the same value of \(k\), the speedup curves of the smaller group are lower than those for the larger group.

Figs. 5 and 6 illustrate the speedup change with the increase in the number of processors for different bus speeds in case of two quaternion groups. From Figs. 5 and 6, it can be seen that there is a fast speedup decrease (below the cutoff level 0.9) for 4 processors and \(k \geq 0.25\) in case of \(C_2 \times C_2\) while the large group \(C_2 \times C_2 \times C_2\) allows \(k = 0.5\) for 4 processors for the chosen cutoff value 0.9. Fast bus with \(k = 0.02\) in case of \(G_2 \times C_2 \times C_2\) allows us to have up to 32 processors for the same restrictions on the relative speedup; for \(k = 0.02\) and \(p = 32\), the speedup will be about \(30 \times 32 \times 0.94\).

For the chosen minimum speedup value, the bus speed of \(k = 0.5\) is about the critical value for small groups.

depends on speedup values to be obtained below; it will be shown that for \(k \geq 2\), the speedup that can be achieved with two processors instead of one becomes too low to compensate the expenses on the additional hardware. With the increase of \(p\), the speedup decreases.

The use of \(p\) processors instead of one reduces 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 the speedup equal to \(p\) for \(p\) processors. The ideal speedup can never be achieved because of the additional time needed for data transfers. However, for a bus that is fast enough, it is possible to obtain the speedup rather close to ideal.

The total time \(T^{(1)}(G)\) needed for spectrum calculation over \(G = G_2 \times G_2 \times \ldots \times G_{m-1}\) was evaluated for 8 sample groups of the order 512, and for 6 sample groups of the order 64, for different number of processors \(p\) and for different bus speeds characterized by the parameter \(k = t_b/t_c\). The \(T^{(1)}(p)\) value has been compared to \(T^{(1)}(p)\) time needed for unprocessor spectrum calculation (no data transfers). Both \(T^{(1)}(p)\) and \(T^{(1)}(p)\) were calculated in terms of the equivalent number of additions (multiplications and data transfers were converted to equivalent numbers of add operations) using the relations \(t_a = t_m, t_a = k \times t_a\).
The relative speedup $T^{11}/(p T^{10})$ for groups $G = C_2^k (N = 512)$ and for $G = C_2 (N = 64)$ as a function of the number of processors and the bus speed. The value of $k = 0.02$ corresponds to the fast bus case; dotted lines show speedup for $C_2^2$.

The relative speedup $T^{11}/(p T^{10})$ for the small quaternion group $G = Q_2 (N = 64)$ as a function of the number of processors and of the bus speed.

The relative speedup $T^{11}/(p T^{10})$ for the large quaternion group $G = Q_2 \times Q_2 \times Q_2 (N = 512)$ as a function of the number of processors and of the bus speed.

$C_2^k$ and $Q_2 \times Q_2$ (two processors give the relative speedup not less than 0.9). For $k > 0.5$, the relative speedup is less than 0.9 for $p = 2$. It means that if we choose the relative speedup $\geq 0.9$ as the only criterion for the choice of the number of processors, it makes no sense to have two processors instead of one, if the communication time is greater than a half of one addition time. For large groups ($C_2^2$ and $Q_2 \times Q_2 \times Q_2$), the bus may be slower under the same restriction on the minimum relative speedup: for $Q_2 \times Q_2 \times Q_2$, it can be shown that the critical value of $k$ is about 1.55. and for $C_2^2$, it is about 1.1 (for two processors). Large groups are "less sensitive" to the bus speed because of the relatively large number of the arithmetic operations compared to the number of data transfers; the bus speed becomes less significant.

The quaternion-based groups $C_2^2$ and $Q_2^k$ compared to $C_2^2$ and $C_2$ have the higher values of relative speedup for the same number of processors and bus speed. However, both quaternion-based groups as well as the groups $C_2^k$, $C_2$, are characterized by rather low speedup values compared to other groups of the same order (see Tables IV and VI and Figs. 7 and 8). The relative speedup $T^{11}/(p T^{10})$ may be considered as an efficiency index of a multiprocessor system or as a measure of how much the expenses on the additional hardware (processors) are covered. With the increase in the number of processors, the relative speedup tends to decrease drastically; however, the total time of spectrum calculation also decreases. To compare different group performance (for different $p$ and $k$) in terms of calculation time, one can compare the equivalent number of additions (multiplications and data transfers converted to the equivalent number of additions).

Table V summarizes the results of calculation for the equivalent numbers of additions (including converted numbers of transfers) which allows one to compare different groups $G$ and to determine the ones with the fastest performance. The two criteria—relative speedup and the equivalent number of additions—are evidently 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 structures of $G$ with $N = 512$ (Table V) reveals that the fastest performance in most cases is provided by $C_2^2 \times Q_2$ group. This group is the best one for all $p$ values in case of the fast bus ($k = 0.02$). With the decrease of bus speed ($k > 0.02$), the group is still the best for $p = 2, 4, 8, 16$, while for the larger values of $p$ and $k > 0.02$, the $C_2^2 \times C_2^2$ group has the smallest number of operations. The groups next to the best are $C_2^4$ (for all $p$ in 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 $k = 0.1$ as well as for $k = 0.5, 1$ and $p = 8, 16$), and $C_2^2$ (for $k = 0.5, 1$ and $p = 32, 64$).

The calculations of the number of equivalent additions were repeated for $N = 512$ for the slow multiplication case $t_m = 5t_s$ to examine the influence of the multiplication speed on the performance of groups that need nontr-
TABLE IV  
RELATIVE SPEEDUP FOR DIFFERENT GROUPS OF THE ORDER N = 512 AS A FUNCTION OF BUS SPEED AND OF THE NUMBER OF PROCESSES (P): L = L;

<table>
<thead>
<tr>
<th>k (M)</th>
<th>( C_2 \times C_8 )</th>
<th>( C_2 \times C_16 )</th>
<th>( C_4 \times C_4 \times C_16 )</th>
<th>( C_4 \times C_8 \times C_16 )</th>
<th>( C_4 \times C_4 \times C_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>0.956</td>
<td>0.956</td>
<td>0.956</td>
<td>0.956</td>
<td>0.956</td>
</tr>
<tr>
<td>0.1</td>
<td>0.908</td>
<td>0.908</td>
<td>0.908</td>
<td>0.908</td>
<td>0.908</td>
</tr>
<tr>
<td>0.5</td>
<td>0.852</td>
<td>0.852</td>
<td>0.852</td>
<td>0.852</td>
<td>0.852</td>
</tr>
<tr>
<td>1.0</td>
<td>0.800</td>
<td>0.800</td>
<td>0.800</td>
<td>0.800</td>
<td>0.800</td>
</tr>
</tbody>
</table>

Fig. 7. The relative speedup \( T^{11}/(pT^{0.02}) \) for eight sample groups of the order 512 as a function of the number of processes (fast bus with \( k = 0.02 \)).

vial multiplications. The results showed the same pattern—the fastest groups were again \( C_2 \times C_8 \), \( C_2 \times C_16 \), \( C_4 \times C_8 \), and \( C_4 \times C_4 \times C_16 \).

It can be assumed that fast performance is typical for the group \( C_4 \) and \( Q_2 \) used as constituent groups in the direct product of \( G \) since the group algorithms for \( C_4 \) and \( Q_2 \) do not need multiplications. It can be expected that the group \( G = C_4 \times C_4 \times C_16 \) will also be characterized by a low number of operations since \( C_4 \) also has only trivial multiplications by \( \pm 1, \pm j \). The total calculation time for the group turns out to be very close to the time for the \( Q_2 \) group, being only about 5 percent greater.

For small sample groups (\( N = 64, \) Table VI), the group \( C_2 \times Q_2 \) is the fastest one (for \( p = 2, 4, 8 \)). The groups next to the best are \( C_4 \) (for the fast bus, \( k = 0.02, 0.1 \) and/or \( p = 2, 4 \)), and \( Q_2 \) (for \( k = 0.5, 1.0 \) and \( p = 8 \)). The slowest groups in both cases of \( N = 64 \) and \( N = 512 \) are those that contain high-order cyclic groups \( C_4 \) and \( C_{16} \) in the direct product of \( G \).

For \( N = 64 \), the \( C_4 \) group is about 1.6/2 times slower compared to the best group \( C_2 \times Q_2 \). It is very likely that, in general, the best performance (the fastest one) can be achieved with a combination of \( C_2 \) and \( Q_2 \) groups in the direct product of \( G \).

V. CONCLUSIONS

Investigations of tradeoffs between the number of operations needed for GDFV over arbitrary finite groups.
### TABLE V

<table>
<thead>
<tr>
<th>$N$</th>
<th>$C_0$</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_4$</th>
<th>$C_{5}C_{1}C_{4}$</th>
<th>$C_{2}C_{3}$</th>
<th>$C_{2}C_{4}$</th>
<th>$C_{2}C_{5}$</th>
<th>$C_{2}C_{6}$</th>
<th>$C_{2}C_{6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2039</td>
<td>3205</td>
<td>4485</td>
<td>4997</td>
<td>3451</td>
<td>2683</td>
<td>2191</td>
<td>4293</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1157</td>
<td>1680</td>
<td>2248</td>
<td>2964</td>
<td>1736</td>
<td>1269</td>
<td>1096</td>
<td>2162</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>653</td>
<td>1609</td>
<td>1129</td>
<td>1237</td>
<td>673</td>
<td>681</td>
<td>523</td>
<td>1081</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>309</td>
<td>414</td>
<td>574</td>
<td>624</td>
<td>448</td>
<td>360</td>
<td>288</td>
<td>544</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>170</td>
<td>217</td>
<td>297</td>
<td>327</td>
<td>219</td>
<td>160</td>
<td>159</td>
<td>318</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>103</td>
<td>118</td>
<td>158</td>
<td>-</td>
<td>126</td>
<td>102</td>
<td>92</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$k = 0.1$:

<table>
<thead>
<tr>
<th>$N$</th>
<th>$C_0$</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_4$</th>
<th>$C_{5}C_{1}C_{4}$</th>
<th>$C_{2}C_{3}$</th>
<th>$C_{2}C_{4}$</th>
<th>$C_{2}C_{5}$</th>
<th>$C_{2}C_{6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2230</td>
<td>3220</td>
<td>4508</td>
<td>5018</td>
<td>3482</td>
<td>2714</td>
<td>2202</td>
<td>4314</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1313</td>
<td>1633</td>
<td>2278</td>
<td>2634</td>
<td>1368</td>
<td>1282</td>
<td>1126</td>
<td>2160</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>653</td>
<td>1640</td>
<td>1185</td>
<td>1293</td>
<td>609</td>
<td>577</td>
<td>509</td>
<td>1117</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>390</td>
<td>470</td>
<td>620</td>
<td>672</td>
<td>502</td>
<td>496</td>
<td>342</td>
<td>568</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>277</td>
<td>363</td>
<td>383</td>
<td>396</td>
<td>299</td>
<td>251</td>
<td>252</td>
<td>492</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>168</td>
<td>199</td>
<td>230</td>
<td>-</td>
<td>136</td>
<td>174</td>
<td>190</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

$k = 0.5$:

<table>
<thead>
<tr>
<th>$N$</th>
<th>$C_0$</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_4$</th>
<th>$C_{5}C_{1}C_{4}$</th>
<th>$C_{2}C_{3}$</th>
<th>$C_{2}C_{4}$</th>
<th>$C_{2}C_{5}$</th>
<th>$C_{2}C_{6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2432</td>
<td>3228</td>
<td>4508</td>
<td>5120</td>
<td>3584</td>
<td>2815</td>
<td>2304</td>
<td>4416</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1408</td>
<td>1702</td>
<td>2432</td>
<td>2988</td>
<td>1920</td>
<td>1536</td>
<td>1360</td>
<td>2536</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>900</td>
<td>1024</td>
<td>1344</td>
<td>1472</td>
<td>906</td>
<td>888</td>
<td>744</td>
<td>1488</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>593</td>
<td>702</td>
<td>912</td>
<td>964</td>
<td>584</td>
<td>568</td>
<td>576</td>
<td>1152</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>384</td>
<td>468</td>
<td>546</td>
<td>596</td>
<td>382</td>
<td>372</td>
<td>416</td>
<td>832</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>240</td>
<td>296</td>
<td>356</td>
<td>396</td>
<td>228</td>
<td>228</td>
<td>456</td>
<td>912</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE VI

<table>
<thead>
<tr>
<th>$G$</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_4$</th>
<th>$C_{5}C_{1}C_{4}$</th>
<th>$C_{2}C_{3}$</th>
<th>$C_{2}C_{4}$</th>
<th>$C_{2}C_{5}$</th>
<th>$C_{2}C_{6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G_{ab}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

$k = 0.02$ (FAST BUS):

<table>
<thead>
<tr>
<th>$N$</th>
<th>$C_0$</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_4$</th>
<th>$C_{5}C_{1}C_{4}$</th>
<th>$C_{2}C_{3}$</th>
<th>$C_{2}C_{4}$</th>
<th>$C_{2}C_{5}$</th>
<th>$C_{2}C_{6}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2566</td>
<td>3456</td>
<td>4736</td>
<td>5248</td>
<td>3740</td>
<td>2944</td>
<td>2422</td>
<td>4844</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1664</td>
<td>1984</td>
<td>2624</td>
<td>2988</td>
<td>2119</td>
<td>1738</td>
<td>1402</td>
<td>2804</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1244</td>
<td>1486</td>
<td>1856</td>
<td>2108</td>
<td>1213</td>
<td>1170</td>
<td>954</td>
<td>1908</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>890</td>
<td>1072</td>
<td>1382</td>
<td>1576</td>
<td>884</td>
<td>868</td>
<td>732</td>
<td>1464</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>593</td>
<td>702</td>
<td>912</td>
<td>964</td>
<td>584</td>
<td>568</td>
<td>576</td>
<td>1152</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>384</td>
<td>468</td>
<td>546</td>
<td>596</td>
<td>382</td>
<td>372</td>
<td>416</td>
<td>832</td>
<td></td>
</tr>
</tbody>
</table>

Sample calculations with several groups of different order and different structures revealed some typical features of the constituent groups and the influence of the group order on the performance speed. Smaller groups ($N = 64$) are more sensitive to the speed of the processor communication network. For these groups, the increase in the communication time relative to the addition time results in a drastic decrease of the speedup which can make the use of multiple processors unreasonable. The fastest groups, in most cases, were the ones combining small cyclic groups ($C_4$) and non-Abelian quaternions ($Q_2$) in the direct product for $G$. As shown by an earlier work [15], the quaternion groups as the components of the direct product for $G$ in many cases show the optimal performance as to the accuracy of calculations.

### ACKNOWLEDGMENT

The authors would like to thank Dr. R. C. Brower and Dr. L. Levitin for useful discussions.

### REFERENCES


