324-77.
05-09-72 XR SR 3,662,161

United States Patent

Bergland et al.

[15] **3,662,161**

[45] **May 9, 1972**

[54] GLOBAL HIGHLY PARALLEL FAST FOURIER TRANSFORM PROCESSOR

[72] Inventors: **Glenn D. Bergland,** Morris Township, Morris County; **Donald E. Wilson,** Brookside, both of N.J.

[73] Assignee: **Bell Telephone Laboratories, Incorporated,** Murray Hill, Berkeley Heights, N.J.

[22] Filed: **Nov. 3, 1969**

[21] Appl. No.: **873,587**

[52] **U.S. Cl.** .....**235/156,** 324/77 G, 444/1

[51] **Int. Cl.** **G06f 7/38,** G06f 15/34

[58] **Field of Search** .....235/156; 324/77 G; 340/15.5

[56] **References Cited**

UNITED STATES PATENTS

3,517,173	6/1970	Gilmartin, Jr. et al.	235/156
3,544,775	12/1970	Bergland et al.	235/151.31

OTHER PUBLICATIONS

Bergland, FFT Hardware Implementations– An Overview " IEEE Trans. on Audio & Electroacoustics" June 1969 pp. 104– 108.

Bergland–Wilson, A FFT Algorithm for a Global, Highly Parallel Processor " IEEE Trans. on Audio & Electroacoustics" Vol. AU– 17, No. 2 June 1969 pp. 125– 127.

Pease, Organization of Large Scale Fourier Processors " Journal of the Association for Computing Machinery," Vol. 16, No. 3, July 1969 pp. 474– 482.

Primary Examiner—Malcolm A. Morrison
Assistant Examiner—David H. Malzahn
Attorney—R. J. Guenther and William L. Keefauver

[57] **ABSTRACT**

A fast Fourier transform processor and associated process wherein an input sequence of samples is broadcast to each of a plurality of parallel processing elements where sets of accumulated sums of products of these samples with appropriate trigonometric function values are maintained. These sets of accumulated sums are then individually Fourier transformed in parallel to form the Fourier coefficients corresponding to the original input sequence.

17 Claims, 13 Drawing Figures

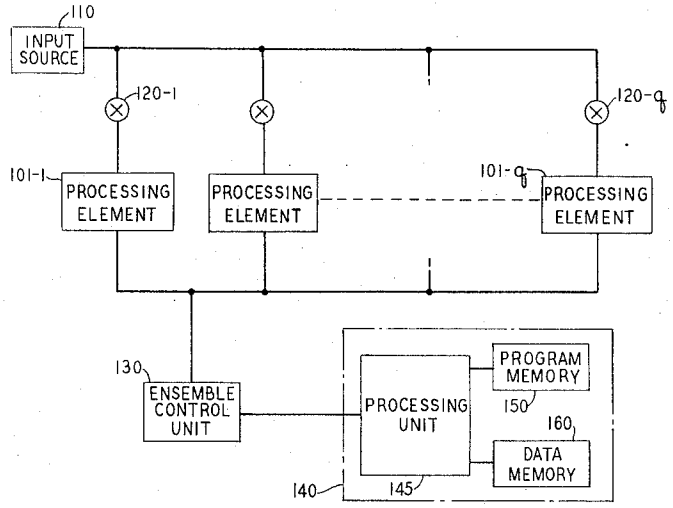


FIG. 1

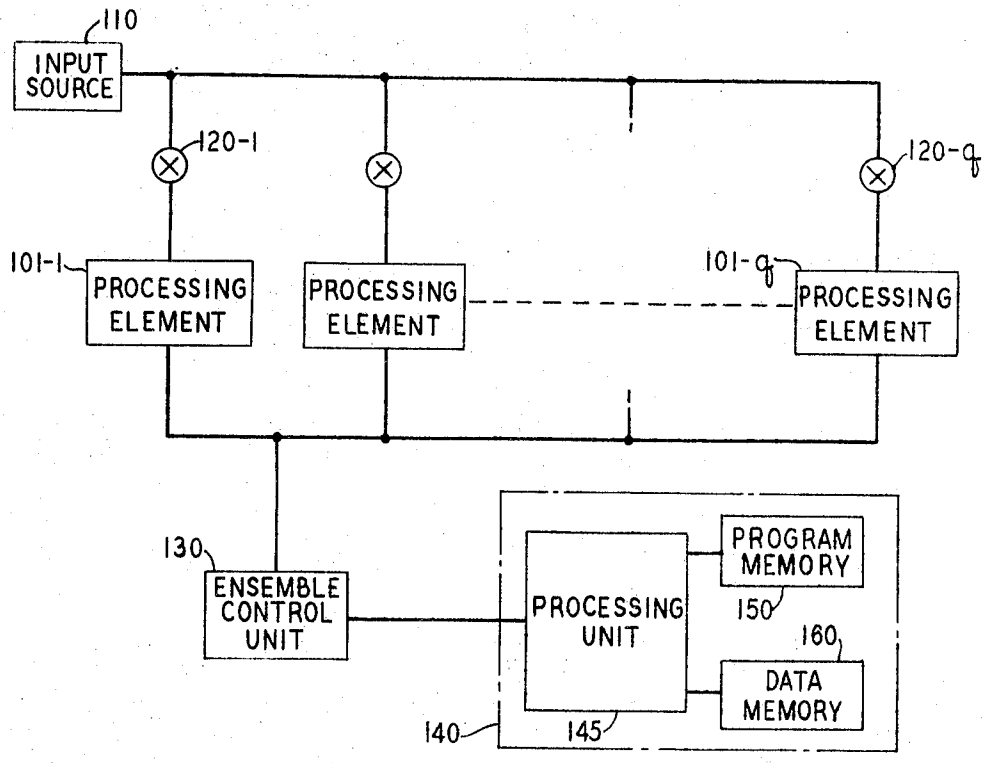
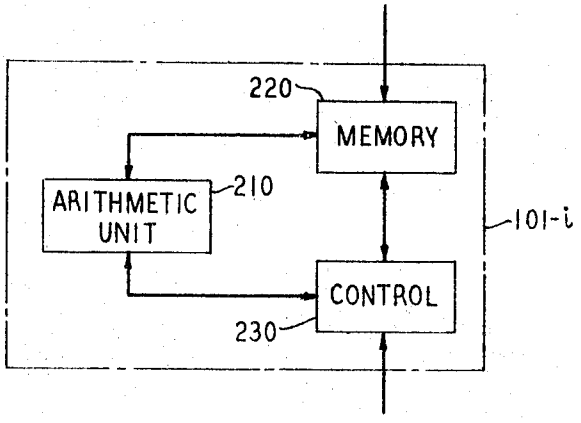


FIG. 2



INVENTORS G. D. BERGLAND
D. E. WILSON
BY R. O. Nimz
ATTORNEY

FIG. 3

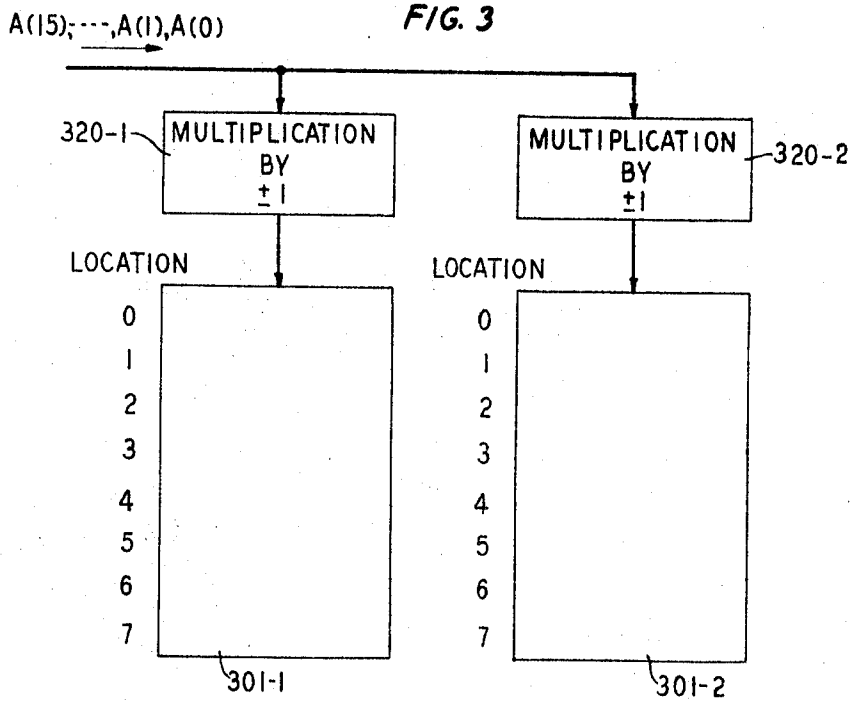


FIG. 4

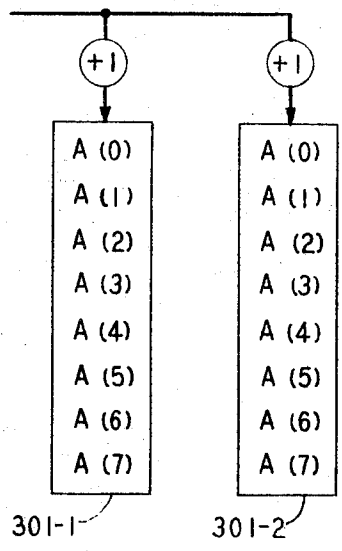


FIG. 5

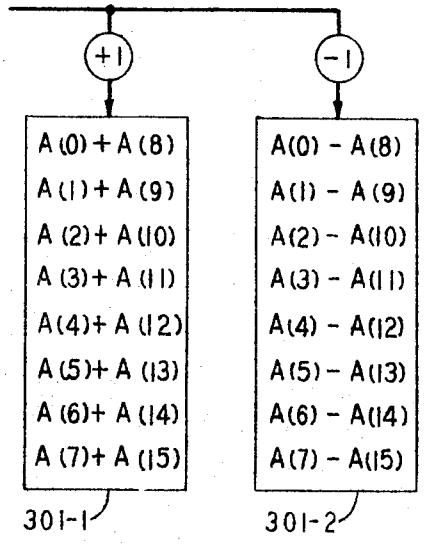


FIG. 6

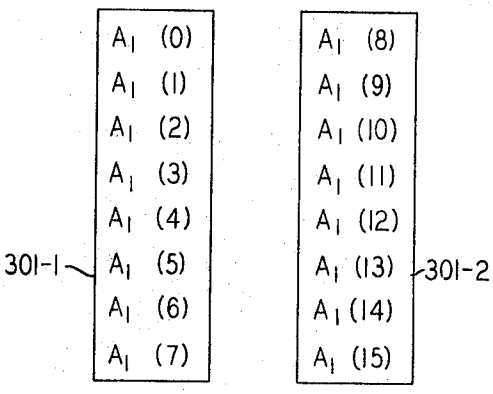


FIG. 7

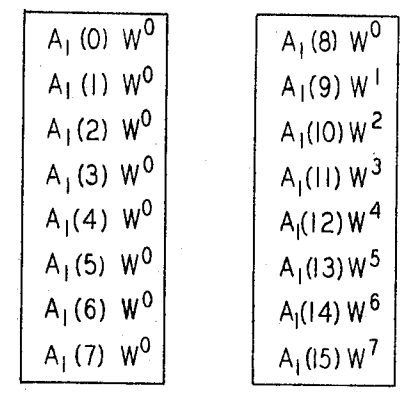


FIG. 8

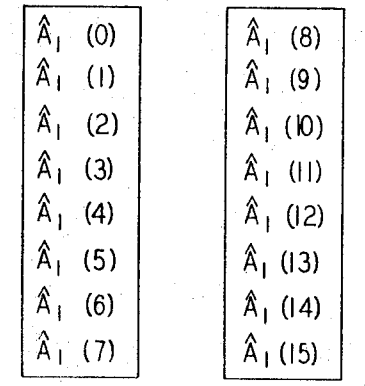


FIG. 9

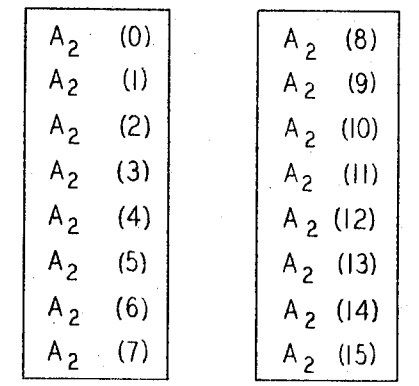


FIG. 10

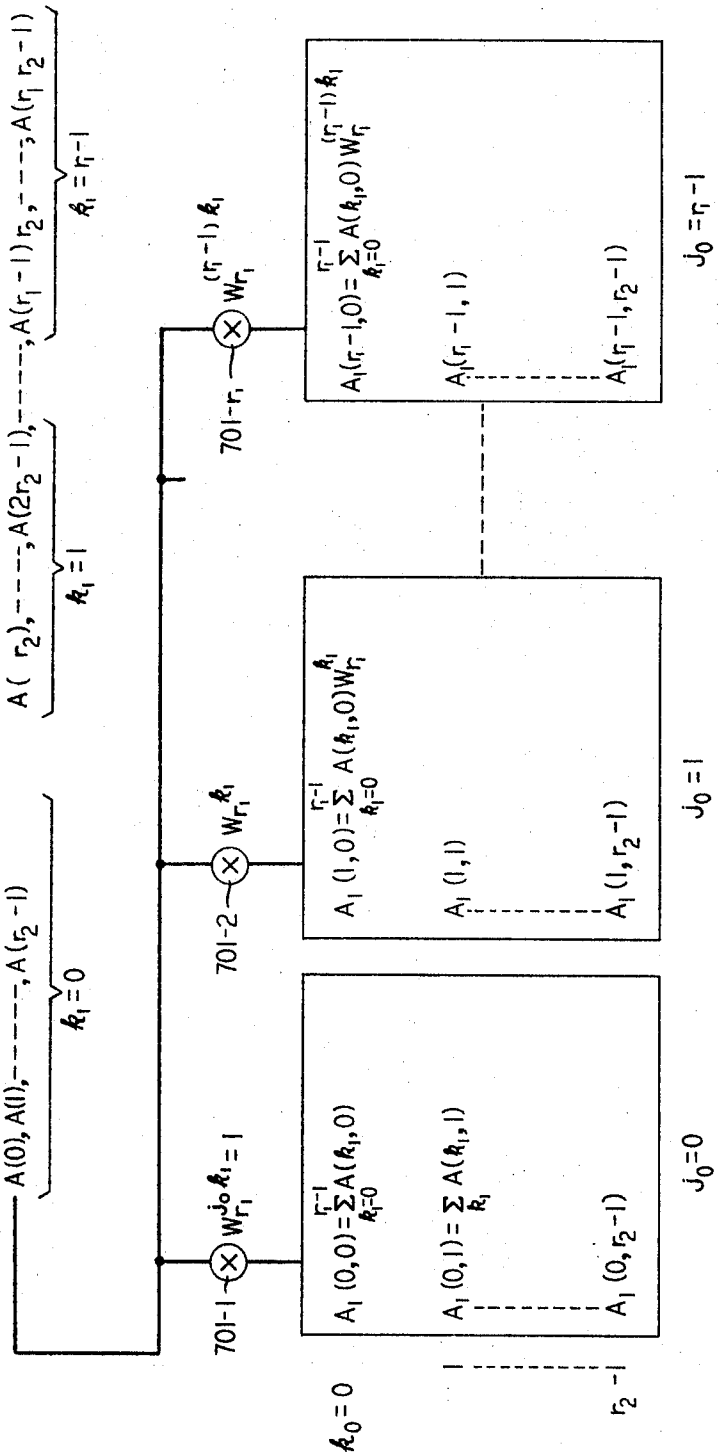

A(0), A(1), ----, A(r2 -1)
k1 = 0
A(r2), ----, A(2r2 -1), ----, A(r1 -1) r2, ----, A(r1 r2 -1)
k1 = 1
k1 = r1 -1
701-1
701-2
701-r1
W^j0 k1 r1 = 1
W r1^k1
W r1^(r1-1) k1
A1 (0,0) = Σ A(k1,0), k1=0 to r1-1
A1 (0,1) = Σ A(k1,1), k1
A1 (0, r2-1)
A1 (1,0) = Σ A(k1,0) W r1^k1, k1=0 to r1-1
A1 (1,1)
A1 (1, r2-1)
A1(r1 -1,0) = Σ A(k1,0) W r1^(r1-1) k1, k1=0 to r1-1
A1(r1 -1, 1)
A1(r1 -1, r2-1)
k0 = 0
1
r2 -1
j0 = 0
j0 = 1
j0 = r1 -1

FIG. 11

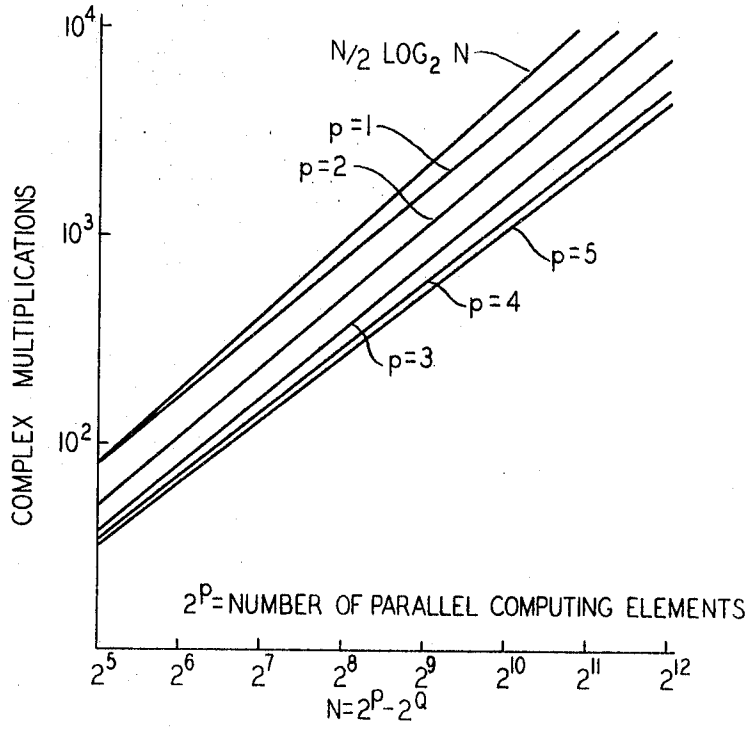

10^4
10^3
10^2
COMPLEX MULTIPLICATIONS
$N/2 \ LOG_2 \ N$
p=1
p=2
p=5
p=4
p=3
2^P=NUMBER OF PARALLEL COMPUTING ELEMENTS
2^5 2^6 2^7 2^8 2^9 2^{10} 2^{11} 2^{12}
$N=2^P-2^Q$

FIG. 12

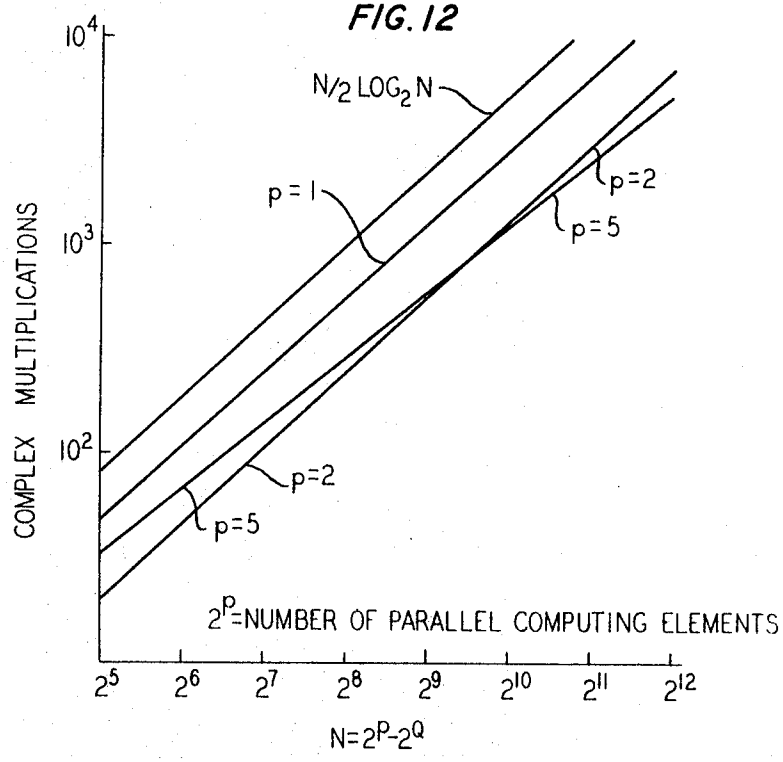
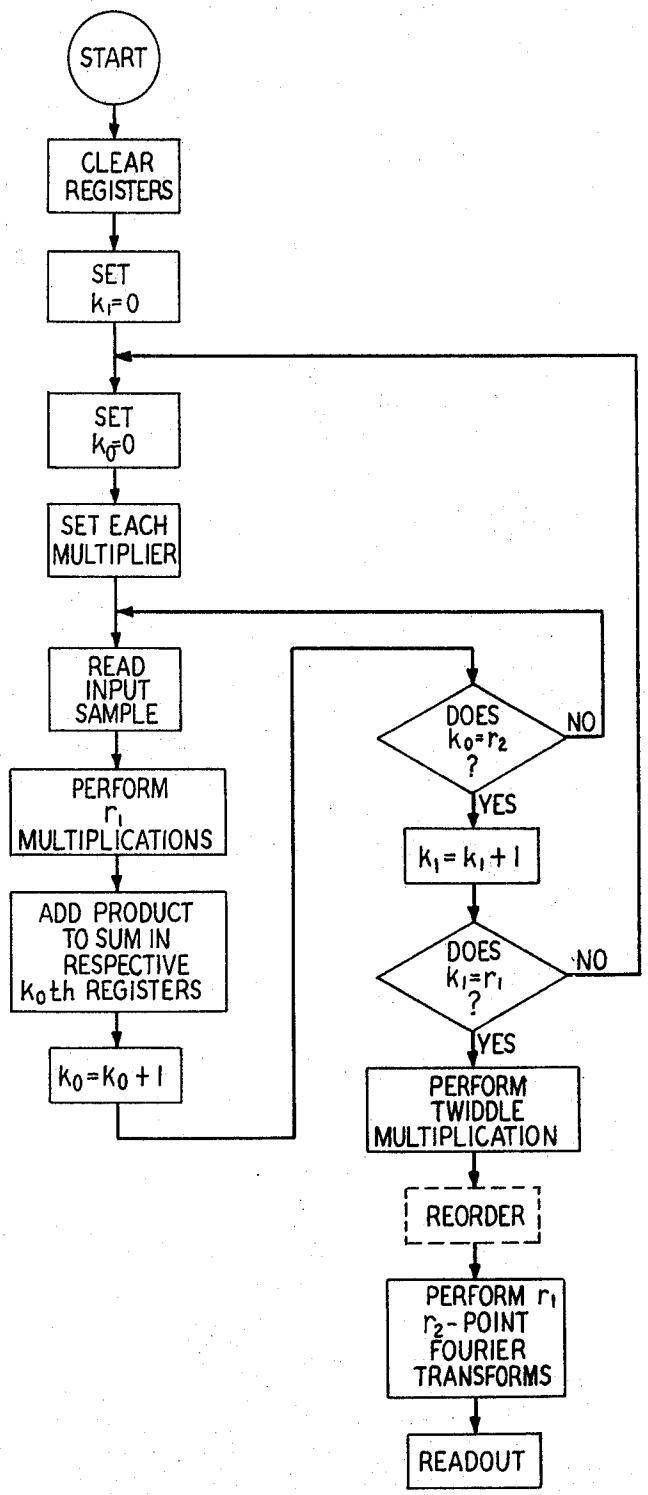

10^4
10^3
10^2
COMPLEX MULTIPLICATIONS
$N/2 \ LOG_2 \ N$
p=1
p=2
p=5
p=2
p=5
2^P=NUMBER OF PARALLEL COMPUTING ELEMENTS
2^5 2^6 2^7 2^8 2^9 2^{10} 2^{11} 2^{12}
$N=2^P-2^Q$

FIG. 13


START
CLEAR REGISTERS
SET $k_1=0$
SET $k_0=0$
SET EACH MULTIPLIER
READ INPUT SAMPLE
PERFORM r_1 MULTIPLICATIONS
ADD PRODUCT TO SUM IN RESPECTIVE k_0th REGISTERS
$k_0=k_0+1$
DOES $k_0=r_2$?
NO
YES
$k_1=k_1+1$
DOES $k_1=r_1$?
NO
YES
PERFORM TWIDDLE MULTIPLICATION
REORDER
PERFORM r_1 r_2-POINT FOURIER TRANSFORMS
READOUT

GLOBAL HIGHLY PARALLEL FAST FOURIER TRANSFORM PROCESSOR

This invention relates to methods and apparatus for signal processing. More particularly, this invention relates to methods and apparatus for the frequency analysis of data signals. Still more particularly, this invention relates to methods and apparatus for performing fast Fourier transforms using parallel processing techniques.

BACKGROUND OF THE INVENTION

Machine methods for frequency analysis and synthesis of signals using Fourier series and integral techniques have long been important areas of scientific and engineering investigation. In recent years there have been developed improved means and methods for performing such analyses and syntheses. Among these improved techniques are included those known collectively as fast Fourier transform (FFT) techniques. These FFT techniques originated in recent history with a paper titled "An Algorithm for the Machine Calculation of Complex Fourier Series," by J. W. Cooley and J. W. Tukey, Mathematics of Computation, Vol. 19, April 1965, pp. 297–301. The computational advantages demonstrated by this paper have spurred research in areas previously felt to be beyond economic feasibility. These advantages, often involving computational savings of time and machine complexity of an order of magnitude or more compared with classical techniques, flow largely from judicious groupings and reorganizations of summation techniques known in the prior art.

Numerous improvements and variations of the original FFT techniques have been developed since the publication of the original Cooley-Tukey paper, several of which were summarized in the June, 1967 and June, 1969 issues of the IEEE Transactions on Audio and Electroacoustics. Other important developments have been disclosed, for example, in U.S. patent applications by M. J. Gilmartin et al., Ser. No. 605,768, filed Dec. 29, 1966 now U.S. Pat. No. 3,517,173 issued June 23, 1970 and G. D. Bergland et al., Ser. No. 605,791, filed Dec. 29, 1966 now U.S. Pat. No. 3,544,775 issued Dec. 1, 1970. Another reference that should prove helpful in understanding the present invention in light of the prior art is one by R. Klahn and R. R. Shively, "FFT - Shortcut to Fourier Analysis," Electronics, Vol. 41, No. 8, Apr. 15, 1968, pp. 124–129.

Many data processing applications require identical operations on multiple sets of data. Most present digital computers accomplish this through the use of a single processor that operates on one data set at a time. When the number of data sets is large, however, even the fastest of these computers is too slow to perform its task in a reasonable amount of time. The need for more efficient "bulk" processing has contributed to recent interest in multiprocessing with highly parallel processors. See, for example, S. H. Unger, "A Computer Oriented Towards Spatial Problems," Proc. IRE, Vol. 46, Oct. 1958, pp. 1744–1750; J. H. Holland, "A Universal Computer Capable of Executing an Arbitrary Number of Subprograms Simultaneously," Proc. Eastern Joint Comp. Conf., Boston, Mass., Dec. 1–3, 1959, p. 108; J. Gregory, and R. McReynolds, "The SOLOMON Computer," IEEE Trans. on Electronic Computers, Vol. 12, Dec. 1963, pp. 774–781; W. T. Comfort, "Highly Parallel Machines," Proc. 1962 Workshop on Computer Organization, Spartan Books, Washington, D. C., 1963, p. 126.

Such machines obtain higher processing rates through the use of a large number of identical or similar processing units that operate simultaneously or in an overlapping mode, each on its own part of the overall task.

Other particularly useful parallel processing machines have been described in B. A. Crane et al., "Bulk Processing in Distributed Logic Memory," IEEE Trans. on Electronic Computers, Vol. 14, April 1965, pp. 186–196; J. A. Githens, "Highly-Parallel Calculating Arrays in Radar Data Processing," presented at the Workshop on the Development of New Computer Organizations, La Jolla, Calif., June 29, 1967; J. H. Huttenhoff and R. R. Shively, "Arithmetic Unit of Computing Element in a Global, Highly-Parallel Computer," IEEE Trans. on Electronic Computers, Vol. C–18, No. 8, August, 1969; and in B. A. Crane et al. U.S. Pat. Nos. 3,376,555 and 3,391,390 issued Apr. 2, 1968 and July 2, 1968, respectively.

BRIEF DESCRIPTION

The present invention recognizes and applies the principles of Fourier transform technology as implemented using selected aspects of the parallel data processing arts, with suitable extensions and modifications. In particular, means are provided in the present invention for decimating a sequence of input samples into a number of subsequences. Each of these subsequences is then processed in an independent computational element, thereby generating the Fourier transform for each subset taken alone. The parallel processing elements may take any one of several well-recognized forms, and each is under the control of a global control unit. Little or no communication or exchange of information is required between the several processing elements.

The results of the fourier transformation on the subsequences are then integrated in a subsequent processing step to generate the desired transform corresponding to the initial input sequence.

By suitably designing the arrangement of each processing element and by controlling the flow of data to and from these elements, it is possible to reduce the number of fundamental arithmetic operations, thereby increasing the speed of Fourier transform data processing. Additionally, the algorithm associated with one aspect of the present invention permits simplification of the format of the generated results, thereby simplifying interaction with the system user and other portions of a comprehensive data processing facility. In particular, by suitable intermediate processing, it is possible to sequentially read the results from each successive element of a parallel processing array while retaining the proper order of Fourier coefficients.

BRIEF DESCRIPTION OF THE DRAWINGS

The several aspects of the present invention will be more fully described below in connection with the several figures wherein:

FIG 1 shows a block diagram of a parallel data processing system suitable for performing fast Fourier transform analysis in accordance with the present invention;

FIG. 2 shows a computational element for use in the parallel processing array of FIG. 1;

FIGS. 3 through 9 show the flow of data and the storage patterns occurring during an FFT computation using the system of FIG. 1 for the special case where the number of processing elements is two;

FIG. 10 shows the formation of certain results for the general case where the number of computational elements is arbitrary;

FIG. 11 shows the number of complex multiplications required for various numbers of computational elements; and

FIG. 12 shows the number of complex multiplications required for the special cases where the number of computational elements is 1 and 2;

FIG. 13 is a flow chart corresponding to one embodiment of the present invention.

NOTATION AND NOMENCLATURE

The detailed description to follow are presented largely in terms of algorithms. These algorithmic descriptions are the means used by those skilled in the data processing art to most effectively convey the substance and meaning of their work to others skilled in the data processing arts.

An algorithm is here, and generally, conceived to be a sequence of steps leading to a desired result. These steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form

of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as samples, values, elements, terms, real-valued quantities, complex-valued quantities, number, or the like. It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Further, the manipulations performed are often referred to in terms, such as "adding," which are commonly associated with mental operations performed by a human being. This is not the sense in which terms such as "adding" are used here. No human operator is necessary (or desirable in most cases) in any of the operations described herein; the operations are machine operations.

Useful machines for performing part or all of the operations of the present invention include general purpose computers of the IBM 7090/94 class, various of the IBM System 360 class, the GE–600 class, or other similar machines. In all cases there should be borne in mind the distinction between the method operations in operating a computer and the (mathematical) method of computation itself. The present invention relates to method steps for operating a computer in processing electrical or other (e.g., mechanical, chemical) physical signals to generate other desired physical signals. The present invention also relates to apparatus for performing these operations.

The algorithms presented herein are not inherently related to any particular computer or other apparatus. In particular, various general purpose machines including those mentioned above, may be used, or it may prove more convenient to construct more specialized apparatus to perform the required method steps. The required structure for various of these machines will appear from the description given below in light of the state of presently existing knowledge in the field of parallel-processing computers.

DETAILED DESCRIPTION

Overview

FIG. **1** is a generalized block diagram of a parallel data processing system for use in computing Fourier coefficients. Shown there is a plurality of parallel processing elements **101–1** through **101–***q*, where *q* is an arbitrary positive integer. Also shown is an input source **110** for supplying sequences of N digital signals (derived by well-known sampling techniques from continuous signals, where appropriate) for which it is desired that Fourier coefficients be calculated. These input digital signals, which are typically representative of radar returns, speech or similar physical phenomena, are delivered to the processing elements **101–1** through **101–***q* (in a manner to be described below) by way of multipliers **120–1** through **120–***q*, respectively.

Also shown in FIG. **1** is an ensemble control unit **130** which exercises global control over the several processing elements **101–1** to **101–***q*. Control unit **130** is arranged to alter the sequence of operations in each of the processing elements by means of selecting or altering a stored or wired program in each of the processing elements or by any of several well-known techniques. Also shown in FIG. **1** is a conventional processing apparatus **140** which may, for example, be a general purpose data processing system having the usual arithmetic and logical capabilities represented by processor **145** and having provision for storing program data signals and other nonprogram data signals. The apparatus for storing these latter two classes of signals are designated by the identification numerals **150** and **160**, respectively. This separation of the various aspects of the processing apparatus **140** is for convenience of discussion only. In actual implementations data processing apparatus **140** may, for example, take the form of any of several of the IBM System 360/ class machines or other similar machines including G.E. 600-series machines.

Processing Elements

FIG. **2** shows a parallel processing element according to one embodiment of the present invention. Shown there is an arithmetic unit **210**, a memory **220** and a control unit **230**. The actual distribution of the circuitry to perform the required arithmetic operations will take any of several well recognized configuration depending, for example, on such things as the required size of memory **220**, the required speed of operation, and the number of parallel processing elements used. In particular, it is possible and may be desirable to integrate memory with the arithmetic and control aspects of a computational element as described, for example in B. A. Crane, et al., "Bulk Processing in Distributed Logic Memory," IEEE Transactions on Electronic Computers, Vol. 14, April 1965, pp. 186–196; and in the Githens reference, supra. The processing element may also conveniently take the form described in the Huttenhoff and Shively reference, supra. This latter reference is also illustrative of a suitable parallel processing environment for the present invention, and is especially helpful with regard to describing the control of the several processing units.

The arithmetic unit **210** may of course assume a more specialized form. In particular, it may be convenient to use the modularized equipment arrangements described in pending U.S. patent applications by G. D. Bergland et al., Ser. No. 605,791 and M. J. Gilmartin, Jr. et al., Ser. No. 605,768 both filed Dec. 29, 1966 and assigned to the assignee of the present application. Also, in appropriate cases the memory units **220** and **230** shown in FIG. **2** may be integrated with the arithmetic unit in the arrangement of Bergland et al., or in accordance with computational elements of the type described in the Crane et al. journal reference, supra. In any event, a further disclosure of the details of computational elements **101–***i*, $i = 1, 2 \ldots, q$, is not essential to a full and complete understanding of the present invention.

FFT Fundamentals

Arithmetic units Processing elements **101–***i*, $i = 1,2, \ldots, q$, may be arranged to perform any of the now well-known alternate versions of the fast Fourier transform. A useful description of the underlying principles of the FFT may be found in the Bergland et al. reference, supra. Only a brief description of the well-known aspects of the FFT will be given here.

By way of review, it is well to consider that the calculation of Fourier coefficients corresponding to a sequence of *N* input signals spaced over an interval of *T* seconds may be represented by

$$X(j) = \sum_{k=0}^{N-1} A(k)\, W^{jk},\quad j = 0,1, \ldots, N-1 \qquad (1)$$

where $W = e^{2\pi i/N}$ and $A(k)$ represents the input sample sequence $A(0), A(1), \ldots A(N-1)$.

The Cooley-Tukey algorithm reduces equation (1) to a recursive relation, the exact form of which depends on the properties of *N*. Starting with an initial list of numbers say b_0 (the input sequence), subsequent lists b_p are calculated, the final one of which comprises the required Fourier coefficients. When *N* is taken equal to 2^m, *m* lists will be calculated in accordance with one version of the algorithm. Each of the *N* entries in a list is identified by an *m*-digit binary integer. For example, one entry takes the form $b_p(j_0, h_1, \ldots, j_{p-1}, k_{m-p-1}, \ldots k_0)$ where each of the *j*'s and *k*'s are binary integers.

For the special case mentioned, $N = 2^m$, the recursive relation takes the form

$$\begin{aligned} &b_p(j_0, j_1, \ldots, j_{p-1}, k_{m-p-1}, \ldots, k_0) = \\ &b_{p-1}(j_0, j_1, \ldots, j_{p-2}, 0, k_{m-p-1}, \ldots k_0) \\ &+ b_{p-1}(j_0, j_1, \ldots, j_{p-2}, 1, k_{m-p-1}, \ldots k_0) \cdot \\ &e^{2\pi i/N (j_{p-1}2^{p-1} + j_{p-2}2^{p-2} + \ldots + j_0)2^{m-p}} \end{aligned} \qquad (2)$$

The binary number specified by the argument $j_0, j_1, \ldots, j_{m-p-1}, \ldots, k_0$ can be written simply as *k*, i.e., $b_p(j_0, j_1, \ldots, j_{p-1}, k_{m-p-1}, \ldots k_0 \equiv b_p(k)$.

As may be seen from equation (2), a highly regular pattern is followed in calculating the terms in the current list. For purposes of constructing the list b_1, the list b_0 is considered to be

portioned into two equal-length sublists or parts. Each term $b_1(k)$ in the list b_1 is derived from two terms from the list b_0. One of these terms from b_0 is multiplied by the complex phasor

$$W^{(j_{p-1}2^{p-1}+j_{p-2}2^{p-2}+\cdots+j_0)2^{m-p}}$$

where, as before, $W=e^{2\pi i/N}$. The first term of b_1 consists of the first term in $b_0$plus the first term in the second half of b_0 multiplied by

$$e^{\frac{i2\pi}{N}(0)2^{(m-1)}}=1+j0$$

Likewise the second term of b_1 is made up of the second term in b_0 plus the second term in second half of b_0 multiplied by the same complex value. This process is continued for all terms in the first half of i b_1. To obtain the terms in the second half of b_1, a similar scheme is followed except that the terms in the second half are multiplied by $e^{i\pi}=-1+j0$.

The Cooley-Tukey algorithm proceeds by transforming the list b_1 into a third list b_2, and so on. The same type of operations apply in each case, except that after each iteration the list is effectively partitioned into twice as many parts as before. Terms from selected parts of the then-current list are used to form the required terms for the list then being constructed.

The computational operations indicated in the various equations recited herein are each fundamental arithmetic operations which are performed using well-known apparatus. For example, adders are used to add, multipliers to multiply, and so forth. The generation of the complex exponential factors is accomplished using appropriate combinations of sine and cosine function values which may be stored for reference or which may be generated as they are required by well-known function generators.

Particularly simple and efficient generation of the required exponential factors may be effected using the techniques described in R. C. Singleton "On Computing the Fast Fourier Transform," Comm. ACM, Vol. 10, pp. 647–654, October 1967.

Special Case of Two Processing Elements

A simplified version of the present invention may be understood from a consideration of FIG. 3 and the discussion of it given below. This view shows a portion of a parallel processor in accordance with FIG. 1 generally, but being restricted to the case where $q = 2$. Thus, in FIG. 3 there are but two parallel processing elements of the form shown generally in FIG. 2 and described above. These are designated **301–1** and **301–2**, respectively.

The exemplary case to be considered is that where the input sequence $A(k)$ contains but 16 input samples, i.e., $N=16$. These samples are shown at the upper left portion of FIG. 3 as $A(0), A(1), \ldots, A(15)$, reading from right to left. Multipliers **320–1** and **320–2** are arranged for this simple example to multiply the input sequences applied to them by plus 1 or minus 1 (in a manner to be described below) before passing them on to their respective processing elements **301–1** and **301–2**.

The processing elements **301–1** and **301–2** are required to have only eight data storage locations designated 0 through 7. These are typically located in that portion of the computational element shown as memory **220** in FIG. 2. There may, of course, be other information storing portions of the computational elements corresponding to the arithmetic and control portions of a computational element e.g., those portions which are designated by the identification numerals **210** and **230** in FIG. 2. The computational elements shown in FIG. 3 are each arranged to calculate an eight-point Fourier transform.

In FIG. 3 (as in FIG. 1) the input sequence is shown as being broadcast to both (all) of the computational elements **301–1** and **301–2**. In accordance with the algorithm of the present invention for the case $N=16$, the first eight data elements in the sequence $A(k)$, that is, $A(0), A(1), \ldots, A(7)$, are weighted or multiplied by +1 and are stored in both of the computational elements as shown in FIG. 4. The remaining eight data signals $A(8), A(9), \ldots, A(15)$ are then weighted by +1and applied to element **301–1** and by −1 as they enter element **301–2**. These numbers are successively algebraically added to the previous contents of memory locations to form the sums shown in FIG. **5**.

It will be recognized that the contents stored in the corresponding locations of the computational elements **301–1** and **301–2** represent the (two-point) Fourier transform coefficients corresponding to the pairs of input data values $A(0)$ and $A(8)$, $A(1)$, and $A(9), \ldots, A(7)$ and $A(15)$.

Before proceeding to the completion of the desired 16-point transform, it is well to recall that the equations for one version of the FFT with $N=r_1r_2$, are given by

$$X(j_1, j_0)=\sum_{k_0=0}^{r_2-1}\left\{\left[\sum_{k_1=0}^{r_1-1}A(k_1, k_0)\,W_{r_1}^{j_0k_1}\right]W_{r_1r_2}^{j_0k_0}\right\}W_{r_2}^{j_1k_0}$$
$$=\sum_{k_0=0}^{r_2-1}\left\{A_1(j_0, k_0)\,W_{r_1r_2}^{j_0k_0}\right\}W_{r_2}^{j_1k_0} \qquad (3)$$

where $j=j_1r_1+j_0; j_1, k_0=0,1,\ldots,r_2-1; k=k_1r_2+k_0; j_0, k_1=0,1,\ldots,r_1-1; W_s=e^{2\pi i/s}$. In the present case $r_1=2$ and $r_2=8$. Thus we have

$$X(j_1, j_0)$$
$$=\sum_{k_0=0}^{7}\left\{\left[\sum_{k_1=0}^{1}A(k_1, k_0)\,(e^{2\pi i/2})^{j_0k_1}\right](e^{2\pi i/16})^{j_0k_0}\right\}(e^{2\pi i/8})^{j_1k_0} \qquad (4)$$

The expression in square brackets (for $k_0=0,1,\ldots,7$) is actually a sequence of eight two-point transforms and is precisely the result calculated by the method described above and stored as shown in FIG. **5**. These terms are those designated $A_1(j_n, k_0)$ in connection with equation (3) above. The contents of memory shown in FIG. **5** may therefore be represented in the manner shown in FIG. **6**. Equation (3) therefore reduces to

$$X(j_1, j_0)=\sum_{k_0=0}^{7}\left[A_1(j_0, k_0)\,(e^{2\pi i/16})^{j_0k_0}\right](e^{2\pi i/8})^{j_1k_0} \qquad (5)$$

It will be observed that the multiplication of the A_1 terms by the exponential factor $(e^{2\pi i/16})^{j_0k_0}$ to form the bracketed quality in equation (5) is merely a rereferencing of A_1 terms in the familiar manner associated with the fast Fourier transform.

This rereferencing is accomplished by well-known internal arithmetic operations by arithmetic unit **210** in FIG. 2 and the results are as shown in FIG. **7**. It proves convenient to relabel the rereferenced A_1 terms as $\hat{A}_1$ so that equation (3) becomes, for the present case

$$X(j_1, j_0)=\sum_{k_0=0}^{7}\hat{A}_1(j_0, k_0)\,(e^{2\pi i/8})^{j_1k_0} \qquad (6)$$

For convenience the $\hat{A}_1$ terms are referred to as shown in FIG. **8**. It will be observed that the operations indicated in equation (6) are precisely those of a conventional Fourier transform taken over the two eight-point sets of $\hat{A}_1$ terms stored in the computational elements as shown in FIG. **8**. These two eight-point Fourier transforms are performed identically in the two computational elements, with only the data sequences operated on being different. This transformation may profitably be performed using well-known FFT techniques. FIG. **9** shows the storage pattern of the results of the transformation of the contents stored in the memory portion of each of the computation elements shown in FIG. **8**. The trans-

formed quantities corresponding to the $\hat{A}_1$ terms are referred to as $A_2(j_0, j_1)$. It should be noted that $X(j_1, j_0) = A_2(j_0, j_1)$, or, since $j_0 = 0, 1$ and $j_1 = 0, 1, \ldots, 7$.

$$\begin{aligned}
X(0) &= A_2(0)\\
X(1) &= A_2(8)\\
X(2) &= A_2(1)\\
X(3) &= A_2(9)\\
X(4) &= A_2(2)\\
X(5) &= A_2(10)\\
X(6) &= A_2(3)\\
X(7) &= A_2(11)\\
X(8) &= A_2(4)\\
X(9) &= A_2(12)\\
X(10) &= A_2(5)\\
X(11) &= A_2(13)\\
X(12) &= A_2(6)\\
X(13) &= A_2(14)\\
X(14) &= A_2(7)\\
X(15) &= A_2(15)
\end{aligned}$$

It should be noted that if in-place reordering of the $\hat{A}_1$ terms in each processing element is accomplished according to well-known techniques prior to transformation, the X terms may be read out from the entire system shown in FIG. 1 (with $q = 2$) by alternately reading first from computational element **101–1** and then from computation element **101–2** while proceeding in sequence through the respective memories. Thus, the sometimes confusing and difficult-to-implement reordering of the Fourier coefficients is readily simplified by the present organization. If the reordering of $\hat{A}_1$ terms is not accomplished prior to transformation, the X terms will require the usual reordering.

Another useful feature of the present invention is that relating to the generation of the required exponential (W) signals required in forming, for example, the expressions shown in FIG. 7. Processing element **101–2** is shown to require eight different W terms, i.e., W^i, $i = 0, 1, \ldots 7$. By noting that for arbitrary m and n, $W^{m+n} = W^m W^n$, it is clear that only W^1 need be stored explicitly on computational element **101–2**. The remaining W terms may then be calculated by simple multiplication as required, e.g., W^2 may be calculated by multiplying W^1 by W^1, and so forth.

Alternately, the techniques described in Singleton, supra, may be used. Still another simplified variation of these exponential function generating techniques is one described by G. Sande at the IEEE Workshop of FFT processing at Arden House, Oct. 1968. There, Sande proposed calculating subsequent values from previous ones according to

$$(W^{K+1} - W^K)/W^K = W - 1 \text{ or } W^{K+1} = W^K(W-1) + W^K.$$

It is said that this particular rearrangement improves roundoff errors in computation. It is often helpful, especially for large values of K, to occasionally insert more accurate, separately calculated W values to prevent the accumulation of roundoff errors.

The Algorithm Generally

With the above notation and the description of the special case of a two-processor algorithm as a background, the general algorithm will now be presented. Briefly stated, the algorithm (illustrated in FIG. **10**) for $N = r_1 r_2$ comprises the steps of

1. Dividing the input sequence $\{A(k)\} = \{A(k_1, k_0)\} = \{A(K_1 r_2 + k_0)\}$ for $k = 0, 1, \ldots, N$, $k_1 = 0, 1, \ldots, (r_1 - 1)$, and $k_0 = 0, 1, \ldots, (r_2 - 1)$; into r_1 subsequences with k_1 fixed within each sequence. That is, let the first subsequent include only the first r_2 elements, the second subsequence only the next r_2 elements, and so forth.

2. Multiplying the terms of the k_1 th subsequence by $W_{r_1}^{j_0 k_1}$ $j_0 = 0, 1, \ldots, (r_1 = 1)$ to form sequences of product terms $\{A(k_1, k_0)\} W_{r_1}^{j_0 k}$, $j_0 = 0, 1, \ldots, (r_1 - 1)$ for each $k_1 = 0, 1, \ldots, (r_1 - 1)$. Here, $W_r = e^{2\pi i/r1}$. The multipliers **701–1** through **701–i** r_1 are conveniently adjusted for each new k_1.

3. Within initially cleared registers in each of r_1 processing elements, combining corresponding product terms formed in step (2) for successive k_1's to form

$$A_1(j_0, k_0) = \sum_{k_1=0}^{r_1-1} A(k_1, k_0)\, W_{r_1}^{j_0 k_1}$$

Only $N = r_1 r_2$ storage locations are required to form the A_1 terms because the previous contents of each location may be updated by appropriate product term formed in step (2), as k_1 varies over its range. That is, the A_1 terms may be summed as shown by accumulating partial sums.

4. Multiplying $A_1(j_0, k_0)$ terms by $W_{r_1 r_2}^{j_0 k_0}$ for each j_0 and k_0 to form corresponding referenced terms $\hat{A}_1(j_0, k_0)$ and replacing the A_1 terms by the corresponding $\hat{A}_1$ terms.

5. Forming the Fourier coefficients of the sets of $\hat{A}_1$ terms formed in step (4) and storing these coefficients in the locations previously occupied by the $\hat{A}_1$ terms. Again, any standard FET technique may be employed, or, non-FET technique may be used.

6. Reading the Fourier coefficients corresponding to the original input sequence by reading successively terms stored in each computational element starting with the first. In each case terms read from each computational element are conveniently read from successive storage locations. The first coefficient is read from the first location in the first computational element, the next from the first location in the second computational element, and so forth.

It should be understood that the Fourier transformation performed at step 5 is conveniently, though not necessarily, performed using a fast Fourier technique. This method is particularly attractive in the present arrangement because no communication between computational elements is then required in performing step (5). When an FFT process is employed in step (5) an in-place reordering according to well-known techniques is conveniently effected prior to transformation so that no part-transformation reordering is required prior to readout.

The above procedures insure that the Fourier coefficients read from memory in accordance with step **6** will be in ascending order, i.e., no reordering of the coefficients generated is required.

The fact that no communication is required between the various computational elements in performing step (5) of the algorithm and the fact that this step may be executed in parallel, greatly increases the number of operations that can be performed in a unit time interval.

The number of global complex multiplications M that are required for the case where $N = r_1 r_2 = 2^p 2^q$ is given by $M = q2^{2-1} + 2^q + 2^{p+q}$. It is this expression which must be minimized to obtain the minimum number of computations required. It can readily be shown that with the constraint that a 2^q-point transform is to be performed in each computational element that, in general, the number of computations required decreases with an increasing number of computational elements. As shown in FIG. **11**, however, the incremental decrease in the number of computations required decreases sharply as p increase. In particular, when $p = 1$ and $p = 2$ the $W_{r_1}^{j_0 k_1}$ terms assume only the values ± 1 and $\pm \sqrt{-1}$. In this special case only additions and conjugations (but no multiplications) are performed on the original input signals. Thus, the number of complex multiplications reduces to $q2^{q-1} + 2^q$ which is plotted in FIG. **12**. It is shown there that $p = 2$ (four computational elements) is a good choice which actually gives better reducer than $p = 5$ for up to 2^9 input data samples.

System Control

While the computational procedures required by the present invention are detailed above, precise circuit arrangements have not been included. Thus, for example, the particular logic circuitry required to gate the input samples to the appropriate registers has not been shown. Similarly, the details

of the multipliers (e.g., **120–1** to **120–q**, in FIG. **1**) have been omitted. This has not been by oversight, but rather by design.

Particular practitioners may wish to use preexisting facilities to carry out one or more of the fundamental operations involved or to coordinate one or more of these operations. The following discussion, taken together with the statement of the algorithm and other materials above, is sufficient to enable one to practice the instant invention in any of a number of particular configurations. It will be assumed in this discussion that program memory **150** in FIG. **1** is arranged to store a control program corresponding to the flow chart given in FIG. **13**. Processing unit **145** is arranged to execute this program with reference to data (including stored trigonometric values, where appropriate) stored in data memory **160** shown in FIG. **1**. Ensemble control unit **130** is then responsive to the program execution to actually close the appropriate switches, clear the appropriate registers, and so forth, that may appear in particular logic configurations. In keeping with the statement of the algorithm in the last section above, N will again be assumed to be equal to r_1r_2.

Referring to FIGS. **1** and **13** then, computer **140** causes all of the data storage registers in the processing elements **101–1** to **101–r_1** (recall $q=r_1$) here to be cleared or reset to zero. An initial value of zero is then set for k_1 and k_0. These values in turn dictate the initial exponential signals to be supplied to (or be selected in) multipliers **120–1** to **120–r_1**. These and later-required values may be stored in memory **160** or, as indicated parenthetically above, may be stored in a limited memory located at the multipliers. Alternately, the required multiplying factors may be generated as required according to one of the techniques given earlier.

When the required multiplier signals are at hand at the multipliers **120–1** to **120–r_1**, the input signals are read one at a time in serial form and presented to each of the multipliers. Each such signal is then multiplied by the multiplier signal and the result combined with the previous signals stored in the first register of the respective computational element. Because with $k_1=0$ the multiplier $W_r^{0\,1}$ is 1 for each j_0, the first input sample will be merely stored in the first register of each processing element. It will be recalled that these registers were initially cleared, so no prior nonzero sum remains to be combined with this input sample.

Next, k_0 is incremented and, since r_2 is assumed not to be equal to 1, another sample is read, multiplied by 1 and stored in the respective register in each processing element. This procedure is repeated until $k_0=r_2-1$, i.e., until the first subsequence of r_1 input terms has been read. Then k_1 is incremented, the multiplier values adjusted accordingly, and the next input sample value is read (with k_0 being reset to 0). Again the multiplications are performed by each of the multipliers on each input sample. Now, however, the multipliers are not all 1 as in the case of the first subsequence, i.e., when $k_1=0$. Thus, the multiplications are not degenerate and the respective products are added to the existing values stored in the first register of the respective processing elements.

This process is then repeated until the second subsequence of k_1 input values is read, multiplied, the products added to previous accumulated sums, and the new sums stored in appropriate registers in the processing elements. Then k_1 is incremented, the next k_1 elements are read and each of the partial sums augmented the products formed. When k_1 is equal to r_1-1, the sums indicated in FIG. **10** are complete.

Each of the accumulated sums in each of the processing element registers is then multiplied by the appropriate twiddle factor. This may be accomplished in a convenient manner by supplying the twiddle factors in the same manner as were the W_{r1} multipliers, and actually performing the multiplication in the multipliers **120-1** to **120–r_1**.

If an FFT is to be used in further processing the twiddled values, it may be convenient to perform a standard (digits reversed) reordering as mentioned earlier. This is shown (as a dotted block) in FIG. **13**. The r_2-point Fourier transformation of the contents of each of the r_1 processing elements is then performed.

The results are then ready to be read in order from the respective processing elements unless an FFT was used without preprocessing reordering. In this case, the reordering of the results in each processing element may be performed prior to or concurrent with readout. In any event, the desired results are stored in the processing elements in a predictable order.

While the above description has proceeded primarily in terms of a plurality of specially designed FFT processing elements (including circuitry of the type described in the Bergland et al. reference, supra, for example), it should be understood that these processing elements can conveniently take the form of general purpose data processors programmed to perform the required individual transformations and other indicated operations. Further, it is clear that the multipliers and processing elements described as separate elements may be suitably combined where appropriate.

What is claimed is:

1. Apparatus for generating data signals representing Fourier coefficients corresponding to a sequence of $N=r_1r_2$ input data signals, where r_1 and r_2 are positive integers with $r_1>2$, comprising

means for generating trigonometric function values,

first means for forming r_1 sets of r_2 intermediate coefficients, each intermediate coefficient corresponding to the sum of the products of selected ones of said input data signals with selected trigonometric function values, and

second means of simultaneously generating r_1 sets of output data signals representative of the Fourier coefficients for each respective set of said intermediate coefficients.

2. Apparatus according to claim **1** wherein said first means comprises means for s segmenting said sequence of input data signals into r_1 subsequences, each including r_2 data signals, and summing means for forming the sums of selected ones of said products.

3. Apparatus according to claim **2** wherein said summing means comprises r_1 sets of r_2 registers and means for adding a subsequent one of said products to a previously accumulated partial sum of said products, said registers being initially cleared.

4. Apparatus according to claim **3** wherein said first means further comprises means for forming the product of said sums and corresponding trigonometric function values, thereby to form rereferenced sums.

5. Apparatus according to claim **1** wherein said second means comprises r_1 fast Fourier transform processors for operating on each of said r_1 sets of intermediate coefficients.

6. Apparatus for generating data signals representing Fourier coefficients corresponding to a sequence of $N=r_1r_2$ input data signals comprising

A. a source of trigonometric signals,

B. a plurality of multipliers for forming product signals representing the product of selected ones of said input data signals with selected trigonometric signals,

C. a plurality of processing elements, each including

1. means for forming sum-of-product signals representing accumulated sums of selected ones of said product signals,

2. means for forming signals representing the product of each of said sum-of-product signals and a corresponding trigonometric signal to form rereferenced signals,

D. means for forming signals representing the Fourier coefficients corresponding to said rereferenced signals.

7. Apparatus according to claim **6** further including in each of said processing elements means for reordering said rereferenced signals.

8. The machine method for generating signals representing the Fourier coefficients corresponding to a set of $N=r_1r_2$ ordered input signals, r_1 and r_2 being positive integers with $r_1>2$ comprising the steps of

A. generating r_1 sets of r_2 intermediate signals representing the sum of products of said input samples with selected trigonometric function signals, and

B. generating simultaneously a set of signals representing

the Fourier coefficients corresponding to each of said sets of intermediate signals.

9. The method of claim 8 further comprising the step of multiplicity said intermediate signals by a rereferencing trigonometric function signal prior to performing step (B).

10. The method of claim 9 wherein said step (B) comprises the step of performing a fast Fourier transformation based on said intermediate signals.

11. The method of claim 10 further comprising the step of reordering the signals formed in accordance with the process of claim 9 prior to performing said fast Fourier transformation.

12. In a digital processor, the machine method of generating signals representing Fourier coefficients corresponding to a sequence of $N = r_1 r_2$ input signals $A(k) = A(k_1 r_2 + k_0)$, $k_1 = 0,1,\ldots,r_1-1$; $k_0 = 0,1,\ldots,r_2-1$, comprising the steps of

1. generating first product signals by multiplying each $A(k)$ by $(e^{2\pi i/r}1)^{0\,1}$ for $j_0 = 0,1,\ldots,r_1-1$,
2. forming sum signals by adding product signals formed in step 1 which correspond to the same value of j_0 and k_0,
3. forming second product signals by multiplying said sum signals by $(e^{2\pi i/N})^{0\,0}$ for corresponding values of j_0 and k_0, and
4. generating r_1 sets of r_2 Fourier coefficients, each set corresponding to that set of r_2 of said second product signals corresponding to a fixed value of j_0.

13. The method of claim 12 wherein step (1) comprises the steps of

A. reading an input signal,

B. reading r_1 stored values of $(e^{2\pi i/r}1)^{0\,1}$ for the input value read at step (A), for $j_0 = 0,1,\ldots,r_1-1$,

C. generating a signal corresponding to the product of the signal read at step (A) with each of the signals read at step (B),

D. repeating steps (A), (B), and (C) of each input signal.

14. The method of claim 12 wherein step (1) comprises the steps of

A. reading an input signal

B. generating values of $(e^{2\pi i/r}1)^{0\,1}$ for the input value read at step (A) for $j_0 = 0,1,\ldots,r_1-1$,

C. generating a signal corresponding to the product of the signal read at step (A) with each of the signals generated at step (B),

D. repeating steps (A), (B), and (C) for each input signal.

15. The method of claim 12 wherein step (2) comprises the steps of

A. clearing each of N registers $R(j_0, k_0)$, $j_0 = 0,1,\ldots,r_1-1$, $k_0 = 0,1,\ldots,r_2-1$,

B. adding each of said first product signals as it is formed to the contents of the register having corresponding values of j_0 and k_0.

16. The method of claim 12 wherein each of said sets of r_2 Fourier coefficients are generated in parallel at substantially the same time.

17. The method of claim 16 wherein at step (4) each of said sets of r_2 coefficients are generated by performing a fast Fourier transform based on those r_2 second product signals corresponding to a fixed value of j_0.

* * * * *

UNITED STATES PATENT OFFICE

CERTIFICATE OF CORRECTION

Patent No. 3,662,161 Dated May 9, 1972

Inventor(s) Glenn D. Bergland; Donald E. Wilson

It is certified that error appears in the above-identified patent and that said Letters Patent are hereby corrected as shown below:

Column 2, line 1, after "of" insert --a--; line 22, change "fourier" to --Fourier--; and line 67, change "description" to --descriptions--. Column 4, line 62, change "$b_p(j_0,h_1,...$" to --$b_p(j_0,j_1,...$--. Column 5, line 16, change "i b_1" to --b_1--. Column 6, Eq. (4), after "$\}$ " insert --$\cdot\cdot\cdot$--; and line 39, change "$A_1(j_{ay},k_0)$" to --$A_1(j_0, k_0)$--. Column 7, line 66, change "$\left\{A(K_1r_2 + k_0)\right\}$" to --$\left\{A(k_1r_2 + k_0)\right\}$--; line 71, change "$W_{r1}{}^{j0k1}$" to --$W_{r_1}{}^{j_0k_1}$--; line 73 change "$k_0))W_{r_1}{}^{j0k}$" to --$k_0)W_{r_1}{}^{j_0k_i}\}$--; line 74, change "$W_r = e^{2}\pi i/r1$" to --$W_{r_1} = e^{2\pi i/r_1}$--; line 74 change "701-i" to --701-r_1--; and delete "r_1". Column 8, line 1, change "Within" to --With--; line 12, after "by" insert --the--; and line 21, change "FET" to --FFT-- and change "non-FET" to --non-FFT--. Column 9, line 24, change "(recall q = r_1) here" to --(recall q = r_1 here)--; line 39, change "$W_r{}^{0\ 1}$" to --$W_{r_1}{}^{j_0k_1}$--; and line 69, change "W_{r1}" to --W_{r_1}--. Column 10, line 29, change "means of" to --means for--; line 33, after "for" delete --s--; and line 76, change "generating simultaneously" to --simultaneously generating--. Column 11, line 4, change "multiplicity" to --multiplying--;

USCOMM-DC 60376-P69
* U.S. GOVERNMENT PRINTING OFFICE : 1969 O—366-334

CERTIFICATE OF CORRECTION - 2

Patent No. 3,662,161 Dated May 9, 1972

Inventors Glenn D. Bergland; Donald E. Wilson

line 18, change "$(e^{2\ i/r\ 1})^{\ 0\ 1}$" to --$(e^{2\pi i/r_1})^{j0k_1}$--;

line 22, change "$(e^{2\ i/N})^{\ 0\ 0}$" to --$(e^{2\pi i/N})^{j0k_0}$--; and

line 31, change "$(e^{2\ i/r_1})^{\ 0\ 1}$" to --$(e^{2\pi i/r_1})^{j0k_1}$--.

Column 12, line 4, change "of" to --for--; and line 9, change "$(e^{2\ i/r_1})^{\ 0\ 1}$" to --$(e^{2\pi i/r_1})^{j0k_1}$--.

Signed and sealed this 12th day of December 1972.

(SEAL)
Attest:

EDWARD M.FLETCHER,JR.
Attesting Officer

ROBERT GOTTSCHALK
Commissioner of Patents