Two Families of Radix-2 FFT Algorithms With Ordered Input and Output Data

Pere Marti-Puig

***Abstract*—Two radix-2 families of fast Fourier transform (FFT) algorithms that have the property that both inputs and outputs are addressed in natural order are derived in this letter. The algorithms obtained have the same complexity that Cooley–Tukey radix-2 algorithms but avoid the bit-reversal ordering applied to the input. These algorithms can be thought as a variation of the radix-2 Cooley–Tukey ones.**

***Index Terms*—Fast algorithms, fast Fourier transform (FFT).**

I. Introduction

THE discrete fast Fourier transform was first discovered by Gauss [1] and rediscovered by Cooley and Tukey [2] in the 1960s. Due to its importance in engineering, many algorithms have been derived since the 1960s, and there is an extensive bibliography concerning this field. There are algorithms referred to as higher radix [3], [4], mixed-radix [5], prime-factor [6], [19], Winograd [7], split-radix [8], [9], [20], [21], identical geometry from stage-to-stage FFT [12], recursive [10], combination of decimation-in-time and the decimation-in-frequency [11], among many variants. Reference [13] provides an interesting overview on the state of the art of FFT. Matrix representations for FFT and some tendencies in the field of fast discrete signal transforms are found in [14]–[18]. One of the interests in FFT research algorithms is to reduce its arithmetic complexity by minimizing the total number of real multiplications and additions as has been done recently in [21], although actual performance of FFT on computers is determined by many other factors such as cache or central processing unit pipeline optimization. In practice, many FFT algorithms need some input or output data permutation, bit-reversal ordering being the one that most frequently appears. Although bit-reversal ordering seems simple, some recent improvements can be found in [22]–[27]. In [28], there are two flow graphs for eight point FFT sorted algorithms according to Stockham. In [28], the authors change the Cooley–Tukey flow graphs to obtain the other algorithms presented in it by applying flow graph transform rules; it is a method which is easy only in some cases and above all when the flow graph has reduced dimensions. The Stockham algorithms are not derived in [28] in which a unique reference links to a private communication. In [30], input-output FFT ordered algorithms are derived from the Cooley–Tokey factorizations inserting a—different—permutation matrix between factors. In [29], they were proposed to be used in vector processors. Reference [30] compiles the actualized know-how about FFTs using a common matrix-vector notation under a computational framework. In this letter, two recursive properties are presented involving matrix $\mathbf{F}_{\mathrm{N}}$ and $\mathbf{F}_{\mathrm{N}/2}$ not reported before. By iterating them and with some algebra, the sorted radix-2 algorithms are easily obtained. These properties can be extended to obtain higher radix factorizations or combined with other recursive properties.

Manuscript received April 24, 2008; revised June 26, 2008. Current version published January 08, 2009. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Per Lowenborg.

The author is with the Department of Digital Information Technologies, University of Vic, Vic (Barcelona) 08500, Spain (e-mail: pere.marti@uvic.cat).

Digital Object Identifier 10.1109/LSP.2008.2003993

II. Used Notation

In this section, we briefly present the used notation. Since we always deal with square matrices in what follows, an $\mathrm{N}\times\mathrm{N}$ square matrix is denoted by a bold capital letter with subscript N. In this letter, the number N is a power of two. The elements of matrix $\mathbf{A}_{\mathrm{N}}$ positioned at the row m and the column n are denoted by a_{mn}. Sometimes we will use the notation $\mathbf{A}_{\mathrm{N}}=\{a_{mn}\}$. A column vector is represented by a bold small letter and, since its length can always be known from the context in this letter, its subscript indicates the position of the column in a matrix. The $\mathrm{N}\times\mathrm{N}$ identity matrix is denoted by $\mathbf{I}_{\mathrm{N}}$, and it can be written by its column vectors $\mathbf{e}_{\mathrm{i}}$ as $\mathbf{I}_{\mathrm{N}}=[\mathbf{e}_1\mathbf{e}_2\cdots\mathbf{e}_{\mathrm{n}}]$. With $\mathbf{0}_{\mathrm{N}}$, we denote the $\mathrm{N}\times\mathrm{N}$ zero matrix. An even-odd permutation matrix $\mathbf{P}_{\mathrm{N}}$ in terms of vectors $\mathbf{e}_{\mathrm{i}}$ takes the form $\mathbf{P}_{\mathrm{N}}=[\mathbf{e}_1\mathbf{e}_3\cdots\mathbf{e}_{\mathrm{n}-1}\mathbf{e}_2\mathbf{e}_4\cdots\mathbf{e}_{\mathrm{n}}]$. We will sometimes find useful to divide a given matrix into submatrices. The symbol $\otimes$ stands for the right Kronecker product and, for arbitrary square matrices $\mathbf{A}_{\mathrm{M}}$ and $\mathbf{B}_{\mathrm{N}}$, the Kronecker product $\mathbf{A}_{\mathrm{M}}\otimes\mathbf{B}_{\mathrm{N}}$ is an $\mathrm{MN}\times\mathrm{MN}$ matrix that can be written using the elements a_{mn} of matrix $\mathbf{A}_{\mathrm{M}}$ as

$$\mathbf{A}_M\otimes\mathbf{B}_N=\begin{bmatrix} a_{11}\mathbf{B}_N & \cdots & a_{1M}\mathbf{B}_N \\ \cdots & & \cdots \\ a_{M1}\mathbf{B}_N & \cdots & a_{MM}\mathbf{B}_N \end{bmatrix}. \tag{1}$$

Most of the times, we will use the Kronecker product to show a particular matrix structure. Next, we recall some useful properties involving the Kronecker product. We have

$$\mathbf{I}_{2^{n1}}\otimes\mathbf{I}_{2^{n2}}=\mathbf{I}_{2^{n1+n2}} \tag{2}$$

$$(\mathbf{A}_M\otimes\mathbf{B}_N)(\mathbf{C}_M\otimes\mathbf{D}_N)=\mathbf{A}_M\mathbf{C}_M\otimes\mathbf{B}_N\mathbf{D}_N. \tag{3}$$

Note that superscript n in a matrix means the power n of this matrix. Finally, the factorization of an arbitrary matrix $\mathbf{M}_{\mathrm{N}}$ in terms of n factors (or stages) $\mathbf{E}_{\mathrm{N}}(\mathrm{i})$ is written as follows:

$$\mathbf{M}_N = \prod_{i=1}^{n} \mathbf{E}_N(i) = \mathbf{E}_N(n) \cdots \mathbf{E}_N(2)\mathbf{E}_N(1). \tag{4}$$

III. Problem Formulation and Derivation of Algorithms

Consider $N = 2^n$ and j the square root of -1. The Fourier transform matrix $\mathbf{F}_N$ is defined as

$$\mathbf{F}_N = \left\{ e^{-j\frac{2\pi}{N}pq} \right\} \quad p, q = 0 : N-1. \tag{5}$$

Being $\mathbf{x} = [x(1) \cdots x(\mathrm{N})]^{\mathrm{T}}$ the ordered input vector, the ordered transformed vector $\mathbf{y} = [X(1) \cdots X(\mathrm{N})]^{\mathrm{T}}$ is obtained by performing the operation $\mathbf{y} = \mathbf{F}_{\mathrm{N}}\mathbf{x}$. As we know, a fast algorithm can be thought as a sparse factorization of the transform matrix in which the new organization of operations reduces the complexity of the direct full matrix vector multiplication problem (of order N^2) drastically (to order $\mathrm{NLog_2N}$). However, as a result of computing the FFT in terms of sparse factors, the output vector appears disordered. In most applications, the order must be re-established by performing a permutation of the elements. The most common permutation appeared in FFT algorithms is the bit-reversal one which is simple to perform by hardware. In some algorithms, the permutation is applied at the input, and in others, it is applied at the output. Our goal is to obtain factorizations that avoid the reordering operation. To do this, we introduce the next two recursion properties involving matrix $\mathbf{F}_{\mathrm{N}}$ and matrix $\mathbf{F}_{\mathrm{N}/2}$.

Let $\mathbf{B}_{2^i}$ denote the matrix defined by

$$\mathbf{B}_{2^n} = \begin{bmatrix} \mathbf{I}_{2^{n-1}} & \mathbf{A}_{2^{n-1}} \\ \mathbf{I}_{2^{n-1}} & -\mathbf{A}_{2^{n-1}} \end{bmatrix} \tag{6}$$

where

$$\mathbf{A}_N = diag\left\{ e^{-j\frac{2\pi}{N}p} \right\} \quad p = 0 : N-1 \tag{7}$$

is a diagonal matrix.

Matrix $\mathbf{B}_{2^n}$ can also be written as

$$\mathbf{B}_{2^n} = \mathbf{H}_{2^n} \begin{bmatrix} \mathbf{I}_{2^{n-1}} & \mathbf{0}_{2^{n-1}} \\ \mathbf{0}_{2^{n-1}} & \mathbf{A}_{2^{n-1}} \end{bmatrix} \tag{8}$$

being

$$\mathbf{H}_{2^n} = \begin{bmatrix} \mathbf{I}_{2^{n-1}} & \mathbf{I}_{2^{n-1}} \\ \mathbf{I}_{2^{n-1}} & -\mathbf{I}_{2^{n-1}} \end{bmatrix}. \tag{9}$$

Then, we have

$$\mathbf{F}_{2^n} = \mathbf{B}_{2^n}\mathbf{P}_{2^n}(\mathbf{F}_{2^{n-1}} \otimes \mathbf{I}_2) \tag{10}$$

$$\mathbf{F}_{2^n} = (\mathbf{F}_{2^{n-1}} \otimes \mathbf{I}_2)\mathbf{P}_{2^n}^T\mathbf{B}_{2^n}^T. \tag{11}$$

We will obtain two sets of solutions, one from (10) and another from (11).

A. First Set of Solutions

The first set of factorizations comes from expression (10) taken into account that $\mathbf{F}_{2^{n-1}}$ can be written as

$$\mathbf{F}_{2^{n-1}} = \mathbf{B}_{2^{n-1}}\mathbf{P}_{2^{n-1}}(\mathbf{F}_{2^{n-2}} \otimes \mathbf{I}_2) \tag{12}$$

then, by means of (10) and (12), we can write $\mathbf{F}_{2^n}$ as

$$\mathbf{F}_{2^n} = \mathbf{B}_{2^n}\mathbf{P}_{2^n}\left(\mathbf{B}_{2^{n-1}}\mathbf{P}_{2^{n-1}}(\mathbf{F}_{2^{n-2}} \otimes \mathbf{I}_2) \otimes \mathbf{I}_2\right). \tag{13}$$

Using property (3), the next factor from (13) can be reorganized in the following way:

$$\mathbf{B}_{2^{n-1}}\mathbf{P}_{2^{n-1}}(\mathbf{F}_{2^{n-2}} \otimes \mathbf{I}_2) \otimes \mathbf{I}_2 = (\mathbf{B}_{2^{n-1}}\mathbf{P}_{2^{n-1}} \otimes \mathbf{I}_2)(\mathbf{F}_{2^{n-2}} \otimes \mathbf{I}_2 \otimes \mathbf{I}_2) \tag{14}$$

and, from property (2), being, $\mathbf{I}_2 \otimes \mathbf{I}_2 = \mathbf{I}_{2^2}$, $\mathbf{F}_{2^n}$ becomes

$$\mathbf{F}_{2^n} = \mathbf{B}_{2^n}\mathbf{P}_{2^n}(\mathbf{B}_{2^{n-1}}\mathbf{P}_{2^{n-1}} \otimes \mathbf{I}_2)(\mathbf{F}_{2^{n-2}} \otimes \mathbf{I}_{2^2}). \tag{15}$$

The same process could be repeated introducing again expression (10), particularized now for $\mathbf{F}_{2^{n-2}}$, in (15). By the repetition of the same operations, the factorization finishes when $\mathbf{F}_2$ is reached. The full factorization takes the form

$$\mathbf{F}_{2^n} = \prod_{i=1}^{n} (\mathbf{B}_{2^i}\mathbf{P}_{2^i} \otimes \mathbf{I}_{2^{n-i}}) \tag{16}$$

with factors (stages) taking the form

$$\mathbf{B}_{2^i}\mathbf{P}_{2^i} \otimes \mathbf{I}_{2^{n-i}} = \mathbf{H}_{2^i} \begin{bmatrix} \mathbf{I}_{2^{n-1}} & \mathbf{0}_{2^{n-1}} \\ \mathbf{0}_{2^{n-1}} & \mathbf{A}_{2^{n-1}} \end{bmatrix} \mathbf{P}_{2^i} \otimes \mathbf{I}_{2^{n-i}}. \tag{17}$$

Note that in (16), the results are presented in an ordered form and no permutation matrices appear at the beginning or at the end of the factor chain.

1) Example: As an example, let us consider $\mathrm{N} = 8$ $(n = 3)$; then, from (16), we have

$$\mathbf{F}_{2^3} = \prod_{i=1}^{3} (\mathbf{B}_{2^i}\mathbf{P}_{2^i} \otimes \mathbf{I}_{2^{n-i}}) = \mathbf{B}_{2^3}\mathbf{P}_{2^3}(\mathbf{B}_{2^2}\mathbf{P}_{2^2} \otimes \mathbf{I}_2)(\mathbf{B}_{2^1}\mathbf{P}_{2^1} \otimes \mathbf{I}_{2^2}) \tag{18}$$

where

$$\mathbf{B}_{2^1}\mathbf{P}_{2^1}\otimes\mathbf{I}_{2^2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$\mathbf{B}_{2^2}\mathbf{P}_{2^2}\otimes\mathbf{I}_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -j & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -j \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & j & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & j \end{bmatrix}.$$

and

$$\mathbf{B}_{2^3}\mathbf{P}_{2^3} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & e^{-j\frac{\pi}{4}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -j & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & e^{-j\frac{3\pi}{4}} \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -e^{-j\frac{\pi}{4}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & j & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -e^{-j\frac{3\pi}{4}} \end{bmatrix}.$$

Note that all three sparse matrices in the example compute two outputs from only two input elements. This basic operation is called (radix-2) butterfly, and it is represented graphically in Fig. 1.

Note also in the same matrices that the indices m, n of the nonzero elements give the interconnection pattern among inputs and outputs stage-to-stage (or factor to factor) seeing that the n input is needed to calculate the m output. Fig. 2 stands for the interconnection pattern of the example in terms of butterflies. Fig. 3, for the same example, shows the signal flow diagram in which the operations are explicitly indicated in order to realize that such factorizations could have an efficient implementation also in terms of multiplication operations.

B. Second Set of Solutions

The second set of factorizations that produces results in an ordered form comes from expression (11). Matrix $\mathbf{F}_{2^{n-1}}$ can be written as

$$\mathbf{F}_{2^{n-1}} = (\mathbf{F}_{2^{n-2}}\otimes\mathbf{I}_2)\mathbf{P}^T_{2^{n-1}}\mathbf{B}^T_{2^{n-1}} \tag{19}$$

Fig. 1. Radix-2 butterfly representation showing dependence between two inputs and two outputs.

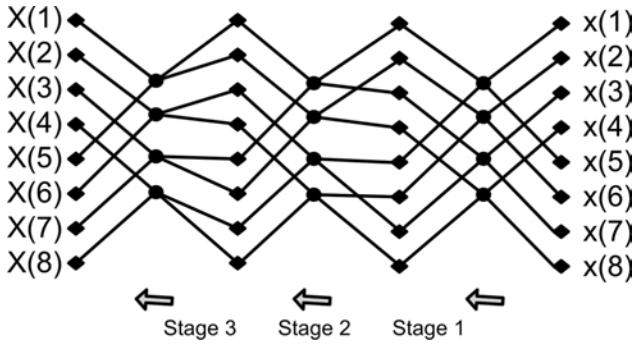

Fig. 2. Interconnection pattern representation for the first factorization in terms of butterflies given $\mathrm{N} = 8$. Inputs and outputs are addressed in natural order.

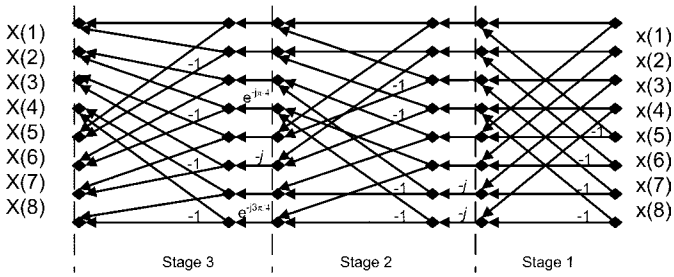

Fig. 3. Signal flow diagram for the same factorization as Fig. 2 $(\mathrm{N} = 8)$. Inputs and outputs are addressed in natural order. The transmittance gain of the arrows with multiplicative factor different than 1 are represented in bold.

And, in the same way as in the previous section, introducing (19) in (11), we have

$$\mathbf{F}_{2^n} = \left((\mathbf{F}_{2^{n-2}}\otimes\mathbf{I}_2)\mathbf{P}^T_{2^{n-1}}\mathbf{B}^T_{2^{n-1}}\otimes\mathbf{I}_2\right)\mathbf{P}^T_{2^n}\mathbf{B}^T_{2^n}. \tag{20}$$

Then, using also the properties (3) and (2), the factor of the right-hand side of (20) can be reorganized as follows:

$$\begin{aligned}(\mathbf{F}_{2^{n-2}}\otimes\mathbf{I}_2)\mathbf{P}^T_{2^{n-1}}\mathbf{B}^T_{2^{n-1}}\otimes\mathbf{I}_2 \\ = (\mathbf{F}_{2^{n-2}}\otimes\mathbf{I}_{2^4})\left(\mathbf{P}^T_{2^{n-1}}\mathbf{B}^T_{2^{n-1}}\otimes\mathbf{I}_2\right)\end{aligned} \tag{21}$$

to obtain

$$\mathbf{F}_{2^n} = (\mathbf{F}_{2^{n-2}}\otimes\mathbf{I}_{2^4})\left(\mathbf{P}^T_{2^{n-1}}\mathbf{B}^T_{2^{n-1}}\otimes\mathbf{I}_2\right)\mathbf{P}^T_{2^n}\mathbf{B}^T_{2^n}. \tag{22}$$

The same steps can be repeated now for $\mathbf{F}_{2^{n-2}}$ using (11). By iterating the process, the factorization will be finished when matrix $\mathbf{F}_2$ is reached. The full factorization of the second solution takes the form

$$\mathbf{F}_{2^n} = \prod_{i=1}^{n}\left(\mathbf{P}^T_{2^{n-i+1}}\mathbf{B}^T_{2^{n-i+1}}\otimes\mathbf{I}_{2^{i-1}}\right). \tag{23}$$

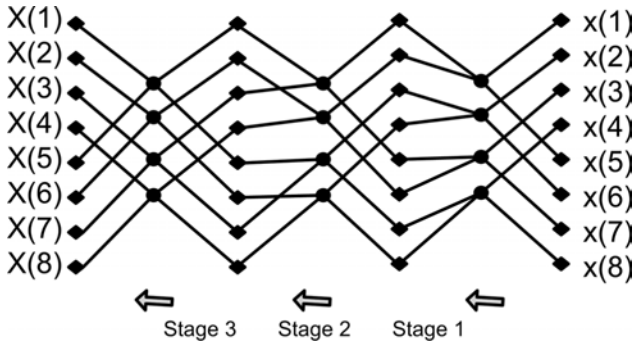

Fig. 4. Interconnection pattern representation for the second factorization in terms of butterflies given $\mathrm{N} = 8$. Inputs and outputs are addressed in natural order.

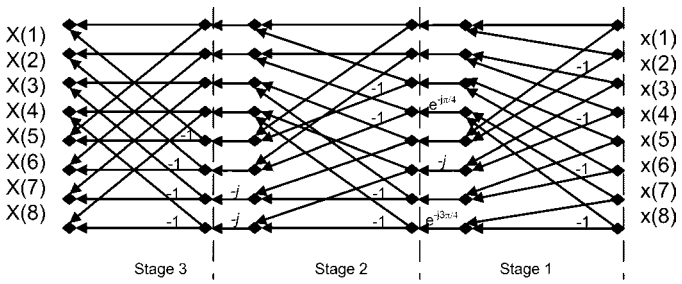

Fig. 5. Signal flow diagram for the second factorization $(\mathrm{N} = 8)$ corresponding with Fig. 4. Inputs and outputs are addressed in natural order. The transmittance gain of the arrows with multiplicative factor different than 1 are represented in bold.

Fig. 4 shows the interconnection pattern and Fig. 5 the signal flow diagram of this second solution given $\mathrm{N} = 8$.

IV. Conclusions

Two radix-2 families of factorizations that have the property that both inputs and outputs are addressed in natural order are derived in this letter. These algorithms can be thought of as variations of the Cooley–Tukey radix-2 ones. Their computational complexity in terms of floating-point operations is the same as a well-programmed Cooley–Tukey-type algorithm because the same number of multiplications and additions are involved, but the algorithms presented avoid the bit-reversed ordering applied at the input or at the output samples. To have an idea of different bit-reversed algorithms and its complexity, see references [22]–[27]. The strategy presented to obtain the factorizations is based on a recursive property relating matrix $\mathbf{F}_{\mathrm{N}}$ and $\mathbf{F}_{\mathrm{N}/2}$ that opens the possibility of deriving sorted radix-4, -8, or -16 algorithms taking advantage of its efficiency in terms of algebraic complexity.

References

[1] M. T. Heideman, D. H. Johnson, and C. S. Burrus, "Gauss and the history of the FFT," *IEEE Acoust., Speech, Signal Process. Mag.*, vol. 1, pp. 14–21, Oct. 1984.

[2] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, Apr. 1965.

[3] G. D. Bergland, "A radix-eight fast-Fourier transform subroutine for real-valued series," *IEEE Trans. Audio Electroacoust.*, vol. 17, no. 2, pp. 138–144, Jun. 1969.

[4] D. Takahashi, "A radix-16 FFT algorithm suitable for multiply-add instruction based on Goedecker method," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing, ICASSP-2003*, Apr. 6–10, 2003, vol. 2, pp. 665–668.

[5] R. C. Singleton, "An algorithm for computing the mixed radix fast Fourier transform," *IEEE Trans. Audio Electroacoust.*, vol. 1, no. 2, pp. 93–103, Jun. 1969.

[6] D. P. Kolba and T. W. Parks, "A prime factor FFT algorithm using high-speed convolution," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 4, pp. 281–294, Aug. 1977.

[7] S. Winograd, "On computing the discrete Fourier transform," *Math. Comput.*, vol. 32, no. 141, pp. 175–199, Jan. 1978.

[8] H. V. Sorensen and C. S. Burrus, "A new efficient algorithm for computing a few DFT points," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 35, no. 6, pp. 849–863, Jun. 1987.

[9] D. Takahashi, "An extended split-radix FFT algorithm," *IEEE Signal Process. Lett.*, vol. 8, no. 5, pp. 145–147, May 2001.

[10] A. R. Varkonyi-Koczy, "A recursive Fast Fourier Transform algorithm," *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 614–616, Sep. 1995.

[11] A. Saidi, "Decimation-in-time-frequency FFT algorithm," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, Apr. 19–22, 1994, vol. 3, pp. 453–456.

[12] M. C. Pease, "An adaptation of the fast Fourier transform for parallel processing," *J. Assoc. Comput.*, vol. 15, pp. 252–264, Apr. 1968.

[13] P. Duhamel and M. Vetterli, "Fast Fourier transforms: A tutorial review and a state of the art," *Signal Process.*, vol. 19, pp. 259–299, 1990.

[14] J. A. Glassman, "A generalization of the fast Fourier transform," *IEEE Trans. Comput.*, vol. C-19, pp. 105–116, Feb. 1970.

[15] M. Drubin, "Kronecker product factorization of the FFT matrix," *IEEE Trans. Comput.*, vol. C-20, pp. 590–593, May 1971.

[16] H. Sloate, "Matrix representations for sorting and the fast Fourier transform," *IEEE Trans. Circuits Syst.*, vol. 21, no. 1, pp. 109–116, Jan. 1974.

[17] J. Granata, M. Conner, and R. Tolimieri, "Recursive fast algorithms and the role of the tensor product," *IEEE Trans. Signal Process.*, vol. 40, no. 12, pp. 2921–2930, Dec. 1992.

[18] S. Egner and M. Püschel, "Automatic generation of fast discrete signal transforms," *IEEE Trans. Signal Process.*, vol. 49, no. 9, pp. 1992–2002, Sep. 2001.

[19] S. C. Chan and K. L. Ho, "On indexing the prime-factor fast Fourier transform algorithm," *IEEE Trans. Circuits Syst.*, vol. 38, no. 8, pp. 951–953, 1991.

[20] P. Duhamel and H. Hollmann, "Split-radix FFT algorithm," *Electron. Lett.*, vol. 20, no. 1, pp. 14–16, 1984.

[21] S. G. Johnson and M. Frigo, "A modified split-radix FFT with fewer arithmetic operations," *IEEE Trans. Signal Process.*, vol. 55, no. 1, pp. 111–119, Jan. 2007.

[22] R. J. Polge, B. K. Bhagavan, and J. M. Carswell, "Fast computational algorithms for bit reversal," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 1–9, Jan. 1974.

[23] J. S. Walker, "A new bit reversal algorithm," *IEEE Trans. Acoustics, Speech, Signal Process.*, vol. 38, no. 8, pp. 1472–1473, Aug. 1990.

[24] A. Biswas, "Bit reversal in FFT from matrix viewpoint," *IEEE Trans. Signal Process.*, vol. 39, no. 6, pp. 1415–1418, Jun. 1991.

[25] A. Edelman, S. Heller, and S. L. Johnsson, "Index transformation algorithms in a linear algebra framework," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 12, pp. 1302–1309, Dec. 1994.

[26] K. Drouiche, "A new efficient computational algorithm for bit reversal mapping," *IEEE Trans. Signal Process.*, vol. 49, no. 1, pp. 251–254, Jan. 2001.

[27] J. Prado, "A new fast bit-reversal permutation algorithm based on a symmetry," *IEEE Signal Process. Lett.*, vol. 11, no. 12, pp. 933–936, Dec. 2004.

[28] W. T. Cochrane, J. W. Cooley, J. W. Favin, D. L. Helms, R. A. Kaenel, W. W. Lang, G. C. Mailing, D. E. Nelson, C. M. Rader, and P. D. Welch, "What is the fast Fourier transform?," *IEEE Trans. Audio Electroacoust.*, vol. 15, pp. 45–55, 1967.

[29] P. N. Swarztrauber, "FFT algorithms for vector computers," *Parallel Comput.*, vol. 1, pp. 45–63, 1984.

[30] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*. Philadelphia, PA: SIAM, 1992.