Lifting-based Alternatives to the FFT for Computing the 4-, 8-, and 16-point Discrete Fourier Transforms

Marek Parfieniuk
Institute of Computer Science
University of Bialystok
Bialystok, Poland
ORCID: 0000-0002-4436-8849

***Abstract*—This paper presents a novel approach to efficiently compute the 4-, 8-, and 16-point variants of the Discrete Fourier Transform (DFT). Our solution requires as many (non-trivial) multiplications and additions of real numbers as fast algorithms of the FFT family. However, we saved operations in a completely different way. Instead of factorizing the complex matrix that describes the transform, we consider separately the real and imaginary parts of the DFT matrix. Both parts are rank-deficient matrices, which contain pairs of linearly dependent rows and columns, but are structured and symmetric. We factorize each of them into a product of sparse matrices by applying shear transforms to rows and columns, similarly as in two-side diagonalization. The obtainable factorizations describe data flow graphs in which non-trivial multiplications are well grouped, as they occur in the midst of sequences of lifting steps related to trivial multiplications by powers of 2, which can be implemented by bit-shifting. Such schemes could be used to implement the DFT in hardware, as they offer alternative possibilities of modularization, pipelining, and folding of digital circuits.**

***Index Terms*—FFT, DFT, discrete, Fourier, transform, algorithm, lifting, shear, matrix**

I. Introduction

The Discrete Fourier Transform (DFT) facilitates harmonic analysis and frequency-domain processing of signals. As these have found numerous applications, this transform is one of mathematical and computational tools that form the foundations of the contemporary science and engineering [1].

The DFT is popular also because fast algorithms have been developed for computing it. Efficient methods for computing this transform usually are based on the FFT approach by Cooley and Tukey [2]. The idea is to represent the DFT as a composition of simpler, elementary transforms, so-called butterflies.

It is less known that methods for efficiently computing the DFT can be developed as well by considering separately the real and imaginary parts of its matrix [3]. These parts comprise real numbers, so they represent subtransforms that produce real-valued coefficients from real-valued signals. An efficient algorithm can be obtained by factorizing each of the parts into matrices that represent sine-cosine transforms for which fast implementations are known. Similar solutions have been derived without matrix notations, by considering real and imaginary parts of twiddle factors of butterflies [4][5][6].

The latter approach allows for computing the DFT by using as many or even fewer operations compared to FFT algorithms. Moreover, one can easily prune data flow graphs so as to adapt them for efficiently processing real-valued signals. FFT-based computational schemes contain complex multiplications in various places, so pruning possibilities are limited.

In this paper, we propose a solution based on considering real and imaginary parts of the DFT matrix. Contrary to the known approaches of this kind, our method is not based on connecting the parts with particular transforms. We use shear transforms to iteratively reduce a part of the DFT matrix, so as to obtain a sparse matrix which represents multiplications that cannot be implemented by bit shifting. This matrix describes the central part of a data flow graph for efficiently computing a given variant of the DFT. This central part is preceded and followed by complementary series of lifting steps [7]. So, our method allows for obtaining algorithms that can be implemented in hardware, as digital circuits which can be modularized, pipelined, and folded in ways different from those of the known solutions.

Lifting schemes have been already used for computing the DFT, but in ways completely different from our one. In [8], lifting schemes have been used to replace complex multiplications and butterflies in data flow graphs of FFT algorithms. In [9] and [10], the DFT has been decomposed into the Hartley transform and some residual transform, for which lifting-based implementations have been developed.

The present work is a continuation of our research described in [11], where we considered the Walsh-Hadamard Transform (WHT). Therein, it has been shown that WHT matrices can be effectively factorized by applying shears to rows and than to columns of a symmetric matrix. This results in innovative, lifting-based data flow graphs that can be used to develop improved programs and circuits for computing the WHT.

This work was financially supported by the Institute of Computer Science, University of Bialystok.

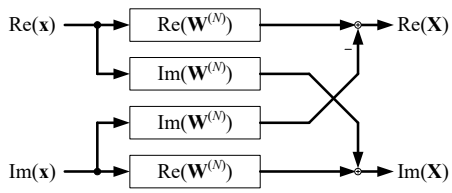

Fig. 1. Data-flow graph of the DFT decomposed into transforms related to real and imaginary parts of its matrix.

II. Real and imaginary parts of the DFT matrix

The N-point DFT of a signal is usually described using the formula for the kth coefficient of the transform:

$$X(k) = \sum_{n=0}^{N-1} e^{-j2\pi \frac{kn}{N}} x(n), \quad k = 0 \ldots N-1 \tag{1}$$

where $x(n)$ denotes n-th sample of the signal, and a series of N samples is convolved by the k-th of the transform basis functions, which are discrete complex exponents.

Alternatively, the transform can be described in a matrix notation [12], as

$$\mathbf{X} = \mathbf{W}^{(N)}\mathbf{x} \tag{2}$$

where $\mathbf{x}$ denotes a vector of N signal samples, $\mathbf{X}$ is the corresponding vector of N DFT coefficients, while $\mathbf{W}^{(N)}$ is the $N \times N$ transform matrix. Elements of this matrix are roots of the unity:

$$[\mathbf{W}^{(N)}]_{k+1,n+1} = e^{-j2\pi \frac{kn}{N}} \tag{3}$$

which are complex numbers that correspond to equidistant points on the unit circle.

Fast algorithms for computing the DFT are usually derived by rewriting (1), without using matrix-vector notations. Nevertheless, such algorithms can be as well described by factorizations of the DFT matrix, so that it is represented as a product of sparse matrices.

Our research is based on the matrix notation. We consider separately the real and imaginary parts of the DFT matrix, decomposing (2) as follows:

$$\begin{bmatrix} \mathrm{Re}(\mathbf{X}) \\ \mathrm{Im}(\mathbf{X}) \end{bmatrix} = \begin{bmatrix} \mathrm{Re}(\mathbf{W}^{(N)}) & -\mathrm{Im}(\mathbf{W}^{(N)}) \\ \mathrm{Im}(\mathbf{W}^{(N)}) & \mathrm{Re}(\mathbf{W}^{(N)}) \end{bmatrix} \begin{bmatrix} \mathrm{Re}(\mathbf{x}) \\ \mathrm{Im}(\mathbf{x}) \end{bmatrix} \tag{4}$$

which describes the data-flow graph in Fig. 1.

Let us define $c_3 = 0.3827$, $c_7 = 0.7071$, and $c_9 = 0.9239$, in order to be able to write more compact equations. By using these constants, we can show how the real and imaginary parts look for the 4-point DFT:

$$\mathrm{Re}(\mathbf{W}^{(4)}) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix} \tag{5}$$

$$\mathrm{Im}(\mathbf{W}^{(4)}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \tag{6}$$

for the 8-point DFT:

$$\mathrm{Re}(\mathbf{W}^{(8)}) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & c_7 & 0 & -c_7 & -1 & -c_7 & 0 & c_7 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & -c_7 & 0 & c_7 & -1 & c_7 & 0 & -c_7 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -c_7 & 0 & c_7 & -1 & c_7 & 0 & -c_7 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & c_7 & 0 & -c_7 & -1 & -c_7 & 0 & c_7 \end{bmatrix} \tag{7}$$

$$\mathrm{Im}(\mathbf{W}^{(8)}) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -c_7 & -1 & -c_7 & 0 & c_7 & 1 & c_7 \\ 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & -c_7 & 1 & -c_7 & 0 & c_7 & -1 & c_7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_7 & -1 & c_7 & 0 & -c_7 & 1 & -c_7 \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 0 & c_7 & 1 & c_7 & 0 & -c_7 & -1 & -c_7 \end{bmatrix} \tag{8}$$

and for the 16-point DFT:

$$\mathrm{Re}(\mathbf{W}^{(16)}) = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & c_9 & c_7 & c_3 & 0 & -c_3 & -c_7 & -c_9 & -1 & -c_9 & -c_7 & -c_3 & 0 & c_3 & c_7 & c_9 \\
1 & c_7 & 0 & -c_7 & -1 & -c_7 & 0 & c_7 & 1 & c_7 & 0 & -c_7 & -1 & -c_7 & 0 & c_7 \\
1 & c_3 & -c_7 & -c_9 & 0 & c_9 & c_7 & -c_3 & -1 & -c_3 & c_7 & c_9 & 0 & -c_9 & -c_7 & c_3 \\
1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\
1 & -c_3 & -c_7 & c_9 & 0 & -c_9 & c_7 & c_3 & -1 & c_3 & c_7 & -c_9 & 0 & c_9 & -c_7 & -c_3 \\
1 & -c_7 & 0 & c_7 & -1 & c_7 & 0 & -c_7 & 1 & -c_7 & 0 & c_7 & -1 & c_7 & 0 & -c_7 \\
1 & -c_9 & c_7 & -c_3 & 0 & c_3 & -c_7 & c_9 & -1 & c_9 & -c_7 & c_3 & 0 & -c_3 & c_7 & -c_9 \\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\
1 & -c_9 & c_7 & -c_3 & 0 & c_3 & -c_7 & c_9 & -1 & c_9 & -c_7 & c_3 & 0 & -c_3 & c_7 & -c_9 \\
1 & -c_7 & 0 & c_7 & -1 & c_7 & 0 & -c_7 & 1 & -c_7 & 0 & c_7 & -1 & c_7 & 0 & -c_7 \\
1 & -c_3 & -c_7 & c_9 & 0 & -c_9 & c_7 & c_3 & -1 & c_3 & c_7 & -c_9 & 0 & c_9 & -c_7 & -c_3 \\
1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\
1 & c_3 & -c_7 & -c_9 & 0 & c_9 & c_7 & -c_3 & -1 & -c_3 & c_7 & c_9 & 0 & -c_9 & -c_7 & c_3 \\
1 & c_7 & 0 & -c_7 & -1 & -c_7 & 0 & c_7 & 1 & c_7 & 0 & -c_7 & -1 & -c_7 & 0 & c_7 \\
1 & c_9 & c_7 & c_3 & 0 & -c_3 & -c_7 & -c_9 & -1 & -c_9 & -c_7 & -c_3 & 0 & c_3 & c_7 & c_9
\end{bmatrix} \tag{9}$$

and

$$\mathrm{Im}(\mathbf{W}^{(16)}) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -c_3 & -c_7 & -c_9 & -1 & -c_9 & -c_7 & -c_3 & 0 & c_3 & c_7 & c_9 & 1 & c_9 & c_7 & c_3 \\
0 & -c_7 & -1 & -c_7 & 0 & c_7 & 1 & c_7 & 0 & -c_7 & -1 & -c_7 & 0 & c_7 & 1 & c_7 \\
0 & -c_9 & -c_7 & c_3 & 1 & c_3 & -c_7 & -c_9 & 0 & c_9 & c_7 & -c_3 & -1 & -c_3 & c_7 & c_9 \\
0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 \\
0 & -c_9 & c_7 & c_3 & -1 & c_3 & c_7 & -c_9 & 0 & c_9 & -c_7 & -c_3 & 1 & -c_3 & -c_7 & c_9 \\
0 & -c_7 & 1 & -c_7 & 0 & c_7 & -1 & c_7 & 0 & -c_7 & 1 & -c_7 & 0 & c_7 & -1 & c_7 \\
0 & -c_3 & c_7 & -c_9 & 1 & -c_9 & c_7 & -c_3 & 0 & c_3 & -c_7 & c_9 & -1 & c_9 & -c_7 & c_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & c_3 & -c_7 & c_9 & -1 & c_9 & -c_7 & c_3 & 0 & -c_3 & c_7 & -c_9 & 1 & -c_9 & c_7 & -c_3 \\
0 & c_7 & -1 & c_7 & 0 & -c_7 & 1 & -c_7 & 0 & c_7 & -1 & c_7 & 0 & -c_7 & 1 & -c_7 \\
0 & c_9 & -c_7 & -c_3 & 1 & -c_3 & -c_7 & c_9 & 0 & -c_9 & c_7 & c_3 & -1 & c_3 & c_7 & -c_9 \\
0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\
0 & c_9 & c_7 & -c_3 & -1 & -c_3 & c_7 & c_9 & 0 & -c_9 & -c_7 & c_3 & 1 & c_3 & -c_7 & -c_9 \\
0 & c_7 & 1 & c_7 & 0 & -c_7 & -1 & -c_7 & 0 & c_7 & 1 & c_7 & 0 & -c_7 & -1 & -c_7 \\
0 & c_3 & c_7 & c_9 & 1 & c_9 & c_7 & c_3 & 0 & -c_3 & -c_7 & -c_9 & -1 & -c_9 & -c_7 & -c_3
\end{bmatrix} \tag{10}$$

The matrices $\mathrm{Re}(\mathbf{W}^{(N)})$ and $\mathrm{Im}(\mathbf{W}^{(N)})$ do not describe invertible transforms. They are singular matrices, having the ranks of $\frac{N}{2}+1$ and $\frac{N}{2}-1$, respectively. Both parts of the DFT matrix are symmetric along the main diagonal.

III. A lifting-based method for factorizing parts of DFT matrix

Let us use $\mathbf{L}^{(N)}(m, n, \alpha)$ to denote the $N \times N$ matrix of a shear transform. Such a matrix has ones on its main

diagonal, while α is the only non-zero element outside the diagonal, at the position where the mth row and the nth column intersect. For example,

$$\mathbf{L}^{(4)}(2,3,\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & \alpha & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

Shear matrices describe data flow graphs that are called lifting steps and schemes [7]. The name is related to their application in constructing wavelets. They are also interesting for persons which have to implement filters and transforms in hardware, as lifting schemes are well suited to be implemented as digital circuits based on fixed-point arithmetic [8].

We propose to use shear matrices to factorize both $\mathrm{Re}(\mathbf{W}^{(N)})$ and $\mathrm{Im}(\mathbf{W}^{(N)})$. Shears can be applied to rows of a part of the DFT matrix, and then to columns of the resulting matrix, again to rows, and so on. In this way, one can obtain a sparse matrix which cannot be reduced more. For example, $\mathrm{Re}(\mathbf{W}^{(4)})$ can be reduced as follows

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}}_{\mathbf{L}^{(4)}(4,2,-1)} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 \\ 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Last row zeroed}} \tag{12}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & -1 & 0 \\ 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{L}^{(4)}(2,4,-1)} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Last column zeroed}} \tag{13}$$

and so on.

One applies a shear to rows/columns of a matrix by left/right-multiplying the matrix by the shear one. Depending on m, n, α, a shear can zero more or fewer elements of a matrix. We propose to choose the parameters in a greedy fashion, so as to zero as many elements as possible. If various shears can zero as many elements, then we prefer the shear which zeroes elements farthest from the upper-left corner of the matrix.

The symmetries of $\mathrm{Re}(\mathbf{W}^{(N)})$ and $\mathrm{Im}(\mathbf{W}^{(N)})$ can be exploited to simplify determination of shears. If the left-multiplication by $\mathbf{L}(m,n,\alpha)$ zeroes some elements in the n-th row of a symmetric matrix, then the right-multiplication by $\mathbf{L}(n,m)$ zeroes the corresponding elements in the n-th column. Such a reduction algorithm is a kind of Gaussian elimination, greedy and constrained. It has been described in more details in [11].

We can determine two categories of lifting steps. Steps of one category reveal the rank of a part of a DFT matrix, by zeroing redundant rows and columns. The remaining full-rank submatrix is made sparse by shears of the second category, but they do not modify its rank. It is trivial to determine lifting steps of the first category.

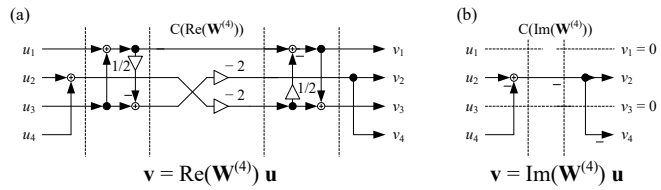

Fig. 2. Data-flow graphs of parts of 4-point DFT.

IV. A LIFTING-BASED FACTORIZATIONS OF PARTS OF DFT MATRIX

A. 4-point DFT

For the 4-point DFT, $\mathrm{Re}(\mathbf{W}^{(4)})$ can be reduced to

$$C(\mathrm{Re}(\mathbf{W}^{(4)})) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{14}$$

and represented as

$$\begin{aligned} \mathrm{Re}(\mathbf{W}^{(4)}) = & \mathbf{L}(4,2,1) \\ & \cdot \mathbf{L}(3,1,1) \cdot \mathbf{L}(1,3,-\tfrac{1}{2}) \\ & \cdot C(\mathrm{Re}(\mathbf{W}^{(4)})) \\ & \cdot \mathbf{L}(3,1,-\tfrac{1}{2}) \cdot \mathbf{L}(1,3,1) \\ & \cdot \mathbf{L}(2,4,1) \end{aligned} \tag{15}$$

which describes the data flow graph in Fig. 2a.

On the other hand,

$$\mathrm{Im}(\mathbf{W}^{(4)}) = \mathbf{L}(4,2,-1) \cdot C(\mathrm{Im}(\mathbf{W}^{(4)})) \cdot \mathbf{L}(2,4,-1) \tag{16}$$

where

$$C(\mathrm{Im}(\mathbf{W}^{(4)})) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{17}$$

The corresponding scheme is shown Fig. 2b.

In this and remaining sections, we write $\mathbf{L}(\ldots)$ instead of $\mathbf{L}^{(N)}(\ldots)$. This makes equations clearer, while N can be easily deduced from the context.

B. 8-point DFT

For the 8-point DFT,

$$\begin{aligned} \mathrm{Re}(\mathbf{W}^{(8)}) = & \mathbf{L}(8,2,1) \cdot \mathbf{L}(7,3,1) \cdot \mathbf{L}(6,4,1) \\ & \cdot \mathbf{L}(5,1,1) \cdot \mathbf{L}(1,5,-\tfrac{1}{2}) \\ & \cdot \mathbf{L}(4,2,-1) \cdot \mathbf{L}(2,4,\tfrac{1}{2}) \\ & \cdot C(\mathrm{Re}(\mathbf{W}^{(8)})) \\ & \cdot \mathbf{L}(4,2,\tfrac{1}{2}) \cdot \mathbf{L}(2,4,-1) \\ & \cdot \mathbf{L}(5,1,-\tfrac{1}{2}) \cdot \mathbf{L}(1,5,1) \\ & \cdot \mathbf{L}(4,6,1) \cdot \mathbf{L}(3,7,1) \cdot \mathbf{L}(2,8,1) \end{aligned} \tag{18}$$

where

$$C(\mathrm{Re}(\mathbf{W}^{(8)})) = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & c_7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

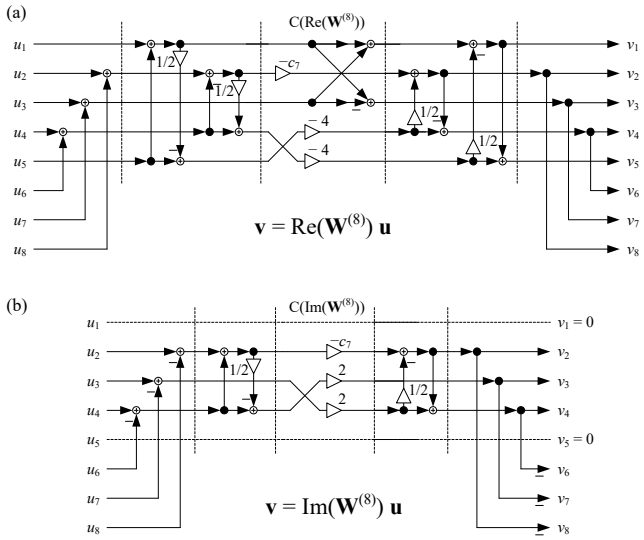

Fig. 3. Data-flow graphs of parts of 8-point DFT.

These equations describe the scheme in Fig. 3a.

The imaginary part can be reduced to

$$C(\mathrm{Im}(\mathbf{W}^{(8)})) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -c_7 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{20}$$

and represented as

$$\begin{aligned} \mathrm{Im}(\mathbf{W}^{(8)}) = & \mathbf{L}(8,2,-1) \cdot \mathbf{L}(7,3,-1) \cdot \mathbf{L}(6,4,-1) \\ & \cdot \mathbf{L}(4,2,1) \cdot \mathbf{L}(2,4,-\tfrac{1}{2}) \\ & \cdot C(\mathrm{Im}(\mathbf{W}^{(8)})) \\ & \cdot \mathbf{L}(4,2,-\tfrac{1}{2}) \cdot \mathbf{L}(2,4,1) \\ & \cdot \mathbf{L}(4,6,-1) \cdot \mathbf{L}(3,7,-1) \cdot \mathbf{L}(2,8,-1) \end{aligned} \tag{21}$$

which describes the scheme in Fig. 3b.

C. 16-point DFT

For the 16-point DFT,

$$\begin{aligned} \mathrm{Re}(\mathbf{W}^{(16)}) = & \mathbf{L}(16,2,1) \cdot \mathbf{L}(15,3,1) \cdot \mathbf{L}(14,4,1) \\ & \cdot \mathbf{L}(13,5,1) \cdot \mathbf{L}(12,6,1) \cdot \mathbf{L}(11,7,1) \cdot \mathbf{L}(10,8,1) \\ & \cdot \mathbf{L}(9,1,1) \cdot \mathbf{L}(1,9,-\tfrac{1}{2}) \cdot \mathbf{L}(8,2,1) \cdot \mathbf{L}(2,8,-\tfrac{1}{2}) \\ & \cdot \mathbf{L}(7,3,1) \cdot \mathbf{L}(3,7,-\tfrac{1}{2}) \cdot \mathbf{L}(6,4,1) \cdot \mathbf{L}(4,6,-\tfrac{1}{2}) \\ & \cdot \mathbf{L}(5,1,1) \cdot \mathbf{L}(1,5,-\tfrac{1}{2}) \cdot \mathbf{L}(4,2,-1) \cdot \mathbf{L}(2,4,\tfrac{1}{2}) \\ & \cdot C(\mathrm{Re}(\mathbf{W}^{(16)})) \\ & \cdot \mathbf{L}(4,2,\tfrac{1}{2}) \cdot \mathbf{L}(2,4,-1) \cdot \mathbf{L}(5,1,-\tfrac{1}{2}) \cdot \mathbf{L}(1,5,1) \\ & \cdot \mathbf{L}(6,4,-\tfrac{1}{2}) \cdot \mathbf{L}(4,6,1) \cdot \mathbf{L}(7,3,-\tfrac{1}{2}) \cdot \mathbf{L}(3,7,1) \\ & \cdot \mathbf{L}(8,2,-\tfrac{1}{2}) \cdot \mathbf{L}(2,8,1) \cdot \mathbf{L}(9,1,-\tfrac{1}{2}) \cdot \mathbf{L}(1,9,1) \\ & \cdot \mathbf{L}(8,10,1) \cdot \mathbf{L}(7,11,1) \cdot \mathbf{L}(6,12,1) \\ & \cdot \mathbf{L}(5,13,1) \cdot \mathbf{L}(4,14,1) \cdot \mathbf{L}(3,15,1) \cdot \mathbf{L}(2,16,1) \end{aligned} \tag{22}$$

where

$$C(\mathrm{Re}(\mathbf{W}^{(16)})) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0}_{1\times 7} \\ 0 & 0 & 0 & 0 & 0 & 0 & -2c_7 & 0 & 0 & \mathbf{0}_{1\times 7} \\ 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & \mathbf{0}_{1\times 7} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & \mathbf{0}_{1\times 7} \\ 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0}_{1\times 7} \\ 0 & 0 & 0 & 0 & 0 & -4c_9 & 0 & 4c_3 & 0 & \mathbf{0}_{1\times 7} \\ 0 & -2c_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0}_{1\times 7} \\ 0 & 0 & 0 & 0 & 0 & 4c_3 & 0 & 4c_9 & 0 & \mathbf{0}_{1\times 7} \\ 0 & 0 & 0 & -4 & 0 & 0 & 0 & 0 & 0 & \mathbf{0}_{1\times 7} \\ \mathbf{0}_{7\times 1} & \mathbf{0}_{7\times 1} & \mathbf{0}_{7\times 1} & \mathbf{0}_{7\times 1} & \mathbf{0}_{7\times 1} & \mathbf{0}_{7\times 1} & \mathbf{0}_{7\times 1} & \mathbf{0}_{7\times 1} & \mathbf{0}_{7\times 1} & \mathbf{0}_{7\times 7} \end{bmatrix} \tag{23}$$

where $\mathbf{0}_{m\times n}$ denotes the matrix of m by n zeros.

As to the imaginary part,

$$\begin{aligned} \mathrm{Im}(\mathbf{W}^{(16)}) = & \mathbf{L}(16,2,-1) \cdot \mathbf{L}(15,3,-1) \cdot \mathbf{L}(14,4,-1) \\ & \cdot \mathbf{L}(13,5,-1) \cdot \mathbf{L}(12,6,-1) \cdot \mathbf{L}(11,7,-1) \cdot \mathbf{L}(10,8,-1) \\ & \cdot \mathbf{L}(8,2,1) \cdot \mathbf{L}(2,8,-\tfrac{1}{2}) \cdot \mathbf{L}(7,3,1) \cdot \mathbf{L}(3,7,-\tfrac{1}{2}) \\ & \cdot \mathbf{L}(6,4,1) \cdot \mathbf{L}(4,6,-\tfrac{1}{2}) \cdot \mathbf{L}(6,8,1) \cdot \mathbf{L}(8,6,-\tfrac{1}{2}) \\ & \cdot C(\mathrm{Im}(\mathbf{W}^{(16)})) \\ & \cdot \mathbf{L}(6,8,-\tfrac{1}{2}) \cdot \mathbf{L}(8,6,1) \cdot \mathbf{L}(6,4,-\tfrac{1}{2}) \cdot \mathbf{L}(4,6,1) \\ & \cdot \mathbf{L}(7,3,-\tfrac{1}{2}) \cdot \mathbf{L}(3,7,1) \cdot \mathbf{L}(8,2,-\tfrac{1}{2}) \cdot \mathbf{L}(2,8,1) \\ & \cdot \mathbf{L}(8,10,-1) \cdot \mathbf{L}(7,11,-1) \cdot \mathbf{L}(6,12,-1) \cdot \mathbf{L}(5,13,-1) \\ & \cdot \mathbf{L}(4,14,-1) \cdot \mathbf{L}(3,15,-1) \cdot \mathbf{L}(2,16,-1) \end{aligned} \tag{24}$$

where

$$C(\mathrm{Im}(\mathbf{W}^{(16)})) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0}_{1\times 8} \\ 0 & -c_3 & 0 & -c_9 & 0 & 0 & 0 & 0 & \mathbf{0}_{1\times 8} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2c_7 & \mathbf{0}_{1\times 8} \\ 0 & -c_9 & 0 & c_3 & 0 & 0 & 0 & 0 & \mathbf{0}_{1\times 8} \\ 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & \mathbf{0}_{1\times 8} \\ 0 & 0 & 0 & 0 & -4 & 0 & 0 & 0 & \mathbf{0}_{1\times 8} \\ 0 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & \mathbf{0}_{1\times 8} \\ 0 & 0 & 2c_7 & 0 & 0 & 0 & 0 & 0 & \mathbf{0}_{1\times 8} \\ \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 1} & \mathbf{0}_{8\times 8} \end{bmatrix} \tag{25}$$

The corresponding data flow graphs are shown in Fig. 4.

D. Remarks about the computational schemes

In order to obtain a complete solution for computing the DFT, one needs to combine the scheme in Fig. 1 with those in Figs. 2, 3, or 4.

In each of the latter schemes, we have denoted its part related to the $C(\ldots)$ matrix, or to multiplications and data reordering. So, both these are performed in the midst of a data flow graph. FFT algorithms require to reorder either inputs or outputs, while non-trivial multiplications are spread over a scheme, being separated by layers of butterflies.

The total numbers of multiplications and additions of real numbers are essentially the same as in the corresponding FFT algorithms. Sign changes described by the $C(\ldots)$ matrix can be combined with lifting steps, so as to avoid superfluous subtractions.

In Figs. 2 – 4, pairs of lifting steps occur that are equivalent to butterflies with scaled outputs. The equivalence is explained in Fig. 5. To the best of our knowledge, nobody reported on similar butterfly-based schemes in the literature.

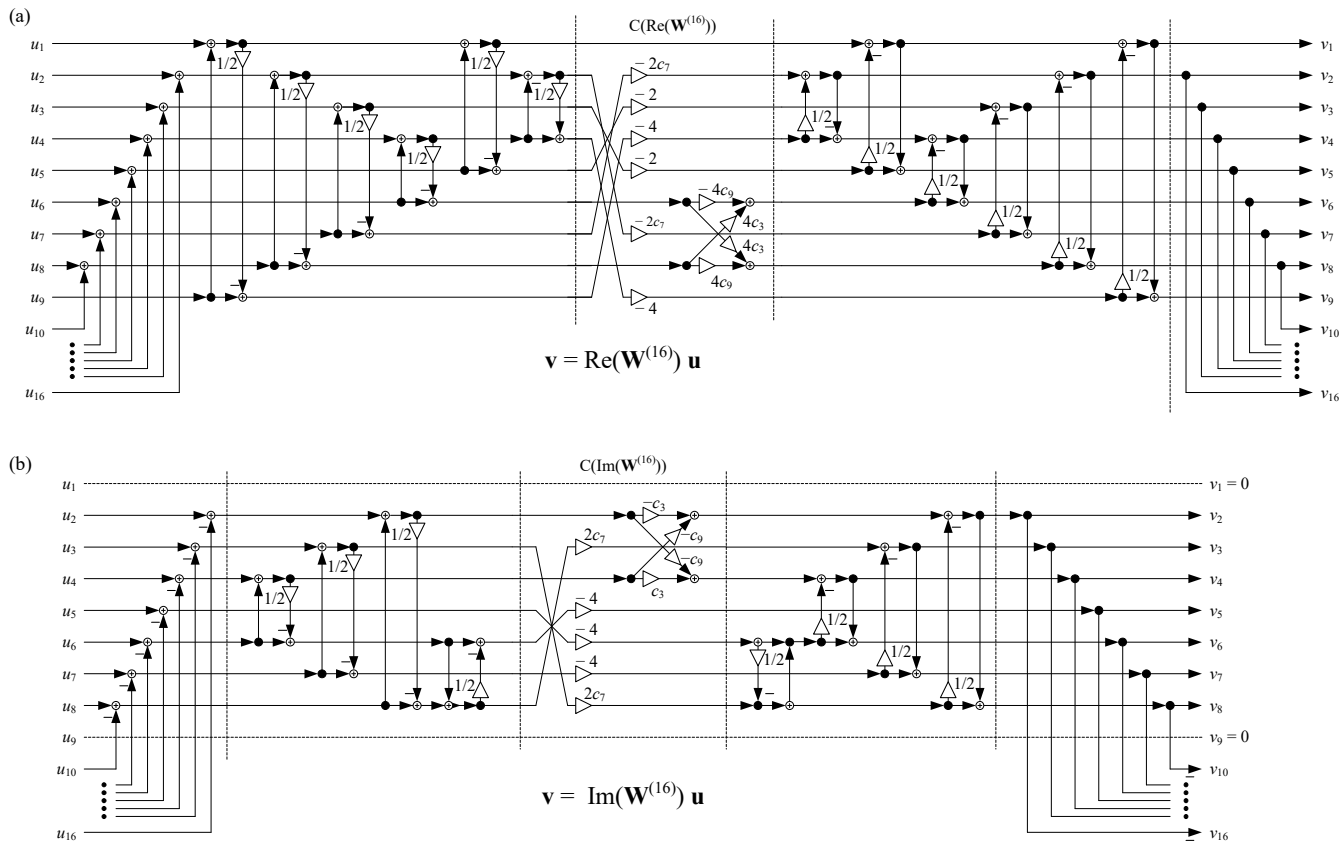

Fig. 4. Data-flow graphs of parts of 16-point DFT.

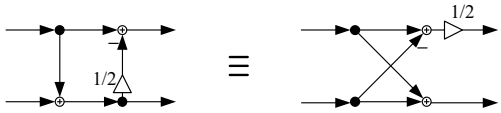

Fig. 5. Equivalence between lifting scheme and butterfly.

V. Conclusion

Even though the related research field has been thoroughly explored by scientists and engineers, we were able to invent a novel approach to computing the DFT. Our method results in data flow graphs which are different from those of the known algorithms. Their specific layouts can be exploited to develop digital circuits modularized, pipelined, and folded so as to better utilize chip area or to improve performance.

References

[1] K. Rao, D. Kim, and J.-J. Hwang, *Fast Fourier Transform: Algorithms and Applications*. Springer, 2010.

[2] G. G. Kumar, S. K. Sahoo, and P. K. Meher, "50 years of FFT algorithms and applications," *Circuits, Systems, Signal Process.*, vol. 38, no. 12, pp. 5665–5698, Dec. 2019.

[3] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations," *Signal Processing*, vol. 6, no. 4, pp. 267 – 278, Aug. 1984.

[4] 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.

[5] D. J. Bernstein, "The tangent FFT," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, S. Boztaş and H.-F. F. Lu, Eds. Springer, 2007.

[6] H. Guo, G. Sitton, and C. Burrus, "The quick Fourier transform: an FFT based on symmetries," *IEEE Trans. Signal Process.*, vol. 46, no. 2, pp. 335–341, Feb. 1998.

[7] W. Sweldens, "The lifting scheme: A custom-design construction of biorthogonal wavelets," *Appl. Comput. Harmon. Anal.*, vol. 3, no. 2, pp. 186–200, Apr. 1996.

[8] S. Oraintara, Y. J. Chen, and T. Q. Nguyen, "Integer Fast Fourier Transform," *IEEE Trans. Signal Process.*, vol. 50, no. 3, pp. 607–618, Mar. 2002.

[9] T. Suzuki, S. Kyochi, Y. Tanaka, M. Ikehara, and H. Aso, "Multiplierless lifting based FFT via fast Hartley transform," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, Vancouver, BC, Canada, 26-31 May 2013, pp. 5603–5607.

[10] T. Suzuki, S. Kyochi, Y. Tanaka, and M. Ikehara, "Multiplierless lifting-based fast X transforms derived from fast Hartley transform factorization," *Multidimensional Systems Signal Process.*, vol. 29, no. 1, pp. 99–118, Jan. 2018.

[11] M. Parfieniuk, "Lifting-based algorithms for computing the 4-point Walsh-Hadamard transform," in *Proc. Signal Processing Symposium (SPSympo)*, 17-19 Sep. 2019, pp. 221–226.

[12] D. Kahaner, "Matrix description of the fast Fourier transform," *IEEE Trans. Audio Electroacoust.*, vol. 18, no. 4, pp. 442–450, Dec. 1970.