Hindawi Publishing Corporation
The Scientific World Journal
Volume 2015, Article ID 348517, 7 pages
http://dx.doi.org/10.1155/2015/348517

Research Article

A Parallel Framework with Block Matrices of a Discrete Fourier Transform for Vector-Valued Discrete-Time Signals

Pablo Soto-Quiros[1,2]

[1]*Escuela de Matemáticas, Instituto Tecnológico de Costa Rica, Apartado 159-7050, 30101 Cartago, Costa Rica*
[2]*Centre for Industrial and Applied Mathematics, University of South Australia, Adelaide, SA 5095, Australia*

Correspondence should be addressed to Pablo Soto-Quiros; sotjp001@mymail.unisa.edu.au

Received 6 May 2015; Revised 18 July 2015; Accepted 2 September 2015

Academic Editor: Bruno Carpentieri

This paper presents a parallel implementation of a kind of discrete Fourier transform (DFT): the vector-valued DFT. The vector-valued DFT is a novel tool to analyze the spectra of vector-valued discrete-time signals. This parallel implementation is developed in terms of a mathematical framework with a set of block matrix operations. These block matrix operations contribute to analysis, design, and implementation of parallel algorithms in multicore processors. In this work, an implementation and experimental investigation of the mathematical framework are performed using MATLAB with the Parallel Computing Toolbox. We found that there is advantage to use multicore processors and a parallel computing environment to minimize the high execution time. Additionally, speedup increases when the number of logical processors and length of the signal increase.

1. Introduction

Let $l^2(\mathbb{Z}_n, \mathbb{C}^d)$ be the space of vector-valued discrete-time signals with n samples, where each sample is a complex vector of length d. The vector-valued discrete-time signals are used very often in several applications in signal processing and electrical engineer, for example, vector quantization of images [1], time-frequency localization with wavelets [2], image coding [3], vector filter bank theory [4], linear time-dependent MISO [5], and analysis of MMSE estimation for compressive sensing of block sparse signals [6].

Now, to analyze the spectra of vector-valued discrete-time signals, a novel tool was developed, and it is called *vector-valued DFT* [7, 8]. This transform has applications in vector analysis in complex, quaternion, biquaternion, and Clifford algebras [8]. Additionally, the vector-valued DFT is used in digital signal processing, for example, the study of new complex valued constant amplitude zero autocorrelation (CAZAC) signals [9], which serve as coefficients for phase coded waveforms with prescribed vector-valued ambiguity function behavior, which is relevant in light of time-frequency analysis, vector sensor, and MIMO technologies [7].

The following paper presents a parallel framework of the vector-valued DFT. The major contributions of this paper are summarized as follows:

(1) The construction of a new mathematical structure for the vector-valued DFT using block matrix theory such that it allows a parallel implementation in multicore processors.

(2) Reducing the elapsed time to compute the vector-valued DFT of a vector-valued discrete-time signal using parallel computing through aforementioned new mathematical framework.

This new framework is developed with a set of block matrix operations, for example, Kronecker product, direct sum, stride permutation, vec operator, and vec inverse operator (see Section 2.1 for details). These block matrix operations contribute to analysis, design, and implementation of parallel algorithms in multicore processors [10–12]. This mathematical framework is inspired in the matrix representation of the Cooley-Tukey fast Fourier transform (FFT) algorithm for complex discrete-time signals, corresponding to the decomposition of the transform size n into the product of two factors r and s, which is developed in [10, 12, 13].

The present paper is organized as follows. Section 2 explains a mathematical background about block matrix operations and discrete Fourier transform. Section 3 defines the concept of vector-valued DFT for vector-valued discrete-time signals. Section 4 develops a mathematical framework of vector-valued DFT in terms of block matrix operations for vector-valued discrete-time signals with length $n = rs$. This mathematical framework contributes to implementation of parallel algorithms. Section 5 explains an implementation and experimental investigation of this mathematical framework using parallel computing in multicore processors with MATLAB. Finally, some conclusions are presented in Section 6.

Throughout the paper, the following notations are used. $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ is the additive group $\mathbb{Z}$ of integers modulo n, $\mathbb{C}^{m\times n}$ is the matrix space of m rows and n columns with complex numbers entries and $\mathbb{C}^n = \mathbb{C}^{n\times 1}$. The rows and columns of $\mathbf{A} \in \mathbb{C}^{m\times n}$ are indexed by elements of $\mathbb{Z}_m$ and $\mathbb{Z}_n$, respectively. $\mathbf{A}(j,k)$, $\mathbf{A}(j,:)$, $\mathbf{A}(:,k)$, and $\mathbf{A}^T$ represent entry (j,k), row j, column k, and transpose matrix of $\mathbf{A}$, respectively. $\mathbf{I}_n \in \mathbb{C}^{n\times n}$ is identity matrix.

2. Background

2.1. Block Matrix Operations. A block matrix $\mathbf{A} \in \mathbb{C}^{mp\times nq}$ with m row partitions and n column partitions and a block vector $\mathbf{x} \in \mathbb{C}^{mp}$ with m row blocks are defined as

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{0,0} & \cdots & \mathbf{A}_{0,n-1} \\ \vdots & \ddots & \vdots \\ \mathbf{A}_{m-1,0} & \cdots & \mathbf{A}_{m-1,n-1} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} \mathbf{x}_0 \\ \vdots \\ \mathbf{x}_{m-1} \end{pmatrix}, \tag{1}$$

respectively, where $\mathbf{A}_{j,k} \in \mathbb{C}^{p\times q}$ designates (j,k) block and $\mathbf{x}_j \in \mathbb{C}^p$ designates j block. In this paper, the following block matrix operations are used: Kronecker product, direct sum, stride permutation, vec operator, and vec inverse operator.

The Kronecker product of two matrices $\mathbf{A} \in \mathbb{C}^{m\times n}$ and $\mathbf{B} \in \mathbb{C}^{p\times q}$ is defined as $\mathbf{A} \otimes \mathbf{B} \in \mathbb{C}^{mp\times nq}$ and it replaces every entry (j,k) of $\mathbf{A}$ by the matrix $\mathbf{A}(j,k)\mathbf{B}$. In the special case $\mathbf{A} = \mathbf{I}_n$, it is called parallel operation [12].

The direct sum of n matrices constructs a block diagonal matrix from a set of matrices, that is, for $\{\mathbf{C}_k\}_{k\in\mathbb{Z}_n}$, such that $\mathbf{C}_k \in \mathbb{C}^{p_k\times q_{n_k}}$:

$$\mathbf{C} = \bigoplus_{k\in\mathbb{Z}_n} \mathbf{C}_k = \operatorname{diag}\left(\mathbf{C}_0, \mathbf{C}_1, \ldots, \mathbf{C}_{n-1}\right), \tag{2}$$

where $\mathbf{C} \in \mathbb{C}^{p\times q}$, $p = \sum_{j\in\mathbb{Z}_n} p_j$, and $q = \sum_{j\in\mathbb{Z}_n} q_j$.

Let $n = rs$. The stride permutation matrix is defined as $\mathbf{L}_s^n \in \mathbb{C}^{n\times n}$ such that it permutes the elements of the input signal $\mathbf{x} \in \mathbb{C}^n$ as $jr + k \to ks + j$, $j \in \mathbb{Z}_s$, and $k \in \mathbb{Z}_r$ [12, 14]. This matrix permutation governs the data flow required to parallelize a Kronecker product computation [12]. We clarify that the superscript n is an index, not power.

The vec operator, $\mathscr{V} : \mathbb{C}^{m\times n} \to \mathbb{C}^{mn}$, transforms a matrix into a vector by stacking all the columns of this matrix one underneath the other. On the other hand, the vec inverse operator, $\mathscr{R}_{m,n} : \mathbb{C}^{mn} \to \mathbb{C}^{m\times n}$, transforms a vector of dimension mn into a matrix of size $m \times n$.

2.2. Discrete Fourier Transform. Let $l^2(\mathbb{Z}_n)$ be the set of $\mathbb{C}$-valued signals on $\mathbb{Z}_n$; that is, $\mathbf{x} \in l^2(\mathbb{Z}_n)$ if and only if $\mathbf{x} \in \mathbb{C}^n$ [9]. Additionally, for each $k_1 \in \mathbb{Z}$, $\mathbf{x}(k_1) = \mathbf{x}(k_2)$, where $k_2 \in \mathbb{Z}_n$ and $k_1 \equiv k_2 \bmod n$. The discrete Fourier transform (DFT) of $\mathbf{x} \in l^2(\mathbb{Z}_n)$ is represented as $\mathscr{F}_\mathbf{x} : \mathbb{Z}_n \to \mathbb{C}$ such that $\mathscr{F}_\mathbf{x}(k) = \sum_{m\in\mathbb{Z}_n} \mathbf{x}(m)\omega_N^{-mk}$, where $\omega_n = \exp(2\pi i/n)$ and $i = \sqrt{-1}$.

As mentioned in [14], there are two different approaches of representing the DFT: as matrix-vector products or using summations. Consequently, fast algorithms using parallel computing are represented with either a matrix formalism as in [10, 12–14] or summations as in most signal processing books. Below, the matrix formalism is introduced and used to express the Cooley-Tukey FFT algorithm, corresponding to the decomposition of the transform size n into the product of two factors r and s; that is, $n = rs$.

The matrix representation of DFT of $\mathbf{x}$ is $\mathscr{F}_\mathbf{x} = \mathbf{F}_n\mathbf{x}$, where $\mathbf{F}_n \in \mathbb{C}^{n\times n}$ such that $\mathbf{F}_n(j,k) = \omega_N^{-jk}$. If $n = rs$, then the matrix formalism can be used to express $\mathbf{F}_n$ as factorizations of matrices using block matrices operations [10, 12, 13]:

$$\mathbf{F}_n = \mathbf{L}_s^n\left(\mathbf{I}_r \otimes \mathbf{F}_s\right)\mathbf{L}_r^n\mathbf{T}_r^n\left(\mathbf{I}_s \otimes \mathbf{F}_r\right)\mathbf{L}_s^n. \tag{3}$$

Here, $\mathbf{T}_r^n$ is a diagonal matrix containing the twiddle factors. We clarify that the superscript n is an index, not power. This factorization of $\mathbf{F}_n$ is the matrix representation of the Cooley-Tukey FFT for $n = rs$. In addition, this representation of $\mathbf{F}_n$ allows the implementation using parallel computing [14].

3. DFT for Vector-Valued Signals

Based on [2, 6–9, 15, 16], the space of vector-valued discrete-time signals with n samples is defined as

$$l^2\left(\mathbb{Z}_n, \mathbb{C}^d\right) = \left\{\left(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}\right)^T : \mathbf{x}_j \in \mathbb{C}^d, j \in \mathbb{Z}_n\right\}. \tag{4}$$

The space $l^2(\mathbb{Z}_n, \mathbb{C}^d)$ is the set of $\mathbb{C}^d$-valued signals on $\mathbb{Z}_n$; that is, $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{C}^d)$ if and only if $\mathbf{x} \in \mathbb{C}^{nd}$. Additionally, for each $k_1 \in \mathbb{Z}$, $\mathbf{x}_{k_1} = \mathbf{x}_{k_2}$, where $k_2 \in \mathbb{Z}_n$ and $k_1 \equiv k_2 \bmod n$. Furthermore, if $d = 1$, then $l^2(\mathbb{Z}_n, \mathbb{C}^d) = l^2(\mathbb{Z}_n)$. Now, for $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{C}^d)$, there is a kind of DFT for vector-valued signals called *vector-valued DFT*. This transform is defined as $\mathscr{F}_\mathbf{x}^d : \mathbb{Z}_n \to \mathbb{C}^d$ such that

$$\mathscr{F}_\mathbf{x}^d(k) = \sum_{m\in\mathbb{Z}_n} \mathbf{W}_n^{-mk}\mathbf{x}_m, \tag{5}$$

where $\mathbf{W}_n \in \mathbb{C}^{d\times d}$ is the matrix kernel. Algorithm 1 shows the implementation of (5). This implementation is a sequential algorithm.

Require: $\mathbf{x} \in \mathbb{C}^{dn}$
Ensure: $\mathbf{y} \in \mathbb{C}^{dn}$
(1) **for** $k \leftarrow 0 : n-1$
(2) **for** $m \leftarrow 0 : n-1$
(3) $\mathbf{y}_k = \mathbf{y}_k + \mathbf{W}_n^{-mk}\mathbf{x}_m$
(4) **end for**
(5) **end for**

Algorithm 1: Vector-valued DFT (sequential algorithm).

From the reviewed literature, there are two kinds of kernels for this transform: the first one is *hypercomplex DFT kernel* [8]:

$$\mathbf{W}_n = \exp\left(\frac{2\pi}{n}\cdot\mathbf{J}\right) = \cos\left(\frac{2\pi}{n}\right)\cdot\mathbf{I}_d + \sin\left(\frac{2\pi}{n}\right)\cdot\mathbf{J}, \quad (6)$$

where $\mathbf{J} \in \mathbb{C}^{d\times d}$ such that $\mathbf{J}^2 = -\mathbf{I}_d$, and the second one is *DFT frame kernel* [7]:

$$\mathbf{W}_n = \frac{1}{\sqrt{d}}\operatorname{diag}\left(\omega_n^{\alpha_0}, \omega_n^{\alpha_1}, \ldots, \omega_n^{\alpha_{d-1}}\right), \quad (7)$$

where $\mathscr{A} = \{\alpha_0, \alpha_1, \ldots, \alpha_{d-1}\} \subset \mathbb{Z}_n$ with $\alpha_j < \alpha_k$ for $j < k$. It is called DFT frame kernel because $\{\mathbf{e}_j\}_{j\in\mathbb{Z}_n} \subset \mathbb{C}^d$, where $\mathbf{e}_j = (1/\sqrt{d})(\omega_n^{j\alpha_0}, \omega_n^{j\alpha_1}, \ldots, \omega_n^{j\alpha_{d-1}})^T$ is a DFT frame. In this paper, subsets $\mathscr{A} \subset \mathbb{Z}^+$ are used, such that $\operatorname{card}(\mathscr{A}) = d$, although it does not represent a DFT frame.

Lemma 1. *Let $\mathbf{W}_n \in \mathbb{C}^{d\times d}$ be a hypercomplex DFT kernel or DFT frame kernel. Then*

(1) $\mathbf{W}_n^{j+r} = \mathbf{W}_n^j \cdot \mathbf{W}_n^r$.
(2) $\mathbf{W}_n^0 = \mathbf{W}_n^n = \mathbf{I}_d$.
(3) *If $k \in \mathbb{Z}$ and $r \in \mathbb{Z}_N$, then* $\mathbf{W}_n^{nk+r} = \mathbf{W}_n^r$.
(4) *If $n = rs$, then* $\mathbf{W}_n^{rk} = \mathbf{W}_s^k$.

Proof. For hypercomplex DFT kernel, the proof of each case is similar to proof of nth roots of unity. For DFT frame kernel, $\mathbf{W}_n$ is a diagonal matrix, and then the proof of each case is straightforward. □

4. A Parallel Framework for $n = rs$

In this section, the main results of this paper are presented. Firstly, a block matrix representation of the vector-valued DFT is given. Secondly, a new mathematical framework from matrix representation of vector-valued DFT is derived, using a block matrix formalism (i.e., Theorem 2). This new result is inspired in the matrix representation of the Cooley-Tukey FFT algorithm for complex discrete-time signals, corresponding to the decomposition of the transform size n into the product of two factors r and s, which is developed in [10, 12, 13]. The result obtained in Theorem 2 is transformed in a new block matrix representation such that it contributes to analysis, design, and implementation of parallel algorithms (i.e., Corollary 3). This new result is inspired in (3). Finally, a computational complexity analysis of new algorithm is developed.

Similar to the DFT matrix representation explained in Section 2.2, there are two different approaches of representing the vector-valued DFT: as summations (see (5)) or using matrix-vector products. Both approaches allow a parallel implementation. In fact, the proof of Theorem 2 is developed using summation notation.

The vector-valued DFT can be presented as matrix-vector products. The block matrix representation of vector-valued DFT of $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{C}^d)$ is defined as $\mathscr{F}_{\mathbf{x}}^d = \mathbf{F}_n^d\mathbf{x}$, where $\mathbf{F}_n^d \in \mathbb{C}^{dn\times dn}$ such that $(\mathbf{F}_n^d)_{j,k} = \mathbf{W}_n^{-jk} \in \mathbb{C}^{d\times d}$, for $j, k \in \mathbb{Z}_n$. We clarify that the superscript d is an index, not power. In this section, a block matrix factorization of $\mathbf{F}_n^d$ is developed, and it is inspired in (3). First, a generalization of stride permutation is defined. Let $n = rs$. The block stride permutation matrix [14, 17] is defined as $\mathbf{L}_s^{n,d} \in \mathbb{C}^{dn\times dn}$ such that $\mathbf{L}_s^{n,d} = \mathbf{L}_s^n \otimes \mathbf{I}_d$, and, for each $\mathbf{x} \in \mathbb{C}^{dn}$ with n blocks $\mathbf{x}_j \in \mathbb{C}^d$, the operation $\mathbf{L}_s^{n,d}\mathbf{x}$ permutes each block of the input block $\mathbf{x}$ as $jr + k \to ks + j$, $j \in \mathbb{Z}_s$, and $k \in \mathbb{Z}_r$.

Theorem 2. *Let $n = rs$ and let $\mathbf{F}_n^d \in \mathbb{C}^{dn\times dn}$ be the block matrix of DFT for vector-valued signals. Then*

$$\mathbf{F}_n^d = \left(\mathbf{F}_s^d \otimes \mathbf{I}_r\right)\mathbf{T}_r^{n,d}\left(\mathbf{I}_s \otimes \mathbf{F}_r^d\right)\mathbf{L}_s^{n,d}, \quad (8)$$

where $\mathbf{T}_r^{n,d} = \bigoplus_{j\in\mathbb{Z}_s}\mathbf{D}_r^j$ such that $\mathbf{D}_r = \bigoplus_{k\in\mathbb{Z}_r}\mathbf{W}_n^{-k}$.

Proof. Let $\mathbf{x} \in \mathbb{C}^{dn}$, let $l_1, k_1 \in \mathbb{Z}_r$, and let $l_2, k_2 \in \mathbb{Z}_s$. The block vector $\mathbf{y} = (\mathbf{I}_s \otimes \mathbf{F}_r^d)\mathbf{L}_s^{n,d}\mathbf{x}$ is defined. Then

$$\mathbf{y}_{k_2 r+l_1} = \sum_{k_1\in\mathbb{Z}_r}\mathbf{W}_r^{-k_1 l_1}\mathbf{x}_{sk_1+k_2}. \quad (9)$$

Now, let $\mathbf{z} = \mathbf{T}_r^{n,d}\mathbf{y}$. From Lemma 1, $\mathbf{W}_r^{-k_1 l_1} = \mathbf{W}_n^{-sk_1 l_1}$; then

$$\begin{aligned}\mathbf{z}_{k_2 r+l_1} &= \mathbf{W}_n^{-k_2 l_1}\mathbf{y}_{k_2 r+l_1} = \mathbf{W}_n^{-k_2 l_1}\sum_{k_1\in\mathbb{Z}_r}\mathbf{W}_n^{-sk_1 l_1}\mathbf{x}_{sk_1+k_2}\\ &= \sum_{k_1\in\mathbb{Z}_r}\mathbf{W}_n^{-(sk_1 l_1 + k_2 l_1)}\mathbf{x}_{sk_1+k_2}.\end{aligned} \quad (10)$$

Let $\mathbf{w} = (\mathbf{F}_s^d \otimes \mathbf{I}_r)\mathbf{z}$. Then

$$\begin{aligned}\mathbf{w}_{l_1+l_2 r} &= \sum_{k_2\in\mathbb{Z}_s}\mathbf{W}_s^{-k_2 l_2}\mathbf{z}_{k_2 r+l_1}\\ &= \sum_{k_2\in\mathbb{Z}_s}\mathbf{W}_s^{-k_2 l_2}\left(\sum_{k_1\in\mathbb{Z}_r}\mathbf{W}_n^{-(sk_1 l_1 + k_2 l_1)}\mathbf{x}_{sk_1+k_2}\right)\\ &= \sum_{k_2\in\mathbb{Z}_s}\mathbf{W}_n^{-rk_2 l_2}\left(\sum_{k_1\in\mathbb{Z}_r}\mathbf{W}_n^{-(sk_1 l_1 + k_2 l_1)}\mathbf{x}_{sk_1+k_2}\right)\\ &= \sum_{k_2\in\mathbb{Z}_s}\left(\sum_{k_1\in\mathbb{Z}_r}\mathbf{W}_n^{-(rk_2 l_2 + sk_1 l_1 + k_2 l_1)}\mathbf{x}_{sk_1+k_2}\right).\end{aligned} \quad (11)$$

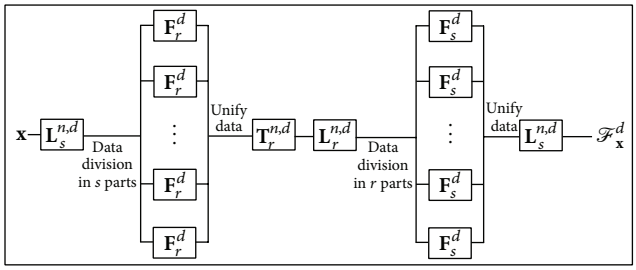

Figure 1: Parallel model of vector-valued DFT for $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{C}^d)$, $n = rs$, using a matrix representation.

But $rk_2l_2 + sk_1l_1 + k_2l_1 \equiv (k_2 + k_1s)(l_1 + l_2r) \bmod n$; then

$$\begin{aligned}\mathbf{w}_{l_1+l_2r} &= \sum_{k_2\in\mathbb{Z}_s}\left(\sum_{k_1\in\mathbb{Z}_r}\mathbf{W}_n^{-(k_2+k_1s)(l_1+l_2r)}\mathbf{x}_{sk_1+k_2}\right)\\ &= \sum_{k_2\in\mathbb{Z}_s}\sum_{k_1\in\mathbb{Z}_r}\mathbf{W}_n^{-(k_2+k_1s)(l_1+l_2r)}\mathbf{x}_{sk_1+k_2}.\end{aligned} \tag{12}$$

Let $m = sk_1 + k_2$, let $k = l_1 + l_2r$, and let $m, k \in \mathbb{Z}_n$ because $l_1, k_1 \in \mathbb{Z}_r$, $l_2k_2 \in \mathbb{Z}_s$, and $n = rs$. Then

$$\sum_{m\in\mathbb{Z}_n}\mathbf{W}_n^{-mk}\mathbf{x}_m = \mathscr{F}_{\mathbf{x}}^d(k). \tag{13}$$

□

Now, if $n = rs$, $\mathbf{A} \in \mathbb{C}^{r\times r}$, and $\mathbf{B} \in \mathbb{C}^{ds\times ds}$, the following equality [17] is obtained:

$$\mathbf{B}\otimes\mathbf{A} = \mathbf{L}_s^{n,d}(\mathbf{A}\otimes\mathbf{B})\mathbf{L}_r^{n,d}. \tag{14}$$

From Theorem 2 and (14), the following corollary presents a matrix factorization of $\mathbf{F}_n^d$ such that it permits an implementation using parallel computing.

Corollary 3. *Let $n = rs$ and let $\mathbf{F}_n^d \in \mathbb{C}^{dn\times dn}$ be the block matrix of DFT for vector-valued signals. Then*

$$\mathbf{F}_n^d = \mathbf{L}_s^{n,d}\left(\mathbf{I}_r\otimes\mathbf{F}_s^d\right)\mathbf{L}_r^{n,d}\mathbf{T}_r^{n,d}\left(\mathbf{I}_s\otimes\mathbf{F}_r^d\right)\mathbf{L}_s^{n,d}, \tag{15}$$

where $\mathbf{T}_r^{n,d}$ was defined in Theorem 2.

Algorithm 2 shows a parallel implementation of (15).

r independent processes in Steps (3)–(5), and $2s$ independent processes in Steps (6)–(8) and (12)–(14) are observed, making this approach a parallel operation. A model of Algorithm 2 is shown in Figure 1.

4.1. Computational Complexity Analysis. In this section, the computational complexity analysis of (15) is developed. First, consider the matrix operation $\mathbf{L}_s^{n,d}\mathbf{v}$. The computational complexity (CC) of $\mathbf{L}_s^{n,d}\mathbf{v}$ is $\mathcal{O}(nd)$ [8] because it is the multiplication between a block matrix in $\mathbb{C}^{dn\times dn}$ and a block vector in $\mathbb{C}^{dn}$. But the operation $\mathbf{L}_s^{n,d}\mathbf{v}$ can be implemented with a CC $\mathcal{O}(sd)$ (see, e.g., [12, 14]).

Require: $\mathbf{x} \in \mathbb{C}^{dn}$, where $n = rs$.
Ensure: $\mathbf{y} \in \mathbb{C}^{dn}$
(1) $\mathbf{y} \leftarrow \mathbf{L}_s^{n,d}\mathbf{y}$
(2) $\mathbf{A} \leftarrow \mathscr{R}_{dr,s}\{\mathbf{y}\}$
(3) **for** $m \leftarrow 0 : s-1$
(4) $\quad\mathbf{A}(:,m) \leftarrow \mathscr{F}^d_{\mathbf{A}(:,m)}$
(5) **end for**
(6) **for** $m \leftarrow 0 : s-1$
(7) $\quad\mathbf{A}(:,m) \leftarrow \mathbf{D}_r^m \cdot \mathbf{A}(:,m)$
(8) **end for**
(9) $\mathbf{y} \leftarrow \mathscr{V}\{\mathbf{A}\}$
(10) $\mathbf{y} \leftarrow \mathbf{L}_r^{n,d}\mathbf{y}$
(11) $\mathbf{A} \leftarrow \mathscr{R}_{ds,r}\{\mathbf{y}\}$
(12) **for** $m \leftarrow 0 : r-1$
(13) $\quad\mathbf{A}(:,m) \leftarrow \mathscr{F}^d_{\mathbf{A}(:,m)}$
(14) **end for**
(15) $\mathbf{y} \leftarrow \mathscr{V}\{\mathbf{A}\}$
(16) $\mathbf{y} \leftarrow \mathbf{L}_s^{n,d}\mathbf{y}$

Algorithm 2: Vector-valued DFT (parallel algorithm).

Let $\mathbf{F}_n^d \in \mathbb{C}^{dn\times dn}$ be the block matrix and vector-valued signal $\mathbf{x} \in l^2(\mathbb{Z}_n, \mathbb{C}^d)$, where $n = rs$. It is known that the CC of operation $\mathbf{y} = \mathbf{F}_n^d\mathbf{x}$ is $\mathcal{O}(n^2d^2) = \mathcal{O}(r^2s^2d^2)$. Now consider operation $\mathbf{y} = \mathbf{F}_n^d\mathbf{x}$ using (15). If we consider each matrix-vector multiplication, we obtain the following:

(1) The CC of $\mathbf{y}_1 = \mathbf{L}_s^{n,d}\mathbf{x}$ is $\mathcal{O}(sd)$.

(2) The CC of $\mathbf{y}_2 = (\mathbf{I}_s\otimes\mathbf{F}_r^d)\mathbf{y}_1$ is $\mathcal{O}(sr^2d^2)$, because it is a block diagonal matrix multiplication.

(3) The CC of $\mathbf{y}_3 = \mathbf{T}_r^{n,d}\mathbf{y}_2$ is $\mathcal{O}(nd)$, because $\mathbf{T}_r^{n,d}$ is a diagonal matrix multiplication.

(4) The CC of $\mathbf{y}_4 = \mathbf{L}_r^{n,d}\mathbf{y}_3$ is $\mathcal{O}(rd)$.

(5) The CC of $\mathbf{y}_5 = (\mathbf{I}_r\otimes\mathbf{F}_s^d)\mathbf{y}_4$ is $\mathcal{O}(rs^2d^2)$, because it is a block diagonal matrix multiplication.

(6) The CC of $\mathbf{y} = \mathbf{L}_s^{n,d}\mathbf{y}_5$ is $\mathcal{O}(sd)$.

Therefore, the CC of $\mathbf{F}_n^d\mathbf{x}$ using (15) is

$$\begin{aligned}&\mathcal{O}(sd) + \mathcal{O}\left(sr^2d^2\right) + \mathcal{O}(nd) + \mathcal{O}(rd) + \mathcal{O}\left(rs^2d^2\right)\\ &\quad + \mathcal{O}(sd) = \mathcal{O}\left(sr(r+s)d^2\right).\end{aligned} \tag{16}$$

Thus, the CC of operation $\mathbf{F}_n^d\mathbf{x}$ is $\mathcal{O}(r^2s^2d^2)$ and the CC of operation $\mathbf{F}_n^d\mathbf{x}$ using (15) is $\mathcal{O}(sr(r+s)d^2)$. The above mentioned shows the efficiency of matrix formulation in (15).

5. Implementation and Experimental Investigation

5.1. General Information. The investigations have been carried out on a computer with multicore processor. The computer consists of 4 cores with Intel Core i7-3632QM CPU processor, system clock of 2.20 GHz, and 8 GB of RAM.

The experiment develops the implementation and testing of Algorithms 1 and 2 with the hypercomplex DFT kernel and the DFT frame kernel is developed. Algorithm 1 does not use any parallel implementation, unlike Algorithm 2. A CAZAC signal in $l^2(\mathbb{Z}_n, \mathbb{C}^d)$ is used; it is generated using a Wiener CAZAC signal in $l^2(\mathbb{Z}_n)$ [9] with $d = 5$ and $n = rs$, where $n = 1024 = 32 \cdot 32$, $n = 2048 = 64 \cdot 32$, $n = 4096 = 64 \cdot 64$, $n = 8192 = 128 \cdot 64$, and $n = 16384 = 128 \cdot 128$.

The implementation of Algorithms 1 and 2 to compute the vector-valued DFT is performed using MATLAB. Algorithm 2 is computed using Parallel Computing Toolbox. MATLAB uses built-in multithreading and parallelism using MATLAB workers. Parallelism using MATLAB workers is used. We can run multiple MATLAB workers (MATLAB computational engines) on a multicore computer to execute applications in parallel with the Parallel Computing Toolbox. This approach allows more control over the parallelism compared to built-in multithreading. With programming constructs, such as parallel-for-loops (`parfor`) and batch, we write the parallel MATLAB programs of the parallel framework for the vector-valued DFT.

5.2. Results and Discussion. Let T_* be the execution time of Algorithm 1 without any parallel implementation, and let T_p be the execution time of Algorithm 2, where p is the number of cores. The value of T_p needs to be less than that of T_* for two reasons: Algorithm 2 has a parallel implementation and the matrix multiplication size is different. Algorithm 2 is computed with matrices in $\mathbb{C}^{dr\times dr}$ and $\mathbb{C}^{ds\times ds}$. Algorithm 1 is computed with matrices in $\mathbb{C}^{dn\times dn}$, where $n = rs$.

The computational performance analysis of Algorithm 2 is evaluated using the metrics speedup (or acceleration) and efficiency. The speedup is the ratio between the execution times of parallel implementations with one core and parallel implementations with two or more cores [18]. The speedup is represented by the formula $S = T_1/T_p$. The efficiency estimates how well utilized the processors are in solving the problem compared to how much effort is wasted in communication and synchronization [18]. The efficiency is determined by the ratio between the speedup and the number of processing elements, represented by the formula $E = T_1/(pT_p)$.

Table 1 shows the execution time, in seconds (s), of both algorithms. A significant reduction in the parallel execution time of the vector-valued DFT is observed. Table 1 shows that Algorithm 1 with hypercomplex kernel for a Wiener CAZAC signal in $l^2(\mathbb{Z}_{8192}, \mathbb{C}^5)$ produces a time of serial execution $T_* = 13408$ s. Using Algorithm 2, however, we obtain $T_1 = 106.7$ (0.80% of T_*), $T_2 = 80.44$ s (0.60% of T_*), $T_3 = 57.35$ s (0.43% of T_*), and $T_4 = 32.67$ s (0.24% of T_*). This result shows the advantage of using multicore processors and a parallel computing environment to minimize the high execution time in the vector-valued DFT. This is because parallel computing is a form of computation in which many calculations are carried out simultaneously [19, 20], operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently, and minimize the execution time [20, 21]. The difference between T_* and T_p is because T_p is computed with matrices in $\mathbb{C}^{dr\times dr}$ and $\mathbb{C}^{ds\times ds}$. Algorithm 1 is computed with matrices in $\mathbb{C}^{dn\times dn}$, where $n = rs$.

Table 1: Computing time of Algorithms 1 and 2 (in seconds).

Kernel	p	n				
		1024	2048	4096	8192	16384
Hypercomplex DFT	$*$	36.80	163.9	853.5	13408	+15000
	1	3.142	8.950	18.78	106.7	263.4
	2	2.363	4.276	17.25	80.44	180.9
	3	1.695	3.222	12.75	57.35	154.7
	4	0.983	2.966	6.481	32.67	82.65
DFT frame	$*$	40.50	173.1	881.0	13913	+15000
	1	2.438	9.749	20.97	95.29	251.6
	2	1.911	5.798	16.38	58.72	179.5
	3	1.531	5.126	12.33	57.40	151.4
	4	1.199	2.329	5.366	31.46	73.42

Test signal in $l^2(\mathbb{Z}_n, \mathbb{C}^d)$, where $n = rs$ and $d = 5$.
$p = *$ is time execution of Algorithm 1.
$p > 0$ is the number of cores.

Table 2: Speedup of Algorithm 2.

Kernel	p	n				
		1024	2048	4096	8192	16384
Hypercomplex DFT	2	1.333	2.093	1.089	1.326	1.456
	3	1.853	2.778	1.473	1.860	1.703
	4	3.196	3.017	2.987	3.265	3.187
DFT frame	2	1.275	1.509	1.281	1.623	1.402
	3	1.592	1.707	1.701	1.660	1.661
	4	2.033	3.757	3.901	3.029	3.426

Test signal in $l^2(\mathbb{Z}_n, \mathbb{C}^d)$, where $n = rs$ and $d = 5$.
p is the number of cores.

Table 3: Efficiency of Algorithm 2.

Kernel	p	n				
		1024	2048	4096	8192	16384
Hypercomplex DFT	2	0.665	1.046	0.544	0.663	0.728
	3	0.463	0.694	0.368	0.465	0.426
	4	0.400	0.377	0.362	0.408	0.398
DFT frame	2	0.637	0.755	0.645	0.811	0.701
	3	0.530	0.427	0.425	0.415	0.415
	4	0.508	0.470	0.489	0.379	0.428

Test signal in $l^2(\mathbb{Z}_n, \mathbb{C}^d)$, where $n = rs$ and $d = 5$.
p is the number of cores.

Table 2 represents the speedup of Algorithm 2. The acceleration of the vector-valued DFT increases when p increases regardless of the value of n. The results show that, using the proposed parallel implementation with p cores, where $p = 2, 3, 4$, the speedup to compute the vector-valued DFT of a Wiener CAZAC signal is 1.09, 1.47, and 2.99, respectively. These results imply that, to get the highest speedup, one should prefer the approach with four cores.

Table 3 represents efficiency of Algorithm 2. The information in this table shows that a good efficiency (greater

than 65%) is reached with $p = 2$. But the efficiency of the vector-valued DFT decreases (until 36%) when p increases regardless of the value of n. It is attributed to a decrease in the share of simultaneous computation of the partial vector-valued DFT in Algorithm 2 (steps (3)–(5) and (12)–(14)), which is responsible for the main effect. The results obtained in Table 3 imply that, to get a better efficiency, one should prefer the approach with two cores, because we obtain the highest efficiency.

6. Conclusion

This work presented a parallel framework of vector-valued DFT for vector-valued discrete-time signals. This mathematical framework was inspired in the matrix representation of the Cooley-Tukey FFT algorithm for complex discrete-time signals, corresponding to the decomposition of the transform size n into the product of two factors r and s, which is developed in [10, 12]. It was expressed in (15) and Algorithm 2. This parallel framework was performed in terms of a matrix representation using a set of block matrix operations: Kronecker product, direct sum, stride permutation, vec operator, and vec inverse operator. These operations contributed to analysis, design, and implementation in parallel. Two kernels are used in the vector-valued DFT: hypercomplex DFT kernel and DFT frame kernel.

The experimental investigation indicated there are profit using MATLAB with the Parallel Computing Toolbox in a computer with multicore processors. First, there was advantage to use multicore processors and a parallel computing environment to minimize the high execution time (with hypercomplex DFT kernel, we obtained $T_* = 13408$ s, $T_1 = 106.7$, $T_2 = 80.44$ s, $T_3 = 57.35$ s, and $T_4 = 32.67$ s). Second, speedup increased when p increased regardless of the value of n, and a good efficiency too was obtained when $p = 2$ (above 65%).

As future work, we would like to extend the proposed parallel framework to vector-valued discrete-time signals in $l^2(\mathbb{Z}_n, \mathbb{C}^d)$, where $n = 2^k$, using the idea of Pease algorithm for complex discrete-time signals [22]. Additionally, we would like to take advantage of more design tradeoffs of different approaches besides what have been shown in this paper, for example, the approach developed in [23].

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by Vicerrectoría de Investigación y Extensión of Instituto Tecnológico de Costa Rica.

References

[1] B. Wegmann and C. Zetzsche, "Feature-specific vector quantization of images," *IEEE Transactions on Image Processing*, vol. 5, no. 2, pp. 274–288, 1996.

[2] J. Huang and B.-Q. Lv, "A feasible algorithm for designing biorthogonal bivariate vector-valued finitely supported wavelets," *Physics Procedia*, vol. 25, pp. 1507–1514, 2012.

[3] W. Li, "Vector transform and image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 1, no. 4, pp. 297–307, 1991.

[4] X.-G. Xia and B. W. Suter, "Multirate filter banks with block sampling," *IEEE Transactions on Signal Processing*, vol. 44, no. 3, pp. 484–496, 1996.

[5] S. A. Avdonin and S. A. Ivanov, "Sampling and interpolation problems for vector valued signals in the Paley-Wiener spaces," *IEEE Transactions on Signal Processing*, vol. 56, no. 11, pp. 5435–5441, 2008.

[6] M. Vehkapera, S. Chatterjee, and M. Skoglund, "Analysis of MMSE estimation for compressive sensing of block sparse signals," in *Proceedings of the IEEE Information Theory Workshop (ITW '11)*, vol. 1, pp. 553–557, IEEE, Paraty, Brazil, October 2011.

[7] J. J. Benedetto and J. J. Donatelli, "Frames and a vector-valued ambiguity function," in *Proceedings of the 42nd Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 8–12, IEEE, Pacific Grove, Calif, USA, October 2008.

[8] S. J. Sangwine and T. A. Ell, "Complex and hypercomplex discrete Fourier transforms based on matrix exponential form of Euler's formula," *Applied Mathematics and Computation*, vol. 219, no. 2, pp. 644–655, 2012.

[9] J. J. Benedetto and J. J. Donatelli, "Ambiguity function and frame-theoretic properties of periodic zero-autocorrelation waveforms," *IEEE Journal on Selected Topics in Signal Processing*, vol. 1, no. 1, pp. 6–20, 2007.

[10] J. R. Johnson, R. W. Johnson, D. Rodriguez, and R. Tolimieri, "A methodology for designing, modifying, and implementing Fourier transform algorithms on various architectures," *Circuits, Systems, and Signal Processing*, vol. 9, no. 4, pp. 449–500, 1990.

[11] D. Rodriguez, J. Seguel, and E. Cruz, "Algebraic methods for the analysis and design of time-frequency signal processing algorithms," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '93)*, vol. 1, pp. 196–199, IEEE, Chicago, Ill, USA, May 1993.

[12] R. Tolimieri, M. An, and C. Lu, *Algorithms for Discrete Fourier Transform and Convolution*, Signal Processing and Digital Filtering, Springer, Berlin, Germany, 1997.

[13] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, Frontiers in Applied Mathematics, SIAM, 2012.

[14] F. Franchetti, M. uschel, Y. Voronenko, S. Chellappa, and J. M. F. Moura, "Discrete Fourier transform on multicore," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 90–102, 2009.

[15] A. Saberi, A. Stoorvogel, and P. Sannuti, *Internal and External Stabilization of Linear Systems with Constraints*, Systems & Control: Foundations & Applications, Springer, Berlin, Germany, 2012.

[16] A. Shirazinia, S. Chatterjee, and M. Skoglund, "Performance bounds for vector quantized compressive sensing," in *Proceedings of the International Symposium on Information Theory and Its Applications (ISITA '12)*, pp. 289–293, October 2012.

[17] R. Tolimieri, M. An, C. Lü, and C. Burrus, *Mathematics of Multidimensional Fourier Transform Algorithms*, Signal Processing and Digital Filtering, Springer, Berlin, Germany, 1997.

[18] M. D. McCool, A. D. Robison, and J. Reinders, *Structured Parallel Programming: Patterns for Efficient Computation*, Morgan Kaufmann Publishers, Elsevier, 2012.

[19] G. S. Almasi and A. Gottlieb, *Highly Parallel Computing*, Benjamin-Cummings Publishing Company, 1989.

[20] R. Trobec, M. Vajteric, and P. Zinterhof, *Parallel Computing: Numerics, Applications, and Trends*, Springer, 2009.

[21] M. O. Tokhi, M. A. Hossain, and M. H. Shaheed, *Parallel Computing for Real-Time Signal Processing and Control*, Springer, Berlin, Germany, 2003.

[22] M. C. Pease, "An adaptation of the fast fourier transform for parallel processing," *Journal of the ACM*, vol. 15, no. 2, pp. 252–264.

[23] Z. Qiuling, B. Akin, H. E. Sumbul et al., "A 3D-stacked logic-in-memory accelerator for application-specific data intensive computing," in *Proceedings of the IEEE International 3D Systems Integration Conference*, pp. 1–7, October 2013.