Chapter 1

Jacket Matrix Based Recursive Fourier Analysis and Its Applications

Daechul Park and Moon Ho Lee

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/59353

1. Introduction

The last decade based on orthogonal transform has been seen a quiet revolution in digital video technology as in Moving Picture Experts Group (MPEG)-4, H.264, and high efficiency video coding (HEVC) [1–7]. The discrete cosine transform (DCT)-II is popular compression structures for MPEG-4, H.264, and HEVC, and is accepted as the best suboptimal transformation since its performance is very close to that of the statistically optimal Karhunen-Loeve transform (KLT) [1-5].

The discrete signal processing based on the discrete Fourier transform (DFT) is popular in wide range of applications depending on specific targets: orthogonal frequency division multiplexing (OFDM) wireless mobile communication systems in 3GPP-LTE [3], mobile worldwide interoperability for microwave access (WiMAX), international mobile telecommunications-advanced (IMT-Advanced), broadcasting related applications such as digital audio broadcasting (DAB), digital video broadcasting (DVB), digital multimedia broadcasting (DMB)) based on DFT. Furthermore, the Haar-based wavelet transform (HWT) is also very useful in the joint photographic experts group committee in 2000 (JPEG-2000) standard [2], [8]. Thus, different applications require different types of unitary matrices and their decompositions. From this reason, in this book chapter we will propose a unified hybrid algorithm which can be used in the mentioned several applications in different purposes.

Compared with the conventional individual matrix decompositions, our main contributions are summarized as follows:

- We propose the diagonal sparse matrix factorization for a unified hybrid algorithm based on the properties of the Jacket matrix [9], [10] and the recursive decomposition of the sparse matrix. It has been shown that this matrix decomposition is useful in developing the fast algorithms [11]. Individual DCT-II [1–3], [6], [7], [12], DST-II [4], [6], [7], [13], DFT [3], [5], [14], and HWT [8] matrices can be decomposed to one orthogonal character matrix and a corresponding special sparse matrix. The inverse of the sparse matrix can be easily obtained from the property of the block (element)-wise inverse Jacket matrix. However, there have been no previous works in the development of the common matrix decomposition supporting these transforms.
- We propose a new unified hybrid algorithm which can be used in the multimedia applications, wireless communication systems, and broadcasting systems at almost the same computational complexity as those of the conventional unitary matrix decompositions as summarized in Table 1 and 2. Compared with the existing unitary matrix decompositions, the proposed hybrid algorithm can be even used to the heterogeneous systems with hybrid multimedia terminals being serviced with different applications. The block (element)-wise diagonal decompositions of DCT-II, DST-II, DFT and DWT have a similar pattern as Cooley-Tukey's regular butterfly structures. Moreover, this unified hybrid algorithm can be also applied to the wireless communication terminals requiring a multiuser multiple input-multiple output (MIMO) SVD block diagonalization systems [15], [11,19], [22] and diagonal channels interference alignment management in macro/femto cell coexisting networks [16]. In [15-16, 19, 22- 23], a block-diagonalized matrix can be applied to wireless communications MIMO downlink channel.

In Section 2, we present recursive factorization algorithms of DCT-II, DST-II, and DFT matrix for fast computation. In Section 3, hybrid architecture is proposed for fast computations of DCT-II, DST-II, and DFT matrices. Also numerical simulations follow. The conclusion is given in Section 4.

Notation: The superscript $(\cdot)^T$ denotes transposition; $\boldsymbol{I}_N$ denotes the $N \times N$ identity matrix; 0 denotes an all-zero matrix of appropriate dimensions; $C_l^i = \cos(i\pi / l)$; $S_l^i = \sin(i\pi / l)$; $\boldsymbol{W} = e^{-\frac{j2\pi}{N}}$; $\otimes$ and $\oplus$, respectively, denote the Kronecker product and the direct sum.

2. Jacket matrix based recursive decompositions of Fourier matrix

2.1. Recursive decomposition of DCT-II

Definition 1*: Let* $J_N = \{a_{i,j}\}$ *be a matrix, then it is called the Jacket matrix when* $J_N^{-1} = \frac{1}{N}\{(a_{i,j})^{-1}\}^T$.

That is, the inverse of the Jacket matrix can be determined by its element-wise inverse [9-11]. The row permutation matrix, $\boldsymbol{P}_N$ is defined by

$$\boldsymbol{P}_2 = \boldsymbol{I}_2 \textbf{ and } \boldsymbol{P}_N = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{1}$$

where $\boldsymbol{P}_N$ elements are determined by the following relation:

$$\begin{cases} p_{i,j}=1, \quad if \quad i=2j, \quad 0 \le j \le \frac{N}{2}-1, \\ p_{i,j}=1, \quad if \quad i=(2j+1)\bmod N, \quad \frac{N}{2} \le j \le N-1, \\ p_{i,j}=0, \qquad others. \end{cases}$$

The block column permutation matrix, Q_N is defined by

$$\boldsymbol{Q}_N = \boldsymbol{I}_2 \textbf{ and } \boldsymbol{Q}_N = \begin{bmatrix} \boldsymbol{I}_{N/4} & \boldsymbol{0}_{N/2} \\ \boldsymbol{0}_{N/2} & \bar{\boldsymbol{I}}_{N/4} \end{bmatrix}, \ N \ge 4. \tag{2}$$

where $\bar{\boldsymbol{I}}_{N/2}$ denotes reversed identity matrix. Note that $Q_N^{-1}=Q_N$ and $\boldsymbol{P}_N^{-1} \neq \boldsymbol{P}_N$, whereas $Q_N^{-1}=Q_N^T$ and $\boldsymbol{P}_N^{-1}=\boldsymbol{P}_N^T$.

Proposition 1: *With the use of the Kronecker product and Hadamard matrices, a higher order block-wise inverse Jacket matrix (BIJM) can be recursively obtained by*

$$\boldsymbol{J}_{2N} = \boldsymbol{J}_N \otimes \boldsymbol{H}_2, \ N \ge 2 \tag{3}$$

then

$$\boldsymbol{J}_{2N}^{-1} = \frac{1}{N}\boldsymbol{J}_{2N}^T \tag{4}$$

where the lowest order Hadamard matrix is defined by $H_2=\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$

Proof: *A proof of this proposition is given in Appendix 6.A.*

Note that since the BIJM requires a matrix transposition and then normalization by its size, a class of transforms can be easily inverted as follows:

$$\boldsymbol{Y}_{2N} = \boldsymbol{J}_{2N}\boldsymbol{X}_{2N}, \textbf{ and } \boldsymbol{X}_{2N} = \boldsymbol{J}_{2N}^{-1}\boldsymbol{Y}_{2N} = \frac{1}{N}\boldsymbol{J}_{2N}^{T}\boldsymbol{Y}_{2N}. \tag{5}$$

Due to a simple operation of the BIJM, we can reduce the complexity order as the matrix size increases. In the following, we shall use this property of the BIJM in developing a hybrid diagonal block-wise transform.

According to [1-4] and [7], the DCT-II matrix is defined as follows:

$$\boldsymbol{C}_N = \sqrt{\frac{2}{N}}\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ C_{4N}^{2k_0\Phi_0} & C_{4N}^{2k_0\Phi_1} & \cdots & C_{4N}^{2k_0\Phi_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ C_{4N}^{2k_{N-2}\Phi_0} & C_{4N}^{2k_{N-2}\Phi_1} & \cdots & C_{4N}^{2k_{N-2}\Phi_{N-1}} \end{bmatrix} = \sqrt{\frac{2}{N}}\mathsf{X}_N \tag{6}$$

where $\Phi_i = 2i + 1$ and $k_i = i + 1$. We first define a permuted DCT-II matrix $\widetilde{C}_N = \boldsymbol{P}_N^{-1} C_N Q_N^{-1} = \sqrt{\frac{2}{N}}\boldsymbol{P}_N^{-1}\mathsf{X}_N Q_N^{-1}$. We can readily show that the matrix X_N can be constructed recursively as follows:

$$\mathsf{X}_N = \boldsymbol{P}_N\begin{bmatrix} \mathsf{X}_{N/2} & \mathsf{X}_{N/2} \\ \boldsymbol{B}_{N/2} & -\boldsymbol{B}_{N/2} \end{bmatrix}\boldsymbol{Q}_N = \boldsymbol{P}_N\begin{bmatrix} \mathsf{X}_{N/2} & 0 \\ 0 & \boldsymbol{B}_{N/2} \end{bmatrix}\begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix}\boldsymbol{Q}_N. \tag{7}$$

Here, the matrix B_N in (7) is given as:

$$\boldsymbol{B}_N = \left\{ B_N(m,n) = C_{4N}^{f(m,n)} \right\} \tag{8}$$

where $f(m,\ 1) = 2m - 1$ and $f(m,\ n+1) = f(m,\ n) + 2f(m,\ 1)$ for $m,\ n \in \{1,\ 2,\ \ldots,\ N/2\}$. For example, the matrix B_4 is given by

$$\boldsymbol{B}_4 = \begin{bmatrix} C_{16}^1 & C_{16}^3 & C_{16}^5 & C_{16}^7 \\ C_{16}^3 & -C_{16}^7 & -C_{16}^1 & -C_{16}^5 \\ C_{16}^5 & -C_{16}^1 & C_{16}^7 & C_{16}^3 \\ C_{16}^7 & -C_{16}^5 & C_{16}^3 & -C_{16}^1 \end{bmatrix}. \tag{9}$$

Since $\mathsf{X}_{N/2}^{-1} = \frac{4}{N}\mathsf{X}_{N/2}^{T}$ and $B_{N/2}^{-1} = \frac{4}{N}B_{N/2}^{T}$, the matrix decomposition in (7) is the form of the matrix product of diagonal block-wise inverse Jacket and Hadamard matrices. The matrix $B_{N/2}$ is recursively factorized using Lemma 1.

Lemma 1*:The matrix B_N can be decomposed as:*

$$\boldsymbol{B}_N = \boldsymbol{L}_N X_N \boldsymbol{D}_N \tag{10}$$

where a lower triangular matrix $\mathbf{L}_N$ *is defined by* $\mathbf{L}_N = \{\mathbf{L}_N(m,\ n)\}$ *with elements*

$$L_N(m,n) = \begin{cases} \sqrt{2}(-1)^{m-1}, \forall m \text{ and } n = 1 \\ 2(-1)^{m-1}(-1)^{n-1},\ m \le n \\ 0,\ m > n \end{cases} \tag{11}$$

and a diagonal matrix $\mathbf{D}_N$ *is defined by* $\mathbf{D}_N = diag\{C_{4N}^{\Phi_0},\ C_{4N}^{\Phi_1},\ \ldots,\ C_{4N}^{\Phi_{N-1}}\}$.

Proof:*A proof of this Lemma is provided in Appendix 6.B.*

Using (10), we first rewrite (7) as

$$\begin{aligned} X_N &= \boldsymbol{P}_N \begin{bmatrix} X_{N/2} & 0 \\ 0 & \boldsymbol{L}_{N/2} X_{N/2} \boldsymbol{D}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix} \boldsymbol{Q}_N \\ &= \boldsymbol{P}_N \begin{bmatrix} \boldsymbol{I}_{N/2} & 0 \\ 0 & \boldsymbol{L}_{N/2} \end{bmatrix} \left[\boldsymbol{I}_2 \otimes X_{N/2} \right] \begin{bmatrix} \boldsymbol{I}_{N/2} & 0 \\ 0 & \boldsymbol{D}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix} \boldsymbol{Q}_N \end{aligned} \tag{12}$$

which can be evaluated recursively as follows:

$$\begin{aligned} X_N = \boldsymbol{P}_N \begin{bmatrix} \boldsymbol{I}_{N/2} & 0 \\ 0 & \boldsymbol{L}_{N/2} \end{bmatrix} \times \left[\boldsymbol{I}_2 \otimes \underbrace{\left[\cdots \left[\boldsymbol{I}_2 \otimes \underbrace{\left[\boldsymbol{P}_4 \begin{bmatrix} \boldsymbol{I}_2 & 0 \\ 0 & \boldsymbol{L}_2 \end{bmatrix} \left[\boldsymbol{I}_2 \otimes X_2 \right] \begin{bmatrix} \boldsymbol{I}_2 & 0 \\ 0 & \boldsymbol{D}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{I}_2 \\ \boldsymbol{I}_2 & \boldsymbol{-I}_2 \end{bmatrix} \boldsymbol{Q}_4 \right]}_{X_4} \right] \cdots \right]}_{X_{N/2}} \right] \\ \times \begin{bmatrix} \boldsymbol{I}_{N/2} & 0 \\ 0 & \boldsymbol{D}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix} \boldsymbol{Q}_N. \end{aligned} \tag{13}$$

Note that in (13) a 2×2 Hadamard matrix is defined by $X_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. Also, applying the Kronecker product of $\boldsymbol{I}_2$ and X_4, X_8 can be obtained. Keep applying the Kronecker product of $\boldsymbol{I}_2$ and $X_{N/2}$, the final equivalent form of X_N is obtained. Thus, the proposed systematic decomposition is based on the Jacket and Hadamard matrices.

In [17], the author proposed a recursive decimation-in-frequency algorithm, where the same decomposition specified in (10) was used. However, due to using a different permutation matrix, a different recursive form was obtained. Different recursive decomposition was proposed in [18]. Four different matrices, such as the first matrix, the last matrix, the odd numbered matrix, and the even number matrix, were proposed. Compared to the decomposition in [18], the proposed decomposition is seen to be more systematic and requires less numbers of additions and multiplications. We show a complexity comparison among the proposed decomposition and other methods in Table 1-2.

Reference number	**Conventional methods**		**Proposed**	
	Addition	**Multiplication**	**Addition**	**Multiplication**
W. H. Chen at el [18] DCT-II	$3N/2(\log_2 N-1)+2$	$N\log_2 N-(3N/2)+4$	$N\log_2 N$	$N/2(\log_2 N+1)$
Z. Wang[13] DST-II	$N\left(\frac{7}{4}\log_2(N)-2\right)+3$	$N\left(\frac{3}{4}\log_2(N)-1\right)+3$	$N\log_2 N$	$N/2(\log_2 N+1)$
Cooley and Tukey [21] DFT	$N\log_2 N$	$(N/2)\log_2 N$	$N\log_2 N$	$(N/2)\log_2 N$

Table 1. The comparison of computation complexity of conventional independent the DCT-II, DST-II, DFT, and hybrid DCT-II/DST-II/DFT

	Matrix Size, N	**Conventional**		**Proposed**	
		Addition	**Multiplication**	**Addition**	**Multiplication**
DCT-II	4	8	6	8	6
	8	26	16	24	16
	16	74	44	64	40
	32	194	116	160	96
	64	482	292	384	224
	128	1154	708	896	512
	256	2690	1668	2048	1152
DST-II	4	9	5	8	6
	8	29	13	24	16
	16	83	35	64	40
	32	219	91	160	96
	64	547	227	384	224
	128	1315	547	896	512
	256	3075	1283	2048	1152
DFT	4	8	4	8	4
	8	24	12	24	12
	16	64	32	64	32

Matrix Size, N	Conventional		Proposed	
	Addition	Multiplication	Addition	Multiplication
32	160	80	160	80
64	384	192	384	192
128	896	448	896	448
256	2048	1024	2048	1024

Table 2. Computational Complexity: DCT-II/DST-II/DFT

Applying (13), we can readily compute $C_N = \sqrt{\frac{2}{N}}\mathbb{X}_N$. The inverse of C_N can be obtained from the properties of the sparse Jacket matrix inverse:

$$(\boldsymbol{C}_N)^{-1} = \sqrt{\frac{N}{2}}(\boldsymbol{Q}_N)^{-1}\begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix}^{-1}\begin{bmatrix} \mathbb{X}_{N/2}^{-1} & 0 \\ 0 & \boldsymbol{B}_{N/2}^{-1} \end{bmatrix}\boldsymbol{P}_N^{-1}$$
$$= \sqrt{\frac{N}{2}}\boldsymbol{Q}_N\begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix}\begin{bmatrix} \mathbb{X}_{N/2}^{T} & 0 \\ 0 & \boldsymbol{B}_{N/2}^{T} \end{bmatrix}\boldsymbol{P}_N^{T}. \tag{14}$$

The corresponding butterfly data flow diagram of C_N is given in Fig. 1.

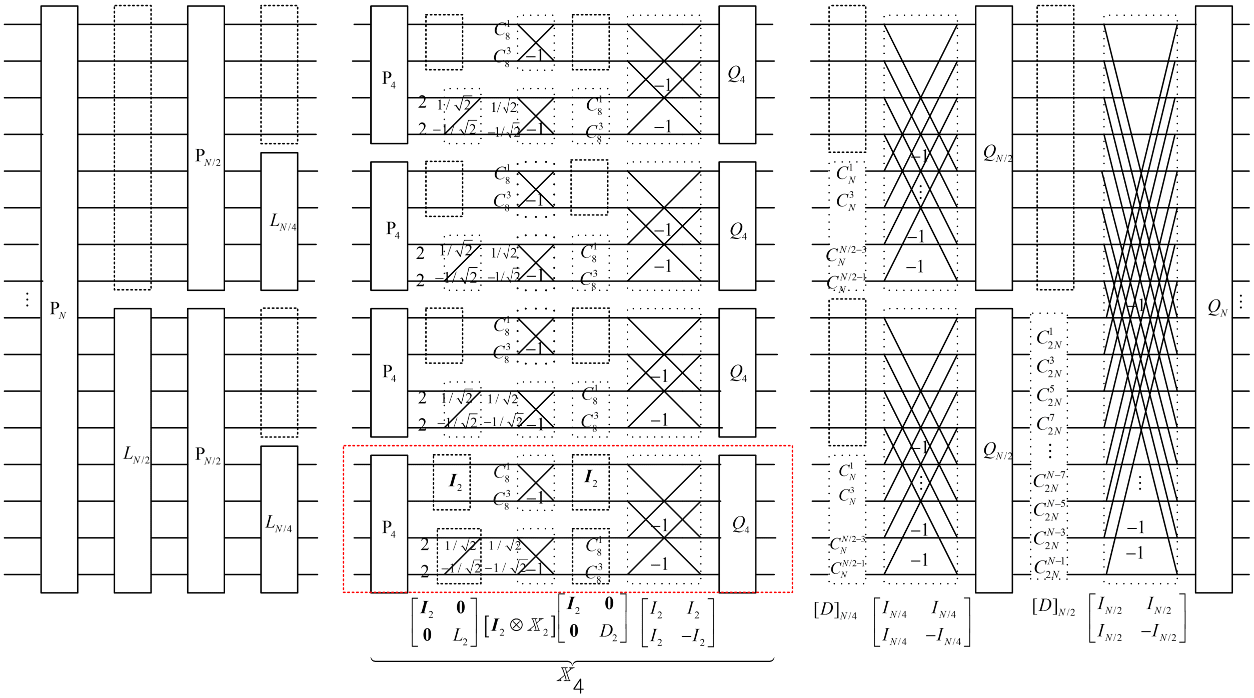


Figure 1. Regular systematic butterfly data flow of DCT-II.

2.2. Recursive decomposition of the DST-II

The DST-II matrix [1-4] and [7] can be expressed as follows:

$$\boldsymbol{S}_N = \sqrt{\frac{2}{N}} \begin{bmatrix} S_{4N}^{2k_0\Phi_0} & S_{4N}^{2k_0\Phi_1} & \cdots & S_{4N}^{2k_0\Phi_{N-1}} \\ S_{4N}^{2k_1\Phi_0} & S_{4N}^{2k_1\Phi_1} & \cdots & S_{4N}^{2k_1\Phi_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ S_{4N}^{2k_{N-2}\Phi_0} & S_{4N}^{2k_{N-2}\Phi_1} & \cdots & S_{4N}^{2k_{N-2}\Phi_{N-1}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \cdots & -\frac{1}{\sqrt{2}} \end{bmatrix} = \sqrt{\frac{2}{N}} \mathsf{Y}_N. \tag{15}$$

Similar to the procedure we have used in the DCT-II matrix, we first define the permuted DST-II matrix, $\tilde{S}_N$ as follows:

$$\tilde{\boldsymbol{S}}_N = \boldsymbol{P}_N^{-1} \boldsymbol{S}_N \boldsymbol{Q}_N^{-1} = \sqrt{\frac{2}{N}} \boldsymbol{P}_N^{-1} \mathsf{Y}_N \boldsymbol{Q}_N^{-1}. \tag{16}$$

From (16), we can have a recursive form for Y_N as

$$\mathsf{Y}_N = \boldsymbol{P}_N \begin{bmatrix} \boldsymbol{A}_{N/2} & 0 \\ 0 & \mathsf{Y}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \text{-}\boldsymbol{I}_{N/2} \end{bmatrix} \boldsymbol{Q}_N \tag{17}$$

where the submatrix A_N can be calculated by

$$\boldsymbol{A}_N = \boldsymbol{U}_N \mathsf{Y}_N \boldsymbol{D}_N \tag{18}$$

where $\boldsymbol{U}_N$ and $\boldsymbol{D}_N$ are, respectively, upper triangular and diagonal matrices. The upper triangular matrix $\boldsymbol{U}_N = \{\boldsymbol{U}_N(m, n)\}$ is defined as follows:

$$U_N(m,n) = \begin{cases} \sqrt{2}(-1)^{m-1}, \forall m \text{ and } n = N \\ 2(-1)^{m-1}(-1)^{n-1}, \ m \geq n \\ 0, \ m < n \end{cases} \tag{19}$$

whereas the matrix $\boldsymbol{D}_N$ is defined as before in (10). The derivation of (18) is given in Appendix C. Recursively applying (18) in (17), Recursively applying (18) in (17), we can find that

$$\begin{aligned} \mathrm{Y}_N &= \boldsymbol{P}_N \begin{bmatrix} \boldsymbol{A}_{N/2} & 0 \\ 0 & \mathrm{Y}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix} \boldsymbol{Q}_N = \boldsymbol{P}_N \begin{bmatrix} \boldsymbol{U}_{N/2}\mathrm{Y}_{N/2}\boldsymbol{D}_{N/2} & 0 \\ 0 & \mathrm{Y}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix} \boldsymbol{Q}_N \\ &= \boldsymbol{P}_N \begin{bmatrix} \boldsymbol{U}_{N/2} & 0 \\ 0 & \boldsymbol{I}_{N/2} \end{bmatrix} \left[\boldsymbol{I}_2 \otimes \mathrm{Y}_{N/2} \right] \begin{bmatrix} \boldsymbol{D}_{N/2} & 0 \\ 0 & \boldsymbol{I}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix} \boldsymbol{Q}_N. \end{aligned} \tag{20}$$

Further applying (17) to the Kronecker product $[\boldsymbol{I}_2 \otimes \mathrm{Y}_{N/2}]$, the following general recursive form for DST-II matrix can be obtained as:

$$\begin{aligned} \boldsymbol{Y}_N = \sqrt{\frac{2}{N}} \boldsymbol{P}_N \begin{bmatrix} \boldsymbol{U}_{N/2} & 0 \\ 0 & \boldsymbol{I}_{N/2} \end{bmatrix} \times & \left[\boldsymbol{I}_2 \otimes \underbrace{\left[\cdots \left[\boldsymbol{I}_2 \otimes \underbrace{\left[\boldsymbol{P}_4 \begin{bmatrix} \boldsymbol{U}_2 & 0 \\ 0 & \boldsymbol{I}_2 \end{bmatrix} \left[\boldsymbol{I}_2 \otimes \mathrm{Y}_2 \right] \begin{bmatrix} \boldsymbol{D}_2 & 0 \\ 0 & \boldsymbol{I}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{I}_2 \\ \boldsymbol{I}_2 & \boldsymbol{-I}_2 \end{bmatrix} \boldsymbol{Q}_4 \right]}_{\mathrm{Y}_4} \right] \cdots \right]}_{\mathrm{Y}_{N/2}} \right] \\ & \times \begin{bmatrix} \boldsymbol{D}_{N/2} & 0 \\ 0 & \boldsymbol{I}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix} \boldsymbol{Q}_N. \end{aligned} \tag{21}$$

Note that if we compare (21) and (13), a similarity can be found in the proposed matrix decompositions. That is, starting from the common lowest order $\mathrm{Y}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, the discrete sine kernel matrix is recursively constructed. Especially, applying the relationship of $\boldsymbol{U}_N = \tilde{\boldsymbol{I}}_N \widetilde{\boldsymbol{L}}_N \tilde{\boldsymbol{I}}_N$, where $\tilde{\boldsymbol{I}}_N = \begin{bmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & 0 \\ \vdots & ⋰ & \vdots & \vdots \\ 1 & \cdots & 0 & 0 \end{bmatrix}$ denotes the opposite diagonal identity matrix, the butterfly data flow of the DST-II matrix can be obtained from the corresponding that of the proposed DCT-II decomposition. The butterfly data flow graph of the DST-II matrix is shown in Fig. 2.

Now utilizing the properties of the BIJM, we can first obtain

$$\begin{bmatrix} \boldsymbol{A}_{N/2} & 0 \\ 0 & \mathrm{Y}_{N/2} \end{bmatrix}^{-1} = \frac{2}{N} \begin{bmatrix} \boldsymbol{A}_{N/2}^T & 0 \\ 0 & \mathrm{Y}_{N/2}^T \end{bmatrix} \tag{22}$$

such that the inverse of the matrix S_N is given by

$$\boldsymbol{S}_N^{-1} = \sqrt{\frac{N}{2}} \boldsymbol{Q}_N \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \boldsymbol{-I}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{A}_{N/2}^T & 0 \\ 0 & \mathrm{Y}_{N/2}^T \end{bmatrix} \boldsymbol{P}_N^T. \tag{23}$$

Note that applying again the properties of the BIJM and (18), a recursive form of the inverse DST-II can be easily obtained.

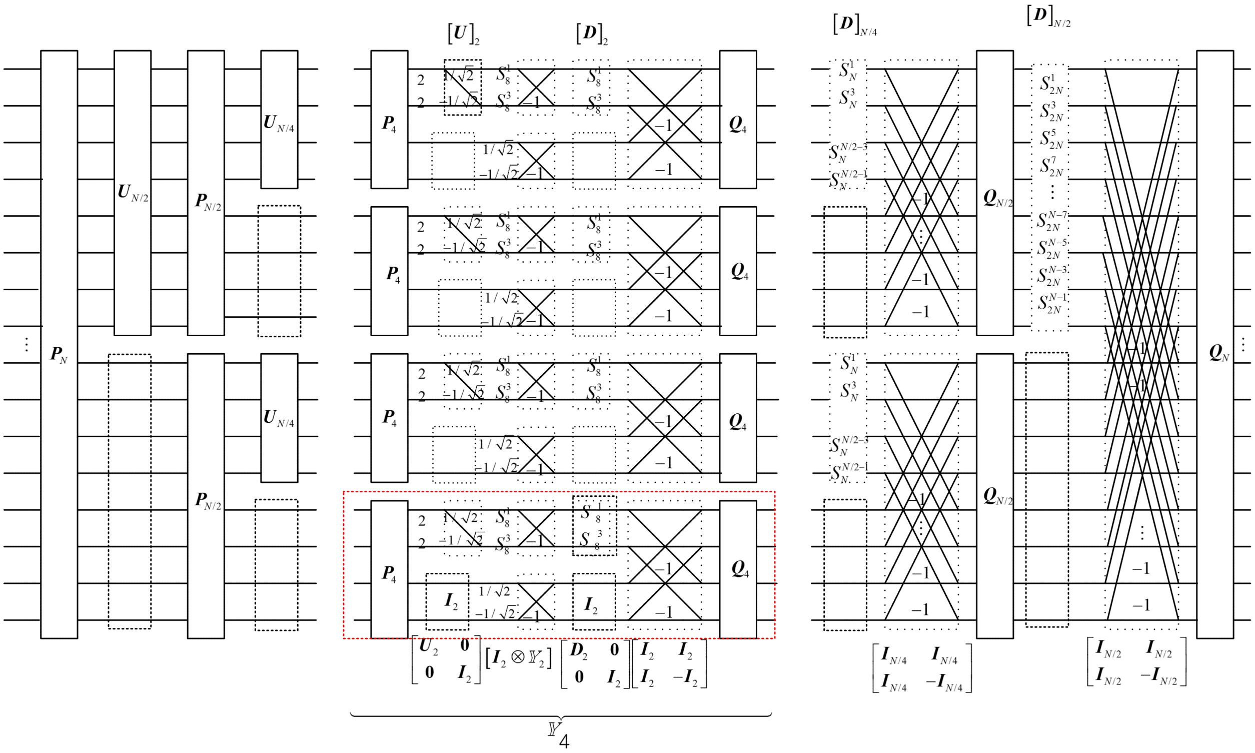


Figure 2. Regular systematic butterfly data flow of DST-II.

2.3. Recursive decomposition of DFT

The DFT is a Fourier representation of a given sequence $\{\boldsymbol{x}(n)\}$,

$$\boldsymbol{X}(n)=\sum_{m=0}^{N-1} x(m)\boldsymbol{W}^{nm},\ 0\le n\le N-1. \tag{24}$$

where $\boldsymbol{W}=e^{-j2\pi/N}$. The N -point DFT matrix can be denoted by $\boldsymbol{F}_N=\{\boldsymbol{W}^{nm}\}$. The $N\times N$ Sylvester Hadamard matrix is denoted by H_N. The Sylvester Hadamard matrix is generated by the successive Kronecker products:

$$\boldsymbol{H}_N=\boldsymbol{H}_2\otimes\boldsymbol{H}_{N/2} \tag{25}$$

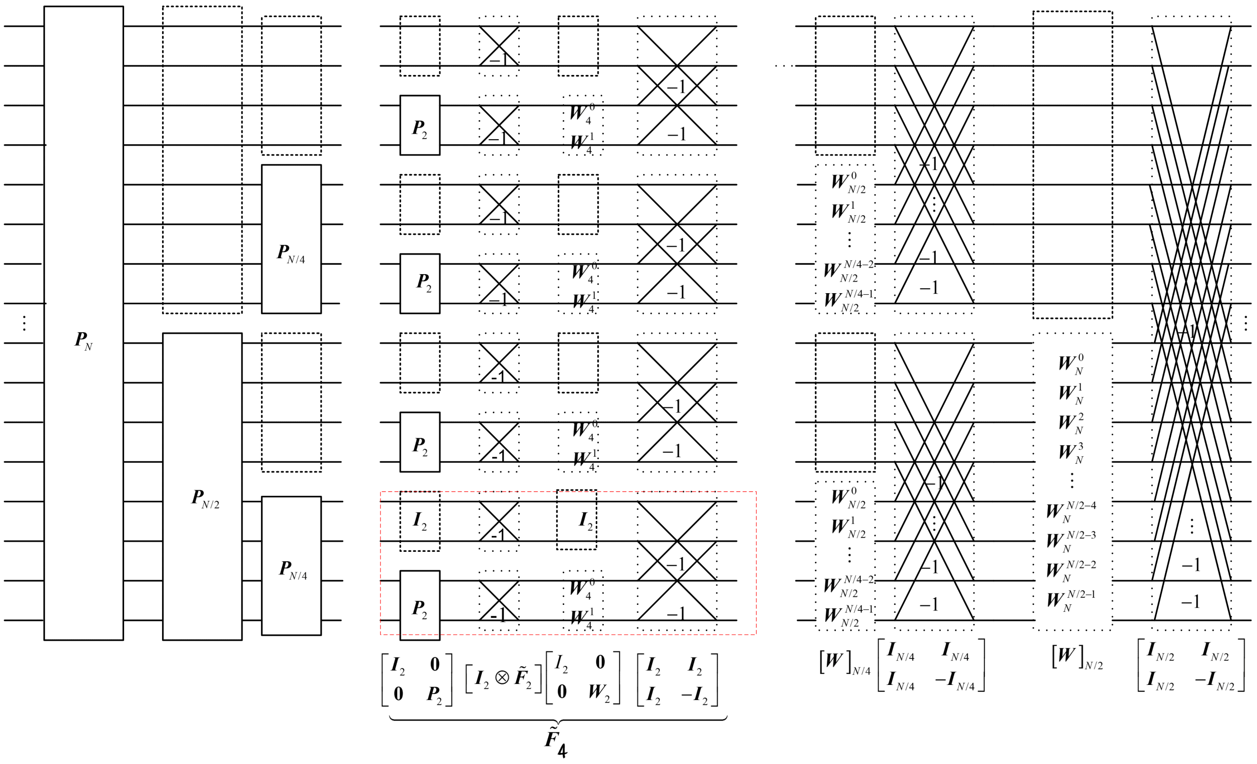
for $N=4,\ 8,\ \ldots$ In (25), we define $H_2=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$. We decompose a sparse matrix $\boldsymbol{E}_N=\boldsymbol{P}_N\widetilde{\boldsymbol{F}}_N\boldsymbol{W}_N$ in the following way:

$$\begin{aligned} \boldsymbol{F}_N &= [\boldsymbol{P}_N]^T \tilde{\boldsymbol{F}}_N \\ \tilde{\boldsymbol{F}}_N &= \begin{bmatrix} \tilde{\boldsymbol{F}}_{N/2} & \tilde{\boldsymbol{F}}_{N/2} \\ \boldsymbol{E}_{N/2} & \textbf{-}\boldsymbol{E}_{N/2} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{F}}_{N/2} & 0 \\ 0 & \boldsymbol{E}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \textbf{-}\boldsymbol{I}_{N/2} \end{bmatrix} \end{aligned} \tag{26}$$

where $\boldsymbol{E}_{N/2}$ is further decomposed by Lemma 1

$$\boldsymbol{E}_{N/2} = \boldsymbol{P}_{N/2} \tilde{\boldsymbol{F}}_{N/2} \boldsymbol{W}_{N/2} \tag{27}$$

where $\boldsymbol{W}_N$ is the diagonal complex unit for the N-point DFT matrix. That is, we have $\boldsymbol{W}_N = diag\{\boldsymbol{W}^0,\ \ldots,\ \boldsymbol{W}^{N-1}\}$.



$\boldsymbol{P}_N$
$\boldsymbol{P}_{N/2}$
$\boldsymbol{P}_{N/4}$
$\boldsymbol{P}_2$
$\boldsymbol{I}_2$
$\boldsymbol{W}_4^0$
$\boldsymbol{W}_4^1$
−1
$\boldsymbol{W}_{N/2}^0$
$\boldsymbol{W}_{N/2}^1$
$\boldsymbol{W}_{N/2}^{N/4-2}$
$\boldsymbol{W}_{N/2}^{N/4-1}$
$\boldsymbol{W}_N^0$
$\boldsymbol{W}_N^1$
$\boldsymbol{W}_N^2$
$\boldsymbol{W}_N^3$
$\boldsymbol{W}_N^{N/2-4}$
$\boldsymbol{W}_N^{N/2-3}$
$\boldsymbol{W}_N^{N/2-2}$
$\boldsymbol{W}_N^{N/2-1}$
$\underbrace{\begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_2 \end{bmatrix} [\boldsymbol{I}_2 \otimes \tilde{\boldsymbol{F}}_2] \begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{W}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{I}_2 \\ \boldsymbol{I}_2 & -\boldsymbol{I}_2 \end{bmatrix}}_{\tilde{\boldsymbol{F}}_4}$
$[\boldsymbol{W}]_{N/4} \begin{bmatrix} \boldsymbol{I}_{N/4} & \boldsymbol{I}_{N/4} \\ \boldsymbol{I}_{N/4} & -\boldsymbol{I}_{N/4} \end{bmatrix}$
$[\boldsymbol{W}]_{N/2}$
$\begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & -\boldsymbol{I}_{N/2} \end{bmatrix}$

Figure 3. Butterfly data flow of DFT.

Similar to the development for DCT-II and DST-II, we first rewrite (26) using (27) as

$$\begin{aligned} \tilde{\boldsymbol{F}}_N &= \begin{bmatrix} \tilde{\boldsymbol{F}}_{N/2} & 0 \\ 0 & \boldsymbol{P}_{N/2}\tilde{\boldsymbol{F}}_{N/2}\boldsymbol{W}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \textbf{-}\boldsymbol{I}_{N/2} \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{I}_{N/2} & 0 \\ 0 & \boldsymbol{P}_{N/2} \end{bmatrix} [\boldsymbol{I}_2 \otimes \tilde{\boldsymbol{F}}_{N/2}] \begin{bmatrix} \boldsymbol{I}_{N/2} & 0 \\ 0 & \boldsymbol{W}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \textbf{-}\boldsymbol{I}_{N/2} \end{bmatrix}. \end{aligned} \tag{28}$$

$[\boldsymbol{I}_2 \otimes \widetilde{\boldsymbol{F}}_{N/2}]$ in (28) can be recursively decomposed in the following way:

$$\tilde{\boldsymbol{F}}_N = \begin{bmatrix} \boldsymbol{I}_{N/2} & 0 \\ 0 & \boldsymbol{P}_{N/2} \end{bmatrix} \times \left[\boldsymbol{I}_2 \otimes \underbrace{\left[\underbrace{\cdots \boldsymbol{P}_4 \begin{bmatrix} \boldsymbol{I}_2 & 0 \\ 0 & \boldsymbol{P}_2 \end{bmatrix} \left[\boldsymbol{I}_2 \otimes \tilde{\boldsymbol{F}}_2 \right] \begin{bmatrix} \boldsymbol{I}_2 & 0 \\ 0 & \boldsymbol{W}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{I}_2 \\ \boldsymbol{I}_2 & \textbf{-}\boldsymbol{I}_2 \end{bmatrix} \cdots}_{\tilde{\boldsymbol{F}}_4} \right]}_{\tilde{\boldsymbol{F}}_{N/2}} \right] \times \begin{bmatrix} \boldsymbol{I}_{N/2} & 0 \\ 0 & \boldsymbol{W}_{N/2} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{N/2} & \boldsymbol{I}_{N/2} \\ \boldsymbol{I}_{N/2} & \textbf{-}\boldsymbol{I}_{N/2} \end{bmatrix}. \tag{29}$$

It is clear that the form of (29) is the same as that of (13), where we only need to change $\boldsymbol{L}_l$ to $\boldsymbol{P}_l$ and $\boldsymbol{D}_l$ to $\boldsymbol{W}_l$ for $l \in \{2, 4, 8, \ldots, N/2\}$ to convert the DCT-II matrix into DFT matrix. Consequently, the butterfly data flow of the DFT matrix can be drawn in Fig. 3 using the baseline architecture of DCT-II.

3. Proposed hybrid architecture for fast computations of DCT-II, DST-II, and DFT matrices

We have derived recursive formulas for DCT-II, DST-II, and DFT. The derived results show that DCT-II, DST-II, and DFT matrices can be unified by using a similar sparse matrix decomposition algorithm, which is based on the block-wise Jacket matrix and diagonal recursive architecture with different characters. The conventional method is only converted from DFT to DCT-II, DST-II. But our proposed method can be universally switching from DCT-II to DST-II, and DFT vice versa. Figs. 1-3 exhibit the similar recursive flow diagrams and let us motivate to develop universal hybrid architecture via switching mode selection. Moreover, the butterfly data flow graphs have $\log_2 N$ stages. From Fig.1, we can generate Figs. 2-3 according to the following proposed ways:

3.1. From DCT-II to DST-II

The N-point DCT-II of x is given by

$$\begin{gathered} X_N^{DCT}(m) = c_m \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \cos \frac{m(2n+1)\pi}{2N} = c_m \sqrt{\frac{2}{N}} C_N \mathbf{x} \\ \text{where} \quad m, n = 0, 1, \ldots, N-1, \; c_m = \begin{cases} 1 & , m \neq 0 \\ 1/\sqrt{2} & , m = 0 \end{cases} \end{gathered} \tag{30}$$

The N-point DST-II of $\mathbf{x}$ is given by

$$X_N^{DST}(m) = s_m\sqrt{\frac{2}{N}}\sum_{n=0}^{N-1} x(n)\sin\frac{(m+1)(2n+1)\pi}{2N} = s_m\sqrt{\frac{2}{N}}S_N\mathbf{x}$$
$$where\ \ m,n = 0,1,\ldots,N-1,\ \ s_m = \begin{cases} 1 & ,m \neq N-1 \\ 1/\sqrt{2} & ,m = N-1 \end{cases} \tag{31}$$

Let C_N and S_N be orthogonal $N\times N$ DCT-II and DST-II matrices, respectively. Also, x=$[\boldsymbol{x}(0)\ \ \boldsymbol{x}(1)\ \ \ldots\ \ \boldsymbol{x}(N-1)]^T$ denotes the column vector for the data sequence x(n). Substituting $m=N-k-1$, $k=1, 2, \ldots, N$ into (30), we have

$$C_N(N-k-1) = c_{N-k}\sqrt{\frac{2}{N}}\sum_{n=0}^{N-1} x(n)\cos\frac{(2n+1)(N-k-1)\pi}{2N},\ \ k = 0,1,2,\ldots,N-1 \tag{32}$$

Using the following trigonometric identity

$$\begin{aligned}
&\cos\left(\frac{(2n+1)\pi}{2} - \frac{(2n+1)(k+1)\pi}{2N}\right) \\
&= \cos\left(\frac{(2n+1)\pi}{2}\right)\cos\left(\frac{(2n+1)(k+1)\pi}{2N}\right) + \sin\left(\frac{(2n+1)\pi}{2}\right)\sin\left(\frac{(2n+1)(k+1)\pi}{2N}\right) \\
&= (-1)^n \sin\left(\frac{(2n+1)(k+1)\pi}{2N}\right)
\end{aligned} \tag{33}$$

(32) becomes

$$C_N(N-k-1) = c_{N-k}\sqrt{\frac{2}{N}}\sum_{n=0}^{N-1}(-1)^n x(n)\sin\frac{(2n+1)(k+1)\pi}{2N} \tag{34}$$

where $C_N=(N-k-1)$ represents the reflected version of $C_N(k)$ and this can be achieved by multiplying the reversed identity matrix $\bar{\boldsymbol{I}}_N$ to C_N. (34) can be represented in a more compact matrix multiplication form [13]:

$$\boldsymbol{S}_N = \bar{\mathbf{I}}_N\boldsymbol{C}_N\boldsymbol{M}_N \Leftrightarrow \boldsymbol{C}_N = \bar{\boldsymbol{I}}_N\boldsymbol{S}_N\boldsymbol{M}_N \tag{35}$$

where, $M_N=[M_1\otimes\boldsymbol{I}_{N/2}]$, $M_1=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$

Then, the DST-II matrix is resulted from the DCT-II matrix. Note that compatibility property exists in the DCT-II and DST-II.

3.2. From DFT to DCT-II

The (m,n) elements of the DCT-II kernel matrix is expressed by

$$[C_N]_{m,n} = c_m \sqrt{\frac{2}{N}} \cos \frac{m(2n+1)\pi}{2N} \tag{36}$$

A new sequence $x^{(1)}(n)$ is defined by

$$\begin{cases} x^{(1)}(n) = x(2n) & for\, 0 \le n \le N/2-1 \\ x^{(1)}(N-n-1) = x(2n+1) & for\, 0 \le n \le N/2-1 \end{cases} \tag{37}$$

For the sequence $x^{(1)}(n)$, we see that we can write

$$\begin{aligned} X_N^{DCT}(m) &= c_m \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x^{(1)}(n) \cos \frac{m(4n+1)\pi}{2N} = c_m \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x^{(1)}(n) \cos 2\pi \frac{m}{2N}\left(2n+\frac{1}{2}\right) \\ &= c_m \sqrt{\frac{2}{N}} \mathrm{R}\left(\sum_{n=0}^{N-1} x^{(1)}(n) \mathrm{e}^{-j2\pi m(2n+1/2)/2N}\right) = c_m \sqrt{\frac{2}{N}} \mathrm{R}\left(e^{-j\pi m/2N} \sum_{n=0}^{N-1} x^{(1)}(n) \mathrm{e}^{-j2\pi mn/N}\right) \\ &= c_m \sqrt{\frac{2}{N}} \mathrm{R}\left(e^{-j\pi m/2N} F_N \mathbf{x}^{(1)}\right) \end{aligned} \tag{38}$$

where **R** indicates a real part.

With the result above we have avoided computing a DFT of double size. We have

$$\mathbf{W}_{4N} = diag\left\{ {W_{4N}}^0, \ldots, W^{N-1} \right\} = diag\left\{ 1, e^{-j\pi/2N}, e^{-j\pi 2/2N}, \cdots, e^{-j\pi(N-1)/2N} \right\} \tag{39}$$

Now, the result can be put in the more compact matrix-vector form

$$C_N = c_m \sqrt{\frac{2}{N}} \mathbf{R}\left(\mathbf{W}_{4N} F_N\right) \tag{40}$$

Then, the DCT-II matrix is resulted from the DFT matrix.

3.3. From DCT-II and DST-II to DFT

We develop a relation between the circular convolution operation in the discrete cosine and sine transform domains. We need to measure half of the total coefficients. The main advantage of a proposed new relation is that the input sequences to be convolved need not be symmetrical or asymmetrical. Thus, the transform coefficients can be either symmetric or asymmetric [21].

From (30) and (31), it changes to coefficient for circular convolution (C) format. Thus, we have the following equations:

$$\begin{aligned}\mathbf{X}_N^{DCT-IIC}(m) &= 2\sum_{n=0}^{N-1} x(n)\cos\left(\frac{m(2n+1)}{2N}\right), \quad m = 0,1,\cdots,N-1\\ \mathbf{X}_N^{DST-IIC}(m) &= 2\sum_{n=0}^{N-1} x(n)\sin\left(\frac{m(2n+1)}{2N}\right), \quad m = 1,\cdots,N\end{aligned} \tag{41}$$

We can rewrite the DFT (24)

$$\boldsymbol{X}(m) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi mn/N}, \ m = 0,1,......,N-1. \tag{42}$$

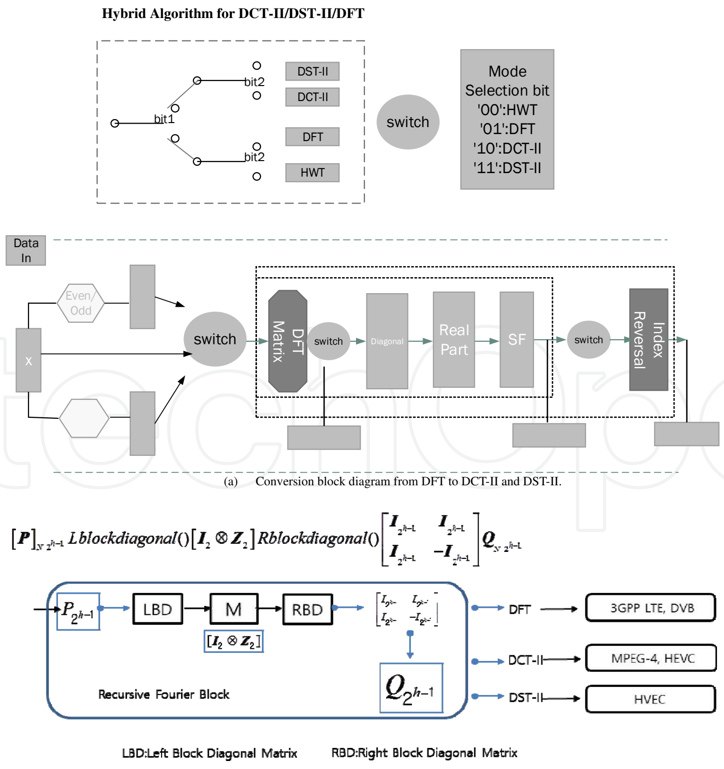
Multiplying (42) by $2e^{-j\pi m/N}$, we can get

$$\begin{aligned}2e^{-j\pi m/N}\boldsymbol{X}(m) &= 2e^{-j\pi m/N}\sum_{n=0}^{N-1} x(n)e^{-j2\pi mn/N} = 2\sum_{n=0}^{N-1} x(n)e^{-j\pi m/N}e^{-j2\pi mn/N}\\ &= 2\sum_{n=0}^{N-1} x(n)e^{-j\left[\frac{m(2n+1)\pi}{N}\right]} = 2\sum_{n=0}^{N-1} x(n)\left(\cos\left[\frac{m(2n+1)\pi}{N}\right] - j\sin\left[\frac{m(2n+1)\pi}{N}\right]\right).\end{aligned} \tag{43}$$

Comparing the first term of (41) with first one of (43), it can be seen that $2\sum_{n=0}^{N-1} x(n)\left(\cos\left[\frac{m(2n+1)\pi}{N}\right]\right)$ is decimated and asymmetrically extended of (41) with index $m=0:N-1$. Similarly, $2\sum_{n=0}^{N-1} x(n)\left(\sin\left[\frac{m(2n+1)\pi}{N}\right]\right)$ is decimated and symmetrically extended of (41) with index $m=1:N$. It is observed that proper zero padding of the sequences, symmetric convolution can be used to perform linear convolution. The circular convolution of cosine and sine periodic sequences in time/spatial domain is equivalent to multiplication in the DFT domain. Then, the DFT matrix is resulted from the DCT-II and DST-II matrices.

3.4. Unified hybrid fast algorithm

Based on the above conversions from the proposed decomposition of DCT-II, we can form a hybrid fast algorithm that can cover DCT-II, DST-II, and DFT. The general block diagram of the proposed hybrid fast algorithm is shown in Fig. 4. The common recursive block of $[\boldsymbol{P}]_{N/2^{h-1}}\boldsymbol{L}\ blockdiagonal()[\boldsymbol{I}_2 \otimes Z_2]Rblockdiagonal()\begin{bmatrix}\boldsymbol{I}_2 & \boldsymbol{I}_2\\ \boldsymbol{I}_2 & -\boldsymbol{I}_2\end{bmatrix}Q_{N/2^{h-1}}$ is multiplied repeatedly according to the size of the kernel with different transforms as like as bracket ((((·)))). The requiring computational complexity of individual DCT-II, DST-II, and DFT is summarized in Table 1 and Table 2. It can be seen that the proposed hybrid algorithm requires little more computations in addition and multiplication compared to Wang's result [13]. However, the proposed scheme requires a much less computational complexity in addition and multiplication compared to those of the decompositions proposed by [11,13,18]. In addition, compared to these transforms, the proposed hybrid fast algorithm can be efficiently extensible to larger transform sizes due to its diagonal block-wise inverse operation of recursive structure. Moreover, the proposed hybrid structure is easily extended to cover different applications. For example, a base station wireless communication terminal delivers a compressed version of multimedia data via wireless communications network. Either DCT-II or DST-II can be used in compressing multimedia data since the proposed decomposition is based on block diagonalization it can significantly reduce its complexity due to simple structure[11,19, 22], for various multimedia sources. The DCT image coding can be easily implemented in the proposed hybrid structure as shown in Fig. 4(b). From (45), the DCT-II is obtained by taking a real part of multiplication result of $e^{-j\pi m/2N}$ with $\boldsymbol{F}_N = \{W^{nm}\}$. If the DCT-II is multiplied by $\bar{\boldsymbol{I}}_N C_N \boldsymbol{M}_N$, then we get DST. If the DCT and DST are convolved in time and frequency domain and multiplied by $2e^{-j\pi m/N}$, the DFT matrix can be obtained. Thus, the proposed hybrid algorithm enables the terminal to adapt to its operational physical device and size.


Hybrid Algorithm for DCT-II/DST-II/DFT
bit1
bit2
bit2
DST-II
DCT-II
DFT
HWT
switch
Mode Selection bit
'00':HWT
'01':DFT
'10':DCT-II
'11':DST-II
Data In
Even/Odd
x
switch
DFT Matrix
switch
Diagonal
Real Part
SF
switch
Index Reversal
$[\boldsymbol{P}]_{N,2^{k-1}}\,Lblockdiagonal()[\boldsymbol{I}_2 \otimes \boldsymbol{Z}_2]\,Rblockdiagonal()\begin{bmatrix} \boldsymbol{I}_{2^{k-1}} & \boldsymbol{I}_{2^{k-1}} \\ \boldsymbol{I}_{2^{k-1}} & -\boldsymbol{I}_{2^{k-1}} \end{bmatrix}\boldsymbol{Q}_{N,2^{k-1}}$
$P_{2^{k-1}}$
LBD
M
RBD
$[I_2 \otimes Z_2]$
$\begin{bmatrix} I_{2^{k-1}} & I_{2^{k-1}} \\ I_{2^{k-1}} & -I_{2^{k-1}} \end{bmatrix}$
$Q_{2^{k-1}}$
Recursive Fourier Block
DFT
DCT-II
DST-II
3GPP LTE, DVB
MPEG-4, HEVC
HVEC
LBD:Left Block Diagonal Matrix
RBD:Right Block Diagonal Matrix

(a) Conversion block diagram from DFT to DCT-II and DST-II.

(b) Block diagram of hybrid DCT-II/DST-II/DFT.

Figure 4. Recursive DCT-II/DST-II/DFT Structure Based on Jacket matrix.

3.5. Numerical simulations

As shown in [7] the coding performance DST outperforms DCT at high correlation values (ρ) and is very close to that of the KLT. Since the basis vectors of DCT maximize their energy distribution at both ends, hence the discontinuity appears at block boundaries due to quantization effects. However, since the basis vectors of DST minimizes their energy distribution at other ends, DST provides smooth transition between neighboring blocks. Therefore, the proposed hybrid transform coding scheme provides a consistent reconstruction and preserves more details, as shown in Fig. 6 with a size of 512 x 512 and 8 bits quantization.

Now consider an $N \times N$ block of pixels, X, containing $x_{i,j}, \ i, \ j = 1, 2, \ \ldots, \ N$. We can write 2-D transformation for the *k*th block X as $Y_S = T_S Q X_k Q^T$ and $Y_C = T_C X_k$.

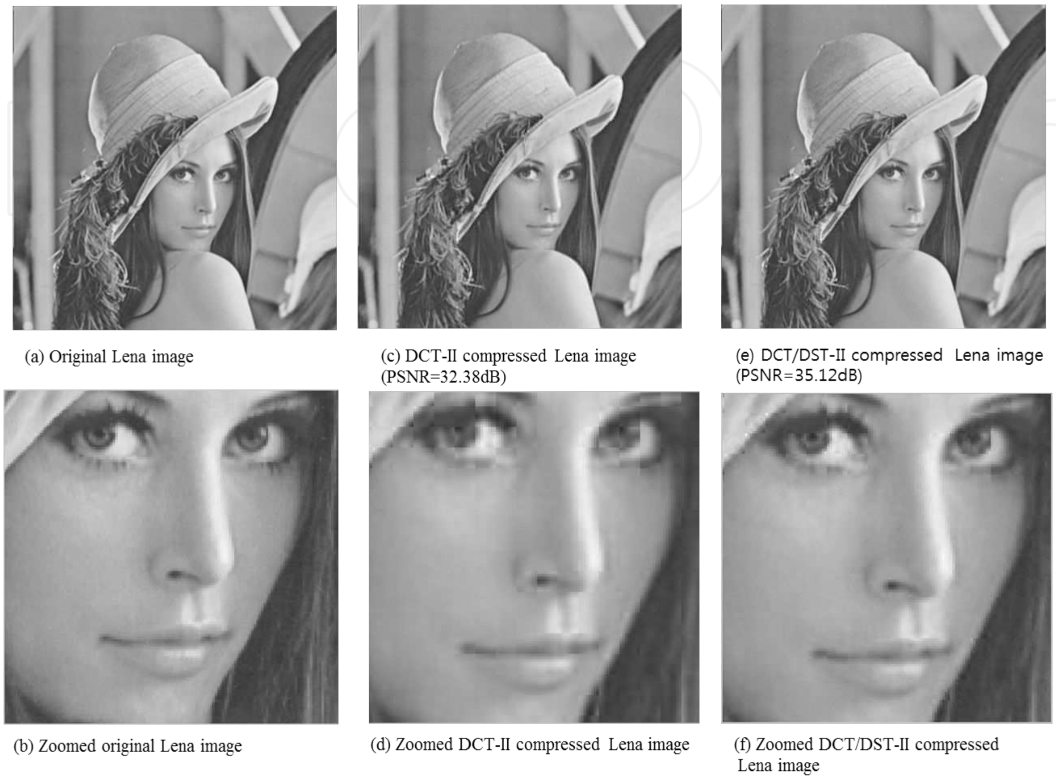
Depending on the availability of boundary values (in top- boundary and left-boundary) in images the hybrid coding scheme accomplishes the 2-D transform of a block pixels as two sequential 1-D transforms separately performed on rows and columns. Therefore the choice of 1-D transform for each direction is dependent on the corresponding prediction boundary condition.

- Vertical transform (for each column vector): employ DST if top boundary is used for prediction; otherwise use DCT.
- Horizontal transform (for each row vector): employ DST if left boundary is used for prediction; otherwise use DCT.

What we observed from numerical experiments is that the combined scheme over DCT-II only performs better in perceptual clarity as well as PSNR. Jointly optimized spatial prediction and block transform (see Fig. 5 (e) and (f)) using DCT/DST-II compression(PSNR 35.12dB) outperforms only DCT-II compression(PSNR 32.38dB). Less blocky artifacts are revealed compared to that of DCT-II. Without *a priori* knowledge of boundary condition, DCT-II performs better than any other block transform coding. The worst result is obtained using DST-II only.

4. Conclusion

In this book chapter, we have derived a unified fast hybrid recursive Fourier transform based on Jacket matrix. The proposed analysis have shown that DCT-II, DST-II, and DFT can be unified by using the diagonal sparse matrix based on the Jacket matrix and recursive structure with some characters changed from DCT-II to DST-II, and DFT. The proposed algorithm also uses the matrix product of recursively lower order diagonal sparse matrix and Hadamard matrix. The resulting signal flow graphs of DCT-II, DST-II, and DFT have a regular systematic butterfly structure. Therefore, the complexity of the proposed unified hybrid algorithm has been much less as its matrix size gets larger. This butterfly structure has grown by a recursive nature of the fast hybrid Jacket Hadamard matrix. Based on a systematic butterfly structure, a unified switching system can be devised. We have also applied the circulant channel matrix in our proposed method. Thus, the proposed hybrid scheme can be effectively applied to the heterogeneous transform systems having various matrix dimensions. Jointly optimized DCT and DST-II compression scheme have revealed a better performance (about 3dB) over the DCT or DST only compression method.


(a) Original Lena image

(c) DCT-II compressed Lena image (PSNR=32.38dB)

(e) DCT/DST-II compressed Lena image (PSNR=35.12dB)

(b) Zoomed original Lena image

(d) Zoomed DCT-II compressed Lena image

(f) Zoomed DCT/DST-II compressed Lena image

Figure 5. Image Coding Results showing DCT-II only and jointly optimized DCT/DST-II compression (a) Original Lena image (b) zoomed original Lena image (c) DCT-II compressed Lena image(PSNR=32.38 dB) (d) Zoomed DCT-II compressed Lena image (e) DCT/DST-II compressed Lena image (PSNR=35.12 dB) (f) Zoomed DCT/DST-II compressed Lena image.

Appendix

Appendix A

A Proof of Proposition 1

We use mathematical induction to prove Proposition 1. The lowest order BIJM is defined as

$$\boldsymbol{J}_8 = \begin{bmatrix} \boldsymbol{I}_2 & \boldsymbol{I}_2 & \boldsymbol{I}_2 & \boldsymbol{I}_2 \\ \boldsymbol{I}_2 & -\boldsymbol{C}_2 & \boldsymbol{C}_2 & -\boldsymbol{I}_2 \\ \boldsymbol{I}_2 & \boldsymbol{C}_2 & -\boldsymbol{C}_2 & -\boldsymbol{I}_2 \\ \boldsymbol{I}_2 & -\boldsymbol{I}_2 & -\boldsymbol{I}_2 & \boldsymbol{I}_2 \end{bmatrix} \tag{44}$$

where $C_2 = \frac{H_2}{\sqrt{2}}$. Since

$$J_8^{-1} = \begin{bmatrix} I_2 & I_2 & I_2 & I_2 \\ I_2 & -C_2^T & C_2^T & -I_2 \\ I_2 & C_2^T & -C_2^T & -I_2 \\ I_2 & -I_2 & -I_2 & I_2 \end{bmatrix} \tag{45}$$

equation (4) holds for $2N = 8$. Now we assume that the BIJM J_N satisfies (4), i.e., $J_N J_N^T = \frac{N}{2} \boldsymbol{I}_N$. Since $J_{2N} J_{2N}^T = (J_N \otimes H_2)(J_N \otimes H_2)^T = (J_N J_N^T) \otimes (H_2 H_2^T) = \frac{N}{2} \boldsymbol{I}_N \otimes 2\boldsymbol{I}_2 = N \boldsymbol{I}_{2N}$, this proposition is proved by mathematical induction that (4) holds for all $2N$. If $N = 1$, certainly $J_2 J_2^T = \boldsymbol{I}_2$.

Appendix B

A Proof of Lemma 1

According to the definition of an $N \times N$ matrix B_N, B_N is given as follows:

$$\boldsymbol{B}_N = \begin{bmatrix} C_{4N}^{\Phi_0} & C_{4N}^{\Phi_1} & C_{4N}^{\Phi_2} & \cdots & C_{4N}^{\Phi_{N-1}} \\ C_{4N}^{(2k_0+1)\Phi_0} & C_{4N}^{(2k_0+1)\Phi_1} & C_{4N}^{(2k_0+1)\Phi_2} & \cdots & C_{4N}^{(2k_0+1)\Phi_{N-1}} \\ C_{4N}^{(2k_1+1)\Phi_0} & C_{4N}^{(2k_1+1)\Phi_1} & C_{4N}^{(2k_1+1)\Phi_2} & \cdots & C_{4N}^{(2k_1+1)\Phi_{N-1}} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ C_{4N}^{(2k_{N-2}+1)\Phi_0} & C_{4N}^{(2k_{N-2}+1)\Phi_1} & C_{4N}^{(2k_{N-2}+1)\Phi_2} & \cdots & C_{4N}^{(2k_{N-2}+1)\Phi_{N-1}} \end{bmatrix} \tag{46}$$

where $k_i = i + 1$. Since $\cos((2k+1)\Phi_m) = 2\cos(2k\Phi_m)\cos(\Phi_m) - \cos((2k-1)\Phi_m)$,we have

$$C_{4N}^{(2k_i+1)\Phi_m} = -C_{4N}^{(2k_i-1)\Phi_m} + 2C_{4N}^{(2k_i)\Phi_m} C_{4N}^{\Phi_m}. \tag{47}$$

Using (47), B_N can be decomposed as:

$$\begin{aligned} \boldsymbol{B}_N &= \boldsymbol{L}_N \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ C_{4N}^{2k_0\Phi_0} & C_{4N}^{2k_0\Phi_1} & C_{4N}^{2k_0\Phi_2} & \cdots & C_{4N}^{2k_0\Phi_{N-1}} \\ C_{4N}^{2k_1\Phi_0} & C_{4N}^{2k_1\Phi_1} & C_{4N}^{2k_1\Phi_2} & \cdots & C_{4N}^{2k_1\Phi_{N-1}} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ C_{4N}^{2k_{N-2}\Phi_0} & C_{4N}^{2k_{N-2}\Phi_1} & C_{4N}^{2k_{N-2}\Phi_2} & \cdots & C_{4N}^{2k_{N-2}\Phi_{N-1}} \end{bmatrix} \boldsymbol{D}_N \\ &= \boldsymbol{L}_N \boldsymbol{X}_N \boldsymbol{D}_N \end{aligned} \tag{48}$$

which proves (10) in Lemma 1.

Appendix C

*A Proof of*Equation (18)

By using the sum and difference formulas for the sine function, we can have

$$\begin{aligned}
&S_{4N}^{(2k_{N-1})\Phi_j} = C_{4N}^{\Phi_j} = S_{4N}^{\Phi_{N-j-1}}, S_{4N}^{(2k_i-1)\Phi_j} = 2S_{4N}^{2k_i\Phi_j} C_{4N}^{\Phi_j} - S_{4N}^{(2k_{i+1}-1)\Phi_j},\\
&S_{4N}^{(2k_{i+1}-1)\Phi_j} = 2S_{4N}^{2k_i\Phi_j} C_{4N}^{\Phi_j} - S_{4N}^{(2k_i-1)\Phi_j},\\
&S_{4N}^{(2k_0-1)\Phi_j} = \left(2S_{4N}^{(2k_0-1)\Phi_j} - 2S_{4N}^{(2k_1)\Phi_j} + \cdots + 2S_{4N}^{(2k_{N-2}-1)\Phi_j}\right) C_{4N}^{\Phi_j}\\
&\qquad = 2S_{4N}^{(2k_0-1)\Phi_j} C_{4N}^{\Phi_j} - S_{4N}^{(2k_1-1)\Phi_j},\\
&S_{4N}^{(2k_1-1)\Phi_j} = \left(2S_{4N}^{(2k_1-1)\Phi_j} - 2S_{4N}^{(2k_1)\Phi_j} + \cdots + 2S_{4N}^{(2k_{N-2}+1)\Phi_j}\right) C_{4N}^{\Phi_j}\\
&\qquad = 2S_{4N}^{(2k_1-1)\Phi_j} C_{4N}^{\Phi_j} - S_{4N}^{(2k_2-1)\Phi_j},\\
&\qquad \vdots\\
&S_{4N}^{(2k_{N-3}-1)\Phi_j} = \left(2S_{4N}^{(2k_{N-3}-1)\Phi_j} - 2S_{4N}^{(2k_{N-2})\Phi_j} + 1\right) C_{4N}^{\Phi_j}\\
&\qquad = 2S_{4N}^{(2k_{N-3}-1)\Phi_j} C_{4N}^{\Phi_j} - S_{4N}^{(2k_{N-2}-1)\Phi_j},\\
&S_{4N}^{(2k_{N-2}-1)\Phi_j} = \left(2S_{4N}^{(2k_{N-2})\Phi_j} - 1\right) C_{4N}^{\Phi_j}\\
&\qquad = 2S_{4N}^{(2k_{N-2})\Phi_j} C_{4N}^{\Phi_j} - S_{4N}^{(2k_{N-1}-1)\Phi_j}
\end{aligned} \tag{49}$$

where $k_i = i+1,\ \Phi_j = 2j+1,\ i,\ \ j = 0,\ 1,\ \cdots,\ N-1.$

By taking (49) and into the right hand side of (18), we have

$$\boldsymbol{U}_N Y_N \boldsymbol{D}_N = \begin{bmatrix} S_{4N}^{(2k_0-1)\Phi_0} & S_{4N}^{(2k_0-1)\Phi_1} & \cdots & S_{4N}^{(2k_0-1)\Phi_{N-1}} \\ S_{4N}^{(2k_1-1)\Phi_0} & S_{4N}^{(2k_1-1)\Phi_1} & \cdots & S_{4N}^{(2k_1-1)\Phi_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ S_{4N}^{(2k_{N-1}-1)\Phi_0} & S_{4N}^{(2k_{N-1}-1)\Phi_1} & \cdots & S_{4N}^{(2k_{N-1}-1)\Phi_{N-1}} \end{bmatrix}. \tag{50}$$

The left hand side of (18) matrix $[A]_N$ from $[\mathrm{Y}]_N$ can be represented by

$$\boldsymbol{A}_N = \begin{bmatrix} S_{4N}^{(2k_0-1)\phi_0} & S_{4N}^{(2k_0-1)\phi_1} & \cdots & S_{4N}^{(2k_0-1)\phi_{N-1}} \\ S_{4N}^{(2k_1-1)\phi_0} & S_{4N}^{(2k_1-1)\phi_1} & \cdots & S_{4N}^{(2k_1-1)\phi_{N-1}} \\ \vdots & \vdots & \ddots & \vdots \\ S_{4N}^{(2k_{N-1}-1)\phi_0} & S_{4N}^{(2k_{N-1}-1)\phi_1} & \cdots & S_{4N}^{(2k_{N-1}-1)\phi_{N-1}} \end{bmatrix}. \tag{51}$$

We can obtain (50) and (51) are the same and the expression of (18) is correct.

Acknowledgements

This work was supported by MEST 2012- 002521, NRF, Korea.

Author details

Daechul Park[1*] and Moon Ho Lee[2]

*Address all correspondence to: fia4joy@gmail.com

1 Hannam University, Department of Information and Communication engineering, Korea

2 Chonbuk National University, Division of Electronics and Information Engineering, Korea

References

[1] Rao, KR. and Yip, P., *Discrete Cosine Transform: Algorithms, Advantages, Applications*. Boston, MA: Academic Press, 1990.

[2] Richardson, IE., *The H.264 Advanced Video Compression Standard*, 2nd ed. Hoboken, New Jersey: John Wiley and Sons.

[3] Rao, KR., Kim, DN., and Hwang, J. J., *Fast Fourier Transform: Algorithm and Applications*. New York, N.Y.: Springer, 2010.

[4] Jain, A. K., *Fundamentals of Digital Image Processing*. Prentice Hall, 1987.

[5] Wang, R., *Introduction to Orthogonal Transforms: With Applications in Data Processing and Analysis*. Cambridge, UK: Cambridge University Press, 2012.

[6] ITU-T SG16 WP3/JCT-VC, CE 7.5, "Performance analysis of adaptive DCT/DST selection," July 2011.

[7] Hai, J., Saxena, A., Melkote, V., and Rose, K., "Jointly optimized spatial prediction and block transform for video and image coding," *IEEE Trans. Image Process.*, vol. 21, no. 4, pp. 1874–1884, April 2012.

[8] Strang, G. and Nguyen, T., *Wavelets and Filer Banks*. Wellesley, MA: Wellesley-Cambridge Press, 1996.

[9] Lee, MH., "A new reverse Jacket transform and its fast algorithm," *IEEE Trans. Circuits Syst. II*, vol. 47, no. 1, pp. 39–47, Jan. 2000.

[10] Chen, Z.,. Lee, MH, and Zeng, G., "Fast cocyclic Jacket transform," *IEEE Trans. Signal Process.*, vol. 56, pp. 2143–2148, May 2008.

[11] Lee, MH., *Jacket Matrices-Construction and Its Application for Fast Cooperative Wireless Signal Processin*g. LAP LAMBERT Academic publishing, Germany, November, 2012.

[12] Wang, CL. and Chen, CY., "High-throughput VLSI architectures for the 1-D and 2-D discrete cosine transform," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, pp. 31–40, Feb. 1995.

[13] Wang, Z., "Fast Algorithm for the Discrete W Transform and for the Discrete Fourier Transform," *IEEE Trans. on Acoustics, Speech and Signal Process*., vol. 32, No. 4, pp. 803 – 816, Aug. 1984.

[14] Lee, MH., "High speed multidimensional systolic arrays for discrete Fourier transform," *IEEE Trans. Circuits Syst. II*, vol. 39, no. 12, pp. 876–879, Dec. 1992.

[15] Kim, KJ., Fan, Y.,Iltis, R. A., Poor, H. V., and Lee, M. H., "A reduced feedback precoder for MIMO-OFDM cooperative diversity system," *IEEE Trans. Veh. Technol.*, vol. 61, pp. 584–596, Feb. 2012.

[16] Jang, U.,Cho, K.,Ryu, W., and Lee, HJ., "Interference management with block diagonalization for macro/femto coexisting networks," *ETRI Journal*, vol. 34, pp. 297–307, June 2012.

[17] Hou, HS., "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Process*., vol. 35, no. 10, pp. 1455–1461, Oct. 1987.

[18] Chen, WH., Smith, CH., and Fralick, SC., "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun*., vol. 25, no. 9, pp. 1004–1009, Sep. 1977.

[19] Spencer, Q. H., Lee, A. Swindlehurst, M. Haardt, "Zero-forcing methods for downlink spatial multiplexing in multiuser MIMO channels", *IEEE Trans. Signal Process*., vol. 52,no. 2, Feb. 2004.

[20] Andrews, HC., Caspari, KL., "A generalized technique for spectral analysis," *IEEE Trans. Computers*, vol. 19, no. 1, pp.16-17, 1970.

[21] Reju, VG.,Koh, SN.,Soon, IY., "Convolution using discrete sine and cosine transforms", IEEE Signal Processing Letters, vol. 14, no. 7, July 2007.

[22] Lee, MH., Khan, MHA., Sarker, MA. L., Guo, Y. and Kim, KJ., "A MIMO LTE precoding based on fast diagonal weighted Jacket matrices", *Fiber and Integrated Optics*, *Taylor and Francis*, Invited paper, vol. 31, no. 2, pp. 111-132, March 2012.

[23] Khan, MHA., Li, J., Lee, MH., "A block diagonal Jacket matrices for MIMO broadcast channel" *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, Brunel University, June 4-7th, 2013, UK.