1-18-2016

Signal Flow Graph Approach to Efficient DST I-IV Algorithms

Sirani M. Perera
Embry-Riddle Aeronautical University, pereras2@erau.edu

Signal Flow Graph Approach to Efficient DST I-IV Algorithms

Sirani M. Perera

Keywords: Discrete Sine Transform; Fast and Efficient Algorithms; Recursive Algorithms; Arithmetic Cost; Sparse and Orthogonal Factors; Signal Flow Graphs

AMS classification: 15A23, 15B10, 65F50, 65T50, 65Y05, 65Y20, 94A12

Abstract

In this paper, fast and efficient discrete sine transformation (DST) algorithms are presented based on the factorization of sparse, scaled orthogonal, rotation, rotation-reflection, and butterfly matrices. These algorithms are completely recursive and solely based on DST I-IV. The presented algorithms have low arithmetic cost compared to the known fast DST algorithms. Furthermore, the language of signal flow graph representation of digital structures is used to describe these efficient and recursive DST algorithms having $(n-1)$ points signal flow graph for DST-I and n points signal flow graphs for DST II-IV.

1 INTRODUCTION

Applications of Fast Fourier Transform (FFT) have spread to a very diverse field in applied mathematics and electrical engineering and even the origin of the FFT goes back to analysis of the rotation of Helium molecule [9]. By now J. Dongarra and F. Sullivan have categorized FFT as one of the top 10 algorithms of the computer age which had the greatest influence on the development and practice of science and engineering in the 20^{th} century. FFT is used to compute Discrete Fourier Transform (DFT) and its inverse efficiently. On the other hand DFT implementation algorithms employ FFT so FFT and DFT are sometimes used interchangeably. Discrete Sine Transform (DST) is a Fourier-related transform similar to the DFT, but using a purely real matrix. Among applications of the DFT; sine and cosine waves of the DFT with different frequencies are used to classify the traffic monitoring sites into different seasonal patterns [35], DST has been identified as the method which generates better results for noise estimation as compared with Discrete Cosine Transform (DCT) and the DFT [10], discrete fractional sine transform has identified as the method for generating fingerprint templates with high recognition accuracy [49], DCT, DEST, and DFT can be approximated to the Karhunen Loeve Transformation (KLT) and the connection of KLT to the color image compression [6, 22, 31, 32], DST can be used to analyze image reconstruction via signal transition through a square-optical fiber lenses [47], spectral interference and additive wideband noise on the accuracy of the normalized frequency estimator can be investigated through discrete-time sine-wave [1], to mention a few. Together with the above, the engagement of DCT and DST in image processing, signal processing, finger print enhancement, quick response code (QR code), and multimode interface can also be seen in e.g., [2, 7, 11, 12, 16, 18, 19, 20, 21, 23, 24, 29, 30, 39, 40, 43, 44, 45].

The family of DFT consists of eight versions (I-VIII) of DCT and DST and these versions appear depending on odd or even type and also with respect to different Neumann and Dirichlet boundary conditions [6, 25, 34, 40]. Though there are eight versions, depending on applications in transform coding and digital filtering of signals, we consider DCT and DST matrices as varying from I to IV types. Let us consider four orthogonal types of DST having

superscripts to denote the type of DST and a subscript to denote the order of DST in the matrix form;

$$\begin{aligned}
DST-I: &\qquad S^{I}_{n-1} = \sqrt{\tfrac{2}{n}}\left[\sin\tfrac{(j+1)(k+1)\pi}{n}\right]^{n-2}_{j,k=0},\\
DST-II: &\quad S^{II}_{n} = \sqrt{\tfrac{2}{n}}\left[\varepsilon_n(j+1)\sin\tfrac{(j+1)(2k+1)\pi}{2n}\right]^{n-1}_{j,k=0}\\
DST-III: &\quad S^{III}_{n} = \sqrt{\tfrac{2}{n}}\left[\varepsilon_n(k+1)\sin\tfrac{(2j+1)(k+1)\pi}{2n}\right]^{n-1}_{j,k=0},\\
DST-IV: &\qquad S^{IV}_{n} = \sqrt{\tfrac{2}{n}}\left[\sin\tfrac{(2j+1)(2k+1)\pi}{4n}\right]^{n-1}_{j,k=0}
\end{aligned} \tag{1}$$

where $\varepsilon_n(0) = \varepsilon_n(n) = \frac{1}{\sqrt{2}}$, $\varepsilon_n(j) = 1$ for $j \in \{1,2,\cdots,n-1\}$ and $n \geq 2$ is an even integer. Among DST I-IV transformations, S^{I}_{n-1} and S^{IV}_{n} were introduced in [15, 14] and S^{II}_{n} and its inverse S^{III}_{n} were introduced in [20] into digital signal processing. DST-II is a complementary or alternative transform to DCT-II which is used in transform coding. Like DFT and DCT, these DST matrices hold linearity, convolution-multiplication, and shift properties.

Among different mathematical techniques used to derive fast algorithms for discrete cosine and sine transformations, the polynomial arithmetic technique (see e.g. [27, 41]) and the matrix factorization technique (see e.g. [6, 28, 48, 37, 38, 46]) can be seen as the dominant techniques. Apart from these two main techniques some other authors (see e.g. [17, 26]) have used different techniques like displacement approach and polynomial division in matrix form to derive factorizations for DCT and DST. Efficient algorithms for DCT or DST of radix-2 length n require about $2\,n\,log_2 n$ flops. Such a DCT or DST algorithm generates a factorization of these matrices having sparse and non-orthogonal matrices. Thus, if the factorization for DCT or DST does not preserve orthogonality the resulting DCT or DST algorithms lead to inferior numerical stability (see e.g. [42]). The matrix factorization for DST I in [48] used the results in [8] to decompose DST I into DCT and DST. Also the decomposition for DCT II in [46] is a slightly different version of the result in [8]. Though one can find orthogonal matrix factorization for DCT and DST in [46], the resulting algorithms in [46] are not completely recursive and hence do not lead to simple recursive algorithms. An alternative factorization for DCT I-IV in [38] and DST I-IV in [37] can be seen in [6, 28] but the factorizations in the latter papers are not solely dependent on DCT I-IV or DST I-IV. Moreover [6] has used the same factorization for DST-II and DST-IV as in [46]. However one can use these [6, 46] results to derive recursive, stable and radix-2 algorithms as stated in [28, 37, 38].

In electrical engineering, control theory, system engineering, theoretical computer science, etc. signal flow graphs represent realizations of systems as electronic devices. The objective here is to build a device to implement or realize algebraic operations used in sparse and orthogonal factorization of fast and recursive DST I-IV algorithms. Based on the factorization of DFT, DCT, and DST matrices one can design signal flow graphs such as: 8-point signal flow graphs on various fast DCT and DST algorithms having sparse and/or orthogonal factorization in [6], signal flow graphs for forward and backward modified DCT implementations with $n = 12$ and also with mix-radix decomposition of $n = 12$ in [3], fast DST-VII and DCT-II algorithms based signal flow graphs for $2n+1$ points and $n(2n+1)$ points DCT-II in [33], signal flow graphs representation of the direct 2-D DCT-II and 2-D DST-II computation and their inverses for $16\times 16, 8\times 16, 4\times 16, 16\times 8$, and 16×4 block sizes in [5], signal flow graphs of the coordinate rotation digital computer-based n points DCT-II, DCT-III and DST-II, DST-III in [13], signal flow graphs based Jacob rotation for $n/2$ points DCT-IV and modified DCT-IV in [4], and signal flow graphs based on hybrid jacket-Hadamard matrix for n points DCT-II, DST-II, and DFT-II in [23]. However, there is no paper on signal flow graphs based on fast and completely recursive DST I-IV algorithms having sparse, scaled orthogonal factorization, rotation, rotation-reflection matrix factorizations and especially the generalization of n points signal flow graphs covering all DST matrices of types I to IV. Hence in this paper we modify the sparse and orthogonal factorizations of stable DST I-IV algorithms proposed in [36, 37] to derive fast (compared to known algorithms), efficient, and completely recur-

sive sole algorithms based on DST I-IV having sparse, scaled orthogonal, rotation, rotation-reflections matrices and to discuss the arithmetic complexity of these fast DST I-IV algorithms. Furthermore, the paper presents generalized $n-1$ points signal flow graph for DST-I and n points signal flow graphs for DST II-IV based on the recursive DST I-IV algorithms.

In section 2 we modify the factorizations derived in [37] to express fast, efficient, and completely recursive sole algorithms for DST I-IV having scaled orthogonal, sparse, rotation, rotation-reflection, and butterfly matrices. Next, in section 3, we derive a number of additions and multiplications required to compute these fast and efficient DST I-IV algorithms and illustrate the numerical results based on that. In section 4, we develop and then generalize signal flow graphs for $n-1$ points DST-I algorithm and n points DST II-IV algorithms.

2 EFFICIENT AND RECURSIVE DST ALGORITHMS HAVING SPARSE, SCALED ORTHOGONAL, AND ROTATIONAL-REFLECTION FACTORS

This section presents fast, efficient, and completely recursive DST algorithms solely defined via DST I-IV having sparse, scaled orthogonal, rotational, and rotational-reflection factors by modifying radix-2, recursive, and stable DST I-IV algorithms having sparse and orthogonal factors introduced in [37]. The purpose of this is to significantly reduce the number of multiplications required to compute DST I-IV algorithms compared to the known fast, efficient, and stable DST algorithms having sparse factorizations.

By applying the permutation matrix to each sine transform matrix and using the trigonometric addition, complementary, and supplementary identities, one can derive the orthogonal matrix factorization for DST I-IV as in [37].

In the following we state the collection of sparse and orthogonal matrices which are frequently used in this paper. For a given vector $\mathbf{x} \in \mathbb{R}^n$, let us introduce an involution matrix $\tilde{I}_n$ by

$$\tilde{I}_n \mathbf{x} = \ [x_{n-1}, x_{n-2}, \cdots, x_0]^T \ ,$$

a diagonal matrix D_n by

$$D_n \mathbf{x} = \begin{cases} [x_0, -x_1, x_2, -x_3, \cdots, x_{n-1}, -x_{n-1}]^T & \text{even } n, \\ [x_0, -x_1, x_2, -x_3, \cdots, -x_{n-1}, x_{n-1}]^T & \text{odd } n \end{cases}$$

and, for $n \geq 3$ an even-odd permutation matrix P_n by

$$P_n \mathbf{x} = \begin{cases} [x_0, x_2, \cdots, x_{n-2}, x_1, x_3, \cdots, x_{n-1}]^T & \text{even } n, \\ [x_0, x_2, \cdots, x_{n-1}, x_1, x_3, \cdots, x_{n-2}]^T & \text{odd } n. \end{cases}$$

For even integer $n \geq 4$, we introduce sparse and orthogonal matrices:

$$\widehat{H}_{n-1} = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{\frac{n}{2}-1} & & \widetilde{I}_{\frac{n}{2}-1} \\ & \sqrt{2} & \\ I_{\frac{n}{2}-1} & & -\widetilde{I}_{\frac{n}{2}-1} \end{bmatrix}, \quad H_n = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{\frac{n}{2}} & \widetilde{I}_{\frac{n}{2}} \\ I_{\frac{n}{2}} & -\widetilde{I}_{\frac{n}{2}} \end{bmatrix},$$

$$V_n = \begin{bmatrix} 1 & & & \\ & \frac{1}{\sqrt{2}} \begin{bmatrix} I_{\frac{n}{2}-1} & -I_{\frac{n}{2}-1} \\ -I_{\frac{n}{2}-1} & -I_{\frac{n}{2}-1} \end{bmatrix} & & \\ & & & -1 \end{bmatrix} \begin{bmatrix} \widetilde{I}_{\frac{n}{2}} & \\ & D_{\frac{n}{2}} \end{bmatrix},$$

and also a rotational-reflection matrix:

$$Q_n = \begin{bmatrix} D_{\frac{n}{2}} & \\ & I_{\frac{n}{2}} \end{bmatrix} \begin{bmatrix} \operatorname{diag} S_{\frac{n}{2}} & \left(\operatorname{diag} C_{\frac{n}{2}}\right) \widetilde{I}_{\frac{n}{2}} \\ -\widetilde{I}_{\frac{n}{2}} \left(\operatorname{diag} C_{\frac{n}{2}}\right) & \operatorname{diag}\left(\widetilde{I}_{\frac{n}{2}} S_{\frac{n}{2}}\right) \end{bmatrix}$$

$$= \begin{bmatrix} \sin\frac{\pi}{4n} & & & & & & \cos\frac{\pi}{4n} \\ & -\sin\frac{3\pi}{4n} & & & & -\cos\frac{3\pi}{4n} & \\ & & \ddots & & \iddots & & \\ & & & -\sin\frac{(n-1)\pi}{4n} & -\cos\frac{(n-1)\pi}{4n} & & \\ & & & -\cos\frac{(n-1)\pi}{4n} & \sin\frac{(n-1)\pi}{4n} & & \\ & & \iddots & & \ddots & & \\ & -\cos\frac{3\pi}{4n} & & & & \sin\frac{3\pi}{4n} & \\ -\cos\frac{\pi}{4n} & & & & & & \sin\frac{\pi}{4n} \end{bmatrix},$$

where

$$C_{\frac{n}{2}} = \left[\cos\frac{(2k+1)\pi}{4n}\right]_{k=0}^{\frac{n}{2}-1} \quad \text{and} \quad S_{\frac{n}{2}} = \left[\sin\frac{(2k+1)\pi}{4n}\right]_{k=0}^{\frac{n}{2}-1}.$$

2.1 Stable, recursive, radix-2 DST I-IV algorithms having sparse and orthogonal factors

Before developing DST matrix factorization based on fast, efficient, and completely recursive DST I-IV algorithms, let us state stable, simple, recursive, radix-2 DST I-IV algorithms having sparse and orthogonal factors derived in [37].

The algorithms for DST I-IV are stated in order S_n^{II}, S_n^{IV}, S_n^{III} and S_{n-1}^{I} respectively.

In [37], orthogonal factorizations for types II and IV of discrete sine transform matrices are given by

$$S_n^{II} = P_n^T \begin{bmatrix} S_{\frac{n}{2}}^{IV} & 0 \\ 0 & S_{\frac{n}{2}}^{II} \end{bmatrix} H_n \qquad \text{and} \qquad S_n^{IV} = P_n^T V_n \begin{bmatrix} S_{\frac{n}{2}}^{II} & 0 \\ 0 & S_{\frac{n}{2}}^{II} \end{bmatrix} Q_n$$

Thus, the recursive algorithms for DST-II and DST-IV can be stated via algorithms $(\mathbf{2.1})$ and $(\mathbf{2.2})$ respectively.

Algorithm 2.1. $\mathbf{sin2}(n)$

Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$.

1. *If* $n = 2$, *then*
   $S2 := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.
2. *If* $n \geq 4$, *then*
   $M1 := \mathbf{sin4}(n_1)$,
   $M2 := \mathbf{sin2}(n_1)$,
   $S2 := P_n^T \left(\text{blkdiag}(M1, M2)\right) H_n$.

Output: $S2 = S_n^{II}$.

Algorithm 2.2. $\mathbf{sin4}(n)$

Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$.

1. *If* $n = 2$, *then*
$$S4 := \begin{bmatrix} \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} \end{bmatrix}.$$

2. *If* $n \geq 4$, *then*
$$M1 := \mathbf{sin2}(n_1),$$
$$M2 := \mathbf{sin2}(n_1),$$
$$L := V_n(\text{blkdiag}(M1, M2))\, Q_n,$$
$$S4 := P_n^T L.$$

Output: $S4 = S_n^{IV}$.

The transpose of DST-II is DST-III. Thus DST-III algorithm can be computed via algorithm (**2**.**1**). Observe that this algorithm executes recursively with DST-II and DST-IV algorithms.

Algorithm 2.3. $\mathbf{sin3}(n)$

Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$.

1. *If* $n = 2$, *then*
$$S3 := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

2. *If* $n \geq 4$, *then*
$$M1 := \mathbf{sin4}(n_1),$$
$$M2 := \mathbf{sin3}(n_1),$$
$$S3 := H_n^T\, (\text{blkdiag}(M1, M2))\, P_n.$$

Output: $S3 = S_n^{III}$.

Following [37], orthogonal factorizations for type I discrete sine transform matrix is given by

$$S_{n-1}^I = P_{n-1}^T \begin{bmatrix} C_{\frac{n}{2}}^{III} & 0 \\ 0 & C_{\frac{n}{2}-1}^I \end{bmatrix} \widehat{H}_{n-1}$$

Thus, the recursive algorithm for DST-I can be stated via algorithm (**2**.**4**). Note that this algorithm runs recursively with DST II-IV algorithms.

Algorithm 2.4. $\mathbf{sin1}(n-1)$

Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$.

1. *If* $n = 2$, *then*
$S1 := 1$.

2. *If* $n \geq 4$, *then*
$$M1 := \mathbf{sin3}(n_1),$$
$$M2 := \mathbf{sin1}(n_1 - 1),$$
$$S1 := P_{n-1}^T\, (\text{blkdiag}(M1, M2))\, \widehat{H}_{n-1}.$$

Output: $S1 = S_n^I$.

2.2 Efficient and completely recursive DST I-IV algorithms

In this section, we present fast, efficient, and completely recursive DST I-IV (say NDST I-IV) algorithms using DST I-IV algorithms stated via 2.1, 2.2, 2.3, and 2.4 i.e. we introduce DST I-IV algorithms having sparse, scaled orthogonal, rotational, rotational-reflection factors so that DST I-IV are orthogonal w. r. t. the scale factor $\frac{1}{\sqrt{n}}$. In order to reduce number of multiplications, we move the factor $\frac{1}{\sqrt{2}}$ in $H_n, \widehat{H}_{n-1}$, and V_n without changing the rotation-reflection matrix Q_n so that we compute $\sqrt{n}\,S_n^{II}, \sqrt{n}\,S_n^{IV}, \sqrt{n}\,S_n^{III}$, and $\sqrt{n}\,S_{n-1}^{I}$ respectively. Let us state the corresponding new algorithms via **nsin2**(n), **nsin4**(n), **nsin3**(n), and **nsin1**$(n-1)$ respectively.

Algorithm 2.5. **nsin2**(n)

Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$.

1. *If* $n = 2$, *then*
$$MS2 := \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$
2. *If* $n \geq 4$, *then*
$$\begin{aligned} M1 &:= \textbf{nsin4}\,(n_1), \\ M2 &:= \textbf{nsin2}\,(n_1), \\ MS2 &:= P_n^T\,(\text{blkdiag}(M1, M2))\left(\sqrt{2}\,H_n\right). \end{aligned}$$

Output: $MS2 = \sqrt{n}\,S_n^{II}$.

Algorithm 2.6. **nsin4**(n)

Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$.

1. *If* $n = 2$, *then*
$$MS4 := \sqrt{2}\begin{bmatrix} \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} \end{bmatrix}.$$
2. *If* $n \geq 4$, *then*
$$\begin{aligned} M1 &:= \textbf{nsin2}\,(n_1), \\ M2 &:= \textbf{nsin2}\,(n_1), \\ L &:= \left(\sqrt{2}V_n\right)(\text{blkdiag}(M1, M2))\,Q_n, \\ MS4 &:= P_n^T L. \end{aligned}$$

Output: $MS4 = \sqrt{n}\,S_n^{IV}$.

The fast, efficient, and completely recursive DST-III algorithm can be computed using the DST-II so that it runs recursively with **nsin2**(n) and **nsin4**(n) algorithms.

Algorithm 2.7. **nsin3**(n)

Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$.

1. *If* $n = 2$, *then*
$$MS3 := \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

2. *If* $n \geq 4$, *then*

$$M1 := \mathbf{nsin4}(n_1),$$
$$M2 := \mathbf{nsin3}(n_1),$$
$$MS3 := \left(\sqrt{2}\, H_n^T\right) (\text{blkdiag}(M1, M2))\, P_n.$$

Output: $MS3 = \sqrt{n}\, S_n^{III}$.

Finally, the fast, efficient, and completely recursive DST I algorithm can be stated as follows. Note that this algorithm runs recursively with **nsin2**(n) , **nsin4**(n), and **nsin3**(n) algorithms.

Algorithm 2.8. **nsin1**$(n-1)$

Input: $n = 2^t (t \geq 1)$, $n_1 = \frac{n}{2}$.

1. *If* $n = 2$, *then*
   $MS1 := 1$.

2. *If* $n \geq 4$, *then*

$$M1 := \mathbf{nsin3}(n_1),$$
$$M2 := \mathbf{nsin1}(n_1 - 1),$$
$$MS1 := P_{n-1}^T (\text{blkdiag}(M1, M2)) \left(\sqrt{2}\widehat{H}_{n-1}\right).$$

Output: $MS1 = \sqrt{n}\, S_{n-1}^I$.

2.3 Examples for computing efficient and completely recursive DST I-IV algorithms

Here we state examples for computing fast, efficient, and recursive DST I-IV algorithms having sparse, scaled orthogonal, rotational, and rotational-reflection matrix factorizations based on DST I-IV algorithms **nsin1**$(n-1)$, **nsin2**(n), **nsin3**(n), and **nsin4**(n) for $n = 8$. Later in section 4, we use the factorizations for DST I-IV matrices to develop and generalize n points signal flow graphs for DST I-IV algorithms.

Example 2.9. By following algorithms (**2.8**), (**2.5**), (**2.7**), and (**2.6**), the factorization for DST-I given by:

$$\begin{aligned} &\sqrt{8}\, S_7^I \\ &= P_7^T \begin{bmatrix} \sqrt{2}\, H_4^T & 0 \\ 0 & P_3^T \end{bmatrix} \begin{bmatrix} \sqrt{2}\, S_2^{IV} & 0 & 0 & 0 \\ 0 & \sqrt{2}\, S_2^{III} & 0 & 0 \\ 0 & 0 & \sqrt{2}\, S_2^{III} & 0 \\ 0 & 0 & 0 & \sqrt{2}\, S_1^I \end{bmatrix} \begin{bmatrix} P_4 & 0 \\ 0 & \sqrt{2}\, \widehat{H}_3 \end{bmatrix} \sqrt{2}\, \widehat{H}_7 \end{aligned} \tag{2}$$

where

$$P_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, S_1^I = 1, \sqrt{2}\, S_2^{III} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \sqrt{2}\, S_2^{IV} = \sqrt{2} \begin{bmatrix} \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} \end{bmatrix}$$

$$\sqrt{2}\, \widehat{H}_3 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & \sqrt{2} & 0 \\ 1 & 0 & -1 \end{bmatrix}, \sqrt{2}\, \widehat{H}_7 = \begin{bmatrix} I_2 & 0 & \tilde{I}_2 \\ 0 & \sqrt{2} & 0 \\ I_2 & 0 & -\tilde{I}_2 \end{bmatrix}$$

Example 2.10. By following algorithms (**2.5**) and (**2.6**), the factorization for DST-II given by:

$$\begin{aligned}
&\sqrt{8}\,S_8^{II} \\
&= P_8^T \begin{bmatrix} P_4^T & 0 \\ 0 & P_4^T \end{bmatrix} \begin{bmatrix} \sqrt{2}V_4 & 0 \\ 0 & I_4 \end{bmatrix} \begin{bmatrix} \sqrt{2}\,S_2^{II} & 0 & 0 & 0 \\ 0 & \sqrt{2}\,S_2^{II} & 0 & 0 \\ 0 & 0 & \sqrt{2}\,S_2^{IV} & 0 \\ 0 & 0 & 0 & \sqrt{2}\,S_2^{II} \end{bmatrix} \\
&\quad \begin{bmatrix} Q_4 & 0 \\ 0 & \sqrt{2}H_4 \end{bmatrix} \sqrt{2}H_8
\end{aligned} \tag{3}$$

where

$$\sqrt{2}V_4 = \begin{bmatrix} 0 & \sqrt{2} & 0 & 0 \\ 1 & 0 & -1 & 0 \\ -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & \sqrt{2} \end{bmatrix}, Q_4 = \begin{bmatrix} \sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \\ 0 & -\sin\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & 0 \\ 0 & -\cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ -\cos\frac{\pi}{16} & 0 & 0 & \sin\frac{\pi}{16} \end{bmatrix}$$

Example 2.11. By following algorithms (**2.5**), (**2.6**) and (**2.7**), the factorization for DST-III given by:

$$\begin{aligned}
&\sqrt{8}\,S_8^{III} \\
&= \sqrt{2}H_8^T \begin{bmatrix} P_4^T & 0 \\ 0 & I_4 \end{bmatrix} \begin{bmatrix} \sqrt{2}V_4 & 0 \\ 0 & \sqrt{2}H_4^T \end{bmatrix} \begin{bmatrix} \sqrt{2}\,S_2^{II} & 0 & 0 & 0 \\ 0 & \sqrt{2}\,S_2^{II} & 0 & 0 \\ 0 & 0 & \sqrt{2}\,S_2^{IV} & 0 \\ 0 & 0 & 0 & \sqrt{2}\,S_2^{III} \end{bmatrix} \\
&\quad \begin{bmatrix} Q_4 & 0 \\ 0 & P_4 \end{bmatrix} P_8
\end{aligned} \tag{4}$$

Example 2.12. By following algorithms (**2.5**) and (**2.6**), the factorization for DST-IV given by:

$$\begin{aligned}
&\sqrt{8}\,S_8^{IV} \\
&= P_8^T\,\sqrt{2}V_8 \begin{bmatrix} P_4^T & 0 \\ 0 & P_4^T \end{bmatrix} \begin{bmatrix} \sqrt{2}\,S_2^{IV} & 0 & 0 & 0 \\ 0 & \sqrt{2}\,S_2^{II} & 0 & 0 \\ 0 & 0 & \sqrt{2}\,S_2^{IV} & 0 \\ 0 & 0 & 0 & \sqrt{2}\,S_2^{II} \end{bmatrix} \\
&\quad \begin{bmatrix} \sqrt{2}H_4 & 0 \\ 0 & \sqrt{2}H_4 \end{bmatrix} Q_8
\end{aligned} \tag{5}$$

where

$$\sqrt{2}V_8 = \begin{bmatrix} 0 & 0 & 0 & \sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{2} \end{bmatrix}$$

$$Q_8 = \begin{bmatrix} \sin\frac{\pi}{32} & 0 & 0 & 0 & 0 & 0 & 0 & \cos\frac{\pi}{32} \\ 0 & -\sin\frac{3\pi}{32} & 0 & 0 & 0 & 0 & -\cos\frac{3\pi}{32} & \\ 0 & 0 & \sin\frac{5\pi}{32} & 0 & 0 & \cos\frac{5\pi}{32} & 0 & 0 \\ 0 & 0 & 0 & -\sin\frac{7\pi}{32} & -\cos\frac{7\pi}{32} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\cos\frac{7\pi}{32} & \sin\frac{7\pi}{32} & 0 & 0 & 0 \\ 0 & 0 & -\cos\frac{5\pi}{32} & 0 & 0 & \sin\frac{5\pi}{32} & 0 & 0 \\ 0 & -\cos\frac{3\pi}{32} & 0 & 0 & 0 & 0 & \sin\frac{3\pi}{32} & 0 \\ -\cos\frac{\pi}{32} & 0 & 0 & 0 & 0 & 0 & 0 & \sin\frac{\pi}{32} \end{bmatrix}$$

3 ARITHMETIC COMPLEXITY OF COMPUTING FAST, EFFICIENT, AND COMPLETELY RECURSIVE DST I-IV ALGORITHMS HAVING SPARSE AND SCALED ORTHOGONAL FACTORS

The number of additions and multiplications required to compute DST I-IV algorithms via **nsin1**$(n-1)$, **nsin2**(n), **nsin3**(n), **nsin4**(n) are considered in this section. The number of additions and multiplications required to compute, say length n, DST-II algorithm (**nsin2**(n)) are denoted by $\#a(\text{NDST-II},n)$ and $\#m(\text{NDST-II},n)$ respectively. Note that the multiplication of ± 1 and permutations are not counted. At the end of the section we illustrate numerical results based on the number of additions and multiplication required to compute these DST I-IV algorithms.

3.1 Arithmetic complexity of DST I-IV algorithms

Here we address the arithmetic cost of computing fast, efficient, and recursive DST I-IV algorithms having sparse, scaled orthogonal, rotational, and rotational-reflection factors. The complexity of computing these DST I-IV algorithms are expressed first by calculating the arithmetic complexity of DST-II algorithm and then using it to compute the complexity of DST-IV, DST-III and DST-I algorithms respectively.

Lemma 3.1. *Let $n = 2^t$ $(t \geq 2)$ be given. If DST-II algorithm (***nsin2**(n)*) is computed using algorithms* (**2.5**) *and* (**2.6**) *then the arithmetic cost of computing length n DST-II algorithm is given by*

$$\begin{aligned} \#a(\textit{NDST-II},n) &= \frac{4}{3}nt - \frac{8}{9}n - \frac{1}{9}(-1)^t + 1, \\ \#m(\textit{NDST-II},n) &= \frac{2}{3}nt + \frac{2}{9}n + \frac{7}{9}(-1)^t - 1. \end{aligned} \tag{6}$$

Proof. From algorithms (**2.5**) and (**2.6**)

$$\begin{aligned} \#a(\text{NDST-II},n) &= \#a\left(\text{NDST-II},\frac{n}{2}\right) + \#a\left(\text{NDST-IV},\frac{n}{2}\right) + \#a\left(\sqrt{2}H_n\right) \\ \#a\left(\text{NDST-IV},n\right) &= \#a\left(\sqrt{2}V_n\right) + 2\cdot\#a\left(\text{NDST-II},\frac{n}{2}\right) + \#a\left(Q_n\right) \end{aligned} \tag{7}$$

Referring the structures of H_n, V_n, and Q_n

$$\begin{aligned} \#a\left(\sqrt{2}H_n\right) = n,\ \#m\left(\sqrt{2}H_n\right) = 0 \\ \#a\left(\sqrt{2}V_n\right) = n-2,\ \#m\left(\sqrt{2}V_n\right) = 2 \\ \#a\left(Q_n\right) = n,\ \#m\left(Q_n\right) = 2n \end{aligned} \tag{8}$$

Thus

$$\#a(\text{NDST-II},n) = \#a\left(\text{NDST-II},\frac{n}{2}\right) + 2\cdot\#a\left(\text{NDST-II},\frac{n}{4}\right) + 2n - 2$$

Since $n = 2^t$ we can obtain the second order linear difference equation with respect to t

$$\#a(\text{NDST-II}, 2^t) - \#a\left(\text{NDST-II}, 2^{t-1}\right) - 2 \cdot \#a\left(\text{NDST-II}, 2^{t-2}\right) = 2^{t+1} - 2.$$

Solving the above under the initial conditions $\#a\left(\text{NDST-II}, 2\right) = 2$ and
$\#a\left(\text{NDST-II}, 4\right) = 8$, one can obtain

$$\#a(\text{NDST-II}, 2^t) = \frac{4}{3}nt - \frac{8}{9}n - \frac{1}{9}(-1)^t + 1.$$

Also using initial conditions $\#m\left(\text{NDST-II}, 2\right) = 0$ and $\#m\left(\text{NDST-II}, 4\right) = 6$, one can derive the analogous result for the number of multiplications as

$$\#m(\text{NDST-II}, 2^t) = \frac{2}{3}nt + \frac{2}{9}n + \frac{7}{9}(-1)^t - 1.$$

□

Corollary 3.2. *Let $n = 2^t$ $(t \geq 2)$ be given. If DST-IV algorithm ((***nsin4**(n)*)) is computed using algorithms* (**2.5**) *and* (**2.6**) *then the arithmetic cost of computing length n DST-IV algorithm is given by*

$$\begin{aligned} \#a(\textit{NDST-IV}, n) &= \frac{4}{3}nt - \frac{2}{9}n + \frac{2}{9}(-1)^t, \\ \#m(\textit{NDST-IV}, n) &= \frac{2}{3}nt + \frac{14}{9}n - \frac{14}{9}(-1)^t. \end{aligned} \tag{9}$$

Proof. The number of additions required to compute DST-IV algorithm (**2.6**) can be found by evaluating (7);

$$\begin{aligned} \#a\left(\text{NDST-IV}, n\right) &= \#a\left(\sqrt{2}\, V_n\right) + 2 \cdot \#a\left(\text{NDCT-II}, \frac{n}{2}\right) + \#a\left(Q_n\right) \\ &= 2 \cdot \#a\left(\text{NDCT-II}, \frac{n}{2}\right) + 2n - 2. \end{aligned}$$

Simplifying the above with (6) at $\frac{n}{2}$ yields

$$\#a(\text{NDCT-IV}, n) = \frac{4}{3}nt - \frac{2}{9}n + \frac{2}{9}(-1)^t.$$

Similarly, the number of multiplications required to compute new DST-IV algorithm can be found by evaluating (7) with (6) at $\frac{n}{2}$ which yields

$$\#m(\text{NDST-IV}, n) = \frac{2}{3}nt + \frac{14}{9}n - \frac{14}{9}(-1)^t.$$

□

The following result is trivial because the DST-III algorithm (**nsin3**(n)) was stated using the DST-II algorithm (**nsin2**(n)).

Corollary 3.3. *Let $n = 2^t$ $(t \geq 2)$ be given. If DST-III algorithm (***nsin3**(n)*) is computed using algorithms* (**2.5**), (**2.7**) *and* (**2.6**) *then the arithmetic cost of computing length n DST-III algorithm is given by*

$$\begin{aligned} \#a(\textit{NDST-III}, n) &= \frac{4}{3}nt - \frac{8}{9}n - \frac{1}{9}(-1)^t + 1, \\ \#m(\textit{NDST-III}, n) &= \frac{2}{3}nt + \frac{2}{9}n + \frac{7}{9}(-1)^t - 1. \end{aligned} \tag{10}$$

Remark 3.4. Using DST-III algorithm (**2.7**) and the arithmetic cost of DST-IV algorithm in corollary (3.2), it is possible to obtain the first order linear difference equation with respect to t. By solving the said equation under initial conditions $\#a(\text{NDST-III}, 2) = 2$ and $\#m(\text{NDST-III}, 2) = 0$ respectively, one can obtain the same results as in corollary (3.3) for the number of additions and multiplications involving in DST-III algorithm.

Lemma 3.5. *Let $n = 2^t$ $(t \geq 2)$ be given. If DST-I algorithm (**nsin1**$(n-1)$) is computed using algorithms* (**2.8**)*,* (**2.5**)*,* (**2.7**) *and* (**2.6**) *then the arithmetic cost of DST-I algorithms of length $n-1$ is given by*

$$\begin{aligned} \#a\,(\textit{NDST-I}, n-1) &= \frac{4}{3}nt - \frac{14}{9}n + \frac{1}{18}(-1)^t - t + \frac{3}{2} \\ \#m\,(\textit{NDST-I}, n-1) &= \frac{2}{3}nt - \frac{10}{9}n - \frac{7}{18}(-1)^t + \frac{3}{2} \end{aligned} \tag{11}$$

Proof. Referring DST-I algorithm (**2.8**)

$$\#a\,(\text{NDST-I}, n-1) = \#a\left(\text{NDST-I}, \frac{n}{2} - 1\right) + \#a\left(\text{NDST-III}, \frac{n}{2}\right) + \#a\left(\sqrt{2}\,\widehat{H}_{n-1}\right) \tag{12}$$

Following the structure of $\widehat{H}_{n-1}$ leads to

$$\#a\left(\sqrt{2}\,\widehat{H}_{n-1}\right) = n-2, \quad \#m\left(\sqrt{2}\,\widehat{H}_{n-1}\right) = 1 \tag{13}$$

Using arithmetic cost of DST-III (6) at $\frac{n}{2}$ and (13), we can rewrite (12)

$$\#a\,(\text{NDST-I}, n-1) = \#a\left(\text{NDST-I}, \frac{n}{2} - 1\right) + \left(\frac{2n}{3}(t-1) - \frac{4n}{9} + \frac{1}{9}(-1)^t + 1\right) + n - 2$$

Since $n = 2^t$ the above simplifies to the first order linear difference equation with respect to $t \geq 2$

$$\#a\,(\text{NDST-I}, 2^t - 1) - \#a\left(\text{NDST-I}, 2^{t-1} - 1\right) = \frac{2}{3}t \cdot 2^t - \frac{1}{9}2^t + \frac{1}{9}(-1)^t - 1$$

Solving the above first order linear difference equation (with respect to t) using the initial condition $\#a\,(\text{NDST-I}, 1) = 0$, one can obtain

$$\#a\,(\text{NDST-I}, 2^t - 1) = \frac{4}{3}nt - \frac{14}{9}n + \frac{1}{18}(-1)^t - t + \frac{3}{2}$$

Also using initial condition $\#m\,(\text{NDST-I}, 1) = 1$, one can derive the analogous result for the number of multiplications as

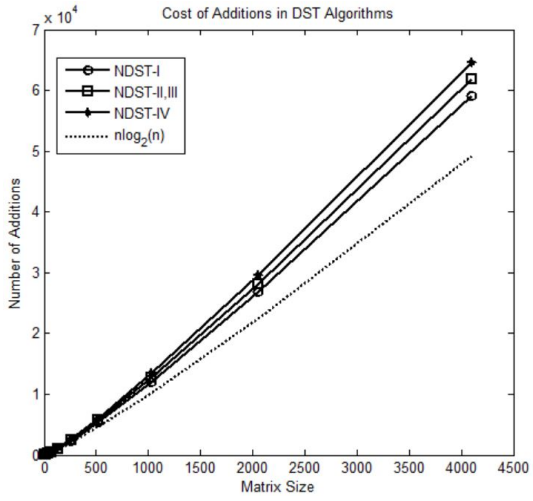
$$\#m\,(\text{NDST-I}, n-1) = \frac{2}{3}nt - \frac{10}{9}n - \frac{7}{18}(-1)^t + \frac{3}{2}$$

□

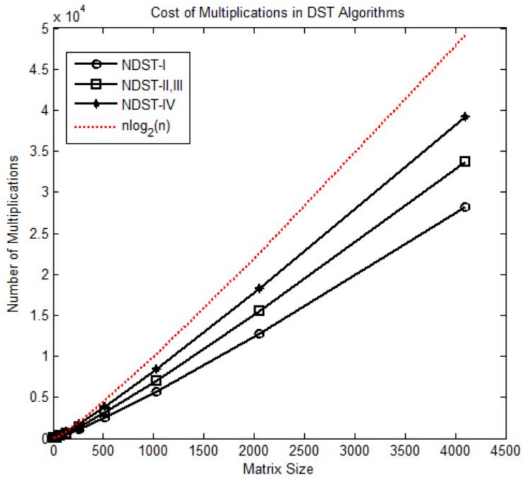
3.2 Numerical illustration of the arithmetic cost of computing fast, efficient, and completely recursive DST I-IV algorithms

The following numerical experiments are done to illustrate the number of additions and multiplications required to compute fast, efficient, and completely recursive DST I-IV algorithms having sparse, scaled orthogonal, rotational, and rotational-reflection factors. Matrices are used with the sizes from 8×8 to 4096×4096. These are implemented using MATLAB version 8.3 (R2014a).

Figure (1a) and (1b) illustrate the number of additions and multiplications required to compute DST I-IV algorithms corresponding to lemma 3.1, corollary 3.2, corollary 3.3, lemma 3.5 respectively with comparison to the $n \log n$ operations.



(a)



(b)

Figure 1: (1a) Number of additions in computing DST I-IV algorithms with $n \log n$ (1b) Number of multiplications in computing DST I-IV algorithms with $n \log n$

4 SIGNAL FLOW GRAPHS FOR FAST, EFFICIENT, AND COMPLETELY RECURSIVE DST I-IV ALGORITHMS

In this section we use signal flow graphs to elaborate fast, efficient, and completely recursive DST I-IV algorithms having sparse, scaled orthogonal, rotation, rotation-reflection, butterfly matrices for $n = 16$ and use those results to elaborate generalized n points flow graphs for these DST algorithms. Note that as stated in section 2, we have developed DST I-IV algorithms to reduce the cost of multiplications. Hence, based on the cheap cost of multiplication, we can develop signal flow graphs for these DST I-Iv only by using few multipliers which is opposed to the existing DST I-IV flow graphs.

These signal flow graphs of DST algorithms are drawn with respect to the decimation-in-frequency having the input signal $\mathbf{x}$ in order and output signal $\mathbf{y}$ in scrambled. So for a given input signal $\mathbf{x}$, this section present signal flow graphs for output signal $\mathbf{y} = \sqrt{n}S^{I}_{n-1}\,\mathbf{x}, \mathbf{y} = \sqrt{n}S^{II}_{n}\,\mathbf{x}, \mathbf{y} = \sqrt{n}S^{III}_{n}\,\mathbf{x}$, and $\mathbf{y} = \sqrt{n}S^{IV}_{n}\,\mathbf{x}$. As shown in the flow graphs, in each graph signal flows from the left to the right. However, it is possible to convert the decimation-in-frequency DST algorithms into decimation-in-time DST algorithms applying multiplications before additions and using the identical computation complexity (same as in section 3) as in decimation-in-frequency DST algorithms.

In each Figure from 2 until 9, multiplication with -1 is denoted by a dotted line and notations $\varepsilon := \frac{1}{\sqrt{2}}$, $C_{i,j} := \cos\frac{i\pi}{2^j}$, and $S_{i,j} = \sin\frac{i\pi}{2^j}$ for positive integers i and j are used.

4.1 Signal flow graphs for DST I-IV algorithms when $n = 16$

Let us state the signal flow graph for DST I-IV computed via **nsin1**$(n-1)$, **nsin2**(n), **nsin3**(n), **nsin4**(n). Here we draw the flow graphs for $n = 16$ with the help of factorizations of DST I-IV algorithms as stated in section 2.2.

Signal flow graphs for 15-point NDST-I ($4S^{I}_{15}$) and 16-point NDST II-IV ($4S^{II}_{16}$, $4S^{III}_{16}$, and $4S^{IV}_{16}$) algorithms are presented via Figures 2, 3, 4, and 5.

The flow graphs for 16-point NDST II-IV algorithms stated via Figures 3, 4, and 5, the input signals $\mathbf{x}$ are in order

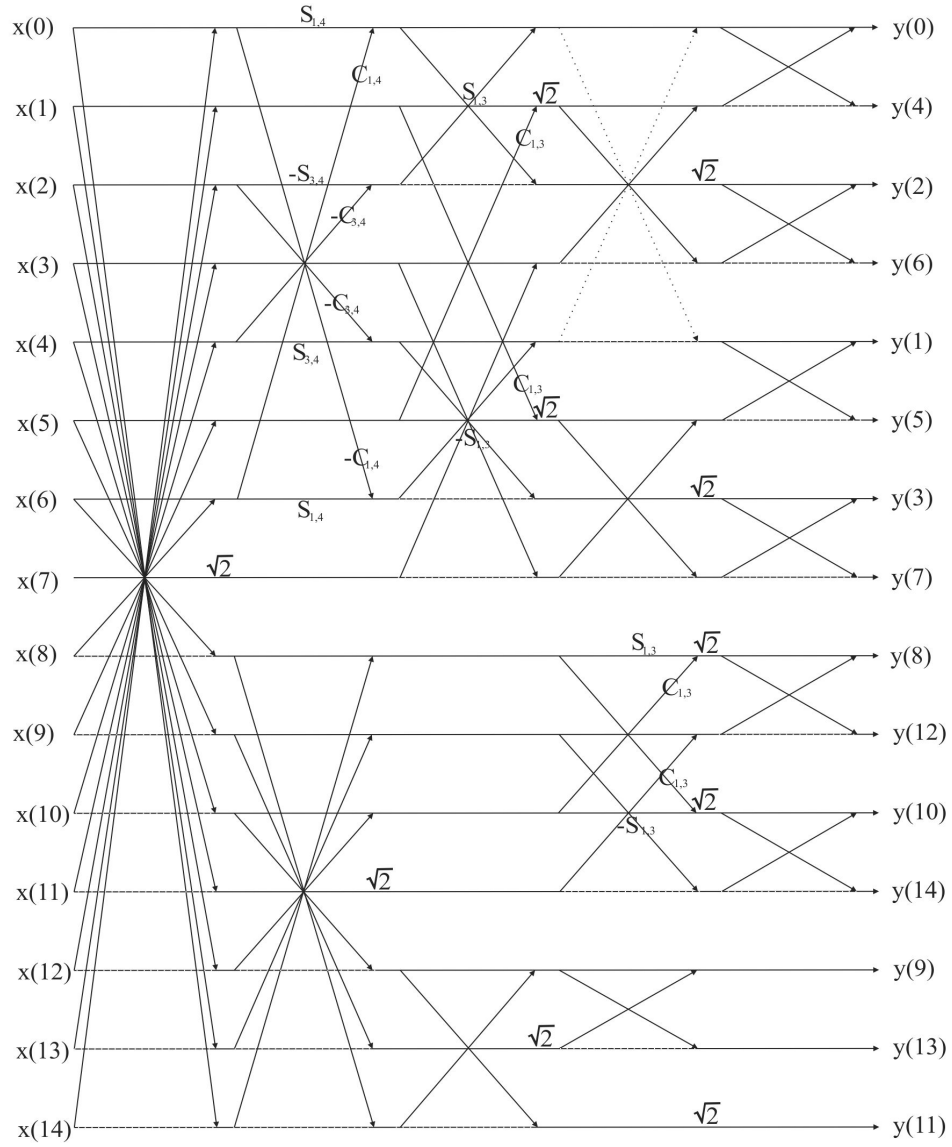


Figure 2: Flow graph for 15-point NDST-I ($4S^I_{15}$)

and output signals **y** are in bit-reversed order. Thus in bit-reversed order, each output index is represented as a binary number and the indices' bits are reversed.

4.2 Generalized signal flow graphs for DST I-IV algorithms

Here we present generalized $(n-1)$ points signal flow graph for DST-I and n points signal flow graphs for DST II-IV based on DST algorithms stated in the section 2.2 and the flow graphs drawn in the section 4.1. The generalized signal flow graphs for fast and completely recursive DST I-IV algorithms can be illustrated via Figures 6, 7, 8, and 9

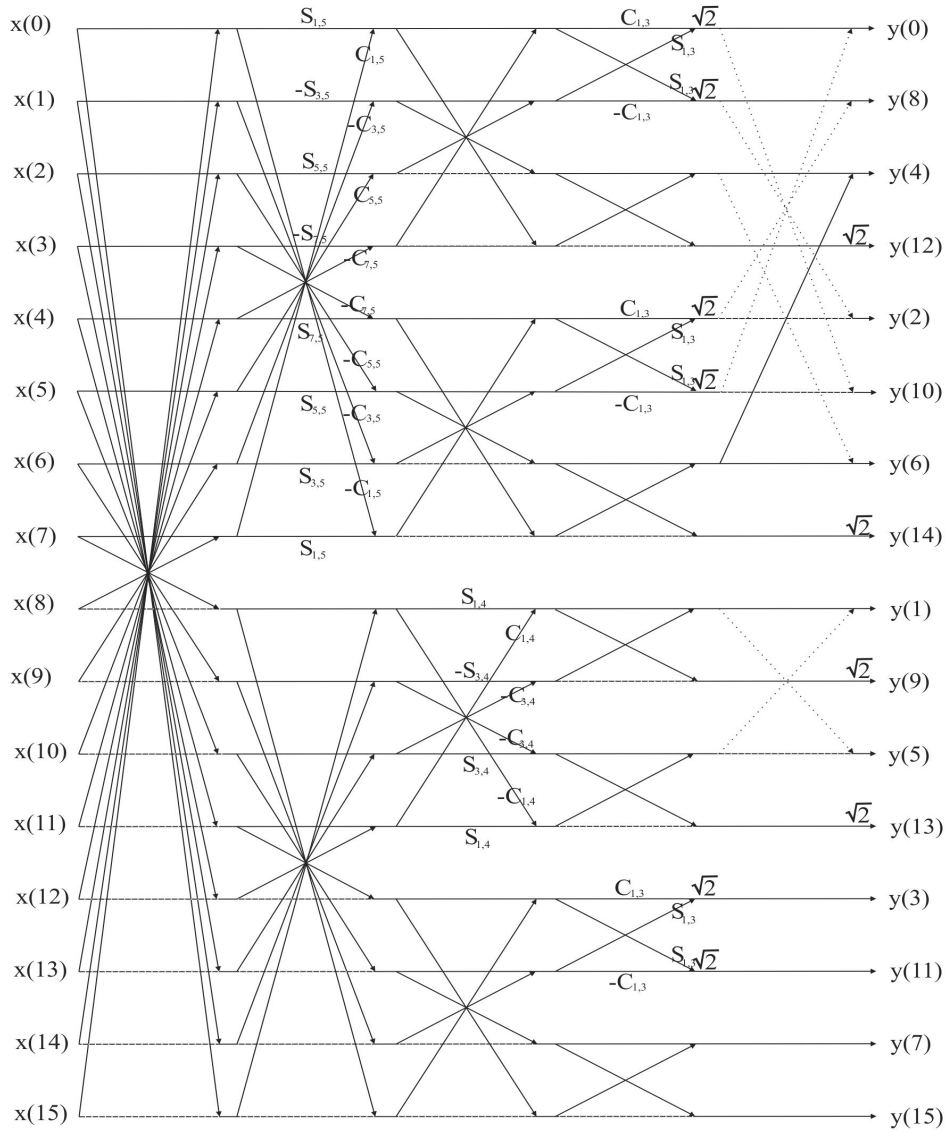


Figure 3: Flow graph for 16-point NDST-II ($4S_{16}^{II}$)

5 CONCLUSION

In this paper, we have provided fast, efficient, and completely recursive DST I-IV algorithms, which are solely defined via DST I-IV, having sparse, scaled orthogonal, rotational, rotational-reflection, and butterfly matrices while providing the corresponding arithmetic complexity of the said algorithms. Moreover, the language of signal flow graphs is used to show the connection between factors of these DST algorithms and $(n-1)$ points DST-I flow graph and n points DST II-IV flow graphs.

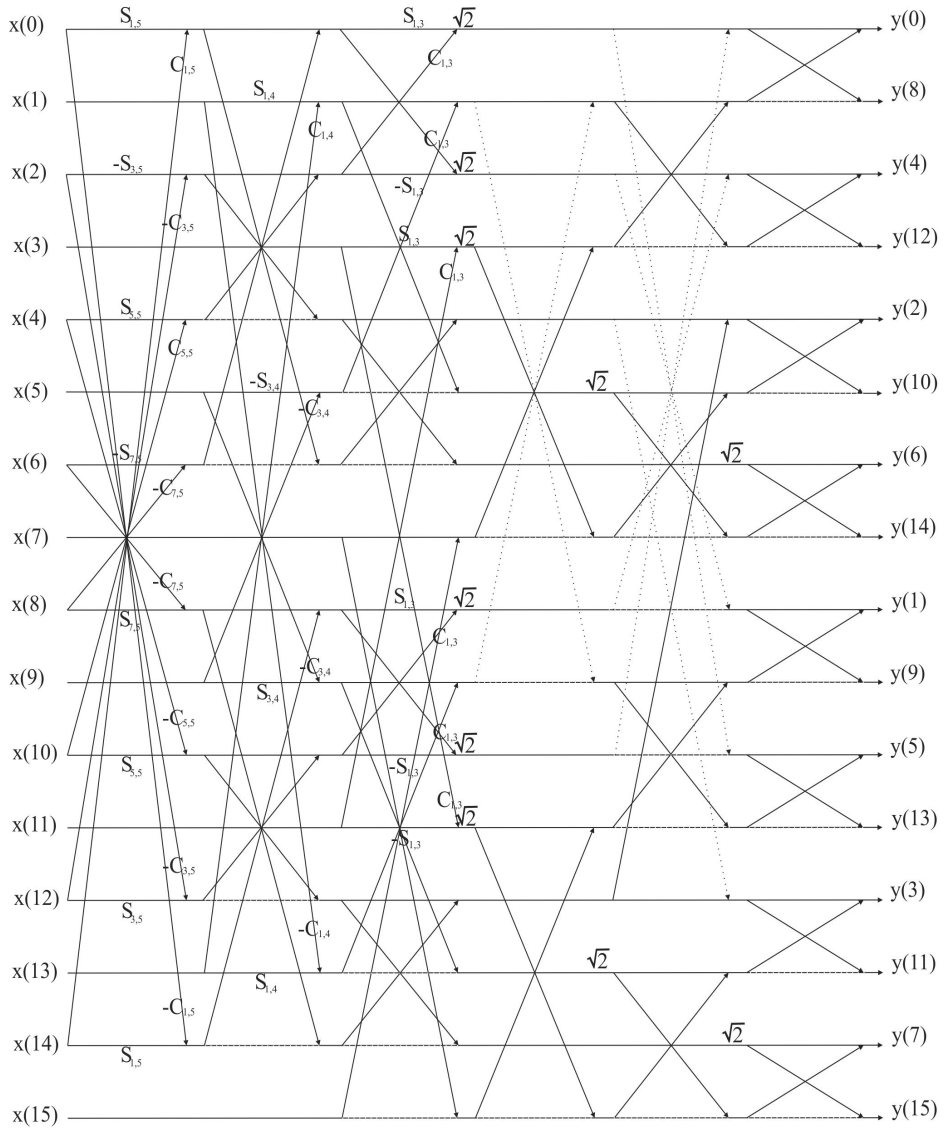


Figure 4: Flow graph for 16-point NDST-III ($4S_{16}^{III}$)

REFERENCES

[1] D. Belega, D. Dallet, and D. Petri. Accuracy of the Normalized Frequency Estimation of a Discrete-Time Sine-Wave by the Energy-Based Method. *IEEE Transactions on Instrumentation and Measurement*, 61(1):111-121, 2012.

[2] V. Britanak. New generalized conversion method of the MDCT and MDST coefficients in the frequency domain for arbitrary symmetric windowing function. *Digital Signal Processing*, 23:1783-1797, 2013.

[3] V. Britanak. A survey of efficient MDCT implementations in MP3 audio coding standard: Retrospective and state-of-the-art. *Signal Processing*, 91:624-672, 2011.

[4] V. Britanak. New universal rotation-based fast computational structures for an efficient implementation of the DCT-IV/DST-IV and analysis/synthesis MDCT/MDST filter banks. *Signal Processing*, 89:2213-2232, 2009.

[5] V. Britanak and K. R. Rao. Two-dimensional DCT/DST universal computational structure for 2*mn* block sizes. *IEEE Transactions on Signal Processing*, 48 (11):3250-3255, 2000.

[6] V. Britanak, P. C. Yip and K. R. Rao. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Academic Press, Great Britten, 2007.

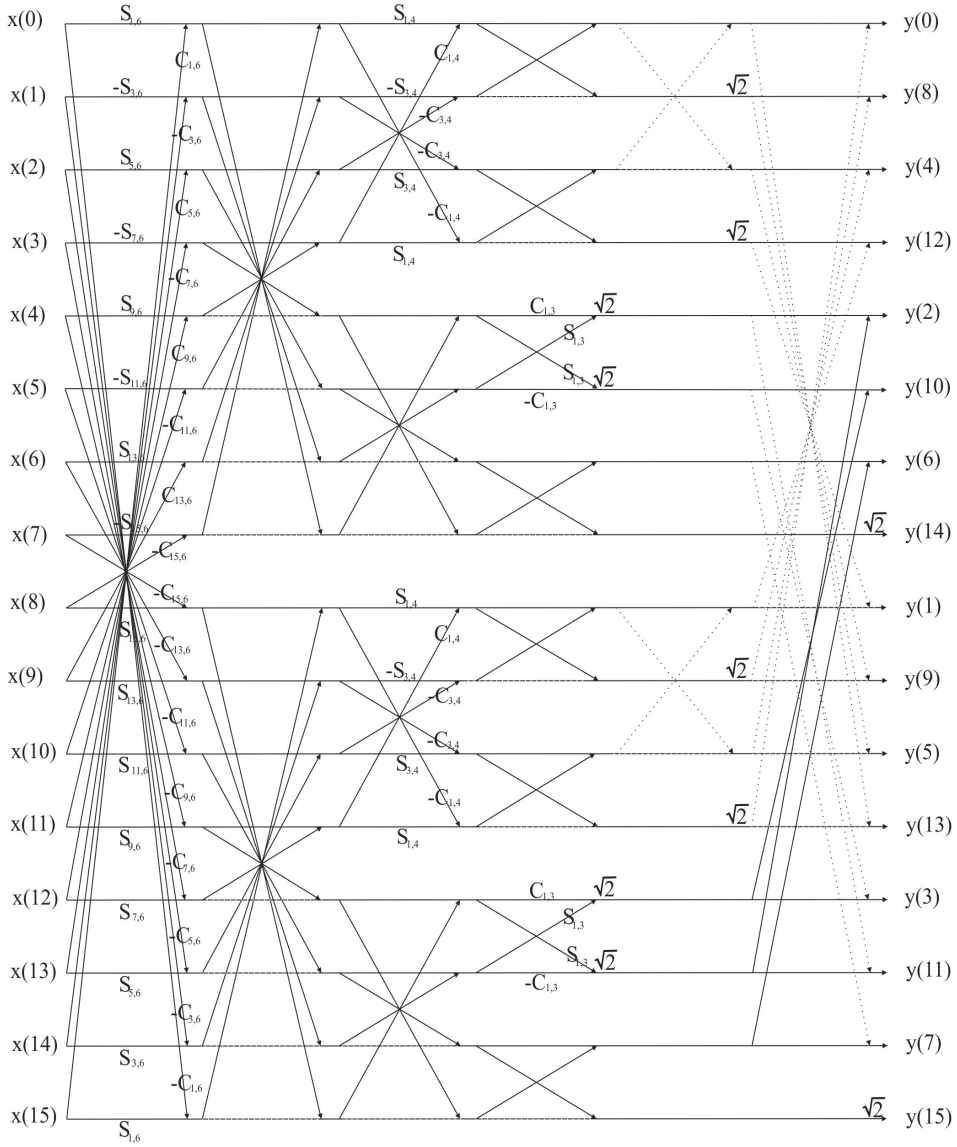


Figure 5: Flow graph for 16-point NDST-IV ($4S_{16}^{IV}$)

[7] S. Chakraborty and K. R. Rao, Fingerprint enhancement by directional filtering, In: *2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*,Thailand, May 2012, doi: 10.1109/ECTICon.2012.6254113.

[8] W. H. Chen, C.H. Smith, and S. Fralick. A fast computational algorithm for the discrete cosine transform. *IEEE Trans. Comm.*, 25:1004-1009, 1977.

[9] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297-301, 1965.

[10] S. Dhamija and P. Jain. Comparative Analysis for Discrete Sine Transform as a suitable method for noise estimation. *International Journal of Computer Science Issues*, 8(5):162-164, 2011.

[11] D. Fan, X. Meng, Y. Wang, X. Yang, X. Peng, W. He, G. Dong and H. Chen. Optical identity authentication scheme based on elliptic curve digital signature algorithm and phase retrieval algorithm. *Applied Optics*, 52(23):5645-5652, 2013.

[12] J. Han, A. Saxena, V. Melkote and K. Rose, Towards jointly optimal spatial prediction and adaptive transform in video/image coding, *IEEE Transactions on Image Processing*, 21(4):1874-1884, 2012.

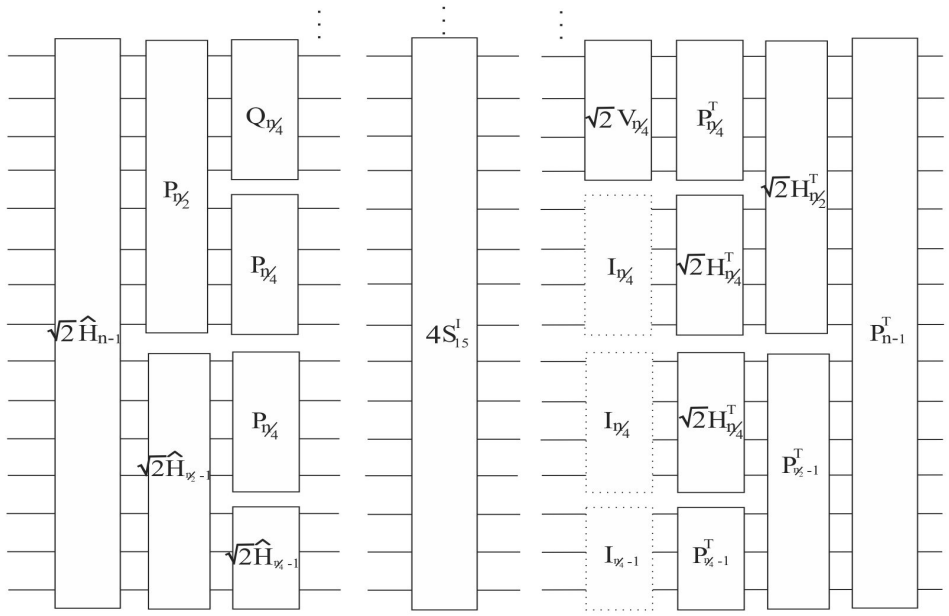


Figure 6: Flow graph for $n-1$ points NDST-I ($\sqrt{n}\,S^{I}_{n-1}$)

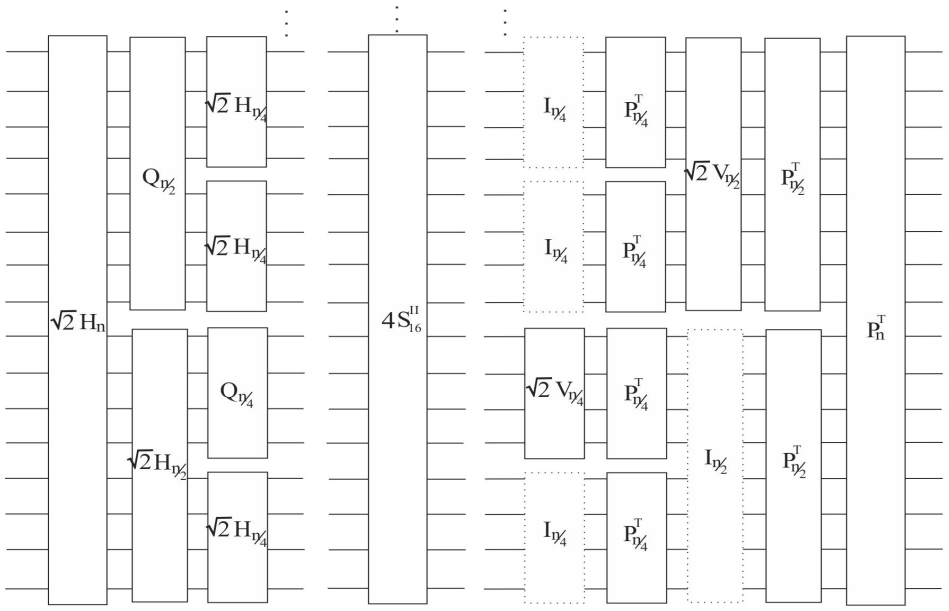


Figure 7: Flow graph for n points NDST-II ($\sqrt{n}\,S^{II}_{n}$)

[13] H. Huang, L. Xiao, and J. Liu. CORDIC-Based Unified Architectures for Computation of DCT/IDCT/DST/IDST. *Circuits Syst Signal Process*, 33:799-814, 2014.

[14] A. K. Jain. A sinusoidal family of unitary transform. *IEEE. Trans. Pattern Anal. Mach. Intell.*, PAMI-1:356-365, 1979.

[15] A. K. Jain. A fast Karhunen-Loeve transform for a class of stochastic processes. *IEEE. Trans. Commun.*, COM-24:1023-1029, 1976.

[16] P. Jain, B. Kumar, and S. B. Jain. Unified recursive structure for forward and inverse modified DCT/DST/DHT. *IETE Journal of Research*, 55(4): 180-191, 2009.

[17] T. Kailath and V. Olshevsky. Displacement structure approach to discrete trigonometric transform based preconditioners of G.Strang and T.Chan types. *Calcolo*, 33(3-4):191-208, 1996.

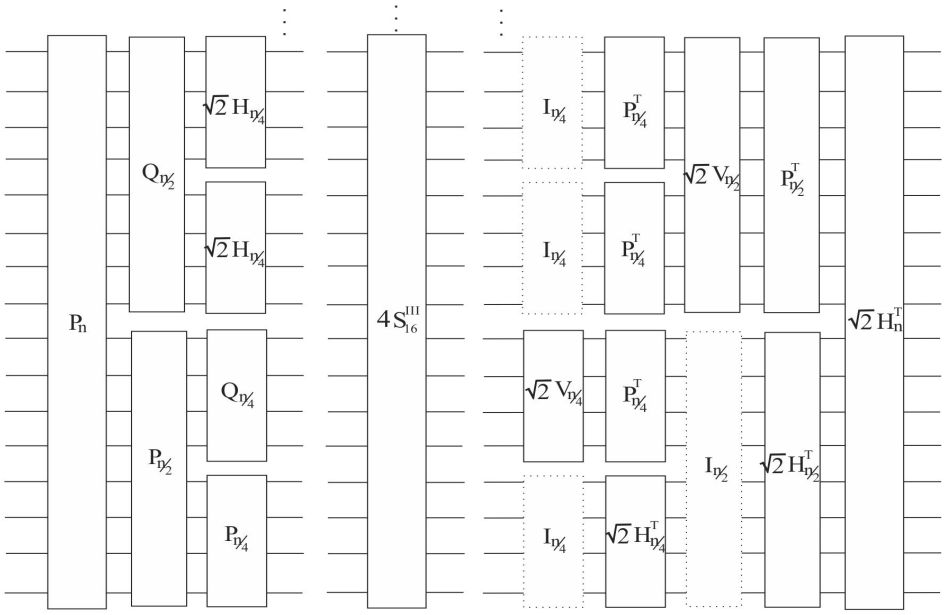


Figure 8: Flow graph for n points NDST-III ($\sqrt{n}\,S_n^{III}$)

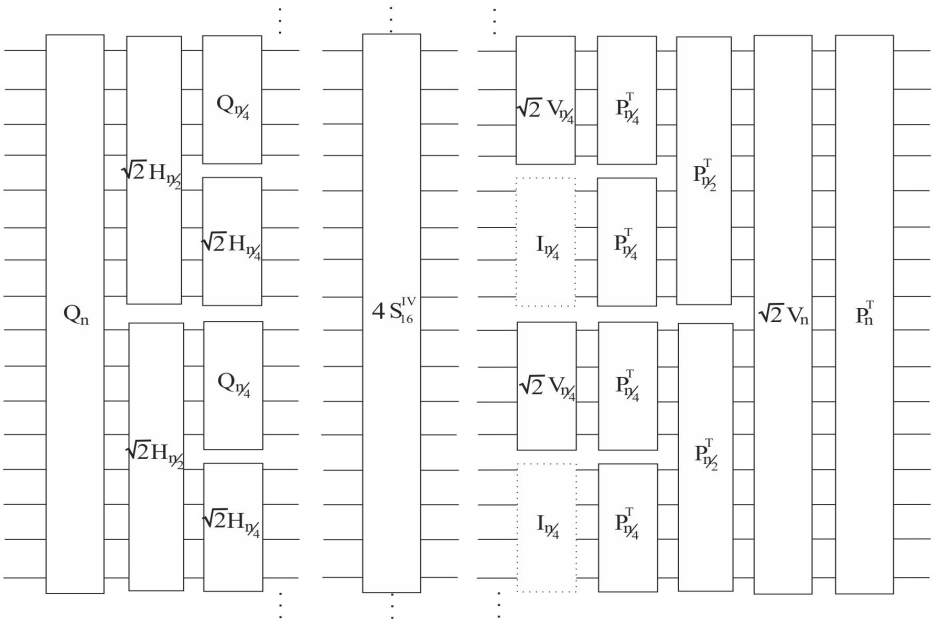


Figure 9: Flow graph for n points NDST-IV ($\sqrt{n}\,S_n^{IV}$)

[18] H. B. Kekre, T. K Sarode, and J. K. Save. Column Transform based Feature Generation for Classification of Image Database. *International Journal of Application or Innovation in Engineering and Management*, 3(7):172-181, 2014.

[19] H. B. Kekre, T. Sarode, and P. Natu. Performance Comparison of Hybrid Wavelet Transform Formed by Combination of Different Base Transforms with DCT on Image Compression. *I.J. Image, Graphics and Signal Processing*, 4:39-45, 2014.

[20] H. B. Kekre and J. K. Solanki. Comparative performance of various trigonometric unitary transforms for transform image coding. *Int. J. Electron.*, 44:305-315, 1978.

[21] D. Kim and K. R. Rao. 2D-DST scheme for image mirroring and rotation. *J. of Electronic Imaging*, 17(1), 2009, doi:10.1117/1.2885257.

[22] R. Kouassi, P. Gouton, and M. Paindavoine. Approximation of the Karhunen Loe‘ve transformation and its application to colour images. *Signal Processing: Image Communication*, 16:541-551, 2001.

[23] M.H. Lee, M.H.A. Khan, K.J. Kim, and D. Park. A Fast Hybrid Jacket-Hadamard Matrix Based Diagonal Block-wise Transform. *Signal Processing: Image Communication*, 29(1):49-65, 2014.

[24] J. Ma, G. Plonka, and M. Y. Hussaini. Compressive Video Sampling with Approximate Message Passing Decoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(9):1354-1364, 2012.

[25] S. A. Martucci. Symmetric convolution and the discrete sine and cosine transforms. *IEEE Trans. on Signal Processing*, 42(5): 1038-1051, 1994.

[26] A. Olshevsky, V. Olshevsky, and J. Wang. A comrade-matrix-based derivation of the eight versions of fast cosine and sine transforms. In: V. Olshevsky(ed.) *Contemporary Mathematics*, 323:119-150, AMS publications, Providence, RI, 2003.

[27] M. Püschel and J. M. Moura. The algebraic approach to the discrete cosine and sine transforms and their fast algorithms. *SIAM J. Comput.*, 32:1280-1316, 2003.

[28] G. Plonka and M. Tasche. Fast and Numerically stable algorithms for discrete cosine transforms *Linear Algebra and its Applications* 394:309-345, 2005.

[29] T.-T. Le. The design of optical signal transforms based on planar waveguides on a silicon on insulator platform. *International Journal of Engineering and Technology*, 2(3):245-251, 2010.

[30] T.-T. Le and L. W. Cahill. The design of 4×4 multimode interfaces coupler based microring resonators on an SOI platform. *Journal of Telecommunications and Information Technology*, 2:58-62, 2009.

[31] K.R. Rao, D.N. Kim, and J.J. Hwang. *Fast Fourier Transform: Algorithm and Applications*. Springer, New York, NY, 2010.

[32] K. R. Rao and P. Yip. *Discrete Cosine Transform:Algorithms, Advantages, Applications*, Academic Press, San Diego, CA, 1990.

[33] Y. A. Reznik. Relationship Between DCT-II, DCT-VI, and DST-VII Transforms. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 5642-5646, Vancouver, BC, 2013.

[34] N. Roma and L. Sousa. A tutorial overview on the properties of the discrete cosine transform for encoded image and video processing. *Signal Processing*, 91:2443-2464, 2011.

[35] L. Shen, C. Lu, F. Zhao, and W. Liu. Discrete Fourier Transformation for Seasonal-Factor Pattern Classification and Assignment. *IEEE Transactions on Intelligent Transportation Systems*, 14(2):511-516, 2013.

[36] S. M. Perera and V. Olshevsky. Stable, Recursive and Fast Algorithms for DST having Orthogonal Factors. *Journal of Coupled Systems Multiscale Dynamics* 1(3):358-371, 2013.

[37] S. M. Perera and V. Olshevsky. Fast and Stable Algorithms for Discrete Sine Transformations having Orthogonal Factors. In: M.G. Cojocaru, I. S. Kotsireas, R. N. Makarov, R. V. N. Melnik, and H. Shodiev(eds.) *Interdisciplinary Topics in Applied Mathematics, Modeling and Computational Science*, 117:347-354, Springer International, Switzerland, 2015.

[38] S. M. Perera. Signal Processing based on Stable radix-2 Discrete Cosine Transformation Algorithms having Orthogonal Factors. submitted to The Electronic Journal of Linear Algebra, 2016.

[39] G. Strang. *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley, MA, 1986.

[40] G. Strang. The Discrete Cosine Transform. *SIAM Review*, 41:135-147, 1999.

[41] G. Steidl and M. Tasche. A polynomial approach to fast algorithms for discrete Fourier-cosine and Fourier-sine transforms, *Math. Comput.*, 56:281-296, 1991.

[42] M. Tasche and H. Zeuner. Roundoff error analysis for fast trigonometry transforms. In: G. Anastassiou(ed.). *Handbook of Analytic-Computational Methods in Applied Mathematics*, 357-406, Chapman and Hall/CRC press, Boca Raton, FL, 2000.

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

[44] R. Veerla, Z. Zhang, and K. R. Rao. Advanced Image Coding and its Comparison with Various Still Image Codecs. *American Journal of Signal Processing*, 2(5):113-121, 2012.

[45] Y. Voronenko and M. PÃ$\frac{1}{4}$schel. Algebraic Signal Processing Theory:Cooley-Tukey Type Algorithms for Real DFTs. *Transactions on Signal Processing*, 57(1):1-19, 2009.

[46] Z. Wang. Fast algorithms for the discrete W transform and the discrete Fourier transform, *IEEE Trans. Acoust. Speech Signal Process*, 32:803-816, 1984.

[47] C. Y. Wu, A. R. D. Somervell, T. G. Haskell, and T. H. Barnes. Optical Sine transformation and image transmission by using square optical waveguide. *Optics Communications*, 175:27-32, 2000.

[48] P. Yip and K. R. Rao. A fast computational algorithm for the discrete sine transform. *IEEE Trans. Commun.*, 28(2):304-307, 1980.

[49] H. Yoshimura. Fingerprint templates with high recognition accuracy and high security generated by discrete fractional sine transform. In:*IEEE International Conference for Internet Technology and Secured Transactions*, 185-190, Abu Dhabi, 2011.