electronics

MDPI

Article

The Development of Fast DST-I Algorithms for Short-Length Input Sequences

Mateusz Raciborski [1,*], Aleksandr Cariow [2,*] and Jakub Bandach [2]

[1] Faculty of Computer Science and Telecommunications, Maritime University of Szczecin, Wały Chrobrego 1-2, 70-500 Szczecin, Poland
[2] Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, Żołnierska 49, 71-210 Szczecin, Poland
* Correspondence: m.raciborski@pm.szczecin.pl (M.R.); atariov@wi.zut.edu.pl (A.C.)

Abstract: The subject of this paper is the development of rationalized algorithms of discrete sinusoidal transform of type I for short sequences of length $N = 2, 3, 4, 5, 6, 7$, and 8. Here, by the word "rationalization", we mean the reduction of the number of arithmetic operations required to implement the algorithms. The arithmetic complexity of the developed algorithms is presented in the final table. For each algorithm, we also provide data flow graphs demonstrating the space–time structure of the computational processes. The algorithms were tested to verify their validity using MATLAB software (version R2023).

Keywords: complexity theory; digital signal processing; discrete sine transform; DST-I; matrix decomposition; signal processing algorithms

check for updates

Citation: Raciborski, M.; Cariow, A.; Bandach, J. The Development of Fast DST-I Algorithms for Short-Length Input Sequences. *Electronics* **2024**, *13*, 5056. https://doi.org/10.3390/electronics13245056

Academic Editor: Costas Psychalinos

Received: 30 November 2024
Revised: 17 December 2024
Accepted: 18 December 2024
Published: 23 December 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Discrete trigonometric (sine and cosine) transforms [1–3] are used today in many digital signal processing applications [4–15]. There are eight types of cosine and eight types of sine transforms. A list of all 16 types of discrete cosine and sine transforms can be found, for example, in [3,16]. Undoubtedly, the most popular are discrete cosine transforms. Discrete sine transforms are less popular. Nevertheless, many articles are devoted to the use of discrete sine transforms [2]. It is well known that any linear transform can be represented as a matrix–vector product. Computing this product directly takes a long time because the multiplicative complexity of this operation is proportional to the square of the order of the matrix. Multiplication is the most expensive of all arithmetic operations, apart from division, and therefore, developers of efficient algorithms are focused on reducing the number of multiplication operations. Traditionally, algorithms with reduced computational complexity are called fast algorithms. Over five decades, many fast algorithms for the efficient computation of one-/two-dimensional discrete trigonometric transform have been developed [17–24].

Among other things, the development of efficient algorithms for the implementation of small-sized discrete trigonometric transforms is of particular interest. Algorithms for some types of small-size discrete trigonometric transforms have already been developed [25–28]. Among other discrete trigonometric transforms, the discrete sine transform type I (DST-I) [29] is also an important tool in signal analysis and data processing, such as noise estimation and image denoising [5,6], discrete multi-tone systems [4,30,31], audio watermarking [11], EEG signal classification [12], and noisy speech enhancement [32], and others [33,34]. However, the purpose of our paper is not to justify the application of DST-I. We believe that since it has been defined, its feasibility has already been proven and is beyond doubt [2]. We focus on rationalizing the computation of this transform for the case of short input data sequence lengths. Thus, this paper is devoted to the design of DST-I algorithms with reduced multiplicative complexity for input data sequences of length $N = 2, 3, 4, 5, 6, 7, 8$.

2. Materials and Methods

First, we would like to present the sources we used while working on the solutions for the discrete sine transform type I (DST-I) algorithms presented here. We believe that they will help us better understand the essence of DST-I and the methodology for constructing our solutions.

DFT, DCT, MDCT, DST, and Fourier spectrum analysis of the signal were performed in [35]. An excellent description of the DCT and DST digital signal processing algorithms has been provided by Nirajan Pant [36]. An extensive description of the different types of DFT, DCT, and DST was given by Perera [37]. The paper in [38] presents a systematic methodology for deriving and classifying fast algorithms for linear transformations.

Our methodology, which we use to achieve the solutions presented in this paper, is based on specific matrix structures. These structures are included in "Table 3. The specific structures of N × N matrix patterns" in the paper [39]. Our research work is to analyze the matrix that we want to rationalize, and then set the values in that matrix so that the best possible matrix pattern can be applied to it.

Moreover, in the literature on the subject, we found many different ways of writing the expression for DST-I [3,18,40–42]. We found the DST-I notation method most similar to the one we use in the work of [43]. So, DST-I can be expressed as follows:

$$y_k = \sqrt{\frac{2}{N+1}} \sum_{n=0}^{N-1} x_n \sin\left(\frac{\pi(k+1)(n+1)}{N+1}\right) \tag{1}$$

where
$k = 0, 1, \ldots, N-1$,
y_k—the output data after performing DST-I,
x_n—input data, and
N—number of signal samples.

Using matrix notation, we can write DST-I as follows:

$$\mathbf{Y}_{N\times 1} = \mathbf{C}_N \mathbf{X}_{N\times 1} \tag{2}$$

where

$$\mathbf{Y}_{N\times 1} = [y_0, y_1, \ldots, y_{N-1}]^{\mathrm{T}}, \quad \mathbf{X}_{N\times 1} = [x_0, x_1, \ldots, x_{N-1}]^{\mathrm{T}},$$

$$\mathbf{C}_N = \begin{bmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,N-1} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N-1,0} & c_{N-1,1} & \cdots & c_{N-1,N-1} \end{bmatrix}, \tag{3}$$

$$c_{k,n} = \sqrt{\frac{2}{N+1}} \sin\left(\frac{\pi(k+1)(n+1)}{N+1}\right), \quad \text{for } k, n = 0, 1, \ldots, N-1. \tag{4}$$

DST-I in matrix notation is as takes after

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \sqrt{\frac{2}{N+1}} \begin{bmatrix} \sin\left(\frac{\pi}{N+1}\right) & \sin\left(\frac{2\pi}{N+1}\right) & \cdots & \sin\left(\frac{N\pi}{N+1}\right) \\ \sin\left(\frac{2\pi}{N+1}\right) & \sin\left(\frac{4\pi}{N+1}\right) & \cdots & \sin\left(\frac{2N\pi}{N+1}\right) \\ \vdots & \vdots & \ddots & \vdots \\ \sin\left(\frac{N\pi}{N+1}\right) & \sin\left(\frac{2N\pi}{N+1}\right) & \cdots & \sin\left(\frac{N^2\pi}{N+1}\right) \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}. \tag{5}$$

We use the following notation in our work: [39,44]:

- $\mathbf{I}_N$—is an order N identity matrix;

- $\mathbf{H}_2$—a 2×2 Hadamard matrix;
- ⊗—the Kronecker product of two matrices; and
- ⊕—direct sum of two matrices.

An empty cell in matrix means that there is a zero there. Multipliers are denoted as $s_m^{(N)}$. In the graphs, we do not enter a superscript in order to maintain their clarity and elegance.

3. Two-Point DST-I Solution

The following is the matrix form expression for two-point DST-I:

$$\mathbf{Y}_{2\times1} = \mathbf{C}_2\mathbf{X}_{2\times1} \tag{6}$$

where

$$\mathbf{Y}_{2\times1} = [y_0, y_1]^{\mathrm{T}}, \quad \mathbf{X}_{2\times1} = [x_0, x_1]^{\mathrm{T}},$$

$$\mathbf{C}_2 = \begin{bmatrix} a_2 & a_2 \\ a_2 & -a_2 \end{bmatrix}, \quad a_2 = 0.7071.$$

Presently, we are able determine the final expression for DST-I for N = 2:

$$\mathbf{Y}_{2\times1} = \mathbf{H}_2\mathbf{D}_2^{(0)}\mathbf{X}_{2\times1} \tag{7}$$

where

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{D}_2^{(0)} = \mathrm{diag}\left(s_0^{(2)}, s_1^{(2)}\right), \quad s_0^{(2)} = s_1^{(2)} = a_2.$$

The data flow graph for our solution for two-point DST-I is shown in Figure 1. The naive, direct computation requires 2 additions and 4 multiplications. As can be observed, our solution uses 2 additions and 2 multiplications, reducing the number of multiplications from 4 to 2.

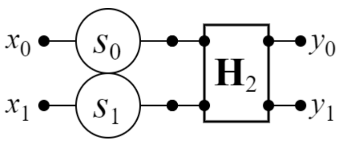

Figure 1. The proposed solution's data flow graph for two-point DST-I computation.

4. Three-Point DST-I Solution

The following is the matrix form expression for three-point DST-I:

$$\mathbf{Y}_{3\times1} = \mathbf{C}_3\mathbf{X}_{3\times1} \tag{8}$$

where

$$\mathbf{Y}_{3\times1} = [y_0, y_1, y_2]^{\mathrm{T}}, \quad \mathbf{X}_{3\times1} = [x_0, x_1, x_2]^{\mathrm{T}},$$

$$\mathbf{C}_3 = \begin{bmatrix} a_3 & b_3 & a_3 \\ b_3 & 0 & -b_3 \\ a_3 & -b_3 & a_3 \end{bmatrix}, \quad a_3 = 0.5, \quad b_3 = 0.7071.$$

Now, we will divide the matrix $\mathbf{C}_3$ into two parts:

$$\mathbf{C}_3 = \mathbf{C}_3^{(\mathrm{a})} + \mathbf{C}_3^{(\mathrm{b})} \tag{9}$$

where

$$\mathbf{C}_3^{(\mathrm{a})} = \begin{bmatrix} a_3 & & a_3 \\ & & \\ a_3 & & a_3 \end{bmatrix}, \quad \mathbf{C}_3^{(\mathrm{b})} = \begin{bmatrix} & b_3 & \\ b_3 & & -b_3 \\ & -b_3 & \end{bmatrix}.$$

Matrix $\mathbf{C}_3^{(a)}$ after omitting terms equal to zero is as takes after

$$\mathbf{C}_2' = \left[\begin{array}{c|c} a_3 & a_3 \\ \hline a_3 & a_3 \end{array}\right]. \tag{10}$$

Presently, we are able determine the final expression for DST-I for $N = 3$:

$$\mathbf{Y}_{3\times 1} = \mathbf{W}_3^{(1)}\mathbf{D}_3\mathbf{W}_3^{(0)}\mathbf{X}_{3\times 1} \tag{11}$$

where

$$\mathbf{W}_3^{(0)} = \left[\begin{array}{c|c|c} 1 & & 1 \\ \hline & 1 & \\ \hline 1 & & -1 \end{array}\right], \quad \mathbf{D}_3 = \mathrm{diag}\left(s_0^{(3)}, s_1^{(3)}, s_2^{(3)}\right),$$

$$s_0^{(3)} = a_3, \quad s_1^{(3)} = s_2^{(3)} = b_3, \quad \mathbf{W}_3^{(1)} = \left[\begin{array}{c|c|c} 1 & 1 & \\ \hline & & 1 \\ \hline 1 & -1 & \end{array}\right].$$

The data flow graph for our solution for three-point DST-I is shown in Figure 2. The naive, direct computation requires 5 additions and 4 multiplications. As can be observed, our solution uses 4 additions and 2 multiplications, reducing the number of additions from 5 to 4 and the number of multiplications from 4 to 2.

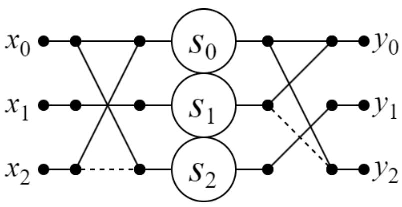

Figure 2. The proposed solution's data flow graph for three-point DST-I computation.

5. Four-Point DST-I Solution

The following is the matrix form expression for four-point DST-I:

$$\mathbf{Y}_{4\times 1} = \mathbf{C}_4\mathbf{X}_{4\times 1} \tag{12}$$

where

$$\mathbf{Y}_{4\times 1} = [y_0, y_1, y_2, y_3]^{\mathrm{T}}, \quad \mathbf{X}_{4\times 1} = [x_0, x_1, x_2, x_3]^{\mathrm{T}}, \quad \mathbf{C}_4 = \left[\begin{array}{c|c|c|c} a_4 & b_4 & b_4 & a_4 \\ \hline b_4 & a_4 & -a_4 & -b_4 \\ \hline b_4 & -a_4 & -a_4 & b_4 \\ \hline a_4 & -b_4 & b_4 & -a_4 \end{array}\right], \quad a_4 = 0.3717, \quad b_4 = 0.6015.$$

Now, we swap the columns and rows in matrix $\mathbf{C}_4$ to group the a_4 and b_4 terms so that the obtained matrix is consistent with the matrix pattern [39]:

$$\left[\begin{array}{c|c} \mathbf{A}_2 & \mathbf{B}_2 \\ \hline \mathbf{B}_2 & -\mathbf{A}_2 \end{array}\right] \quad \text{where} \quad \mathbf{A}_2 = \left[\begin{array}{c|c} a_4 & a_4 \\ \hline a_4 & -a_4 \end{array}\right], \quad \mathbf{B}_2 = \left[\begin{array}{c|c} b_4 & b_4 \\ \hline b_4 & -b_4 \end{array}\right].$$

This is accomplished by the permutation $\pi_4^{(0)}$:

$$\pi_4^{(0)} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \end{pmatrix}. \tag{13}$$

Considering this, we may obtain the following expression for the first stage of decomposition:

$$\mathbf{Y}_{4\times 1} = \mathbf{P}_4^{(\pi_4^{(0)})}\mathbf{W}_{4\times 6}\mathbf{D}_6^{(0)}\mathbf{W}_{6\times 4}\mathbf{P}_4^{(\pi_4^{(0)})}\mathbf{X}_{4\times 1} \tag{14}$$

where

$$\mathbf{P}_4^{(\pi_4^{(0)})} = \begin{bmatrix} 1 & & & \\ & & & 1 \\ & & 1 & \\ & 1 & & \end{bmatrix}, \quad \mathbf{W}_{6\times 4} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \\ 1 & & 1 & \\ & 1 & & 1 \end{bmatrix},$$

$$\mathbf{D}_6^{(0)} = \begin{bmatrix} \mathbf{F}_2 & & \\ & \mathbf{G}_2 & \\ & & \mathbf{B}_2 \end{bmatrix}, \quad \mathbf{F}_2 = \mathbf{A}_2 - \mathbf{B}_2, \quad \mathbf{G}_2 = -\mathbf{A}_2 - \mathbf{B}_2,$$

$$\mathbf{W}_{4\times 6} = \begin{bmatrix} 1 & & & & 1 & \\ & 1 & & & & 1 \\ & & 1 & & 1 & \\ & & & 1 & & 1 \end{bmatrix}.$$

The matrices $\mathbf{F}_2$ and $\mathbf{G}_2$ take after

$$\mathbf{F}_2 = \begin{bmatrix} a_4 - b_4 & a_4 - b_4 \\ a_4 - b_4 & -a_4 + b_4 \end{bmatrix}, \quad \mathbf{G}_2 = \begin{bmatrix} -a_4 - b_4 & -a_4 - b_4 \\ -a_4 - b_4 & a_4 + b_4 \end{bmatrix}.$$

As we can see above, these matrices do not require any operations to reduce complexity, because they both follow this pattern:

$$\begin{bmatrix} a & a \\ a & -a \end{bmatrix}.$$

Presently, we are able determine the final expression for DST-I for N = 4:

$$\mathbf{Y}_{4\times 1} = \mathbf{P}_4^{(\pi_4^{(0)})}\mathbf{W}_{4\times 6}\mathbf{D}_6^{(1)}\mathbf{W}_6^{(0)}\mathbf{W}_{6\times 4}\mathbf{P}_4^{(\pi_4^{(0)})}\mathbf{X}_{4\times 1} \tag{15}$$

where

$$\mathbf{W}_6^{(0)} = \mathbf{H}_2 \oplus \mathbf{H}_2 \oplus \mathbf{H}_2 = \begin{bmatrix} \mathbf{H}_2 & & \\ & \mathbf{H}_2 & \\ & & \mathbf{H}_2 \end{bmatrix}, \quad \mathbf{D}_6^{(1)} = \mathrm{diag}\left(s_0^{(4)}, s_1^{(4)}, \ldots, s_5^{(4)}\right),$$

$$s_0^{(4)} = s_1^{(4)} = a_4 - b_4, \quad s_2^{(4)} = s_3^{(4)} = -a_4 - b_4, \quad s_4^{(4)} = s_5^{(4)} = b_4.$$

The data flow graph for our solution for four-point DST-I is shown in Figure 3. The naive, direct computation requires 12 additions and 16 multiplications. As can be observed, our solution uses 12 additions and 6 multiplications, reducing the number of multiplications from 16 to 6.

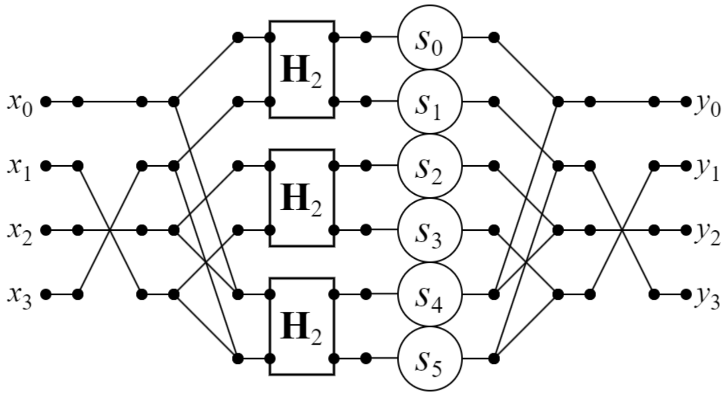

Figure 3. The proposed solution's data flow graph for four-point DST-I computation.

6. Five-Point DST-I Solution

The following is the matrix form expression for five-point DST-I:

$$\mathbf{Y}_{5\times1} = \mathbf{C}_5\mathbf{X}_{5\times1} \tag{16}$$

where

$$\mathbf{Y}_{5\times1} = [y_0, y_1, y_2, y_3, y_4]^{\mathrm{T}}, \quad \mathbf{X}_{5\times1} = [x_0, x_1, x_2, x_3, x_4]^{\mathrm{T}},$$

$$\mathbf{C}_5 = \begin{bmatrix} a_5 & b_5 & c_5 & b_5 & a_5 \\ b_5 & b_5 & 0 & -b_5 & -b_5 \\ c_5 & 0 & -c_5 & 0 & c_5 \\ b_5 & -b_5 & 0 & b_5 & -b_5 \\ a_5 & -b_5 & c_5 & -b_5 & a_5 \end{bmatrix}, \quad \begin{array}{l} a_5 = 0.2887, \\ b_5 = 0.5, \\ c_5 = 0.5774. \end{array}$$

Now, we swap the columns and rows in matrix $\mathbf{C}_5$ to be able to perform operations that are beneficial to us. This is accomplished by the permutation $\pi_5^{(0)}$ for the columns and permutation $\pi_5^{(1)}$ for the rows:

$$\pi_5^{(0)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 5 & 3 & 4 & 2 \end{pmatrix}, \quad \pi_5^{(1)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 3 & 5 & 2 \end{pmatrix}. \tag{17}$$

After permutations, we divide the matrix $\mathbf{C}_5$ into two parts:

$$\mathbf{C}_5 = \mathbf{C}_5^{(\mathrm{a})} + \mathbf{C}_5^{(\mathrm{b})} \tag{18}$$

where

$$\mathbf{C}_5^{(\mathrm{a})} = \begin{bmatrix} a_5 & a_5 & & b_5 & b_5 \\ b_5 & -b_5 & & b_5 & -b_5 \\ & & & & \\ a_5 & a_5 & & -b_5 & -b_5 \\ b_5 & -b_5 & & -b_5 & b_5 \end{bmatrix}, \quad \mathbf{C}_5^{(\mathrm{b})} = \begin{bmatrix} & & c_5 & & \\ & & & & \\ c_5 & c_5 & -c_5 & & \\ & & c_5 & & \\ & & & & \end{bmatrix}.$$

The matrix $\mathbf{C}_5^{(\mathrm{a})}$ after omitting terms equal to zero is as follows:

$$\mathbf{C}_4 = \begin{bmatrix} a_5 & a_5 & b_5 & b_5 \\ b_5 & -b_5 & b_5 & -b_5 \\ a_5 & a_5 & -b_5 & -b_5 \\ b_5 & -b_5 & -b_5 & b_5 \end{bmatrix}.$$

The matrix $\mathbf{C}_4$ matches the matrix pattern:

$$\begin{bmatrix} \mathbf{A}_2 & \mathbf{B}_2 \\ \mathbf{A}_2 & -\mathbf{B}_2 \end{bmatrix} \quad \text{where} \quad \mathbf{A}_2 = \begin{bmatrix} a_5 & a_5 \\ b_5 & -b_5 \end{bmatrix}, \quad \mathbf{B}_2 = \begin{bmatrix} b_5 & b_5 \\ b_5 & -b_5 \end{bmatrix}.$$

Having stated this, we can write down the following expression for the first stage of rationalization:

$$\mathbf{Y}_{5\times1} = \mathbf{P}_5^{(\pi_5^{(1)})}\mathbf{W}_{5\times7}\mathbf{W}_{7\times6}\mathbf{D}_6^{(2)}\mathbf{W}_{6\times7}\mathbf{W}_{7\times5}^{(0)}\mathbf{P}_5^{(\pi_5^{(0)})}\mathbf{X}_{5\times1} \tag{19}$$

where

$$
\mathbf{P}_5^{(\pi_5^{(0)})} = \begin{bmatrix} 1 & & & & \\ & & & & 1 \\ & & 1 & & \\ & & & 1 & \\ & 1 & & & \end{bmatrix}, \quad \mathbf{W}_{7\times 5}^{(0)} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & & 1 & \\ & & & & 1 \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \end{bmatrix},
$$

$$
\mathbf{W}_{6\times 7} = \begin{bmatrix} \mathbf{I}_4 & & & \\ & & & 1 \\ & 1 & 1 & -1 \end{bmatrix}, \quad \mathbf{D}_6^{(2)} = \begin{bmatrix} \mathbf{A}_2 & & & \\ & \mathbf{B}_2 & & \\ & & c_5 & \\ & & & c_5 \end{bmatrix}, \quad \mathbf{W}_{7\times 6} = \begin{bmatrix} \mathbf{W}_4 & & \\ & 1 & \\ & & 1 \\ & 1 & \end{bmatrix},
$$

$$
\mathbf{W}_4 = \mathbf{H}_2 \otimes \mathbf{I}_2, \quad \mathbf{W}_{5\times 7} = \begin{bmatrix} 1 & & & & 1 & & \\ & 1 & & & & & \\ & & & & & 1 & \\ & & 1 & & & & 1 \\ & & & 1 & & & \end{bmatrix}, \quad \mathbf{P}_5^{(\pi_5^{(1)})} = \begin{bmatrix} 1 & & & & \\ & & & & 1 \\ & & 1 & & \\ & 1 & & & \\ & & & 1 & \end{bmatrix}.
$$

Next, matrices $\mathbf{A}_2$ and $\mathbf{B}_2$ have structures that correspond to matrix patterns that are beneficial to us. We will immediately derive the final expression for DST-I for $N = 5$:

$$
\mathbf{Y}_{5\times 1} = \mathbf{P}_5^{(\pi_5^{(1)})} \mathbf{W}_{5\times 7} \mathbf{W}_{7\times 6} \mathbf{D}_6^{(3)} \mathbf{W}_6^{(1)} \mathbf{W}_{6\times 7} \mathbf{W}_{7\times 5}^{(0)} \mathbf{P}_5^{(\pi_5^{(0)})} \mathbf{X}_{5\times 1} \tag{20}
$$

where

$$
\mathbf{W}_6^{(1)} = \begin{bmatrix} \mathbf{H}_2 & & \\ & \mathbf{H}_2 & \\ & & \mathbf{I}_2 \end{bmatrix}, \quad \mathbf{D}_6^{(3)} = \mathrm{diag}\left(s_0^{(5)}, s_1^{(5)}, \ldots, s_5^{(5)}\right),
$$

$$
s_0^{(5)} = a_5, \quad s_1^{(5)} = s_2^{(5)} = s_3^{(5)} = b_5, \quad s_4^{(5)} = s_5^{(5)} = c_5.
$$

The data flow graph for our solution for five-point DST-I is shown in Figure 4. The naive, direct computation requires 16 additions and 9 multiplications. As can be observed, our solution uses 12 additions and 3 multiplications, reducing the number of additions from 16 to 12 and the number of multiplications from 9 to 3.

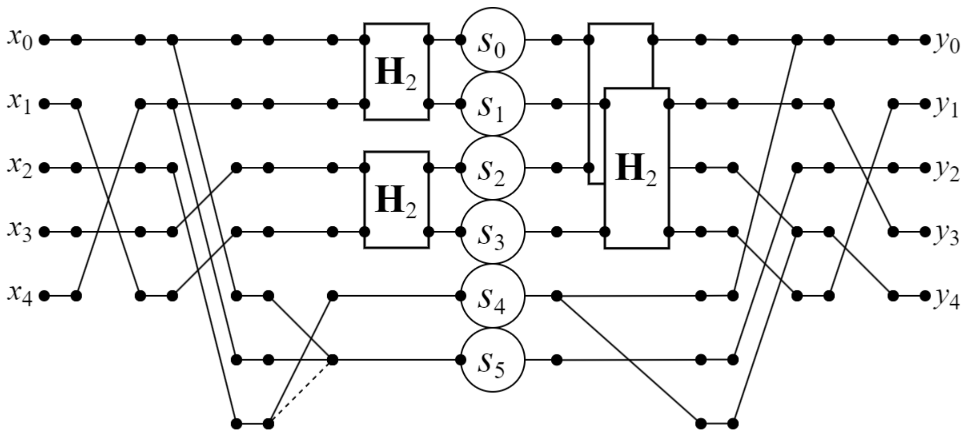

Figure 4. The proposed solution's data flow graph for five-point DST-I computation.

7. Six-Point DST-I Solution

The following is the matrix form expression for six-point DST-I:

$$
\mathbf{Y}_{6\times 1} = \mathbf{C}_6 \mathbf{X}_{6\times 1} \tag{21}
$$

where

$$\mathbf{Y}_{6\times1} = [y_0, y_1, y_2, y_3, y_4, y_5]^{\mathrm{T}}, \quad \mathbf{X}_{6\times1} = [x_0, x_1, x_2, x_3, x_4, x_5]^{\mathrm{T}},$$

$$\mathbf{C}_6 = \left[\begin{array}{c:c:c:c:c:c} a_6 & b_6 & c_6 & c_6 & b_6 & a_6 \\ \hdashline b_6 & c_6 & a_6 & -a_6 & -c_6 & -b_6 \\ \hdashline c_6 & a_6 & -b_6 & -b_6 & a_6 & c_6 \\ \hdashline c_6 & -a_6 & -b_6 & b_6 & a_6 & -c_6 \\ \hdashline b_6 & -c_6 & a_6 & a_6 & -c_6 & b_6 \\ \hdashline a_6 & -b_6 & c_6 & -c_6 & b_6 & -a_6 \end{array}\right], \quad \begin{array}{l} a_6 = 0.2319, \\ b_6 = 0.4179, \\ c_6 = 0.5211. \end{array}$$

Now, we swap the columns and rows in matrix $\mathbf{C}_6$ to match the matrix pattern:

$$\left[\begin{array}{c:c} \mathbf{A}_3 & \mathbf{A}_3 \\ \hdashline \mathbf{B}_3 & -\mathbf{B}_3 \end{array}\right] \quad \text{where} \quad \mathbf{A}_3 = \left[\begin{array}{c:c:c} a_6 & b_6 & c_6 \\ \hdashline b_6 & -c_6 & a_6 \\ \hdashline c_6 & a_6 & -b_6 \end{array}\right], \quad \mathbf{B}_3 = \left[\begin{array}{c:c:c} c_6 & -a_6 & -b_6 \\ \hdashline b_6 & c_6 & a_6 \\ \hdashline a_6 & -b_6 & c_6 \end{array}\right].$$

This is accomplished by the permutation $\pi_6^{(0)}$ for the columns and permutation $\pi_6^{(1)}$ for the rows:

$$\pi_6^{(0)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 6 & 5 & 4 \end{pmatrix}, \quad \pi_6^{(1)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 5 & 3 & 4 & 2 & 6 \end{pmatrix}. \tag{22}$$

Considering this, we may obtain the following expression for the first stage of decomposition:

$$\mathbf{Y}_{6\times1} = \mathbf{P}_6^{(\pi_6^{(1)})} \mathbf{D}_6^{(4)} \mathbf{W}_6^{(2)} \mathbf{P}_6^{(\pi_6^{(0)})} \mathbf{X}_{6\times1} \tag{23}$$

where

$$\mathbf{P}_6^{(\pi_6^{(0)})} = \left[\begin{array}{c:c:c:c} \mathbf{I}_3 & & & \\ \hdashline & & & 1 \\ \hdashline & & 1 & \\ \hdashline & 1 & & \end{array}\right], \quad \begin{array}{l} \mathbf{W}_6^{(2)} = \mathbf{H}_2 \otimes \mathbf{I}_3, \\ \\ \mathbf{D}_6^{(4)} = \mathbf{A}_3 \oplus \mathbf{B}_3, \end{array} \quad \mathbf{P}_6^{(\pi_6^{(1)})} = \left[\begin{array}{c:c:c:c:c:c} 1 & & & & & \\ \hdashline & & & & 1 & \\ \hdashline & & 1 & & & \\ \hdashline & & & 1 & & \\ \hdashline & 1 & & & & \\ \hdashline & & & & & 1 \end{array}\right].$$

Now, we focus on the matrices $\mathbf{A}_3$ and $\mathbf{B}_3$. We define the permutation $\pi_3^{(0)}$ as follows:

$$\pi_3^{(0)} = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}. \tag{24}$$

Now, we swap the rows in matrix $\mathbf{A}_3$ using the $\pi_3^{(0)}$ permutation. Furthermore, we will change the signs in the first column and first row of this matrix. After these operations, the matrix has the following form:

$$\mathbf{A}'_3 = \left[\begin{array}{c:c:c} a_6 & -c_6 & -b_6 \\ \hdashline -b_6 & a_6 & -c_6 \\ \hdashline -c_6 & -b_6 & a_6 \end{array}\right].$$

Next, we apply a circular convolution [44] to the $\mathbf{A}'_3$ matrix. Below are the expressions to calculate the circular convolution values for a matrix of size 3.

$$\left[\begin{array}{c:c:c} h_0 & h_2 & h_1 \\ \hdashline h_1 & h_0 & h_2 \\ \hdashline h_2 & h_1 & h_0 \end{array}\right], \quad \begin{array}{ll} s_0 = \frac{1}{3}(h_0 + h_1 + h_2), & s_2 = h_1 - h_2, \\ s_1 = h_0 - h_2, & s_3 = \frac{1}{3}(h_0 + h_1 - 2h_2). \end{array}$$

So, for $\mathbf{A}_3'$, we have

$$s_0^{(6)} = \frac{a_6 - c_6 - b_6}{3}, \quad s_1^{(6)} = a_6 + c_6, \quad s_2^{(6)} = -b_6 + c_6, \quad s_3^{(6)} = \frac{a_6 - b_6 + 2c_6}{3}. \tag{25}$$

The calculation procedure for matrix $\mathbf{A}_3$ is as follows:

$$\mathbf{A}_3 = \mathbf{P}_3^{(1)}\mathbf{T}_3^{(1)}\mathbf{A}_{3\times4}\mathbf{D}_4^{(0)}\mathbf{A}_{4\times3}\mathbf{T}_3^{(0)}\mathbf{P}_3^{(0)} \tag{26}$$

where

$$\mathbf{P}_3^{(0)} = \begin{bmatrix} -1 & & \\ & & 1 \\ & 1 & \end{bmatrix}, \quad \mathbf{T}_3^{(0)} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & & -1 \\ & 1 & -1 \end{bmatrix}, \quad \mathbf{A}_{4\times3} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \\ & 1 & 1 \end{bmatrix},$$

$$\mathbf{D}_4^{(0)} = \operatorname{diag}\left(s_0^{(6)}, s_1^{(6)}, s_2^{(6)}, s_3^{(6)}\right), \quad \mathbf{A}_{3\times4} = \begin{bmatrix} 1 & & & \\ & 1 & & -1 \\ & & 1 & -1 \end{bmatrix},$$

$$\mathbf{T}_3^{(1)} = \begin{bmatrix} 1 & 1 & \\ 1 & -1 & -1 \\ 1 & & 1 \end{bmatrix}, \quad \mathbf{P}_3^{(1)} = \begin{bmatrix} -1 & & \\ & 1 & \\ & & 1 \end{bmatrix}.$$

In the matrix $\mathbf{B}_3$, we will change the signs in the third column and third row. After these operations, the matrix has the following form:

$$\mathbf{B}_3' = \begin{bmatrix} c_6 & -a_6 & b_6 \\ b_6 & c_6 & -a_6 \\ -a_6 & b_6 & c_6 \end{bmatrix}.$$

And again, a circular convolution matrix will be used:

$$s_4^{(6)} = \frac{c_6 + b_6 - a_6}{3}, \quad s_5^{(6)} = c_6 + a_6, \quad s_6^{(6)} = b_6 + a_6, \quad s_7^{(6)} = \frac{c_6 + b_6 + 2a_6}{3}. \tag{27}$$

The calculation procedure for matrix $\mathbf{B}_3$ is as follows:

$$\mathbf{B}_3 = \mathbf{P}_3^{(2)}\mathbf{T}_3^{(1)}\mathbf{A}_{3\times4}\mathbf{D}_4^{(1)}\mathbf{A}_{4\times3}\mathbf{T}_3^{(0)}\mathbf{P}_3^{(2)} \tag{28}$$

where

$$\mathbf{P}_3^{(2)} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & -1 \end{bmatrix}, \quad \mathbf{D}_4^{(1)} = \operatorname{diag}\left(s_4^{(6)}, s_5^{(6)}, s_6^{(6)}, s_7^{(6)}\right).$$

In regard to this, we can determine the final expression for DST-I for $N = 6$:

$$\mathbf{Y}_{6\times1} = \mathbf{P}_6^{(\pi_6^{(1)})}\mathbf{P}_6^{(2)}\mathbf{A}_6^{(2)}\mathbf{A}_{6\times8}\mathbf{D}_8^{(0)}\mathbf{A}_{8\times6}\mathbf{A}_6^{(1)}\mathbf{P}_6^{(1)}\mathbf{W}_6^{(2)}\mathbf{P}_6^{(\pi_6^{(0)})}\mathbf{X}_{6\times1} \tag{29}$$

where

$$\mathbf{P}_6^{(1)} = \begin{bmatrix} \mathbf{P}_3^{(0)} & \\ & \mathbf{P}_3^{(2)} \end{bmatrix}, \quad \mathbf{A}_6^{(1)} = \begin{bmatrix} \mathbf{T}_3^{(0)} & \\ & \mathbf{T}_3^{(0)} \end{bmatrix}, \quad \mathbf{A}_{8\times6} = \begin{bmatrix} \mathbf{A}_{4\times3} & \\ & \mathbf{A}_{4\times3} \end{bmatrix},$$

$$\mathbf{D}_8^{(0)} = \begin{bmatrix} \mathbf{D}_4^{(0)} & \\ & \mathbf{D}_4^{(1)} \end{bmatrix}, \quad \mathbf{A}_{6\times8} = \begin{bmatrix} \mathbf{A}_{3\times4} & \\ & \mathbf{A}_{3\times4} \end{bmatrix}, \quad \mathbf{A}_6^{(2)} = \begin{bmatrix} \mathbf{T}_3^{(1)} & \\ & \mathbf{T}_3^{(1)} \end{bmatrix},$$

$$\mathbf{P}_6^{(2)} = \left[\begin{array}{c|c} \mathbf{P}_3^{(1)} & \\ \hline & \mathbf{P}_3^{(2)} \end{array}\right].$$

The data flow graph for our solution for six-point DST-I is shown in Figure 5. The naive, direct computation requires 30 additions and 36 multiplications. As can be observed, our solution uses 28 additions and 8 multiplications, reducing the number of additions from 30 to 28 and the number of multiplications from 36 to 8.

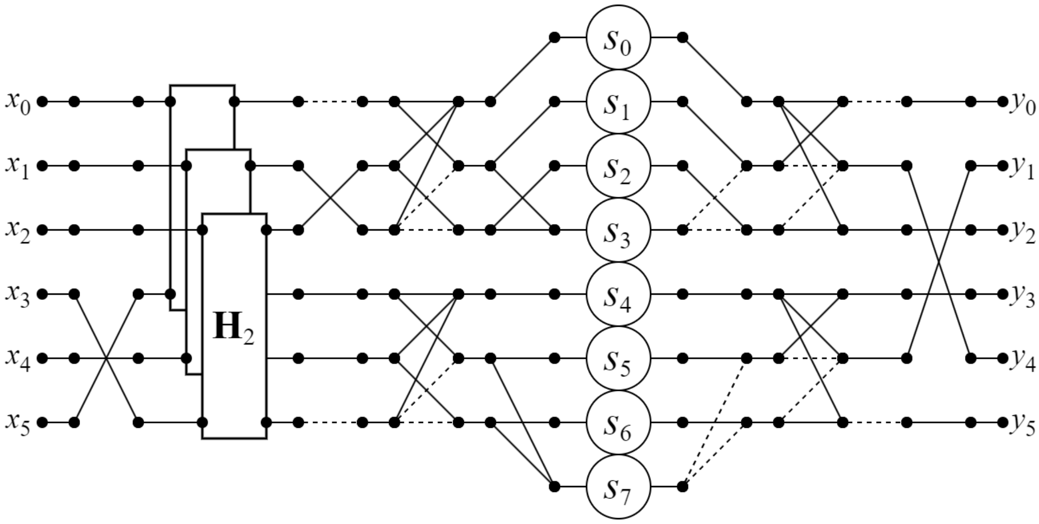

Figure 5. The proposed solution's data flow graph for six-point DST-I computation.

8. Seven-Point DST-I Solution

The following is the matrix form expression for seven-point DST-I:

$$\mathbf{Y}_{7\times1} = \mathbf{C}_7\mathbf{X}_{7\times1} \tag{30}$$

where

$$\mathbf{Y}_{7\times1} = [y_0, y_1, y_2, y_3, y_4, y_5, y_6]^{\mathrm{T}}, \quad \mathbf{X}_{7\times1} = [x_0, x_1, x_2, x_3, x_4, x_5, x_6]^{\mathrm{T}},$$

$$\mathbf{C}_7 = \begin{bmatrix} a_7 & b_7 & c_7 & d_7 & c_7 & b_7 & a_7 \\ b_7 & d_7 & b_7 & 0 & -b_7 & -d_7 & -b_7 \\ c_7 & b_7 & -a_7 & -d_7 & -a_7 & b_7 & c_7 \\ d_7 & 0 & -d_7 & 0 & d_7 & 0 & -d_7 \\ c_7 & -b_7 & -a_7 & d_7 & -a_7 & -b_7 & c_7 \\ b_7 & -d_7 & b_7 & 0 & -b_7 & d_7 & -b_7 \\ a_7 & -b_7 & c_7 & -d_7 & c_7 & -b_7 & a_7 \end{bmatrix}, \quad \begin{array}{l} a_7 = 0.1913, \\ b_7 = 0.3536, \\ c_7 = 0.4619, \\ d_7 = 0.5. \end{array}$$

Now, we swap the columns and rows in matrix $\mathbf{C}_7$. For this purpose, we define the permutation $\pi_7^{(0)}$ and $\pi_7^{(1)}$ as follows:

$$\pi_7^{(0)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 6 & 3 & 4 & 7 & 2 & 5 \end{pmatrix}, \quad \pi_7^{(1)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 7 & 6 & 5 \end{pmatrix}. \tag{31}$$

Permute columns of matrix $\mathbf{C}_7$ according to $\pi_7^{(0)}$ and rows according to $\pi_7^{(1)}$. Moreover, we need to change the sign in row 6 and sign in column 2. These operations are described as follows:

$$\mathbf{C}_7' = \mathbf{P}_7^{(\pi_7^{(1)})}\mathbf{P}_7^{(1)}\mathbf{C}_7\left(\mathbf{P}_7^{(\pi_7^{(0)})}\mathbf{P}_7^{(0)}\right)^{\mathrm{T}} \tag{32}$$

where

$$\mathbf{P}_7^{(0)} = \begin{bmatrix} 1 & & & & & & \\ & -1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \end{bmatrix}, \quad \mathbf{P}_7^{(\pi_7^{(0)})} = \begin{bmatrix} 1 & & & & & & \\ & & & & & 1 & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & & & 1 \\ & 1 & & & & & \\ & & & & 1 & & \end{bmatrix},$$

$$\mathbf{P}_7^{(1)} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & -1 & \\ & & & & & & 1 \end{bmatrix}, \quad \mathbf{P}_7^{(\pi_7^{(1)})} = \begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & & & 1 \\ & & & & & 1 & \\ & & & & 1 & & \end{bmatrix},$$

The matrix $\mathbf{C}_7'$ is as takes after:

$$\mathbf{C}_7' = \begin{bmatrix} a_7 & b_7 & c_7 & d_7 & a_7 & -b_7 & c_7 \\ b_7 & -d_7 & b_7 & & -b_7 & -d_7 & -b_7 \\ c_7 & b_7 & -a_7 & -d_7 & c_7 & -b_7 & -a_7 \\ d_7 & & -d_7 & & -d_7 & & d_7 \\ a_7 & -b_7 & c_7 & -d_7 & a_7 & b_7 & c_7 \\ -b_7 & -d_7 & -b_7 & & b_7 & -d_7 & b_7 \\ c_7 & -b_7 & -a_7 & d_7 & c_7 & b_7 & -a_7 \end{bmatrix}.$$

Now we will divide the matrix $\mathbf{C}_7'$ into two parts:

$$\mathbf{C}_7' = \mathbf{C}_7^{(\mathrm{a})} + \mathbf{C}_7^{(\mathrm{b})} \tag{33}$$

where

$$\mathbf{C}_7^{(\mathrm{a})}=\begin{bmatrix} a_7 & b_7 & c_7 & & a_7 & -b_7 & c_7 \\ b_7 & -d_7 & b_7 & & -b_7 & -d_7 & -b_7 \\ c_7 & b_7 & -a_7 & & c_7 & -b_7 & -a_7 \\ & & & & & & \\ a_7 & -b_7 & c_7 & & a_7 & b_7 & c_7 \\ -b_7 & -d_7 & -b_7 & & b_7 & -d_7 & b_7 \\ c_7 & -b_7 & -a_7 & & c_7 & b_7 & -a_7 \end{bmatrix},$$

$$\mathbf{C}_7^{(\mathrm{b})}=\begin{bmatrix} & & & d_7 & & & \\ & & & & & & \\ & & & -d_7 & & & \\ d_7 & & -d_7 & & -d_7 & & d_7 \\ & & & -d_7 & & & \\ & & & & & & \\ & & & d_7 & & & \end{bmatrix}.$$

The matrix $\mathbf{C}_7^{(\mathrm{b})}$ has one element in the first, third, fifth, and seventh rows and four elements with the same value in the fourth row, which allows us to reduce the number of operations without the need for further transformations. The matrix $\mathbf{C}_7^{(\mathrm{a})}$ after removing terms equal to zero is as follows:

$$\mathbf{C}_6=\begin{bmatrix} a_7 & b_7 & c_7 & a_7 & -b_7 & c_7 \\ b_7 & -d_7 & b_7 & -b_7 & -d_7 & -b_7 \\ c_7 & b_7 & -a_7 & c_7 & -b_7 & -a_7 \\ a_7 & -b_7 & c_7 & a_7 & b_7 & c_7 \\ -b_7 & -d_7 & -b_7 & b_7 & -d_7 & b_7 \\ c_7 & -b_7 & -a_7 & c_7 & b_7 & -a_7 \end{bmatrix}.$$

Now, we can see that the matrix $\mathbf{C}_6$ matches the following matrix pattern:

$$\begin{bmatrix} \mathbf{A}_3 & \mathbf{B}_3 \\ \mathbf{B}_3 & \mathbf{A}_3 \end{bmatrix}$$

where

$$\mathbf{A}_3=\begin{bmatrix} a_7 & b_7 & c_7 \\ b_7 & -d_7 & b_7 \\ c_7 & b_7 & -a_7 \end{bmatrix},\quad \mathbf{B}_3=\begin{bmatrix} a_7 & -b_7 & c_7 \\ -b_7 & -d_7 & -b_7 \\ c_7 & -b_7 & -a_7 \end{bmatrix}.$$

Considering this, we may obtain the following expression for the first stage of decomposition:

$$\mathbf{Y}_{7\times1}=\mathbf{P}_7^{(\pi_7^{(1)})}\mathbf{P}_7^{(1)}\mathbf{W}_{7\times11}\mathbf{W}_{11\times8}\mathbf{D}_8^{(1)}\mathbf{W}_{8\times11}\mathbf{W}_{11\times7}\mathbf{P}_7^{(\pi_7^{(0)})}\mathbf{P}_7^{(0)}\mathbf{X}_{7\times1} \tag{34}$$

where

$$\mathbf{W}_{11\times7}=\begin{bmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ & & & & & & 1 \\ 1 & & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & & 1 \end{bmatrix},\quad \mathbf{W}_{8\times11}=\begin{bmatrix} \mathbf{H}_2\otimes\mathbf{I}_3 & & & & & \\ & & & 1 & & \\ & 1 & -1 & & -1 & 1 \end{bmatrix},$$

$$\mathbf{D}_8^{(1)}=\begin{bmatrix} \mathbf{F}_3 & & & \\ & \mathbf{G}_3 & & \\ & & d_7 & \\ & & & d_7 \end{bmatrix},$$

$$\mathbf{F}_3=(\mathbf{A}_3+\mathbf{B}_3)/2,\quad \mathbf{G}_3=(\mathbf{A}_3-\mathbf{B}_3)/2,$$

$$\mathbf{W}_{11\times 8} = \begin{bmatrix} \mathbf{H}_2 \otimes \mathbf{I}_3 & & \\ & 1 & \\ & -1 & \\ & & 1 \\ & -1 & \\ & 1 & \end{bmatrix}, \quad \mathbf{W}_{7\times 11} = \begin{bmatrix} 1 & & & & & & 1 & & & & \\ & 1 & & & & & & & & & \\ & & 1 & & & & & 1 & & & \\ & & & & & & & & 1 & & \\ & & & 1 & & & & & & 1 & \\ & & & & 1 & & & & & & \\ & & & & & 1 & & & & & 1 \end{bmatrix}.$$

The matrices $\mathbf{F}_3$ and $\mathbf{G}_3$ are as takes after:

$$\mathbf{F}_3 = \frac{\begin{bmatrix} a_7 & b_7 & c_7 \\ b_7 & -d_7 & b_7 \\ c_7 & b_7 & -a_7 \end{bmatrix} + \begin{bmatrix} a_7 & -b_7 & c_7 \\ -b_7 & -d_7 & -b_7 \\ c_7 & -b_7 & -a_7 \end{bmatrix}}{2} = \begin{bmatrix} a_7 & & c_7 \\ & -d_7 & \\ c_7 & & -a_7 \end{bmatrix}, \tag{35}$$

$$\mathbf{G}_3 = \frac{\begin{bmatrix} a_7 & b_7 & c_7 \\ b_7 & -d_7 & b_7 \\ c_7 & b_7 & -a_7 \end{bmatrix} - \begin{bmatrix} a_7 & -b_7 & c_7 \\ -b_7 & -d_7 & -b_7 \\ c_7 & -b_7 & -a_7 \end{bmatrix}}{2} = \begin{bmatrix} & b_7 & \\ b_7 & & b_7 \\ & b_7 & \end{bmatrix}. \tag{36}$$

Now, we will divide the matrix $\mathbf{F}_3$ into two parts:

$$\mathbf{F}_3 = \mathbf{F}_3^{(a)} + \mathbf{F}_3^{(b)} \tag{37}$$

where

$$\mathbf{F}_3^{(a)} = \begin{bmatrix} a_7 & & c_7 \\ & & \\ c_7 & & -a_7 \end{bmatrix}, \quad \mathbf{F}_3^{(b)} = \begin{bmatrix} & & \\ & -d_7 & \\ & & \end{bmatrix}.$$

The matrix $\mathbf{F}_3^{(a)}$ after removing terms equal to zero is as follows:

$$\mathbf{F}_2 = \begin{bmatrix} a_7 & c_7 \\ c_7 & -a_7 \end{bmatrix} \text{ and matches the matrix pattern } \begin{bmatrix} a & b \\ b & -a \end{bmatrix}.$$

The calculation procedure for matrix $\mathbf{F}_3$ is as follows:

$$\mathbf{F}_3 = \mathbf{W}_{3\times 4}\mathbf{D}_4^{(2)}\mathbf{W}_{4\times 3} \tag{38}$$

where

$$\mathbf{W}_{4\times 3} = \begin{bmatrix} 1 & & \\ & & 1 \\ 1 & & 1 \\ & 1 & \end{bmatrix}, \quad \mathbf{D}_4^{(2)} = \operatorname{diag}\left(s_0^{(7)}, s_1^{(7)}, s_2^{(7)}, s_3^{(7)}\right), \quad s_0^{(7)} = a_7 - c_7,$$

$$s_1^{(7)} = -a_7 - c_7, \quad s_2^{(7)} = c_7, \quad s_3^{(7)} = -d_7, \quad \mathbf{W}_{3\times 4} = \begin{bmatrix} 1 & & 1 & \\ & & & 1 \\ & 1 & 1 & \end{bmatrix}.$$

The calculation procedure for matrix $\mathbf{G}_3$ is as follows:

$$\mathbf{G}_3 = \mathbf{W}_{3\times 2}\mathbf{D}_2^{(1)}\mathbf{W}_{2\times 3} \tag{39}$$

where

$$\mathbf{W}_{2\times3}=\left[\begin{array}{c:c:c}1&&1\\ \hdashline &1&\end{array}\right],\quad \mathbf{D}_{2}^{(1)}=\mathrm{diag}\left(s_4^{(7)},s_5^{(7)}\right),\quad s_4^{(7)}=s_5^{(7)}=b_7,\quad \mathbf{W}_{3\times2}=\left[\begin{array}{c:c}&1\\ \hdashline 1&\\ \hdashline &1\end{array}\right].$$

In regard to this, we can determine the final expression for DST-I for N = 7:

$$\mathbf{Y}_{7\times1}=\mathbf{P}_7^{(\pi_7^{(1)})}\mathbf{P}_7^{(1)}\mathbf{W}_{7\times11}\mathbf{W}_{11\times8}\mathbf{W}_8^{(1)}\mathbf{D}_8^{(2)}\mathbf{W}_8^{(0)}\mathbf{W}_{8\times11}\mathbf{W}_{11\times7}\mathbf{P}_7^{(\pi_7^{(0)})}\mathbf{P}_7^{(0)}\mathbf{X}_{7\times1} \tag{40}$$

where

$$\mathbf{W}_8^{(0)}=\left[\begin{array}{c:c:c}\mathbf{W}_{4\times3}&&\\ \hdashline &\mathbf{W}_{2\times3}&\\ \hdashline &&\mathbf{I}_2\end{array}\right],\quad \mathbf{D}_8^{(2)}=\left[\begin{array}{c:c:c}\mathbf{D}_4^{(2)}&&\\ \hdashline &\mathbf{D}_2^{(1)}&\\ \hdashline &&\mathbf{D}_2^{(2)}\end{array}\right],$$

$$\mathbf{D}_2^{(2)}=\mathrm{diag}\left(s_6^{(7)},s_7^{(7)}\right),\quad s_6^{(7)}=s_7^{(7)}=d_7,\quad \mathbf{W}_8^{(1)}=\left[\begin{array}{c:c:c}\mathbf{W}_{3\times4}&&\\ \hdashline &\mathbf{W}_{3\times2}&\\ \hdashline &&\mathbf{I}_2\end{array}\right].$$

The data flow graph for our solution for seven-point DST-I is shown in Figure 6. The naive, direct computation requires 37 additions and 32 multiplications. As can be observed, our solution uses 23 additions and 5 multiplications, reducing the number of additions from 37 to 23 and the number of multiplications from 32 to 5.

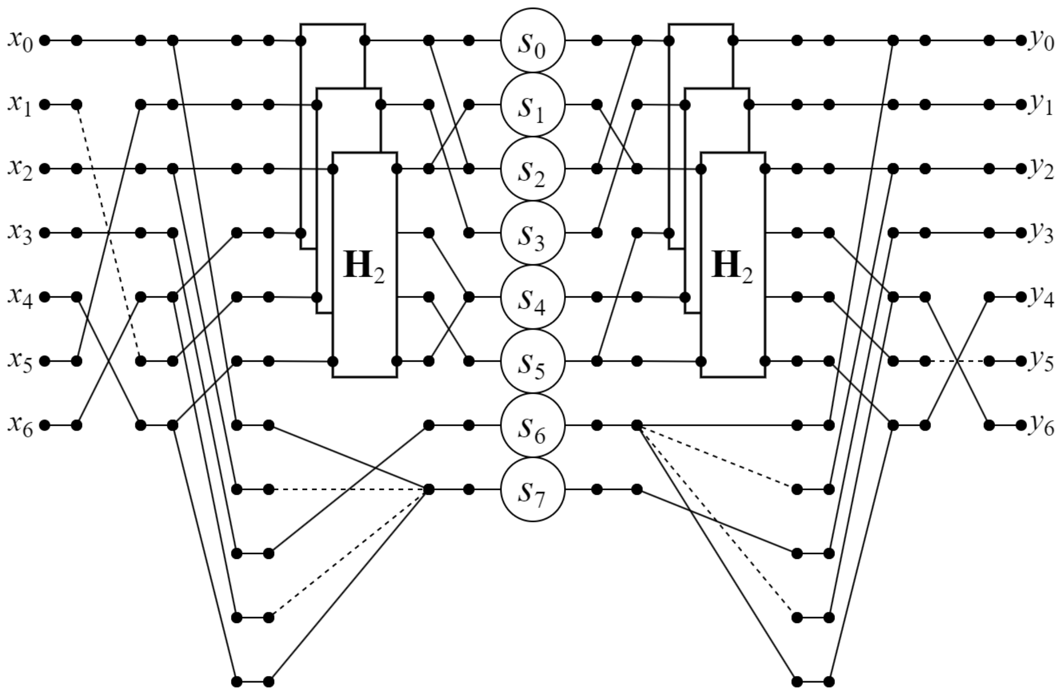

Figure 6. The proposed solution's data flow graph for seven-point DST-I computation.

9. Eight-Point DST-I Solution

The following is the matrix form expression for eight-point DST-I:

$$\mathbf{Y}_{8\times1}=\mathbf{C}_8\mathbf{X}_{8\times1} \tag{41}$$

where

$$\mathbf{Y}_{8\times 1} = [y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7]^{\mathrm{T}}, \quad \mathbf{X}_{8\times 1} = [x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7]^{\mathrm{T}},$$

$$\mathbf{C}_8 = \begin{bmatrix} a_8 & b_8 & c_8 & d_8 & d_8 & c_8 & b_8 & a_8 \\ b_8 & d_8 & c_8 & a_8 & -a_8 & -c_8 & -d_8 & -b_8 \\ c_8 & c_8 & 0 & -c_8 & -c_8 & 0 & c_8 & c_8 \\ d_8 & a_8 & -c_8 & -b_8 & b_8 & c_8 & -a_8 & -d_8 \\ d_8 & -a_8 & -c_8 & b_8 & b_8 & -c_8 & -a_8 & d_8 \\ c_8 & -c_8 & 0 & c_8 & -c_8 & 0 & c_8 & -c_8 \\ b_8 & -d_8 & c_8 & -a_8 & -a_8 & c_8 & -d_8 & b_8 \\ a_8 & -b_8 & c_8 & -d_8 & d_8 & -c_8 & b_8 & -a_8 \end{bmatrix}, \quad \begin{aligned} a_8 &= 0.1612, \\ b_8 &= 0.3030, \\ c_8 &= 0.4082, \\ d_8 &= 0.4642. \end{aligned}$$

Now, we swap the columns and rows in matrix $\mathbf{C}_8$ to match the following matrix pattern:

$$\begin{bmatrix} \mathbf{A}_4 & \mathbf{A}_4 \\ \mathbf{B}_4 & -\mathbf{B}_4 \end{bmatrix}$$

where

$$\mathbf{A}_4 = \begin{bmatrix} a_8 & b_8 & c_8 & d_8 \\ b_8 & -d_8 & c_8 & -a_8 \\ c_8 & c_8 & 0 & -c_8 \\ d_8 & -a_8 & -c_8 & b_8 \end{bmatrix}, \quad \mathbf{B}_4 = \begin{bmatrix} d_8 & a_8 & -c_8 & -b_8 \\ c_8 & -c_8 & 0 & c_8 \\ b_8 & d_8 & c_8 & a_8 \\ a_8 & -b_8 & c_8 & -d_8 \end{bmatrix}.$$

This is accomplished by the permutation $\pi_8^{(0)}$ for the columns and permutation $\pi_8^{(1)}$ for the rows:

$$\pi_8^{(0)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 8 & 7 & 6 & 5 \end{pmatrix}, \quad \pi_8^{(1)} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 7 & 3 & 5 & 4 & 6 & 2 & 8 \end{pmatrix}. \tag{42}$$

Considering this, we may obtain the following expression for the first stage of decomposition:

$$\mathbf{Y}_{8\times 1} = \mathbf{P}_8^{(\pi_8^{(1)})} \mathbf{D}_8^{(3)} \mathbf{W}_8^{(2)} \mathbf{P}_8^{(\pi_8^{(0)})} \mathbf{X}_{8\times 1} \tag{43}$$

where

$$\mathbf{P}_8^{(\pi_8^{(0)})} = \begin{bmatrix} \mathbf{I}_4 & & & & \\ & & & & 1 \\ & & & 1 & \\ & & 1 & & \\ & 1 & & & \end{bmatrix}, \quad \begin{aligned} \mathbf{W}_8^{(2)} &= \mathbf{H}_2 \otimes \mathbf{I}_4, \\ \mathbf{D}_8^{(3)} &= \mathbf{A}_4 \oplus \mathbf{B}_4, \end{aligned}$$

$$\mathbf{P}_8^{(\pi_8^{(1)})} = \begin{bmatrix} 1 & & & & & & & \\ & & & & & & 1 & \\ & & 1 & & & & & \\ & & & & 1 & & & \\ & & & 1 & & & & \\ & & & & & 1 & & \\ & 1 & & & & & & \\ & & & & & & & 1 \end{bmatrix}.$$

Now, we will deal with matrices $\mathbf{A}_4$ and $\mathbf{B}_4$. Permute rows of $\mathbf{A}_4$ according to $\pi_4^{(0)}$, change the sign in row 1 and 4 and change the sign in column 4. These operations are described in the expression below:

$$\mathbf{A}_4' = \mathbf{P}_4^{(1)} \mathbf{P}_4^{(\pi_4^{(0)})} \mathbf{A}_4 \mathbf{P}_4^{(0)} \tag{44}$$

where

$$\mathbf{P}_4^{(0)} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{bmatrix}, \quad \mathbf{P}_4^{(\pi_4^{(0)})} = \begin{bmatrix} 1 & & & \\ & & & 1 \\ & & 1 & \\ & 1 & & \end{bmatrix}, \quad \mathbf{P}_4^{(1)} = \begin{bmatrix} -1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{bmatrix}.$$

As a result of the above Equation (44), matrix $\mathbf{A}_4'$ looks like the following:

$$\mathbf{A}_4' = \begin{bmatrix} -a_8 & -b_8 & -c_8 & d_8 \\ d_8 & -a_8 & -c_8 & -b_8 \\ c_8 & c_8 & 0 & c_8 \\ -b_8 & d_8 & -c_8 & -a_8 \end{bmatrix}.$$

Now, we will divide the matrix $\mathbf{A}_4'$ into two parts:

$$\mathbf{A}_4' = \mathbf{A}_4^{(\mathrm{a})} + \mathbf{A}_4^{(\mathrm{b})} \tag{45}$$

where

$$\mathbf{A}_4^{(\mathrm{a})} = \begin{bmatrix} -a_8 & -b_8 & & d_8 \\ d_8 & -a_8 & & -b_8 \\ & & & \\ -b_8 & d_8 & & -a_8 \end{bmatrix}, \quad \mathbf{A}_4^{(\mathrm{b})} = \begin{bmatrix} & & -c_8 & \\ & & -c_8 & \\ c_8 & c_8 & & c_8 \\ & & -c_8 & \end{bmatrix}.$$

The matrix $\mathbf{A}_4^{(\mathrm{a})}$ after removing terms equal to zero is as follows:

$\mathbf{A}_3 = \begin{bmatrix} -a_8 & -b_8 & d_8 \\ d_8 & -a_8 & -b_8 \\ -b_8 & d_8 & -a_8 \end{bmatrix}$ and takes the form of a circular convolution matrix.

So, we can again use the properties of a circular convolution matrix:

$$s_0^{(8)} = \frac{-a_8 + d_8 - b_8}{3}, \quad s_1^{(8)} = -a_8 + b_8, \quad s_2^{(8)} = d_8 + b_8, \quad s_3^{(8)} = \frac{-a_8 + d_8 + 2b_8}{3}. \tag{46}$$

The calculation procedure for matrix $\mathbf{A}_4$ is as follows:

$$\mathbf{A}_4 = \left(\mathbf{P}_4^{(1)} \mathbf{P}_4^{(\pi_4^{(0)})} \right)^{\mathrm{T}} \mathbf{W}_{4\times7}^{(0)} \mathbf{W}_{7\times5}^{(1)} \mathbf{W}_{5\times6} \mathbf{D}_6^{(5)} \mathbf{W}_{6\times5} \mathbf{W}_{5\times7}^{(1)} \mathbf{W}_7 \mathbf{P}_4^{(0)} \tag{47}$$

where

$$\mathbf{W}_7 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & & 1 \\ 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}, \quad \mathbf{W}_{5\times7}^{(1)} = \begin{bmatrix} \mathbf{T}_3^{(0)} & & & & \\ & & & 1 & \\ & 1 & 1 & & 1 \end{bmatrix}, \quad \mathbf{W}_{6\times5} = \begin{bmatrix} \mathbf{A}_{4\times3} & \\ & \mathbf{I}_2 \end{bmatrix},$$

$$\mathbf{D}_6^{(5)} = \mathrm{diag}\left(s_0^{(8)}, s_1^{(8)}, \ldots, s_5^{(8)}\right), \quad \mathbf{W}_{5\times6} = \begin{bmatrix} \mathbf{A}_{3\times4} & \\ & \mathbf{I}_2 \end{bmatrix}, \quad \mathbf{W}_{7\times5}^{(1)} = \begin{bmatrix} \mathbf{T}_3^{(1)} & & \\ & -1 & \\ & -1 & \\ & & 1 \\ & -1 & \end{bmatrix},$$

$$s_4^{(8)} = s_5^{(8)} = c_8,$$

$$\mathbf{W}_{4\times7}^{(0)} = \begin{bmatrix} 1 & & & 1 & & & \\ & 1 & & & 1 & & \\ & & & & & 1 & \\ & & 1 & & & & 1 \end{bmatrix}.$$

In matrix $\mathbf{B}_4$, change the sign in row 1 and change the sign in column 2. These operations are described in the expression below:

$$\mathbf{B}_4' = \mathbf{P}_4^{(3)}\mathbf{B}_4\mathbf{P}_4^{(2)} \tag{48}$$

where

$$\mathbf{P}_4^{(2)} = \begin{bmatrix} 1 & & & \\ & -1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}, \quad \mathbf{P}_4^{(3)} = \begin{bmatrix} -1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}.$$

As a result of the above Equation (48), matrix $\mathbf{B}_4'$ looks like this:

$$\mathbf{B}_4' = \begin{bmatrix} -d_8 & a_8 & c_8 & b_8 \\ c_8 & c_8 & 0 & c_8 \\ b_8 & -d_8 & c_8 & a_8 \\ a_8 & b_8 & c_8 & -d_8 \end{bmatrix}.$$

Now, we will divide the matrix $\mathbf{B}_4'$ into two parts:

$$\mathbf{B}_4' = \mathbf{B}_4^{(a)} + \mathbf{B}_4^{(b)} \tag{49}$$

where

$$\mathbf{B}_4^{(a)} = \begin{bmatrix} -d_8 & a_8 & & b_8 \\ & & & \\ b_8 & -d_8 & & a_8 \\ a_8 & b_8 & & -d_8 \end{bmatrix}, \quad \mathbf{B}_4^{(b)} = \begin{bmatrix} & & c_8 & \\ c_8 & c_8 & & c_8 \\ & & c_8 & \\ & & c_8 & \end{bmatrix}.$$

The matrix $\mathbf{B}_4^{(a)}$ after removing terms equal to zero is as follows:

$$\mathbf{B}_3 = \begin{bmatrix} -d_8 & a_8 & b_8 \\ b_8 & -d_8 & a_8 \\ a_8 & b_8 & -d_8 \end{bmatrix}$$ and takes the form of a circular convolution matrix.

So, we can again use the properties of a circular convolution matrix:

$$s_6^{(8)} = \frac{-d_8 + b_8 + a_8}{3}, \quad s_7^{(8)} = -d_8 - a_8, \quad s_8^{(8)} = b_8 - a_8, \quad s_9^{(8)} = \frac{-d_8 + b_8 - 2a_8}{3}. \tag{50}$$

The calculation procedure for matrix $\mathbf{B}_4$ is as follows:

$$\mathbf{B}_4 = \mathbf{P}_4^{(3)}\mathbf{W}_{4\times7}^{(1)}\mathbf{W}_{7\times5}^{(2)}\mathbf{W}_{5\times6}\mathbf{D}_6^{(6)}\mathbf{W}_{6\times5}\mathbf{W}_{5\times7}^{(1)}\mathbf{W}_7\mathbf{P}_4^{(2)} \tag{51}$$

where

$$\mathbf{D}_6^{(6)} = \operatorname{diag}\left(s_6^{(8)}, s_7^{(8)}, \ldots, s_{11}^{(8)}\right), \quad s_{10}^{(8)} = s_{11}^{(8)} = c_8, \qquad \mathbf{W}_{7\times 5}^{(2)} = \left[\begin{array}{c:c:c} \mathbf{T}_3^{(1)} & & \\ \hdashline & 1 & \\ \hdashline & & 1 \\ \hdashline & 1 & \\ \hdashline & 1 & \end{array}\right],$$

$$\mathbf{W}_{4\times 7}^{(1)} = \left[\begin{array}{c:c:c:c:c:c:c} 1 & & & 1 & & & \\ \hdashline & & & & 1 & & \\ \hdashline & 1 & & & & 1 & \\ \hdashline & & 1 & & & & 1 \end{array}\right].$$

Taking all the transformations together, we obtain the following expression for the fast DST-I algorithm for $N = 8$:

$$\mathbf{Y}_{8\times 1} = \mathbf{P}_8^{(\pi_8^{(1)})} \mathbf{P}_8^{(1)} \mathbf{W}_{8\times 14} \mathbf{W}_{14\times 10} \mathbf{W}_{10\times 12} \mathbf{D}_{12} \mathbf{W}_{12\times 10} \mathbf{W}_{10\times 14} \mathbf{W}_{14\times 8} \mathbf{P}_8^{(0)} \mathbf{W}_8^{(2)} \mathbf{P}_8^{(\pi_8^{(0)})} \mathbf{X}_{8\times 1} \quad (52)$$

where

$$\mathbf{P}_8^{(0)} = \left[\begin{array}{c:c} \mathbf{P}_4^{(0)} & \\ \hdashline & \mathbf{P}_4^{(2)} \end{array}\right], \quad \mathbf{W}_{14\times 8} = \left[\begin{array}{c:c} \mathbf{W}_{7\times 4} & \\ \hdashline & \mathbf{W}_{7\times 4} \end{array}\right], \quad \mathbf{W}_{10\times 14} = \left[\begin{array}{c:c} \mathbf{W}_{5\times 7}^{(1)} & \\ \hdashline & \mathbf{W}_{5\times 7}^{(1)} \end{array}\right],$$

$$\mathbf{W}_{12\times 10} = \left[\begin{array}{c:c} \mathbf{W}_{6\times 5} & \\ \hdashline & \mathbf{W}_{6\times 5} \end{array}\right], \quad \mathbf{D}_{12} = \left[\begin{array}{c:c} \mathbf{D}_6^{(5)} & \\ \hdashline & \mathbf{D}_6^{(6)} \end{array}\right], \quad \mathbf{W}_{10\times 12} = \left[\begin{array}{c:c} \mathbf{W}_{5\times 6} & \\ \hdashline & \mathbf{W}_{5\times 6} \end{array}\right],$$

$$\mathbf{W}_{14\times 10} = \left[\begin{array}{c:c} \mathbf{W}_{7\times 5}^{(1)} & \\ \hdashline & \mathbf{W}_{7\times 5}^{(2)} \end{array}\right], \quad \mathbf{W}_{8\times 14} = \left[\begin{array}{c:c} \mathbf{W}_{4\times 7}^{(0)} & \\ \hdashline & \mathbf{W}_{4\times 7}^{(1)} \end{array}\right],$$

$$\mathbf{P}_8^{(1)} = \left[\begin{array}{c:c} \left(\mathbf{P}_4^{(1)} \mathbf{P}_4^{(\pi_4^{(0)})}\right)^{\mathrm{T}} & \\ \hdashline & \mathbf{P}_4^{(3)} \end{array}\right].$$

The data flow graph for our solution for eight-point DST-I is shown in Figure 7. The naive, direct computation requires 52 additions and 60 multiplications. As can be observed, our solution uses 40 additions and 12 multiplications, reducing the number of additions from 52 to 40 and the number of multiplications from 60 to 12.

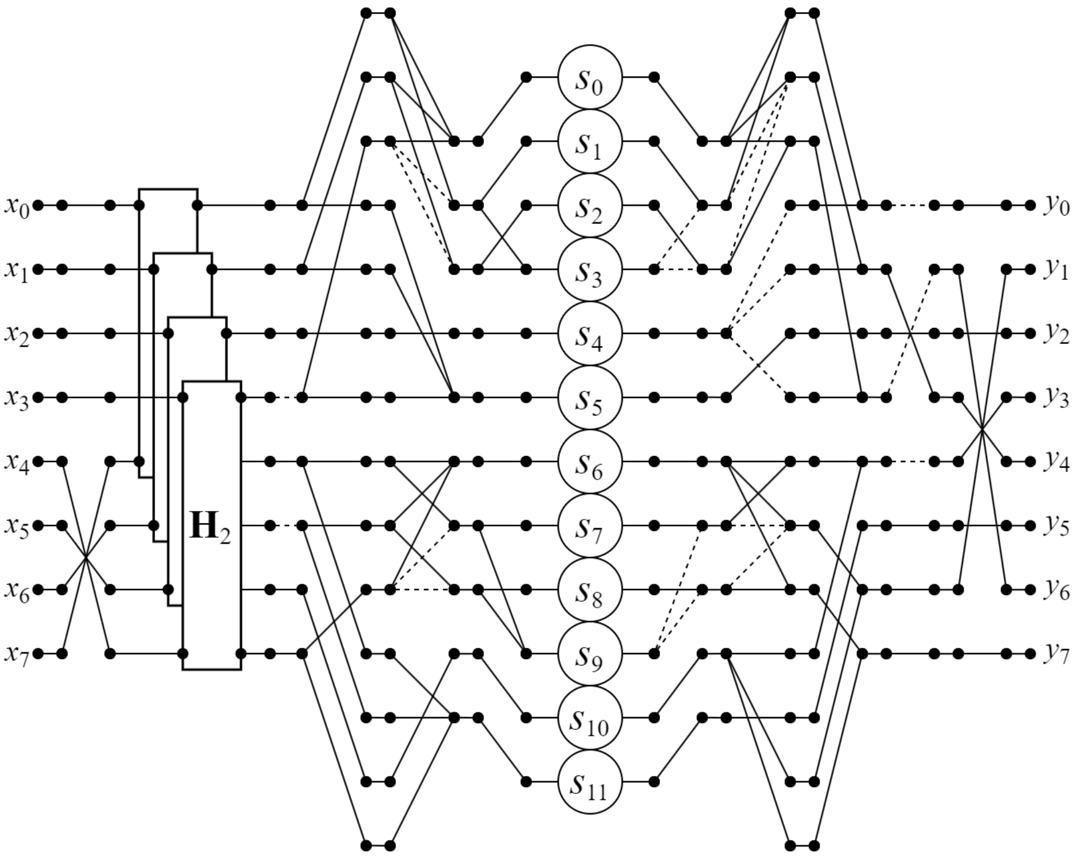

Figure 7. The proposed solution's data flow graph for eight-point DST-I computation.

10. Results

The work shows how it is possible to reduce the number of multiplication operations in DST-I algorithms of sizes two to eight. At the same time, the number of addition operations was slightly reduced. The number of addition operations was reduced by an average of 21% and the number of multiplication operations was reduced by an average of 74%. The achieved results are presented in the Table 1.

This allows for a significant reduction in the amount of resources used on a signal processor, while speeding up work and allowing for easier operation in real time. A significant reduction in multiplication operations contributes to this, because, due to their characteristics, they are more expensive to use than addition operations.

Table 1. Comparison of the direct method with the proposed solutions.

	Direct Method		Proposed Solutions	
N	**Additions**	**Multiplications**	**Additions**	**Multiplications**
2	2	4	2	2
3	5	4	4	2
4	12	16	12	6
5	16	9	12	3
6	30	36	28	8
7	37	32	23	5
8	52	60	40	12

Each proposed algorithm has been implemented in the MATLAB environment and we are sure that they all work correctly. We have published the program code in an open dataset repository, which we reference in the Data Availability Statement section.

11. Discussion of Computational Complexity

For the direct DST-I calculation approach and suggested solutions, we first describe how to determine the number of multiplication and addition operations. A bit shift can be used in place of a multiplication operation for any number that is a power of two. We do not count addition and multiplication operations for a value of zero.

The above appear in the following matrices: $\mathbf{C}_3$—one 0 and four values of 0.5; $\mathbf{C}_5$—four 0s and twelve values of 0.5; $\mathbf{C}_7$—five 0s and twelve values of 0.5; $\mathbf{C}_8$—four 0s. And in the proposed solutions in diagonal matrices, we have the following: $\mathbf{D}_3$—one value of 0.5; $\mathbf{D}_6^{(3)}$—three values of 0.5; $\mathbf{D}_8^{(2)}$—three values of 0.5; $\mathbf{D}_8$—one value 0.5.

Additionally, Table 2 provides a comparison with the results obtained by other researchers of the topic we addressed. Yip and Rao used the sparse-matrix factorization technique and Sun and Yip used the idea of split radix algorithm. However, these works do not present the exact step-by-step achievement of the results, as we show for each solution. Our solutions are transparent.

In this regard, we note that both works in the table below do not include normalizing coefficients in the number of multiplication operations. To the Yip and Rao solution for $N = 4$, we have added four multiplication operations, which correspond to multiplications by normalizing coefficients. Similarly, for the DST algorithms for $N = 8$, we have added eight multiplication operations in both cases.

The rigorous mathematical derivation of the final computational procedures for each case is presented in full. To ensure the correctness of these procedures, we have written validation computer programs, which we have included in our paper. We do not claim that the presented solutions are optimal. We show what we have obtained so far and would be glad if someone publishes better solutions.

Table 2. Comparison of the proposed solutions with other algorithms.

	$N = 4$		$N = 8$	
	Additions	**Multiplications**	**Additions**	**Multiplications**
Yip and Rao [40]	4	2 + 4	22	8 + 8
Sun and Yip [45]	-	-	18	6 + 8
Our solutions	12	6	40	12

Author Contributions: Conceptualization, A.C.; methodology, A.C., J.B. and M.R.; software, M.R.; validation, M.R.; formal analysis, A.C. and J.B.; investigation, J.B. and M.R.; writing—original draft preparation, M.R. and A.C.; writing—review and editing, M.R.; supervision, A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The MATLAB programming code implementing the developed algorithms for DST-I is available at [46] (accessed on 28 November 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Britanak, V.; Yip, P.C.; Rao, K.R. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*; Academic: Amsterdam, The Netherlands; Boston, MA, USA, 2007.
2. Yip, P.; Rao, K. On the computation and the effectiveness of discrete sine transform. *Comput. Electr. Eng.* **1980**, *7*, 45–55. [CrossRef]
3. Jain, A.K. A Sinusoidal Family of Unitary Transforms. *IEEE Trans. Pattern Anal. Mach. Intell.* **1979**, *PAMI-1*, 356–365. [CrossRef]
4. Elhadad, M.; El-Dolil, S.A.; Albagory, Y.A. Application of trigonometric transforms in discrete multi tone systems. In Proceedings of the 2009 International Conference on Computer Engineering & Systems, Cairo, Egypt, 14–16 December 2009; IEEE: Piscataway, NJ, USA, 2010. [CrossRef]
5. Dhamija, S.; Jain, P. Comparative Analysis for Discrete Sine Transform as a suitable method for noise estimation. *Int. J. Comput. Sci. Issues* **2011**, *8*, 162–164.
6. Malini, S.; Moni, R. Use of Discrete Sine Transform for A Novel Image Denoising Technique. *Int. J. Image Process. (IJIP)* **2014**, *8*, 204–213.

7. Choi, J.w.; Kim, N.U.; Lim, S.C.; Kang, J.; Kim, H.Y.; Lee, Y.L. Shuffled Discrete Sine Transform in Inter-Prediction Coding. *ETRI J.* **2017**, *39*, 672–682. [CrossRef]
8. Zhou, X.; Wang, C.; Jiang, B. All Phase Inverse Discrete Sine Biorthogonal Transform and Its Application in Image Coding. *J. Commun.* **2017**, *12*, 72–80. [CrossRef]
9. Joshi, R.; Reznik, Y.A.; Karczewicz, M. Efficient large size transforms for high-performance video coding. In Proceedings of the Applications of Digital Image Processing XXXIII, San Diego, CA, USA, 7 September 2010; Tescher, A.G., Ed.; SPIE: Bellingham, DC, USA, 2010. [CrossRef]
10. Saxena, A.; Fernandes, F.C. DCT/DST-Based Transform Coding for Intra Prediction in Image/Video Coding. *IEEE Trans. Image Process.* **2013**, *22*, 3974–3981. [CrossRef]
11. Budiman, G.; Suksmono, A.B.; Danudirdjo, D.; Pawellang, S. QIM-Based Audio Watermarking with Combined Techniques of SWT-DST-QR-CPT Using SS-Based Synchronization. In Proceedings of the 2018 6th International Conference on Information and Communication Technology (ICoICT), Bandung, Indonesia, 3–5 May 2018; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
12. Ganesh, P.; Menaka, R. Use of Discrete Sine Transform in EEG signal classification for early Autism detection. In Proceedings of the 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, Ramanathapuram, India, 8–10 May 2014; IEEE: Piscataway, NJ, USA, 2014. [CrossRef]
13. Wang, Z.; Wang, L. Interpolation using the fast discrete sine transform. *Signal Process.* **1992**, *26*, 131–137. [CrossRef]
14. Wang, Z.; Jullien, G.; Miller, W. Interpolation using the discrete sine transform with increased accuracy. *Electron. Lett.* **1993**, *29*, 1918. [CrossRef]
15. Kim, M.; Lee, Y.L. Discrete Sine Transform-Based Interpolation Filter for Video Compression. *Symmetry* **2017**, *9*, 257. [CrossRef]
16. Püschel, M.; Moura, J.M.F. The Algebraic Approach to the Discrete Cosine and Sine Transforms and Their Fast Algorithms. *SIAM J. Comput.* **2003**, *32*, 1280–1316. [CrossRef]
17. Wang, Z. Fast discrete sine transform algorithms. *Signal Process.* **1990**, *19*, 91–102. [CrossRef]
18. Gupta, A.; Rao, K. A fast recursive algorithm for the discrete sine transform. *IEEE Trans. Acoust. Speech Signal Process.* **1990**, *38*, 553–557. [CrossRef]
19. Puschel, M.; Moura, J. The discrete trigonometric transforms and their fast algorithms: An algebraic symmetry perspective. In Proceedings of the 2002 IEEE 10th Digital Signal Processing Workshop, 2002 and the 2nd Signal Processing Education Workshop, Pine Mountain, GA, USA, 16 October 2002; IEEE: Piscataway, NJ, USA, 2002; DSPWS-02. [CrossRef]
20. Nikara, J.A.; Takala, J.H.; Astola, J.T. Discrete cosine and sine transforms—Regular algorithms and pipeline architectures. *Signal Process.* **2006**, *86*, 230–249. [CrossRef]
21. Murty, M. Realization of prime-length discrete sine transform using cyclic convolution. *Int. J. Eng. Sci. Technol.* **2013**, *5*, 583–589.
22. Murty, M.N.; Padhy, B. Radix-3 Algorithm for Realization of Type-II Discrete Sine Transform. *Int. J. Eng. Res. Appl.* **2015**, *5*, 9–15.
23. Yip, P.; Wang, F. A prime-factor decomposed algorithm for the discrete sine transform. *Comput. Electr. Eng.* **1990**, *16*, 43–49. [CrossRef]
24. Tsmots, I.; Rabyk, V.; Kryvinska, N.; Yatsymirskyy, M.; Teslyuk, V. Design of the Processors for Fast Cosine and Sine Fourier Transforms. *Circuits Syst. Signal Process.* **2022**, *41*, 4928–4951. [CrossRef]
25. Cariow, A.; Makowska, M.; Strzelec, P. Small-Size FDCT/IDCT Algorithms with Reduced Multiplicative Complexity. *Radioelectron. Commun. Syst.* **2019**, *62*, 559–576. [CrossRef]
26. Cariow, A.; Lesiecki, L. Small-Size Algorithms for Type-IV Discrete Cosine Transform with Reduced Multiplicative Complexity. *Radioelectron. Commun. Syst.* **2020**, *63*, 465–487. [CrossRef]
27. Kolenderski, M.; Cariow, A. Small-Size Algorithms for the Type-I Discrete Cosine Transform with Reduced Complexity. *Electronics* **2022**, *11*, 2411. [CrossRef]
28. Bielak, K.; Cariow, A.; Raciborski, M. The Development of Fast DST-II Algorithms for Short-Length Input Sequences. *Electronics* **2024**, *13*, 2301. [CrossRef]
29. Murty, M. Algorithm for realization of Type-I Discrete Sine Transform. *J. Ultra Sci. Phys. Sci.* **2015**, *27*, 164–168.
30. Al-Fuhaidy, F.A.K.; Al-Sofy, K.A.; Alkamali, F.S. Discrete Sine Transform Based OFDMA System for Wireless Broadband Communications. *AASCIT Commun.* **2019**, *6*, 13–21.
31. Al-kamali, F. New single-carrier transceiver scheme based on the discrete sine transform. *J. Eng.* **2014**, *2014*, 214–218. [CrossRef]
32. Li, X.; Xie, H.; Cheng, B. Noisy Speech Enhancement Based on Discrete Sine Transform. In Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06), Hangzhou, China, 20–24 June 2006; IEEE: Piscataway, NJ, USA, 2006. [CrossRef]
33. Tseng, C.C.; Lee, S.L. Closed-form design of fixed fractional hubert transformer using discrete sine transform. In Proceedings of the 2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Ishigaki, Japan, 17–20 November 2014; IEEE: Piscataway, NJ, USA, 2014; Volume 4, pp. 479–482. [CrossRef]
34. Alonso, P.; Bernabeu, M.O.; Vidal-Maciá, A.M., An Adaptive Interface for the Efficient Computation of the Discrete Sine Transform. In *Parallel Processing and Applied Mathematics*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 89–98. [CrossRef]
35. Yaroslavsky, L.; Wang, Y. DFT, DCT, MDCT, DST and signal Fourier spectrum analysis. *Eur. Signal Process. Conf.* **2015**, *2015*, 1–14.
36. Pant, N. Discrete Sine and Cosine Transforms on Parallel Processors. Master's Thesis, Tampere University of Technology, Tampere, Finland, 2015.
37. Perera, S.M. Signal Flow Graph Approach to Efficient DST I-IV Algorithms. *arXiv* **2016**, arXiv:1601.04662.

38. Pueschel, M.; Moura, J.M.F. Algebraic Signal Processing Theory: Cooley-Tukey Type Algorithms for DCTs and DSTs. *IEEE Trans. Signal Process.* **2008**, *56*, 1502–1521. [CrossRef]
39. Cariow, A. Strategies for the Synthesis of Fast Algorithms for the Computation of the Matrix-vector Products. *J. Signal Process. Theory Appl.* **2014**, *3*, 1–19. [CrossRef]
40. Yip, P.; Rao, K. A Fast Computational Algorithm for the Discrete Sine Transform. *IEEE Trans. Commun.* **1980**, *28*, 304–307. [CrossRef]
41. Agarwal, N.; Solanki, R.; Khan, A. Application of Discrete Sine Transform in Image Processing. *Int. J. Eng. Res. Technol. (IJERT) NCETRASECT* **2015**, *3*, 23. [CrossRef]
42. Olshevsky, A.; Olshevsky, V.; Wang, J. A comrade-matrix-based derivation of the different versions of fast cosine and sine transforms. In Proceedings of the Advanced Signal Processing Algorithms, Architectures, and Implementations XIII, San Diego, CA, USA, 6–8 August 2003; Luk, F.T., Ed.; SPIE: Bellingham, DC, USA, 2003; Volume 5205, pp. 399–410. [CrossRef]
43. Madhukar, B.N.; Jain, S. A duality theorem for the discrete sine transform (DST). In Proceedings of the 2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), Davangere, India, 29–31 October 2015; IEEE: Piscataway, NJ, USA, 2015. [CrossRef]
44. Blahut, R.E. *Fast Algorithms for Signal Processing*; Cambridge University Press: Cambridge, UK, 2010. [CrossRef]
45. Sun, C.; Yip, P. Split-radix algorithms for DCT and DST. In Proceedings of the Twenty-Third Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 30 October–1 November 1989; IEEE: Piscataway, NJ, USA, 1989; pp. 508–512. [CrossRef]
46. Raciborski, M. The Development of Software for Fast DST-I Algorithms for Short-Length Input Sequences; RepOD: Warszawa, Poland, 2024. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.