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

Krystian Bielak 1, Aleksandr Cariow 1,● and Mateusz Raciborski 2,●

1 Faculty of Computer Science and Information Technology, West Pomeranian University of Technology in Szczecin, Zolnierska 49, 71-210 Szczecin, Poland; bikey13@gmail.com
2 Faculty of Computer Science and Telecommunications, Maritime University of Szczecin, Wal CHrobrego 1-2, 70-500 Szczecin, Poland
● Correspondence: atariov@wi.zut.edu.pl (A.C.); m.raciborski@pm.szczecin.pl (M.R.)

Abstract: The subject of this work is the development of fast algorithms for the discrete sinusoidal transformation of the second type (DST-II) for sequences of input data of small length $N = 2, 3, 4, 5, 6, 7, 8$. The starting point for the development of algorithms is the well-known possibility of representing any discrete transformation in the form of a matrix–vector product. Due to the remarkable structural properties of the matrices of the DST-II transformation base, these matrices can be successfully factorized, which should lead to a reduction in the computational complexity of the procedure as a whole. You can factorize matrices in different ways. The art of designing fast algorithms is to find the factorization that produces the maximum effect. We justified the correctness of the obtained algorithmic solutions theoretically, using strict mathematical derivations of each of them. The developed algorithms were then further tested using MATLAB R2023b software to finally confirm their performance. Finally, we presented estimates of the computational complexity for each solution obtained and compared them with direct computational methods that rely on the direct calculation of matrix–vector products.

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

1. Introduction

Discrete trigonometric transforms are widely used in solving problems in many modern computing systems for digital signal and image processing, including filtering and denoising, noisy speech enhancement, interpolation, video coding, etc. [1–10]. There are eight different types of discrete cosine transform and eight types of discrete sine transform [11]. The popularity of the discrete cosine transform is based on the fact that it closely approximates the optimal Karhunen–Löwe transform (KLT) under a stationary first-order Markov condition with strong inter-pixel correlations. However, for low-correlation input signals, discrete sine transform (DST) provides lower data rates [12–14] because, like other orthogonal transforms, implementing the discrete sine transforms requires much time to search for algorithmic solutions. To reduce this time is an important task; this problem can be solved in two ways. One direction is the hardware implementation of calculations [15–18] and the other one is the reduction of the number of arithmetic operations necessary to implement the transform. A large number of papers is devoted to the development of effective algorithms for the implementation of various discrete cosine transforms (DCTs) and DSTs, but most of them pursue the search for universal solutions that allow reducing the number of arithmetic operations for arbitrary lengths of input data sequences [19–25]. There is a third way, which also has a right to exist. This is an approximation of discrete trigonometric transforms. To date, a large number of algorithms have been developed that use approximations of the DCT/DST transforms.
Approximation algorithms for sequences of standard lengths $N = 4$, 8, and 16 are known \cite{26–30}. However, the development of reduced complexity algorithms for traditional small-size DCT/DST transforms has not been canceled. Some applications require the use of conventional DCT/DST transforms for various short-length input data sequences. This is explained by the fact that algorithms for small-size transforms can serve as kernels for the synthesis of larger algorithms \cite{30,31}. A fairly large number of works have been devoted to the development of small-sized DCT algorithms, and much less attention has been paid to similar algorithms for the DST. Among the other types of discrete trigonometric transforms, DCT-II/DST-II plays an important role \cite{29,32–35}. For small-sized DCT-II, the algorithms were shown in one of our previous papers \cite{36}. We did not find any small-sized type II DSTs in the sources known to us. To fill this gap, we are developing fast algorithms for low-dimensional discrete trigonometric transformations to expand their collection. This paper is devoted to reduced complexity DST-II algorithms for input sequences of length $N = 2, 3, 4, 5, 6, 7, 8$.

2. Short Background

The discrete sine transform is one of the orthogonal transforms used, among others, for the analysis and processing of sounds and signals. DST-II can be represented by the following expression:

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

where

- $k = 0, \ldots, N - 1$;
- $\epsilon_k$ equals $\frac{1}{\sqrt{2}}$ for $k = N - 1$ and equals 1 for the remaining $k$;
- $y$ is the output sequence after the DST-II operation is performed;
- $x_n$ is the sequence of input data;
- $N$ is the number of signal samples.

In matrix notation, DST-II can be represented as follows:

$$Y_{N \times 1} = C_N X_{N \times 1}$$  \hspace{1cm} (2)

where

$$Y_{N \times 1} = [y_0, y_1, \ldots, y_{N-1}]^T, \quad X_{N \times 1} = [x_0, x_1, \ldots, x_{N-1}]^T,$$

$$y_{k,l} = \sqrt{\frac{2}{N}} \epsilon_k \sin \left[ \frac{(l + \frac{1}{2})(k + 1)\pi}{N} \right]$$

where

- $k, l = 0, \ldots, N - 1$;
- $\epsilon_k$ equals $\frac{1}{\sqrt{2}}$ for $k = N - 1$ and equals 1 for the remaining $k$.

DST-II in matrix notation is as follows:

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,N-1} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N-1,0} & c_{N-1,1} & \cdots & c_{N-1,N-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}.$$
In this paper, we use the following markings and signs:

- $I_N$ is an order $N$ identity matrix;
- $H_2$ is a $2 \times 2$ Hadamard matrix;
- $\otimes$ is the Kronecker product of two matrices;
- $\oplus$ is the direct sum of two matrices.

An empty cell in a matrix means it contains zero. We mark the multipliers as $s_{m}^{(N)}$, but we do not use a superscript in the data flow graphs in order to maintain greater readability and elegance.

3. Algorithm for 2-Point DST-II

The expression for two-point DST-II is as follows:

$$Y_{2 \times 1} = C_2 X_{2 \times 1}$$

where

$$Y_{2 \times 1} = [y_0, y_1]^T, \quad X_{2 \times 1} = [x_0, x_1]^T, \quad C_2 = \begin{bmatrix} a_2 & a_2 \\ a_2 & -a_2 \end{bmatrix}, \quad a_2 = 0.7071.$$  

The expression for DST-II for $N = 2$ can be presented as follows:

$$Y_{2 \times 1} = H_2 D_2 X_{2 \times 1}$$

where

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad D_2 = \text{diag}(s_{2}^{(2)}, s_{1}^{(2)}), \quad s_{2}^{(2)} = s_{1}^{(2)} = a_2.$$  

Figure 1 shows a data flow graph of the synthesized algorithm for the two-point DST-II. As can be seen, we are able to reduce the number of multiplication operations from 4 to 2, while the number of addition operations is 2, which is the same as when using the direct method.

![Figure 1](image)

**Figure 1.** The data flow graph of the proposed algorithm for computation of two-point DST-II.

4. Algorithm for 3-Point DST-II

The expression for three-point DST-II is as follows:

$$Y_{3 \times 1} = C_3 X_{3 \times 1}$$

where

$$Y_{3 \times 1} = [y_0, y_1, y_2]^T, \quad X_{3 \times 1} = [x_0, x_1, x_2]^T, \quad C_3 = \begin{bmatrix} a_3 & d_3 & a_3 \\ b_3 & 0 & -b_3 \\ c_3 & -c_3 & c_3 \end{bmatrix}, \quad a_3 = 0.4082, \quad b_3 = 0.7071, \quad c_3 = 0.5774, \quad d_3 = 0.8165.$$  

Now, we will decompose the matrix $C_3$ into two components:

$$C_3 = C_3^{(a)} + C_3^{(b)}$$

where

$$C_3^{(a)} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ c_3 & -c_3 & c_3 \end{bmatrix}, \quad C_3^{(b)} = \begin{bmatrix} a_3 & d_3 & a_3 \\ b_3 & 0 & -b_3 \\ c_3 & -c_3 & c_3 \end{bmatrix}.$$
After eliminating redundancy in matrix $C^{(b)}_3$ and eliminating rows and columns containing only zero entries, we obtain matrix $C_2$:

$$C_2 = \begin{bmatrix} a_3 & b_3 \\ b_3 & -b_3 \end{bmatrix}.$$ 

Thanks to the already noted remarkable properties of structural matrices, the final computational procedure for the three-point DST-II takes the following form:

$$Y_{3 \times 1} = P_{3 \times 4} D_4^{(0)} W_4^{(0)} P_{4 \times 3} X_{3 \times 1}$$

where

$$P_{4 \times 3} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \\ 1 & -1 & -1 \end{bmatrix}, \quad W_4^{(0)} = H_2 \oplus I_2, \quad D_4^{(0)} = \text{diag} \left( s_0^{(3)}, s_1^{(3)}, s_2^{(3)}, s_3^{(3)} \right),$$

$$s_0^{(3)} = a_3, \quad s_1^{(3)} = b_3, \quad s_2^{(3)} = c_3, \quad s_3^{(3)} = d_3, \quad P_{3 \times 4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{bmatrix}.$$ 

Figure 2 shows a data flow graph of the synthesized algorithm for the three-point DST-II. As can be seen, we are able to reduce the number of multiplication operations from 9 to 4, while the number of addition operations is 5, which is the same as when using the direct method.

5. Algorithm for 4-Point DST-II

The expression for four-point DST-II is as follows:

$$Y_{4 \times 1} = C_4 X_{4 \times 1}$$

where

$$Y_{4 \times 1} = [y_0, y_1, y_2, y_3]^T, \quad C_4 = \begin{bmatrix} a_4 & c_4 & c_4 & a_4 \\ b_4 & b_4 & -b_4 & -b_4 \\ c_4 & -c_4 & c_4 & -c_4 \\ -b_4 & b_4 & -b_4 & b_4 \end{bmatrix}, \quad a_4 = 0.2706, \quad b_4 = 0.5, \quad c_4 = 0.6533.$$ 

Now, we need to change the order of columns and rows. Let us define the permutations $\pi_4^{(0)}$ and $\pi_4^{(1)}$ in the following form:

$$\pi_4^{(0)} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 4 & 3 \end{bmatrix}, \quad \pi_4^{(1)} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \end{bmatrix}.$$
Permute columns of $C_4$ according to $\pi_4^{(0)}$ and rows according to $\pi_4^{(1)}$. After permutations, the matrix acquires the following structure:

$$\begin{bmatrix}
A_2 & 0 & A_2 \\
0 & B_2 & 0 \\
B_2 & 0 & B_2
\end{bmatrix}$$

where $A_2 = \begin{bmatrix} a_4 & c_4 & 0 \\
c_4 & -a_4 & 0 \\
0 & 0 & 0
\end{bmatrix}$, $B_2 = \begin{bmatrix} b_4 & c_4 & 0 \\
c_4 & -b_4 & 0 \\
0 & 0 & 0
\end{bmatrix}$.

Matrices with such a structure allow effective factorization, which leads to a reduction in the number of arithmetic operations when calculating matrix–vector products [37]. In this work, we preserve designations of matrices $T_{2\times3}^{(4)}$, $T_{3\times2}^{(3)}$ and $T_{2\times3}^{(3)}$ taken from [37]. Matrices $A_2$ and $B_2$ also have remarkable structures that reduce computational complexity.

Taking this into account, we can derive the final expression:

$$Y_{4\times1} = P_4^{(1)} W_{4\times5} D_5 W_{5\times4} W_4^{(1)} P_4^{(0)} X_{4\times1}$$

where

$$P_4^{(0)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix}, \quad W_4^{(1)} = H_2 \otimes I_2, \quad W_{5\times4} = T_{3\times2}^{(5)} \oplus H_2, \quad T_{3\times2}^{(5)} = \begin{bmatrix} 1 & -1 & -1 \\
-1 & 1 & -1 \\
-1 & -1 & 1
\end{bmatrix},$$

$$D_5 = \text{diag}(s_0^{(4)}, s_1^{(4)}, \ldots, s_4^{(4)}), \quad s_0^{(4)} = a_4 - c_4, \quad s_1^{(4)} = a_4 + c_4,$$

$$s_2^{(4)} = c_4, \quad s_3^{(4)} = s_4^{(4)} = b_4,$$

$$W_{4\times5} = T_{2\times3}^{(4)} \oplus I_2, \quad T_{2\times3}^{(4)} = \begin{bmatrix} 1 & 1 \\
1 & 1
\end{bmatrix}, \quad P_4^{(1)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix}.$$

Figure 3 shows a data flow graph of the synthesized algorithm for four-point DST-II. As can be seen, we are able to reduce the number of multiplication operations from 8 to 3 and the number of addition operations from 12 to 9.

Figure 3. The data flow graph of the proposed algorithm for computation of four-point DST-II.

6. Algorithm for 5-Point DST-II

The expression for five-point DST-II is as follows:

$$Y_{5\times1} = C_5 X_{5\times1}$$

(9)
where

\[
Y_{5 \times 1} = [y_0, y_1, y_2, y_3, y_4]^T, \quad X_{5 \times 1} = [x_0, x_1, x_2, x_3, x_4]^T,
\]

\[
C_5 \begin{bmatrix} a_5 & c_5 & f_5 & c_5 & a_5 \\ b_5 & d_5 & 0 & -d_5 & -b_5 \\ c_5 & -d_5 & 1 - f_5 & a_5 & c_5 \\ d_5 & -b_5 & 0 & b_5 & -d_5 \\ e_5 & -e_5 & 1 - e_5 & e_5 & 1 \end{bmatrix},
\]

\[
a_5 = 0.1954, \quad b_5 = 0.3717, \quad c_5 = 0.5117, \quad d_5 = 0.6015, \quad e_5 = 0.4472, \quad f_5 = 0.6325.
\]

Now, we will decompose the matrix \( C_5 \) into two components:

\[
C_5 = C_5^{(a)} + C_5^{(b)}
\]

where

\[
C_5^{(a)} \begin{bmatrix} a_5 & c_5 & f_5 & c_5 & a_5 \\ b_5 & d_5 & 0 & -d_5 & -b_5 \\ c_5 & -d_5 & 1 - f_5 & a_5 & c_5 \\ d_5 & -b_5 & 0 & b_5 & -d_5 \\ e_5 & -e_5 & 1 - e_5 & e_5 & 1 \end{bmatrix},
\]

\[
C_5^{(b)} \begin{bmatrix} a_5 & c_5 & f_5 & c_5 & a_5 \\ b_5 & d_5 & 0 & -d_5 & -b_5 \\ c_5 & -d_5 & 1 - f_5 & a_5 & c_5 \\ d_5 & -b_5 & 0 & b_5 & -d_5 \\ e_5 & -e_5 & 1 - e_5 & e_5 & 1 \end{bmatrix}.
\]

The matrix \( C_5^{(a)} \) has one entry in the first and third rows and five entries with the same value in the fifth row, which means that the number of operations is small and we do not need to perform further transformations for this matrix.

After eliminating redundancy in matrix \( C_5^{(b)} \) and eliminating rows and columns containing only zero entries, we obtain matrix \( C_4 \):

\[
C_4 \begin{bmatrix} a_5 & c_5 & f_5 & c_5 & a_5 \\ b_5 & d_5 & 0 & -d_5 & -b_5 \\ c_5 & -d_5 & 1 - f_5 & a_5 & c_5 \\ d_5 & -b_5 & 0 & b_5 & -d_5 \\ e_5 & -e_5 & 1 - e_5 & e_5 & 1 \end{bmatrix}.
\]

We permute columns of \( C_4 \) according to \( \pi_4^{(0)} \) and rows according to \( \pi_4^{(1)} \). After permutations, the matrix matches the matrix pattern:

\[
\begin{bmatrix} A_2 & A_2 \\ B_2 & -B_2 \end{bmatrix}
\]

where

\[
A_2 \begin{bmatrix} a_5 & c_5 \\ c_5 & a_5 \end{bmatrix}, \quad B_2 \begin{bmatrix} b_5 & d_5 \\ d_5 & -b_5 \end{bmatrix}.
\]

Considering the structures of the resulting matrices, the final computational procedure can be derived as:

\[
Y_{5 \times 1} = P_{5 \times 6} W_{6 \times 7} D_7 W_{7 \times 6} W^{(0)}_6 P_{6 \times 5} X_{5 \times 1} \tag{10}
\]

where

\[
P_{6 \times 5} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},
\]

\[
W^{(0)}_6 = 1 \oplus W^{(1)}_4 \oplus 1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},
\]

\[
W_{7 \times 6} = 1 \oplus H_2 \oplus T^{(5)}_{3 \times 2} \oplus 1,
\]
\[
D_7 = \text{diag}\left(s_0^{(5)}, s_1^{(5)}, \ldots, s_6^{(5)}\right), \quad s_0^{(5)} = f_5, \quad s_1^{(5)} = \frac{a_5 + c_5}{2}, \quad s_2^{(5)} = \frac{a_5 - c_5}{2},
\]

\[
s_3^{(5)} = b_5 - d_5, \quad s_4^{(5)} = b_5 + d_5,
\]

\[
s_5^{(5)} = d_5, \quad s_6^{(5)} = e_5,
\]

\[
W_{6 \times 7} = 1 \oplus H_2 \oplus T_{2,3}^{(4)} \oplus 1,
\]

Figure 4 shows a data flow graph of the synthesized algorithm for five-point DST-II. As can be seen, we are able to reduce the number of multiplication operations from 23 to 7 and the number of addition operations from 18 to 17.

7. Algorithm for 6-Point DST-II

The expression for six-point DST-II is as follows:

\[
Y_{6 \times 1} = C_6 X_{6 \times 1}
\]

where

\[
Y_{6 \times 1} = [y_0, y_1, y_2, y_3, y_4, y_5]^T, \quad X_{6 \times 1} = [x_0, x_1, x_2, x_3, x_4, x_5]^T,
\]

\[
C_6 = \begin{bmatrix}
  a_6 & c_6 & 0 & 0 & 0 & 0 \\
  d_6 & f_6 & 0 & 0 & 0 & 0 \\
  c_6 & -c_6 & a_6 & b_6 & 0 & 0 \\
  d_6 & f_6 & 0 & 0 & 0 & 0 \\
  c_6 & -c_6 & a_6 & b_6 & 0 & 0 \\
  c_6 & -c_6 & a_6 & b_6 & 0 & 0 \\
\end{bmatrix}, \quad a_6 = 0.1494, \quad b_6 = 0.2887,
\]

\[
C_6 = \begin{bmatrix}
  a_6 & c_6 & 0 & 0 & 0 & 0 \\
  d_6 & f_6 & 0 & 0 & 0 & 0 \\
  c_6 & -c_6 & a_6 & b_6 & 0 & 0 \\
  d_6 & f_6 & 0 & 0 & 0 & 0 \\
  c_6 & -c_6 & a_6 & b_6 & 0 & 0 \\
  c_6 & -c_6 & a_6 & b_6 & 0 & 0 \\
\end{bmatrix}, \quad a_6 = 0.4082, \quad d_6 = 0.5,
\]

\[
C_6 = \begin{bmatrix}
  a_6 & c_6 & 0 & 0 & 0 & 0 \\
  d_6 & f_6 & 0 & 0 & 0 & 0 \\
  c_6 & -c_6 & a_6 & b_6 & 0 & 0 \\
  d_6 & f_6 & 0 & 0 & 0 & 0 \\
  c_6 & -c_6 & a_6 & b_6 & 0 & 0 \\
  c_6 & -c_6 & a_6 & b_6 & 0 & 0 \\
\end{bmatrix}, \quad e_6 = 0.5577, \quad f_6 = 0.5774.
\]

Now, we will decompose the matrix \( C_6 \) into two components:

\[
C_6 = C_6^{(a)} + C_6^{(b)}
\]
where
\[
C_6^{(a)} = \begin{bmatrix}
\begin{array}{cccc}
\cdots & c_6 & \cdots & f_6 \\
\cdots & c_6 & \cdots & -f_6 \\
\cdots & -c_6 & \cdots & c_6 \\
\cdots & c_6 & \cdots & -c_6 \\
\cdots & c_6 & \cdots & c_6 \\
\cdots & -c_6 & \cdots & -c_6
\end{array}
\end{bmatrix}
\quad \text{and} \quad
C_6^{(b)} = \begin{bmatrix}
\begin{array}{cccc}
a_6 & e_6 & e_6 & a_6 \\
\cdots & b_6 & \cdots & b_6 \\
\cdots & d_6 & \cdots & d_6 \\
\cdots & e_6 & \cdots & e_6 \\
\cdots & a_6 & \cdots & a_6 \\
\cdots & d_6 & \cdots & d_6 \\
\cdots & c_6 & \cdots & c_6
\end{array}
\end{bmatrix}
\]

Matrix $C_6^{(a)}$ has two of the same entries in the first, second, and fifth rows and six entries with the same value in the third and sixth rows, which allows us to reduce the number of operations without the need for further transformations.

After eliminating redundancy in matrix $C_6^{(b)}$ and eliminating rows and columns containing only zero entries, we obtain matrix $C_4$:
\[
C_4 = \begin{bmatrix}
\begin{array}{cccc}
\cdots & a_6 & e_6 & e_6 \\
\cdots & b_6 & \cdots & b_6 \\
\cdots & d_6 & \cdots & d_6 \\
\cdots & e_6 & \cdots & e_6 \\
\cdots & a_6 & \cdots & a_6 \\
\cdots & d_6 & \cdots & d_6 \\
\cdots & c_6 & \cdots & c_6
\end{array}
\end{bmatrix}
\]

Let us define the permutation $\pi_4^{(2)}$ in the following form:
\[
\pi_4^{(2)} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix}
\]

We permute columns of $C_4$ according to $\pi_4^{(0)}$ and rows according to $\pi_4^{(2)}$. After permutation, the matrix matches the matrix pattern:
\[
P_{8\times 6} = \begin{bmatrix}
\begin{array}{cccccccc}
A_2 & B_2 & A_2 \\
\cdots & \cdots & \cdots
\end{array}
\end{bmatrix}
\]

where
\[
A_2 = \begin{bmatrix}
\begin{array}{ccc}
a_6 & e_6 & e_6 \\
e_6 & c_6 & e_6 \\
\end{array}
\end{bmatrix}, \quad
B_2 = \begin{bmatrix}
\begin{array}{ccc}
b_6 & b_6 & b_6 \\
d_6 & d_6 & d_6 \\
\end{array}
\end{bmatrix}
\]

Taking this into account, we can derive the final expression:
\[
Y_{6\times 1} = P_{6\times 8} W_8^{(2)} D_8 W_8^{(1)} W_8^{(0)} P_{8\times 6} X_{6\times 1}
\]

where
\[
P_{8\times 6} = \begin{bmatrix}
\begin{array}{cccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{array}
\end{bmatrix}
\]

with
\[
W_8^{(0)} = 1 \oplus W_4^{(1)} \oplus I_3, \quad W_8^{(1)} = 1 \oplus H_2 \oplus H_2 \oplus I_3, \quad D_8 = \text{diag}(s_0^{(6)}, s_1^{(6)}, \ldots, s_7^{(6)}),
\]

and
\[
s_0^{(6)} = f_6, \quad s_1^{(6)} = \frac{a_6 + e_6}{2}, \quad s_2^{(6)} = \frac{a_6 - e_6}{2}, \quad s_3^{(6)} = b_6, \quad s_4^{(6)} = d_6, \quad s_5^{(6)} = s_6^{(6)} = s_7^{(6)} = c_6,
\]

where
\[
W_8^{(2)} = 1 \oplus H_2 \oplus I_5,
\]
Figure 5 shows a data flow graph of the synthesized algorithm for six-point DST-II. As can be seen, we are able to reduce the number of multiplication operations from 30 to 7 and the number of addition operations from 28 to 25.

Figure 5. The data flow graph of the proposed algorithm for computation of six-point DST-II.

8. Algorithm for 7-Point DST-II

The expression for seven-point DST-II is as follows:

\[ Y_{7 	imes 1} = C_7 X_{7 	imes 1} \] (13)

where

\[
Y_{7 	imes 1} = \begin{bmatrix} y_0, y_1, y_2, y_3, y_4, y_5, y_6 \end{bmatrix}^T, \quad X_{7 	imes 1} = \begin{bmatrix} x_0, x_1, x_2, x_3, x_4, x_5, x_6 \end{bmatrix}^T
\]

\[
C_7 = \begin{bmatrix}
    a_7 & c_7 & e_7 & h_7 & e_7 & c_7 & a_7 \\
    b_7 & f_7 & d_7 & 0 & d_7 & -f_7 & -b_7 \\
    c_7 & -d_7 & -c_7 & -h_7 & -d_7 & c_7 & e_7 \\
    d_7 & b_7 & 0 & f_7 & -b_7 & -d_7 \\
    e_7 & -c_7 & -h_7 & -e_7 & -c_7 & -h_7 & -e_7 \\
    f_7 & -d_7 & b_7 & 0 & -b_7 & -d_7 & -f_7 \\
    g_7 & -g_7 & g_7 & -g_7 & g_7 & -g_7 & g_7 \\
\end{bmatrix}
\]

\[ a_7 = 0.1189, \quad b_7 = 0.2319, \quad c_7 = 0.3333, \quad d_7 = 0.4179, \quad e_7 = 0.4816, \quad f_7 = 0.5211, \quad g_7 = 0.3780, \quad h_7 = 0.5345. \]

Now, we will decompose the matrix \( C_7 \) into two components:

\[ C_7 = C_7^{(a)} + C_7^{(b)} \]
Electronics 2024, 13, 2301

The calculation procedure for the circular convolution matrix for \( N = 3 \) is presented below:

\[
H_3 = T_3^{(1)} T_{3\times4} D_{4\times6}^{(1)} T_{4\times3} T_3^{(0)}
\]
where

\[ T^{(0)}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix}, \quad T_{4 \times 3} = \begin{bmatrix} 1 & 1 & \cdots \\ \vdots & \vdots & \ddots \\ 1 & 1 & \cdots \end{bmatrix}, \quad D^{(1)}_4 = \text{diag}(s_0, s_1, s_2, s_3). \]

\[ T_{3 \times 4} = \begin{bmatrix} 1 & 1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ 1 & 1 & 1 & \cdots \end{bmatrix}, \quad T^{(1)}_3 = \begin{bmatrix} 1 & 1 & \cdots \\ \vdots & \vdots & \ddots \\ 1 & 1 & \cdots \end{bmatrix}. \]

To make the \( A_3 \) and \( B_3 \) matrices consistent with the circular convolution expression, we need to modify them. In the \( A_3 \) matrix, we change the sign of all terms in the second column and third row. In the \( B_3 \) matrix, we change the sign in the first row and first column. In this way, we obtain the matrices:

\[ A^{'}_3 = \begin{bmatrix} a_7 & -c_7 & c_7 \\ -c_7 & a_7 & -c_7 \\ c_7 & -c_7 & a_7 \end{bmatrix}, \quad B^{'}_3 = \begin{bmatrix} b_7 & -f_7 & -d_7 \\ -f_7 & b_7 & -f_7 \\ -d_7 & -f_7 & b_7 \end{bmatrix}. \]

Using the three-point convolution algorithm, the values \( s_i \) for matrices \( A^{'}_3 \) and \( B^{'}_3 \) take the following form:

\[ s^{(7)}_1 = \frac{a_7 + c_7 - c_7}{3}, \quad s^{(7)}_2 = a_7 + c_7, \quad s^{(7)}_3 = c_7, \quad s^{(7)}_4 = \frac{a_7 + c_7 + 2c_7}{3}, \]

\[ s^{(7)}_5 = \frac{b_7 - d_7 - f_7}{3}, \quad s^{(7)}_6 = b_7 + f_7, \quad s^{(7)}_7 = -d_7 + f_7, \quad s^{(7)}_8 = \frac{b_7 - d_7 + 2f_7}{3}. \]

Considering the presented derivations, the final computational procedure for 7-point DST-II will look as follows.

\[ Y_{7 \times 1} = P_{7 \times 8} W^{(7)}_8 W^{(6)}_8 W^{(5)}_{8 \times 10} D_{10} W^{(0)}_{10 \times 8} W^{(4)}_8 W^{(3)}_8 P_{8 \times 7} X_{7 \times 1} \quad (14) \]

where

\[ W^{(3)}_8 = 1 \oplus W^{(1)}_6 \oplus 1, \quad W^{(1)}_6 = H_2 \otimes I_3, \quad W^{(4)}_8 = I_2 \oplus (-1) \oplus 1 \oplus (-1) \oplus I_3, \]

\[ W^{(5)}_8 = 1 \oplus T^{(0)}_3 \oplus T^{(0)}_3 \oplus 1, \]

\[ W^{(0)}_{8 \times 10} = 1 \oplus T^{(3)}_3 \oplus T^{(3)}_3 \oplus 1, \quad D_{10} = \text{diag}(s^{(7)}_0, s^{(7)}_1, \ldots, s^{(7)}_9), \]

\[ P_{8 \times 7} = \begin{bmatrix} 1 & 1 & \cdots \\ \vdots & \vdots & \ddots \\ 1 & 1 & \cdots \end{bmatrix}, \quad P_{7 \times 8} = \begin{bmatrix} 1 & 1 & \cdots \\ \vdots & \vdots & \ddots \\ 1 & 1 & \cdots \end{bmatrix}. \]
Figure 6 shows a data flow graph of the synthesized algorithm for seven-point DST-II. As can be seen, we are able to reduce the number of multiplication operations from 46 to 10 and the number of addition operations from 39 to 37.

9. Algorithm for 8-Point DST-II

The expression for eight-point DST-II is as follows:

$$Y_{8\times 1} = C_8 X_{8\times 1}$$  \hspace{1cm} (15)$$

where

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

$$C_8 = \begin{bmatrix}
    a_8 & c_8 & g_8 & c_8 & c_8 & a_8 & b_8 & f_8 \\
    b_8 & f_8 & a_8 & c_8 & g_8 & f_8 & b_8 & c_8 \\
    c_8 & g_8 & a_8 & b_8 & c_8 & c_8 & a_8 & g_8 \\
    g_8 & f_8 & a_8 & b_8 & c_8 & c_8 & a_8 & g_8 \\
    c_8 & g_8 & a_8 & b_8 & c_8 & c_8 & a_8 & g_8 \\
    g_8 & f_8 & a_8 & b_8 & c_8 & c_8 & a_8 & g_8 \\
    b_8 & f_8 & a_8 & c_8 & g_8 & f_8 & b_8 & c_8 \\
    f_8 & b_8 & f_8 & a_8 & c_8 & g_8 & f_8 & b_8 \\
\end{bmatrix}, \quad a_8 = 0.0975, \quad b_8 = 0.1913, \quad c_8 = 0.2778, \quad d_8 = 0.3536, \quad e_8 = 0.4157, \quad f_8 = 0.4619, \quad g_8 = 0.4904.$$

Let us define the permutations $\pi_8^{(0)}$ and $\pi_8^{(1)}$ in the following form:

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

We permute columns of $C_8$ according to $\pi_8^{(0)}$ and rows according to $\pi_8^{(1)}$. After permutation, the matrix matches the matrix pattern:

$$\begin{bmatrix}
    A_4 & A_4 \\
    B_4 & -B_4 \\
\end{bmatrix}$$
where
\[
A_4 = \begin{bmatrix}
\varphi & c_8 & e_8 & g_8 \\
\varphi & a_8 & -c_8 & e_8 \\
\varphi & g_8 & -a_8 & -c_8 \\
\varphi & -e_8 & c_8 & -a_8
\end{bmatrix}, \quad B_4 = \begin{bmatrix}
\varphi & f_8 & g_8 & d_8 \\
\varphi & -b_8 & -d_8 & f_8 \\
\varphi & d_8 & f_8 & -b_8 \\
\varphi & -d_8 & -f_8 & d_8
\end{bmatrix}.
\]

After this operation, the calculation procedure is as follows:
\[
Y_{8 \times 1} = P_8^{(1)} D_8 W_8^{(8)} P_8^{(0)} X_{8 \times 1}
\]

where
\[
W_8^{(8)} = H_2 \otimes I_4,
\]
\[
P_8^{(0)} = \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix}, \quad P_8^{(1)} = \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix}.
\]

Now, we will deal with matrices \(A_4\) and \(B_4\). The matrix \(A_4\) does not fit any pattern and we need to modify it. We do this by changing the sign for the third column. In this way, we obtain a matrix that looks like this:
\[
A'_4 = \begin{bmatrix}
\varphi & c_8 & -e_8 & g_8 \\
\varphi & a_8 & -c_8 & e_8 \\
\varphi & g_8 & -a_8 & -c_8 \\
\varphi & e_8 & c_8 & -a_8
\end{bmatrix}
\]

Let us define the permutation \(\pi_4^{(3)}\) in the following form:
\[
\pi_4^{(3)} = \begin{pmatrix}
1 & 2 & 3 & 4 \\
2 & 1 & 3 & 4
\end{pmatrix}.
\]

We permute columns of \(A'_4\) according to \(\pi_4^{(0)}\) and rows according to \(\pi_4^{(3)}\). Now, \(A'_4\) fits the pattern:
\[
A'_4 = \begin{bmatrix}
A_2 & B_2 \\
B_2 & -A_2
\end{bmatrix} \quad \text{where} \quad A_2 = \begin{bmatrix}
\varphi & c_8 \\
\varphi & a_8
\end{bmatrix}, \quad B_2 = \begin{bmatrix}
\varphi & g_8 \\
\varphi & -e_8
\end{bmatrix}.
\]

Let us permute columns of \(B_4\) according to \(\pi_4^{(0)}\). Then, we are able to use the matrix pattern:
\[
B'_4 = \begin{bmatrix}
E_2 & E_2 \\
F_2 & F_2
\end{bmatrix} \quad \text{where} \quad E_2 = \begin{bmatrix}
\varphi & f_8 \\
\varphi & -b_8
\end{bmatrix}, \quad F_2 = \begin{bmatrix}
\varphi & d_8 \\
\varphi & -d_8
\end{bmatrix}.
\]

After this operation, the calculation procedure is as follows:
\[
Y_{8 \times 1} = P_8^{(1)} P_8^{(3)} W_8^{(1)} D_8 W_8^{(1)} W_8^{(0)} P_8^{(2)} W_8^{(9)} W_8^{(8)} F_8^{(0)} X_{8 \times 1}
\]
As can be seen, we are able to reduce the number of multiplication operations from 64 to 14 and the number of addition operations from 56 to 32.

Finally, we will deal with five matrices of size 2. In the matrices $G_2, J_2,$ and $B_2$ we need to swap every row. Then, all matrices fit the pattern of the matrix:

$$\begin{bmatrix}
1 & 1 \\
-1 & -1 \\
0 & 0
\end{bmatrix}$$

The matrices $E_2$ and $F_2$ immediately fit this pattern:

$$\begin{bmatrix}
a & b \\
-c & d
\end{bmatrix}$$

Taking into account the matrix structures described above, the final computational procedure for the 8-point DST-II can be written as follows:

$$Y_{8 \times 1} = P_{8}^{(1)} P_{8}^{(3)} W_{8 \times 10}^{(1)} P_{10} W_{10 \times 14}^{(1)} D_{14} W_{14 \times 10}^{(1)} W_{10 \times 8}^{(1)} P_{8}^{(2)} W_{8}^{(9)} P_{8}^{(0)} X_{8 \times 1}$$

where

$$W_{14 \times 10} = T_{3 \times 2}^{(3)} \oplus T_{3 \times 2}^{(3)} \oplus T_{3 \times 2}^{(3)} \oplus T_{3 \times 2}^{(3)} \oplus I_{2}, \quad D_{14} = \text{diag}(s_{0}^{(8)}, s_{1}^{(8)}, \ldots, s_{13}^{(8)}),$$

$$s_{0}^{(8)} = e_{8} - c_{8} - a_{8} + g_{8}, \quad s_{1}^{(8)} = c_{8} + e_{8} - a_{8} + g_{8}, \quad s_{2}^{(8)} = a_{8} - g_{8},$$

$$s_{3}^{(8)} = -e_{8} - c_{8} + a_{8} + g_{8}, \quad s_{4}^{(8)} = -c_{8} + e_{8} + a_{8} + g_{8}, \quad s_{5}^{(8)} = -a_{8} - g_{8},$$

$$s_{b}^{(8)} = p_{b}^{(8)} \quad s_{12}^{(8)} = s_{13}^{(8)} = d_{8},$$

$$W_{10 \times 14} = T_{2 \times 3}^{(3)} \oplus T_{2 \times 3}^{(3)} \oplus T_{2 \times 3}^{(3)} \oplus T_{2 \times 3}^{(3)} \oplus I_{2}, \quad T_{2 \times 3}^{(3)} = \begin{bmatrix}
1 & 1 \\
1 & 1 \\
1 & 1
\end{bmatrix},$$

$$W_{10 \times 8}^{(1)} = I_{8} \oplus H_{2}, \quad P_{10} = P_{2} \oplus P_{2} \oplus P_{2} \oplus I_{4}, \quad P_{2} = \begin{bmatrix}
1 & 1 \\
1 & 1
\end{bmatrix}.$$
Figure 7. The data flow graph of the proposed algorithm for computation of eight-point DST-II.

10. Discussion of Computational Complexity

Firstly, we explain how to calculate the number of multiplication and addition operations for the direct DST-II calculation method and proposed solutions. For any number that is a power of two, a shift can be used instead of a multiplication operation. If the value is zero, then we do not count addition and multiplication operations for it.

The above appear in the matrices: $C_3$—one zero; $C_4$—eight values of 0.5; $C_5$—two zeros; $C_6$—two zeros and four values of 0.5; $C_7$—three zeros. In the proposed solutions, in diagonal matrices are the following: $D_5$—two values of 0.5; $D_8$—one value of 0.5.

The work shows how it is possible to reduce the number of multiplication operations in DST-II algorithms of sizes 2 to 8. At the same time, the number of addition operations was slightly reduced. The number of addition operations was reduced by an average of 20%, and the number of multiplication operations was reduced by an average of 74%. The achieved results are presented in Table 1, which contains data by taking into account the above rules.

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

<table>
<thead>
<tr>
<th></th>
<th>Direct Method</th>
<th>Proposed Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>Additions</td>
<td>Multiplications</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>23</td>
</tr>
<tr>
<td>6</td>
<td>28</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>39</td>
<td>46</td>
</tr>
<tr>
<td>8</td>
<td>56</td>
<td>64</td>
</tr>
</tbody>
</table>

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

Each proposed algorithm has been implemented in the MATLAB environment and we are confident that they all work correctly.

11. Conclusions

To date, many papers have already been published concerning the development of fast algorithms for implementing discrete trigonometric transforms [1,2,11,20,27]. These studies have not lost their relevance today. The presented article is a continuation of these studies. For well-known reasons, we have focused on developing fast algorithms for short sequences of input data. Generally speaking, small-sized, fast discrete trigonometric transform algorithms are of particular interest because they are subsequently used as building blocks for larger-sized algorithms. We plan to collect a library of fast short-length algorithms for all types of discrete trigonometric transforms. For some types of trigonometric transformations, such as DCT-I, DCT-II, and DCT-IV (as well as some others), such algorithms have already been developed [32,36,37]. The subject of our research is fast algorithms for small-sized DST-II transforms. The solutions presented here are intended to replenish the collection of fast discrete trigonometric transformation algorithms that many researchers have been working on for several decades. We present here the new algorithms we have obtained, without, however, claiming that they are optimal. This is what we managed to obtain, and we want to share our solutions with the scientific community. If someone manages to achieve better results, we will only be happy about it.

Author Contributions: Conceptualization, A.C.; methodology, A.C., K.B. and M.R.; software, K.B. and M.R.; validation, M.R.; formal analysis, A.C. and K.B.; investigation, K.B. and M.R.; writing—original draft preparation, M.R. and A.C.; writing—review and editing, A.C. and 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: Data are contained within the article.

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

References

2. Britanak, V.; Rao, K.R. Cosine-Sine-Modulated Filter Banks: General Properties, Fast Algorithms and Integer Approximations; Springer International Publishing: Cham, Switzerland, 2018. [CrossRef]
32. Kolenderski, M.; Cariow, A. Small-Size Algorithms for the Type-I Discrete Cosine Transform with Reduced Complexity. Electronics 2022, 11, 2411. [CrossRef]