Visualization of the Structure of DCT Based on the Synthesized Algorithm Using Cyclic Convolutions

Ihor Prots'ko[a], Yurij Bilas[a], Vasyl Teslyuk[a]

[a]*Lviv Polytechnic National University, str. S.Bandery, 12, Lviv, 79013, Ukraine*

Abstract
The visualization of the general structure of discrete cosine transform (DCT) obtained, as a result of the synthesis of the algorithm based on cyclic convolutions is considered. A description of the synthesis of a fast cosine transform (FCT) algorithm has been implemented in the form of a set of operations of cyclic convolutions over combined sequences of input data and coefficients of the basic function of transform. The structure rendering software implementation of FCT based on convolvers is described. The software implementation of the visualization of the structure of FCT is made in the TypeScript language using the Phaser 3 framework. The program for visualization of the structure of FCT for processing the description file of algorithm uses modules: tokenizer, parser, optimizer, visualizer. The results of visualization of the computer structural schemes of the FCT for specific sizes of transform demonstrably and clearly reflect the interaction of its parts, which is important at the system engineering stage of designing the processors DCT based on cyclic convolutions. Data visualization software are used for the purpose of clearly structured display of fast computing algorithms.

Keywords 1
Discrete cosine transform, structure scheme, cyclic convolution, block-cyclic matrix, visualization, convolver.

1. Introduction

Computer visualization uses specific processes of graphic construction, which are determined by the field of applications [1]. In many science and technology areas, the problems numerical solution is not enough for the obtained results implementation further development. Among the existing information visualization approaches, one is chosen that allows for the effective disclosure of unstructured effective ideas, generalization or simplification of the analysis of the received data. Modern possibilities of visual presentation of information allow you to interpret data using procedures for constructing surface graphs, creating data arrays for three-dimensional graphics, turning on and off the scale grid, controlling the properties of graph axes, and others [2]. These tools for displaying the results of algebraic analytics can facilitate understanding of complex concepts and decision-making for specific researchers and developers.

There are many visualization packages and programming languages and software environments for analysis and graphical representation of the obtained calculation results [3]. Some of them are quite simple: you just need to load the data and choose a display method. Other programs are more complex, that require settings and relevant knowledge to display the features of the received solutions.

Scientific developments in efficient computing and modern systems for generating fast algorithms describe them in the form of a sequence of algebraic operations on input data to obtain the final resulting data. Presenting information in such a way that it highlights certain features among others in an analytical method is quite a difficult task.

CITI'2023: 1st International Workshop on Computer Information Technologies in Industry 4.0, June 14–16, 2023, Ternopil, Ukraine
EMAIL: ihor.o.protsko@lpnu.ua (I. Prots'ko); yurii.bilas.mknus.2021@lpnu.ua (Y. Bilas); vasyl.m.teslyuk@lpnu.ua (V. Teslyuk).
ORCID: 0000-0002-3514-9265 (I. Prots'ko); 0009-0005-5313-7494 (Y. Bilas); 0000-0002-5974-9310 (V. Teslyuk).

For this purpose, in the same software environment, where the computational algorithm is implemented, appropriate software tools of data visualization are used for the purpose of clearly structured display of complex information [4]. That is, the results of the synthesis of effective computational algorithms need their interpretation not only in the form of flowcharts, but also in a generalized structured form. For example, this is required at the system engineering stage of designing various information systems.

In addition to the Fourier transform, real discrete cosine or sine transform (DCT or DST) and discrete Hartley transform (DHT) algorithms are used to represent information data in a spectral image. These transforms are used in information technologies for various purposes. Computation of discrete Fourier transforms is one of the most time-consuming procedures in information technology [5]. A number of approaches have been developed that will reduce computational complexity and, accordingly, speed up the work of software and hardware for performing transforms. One approach is to use cyclic convolutions for efficient computation the discrete Fourier transforms [6].

The paper considers the visualization of the general structure of fast cosine transform (FCT) in the domain of real numbers, obtained as a result of the synthesis of the algorithm based on cyclic convolutions. In general, the process of obtaining the FCT algorithm in the form of a set of operations of cyclic convolutions over sequences of input data and coefficients of the basic function DCT is described. The structure rendering software implementation of FCT based on convolvers is described. The results of visualization of variant of the computer structural schemes of the FCT for a fixed size of transform demonstrably and clearly reflect the interaction of its parts and allow to evaluate one or another variant of the FCT algorithm based on cyclic convolutions.

2. Related Works

The variety of efficient algorithms have been developed for computing one-, two-, and multidimensional discrete Fourier transforms. The multivariate of effective computations is divided into algorithms with a base of two, a split base, a mixed base, an odd volume, a compound volume, and an algorithm of simple factors. For the visual display of these fast transformations, which mostly correspond to Cooley-Tukey algorithms, graph flows with a basic operation in the form of a so-called "butterfly" are widely used [7, 8].

However, unlike structural schemes in the form of flow graphs, algorithms for computing discrete Fourier transforms based on cyclic convolutions require other basic operations for their visual representation. Among these main basic operations are the actions of element-wise addition/subtraction of data sequences, the computation of p-point cyclic convolutions, the results of which are element-wise added/subtracted appropriately among themselves. For sizes of transforms of the Fourier class, which can be decomposed into a large number of prime factors, the algorithms contain a large number of these operations. This requires them to be adaptively placed with the desired semantic or visual correlations in the final rendering layout of the structure [9].

Issues of simplicity, flexibility and scalability of visualization of large data structures are investigated in the work [10]. In works [11, 12] it is noted that a better understanding of the concept of data structure and algorithms is to strengthen the basics of object-oriented programming. Understanding the features of the obtained results allows you to highlight and select the necessary visualization objects for a clear and understandable display of elements and variants of computer structural schemes of discrete transforms of Fourier class.

3. Proposed technique the synthesis of fast algorithms of DCT

The ISO/IEC and ITU-T organizations have standardized eight types of DCT I-VIII, which are widely used in the process of processing information data and signals [13]. These trigonometric transforms are a further development of discrete Fourier transforms performed in the real domain. One of the approaches to the effective computation of DCT is to bring the harmonic basis to the block-cyclic matrix structures followed by the computation of transforms using fast cyclic convolutions [14]. In the work [15] describes the reduction of the harmonic basis of the DCT to a set of left-circulant submatrices using generating arrays.

In a result of applying the approach, the structure of the base matrix can be specified by a hashing array

$$H(L) = H_1(L_1)H_2(L_2)...H_k(L_k) = (h_{11}, h_{12},...h_{1L1})(h_{21}, h_{22},...h_{2L2})...(h_{kL1}, h_{kL2},...h_{kLk}). \quad (1)$$

The size of the hashing array L is equal to the sum of the sizes of the subarrays L_i

$$L = (L_1 + L_2 + ... + L_k) \; . \quad (2)$$

where h_{ij} are integer elements of the hashing subarray $Hi(Li)$, which are arguments of the basic harmonic function (i=1,2,...,k; j=1,2,...,Li) and $h_{ij} < T/2$ smaller than the repetition period of the basic harmonic function; and k is the number of subarrays in the hashing array $H(L)$, which is determined by a specific value.

Let us consider the synthesis of a fast discrete transform algorithm for DCT-II, which was first derived on the basis of the discrete Fourier transform [16] and simply called DCT.

Direct DCT-II can be represented by the formula:

$$X^{c2}(n) = a(n)\sum_{m=0}^{N-1}\cos[\frac{n(2m+1)\pi}{2N}]x(m) = \sum_{m=0}^{N-1}c(n,m)x(m), \quad n = 0,1,...,N-1, \quad (3)$$

where $c(n, m) = \cos(n(2m+1)\pi/2N)$.

All rows $c(n,m)$ of vectors $[c(k,0),\dots,c(k,N\text{-}1)]$ are orthogonal and normalized except the first. The coefficient $a(n)$ makes them orthonormal

$$a(n) = \begin{cases} \sqrt{1/N}, & n = 0 \\ \sqrt{2/N}, & n = 1,2,...,N-1 \end{cases} . \quad (4)$$

Table 1 shows the value of periodicity T and the size of the hashing array L depending on N of the size of DCT transform.

Table 1
The value of periodicity of DCT and the size of hashing array

Type of transform	The periodicity, T	The size of hashing array, L
DCT -II	4N	2N

The hashing array $H(L)$ is determined by substituting the rows/columns of the arguments of the harmonic function $c(n,m) = \cos((n(2m+1)\pi)/2N)$ of the base matrix. Based on the hashing array, the rows/columns of the basic matrix are re-indexed, which ultimately leads to the formation of block-cyclic submatrix structures in the basic matrix DCT. As a result of different expressions of column and row indices included in the arguments of the basic function (3), for re-indexing, the hashing array $Hr(n_1)$ of rows and the hashing array $Hc(n_2)$ of columns are used. So, we get matrix of a dimension ($n_1 \times n_2$), which contains a set of integer left-circulant submatrices of different sizes, where the dimensions $n_1 = 2N$, $n_2 = N$ are determined by the size of the hashing arrays for rows and columns. Each of the cyclic square submatrices contains integer elements that belong to one of the hashing subarrays $Hi(Li)$. The sizes Li of each of the hashing subarrays depend on the coefficients of the factorization of the size of the transform N, and in sum meet condition (2).

The property of the asymmetric/symmetric of the basis functions DCT prove efficient representation over less value of elements $e^c_{n,m}$ with the addition of corresponding arrays of signs $z^c_{n,m}$. The arrays of signs consist of the values of elements equal to +1, -1, 0.

$$z^c{}_{nm} = \begin{cases} +1, & if \quad 3N < e^c_{n,m} < N \\ 0, & if \quad e^c_{n,m} = N, 3N \\ -1, & if \quad N < e^c_{n,m} < 3N \end{cases} \quad (5)$$

According to the properties of symmetric/asymmetric of basis function of DCT the simplified matrix consists of the arguments $e^c_{n,m}$, which are determined by the consistent arithmetic operations for DCT

$$e^c_{n,m} = 4N - [e^c_{n,m} \bmod (4N)], \quad if\ (e^c_{n,m} \bmod (4N)) > 2N; \tag{6}$$

and

$$e^c_{n,m} = 2N - \{4N - [e^c_{n,m} \bmod (4N)], \quad if\ (4N - e^c_{n,m} \bmod (4N)) > N; \quad otherwise\, e^c_{n,m} = e^c_{n,m}\,. \tag{7}$$

The simplified matrix of arguments of DCT is complemented the matrices of cosine signs.

A universal software tool for the analysis of integer matrices has been developed for the study of the basic matrices of the DCT, which scans the entire set of elements of the block-cyclic matrix. To search for a given fragment by the maximum width and height *Li*, a step-by-step scan is performed in the matrix with movement from top to bottom and from left to right.

For the synthesis of a fast DCT algorithm, it is necessary to analyze the structure of the obtained block-cyclic matrix in order to determine identical blocks placed horizontally and vertically relative to each other. The presence of identical blocks leads to a decrease in computational complexity and provides the possibility of parallelizing the execution of the DCT computation. In the process of automatic synthesis of algorithms for calculating DCTs of arbitrary size *N* based on cyclic convolutions, the completion of the analysis stage of block-cyclic matrix structures of the basic DCT matrix is the formation of data on the number of identical cyclic subarrays and their location in the basic matrix.

Identical cyclic submatrices placed vertically relative to each other lead to a one-time computation of cyclic convolutions, the results of which in the process of unification are used to determine different *Xc*(*m*) initial values of the transform.

Identical cyclic submatrices placed horizontally relative to each other lead to the unification of groups of input values *x*(*n*) of the transform and a one-time computation of cyclic convolutions. The results of cyclic convolutions in the merging process are used for one group Xc(*m*) of the initial transform values. Performing of element-wise addition of input values will be used for cyclic submatrices of the same type placed horizontally.

The output values *Xc*(*m*) of the DCT transform are obtained by combining the results of convolutions horizontally based on the corresponding coordinates.

An example of the synthesis of the fast DCT algorithm for the size *N*=27, the hashing array $Hr(n_1)$ of rows and the creative array $Hc(n_2)$ of columns contain the following elements:
Hashing Array(col) for-row sym 2*N*

Hr(54)=(0) (1 5 25 17 23 7 35 41 11 53 49 29 37 31 47 19 13 43) (2 10 50 34 46 14 38 26 22) (52 44 4 20 8 40 16 28 32) (3 15 33 51 39 21) (6 30 42) (48 24 12) (9 45) (18) (36) (27);

Hashing Array(col) for-row sym *N*

Hc(27)= (0) (1 5 25 17 23 7 19 13 11) (26 22 2 10 4 20 8 14 16) (3 15 21) (24 12 6) (9) (18).

The result of the formation of block-cyclic matrix structures of arguments in the basic matrix of the DCT is presented in Figure 1.

In a result of the analysis of the block-cyclic structure of the base matrix, we obtain data on the number of identical cyclic subarrays and their location coordinates in the base matrix (Figure 1), where the value of the first element of the cyclic submatrices is highlighted in bold.

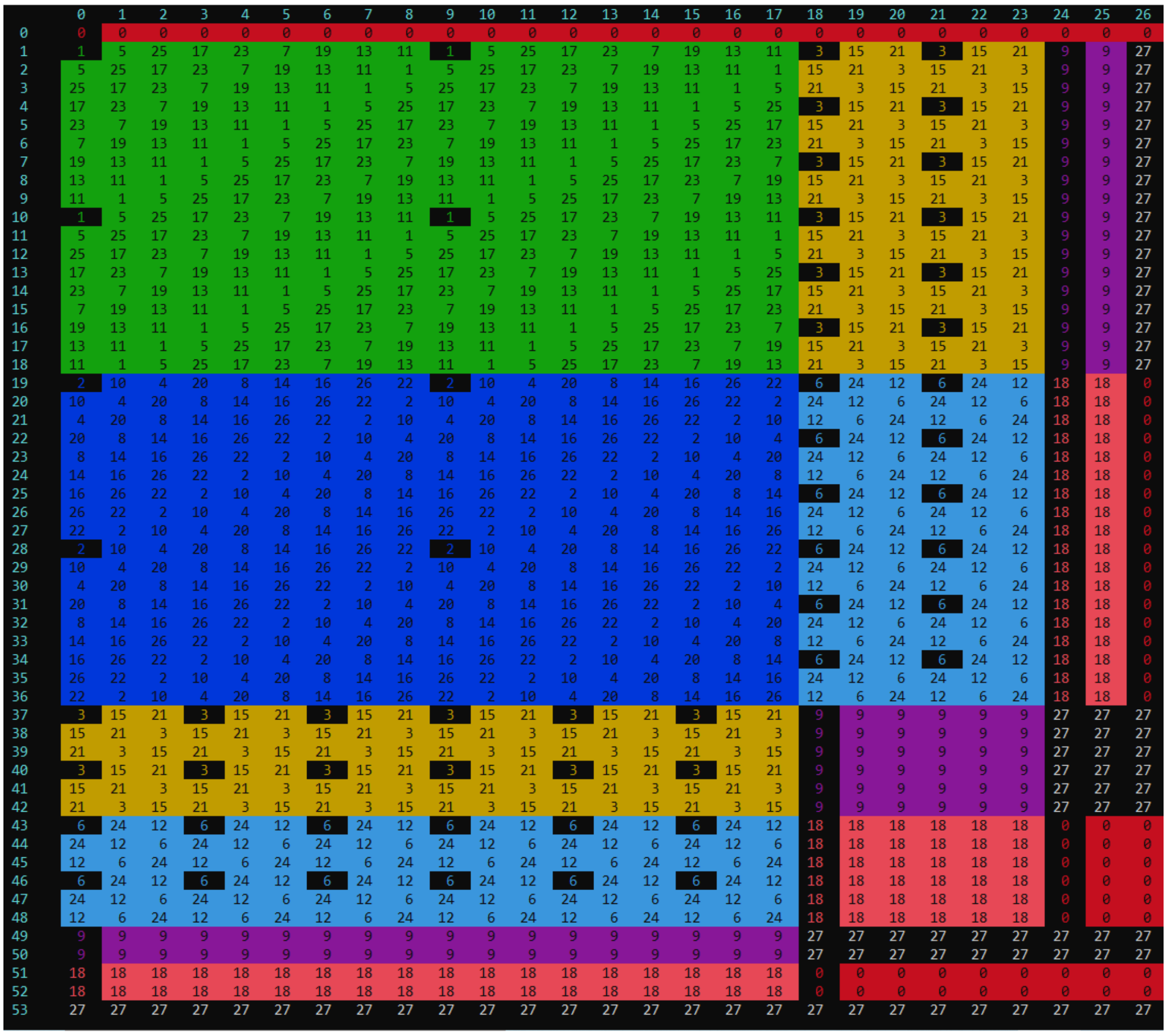
0: { 0, 0 }, { 51, 18 }, { 52, 18 }, { 43, 24 }, { 44, 24 }, { 45, 24 }, { 46, 24 }, { 47, 24 }, { 48, 24 }, {19, 26}, { 20, 26 }, { 21, 26 }, { 22, 26 }, { 23, 26 }, { 24, 26 }, { 25, 26 }, { 26, 26 }, { 27, 26 }, { 28, 26 }, { 29, 26 }, { 30, 26 }, { 31, 26 }, { 32, 26 }, { 33, 26 }, { 34, 26 }, { 35, 26 }, { 36, 26 },

1: { 1, 0 }, { 10, 0 }, { 1, 9 }, { 10, 9 },

2: { 19, 0 }, { 28, 0 }, { 19, 9 }, { 28, 9 },

3: { 37, 0 }, { 40, 0 }, { 37, 3 }, { 40, 3 }, { 37, 6 }, { 40, 6 }, { 37, 9 }, { 40, 9 }, { 37, 12 }, { 40, 12 }, { 37, 15 }, { 40, 15 }, { 1, 18 }, { 4, 18 }, { 7, 18 }, { 10, 18 }, { 13, 18 }, { 16, 18 }, { 1, 21 }, { 4, 21 }, { 7, 21 }, { 10, 21 }, { 13, 21 }, { 16, 21 },

6: { 43, 0 }, { 46, 0 }, { 43, 3 }, { 46, 3 }, { 43, 6 }, { 46, 6 }, { 43, 9 }, { 46, 9 }, { 43, 12 }, { 46, 12 }, {43, 15 }, { 46, 15 }, { 19, 18 }, { 22, 18 }, { 25, 18 }, { 28, 18 }, { 31, 18 }, { 34, 18 }, { 19, 21 }, { 22, 21 }, {25, 21 }, { 28, 21 }, { 31, 21 }, { 34, 21 },



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	5	25	17	23	7	19	13	11	1	5	25	17	23	7	19	13	11	3	15	21	3	15	21	9	9	27
2	5	25	17	23	7	19	13	11	1	5	25	17	23	7	19	13	11	1	15	21	3	15	21	3	9	9	27
3	25	17	23	7	19	13	11	1	5	25	17	23	7	19	13	11	1	5	21	3	15	21	3	15	9	9	27
4	17	23	7	19	13	11	1	5	25	17	23	7	19	13	11	1	5	25	3	15	21	3	15	21	9	9	27
5	23	7	19	13	11	1	5	25	17	23	7	19	13	11	1	5	25	17	15	21	3	15	21	3	9	9	27
6	7	19	13	11	1	5	25	17	23	7	19	13	11	1	5	25	17	23	21	3	15	21	3	15	9	9	27
7	19	13	11	1	5	25	17	23	7	19	13	11	1	5	25	17	23	7	3	15	21	3	15	21	9	9	27
8	13	11	1	5	25	17	23	7	19	13	11	1	5	25	17	23	7	19	15	21	3	15	21	3	9	9	27
9	11	1	5	25	17	23	7	19	13	11	1	5	25	17	23	7	19	13	21	3	15	21	3	15	9	9	27
10	1	5	25	17	23	7	19	13	11	1	5	25	17	23	7	19	13	11	3	15	21	3	15	21	9	9	27
11	5	25	17	23	7	19	13	11	1	5	25	17	23	7	19	13	11	1	15	21	3	15	21	3	9	9	27
12	25	17	23	7	19	13	11	1	5	25	17	23	7	19	13	11	1	5	21	3	15	21	3	15	9	9	27
13	17	23	7	19	13	11	1	5	25	17	23	7	19	13	11	1	5	25	3	15	21	3	15	21	9	9	27
14	23	7	19	13	11	1	5	25	17	23	7	19	13	11	1	5	25	17	15	21	3	15	21	3	9	9	27
15	7	19	13	11	1	5	25	17	23	7	19	13	11	1	5	25	17	23	21	3	15	21	3	15	9	9	27
16	19	13	11	1	5	25	17	23	7	19	13	11	1	5	25	17	23	7	3	15	21	3	15	21	9	9	27
17	13	11	1	5	25	17	23	7	19	13	11	1	5	25	17	23	7	19	15	21	3	15	21	3	9	9	27
18	11	1	5	25	17	23	7	19	13	11	1	5	25	17	23	7	19	13	21	3	15	21	3	15	9	9	27
19	2	10	4	20	8	14	16	26	22	2	10	4	20	8	14	16	26	22	6	24	12	6	24	12	18	18	0
20	10	4	20	8	14	16	26	22	2	10	4	20	8	14	16	26	22	2	24	12	6	24	12	6	18	18	0
21	4	20	8	14	16	26	22	2	10	4	20	8	14	16	26	22	2	10	12	6	24	12	6	24	18	18	0
22	20	8	14	16	26	22	2	10	4	20	8	14	16	26	22	2	10	4	6	24	12	6	24	12	18	18	0
23	8	14	16	26	22	2	10	4	20	8	14	16	26	22	2	10	4	20	24	12	6	24	12	6	18	18	0
24	14	16	26	22	2	10	4	20	8	14	16	26	22	2	10	4	20	8	12	6	24	12	6	24	18	18	0
25	16	26	22	2	10	4	20	8	14	16	26	22	2	10	4	20	8	14	6	24	12	6	24	12	18	18	0
26	26	22	2	10	4	20	8	14	16	26	22	2	10	4	20	8	14	16	24	12	6	24	12	6	18	18	0
27	22	2	10	4	20	8	14	16	26	22	2	10	4	20	8	14	16	26	12	6	24	12	6	24	18	18	0
28	2	10	4	20	8	14	16	26	22	2	10	4	20	8	14	16	26	22	6	24	12	6	24	12	18	18	0
29	10	4	20	8	14	16	26	22	2	10	4	20	8	14	16	26	22	2	24	12	6	24	12	6	18	18	0
30	4	20	8	14	16	26	22	2	10	4	20	8	14	16	26	22	2	10	12	6	24	12	6	24	18	18	0
31	20	8	14	16	26	22	2	10	4	20	8	14	16	26	22	2	10	4	6	24	12	6	24	12	18	18	0
32	8	14	16	26	22	2	10	4	20	8	14	16	26	22	2	10	4	20	24	12	6	24	12	6	18	18	0
33	14	16	26	22	2	10	4	20	8	14	16	26	22	2	10	4	20	8	12	6	24	12	6	24	18	18	0
34	16	26	22	2	10	4	20	8	14	16	26	22	2	10	4	20	8	14	6	24	12	6	24	12	18	18	0
35	26	22	2	10	4	20	8	14	16	26	22	2	10	4	20	8	14	16	24	12	6	24	12	6	18	18	0
36	22	2	10	4	20	8	14	16	26	22	2	10	4	20	8	14	16	26	12	6	24	12	6	24	18	18	0
37	3	15	21	3	15	21	3	15	21	3	15	21	3	15	21	3	15	21	9	9	9	9	9	9	27	27	27
38	15	21	3	15	21	3	15	21	3	15	21	3	15	21	3	15	21	3	9	9	9	9	9	9	27	27	27
39	21	3	15	21	3	15	21	3	15	21	3	15	21	3	15	21	3	15	9	9	9	9	9	9	27	27	27
40	3	15	21	3	15	21	3	15	21	3	15	21	3	15	21	3	15	21	9	9	9	9	9	9	27	27	27
41	15	21	3	15	21	3	15	21	3	15	21	3	15	21	3	15	21	3	9	9	9	9	9	9	27	27	27
42	21	3	15	21	3	15	21	3	15	21	3	15	21	3	15	21	3	15	9	9	9	9	9	9	27	27	27
43	6	24	12	6	24	12	6	24	12	6	24	12	6	24	12	6	24	12	18	18	18	18	18	18	0	0	0
44	24	12	6	24	12	6	24	12	6	24	12	6	24	12	6	24	12	6	18	18	18	18	18	18	0	0	0
45	12	6	24	12	6	24	12	6	24	12	6	24	12	6	24	12	6	24	18	18	18	18	18	18	0	0	0
46	6	24	12	6	24	12	6	24	12	6	24	12	6	24	12	6	24	12	18	18	18	18	18	18	0	0	0
47	24	12	6	24	12	6	24	12	6	24	12	6	24	12	6	24	12	6	18	18	18	18	18	18	0	0	0
48	12	6	24	12	6	24	12	6	24	12	6	24	12	6	24	12	6	24	18	18	18	18	18	18	0	0	0
49	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	27	27	27	27	27	27	27	27	27
50	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	27	27	27	27	27	27	27	27	27
51	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	0	0	0	0	0	0	0	0	0
52	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	0	0	0	0	0	0	0	0	0
53	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27

Figure 1: The block-cyclic structure of the basic matrix of DCT for the size *N*=27

9: { 49, 0 }, { 50, 0 }, { 37, 18 }, { 38, 18 }, { 39, 18 }, { 40, 18 }, { 41, 18 }, { 42, 18 }, { 1, 24 }, { 2, 24 }, { 3, 24 }, { 4, 24 }, { 5, 24 }, { 6, 24 }, { 7, 24 }, { 8, 24 }, { 9, 24 }, { 10, 24 }, { 11, 24 }, { 12, 24 }, { 13, 24 }, { 14, 24 }, { 15, 24 }, { 16, 24 }, { 17, 24 }, { 18, 24 },

18: { 51, 0 }, { 52, 0 }, { 43, 18 }, { 44, 18 }, { 45, 18 }, { 46, 18 }, { 47, 18 }, { 48, 18 }, { 19, 24 }, { 20, 24 }, { 21, 24 }, { 22, 24 }, { 23, 24 }, { 24, 24 }, { 25, 24 }, { 26, 24 }, { 27, 24 }, { 28, 24 }, { 29, 24 }, { 30, 24 }, { 31, 24 }, { 32, 24 }, { 33, 24 }, { 34, 24 }, { 35, 24 }, { 36, 24 }

These data make it possible to determine the minimum number and sizes of cyclic convolutions, the possibility of their parallel computation and combining the calculated values of cyclic convolutions to determine the output values of DCT.

4. Software implementation of the visualization of the structure of FCT

The TypeScript programming language (which is developed by Microsoft) was chosen for the software implementation of visualization of the structure of FCT. The TypeScript language extends its predecessor JavaScript by providing backward compatibility [17]. The TypeScript language is statically typed, which will make it possible to detect errors at the stage of writing code or compilation. To work with TypeScript, Microsoft's development environments are most suitable: Visual Studio and Visual Studio Code. The Visual Studio environment is a more "heavy" solution, because it works slower, but also has more functionality. The Visual Studio Code environment was created specifically for working with web applications and is extremely simple, extensible with a large number of plugins.

The popular and widespread Git system is used to work with program versions. The package manager for JavaScript npm will be used due to its widespread use and simplicity. Webpack is used for packaging modules, as a standard in the modern web application industry. Using npm and Webpack requires using Node.js. The "file-saver" library will be used to work with files (reading and writing). A set of libraries will be used to develop unit tests (unit tests): chai, mocha, sinon.

The most popular tools for providing visualization are: Pixi.js; Phaser 2; Phaser 3. The first tool is very fast, but has only basic features, so development will be a bit slower than the other options. Since execution speed is not the most important attribute of the quality of the visualization system being developed, we will consider other options. Phaser 2 is a popular solution that is feature-rich and easy to work with. However, this framework is somewhat outdated and not supported by developers. Phaser 3 is a framework that replaced its predecessor, Phaser 2, and is a complete rethinking and reworking of it. This solution has become a new standard in the industry and is actively developing. Therefore, Phaser 3 was chosen as the visualization tool [18].

Interaction with the system is carried out through the user interface, in which settings are made to display the structural scheme of the FCT. As a result, we get a display of the system on the screen, as well as the ability to download images into files that will contain the structural scheme of the FCT or part of it.

The initial data for the operation of the visualization system is a text file describing the convolutions in the form, as an example, for the FCT for size *N*=27,

1. a) (+0) (X) { +x(0) +x(2) +x(12) +x(8) +x(11) +x(3) +x(17) +x(20) +x(5) +x(26) +x(24) +x(14) +x(18) +x(15) +x(23) +x(9) +x(6) +x(21) +x(1) +x(7) +x(16) +x(25) +x(19) +x(10) +x(4) +x(22) +x(13) }

2. a) (+1 +5 +25 +17 +23 +7 -19 -13 +11) (X) { +(x(0), x(2), x(12), x(8), x(11), x(3), x(17), x(20), x(5)) -(x(26), x(24), x(14), x(18), x(15), x(23), x(9), x(6), x(21)) }

   b) (+3 +15 -21) (X) { +(x(1), x(7), x(16)) -(x(25), x(19), x(10)) }

   c) (+9) (X) { +x(4) -x(22) }

3. a) (+2 +10 -4 -20 -8 +14 -16 +26 +22) (X) { +(x(0), x(2), x(12), x(8), x(11), x(3), x(17), x(20), x(5)) +(x(26), x(24), x(14), x(18), x(15), x(23), x(9), x(6), x(21)) }

   b) (+6 -24 -12) (X) { +(x(1), x(7), x(16)) +(x(25), x(19), x(10)) }

   c) (+18) (X) { +x(4) +x(22) }

   d) (-0) (X) { -x(13) }

4. a) (+3 +15 -21) (X) { +(x(0), x(2), x(12)) -(x(8), x(11), x(3)) +(x(17), x(20), x(5)) -(x(26), x(24), x(14)) +(x(18), x(15), x(23)) -(x(9), x(6), x(21)) }

   b) (+9) (X) { +x(1) -x(7) +x(16) -x(25) +x(19) -x(10) }

5. a) (+6 -24 -12) (X) { +(x(0), x(2), x(12)) +(x(8), x(11), x(3)) +(x(17), x(20), x(5)) +(x(26), x(24), x(14)) +(x(18), x(15), x(23)) +(x(9), x(6), x(21)) }

   b) (+18) (X) { +x(1) +x(7) +x(16) +x(25) +x(19) +x(10) }

   c) (-0) (X) { -x(4) -x(22) -x(13) }

6. a) (+9) (X) { +x(0) -x(2) +x(12) -x(8) +x(11) -x(3) +x(17) -x(20) +x(5) -x(26) +x(24) -x(14) +x(18) -x(15) +x(23) -x(9) +x(6) -x(21) }

7. a) (+18) (X) { +x(0) +x(2) +x(12) +x(8) +x(11) +x(3) +x(17) +x(20) +x(5) +x(26) +x(24) +x(14) +x(18) +x(15) +x(23) +x(9) +x(6) +x(21) }

   b) (-0) (X) { -x(1) -x(7) -x(16) -x(25) -x(19) -x(10) -x(4) -x(22) -x(13) } where (X) means performing a cyclic convolution over sequences of cosine functions with arguments given by ( ) and sequences { } of transform input data x(i), i=0,1,...,N-1, which must be previously element-wise combined.

Next, it is worth considering and explaining what data structures represent cyclic convolutions in the program. In Figure 2 are showed the demonstration, as an example, for what parts the components of input data is divided into

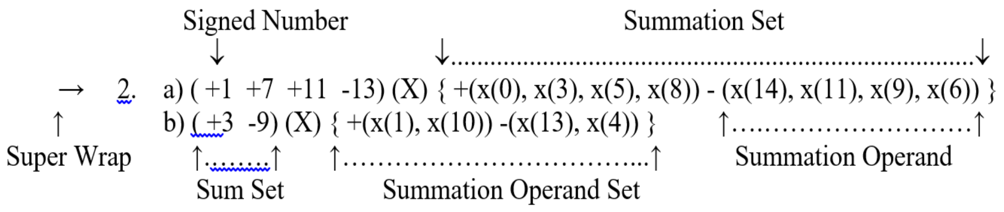

Signed Number
Summation Set
2. a) (+1 +7 +11 -13) (X) { +(x(0), x(3), x(5), x(8)) - (x(14), x(11), x(9), x(6)) }
b) (+3 -9) (X) { +(x(1), x(10)) -(x(13), x(4)) }
Super Wrap
Sum Set
Summation Operand Set
Summation Operand

Figure 2: The parts of the components of input data of text file describing the convolutions

The main block (Fig. 2.) is the Super Wrap element, which is responsible for one wrap. It consists of one or more Wrap elements that represent the branches of the wrap. Each Wrap consists of components: Sum Set, Summation Set. The Sum Set consists of a set of Signed Number elements. A Summation Set consists of a set of type Summation OperandSet, characterized by a sign ("+" or "-"), and a set of Summation Operand elements.

The program for visualization of the structure of FCT for processing the description text file of algorithm uses modules: tokenizer, parser, optimizer, visualizer.

The tokenizer processes "raw" character data from the description text file and breaks it into a sequence of tokens. In the algorithm of the general logic of the tokenizer module, repeated actions are performed until the end of the data is reached. Among the actions performed is the reading of a character and, depending on its type, the corresponding sub-procedure for reading the token is performed (tokens can consist of several characters). In addition, a check is performed to see if the read character is supported. The received set of tokens is sent for parsing.

Before the parsing of a separate convolution, the input set of tokens is divided into groups that form a certain hierarchy. The need for this algorithm is caused by the fact that the format of the input data assumes a hierarchy, where the first level contains numerical indexes ("1.", "2.", etc.), and the second - symbolic ones ("a)", "b)", etc.). This algorithm first divides tokens into numerically indexed groups, and then divides each group into character-indexed subgroups. Then these groups and subgroups are used in the parser module.

The parser receives the tokens and converts them into a syntax tree. That is, each line is read from the text file describing the convolutions in the parsing algorithm and the appropriate grammatical rules are used to analyze only one convolution. Next, the first convolution sequence (separator X) and second sequence is read. Next, it is checked whether the data remains. If so, then a syntactic tree is formed, otherwise, we get an error.

The next module is the optimizer, whose task is, after examining the structure of the obtained tree, to make certain transforms that would make the tree more compact and convenient for visualization. At that moment, the optimization is implemented only as a base and will be refined in the next versions of the software system.

The visualizer module provides imagery of the structural scheme of the FCT. For this, a simple linear sequence of steps is performed:

– the calculation of parameters that will help to perform visualization is performed (it is performed on the basis of a syntactic tree);
– the rectangles and signatures related to them are displayed;
– various arrows and their captions are displayed.

Thus, the optimized syntax tree, as well as the settings received from the user interface, are fed into the renderer. It creates a visual image of the structure of FCT.

In Figure 3, we see an example of how the display of the structural scheme of the FCT for size N=27. The structure consists of a set of arrows (input, intermediate, output), their labels (such as x'_{6a}), and rectangles (CC and U). CC are blocks of p-point convolutions, U are blocks of element-wise combination of input data x(i), i=0,1,...,N-1of the transform.

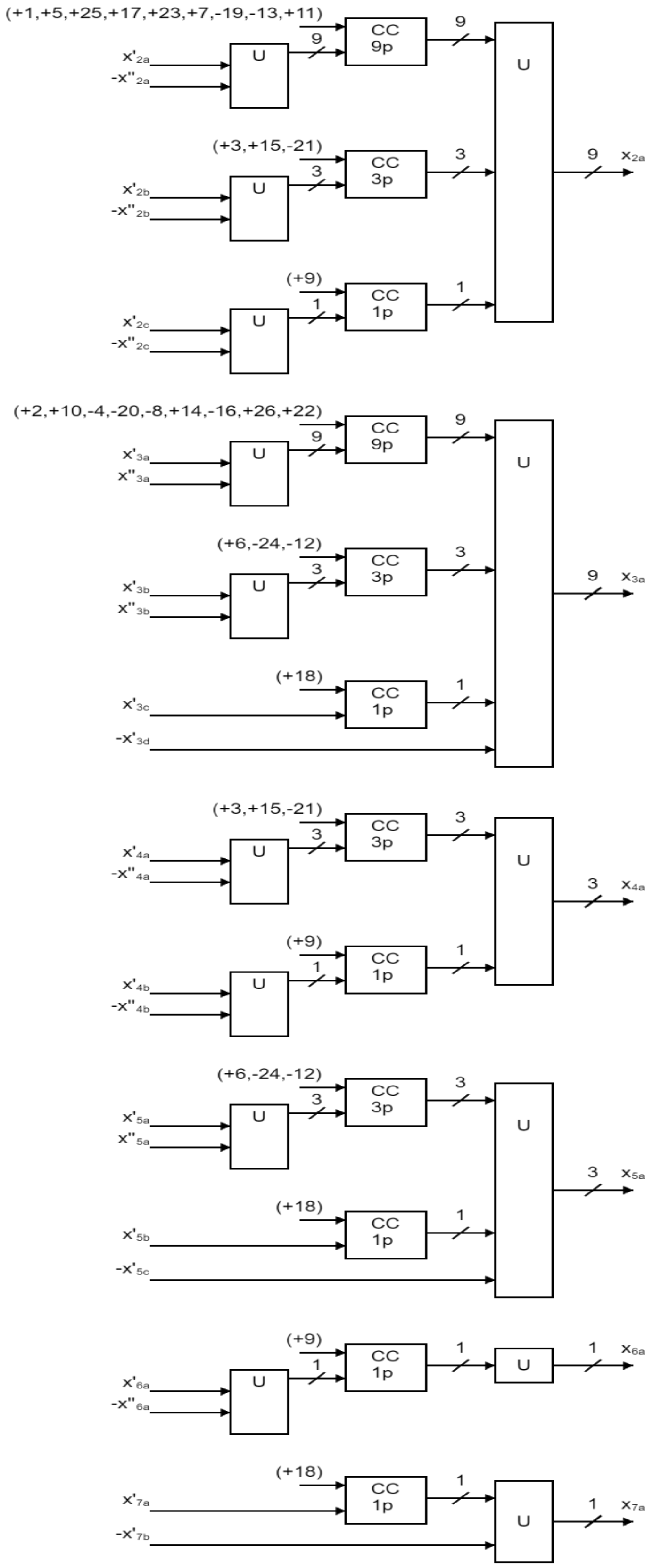


Figure 3: The visual representation of the structure of FCT for size N=27

In addition, you can also see the module that performs the generation of the "legend" (Figure 4), that is, the textual explanation for the markings on the diagram.

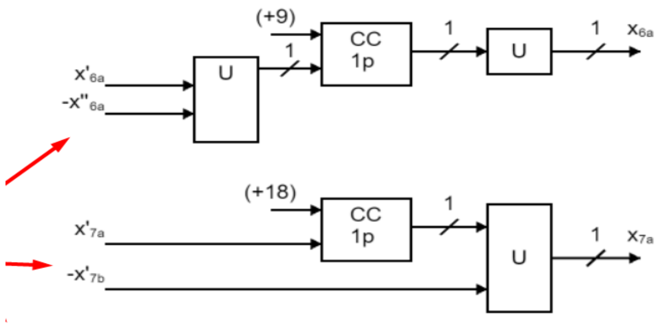


Figure 4: Components of display zones

$x'_{2a}=x'_{3a}$=(x(0), x(2), x(12), x(8), x(11), x(3), x(17), x(20), x(5))
$x''_{2a}=x''_{3a}$=(x(26), x(24), x(14), x(18), x(15), x(23), x(9), x(6), x(21))
$x'_{2b}=x'_{3b}$=(x(1), x(7), x(16))
$x''_{2b}=x''_{3b}$=(x(25), x(19), x(10))
x'_{2c}=x(4)
x''_{2c}=x(22)

x'_{3c}=(x(4), x(22))
x'_{3d}=x(13)

$x'_{4a}=x'_{5a}$=(x(0), x(2), x(12), x(17), x(20), x(5), x(18), x(15), x(23))
$x''_{4a}=x''_{5a}$=(x(8), x(11), x(3), x(26), x(24), x(14), x(9), x(6), x(21))
x'_{4b}=(x(1), x(16), x(19))
x''_{4b}=(x(7), x(25), x(10))

x'_{5b}=(x(1), x(7), x(16), x(25), x(19), x(10))
x'_{5c}=(x(4), x(22), x(13))

x'_{6a}=(x(0), x(12), x(11), x(17), x(5), x(24), x(18), x(23), x(6))
x''_{6a}=(x(2), x(8), x(3), x(20), x(26), x(14), x(15), x(9), x(21))

x'_{7a}=(x(0), x(2), x(12), x(8), x(11), x(3), x(17), x(20), x(5), x(26), x(24), x(14), x(18), x(15), x(23), x(9), x(6), x(21))
x'_{7b}=(x(1), x(7), x(16), x(25), x(19), x(10), x(4), x(22), x(13))

The outputs of the structure of FCT for size *N*=27 (Fig. 3) include such coefficients of discrete cosine transform (3)

$X_{2a} = X^{c2}(1), X^{c2}(5), X^{c2}(25), X^{c2}(17), X^{c2}(23), X^{c2}(7), X^{c2}(19), X^{c2}(13), X^{c2}(11);$

$X_{3a} = X^{c2}(2), X^{c2}(10), X^{c2}(4), X^{c2}(20), X^{c2}(8), X^{c2}(14), X^{c2}(16), X^{c2}(26), X^{c2}(22);$

$X_{4a} = X^{c2}(3), X^{c2}(15), X^{c2}(21);$

$X_{5a} = X^{c2}(6), X^{c2}(24), X^{c2}(12);$

$X_{6a} = X^{c2}(9);$

$X_{7a} = X^{c2}(18);$

The "legend" generation module, like the visualizer, uses an optimized syntax tree, and transmits the results of its operation to the user interface input.

This image is transferred to the user interface, which in turn, as already mentioned, displays everything on the screen for the user, and gives the opportunity to save the received image to a file also.

In Figure 5 shows the UML class diagram of the system for visual display of the structural scheme of the FCT. The UML class diagram forms an idea of what the static structure of the system's software is.

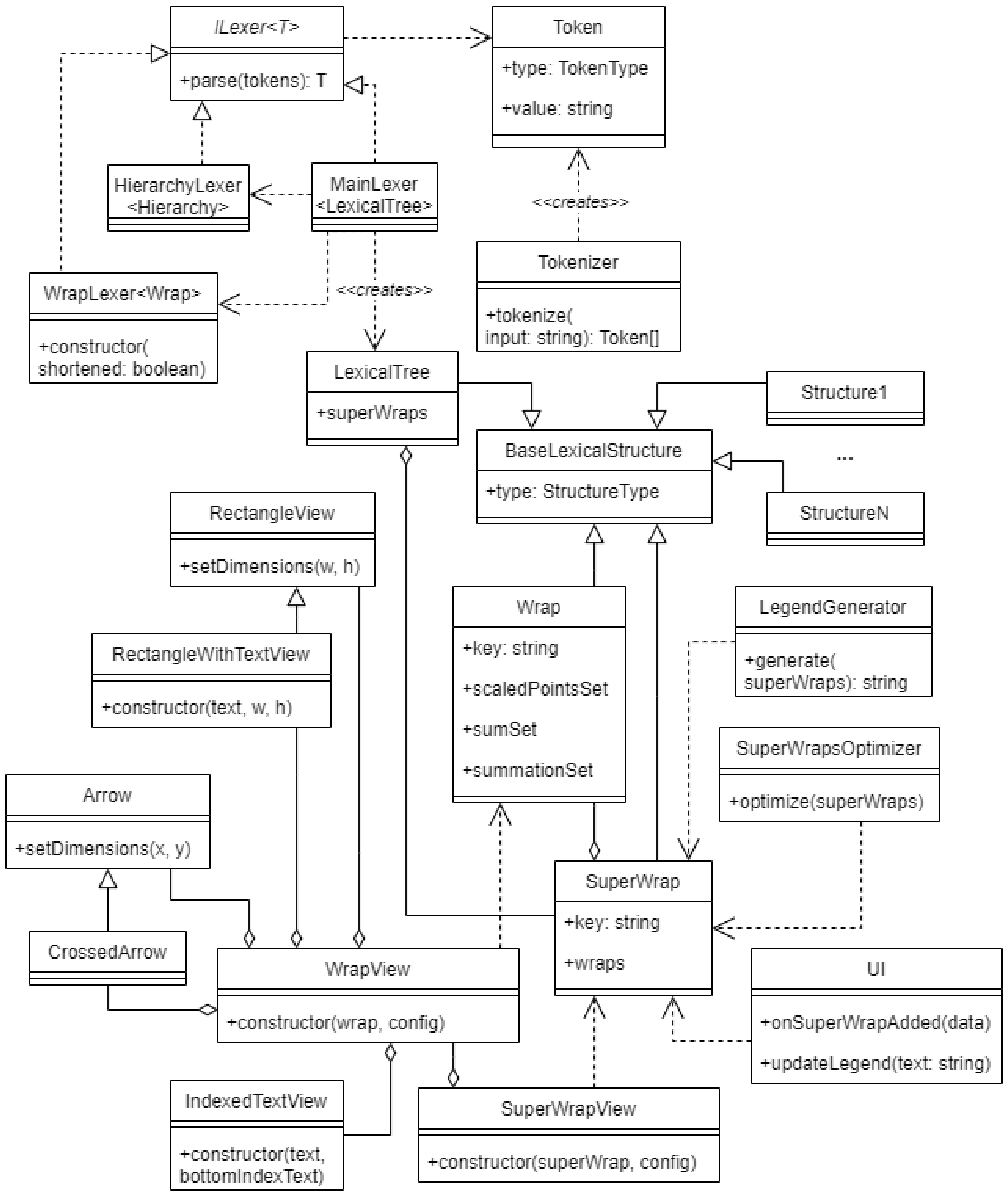


Figure 5: Class diagram of the software system of the structure of FCT

It is also worth noting that only the main key classes, methods and fields are displayed in this diagram. In reality, the system will contain more small objects, including enumerations, interfaces, some trivial helper classes for parsing, configurations, utility classes, and so on.

5. Discussions

The program for the synthesis of algorithms for the efficient computation of DCT based on cyclic convolutions includes a developed visualization subsystem. The result of synthesis is received textual description of the algorithm FCT, what has been rendered into a structure of FCT based on convolvers. This will make it possible to quickly estimate, without detailed analysis, the number of convolvers, their relationships, the scope of performing cyclic convolutions, the sequence of combining input data at the system-technical stage of designing a FCT processor. The implementation of FCT processors in the form of microcircuits is characterized the reduction hardware cost [19, 20].

The developed visualization program uses web technologies, thereby achieving ease of development, support, and cross-platform compatibility. The leading technology of Phaser 3 is used for easy implementation of visualization of vector primitives. In order for the system not to have such a harmful property inherent in some software solutions as "strong connectivity", it is clearly divided into two parts:

- core, that is, the core of the system, which is responsible for the logic of the software application; this includes, for example:
  - tokenization;
  - syntactic analysis;
  - optimization;
- view, i.e. that part of the program that deals with displaying anything on the screen; this includes, for example:
  - graphical user interface;
  - visualization module for convolutions;
  - the generator of the "legend".

Since the system is saturated with algorithms, it is easy to make a mistake, which later, when the development reaches a large scale, it will be difficult to "catch" and correct, so an important decision was made - during the development of the system, unit tests should also be developed. In total, 12 tests have been written that check the correctness of 5 classes, which relate to the most important part of the system's logic - tokenization and parsing. In the course of development, it was repeatedly noticed that the results of the tests indicated the presence of errors in the execution of the program, and thus all problems that arose due to accidentally made errors in the code were easily solved.

The system has the minimum necessary set of functions, so it can be expanded, in particular by improving the optimization of the lexical tree. In the future, it is possible to make the visualized scheme more interactive and more flexible for customization.

6. Conclusion

The results of the synthesis of effective computing algorithms require their interpretation not only in the form of charts, but also in a generalized structured form. The paper considers the visualization of the general structure of fast cosine transform, obtained, as a result of the synthesis of the algorithm based on cyclic convolutions. On the basis of the created description of the algorithm, which includes a multi-level set of cyclic convolutions, a program for visualization of the structure of discrete cosine transform was developed. The rendering program is implemented in the TypeScript programming language using the Phaser 3 framework for rendering vector primitives. The program for visualization of the structure of FCT for processing the description file of the algorithm uses modules: tokenizer, parser, optimizer, visualizer. The practical significance of the work lies in the fact that the obtained results of visualization of the computer structural schemes of the FCT for specific sizes of transform is important for the system engineering stage of designing a FCT based on cyclic convolutions, because clearly reflect the general interaction of its parts on algorithmic level.

7. References

[1] Data Science, 2020. URL: https://coggle.it/diagram/XMltPKW0VBuSKAMC/t/data-science

[2] M. M. Gomes, 2021. Data Visualization: Best Practices and Foundations, URL: https://www.toptal.com/designers/data-visualization/data-visualization-best-practices

[3] 36 best data visualization tools, 2021. URL: https://toplead.com.ua/ua/blog/id/38-luchshih-instrumentov-dlja-vizualizacii-dannyh-160/

[4] F. Zhang, W. Liu, N. Feng, J. Zhai, X. Du, Performance evaluation and analysis of sparse matrix and graph kernels on heterogeneous processors. CCF Transactions on High Performance Computing 1 (2019) 131-143.

[5] A. V. Oppenheimer, R. W. Schafer, Discrete-Time Signal Processing. Third ed., Englewood Cliffs, NJ: Prentice Hall. Pearson Education Limited, 2014, 1042p.

[6] R. E. Blahut, Fast algorithms for signal processing. Cambridge: University Press, 2010, -469 p. doi:10.1017/CBO9780511760921

[7] L. Stankovic, Digital signal processing. Basic Theory and Applications. Revised edition, North Charleston, South Carolina, USA, 2020, 591p.

[8] I. Tsmots, V. Rabyk, N. Kryvinska, M. Yatsymirskyy, V. Teslyuk, Design of the Processors for Fast Cosine and Sine Fourier Transforms. Circuits, Systems, and Signal Processing, 41.9 (2022) 4928–4951. doi:10.1007/s00034-022-02012-8

[9] J. Garriga, F. Bartumeus, Towards a comprehensive visualization of structure in data, 2021. URL: https://arxiv.org/abs/2111.15506?context=cs

[10] X. Han, C. Zhang, W. Lin, M. Xu, B. Sheng, and T. Mei, Tree-based Visualization and Optimization for Image Collection. IEEE Transactions on Cybernetics, 46.6 (2016) 1-14.

[11] S. Simonak, Increasing the Engagement Level in Algorithms and Data Structures Course by Driving Algorithm Visualizations. Informatica, 44.3(2020). doi: 10.31449/inf.v44i3.2864

[12] S. Ghadge, V. Mane, A Survey paper on data structure and algorithm vizualization. International Research Journal of Modernization in Engineering Technology and Science 4.4 (2022) 232-236.

[13] ISO/IEC 14496-2:2004. Information technology – Coding of audio-visual objects – Part 2: Visual, ISO, 2004.

[14] I. Prots'ko, M. Mishchuk, Block-Cyclic Structuring of the Basis of Fourier Transforms Based on Cyclic Substitution. Cybernetics and Systems Analysis, 57.6 (2021) 1008–1016. doi: 10.1007/s10559-021-00426-x

[15] I. Prots'ko, Algorithm of Efficient Computation of DCT I-IV Using Cyclic Convolutions, Int. J. Circuits, System and Signal Processing, 7.1 (2013) 1–9.

[16] I. Prots'ko, V. Teslyuk, Relationship of Fast Computing DCT-II and DST-II Based on Cyclic Convolutions, International Journal of Condition Monitoring and Diagnostic Engineering Management (International Journal of COMADEM), 25.3 (2022). 9–19.

[17] TypeScript, 2022. URL: https://www.typescriptlang.org/

[18] Phaser 3, 2022. URL: https://phaser.io/phaser3

[19] D. F. Chiper, A. Cracan and V.-D. Andries, An Overview of Systolic Arrays for Forward and Inverse Discrete Sine Transforms and Their Exploitation in View of an Improved Approach. Electronics, 11.15, (2022) 1-22. doi: 10.3390/electronics11152416

[20] M. Ulicny, V. A. Krylov, R. Dahyot, Harmonic convolutional networks based on discrete cosine transform, Pattern Recognition, 129 (2022) 1-12. doi:10.1016/j.patcog.2022.108707