High-accuracy implementation of fast DCT algorithms based on algebraic integer encoding

Maxim Vashkevich
Computer Engineering Department
Belarusian State University
of Informatics and Radioelectronics
Minsk, Belarus 220013
Email: vashkevich@bsuir.by

Marek Parfieniuk
Department of Digital Media
and Computer Graphics
Bialystok University of Technology
Bialystok, Poland 15-351
Email: m.parfieniuk@pb.edu.pl

Alexander Petrovsky
Department of Digital Media
and Computer Graphics
Bialystok University of Technology
Bialystok, Poland 15-351
Email: palex@bsuir.by

***Abstract*— The paper presents a systematic approach to synthesis and implementation of fast algorithms for computing the DCT of a power-of-two size. The main features of the obtainable algorithms are regularity of their signal flow graphs and low arithmetic complexity. Multiplierless implementation of the algorithms is based on the algebraic integer (AI) technique. A general AI encoding scheme for fast DCT algorithms is presented. The approach is demonstrated by using it to derive an AI-based low-error fast implementation of the 16-point DCT.**

I. Introduction

The Discrete Cosine Transform of type 2 (DCT-2) is the key operation used by the majority of video and image compression algorithms. The reasons of its popularity are its property of energy compaction and its asymptotic equivalence to the Karhunen-Loéve transform for signals produced by the first-order Gaussian Markov process.

Although the 8-point DCT-2 is the kernel of the H.261, JPEG, and MPEG-2 standards for image/video compression, more recent solutions such as the VC-1, AVS and HEVC use a number of transforms of sizes ranging from 4 to 64. It is possible that even larger transforms will be considered in the future since video resolutions keep increasing [1].

Many different fast DCT algorithms that use reduced numbers of multiplications have been developed in the last decades [2], [3]. The majority of them have been found by insightful manipulations of the entries of the transform matrix using algebraic relationships. Recently, an approach to derivation of fast DCT algorithms based on the notion of polynomial algebra has been presented in [4]. Now this approach is referred to as the *algebraic signal processing theory* (ASP) [5].

Multiplication by nontrivial factors is another problem to be solved in practical implementation of DCT algorithms. It is an important issue since floating-point multiplication is rather slow in hardware/software realizations. The structure of many fast DCT algorithms [6], [7] contains plane rotation blocks, where all nontrivial multipliers are concentrated. An efficient approach to fixed-point implementation of plane rotation using lifting schemes has been suggested in [8]. Another way to avoid floating-point multiplications in plane rotation is to employ the CORDIC algorithm [9]. This approach led to efficient approximations of the 8-point DCT with short critical paths [10], [11]. The main drawback of the mentioned approaches is that they use Loeffler's algorithm, which cannot be generalized to sizes larger than 16. Another problem is that resulting algorithms are characterized by have coding gain lower than the original DCT.

In this paper, we present an ASP-based systematic approach to synthesizing and implementing fast DCT-2 algorithms of power-of-two sizes. In order to obtain multiplierless high-accuracy implementations of the fast DCT algorithms, the algebraic integer (AI) technique proposed in [12] is used. However, unlike in [12], we derive a general AI encoding scheme for the proposed fast DCT algorithms. As a practical example, an AI-based fast implementation of the 16-point DCT is given, which offers high coding gains.

The paper is organized as follows. The mathematical framework of the ASP is introduced in Section II. Section III presents a derivation of a recursive fast DCT-2 algorithm of a power-of-two size. A general AI encoding scheme for the proposed DCT algorithms and an AI-based implementation of the $\mathrm{DCT\text{-}2}_{16}$ are described in Section IV. Additional considerations and conclusions are given in Section V.

In our derivations, the following matrices are used:

$$I_n = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} \quad J_n = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}.$$

II. Algebraic derivation of fast DCT algorithms

A. Background: polynomial algebra

In [4], it has been shown that fast DCT algorithms can be systematically derived using the notion of *polynomial algebra*. A polynomial algebra is a vector space over the field $\mathbb{F}$ and can be denoted as

$$\mathcal{A}_{\mathbb{F}} = \mathbb{F}[x]/p(x). \tag{1}$$

The elements of the algebra belong to the set of all polynomials in x over $\mathbb{F}$ of degrees smaller than $\deg(p) = n$. In the remainder of the paper, it is assumed that the polynomial

$p(x)$ has pairwise distinct zeros $\alpha = (\alpha_0, \dots, \alpha_{n-1})$. In $\mathcal{A}_{\mathbb{F}}$, additions and multiplications are performed modulo $p(x)$.

The Chinese remainder theorem (CRT) allows for decomposing the polynomial algebra (1) into a direct sum of one-dimensional subalgebras as

$$\mathcal{F}: \ \mathbb{F}[x]/p(x) \to \bigoplus_{0 \le k < n} \mathbb{F}_e[x]/(x - \alpha_k), \tag{2}$$

where $\mathbb{F}_e$ is some extension of the field $\mathbb{F}$. The mapping $\mathcal{F}$ can be represented in a matrix form as

$$\mathcal{F} = \mathcal{P}_{b,\alpha} = [p_\ell(\alpha_k)]_{0 \le k, \ell < n}, \tag{3}$$

provided a basis $b = (p_0, \dots, p_{n-1})$ with $\deg(p_i) < n$ is set for $\mathbb{F}[x]/p(x)$, and the unit basis $(x^0) = (1)$ is chosen in each $\mathbb{F}_e[x]/(x - \alpha_k)$. $\mathcal{P}_{b,\alpha}$ is referred to as a *polynomial transform* for $\mathcal{A}_{\mathbb{F}}$ with the basis b [5]. A *scaled polynomial transform* is obtained for a different basis β_k in each of subalgebras in (2):

$$\mathcal{F} = \mathrm{diag}(1/\beta_1, \dots, 1/\beta_{n-1}) \cdot \mathcal{P}_{b,\alpha}. \tag{4}$$

B. Fast algorithms: brief explanation

From the ASP theory, it is known that the DCT is a polynomial or scaled polynomial transform for the corresponding polynomial algebra $\mathbb{F}[x]/p(x)$ with the basis b. It can be seen from (2) that $\mathcal{F}$ decomposes $\mathbb{F}[x]/p(x)$ into one-dimensional polynomial algebras. A fast algorithm is obtained by applying this decomposition gradually (in steps) using intermediate subalgebras.

The most common way to decompose $\mathbb{F}[x]/p(x)$ is to employ the factorization $p(x) = q(x) \cdot r(x)$. If $\deg(q) = k$ and $\deg(r) = m$, then

$$\begin{aligned}
& \mathbb{F}[x]/p(x) \\
\to \ & \mathbb{F}[x]/q(x) \oplus \mathbb{F}[x]/r(x) && (5)\\
\to \ & \bigoplus_{0 \le i < k} \mathbb{F}[x]/(x - \beta_i) \oplus \bigoplus_{0 \le j < m} \mathbb{F}[x]/(x - \gamma_j) && (6)\\
\to \ & \bigoplus_{0 \le i < n} \mathbb{F}[x]/(x - \alpha_i) && (7)
\end{aligned}$$

where β_i and γ_j are the zeros of $q(x)$ and $r(x)$, respectively. If c and d are the bases of $\mathbb{F}[x]/q(x)$ and $\mathbb{F}[x]/r(x)$, respectively, then (5)–(7) can be expressed in a matrix form as follows [5]:

$$\mathcal{P}_{b,\alpha} = P(\mathcal{P}_{c,\beta} \oplus \mathcal{P}_{d,\gamma})B, \tag{8}$$

where $A \oplus B = \left[\begin{smallmatrix} A & \\ & B \end{smallmatrix}\right]$ denotes the direct sum of matrices. In the step (6), the CRT is used to decompose $\mathbb{F}[x]/q(x)$ and $\mathbb{F}[x]/r(x)$. This step corresponds to the direct summation of the matrices $\mathcal{P}_{c,\beta}$ and $\mathcal{P}_{d,\gamma}$. Finally, the permutation matrix P maps the concatenation (β, γ) onto the ordered list of the zeros α_i in (7). Assuming that B in (8) is sparse , this leads to a fast algorithm.

C. Polynomial algebras related to the DCT-2 and DCT-4

First, we consider the polynomial algebra associated with the DCT-4_n

$$\mathcal{A}_{\mathbb{F}} = \mathbb{F}[x]/2T_n(x), \quad b = (V_0, \dots, V_{n-1}), \tag{9}$$

where T_n and V_n denote the Chebyshev polynomials of the first and third kind, respectively, which can be represented using the following closed-form expressions

$$T_n(x) = \cos(n\theta), \ \ V_n(x) = \tfrac{\cos(n+\frac{1}{2})\theta}{\cos\frac{1}{2}\theta}. \tag{10}$$

assuming that $\cos\theta = x$, and $\alpha_k = \cos(k + \frac{1}{2})\frac{\pi}{n}$, $0 \le k < n$ are zeros of $2T_n(x)$. In accordance with (3), (9) corresponds to the polynomial transform

$$\mathcal{P}_{\alpha,b} = [V_\ell(\alpha_k)]_{0 \le k, \ell < n} = \left[\frac{\cos(k + \frac{1}{2})(\ell + \frac{1}{2})\frac{\pi}{n}}{\cos(k + \frac{1}{2})\frac{\pi}{2n}}\right]. \tag{11}$$

In order to realize the DCT-4_n matrix, (11) needs to be left-multiplied by the diagonal scaling matrix

$$D_n^{(C4)} = \mathrm{diag}_{0 \le k < n}\left(\cos(k + \tfrac{1}{2})\tfrac{\pi}{2n}\right)$$

which yields

$$\text{DCT-}4_n = \left[\cos(k + \tfrac{1}{2})(\ell + \tfrac{1}{2})\tfrac{\pi}{n}\right]_{0 \le k, \ell < n}. \tag{12}$$

The DCT-2_n is related to the polynomial algebra

$$\mathcal{A}_{\mathbb{F}} = \mathbb{F}[x]/(x-1)U_{n-1}(x), \quad b = (V_0, \dots, V_{n-1}), \tag{13}$$

where U_n denotes the Chebyshev polynomial of the second kind, which can be written for $(\cos\theta = x)$ as

$$U_n(x) = \tfrac{\sin(n+1)\theta}{\sin\theta}.$$

Since zeros of $U_n(x)$ are given by $\alpha_k = \cos\frac{(k+1)\pi}{n+1}$, $0 \le k < n$, the polynomial transform for (13) takes the form

$$\mathcal{P}_{\alpha,b} = [V_\ell(\alpha_k)]_{0 \le k, \ell < n} = \left[\frac{\cos k(\ell + \frac{1}{2})\frac{\pi}{n}}{\cos\frac{k\pi}{2n}}\right]. \tag{14}$$

In order to realize DCT-2, the matrix in (14) needs to be left-multiplied by the diagonal scaling matrix

$$D_n^{(C2)} = \mathrm{diag}_{0 \le k < n}\left(\cos\tfrac{k\pi}{2n}\right).$$

The polynomial transform that corresponds to the discrete trigonometric transform (DTT) is denoted as $\overline{\text{DTT}}$. For instance, $\overline{\text{DCT-4}}_n$ stands for the matrix in (11).

In the following, we need the skew DCT-$4(r)$. In [5], this transform has been introduced since it appears to be an important building block for Cooley-Tukey-type algorithms for computing the DCT. The skew DCT-$4(r)$ is associated with the polynomial algebra

$$\mathcal{A}_{\mathbb{F}} = \mathbb{F}[x]/(2T_n(x) - 2\cos r\pi)$$

where $0 < r < 1$, and with the basis $b = (V_0, \dots, V_{n-1})$. The conventional DCT-$4_n$ is a special case of the skew DCT-$4_n(r)$ for $r = 1/2$.

III. Synthesis of fast algorithms for computing the DCT-2 of power-of-two sizes

A. Selection of the base field

An important issue is to choose the base field $\mathbb{F}$ in (13). Since the Chebyshev polynomials have integer coefficients, like $V_2(x) = 4x^2 - 2x - 1$), it is natural to define $\mathbb{F}$ to be the field of rational numbers, $\mathbb{Q}$. The field is extended during factorization of the polynomial $U_{2^k-1}(x)$, since in the general case, the Chebyshev polynomials cannot be factorized over $\mathbb{Q}$.

B. Fast algorithm for DCT-2_{2n}

It is well known, see [3], that the DCT-2_{2n} can be reduced to the DCT-2_n and DCT-4_n. From the point of view of the ASP theory, this is a consequence of the existence of the following factorization

$$U_{2n-1}(x) = U_{n-1}(x) \cdot 2T_n(x),$$

which allows for decomposing the algebra $\mathbb{Q}[x]/(x-1)U_{2n-1}(x)$ with the basis $b = (V_0, \dots V_{2n-1})$ as follows

$$\begin{aligned} & \mathbb{Q}[x]/(x-1)U_{2n-1}(x) \\ \rightarrow \quad & \mathbb{Q}[x]/(x-1)U_{n-1}(x) \oplus \mathbb{Q}[x]/2T_n(x), \end{aligned} \tag{15}$$

According to (5)–(7), this leads to the following fast algorithm [5]

$$\overline{\text{DCT-2}}_{2n} = L_n^{2n}(\overline{\text{DCT-2}}_n \oplus \overline{\text{DCT-4}}_n)B_{2n}, \tag{16}$$

where L_n^{2n} is a stride permutation matrix, and B_{2n} is a basis-change matrix. B_{2n} maps the basis b onto the concatenation (c, d), where $c = d = (V_0, \dots V_{n-1})$ are the basis for the subalgebras in the right-hand side of (15). The first n columns of B_{2n} are

$$B_{2n} = \begin{bmatrix} I_n & * \\ I_n & * \end{bmatrix},$$

since the elements $V_\ell \in b$ for $0 \le \ell < n$ are already contained in c and d. The rest of entries are determined by the expressions

$$V_{n+\ell} \equiv \quad V_{n-\ell-1} \mod (x-1)U_n \tag{17}$$

$$V_{n+\ell} \equiv \quad -V_{n-\ell-1} \mod 2T_n, \tag{18}$$

which yields

$$B_{2n} = \begin{bmatrix} I_n & J_n \\ I_n & -J_n \end{bmatrix}.$$

One can deduce (17)–(18) using the following relation $2T_n = V_n + V_{n-1}$, $(x-1)U_{n-1} = V_n - V_{n-1}$ and $V_n = 2xV_{n-1} - V_{n-2}$. Please also notice that the decomposition (15) does not require extending the base field $\mathbb{Q}$. Thus, the matrix B_{2n} describes a basis change that needs no multiplications.

C. Fast algorithm for DCT-4_{2n}

If the size of DCT-2 is a power of two, then (16) can be applied recursively in order to obtain a fast algorithm. Hence, the problem of deriving a fast DCT-2_{2^k} algorithm reduces to deriving a fast DCT-$4_{2^{k-1}}$ algorithm. From the ASP point of view, the question is how to factorize the polynomial $2T_n$ in steps, when n is a power of 2. We propose to use the following general recursive formula

$$\begin{aligned} 2T_{2n}(x) - 2\cos r\pi = \ & \left(2T_n(x) - 2\cos\tfrac{r\pi}{2}\right) \\ & \times \left(2T_n(x) - 2\cos\pi(1-\tfrac{r}{2})\right), \end{aligned} \tag{19}$$

which can be proved using the closed form of T_{2n}, if parameter $r \in (0,\ 1)$. The special case of $r = 1/2$ in (19) for specify factorization of $2T_{2n}$. Using (19), the polynomial algebra related to DCT-$4_{2n}(r)$ is decomposed as follows[1]

$$\begin{aligned} & \mathbb{Q}_{\cos r\pi}[x]/(2T_{2n}(x) - 2\cos r\pi) \\ \rightarrow \quad & \mathbb{Q}_{\cos\frac{r\pi}{2}}[x]/(2T_n(x) - 2\cos\tfrac{r\pi}{2}) \oplus \\ & \mathbb{Q}_{\cos\frac{r\pi}{2}}[x]/(2T_n(x) - 2\cos\pi(1-\tfrac{r}{2})). \end{aligned} \tag{20}$$

The decomposition leads to the following fast algorithm

$$\begin{aligned} \overline{\text{DCT-4}}_{2n}(r) = P \cdot (&\overline{\text{DCT-4}}_n(\tfrac{r}{2}) \\ &\oplus \overline{\text{DCT-4}}_n(1-\tfrac{r}{2})) \cdot B_{2n}^{(C4)}(r), \end{aligned} \tag{21}$$

where P is the permutation matrix of the form

$$P = \begin{bmatrix} 1 & & & & \\ & & & I_2 & \\ & I_2 & & & \\ & & & & \ddots \\ & & \ddots & & \\ & & & & I_2 \\ & & 1 & & \end{bmatrix},$$

and $B_{2n}^{(C4)}(r)$ is the basis-change matrix

$$\begin{aligned} B_{2n}^{(C4)}(r) &= \begin{bmatrix} I_m & (2\cos\frac{r\pi}{2}I_m - J_m) \\ I_m & (-2\cos\frac{r\pi}{2}I_m - J_m) \end{bmatrix} \\ &= \begin{bmatrix} I_m & I_m \\ I_m & -I_m \end{bmatrix} \cdot \begin{bmatrix} I_m & -J_m \\ & 2\cos\frac{r\pi}{2}I_m \end{bmatrix}, \end{aligned} \tag{22}$$

which is determined by

$$\begin{aligned} V_{n+\ell} &\equiv -V_{n-\ell-1} + 2\cos\tfrac{r\pi}{2}V_\ell \mod 2T_n - 2\cos\tfrac{r\pi}{2} \\ V_{n+\ell} &\equiv -V_{n-\ell-1} - 2\cos\tfrac{r\pi}{2}V_\ell \mod 2T_n - 2\cos\pi(1-\tfrac{r}{2}). \end{aligned}$$

The decomposition (20) requires the base field $\mathbb{Q}_{\cos r\pi}$ to be extended to $\mathbb{Q}_{\cos\frac{r\pi}{2}}$. The new elements of the field appear in the matrix $B_{2n}^{(C4)}(r)$.

The joint use of the factorizations (16) and (21) leads to a fast recursive DCT-2_{2^k} algorithm. The key step of the algorithm is the multiplication by the matrix $B_{2n}^{(C4)}(r)$. All nontrivial multiplications are concentrated in it, which is very similar to the butterfly operation in the FFT algorithm. As an illustration, a Matlab implementation of the algorithm (21) is given in Listing 1.

Listing 1. Matlab implementation of the developed algorithm.

```
function [y] = dct4(r, x)
%DCT4 Computation of skew DCT4(r).
m  = length(x)/2;
Im = eye(m);          Jm = rot90(Im);
a  = 2*cos(pi*r/2); Zm = zeros(m,m);
B1 = [Im -Jm; Zm a*Im];
B2 = [Im Im; Im -Im];
x = B2*B1*x;
P = eye(2);
if m~=1
    x1 = dct4(r/2,   x(1:m));
    x2 = dct4(1-r/2, x(m+1:2*m));
    x  = [x1; x2];
    P  = zeros(2*m, 2*m);
    P(1,1) = 1; P(2*m,m) = 1;
    row = 2;  col = m+1; sign = -1;
    for i=1:m-1
        P(row,col) = 1;
        row = row+1; col = col+1;
```

[1] $\mathbb{Q}_{\cos r\pi}$ is used here as a short notation for the field extension $\mathbb{Q}[\cos r\pi]$.

```
        P(row,col) = 1;
        row = row+1; col = col+sign*m;
        sign= sign*(-1);
    end
end
y = P*x;
end
```

D. *Fast* DCT-2_{16} *algorithm*

In order to derive a fast DCT-2_{16} algorithm, the transform needs to be expressed as the product

$$\text{DCT-2}_{16} = D_{16}^{(C2)} \cdot \overline{\text{DCT-2}}_{16}. \tag{23}$$

Then, the factorizations (16) and (21) are applied recursively so as to obtain a fast algorithm. Fig. 1 shows the signal flow graph of this algorithm (for simplicity, the output scaling is omitted). In Fig. 2, the basic building block of the algorithm is shown that performs the multiplication by the matrix (22). As all operations inside the block are performed on m-component vectors, $3m$ additions and m multiplications are necessary to compute its output.

IV. AI-BASED HIGH-ACCURACY DCT IMPLEMENTATION

In [12], [13], the AI encoding of the basis functions of the DCT has been used to design low-complexity and parallel DCT processors. The essential idea of the AI technique is to represent nontrivial multipliers by polynomials with integer coefficients in z, where z is an algebraic integer. In this section, we use the AI encoding to derive a high-accuracy fast algorithm for the 16-point DCT-2. In the previous section, it has been shown that the DCT-2 can be recursively decomposed into the half-size DCT-2 and DCT-4. It is easy to notice that all nontrivial multiplications are concentrated in the DCT-4 stage. Since the fast DCT-2_{16} algorithm uses three DCT-4 stages of sizes 2, 4 and 8, three different AI encoding schemes are necessary to implement the 16-point DCT-2.

A. *Algebraic integers*

In the general case, an algebraic integer is a complex number that is a root of a certain monic polynomial with coefficients in $\mathbb{Z}$ (the set of integers). For example, $z = 2\cos(\pi/8) = \sqrt{2+\sqrt{2}}$ is a root of the polynomial $p(z) = z^4 - 4z^2 + 2$. If z is adjoined to the rational numbers, then the associated ring of AIs is denoted by $\mathbb{Q}[z]$ and can be considered the vector space that consists of polynomials in z of degree 3 with integer coefficients. Addition and multiplication of elements of $\mathbb{Q}[z]$ are performed modulo $p(z)$.

B. *1-D algebraic integer encoding*

To obtain AI encoding scheme for a concrete algorithm, it is necessary to define both appropriate algebraic integer z and polynomial $p(z)$, such that $p(z) = 0$, and then to express all nontrivial multipliers in the algorithm by polynomials in z with integer coefficients. Let us consider the general case of constructing an AI encoding scheme for the DCT-4_n, where n is a power of two. From the previous section, it is known that all multipliers that occur in the fast DCT-4_n algorithm have the form $2\cos(\frac{k\pi}{2n})$, where $0 \le k < n$. Thus it is reasonable to define $z = 2\cos(\frac{\pi}{2n})$, so that the polynomial $p(z)$ and all multipliers $2\cos(\frac{k\pi}{2n})$ can be expressed using the Chebyshev polynomials of the first kind:

$$p(z) = 2T_n(z/2),$$
$$2\cos(k\pi/8) = 2T_k(z/2).$$

This approach was applied to obtain AI encoding schemes for DCT-4_2, DCT-4_4 and DCT-4_8 that are appears in the signal flow graph of fast DCT-2_{16} algorithm (Table I).

TABLE I
AI ENCODING SCHEMES FOR DCT-4_2, DCT-4_4 AND DCT-4_8

AI representation of multiplier for DCT-4_2 ($z = 2\cos(\pi/4)$)
$p(z) = z^2 - 2$
$2\cos(\pi/4) = z$

AI representation of multipliers for DCT-4_4 ($z = 2\cos(\pi/8)$)
$p(z) = z^4 - 4z^2 + 2$
$2\cos(\pi/8) = z$
$2\cos(2\pi/8) = z^2 - 2$
$2\cos(3\pi/8) = z^3 - 3z$

AI representation of multipliers for DCT-4_8 ($z = 2\cos(\pi/16)$)
$p(z) = z^8 - 8z^6 + 20z^4 - 16z^2 + 2$
$2\cos(\pi/16) = z$
$2\cos(2\pi/16) = z^2 - 2$
$2\cos(3\pi/16) = z^3 - 3z$
$2\cos(4\pi/16) = z^4 - 4z^2 + 2$
$2\cos(5\pi/16) = z^5 - 5z^3 + 5z$
$2\cos(6\pi/16) = z^6 - 6z^4 + 9z^2 - 2$
$2\cos(7\pi/16) = z^7 - 7z^5 + 14z^3 - 7z$

Please notice that there is no longer any precision problem since the AI encoding provides an exact representation of irrational numbers. An example of using AI encoding in computations is shown in Fig. 3, where the signal flow graph of a fast DCT-4_4 algorithm is depicted.

In a similar way, computational schemes for DCT-4_2 and DCT-4_8 can be obtained.

C. *Reconstruction step*

The AI technique requires representing the outputs of a transform as polynomials of the form $f(z) = \sum_{i=0}^{n-1} a_i z^i$. To map such a polynomial onto a binary output value, a final reconstruction stage (FRS) is necessary, which can be based on Horner's rule [14]:

$$f(z) = (\dots(a_{n-1}z + a_{n-2})z + a_{n-3})z + \dots + a_1)z + a_0.$$

For DCT-4_2, DCT-4_4, and DCT-4_8, we use in the FRS the following approximations of z:

$$2\cos(\pi/4) \approx 1 + 2^{-1} - 2^{-3} + 2^{-5} + 2^{-7},$$
$$2\cos(\pi/8) \approx 2 - 2^{-3} - 2^{-5} + 2^{-8},$$
$$2\cos(\pi/16) \approx 2 - 2^{-5} - 2^{-7} + 2^{-11}.$$

The signal flow graph of the FRS for the DCT-4_4 is shown in Fig. 4. The result is obtained after four cycles, in which

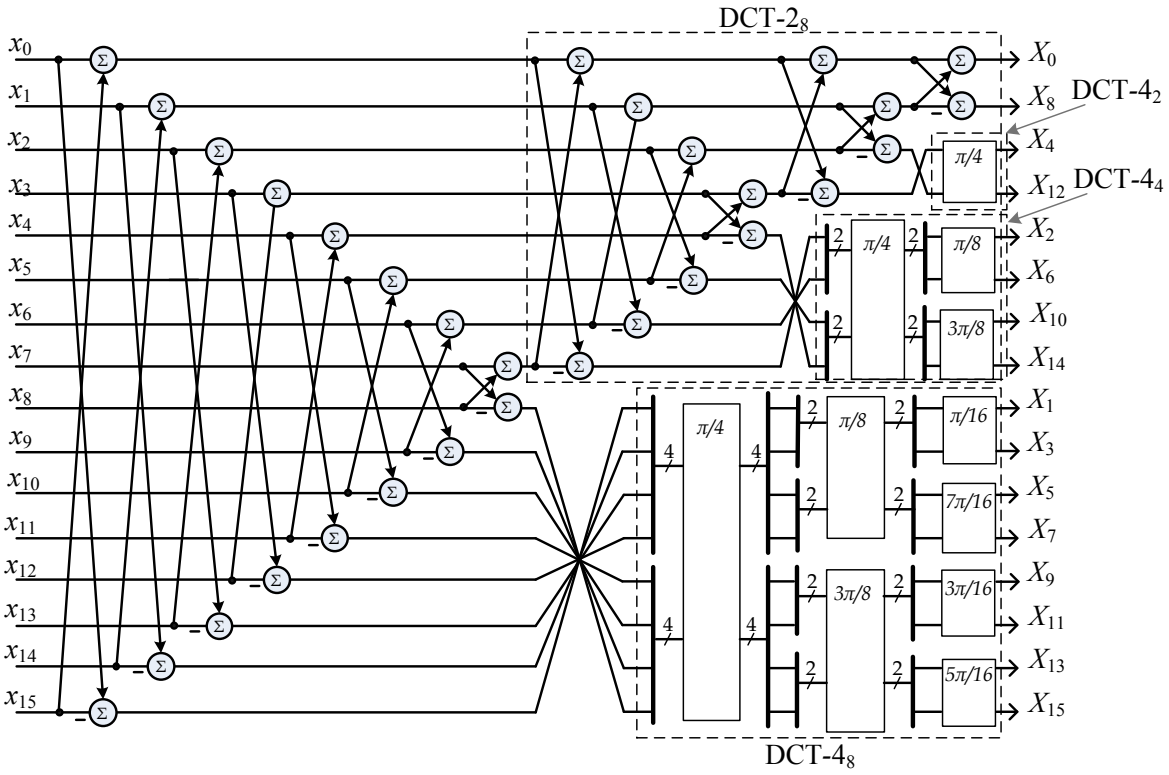

Fig. 1. Signal flow graph of 16-point DCT algorithm

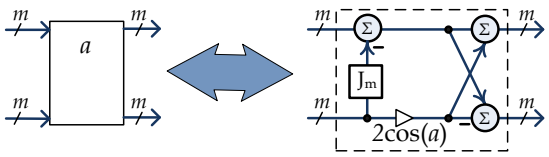

Fig. 2. The building block for fast DCT algorithms

the sequence of the polynomial coefficients, $a_3, \ldots, a_0$, are passed through the scheme. The reconstruction of DCT-4_2 and DCT-4_8 outputs is performed in a similar way.

It should be noted that the FRS introduces some rounding errors, but they occur at the very end of the transformation process and are not distributed throughout the computations as in a fixed-point implementation.

D. Coding gain

The performance of the proposed AI-based fast DCT-16_2 algorithm was measured using biorthogonal coding gain, defined as in [8]:

$$C_g \triangleq 10 \log_{10} \frac{\sigma_x^2}{\left(\prod_{k=0}^{n-1} \sigma_{x_k}^2 \|f_k\|^2 \right)^{\frac{1}{n}}},$$

where n is the number of transform basis functions, σ_x^2 is the input variance, $\sigma_{x_k}^2$ is the variance of the k-th transform output, and $\|f_k\|^2$ is the norm of the k-th synthesis basis function. As an input signal is a first-order Gaussian Markov process with zero-mean, unit variance and correlation coefficient $\rho = 0.95$ was used (a good approximation of natural images).

TABLE II
MEASURES OF CODING GAIN

Transform	Coding gain
Floating-point DCT-16_2	9.4555 [dB]
16-point binDCT [8]	9.4499 [dB]
Proposed algorithm (9 bit fractional part)	9.4546 [dB]
Proposed algorithm (12 bit fractional part)	9.4553 [dB]

Coding gain is the critical parameter of a transform for image compression, as its higher value indicates that the transform compacts more energy into a fewer number of coefficients, and better quality of decoded images can be achieved. Table II lists the coding gains of the original DCT-16_2, 16-point binDCT algorithm [8], and proposed algorithm for computing DCT-16_2. The results shows that using the AI technique, multiplierless algorithms for computing the DCT can be obtained, which have coding gains very close to the true 16-point DCT.

V. CONCLUSION

A fast algorithm for the DCT-2 of a power-of-two size is presented, which is based on ASP. The main features of the

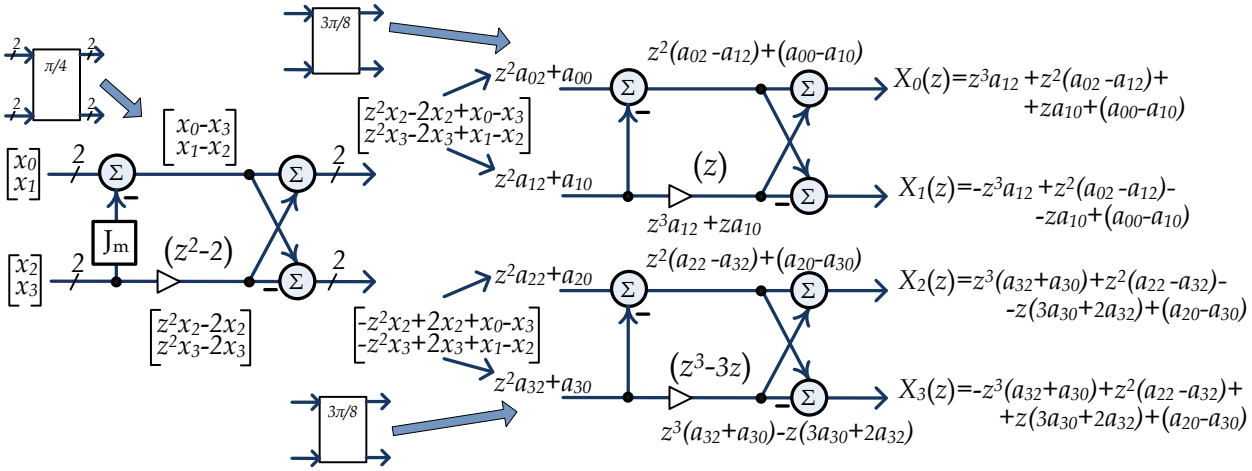

Fig. 3. Computation of DCT-4_4 based on algebraic integer encoding. Computational cost: 13 addition and 4 shifts.

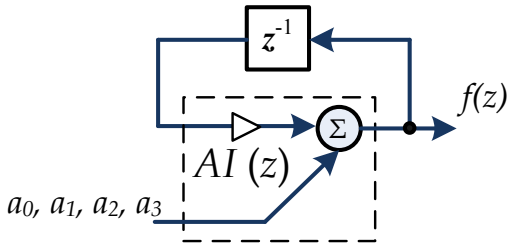

Fig. 4. Final reconstruction step (DCT-4_4)

proposed algorithms are regularity of the signal flow graph and low arithmetic complexity. Also the implementation of a fast DCT-2_{16} algorithm based on AI technique is presented. Algorithms for computing the DCT-2 of a higher size can be obtained using the same design method. The proposed AI-based fast implementation of the DCT-2_{16} is characterized by high performance and can be applied in different image/video compression applications.

Acknowledgment

This work was supported by the Bialystok University of Technology under the grant S/WI/4/08 and by the Belarusian Fundamental Research Fund (F11MS-037).

References

[1] R. Joshi, Y. A. Reznik, and M. Karczewicz, "Efficient large size transforms for high-performance video coding," in *SPIE*, vol. 7798, 2010.

[2] V. Britanak, P. Yip, and K. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Academic, 2007.

[3] K. Rao and P. Yip, *Discrete cosine transform: algorithms, advantages, applications*. Academic Press, 1990.

[4] M. Püschel and J. M. F. Moura, "The algebraic approach to the discrete cosine and sine transforms and their fast algorithms," *SIAM Journal on Computing*, vol. 32, no. 5, pp. 1280–1316, 2003.

[5] ——, "Algebraic signal processing theory: Cooley-Tukey type algorithms for DCTs and DSTs," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1502–1521, 2008.

[6] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1989, pp. 988–991.

[7] W.-H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Transactions on Communications*, vol. 25, no. 9, pp. 1004–1009, 1977.

[8] J. Liang and T. D. Tran, "Fast multiplierless approximations of the DCT with the lifting scheme," *IEEE Trans. on Signal Processing*, vol. 49, pp. 3032–3044, 2001.

[9] M. Parfieniuk and A. Petrovsky, "Structurally orthogonal finite precision implementation of the eight point DCT," in *Proceedings of International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 3, 2006, pp. 936–939.

[10] M. Parfieniuk, "Shortening the critical path in cordic-based approximations of the eight-point DCT," in *Proceedings of International Conference on Signals and Electronic Systems (ICSES)*, 2008, pp. 405–408.

[11] M. Vashkevich, M. Parfieniuk, and A. Petrovsky, "FPGA implementation of short critical path CORDIC-based approximation of the eight-point DCT," in *Proceedings of International Conference on Pattern Recognition and Information Processing (PRIP)*, 2009, pp. 161–164.

[12] V. Dimitrov and K. Wahid, "Multiplierless dct algorithm for image compression applications," *International Journal: Information Theories and Applications*, vol. 11, no. 2, pp. 162–169, 2004.

[13] ——, "On the error-free computation of fast cosine transform," *International Journal: Information Theories and Applications*, vol. 12, no. 4, pp. 321–327, 2005.

[14] D. E. Knuth, *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.