

# 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 synthesizing and implementing 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-Loève 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 implementations of DCT algorithms. It is an important issue since floating-point multiplication is rather slow in hardware/software realizations. Many structures of fast DCT algorithms [6], [7] contain 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 has resulted

in 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 the obtained algorithms are characterized by coding gains lower than the original DCT.

In this paper, we present an ASP-based systematic approach to synthesizing and implementing fast DCT-2 transforms 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 16-point DCT-2 are described in Section IV. Additional considerations and conclusions are given in Section V.

In our derivations, the following matrices are used:

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

## 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

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

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

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

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

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

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  $\beta_k$  in each of subalgebras in (2):

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

### 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

$$\begin{aligned} & \mathbb{F}[x]/p(x) \\ \rightarrow & \mathbb{F}[x]/q(x) \oplus \mathbb{F}[x]/r(x) \end{aligned} \quad (5)$$

$$\rightarrow \bigoplus_{0 \leq i < k} \mathbb{F}[x]/(x - \beta_i) \oplus \bigoplus_{0 \leq j < m} \mathbb{F}[x]/(x - \gamma_j) \quad (6)$$

$$\rightarrow \bigoplus_{0 \leq i < n} \mathbb{F}[x]/(x - \alpha_i) \quad (7)$$

where  $\beta_i$  and  $\gamma_j$  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]:

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

where  $A \oplus B = \begin{bmatrix} A & \\ & B \end{bmatrix}$  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  $(\beta, \gamma)$  onto the ordered list of the zeros  $\alpha_i$  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<sub>n</sub> (abbreviation for the  $n$ -point DCT-4)

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

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

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

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

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

In order to realize the DCT-4<sub>n</sub> matrix, (11) needs to be left-multiplied by the diagonal scaling matrix

$$D_n^{(C4)} = \text{diag}_{0 \leq k < n} \left( \cos(k + \frac{1}{2})\frac{\pi}{2n} \right)$$

which yields

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

The DCT-2<sub>n</sub> is related to the polynomial algebra

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

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

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

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

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

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

$$D_n^{(C2)} = \text{diag}_{0 \leq k < n} \left( \cos \frac{k\pi}{2n} \right).$$

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<sub>n</sub>( $r$ ). This transform has been introduced in [5] because it appears to be an important building block for Cooley-Tukey-type algorithms for computing the DCT. The skew DCT-4<sub>n</sub>( $r$ ) 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<sub>n</sub> is a special case of the skew DCT-4<sub>n</sub>( $r$ ) 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, e.g.  $V_2(x) = 4x^2 - 2x - 1$ , 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  $U_{2k-1}(x)$ , since in the general case, the Chebyshev polynomials cannot be factorized over  $\mathbb{Q}$ .



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  $B_{2^n}^{(C^4)}(r)$ . 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.

#### 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

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

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 (the output scaling has been omitted for clarity). 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 as 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

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

<b>AI representation of multiplier for DCT-<math>4_2</math></b> ( $z = 2\cos(\pi/4)$ )
$p(z) = z^2 - 2$
$2\cos(\pi/4) = z$
<b>AI representation of multipliers for DCT-<math>4_4</math></b> ( $z = 2\cos(\pi/8)$ )
$p(z) = z^4 - 4z^2 + 2$
$2\cos(\pi/8) = z$
$2\cos(2\pi/8) = z^2 - 2$
$2\cos(3\pi/8) = z^3 - 3z$
<b>AI representation of multipliers for DCT-<math>4_8</math></b> ( $z = 2\cos(\pi/16)$ )
$p(z) = z^8 - 8z^6 + 20z^4 - 16z^2 + 2$
$2\cos(\pi/16) = z$
$2\cos(2\pi/16) = z^2 - 2$
$2\cos(3\pi/16) = z^3 - 3z$
$2\cos(4\pi/16) = z^4 - 4z^2 + 2$
$2\cos(5\pi/16) = z^5 - 5z^3 + 5z$
$2\cos(6\pi/16) = z^6 - 6z^4 + 9z^2 - 2$
$2\cos(7\pi/16) = z^7 - 7z^5 + 14z^3 - 7z$

all multipliers that occur in the fast DCT- $4_n$  algorithm have the form  $2\cos(\frac{k\pi}{2n})$ , where  $0 \leq k < n$ . 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 has been used 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).

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  $z$ :

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

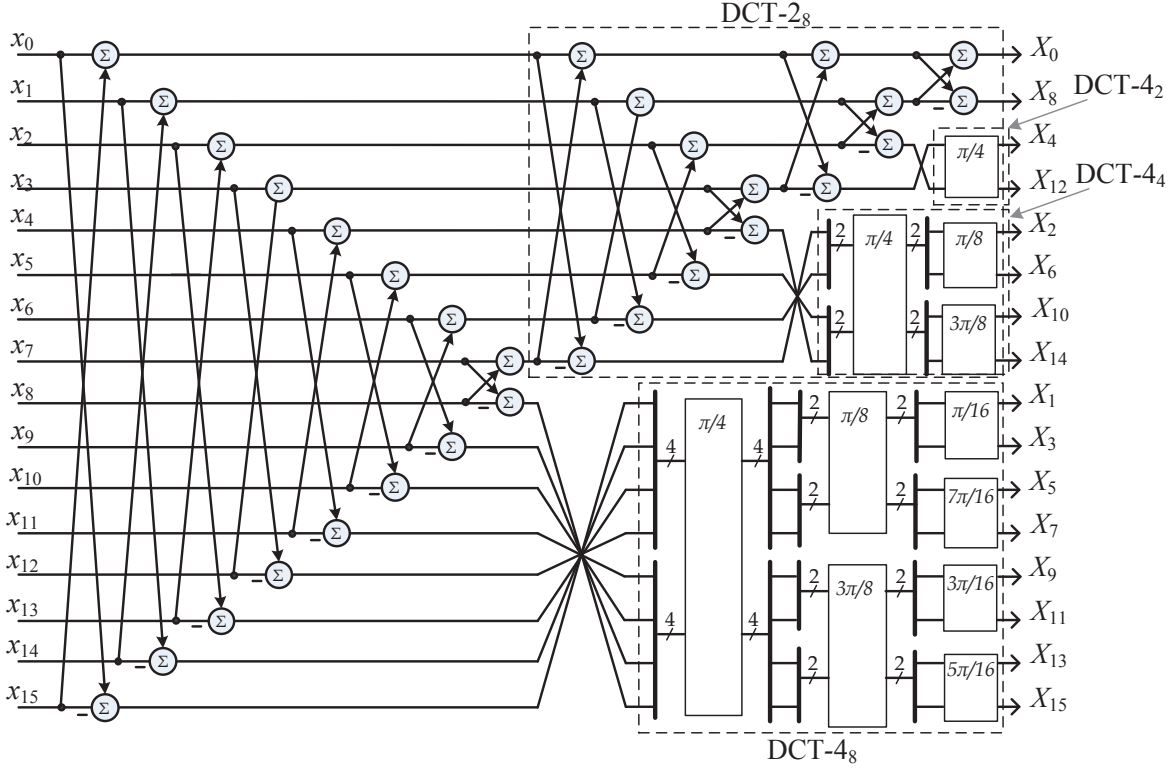


Fig. 1. Signal flow graph of 16-point DCT algorithm

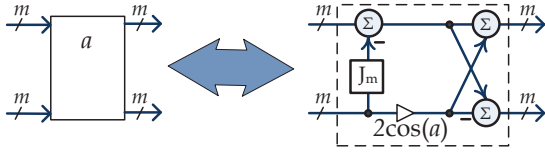


Fig. 2. The building block for fast DCT algorithms

The signal flow graph of the FRS for the DCT-4<sub>4</sub> is shown in Fig. 4. The result is obtained after four cycles, in which the sequence of the polynomial coefficients,  $a_3, \dots, a_0$ , are passed through the scheme. The reconstruction of DCT-4<sub>2</sub> and DCT-4<sub>8</sub> 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-2<sub>16</sub> algorithm was measured using biorthogonal coding gain, defined in [8] as:

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

where  $n$  is the number of the basis functions of a transform,  $\sigma_x^2$  is the input variance,  $\sigma_{x_k}^2$  is the variance of the  $k$ th

TABLE II  
MEASURES OF CODING GAIN

Transform	Coding gain
Floating-point DCT-2 <sub>16</sub>	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]

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).

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-2<sub>16</sub>, 16-point binDCT algorithm [8], and proposed algorithm for computing DCT-2<sub>16</sub>. 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.

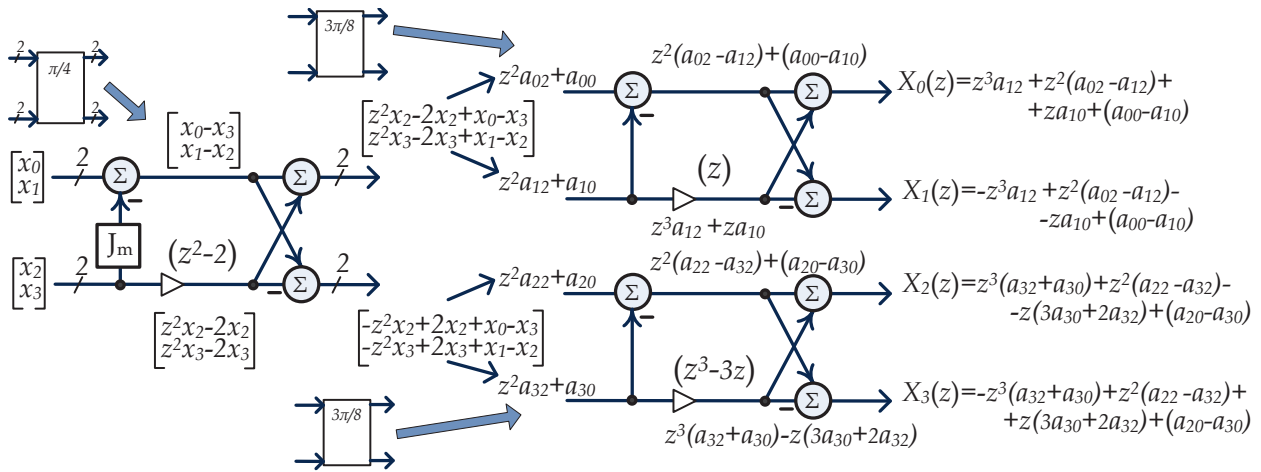


Fig. 3. Computation of DCT-44 based on algebraic integer encoding. Computational cost: 13 addition and 4 shifts.

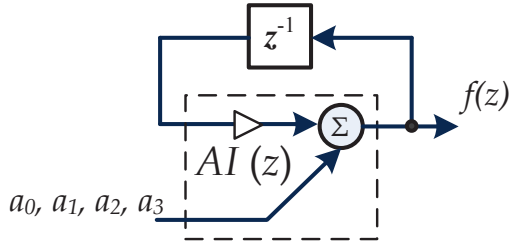


Fig. 4. Final reconstruction step (DCT-44)

## V. CONCLUSION

A fast algorithm for the DCT-2 of a power-of-two size has been developed, which is based on ASP. The main features of the proposed algorithms are regularity of the signal flow graph and low arithmetic complexity. Also an implementation of the fast DCT-2<sub>16</sub> 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<sub>16</sub> 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. Reznik, and M. Karczewicz, "Efficient large size transforms for high-performance video coding," in *Proc. SPIE*, vol. 7798 (Applications of Digital Image Processing XXXIII), San Diego, CA, Aug. 2010, pp. 1–7, paper no. 7798-31.
- [2] V. Britanak, P. C. Yip, and K. R. Rao, *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations*. Amsterdam: Elsevier/Academic Press, 2007.
- [3] K. R. Rao and P. Yip, *Discrete Cosine Transform: algorithms, advances, applications*. New York, NY: Academic Press, 1990.

- [4] M. Püschel and J. Moura, "The algebraic approach to the discrete cosine and sine transforms and their fast algorithms," *SIAM J. Computing*, vol. 32, no. 5, pp. 1280–1316, Mar. 2003.
- [5] —, "Algebraic signal processing theory: Cooley-Tukey type algorithms for DCTs and DSTs," *IEEE Trans. Signal Process.*, vol. 56, no. 4, pp. 1502–1521, Apr. 2008.
- [6] C. Loeffler, A. Lightenberg, and G. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 2, Glasgow, Scotland, 23–26 May 1989, pp. 988–991.
- [7] W. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the Discrete Cosine Transform," *IEEE Trans. Commun.*, vol. 25, no. 9, pp. 1004–1009, Sep. 1977.
- [8] J. Liang and T. D. Tran, "Fast multiplierless approximations of the DCT with the lifting scheme," *IEEE Trans. Signal Process.*, vol. 49, no. 12, pp. 3032–3044, Dec. 2001.
- [9] M. Parfieniuk and A. Petrovsky, "Structurally orthogonal finite precision implementation of the eight point DCT," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 3, Toulouse, France, 14–19 May 2006, pp. 936–939.
- [10] M. Parfieniuk, "Shortening the critical path in CORDIC-based approximations of the eight-point DCT," in *In Proc. Int. Conf. on Signals and Electronic Systems (ICSES)*, Cracow, Poland, 14–17 Sep. 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 *Proc. 10th Int. Conf. "Pattern Recognition and Information Processing" (PRIP)*, Minsk, Belarus, 19–21 May 2009, pp. 161–164.
- [12] V. Dimitrov and K. Wahid, "Multiplierless DCT algorithm for image compression applications," *Int. J. "Information Theories and Applications"*, vol. 11, no. 2, pp. 162–169, 2004.
- [13] —, "On the error-free computation of fast cosine transform," *Int. J. "Information Theories and Applications"*, vol. 12, no. 4, pp. 321–327, 2005.
- [14] D. E. Knuth, *Art of Computer Programming, Volume 2: Seminumerical Algorithms*, 3rd ed. Boston, MA: Addison-Wesley, 1997.