Parallel Architectures for 8×8 Discrete Cosine Transforms

S. Wolter, D. Birreck, M. Heine, R. Laur

Institut für Mikroelektronik, Universität Bremen
P.O. Box 330 440, D-2800 Bremen 33, Germany
Tel.: +49 421 218-4079, Fax: +49 421 218-4434

Abstract - **This paper addresses the design of parallel architectures for computing the 8×8 Discrete Cosine Transform (DCT). It concentrates on direct methods, which avoid a row-column decomposition. Two novel multiplier-free parallel architectures for high-speed 8×8 DCT calculation are proposed. The first architecture, which uses polynomial transforms, is compared with a second architecture, which computes the DCT via the Walsh-Hadamard Transform (WHT). Both architectures achieve a high degree of parallelism and regularity. The proposed architectures are designed for HDTV sampling rates and can be efficiently realized in CMOS technology.**

1. Introduction

The design of parallel architectures for the Discrete Cosine Transform is an important issue in the field of data compression. In particular the 8×8 DCT is now widely used in image coding and there is an increasing demand for circuits, which can achieve HDTV pixel rates [1,2].
A classification for 2D DCTs is proposed in [3]. The designs are divided into two categories, row-column algorithms (RCAs) and not-row-column algorithms (NRCAs). The later avoid the matrix transposition, which arises from the row-column decomposition. They work directly on the 2D data set. Therefore, they are able to compute all N^2 transform coefficients at the same time. So NRCAs have a natural advantage over RCAs. Unfortunatly, NRCAs normally have a very complex structure, which does not allow an easy implementation [6,8]. The aim of this paper is to present two architectures using NRCAs, which achieve a high degree of parallelism and of regularity and modularity. This is obtained by a higher number of multiplications and additions, than necessary for recursive fast algorithms. But the goal is to get a regular data path and circuit structure.
The first proposed architecture computes the 2D DCT via the two dimensional Discrete Fourier Transfom (2D DFT), using a decomposition of the 2D DFTs into modularized fast polynomial transforms and 1D DFTs. The evaluation of the 1D DFTs and the conversion of the DFT into the DCT are realized with distributed arithmetic (DA) modules. Therefore, the architecture is multiplier-free.
The second architecture computes the 2D DCT via the 2D Walsh-Hadamard Transform (WHT). The computation of the WHT requires no multiplications. The conversion of the Walsh-Hadamard Transform data into the DCT coefficients can be done using a precomputed and stored conversion matrix. This approach also leads to the use of distributed arithmetic and avoids multipliers.
In the next section, algorithms and architectures for both approaches are presented. The last section deals with the comparision of the architectures. The paper is summarized and some conclusions are drawn.

2. Parallel architectures for 8×8 DCTs

The 8×8 Discrete Cosine Transform of type II [1] is defined as

$$Z(k_1,k_2) = n(k_1,k_2) \sum_{n_1=0}^{7} \sum_{n_2=0}^{7} x(n_1,n_2) \cos\frac{k_1(2n_1+1)\pi}{16} \cos\frac{k_2(2n_2+1)\pi}{16} \quad (1)$$

where $n(k_1,k_2)$ is a normalizing factor. The matrix representation is given by

$$Z = CXC^T. \quad (2)$$

Z, C and X are 8×8 real-valued matrices.

2.1 Architecture using polynomial transforms

Following the derivation in [5], the 2D DCT can be calculated by cosine- and sine-Discrete Fourier Transforms. In the following we derive a polynomial transform scheme for an 8×8 DFT.
The DFT defined as

$$DFT(k_1,k_2) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} W_N^{n_1k_1+n_2k_2} x(n_1,n_2) \quad (3)$$

can be represented by a set of three polynomials:

$$X_{n_1}(z) = \sum_{n_2=0}^{N-1} x(n_1,n_2)\, z^{n_2} \; ; \quad n_1=0 \ldots N-1 \quad (4a)$$

$$X_{k_1}(z) = \sum_{n_1=0}^{N-1} X_{n_1}(z)\, W_N^{n_1k_1} \; (mod(z^N-1)) \quad (4b)$$

$$DFT(k_1,k_2) = X_{k_1}(z)\, mod(z-W_N^{k_2}) \quad (4c)$$

To develop a modularized polynomial transform structure a decomposition of the operands in the modulo operation in (4b) and (4c) is favourable. This leads to a divide and conquer strategy, in which the equations (4a), (4b) and (4c) can be reduced to smaller lengths. Furthermore, it is favourable to avoid a recursive computation, in order to get a modular structure. The decomposition depends on the parity of the index k_2. We select two subsets for k_2: $k_2 = odd$ and $k_2 = even$.

a) Let us first consider the case $k_2 = odd$.
Equation (4b) can be written as a polynomial transform by a permutation technique [4], which is repeated in the following. Let s be any positive integer relatively prime to $N = 2^r$, r integer. Let k_1 be expressed as $((s\,k_1)) = sk_1 mod N$. For $k_2 = odd$, s can be replaced by k_2. The permutation $((k_2k_1))$ maps all values of k_1.
For N= 8 this permutation reorders the output index of eq. (4c):

$$DFT(((k_2k_1)),k_2) = X_{((k_2k_1))}(z) mod(z - W_8^{k_2}) \quad (5)$$

Using the fact, that

$$W_8^{n_1((k_2k_1))} = W_8^{n_1k_2k_1} \quad (6)$$

0-7803-0593-0/92 $3.00 1992 IEEE

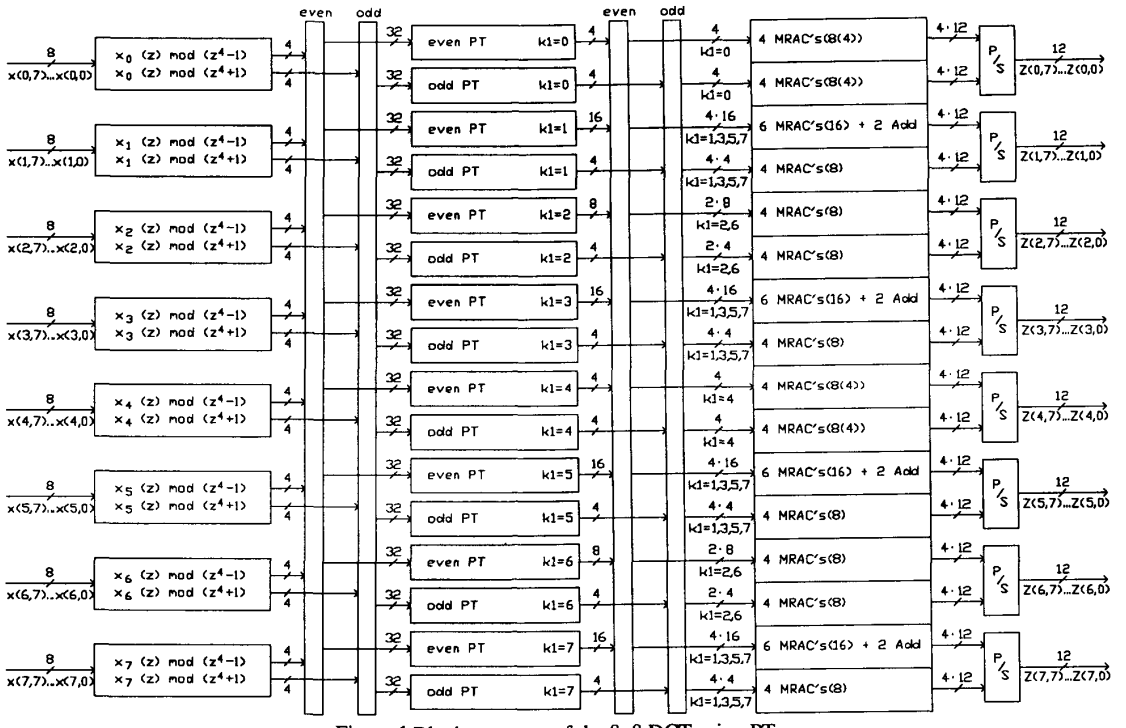

Figure 1:Block structure of the 8x8 DCT using PTs

and that z is substituted by $z = W_8^{k_2}$ by the modulo operation in eq. (5), we can replace (4b) as follows:

$$X_{((k_2\,k_1))}(z) = \sum_{n_1=0}^{7} X_{n_1}(z) \cdot z^{n_1 k_1} mod(z^8 - 1) \tag{7}$$

This is exactly a polynomial transform, as defined in [4]. The PT can be computed without multiplications. Only shifts and additions are needed. Now, we concentrate on the second modulo operation in eq. (7). Since z^8-1 can be factorized as

$$z^8-1 = (z^4-1)\,(z^4+1) \tag{8}$$

and $z-W_8^{k_2}$ is a root of z^4+1 for $k_2 = odd$, the polynomial transform (7) can be reduced to

$$Xo_{((k_2 k_1))}(z) = \sum_{n_1=0}^{7} Xo_{n_1}(z)\, z^{n_1 k_1}\, mod(z^4+1) \tag{9}$$

Under these conditions, the polynomial in eq. (4a) has to be reduced with $mod(z^4+1)$ to $Xo_{n_1}(z)$.

To summarize the 8×8 DFT computation for $k_2 = odd$, we separates this into three steps:

1. A polynomial reduction:

$$Xo_{n_1}(z) = X_{n_1}(z)\ mod(z^4+1) \tag{10}$$

2. A polynomial transform, as expressed in eq. (9).

3. Evaluation of $Xo_{((k_2 k_1))}(z)$ at the points $z = W_8^{k_2}$ for $k_2 = 1,3,5,7$:

$$odd\text{–}DFT(\,((k_2 k_1)), k_2) = Xo_{((k_2 k_1))}(z)\ mod(z - W_8^{k_2}) \tag{11}$$

b) The part with $k_2 = even$ can be derived from the part with $k_2 = odd$. In order to understand the following DFT computation for $k_2 = even$, we explain some properties of the DFT matrix as defined in () for $k_2 = odd$ and $k_2 = even$. The matrix W_8 shall describe the DFT-matrix for all values of n_1 and n_2:

$$W_8 = \begin{pmatrix} W_8^{0+0} & W_8^{k_2+0} & \dots & W_8^{7k_2+0} \\ W_8^{0+k_2} & W_8^{k_2+k_1} & \dots & W_8^{7k_2+k_1} \\ \cdot & \cdot & \dots & \cdot \\ W_8^{0+7k_1} & W_8^{k_2+7k_1} & \dots & W_8^{7k_2+7k_1} \end{pmatrix} \tag{12}$$

This matrix can be partitioned into four quadrants. In the case, where $k_2 = even$, the block matrix representation can be written as:

$$W_8^{even} = \begin{pmatrix} T_4 & T_4 \\ \pm T_4 & \pm T_4 \end{pmatrix} \tag{13}$$

T_4 is the upper left quadrant in W_8:

$$T_4 = \begin{pmatrix} T_{0,0} & T_{0,1} & T_{0,2} & T_{0,3} \\ T_{1,0} & T_{1,1} & T_{1,2} & T_{1,3} \\ T_{2,0} & T_{2,1} & T_{2,2} & T_{2,3} \\ T_{3,0} & T_{3,1} & T_{3,2} & T_{3,3} \end{pmatrix} \tag{14}$$

The upper sign for the two lower matrices is valid for $k_1 = even$ and the lower sign is valid for $k_1 = odd$.

In the case, where $k_2 = odd$, the partioning for the corresponding values looks like

$$k_2^{even} = k_2^{odd} - 1 \tag{15}$$

of k_2 as follows:

$$W_8^{odd} = \begin{pmatrix} T_4^* & -T_4^* \\ \pm T_4^* & -(\pm)\,T_4^* \end{pmatrix} \tag{16}$$

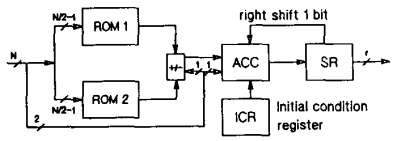

Figure 2: The MRAC(N)

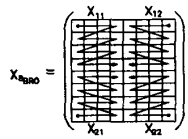

Figure 3: Permutaion of the input matrix (WHT architecture)

where the upper sign is again valid for $k_1 = even$ and the lower sign is valid for $k_1 = odd$.

The matrix T_4^* is defined as:

$$T_4^* = \begin{pmatrix} T_{0,0}\cdot W_8^0 & T_{0,1}\cdot W_8^1 & T_{0,2}\cdot W_8^2 & T_{0,3}\cdot W_8^3 \\ T_{1,0}\cdot W_8^0 & T_{1,1}\cdot W_8^1 & T_{1,2}\cdot W_8^2 & T_{1,3}\cdot W_8^3 \\ T_{2,0}\cdot W_8^0 & T_{2,1}\cdot W_8^1 & T_{2,2}\cdot W_8^2 & T_{2,3}\cdot W_8^3 \\ T_{3,0}\cdot W_8^0 & T_{3,1}\cdot W_8^1 & T_{3,2}\cdot W_8^2 & T_{3,3}\cdot W_8^3 \end{pmatrix} \quad (17)$$

First, we can see that the right matrices interchange their sign. If we want to compute the $k_2 = even$ values of the DFT from the $k_2 = odd$ values, this is the first difference that we have to take into account. The second difference between both matrices are the additional twiddle factors in the matrix T_4^*. Taking into account both differences between $k_2 = odd$ and $k_2 = even$ this leads to a similiar computational scheme for $k_2 = even$.

The resulting three equations are:

1. A polynomial reduction:

$$Xe_{n_1}(z) = X_{n_1}(z) \; mod(z^4-1), \quad (18)$$

2. A polynomial transform:

$$Xe_{(((k_2+1)k_1))}(z) = \sum_{n_1=0}^{7} Xe_{n_1}(z)\, z^{n_1 k_1} \; mod(z^4+1) \quad (19a)$$

$$= \sum_{i=0}^{3} X^{(e,i)}_{(((k_2+1)k_1))}\, z^i \quad (19b)$$

3. The evaluation of $Xe_{(((k_2+1)k_1))}(z)$ at the points $z = W_8^{k_2}$ corrected by the factors W_8^{-i} for $k_2 = 0,2,4,6$:

$$even\text{-}DFT(((k_2+1)k_1)),k_2) = \sum_{i=0}^{3} X^{(e,i)}_{(((k_2+1)k_1))} z^i W_8^{-i} \; mod(z - W_8^{k_2}) \quad (20)$$

Hence we have achieved a modularized structure, in which the reduced polynomials $Xo_{n_1}(z)$ and $Xe_{n_1}(z)$ have a degree of 3 and the odd-DFTs and the even-DFTs are of length 4.

The block structure of the proposed architecture to calculate the 8×8 DCT using the modularized PT scheme is depicted in Fig. 1. All DCT coefficients are calculated concurrently.

The block diagram shows three stages:

1) Input permutation and polynomial reduction,
2) Polynomial transform evaluation,
3) DFT evaluation and DFT into DCT conversion.

The elements of the input matrix $x(n_1,n_2)$ are read in row for row in a lexicographical order. The first stage realizes the input permutation necessary to convert the DCT into a DFT and the polynomial reductions for each row. The permutation of the elements in each row is taken into account in the polynomial reduction device. The input modules are all identical. Then, the polynomial transforms (PTs) for the corresponding odd and even values of k_2 are calculated in adjacent blocks. In the last stage, the evaluation of the necessary DFT values and the DFT into DCT conversion is performed by a simple matrix-vector product. The required multiplications after the PTs by the cos- and sin-values for the DFT/DCT conversion and the required additions for one output can be executed as inner products. The inner products are realized with distributed arithmetic (DA) using the modifiered ROM accumulator modules (MRACs) presented in [2]. The structure of an MRAC is shown in figure 2. The outputs of the MRACs for the odd and even values are the DCT coefficients of one row. The row elements are written into output registers and read out in natural order.

The input and the output values are read in and read out in a word-serial/bit-parallel fashion. The interconnections between the three stages are word-parallel/bit-serial. The adders for the polynomial reduction, the PT computation and the DA computation work bit-serial.

2.2 Architecture using the 2D WHT

In this section we derive a parallel architecture using the two-dimensional Walsh-Hadamard Transfom (2D WHT).

Let X_8 and Y_8 denote the properly scaled and normalized two-dimensional input data and its WHT of size 8×8 , respectively. Then the WHT is defined as:

$$Y_8 = WH_8 X_8 WH_8^T \quad (21)$$

where WH_8 and WH_8^T is the 8×8 Walsh-sorted Hadamard matrix and its transposed, respectively.

The natural-sorted Hadamard matrix has a recursive structure, such as

$$H_8 = \begin{pmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{pmatrix} \quad (22)$$

We can translate the Walsh-sorted-Hadamard-matrix into the natural-sorted-Hadamard-matrix by using the bitreversal on both the rows and columns, and by sorting the input data in the permutated way as shown in figure 3. The subscript BRO indicates bit reversal in both the rows and columns. We use the BRO in eq. (21) and partition Y_8 in 4 submatrices of size 4×4. So we can compute the 2D WHT of size 8×8 in the following way:

$$Y_{8_{BRO}} = \begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix} = \begin{pmatrix} H_4 & H_4 \\ H_4 & -H_4 \end{pmatrix} \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} \begin{pmatrix} H_4^T & H_4^T \\ H_4^T & -H_4^T \end{pmatrix} \quad (23)$$

and expanded:

$$Y_{11} = H_4\cdot (X_{11} + X_{12} + X_{21} + X_{22})\cdot H_4^T \quad (24a)$$

$$Y_{12} = H_4\cdot (X_{11} - X_{12} + X_{21} - X_{22})\cdot H_4^T \quad (24b)$$

$$Y_{21} = H_4\cdot (X_{11} + X_{12} - X_{21} - X_{22})\cdot H_4^T \quad (24c)$$

$$Y_{22} = H_4\cdot (X_{11} - X_{12} - X_{21} + X_{22})\cdot H_4^T \quad (24d)$$

The aim is to compute the 2D DCT via the 2D WHT. The 2D DCT of the input matrix X_8 is given by eq. (2). The relationship between the 2D WHT and 2D DCT is given by:

$$Z_8 = CM_8 Y_8 CM_8^T \quad (25)$$

where CM_8 is a conversion matrix, which is obviously given by

$$CM_8 = C_8 H_8^T \quad (26)$$

Using the BRO, CM_8 has the following diagonal structure:

$$CM_{8_{BRO}} = \begin{pmatrix} CM_{4_{BRO}} & 0 \\ 0 & RLH_4 \end{pmatrix} \quad (27)$$

where RLH_4 is the right lower half of $CM_{8_{BRO}}$ [7].

Using the BRO in equation (25), we achieve

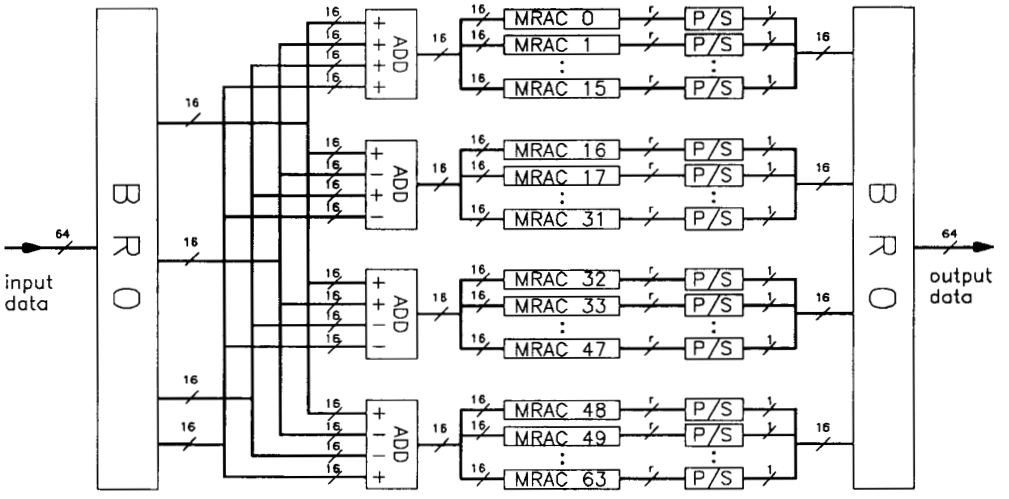

Figure 4: Block structure of the 8x8 DCT using WHT

$$Z_{8_{BRO}} = CM_{8_{BRO}} Y_{8_{BRO}} CM^{T}_{8_{BRO}} \tag{28}$$

This equation can be partitioned into 4 submatrices. If we define the BRO sorted output matrix as

$$Z_{8_{BRO}} = \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix} \tag{29}$$

then we can compute the submatrices by the following equations:

$$Z_{11} = CM_{4_{BRO}} \cdot H_4 \cdot (X_{11} + X_{12} + X_{21} + X_{22}) \cdot H_4^T \cdot CM_{4_{BRO}}^T \tag{30a}$$

$$Z_{12} = CM_{4_{BRO}} \cdot H_4 \cdot (X_{11} - X_{12} + X_{21} - X_{22}) \cdot H_4^T \cdot RLH_4^T \tag{30b}$$

$$Z_{21} = RLH_4 \cdot H_4 \cdot (X_{11} + X_{12} - X_{21} - X_{22}) \cdot H_4^T \cdot CM_{4_{BRO}}^T \tag{30c}$$

$$Z_{22} = RLH_4 \cdot H_4 \cdot (X_{11} - X_{12} - X_{21} + X_{22}) \cdot H_4^T \cdot RLH_4^T \tag{30d}$$

These equations lead to a fully parallel architecture, which is shown in figure 4. The whole input matrix is readin in bit-serial order. The adder blocks consist of 16 adders, respectively, in which the additions/subtractions are done using bit-serial adders, according to equation (30). The necessary multiplications are carried-out as inner product computations. This computations take place in MRACs, refer to the architecture of section 2.1. The outputs of the MRACS form the output matrix, which is read out bit-serial after parallel/serial conversion.

3. Comparison of the architectures

The main element to calculate one frequency domain coefficient is the MRAC. A recent simple carry-ripple adder design of this element shows that a MRAC with 8 inputs can be implemented with about 1700 transistors and a MRAC with 16 inputs with about 3000 transistors. The architecture using polynomial transforms needs only 24 MRACs of size 16 and 48 MRACs of size 8, against 64 MRACs of size 16 used in the second architecture. But it has a more complex structure and far more adders are needed.

Both proposed architectures are based on bit-serial computations. An output can be read out after r_{max} clock cycles. If we assume that the internal word length r_{max} is 12 bit, then every 12 cycles the MRACs will compute all 64 DCT values. Therefore the pixel rate is limited by

$$F_P = \frac{64}{12} F_T \tag{31}$$

with pixel rate F_P and system clock rate F_T.

Recent simulations have shown that the above-mentioned ripple adder architecture for the accumulator leads to a delay time of about 76 ns in the worst case (typ. 35 ns). The accumulator was designed as a standard cell circuit in a 1.5 µm CMOS technology. So, a delay time of about a half should be possible in an advanced CMOS technology and a careful full-custom design. In this case, the architectures are suited for HDTV pixel rates up to 144 MHz in the worst case (typ. 300 MHz).

4. Summary and conclusuions

In this paper two novel parallel architectures for DCTs are presented. It is shown, that two very different algorithms lead to structures of comparable complexity and speed. The architectures are well-suited for VLSI implementation because they have a high degree of regularity and modularity and need no multipliers. Both architectures meet the requirements for HDTV pixel rates, even if progressive scan (144 MSamples/s) is needed.

References

[1] K.R. Rao, P. Yip, "Discrete Cosine Transform", Academic Press Inc., San Diego, 1990

[2] S. Wolter, D. Birreck, R. Laur, "Classification for 2D-DCT's and a new architecture with Distributed Arithmetic", Proc. IEEE ISCAS 1991, June 1991, Singapore, pp. 2204-2207

[3] S. Wolter, D. Birreck, "2D-DCTs: A Classification and a New Architecture Suited For VLSI", 4th International Workshop on HDTV and beyond, Sept. 1991, Turin, Italy, Proceedings, Vol. I

[4] H.J. Nussbaumer, P. Quandalle, "Fast Computation of Discrete Fourier Transform using Polynomial Transforms", IEEE Trans. on ASSP, vol. 27, 1979, pp. 169-181

[5] M. Vetterli, "Fast 2-D Discrete Cosine Transform", Proc. IEEE ICASSP 1985, March 1985, pp. 1538-1541

[6] P. Duhamel, C. Guillemot, "Polynomial Transform Computation of the 2D-DCT", Proc. IEEE ICASSP 1990, pp. 1515-1518

[7] B. G. Kashef, A. Habibi, "Direct computation of high order DCT coefficients from low order DCT coefficients", 28th SPIE Annual Int. Tech. Symp., San Diego, CA, 1984

[8] N. I. Cho, S. U. Lee, "Fast Algorithm and Impelementation of 2-D Discrete Cosine Transform", IEEE Trans. on Circuits and Systems, Vol. 38, No. 3, March 1991