# ORTHOGONAL APPROXIMATION OF THE DISCRETE COSINE TRANSFORM

Peter Rieder, Jürgen Götze, Matthias Sauer and Josef A. Nossek

*Institute for Network Theory and Circuit Design, TU Munich, Germany*

**Abstract**

The efficient implementation of the discrete cosine tansform is discussed in this paper. The architecture, that is used, is a systolic processor array consisting of orthonormal rotations. The angles of these rotations are denoted by $\beta_{ij}$. With respect to a simple VLSI implementation an approximation of the DCT is realized. The approximation is obtained by using approximate (orthogonal) rotations. I.e. the exact rotations ($\beta_{ij}$) are replaced by approximate rotations ($\tilde{\beta}_{ij}$), whereby a rotation over $\tilde{\beta}_{ij}$ can easily be implemented using simple shift and add operations. The use of approximate rotations guarantees an efficient implementation of each rotation and, therefore, for the whole transform. The orthogonality of the transform is preserved, and therefore, also the possibility of perfect reconstruction. The coefficients of the transform matrix are approximated to a high accuracy, such that the difference to the exact DCT can be neglected with respect to practical applications.

## I INTRODUCTION

The Discrete Cosine Transform (DCT) [5, 8] is the core of many signal processing algorithms. This is due to the fact, that this transform seems to be nearly optimal to decorrelate certain signals. Therefore, the DCT has its firm place in image processing [6]. An image is usually divided in subblocks of size 8x8 or 16x16, which are then transformed by the DCT. The transformed subblocks are quantized and coded, before being sent to the receiver, where the image is restored by using the inverse Discrete Cosine Transform (IDCT). With the advent of High Definition TV (HDTV) and its large amount of image data, it is more important than ever to compress the data without loosing too much information. In addition HDTV demands processing of the image data in real time necessitating an efficient implementation of the used algorithm.

In this paper an architecture for the DCT using a systolic processor array is introduced [1]. This array can also be used for any other orthogonal signal transform. The systolic array performs the transform by $n(n-1)/2$ orthogonal rotations. Each processor must execute an orthogonal rotation. Implementing a fast DCT would be an alternative, but less flexible. For the presented approach not the overall implementation of the transform is of primary interest, but the realization of the orthonormal rotation in each processor.

In order to obtain a simple implementation of these rotations, the exact rotations are approximated by a sequence of $\mu$–rotations [7]. Thereby, the exact angle can be approximated to an arbitrary high accuracy, whereby orthonormality is preserved independent of the accuracy. The $\mu$–rotations can be executed with shift- and add operations and, therefore, guarantee a simple implementation of the whole approximate rotation. There are three classes of simple $\mu$–rotations [3]. One class includes the CORDIC rotations [2, 9, 10], which find applications in many fields. With the three classes of rotations flexibility is offered, which must be exploited in order to find the best approximation with the least computational effort. There is not only the possibility of implementing the DCT, but the presented approach can be applied to other orthogonal signal transforms (e.g. orthogonal wavelet transforms [7]) or to matrix decompositions (e.g. the eigenvalue decomposition[3, 4]).

This paper is organized as follows: Section 2 deals with the systolic processor arrays, the architecture being used to parallelize the orthogonal transform. Section 3 introduces the three classes of $\mu$–rotations, that are used for the approximation of the exact rotation angles. Section 4 gives an example of an approximated simple implementable DCT of size 8x8, as used in image processing. After comparing the approximated DCT with the exact in section 5 the results are summarized.

## II SYSTOLIC PROCESSOR ARRAY FOR IMPLEMENTING THE DCT

The systolic processor array we use (Figure 1), was already applied for the solution of dense systems of

linear equations by Bojanczyk et al. [1]. With this array the $\boldsymbol{QR}$–decomposition of any given matrix can be parallelized. For an $n$x$n$ matrix $\frac{n(n-1)}{2}$ processors are necessary, which all execute an othonormal rotation. Of course, also the DCT transform matrix $\boldsymbol{C}$ can be decomposed with the array, whereby the rotation angles $\beta'_{ij}$ $(1 \leq j \leq n-1,\ j < i \leq n)$ are needed. Because of the orthonormality, $\boldsymbol{C}$ is decomposed to a matrix $\boldsymbol{Q}$ and the matrix $\boldsymbol{R}$, which is the identity matrix.

$$\boldsymbol{C} = \boldsymbol{QR} = \boldsymbol{QI}.$$

Processing the matrix $\boldsymbol{C}$ with the now known rotations $\beta_{ij}$ leads to the identity matrix.

$$\boldsymbol{Q}^T\boldsymbol{C} = \boldsymbol{I} \Rightarrow \boldsymbol{Q} = \boldsymbol{C}.$$

Processing a signal vector $\boldsymbol{s}$ with the array based on the known rotations $\beta'_{ij}$ can be interpreted as matrix vector multiplication or as inverse discrete cosine transform IDCT, what leads to the transformed vector $\boldsymbol{t}$.

$$\boldsymbol{Q}^T\boldsymbol{s} = \boldsymbol{t} \Rightarrow \boldsymbol{C}^T\boldsymbol{s} = \boldsymbol{t}$$

Each orthonormal matrix vector multiplication can be done with $\frac{n(n-1)}{2}$ rotations. While the rotations $\beta'_{i,j}$, leading to the IDCT, can be found by the $\boldsymbol{QR}$–decomposition of $\boldsymbol{C}$,of course, the rotations $\beta_{i,j}$ for the DCT can be found by $\boldsymbol{QR}$–decomposition of the matrix $\boldsymbol{C}^T$. The angles $\beta_{i,j}$ and $\beta'_{i,j}$ are the same, only the indices differ.

## III CLASSES OF ORTHONORMAL ROTATIONS

Let an orthonormal $\mu$–rotation be defined by the matrix $\boldsymbol{W}$, which is the product of an orthogonal rotation $\boldsymbol{W}'$ and a scaling factor $r$.

$$\boldsymbol{W} = r\boldsymbol{W}' = r \begin{pmatrix} c' & \mp s' \\ \pm s' & c' \end{pmatrix},$$

where the scaling factor $r$ is given by

$$r = \frac{1}{\sqrt{c'^2 + s'^2}}.$$

The angle $\alpha_i$ of this $\mu$–rotation is computed by

$$\alpha_i = \pm arctan(\frac{s'}{c'})$$

With respect to an efficient implementation, we must choose the coefficients $c'$ and $s'$ as simple as possible. Also the scaling factor $r$ must be implemented with a small number of shift and add operations, in order to obtain an efficient orthonormal $\mu$–rotation. The price we have to pay for the simplicity of the $\mu$–rotations is, that only a limited (finite) set of $\mu$–rotation angles is available. There are several possibilities, leading to several classes of such simple $\mu$–rotations. These were also shown in [3, 7].

Class $I$ is one $\mu$–rotation of the CORDIC [2, 9, 10]. The $\mu$–rotations of this class are given by

$$\boldsymbol{W}_i^I = \frac{1}{\sqrt{1+2^{-2i}}} \begin{pmatrix} 1 & \mp 2^{-i} \\ \pm 2^{-i} & 1 \end{pmatrix}.$$

With these $\mu$–rotations the angles $\alpha_i^I = \pm arctan(2^{-i})$ can be realized.

By adding coefficients to the entries of the rotation matrix $\boldsymbol{W}_i^I$, it is possible to realize other classes of $\mu$–rotations. Thereby, additional angles and a simpler implementation of the scaling factor can be guaranteed. Class $II$ is given by

$$\boldsymbol{W}_i^{II} = \frac{1}{\sqrt{1+2^{-4i-2}}} \begin{pmatrix} 1-2^{-2i-1} & \mp 2^{-i} \\ \pm 2^{-i} & 1-2^{-2i-1} \end{pmatrix}.$$

The angles $\alpha_i^{II} = \pm arctan(\frac{2^{-i}}{1-2^{-2i-1}})$ can be realized with these $\mu$–rotations.

Class $III$ is based on $\boldsymbol{W}_i^{III}$, where for smaller $i$ the difference to the standard rotations (class $I$) are more significant than for larger $i$:

$$\boldsymbol{W}_i^{III} = \frac{1}{\sqrt{1+2^{-6i-6}}} \begin{pmatrix} 1-2^{-2i-1} & \mp 2^{-i} \pm 2^{-3i-3} \\ \pm 2^{-i} \mp 2^{-3i-3} & 1-2^{-2i-1} \end{pmatrix}.$$

With these $\mu$–rotations only the special rotation angles $\alpha_i^{III} = \pm arctan(\frac{2^{-i}-2^{-3i-3}}{1-2^{-2i-1}})$ can be realized.

A simple implementation of the orthogonal rotation is guaranteed for all 3 classes, as the matrix entries are computable only with shift and add operations. In order to avoid calculating the square root in the scaling factor, two $\mu$–rotations with the same scaling factor $r$ must be combined. With the 3 classes one is not compelled to use the same $\mu$–rotation twice (as in [4]), as there is the possibility of combining rotations of different classes. By the factorization of the scaling factors $r$, the whole transform can be implemented with a small number of shift and add operations.

$$\frac{1}{1+2^{-2k}} = (1-2^{-2k})(1+2^{-4k})(1+2^{-8k})\ldots \quad (1)$$

Of course, $k$ depends on the class of $\mu$–rotation, i.e.
for class $I$, $k = i$,
for class $II$, $k = 2i+1$,
for class $III$, $k = 3i+3$.

## IV IMPLEMENTATION OF THE DISCRETE COSINE TRANSFORM WITH APPROXIMATE ROTATIONS

The DCT requires $n(n-1)/2$ rotations, whereby any angle $\beta$ could be needed. Therefore, many multiplications could be necessary in order to implement the exact rotations. In section 3 methods were presented, how special $\mu$–rotations can be implemented only with shift and add operations. As far as the DCT is concerned, the necessary rotations of the systolic processor array can be approximated with a product of the discussed rotations $\boldsymbol{W}_i^I$, $\boldsymbol{W}_i^{II}$ and $\boldsymbol{W}_i^{III}$. The exact angle $\beta$ is approximated by a sum of angles $\alpha_i^C$ ($C \in \{I, II, III\}$) what leads to $\tilde{\beta}$, whereby the difference $\Delta\beta$ has to be minimized as well as the necessary amount of angles $\alpha_i^C$.

$$\beta = \tilde{\beta} + \Delta\beta = \sum \pm\alpha_i^C + \Delta\beta \qquad C \in \{I, II, III\}$$

As an example, we show the approximation not for all 28 rotations of Figure 1, but only for $\beta_{31} = 35.2644^o$. With the following approximation, the difference to the exact angles can be reduced to $\Delta\beta_{31} = 0.1851^o$ using the presented angles of section 3. Here only a few simple $\mu$–rotations are needed

$$\beta_{31} = \alpha_3^I + \alpha_1^{II} - \alpha_5^{III} + \Delta\beta_{31} = 35.0793^o + 0.1851^o.$$

As far as the scaling factors of the $\mu$–rotations are concerned, the scaling factors of $\alpha_3^I$ and $\alpha_1^{II}$ are combined to $1/(1+2^{-6})$. If we suggest a wordlength of 32 bits, the rotation $\alpha_5^{III}$ does not produce a scaling factor, because $\frac{1}{\sqrt{1+2^{-36}}} < \frac{1}{1+2^{-32}}$. The scaling factors are factorized according to (1). The resulting structure of Figure 2, which realizes a rotation with $\tilde{\beta}_{31}$, is a good approximation to the exact rotation with $\beta_{31}$. All rotation angles $\beta_{ij}$ are approximated in a similar way, leading to a good approximation of the whole DCT. The more $\mu$–rotations are used, the better the accuracy of the approximated DCT. But of course, with the number of $\mu$–rotations also the computational complexity increases. As far as the computational complexity of our example is concerned, the implementation of the approximate rotation ($\tilde{\beta}_{31}$) needs 18 shift and add operations, while the exact rotation $\beta_{31}$ needs 128 shift and add operations (we consider, that one rotation consists of 4 multiplications, and with a wordlength of 32 bits one multiplication consists of 32 shift and add operations). As the computational efforts of all other rotations are similar to the presented example, also the relation between exact and approximate cosine transform are similar.

## V COMPARISON OF EXACT AND APPROXIMATED DCT

The only item remaining to discuss is the question, how good is the resulting approximated DCT. The reason for doing a DCT in image processing is, that this transform is nearly optimal for data compression. For our tests, the Lena image of size 224x224 is divided in 784 blocks of size 8x8. The blocks are cosine transformed with the exact $\boldsymbol{C}$ and with the approximated $\tilde{\boldsymbol{C}}$ and the energy compaction of both transforms are compared. The norms of the 2x2 and 4x4 blocks, containing the most significant low pass information are analyzed (upper left in the transformed domain, Figure 3 right). $E_2$ is the norm of the 2x2 block after the exact DCT, $\tilde{E}_2$ is the norm of the 2x2 block after the approximated DCT, $E_4$ is the norm of the 4x4 block after the exact DCT and $\tilde{E}_4$ is the norm of the 4x4 block after the approximated DCT. As the mean values of the ratios $m_2 = \tilde{E}_2/E_2 = 0.9997$ and $m_4 = \tilde{E}_4/E_4 = 0.9998$ are nearly 1, the energy compaction of the approximated DCT is almost as good as the energy compaction of the exact DCT. Figure 3 shows for the 784 imagesubblocks the deviations of $\tilde{E}_2/E_2$ and $\tilde{E}_4/E_4$ from 1. As there are negative values for $1-\tilde{E}_2/E_2$ and $1-\tilde{E}_4/E_4$ (Figure 3), it can be seen, that for many imagesubblocks the energy compaction is even better for the approximated DCT than for the exact DCT. In general, the energy compaction of the approximated DCT is almost the same as the energy compaction of the exact DCT, supporting the use of the simple implementable approximated DCT.

## VI CONCLUSION

In this paper a method was presented for approximating the DCT in order to enable a simple implementation, realizable only with shift and add operations. A sequence of different classes of $\mu$–rotations is used to approximate the exact rotations, that appear in the systolic array realizing the discrete cosine transform. These different classes enable an accurate approximation of all angles with a small number of $\mu$–rotations. The better the accuracy should be, the more $\mu$–rotations are required. While orthonormality is preserved exactly, there are small deviations between the coefficients of the transform matrices $\boldsymbol{C}$ and $\tilde{\boldsymbol{C}}$. It can be shown, however, that for practical applications, these differences can be neglected. The presented approximation method can be extended to other architectures, implementing the DCT. The architectures for the fast DCT, that are also based on orthonormal rotation blocks, can be implemented using approximate rotations. The systolic array is used for the discussions in this paper, since in this way the de-

rivations extend to all other orthonormal signal transformations.

## References


[1] A. Bojanczyk, R.P. Brent, and H.T. Kung. Numerically Stable Solution of Dense Systems of Linear Equations Using Mesh-Connected Processors. *SIAM J. Sci. Stat. Comput.*, 1(1):95–104, 1984.

[2] E.F. Deprettere, A.A.J. De Lange, and P. Dewilde. The Synthesis and Implementation of Signal Processing Applications Specific VLSI CORDIC Arrays. *Proc. Int. Conf. Acoustics Speech and Signal Processing*, pages 974–977, 1990.

[3] J. Götze and G.J. Hekstra. Adaptive Approximate Rotations for Computing the EVD. In M. Moonen and F. Cathoor, editors, *Algorithms and Parallel VLSI Architectures*. Elsevier Science Publishers, 1994.

[4] J. Götze, S. Paul, and M. Sauer. An Efficient Jacobi-like Algorithm for Parallel Eigenvalue Computation. *IEEE Trans. on Computers*, 42(9):1058–1065, 9 1993.

[5] H.S. Hou. A Fast Recursive Algorithm for Computing the Discrete Cosine Transform. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-35(10):1455–1461, October 1987.

[6] A.K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, 1989.

[7] P. Rieder, J. Götze, and Josef A. Nossek. Orthogonal Approximation of Discrete Wavelet Transforms. *submitted to Trans. on Circuits and Systems*.

[8] S.K. Tang, S.C. Chan, K.L. Ho, and F.K. Lam. Implementation of Fast Cosine Transform on the Motorola DSP 96002 Digital Signal Processor. *IEEE Int. Symp. on Circuits and Systems*, pages 73–76, April 1991.

[9] J.E. Volder. The CORDIC trigonometric computing technique. *IRE Trans. Electron. Comput.*, EC-8:330–334, 59.

[10] J. Walther. A Unified Algorithm for Elementary Functions. *Joint Comput. Conf. Proc.*, 71.


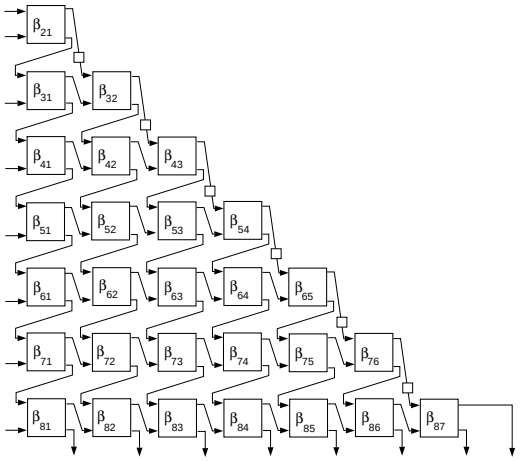


Figure 1: Systolic processor array implementing a DCT

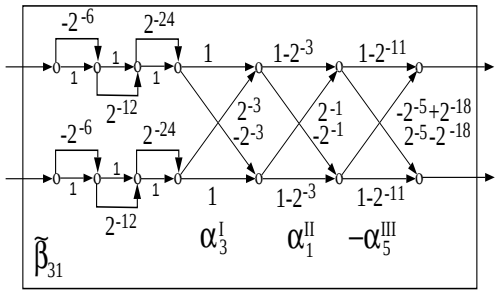


Figure 2: Structure of one processor implementing one rotation of the transform

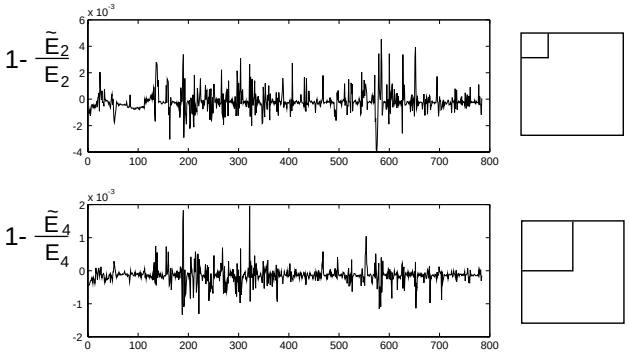


Figure 3: Comparison of the energy compaction of the exact and the approximated DCT