A Power Efficient Implementation of the Discrete Cosine Transform

Christian V. Schimpfle, Peter Rieder and Josef A. Nossek
Institute for Network Theory and Circuit Design
Munich University of Technology
Arcisstr. 21, 80290 München, Germany
chsc@nws.e-technik.tu-muenchen.de

Abstract

In this paper a new efficient implementation of a IEEE-Standard conform 8 point discrete cosine transform (DCT) is presented. The architecture is based on different classes of orthogonal 2 × 2 μ-rotations used to approximate the angles of the DCT. By using only orthogonal μ-rotations it is guaranteed, that the whole transform remains orthogonal and perfect reconstruction of the signal can be achieved. It is shown that for the implementation of the DCT with approximated rotation angles (angle quantization) about 28% less shift and add operations are necessary than for a standard conform implementation with coefficient quantization. This leads to a large power benefit due to less adder hardware and less capacitive load of the global interconnects. Besides this, there are some other advantageous aspects concerning area and delay. To support the full custom design of the layout, module generators for all the different classes of μ-rotations can be used to generate the necessary rotations automatically.

1. Introduction

The Discrete Cosine Transform (DCT) [5],[11] has wide application in a large variety of image and signal processing algorithms [6]. Due to its property of decorrelating certain signals almost optimal, the DCT is the basis of many image processing systems. Mostly the image is splitted in blocks of 8×8 or 16×16 pixels each before it is transformed by the DCT. The transformed blocks are then quantized and coded and sent to the receiver, where the original image is restored by the inverse Discrete Cosine Transform (IDCT). By using the DCT for image coding, large compression rates can be achieved, which is important for all practical applications.

On the other hand, an efficient implementation in terms of processing speed, power consumption, chip area and hence of the hardware effort is necessary. Especially the reduction of the power consumption has become one of the most important aspects of circuit design in the recent years due to increasing device density and the demand for complex portable systems.

In this paper the fast DCT (FDCT) architecture introduced by Chen [2] is used to implement the algorithm. This architecture is based on the matrix decomposition of the transform matrix $\boldsymbol{C}_N$ where N represents the size of the $N \times N$ blocks. It is shown that the whole transform can be decomposed into orthogonal 2×2 rotations which are then approximated by angle quantization. Each exact rotation is approximated by a sequence of different classes of orthogonal μ-rotations. Thereby the orthogonality of the whole transform is preserved and perfect reconstruction of the original signal is possible. Only two different classes of μ-rotations, described in section 3, are used for the implementation presented here. The μ-rotations of all classes are in a VLSI suited form where only shift and add operations are necessary. The implementation of the 8×8 DCT presented in this paper is conform to the IEEE standard specifications for the implementations of 8×8 discrete cosine transforms [1].

This paper is structured as follows: In Section 2 the FDCT architecture is presented. Section 3 introduces the different classes of μ-rotations, used to approximate the exact rotation angles. In Section 4 the optimized signal flow graph of the complete architecture with approximated rotations is presented. In Section 5 methods for efficient full custom implementation of the whole DCT are discribed and the complete mask layout is shown. In Section 6 some results concerning area and power consumption are given. Finally, some concluding remarks are made in Section 7.

2 The FDCT Architecture

In [2], Chen et. al. present a structured fast algorithm for the computation of the DCT. The method is based on the decomposition of the transform matrix $\boldsymbol{C}_N$. Matrix $\boldsymbol{C}_N$

1058-6393/98 $10.00 © 1998 IEEE

can be written in the following recursive form:

$$\boldsymbol{C}_N = \boldsymbol{P}_N \begin{bmatrix} \boldsymbol{C}_{\frac{N}{2}} & \boldsymbol{0}_{\frac{N}{2}} \\ \boldsymbol{0}_{\frac{N}{2}} & \boldsymbol{R}_{\frac{N}{2}} \end{bmatrix} \boldsymbol{B}_N. \quad (1)$$

$\boldsymbol{P}_N$ describes a permutation matrix of order N which permutates a bit reversed vector in to the correct order. Matrix $\boldsymbol{B}_N$ is defined by:

$$\boldsymbol{B}_N = \begin{bmatrix} \boldsymbol{I}_{\frac{N}{2}} & \boldsymbol{J}_{\frac{N}{2}} \\ \boldsymbol{J}_{\frac{N}{2}} & -\boldsymbol{I}_{\frac{N}{2}} \end{bmatrix}, \quad (2)$$

where $\boldsymbol{I}_N$ is the identity matrix and $\boldsymbol{J}_N$ is the opposit diagonal identity matrix of order N. Matrix $\boldsymbol{R}_N$ is now decomposed into orthogonal 2×2 rotations. For an 8 point DCT ($N = 8$), matrix $\boldsymbol{R}_4$ is decomposed as follows:

$$\boldsymbol{R}_4 = \begin{bmatrix} \cos(\frac{-7\pi}{16}) & & & -\sin(\frac{-7\pi}{16}) \\ & \cos(\frac{-3\pi}{16}) & -\sin(\frac{-3\pi}{16}) & \\ & \sin(\frac{-3\pi}{16}) & \cos(\frac{-3\pi}{16}) & \\ \sin(\frac{-7\pi}{16}) & & & \cos(\frac{-7\pi}{16}) \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & & \\ 1 & -1 & & \\ & & -1 & 1 \\ & & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & 0 \\ & -\sin(\frac{\pi}{4}) & \cos(\frac{\pi}{4}) & \\ & \cos(\frac{\pi}{4}) & \sin(\frac{\pi}{4}) & \\ 0 & & & 1 \end{bmatrix} \quad (3)$$

Fig. 1 shows the resulting SFG for $N = 8$. Note that only orthogonal 2×2 rotations have to be executed. Now

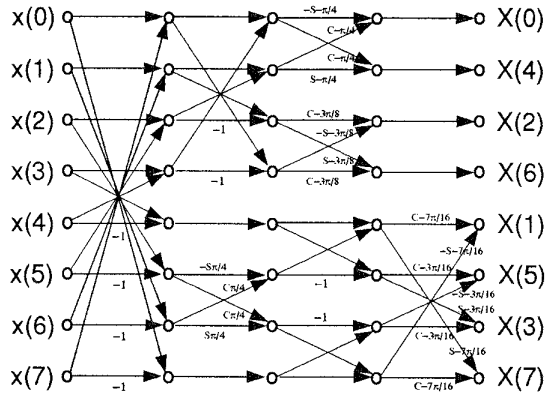

Figure 1. The signal flow graph of a FDCT with $N = 8$.

the rotation angles will be approximated by sequences of easy implementable μ-rotations, instead of quantizising the multiplication coefficients due to the limited wordlength. It will be shown that the hardware effort and the total switching activity of the whole circuit is less for angle quantization than for coefficient quantization.

In the next section, the two different classes of μ-rotations used for angle quantization will be presented.

3 The Different Classes of μ-Rotations

An orthogonal 2×2 rotation can be defined by:

$$\boldsymbol{G} = K\boldsymbol{G}' = K \begin{bmatrix} c & -s \\ s & c \end{bmatrix}, \quad (4)$$

with scaling factor

$$K = \frac{1}{\sqrt{c^2 + s^2}} \quad (5)$$

and the rotation angle

$$\alpha = \arctan(\frac{s}{c}). \quad (6)$$

The exact rotation now has to be decomposed into a product of simple μ-rotations in order to have a smaller number of shift and add operations. There exist three classes of simple μ-rotations [4],[8], only classes I and II are given here, as these two classes are sufficient to approximate the exact DCT in a standard conform manner. With these simple μ-rotations, only discrete angles can be realized. A μ-rotation of class I corresponds to one step of the CORDIC algorithm [3]:

$$\boldsymbol{G}_i^I = \frac{1}{\sqrt{1 + 2^{-2i}}} \begin{bmatrix} 1 & \mp 2^{-i} \\ \pm 2^{-i} & 1 \end{bmatrix} \quad (7)$$

With this class I rotation angles $\alpha_i^I = \pm \arctan(2^{-i})$ can be realized.

The μ-rotations of class II are defined by:

$$\boldsymbol{G}_i^{II} = \frac{1}{\sqrt{1 + 2^{-4i-2}}} \begin{bmatrix} 1 - 2^{-2i-1} & \mp 2^{-i} \\ \pm 2^{-i} & 1 - 2^{-2i-1} \end{bmatrix}, \quad (8)$$

for rotation angles $\alpha_i^{II} = \pm \arctan(\frac{2^{-i}}{1 - 2^{-2i-1}})$. Only simple shift and add operation are necessary to compute orthogonal rotations of both classes.

Now it is possible to replace all rotations in the SFG of Fig. 1 by sequences of the presented μ-rotations.

4 Optimization of the SFG

In the following the architecture of the DCT with approximated rotations used for a full custom VLSI implementation is presented. In order to have a standard conform transform, there were some iterations necessary, where the approximation was refined stepwise. In the first step a coarse approximation with only one rotation per angle was chosen. Than the approximation was refined until the requirements of the standard were fulfilled. Tab. 1 shows a comparison between the exact angles and the approximated angles of the proposed architecture.

The pixels of the considered images have a resolution of 9 bit for ranges between -255 and 256. In the worst case

exact angle	approx. angle	Number of approx. rotations
33.75^o	33.7342^o	4
45^o	45^o	1
67.5^o	67.508^o	3
78.75^o	78.7885^o	3

Table 1. Angle approximations.

the input data is multiplied by a factor $\sqrt{\frac{2}{N}} \cdot \frac{N}{\sqrt{2}}$, which follows from the definition of the DCT [2]. For $N = 8$ this factor is 2.83 which leads to two additional necessary bits to avoid overflow.

Also, due to quantization effects, the data word has to be extended at the least significant part. Tab. 2 illustrates the number, the location and the reason for additional bits in the data word for the architecture with angle quantization and for the version with coefficient quantization. The coefficients are assumed to be realized in canonical signed digit code. Note that the architecture with angle quantization is

Reason	MSB	LSB	angle qu.	coeff. qu.
overflow	x		2	2
quant. effects		x	5	4

Table 2. Wordlength extension due to overflow and quantization effects for angle- and coefficient quantization.

more sensitive for wordlength quantization effects, therefore one bit more is added at the LSB part. From Tab. 2 follows, that wordlengths of 16 and 15 bit have to be chosen for architectures with angle- and coefficient quantization, respectively.

The scaling factor for the approximated rotations is canonical signed digit coded and then factorized in order to get a minimum number of fixed wired shift and add operations. Fig. 2 shows the complete SFG of the standard conform 8 point FDCT with approximated rotations.

5 Hardware Implementation

In this section, the VLSI implementation of the proposed architecture is presented. For this purpose a two's complement number representation was chosen because of its simple implementation. In order to achieve higher throughput rates with a propagation delay $O(1)$, each interconnection must contain at least one timing element. This registers or latches are added at the primary inputs and than moved to the inner interconnections by pipelining. Redundant number systems like carry save or signed digit number representation (SDNR) would be an alternative but would require more hardware and thus more area for implementation. For example carry save arithmetic requires twice the number of adders and interconnections as two's complement carry ripple arithmetic. The additional latches for pipelining only lead to an increase in area by a factor 1.48, but require a larger clock tree. Another effect is that glitching and thus unnecessary power consumption is reduced drastically by pipelining [9], [7].

The cross wiring of the rotations can be avoided by wordwise interleaving of the parallel data streams $X(i)$. Fig. 3 shows the interleaving of two rotations with together four parallel data streams. With this strategy, complex global cross wiring is transvered into simpler local wiring. Another advantage of this method is, that the resulting layout always has a rectangular structure, even when the number of computational steps from the inputs to the outputs are different, e. g. for $X(0)$ and $X(1)$ 8 steps, for $X(2)$ and $X(3)$ 7 steps, for $X(4)$ and $X(7)$ 7 steps and for (5) and $X(6)$ 15 steps, see Fig. 2.

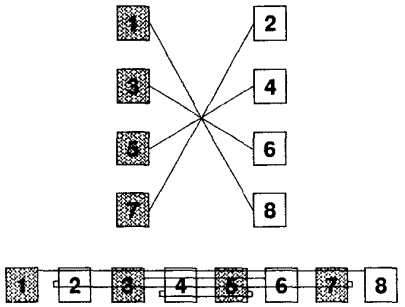

Figure 3. Wordwise interleaving of rotations.

To support the full custom layout process, module generators can be used to generate the layouts of the different rotations automatically. A module generator was developed for parameterized μ-rotations of all classes. A module generator for CORDIC μ-rotations was already presented in [10].

The whole layout is built up in a bit slice architecture. Local wire cells are used for shift and global interconnect wiring. For details see [10]. The shift wire cells can also be generated by a module generator. The complete layout is then generated by simple abutment of the bit slices.

Fig. 4 shows the complete layout implementation of the SFG from Fig. 2. The layout is realized in a $0.7\mu m$ technology on an area of $6.41mm^2$ with a transistor density of 6779.4 transistors/mm^2.

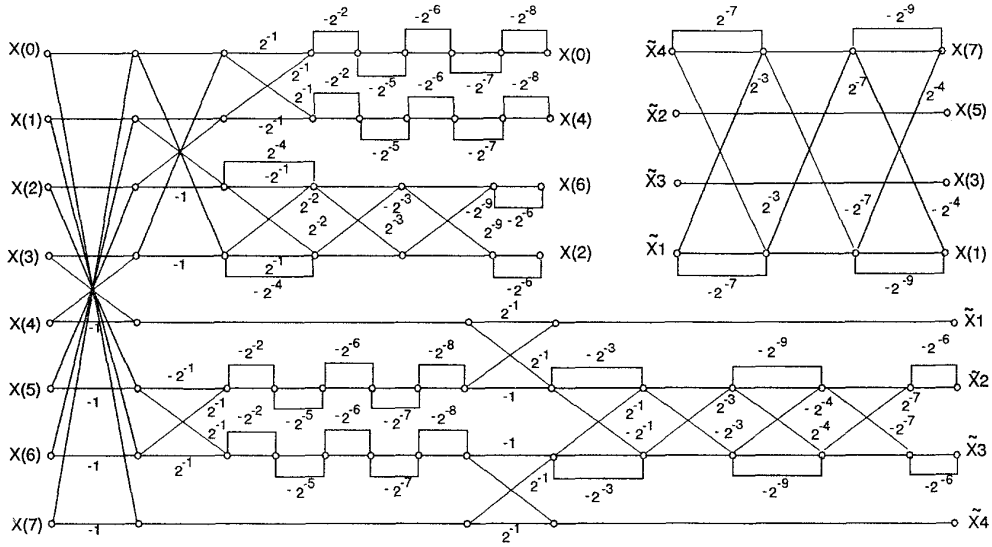

Figure 2. SFG of the FDCT with approximated rotations.

6 Results

To examine the different implementations in terms of power consumption and area, a comparison is made between the standard DCT implementation with coefficient quantization and the proposed implementation with approximated rotation angles.

The complete transform with quantizised coefficients requires 82 shift and add operations, whereas the version with angle quantization requires only 64 shift and add operations. This results in 1230 full adders for the version with coefficient quantization where the wordlength is 15 bit, and 1064 full adders for the version with angle quatization and a wordlength of 16 bit. This means that the standard implementation requires about 20% more adder hardware than the proposed implementation with approximated rotations. Due to a smaller number of shift and add operations, there are also less registers or latches necessary for pipelining. This hardware reductions lead to less overall capacitive load and less average switching power.

	angle quantization.	coeff. quantization
switches per adder	0.516	0.592
power	$17.96\,mW$	$22.46\,mW$

Table 3. Switching activities and power consumption of the compared DCT implementations.

For a detailed examination of the switching activities and thus the switching power consumption of the compared DCT architectures, both implementations were synthesized with VHDL and mapped on a $1\mu m$ library. The simulation results for power consumption and switching activitiies are presented in Tab. 3. The values for the switching activities are the average numbers of switches per adder output and per input pattern change, measured for a sequence of 10000 equally distributed random input patterns. The reduction of the switching activity due to angle quantization is 13% per full adder, the power consumption is reduced by 20%.

7 Conclusion

In this paper a new efficient implementation of the DCT based on a well known fast DCT architecture is presented. The basic idea is to quantize the angles of every 2×2 rotation of the transform by certain classes of 2×2 μ-rotations rather than quantizing the multiplication coefficients. With this method about 28% less shift and add operations are required to implement the whole transform. The presented DCT implementation is conform to the IEEE standard.

In order to achieve high throughput with a minimum cost in area, a full custom design is chosen to realize the architecture in hardware. Special module generators, developed to support the process of manual layouting, lead to a reduction of the design time.

Simulation results show, that the switching activity in the proposed architecture with angle quantization is 13% less per full adder than in a standard architecture with quantized multiplication coefficients. Together with a reduction of the overall capacitance this leads to remarkable power savings.

Future research will focus on the application of the presented methods for other signal transforms as multiwavelet transforms or the LOT.

Acknowledgement

The authors would like to thank Lidmila Barth for preparing the layout and for her untiring patience while carrying out this work.

References

[1] *IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform.* The Institute of Electrical and Electronics Engineers, New York, 1991.

[2] W. H. Chen, C. Harrison Smith, and S. C. Fralick. A Fast Computational Algorithm for the Discrete Cosine Transform. *IEEE Transactions on Communications*, 25(9):1004–1008, September 1977.

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

[4] 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*, 1994.

[5] H. S. Hou. A Fast Recursive Algorithm for the Discrete Cosine Transform. *IEEE Transactions 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] J. Leijten, J. van Meerbergen, and J. Jess. Analysis and Reduction of Glitches in Synchronous Networks. *Proc. ED&TC*, pages 398–403, 1995.

[8] P. Rieder, J. Götze, M. Sauer, and J. A. Nossek. Orthogonal Approximation of the Discrete Cosine Transform. *Proc. ECCTD'95*, pages 1003–1006, 1995.

[9] C. V. Schimpfle, S. Simon, and J. A. Nossek. Optimal Placement of Registers in Data Paths for Low Power Design. *IEEE Int. Symposium on Circuits and Systems, Hong Kong*, 1997.

[10] S. Simon, P. Rieder, C. Schimpfle, and J. A. Nossek. CORDIC-Based Architectures for the Efficient Implementation of Discrete Wavelet Transforms. *IEEE Int. Symposium on Circuits and Systems, Atlanta*, pages IV 77–IV 80, 1996.

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

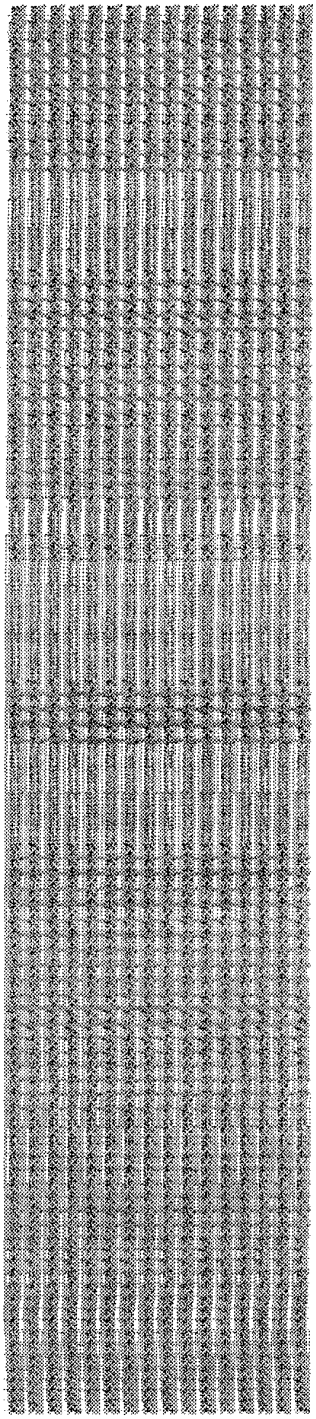

Figure 4. Layout of the 8 point DCT.