Polynomial transform based DCT implementation

Domagoj Babić
advisor: Prof.dr.sc. Mario Kovač

Faculty of Electrical Engineering and Computing of Zagreb University

Abstract. *Discrete Cosine Transform (DCT)* is an important transform of particular interest in still image compression and compression of individual video frames, while multidimensional DCT is mostly used for compression of video streams and volume spaces. An FPGA implementation of a *Polynomial Transform DCT (PTDCT)* algorithm, recently proposed by Zeng et al. [10], is presented. The regularity of Zeng's algorithm and careful operation scheduling have resulted in a very efficient implementation of a two-dimensional DCT in Xilinx Virtex-II FPGA in the terms of logic requirements.

I. Introduction

A class of transforms, called polynomial transforms, has been heavily used for the realization of efficient multidimensional algorithms in digital signal processing. Some of the examples of significant computational savings achieved by using the results from number theory and polynomial transforms include multidimensional discrete Fourier transforms, convolutions and also DCT. The application of polynomial transforms to DCT is not so straightforward as it is the case with discrete Fourier transform and convolutions. A suitable polynomial transform based multidimensional DCT algorithm has emerged very recently and it will be introduced as PTDCT algorithm. According to the best of author's knowledge no hardware implementation has been made. The algorithm has significant advantages in front of other polynomial transform based DCT algorithms because it can be used for M-dimensional algorithms and symmetries in higher dimensions are easily and elegantly expressible. It is optimal if the first dimension is computed via an optimal 1-D DCT algorithm. As polynomial transform can be computed by only using additions, higher-dimensions are implemented with a butterfly-like addition structure, similar to FFT butterfly. PTDCT has simpler addition butterfly than Duhamel's polynomial transform DCT algorithm [3, 7], which has been implemented in FPGA [2]. Additional advantage of PTDCT is that it does not require permutation of output data, since values are already in the correct order. However, simple sequence inversion is required for some rows.

It will be interesting to compare hardware implementations of PTDCT and Duhamel's algorithm. In addition, advantage of PTDCT over classical row-column approach will be scrutinized.

The paper is organized as follows. After explaining the mathematical background of PTDCT, we discuss of the implementation itself. Finally, we analyze implementation results.

II. Polynomial Transform DCT

An algorithm invented by Duhamel and Prado [7] has symmetries on the different stages of the algorithm, which reduce the number of additions by approximately 50%. But the problem was in expressing those symmetries in an elegant way, especially because they used complex polynomial rings. If modified real polynomial ring is used, like in [10], symmetries can be easily expressed in the same way as FFT butterfly symmetries. It is a well-known fact that polynomial transforms

defined modulo $z^N + 1$ can be implemented with a reduced number of additions by using a radix-2 FFT-type algorithm. We will consider two dimensional NxN case of the algorithm, where N is a power of 2.

First, let us use a simple permutation described by Nussbaumer in [5, 6] of input values to obtain $4m$ in the nominator of two-dimensional cosine function parameter.

$$\hat{X}_{k,l} = \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} y_{n,m} \cos\left(\frac{\pi\,(4n+1)\,k}{2N}\right) \cos\left(\frac{\pi\,(4m+1)\,l}{2N}\right) \tag{1}$$

Now, we will split previous equation in two parts 2 and assemble a sequence $B_k(z)$:

$$X(k,l) = \frac{1}{2}\left[A(k,l) + B(k,l)\right] \tag{2}$$

$$B_k(z) = \sum_{l=0}^{N-1} B_{k,l} z^l - \sum_{l=N}^{2N-1} A_{k,2N-l} z^l. \tag{3}$$

By introducing the following permutation and substitution:

$$p(m) = [(4p+1)m + p] \;\; mod\; N \tag{4}$$

$$V_p(j) = \sum_{m=0}^{N-1} y\,(p(m), m) \cos\left(\frac{\pi(4m+1)j}{2N}\right), \tag{5}$$

$B_k(z)$ can be rewritten as:

$$B_k(z) \equiv \sum_{p=0}^{N-1} \sum_{l=0}^{2N-1} V_p\,(l - (4p+1)k)\, z^l \;\; mod\; (z^{2N}+1) \tag{6}$$

$$\equiv \sum_{p=0}^{N-1} \sum_{l=0}^{2N-1} V_p\,(l)\, z^{l+(4p+1)k} \;\; mod\; (z^{2N}+1) \tag{7}$$

$$\equiv \left(\sum_{p=0}^{N-1} U_p(z) \hat{z}^{pk}\right) z^k \;\; mod\; (z^{2N}+1) \tag{8}$$

$$\equiv C_k(z) z^k \;\; mod\; (z^{2N}+1). \tag{9}$$

Fast polynomial transform is based on the properties of polynomial ring and can be computed with butterfly-like addition stage. Previous equations satisfy conditions for application of polynomial transform because it can be easily shown that:

$$\hat{z}^N \equiv 1 \;\; mod\; (z^{2N}+1) \tag{10}$$

$$\hat{z}^{N/2} \equiv -1 \;\; mod\; (z^{2N}+1). \tag{11}$$

Hence, $C_k(z)$ can be decomposed in similar way as DFT in the polynomial ring $z^{2N} + 1$:

$$C_k(z) = \sum_{p=0}^{N/2-1} U_{2p}(z) \hat{z}^{2pk} + \hat{z}^k \sum_{p=0}^{N/2-1} U_{2p+1}(z) \hat{z}^{2pk} \tag{12}$$

$$C_{k+N/2}(z) = \sum_{p=0}^{N/2-1} U_{2p}(z) \hat{z}^{2pk} - \hat{z}^k \sum_{p=0}^{N/2-1} U_{2p+1}(z) \hat{z}^{2pk}. \tag{13}$$

III. PTDCT Implementation

An often used algorithm for implementing DCT is based on *distributed arithmetic (DADCT)* [8, 9]. It can be implemented in FPGA by storing DCT coefficients in ROM tables. The computational accuracy is directly related to the width of ROM word. DCT matrix can be factorized as described in [4]. For an FPGA implementation of a reference 8x8 DCT, used for the evaluation of PTDCT, we have factorized the matrix into two 4x4 matrices. Further factorization does not seem to be worth the effort, because four input dot products are very suitable for implementation in FPGA look-up tables with distributed arithmetic.

Separability property of DCT makes it possible to compute the first dimension, transpose the result and then compute the second. On the other hand, polynomial transform based algorithms treat the problem directly as multidimensional. Therefore, no transposition is needed. Second important advantage of PTDCT is that no multiplications are required for the computation of the second dimension.

Although more regular then previous polynomial transform based algorithms, Zeng's algorithm does require quite complex operation scheduling. Symmetries have been analyzed in *Wolfram's Mathematica* and all elements generated in the same cycle in a particular stage have been grouped together. Iteratively grouping elements from the final stage we have achieved an implementation of second dimension with a very small number of adders (31) and dynamic shift buffers (30).

A. First Dimension

The implementation of the first dimension is relatively simple. First stage is the permutation processor that reorders the input data matrix. Permutation of the input data requires 8 cycles. After permutation the first dimension is computed in parallel using 8 one-dimensional DADCT blocks. This stage requires 10 cycles. Each row is then serialized and represents a sequence.

It has been determined by simulation that the best cost/accuracy factor for 8-bit input is achieved with 11-bit ROM word width 14-bit output from the first dimension.

B. Second Dimension

Final stages are more irregular, so it is simpler to organize computation starting with the final stages. Some stages are decoupled by dynamic shift buffers that are used to permute the sequence and are easily implementable in Virtex FPGAs. In this way sequence can be generated in one order, while the elements can be used in another.

The implementation dataflow diagram is shown in Fig. 1 and 2. Pipeline registers have not been shown in order to maintain the flow as simple as possible. Let us denote individual stages as *F*, *S*, *C* and *X*. Stage *F* is completely regular. The next stage (*S*) is irregular for sequences *S5'* and *S7'*. To minimize the number of adders in the *S* stage, some results from *F* stage, namely sequences *F51*, *F52*, *F71* and *F72*, have to be permuted. Other sequences are simply delayed through FIFO in order to maintain proper timing. Sequence *S6* has to be reversed to be used in *C* stage for computing *C2'* and *C6'* sequences. As *S2* is also required for that computation, it has to be buffered. A simple approach would be to insert FIFOs in all other sequences, but if we note that two halves of the butterfly are not crossed in the later stages, we can avoid FIFO insertion. The other half of the butterfly requires dynamic shift buffers for generating sequences *X0*,*X1*,*X3-5* and *X7*, thus the final results are obtained at the same time.

X sequences represent the final result and some of them require dynamic shift buffers to be reversed in time, while others are simply delayed by shifting through FIFOs.

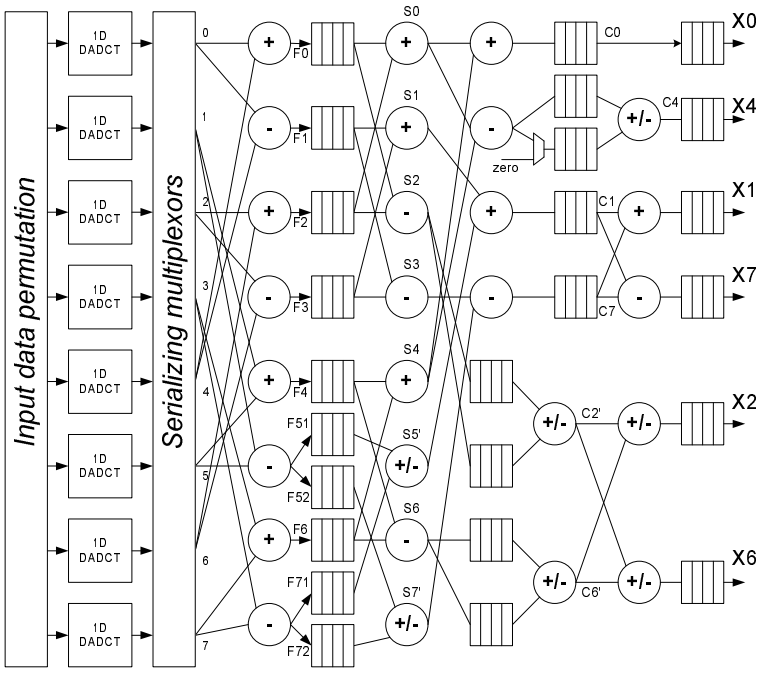

Figure 1: PTDCT dataflow.

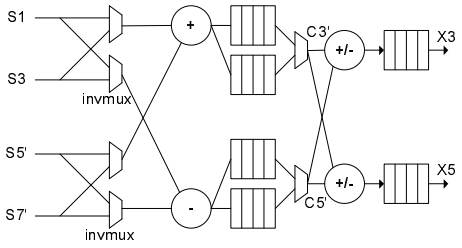

Figure 2: Computation of *X3* and *X5*.

C. Implementation Results

Accuracy of the forward-inverse DCT transformation process influences significantly the quality of multimedia content. The recommendations for the accuracy of 2-D NxN IDCT implementations are standardized by IEEE [1]. PTDCT algorithm has been simulated with *Boats* and *Lena* pictures and the implementation parameters (like the ROM width for the first dimension) have been determined. Results are shown in Table 1.

The design has been synthesized in Virtex-II FPGA device, speed grade -6. Input-output pin insertion was disabled. Although not yet fully optimized, PTDCT implementation of 8x8 DCT required 3949 LUTs, what is less then reported by Dick in [2] for Duhamel's algorithm. His algorithm requires 4496 XC4000 LUTs. The direct comparison is hard because Virtex-II devices have some features that are not available in XC4000 family. In addition, our PTDCT algorithm is not yet fully optimized and control logic is implemented in a very naive manner. Therefore we

Metric	Value obtained by simulation
PPE	1
PMSE	0.00138889
OMSE	0.000477431
PME	0
OME	0.0000385802
PSNR	81.3417 dB

Table 1: MPTDCT implementation accuracy

have implemented classical 2D DADCT algorithm in Virtex-II in 6400 LUTs in order to obtain a referent implementation for a comparison. Our PTDCT requires 37,5% less logic resources than highly optimized DADCT while maintaining the same throughput and working frequency. Even better results are expected after we optimize the implementation and redesign the control logic. An implementation of Duhamel's 8x8 DCT algorithm in XC4000 saved 33% of resources comparing to DADCT implementation in XC4000.

References

[1] IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform, December 1990.

[2] Chris Dick. Computing Multidimensional DFTs Using Xilinx FPGAs. In *The 8th International Conference on Signal Processing Applications and Technology, Toronto Canada*, September 1998.

[3] Pierre Duhamel and C. Guillemot. Polynomial Transform Computation of the 2-D DCT. In *Proceedings IEEE International Conference Acoustics, Speech and Signal Processing*, pages 1515–1518, April 1990.

[4] Ephraim Feig and Shmuel Winograd. Fast algorithms for the discrete cosine transform. *IEEE Trans. on Signal Processing*, 40(9), 1992.

[5] Henri J. Nussbaumer. Fast polynomial transform computation of the 2-D DCT. In *International Conference on Signal Processing*, pages 276–283, 1981.

[6] Henri J. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms*. Springer Verlag, second edition, 1982.

[7] Jacques Prado and Pierre Duhamel. A polynomial-transform based computation of the 2-D DCT with minimum multiplicative complexity. In *Proceedings IEEE International Conference Acoustics, Speech and Signal Processing*, pages 1347–1350, April 1996.

[8] S. White. Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review. *IEEE ASSP Magazine*, pages 4–19, July 1989.

[9] Sungwook Yu and Earl E. Swartzlander. DCT Implementation with Distributed Arithmetic. *IEEE Transactions on Computers*, 50(9):985–991, September 2001.

[10] Yonghong Zeng, Guoan Bi, and A. R. Leyman. New polynomial transform algorithm for multidimensional DCT. *IEEE Trans. Signal Processing*, 48(10):2814–2821, 2000.