OPTIMIZATION AND IMPLEMENTATION ON FPGA OF THE DCT/IDCT ALGORITHM

A. Ben Atitallah[(1,2)], P. Kadionik[(2)], F. Ghozzi[(1)], P.Nouel[(2)], N. Masmoudi[(1)], Ph.Marchegay[(2)]

[(1)] Laboratory of Electronics and Information Technology
National Engineers School of Sfax (E.N.I.S.), BP W 3038 SFAX - TUNISIA
Nouri.Masmoudi@enis.rnu.tn

[(2)] IXL laboratory –ENSEIRB - University Bordeaux1 - CNRS UMR 5818,
351 Cours de la Libération, 33 405 Talence Cedex, France
{benatita;kadionik;nouel;marchegay}@enseirb.fr

ABSTRACT

In this paper, we present a comparison between two methods, the modified Loeffler algorithm (11 MUL and 29 ADD) and Distributed Arithmetic, to implement the DCT/IDCT algorithm for MPEG or H.26x video compression using VHDL description language. The implementation has been achieved on Altera Stratix EP1S10 FPGA which provides a dedicated DSP blocks required for common signal processing functions. A new solution based on this DSP blocks used for to implement multipliers for the modified Loeffler algorithm in order to optimize speed and area.

1. INTRODUCTION

Digital video applications are becoming more popular in our everyday lives. Currently, there are several video standards established for different purposes, such as MPEG-1 [1], MPEG-2 [2] and MPEG-4 [3] for multimedia applications and H.261 [4] and H.263 [5] for videophone and video-conferencing applications. All these standards use the discrete cosine transform (DCT) [6,7], which transforms a signal or image from the spatial domain to the frequency domain. Since the introduction of the DCT in 1974 [8], numerous fast algorithms have been developed. For example Loeffler [9] proposed an efficient 8-point DCT algorithm that requires only 11 multiplications and 29 additions.

In this paper, several efficient architectures for 8 points 1D-DCT are presented. Many researches have already been done to optimize the 1D-DCT computational effort like Lee [10], Chen [11], Loeffler [9] and Ben ayed [12]. Most of the efforts have been devoted to reduce the number of operations, mainly multiplications and additions. However in hardware implementation, not only the effort of calculation has to be optimized but also the silicon area occupation has to be reduced.

This paper is organized as follows. Section 2 begins by introducing the DCT from a theoretical point of view. Section 3 deals with the architecture for computing the DCT with distributed arithmetic. Section 4 describes the modified Loeffler algorithm. Section 5 discusses the methodology of hardware implementation on FPGA. Finally, concluding remarks are presented in section 6.

2. DISCRETE COSINE TRANSFORM

In a transform coding system, pixels are grouped into blocks. A block of pixels is transformed into another domain to produce a set of transform coefficient. Those coefficients represent the spatial frequency components which make up the original block. Each of them can be thought of as a weight which is applied to an appropriate basis function. Typically an image is transformed into blocks of 8x8 pixels. This size of block is optimum for trade-off between compression efficiency and computational complexity. Instead of performing the two-dimensional transform, the same result can be achieved by applying one dimensional transform along all rows and then down the columns of the block [13,14] in order to reduce the number of calculations required. The computational complexity can be further reduced by replacing the transform cosine form with a fast algorithm [9] which reduces the operations to a short series of multiplications and additions. The N-point 1-D DCT is defined as:

$$x_k = \frac{2}{N}\ c(k) \sum_{j=0}^{N-1} y_j \cos\left(\frac{(2j+1)k\pi}{2N}\right), k = 0,1,...,N-1 \quad (1)$$

The N-point 1-D IDCT is defined as:

$$y_j = \frac{2}{N} \sum_{k=0}^{N-1} c(k) x_k \cos\left(\frac{(2j+1)k\pi}{2N}\right), j = 0,1,...,N-1 \quad (2)$$

$$with \quad c(k) = \begin{cases} 1/\sqrt{2} & for\ k = 0 \\ 1 & for\ k \neq 0 \end{cases}$$

3. DISTRIBUTED ARITHMETIC FOR THE 1D-DCT

Distributed Arithmetic (DA) is a technique that allows the hardware implementation of a sum-of-product without using multipliers. By storing first a finite number of intermediate results, a sum-of-product can be obtained through repeated additions and shifting operations without the use of any multiplication. This allows the design of signal processors with a reduction in the gate numbers. In this section, we will have a close look on the arithmetic used for calculating the forward and the inverse DCT. For these transformations on 8x8 pixel blocks distributed arithmetic is used [15-18]. This leads to a bit serial computation where only 16 word look-up tables (ROM) and accumulator but not multipliers need to be utilized. The ROMs with 16 words accessed through a 4-bits address. Below, we give a brief description of the usage of distributed arithmetic for calculating the 1D-DCT.

Based on equation (1), the 8-point 1D-DCT can be expressed as follows:

$$x_k = \frac{c(k)}{2}\sum_{n=0}^{7} y_n \cos\left(\frac{(2n+1)k\pi}{16}\right) = \sum_{n=0}^{7} a_n^k y_n \qquad (3)$$

$$with \quad a_n^k = \frac{c(k)}{2}\cos\left(\frac{(2n+1)k\pi}{16}\right)$$

Utilizing the property of the factors a^k_n to be symmetric in n, we only need to sum up four product terms:

$$x_k = \begin{cases} \sum_{n=0}^{3} a_n^k (y_n + y_{7-n}) for \quad k \quad even \\ \sum_{n=0}^{3} a_n^k (y_n - y_{7-n}) for \quad k \quad odd \end{cases} \qquad (4)$$

For easier writing, we define v_n as a shortcut for the sums and differences of y_n:

$$x_k = \sum_{n=0}^{3} a_n^k v_n$$
$$with \quad v_n = \begin{cases} (y_n + y_{7-n}) for \quad k \quad even \\ (y_n - y_{7-n}) for \quad k \quad odd \end{cases} \qquad (5)$$

We can write the v_n as a sum of weighted bits (note: B is the data width of v_n):

$$v_n = -v_n^{(0)} + \sum_{i=1}^{B-1} v_n^{(i)} 2^{-i} \qquad (6)$$

Then equation (5) and (6) become:

$$x_k = \sum_{n=0}^{3} a_n^k \left(-v_n^{(0)} + \sum_{i=1}^{B-1} v_n^{(i)} 2^{-i}\right) \qquad (7)$$

The above equation can be rewritten as follow:

$$x_k = \sum_{i=1}^{B-1}\left(\sum_{n=0}^{3} a_n^k v_n^{(i)}\right)2^{-i} - \sum_{n=0}^{3} a_n^k u_n^{(0)} \qquad (8)$$

We define:

$$F(a^k, v^{(i)}) = \sum_{n=0}^{3} a_n^k v_n^{(i)} \qquad (9)$$

Then equation (8) and (9) become:

$$x_k = \sum_{i=1}^{B-1} F(a^k, v^{(i)}) 2^{-i} - F(a^k, v^{(0)}) \qquad (10)$$

The equation (10) characterizes the distributed arithmetic scheme in which the initial multiplications are distributed to another computation pattern.

For any index k the $F(a^k,v^{(i)})$ can become precalculated and stored in a ROM with 16 coefficients. So bit-serial evaluation of the summation formula (10) for x_k only requires one "rom and-accumulator" (RAC) element for each k as shown in Figure 1, with preprocessed v_n (sums or differences of y_n and y_{7-n}, n=0..4) as bit-serial input. One 1D-DCT thus consists of eight RACs plus a bit-serial preprocessor which calculates the $v^{(i)}_n$. In Figure 2 you can see the serial preprocessor unit connected with the RAC devices. After B steps (note that B is the data width of v_n), the results of the 1D-DCT appear parallel at x_0 to x_7.

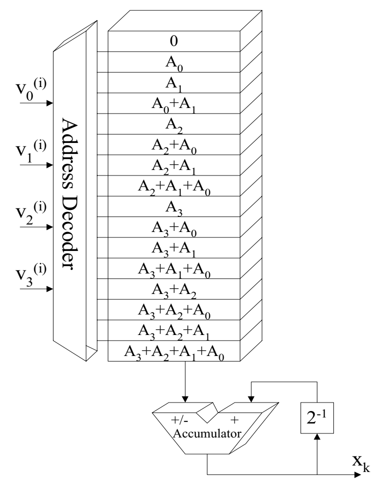


Figure 1. ROM and Accumulator (RAC) Structure

Figure 2 shows a block diagram of a parallel hardware design of an eight point DCT using distributed arithmetic. Two data buses distribute the bits into the two sets of four RAC respectively. After B cycles, the coefficients of the DCT will be computed.

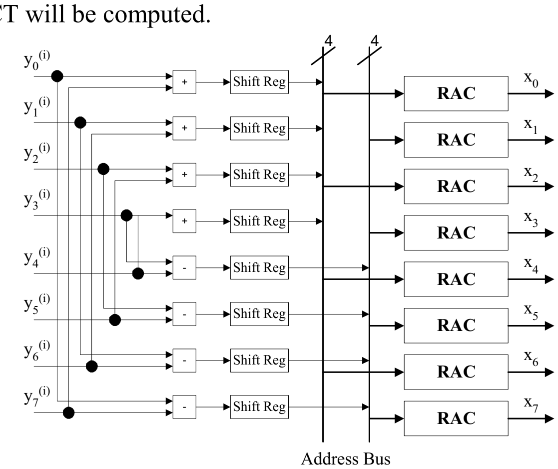


Figure 2. Scheme of the 1D-DCT circuit

4. LOEFFLER ALGORITHM FOR THE 1D-DCT

Based on equation (1), DCT and IDCT are highly computational intensive, which creates prerequisites for performance bottlenecks in systems utilizing them. To overcome this problem, a number of algorithms have been proposed for more efficient computations of these transforms. In our experiments, we use an 8-point 1-D DCT/IDCT algorithm, proposed by van Eijdhoven and Sijstermans [19]. It was selected due the minimum required number of additions and multiplications (11 Mul and 29 add). This algorithm is obtained by a slight modification of the original Loeffler algorithm [9], which provides one of the most computationally efficient 1-D DCT/IDCT calculations [20]. The modified Loeffler algorithm for calculating 8-point 1-D DCT is illustrated in Figure 3.

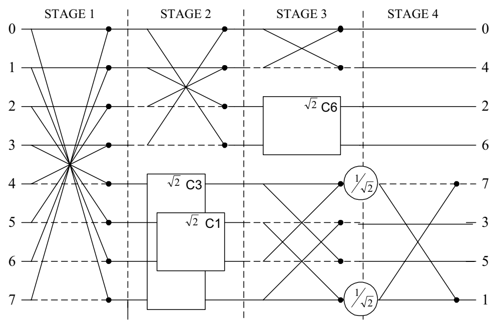
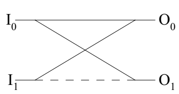


Figure 3. The 8-point DCT modified Loeffler algorithm

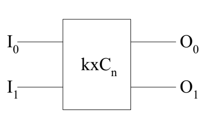
The stages of the algorithm numbered 1 to 4 are parts that have to be executed in serial mode due to the data dependency. However, computation within the first stage can be parallelized. In stage2, the algorithm splits in two parts. One for the even coefficients, the other for the odd ones. The even part is nothing else than a 4 points DCT, again separated in even and odd parts in stage3. The round block in figure 3 signifies a multiplication by $1/\sqrt{2}$. In figure 4, we present the butterfly block and the equations associated.



$$O_0 = I_0 + I_1$$
$$O_1 = I_0 - I_1$$

Figure 4. The Butterfly block and its associated equations

The rectangular block depicts a rotation, which transforms a pair of inputs $[I_0,I_1]$ into outputs $[O_0,O_1]$. The symbol and associated equations are depicted in figure 5.



$$O_0 = I_0 k \cos\left[\frac{n\pi}{16}\right] - I_1 k \sin\left[\frac{n\pi}{16}\right]$$
$$O_1 = -I_0 k \sin\left[\frac{n\pi}{16}\right] + I_1 k \cos\left[\frac{n\pi}{16}\right]$$

Figure 5. The rotator block and its associated equations

The rotator block operation can be calculated using only 3 multiplications and 3 additions instead of 4 multiplications and 2 additions. This can be done by using the equivalence showed in the following equations.

$$\begin{aligned} O_0 &= a.I_0 + b.I_1 = (b-a).I_1 + a.(I_0 + I_1) \\ O_1 &= -b.I_0 + a.I_1 = -(b+a).I_0 + a.(I_0 + I_1) \end{aligned} \qquad (11)$$

This modified Loeffler algorithm is described with VHDL language in order to be implemented on FPGA which is presented in next section.

5. IMPLEMENTATION ON FPGA

5.1. Overview of the STRATIX architecture

The Altera Stratix EP1S10 FPGA is based on 1.5V, 0.13µm technology with a density that reaches 10570 Logic Elements (LEs), 113KB of Embedded System Blocs (ESBs), 48 DSP block and 427 I/O pins [21,22]. An overview of the resources available in a Stratix die is shown in Figure 6.

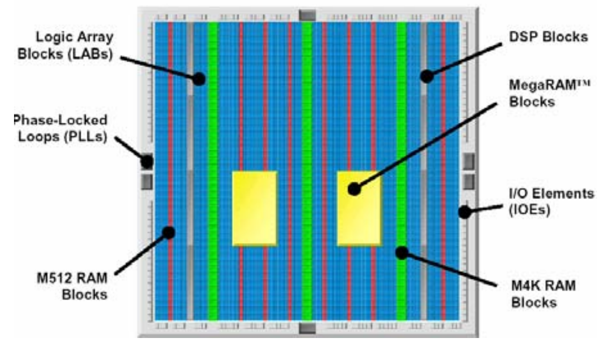


Figure 6. Overview of Stratix Die

Stratix introduces DSP cores for signal processing applications. These embedded DSP Blocks have been optimized to implement several DSP functions with maximum performance and minimum logic resource utilization [23]. The DSP blocks comprise a number of multipliers and adders. These can be configured in various widths to support multiply-add operations ranging from 9x9bits to 36x36bits [21], and including a wide range of operations from multiplication only, to sum of products, and complex arithmetic multiplication. Internal connectivity in the DSP blocks also supports pipeline registers for common DSP building blocks to support applications such as digital filters. In the purpose of optimizing the Loeffler architecture implementation, we can use these DSP blocks to implement multipliers.

5.2. Implementation results

We simulated and synthesized the VHDL source code for Stratix FPGA using Modelsim™ simulator from Model Technology and Quartus II tools from Altera for the synthesis.

Figure 7 and 8 show implementation results for the LEs and the operation frequency of the three methods used for the DCT/IDCT implementation (Loeffler architecture, Loeffler architecture with DSP Blocks and the Distributed arithmetic methods). From these two figures, we can conclude that the implementation of the DCT/IDCT using the modified

Loeffler architecture with the DSP Blocks achieves better performance in term of silicon occupation area (DCT: 5% of the LEs, 45% of the DSP Blocks and IDCT: 4% of the LEs, 45% of the DSP Blocks) and operation frequency (83.33 MHz for DCT and IDCT).

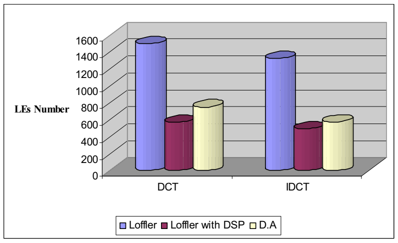


Figure 7. LEs Numbers for the DCT and IDCT with different implementation methods

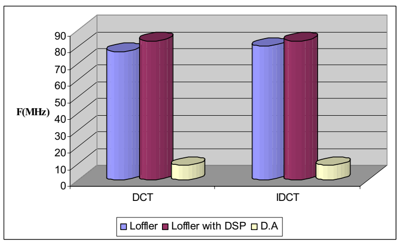


Figure 8. Max Clock Frequency for the DCT and IDCT with different implementation methods

6. CONCLUSIONS

In this paper, we have made an effective comparison between two methods to implement the DCT/IDCT algorithm for video standard. We have presented the modified Loeffler algorithm and the Distributed Arithmetic method for DCT/IDCT implementation. Our design is implemented on Altera Stratix FPGA which provides a dedicated DSP blocks for signal processing functions. We have proposed a new solution to implement multipliers based on these DSP blocks for speed and area optimization. We have shown that better results can be obtained with the modified Loeffler algorithm by using DSP blocks for the DCT/IDCT hardware implementation.

For future work, this design provides an important solution to implement the H.263 standard for video compression.

7. REFERENCES

[1] Standard MPEG-1 : ISO/IEC 11172-2, Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s.

[2] Standard MPEG-2: ISO/IEC 13818-2, Information Technology – Generic coding of moving pictures and associated audio information.

[3] Standard MPEG-4: ISO/IEC 14496-2, Information Technology – Coding of Audio-Visual Objects.

[4] ITU-T Rec. H.261, Video Codec for Audiovisual Services at p_64 kbps, Recommendation. 1993.

[5] ITU-T Rec. H.263, Video Coding for Low Bit Rate communication. 1998.

[6] N.J. August and D.S. Ha, "Low Power Design of DCT and IDCT for Low Bit Rate Video Codecs," *IEEE Trans On Multimedia*, VOL. 6, NO. 3, June 2004.

[7] I.M. Pao and M.T. Sun, "Modeling DCT Coefficients for Fast Video Encoding," *IEEE Trans, On Circuits and Systems for Video Technology*, VOL. 9, NO. 4, JUNE 1999.

[8] N. Ahmed, T. Natarajan and K. R. Rao, "On image processing and a discrete cosine transform," *IEEE Trans, On Computers*, vol. C-23,pp. 90-93, 1974.

[9] C. Loeffler and A. Lightenberg, "Practical fast 1-D DCT algorithms with 11 Multiplications," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP '89),* Scotland, pp. 988-991, May 1989.

[10] Y.P Lee and all "A cost effective architecture for 8x8 two-dimensional DCT/IDCT using direct method," *IEEE Trans. On Circuit and System for video technology*, VOL 7, NO.3, 1997

[11] W.c Chen, C.h Smith and S.C. Fralick, "A fast Computational Algorithm for thr Discrete Cosine Transform,"*IEEE Trans. On Communications*, Vol. COM-25, No. 9, pp.1004-1009, Sept.1997

[12] M.A.Ben Ayed, L.Dulau, P.Nouel, Y.Berthoumieu, N.Masmoudi, P.Kadionik and L.Kamoun, "New Design Using VHDL Description for DCT Based Circuits," *Proceedings of ICM'98*, pp 87-90,1998.

[13] H. Lim, V. Piuri and E. E. Swartzlander, "A Serial-Parallel Architecture for Two Dimensional Discrete Cosine and Inverse Discrete Cosine Transforms," *IEEE Trans On Computers*, VOL. 49, NO. 12, December 2000.

[14] T.S.Chang, C.S.Kung, and C.W. Jen, "A Simple Processor Core Design for DCT/IDCT," *IEEE TRANS On Circuits and Systems For Video Technology*, VOL. 10, NO. 3, APRIL 2000.

[15] Y.H. Chen and W.C. Siu, "On the Realization of Discrete Cosine Transform Using the Distributed arithmetic," *IEEE Trans. On Circuits and Systems*, VOL. 39, NO. 9, Sept. 1992.

[16] S. Yu and E. E. Swartzlander, "DCT Implementation with Distributed Arithmetic," *IEEE Trans. On Computers*, VOL. 50, NO. 9, Sept. 2001

[17] Radhika S. Grover, Weijia Shang and Qiang Li A, "Faster Distributed Arithmetic Architecture for FPGAs," *FPGA'02*, Monterey, California, USA,pp. 31-39, February 24-26 , 2002.

[18] S. Khawam, T. Arslan and F. Westall, "Synthesizable Reconfigurable Array Targeting Distributed Arithmetic for System-on-Chip Applications,"Proceedings of the 18th International Parallel and Distributed Processing Symposium,2004.

[19] T.J. van Eijndhven and F.W. Sijstermans, "Data Processing Device and method of Computing the Cosine Transform of a Mtrix", PCT Patent WO 99948025, to Koninklijke Philips Electronics, World Intellectual Property Organization, International Bureau, 1999.

[20] R. Krishnan, O.P.Gangwal, Jos.v.Eijndhoven and A.Kumar, "Design of a 2D DCT/IDCT application specific VLIW processor supporting scaled and sub-sampled blocks," *Proceedings of the 16th International Conference on VLSI Design*, 2003.

[21] Altera "*Stratix Device Family*" Data Sheet, version 3.1, September 2004.

[22] High-Performance Stratix Architecture "http://www.altera.com/products/devices/stratix/ features/stx-architecture.html".

[23] D.Lewis and Al, "The Stratix™ Routing and Logic Architecture", *FPGA'03,* February 23-25, 2003, Monterey, California, USA.