Low complexity DCT engine for image and video compression

Maher Jridi, Yousri Ouerhani, Ayman Alfalou

Low complexity DCT engine for image and video compression

Maher JRIDI and Yousri OUERHANI and Ayman ALFALOU
Équipe Vision, L@bIsen, ISEN-Brest, CS 42807 Brest, France

ABSTRACT

In this paper, we defined a low complexity 2D-DCT architecture. The latter will be able to transform spatial pixels to spectral pixels while taking into account the constraints of the considered compression standard. Indeed, this work is our first attempt to obtain one reconfigurable multistandard DCT. Due to our new matrix decomposition, we could define one common 2D-DCT architecture. The constant multipliers can be configured to handle the case of RealDCT and/or IntDCT (multiplication by 2). Our optimized algorithm not only provides a reduction of computational complexity, but also leads to scalable pipelined design in systolic arrays. Indeed, the 8×8 StdDCT can be computed by using 4×4 StdDCT which can be obtained by calculating 2×2 StdDCT. Besides, the proposed structure can be extended to deal with higher number of N (i.e. 16×16 and 32×32). The performance of the proposed architecture are better when compared with conventional designs. In particular, for $N = 4$, it is found that the proposed design have nearly third the area-time complexity of the existing DCT structures. This gain is expected to be higher for a greater size of 2D-DCT.

Keywords: DCT, Multistandard, Optimization, FPGA implementation

1. INTRODUCTION

The use of multimedia data (image and video) as well as the widespread adoption of embedded devices have increased significantly in recent years. As a result, numerous compression standards have been proposed and validated according to several applications: MPEG,[1] H264[2] and HEVC[3] for video compression, and JPEG and JPEG XR for image compression. In order to reduce the heterogeneity of decoding devices and converge to a universal decoder, it becomes useful to avoid the design of image processing algorithms for a specific standard and to promote the design for multistandards. More particularly, the Discrete Cosine Transform (DCT) is used for JPEG and MPEG in order to contribute on the reduction of spatial redundancies by transforming the spatial domain in the spectral domain. The direct realization (as opposed to the line-column separation) of the 2D-DCT requires $2N^3$ multiplications and $2N^2(N-1)$ additions, where $N \times N$ is the pixel size of the elemental block to be transformed. In order to alleviate the hardware requirements, new standards like JPEG XR and H264 have adopted the Integer DCT (IntDCT) which has a multiplierless architecture based on Hadamard transform.

In this paper, in order to keep a certain degree of interoperability between different standards, we propose a first attempt of common hardware architecture for the 2D-DCT. We believe that this work is an introduction to a series of future work dedicated to obtain one multistandard DCT circuit. Indeed, the matrix multiplication used to compute the 2D-DCT coefficients is reformulated in order to extract some similarities with the matrices used in Int-2D-DCT. We concluded that it was possible to move from DCT to IntDCT by maintaining a scale factor and by changing one constant in the matrix representation of the IntDCT. Following these decompositions, we defined a new 2D-DCT algorithm and we have named it StdDCT. It is an invertible and a standard adaptive DCT and it has a butterfly-based architecture which is efficient in terms of Area-Delay product. Moreover, StdDCT is a multiplierless architecture when it is used as IntDCT. It can compute the 2D-DCT coefficients with a reduced number of multipliers when it is used as DCT (Real DCT). This number can reach 16 constant multipliers for N=4 instead of 128 with the direct realization. Finally, the above mentioned scale factor can be used in the quantization matrices to obtain the exact values of the transformed and quantized coefficients.

The proposed design is scalable and is validated with Xilinx FPGA implementation for several block sizes of 2×2, 4×4 and 8×8. It is found that the proposed design offers the same performances in terms of latency and throughput when compared with IntDCT. However, when compared with the direct realization of RealDCT,

Further author information: (Send correspondence to Maher JRIDI)
Maher JRIDI: E-mail: maher.jridi@isen.fr, Telephone: +33 (0)2 98 03 84 40

the proposed architecture presents the same latency and involves nearly 87% of saving in hardware which is estimated by evaluating the number of slices. In addition, when compared with the architecture based on the line-column separation, the proposed design offers a significant gain in terms of maximum operating frequency. These comparisons are performed with N=4 and for a greater N ($N > 4$), the gain in frequency and area are higher.

The remainder of the paper is organized as follows: an overview of DCT and fundamental design issues are given in Section 2. The extention of low-complexity DCT and the functional validation of the proposed DCT are described in Section 3. Finally, a signal flow graph of the proposed DCT is proposed in Section 4 before the conclusion.

2. REVIEW OF THE DISCRETE COSINE TRANSFORM

2.1 Definition

The DCT is commonly used in data compression applications due to its high reconstruction capabilities.[4] Indeed, when applied to image and video, the DCT decorrelates each block of input pixels. The energy of the correlated images is packed into the low frequency region (i.e., top left region). Consequently, the Inverse DCT (IDCT) can use the last region to reconstrut the original iamge.
Given an input sequence $\{X(n)\}$, $n \in [0, N-1]$, the N-point DCT is defined as:

$$Y(n) = \sqrt{\frac{2}{N}} C(n) \sum_{k=0}^{N-1} X(k) \cos \frac{(2k+1)\, n\pi}{2N} \tag{1}$$

where $C(0) = 1/\sqrt{2}$ and $C(n) = 1$ if $n \neq 0$.
The 2D-DCT can be represented by matrix multiplication as mentionned in (2):

$$Y = COS \times X \times COS^T = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} \times \begin{pmatrix} X(1,1) & X(1,2) & X(1,3) & X(1,4) \\ X(2,1) & X(2,2) & X(2,3) & X(2,4) \\ X(3,1) & X(3,2) & X(3,3) & X(3,4) \\ X(4,1) & X(4,2) & X(4,3) & X(4,4) \end{pmatrix} \times \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{pmatrix} \tag{2}$$

where Y are the output coefficients, X is a block of the input image and COS is the cosine matrix used to compute the DCT coefficients. The entries of COS are $a = \frac{1}{2}$, $b = \sqrt{\frac{1}{2}} \times cos\left(\frac{\pi}{8}\right)$ and $c = \sqrt{\frac{1}{2}} \times cos\left(\frac{3\pi}{8}\right)$. In the same way, the IDCT can be obtained by:

$$X = COS^T \times Y \times COS \tag{3}$$

Here we would like to underline that the matrix multiplications form presented in (2) and in (3) requires 128 multiplications and 96 additions.

2.2 Existing implementations

Reducing the computational complexity of DCT/IDCT transforms is considered by researchers and industrials as an attractive thematic.[5] The optimization of DCT/IDCT has focused generally on reducing the number of required arithmetic operators and especially the number of multipliers. Indeed, the multipliers are the most power and area consuming circuits. Moreover, in digital electronic design, multipliers are caracterized by a higher latency and are considered as the bottleneck for achieving the real-time requirements. Then, it has been demonstrated that the theoretical lower limit of 8-point DCT algorithm is 11 multiplications. In literature, many fast DCT algorithms are reported and all of them use the symmetry of the cosine function to reduce the number of multipliers. In[6] a summary of these algorithms is presented. In Table 1, we have listed the number of multipliers and adder involved in different DCT algorithms.

As mentionned in Table 1, the number of required arithmetic operators stills high. Therefore, many multiplierless DCT algorithms have been introduced for efficient implementation of constant multiplications. All those methods can be classified as : the Distributed Arithmetic (DA)-based design[13] and[14] , the New Distributed

Table 1. Complexity of different DCT algorithms

Reference	Chen[7]	Lee[8]	Vitterli[9]	Suehiro[10]	Hou[11]	Loeffler[12]
Multipliers	16	12	12	12	12	11
Adders	26	29	29	29	29	29

Table 2. Multiplierless desing for 1-D DCT calculation

Method	DA[13]	DA[14]	NEDA[15]	CSD[16]	CSD[18]
Adders	136	144	85	123	72

Arithmetic (NEDA)-based design[15] and CSD-based design.[16] We have proposed in[17] and[18] a CSD-based design to eliminate multipliers and to use a reduced number of adders. The hardware ressources occupation of these methods are listed in table 2. It is found that our design uses fewer adders than the other. The direct realization of DA-based DCT design requires 308 adders. Optimizations presented in[15] reduce the number of adders to 85. Regarding the CSD-based design,[16] for 8-bit constant width, we found that design of[16] consumes 123 adders (67 inta-strucutral adders + 56 inter-sutructural adders) while the proposed design involves the DCT with 72 adders.

As mentionned before, for image and video processing, the DCT is used in 2D form. One efficient way for the 2D-DCT calculation is by row/column decomposition. For a block size of 4×4 pixels, the decomposition consist in running the DCT 4 times on lines and then running the output of the first DCT 4 times on columns. Consequently, the use of row/column strategy requires an additional transpose memory to save the 1D-DCT outputs. Moreover, the absolute latency which is measured by evaluating the number of used clock cycles to obtain the output coefficients is relatively high. Indeed, for an input block of 4×4 pixels, the obsolute latency is equal to 15 clock cycles by using Loeffler or Chen algorithms.

2.3 Low-complexity direct realization of DCT

To eleminate the use of transpose memory and to reduce the absolute latency, Hallapuro et al. have introduced in[19] a low complexe direct 2D-DCT design based on the direct realization. Indeed, by means of matrix manipulations, equation (2) can be rewritten by:

$$Y = Y_{std} \otimes E \tag{4}$$

where $\otimes$ denotes element by element multiplication and Y_{std} is expressed by:

$$Y_{std} = C \times X \times C^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & d & -d & -1 \\ 1 & -1 & -1 & 1 \\ d & -1 & 1 & -d \end{pmatrix} \times \begin{pmatrix} X(1,1) & X(1,2) & X(1,3) & X(1,4) \\ X(2,1) & X(2,2) & X(2,3) & X(2,4) \\ X(3,1) & X(3,2) & X(3,3) & X(3,4) \\ X(4,1) & X(4,2) & X(4,3) & X(4,4) \end{pmatrix} \times \begin{pmatrix} 1 & 1 & 1 & d \\ 1 & d & -1 & -1 \\ 1 & -d & -1 & 1 \\ 1 & -1 & 1 & -d \end{pmatrix} \tag{5}$$

Matrix E can be used as scale factor and can be combined with the quantization matrix at the encoder or with the dequantization table at the decoder. Note that $d = c/b$ and E is expressed by:

$$E = \begin{pmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{pmatrix} \tag{6}$$

Compared to the more traditional formula of DCT presented in (2), Y_{std} calculaed with (5) has many trivial operations like the multiplications by ± 1. Hence, the number of multiplications used in (5) is equal to 16 which is less than 128 multiplications required by (2). Moreover, Y_{std} can be calculated without transpose memory. Indeed, many symetries can be obtained with matrix C in (5) which facilitate the representation of the Y_{std} by a signal flow graph based butterflies. Another advantage of the reresentation given in (5) consists in the possibility of proposing a generalized DCT for multistandard. Indeed, matrix C of (5) can be used in MPEG or in HEVC standards. This matrix is composed of ± 1 and $d = \sqrt{2} - 1 = 0.4142$ entries. Coefficient d can be sustituted by $1/2$. In terms of hardware complexity, the multiplication by $1/2$ can be implemented by means of shifter.

Then, in order to avoid truncation erros, authors of[19] proposed to scale C matrix by 2. The forward transform becomes:

$$Y_{std}^{int} = C \times X \times C^T = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} \times \begin{pmatrix} X(1,1) & X(1,2) & X(1,3) & X(1,4) \\ X(2,1) & X(2,2) & X(2,3) & X(2,4) \\ X(3,1) & X(3,2) & X(3,3) & X(3,4) \\ X(4,1) & X(4,2) & X(4,3) & X(4,4) \end{pmatrix} \times \begin{pmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{pmatrix} \quad (7)$$

Under these conditions, matrix E is expressed by:

$$E^{int} = \begin{pmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{pmatrix} \quad (8)$$

3. EXTENTION AND FUNCTIONAL VALIDATION

3.1 Extention

DCT presented in[19] is defined for block size of 4×4 pixels. In order to support all video coding standards in a single plateforms, it becomes necessay to develop a generalized DCT architecture. Equation (7) is devoted to compute DCT for H264 video coding standard as well as JPEG XR standard. However, for HEVC and JPEG standards, the pixel block to be transformed has a size of 8×8 *. Under these conditions, matrix COS presented (2) is updated according to (9):

$$COS = \begin{pmatrix} d & d & d & d & d & d & d & d \\ a & c & e & g & -g & -e & -c & -a \\ b & f & -f & -b & -b & -f & f & b \\ c & -g & -a & -e & e & a & g & -c \\ d & -d & -d & d & d & -d & -d & d \\ e & -a & g & c & -c & -g & a & -e \\ f & -b & b & -f & -f & b & -b & f \\ g & -e & c & -a & a & -c & e & -g \end{pmatrix} \quad (9)$$

where $a = cos(\pi/16)$, $b = cos(2\pi/16)$, $c = cos(3\pi/16)$, $d = cos(4\pi/16)$, $e = cos(5\pi/16)$, $f = cos(6\pi/16)$ and $g = cos(7\pi/16)$.
As in (5), we calculate matrix C in order to compute Y for block size of 8×8.

$$Y = B \times C \times X \times C^T \times B^T = Y_{std} \otimes E \quad (10)$$

where $Y_{std} = C \times X \times C^T$ and matrices B, C and E are defined by:

$$B = \begin{pmatrix} d & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & b & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & d & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & e & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & f & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g \end{pmatrix} \quad (11)$$

*For HEVC, the pixel block size varies from 2×2 to 32×32

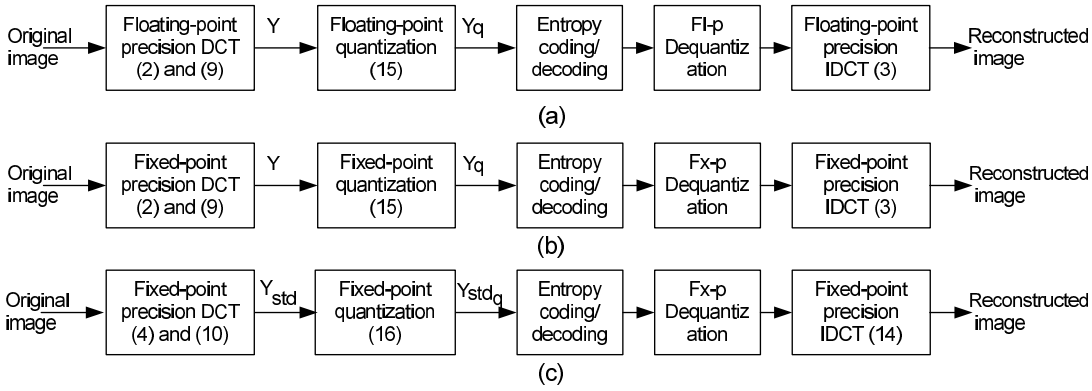

Figure 1. Proposed compression schemes

$$C = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & c/a & e/a & g/a & -g/a & -e/a & -c/a & -1 \\ 1 & f/b & -f/b & -1 & -1 & -f/b & f/b & 1 \\ 1 & -g/c & -a/c & -e/c & e/c & a/c & g/c & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -a/e & g/e & c/e & -c/e & -g/e & a/e & -1 \\ 1 & -b/f & b/f & -1 & -1 & b/f & -b/f & 1 \\ 1 & -e/g & c/g & -a/g & a/g & -c/g & e/g & -1 \end{pmatrix} \tag{12}$$

$$E = \begin{pmatrix} d^2 & da & db & dc & d^2 & de & df & dg \\ ad & a^2 & ab & ac & ad & ae & af & ag \\ bd & ba & b^2 & bc & bd & be & bf & bg \\ cd & ca & cb & c^2 & cd & ce & cf & cg \\ d^2 & da & db & dc & d^2 & de & df & dg \\ ed & ea & eb & ec & ed & e^2 & ef & eg \\ fd & fa & fb & fc & fd & fe & f^2 & fg \\ gd & ga & gb & gc & gd & ge & gf & g^2 \end{pmatrix} \tag{13}$$

To obtain a low-complexity architecture, we calculate Y_{std} instead of Y. Note that the element by element multiplication with matrix E will be performed in the quantization side. Also, we would like to underline that $C \times C^T$ is proportional to the identity matrix; which means that the IDCT can be obtained by same equation as in (3):

$$X = C^T \times Y_{std} \times C \tag{14}$$

3.2 Functional validation

In this section we analyse the effects of the proposed matrix rewriting in the quality of reconstructed images. A simplified block diagrams of the proposed compression schemes are presented in Figure 1.

Indeed, Figure 1.(a) shows the data flow graph (DFG) with 64-bits floating-point precision. Hence, the last is considered as the theoretical DFG. In the other hand, Figures 1.(b) and 1.(c) show DFGs with Fixed-point precision respectively for theoretical matrix representation (equations (2) and (9)) and optimized matrix representations (equations (4) and (10)).

Accordingly, the 2D-DCT of 4×4 or 8×8 blocks of the image is performed to decorrelate each block of input pixels. The DCT coefficients are then quantized to represent them in a reduced range of values using a quantization matrix. Finally, the quantized components are scanned in a zigzag order, and the encoder employs run-length encoding (RLE) and Huffman coding for entropy coding. Remember that the quantized

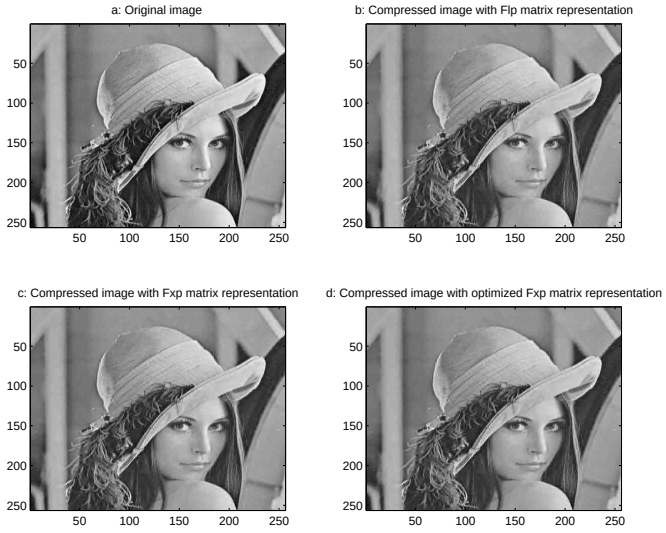

Figure 2. Reconstructed image with block size of 4×4

DCT coefficients in JPEG, MPEG-4 and H263 are defined in[1, 20]

$$Y_q(u,v) = \left\lfloor \frac{(Y(u,v) \times 16)}{Q^s(u,v) \times Q_p} \right\rfloor \tag{15}$$

where $\lfloor x \rfloor$ denotes the nearest integer less than or equal to x, $Q^s(u,v)$ are the elements of the quantization matrix given by the standard, $Q_p \in [1:31]$ is the quantization parameter and $Q(u,v) = Q^s(u,v) \times Q_p$ is the used quantization matrix for a set of Q_p parameter. Note that quantizer scaling does not affect the quantization of the DC coefficient.

For our case, when we use Y_{std} to compute DCT coefficients, (15) is updated to include E:

$$Y_{std_q}(u,v) = \left\lfloor \frac{Y_{std}(u,v) \times 16 \times E}{Q^s(u,v) \times Q_p} \right\rfloor \tag{16}$$

Simulation results are perfomed using Matlab tool. Figures 2 and 3 show the compressed Lena and peppers images respectively for block size of 4×4 and 8×8. With Figures 2.(b) and 3.(b) we used the theoretical compression scheme of Figure 1.(a). The obtained PSNRs are respectively equal to 28.65 dB and 24.84 dB. Similarly, in Figures 2.(c) and 3.(c) are obtained the reconstructed images with the compression scheme of Figure 1.(b). The obtained PSNRs are equal to 28.65 dB and 24.68 dB. The decrease in PSNR is due to the effect of truncation in the quantization matrix as well as COS matrices. Finaly, we obtain reconstructed images in Figures 2.(d) and 3.(d) according to the compression scheme of Figure 1.(c). The equivalent PSNR are equal to 27.02 dB and 26.46 dB. A more complete analysis of PSNR evaluation is given in Figure 4. Indeed, Figures 4 (a) and 4 (b) show PSNR evaluations with block sizes of 4×4 while Figures 4 (c) and 4 (d) show PSNR evaluations with block sizes of 8×8 respectively for Lena and peppers images. It is mentionned that the PSNR decreases when the Q_p parameter increases. This is in accordance with the quantization process. Moreover, it is mentionned that the deterioration in PSNR obtained with the architectural optimization is less thant 1 dB. In some cases, the PSNR given with the optimized DCT is higher than that calculated by DCT of Matlab. To sum up, we can confirm that the proposed DCT does not affect the image quality.

4. DCT DESIGN

The low complexity architecture of DCT can be easily presented by butterfly structure. As mentionned in Figure 5, the 4×4 2D-DCT is transpose-memory free and has a regular structure which fits well with FPGA

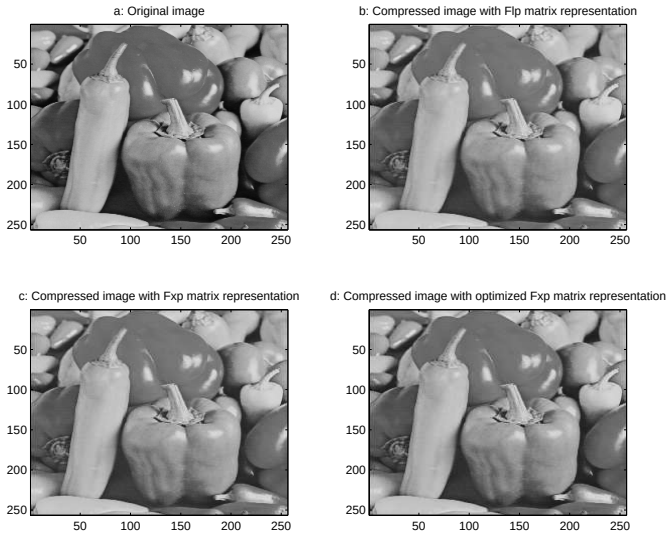

Figure 3. Reconstructed image with block size of 8×8

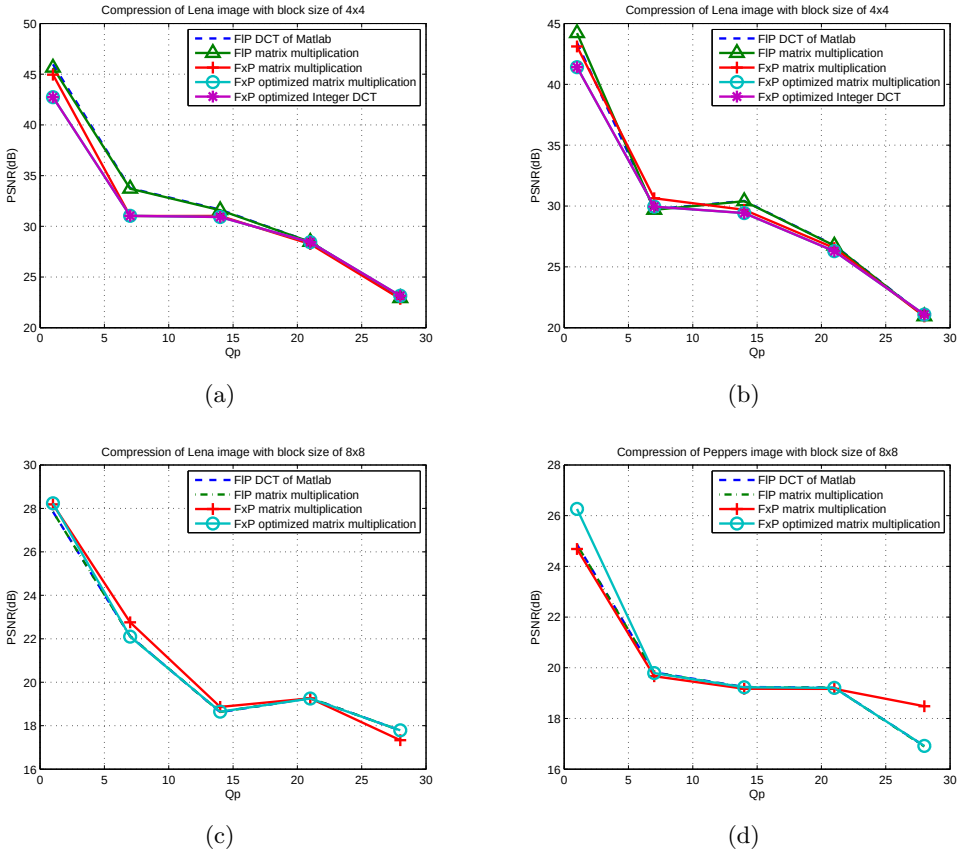

Figure 4. PSNR for proposed compression schemes

implementation. As we can see in Figure 5, the 2D-DCT is computed by means of two types of buterflies. The first one uses one addition and one subtraction while the second butterfly uses the same ressources along with

two constant multipliers.

Hence, when we use the matrix multiplication according to equation (2) we consume 128 multiplications and

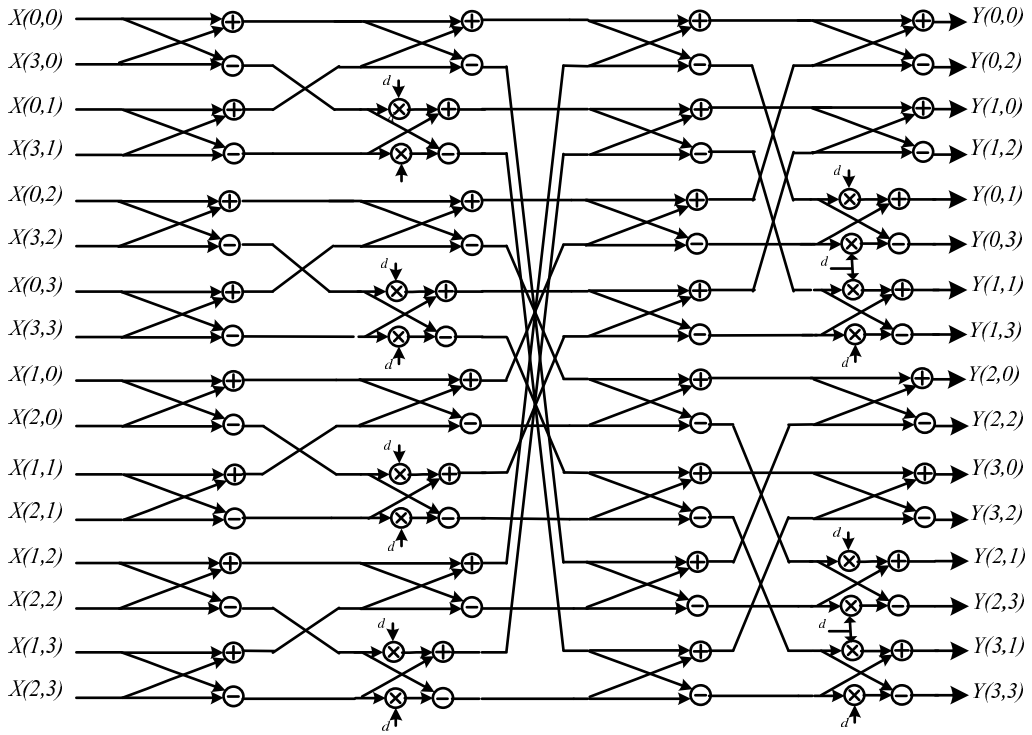

Figure 5. Low complexity 4×4 2D-DCT architecture

96 additions for block size of 4×4. When the block size is 8×8, the number of multiplications is about 1024 and the number of addition is about 896. In the other hand, the row-column decomposition of Leoffler gives 32 multiplications and 64 additions for 4×4 block size and 176 multiplications and 416 addition for 8×8 block size. Unfortunately, the use of this decomposition requires one transposition memory to save intermediate results which consumes hardware ressources. Moreover, with this configuration, the computation is recursive and the latency is increased. For all these reasons, the proposed design consists a good compromise between hardware ressources and latency. It consumes 64 additions and 16 multiplications for 4×4 block size and requires 256 multiplications and 416 additions for 4×4 block size without any memory transposition.

5. CONCLUSION

In this paper, we have presented a low-complexity DCT for image and video compression. We extended an existing work for a block size of 8×8 to be used for multistandad. We showed that the proposed DCT is hardware implementation friendly. Then, we demonstrated that the proposed design consumes less hardware ressources. Morevoer, we proved that the modification in the matrix representation doest not affect the image quality. Nevertheless, this work is our first tentative in the domain of multistandard encoders. In our future work, we aim to present an automatic methodologie to extend the matrix reformulation for a higher block sizes. Also we expect to present a generalized DCT design for many encoding standards. FPGA implementation results will be given in our future work.

REFERENCES

[1] Pereira, F. C. N. and Ebrahimi, T., [*The MPEG-4 book*], IMSE (2002).

[2] VCEG, I. M.-T., [*Joint Video Team (JVT)*], ISO/IEC (2003).
[3] Bross, B., Han, W.-J., Sullivan, G. J., Ohm, J.-R., and Wiegand, T., "High efficiency video coding HEVC text specification draft 8,"
[4] N. Ahmed, T. N. and Rao, K. R., "Discrete cosine transform," *IEEE Trans. Compt.* **C-32**, 90–94 (Jan. 1974).
[5] Jridi, M. and Alfalou, A., "A low-power, high-speed DCT architecture for image compression: Principle and implementation," in [*Proc. IEEE/IFIP VLSI Syst. on Chip Conf. (VLSI-SoC)*], 304–309 (Sept. 2010).
[6] Pai, C. Y., Lynch, W. E., and Al-Khalili, A. J., "Low-power data-dependant 8x8 DCT/IDCT for video compression," *Proc. IEE Vis. Image Signal Process.* **150**, 245–254 (Aug. 2003).
[7] Chen, W. A., Harrison, C., and Fralick, S. C., "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.* **25**, 1004–1011 (1977).
[8] Lee, B., "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing* **32**, 1243–1245 (1984).
[9] Vitterli, M. and Nussbaumer, H., "Simple FFT and DCT algorithms with reduced number of operation," *Signal Processing* **6**, 264–278 (1984).
[10] Suehiro, N. and Hatori, M., "Simple FFT and DCT algorithms with reduced number of operation," *IEEE Trans. Acoust., Speech, Signal Processing* **34**, 642–664 (1986).
[11] Hou, H., "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing* **35**, 1455–1461 (1987).
[12] Loeffler, C., Lightenberg, A., and Moschytz, G. S., "Practical fast 1-D DCT algorithm with 11 multiplications," in [*Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing*], 988–991 (May 1989).
[13] Madisetti, A. and Willson, A. N., "A 100 MHz 2-D 88 DCT/IDCT processor for HDTV applications," *IEEE Trans. Acoust., Speech, Signal Processing* **5**, 158–165 (Apr. 1995).
[14] Kim, D. W., Kwon, T. W., Seo, J. M., Yu, J. K., , Lee, S. K., Suk, J. H., and Choi, J. R., "A compatible DCT/IDCT architecture using hardwired distributed arithmetic," in [*Proc. IEEE Symp. on Circuits and Syst*], 457–460 (May 2001).
[15] Shams, A., Pan, W., Chidanandan, A., and Bayoumi, M., "A low power high performance distributed DCT architecture,"
[16] Vinod, A. P. and Lai, E. M.-K., "Hardware efficient DCT implementation for portable multimedia terminals using subexpression sharing,"
[17] Jridi, M. and AlFalou, A., [*VLSI-SoC: Forward-Looking Trends in IC and System Design*], VLSI-SoC 2010 IFIP WG 10.5, Springer Verlag (2012).
[18] Jridi, M., AlFalou, A., and Meher, P. K., "Optimized architecture using a novel subexpression elimination on loeffler algorithm for DCT-based image compression," *VLSI Design* **2012** (2012).
[19] A. Hallapuro, M. K. H. M., "Low complexity transform and quantization - part i: Basic implementation,"
[20] ITU-H263, "Coding of moving video: video coding for low bit rate communication," *ITU-T H.263-ISO/IEC 10918-3* (2005).