Efficient hardware architecture for direct 2D DCT computation and its FPGA Implementation

Conference Paper · December 2013

DOI: 10.1109/ICM.2013.6734995

4 authors, including:

Hatim Anas
Ecole Nationale des Sciences Appliquées Agadir

Sadiki Tayeb
Université Internationale de Rabat

M. M. Hassani
Cadi Ayyad University

Efficient hardware architecture for direct 2D DCT computation and its FPGA Implementation

Anas Hatim, Said Belkouch
Embedded Systems and Digital Control Laboratory,Ecole Nationale des Sciences Appliquées, University of Cadi Ayyad, PB. 575, Av. Abdelkarim Khattabi, Guéliz Marrakech,
Morocco
Hatim.anas@gmail.com,s.belkouch@uca.ma

Tayeb Sadiki,
Electronic, Logistic,informatics,and Telecom Laboratory
Technopolis Rabat-Shore, Int. Univ. of Rabat, Sala el Jadida,
Morocco
tayeb.sadiki@uir.ma

Moha M'Rabet Hassani
Technology, Electronics and Instrumentation Lab
Faculty of Sciences Semlalia, Cadi Ayyad University, Marrakech,
Morocco
hassani@ucam.ac.ma

Abstract—**In this paper, we propose a low complexity architecture for direct 2D-DCT computation. The architecture will transform the pixels from spatial to spectral domain with the required quality constraints of the compression standards. In our previous works we introduced a new fast 2D_DCT with low computations: only 40 additions are used and no multiplications are needed. Based on that algorithm we developed in this work a new architecture to achieve the computations of the 2D DCT directly without using any transposition memory. We defined S_k functions blocks to build the 2D DCT architecture. The S_k block perform 8 function depending on the control signals of the system. The number of additions/subtractions used is 63, but no multiplication or memory transposition is needed. The architecture is suitable for usage with statistical rules to predict the zero quantized coefficients, which can considerably reduce the number of computation. We implemented the design using an FPGA Cyclone 3. The design can reach up to 244 MHz and uses 1188 logic elements, and it respect the real time video requirements.**

Key words: DCT, FPGA, Image Compression, Video processing.

I. INTRODUCTION

The multimedia and data processing specially image and video are used increasingly in the word. Many applications integrate embedded devices with special circuits for video and image processing. The compression standards are the main blocks used in every video and image application. As a result, multiple compression schemes have been introduced and oriented to several applications such as: MPEG4, H2642[1,2] for video compression, and JPEG and JPEG 2000 for image compression. The main block present in the compression scheme is the transformation domain block. The Discrete Cosine Transform (DCT) is used for JPEG and MPEG in order to reduce the spatial redundancies by transforming the spatial domain in the spectral domain. The principal behind the transformation with DCT algorithm is the removal of redundancy between neighboring pixels [3]. Efficiency of the coder can be evaluated by its ability to compress data into as few coefficients as possible. This allows the quantizer to eliminate the coefficients with small amplitudes without introducing visual degradation [4,5]. DCT achieves important energy reduction for highly correlated images. The direct computation of the 2D-DCT requires $2N^3$ multiplications and $2N^2\times(N-1)$ additions, where N×N is the size of a block of pixels. For an 8×8 transform the number of multiplications is 1024 and additions is 896.In order to reduce the complexity of the algorithm many works has introduced modified algorithms and scaled DCT [6,7,8]. The hardware complexity of DCT architecture can still be greatly reduced by taking advantage of the fact that a scaled version of the DCT is enough in most current DCT applications [9]. In this paper, we present hardware architecture for implementing a new proposed scaled DCT [10]. The algorithm is multiplier-less, all the multiplications are introduced into the quantization block. Indeed, the matrix multiplication used to compute the 2D-DCT coefficients is composed of three matrixes building three stages architecture. The Row column separability uses a transposition memory which is the main disadvantage of this method. In this work we present architecture for direct computation of the 2D DCT coefficients. The paper is organized as follows: Section 2 presents the development of the direct 2D DCT computations, section 3 present the hardware architecture proposed, section 4 present the implementation results of the proposed architecture and the section 5 a conclusion is given.

II. DIRECT 2D DCT COMPUTATION

The DCT algorithm in [10] has a low complexity because all the multiplications were gathered at the end of the transform and merged on the quantization stage.

Let $X = \left(x_{(i,j)}\right)_{i,j\in\{0,..,7\}}$ an 8×8 matrix the input pixels of the transform. The transform introduced in [10] is:

$$\mathrm{Y} = (\mathrm{T}\times\mathrm{X}')'\times\mathrm{T}' \tag{1}$$

Where Y is the output of the system.

The matrix T is computed as follows:

$$T = T_3 \times T_2 \times T_1 \tag{2}$$

Where:

$$\mathrm{T1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{3}$$

$$\mathrm{T2} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

$$\mathrm{T3} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

To compute the 2D_DCT using the row column separability, we must fed the first 1D_DCT inputs with vectors of 8 pixels during 8 cycles and store the results on a transpose buffer , after that we will be able to process the 1D_DCT again on the rows of the resulting matrix . In this paper we will compute directly the 2D_DCT outputs without transposition memory. The mathematical development is as follow:

Using eq (2), the matrix T is calculated and it's equal to

$$T = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & -1 & -1 & -1 & 0 \\ 1 & 0 & -1 & -1 & -1 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & -1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 1 \\ 1 & 0 & -1 & 1 & -1 & 0 & 1 & 0 \\ 1 & -1 & 1 & 0 & -1 & 1 & -1 & 0 \\ 1 & -1 & 1 & 0 & 1 & -1 & 1 & 0 \end{bmatrix} \quad (6)$$

The eq (1) can be written as follows:

$$Y = (A)' \times T' \quad (7) \quad \text{where} \quad A = X' \times T' \quad (8)$$

The results of the multiplication of T' with an 8 pixels vector $I = \{I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7\}$ is an 8 pixel vector S = I × T'.

The elements of the vector S are expressed in table 1.

TABLE 1 THE COMPOSITION OF THE ELEMENTS OF THE VECTOR S

Pixel order	Combination
S_0	$I_0 + I_1 + I_2 + I_3 + I_4 + I_5 + I_6 + I_7$
S_1	$I_0 + I_1 - I_6 - I_7$
S_2	$I_0 + I_1 - I_2 - I_3 - I_4 - I_5 + I_6 + I_7$
S_3	$-I_2 + I_5$
S_4	$I_0 - I_1 - I_2 + I_3 + I_4 - I_5 - I_6 + I_7$
S_5	$I_0 - I_1 + I_6 - I_7$
S_6	$I_0 - I_1 + I_2 - I_3 - I_4 + I_5 - I_6 + I_7$
S_7	$-I_3 + I_4$

In our system the input will be an 8×8 matrix instead of an 8 pixels vector in the row column design. The 2D_DCT coefficients will be computed directly using the table1.

Let's call $S_k(V)_{i\in\{0,..,7\}}$ 8 functions with 8 pixel vector as input. The definitions of the S_k function are exactly like in table 1, for example:

$$S_0(v) = v_0 + v_1 + v_2 + v_3 + v_4 + v_5 + v_6 + v_7$$

Using Eq(9) and the S_k functions A will expressed as follows:

$$A = \begin{bmatrix} S_0(X(0,i)) & S_1(X(0,i)) & \dots\dots\dots & S_7(X(0,i)) \\ S_0(X(1,i)) & \dots\dots\dots & \dots\dots\dots & \dots\dots\dots \\ \dots\dots\dots & \dots\dots\dots & \dots\dots\dots & S_7(X(6,i)) \\ S_0(X(7,i)) & \dots\dots\dots & S_6(X(7,i)) & S_7(X(7,i)) \end{bmatrix}_{i\in\{0,..,7\}} \quad (9)$$

we will express the 2D_DCT outputs Y using eq (8) as follows:

$$Y = \begin{bmatrix} S_0(A'(0,i)) & S_1(A'(0,i)) & \dots\dots\dots & S_7(A'(0,i)) \\ S_0(A'(1,i)) & \dots\dots\dots & \dots\dots\dots & \dots\dots\dots \\ \dots\dots\dots & \dots\dots\dots & \dots\dots\dots & S_7(A'(6,i)) \\ S_0(A'(7,i)) & \dots\dots\dots & S_6(A'(1,i)) & S_7(A'(7,i)) \end{bmatrix}_{i\in\{0,..,7\}} \quad (10)$$

Let's detail the elements of Y.

$Y(0,0) = S_0(A'(0,i))_{i\in\{0,..,7\}}$

$= S_0(S_0(X(0,i)), S_0(X(1,i)), S_0(X(2,i)), S_0(X(3,i)), S_0(X(4,i)), S_0(X(5,i)), S_0(X(6,i)), S_0(X(7,i)))$

Y (0, 0) is computed using the S_0 function twice, the first time on all the rows and the second time on the resulting vector.

$Y(1,0) = S_0(A'(0,i))_{i\in\{0,..,7\}}$

$= S_0(S_1(X(0,i)), S_1(X(1,i)), S_1(X(2,i)), S_1(X(3,i)), S_1(X(4,i)), S_1(X(5,i)), S_1(X(6,i)), S_1(X(7,i)))$

Y (1, 0) is computed using first the S_1 function on all the rows and after the S_0 function on the resulting vector. With the same manner we can compute every output of the 2D_DCT directly using the S_k function two times.

III. HARDWARE ARCHITECTURE

From the equations cited above we built a two stage architecture based on the S_k functions to compute every 2D DCT coefficient. The first stage is composed of 8 S_k blocks, the second with one S_k block the architecture is illustrated in Figure 1.

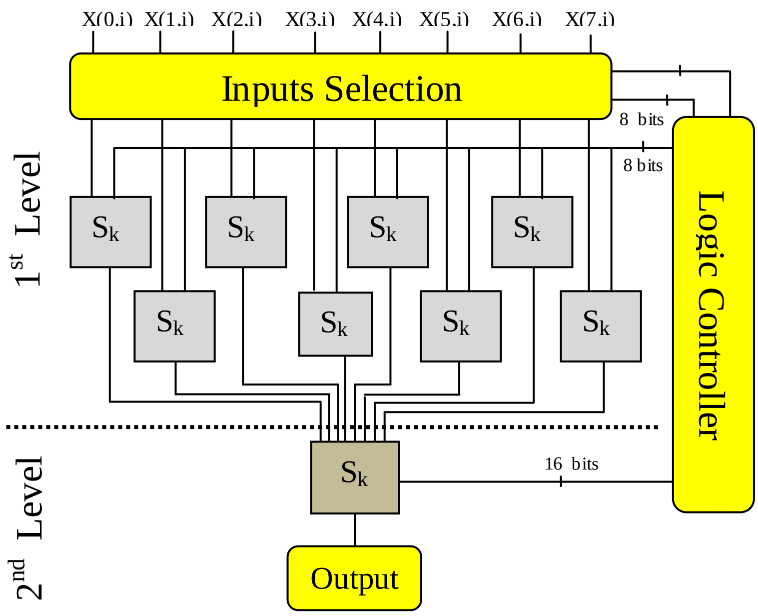

Fig.1 Architecture of the Direct 2D DCT

Every S_k block can compute the 8 S_k functions with the same architecture. Based on the logic controller signals the S_k block choose between add/ sub arithmetic operations to perform the desired S_k function. The S_k block is 3 pipelined stages. Figure 2 shows the design of an S_k block. The logic controller also generates signals used to choose if the inputs must be unchanged or zero if necessary like in the function S_5. The logic controller is composed of a counter and 2 memories. The memories have eight 15 bit words, every word is divided into two parts. The first 7 bits are responsible of building the desired S_k functions of the two levels. Every bit of the first part is used to control an ADD/SUB component, zero is used to choose an addition and one is used to choose substraction. The memory words values are found as follows, for example:

$$S_2 = I_0 + I_1 - I_2 - I_3 - I_4 - I_5 + I_6 + I_7$$
$$= (I_0 + I_1) - (I_2 + I_3) - (I_4 + I_5) + (I_6 + I_7)]$$
$$= [(I_0 + I_1) - (I_2 + I_3)] - [(I_4 + I_5) - (I_6 + I_7)]$$

So the part of the memory word used to build the S2 function is "1110000" where the bit0 is used to configure the arithmetic operator Unit0, the bit 1 is used to configure the arithmetic operator Unit1 and so one. With the same manner the 8 memory word are expressed and are presented in table 2.

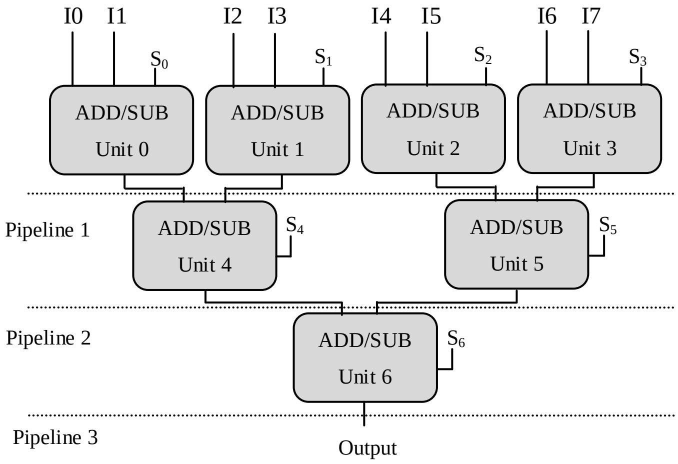

Fig. 2 The Architecture of an Sk block

The second part of the memory word is composed of 8 bits .the second pat is used to select the inputs of the S_k blocks depending on which function will be used. The input may be zero or not changed. A multiplexer is used to choose between the two inputs using the selection memory second part word as selection bits. If the selection bit is "0" then the MUX output is the input, if the selection bit is "1" then the MUX output is zero. The logic controller must generate two 3 bits signals used as address of the memory of each level. The selection part of the memory words are constructed as follows, for example:

$S_7 = -I_3 + I_4$

$S_7 = 0 \times I_0 + 0 \times I_1 + 0 \times I_2 - I_3 + I_4 - 0 \times I_5 + 0 \times I_6 + 0 \times I_7$

Taking in consideration the arithmetic operation used, the selection word is "11100111". Where the first bit is used to choose the first input and respectively. The input selection words are shown in table 2.

TABLE 2 THE LOGIC CONTROLLER'S MEMORY WORDS

Function	Selection words	
	Arithmetic selection Part	Input selection Part
S_0	"1111111"	"00000000"
S_1	"1101111"	"00111100"
S_2	"0001111"	"00000000"
S_3	"1101111"	"11011011"
S_4	"1000000"	"00000000"
S_5	"1110110"	"00111100"
S_6	"0110000"	"00000000"
S_7	"1111101"	"11100111"

The outputs of the 2D_DCT are respectively Y(0,0), Y(1,0), Y(2,0) …….Y(6,7), Y(7,7). According to that succession the combination of the S_k functions in the first stage and in the Sk function of the second stage are shown in Table 3. A 64 modulo counter is designed to increment the address of the memory selection. Since the S_k block of the second level shifts 8 times fast than the S_k block in the first stage, the least 3 significant bits of the counter are used to control the second level and the most 3 significant bits are used to control the first level .

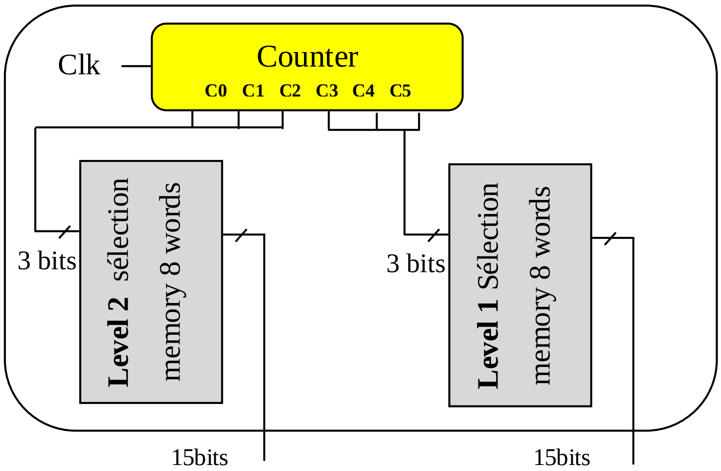

Fig.4 The logic controller architecture

The number of the Add/sub modules used to build an Sk block is 7, and the number of the Sk blocks needed to compute a 2D DCT coefficient is 9. The total number of Add/Sub modules is 63, the total multipliers is zero. Since we use a pipelined architecture and we compute directly the outputs of the 2D DCT, the architecture has a throughput of one pixel per clock. The total memory used is 2 eight 15 bits words memories to build the logic controller. We notice that any transpose buffers were used. The work in [11] presents a direct 2D DCT computation. The work is an extension of the work presented in [12] which present a scaled 4×4 2D_DCT. The final 8×8 algorithm presented has a complexity of 128 Addition/subtraction and 112 multiplication. No implementation has been done in the presented work. The method in [13] presents a cost effective method without transposition memory. Instead of SRAM memory, to store the cosine coefficients, the architecture uses a state machine to generate them. The architecture accumulates intermediate results to produce a processed coefficient. It uses 36 additions and any multiplication but uses complex state machine to build the final results. It achieves the 2D DCT with a low throughput: it takes 8 cycles to compute one coefficient. In literature many works introduced the polynomial transforms to eliminate the transposition buffers and compute directly the 2D DCT. It consists on mapping the multidimensional transforms on one dimensional one with a lower dimension. Table 4 shows a comparison with other techniques.

TABLE 4 COMPLEXITY COMPARISON WITH OTHER TECHNIQUES

Design	Our work	M.Jridi et al.'s [11]	Tomeo et al.'s [4]	Kusuma, E.D et al.'s[5]
Add/Sub	63	128	36	58
Mult	0	112	22	16

The statistical and heuristic rules [14] are introduced in the literature. The main purpose of these rules is to avoid the computation of a predicted zero quantized coefficient. A preprocessor is added before the DCT block to take the decision of computing or not a 2D DCT coefficient.

TABLE 3 THE EVOLUTION OF SK BLOCKS OF THE TWO LEVELS

Level2	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_0	….	….	…..	S_5	S_6	S_7
Level1	S_0								….		….	S_7			
Count	0	1	2	3	4	5	6	7	8	….	….	…..	61	62	63

These methods are used with DCT architectures producing one pixel per clock cycle. The efficiency of these methods is between 25% to 50% savings on the number of the computations. Our architecture is very suitable to be combined with these methods, it will reduce considerably the number of the computation of an 8×8 block. Implementation results

A. MATLAB Modeling

We proved the efficiency of our method using MATLAB .We calculated the PSNR to evaluate the quality of the reconstructed image, with various parameters. Figure 5 shows the PSNR in function of the quantization step for different number of bits precision after the quantization stage. The results show that our method respects the standards requirements. When increasing the number of bits after the quantization step the PSNR increase which is normal since the precision of the data transmitted will be increased. We choose 2 bits after the quantization stage, which allows us to have a maximum PSNR around 42 db.

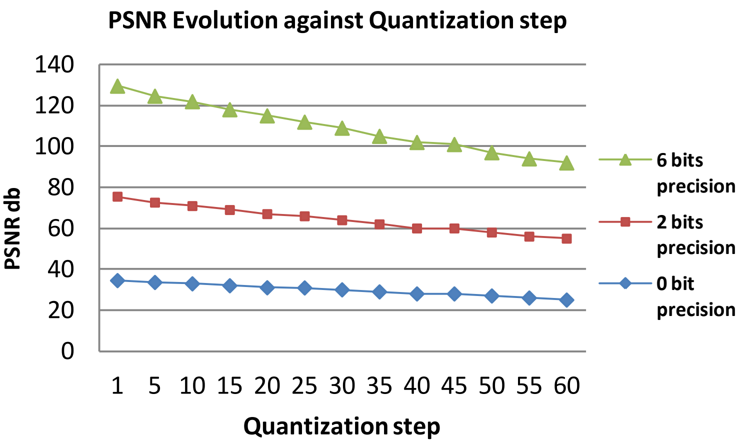

Fig.5 The PSNR against the quantization step with different precision values

B. FPGA Implementation

The architecture of the direct 2D DCT has been implemented on FPGA cyclone 3, using Quartus 2 development Tools.

The maximum frequency of the architecture is 244 Mhz, it consumes 1188 logic elements and 1452 sequential elements. In comparison with other state of the art techniques we gained up to two times in frequency and reduced by 22% average the consumed logical elements. Table 5 shows a comparison with other techniques.

IV. CONCLUSION

In this paper, we proposed a new direct 2D DCT architecture for a fast efficient scaled transform. The architecture computes the 2D_DCT coefficient without transposition memory, which is the drawback of row column separability technique. The algorithm is multiplier-less and uses defined Sk blocks to perform functions applied on the lines and resulting columns. The architecture consists of two stages, and has a throughput of one pixel per clock cycle. The architecture can be used with preprocessors based on statistical and heuristic rules to reduce the computation of the 2D DCT. The preprocessors predict the zero quantized coefficients and can reduce up to 50% of the total operations. The total add/Sub used is 63, we notice that no multiplication is needed and any transposition memory is used. The design is regular and can be used as codec. Our FPGA implementation of the architecture works with 244 MHz and uses 1188 logical elements. We reduced the used resources by 30% in comparison with some techniques in the state of the art and increase considerably the max frequency. For future work we will improve the architecture by reducing its complexity.

TABLE 5 COMPARISON BETWEEN DIFFERENT ARCHITECTURE

Design	Our design	Tomeo et al.'s [4]	Kusuma, E.D et al.'s [5]
Max frequency	244	107	84
Latency	6	24	22
Throughput pixel/s	1	8	8
Logic element	1188	2618	1750

V. 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] Rafael C. Gonzalez, Richard E. Woods "Digital Image Processing". ISBN-13: 978-0131687288

[4] Tumeo A, Monchiero M, Palermo G, Ferrandi F, Sciuto D (2007) A Pipelined Fast 2D-DCT Accelerator

[5] Kusuma ED, Widodo TS "FPGA implementation of pipelined 2D-DCT and quantization architecture for JPEG image compression". Intern Symp Inform Tech (ITSim) pp 1–6,2010.

[6] Jridi, M., AlFalou, A., and Meher, P. K., "Optimized architecture using a novel sub expression elimination on loeffler algorithm for DCT-based image compression," VLSI Design 2012.

[7] Wu Z, Sha J, Wang Z, Li L, Gao M "An improved scaled DCT architecture". IEEE Trans Consum Electron 55–2:685–689,2009

[8] Xiuhua J, Caiming Z, Xuefen Z "An efficient joint implementation of three stages for fast computation of color space conversation in image coding/decoding". Multimed Tool Appl pp 1–15, 2011.

[9] S.Bougzeel and M.Omair Ahmed and M.NS.Swamy "A fast 8×8 for image compressing" *Proc. of the International Conference on Microelectronics*, pp. 74-77, Dec 2009.

[10] A. Hatim , S. Belkouch et al "Design optimization of the quantization and a pipelined 2D-DCT for real-time applications". Multimedia Tool Appl 2012

[11] M.Jridi ,Y.Ouerhani and A. Alfalou "Low complexity DCT engine for image and video compression" Proc. SPIE 8656, Real-Time Image and Video Processing , 86560F (February 19, 2013); doi:10.1117/12.2006174.

[12] A. Hallapuro, M. K. H. M., "Low complexity transform and quantization - part i: Basic implementation" Circuits and Systems for Video Technology, IEEE Transactions on (Volume: 13, Issue: 7),2003.

[13] S.C.Hsia, C.F.Tsai, S.H. Wang ,K.C.Hung "Transposed-Memory Free Implementation for Cost-Effective 2D-DCT Processor" J Sign Process Syst (2010) 58:161–172, DOI 10.1007/s11265-009-0344-5

[14] Saponara S, Fanucci L, Terreni P "Low- Power VLSI architectures for 3D Discrete Cosine Transform" IEEE 48^{th} Midwest Symposium on Circuits and Systems, pp 1567-1570.vol 3 Dec 2013.