Fast 8×8×8 RCF 3D_DCT/IDCT transform for real time video compression and its FPGA Implementation

Anas Hatim[1] , Said Belkouch[1] ,Moha M'rabet Hassani[2]
[1]National School of Applied Sciences, University of Cadi Ayyad,
PB. 575, Av Abdelkarim Khattabi, Guéliz, Marrakech, Morocco
[2]Technology, Electronics and Instrumentation Lab,
Faculty Of Sciences and Techniques, University of Cadi Ayyad, Marrakech, Morocco

Abstract
Video compression takes place increasingly in many applications which are constantly involving. It becomes more demanding in terms of performance at the expense of more power consumption. Discrete Cosine transform is the most common technique used in the compression field. In this paper we present an efficient fast Row Column Frame (RCF) 3D_DCT algorithm recently introduced in video compression. The optimization consists on the elimination of all the multiplications needed to compute the coefficients of the 3D DCT. The multiplications are gathered at the end of the 3D DCT and merged in the quantization cube. The mathematical demonstration and complexity comparisons with other techniques are presented, showing that new (RCF) 3D _DCT introduced makes important savings on arithmetic operations, even if there is a little decrease on the quality of the videos but it respects the video standards. We made 59% savings on the number of additions and eliminate totally (100%) the multiplications in reference to the standard RCF technique. The resulting transform is regular and symmetric. We present hardware architectures to implement the 3D DCT and detailed their properties. We proposed ping pong buffers as transpose memory to increase the performance of the architectures. An implementation on a 180 nm FPGA chip has been done and discussed at the end. We can process high resolution video standards like HDTV using our architectures with both reduced power and area.

Keywords *3D DCT, video and image compression, real time processing, FPGA.*

I. INTRODUCTION

The cosine transform is a technique which directly eliminates duplication of data using the frequency domain. 2D DCT removes spatial redundancies in the frames of a video by eliminating inter correlations between each pixel. The 3D DCT goes beyond the spatial correlation and eliminates the temporal redundancies in the spatiotemporal N×N×N cubes. The 3D DCT is now widely used for 2D and 3D video compression standards. The 2D video compression standard including H264/AVC uses the prediction and motion compensation [1, 2] which are recently replaced by 3D DCT algorithm. The inter-frame prediction and motion compensation are heavy operations demanding blocks because of their complex mathematical manipulation. As consequence both it degrades the performance of the coder and it is very demanding in term of needed resources and power consumption.
3D DCT is used in 3D video compression to eliminate interview redundancies in the MVC pattern (multi-view coding) [3,4]. Recent works has have proven that the use of the 3D DCT is more efficient than MPEG-2 but can't outperform the MPEG-4 in very high data rates [5,6]. Performances of 3D DCT in terms of compression, video quality, and the statistical distribution coefficients were analyzed in several studies [7-14]. Even if the 3D DCT is less complicated than the coders cited above, the computation of its coefficients requires a lot of mathematical and arithmetic manipulations including multiplication [15]. Several studies were performed to simplify the 3D DCT. The row column (RC) technique is extended to row column frame (RCF) technique, which consist of applying the 1D DCT

on each dimension of the cube using transposition buffers. Many works simplifies the 1D DCT in order to reduce the computation of the 3D DCT. The minimum of multiplications needed to perform a 1-D DCT is 11 [16]. A proposed method by Loeffler [17], gave indeed the most efficient calculations because it requires 11 multiplications and 29 additions. The scaled DCT makes the hardware complexity of DCT designs lower than direct implementation [18-22]. The coder in figure 1 shows the data flow of a 3D DCT based video compressor. The video stream is divided by a group of 8×8×8 cubes, each cube is computed by the 3D DCT block, and it’s followed by a 3D quantization which is a point wise division of each 3D coefficient.

N.Bozˇinovic et al [23] proposed a method based on a translational motion property in the DCT domain. They proposed a motion estimation algorithm based on a plane fitting to high-energy DCT coefficients. The performance of this method is the same in comparison with block matching algorithm [24]. S.saponara [25] proposed an optimization flow to meet real time and low power requirement. He proposed a context aware fast 3D transform and introduces a preprocessor based on statistical rules to avoid the computation of a zero input. J.li et al [26] proposed an analytical model to avoid the computation of redundant DCT coefficients without visual quality degradation. Also S. Boussakta et al [27] proposed a 3-D vector-radix DIF algorithm to compute the 3-D DCT. The algorithm was developed to optimize the number of multiplications; the number of additions was untouched.

In literature few works have been done on hardware implementation for real time and low power processing of the 3D DCT algorithm. The direct FPGA implementation of such complex algorithm needs lot of multiplications and a larger area, in addition the multiplication reduces the performances of the design in comparison with additions, which makes it difficult to respect the real time processing requirements of video standards. Our approach performs the 3D DCT using only additions and zero multiplications. With mathematical operations on a matrix based representation of the DCT we gathered all the multiplications at the end of the transform and merged them on the quantization stage.

This paper is organized as follows: section 2 describes the 3D DCT algorithm and its mathematical formulation. In section 3 we introduce our approach to optimize the (RCF) 3D DCT and give a comparison on the computation saving with other state of the art techniques. In section 4 we detail the properties of the hardware architectures of the 3D DCT, and discuss its FPGA implementation in section 5. Before concluding the work in Section 7, we present some future perspectives in section 6.

II. THE 3D DCT ALGORITHM

The mathematical properties of the 1D DCT [28, 29] is extended to the third dimension. The forward 3D-DCT of an image data volume of dimensions NxLxM is defined by the following equations:

$$F(u,v,w) = K_{u,L}.K_{v,M}.K_{w,N}.\sum_{l=0}^{L-1}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(l,m,n).C_{l,u,L,}.C_{m,v,M}.C_{n,w,N} \quad (1)$$

where f (l,m, n) is the pixel value at position l,m,n of the cube , and F(u,v,w) is the 3d DCT coefficient at position u, v,w of the transformed cube. The quantities K $_{a,d}$ and C $_{a,b,c}$ are given by Eq.2 and Eq.3 respectively.

$$K_{a,d} = \left\{\begin{array}{c} \sqrt{1/d}\ \text{ if } a = 0 \\ \sqrt{2/d}\ \text{ if } 1 \le a \le d-1 \end{array}\right\} C_{a,b,c} = \cos\frac{\pi(2a+1)b}{2c} \quad (2)$$

The inverse transform is given by a set of similar equations as described in what follows:

$$f(l,m,n) = \sum_{i=0}^{L-1}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} K_{u,L}.K_{v,M}.K_{w,N}.F(u,v,w).C_{l,u,L,}.C_{m,v,M}.C_{n,w,N} \quad (3)$$

where K a,d , C a,b,c as defined in Eq.2 and Eq.3 respectively. The optimal dimensions of the cube, used in literature are 8×8×8. wish alows a minimum of rate distortion .we will optimize the 8×8×8 3D DCT in the next chapter.

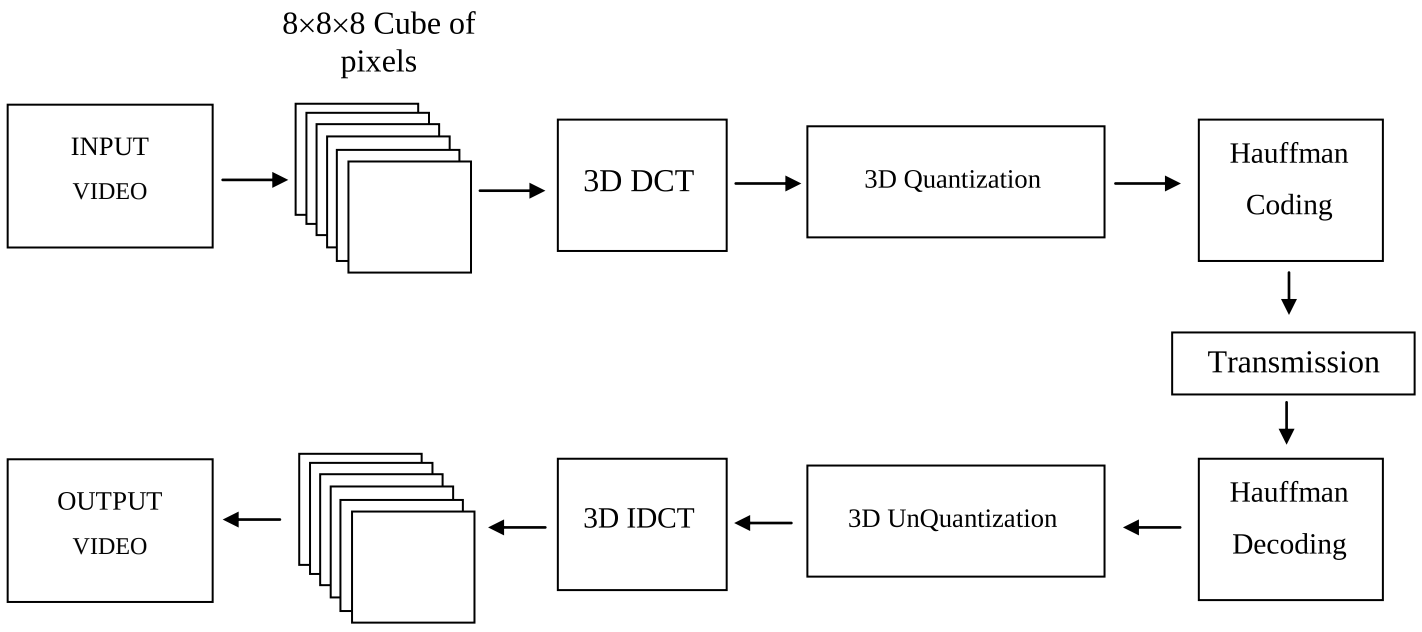


Fig.1 The 3D DCT based video codec

III. Development of Multipliers Merged 3D DCT Transform

The transform can be implemented either directly by using a fast 3D-DCT algorithm as the one presented in [29, 30] or by applying three one dimensional DCTs sequentially on each dimension, based on the separability property (RCF) of multidimensional DCT [28]. In this paper we will use the row, column, frame separability to optimize the algorithm. Our approach relies on a fast 1D SDCT transform [30] which it is given by:

$$C = D \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 0 & 0 & 0 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \end{bmatrix} = D \times T \qquad (4)$$

Where $D = \mathrm{diag}\,(\frac{1}{2\sqrt{2}}, \frac{1}{2}, \frac{1}{2\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2}, \frac{1}{2\sqrt{2}}, \frac{1}{\sqrt{2}})$

It can be verified that the proposed 8×8 transform matrix given by (4) is orthogonal (i.e, $C^{-1} = C' = T' \times D'$) where t denotes the matrix transpose-operation. The matrix T is expressed as the product of 3 matrixes:

T = T3×T2×T1, with:

$$T1=\begin{bmatrix}1&0&0&0&0&0&0&1\\0&1&0&0&0&0&1&0\\0&0&1&0&0&1&0&0\\0&0&0&1&1&0&0&0\\0&0&0&1&1&0&0&0\\0&0&1&0&0&1&0&0\\0&1&0&0&0&0&1&0\\1&0&0&0&0&0&0&1\end{bmatrix} \tag{5}$$

$$T2=\begin{bmatrix}1&0&0&1&0&0&0&0\\0&0&0&0&0&0&1&1\\0&1&1&0&0&1&0&0\\0&0&0&0&0&-1&0&0\\0&1&-1&0&0&0&0&0\\0&0&1&0&0&0&-1&1\\1&0&0&-1&0&0&0&0\\0&0&0&0&-1&0&0&0\end{bmatrix} \tag{6}$$

$$T3=\begin{bmatrix}1&0&1&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&0&0&1&0&1&0\\0&0&0&1&0&0&0&0\\1&0&-1&0&0&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&-1&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix} \tag{7}$$

3.1 Mathematical development

Let define X an 8×8×8 pixels input cube and Y_1, Y_2, Y_3 the 1D, 2D and 3D_DCT outputs respectively. Applying the 1D DCT on the lines of the cube we get:

$$\begin{aligned}\forall\, k \in \{1,8\}:\ Y_{1(*,*,k)} &= (D \times T \times X'_{(*,*,k)})' \\ Y_{1(*,*,k)} &= (D_{pp}\ .\!\times T \times X'_{(*,*,k)})' \\ Y_{1(*,*,k)} &= (X_{(*,*,k)} \times T')\ .\!\times D_{pp}{}'\end{aligned} \tag{8}$$

Where $.\times$ denotes the point wize multiplication , and the matrix D_{pp} is defined by:

$$D_{pp} = [\mathrm{Diag}(D), \mathrm{Diag}(D), \mathrm{Diag}(D), \mathrm{Diag}(D), \mathrm{Diag}(D), \mathrm{Diag}(D), \mathrm{Diag}(D), \mathrm{Diag}(D)]$$

Applying the 1D DCT on the columns of Y_1 we get:

$$\begin{aligned}\forall\, k \in \{1,8\}:\ Y_{2(*,*,k)} &= D \times T \times Y_{1(*,*,k)} \\ Y_{2(*,*,k)} &= D_{pp}\ .\!\times\ (T \times Y_{1(*,*,k)}) \\ Y_{2(*,*,k)} &= D_{pp}\ .\!\times\ (T \times X_{(*,*,k)} \times T')\ .\!\times D_{pp}{}' \\ Y_{2(*,*,k)} &= (D_{pp}\ .\!\times\ D_{pp}{}') \times (T \times X_{(*,*,k)} \times T')\end{aligned} \tag{9}$$

Let's define the matrix S and the cube $M_{(i,j,k)}$ as:

$$S = D_{pp}\ .\!\times\ D_{pp}{}' \tag{10}$$

$$M_{(i,j,k)} = T \times X_{(*,*,k)} \times T' \tag{11}$$

The cube Y_2 will be written as:

$$Y_{2(*,*,k)} = S \times M_{(*,*,k)} \tag{12}$$

Let's define C_j matrixes as:

$$\forall\, j \in \{1,8\}:\; C_j = [S_{(*,j)}, S_{(*,j)}, S_{(*,j)}, S_{(*,j)}, S_{(*,j)}, S_{(*,j)}, S_{(*,j)}, S_{(*,j)}] \tag{13}$$

The cube Y_2 will be rewritten as:

$$\forall\, j \in \{1,8\}: \qquad Y_{2(*,j,*)} = C_j \,.\!\times M_{(*,j,*)} \tag{14}$$

We apply the 1D_DCT on the z axe of the Y_2 cube, and we will obtain:

$$\begin{gathered}
\forall\, j \in \{1,8\}:\; Y_{3(*,j,*)} = (D \times T \times Y_{2(*,j,*)}{}')' \\
Y_{3(*,j,*)} = (D_{pp} \,.\!\times T \times Y_{2(*,j,*)}{}')' \\
Y_{3(*,j,*)} = (Y_{2(*,j,*)} \times T') \,.\!\times D_{pp}{}' \\
Y_{3(*,j,*)} = (Y_{2(*,j,*)} \times T') \,.\!\times D_{pp}{}' \\
Y_{3(*,j,*)} = (C_j \,.\!\times M_{(*,j,*)} \times T') \,.\!\times D_{pp}{}' \\
Y_{3(*,j,*)} = \left(C_j \,.\!\times D'_{pp}\right) .\!\times (M_{(*,j,*)} \times T') \\
Y_{3(*,j,*)} = \text{Multi}_j .\!\times \; \text{3D\_DCT\_Coeff}_{(*,j,*)}
\end{gathered} \tag{15}$$

$$\begin{gathered}
\text{With } \forall\, j \in \{1,8\}:\; \text{Multi}_j = C_j \,.\!\times D'_{pp} \\
\text{3D\_DCT\_Coeff}_{(*,j,*)} = M_{(*,j,*)} \times T'
\end{gathered} \tag{16}$$

The cube Multi contains all the multiplications needed to compute the 3D DCT transform. All the multiplications are gathered at the end of the 3D DCT and will be merged on the quantification stage. The 3D DCT will be computed using only the matrix T which means only additions are used. The quantized 3D DCT coefficient will be calculated as:

$$\begin{aligned}
\text{Quantized\_3D\_DCT}_{(i,j,k)} &= .\frac{\text{Multi}_{(i,j,k)} \,.\!\times\; \text{3D\_DCT\_Coeff}_{(i,j,k)}}{Q_{(i,j,k)}} \\
&= \text{3D\_DCT\_Coeff}_{(i,j,k)} \,.\!\times \left[\, .\frac{\text{Multi}_{(i,j,k)}}{Q_{(i,j,k)}} \right] \\
&= .\frac{\text{3D\_DCT\_Coeff}_{(i,j,k)}}{\text{QNew}_{(i,j,k)}}
\end{aligned} \tag{17}$$

Where $\text{QNew}_{(i,j,k)} = .\frac{Q_{(i,j,k)}}{\text{Multi}_{(i,j,k)}}$ is the new quantitizing cube.

The same 3D_IDCT optimization is done.

let's define the UQuantized an 8×8×8 cube which contains the unquantized 3D DCT coefficients.

$$\text{UQuantized}_{3D_{DCT}\,(i,j,k)} = \text{Quantized\_3D\_DCT}_{(i,j,k)} \,.\!\times\; Q_{(i,j,k)} \tag{18}$$

We apply the 1D IDCT on the unquantized 3D DCT coefficients cube:

$$\begin{aligned}
IY_{2(*,j,*)} &= ((T' \times D') \times \text{UQuantized\_3D\_DCT}_{(i,j,k)}{}')' \\
&= (\text{UQuantized\_3D\_DCT}_{(i,j,k)} \,.\!\times\; D_{pp}) \times T
\end{aligned} \tag{19}$$

Applying 1D IDCT on IY2:

$$\begin{aligned}
\forall\, k \in \{1,8\}:\; IY_{1(*,*,k)} &= \left(T' \times D'\right) \times IY_{2(*,*,k)} \\
&= T' .\!\times (D_{pp}{}' .\!\times IY_{2(*,*,k)})
\end{aligned} \tag{20}$$

Applying 1D IDCT on IY1 :

$$\begin{aligned}
\forall\, k \in \{1,8\}:\; X_{(*,*,k)} &= ((T' \times D') \times IY_{1(*,*,k)}{}')' \\
&= \left(IY_{1(*,*,k)} \,.\!\times\; D_{pp}\right) .\!\times\; T \\
&= \left(\; T' .\!\times (D_{pp}{}' .\!\times IY_{2(*,*,k)}) \,.\!\times\; D_{pp}\right) .\!\times\; T \\
&= \; T' .\!\times (D_{pp}{}' .\!\times IY_{2(*,*,k)} \,.\!\times\; D_{pp}\,) .\!\times\; T \\
&= \; T' .\!\times (D_{pp}{}' .\!\times\; IY_{2(*,*,k)} \,.\!\times\; D_{pp}\,) .\!\times\; T
\end{aligned}$$

$$\begin{aligned} &= T' .\times ((D_{pp}{}' .\times D_{pp}) .\times IY_{2(*,*,k)}) \ .\times T \\ &= T' .\times \ M(*,*,k) .\times T \\ &= T' .\times (T' .\times M(*,*,k)')' \\ &= T' .\times (T' .\times M(*,*,k)')' \end{aligned} \tag{21}$$

(22)

$$\text{Where } \forall k \in \{1,8\} : \ M_{(*,*,k)} = (D_{pp}{}' .\times D_{pp}) .\times IY_{2(*,*,k)}) \\ = S .\times IY_{2(*,*,k)}$$

$$\begin{aligned} \forall j \in \{1,8\} : \ M_{(*,j,*)} &= C_j .\times IY_{2(*,j,*)} \\ &= C_j .\times (\ UQuantized\_3D\_DCT_{(*,j,*)} .\times D_{pp}) \times T \\ &= (\ UQuantized\_3D\_DCT_{(*,j,*)} .\times (C_j .\times D_{pp})) \times T \\ &= Quantized\_3D\_DCT_{(*,j,*)} .\times Q_{(*,j,*)} .\times (C_j .\times D_{pp}) \times T \\ &= (Quantized\_3D\_DCT_{(*,j,*)} .\times (Q_{(*,j,*)} .\times (C_j .\times D_{pp}))) \times T \end{aligned} \tag{23}$$

The unquantizition cube will be calculated as follows:

$$\begin{aligned} UQuantized_{3DDCT} &= Quantized\_3D\_DCT .\times (Q .\times (C_j .\times D_{pp})) \\ &= Quantized\_3D\_DCT .\times UQNew \end{aligned} \tag{24}$$

The new unquantitizing matrix is

$$UQNew_{(i,j,k)} = Q_{(*,j,*)} .\times (C_j .\times D_{pp}) \tag{25}$$

All the multiplications of the 3D IDCT are gathered and merged on the unquantitizing stage. The 3D_IDCT will be computed by the matrix T’. The quantization cube used in this work is defined by Zamarin et al [3] as :

$$Q_{(i,j,k)} = 0.69 \times 2^{(Q\_step/6)} \times \Delta_{max\,(i,j,k)} \tag{26}$$

Where

$$\Delta_{max\,(i,j,k)} = [8,16,23,25,27,29,30,34]/8$$

And Q_step is the quantization step.

Based on the row, column, frame separability (RCF), which means that the 1D_DCT transform is applied on each dimension, we can conclude from the equations 11,16,19 and 21 that the 1D_DCT/IDCT transforms in our case will be computed by the matrix T and T’. The flow graphs of the 1D_DCT/IDCT have been established and shown in figure 2 and figure 3.

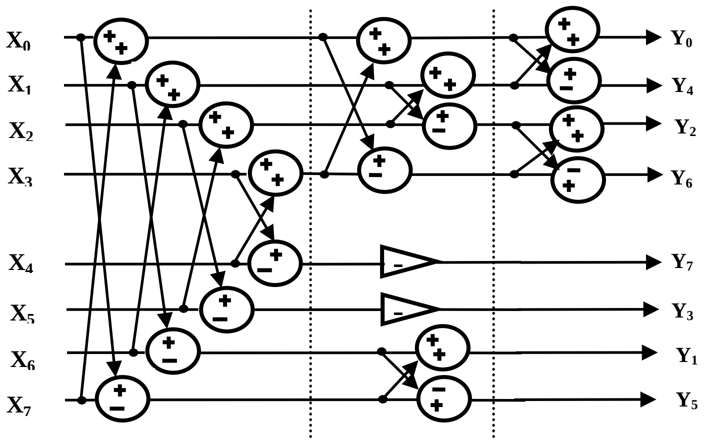


Fig.2 The flow graph of the 1D_DCT

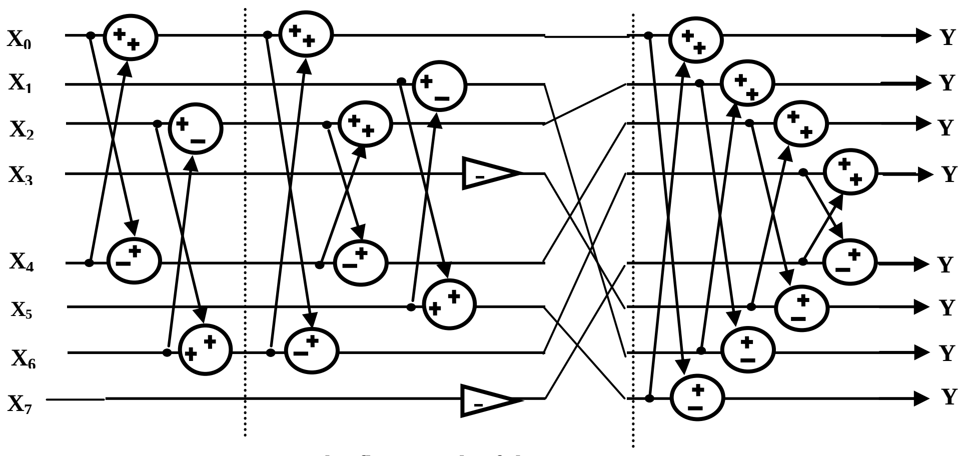


Fig. 3 The flow graph of the 1D_IDCT

3.2 Visual Quality validation

In order to test the quality of video produced by the proposed algorithm, an implementation on Matlab software was done. Figure 4,5 present respectively the PSNR and SSIM quality factors in term of quantization step both for our approach and the conventional RCF (row,column,frame) one. The obtained results of our approach respects the quality requirements of video standards, even if the PSNR is reduced by an average of 0.9 db and SSIM reduced by an average of 0.017 db in comparison with the standard algorithm.

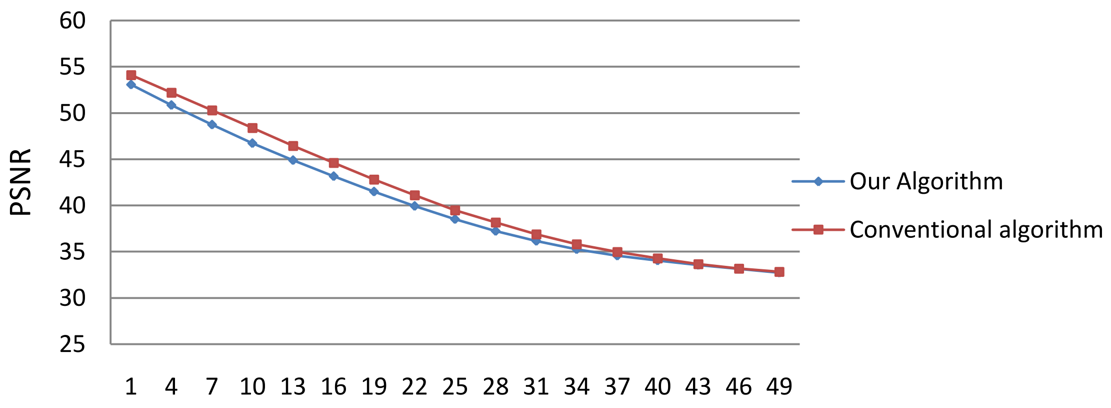


Fig. 4 The evolution of the PSNR results of our method against the conventional one for different quantization steps

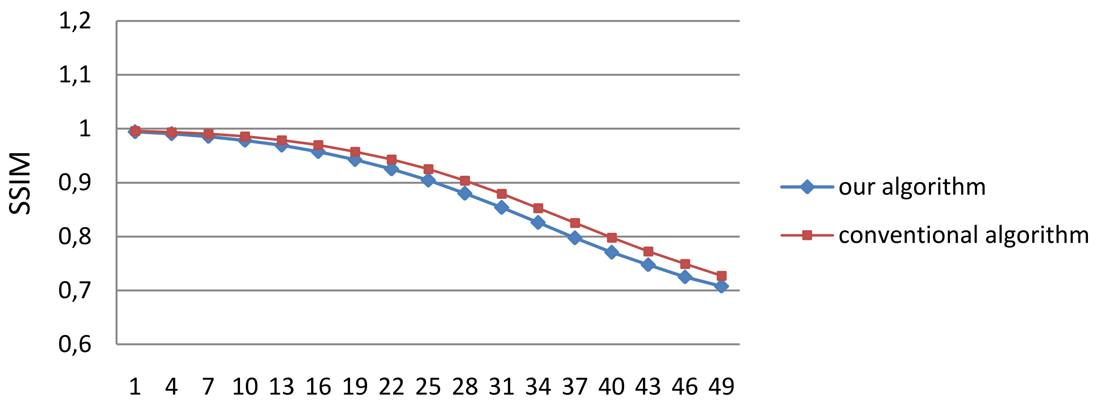


Fig. 5 The evolution of the SSIM results of our method against the conventional one for different quantization steps

3.3 Related works

It is obvious that the number of arithmetic operations needed to compute a 8×8×8 video cube is considerably reduced using our algorithm. In comparison to the separability technique (RCF) of the the standard algorithm we achieved 59% savings on the number of additions, and the multiplications are completely removed. The algorithms introduced in [25,26] uses statistical and heuristic roles to choose the coefficient which needs to be computed. The performances of these methods depend on the motion of the video. They can be very effective on videos with constant background and low motion. These algorithms use preprocessors before each 1D DCT, based on the computation of SAD (Sum of Absolute Difference) between the inputs. The arithmetic operations savings of these methods are evaluated by the percentage of non computed coefficients, and were estimated between 28% and 50% depending on the dynamic of the video sequence. These methods lead to solutions with larger area. A class of algorithms based on polynomial transform was introduced in the literature [31,32, 33]. These algorithms are based on mapping the multidimensional transforms into 1D DCT cosine and polynomial transforms. The algorithm in [32] present an efficient optimization, it achieved 66% savings on the number of multiplications and about 40% savings on the number of additions operation. The main disadvantage of this class of algorithms is they don't have a regular form which makes them unsuitable either for hardware solutions or for software implementation. The method in [27] is a vector radix decimation-in-frequency (VR DIF) optimization, it allows computing 8 3D DCT coefficients by directly exploiting the multiplication redundancies. It achieving 40% savings on the number of multiplications and keeps the same number of-additions. Even if the proposed VR DIF has a regular form and uses one butterfly stage, it must include a first stage of reordering inputs before the computation but achieves the computations without using any transpose buffers. The work [36] combined between the polynomial transform and the approximation of the cosine with the ramanujan numbers. The ramanujan ordered numbers are those which approximate $2\pi/N$ by $2^{-l} + 2^{-m}$, where l and m are integers. The cosine angles can then be computed using Chebyshev type of recursion using only shifters and adders. Another algorithm was introduced using the tensor product [35] it has the same performances as the polynomial transform but is more suitable for hardware implementation. Table 1 shows a comparison between the total of the operations needed to compute a 8×8×8 cube of pixels using our algorithm and other state of the art .the savings of the methods are shown in the table 2. The savings are calculated in reference to the standard RCF 3D DCT algorithm.

Table 1 Comparison of the arithmetic operations needed to process an 8×8×8 cube of pixels

Method	Add/Sub	Shift	Mult
Our algorithm	3840	0	0
Radix-2 RCF [25]	5376	1536	1536
VR DIF [27]	5568	0	1344
Poly Ramanujan[36]	5600	768	0
Tensor Product [37]	5594	0	768
Polynomiale [32]	5600	0	768

Our proposed algorithm is a very simple and fast 3D DCT algorithm and suitable for hardware implementation even if it requires considerable memory as transpose buffers.

Table 2 Comparison of the arithmetic operations savings made by our method and other state of the art techniques

Method	Add/Sub	Mult
Our algorithm	59%	100%
Context aware [25,26]	28% - 50%	
Radix-2 RCF [25]	43%	75%
VR DIF [27]	41%	89%
Polynomiale [32]	40%	94%
Poly Ramanujan[36]	40%	94%
Tensor Product [37]	41%	94%

IV. FPGA Implementation and discussion

4.1 Hardware Architectures

To perform the 3D DCT using the separability technique (RCF), the 1D_DCT/IDCT must be applied on each dimension of the cube, this leads to the standards architectures based on 1D_DCT/IDCT. The difference between architectures is the number of 1D_DCT/IDCT blocks, this impacts it's properties such as the throughput, area, power consumption and clock frequency. The state of the art architectures are mainly parallel, serial and iterative. We modified the parallel architecture developed in [25]. The architectures are presented and analyzed based on their properties. Table 3 summarizes the properties of each architecture.

4.2.1. Full parallel

The architecture shown in figure 6 gives a maximum parallelization degree of the 3D-DCT algorithm. It allows us to process 8×8 pixels at every clock edge. The number of 1D DCT block is 3×8. To compute the second stage an 8×8 memory registers is needed for every 1D DCT. The memory is designed as follows: The lines of the memory are right shift registers, every register is used to store a processed pixel. The columns of the memory are also shift registers. The memory works as follows: During 8 Clock cycles, the outputs of every 1D DCT in the first stage are stored in the first column of every corresponding memory and the old values are right shifted. The inputs of every 1D DCT in the second stage are the last line of every memory registers. At every Clock edge the old values in the memory are shifted down vertically during 8 cycles and fed to the second stage to compute the 1D DCT on the columns of the matrix. This memory has a latency of 8 cycles, because the second stage has to wait 8 cycles before starting the computations and the first stage has to wait 8 cycles when the second stage is processing. To eliminate this latency we doubled the memory to build a ping pong memory with a more complex memory controller. The memory works as follows: when the first stage is writing on a side of the memory the second stage is reading from the other during 8 cycles, after that the stages switch the sides. Using this modification we will process, for each 1D DCT, 8 pixels every single clock cycle at the expense of a doubled memory area. Figure 7 shows the detailed memory register scheme. In the second stage, 1×8 memory register is used for each 1D DCT. The outputs of every DCT in the second stage are stored in a buffer of 8 registers every register stores a computed pixel. The registers are named Ri with i Є {0,…,7} where i is the number of the DCT block and Ri(j) is the pixel number j stored in the register Ri , with j Є {0,…,7}. In the third stage 8 DCTs are necessary to compute in parallel the outputs of the second stage. The inputs of the 1D DCT number i will be: Rj(i) where j Є {0,…,7} , for example the inputs of the first DCT are the first pixels of each registers . This memory present a latency of 1 clock cycle , to eliminate this latency we built, like in the first memory a ping pong memory wich allows us to compute 8×8 pixels per clock cycle. The Work in [34] uses RAM memory to transpose data. 8 SRAM are used before the third stage every SRAM store 8×8 pixels, the memory addresses are chosen carefully and are extremely important as explained in [35].The total size of memory word registers needed is $2\times8^2\times$ (8+1). The detailed hardware design of this architecture is illustrated in figure 8.

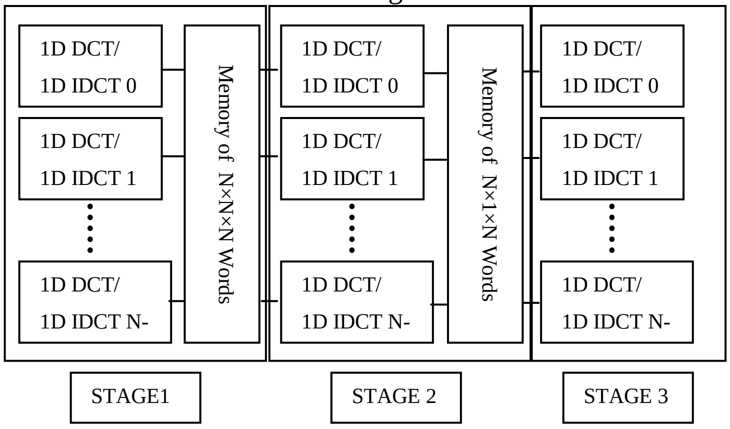


Fig.6 The 3 stages full parallel architecture of the 3D_DCT

Since we can produce 8×8 pixels per every clock cycle the throutput of this architecture is 8×8×Fclk, where Fclk is the operating frequency of the 1D DCT block. A parallel architecture was introduced in [25]. This architecture uses 2×8×1D_DCT/IDCT in the two first stages and only one at the third stage. Since the second stage produces 8×8 coefficients at every clock edge, the last stage must compute them all in the same time, this leads to the use of two clock domains, the frequency of the first is fclk used by the DCT blocks in the first and second stages and 8×fclk used by the DCT in the last stage.

4.2.2 Serial architecture

This architecture uses 3 1D DCT blocks as shown in figure 9, it compute 8 pixels per clock cycle. The lines of each bloc of the cube are multiplexed on the input of the system. The memory used remains the same as before with an additional logic to do the multiplexing of the output into the corresponding 8×8 buffers. In this stage we use a Memory of 8×1×8 Words to store the results of the second one so the total memory needed using ping pong memory registers is : $2\times8^2\times(8+1)$ with additional controller. Every 8^2 and 1×8 memory block in each stage is used to store the results of a transformed frame .The throughput of this architecture is Fclk.

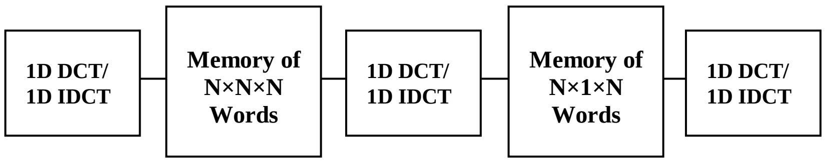


Fig. 9 The serial architecture of the 3D_DCT

4.2.3 Iterative architecture

The iterative scheme uses only one 1D DCT block, which is used to compute the three levels of the 3D DCT. The 1D_DCT/IDCT is used to compute first all the lines of the cube, then to compute all columns of the resulting cube and finally used to compute the transform on the third dimension. The hardware design of this system is shown in the figure 10. The throughput of the system is Fclk/3.Table 3 summarizes the complexity of the 3 architectures.

Table 3 Complexity of the different architectures

Architecture	1D DCT Blocks	Throutput	Nbr of Clock cycles to process a cube	Memory register of 8bits
Parallel	3×8	8×8×Fclk	8	$2\times8^2\times(8+1)$
Serial	3	Fclk	64	$2\times8^2\times(8+1)$
Iteratif	1	Fclk/3	192	8^3

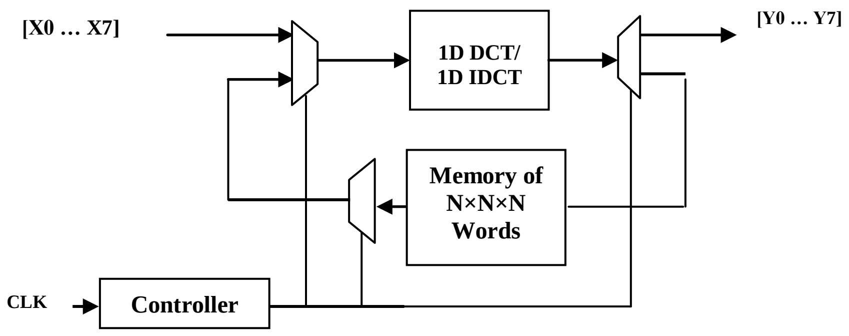


Fig.10 The iterative architecture of the 3D_DCT

V. IMPLEMENTATION RESULTS

In this section we present the implementation results of the 3D DCT introduced. In the literature we did not find much works on FPGA implementation of the 3D DCT. The implementation has been achieved on Altera's 180 nm cyclone 2 chip using the VHDL hardware language, and the synthesis and implementation tool Quartus II. The power consumption was obtained by the Power Play tools analyzer of Quartus tools. The implementation will allow us to evaluate the performances of different proposed architectures and check whether they meet the real time requirements of video standards.

5.1 logic usage and frequency

In this part we will present and discuss the frequency and the logic usage of the FPGA implementation. There is an increase of the used logic when going respectively from iterative, serial to full parallel architecture. This is essentially due to the number of 1D_DCT/IDCT blocks used to achieve better parallelization level. This can be explained by the same needed storing capacity for all the architectures and the decreasing complexity of logic used before and after each stage of memory registers. The total registers used include those used to pipeline data in the 1D_DCT blocks. Table 4 summarizes the resources of each architecture.

Table 4 Logic usage of the different architectures

Design	Parallel	Serial	Iterative
Registers	13250	9550	4457
Combinational functions	15320	10820	7502

The parallel and serial architectures can operate with high frequencies up to 300 MHz, 200 MHz respectively, this allows the processing of the most demanding video standard. Table 5 summarizes the frequency requirement to real time processing of some video standards. The frequency requirements calculated are based on the throughput, the size of the video. The video format is 4:2:0 and the sampling frequency is 30 Hz.

Table 5 Frequency needed to real time processing different video standards

	QCIF 176*144	SIF 352*288	CCIR-TV 576*704	16CIF 1408*1152	HDTV 1152*1928
Parallel	0.05 MHz	0.21 MHz	0.85 MHz	3.42 MHz	4.68 MHz
Serial	0.42 MHz	1.71 MHz	6.84 MHz	27.36 MHz	37.48 MHz
Iteratif	1.28 MHz	5.13 MHz	20.52 MHz	82.08 MHz	112.44 MHz

Analyzing the results of table 5, we can observe that when the parallelization level increases the frequency requirement is reduced. This is due essentially to the increasing of the throughput which allows more data to be processed at each clock cycle. Table 6 summarizes the max frequency of each architecture. According to table 5 and table 6 we can conclude that serial and parallel architectures can achieve the real time processing of the entire video format. The iterative architecture can reach a frequency up to 100 MHz while to process the HDTV video standard the design must run with 113 MHz according to table 5. This means that the iterative architecture can process the entire video format except the HDTV. Architecture reported in [38], works with an operating frequency of 127 MHz, it’s based on a distributed arithmetic based algorithm which requires complex logic to achieve accumulations in the final stage. The work in [39] has a structure composed of an interface and three successive stages. The interface reads the data to be calculated. Stage l performs the multiplication operations. Stage 2 performs the accumulation of stage 1 outputs, and finally stage 3 stores the intermediate results and output the resultant DCT coefficients. The architecture in [39] has an operating frequency of 50 MHz, and the architecture presented in [40] works with a max frequency of 12.5 MHz in the encoder side and 24.15 MHz at the decoder side. A comparison between the operating frequencies is shown in table 7.

Table 6 The max operating frequency of the different architectures

Design	Parallel	Serial	Iterative
Frequency	310 Mhz	202 Mhz	106 Mhz

Table 7 Comparison between operating frequency of our methods and others in the stat of the art

Architectures	**Our work**			**Work**	**Work**	**Work**
	parallel	**serial**	**iterative**	**in [38]**	**in [39]**	**in [40]**
Op Frequency	310 Mhz	202 Mhz	106 Mhz	127 Mhz	50 Mhz	12.5Mhz encoder 24.55 Mhz decoder

5.2Power consumption

The power consumption corresponding to the frequency required for r real time processing of the video formats is summarized in table 8. The minimum power consumption to process a same video format is consumed by the full parallel architecture and the maximum with the iterative one. These results can be explained by the fact that the operating frequency used to run the full parallel architecture is very low in comparison with the one used to run the iterative one, thanks to the throughput and to the fact that the used resources are note considerably increased from a level of parallelism to another as explained above. All the cited reasons and taking in consideration the simplicity of the algorithm, we can easily conclude that increasing the parallelization level reduces effectively the power consumption. Analyzing all the implementation results and taking in consideration the complexity of our optimized method, the choice between the different architectures is application oriented. The full parallel architectures uses the largest area but achieves good performances and maximizes the throughput, it consumes less power which makes it suitable for application requiring lower power consumption .The serial architecture complexity is lower than the parallel architectures but consumes more power, and present important latency namely 64 clock cycle to process an 8×8×8 cube. Our implementation of parallel and serial architectures allows real time processing of different video format. The iterative architecture uses the minimum area and has lower complexity, at the expense of low throughput which increases the constraints for to real time processing of presented video formats, thus the power consumption becomes the highest in comparison with the others.

Table 8 Power consumption of the different architectures in different video formats

Architectures/Video's formats	QCIF 176*144	SIF 352*288	CCIR-TV 576*704	16CIF 1408*1152	HDTV 1152*1928
Parallel	0.02	0.07	0.3	1.21	1.65
Serial	0.12	0.51	2.12	7.01	12.3
Iterative	0.51	2.62	10.46	41,89	Not defined

VI. Future Work

The next work will handle a multiplier less multidimensional DCT based on the concept presented in this work. We will work also on other hardware architectures to offer more choice depending on the applications. We will study how we can use the various architectures of the 3D DCT to build a reconfigurable video compressor.

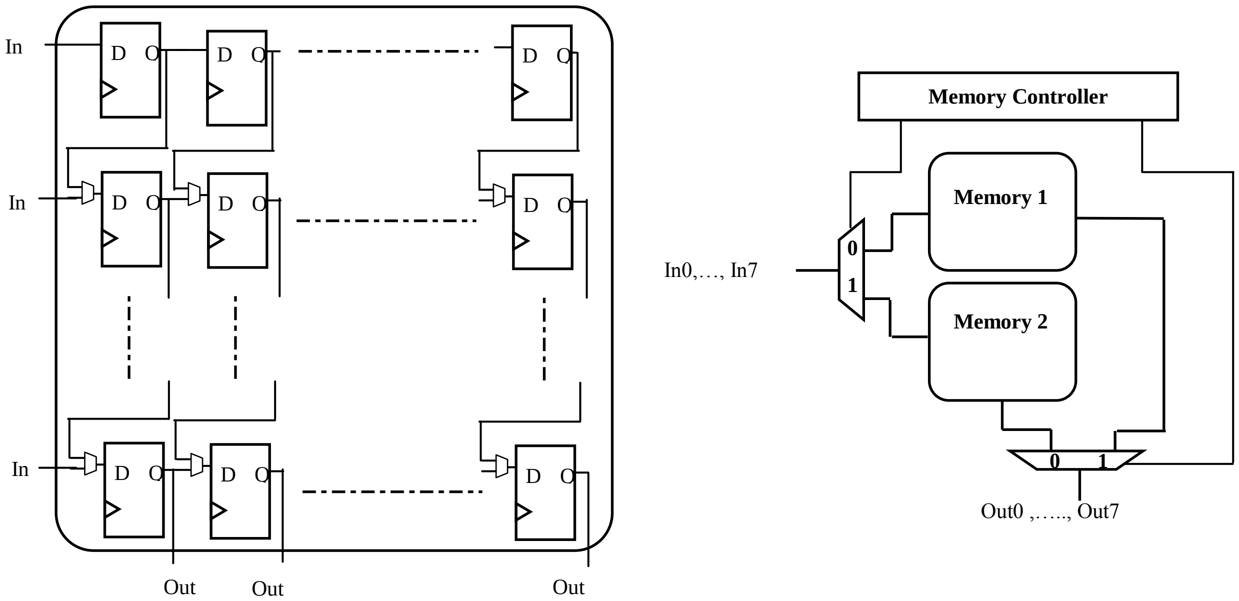


Fig. 7 The Hardware Architecture of the transposition memory

VII. Conclusion

An optimized RCF 3D DCT/IDCT is presented in this paper. The main contribution of our work has been the gathering of all multiplications needed to achieve the 3D DCT/IDCT at the end of the transform, and then merged in the quantization stage. This makes the transform possible using only additions. Consequently, The new proposed fast (RCF) 3D DCT/IDCT saves 59% on the number of additions and eliminates totally the multiplications which increase significantly its performance. In addition new quantitizing and unquantitizing cubes are introduced. A study of the implementation the 3D DCT/IDCT architectures leads us to conclude that the parallel architecture has the lowest power consumption and it offers low operating frequencies while it compute video standards with high resolution. The serial one can compute also all the video standards using reduced resources. The iterative architecture is the best solution for area optimization at the expense of important power consumption. The presented fast RCF 3D DCT/IDCT in this paper is an efficient algorithm which allows us to meeting the real time requirement with the required visual quality.

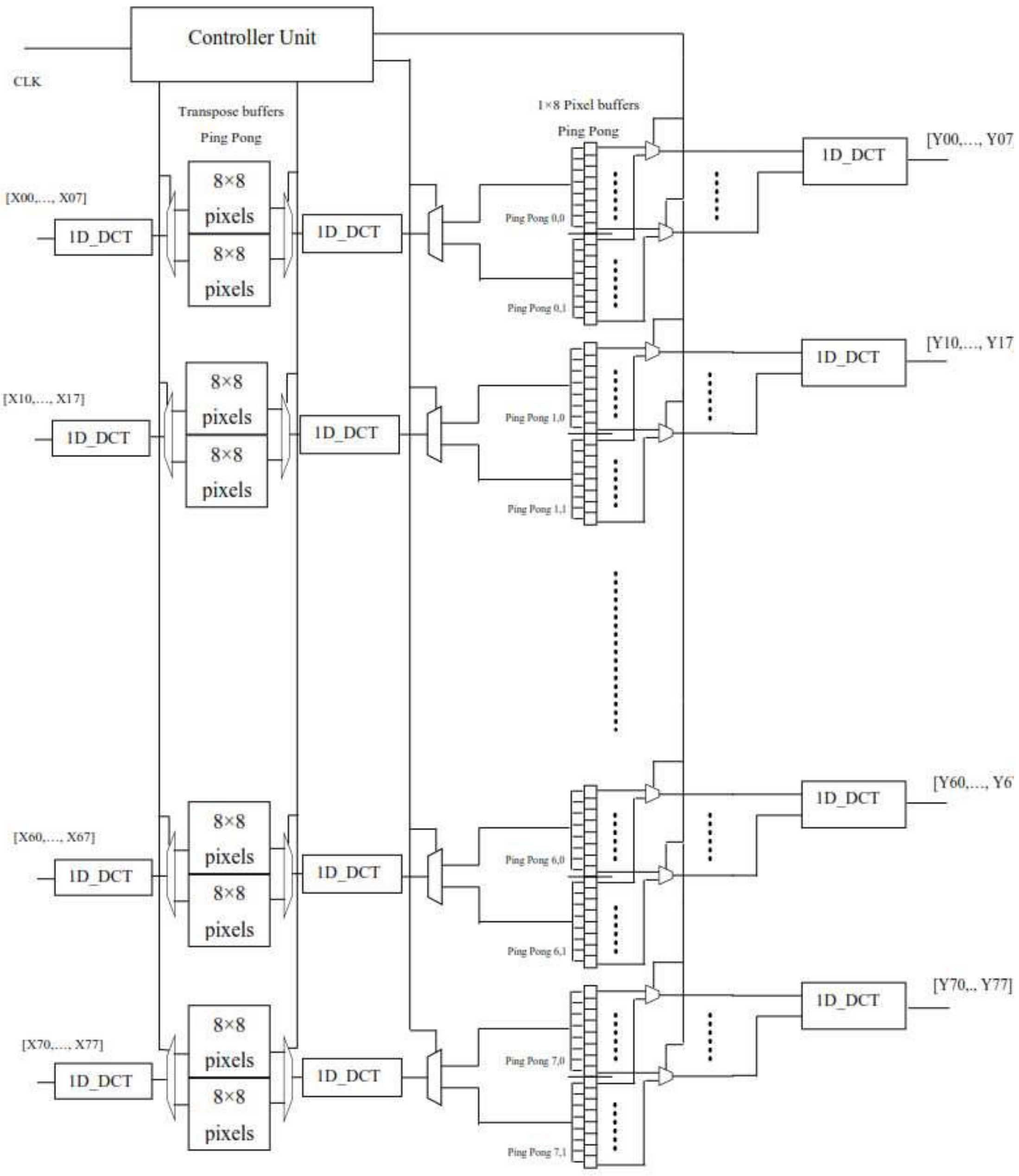


Fig. 8 The Hardware Architecture of full parallel architecture of the 3D_DCT

References

[1]. Saponara, S., et al.: Dynamic control of motion estimation search parameters for low complex H.264 video coding. IEEE. Trans.Consum. Electron. 52(1), 232–239 (2006)
[2] Saponara, S., Denolf, K., Blanch, C., Lafruit, G., Bormans, J.: Performance and complexity co-evaluation of the advanced video coding standard for cost-effective multimedia communications.J. Adv. Signal. Process. 2004(2), 220–235 (2004)
[3] Zamarin Marco , Milani, S, Zanuttigh P, Cortelazzo, G.M. (2010) A Novel Multi-View Image Coding Scheme based on View-Warping and 3D-DCT ", Journal of Visual Communication and Image representation, vol: 21, issue: 5-6, pages: 462-473
[4] Chan R, Lee M (1997) 3D-DCT quantization as a compression technique for video sequences, International Conference on Virtual Systems and Multimedia

[5] Li J,Takala, J, Gabbouj M, Chen H (2007) Variable temporal length 3D DCT-DWT based video coding , IEEE ISPCAS 2007, pp. 506–509
[6] J. Li et al, (2008) Modeling of 3-D DCT Coefficients for Fast Video Encoding", ISCCSP 2008, Malta
[7] G. Gunlu et al, (2009) "Feature extraction and discriminating feature selection for 3D face recognition", IEEE ISCIS'09, pp. 44-49
[8] Y. Fu et al (2009) "Secure Spread Image Watermarking Scheme in 3D-DCT Domain", IEEE CISP'09, pp. 1-4
[9] R. Westwater and B. Furht, (1996) " The XYZ Algorithm fro Real-Time Compression of Full-Motion Video", Journal of Real-Time Imaging, Vol. 2, No. 1, February, pp. 19-34.
[10] R. Westwater and B. Furht,(1996) "Real-Time Video Compression: Techniques and Algorithms," Kluwer Academic Publishers, Norwell, MA,.
[11] T. Ouni et al., (2009) "New low complexity DCT based video compression method", IEEE ICT'09, pp. 202-207
[12] M Bhaskaranand et al, (2009) "Distributions of 3D DCT coefficients for video", IEEE ICASSP'09, pp. 793-796
[13] J. Li et al, (2008) "Modeling of 3-D DCT coefficients for fast video encoding", IEEE ISCCSP'08, pp. 634-638
[14] V. Testoni et al, (2008) "Three-Dimensional Transforms and Entropy Coders for a Fast Embedded Color Video Codec", SIBGRAPI'08, pp. 147-154
[15] Radu Radescu, George Paunaa (2006) New approach to video compression using the 3D-DCT Algorithm , 8th International Conference on Development And Application Systems , Suceava , Romania, May 25-27
[16] Andraka R (1998) "A survey of CORDIC algorithms for FPGA based computers". Proceedings of the ACM/ SIGDA sixth international symposium on Field programmable gate arrays, pp 191–200
[17] Arai Y, Agui T, Nakajima M (1988) "A fast DCT-SQ scheme for images". Trans IEICE E71:1095–1097
[18] Hatim.A,Belkouch .S et al (2013) "Efficient hardware architecture for direct 2D DCT computation and its FPGA implementation" International conference on Microelectronics (ICM) circuits and systems 2013
[19] Hatim.A,Belkouch .S et al (2011) "FPGA implementation of a pipelined 2D-DCT and simplified quantization for real-time applications" International Conference on Multimedia Computing and System ICMCS 11
[20] Kassem A, Hamad M, Haidamous E (2009) Image compression on FPGA using DCT. Intern Conf Advance Comput Tools Engineer Appl ACTEA pp 320–323
[21] Xiuhua J, Caiming Z, Xuefen Z (2011) An efficient joint implementation of three stages for fast computation of color space conversation in image coding/decoding. Multimedia Tool Appl pp 1–15
[22] A. Hatim , S. Belkouch et al "Design optimization of the quantization and a pipelined 2D-DCT for real-time applications". Multimedia Tool Appl
[23] Nikola Boz ˇinovic, Janusz Konrad (2005) "Motion analysis in 3D DCT domain and its application to video coding" Elsevier, Signal Processing: Image Communication 20 (2005) 510–528.
[24] Ahmed Z, Hussain A.J, Al-Jumeily D. (2011) Mean Predictive Block Matching (MPBM) for fast block-matching motion estimation, Workshop on, Visual Information Processing (EUVIP), 2011 3rd European . 4-6 July 2011 pp 67 - 72
[25] S.Saponara et al (2011) "Real-time and low-power processing of 3D direct/inverse discrete cosine transform for low-complexity video codec". J Real-Time Image Proc DOI 10.1007/s11554-010-0174-5
[26] J. Li et al, (2008) "Simplified video coding for digital mobile devices", IEEE ICSP'08, 1247-1250
[27] Said Boussakta ,Hamoud O. Alshibami,(2004) "IEEE Fast Algorithm for the 3-D DCT-II" , IEEE transactions on signal processing, VOL. 52, No. 4, April
[28] K. R. Rao and P. C. Yip, (2001) The Transform and Data Compression Handbook", Boca Raton: CRC Press LLC, ch.4
[29] Supriya Sawant and Donald A. Adjeroh (2011) Balanced Multiple Description Coding for 3D DCT Video" IEEE transactions on broadcasting, vol . 57, No 4
[30] S.Bougzeel ,M.Omair Ahmed and M.NS.Swamy (2009) A fast 8×8 for image compressing" Proc. of the International Conference on Microelectronics, pp.74-77
[31]Y. Zeng, G. Bi, and A. R. Leyman (2000) New polynomial transform algorithm for multidimensional DCT," IEEE Trans. Signal Process., vol. 48, pp. 2814–2821
[32]Y. Zeng, G. Bi, and A. C. Kot (2000) New algorithm for multidimensional type-III DCT" IEEE Trans. Circuits and Syst. vol. 47, pp. 1523–1529
[33]Y. Zeng, G. Bi, and Z. Lin (2002) "Combined polynomial transform and radix-q algorithm for multi-dimensional DCT-III," Multidimensional Syst. Signal Process., vol. 13, pp. 79–99
[34] Fan. Y, Shan-Ann C, Kuo-Gi W, Jun-Lin Y, (2012) 3D-DCT Chip Design for 3D Multi-view Video Compression .Appl. Math. Inf. Sci. 6 No. 2S pp. 567S-572S

[35] A. Sinha, A. Karmakar, K. Maiti, and P. Halder, (2001) A Reconfigurable Architecture for a Class of Digital Signal/Image Processing Applications, IEEE Conference on Computers and Signal Processing, Vol. 1, pp. 71-74

[36] K.Geetha,, M.Uttarakumari (2012) “Multidimensional Fast Multiplierless DCT Algorithm Using Ramanujan Ordered Numbers” International Journal of Computer and Electrical Engineering, Vol. 4, No. 4, August .

[37] C.Xinjian , Q.Dai, and C.Li (2003) A Fast Algorithm for Computing Multidimensional DCT on Certain Small Sizes ,IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 51, NO. 1, JANUARY

[38] M. B. Abdelhalim and A. E. Salama (2003) Implementation of 3D DCT based Video Encoder Decoder System”. IEEE International Symposium on signals, Circuits and Systems, vol2 pp 389-392

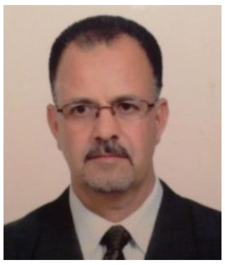
[39]A.Aggoun (2013) International Journal of Advances in Engineering & Technology, Vol 6, Issue 2, pp. 648-658

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

BIBLIOGRAPHY

Hatim Anas was born in Marrakech, Morocco, in 1987. He received the Baccalaureate degree from Ibn toumert, the Electrical engineer degree from the National School of applied science. He works at université international de rabat as R&D engineer on electrical field. He is a PHD student at the National School of applied science, Cadi ayaad University. His research interests are primarily in the area of video compression, optimization using FPGA.

Said Belkouch has completed his PhD in Microelectronics at University Joseph Fourier-Grenoble in France in 1989. From 1989 to 2003, he worked respectively as Assistant Researcher at University of Sherbrooke in Canada, Research Officer at National Council of Canada, and embedded ASICs Design Engineering at Tundra Semiconductor Corporation, Ottawa, Canada (Tundra acquired recently by Integrated Devices Technology Company). Since 2003, He is professor at Electrical Engineering Department, National School of Applied Sciences-Marrakech, Morocco. His area of research includes embedded systems and microelectronics. He has published several research papers in Journals and Proceedings.



Moha M’Rabet Hassani. He received the a doctorate thesis in automatic from Nice University, France, in 1982 and Ph. D degree in electrical engineering from Sherbrook university, Canada, in 1992. He is now Professor in the Faculty of Sciences Semlalia, Cadi Ayyad University, and Marrakech, Morocco. He heads the Technology, Electronics and Instrumentation Lab. His research interests include statistical signal processing, nonlinear system, identification and equalization fields.