FORWARD AND INVERSE 2-D DCT ARCHITECTURES TARGETING HDTV FOR H.264/AVC VIDEO COMPRESSION STANDARD

L. AGOSTINI[†‡], R. PORTO[†], M. PORTO[†], T. SILVA[†], L. ROSA[†], J. GÜNTZEL[†], I. SILVA[§], and S. BAMPI[‡]

[†] *Group of Architectures and Integrated Circuits (GACI), UFPel, Pelotas, Brazil*
{agostini, rogerecp, porto, thleal, lrosa.ifm, guntzel} @ufpel.edu.br
[‡] *Microelectronics Group (GME), UFRGS, Porto Alegre, Brazil*
{agostini, bampi}@inf.ufrgs.br
[§] *DIMAp, UFRN, Natal, Brazil*
ivan@dimap.ufrn.br

Abstract — **This paper presents the architecture and the VHDL design of the integer Two-Dimensional Discrete Cosine Transform (2-D DCT) used in the H.264/AVC codecs. The forward and inverse 2-D DCT architectures were designed and their synthesis results mapped to Altera FPGAs are presented. The 2-D DCT calculation is performed by exploring the separability property, in such way, each 2-D DCT architecture is divided in two 1-D DCT calculations that are joined through a transpose buffer. The 1-D DCT transforms implemented and herein described are multiplierless, hence optimized shift-add operations are used. The architectures have a dedicated pipeline, optimized to process one input data per clock cycle. These architectures are able to cope with H.264/AVC encoder or decoder requirements targeting High Definition Digital Television (HDTV), with 1920x1080 pixel/frame at 30 frames per second.**

Keywords — **H.264/AVC video compression, dedicated hardware for video compression, 2-D FDCT, 2-D IDCT, integer transforms.**

I. INTRODUCTION

The H.264/AVC is the new international standard for video compression (Joint Video Team, 2003). The main characteristic of the H.264/AVC is to provide significantly higher compression rates than the previous standards as MPEG-2, MPEG-4 and H.263 (Wiegand *et al.*, 2003; Sullivan *et al.*, 2004; Sullivan and Wiegand, 2005). The H.264, also known as MPEG4 part 10 or AVC, was developed to supply the processing rates demanded by video applications in high definition.

An important factor that differentiates the H.264/AVC from other video compression standards is the use of integer transforms (Richardson, 2003; Malvar *et al.*, 2003). The integer transforms have a final result that is an approximation of the real transforms result. The H.264/AVC standard defines the use of two different integer transforms: 2-D DCT and Hadamard. These integer transforms are multiplierless and all operations could be realized using just additions, subtractions and shift-adds.

Differently from other standards, the H.264/AVC defines a minimum image partition matrix (called block) with 4 x 4 samples. This option reduces the block and borders artifacts in the compressed image.

With the use of integer coefficients and 4x4 input blocks, the complexity of the transforms in the H.264/AVC is significantly reduced in relation to other image and video compression standards.

This paper will present the architectural design of a 2-D FDCT and a 2-D IDCT targeting the H.264/AVC compression standard. The 2-D DCT separability property was used in these architectures: the 2-D transforms are calculated by applying twice the transforms in one dimension which are joined by a transpose buffer. The designed architectures were described in VHDL (Airan *et al.*, 1994) and synthesized to Altera FPGAs (Altera, 2006). The synthesis results are also presented.

This paper is organized as follows. Section 2 presented an introduction to the H.264/AVC standard. Section 3 details the integer transform used in this standard. Section 4 presents the designed architectures. The synthesis results for the designed architectures are showed in section 5. Section 6 presets a comparison with related works. Conclusions and future works are presented in section 7.

II. THE H.264/AVC STANDARD

An increasing number of services and the growing popularity of high definition TV are creating greater needs for higher coding efficiency. As a result of the ongoing demand for better compression performance the latest video coding standard, the H.264/AVC (Advanced Video Coding) (Joint Video Team, 2003) was developed. The H.264/AVC is also known as MPEG-4 Part 10 (Wiegand *et al.*, 2003; Sullivan *et al.*, 2004; Richardson, 2003).

H.264/AVC uses the state-of-the-art coding tools and provides enhanced coding efficiency for a wide range of applications (Wiegand *et al.*, 2003; Sullivan *et* *al.*, 2004, Richardson, 2003). This standard was created to improve the coding efficiency by a factor of, at least, two (on average) over MPEG-2 – the nowadays most widely used video coding standard (Sullivan *et al.*, 2004; Kamaci and Altunbasak, 2003). In that way, it provides significantly higher compression rates than the previous standards and the most interesting balance between the coding efficiency, implementation complexity and cost, besides of to introducing many new features. But as might be expected, the increase in coding efficiency and coding flexibility comes with an increase in global compressor complexity with respect to earlier standards (Puri *et al.*, 2004).

Two distinct layers are used in the H.264/AVC standard: a video coding layer (VCL) and a network adaptation layer (NAL) (Joint Video Team, 2003; Sullivan and Wiegand, 2005; Richardson, 2003). The standard also specifies the use of an improved de-blocking filter within the motion compensation loop in order to reduce visual artifacts and improve prediction. The standard also features a more complex and efficient context-based arithmetic coding (CABAC) for entropy coding. With the exception of the de-blocking filter, most of the basic functional elements (prediction, transform, quantization, entropy coding) are present in previous standards (MPEG-1, MPEG-2, MPEG-4, H.261, H.263) but the important changes in H.264/AVC occur in the internal construction of each functional block (Richardson, 2003).

A block diagram of the H.264/AVC encoder is presented in Fig. 1. The main blocks of the encoder (Richardson, 2003), as shown in Fig. 1, are motion estimation (ME), motion compensation (MC), intra prediction, forward (T) and inverse (T^{-1}) transforms, forward (Q) and inverse (Q^{-1}) quantization, entropy coder and de-blocking filter.

The decoder blocks, presented in Fig. 2, are a sub-group of the coder blocks (Richardson, 2003), including entropy decoder, inverse quantization (Q^{-1}) inverse transforms (T^{-1}), motion compensation (MC), intra prediction and the de-blocking filter.

This work focuses just on the design of a part of the forward and the inverse transforms (T and T^{-1} in Fig. 1 and T^{-1} in Fig. 2).

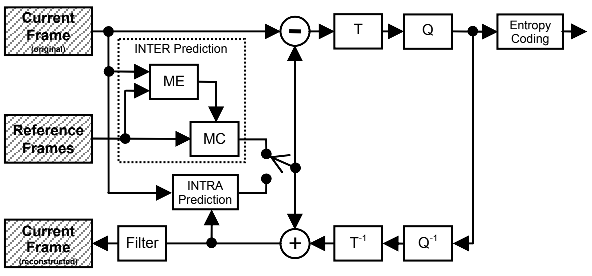


Figure 1. Block diagram of a H.264/AVC coder

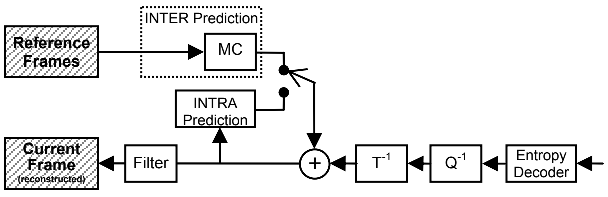


Figure 2. Block diagram of a H.264/AVC decoder

The forward transform block (T) uses three different two dimensional transforms, depending on the type of input data. These transforms are 4x4 2-D FDCT, 4x4 2-D forward Hadamard and 2x2 2-D forward Hadamard (Richardson, 2003). The inverse transform block (T^{-1}) uses also three different two dimensional transforms: 4x4 2-D IDCT, 4x4 2-D inverse Hadamard and 2x2 2-D inverse Hadamard (Richardson, 2003). T and T^{-1} blocks must synchronize the operation of their three transforms to generate a correct data flow in their outputs.

This work presents the architecture design of the forward and inverse 2-D DCT to be used in H.264/AVC coders and decoders.

An advantage of the H.264/AVC is the simplicity of its transform (Malvar *et al.*, 2003; Wien, 2003; Wang *et al.*, 2003). The standard adopts integer transforms. Since the coefficients are integer, there will be no mismatch between encoder and decoder. Integer coefficients also imply in a simpler hardware implementation. This transform also reduces the occurrence of blocking and ringing artifacts (Kamaci and Altunbasak, 2003).

III. THE H.264/AVC INTEGER 2-D DCT

The two dimensional DCT (2-D DCT) is a mathematical tool that is used to transform the information from the space domain to the frequency domain. In image and video compression, many standards like JPEG, MPEG and H.264/AVC use the 2-D DCT to transform the input data to the frequency domain. The information represented in the frequency domain could be handled to discard the frequencies that are less important to the human visual system perception. This operation reduces the amount of information used to represent the image or video, allowing higher compres-sion rates with little impact in the image quality.

There are many image and video compression standards that apply the 2-D DCT over 8x8 input blocks. However the DCT used in the H.264/AVC is applied over 4x4 input blocks, reducing the computational complexity of this calculation.

The H.264/AVC is the first video standard that uses an integer transform (Sullivan and Wiegand, 2005; Malvar *et al.*, 2003). The previous standards use floating point transforms. The final results of the integer transform are an approximation of the real DCT results, but this integer conversion causes a minimal loss of accuracy in this calculation (Richardson, 2003). The implementation of an integer transform generates some important advantages. The first one is that this is a multiplierless transform and just additions and shifts are needed to perform its calculations. The second advantage is that this integer solution reduces the computational complexity of the 2-D DCT calculations, thus simplifying the task of the hardware implementation. The third advantage is that the use of integer numbers makes the transformation faster than those that use floating point numbers and can reduce signifi-cantly the power consumption, allowing a wide use of the H.264/AVC in mobile applications (Richardson, 2003).

The forward 2-D DCT adopted in this paper uses the matrix operation presented in (1). This is an integer and scaled formula to calculate the 2-D DCT. This alternative was proposed by Malvar (2003) and reduces the 2-D DCT calculation complexity, once the scalar multiplication (⊗ **E**) is joined with the quantization process without increase the complexity of this compression step (Richardson, 2003). Then, the hardware design makes just the $\mathbf{CXC}^{\mathbf{T}}$ calculation that is a matrix multiplication. **C** is the 1-D DCT integer and scaled matrix, **X** is the 4x4 input matrixes and $\mathbf{C}^{\mathbf{T}}$ is the transposed of the 1-D DCT integer and scaled matrix.

$$Y = C\,X\,C^{T} \otimes E =$$

$$= \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} \\ X \\ \\ \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \right) \otimes \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix} \quad (1)$$

The inverse transform is given by the equation showed in (2). Note that just additions and shifts are also needed to implement the inverse 2-D DCT calculation.

$$Y = C^{T}\,(Y \otimes E)\,C =$$

$$= \begin{bmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} \left(\begin{bmatrix} \\ X \\ \\ \end{bmatrix} \otimes \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix} \right) \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix} \quad (2)$$

The scale factors in (1) and (2) are constants, where **a** is $1/2$ and **b** is $\sqrt{2/5}$.

IV. DESIGNED ARCHITECTURES

In this work, the 2-D FDCT and the 2-D IDCT architectures were designed using the separability property of this transform: each 2-D transform (forward or inverse) is divided in two 1-D transforms that are joined through a transpose buffer. Thus, the output data from the first transform is reorganized by a transpose buffer and used by the second transform. This division is generically presented in Fig. 3 (a) for the 2-D FDCT and Fig. 3 (b) for the 2-D IDCT. This solution allows a significant reduction in the 2-D FDCT and in the 2-D IDCT calculation complexity.

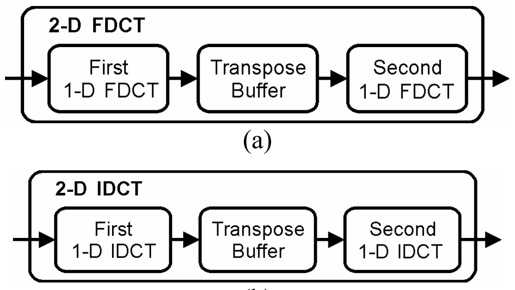


Figure 3. (a) 2-D FDCT blocks and **(b)** 2-D IDCT blocks

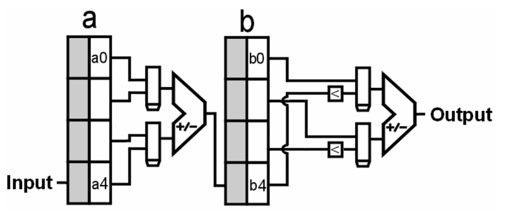
The 2-D transforms architectures consume one input value at each clock cycle. The global latency of the 2-D DCT and 2-D IDCT are equal, once these architectures are organized in a similar structure. This latency is of 32 clock cycles and, when the pipeline is full, a new output data is available at each clock cycle.

The input dynamic range of 2-D FDCT is from -256 to +255, i.e. 9 bits. The first 1-D FDCT inputs are the same 2-D FDCT inputs. The dynamic range of the first 1-D FDCT is increased in 3 bits and its outputs will have 12 bits. The bit width input and output of the transpose buffer are the same and it is determined for the first 1-D FDCT output that is 12 bits. The dynamic range of the second 1-D FDCT also is increased in 3 bits, and then its 12 input bits are transformed in 15 output bits. The second 1-D FDCT outputs are the 2-D FDCT outputs, then the 2-D FDCT dynamic range is increased in 6 bits, from 9 input bits to 15 output bits.

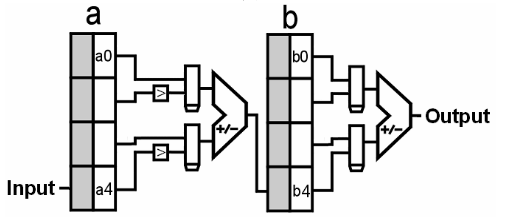
The input of the 2-D IDCT is 15 bits wide. The first and second 1-D IDCT dynamic ranges are increased in 2 bits and the transpose buffer does not increase its the dynamic range. The first 1-D IDCT inputs are of 15 bits and the outputs are 17 bits wide. The second 1-D IDCT inputs are 17 bits wide and the outputs are 19 bits wide. The second 1-D DCT outputs pass trough a shift right of 6 bits (Wang *et al.*, 2003) and this shifted value, with 13 bits, is sent to the 2- D IDCT outputs.

Four different 1-D transforms were designed in this work, two for the 2-D FDCT and two for the 2-D IDCT. The two 1-D FDCT architectures are similar and only the bit widths of their inputs and outputs are different. The second 1-D FDCT architecture could be used also to realize the first 1-D FDCT calculation, but the first 1-D FDCT was optimized to save logic elements. The two 1-D IDCT are also similar and just the bit width is different between these architectures.

The designed architectures for 1-D FDCT and 1-D IDCT calculations are composed of two pipeline stages, to increase the calculation performance. The latency of these 1-D FDCT and 1-D IDCT architectures is of eight clock cycles. Figure 4 (a) shows the 1-D FDCT architecture and Fig. 4 (b) shows the 1-D IDCT architecture.



(a)



(b)

Figure 4. Designed architecture for **(a)** 1-D FDCT and **(b)** 1-D IDCT

Each 1-D DCT employs two ping-pong buffers, four multiplexers, two shifters and two adders. The ping-pong buffers are necessary to maintain the four input data stable for each adder during four clock cycles, allowing the use of pipeline and the use of just one operator in each pipeline stage. These buffers allow maintaining the input rate of one data per clock cycle, storing the values in the gray registers, when the data are stable in the white registers to be used in each calculation step. The multiplexers are used to select the correct adder inputs at each clock cycle. The shifters are used to make the multiplications or divisions by two presented in formulas (1) and (2). A finite state machine is used to control the 1-D DCT block producing all the control signals.

Figure 5 shows the transpose buffer architecture that was designed to connect the two 1-D FDCTs into the 2-D FDCT architecture. The transpose buffer used into the 2-D IDCT architecture is similar to that presented in Fig. 5. The transpose buffers are composed of two 16-word RAMs and three multiplexers besides various control signals. The RAM memories operate in an intercalated way: while one of them is used for writing, the other is used for reading. Thus, the first 1-D FDCT or 1-D IDCT architecture writes the results line by line in one memory (RAM1 or RAM2) and the second 1-D FDCT or 1-D IDCT architecture reads the input values column by column from the other memory (RAM2 or RAM1).

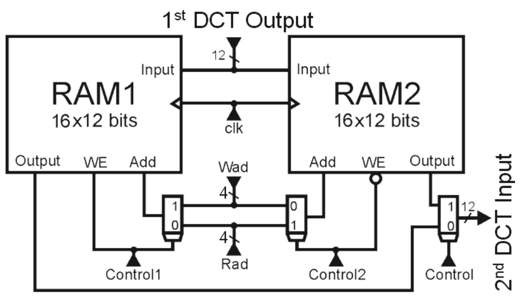


Figure 5. 2-D FDCT Transpose Buffer Architecture

The memories used to implement the transpose buffers are the internal RAM macroblocks available in the target FPGA. The memories used into the transpose buffer of the 2-D FDCT implementation have 12-bit words because this is the bit width of the first 1-D FDCT output. The memories used into the 2-D IDCT have 17-bit words and this is the difference between the two transpose buffer architectures. The bit width of the first 1-D IDCT output is 17 bits and hence, the transpose buffer memories must have the same bit width. The transpose buffer latency is of 16 clock cycles, which are necessary to complete the first write operation and to start the first read operation.

V. SYNTHESIS RESULTS

Table 1 presents the synthesis results of the 2-D FDCT architectures. The 2-D FDCT architecture is composed of two 1-D DCT calculations and a transpose buffer. These architectures, which were presented in previous sections, were described in VHDL and synthesized to Altera FPGAs (Altera, 2006). These three architectures were joined to construct the 2-D FDCT and were synthesized again. Table 2 shows the synthesis results for the 2-D IDCT and its internal blocks.

Table 1. 2-D FDCT synthesis results

Hardware Block	LEs	Period (ns)	Memory Bits	Throughput (Msamples/s)
1st 1-D FDCT	168	4.23	0	236.52
Transpose Buffer	33	3.15	384	317.86
2nd 1-D FDCT	216	4.86	0	205.97
2-D FDCT	**395**	**5.92**	**384**	**169.03**

Altera device: Stratix EP1S10F484C5

The syntheses of all architectures were mapped to EP1S10F484C5 Altera Stratix device (Altera, 2006). From Tab. 1 and Tab. 2 it is possible to notice that in both 2-D FDCT and 2-D IDCT architectures the transpose buffers are the blocks that use the lowest amount of logic elements and are the blocks that are able to operate at the highest throughput. On the other hand, just the transpose buffers use memory bits.

Table 2. 2-D IDCT synthesis results

Hardware Block	LEs	Period (ns)	Memory Bits	Throughput (Msamples/s)
1st 1-D IDCT	248	5.19	0	192.64
Transpose Buffer	38	3.13	544	319.18
2nd 1-D IDCT	278	5.19	0	192.57
2-D IDCT	**465**	**6.09**	**544**	**164.10**

Altera device: Stratix EP1S10F484C5

Comparing the results presented in Tab. 1 and Tab. 2 it is possible to notice that the 2-D FDCT architecture uses 395 logic elements of the target device and reaches a period of 5.92ns. On the other hand the 2-D IDCT uses 465 logic elements and reaches a minimum period of 6.09ns.

According to the results presented in Tab. 1 and Tab. 2, the 2-D FDCT integer transform is able to process 169.03 million of samples at each second, and the 2-D IDCT integer transform is able to process 164.1 million of samples per second. These processing rates are higher than that defined to HDTV (High Definition Digital Television), which must process 1920x1080 pixels per frame at 30 frames per second with a color relation of 4:2:2. Hence, the 2-D DCT designed architectures are able to be used in a complete H.264/AVC codec targeting HDTV.

VI. RELATED WORKS

There are a few papers in the literature about dedicated hardware architectures for H.264/AVC transforms. We do not find any similar solution for the 2-D FDCT and 2-D IDCT transforms that realizes the calculations in a serial fashion like the solution presented in this paper. This section will compare our design with the previously published solutions. The literature papers are based on standard-cells implementations and the comparisons are not easy. We will compare just the architecture design strategy and the performance. The used hardware resources will not be compared, once we just have results for FPGAs.

The solution presented in (Kordasiewicz and Shirani 2004) realizes just the 2-D FDCT calculation. The solutions proposed in (Agostini *et al.*, 2006; Cheng *et al.*, 2004; Chen *et al.*, 2005; Wang *et al.*, 2003) are multitransform architectures that are able to process the calculations related to the four 4x4 forward and inverse transforms (FDCT, IDCT, forward Hadamard and inverse Hadamard). The solution (Agostini *et al.*, 2006) process also the 2x2 forward Hadamard and is able to select the level of parallelism desired in its calculations. The designs presented in (Wang *et al.*, 2003) grouped individually each 4x4 transform (FDCT and forward Hadamard) with their specific quantization. The quantization calculation is a multiplication by constants and it has a low level of complexity.

The number of samples processed in each clock cycle varies form 4 to 16 in the related designs. Four solutions process 8 samples per clock cycle. This parallelism allows a very high processing rate from 273 (Lin *et al.*, 2005) to 3,499 (Agostini *et al.*, 2006) millions of samples per seconds. These processing rates surpass the high resolutions application requirements. A HDTV resolution video with frames of 1920x1080 pixels, at 30 frames per second, need a processing rate near to 95 millions of samples per second.

The very high performance obtained in the literature solutions, if used, implies in several difficulties to use these architectures in a complete T or T^{-1} block and in a H.264/AVC codec. In this case, the memory overhead will be an important challenge to be solved. Other important question is that the connection of these blocks with the other H.264/AVC blocks will use a lot of routing resources, once the I/O interface will use a lot of wires. Finally, it is really very difficult to design parallel H.264/AVC inter and intra prediction blocks with the necessary throughput required by the parallel transforms. Then, it is probable that these parallel transforms will be underutilized when integrated in a H.264/AVC codec. For these reasons we decided to design an architecture that process just one sample per clock cycle.

In the literature was not found a solution that process just one sample per clock cycle and, then, our throughput is lower than that obtained in the parallel solutions. But is important to notice that our solution reaches the performance requirements for HDTV and a higher throughput was not necessary in this case. The hardware consumption of our transforms is lower, once we process just one sample per clock cycle. A complete comparison in relation with the hardware resources consumption will be available when we finished a standard-cell version of our architectures.

VII. CONCLUSIONS AND FUTURE WORK

This paper presented an architectural design of two serial 2-D DCT transforms (forward and inverse) for the H.264/AVC standard. The architectures were described in VHDL and were synthesized for Altera FPGAs. The synthesis results were presented.

The forward 2-D DCT transform architecture uses 395 logic elements of the device and reaches a throughput of 169.03 millions of samples per second. On the other hand, the inverse 2-D DCT transform uses 465 logic elements and reaches a maximum throughput of 164.10 millions of samples per second. With these results it is possible to notice that the designed 2-D DCT architectures are able to be used in a H.264/AVC encoder and/or decoder hardware targeting HDTV, with frames of 1920x1080 pixels at 30 frames per second.

Our 2-D FDCT and 2-D IDCT designed architectures are serial and pipelined. These architectures were designed targeting FPGAs, but they can be synthesized in a standard-cell version. The DCT architectures are simple and goals to minimize the resources consumption trough a serial implementation. This reduced consumption of resources is very important, once that a complete H.264/AVC encoder or decoder is formed by various complex blocks that will consume a lot of hardware resources. Then, the implementation of all blocks must minimize its resources consumption to allow a complete coder or encoder mapping in FPGAs that is the main goal of the project that supports this work. But it is very important to emphasize that all this H.264/AVC blocks must reach the performance requirements that are defined to the complete

H.264/AVC encoder or decoder. The 2-D FDCT and 2-D IDCT architectures designed in this paper are able to operate at a processing rate higher than 160 millions of samples per second, considering the target FPGA. This processing rate is enough to respect the HDTV requirements and, for consequence this design is able to be used in all other video transmission standards with resolutions lower than HDTV.

As future work we plain to design a standard-cell version of our 2-D FDCT and 2-D IDCT architectures. With the synthesis results will be possible a comparison in terms of use of hardware resources between our solution and the literature solutions. Other work that are being developed is the design of architectures for the two Hadamard transforms. Other H.264/AVC blocks, like quantization and entropy coding are planned to be designed as next activities.

REFERENCES

Agostini, L.V., R.C. Porto, J.A. Güntzel, I.S. Silva and S. Bampi, "High Throughput Multitransform and Multiparallelism IP for H.264/AVC Video Compression Standard", *ISCAS 2006 – IEEE International Symposium on Circuits and Systems*, Kos, Greece, 5419 - 5422 (2006).

Airan, R., J. Berge and V. Olive, *Circuit Synthesis with VHDL*, Kluwer, Boston (1994).

Altera Corporation, "Home Site of the Altera Corporation", <www.altera.com> (2006).

Chen, K., J. Guo and J. Wang, "An Efficient Direct 2-D Transform Coding IP Design for MPEG-4 AVC/H.264", *ISCAS 2005 – IEEE International Symposium on Circuits and Systems*, Kobe, Japan, 4517-4520 (2005).

Cheng, Z., C. Chen, B. Liu and L. Yang, "High Throughput 2-D Transform Architectures for H.264 Advanced Video Coders", *IEEE Asia-Pacific Conference on Circuits and Systems*, Fukuoka, Japan, 1141-1144 (2004).

Joint Video Team of ITU-T and ISO/IEC JTC 1, "*Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 or ISO/IEC 14496-10 AVC)*" (2003).

Kamaci, N. and Y. Altunbasak, "Performance Comparison of the Emerging H.264 Video Coding Standard with the Existing Standards," *ICME 2003 - IEEE International Conf. on Multimedia and Expo*, Baltimore, USA, 345-348 (2003).

Kordasiewicz, R. and S. Shirani, "Hardware Implementation of the Optimized Transform and Quantization Blocks of H.264", *Canadian Conference on Electrical and Computer Engineering*, Ontario, Canada, 943-946, (2004).

Lin, H., Y. Chao, C. Chen, B. Liu and L. Yang, "Combined 2-D Transform and Quantization Architectures for H.264 Video Coders", *ISCAS 2005 – IEEE International Symposium on Circuits and Systems*, Kobe, Japan, 1802-1805 (2005).

Malvar, H., A. Hallapuro, M. Karczewicz and L. Kerofsky, "Low-Complexity Transform and Quantization in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, **13**, 598-603, (2003).

Puri, A., X. Chen and A. Luthra, "Video coding using the H.264/MPEG-4 AVC compression standard," *Signal Processing: Image Communication*, **19**, 793-849 (2004).

Richardson, I., *H.264 and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia*, John Wiley and Sons, Chichester (2003).

Sullivan, G. and T. Wiegand, "Video Compression – From Concepts to the H.264/AVC Standard," *Proceedings of the IEEE*, **93**, 18-31 (2005).

Sullivan, G., P. Topiwala and A. Luthra, "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions", *XXVII SPIE Conference on Applications of Digital Image Processing*, San Diego, USA, 454-474 (2004).

Wang, T., Y. Huang, H. Fang and L. Chen, "Parallel 4x4 2D Transform and Inverse Transform Architecture for MPEG-4 AVC/H.264", *ISCAS 2003 - IEEE International Symposium on Circuits and Systems*, Bangkok, Thailand, 800-803 (2003).

Wiegand, T., G. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, **13**, 560-576 (2003).

Wien, M., "Variable Block-Size Transforms for H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, **13**, 604-613 (2003).

Received: April 14, 2006.
Accepted: September 8, 2006.
Recommended by Special Issue Editors Hilda Larrondo, Gustavo Sutter.