Low-Cost and High-Throughput Hardware Design for the HEVC 16x16 2-D DCT Transform

Ruhan Conceição, José Cláudio de Souza Jr., Ricardo Jeske, Bruno Zatt, Marcelo Porto, Luciano Agostini

GACI – Group of Architectures and Integrated Circuits
CDTec – Technology Development Center
UFPel – Federal University of Pelotas, Pelotas, Brazil
{radconceicao, jcdsouza, rgjeske, zatt, porto, agostini}@inf.ufpel.edu.br

ABSTRACT

This article presents the hardware design of the 16x16 2-D DCT used in the new video coding standard, the HEVC – High Efficiency Video Coding. The transforms stage is one of the innovations proposed by HEVC, since a variable size transforms stage is available (from 4x4 to 32x32), allowing the use of transforms with larger dimensions than used in previous standards. The presented design explores the 2-D DCT separability property, using two instances of the one-dimension DCT. The architecture focuses on low hardware cost and high throughput, thus the HEVC 16-points DCT algorithm was simplified targeting a more efficient hardware implementation. Operations and hardware minimization strategies were used in order to achieve such simplifications: operation reordering, factoring, multiplications to shift-adds conversion, and sharing of common sub-expressions. The 1-D DCT architectures were designed in a fully combinational way in order to reduce control overhead. A transposition buffer is used to connect the two 1-D DCT architectures. The synthesis was directed to Stratix III FPGA and TSMC 65nm standard cells technologies. The complete 2-D DCT architecture is able to achieve real-time processing for high and ultra-high definition videos, such as Full HD, QFHD and UHD 8K. When compared with related works, the architectures designed in this work reached the highest throughput and the lowest hardware resources consumption.

Index Terms: video coding, HEVC, 16x16 DCT, FPGA design, ASIC design

I. INTRODUCTION

Nowadays, the resolution and the quality of digital videos have been increasing in a fast and steady manner. Additionally, such videos are becoming supported by an increasing number of electronic devices (smartphones, set-top-boxes, blu-ray players etc.). As consequence, the study and the improvement of video encoders/decoders are extremely relevant topics, since many consumer electronic devices that process digital videos, with their diverse features, must be capable of processing high-resolution videos in real time. For this reason, topics such as compression rate, video quality, computational effort, hardware resources, and energy consumption must be addressed by the research community in order to continuously find improved solutions for the consumer.

Video coding is imperative in applications that handle digital videos, since an uncompressed video requires a gigantic amount of bits to be represented and enormous resources for storage and transmission, making such applications unfeasible in the current technology [1].

Until early 2013, the H.264/AVC [2] was the most advanced video coding standard, presenting significant gains in compressions when compared to its predecessor, the MPEG-2 [3]. In face of new video-related challenges, on January 2010, the JCT-VC (Joint Collaborative Team – Video Coding) was created, composed of experts from ITU-T and ISO/IEC, to start the development of a new video coding standard, the so called HEVC – High Efficiency Video Coding [4]. The goal of the JCT-VC was to increase video compression by 50%, while maintaining the same image quality and the same computational effort in relation to the H.264/AVC. Although computational effort has been increased, the HEVC meets the coding performance goals [5] and, since its release in April 2013, is considered the state-of-art in video coding.

HEVC brings a set of new video coding tools that include novel picture partitioning strategy and shapes, prediction modes, in-loop filters, transform sizes, etc [6]. During the HEVC encoding process, each frame is divided into square-shaped units, named Coding Tree Units (CTU), containing up to 64x64 luminance samples (16x16, 32x32, or 64x64) and the associated chrominance samples [7][8]. The chrominance blocks dimensions depend on the color subsampling used. The concept of a CTU is broadly analogous to the macroblock concept in previous standards, such as the H.264/AVC.

Each CTU is composed of one or more Coding Units (CU). In order to decide the size of each CU, these units are recursively divided into four blocks (smaller CUs) of the same size. This process starts at the CTU and goes all the way down to a minimum of 8x8-sized blocks. This recursive process forms a quadtree composed of CU blocks, as depicted in Fig. 1. Inside each CU, prediction, transforms, quantization, and residual coding take place. Fig. 1 illustrates one possible partitioning of a CTU forming the quadtree structure.

To optimize the prediction process, the HEVC defines that each CU can be composed of one or more Prediction Units (PU) during the prediction process. The PUs can assume not only symmetric forms (square or rectangular), but also asymmetric forms in order to better adjust the prediction to the image characteristics [10]. Each PU has its own intra prediction mode (if intra prediction is used) or motion vectors (if inter prediction is used).

At transforms perspective, each CU may be processed as a single Transform Unit (TU) or further divided into four TUs for the transforms and quantization processes. If the CU is divided, each TU can be divided again in four smaller TUs. The TUs are square-shaped partitions varying from 4x4 up to 32x32 samples. As each CU may contain one or more TUs, the TU decision introduces a quadtree structure similar to the PU quadtree.

The transforms stage, focus of this work, holds an important role in the encoding process by converting the image information from the spatial domain to the frequency domain. The purpose of the transforms is to concentrate the energy of an image block in a few

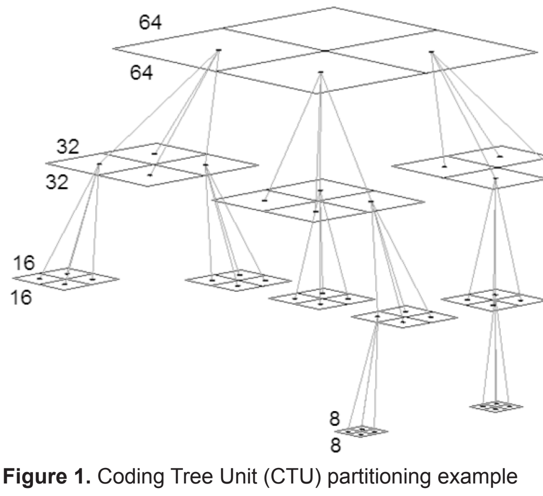

Figure 1. Coding Tree Unit (CTU) partitioning example

frequency coefficients. Thanks to this concentration the quantization step is able to eliminate/attenuate high frequency coefficients, less relevant to the human visual system, allowing the entropy encoder to reach very high compression rates. HEVC defines an integer approximation of the Discrete Cosine Transform (DCT) as the main transform used in the standard. Hadamard is also used in some specific cases.

The transforms, especially the DCT, are processing intensive tasks in the HEVC encoder demanding about 4-9% [12] of the total encoding time. Furthermore, once transforms belong to the reconstruction loop, there is a need for low-latency transform implementations in order to avoid encoder stalls. Thus, dedicated DCT hardware architectures are mandatory for real-time HEVC encoding systems.

Since HEVC was recently released, there are few publications available in the literature related to hardware implementations of the coding tools described for this video coding standard.

The work described in [16] presents a technique for IDCT calculation that analyses the DC coefficient and the three lowest frequency coefficients of the input matrix. The work proposed in [17] presents a multi-standard architecture for DCT computation of H.264/AVC and HEVC, processing either forward or inverse DCT. The paper presented in [18] follows the same idea than [17], presenting a forward/inverse DCT and Hadamard. These works focus on multi-sized transforms. This approach is hardware efficient and adapts very well to the decoder size where one transform type/size is performed at a time. However, during the video coding process many transform sizes have to be evaluated in order to decide the ideal transform to each block. Thus, the parallelism exploration by employing size-specific transform accelerators is necessary, especially when high resolution videos are considered.

The work proposed by Ahmed [14] shows an architecture solution for the 16-points DCT. It was implemented in 90nm standard-cells technology. Although this work presents a single-sized solution, the hardware design presented in [14] is not capable to process UHD video sequences at 60 frames per second. This processing rate is very important considering the latest technologies.

Edirisuriya [15] proposes an architecture for a transform engine capable of computing the 16×16 2-D DCT/DST multitransform engine without the use of multipliers. The proposed architecture was implemented using a FPGA and mapped to a 45nm technology. In spite of the multitransform engine proposed by Edirisuriya [15], the proposed architecture does not perfectly fit to the numbers of input bits of the HEVC DCT. Thus, the hardware design is not optimized at fine-grained level (bit level).

In this work is presented an efficient hardware design of the HEVC 16x16 2-D DCT. The 2-D DCT hardware was designed using two instances of the 1-D DCT, exploring the 2-D DCT separability property. The two 1-D DCT instances were designed in a purely combinational way targeting on high throughput, low latency and low hardware resources demand. The two 1-D DCTs are connected through a transposition buffer, implemented using registers, completing the 2-D DCT. Hardware optimization techniques were applied to the 1-D DCT design resulting in a meaningful reduction of the hardware resources required to implement the transform whereas introducing an expressive throughput increase when compared to the implementation obtained from the non-optimized HEVC 1-D DCT.

This work extends our previous solution proposed in [21] with novel contributions on the algorithm optimizations, where a best arrangement of the sub expression sharing was found. At the hardware design perspective, this work proposes a 2-D DCT hardware - only a 1-D DCT architecture was presented in [21] - along with its ASIC implementation and the analyses and characterization of the power consumption. Finally, unlike our previous hardware design published in [21], this work presents an architecture capable to process UHD 8K - Ultra High Definition (7680x4320 pixels) - video sequences in real time at 120 frames per second.

This article is organized as follows: Section 2 introduces the 2-D DCT transform; Section 3 presents the algorithm optimization for the HEVC 16-points 1-D DCT; Section 4 shows the architectural design of the 1-D 16-points DCT, and Section 5 shows the full 2-D 16x16 DCT architectural design. Synthesis results for both designs are presented and discussed on Section 6 and comparison with related works is shown in Section 7. Section 8, finally, discusses the conclusions of this work.

II. 2-D DCT TRANSFORM

A generic 2-D transform can be defined as presented in (1), where N^2 is the number of DCT inputs, *p(x,y)* is the input at the position *(x,y)* of the input matrix, and $G_{i,j}$ is the output coefficient. Since this work is focused in a 16x16 DCT, then $N=16$.

$$G_{i,j} = \frac{1}{\sqrt{2N}} \in_i \in_j \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} p(x,y).\cos\left[\frac{(2y+1)j\pi}{2N}\right]\cos\left[\frac{(2x+1)i\pi}{2N}\right] \quad (1)$$

$$\in_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k=0 \\ 1 & \text{otherwise} \end{cases}$$

A direct hardware implementation of equation (1) for a 16x16 transform, without any simplification, will use 65,536 multiplications and 65,280 additions to generate all the 256 transformed coefficients.

A traditional approach to reduce the 2-D DCT complexity is the use of the separability property. The separability considers that two 1-D DCT can be used to calculate the 2-D DCT.

A generic 1-D transform can be defined as presented in (3), where *N* is the number of 1-D DCT inputs (16), *p(t)* is the input at the position *t* of the input matrix, and G_f is the output coefficient.

$$G_f = \frac{1}{2} \in_f \sum_{t=0}^{N-1} p(t).\cos\left[\frac{(2t+1)f\pi}{2N}\right] \quad (2)$$

$$\in_k = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } k=0 \\ 1 & \text{otherwise} \end{cases}$$

The 2-D DCT is composed of two subsequent steps of 1-D DCT transforms, connected by a transposition step. Initially, the first 1-D DCT reads data from the input matrix line by line, then a series of calculations is performed and the results are stored in an intermediate matrix, column by column. After the input matrix is completely processed and the intermediate matrix is filled out, the second 1-D DCT is started, repeating the same process with the intermediate matrix as input, reading the transposed results from the first 1-D DCT. The final results are then stored in the output matrix. This process is illustrated in Figure 2. Since this work is focused in the HEVC 16x16 DCT, then two 1-D DCTs with 16 points are necessary to implement the 2-D DCT.

The use of the separability property itself reduces the number of operations to 8,192 multiplications and 7,680 additions. This is an expressive reduction; however, other simplifications must be done to further optimize the hardware design, as will be presented in this paper.

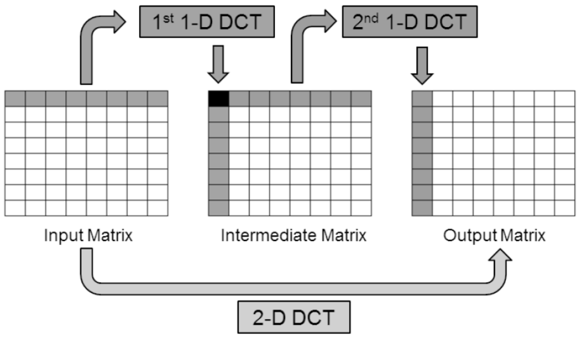

Figure 2. 2-D DCT implemented as two instances of a 1-D DCT

III. HEVC DCT ALGORITHM OPTIMIZATION

The HEVC DCT is an integer approximation of the DCT mathematical definition presented on Section 2. This integer approximation is useful to reduce the transform complexity and to avoid mismatches between coders and decoders [9].

In this work, the HEVC 16-points DCT was designed using two optimization techniques: (A) multipliers elimination and (B) common sub expression sharing. Such optimizations, as described in the current section, were based on an extensive analysis of the DCT equations.

A. Multipliers Elimination

Table I presents the first calculations of the simplified 1-D DCT algorithm. In this stage, only sums and subtractions are performed over the input samples. In Table I, W_n represents input samples whereas a_n denotes the outputs of this stage.

In Table II are presented another set of equations related to the 1-D DCT transform. This stage takes place after the stage described by Table I. Note that the outputs of Table I (a_n) work as input for Table II. X_n represents the 1-D transform outputs.

The cc_n and bb_n values are calculated by additional operations considering a_n values as input. Such operations are omitted from Table II for better visualization. In this case, cc_n and bb_n values are produced only through sums and subtractions, as observed in the equation (3), where the operation correspondent to bb_1 is presented.

Table I. First Operations in the Algorithm

Output	Input
a_0	$W_0 + W_{15}$
a_1	$W_0 - W_{15}$
a_2	$W_1 + W_{14}$
a_3	$W_1 - W_{14}$
…	…
a_{15}	$W_7 - W_8$

Table II. Equation Arrangements

Group	Output	Equations
4	X_0	$(64*cc_0 + 64*cc_2 + 4) >> 3$
1	X_1	$(90*a_1 + 87*a_3 + 80*a_5 + 70*a_7 + 57*a_9 + 43*a_{11} + + 25*a_{13} + 9*a_{15} + 4) >> 3$
2	X_2	$(89*bb_1 + 75*bb_3 + 50*bb_5 + 18*bb_7 + 4) >> 3$
1	X_3	$(87*a_1 + 57*a_3 + 9*a_5 - 43*a_7 - 80*a_9 - 90*a_{11} + - 70*a_{13} - 25*a_{15} + 4) >> 3$
3	X_4	$(83*cc_1 + 36*cc_3 + 4) >> 3$
…	…	…
1	X_{15}	$(9*a_1 - 25*a_3 + 43*a_5 - 57*a_7 + 70*a_9 - 80*a_{11} + + 87*a_{13} - 90*a_{15} + 4) >> 3$

$$bb_1 = a_1 - a_{13} \qquad (3)$$

The equations for X_n were grouped in four subgroups, grouping up similar equations, as observed in the leftmost column of Table II. After being grouped up, all equations are submitted to an algebraic manipulation process, which is illustrated in equations (4), (5), (6), (7), and (8) for the output X_1. This equation was taken as example because it belongs to group 1, which corresponds to the largest equations, making it possible to demonstrate all the simplification steps applied.

Additionally, the equations are factorized in order to decrease the bit width of the operators. This improvement does not alter results and enables a lower cost hardware design.

$$X_1 =(90*a_1 + 87*a_3 + 80*a_5 + 70*a_7 +57*a_9 + + 43*a_{11} + 25*a_{13} + 9*a_{15} + 4) >> 3 \qquad (4)$$

$$x_{1a} = 8 * (a_1 + a_3 + a_5 + a_7 +a_9 + a_{11} +a_{13} + a_{15}) \qquad (5)$$

$$x_{1b} = 82*a_1 + 79*a_3 + 72*a_5 + 62*a_7 + + 49*a_9 + 35*a_{11} + 17*a_{13} + a_{15} \qquad (6)$$

$$x_{1c} = x_{1a} + x_{1b} \qquad (7)$$

$$X_1 = (x_{1c} + 4)>> 3 \qquad (8)$$

In this example, equation (4) shows the original equation, which was divided into two parts. The multiplications by constants in equation (4) are mapped to a linear combination of two terms: (i) x_{1a} with all inputs multiplied by the constant eight; (ii) x_{1b} with the inputs multiplied by their original multipliers minus eight. This is the first stage of the factorization process and allows the operators to use a reduced bit width as the multiplying constants are lower. Equations (5) and (6) are summed up in (7) and the result is then used on equation (8).

The multiplying constant 8 presented in equation (4) was used in this example as it is the highest radix-2 value, which can be used to subtract all the constants present in equation (3) simultaneously, without resulting in negative constants in (6). Since it is a radix-2 value, the multiplication by 8 can be performed by simply shifting the input three bits to the left.

Avoiding negative constants allow the architecture to use simpler hardware operators, since it would not be necessary to implement subtractors, which are composed only by full-adders. Although only the most significant bit would be simplified to a half-adder, the resulting hardware reduction is applied to every adder present in the architecture.

All equations were individually and jointly analyzed in order to find the best set of sum pairs, resulting in the highest number of operations reuse. Consequently, hardware resources consumption can be saved.

Table III demonstrates how the multiplications presented in equation (6) are produced with the use

Table III. Translation of the Multiplying Constants from Equation (6) to Sums and Shifts

a_n	Constants	Shifts Used (+: Sums / - : Subtractions)							Operations
		64	*32*	*16*	*8*	*4*	*2*	*1*	
		<< 6	*<< 5*	*<< 4*	*<< 3*	*<< 2*	*<< 1*	*±1*	
a_{15}	1							+	1
a_{13}	17			+				+	<< 4 + 1
a_{11}	35		+				+	+	<< 5 + << 1 + 1
a_9	49		+	+				+	<< 5 + << 4 + 1
a_7	62	+					-		<< 6 - << 1
a_5	72	+			+				<< 6 + << 3
a_3	79	+		+				-	<< 6 + << 4 – 1
a_1	82	+		+			+		<< 6 + << 4 + << 1

of shit-adds. Shifting operations can be easily implemented with a concatenation of zeroes to the right side of the variable. These procedures are carried out for all equations from the four groups, always keeping the smallest possible number of operations per constant.

Each line in Table III corresponds to a multiplication presented in equation (6), and the last column corresponds to the shifting, sum or subtraction operations required for each a_n variable. After producing all the sequences of adders/subtractors needed to perform the multiplications with sums and shifts, the most suitable configuration for the sequence of operations is selected, leaving shifting operations (zero concatenations) to the end of the equation whenever it was possible. By doing so, it was possible to save bits from the adders involved.

For example, considering the multiplications presented in the last line of the Table III, if it is performed firstly a_1 shifted six times to left plus a_1 shifted four times left, it would be necessary a 15-bit adder to perform this operation. Moreover, a 16-bit adder will be necessary to sum the result generated with a_1 shifted twice left. However, if it is performed firstly the sum of a_1 shifted four times left with a_1 shifted twice left, it would be necessary a 13-bit adder to do this sum, and another 14-bit adder to sum the result generated with a_1 shifted six times left.

B. Common Sub-expressions Sharing

The second optimization applied is the sub-expression sharing among the equations. Table IV displays only the sums over the pairs of a_n values that are used in equation (4) and that are shared among other equations. In total, 34 groups of sums over a_n were produced. Table IV presents examples of redundant operations. This operation reuse was used in every calculation, achieving a significant optimization among the other simplifications. The number of occurrences of this operation in all DCT processing is also shown in Table IV.

Table IV. Sub-expression Sharing Example

Adders	Operations	Occurrences
A	$a_1 + a_3$	7
B	$a_5 + a_7$	4
C	$a_9 + a_{11}$	3
D	$a_{13} + a_{15}$	3
E	$a_9 + a_{13}$	2
F	$a_7 – a_{11}$	2
…	…	

By factoring equation (6), using the operations presented in Table IV, equation (9) was created. This is the optimized equation for x_{1b}. Note, only sums and shifting operations are used in this equation.

$$x_{1b} = ((A+B+C+D)<<3) + ((A+B)<<6) + \\ + (C<<5) + ((A+E)<<4) + (a_5<<3) + \\ + ((a_1\text{-}F)<<1) + (C+D\text{-}a_3) \quad (9)$$

Equation (10) presents the same calculation illustrated in equation (9), but now related to the X3 output. In (10), the sub-expressions a_1, b_0, b_4 and (b_2+b_3), which are shared with equation (9), are highlighted. This reuse process is improved when all equations are analyzed together.

$$x_{3b} = ((A+J\text{-}(\ C+D\ ))<<3) + ((G\text{-}E)<<6) + \\ + (H<<5) + ((G+I)<<4) – (a_9<<3) + \\ + ((a_{13}\text{-}L)<<1) + (J+I\text{-}\ a_1) \quad (10)$$

Finally, the last part of the equations, which is common to all of them, is also simplified. This rounding stage presented in equation (8) can be directly mapped to (11). Such modification produces the same output as equation (8) whereas reducing the hardware needs since two bits can be discarded before the sum operation.

$$X_1 = ((x_{1c}\ >> 2)+1)>>1 \quad (11)$$

Considering all the improvements performed in the 1-D DCT algorithm, 223 sums/subtractions operations are required to produce the final outputs of the 1-D DCT. It is important to highlight that no multiplications are needed. Our baseline for comparison, the DCT implemented in the HEVC Reference Software - named HEVC Model (HM) -, needs 116 sums/subtractions and 88 multiplications to produce the same result. Considering the high cost required to perform multiplications, it is clear that the gains obtained through the simplifications performed in the algorithm are expressive, especially when considering hardware implementation. Other important optimization is related to the number of bits per operator (adder or subtractor) which has been reduced when compared to the original solution.

All the algorithmic modifications were validated by simulation, proving that the simplified equations were equivalent to the original ones. Henceforth, it was possible to start a hardware design for these equations.

IV. 16-POINT 1-D DCT ARCHITECTURAL DESIGN

The 16-points 1-D DCT architectural design, presented in Fig. 3, was based on the optimized algorithm presented in the previous section. The architecture was described in VHDL in a purely combinational design using a structural description. The architecture processes 16 samples per clock cycle, with nine bits per input sample (which is the output bit width from the previous video coder stage). The outputs were defined with seventeen bits, in order to keep the precision of the results and to guarantee that no overflow will occur. The adders and subtractors were based on ripple carry adders.

The designed 1-D DCT architecture used 223 sum /subtraction operators, varying from nine to nineteen input bits. In the worst case, eight adders or subtractors are serially connected.

As previously discussed, each multiplication was implemented through zero concatenations to the right, representing shifts to the left, and additions. These operations were implemented according to the data presented in Table III.

The multiplication displayed in (12) is one of the multiplications performed in equation (6). Observing Table III, it is possible to notice that this multiplication was represented with shift-adds, accordingly (13). The 6 and 4 shifts to the left were replaced by 6 and 4 zeros concatenations to the right of operand a_3, producing equation (14). In equation (14), the three terms of the sum have different bit widths and, therefore, zeros or ones must be concatenated to the left side of these terms, since the operations must be performed with operands of the same bit width, which is shown in (15). These zeros or ones (signal extension) are concatenated to the left-side of the operand a_3 according to its signal as shown in equation (17).

To save bits in the adders, the subtraction is performed first, avoiding the use of two extra bits. These

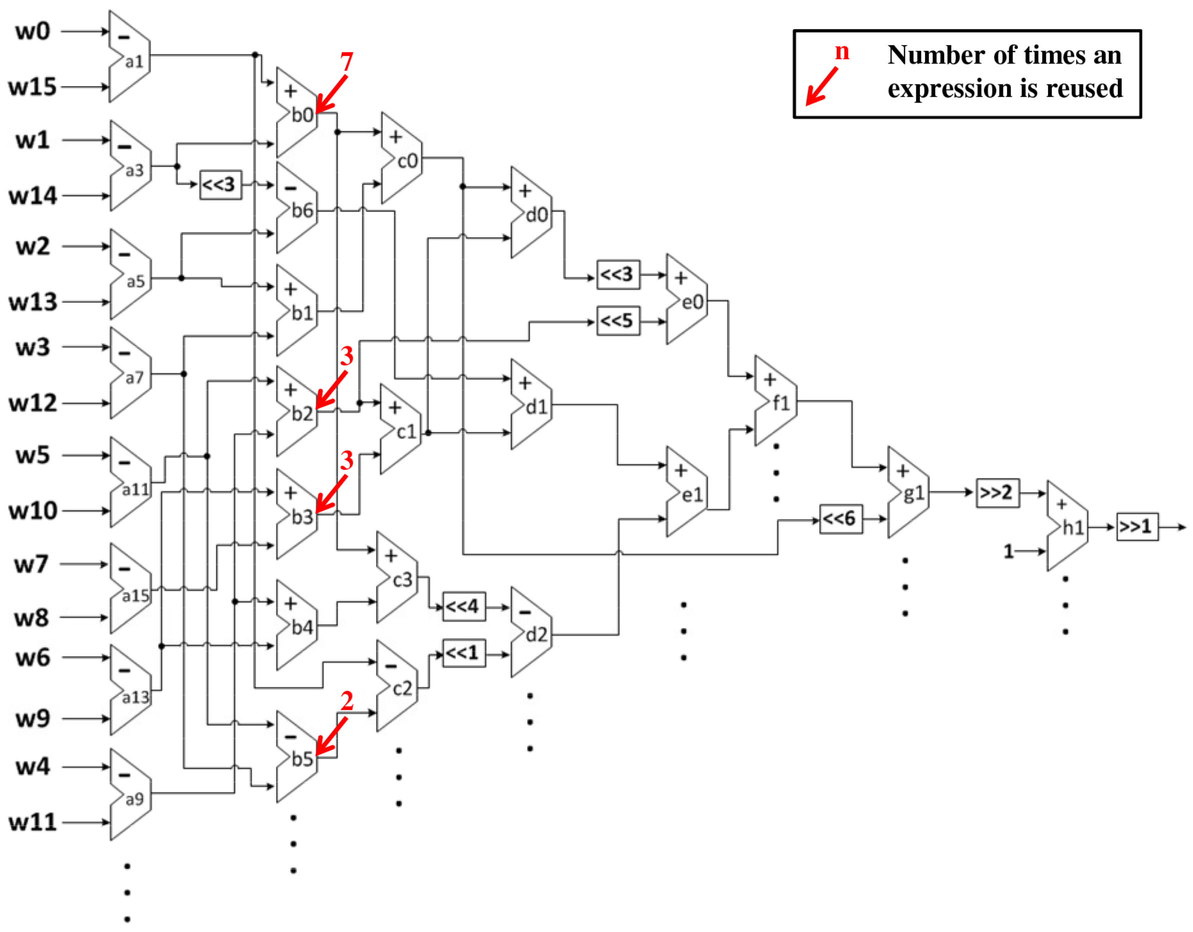

Figure 3. Partial Diagram of the Designed HEVC 1-D DCT Architecture

two extra bits are concatenated to the subtraction output, which is accordingly presented in (16), where X denotes the value of the most significant bit of a_3.

$$t = 79 * a_3 \tag{12}$$

$$t = (a_3 << 6) + (a_3 << 4) - a_3 \tag{13}$$

$$t = (a_3 | \text{‘000000’}) + ((a_3 | \text{‘0000’}) - a_3) \tag{14}$$

$$t = (a_3 | \text{‘000000’}) + ((\text{‘XX’} | a_3 | \text{‘0000’}) - (\text{‘XXXXXX’} | a_3)) \tag{15}$$

$$t = (a_3 | \text{‘000000’}) + (\text{‘XX’} | ((a_3 | \text{‘0000’}) - (\text{‘XXXX’} | a_3))) \tag{16}$$

$$X = \begin{cases} 0 & \text{if a3} \geq 0 \\ 1 & \text{otherwise} \end{cases} \tag{17}$$

The blocks diagram of the designed architecture is shown in Fig. 3. In this diagram, only the operations related to the X_1 output are listed. Other 15 outputs are generated with an equivalent hardware, but many of the sub expressions are shared among these calculations. Some shared signals are highlighted in the figure (using red arrows), showing the number of occurrences of this signal in other outputs. From Fig. 3, it is possible to verify the hardware designed using only sequences of simple sums and shifts. The shifts, in turn, were implemented with concatenations of zeros, as previously explained.

In order to allow an analysis of the gains in hardware resources consumption and throughput introduced by the optimizations in the algorithm, a second architecture was described in VHDL. This second architecture was a direct VHDL transcription of the HEVC 1-D DCT algorithm, extracted from the HM software. Consequently, this architecture is also purely combinational and it is able to process 16 samples per cycle. The synthesis results of both architectures, as well as the comparison between these results, are discussed in Section 6 of this paper. Comparison against related works is presented in Section 7.

V. 16X16 2-D DCT ARCHITECTURAL DESIGN

As previously described, the 2-D DCT separability property allows a complexity reduction in the DCT calculations applying the 1-D DCT two times, being the second one applied over the transposed results of the first one. Therefore, the hardware architecture that describes the 16x16 DCT implements two 16-point 1-D DCT modules. In addition to these two modules, two local buffers that store the intermediate matrix produced after the first 1-D DCT were designed to be used as the transposition buffer. Figure 4 presents the proposed 16x16 2-D DCT architecture block diagram. Each transposition buffer was implemented as a 16x16 register file with 16 bit-wide registers. The register files used allow a more efficient data access, avoiding address generation overhead, which is necessary when memory is used. The results of the first 1-D DCT are stored line by line. When one line is stored, this line

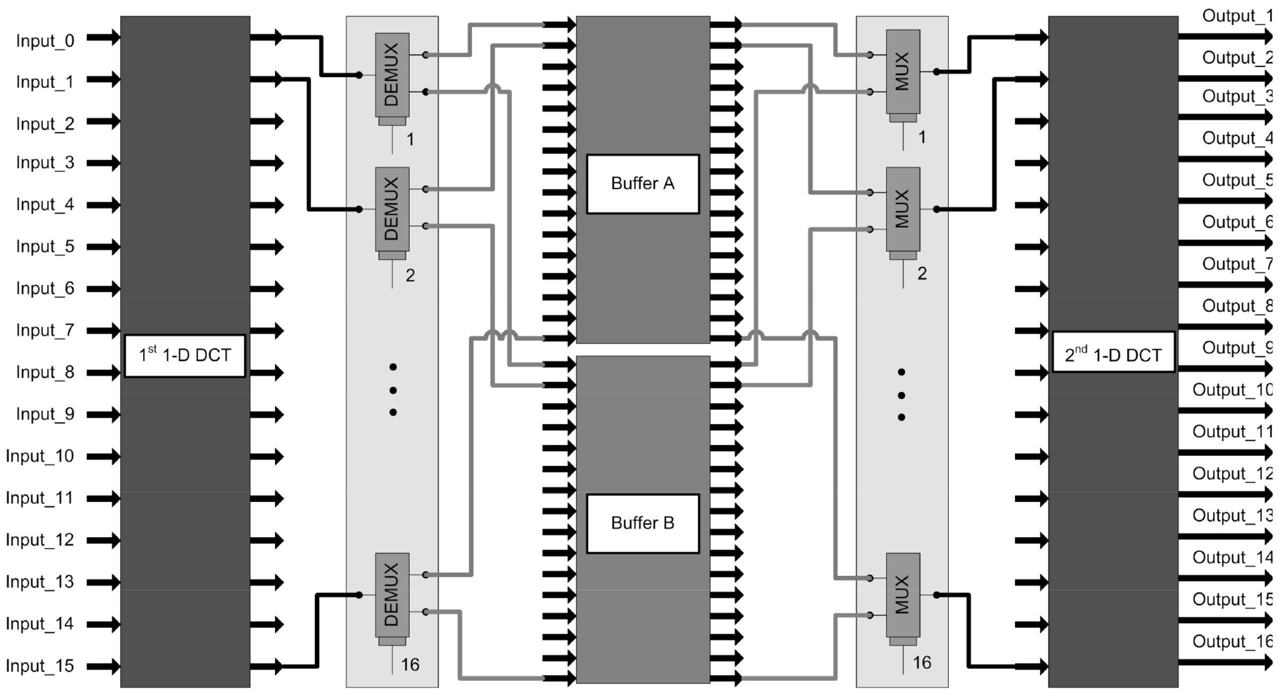

Figure 4. 16x16 DCT Architecture

is shifted down and the new line is stored as the first line of this buffer. When the first buffer is full, the results of the first 1-D DCT are stored in the second buffer. Simultaneously, the data from the first buffer are read column by column to feed the second 1-D DCT. The read operation is similar to the write operation, however, in this case, the data are accessed column by column and when one column is read, the remaining columns are right-shifted.

When the second buffer is empty, then it starts to be used again to write the results of the first 1-D DCT. This way, the two buffers operate in an interchangeable way.

The 2-D DCT hardware architecture also integrates a column with 16 multiplexers, and another column with 16 demultiplexers, as illustrated by the light-grey boxes in Fig. 4.

The control logic switches both multiplexers and demultiplexers simultaneously at each 16 clock cycles. The control signals to such components are alternated in a way that, when multiplexers receive the control signal "0", demultiplexers receive signal "1". In doing so, while data is being stored in "Buffer A", the demuxes are receiving inputs from "Buffer B", and after 16 clock cycles the selection signal is inverted, so that the new input data is now stored in "Buffer B", while data that was previously stored in "Buffer A" is provided to the second 1-D DCT module.

Lastly, two adaptations were made in the second 1-D DCT architecture in order to implement a 2-D DCT: (1) the final sum and rounding step was modified, because both instances use different values for these operations; and (2) the bit width of every adder was increased by seven, due to the carry-out propagation resulted from the sums performed in the first 1-D DCT.

The 2-D DCT architecture presents a latency of 18 clock cycles to generate the first column of results. Sixteen (16) cycles are used to process the first 1-D DCT over the 16x16 input (16 samples processed per clock cycle). Each line of the first 1-D DCT results is stored in the transposition buffer. When the last line of the first block is processed, this block is immediately sent to the second 1-D DCT, in the 16th cycle. Then, in the next cycle, the second 1-D DCT will deliver the first column of results. Considering that, this architecture needs 18 clock cycles to process a 16x16 samples block.

VI. SYNTHESIS RESULTS

This section presents the synthesis results for the designed architectures. The results are presented in two subsections: one for the 1-D DCT and other for the 2-D DCT.

A. 16-Point 1-D DCT results

As previously cited, two architectural versions of the 16 points 1-D DCT algorithm were designed in hardware: one based on the original HM algorithm and other based on the optimized algorithm proposed in this work. The architectures were described in VHDL and synthesized using the Altera Quartus II software. The FPGA family selected for the synthesis was the Stratix III for both architectures, to allow the desired comparison. The synthesis results obtained for both architectures are displayed in Table V, where the number of required ALUTs and maximum frequency are provided. Additionally, gain percentages regarding ALUTs consumption and frequency are also presented.

The results presented in Table V show expressive improvements when using the optimized algorithm in the hardware design. The gain was especially significant when considering the operation frequency, surpassing 445% increase, with a hardware consumption reduction of 72%. These results show the relevance of the proposed optimizations in the 1-D DCT algorithm when targeting the hardware design.

B. 2-D DCT results

The 16x16 2-D DCT architecture was also described in VHDL. The synthesis targeted two technologies: Altera Stratix III FPGA and TSMC 65nm standard-cells technology. The FPGA synthesis was done using the Quartus II tool and the standard-cells version was synthesized using the Synopsys DC Compiler tool.

The synthesis results are presented in Table VI. The reached results targeting FPGA show that the 2-D DCT implementation presented a decrease in its maximum operation frequency, when compared to the standalone 1-D DCT architecture. This was already

Table V. 1-D DCT Synthesis Results

Architecture	FPGA	#ALUTs	ALUTs Gain	Freq. (MHz)	Freq. Gain
Original	Stratix III	18,484	-	19.66	-
Optimized	Stratix III	5,168	72.0%	87.60	445.6%

Stratix III device: EP3SL50F780C2

Table VI. 2-D DCT Synthesis Results

	Stratix III	TSMC 65nm
Hardware Consumption	16,002 ALUTs	14,954 Logic Gates
Frequency (MHz)	27.05	742.02
Power Consumption @ Max Frequency (mW)	-	135.4
Power Consumption for 30 QFHD at 30 fps (mW)	-	4.4

Stratix III device: EP3SL50F780C2

expected, since the use of more hardware and wider adders were expected to impact the performance of the complete architecture.

The synthesis for the TSMC 65nm standard-cells technology presented a very high maximum operation frequency, which was 27 times higher than that reached in the Stratix III FPGA synthesis.

Table VI also presents the power consumption of the standard-cells version, considering its highest operation frequency and targeting QFHD video sequences at 30 fps. The power consumption was 135.4mW in the first scenario and 4.4 mW in the second one. This is an indication that the proposed solution dissipates reduced power under real-time scenarios.

Table VII presents the 2-D DCT architecture throughput estimative, considering its maximum operation frequency for both cases: FPGA and standard-cells synthesis. In this table it is presented the throughput (in millions of samples per second) and the reached frame rate considering Full HD (1920x1080 pixels), QFHD – Quad Full HD (3840x2160 pixels) and UHD 8K – Ultra High Definition (7680x4320 pixels) resolutions. These resolutions were selected, because the HEVC standard was developed focusing in high resolutions. The results presented in Table VII considered that videos used the traditional 4:2:0 color sub-sampling [13].

As Table VII shows, the designed architecture is able to reach a very high throughput, even when targeted to a FPGA. In this case, the architecture achieved real time (at least 30 fps) when encoding Full HD or QFHD videos. If the standard-cells version is considered, the architecture surpassed the highest frame rate used in very high definitions (120 fps) even for UHD 8K videos.

These results were possible in function of the proposed algorithmic optimization and also in function of the efficient hardware design which is able to process 16 samples in parallel at high operation frequency.

Table VII. 2-D DCT Throughput Estimative

	Processing Rate (Msamples/sec)	Full HD fps	QFHD fps	UHD 8K fps
Stratix III	432.8	139.1	34.8	-
TSMC 65nm	11,872.3	3,817.4	954.3	238.5

VII. COMPARISONS WITH RELATED WORKS

As previously discussed, there are few works focusing on hardware design targeting FPGA devices for the 2-D 16x16 DCT.

Edirisuriya [15] proposed a hardware design, which achieved interesting results in terms of performance. His paper claims that the described architecture is capable to achieve about 0.9 GHz operation frequency when targeting an ASIC implementation. Moreover, this architecture considers input words of four, eight or twelve bits whereas the HEVC defines the DCT input as nine bits, since the DCT input is given by a subtraction between eight-bits samples, generating nine-bits results. Thus, even targeting a HEVC implementation, the architecture described in [15] is not completely optimized for HEVC standard.

The architectures presented in [16], [17], [18], [19] and [20] do not focus on a single transform type or size, hampering a fair comparison with our work. However, the video coding process must take a lot of decisions along the whole processing and then, the parallelism exploration is necessary to allow real-time processing, especially when high resolution videos are considered. Thus, the use of only one multi-transform hardware to serially process all transforms sizes and types is not an interesting approach since the transforms will be a new system bottleneck. As consequence, the coding time will increase when compared to multiple single-sized architectures working in parallel.

The work described in [16] presents a technique for IDCT calculation that analyses the DC coefficient and the three lowest frequency coefficients of the input matrix. According to the value of them, the IDCT is only performed in these coefficients. This solution decreases hardware energy and area consumption. However, this technique compromises the image quality. It is also proposed a hardware design for this technique targeting 90nm ASIC and Virtex-6 FPGA implementations. Both implementations are able to process 48 QFHD frames per second operating at 128MHz, consuming 128K logical gates and 34,344 LUTs, respectively. Our work is focused in the forward DCT, but even with this difference, our design consumes less hardware and is able to reach a highest processing rate than [16].

The work proposed in [17] presents a multi-standard architecture for DCT computation of H.264/AVC and HEVC, processing either forward or inverse DCT. This architecture was synthesized in a Virtex-7 FPGA device and is able to process 2.2 billion samples per second, being able to real-time processing on UHD 8K videos. Our work is focused on a single-sized transform and targeted to other technologies, but even with these differences, our design is able to process almost 12 billion samples per second. This is a processing rate five times higher than [17].

The paper presented in [18] follows the same idea than [17], presenting a forward/inverse DCT and Hadamard. Two architectures were proposed, the first one using SRAM memory and another one using registers for the transposition matrix. Both architectures were synthesized for TSMC 90nm ASIC technology. The architecture based on registers used 412.5K gates and is able to work at 311MHz, consuming 30.5mW.

The reached throughput for 16x16 DCT was of 4.9 billion samples per second, with a latency of 38 cycles. Once again, our work targets a single-size transform for 45nm technology but is able to process twice more samples per second, use 27 less hardware resources and present lower latency if compared with [18]. The power consumption presented in [18] is contradictory, since two different information are presented. However, in both cases, our power consumption is higher than that presented in [18].

The work described in [20] presented a hardware implementation for the HEVC DCTs considering multiple transform sizes. Two types of architectures were proposed. The first one consists in an unique module of DCT 1-D calculation named folded architecture. The second one is a full parallel architecture, using two DCT calculation units. The architectures were synthesized in TSMC 90nm technology. The fully parallel implementation consumes 347K logical gates and process 32 samples per cycle (5.9 billion samples per second) whereas dissipating 67.57mW. The operation frequency considered is 187 MHz and the latency is of 32 cycles. When compared with our work, even targeting a different technology and processing only one transform size, it is possible to conclude that our work reached a processing rate twice higher than [20], using 23 times less hardware resources. This higher throughput comes at the cost of a power consumption twice higher than [20].

The work described in [14] presents an ASIC implementation for the 16x16 2-D DCT. Table VIII presents a comparison between our work, targeting the standard-cell synthesis, and the work described in [14]. Our design achieved lower latency when compared to [14]. Unfortunately, the work presented in [14] did not present its hardware consumption results, becoming impossible that comparison.

Table VIII. Comparison between related works

Parameter	This Work (ASIC)	Ahmed [14]
Technology	65nm	90nm
Hardware Consumption	14,954 Logic Gates	-
Max Frequency (MHz)	742	150
Power Consumption @ Max Frequency (mW)	135	10.580
Power Consumption @ 30 QFHD fps (mW)	4.4	-
Latency (cycles)	18	33
Samples per Clock Cycle	16	16
Samples per Second (x10^6) @ Max Frequency	11,872	2,400
HD 1080p fps (4:2:0) @ Max Frequency	3,817	768
QFHD fps (4:2:0) @ Max Frequency	954	192
UHD 8K fps (4:2:0) @ Max Frequency	238	48

VIII.CONCLUSIONS

This work proposes the design of a low hardware cost and high-throughput architecture for the 16x16 2-D DCT of the HEVC standard. The 2-D DCT architecture was designed exploring the separability property, then two 1-D DCT and one transposition buffer (with two register files) were used to generate the 2-D DCT hardware. The architecture was designed to be able to process 16 samples at each clock cycle, with a latency of 18 clock cycles.

The original 1-D DCT algorithm was analyzed and a variety of simplifications were proposed, aiming an efficient hardware implementation. The multiplications were replaced by shift-adds, in order to generate a multiplierless architecture. The equations were factorized to reduce the bit width of operators and sub-expressions were shared to reduce the hardware cost.

The 1-D DCT architectures were designed in a purely combinational fashion. An architectural design derived from the original HM 1-D DCT algorithm was also designed to evaluate the gains introduced by the proposed simplifications. Both architectures were described in VHDL and synthesized to a Stratix III Altera FPGA. The synthesis results showed that the optimized 1-D DCT consumes 72% less hardware resources and achieves an operation frequency 445% higher than the architecture based in the original algorithm. The 2-D DCT architecture was synthesized to Stratix III Altera FPGA and to TSMC 65nm standard-cells technology. The synthesis results show that the designed architecture is able to reach a throughput of up to 11.8 billion samples per second when targeting the standard-cells implementation at the maximum operation frequency. Thus, the architecture is able to process high resolutions videos in real time. For example, the 2-D DCT architecture is able to process 238 UHD 8K (7680x4320 pixels) frames per second, surpassing the highest frame rate defined in the literature for this resolution (120 fps). Even with this high throughput, the power consumption of this architecture was of 135.4mW considering the maximum frequency.

Comparing our work with the 16x16 DCT related work [11], it as possible to conclude that our architecture presented higher throughput, a lower latency, and a lower power consumption. The other published works focus on hardware designs for multiple transforms or are not optimized to the HEVC standard. However, even in these cases, our solution reached the highest throughput among all related works whereas presenting the lowest use of hardware.

ACKNOWLEDGMENT

We would like to thank CNPq, CAPES, and FAPERGS funding agencies for partially supporting this work.

REFERENCES

[1] Agostini, L. V. ; Azevedo, Arnaldo ; Staehler, W. ; Rosa, Vagner ; Zatt, Bruno ; Pinto, Ana Cristina ; Porto, Roger Endrigo Carvalho ; BAMPI, Sergio ; SUSIN, Altamiro . Design and FPGA Prototyping of a H.264/AVC Main Profile Decoder for HDTV. Journal of the Brazilian Computer Society, v. 13, p. 25-36, 2007.

[2] International Telecommunication Union. "ITU-T Recommendation H.264/AVC (03/05): advanced video coding for generic audiovisual services". 2005.

[3] International Telecommunication Union. "IT U-T Recommendation H.262 (11/94): generic coding of moving pictures and associated audio information – part 2: video". 1994.

[4] Joint Collaborative Team on Video Coding (JCT-VC). Available at: http://www.itu.int/en/ITU-T/studygroups /com16/video/Pages/jctvc.aspx

[5] D. Grois. "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders", in Picture Coding Symposium (PCS), San Jose, CA, 2013.

[6] G. Sullivan, et al. Overview of the High Efficiency Video Coding (HEVC) Standard. IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), volume 22, issue 12, p. 1649 – 1668, 2012.

[7] Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 – "HM3: High Efficiency Video Coding (HEVC) Test Model 3 Encoder Description", 5th Meeting: Geneva, CH, 16-23 March, 2011.

[8] W. Han, et al. "Improved video compression efficiency through Flexible Unit Representation and Corresponding Extension of Coding Tools", IEEE Transactions on Circuits and Systems for Video Technology. Vol. 20, pp. 1709-1720. December 2010.

[9] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," IEEE Trans. Commun.,vol. COM-25, no. 9, pp. 1004–1009, Sep. 1977.

[10] Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11 – "HM5: High Efficiency Video Coding (HEVC) Test Model 5 Encoder Description", 7th Meeting: Geneva, CH, 21-30 November, 2011.

[11] ISO/IEC-JTC1/SC29/WG11, "HEVC Reference Software Manual", ed. Geneva, Switzerland, 2011.

[12] F. Bossen, et al. HEVC Complexity and Implementation Analysis. IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), volume 22, issue 12, p. 1685 – 1696, 2012.

[13] I. Richardson. H.264/AVC and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia. Chichester: John Wiley and Sons, 2003.

[14] A. Ahmed, et al. "VLSI implementation of 16-point DCT for H.265/HEVC using walsh hadamard transform and lifting scheme". in Multitopic Conference (INMIC), Pakistan, 2011.

[15] A. Edirisuriya, et al. "A Multiplication-free Digital Architecture for 16x16 2-D DCT/DST Transform for HEVC". in Electrical & Electronics Engineers in Israel (IEEEI), Israel, 2012.

[16] E. Kalali, et al. "A Low Energy HEVC Inverse DCT Hardware". in IEEE Third International Conference on Consumer Electronics (ICCE-Berlin), Germany, 2013.

[17] T. Dias; N. Roma; L. Souza. "High Performance Multi-Standard Architecture for DCT Computation in H.264/AVC High Profile and HEVC Codecs". in Conference on Design and Architectures for Signal and Image Processing (DASIP), Cagliari, Italy, 2013.

[18] J. Zhu; Z. Liu; D. Wang. "Fully Pipelined DCT/IDCT/Hadamard Unified Transform Architecture for HEVC Codec". in IEEE International Symposium on Circuits and Systems (ISCAS), Beijing, China, 2013.

[19] M. Budagavi, et al. Core Transform Design in the High Efficiency Video Coding (HEVC) Standard. IEEE Journal of Selected Topics in Signal Processing, volume 7, issue 6, p. 1029 – 1041, 2013.

[20] P. Kumar, et al. Efficient Integer DCT Architectures for HEVC, IEEE Transactions on Circuits and Systems for Video Technology, volume 24, issue 1, p. 168 – 178, 2014.

[21] R. Jeske, et. al."Low cost and high throughput multiplierless design of a 16 point 1-D DCT of the new HEVC video coding standard".in Programmable Logic (SPL), Bento Gonçalves, RS, 2012, pp.1-6.