Advanced Design of TQ/IQT Component for H.264/AVC Based on SoPC Validation

A. Ben Atitallah[(1)], H. Loukil[(2)] , P. Kadionik[(3)], N. Masmoudi[(2)]
[(1)] University of Sfax, High Institute of Electronics and Communications, BP 868, 3018 Sfax, TUNISIA
[(2)] University of Sfax, National School of Engineering, BP W, 3038 Sfax, TUNISIA
[(3)] IMS laboratory –ENSEIRB-MATMECA - University Bordeaux 1 - CNRS UMR 5218,
351 Cours de la Libération, 33 405 Talence Cedex, France
ahmed.benatitallah@isecs.rnu.tn

Abstract: - This paper presents an advanced hardware architecture for integer transform, quantization, inverse quantization and inverse integer transform modules dedicated to the macroblock engine of the H.264/AVC video codec standard. Our highly parallel and pipelined architecture is designed to be used for intra and inter prediction modes in H.264/AVC. The TQ/IQT design is described in VHDL language and synthesized to Altera Stratix II FPGA and to TSMC 0.18µm standard-cells. The throughput of the hardware architecture reaches a processing rate up to 1070 millions of pixels per second at 171.4 MHz when mapped to standard-cells. In addition, a system on a programmable chip (SoPC) implementation and validation of the proposed design as an IP core is presented using the embedded Altera development board.

Key-Words: - H.264/AVC, video coding, FPGA, SoPC.

1. Introduction

The H.264/AVC standard , known as MPEG-4 part 10 [1, 2], achieves significant improvements over the previous standards such as H.263 [3] and MPEG-4 [4] simple profile in terms of compression rates [5]. The H.264/AVC encoder includes several blocks such as Motion Estimation and Motion Compensation (ME/MC), Intra prediction, Transform and Quantization (TQ), Inverse Quantization and Transform (IQT) and entropy coder. Fig. 1 shows the H.264 encoder scheme that is a hybrid encoder similar to previous standards [1].

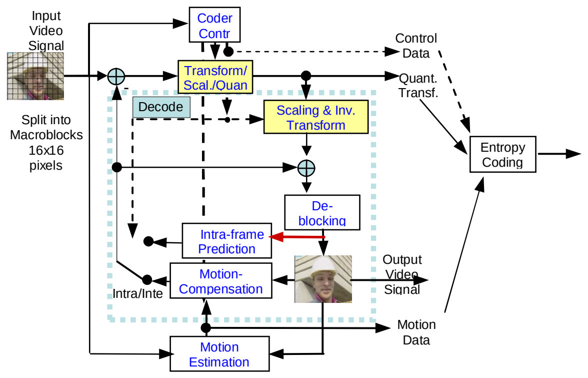

Fig. 1 The H.264 encoder scheme

A coded video sequence in H.264/AVC consists of a sequence of coded pictures. Each picture is divided into MacroBlocks (MB) of 16x16 pixels. Each MB performs intra and inter prediction mode to find the best predictor in the spatial and temporal domains. There are two kinds of intra prediction modes in H.264. One is intra 4x4 prediction and the other is the intra 16x16 prediction. The inter prediction is implemented by motion estimation prediction on several reference frames. The residual MB is then obtained by subtracting predictor from the original. The residual MB is transformed using an integer transform, and the transform coefficients are quantized followed by zigzag ordering and entropy coding. For more details, interested readers can refer to [6, 7, 8] for a quick and thorough study.

For coding the residual data block into inter or intra 4x4 prediction mode, the TQ/IQT component is composed by Integer Cosine Transform (ICT), Quantization (Q), Inverse Quantization (IQ) and Inverse ICT (IICT). But in the intra 16x16 prediction mode, the TQ/IQT component uses both 4x4 ICT and Hadamard transforms with a quantization of the transformed Hadamard coefficients. The different types of prediction modes make the implementation of the control flow more complex. In literature, there are several papers [9, 10, 11, 12] discussing hardware implementation of the transform and quantization block only. But few works [13] describe a VLSI design with all the parts of the TQ/IQT component for different types of the prediction modes.

In order to reduce complexity and to improve performances of the H.264/AVC video algorithm, we focus in this paper on the development on a customized and optimized fast hardware module for the TQ/IQT component that can be integrated and evaluated into the form of a hardware IP block (Intellectual Property) with the other H.264/AVC blocks in a system on a programmable chip (SoPC). The main idea of our IP block is to exploit advantages of the parallel and pipelined structures that can be efficiently implemented in hardware using VHDL (VHSIC Hardware Description Language) language.
The rest of the paper is organized as follows: section 2 presents an overview of the H.264 TQ/IQT algorithm. Section 3 describes the proposed TQ/IQT design in detail and shows the implementation results and the comparison with previous works. The performance evaluations of the TQ/IQT component under the Altera system on a programmable chip (SoPC) is presented in section 4. Finally, section 5 concludes the paper.

2. Overview of The H.264 Transform and The Quantization Algorithms

A more detailed flow of the TQ/IQT component is presented in Fig. 2. The input to the forward transform algorithm is a 4x4 block of residual data obtained by dividing the residual MB into sixteen 4x4 blocks as shown in Fig. 3. From Fig. 3, we can see that there are three different transforms used in H.264/AVC [1], one for all 4x4 residual data, another 4x4 luminance DC coefficients of the MB that are coded in intra 16x16 mode, and the last one for 2x2 DC chrominance.

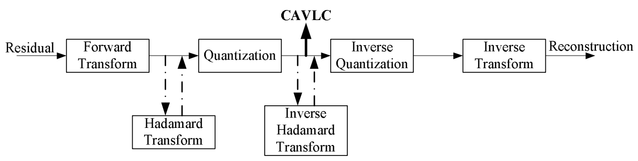

Fig. 2 Block diagram of TQ/IQT component

The transform and quantization algorithms process the residual blocks according to the prediction mode and send the resulting data to the entropy coding and reconstruction process in order to obtain a reference block for the next block. In this section, we present the theory of the different blocks constituting the TQ/IQT component.

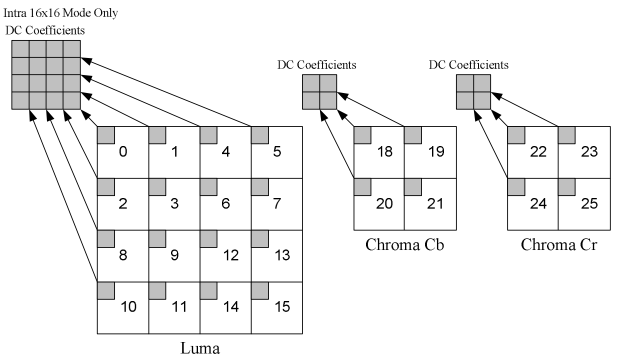

Fig. 3 Processing order of blocks in a macroblock

2.1 4x4 Integer Transform Algorithm

In recent years, there are many researchers working to design and develop the integer transform and integer DCT (Discrete Cosine Transform) for video coding. The DCT has been widely used in image and video coding standards like the popular 8x8 DCT used in previous standards while the H.264/AVC encoder is based on a 4x4 Integer Cosine Transform (ICT) that can be computed exactly with integer arithmetic in order to avoid inverse transform mismatch problems.
There are two types of 4x4 integer transforms for the residual coding. The first one is for luminance residual blocks and is described by (1) [2].

$$Y = MXM^T \tag{1}$$

Where the matrix X is the input 4x4 residual block and M is specified by the following:

$$M = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}$$

With: $a = 1/2, b = \sqrt{1/2}\cos(\pi/8), c = \sqrt{1/2}\cos(3\pi/8)$

Thus, (1) can be factorized in the following form [2]:

$$Y = (CXC^T) \otimes E \tag{2}$$

With:

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & d & -d & -2 \\ 1 & -1 & -1 & 1 \\ d & -2 & 2 & -d \end{bmatrix} \quad and \quad E = \begin{bmatrix} a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \\ a^2 & ab & a^2 & ab \\ ab & b^2 & ab & b^2 \end{bmatrix}$$

Where E is a matrix of scaling factors. The symbol $\otimes$ means that each component of CXC^T is multiplied by the corresponding coefficient in E. To reduce hardware implementation of the transform, the constant d is approximated by 0.5 and the constant b by $\sqrt{2/5}$. The final forward transform becomes [2]:

$$Y = (C_f X C_f^T) \otimes E_f \tag{3}$$

Where:

$$C_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad and \quad E_f = \begin{bmatrix} a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \\ a^2 & ab/2 & a^2 & ab/2 \\ ab/2 & b^2/4 & ab/2 & b^2/4 \end{bmatrix}$$

So, the scaling matrix E_f can be incorporated into the quantization process. Then $C_f X C_f^T$ becomes the core of a 2-D integer forward transform without multiplications. In fact, the C_f is the transform matrix of the 1-D forward transform and contains only 4 coefficients, 1, -1, 2 and -2 that can be implemented by shift and addition operations. The fast implementation for the 1-D forward transform is shown Fig. 4.

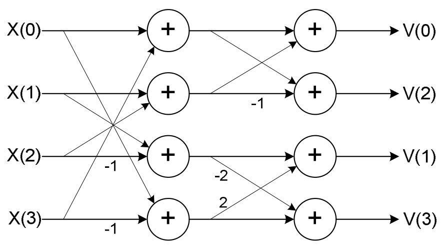

Fig. 4 Fast implementation of 4x4 ICT transform

The inverse transform is very similar to the forward transform and the complexity is the same. The coefficient of 1-D inverse transform C_i is given by (4).

$$C_i = \begin{bmatrix} 1 & 1 & 1 & 0.5 \\ 1 & 0.5 & -1 & -1 \\ 1 & -0.5 & -1 & 1 \\ 1 & -1 & 1 & -0.5 \end{bmatrix} \tag{4}$$

The other kind of transform is Hadamard Transform (HT). It is applied to the luminance DC terms in 16x16 intra prediction mode. The Hadamard transform is defined by (5).

$$Y = H_f X H_f^T \tag{5}$$

With:

$$H_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

The Hadamard transform matrix is very similar to the forward transform matrix. The difference is to replace the coefficient 2 by 1 in the transform matrix. Therefore, the fast implementation for 1-D Hadamard transform is given in Fig. 5. The Inverse Hadamard Transform (IHT) is the same as the forward Hadamard transform because the transform matrix is symmetric.

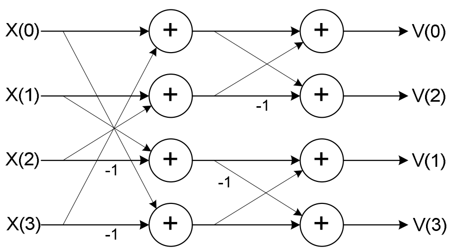

Fig. 5 Fast implementation of 4x4 Hadamard transform

2.2 4x4 Quantization Algorithm

The quantization is a significant source of compression in the encoded bit stream. Quantization takes advantage of the low sensitivity of the eye to reconstruction errors related to high spatial frequencies as opposed to those related to low frequencies [14]. Quick high frequency changes can often not be seen and may be discarded. Slow linear changes in intensity or colour are important to the eye. Therefore, the basic idea of the quantization is to suppress many of the nonzero transformed coefficients corresponding to high frequency components. In H.264/AVC, there are two types of quantization algorithm for the 4x4 integer transform. The first one is for the transformed coefficients of luminance residual block. The AC Quantization Operation (ACQ) is shown in (6) [2].

$$Z_{ij} = round(Y_{ij} \frac{PF}{QStep}) \tag{6}$$

Where, Y_{ij} is the coefficient after integer core transformation, PF is the scaling factor of integer transform, *QStep* is the quantization step size and Z_{ij} is the coefficient after quantization. To simplify the arithmetic, the quantization stated in (6) can be rewritten as (7) and *PF/QStep* is implemented as a multiplication by a MF factor (Multiplication Factor) and a right-shift register to avoid division operations.

$$Z_{ij} = round(Y_{ij} \frac{MF}{2^{qbits}}) \tag{7}$$

Where:

$$\frac{MF}{2^{qbits}} = \frac{PF}{Qstep} \tag{8}$$

$$qbits = 15 + floor(QP/6) \tag{9}$$

In H.264/AVC, *QStep* can be varied from 0.625 to 224 and is controlled by a Quantization Parameter (QP). There are 52 quantization parameter values from 0 to 51. These values are arranged so that an increase of 1 in QP means an increase of *QStep* by approximately 12 % [2].

Table 1. MF Multiplication Factor in H.264/AVC

QP	Positions (0,0),(2,0),(0,2),(2,2)	Positions (1,1),(1,3),(3,1),(3,3)	Other positions
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

The MF value depends on QP and the position (i,j) of the element in the matrix as shown in Table 1. The MF factor remains unchanged for QP>5 that can be calculated by using (10).

$$MF_{QP>5} = MF_{QP=QP\%6} \qquad (10)$$

Then (7) can be represented by using integer arithmetic [2] as:

$$|Z_{ij}| = (|Y_{ij}|.MF + f) >> qbits \qquad (11)$$

Where f is a parameter used to avoid rounding errors. It depends on prediction type of the block and QP.

After calculation of $|Z_{ij}|$, the sign of the Y_{ij} is added to obtain Z_{ij}:

$$sign(Z_{ij}) = sign(Y_{ij}) \qquad (12)$$

The Inverse of AC Quantization (IACQ) is done by using the following equation:

$$Y_{ij} = Z_{ij}.QStep.PF.64 \qquad (13)$$

Where Z_{ij} is the quantized coefficient, Y_{ij} is a scaled coefficient, PF is the prescaling factor for the inverse transform and the factor 64 is used to avoid rounding errors. We can write (13) as:

$$Y_{ij} = Z_{ij}.V_{ij}.2^{floor(QP/6)} \qquad (14)$$

V_{ij} is specified in the standard as shown in Table 2.

Table 2. Multiplication Factor V in H.264/AVC

QP	Positions (0,0),(2,0),(0,2),(2,2)	Positions (1,1),(1,3),(3,1),(3,3)	Other positions
0	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

The other type of quantization is for DC coefficients of 4x4 Hadamard transform. The DC Quantization (DCQ) is shown in 15.

$$|Z_{ij}| = (|Y_{ij}|.MF_{(0,0)} + 2f) >> qbits + 1 \qquad (15)$$

$$sign(Z_{ij}) = sign(Y_{ij})$$

Where $MF_{(0,0)}$ is the multiplication factor for position (0,0) in Table 1. The inverse of DC quantization (IDCQ) is defined as:
If QP≥12 then:

$$Y_{ij} = Z_{ij}.V_{(0,0)}.2^{floor(QP/6)+2} \qquad (16)$$

Otherwise:

$$Y_{ij} = \left[Z_{ij}.V_{(0,0)} + 2^{1-floor(QP/6)}\right] >> (2 - floor(QP/6))$$

Where $V_{(0,0)}$ is the multiplication factor for position (0,0) in Table 2.

3. Hardware Architecture of The H.264 Transform and Quantization

This section presents an efficient parallel hardware architecture for H.264 TQ/IQT component in order to support large spectrum of real-time applications such as HDTV (High Definition TV) 720p (1280x720) and 1080i (1920x1088). This component is composed essentially by two parts: the transform (ICT, IICT, HT and IHT) part and the quantization (ACQ, IACQ, DCQ and IDCQ) part. The hardware architecture and the organization of the different blocks that compose these two parts affect the TQ/IQT component performances and the silicon area cost. In this section, we present then the hardware design of the internal modules and of the whole TQ/IQT component, the synthesized results in to Altera Stratix II FPGA and to TSMC 0.18µm standard-cells and the comparison with the previous works.

3.1 Implementation of the 4x4 Integer Transform

There are several papers discussing on the VLSI implementation of 2-D integer transform for H.264. Thus, implementation of fast 2-D transform can be classified into two categories: row/column decomposition approach [9] and direct two-dimensional approach [15]. However, the implementation of the direct 2-D transform requires much more effort and large silicon area than that for the row/column approach [16] that is used to implement 2-D transform.

The proposed architecture for the 2-D integer transform uses 4x4 parallel input data. A block diagram of this architecture is shown in Fig. 6. This diagram contains two 1-D transform units and a control unit that provides clocks and others control signals such as the *Done_ICT* output flag signal to indicate that outputs coefficients are valid.

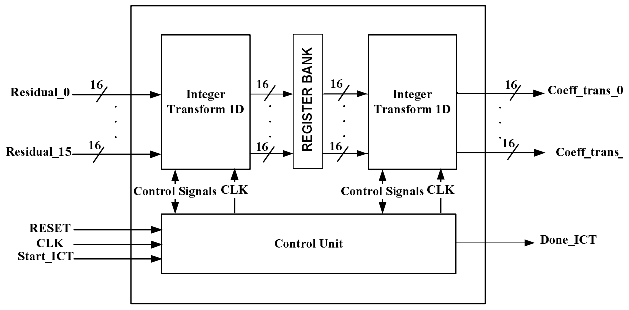

Fig. 6 Architecture of the 2-D integer transform

The 1-D transform unit is presented by Fig. 7 and is implemented by using the fast data-flow algorithm like Fig. 4 and 5. This fast algorithm uses only addition, subtraction and shift operations. Thus, the 1-D transform is designed to process 16 pixels/cycle by computing the transform of four lines in parallel.

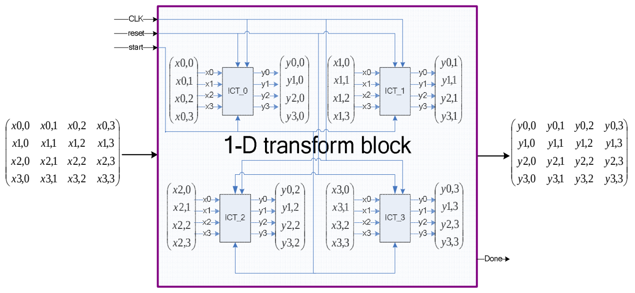

Fig. 7 Architecture of the 1-D integer transform

In Fig. 6, the 16 x 16-bit residual inputs data of the transform is captured from the outside environment through *residual_0..15* signal. Moreover, after intra or motion estimation prediction, the dynamic range of the inputs data is 9 bits, i.e. from -256 to +255. Because we have used operations like additions, subtractions and shifts, the dynamic range of the pixel data is extended to a 16-bit value [17]. So, the 4x4 residual data are processed in parallel by the transform block. This block consists of two cascaded 1-D transform units, i.e. one 1-D row transform and one 1-D column transform. The separable nature of the 2-D transform given by (1) is exploited by computing the 1-D transform on the rows and then the 1-D transform on the columns. In fact, the first transform calculates $V = C_f X$ and the second calculates $Y = VC_f^T$. The first 1-D transform computes the row of C_f and column of X while the second 1-D computes the row of V and column of C_f^T where C_f is the transform matrix, X is the input coefficient, V is the intermediate row/column matrix and Y contains the transformed coefficients.

3.2 Implementation of the 4x4 Quantization

The purpose to design a hardware quantization module is to reduce computation complexity in order to calculate the quantization coefficients in real-time. It allows this module to be used as a computing resource module by the other modules of the H.264 encoder. The hardware quantization components for the AC and DC coefficients rescale the transformed coefficients according to the quantization step as defined by (11) and (15). The proposed architecture for 4x4 AC and DC quantization is shown in Fig. 8.

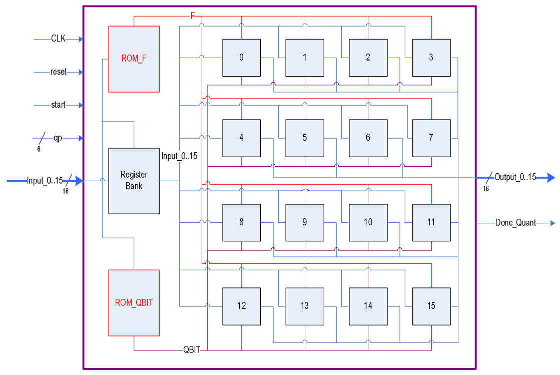

Fig. 8 Architecture of AC and DC quantization module

It contains sixteen Processing Elements (PE), the register bank for storing the input pixels noted *input_0..15* and two read only memories (ROM) for storing QBIT and F values noted ROM_F and ROM_QBIT, respectively. The AC and DC quantization modules receive the sixteen 16 bits transformed coefficients in the same time and quantize these coefficients according to the QP factor in four clock cycles.

The main component of the quantization architecture is the PE which shown in Fig. 9. It is composed by four basic components and a control unit and is designed to quantize one transformed coefficient every four clock cycles. An integer multiplier assures the multiplication of AC and DC transformed coefficients with the corresponding MF(i,j) factor that is stored into the ROM_MF memory as shown in Table 1 and selected according to the QP modulo 6 value. The adder makes the sum of value given by the multiplier with the F parameter given by the ROM_F memory. A shifter register shifts the result set by the adder by *qbits* (varies 15 to 23 according to the value of QP). The multiplier, the adder, the shifter and the ROM_MF memory modules take one clock cycle each one. The control unit receives input control signals (*Reset*, *Clk*, *Start_Quant*) and generates all internal control signals for each stage and the output flag (*Done_Quant*) signal to indicate that the quantized coefficient is valid.

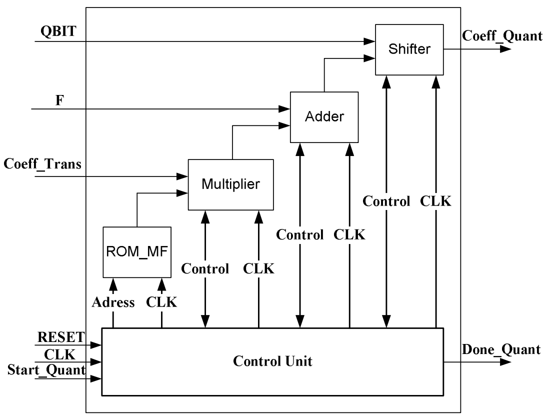

Fig. 9 PE module of the quantization architecture

The quantization of the 4x4 AC and DC transformed coefficients are made by using the PE module. In fact, we can discuss on two methods to implement the quantization architecture:

- For a speed optimization, we use 16 PE modules that are executed in parallel and they are structured in a 4x4 array as depicted in Fig. 8. The role of the (i,j) module ($0 \leq i, j \leq 3$) is to calculate the (i,j) quantized coefficients. To accomplish this task, the quantization architecture fetches sixteen transformed coefficients and dispatches them to the sixteen modules. Therefore, this architecture receives each block of the 4x4 transformed blocks in four cycles and provides the quantized coefficients of the MB corresponding to the PE module with 64 clock cycles.
- For a silicon area optimization, we can call sequentially 16 times the PE module to calculate the 4x4 quantized coefficients. In this case, the quantization architecture processed all 4x4 transformed blocks with 1024 clock cycles. We can conclude that the second method is about 16 times slower than the first one, but it also requires considerably less silicon area. We use then the first method for a hardware implementation of the AC and DC quantization architecture.

The inverse AC and DC quantization components share the same architecture design with AC and DC quantization presented in Fig. 8. The differences between the architecture for the quantization and for the inverse quantization are presented in the PE module. In fact, for computing the inverse AC quantization values respecting (14), we have just eliminated the addition block from the PE module depicted according to Fig. 9. On the contrary, to implement (16) the inverse DC quantization, we use the same PE module of the DC quantization. But the shifter block is implicated when QP<12. The AC and DC inverse quantization architecture is designed to provide sixteen coefficients every three and four clock cycles, respectively.

3.3 Design architecture of the TQ/IQT component

The block diagram of the proposed hardware architecture for H.264 TQ/IQT component is shown in Fig. 10 which contains the transform and quantization parts and the control unit. Our proposed architecture is used to code and decode the residual coefficients obtained by intra 4x4, intra 16x16 and inter prediction modules. The TQ/IQT architecture has a 16 x 16-bit inputs and outputs data. It receives in parallel the sixteen residual coefficients each 2 clock cycles and provides two types of data that are obtained by the TQ module, one for coding the entropy and the other one for the IQT module in order to reconstruct the residual pixels. Our architecture could process sixteen coefficients per *N* clock cycles and depends on the prediction mode. The *valid_mode* signal has a 2-bit length and can select the prediction mode (“00”: intra 4x4, “01”: intra 16x16 and “10”: inter).

The TQ/IQT processing cycle reduction is a crucial point in implementing the H.264/AVC. So, in the proposed hardware architecture, we have used sixteen parallel input data sets and treated sixteen data sets simultaneously in order to reduce clock cycles for TQ/IQT computation. For implementing the quantization and inverse quantization modules, we can see from Fig. 10 that sixteen quantization and inverse quantization modules are used in parallel for fast processing. In Fig. 10, the FIFO (First In First Out) memory and DC coefficient register files are used to store 240 x 23-bit dequantized coefficients and 16 x 16-bit DC coefficients respectively. The MUX block selects the dequantized coefficient form the FIFO and the IDCQ modules when “*Valid_mode*=01” and from the IACQ module when “*Valid_mode*=00 or 10” and transfers these coefficients to the inverse 2-D transformation module. The control unit receives input control signals (*Clk*, *Reset*, *Start*, *Valid_mode*) and generates all internal control signals for each stage and output control signals the communication with other hardware modules.

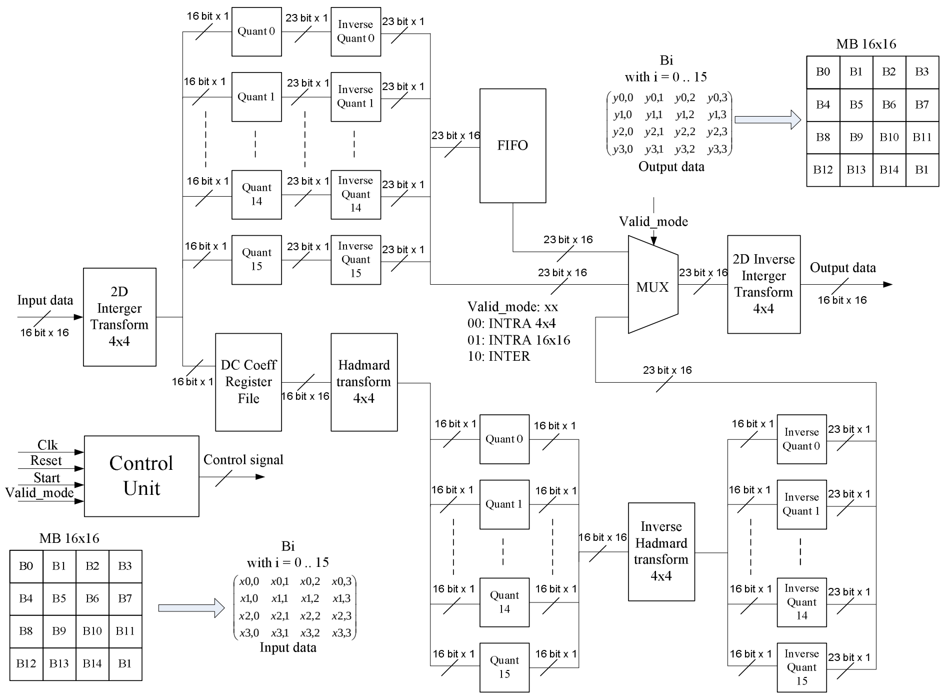

Fig. 10 Hardware architecture for the TQ/IQT component

Referring to Fig. 11, when the prediction mode is intra 4x4, 11 clock cycles are needed to perform one 4x4 block. In fact, the data processing is made in successively for each block by the following steps: (1) the block is processed by the 4x4 ICT in two clock cycles, (2) and (3), the ACQ and IACQ are performed in four and three clock cycles, respectively and (4) the 4x4 IICT is applied in two clock cycles. After, only the border samples of each 4x4 block are sent to the prediction module and the other samples are discarded. In this case, the intra prediction must be idle only during eleven clock cycles. Then, 176 clock cycles are needed to process one MB (11*16=176 cycles) if intra 4x4 mode is chosen. But, we can note that a pipelined processing is applied on the inter mode for data independency.

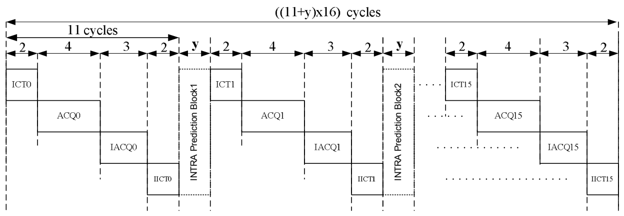

Fig. 11. Hardware cycles of the TQ/IQT module in intra 4x4 mode

As shown in Fig. 12, eleven cycles are required for processing the first block of the MB and two cycles for the other fifteen blocks. So, 41 cycles are needed for the TQ/IQT component to reconstruct one MB when the inter mode is selected. To reduce the number of processing cycles for one MB into intra 4x4 mode to 41 cycles, we can choose the method that gives data independence [18].

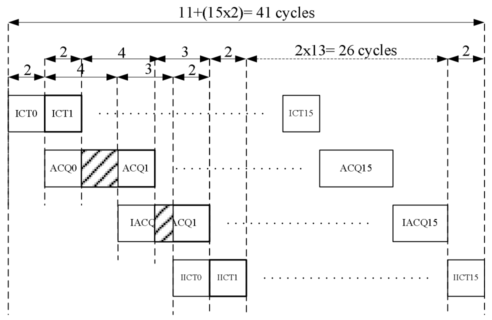

Fig. 12 Hardware cycles of the TQ/IQT module in inter mode

Referring to Fig. 10, when the intra 16x16 mode is chosen, the transformation method uses both 4x4 integer and Hadamard transforms. In fact, for coding a MB in intra 16x16 mode, sixteen blocks have to be 4x4 integer transformed with AC quantization and dequantization. The FIFO is used to store the AC dequantized coefficients until the reconstruction of the 4x4 DC values are obtained from a 4x4 ICT transform and reconstructed by using the HT, DCQ, IHT and IDCQ modules. Finally, the AC and DC coefficients are combined to apply the IICT module. Fig. 13 shows that 77 clock cycles are needed to process one MB by the TQ/IQT component in intra 16x16 mode. Indeed, the ICT, ACQ and IACQ modules are used together in pipelined mode and take 39 cycles. In fact, the reconstruction of the DC coefficient requires twelve cycles. One cycle is needed to read data from the FIFO. The IICT module takes 32 cycles.

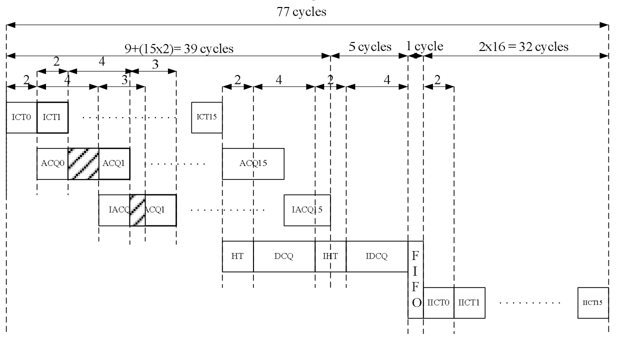

Fig. 13 Hardware cycles of the TQ/IQT module in intra 16x16 mode

The designed architecture for the TQ/IQT was described in VHDL language. The architecture was validated using Mentor Graphics ModelSim and synthesized considering two different technologies: Altera Stratix II EP2S60F1020 FPGA circuit [19] with speed 3 grade and TSMC 0.18µm standard-cells [20] technology. The synthesis targeted the Altera FPGA was made using the Altera Quartus II tool and the standard-cells version was generated using the Loenardo Spectrum synthesis tool.
Table 3 shows the hardware cost in terms of ALUTs (Adaptive Look-Up Tables) and DLRs (Dedicated Logic Registers) count for FPGA and gate count for standard-cells, operation frequency and data throughput rate (Mpixels/s) of the each proposed internal module and the whole TQ/IQT proposed component.

3.4 Synthesis Results

Table 3. H.264/AVC TQ/IQT Component Synthesis Results

Module	**Altera EP2S60F1020C3**				**TSMC 0.18µm**		
	# of LUTs	**# of DLRs**	**Freq. (MHz)**	**Throughput (Mpixels/s)**	**# of Gates**	**Freq. (MHz)**	**Throughput (Mpixels/s)**
ICT	1024	514	480.31	3842	8716	258.8	2070
IICT	1504	936	405.0	3240	13199	195.9	1567
HT	1088	506	418.0	3344	8570	241.0	1928
IHT	1056	516	436.87	3495	8482	258.5	2068
ACQ	4071	292	221,09	884	18765	180.4	722
IACQ	1974	387	269.4	1437	9988	243.6	1299
DCQ	4009	276	211.15	885	19524	180.6	722
IDCQ	5419	388	230.36	921	23677	222.6	890
TQ/IQT: Inter	21413	5363	152.67	953	116437	171.4	1070
TQ/IQT: Intra 16x16				508			570
TQ/IQT: Intra 4x4				222			249

In Table 3, we can find that the proposed TQ/IQT design achieved 171.4 MHz as maximum operation frequency when mapped to standard-cells. With this operating clock frequency, the data throughput of our proposed architecture can achieve up to 570 Mpixels/s, 249 Mpixels/s and 1,070 Mpixels/s that depends on the prediction mode, intra 16x16, intra 4x4 and inter modes, respectively. Furthermore, Table 3 also shows that the proposed design uses 116437 gates when the TSMC 0.18µm technology is adopted.
The most important result presented in Table 3 is the maxima throughput of the internal TQ/IQT component that, in all case, is sufficient to operate in H.264/AVC encoder for HDTV. Considering a HDTV 1080i (1920x1088@30Hz) video format and a downsampling relation of 4:2:0 then the required throughput is 94 Mpixels/s. The TSMC 0.18µm standard-cell design of the TQ/IQT component is able in worse case, i.e., when the intra 4x4 prediction mode is always chosen, to reach a processing rate of 249 Mpixels/s which is outperforming the HDTV requirement. The FPGA design can reach a throughput superior to 222 Mpixels/s, also surpassing the performance demanded by H.264/AVC encoder. So, aiming the target application, appropriate frequency can be chosen for the specific application in order to achieve lower power consumption.

E. Comparison with Previous Works

The main purpose of the proposed TQ/IQT architecture is to optimize the hardware resource by using same hardware architecture for the intra 16x16, intra 4x4 and inter prediction modes in H.264/AVC and increase the data throughput rate by exploiting the advantages of the parallel and pipelined structures.
In this section, we will compare the performance of the each proposed internal module and the whole TQ/IQT design with other exiting design found in literature which is always a difficult work due to that different designs might adopt different design considerations,

diverse technologies, various supply voltages and so on.

Considering the hardware efficiency on a design, we adopt the performance index of Data Throughput rate per Unit Area (denoted as DTUA) defined as the ratio of data throughput rate over hardware cost (in terms of gate count). When adopting the DTUA as the comparison index, the higher the DTUA index is, the more efficient the design is.

Table 4 shows the performance comparisons of the high efficiency proposed TQ/IQT design with reported data from the existed designs [9, 10, 11, 12, 13] in terms of gate count for standard-cells, data throughput rate (Mpixels/s) and DTUA (pixels/s/gate). In fact, the designs [9] and [10] use the R-C decomposition method to implement the H.264/AVC 2-D integer transform and were designed in a SMIC 0.35µm and UMC 0.18µm technology, respectively. Moreover, according to the Table 4, our proposed ICT design is better than the corresponding designs in [9] and [10] in terms of 16.04 and 2.58 times higher data throughput rate as well as 6.48 and 4.05 times (DTUA index) more efficient than the designs [9] and [10], respectively.

The design [11] proposes the implementation only of the integer transform and quantization modules for H.264 on FPGA technology. The results shown in Table 4 indicate, when the intra 4x4 prediction is selected (worse case), that the whole proposed TQ/IQT design owns 3.6 times higher data throughput rate and 30.57 times more efficient in terms of the DTUA index than the design [11].

The design presented in [12] realizes also just implementation of the transform and quantization modules which were designed in a TSMC 0.18µm technology. The DTUA index of the proposed ICT and ACQ designs shown in Table 4 indicate that they are 1.09 and 1.41 times more efficient than the design [12], respectively with similar throughput.

The last design analyzed was presented in [13] which achieves the low cost hardware implementation of H.264 forward transform and quantization and inverse transform and quantization in UMC 0.18µm technology and reaches an operation frequency of 210 MHz. This design can be used for different H.264 prediction modes. Furthermore, considering the worst case, i.e., when intra 4x4 prediction mode is chosen, and according to the reported data in the design [13], the whole proposed TQ/IQT design provides 11.31 times higher data throughput rate on the other hand the DTUA index in Table 4 tell that is 13.37 times more efficient than the corresponding design in [13] when our TQ/IQT design operates at 171.4 MHz.

Thus, our proposed TQ/IQT design can increase the data throughput rate with less hardware resource compared to the previous works.

Table 4. Performance Comparisons of the High Efficiency Proposed TQ/IQT Design with Reported Data from the Existed Designs

Module	[9] SMIC 0.35µm			[10] UMC 0.18µm			[11] FPGA		
	# of Gates	Throughput (Mpixels/s)	DTUA (pixels /s/gate)	# of Gates	Throughput (Mpixels/s)	DTUA (pixels /s/gate)	# of Gates	Throughput (Mpixels/s)	DTUA (pixels /s/gate)
ICT	3524	129	36.61 K	13651	800	58.6 K	1057000	69	0.07 K
ACQ	-	-	-	-	-	-			
IICT	-	-	-	-	-	-	-	-	-
IACQ	-	-	-	-	-	-	-	-	-
HT	-	-	-	-	-	-	-	-	-
DCQ	-	-	-	-	-	-	-	-	-
IHT	-	-	-	-	-	-	-	-	-
IDCQ	-	-	-	-	-	-	-	-	-
TQ/IQT: Inter	3524	129	36.61 k	13651	800	58.6 K	1057000	69	0.07 K
TQ/IQT: Intra 16x16									
TQ/IQT: Intra 4x4									

Module	[12] TSMC 0.18µm			[13] UMC 0.18µm			Proposed TSMC 0.18µm		
	# of Gates	Throughput (Mpixels/s)	DTUA (pixels /s/gate)	# of Gates	Throughput (Mpixels/s)	DTUA (pixels /s/gate)	# of Gates	Throughput (Mpixels/s)	DTUA (pixels /s/gate)
ICT	11727	2552	217.62 K	x	-	-	8716	2070	237.54 K
ACQ	39892	1085	27.2 K	x	-	-	18765	722	38.45 K
IICT	-	-	-	x	-	-	13199	1567	118.74 K
IACQ	-	-	-	x	-	-	9988	1299	130.07 K
HT	-	-	-	x	-	-	8570	1928	224.97 K
DCQ	-	-	-	x	-	-	19524	722	37.0 K
IHT	-	-	-	x	-	-	8482	2068	243.81 K
IDCQ	-	-	-	x	-	-	23677	890	37.6 K
TQ/IQT: Inter	47762	644	13.48 K	130505	22	0.16 K	116437	1070	9.19 K
TQ/IQT: Intra 16x16								570	4.9 K
TQ/IQT: Intra 4x4								249	2.14 K

4. Validation of The TQ/IQT Component in The HW/SW Context

4.1 Overview

With increasing of FPGA device densities, audacious challenges become feasible and the integration of embedded SoPC (System on Programmable Chip) systems is significantly improved [21]. Furthermore, SoPC became a reality with softcore processor, which is a microprocessor fully described in software, usually in a VHDL, and capable to be synthesized in programmable hardware, such as FPGA. Softcore processors can be easily customized to the needs of a specific target application (e.g. multimedia embedded systems). The two major FPGA manufacturers provide commercial softcore processors. Xilinx offers its MicroBlaze processor [22], while Altera has Nios and Nios II processors [23]. The benefit of a softcore processor is to add a micro-programmed logic that introduces more flexibility. A HW/SW approach is then possible and a particular functionality can be developed in software for flexibility and upgrading completed with hardware IP blocks (Intellectual Property) for cost reduction and performances.

4.2 The SoPC embedded platform

For SW implementation of image and video algorithms, the use of a microprocessor is required. The use of additional HW for optimization contributes to the overall performance of the algorithm. For the highest degree of HW/SW integration, customization and configurability, a softcore processor was used.
To verify functionality and performances of our TQ/IQT coprocessor, we have integrated the core into a SoPC platform using an Altera Nios II development board. The heart of the board is the Altera Stratix II EP2S60F672C3 FPGA circuit that was chosen for its great capability for integrating both hardware and software into one codesign flow [24]. The main components of the SoPC embedded platform are illustrated in Fig. 14. The proposed embedded SoPC platform shown in Fig. 14 consists of three major parts, including the NIOS II softcore processor, the TQ/IQT coprocessor and the peripheral interface modules. All these modules are connected to the Avalon Bus that is a configurable bus architecture that is auto generated for interconnecting peripherals. The Altera NIOS II softcore processor (*FAST version*) is configured as follows: a 32-bit scalar RISC processor with Harvard architecture, 6 stage pipeline, 1-way direct-mapped 64KB data cache, 1-way direct-mapped 64KB instruction cache and can gives up to 200 MIPS. The peripheral I/O modules are interfaces for 16MB flash, 16MB SDRAM and a serial UART port.
In this work, we use the μClinux as an operating system to control the functionality of the design. Linux for embedded systems (or embedded Linux) gives us several benefits: It is ported to most of processors with or without Memory Management Unit (MMU). A Linux port is available for the NIOS-II softcore. Most of classical peripherals are ported to Linux. A file system is available for data storage. A network connectivity based on Ethernet protocols is well suited for data recovering.

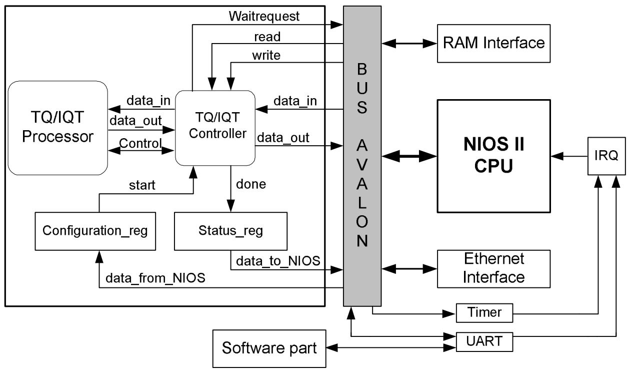

Fig. 14 Our SoPC embedded platform

Fig. 14 presents communications between the NIOS II processor and the TQ/IQT coprocessor. The NIOS II processor executes a software program that is loaded into the SDRAM memory. This software is written in C language and is used to communicate with the PC host through the UART serial port. In fact, the software program receives data through the UART port and checks if the TQ/IQT coprocessor is not busy with the *waitrequest* signal. In this case, our coprocessor loads the residual coefficients of the MB through the 32-bit *data_in* signal and activates the data processing. During the calculation step, the coprocessor is busy and can not be accessed. At the end of processing, the *waitrequest* signal has a low level state and the coprocessor provides the processed coefficients through the 32-bit *data_out* signal. Indeed, in the purpose of using the 32-bit bus size, each two 16-bit residual and processed coefficients must be processed as a 32-bit long word in order to decrease the memory access. The configuration and status register control the state of the TQ/IQT coprocessor.
The SoPC embedded platform is designed for accelerating computation for the H.264/AVC encoder and can be easily modified or extended for different video applications. It is synthesized with Quartus II tools for FPGA target and it uses 65 % of the ALUTs,

45 % of the RAM blocks, 50 % of the DSP blocks and 27 % of the IOBs. The platform architecture is running at 140 MHz or 7.14 ns for one clock cycle.

4.3 Performance Evaluation

For the FPGA HW/SW (Hardware/Software) performance evaluation, we have developed an optimized C language reference model of H.264/AVC encoder compatible with the NIOS II system. We have compared the output results of our C reference model with the JM 10.1 model [25] and we have confirmed the correctness of our model.

The H.264/AVC reference model is used to measure correctness of our TQ/IQT coprocessor in HW/SW context. For all experiments, the CIF 4:2:0 (352x288 pixels) test sequences coded at 30 frames/s. We focus on the following standard video test sequences: “Foreman”, “News”, “Claire”, and “Tb420”. These test sequences have different movement and camera operations. The average peak signal-to-noise ratio (PSNR) is used as a measure of objective quality.

Considering the performances of the SoPC embedded platform, we have measured the execution times of the TQ/IOT part in SW and HW/SW by using the NIOS II timer “*high_res_timer*” which can be used for the cycle-accurate time-frame estimation of a focused part of the SW code execution. The SW implementation of the TQ/IQT part in the SoPC embedded system takes about 1677µs which is an average time between the different video test sequences to compute one MB by the TQ/IQT part at 140 MHz in worse case i.e., when the intra 4x4 prediction mode is always chosen. On the other hand, the TQ/IQT coprocessor takes about 1.25µs and 47µs which is an average time to calculate the same MB by the HW and HW/SW solutions, respectively. From these results, we can conclude that the FPGA HW/SW solution is estimated up to 35 times faster than the SW solution. But, it is slower than the HW solution because the transfers data between the TQ/IQT coprocessor and SDRAM memory is very significant and can be improved by using DMA (Direct Memory Access) transfers.

Finally, Fig. 15 presents the original and the two reconstructed (one from SW, the other from HW/SW) of the 12th frame of the test video sequences. We can see from Fig. 15 that the HW/SW solution has a same image quality compared to the SW solution since we work in an integer environment. These results show an efficient and a high performance hardware architecture of the proposed TQ/IQT component. In fact, the parallel and pipelined hardware design can increase the data throughput with same image quality compared to the SW solution.

5. Conclusion

In this paper, the proposed TQ/IQT architecture is used to code and decode the residual coefficients obtained by the intra and inter prediction modes. It can operate at a maximum frequency of 171.4 MHz in TSMC 0.18µm standard-cells implementation. We have presented a modern implementation of the complex video application such as H.264/AVC codec in HW/SW codesign context. In fact, The TQ/IQT component has been integrated as an IP core into a SoPC platform for improving the system performances. We have estimated a 35 time improvement in coding speed at 140 MHz compared to the all software implementation with same image quality. The performances of our SoPC platform may be improved with another FPGA platform having higher operating frequency or by design ASIC circuit.

References:

[1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITU-T Rec. H.264 and ISO/IEC 14496-10 AVC,2003.

[2] I. E. G. Richardson, “H.264 and MPEG 4 Video Compression-Video Coding for Next Generation Multimedia”, New York: Wiley, 2003.

[3] *Video Coding for Low Bit Rate Communication*, ITU-T Recommendation H.263, Feb. 1998.

[4] *Information Technology—Coding of Audio-Visual Objects—Part 2: Visual*, ISO/IEC 14496-2, 1999.

[5] A. Joch, F. Kossentini, H. Schwarz, T. Wiegand, and G. J. Sullivan, “Performance Comparison of Video Coding Standards Using Lagragian Coder Control,” in *Proc. IEEE Int. Conf. Image Processing (ICIP’02)*, 2002, pp. 501–504.

[6] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, “Overview of the H.264/AVC Video Coding Standard”, IEEE Trans. On Circuits and Systems for Video Technology vol. 13, no. 7, pp. 560–576, July 2003.

[7] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, “Video Coding with H.264/AVC; Tools, Performance, and Complexity,” *IEEE Circuits Syst. Mag.*, vol. 4, no. 1, pp. 7–28, 1Q, 2004.

[8] A. Puri, X. Chen, and A. Luthra, “Video Coding using the H.264/MPEG-4 AVC Compression Standard,” in *Signal Process.:*

Image Commun., Oct. 2004, vol. 19, no. 9, pp. 793–849.

[9] L. Ling-zhi, Q. Lin, R. Meng-tian, J. Li, "A 2-D Forward/Inverse Integer Transform Processor of H.264 Based on Highly-Parallel Architecture", in Proc IEEE IWSOC'04, pp. 158-161, July 2004.

[10] Y. Li, Y. He and S. MEI, "A Highly Parallel Joint VLSI Architecture for Transforms in H.264/AVC", Journal of Signal Processing Systems, vol.50, PP. 19-32, January 2008.

[11] N. Keshaveni, S. Ramachandran and K. S. Gurumurthy, "Design and Implementation of Integer Transform and Quantization Processor for H.264 Encoder on FPGA", in Proc IEEE ACT'09, pp. 646-649, December 2009.

[12] R. Kordasiewicz and S. Shirani, "On Hardware Implementation of DCT and Quantization Blocks for H.264/AVC", Journal of VLSI Signal Processing, vol.47, pp. 189-199, 2007.

[13] O. Tasdizen and I. Hamzaoglu, "A High Performance and Low Cost Hardware Architecture for Transform and Quantization Algorithm" in Proc EUSIPCO'05, September 2005, Turkey.

[14] J. Johnston, N. Jayant, and R. Safranek, "Signal Compression Based on Models of Human Perception", Proc. IEEE, vol. 81, pp. 1385-1422, Oct. 1993.

[15] C. P. Fan, "Fast 2-Dimensional 4x4 Forward Integer Transform Implementation for H.264/AVC", IEEE Trans. On Circuits and Systems, vol.53, no.3, pp. 174-177, March 2006.

[16] TC. Y. Lu, K. A. Wen, "On the Design of Selective Coefficient DCT Module", IEEE Trans. On Circuits and Systems Video Technology, vol. 8, pp. 143–146, Dec. 2002.

[17] H.S. Malvar, A. Hallapuro, M. Karczewicz, L. Kerofsky, "Low-complexity Transform and Quantization in H.264/AVC", IEEE Trans. On Circuits and Systems Video Technology, vol. 13, pp. 598–603, July. 2003.

[18] Y. L. Lee, K. H. Han, D. G. Sim, and J. Seo, "Adaptive Scanning for H.264/AVC Intra Coding" ETRI Journal, vol. 28, no. 5, October 2006.

[19] Stratix II device, http://www.altera.com/

[20] Artisan Components. TSMC 0.18µm 1.8-Volt SAGE-X^{TM} Standard Cell Library Databook, 2001.

[21] R. Tessier and W. Burleson, "Reconfigurable Computing for Digital Signal Processing: a Survey," Journal of VLSI Signal Processing 28, 7-27, 2001

[22] Microblaze Integrated Development Environment http://www.xilinx.com/technology/embedded.htm

[23] Nios II Integrated Development Environment http://www.altera.com/products/ip/processors/nios2/ni2-index.html

[24] A. Ben Atitallah, P. Kadionik, F. Ghozzi, P. Nouel, N. Masmoudi, H. Levi "An FPGA Implementation of HW/SW Codesign Architecture for H.263 Video Coding", AEU - International Journal of Electronics and Communications, vol. 61, Issue 9, 1 October 2007, Pages 605-620.

[25] JVT H.264 Reference Software Version JM10.1, http://iphome.hhi.de/suehring/tml/download/old_jm/

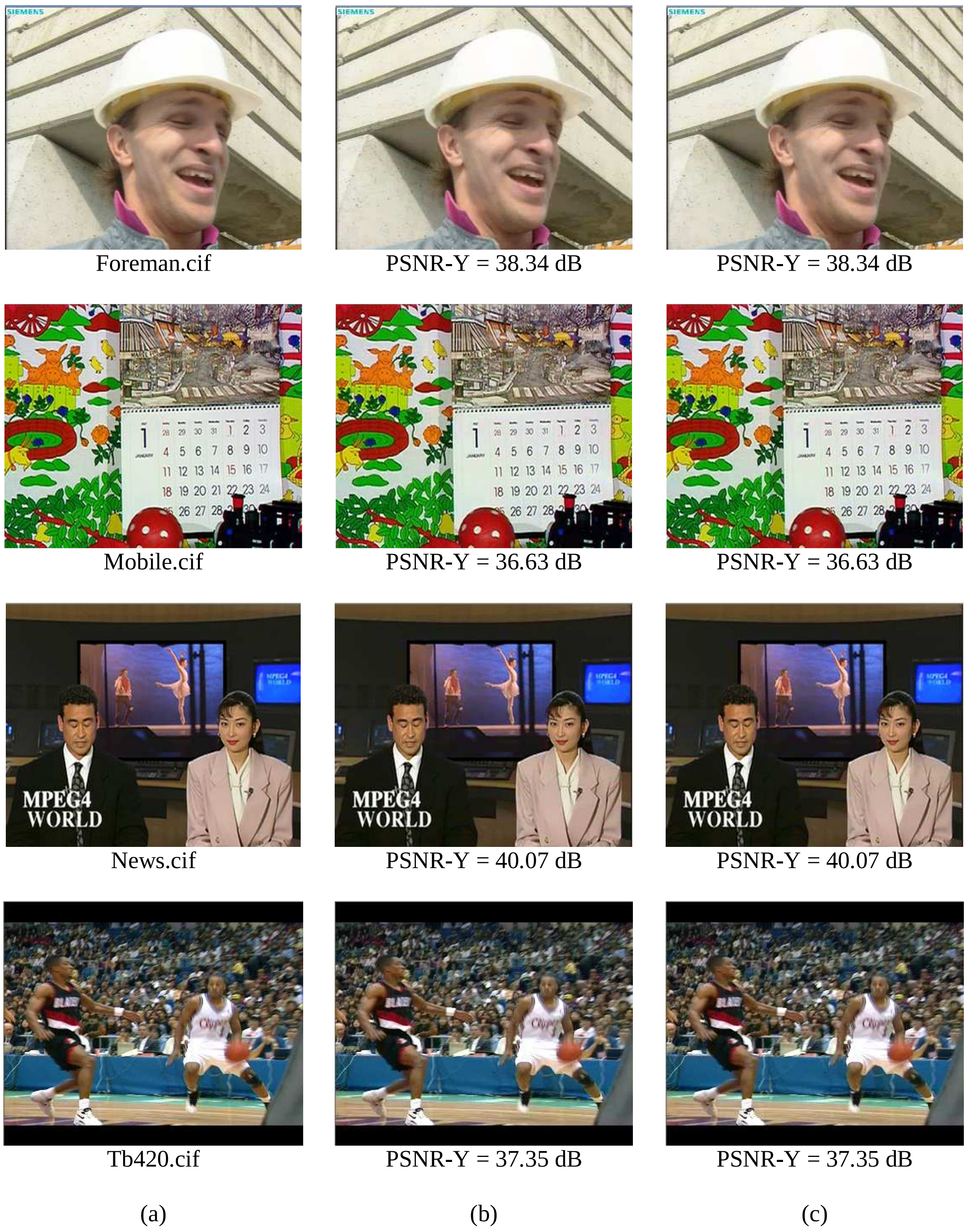

Fig. 15 (a) Original, (b) Reconstructed from SW and (c) Reconstructed from HW/SW of the 12th frame of the test video sequences