High Performance Unified Architecture for Forward and Inverse Quantization in H.264/AVC

Tiago Dias[§‡†], Luís Rosário[§‡], Nuno Roma[§‡] and Leonel Sousa[§‡]
[§]*INESC-ID Lisbon* / [‡]*IST-TU Lisbon* / [†]*ISEL-PI Lisbon*
Rua Alves Redol, 9
1000-029 Lisboa, Portugal
{*Tiago.Dias, Nuno.Roma, Leonel.Sousa*}*@inesc-id.pt, lrosario@sips.inesc-id.pt*

***Abstract*—A new high-performance and reduced hardware architecture for the computation of the H.264/AVC forward and inverse quantization operations is presented in this paper. This architecture is based on a highly flexible processing structure that is suitable for very efficient implementations using both FPGA and ASIC technologies. Moreover, it offers several different configurations, in order to provide different trade-offs in terms of performance and hardware cost. Experimental results concerning implementations using a Xilinx Virtex-5 FPGA and a 90 nm CMOS process from UMC demonstrated that the proposed architecture can be used to compute, in real-time, the forward and inverse quantization operations for videos with resolutions up to the Digital Cinema format (4096 × 2048 @ 30fps).**

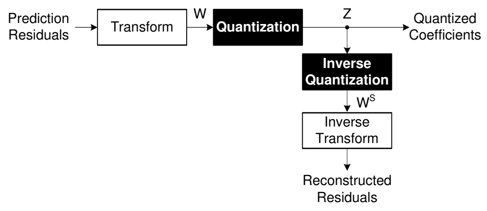
***Keywords*-Video coding; H.264/AVC; Quantization; Unified architecture; FPGA; ASIC.**

I. Introduction

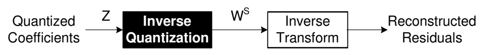
H.264/AVC [1] is currently considered the *de-facto* standard for modern multimedia applications based on digital video. Such achievement is the consequence of its extraordinary flexibility to allow efficient implementations in several different and distinct application domains, as well as from its higher compression efficiency levels.

Likewise any other block-based motion compensated transform coding scheme, compression in H.264/AVC is mostly achieved by lossy quantization. Such process is implemented by two distinct modules in the encoder loop (the forward quantizer and the inverse quantizer), and by the inverse quantizer module present at the decoder structure (see Figure 1). However, in order to guarantee the desired high coding efficiency levels, this standard adopts significantly more complex quantization algorithms. Such augment in complexity is mainly owed to the inclusion of multiplication operations by rational numbers in the quantization process, as well as to the several memory accesses that it involves [2]. In addition, the very tight interconnection between the transform and quantization operations further increases these complexity requirements [1], [2].

Altogether, this poses several difficult challenges to the design of H.264/AVC video codecs aiming at the processing of real-time and high definition video content, since quantization significantly influences the performance of such systems in terms of throughput and latency. To address this problem, several different solutions have been proposed,



(a) Encoder.



(b) Decoder.

Figure 1. Block diagrams of the lossy coding part of video codecs.

which can be classified as single, unified or integrated quantization architectures, as described below.

Single quantization architectures consist of independent processing structures that only compute either the forward or the inverse quantization algorithms. Typically, they combine generic multipliers and Look-Up Tables (LUTs) to directly implement the desired H.264/AVC quantization function. However, a couple of alternative implementations have also been proposed that simplify the quantization scheme, in order to reduce the complexity of its implementation in hardware. In [3] and [4] the multiplication operation was rather simplified by modifying the quantization parameter so as to use smaller bit-width adders. Although this greatly reduces the complexity of the multiplication operation, it also introduces a mismatch error between the encoder and the decoder. Other more recent proposals of forward quantization circuits [5] use time-multiplexed Multiple Constant Multipliers to reduce the implementation area and improve the performance of the quantization process. Nonetheless, most known proposals of quantization circuits for H.264/AVC are still based on generic multipliers.

Kordasiewicz [6] proposed both speed and area optimized structures for the computation of the forward quantization operation. However, the offered processing rates are more suitable for low and medium performance encoders. For high performance coding systems, parallel realizations of the H.264/AVC quantizer with 4, 8, 16 and 32 fundamental elements have also been proposed [7], [8], [9], [10], [11]. Among these proposals, the architectures presented by Lin [7] and Husemann [10] evidence some improvements in what concerns the offered hardware efficiency, since they share some of the hardware resources between all the quantizers. Moreover, as a result of these simplifications, these structures can also be used to implement high performance inverse quantizers.

Korah [12] proposed a different type of design based on dedicated pipelined add-and-shift multipliers that is also suitable to implement forward and inverse high-performance quantizers, but with a more reduced hardware cost. Similarly, Lee [13] also presented a processing structure for the computation of the two H.264/AVC quantization algorithms. However, such circuit consists of an unified architecture. This means that the desired quantization operation is not defined at design time. Instead, it is specified in run-time and as needed by the control unit of the video coding system.

Finally, integrated transform-quantization architectures were also proposed in [14] and [15], but using two quite different approaches. The architecture proposed by Tasdizen [14] combines in a single pipelined hardware structure a unified architecture for quantization and a transform computation circuit. Conversely, the highly specialized and parallel processing structure presented in [15] makes use of the same hardware resources to compute the transform and the forward quantization operations, which greatly increases its hardware efficiency.

In this paper, a new unified architecture for the computation of the H.264/AVC quantization operations is presented. Unlike other similar structures, the proposed architecture uses very few hardware resources and can be easily configured in run-time to implement either the forward or the inverse quantization algorithm. Moreover, its highly flexible structure also supports several different designs that allow to obtain implementations with distinct performance *vs* hardware cost trade-offs, and thus to be used in multiple applications with distinct performance requirements. In fact, the application scenarios of the proposed computational circuits range from the implementation, using both ASIC and FPGA technologies, of hardware accelerators in modern System-on-Chips (SoCs) to specialized functional units of Application Specific Instruction-Set Processors (ASIPs). In addition, the proposed architecture can also be integrated with other existing processing structures for the computation of the H.264/AVC transforms, in order to develop integrated transform-and-quantization specialized processors that can include either a single or multiple instances of the proposed architecture when parallel quantizers are required.

The rest of this paper is organized as follows. In section II the H.264/AVC forward and inverse quantization operations are analyzed. The proposed configurable architecture for implementing the forward and inverse quantization algorithms is presented in section III, together with its corresponding supporting mathematical model. Section IV describes the most relevant implementation details of such processing structure and discusses the experimental results that were obtained with its implementation using FPGA and ASIC technologies. Finally, section V concludes the presentation.

II. The H.264/AVC Quantization Process

The H.264/AVC quantization operation is based on an improved scalar quantizer that significantly differs from the ones implemented in previous ITU-T and MPEG video standards.

On the one hand, because it consists of a non-linear function supporting 52 distinct $QuantizationSteps(Qsteps)$ of rational values. Such values, which are specified as a function of a Quantization Parameter (QP), are in the range between 0.625 and 224 and increase by $\sqrt[6]{2}$ (i.e., 12%) for each increment of QP. Moreover, any value of the $Qstep$ function can be derived from its first 6 values [2]. As a result, a wide range of quality levels can be efficiently addressed, since fine control is possible at low quantization and coarse quantization is not burdened.

On the other hand, the very strong inter-dependencies that H.264/AVC introduced between the transform and the quantization procedures also significantly influence the quantizer. More specifically, the complexity of the H.264/AVC quantization function was greatly increased due to the incorporation of the Scaling Factors (SFs) that were left over from the H.264/AVC integer transform path [2].

By following the above considerations, Equation 1 represents the H.264/AVC quantization procedure, where W_{ij} is the transform coefficient (see Figure 1(a)), Z_{ij} is the resulting quantized coefficient, and i and j are the line and column indexes for the considered blocks of coefficients, respectively. SF_{ij} and $Qstep(QP)$ are the applicable scaling factor and quantization step, respectively.

$$Z_{ij} = round\left(W_{ij}\frac{SF_{ij}}{Qstep(QP)}\right) \tag{1}$$

As it can be seen, despite its simplicity this formulation considers two quite complex operations: a multiplication and a division involving rational numbers. Hence, the alternative representation based on integer arithmetic operations shown in Equation 2 is usually adopted for Equation 1, so as to simplify the computational complexity of such operation.

$$Z_{ij} = \left(W_{ij} \times MF(QP)_{ij} + f \times 2^{h}\right) \times \frac{1}{2^{15+\lfloor\frac{QP}{6}\rfloor+h}} \tag{2}$$

This simpler formulation merges into a single non-linear function (MF) all the possible combinations for the ratio between SF and $Qstep$, therefore allowing to speed up the computation procedure and to significantly reduce the memory requirements. In fact, the MF function consists only of 6×3 different 14-bit positive integer constant values, as it can be seen from Equation 3 and Table I.

$$MF(QP)_{ij} = \begin{bmatrix} m(QP,0) & m(QP,2) & m(QP,0) & m(QP,2) \\ m(QP,2) & m(QP,1) & m(QP,2) & m(QP,1) \\ m(QP,0) & m(QP,2) & m(QP,0) & m(QP,2) \\ m(QP,2) & m(QP,1) & m(QP,2) & m(QP,1) \end{bmatrix} \tag{3}$$

In addition, the f and h terms in Equation 2 provide finer control of the quantization procedure near the origin

Table I
DEFINITION OF THE VALUES FOR $m(QP, n)$.

QP%6	**0**	**1**	**2**	**3**	**4**	**5**
n=**0**	13107	11916	10082	9362	8192	7282
n=**1**	5243	4660	4194	3647	3355	2893
n=**2**	8066	7490	6554	5825	5243	4559

("the dead zone") for all types of macroblock (INTRA and INTER) and transforms (i.e., 4×4 Hadamard and 2×2 Hadamard for the DC coefficients and the DCT for the AC coefficients), as it is shown in Equation 4 and Equation 5.

$$f = \begin{cases} \frac{2}{3}^{\lfloor \frac{QP}{6} \rfloor} & ,if\ INTRA\ block \\ \frac{2}{6}^{\lfloor \frac{QP}{6} \rfloor} & ,otherwise \end{cases} \quad (4)$$

$$h = \begin{cases} 1 & ,if\ H_{4\times4} \bigvee H_{2\times2} \\ 0 & ,otherwise \end{cases} \quad (5)$$

In what concerns the inverse quantization operation, the algorithm considered in the H.264/AVC standard also reflects the tighter coupling with the inverse transform process and, obviously, the improved $Qstep$ values. Hence, similarly to quantization, the complexity of the H.264/AVC inverse quantization algorithm is also higher than in previous video standards, and mostly for the same reasons.

As it can be seen in Equation 6, which formulates this inverse quantization operation, in H.264/AVC the computation of the reconstructed coefficients (W^S) corresponding to the quantized terms (Z) requires, fundamentally, two multiplications involving rational numbers: the $Qstep$ and the Pre-scaling Factors (PFs) left over from the inverse transform procedure [2] (see Figure 1(b)). The multiplication by the constant value 64, which is used only to improve the accuracy in the computation of the inverse transforms, does not contribute to such complexity augment, since it can be easily computed by a shift-left operation.

$$W^S_{ij} = Z_{ij} \times Qstep(QP) \times PF_{ij} \times 64 \quad (6)$$

Equation 1 and Equation 6 evidence the quite similar characteristics of the operands involved in the computation of the forward and inverse quantization operations, respectively. As a result, all the considerations mentioned above focusing on the optimization of the quantization operation can also be applied to Equation 6, in order to obtain a less complex representation of the inverse quantization function.

Such alternative formulation is shown in Equation 7, which is also based on integer arithmetic operations and makes use of a non-linear function (V) to compute all the possible combinations of $PF \times Qstep$. In fact, V has a definition that is identical to the one presented in Equation 3 for MF, but involving the 5-bit width positive integer values presented in Table II. The values of α and τ are shown in Equation 8 and Equation 9, respectively.

$$W^S_{ij} = (Z_{ij} \times V(QP)_{ij} + \alpha) \times 2^{\lfloor \frac{QP}{6} \rfloor - \tau} \quad (7)$$

$$\alpha = \begin{cases} 2^{1-\lfloor \frac{QP}{6} \rfloor} & ,if\ H_{4\times4} \bigwedge (QP < 12) \\ 0 & ,otherwise \end{cases} \quad (8)$$

Table II
DEFINITION OF THE VALUES FOR $\nu(QP, n)$.

QP%6	**0**	**1**	**2**	**3**	**4**	**5**
n=**0**	10	11	13	14	16	18
n=**1**	16	18	20	23	25	29
n=**2**	13	14	16	18	20	23

$$\tau = \begin{cases} 2 & ,if\ H_{4\times4} \\ 1 & ,if\ H_{2\times2} \\ 0 & ,otherwise \end{cases} \quad (9)$$

This generic formulation for the inverse quantization algorithm addresses the reconstruction of both the AC and the DC coefficients for all types of macroblock, since it not only maximizes the dynamic range for all transform types, but it also takes into consideration the differences in the quantization step sizes for luma and chroma blocks.

III. UNIFIED QUANTIZATION ARCHITECTURE

The very tight coupling of the H.264/AVC transform and quantization algorithms (see Figure 1) has for long motivated the proposal of hardware structures combining the two operations. Most of such processing structures have been specifically designed to efficiently implement the operations either of the coding or of the decoding path of the video codec. However, most of the designs that are nowadays being proposed to implement the H.264/AVC transform modules consist of unified architectures that are able to compute all the transform functions defined in the standard. Consequently, the design of modern hardware structures combining the transform and quantization functionalities also requires the development of efficient, fast and unified architectures for the computation of the forward and inverse quantization operations.

In the following subsections, the mathematical model of one of such processing structures is formulated and its corresponding hardware design is presented. Moreover, specific optimizations targeting efficient implementations using distinct technologies are also discussed.

A. Mathematical Model

A careful analysis of the formulations of the H.264/AVC forward and inverse quantization operations shown in Equation 2 and Equation 7, respectively, reveals that their corresponding algorithms and involved operands are quite similar. In fact, both expressions require a multiplication by a quantization parameter (σ), include a dead-zone control value (φ) and involve a scaling factor (ε). Hence, the two operations can be represented using a generic expression:

$$O_{ij} = [S_{ij} \times \sigma(QP)_{ij} + \varphi] \times 2^{\varepsilon} \quad (10)$$

where S_{ij} can be either the transform or the quantized coefficient of line i and column j of the block of coefficients that is being forward or inverse quantized, respectively.

In what concerns its complexity, this new formulation can be optimized by taking into consideration the quite specific properties of the involved operands. In particular, by recalling that all operands are either positive or negative integer values, which provides the means required to realize all the computations in integer arithmetic. As a result of this observation, the following simplifications can be applied in the implementation of the two algorithms.

For the quantization (see Equation 2), the dead-zone control parameter can be computed as shown in Equation 11, where $<<$ is a logical shift left operation, $>>$ denotes an arithmetic shift right operation and β and h are given by Equation 12 and Equation 5, respectively.

$$\begin{aligned} \varphi_Q &= f \times 2^h \\ &= \left(\frac{2}{3}^{\lfloor \frac{QP}{6} \rfloor} \times 2^{-\beta}\right) \times 2^h \\ &= \left(\frac{2}{3}^{\lfloor \frac{QP}{6} \rfloor} >> \beta\right) << h \end{aligned} \quad (11)$$

$$\beta = \begin{cases} 0 & ,if\ INTRA\ block \\ 1 & ,otherwise \end{cases} \quad (12)$$

In addition, the multiplication involving ε can also be realized with an arithmetic shift right operation, owing to the fact that this scaling factor always takes negative integer values, as it is shown in Equation 13.

$$\varepsilon_Q = -\left(15 + \left\lfloor \frac{QP}{6} \right\rfloor + h\right) \quad (13)$$

As a result of these simplifications, Equation 10 can be rewritten for the quantization operation as shown in Equation 14, where the quantization parameter σ_Q consists on function MF presented in Equation 3.

$$O_{Q_{ij}} = \left[S_{ij} \times \sigma_Q(QP)_{ij} + \varphi_Q\right] >> |\varepsilon_Q| \quad (14)$$

Regarding to the inverse quantization operation, the computation of the dead-zone control parameter can be greatly simplified by carefully analyzing Equation 8, which reveals that φ_{IQ} can only take the three values shown in Equation 15.

$$\varphi_{IQ} = \begin{cases} 0 & ,if\ H_{4\times4} \bigwedge (QP \geq 12) \\ 1 & ,if\ H_{4\times4} \bigwedge (QP \geq 6 \bigwedge QP < 12) \\ 2 & ,otherwise \end{cases} \quad (15)$$

In what concerns the scaling factor, ϵ_{IQ} can take both positive and negative integer values, as it is shown in Equation 16. Consequently, the multiplication involving ϵ_{IQ} can be implemented by an arithmetic shift right or logical shift left operation, depending on the value of QP and on the inverse transform function that are being considered.

$$\epsilon_{IQ} = \begin{cases} -\left(2 - \lfloor \frac{QP}{6} \rfloor\right) & ,if\ c_1 \\ -\left(1 - \lfloor \frac{QP}{6} \rfloor\right) & ,if\ c_2 \\ \lfloor \frac{QP}{6} \rfloor & ,otherwise \end{cases} \quad (16)$$

$$\begin{aligned} c_1 &: H_{4\times4} \bigwedge QP < 12 \\ c_2 &: H_{2\times2} \bigwedge QP < 6 \end{aligned}$$

By considering all the above simplifications, Equation 17 presents the integer arithmetic representation of Equation 10 for the inverse quantization operation, where the quantization parameter σ_{IQ} consists on function V.

$$O_{ij} = \begin{cases} \left[S_{ij} \times \sigma_{IQ}(QP)_{ij} + \varphi_{IQ}\right] >> |\epsilon_{IQ}| & ,if\ c_1 \bigvee c_2 \\ \left[S_{ij} \times \sigma_{IQ}(QP)_{ij} + \varphi_{IQ}\right] << |\epsilon_{IQ}| & ,otherwise \end{cases} \quad (17)$$

B. Proposed Architecture

The architecture herein proposed to realize the H.264/AVC forward and inverse quantization operations simultaneously implements the functionalities represented in Equation 14 and Equation 17 (or more generally, in Equation 10), but exclusively using integer arithmetic computational circuits. As a result, such processing structure mostly consist of a very simple and yet efficient integer datapath with four different processing phases and only three distinct computation circuits (a 16 $\times$ 15-bits signed multiplier, a 31-bit adder and a 32-bit barrel-shifter), which are shared for the implementation of the two quantization operations. Nonetheless, the proposed architecture also includes other less complex logical elements, as it can be seen in Figure 2. Such logic is required not only to support the combined forward/inverse functionalities, but also for the computation of some intermediate values, such as a 4-bit adder to obtain the value of ϵ and ROM devices to provide all the required constant values (i.e., f, MF, V, $QP\%6$, etc). Table III describes the functionality of block η, which allows using the same adder to compute ϵ_Q (Equation 13) and ϵ_{IQ} (Equation 16).

Figure 2 also evidences the very flexible structure of the proposed architecture, being capable of supporting multiple configurations with distinct hardware cost/performance/latency characteristics, in order to optimally address the requirements of any given application. Such configurations consist of non-pipelined and pipelined versions of the proposed architecture, as shown in Table IV.

The configuration with the most reduced hardware cost is also the one providing the lowest latency and consists of the non-pipelined version of the architecture, where all the pipeline registers (represented with black circles in Figure 2) are replaced by direct point-to-point circuit interconnections. On the other hand, the highest performance levels are obtained with the design implementing a fully pipelined architecture with the four stages A, B, C and D, as defined

Table III
FUNCTIONALITY OF THE BLOCK η.

Opcode	**T_TYPE**	$QP < 12$	$QP < 6$	η
0	-	-	-	1
1	00	-	0	0
1	00	-	1	1
1	01	0	-	0
1	01	1	-	1
1	10	-	-	0
1	11	-	-	0

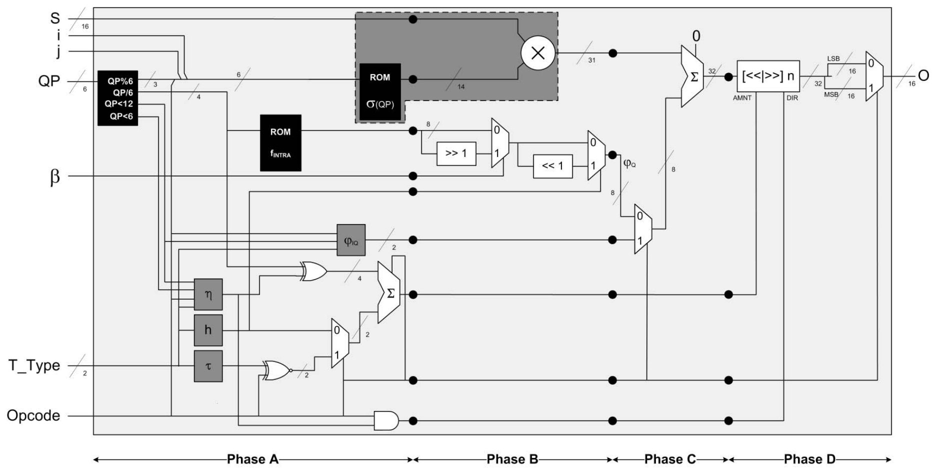


Figure 2. Proposed unified configurable architecture for the computation of the H.264/AVC quantization operations.

Table IV
CONFIGURATIONS OF THE PROPOSED ARCHITECTURE.

Architecture Configuration	Pipeline Registers		
	A/B	B/C	C/D
Non-pipelined	-	-	-
2 pipeline stages	-	√	-
3 pipeline stages	√	√	-
4 pipeline stages	√	√	√

in the bottom of Figure 2. The remaining configurations are also pipelined versions of the architecture, but using fewer pipeline stages so as to guarantee the different trade-offs between hardware cost, performance and latency. For all these configurations, the obtained throughput is always one quantized/scaled transform coefficient per clock cycle. Moreover, the processing rate is also maximized in each configuration, as a result of all the efforts that were devised to keep the pipeline stages properly balanced.

In what concerns the functionality of the proposed architecture, phase A is used to fetch from the ROMs all the constant coefficients depending on QP, as well as the scaling factors (MF and V). The amount and direction of the shift for the final adjustment of the processed values are also computed in this phase, together with the rounding factors (φ) for the forward and inverse quantization operations. Nonetheless, the final value to be used in the rounding operation is only definitely computed in the multiplier in phase B, in order to keep all the processing phases as balanced as possible. Consequently, the rounding operation is realized in phase C, using the scaled data value that is selected in phase B. In phase D, the final values of the quantized/scaled transform coefficients are adjusted using the barrel shifter and the correct 16-bit value of either the quantized or scaled transform coefficient is provided at the output port of the architecture.

C. Optimized Multiplier for ASIC Implementations

The multiply operation that is required by the forward and inverse quantization algorithms can be regarded as a Multiple-Constant Multiplier (MCM) problem, because it consists in the computation of a product involving a transform coefficient and a constant that can take one out of several different values (see Tables I and II). This approach is highly advantageous when considering ASIC implementations of the proposed architecture, since it allows avoiding the usage of hardware costly generic binary multiplication circuits to compute the required product values.

Although different approaches are available to implement MCMs, the time-multiplexed Multiple Constant Multiplier (mux-MCM) method was adopted [16] to design the specialized multiplier unit for implementations of the proposed architecture in ASIC. Unlike parallel MCM solutions, this approach provides a multiplier structure containing several multiplexers that are switched by control logic to compute the products for different constants. This allows to greatly increase the circuit's hardware efficiency for applications that only compute one product at a time, just like the H.264/AVC quantization operations.

In this work a mux-MCM circuit was developed to implement the multiplier of the proposed architecture. The Directed Acyclic Graph of such specilized arithmetic unit is based on [16] and was generated using the SPIRAL project framework (`www.spiral.net`).

IV. EXPERIMENTAL RESULTS

To demonstrate the advantages offered by the proposed architecture in terms of performance and hardware cost, as well as to validate its functionality, this dedicated processing structure was described using IEEE-VHDL and synthesized for both FPGA and ASIC implementations.

A single description of the circuit presented in Figure 2 was developed to implement the four different configurations listed in Table IV, for which `generic` type parameterization inputs were used to specify the design to be implemented. Such description was carried out by using a quite generic VHDL coding style, in order to achieve efficient implementations when using the two considered technologies. Nonetheless, a special attention was given to the description of the most performance critical blocks of the proposed architecture (i.e., the multiplier, the adders and the ROMs), in order to assist the synthesis tools in inferring the most efficient primitives for its implementation, according to the chosen synthesis strategy and implementation constraints.

A. FPGA Implementation

In what concerns to the conducted FPGA implementations, such design effort was especially relevant for the adopted FPGA device (Virtex5 XC5VFX70TFFG1136) and synthesis tool (`xst`, from Xilinx Design Suite 13.2i), since it allowed to use the `DSP48E` slice to efficiently implement the multiplication operation. The impact of these optimizations in terms of hardware cost and performance is demonstrated in Table V, which presents the most relevant implementation results obtained for the adopted performance oriented synthesis procedure.

Regarding to the occupied hardware resources, the results shown in Table V demonstrate the rather insignificant requirements of the proposed architecture for any of its four designs, despite the offered flexibility in terms of functionality. In fact, a single `DSP48E` slice and less than 40 ordinary Virtex-5 slices (about 1% of the total capacity of the adopted FPGA device) are used in the implementation of the most hardware costly (and also faster) configuration of this processing structure. The maximum allowed clock frequencies presented in Table V demonstrate the high processing rates that are offered by the proposed architecture, i.e., between 126 and 311 Mcoefs/s. These differences in the attained performance are not only owed to the application of the multi-stage pipeline technique, but also to a better usage of the `DSP48E` slice in the case of pipelined configurations. In such slightly more hardware costly configurations, the `DSP48E` macrocell implements a very optimized and fast multiply-and-accumulate unit, required in processing phases B and C, where the flip-flops of the pipeline register are pushed into the `DSP48E` slice, as a result of the synthesis tool being capable to successfully infer the multiply-and-accumulate coding pattern. Hence, it can be concluded that the four configurations of the proposed architecture effectively allow to trade-off latency for performance in FPGA implementations. Figure 3 shows the upper bound limits for the performance offered by the four configurations when operating with the clock frequencies presented in Table V. According to these results, it can be concluded that the proposed processing structure is able to comply with the real-time requirements of video codecs up to the Digital Cinema format (4096×2048 @ 30fps).

B. ASIC Implementation

A very similar IEEE-VHDL description of the proposed architecture was also used to implement in ASIC the four configurations presented in Table IV. The only difference between such description and the one used for the FPGA implementations concerns strictly to the multiplier, which for the ASIC implementations consisted of a structural description of the mux-MCM unit described in section III-C.

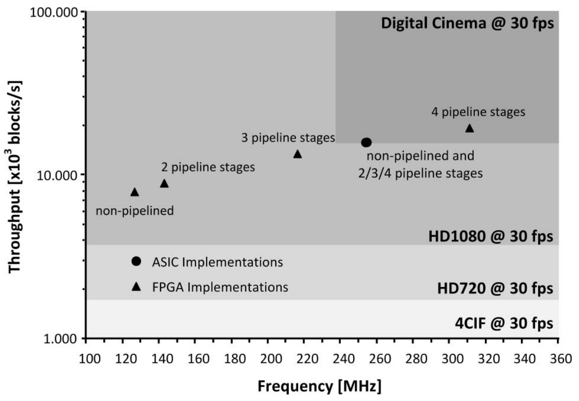


Figure 3. Performance comparison of several configurations of the proposed architecture.

Table VI presents the implementation results obtained with the synthesis of the four configurations using a standard cell library based on a 90 nm CMOS process from UMC [17], and by considering the typical operating conditions (Vdd=1.2V, T=25°C). Likewise the FPGA implementations, a performance oriented synthesis procedure was also adopted. However, specific timing constraints were used to guide the synthesis tool, in order to obtain the most hardware efficient designs capable of complying with the desired performance requirements, i.e., support for the HDTV 1080p video format.

The results shown in Table VI evidence that the hardware resources required to implement each of the four designs are quite small and also very similar. In fact, the minor differences that can be observed concern only to the resources required to implement the several pipeline registers. This is a direct consequence of the timing constraints that were imposed to the synthesis tool, which generated a very similar multiplication circuit for all the four configurations of the architecture. As it can be seen in Table VI and Table VII, the adopted mux-MCM occupies about 2/3 of the circuits implementation areas and restricts their maximum operating frequencies. Nonetheless, such clock frequencies are still compliant with the requirements of the HDTV 1080p video format, as it is shown in Figure 3. Moreover, the reduced amount of hardware required by the considered mux-MCM circuit fully justifies their usage, since they provide a significant reduction of the silicon area when compared with a

Table V
FPGA IMPLEMENTATION RESULTS.

Configuration	Slices	FFs	LUTs	DSP48Es	Max. F.
Non-pipelined	193	0	193	1	126.5 MHz
2 pipeline stages	195	6	194	1	142.8 MHz
3 pipeline stages	208	20	191	1	216.3 MHz
4 pipeline stages	214	26	192	1	311.0 MHz

Table VI
ASIC IMPLEMENTATION RESULTS.

Configuration	Area	Gate Count	Max. F.
Non-pipelined	0.030 mm^2	5942 kgates	253.9 MHz
2 pipeline stages	0.027 mm^2	5308 kgates	254.4 MHz
3 pipeline stages	0.028 mm^2	5410 kgates	253.8 MHz
4 pipeline stages	0.029 mm^2	5610 kgates	254.5 MHz

Table VII
COMPARISON OF THE MULTIPLICATION CIRCUITS.

Architecture	Area	Gate Count	Max. F.
Parallel MCM with 3 levels	$0.033\ mm^2$	6377 kgates	365.0 MHz
Parallel MCM with 4 levels	$0.032\ mm^2$	6318 kgates	325.7 MHz
Parallel MCM with 5 levels	$0.031\ mm^2$	6110 kgates	282.5 MHz
mux-MCM	$0.019\ mm^2$	3712 kgates	254.4 MHz

binary multiplier alternative solution, without compromising the attained performance levels (see Table VII). Hence, it can be concluded that the best compromise of the proposed architecture when implemented in an ASIC corresponds to the non-pipelined configuration, owing to its smaller latency and to the almost identical hardware cost/efficiency and performance levels offered by the four designs.

C. Discussion

The advantages offered by the proposed architecture in terms of performance and hardware cost were assessed by comparing it with the most prominent related designs. Table VIII presents the results of such comparative analysis for the subset of designs that were reviewed. These designs not only present different functionalities but were also implemented using distinct technologies, i.e., FPGA and ASIC. As a result, a more comprehensive figure of merit was adopted to conduct this comparison: the Data Throughput per Unit of Area (DTUA). The DTUA is defined as the ratio of the data throughput rate (in coefficients per second), over the hardware cost (in terms of unit of area). In ASIC implementations, the number of gates was chosen to represent the unit of area, while in FPGA implementations it was considered a slice as a unit of area.

As it can be seen in Table VIII, the proposed architecture is one of the most hardware efficient structures for both FPGA and ASIC implementations, despite being the only one that is capable of computing the forward and the inverse quantization operations. In fact, the proposed structure outperforms all the other designs, even when its lower performance configuration (non-pipelined) is used. The only exception is the set of structures presented in [10] that achieve higher DTUAs. However, it is important to observe that this set of designs only implements a single quantization operation and presents latency values equal to the 3-stages and the 2-stages pipelined configurations of the proposed architecture. Consequently, it can be concluded that the quantizer proposed in [10] has a much lower DTUA value than the 3-stages pipelined configuration of the proposed architecture. Still, the DTUA value of the inverse quantizer presented in [10] is relatively higher than that of the 2-stages pipelined configuration of the proposed architecture, owing to its highly parallel structure that shares many of its hardware resources. Nevertheless, an entirely similar approach could equally be adopted to design parallel quantizers based on the proposed base architecture, since many of its hardware resources can also be shared by multiple instances of the architecture (i.e., all the circuits except the ROM, the multiplier, the 32-bit adder and the barrel-shifter). In such cases, it is expected that these designs would present even higher DTUA values, similar to the ones provided by the inverse quantizer in [10].

Table VIII also evidences that the reduced hardware characteristic of the proposed architecture does not compromises its performance regarding the other alternative solutions. In fact, the proposed unified quantization circuit not only presents the highest DTUA values, but it is also capable of complying with the requirements of real-time operation up to the Digital Cinema format (4096×2048 @ 30fps) for implementations in FPGA and in ASIC. Nevertheless, higher performance levels targeting superior video formats can also be achieved by exploiting the scalable nature of the proposed architecture, in order to combine several of its instances in a single design, as it was stated before. For example, the throughput of a circuit using four instances of the proposed base architecture can be as high as 1244 MCoefs/s, which is fully compliant with the requirements for real-time processing of the Ultra High Definition Video (UHDV) format (4320×7680 @ 30fps).

From Table VIII it is also possible to conclude that the proposed architecture allows to adjust the hardware cost / performance trade-off in a different dimension. In fact, by alternating its operation (forward and inverse quantization) along the time, a codec that includes a single instance of this processing structure is able to compute both quantization operations and still saves over 70% in hardware cost, when compared to a codec that uses two independent and dedicated functional units to realize the same operations. This is especially relevant when such video codecs are implemented in ASIC, since the final cost of the corresponding ICs is directly and significantly affected by the amount of required hardware resources. Such codec design, using only one instance of the proposed architecture, is most suitable for the implementation of reduced complexity and low cost video coding systems with moderate requirements in terms of performance. Conversely, higher performance video coding systems will use the proposed architecture, by considering a more hardware costly structure that integrates two instances of the proposed architecture to compute the forward and inverse quantization operations in parallel.

V. CONCLUSION

An innovative unified architecture for the computation of the H.264/AVC quantization operations was proposed. This high performance dedicated processing structure is characterized by a quite flexible architecture, exclusively based on integer arithmetic computational circuits, which are shared for the computation of both the forward and inverse quantization algorithms. As a result, it presents several advantages in what concerns to hardware efficiency and cost, which is quite diminished. Conversely, its highly flexible nature also provides significant advantages in terms of performance, since the most critical computational elements can be not only efficiently implemented using different technologies (e.g., mux-MCMs for ASICs or optimized fabric slices for FPGAs), but also instantiated using several

Table VIII
COMPARISON WITH OTHER RELATED ARCHITECTURES.

Design	Quantization Function	Technology	Area	Max. F. [MHz]	Latency [CC]	Throughput [Coefs/CC]	DTUA ($\times 10^3$)
[6] (area)	Forward	Virtex2 Pro	143 slices	135	4	0.25	236.3
[6] (speed)	Forward	Virtex2 Pro	192 slices	97	1	1.00	97.9
[10]	Forward	Virtex2 Pro	483 slices	108	3	4.00	892.4
[10]	Inverse	Virtex2 Pro	222 slices	136	2	4.00	2457.0
Proposed (no pipeline)	Unified	Virtex5	193 slices	127	1	1.00	655.5
Proposed (2 stages)	Unified	Virtex5	195 slices	143	2	1.00	732.5
Proposed (3 stages)	Unified	Virtex5	208 slices	216	3	1.00	1039.7
Proposed (4 stages)	Unified	Virtex5	214 slices	311	4	1.00	1453.5
[5]	Forward	SMIC 180 nm	28.56 kgates	250	4	4.00	35.0
[6] (area)	Forward	TSMC 180 nm	1.75 kgates	85	4	0.25	12.2
[6] (speed)	Forward	TSMC 180 nm	39.90 kgates	68	1	1.00	1.7
[11]	Forward	Tower 180 nm	104.05 kgates	76	1	32.00	23.4
[11]	Inverse	Tower 180 nm	76.11 kgates	76	1	32.00	32.0
Proposed (no pipeline)	Unified	UMC 90 nm	5.94 kgates	254	1	1.00	42.7

different configurations (i.e., non-pipelined and 4-stage fully pipelined circuits). The experimental results obtained for implementations in FPGA and ASIC have proved the above observations and showed that the proposed architecture is capable of processing, in real-time, video sequences up to the Digital Cinema format (4096×2048 @ 30fps).

ACKNOWLEDGMENT

This work was supported by the Portuguese Foundation for Science and for Technology (INESC-ID multiannual funding) through the PIDDAC Program funds and under project HELIX: Heterogeneous Multi-Core Architecture for Biological Sequence Analysis (PTDC/EEA-ELC/113999/2009), and by the PROTEC Program funds under the research grant SFRH / PROTEC / 50152 / 2009.

REFERENCES

[1] T. Wiegand, G. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[2] I. E. Richardson, *The H.264 Advanced Video Compression Standard*, 2nd ed. John Wiley & Sons, Ltd, 2010.

[3] Y. Zhang, G. Jiang, and M. Yu, "Low-complexity quantization for H.264/AVC," *J. Real-Time Image Processing*, vol. 4, no. 1, pp. 3–12, 2009.

[4] M. Michael and K. Hsu, "A low-power design of quantization for H.264 video coding standard," in *2008 IEEE Int. SOC Conf.*, Sep. 2008, pp. 201–204.

[5] J. Ying, X. Chen, Y. Fan, and X. Zeng, "MUX-MCM based quantization VLSI architecture for H.264/AVC high profile encoder," in *IEEE/IFIP 19th Int. Conf. VLSI and Syst.-on-Chip*, Oct. 2011, pp. 72–77.

[6] R. Kordasiewicz and S. Shirani, "ASIC and FPGA implementations of H.264 DCT and quantization blocks," in *IEEE Int. Conf. Image Processing, 2005*, vol. 3, Sep. 2005, pp. 1020–1023.

[7] H.-Y. Lin, Y.-C. Chao, C.-H. Chen, B.-D. Liu, and J.-F. Yang, "Combined 2-D transform and quantization architectures for H.264 video coders," in *IEEE Int. Symp. Circuits and Syst.s, 2005*, vol. 2, May 2005, pp. 1802–1805.

[8] M. Owaida, M. Koziri, I. Katsavounidis, and G. Stamoulis, "A high performance and low power hardware architecture for the transform & quantization stages in H.264," in *IEEE Int. Conf. Multimedia and Expo, 2009*, Jul. 2009, pp. 1102–1105.

[9] M. Elhaji, A. Zitouni, S. Meftali, J.-l. Dekeyser, and R. Tourki, "A low power and highly parallel implementation of the H.264 8x8 transform and quantization," in *2010 IEEE Int. Symp. Signal Processing and Information Technol.*, Dec. 2010, pp. 528–531.

[10] R. Husemann, M. Majolo, V. Guimaraes, A. Susin, V. Roesler, and J. Lima, "Hardware integrated quantization solution for improvement of computational H.264 encoder module," in *18th IEEE/IFIP VLSI Syst. on Chip Conf.*, Sep. 2010, pp. 316–321.

[11] G. Pastuszak, "Transforms and quantization in the high-throughput H.264/AVC encoder based on advanced mode selection," in *2008 IEEE Computer Society Annual Symp. on VLSI*, Apr. 2008, pp. 203–208.

[12] R. Korah and J. Perinbam, "FPGA implementation of integer transform and quantizer for H.264 encoder," *Journal of Signal Processing Syst.s*, vol. 53, pp. 261–269, 2008.

[13] S. Lee and K. Cho, "Implementation of an AMBA-compliant IP for H.264 transform and quantization," in *2006 IEEE Asia Pacific Conf. Circuits and Syst.s*, Dec. 2006, pp. 1071–1074.

[14] O. Tasdizen and I. Hamzaoglu, "A high performance and low cost hardware architecture for H.264 transform and quantization algorithms," in *13th European Signal Processing Conf.*, 2005, pp. 4–8.

[15] S. Lee and K. Cho, "Design of high-performance transform and quantization circuit for unified video CODEC," in *2008 IEEE Asia Pacific Conf. Circuits and Syst.s*, Dec. 2008, pp. 1450–1453.

[16] P. Tummeltshammer, J. C. Hoe, and M. Püschel, "Time-multiplexed multiple constant multiplication," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 9, pp. 1551–1563, 2007.

[17] *Faraday ASIC Cell Library FSD0A_A 90nm Standard Cell*, Faraday Technology Corporation, Feb. 2009.