# Reconfigurable Unified Architecture for Forward and Inverse Quantization in H.264/AVC

Tiago Dias[†ǂ§], Nuno Roma[†ǂ], Leonel Sousa[†ǂ]
[†]*INESC–ID Lisbon,* [ǂ]*IST–TU Lisbon,* [§]*ISEL–PI Lisbon*
*Rua Alves Redol, 9*
*1000-029 Lisbon, Portugal*
{*Tiago.Dias, Nuno.Roma, Leonel.Sousa*}*@inesc-id.pt*

## Abstract

*A new reconfigurable architecture for unified computation of the H.264/AVC quantization operations is presented in this paper. This processing structure presents a highly flexible architecture involving very few hardware resources and can be configured to provide different trade-offs in terms of performance and hardware cost. Experimental results obtained using a Xilinx Virtex-5 FPGA demonstrated that the proposed architecture is capable of computing the forward and inverse quantization for Digital Cinema applications in real-time (*$4096 \times 1714$*@24 fps).*

## 1. Introduction

In the last few years, there has been an evident growth of multimedia applications based on digital technologies, such as IPTV, 3GPP mobile multimedia telephony, HDTV video cameras or video surveillance. At the same time, the digital convergence design trend concerning the development of new electronic devices was also significantly reinforced. As a result, all the new digital products that are nowadays being developed present several multimedia capabilities, mostly focusing on video coding and streaming, despite its fundamental functionality (e.g., cell phones with digital cameras or MP3 players with video playback capabilities). Therefore, video codecs have become mandatory and key components of these systems, which depending on the type of the applications being supported can include a single or more codecs to address the multiple existing video standards.

Currently, the vast majority of such multimedia devices implement, at least, the H.264/AVC standard [1] for three different reasons. Firstly, to increase the interoperability, as a result of the widespread usage of the H.264/AVC standard. Secondly, because of the extraordinary flexibility that it allows so as to be efficiently used in several distinct classes of applications, mainly through the use of multiple combinations of coding profiles and levels [2]. Lastly, and also very importantly, due to the offered high compression efficiency levels, which provide a reduction of at least 50% in the bit-rate when compared with MPEG-2 and up to 30% better compression levels when compared to H.263+ and MPEG-4 for the same visual quality of the encoded video [3]. This high coding efficiency results not only from the new coding tools introduced by H.264/AVC, but also from the more complex and computational intensive algorithms that they involve. Nonetheless, H.264/AVC still implements the classical block-based motion compensated transform coding scheme, depicted in Figure 1, where compression is mostly achieved in the quantization module.

As a consequence, quantization significantly influences both the efficiency and the performance of a video codec. In particular, high coding efficiency can only be achieved provided that effective quantization algorithms are adopted, in order to maximize the trade-off between compression and visual distortion. However, the complexity of such algorithms is also relatively high, which can greatly affect the performance of the codec in terms of throughput and latency. In fact, this is the case of H.264/AVC, whose quantization algorithms are of relatively high complexity due to involving multiplications by rational numbers and several memory handling operations [4]. Moreover, the very tight interconnection of the transform and quantization blocks in H.264/AVC codecs further increases such complexity requirements [5]. Altogether, this poses several difficult challenges to the design of digital systems aiming at the processing of high definition video content, as well as to the development of real-time encoders and decoders, as it can be seen in Table 1. To address this problem, several different solutions have been proposed in the literature that can be classified as single, unified or integrated quantization architectures, as described below.

Single quantization architectures consist of independent processing structures that compute either the forward or the inverse quantization algorithms. Typically, they include a multiplier, an adder and a barrel-shifter, which are used to directly implement the desired H.264/AVC quantization function. Nonetheless, several alternative implementations

| Video Standard | Image Resolution | Frame Rate | Throughput |
|---|---|---|---|
| CIF | $352 \times 288$ pixels | 15 fps | 1.6 Mpixels/s |
| 4CIF | $704 \times 576$ pixels | 30 fps | 12.2 Mpixels/s |
| SDTV 576p | $720 \times 576$ pixels | 30 fps | 12.4 Mpixels/s |
| HDTV 720p | $1280 \times 720$ pixels | 30 fps | 27.7 Mpixels/s |
| HDTV 1080p | $1920 \times 1080$ pixels | 30 fps | 62.2 Mpixels/s |
| Digital Cinema 4k | $4096 \times 1714$ pixels | 24 fps | 168.5 Mpixels/s |
| UHDTV | $7680 \times 4320$ pixels | 60 fps | 1990.7 Mpixels/s |

Table 1. Processing rate requirements of the H.264/AVC quantization block for real-time operation.

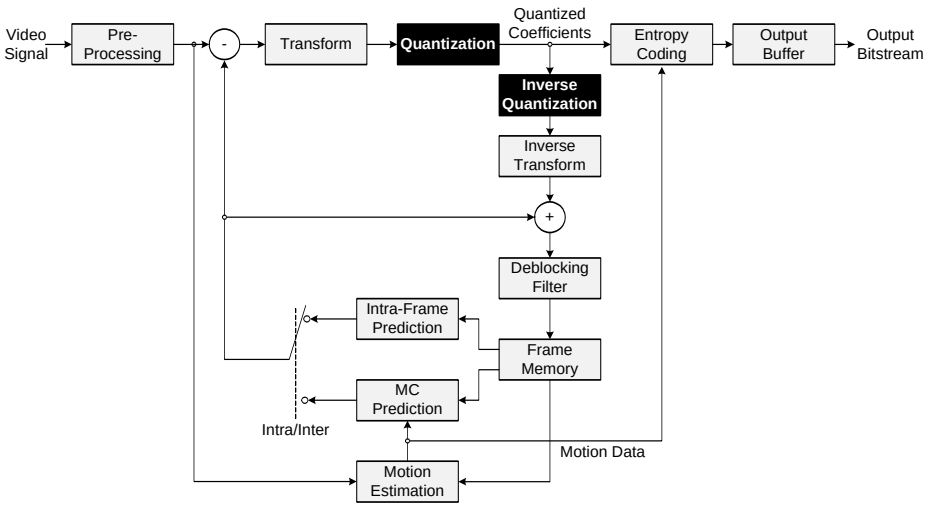


Figure 1. Generic block diagram of a H.264/AVC encoder.

of this elementary computational element have been presented. Kordasiewicz [6] proposed both speed and area optimized structures for the computation of the forward quantization operation, whose 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 and 16 fundamental elements have also been proposed [7, 8, 9, 10]. Among these proposals, the architectures proposed by Lin [7] and Husemann [10] present 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 [11] 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 [12] 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, but instead is specified in run-time and as needed by the control unit of the video coding system. Finally, an integrated transform-quantization architecture was also proposed in [13]. This highly specialized and parallel processing structure makes use of the same hardware resources to compute the transform and the forward quantization operations, which greatly increases its hardware efficiency. Nevertheless, the offered performance levels are only compatible with the HD1080p format ($1920 \times 1080$ pixels).

In this paper, a new unified architecture for the computation of the H.264/AVC quantization operations is presented. Unlike the architectures previously discussed, the proposed structure uses very few hardware resources and can be easily reconfigured in run-time to implement either the forward or the inverse quantization algorithm. Moreover, its highly flexible structure also supports several different configurations 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 for such 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 [14], in order to develop integrated transform-and-quantization specialized processors that can include a single or multiple instances of the proposed architecture when parallel quantizers are demanded.

The rest of this paper is organized as follows. In section 2 the H.264/AVC forward and inverse quantization processes are analyzed. The proposed reconfigurable architecture for implementing the forward and inverse quantization algorithms is presented in section 3, together with its corresponding supporting mathematical model. Section 4 describes the most relevant implementation details of the proposed specialized processing structure and discusses the experimental results that were obtained with its implementation in a Xilinx Virtex5 FPGA device. Finally, section 5 concludes the presentation.

## 2. The H.264/AVC Quantization Process

The H.264/AVC quantization operation is based on an improved scalar quantizer that was designed to allow video encoders not only to maximize the trade-off between bitrate and image quality, but also to more accurately manage it. Consequently, the quantization function implemented by this quantizer 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 Quantization Steps (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 using Equa-

tion 1. 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 [5].

$$Qstep(QP) = Qstep(QP\%6) \times 2^{\lfloor \frac{QP}{6} \rfloor} \quad (1)$$

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 operation of 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 improved H.264/AVC transform path [4]. Nonetheless, the benefits to the whole coding algorithm that resulted from realizing the transform coding operations in integer arithmetic greatly compensate such penalty.

Following the above considerations, Equation 2 represents the H.264/AVC quantization procedure, where $Z_{ij}$ is a quantized coefficient, $W_{ij}$ is the corresponding transform coefficient, and $i$ and $j$ are the line and column indexes for the considered blocks of coefficients, respectively.

$$Z_{ij} = round\left(W_{ij}\frac{SF_{ij}}{Qstep(QP)}\right) \quad (2)$$

As it can be seen, despite its simplicity this formulation considers two quite complex operations: a multiplication and a division involving rational numbers. Nonetheless, an alternative representation based on integer arithmetic operations can also be adopted for the H.264/AVC quantization algorithm, as it is presented in [15]. Such formulation is shown in Equation 3 and replaces the division by an integer multiplication and an arithmetic shift right operation, which greatly simplifies the computational complexity of the quantization operation.

$$Z_{ij} = round\left[\left(W_{ij} \times MF(QP)_{ij} + f\right) \times \frac{1}{2^{15+\lfloor \frac{QP}{6} \rfloor}}\right] \quad (3)$$

In addition, the merge into a single function ($MF$) of all the possible combinations for the ratio between SF and $Qstep$ allows not only to speed up the computation procedure, but also to significantly reduce its memory requirements. In fact, a Look-Up Table (LUT) implementation of such non-linear function $MF$ consists only of $6 \times 3$ different pre-computed 14-bit positive integer values, as it can be seen from Equation 4 and Table 2.

$$MF(QP)_{ij} = \begin{bmatrix} m(QP,0) & m(QP,2) & m(QP,0) & m(QP,2) \\ m(QP,2) & m(QP,1) & m(QP,2) & m(QP,1) \\ m(QP,0) & m(QP,2) & m(QP,0) & m(QP,2) \\ m(QP,2) & m(QP,1) & m(QP,2) & m(QP,1) \end{bmatrix} \quad (4)$$

Finally, the formula presented in Equation 3 also includes a parameter $f$ to provide a finer control of the quantization procedure near the origin ("the dead zone") for all types of macroblock, as it is shown in Equation 5.

$$f = \begin{cases} \frac{2}{3}\lfloor \frac{QP}{6} \rfloor & ,if\ INTRA\ block \\ \frac{2}{6}\lfloor \frac{QP}{6} \rfloor & ,otherwise \end{cases} \quad (5)$$

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

Table 2. Definition of the values for $m(QP,n)$.

However, such formula is only suitable to quantize the transform coefficients computed in the first level of the H.264/AVC hierarchical transform path [5], i.e., the AC coefficients resulting from the application of the 2D integer Discrete Cosine Transform (DCT) function to the $4 \times 4$ blocks of transform coefficients. For the remaining coefficients of the macroblock, which consist of all the luma and chroma DC coefficients computed using either a $4 \times 4$ Hadamard ($H_{4\times4}$) or a $2 \times 2$ Hadamard ($H_{2\times2}$) transform, the dead-zone control parameter $f$ has to be adjusted, in order to compensate their smaller dynamic range. Such correction factor is shown in Equation 6 that presents a general formulation for the H.264/AVC quantization algorithm, valid for the quantization of the AC ($h=0$) and the DC ($h=1$) coefficients of both INTRA and INTER macroblocks.

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

In what concerns the inverse quantization operation, the algorithm considered in H.264/AVC also reflects the tighter coupling with the inverse transform process and, obviously, the improved $Qstep$ values. Hence, similarly to quantization, the complexity of the inverse quantization algorithm is also higher than in previous video standards, and mostly for the same reasons. As it can be seen in Equation 7, which formulates this inverse quantization operation, in H.264/AVC the reconstruction of the quantized data ($Z$) requires, fundamentally, two multiplications involving rational numbers: the $Qstep$ and the Pre-scaling Factors (PFs) left over from the inverse transform procedure [4]. 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.

$$W_{ij}^S = Z_{ij} \times Qstep(QP) \times PF_{ij} \times 64 \quad (7)$$

Equation 7 evidences the quite similar characteristics of the operands involved in the computation of the quantization (see Equation 2) and inverse quantization algorithms. Consequently, all the considerations mentioned above focusing on the optimization of the quantization operation can also be applied to Equation 7, in order to simplify its computation. In particular, all the possible combinations of the term $Qstep(QP) \times PF_{ij}$ can be precomputed and mapped into a single LUT, while the multiplications can be

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

Table 3. Definition of the values for $v(QP,n)$.

realized in integer arithmetic (some can even be converted into shift-left operations). Equation 8 shows this alternative formulation, where the LUT $V$ representing the product of $Qstep(QP)$ by $PF_{ij}$ ($v(QP,n)$) has an organization identical to the one presented in Equation 4 for $MF$, but involving the 5-bit width positive integer values presented in Table 3.

$$W_{ij}^{S} = Z_{ij} \times V(QP)_{ij} \times 2^{\left\lfloor \frac{QP}{6} \right\rfloor} \tag{8}$$

Furthermore, to extend this new formulation in order to address the reconstruction of both the AC and the DC coefficients for all types of macroblock it is only required that the dynamic range is maximized for all transform types and that the differences in the quantization step sizes for luma and chroma blocks are processed accordingly. This more generic formula for the inverse quantization algorithm is presented in Equation 9, where the values of $\alpha$ and $\tau$ are shown in Equation 10 and Equation 11, respectively.

$$W_{ij}^{S} = \left(Z_{ij} \times V(QP)_{ij} + \alpha\right) \times 2^{\left\lfloor \frac{QP}{6} \right\rfloor - \tau} \tag{9}$$

$$\alpha = \begin{cases} 2^{1-\left\lfloor \frac{QP}{6} \right\rfloor} & ,if\ inverse\ H_{4\times4} \bigwedge (QP < 12) \\ 0 & ,otherwise \end{cases} \tag{10}$$

$$\tau = \begin{cases} 2 & ,if\ inverse\ H_{4\times4} \\ 1 & ,if\ inverse\ H_{2\times2} \\ 0 & ,otherwise \end{cases} \tag{11}$$

## 3. Unified Quantization Architecture

As it was shown in section 1, video encoders are required to implement the forward and the inverse quantization operations. On the contrary, video decoders are only required to reconstruct the transform coefficients (i.e., realize inverse quantization). Nevertheless, Equation 6 and Equation 9 evidence that the same combination of arithmetic operators (i.e., integer arithmetic adders and multipliers) is used to compute the two operations, whose operands also present very similar characteristics.

The previous observation motivates the development of a single and fast hardware processing structure for the computation of both the forward and inverse quantization operations. This allows not only to significantly increase the efficiency of such dedicated functional unit, in terms of hardware cost and performance, but also its usability. In the following subsections, the mathematical model behind such processing structure is provided and the corresponding hardware implementation is presented.

### 3.1. Mathematical Model

According to Equation 6 and Equation 9, the comprehensive formulations of the H.264/AVC forward and inverse quantization algorithms are quite similar. In fact, both expressions involve a multiplication by a quantization parameter ($\sigma$), include a dead-zone control value ($\varphi$) and involve a scaling factor ($\varepsilon$). Hence, a more generic expression can be used to jointly represent the two operations:

$$O_{ij} = \left(S_{ij} \times \sigma(QP)_{ij} + \varphi\right) \times 2^{\varepsilon} \tag{12}$$

where $S_{ij}$ can be either the transform or 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 still 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 quantization, the dead-zone control parameter can be computed as shown in Equation 13, where $<<$ is a logical shift left operation, $>>$ denotes an arithmetic shift right operation and $\beta$ and $h$ are given by Equation 14 and Equation 15, respectively.

$$\begin{aligned} \varphi_Q &= f \times 2^{h} \\ &= \left(\frac{2}{3}^{\left\lfloor \frac{QP}{6} \right\rfloor} \times 2^{-\beta}\right) \times 2^{h} \\ &= \left(\frac{2}{3}^{\left\lfloor \frac{QP}{6} \right\rfloor} >> \beta\right) << h \end{aligned} \tag{13}$$

$$\beta = \begin{cases} 0 & ,if\ INTRA\ block \\ 1 & ,otherwise \end{cases} \tag{14}$$

$$h = \begin{cases} 1 & ,if\ H_{4\times4} \bigvee H_{2\times2} \\ 0 & ,otherwise \end{cases} \tag{15}$$

In addition, the multiplication involving $\varepsilon$ can also be realized with an arithmetic shift right operation owing to the fact that this scaling factor always takes negative integer values, as shown in Equation 16.

$$\varepsilon_Q = -\left(15 + \left\lfloor \frac{QP}{6} \right\rfloor + h\right) \tag{16}$$

As a result of these simplifications, Equation 12 can be rewritten for the quantization operation as

$$O_{Q_{ij}} = \left(S_{ij} \times \sigma_Q(QP)_{ij} + \varphi_Q\right) >> |\varepsilon_Q| \tag{17}$$

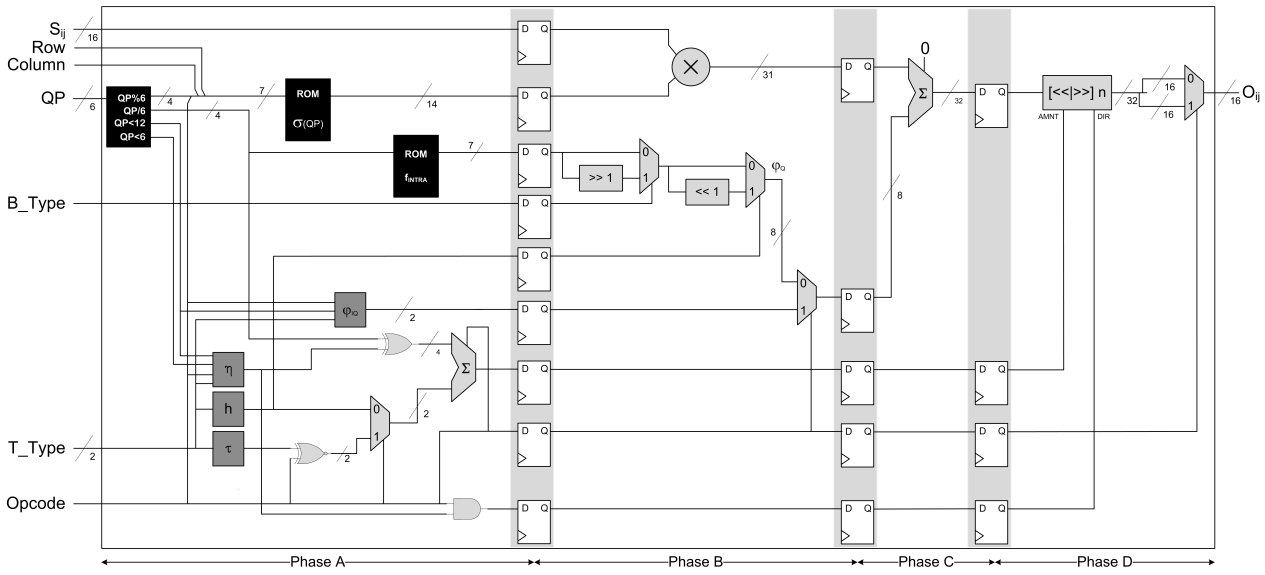


Figure 2. Proposed architecture for the reconfigurable forward/inverse quantization unit.

where the quantization parameter $\sigma_Q$ consists on function $MF$ presented in Equation 4.

Regarding to the inverse quantization operation, the computation of the dead-zone control parameter can be greatly simplified by carefully analyzing Equation 10, which reveals that $\varphi_{IQ}$ can only take the three values shown in Equation 18.

$$\varphi_{IQ} = \begin{cases} 0 & ,if\ inverse\ H_{4\times4} \bigwedge (QP \geq 12) \\ 1 & ,if\ inverse\ H_{4\times4} \bigwedge (QP \geq 6 \bigwedge QP < 12) \\ 2 & ,otherwise \end{cases} \quad (18)$$

In what concerns the scaling factor, it can either take positive or negative integer values, as it is shown in Equation 19. Consequently, the multiplication involving $\varepsilon_{IQ}$ can be implemented by an arithmetic shift right or logical shift left operation, depending on the QP value and inverse transform function that are being considered.

$$\varepsilon_{IQ} = \begin{cases} -\left(2 - \left\lfloor \frac{QP}{6} \right\rfloor\right) & ,if\ c_1 \\ -\left(1 - \left\lfloor \frac{QP}{6} \right\rfloor\right) & ,if\ c_2 \\ \left\lfloor \frac{QP}{6} \right\rfloor & ,otherwise \end{cases} \quad (19)$$

$c_1$ : $inverse\ H_{4\times4} \bigwedge QP < 12$
$c_2$ : $inverse\ H_{2\times2} \bigwedge QP < 6$

By following all the above simplifications, Equation 20 presents the integer arithmetic representation of Equation 12 for the inverse quantization operation, where the quantization parameter $\sigma_{IQ}$ consists on function $V$.

$$O_{ij} = \begin{cases} \left(S_{ij} \times \sigma_{IQ}(QP)_{ij} + \varphi_{IQ}\right) >> \left|\varepsilon_{IQ}\right| & ,if\ c_1 \bigvee c_2 \\ \left(S_{ij} \times \sigma_{IQ}(QP)_{ij} + \varphi_{IQ}\right) << \left|\varepsilon_{IQ}\right| & ,otherwise \end{cases} \quad (20)$$

### 3.2. Architecture Implementation

The proposed generic formulation for the forward and inverse quantization algorithms, defined in Equation 17 and Equation 20, presents several advantages regarding its implementation using dedicated hardware circuits, both in terms of performance and hardware cost/efficiency.

Firstly, it enables the development of an unified circuit to implement the two operations. Secondly, it only involves integer operands, which allows realizing all the calculations using exclusively integer arithmetic computational circuits. Thirdly, such implementations require the usage of solely two different arithmetic operators, i.e., an integer multiplier and an integer adder. Finally, because it can be shown that the computation of a quantized/scaled transform coefficient may be represented as a four stage process involving very few operations. Such property is especially relevant for the development of efficient processing structures that support multiple configurations with distinct hardware cost/performance/latency characteristics, in order to optimally address the requirements of any given application. Nonetheless, this unified formulation presents a minor and unavoidable drawback for applications that require a dedicated circuit for the computation of only one of the two operations, i.e., the extra hardware cost. Still, this is a small price to pay for the offered increased flexibility in what concerns the circuit's functionality.

The architecture herein proposed to realize the forward and inverse quantization operations jointly implements the functionalities represented in Equation 17 and Equation 20, by taking into consideration all the above observations. As it can be seen from Figure 2, which depicts the block diagram of the proposed H.264/AVC Forward/Inverse Quantization Unit (FIQU), this processing structure presents a regular architecture with four different processing phases, where the most complex and performance critical computation circuits are shared for the implementation of the for-

| Signal | Description |
|---|---|
| B_Type | Type of the block of coefficients ($\beta$): 0-INTRA; 1-INTER |
| Opcode | Quantization operation: 0-Forward; 1-Inverse |
| QP | Quantization parameter |
| T_Type | Transform used to code the block of coefficients: 00-$H_{2\times2}$; 01-$H_{4\times4}$;10-$DCT_{4\times4}$ |

Table 4. Description of the FIQU control signals.

| Opcode | T_TYPE | $QP < 12$ | $QP < 6$ | $\eta$ |
|---|---|---|---|---|
| 0 | - | - | - | 1 |
| 1 | 00 | - | 0 | 0 |
| 1 | 00 | - | 1 | 1 |
| 1 | 01 | 0 | - | 0 |
| 1 | 01 | 1 | - | 1 |
| 1 | 10 | - | - | 0 |
| 1 | 11 | - | - | 0 |

Table 5. Functionality of the block $\eta$.

ward and inverse quantization operations, i.e., the multiplier, the two adders and the barrel shifter. Moreover, the FIQU uses a single ROM to store the quantization parameters for the two operations ($\sigma_Q$ and $\sigma_{IQ}$), which not only reduces the complexity of the hardware implementation but also improves its performance. Table 4 describes the relationship between the control signals depicted in Figure 2 and the variables in Equation 17 and Equation 20, while Table 5 describes the functionality of block $\eta$ that allows to use the same adder to compute $\varepsilon_Q$ (Equation 16) and $\varepsilon_{IQ}$ (Equation 19).

Figure 2 also evidences that the FIQU has a very flexible structure that supports the four distinct configurations listed in Table 6, therefore allowing the designer to trade-off performance for hardware resources. More specifically, the configuration with the most reduced hardware cost is also the one providing the lowest latency and consists of a non-pipelined version of the FIQU architecture, where all the pipeline registers are replaced by direct point-to-point circuit interconnections. On the other hand, the highest performance levels of the FIQU are obtained with the design implementing a fully pipelined architecture with four stages. The remaining configurations are also pipelined versions of the FIQU architecture but using fewer pipeline stages, in order 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, its 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 FIQU, phase *A* is used to fetch the scaling factors (*MF* and *V*) from the ROMs, as well as all the data depending on *QP*. 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 ($\varphi$) for the forward and inverse quantization operations. Nonetheless, the final value to be used in the rounding operation is only definitely computed 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*. Phase *D* is used to adjust the final values of the quantized/scaled transform coefficients using the barrel shifter and to provide the correct 16-bit value of either the quantized or scaled transform coefficient at the output port of the FIQU.

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

Table 6. Configurations of the proposed architecture.

## 4. Implementation and Experimental Results

To demonstrate the functionality of the FIQU and to assess both its hardware requirements and offered performance levels, the architecture of this dedicated processing structure was described using the IEEE-VHDL hardware description language and synthesized for a general purpose Virtex5 XC5VFX70TFFG1136 FPGA device.

A single behavioral description of the circuit presented in Figure 2 was developed to implement the four different configurations supported by the FIQU (see Table 6), for which `generic` type configuration inputs were used to specify the desired design. Such description was carried out by using a quite generic VHDL coding style, in order to allow efficient implementations of the proposed architecture using several different technologies (e.g., FPGA and ASIC). Nonetheless, a special attention was given to the description of the most performance critical blocks of the FIQU (i.e., the ROMs, the $16 \times 14$-bits multiplier and the 32-bits adder), in order to assist the synthesis tools in inferring the most efficient primitives for its implementation, according to the chosen synthesis strategy.

Such design effort was especially relevant for the considered FPGA device and synthesis tool (`xst` from Xilinx Design Suite 12.4i), since it allowed to make a good use of the `DSP48E` slice [16]. In particular, for the non-pipelined configurations of the FIQU, this slice was used to implement an efficient computation of the multiplication operation. Similarly, the `DSP48E` slice was also used in the remaining possible configurations of the FIQU, but to implement the multiply-and-accumulate operation required in the *B* and *C* processing phases. The impact of these optimizations in terms of hardware cost and performance is demonstrated in Table 7, which presents the most relevant implementation results obtained for the adopted timing performance oriented synthesis procedure.

In what concerns to hardware resources, the results shown in Table 7 demonstrate the rather insignificant requirements of the proposed architecture for any of its four designs, despite the offered flexibility in terms of function-

| Architecture | FFs | LUTs | DSP48Es | Max. F. |
|---|---|---|---|---|
| Non-pipelined | 0 | 195 | 1 | 138.4 MHz |
| 2 pipeline stages | 6 | 198 | 1 | 154.5 MHz |
| 3 pipeline stages | 38 | 203 | 1 | 236.7 MHz |
| 4 pipeline stages | 55 | 208 | 1 | 331.4 MHz |

Table 7. Implementation results of the proposed circuits in a Xilinx Virtex5 FPGA.

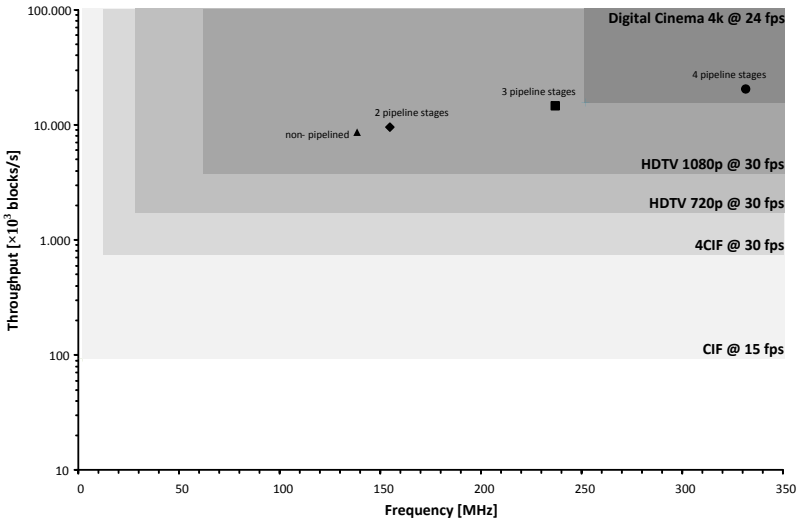


Figure 3. Performance comparison of the configurations of the proposed architecture.

ality. In fact, a single DSP48E slice and less than 60 ordinary Virtex-5 slices are used in the implementation of the most hardware costly, and also faster, configuration of the FIQU. Since these resources consist only of about 1% of the total capacity of the adopted FPGA device, which is considered to be of medium size, it is possible to conclude that multiple instances of the FIQU can be combined in parallel in a single implementation of a forward/inverse quantizer, thus increasing the offered performance level by allowing the processing of more than one coefficient at each clock cycle. Nonetheless, the maximum allowed clock frequencies presented in Table 7 already demonstrate the high processing rates that are offered by a single instance of the FIQU, i.e., about 330 Mega Operations Per Second (MOPS). 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 ($4096 \times 1714$ pixels). The upper bound limits for the performance of each configuration of FIQU are shown in detail in Figure 3.

As it can be seen in Figure 3, the non-pipelined configuration of the proposed architecture allows to process up to $8.6 \times 10^6$ blocks per second (HDTV 1080p format), while the throughput of the fastest pipelined design is over $20 \times 10^6$ blocks per second (Digital Cinema 4k format). These differences in 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 pipelined configurations of the FIQU. As it was previously mentioned, the DSP48E slice is exclusively used to implement a fast multiplier for the non-pipelined configurations, therefore avoiding a much slower LUT-based implementation. On the other hand, for pipelined FIQU configurations the DSP48E macrocell implements a very optimized and fast multiply-and-accumulate unit, since the synthesis tool pushed a pipeline register into the DSP48E slice as a result of having been assisted to infer the multiply-and-accumulate coding pattern. Similarly, the performance differences between the 2-stage and the 3/4-stage pipelined designs are also owed to the architecture of the DSP48E slice and the synthesis tool, which was able to reduce the setup time of the multiplier operands by pushing their corresponding pipeline registers into the DSP48E slice.

The obtained implementation results have also demonstrated that the four devised configurations for the FIQU architecture effectively allow to trade-off performance and latency for hardware cost. The data presented in Table 7 for the non-pipelined and pipelined designs demonstrates that the amount of hardware resources required by these processing structures scales with the increase of its maximum allowed clock frequencies. In fact, while the 2-stages pipelined implementation requires a couple more slices than the non-pipelined design, its performance levels are also higher (about 13%) than those offered by the less hardware costly configuration. The comparison between the other pipelined designs reveals a similar tendency, but with a higher relative increase in the performance gains (on average, about 45%).

Finally, it is worth mentioning that the usage of the FIQU architecture in a video codec also allows to trade-off hardware cost for performance in a different dimension. A codec that includes an instance of the FIQU is able to compute both the forward and the inverse quantization operations and still saves up to 20% in hardware cost, when compared to a codec that uses two independent and dedicated functional units to realize the same operations [17]. 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 hardware resources required for the design of these circuits. Nonetheless, a codec design using two dedicated functional units offers the added possibility of having the forward and inverse quantization modules working in parallel, while the one using the proposed FIQU can only process one of the two operations for a macroblock at a given time instant. Consequently, the macroblock processing rate of the codec using a single instance of the FIQU can be reduced in up to 50% regarding to the more hardware costly one. Still, the high performance levels offered by the FIQU allow to time share its use for the alternate computation of the quantiza-

tion and inverse quantization operations for the most typical application scenarios. It can therefore be concluded that the FIQU design is most suitable for the implementation of reduced complexity and cost video coding systems with moderate requirements in terms of performance. Nonetheless, high performance video coding systems can also make use of the proposed architecture, by using one instance of FIQU to realize quantization and another one to compute the inverse quantization operation.

## 5. Conclusion

An innovative reconfigurable architecture for an unified computation of the H.264/AVC forward and inverse quantization operations was proposed. This dedicated processing structure is characterized by a quite simple and flexible architecture, based exclusively on integer arithmetic computational circuits. Such circuits consist mostly of a $16 \times 14$-bit multiplier, a 32-bit adder and a 32-bit barrel-shifter, which are all used in the computation of both the forward and inverse quantization operations. As a consequence, the proposed architecture presents several advantages in what concerns to hardware efficiency and cost, which is quite reduced considering that two different quantization operations are supported. Conversely, the highly flexible nature of the presented processing structure also provides significant advantages in terms of performance, since its most critical computational elements can be interconnected in several different configurations (e.g., non-pipelined and 4-stage fully pipelined circuits). As a result, several different implementations with distinct performance *vs* hardware cost trade-offs can be obtained using the proposed architecture, which allows its usage in a much wider range of video coding systems (from low-end systems to high performance codecs). The experimental results obtained with the implementation of the proposed architecture in a Xilinx Virtex-5 FPGA device operating at 330 MHz demonstrated the above observations and revealed that it can process video sequences with resolutions up to $4096 \times 1714$ (Digital Cinema 4k format) in real-time (24 fps).

## Acknowledgments

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] Joint Video Team of ITU-T and ISO/IEC JTC1. *ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG4-AVC), "Advanced Video Coding for Generic Audiovisual Services"*. ITU-T, May 2003.

[2] T. Wiegand, G. Sullivan, G. Bjntegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7):560–576, July 2003.

[3] 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.*, 4(1):7–28, January 2004.

[4] I. E. G. Richardson. H.264/MPEG-4 Part 10: Transform & quantization, a white paper, March 2003. [Online. Available: http://www.vcodex.com].

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

[6] R.C. Kordasiewicz and S. Shirani. ASIC and FPGA implementations of H.264 DCT and quantization blocks. In *Image Processing, 2005 (ICIP 2005). IEEE International Conference on*, volume 3, pages 1020–1023, September 2005.

[7] Heng-Yao Lin, Yi-Chih Chao, Che-Hong Chen, Bin-Da Liu, and Jar-Ferr Yang. Combined 2-D transform and quantization architectures for H.264 video coders. In *Circuits and Systems, 2005 (ISCAS 2005). IEEE International Symposium on*, volume 2, pages 1802–1805, May 2005.

[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 *Multimedia and Expo, 2009 (ICME 2009). IEEE International Conference on*, pages 1102–1105, July 2009.

[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 *Signal Processing and Information Technology (ISSPIT 2010). 2010 IEEE International Symposium on*, pages 528–531, December 2010.

[10] R. Husemann, M. Majolo, V. Guimaraes, A. Susin, V. Roesler, and J.V. Lima. Hardware integrated quantization solution for improvement of computational H.264 encoder module. In *VLSI System on Chip Conference (VLSI-SoC 2010), 2010 18th IEEE/IFIP*, pages 316–321, September 2010.

[11] Reeba Korah and J. Perinbam. FPGA implementation of integer transform and quantizer for H.264 encoder. *Journal of Signal Processing Systems*, 53:261–269, 2008.

[12] Seonyoung Lee and Kyeongsoon Cho. Implementation of an AMBA-compliant IP for H.264 transform and quantization. In *Circuits and Systems, 2006 (APCCAS 2006). IEEE Asia Pacific Conference on*, pages 1071–1074, December 2006.

[13] Seonyoung Lee and Kyeongsoon Cho. Design of high-performance transform and quantization circuit for unified video CODEC. In *Circuits and Systems, 2008 (APCCAS 2008). IEEE Asia Pacific Conference on*, pages 1450–1453, December 2008.

[14] T. Dias, S. Lopez, N. Roma, and L. Sousa. A flexible architecture for the computation of direct and inverse transforms in H.264/AVC video codecs. *Consumer Electronics, IEEE Transactions on*, 57(2):936–944, May 2011.

[15] H.S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky. Low-complexity transform and quantization in H.264/AVC. *IEEE Trans. Circuits Syst. Video Technol.*, 13(7):598–603, July 2003.

[16] Xilinx, Inc. *Virtex-5 FPGA XtremeDSP Design Considerations - User Guide*, 3.4 edition, June 2010.

[17] T. Dias, N. Roma, and L. Sousa. Optimized forward/inverse quantization unit for H.264/AVC codecs. In *Conf. Electronics, Telecommunications and Computers (CETC 2011)*, pages CD–ROM, November 2011. ISBN: 978-989-97531-0-5.