Int J Parallel Prog (2013) 41:236–260
DOI 10.1007/s10766-012-0221-x

Scalable Unified Transform Architecture for Advanced Video Coding Embedded Systems

Tiago Dias · Sebastián López · Nuno Roma · Leonel Sousa

Received: 22 March 2012 / Accepted: 6 September 2012 / Published online: 2 October 2012

Abstract A novel high throughput and scalable unified architecture for the computation of the transform operations in video codecs for advanced standards is presented in this paper. This structure can be used as a hardware accelerator in modern embedded systems to efficiently compute all the two-dimensional 4×4 and 2×2 transforms of the H.264/AVC standard. Moreover, its highly flexible design and hardware efficiency allows it to be easily scaled in terms of performance and hardware cost to meet the specific requirements of any given video coding application. Experimental results obtained using a Xilinx Virtex-5 FPGA demonstrated the superior performance and hardware efficiency levels provided by the proposed structure, which presents a throughput per unit of area relatively higher than other similar recently published designs targeting the H.264/AVC standard. Such results also showed that, when integrated in a multi-core embedded system, this architecture provides speedup factors of about 120× concerning pure software implementations of the transform algorithms,

T. Dias (✉)
ISEL-PI Lisbon/INESC-ID Lisbon/IST-TU Lisbon, Rua Conselheiro Emídio Navarro 1,
1959-007 Lisbon, Portugal
e-mail: tdias@deetc.isel.ipl.pt

S. López
IUMA/University of Las Palmas GC, Campus Universitario de Tafira,
35017 Las Palmas de Gran Canaria, Spain
e-mail: seblopez@iuma.ulpgc.es

N. Roma · L. Sousa
INESC-ID Lisbon/IST-TU Lisbon, Rua Alves Redol 9, 1000-029 Lisbon, Portugal
e-mail: Nuno.Roma@inesc-id.pt

L. Sousa
e-mail: Leonel.Sousa@inesc-id.pt

therefore allowing the computation, in real-time, of all the above mentioned transforms for Ultra High Definition Video (UHDV) sequences (4, 320 × 7, 680 @ 30 fps).

Keywords Video coding · H.264/AVC · Unified transform · Scalable architecture · Systolic array · FPGA

1 Introduction

Modern embedded systems differ from their classical counterparts in several different characteristics, being the offered computational power and the target application areas two of the most relevant aspects. This shift of trend in the design of embedded systems mostly resulted from all the recent innovations in VLSI technology, which allowed designing processors with much higher performance levels. In addition, this was also motivated by all the new applications that have been proposed throughout the last few years, in order to cope with the ever increasing and more demanding requirements of modern users.

In this scope, multimedia applications have shown to be particularly appealing, especially those focusing on video coding. In fact, video encoders and decoders (codecs) have become one of the fundamental building blocks of modern embedded systems and, depending on the type of application, one or even more codecs supporting different standards are frequently made available in a single system. Nonetheless, the majority of current multimedia platforms typically implements the H.264/AVC standard [25], owing to its widespread usage. On the one hand, this is mostly due to the high compression efficiency levels offered by H.264/AVC, which provides 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 [32]. On the other hand, the extraordinarily high flexibility of H.264/AVC allows it to be efficiently used in several different and distinct application domains, such as low bit-rate Internet streaming applications, HDTV broadcasting, and Digital Cinema [24]. However, most H.264/AVC coding tools still impose a significant cost in terms of computational complexity for both video encoding and decoding operations. This is mostly owed to the more elaborate and highly compute intensive algorithms that have been defined by this standard.

As a consequence, distinct approaches have been adopted by system designers to comply with the performance and flexibility requirements of H.264/AVC codecs. For example, homogeneous multi-core architectures with a significant amount of processing cores have been considered in the General Purpose Processor (GPP) market [26]. More recently, multi-core architectures based on GPPs and Graphics Processing Units (GPUs) have also been considered to improve the performance of such general purpose systems [22]. In these architectures, the encoding and the decoding times are reduced by offloading the computation of the most time critical operations to the GPU. Such parallelization task usually allows to greatly accelerating the execution of the most compute intensive algorithms of a video codec, e.g., motion estimation, deblocking filter, transform coding, etc. Nonetheless, the whole speedup values offered by these

systems are usually not very impressive, as a result of the high data dependencies that characterize the H.264/AVC coding algorithms.

On the other hand, alternative solutions must be considered in the design of embedded systems, owing to the inherent strict constraints in terms of power consumption and cost. For such implementations, heterogeneous multi-core architectures are commonly adopted. The basis of these designs usually consists of a couple of very efficient programmable processors (typically two and no more than four), which support the implementation of both the less complex and the most irregular algorithms and operations of the considered applications [3]. Depending on the desired performance levels, such processing cores may consist of moderate complexity and very efficient GPPs or of VLIW processors, when higher processing capabilities are required. This is the case of media processors, like the TI OMAP platform that combines powerful DSP and RISC cores with very efficient multimedia hardware accelerators in a single multi-core processing structure. Nevertheless, although the ISA of these programmable processors quite often includes SIMD and other specialized instructions, the realization of the most typical signal processing and multimedia operations often requires several clock cycles (e.g. matrix transposition [27]). As a result, several different specialized hardware accelerators are also usually embedded in these heterogeneous architectures, so as to optimize the implementation of these critical applications (i.e., reduction of the processing time, increase the energy efficiency, etc.). For example, the TI OMAP platform uses its imaging, video and audio accelerator (IVA) to speed up several video coding operations (e.g., motion estimation, the forward and inverse DCT and pixel interpolation) [4].

Due to these heterogeneity and complexity issues, the development of such architectures is usually based on efficient hardware/software co-design strategies [33], where both the hardware and the software components are designed together to optimize the system performance, power consumption and hardware cost. Nevertheless, these two components focus distinct optimization directions. The software component tries to improve the performance and size of the video coding software not only by fine tuning the implementation of its several modules, but also by optimizing all the involved data transfers and control communications. Conversely, the hardware component aims at populating the multi-core structure with computational/power/area efficient processing cores, as well as with interconnection structures implementing fast communication protocols. With this hardware/software co-design approach, the intended global system performance can be achieved, provided that the application can be efficiently parallelized and the hardware accelerators that are adopted in such multi-core platforms offer relevant performance gains, i.e., effective speedup values.

In what concerns to multimedia systems for video coding, and in particular those implementing the H.264/AVC standard, several different modules of both the video encoding and decoding algorithms are typically mapped into specialized hardware accelerators, as a consequence of its huge impact in the performance of the video codec (e.g., motion estimation [6]). This results from three basic needs: achieving real-time operation, fulfilling High Definition (HD) requisites and attaining efficient system implementations in terms of hardware cost and power consumption [2]. In this sense, the transform coding module is especially attractive to be implemented in hardware accelerators. Firstly, because it is a fundamental and mandatory operation that

is present in both the video encoder and decoder processing paths (see Fig. 1). Then, because it is also one of the most regular parts of the encoding and decoding algorithms, which allows to design simple and very efficient hardware structures to support the involved operations. Yet more important, because the transform coding algorithms considered in the H.264/AVC standard significantly influence the performance of both video encoding and decoding systems, due to their throughput requirements, high computational cost and involved latency. Such constraints are mostly owed to the multi-transform nature and to the finer granularity of the transform coding tool used in H.264/AVC, which involve processing rates as high as 388 Mpixels/sec for the HD720p format or 889 Mpixels/sec for the HD1080p format [11].

From the previous discussion, it can be easily concluded that efficient, scalable and unified (i.e., multi-transform) architectures are required for the development of video encoding and decoding embedded systems. On the one hand, to guarantee the realization of all transform operations in an efficient manner not only in terms of processing speed, but also in what concerns to the hardware efficiency and power consumption. On the other hand, to achieve scalable systems that can be easily configured or adapted in run-time, to better comply with the hardware cost, the performance and the power

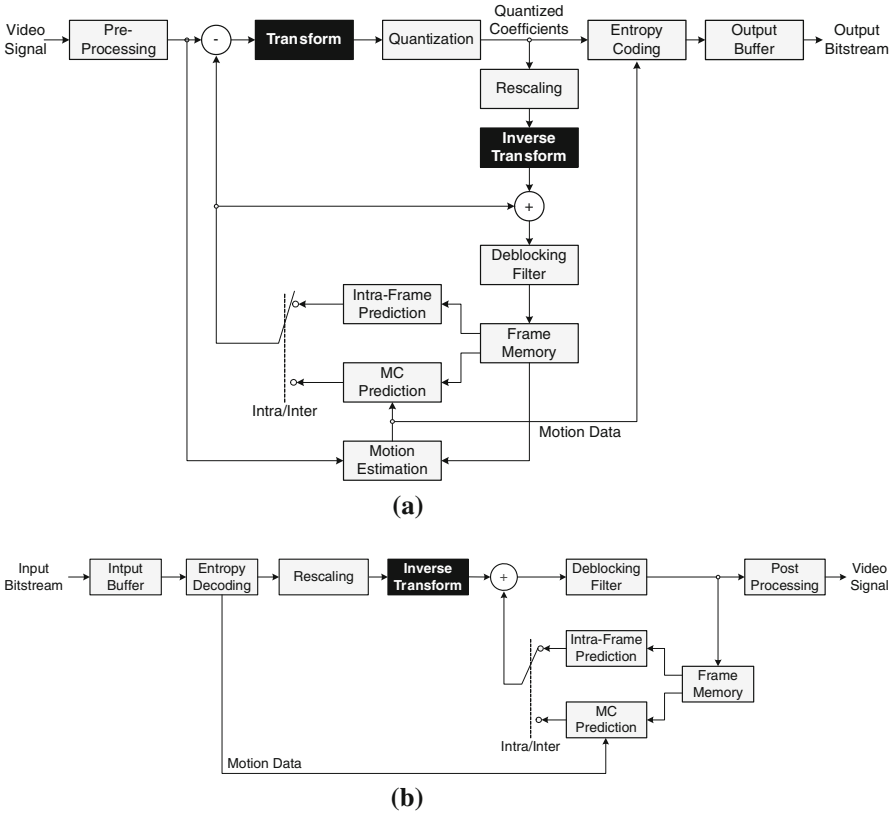

Fig. 1 Generalized block diagrams of a H.264/AVC encoder and decoder. **a** Encoder. **b** Decoder

consumption requisites of video coding applications. In accordance, this manuscript presents a high throughput and scalable architecture for unified transform coding in H.264/AVC that is able to fulfill the above requirements.

The rest of this manuscript is organized as follows. Section 2 presents the H.264/AVC transform functions and briefly reviews the most prominent related work. Section 3 introduces the proposed unified architecture for transform coding in H.264/AVC, while Sect. 4 describes a prototyping multi-core embedded system that uses this design as a hardware accelerator. Experimental results considering implementations of such processing structures in a Xilinx Virtex-5 FPGA device, embedded in a ML506 development board, are presented and discussed in Sect. 5. Finally, Sect. 6 concludes the presentation.

2 Transform Coding in H.264/AVC

2.1 Transform Algorithms

The H.264/AVC transform coding unit implements a two level hierarchical transform path, based on multiple and finer granularity integer approximate transforms of the classical 8-point two-dimensional (2-D) Discrete Cosine Transform (DCT) [25]. In total, six different transforms are used to encode each block of residual data, namely, the 8×8 and the 4×4 forward and inverse integer DCTs, the 4×4 and the 2×2 Hadamard transforms. Nonetheless, implementations of H.264/AVC codecs targeting embedded systems typically consider only the 4×4 and the 2×2 transforms. Furthermore, all these 4×4 and 2×2 transforms can be used in every H.264/AVC profile/level combination. In contrast, the 8×8 transforms are only allowed in the High Profile levels [32], mainly intended to be used in very high resolution applications.

This combination of transform operations, together with the usage of smaller block sizes, allows to greatly improving the compression performance of the encoders. Moreover, inverse transform mismatch problems are also avoided in H.264/AVC, as a result of all the transforms being computed with integer resolution. Such important characteristic allows them to be implemented using only addition and shift operations, which greatly reduces the computational complexity and the hardware requirements for implementations based on dedicated hardware structures.

In H.264/AVC, the 4×4 forward integer DCT is applied in the first transform level to all 4×4 blocks of residual data, which result from both the motion-compensated prediction and the intra-prediction stages. Likewise the conventional DCT, this integer transform is also orthogonal, separable, and presents a strong decorrelating performance. In practice, it consists of a simplified 4×4 DCT, where the scaling factor component was transferred to the quantization stage of the encoding algorithm [32]. As a result, the forward transform contains only the $1, -1, 2$ and -2 coefficient values, as shown in Eq. 1, which minimizes its implementation requirements. Moreover, this transform can also be implemented using the usual row-column decomposition process, due to its separable nature. As a consequence of its orthogonality property, the same considerations also apply to its corresponding inverse transform, whose kernel is presented in Eq. 2.

$$C_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \tag{1}$$

$$C_i = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix} \tag{2}$$

The second transform level is based on the Hadamard transform and provides the required means to better exploit the redundancies in smoother areas of a picture. More specifically, the Hadamard transform is used to process the DC coefficients of each transformed block computed in the previous path level. Nonetheless, two distinct Hadamard transforms are used to process the gathered DC coefficients blocks of luma and chroma pixels.

For macroblocks being encoded using the 16×16 intra-prediction mode, the 4×4 Hadamard transform is used to transform the sixteen DC coefficients of all the 4×4 luminance blocks within a macroblock. Once more, this transform can be regarded as a simplified version of the forward integer DCT, where the coefficients 2 and -2 are replaced by the coefficients 1 and -1, respectively. Hence, the 4×4 Hadamard transform retains all the properties of the 4×4 integer DCT, but it further reduces the complexity requirements of the corresponding implementation algorithm. Namely, the forward and inverse transform functions are exactly the same, due to its symmetric nature, and can be implemented using only integer additions and subtractions.

Similarly, the four DC coefficients of each of the two 2×2 chroma blocks of an intra-predicted macroblock are also encoded using a Hadamard transform. However, this is an even simpler transform, with only four coefficients, which shares the same properties offered by the 4×4 Hadamard transform. Equations 3 and 4 show these 4×4 and 2×2 Hadamard transform kernels, respectively.

$$H_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{3}$$

$$H_{2\times2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4}$$

2.2 Architectures for Transform Computation

Several different specialized and dedicated architectures have been proposed in the literature to implement the H.264/AVC transform coding operations [7–10,12,17–21,23,29–31]. Such proposals mostly consist of efficient VLSI designs implementing fast and optimized transform algorithms, in order to mitigate the involved computational

complexity constraints and speedup the corresponding computation tasks. They are usually classified either as *dedicated transform kernels* or *unified transform kernels*.

Dedicated transform kernels implement a single transform function, which can be any of the forward or inverse DCT or Hadamard transforms defined in the H.264/AVC. Typically, direct 2-D transform architectures are used when high performance implementations are desired [8,17]. These structures often implement fast and simplified transform algorithms involving huge amounts of hardware resources, which are required to assure the parallel processing of all the transform coefficients. Consequently, they are characterized by having significantly high hardware costs and power consumption requirements. Moreover, the typical cascaded processing design of these architectures (with long depth processing data paths), further characterizes them as being long latency architectures. This poses significant constraints when real-time operation is required.

To overcome such problems, several of the designs that have been proposed actually follow a row-column decomposition strategy [18,19]. In this approach, the considered 2-D transform function is computed using a single one-dimensional (1-D) transform architecture and transposition logic. Such circuit, which typically consists of a transposition memory, is used to transpose the 1-D intermediate results and to feed them back into the 1-D transform structure. Consequently, the hardware efficiency level of these designs is relatively high, since the same 1-D transform processing circuit is used twice in the computation of the 2-D transform function. Moreover, such improved hardware efficiency also contributes to a significant reduction of the power consumption and hardware requirements associated with this type of architectures. Nonetheless, in most cases such gains are usually not as high as it would be expected, owing to the usage of the transposition memory. In addition, this circuit also introduces some delay in the processing of the 2-D transform coefficients that, again, can be critical to achieve real-time operation.

As a final remark, it should be noted that the usage of such specialized and highly optimized 2-D and 1-D computation structures also greatly restricts their reutilization in the design of newer dedicated transform kernels. In particular, the rigid hardware structure that usually characterizes these architectures often prevents any modifications to their base structure, therefore making it very difficult to reutilize them as efficient hardware accelerators in multi-standard embedded systems.

Unified transform kernels are another class of architectures for transform coding that overcome these problems by being capable of implementing multiple transform functions [7,9,12,21,23,29,30]. The supported transform functions can be computed either in parallel [12,29,30] or at distinct time instants [7,9,23], and using transform engines with both direct 2-D architectures and 1-D architectures based on the row-column decomposition approach. Recently, this class of designs has had its functionality further extended with the integration of the quantization and rescaling operations into the transform coding design [10,16,28]. For both cases, there are also a couple of reconfigurable solutions that have been recently proposed [20,31], in order to improve the hardware efficiency of such flexible and hardware costly processing structures. Nevertheless, not so many unified transform architectures have been proposed. Moreover, most of these designs present significant limitations, such as:

– High throughput solutions targeting high resolution image formats are typically based on direct 2-D transform circuits that present huge hardware cost and power consumption requirements, thus reducing their attractiveness for low power applications (e.g., portable handheld and other battery supplied devices);
– Poor hardware efficiency rates of most unified transform designs, resulting in significant hardware overheads and increased power consumption values, which also reduces their attractiveness for low power applications;
– None of such structures can be effectively scaled and very few can be reconfigured in run-time to meet the requirements of the target video coding application, i.e., performance, power consumption and hardware cost. In practice, the majority of these architectures consist of pseudo-reconfigurable structures with fixed hardware functional blocks that can have their functionality reprogrammed in run-time.

To simultaneously comply with all these limitations, a different category of architectures is herein proposed. This class of architectures is based on a systolic array structure, which has proved to provide rather efficient solutions in terms of performance, hardware efficiency, scalability and power consumption [15]. Furthermore, such structure is also especially adequate to implement regular algorithms with high data throughput and computational rates, which are characteristic of the H.264/AVC transforms. However, few systolic designs have been proposed to implement transform kernels for video coding, and most of them concern the 8×8 DCT adopted in previous ISO MPEG-x and ITU-T H.26x standards [5]. In fact, to our best knowledge, the proposed architecture is the first scalable systolic array structure for unified transform coding in H.264/AVC that has been proposed.

3 Unified Architecture for Transform Coding

The 4×4 DCTs and the 4×4 and 2×2 Hadamard transforms defined in the H.264/AVC standard, whose implementation is mandatory for all video coding modes, are characterized by their high regularity and low complexity. This can be seen in Eq. 5 that represents a generic 2-D transform, where X, Y and C denote the input data, the output data and a transform kernel, respectively.

$$Y_{ij} = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} X_{kl} C_{ik} C_{jl}, \quad i, j = 0, \ldots, N-1 \tag{5}$$

Nevertheless, it can also be concluded from Eq. 5 that the realization of such algorithms in hardware demands high data throughput and computational rates, especially for high definition videos and real-time operation. Moreover, in order to improve the hardware efficiency and to better adapt the architecture to the requirements of the target application in terms of performance, hardware usage and power consumption, the circuits that implement these algorithms are required to present highly flexible scalability characteristics. Altogether, these are highly critical factors in the design of efficient hardware structures implementing this class of algorithms. At this respect, systolic array processors have proved to provide

rather convenient solutions to overcome all of the above mentioned problems and requisites.

3.1 Mapping the Transform Algorithms into a Systolic Architecture

Systolic arrays consist of locally connected Processor Elements (PEs), each one computing a restricted and very well defined set of operations, which exchange the data to be processed with each other using a pipelined dataflow. This design approach is usually regarded as requiring restricted control and interconnection circuitry and offering high clock frequencies. Consequently, such structures are usually able to provide high computational rates and high data throughput, while still providing very high hardware usage efficiency levels.

The structure of a systolic array targeting a given algorithm can be obtained by using a well known methodology that is described in [15]. By using such methodology, it is possible to derive dependence graphs for the considered algorithms, in which the computation nodes represent the basic operations that need to be performed and the edges connecting such nodes define the order by which the operations need to be executed. Hence, hardware implementations of such graphs will include two different types of units: (i) PEs, that are designed to realize the operations specified by the computational nodes; and (ii) communication links, which implement the edges.

To improve the hardware utilization rate and to achieve viable hardware implementations, where the graphs should be defined in indexed spaces with no more than two dimensions, more elaborate techniques must be used, like projection, algorithm decomposition into subparts or scheduling [15]. In fact, this is the case for the H.264/AVC transforms, which are defined over a four dimensional indexed space owing to the involved operations, as it can be seen in Eq. 5. Equations 6 and 7 illustrate this decomposition into two 1-D terms for a generic H.264/AVC transform algorithm, where M denotes the intermediate data results of the 1-D transform function.

$$M_{il} = \sum_{k=0}^{N-1} X_{kl} C_{ik}, \quad i, l = 0, \ldots, N-1 \tag{6}$$

$$Y_{ij} = \sum_{l=0}^{N-1} M_{il} C_{jl}, \quad i, j = 0, \ldots, N-1 \tag{7}$$

As it can be seen, the first part deals with indices l and k of Eq. 5 and thus computes a 1-D column-wise transform function (Eq. 6), while the second part is spawned across indices i and j of Eq. 5 and computes a 1-D row-wise transform function using the transposed results obtained from the first part of the algorithm (Eq. 7).

As a result of the previous discussion, it is clear that several different systolic arrays implementing the H.264/AVC transforms can be derived using the methodology presented in [15] and by considering different optimization techniques. For example, low performance designs can be obtained using serial implementations based on a single PE, while high performance architectures can be obtained using fully parallel 2-D array

structures. However, not so many of such structures are able to simultaneously comply with the aimed high performance, scalability and reduced hardware cost requirements.

To achieve such goals, three different techniques were applied to obtain a viable hardware implementation of the unified transform architecture for H.264/AVC that is now proposed. Firstly, the base algorithm, depicted in Eq. 5, was *decomposed in two sub-parts*, in order to follow the row-column decomposition approach depicted in Eqs. 6 and 7. As it was previously mentioned, this approach not only reduces the hardware cost of the circuit implementing the transform algorithm, which consists only of a simpler 1-D transform kernel, but also significantly increases the utilization rate of its PEs. Moreover, it also provides significant savings in terms of power consumption, owing to its fewer hardware requisites, without greatly compromising the performance requirements. Then, proper *projection* and *scheduling* techniques were applied to further improve the hardware efficiency of the conceived architecture, thus resulting in a 2-D systolic array structure. Finally, a hybrid configurable and dedicated architecture was developed for the proposed PEs, in order to further improve the hardware efficiency rate of such structure and to allow the computation of all the 4×4 and 2×2 H.264/AVC transforms.

3.2 Proposed Architecture

The architecture of the proposed unified transform kernel for H.264/AVC is composed by a control unit and the three functional modules depicted in Fig. 2, i.e., a scalable *array of PEs*, a *row-column transposition switch* and an *input buffer*.

The PE array is used to compute the 1-D transforms corresponding to the row-column decomposition of the transform algorithms. In its base configuration, illustrated in Fig. 2, it is composed by 16 PEs in order to allow a straightforward computation of all the 4×4 transforms using a quite simple control unit. However, this 4×4 configuration of PEs also allows the simultaneous computation of two 2×2 transforms using the two upper rows of PEs. Such important feature is achieved with an optimal trade-off between architecture functionality, design complexity and hardware efficiency. In fact, the proposed multi-transform functionality allows to fully process a macroblock (i.e., compute both the 4×4 and the two 2×2 2-D transforms) at the cost of a small hardware overhead (the two 2×1 multiplexers depicted in the lower end of the transposition switch, shown in Fig. 2) and a minimal increase in the complexity of the control unit.

Within the systolic array all the PEs compute the same operations and share an identical architecture. Moreover, they communicate with each other by using a simple and reduced signal interface, as it can be seen in Fig. 3. To minimize the delays resulting from control operations, and thus to maximize the data processing rate, all the control logic required for the correct circuit operation is also distributed and embedded in the architecture of the PEs.

According to Fig. 3, the architecture of the proposed PE is composed by two main blocks: the *arithmetic module* and the *control module*. The transform operations are computed in the *arithmetic module*, by using an accumulator and a specialized multiplication circuit that has its multiplier value programmed according to the type of

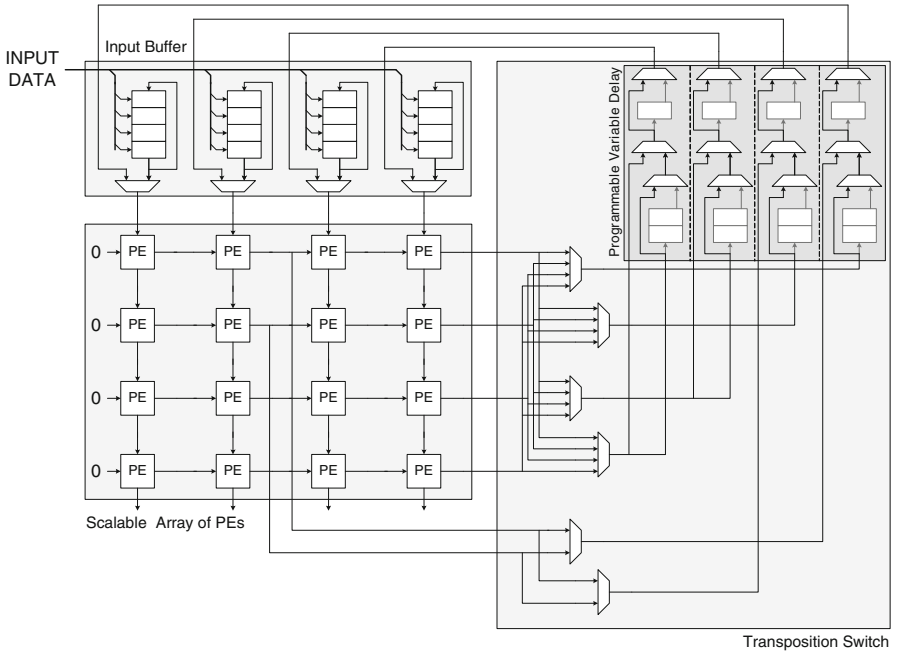

Fig. 2 Block diagram of the proposed scalable unified transform kernel

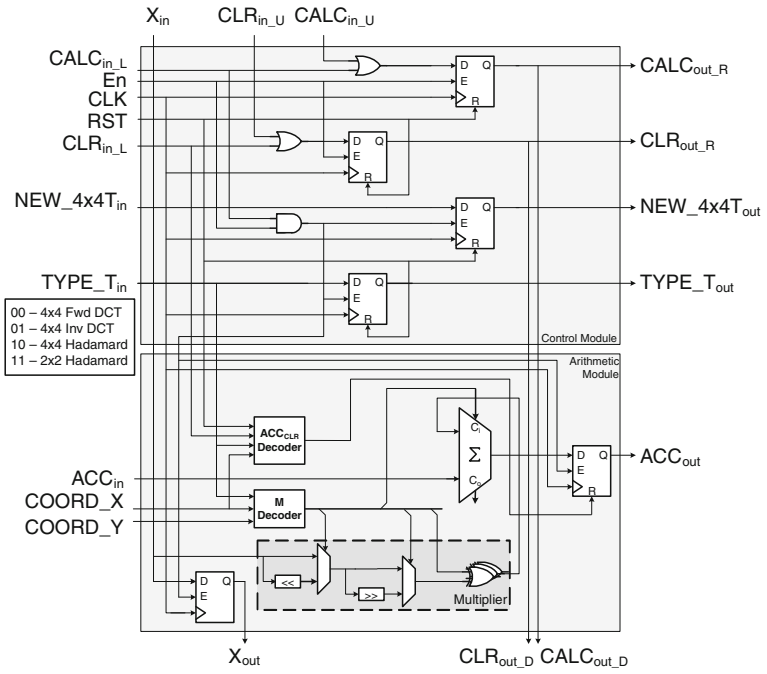

Fig. 3 Architecture of the PEs composing the processing array

transform being implemented. In fact, since all the multiplier values are powers of two (i.e., $1, -1, 2, -2, \frac{1}{2}$ and $-\frac{1}{2}$), this multiplication circuit consists of an efficient and flexible arithmetic barrel-shifter, properly specialized for the computation of the H.264/AVC transforms. By using such circuits, the required transform operations are computed in a 3-step process.

Firstly, the data values to be processed (X_{in}) are loaded into an internal standing-data register. Moreover, the partial value of the transform operation being computed, which is calculated by a different PE in a previous clock cycle, is also placed at one of the inputs of the PE's accumulator (ACC_{in}). Then, depending on the transform to be implemented (specified by the TYPE_T signal) and on the coordinates of the coefficient under processing (identified by the COORD_X and COORD_Y signals), such partial value is updated with the result of the multiplication of the residue value (or the intermediate transform coefficient value) by the selected multiplier value. Finally, the result of this multiplication is stored in another internal standing-data register before being propagated to the next PEs, in order to shorten the critical path of the circuit.

On the other hand, the *control module* is responsible for generating the signals required to command the operation of the PEs, as well as for guaranteeing the correct flow of such signals along the row-column transposition switch. For example, the control module of each PE manages the propagation of the NEW_4x4T signal throughout the PE array, so that the command to reset all the control logic in the transposition switch, which results from the processing of a new 4×4 transform function, is only activated when the first results of that operation effectively reach the transposition switch.

Regarding to the specific control signals of the proposed PE, they allow to command the circuit operation and to clear the accumulated values. The accumulator is cleared whenever the system global reset signal (RST) is activated, or on demand by the system control unit (CLR). Likewise, the PE operation is controlled by the system global enable signal (EN) and by the local enable signal (CALC) generated by the system control module. The local reset and enable signals are used to directly command the PE in the top-left corner of the array, which then uses the CLR and CALC signals to propagate them to the remaining PEs of the array in the horizontal and vertical directions, respectively. By doing so, it is therefore possible to maximize the data processing rate within the array, since the only PEs that will be stalled in any given time instant are those lacking the data to be processed (for example, when the input buffers are empty).

The *transposition switch* is used to implement a hardwired row-column transposition of the data processed by the 1-D transform kernel. Unlike most of the existing transposition units, this one does not include any memory circuits. In fact, it mostly consists of a set of multiplexers that allow direct row-column transposition not only for 4×4 blocks of data (using the four 4×1 multiplexers), but also for two simultaneous 2×2 blocks of data (the lower two 4×1 multiplexers and the two 2×1 multiplexers). This innovative design allows to save significant hardware resources in the implementation of the transposition switch, since it avoids the usage of the typical memory companion cells. Moreover, it does not impose any performance penalty on

the whole circuit, owing both to the pipelined processing nature of the PE array and to the much lower propagation time of the transposition switch circuits.

Finally, the *input buffer* is used to feed the PEs of the systolic array either with the residue data coming from the intra- and inter-predictions (in the video encoders), or with the transform coefficient values (in the video decoders). This unit is highly required to minimize the delays when accessing the external data memories where such data is stored, and also to guarantee a regular dataflow within the systolic array. Consequently, the input buffer was designed to work concurrently with the transform computation circuits, in order to optimize the overall data processing rate. Moreover, to better exploit cache access patterns, it also allows the parallel loading of a full line of residue values (or transform coefficients, in video decoders) from the external memory. In addition, this unit is also capable of feeding the transform processing array with the previously processed row-column transposed data. Similarly to what happens with the residue values, these data elements are serially transferred to the several columns of the array, in order to comply with its pipelined dataflow.

3.3 Dataflow Model

The dataflow model of the proposed architecture was designed to maximize the data processing rate within the PE array. Consequently, it is based on a quite simple and distributed control scheme that allows computing the desired transforms in a pipelined fashion. Figure 4 illustrates such dataflow for the processing of a complete 1-D 4×4 transform. The three represented data-sets correspond to the processing of two consecutive 4×4 blocks: two data-sets of residue values (or transform coefficients), depicted using a solid-line, and one data-set of intermediate values for the row-column decomposition algorithm, represented using a dashed-line.

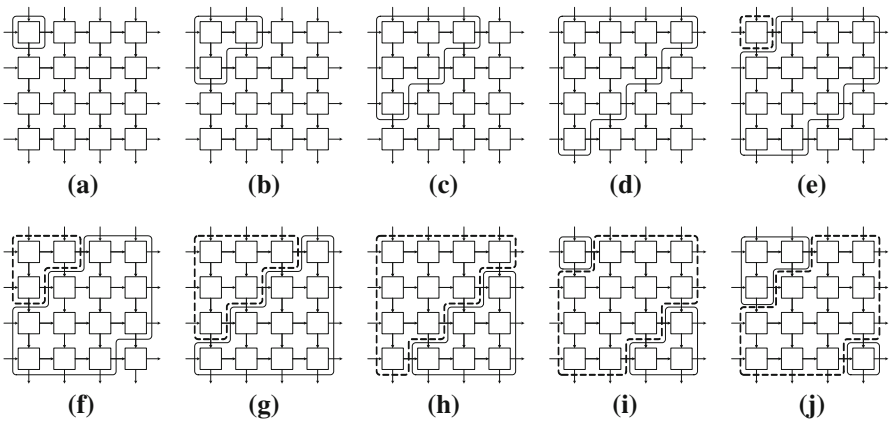

Fig. 4 Dataflow in the PE array for one 4×4 data block. **a** $t = T_0$, **b** $t = T_0 + 1$, **c** $t = T_0 + 2$, **d** $t = T_0 + 3$, **e** $t = T_0 + 4$, **f** $t = T_0 + 5$, **g** $t = T_0 + 6$, **h** $t = T_0 + 7$, **i** $t = T_0 + 8$ **j** $t = T_0 + 9$

As it can be seen in Fig. 4, the transform coefficients (or residue values) are computed in a wave front manner within the PE array. The data to be processed is fed through the input buffers into the PEs in the top row of the array, and are then propagated in the vertical direction to the remaining PEs of the array. Conversely, the control signals for all the PEs enter the array through the PE on its top-left corner, which then propagates them to the remaining PEs of the array in both the horizontal and vertical directions, synchronously with the data shifting. Such signals are also propagated into the transposition switch, where they are used by the control logic to select the proper data to be feedback to each column of the PE array.

Altogether, the proposed processing scheme makes it possible to start the computation of a different coefficient in a new row of the array on each clock cycle, provided that the input buffers are not empty. Simultaneously, it also allows executing another iteration of the considered 1-D transform algorithm for all the coefficients that are being processed in the remaining rows of PEs. Consequently, this approach offers a maximization of the data processing rate within the array, since the only PEs that will be stalled at any given time instant are those lacking input data to be processed. In such conditions, which correspond to a full pipelined processing, the proposed structure is able to compute a 2-D 4×4 transform in 14 clock cycles.

3.4 Scalability

In what concerns scalability, the proposed unified transform architecture presents increased advantages when compared to other existing processing structures with similar functionality. Such property strictly concerns the array of PEs, which can be configured to support the following three setups: a 4×4 PE array, a 2×4 PE array and a 1×4 PE array.

The setup with 4×4 PEs, which consists of the base configuration described in Sect. 3.2, offers the highest processing rate. However, it also imposes the highest hardware cost. Nevertheless, it is the most suitable configuration for applications that mainly focus on throughput and performance, such as real-time systems or high resolution video codecs.

Conversely, the 1×4 PEs setup offers the greatest savings in terms of hardware resources, due to the usage of only a single line with four columns of PEs, as it can be seen in Fig. 5a. However, this comes at the cost of a significant reduction in terms of computational performance, since for this array configuration four (two) times more clock cycles are required to process a 4×4 (2×2) H.264/AVC transform. As a result, this setup is more suitable for applications that need to comply with strict and reduced power consumption constraints or that do not require very high performance levels.

Finally, the 2×4 PEs setup, presented in Fig. 5b, comes as a compromise solution between performance and hardware cost. When compared to the base setup, it offers a reduction of the required hardware resources by only using two lines with four columns of PEs. On the other hand, it only imposes a moderate penalty in the resulting throughput of the architecture. In fact, the performance is only affected for the computation of the 4×4 transforms, which require twice as much clock cycles as the base setup.

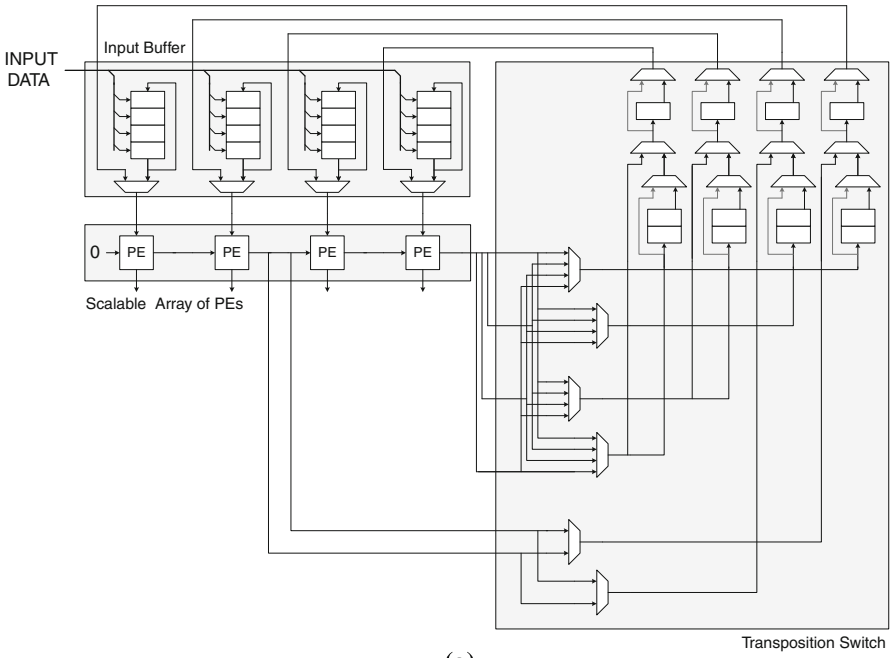

Fig. 5 Alternative setups for the proposed scalable unified transform kernel. **a** Setup with 1×4 PEs, **b** Setup with 2×4 PEs

Independently of the adopted setup for the array of PEs, it is important to note that the hardware structure of all the remaining functional modules of the proposed architecture remains unchanged. In fact, extending the scalability property to the remaining modules of the architecture does not bring any advantage: not only it significantly increases the complexity of the control module, but it also involves the reconfiguration of very fine grained hardware structures, which represents a quite inefficient task (e.g., multiplexers and registers). Hence, to comply with the more static nature of the internal structures of the input buffer and transposition switch modules, some configurable interconnection structures and auxiliary processing elements were embedded in the proposed architecture.

Firstly, the output interface of the PE array was carefully designed in order to always provide the same interface signals, through which the correct data for each of the three possible setups of the PE array is exchanged with the row-column transposition switch. Secondly, the row-column transposition switch also includes programmable Variable Delay Elements (VDEs), as shown in Fig. 2, which allow the realization of the transposition operation for all the three setups of the PE array. The need for these delay elements results from the fact that with the elimination of two or more rows of PEs it becomes impossible to do the transposition operation without additional transposition registers to temporarily store the intermediate results that are being computed within the PE array. Hence, such delay elements mostly consist of a set of configurable bypass multiplexers and standing data registers, which are activated according to the amount of lines of PEs that are removed from the base array configuration, as it is depicted in Fig. 5. Consequently, the performance of the architecture is not influenced by the programmable VDEs, since they merely extend the PE array pipeline through the transposition switch.

The configuration of the PE array to implement any of the three presented setups is commanded by the main control module of the configurable system. This command can be triggered in several different scenarios or operation modes of the video coding system. For example, as a result of (i) a change of the video resolution or frame rate, (ii) the energy supply capabilities of the system's power supply module, or (iii) on demand, by the application itself, to adjust the allocated hardware resources to the requirements of the whole system.

4 Multi-Core Processor for Advanced Video Coding

To validate the functionality and evaluate the performance gains offered by the proposed unified transform architecture in a practical realization, a prototype of a multi-core system for H.264/AVC video coding was conceived targeting embedded implementations using Xilinx FPGA devices.

As it is illustrated in Fig. 6, a microBlaze processor was used to implement the Central Processing Unit (CPU) of this embedded system, which includes other IP cores from the Xilinx EDK library for added functionality: programmable memory controllers to access the program and data RAMs connected to the Local Memory Bus (LMB) of the microBlaze processor; a RS-232 UART for I/O operations and user interaction; a 32-bit timer for profiling purposes; an IRQ controller to prioritize the

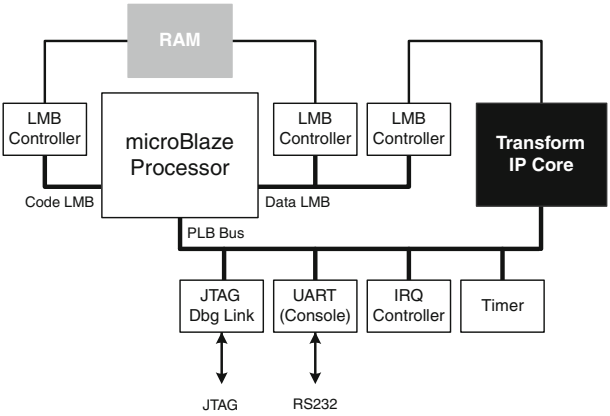

Fig. 6 Block diagram of the developed multi-core embedded system

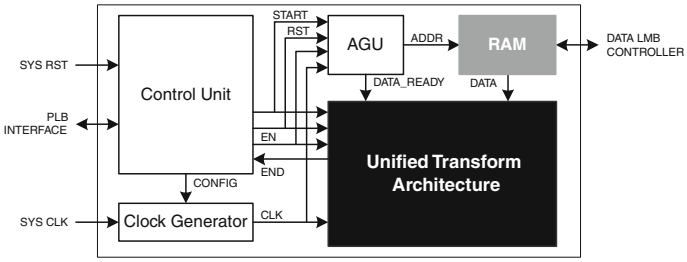

Fig. 7 Block diagram of the transform coding IP core

interrupt requests from the timer, the UART and from other peripherals; and a JTAG controller for PROM programming and debugging purposes. The developed multi-core system also includes an additional IP core for the computation of the H.264/AVC transforms. All these hardware structures were interconnected using the LMB and Processor Local Bus (PLB) interfaces, as a result of their different I/O bandwidth requirements and communication interfaces.

In what concerns to the specialized IP core for transform coding, it can implement any of the three configurations of the proposed unified transform architecture presented in Sect. 3.2 and includes several different hardware circuits to guarantee optimal performance levels, as it is shown in Fig. 7. In fact, to avoid any data processing delays resulting from eventual communication bottlenecks, the IP core supports two distinct communication protocols and includes a local memory bank with its own Address Generation Unit (AGU). Furthermore, the conceived IP core also includes a local clock generator module that enables the unified transform architecture to operate with a different clock frequency than the system's CPU.

All the data transfer operations required to exchange the residue values and the transform coefficients between the local data RAM of the IP core and the system data RAM are conducted using the higher bandwidth and low latency LMB of the microBlaze processor, in order to not compromise the performance of the implemented

video coding system. Conversely, the commands required to control and evaluate the operation of the IP core are exchanged using the PLB and a very simple control block, owing to the reduced amount of data to be transferred and the performance requirements for such communications.

The IP core internal memory module consists of a true dual-port RAM with a total capacity of 4 kB, organized in lines of 64 bits, that is mapped in the address space of the global system, as shown in Fig. 8. This not only allows faster accesses to this memory by the system CPU, but also a better adaptation of the memory bank interface to both the LMB and the proposed architecture for transform coding. In addition, the considered memory capacity supports the storage of the data corresponding to at least two different macroblocks inside the IP core, and thus is able to parallelize the data transfers and the computation tasks using a double buffering technique.

Regarding to the control unit of the IP core, it is mostly composed by hardware registers and other simpler control logic (e.g.: address decoders and data multiplexers), which support the PLB protocol and the core user configuration interface. As it can be seen in Fig. 8, such interface encompasses three different sets of memory mapped registers for a complete system interaction with the core. The set/clear control registers and the status registers provide the necessary means to control and evaluate the core operation, respectively. The core configuration register is used to configure the PE array into one of its three possible setups, as discussed in Sect. 3.4. Finally, the clock configuration register allows to program the IP core local clock generator module to operate with the desired clock frequency. The memory region identified as

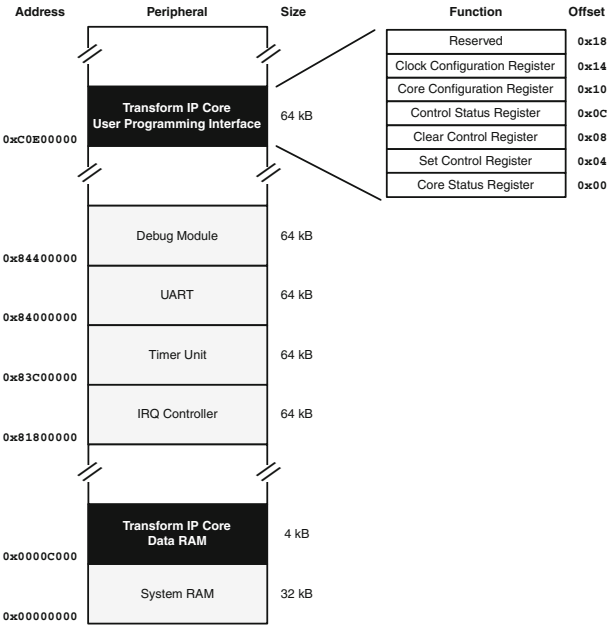

Fig. 8 Memory map of the multi-core processor

Reserved in Fig. 8 hides a memory mapped register from the user interface, which can be used for debugging the core operation in run-time.

5 Implementation and Results Assessment

5.1 Implementation Results

The proposed unified transform architecture was fully described using the IEEE-VHDL hardware description language and implemented in a Virtex-5 XC5VSX50T-1FFG1136 FPGA device, in order to assess its advantages in terms of performance and hardware cost. Such parameterizable VHDL description follows a strict modular approach, using independent and self-contained functional blocks, and makes extensive use of *generic* configuration inputs to comply with the scalability requirements of the architecture. Table 1 presents the results that were obtained with the implementation of the three setups of the proposed processing structure (described in Sect. 3.4) in the adopted general purpose FPGA device. Such implementation procedure was realized using the Xilinx ISE 13.2i development toolchain, including the whole synthesis and place and route tools, and by considering optimizations targeting the maximization of the operating clock frequency of the circuit.

In terms of implementation resources, the data presented in Table 1 expresses the very low hardware cost of the proposed architecture. As it will be discussed in Sect. 5.2, the higher hardware efficiency of this structure allows it to compute all the H.264/AVC 4×4 and 2×2 transforms requiring fewer hardware resources than most of the existing architectures that only compute a single transform function. Moreover, the presented results also demonstrate that the required hardware resources scale well with the PE array configuration. This is a very important factor for reconfigurable implementations, since despite the low hardware cost of the proposed PEs, the systolic array requires between 50 % and 80 % of the total hardware resources of the architecture. Consequently, the ability to choose different setups for the proposed structure enables its implementation in FPGA devices with both reduced and greater amounts of hardware resources.

Regarding to the obtained performance levels, the maximum allowed clock frequencies presented in Table 1 evidence the high processing rate that is offered by the proposed structure (up to 5.09 GOPS). Furthermore, by taking into account these results and the data presented in Table 2, it can also be concluded that the proposed architecture allows to compute, in real-time (38 fps), the whole set of mandatory forward and inverse transforms defined in the H.264/AVC standard for video sequences with resolutions up to $4,320\times7,680$ pixels (UHDV).

Table 1 Implementation results of the proposed scalable architecture in a Xilinx Virtex-5 XC5VSX50T-1FFG1136 FPGA

Configuration	Registers	LUTs	Max. Freq. (MHz)
4×4 PEs	1,167	1,073	318
2×4 PEs	896	780	321
1×4 PEs	641	603	323

5.2 Comparative Analysis

In order to evaluate the performance and hardware cost benefits of the proposed architecture when compared with other alternative solutions, the most related and prominent designs described in the literature were thoroughly reviewed [1,7,10,14]. Table 2 presents the results of the conducted comparative analysis. It uses as figures of merit the hardware cost in terms of slice-count, the maximum clock frequency, the latency in Clock Cycles (CCs) and the data throughput rate (computed as the number of transform coefficients processed in one CC). However, due to the distinct functionalities offered by all the considered architectures and to the diverse FPGA device technologies that were selected for their implementation, this comparison was further extended by defining a more comprehensive figure of merit: the Data Throughput per Unit of Area (DTUA), which is calculated as the ratio of the data throughput rate over the hardware cost in terms of the considered unit of area (a slice).

A straightforward analysis of the presented data clearly shows that the proposed architecture has one of the highest computation rates, despite being one of the few architectures with multiple transform capabilities. As it can be seen, only the structure previously presented in [14] is capable of providing a DTUA very close to the one offered by the proposed unified transform architecture. Nevertheless, it is important to observe that such structure only implements the 4×4 forward integer DCT and that the reported data only concerns the transform kernel module itself. Hence, it can be expected that the real DTUA of this architecture is much lower. In fact, considering that the architecture proposed in [14] requires 32 16-bit adders and 32 16-bit subtractors to implement a direct 2-D 4×4 forward integer DCT kernel, while the one that is now proposed only requires 16 32-bit adders to compute the same 2-D transform function, it is natural to expect that the architecture herein proposed is much more efficient in terms of DTUA.

On the other hand, when compared with other structures for unified transform coding, the proposed architecture clearly presents much higher DTUA values. For example, even the simpler setup of the proposed architecture presents a DTUA that is at least over 2.5 times higher than that of [7], which is one of the most efficient unified transform architectures among the ones that were reviewed. Such results are mainly

Table 2 Comparison of FPGA implementations

Design	[1]	[7]	[10]	[14]*	Proposed		
					4×4	2×4	1×4
4×4 Transforms	Inv	Uni	Fwd	Fwd	Uni	Uni	Uni
Virtex FPGA	V2 Pro	V4	V2 Pro	V2 Pro	V5	V5	V5
Slices ($\times 10^3$)	4.2	2.1	1.6	0.6	0.4	0.3	0.2
Max. Freq. (MHz)	132	167	225	107	318	321	323
Latency (CC)	64	8	8	1	8	16	32
Throughput (Coefs/CC)	1	8	8	16	4	2	1
DTUA ($\times 10^3$)	31.5	636.2	562.5	2867.4	2927.5	2383.6	1624.2
Applications	HD1080p	UHDV	Digital Cinema	UHDV	UHDV	Digital Cinema	Digital Cinema

*The reported data concerns only to the transform kernel module

owed both to the very low hardware cost and to the high clock frequency of the PEs that compose the array, which results in very high throughput rates.

Finally, the experimental results presented in Tables 1 and 2 also emphasize the advantages of the proposed architecture in terms of scalability. As it can be observed, the scaling of the PE array allows to efficiently trade-off the achieved performance (in terms of throughput) for hardware savings. More specifically, the 2×4 setup allows to reduce the hardware cost in about 38 % when compared with the base 4×4 PEs setup. For greater hardware savings, the 1×4 setup reduces the hardware requirements of the proposed architecture by 54 %. Nonetheless, due to the highly efficient pipeline structure of the PE array, which offers the maximum achievable throughput rate (without any need for data stalls), and to the significantly higher operating frequency values, both the 1×4 and the 2×4 setups are still capable of achieving real-time operation for video sequences in the HD1080p format ($1,920 \times 1,080$ @ 30 fps).

5.3 Performance Evaluation in a Practical Realization

The advantages offered by the proposed unified transform architecture to the development of multi-core video coding embedded systems were experimentally assessed with the practical realization of a prototype, as described in Sect. 4, in a Xilinx ML506 development platform [34] equipped with a Virtex-5 XC5VSX50T-1FFG1136 FPGA device. Such realization was conducted using the Xilinx XPS 13.2i and the SDK 13.2i tools and by considering an IP core for transform coding based on a 4×4 PEs array, as well as the same clock signal for all the processors composing the multi-core structure. The obtained implementation results are presented in Table 3, which also includes another set of results concerning the implementation of a reference system for comparison purposes. Such system consists of a simplified version of the developed prototype, which does not include the IP core for transform coding.

As it can be seen in Table 3, the implementation of the proposed multi-core processor presents a very reduced hardware cost, which makes it quite suitable for applications targeting small and medium complexity embedded systems. Such results also show the marginal increase in the implementation area of the whole system owed to the insertion of the hardware accelerator for the computation of the H.264/AVC transform functions, which only requires about 36 % of the reference circuit implementation resources. However, it should be noted that only about 68 % of these hardware resources concern

Table 3 Implementation results of the multi-core processor in a Xilinx Virtex-5 XC5VSX50T-1FFG1136 FPGA

Design	Reference system	Multi-core system
Slices	4,374	5,948
Registers	2,849	4,085
LUTs	2,604	3,759
BRAMs	8	10
DSP48Es	3	3
DCMs	1	1
Max. Freq.	147 MHz	147 MHz

the implementation of the processing structure for transform coding, since all the remaining logic is used to support the other peripheral circuits composing the IP core.

In what concerns the maximum allowed operating frequencies, the obtained results reveal that the microBlaze processor can operate with a maximum clock frequency of 147 MHz, while a 318 MHz clock frequency can be used for the transform coding IP core. Since in the considered prototype the same clock signal was adopted for all the processors of the multi-core system, the maximum operating frequency of such prototype is therefore constrained to 147 MHz, despite the much higher computation rates offered by the IP core for transform coding. In fact, these distinct clock frequency results fully justify the design option of including a clock generator module in the devised IP core for transform coding. With this approach, and by taking into consideration the large storage capacity and the double buffering capabilities of the IP core RAM bank, it becomes possible for other realizations of this multi-core processor to rapidly compute several different transforms without having to push the clock frequency of the whole system to its upper bound limits, which results in an efficient means to reduce the system power consumption.

Regarding to the performance gains provided by the hardware accelerator for transform coding, they were experimentally assessed by computing the forward and inverse 4×4 integer DCT and the 4×4 and 2×2 Hadamard transforms for a set of macroblocks of the following benchmark QCIF video sequences: *bream*, *carphone*, *foreman*, *mobile* and *table-tennis*. Such transform operations were computed both in the IP core for transform coding and in the microBlaze processor, by executing the corresponding transforms of the H.264/AVC reference software [13]. Table 4 presents the average computation times obtained for each of the two platforms, as well as the corresponding speedup values provided by the multi-core structure using the developed IP core for transform coding.

As it can be seen in Table 4, the integration of an hardware accelerator for the computation of the H.264/AVC transforms within the multi-core system provides a significant reduction in the computation times of all the transform operations, despite the simplicity of this prototyping system. Consequently, the significant speedup values obtained with the usage of such IP core can be used to draw a very important conclusion: this dedicated processing structure allows multi-core video coding systems to efficiently support higher resolution video formats and still to comply with the requirements of real-time operation. In fact, the performance of the video coding system is improved not only by having the IP core and the system CPU working in

Table 4 Computation time of the transform operations using the microBlaze processor and the IP core for transform coding

Transform function	Processing core		Speedup
	microBlaze (ms)	IP core (ns)	
4×4 forward DCT	6.87	54	126
4×4 inverse DCT	6.94	54	127
4×4 Hadamard	6.58	54	121
2×2 Hadamard	0.55	27	10

parallel, but also as a result of the high data processing rates offered by such hardware accelerator.

6 Conclusion

This paper presented a high throughput and scalable unified architecture for the computation of the transforms defined in the H.264/AVC standard. Such processing structure is based on a 2-D systolic array of specialized PEs and on a memory free transpose circuit, which are used to compute the mandatory 4×4 and 2×2 forward and inverse transforms using the row-column decomposition approach. Due to the high efficiency of the developed pipelined array of PEs, both in terms of performance and hardware cost, the proposed multi-transform architecture can achieve high data processing rates using very few hardware resources. Moreover, the flexibility of this structure also confers an easy scalability in terms of the number of PEs within the array, in order to adapt the architecture to the specific requirements of the target application in terms of performance, hardware cost and power consumption. Experimental results concerning the implementation of the proposed architecture in a Xilinx Virtex-5 FPGA device operating at about 300 MHz demonstrated that it can process video sequences with resolutions up to Ultra High Definition Video (UHDV) in real-time (38 fps). In addition, such results also revealed that by implementing this specialized processing structure as a hardware accelerator in a multi-core embedded system it is possible to speedup the transform computation task by about $120\times$.

Acknowledgments This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under the project HELIX: Heterogeneous Multi-Core Architecture for Biological Sequence Analysis with reference PTDC/EEA-ELC/113999/2009 and project PEst-OE/EEI/LA0021/2011 and by the Programa de apoio à formação avançada de docentes do Ensino Superior Politécnico (PROTEC) program funds under the research grant SFRH/PROTEC/50152/2009. It was also partially supported by the Spanish Government under project *DREAMS: Dynamically Reconfigurable Embedded Platforms for Networked Contextaware Multimedia Systems* (TEC2011-28666-C04-04), and by the 7th Framework Program of the HiPEAC European Network of Excellence.

References

1. Agostini, L., Porto, M., Guntzel, J., Porto, R., Bampi, S.: High throughput FPGA based architecture for H.264/AVC inverse transforms and quantization. In: 49th IEEE International Midwest Symposium on Circuits Systems, vol. 1, pp. 281–285 (2006)
2. Azevedo, A., Meenderinck, C., Juurlink, B., Terechko, A., Hoogerbrugge, J., Alvarez, M., Ramirez, A.: Parallel H.264 decoding on an embedded multicore processor. In: 4th International Conference on High Performance Embedded Architectures and Compilers, pp. 404–418. Springer, Berlin, Heidelberg (2009). doi:10.1007/978-3-540-92990-1_29
3. Bertels, K., Sima, V., Yankova, Y., Kuzmanov, G., Luk, W., Coutinho, G., Ferrandi, F., Pilato, C., Lattuada, M., Sciuto, D., Michelotti, A.: Hartes: hardware-software codesign for heterogeneous multicore platforms. IEEE Micro. **30**(5), 88–97 (2010). doi:10.1109/MM.2010.91
4. Chaoui, J., Cyr, K., Giacalone, J.P., de Gregorio, S., Masse, Y., Muthusamy, Y., Spits, T., Budagavi, M., Webb, J.: OMAP: enabling multimedia applications in third generation (3G) wireless terminals. In: White Paper: Extensible Processing Platform. Texas Instruments (2000)
5. Cheng, C., Parhi, K.: A novel systolic array structure for DCT. IEEE Trans. Circuits Syst. II **52**(7), 366–369 (2005)

6. Dias, T., Roma, N., Sousa, L., Ribeiro, M.: Adaptive motion estimation processor for autonomous video devices. EURASIP J. Embed. Syst. - Special Issue on Embedded System for Portable and Mobile Video Platforms **57234**, 1–10 (2007)
7. Do, T., Le, T.: High throughput area-efficient SoC-based forward/inverse integer transforms for H.264/AVC. In: Proceedings of 2010 IEEE International Symposium Circuits Systems, pp. 4113–4116 (2010)
8. Fan, C.P.: Fast 2-dimensional 4x4 forward integer transform implementation for H.264/AVC. IEEE Trans. Circuits Syst. II **53**(3), 174–177 (2006). doi:10.1109/TCSII.2005.858748
9. Ho, T., Le, T., Vu, K., Mochizuki, S., Iwata, K., Matsumoto, K., Ueda, H.: A 768 Megapixels/sec inverse transform with hybrid architecture for multi-standard decoder. In: IEEE 9th International Conference on ASIC, pp. 71–74 (2011). doi:10.1109/ASICON.2011.6157125
10. Husemann, R., Majolo, M., Susin, A., Roesler, V., Lima, J.: Highly efficient transforms module solution for a H.264/SVC encoder. In: 2010 IEEE Computer Society Annual Symposium on VLSI, pp. 86–91 (2010)
11. Hwangbo, W., Kyung, C.M.: A multitransform architecture for H.264/AVC high-profile coders. IEEE Trans. Multimed. **12**(3), 157–167 (2010). doi:10.1109/TMM.2010.2041099
12. Jiang, C., Yu, N., Gu, M.: A novel VLSI architecture of 8x8 integer DCT based on H.264/AVC FRext. In: 3rd International Symposium on Knowledge Acquisition and Modeling, pp. 59–62 (2010). doi:10.1109/KAM.2010.5646328
13. JM H.264/AVC Reference Software-version 13.0. http://iphome.hhi.de/suehring/tml/ (2007)
14. Kordasiewicz, R., Shirani, S.: Hardware implementation of the optimized transform and quantization blocks of H.264. In: 2004 Canadian Conference Electrical and Computer Engineering, vol. 2, pp. 943–946 (2004)
15. Kung, S.Y.: VLSI Array Processors. Prentice Hall, Englewood Cliffs (1988)
16. Lee, S., Cho, K.: Design of high-performance transform and quantization circuit for unified video CODEC. In: 2008 IEEE Asia Pacific Conference Circuits and Systems, pp. 1450–1453 (2008)
17. Li, J., Ahamdi, M.: Realizing high throughput transforms of H.264/AVC. In: 2008 IEEE International Symposium on Circuits Systems, pp. 840–843 (2008)
18. Ling-Zhi, L., Lin, Q., Meng-Tian, R., Li, J.: A 2-D forward/inverse integer transform processor of H.264 based on highly-parallel architecture. In: 4th IEEE International Workshop on System-on-Chip for Real-Time Applications, pp. 158–161 (2004). doi:10.1109/IWSOC.2004.1319870
19. Liu, Z., Wang, D., Ikenaga, T.: Hardware optimizations of variable block size Hadamard transform for H.264/AVC FREXt. In: 16th IEEE International Conference Image Processing, pp. 2701–2704 (2009)
20. Lo, C.C., Tsai, S.T., Shieh, M.D.: Reconfigurable architecture for entropy decoding and inverse transform in H.264. IEEE Trans. Consum. Electron. **56**(3), 1670–1676 (2010). doi:10.1109/TCE.2010.5606311
21. Minasyan, S., Astola, J., Guevorkian, D.: On unified architectures for synthesizing and implementation of fast parametric transforms. In: 5th International Conference on Information, Communications and Signal Processing, pp. 710–714 (2005). doi:10.1109/ICICS.2005.1689140
22. Momcilovic, S., Roma, N., Sousa, L.: Multi-level parallelization of advanced video coding on hybrid CPU+GPU platforms. In: International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (2012)
23. Nadeem, M., Wong, S., Kuzmanov, G.: An efficient realization of forward integer transform in H.264/AVC intra-frame encoder. In: International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (2010)
24. Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T., Wedi, T.: Video coding with H.264/AVC: tools, performance and complexity. IEEE Circuits Syst. Mag. **4**(1), 7–28 (2004)
25. Richardson, I.E.: The H.264 Advanced Video Compression Standard. 2nd edn. Wiley, New York (2010)
26. Rodrigues, A., Roma, N., Sousa, L.: p264: open platform for designing parallel H.264/AVC video encoders on multi-core systems. In: 20th International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 81–86. ACM, New York, NY, USA (2010). doi:10.1145/1806565.1806586.
27. Sihvo, T., Niittylahti, J.: Row-column decomposition based 2D transform optimization on subword parallel processors. In: 2005 International Symposium on Signals, Circuits and Systems, vol. 1, pp. 99–102 (2005). doi:10.1109/ISSCS.2005.1509860

28. Tasdizen, O., Hamzaoglu, I.: A high performance and low cost hardware architecture for H.264 transform and quantization algorithms. In: 13th European Signal Processing Conference, pp. 4–8 (2005)
29. Wahid, K., Martuza, M., Das, M., McCrosky, C.: Resource shared architecture of multiple transforms for multiple video codecs. In: 24th Canadian Conference on Electrical and Computer Engineering, pp. 947–950 (2011). doi:10.1109/CCECE.2011.6030599
30. Wang, K., Chen, J., Cao, W., Wang, Y., Wang, L., Tong, J.: A reconfigurable multi-transform VLSI architecture supporting video codec design. IEEE Trans. Circuits Syst. II **58**(7), 432–436 (2011). doi:10.1109/TCSII.2011.2158265
31. Wei, C., Hui, H., Jinmei, L., Jiarong, T., Hao, M.: A high-performance reconfigurable 2-D transform architecture for H.264. In: 15th IEEE International Conference on Electronics, Circuits and Systems, pp. 606–609 (2008)
32. Wiegand, T., Sullivan, G., Bjntegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 560–576 (2003)
33. Wolf, W.H.: Hardware-software co-design of embedded systems. In: IEEE, pp. 967–989 (1994)
34. Xilinx Inc.: ML505/ML506/ML507 Evaluation Platform User Guide v3.1.2 (2011)