High Throughput and Scalable Architecture for Unified Transform Coding in Embedded H.264/AVC Video Coding Systems

Tiago Dias[‡§†], Sebastian Lopez[††], Nuno Roma [‡†], Leonel Sousa [‡†]
[‡]INESC-ID Lisbon, [§]ISEL–PI Lisbon, [†]IST–TU Lisbon, [††]IUMA–ULPGC
Rua Alves Redol, 9
1000-029 Lisbon, Portugal
Email: {Tiago.Dias, Nuno.Roma, Leonel.Sousa}@inesc-id.pt, seblopez@iuma.ulpgc.es

***Abstract*—An innovative high throughput and scalable multi-transform architecture for H.264/AVC is presented in this paper. This structure can be used as a hardware accelerator in modern embedded systems to efficiently compute the 4×4 forward/inverse integer DCT, as well as the 2-D 4×4 / 2×2 Hadamard transforms. 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-4 FPGA demonstrate the superior performance and hardware efficiency levels provided by the proposed structure, which presents a throughput per unit of area at least $1.8\times$ higher than other similar recently published designs. Furthermore, such results also showed that this architecture can compute, in real-time, all the above mentioned H.264/AVC transforms for video sequences with resolutions up to UHDV.**

***Index Terms*—Video coding, H.264/AVC, Unified transform kernel, Scalable architecture, FPGA.**

I. Introduction

A. Motivation

Modern embedded systems differ from their classical counterparts in several different features, being the available computational power and the target application areas two of the most relevant aspects. This shift of trend in the design of embedded systems was motivated not only by all the innovations in VLSI technology, which allowed designing processors with much higher performance levels, but also by all the new software applications that have blossomed throughout the last few years, in order to cope with the ever increasing and more demanding requirements of modern users [1]. In this scope, multimedia applications, especially video coding, are the ones that have been attracting the most users' attention.

Distinct approaches can be adopted by system designers to comply with the performance and flexibility requirements of these applications. For instance, homogeneous multi-core architectures with a significant amount of processing cores have been widely adopted in the General Purpose Processor (GPP) market. However, in the design of embedded systems alternative solutions must be considered, owing to the strict constraints of such systems in terms of power consumption and cost. Hence, a heterogeneous multi-core architecture is commonly adopted, where a few moderate complexity and very efficient GPPs (typically two and no more than four) are included to support the implementation of both the less complex and the most irregular algorithms and operations of the considered applications [2]. In addition, several different specialized hardware accelerators are also usually embedded in these structures, so as to optimize the implementation of the most critical parts of such applications (i.e., reduction of the processing time, increase the energy efficiency, etc.).

Nonetheless, the more specific and optimized nature of this class of architectures also presents some significant drawbacks, even when very successful chip designs are achieved. Namely, the inherent immutable property of VLSI implementations prevents the adjustment and the modification of such embedded systems after its fabrication, in order to being able to either improve its performance or comply with other new features and requirements introduced by modern applications of the same class (e.g., in video coding this can result from the addition of new extensions to a video standard or from the proposal of a new video standard). Consequently, a new design usually outcomes for such embedded systems as a result of this need to improve the original one. However, this implicates huge costs due to the involved design time, the underlying process fabrication techniques and all the extra hardware circuits that are included to support the new functionalities. To minimize these costs, alternative implementation platforms with reconfigurable capabilities must therefore be considered.

B. Current trends in embedded systems design

Current FPGAs already fit incredibly well into the near future design trends [3]. Firstly, due to the increased hardware resources that these platforms have made available to system designers, thus significantly increasing the available computational power. Secondly, because its classical power consumption problematic is no longer a quite so serious issue,

This work was supported by the Portuguese Foundation for Science and for Technology (INESC-ID multiannual funding) through the PIDDAC Program funds and under the research project HELIX: Heterogeneous Multi-Core Architecture for Biological Sequence Analysis (PTDC / EEA-ELC / 113999 / 2009), by the PROTEC Program funds under the research grant SFRH / PROTEC / 50152 / 2009, by the Spanish Government under project *DR.SIMON: Dynamic Reconfigurability for Scalability In Multimedia Oriented Networks* (TEC2008-06846-C02-02), and by the 7^{th} Framework Program of the HiPEAC European Network of Excellence.

978-1-4577-0801-5/11/$26.00 ©2011 IEEE

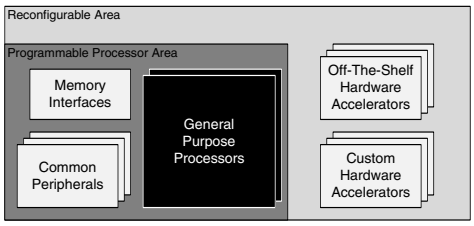


Fig. 1. Generic block diagram of current reconfigurable embedded systems.

as a result of all the progresses in the circuit fabrication process that FPGA vendors have registered in the last few years. Thirdly, owing to the especially relevant maturing of the tools and of the procedures involved in the design flow of this class of devices, which nowadays makes it quite easy to implement an efficient hardware/software co-design approach when developing new System-on-Chips (SoCs). Lastly, due to the existence of very large libraries of IP cores and GPPs that can be used by system designers to compose any given SoC in a quite reduced amount of time and with minimum effort.

For all the former reasons, many companies in the embedded systems market are already developing their next line of products targeting FPGA devices. Furthermore, the recently established joint ventures between the two leading supplier companies of FPGA devices (Xilinx and Altera) and the leading companies in the development of GPPs for the embedded systems (ARM) and the PC (Intel) markets (i.e., the Xilinx/ARM Extensible Processing Platform [4] and the Intel /Altera Atom processor E6x5C series [5], respectively) proves that the future trend in embedded systems design will definitely be based on platforms composed by the traditional GPP, its typical companion circuits (memory interfaces, common peripherals, etc.) and of dynamically reconfigurable logic areas for the integration of custom processing cores, as depicted in Figure 1.

This will help system designers to develop embedded systems that are capable of dynamically adapting themselves to the requirements of the considered applications and the system status, not only in terms of its frequency of operation and switching down capabilities but also regarding to its hardware configurations (e.g., changing the amount, the type and the internal structure of its processing cores). In this scope, extra effort will have to be put by designers for the development of the future hardware accelerators, which will have to present modular and scalable architectures in order to being possible to efficiently reconfigure the system in run-time.

C. H.264/AVC requirements

In what concerns to multimedia systems for video coding, and in particular to the ones implementing the H.264/AVC standard (currently supported by almost all classes of consumer electronics devices), several different modules of both the video encoding and decoding algorithms will necessarily have to be mapped into specialized hardware accelerators. Such requirement results from two basic needs: achieving real-time operation and attaining efficient system implementations in terms of hardware cost and power consumption [6]. In this sense, the transform coding module is especially attractive for implementation in hardware accelerators, due to being a fundamental and mandatory operation that is present in both the video encoder and decoder processing paths. Furthermore, the transform coding algorithms also significantly influence the performance of H.264/AVC encoding and decoding systems, due to its high computational cost and involved latency [7]. Such constraints mostly result from the multi-transform nature and the finer granularity of the transform coding tool used in H.264/AVC, which supports block sizes of 8×8, 4×4 and 2×2 residue values and implements six different transforms: 8×8 and 4×4 forward and inverse integer Discrete Cosine Transform (DCT), 4×4 and 2×2 Hadamard transforms.

All these transforms consist of simplified integer versions of the type-II DCT kernel used in previous video standards, where the scaling and normalization factors were transferred to the quantization stage of the encoding algorithm. Hence, its implementation is based on very few different coefficient values and can be realized using only integer additions and shifts. Nonetheless, they are used in distinct coding scenarios. The 4×4 integer DCT is applied to all 4×4 blocks of residual data resulting from both the motion-compensated prediction and the intra-prediction stages. Then, a Hadamard transform is used to encode the DC coefficients of the two 2×2 chroma blocks of a macroblock. For macroblocks being encoded using the 16×16 intra-prediction mode, which is employed to better exploit the redundancies in smoother areas of a picture, a 4×4 Hadamard transform is further used to transform the sixteen DC coefficients of all the luma blocks within a macroblock. Additionally, in the High Profile an 8×8 integer DCT can further be used to encode each residual block of data. Equations 1 to 4 depict the transform coefficients for the 4×4 integer DCT forward (C_f) and inverse (C_i) transforms, for the 4×4 Hadamard transform ($H_{4\times4}$) and for the 2×2 Hadamard transform ($H_{2\times2}$), respectively.

$$C_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad (1) \qquad C_i = \begin{bmatrix} 1 & 1 & 1 & \frac{1}{2} \\ 1 & \frac{1}{2} & -1 & -1 \\ 1 & -\frac{1}{2} & -1 & 1 \\ 1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} \quad (2)$$

$$H_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad (3) \qquad H_{2\times2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (4)$$

From the previous discussion, it can be easily concluded that efficient, scalable and multi-transform (i.e., unified) architectures are required for the development of future 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 performance, the power consumption and the hardware requirements of video coding applications. In accordance, this paper presents a high throughput and scalable architecture for unified transform coding in H.264/AVC that is able to fulfill all of the above requirements.

The rest of this paper is organized as follows. Section II

briefly reviews already presented related works. Section III introduces the proposed processing structure for unified transform coding in H.264/AVC. Experimental results considering an implementation of such architecture in a Xilinx Virtex4 FPGA device are presented and discussed in Section IV. Finally, Section V concludes the presentation.

II. Related Work

In the last few years, several different specialized and dedicated architectures for transform coding in H.264/AVC have been proposed in the literature [8], [9], [10], [11], [12], [13], [14]. However, such proposals mostly consist of efficient VLSI designs implementing fast and optimized algorithms for transform coding, in order to mitigate the involved computational complexity. They can be classified either as *dedicated transform kernels* or *unified transform kernels*.

Dedicated transform kernels implement a single forward or inverse H.264/AVC transform function. Typically, fast direct two-dimensional (2-D) transform architectures are used when high performance implementations are being considered [8]. This type of structures usually implements fast transform algorithms and involves huge amounts of hardware resources, which are required to assure the parallel processing of all the transform coefficients. Consequently, they are often characterized by having significantly high hardware costs and power consumption values. Moreover, the typical cascaded processing design of these architectures (with long depth processing data paths), further characterizes them as being high 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 approach. In this approach, a single one-dimensional (1-D) transform architecture with either a transpose memory or a transposition switch is used to compute the considered 2-D transform function [9]. Such structures greatly improve the level of hardware efficiency, since the same 1-D transform processing circuit is used twice in the computation of the 2-D transform function. In addition, they also diminish the power consumption requirements and the hardware cost of 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 a transposition memory. Moreover, this circuit also introduces some delay in the processing of the 2-D transform coefficients that, again, can be critical to achieving real-time operation. On the other hand, the highly optimized implementations of both the 2-D and the 1-D transform kernels supporting this class of architectures prevents any modifications to its base structure, and thus its usage as efficient hardware accelerators in future run-time reconfigurable, or scalable, embedded systems.

Unified transform kernels are another class of architectures for transform coding that are capable of implementing multiple transform functions [10], [12]. The supported transform functions can be computed either in parallel or at distinct time instants, and using transform engines with both direct 2-D architectures and 1-D architectures based on the row-column 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 [11]. For both cases, there were also a couple of reconfigurable solutions being recently proposed in the literature [13], [14]. Nevertheless, not so many unified transform architectures have been proposed, and in fact most of these designs present significant limitations:

- 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 its 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 its 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 reality, the majority of these architectures consist of pseudo-reconfigurable structures with fixed hardware functional blocks that can have its functionality reprogrammed in run-time.

To simultaneously comply with all these requirements, a different category of architectures is now proposed based on a systolic array structure. This type of architectures has proved to provide rather efficient solutions in terms of performance, hardware efficiency, scalability and power consumption [15]. Moreover, it is also especially adequate to implement regular algorithms with high data throughput and computational rates. 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 [16]. 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.

III. Scalable Unified Transform Kernel

The H.264/AVC transform algorithms are characterized by their high regularity and low complexity, as it can be seen from Equation 5 that represents a generic H.264/AVC transform function, where X, Y and C denote the input data, the output data and the transform kernel, respectively.

$$Y_{ij} = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} C_{ik} X_{lk} C_{lj}^{T}, \quad i,j = 0, ..., N-1 \qquad (5)$$

Nevertheless, the implementation of these algorithms also demands high data throughput and computational rates, especially for real-time operation. Moreover, in order to improve the hardware efficiency levels and to better adapt the architecture to the requirements of the target application in terms

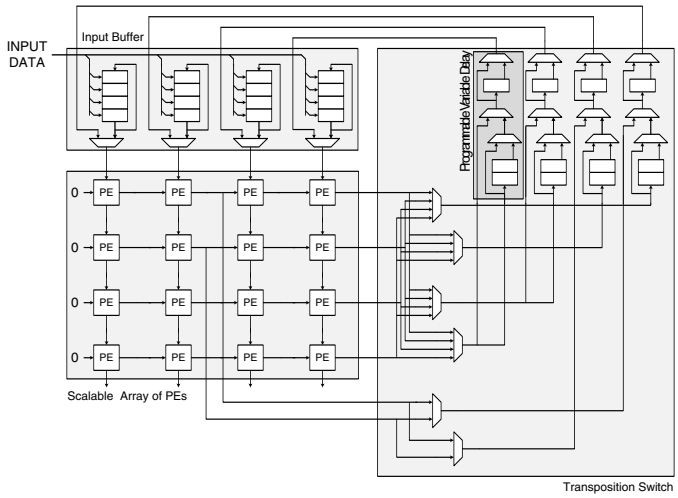


Fig. 2. Block diagram of the proposed scalable unified transform kernel.

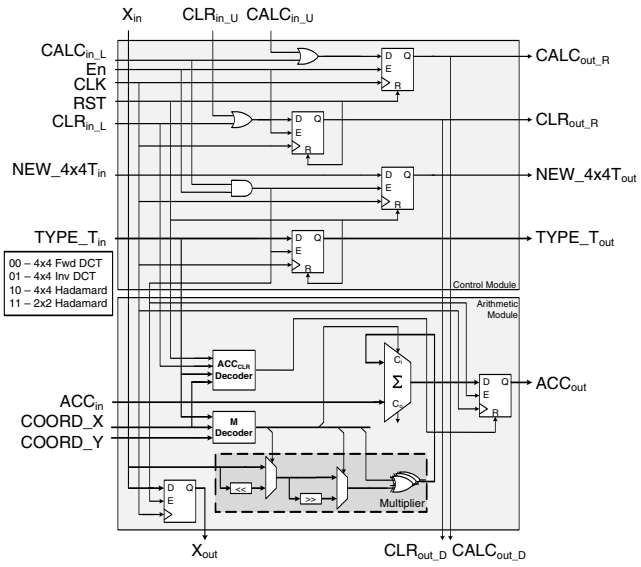


Fig. 3. Architecture of the PEs composing the processing array.

of performance, hardware efficiency and power consumption, such algorithm implementations are required to present highly flexible scalability characteristics. All together, these are highly critical factors in the design of efficient hardware structures implementing this class of algorithms. Still, it will be demonstrated that the challenges they present to system designers can be easily overcome with the usage of systolic array architectures, and by following a methodology entirely similar to the one that is described in [15].

To design the processing structure that is now proposed, three different techniques were applied to achieve a viable hardware scalable and performance efficient implementation of a unified transform architecture for H.264/AVC. Firstly, the base algorithm, depicted in Equation 5, was *decomposed in two sub-parts*, so as to follow a row-column decomposition approach. As it was previously mentioned, this approach not only reduces the hardware cost of the circuit implementing the 2-D transform algorithm, which consists only of a simpler 1-D transform kernel, but also significantly increases the utilization rate of its Processor Elements (PEs). Moreover, it also provides significant savings in terms of power consumption owing to its more reduced hardware cost, without greatly compromising the performance requirements. Then, *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 programmable and dedicated architecture of the PEs that integrates the proposed array was developed, in order to improve even more the hardware efficiency rate of such structure and to allow the computation of all the 4×4 and 2×2 H.264/AVC transforms. In the following subsections it is presented the main features, the dataflow and the scalability properties of the proposed architecture, whose block diagram is depicted in Figure 2.

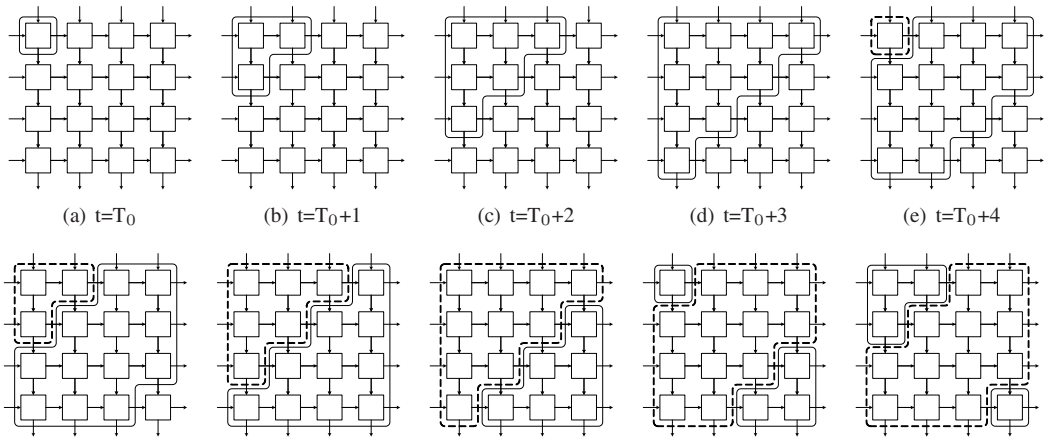
A. Architecture

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 Figure 2, i.e., an *array of PEs*, a *row-column transposition switch* and an *input buffer*.

The PE array is used to compute the 1-D transform functions corresponding to the row-column decomposition of the transform algorithms. In its base configuration, illustrated in Figure 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 top rows of acpPE. 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 transform functions) at the cost of a small hardware overhead (the two 2×1 multiplexers depicted in the lower end of the transposition switch shown in Figure 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 Figure 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.

As it can be seen in Figure 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* using an accumulator and a specialized multiplier, which has its multiplier programmed according to the type of transform being implemented. In this scope, the data values to be processed (`X_in`) are first 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 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



(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

Fig. 4. Data flow in the PE array for one 4×4 data block.

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 a specific multiplier value (i.e., 1, -1, 2, -2, $\frac{1}{2}$ or $-\frac{1}{2}$). This value is also stored in another internal standing-data register before being propagated to the next PEs, in order to shorten the critical path of the circuit.

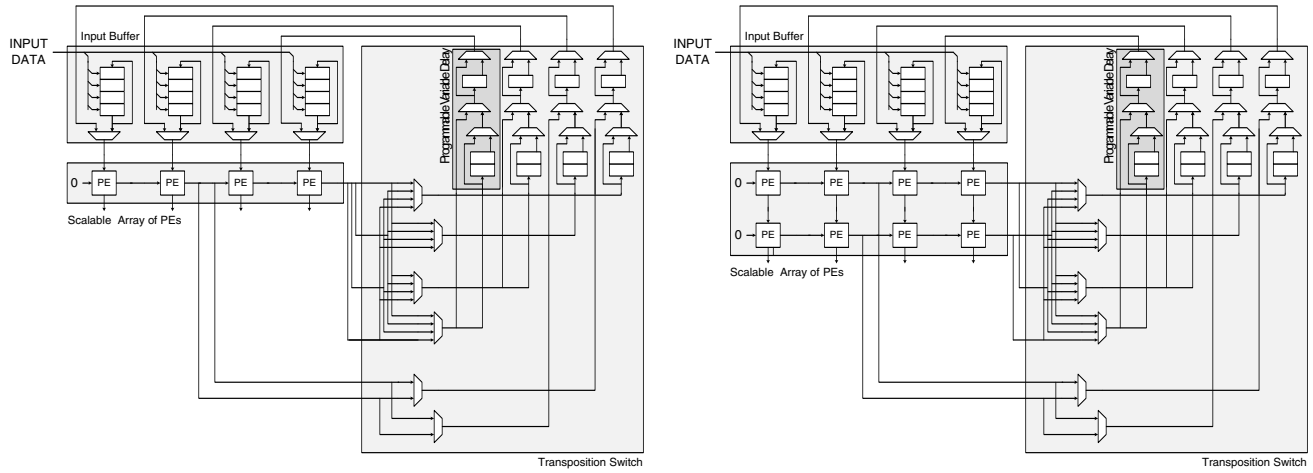
On the other hand, the *control module* is responsible for generating the control signals required to command the operation of the PEs, as well as by guaranteeing the correct flow of such signals for the row-column transposition switch. For example, the control module of each PE manages the propagation 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. Likewise, the PE operation is controlled by the system global enable signal (EN) and the local enable signal generated by the system control module. It should be noted that 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 the ones lacking the data to be processed (for example, when the input buffers are empty).

Altogether, this set of signals provides the necessary mechanisms to guarantee the desired dataflow within the systolic processing array. Such dataflow is illustrated in Figure 4 for the processing of a complete 1-D 4×4 transform. Three data-sets are represented in this figure, corresponding to the processing of two consecutive 4×4 blocks: two data-sets of either 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. As it can be seen from Figure 4, the proposed structure has a maximum data throughput rate of 4 data values per clock cycle and a minimum latency of 4 clock cycles. Moreover, it can also be seen that full pipelined processing is possible to be attained. In such cases, a 2-D 4×4 transform can be computed in 14 clock cycles.

The *transposition switch* is used to implement a hardwired row-column transposition of the data processed by the 1-D transform kernel, and unlike most of the existing efficient transposition units it 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 top two 4×1 multiplexers and the lower two 2×1 multiplexers). This innovative design allows to save significant hardware resources for the implementation of the transposition switch, since it avoids the usage of the typical memory companion cells. Moreover, there is no penalty in the performance of the whole circuit owing both to the pipelined processing nature of the PE array, and the much lower propagation time of the transposition switch circuits regarding to the data processing circuits of the proposed PEs. Furthermore, the proposed transposition switch module also includes some programmable Variable Delay Elements (VDEs) to support the row-column transposition operation when other configurations of the array with fewer PEs are considered, as it is shown in section III-B. Again, such programmable VDEs do not influence the circuit performance because they merely extend the PE array pipeline into the transposition switch.

Finally, the *input buffer* is used to feed the PEs of the systolic array either with the residue data from the intra- and inter-predictions for video encoders, or with the transform coefficient values for 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



(a) Setup with 1×4 PEs (b) Setup with 2×4 PEs

Fig. 5. Alternative setups for the proposed scalable unified transform kernel.

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 (and the transform coefficients, in the decoder), these data elements are sent in a serial fashion to the several columns of the array, in order to comply with its pipelined dataflow.

B. Scalability

In what concerns the offered 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 section III-A, offers the highest performance levels. However, it also imposes the highest hardware cost and power consumption values. 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 encoders.

Conversely, the 1×4 setup offers the greatest savings in terms of hardware resources and power consumption, due to the usage of only a single line with four columns of PEs, as it can be seen in Figure 5(a). However, this comes at the cost of a significant reduction in terms of computation 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 low power consumption constraints or that do not require very high performance levels.

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

It is important to note that independently of the adopted setup for the array of PEs, the hardware structure of all the remaining functional modules of the proposed architecture remains unchanged. Extending the scalability property to the remaining modules of the architecture does not bring any advantage: it not only significantly increases the complexity of the control module, but also involves the reconfiguration of very fine grain hardware structures that is a quite inefficient task. Hence, to comply with the more static nature of the internal structures of the input buffer and transposition switch modules, some programmable interconnection structures and auxiliary processing elements were embedded in the proposed architecture. Namely, 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. Furthermore, the row-column transposition switch includes also programmable VDEs to allow the realization of the transposition operation for all the three PE array setups. 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. Figure 5 highlights these delay elements, which mostly consist of a set of standing data registers and programmable bypass multiplexers.

The reconfiguration of the PE array to implement any of the three presented setups is commanded by the main control module of the reconfigurable system. Such command can be triggered in several different scenarios, such as: *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.

IV. Implementation and Results

The proposed architecture was fully described using the IEEE-VHDL hardware description language. Moreover, to comply with the imposed scalability requirements, the implemented parameterizable description makes extensive use of *generic* configuration inputs and follows a strict modular approach, using independent and self-contained functional blocks.

The resulting description was used to implement the three setups of the proposed processing structure (described in section III-B) in a general purpose Virtex4 XC4VLX100 FPGA device using the Xilinx ISE 10.0i synthesis tool, in order to assess their individual performance levels and hardware cost. Table I presents the obtained results for such implementations by considering a synthesis procedure targeting speed optimized circuits, which demonstrates the several advantages offered by the proposed multi-transform architecture.

In terms of implementation requirements, the data presented in Table I expresses the very low hardware cost of the proposed architecture. In fact, the high 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 unique transform function, as it can be seen in Table II. For the subset of transform architectures with FPGA implementations that were reviewed [10], [11], [17], [18], this table shows the hardware cost in terms of slice-count, the maximum clock rate, the latency in Clock Cycles (CCs) and the data throughput rate (computed as the number of transform coefficients processed in one CC). Moreover, Table II reports only data concerning the implementation of the architectures' transform coding blocks and without considering the 8×8 transform operations, to facilitate comparisons between the different proposals.

Regarding to the obtained performance levels, the maximum allowed clock frequencies presented in Table I evidence the high processing rate that is offered by the proposed structure. Furthermore, they also reveal that the proposed architecture allows to compute, in real-time, the whole set of forward/inverse 4×4 H.264/AVC transforms for video sequences with resolutions up to 4320×7680 pixels (UHDV). Such conclusion can be easily extrapolated from Table II, which depicts the latency and the data throughput rate values for each of the three setups described in section III-B. In addition, Table II can also be used to compare the performance of the proposed scalable architecture against other existing structures with similar functionality.

A straightforward analysis clearly shows that the proposed structure presents one of the highest computation rates, despite being one of the few architectures with multi-transform capability. Due to the different functionalities offered by all the considered architectures and the diverse FPGA devices 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 Virtex slice).

TABLE I
Implementation results of the proposed scalable architecture in a Xilinx Virtex4 XC4VLX100 FPGA.

Configuration	Slices	LUTs	Max. Freq.
4×4 PEs	891	1438	265 MHz
2×4 PEs	670	1074	264 MHz
1×4 PEs	394	700	268 MHz

As it can be seen in Table II, only the structure previously presented in [17] seems to provide a greater DTUA than the proposed multi-transform architecture. Nevertheless, it is important to observe that such structure only implements the forward 4×4 integer DCT transform 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, knowing that the architecture proposed in [17] 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 being proposed only requires 16 32-bit adders to compute the same 2-D transform function, one might dare to say 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, the DTUA of even the simpler setup of the proposed architecture is at least over 1.2 times higher than that of [10], which is one of the most efficient unified transform architectures among the ones that were reviewed. Such results are mainly owed both to the very low hardware cost and the high clock frequency of the PEs that compose the array, which results in very high throughput rates.

Finally, the data presented in Table II also emphasis 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, and thus power consumption. More specifically, the 2×4 setup allows to reduce the hardware cost in about 25% (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 55%. 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 (1920×1080@30fps).

V. Conclusions

In this paper, a novel high throughput and scalable architecture for unified transform coding in H.264/AVC was presented. Such processing structure makes use of a 2-D array

TABLE II
COMPARISON OF FPGA IMPLEMENTATIONS.

Design	Transform Function		FPGA Device	Slices ($\times 10^3$)	Max. Freq. (MHz)	Latency (CC)	Throughput (Coefs/CC)	DTUA ($\times 10^3$)
[17] *	4×4	Forward	Virtex-2 Pro	0.6	107	1	16	2867.4
[18]	4×4	Inverse	Virtex-2 Pro	4.2	132	64	1	31.4
[10]	4×4	Unified	Virtex-4	2.1	167	8	8	636.2
[11]	4×4	Forward	Virtex-2 Pro	1.6	225	8	8	562.5
Proposed 4×4	4×4	Unified	Virtex-4	0.9	265	8	4	1189.9
Proposed 2×4	4×4	Unified	Virtex-4	0.7	264	16	2	788.5
Proposed 1×4	4×4	Unified	Virtex-4	0.4	268	32	1	680.1

* The reported data concerns only to the transform kernel module.

of PEs and of a memory free transpose circuit to realize the H.264/AVC 2-D 4×4 forward/inverse integer DCT and the 4×4 / 2×2 Hadamard transforms, by following a row-column decomposition. Due to the high efficiency of the developed pipelined PE array in terms of performance and hardware usage, the proposed multi-transform architecture can achieve significant data processing rates with reduced hardware cost. 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, power consumption and hardware cost. Experimental results concerning the implementation of the proposed architecture in a Xilinx Virtex-4 FPGA device operating at 265 MHz revealed that it can process real-time video sequences with resolutions up to UHDV. Moreover, they also demonstrated its superiority over other existing architectures with similar functionality in terms of performance, hardware efficiency and scalability.

REFERENCES

[1] A. Gerstlauer, C. Haubelt, A.D. Pimentel, T.P. Stefanov, D.D. Gajski, and J. Teich, "Electronic system-level synthesis methodologies," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1517–1530, Oct. 2009.

[2] K. Bertels, V. Sima, Y. Yankova, G. Kuzmanov, W. Luk, G. Coutinho, F. Ferrandi, C. Pilato, M. Lattuada, D. Sciuto, and A. Michelotti, "Hartes: Hardware-software codesign for heterogeneous multicore platforms," *IEEE Micro*, vol. 30, no. 5, pp. 88–97, Sept. 2010.

[3] L. Bauer and J. Henkel, *Run-time Adaptation for Reconfigurable Embedded Processors*, Springer, 1st edition, 2010.

[4] K. DeHaven, "Extensible processing platform - ideal solution for a wide range of embedded systems," in *White Paper: Extensible Processing Platform*. Xilinx, Inc., 2010.

[5] "Intel Atom processor E6x5C series-based platform for embedded computing," in *Platform Brief: Intel Atom Processor E6x5C Series*. Intel Corporation, 2010.

[6] A. Azevedo, C. Meenderinck, B. Juurlink, A. Terechko, J. Hoogerbrugge, M. Alvarez, and A. Ramirez, "Parallel h.264 decoding on an embedded multicore processor," in *Proc. of the 4th Int. Conf. High Performance Embedded Architectures and Compilers*, Berlin, Heidelberg, 2009, HiPEAC '09, pp. 404–418, Springer-Verlag.

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

[8] J. Li and M. Ahamdi, "Realizing high throughput transforms of H.264/AVC," in *Proc. 2008 IEEE Int. Symposium Circuits Syst.*, May 2008, pp. 840–843.

[9] Z. Liu, D. Wang, and T. Ikenaga, "Hardware optimizations of variable block size hadamard transform for H.264/AVC FREext," in *16th IEEE Int. Conf. Image Processing*, Nov. 2009, pp. 2701–2704.

[10] T.T.T. Do and T.M. Le, "High throughput area-efficient SoC-based forward/inverse integer transforms for H.264/AVC," in *Proc. 2010 IEEE Int. Symp. Circuits Syst.*, June 2010, pp. 4113–4116.

[11] R. Husemann, M. Majolo, A. Susin, V. Roesler, and J. Lima, "Highly efficient transforms module solution for a H.264/SVC encoder," in *2010 IEEE Computer Society Annual Symp. on VLSI*, July 2010, pp. 86–91.

[12] M. Nadeem, S. Wong, and G. Kuzmanov, "An efficient realization of forward integer transform in H.264/AVC intra-frame encoder," in *Int. Conf. Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, July 2010.

[13] C. Wei, H. Hui, L. Jinmei, T. Jiarong, and M. Hao, "A high-performance reconfigurable 2-D transform architecture for H.264," in *15th IEEE Int. Conf. Electronics, Circuits and Systems*, 2008, pp. 606–609.

[14] C.-C. Lo, S.-T. Tsai, and M.-D. Shieh, "Reconfigurable architecture for entropy decoding and inverse transform in h.264," *IEEE Trans. Consum. Electron.*, vol. 56, no. 3, pp. 1670–1676, Aug. 2010.

[15] S. Y. Kung, *VLSI Array Processors*, Prentice Hall, 1988.

[16] C. Cheng and K.K. Parhi, "A novel systolic array structure for DCT," *IEEE Trans. Circuits Syst. II*, vol. 52, no. 7, pp. 366–369, 2005.

[17] R. Kordasiewicz and S. Shirani, "Hardware implementation of the optimized transform and quantization blocks of H.264," in *2004 Canadian Conf. Elect. Comput. Eng.*, May 2004, vol. 2, pp. 943–946.

[18] L. Agostini, M. Porto, J. Guntzel, R. Porto, and S. Bampi, "High throughput FPGA based architecture for H.264/AVC inverse transforms and quantization," in *49th IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2006, vol. 1, pp. 281–285.