HARDWARE/SOFTWARE CO-DESIGN OF H.264/AVC ENCODERS FOR MULTI-CORE EMBEDDED SYSTEMS

Tiago Dias, Nuno Roma, Leonel Sousa

INESC-ID Lisbon, ISEL–PI Lisbon, IST–TU Lisbon
Rua Alves Redol, 9
1000-029 Lisbon, Portugal

ABSTRACT

This paper presents a multi-core H.264/AVC encoder suitable for implementations in small and medium complexity embedded systems. The proposed structure results from an efficient hardware/software co-design methodology, where the encoder software application is highly optimized and structured in a very modular and efficient manner, so as to allow its most complex and time consuming operations to be offloaded to dedicated hardware accelerators. The considered methodology adopts a simple and efficient core interconnection mechanism to easily allow the inclusion and the removal of such optimized processing cores. Experimental results obtained with the implementation in a Virtex4 FPGA of an H.264/AVC encoder using an ASIP IP core as a ME hardware accelerator have proven the advantages of this methodology. For the considered system, speedup factors greater than 15 were obtained with a very modest increase of the involved hardware resources.

Index Terms— Hardware/software co-design, Multi-core, Embedded systems, Video coding, H.264/AVC

1. INTRODUCTION

The H.264/AVC standard is nowadays the *de-facto* standard for video applications, due to its high coding efficiency and flexibility[1]. Such powerful characteristics are mostly owed to: *i)* the extensive set of novel coding tools introduced in H.264/AVC, such as inter-prediction using multiple reference frames and variable block sizes, quarter-pixel precision for the Motion Estimation (ME) and Motion Compensation (MC) techniques, several intra-prediction modes, improved 4×4 integer transform, or entropy coding techniques; *ii)* the several combinations of profiles and levels that are made available by this standard, which allow to select the most suitable encoder configuration for a given application.

This work was supported by the Portuguese Foundation for Science and for Technology (INESC-ID multiannual funding) through the PIDDAC Program funds and by the PROTEC Program funds under the research grant SFRH / PROTEC / 50152 / 2009.

For the most typical applications usually implemented in small and medium complexity embedded systems, such as video telephony, video conferencing or low resolution video recording (e.g.: web cameras), the *baseline* and *main* profiles are typically adopted. This design option allows discarding the most compute intensive and memory consuming operations of the video encoder, and therefore to fine tune the video encoder implementation to the more modest computational and memory capabilities offered by this class of devices. Nevertheless, the increased computational complexity of the coding tools that are considered in these profiles still imposes very tight constraints to such implementations. Consequently, the development of embedded video coding systems supporting the above mentioned applications usually results from an efficient hardware/software co-design approach.

In this procedure, both the hardware and the software are designed together in order to make sure that the target application functions properly and meets the performance goals. Typically, in modern multimedia embedded systems the hardware consists of a heterogeneous multi-core structure populated with one General Purpose Processor (GPP) and multiple specialized hardware accelerators that implement the most critical operations. The partitioning and load balancing issues are addressed by the software component, which is designed and compiled onto the target processing structure in the most optimized way as possible. By following this hardware/software co-design approach it becomes possible to achieve the intended global system performance, provided that the application execution can be efficiently parallelized and that the hardware accelerators offer effective speedup values.

In the latter years, several dedicated and programmable hardware structures have been proposed as hardware accelerators for different operations of the H.264/AVC video encoder [2, 3, 4]. The reported speedup values for such structures are generally high, and quite often very impressive. Nevertheless, they only focus on the implementation of the considered algorithm and on its execution within the hardware structure, disregarding the problem of embedding the accelerator in the overall multi-core hardware structure.

Such integration task consists of two different aspects. On the one hand, it addresses the interconnection of the hardware accelerator with the multi-core structure, which may require an extra effort to develop supplementary circuitry so as to adapt the accelerator interface to the existing multi-core communication structure. In fact, this is a quite frequent task, owing to the fact that hardware accelerators typically present custom and optimized specific interfaces. On the other hand, it considers the development of an efficient software device driver for the hardware accelerator, in order to being possible to have fast, low latency and sustained communication with such processing core within the multi-core structure.

Although core integration might seem like an implementation specific problem, its impact on the effective speedup values provided by the hardware accelerators is very significant, since it greatly restricts the parallel execution of the encoder tasks. Such performance impact in the global system speedup (S) can be evaluated using Eq. 1, which presents Amdahl's law taking into consideration not only the speedup provided by the hardware accelerator (S_{HA}) and its contribution to the whole system (F), but also the parallelization overhead and communication time (κ_0).

$$S = \frac{1}{(1-F) + \frac{F}{S_{HA}} + \kappa_0} \quad (1)$$

To reduce κ_0 and minimize the core integration effort, standardized interconnection and communication structures can be adopted in the design of the multi-core hardware structure. Such design option further allows the reuse of the hardware accelerator in distinct systems, as well as it eases its runtime insertion and removal in the multi-core structure, which is highly relevant for reconfigurable platforms such as the ones supporting Reconfigurable Video Coding (RVC).

Several different bus solutions are typically available for small and medium scale embedded systems that can be used to achieve these goals, such as the ARM Advanced Microcontroller Bus Architecture (AMBA), the IBM Core Connect bus, the Altera Avalon Interface Bus or the OpenCores Wishbone bus. Despite having different architectures and interfaces and supporting distinct protocols, the communication performance levels provided by each of these buses are somewhat similar. Hence, the selection of a particular bus interconnection structure in the implementation of a given embedded system is often a consequence of the bus interface that is already available in the GPP of such system. For this reason, the widespread usage of ARM processors facilitated the gradual emergence of the royalty free AMBA bus has one of the most efficient and flexible bus type solutions in commercial embedded systems. Nevertheless, the specific bus protocols and interconnection structures provided by this bus for both high performance and low power and complex systems have significantly contributed to this end.

This paper presents an efficient hardware/software codesign methodology to develop multi-core H.264/AVC encoders using the AMBA bus. Such methodology addresses not only the realization of a modular and optimized software application for video coding in medium complexity embedded systems, but also the design of the supporting multi-core structure. In addition, the proposed methodology also allows to easily adapt the interface of a custom hardware accelerator to the AMBA bus interface. The experimental results obtained with the implementation of an H.264/AVC encoder using an Application Specific Instruction Set Processor (ASIP) core as a ME hardware accelerator are very promising and demonstrate the advantages of this methodology.

This paper is organized as follows. In Section 2 the hardware/software partitioning issue is discussed and the basic components of the multi-core architecture are introduced. Sections 3 and 4 present the hardware and software components of the proposed H.264/AVC video encoder, respectively. Experimental results are provided in Section 5 and Section 6 concludes the paper.

2. HARDWARE/SOFTWARE PARTITIONING

Partitioning a design into its hardware and software components is the first and one of the most critical tasks in the design process of an embedded system. At the hardware level, this usually consists in the selection (and quite often in the implementation) of the system Central Processing Unit (CPU), memory and peripheral devices, as well as the design of an interconnection network to accommodate all the devices. These are time consuming and error prone tasks that have became increasingly difficult and far too expensive, due to the involved complexity of modern embedded systems and the tight constraints on area, power consumption and performance that system developers face. As a result, modern hardware designs are typically based on pre-designed and pre-tested Intellectual Property (IP) cores to mitigate these issues.

Soft-core processors are HDL models of microprocessors that can be customized and implemented as IP cores. They provide several advantages over custom designed processors, such as reduced cost, increased flexibility and easy customization (to better comply with the target application requirements), platform and technology independency and greater immunity to obsolescence. As a result, the use of soft-core processors has widespread in the embedded systems domain. Several different soft-core processors are nowadays provided by commercial vendors and open source communities, including the Altera Nios II processor, the Xilinx MicroBlaze processor, the Sun OpenSPARC, the Gaisler Leon3 processor and the OpenCores OpenRISC processor.

Despite their general-purpose nature, each of these soft-cores offers different performance characteristics and features that are suitable for some specific applications [5]. For example, the Nios II and the MicroBlaze processors are specifically optimized for FPGA implementations. On the other hand, both the OpenRISC and the Leon3 processors are free soft-

cores with similar performance levels, but that can be implemented both in FPGA and ASIC platforms. Furthermore, the Leon3 processor presents two other characteristics that give it significant advantages over the other soft-cores: *i)* it comes with a vast library of peripheral IP cores that can be used without additional costs to implement embedded systems; *ii)* it makes use of the well-known and extensively used AMBA bus to interface the processor with both on-chip and off-chip memories and other peripherals. As a result, the proposed hardware/software co-design methodology is herein demonstrated using a Leon3 processor as the system CPU and an FPGA device as the prototyping platform. Nonetheless, it can be successfully applied using any other soft-core processor and implementation technology.

In what concerns the software component of the embedded system, it addresses the design of a program to implement the considered application algorithms and to support the communication between all the system hardware components. Such development task is usually realized in two distinct phases. The first phase, which is quite independent of the system hardware component, consists in the description of the application algorithm using a software programming language. This involves the decomposition of the application into elementary functions, which are subsequently converted into functional modules and coded. Then, code profiling tools are usually applied to assess the performance and memory requirements of such functional modules. By using such approach, system designers are able to identify the most critical functional modules of the application during the earlier development stages and promptly optimize them so as to comply with the application specification. For embedded system applications, one of the commonly adopted optimization techniques is in fact the use of hardware accelerators to implement the most complex and time consuming functional modules. On the other hand, the second phase of the software design procedure takes place only after the system hardware component has been completely specified. It concerns not only the coding of the developed algorithms in an especially efficient manner, by taking into consideration the system hardware components and using the most complex and efficient modes of the software compiler tools, but also the data transfers between all hardware components required for the proper system operation. In addition, it also encompasses the joint test of the hardware and software components after its integration to guarantee the desired system performance levels.

In the last few years, several software applications of H.264/AVC encoders have been proposed. Among them, the JM versions of the reference software (or some adaptations of it) are the most commonly used. Moreover, several complexity analyses of the H.264/AVC standard have also been reported in literature [1], which have allowed to identify the most critical functional blocks of a H.264/AVC encoder. The results of such studies, which are summarized in Fig. 1, have demonstrated that the Inter Prediction module is undoubtedly

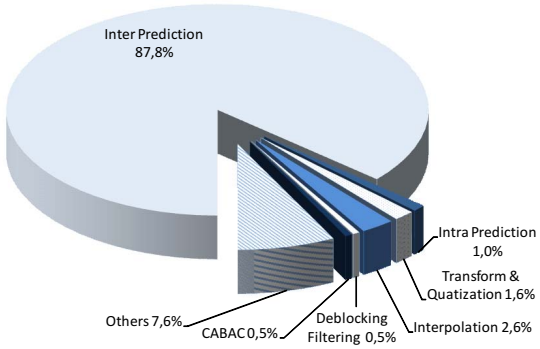


Fig. 1. Relative complexity requirements of the H.264/AVC coding software.

the most time consuming function of a video encoder. Consequently, the proposed hardware/software co-design methodology will be applied to implement a multi-core encoder conformant to the H.264/AVC reference software model (version 14), where the ME operation will be offloaded to a hardware accelerator in order to improve the encoder performance. A Leon3 processor is used to realize all the remaining operations of the encoder. Nevertheless, it is worth noting that the same design methodology can equally be applied when considering different video coding software applications and hardware accelerators.

3. MULTI-CORE PROCESSOR FOR VIDEO CODING

The hardware component of the proposed H.264/AVC encoder consists of a multi-core processing structure optimized for small and medium complexity embedded system implementations. In this system, the CPU consists of a Leon3 soft-core processor that was configured to give support to most operations involved in the video encoder software application. Besides this CPU, the multi-core architecture also includes additional processing units capable of executing other operations of the video encoder in a more efficient manner and in parallel with the CPU, such as the estimation of Motion Vectors (MVs) for Inter Prediction coding or user I/O interfacing. In the following, more detailed descriptions of the Leon3 processor and of the IP core adopted as the system ME processor are presented. The adopted methodology to efficiently integrate such ME hardware accelerator in the proposed multi-core structure is also revealed.

3.1. The Leon3 CPU

The Leon3 processor is one of the most used free processor soft-cores available today [5]. It has been specifically designed for embedded applications by the European Space Agency, although nowadays it is maintained by Gaisler Research. It consists of a highly configurable and fully synthesizable core written in VHDL, implementing a 32-bit RISC architecture conforming to the SPARC v8 definition.

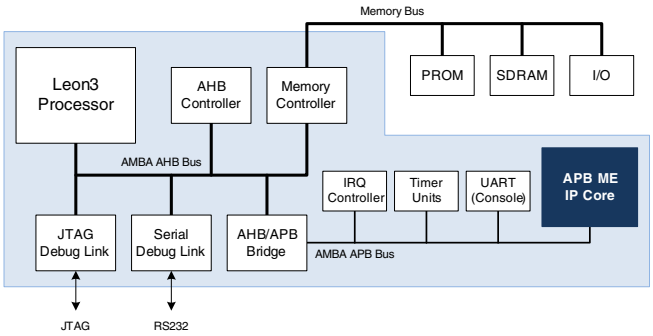


Fig. 2. Block diagram of the conceived multi-core processor for video coding.

This processor soft-core is based on a 7-stage instruction pipeline Harvard micro-architecture with 32-bit internal registers. The core functionality can be easily and greatly extended by means of the AMBA-2.0 AHB/APB on-chip buses. The Advanced High-performance Bus (AHB) is used to connect the Leon3 processor with high-speed controllers, such as the cache and the memory controllers. This programmable memory controller provides interfaces to PROM, SRAM and SDRAM chips, and supports external memory access and memory mapped I/O operation. On the other hand, the Advanced Peripheral Bus (APB) is used to access most on-chip peripherals and is connected to the Leon3 processor via an AHB/APB Bridge.

The Leon3 processor considered in this work is based on version gpl-1.0.20-b3403 of the GRLIB and results from a careful customization process to guarantee the desired performance levels and minimize the hardware resources required for its implementation. Hence, the adopted processor configuration incorporates a hardware divide and multiply unit, an interrupt controller, separate data and instruction associative caches and an SRAM memory controller, all with AMBA-AHB interface. Moreover, it also encompasses the DSU and JTAG controllers for FPGA/PROM programming and debugging purposes, a RS-232 UART for I/O operations and user interaction, two 32-bit timers, and a hardware accelerator to realize the ME operation. All these peripherals are connected to the system AMBA-APB bus, as it can be seen in Fig. 2.

3.2. The ME hardware accelerator

The ME unit of the proposed multi-core video encoder consists of an efficient ASIP IP core that was recently proposed [4]. Such processing structure significantly optimizes the ME operation, by allowing the programming of a broad class of powerful, fast and/or adaptive ME search algorithms. Moreover, its increased flexibility supports the implementation of the most used Macroblock structures in block matching algorithms, i.e., the traditional fixed 16×16 pixels block size, as well as any of the variable block size structures adopted in the H.264/AVC standard [1].

To achieve these goals, this ASIP implements a minimum and specialized instruction set supported by an optimized datapath, that includes a specialized arithmetic unit to efficiently compute the Sum of Absolute Differences (SAD) similarity function for block matching algorithms. The efficiency of this functional unit is owed both to the simplicity of its internal architecture and to the implementation of early-termination techniques in the ME procedure, which allow it to provide for a reduction in the ME computation time of up to 50%. The processor datapath also includes a dedicated Address Generation Unit (AGU), that is responsible for fetching all the pixels involved in the estimation of a MV and to store them in local scratchpad memories. This feature allows exploiting the data locality in the ME procedure and to minimize the communication between the ASIP and the main frame memory of the encoding system. The AGU is also capable of working in parallel with the remaining functional units of the ASIP, thus providing concurrent operation for the block matching and the data transfer tasks, and therefore to further speedup the ME operation.

Regarding to the interface of the ASIP IP core, it presents a quite reduced number of interface signals and uses very simple communication protocols to exchange data and accept commands from external devices. This design feature not only eases the implementation in ASIC of such an IP core, but also permits the straightforward integration of such chips in almost any video coding platform. However, such enormous flexibility also inflicts some penalty in the effective speedup value provided by this specialized processing structure for two main reasons. Firstly, due to the diminished width of the I/O data bus (8-bit wide) that significantly constraints the communication bandwidth to the core. Secondly, due to the use of full interlocked protocols for all communication operations, which greatly delays the data transfer operations. As a result, several modifications must be introduced to the interface of such ASIP IP core, so that it can be efficiently integrated in the proposed multi-core as a ME hardware accelerator.

3.3. Interconnection strategy

Computational demanding embedded systems can have their functionality optimized or extended by making use of hardware accelerators. In the particular case of the Leon3 processor, such accelerators can be integrated in the system architecture either as Leon3 co-processors (a single co-processor is allowed per Leon3 core) or through the AMBA-2.0 AHB and APB on-chip buses. However, the bus integration strategy is often highly preferable, due to the "bus-centric" nature of Leon3 processor systems. Moreover, this strategy also allows a less complex and time consuming integration of existing IP cores, which can be reutilized from other system designs.

The AMBA-2.0 AHB interface addresses the requirements of high-performance designs. It supports multiple bus masters and enables a highly efficient interconnection be-

tween them in a single frequency system. Several different features are provided by this bus specification to guarantee its high-performance, namely: wide data bus configurations (128/64/32 bits), burst transfers, split transactions, single clock edge operation, etc. On the other hand, the AMBA-2.0 APB interface is optimized for minimal power consumption and reduced interface complexity. Hence, it appears as a local secondary bus of the AHB interface to isolate the low bandwidth transactions necessary to access the peripheral configuration registers and all data traffic involving low bandwidth peripherals. Revision 2.0 of the APB specification introduces significant changes that greatly improve the performance and flexibility of this bus interface. More specifically, after revision 2.0 it became easier to achieve much higher operating frequencies for the APB peripherals.

For the particular case of the proposed multi-core hardware structure for video coding either one of the two AMBA-2.0 interfaces can be used. Nevertheless, since a single hardware accelerator is considered in this system and owing to the reduced amount of data to be transferred between the ME IP core and the main encoding system [4], the APB interface outcomes as the optimal choice in terms of integration complexity and performance.

The adaptation of the ASIP IP core introduced in subsection 3.2 to the AMBA-2.0 APB bus was realized in three different steps. Firstly, the programming interface of the APB ME IP core was specified by taking into consideration the existing interface signals of the ASIP and the commands required for its operation. Such interface is depicted in Fig. 3 and encompasses three different sets of memory mapped registers for a complete system interaction with the core. The set/clear control registers and the core status register provide the necessary means to control and evaluate the core operation, respectively. The *Address*, *Data Input* and *Data Output*

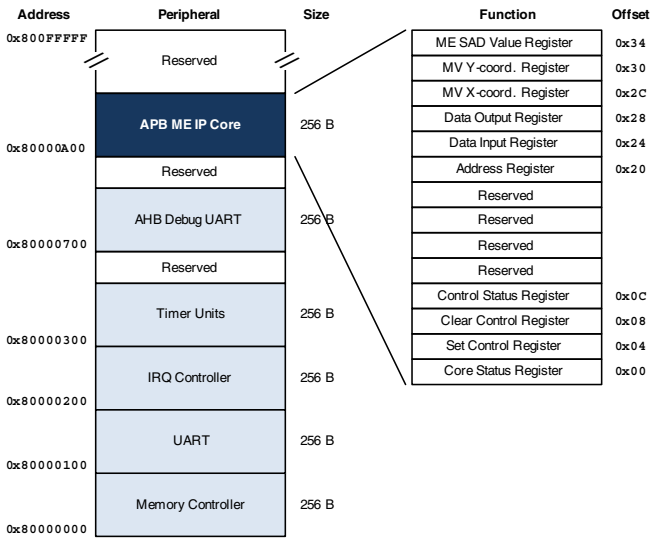


Fig. 3. Multi-core processor memory map for the APB peripherals and the ME IP core user interface.

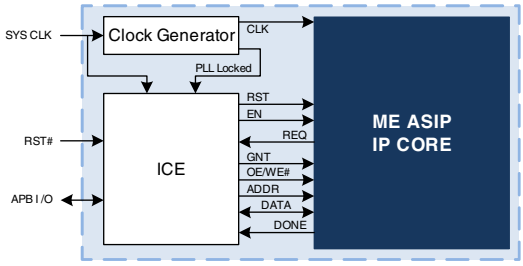


Fig. 4. Block diagram of the APB ME IP core.

registers allow to upload the pixel data and the core firmware, i.e., the ME algorithm implemented by the processor, to the ASIP. Finally, the remaining three 32-bit registers are used to retrieve the most important output results of the ME operation: the best matching MV coordinates and its corresponding SAD value. The memory region identified as *Reserved* in Fig. 3 hides a set of memory mapped registers from the user interface that can be used for debugging the core operation in run-time.

The second step of the implemented interconnection strategy focused on a set of optimizations to the ASIP motion estimator in order to increase its communication bandwidth. In this scope, the ASIP interface had its data bus width upgraded to 32-bits, which also imposed some adjustments in the design of the processor's AGU. Nonetheless, other very important changes were also introduced in the AGU, aiming at some optimizations of the implemented communication protocols. As a result of such modifications, the full interlocked protocols were discontinued and data read/write operations are now supported using plain load/store instructions over the *Data Input* and *Data Output* core interface registers.

The last step in the development of the APB ME hardware accelerator consisted in the design of a hardware circuit to adapt the AMBA APB interface signals to the ASIP IP core interface. Such APB wrapper is composed mostly by hardware registers and simple control logic (e.g.: address decoders and data multiplexers), which support the user programming interface depicted in Fig. 3 and implement the APB bus protocol. The same circuitry also gives support to the run-time debug facility that was previously mentioned, by making available two hardware breakpoints and the usual `step`, `goto` and `run` commands which is why it is identified as *In-Circuit Emulator (ICE)* in Fig. 4. In addition, the conceived APB wrapper also includes a local clock generator module in order to enable the ME hardware accelerator to operate at a different frequency than the system's CPU and AMBA interfaces.

4. THE P264 VIDEO CODING SOFTWARE

The software component of the proposed video encoder is based on a parallel programming platform for H.264/AVC video encoders that has been recently proposed in the litera-

ture: the p264 [6]. The p264 implementation herein described is intended for embedded system applications. Consequently, it presents some modifications to its base software description so as to improve its performance for such class of systems.

In its original distribution, the p264 is a flexible and highly modular parallel framework derived from version JM14.0 of the H.264/AVC Reference Software Model. It aims at providing efficient implementations of video encoders in multi-core systems, while still maintaining full compliancy with the original reference encoder. To achieve such goals, and by comparison with the H.264/AVC Reference Software Model, the p264 presents several improvements and optimizations in its most elementary algorithms and functions that significantly increase its performance levels and reduce the application memory requirements. For embedded system implementations, these are very important issues due to the modest computation and memory capabilities offered by most of these systems.

Among all the modifications that were introduced in the p264, there are two particular aspects that are worth highlighting. First of all, the redesign of all its most important data structures in order to allow the cache access patterns to be efficiently exploited. This task consisted not only in the reorganization of such data structures, but also in its resizing and consequent shrinking, to guarantee maximum exploitation of the spatial and temporal locality properties of caches. The second aspect concerns to the most important code improvements that were performed in p264 to minimize code redundancy and speedup the program execution. In this scope, the several software modules of the encoder were re-organized in a more compact and modular way, which also involved a careful cleaning of the code and reutilization of common functions to discard redundant and pointless time consuming operations. Furthermore, a set of initialization functions was also made available for each software module. Such functions minimize the time consumed by control tasks during the video encoding operation, by pre-computing and gathering all the required data in its local structures. This strategy also included the adaptation of the memory allocation mechanism to static memory allocations.

Other more specific modifications that were also required to be introduced in the p264 software platform in order to better adapt it to the embedded system application realized in this work focused the interaction with the hardware accelerators. More specifically, it consisted in the development of modular and low complexity device drivers for the hardware accelerators, based on a simple yet effective Application Programming Interface (API). Due to the modularity of both the p264 and of the proposed device driver, the developed API allows to easily replace the original software implementation of any functional module of the video encoder application by a system call to its corresponding hardware accelerator. Consequently, a special attention was given in the development of all API functions concerning the data transfers to the hardware accelerator, so as to guarantee the optimal efficiency and maximize the speedup provided by such structure.

Table 1. Implementation results of the multi-core processor in the Xilinx Virtex4 XC4VLX100.

Unit	Leon3 Uniprocessor		ME IP Core		Multi-core Processor	
Slices	7615	15%	1272	2%	8594	17%
LUTs	14458	14%	2325	2%	16309	16%
BRAMs	32	13%	10	4%	42	17%
DCMs	1	8%	1	8%	2	16%
DSP48	1	1%	2	2%	3	3%
Frequency	60 MHz		90 MHz		60/90 MHz	

Table 2. Implementation results of the ME IP core in the Xilinx Virtex4 XC4VLX100.

Unit	ME ASIP		ICE	
Slices	920	1%	558	1%
LUTs	1334	1%	994	1%
BRAMs	10	4%	0	0%
DCMs	0	0%	0	0%
DSP48	2	2%	0	0%
Frequency	90 MHz		675 MHz	

5. IMPLEMENTATION AND EXPERIMENTAL RESULTS

The functionality of the proposed multi-core H.264/AVC video encoder was properly validated in a practical realization based on FPGA technology. At the same time, such implementation was also used to assess the performance gains provided by the presented hardware/software co-design methodology. The considered implementation platform consisted in the GR-CPCI-XC4V development board from Pender Electronic Design. Such development system includes a Virtex4 XC4VLX100 FPGA device from Xilinx, a 133 MHz 256 MB SRAM memory bank, and several peripherals for control, communication and storage purposes. The implemented video encoder makes use of some of these resources, such as the on-board FPGA configuration PROMs, the JTAG interface and the standard RS-232 UART port, which were both used for FPGA/PROM programming and also for I/O and debugging purposes.

5.1. Implementation results

Tables 1 and 2 present the implementation results that were obtained with the realization of the proposed multi-core hardware structure using the Virtex4 XC4VLX100 FPGA device. Such implementation was conducted using the Xilinx ISE 10.0i tools and making use of the configuration and constraints files for the GR-CPCI-XC4V development board contained in the gpl-1.0.20-b3403 version of the GRLIB.

The results depicted in Table 1 evidence the low area usage requirements of the proposed multi-core realization, and thus its suitability for applications on small and medium complexity embedded systems. These results also show the

marginal increase in the implementation area of the multi-core structure owed to the insertion of the ME hardware accelerator, which costs less than 13% of the initial circuit implementation requirements and a couple of extra DSP slices. However, it should be noted that about 40% of the hardware resources required to implement the APB ME IP core concern the implementation of the ICE module, as it can be seen in Table 2. More specifically, almost half of such logic circuitry is used to provide the run-time debug facility. Hence, the total cost of having the ME hardware accelerator in the proposed multi-core structure can be reduced in about 20% for implementations of the APB ME IP core not requiring run-time debug support.

In what concerns the operating frequencies of the two processors, the obtained experimental results reveal that the Leon3 processor can operate with a maximum clock frequency of 60 MHz, while the ME hardware accelerator is capable of operating with clock frequencies up to 90 MHz. As a result, the maximum operating frequency of the proposed video encoder is constrained to 60 MHz, which is a relatively low value considering the typical clock frequencies reported for multimedia applications. Nevertheless, it is important to note that higher operating frequencies are possible to be attained for the Leon3 processor by considering FPGA devices other than the existing in the adopted prototyping platform or even by considering different implementation technologies. As an example, for ASIC implementations the Leon3 processor is capable of operating with clock frequencies as high as 400 MHz [5]. The clock frequency results that are depicted in Table 1 also fully justify the design option of including a clock generator module in the wrapper circuit of the APB ME IP core. With this approach, it becomes possible to implement several ME algorithms with different complexity constraints in the same ME processor, thus allowing a better balance of the required operating frequency and the resulting power consumption.

5.2. Performance analysis

The performance gains provided by the multi-core H.264/AVC encoder were experimentally assessed by encoding a set of benchmark QCIF video sequences with quite different characteristics, both in terms of movement and spatial detail, namely: *bream*, *carphone*, *foreman*, *mobile* and *table-tennis*. Such assessment was accomplished by programming the Full-Search Block Matching (FSBM), the Three-Step Search (3SS) and the Diamond Search (DS) ME algorithms in the ME hardware accelerator and by considering the video coding parameters presented in Table 3. Figures 5 and 6 depict the obtained experimental results.

Figure 5 clearly demonstrates the huge reduction in the computation time of the ME operation that was obtained with the integration of the ME hardware accelerator within the multi-core system, which validates the proposed hardware/software co-design methodology. Such results also validate the usage of the AMBA APB bus as the optimal interface for the considered ME processor. In fact, and despite its low complexity, it provides a negligible communication penalty (κ_0 in Eq. 1). Such conclusion can be easily drawn by comparing the ME computation time required by the APB ME IP core against the transfer times for the pixel data and for the ME results, which are detailed in Table 4 and are about 0% of the total encoding time.

Table 3. Configuration parameters of the implemented video encoding system.

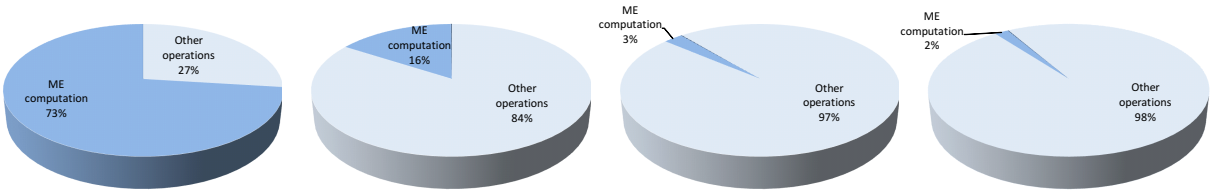
Profile / Level	Baseline / 4.4
GOP Sequence	IP...P
INTRA Prediction Modes	All
INTER Prediction Modes	16×16
Reference Frames	1
ME Algorithm	FSBM
ME Search Range	32×32
ME Precision	Integer-Pixel
ME Error Metric	SAD
Entropy Coding	UVLC
In-loop Deblocking Filter	Enabled
Rate Control Engine	Disabled

By using such data it is therefore possible to compute the speedup values for the ME operation owed to the usage of a ME hardware accelerator. In addition, by considering Eq. 1 as well as the data in Fig. 1 it is possible to estimate the global speedup value that is achieved for the whole video encoding system, and a more precise assessment of the advantages resulting from the proposed hardware/software co-design methodology. Fig. 6 depicts the partial and global speedup values for all the considered ME algorithms, by taking a Leon3 uniprocessor video encoding system as the reference platform.

Although impressive, the speedup values that were achieved for the ME task with the adopted design strategy were somewhat expected due to the performance levels provided by the adopted ME IP core [4]. Nonetheless, despite the significant impact of ME in the global video encoding operation, the global speedup values obtained for the whole video encoder are quite more modest, owing to the low operating frequency of the Leon3 processor. As a result, although the ME hardware accelerator allows the computation of MVs in real-time for the QCIF image format, the implemented video encoding system is only capable of encoding one video frame every couple of seconds. Still, better performance levels are attainable for different FPGA devices or implementation technologies. In such performance enhanced video coding

Table 4. Time consumption of the ME operation using the APB ME IP core.

ME Algorithm	**FSBM**	**FSBM**	**DS**	**3SS**
ME computation time [ms]	2912	200.9	28.0	19.7
SA transfer time [ms]	–	0.71	0.71	0.71
MB transfer time [ms]	–	0.16	0.16	0.16
MV transfer time [ms]	–	0.99	0.99	0.99
Processing core	**Leon3**	**ME IP Core**		



(a) FSBM on Leon3. (b) FSBM on ME IP Core. (c) DS on ME IP Core. (d) 3SS on ME IP Core.

Fig. 5. Average relative time consumption per frame encoded using different ME algorithms and hardware structures.

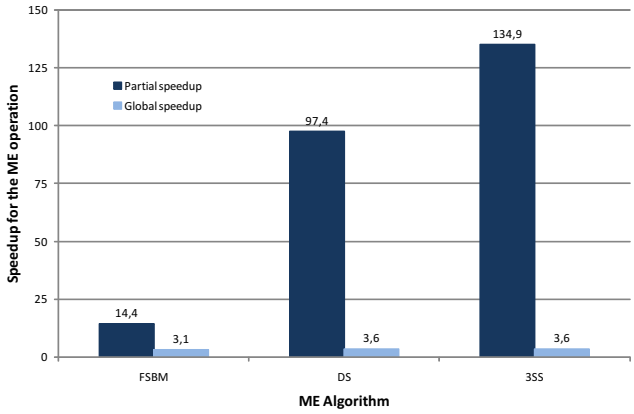


Fig. 6. Obtained speedup using the ME IP core and different ME algorithms.

systems, the proposed hardware/software co-design methodology could even be successfully applied to implement video encoders with multiple ME processors working in parallel, thus easily allowing real-time ME for higher resolution image formats (i.e., SD or HDTV).

6. CONCLUSIONS

A multi-core H.264/AVC encoder suitable for implementations of small and medium complexity embedded systems was presented. The development of such video encoder was done by using an efficient hardware/software co-design strategy to guarantee that the video encoder functions properly meet its performance goals. Using such methodology, the hardware component of the video encoder was implemented using a heterogeneous multi-core structure, composed by a Leon3 CPU, and a flexible core interconnection structure capable of accommodating several hardware accelerators. In the proposed implementation, only the ME task of the video encoder was implemented using a hardware accelerator, due to its huge complexity requirements. Such specialized processor consisted of a flexible and efficient ASIP IP core that was recently proposed in the literature. Several optimizations were also applied to such accelerating core in order to increase its communication bandwidth. For the software component of the video encoder, an implementation of the p264 video encoding software, conveniently optimized for embedded system applications, was considered. Experimental results obtained with the implementation in a Virtex4 FPGA of the proposed H.264/AVC encoder demonstrated the advantages of the adopted hardware/software co-design methodology. For the considered system, where a single hardware accelerator was used to speedup the ME task, speedup factors greater than 15 were obtained for the ME task and over 3 for the global encoding operation. Such performance gains were obtained with a very modest increase of the involved hardware resources.

7. REFERENCES

[1] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: tools, performance, and complexity," *IEEE Circuits and Systems Magazine*, vol. 4, no. 1, pp. 7–28, Jan. 2004.

[2] M. Owaida, M. Koziri, I. Katsavounidis, and G. Stamoulis, "A high performance and low power hardware architecture for the transform quantization stages in H.264," in *IEEE International Conference on Multimedia and Expo 2009 (ICME 2009)*, June 2009, pp. 1102–1105.

[3] S.-C. Hsia, S.-H. Wang, and Y.-C. Chou, "A configurable IP for mode decision of H.264/AVC encoder," in *Second NASA/ESA Conference on Adaptive Hardware and Systems 2007 (AHS 2007)*, Aug. 2007, pp. 146–152.

[4] N. Sebastião, T. Dias, N. Roma, P. F. Flores, and L. Sousa, "Application specific programmable IP core for motion estimation: Technology comparison targeting efficient embedded co-processing units," in *11th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD2 008)*, Sept. 2008, pp. 181–188.

[5] J.G. Tong, I.D.L. Anderson, and M.A.S. Khalid, "Soft-core processors for embedded systems," in *International Conference on Microelectronics 2006 (ICM 2006)*, 16-19 2006, pp. 170–173.

[6] A. Rodrigues, N. Roma, and L. Sousa, "p264: Open platform for designing parallel H.264/AVC video encoders on multi-core systems," in *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2010)*, June 2010, pp. 81–86.