# H.264/AVC framework for multi-core embedded video encoders

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

***Abstract*— A highly modular framework for developing parallel H.264/AVC video encoders in multi-core systems is presented. Such framework implements an efficient hardware/software co-design methodology, which enables replacing the software implementation of any operation in the video encoder application by a corresponding system call to a hardware accelerator. To achieve such goal, this design strategy adopts a simple and straightforward method to model all functional blocks of the video encoder into self-contained software modules. Such method takes into consideration not only the data structures required to implement the considered operations, but also the available interface of the target hardware structure. To prove the validity of the proposed framework, an implementation of a multi-core H.264/AVC video encoder using an ASIP IP core as a ME hardware accelerator is presented. The obtained results evidence the advantages of this methodology and demonstrate the performance gains it can provide. For the considered system, speedup factors greater than 15 were obtained for the ME operation.**

## I. Introduction

Nowadays, the H.264/AVC standard is regarded as the *de-facto* standard for video applications, owing to its high coding efficiency levels and an increased flexibility to efficiently support different implementations targeting all classes of applications [1]. Nevertheless, such powerful characteristics come at a huge cost in terms of computational power, memory size and bandwidth. To alleviate such constraints and achieve real-time video coding, many embedded implementations are based on parallel software applications supported by multi-core hardware structures [2], [3]. Such structures typically consist of an array of heterogeneous processing elements, integrated in a System-on-Chip (SoC) architecture, often composed by a General Purpose Processor (GPP) and multiple dedicated or specialized processors, used as hardware accelerators for the most critical parts of the video coding algorithm.

Due to the involved heterogeneity and complexity issues, the development of this type of architectures is usually based on efficient hardware-software co-design strategies [4], where both the hardware and the software components are designed together to optimize the system performance, power consumption and area usage. These two components focus distinct optimization directions. The hardware component aims at populating the multi-core structure with computational/power/area efficient processing cores and interconnection structures implementing fast communication protocols. On the other hand, the software component tries to improve the performance/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 and control communications. Hence, this hardware/software co-design approach allows achieving the intended global system performance, provided that the application can be efficiently parallelized and the hardware accelerators that are adopted in the SoC offer relevant performance gains, i.e., effective speedup values.

Unfortunately, the core integration task frequently imposes a significant impact on the effective system speedup provided by the hardware accelerators, since it often restricts the parallel execution capabilities of the encoder. Furthermore, the implementation of this hardware/software design strategy is also very time consuming and error prone, since it generally consists of a customized implementation process so as to guarantee the desired performance levels. As a result, it has become increasingly difficult and far too expensive for developing modern medium and high complexity multi-core systems, due to the tight constraints on area, power consumption and performance that system developers face.

To overcome all these constraints, this paper presents a highly modular framework for developing parallel H.264/AVC video encoders in generic multi-core systems. Such framework aims at reducing the parallelization overhead and communication time, as well as to minimize the core integration effort and the system development time.

This paper is organized as follows. In Section II the proposed framework is presented. Section III describes the implementation steps involved in the design of a multi-core H.264/AVC encoder using the proposed framework. Experimental results concerning such implementation are provided in Section IV and Section V concludes the paper.

## II. Hardware/software H.264/AVC modular framework

The proposed framework is based on a hardware/software co-design approach and can be used to develop generic H.264/AVC video encoding multi-core SoCs. To achieve such goals, its hardware and software components consist of highly modular and flexible architectures that do not depend on each other, and are capable of supporting any multi-core hardware

978-1-4244-8277-1/10/$26.00 ©2010 IEEE 

structures. In the following, the two framework components are described in more detail.

### A. Modular video encoding software application

The software component of the proposed H.264/AVC framework consists of the software program that implements the video encoding application. Such program is responsible for guaranteeing the control flow within the video encoder, as well as for carrying out all the video encoding algorithms that are not implemented in hardware accelerators. Moreover, it also supports the interactions between processing cores in the multi-core array and controls the communication between all system hardware components. The offloading of some operations of the video encoding application to hardware accelerators is also managed by this program, so as to guarantee the desired performance levels.

In the current version of the proposed H.264/AVC framework, this software component consists of an adapted version of the p264 programming platform [5] targeting SoC implementations. The p264 platform is a configurable software application derived from version JM14.0 of the H.264/AVC Reference Software Model (see `http://sips.inesc-id.pt/prototypes.php#p264` for more information), that presents a highly modular and flexible structure where all the functional modules of the video encoder are implemented as independent and self-contained software modules. Hence, it allows to easily and swiftly replacing a software realization of any given function of the video encoder by a system call to a hardware accelerator implementing that same function whenever higher performance levels are required. By being functionality self-contained, and more importantly, independent in terms of the processed data inside each module, the proposed architecture for the video encoding application allows to avoid delays in the program execution due to synchronization and data coherency issues.

### B. Scalable multi-core hardware structure

The other component of the proposed framework addresses the hardware implementation issues concerning to the integration of new processor elements, such as hardware accelerators, in multi-core SoCs. To be precise, it implements a methodology that not only allows to significantly reducing the time required for this operation, but also guarantees minimum losses in system performance considering the maximum theoretical performance levels that could be obtained with the speedups provided by each hardware accelerator. Furthermore, such methodology also includes the techniques required to efficiently insert and remove processing cores from the multi-core array in run-time, which is highly relevant for reconfigurable platforms like the ones supporting Reconfigurable Video Coding (RVC).

The implemented methodology is independent of the considered target hardware platform, and consists mostly of three different steps. The first step consists in the definition of the communication interface to be used for interconnecting the core to the multi-core system. Although this is a trivial design stage for multi-core systems based on a single communication structure, that is not the case for the vast majority of multi-core SoCs, where multiple communication interfaces with different characteristics are available. In such cases, the selection of the most suitable interface results from a careful assessment of the core communication requirements (e.g.: latency and bandwidth), and of a thorough analysis of the interface signals made available by both the core and the communication structures under consideration.

The next step of the considered methodology provides the user with a programming interface for the new processing core. Such interface is derived from the set of functions implemented by the considered core, and by taking into consideration the amount of data typically required to communicate with the core. Hence, depending on the specific characteristics of each core, this interface may consist solely of a set of memory mapped registers (allowing simple and direct access to the core functionalities and status), or of more complex solutions like Direct Memory Access (DMA) operations using burst transfers and split transactions. Combinations of several techniques may also be used to optimize the interaction with the core.

The final step in this methodology involves the development and the implementation of a hardware wrapper for the core. This circuit makes the adaptation of the core interface signals to the SoC communication interface where it will be connected to. Moreover, such circuit is also responsible for providing the hardware support required for the implementation of the previously defined core programming interface.

## III. Multi-core video encoder design

To validate the proposed framework and prove its advantages in the development of H.264/AVC coding systems, the implementation of a multi-core SoC for H.264/AVC video encoding using the proposed platform was realized.

Although any other multi-core structure could have been considered, the hardware component of the implemented system is based on version gpl-1.0.20-b3403 of the GRLIB [6] platform due to its versatility. Such hardware framework is maintained by Gaisler Research and makes available several Intellectual Property (IP) cores that can be freely used to realize modern multi-core SoCs based on the Leon3 soft-core processor. This soft-core implements a 32-bit RISC architecture conforming to the SPARC v8 definition, which is supported by a 7-stage integer pipeline Harvard micro-architecture. The internal core functionality can be greatly customized and easily extended externally using the AMBA Advanced High-performance Bus (AHB) and Advanced Peripheral Bus (APB). Hence, it completely fulfils the requirements of the proposed framework.

In the considered implementation, whose block diagram is presented in Fig. 1, the Leon3 soft-core processor was used to implement the system Central Processing Unit (CPU) and other IP cores from the GRLIB library were also used to populate the multi-core array: a programmable memory controller for PROM and SDRAM memory access; a DSU and a JTAG controller for PROM programming and debugging purposes;

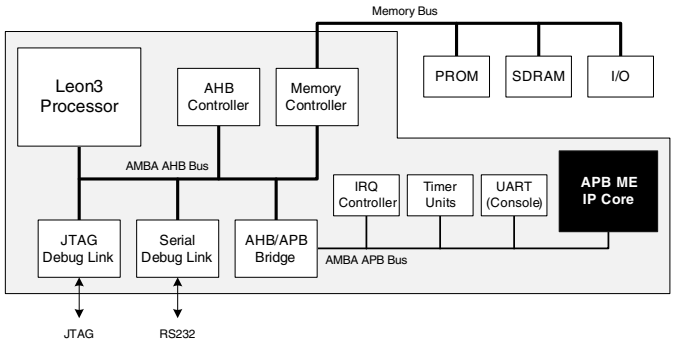


Fig. 1. Block diagram of the multi-core H.264/AVC video encoder.

a RS-232 UART for I/O operations and user interaction; and two 32-bit timer units. All these hardware structures were interconnected using the AMBA AHB and APB interfaces.

As Fig. 1 illustrates, the proposed SoC also uses a specialized processor to speedup the Motion Estimation (ME) operation, which is reported in literature as the most complex operation of a H.264/AVC encoder [1]. Such processing core is based on an improved version of an Application Specific Instruction Set Processor (ASIP) IP core for ME recently proposed [7], which was modified to provide a 32-bit interface and more efficient communication protocols. This accelerator was embedded in the multi-core array following the methodology proposed in section II-B.

Firstly, the AMBA APB interface was selected to interconnect the core to the main system bus, due to the simplicity of this interface and the amount of data that is required to be transferred between the hardware accelerator and the main system memory. Then, the user programming interface of the hardware accelerator was specified by taking into consideration the set of commands required for the core operation. Such interface is revealed in Fig. 2 and consists of three sets of memory mapped registers that allow to control and evaluate the core operation, upload the firmware and the pixel data to the core internal memory banks, and retrieve the results of the completed ME operations. Finally, a hardware circuit was realized to adapt the ASIP original interface to the AMBA APB interface, thus giving hardware support to the user programming interface and the AMBA APB bus protocol. Such wrapper is depicted in Fig. 3.

In what concerns to the software component of the multi-core SoC, it consists of the embedded systems version of the p264 platform described in section II-A that was fine tuned to optimize the interaction between the software program and the ME core. More specifically, a modular and low complexity device driver with a simple and yet effective Application Programming Interface (API) was developed for the ME processor based on the user programming interface model depicted in Fig. 2. This API was used to replace the original software implementation of the ME operation in the p264 platform by a system call to the APB ME hardware accelerator, as well as to realize all the data transfers involving such structure. These operations were scheduled to occur in parallel with the estimation of Motion Vectors (MVs) for reference image

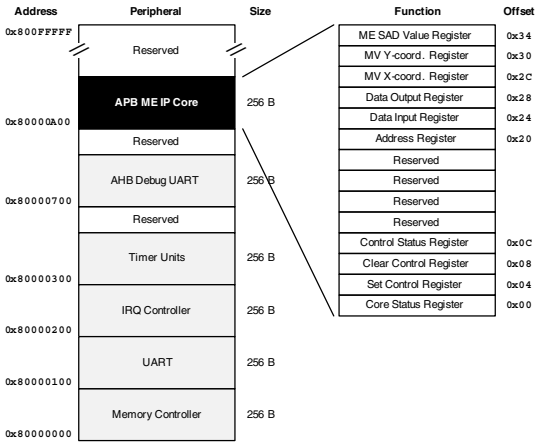


Fig. 2. Memory map of the multi-core processor for the APB peripherals and ME IP core user interface.

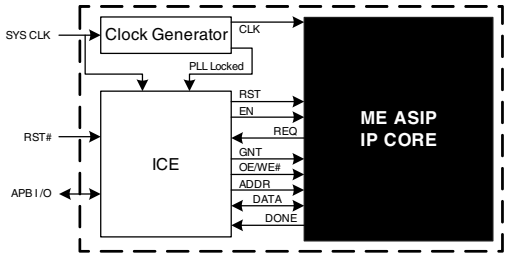


Fig. 3. Block diagram of the APB ME IP core.

blocks in order to guarantee optimal efficiency.

## IV. Experimental results

To validate the functionality and assess the performance gains provided by the proposed H.264/AVC framework, the multi-core SoC presented in section III was implemented in a GR-CPCI-XC4V FPGA prototyping platform [8] equipped with a Virtex4 XC4VLX100 FPGA. Such evaluations were realized by encoding a set of benchmark QCIF video sequences, namely: *bream*, *carphone*, *foreman*, *mobile* and *table-tennis*. In this scope, the Full-Search Block Matching (FSBM), the Three-Step Search (3SS) and the Diamond Search (DS) algorithms were programmed in the ME hardware accelerator using the typical set of parameters for the ME operation, i.e., macroblocks with $16 \times 16$ pixels and search areas with $32 \times 32$ pixels.

Table I presents the obtained implementation results. It is worth noting the reduced amount of hardware resources that are required to implement this multi-core SoC, and thus its suitability for system implementations with small or medium complexity. According to these results, the video encoder can operate with a maximum clock frequency of 60 MHz. Such relatively low value (considering the typical values of clock frequencies reported for multimedia applications) is imposed by the Leon3 processor architecture. Nevertheless, higher operating frequencies for this processor are possible to be attained for real-world SoCs, by considering other FPGA devices or different implementation technologies. For example, the Leon3 processor is capable of operating with clock

TABLE I

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% |
| **Max. Freq.** | 60 MHz | | 90 MHz | | 60/90 MHz | |

TABLE II

OBTAINED SPEEDUP USING THE ME IP CORE.

| Operation | ME Algorithm | | |
|---|---|---|---|
| | FSBM | DS | 3SS |
| **Motion estimation** | 14.4 | 134.9 | 97.4 |
| **Frame coding** | 3.1 | 3.6 | 3.6 |

frequencies as high as 400 MHz in ASIC implementations [9]. In such cases, and provided that the ME IP core is operating using a clock frequency compatible with the implemented ME algorithm (which for the 3SS and the DS algorithms is below 100 MHz [7]), it is therefore possible for the video encoder to achieve real-time operation.

The performance gains provided by the proposed H.264/AVC framework regarding to an implementation of the original p264 platform running on a Leon3 uniprocesor SoC are emphasized in Fig. 4 and Table II. Figure 4 clearly demonstrates that by significantly reducing the time allocated for the ME operation, the overall encoding time is highly reduced and the ME operation is no longer the most demanding operation of the encoding procedure, thus providing a significant overall speedup. Moreover, it also allows to validate the usage of the AMBA APB interface to integrate the ME hardware accelerator in the multi-core SoC, as well as the proposed methodology to maximize the performance of the communications involving hardware accelerators. Such conclusion can be easily drawn by comparing the ME computation time required by the APB ME IP core against the pixel data and the ME results transfer times, which are almost null. This analysis is reinforced by the data presented in Table II, which reveals impressive speedup values for the ME task at the cost of a very modest increase of the required hardware resources. Nonetheless, the effective speedup values obtained for the global video encoder are quite more modest, owing to the low operating frequency of the considered implementation of the Leon3 processor.

## V. CONCLUSIONS

This paper presented a highly modular framework for developing parallel H.264/AVC video encoders in multi-core systems. Such framework implements an efficient hardware/software co-design methodology, where standard interconnection and communication structures are used to design the multi-core array in order to reduce the core integration

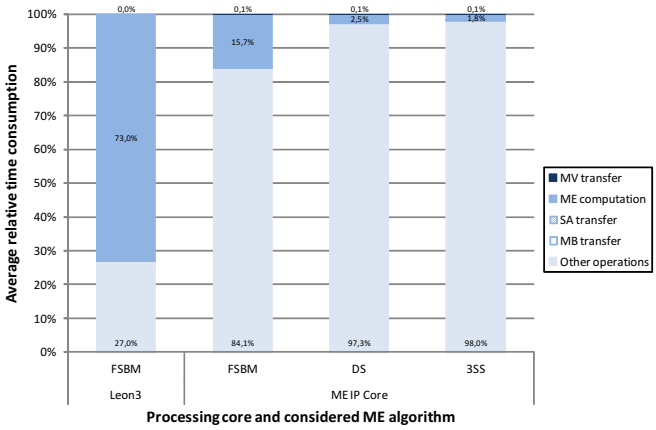


Fig. 4. Average relative time consumption per frame encoded.

effort. Moreover, the system software component is designed using a flexile and modular approach, so that it is possible to replace any operation in the software video encoder application by a corresponding system call to a hardware accelerator with minimum parallelization and communication penalties. Experimental results obtained with the implementation of a multi-core SoC of an H.264/AVC video encoder in a Virtex4 FPGA using the proposed framework have proven its advantages and demonstrated that speedups greater than 15 can be obtained for the ME task and over 3 for the global encoding operation.

## ACKNOWLEDGMENTS

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.

## 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] T.-C. Chen, Y.-W. Huang, and L.-G. Chen, "Analysis and design of macroblock pipelining for h.264/AVC VLSI architecture," in *Circuits and Systems (ISCAS'04), Proceedings of the 2004 International Symposium on*, vol. 2, May 2004, pp. 273–276.

[3] Y. Kun, Z. Chun, D. Guoze, X. Jiangxiang, and W. Zhihua, "A hardware-software co-design for H.264/AVC decoder," in *Solid-State Circuits Conference, 2006 (ASSCC 2006), IEEE Asian*, Nov. 2006, pp. 119–122.

[4] W. H. Wolf, "Hardware-software co-design of embedded systems," in *Proceedings of the IEEE*, 1994, pp. 967–989.

[5] 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)*, Jun. 2010, pp. 81–86.

[6] *GRLIB IP Core User's Manual*, 1st ed., Aeroflex Gaisler AB, Apr. 2010.

[7] 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 (DSD2008)*, Sep. 2008, pp. 181–188.

[8] *GR-CPCI-XC4V Development Board – User Manual*, 0th ed., Gaisler Research / Pender Electronic Design, Apr. 2006.

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