Design of OpenCL-Compatible Multithreaded Hardware Accelerators with Dynamic Support for Embedded FPGAs

Alfonso Rodríguez, Juan Valverde, Eduardo de la Torre

***Abstract*—ARTICo^3 is an architecture that permits to dynamically set an arbitrary number of reconfigurable hardware accelerators, each containing a given number of threads fixed at design time according to High Level Synthesis constraints. However, the replication of these modules can be decided at run-time to accelerate kernels by increasing the overall number of threads, add modular redundancy to increase fault tolerance, or any combination of the previous. An execution scheduler is used at kernel invocation to deliver the appropriate data transfers, optimizing memory transactions, and sequencing or parallelizing execution according to the configuration specified by the resource manager of the architecture. The model of computation is compatible with the OpenCL kernel execution model, and memory transfers and architecture are arranged to match the same optimization criteria as for kernel execution in GPU architectures but, differently to other approaches, with dynamic hardware execution support.**

In this paper, a novel design methodology for multithreaded hardware accelerators is presented. The proposed framework provides OpenCL compatibility by implementing a memory model based on shared memory between host and compute device, which removes the overhead imposed by data transferences at global memory level, and local memories inside each accelerator, i.e. compute unit, which are connected to global memory through optimized DMA links. These local memories provide unified access, i.e. a continuous memory map, from the host side, but are divided in a configurable number of independent banks (to increase available ports) from the processing elements side to fully exploit data-level parallelism. Experimental results show OpenCL model compliance using multithreaded hardware accelerators and enhanced dynamic adaptation capabilities.

***Index Terms*—Multithreading, Hardware Accelerators, Dynamic and Partial Reconfiguration, Fault Tolerance, OpenCL, FPGAs**

I. Introduction

The saturation of single core technologies in terms of both performance and energy efficiency, forces inevitably to explore different levels of parallelism other than Instruction-Level Parallelism (ILP) to continue increasing computing power while at the same time keeping affordable energy consumption rates. However, parallelism at application, language and architecture levels must be consistent in order to get the expected performance by means of an efficient acceleration.

At application level, two different types of parallelism (apart from ILP) can be identified: Task-Level and Data-Level Parallelism (TLP and DLP, respectively) [1]. TLP is typically used in multicore platforms, since it is based on Multiple Instruction Multiple Data (MIMD) execution. DLP, on the other hand, is mainly used in manycore architectures, e.g. GPUs, for it uses Single Instruction Multiple Data (SIMD) execution. However, the success obtained by GPUs, though significant, is limited by their power consumption, since, after all, execution is done in software, which cannot compete with the lower energy used in dedicated hardware implementations.

At language level, and although there are several alternatives in the state of the art, OpenCL (Open Computing Language) is currently gaining more importance, since it supports both TLP and DLP, as well as heterogeneous platforms. In these platforms, parallelism has to be explicitly declared by using the OpenCL programming model. Hence, developers have to define which is the most efficient way of programming each platform, always taking into account the underlying hardware. In addition, the OpenCL specification includes an execution model that enables efficient implementation on GPUs and, crucial for the motivation of the present work, on any other platform which uses a similar Model of Computation (MoC).

At architecture level, this work is based on the ARTICo^3 architecture [2], which is an FPGA-based architecture similar to GPUs, but in which processing to exploit DLP is done in hardware rather than in software. The ARTICo^3 framework appears in a context where embedded systems tend to be more adaptive, self-aware and context-aware, with variable task requirements in terms of dependability, acceleration and energy along time. Hence, the architecture is able to support the execution of tasks on a variable number of hardware accelerators, which can be dynamically configured to better adapt to changing fault tolerance and acceleration requirements, by means of Dynamic and Partial Reconfiguration (DPR). Moreover, self-aware resource management strategies to select the optimum operating point for any task provide complete independence between the task itself and the requirements at any given instant of time. Self-awareness and adaptivity are

enhanced by accurate prediction models of task execution [3] that, together with the predictability given by the High Level Synthesis (HLS) tools proposed for the development of hardware accelerators, are an important step towards predictability in adaptive multithreaded hardware-accelerated architectures.

The main contribution of this paper is the model of parallelism, as well as the underlying hardware architecture, envisaged to efficiently support the different OpenCL models, i.e. platform, execution and memory models. Experimental results showing execution times and energy consumption with three different fault-tolerance levels using a parallel block cipher algorithm confirm model compatibility between the ARTICo^3 and OpenCL frameworks, with additional dynamic adaptation capabilities.

The rest of this paper is organized as follows: section II presents the ARTICo^3 framework, its architecture, and establishes the relationships with the different models present in the OpenCL specification (platform, execution and memory models); section III reviews the current state of the art to highlight the main contributions of the proposed approach; section IV details the implementation of the block cipher example, and the obtained results are exposed and analyzed in section V, whereas conclusions and future work are presented in section VI.

II. ARTICo^3 Framework: OpenCL Support

ARTICo^3 is a bus-based architecture to support multithreaded hardware acceleration in SRAM-based FPGAs due to their DPR capabilities. This is one of the main pillars of the architecture itself. Moreover, the reconfigurable region makes use of slot-based reconfiguration. These slots can be loaded at runtime with different hardware accelerators, which are application-specific and may consist of one or more parallel threads, thus having multithreading capabilities.

The static region contains a DMA engine to enable burst transactions over a dedicated data bus and feed the reconfigurable accelerators through a unique gateway called data shuffler, which acts as a bridge between the static and the reconfigurable regions. Data delivery and processing are done differently depending on the mode on which the architecture is working, supporting overlapped execution, or even hardware redundant execution to deploy fault-tolerance techniques such as Double and Triple Module Redundancy (DMR and TMR, respectively) by adding a voter unit in the returning datapath from the accelerators. The accelerators to be loaded in each reconfigurable slot and their corresponding operation mode are decided at runtime by the resource manager of the architecture, which takes into account the requirements of each task (either internal or external to the system), whereas the sequences of data transfers, as well as the execution sequencing process, are programmed by an embedded scheduler. Therefore, it is the resource manager the one in charge of guaranteeing task and requirements independence, as well as receiving kernel invocations from the host and offloading the programming of their execution to the scheduler.

In this section, OpenCL support within the ARTICo^3 framework is covered using a bottom-up approach: the underlying hardware, which has already been detailed in previous paragraphs, acts as the starting point, upon which all specific OpenCL models are built and their abstraction derived. Furthermore, an additional subsection regarding dynamic adaptation within the framework is included to further explain system capabilities that are transparent to OpenCL, and thus still compliant with the standard, but provide additional functional extensions.

A. Platform Model

In each ARTICo^3 implementation there is, at least, one host microprocessor. Applications running in the host make use of application-specific hardware accelerators in order to speed up data intensive program sections. Therefore, the analogy between the ARTICo^3 framework and the OpenCL platform model is easy to establish, as it can be seen in Fig. 1 (a). The data shuffler, together with the resource manager, the scheduler and the reconfigurable region constitutes an OpenCL device. Going down in the hierarchy, each hardware accelerator can be thought of as an OpenCL compute unit. At the lowest level of this hierarchy, parallel threads inside each hardware accelerator behave as OpenCL processing elements. Notice that some accelerators might have only one thread and thus, one OpenCL processing element. Also note that in the ARTICo^3 framework, host and OpenCL device share the same FPGA fabric, thus reducing silicon footprint.

B. Execution Model

The embedded scheduler programs optimized DMA transfers between the external memory and one or more accelerators through the data shuffler, making full use of bus occupancy and memory bandwidth. Hence, the limitation of hardware accelerators in ARTICo^3, which otherwise could not access the external RAM memory (shared among all accelerators) directly, is eliminated. Furthermore, the combination of any accelerator, the scheduler and the data shuffler has full master capabilities regarding the data bus and, therefore, the external memory.

DPR enables module replication and relocation, and thus, having more than one copy of the same accelerator. By an optimized data delivery, these replicated accelerators behave in SIMD-like fashion, assuming no data dependencies between them. Since the architecture is bus-based, data transfers are sequenced in time through the data bus. Therefore, a whole burst transaction coming from the external RAM memory can be split in as many blocks as the number of replicated accelerators without introducing additional latency. This process, which is done on-the-fly by the data shuffler once programmed, leads to an overlapping between the execution and the data transfer itself, since once each accelerator has its data, it can start processing.

The relationship between the ARTICo^3 MoC and the OpenCL execution model is shown in Fig. 1 (b) and (c). Each work-group is mapped into a replicated hardware accelerator,

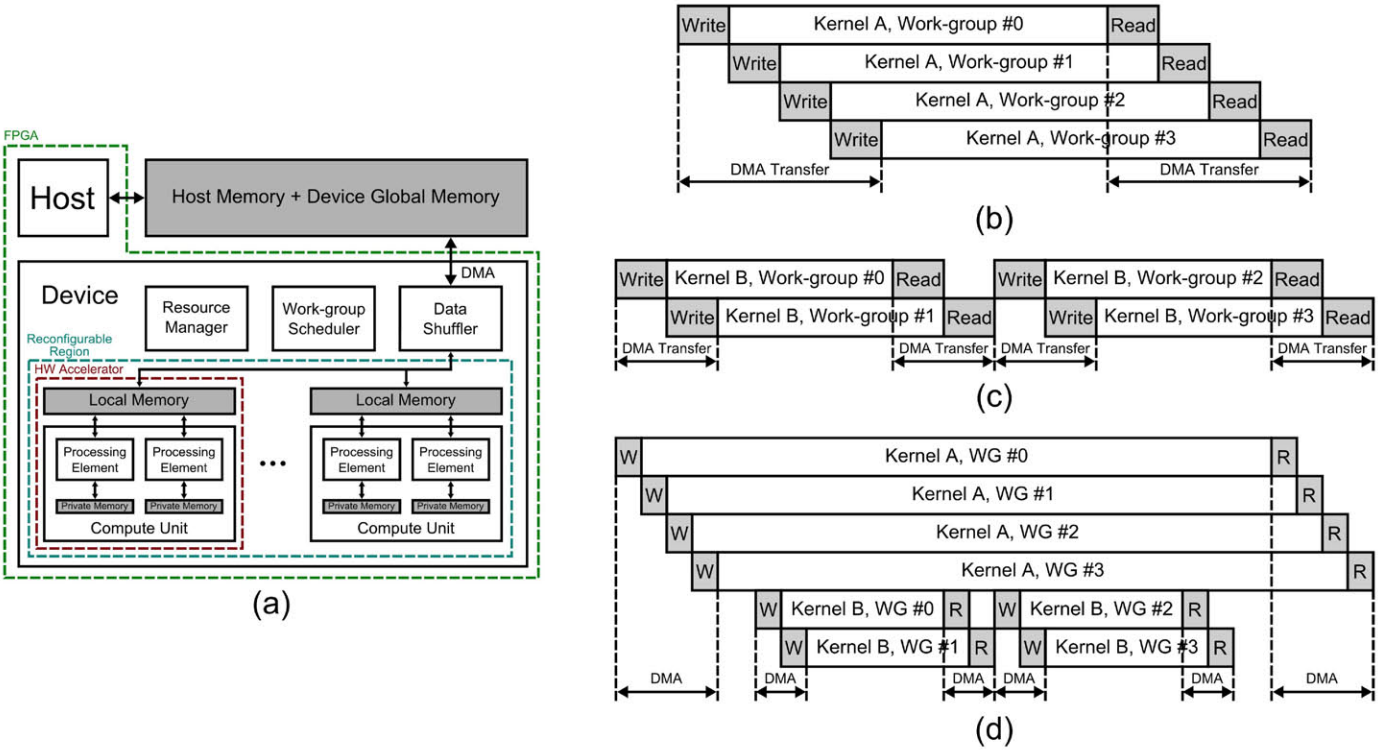

Fig. 1. ARTICo^3 framework: OpenCL model support. Platform and memory models are represented in (a), whereas the execution model is presented in (b), where all work-groups of a given kernel are executed in an overlapped way; and in (c), where two rounds of work-groups are required to finish kernel execution. Note that the main difference between (b) and (c) is resource availability (i.e. the number of available compute units). Dynamic support enabling multikernel execution within the architecture is shown in (d).

and work-items are mapped into the parallel threads inside the accelerator. Module replication through DPR, and data independences between accelerators enable OpenCL support for NDRange kernel invocation, as well as transparent scalability during execution within the architecture. Hence, in Fig. 1 (b) a kernel is invoked with 4 work-groups having 4 available slots, and executes in one round; in Fig. 1 (c), on the other hand, another kernel is invoked with 4 work-groups having only 2 available slots, thus leading to a sequential execution of two rounds to achieve full kernel completion.

C. Memory Model

ARTICo^3 relies on an external RAM memory where both host and the DMA engine of the architecture perform read and write operations. In order to enhance processing performance, hardware accelerators have a local memory which, as it is shown in Fig. 2, can be divided in a configurable number of independent banks to fully exploit DLP, e.g. several threads accessing different memory positions at the same time. Inside each hardware accelerator there is additional logic that performs address translation so that the independent memory banks appear as a continuous map from the data shuffler side. Furthermore, this translation process is also configurable, and can be tailored to any specific application.

In OpenCL terminology, the external RAM memory would be both host and device global memory. Being a shared memory between host and device, the main bottleneck of

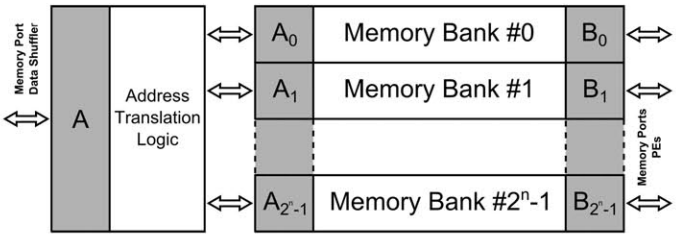

Fig. 2. Generic local memory structure inside the hardware accelerators (compute units), with multibank support to enable parallel accesses from the processing elements.

other parallel processing devices such as GPUs, i.e. memory bandwidth between host and device, is clearly mitigated. The OpenCL specification also defines two additional memory levels: local memory, which in ARTICo^3 is the multibank memory inside each accelerator; and private memory, which corresponds to internal storage elements such as registers inside each work-item. The memory hierarchy of OpenCL within an ARTICo^3 implementation can be seen in Fig. 1 (a).

D. Dynamic Adaptation

The ARTICo^3 architecture was initially conceived to dynamically change its operating point by establishing a trade-off between computation, energy consumption and dependability. So far, only computation-related features have been discussed, focusing on the similarities between the ARTICo^3

and OpenCL frameworks. However, the architecture is also capable of setting up more than one accelerator to work on the same data, so that fault-tolerant execution is achieved. Data are delivered strictly in parallel to the accelerators and the results are read also in parallel but through a voter unit that is capable of masking faults in any position of the reconfigurable logic. This operation mode is completely compatible with the aforementioned overlapped execution to enhance computation, being the only limitation the overall amount of physical resources inside the FPGA. Moreover, the resource manager can change between modes to better adapt to changing requirements.

Furthermore, the architecture does also provide additional features that enable self-awareness. The resource manager has access in real time to runtime metrics such as power consumption, execution times or even bus occupancy, and it uses embedded models to achieve energy-efficient or even predictable execution [3]. As an example, Fig. 1 (d) shows how two kernels are scheduled to be executed concurrently. The resource manager detects, by means of a bus monitor, that kernel A is not making full usage of memory bandwidth and then tells the scheduler to launch two rounds of kernel B (already enqueued) work-groups, for it also detects that there are two available slots.

The main difference between other OpenCL-compliant devices and ARTICo^3 is indeed this dynamic adaptation capability. The use of DPR allows to dynamically change the computing matrix, or more precisely, to dynamically modify the number and type of OpenCL computing units, which are application-specific and thus highly optimized, at runtime in an autonomous way. Hence, this is what makes ARTICo^3 a powerful, OpenCL-compliant computing platform for embedded systems. To the best of the authors' knowledge, there is no other architecture in the state of the art with dynamic support for multithreaded hardware acceleration and OpenCL support.

III. Related Work

Hardware acceleration has become an important feature in computation-intensive environments due to technology limitations. In order to support this kind of acceleration, and in the context of reconfigurable computing, it is mandatory to implement efficient architectures. Moreover, languages and computing models have to be developed on top of these architectures to achieve the expected results [4].

Throughout the literature there are several examples of this integration between architectures that provide hardware acceleration and computing models. In [5], a multithreaded computing model to support parallelism in chip multiprocessors is proposed. In [6], on the other hand, a slot-based virtual architecture to support streaming processing in reconfigurable FPGAs is presented. In the last few years, some architectures have enabled support for multiple hardware threads, as in [7], where the processing is also streaming-based.

Apart from the architecture, it is also important the method by which hardware accelerators are generated. Traditionally, the approach has been to develop accelerators using Hardware Description Languages (HDLs), as it can be seen in [8]. Nevertheless, HLS approaches, in which accelerators are obtained from a high-level language description, have gained more importance recently [9]. A good example can be found in [10], where multithreaded hardware accelerators are obtained by using some extensions over the Nymble framework [11]. In addition, more complex examples can be found in the literature, such as in [12], where the HLS flow is embedded in the reconfigurable platform.

Moreover, high-level languages with explicit definitions of parallelism at different levels are now part of HLS methodologies, either directly or by using the LLVM framework [13]. For instance, OpenMP has been used in [14] and in [15]. FPGA vendors, on the other hand, have decided to include OpenCL support in their design flows. Both Altera [16] and Xilinx [17] provide tools to implement OpenCL-compliant applications in their hardware platforms. Hence, researchers from academia have also put their effort in working with OpenCL in order to generate custom accelerators, with tools such as SOpenCL [18], methodologies based upon commercial tools [19] or by using LLVM to accelerate synthesis processes [20].

Differently from the aforementioned alternatives, the ARTICo^3 framework provides an architectural model that enables dynamic adaptation at runtime, with a finer reconfiguration granularity than other approaches, even though accelerator generation is also based upon HLS from OpenCL code, supporting parallelism at different levels.

IV. Application Example

Due to the application-specific nature of the hardware accelerators, the OpenCL platform and execution model terminologies can be merged into one single entity. Hence, compute units and processing elements will be referred to as work-groups and work-items respectively in the following.

In order to demonstrate the integration of ARTICo^3 in the OpenCL framework, a block cipher algorithm has been selected as testbench. More precisely, the chosen algorithm is the Advanced Encryption Standard (AES) in its 256-bit key-length version working in counter mode (CTR), since it is an operation mode that provides good results in terms of both confidentiality and parallelization capabilities. Two implementations of the same algorithm have been developed: one with 1 work-item per work-group, and another with 2 work-items per work-group.

Local memory partitioning depends on the number of work-items, even though each hardware accelerator has the same local memory size (64 kB in this particular example). Fig. 3 shows the partitioning in accelerators containing 1 work-item. Notice that the general architecture presented in Fig. 2 has been simplified to ease application understanding. Two banks are required: one for storing the plain text, and another to store the cipher text. The work-item accesses sequentially these two banks until all operations have finished. Fig. 4, on the other hand, shows the partitioning in accelerators containing 2 work-items. In this particular case, the number of memory banks is

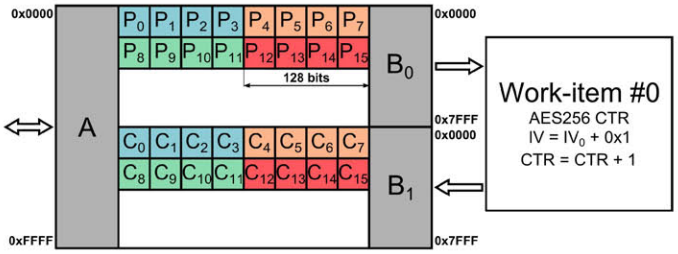

Fig. 3. Data distribution in local memory of an AES256 CTR work-group with 1 work-item.

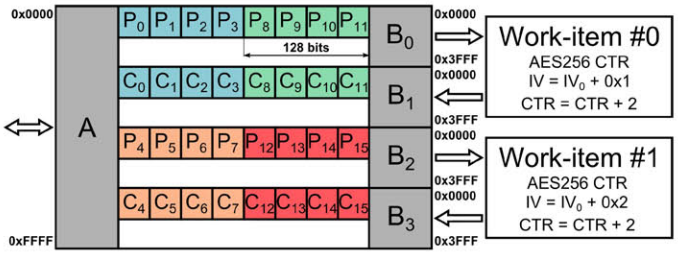

Fig. 4. Data distribution in local memory of an AES256 CTR work-group with 2 work-items.

TABLE I
RESOURCE UTILIZATION: AES256 CTR KERNEL

Resource	Work-group utilization (%Slot/%FPGA)			
	1 Work-item		2 Work-items	
	Left	Right	Left	Right
Slices	1115 (40/2.2)	1098 (45/2.1)	2057 (73/4)	2040 (83/4)
BRAMs	16 (67/3.6)	16 (67/3.6)	16 (67/3.6)	16 (67/3.6)

doubled, and the address translation logic distributes workload in a balanced fashion: 128-bit data blocks are shuffled between banks 0 and 2 so that each work-item has to process the same amount of data. Note that the address translation logic only has to be enabled in the implementation with 2 work-items, whereas it is idle, i.e. acts as a bypass, when having only 1 work-item. Moreover, address translation logic is synthesized with the hardware accelerator, and its complexity may range from address line exchanges for simple cases spanning the whole bank size, to very complex routing schemes.

The implementation has been made in a KC705 development board, which features a Xilinx XC7K325T-2FFG900C FPGA. Layouts of the placed and routed designs are shown in Fig. 5. Moreover, Table I contains information regarding resource utilization in work-groups with 1 and 2 work-items. Notice that the ARTICo^3 architecture in this specific device has forced the design to be split in two different regions, i.e. left and right. Therefore, although module relocation can be performed during reconfiguration, it can only be done in the same region due to resource heterogeneities between left and right sides. Hence, partial bitstreams from each region have to be extracted, increasing memory footprint.

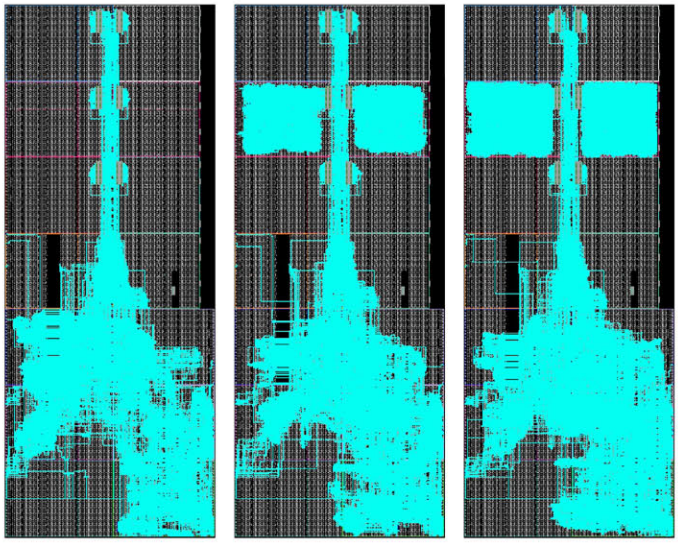

Fig. 5. ARTICo^3 layout inside the KC705 development board with no accelerators loaded (left), with 2 AES256 CTR work-groups with 1 work-item each (center), and with 2 AES256 CTR work-groups with 2 work-items each (right).

V. EXPERIMENTAL RESULTS

A comparison between accelerators containing 1 or 2 work-items is mandatory. In terms of functionality, both implementations show the same behavior, since the inherent transparent scalability of OpenCL is shared within the ARTICo^3 framework, enabled by the optimized data transfers and the address translation logic inside local memories. However, in terms of execution performance, each implementation shows a different behavior. Take for instance Fig. 6, where power consumption in the FPGA core and in the external memory, as well as the execution time are shown for a kernel invocation using 6 accelerators with 1 work-item to cipher 192 kB of raw data. When compared against Fig. 7, which shows exactly the same situation but using accelerators with 2 work-items, it is clear that the power consumption prior to kernel invocation is quite similar, but during execution these similarities disappear, showing differences in both power consumption and elapsed time.

Furthermore, the solution space of the architecture has been explored using the developed accelerators. This is shown in Fig. 8, where a fixed amount of 2.5 MB of raw data is ciphered by a changing number of hardware accelerators (either containing 1 or 2 work-items), and under changing requirements that include different fault tolerance levels (Simplex for no redundancy, DMR, and TMR). The KC705 development board only has resources to host 6 accelerators, and therefore the embedded models, previously validated and verified experimentally with other kernels and FPGA boards [3], have been used in order to predict the behavior of the system when the number of accelerators is increased above this limit (dotted lines in the figures). Hence, the transition from computing-bounded to memory-bounded execution is clear: when maximum bus occupancy is reached, no further speedup

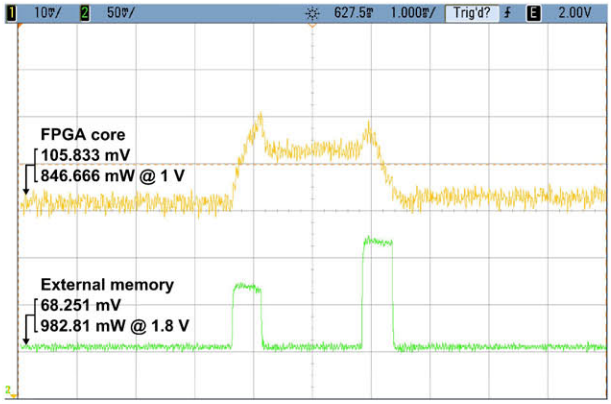

Fig. 6. Power consumption in the FPGA core and in the external memory during the execution of 6 AES256 CTR work-groups with 1 work-item each.

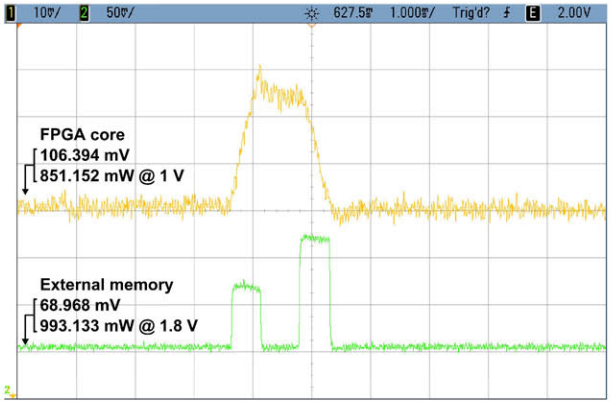

Fig. 7. Power consumption in the FPGA core and in the external memory during the execution of 6 AES256 CTR work-groups with 2 work-items each.

can be achieved. Notice that once this memory bandwidth limitation is reached, the actual bus occupancy is almost 96% with respect to the theoretical maximum.

Hence, execution speedup in ARTICo^3 shows two main limiting factors, as it can be seen in Fig. 9: on the one hand, technology limitations, which impose the maximum number of reconfigurable slots that can be loaded with accelerators; on the other hand, memory bandwidth limitations, which impose the maximum data rate that can be delivered to the loaded accelerators. Regarding the latter, it is possible to compute the optimal operating point, i.e. the working point immediately before entering memory-bounded execution, by analyzing the ratio between accelerator execution and data transfer times (both measurable at runtime with internal monitors). The optimal operating points for both accelerators are also shown in Fig. 9. Notice that these optimal points are not supported in the KC705 board due to technology limitations.

The aforementioned limitations have an additional impact on the overall energy efficiency of the system, as it is shown in Fig. 10. SRAM-based FPGAs still have a large component of static power consumption, which in most cases is larger than the dynamic contribution itself. Therefore, the faster the processing is done, the more energy-efficient the execution becomes. Once the memory-bounded region is reached, energy savings start decreasing. In this case, the optimal operating points have also been highlighted in the graph.

VI. Conclusions and Future Work

The ARTICo^3 framework provides native OpenCL-compliant platform, execution and memory models, being its architecture similar to other OpenCL devices such as GPUs. However, the main difference is that compute units are hardware-based processing units, with DPR support to enhance system flexibility up to the level of similar software-based approaches, increasing the virtual number and type of compute units in the device. Moreover, DPR is part of a set of tools (system monitors, embedded prediction models, etc.) that provide dynamic adaptation at runtime, going beyond the OpenCL specification but being transparent to the developer.

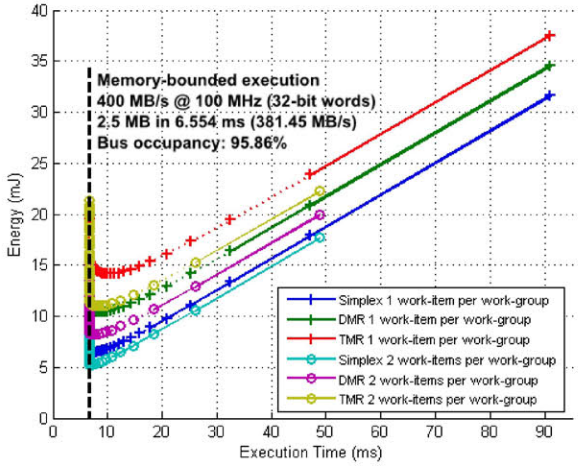

Fig. 8. Solution space of the AES256 CTR implementations with work-groups of 1 or 2 work-items each in ARTICo^3.

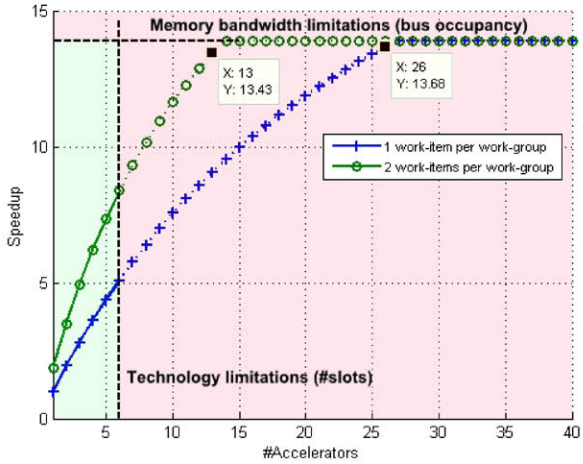

Fig. 9. Speedups of the AES256 CTR implementations with work-groups of 1 or 2 work-items each in ARTICo^3 (reference value is 1 work-group with 1 work-item).

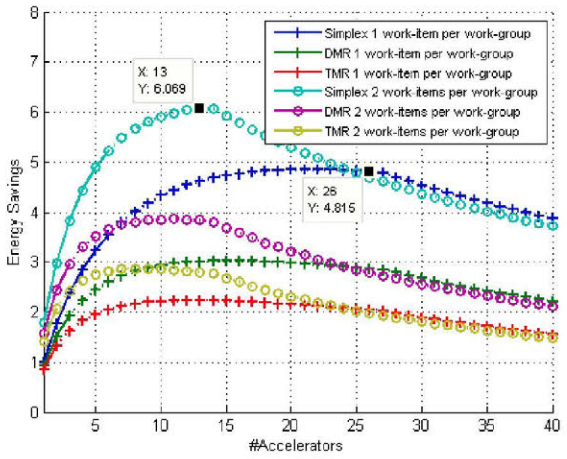

Fig. 10. Energy savings of the AES256 CTR implementations with work-groups of 1 or 2 work-items each in ARTICo^3 (reference value is 1 work-group with 1 work-item).

Multibank local memories with address translation units enable optimized data transferences and grouping (which can favor coalesced data accesses from work-items), but keeping full compliance with the OpenCL specification. Experimental results show that integration of work-groups with variable number of work-items does not require additional effort on the software developer side. Moreover, multithreaded hardware accelerators provide better results in terms of energy efficiency and execution times in SRAM-based FPGAs.

The proposed architecture and framework provide a balance between computing performance, dependability requirements and energy-efficient execution independently on the OpenCL tasks specified in the application code. This independence is guaranteed by the resource manager and its dynamic adaptation capabilities.

Currently, the work is focused on providing full support for the OpenCL programming model. Developers' code is to be parsed to extract memory access patterns and program the resource manager and the work-group scheduler, and kernel code is to pass through HLS design flows to generate multithreaded hardware accelerators. With this, ARTICo^3 implementations will behave as any other OpenCL-compliant device, such as GPUs or multicore processors, and thus will support functional portability of applications among them.

ACKNOWLEDGMENTS

The authors would like to thank the Spanish Ministry of Education, Culture and Sport for its support under the FPU grant program.

This work was also partially supported by the Spanish Ministry of Economy and Competitiveness under the project REBECCA, with reference number TEC2014-58036-C4-2-R.

REFERENCES

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[2] J. Valverde, A. Rodriguez, J. Camarero, A. Otero, J. Portilla, E. de la Torre, and T. Riesgo, "A dynamically adaptable bus architecture for trading-off among performance, consumption and dependability in cyber-physical systems," in *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*, Sept 2014, pp. 1–4.

[3] A. Rodriguez, J. Valverde, C. Castañares, J. Portilla, E. de la Torre, and T. Riesgo, "Execution modeling in self-aware fpga-based architectures for efficient resource management," in *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2015 10th International Symposium on*, June 2015, pp. 1–8.

[4] R. Tessier, K. Pocek, and A. DeHon, "Reconfigurable computing architectures," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 332–354, March 2015.

[5] M. Watkins and D. Albonesi, "Remap: A reconfigurable heterogeneous multicore architecture," in *Microarchitecture (MICRO), 2010 43rd Annual IEEE/ACM International Symposium on*, Dec 2010, pp. 497–508.

[6] A. Jara-Berrocal and A. Gordon-Ross, "Vapres: A virtual architecture for partially reconfigurable embedded systems," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, March 2010, pp. 837–842.

[7] Y. Wang, X. Zhou, L. Wang, J. Yan, W. Luk, C. Peng, and J. Tong, "Spread: A streaming-based partially reconfigurable architecture and programming model," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 12, pp. 2179–2192, Dec 2013.

[8] R. Weber, A. Gothandaraman, R. Hinde, and G. Peterson, "Comparing hardware accelerators in scientific applications: A case study," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 22, no. 1, pp. 58–68, Jan 2011.

[9] S. Windh, X. Ma, R. Halstead, P. Budhkar, Z. Luna, O. Hussaini, and W. Najjar, "High-level language tools for reconfigurable computing," *Proceedings of the IEEE*, vol. 103, no. 3, pp. 390–408, March 2015.

[10] J. Huthmann, J. Oppermann, and A. Koch, "Automatic high-level synthesis of multi-threaded hardware accelerators," in *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*, Sept 2014, pp. 1–4.

[11] J. Huthmann, B. Liebig, J. Oppermann, and A. Koch, "Hardware/software co-compilation with the nymble system," in *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2013 8th International Workshop on*, July 2013, pp. 1–8.

[12] G. Stitt and F. Vahid, "Thread warping: A framework for dynamic synthesis of thread accelerators," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2007 5th IEEE/ACM/IFIP International Conference on*, Sept 2007, pp. 93–98.

[13] C. Lattner and V. Adve, "Llvm: a compilation framework for lifelong program analysis transformation," in *Code Generation and Optimization, 2004. CGO 2004. International Symposium on*, March 2004, pp. 75–86.

[14] D. Cabrera, X. Martorell, G. Gaydadjiev, E. Ayguade, and D. Jimenez-Gonzalez, "Openmp extensions for fpga accelerators," in *Systems, Architectures, Modeling, and Simulation, 2009. SAMOS '09. International Symposium on*, July 2009, pp. 17–24.

[15] J. Choi, S. Brown, and J. Anderson, "From software threads to parallel hardware in high-level synthesis for fpgas," in *Field-Programmable Technology (FPT), 2013 International Conference on*, Dec 2013, pp. 270–277.

[16] T. Czajkowski, U. Aydonat, D. Denisenko, J. Freeman, M. Kinsner, D. Neto, J. Wong, P. Yiannacouras, and D. Singh, "From opencl to high-performance hardware on fpgas," in *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, Aug 2012, pp. 531–534.

[17] X. Inc., *UG1023 SDAccel Development Environment User Guide*. [Online]. Available: http://www.xilinx.com/

[18] M. Owaida, N. Bellas, K. Daloukas, and C. Antonopoulos, "Synthesis of platform architectures from opencl programs," in *Field-Programmable Custom Computing Machines (FCCM), 2011 IEEE 19th Annual International Symposium on*, May 2011, pp. 186–193.

[19] K. Shagrithaya, K. Kepa, and P. Athanas, "Enabling development of opencl applications on fpga platforms," in *Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on*, June 2013, pp. 26–30.

[20] J. Coole and G. Stitt, "Fast, flexible high-level synthesis from opencl using reconfiguration contexts," *Micro, IEEE*, vol. 34, no. 1, pp. 42–53, Jan 2014.