Self-adaptive loop for CPSs: is the Dynamic Partial Reconfiguration profitable?

Gabriella D'Andrea
DISIM Department
Università degli Studi dell'Aquila
Via Vetoio, L'Aquila Italy
Email:gabriella.dandrea@graduate.univaq.it

Tania Di Mascio
Center of Excellence DEWS
Università degli Studi dell'Aquila
Via Vetoio, L'Aquila Italy
Email:tania.dimascio@univaq.it

Giacomo Valente
DISIM Department
Università degli Studi dell'Aquila
Via Vetoio, L'Aquila Italy
Email:giacomo.valente@univaq.it

***Abstract*—Nowadays, Cyber-Physical Systems play an important role in the context of several large industries; the need for interaction with a changeable physical world leads the system adapting itself to physical changes. Adaptivity, dependability and reducing communication overheads then appear as the most wanted requirements that are moving on the adoption of the edge-computing. In turn, in this world, the demand for HW platforms able to manage increasing requirements is leading to the use of FPGAs, due to their inherent run-time reconfigurability. However, the dynamic partial reconfiguration process of an FPGA has a timing performance impact that cannot be neglected. This impact, if not well considered, can nullify the advantage obtained using a Dynamic Partial Reconfiguration. Therefore, when exploiting FPGAs with dynamic partial reconfiguration, a crucial problem is to understand whether is profitable or not to dynamically reconfigure them. In this paper, an innovative run-time manager adopting a metric to evaluate the impact of reconfiguration time is introduced, together with its validation through its usage on a basic application implemented on FPGA.**

Keywords: Edge-Computing, Dynamic Partial Reconfiguration, Embedded Systems

I. Introduction

Nowadays, Cyber-Physical Systems (CPSs) play an important role in the context of several large industries, including electronics, energy, automotive, defense and aerospace, telecommunications and industrial automation [1]. CPSs combine a cyber side, composed of computing and networking parts, with a physical side (e.g., mechanical, electrical and chemical processes).

The need for interaction with a changeable physical world leads the system adapting itself to physical changes (e.g., a sudden temperature variation), implying the requirement of *adaptivity* [2]; moreover, the criticality of the physical entities controlled by CPSs requires that systems enforce *dependability* [3]. These two requirements, together with the further requirement of *reducing communication overhead*, are moving on the adoption of edge-computing [4], i.e., transferring computation close to sensors. In order to do that, it is mandatory using HW platforms supporting the above mentioned requirements. Nowadays, one of the valid solutions is based on the usage of Field Programmable Gate Arrays (FPGAs) [5]. These platforms allow the designer the opportunity to implement applications both in SW, using soft-processors or existing hard-processors (in this case, they are called System-on-Programmable Chips (SoPCs) [6], [7]), and in HW, with custom design circuits. FPGAs platform provide both the flexibility of SW and the high-performance of HW. Regarding dependability and adaptivity, the most interesting feature of FPGAs is their capability to dynamically reconfigure a portion of HW, while the remainder of the processes continues to operate: a process called *dynamic partial reconfiguration* (DPR) [8]. This process is valuable in different contexts, such as power aware implementations [9] and timing related concerns: for example, when a system executes different applications and has to satisfy some timing requirements, it can use a DPR to instantiate application specific HW accelerators to meet deadlines [10].

When exploiting FPGAs with DPR, a crucial problem is to understand whether is profitable or not to dynamically reconfigure it. In fact, the process requires that a configuration file is transferred from a storage memory to a reconfiguration memory, requiring some time, depending on both configuration **file size** and available **bandwidth**; this impact, if not well considered, can nullify the advantage obtained using a DPR [8]. Even if in the latest years there has been a trend showing that reconfiguration times in FPGAs are progressively decreasing [11], the available bandwidth to perform DPR can have a strong impact on actual reconfiguration time, as confirmed by the study of Papadimitriou et al. [12]. There is a necessity of models to evaluate the reconfiguration time, also referred as *DPR time* hereinafter, with necessary accuracy; this, in turn, would enable the possibility to evaluate the profitability of this operation, before to apply it.

Similarly to some existing works [12] [13], this paper tries to address the problem of DPR time estimation and the associated evaluation of its profitability by firstly proposing a model to predict the DPR time, targeting Zynq7000 SoPC [6]. Then, the model is introduced inside a run-time manager that innovatively manages the self-adaptivity of CPSs at the edge, properly using the DPR time estimation to evaluate the profitability of the operation. Differently from [14], the proposed solution does not require a dedicated memory to store the configuration file, with an advantage on required area for the implementation.

The paper is organized as follows: Section II reports a detailed analysis of related works, whereas Section III

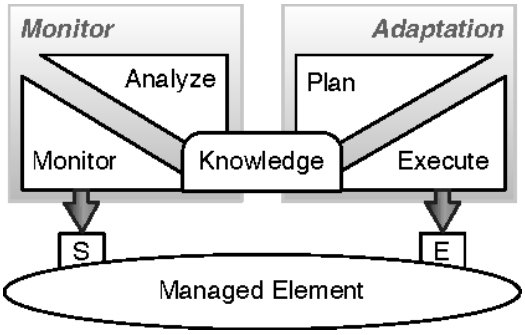

Fig. 1. The customary self-adaptive loop, where the four phases "monitor, analyze, plan, execute" [17] are shown. The managed element represents the system whereon the adaptation is applied.

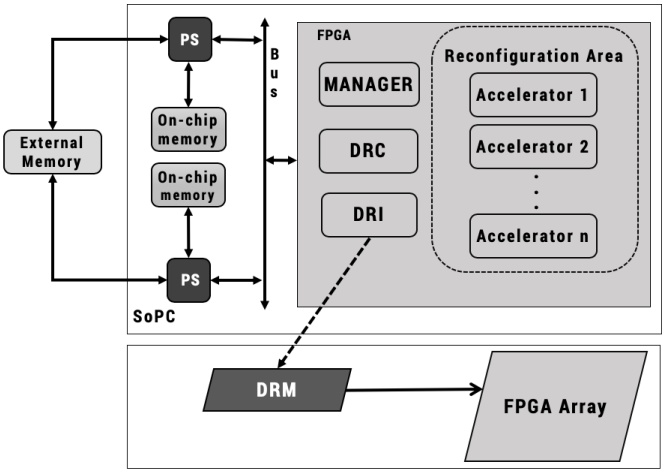

Fig. 2. Reference platform composed of an external shared memory, accessed by two Processor Systems (PSs) with related on-chip private memory. PSs are masters on a bus used to communicate with hardware on FPGA, for example with accelerators put in reconfiguration area. The FPGA contains also the proposed run-time manager, a Dynamic Reconfiguration Controller (DRC) and a Dynamic Reconfiguration Interface (DRI) to access the Dynamic Reconfiguration memory (DRM) and perform the DPR. DRM content acts directly on the FPGA array. DRM and FPGA array represent the internal part of the FPGA.

presents the proposed system, highlighting the impact on edge-computing of the proposed manager, validated in Section IV. Finally, Section V discusses some conclusions and future works.

II. Related works

The evaluation of DPR impact is a huge research area. The work in [15] evaluates the impact of DPR on power dissipation, in order to exploit that information inside a run-time manager to select different working points. Some models to estimate DPR time have been proposed in [12] and [13]: they both target previous versions of FPGAs, compared to those currently adopted, such as the Zynq7000 SoPC considered in this work. A further work related to timing impact of accelerators instantiated using DPR has been done in [16]: the difference with this work is that they do not consider the impact of reconfiguration time, that can be significative.

The DPR has been also considered in the context of real-time systems, with the goal of growing the cardinality of the task-set by adding HW-tasks at run-time, i.e., tasks to be executed with accelerators on FPGA. In this context, the work of Biondi et al. [10] proposes a model for analyzing the temporal impact of a DPR, with the goal to analyze the schedulability of real-time systems; in their model, they consider the reconfiguration time as constant. This is not a valid assumption, as demonstrated in [14], where it has been shown how the reconfiguration bandwidth can vary up to 21x, and consequently also the reconfiguration time. Still in the same field, the work of Damschen et. al. [14] proposes a new controller to have an upper bounded reconfiguration time, in order to provide hard real-time guarantees: the controller requires a guaranteed bandwidth access memory for reconfiguration file storage.

All the works above reported originate from different fields of application. Differently, the proposed run-time manager aims to be applicable to different contexts that involve systems where DPR is adopted. This is guaranteed by a model that generalize the factors that contribute to DPR time, as detailed in the next section.

III. The proposed system

The proposed system is constituted of a run-time manager that allows to face timing performance losses in edge-computing systems: in case of timing performance losses, the manager evaluates whether a DPR can help on recovering them by instantiating accelerators on FPGA. This kind of adaptation phenomena is well described by the self-adaptive loop model [17], shown in Fig. 1: for this reason, in next sub-sections it is taken as reference to describe the proposed manager. Moreover, without loss of generality, a reference platform for edge-computing is considered, shown in Fig. 2. With reference to it, a complete DPR operation is established when an accelerator is loaded within the reconfiguration area: this means that a reconfiguration file, called bitstream (BS), is transferred from external memory to dynamic reconfiguration memory (DRM). In more detail, BS is read from external memory by a processor system (PS) and transferred to its on-chip memory; then, PS transfers the BS to a dynamic reconfiguration branch (DRB), composed of a dynamic reconfiguration controller (DRC), a dynamic reconfiguration interface (DRI) and a dynamic reconfiguration memory (DRM). When the BS is stored in DRM, the DPR is considered completed.

A. Monitor

The monitoring action in the proposed run-time manager is implemented on the basis of the following considerations:

- edge-computing involves applications that can be executed on heterogeneous platforms; this means that the same monitoring system can be required to work, for example, on different processors/accelerators. This, in

turn, requires the adoption of a re-usable monitoring system.
- What needs to be monitored can be very different, depending on the executed application and on the metrics to be evaluated; this requires a flexible monitoring system, i.e., that can quickly customized for different measures.
- The monitoring action is performed on systems with limited computing resources and possible strict timing performance requirements; this requires a monitoring system that limits the intrusiveness on the application execution time.

For these reasons, the monitoring system adopted in the proposed run-time manager is implemented starting from a framework that enforces flexibility and re-usability of the produced components. The framework, called AIPHS [18] [19], allows to implement custom HW monitoring systems, in order to limit the intrusiveness, and it is based on a library of IP-cores.

B. Analyze

Analysis is a phase where a component, called analyzer, interprets the raw information coming from the monitoring system, in order to obtain indications about performance. Similar to monitoring, this phase strongly depends on the application under analysis. The analyzer in the proposed manager, again, is built using the AIPHS framework [19].

C. Plan

The planning phase, in the proposed run-time manager, involves a decision on whether to adapt the system or not, i.e., whether to apply DPR or not. It depends on two main factors: (i) the availability of an accelerator to regain performance and (ii) the necessary reconfiguration time (RT) to perform a DPR. Supposing that the first point is always true, the decision depends only on the second one: differently from previous works, the manager estimates the RT at run-time and with a fixed latency. The estimation of RT at run-time is required since it depends on two factors: the BS size and the available bandwidth. The BS size is a parameter known since design time: however, there can be cases where new BS can be generated to adapt the system along its lifetime, for example to update a system functionality to new standards, without changing the HW platform; in that case, the manager would keep into account the new BS size. On the other hand, available bandwidth is a parameter that depends on the whole path to be traversed to bring the bitstream from shared (eventually external) memory to reconfiguration memory: the bandwidth can be affected by shared memory arbitration and shared system bus arbitration.

Considering the reference platform shown in Fig. 2, the reconfiguration time can be calculated with the following formula:

$$RT = RT_{STO} + RT_{CNTR} + RT_{CONF} + RT_{SH} \quad (1)$$

where RT_{STO}, RT_{CNTR}, and RT_{CONF} are the times to move the bitstream, respectively, from external memory to processor memory, from processor memory to DRI and from DRI to DRM. On the other hand, RT_{SH} represents the impact factor of the sharing of memory and buses.

The planning is implemented by a decision maker that, in the proposed run-time manager, has three inputs: the output of estimator, the knowledge related to target and some information from execution phase (discussed in the next sub-section). The output of estimator allows checking, at run-time, whether the required timing performance is guaranteed. The knowledge related to target represents factors that contribute to RT, such as the status of shared elements. The information from execution phase is information related to configuration files related to accelerators to be dynamically downloaded on the FPGA: in this case, the information is given by the bitstream size.

The decision maker, in the proposed run-time manager, has a fixed latency on estimating RT, since it is completely implemented in HW inside the FPGA, as shown in Fig. 2.

D. Execute

In the execution phase, the adaptation to be applied to regain timing performance losses is produced. In this context, producing an adaptation means to enable the DPR with the selected bitstream. The configuration files are prepared at design-time, due to the complexity of the FPGA synthesis operation.

IV. Experimental Results

Dedicated experimental applications have been designed and developed to show the run-time manager profitability, i.e., the capability to adapt the system to possible application changes: it has to guarantee that timing performance are hold, in a scenario where *asynchronous* disturbances, i.e., that happens independently from the application, are present. An application running on top of an edge-computing platform has been also designed and developed, together with the proposed run-time manager. In order to show the effectiveness of the proposed manager, it has been assumed, without loss of generality, that the accelerator is not already present on FPGA. This is a customary assumption for systems that will use this type of manager, e.g., systems with a lot of accelerators that represent HW tasks for processors.

A. Selected platform and model instantiation

The selected edge-computing platform is a Xilinx Zynq-7000 SoPC [6], soldered on a Digilent ZedBoard [20], whereas the development tool is Xilinx Vivado IPI [21]. For a bitstream stored inside ZedBoard DDR3 external memory, a DPR can be managed by ARM Cortex A9 core: depending on the dynamic reconfiguration interface (DRI), that can be either Processor Configuration Access Port (PCAP) or Internal Configuration Access Port (ICAP), a different reconfiguration path is involved. By means of a comparison between Fig. 2 and Zynq7000 datasheet [6], it is possible to identify the reconfiguration path. Supposing the use of PCAP, starting from the external shared memory, there are three levels of arbitration inside the memory controller, in order to handle the memory

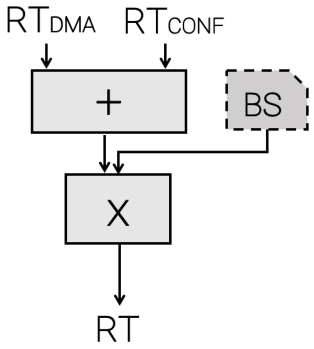

Fig. 3. Circuit schematic of the DPR time estimator.

accesses among 4 ports; each port has two 64 bit channels (for reading and writing) [6]. Then, there is a system bus that supports a burst type access of 8 beats size, each one with 64 bit data width [6]. The dynamic reconfiguration branch (DRB) is composed of a dynamic reconfiguration controller (DRC), represented by a Direct Memory Access (DMA) contained within a Device Configuration Controller (DevC) [6], and a PCAP. The DMA supports a burst type access of 8 beats size, each one with 32 bit data width [6]; it stores data in a memory buffer, represented by a queue with a depth of 127 32-bit words. The PCAP controller takes data from the queue and pushes them into reconfiguration memory [6]. All these operations are done working at different clock domains.

In the first version of the model, an exclusive access path to external memory has been assumed: this assumption has been implemented by acting on memory arbiter configuration. This means that, in (1), RT_{SH} is zero. Basing on the path above described, the instantiated model provides RT_{STO}, RT_{CNTR} and RT_{CONF}. It is worth noting that RT_{STO} and RT_{CNTR} are merged in a unique parameter, named RT_{DMA}, since there is a DMA that makes direct access to memory, without passing through processor. RT_{CONF} is the parameter related to loading of reconfiguration memory using PCAP. RT_{DMA} and RT_{CONF} are added together and, then, multiplied by bitstream size. The result, representing RT, is provided as input to the decision maker of the run-time manager. The final circuit schematic that implements the model is shown in 3. When implemented on the FPGA side of Zynq7000, it makes use of one DSP. Putting it in the decision maker, also one comparator needs to be considered.

B. First test

In the first proposed test, an application constituted of some matrix operations, organized in batches, has been executed on one of the ARM-Cortex A9 cores, and an asynchronous disturbance task, composed of a number of sums, has been provided, enabled by a push-button. Two timing performance requirements have been provided: (i) the execution time of each batch equal at maximum to *1.5 ms* and (ii) the deadline of the application equal to *35 ms*. An accelerator using the classical DPR flow proposed by Xilinx has been prepared and the related BS stored within the external DDR3 memory;

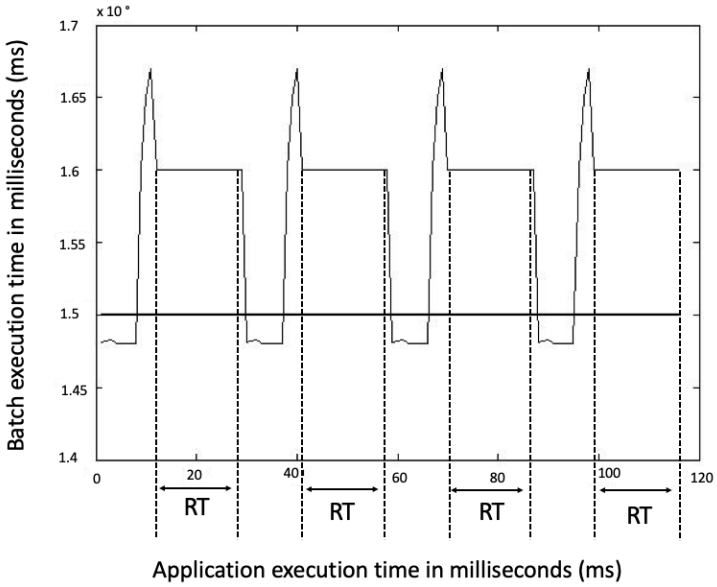

Fig. 4. The figure shows the results plot of the first test. Initially, the application is totally executed on one of the dual-core ARM Cortex A9 and monitored by the run-time manager to evaluate the execution time. If it goes over the required 1.5 ms, the manager evaluates the DPR time using the proposed model, and it checks whether the DPR is profitable or not by considering how much time remains before the deadline and what is the available speed-up. It is possible to observe that there is a peak at 1.65 ms when disturbance is triggered; then, the manager evaluates the profitability in 50 ns and it performs a DPR. The time between the peak and the return of batch execution time under 1.5 ms represents the reconfiguration time (RT). The application is launched four times, and the manager decides, in all four cases, to perform the DPR, since the task can be completed in time.

the accelerator provides a speed-up of 2x for each batch execution. When a disturbance was triggered, the manager was responsible for guaranteeing either the required quality of service or to give an alarm in 50 ns. Results of first test are reported in Table I, and shown in Fig. 4.

TABLE I
RESULTS OF THE FIRST PROPOSED TEST

Parameter	Value
Maximum execution time for each batch	1.5 ms
Deadline of application	35 ms
Time to response for manager	50 ns
Estimated reconfiguration time	18934577 ns
Actual reconfiguration time	18759380 ns
Absolute/Relative estimation error	175197 ns / 1%

C. Second test

In the second proposed test, the run-time manager has been introduced within a framework called Artico3 [5], with the future goal of making it more accessible to designers.

TABLE II
RESULTS OF THE SECOND PROPOSED TEST

Parameter	Value
Bitstream size	857740 bytes
Estimated reconfiguration time	6754702 ns
Actual reconfiguration time	6588656 ns
Absolute/Relative estimation error	166046 ns / 2.5%

Artico3 is a framework to develop edge-computing platforms that make use of FPGA and DPR. The tool is provided for applications running in Linux user space: in order to work with the proposed run-time manager, it has been adapted to work with bare-metal applications on ZedBoard. In this test, only the quality of reconfiguration time prediction has been evaluated; results are reported in Table II.

V. DISCUSSION AND CONCLUSIONS

This paper proposes an innovative run-time manager that handles the self-adaptivity of CPSs at the edge, by means of a model to estimate, in advance and at run-time, the DPR time, in order to actually evaluate when the DPR is profitable. The proposed model is quite general and it has been proposed since, in the state of art, there are a number of local solutions to the DPR profitability evaluation, and only part of them consider the impact of DPR time. The paper proposes the manager, describes the model and instantiate it for a Zynq7000 platform. However, it is reusable for other platforms, with opportune modifications. Observing Table I and Table II, it can be noticed that the model is accurate, providing a maximum error of 2.5 % respect to actual reconfiguration time. The effectiveness of the manager on providing adaptation by means of a DPR can be noticed by showing Fig. 4. First validation activities have been done considering simple applications and, in this context, future works will include the manager in more complex scenarios with edge-computing, where adaptivity is required to face performance variations. In this context, preliminary works have been done on developing HW platforms for unmanned aerial vehicles able to guarantee certain timing performance on image computation using DPR; the platform development has been done using Artico3 with the proposed run-time manager.

ACKNOWLEDGMENT

This work has been partially supported by the ECSEL RIA 2017 FITOPTIVIS project. Authors would like to thank you ARTICO 3 developers, in particular Alfonso Rodriguez, for the support on using the tool.

REFERENCES

[1] Y. Z. Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. D. Benedetto, "State of the art of cyber-physical systems security: An automatic control perspective," *Journal of Systems and Software*, vol. 149, pp. 174 – 216, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121218302681

[2] H. Muccini, M. Sharaf, and D. Weyns, "Self-adaptation for cyber-physical systems: A systematic literature review," in *2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, May 2016, pp. 75–81.

[3] 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 *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, Sep. 2014, pp. 1–4.

[4] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017.

[5] A. Rodriguez, J. Valverde, J. Portilla, A. Otero, T. Riesgo, and E. de la Torre, "Fpga-based high-performance embedded systems for adaptive edge computing in cyber-physical systems: The artico3 framework," *Sensors*, vol. 18, p. 1877, 06 2018.

[6] (2018) Zynq-7000 soc technical reference manual (ug585). [Online]. Available: \url{https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf}

[7] (2019) Intel cyclone v soc fpgas. [Online]. Available: \url{https://www.intel.it/content/www/it/it/products/programmable/soc/cyclone-v.html}

[8] (2018) Xilinx partial reconfiguration user guide. [Online]. Available: \url{https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug909-vivado-partial-reconfiguration.pdf}

[9] F. Palumbo, T. Fanni, C. Sau, and P. Meloni, "Power-awarness in coarse-grained reconfigurable multi-functional architectures: a dataflow based strategy," *Journal of Signal Processing Systems*, vol. 87, no. 1, pp. 81–106, Apr 2017. [Online]. Available: https://doi.org/10.1007/s11265-016-1106-9

[10] A. Biondi, A. Balsini, M. Pagani, E. Rossi, M. Marinoni, and G. Buttazzo, "A framework for supporting real-time applications on dynamic reconfigurable fpgas," in *2016 IEEE Real-Time Systems Symposium (RTSS)*, Nov 2016, pp. 1–12.

[11] M. Pagani, M. Marinoni, A. Biondi, A. Balsini, and G. Buttazzo, "Towards real-time operating systems for heterogeneous reconfigurable platforms," in *Conference on Operating Systems Platforms for Embedded Real-Time applications (OSPERT)*, 07 2016, p. 6.

[12] K. Papadimitriou, A. Dollas, and S. Hauck, "Performance of partial reconfiguration in fpga systems: A survey and a cost model," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 4, no. 4, pp. 36:1–36:24, Dec. 2011. [Online]. Available: http://doi.acm.org/10.1145/2068716.2068722

[13] C. Claus, B. Zhang, W. Stechele, L. Braun, M. Hubner, and J. Becker, "A multi-platform controller allowing for maximum dynamic partial reconfiguration throughput," in *2008 International Conference on Field Programmable Logic and Applications*, Sept 2008, pp. 535–538.

[14] M. Damschen, L. Bauer, and J. Henkel, "Corq: Enabling runtime reconfiguration under wcet guarantees for real-time systems," *IEEE Embedded Systems Letters*, vol. 9, no. 3, pp. 77–80, Sep. 2017.

[15] 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 *2015 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, June 2015, pp. 1–8.

[16] S. Garcia and B. Granado, "Ollaf: A fine grained dynamically reconfigurable architecture for os support," *EURASIP Journal on Embedded Systems*, vol. 2009, no. 1, p. 574716, Nov 2009. [Online]. Available: https://doi.org/10.1155/2009/574716

[17] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan 2003.

[18] A. Moro, F. Federici, G. Valente, L. Pomante, M. Faccio, and V. Muttillo, "Hardware performance sniffers for embedded systems profiling," in *2015 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES)*, Oct 2015, pp. 29–34.

[19] G. Valente, V. Muttillo, L. Pomante, F. Federici, M. Faccio, A. Moro, S. Ferri, and C. Tieri, "A flexible profiling sub-system for reconfigurable logic architectures," in *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, Feb 2016, pp. 373–376.

[20] (2014) Zedboard hardware user's guide. [Online]. Available: \url{http://zedboard.org/sites/default/files/documentations/ZedBoard_HW_UG_v2_2.pdf}

[21] (2017) Vivado design suite. [Online]. Available: \url{https://www.xilinx.com/products/design-tools/vivado.html}