DEMO: Multi-Grain Adaptivity in Cyber-Physical Systems

Alfonso Rodríguez*, Tiziana Fanni†

*Universidad Politécnica de Madrid, Email: alfonso.rodriguezm@upm.es

†Università degli Studi di Cagliari, Email: tiziana.fanni@diee.unica.it

Abstract—Cyber-Physical Systems (CPS) operate in increasingly complex and demanding application scenarios, while requiring also high adaptivity levels to satisfy several requirements that usually change over time. Multi-grain reconfigurable hardware architectures are an appealing solution to provide high, heterogeneous flexibility, thus reaching the advanced runtime adaptivity support necessary for CPS. This work presents a demonstration of an automated framework for the development and runtime management of multi-grain reconfigurable hardware systems. The framework supports different reconfiguration mechanisms, each with different overheads and depth in terms of system behavior modification.

I. CONTEXT AND OBJECTIVES

The H2020 CERBERO European Project aims at developing a continuous design environment for Cyber-Physical Systems (CPS), including modeling, deployment and verification (<http://www.cerbero-h2020.eu/>). CPS are complex systems composed of different interactive components, which need to meet several requirements imposed by the environment, the user and even their internal status. The presence of different, concurrent requirements influencing the system during operation introduces the need for an advanced adaptivity support. The efficient support for run time reconfiguration, taking into account an uncertain environment with changing requirements, is among the CERBERO expected outcomes. In particular adaptivity triggers are classified as:

- functional-oriented: adaptation needed to offer different functionalities over the same substrate or to maintain correct functionality. It may be parametric (e.g., a constant changes) or fully functional (e.g., the algorithm changes).
- non-functional-oriented: functionality is fixed, but system requires adaptation caused by non-functional requirements, such as performance or available energy.
- repair-oriented: for safety and reliability purposes, adaptation may be used in case of faults. Adaptation may add self-healing or self-repair features.

Reconfigurable systems provide a valuable solution to this problem: lying in the middle between general purpose computing platforms and application-specific circuits, they offer a tradeoff between software-like flexibility and hardware-based execution performance. According to the type of reconfiguration adopted, these tradeoffs can be highly different. For instance, the fine-grain reconfigurable systems offer high computing flexibility, being able to totally change the system behavior, at the cost of high time and power

overheads. Dynamic Partial Reconfiguration (DPR) mitigates those limitations by reducing the size of the substrate partition that changes its functionality while the rest of the systems continues its execution. In these architectures the deployed circuit is always optimized in terms of resource usage and frequency, but reconfiguration still requires a certain amount of time and an associated power consumption that is proportional to the size of reconfigurable partitions [1].

At coarser level of granularity are placed systems where reconfiguration is virtual (all the resources belonging to all the possible configurations are always instantiated in the substrate), in this case we talk about Virtual Reconfigurable Circuits (VRC). Reconfiguration is achieved by multiplexing the resources in time, switching among configurations. The outcome is high-performance and fast reconfiguration, while the main drawbacks are related to: (1) maximum operating frequency, which is less than or equal to the one of the isolated configurations; (2) area utilization, due to the presence of resources that are not involved in the configuration that is active; (3) flexibility, being able to compute only the functionalities the system has been designed for [2].

Given the different cost of the above described two types of reconfiguration their combination would bring the best of both, enabling adaptive multi-grain reconfigurable fabrics, which can meet the changing functional and non-functional requirements of CPS designs. This paper describes a demonstrator built with an automated framework for the development and runtime management of multi-grain reconfigurable hardware systems. The toolchain used goes from high-level description of the applications to be executed to the hardware implementation of multi-grain adaptive systems, where different DPR-enabled reconfigurable partitions are filled in with heterogeneous and irregular application-specific VRCs.

Adaptivity and reconfiguration have been actively studied at different levels of granularity. For instance Morpheus [3] constitutes heterogeneous reconfigurable systems on chip that integrates application-oriented reconfigurable cores for implementing applications with varying characteristics on the same hardware architecture and a toolchain that, eases the applications implementation through a software oriented approach. Yuan et al. [4] present a multi-grain FPGA aimed for mobile computing and focused on two key steps towards higher efficiency: interconnection network and coarse-grain reconfigurable digital signal processors. The chip incorporates fine-grain configurable logic blocks, medium-grain digital sig-

nal processors along with reconfigurable block RAMs, and two coarse-grain kernels. Liu et al. [5] proposed a hybrid-grained reconfigurable architecture (HReA) to process 13-Dwarfs computation. HReA combines 32-bit coarse-grain data path with 1-bit fine-grain data path to accommodate co-existence of multiple computing granularities in 13-Dwarfs. The two data paths with different granularities can interact with each other in an ALU calculation.

The above-described architectures exploit coarse-grain reconfigurable arrays where, in most of the cases, the processing elements are not directly derived from the applications to be accelerated. This lack of specialization provides high flexibility, but may limit performance, especially if interconnection infrastructures are used. The multi-grain architecture described in this paper is flexible enough to be suitable to support different adaptivity types: functional, non-functional and repair-oriented ones.

II. HARDWARE RECONFIGURATION IN THE H2020 CERBERO EUROPEAN PROJECT

In CERBERO, two tools offer support for hardware reconfiguration: ARTICo³ and MDC. The ARTICo³ framework provides adaptive and scalable hardware acceleration by using DPR [6]. ARTICo³ is able to support runtime tradeoffs between computing performance, energy consumption and fault tolerance. The MDC design suite, on the other hand, delivers VRCs based on the dataflow model of computation, and has been already proved to be a viable solution to reach adaptivity in CPS, trading off among functional and non-functional requirements [7].

A. The Multi-Dataflow Composer Tool

MDC tool [7] is meant to address the difficulty of mapping a set of different applications onto a multi-functional accelerator, delivering the automatic generation and management of coarse-grain VRCs based on the dataflow model of computation. Figure 1 illustrate the MDC design flow; MDC front-end combines together a set of input dataflow specifications (describing desired system configurations/behaviors), sharing actors through a datapath-merging problem-solving heuristic algorithm, then the MDC back-end generated the corresponding HDL code. Beside this baseline functionality, MDC is composed of three additional components:

- *Structural Profiler*: performing the design space exploration of the implementable multi-functional systems to determine the optimal VRC according to the given input constraints [8].
- *Dynamic Power Manager*: performing, at the dataflow level, the logic partitioning of the substrate to implement at the hardware level a power-saving strategy [9], [10].
- *Co-Processor Generator*: performing the complete dataflow-to-hardware customization of a Xilinx compliant multi-functional accelerator. Starting from the input dataflow specifications set, such an accelerator can be either loosely coupled or tightly coupled, according to the design needs, and also its drivers are derived [11].

Fig. 1. Multi-Dataflow Composer Tool - Merging Process

B. The ARTICo³ Framework

The ARTICo³ framework [6] provides adaptive and scalable hardware acceleration, actively altering the computing substrate to change the available functionality using DPR. It can be exploited to achieve user-driven runtime tradeoffs between performance, energy efficiency and fault tolerance by means of accelerator replication. The ARTICo³ execution model enables scalable task- and data-parallel execution of applications that can benefit from a processor-coprocessor approach.

The ARTICo³ framework is composed of three main components (whose interaction can be seen in Figure 2):

- a hardware architecture that exploits a DPR-enabled multi-accelerator computing scheme, with a configurable datapath that can be adapted to different processing profiles (e.g., parallel, redundant);
- a toolchain to automatically build custom accelerator-based computing systems starting from already partitioned hardware/software systems, and using either C/C++ or low-level HDL descriptions for the accelerated functionality;
- a *Runtime* library to transparently manage application execution and computation offloading to the hardware accelerators, used to hide from the user both the deployment of the scalable parallelism of the application and the DPR-based reconfiguration processes.

III. DEMONSTRATION

ARTICo³ has been demonstrated to be able to support runtime tradeoffs between computing performance, energy consumption and fault tolerance [6], while MDC already proved to be a viable solution to provide adaptivity in CPS, trading off functional and non-functional requirements [7]. Given the nature of the underlying hardware reconfiguration mechanism, ARTICo³ and MDC show opposite tradeoffs

Fig. 2. ARTICo³ framework components.

Integrated Design Flow

Runtime Setup

Fig. 3. Integrated Toolchain for Multi-Grain Reconfigurable Systems Design

between flexibility and reconfiguration overhead when considered individually. Their combination brings together all the benefits from both DPR and VRCs, leading to more flexible solutions that can cope with the changing of functional and non-functional requirements affecting CPS operating contexts.

The integration of ARTICo³ and MDC offers a unique toolchain capable of automatically implementing and managing multi-grain reconfigurable systems, while at the same time offering support for advanced adaptivity. Figure 3 shows the integrated design flow (left-hand side) and the runtime setup of the ARTICo³/MDC resulting integration (right-hand side). The hardware generation flow starts from high-level dataflow descriptions of the configurations/behaviors to be implemented in the configurable logic, and the integrated toolchain derives the corresponding virtual-reconfigurable HDL computational kernel, properly wrapping it with the glue logic necessary to serve as an ARTICo³ DPR reconfigurable partition. Finally, the

toolchain generates the bitstreams related to the system (static part) and to the hardware accelerators (reconfigurable partitions). On the software side, the toolchain keeps the capability, inherited from the ARTICo³ framework, of generating the application executable that manages operation execution and computation offloading to the hardware accelerators also when these latter are MDC-generated VRCs. Both reconfiguration mechanisms are transparently managed by the user code running in the host processor.

In this demonstrator, the multi-grain reconfiguration capabilities of the combined MDC-ARTICo³ flow are shown in an image-processing application scenario. The setup features are: ARTICo³ on a Zynq board running Linux, a camera to acquire live video and a monitor to show the resulting output. The input images are sent to a configurable number of hardware accelerators where different edge detection kernels have been implemented. In order to switch from one kernel to another, the user can decide to use the DPR approach of ARTICo³ to completely change the logic instantiated in each slot, or to use the Virtual CGR approach of the MDC-generated accelerators to multiplex the internal datapath of the accelerators. As a result, it is possible to see, in real time, the runtime overheads of each type of reconfiguration mechanism. Additional adaptivity evaluation can be performed by changing the working point of the application, which is based on several parameters: input image size, number of hardware accelerators used to exploit data-level parallelism, and hardware redundancy level (Simplex, DMR, TMR) for fault-tolerant execution. The demonstrator GUI can be seen in Figure 4.

A summary of some experimental results obtained when switching the working point of the system are presented. In this case two simple edge detectors, Sobel and Roberts, have been considered. Table I depicts processing time employed by one hardware accelerator to compute Sobel and Roberts algorithms on a 512x512 input image. Results are related to the

Fig. 4. GUI of the demonstrator. On the left, both original and processed images are shown. On the right, reconfiguration control and timing reports are shown for processing, DPR and VRC (CGR).

TABLE I
PROCESSING TIME AND RECONFIGURATION OVERHEADS: SOBEL AND ROBERTS KERNELS (IMAGE SIZE 512X512).

	Processing Time [ms]		Overhead Time [ms]	
	<i>standalone</i>	<i>reconfigurable</i>	<i>standalone</i>	<i>reconfigurable</i>
<i>Sobel</i>	151	148	16.7	0.056
<i>Roberts</i>	150	151	17.5	0.053

execution of two (one for each algorithm) non-reconfigurable accelerators (*standalone*, regular ARTICO³ accelerators), and of one VRC accelerator (*reconfigurable*) implementing the two functionalities together. *Overhead Time* column shows the time necessary to reconfigure the system (i.e. change between edge detection kernels). Graphs in Figure 5 show the time trend on processing two images with different sizes (256x256 and 512x512) when the number of adopted accelerators changes.

Fig. 5. Effect on processing time of the parallelism scalability.

Acknowledgments: This work has received funding from the EU Commissions H2020 Programme under grant agreement No 732105. The authors would like to thank the Spanish Ministry of Education, Culture and Sport for its support under the FPU grant program.

REFERENCES

- [1] M. Lombardo *et al.*, "Power management techniques in an fpga-based wsn node for high performance applications," in *Workshop on Reconfigurable and Communication-Centric Systems-on-Chip*, 2012.
- [2] R. Hartenstein, "Coarse grain reconfigurable architecture (embedded tutorial)," in *Conf. of the Asia and South Pacific Design Automation*, 2001.
- [3] N. Voros *et al.*, "Morpheus: A heterogeneous dynamically reconfigurable platform for designing highly complex embedded systems," *ACM Trans. Embed. Comput. Syst.*, vol. 12, pp. 70:1–70:33, Apr. 2013.
- [4] F. L. Yuan *et al.*, "A multi-granularity fpga with hierarchical interconnects for efficient and flexible mobile computing," *IEEE Journal of Solid-State Circuits*, vol. 50, pp. 137–149, Jan 2015.
- [5] L. Liu, Z. Li, C. Yang, C. Deng, S. Yin, and S. Wei, "Hrea: An energy-efficient embedded dynamically reconfigurable fabric for 13-dwarfs processing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, pp. 381–385, March 2018.
- [6] A. Rodríguez *et al.*, "Fpga-based high-performance embedded systems for adaptive edge computing in cyber-physical systems: The artico3 framework," *Sensors*, vol. 18, no. 6, 2018.
- [7] C. Sau *et al.*, "Challenging the best hevc fractional pixel fpga interpolators with reconfigurable and multi-frequency approximate computing," *IEEE Embedded Systems Letters*, vol. 9, no. 3, pp. 65–68, 2017.
- [8] F. Palumbo *et al.*, "Power-awareness in coarse-grained reconfigurable multi-functional architectures: a dataflow based strategy," *Journal of Signal Processing Systems*, 2016.
- [9] T. Fanni *et al.*, "Automated Power Gating Methodology for Dataflow-based Reconfigurable Systems," in *Conference on Computing Frontiers*, 2015.
- [10] T. Fanni *et al.*, "Power and Clock Gating Modelling in Coarse Grained Reconfigurable Systems," in *Conference on Computing Frontiers*, 2016.
- [11] C. Sau *et al.*, "Reconfigurable coprocessors synthesis in the MPEG-RVC domain," in *Conference on ReConFigurable Computing and FPGAs*, 2015.