Pre-print version. The final publication is available at IEEE via
https://doi.org/10.1109/RECONFIG.2017.8279772

C. Rubattu, F. Palumbo and M. Pelcat, "Adaptive software-augmented hardware reconfiguration with dataflow design automation," 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), Cancun, 2017, pp. 1-4. doi: https://doi.org/10.1109/RECONFIG.2017.8279772

Adaptive Software-Augmented Hardware Reconfiguration with Dataflow Design Automation

Claudio Rubattu[1], Francesca Palumbo[1], and Maxime Pelcat[2]

[1] University of Sassari, 07100 Sassari, Italy
(crubattu,fpalumbo)@uniss.it
[2] Univ Rennes, INSA Rennes, IETR, UMR CNRS 6164, 35708 Rennes, France
mpelcat@insa-rennes.fr

Abstract. Demand of adaptive hard-constrained devices is continuing to increase. Developing efficient implementations of such systems means to address trade-offs among different specifications, i.e. real-time processing, low power consumption and partial context switching at runtime. In this PhD Plan, we will focus on the hardware perspective presenting how we intend to study and experience with adaptive co-processing architectures, considering software as a supporting element.

1 Introduction

Systems-on-a-Chip (SoCs) embed a growing number of reconfigurable and heterogeneous processing units. The design of devices based on SoCs has become increasingly complex and new abstractions and languages aiming at limiting this complexity have appeared. The most common state-of-the-art approach, for example adopted by the OpenCL language, focuses on software and considers hardware (FPGA or specialized cores) as a "necessary evil" increasing the local performance of a predominantly software system at the cost of a higher design complexity. This approach can be qualified as *hardware-augmented software*.

This thesis is imagined as a counterpoint to the state-of-the-art approaches and considers hardware as the main element of the computation, while software comes as an enabler for services hardly supported by hardware. It can be qualified as *software-augmented hardware*. This approach makes sense in environments where constraints (very limited energy, strict real time, very high reliability, etc.) impose performance infeasible in software. Moreover, we imagine this approach as fundamental in highly dynamic environments, where changing trade-offs among such a given tight constraints have to be guaranteed at runtime.

The PhD plan we are presenting will be part of the activities of the H2020 CERBERO European project[3], started in January 2017, whose overall objective is to create a design environment for Cyber-Physical Systems (CPS). The CERBERO project is based on two main elements [1]:

[3] http:http://www.cerbero-h2020.eu/

– a cross-layer and model-based approach to describe, optimize and analyze the system according to different views; and
– an extended adaptivity of the calculation to the system state as well as to its environment, adaptivity provided by an autonomous reconfiguration engine.

The thesis we are presenting here, which will start in October 2017, covers parts of the second CERBERO element and will study the ability of the system to dynamically reconfigure itself according to its state and to its environment. From the hardware perspective, this would require to model and implement efficient reconfigurable co-processing units (see Section 2), flexible enough to switch among different working points, upon external requests, and smart enough to be capable of automatically handle reconfigurability, by means of self-reconfiguration.

The starting point of this thesis is the dataflow Model of Computation (MoC), an intrinsically parallel alternative to the imperative languages of common use (C, C++, Java, etc.). Dataflow representations make it possible to separate temporal problems and functional problems during hardware design. They also foster a natural separation of the computation into parallel blocks whose mapping onto highly heterogeneous architectures is possible automatically. Thus, according with the CERBERO model-based approach to system design, our idea is to leverage on a dataflow-to-hardware design flow to implement, optimize and manage dataflow-based self-reconfigurable systems. This thesis will be based on more than fifteen years of research in dataflow and hardware design at INSA Rennes and UNISS. It will also be part of a long-term approach that led to the PREESM and MDC softwares, which are going to be described in Section 3.

Last but not least, to provide the requested flexibility level, which is among the primary requirements of modern systems architectures, methods as Coarse-Grained Reconfiguration (CGR) and Dynamic Partial Reconfiguration (DPR) are going to be adopted. These methodologies, discussed in Section 4, present different pros and cons and can certainly be combined with the hardware co-processing units to improve the adaptivity support in *software-augmented* infrastructures.

Besides concluding in Section6 with some final remarks on the assessment of the envisioned methodologies, in Section 5 we will introduce the preliminary PhD plan with all the envisioned and already started activities.

2 Co-Processing Architectures

The core of this PhD plan is based on the idea that purely software execution, in modern system devices, may be too costly both in terms of execution time and resource usage. Referring to *software-augmented hardware* we intend to set-up a design environment where dedicated hardware co-processing units are coupled to the main processor, and are basically responsible of most of the processing. Host processor can be seen as an enabler that delegates them (according to functional needs and execution constraints) the computation. Study, implementation and dynamic management of such co-processing units depend on the application

characteristic and on the communication management (between host and the co-processing units), according to the selected coupling level.

Generally speaking, co-processing units offer different degrees of coupling with respect to the host processor they are connected to. Loosely coupled units are accessed via the system bus and are typically affected by medium/high communication latencies for both control and data transfers. Memory-mapped co-processing units implement this kind of coupling. Tightly coupled units are accessed by means of dedicated full-duplex links and often share with the host processor high-level memories. Stream-based co-processing units implement this kind of coupling.

Stream-based solutions are naturally compliant with dataflow-based approaches to system specification. On the contrary, the tight coupling makes their porting among targets practically impossible, since dedicated links and memory accesses are used. Memory-mapped co-processors are more generic, but they do not suite stream-based processing typical of signal processing applications: they do not favor following the natural flow of data and do not allow taking advantage of the logical pipelining of this kind of applications.

To offer effective solutions for real *software-augmented hardware* implementations in this PhD Thesis, we will:

- **Analyse and model different co-processing solutions** - Figure 1 depicts different combinations of stream-based and memory-mapped models that can be implemented in a co-processing unit. In this architecture, only one co-processor executing the whole dataflow network is present. However, several co-processing units could be instantiated in order to address part of the processing. Also, it is not necessary splitting workload equally between them. Regarding the communication protocol, system components (host, co-processing unit and actors of the dataflow network) can use their FIFO- and/or bus-based interface (if they are present). In Figure 1, a purely FIFO-based approach is selected, within the co-processing unit and to connect it with the host. This strategy allows parallelizing transfers (if more than one port is present), avoiding bottlenecks. Nevertheless, a memory-mapped model is more software-oriented. Indeed, whether inputs and outputs are not dependent upon stream-based transfers (third and fourth implementation options within the co-processor in Figure 1), dedicated communication channels or protocol adapters (i.e. AXI DMA in Vivado) are not required.
- **Develop novel approaches** - Implementation of a generic memory-based stream friendly communication infrastructure to enable distributing a given computing network into multiple groups of actors, mapped over different co-processing units, interconnected by generic memories configured to act as FIFOs. This mimic in hardware what is normally done by software dataflow-based executions and allows adapting the co-

processing infrastructure to the natural flow of data by fully exploiting the intrinsic pipelining of the given applications.

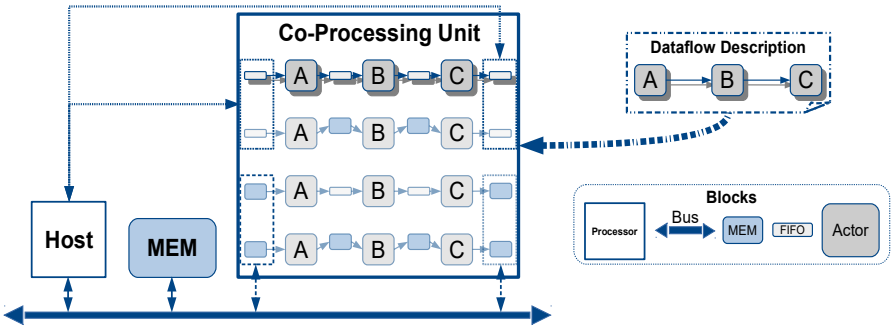

Fig. 1. Generic co-processing architecture. The co-processor unit internally shows 4 different possible implementations that consider all the combinations of memory accesses of the actor-to-actor communication infrastructure and of the host-to-co-processor one. Co-processor Input(s) and output(s) are connected either to the FIFO or to the bus interfaces (depending on the host-to-co-processor communication scheme).

3 CAD Support

To design and handle co-processing units, in this PhD Thesis we intend to leverage on two tools: MDC and PREESM. The first one is going to provide the hardware support; the second one the design-time profiling and software support.

The Multi Dataflow Composer (MDC) Tool [4] is a design suite for the development of coarse-grained reconfigurable systems based on the RVC-CAL dataflow MoC with the capability, by exploiting one of its extensions [2], of generating co-processing units. The baseline MDC functionality is split into sub-functionalities/components:

- the Multi-Dataflow Generator (MDG) - a model-to-model compiler that, given an input set of dataflow specifications (functionalities to be executed in hardware), derives a unique high-level multi-dataflow specification of the system leveraging on datapath merging techniques;
- the Platform Composer (PC) - a dataflow-to-hardware synthesizer that, given the multi-dataflow description generated by the MDG, the hardware

[4] `http://sites.unica.it/rpct/risultati-ricerca/`

blocks capable of implementing its actors and the communication protocol to be used among them, deploys the RTL description of a reconfigurable datapath.

The MDC co-processor generator extension is currently capable of taking the generated reconfigurable datapath and embedding it onto ready-to-use platform-dependent (Xilinx FPGA) IPs for their rapid testing and usage as co-processing units. These units can be configured by the users to offer either memory-mapped or stream-based communication support.

Another objective of this PhD Thesis is to:

- **Extend MDC co-processing support**: to be able to handle all the communication schemes (studied in the PhD time-frame) and to make it platform-agnostic.

PREESM [5] is an open source rapid prototyping tool. It simulates signal processing applications, that are described using a dataflow-based language, and generates code for heterogeneous multi/many-core embedded systems. Its main functionalities currently are:

- A fully automated mapping of computational tasks (actors) to multiple processing cores with the objective of optimizing statically the execution latency and the load balancing of cores.
- A state-of-the-art and fully automated optimization of the generated application memory footprint.

This PhD Thesis will be strongly connected to PREESM extension activities to be carried out within the CERBERO project, in particular:

- **Management of dedicated hardware co-processing units**: PREESM will be connected with MDC to enable delegating specific computations (an actor, a network of actors or a set of networks) to proper co-processing units.

4 Adaptation Strategies

Runtime hardware reconfiguration is required in order to change the system functionalities depending on specific events. With respect to the desired adaptations, basically speaking, hardware reconfiguration can be implemented by means of fine- or coarse-grained approaches, which pros and cons are presented in Table 1.

[5] `http://preesm.sourceforge.net`

Table 1. Comparison between CGR and DPR

Reconfiguration	Overhead			Working Point
	Time	Power	Resources	Flexibility
CGR	low	low	low	low
DPR	high	high	high	high
CGR + DPR	medium	medium	medium	high

Dynamic Partial Reconfiguration (DPR) allows fine-grained variations on selected areas of an FPGA device at runtime [3]. Indeed, partial context switching with different power constraints is possible considering DPR.

Coarse-Grained Reconfigurable (CGR) approach allows just a limited number of contexts among which you can switch, which makes DPR superior in flexibility with respect to the number of implementable functionalities [4]. However, using CGR architectures leads to reduce overhead problems related to the reconfiguration time, power consumption and hardware resources. Indeed, in case of DPR-based designs, additional memory resources have to be considered in order to contain the bitstreams for each configuration [5]. In MDC environment, a CGR implementation allows quick reconfiguration at runtime by setting configuration parameters of the network.

As it will be clarified in Section 5, important objectives of this PhD Thesis are:

- **Analyze and compare reconfiguration strategies**: The CERBERO project will facilitate to meet this objective, since there are activities on combining the ARTICo3 framework (dealing with DPR accelerators) with PREESM [6]. Therefore, comparison among CGR and DPR methods in co-processing units will be quite straightforward.
- **Definition of hybrid reconfigurable co-processing units**: embedded systems design often requires to find compromises among solutions. Our idea is to try to mix together these approaches on FPGA targets to offer flexibility at limited resource and power overhead.

5 PhD Challenges

The first challenges that this PhD Plan will have to address are summarized in the following list of open issues, with related PhD activities list.

1. *How dynamic can a piece of hardware be by adopting Coarse-Grained Reconfiguration only?* - Dynamic partial reconfiguration, as discussed above, basically allows to re-loading on the given partitions a brand new bitstream, which means a complete substitution of the executed dataflow network. Nevertheless, considering that we are targeting embedded hardware, enabling the

storage of several different bitstreams may easily lead to an unsustainable overhead, especially if the granularity of the reconfiguration is at the single actor level. To answer this question we plan to:
   - Compare CGR and DPR, to assess their respective pros and cons, given the CERBERO project application scenarios. We already made some studies on the combination of CGR and functional approximate computing [7], where we demonstrated that CGR is particularly effective in providing dynamic trade-off management with little to none overhead, which is something that is certainly not true for DPR.
   - Evaluate the possibility of combining these techniques by enabling partial reconfiguration among CGR datapaths. We strongly believe that combining DPR and CGR in accelerators definition may be particularly useful. CGR can be used to serve rapid and frequent functional reconfiguration or to adjust the working point, while DPR may be used to change the entire accelerator (given that a certain set of functionalities is not required anymore).
2. *Which are the most suitable CGR strategies to adapt to failures and/or to be able to effectively self-address external reconfiguration triggers?* - CERBERO project tackles uncertain hybrid environments, by proving system for deep sea monitoring and for planetary explorations. Failures and rapid/unpredictable changes of the execution constraints and use case needs have to be supported. We do not know yet which CGR solutions are most suitable to address those issues; therefore, within the PhD activities we want to:
   - Analyse and classify different approaches to determine the degree of flexibility they can expose [8].
   - Map the classified substrates with respect to the given scenario needs, to understand the limits of CGR platforms with respect to the possible reconfiguration triggers, i.e.
     - battery level is low, self-reduce consumption by X%.
     - co-processing unit N is corrupted, avoid using it, check if autonomous solutions are possible or notify the failure.
     - an highly critical task is executed, ensure total reliability for the next Y minutes.

At the CAD level, in the short term, we intend to fully integrate MDC and PREESM to enable design-time automatic system configuration. Complete automatic system deployment requires to generate both the software (PREESM generates the software and will have to handle the APIs to use MDC generated co-processing units) and the CGR co-processing units. MDC generates the co-processor template, but actors synthesis requires the usage of a dedicated compiler. To augment the portability of the envisioned solutions into different kinds of platforms, we already started to work on the integration of MDC with CAPH[6], a dataflow-to-hardware synthesis tool. The integration is meant to guarantee the composition and synthesis of completely platform-agnostic reconfigurable co-processing units.

[6] `http://caph.univ-bpclermont.fr/CAPH/CAPH.html`

6 Conclusions

This PhD plan, as already said, is centered around the activities of the CERBERO project, which includes three use cases: a self-healing system for planetary exploration (provided by: Thales Alenia Space), autonomous marine vehicles for monitoring isolated sites (provided by: AmbieSense), and an intelligent electric vehicle management system (provided by: TNO in cooperation with Centro Ricerche Fiat). In our opinion, the applications from Thales Alenia Space and AmbieSense are particularly suited to prove the *software-augmented hardware* approach we have in mind. These applications will be used to test the ideas developed in the PhD Thesis and to obtain methods both innovative and usable in practice.

Acknowledgment

The authors would like to thank the European Commission for funding the CERBERO Project (H2020 Programme, grant agreement 732105).

References

1. M. Masin et al, *Cross-layer design of reconfigurable cyber-physical systems*, Design, Automation & Test in Europe Conference & Exhibition, 2017. https://doi.org/10.23919/DATE.2017.7927088
2. C. Sau et al, *Reconfigurable Coprocessors Synthesis in the MPEG-RVC Domain*, Conf. on ReConFigurable Computing and FPGAs, 2015. https://doi.org/10.1109/ReConFig.2015.7393351
3. T. Kalb et al, *Enabling dynamic and partial reconfiguration in Xilinx SDSoC*, Conf. on ReConFigurable Computing and FPGAs, 2016. https://doi.org/10.1109/ReConFig.2016.7857168
4. T. Nowatzki et al, *Stream-Dataflow Acceleration*, Intl. Symp. on Computer Architecture, 2017. https://doi.org/10.1145/3079856.3080255
5. A.K. Jain et al, *Are Coarse-Grained Overlays Ready for General Purpose Application Acceleration on FPGAs?*, Intl. Conf. on Pervasive Intelligence and Computing, 2016. https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2016.110
6. L. Suriano et al, *Analysis of a Heterogeneous Multi-Core, Multi-HW-Accelerator-Based System Designed Using PREESM and SDSoC*, To appear in: Conf. on Reconfigurable Communication-centric Systems-on-Chip, 2017.
7. C. Sau et al, *Challenging the Best HEVC Fractional Pixel FPGA Interpolators with Reconfigurable and Multi-frequency Approximate Computing*, IEEE Embedded Systems Letters, 2017. https://doi.org/10.1109/LES.2017.2703585
8. M. Wijtvliet et al, *Coarse grained reconfigurable architectures in the past 25 years: Overview and classification*, Conf. on Embedded Computer Systems: Architectures, Modeling and Simulation, 2016. https://doi.org/10.1109/SAMOS.2016.7818353