

# AN EXPERIMENTAL TOOLCHAIN BASED ON HIGH-LEVEL DATAFLOW MODELS OF COMPUTATION FOR HETEROGENEOUS MPSOC

Julien Heulot, Karol Desnos, Jean François Nezan, Maxime Pelcat, Mickaël Raulet, Hervé Yviquel, Pierre-Laurent Lagalaye, Jean-Christophe Le Lann

▶ **To cite this version:**

Julien Heulot, Karol Desnos, Jean François Nezan, Maxime Pelcat, Mickaël Raulet, et al.. AN EXPERIMENTAL TOOLCHAIN BASED ON HIGH-LEVEL DATAFLOW MODELS OF COMPUTATION FOR HETEROGENEOUS MPSOC. DASIP, Oct 2012, Karlsruhe, Germany. hal-00749175

**HAL Id: hal-00749175**

**https://ensta-bretagne.hal.science/hal-00749175**

Submitted on 29 Oct 2013

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# AN EXPERIMENTAL TOOLCHAIN BASED ON HIGH-LEVEL DATAFLOW MODELS OF COMPUTATION FOR HETEROGENEOUS MPSOC

*J. Heulot, K. Desnos*
*J.-F. Nezan, M. Pelcat, M. Raulet*

INSA, IETR, UMR 6164, UEB
20 av. Buttes de Coësmes 35708 Rennes

*H. Yviquel*

IRISA, Univ. Rennes 1
6 rue de Kerampont
22300 Lannion

*P.-L. Lagalaye, J.-C. Le Lann**

Modaë Technologies
16, Rue Isaac Le Chapelier
35000 Rennes

## ABSTRACT

A chain of three state-of-the-art tools is demonstrated to generate efficient code for Multi-Processors System-on-Chips (MPSoCs) from a high-level dataflow language. The experimental platform is based on a 5-core Texas Instruments OMAP4 heterogeneous MPSoC running an image processing application.

***Index Terms*—** Embedded software, Multicore processing, Data flow computing, Signal processing

## 1. EXTENDED ABSTRACT

High-level languages respecting dynamic dataflow Models of Computation (MoCs) are convenient to specify an algorithm in a user-friendly fashion. However, static dataflow MoCs provide more compile-time knowledge of an application parallelism. This knowledge is necessary to produce efficient code for a MPSoC. In this paper, we demonstrate a transformation flow that eases the design of retargetable applications for heterogeneous MPSoC by transforming a high-level specification based on a dynamic dataflow MoC into an MPSoC-optimized application based on a static dataflow MoC. This flow consists of three dataflow-based tools: Modaë Studio, Open RVC-CAL Compiler (Orcc), and Preesm. The demonstrated hardware is based on a 5-core Texas Instruments OMAP4 heterogeneous MPSoC.

## 2. INTRODUCTION

Modern handheld embedded systems offer an increasing number of functionalities and processing capabilities while respecting a fixed power budget of a few Watts. Recent improvements in embedded systems are due to MPSoCs that combine general purpose cores, dedicated cores, and hardware accelerators within a single chip. Generating efficient code for heterogeneous MPSoCs remains a complex and error-prone task. This demonstration illustrates how dataflow MoCs, having precise semantics, favor interoperability between tools and can be used to program MPSoCs. Three dataflow-based tools (Section 3): Modaë Studio, Orcc, and Preesm, are combined to program a heterogeneous MPSoC from a high-level retargetable code with Kahn Process Network (KPN) MoC semantics [1]. The demonstrator is then presented in Section 4.

This work was supported by the ANR COMPA project
* J.-C. Le Lann is now with Labsticc-ENSTA Bretagne

## 3. MODEL TRANSFORMATION FLOW

The transformation flow presented in Figure 1 is performed by a tool chain composed of: Modaë Studio, Orcc, and Preesm. Each of these tools is successively used on a high-level description of the application in order to introduce more behavioral knowledge into its MoC. The transformation flow both offers a high-level front-end to the application designer, and generates an efficient mapping of the application on the targeted architecture.

**Modaë Studio**[1] is the first tool of this chain and is used as a schematic entry tool: a system is described as a set of interacting processes, communicating via channels, usually depicted as boxes and arrows. Channels link ports point-to-point. Different MoCs can be tagged on channels. Processes are organized as KPNs: processes communicate via infinite FIFOs, with blocking read semantics.

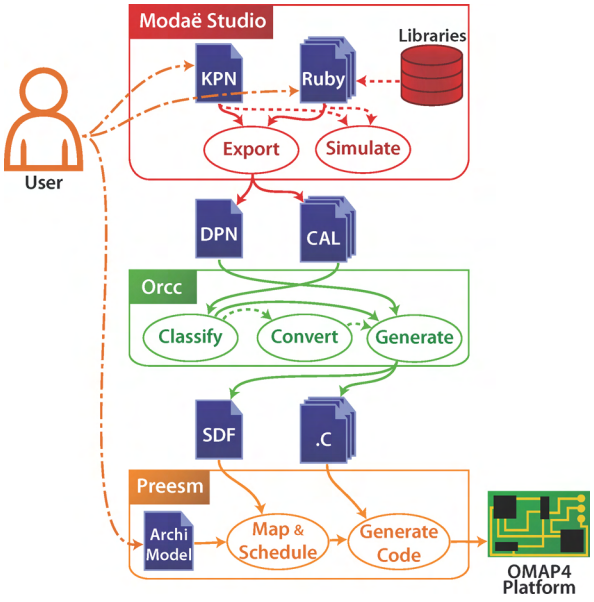



**Fig. 1**. Transformation Flow

The main novelty of Modaë approach relies on the resort to Ruby and Python interpreted languages to describe the algorithmic content of the processes. These languages are dynamically typed and user-friendly. Processes are described using an internal Domain Specific Language (DSL), a technique that popularized Ruby-on-rails web

[1] http://www.modae-tech.com

framework. Such an internal DSL facilitates a seamless integration of domain specific concepts into the host language. The object-oriented nature of Ruby and Python also simplifies the design of high-level libraries. Modaë Studio provides libraries for image and signal processing, but also for simple probe displays (xy plots, pixel grids, ...).

Once this closed system is captured, Modaë compiler and simulators retrieve the bit-accurate type of each variable, hence filling the gap between user representations and machine-oriented representations. In the experimental tool chain described here, the transformation engine generates a Dataflow Process Network (DPN) [1] in RVC-CAL dataflow language syntax [2].

**Orcc**[2], the second tool of the transformation flow, is an open source compiler for applications modeled with RVC-CAL and a DPN. When converted into DPNs, processes from KPN are replaced with actors that still communicate via infinite FIFOs. Contrary to KPN processes, actors have a set of firing rules that dictate when an actor is fired and how many data tokens are exchanged [1]. DPNs actors also have the ability to peek in FIFOs, i.e. they can read data-tokens in input FIFOs without consuming them.

The main purpose of Orcc in the flow is to analyze the behavior of the actors generated by Modaë Studio and classify their behavior as Synchronous Dataflow (SDF), Cyclo-Static Dataflow (CSDF) [3], or DPN. CSDF is a restriction of DPN where actor firings follow a cyclic fashion and SDF is a restriction of CSDF where the exchanged tokens are constant over firings. The advantage of CSDF and SDF is that they both have a strong predictability, thus opening the way to efficient compile-time optimizations. If the DPN is classified as CSDF, Orcc is asked to convert the graph into a SDF graph before generating the tool's outputs. Applications classified as DPN are currently not supported by this tool chain. Orcc outputs C files, each implementing the internal behavior of an actor, and a SDF graph interconnecting the actors.

**Preesm**[3], the last element of the tool chain, is an open source rapid prototyping tool that automatically maps and schedules hierarchical SDF graphs on heterogeneous MPSoCs. Beside the SDF graph provided by Orcc, another input of Preesm is a graphically edited System-Level Architecture Model (S-LAM) of the targeted architecture. Using what is called a scenario, the user can also specify a set of parameters and constraints for the mapping and scheduling tasks, restricting for instance the mapping of an actor on a subset of cores of the architecture.

Before mapping the actors on the heterogeneous MPSoC, Preesm performs a set of conversions on the application model so as to reveal the parallelism embedded in the MoC. Once a static mapping and schedule are obtained, Preesm generates a specific C file for each core of the architecture, handling inter-core communication and synchronization, and containing ordered calls to the C functions of the actors generated by Orcc.

## 4. DEMONSTRATOR

Developed by Texas Instruments, Open Multimedia Applications Platform (OMAP) is a family of MPSoC designed for embedded handheld multimedia applications. OMAP devices include a general-purpose ARM processor core with one or more specialized co-processors. The 4th generation OMAP processors are based on a dual-core ARM Cortex-A9 as main processing unit. The OMAP4460 additionally contains two Cortex-M3 microcontrollers

[2]http://orcc.sourceforge.net

[3]http://preesm.sourceforge.net

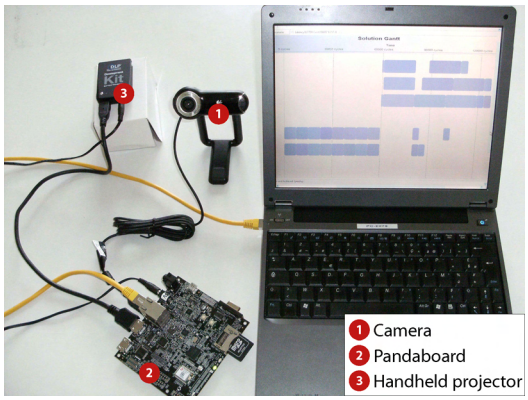



**Fig. 2**. Demonstrator material including a Pandaboard, a computer, a camera, and a projector

increasing power efficiency as well as a c64xT Digital Signal Processor (DSP) that can be used to speed-up some DSP-friendly algorithms. Hardware coprocessors are not covered by this demonstration. In this processor, 5 different cores can thus run simultaneously, each running a different program. Inter-core communications are possible through a shared Level-3 memory.

The demonstrator (Figure 2) consists of:

- a camera that captures a video stream,
- a Pandaboard ES, including an OMAP4460 MPSoC, that receives the video stream and computes an image processing algorithm distributed on up to 5 cores,
- a handheld projector that displays the resulting video, and
- a PC that generates and reloads code on each of the 5 cores, connected to the Pandaboard ES via Ethernet.

During this demonstration, high-level programming of an OMAP4 MPSoC is shown. A Sobel filter application is described, simulated and exported as RVC-CAL code using the Modaë Studio tool. Then, the application MoC is converted into the SDF MoC by the Orcc compiler. The algorithm is automatically mapped and scheduled by the Preesm tool, based on knowledge of the application SDF MoC and of the architecture model. Finally, an executable is built for each MPSoC core, then loaded and run on the OMAP4 processor. The dual-core ARM Cortex-A9 main processing unit of the OMAP4 processor runs a Linux operating system and drives the 3 other cores. The obtained process is highly customizable and, yet, most of the steps are fully automated and performed within seconds. The mapping constraints of actors, as well as the actors themselves, are modified during demonstration, resulting in a different code, mapped and scheduled efficiently on all cores of the OMAP4 processor. Intermediate generated codes and architecture model are demonstrated.

## 5. REFERENCES

[1] E. A. Lee and T. M. Parks, "Dataflow process networks," *Proceedings of the IEEE*, vol. 83, no. 5, pp. 773–801, 1995.

[2] Matthieu Wipliez, *Compilation infrastructure for dataflow programs*, Ph.D. thesis, INSA de Rennes, 2010.

[3] G. Bilsen, M. Engels, R. Lauwereins, and J. Peperstraete, "Cycle-static dataflow," *Signal Processing, IEEE Transactions on*, vol. 44, no. 2, pp. 397408, 1996.