HAL open science

Orcc: multimedia development made easy

Hervé Yviquel, Antoine Lorence, Khaled Jerbi, Gildas Cocherel, Alexandre Sanchez, Mickaël Raulet

To cite this version:

Hervé Yviquel, Antoine Lorence, Khaled Jerbi, Gildas Cocherel, Alexandre Sanchez, et al.. Orcc: multimedia development made easy. The 21st ACM International Conference on Multimedia, Oct 2013, Barcelone, France. pp.863-866. hal-00909401

HAL Id: hal-00909401

https://hal.science/hal-00909401v1

Submitted on 26 Nov 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.

Orcc: Multimedia Development Made Easy

Hervé Yviquel
University of Rennes 1
IRISA, Inria
Rennes, France
herve.yviquel@irisa.fr

Antoine Lorence,
INSA of Rennes, IETR
Rennes, France
antoine.lorence@insa-rennes.fr

Khaled Jerbi,
INSA of Rennes, IETR
Rennes, France
khaled.jerbi@insa-rennes.fr

Gildas Cocherel,
INSA of Rennes, IETR
Rennes, France
gildas.cocherel@insa-rennes.fr

Alexandre Sanchez,
INSA of Rennes, IETR
Rennes, France
alexandre.sanchez@insa-rennes.fr

Mickaël Raulet
INSA of Rennes, IETR
Rennes, France
mickael.raulet@insa-rennes.fr

ABSTRACT

In this paper, we present Orcc[1], an open-source development environment that aims at enhancing multimedia development by offering all the advantages of dataflow programming: flexibility, portability and scalability. To do so, Orcc embeds two rich eclipse-based editors that provide an easy writing of dataflow applications, a simulator that allows quick validation of the written code, and a multi-target compiler that is able to translate any dataflow program, written in the RVC-CAL language, into an equivalent description in both hardware and software languages. Orcc has already been used to successfully write tens of multimedia applications, such as a video decoder supporting the new High Efficiency Video Coding standard, that clearly demonstrates the ability of the environment to develop complex applications. Moreover, results show scalable performances on multi-core platforms and achieve real-time decoding frame-rate on HD sequences.

Categories and Subject Descriptors

12. [**Systems and Middleware**]: Tools; 5. [**Mobile & Multi-device**]: Miscellaneous

Keywords

Multimedia development, Dataflow programming, Integrated Development Environment (IDE), MPEG Reconfigurable Media Coding (RMC), High Efficiency Video Coding (HEVC)

1. INTRODUCTION

The growing demand and popularity of multimedia technologies has made multimedia application mainstream and is now used for a wide range of application domains. These multimedia applications have been achievable thanks to the advances in computing and communication technologies as well as algorithmic techniques. However, the deployment of these technologies is slowed down by the complex development process of these multimedia applications that involve time-consuming and error-prone tasks, and by the variety of the multimedia devices available on the market that have to support these multimedia applications.

In this paper, we present Orcc, an Integrated Development Environment that aims to make easier multimedia development by providing a set of modern tools based on dataflow programming. In fact, dataflow programming reduces the complexity of multimedia development by allowing a flexible development process, by describing the application in a portable manner, and by allowing scalable performances of the applications over multi-core platforms.

In the following, Section 2 gives an overview of dataflow programming. In Section 3, we provide a presentation of the main tools of Orcc and its implemented features. Finally, we give some results in Section 4.

2. DATAFLOW PROGRAMMING

A dataflow program is defined as a graph composed of a set of computational units interconnected by communication channels. In the dataflow approach, the communication corresponds to a stream of data composed of a list of tokens.

During the last twenty years, dataflow programming has been studied for the development of signal processing applications due to its consistency with the natural representation of the processing of digital signals, and particularly:

- The opportunity to use visual programming to describe the interconnection between its components. Such a graphical approach is very natural and makes it more understandable by programmers that can focus on *how things connect*.

- Its ability to express concurrency without complex synchronization mechanism. The internal representation of the application is a network of processing blocks that only communicate through the communication channels, making the blocks independent and side-effect free.

[1]Orcc is available at http://orcc.sf.net

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MM'13, October 21–25, 2013, Barcelona, Spain.
Copyright 2013 ACM 978-1-4503-2404-5/13/10 ...$15.00.
http://dx.doi.org/10.1145/2502081.2502231.

However, most of the studies on dataflow programming focus on the statically schedulable dataflow Models of Computation because of the efficiency of synthesis techniques on such models. Unfortunately, they do not take into consideration the expressiveness and the practicality offered to the programmers by more general dataflow models, such as dynamic dataflow [10].

2.1 Enhancing multimedia development

In fact, dataflow programming can enhance the development of multimedia applications thanks to the following abilities:

- **Modularity:** The strong separation between the structural and behavioral modeling makes the application description very modular. For instance, a component can easily be replaced by another one while its interfaces (input and output ports) are strictly identical. Moreover, such descriptions are typically hierarchical, in that a component of the graph may represent another graph.
- **Parallelism:** A dataflow program states an abundance of parallelism thanks to the explicit exposition of the concurrency. In its structural view, the dataflow model presents three potential degrees of parallelism (task, data and pipeline) that can be applied to different granularities of description.
- **Portability:** The heterogeneity of multimedia devices makes portability an interesting properties of multimedia applications. High-level dataflow-based descriptions aim to be compiled in lower-level languages in order to bridge the gap between the existing programming models.

2.2 Innovative development framework

During the last twenty years, several new programming languages based on dataflow models were proposed by the scientific community to solve a large panel of problems, from parallel programming to signal processing, but only a few have been emerging enough to be involved in the development of *real world applications*. One of them, the CAL Actor Language (CAL) [5], is particularly interesting due to the practicality offered by the language specification and the expressiveness offered by its ability to describe data-dependent behavior specified by the dynamic dataflow Model of Computation [10].

Reconfigurable Media Coding (RMC) [3] is an innovative framework introduced by MPEG to overcome the lack of interoperability between the several video codecs deployed in the market. The framework is dedicated to the development of video coding tools in a modular and reusable fashion thanks to dataflow programming. The inclusion of a subset of CAL in the RMC framework enables the development of several video decoders along other applications using dataflow programming.

3. FULL DEVELOPMENT ENVIRONMENT

The development of multimedia applications, such as video codecs, is a time-consuming and error-prone task due to the complexity of the algorithm and the variety of the multimedia devices. The development of those applications is even getting harder since the emergence of parallel platforms, as a consequence of frequency wall, due to the difficulties to synchronize the different tasks, balance their loads, etc. Consequently, the need for efficient development methods and tools is becoming increasingly important to meet the time-to-market requirement:

1. **Assisted writing of the applications:** The development of applications is made easier thanks to an Integrated Development Environment.
2. **Fast validation of the code:** A fast functional verification is making possible by the integrated simulator.
3. **Develop once, run everywhere:** The embedded trans-compiler is able to generate both hardware and software code from a single description that can be executed on large panel of platform thanks to the availability of dedicated runtime libraries.

3.1 Rich eclipse-based editors

Orcc is delivered with an entire Eclipse-based Integrated Development Environment (IDE) presented in Figure 1, making easier the development of RVC-based applications thanks to graphical interfaces. This environment is composed of two dedicated editors to handle both hierarchical levels, actor and network, of dataflow programming:

First, an intuitive **graph editor** that enables fast and easy building of the actor network using visual programming. A few mouse clicks are sufficient to create a node and assign it to an existing component from the project, or to create an edge that represents the communication channels between two nodes. The editor also supports hierarchical representation, assigning a whole subnetwork to a graph node, and hierarchical navigation, opening a subnetwork with a simple click on a graph node.

Finally, a full-blown **RVC-CAL editor** with advanced features, such as syntax coloring, content assist and code validation for helping the development of the actors. Based on the Xtext framework [4], Orcc implements all features expected for a modern and efficient Domain-Specific Language editor. The development environment is able to parse the actors and build their intermediate representation on-the-fly, in a incremental fashion, allowing fast simulation and compilation.

3.2 Integrated simulator

Additionally to the editors, Orcc brings a complete Java-based simulator which allows developers to quickly test their applications without taking in consideration low-level details relative to the target platform. The simulator can be launched directly from eclipse to execute any RVC-CAL application. Indeed, the simulator simply interprets our intermediate representation of networks and actors, but it is however able to perform all basic interactions required to perform a functional validation, such as displaying text, images or videos to the screen.

3.3 Multi-target development tools

Orcc is delivered with a compiler and a runtime library that allows to write a single description of the application to target a variety of executing platforms such as General-Purpose Processors, FPGAs, embedded processors and so on.

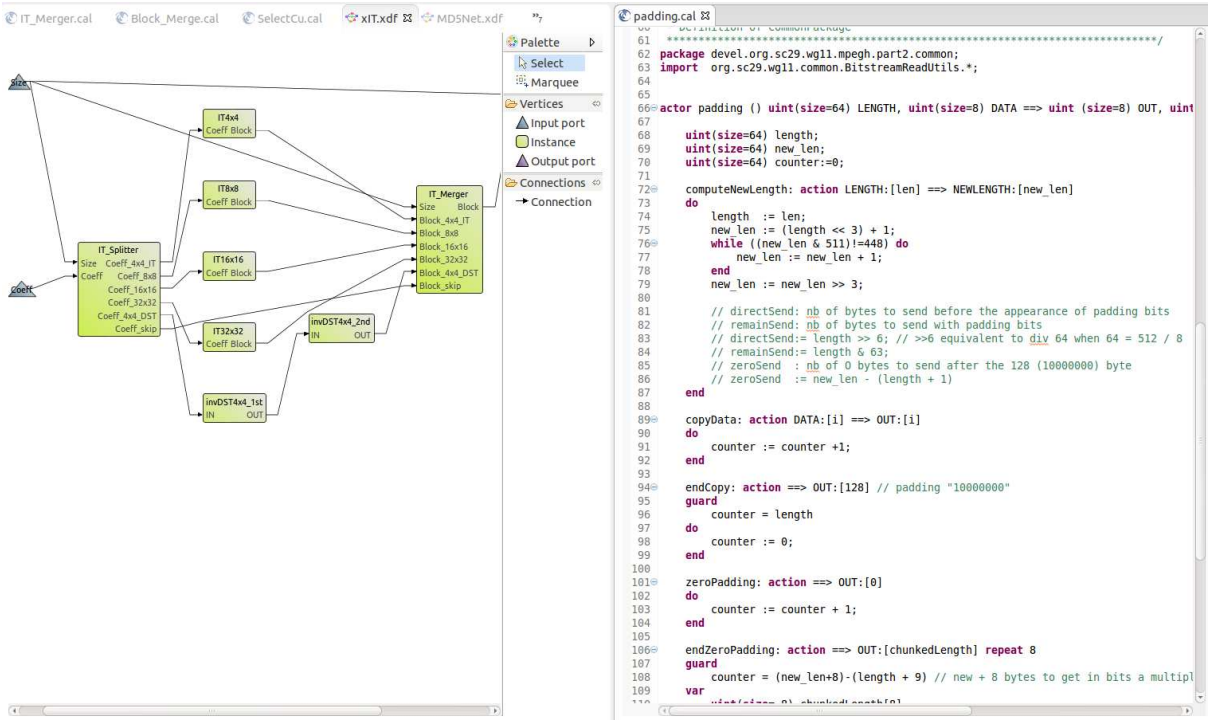

Figure 1: The Graph and RVC-CAL editors integrated into Eclipse IDE

Efficient dataflow trans-compiler.

The Orcc's compiler is able to translate a unique high-level dataflow program, written in RVC-CAL, into an equivalent description in both hardware and software languages for various platforms. A specific compiler back-end has been written to tackle each configuration case:

- Two **software** back-ends that generates C/C++ programs with multi-core abilities usable on most of the programmable processors [12]. A rapid prototyping can additionally be perform on static applications [9].
- Two **hardware** back-ends using well-known High-Level Synthesis tools to generate synthesizable HDL code for FPGA and ASIC implementations [1, 2].
- A back-end that targets **embedded multi-core platforms** [13]. The back-end is implemented as an entire co-design flow built upon the TCE toolset [7] that generates the software code as well as the hardware design that executes it.
- Two back-ends that produce **libraries of components** (in LLVM bitcode and Java), to implement the *Abstract Decoder Model* of the RMC framework [3]. The libraries of components are then used to perform adaptive execution based on virtual machine mechanism [8].

Each back-end is fully configurable using a set of parameters that affects the generated code. For example, the C back-end can use different scheduling strategies to execute a network of actors [14].

Additionally, some advanced analysis dedicated to dataflow applications can be performed during the compilation. A dynamic analysis, called actor classification [11], can detect predictable behavior within a network that may allow compile-time optimization such as static actors scheduling. Another analysis, based on model-checking techniques [6], can prune all unreachable execution paths to remove the unnecessary tests and accelerate the execution.

Cross-platform runtime library.

To allow developers creating full featured applications executable on any platform, Orcc defines a set of native functions callable from any CAL actor. Each one of these functions is implemented in a library exported at compile-time with the application, depending on the back-end used to produce source code. With these methods, developers are able to perform some basic I/O actions, such as opening and reading a file on disk, or initializing a window and sending raw data to display frames of a video sequence. All native functions available in Orcc library are implemented in the simulator as well.

4. RESULTS

This section presents the practicality of our development tool supported by the number of multimedia applications developed within it.

Library of applications.

Orcc is delivered with tens of open-source multimedia applications, developed by its community, that clearly demonstrate the ability of the environment to develop all kind of applications, from simple algorithm to complex system.

Additionally, these applications involve a large panel of domains in the multimedia including: **Video compression** with, for example, the descriptions of an MPEG-4 part 2

SP decoder, An MPEG-4 AVC decoder with 2 profiles, and an HEVC decoder; **Image compression** with the description of JPEG and JPEG2000 decoders; **File compression** (GZIP decoder); **Cryptography** (AES, DES, SHA); **Digital radio** (Zigbee protocol); **Digital filtering** (FIR and IIR filters); And so on.

Scalable performances of the applications.

We have experimented the 3 video decoders, including the one implementing the new High Efficiency Video Coding standard. The processor used during these experiments is an Intel Xeon W3530 composed of 4 homogeneous cores clocked at 2.8GHz. The processor is executing the compiled source code generating by the C back-end of Orcc. The experiments have been made from the following 720P sequences containing I/P/B frames: Old town cross (25fps and 6Mbps) for MPEG-4 Part 2; A Place at the Table (25fps and 6Mbps) for MPEG-4 AVC; Four People (60fps and 1Mbps) for MPEG HEVC.

Number of cores	1	2	4
MPEG-4 Part 2	30	54	93
MPEG AVC / H.264	4	7	12
MPEG HEVC / H.265	13	24	25

Table 1: Scalable video decoding frame rates (FPS) on a desktop multi-core processor

The results show that the RVC-CAL descriptions of the 3 video decoders can naturally take advantage of the multi-core processor thanks to their explicit parallelism by offering scalable frame rates. Since the HEVC decoder is still under development, it does not yet fully take advantage of the parallel processing capabilities, even if its usage with 2 cores already allow to display a video sequence at real-time performance.

5. CONCLUSION

Orcc is a complete development environment, now used by industrials and research laboratories worldwide, that aims to make easier the development of multimedia applications. Based on dataflow programming that enables flexibility, scalability and portability, and especially on the RVC-CAL language that offers expressiveness and practicality, Orcc provides a set of tools that accelerates the development of multimedia applications in order to meet the requirement of the time-to-market.

6. ACKNOWLEDGMENTS

We would give special thanks to the Orcc community as a whole for actively participating in the development of the tools which offers solid basements to this work.

This work is done as part of COMPA, a French ANR project (ANR-11-INSE-0012), and 4EVER, a French national project with support from Europe (FEDER), French Ministry of Industry, and local authorities.

7. REFERENCES

[1] M. Abid, K. Jerbi, M. Raulet, O. Déforges, and M. Abid. System Level Synthesis Of Dataflow Programs: HEVC Decoder Case Study. In *ESLSyn*, 2013.

[2] E. Bezati, S. C. Brunet, M. Mattavelli, and J. W. Janneck. Synthesis and Optimization of High-Level Stream Programs. In *ESLSyn*, 2013.

[3] S. S. Bhattacharyya, J. Eker, J. W. Janneck, C. Lucarz, M. Mattavelli, and M. Raulet. Overview of the MPEG Reconfigurable Video Coding Framework. *Journal of Signal Processing Systems*, 63(2):251–263, July 2009.

[4] S. Efftinge, M. Eysholdt, J. Köhnlein, S. Zarnekow, W. Hasselbring, R. von Massow, and M. Hanus. Xbase: implementing domain-specific languages for Java. In *Proceedings of the 11th International Conference on Generative Programming and Component Engineering*, pages 112–121, 2012.

[5] J. Eker and J. W. Janneck. CAL language report: Specification of the CAL actor language. Technical report, University of California, Berkeley, Berkeley, 2003.

[6] J. Ersfolk, G. Roquier, F. Jokhio, J. Lilius, and M. Mattavelli. Scheduling of dynamic dataflow programs with model checking. In *Systems (SiPS), 2011*, pages 37–42, 2011.

[7] O. Esko, P. Jääskeläinen, P. Huerta, C. S. de La Lama, J. Takala, and J. I. Martinez. Customized Exposed Datapath Soft-Core Design Flow with Compiler Support. In *Proceedings of the 2010 International Conference on Field Programmable Logic and Applications*, pages 217–222, Washington, DC, USA, Aug. 2010. IEEE Computer Society.

[8] J. Gorin, H. Yviquel, F. Prêteux, and M. Raulet. Just-in-time adaptive decoder engine. *Proceedings of the 19th ACM international conference on Multimedia - MM '11*, page 711, 2011.

[9] J. Heulot, K. Desnos, M. Pelcat, H. Yviquel, J.-F. Nezan, M. Raulet, P.-L. Lagalaye, and J.-C. Le Lann. An experimental toolchain based on high-level dataflow models of computation for heterogeneous MPSoC. In *Design and Architectures for Signal and Image Processing (DASIP)*, pages 1–2, 2012.

[10] E. A. Lee and T. Parks. Dataflow process networks. *Proceedings of the IEEE*, 83(5):773–801, 1995.

[11] M. Wipliez and M. Raulet. Classification and transformation of dynamic dataflow programs. In *Design and Architectures for Signal and Image Processing (DASIP), 2010 Conference on*, 2010.

[12] M. Wipliez, G. Roquier, and J.-F. Nezan. Software Code Generation for the RVC-CAL Language. *Journal of Signal Processing Systems*, 63(2):203–213, June 2009.

[13] H. Yviquel, J. Boutellier, M. Raulet, and E. Casseau. Automated design of networks of Transport-Triggered Architecture processors using Dynamic Dataflow Programs. *Signal Processing Image Communication, Special issue on Reconfigurable Video Coding*, 2013.

[14] H. Yviquel, E. Casseau, M. Wipliez, and M. Raulet. Efficient multicore scheduling of dataflow process networks. In *IEEE Workshop on Signal Processing Systems (SiPS)*, pages 198–203, Washington, DC, USA, 2011. IEEE Computer Society.