

An integrated environment for HW/SW co-design based on a CAL specification and SW/HW code generators

Ghislain Roquier², Christophe Lucarz², Marco Mattavelli²,
 Matthieu Wipliez¹, Mickaël Raulet¹,
 Jörn W. Janneck³, Ian D. Miller³, Dave B. Parlour³

¹ IETR, UMR CNRS 6164, Image and Remote Sensing laboratory, F-35043, Rennes, France
 Contacts: {name.lastname}@insa-rennes.fr

² Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland
 Contact: {name.lastname}@epfl.ch

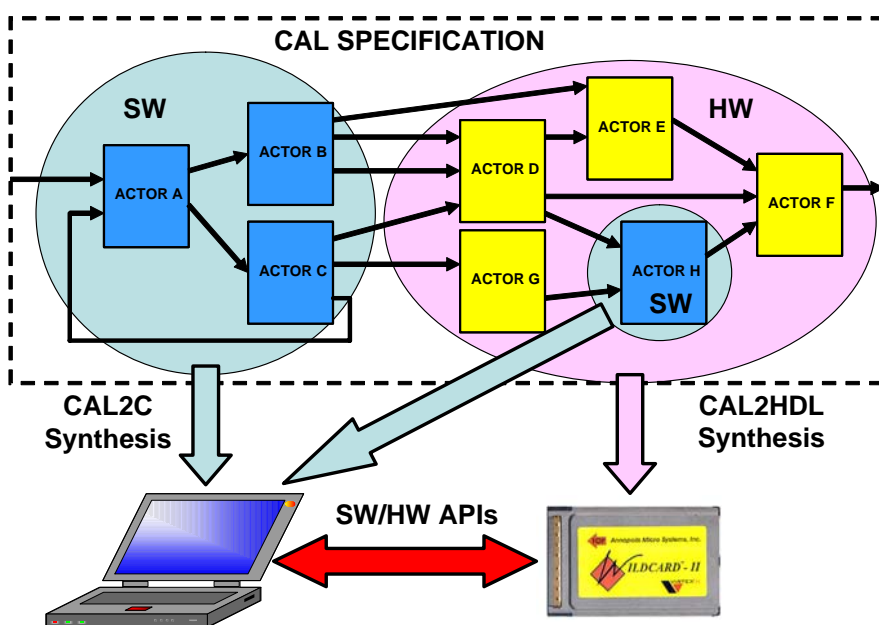
³ Xilinx, Inc. San Jose, CA, U.S.A
 jorn.janneck@xilinx.com

Abstract

The possibility of specifying both SW and HW components using the same language is a very attractive design approach. However, despite the efforts spent for implementing such approach using common programming languages such as C and C++, it has not yet shown to be viable and efficient for complex design. The main reason is the difficulty of expressing architectural properties at the level of a common description, difficulty that finally results into the failure of synthesizing efficient HW and efficient parallel SW. This is not the case for dataflow programming that is well-suited to the description of complex real-world computational systems that may contain appreciable amounts of concurrency. A CAL [1, 2] dataflow program is a collection of actors connected by lossless first-in/first-out (FIFO) communication channels, through which they exchange packets of data called tokens. Each actor executes in a sequence of steps, during which it can (1) consume input tokens, (2) produce output tokens, (3) modify its own local state. Actors strictly encapsulate their state, i.e. they cannot directly use or modify the state of other actors. Consequently, executing actors concurrently does not cause race conditions on any state variables, which makes dataflow an excellent (and scalable) parallel programming model. This in turn is key to the generation of efficient HW and SW implementations. This demonstration presents an integrated environment that embeds the synthesis tool that translates a CAL-based dataflow specification into a heterogeneous implementation, composed by HDL and C codes. The demonstration focuses on the capability of the co-design environment to automatically build an executable heterogeneous system implementation running on a platform composed by a processor and a FPGA platform just by annotating the CAL specification. The possibility of direct synthesis from a high level specification is a crucial issue for enabling efficient re-design cycles that include rapid prototyping and validation of performances of the final implementation. The design approach enabled by such integrated environment is particularly suited for development of complex processing systems such as video codecs. As a case study, the demonstration provides the analysis and validation of different SW and HW partitioning of a MPEG-4 Simple Profile decoder.

Introduction

When a high-level CAL system specification is available, such as in the case of a MPEG RVC description [1], the possibility of generating heterogeneous system implementations from such unique specification is a very attractive feature for any system designer. In fact, the decision of partitioning the processing on SW and HW (such as C and VHDL) at a very early stage of the design is simple, but may result not a good option on the final platform. When the development of final SW and HW implementations requires large designer efforts, redesign cycles that restart with a new high level SW and HW partitioning are prohibitive. However when synthesis tools for both HW and SW such as Cal2C [3] and CAL2HDL [4] can directly generate an integrated implementation that can be profiled and validated, successful redesign cycles can be efficiently iterated.



SW/HW Co-generation

From an annotated CAL model, indicating which actor should be implemented in SW or HW, the integrated co-design generator composed by a graphical editor a behavioral simulator and by the CAL2C [3] and CAL2HDL [4] synthesis tools, generates C and HDL implementations corresponding to the selected partitions. The result is an implementation composed of C and C++ code compiled on a PC platform and HDL code synthesized to RTL and mapped on a FPGA platform (Wildcard II). For instance, if actor B is mapped to SW and actor D to HW, two C functions, *actorB()* and *actorD()* are created by the co-design generator. Then, the CAL2C synthesis tool generates C code for actor B and the synthesized code is automatically wrapped into the function *actorB()*, whereas the *actorD()* function will simply consist of a call to a HW function generated automatically and inserted into the *actorD()* function. The HW function call will make use of the API of the Wildcard (*WC_PeRegWrite()* and *WC_PeRegRead()*) thus establishing the connection modeled in CAL by the FIFO channels. In this way the actors mapped as HDL code on the FPGA can execute their tasks

when they receive the tokens via the virtual channels established by the API. Communications channels between two actors mapped on SW are implemented as SystemC FIFOs. The scheduling of the execution of the C code corresponding to the different firing of actions of the CAL actors, is dynamically determined by a SystemC scheduler. In summary the top level model composing the executable version of the complete CAL specification is composed by:

- C functions, one for each actor. Each of these C modules either contains C code in case the actor has been mapped to SW or call to the hardware API in case the actor has been mapped to HW.
- A SystemC scheduler, handling the scheduling of the execution of the different actors mapped to SW.
- RTL implementations on a FPGA for all actors mapped to HW.

The Hardware Platform

The platform with an FPGA and embedded memory providing the APIs from the SW components is a PCMCIA WILDCARD™-II card. It delivers the processing power of the Xilinx® Virtex™-II series FPGA (XC2V3000-4). The Wildcard is provided with a built-in API in order to establish the communication with the FPGA. The API is written in C and has been used to implement the channels of the CAL model that connects actors mapped on the FPGA with the one mapped on the SW on the PC. Inside the FPGA the following components establish the communication:

- INPUT blocks which implement the connections between the card bus and the FPGA (LAD bus) and the generated RTL code.
- OUTPUT blocks which implement the connections between the generated RTL code and the LAD bus.
- The generated RTL code mapped on the FPGA CLBs implementing the CAL actors mapped on HW.

An INPUT block for each input port of the corresponding CAL model actor is instantiated on the FPGA as well as an OUTPUT block for each output port of the corresponding CAL model. The function of the INPUT block is to feed the RTL with the data coming from the local BUS which is driven by the C API with the function *WC_PeRegWrite()* in order to send data tokens to the card. The function of the OUTPUT block is to send the data resulting from the execution of the actions to the local bus (LAD Bus) so that the API can transmit the data back into the C program using the function *WC_PeRegRead()*.

The Demonstration

The demonstration will show examples of partitions into SW and HW components with associated synthesis to SW and HW using the OpenDF integrated environment for the design case of a MPEG-4 SP decoder.

REFERENCES

- [1] C. Lucarz, M. Mattavelli, J. Thomas-Kerr, and J. Janneck. Reconfigurable Media Coding: a new specification model for multimedia coders. In Proceedings of SIPS'07, October 2007.
- [2] J. Eker and J. Janneck. CAL Language Report. Tech. Rep. ERL Tech. Memo UCB/ERL M03/48, University of Berkeley, Dec. 2003.
- [3] G. Roquier, M. Wipliez, M. Raulet, J. W. Janneck, I. D. Miller and D. B. Parlour, "Automatic Software Synthesis of Dataflow Program: An Mpeg-4 Simple Profile Decoder Case Study Signal Processing Systems", In proceedings of SIPS'08. IEEE Workshop on, 2008.
- [4] J. W. Janneck, I. D. Miller, D. B. Parlour, G. Roquier, M. Wipliez and M. Raulet "Synthesizing Hardware from Dataflow Programs: an MPEG-4 Simple Profile Decoder Case Study", In proceedings of SIPS'08. IEEE Workshop on, 2008.
- [5] "Open dataflow sourceforge project," <http://opendf.sourceforge.net/>.