Code Generation for Simulation of Parallel Process Networks

Matthieu Moy – Maître de conférences Université Lyon1/LIP
Christophe Alias – Chargé de recherche INRIA/LIP

Keywords Compilation, Simulation, High-level Synthesis, Parallelism, SystemC, Dataflow model, Polyhedral Model.

Location Laboratoire de l'Informatique du Parallélisme (LIP)
École Normale Supérieure de Lyon, France.

Context

In the beginning of the 2000's, the clock frequency of computation units reached its limits. Energy-efficiency is becoming a major bottleneck for supercomputers [1]. Increasing the clock frequencies implies a loss of energy efficiency that is no longer acceptable. Most gains in performance now come from the augmentation of the number of computation units (processor cores, specialized processors). New programming paradigms have to be found to continue increasing performance in a given energy budget.

One solution is to implement the main algorithms of a computation in hardware, and map it to reconfigurable circuits (FPGA, Field Programmable Gate Array) [2]. To execute an application on FGPA, new technological locks must be overcome. Among them is the automatic and efficient translation of an algorithm into a circuit design. This operation is called HLS (High-level synthesis).

Translating a program into a circuit is done in several steps. First, the front-end generates an intermediate representation adapted to circuit synthesis. In the tools developed by CASH, this formalism is called "Data-aware Process Network" (DPN) and represents a network of processes that captures the parallelism of an application and the communications between parallel processes. Then, the back-end translates each component of the process network into hardware while ensuring a good reuse of hardware resources. In the end, the circuit can be seen as a very large network of pipelined processes, reading inputs and producing outputs periodically.

The newly created CASH[1] team works on novel approaches to extract parallelism from an imperative program to the DPN intermediate representation. The final code generation to FPGA is out of the scope of the team: we work on source-to-source approaches that read sequential C code, extract parallelism and produce code that makes this parallelism explicit. The generated code will then be given as input to another HLS tool that will generate the actual circuit description.

To evaluate the quality and correctness of the generated process network, one option would be to run the generated process network through the back-end and execute the result on an FPGA. However, the back-end and synthesis are time-consuming operations and running on an FPGA provides only limited debugging tools. The other option is to *simulate* the process network before the back-end. Historically, we used a minimal simulator based on POSIX threads, using one thread per process. This solution is operational but slow due to the number of context-switches required. A new simulator was started in spring 2018. This new simulator uses the principles of discrete-event simulation, and relies on the SystemC [3] simulator for this. SystemC is the standard tool for high-level circuit simulation. It has an efficient scheduler using a cooperative scheduling policy for which context-switches are efficient. The new simulator is only an incomplete proof-of-concept.

Generating SystemC code is also a way to implement the HLS back-end: most HLS tools, including Vivado HLS [4] that we are using, accept SystemC code as input. Compared to plain C or C++, SystemC allows making the parallelism explicit. Although optimal code generation strategies are different for HLS and for simulation, at least part of the code generator can be common to both.

Objectives of the Internship

The overall objectives of the internship are:

1. Understand the principles of DPN and of the existing incomplete SystemC simulator

[1] `http://www.ens-lyon.fr/LIP/CASH/`

2. Implement the missing features in the simulator. In particular, process communication is only possible through FIFO buffers in the current simulator, while the DPN formalism allows a much more general communication and synchronization scheme based on the polyhedral model.

3. Explore the possibilities of Vivado HLS with respect to SystemC. Experiment with a few handwritten examples.

4. Modify the SystemC code generator to target the subset of SystemC accepted by Vivado HLS. Adapt it to allow generating pragma statements to activate the relevant Vivado HLS optimizations.

5. Find the best code generation strategies to get the best simulation and synthesis performance. The best choices may be different for simulation and for synthesis.

The internship will contain a substantial "implementation" part. Depending on the profile and motivation of the student (research or engineering), the internship can either focus on implementation (to produce a very robust code generator) or on theory (propose program transformations like merging processes and loops, to generate the best code possible).

Supervision

This thesis will be supervised by Christophe Alias (Inria Researcher, ENS-Lyon) and Matthieu Moy (Assistant professor, HDR, UCBL).

Christophe Alias (`http://perso.ens-lyon.fr/christophe.alias/`)'s research interests includes automatic parallelization (in the polyhedral model), static analysis and high-level synthesis for FPGA circuits. He wrote the DPN compiler which serve as a basis for this PhD thesis.

Matthieu Moy (`https://matthieu-moy.fr`)'s main research area is hardware simulation (using SystemC) and formal verification (model-checking, abstract interpretation, SMT solving). More recently, he started working on worst-case execution time for software and worst-case traversal time for networks-on-chip, and compilation for critical systems. He joined the LIP laboratory in 2017 and started working on HLS and polyhedral methods.

Expected Skills

Solid notions of compilation and parallelism. Knowledge of computer architecture. Solid knowledge of C++.

Candidatures

Send applications by email to: Christophe.Alias@ens-lyon.fr and Matthieu.Moy@univ-lyon1.fr.

References

[1] Nor Zaidi Haron and Said Hamdioui. Why is cmos scaling coming to an end? In *Design and Test Workshop, 2008. IDT 2008. 3rd International*, pages 98–103. IEEE, 2008.

[2] Altera Corporation. Altera FPGAs achieve compelling performance-per-watt in cloud data center acceleration using CNN algorithms. `http://www.prnewswire.com/news-releases/altera-fpgas-achieve-compelling-performance-per-watt-in-cloud-data-center-acceleration-using-cnn-al html`, 2015.

[3] Accelera Systems Initiative. Systemc. `http://www.accellera.org/downloads/standards/systemc`.

[4] Xilinx. Vivado high-level synthesis. `https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html`.