A Virtual Machine Approach for High-level FPGA Programming

Loïc Sylvestre, Jocelyn Sérot, Emmanuel Chailloux

A Virtual Machine Approach for High-level FPGA Programming

Loïc Sylvestre
Sorbonne Université, CNRS, LIP6
F-75005 Paris, France
Email: loic.sylvestre@lip6.fr

Jocelyn Sérot
Université Clermont Auvergne, CNRS, SIGMA
Institut Pascal F-63000 Clermont-Ferrand, France
Email: jocelyn.serot@uca.fr

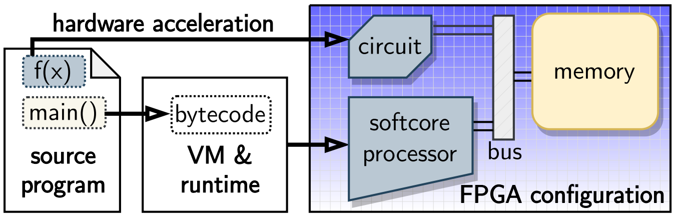
Emmanuel Chailloux
Sorbonne Université, CNRS, LIP6
F-75005 Paris, France
Email: emmanuel.chailloux@lip6.fr

***Abstract*—We introduce a virtual machine approach to program FPGAs using a high-level programming language (with automatic memory management) while hardware-accelerating a subset of it. This offers an interesting trade-off between high-level synthesis tools and pure software approaches. We describe a preliminary implementation of this hybrid approach using the OCaml language on Intel FPGAs. The associated toolset fully automatizes the compilation process from the OCaml source program to the SoPC hardware and software configuration. First results are encouraging, both for programmability and efficiency.**

I. Overview

FPGA design requires more and more high-level programming features for meeting the needs of productivity and customization from hardware designers and programmers.

To fulfill these needs, we introduce the following virtual machine approach: given an existing compiler for a high-level programming language producing bytecode interpretable by a virtual machine (VM), a port of this VM to a softcore processor suffices to fully support such language on FPGA. Then, hardware acceleration of a subset of this language allows to exploit parallelism and customization possibilities of the FPGA. The resulting circuit can interoperate with the softcore processor, especially to access data structures (such as arrays, matrices and trees) dynamically allocated by the VM runtime.



We assess this approach with OCaml, a statically-typed multi-paradigm (functional, imperative, modular and object-oriented) programming language. Our contributions are:

- O2B (*OCaml on board*) [1], an implementation of the OCaml VM based on OMicroB [2], targeting the Nios II softcore and allowing to use custom hardware from them;
- Macle (*ML accelerator*) [3], a compiler for a subset of OCaml producing VHDL descriptions of custom hardware, scripts and glue code to automatically extend O2B.

II. Preliminary Evaluation

First results on FPGA programming in OCaml suggest that using O2B and Macle achieves good performances. To illustrate this, we compare two formulations of the *gcd* algorithm. One is written in C compiled by gcc -O2 targeting the Nios II and the other in OCaml compiled by Macle. We observe a speedup of almost 30 for the OCaml version against the C one.

Macle also exploits parallelism. For instance, the expression $gcd(a, b)+gcd(b, a)$ is compiled as a synchronization barrier in which the circuit implementing the *gcd* function is duplicated, hence providing an extra 2× speedup. Macle specializes as well a *map* OCaml function to produce parallel code: let *p* be a local static array (or ”packet”) of size 16, the expression $map(gcd, p)$ allows an additional speedup of almost 16. This parallel skeleton is generalized to OCaml arrays (dynamically allocated by the VM runtime in the on-chip memory). The latter processes each array element in parallel by packet of a fixed size (given by the programmer) while optimizing transfers to compensate the overhead of accessing the memory bus.

III. Conclusion

Benefits of our VM approach includes:

- ease of implementation on a softcore processor communicating with custom circuits through shared memory;
- productivity gain, especially for dynamic allocation and processing of complex data structures;
- hardware acceleration of a subset of the source language facilitating both prototyping and simulation.

Future work will focus on implementing the VM heap in external memory to dynamically allocate large data structures while optimizing processing over them from the hardware-accelerated code. This will allow to program realistic applications mixing numerical and symbolic computations.

IV. Acknowledgment

This work was partially supported by the Center for Research and Innovation on Free Software (IRILL).

References

[1] “O2B.” [Online]. Available: https://github.com/jserot/O2B

[2] S. Varoumas, B. Vaugon, and E. Chailloux, “A Generic Virtual Machine Approach for Programming Microcontrollers: the OMicroB Project,” in *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, Toulouse, France, Jan. 2018.

[3] “Macle.” [Online]. Available: https://github.com/lsylvestre/macle