Bridging the Gap Between Theory and Hardware

Mario Marino, Gabriel Robins, Kevin Skadron, and Liang Wang
Department of Computer Science, University of Virginia
{mdm9uw, robins, skadron, lw2aw}@cs.virginia.edu

"In theory, there is no difference between theory and practice. But in practice, there is." - Yogi Berra

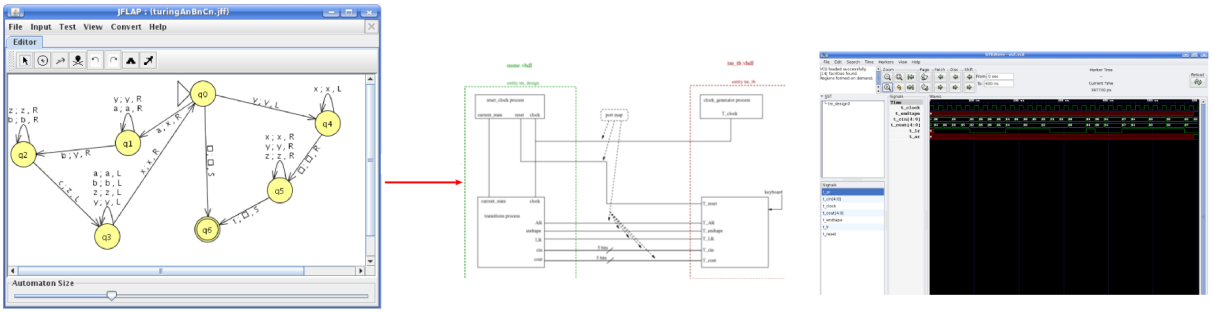
This work explores how theoretical models can be leveraged to improve the effectiveness of special-purpose (e.g., FPGA-based) processors. Turing machines (TM) and other theoretical models are widely employed to address problems such as parsing and lexical analyses, searching and pattern matching, etc [5].

The traditional way to write a program is to start codifying it in a generic language and then, using a compiler to translate and then optimize it to an appropriate binary. However, going from the development of the algorithm towards an efficient program implementation on the hardware, we create a big room between theory and implementation. To help narrow this gap between theory and practice, we developed a tool that shows the potential of automatically translating theoretical computation models into an implementation-ready hardware description language (VHDL) code and an appropriate test bench [3].

Future improvements of the tool show that the approach of the behavioral description aspect of VHDL reflecting the specification of the theoretical models has the prospective contributions:

- Hardware implementation of any Turing-based concept, which covers any type of application. Example of such novelties would be hardware implementations of a compiler or an OS, a different approach of the traditional combined software/general purpose computer. A FPGA-based implementation of a compiler is potentially faster than a typical implementation in software running in a general purpose computer and also improves speed on debugging.
- Implementation of accelerators based on the algorithmic description rather than requiring detailed implementation in C or VHDL.
- Due to the behavioral description side of VHDL, a hardware implementation can closely reflect the theory, with smaller overheads of language translation and code generation, reducing the gap between these two areas.
- Since it is closer to the TM model, it facilitates the development of new theoretical computational models, such as the Bounded-Error Quantum Polynomial-Time model employed in quantum computing [2].
- Enhancing computer science education and pedagogy by building a new bridge between these traditionally disparate core areas of theory and architecture, thus aiding students tying together concepts across these two fields.

Using as inputs to the translator tool, we described numerous TMs with JFLAP visualization tool [4]. As a result of the translation, we have parallel VHDL TM implementations, that were validated in the VHDL ghdl simulator [1]. In all cases, they performed correctly and efficiently with respect to the target TM functionality. In the particular example of the parallel TM machine we have two processes, the first is responsible to generate clock and reset, while the second is responsible for inducing the TM transitions that need to occur. The generated VHDL code detects and disallows illegal transitions, and appropriately determines when an input string is eventually accepted or rejected by the target TM.



Starting from a theoretical compiler model, future work may include a complete theoretical model of typical phases of compiler analyses, a translation to a VHDL implementation and a comparison in terms of speedup, power usage, support to find errors, and overheads when compared to a traditional compiler running in a general purpose computer. In addition, we will investigate how grammars can be benefited from the approach and also intend to investigate other interesting applications such as the most used parts of an OS kernel and do a similar analysis. In addition, by the inherent proximity with theory, we plan to investigate how the approach can facilitate the use of Quantum Computing [2] applications.

References

[1] GHDL home page. Accessed date: 10/15/2009 `http://ghdl.free.fr`.

[2] S. Aaronson. BQP and the Polynomial Hierarchy. In *to appear in STOC 2010, 42nd ACM Symposium on Theory of Computing*, Cambridge, MA, USA, 2010. ACM.

[3] P. Ashenden. *The Designer's Guide to VHDL*. Elsevier Science & Technology, Third edition, 2006.

[4] S. H. Rodger, E. Wiebe, K. M. Lee, C. Morgan, K. Omar, and J. Su. Increasing Engagement in Automata Theory with JFLAP. In *SIGCSE Bull.*, volume 41, pages 403–407, New York, NY, USA, 2009. ACM.

[5] M. Sipser. *Introduction to the Theory of Computation*. Thompson Course Technology, Second edition, 2006.