Formal verification of hardware synthesis

Thomas Braibant
University of Grenoble, currently visiting MIT

Adam Chlipala
MIT

Coq Workshop 2012

Verification of hardware designs has been thoroughly investigated, and yet, obtaining provably correct hardware of significant complexity is usually considered challenging and time-consuming. On the one hand, a common practice in hardware verification is to take a given design written in an hardware description language like Verilog or VHDL, and argue about this design in a formal way using a model checker or an SMT solver. On the second hand, a completely different approach is to design hardware via a shallow-embedding of circuits in a theorem prover [3, 5–8]. Yet, both kind of approach suffer from the fact that most hardware designs are expressed in low-level RTL languages like Verilog or VHDL, and that the level of abstraction they provide may be too low to do short and meaningful proof of high-level properties.

To raise this level of abstraction, industry moved to *hardware synthesis* using higher-level languages, e.g., System-C, Esterel [2] or Bluespec [1], in which a high-level source program is compiled to an RTL description. High-level synthesis has two benefits. First, it reduces the effort necessary to produce an hardware design. Second, writing or reasoning about a high-level program is simpler than reasoning about the (much more complicated) RTL description generated by a compiler. However, the downside of high-level synthesis is that there is no formal guarantee that the generated circuit description behaves exactly as prescribed by the semantics of the source program, making verification on the high-level program useless in the presence of compiler-introduced bugs.

We are currently working on a project that address this issue. That is, we investigate the formal verification of a (toy) compiler from a Bluespec-inspired language called BabyHardCert to RTL, quite literally applying the ideas behind the CompCert project [9] to hardware synthesis.

BabyHardCert can be seen as a stripped-down and simplified version of Bluespec: in both languages, hardware designs are described in terms of *guarded atomic actions* on storage elements. In our development, we define a (dependently-typed) deep-embedding of the BabyHardCert programming language in Coq using *parametric higher-order abstract syntax (PHOAS)* [4], and give it a semantics using an interpreter: the semantics of a program is a Coq function that takes as inputs the current state of the storage elements and a list of updates to be commited to this storage elements, and produces another list of updates to be commited. The one odditiy here is that this language and its semantics have a flavour of *transactional memory*, where updates to state elements are not visible before the end of the transaction (a time-step). Our target language can be sensibly interpreted as *clocked sequential machines*: we generate an RTL description syntactically described as combinational definitions and next-state assignements. (Note that we do not investigate yet the correctness of an *actual* implementation of this RTL description using the synthesisable subsets of Verilog or VHDL.)

In this talk, we shall present the current state of our development, and highlight some of its features. For instance, we look forward to discuss the oddities of the semantics of our hardware description languages, as well as the payoffs of our various implementation choices, including the use of PHOAS.

References

[1] L. Augustsson, J. Schwarz, and R. S. Nikhil. *Bluespec Language definition*. 2001.

[2] G. Berry. "The foundations of Esterel". In: *Proof, Language, and Interaction, Essays in Honour of Robin Milner*. The MIT Press, 2000. ISBN: 978-0-262-16188-6.

[3] S. Beyer, C. Jacobi, D. Kröning, D. Leinenbach, and W. J. Paul. "Putting it all together - Formal verification of the VAMP". In: *STTT* 8.4-5 (2006), pp. 411–430.

[4] A. Chlipala. "Parametric higher-order abstract syntax for mechanized semantics". In: *Proc. ICFP*. ACM, 2008, pp. 143–156.

[5] S. Coupet-Grimal and L. Jakubiec. "Certifying circuits in Type Theory". In: *Formal Asp. Comput.* 16.4 (2004), pp. 352–373.

[6] M. Gordon. *Why Higher-Order Logic is a Good Formalism for Specifying and Verifying Hardware*. Tech. rep. UCAM-CL-TR-77. Cambridge Univ., Computer Lab, 1985.

[7] F. K. Hanna, N. Daeche, and M. Longley. "Veritas$^+$: A Specification Language Based on Type Theory". In: *Hardware Specification, Verification and Synthesis*. LNCS. Springer, 1989, pp. 358–379.

[8] W. A. Hunt Jr. and B. Brock. "The Verification of a Bit-slice ALU". In: *Hardware Specification, Verification and Synthesis*. Vol. 408. LNCS. Springer, 1989, pp. 282–306.

[9] X. Leroy. "Formal verification of a realistic compiler". In: *CACM* 52.7 (2009), pp. 107–115.