Mechanized Certification of Secure Hardware Designs (Extended Abstract)

Sandip Ray Warren A. Hunt Jr.

Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712. USA.
{sandip,hunt}@cs.utexas.edu
http://www.cs.utexas.edu/users/{sandip,hunt}

We develop a framework for mechanized certification of secure hardware systems built out of commercial off-the-shelf (COTS) components purchased from untrusted vendors. Certification of secure systems requires the interaction between designers, consumers, and regulatory evaluators to guarantee that the fabricated system satisfies the requisite safety and security properties. Our framework facilitates such certification by (1) providing an unambiguous description of the requirements specification in a formal, computational logic, (2) a formalized hardware description language (HDL) to describe the implementation, and (3) mechanical tools and techniques for providing a certification of correctness and security. The mechanical tools and techniques include reasoning engines such as theorem proving and SAT-based decision procedures, simulation of the formal design artifact and specification requirement, and a hierarchical, formalized approach to structural and signal dependency analysis. Although our focus is on certification of secure systems, the approach is also useful in the context of functional verification of general-purpose hardware designs, by providing a uniform, mechanized framework for application of both simulation and formal tools.

We have used our framework to certify the correctness and security properties of the netlist implementation of a voting machine. To do so, we develop a general-purpose executable specification of the machine, and use a notion of correspondence based on stuttering trace containment [2] to relate the executions of the implementation with that of the specification. The notion allows us to formally relate a concrete implementation of the design with a specification modeled at a different level of abstraction. The notion also affords compositionality, which facilitates application of formal verification in a modular fashion. We show how the use of the generic, formal certification framework assists in the development of the specification and helps avoid common specification errors. On the other hand, the viability of the formal proof depends on certain assumptions on the environmental stimuli of the system; we show how to use simulation *within the formal framework* to validate such assumptions. Finally, we specify and prove security and regulatory checks on the implementation. A key security property is *separation*, that is, the votes for one candidate are not affected by any signals other than those used to record the casting of votes for that candidate. These properties ensure that (a) the machine cannot be rigged, and (b) there is no trapdoor to leak secure information out of the system. We show how to effectively verify such properties by structural analysis of the design.

Our work targets the dual requirement of verification and security. The pervasive application of computing systems in critical applications has brought significant attention to mechanized certification in both the government and the industry. For instance, the Common Criteria [1] requires a uniform *lingua franca* for the communication of designers, consumers, and evaluators. Nevertheless, such communications in current practice contain a number of informal components, namely requirement description as text graphs and charts, implementation in languages with incomplete or complicated semantics, and code review together with incomplete testing as the primary validation procedure. The *buyer-seller paradigm* [7] aims to alleviate this problem by proposing a formal infrastructure for communication between a buyer and an untrusted seller, allowing an independent evaluator to provide regulatory and security approval of the developed artifact. Our approach can be viewed as a step towards making such paradigm practicable.

Our framework is based on two key ingredients: the ACL2 system [9] as the formal foundation and specification language, and the **DE** language [8] as the HDL for describing netlists. ACL2 is a theorem prover for a first order logic of recursive functions, formalizing a large subset of Common Lisp; it has been used to verify some of the largest hardware designs that have been subjected to formal analysis [4, 5]. A key feature of the logic is the support for efficient execution, which makes it possible for ACL2 models to execute at speeds close to optimized C implementations [6]. This simulation support of the logical formulas, together with a clearly defined logical semantics, makes it an ideal foundation both as the specification language as well as the basis for implementation of mechanical automatic tools for design analysis.

The HDL we use for implementing hardware designs, namely **DE**, is a language for modeling synthesizable, finite-state Mealy machines. Unlike VHDL and Verilog, the semantics of **DE** is formal, and defined by a *deep embedding* of the language in the logic of the ACL2. Deep embedding is achieved by defining a formal interpreter of the language constructs in the underlying logic [3]. The **DE** interpreter is defined hierarchically, by first defining simple evaluators characterizing the functionality of the basic hardware primitives (such as AND and OR gates), and then defining a semantics of modules by composing these primitive evaluators. An important aspect of **DE** is the economy of its semantics: the entire language definition, other than the definition of the functional evaluators, is expressed by two mutually recursive definitions which constitute no more than one page of formal definition. On the other hand, **DE** is still expressive enough for specification of practical hardware systems; for instance it has been used to formalize components of the TRIPS processor [11]. Furthermore, there has been recent work on supporting formal proofs of a subclass of ACL2 formulas called SULFA [10] using an efficient, SAT-based decision procedure. The subclass encompasses hardware invariant properties; thus the invariants on **DE** modules that are involved in the verification of trace containment theorem mentioned above can be discharged with significant automation.

Finally, we briefly mention our approach to dependency analysis. The approach exploits the nature of the **DE** language. In particular, since the semantics of **DE** is based on a hierarchical composition of the primitive evaluators, we can formalize the dependency (or information flow) properties of a module by replacing the primitive *functional evaluators* with another evaluator (for instance, one evaluating signal dependency). The approach provides a general-purpose methodology for analysis of non-functional hardware properties, which is powerful and compositional, while avoiding the need to retrofit structural analysis tools on a model designed primarily to capture functional properties.

References

[1] Common Criteria for Information Technology Security Evaluation. See URL: `http://csrc.nist.-gov/cc/CC-v2.1.html`.

[2] M. Abadi and L. Lamport. The Existence of Refinement Mappings. *Theoretical Computer Science*, 82(2):253–284, May 1991.

[3] R. J. Boulton, A. Gordon, M. J. C. Gordon, J. Harrison, J. Herbert, and J. Van Tassel. Experience with Embedding Hardware Description Languages in HOL. In V. Stavridou, T. F. Melham, and R. T. Boute, editors, *Proceedings of the IFIP TC10/WG* 10.2 *International Conference on Theorem Provers in Circuit Design: Theory, Practice and Experience (TPCD 1992)*, volume A-10 of *IFIP Transactions*, pages 129–156. North-Holland, 1992.

[4] B. Brock and W. A. Hunt, Jr. Formal Analysis of the Motorola CAP DSP. In *Industrial-Strength Formal Methods*. Springer-Verlag, 1999.

[5] B. Brock, M. Kaufmann, and J S. Moore. ACL2 Theorems about Commercial Microprocessors. In M. Srivas and A. Camilleri, editors, *Proceedings of the* 1*st International Conference on Formal Methods in Computer-Aided Design (FMCAD 1996)*, volume 1166 of *LNCS*, pages 275–293. Springer-Verlag, 1996.

[6] D. Greve, M. Wilding, and D. Hardin. High-Speed, Analyzable Simulators. In M. Kaufmann, P. Manolios, and J S. Moore, editors, *Computer-Aided Reasoning: ACL2 Case Studies*, pages 89–106, Boston, MA, June 2000. Kluwer Academic Publishers.

[7] W. A. Hunt, Jr. Buyer-Seller Approach to Validation of Assurance. In *Conference on High Confidence Software and Systems*, 2006.

[8] W. A. Hunt, Jr. and E. Reeber. Formalization of the DE2 Language. In W. Paul, editor, *Proceedings of the* 13*th Working Conference on Correct Hardware Design and Verification Methods (CHARME 2005)*, LNCS, Saarbrücken, Germany, 2005. Springer-Verlag.

[9] M. Kaufmann, P. Manolios, and J S. Moore. *Computer-Aided Reasoning: An Approach.* Kluwer Academic Publishers, 2000.

[10] E. Reeber and W. A. Hunt, Jr. A SAT-Based Decision Procedure for the Subclass of Unrollable List Formulas in ACL2 (SULFA). In U. Furbach and N. Shankar, editors, *Proceedings of the* 3*rd International Joint Conference on Computer-Aided Reasoning (IJCAR 2006)*, volume 4130 of *LNAI*, pages 453–467, 2006.

[11] E. Reeber and W. A. Hunt, Jr. Applications of the DE2 Language. In M. Sheeran and T. Melham, editors, 6*th International workshop on Designing Correct Circuits (DCC 2006)*, 2006.