On Formal Equivalence Verification of Hardware

Zurab Khasidashvili

Formal Technology and Logic Group
Intel Corporation, Haifa, Israel

When modeling the logic functionality, hardware can be viewed as a Finite State Machine (FSM) [7]. The power-up state of hardware cannot be determined uniquely, therefore the FSM modeling the hardware does not have an initial state (or a set of initial states). Instead, it has a set of legal *operation states*, and it must be brought into this set of operation states from any power-up state by a *reboot sequence*.

Hardware is specified with a Hardware Description Language. In contemporary design, the *specification model* description is very close to its logic description. On the other hand, hardware is manufactured based on an *implementation model* originated from a transistor-level description. To make sure that the specification model has the intended logic functionality, *assertions* are written for the specification model in a temporal logic language, such as Linear Temporal Logic [1]. For the correct operation of hardware it is therefore necessary to

- Make sure that the reboot sequence brings the specification and implementation models into the intended set of *operation states*;
- Make sure that the specification model *satisfies the temporal assertions*, in the operation states;
- Make sure that the specification and implementation models are *equivalent*, in the operation states.

Several concepts of hardware equivalence have been proposed in the literature. They can be divided into two categories:

1. *combinational equivalence*, whose application is limited to comparing FSMs that have the same amount of state elements. The corresponding state elements in the two combinatoionally equivalent FSMs must have the same next-state functions. Thus, for combinational equivalence verification, *propositional satisfiability checking* is enough;
2. *sequential equivalence*, where FSMs with or without a set of given initial states can be compared using *model checking techniques* [1]. Sequential equivalence concepts include several forms of *replaceability equivalence*, 3-*valued equivalence*, *steady-state equivalence*, *alignability equivalence*, and *post-reboot equivalence* [7,2,9,4,6].

In this talk, we discuss recent advances in sequential equivalence verification of hardware. We will focus on post-reboot equivalence, since it preserves the validity of temporal properties between equivalent FSMs. In particular, we will

E.A. Hirsch et al. (Eds.): CSR 2008, LNCS 5010, pp. 11–12, 2008.

discuss compositional methods for proving post-reboot equivalence in a divide-and-conquer fashion [2,5,6,8]. Further, we will briefly describe the underlying algorithms and model-checking techniques used in Intel's equivalence verification tool [3]. Finally, we will present some experimental data on equivalence verification of Intel designs.

References

1. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (1999)
2. Huang, S.-Y., Cheng, K.-T.: Formal Equivalence Checking and Design Debugging. Kluwer Academic Publishers, Dordrecht (1998)
3. Kaiss, D., Goldenberg, S., Hanna, Z., Khasidashvili, Z.: Seqver: A Sequential Equivalence Verifier for Hardware Designs. In: ICCD (2006)
4. Khasidashvili, Z., Hanna, Z.: TRANS: Efficient sequential verification of loop-free circuits. In: HLDVT (2002)
5. Khasidashvili, Z., Skaba, M., Kaiss, D.: Theoretical Framework for Compositional Sequential Hardware Equivalence Verification in Presence of Design Constraints. In: ICCAD (2004)
6. Khasidashvili, Z., Skaba, M., Kaiss, D.: Post-reboot equivalence and compositional verification of hardware. In: FMCAD (2006)
7. Kohavi, Z.: Switching and Finite Automata Theory. McGraw-Hill, New York (1978)
8. Moon, I.-H., Bjesse, P., Pixley, C.: A compositional approach to the combination of combinational and sequential equivalence checking of circuits without known reset states. In: DATE (2007)
9. Pixley, C.: A theory and implementation of sequential hardware equivalence. IEEE transactions on CAD (1992)