Formal Verification of Abstract SystemC Models

Daniel Große, Hoang M. Le and Rolf Drechsler

Institute of Computer Science
University of Bremen
28359 Bremen, Germany
{grosse,hle,drechsle}@informatik.uni-bremen.de

1 Introduction

System-on-Chips (SoCs) combine hardware and embedded software on a single chip. To successfully develop such complex systems, an abstract model of the system is required to focus only on the relevant aspects at the beginning of the design process. This procedure has been systematized and the so-called *Electronic System Level* (ESL) design emerged. *SystemC* [1] has become the de facto standard for ESL design. Especially, the concept of *Transaction Level Modeling* (TLM), which enables the description of communication in terms of abstract operations (transactions), improved the success of SystemC SystemC is a C++ class library and provides modules, ports, interfaces and channels as the fundamental modeling components whereas the functionality is described by processes.

The verification of SystemC designs, i.e. ensuring the correct functional behavior is a crucial task since otherwise the follow-up costs in the subsequent design steps will explode. Hence, intensive research has been started to improve the inherent simulation-based verification methodology of SystemC. Several approaches have been introduced which allow assertion-based verification for SystemC designs. For this task properties describing the intended behavior are specified and checked during the execution of the design.

In contrast, formal verification – here we focus on property checking – either proves mathematically that a property holds or returns a counter-example which shows the violation of the property. However, only very few formal approaches which target SystemC TLM have been proposed. As pointed out in [2] the object-oriented nature of SystemC in combination with the event-driven simulation semantics makes the development of formal approaches a formidable task.

2 SystemC TLM Property Checking

Here we describe our formal property checking approach for SystemC TLM designs [3]. The properties are specified using an extension of PSL following [4]. In the properties the user can specify non-trivial behavior like for example ordering of events or relations between transactions and events. Thereby, the temporal resolution can be changed, for example to sample only at certain events. The

Dagstuhl Seminar Proceedings 09461
Algorithms and Applications for Next Generation SAT Solvers
http://drops.dagstuhl.de/opus/volltexte/2010/2510

expressiveness of the properties distinguishes our approach clearly from existing methods. As an example consider the following properties which have been specified for a TLM FIFO:

- After a *write* transaction, the FIFO is not empty:
  always (write:**exit** −>num_elements > 0)
- If the FIFO is full, the next event notified is *read_event*:
  default clock = read_event.**notified** || write_event.**notified**;
  always (num_elements == max −>**next** read_event.**notified**)
- After a notification of *read_event*, the next 10 (the FIFO size) notifications includes at least one notification of *write_event*:
  default clock = read_event.**notified** || write_event.**notified**;
  always (read_event.**notified** −>**next_e**[1:10] write_event.**notified**)

The basis of the proposed technique is provided by our fully automatic transformation from SystemC TLM to C and the generation of monitoring logic for TLM properties by means of assertions. The resulting C model including the monitoring logic can then be interpreted as a state transition system, where the transition relation is determined by the outermost loop of the scheduler. This interpretation enables a *Bounded Model Checking* (BMC) [5] formulation at the level of C models. The BMC-based verification, i.e. search for a violation of the property, is performed by running CBMC [6] on the transformed model. However, the method is not complete in our case since CBMC cannot check whether the complete state space of the SystemC design has been traversed.

In addition, for efficiency and completeness we propose an induction-based proof technique for the transformed model. The basic idea is to verify in the base case, that the property holds after the first k transitions. Then, the inductive step proves that, if after any k transitions the property is not violated, the property also holds after the $(k+1)$-th transition. In traditional induction-based methods for BMC (see e.g. [7]) both steps are performed by checking Boolean formulas. Our method, in contrast, operates on a higher level of abstraction, i.e. verifying C models. The approach was implemented and evaluated on several SystemC TLM designs. SAT/SMT-based proof techniques – available as backends for CBMC – allow proving important non-trivial behaviors of the designs efficiently.

References

1. IEEE Std. 1666: IEEE Standard SystemC Language Reference Manual. (2005)
2. Vardi, M.Y.: Formal techniques for SystemC verification. In: DAC. (2007) 188–192
3. Große, D., Le, H.M., Drechsler, R.: Induction-based formal verification of SystemC TLM designs. In: MTV. (2009)
4. Tabakov, D., Vardi, M., Kamhi, G., Singerman, E.: A temporal language for SystemC. In: FMCAD. (2008) 1–9
5. Biere, A., Cimatti, A., Clarke, E.M., Zhu, Y.: Symbolic model checking without BDDs. In: TACAS. (1999) 193–207
6. Clarke, E.M., Kroening, D., Lerda, F.: A tool for checking ANSI-C programs. In: TACAS. (2004) 168–176
7. Sheeran, M., Singh, S., Stålmarck, G.: Checking safety properties using induction and a SAT-solver. In: FMCAD. (2000) 108–125