Using RTL statespace information and state encoding for induction based property checking

Markus Wedler, Dominik Stoffel, Wolfgang Kunz
Department of EE and IT, University of Kaiserslautern/Germany
email: wedler@eit.uni-kl.de

Abstract

This paper focuses on checking safety properties for sequential circuits specified on the RT-level. We study how different state encodings can be used to create a gate-level representation of the circuit that facilitates the computation of effective invariants for induction-based property checking. Our experiments show the strong impact of state encoding on the efficiency of the induction process.

1 Introduction

Checking safety properties is an unsolved problem in design automation. We propose a method to generate invariants by using state encoding and RT-level information. For large machines that can be independently controlled we suggest a binary encoding and strengthen the property by using the upper bound given by the RT-level description. For coupled machines the proposed method uses *structural FSM traversal* [6], [7], [8], to find an over-approximation of the set of reachable states. This is only effective if an appropriate state encoding is chosen. The paper is organized as follows: Section 2 gives a brief review on induction-based property-checking. Section 3 shows how state encoding can be used to strengthen properties automatically. Section 4 gives an overview over our implementation of the ideas presented in Section 3. Section 5 reports the experimental results.

2 Induction-based Property Checking

In [5] and [1] the authors study the use of induction to verify safety properties. The basic model for sequential circuits is a finite state machine M which is a 6-tuple $M = (I, S, \delta, S_0, O, \lambda)$ where I is the input alphabet, S is the set of states, $\delta : S \times I \longrightarrow S$ is the next-state function, S_0 is a set of initial states, O is the output alphabet and $\lambda : S \times I \longrightarrow O$ is the output function. Given such a machine, we would like to check whether a condition P holds for all reachable states. If this is the case we call the machine P-safe. In the following, we identify P with the set of all states $s \in S$ where P holds.

Lemma 1. *A finite state machine M is P-safe iff there is a $k \geq 0$ such that the following conditions hold:*

- *For all paths $s_0, s_1, ..., s_k$ in the state transition graph of M with $s_0 \in S_0$, P holds in all s_i.*
- *If P holds in the states on some path $s_0, ..., s_k$ in the state transition graph of M, with pairwise different s_i then it also holds in all next states reachable from s_k, i.e., $\forall i \in I : \delta(s_k, i) \in P$.*

However for practical use of induction-based methods, we can afford only small values of k. An algorithm that is based on Lemma 1 has to unroll the transition relation $k+2$ times and solve two satisfiability problems on this. If k is unknown, as is the case in most applications the algorithm has to perform a linear search for the right k. This implies that the algorithm has to solve $2 * (k + 2)$ SAT problems.

In the next section we are going to analyze conditions that help to minimize k in Lemma 1.

3 State Encodings

Standard property checking tools start from an RT-level design specification in VHDL or Verilog that is augmented by some property to be verified. From this specification the front-end of the property checker generates a gate level description of the design and the property. This description is the basis for applying standard boolean proving techniques like SAT solving.

Several degrees of freedom exist when generating the gate level model from the RTL specification. It should be noted that the choice of an appropriate state encoding can help significantly to reduce the proving complexity. For example, in the context of *symbolic FSM traversal* it was shown in [2] that retiming can have a strong impact on the BDD sizes. In this paper, we demonstrate that this freedom can also be used to reduce the induction length k of a prover based on Lemma 1.

We will outline a method to strengthen the property with information from the RT-level and with state space information derived from *synthesis for state space representability* and a *structural FSM traversal* [6], [7], [8].

For each component of the design, we choose either binary or one hot encoding, depending on the size of the machine. Large machines like counters are encoded binary. Small machines as they occur in many controllers are encoded one hot. To catch the dependencies between the machines, we perform a *structural FSM traversal*. From this we derive a set of implications between the registers of the model. These implications together with the one hot property are an invariant that is usefull for induction methods.

4 Implementation of the property checker

In this section we give a brief overview over our implementation based on the ideas in this paper.

The VHDL of the design is augmented by combinational logic that calculates the property. We create a gate level representation of our designs using the front-end of an industrial property checker. Note again, that the encoding for verification can be chosen in the RT-to-gate front-end of the verification tool independently of the encoding that is used for the actual implementation. The property is strengthened with upper bounds for binary encoded state variables that do not consume the entire 2^n state space and with the results of a *structural FSM traversal*. After this we prove the property by induction starting with induction length 0, increasing the induction length until a proof is found, a counterexample is generated or a user-defined upperbound for the induction length is reached.

The proofs for each induction length are translated into SAT problems and given to the well-known SAT-solver CHAFF [4]. During *structural FSM traversal* implications are calculated using *recursive learning* [3]

5 Experimental Results

To examine the benefits and the limitations of our ideas we created designs that are hard to prove for induction-based methods. Table 1 shows the results for some of these designs containing k copies of identical state machines using two interchangeable resources in common. We consider the property that never more than two of them try to use the resource at the same time. The induction length and CPU-time needed to prove is reported in columns three to six for both binary (B) and one-hot(O) encoding of the component machines. The results show that *structural FSM traversal* can generate powerful invariants in the case of one-hot encoding. Sometimes it also reduces induction length in the case of binary encoding. But this reduction is not as powerful as in the case of a one-hot encoding. Instead of minutes it takes hours to finish the proof.

k	enc	induction length		CPU-time (hh:mm:ss)	
		P	**P∧Impl**	**P**	**P∧Impl**
4	O	>32	**3**	aborted	**0:00:57**
5	O	>32	**3**	aborted	**0:01:16**
6	O	>32	**3**	aborted	**0:02:15**
7	O	>32	**3**	aborted	**0:05:59**
4	B	>32	24	aborted	0:56:02
5	B	>32	26	aborted	2:49:22
6	B	>32	21	aborted	10:09:31
7	B	>32	>27	aborted	aborted

Figure 1. Multiple coupled machines with non-binary dependencies

6 Conclusion

In this paper we exploit that the choice of state encoding during verification is independent from the state encoding of the actual implementation. We show that this can be used to reduce the complexity of induction-based property checking. In particular, we introduce invariants for binary encodings and demonstrate that one-hot encoding together with *structural FSM traversal* creates useful invariants for many applications.

References

[1] P. A. Abdulla, P. Bjesse, and N. E'en. Symbolic reachability analysis based on sat solvers. In *Proc. Sixth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS-00)*, pages 411–425, March 2000.

[2] A. Kühlmann and J. Baumgartner. Transformation-based verification using generalized retiming. In *Proc.Intl. Conf. Computer Aided Verification(CAV-01)*, pages 104–117, July 2001.

[3] W. Kunz and D. Pradhan. Recursive learning: A new implication technique for efficient solutions to CAD problems: Test, verification and optimization. *IEEE Transactions on Computer-Aided Design*, 13:1143–1158, Sep. 1994.

[4] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient sat solver. In *Proc. Intl. Design Automation Conference (DAC-01)*, pages 530–535, June 2001.

[5] M. Sheeran, S. Singh, and G. Stalmarck. Checking safety properties using induction and a sat solver. In *Proc. Intl. Conf.Formal Methods in Computer-Aided Design(FMCAD 2000)*, volume 1954 of *Lecture Notes in Computer Science*. Springer, November 2000.

[6] D. Stoffel and W. Kunz. Record & play: A structural fixed point iteration for sequential circuit verification. In *Proc. Intl. Conference on Computer-Aided Design (ICCAD-97)*, pages 394–399, Nov 1997.

[7] C. van Eijk. Sequential equivalence checking without state space traversal. In *Proc. Conference on Design, Automation and Test in Europe (DATE-98)*, pages 618–623, Paris, France, March 1998.

[8] M. Wedler, D. Stoffel, and W. Kunz. Improving structural fsm-traversal by constraint-satisfying simulation. In *Proc. IEEE Computer Society Annual Symposium on VLSI 2002 (ISVLSI 2002)*, pages 151–158, April 2002.