Language-driven Validation of Pipelined Processors using Satisfiability Solvers

Prabhat Mishra  Heon-Mo Koo  Zhuo Huang

Department of Computer and Information Science and Engineering
University of Florida, Gainesville, FL 32611, USA.
{prabhat, hkoo, zhuang}@cise.ufl.edu

Extended Abstract

Due to increasing demand for faster computations, deeply pipelined processor architectures are being employed to meet desired system performance. Functional validation of such pipelined processors is one of the most complex and expensive tasks in the current Systems-on-Chip (SOC) design methodology. Language-based processor validation techniques have received considerable research interest in recent years. These techniques use an architecture description language (ADL) such as EXPRESSION [1], LISA [14], MIMOLA [13], and nML [7] to enable various validation efforts including test generation [5, 9, 10, 13] and equivalence checking [11].

![Figure 1. Language-driven Top-Down Validation Methodology](image)

Figure 1 shows a top-down validation methodology for pipelined processors. In this methodology the processor architecture is captured using EXPRESSION ADL. Next, the ADL specification is validated by

---

1This is a modified version of the flow used in the book by Mishra and Dutt [12]
analyzing both static and dynamic behaviors of the specified architecture. The validated specification is then used to generate various executable models including simulator and RTL models. Finally, the implementation is validated using a combination of simulation techniques and formal methods. Necessary test programs are generated from the ADL specification using model checking. These test programs are used for simulation of the implementation. Similarly, the generated RTL model is used to verify the implementation using equivalence checking.

While language-based validation techniques have proposed several promising ideas, many challenges remain in applying them to realistic pipelined processors. This paper addresses two practical challenges in this methodology: test generation and equivalence checking. In the remainder of this section we briefly describe the two challenges and outline our plan to address these challenges using satisfiability (SAT) checking.

1 Test Generation

Mishra et al. [10] proposed a pipeline coverage driven test generation techniques using model checking. The test generation process consists of creating necessary properties based on pipeline coverage and applying those properties using model checker to generate counterexamples. The time and resources required for test generation using this approach can be extremely large for today’s pipelined processors. Sometimes it is impossible to generate test programs when multiple pipeline stages are involved e.g., creating multiple exceptions scenarios.

We plan to use SAT solvers for test generation. SAT solvers have been successfully used in the context of counterexample generation by various researchers [6, 8]. Our initial studies have shown that SAT-based bounded model checking performs well for finding shallow counterexamples [4]. Currently, we are modifying an existing SAT solver to tune it for efficient test generation for pipelined processors.

2 Equivalence Checking

Mishra et al. [11] presented a verification flow using equivalence checking. This flow has one limitation: the structure of the generated hardware model (reference) needs to be similar to that of the processor implementation. This requirement is primarily due to the limitation of the equivalence checkers. The equivalence checkers assume structural similarity and uses the similarity to reduce the complexity. Equivalence checking is not possible if the reference and implementation designs are large and drastically different. In reality, the implementation goes through numerous optimizations due to various requirements such as cost, area, power and performance. As a result, the final implementation may not have similar structure as intended in the original specification, whereas the generated hardware model has same structure as in the specification. In other words, traditional equivalence checkers are not useful in the context of language-driven model generation and functional validation.

We plan to perform equivalence checking using SAT solvers. SAT solvers along with BDD and ATPG based techniques have been successfully used in the context of hardware verification [2, 3]. Initial studies with our SAT-based combinational equivalence checker show some promising results by exploiting the structural similarities of the designs [15]. Currently, we are developing a SAT-based sequential equivalence checker that can efficiently compare designs without any structural similarity.
References


