A Parameterized Benchmark Suite of Hard Pipelined-Machine-Verification Problems[⋆]

Panagiotis Manolios[1] and Sudarshan K. Srinivasan[2]

[1] College of Computing
[2] School of Electrical & Computer Engineering
Georgia Institute of Technology
Atlanta, GA-30318
{manolios@cc.gatech.edu, darshan@ece.gatech.edu}

Abstract. We present a parameterized suite of benchmark problems arising from our work on pipelined machine verification, in the hopes that they can be used to speed up decision procedures. While the existence of a large number of CNF benchmarks has spurred the development of efficient SAT solvers, the benchmarks available for more expressive logics are quite limited. Our work on pipelined machine verification has yielded many problems that not only have complex models, but also have complex correctness statements, involving invariants and symbolic simulations of the models for dozens of steps. Many of these proofs take hundreds of thousands of seconds to check using the UCLID decision procedure and SAT solvers such as Zchaff and Siege. More complex problems can be generated by using PiMaG, a Web application that we developed. PiMaG generates problems in UCLID, SVC, and CNF formats based on user-provided parameters specifying features of the pipelined machines and their correctness statements.

1 Introduction

Fueled in part by advances in SAT solving, there is currently wide interest in obtaining efficient decision procedures for richer logics [1, 4]. As is the case with SAT solving technology [10, 11], efficiency does not mean better worst-case behavior; rather, it means better behavior on problems "arising in practice." In contrast to the situation for SAT, where hard CNF problems arising in practice are readily available, the supply of hard benchmark problems for more expressive logics is quite limited. We believe that by providing such problems we can help spur the growth of efficient decision procedures; to this end, we provide a parameterized suite of benchmarks in UCLID [2], SVC, and CNF formats that can be used to evaluate decision procedures and SAT solvers.

The core benchmark suite comprises of 210 benchmarks generated from our work on refinement based pipelined machine verification [7, 8]. We also developed PiMaG (Pipelined Machine Generator), a Web application that can be used to automatically generate complex benchmarks based on user provided parameters [9]. The benchmarks include not only the models, but also the properties to be proved, which include invariants, symbolic simulation steps, and arithmetic.

[⋆] This research was funded in part by NSF grants CCF-0429924 and IIS-0417413.

The paper is organized as follows. In Section 2, we briefly describe the refinement-based correctness theorems. In Section 3, we describe the pipelined machine models being verified, and in Section 4, we describe the benchmark suite, the tool, and give an overview of the running times we obtained in checking these benchmarks using the UCLID decision procedure [2] and the Siege SAT solver [11]. We conclude in Section 5.

2 Correctness Theorems

The benchmarks arise from our work on refinement-based pipelined machine verification. We use the notion of Well-Founded Equivalence Bisimulation (WEB) refinement to show pipelined machines and their instruction set architecture (ISA) have the same safety and liveness properties up to stuttering [5, 6]. Refinement proofs are relative to *refinement maps*, functions from pipelined machine states to ISA states, that show us how to view a pipelined machine state as an ISA state. For example, refinement maps have to hide the pipeline components that do not appear in the ISA. In [7], it is shown how to automate the proof of WEB-refinement in the context of pipelined machine verification. Our benchmark problems use three different refinement maps; two of them are based on commitment [5, 6] and one is based on flushing [3].

The idea with commitment is that partially completed instructions are invalidated and the programmer visible components are rolled back to correspond with the last committed instruction. Flushing is a kind of dual of commitment, where partially completed instructions are made to complete without fetching any new instructions. Using refinement maps based on commitment requires the use of invariants, but they can be automatically generated [7]. We use two different types of invariants that lead to two types of commitment proofs, one of which tends to lead to significantly faster verification times [8].

3 Pipelined Machine Models

The pipelined machine models are obtained by starting from a base model and extending it with various features to obtain more complex models. The most complex model in the core benchmark suite is shown in Figure 1. The base processor model is a 6 stage pipelined machine with fetch, decode, execute, memory1, memory2, and write back stages. The pipeline stages memory1 and memory2 provide for a two-cycle memory access. Instruction types such as ALU instructions with register-register and register-immediate addressing modes, branch, loads, and stores are implemented. The base processor model is extended with features such as a pipelined fetch stage, branch prediction, an instruction queue, an instruction cache, a data cache, and a write buffer.

The pipelined machine models are described using the UCLID specification language at the term-level. The data path is abstracted away using terms (integers) and much of the combinational circuit blocks that are common between the pipelined machine and its instruction set architecture (ISA) are abstracted using uninterpreted functions. The register file and the memory are modeled using lambda expressions.

We use the following naming convention for the pipelined machine models. The model name starts with a number followed optionally by the characters “i”, “d”, “w”,

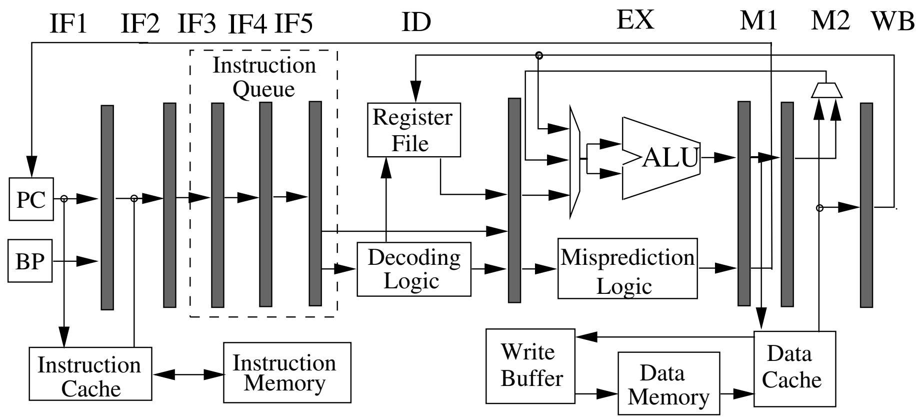

Fig. 1. High-level organization of the most complex processor model with 10 pipeline stages, instruction queue of length 3, instruction and data cache, and write buffer.

"b", and "n", which indicate the presence of an instruction cache, a data cache, a write buffer, branch prediction abstraction scheme 1, and branch prediction abstraction scheme 2, respectively. The number indicates the number of pipeline stages. If no branch prediction is used, the model predicts not taken.

4 Benchmarks

We have generated a core suite of 210 benchmarks that are available in UCLID, SVC, and CNF formats. The benchmarks are obtained from the pipelined machine models described in Section 3 using flushing and the two commitment refinement maps. Even more complex benchmarks can be generated by our tool PiMaG. The benchmarks and tool are available on the Web [9].

The benchmark naming conventions are as follows: the first letter is either "f", "c", or "g" and indicates the use of flushing, commitment approach 1, or commitment approach 2, respectively. Then the name of the pipelined machine model, as described in Section 3, follows. For machines based on commitment approach 1 only, there is an optional suffix which can either be "-i" or "-r", indicating that only the invariant proof, or only the refinement proof should be generated, respectively.

We checked many of the benchmarks using the UCLID decision procedure (Version 1.0), and the Siege SAT solver (variant 4). The UCLID decision procedure compiles UCLID specifications to SAT problems or to SVC formulas. All the benchmarks are unsatisfiable and the verification times vary from a few seconds for the simpler models to hundreds of thousands of seconds to being too complex for Siege to handle (*e.g.*, f9idw and f10id, f10idw, f9bidw, f10bid, f10bidw, f9nidw, f10nid, and f10nidw).

To obtain even more complex problems, PiMaG can be used to automatically generate pipelined machine models, their ISA specifications, and their refinement theorems. PiMaG takes seven parameters, the first specifies if the base model has 6 or 7 stages, the second selects the refinement map used, the third provides the length of the instruction queue, the fourth specifies what combination of the following three optional features to include: instruction cache, data cache, and write buffer, the fifth provides the length of the write buffer, the sixth specifies the type of branch prediction abstraction scheme, and the final parameter specifies what set of formats to generate benchmarks for. Some combinations of parameters can be too large for the tools to handle, and therefore PiMaG enforces restrictions on the size of the instruction queue and write buffer.

5 Conclusions

We presented a parameterized suite of benchmarks in various formats arising from pipelined machine verification and developed PiMaG, a Web application that can generate arbitrarily complex models and their correctness statements. Some of the benchmarks are quite complex and their verification takes hundreds of thousands of seconds; other benchmarks cannot be handled using state-of-the art tools such as UCLID and the Siege SAT solver. Our goal in making these benchmarks readily available is to help evaluate and stimulate further research in efficient decision procedures and SAT solvers.

References

[1] C. Barret, L. M. de Moura, and A. Stump. The satisfiability modulo theories competition (SMT-COMP'05), 2005. See URL `http://www.csl.sri.com/users/demoura/-smt-comp`.

[2] R. E. Bryant, S. K. Lahiri, and S. Seshia. Modeling and verifying systems using a logic of counter arithmetic with lambda expressions and uninterpreted functions. In E. Brinksma and K. Larsen, editors, *Computer-Aided Verification–CAV 2002*, volume 2404 of *LNCS*, pages 78–92. Springer-Verlag, 2002.

[3] J. R. Burch and D. L. Dill. Automatic verification of pipelined microprocessor control. In *Computer-Aided Verification (CAV '94)*, volume 818 of *LNCS*, pages 68–80. Springer-Verlag, 1994.

[4] L. M. de Moura and H. Rueß. An experimental evaluation of ground decision procedures. In *Computer aided verification (CAV'04)*, pages 162–174, 2004.

[5] P. Manolios. Correctness of pipelined machines. In W. A. Hunt, Jr. and S. D. Johnson, editors, *Formal Methods in Computer-Aided Design–FMCAD 2000*, volume 1954 of *LNCS*, pages 161–178. Springer-Verlag, 2000.

[6] P. Manolios. *Mechanical Verification of Reactive Systems*. PhD thesis, University of Texas at Austin, August 2001. See URL `http://www.cc.gatech.edu/~manolios/-publications.html`.

[7] P. Manolios and S. Srinivasan. Automatic verification of safety and liveness for XScale-like processor models using WEB-refinements. In *Design Automation and Test in Europe, DATE'04*, 2004.

[8] P. Manolios and S. K. Srinivasan. A computationally efficient method based on commitment refinement maps for verifying pipelined machines. In *Formal Methods and Models for Codesign (MEMOCODE)*, July 2005. Accepted to appear.

[9] P. Manolios and S. K. Srinivasan. A parameterized benchmark suite of hard pipelined-machine-verification problems, 2005. See URL `http://www.cc.gatech.edu/-manolios/benchmarks/charme.html`.

[10] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. *Design Automation Conference (DAC'01)*, pages 530–535, 2001.

[11] L. Ryan. Siege homepage. See URL `http://www.cs.sfu.ca/ ~loryan/personal`.