Some VIS Benchmarks

AVACS S1

July 20, 2007

We considered the benchmarks `s1269` and `PicoJava/biu` from the VIS benchmark suite [10]. The `s1269` benchmark implements an arithmetic logic unit (ALU). `PicoJava/biu` is part of the PicoJava core that implements the Java Virtual Machine Instruction Set.

1 Description

From the benchmark suite that comes with the state-of-the-art model checking environment VIS [10], we have selected the examples `s1269` and `PicoJava/biu` for generating partial design problems. The `s1269` benchmark implements an *Arithmetic Logic Unit (ALU)*, however there does not exist a detailed specification.

`PicoJava/biu` is part of the PicoJava core that itself implements the Java Virtual Machine Instruction Set. The `PicoJava/biu` component implements a *Bus Interface Unit* to the external world for the PicoJava core. Its main task is to generate requests to memory and I/O devices. Internally, it provides also an interface to the *Instruction Cache Unit (ICU)* and *Data Cache Unit (DCU)*.

The circuit `s1269` comes with 5 invariant properties and `PicoJava/biu` comes with only 1 invariant property. To have a meaningful data set for our analysis, we have introduced faults into the flattened circuit description according to three different fault models (for each fault model we have introduced 5 faults that violate at least one of the given invariant properties). For each of these faults we have randomly added black-box components "around" the faults in several configurations (i.e., differing in the number of black boxes and the size of the black boxes w.r.t. the total cost of the complete circuit).

2 Results

For examples having 1 black-box component that covers approximately 5% of the original circuit, BMX is able to compute uniform counterexamples using the $(0, 1, X)$-based approach for about 42% of all black-box configurations. This number decreases to about 35% on the average when looking at designs with 1 black-box component that covers

10% of the original circuit. For "larger" black-box components, i.e., at least 2 black-box components or more than 20% of the original circuit covered by the black-box component(s), the error detection goes below 20%. More detailed results for the $(0, 1, X)$-based approach can be found in [4, 6].

The examples of `PicoJava/biu` have also been used to evaluate the performance of state-of-the-art QBF solvers by using the problem formalization suggested in [6]. In 2006, we submitted 28 QBF benchmarks [2] to the competitive QBF evaluation that takes place every year since 2004, and as a result, only 1 out of 21 QBF solvers was able to solve our submitted QBF problems [8].

Using the approach of combining $(0, 1, X)$-logic and QBF presented in [5], we have submitted 450 QBF examples to the QBF evaluation in 2007 [3]. None of the solvers was able to solve all problems. The winner of the QBF competition, `sKizzo-0.10-qck`, was only able to solve 209 out of 450 problems in 145997 seconds. Regarding our black-box QBFs, the best QBF solver turned out to be `AQME-1NN` that was able to solve 313 problems in 63507 seconds, see [9] for details.

However, while QBF solvers are still a very young research area, there is a tremendous interest both in industry and academia and the large application domain of QBF, ranging from planning tasks [1], over vertex eccentricity calculation in hardware circuits [7] to our bounded model checking domain.

In the second funding phase, we will contribute to this research area by putting considerable effort into the development of more elaborate QBF solver concepts and methods, pushing forward QBF-based bounded model checking for incomplete circuits.

References

[1] Abdelwaheb Ayari and David A. Basin. Qubos: Deciding quantified boolean logic using propositional satisfiability solvers. In Mark Aagaard and John W. O'Leary, editors, *FMCAD*, volume 2517 of *Lecture Notes in Computer Science*, pages 187–201. Springer, 2002.

[2] Marc Herbstritt. QBF `blackbox_design` family benchmarks, 2006. Available online at `http://www.qbflib.org/family_detail.php?idFamily=616` [2007-06-08].

[3] Marc Herbstritt. QBF `blackbox-01X-QBF` family benchmarks, 2007. Available online at `http://www.qbflib.org/family_detail.php?idFamily=710` [2007-06-08].

[4] Marc Herbstritt and Bernd Becker. On SAT-based Bounded Invariant Checking of Blackbox Designs. In *Proc. of 6th IEEE Int'l Workshop on Microprocessor Test and Verification (MTV)*, pages 23–28, Austin (TX), USA, 2005. IEEE Computer Society.

[5] Marc Herbstritt and Bernd Becker. On Combining 01X-Logic and QBF. In *Proc. of 11th International Conference on Computer Aided Systems Theory - Applied Formal Verification Track*, Lecture Notes in Computer Science. Springer-Verlag, 2007. in press.

[6] Marc Herbstritt, Bernd Becker, and Christoph Scholl. Advanced SAT-techniques for bounded model checking of blackbox designs. In *Proc. of 7th IEEE Int'l Workshop on Microprocessor Test and Verification (MTV)*, pages 37–44, Austin (TX), USA, Dec. 2006. IEEE Computer Society.

[7] Maher N. Mneimneh and Karem A. Sakallah. Computing vertex eccentricity in exponentially large graphs: Qbf formulation and solution. In Enrico Giunchiglia and Armando Tacchella, editors, *SAT*, volume 2919 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2003.

[8] Massimo Narizzano, Luca Pulina, and Armando Tacchella. QBF Evaluation, 2006. Available on-line at `www.qbflib.org/qbfeval` [2006-08-02].

[9] Massimo Narizzano, Luca Pulina, and Armando Tacchella. QBF Evaluation, 2007. Available on-line at `www.qbflib.org/qbfeval` [2007-06-08].

[10] The VIS Group. VIS: A system for verification and synthesis. In *Int'l Conf. on Computer Aided Verification*, volume 1102 of *LNCS*, pages 428–432. Springer Verlag, 1996.