Shared Memory Verification for Multicore Chip Designs

Marleson Graf[1], Luiz C. V. dos Santos[1] (advisor)

[1]Graduate Program in Computer Science (PPGCC)
Federal University of Santa Catarina (UFSC)
Florianópolis – SC – Brazil

marleson.graf@posgrad.ufsc.br, luiz.santos@ufsc.br

Abstract. *A multicore chip usually provides a shared memory abstraction implemented by a cache coherence protocol. On-chip coherence can scale gracefully as the number of cores grows, and it plays a major role for general purpose applications. Besides, multicore architectures are likely to relax constraints on store atomicity and on the ordering between loads and stores. As a result, the validation of shared memory faces two main challenges: the higher number of valid execution behaviors and the larger coherence protocol's state space. This dissertation faces those challenges and targets an important design automation phase: the (pre-silicon) functional verification of the shared memory subsystem of a multicore chip, whose behavior is specified by a memory consistency model (MCM). The main scientific contribution is a novel approach to the building of MCM checkers, along with technical contributions on random test generation and directed test generation. The contributions were reported by two papers in a premier IEEE/ACM conference and two articles in the most prestigious IEEE journal on Computer Aided Design of Integrated Circuits and Systems.*

1. Motivation, relevance and challenges

A multicore chip usually relies on a coherent shared-memory abstraction implemented by a cache coherence protocol. It has been shown that (with proper design decisions) on-chip hardware coherence can scale gracefully as the number of cores grows [Martin et al. 2012]. Since scaling estimates show that the number of active cores is limited by thermal power, cache coherence can be expected to keep playing a major role for multicore chips targeting general purpose applications. Besides, multicore architectures are likely to relax sequential consistency constraints on store atomicity and on the ordering between loads and stores [ARM 2018, RISC-V 2019]. Such constraints were originally imposed by sequential consistency for keeping a simple abstraction for parallel programming, but they are not mandatory anymore [Hennessy and Patterson 2017], because nowadays most programs are synchronized. Such trends impose two main challenges to the validation of the shared memory subsystem in a multicore chip: (1) the higher number of valid behaviors (arising from the relaxation of sequential consistency) and (2) the larger coherence protocol's state space (induced by growing core counts).

In such a context, the dissertation targets an important phase of the design automation of a multicore chip: the (pre-silicon) *functional verification* of the coherent shared memory subsystem, whose behavior is specified by a *memory consistency model* (MCM). It involves the generation of parallel programs (tests), the simulation of their execution over a *design representation* of the actual chip, and the verification of the observed behaviors according to the MCM.

On the one hand, litmus test generation is an effective approach to exposing shared memory errors without the need for specialized checkers, but it has limited coverage when used for functional verification. On the other hand, *random test generation* (RTG) and *directed test generation* (DTG) are alternative approaches to functional verification leading to higher coverage, but they require an independent checker to validate the observed memory behaviors at runtime. This tool is called an *MCM checker*. The checking is based on the analysis of memory traces, which are obtained by *monitors* in each core domain.

However, when the observability is limited to a single monitor per core (like it happens in conventional approaches for prototype validation), such analysis results in an intractable problem. That is why runtime MCM checkers should exploit the higher observability of the design representation and observe multiple monitors per core domain, thereby allowing scalability without jeopardizing verification guarantees.

Moreover, conventional checkers assume that stores are atomic at the architecture level, albeit they are inherently non-atomic operations when observed at the microarchitecture level, which makes it harder to tell valid from invalid behaviors. Although some architectures have been simplified so that the programmer always sees atomic store behaviors (like ARMv8 and RISC-V), aggressive implementations of such architectures tend to exhibit non-atomic behaviors. As a result, a checker that exploits higher observability (to provide higher verification guarantees) may end up exposing non-atomic store behaviors in face of aggressive implementations, which leads to false positives (thereby compromising verification guarantees).

Thus, a new checker should be able to exploit higher observability without raising false positives due to non-atomic behaviors.

2. Contributions

The main scientific contribution of the dissertation is a **novel approach for building MCM checkers**. It properly handles speculative effects and non-atomic store behavior under extended observability [Graf et al. 2019].

Since test generators were required to properly evaluate the new checker, the dissertation also involved innovative technical contributions on **algorithms for random test generation** [Andrade et al. 2018] and **directed test generation** [Andrade et al. 2020a, Andrade et al. 2020b].

All such techniques were integrated within a **verification framework,** which was developed under a collective research and implementation effort involving the author and other fellow students.

2.1. Originality and impact

The proposed approach allows the customization of efficient runtime checkers according to architecture and microarchitecture targets. To ensure scalability without compromising verification guarantees, multiple monitors per core domain are defined. As opposed to conventional checkers, which are unable to properly handle behavior arising from relaxed store atomicity, the proposed approach allows the building of checkers compliant with either relaxed or strict atomicity.

The keys to the originality of the main contribution are (1) an **abstract specification** (where shared memory behavior is captured by axioms in terms of *abstract* load/store events), (2) an **observability template** (where *physical* events are used as proxies for the abstract events), and (3) a novel event-driven, **scalable algorithm** (where the abstract axioms are checked over the observability template).

The potential impact of the approach comes from two pragmatic reasons: (1) the abstract specification is not tied to a particular implementation, and it combines architecture-independent (reusable) axioms and architecture depend (customizable) axioms; (2) the observability template is largely independent of microarchitecture. This allows the practical use of the approach for a broad range of architectures and a large variety of microarchitectures.

2.2. Publications

A significant part of the text of the dissertation reflects other documents written in co-authorship, and were published in the proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD) and on the journal IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems (TCAD), as follows:

- ICCAD 2019 paper on MCM checking [Graf et al. 2019];
- ICCAD 2018 paper on DTG [Andrade et al. 2018];
- TCAD 2020 article on RTG [Andrade et al. 2020a];
- TCAD 2020 article on DTG [Andrade et al. 2020b].

It should be noted that in the research area of Computer Aided Design of Integrated Circuits and Systems (also known as Design Automation), TCAD is the most prestigious journal and ICCAD is the premier and most selective conference devoted to technical innovations in design automation (along with the Design Automation Conference). Besides, a journal article generalizing the technique proposed in [Graf et al. 2019] is in final elaboration for submission to TCAD in the second quarter of 2021.

3. The verification framework

The framework relies on coverage-directed random test generation and MCM checking, and it is split into different cooperating engines, as depicted in Figure 1. The main contribution of the author lies in the MCM checker, but he also contributed to the algorithms of the Directing Engine and the RTG Engine, and he implemented the Coverage Analyzer.

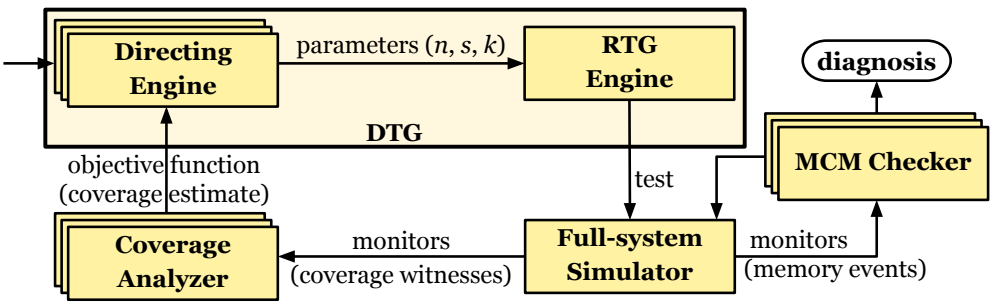

Figure 1. An overview of the framework under construction.

The framework's generation engines were designed to be reusable across derivative designs, different protocol variants, and distinct coverage metrics. The RTG engine

relies on three parameters to constrain the building of a test program: the number of memory operations (n), the number of shared memory locations (s), and the number of distinct cache sets to which the locations can be mapped (k).

While the simulator executes a test program, monitors observe memory events at relevant points of each core domain. A checker analyzes the monitored events at runtime according to the axioms of the target MCM. Besides, other monitors observe events that serve as coverage witnesses from which a coverage analyzer computes the cumulative coverage of all tests executed so far. The directing engine takes that coverage value into account before selecting the next setting of parameters for RTG.

4. Methodology and main experimental results

We compared the checkers built under the proposed approach with a conventional baseline checker, which is based on multiple relaxed scoreboards (the chosen baseline is the most recently published runtime checker whose algorithm is available).

The new checkers and the baseline checker were evaluated within the same verification framework (described above) in two distinct designs: (1) a conservative design with strict *atomic* store behavior; (2) an aggressive design with relaxed *non-atomic* store behavior. They were evaluated when running the same test suites, each built with many programs of fixed size.

For the experimental evaluation, we employed design representations with 8, 16, and 32 cores for a 2-level MOESI cache coherence protocol. The target architecture was ARMv8. A microarchitecture supporting *out-of-order* execution was adopted for all cores. For test generation, we adopted a DTG technique [Andrade et al. 2018], and employed different test sizes (1ki, 2ki, and 4ki memory operations). For each test size, the generator was executed 12 times exploiting distinct random seeds, leading to 12 test suites containing different programs.

To quantify false positives, we employed *correct* designs. When verifying correct designs with non-atomic store behavior, the baseline checker exhibited non-negligible fractions of false positives, as shown in Figure 2. For a given core count, this fraction increases significantly with test size, which is rather inconvenient, because large tests are usually required to expose the most subtle errors. For programs with 4Ki instructions, the conventional checker raised false positives for 1/3 of the test suites when targeting correct, aggressive 32-core designs On the other hand, the new checker did not raise any false positive at *all* in every scenario.

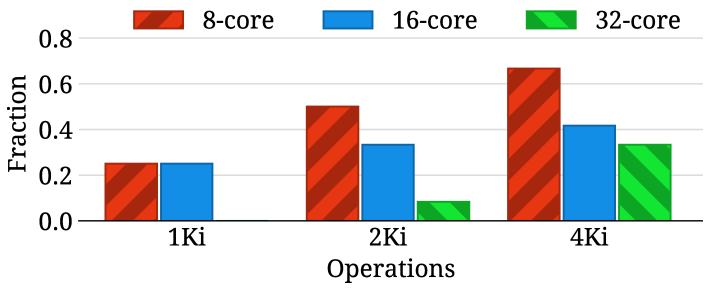

Figure 2. Fraction of false positives raised by the baseline checker for correct, aggressive designs.

To evaluate the overhead of the proposed approach, *faulty* designs were employed, and the times required to expose an error were compared. To build faulty designs, nine different errors were inserted to challenge the checkers. As a result, nine distinct designs were obtained (D1 to D9), each with a single different error.

The checker built under the proposed approach was able to find all the studied design errors, while the baseline checker was unable to find one of them. The new checker displayed negligible effort overhead when compared to the conventional one, and often led to effort reduction, as depicted in Figure 3. The maximum overhead was 2.5%, while the maximum improvement in effort was 16%.

The results indicate that the versatility of the approach and the improvement of the verification quality require negligible additional effort, being adequate to verify designs with different degrees of relaxation of store atomicity.

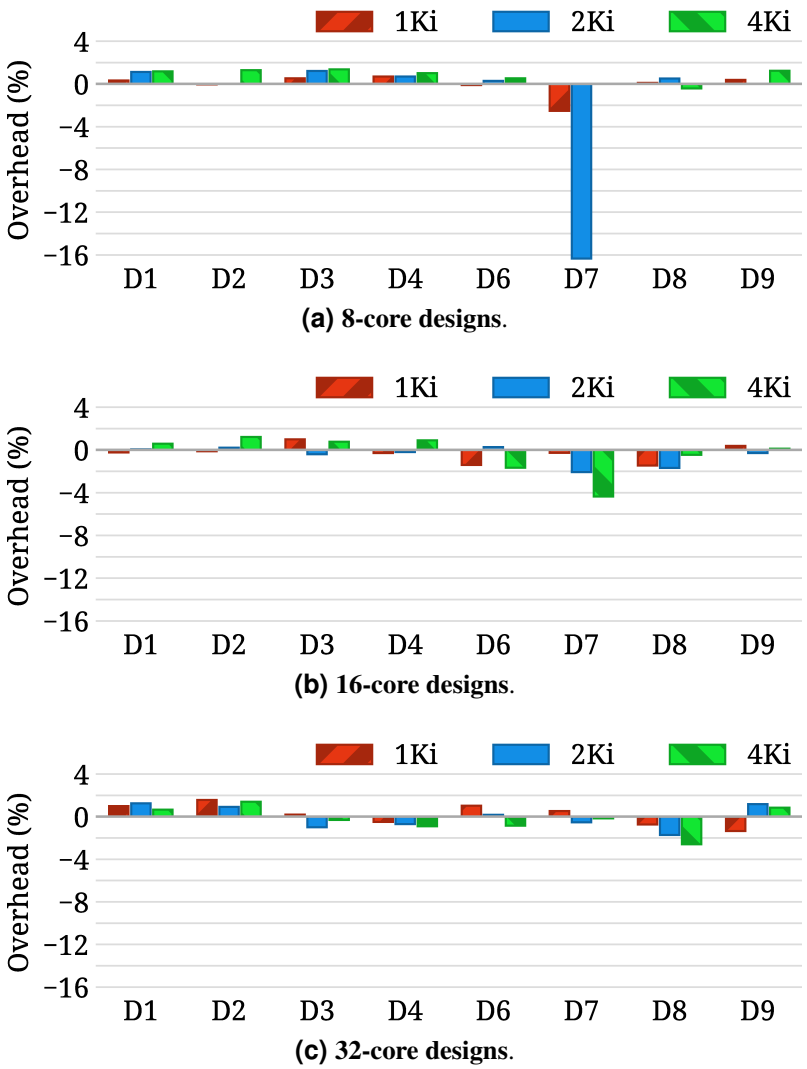

Figure 3. Effort overhead for faulty conservative designs.

5. Conclusions

The dissertation proposed a novel approach to build checkers that is able to handle aggressive designs with speculative effects and non-atomic store behavior. The experimental evidence indicates that a checker produced with the novel approach is effective, it often reduces the effort to detect an error, and it is suitable to checking designs with different degrees of store atomicity relaxation.

The dissertation not only proposed a novel approach as its main scientific contribution [Graf et al. 2019], but it also provided innovative technical contributions to test generation leading to high coverage with small effort [Andrade et al. 2018, Andrade et al. 2020a, Andrade et al. 2020b]. Finally, the required implementation effort contributed to the building of a verification framework that is effective to discover errors and can support different architectures and distinct microarchitecture variants.

References

Andrade, G. A. G., Graf, M., and dos Santos, L. C. V. (2020a). Chaining and Biasing: Test Generation Techniques for Shared-Memory Verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(3):728–741.

Andrade, G. A. G., Graf, M., Pfeifer, N., and dos Santos, L. C. V. (2018). Steep Coverage-ascent Directed Test Generation for Shared-memory Verification of Multicore Chips. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

Andrade, G. A. G., Graf, M., Pfeifer, N., and dos Santos, L. C. V. (2020b). A Directed Test Generator for Shared-Memory Verification of Multicore Chip Designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):5295–5303.

ARM (2018). *ARM Architecture Reference Manual ARMv8, for ARMv8-A architecture profile*.

Graf, M., Henschel, O. P., Alevato, R. P., and dos Santos, L. C. V. (2019). Spec&Check: An Approach to the Building of Shared-Memory Runtime Checkers for Multicore Chip Design Verification. In *International Conference on Computer-Aided Design*, pages 1–7.

Hennessy, J. L. and Patterson, D. A. (2017). *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 6th edition.

Martin, M. M., Hill, M. D., and Sorin, D. J. (2012). Why on-chip cache coherence is here to stay. *Communications of the ACM*, 55(7):78–89.

RISC-V (2019). *The RISC-V Instruction Set Manual Volume I: Unprivileged ISA*. Waterman, Andrew and Asanovi, Krste.