Approaches to Stand-alone Verification of Multicore Microprocessor Caches

Mikhail Petrochenkov <petroch_m@mcst.ru>
Irina Stotland <stotl_i@mcst.ru>
Ruslan Mushtakov <mushtakov_r@mcst.ru>
MCST, 1 Nagatinskaya st., Moscow, 117105, Russia

Abstract. The paper presents an overview of approaches used in verifying correctness of multicore microprocessors caches. Common properties of memory subsystem devices and those specific to caches are described. We describe the method to support memory consistency in a system using cache coherence protocol. The approaches for designing a test system, generating valid stimuli and checking the correctness of the device under verification (DUV) are introduced. Adjustments to the approach for supporting generation of out-of-order test stimuli are provided. Methods of the test system development on different abstraction levels are presented. We provide basic approach to device behavior checking — implementing a functional reference model, reactions of this model could be compared to device reactions, miscompare denotes an error. Methods for verification of functionally nondeterministic devices are described: the «gray box» method based on elimination of nondeterministic behavior using internal interfaces of the implementation and the novel approach based on the dynamic refinement of the behavioral model using device reactions. We also provide a way to augment a stimulus generator with assertions to further increase error detection capabilities of the test system. Additionally, we describe how the test systems for devices, that support out of order execution, could be designed. We present the approach to simplify checking of nondeterministic devices with out-of-order execution of requests using a reference order of instructions. In conclusion, we provide the case study of using these approaches to verify caches of microprocessors with “Elbrus” architecture and “SPARC-V9” architecture.

Keywords: multicore microprocessors; cache memory; out-of-order execution; test system; nondeterministic behavior; model-based verification; stand-alone verification; “SPARC-V9”; “Elbrus-8C”.

DOI:10.15514/ISPRAS-2016-28(3)-10

For citation: Petrochenkov M., Stotland I., Mushtakov R. Approaches to Stand-alone Verification of Multicore Multiprocessor Cores. *Trudy ISP RAN/Proc.ISP RAS*, vol. 28, issue 3, 2016, pp. 161-172. DOI: 10.15514/ISPRAS-2016-28(3)-10

1. Introduction

The key feature of modern microprocessor architecture is multicoreness — combining several computational cores on a single system on a chip (SOC). To reduce time needed to access RAM (Random Access Memory), device can incorporate several levels of cache hierarchy. Access to smaller caches could be executed faster than access to larger caches of the next level of the hierarchy. Caches can keep data for a single computational core or serve as data storage for several of them at the same time. A memory subsystem of a multicore microprocessor must maintain coherence of the memory. Task of maintaining correct state of memory is usually solved by implementing cache coherence protocol that defines a set of data states and actions on transitions between states in a cache [1]. To optimize the design and the implementation of coherency protocol, caches can include a local directory — the device which keeps information on states of data in different components of the memory subsystem. Sufficient complexity of protocols and their implementations in multilevel memory subsystems can lead to hard to find errors.

To ensure the robustness of a microprocessor, one must thoroughly verify its memory subsystem. The importance of the *functional verification* — the checking of correspondence between specifications of designs and their implementations — is obvious for many reasons. This activity could be found out to take more than 70% out of the total design development time. Two main approaches to functional verification of microprocessors are formal verification and simulation-based methods [2]. Formal methods are exhaustive and based on analyzing static formal model. Models are large and formal verification techniques face the “combinatorial explosion” issue. Simulation-based methods are not exhaustive, but they are much more flexible and thereby employed at different stages. We can verify not only the static model of system, but also implementation. The object of simulation-based verification is RTL (Register Transfer Level) model of device.

One of the approaches to microprocessor verification is execution of test programs on the microprocessor model and on the reference implementation of its instruction set, and comparison between them. Such approach is called *system verification*. It should be noticed that caches are often invisible from the point of view of a programmer. That is why designing programs capable for sufficient verification of a microprocessor caches is a complex task.

One way to shorten the design of microprocessors is the application of unit-based verification. It is assumed that system is divided into a set of components and the general functionality of the components does not change [3]. Such a way of verification is called *stand-alone verification*. This paper addresses the problem of stand-alone verification of microprocessor caches of different levels.

The rest of the paper is organized as follows. Section 2 suggests an approach to the problem. Section 3 presents the test stimui generation methods. Section 4 reviews the existing techniques for designing test oracles. Section 5 describes a case study

on using the suggested approach in an industrial setting. Section 6 concludes the paper.

2. Common View on Stand-alone Verification of Microprocessor Caches

The object of stand-alone verification is model of the device under verification (DUV) implemented in hardware description language (usually, Verilog or VHDL). It defines the behavior of the device on a register transfer level. The device specification defines a set of stimuli and reactions based on the state of the device. To check the correctness of the device it is included in a test system — a program that generates test stimuli, checks validity of reactions and determines verification quality. Based on its functions test system can be divided into separate modules — stimulus generator, correctness checking module (test oracle) and coverage collector. Methods of estimation of verification quality are similar to that of other devices: information on functional code coverage is used to identify unimplemented test scenarios and refine stimulus generation by adding new test scenarios and improving existing stimulus generator. This approach is called coverage driven constrained random verification. Besides this, there are some approaches to microprocessor caches verification. In paper [4] authors propose using decomposition and abstraction for standalone verification. In our previous projects, we have used the decomposition methods for L2-cache verification: L2-cache was divided into several submodules for which reference cycle-accurate bit-to-bit models and test systems were implemented [3]. This approach allowed to find bugs in submodules, but did not give the chance to check the cache in general. We also can use a SystemC reference model as presented in [5] but it is employing if SystemC models are used in other stages of an ASIC design flow. In paper [6] the approach to test oracle development for nondeterministic models is presented. However, this approach refers only test oracle developing of in-order cache execution and has no recommendation for caches with out-of-order execution. In such a way, the main goal of this work lies in developing some new techniques of standalone verification of microprocessor caches with different ordering of stimulus execution.

Cache behavior exhibits a set of properties that should be considered while designing a test system for verification of the device:

- Transactions (or requests) in the microprocessor system can be separated into three groups: *primary requests* — requests from subscribers (other caches, cores, etc.) to perform an operation with the memory (load/store), *secondary requests* — responses of the test system to some reaction of the cache, and *reactions* — output transactions from the cache
- A device implements a part of cache coherence protocol
- A device works independently with different cache lines — areas of memory of fixed size

- Requests that work with the same cache line are serialized. It means that requests complete in the same order as they are received
- Device implements data eviction mechanism and protocol to determine victim line (usually some variant of least recent used algorithm (LRU)).

Using these properties of the device under testing while designing a test system could lead to the simplified structure using separate stimulus generators — the primary requests generator and the secondary requests generator. We also can use the fact that requests are serialized for checking the correctness of caches with out-of-order execution.

3. Test Stimuli Generation

3.1 The common approach

Test stimuli are usually generated at more abstract level than register transfers and interface signals. Based on the logical and functional similarity, groups of device ports are combined into interfaces. Interfaces are used to transfer transaction level packets [7]. To transform packets between different representations on signal and transaction level, serializer and deserializer modules are implemented[8].

Test system should generate stimuli similar to that in a real system. Should be noted that primary requests in real microprocessor are consequences of some memory access operation (loading, storing data, eviction, prefetch, atomic swap, etc). Secondary requests are answers for reaction packets from the device. It is usually convenient to use only a sequence of primary requests as a test sequence, and generate secondary requests automatically in corresponding modules. Properties of secondary requests could be changed based on secondary request generation modules configuration.

In the test system interfaces are combined into groups that represent working with some devices. A test system should simulate the state of these devices to generate correct responses from it.

3.2 Generation of Primary Requests for Caches with Out-of-order Execution

Properties of the devices that support out-of-order execution should be considered while designing a stimulus generator:

- Order of primary request can be different from the order of memory accesses in initial program
- Primary request could be divided into several messages accepted at different times. Messages for one primary request are identified by common value of tag field
- Request canceling mechanism is present.

To support out-of-order execution of memory access requests in a cache common approach was augmented. The module responsible for transferring of primary requests was replaced with high-level module that includes components working with interfaces of primary request parts. The order of the request for the module is identical to that of the test program, and reordering of request parts is executed based on module settings.

4. Correctness Checking

Let us consider the existing approaches to reaction checking. Richard Ho suggested two main methods: self-checking tests and co-simulation [9]. Co-simulation is a method for reaction checking in which an independent reference model is used along with the target design model [4]. The two models are co-simulated using the same stimuli and their reactions are verified.

A reference model is implemented either in general purpose programming language (C, C++) or in specialized hardware verification language (SystemVerilog, “e”, Vera). If test stimuli are the same, difference in model and device reactions means an error somewhere in the system[8]. Reference models could be cycle-accurate or untimed functional. To implement the cycle-accurate model, behavior of the device must be specified on a register transfer level. Behavior of caches usually defined on a higher level of abstraction, because cache is not an essential part of a computational pipeline of a microprocessor. A cache is not a subject of strict temporal requirements. Besides this, the development of cycle-accurate model is labor-intensive when the design specification is changing and no stable through the verification phase. To simplify the development of reference models TLM (Transaction Level Modeling) is often used [4]. To verify caches we also propose to implement functional models working on transaction level.

4.1 Checking of nondeterministic caches

If one wants to develop functional model of cache, its specification must have property of transaction level indeterminism. That is, identical transaction level traces of stimuli (a set of RTL traces is mapped into this single transaction level trace) must cause identical transactional reaction trace. It should be noted that caches often include a set of components (eviction arbiter, primary request arbiter serving different requesters), that do not hold that property. That is, different RTL traces that are mapped into the single transaction level trace could lead to different reaction traces. There are several methods to check the behavior of nondeterministic devices.

4.1.1 “Gray box” checking

One of the ways to solve aforementioned problem is to replace usual “black box” method of device verification. That is, we should not consider only external interfaces of the device while analysing its behavior. To determine which variant of

behavior was happened in the cache one could use “hints” from the implementation. To use this approach, a set of internal interfaces and signals is defined and its behavior is specified. This interfaces must be chosen in a way that information on their state could be used to eliminate nondeterminism. In general, for caches such signals are the results of primary request arbitration and the interfaces of finite automata of the cache eviction mechanism. Additionally, that information can be used in a request generator and for the estimation of verification quality. This method is usually easy to implement. Drawbacks of this methods are additional requirements for specification and reliance on interfaces that could also exhibit erroneous behaviour.

4.1.2 Dynamic refinement of transaction level model

Another approach is to create additional instances of model for each variant of behavior in case of nondeterministic choice in the device [6]. Each reaction is checked against every spawned device model. If reaction is impossible for one variation of behavior, then it is removed from set. If set of possible states after some reaction becomes empty, the system must return an error. In general, this approach may cause exponential growth of number of states with each consecutive choice. However, for caches it could be implemented efficiently, because of several properties of caches: serialization of requests and cache line independence. Information on which nondeterministic choice was made in the device (for use in a request generator or for verification quality estimation) could also be extracted from reactions. The strong point of the approach compared to “gray box” method is elimination of reliance on implementation details of the device. Drawback is additional complexity of implementation.

4.1.3 Assertions

A test system generator imitates an environment of DUV. It also should be noticed that interaction between the device and its environment must adhere to some protocol. Based on that protocol, we can include functional requirements of protocols as an assertions in the generator. Then, violation of an assertion signals an error. Usage of assertions is an effective method of detection of a broad class of errors. In addition, to assertions that are common for all memory subsystem devices, several cache-specific assertions could be included. They represent invariants of cache coherence protocol. To check this invariants, coherence of states of a single cache line is analyzed in all parts of test system after each change.

4.2 Checking caches with out-of-order execution

Caches that support out-of-order request execution exhibit properties of limited nondeterminism. That is the memory access request are received in the device in

multiple parts from several interfaces, with different unspecified timing characteristics. On the other hand, there is the "reference" order of memory access operations presented in original test program. If out-of-order execution introduces error to the canonical order, device must be cleaned and erroneous transactions must be restarted. Results of operations that completed successfully are deterministic. Based on these properties of the device, we propose to implement models of two types:

- "Ignoring the cancelled transactions" mode
- Strict checking mode

In the first mode the result of checking is delayed until the moment of the request full completion. If completion was unsuccessful, checks are not made. In the strict mode we use the approaches which is similar to the dynamic refinement of model. Set of possible device states is maintained, and it is augmented with each stimulus and reaction. The number of possible states is limited by the number of simultaneously executed out-of-order requests. Shortcomings of the first mode are delays between erroneous transaction and the execution of actual checking and reduction of the set of errors that could be detected (for example, unnecessary cancel of request will not be detected). On the other hand, implementing that mode is much simpler task, so verification could be started sooner.

5. Case Study

The approaches described above were used for stand-alone verification of L2-cache[3] and the L3-cache[6] of the microprocessor with "Elbrus" architecture and L1Data-cache (L1dc) of the microprocessor with "SPARC-V9" architecture. The test systems for stand-alone verification of this caches were developed using Universal Verification Methodology (UVM) [10].

5.1 Checking the "SPARC-V9" L1Data-cache with out-of-order execution

L1dc supports out-of-order execution of memory access operations. The test system structure for L1DC is presented in fig. 1.

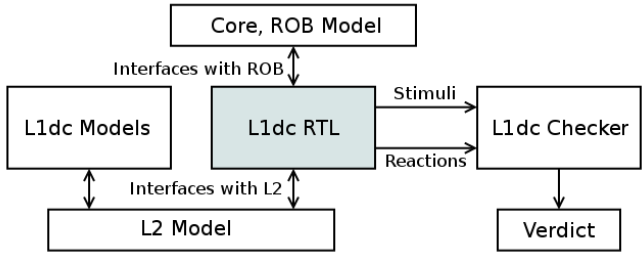

Fig. 1. The principal structure of the test sytem forL1Data-cache of the"SPARC-V9" microprocessor.

The test sequences for L1dc are the memory access assembly instructions. They are sent to computational core and the reordering buffer (Core, ROB Model). In this module, the instructions are split into multiple messages (containing either operation type, address or data). These messages are reordered and sent to DUV. Additionally, this module keeps information about initial order of instructions received, to send completion messages in correct order.

5.2 Checking the "Elbrus" L3-cache with nondeterministic behavior

The test stimulus generator was developed to verify the L3-cache of the "Elbrus" microprocessor[6]. It is based on simplified model of microprocessor core with the L2-cache and the model of system commutator that simulates work in multiprocessor environment. If multiple cores request access to a single cache line, then the order of their execution is unspecified and defined by the device microarchitecture. Internal structure of a cache is also a subject of change, due to changes to requirements of physical design. To verify the device the approach based on dynamic refinement of behavioral model was chosen. To supplement that approach, a set of assertions were implemented in stimulus generator to check validity of the system state. The approach allowed using the same test system with minimal alteration for the next iteration of the "Elbrus" microprocessor.

6. Conclusion

The approaches described in this paper allows avoiding some shortcomings. It could help to avoid excessive subdivision of the verified unit on small subdevices and developing cycle-accurate models of them (as we done in our previous projects) on the one hand and the development and maintaining of complex cycle-accurate reference models of caches on the other. The approaches were used for stand-alone verification of caches of microprocessors developed by MCST. Stand-alone verification allowed finding several errors in different caches. The intermediate results of application introduced approaches in the multicore microprocessor caches verification if presented in table 1. We already had verified the L3-cache of the "Elbrus" microprocessor using another approach and we could find new 7 errors more with help of developed tests system based on nondeterministic caches checking approach.

	Verified caches		
	L2-cache "Elbrus"	***L3-cache "Elbrus"***	***L1 data cache "SPARC-V9"***
Number of bugs	4	7	12

The test systems are developed as a UVM-environment. They were implemented to be flexible enough to set both the pseudorandom and directed test sequences. Using

of aforementioned approaches while developing test systems helped find some new errors and simplify the test system development. Approaches could be used to verify other caches of different multicore microprocessors regardless of its architectures.

Our future research is connected with improving the error diagnostics and localization of found bugs.

References

[1]. Sorin D.J., Hill M.D., Wood D.A. A Primer on Memory Consistency and Cache Coherence. Morgan and Claypool, 2011, 195 p.

[2]. Bergeron J. Writing testbenches: functional verification of HDL models. Boston: Kluwer Academic Publishers, 2003.

[3]. Stotland I, Meshkov A., Kutsevol V. Standalone functional verification of multicore microprocessor memory subsystem units based on application of memory subsystem models. Proc. of IEEE East-West Design & Test Symposium (EWDTS 2015), Batumi, Georgia, September 26-29, 2015, pp. 326-330.

[4]. Kamkin A., Chupilko M. A TLM-based approach to functional verification of hardware components at different abstraction levels. Proc. of the 12th Latin-American Test Workshop (LATW), 2011, pp. 1-6.

[5]. Tessier T., Lin H., Ringoen D., Hickey E., Anderson S. Designing, verifying and building an advanced L2 cache sub-system using SystemC. Proc. of Design and Verification Conference (DV-CON), 2012, pp.1-8.

[6]. Kamkin A., Petrochenkov M. A Model-Based Approach to Design Test Oracles for Memory Subsystems of Multicore Multiprocessors. Trudy ISP RAN / Proc. ISP RAS, vol. 27, issue 3, pp. 149-157.

[7]. TLM-2.0.1. TLM Transaction-Level Modeling Library. (online publication). Available at: http://www.accellera.org/downloads/standards/systemc (accessed 20.05.2016).

[8]. Stotland I., Lagutin A. Using stand alone behavioural models to verify microprocessor components. Voprosy radioelektroniki, seriya EVT [Issues of radio electronics], 2014, issue 3, pp. 17-27.

[9]. Ho R. Validation tools for complex digital designs. PhD thesis, Standford University, 1996.

[10]. Standard Universal Verification Methodology (online publication). Available at: http://accellera.org/downloads/standards/uvm (accessed 20.05.2016).

Подходы к автономной верификации кэш-памятей многоядерных микропроцессоров

[1] *М.В. Петроченков <petroch_m@mcst.ru>*
[1] *И.А. Стотланд <stotl_i@mcst.ru>*
[1] *Р.Е. Муштаков <mushtakov_r@mcst.ru>*

[1] *АО «МЦСТЭ», 117105, Россия, г. Москва, ул. Нагатинская, д.1, стр.1*

Аннотация. В статье приведен обзор методов, применяемых при проверке корректности поведения кэш-памятей многоядерных микропроцессоров. Описаны общие свойства устройств подсистемы памяти микропроцессора, а также свойства, специфичные для кэш-памятей, и метод поддержки согласованности состояния памяти в системе на основании протокола когерентности. Представлены подходы к проектированию тестовой системы, генерации корректных тестовых воздействий и проверке правильности поведения тестируемого устройства. Предложены модификации общего подхода к генерации тестовых воздействий для устройств с внеочередным исполнением инструкций. Приведены способы разработки тестовых систем на различных уровнях абстракции. В статье описан основной способ проверки поведения устройства на уровне транзакций — разработка эталонной поведенческой модели для последующего сравнения реакций устройства с эталонными; расхождения в реакциях сигнализируют об ошибке. Выделены критерии применимости данного подхода. Описаны методы верификации устройств, поведение которых функционально не детерминировано на уровне транзакций: метод «серого ящика», базирующийся на анализе внутренних интерфейсов устройства, для устранения возникающей неопределенности в поведении устройства. Кроме того, приведен новый метод, основанный на динамическом уточнении поведенческой модели на основе реакции устройства. Также рассмотрены преимущества использования утверждений утверждения в генераторе тестовых воздействий в качестве дополнительных методов обнаружения ошибок. В работе приведен метод, позволяющий упростить проверку поведения устройств с внеочередным исполнением инструкций, основанный формировании эталонной очереди их выполнения. В заключение представлены результаты применения предложенных подходов к верификации кэш-памятей многоядерных микропроцессоров архитектуры «Эльбрус» и «SPARC-V9».

Ключевые слова: многоядерный микропроцессор; кэш-память; внеочередное исполнение; тестовая система; недетерминированное поведение; верификация на основе эталонных моделей; автономная верификация; «SPARC-V9»; микропроцессор «Эльбрус».

DOI:10.15514/ISPRAS-2016-28(3)-10

Для цитирования: Петроченков М.В., Стотланд И.А., Муштаков Р.Е. Подходы к автономной верификации кэш-памятей многоядерных микропроцессоров. Труды ИСП РАН, том 28, вып. 3, 2016 г., стр. 161-172 (на английском). DOI: 10.15514/ISPRAS-2016-28(3)-10.

Список литературы

[1]. Sorin D.J., Hill M.D., Wood D.A. A Primer on Memory Consistency and Cache Coherence. Morgan and Claypool, 2011. 195 p.

[2]. Bergeron J. Writing testbenches: functional verification of HDL models. Boston: Kluwer Academic Publishers, 2003.

[3]. Stotland I, Meshkov A., Kutsevol V. Standalone functional verification of multicore microprocessor memory subsystem units based on application of memory subsystem models. Proc. of IEEE East-West Design & Test Symposium (EWDTS 2015), Batumi, Georgia, September 26-29, 2015, pp.326-330.

[4]. Kamkin A., Chupilko M. A TLM-based approach to functional verification of hardware components at different abstraction levels. Proc. of ehe 12th Latin-American Test Workshop (LATW), 2011, pp. 1-6.

[5]. Tessier T., Lin H., Ringoen D., Hickey E., Anderson S. Designing, verifying and building an advanced L2 cache sub-system using SystemC. Proc. of Design and Verification Conference (DV-CON), 2012, pp.1-8.

[6]. Kamkin A., Petrochenkov M. A Model-Based Approach to Design Test Oracles for Memory Subsystems of Multicore Multiprocessors. Trudy ISP RAN, vol. 27, 3, p 149-157.

[7]. TLM-2.0.1. TLM Transaction-Level Modeling Library. (online). Доступно по ссылке: http://www.accellera.org/downloads/standards/systemc (дата обращения 20.05.2016).

[8]. Стотланд И.А., Лагутин А.А. Применение эталонных событийных моделей для автономной верификации модулей микропроцессоров. // Вопросы радиоэлектроники, сер. ЭВТ, 2014, вып. 3, стр. 17-27.

[9]. Ho R. Validation tools for complex digital designs.PhD thesis, Standford University, 1996.

[10]. Standard Universal Verification Methodology (online). Доступно по ссылке: http://accellera.org/downloads/standards/uvm (дата обращения 20.05.2016).