Snoop Filter Efficiency Analysis by Directed Random Testing

A M Antonova\textsuperscript{1,2}, S I Aryashev\textsuperscript{1}, N A Grevtsev\textsuperscript{1}, P S Zubkovsky\textsuperscript{1},
A A Krasnyuk\textsuperscript{1,2}, P A Chibisov\textsuperscript{1}

\textsuperscript{1}Scientific Research Institute of System Development of the Russian Academy of Sciences (SRISA RAS), Nakhimovsky Ave. 36, 1, Moscow, 117218, Russia
\textsuperscript{2}National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Kashirskoe highway 31, Moscow, 115409, Russia

E-mail: amantonova@cs.niisi.ras.ru, ngrevcev@cs.niisi.ras.ru, chibisov@cs.niisi.ras.ru

Abstract. This paper presents our efficiency analysis of snoop filters for snoop based cache coherency protocols. The theoretical background shows possibilities of a performance increase for SMP system with a snoop filter; however, the false sharing problem can limit this advantage. That research requires for designing a snoopy bus-based SMP system contained a snoop filter.

1. Introduction

The theoretical description of many scientific areas, such as elementary particle physics, cosmology, astrophysics, gravitational physics, nuclear physics, and etc., require complex calculations enabled to test these theories, subsequently, to prove after it by experiment. Multithreading allows speeding up the calculations; this acceleration gives advantages for scientists. Symmetric multiprocessing (SMP) makes parallel multiple processes possible to organize that decreases the time spent on scientific estimates. SMP is a multiprocessor architecture with either two or more identical processor cores connected to a shared main memory (for example, like in figure 1); however, despite the significant advantages of the architecture, the development of SMP’s scalability has been limited by a cache coherence problem.

A coherence problem can arise if multiple cores have access to multiple copies of a datum in multiple caches. For snoop-coherence, bus-based SMP systems the ongoing problem is the need to reduce the number of snoop cache look-ups. Every transaction on the bus induces a look-up in the data cache of all remote CPUs in-order to check data consistency. The performance of multicore systems can be decreased because of a significant number of snoops miss in the caches because, in addition to the fundamental cache requests, snoop look-ups of snoopy based cache coherence protocols are dealt with a cache. Also, increasing of a number of snoops miss may lead to an overall increase in energy consumption, because these misses consume excessive energy [1, 2].

Broadcast-based snoopy hardware coherence protocols play an important role in small-scale multiprocessor system. However, as discussed above, snoop based cache coherence protocols inherently lead to extensive coherence traffic on the bus in a multi core system. All this traffic leads to tag lookups in remote data caches. On the other hand, recent research shows that these lookups and...
coherency traffic are, by a large extent, unnecessary. As an example of a way to solve the problem, snoop filters are required for reducing these requests.

![Symmetric Multiprocessing System Diagram](image)

**Figure 1.** Symmetric Multiprocessing System Diagram.

2. **Snoop Filters**

As stated above, a way to reduce number of snoops miss is to use snoop filters. A snoop filter is a small cache-like structure that is placed in-between the bus and the L2 backside of each processor, but provides inexact hit/miss information.

Snoop filters can be as either inclusive or exclusive. A superset of all the cached addresses is held by an inclusive filter and a subset of all addresses not currently cached is held by an exclusive filter. Moshovos et al. [1, 2] introduced the first set of widely recognized snoop filters named JETTY. Their filters, in general, have three variants: the exclude-JETTY, the include-JETTY, and the hybrid-JETTY. Alternatively, filters based Stream Registers (SR) [5, 8] were introduced into the IBM Blue Gene/P supercomputer. They are small, compared to the JETTYs inclusive filters. However they contain several disadvantages, such as wraparound detection. This snoop filter architecture is inclusive, and exploits stream registers to encode cache blocks, rather than table entries. Instead, the mask register represents a super-set of the offsets that have been accessed. Also there is the Counting Stream Register snoop filter architecture (CSR) [3, 6], which improves the performance of snoop filters based on stream registers. In addition to the base and mask fields, the counter field is appended to every stream register. The counter is increased upon every new data cache line load and decremented on evictions or replacements.

3. **Cache coherence problem**

3.1. **Cache coherence protocols**

Cache coherence protocol identifies the states in which a cache line can be relative to the analogous cache line in another core. It does not affect operations on other cache lines. The state of a cache line refers to the state of the corresponding cache control unit. The state of cache line is either basic or
transitional. There might be five possible valid states for MOESI protocol: Modified, Owned, Exclusive, Shared, Invalid. Figure 2 shows all possible state transitions from load/store requests from/to memory, which are performed by processing elements (the example is given for two cores). The transitions from one state to another in modern protocols occur non-atomic but by means of transitional states.

![Figure 2. Coherence protocol MOESI.](image)

The verification of cache coherence protocol is approved to be a bottle-neck of the multi-core CPU design process. Moreover, the exhaustive search is not applicable due to the combinatorial explosion problem. In the paper [10] a technique for scalable functional verification of cache coherence protocols based on the model checking technique is introduced. However, in this paper we propose the verification method of not only a cache protocol but also a memory subsystem with snoop-filter and other microarchitecture details as a whole.
We demonstrate the applicability of our verification method [18] extension as a performance measurement tool where using commercial multicore performance benchmarks are not required. The solution scheme deals with the adaptation of some available stochastic testing approaches to multicore testing tool. Directed stochastic test program generation is one of the most widely used approach to functional verification of microprocessor RTL-models [19]. This method has proven to be a high efficient and multifunctional [11].

3.2. The False Sharing problem
The proposed approach is based on the fact that data collision between threads in a shared cache line (the “False Sharing” problem) in multicore microprocessors with SMP can cause poor performance. The whole cache line will be updated even if only one byte is modified. In such a case two or more threads in parallel programs are assigned to modify variables which are different data elements in exactly the same cache line. Moreover, the cache line needs to be evicted and fetched at a time when each thread modifies different variables [4].

Several methods for the false sharing detecting and false cache line sharing effects analysis on multicore systems performance have evolved throughout the last years [12], [13]. To take advantage of multicore processors, it is required to avoid the described above data collisions. However, our experience shows that it is instructive to generate tests which have memory allocation scheme intentionally forced to have many false sharing contentions to verify cache coherency and communication between CPU’s cores. To reach functional coverage of cache coherence protocol it is

Figure 3. Memory card.
mandatory to create different scenarios of memory sharing dependencies including simultaneous
access to the same cache line by at least two cores.

The basic idea of the proposed shared memory scenario is to interleave memory by structures with
a size less than cache line size. In this case one core gains an access to all odd-numbered elements,
while another core – to all even-numbered elements (Figure 3). The great advantage of the proposed
memory allocation structure is that both cores gain access to the same cache-line simultaneously for
read and write. Despite this accessibility, the methodology avoids losses of data consistency due to
cores access to different bytes. Furthermore, each common area that is allowed to write to more than
one core will contain deterministic values at every time. Some memory regions are created for “read
only” and some – for “write only” – to create more realistic scenarios. The proposed approach aims to
detect failures in cache coherence protocol, memory subsystem and memory buffers. Also, this testing
system helps to allow for quantitative estimates about the impact of the snoop-filter in the multicore
processor design.

4. The experimental results
We analyzed snoop filters for snoop based cache coherency protocols to design a snoopy bus-based
SMP system [14, 15, 17] contained a snoop filter. The snoop filter builds on findings of predecessors,
and improvement tag look-ups is core parameter used for comparison of snoop filter's different
realization [9]. Total number of unnecessary tag look-ups with and without snoop filters was examined
for benchmarks, which is used for the simulation. Benchmark suites represent fragments of real
computing tasks from different areas. In this case, a set of tests (benchmarks) is chosen so that they
cover different architecture and micro-architecture aspects that affect performance as far as possible.
However, these tests are extremely time-consuming (billions of instructions for each test), since the
speed of program execution on the RTL model of the microprocessor is much less than on the real
platform.

Moreover, benchmark suites such as SPEC2001-OMP, PARSEC and SPLASH-2 show overall
performance on specific tasks, so it cannot be used for measuring the impact of the project micro-
changes to the performance. While random tests are generated according to a predefined template,
they can be directed to specific requested situations or scenario.

It would seem that the link must be established relations between tag look-ups of SMP system
included snoop filter and not included it. The total number of tag look-ups in SMP system is called all
tag look-ups and is denoted by \( N_{all} \), it’s equal in both the absence and the presence of a snoop filter.
Further, in case snoop filter exists, \( N_{miss}^{no} \) denotes unnecessary tag look-ups filtered through the snoop
filter, both \( N_{hit}^{yes} \) and \( N_{miss}^{yes} \) are necessary and unnecessary tag look-ups checking in cache,
respectively.

Obviously, summing tag look-ups \( N_{s}^{no} \), \( N_{hit}^{yes} \) and \( N_{miss}^{yes} \), we get
\[
N_{all} = N_{miss}^{no} + N_{hit}^{yes} + N_{miss}^{yes},
\]
(1)

It is readily seen that all unnecessary tag look-ups \( N_{miss}^{all} \) is the sum of both \( N_{miss}^{no} \) and \( N_{miss}^{yes} \) in
SMP system without a snoop filter. Finally, we obtain a formula of snoop filter efficiency for inclusive
snoop filters denoted by \( EF \), taking into account the progress made by [6, 16]:
\[
EF = \frac{N_{miss}^{no}}{N_{miss}^{all}}, \quad 100\%.
\]
(2)

This formula shows that Inclusive Snoop Filter Efficiency can be represented as the ratio between
filtered unnecessary tag look-ups \( N_{miss}^{no} \) and all unnecessary tag look-ups \( N_{miss}^{all} \), it is defined as percentages.
For the purpose of the experimental analysis, we compared filtering effectiveness of unnecessary tag lookups (hereinafter referred to as Filtering Percentage) using include-JETTY, SR and CSR filters. The total number of unnecessary tag look-ups with and without snoop filters was examined for each benchmark, which is used for the simulation. Figure 4 reports the filtering percentage achieved by three filters, here random tests are observed. Results from the study showed that the CSR filter is especially the need to consider because the results seem more efficient. We analyzed Inclusive Snoop Filter Efficiency considered separately for CSR filter $N^{no}_{CSR}$, it shows the significant filtered part of the unnecessary tag look-ups.

Also, we can identify the performance increasing (or decreasing) $PE$ as follows:

$$PE = \left( \sum_{i,j,k=1}^{l+i+j+k=N_{all}} N_{miss}^{no}(t_{L2i} - t_{SFi}) - N_{hit}^{yes}(t_{L2j} + t_{SFj})\right) - N_{miss}^{yes}(t_{L2k} + t_{SFk}) \frac{1}{N_{all}}$$

(3)

here $t_{L2}$ is L2 cache response latency and $t_{SF}$ is snoop filter response latency. When a snoop filter connected in parallel with the object CPU and L2 cache, we can measure differences between these parameters. After receiving of a tag lookup from the initiative CPU, if this tag lookup is unnecessary, then a snoop filter can respond faster than L2 cache (Figure 5).

![Figure 4. Directed Random test for JETTY, SR and CSR filters.](image)

![Figure 5. The number of responses over the number of CPU cycles indicated response time of both snoop filters and L2 cache using snoop filters in SMP.](image)
Clearly, that if SMP system is operating in the normal mode then we will have to increase of the performance, on the other hand, the false sharing can decrease a possible performance improvement. Having said this, it is presumed that program methods allow avoid it.

5. Conclusion
The proposed technique has been successfully applied to test RTL-model of dual-core microprocessor with SMP developed in SRISA. The discussed approach was initially considered to be the first stage of RTL-model testing, but the possibilities of the approach are also of interest for testing the model at the later stages of its design and functional maturity.

The testing process begins by creating random tests that check the MOESI coherence protocol. The new advanced random testing method based on the usage of proposed interleaved memory structures is developed to increase the probability of finding rare and hard to detect bugs in the memory subsystem.

The great advantage of the proposed memory allocation structure is that both cores gain access to the same cache-line simultaneously to read and write. Despite this accessibility the methodology avoids losses of data consistency due to cores access to different bytes. Furthermore, each common area that is allowed to write to more than one core will contain deterministic values at every time.

In conclusion, this work shows that the development of the snoop filter required for an efficient work of the SMP system. The CSR filter had higher efficiency ratings than those of other snoop filters; this configuration was preferable for our SMP system under development.

Acknowledgements
This work was supported by Competitiveness Program of National Research Nuclear University MEPhI.

References
[1] Moschovos A, Memik G, Falsafi B, and Choudhary A 2001 JETTY: Filtering Snoops for Reduced Energy Consumption in SMP Servers Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture 1 85-96
[2] Moschovos A, Memik G, Mittal G, Baniasadi A, and Choudhary A 2005 JETTY: Reducing Snoop-Induced Power Consumption in Small-Scale Bus-Based SMP Systems Center for Parallel and Distributed Computing 1-13
[3] Ranganathan A, Bayrak A G, Kluter T, and Brisk P 2012 Counting Stream Registers: An Efficient and Effective Snoop Filter Architecture International Conference on Embedded Computer Systems (SAMOS XII) 120-127
[4] Sorin D J, Hill M D, and Wood D A 2011 A Primer on Memory Consistency and Cache Coherence Synthesis Lectures on Computer Architecture 6 1-212
[5] Salapura V, Blumrich M A, and Gara A 2008 Design and implementation of the Blue Gene/P snoop filter High Performance Computer Architecture 5–14
[6] Ifsnes R 2013 Design of a Snoop Filter for Snoop Based Cache Coherency Protocols Norwegian University of Science and Technology
[7] Ekman M, Dahlgren F, and Stenstrom P 2002 Evaluation of snoop energy reduction techniques for chip-multiprocessors Workshop on Duplicating Deconstructing, and Debunking (WDDD-1) 1
[8] Salapura V Blumrich M A and Gara A 2008 Exploring the architecture of a stream register-based snoop filter Transactions on High-Performance Embedded Architectures and Compilers 89-118
[9] Antonova A M, Barskyh M E and Zubkovsky P S 2017 Snoop Filtering in Multicore Microprocessors Works of SRISA RAS 7 27-30
[10] Kamkin A S and Burenkov V S 2016 A method for scalable verification of PROMELA models of cache coherence protocols Models Problems of Advanced Micro- and Nanoelectronic
[11] Grevtsev N A, Khisambeev I Sh and Chibisov P A 2016 Methods to improve efficiency of microprocessor model stochastic tests. Problems of Advanced Micro- and Nanoelectronic Systems Development (MES) 2 8-15

[12] Liu T et al. 2014 PREDATOR: Predictive false sharing detection. PPoPP ACM SIGPLAN Notices 49 3-14

[13] Tongping L, Xu L 2016 Cheetah: detecting false sharing efficiently and effectively. Proceedings of the 2016 International Symposium on Code Generation and Optimization 1-11

[14] Aryashev  S I and Bychkov K S 2016 Optimizing the Prefetch Mechanism in the Secondary Cache Memory. Problems of Advanced Micro- and Nanoelectronic Systems Development (MES) 2 274-279

[15] Aryashev  S I, Bobkov S G, Zubkovsky P S, Morev S A and Rogatkin B Yu 2017 High-Performance Microprocessor 1890BM118 for Trusted Computing Systems. Problems of Advanced Micro- and Nanoelectronic Systems Development (MES) 3 345-352

[16] Antonova A M and Zubkovsky P S 2018 The Snoop Filter Development. Works of SRISA RAS 8 21-24

[17] Bogdanov A Y 2017 Experience in applying the Protium SOC prototyping platform for verification of microprocessors. Works of SRISA RAS 7 46-49

[18] Grevtsev N A and Chibisov P A 2018 A Practical Approach to Verification of Multicore Microprocessor Models. Problems of Advanced Micro- and Nanoelectronic Systems Development (MES) 2 52-58

[19] Smirnov A V and Chibisov P A 2018 Random test generator for multicore microprocessor cache coherence verification (ristretto). Problems of Advanced Micro- and Nanoelectronic Systems Development (MES) 2 31-38