Performance Evaluation of Heterogeneous Microprocessor Architectures

G. Dimova, M. Marinova, V. Lazarov

Abstract. *This paper focuses on the evaluation of the performance of heterogeneous microprocessor architectures using user and system perspective metrics. The tests have been conducted using a new simulation technique – Interval simulation, and a computer architecture simulator which is based on this technique. The evaluation is done by using multi-programmed multithreaded workloads constructed by using the benchmarks included in the two most popular free benchmark suites available – PARSEC, developed and maintained by Princeton University, and SPLASH2 – developed and maintained by The University of Stanford.*

I. Introduction

Simulators have been and will continue to be an effective, inexpensive and versatile way of proposing, evaluating and exploring new and existing computer architectures. With each new processor generation, designs are growing larger and more complex, and while we find ourselves increasingly dependent on simulation, it is a constant challenge to ensure that our simulations remain both fast and accurate. For a considerable amount of time now, there have been two prevailing simulation models which are focused on the two antipodes – performance and speed. These are the cycle accurate simulation and the functional simulation. The cycle accurate simulation is focused on delivering the most accurate result and is still a widely used method whenever accuracy is critical. Simulation probes are made in every instruction cycle which results in the increase of simulation time. Functional simulation, on the other hand, is focused on modeling the exact properties and operations that the system is able to perform. This provides for faster simulation but at the cost of decreased accuracy.

Numerous efforts have been made to reduce the amount of tradeoff in performance or, respectively, speed and hence numerous new simulation methods have been proposed. [1] Among others, two novel approaches to computer architecture simulation have been proposed which caught our attention – the One-IPC model and the Interval simulation.

What this paper is set to accomplish is an evaluation of modern computer architecture design approaches and specifically the evaluation of heterogeneous architecture models against one of the most wide-spread and popular design choices today – the Chip Multi-processors (CMPs).

The benchmark suites which have been chosen are the two most popular free for use benchmark suites available today which have been developed and maintained by respectable educational institutions such as the Princeton University and the University of Stanford.

II. Heterogeneous Architectures

Design of modern microprocessor architectures is currently moving in several main directions:

1. Multi-Core

Currently this design prevails over the other two, at least with respect to PCs, Mobile devices etc. which form a great part of the market. The problems that this architecture encounters are related to the lack of software to fully make use of the complex hardware.

2. Many-Core

These systems reach high levels of productivity whenever the degree of parallelism is high, as well. A good example of high degree of parallelism is the image processing.

3. Heterogeneous

A still emerging design approach is the combination of the former two, an approach that has come to be known as the heterogeneous design. It combines the performance of multi-core with the simplicity of many-core into a new diverse architecture.Heterogeneity (heterogeneousness) can be explored in different aspects. It can be expressed in the context of various computing elements with distinct Instruction Set Architectures (ISAs) cooperating to the completion of a common task. An example of such a case is the use of GPUs together with CPUs. In this case the computing elements themselves are homogeneous but the system, as a whole, is heterogeneous.

What this paper has set to explore is On-ChipHeterogeneity and more specifically Single-ISA Heterogeneous Systems. It has been demonstrated in the previous research [2] that such a design approach can achieve higher or similar productivity in a specific area compared to standard Chip Multi-Processors (CMPs). This is tightly dependent on the scheduling algorithm that has been chosen. It needs to reflect the requirements of every program in the current workload, as well as the shift in these requirements between different phases of execution of this program.

Featuring cores which emphasize on a specific aspect of productivity on the same chip allows the system to be more adaptable to the specific demands of the executing program. For example, a memory bound program could benefit

greatly from the increase of cache size and, supposing it does not offer a high degree of Instruction level parallelism (ILP), it would not benefit from Out-of-Order execution. This profile being described, an In-order-large-cache processor core would satisfy perfectly the execution demands of such a program.

An advantage of On-chip heterogeneousy, derived from the above, would be the more effective use of chip area for a given degree of instruction level parallelism [2].

III. Simulator Choice

Not long ago probably the most popular simulator choice would be SimpleScalar[3]. Unfortunately, its inability to simulate multi-core systems has made such a choice infeasible for our purposes. Moreover, SimpleScalar implements functional simulation while new simulation approaches which overcome to a greater degree the performance/speed tradeoff have recently been developed. We were determined to explore novel simulation techniques, hence our choices of a computer simulation approach and, therefore, of a simulator that employs this approach came down to Interval simulation.

1. Interval Simulation

Interval simulation is a recently proposed method for simulation of multi-core and many-core processor architectures. It is based on the mechanistic model for out-of-order processors, proposed by Eyermanand al [4].

Using interval simulation the process of performance analysis of the architecture in question can be significantly sped up. The interval model is based on the fact that the continuous flow of currently executing instructions is only interrupted by miss events. Each one of these events is characterized with a specific duration.

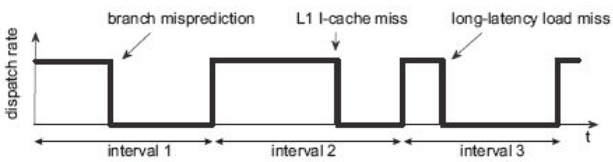


Figure 1

Types of miss events are: branch miss-prediction, L1 cache miss, long-latency load miss etc. Branch prediction, memory hierarchy, cache coherency and interconnection network define miss events. The characteristics of the former define the duration of the latter.

2. Sniper

Sniper is a multi-core architecture simulator based on the interval method which models the timing for the individual cores. The simulator maintains a 'window' of instructions for each simulated core. This window of instructions in reality corresponds to the reorder buffer of a superscalar out-of-order processor. Miss events, as described above, are determined with the assistance of the window and attention is paid to overlapping misses of different duration. Instructions are fed into the window tail. The progress of each core is determined by considering the instruction at the head of the window [5].

Sniper offers the following options which made it an adequate choice:

- Interval core model.
- CPI Stacks to gain insight into lost cycles.
- Parallel, multi-threaded simulator.
- Multi-program and multi-threaded application support, x86 and x86-64, SSE2.
- Validated against the Intel Core 2 microarchitecture (See the FAQ for details).
- Full DVFS support.
- Shared and private caches.
- Heterogeneous configuration support.
- Modern branch predictor.
- Supports parallel applications using p-threads, OpenMP, TBB, OpenCL.
- Runs SPLASH-2, Rodinia, SPEC OMP and most of PARSEC (See our integrated benchmarks quick-start guide).
- SimAPI and Python interfaces for monitoring and controlling the simulator's behavior at runtime.
- Single-option debugging of simulator or the application itself.
- Modern Linux-OS support (Redhat EL 5,6/Debian Lenny+/Ubuntu 10.04-12.04+/etc.).
- Open source software, licensed under the MIT and the Interval Academic License.
- Power modeling assisted by McPAT[6].

IV. Benchmarks

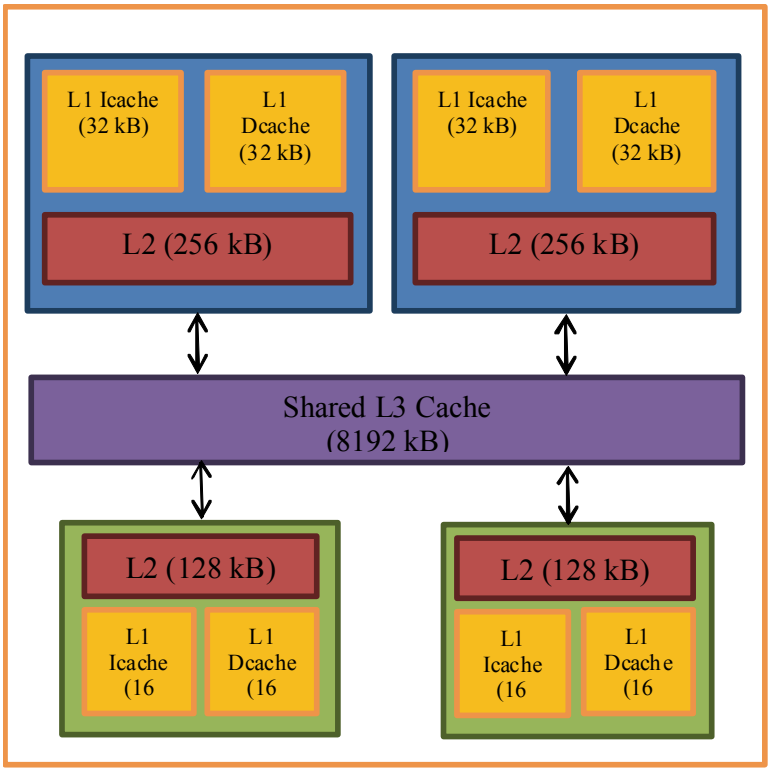
The benchmark suites that have been picked for the purpose of our experiments are The SPASH2 and the PARSEC benchmark suites which, combined together represent, as we believe, a significant part of the types of workloads that can be encountered in modern computer systems [7,8].

Both suites offer a variety of benchmarks which are mostly multithreaded and cover various areas of application. For example, the PARSEC benchmark suite features the c-anneal – a benchmark which simulates annealing, the fluid-animate – the fluid motion simulation, the ray-trace – the tracing a ray of light through a scene and more. The SPLASH2 benchmark suite features programs such as the ocean – the particle simulation, LU – the matrix factorization, Blacks-Scholes – the stock exchange analysis based on the Blacks-Scholes algorithm etc.

The workloads used in our experiments are comprised of all possible combinations of benchmarks from both suites. This has been done in order to gather enough data for a statistic estimation of the performance of the examined configurations since scheduling for heterogeneous systems is not implemented in this simulator.

V. Experimental Setup

Experimental results have been collected using the



Sniper Multi-Core simulator running on a 2-core Intel i3 processor with two more virtual cores, 8 Gb of RAM, 3 Mb LLC with Linux Ubuntu 12.04 installed.

All possible combinations of workloads including benchmarks from both suites have been constructed to avoid bias that might be presented by the lack of a scheduling policy.

A heterogeneous configuration has been built based on the balanced processor design suggested by Eyerman et al. executing multi-threaded in a multi-programmed environment and CPI_{SP} is the CPI application j when executing single-threaded in an uniprogrammed environment. ANTT is a lower-is-better type of metric.

VI. Simulation Results and Evaluation

1. Cache Test

The first experimental setup is focused on the cache size variance. The Cache related parameters of the configuration were modified as follows:

Table 1

Heterogeneous base (het-base)	
L1 I-cache L1 D-cache L2 Cache L3 Cache	16,16,32,32 16,16,32,32 128,128,256,256 8192
Heterogeneous 64k L1 (het-64k)	
L1 I-cache L1 D-cache L2 Cache L3 Cache	32,32,32,32 32,32,32,32 256,256,256,256 8192
Heterogeneous 512k L2 (het-512k)	
L1 I-cache L1 D-cache L2 Cache L3 Cache	32,32,32,32 32,32,32,32 512,512,256,256 8192
Homogeneous (gen)	
L1 I-cache L1 D-cache L2 Cache L3 Cache	32 32 256 8192

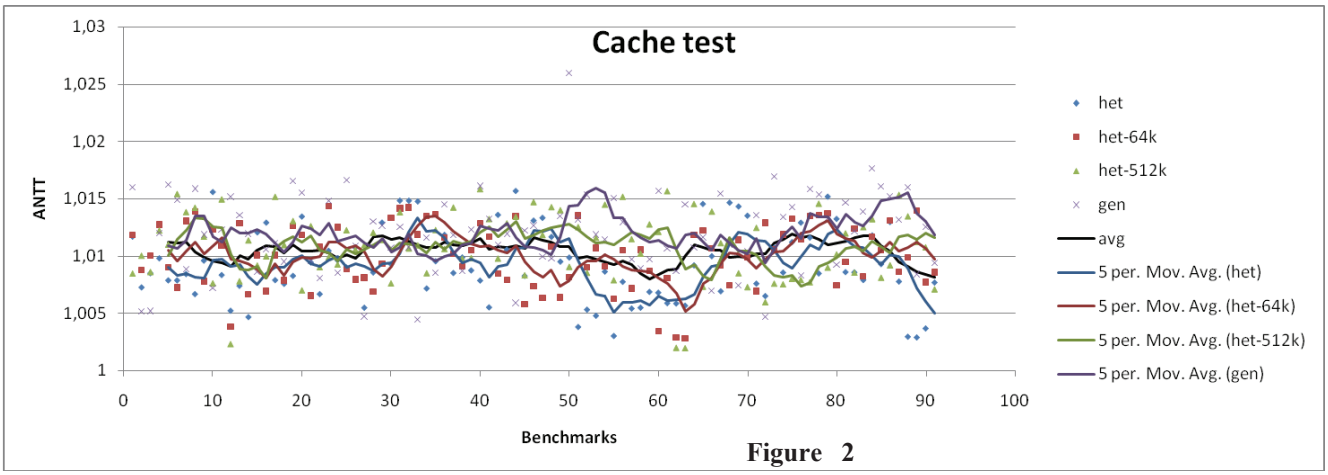


Figure 2

The experimental results are presented below. Trend-lines represent the moving average for each data series. The series ***avg*** represents the average of all trend-line and acts as a base for comparison.

One can easily see that the greater part of the moving average for the homogeneous configuration is found above the absolute average for all series, i.e. the parallel execution overhead using this configuration and the above mentioned workloads is to a degree greater than the overhead incurred when executing these workloads using the other configurations. Contrary to this trend is the trend observed in the ***het-base*** series. It is mostly concentrated below the absolute average. It is difficult to compare the other two configurations since their observed performance is similar to a great extent. We conclude that for most workloads of the base heterogeneous configuration offers the best or close to the best performance which makes it the configuration of choice.

2. Width Test

The second experimental setup is focused on the "issue width" variance. The Issue width related parameters of the configuration were modified as follows:

Among these experimental results there is a definite trend visible even to the naked eye. In almost all of the tests the heterogeneous configurations with wider issue than the base heterogeneous configuration exhibit worse performance. The 6-wide issue configuration is close to the absolute average though. Both configurations keep the same ROB size as the base heterogeneous configuration. Although the beta-quadritic relationship between issue width and ROB size for a balanced processor design [4] has been sustained in the 4-wide configuration, it manifests the greatest slowdown. An explanation of this might be sought in the assumption that the ROB becomes occupied with waiting instructions i.e. the ROB size must be increased even more.

Optimal results in this experiment are observed in the tests with the base heterogeneous configuration and the homogeneous configuration with the base heterogeneous configuration exhibiting the lowest ANTT and thus best performance.

3. ROB test

The third experimental setup is focused on the ROB

Table 2

Base heterogeneous (het-base)	
Dispatch width Miss-predict penalty ROB size	2, 2,4, 4 14, 14, 17, 17 64,64,128,128
Heterogeneous 4 wide issue (het-4-wide)	
Dispatch width Miss-predict penalty ROB size	4, 4,4, 4 14, 14, 14, 14 64,64,128,128
Heterogeneous 6 wide issue (het-4-wide)	
Dispatch width Miss-predict penalty ROB size	6, 6,4, 4 10, 10, 14, 14 64,64,128,128
Homogeneous (gen)	
Dispatch width Miss-predict penalty ROB size	4 17 128

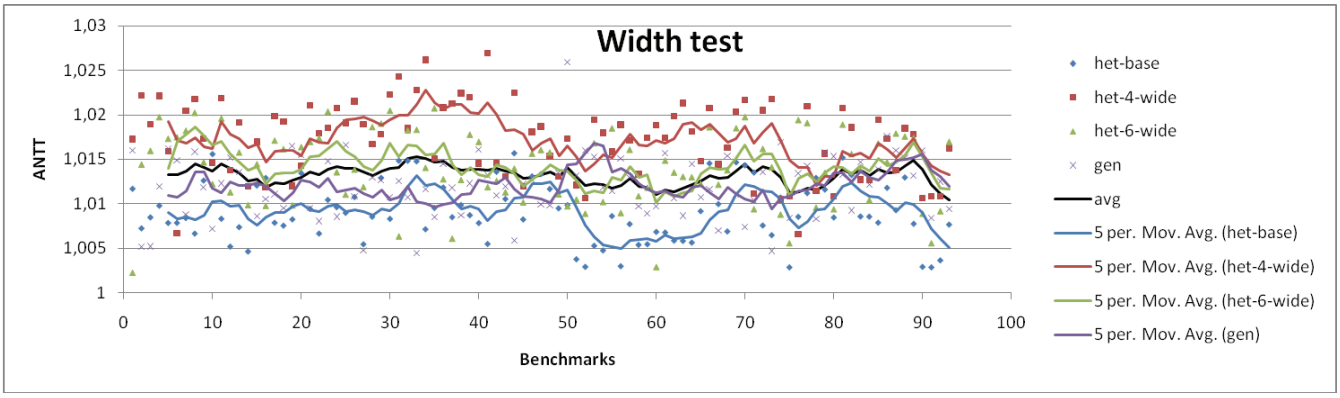


Figure 3

size variance. The ROB size related parameters of the configuration were modified as follows:

Table 3

Base heterogeneous (het-base)	
Dispatch width Miss-predict penalty ROB size	2, 2,4, 4 14, 14, 17, 17 64,64,128,128
Heterogeneous ROB size 128 entries (het-rob-128)	
Dispatch width Miss-predict penalty ROB size	2, 2,4, 4 14, 14, 17, 17 128,128,128,128
Heterogeneous ROB size 256 entries (het-rob-256) Dispatch width	2, 2,4, 4
Miss-predict penalty ROB size Homogeneous (gen)	14, 14, 17, 17 128,128, 256,256
Dispatch width Miss-predict penalty ROB size	4 17 128

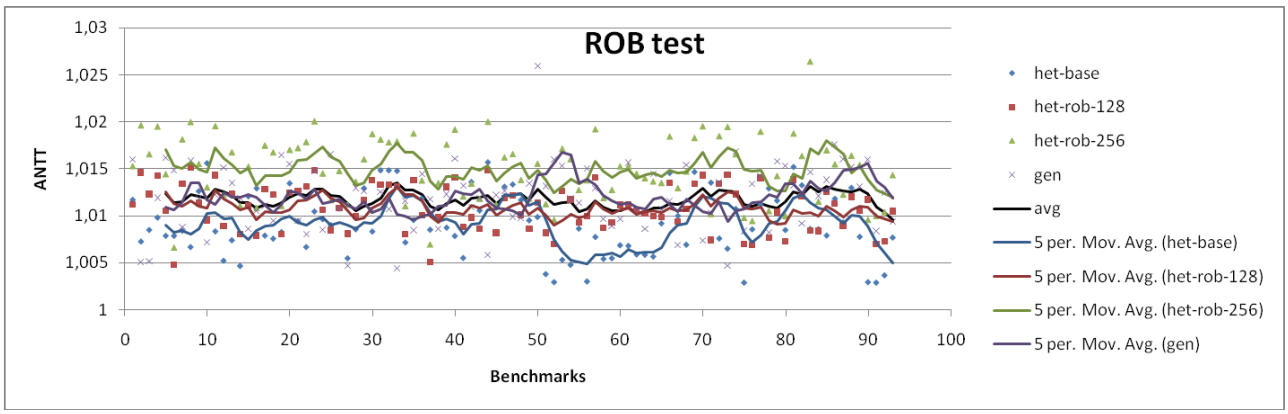


Figure 4

In this scenario the heterogeneous configuration with ROB size 128 and the homogeneous configuration display very similar results close to the absolute average. This is expected since both configurations employ equally sized ROBs. The heterogeneous configuration with the largest ROB – 256, is the slowest with respect to performance. This might be explained with the ROB being a memory structure – a too large size inevitably results in performance degradation. The base heterogeneous configuration again manifests best performance for a prevailing amount of workloads.

This experiment clearly shows the well-known truth that larger memory structures do not guarantee better performance as the microprocessor design requires subtle balance between all components.

VII. Conclusion

In all three experiments that have been conducted, the base heterogeneous configuration which has been built

using the principles proposed in [4] for a balanced processor design manifested optimal or close to the optimal performance. This paper strived to demonstrate that the heterogeneous microprocessor architectures are a feasible design and are able to provide adequate performance for different types of workloads. Therefore, heterogeneous design approach can be considered a reasonable alternative to homogeneous. It is worth to mention that whenever greater performance per Watt is necessary the heterogeneous design is a clear winner since simpler cores yield smaller die size, at the very least, and overall lower energy consumption [10].

Acknowledgement

This paper is supported by the NSF Conract № DCVP 02/1.

References

1. Sherwood, T. and J. J. Yi. Computer Architecture Simulation and Modeling. – *IEEE Micro,* August 2006.

2. Kumar, R. D., M. Tullsen, P. Ranganathan, N. P. Jouppi and K. I. Farkas. Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance. 31st International Symposium on Computer Architecture, 2004.

3. Austin, T., E. Larson and D. Ernst. SimpleScalar: An Infrastructure for Computer System Modeling. – *IEEE Micro,* 2002.

4. Eyerman, S., L. Eeckhout, T. Karkhanis and J. E. Smith. Processors, A Mechanistic Performance Model for Superscalar Out-of-Order. – *ACM Transactions on Computer Systems,* 2009.

5. Carlson, T. E., W. Heirman and L. Eeckhout. Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-Core Simulation, 2011.

6. Li, S., J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen and N. P. Jouppi. McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures, 2009.

7. Bienia, C., S. Kumar, J. P. Singh and K. Li. The PARSEC Benchmark Suite: Characterization and Architectural Implications, 2008.

8. Woo, S. C., M. Ohara, E. Torrie, J. P. Singh and A. Gupta. The SPLASH-2 Programs: Characterization. 22nd Annual International Symposium on Computer Architecture, 1995.

9. Eyerman, S. and L. Eeckhout. System-Level Performance Metrics For Multiprogram Workloads. – *IEEE Micro Magazine,* 2008.

10. Morad, T. Y., U. C. Weiser, A. Kolodny, M. Valero and E. Ayguade. Performance, Power Efficiency and Scalability of Asymmetric Cluster Chip Multiprocessors, 2005.

11. Lazarov, V., M. Marinova. Dependencies Evaluation in Superscalar Processors. ComSysTech'04, 17-19 June, 2004, Ruse, Proceedings, I.3-1, I.3-4.

12. Marinova, M., V. Petkova, V. Lazarov. Cache Performance Relations with Traces' Properties. IEEE Computer Society, Bulgarian Section, IIIrd International Scientific Conference Computer Science 2006, Proceedings Part II, Istanbul, 12–16 October 2006, 66-71.

Contacts:
Prof. ***Vladimir Lazarov***
e-mail: lazarov@bas.bg

Manuscript reseived on 11.5.2012