Proceedings of the 2013 Federated Conference on
Computer Science and Information Systems pp. 751–758

Hands-On Exercises to Support Computer Architecture Students Using EDUCache Simulator

Sasko Ristov, Blagoj Atanasovski, Marjan Gusev, and Nenad Anchev
Ss. Cyril and Methodius University,
Faculty of Information Sciences and Computer Engineering,
Rugjer Boshkovikj 16, PO Box 393,
1000 Skopje, Macedonia
Email:sashko.ristov@finki.ukim.mk, blagoj.atanasovski@gmail.com, marjan.gushev@finki.ukim.mk,
nenad_ancev@hotmail.com

***Abstract*—EDUCache simulator [1] is developed as a learning tool for undergraduate students enrolled the Computer Architecture and Organization course. It gives the explanations and details of the processor and exploitation of its cache memory. This paper shows a set of laboratory exercises and several case studies with examples on how to use the EDUCache simulator in the learning process. These hands-on laboratory exercises can be also used in learning software performance engineering and to increase the student willingness to learn more hardware based courses in their further studying.**

***Index Terms*—Education; HPC; CPU Cache; Multiprocessor.**

I. Introduction

THE Computer Architecture and Organization course is devoted to help the students to understand how the computers work. This course is usually in the first study year and the teaching material is almost always totally new for the students. Computer architecture is acknowledged as a significant part of the body of knowledge and an important area in undergraduate computer science curricula [2], [3]. Learning the course requires huge efforts by the students, especially in case of computer science students. Instead of wanting to know how the hardware (computer) works, they just want to use it as a necessary tool to execute their software programs. While developing programs by using some high-level programming language, the students do not get a clear picture of how they are executed by the computer. This decreases the students' interest and deeper understanding in learning of the Computer Architecture and Organization course. Therefore, it makes the teaching even more difficult requiring a lot of effort from both instructors and students [4]. Teachers must not only cover a body of knowledge, but they must motivate students and make the course exciting by selecting appropriate topics, such as which processor should be learned [5].

Today's modern multi-processors consist of multilayer cache memory system [6] to speedup data access balancing the gap between CPU and main memory. This complicates the learning process even more since the students must learn the organization inside the multi-processor, and not only the architecture. We have developed EDUCache simulator [1] that visually presents cache hits and misses, cache line fulfillment, cache associativity problem [7], for both sequential and parallel algorithm execution. In this paper we present several hands-on exercises for EDUCache simulator that will improve the teaching and alleviate the students' learning process. Several predefined examples for special memory patterns that cover data locality and cache set associativity are also presented.

The rest of the paper is organized as follows. In Section II we discuss the related work about improving the teaching and learning of computer architecture and other hardware courses for computer science students. Section III briefly describes the Computer Architecture and Organization course. The EDUCache architecture, user interface and different working modes are described in Section IV. The newly proposed hands-on laboratory exercises and some predefined examples are presented in Section V. The final Section VI is devoted on conclusion and future work.

II. Related Work

This section presents the existing similar visual simulators that cover the area of computer architecture and organization. We also present the proposed methodologies and laboratory exercises in order to lighten the learning and teaching of the course.

A. Hands-on Exercises: Simulation or Real Hardware

Introducing appropriate hands-on exercises, homework assignments and projects besides the lectures will make the course more interesting and will provoke the students to dive more deeply to learn how the computer works. Liang [8] performed a nice survey of hands-on assignments and projects.

Two approaches exist in organizing the laboratory exercises, i.e., either with visual simulators or working on real hardware. Using appropriate visual simulators on hands-on exercises lightens the teaching process and can significantly improve the students' interest in hardware generally [9], [10]. The simulators or web online tools share the laboratory equipment, thus removing the obstacles of cost, time-inefficient use of facilities, inadequate technical support and limited access to design and laboratory resources [11].

Practical work on real hardware is also very important [12], [13], [14]. Wang [15] and Lee [16] proposed FPGA-based configurable processors to be used in laboratory exercises.

978-1-4673-4471-5/$25.00

The students can develop, implement and monitoring both hardware and software of multi-core processor systems on real hardware.

B. Teaching Methodologies

Several methodologies are developed to teach the hardware courses in general for computer science students. Reinbrecht et al. [17] present a methodology that integrates functionally verified ASIC (Application-Specific Integrated Circuit) soft cores into a FPGA in order to allow the students to learn the fundamentals of hardware and its designs challenges, not only development, but also verification and physical implications.

Ackovska and Ristov [18] improved the hands-on laboratory exercises and introduced a new teaching methodology. They divided the exercises in tutorials and tasks. The former are published a week before the exercises as students can prepare for the exercise, while the latter are given to the students during the laboratory exercises. These changes improved students' grades without making the course exams easier. Hatfield and Jin [19] designed and developed many laboratory exercises to design and implementation of an operating model of a pipelined processor.

Da [20] elaborated the common methods of classroom teaching and experimental teaching and some efficient methods for classroom and laboratory teaching. He [21] addressed five problems related to computer architecture education in multi-core era and suggested a possible solution.

C. Visual Simulators

We have found many visual simulators that cover a particular fundamental part of computer architecture and organization. Nikolic et al. [9] evaluated many simulators and concluded that some simulators are designed for teaching, some for data profiling, and none of them covers all topics in computer architecture and organization. We [1] have presented a very comprehensive overview of several visual simulators:

- EduMIPS64 [22] for instruction pipelining, hazard detection and resolution, exception handling, interrupts, and memory hierarchies;
- Dinero IV [23] for memory hierarchy with various caches on single core systems;
- CMP$im [24] - based on the Pin binary instrumentation tool;
- HC-Sim [25] generates traces during runtime and simulates multiple cache configurations in one run;
- Herruzo's [26] simulator for understanding the cache look up process and writing elements in the cache memory.
- Misev's [27] visual simulator for ILP dynamic out-of-order executions;
- Valgrind [28] (with its module Cachegrind) profiler for cache behavior;

All these simulators were not primarily developed for teaching the cache memory although most of them are visual. They lack educational features since they are built to complete the simulation as fast as possible rather than to present the architecture and organization of cache memory system in a modern multi-processor. Our EDUCache simulator [1] offers step by step simulation allowing the students to pause the simulation and analyze the cache hits and misses in each cache level. Its power increases with appropriate hands-on exercises.

SimpleScalar [29] and SMPCache [30] are additional simulators selected by William Stallings as a simulation tool for the implementation of student projects [31].

SimpleScalar is used for program performance analysis, detailed microarchitectural modeling, and hardware-software co-verification. It supports non-blocking caches, speculative execution, and branch prediction.

SMPCache is a widely used simulator in more than 100 universities and research centers. It is a trace-driven simulator for the analysis and teaching of cache memory systems on symmetric multiprocessors, analyzing program locality; influence of the number of processors, cache coherence protocols, schemes for bus arbitration, mapping, replacement policies, cache size (blocks in cache), number of cache sets (for set associative caches), number of words by block (memory block size).

Although SMPCache is primary developed as a learning tool for the Computer Architecture and Organization course, it should be redeveloped since it has an option for one cache level and symmetric processor only, while our EDUCache simulator offers simulation of heterogeneous multiprocessor with three cache levels. Our hands-on laboratory exercises proposed in this paper supplements its value.

III. The Computer Architecture and Organization Course

This section briefly describes the Computer Architecture and Organization Course.

The course's main objective is to offer the students a clear understanding of the main computer architectures, performance of the computer parts and the whole computer system. It also covers the topics of today's modern multi-chip and multicore multiprocessors, as well as the digital logic circuits.

A. Course Organization

The teaching of the course is organized in three parts: theoretical lectures with 2 classes per week, theoretical exercises with 2 classes per week and practical exercises with 1 class per week in laboratory. Lectures and theoretical exercises are organized in larger groups of around 100 students, while practical exercises are carried out in computer laboratories in groups of up to 20 students, with each student working on its own workstation. Prerequisites for enrolling in the course are previously completed course in Discrete Mathematics.

Theoretical lectures cover the computer abstractions and technology, the computer language (MIPS), computer arithmetic, the processor, memory, storage, and multichip multicore multiprocessors [32].

Theoretical exercises are divided in two parts. The first midterm covers the topics: computer arithmetic, codes and performance parameters, while the second part deals with

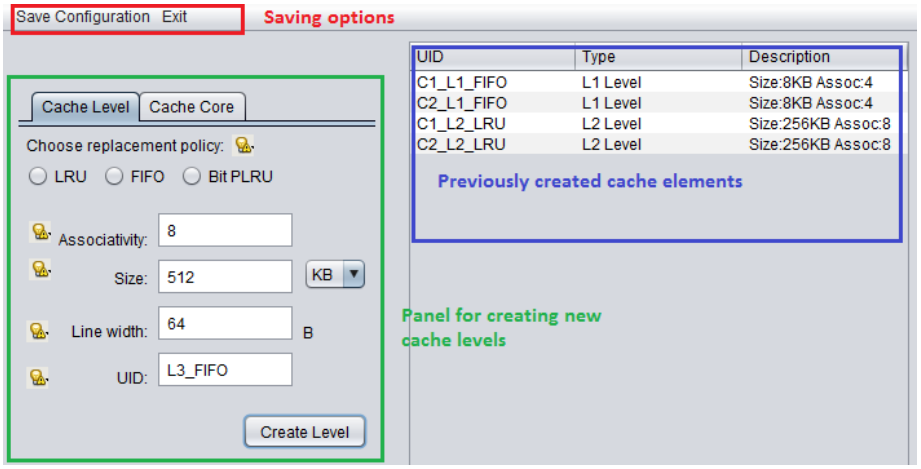

Fig. 1: Overview of design mode of EDUCache simulator - Creating L3 cache instance

digital logic. Hands-on laboratory exercises follow the topics of theoretical exercises.

The course can be passed in two ways, i.e., through midterms or final exam. The students must take the theoretical lecture part and exercise part (plus logic circuits) in either way.

B. Course Obstacles

The previous section briefly describes the course organization. We have analyzed the student results and determined that they had more problems with the topics of theoretical lectures compared to the exercises, and more precisely, the material of the second midterm, i.e., the processor, memory, I/O and parallelization. Our analysis show that although these subjects are covered during the theoretical lectures, neither theoretical nor practical exercises are provided for these topics, since the exercises cover to the design of logic circuits. Even more, IEEE Computer Society and ACM stated that more attention should be given to the multi-core processors architecture and organization, instead of the logic design level [33].

Therefore, we developed the EDUCache simulator that covers these topics. In this paper, we present the hands-on laboratory exercises that will make it even more appropriate in the teaching process, mainly focused on multiprocessor, cache and main memory.

IV. EDUCache Simulator

This section briefly describes the main features, interfaces and user interface of the EDUCache simulator. More details about the EDUCache simulator are presented in [1].

The EDUCache simulator is a platform independent simulator developed in JAVA whoose main simulation is described by a set of Java classes, each for a different CPU cache parameter. It allows the students to design a multi-layer cache system with different multi-core multi-cache hierarchy and to analyze sequential and parallel execution of user algorithm. Each chip can have one or several homogeneous cores. Each core has access to some cache of different cache level (generally L1 to L3). Particular cache can be owned by one, several or all cores of the chip. In general, L1 and L2 caches are private per core in modern multi-processors, while L3 cache is shared among several or all cores.

The particular cache level parameters can vary. Cache is determined by cache memory size, cache line size, cache associativity and replacement policy. EDUCache simulator allows the students to configure all these cache parameters and cache levels.

A. User Interface

The EDUCache simulator user interface is visual and user friendly. It uses the Multiple Document Inteface (MDI) paradigm. The EDUCache simulator works in two modes: *Design* and *Simulation*.

1) Design Mode: The students can configure various cache parameters and levels to create instances of cache levels and share them among chip cores.

The students create the cache levels with unique ID (UID). Figure 1 depicts an overview of EDUCache simulator user interface in design mode and an example on how a student can very easily create a particular cache level instance, select a replacement policy, set associativity, cache line size, cache size and select unique cache level UID.

After creating cache level instances, the students can create a core, selecting cache instances from the list of previously created ones (visible in the table in the right frame) for each cache level. Figure 2 depicts a user interface to create a core with unique Core UID, using previously created cache level instances.

Finally, the students can save the created configuration that represents a CPU chip. They configure the core instances, which they prefer to include on the chip and they are prompted where to save the configuration file.

After completing a multi-core chip with different caches in the design mode, the students can move to the simulation mode in order to simulate some memory accesses and analyze which of them will generate hit or miss in particular cache level of particular core.

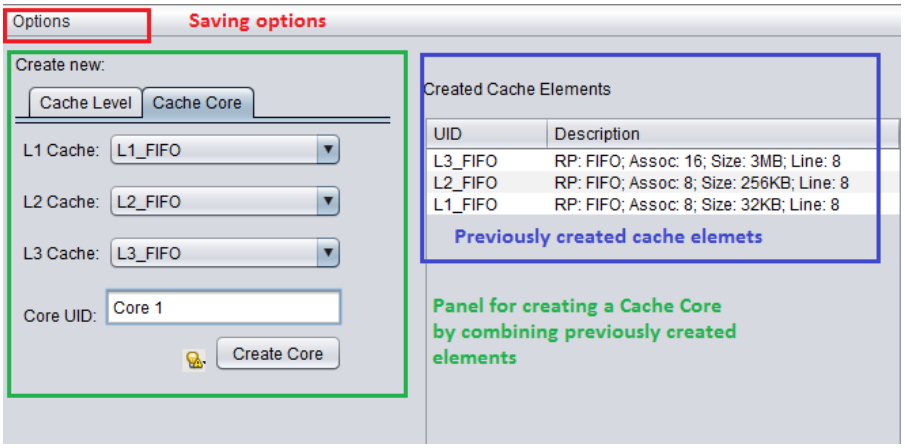

Fig. 2: Overview of design mode of EDUCache simulator - Creating a CPU core

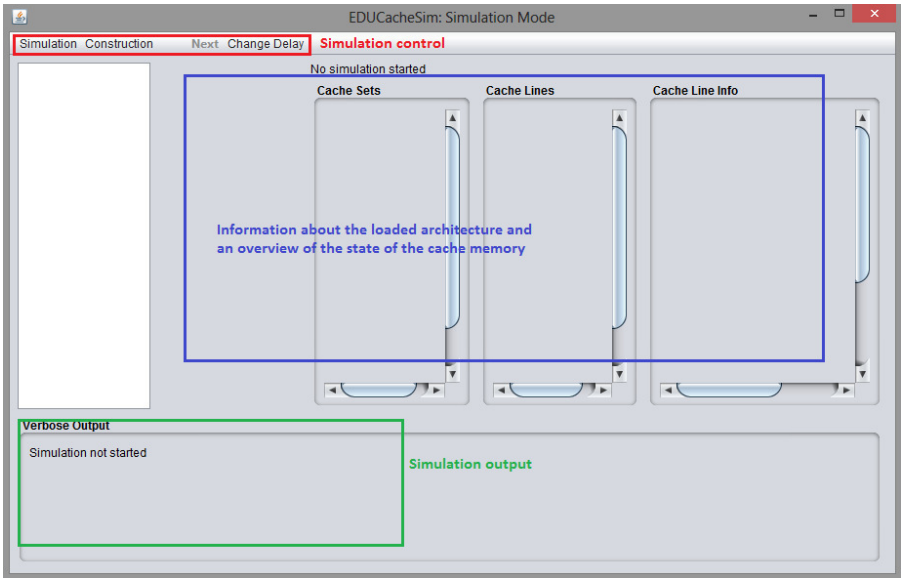

Fig. 3: Overview of Simulation mode - hit in L1 level of core C2, set #8, line #1, address 123416

2) Simulation Mode: After a configuration of the CPU chip with multiple cores per chip and multiple cache levels per each core, the students should load the memory addresses and run the simulation.

Figure 3 depicts the Simulation mode. Its main window consists of:

- *Simulation Control Menu Bar* - is the central control hub for the simulation process. It contains 2 menus, i.e., *Simulation* and *Construction*. The latter creates new or loads the existing configuration file for a core. The former loads a study case file, trace file, and operates the simulations (start, pause, stop, or step by step working mode);
- *Loaded Address Trace Frame* - shows the contents of the trace file, i.e., which core should read the address and the physical address that is loaded.
- *Verbose Output Frame* - shows the addresses that are read by cores, the search in L1 cache and selecting the set in which the address is supposed to map, the result, i.e. cache hit or miss, the cache line number if it is hit and the evicted line if the chosen set was full and read miss is generated; and
- *Visual Representation Frame* - is the main feature of simulation mode which gives a visual representation to the lookup process. It represents different levels of the cache level architecture: Core Pane, Cache Sets Pane, Cache Lines Pane, and Cache Line Info Pane.

V. Hands-on Exercises for EDUCache Simulator

This section presents how the EduCache simulator can be used as a tool in the laboratory exercises to introduce the students with the basic concepts of processor and its cache memory.

A. General Terms

All hands-on laboratory exercises follow the same concept. Each hands-on exercise starts with an explanation of the goal and objectives. Next, a brief coverage of the required topics is presented. This will remind the student of the topics that need to be learned or revised in order to be able to prepare for the exercise and complete it. The exercises should be given to the students a week before the exercise [18].

The objectives are step by step guides on what the student is supposed to do, e.g., configure the simulator, create a certain architecture, execute a simulation or analyze the results from an executed simulation. The guidelines are posed as simple instructions or as learning objectives. Questions are placed between the guidelines alerting the students which areas require more attention. In the end, the students must answer all questions, create the configuration file of the simulator and the simulation result file.

B. The Hands-on Exercises

This section presents the hands-on laboratory exercises for the EDUCache simulator. We present several exercises, some of which can be gathered or divided according to the available time for the hands-on laboratory exercises.

1) Exercise 1: Intro to EDUCache Environment: The first laboratory exercise is designed to introduce the EDUCache simulator to the students. The exercise goes over the different types of files that the simulator uses. The basic commands require the students to go through a simulation and to analyze the results. Learning objectives include: EDUCache design and simulating modes, loading a cache configuration file, and basic cache memory elements. The exercise concludes with running a simulation on a loaded trace file, creating a new trace file and running the simulation again, finishing with an analysis of the statistics that the simulator presents after the simulation is finished.

Although this exercise does not require a lot of students' effort, it should be graded. Otherwise, the students may not pay enough attention on learning the elementary controls of the simulator, which will cause them to have trouble with later exercises.

2) Exercise 2: Different Cache Parameters: The second laboratory exercise aims to present the basic parameters of cache memory to the students. That is, the size, associativity and the principles of multiple cache levels. The learning objective of this exercise is for the students to understand how these parameters impact on specific program execution. The principles of time and data locality are covered. The simulator is used to create multiple configurations with different parameters regarding to the cache size. Simulation is realized on a single memory trace. The students must observe into the results of the simulation and compare to find how the different sizes effect the program execution.

The final part of the exercise is to determine the smallest size for a cache level that gives the same performance as an infinite cache size. The grading should include optional questions for extra credit to inspire the students to show interest in the exercises since this exercise contains a significant number of tasks (and objectives) that the students must complete.

TABLE I: Example 1 that generates cache misses

Parameter	L1	L2	L3
Size	32B	64B	128B
Associativity	2	2	4
Replacement	FIFO	FIFO	FIFO
Cache line	8B	8B	8B

TABLE II: Results of the simulation of the Example 1

Parameter	L1	L2	L3
Total reads	50	50	50
Cache hits	0	0	0
Cache misses	50	50	50

3) Exercise 3: Overview of Cache Set Associativity and Replacement Policies: The third laboratory exercise goes over the concepts of cache set associativity and replacement policies as one of the more complex cache memory parameters. The exercise shows the impact of these parameters on a specific program execution. The students must create configurations and use them to execute and analyze multiple simulations.

A set of address traces is given and the students' task is to observe and conclude the optimal replacement policy for each address trace. The exercise also offers the possibility to create experimental configurations which do not usually appear in real systems, such as certain cache levels with certain replacement policy. Another set of objectives takes a look at the influence of the set associativity.

C. The Demo Examples

In this section we present several demo examples for characteristic memory access patterns in order to lighten the learning and understanding of the processor and its cache memory architecture and organization.

1) Example 1: Cache Miss due to "Loosely" Data: This example demonstrates the continuous cache misses for the loosely data. Table I presents the example of cache parameters. The trace file forces the access of the elements with 8B offset since we want to force a cache miss for each memory read (each read accesses the element of different cache line). Total 50 reads are realized and the results of the simulation are presented in Table II.

2) Example 2: Each Second Access is Cache Hit due to Data Locality: This example accesses pairs of elements such that each pair is placed in the unique cache line. The cache parameters are presented in Table III. Total 10 reads are realized. The results of the simulation are presented in Table IV. That is, we forced 5 pairs of miss and hit in L1 cache.

3) Example 3: Always Cache Hit due to Data Locality: This example demonstrates how the set associative cache memory generates cache hits for "tightly" data (data locality) of a single cache line. The cache parameters are the same

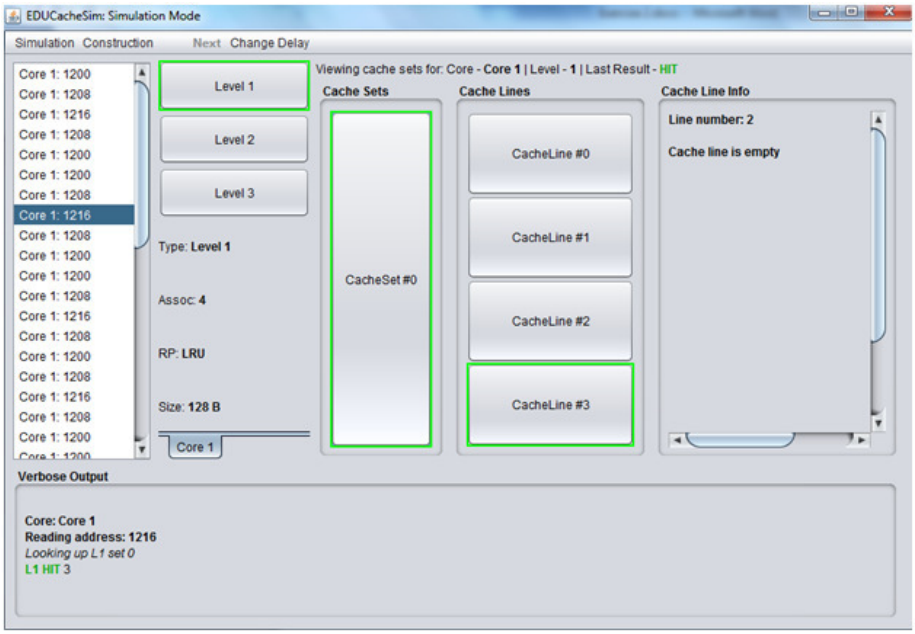

Fig. 4: Simulation of Example 3

TABLE III: Example 2 that generates cache hits for the second access

Parameter	L1	L2	L3
Size	128B	192B	256B
Associativity	4	4	8
Replacement	LRU	LRU	LRU
Cache line	32B	32B	32B

TABLE IV: Results of the simulation of the Example 2

Parameter	L1	L2	L3
Total reads	10	5	5
Cache hits	5	0	0
Cache misses	5	5	5

as the Example 2 presented in Table III. Since we want to generate a cache hit for each memory access, all addresses in the memory trace are in the range of a single cache line. Total 42 reads are realized. The results of the simulation are presented in Table V. That is, 1 cache miss is generated by the first access, and 41 cache hits by all others.

Figure 4 depicts a hit occurring on L1 cache always in the same cache line, as the rightmost pane shows the other cache

TABLE V: Results of the simulation of the Example 3

Parameter	L1	L2	L3
Total reads	42	1	1
Cache hits	41	0	0
Cache misses	1	1	1

TABLE VI: Configuration for Example 4

Parameter	L1	L2	L3
Size	16B	32B	64B
Associativity	2	2	4
Replacement	FIFO	FIFO	FIFO
Cache line	8B	8B	8B

lines are empty because all the required elements have been loaded into a single cache line Line #3.

4) Example 4: Cache Associativity Problem: This example demonstrates how the set associative cache memory can generate continuous cache misses if the data access are always in the same cache set, i.e., cache associativity problem [7]. The cache parameters are presented in Table VI. Total of 15 reads are realized, but only 3 different addresses are accessed. The results of the simulation are presented in Table VII.

This example results in constant L1 cache misses because of constant eviction of cache lines in the same set. Because we want to generate a cache miss by looking up the same cache set (in our case $CacheSet\#0$) for each memory read, all addresses in the memory trace must satisfy $Block\_address = X \cdot Number\_of\_cache\_sets$.

For our configuration, the cache line is 4 bytes, which yields that if the address in main memory is N bytes long, the block address will be the first $N - 2$ bits [6].

Figure 5 depicts a step in the simulation of this exercise. Reading the element stored in address 0 generates a cache miss on the Level 1 cache, because the previous read replaced it from the cache set.

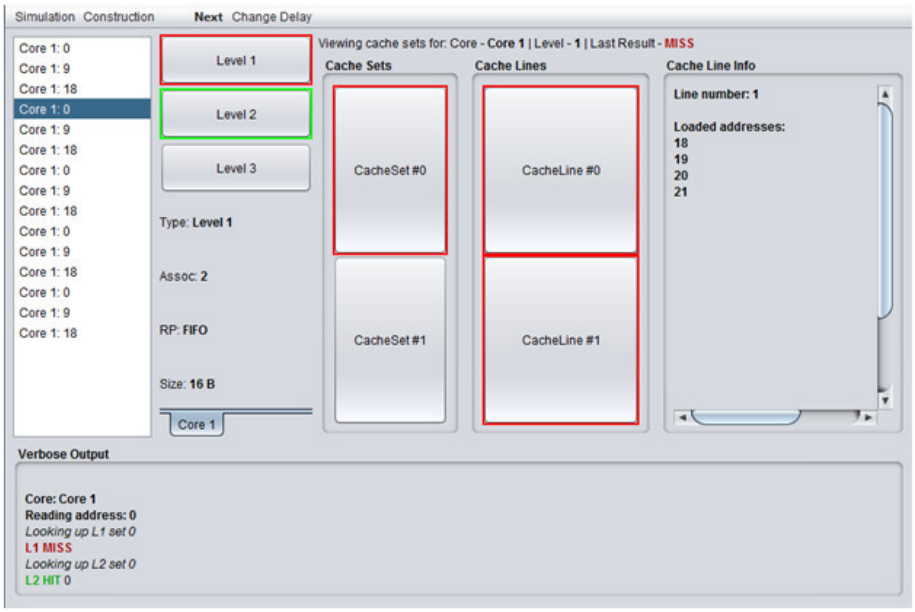

Fig. 5: Simulation of Example 4

TABLE VII: Results of the simulation of the Example 4

Parameter	L1	L2	L3
Total reads	15	15	3
Cache hits	0	12	0
Cache misses	15	3	3

Reading the element will generate cache hit in the Level 2. The rightmost panel shows the loaded addresses in $CacheLine\#0$ in $CacheSet\#0$.

VI. Conclusion and Future Work

EDUCache visual simulator offers the students a tool to design their own CPU core with multi level cache memories. It simulates cache misses and hits in particular cache set and memory location for sequential and parallel execution of an algorithm. The students can interactively learn about the cache hierarchy, architecture and organization of private cache level per core or shared cache level among all or a group of cores, the cache capacity and associativity problem, cache line, cache replacement policy, data locality etc.

This paper presents several hands-on laboratory exercises to support the students for the Computer Architecture and Organization course, i.e., using the EDUCache simulator they will better understand the architecture and organization of the modern processor and its cache memory. Several predefined examples are also presented to lighten the learning process and increase the students' willingness for the Computer Architecture and Organization course. This will help the students to develop their algorithms to achieve maximum performance using the same hardware resources.

We will introduce the EDUCache simulator and the hands-on exercises to this semester in courses Computer Architecture and Organization and Parallel and Distributed Processing, and survey the students about the impact to their willingness for learning the processor and its cache memory. Additional analysis will be realized after finishing the course this year to determine the results of the exams for the topics that cover the EDUCache simulator and the proposed hands-on exercises and examples.

References

[1] B. Atanasovski, S. Ristov, M. Gusev, and N. Anchev, "EDUCache simulator for teaching computer architecture and organization," in *Global Engineering Education Conference (EDUCON), 2013 IEEE*, March 2013, pp. 1015–1022.

[2] R. Shackelford, A. McGettrick, R. Sloan, H. Topi, G. Davies, R. Kamali, J. Cross, J. Impagliazzo, R. LeBlanc, and B. Lunt, "Computing curricula 2005: The overview report," *SIGCSE Bull.*, vol. 38, no. 1, pp. 456–457, Mar. 2006.

[3] M. Stojcev, I. Milentijevic, D. Kehagias, R. Drechsler, and M. Gusev, "Computer architecture core of knowledge for computer science studies," *Cyprus Computer Society J.*, vol. 5, no. 4, pp. 39–42, 2003.

[4] M. Stolikj, S. Ristov, and N. Ackovska, "Challenging students software skills to learn hardware based courses," in *Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on*, june 2011, pp. 339 –344.

[5] A. Clements, "Arms for the poor: Selecting a processor for teaching computer architecture," in *Frontiers in Education Conference (FIE), 2010 IEEE*, 2010, pp. T3E–1–T3E–6.

[6] J. L. Hennessy and D. A. Patterson, "Computer Architecture, Fifth Edition: A Quantitative Approach," MA, USA, 2012.

[7] M. Gusev and S. Ristov, "Performance gains and drawbacks using set associative cache," *Journal of Next Generation Information Technology (JNIT)*, vol. 3, no. 3, pp. 87–98, 31 Aug 2012.

[8] X. Liang, "A survey of hands-on assignments and projects in undergraduate computer architecture courses," in *Advances in Computer and Information Sciences and Engineering*, T. Sobh, Ed. Springer Netherlands, 2008, pp. 566–570.

[9] B. Nikolic, Z. Radivojevic, J. Djordjevic, and V. Milutinovic, "A survey and evaluation of simulators suitable for teaching courses in computer architecture and organization," *Education, IEEE Transactions on*, vol. 52, no. 4, pp. 449 –458, nov. 2009.

[10] S. Ristov, M. Stolikj, and N. Ackovska, "Awakening curiosity - hardware education for computer science students," in *MIPRO, 2011 Proceedings of the 34th International Convention, IEEE Conference Publications*, 2011, pp. 1275 –1280.

[11] D. Pop, D. G. Zutin, M. E. Auer, K. Henke, and H.-D. Wuttke, "An online lab to support a master program in remote engineering," in *Proceedings of the 2011 Frontiers in Education Conference*, ser. FIE '11. USA: IEEE Computer Society, 2011, pp. GOLC2–1–1–GOLC2–6.

[12] I. Kastelan, D. Majstorovic, M. Nikolic, J. Eremic, and M. Katona, "Laboratory exercises for embedded engineering learning platform," in *MIPRO, 2012 Proc. of the 35th Int. Conv.*, 2012, pp. 1113–1117.

[13] J. Qian, R. Wang, S. Shi, Y. Zhu, and Z. Xie, "Simplifying and integrating experiments of hardware curriculums," in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 9, 2010, pp. 610–614.

[14] D. Kehagias and M. Grivas, "Software-oriented approaches for teaching computer architecture to computer science students," *Journal of Communication and Computer*, vol. 6, no. 12, pp. 1–9, Dec. 2009.

[15] X. Wang, "Multi-core system education through a hands-on project on fpgas," in *Frontiers in Education Conference (FIE), 2011*, 2011, pp. F2G–1–F2G–6.

[16] J. H. Lee, S. E. Lee, H.-C. Yu, and T. Suh, "Pipelined cpu design with fpga in teaching computer architecture," *Education, IEEE Transactions on*, vol. 55, no. 3, pp. 341–348, 2012.

[17] C. Reinbrecht, J. Da Silva, and E. Fabris, "Applying in education an FPGA-based methodology to prototype ASIC soft cores and test ICs," in *Programmable Logic (SPL), 2012 VIII Southern Conference on*, 2012, pp. 1–5.

[18] N. Ackovska and S. Ristov, "Hands-on improvements for efficient teaching computer science students about hardware," in *Global Engineering Education Conference (EDUCON), 2013 IEEE*, March 2013, pp. 295–302.

[19] B. Hatfield and L. Jin, "Improving learning effectiveness with hands-on design labs and course projects for the operating model of a pipelined processor," in *Frontiers in Education Conference (FIE), 2010 IEEE*, 2010, pp. F1E–1–F1E–6.

[20] L. Da, "Computer hardware curriculums, curriculum contents and teaching methods," in *Computer Science Education, 2009. ICCSE '09. 4th International Conference on*, 2009, pp. 1506–1511.

[21] L. He, "Computer architecture education in multicore era: Is the time to change," in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, vol. 9, 2010, pp. 724–728.

[22] D. Patti, A. Spadaccini, M. Palesi, F. Fazzino, and V. Catania, "Supporting undergraduate computer architecture students using a visual mips64 cpu simulator," *Education, IEEE Transactions on*, vol. 55, no. 3, pp. 406 –411, aug. 2012.

[23] J. Edler and M. D. Hill, "Dinero iv trace-driven uniprocessor cache simulator," 2012. [Online]. Available: http://pages.cs.wisc.edu/~markhill/DineroIV/

[24] A. Jaleel, R. S. Cohn, C.-K. Luk, and B. Jacob, "Cmpsim: A pin-based on-the-fly multi-core cache simulator," in *The Fourth Annual Workshop MoBS, co-located with ISCA '08*, 2008.

[25] Y.-T. Chen, J. Cong, and G. Reinman, "Hc-sim: a fast and exact l1 cache simulator with scratchpad memory co-simulation support," in *Proc. of the 7-th IEEE/ACM/IFIP Int. conf. on HW/SW codesign and system synthesis (CODES+ISSS '11)*. USA: ACM, 2011, pp. 295–304.

[26] E. Herruzo, J. Benavides, R. Quislant, E. Zapata, and O. Plata, "Simulating a reconfigurable cache system for teaching purposes," in *Microelectronic Systems Education (MSE '07). IEEE International Conference on*, 2007, pp. 37 –38.

[27] A. Misev and M. Gusev, "Visual simulator for ILP dynamic OOO processor," in *WCAE '04, Proceedings of the workshop on Computer architecture education: in conduction with the 31st International Symposium on Computer Architecture*, E. F. Gehringer, Ed. ACM, USA, 2004, pp. 87 –92.

[28] Valgrind, "System for debugging and profiling linux programs," [retrieved: May, 2013]. [Online]. Available: http://valgrind.org/

[29] SimpleScalar LLC, "Simplescalar tool set," [retrieved: May, 2013]. [Online]. Available: http://www.simplescalar.com/

[30] University of Extremadura, "Smpcache - simulator for cache memory systems on symmetric multiprocessors," [retrieved: May, 2013]. [Online]. Available: http://arco.unex.es/smpcache/

[31] W. Stallings, *Computer Organization and Architecture: Designing for Performance*, 6th ed. Prentice Hall, 2003.

[32] D. A. Patterson and J. L. Hennessy, "Computer organization and design, forth edition: The hardware/software interface," MA, USA, 2009.

[33] ACM/IEEE-CS Joint Interim Review Task Force, "Computer science curriculum 2008: An interim revision of cs 2001, report from the interim review task force," 2008. [Online]. Available: http://www.acm.org/education/curricula/ComputerScience2008.pdf