

Visual Software System for Memory Interleaving Simulation

Katarina Milenković¹, Žarko Stanisavljević¹, Jovan Đorđević¹

Abstract: This paper describes the visual software system for memory interleaving simulation (VSMIS), implemented for the purpose of the course Computer Architecture and Organization 1, at the School of Electrical Engineering, University of Belgrade. The simulator enables students to expand their knowledge through practical work in the laboratory, as well as through independent work at home. VSMIS gives users the possibility to initialize parts of the system and to control simulation steps. The user has the ability to monitor simulation through graphical representation. It is possible to navigate through the entire hierarchy of the system using simple navigation. During the simulation the user can observe and set the values of the memory location. At any time, the user can reset the simulation of the system and observe it for different memory states; in addition, it is possible to save the current state of the simulation and continue with the execution of the simulation later.

Keywords: Computer architecture and organization, Memory interleaving, Software system, Simulation, Main memory, Random-access memory, Laboratory exercise.

1 Introduction

Computer Architecture and Organization (CAO) is an important area of computer science, and it is essential for education of future computer science engineers [13]. Because of this, great attention is paid to teaching CAO at undergraduate studies at universities, both around the world and in our country. The aim of courses dealing with this topic is to present this complex matter in a good and systematic way, so that students can easily understand it. At the School of Electrical Engineering, University of Belgrade (SEE-UB) these topics are covered by six different undergraduate courses.

The main task of these courses is to introduce the basic concepts of CAO. In addition, students are given an insight into the way in which a typical computer system functions – its performance, characteristics, and the way it performs operations. After completing these courses, students should be able to design their own computer systems and develop high-performance applications, thanks to a good knowledge of CAO.

¹University of Belgrade, School of Electrical Engineering, Bulevar kralja Aleksandra 73, 11020 Belgrade, Serbia; E-mails: katarina.milenkovic@etf.bg.ac.rs; zarko@etf.bg.ac.rs; jrdjordjevic@etf.bg.ac.rs

Experience has shown that attending lectures and auditory exercises is sometimes not enough to fully understand and master these areas. Students achieve better results if, along with traditional teaching methods, they are provided with opportunities for practical work in the laboratory. Such course organization allows students to verify and broaden the knowledge they acquired in traditional classes. Following a joint recommendation for computer science course curricula of ACM and IEEE [1] associations, courses in these areas should be accompanied by laboratory exercises in order to enable students to develop their skills through practical work. In accordance with the recommendation, courses that cover the topic of CAO at SEE-UB consist of lectures, auditory exercises, and laboratory exercises. In laboratory exercises, students use software systems [2 – 5], which are specially designed for various topics that courses cover. This paper describes the Visual software system for memory interleaving simulation (VSMIS), implemented for the purpose of the course Computer Architecture and Organization 1.

The rest of the paper presents the motivation for the implementation of the VSMIS simulator, the structure of the implemented system, the laboratory exercise based on the simulator, its comparison with existing simulators, and the conclusion at the end.

2 Problem Description

The course in Computer Architecture and Organization 1 at the SEE-UB is a mandatory course in the fifth semester of the Computer Engineering undergraduate study program, and an elective course in the fifth semester of the Software Engineering undergraduate study program. It is also offered as an elective course to students of other electrical engineering study programmes, who believe the course can provide them with knowledge they will need in their future work. The course consists of two classes of lectures, two classes of auditory exercises and one class of laboratory exercises per week.

The aim of the course is to introduce students to the basic elements of the memory hierarchy system and the pipeline organization of the processor, help them understand their structures, basic principles and the way of functioning. The first part of the course is dedicated to the memory system [6]. At the beginning, students are introduced to the cache memory mechanism which is used to improve the memory response time. After that, the virtual memory technique is explained. This hardware-software technique can be used to increase the size of the address space. Finally, the course covers a hardware technique for accelerating memory access, called memory interleaving. The second part of the course is dedicated to the pipeline organization of the processor. The VSMIS simulator, described in this paper, deals with the mechanism of memory interleaving.

The method for accelerating memory access using memory interleaving is one of the advanced techniques, and is therefore more difficult to understand. In order to understand this mechanism, it is necessary to understand how operating memory needs to be organized, how the system bus cycles should be organized, as well as understanding the specific communication protocols between devices that access the memory and the memory modules. All this is much easier through practical work. Work in the laboratory is aimed at showing the specific behavior of computer systems and understanding their desired characteristics. In these cases, the computer system simulators have proved to be an appropriate substitute for complex and expensive real hardware components [7].

An important question which the authors of courses in the field of CAO face, is which simulator is best suited for a course they teach. The ideal simulator should cover all the topics that are taught during the course, provide the possibility of visual presentation of simulation on multiple hierarchical levels, and it should be easy to use and relate to the previous knowledge of students who attend the course.

In existing literature there is a large number of software systems for simulating computer systems which cover various topics related to CAO. Many of them are available freely online and can be used in courses. Nevertheless, it is not easy to find a system that covers all the topics taught in the course and fully suits it. The existing software systems are designed for different purposes with some specific requirements and restrictions. In literature, one can find papers [8] which analyze, evaluate and classify some of the existing software systems, according to the basic criteria (the area of coverage, the way and level of displaying the implementation details, the way of simulation execution, availability, etc.). Based on literature overview, it can be concluded that the systems that cover the topic of memory interleaving are *ISE Design Suite* [9], *M5* [10], *Quartus II* [11] and *Simics* [12]. Although these simulators have been successfully used for a long time, detailed analysis shows they are not suitable for laboratory exercises of the course of Computer Architecture and Organization 1 at the SEE-UB. These systems are either commercial, without the possibility of visual presentations or simply too complicated to use and not suitable for integration with the existing system used in the course for other topics. Therefore, it has been decided to develop a new simulator.

3 Software System Description

The technique of memory interleaving implies execution of multiple operations on different memory modules simultaneously. It may have different effects in terms of speed acceleration, depending on computer system configuration.

In order to be able to apply the technique of memory interleaving, it is necessary that the memory consists of several smaller modules. In case of multiple access requests to the memory at a given time, there is a possibility that these requests relate to memory locations from different memory modules. Therefore, with a proper organization of memory modules and the system bus, the requests can be processed in parallel. In this way, the duration of a single memory access remains the same or even slightly increases, due to additional overhead operations that have to be performed. However, the total time of execution of multiple memory accesses can be significantly reduced. Thus, the acceleration is achieved at the level of program execution.

In order to provide parallel access to different memory modules, the system bus has to be organized in an appropriate way, with (non-atomic) split cycles. With this type of system bus cycles, a device which initiates an operation on the bus (called a master) holds the bus busy only for the amount of time needed to exchange information that is necessary to complete the operation with a device that should perform the operation (called a slave). The system bus is free while the memory modules process the requests. The bus cycles that can be performed on this type of bus are: the initiation of a write request, the initiation of a read request, and the response to a read request.

A typical computer system consists of a processor, memory and a few I/O devices, which are all connected by the system bus. During the cycle of initiation of a write request or the cycle of initiation of a read request, a master device can be a processor or a direct memory access (DMA) controller, and a slave device is a memory module. During the cycle of the response to a read request, a master device is a memory module, and a slave device can be a processor or a DMA controller.

VSMIS is a simulator of a fixed computer system, with a system bus (SYSTEM BUS), arbitrator (ARBITRATOR), 16 I/O devices (DEVICES) and 16 memory modules (MODULES). The devices can be with single or burst memory access and can generate requests for accessing memory. Memory modules have a capacity of 2^{20} bytes each, and an addressable unit is byte. The way in which the addresses are arranged per memory modules is determined by special microswitches that are common to all modules. The system bus connects all I/O devices and memory modules. It consists of 24 address lines (ABUS23...0), 8 lines of data (DBUS7...0) and four control lines (WRBUS, RDBUS, DABUS and ACKBUS). The arbitrator determines which of the devices or memory modules can realize the next cycle on the bus. It uses a parallel scheme for arbitration. The arbitrator receives requests for the use of the bus from the I/O devices or the memory modules through the line BRQ, and sends back a grant for the use of the bus to the I/O devices and the memory modules through the line BG. All I/O devices, memory modules and the

arbitrator operate synchronously to a common clock signal *Tclk*. The structure of the implemented system is given in Fig. 1, followed by a description of the structure of some specific parts of a computer system.

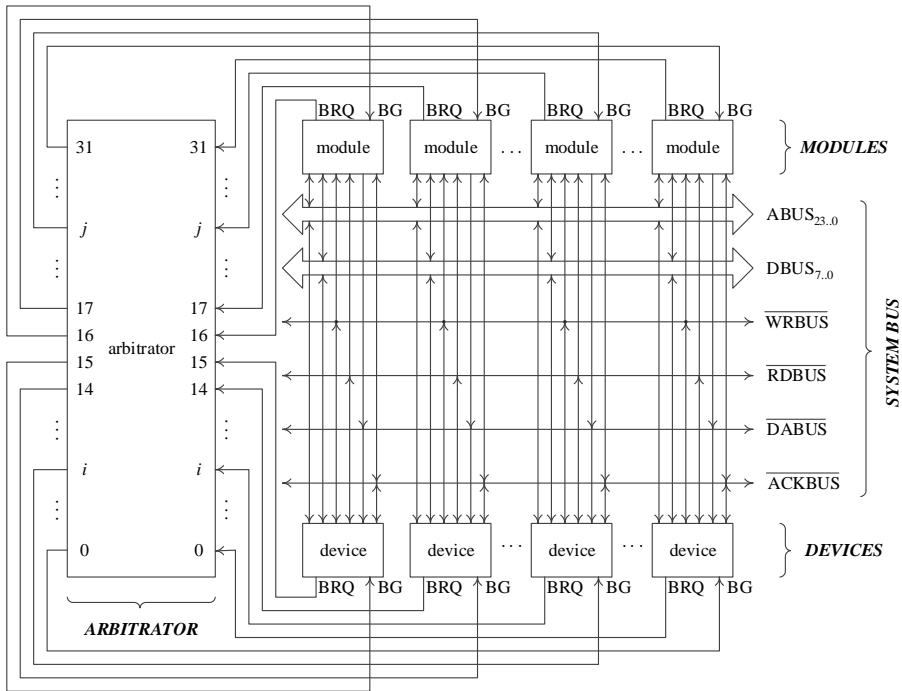


Fig. 1 – Computer system structure.

3.1 Arbitrator

In the previously described computer system, the arbitrator (shown in Fig. 2) determines which of the I/O devices or memory modules can perform the next cycle on the bus, using a parallel scheme for arbitration. The requests for the use of the bus in the arbitrator come through lines BRQ_{31...0}. Through the lines BRQ_{31...16}, the arbitrator receives requests from the memory modules and through the lines BRQ_{15...0} it receives requests from the I/O devices. The arbitrator's entrance priorities are fixed. The input 31 has the highest and the input 0 the lowest priority. The arbitrator sends back grants for the use of the bus through the lines BG_{31...0}. Through the lines BG_{31...16}, the arbitrator sends back grants to the memory modules and through lines BG_{15...0} grants to the I/O devices. This way of connecting input lines to the arbitrator ensures that requests for the use of the bus coming from memory modules have higher priority than requests coming from the I/O devices. All request signals and

grants are generated simultaneously to a common clock signal Tclk. The grant signal is given for only one period of the signal Tclk.

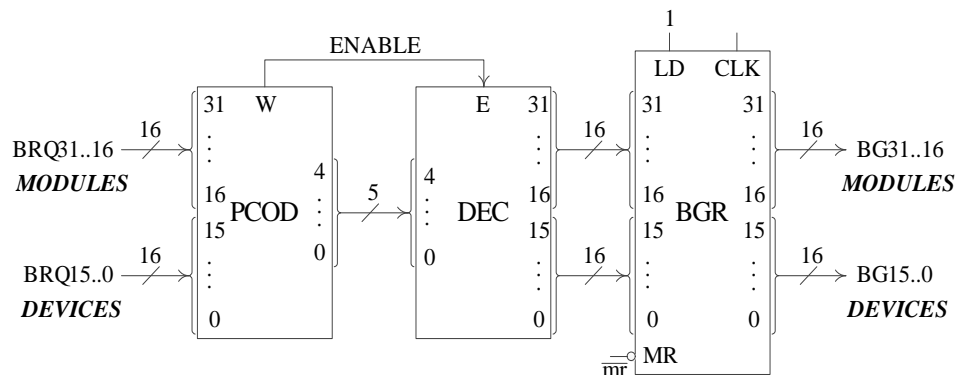


Fig. 2 – Arbitrator.

3.2 System Bus

In the computer system described in Section 3, the system bus was implemented with (non-atomic) split cycles. This type of bus supports the cycle of sending a request to write byte of data, the cycle of sending a request to read byte of data, and the cycle of sending a request to fetch byte of data. The operation of writing into memory module is implemented with the cycle of sending a request to write byte of data, and the operation of reading from memory module is implemented with the cycle of sending a request to read byte of data and the cycle of sending a request to fetch byte of data. Before the initialization of any cycle on the bus, the arbitration must be performed to determine which of the I/O devices or memory modules can perform the cycle on the bus. Through values one and zero of the BRQ line, the I/O device or memory module places and withdraws request for the use of the bus, respectively. Through values one and zero of the BG line, the I/O device or memory module obtains and loses permission for the use of the bus, respectively. When it gets value one, the signal BRQ will hold that value until a grant is obtained. When it gets value one, the signal BG will hold that value for only one period of the clock signal Tclk during which I/O device or memory module which got the grant to use the bus performs a cycle on the bus and sets the value of signal BRQ back to zero. Fig. 3 shows the cycle of sending a write request in case of a device with single memory access and burst memory access.

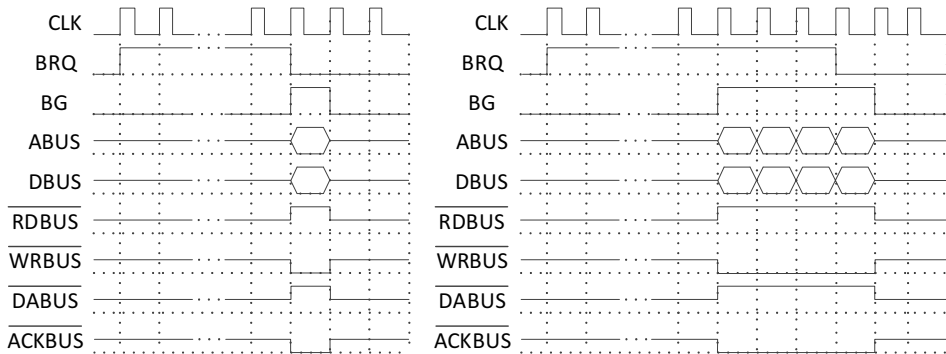


Fig. 3 – Cycle of sending a write request in case of a device with single memory access (left) and burst memory access (right).

3.3 I/O Devices

Devices in the computer system described in Section 3, consist of two parts: the working part and the interface. These two parts exchange information through the corresponding lines. The interface connects the working part to the system bus. The working part of the device is part whose structure and functionality vary from device to device. The working part can turn to the interface with a request to carry out an operation. The working part, before initiating any operation, must ensure that the registries relevant to connection to the interface contain all the necessary values for the operation. I/O devices can be with single memory access and burst memory access.

3.3 Memory Modules

Memory modules in the computer system described in Section 3, consist of two parts: the working part and the interface. These two parts exchange information through the corresponding lines. The interface connects the working part to the system bus. The working part of the memory module is the part whose functionality is the same for each module, regardless of how it is implemented. Working parts of different memory modules connect with interfaces in the same way.

This computer system gives the possibility of forming five different configurations of memory modules, more precisely five different ways of forming module numbers and scheduling addresses per modules: the adjacent addresses in the same module, the adjacent addresses in adjacent modules, and three different variants of mixed arrangements of addresses (combination of the previous two ways).

VSMIS has a visual interpretation: it displays details of implementation from the block level to the register transfer level (RTL), it has the ability to manage simulation process, and it is interactive. On the one hand, this system is

complex enough to represent the desired topic, and yet it is well adapted to the level of students' knowledge.

4 Laboratory Exercise Based on the VSMIS Simulator

This section describes the laboratory exercise on the course Computer Architecture and Organization 1 at SEE-UB, based on the use of VSMIS simulator, and it provides an example of the system use case.

4.1 Description of Laboratory exercise

The laboratory part of the course Computer Architecture and Organization 1 at SEE-UB is carried out through five laboratory exercises which accompany the materials covered in classes of lectures and auditory exercises. VSMIS simulator covers the fourth laboratory exercise, for which the topic is memory interleaving.

In the context of prepared material for this laboratory exercise, students receive a few different computer system configurations that support memory interleaving as well as explanations of the scenarios according to which operations and request for access to memory are performed in these simulations. VSMIS enables teachers to predefine the desired system configuration, save it in a separate file (with the extension .mmi) and make prepared files available to students. Based on some typical computer systems configurations that support memory interleaving, there are a few different cases:

- A computer system with a single I/O device with burst memory access.
- A computer system with multiple I/O devices with a single memory access.
- A computer system with multiple I/O devices with a single memory access and a single device with burst memory access.

The student's task is to trigger the execution of simulation clock by clock, watch the events of the proper part of the system, and respond to the questions regarding the execution of the simulation.

VSMIS simulator is available to students for independent work at home. With this option students have the ability to define an arbitrary computer system configuration and observe the execution of a simulation.

4.2 Use Case Example

The VSMIS system will be described using the case of a computer system with multiple I/O devices with a single memory access and a single device with burst memory access.

After starting VSMIS, the user will see the initial system window (shown in Fig. 4). This window provides the option to choose the way to start or to exit the

system. Users are given two options for getting started: *Set simulation parameters* and *Load simulation*. In laboratory exercises, when students receive a prepared configuration file, they should select the option *Load simulation*, then choose the appropriate file and configure the system. In this section the manual configuration of the system will be explained and the option *Set simulation parameters* will be selected.

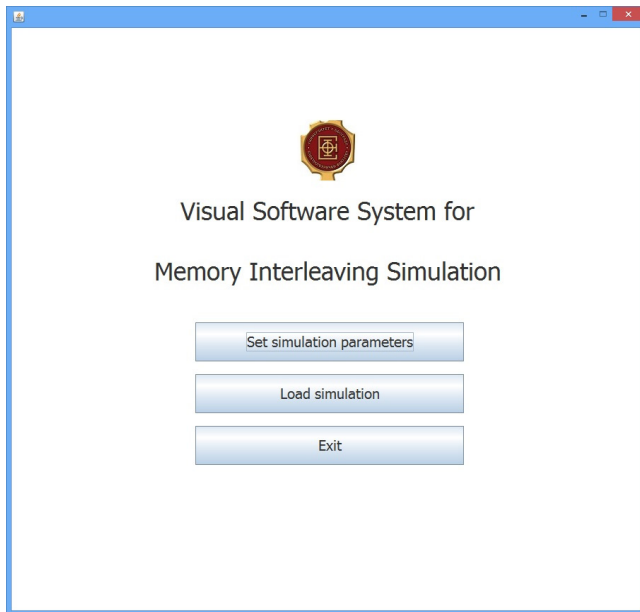


Fig. 4 – *Initial window.*

The first step in the simulation is to configure the computer system. In VSMIS, it is done in two separate dialogues. In the first dialogue (shown in Fig. 5), memory configuration is set. It is possible to select the desired schedule of addresses per memory module, and change the parameters of the memory modules. Each module can be set with an identifier and the time of access.

In the second dialogue (shown in Fig. 6), the I/O device configuration is performed. It is possible to change the parameters of devices separately. Each device can be set with an identifier and one of the two types (burst or single memory access). If a particular device generates a request for memory access, it is necessary to define the parameters of that access. All relevant information, necessary to perform appropriate access (the clock when access starts, the type of operation and other) should be entered into the corresponding dialogue, shown in Fig. 7. The I/O devices that do not define the parameters of access do not generate requests for memory access.

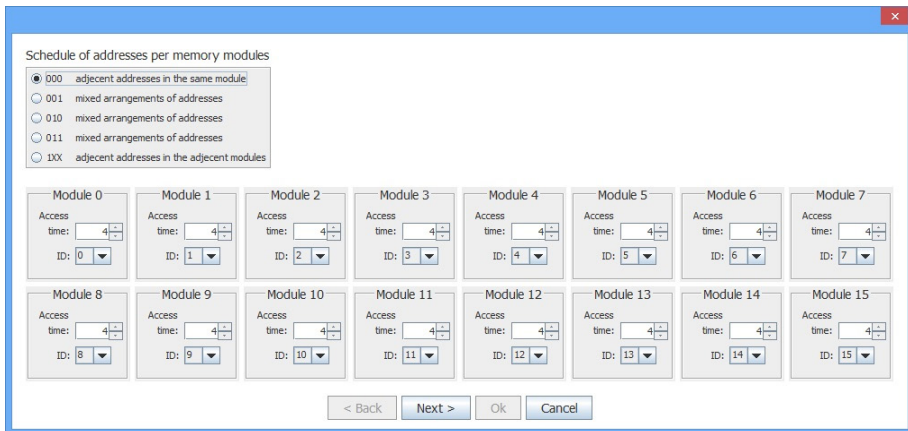


Fig. 5 – Dialogue for computer system configuration (1).

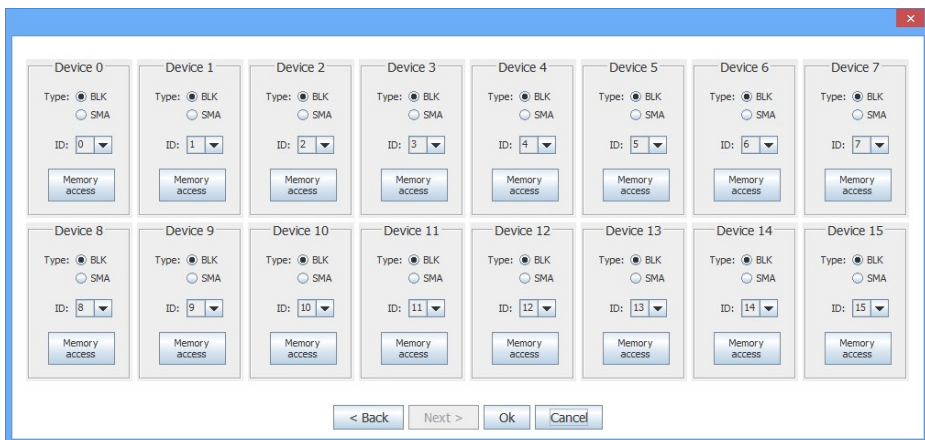


Fig. 6 – Dialogue for computer system configuration (2).

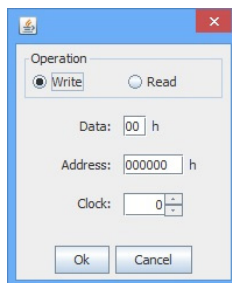


Fig. 7 – Dialogue for setting memory access parameters for single memory access device.

After completing the configuration process, the user will be shown the main window, with the structural scheme of the system (shown in Fig. 8). Structural schemes in the system are displayed from the highest level of abstraction (block level), down to the lowest RTL level. In the main window, it is possible to move through the entire hierarchy of the system and to observe the desired part of the system at a given time. Returning to a higher authority (parental view) is possible by selecting UP option from the panel with the information about the current view on the system.

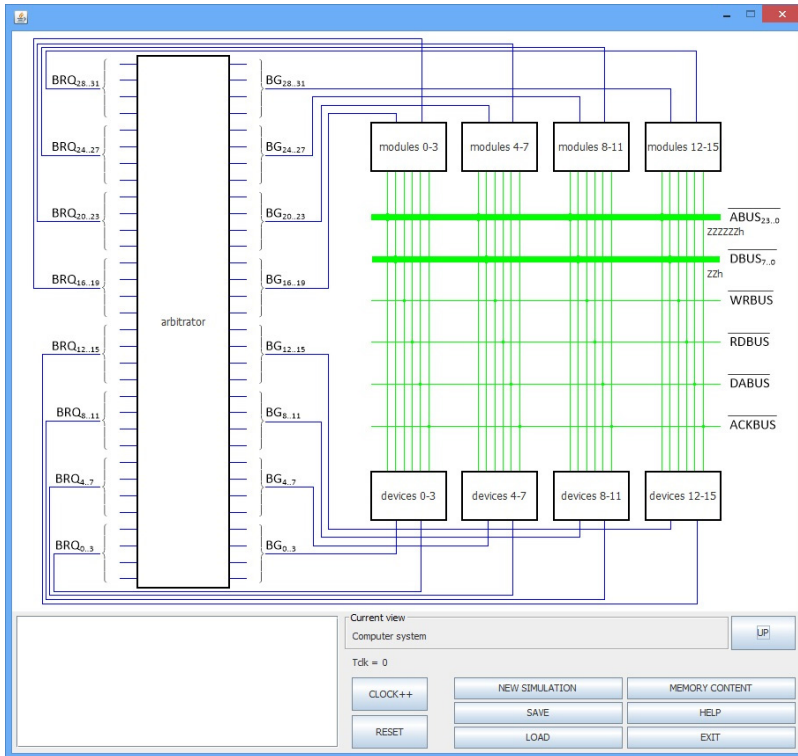


Fig. 8 – Main window.

At the beginning, the panel for showing the structural schemes displays the structural scheme of the entire system, located at the highest level of the hierarchy view. The signals in the system are displayed as lines. Depending on the value of the signal, it is displayed in one of the following four colors: blue (value 0), red (value 1), green (high Z) or black (value for group of lines different than high Z). All signals that consist of more than one bit are associated with a text label which represents signal value. The values in text labels can be displayed in one of the binary, octal, decimal, or hexadecimal numeral system.

The simulation can be controlled by options CLOCK++ and RESET. Option CLOCK++ allows running the simulation in advance for one clock. The value of the label Tclk shows which clock signal is obtained in the execution of the simulation. Option RESET brings the simulation to its initial state.

Besides the panel for displaying structural schemes, the main window contains the panel for displaying information about the microprogram (shown in Fig. 9). It shows the current step of the algorithm which generates control signals for the control unit of the system component, which is displayed in the view.

```

step0: if(WR·EQ, (ldDIR, ldAR, OENACK),
      if(RD·EQ, (ldDIR, ldAR, OENACK),
      if(WR·EQ) then step1
      else (if(RD·EQ) then step2 else step0))
    
```

Fig. 9 – Panel with information about a microprogram step.

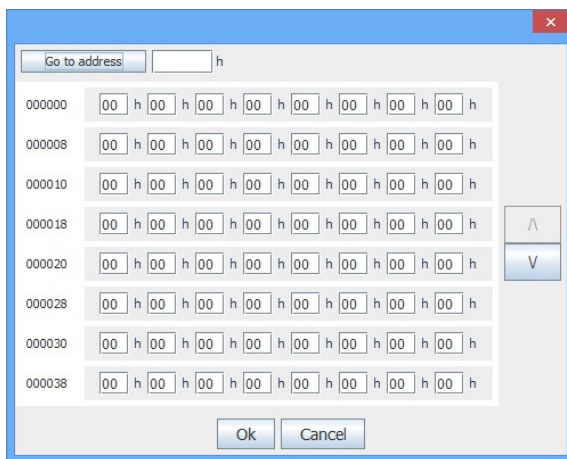


Fig. 8 – Dialogue showing memory content.

At any time during the execution of the simulation, it is possible to choose one of the options from the main window. The option NEW SIMULATION allows the new initialization of the system. It causes the opening of a dialogue for system initialization. The option SAVE is used to save the current state of the simulation. It allows the user to terminate its work, but to preserve the

current state of the simulation in order to make it possible to continue with the execution of the simulation later on. Files with the simulation state are saved with the extension .mmi. The option LOAD is used to initialize the system by restoring an old simulation, which was previously saved. If the user wants to view or set the values of memory locations, they have to choose the option MEMORY CONTENT in the main window. That will open a new dialogue, shown in Fig. 10, which provides the user with the aforementioned features. The option HELP provides some basic information about the application. The option EXIT enables the user to exit the system.

5 Comparison with Existing Simulators

In this section VSMIS is compared to the existing simulators that cover the topic of memory interleaving. A thorough analysis of different simulators for teaching computer architecture and organization can be found in [8]. Only those which address the issue of memory interleaving are analyzed.

The main categorization of existing software systems can be made based on the manner of their implementation. Based on this criterion, software systems can be divided into two groups: fixed computer systems simulators and configurable computer systems simulators. The first group gives less flexibility, but is suitable for giving users an insight into the architecture of the system, its behavior in different situations, and performance analysis. The second group provides greater flexibility, but it is hardly appropriate for complex computer system educational needs. As the objective of the course of Computer Architecture and Organization 1 at the SEE-UB is to introduce students to the basic elements of the memory hierarchy system (their structures, basics of functioning, etc.), the simulators of fixed computer systems perfectly meet its needs. Among the aforementioned software systems that cover the topic of memory interleaving, *Simics* [12] belongs to the group of simulators of fixed computer systems, and *Quartus II* [11], *M5* [10] and *ISE Design Suite* [9] to the group of simulators of configurable computer systems. The created software system VSMIS belongs to the group of simulators of fixed computer systems and therefore provides some limitation.

Another important feature of the software system is the existence of graphical presentation. For the purposes of the course, graphical presentation is very important because it greatly facilitates the understanding of complex matter. Among the aforementioned software systems that cover the topic of memory interleaving, *Simics* [12] and *M5* [10] do not have graphical presentation. *Quartus II* [11] and *ISE Design Suite* [9] have graphical presentation and they display the implementation details up to the RTL level. VSMIS has graphical presentation and it also displays the implementation details up to the RTL level.

The important thing when observing a computer system is the way of simulation execution. Some software systems allow the execution of simulation on clock level, some allow the execution on instruction level, and some allow the execution on program level. For the purposes of the course and complete understanding of how the system functions, it is important to have the possibility to execute simulation on clock level. Among the aforementioned software systems that cover the topic of memory interleaving, *Quartus II* [11], *M5* [10], and *ISE Design Suite* [9] allow the execution of simulation on clock level, and *Simics* [12] allows the execution of simulation on instruction level. *VSMIS* allows the execution of simulation on clock level.

Table 1
Comparison of VSMIS with existing simulators.

	Implementation	Graphics	Execution
<i>Simics</i>	fixed	No	instruction level
<i>M5</i>	configurable	No	clock level
<i>Quartus II</i>	configurable	Yes, RTL	clock level
<i>ISE Design Suite</i>	configurable	Yes, RTL	clock level
<i>VSMIS</i>	fixed	Yes, RTL	clock level

The previously analyzed software systems were developed to meet very specific requirements and none of them had memory interleaving in focus. These software systems only partially fulfill the need at the Computer Architecture and Organization 1 course, since some of them allow memory interleaving to be simulated, but none of them makes it possible to cover all of the desired situations. Therefore with the *VSMIS* we aimed to close the gap between the available software systems and the needs of the Computer Architecture and Organization 1 course.

6 Conclusion

This paper presents the software system that provides a visual simulation of computer systems with memory interleaving. *VSMIS* was created for the laboratory exercises at the course Computer Architecture and Organization 1 at SEE-UB. The implemented system provides a simple user interface for configuration, initialization and execution of simulations. All the options are quite intuitive, thus getting familiar with system does not require much time. It is sufficient that the user has some basic knowledge of the memory system architecture and organization. Visual representation of simulation flow is implemented in a way that enables the user to gain an insight into all parts of the computer system and to provide the necessary information for simulation

monitoring. Special attention was given to the simplicity of the visual display. It is clear and unencumbered by superfluous information. VSMIS is beneficial for students because it allows them to learn course materials in an easier and a more interesting way, as well as for teachers who have the opportunity to demonstrate the functioning of many different computer systems thanks to VSMIS.

Various simulators have already been developed for the laboratory exercises at the course Computer Architecture and Organization 1 at SEE-UB, such as software systems for simulation of cache memory and virtual memory, which were developed a few years ago. The simulator described in this paper, along with the previously mentioned simulators for cache memory and virtual memory, completely covers the area of the memory hierarchy system taught in the course.

It is possible to upgrade the VSMIS system in the future in order to make the execution of the simulation flexible. In addition to the already existing options, it is possible to add the option of returning the simulation flow for one clock backwards or positioning it to the desired clock during the simulation. The introduction of an automatic review and assessment of student work is also considered.

The system is in use in the course in the current school year (2016/2017) and the analysis of students' results and their satisfaction with the system will be carried out at the end of school year.

7 Acknowledgment

This paper was partially supported by the Ministry of Education, Science and Technological Development of Republic of Serbia, project number III44009. Special thanks to Dunja Živanović for proofreading.

8 References

- [1] Computer Science Curricula 2013, Association for Computing Machinery and IEEE Computer Society, Dec.2013.
Available at: <https://www.acm.org/education/CS2013-final-report.pdf>
- [2] N. Grbanovic, J. Djordjevic, B. Nikolic: Software Package for an Educational Computer System, International Journal of Electrical Engineering Education, Vol. 40, No. 4, 2003, pp. 270 – 284.
- [3] J. Djordjevic, B. Nikolic, A. Milenkovic: Flexible Web-based Educational System for Teaching Computer Architecture and Organization, IEEE Transactions on Education, Vol. 48, No. 2, May 2005, pp. 264 – 273.
- [4] J. Djordjevic, B. Nikolic, T. Borozan, A. Milenkovic: CAL2: Computer Aided Learning in Computer Architecture Laboratory, Computer Applications in Engineering Education, Vol. 16, No. 3, 2008, pp. 172 – 188.

- [5] Z. Stanisavljevic, V. Pavlovic, B. Nikolic, J. Djordjevic: SDLDS—System for Digital Logic Design and Simulation, *IEEE Transactions on Education*, Vol. 56, No. 2, May 2013, pp. 235 – 245.
- [6] J. Djordjevic: *Computers Architecture and Organization*, Academic Mind, Belgrade, Serbia, 2015. (In Serbian).
- [7] G.S. Wolffe, W. Yurcik, H. Osborne, M.A. Holliday: Teaching Computer Organization/Architecture with Limited Resources using Simulators, 33rd SIGCSE Technical Symposium on Computer Science Education, Cincinnati, OH, USA, 27 Feb-03 March 2002, pp. 176 – 180.
- [8] B. Nikolic, Z. Radivojevic, J. Djordjevic, V. Milutinovic: A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization, *IEEE Transactions on Education*, Vol. 52, No. 4, Nov. 2009, pp. 449 – 458.
- [9] ISE Design Suite, Xilinx.
Available at: <http://www.xilinx.com/products/design-tools/ise-design-suite.html>.
- [10] N.L. Binkert, R.G. Dreslinski, L.R. Hsu, K.T. Lim, A.G. Saidi, S.K. Reinhardt: The M5 Simulator: Modeling Networked Systems, *IEEE Micro*, Vol. 26, No. 4, July/Aug. 2006, pp. 52 – 60.
- [11] QUARTUS II Software, Altera.
Available at:
<https://www.altera.com/products/design-software/fpga-design/quartus-ii/overview.html>.
- [12] Simics User Guide for Unix, Version 3.0, Virtutech, 2005.
Available at:
<http://www.ece.cmu.edu/~profotflex/lib/exe/fetch.php?media=documentation:simics-user-guide-unix.pdf>.
- [13] Computer Science Curricula 2016, Association for Computing Machinery and IEEE Computer Society, Dec. 2016.
Available at:
<http://www.acm.org/binaries/content/assets/education/ce2016-final-report.pdf>.