Title: Visualization of VHDL-based Simulations as a Pedagogical Tool for Supporting Computer Science Education

Authors: Godofredo R. Garay, Andrei Tchernykh, Alexander Yu. Drozdov, Sergey N. Garichev, Sergio Nesmachnow, Moisés Torres-Martinez

PII: S1877-7503(17)30385-X
DOI: http://dx.doi.org/doi:10.1016/j.jocs.2017.04.004
Reference: JOCS 652

To appear in:

Received date: 29-7-2016
Revised date: 17-3-2017
Accepted date: 4-4-2017

Please cite this article as: Godofredo R.Garay, Andrei Tchernykh, Alexander Yu.Drozdov, Sergey N.Garichev, Sergio Nesmachnow, Moisés Torres-Martinez, Visualization of VHDL-based Simulations as a Pedagogical Tool for Supporting Computer Science Education, Journal of Computational Sciencehttp://dx.doi.org/10.1016/j.jocs.2017.04.004

Visualization of VHDL-based Simulations as a Pedagogical Tool for Supporting Computer Science Education

Godofredo R. Garay[1], Andrei Tchernykh[2*], Alexander Yu. Drozdov[3], Sergey N. Garichev[3], Sergio Nesmachnow[4], Moisés Torres-Martinez[5]

[1] *University of Camaguey, Camaguey, Cuba. godofredo.garay@reduc.edu.cu*

[2] *Computer Science Department, CICESE Research Center, Ensenada, Mexico chernykh@cicese.mx*

[3] *Moscow Institute of Physics and Technology, Moscow/ Dolgoprudny, Russia, alexander.y.drozdov@gmail.com, sng355@gmail.com*

[4] *Universidad de la República, Montevideo, Uruguay, sergion@fing.edu.uy*

[5] *University of Guadalajara, Guadalajara, Mexico,mtorres1028@gmail.com*

[*] Corresponding author

Highlights

- We address concepts of simulation visualization as a pedagogical tool for supporting undergraduate computer science students
- We demonstrate the practicability and benefits of the proposed approach on example of a VHDL model of the network-to-memory data path in a network node.
- We discuss three VHDL-based visualization techniques to graphically illustrate various concepts of computer science: Block diagram, Signal waveform, and Performance-oriented signal visualizations.
- We described how an effectiveness study of simulation visualization could map a particular topic of computer architecture cognitive domain to Bloom's taxonomy.

Abstract

Communication between information processing systems becomes a challenge, especially in the "big data" era. It is a mandatory subject in the topic "Architecture and organization" of the computer science curriculum. However, in our experience, it is a rather complex topic for students. Simulation visualization can be used to graphically illustrate various concepts of computer science. In this paper, we present the application of NICSim-vhd, which is an acronym for VHDL-based Network Interface Card simulation model, as a pedagogical tool for supporting undergraduate computer science students' education. NICSim-vhd allows simulating the network-to-memory data path at a network node and generating Value Change Dump (VCD) files for simulation visualization of hardware description languages-based models. We provide a taxonomy of learner engagement with simulation visualization. Grounded in Bloom's well recognized taxonomy of understanding, we suggest how to assess the learning outcomes to which such engagement may lead.

Keywords: Active learning, computer science education, visualization, Bloom's taxonomy, VHDL

1 Introduction

The joint ACM and IEEE curriculum guidelines for undergraduate computer science degree programs emphasize the relevance of the knowledge area "Architecture and Organization (AR)" [1]. According to these instructional guidelines and curriculum suggestions, students should acquire an understanding and appreciation of computer system functional components, their characteristics, performance, interactions, and, in particular, the challenge of harnessing parallelism to sustain performance improvements.

One of the knowledge units included in AR is "Interfacing and Communication". The focus here is on the hardware mechanisms for supporting input/output (I/O) device interfacing. Topics like I/O fundamentals (e.g., handshaking, buffering), buses (bus protocols, arbitration, direct-memory access), and introduction to networks (communications networks as another layer of remote access, among others that should be considered by instructors) are included.

Indeed, teaching computer architecture requires a lot of effort by the instructor. Simulators can improve the teaching process, increase student willingness and ease ability to learn the material [2], [3]. Commonly, computer simulation is used as a supporting tool in the process of understanding the concepts of both computer architecture and computer organization, e.g., CPU [4], [5], assembly language [6], cache memory system [7], on-chip hardware components [5], and so on.

Understanding how computers work is hardly possible without having specialized computer laboratories or tools suitable for courses. These laboratories are too expensive to be available in all universities, especially, in poor countries [8], [9]. Also, it is necessary to periodically invest money to upgrade them.

An important question arises: are these labs flexible enough to be appropriate for assessing the workloads in various testing environments? Using computer architecture simulators in lab activities adds a new dimension to textbook theory by strengthening practical teaching.

Computer simulators are programs that contain a representation of authentic systems or hypothetical situations. They have a number of features that are of particular help in the teaching of science [10]. By changing parameter settings of system-under-test (SUT) a simulation model, professors and students can test “what if” cases, and gain insight on "unusual" workload patterns.

Regarding the I/O subsystem topic, Larraza-Mendiluze et al. [11] highlight the need for more educational research in order to make it less abstract and more attractive. To this end, developing and using different resources and educational methodologies based on a theory of learning should be considered [12], [13].

A traditional course model, in which the lecturer follows a text book, exhibits slides, and presents some theoretical exercises, is not enough to assure a through comprehension of what is being taught. The problem is due to both the teaching model and the lack of appropriate tools capable of translating the theory being presented into a more practical reality. Without a practical vision, the student tends to lose touch and just “float” around the introduced concepts and mechanisms without gaining insight into of what is really going on [14].

The study of computer hardware usually involves a lot of abstract concepts, complex hardware structural interconnections and dynamic hardware behavior. Commonly, it is hard for students to imagine how digital signals propagate inside computers to operate in different functional units.

Visualization of the activities, which occur inside a computer, might be an important aspect for improving computer science education [15], [16]. Computer-based visualizations like animations and simulations are effective teaching-learning resources across computer science domains.

In [17], the authors argue that such a technology, no matter how well it is designed, is of little educational value unless it engages learners in an active learning activity.

In this paper, we present the application of the NICSim-vhd tool [18], [19] as an experimental learning environment to teach computer architecture. The tool leads students through a more active participation in the learning process. We strongly believe that visualization is better than a thousand words when it comes to constructing a mental model of a machine operation. A major idea behind our approach is to take advantage of VHDL simulation visualization for skills training.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we explain the pedagogical foundations of our work. In Section 4, we present the

learning platform. In Section 5, we show how the learning platform can be used in classes for simulation visualization. In Section 6, we present simulation visualization in the context of Bloom's Taxonomy. In Section 7, we present our conclusions.

2 Related Work

Visualization plays an important role in understanding and designing computers, and is used in many areas of computer science (CS), e.g., algorithm animation, software engineering, etc.

The history of visualization in CS education focusing on artifacts that have a documented positive educational assessment is surveyed in [20]. The authors discuss how changes in computing technology have affected the development and incorporation of such visualization artifacts in CS education, and how recent technology changes are leading to progress in developing on-line textbooks.

However, despite the fact that visualization tools are one of the most studied research fields in CS education, most teachers and students neglect utilizing existing visualization tools for teaching and learning, according to [21]. Possible reasons for this problem, as well as directions for future research based on Activity Theory, and a theoretical framework borrowed from developmental psychology are discussed by the authors. Aspects of Activity Theory that are most relevant to the development of program visualization tools, and pursuing the implications of this theory for deepening our understanding of how these tools impact teaching and learning are also considered.

In [22], the effectiveness of providing visualization as part of the feedback in a problem solving software tutor on arithmetic expression evaluation is discussed. Data are collected over six semesters from multiple institutions. ANOVA analysis of the collected data is conducted in three stages. The authors conclude that visualization helped students learn significantly more concepts.

In [23], the M2S-Visual interactive cycle-by-cycle trace-driven visualization tool is presented. It was designed as an educational resource to familiarize students with parallel code execution on both CPUs and GPUs.

In [24], a web-based education platform for the visualization and animation of the digital logic design process is presented. It includes the design of combinatorial circuits using logic gates, multiplexers, decoders and look-up-tables, as well as the design of finite state machines.

Because programming is one of the most complex subjects in computer science, in [25], program visualization is adopted to make programming concepts more accessible to students. Two instructional scenarios are discussed. One of them is based on viewing animations, the other on the traditional instructions without systematic use of animations. The authors conclude that animations improve learning from several educational aspects: short-term and long-term knowledge acquisition, and drop-out rates.

In order to help students to enhance the learning of object-oriented programming concepts, in [26], a visualization tool is used. Visualization is found to be a promising approach in facilitating student concept images of basic object-oriented notions.

In [27], an educational MIPS simulator, DrMIPS, is described. The tool simulates the execution of an assembly program on the CPU, and displays the data-path graphically. Registers, data memory and assembled code are also displayed.

3 Pedagogical foundations

3.1 Constructivism in Computer Science education

In the 20s and 30s of the last century the founding works of Vygotsky studied how children construct an understanding of the world around them. Social constructivism and cognitive constructivism become two predominant educational theories. They developed by Lev Vygotsky and Jean Piaget and form the basis of many of today's educational technology tools in the classroom. The theories state the central role of social factors in child development, internalization not as the process of copying material from the environment, but as a transformative process, as well as, the individual as what develops. The differences pertain to the nature of the stimulus, nature and origin of psychological instruments, nature of self-regulation and novelty in development, direction of development, the concept of social development, and finally the role of language in development [28]. An effective classroom, where instructor and students are communicating optimally, is dependent on using constructivist strategies, tools and practices. Teaching techniques derived from the theory of constructivism are thought to be more successful than traditional techniques, because they explicitly address the necessary process of knowledge construction.

In [12], the authors discussed to what extent constructivism is applicable to CS education. According to constructivism, students construct knowledge by combining the experiential world with existing cognitive structures. The author claims that the application of constructivism to CS education must take into account characteristics that do not appear in the natural sciences.

For example, a (beginning) CS student has no effective model of a computer. By effective model the authors mean a cognitive structure that the student can use to make viable constructions of knowledge, based upon sensory experiences such as reading, listening, lectures, and working with a computer. They do not think that beginning CS students come to class with the effective model of a computer. The lack of such a model is a serious learning obstacle to CS. Thus, if the student does not bring a preconceived model to class, then we must ensure that a viable hierarchy of models is constructed and refined as learning progresses. This means that the computer model (e.g., CPU, memory, I/O peripherals) must be explicitly taught and discussed, not left to haphazard construction and not glossed over with facile analogies.

The classic pedagogical model at all levels of education is based upon the instructive model, where instructional sequences tackle the task of transferring the maximum amount of information from an active teacher to a passive learner. In general, the instructive model tends to be standardized and homogenized in the sense that the teaching is mostly directed to the class as a whole, and not to individuals within the class.

One way to overcome the limitations imposed by the instructive model is to include concepts from constructivism theory – the teacher/instructor plays not only the classical role of transmitting knowledge the best it can, but also serving as a "facilitator" of the learning process. In the constructivist model, the student is the central focus of the whole process of knowledge construction. The development of students' investigational/critical predicates and his ability to work cooperatively in group/teams are equally relevant tasks for the teacher.

3.2 A constructivism framework

Our pedagogical framework is inspired by the work of Maia et al. [14] and Moreno et al. [29]. Many computer architecture courses are based upon teacher presentation and explanation of concepts, rather than allowing the students to construct its own knowledge. This model may turn lectures into an extremely abstract and boring process. The constructivist theory provides an option for developing pedagogic proposals, possibly leading to better learning outcomes than those obtained with instructive models.

In this paper, we propose a constructivism framework to support learning in the knowledge area "Architecture and organization" of computer science curriculum [1]. Specifically, we concentrate on the knowledge unit "Interface and Communication". The main guidelines followed in developing this model are listed below:

- Teaching should be conducted in an individualized manner; the teacher should pay close attention to each student's absorption capability.
- The student-teacher interaction should have a strong emphasis on searching for practical and interesting questions.
- Work groups should be proposed as a forum to achieve cooperative learning.
- Knowledge can be constructed through different activities including discussion, mediation, and experimentation (Active Learning).
- The teacher should use VHDL-based simulation in conjunction with theoretical lectures, so that complex concepts underlying the subject may be better illustrated.
- The students should use VHDL-based simulation in the classroom and in homework as a form of understanding situations which are difficult to generate in a real environment.

The facility to develop and test hypotheses to create alternative solution proposals and discuss them with the other students and the teacher makes the simulator an essential tool in the learning process. The simulator emphasizes knowledge construction, as it makes multiple displays of reality possible, allowing students to test their own hypotheses, and learn from their successes and mistakes. Once faced with a specific problem, students can find real support in the simulator that helps them to actively search for a solution, improving their ability to identify, describe, and solve problems.

3.3 Learner understanding and Bloom's taxonomy

In order to study the effectiveness of various strategies for engaging learners in visualization, we have first to point out what we expect from learners studying a particular topic.

Rather than attempting to provide an all-encompassing breakdown for all of computer science, we use a general taxonomy developed by Bloom in 1956 [30]. It becomes incumbent upon any particular study of visualization effectiveness to define understanding within the particular area of CS in which that study is being conducted.

Bloom's taxonomy structures learner understanding along six increasingly sophisticated levels:

- Level 1: The knowledge level. This is characterized by mere factual recall with no real understanding of the deeper meaning behind presented facts.
- Level 2: The comprehension level. At this level, the learner is able to discern the meaning behind the facts.
- Level 3: The application level. The learner can now apply the learned material in specifically described new situations.
- Level 4: The analysis level. The learner can identify the components of a complex problem and break it down into smaller parts.
- Level 5: The synthesis level. The learner is able to generalize and draw new conclusions from the facts learned at prior levels.
- Level 6: The evaluation level. The learner is able to compare and discriminate among different ideas and methods. By assessing the value of these ideas and methods, the learner is able to make choices based on reasoned arguments.

4 Learning platform

There are a number of development environments available for designing circuits using a Hardware Description Language (HDL). However, most of these systems are commercial tools. Further, since they are aimed at developing commercial designs, most of the available features are often not necessary in an introductory level course. For a basic HDL development, students only need to edit, compile and simulate simple programs, typically contained within a single file. In this section, we provide further details about the individual components combined in the light-weight IDE.

Compiler and Simulator. The compiler and simulator should be light, open source and cross-platform. GHDL [31] meets these requirements and is suited for our purpose. It allows the user to compile and execute VHDL code directly. It has several commands, allowing the user to analyze, elaborate and run VHDL code/test with various options. It is a command-line tool, and can often be hard to use for a beginner.

Waveform Viewer. The VCD file is a text file with the values of signals at various time points. For easy visualization of results, we need a program which shows this information graphically. We use GTKWave, an open-source GTK+ based wave viewer, which runs on Unix, Windows and MacOSX [32]. It supports several file formats including standard VCD files. As shown in the next section, we can select the signals to be displayed, their radix and zoom-level.

5 Use of visualization techniques in class

Visualization techniques help students understand details of system architecture at various levels of complexity, and provide important supporting roles to instructors in the classroom. In this section, we propose three different visualization techniques: Block diagram visualization, Signal waveform visualization, and Performance-oriented signal visualization.

5.1 Block diagram visualization

A block diagram representation of computer concepts enables students to approach course material in more concrete way, and to visualize abstract behavior of computer hardware architecture more clearly and effectively. Describing the system in block diagrams provides a purely descriptive approach to its functionality and operation. In this approach, only a description of the computer I/O subsystem is given to the students, who are then expected to be able to describe the concepts. They could be asked to identify relationships between concepts. This is an easy way to introduce the topic, which could be used with students who are not majoring in computing.

Let us consider Network Interface Card (NIC) hardware as an example. Figure 1 shows the block diagram to visualize the functionality of the NIC as a physical interface between the computer and network cables. Using this diagram, instructor can explain NIC functionality in terms of the Open Systems Interconnection (OSI) reference model.

Conventional NICs perform Layer-1 (Physical) and Layer-2 (Data-link) processing. Typical questions at the physical layer (e.g., what electrical signals should be used to represent 1 and 0, or in how many nanoseconds a bit is transmitted?) can be addressed by instructors.

During the analysis, students should consider the characteristics of serial communication (Ethernet link) and parallel communication (PCI bus), and how a parallel data stream is converted to a serial data stream and vice-versa. At the data-link layer level, frame processing at the NIC can be discussed. Additionally, the need of a buffer for matching the rate at which the data is received from the network and the rate at which the NIC is sending the data across the I/O bus (and vice-versa), can be studied.

Analysis of the block diagram allows students to explore the functionality of the NIC hardware and enhance their understanding of the network I/O. Instructors can use block diagrams to stimulate in-class discussion, engage students in active learning and initiate a collaborative effort among students for finding answers and solutions to the functionality of circuits.

5.2 Signal waveform visualization

A solid knowledge of electronics is of major importance for a CS student. The means for achieving a good level of understanding, especially of the practical aspects, are an issue that is generally allocated in a CS program. Due to the applied nature of the subject, a pragmatic practice-based approach can be an appropriate solution to complete the technical preparation of students [33].

A typical computer science curriculum incorporates three topics in the hardware track: digital design, computer organization, and computer architecture. In such a curriculum, detailed study of the electrical aspects has to be borrowed from electrical engineering (a student takes a course in basic electricity followed by another in transistor electronics). The majority of the digital design textbooks in computer science either skips the electronics aspects of gates, or discusses topics assuming previous knowledge of the electrical aspects. However, to fully understand the electrical constraints, a digital design course (as a first course in computer science) requires previous knowledge in electrical and electronics concepts. The concepts are acquired through a sequence of courses in electronics. To present electrical topics under the

limited constraints of classes in computer science is a challenge to the computer science educator. This is especially true as related to coverage in many digital design texts.

Waveforms visualization can help students to find relationship among multiple signals, and to visualize signal patterns. Figure 2 shows the simulation waveform for the VHDL simulation model described in Figure 1.

In Figure 2, PCI-bus signals (PCI clock, Request, Grant, Frame, Address[†], Initiator ready, and Target ready) are shown. Using their waveforms, instructor can analyze basic principles of digital interfaces, such as two-state and tri-state logic (note that FRAME signal is tri-state).

5.3 Performance-oriented signal visualization

A performance-oriented approach is crucial in CS education. The computer I/O subsystem can be a bottleneck in computer systems. Its design has a major effect on computer performance. Performance evaluation of the I/O subsystem is coupled with different knowledge areas of CS curricula such as Architecture and Organization, Parallel and Distributed Computing, and Networking and Communications. Students should to be able to design and improve a system based on a quantitative and qualitative assessment of its functionality, usability and performance. From the performance evaluation viewpoint, the students are asked to calculate the performance of a computer system using different I/O techniques. This kind of question is considered a higher-level question, since it requires the application of knowledge, and often the evaluation of the results, in order to determine which technique is the most appropriate in the given context.

Since traditional classroom teaching and exercises are not capable of obtaining the goals mentioned above, we advocate performance-oriented signal visualization.

To analyze the dynamic behavior of interconnected computer system components and provide a more complete view of how computer hardware works, performance-oriented signal visualization is used. It is beneficial for students to visualize the hardware complexity in a more comprehensible way.

In VHDL simulations, the network workload is modeled by using signals (sig_ethclk, sig_pktarrival, sig_pktsize, and sig_pktreceived). The sig_buffer_fill_level_in_bytes signal allows user monitoring of how the NIC buffer gets filled and drained (Figure 3).

CS student should understand that Ethernet standard specifications impose a limit on the theoretical throughput achieved at system level. He should be able to compute maximum packet rates for full-duplex Ethernet [34]. To this end, he needs to obtain the packet duration times on the wire (as Ethernet uses a bit-serial transmission scheme, where the bit rate can be 10 Mbit/s, 100 Mbit/s, 1 Gbit/s, 10 Gbit/s, etc. and the bit time (i.e., time per bit) is the reciprocal of the bit rate).

In Figure 3, the arrival of minimum-size (72-bytes) packets is simulated. This scenario represents the worst case, requiring the most processing power. In general, a potential problem that should be analyzed with students from the performance evaluation viewpoint is buffer overflow at the NIC level.

Commonly, NIC hardware maintains an internal circular descriptor-ring structure. Notice that although buffer descriptors (BDs) are not transmitted over the network, a descriptor is

[†] In our simplified bus model, a 1-bit address (AD) line is used.

stored into the onboard buffer for each received packet. A DMA transfer across the I/O bus included sending packet payload and the corresponding 16-bytes buffer descriptor.

In order to create a performance-oriented way of thinking, students should evaluate fundamental performance indicators of the communication between information processing systems (e.g., bandwidth, latency, overhead, and throughput).

For example, the bandwidth of a parallel bus (e.g., PCI) can be computed taking into account its width and frequency. However, in our case study (Figure 1), such a bandwidth cannot be achieved due to overhead cycles occurring in the network-to-memory data path.

Performance-oriented signal visualization can be an effective alternative to illustrate these issues (Figure 3). Both NIC-side processing latency and (random) bus access latency impose an overhead on communication. To obtain a quantitative assessment of both overhead and actual data transfer cycles (DMA cycles), NICsim-vhd includes counters for latency cycles and DMA cycles whose outputs can be displayed by means of signals. For off-line analysis, the values of these counters and buffer behavior statistics are written to disk trace files.

Note that PCI is a shared bus. When a bus master (such as the NIC) asserts REQ, a finite amount of time expires until the first data element is actually transferred. This is referred to as bus access latency and consists of several components (arbitration latency, acquisition latency and initial target latency); see the enlarged detail in Figure 3.

Signal flow from a performance perspective can be fully explained by instructors, or partially by instructors and partially by students through questions and problem-solving in the classroom. Such a practice increases the student's curiosity about course content, and promotes meaningful learning experiences.

6 Simulation visualization in the context of Bloom's Taxonomy

As an example of how an effectiveness study (pragmatic trials) could map a particular area to Bloom's breakdown, we develop sample tasks in the area of computer architecture. We recognize that creating such a mapping is not a trivial task and the following classification is a starting point for deliberation.

Table 1 shows sample tasks for Bloom's comprehension levels. All tasks and assignments from Levels 2 to 6 should be solved by students individually or in groups with the help of visualization produced by NICsim-vhd (Figures 2 and 3).

The tasks at Level-1 (Knowledge), Level-2 (Comprehension) and Level-3 (Application) are of lesser complexity and can be assigned to students in the form of exercises. That is, well-defined assignments should be provided, in which the solving process and the expected results are known in advance and learners can check if they lead to the right solutions (Table 1).

The tasks at Level-4 (Analysis) and Level-5 (Synthesis) are of medium complexity and can be assigned to students as problems for them to solve.

The tasks at Level-6 (Evaluation) should be considered as projects of higher complexity. Problems are open-ended small-scale tasks, in which students might arrive at different solutions or use different solving methods. The proposed solution must meet given specifications and constraints. Projects are challenging ill-defined tasks in which students take part in determining both the objectives and the resources required for a system development.

The project is aimed at fostering participants' technical knowledge, collaborative work, aspiration and imagination, and, in our view, more important from a teaching and learning perspective.

7 Conclusions

In this paper, we show that simulation visualization can be used to graphically illustrate various concepts of computer science. We present the NICSim-vhd, VHDL-based Network Interface Card simulation model, as a pedagogical tool for supporting undergraduate computer science students. We discuss three different visualization techniques to allow students to engage in computer architecture topics from different perspectives.

Our approach allows students to visualize computer hardware concepts in more tangible ways, in order to improve their learning experience. We describe how Bloom's taxonomy can be used to differentiate levels of understanding in the areas of computer architecture. We show that once Bloom levels have been applied to learning objectives, the teacher's activity in designing a lecture to cover a particular topic becomes easier, less nebulous, and more clearly defined.

Acknowledgements. The work is partially supported by the Ministry of Education and Science of the Russian Federation under contracts RFMEFI58214X0003 and 02.G25.31.0061/12/02/2013 (Government Regulation No 218 from 09/04/2010), CONACYT (Consejo Nacional de Ciencia y Tecnología, México), grant no. 178415. The authors wish to thank Mr. Adrian Bondy who assisted in the proof-reading of the manuscript.

References

[1] ACM/IEEE-CS Joint Task Force on Computing Curricula, "Computer Science Curricula 2013," ACM Press and IEEE Computer Society Press, Dec. 2013.

[2] N. Rutten, W. R. van Joolingen, and J. T. van der Veen, "The learning effects of computer simulations in science education," *Comput. Educ.*, vol. 58, no. 1, pp. 136 – 153, 2012.

[3] P. W. C. Prasad, A. Alsadoon, A. Beg, and A. Chan, "Using simulators for teaching computer organization and architecture," *Comput. Appl. Eng. Educ.*, vol. 24, no. 2, pp. 215–224, 2016.

[4] D. Patti, A. Spadaccini, M. Palesi, F. Fazzino, and V. Catania, "Supporting Undergraduate Computer Architecture Students Using a Visual MIPS64 CPU Simulator," *IEEE Trans. Educ.*, vol. 55, no. 3, pp. 406–411, Aug. 2012.

[5] R. Poss, M. Lankamp, Q. Yang, J. Fu, I. Uddin, and C. R. Jesshope, "MGSim—A simulation environment for multi-core research and education," in *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), 2013 International Conference on*, 2013, pp. 80–87.

[6] T. Sondag, K. L. Pokorny, and H. Rajan, "Frances: A Tool for Understanding Computer Architecture and Assembly Language," *Trans Comput Educ*, vol. 12, no. 4, pp. 14:1–14:31, Nov. 2012.

[7] 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*, 2013, pp. 1015–1022.

[8] G. S. Wolffe, W. Yurcik, H. Osborne, and M. A. Holliday, "Teaching Computer Organization/Architecture with Limited Resources Using Simulators," *SIGCSE Bull*, vol. 34, no. 1, pp. 176–180, Feb. 2002.

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

[10] N. Srisawasdi and P. Panjaburee, "Exploring effectiveness of simulation-based inquiry learning in science with integration of formative assessment," *J. Comput. Educ.*, vol. 2, no. 3, pp. 323–352, 2015.

[11] E. Larraza-Mendiluze and N. Garay-Vitoria, "Approaches and Tools Used to Teach the Computer Input/Output Subsystem: A Survey," *IEEE Trans. Educ.*, vol. 58, no. 1, pp. 1–6, Feb. 2015.

[12] M. Ben-Ari, "Constructivism in Computer Science Education," *SIGCSE Bull*, vol. 30, no. 1, pp. 257–261, Mar. 1998.

[13] L. Moreno, C. Gonzalez, I. Castilla, E. Gonzalez, and J. Sigut, "Applying a constructivist and collaborative methodological approach in engineering education," *Comput. Educ.*, vol. 49, no. 3, pp. 891 – 915, 2007.

[14] L. P. Maia, F. B. Machado, and A. C. Pacheco Jr., "A Constructivist Framework for Operating Systems Education: A Pedagogic Proposal Using the SOsim," *SIGCSE Bull*, vol. 37, no. 3, pp. 218–222, Jun. 2005.

[15] G. Banerjee, M. Patwardhan, and M. Mavinkurve, "Teaching with Visualizations in Classroom Setting: Mapping Instructional Strategies to Instructional Objectives," in *Technology for Education (T4E), 2013 IEEE Fifth International Conference on*, 2013, pp. 176–183.

[16] J. Pang, "Visualization and Computer Aided Design Techniques for Teaching Computer Hardware Design Course," in *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*, 2014, p. 1.

[17] T. L. Naps *et al.*, "Exploring the Role of Visualization and Engagement in Computer Science Education," *SIGCSE Bull*, vol. 35, no. 2, pp. 131–152, Jun. 2002.

[18] G. R. Garay, A. Tchernykh, A. Y. Drozdov, S. V. Novikov, and V. E. Vladislavlev, "A VHDL-Based Modeling of Network Interface Card Buffers: Design and Teaching Methodology," in *Communications in Computer and Information Science*, vol. 595, I. Gitler and J. Klapp, Eds. Cham: Springer International Publishing, 2016, pp. 250–273.

[19] G. R. Garay, J. Ortega, A. F. Díaz, L. Corrales, and V. Alarcón-Aquino, "System performance evaluation by combining RTC and VHDL simulation: A case study on NICs," *J. Syst. Archit.*, vol. 59, no. 10, Part D, pp. 1277–1298, Nov. 2013.

[20] E. Fouh, M. Akbar, and C. A. Shaffer, "The Role of Visualization in Computer Science Education," *Comput. Sch.*, vol. 29, no. 1–2, pp. 95–117, 2012.

[21] M. Knobelsdorf, E. Isohanni, and J. Tenenberg, "The Reasons Might Be Different: Why Students and Teachers Do Not Use Visualization Tools," in *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, New York, NY, USA, 2012, pp. 1–10.

[22] A. N. Kumar, "The Effectiveness of Visualization for Learning Expression Evaluation," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, New York, NY, USA, 2015, pp. 362–367.

[23] A. K. Ziabari, R. U. Tena, D. Schaa, and D. Kaeli, "A Framework for Visualization of OpenCL Applications Execution: A Tutorial," in *Proceedings of the 3rd International Workshop on OpenCL*, New York, NY, USA, 2015, pp. 22:1–22:2.

[24] A. Shoufan, Z. Lu, and S. A. Huss, "A Web-Based Visualization and Animation Platform for Digital Logic Design," *IEEE Trans. Learn. Technol.*, vol. 8, no. 2, pp. 225–239, Apr. 2015.

[25] J. Urquiza-Fuentes and J. Á. Velázquez-Iturbide, "Toward the effective use of educational program animations: The roles of student's engagement and topic complexity," *Comput. Educ.*, vol. 67, pp. 178 – 192, 2013.

[26] I. Cetin, “Visualization: a tool for enhancing students’ concept images of basic object-oriented concepts,” *Comput. Sci. Educ.*, vol. 23, no. 1, pp. 1–23, 2013.
[27] B. Nova, J. C. Ferreira, and A. Araújo, “Tool to support computer architecture teaching and learning,” in *Engineering Education (CISPEE), 2013 1st International Conference of the Portuguese Society for*, 2013, pp. 1–8.
[28] R. DeVries, “Vygotsky, Piaget, and Education: a reciprocal assimilation of theories and educational practices,” *New Ideas Psychol.*, vol. 18, no. 2–3, pp. 187 – 213, 2000.
[29] L. Moreno, C. Gonzalez, I. Castilla, E. J. Gonzalez, and J. Sigut, “Use of Constructivism and Collaborative Teaching in an ILP Processors Course,” *IEEE Trans. Educ.*, vol. 50, no. 2, pp. 101–111, May 2007.
[30] B. S. Bloom, “Taxonomy of educational objectives: The classification of educational goals .,” *Handb. Cogn. Domain*, p. 200, 1956.
[31] T. Gingold, “Ghdl-where vhdl meets gcc,” 2010. [Online]. Available: http://ghdl.free.fr.
[32] T. Bybell, “Gtkwave.” [Online]. Available: http://gtkwave.sourceforge.net.
[33] D. Rosner, G. Sârbu, R. Tătăroiu, and R. Deaconescu, “Applied electronics curriculum for Computer Science students,” in *EUROCON - International Conference on Computer as a Tool (EUROCON), 2011 IEEE*, 2011, pp. 1–4.
[34] S. C. Karlin and L. Peterson, “Maximum Packet Rates for Full-Duplex Ethernet,” Department of Computer Science, Department of Computer Science, TR-645-02, 2002.
[35] G. R. Garay, A. Tchernykh, and A. Drozdov, “An Approach for the Performance Evaluation of Multi-tier Cloud Applications,” in *2015 International Conference on Engineering and Telecommunication (EnT)*, 2015, pp. 63–66.

Godofredo R. Garay received the B.E. degree in Computer Engineering from the Universidad Tecnológica de La Habana José Antonio Echeverría (CUJAE), Havana, Cuba, in 1994, and the Ph.D. degree in 2012, from the University of Granada, Spain. He is currently an Assistant Professor at the University of Camaguey’s Faculty of Informatics, Cuba. His research interests include studying performance bottlenecks in current computers, and performance modelling and evaluation.

Andrei Tchernykh received the Ph.D. degree from Institute of Precise Mechanics and Computer Technology of the Russian Academy of Sciences, Russia in 1986. Currently he is a Full Professor in Computer Science Department at CICESE Research Center, Ensenada, Baja California, Mexico, and a head of Parallel Computing Laboratory. He is a member of the National System of Researchers of Mexico (SNI), Level II. He leads a number of national and international research projects. He delivered 50 keynote speeches and invited lectures, served as a program committee member and general co-chair of more than 100 professional peer-reviewed professional conferences. His main interests include parallel computing, resource optimization techniques, adaptive resource provisioning, multi-objective optimization, real time systems, computational intelligence, and incomplete information processing.

Alexander Yu. Drozdov received the M.Sc. degree in mathematics in 1988 from the Moscow State University, Russia. He is currently a Full Professor at The Moscow Institute of Physics and Technology, Russia, and a head of the laboratory of design and modelling of special-purpose computer systems. His research interests are in the fields of research and development of new high-performance architectures and embedded computing systems, embedded control systems, together with the development of tools, embedded and system software.

Sergey Garichev received the M.Sc. and Ph.D. degree from Moscow Institute of Physics and Technology (State University) - MIPT. Currently he is a Dean of the Department (Faculty) of Radio Engineering and Cybernetics of MIPT, senior researcher with a specialization in Systems of design automation, head of the department "Radio engineering and control systems." His major research and educational interests are in the areas of telecommunications, radar and radio communication equipment, microprocessor and computer technology, control systems design, and application software development for special-purpose technical equipment.

Sergio Nesmachnow is a Full Time Professor at Universidad de la República, Uruguay He is Researcher at National Research and Innovation Agency (ANII) and National Program for the Development of Basic Sciences (PEDECIBA), Uruguay. His main research interests are scientific computing, high performance computing, and parallel metaheuristics applied to solve complex real-world problems. He holds a Ph.D. (2010) and a M.Sc. (2004) in Computer Science, and a degree in Engineering (2000) from Universidad de la República, Uruguay. He has published over 90 papers in international journals and conference proceedings. Currently, he works as Director of the Multidisciplinary Center for High Performance Computing (Universidad de la República, Uruguay) and as Editor-in-Chief for International Journal of Metaheuristics, while he is also Guest Editor in Cluster Computing and The Computer Journal. He also participates as speaker and member of several technical program committees of international conferences and is a reviewer for many journals and conferences.

Moisés Torres-Martinez received his Bachelors of Science in Information and Computer Science from the University of California Irvine, and Ph.D. in Educational Administration from the University of California Los Angeles and Irvine. He obtained a certificate of Management Development Program at Harvard Institute for Higher Education. He is Adjunct Professor at the University of Guadalajara, and a Senior Consultant in Hitum S.A. de C.V. He teaches technology and business administration. In Hitum, he is responsible for promoting projects with academia, government and private industry in the area of information technology. He has served in various executive positions in higher education institutions in Mexico (UdeG, IPICYT), and the United States (University of California System). He is founder of several organizations related to technology, and education. He is a recipient of awards and member of international science, engineering, technology and education societies.

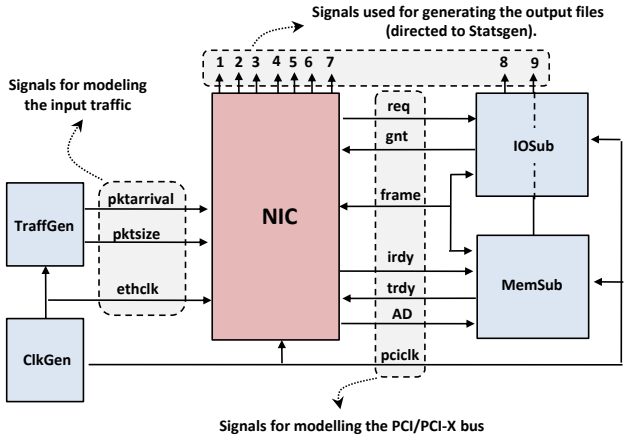

Figure 1. The functionality diagram of Network Interface Card

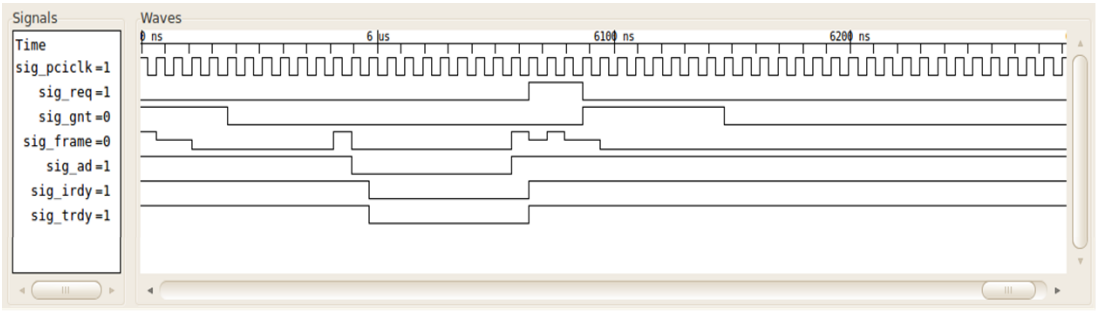

Figure 2 Visualization of PCI-bus signal waveforms.

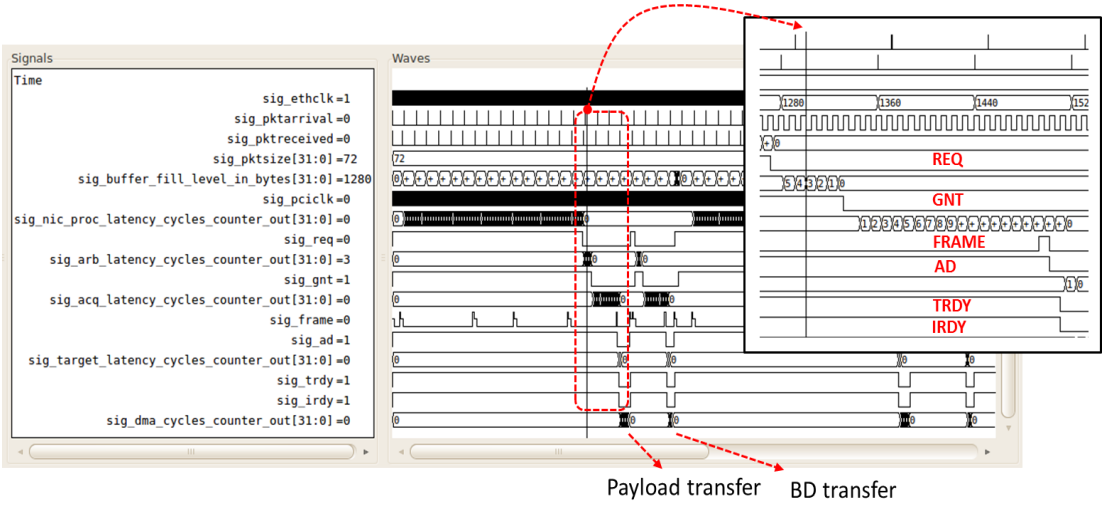

Figure 3. Visualization of overhead cycles and DMA cycles for receiving a packet.

Table 1: Sample Tasks for Bloom's Comprehension Levels

Level	What the learner can do	Sample tasks and assignments
1	- Recognize and informally define specific concepts in a network, like input/output processing and management, NIC, I/O bus, bus protocols, Ethernet networks, or basic analysis concepts such as bandwidth, latency, overhead, and throughput.	- Define the following concepts: PCI bus bandwidth and PCI throughput.
2	- Understand the general principles and essential properties of NIC, Ethernet and PCI protocols and explain how they work using words and figures. - Understand the role of the network-to-memory data path on system performance. - Understand the behavior of a network node (or its model) subjected to worst-case Ethernet traffic.	- Explain why PCI throughput decreases as the number of bus master devices attached to the bus increases.
3	- Construct the best-case and worst-case analysis of NIC-side processing, I/O bus operation, network workload.	- Demonstrate the best-case of bus latency for achieving the highest bus performance, and calculate bus throughput.
4	- Be able to analyze bottleneck detection problems on the network-to-memory data path, identify essential objects, and split it into smaller problems.	- Explain why the bus is the bottleneck for 9000-bytes input packet with a DMA burst size of 256 bytes, but not with a burst size of 4096 bytes.
5	- Design solutions to complex problems where several different data structures, algorithms and techniques are needed. - Analyze the efficiency of bus transactions for a network workload consisting of maximum-size network packets. - Set up criteria for comparing various solutions.	- Design the Finite States Machines needed for modeling the arbitration process in a PCI bus.
6	- Argue how and why DMA burst size should be tuned to avoid buffer overflow at NIC level in 10GbE networks. - Discuss the pros and cons of parallel I/O bus architecture (e.g., PCI and PCI-X) and serial link (PCI Express) that solve the same or similar problems. - Carry out an evaluation of a design or analysis.	- Compare PCI-X and PCI Express as NIC-to-System interconnect options. - Discuss the design of an experiment for measuring the I/O bus throughput.