Multimedia components for the visualization of dynamic behavior in computer architectures

Peter Marwedel, Birgit Sirocic
Dept. of Computer Science,
University of Dortmund,
44221 Dortmund, Germany
peter.marwedel@udo.edu

Abstract

Understanding modern processors requires a good knowledge of the dynamic behavior of processors. Traditional media like books use text for describing the dynamic behavior of processors. Visualization of this behavior, however, is impossible, due to the static nature of books. In this paper, we describe multimedia components for visualizing the dynamic behavior of hardware structures, called RaVi (abbreviation for the German equivalent of "computer architecture visualization"). Available RaVi components[1] include models of a microcoded MIPS architecture, of a MIPS pipeline, of scoreboarding, Tomasulo's algorithm and the MESI multiprocessor cache protocol.

1 Introduction

The presented project aims at facilitating understanding the dynamics of modern processor architectures, thereby overcoming an important limitation of books. Videos tapes and video distribution techniques have made it possible to show non-interactive media elements to students. However, video tapes have to be accepted by teachers and users on an "as-is" basis. It is not possible to use instruction streams other than those employed for the production of the video. Also, it is not possible to modify hardware structures in order to see the effect of hardware changes on the dynamic behavior. In short, videos are very inflexible and cannot provide interactiveness (except the simple type of interactiveness possible with DVDs).

Providing this interactiveness, however, is difficult, since it requires the simulation of hardware structures. This can be a challenging task which cannot be solved within the time-frame available for preparing a course. Why not just use available hardware simulators? These simulators are frequently designed for optimum simulation speed and complex design projects. Ease of use, excellent visualization and portability have normally not been top goals for simulator design. Also, powerful simulators are typically proprietary and come at high costs, preventing their widespread deployment to classrooms and into the hands of students.

Therefore, we tried to design RaVi models for the simulation-based visualization of the dynamic behavior of hardware architectures. In contrast to available models, emphasis is on visualization.

This paper is structured as follows: a short description of related work is provided in section 2. Section 3 describes the multimedia units developed so far. Section 4 discusses some design consideration regarding the availability and access-ability to various groups of students. Section 5 contains some of the results. The final section comprises a conclusion.

2 Related work

Simulators provide information about the dynamic behavior of computing systems. Plenty of simulators are available either commercially or in the form of public domain tools. They have been used for decades already. However, these simulators have hardly been designed for class room use. For such use, the limited resolution of screens must be taken into account. These days, it is also required that the simulators can be given into the hands of students. Expensive commercial simulators cannot be used for this reason. Also, feature-rich simulators requiring special training are not appropriate for this type of application. The target group for

[1]We gratefully acknowledge the funding of the RaVi-project (which is a subproject of the SIMBA-project) by the German ministry of research and development (BMBF).

this material includes first year students. It would be impossible to teach these students how to use a hardware description language: teaching the syntax of complex languages and to use tools such as Modelsim [6] would take too much time in a course which also has to cover a number of other important computer engineering subjects. The effort of generating models, for example in VHDL, should not be underestimated. Hence, most of the available simulators do not provide the required functionality. Notable exceptions include the JCachesim [1]. JCachesim is a simulator for cache architectures. However, JCachesim focusses on generating quantitative data (statistics etc.). In contrast, the work in this paper focusses of giving insight into how computer systems work.

3 Available multimedia units

3.1 Microcoded version of MIPS

Architectural models capable of executing a reasonable subset of some instruction set require a certain complexity of the model. The program counter, main memory, register file, ALU, control logic and a number of multiplexers all have to be included in the model. Otherwise, it would be impossible to demonstrate how instructions are executed. Many of these hardware components are connected. According to our observation, it is typically difficult for the students to understand how all the wires in a computer architecture are used. Also, the function of multiport memories seems to be a problem for students grown-up with von-Neumann languages.

Courses on computer architecture typically follow the sequence of Hennessy/Patterson's book for undergraduates [4]. Consequently, a microprogrammed version of the MIPS-machine is the first hardware structure which is introduced. It has to be introduced in such a way that students are able to comprehend how it works. We have therefore designed a multimedia unit highlighting the paths which are used during a certain micro-step (see fig. 1).

Just color-coding the values on the wires would lead to an abundance in color-coding and it would be difficult to find out, which of the lines are actually important. Therefore, only those paths leading to non-redundant inputs are marked. Lines printed in bold in fig. 1 correspond to the paths used in the final state of the store word instruction. The address input of memory Mem is driven by the computed effective address, as stored in temporary register T. The value stored is coming from the register file Reg. General simulators would typically not implement such a feature and would therefore create unnecessary barriers for the students.

Experimentation with this architecture is possible. For example, the contents of the register file Reg as well as the contents of the main memory Mem can be changed. A small dedicated assembler is provided such that assembly language programs can be assembled and loaded into the main memory.

Modifications of the wiring are possible, using the integrated schematic editor. A number of standard components are provided. These include multiplexers, registers, memories and ALUs. Modifications using these standard components can be done by the user without any programming. Adding new components not yet available in the library requires programming the behavior of these components in Java, however. This possibility is rarely used, except by the designer of the multimedia units.

3.2 MIPS-Pipeline

The operation of the MIPS-pipeline is described in the book by Hennessy and Patterson [4]. Several pages of the book are used for showing the different states the pipeline can be in. Nevertheless, this technique for explaining the operation of the pipeline has its limits: it is difficult to imagine, which situations arise for other code sequences. This is especially true for stall cycles. From available descriptions, it is difficult to understand which of the pipeline stages are stalled when.

Furthermore, it would be nice to use interactive elements in education. For example, students can be motivated to think about the behavior of the architectures by letting them "play around" with it.

All this is possible with RaVi models of the pipeline. There are essentially three models:

- The first model is a simple model without any bypassing. It can be used to demonstrate the wrong implementation of the instruction set.
- The second model includes bypassing, but does not have a separate adder for branches. This model can be used for demonstrating the advantage of bypassing.

  Also, this model implements two phase clocking. Using appropriate color coding, it can be shown that the register file is updated as a result of falling clock edges.

  The same model can be used to demonstrate the problems with branches if no special comparator for the instruction fetch stage and no special adder for calculating branch target addresses are added. Large branch delay penalties can be shown.

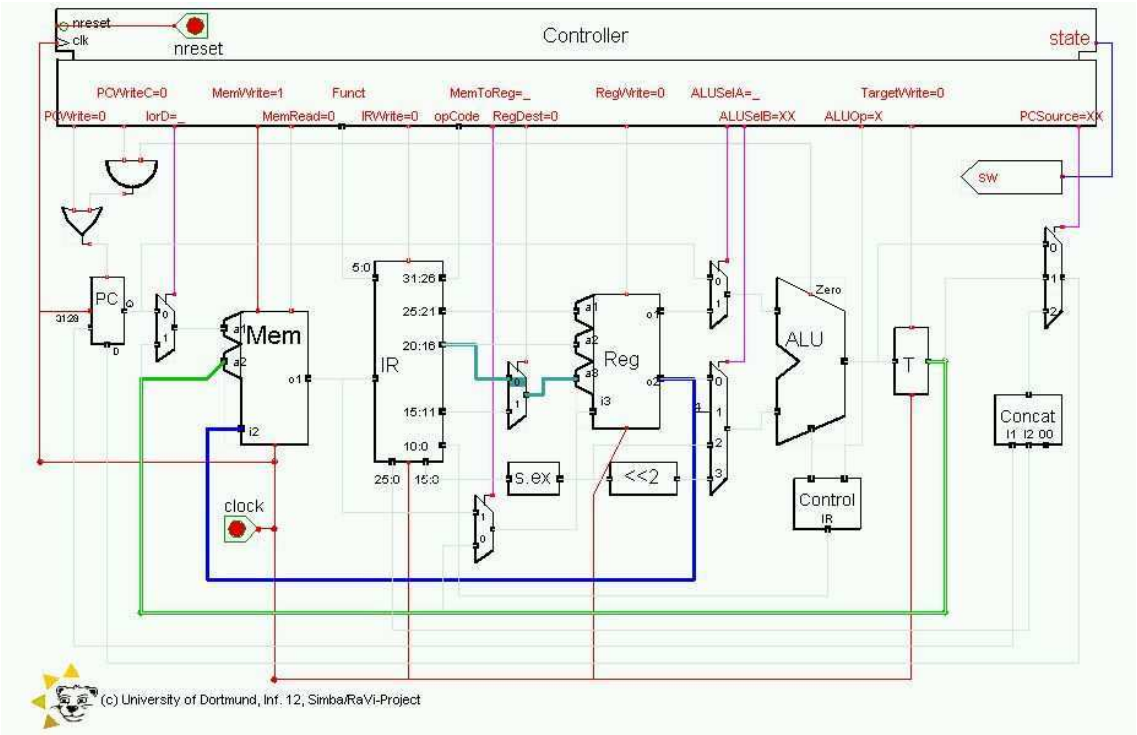

Figure 1. Microprogrammed version of the MIPS machine (segment of a screenshot)

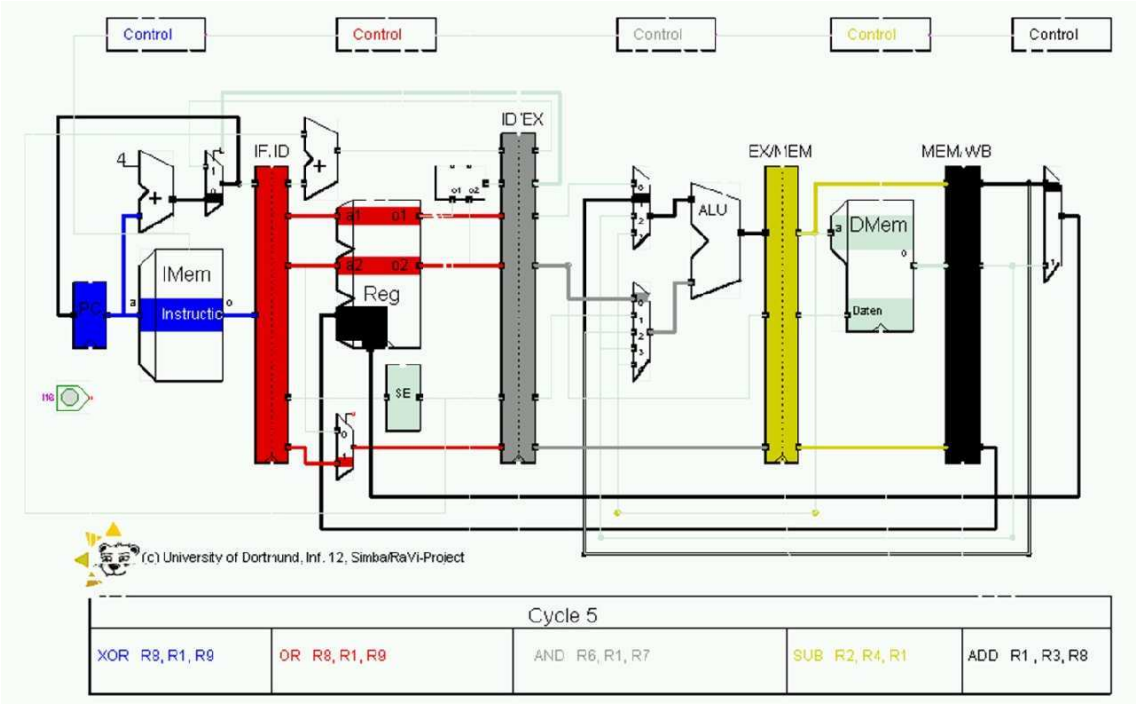

Figure 2. Segment from screen-shot from pipeline unit

- The third model includes the special hardware circuits for reducing branch delay penalties (see fig. 2).

All instructions are color-coded so that it is easy to see how instructions propagate down the pipeline. Implemented models support all major opcodes as well as mi-

nor opcodes in the register-to-register class (major opcode 0). A full implementation of all opcodes as well as exception handling is not consistent with the goal of keeping things simple so that students understand the models. Accordingly, function registers (like EPC) are not implemented. The same applies for special registers HI and LO. Irregular multiply instructions leaving their results in these registers (e.g. mult) have been replaced by their more regular pseudo instruction counterparts supported by the MIPS assembler (e.g. mul). Otherwise, too many hardware components would have to be on the screen.

3.3 MESI-protocol for a single cache block

The MESI protocol is typically included in the education of computer scientist in their third year. According to this protocol, single read requests for certain addresses cause the corresponding cache line to be in the exclusive state. Subsequent reads by other processors will cause the same cache line to be in the shared state. Writes in one processor will set the state in other caches to invalid. Due to its distributed nature, it is more difficult to understand than algorithms for mono-processors. We have therefore developed two multimedia units helping students to understand this protocol. The first unit shows the behavior of just a single cache block, of which copies may be available at four different machines (see fig. 3). The three state finite state machine used by Hennessy/Patterson [5] is replaced by the commonly used 4-state FSM.

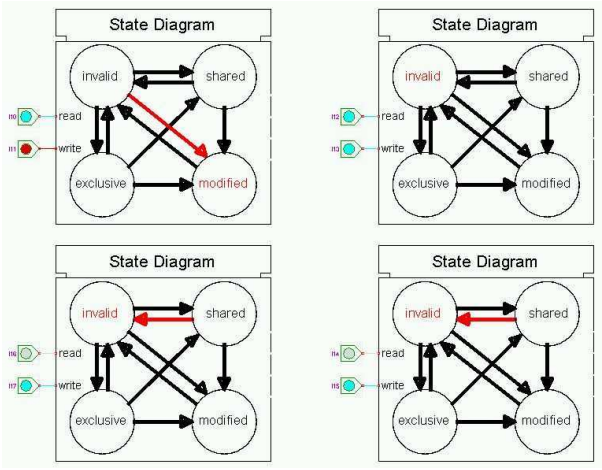

Figure 3. RaVi visualization of the MESI protocol for a single block

By generating read and write requests, the lecturer or the student can explain the behavior of these finite state machines. We found that students realized much faster that, once the shared state is reached, there is no way back to the exclusive state (in hardware, there is usually no signal which would allow going back to state exclusive except through state invalid).

3.4 MESI-protocol for the entire cache

After demonstrating the behavior of the four state MESI FSM for a single block, we are typically explaining the full MESI model for a number of cache blocks, also including tag bits. Fig. 4 shows a screen-shot of that model. Read and write requests can be generated interactively. Addresses and data for all read and write requests can be changed by using the context menue of the processors (shown at the top).

We found that students were surprised about the behavior of that model in case the same index bits but different tag bits are used in accesses to the different caches. Also, students did not expect the complexity of the operations on the bus.

3.5 Scoreboarding

Scoreboarding is known as one of the early techniques for increasing processor speeds. Due to the distributed nature of the algorithm, we found that students had problems with understanding the algorithm exactly. In order to change this situation, we have developed a multimedia unit for this algorithm as well. In order to let students make experiments with the model, different instruction streams can be used and the effect of the resulting parallelism can be studied. Fig. 5 shows a screen-shot.

We found that the resolution of currently available projection equipment puts a tight constraint on the level of detail that can be shown for this algorithm.

3.6 Tomasulo algorithm

The Tomasulo algorithm is a more advanced algorithm for speeding up processor architectures. The Tomasulo algorithm employs a more decentralized control, making it even more difficult to understand the overall behavior. The corresponding RaVi unit avoids this problem. Again, the students can “play” around with different instruction streams and observe the behavior of the architecture. Functional components can be deleted by the user (lecturer or student) and new components can be added. No programming is required as long as standard components are added.

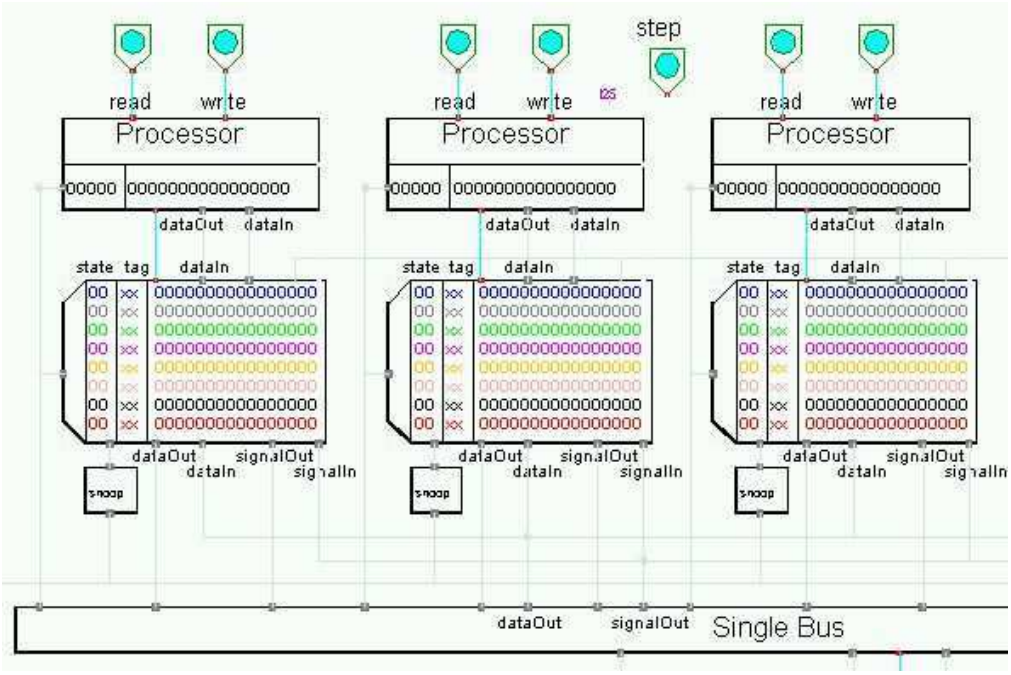

Figure 4. Segment of a screen-shot from RaVi cache protocol unit (Memory omitted)

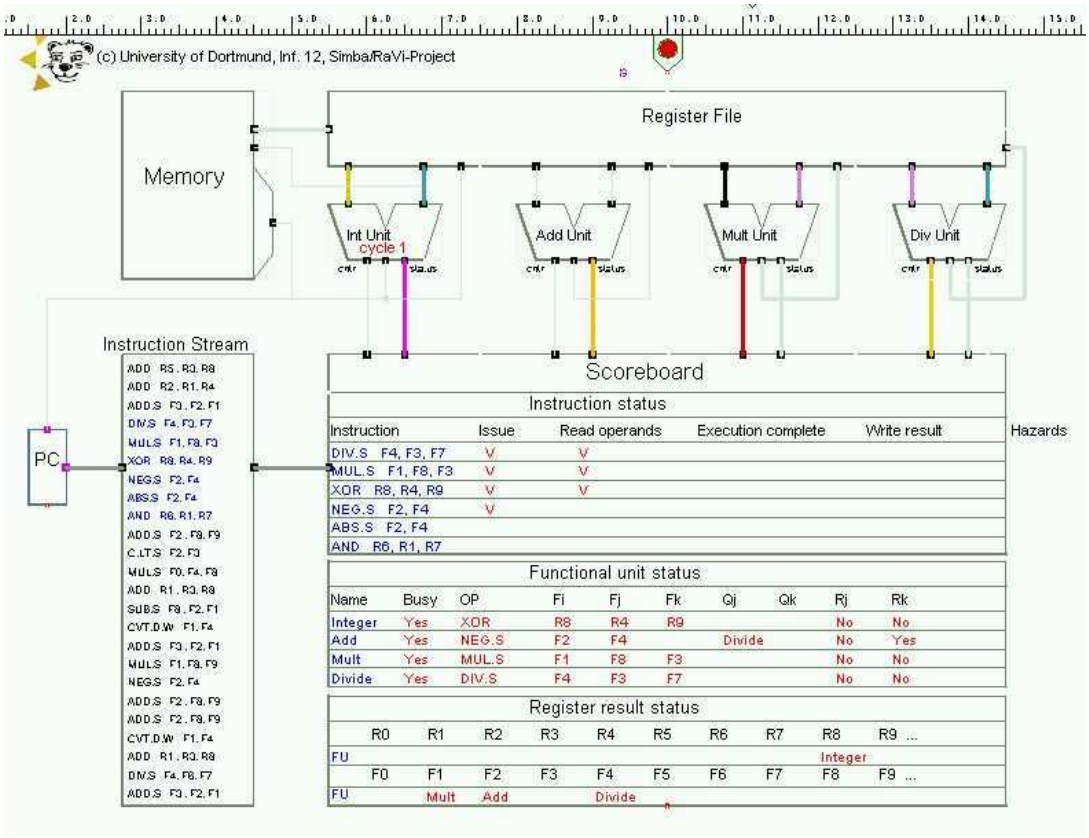

Figure 5. Screen-shot from RaVi scoreboard unit

4 Implementation aspects

4.1 Availability

The RaVi system is built on top of the HADES visualization framework for computer structures [3]. HADES is implemented in Java. The entire RaVi model follows the object-oriented paradigm. Every RaVi component is an instance of the corresponding hardware component class.

Due to being implemented in Java, RaVi can be used at a variety of platforms. We decided to make RaVi freely available on the Internet in order to promote its use. Initial versions of RaVi required a download of the software. Current versions are available as an applet and can be used without any software installation effort (provided Java is already installed). RaVi is available from //ls12.cs.uni-dortmund.de/ravi.

4.2 Gender-specifi caspects

One of the goals of RaVi is to motivate also female students to study computer engineering. A number of considerations (see e.g. Fisher et al. [2]) have been taken into account during the design of RaVi:

- Before enterering the University, women typically have less hands-on-experience with computers in general and with computer engineering in particular, compared to most men. Therefore, a very careful definition of all technical terms must be used in the accompanying technical material.
- Educational material should avoid unjustified stereotypic views of computer users. For example, female computer users also include scientists and not only secretaries (in contrast, for example, to the cliparts provided by Microsoft).

4.3 Limitations

Simulation in the underlying HADES library is based on a VHDL-like two-phase simulation of synchronous architectures. Communication is based on explicit interconnections (which can be hidden on the screen). Simulation is less suited for applications in which explicit interconnections are difficult to use. Nevertheless, it was possible to use this simulation approach for demonstrating search in binary trees. Visualization is focussing on 2D models. 3D models are beyond the scope of the current approach.

5 Results

The RaVi project led to several results:

- We found that the generation of the multimedia units required significantly more time than expected. Due to using HADES, first versions could be designed rather quickly, requiring production efforts of a few weeks at most. However, the use of these units in the classroom led to requirements for improving the units. Only almost perfect units can be used in the classroom environments and given into the hands of students. Fine tuning of the units required as much work as their original design. A total of about 2 person years have been spent on the project so far.
- It is good scientific practice to try to measure by how much the quality of teaching can be improved by using the multimedia units. Following the advice by researchers from social sciences, we tried to get quantitative information on the level of understanding achieved through the use of these units. Even though we had two large groups of students (about 200 each) which could be compared, no quantitative conclusions could be drawn. A number of other effects (date and time of the teaching, characteristics of the students etc.) resulted in a wide variation of the results and prevented any meaningful conclusions. According to more recent advice from an expert in the area [7], attempts to quantitatively measure the effect of multimedia-based education are in fact bound to fail and a waste of time. One cannot expect more than just qualitative information on the improvements achieved. According to this qualitative information, the goals of the project have been reached.
- Students really like the presented units and appreciate their availability. They are typically highly motivated trying out these units at home and ask for download options. Also, colleagues typically comment very positively on the availability of these units. The most important argument is the added value of the units. While online-versions of static material provide only limited added value, if compared to books, visualization of dynamic properties adds a completely new quality.
- Visualization of the dynamic behavior has proven being indeed one of the key technologies for improving the teaching further and for exploiting modern equipment.
- Simulation based on the HADES simulation framework was found to be appropriate for various kinds

of digital circuits. While is was possible to use HADES for visualizing algorithms like tree-search, it is less appropriate of analog and time-continuous simulations. Simulation speed is sufficient even in applet-based versions of RaVi.

6 Conclusion

In the RaVi project, we have demonstrated how a deficiency of classical media for teaching computer architecture can be removed. We have shown that the visualization of computer architecture dynamics is appreciated by the students and helps them to understand the subjects. In general, RaVi units seem to improve the motivation of students. Unfortunately, it seems to be impossible to measure the effect of the new teaching aids on the student's success. In the future, we will be extended to approach to other areas of computer engineering. For example, we have started designing similar material to complement a book on embedded system design, which is currently being written at Dortmund.

References

[1] I. Branovic, R. Giargi, and A. Prete. Web-based training on computer architecture: The case of jcachesim. *Proceedings of the workshop on computer architecture education*, pages 56–60, 2002.

[2] A. Fisher and J. Mangolis. Unlocking the clubhouse. *SIGCSE bulletin, Vol. 34, no. 2, Women and Computing*, pages 79–83, 2002.

[3] N. Hendrich. A Java-based framework for simulation and teaching. *Proceedings of the 3rd European Workshop on Microelectronics Education*, pages 285–288, 2000.

[4] J. L. Hennessy and D. A. Patterson. *Computer Organization – The Hardware/Software Interface*. Morgan Kaufmann Publishers Inc., 1995.

[5] J. L. Hennessy and D. A. Patterson. *Computer Architecture – A Quantitative Approach*. Morgan Kaufmann Publishers Inc., 1996.

[6] Model Technology. home page. *//www.model.com*, 2003.

[7] C. Moreau. Universite de Compiegne. *Oral communication*, 2003.