Volume 15, Number 1, 2013 © WIETE 2013

Global Journal of Engineering Education

Transfer of innovation - using research tools for engineering education

Carlos Rioja del Río†, José María Rodriguez Corral†, Antón Civit Balcells‡, Arturo Morgado Estévez† & Fernando Pérez Peña†

University of Cádiz, Cádiz, Spain†
University of Seville, Seville, Spain‡

ABSTRACT: In this article, the authors show how, in the pursuit of research results, they can obtain excellent tools and data for engineering education. In particular, they describe one example of computer architecture in the Computer Engineering Degree Programme at the University of Cádiz in Spain. This research topic is of particular importance as it influences the execution of a range of the computer's I/O operations, due to operations of peripherals and information devices, and determines processor performance. The simulator used in research and teaching in several engineering degree programmes at the University of Cádiz is also demonstrated in this article.

Keywords: Simulation, hardware, engineering, TOI

INTRODUCTION

The advancement of processor architecture is very important and it has moved on from an experimental stage to full implementation in engineering practice in just a few years. From sequential instructions to execution processors, the developers have passed to processors in the pipeline [1] and from this to superscalar ones, the super-pipeline [2][3] and VLIW (Very Long Instruction Word) [4-6].

Currently, there are several fields of investigation dedicated to the design of complex single-processor architectures [SIL 00]. In these, one can include trace processors, multi-scalar processors and data-scalars. The trace cache saves the program's execution derived traces, and the processors search for instructions in the trace cache instead of doing it in the instruction cache. The trace and multi-scalar processors define several of the elements of processing that execute different parts of a sequential program speculatively and in parallel. The difference rests in the fact that the last ones use a compiler to partition the program in segments, while the first ones use trace cache to generate trace segments dynamically for processing elements. Finally, the data-scalar processors execute the same sequential program in a redundant way in various process elements, where each one of these uses a different data set.

To realise this objective, it is necessary to develop a program for simulating a current characteristics processor (processor - main memory subsystem simulator program), which will be configured with a set of typical parameters for each type of a processor. Likewise, it is necessary to design a set of I/O scenarios which, together with mentioned parameters, will be used as input data for different simulations. Once the code fragment has selected which execution is going to be simulated, each result will be expressed in a function of two variables: the concrete type of processor and the selected I/O scenario.

In this way, the I/O subsystem simulator will generate the blockade file from the mentioned scenarios, corresponding to each scenario. With each blockade file, a different execution time will be obtained for the same processor and the same code fragment, for which execution is simulated. Finally, relating the time of execution by instructions from the code instance in each scenario with the time execution by instruction in code instance in ideal conditions, one will obtain the deceleration of simulated processor in each scenario, from which one will extract the corresponding conclusions. These experiments will be repeated for each kind of processor.

What most important is in this article, is the *discovery* of several different applications using both simulation tools. Most of them are closely related to pipeline execution teaching, computer architecture, multitasking and computer performance. The general simulator has been used in several subjects and seminars carried out at the Faculty of Engineering at the University of Cádiz.

SIMULATOR DESIGN

It is true that there are other modern tools that have been designed to simulate processors, which possess a great number of interesting characteristics and features, such as SimpleScalar [7][8], HASE [9], RSIM [10], POWER RISC [11] and CASTLE [12]. Many of these are provided with an interactive interface and are more user-friendly. These tools are relatively recent and were not available when simulator development began.

In addition, many of these simulators do not allow the inclusion of processor external elements in their models. The ones that do permit it (such as SimpleScalar, HASE and RSIM) do not consider the peripheral modelling of the behaviour the authors are interested in (that is, by simulating for determining the influence of input/output operations that these do on the system processor performance), which constitutes the objective of this study. Finally, DESP-C++ accomplishes important number of requirements that are relevant at the moment; the authors have chosen a simulator or simulating language [13] and that, definitely, advise its utilisation:

- The ease of use and flexibility. DESP-C++ consists fundamentally of data structures and a set of routines or functions written in the easy-to-use C++ language, which manipulate the mentioned data. These routines are integrated in a C++ program written by the user. In this way, it is possible to develop a simulator adjusted as much as possible to the needs. Also, having these routines as source code, it is possible to modify them in case one needs to do so.
- Transportability, due to the fact that, as a last resort, what one obtains is a program written in C++ language. This can be transported to any computer provided with an operating system that includes a C++ compiler.
- Incorporates a utility for obtaining traces from simulations in an interactive operation mode that allows for the detection of errors that can appear during the simulation.
- Random number generator that can be initialised with different seeds values.
- Statistical ease. Includes necessary code for generating values for common distribution probability functions (normal, exponential, uniform, etc). It also allows for calculating the intervals of confidence susceptible of use for understanding the statistical significance of the results obtained.
- Generation of independent replication (with a different set of numbers for each).
- Reports generation: utilisation grade of resources that took part in a simulation, total time consumption, etc.

Once the tool is selected, the type of processor, whose modelling is going to be implanted in the simulator, can be selected. The characteristics are as follow:

- RISC processor, with three types of instructions: arithmetical-logical, load and save, and jump (conditional and unconditional).
- Super-scalar with a slide window. This way, after emitting those instructions that do not present dependencies with the data, they are sent to corresponding functional units (arithmetical/logical or memory access). If some have a data dependency respecting the issued ones or if a structural blockade exists that prevents its issue, this instruction will pass to be the first in the next issue (when the blockade disappears) and the partially empty window due to already emitted instructions is refilled with new instructions till the completion of the grade of super-scalar.
- Data and instruction separated caches. Data cache permits simultaneous accesses and is resistant to blockades [4][14], so the cache miss in one line does not block the rest of the accesses.
- Arithmetic integer and float functional units. With the goal of easing the development of the simulator, as much as possible, without having to lose the validity in the results obtained because of it, it was decided not to use segmented functional units. In this way, the need for *n* stages can be simulated using *n* segmented functional units of *n* latency.
- Branch target buffer that can be activated or not for each concrete simulation [15-18].

Further, once the simulator is running, it asks for the following data:

- Text file name that contains the program fragment whose execution will be simulated.
- Text file name that contains the blockade trace of primary memory due to the input/output operations realised by corresponding peripherals for a concrete scenario. Aspects relative to input/output scenarios and blockade files generated from them will be dealt with in the next section.
- Text file name that stores emission rules for instructions, in which it is implied the super-scalar grade of the processor.
- Primary memory access time.
- Seed for generating random numbers.
- Probability that a condition jump occurs. If the introduced value is a unit, than the simulator asks for the number of instructions (number of samples) to consider in the simulation.

- Hit probability in instruction and data cache.
- Length of instruction cache line. This last value serves for knowing, given the instructions start direction, the cache line this belongs. That is used to provide higher realism to the corresponding module of instructions cache, which by virtue of a static variable is capable of recording the corresponding cache line of the last access, without mattering if it was a cache miss or hit.
- Normal distribution function mean and standard deviation in charge of generation line numbers corresponding to data cache.
- The most important thing is to choose between static scheduling and dynamic scheduling, including the number of the Reservation Station. (This is one of the concepts shared with the students of last year´s degree in Computer Science Engineering).
- Consecutive miss number that data cache can suffer without blocking.
- Instruction queue size, expressed as a maximum number of instructions that can be stored.
- Simultaneous search of instructions number in instructions cache and simultaneous writings number that can be made over a processor's registry file.
- Functional units number and latency corresponding to the following types: integer (EX), float sum and subtraction (ADDF), integer and float product (MULF), integer and float division (DIVF) and memory access (MEM). Logically, the number of units of the last type is the number of simultaneous accesses that data cache is capable of bearing.

The authors have worked with four different scenarios and processors:

- Video capture (CAPT).
- Video capture with reproduction (CAPT&RE).
- High speed network (RED).
- Serial transmission (UART).
- Two grade processor with unlimited resources:

    - Six floating point sum units (6 ADDF). Twelve floating point product units (12 MULF). Four memory access units (4 MEM). Two integer units (2 EX).

1. Two grade processor with limited resources:

    - Three floating point sum units (3 ADDF). Six floating point product units (6 MULF). Two memory access units (2 MEM). One integer unit (1 EX).

2. Four grade processor with unlimited resources:

    - Twelve floating point sum units (12 ADDF). Twenty-four floating point product units (24 MULF). Eight memory access units (8 MEM). Four integer units (4 EX).

- Four grade processor with limited resources:

    - Six floating point sum units (6 ADDF). Twelve floating point product units (12 MULF). Four memory access units (4 MEM). Two integer units (4 EX).

RESULTS

Suppose the authors have X1, X2, …, Xn as results obtained from various experiments of the simulation. The parameters that would normally be of interest are the mean (Equation (1)) and variance (Equation (3)):

$$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} Xi \tag{1}$$

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(Xi-\overline{X})^2 \tag{2}$$

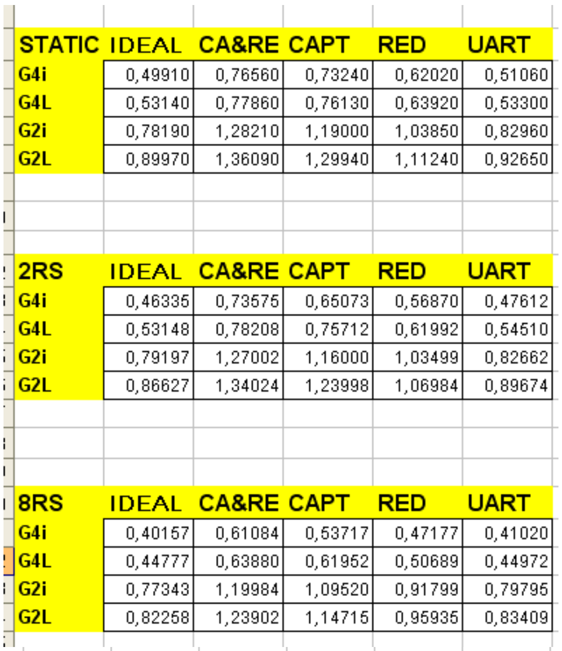
If the value of N is high enough, the previous equations results are approximately the same as really important parameters: the expected value of X (Equation (3)) and its second moment:

$$\mu = \lim_{n\to\infty} E(X_n) \tag{3}$$

Since the X_i values are normally distributed when N is high enough, the confidence interval for the estimated value $\overline{X}$ is defined by Equation (4), where $t_{\alpha/2,\, n-1}$ is the value that leaves ($\alpha/2$ x 100) percent of the distribution area t of student on the left. That way it is fulfilled, P [$\overline{X}$ - H ≤ μ ≤ $\overline{X}$ + H] = 1 - α, where 1 - α is the confidence interval level:

$$H = t_{\alpha/2,N-1} \frac{S}{N^{1/2}} \tag{4}$$

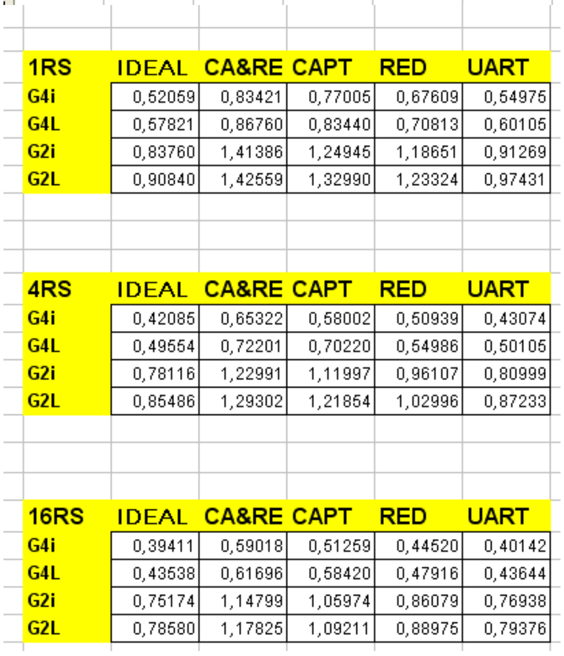
Here is the cycle per instruction for the experiments, static scheduling and dynamic scheduling with 1, 2, 4, 8 and 16 Reservation Stations:



STATIC	IDEAL	CA&RE	CAPT	RED	UART
G4i	0,49910	0,76560	0,73240	0,62020	0,51060
G4L	0,53140	0,77860	0,76130	0,63920	0,53300
G2i	0,78190	1,28210	1,19000	1,03850	0,82960
G2L	0,89970	1,36090	1,29940	1,11240	0,92650

2RS	IDEAL	CA&RE	CAPT	RED	UART
G4i	0,46335	0,73575	0,65073	0,56870	0,47612
G4L	0,53148	0,78208	0,75712	0,61992	0,54510
G2i	0,79197	1,27002	1,16000	1,03499	0,82662
G2L	0,86627	1,34024	1,23998	1,06984	0,89674

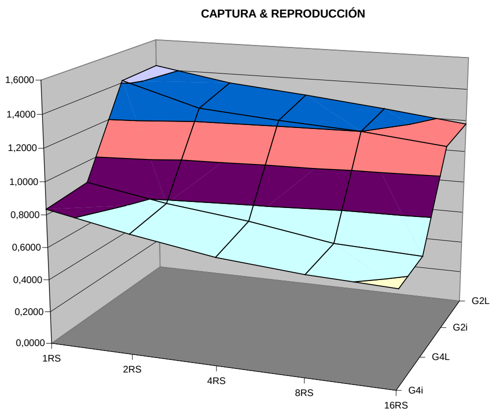
8RS	IDEAL	CA&RE	CAPT	RED	UART
G4i	0,40157	0,61084	0,53717	0,47177	0,41020
G4L	0,44777	0,63880	0,61952	0,50689	0,44972
G2i	0,77343	1,19984	1,09520	0,91799	0,79795
G2L	0,82258	1,23902	1,14715	0,95935	0,83409

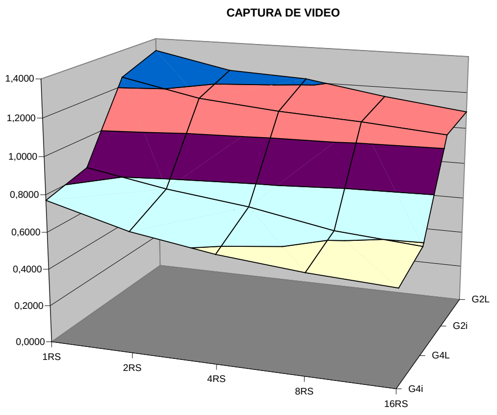


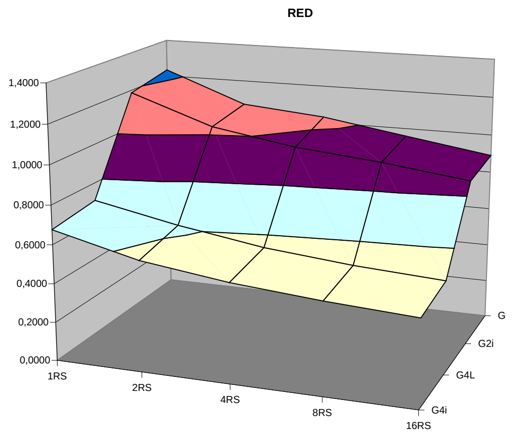
1RS	IDEAL	CA&RE	CAPT	RED	UART
G4i	0,52059	0,83421	0,77005	0,67609	0,54975
G4L	0,57821	0,86760	0,83440	0,70813	0,60105
G2i	0,83760	1,41386	1,24945	1,18651	0,91269
G2L	0,90840	1,42559	1,32990	1,23324	0,97431

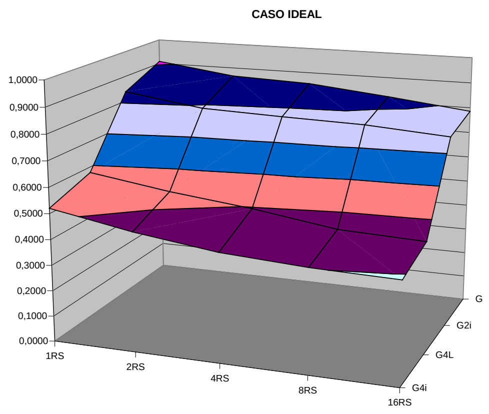
4RS	IDEAL	CA&RE	CAPT	RED	UART
G4i	0,42085	0,65322	0,58002	0,50939	0,43074
G4L	0,49554	0,72201	0,70220	0,54986	0,50105
G2i	0,78116	1,22991	1,11997	0,96107	0,80999
G2L	0,85486	1,29302	1,21854	1,02996	0,87233

16RS	IDEAL	CA&RE	CAPT	RED	UART
G4i	0,39411	0,59018	0,51259	0,44520	0,40142
G4L	0,43538	0,61696	0,58420	0,47916	0,43644
G2i	0,75174	1,14799	1,05974	0,86079	0,76938
G2L	0,78580	1,17825	1,09211	0,88975	0,79376









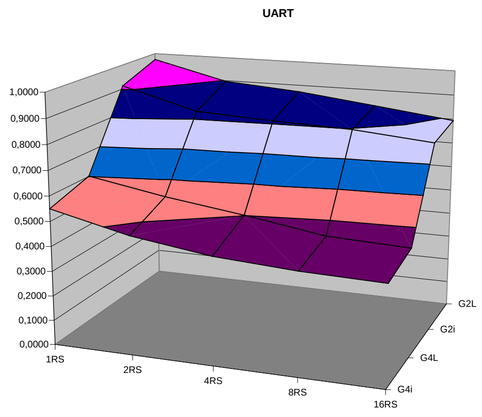
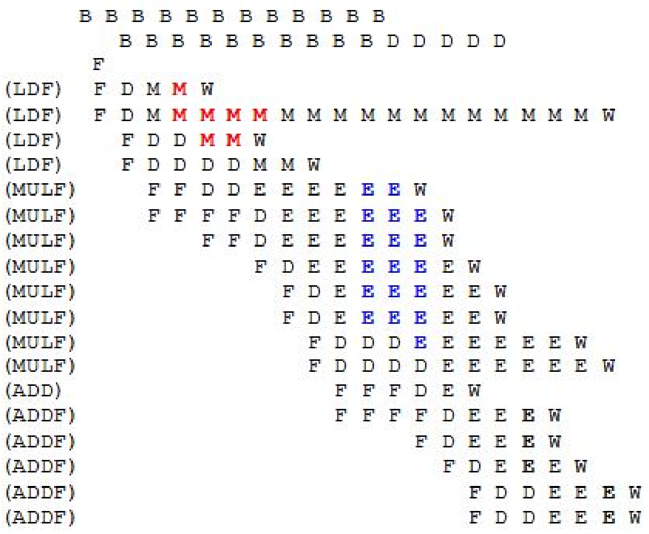


Figure 1: The cycle per instruction for the experiments.

Using interpolation and regression, the authors obtained equations to handle the I/O impact in processors' performance. It is intended to establish the way to develop specific-purpose PCs. The authors can use that information for the desired balance between last generations-compiler and hardware complexity.

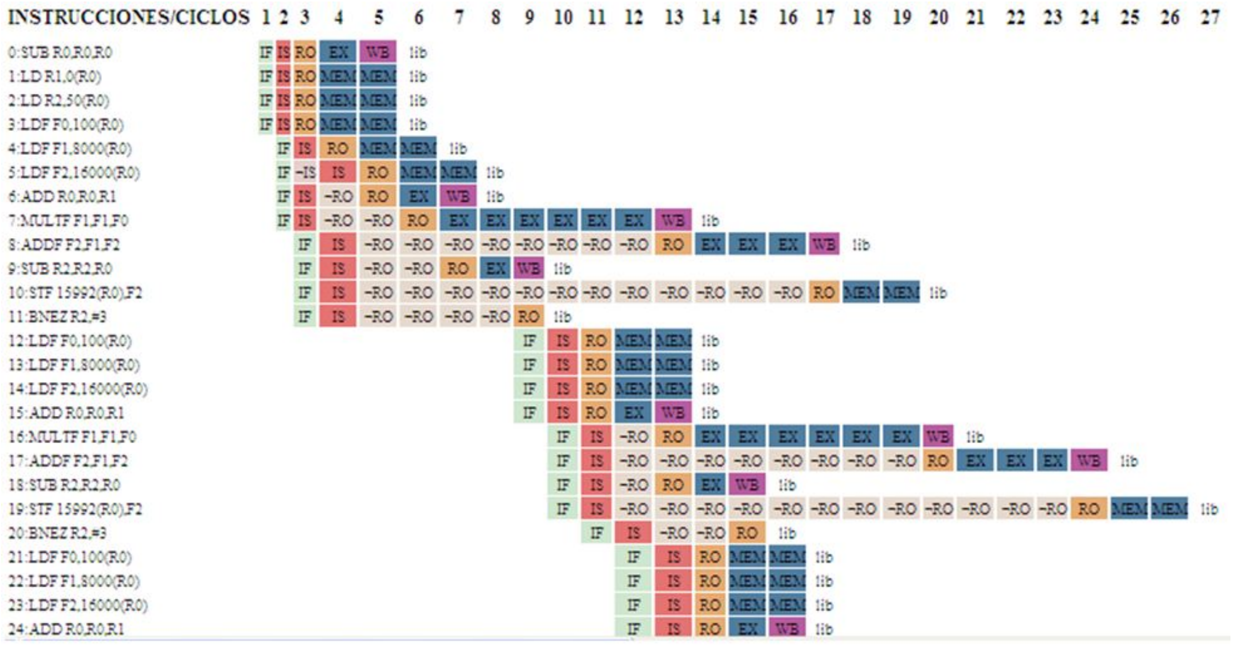
TRANSFER OF INNOVATION - SIMULATOR REVISITED

During the development of the simulator, and much more after the results were used, the tool was revisited and customised for the students of the Faculty of Engineering, and several utilities were added. For example, a complete graphic set of instructions. Figure 2a and Figure 2b show a static versus dynamic scheduling example.



```
          B B B B B B B B B B B B
             B B B B B B B B B B D D D D D
           F
(LDF)      F D M M W
(LDF)      F D M M M M M M M M M M M M M M M M M M W
(LDF)        F D D M M W
(LDF)        F D D D D M M W
(MULF)         F F D D E E E E E E W
(MULF)         F F F F D E E E E E E W
(MULF)             F F D E E E E E E W
(MULF)                 F D E E E E E E W
(MULF)                   F D E E E E E E W
(MULF)                   F D E E E E E E W
(MULF)                     F D D D E E E E E E W
(MULF)                     F D D D D E E E E E E W
(ADD)                        F F F D E W
(ADDF)                       F F F F D E E E W
(ADDF)                             F D E E E W
(ADDF)                               F D E E E W
(ADDF)                                 F D D E E E W
(ADDF)                                 F D D E E E W
```

a)



INSTRUCCIONES/CICLOS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0:SUB R0,R0,R0	IF	IS	RO	EX	WB	lib																					
1:LD R1,0(R0)	IF	IS	RO	MEM	MEM	lib																					
2:LD R2,50(R0)	IF	IS	RO	MEM	MEM	lib																					
3:LDF F0,100(R0)	IF	IS	RO	MEM	MEM	lib																					
4:LDF F1,8000(R0)		IF	IS	RO	MEM	MEM	lib																				
5:LDF F2,16000(R0)		IF	-IS	IS	RO	MEM	MEM	lib																			
6:ADD R0,R0,R1		IF	IS	-RO	RO	EX	WB	lib																			
7:MULTF F1,F1,F0		IF	IS	-RO	-RO	RO	EX	EX	EX	EX	EX	EX	WB	lib													
8:ADDF F2,F1,F2			IF	IS	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	RO	EX	EX	EX	WB	lib									
9:SUB R2,R2,R0			IF	IS	-RO	-RO	RO	EX	WB	lib																	
10:STF 15992(R0),F2			IF	IS	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	RO	MEM	MEM	lib							
11:BNEZ R2,#3			IF	IS	-RO	-RO	-RO	-RO	RO	lib																	
12:LDF F0,100(R0)									IF	IS	RO	MEM	MEM	lib													
13:LDF F1,8000(R0)									IF	IS	RO	MEM	MEM	lib													
14:LDF F2,16000(R0)									IF	IS	RO	MEM	MEM	lib													
15:ADD R0,R0,R1									IF	IS	RO	EX	WB	lib													
16:MULTF F1,F1,F0										IF	IS	-RO	RO	EX	EX	EX	EX	EX	EX	WB	lib						
17:ADDF F2,F1,F2										IF	IS	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	RO	EX	EX	EX	WB	lib		
18:SUB R2,R2,R0										IF	IS	-RO	RO	EX	WB	lib											
19:STF 15992(R0),F2										IF	IS	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	-RO	RO	MEM	MEM	lib
20:BNEZ R2,#3											IF	IS	-RO	-RO	RO	lib											
21:LDF F0,100(R0)												IF	IS	RO	MEM	MEM	lib										
22:LDF F1,8000(R0)												IF	IS	RO	MEM	MEM	lib										
23:LDF F2,16000(R0)												IF	IS	RO	MEM	MEM	lib										
24:ADD R0,R0,R1												IF	IS	RO	EX	WB	lib										

b)

Figure 2: A static versus dynamic scheduling example.

FINAL RESULTS AND FUTURE

Using simulation as a tool, but also as a goal, the authors have been working with several topics of considerable interest to the students, such as superscalars processors, multitasking, multiprocessing, scheduling and even specific-purpose architecture. Due to the nature of the general simulator, the authors can develop a set of instructions and a computer architecture according to a concrete task or way of execution. In fact, this is one of the areas of collaboration with research groups at other universities.

New scenarios and new devices are possible and nowadays their development is mandatory This will help students in the development of specific purpose machines, and customise the tool for Smartphones, Pads, Galaxies.

REFERENCES

1. Stone, H.S., *High-Performance Computer Architecture*. (2nd Edn), Addison-Wesley Publishing Company (1990).
2. Hennessy, J.L. and Jouppi, N.P., Computer technology and architecture. An evolving interaction. *Computer*, 24, **9** (1991).
3. Stone, H.S. and Cocke, J., Computer architecture in the 1990s. *Computer*, 24, **9** (1991).
4. Hennessy, J.L. and Patterson D.A., *Computer Architecture. A Quantitative Approach*. (2nd Edn), Morgan Kaufmann Publishers, Inc. (1996).
5. Lenell, J. and Bagherzadeh, N., A performance comparison of several superscalar processor models with a VLIW processor. *Microprocessors and Microsystems*, 18, **3** (1994).
6. Moreno, J.H., Moudgill, M., Ebcioglu, K., Altman, E., Hall, C.B., Miranda, R., Chen, S.-K. and Polyak, A., Simulation/evaluation environment for a VLIW processor architecture. *IBM J. of Research and Develop.*, 41, **3** (1997).
7. Burger, D. and Austin, T.M., The SimpleScalar Tool Set, Version 2.0. Department of Computer Sciences. University of Wisconsin-Madison. Technical Report 1342, June (1997).
8. Sankaralingam, K., Nagarajan, R., Keckler, S.W. and Burger, D., SimpleScalar Simulation of the PowerPC Instruction Set Architecture. Department of Computer Sciences. The University of Texas at Austin. Technical Report TR2000-04. February (2001).
9. Ibbet, R.N., Computer architecture visualisation techniques. *Microprocessors and Microsystems*, 23, **5** (1999).
10. Pai, V., Ranganathan, P. and Adve, S., RSIM: an execution-driven simulation for ILP-based shared memory multiprocessors and uniprocessors. *Proc. IEEE Third Annual Workshop on Computer Architecture Educ. at HPCA 3*, Texas, USA (1997).
11. Starostenko, O., Sánchez, A. and Lobato, S., Simulation facilities for Risc processors data flow and performance optimizations. *Proc. 21st Inter. Conf. on Computers and Industrial Engng.*, 33, **1-2** (1997).
12. Zhang, Y. and Adams, C.B., An interactive visual simulator for the DLX Pipeline. *Proc. IEEE Third Annual Workshop on Computer Architecture Educ. at HPCA 3*, Texas, USA (1997).
13. Nikoukaran, J. and Paul, R.J., Software selection for simulation in manufacturing: a review. *Simulation Practice and Theory*, 7, **1**, 1-14 (1999).
14. Yeager, K.C., The Mips R10000 Superscalar Microprocessor. *IEEE Micro*, 16, **2** (1996).
15. Fagin, B., Partial resolution in branch target buffers. *IEEE Trans. on Computers*, 46, **10** (1997).
16. Smith, A. and Lee, J., Branch prediction strategies and branch target buffer design. *Computer*, 17, **1** (1984).
17. Perleberg, C. and A. Smith, A., Branch target buffer design and optimization. *IEEE Trans. on Computers*, 42, **4** (1993).
18. Uchiyama, K., Arakawa, F., Narita, S., Aoki, H., Kawasaki, I., Matsui, S., Yamamoto, M., Nakagawa, N. and Kudo, I., The Gmicro/500 superscalar microprocessor with branch buffers. *IEEE Micro*, 13, **5** (1993).

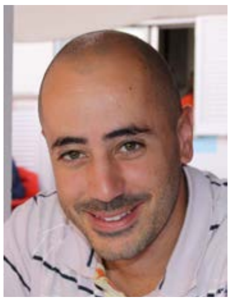
BIOGRAPHIES

Professor Dr Carlos Rioja del Río is currently Vice-Dean of the Faculty of Engineering, responsible of Institutional Agreements and Social Projection. He has been working in engineering education and international cooperation for the last 10 years. He is member of the Research Group TCP90, Applied Robotics, and the coordinator of some international projects, including specific collaboration with Universität Koblenz-Landau in Germany (Double Degree in Engineering) and Princess Sumaya University for Technology in Jordan (Quality Assessment). Prof. Rioja has presented lectures at the following universities: University of California: Berkeley, Jyväskylän yiliopisto (Finland), University of London Royal Holoway, Université Liege (Belgique). Allesund University (Norway), Université 7 Novembre á Carthage (Tunisia), Cujae Havana (Cuba), Buap Puebla (Mexico), University San Diego (California, USA), State New York University, College at Geneseo (NYC, USA), Keiser University (Florida, USA), Abdelmalek Essadi (Morocco), Saint Joseph University of Beirut (Lebanon) John Paul II Catholic University, Lublin (Poland), Universita ca Foscari di Venezia and La Sapienza Roma (Italy), Universite Constantine (Algeria), Southern Federal University in Rostov-on-Don (Russia) and Queens University, Belfast (UK). He teaches in five languages (English, French, Spanish, German and Italian).

José María Rodríguez Corral received his Master's degree in information technology and his PhD from the University of Seville (Spain) in 1993 and 2002, respectively. From November 1993 to June 1995, he worked on robot control with the Robotics and Computer Technology for Rehabilitation Group at the University of Seville. After working as an Associate Professor at the University of Cádiz (Spain) from 1995 to 1998, he is currently Professor Titular of Computer Languages and Systems at the same University, and his research interests include robotics, bus emulation and parallel systems. He is also a member of the Applied Robotics Group at the University of Cádiz, and he is the author of various papers and research reports on computer architecture.

Antón Civit Balcells received his Master's degree in physics (electronics) and his PhD from the University of Seville, Spain, in 1984 and 1987, respectively. After working for several months with Hewlett-Packard he joined the University of Seville, where he is currently a full Professor of Computer Architecture. He is the author of various papers and research reports on computer architecture, rehabilitation technology and robotics. He is the Director of the Robotics and Computer Technology for Rehabilitation Group at the University of Seville. His research interests include advanced wheelchairs, robotics and real-time architectures.

Arturo Morgado Estévez is a full-time professor in the field of applied robotics. He received the MS degree in Industrial Organisation Engineering and his PhD from the University of Cádiz in 1997 and 2003, respectively. He worked on ASIC design with the Microelectronics Group at the same university from 1989 to 1998. Afterwards, he worked with the Robotics and Computer Technology for Rehabilitation Group at the University of Seville from 1998 to 2010. He has been Head of the Robotics Application Group at the University of Cádiz since 2010 and his research interests include robotics, bio-inspired electronics design and AER communication.



Fernando Pérez-Peña received an engineering degree in telecommunications from the University of Seville, Spain, in 2009. After working for the Spanish navy in Navantia (the main Spanish shipyard) for four years, he joined the University of Cadiz for one year, teaching in the computer networking area. In 2010, he was granted a fellowship from the same University. Currently, he is working towards a PhD degree in the field of robotics control by FPGA. His research interests are networking, FPGA digital design, computer architectures, motor control and robotics.