Exploring Multi-core Design Space: Heracles vs. Rocket Chip Generator

Eduardo André Neves*, Samuel Xavier-de-Souza
Department of Computer Engineering and Automation Federal University of Rio Grande do Norte, Natal, RN, Brazil.

* Corresponding author. Tel.: +5584999226417; email: eduardo.andre.neves@gmail.com
Manuscript submitted May 10, 2017; accepted July 25, 2017.
doi: 10.17706/jcp.13.5.555-563

Abstract: This article presents the analysis and comparison of two powerful tools to explore design space and study multi-core microprocessors. Heracles, developed at the MIT, is a modular tool to create MIPS multi-core processors. Its modularity allows quick development by varying the types of processor, memory, network interconnect and cache. RISC-V is a new instruction set architecture, developed at the University of California, Berkeley, that has several tools for designing architectures and processors that use this instruction set. The Rocket Chip Generator is one of these tools. It is the equivalent of Heracles for the MIPS architecture and, as such, provide several features that allows rapid development of RISC-V multi-core processors. We compared the target hardware and software features of these tools. The Rocket Chip Generator proved to be an excellent tool for the development of new RISC-V processors whereas Heracles seemed a more adequate tool for teaching and parallel architecture research at higher levels of abstraction.

Key words: Design space exploration, multi-core, heracles, rocket chip generator, MIPS, RISC-V.

1. Introduction

Parallel computing has passed from alternative to necessity since the last decade. Today there are almost no personal computers or even smartphones with processors that does not require parallel computing to get maximum performance. Multi-core processors came enabling an improvement in the performance of computer systems impossible to be achieved with a single core due to physical limits. Simply, single-core processors need to increase the frequency to improve the performance, thus resulting in higher energy consumption and heat dissipation. Multi-core processors allow performance to grow by increasing the number of cores without increasing clock rate, which has a quadratic effect on power consumption increase.

Most current parallel systems aim to achieve a balance between performance and energy consumption, in most cases prioritizing performance. Indeed, increasing the number of processor cores increases its nominal performance expressed in the number of operations per second. Nevertheless, this makes these processors more difficult to program due to the increase in complexity caused by the necessity to manage communication and synchronization in parallel programs. There are, however, many applications where performance cannot be given priority over battery life, such as in mobile computing, or over the financial cost of the electricity bill, such as in data centers and in high performance computing systems. For these applications, increasing the number of cores may be used as an alternative to reduce the power

consumption of such systems. In [1], for example, the authors describe how increasing the number of cores combined with reduced clock rate can result in a reduction in energy consumption of a parallel task for the same level of performance. However, when the resources for manufacturing a processor are limited, the increase of the number of cores does not always leads a reduction in energy consumption [2] or even in a performance boost [3]. Theoretically, there is an optimal number of cores that depends on the application and is not the same when the goal is to increase performance or reduce the power consumption [2].

Therefore, although the number of cores is not the only relevant factor in the design space exploration of multi-core processors, it is perhaps the most important one. It is arguably not trivial to manually change the number of processing cores of a project in order to enable rapid exploration of design options. Tools to assist in this task are thus potentially quite useful.

In this article two powerful tools for exploring multi-core processor design space are analyzed and compared, especially with regard to the possibility of variation in the number of processor cores. The Heracles tool, developed at the Massachusetts Institute of Technology (MIT), enables the exploration of options in the design space of the MIPS architecture. The Rocket Chip Generator, allows design space exploration of multi-core processors in the RISC-V architecture, which is developed openly for academic and industrial use at the University of California, Berkeley.

This article is organized as follows. Section 2 presents the Heracles tool and their main features. Section 3 presents a brief introduction to the RISC-V architecture and the main features of the tool Rocket Chip Generator. In Section 4 both tools are compared highlighting their main strengths. Finally, in Section 5, we present our conclusions about the advantages and disadvantages of using each tool for different scenarios.

2. Heracles

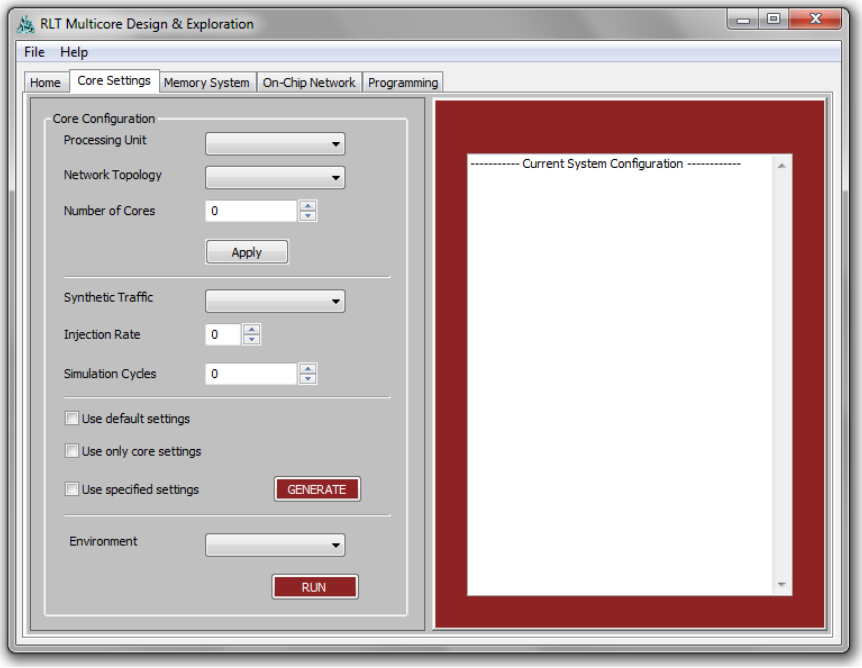

Fig. 1. Heracles graphical user interface.

Heracles [4] is an open source tool built for the development of multi-core processors making it possible to generate FPGA implementations. Developed at MIT, it is a completely modular and parameterized tool. Each of the hardware modules are developed in Verilog and it has the processing unit based on the MIPS architecture. The modules can be fully configured, which allows the developer to explore several

implementation approaches, reconfiguring and synthesizing different processors by varying attributes such as the number of cores, the network topology, and the details of cache and main memory. This makes it possible to the developer to quickly measure the impact of each setting on system performance.

Another strength of this tool is the Graphical User Interface (GUI) where the developer can quickly change various processor configurations as shown in Fig. 1. Although the GUI runs only on Windows, the RTL (*Register Transfer Level*) modules can be simulated in all operating systems and its compiler, the *GCC MIPS cross-compiler*, runs on Linux environments [5].

Heracles allows a complete processor to be built, simulated and synthesized on a FPGA with minimal effort. Fig. 2 illustrates its development flow. Parallel or serial applications written in C/C++ are mapped and compiled for the processing units using the Heracles MIPS-based GCC compiler [6].

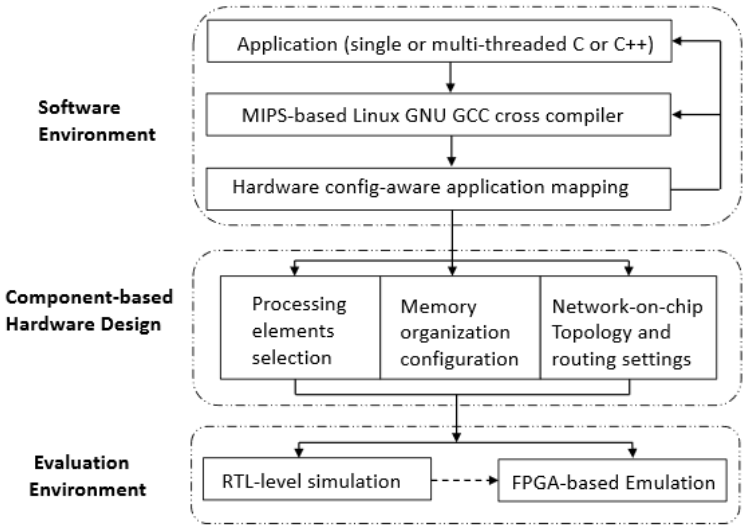

Fig. 2. Heracles based development flow.

2.1. Processing Units

Heracles provides four processing unit modules. The user can choose any of them specifically or a variation of these processors.

- Injector Core (iCore): It is the simplest processing unit. Issues and/or collects data streams defined by the user network and traffic patterns. Although it does not perform any useful computation, this type of core is useful when the user is only interested in the network behavior in the chip.
- Single Hardware-Threaded MIPS Core (sCore): Implements a 7-tages MIPS microprocessor. This RISC architecture is widely used in commercial products and educational purposes. Runs with MIPS-III instruction set without floating point. Instruction and data caches are implemented using RAM blocks and the search instruction and access data memory takes two cycles. The instructions are issued and executed in order, and the access to the data memory are also in order.
- Two-way Hardware-threaded MIPS Core (dCore): A fully functional multithreaded MIPS processor. The execution data path for each thread is similar to the sCore. Each of the two threads has its own context, which includes a program counter, a set of 32-bit data registers, and a 32-bit status register.
- Two-way Hardware-threaded MIPS Core with Migration (mCore): The fourth type of core is also a two-way hardware multi-threaded processor, but enhanced to support the migration of tasks at the hardware level.

2.2. Memory System Organization

The Heracles memory system is parameterized and can be modified in many ways independently of the processor.

- Main memory configuration: Allows multiple address settings, such as shared, centralized and distributed memory. Remote access mechanism is also supported.
- Cache System: The developer can create instances directly from Level 1 or Level 1 and 2 with the choice of level 2 cache be inclusive, that is, it can keep a copy of the cache Level 1 data.

2.3. Network-on-Chip

In order to allow scalability, Heracles uses a Network on Chip (NoC) architecture for its data communication infrastructure. The NoC architecture is defined by its topology, its flow control mechanism and its routing algorithm. The parameterization of the number of input and output ports on the router and routing capability based on table gives Heracles flexibility and the ability to be molded in different network topologies, for example, k-ary, n-cube, 2D-mesh, 3D mesh, hypercube, ring, or tree.

2.4. Programming Module

Heracles has no operating system implemented, but through Application Programming Interfaces (API), the developer can create programs following the sequential or parallel programming models. For each program there are three memory spaces associated: the instruction memory, data memory and stack memory. In sequential programming, the program has only one access point and only one thread running. In the parallel programming model, Heracles use the OpenMP programming model [7]. As previously mentioned, in software environment it has a MIPS-based GCC cross compiler and the graphical interface that helps in a quick set of parameters.

3. RISC-V and the Rocket Chip Generator

The RISC-V [8] is a new Instruction Set Architecture (ISA) based on the RISC architecture. It was developed by Computer Science Division of the University of California, Berkeley, and had an initial purpose to be used in research and teaching, but is currently set to become a standard open architecture for industry implementations. The advantage of RISC-V is that it is simpler than other commercial Instruction Set Architectures (ISA). It supports applications with 32-bit, 64-bit and 128-bit memory space. The ISA has also support to implementations of multi-core and many-core processors, also including heterogeneous multiprocessors [9].

The RISC-V architecture allows the fast development of a new architecture with basically no license restrictions, it's possible to create and use open and free works and to create non open-source products, the only restriction is you acknowledge the authors, in this case UC Berkeley. Because of its flexibility and its modular design, it allows the same architecture and instruction set to be used in many different processors, from high performance processor designs to small projects of embedded processors.

3.1. Chisel

In order to facilitate and speed up hardware development, the University of California, Berkeley, also developed Chisel (Constructing Hardware In the Scala Embedded Language), a hardware description language based on the Scala programming language [10]. Chisel is open source and was designed to be a simple development platform, yet powerful allowing even object orientation. Chisel can generate Verilog, which is a lower level language, allowing either FPGA emulation or application specific integrated circuit (ASIC) synthesis. Chisel can also generate a high-speed C++ based cycle-accurate software simulator [11]. Due to its characteristics of supporting advanced hardware design, Chisel is the main hardware description

language used in the Rocket Chip Generator.

3.2. Rocket Chip Generator

The Rocket Chip Generator is an open source tool to generate Systems on Chip (SoC) based on RISC-V that was also developed at the University of California, Berkeley. It has industrial and research purposes. Rather than being just a single chip generator, it allows the creation of many templates of chips [12].

The tool consists of a set of parameterized libraries that can be used to generate different types of SoC. By standardizing the interfaces used to connect generators of different libraries to each other, it creates a plug-and-play environment, which makes it easy to change hardware components without the need to change the source code of each one [13]. The generators available are listed as follows:

- Core: Scalar Rocket Core generator
- Caches: a family of cache generators and TLB (Translation Lookaside Buffer).
- Rocket Custom Coprocessor: RoCC, is a coprocessor model for specific applications.
- Tile: a Tile model generator for cache coherency in Tiles.
- TileLink: a generator for cache coherency agents network and associated cache controllers.
- Peripherals: generators compatible with AMBA bus (AXI, AHB-Lite and AHB) and a variety of controllers and converters.

3.3. Rocket Core

Rocket Core is one of the chip models that can be generated with the support of the Rocket Chip Generator. It has a 6-stage pipeline shown in Fig. 3 [14], which executes the RISC-V 64-bit instruction set. The processor implements a memory management unit that supports virtual memory paging and can still run modern operating systems such as Linux. It has 31 entries, the processor has a user mode and a supervisor mode.

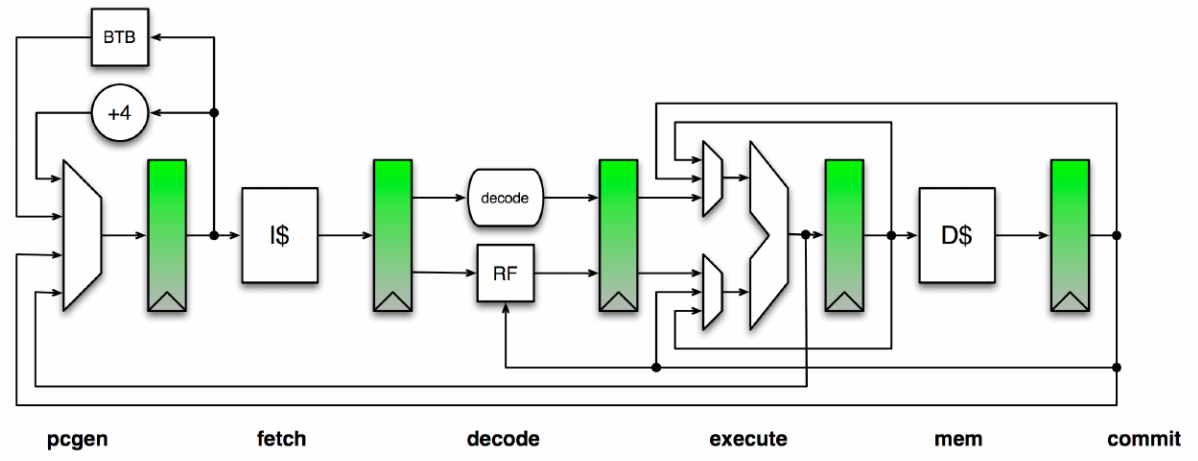

Fig. 3. Rocket pipeline.

4. Comparison

In this section we compare Heracles with the set of tools developed around the RISC-V architecture, with special emphasis on the Rocket Chip Generator. As detailed in the previous two sections, Heracles and RISC-V have two distinct concepts. While the first is a tool created specifically for assisting in research and learning of multi-core architectures, the second is an instruction set architecture with a wide range of tools that has the same purpose of Heracles.

Both tools were developed in Universities with large reputation, Heracles at the MIT and RISC-V and tools at the University of California, Berkeley. While Heracles is based on the MIPS introduction set, Berkeley's set of tools has its own instruction set. However, both MIPS and RISC-V ISAs are based on the RISC architecture, i.e., they have the same concept of reduced instruction set.

Heracles only supports processors with 32-bit memory addressing whereas RISC-V has support for 32, 64 and 128 bits. This is a limitation of Heracles. It means it can only access 4GB.Many applications and hardware nowadays require more than 32 bits.

In the study of multi-core systems, the capacity to model cache is important due to the need of cache coherence among different caches to simplify programming. Heracles has only support to L1 and L2 caches, but in its GUI it is possible to choose support for cache coherence. The Rocket Chip Generator has a parameterized module for the creation of cache that allows cache creation of levels L1, L2 and even L3. In both tools, the cache size, the number of lines and levels may be configured.

One of the fundamental decisions to be made in the design of a microprocessor is the choice of the structure of the pipeline [15]. Regarding the pipeline stages, Heracles has 7 stages while the RISC-V ISA does not have any restrictions on the number of stages. The choice and design is left to the developer, though the Rocket Core has been implemented with a pipeline of 6 stages.

The network topology of Heracles processors is a NoC and can be k-ary n-cube, 2D-mesh, 3D mesh, hypercube, ring, or tree. NoCs have been recognized as the new paradigm to interconnect and organize a high number of cores because of the ablility to accommodate a large number of cores, satisfying the need for on-chip communication and data transfers [16]. Future systems containing tens to hundreds of IP cores having billions of transistors, executing tasks more and more complex. This feature makes Heracles more suitable to study complex parallel communication patterns when compared to the RISC-V set of tools. Rocket Core has a bus for communication instead of a NoC. Buses have successfully been implemented in virtually all complex System on Chip (SOC) design [17]. Generally, busses are simpler, faster and inexpensive to build, but when constructing a many-core system they may not handle the necessary throughput and may generate contention that severely degrade performance. However, RISC-V designs can be structured to have support for chip networks.

The number of cores can be configured in both tools. In Heracles, the user can indicate the desired number of cores by just modifying one variable in the GUI. In Rocket Chip Generator, It is necessary to configure the module code to be able to run a multi-core system, but it modularity makes this easy to achieve.

Although Heracles GUI helps in development by easily allowing the user to customize settings such as the number of cores, memory system, NoC and programming, it runs on Windows only. RISC-V set of tools has no GUI, but it has its own ISA simulator, Spike [18], that implements a functional model of one or more RISC-V processors. It also has its own simulator riscv-qemu [19], that supports a full system emulation in 32 or 64 bits.

Heracles and RISC-V set of tools use different Hardware Description Languages (HDL). HDLs is a programing language oriented to the hardware structure and behavior. By joining a HDL with a component library it is possible to do synthesis and automatically generate a digital circuit [20]. Heracles have its hardware modules developed in Verilog, which is standard a language. Rocket Chip Generator uses Chisel, a higher level language that can generate low-level Verilog. So, both Heracles and Rocket Chip Generator design can be easily configured to be imported into any FPGA. Specifically, both tools have support to have designs directly imported into a FPGA from Xilinx. Heracles has direct compatibility with the Xilinx Virtex chip 6 and Rocket with the Xilinx Zynq board.

In Table 1, the main features of the compared tools are presented side-by-side.

Table 1. Comparative Table Heracles vs. RISC-V

	Heracles	RISC-V (Software Tools)
Developed at	MIT	Berkeley
ISA	MIPS	RISC-V
Address bits	32	32, 64, 128
Cache	L1 and L2	L1, L2 and L3
Pipeline-Stages	7	6
Communication	k-ary, n-cube, 2D-mesh, 3D mesh, hypercube, ring, or tree	Bus
Many-Core	In the GUI is possible to set 1024 cores.	Parameterized, can be configured.
Multicore	Easily parameterized in the GUI.	Can be configured.
Supported FPGAs	Xilinx Virtex6	Xilinx Zynq Board
Operating Systems	Windows & Linux	Developed in Linux
Licensing	MIT	BSD
HDLs	Verilog	Chisel
Heterogeneous Architecture	×	✓
GUI	✓	×
Own Simulator	×	✓
Own Emulator	×	✓

The easy of use and configuration aided by its GUI makes Heracles more suitable for studying the higher levels of the multi-core design space. The iCore option combined to the many different NoC topologies that can be configured also makes Heracles suitable for studying parallel computing communication patterns, which becomes more and more relevant with the increase in the number of cores. Considering that increasing the number of cores is the actual default way to increase performance, the ability to study these patterns seems to be a very desirable and useful feature.

The RISC-V set of tools provides a very good environment to develop new processors without having to redesign everything from scratch. The scandalized interfaces that allow different modules to be connected without adjusts or tweaks makes it easier to put together new architectures at moderately low levels of abstraction. This allows designs that can easily compromise among different design goals such as low power, high performance, robustness and application specificity.

5. Conclusion

We compared two academic tools for rapid development and exploration of multi-core processor design space.

The RISC-V is an instruction set architecture with many tools that assist in the creation of new architectures. These tools are powerful for the development of research and teaching, but they are also suitable for creating complete new industry standard architectures. The flexibility and modularity of RISC-V allow it to be used both for academic as for industrial designs and enables the creation of processors for purposes that goes from embedded systems to high-performance systems.

However, if the processor internal architecture is already defined or is not relevant to the study, Heracles will probably be the best choice because with its GUI and a rapid change of the parameters it is possible to build various types of parallel architectures easily. Thus, Heracles becomes a dedicated tool for research and teaching in the area of parallel microprocessors and a powerful and versatile tool for the exploration of parallel systems design space.

This research may serve as basis for the development of new parallel systems. For example, the analysis and study of the cache in parallel systems may lead to better understanding of the positive and negative effects of this feature. Cache coherency, data sharing and other factors are of special interest for cache

analysis.

References

[1] Arthur, D., Barros, C. A., Silveira, L. F. Q., Xavier-De-Souza, S., & Valderrama, C. A. (2014). Parallel cyclostationarity-exploiting algorithm for energy-efficient spectrum sensing. *IEICE Transactions on Communications, 97(2)*, 326-333.

[2] Barros, C., Silveira, L., Valderrama, C., & Xavier-de, S. (2015). Optimal processor dynamic-energy reduction for parallel workloads on heterogeneous multi-core architectures. *Microprocessors and Microsystems, 39(6)*, 418-425.

[3] Hill, M. D., & Marty, M. R. (2008). Amdahl's law in the multicore era. *IEEE Computer, 41*, 33-38.

[4] Heracles. (2016). Retrieved August 1, 2016, from http://projects.csail.mit.edu/heracles/

[5] Kinsy, M. A., Pellauer, M., & Devadas, S. (2011). Heracles: Fully synthesizable parameterized mips-based multicore system. *Proceedings of the 21st International Conference on Field Programmable Logic and Applications* (pp. 356-362).

[6] Kinsy, M. A., Pellauer, M., & Devadas, S. (2013). Heracles: A tool for fast rtl-based design space exploration of multicore processors. *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (pp. 125-134).

[7] OpenMP. (2016). Retrieved August 1, 2016, from http://openmp.org/wp/

[8] RISC-V. (2016). Retrieved August 5, 2016, from https://riscv.org

[9] Waterman, A., Lee, Y., Patterson, D. A., & AsanoviA, K. (2016). *The Risc-V Instruction Set Manual.* Berkeley: EECS Department, University of California.

[10] The Scala Programming Language. (2016). Retrieved August 5, 2016, from http://www.scala-lang.org/

[11] Bachrach, J., Vo, H., Richards, B., Lee, Y., Waterman, A., Avizienis, R., Wawrzynek, J., & Asanovic, K. (2012). Chisel: Constructing hardware in a scala embedded language. *Proceedings of the 49th Annual Design Automation Conference* (pp. 1216-1225).

[12] AsanoviA, K., Avizienis, R., Bachrach, J., Beamer, S., Biancolin, D., Celio, C., Cook, H., Dabbelt, D., Hauser, J., Izraelevitz, A., Karandikar, S., Keller, B., Kim, D., Koenig, J., Lee, Y., Love, E., Maas, M., Magyar, A., Mao, H., Moreto, M., Ou, A., Patterson, D. A., Richards, B., Schmidt, C., Twigg, S., Vo, H., & Waterman, A. (2016). *The Rocket Chip Generator.* Berkeley: EECS Department, University of California.

[13] Lee, Y., Waterman, A., Cook, H., Zimmer, B., Keller, B., Puggelli, A., Kwak, J., Jevtic, R., Bailey, S., Blagojevic, M. (2016). An agile approach to building risc-v microprocessors. *IEEE Micro, 36(2)*, 8-20.

[14] Keller, B. (2013). Risc-v, Spike, and the Rocket Core. Retrieved August 5, 2016, from http://www-inst.eecs. berkeley.edu/˜cs250/fa13/handouts/lab2-riscv.pdf

[15] Hartstein, A., & Puzak, T. R. (2002). The optimum pipeline depth for a microprocessor. *ACM Sigarch Computer Architecture News*, *30(2),* 7-13.

[16] Anagnostopoulos, I., Bartzas, A., Filippopoulos, I., & Soudris, D. (2012). High-level customization framework for application-specific NoC architectures. *Design Automation for Embedded Systems, 16(4)*, 339-361.

[17] Sharma, S., Mukherjee, C., & Gambhir, A. (2014). A comparison of network-on-chip and buses. *Proceedings of National Conference on Recent Advances in Electronics and Communication Engineering* (pp. 28-29).

[18] Spike. (2016). Retrieved September 30, 2016, from https://github.com/riscv/riscv-isa-sim

[19] Qemu. (2016). Retrieved September 30, 2016, from https://github.com/qemu/qemu

[20] Midorikawa, E. T. (2007). Uma introdução às linguagens de descrição de hardware. Apostila de PCS2304. Retrieved September 30, 2016, from

http://www.pcs.usp.br/~pcs2355/material/intro-hdl.pdf

Eduardo Neves received the bachelor's degree in computer engineering from the Universidade Federal do Rio Grande do Norte, Natal, Brazil in 2012. He is currently working toward the master's degree in an experimental multi-core architecture with frequency adjustment for minimizing energy consumption of parallel applications at Universidade Federal do Rio Grande do Norte, Brazil. His research interests are parallel computation, multi-core processors, embedded systems and field-programmable gate array.

Samuel Xavier-de-Souza was born in Natal, Brazil in 1976. He was a computer engineer at Universidade Federal do Rio Grande do Norte, Brazil in 2000. He received the Ph.D. in electrical engineering at Katholieke Universiteit Leuven, Belgium, 2007.

He worked for IMEC, Belgium, as a software and hardware engineer and for the Flamish Supercomputing Center, Belguim, as a high performance computing consultant. Since 2009, he is a professor of the Department of Computer Engineering and Automation of Universidade Federal do Rio Grande do Norte, Brazil.