Hey Fellows, We Shrunk the Server

http://dx.doi.org/10.3991/ijoe.v8iS2.1960

Valentim Sousa[1], Paulo Ferreira[2] and Manuel Gericota[2]
[1] Bullet Solutions, S.A., Porto, Portugal
[2] Instituto Superior de Engenharia do Porto, Porto, Portugal

***Abstract*—Remote laboratories are an essential part of Web-based engineering lecturing, enabling future engineers 24/7 access to lab resources. Furthermore, they allow sharing expensive resources among multiple universities and research centres. Typical remote lab architectures feature a server, normally a computer that may serve one or more experiments. Computers are generally bulky, somewhat expensive and require heavy resources to run complex operating systems. In this paper, a remote lab for the test of printed circuit boards and the programming/configuration of programmable logic devices and memories through a JTAG interface is presented. This lab is based on open source software and on a cheap router with OpenWrt firmware, a Linux distribution targeted at embedded systems, which acts as a processing unity. A router acting as a server is not a common solution in remote labs. When compared to a "normal" computer, the router has a lower processing and memory capacity. However, our results proved that it has a very good performance, and is able to cope with the desired task.**

***Index Terms*—Electronic engineering education, Embedded software, Programmable logic devices, Remote laboratories.**

I. INTRODUCTION

This work focus two main areas: the effective remote control of the JTAG interface [1] for configuration of the programmable logic, and the search for a sustainable hardware platform for remote laboratory development. Sustainability means a low cost platform, with reduced power consumption, small size, easily reproducible, and whose framework and features may easily be migrated to another platform, if the original becomes obsolete, or if an upgrade in performance is needed.

The remote control of a JTAG interface is very easy if using an Ethernet enabled JTAG interface, but its price is very high, when compared to a more common USB controlled interface. When teaching reconfigurable logic, one of the proposed methodologies is to lend (or force the buy of) one development board per student, and hope the students will not damage them [2]. Another alternative is to use a remote laboratory, which is also interesting in research, where sometimes the development board's setup is complicated and time consuming.

In a typical reconfigurable logic workflow, the target board with the programmable logic devices (Complex Programmable Logic Devices - CPLDs or Field-Programmable Gate Arrays - FPGAs) is used just during a small fraction of the total development time. First there are the design and coding phases, and after that, the simulation phase. If the simulation is correct, the design is synthesized (transformed in the correct pattern of bits for the concrete logic device to produce the desired circuit), and finally the logic devices are programmed.

Except for the final phase, all the other phases can be done without access to the physical board, and using no-cost software, available from programmable logic device manufacturers. Therefore, one board can be shared between many students or groups, because they only need the hardware during a small fraction of the total development time.

The low cost of some programmable logic development boards may appear as a strong motive against programmable logic remote labs. But the development boards are only a part of what students need to do their work. Sometimes they need oscilloscopes, logic analyzers, and they always need programming cables. The transition from parallel port connected programming cables to USB connected ones leads to a greater performance, although their price may be higher than that of the development board.

Another hidden problem is that the low cost of some electronic boards or devices is really a mirage in a peripheral country, where the shipping costs (and sometimes custom taxes) must be added to the original cost, because the boards are not available locally.

On the other side of the price spectrum there are very complex FPGA boards used in Application Specific Integrated Circuit (ASIC) simulations, embedded (multi)processor system development, signal processing, and other high performance computational tasks. On these cases, the creation of a remote lab around the programmable logic board is an excellent way of protecting the board from mishaps, while making it available to a maximum number of students or researchers.

II. REMOTE LABS

The reasons behind remote labs use in engineering degrees are exposed in [3], where the economic and pedagogical basis of remote labs are discussed.

In [4], recent trends in the remote labs are surveyed and several standardization efforts are analyzed and evaluated.

A remote laboratory infrastructure may be divided into four different aspects:

- The connection, with the physical instruments or devices;
- The server, which places the experiments on the Internet;
- The authentication of the users and the scheduling of the experiments;
- The network security.

The authentication of the users is not included in the network security, because it is usually (or should be) dele-

gated to already existing authentication servers. The scheduling of the experiments is associated with the authentication of the users because both tasks provide (or deny) access to the experiments. The network security (use of firewalls, use of secure protocols) must be built in the system and should not be an add-on.

On figure 1 is a diagram showing the main components of a remote lab. In order to have a sustainable system, the research effort was focused on the computer (or computers) controlling the device(s) under test. The remote laboratory architecture needs authentication and protocols for communication, but those can be changed with software upgrades, while hardware upgrades are usually expensive and difficult.

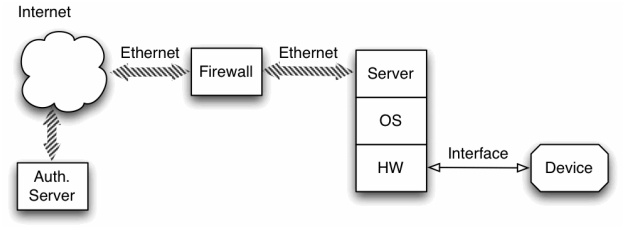

Figure 1. Generic Remote Lab Architecture

USB and Ethernet are nowadays the most prevalent physical interface connections in instruments or devices.

Due to the standardization of USB devices it is easy to find instruments/devices with an USB interface. With the popularization of the Internet many other instruments/devices can *talk* with computers or other devices using the TCP/IP protocols (example: LXI [5]).

The choice of the operating system (OS) is guided by two opposite goals: a rich set of network services and development tools (if possible) on one hand and a low cost on the other. Linux is a good choice because it provides a good compromise between the two constraints.

The use of microservers, defined as microcontroller boards with Ethernet interface and a limited kernel with some available internet protocols (like http and telnet), is not very attractive due to several reasons:

- The available CPU power is limited;
- The set of available TCP/IP protocols is usually limited;
- The programming tools/environments are very device/vendor dependent;
- The price is low, but microservers need usually other auxiliary boards;
- If the microserver uses an operating system, it is an OS with reduced functionality;
- The upgrade path to devices with a better performance is not easy.

The use of small Single Board Computers (SBCs) offers other interconnection possibilities, a greater performance, and more capable operating systems. However, care must be exercised in its choice because sometimes they use non-standard connectors or buses and require development toolchains that are not free.

A common option is the use of a generic PC as a server for remote labs due to their good price/performance ratio. But generic PCs need a careful maintenance if used as full time servers (due to the existence of fans and hard disks), and unless fitted with special boards, their input/output connections are normally limited. However, one of their great attraction and advantage is the existence of many free operating system variants (Linux, FreeBSD, NetBSD, OpenBSD and others), with a solid set of device drivers for many peripherals, and an extensive choice of development environments and tools.

Based on the above considerations its is possible now to synthesize the desirable system requirements, in order to choose the ideal target platform:

- Operating System: Linux
- Small Size
- Low Power
- Low Cost
- Interfaces:
  - Ethernet
  - USB

The set of interfaces was reduced to Ethernet and USB, for connection convenience.

III. JTAG INTERFACE

In the late 80s the use of the traditional beds-of-nails, requiring physical access to integrated circuit (IC) pins or PCB test points, to test printed circuit boards (PCB) became unfeasible due to the increasing integration and miniaturization of ICs. The JTAG infrastructure, later standardized under IEEE Std. 1149.1 [1], was developed to cope with the problem. The initial aim was to provide virtual access to IC pins through a boundary scan infrastructure accessible by a four-pin Test Access Port (TAP), also known as JTAG interface. The four pins are:

- Test clock input (TCK);
- Test mode select input (TMS);
- Test data input (TDI);
- Test data output (TDO).

The first two enable the control of an internal finite state machine that manages the different functionalities implemented by the infrastructure, while the latter two are serial instructions or data input/output. Several ICs may be serially connected, linking the TDO from one IC to the TDI of the next IC. TCK and TMS are connected in parallel to all ICs in the chain. Instructions are serially scanned to the different ICs, enabling each one to perform different functions concurrently.

The standard was so successful that the same access port started to be used by manufacturers to perform other functions beyond the strict functionalities of test that were the reason of its original inception. The possibility open by the standard of inserting optional internal registers and instructions was favoured by manufacturers of programmable logic devices (PLDs), which started using the JTAG interface as a programming port for CPLDs and FPGAs. This new functionality became IEEE Std. 1532 [6], and was reused in the work described herein to gain access to the FPGA.

IV. RELATED WORK

The description of one of the first systems oriented for academic use can be found in [7-8]. The LABOMAT system is based on a custom made SBC with a 68000 family

processor, 10BaseT Ethernet, with a special real-time operating system (RTEMS) port, and uses two Xilinx FPGAs (XC4013E and XC6216) for the reconfigurable part. The system construction required, besides the "normal" software development, the adaptation of an operating system and the full hardware development.

The MEDICIS system described in [9] consists of the remote control of a Logical Analyzer and associated pattern generator for testing of FPGA-based circuits. The remote control is implemented connecting a workstation to the logic analyzer's serial port. It is not clear in the article how the FPGA is programmed.

In order to teach microprocessor architecture and design, the LABOMAT3 system is used in [10], with an additional debugger interface, but with all the fixed hardware and software specifications of the former LABOMAT system.

The system proposed in [11] is very well structured, with the use of a Virtual Private Network for experiment access and of a very precise separation of tasks like authentication, experiment scheduling and instrumentation access among different modules. Once again, a logic analyzer is used, but this time a special, and therefore expensive, logic analyzer with a PC based internal architecture. The JTAG control is done through a parallel port interfaced JTAG cable, connected to a PC, where a Java-based server is running.

On the LADIRE remote lab [12] the remote control is simply done using a remote desktop protocol (RDP). This means that any PC application (like the Quartus Altera software) can be used, but also means the user can fully control the PC (limited only by the Windows security mechanisms), and not only the JTAG interface. The permissions issue is something that needs to be carefully defined, because the user's software needs low-level hardware access to the parallel port (where the JTAG interface is connected), but the user should not be able to mess with the rest of the computer's hardware, bringing the remote lab to a non-functional state.

The development of an FPGA remote lab is discussed in [13], with three different versions: a "normal" client-server based solution; a middleware (Jini based solution); and a Ptolemy integrated version. The added value of this work is the integration of the remote lab in a Ptolemy-based distributed system workflow, very useful if one uses the Ptolemy based tools.

The remote lab proposed in [14] also uses the RDP protocol and relies on Labview driven data acquisition interfaces with the FPGA board. The Labview software and data acquisition interfaces make this solution very expensive.

A very interesting system is proposed in [15]. A set of 64 Xilinx ML-310 boards is remotely controlled through command-line-based controls. Each board has a serial port, connected to a server via a group of USB to serial converters. The reset of the boards is made using an individual internet controlled power switch (PDU - power delivery unit). The boards boot from an existing compact flash (CF) board, needed because the FPGA available on the board has an embedded hard core PowerPC. In sum, while the control task is simple, the board is rather complex. The system is oriented for complex embedded systems development, where the reconfigurable logic is used together with a hard core RISC processor.

A microserver is used in [16]. It is a very economical solution in terms of parts cost, but their functionality is limited, and the software development is very specific to the concrete model of microprocessor. Microservers are a good option for fixed functionality devices, produced in a very large scale and very cost sensitive, but for small scale systems, like remote labs, the additional cost of using a more powerful processor with a standard operating system, simplifies the software development, makes available additional development tools and eases the future migration to other platforms.

The work presented in [17] makes a very different use of the FPGAs. Instead of being the object of study, two FPGAs are used as a stimulus generator and logic analyzer connected to a set of 74HC/HCT logic devices. A PIC-based microserver supports the FPGA control and the internet connection.

A migration from a Labview solution to a custom solution running in Linux and Windows is described in [18], but no technical details are given in the article. There is no info about the programming language used, client server protocols, server hardware or PC to FPGA connections.

An innovative architecture for the deployment of a remote laboratory is exposed in [19]. The modularity and the flexibility of the system are very good, and some details of the description are very revealing. The limitations of microserver-based solutions (low performance, lack of flexibility, and no administration utilities) are exposed. To sidestep these limitations, a small size PC is used with two microservers, with one of them connected to the PC using the USB interface. This means that a simpler alternative to the creation of Ethernet enabled instruments is the creation of USB interfaced instruments, because microservers will almost always need an additional PC, and the connection via USB avoids the need of a TCP/IP stack and Ethernet interfaces on the microcontroller boards.

A very ingenious scheme for simplifying the creation and use of a remote FPGA laboratory can be found on [20]. The system is PC based and supports a set of virtual inputs and outputs through the inclusion in the FPGA design of a special hardware block that acts as a communication support structure - a dual port RAM that can be written/read by the PC, using a special interface, and at the same time by the FPGA.

Also in some articles outside the remote labs research area, interesting remote JTAG access solutions are proposed. For example, on [21] a solution for high-speed JTAG remote access is described. Sadly the solution is patented. One of the proposed mechanisms for Ethernet remote programming of FPGAs is the inclusion of a PIC microserver like in [22]. Another more powerful alternative is explained in [23], where the proposed solution consists of an SBC with a Coldfire processor, and a small real time support OS. The proposed technologies are the same used by other authors in the remote laboratories field.

V. THE SHRUNKEN SERVER

The idea of using a conventional wireless router as a computational platform for remote labs arose from the analysis of their requirements and also due to economical reasons.

In hardware terms, the routers available in the market use 32 bits processors (MIPS or ARM) at around 400 Mhz and have typically 8 MBytes of Flash ROM and 32

MBytes of RAM (a typical configuration of an Unix workstation 15 years ago). Some have USB ports, and can run special Linux distributions. The low price (already with power supply unit) is very attractive. Furthermore, the absence of moving parts (no fans or hard disks) contributes to their high reliability. The number of USB ports is usually limited to one or two, but an USB hub can be used to connect more USB peripherals. The wireless part is an additional market scale advantage, but the great interest of the router is the existence of the built-in Ethernet switch for connecting more Ethernet accessible devices and the firewall incorporated in the Linux distributions running on it. In sum, instead of a server, a switch and a firewall, only one low-cost device is needed.

The price and the greater availability of the wireless routers (any PC shop sells them) are offset by their limited production lifespan. While SBC manufacturers offer, at least, an end-of-life warning, the router market is very volatile. The volatility of the router models is a danger for the sustainability of a router based remote lab. However, since the complete solution will run over Linux, the independence relative to the underlying hardware is assured if a suitable Linux distribution is found.

OpenWrt [24] can be described as a Linux distribution oriented to small network aware embedded devices, like the common wireless routers. OpenWrt can be use just like a normal Linux distribution, replacing the original router firmware (frequently also Linux based), but it is in fact more powerful than that. It includes a full set of cross compiling tools, utilities for the Linux kernel customization and compilation, the creation and maintenance of software packages, and the creation of custom firmware for embedded systems based on ARM, AVR32, MIPS, PowerPC and X86 processors. To simplify the software development of special packages, one can use only the software development kit part of OpenWrt, and not the full distribution, which can create everything that OpenWrt needs.

One of the problems of embedded development is the dependency between the host system OS and libraries, and the cross compilation toolchains. Many toolchains are distributed in binary format, needing a very specific version of the OS and of the installed libraries, barring sometimes any upgrade. In the case of OpenWrt, all the distribution is source-based. There is a small set of tools that must be present in the host system, but all the rest is "freshly" compiled from sources downloaded on request, including the Linux kernel and the cross compilation toolchains. This eliminates any host incompatibilities that sometimes are a big issue in embedded development.

In some embedded programming environments, the lack of immediate feedback delays the development, as the programming cycle includes not only the compile and run phases, but also the "transfer image to target" phase. Sometimes, this is done by reprogramming the target's flash ROM, something very slow. In the OpenWrt case this can be speeded, exporting by Network File System (NFS) the development PC's file system and using it on the router. This gives the router a hard disk based file system for the development phase.

Should the need of low level debugging arise, there is always the possibility of connecting to the router's JTAG chain an On Chip Debug (OCD) JTAG cable, as those pins are always available. This involves only opening the router's box, and soldering a connector in the router's circuit board.

For rapid development, the OpenWrt distribution includes a series of scripting languages (several Unix shell dialects, Perl, Python, Tcl and others), which can be used for the creation of prototypes, diagnostics and system administration. OpenWrt also offers a good selection of low footprint modular web servers, easing the creation of embedded web servers or services.

Portability is also an important issue. Some Linux software is written with disregard of the POSIX standards, breaking when one tries to use a different Linux distribution, but OpenWrt is very POSIX compliant, running also in BSD based systems like OSX.

The big problem with OpenWrt is the lack of documentation, or, in other words, the use of source code as documentation. The "overloading" of makefiles is an example where that can be seen. As (almost) all the Unix-based software, OpenWrt uses makefiles for its building and for the building of everything that it needs. But, to place in the makefiles the information about a specific software package (version number, where to download, file hash validation, dependencies) means that OpenWrt have a special unique format, and thus the developer must be familiar with the OpenWrt peculiarities of the makefiles, using the available ones as an example and documentation.

The big advantage of OpenWrt is to provide a Linux/POSIX programming environment for small and low cost embedded devices. With OpenWrt the developed software is independent of a specific router model, being available for many routers, solving the problem of sustainability due to rapid hardware obsolescence. The Linux based nature of OpenWrt also provides an easy upgrade path if the performance of the router is insufficient. As the developed software is Linux based and uses only USB/Ethernet driven devices, any other Linux machine with USB and Ethernet can be used as a replacement or upgrade, being only necessary to recompile the source code.

VI. IMPLEMENTATION

The choice of a suitable USB-JTAG interface is related to the choice of the software available for JTAG control. To build such an interface (commonly called cable) implies building not only the hardware, but also the software for JTAG control and having the custom software locked to that specific interface. By adopting a standard software package, with support for several cables, the development work is simplified and the choice of cables enlarged.

There is an open source software package for JTAG programming and control, called UrJTAG [25], that works with a big set of cables on Linux, and is suitable for the desired task. But UrJTAG is a command-line oriented "non networked" program. The UrJTAG related development work was structured in the following parts:

1. Analysis of the UrJTAG dependencies (libraries and building tools);
2. Compiling UrJTAG and running it on a desktop Linux machine (validating the analysis);
3. Porting UrJTAG to the OpenWrt distribution;
4. Testing UrJTAG in the OpenWrt based router;
5. Writing a server for UrJTAG;

6. Writing a portable client for the server.

Being a command-line-based program, UrJTAG can always be used remotely, using telnet or ssh connections to the router. But the user-friendliness and reliability of the system is improved if the command line is hidden, for the student's use of the remote lab. For this purpose a C based server controlled by a TCP/IP socket and able to fully control the UrJTAG command-line through a terminal simulation implemented with the aid of POSIX pseudo terminals was written. The server was implemented in C because it needs to be lightweight to run on the router, while the client application was implemented in Java in order to be usable without recompilation on different operating systems.

The full block diagram can be seen on figure 2, where the different blocks are detailed. The device under test (DuT) is connected via JTAG to the cable that connects via USB to the router. The cable used can be selected from all the cables that UrJTAG supports. As UrJTAG is Linux based, the "router hardware" can be changed and replaced by any kind of Linux supported hardware (with USB and Ethernet connections, of course).

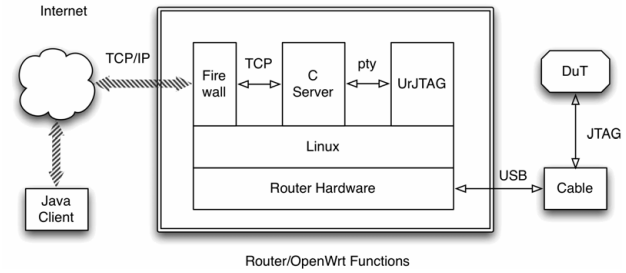

Figure 2. System Architecture

The developed C server uses the Linux system calls to control the UrJTAG program, and serves requests from a TCP/IP socket. The additional firewall placed between the external network and our server is a "bonus feature" from OpenWrt. As OpenWrt is a router oriented Linux distribution, it already comes with an *iptables* based firewall.

The system was tested with several FPGA and CPLD-based boards and three different types of USB-JTAG interfaces (Xilinx, Altera and Segger). Using all the three interfaces is possible for simple JTAG chain detection and manipulation. For more complex tasks like Serial Vector Format Specification (SVF) file replay [26], the Altera USB-Blaster cable has no incompatibilities (even when programming Xilinx FPGAs), programming every tested FPGA without any perceptible delay, although being the cheapest cable.

The authentication and scheduling features of a typical remote lab are not yet implemented by two different reasons: the purpose of this work was to evaluate the suitability in terms of performance and development environment of a router based remote lab and with that aim the OpenWrt development environment was fully tested and evaluated with the developed system; the authentication and scheduling features will depend on external servers, still in development, and do not have any critical performance requirements.

A real test of the system's performance is the broadcast of a video stream of the experience. This was accomplished in our system by the addition of a low cost USB video camera and the installation of a suitable video streamer server on the router. In tests with a VGA quality webcam there was no perceived drop of performance with the video streaming on.

VII. CONCLUSION

The developed system meets all the expectations. The software development is easy as the majority of it can be first done on a common desktop Linux machine, using the desktop machine as target. The solution of "enveloping" an existing command-line-based application using pseudo terminals allows the reuse of applications not ready for remote control. The chosen hardware has a low cost and a good performance, demonstrated by video streaming in parallel with the execution of the developed application.

The proposed router solution has several relevant aspects: the acquisition price is low; the development environment is open-source and a rich set of development tools is available; the energy consumption and required space when compared to a conventional PC are also very low; and the absence of moving parts (no fans or hard disks) leads to a high system reliability.

The existence of the built-in firewall also helps to lower the (security) costs and, together with other network services available in OpenWrt, opens the possibility of remote maintenance, upgrade and diagnostics in a low cost solution. Besides the remote lab software, diagnostics software for remote maintenance may also be implemented.

As the application is developed under Linux and all the hardware interfaces are USB based, changing to a different router, may, in the worst case, imply a recompilation only. In the future, if the computational needs of the system outgrow the ones available in a low cost router, the migration to a more powerful SBC or to a PC running Linux will only imply the recompilation of the developed software.

ACKNOWLEDGMENT

The authors would like to thank Paulo Matos and Jaime Neto from DEI-ISEP for their help in recycling hardware resources on the initial phase of this work.

REFERENCES

[1] "IEEE standard test access port and boundary-scan architecture," *IEEE Std 1149.1-2001*, 200 pp., 2001.

[2] M. Radu, C. Cole, M. Dabacan, J. Harris, and S. Sexton, "The impact of providing unlimited access to programmable boards in digital design education," *IEEE Transactions on Education*, vol. 54, no. 2, pp. 174– 183, May 2011. http://dx.doi.org/10.1109/TE.2009.2037735

[3] J. García-Zubia and A. del Moral, "Suitability and implementation of a WebLab in engineering," in *10th IEEE Conference on Emerging Technologies and Factory Automation, ETFA 2005*, vol. 2, Sept. 2005, pp. 49–56.

[4] L. Gomes and S. Bogosyan, "Current trends in remote laboratories," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 12, pp. 4744– 4756, Dec. 2009. http://dx.doi.org/10.1109/TIE.2009.2033293

[5] "LXI device specification 2011," LXI Consortium, Inc., 2011. [Online]. Available: http://www.lxistandard.org/Documents/Specifications/LXI%20Device%20Specification%202011%20rev%201.4.pdf

[6] "IEEE standard for in-system configuration of programmable devices," *IEEE Std 1532-2002 (Revision of IEEE Std 1532-2001)*, 141 pp., 2003.

[7] C. Teuscher, J.O. Haenni, F. Gomez, H. Restrepo, and E. Sanchez, "A reconfigurable platform for academic purposes," in *Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines,FCCM'99*, 1999, pp. 282–283.

[8] ——, "A tool for teaching and research on computer architecture and reconfigurable systems," in *Proceedings 25th EUROMICRO Conference*, vol. 1, 1999, pp. 343–350.

[9] P. Nouel, P. Kadionik, P. Gressier, P. Dufrene, and S. Lemasson, "MEDICIS: a new tool for remote programmable FPGA circuit testing," in *Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference,IMTC 2000*, vol. 1, 2000, pp. 327–329.

[10] J. Pastor, I. Gonzalez, J. Lopez, F. Gomez-Arribas, and J. Martinez, "A remote laboratory for debugging FPGA-based microprocessor prototypes," in *IEEE International Conference on Advanced Learning Technologies*, 2004, pp. 86–90. http://dx.doi.org/10.1109/ICALT.2004.1357380

[11] N. Fujii and N. Koike, "A new remote laboratory for hardware experiment with shared resources and service management," in *Third International Conference on Information Technology and Applications,ICITA 2005*, vol. 2, July 2005, pp. 153–158.

[12] G. Persiano, S. Rapuano, F. Zoino, A. Morganella, and G. Chiusolo, "Distance learning in digital electronics: Laboratory practice on FPGA," in *IEEE Instrumentation and Measurement Technology Conference Proceedings, IMTC 2007*, May 2007, pp. 1–6.

[13] L. Indrusiak, M. Glesner, and R. Reis, "On the evolution of remote laboratories for prototyping digital electronic systems," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 6, pp. 3069–3077, Dec. 2007. http://dx.doi.org/10.1109/TIE.2007.907010

[14] R. Hashemian and J. Riddley, "FPGA e-Lab, a technique to remote access a laboratory to design and test," in *IEEE International Conference on Microelectronic Systems Education, MSE'07*, June 2007, pp. 139–140.

[15] K. Datta and R. Sass, "RBoot: Software infrastructure for a remote FPGA laboratory," in *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM'2007*, April 2007, pp. 343–344.

[16] J. García-Zubia, I. Angulo, U. Hernandez, and P. Orduña, "Plug&play remote lab for microcontrollers: WebLab-DEUSTO-PIC," in *7th European Workshop on Microelectronics Education*, 2008. [Online]. Available: https://www.weblab.deusto.es/web/images/publications/ewme2008.pdf

[17] L. Gomes, G. Patricio, R. Ferreira, and A. Costa, "Remote experimentation for introductory digital logic course," in *3rd IEEE International Conference on E-Learning in Industrial Electronics, ICELIE'09*, Nov. 2009, pp. 98–103.

[18] R. Hashemian and T. Pearson, "A low-cost server-client methodology for remote laboratory access for hardware design," in *39th IEEE Frontiers in Education Conference, 2009. FIE'09*, Oct. 2009, pp. 1–5.

[19] J. García-Zubia, I. Angulo, U. Hernandez, M. Castro, E. Sancristobal, P. Orduña, J. Irurzun, and J. de Garibay, "Easily integrable platform for the deployment of a remote laboratory for microcontrollers," in *2010 IEEE Education Engineering (EDUCON)*, 2010, pp. 327–334.

[20] J. Soares and J. Lobo, "A remote FPGA laboratory for digital design students," in *REC2011–7th Portuguese Meeting on Reconfigurable Systems*, 2011.

[21] P. Collins, I. Reis, M. Simonen, and M. van Houcke, "A transparent solution for providing remote wired or wireless communication to board and system level boundary-scan architectures," in *IEEE International Test Conference*, 2005, Nov. 2005, pp. 8–16. http://dx.doi.org/10.1109/TEST.2005.1583957

[22] P. Kammerling, A. Ackens, H. Loevenich, A. Borga, P. Wustner, G. Kemmerling, W. Erven, K. Zwoll, H. Kleines, and M. Drochner, "FPGA configuration by TCP/IP and Ethernet," in *15th IEEE-NPSS Real-Time Conference*, 2007, May 2007, 4 pp. http://dx.doi.org/10.1109/RTC.2007.4382790

[23] A. Sukhanov, I. Sukhanov, S. Kim, A. Shutov, and S. Bazylev, "Online monitoring and remote FPGA configuration using JTAG over Ethernet," in *15th IEEE-NPSS Real-Time Conference*, 2007, May 2007, 2 pp. http://dx.doi.org/10.1109/RTC.2007.4382778

[24] "OpenWrt home page." [Online]. Available: http://openwrt.org

[25] "UrJTAG." [Online]. Available: http://urjtag.org/

[26] ASSET InterTech, Inc., *Serial Vector Format Specification*, 1999. [Online]. Available: http://www.asset-intertech.com/support/svf.pdf

AUTHORS

Valentim Sousa is with Bullet Solutions, S.A, Porto, Portugal (e-mail: valentim.sousa@bulletsolutions.com).

Paulo Ferreira is with the Department of Informatics Engineering, Instituto Superior de Engenharia do Porto (e-mail: pdf@isep.ipp.pt).

Manuel Gericota is with the Department of Electrical Engineering, Instituto Superior de Engenharia do Porto (e-mail: mgg@isep.ipp.pt).

This work was supported by Calouste Gulbenkian Foundation, Lisbon, Portugal. It is an extended version of a presentation given during the 1st Experiment@ International Conference, 17/18 November 2011 in Lisbon, Portugal. Manuscript received 20 January 2012. Published as resubmitted by the authors 18 March 2012.