A Remote Laboratory for Debugging FPGA-Based Microprocessor Prototypes

Javier Sanchez Pastor, Ivan Gonzalez, Jorge Lopez, Francisco Gomez-Arribas, Javier Martinez
Laboratorio de Microelectronica. Departamento de Ingenieria Informatica
Escuela Politecnica Superior, Universidad Autonoma de Madrid
Ciudad Universitaria de Cantoblanco, Ctra. de Colmenar Viejo, km 15, 28049 Madrid, Spain
Phone: +34 91 497 2268 – Fax: +34 91 497-2235
{Javier.Sanchez, Ivan.Gonzalez, Jorge.Lopez, Francisco.Gomez, Javier.Martinez}@ii.uam.es

Abstract

In this paper, a framework for testing microprocessor prototypes is presented. A RISC microprocessor is designed by students using VHDL language and adapted to be implemented on a FPGA device. The correct behaviour of the designed microprocessor is checked executing test programs written and compiled by the students for this microprocessor. Using a web client, users send test programs and a file with the design to a remote laboratory where it is loaded on a real FPGA device. A set of tools for debugging the remote execution of the tests has been developed, using a graphical interface similar to other debugging tools. Groups of selected students of a computer architecture course have participated in this experience. The good opinions received from the students, suggest the incorporation of this remote laboratory experience in the next regular course.

1. Introduction

Propelled by the widening adoption of Internet/web technologies, and the evolution of reconfigurable devices, the online experimentation using a remote laboratory is becoming a well suited complement for traditional courses in digital circuit design and computer architecture [1][2].

It is commonly accepted that laboratory experiments are an essential part of electronic engineering and computer science education [3][4]. Basic principles about computer architecture can be studied with real examples. In order to practice their skills, students are asked to build a real microprocessor. An extended experimental approach is the development of a prototype by means of hardware description languages (HDL). Several simulation tools are supplied by the HDL compiler software to verify the correctness of the microprocessor design. In addition, the ease of prototyping a digital system using Field Programmable Gate Arrays (FPGA) based platforms allows a physical implementation of the microprocessor design. However a hardware developing FPGA system with tools to check the microprocessor behaviour is not always available in a traditional student laboratory. In this case a web access to a remote laboratory that provides these tools, can resolve the problem.

This paper presents a new approach to introduce experimentation in Computer Science (CS) courses. The following section describes the exercise proposed to be carried out by the students in the computer Architecture course at Univ. Autonoma de Madrid. The second section explains the traditional experimental procedure into an "in-situ" laboratory. Section 3 describes the remote laboratory approach. The most innovative contribution of this work is to implement a solution for the verification of the microprocessor design based on a FPGA, which is a hardware real device. The device is located in a remote laboratory. For this goal, we have developed a group of remote tools accessed by a friendly graphical user interface from a web client. The remainder of this paper describes the implementation of the remote laboratory, the reconfigurable platform and the network service developed. After that, some considerations and the future work are pointed out.

2. A Computer Science experiment: design of a RISC microprocessor

In a Computer Architecture course the students are able to design, using hardware description languages, a complete processor with a simple and reduced instruction set (RISC).

2.1. Microprocessor description

The design specifications of the simple RISC microprocessor to be carried out by the students are:

- 16-bit architecture.
- Harvard architecture with different memories for code and data.
- Four general purpose internal registers.
- Conditional jumps.
- Simple instruction set with nine basic instructions (LOAD, STORE, MOVE, ADD, SUB, comparison, conditional and unconditional jumps, and HALT).

The external scheme of the microprocessor design is shown in figure 1. The design inputs are the clock and reset signals. There are also a 16-bit address bus and a 16-bit data bus to connect the program memory. The data memory connection presents an 8-bit address bus and two 16-bit data buses, one for input and one for output. Also, different lines are necessary to manage the memories: read mode, write mode, etc.

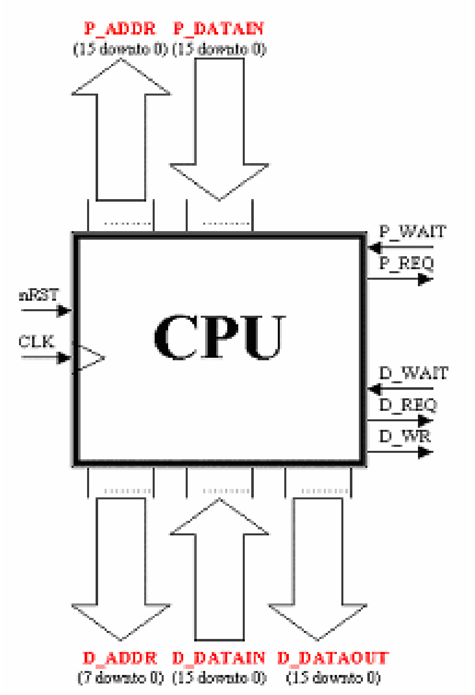


Figure 1. Microprocessor Architecture

2.2. Design, simulation and test

This experimental processor can be developed using the VHDL language. Once the microprocessor is designed, to verify the correctness of the design it is usual to run a test bench using simulation tools. The test bench is a set of programs compiled for the RISC microprocessor.

The student has to write a test bench program to be executed by the microprocessor. This program is compiled using a custom cross-compiler specific for this architecture. The cross-compiler generates two files: the program code file and the data file. These files are used to load the memories of the designed microprocessor system. A VHDL simulation tool uses the test bench program to check the correct functionality of the microprocessor. Students are suggested to write several tests in order to assure the complete functionality of the design.

2.3. Hardware implementation

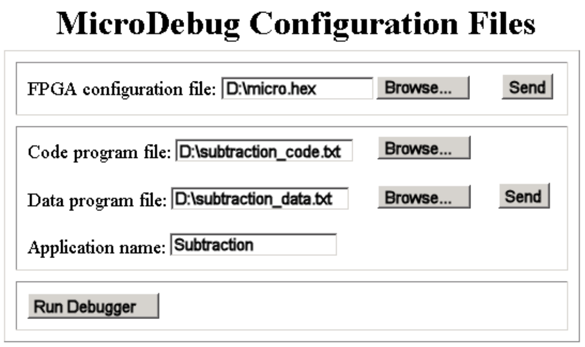
A challenge that requires an additional effort is to implement the design using hardware. In this case, it is possible to go beyond simulation, testing the designs in a real system, which allows facing problems that generally do not appear in simulation. The student can implement his design in a real hardware device, to check the correct behaviour, that is, to run the test bench program in the real microprocessor implemented in a FPGA. This work presents additional complexity because a RTL (Register Transfer Language) codification of the design in VHDL is necessary to be performed in order to pass the synthesis stage. After the design entry stage using a HDL language, the design process continues with two critical stages: the design synthesis and the design implementation. The design synthesis translates the VHDL code to a hardware description list. Next, the design implementation translates the hardware description to a configuration file that can be understood by the FPGA. These stages are automatically performed by commercial manufacturer tools. The last stage is the device programming where the configuration file has to be loaded into the FPGA device.

3. The remote laboratory for microprocessor prototype verification

Once the files explained in previous sections have been obtained using the "in-situ" laboratory tools, the students access to the remote laboratory through a web interface. The tool developed to verify remotely the microprocessor prototypes is called Microdebug. This section present the functionality of the friendly user interface developed for this tool. It is very easy to use by the students because it has been built in a similar way to other debugging tools. Microdebug has two parts that will be explained in the next sub-sections

3.1. MicroDebug: configuration files

When the students start the Microdebug application, a page with two forms is presented, as shown in Figure 2. The first form allows the student to load his synthesized design into the remote FPGA and the second one is used to send to the remote laboratory the test bench which will be executed later by the microprocessor. This last form consists of three fields, one for the program code file, another for the program data file and the last one for a reference name that will allow the debugger to identify this test. Students can send as many tests as they want.



MicroDebug Configuration Files

FPGA configuration file: D:\micro.hex

Code program file: D:\subtraction_code.txt

Data program file: D:\subtraction_data.txt

Application name: Subtraction

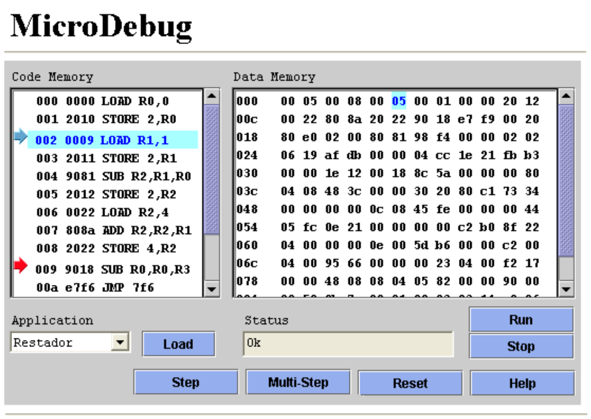
Figure 2. Configuration files

3.2. MicroDebug: debugger tool

The debugger tool starts loading an applet from the server in the local computer and shows a similar interface to the commercial debug tool used in microprocessors development kits (figure 3). The figure has two windows. In the left one the content of the code memory in hexadecimal format is shown. An interpretation of this byte using instruction mnemonics can be observed. The right window shows the data memory in hexadecimal format. At the bottom of the figure a combo-box allows choosing between different test benches. These test benches were stored previously by the student or by the teacher using the process explained before. The teacher can consider that some tests are essential to determine the correct behaviour of the design performed. By pressing the load button, the program code and the initial data associated with the test name are loaded in the code and data memory of the microprocessor.

In the right side of the figure several buttons with different options can be observed. These options are useful to debug the execution of the loaded program:

- By using the reset button, the program counter (PC) register is set with the 0 value. Programs have always to start at this address. A blue arrow located at the left of the code window shows the memory position currently pointed by the PC register. The next instruction to be executed is shown over a blue background.
- By double-clicking an instruction in the code window, a breakpoint can be set on this line. The breakpoint presence is represented by a red arrow. Several breakpoints can be set.
- The Run button executes the program starting from the position indicated by the PC register, until a HALT instruction or a breakpoint is found. When the execution finishes, the contents of the data memory window are refreshed and the modified values are printed in blue background.
- The Step button executes only one instruction. The data contents are also refreshed.
- The Multi-step button starts the execution of the instructions step by step. This execution finishes by the same conditions as the Run button, and can also be stopped by pressing the Stop button.



MicroDebug

Code Memory

```
000 0000 LOAD R0,0
001 2010 STORE 2,R0
002 0009 LOAD R1,1
003 2011 STORE 2,R1
004 9081 SUB R2,R1,R0
005 2012 STORE 2,R2
006 0022 LOAD R2,4
007 808a ADD R2,R2,R1
008 2022 STORE 4,R2
009 9018 SUB R0,R0,R3
00a e7f6 JMP 7f6
```

Data Memory

```
000   00 05 00 08 00 05 00 01 00 00 20 12
00c   00 22 80 8a 20 22 90 18 e7 f9 00 20
018   80 e0 02 00 80 81 98 f4 00 00 02 02
024   06 19 af db 00 00 04 cc 1e 21 fb b3
030   00 00 1e 12 00 18 8c 5a 00 00 00 80
03c   04 08 48 3c 00 00 30 20 80 c1 73 34
048   00 00 00 00 0c 08 45 fe 00 00 00 44
054   05 fc 0e 21 00 00 00 00 c2 b0 8f 22
060   04 00 00 00 0e 00 5d b6 00 00 c2 00
06c   04 00 95 66 00 00 00 23 04 00 f2 17
078   00 00 48 08 08 04 05 82 00 00 90 00
```

Application: Restador

Status: Ok

Figure 3. Debugger Tool

4. Services and network architecture developed for the Web laboratory

The Web laboratory, accessible from anywhere through an Internet connection, has been built according to the general methodology to control instruments through Internet [5][6].

This section points out the group of services and utilities developed, based on client-server architecture that allows remote management of available resources in the reconfigurable platform, as well as the source code compilation, FPGA configuration, memory load and the control of the design implemented in the FPGA. Although the use of these resources is transparent to the user, it is important to describe the amount of hardware equipment and networks services necessaries to implement the remote Microdebug tool.

4.1. The hardware resources

To implement this experiment is necessary a reconfigurable platform with a FPGA which can store the designed microprocessor, a processor which can communicate with the FPGA and runs a control program, and at least a bank of memory which can be accessed either by the processor or by the FPGA. This memory stores the data and the program code to be executed by the microprocessor. Nowadays, there are many platforms which have these features: Labomat3, Celoxica's RC1000PP, XESS, etc. In our case, we have used a Labomat3 board because all the resources are available in the same platform.

The Labomat3 board was developed by the Logic System Laboratory at the Swiss Federal Institute of Technology in Lausanne [7]. The architecture is divided in two main parts:

- The processor part is built around a Motorola 32-bit MC68EN360 processor with integrated Ethernet protocol management [8]. It is connected to 512 KB of boot EPROM, 32 MB of DRAM and 4KB of dual port SRAM.
- The reconfigurable part is built around a XC4013E and a XC6216 FPGA. The XC6216 is connected to 128 KB of additional SRAM. The XC4013E is connected to the dual-port SRAM, which in turn is connected to the processor.

Because of its higher capacity, we have chosen the XC4103E FPGA to store the designed microprocessor. The processor can directly access the FPGA as a peripheral through its data and address buses. Data sharing between the processor and the XC4103E FPGA is possible by using the 4KB dual-port SRAM.

The Labomat3 platform offers a powerful set of communication interfaces to configure and control the whole board. RTEMS [9], a pre-emptive multitasking operating system runs on the platform. It contains Ethernet drivers and provides a TCP/IP stack, which allows usage of high-level standard TCP/IP based protocols.

4.2. Network services

A network architecture description of the Labomat3 platform remote control is published elsewhere [10]. Based on this approach, a server program executed on the platform by the Motorola processor waits for commands from the client application, so all possibilities of the reconfigurable platform Labomat3 are available through a Web browser:

- When the server program receives a command with a new microprocessor design, it loads the XC4013E FPGA with the new configuration data sent by the client.
- If the server receives a command with a test program, it will be stored in the board RAM memory, making it available for later use. After a load application command, the selected data and program code are loaded into the dual-port SRAM memory. Two Kbytes of its total amount are used for code and the rest two Kbytes for data. At this moment, the application is available to be executed by the designed microprocessor.
- When a step-by-step execution command is received, the server program generates a clock pulse which causes the execution of one microprocessor instruction, and the new PC value is obtained through the address bus of the code memory, sending this information to the client. The two Kbytes of the data memory are read and sent to the client, allowing the user to check changes. The use of a double port memory simplifies these operations, because it is possible to read the data memory without interrupting the microprocessor execution or adding extra logic into the FPGA.
- A run command requests the server program to generate a clock pulse which causes the execution of one instruction and the new PC value is obtained. However, in this case, the user is not informed of changes in memory. The server program checks if a breakpoint is associated with the current PC value or a HALT is the next instruction to be executed. While these conditions are not satisfied, the server program generates new clock pulses. Otherwise, the program counter and the data memory value are returned to the client.
- The multi-step command is based on the last one, but after each cycle, the PC and the data memory content are sent to the client. The microprocessor runs the program step-by-step until the user sends a Stop command.

5. Evaluation

At this early stage of the computer architecture program, no formal course and teaching evaluation were taken. All students interested to participate in the Microprocessor prototypes were assigned to a "focus groups". These groups have been regularly meeting with the academic staff members. Useful discussion have taken place about the deficiencies in contents or not well explained concepts.

The focus groups indicate that the experience was very popular with students and the desired outcome has been achieved. No problems with the use of the MicroDebug tool have been reported by the students. Some suggestions have been received. They are related with the possibility of display the contents of the internal microprocessor registers during the debug execution. Other are concerned with small change in the MicroDebug window presentation. Only a few percentages of students, around 12%, obtained bad results because the design was not correct. Performing a step by step debug is easy to find the error. Many students would also modify their designs in order to improve some partial features. The good results obtained, indicate that a formal course will be implemented for the following year.

6. Conclusions and future work

This work offers an interesting method to introduce experimentation into CS courses. Students are able to go beyond simulation, by testing their microprocessor designs on hardware, and interacting with a real system.

Learning through experimentation is highly beneficial. Using FPGA - based devices to simulate a microprocessor will give an authentic learning experience. This will increase motivation provided and the students have a high probability to succeed in the tasks assigned.

The web laboratory that allows the remote access to the Labomat3 platform and the synergy of programmable resources has been exploited in the microprocessor debugger tool. A full set of services have been developed to remotely manage all the available resources.

As future work the microprocessor debugger features will be improved. It would be very useful to display all the internal register values and be able to modify its content in execution time. A first approach is the emulation of a monitor program mapping the register on memory. Before a break point, a subroutine saves all register in memory and previously to continue with the execution restores the register current values. This solution avoids the specification of design restriction on the register microprocessor design.

7. References

[1] K. Smith, J. O. Strandman, R. Berntzen, T. A. Fjeldly, M. S. Shur and H. Shen, "Advanced Internet Technology in Laboratory Modules for Distance Learning", *Proc. ASEE Annual Conference (ASEE'01)*, Albuquerque, New Mexico, June (2001).

[2] F.J. Gomez, I.Gonzalez, J. Martinez, H. Effinger, W. Seifert, D.Geoffroy, T. Zimmer, "Cap3: Instrumentation on the Web". LAB ON THE WEB Running Real Electronics Experiments via the Internet, 2003. pp 89-144, Wiley & Sons, ISBN 0-471-41375-5

[3] AhYi-Shiou Wu, Der-Yuh Lin, and Sin-Min Tsai, "A Distance Learning and Diagnostic System for a Digital Circuit Laboratory, Innovations 2003, INNER, Edited by Aung et al, 2003, pp215-223

[4] Cabello R., Gonzalez I., Gomez F.J., and Martinez J., "A Web Laboratory for a Basic Electronics Course", Proceedings of the World conference of the WWW and Internet. WebNet 2001, Vol 1, Orlando USA Oct 2001, pp. 816- 821

[5] Gomez F.J., Cervera M., and Martinez J. "A World Wide Web Based Architecture for the Implementation of a Virtual Laboratory", Proceedings of the 26th Euromicro Workshop On Multimedia And Telecommunications, Vol II, The Netherlands, Sept 2000, pp. 56-62

[6] Retwine: REmoTe Worldwide Instrumentation Network. http://www.retwine.net

[7] Teuscher C., Haenni J.O., Gomez F.J., Restrepo H.F. and Sanchez E. "Labomat3: A Reconfigurable Platform for Academic Purposes". Proceedings of the IEEE Symposium Field-Programmable Custom Computing Machines. FCCM '99. Napa USA. April 1999.pp 282-283.

[8] Motorola MC68360. "Quad Integrated Communication Controller". User's Manual. 1993.

[9] RTEMS Real-Time multi-tasking operating system, OAR Corporation, http://www.rtems.com.

[10] I. Gonzalez, R. Cabello, F. Gomez-Arribas, J. Martinez. "LabomatWeb: A Web Laboratory Based on Reconfigurable Computing Technology". ICEE 2003, Valencia, Spain