Implementation of a hardware and software framework for a simple academic processor

J. Ruiz, D. Guerrero, I. Gomez and J. Viejo
Departamento de Tecnología Electrónica
Universidad de Sevilla
Sevilla, España
jonathanruizpaez@gmail.com, guerre@dte.us.es , igomez@us.es, julian@dte.us.es

Abstract—**An academic processor to be used in the "Computer Structure" subject has been developed in this work. During the lab sessions students will apply their knowledge about digital systems to design and implement this processor so they will interact with a real implementation of the system in several ways: modifying it to increase its functionality, programming it and watching its internal state while executing instructions. The fact that students are able to use a functional implementation improves remarkably their motivation.**

Keywords-trainer; laboratory; digital systems; processors; learning

I. INTRODUCTION

Developing a syllabus requires establishing the main goals of the subject. Once the set of topics to be learned is set, the syllabus summary is structured to teach theses topics in order and progressively. Usually, these topics are introduced in lectures, but a set of lab sessions is also necessary in order to reach the learning objectives. During the lab sessions, students have to manipulate real systems, which enhance their practical skills and clarify the topics learned at class. When teaching electronics, lab sessions are crucial since every lecture refers to the behaviour of real electronic devices. That is why every lab session must be selected carefully.

In this paper a simple academic processor will be introduced. It is used in the introductory subject *computer structure* of the new grades in Computer Engineering of the University of Seville. The lesson describing it precedes others describing commercial processors. The processor is extremely straightforward so it possible to describe comprehensively its programming model and implementation to the students. It also makes it easier to implement a system based on this processor within an FPGA prototyping board as well as developing programming and debugging tools for such a system. Students can use this set of tools in lab sessions to interact with a real implementation of the processor in several ways: modifying it to increase its functionality, programming it and watching its internal state while executing instructions. This approach is more effective than others based on simulation [1]. To carry out the lab sessions students must use the processor instruction set (ISP) and understand the implementation internal behaviour at register transfer level (RT), which requires learning and understanding key topics he will need when dealing with commercial systems. Also, the students' motivation is increased if they can use what they learn to build a functional implementation of the design described during the lectures. [2], [3] and [4] describe the advantages of active learning of microprocessor related topics.

This paper is structured as follows: in the next section the content of the *Computer Structure* subject will be outlined. Then, the academic processor will be described. After that, the proposed lab session will be presented. The hardware and software to be used will be described in section V. Section VI will show the guide to be used by the students during the lab session. Finally, the most relevant conclusions will be resumed.

II. SUBJECT OUTLINE

In order to explain the benefits of using the proposed processor for teaching we must describe the content and goals of the subject. *Computer Structure* is a compulsory subject of the Computer Engineering grades of the University of Seville that is taught during the second half of the first year. Its objective is introducing how computers work internally. To reach such a goal, the following lessons are included in the syllabus:

- Digital systems
- A simple computer design
- Commercial processors

In order to develop these lessons it was necessary to choose the simple computer as well as the commercial system to be used. The commercial system chosen was the ATmega328P microcontroller [5], since it had already been used in lab sessions of others subjects so professors and students were already familiarized with it. Also, there is a lot of didactic information as well as inexpensive educational prototyping boards for this microcontroller [6]. Regarding the simple academic computer to be described in the lectures, it was desirable that it were similar to commercial systems in use nowadays, specially to the chosen ATmega328P. However, it must be taken into account that this is an introductory subject. Hence, the academic processor should not be too complex. None of the commercial architectures suggested was simple enough to be utterly analyzed, implemented and modified by the students. That is why a new processor was designed for that lesson. It was called SAP (for Simple Academic Processor).

III. ARCHITECTURE

The goals when designing the processor architecture were the following:

- Simplifying the user model so it could be easily understood by the students.
- Making the user model similar to the commercial system to be described later.
- Providing a reference implementation of the data processing unit as simple as possible.
- Prioritizing the control unit simplicity (students should be able to implement it).

A. User model

After several modifications requested by the lecturers we got an architecture suitable for their needs. In order to simplify the control and data units, the processor includes the following features:

- There are few visible registers, and they are all of the same size.
- There are few instructions, and all of them have the size of a code memory location.
- There are very few instructions formats, and they share fields.
- Harvard architecture.
- Load/Store architecture.
- Memory mapped I/O so there is no need of special input/output instructions.
- There are no interrupt system, special operation modes, memory management units nor memory caches.

In order to use this processor at low level it is necessary to know the set of visible registers as well as the format and semantic of the instructions. The memory system model is also required. The effect of any instruction is basically the same of it homologous in the AVR8 architecture [5]. The remaining user model is detailed below.

1) Memory and Visible registers

The SAP has a 256x8 data memory space and a 256x16 code memory space. The visible registers are 8-bit wide. Within them, the general purpose ones are labelled R0, R1, R2, R3, R4, R5, R6 and R7. The special purpose registers are the following:

- PC (Program Counter): It contains the address of the next instruction to be executed. Its initial value is zero, and it is incremented every time a non-branch instruction is executed.
- SR (Status Register): Only four of its bits are used. There are labelled Z (Zero), V (oVerflow), N (Negative) and C (Carry). They indicate, respectively, if the result of the last logic/arithmetic instruction executed was zero, that it can not be represented assuming two's-complement notation, that it is negative assuming two's-complement notation or that it can not be represented assuming unsigned notation.
- SP (Stack Pointer): It is used to implement the call stack. It contains the address of the next available memory location for the stack, and its initial value is $FF. It is decremented when return address are stacked. When they are retrieved, SP is incremented.

2) Instruction set

The SAP assembler instruction set is intended to be a very small subset of the AVR8 ISP. It does not imply that their machine code format is similar. In fact, the SAP machine code formats are far simpler, as shown in table I. Their opcode fields are always 5 bit width, so there are 32 available operation codes. They have been assigned in such a way that instruction decoding is extremely straightforward. Also, the AVR8 assembler mnemonics have been employed so learning to use the ATmega microcontroller of the following lesson would be easier.

B. Proposed implementation

In order to implement the processor architecture, it is necessary to define its internal and external structure. The data processing unit proposed to the students is depicted in Fig. 1. The general purpose registers have been implemented a little synchronous RAM (register file) with two read ports and one write port. An 8-bit ALU carries out the arithmetic/logic operations. Most registers are interconnected by a common data bus. Also, four hidden registers have been introduced: the 16-bit IR register stores the instruction being executed; the 8-bit MAR register address the data memory; the 8-bit MDR register is an interface between the data memory bus and the internal data bus; the 8-bit AC register stores the ALU results. Respecting the control unit, it has the following input/output signals:

- W_{MEM} (output): Used to request a data memory write.
- R_{MEM} (output): Used to request a data memory read.
- STOP (output): Used to indicate if the processor is running.

TABLE I. SAP INSTRUCTION FORMATS

<table>
<tr><th>format</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>A
instruction with register operands only</td><td colspan="5" rowspan="3">operation code</td><td colspan="3" rowspan="2">target register (source in ST)</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td colspan="3">source register (base register in ST/LD)</td></tr>
<tr><td>B
instruction with memory / immediate operand</td><td colspan="10">immediate data / memory address</td></tr>
<tr><td>C
jump instruction</td><td colspan="3">jump condition</td><td colspan="10">target jump address</td></tr>
</table>

- W_{AC}, W_{REG}, W_{PC}, W_{MAR} (output): Used to order a write operation to AC, the register file, PC or MAR respectively.
- I/O^*_{MDR} (output): When this signal is 1, MDR puts its content on the internal data bus. When it is 0, MDR puts its content on the external data memory bus.
- W_{MDR} (output): Used to order MDR to be written with the data at the internal data bus (if I/O^*_{MDR} is equal to 1) or with the data at the external data memory bus (if I/O^*_{MDR} is equal to 0).
- W_{IR} (output): Used to order a write operation to IR.
- W_{SR} (output): Used to order a write operation to SR.
- I_{SP}, I_{PC} (output): Used to order an increment operation to SP and PC respectively.
- D_{SP}, C_{SP} (output): Used to order a decrement and clear operation to SP respectively.
- R_{AC}, R_{PC}, R_{SP} (output): Used to order to put their content on the internal data bus to AC, PC and SP respectively.
- INM (output): When this signal is 1, the data input B of the ALU is fed with the bits from 0 to 7 of IR (immediate operand). Otherwise it is fed with the content of the general purpose register labelled with the number written at the bit 0, 1 and 2 of IR.
- OP_{3-0} (output): Used to tell the ALU the arithmetic/logic instruction to be performed.
- I_{15-8} (input): They are connected to the corresponding bits of IR and inform about the operation code as well as the jump condition.
- START (input): Used to start the execution.

Respecting the external structure of the SAP, it includes the following input/output signals:

- CODE_ADD (output): Used to address the code memory. It is supposed to have unconditional output.
- DATA_ADD (output): Used to address the data memory.
- CODE (input): They are connected to the code memory output.
- DATA (input/output): They are connected to the data memory bus.

IV. Proposed Lab Session

A set of compulsory lab sessions covering several topics of the subject are carried out. In the proposed lab session students must implement and program the SAP. This implies learning many basic topics introduced during the lectures, specifically the following:

- How a digital system composed of data and control units works.
- Design and implementation skills.
- How to implement and test in the field a system at RT and ISP level.

Once the session goals are set, its content must be detailed. As in the other sessions, the session wording is provided to the students in advance. The wordings include questions that must be answered by the students before they carry out each session. In this way they have to study the topics related to each session previously. For this proposed session, students must write simple assembler programs using the original SAP instruction set, modify the SAP control unit to add new instructions and rewrite the previous programs using the new instructions. To do so they have to modify the provided Verilog description. During the session the students test the modifications executing the programs they wrote.

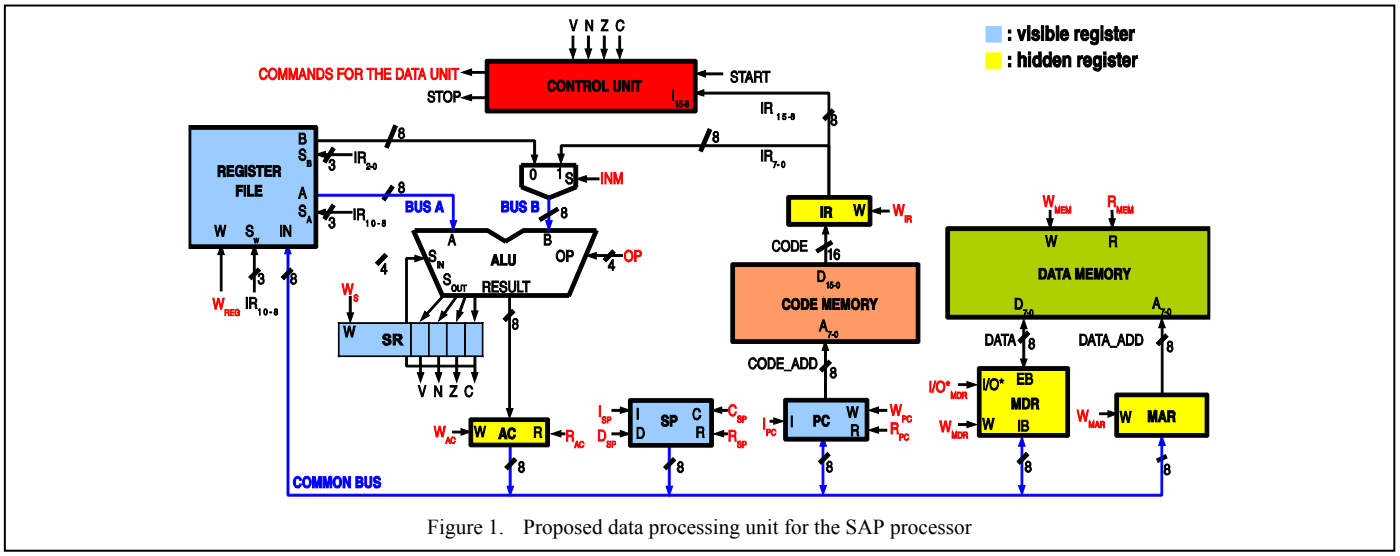

Figure 1. Proposed data processing unit for the SAP processor

V. LAB SESSION SETUP

A. Hardware

The Basys2 [7] FPGA prototyping board shown in Fig. 2 was chosen to implement the SAP since it had already been used in other lab sessions so professors and students were already used to it. The board includes a Xilinx [8] Spartan-3E FPGA. A full system based on the SAP processor was described in Verilog, that is de HDL thought during the lectures. The system includes the following components:

- The SAP processor.
- Memory mapped I/O devices.
- The code memory.
- The data memory.
- A debugging unit.

It must be remarked that it is not possible to implement the chosen platform the data unit exactly as it was described in the lectures using the chosen platform. For example, the Spartan-3E FPGA can not implement tri-state buses. This is not relevant for the students, since they must focus in the control unit design, and the interface lines with the data processing unit do not change. Respecting the I/O devices, eight of them available in the prototyping board were mapped in the SAP data memory space. The external memories are basically as the ones described during the lectures, but they have additional ports to be used by the debugging unit. This unit communicates with a PC through a serial port so the computer can read and write both memories. Also, the debugging unit makes it possible to debug the uploaded programs with the following options:

- Cycle-by-cycle execution.
- Instruction-by-instruction execution.
- Memory and register inspection.
- Control lines inspection.
- Processor clock enabling/disabling.

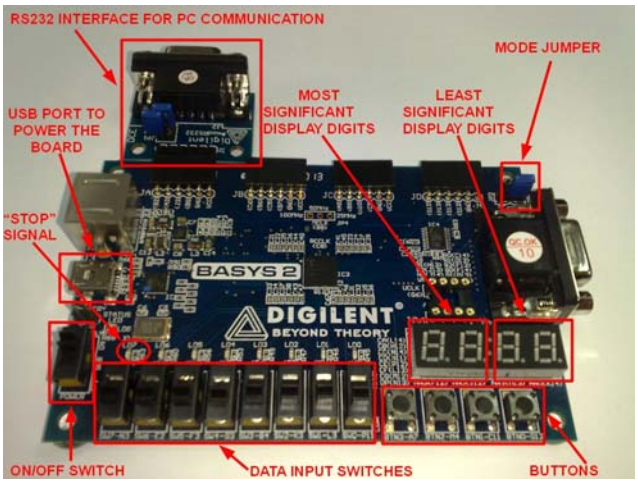

Figure 2. Prototyping board to be used during the lab session

B. Software

In addition to a cross-assembler and a cross-disassenmbler for the SAP, it was necessary to develop a program for the lab PCs to interact with the debugging unit. Its graphical interface is shown in Fig. 3. The upper part of the window contains a set of controls to interact with the CS2010 debug unit:

- "Puerto Serie" field: This field contains the path of the PC serial port connected to the board. Students do not need to change the default value.
- "Código (BIN)" field: This field contains the path of the binary file whose content will be written in the program memory when the processor is initialized. After filling it, the code will be disassembled and shown on the "Desensamblado" field. Normally there is no need to fill this field since assembler files are used instead.
- "Código (ASM)" field: This field must contain the path of the assembler file corresponding to the program that will be written in the code memory when the processor is initialized. After filling it, the front-end assembles the file informing about possible mistakes and writes the path of the resulting binary file in the "Código (BIN)" field.
- "Memoria Datos" field: if the data memory has to be initialized with the content of a binary file, its path must be written in this field.
- "Conectar" button: when this button is pressed the software establishes a connection with the debug unit through the serial port, orders to initialize the SAP and to write the program and data memories with the content of the specified files. Only after this it is possible further interaction with the debug unit.
- "Mostrar Unidad De Datos" button: when this button is pressed, a window with a picture of the SAP data unit appears. During cycle-to-cycle execution the active (high) signals will be highlighted in this window.
- "Habilitar Reloj/Deshabilitar Reloj" button: If the SAP clock is disabled, pressing this button will enable it so the program can be executed autonomously. If the clock is enabled, pressing this button will disable it to make it possible to inspect the state of the processor and the memory.
- "Activar Start" button: When this button is pressed, a high pulse is generated in the start signal of the SAP.
- "Ejecutar Un Ciclo" button: If this button is pressed, the debug unit will enable the clock for just a cycle.
- "Ejecutar Una Instrucción" button: If this button is pressed, the debug unit will enable the clock till the current instruction finishes or till the wait state is reached.

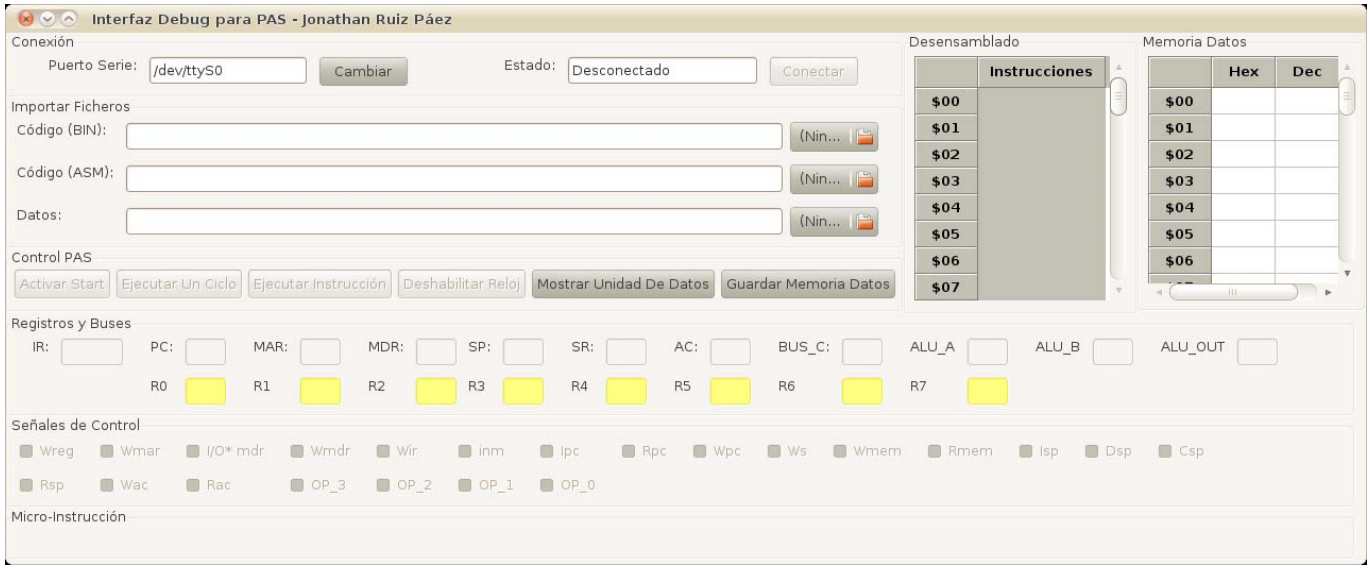

Figure 3. Graphic interface of the software to be used to program and debug the SAP processor

When the SAP clock is disabled, the system state is shown including the code and data memories content, the registers content, the control lines state and the value of the buses connected to the ALU. Also, the current instruction will be highlighted. The content of the data memory can be edited manually. It is also possible to save the content of the data memory in a file by pressing the "Guardar Memoria Datos" button.

VI. LAB SESSION EXECUTION

As mentioned, during the lab session each student must test his SAP modifications by executing programs he has previously written. He is given a set of Verilog files that make it possible to implement the SAP processor in the prototyping board as well as a detailed guide describing the steps he must follow. These steps are grouped in the two parts described bellow:

A. Using the original architecture

In this part students have to configure the board so the bitstream written in its non-volatile memory will be loaded into the FPGA at startup. This is the bitstream for the original implementation of the SAP processor. After feeding the board and connecting it to the serial port of the PC, students must write into the code memory their programs for the original architecture and check if they work as expected.

B. Using the modified architecture

In this part students must implement and check their modified SAP architecture. To do so they have to use the Xilinx ISE [9] FPGA development environment. This tool is used in previous lab sessions. After including their modified version of the control unit and implementing the system in the FPGA, they have to write into the code memory the programs that use the new instructions and check if they work as expected.

VII. CONCLUSIONS

A simple academic processor has been developed to be used in the in the introductory subject *computer structure* of the new grades in Computer Engineering. The objective of this subject is showing how system based on microprocessors works, so this contribution covers many of its topics. In order to complement the lectures, a lab session for implementing, modifying and programming the processor has been developed so students can interact with it and analyze its state while executing instructions. For this session a synthesizable Verilog description of a system based on this processor has been developed, as well as a set of software tools to interact with it. Making it possible for the students to implement the system described during the lectures and to interact with such implementation increases remarkably their learning motivation.

ACKNOWLEDGEMENTS

This work has been partially supported by the Ministerio de Ciencia e Innovación of the Spanish Government under project TEC2011-27936 (HIPERSYS) and by the European Regional Development Found (ERDF).

REFERENCES

[1] Sandro Neves Soares and Flávio Rech Wagner. T&D-Bench—Innovative Combined Support for Education and Research in Computer Architecture and Embedded Systems . IEEE Transactions on Education. Vol. 54. No. 4. November 2011.

[2] Antonio Carpeño, Jesús Arriaga, Javier Corredor, and Javier Hernández .The Key Factors of an Active Learning Method in a Microprocessors Course. IEEE Transactions on Education. Vol. 54, No.2, May 2011.

[3] Debiec, P.; Byczuk, M.Teaching Discrete and Programmable Logic Design Techniques Using a Single Laboratory Board. IEEE Transactions on Education. Vol. 54. No. 4. November 2011.

[4] Kim, J. An Ill-Structured PBL-Based Microprocessor Course Without Formal Laboratory. IEEE Transactions on Education.Vol. 55, No.1. February 2012.

[5] Atmel ATmega328P datasheet, http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf

[6] Arduino homepage, http://arduino.cc/

[7] Digilent Basys2 Board Reference Manual, http://www.digilentinc.com/Data/Products/BASYS2/Basys2_rm.pdf

[8] Spartan-3 Generation FPGA User Guide, http://www.xilinx.com/support/documentation/user_guides/ug331.pdf

[9] Xilinx ISE 11 User Guides, http://www.xilinx.com/support/documentation/dt_ise11-1_userguides.htm