Educational applications of a pico-processor design

Jiménez-Fernández, Carlos J.;Baena, Carmen; Parra, Pilar; Valencia, Manuel
Dpto. de Tecnología Electrónica, Universidad de Sevilla / Instituto de Microelectrónica de Sevilla (Universidad de Sevilla / CSIC)
Sevilla, España
cjesus@us.es

Lopéz-Hinojo, Antonio A.
Universidad de Sevilla
Sevilla, España

***Abstract*—Knowledge of the internal structure and operating mechanism of microprocessors is a very important part in for engineers in electronics and computer science. This knowledge can be deepened with experiences of processor design, which also meet many aspects linked to other basic skills. However, due to its complexity, the design of commercial processors is not effective from an educational point of view. In this communication we present a VHDL design experience of a very simple processor that shows multiple learning posed to the student.**

Keywords—VHDL; processors design; digital design teaching.

I. INTRODUCCIÓN

Digital Electronics is a field so large that its teaching is carried out in different subjects. The first subjects tend are usually focused in logic design, with logic gates and flip-flops. From there, other subjects engaged in teaching digital design from higher levels of abstraction, such the RT level. There are also subjects dedicated to the teaching of the use of processors, in general following the process of programming of a particular processor. Other more advanced subjects engaged in the realization of digital design using hardware description languages (HDLs). It is quite frequent that subjects dedicated to the teaching of the use of the processors are no closely related with subjects devoted to the design with HDLs.

For this it is very helpful to find examples of applications that combine the contents taught in several subjects, and provide to the student a global vision of the process of design of digital circuits. From this point of view, the design of processors in a very good example, because it can combine the contents of subjects related to the use of processors and to subjects linked to the design using HDLs. However the complexity of current processors, even the simplest, makes impractical its realization as a complete design.

With this background, this communication presents a experience design of an academic processor called pico-processor for its simplicity. This experience has been the design and documentation of a library of basic cells, necessary for the design of processors, and their instantiation and interconnection to design and verify the pico-processor. This experience has been carried out as a Degree Project [1].

The description of the basic cell and the pico-processor has been carried out using VHDL hardware description language. These descriptions have been made at RT level and complying with the restrictions of synthesis tools. So obtained circuits can be synthesized and implemented using technology libraries.

This paper has given great importance to the documentation. A good documentation of a design is an extremely important aspect, both in its development and its subsequent maintenance. A design documentation starts at the same time that the construction and ends just before delivery to the customer. Once the design is complete, the documents to be delivered are a technical guide, a user guide and an installation guide.

There are many problems caused by poor documentation or lack of it. Any change in design can result in having to review it completely, this leads to a considerable waste of time and makes its maintenance becomes virtually impossible.

Thus for each of the developed cells a datasheet has been created, which includes all aspects of operation and interconnection required to use it. Thus the library of cells can be used easily in other designs, other microprocessors or other types of circuits.

The outline of this presentation is as follows: The following section provides a detail description of the processor to be designed, in section III the cells designed are presented, in section IV construction and verification of the processor is explained and finally some conclusions are extracted.

978-1-5090-2264-9/16/$31.00 ©2016 IEEE

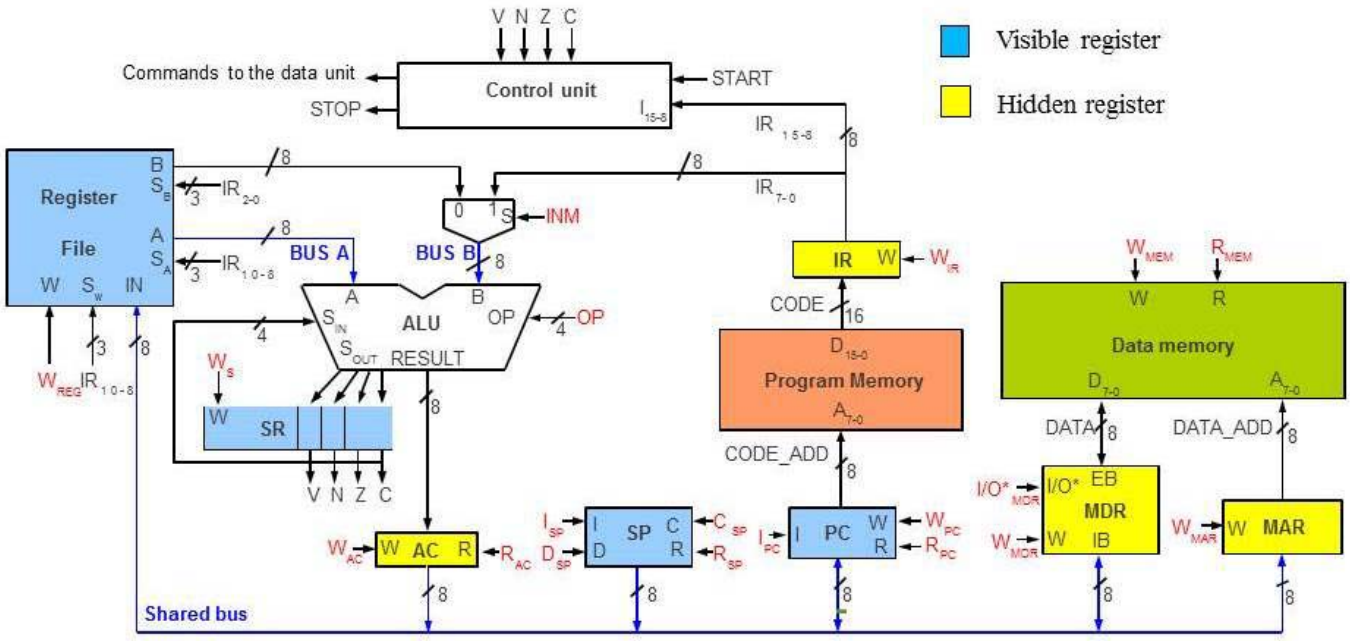

Figure 1. Hardware Architecture of the pico-processor.

II. Description of the Pico-Processor

The processor being designed has Harvard architecture, since has a memory for data and other memory for instructions, and a reduced instruction set (RISC). This processor, because is very simple, do not have segmentation, i.e. do not have stages of pipeline to execute instructions in parallel. The following paragraphs describe in detail the architecture and instruction set of the processor.

A. *Processor architecture*

The design we are presenting is an "academic" processor [2] but with the instruction set compatible with a real microcontroller (Atmel AVR). The processor architecture is shown in Figure 1.

The architecture has the following basic components:

- A Register File containing 8 general purpose data registers. This block interacts with the ALU supplying up to two data through the bus A and B. On the other hand, it also receives data through the shared internal bus. Data B comes from the register selected by the three less significant bits of the Instructions Register ($IR_{2:0}$) while the data A comes from the register selected by $IR_{10:8}$, that select the target register too.
- Six registers of specific purpose: the Program Counter (PC) containing the address of the next instruction to be executed; the Instruction Register (IR), which contains the current instruction; the Status Register (SR), which gives information about the result of the last operation performed in the ALU; The Stack Pointer (SP) which is a register that contains the address of the first free position in the stack; Data Memory Register (MDR) containing the word to be written or read from the data memory and Memory Address Register (MAR) containing the address which is accessed in the data memory.
- A data memory, with 256 words of 8 bits which is accessed via an address bus and a bidirectional data bus. Reading or writing in the memory is controlled by two control signals, W_{MEM} and R_{MEM}.
- •An arithmetic logic unit (ALU) with operations of addition, subtraction, shifts to the right or left of the input data and transfer of the inputs to the output. It also has State Outputs which give information about the result of the last operation: Z if it is zero, V if there has been overflow in numbers with sign, N if it is negative and C if carry or borrow has occurred.
- A Control Unit that adequately perform the micro-operations sequence for each processor instruction.

B. *Instruction set*

This processor instruction set includes operations of access to registry and memory, jump, State and arithmetic or logic. Figure 2 shows the list of sorted by their machine code instructions including their descriptions in mnemonic and at RT level.

The instruction set of this processor includes register and memory instructions, jump instructions, and arithmetic or logic instructions. Figure 2 shows the list of instructions sorted by

Instruction set (ordered by the operation code, COP)					
COP IR_{15} : IR_{11}	For-mat	Mnemonic And syntax	Type	Effect	V N Z C [1]
0 0000	A	ST (Rb), Rf	Mem	M(Rb) ← Rf	- - - -
0 0001	A	LD Rd, (Rb)	Mem	Rd ← M(Rb)	- - - -
0 0010	B	STS dir, Rf	Mem	M(dir) ← Rf	- - - -
0 0011	B	LDS Rd, dir	Mem	Rd ← M(dir)	- - - -
0 0100	C	CALL dir	Jump	M(SP)←PC; SP←SP-1; PC←dir	- - - -
0 0101	-	RET	Jump	PC←M(SP+1); SP←SP+1	- - - -
0 0110	C	BRxx dir	Jump	Xx: PC ← dir	- - - -
0 0111	C	JMP dir	Jump	PC ← dir	- - - -
0 1000	A	ADD Rd, Rf	Arit/Log	Rd ← Rd + Rf	* * * *
0 1010	A	SUB Rd, Rf	Arit/Log	Rd ← Rd - Rf	* * * *
0 1011	A	CP Rd, Rf	State	NOP [Rd –Rf ⇒ VNCZ]	* * * *
0 1111	A	MOV Rd, Rf	Mov	Rd ← Rf	- - - -
1 0010	-	CLC	State	NOP [⇒ C←0]	- - - 0
1 0011	-	SEC	State	NOP [⇒ C←1]	- - - 1
1 0100	A/B	ROR Rd	Shift	Rd ← SHR(Rd, C) [⇒ C←Rd0]	* * * Rd0
1 0101	A/B	ROL Rd	Shift	Rd ← SHL(Rd, C) [⇒ C←Rd7]	* * * Rd7
1 0111	-	STOP	Special		- - - -
1 1000	B	ADDI Rd, dato	Arit/Log	Rd ← Rd + dato	* * * *
1 1010	B	SUBI Rd, dato	Arit/Log	Rd ← Rd - dato	* * * *
1 1011	B	CPI Rd, dato	State	NOP [Rd –dato ⇒ VNCZ]	* * * *
1 1111	B	LDI Rd, dato	Mem	Rd ← dato	- - - -
COPs without use: $(9, C, D, E, 10, 11, 16, 19, 1C, 1D, 1E)			[1] The character "-" means "without modification"; the character "*" means "modified"; other COP, not documented.		

Figure 2. Instruction set of the pico-processor.

machine code and including their descriptions in mnemonic and RT level.

Instructions only have a word of machine code and consist of a total of 16 bits. The five most significant bits contain the operation code and the eleven remaining bits contain information for locating the operands. This information may have different formats which are shown in Figure 3, which lists the fields and the function of each of them.

The addressing modes supported are the following:

- •Register Direct: in the instruction it is included the register address with the data.
- •Memory Direct (or Absolute): in the instruction it is included the memory address with the data.
- Indirect mode: the effective address of the operand is the contents of a register or main memory location, location whose address appears in the instruction.
- Immediate mode: the operand is an immediate value is stored explicitly in the instruction.

Figure 3. Instructions format.

There are three types of jump instructions:

- Unconditional.
- •Conditional, in which the jump will take place depending on some state bits values. In this pico-processor conditions are zero result, less than (for magnitudes and for sign data), and overflow after operate with signed numbers.
- Subroutine, jump to and from a subroutine.

C. Control Unit

Finally, the processor has a control unit responsible for selecting and timing the different micro-operations to be performed in the execution of each instruction. The description of this control unit is made through an ASM chart [3]. However its complexity is quite large, so it will not be reproduced here in full. For example, Figure 4 shows ASM chart of the selection and execution of instructions from program memory. In this ASM chart three types of instructions, data management, memory access and jumping have different executions. Each of these types of instructions is executed according to a description in the form of an ASM chart which have not been included here.

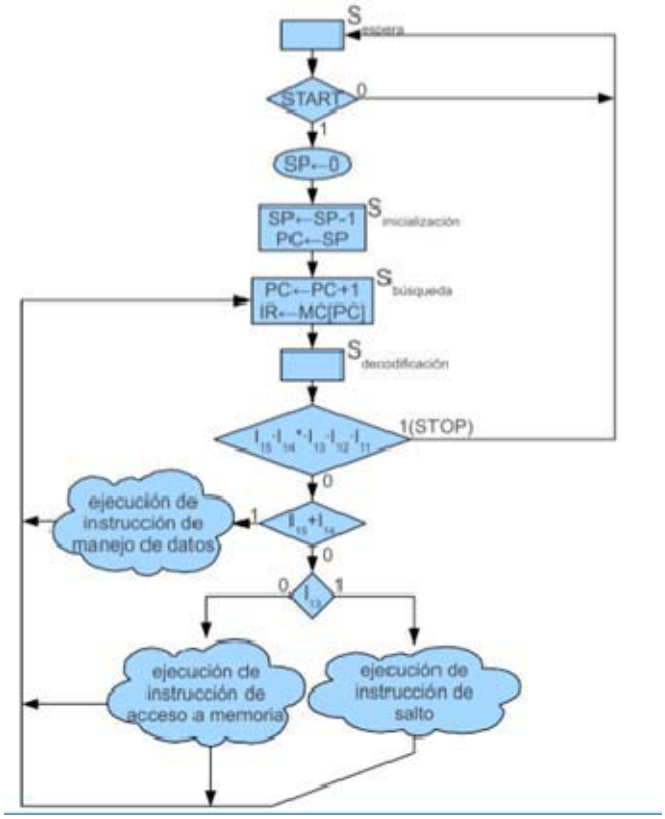

Figura 4. ASM chart of the Control Unit.

III. Development and Documentation of a Cell Library

In the work of the processor design, the first part of the work consisted of the selection, description and verification of the cells required for the construction of the architecture of the pico-processor. This selection was made based on the blocks that present the architecture and that is showed in Figure 1. Table I shows the list of developed cells.

Each of the cells was described using the hardware description language VHDL and simulated using Modelsim software. As can be concluded seeing the processor architecture, there is no particular difficulty in the design and verification of the cells. Much of the cells correspond to registers with small variations in performance. There is a registers file (eight 8-bit registers with two inputs for reading and one for writing), and the rest are 8-bit registers, which differ in the way they access and if they have the increase capacity. The only not common situation is the use of a shared bus, forcing the use of bidirectional lines in some registers and therefore the use of high impedance signals.

TABLE I. Library of Cells

Name	Description
PC	Program Counter register
MEMCOD	Program Memory
IR	Instruction Register
ALU	Arithmetic Logic Unit
REGISTRO	Register File
MULTIPLEXOR	ALU Multiplexor input
MEMDAT	Data Memory
MDR	Memory Data Register
MAR	Memory Address Register
AC	ALU output register
SR	State Register
SP	Stack Pointer

As already mentioned in the introduction, one of the important aspects of this work is the creation of a documentation that would allow the reuse of the designed cells. This requires a data sheet for each of the developed cells. To select the information they should contain it was consulted data sheets of cells of different technological libraries. Many of these data sheets include information relating to technological aspects (delay time, behavior for different voltages, etc.). These issues do not apply in our case, since we present descriptions are technology independent. The documentation carried out includes information relating to aspects of interface and functional, including:

- Symbol.
- Introduction (textual description of its operation).
- Forma description of their RTL behavior.
- Inputs and outputs and their description.

In addition to the description of each of these cells the description of the control unit was performed. This is the most complex of all blocks. For its design it was followed the guidelines for the design of state machines in hardware description languages.

IV. Design of the Pico-Processor

The next step in the design process consisted in the construction of the pico-processor. Because of the

methodology followed, the description of each of the blocks that make up the processor as basic cells, the processor design consisted of instantiation and interconnection of all components.

The verification process was conducted in two phases: in a first phase the correct execution of the instructions was tested: transfer of data from memory to registers and from registers to memory, performing operations by the ALU, etc. In a second phase the verification was carried out checking program implementation. In this paper we present one of the tests. The example presented is a little routine to keep in a register the largest data stored in two other registers. In this case the data are signed numbers.

A RT level operation is as follows:

$$R0 \leftarrow \text{Max}\ (R1, R2)$$

The routine that performs this operation is as follows:

```
          LDI R1, $48 ; data to R1
          LDI R2, $D1 ; data to R2
MAX:      CP R1, R2   ; Compare
          BRLT R2MAYOR; Is R1 less?
          MOV R0, R1  ; No. Return R1
          RET         ; Return
R2GREAT:  MOV R0, R2  ; Return R2
          RET         ; Return
```

In this routine, first are charged the number $48 in the register R1 and the number $D1 in the register R2. Then begins the subroutine (label MAX) with the CP command, to make the comparison. This instruction performs the subtraction R1 - R2 but does not store the result, only updates the State Register. The next instruction (BRLT, branch if less than) makes the jump to the label if the status log indicates that the result of the subtraction have been negative (N⊕V). However it must be noted that the data subtracted are signed numbers (coded 2's complement). Depending on the result of the comparison the value of R1 (if not jump) or R2 is saved in R0 if the conditional jump occurs, and then returned to the main program.

Once this routine has been made, it must be translate to the machine code to be programmed in the program memory. The result is shown in Table II.

Una vez realizada esta rutina hay que pasarla al código máquina que hay que introducir en la memoria de programa. El resultado se muestra en la tabla II.

TABLE II. PROGRAM FOR TEST.

Position	Machine Code
$00	11111 001 01001000
$01	11111 010 11010001
$02	01011 000 00000001
$03	00110 011 00000110
$04	01111 000 00000001
$05	00101 000 00000000
$06	01111 000 00000010
$07	00101 000 00000000

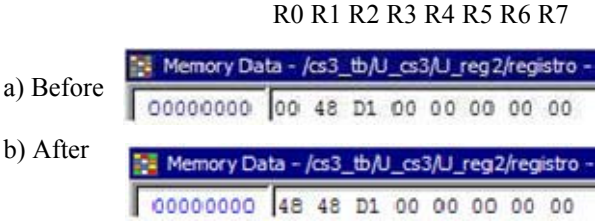

Figure 5. Internal register data, a) before and b) after the execution of the subroutine.

Figure 5 shows the contents of the registers before run the subroutine MAX and after its execution. As it can be seen in the results, the part (a) of the figure show the data stored in the registers after loading R1 and R2 (R0 remains at $00), and part (b) shows the data stored in the registers once the execution of the routine. In the register R0 appears stored value $48 ($48 = $72_{(10)}$), which is the largest of the two, since as they are encoded in 2's complement, $D1 data corresponds to a negative number ($D1 = - $47_{(10)}$).

V. CONCLUSSIONS

This communication presents design experience of design of a pico-processor for educational applications. Experience, which has been conducted as a Final Degree Project, consisted in the design, verification and documentation of a library of cells, which are the basis for the construction of simple processors. In addition, using this library of cells a pico-processor has been built, and has also been verified.

Descriptions, both the cells and the processor, have been carried out in VHDL at RT level and in compliance with the restrictions of synthesis tools, so that the design can be synthesized and implemented in technologies both as FPGA ASIC.

This work has led to learning in various disciplines: on the one hand it has been necessary to know aspects of digital design at RT level using HDLs, but also includes aspects of architecture of processors. Finally the importance given to the documentation allows the reuse of the cell library developed.

REFERENCES

[1] Antonio Alberto López Hinojo, “Diseño y docuementación de una librería de celdas para el diseño de procesadores”, Final Degree Proyect, Ingeniería Técnica Industrial, especialidad en Electrónica Industrial, Escuela Politécnica Superior, University of Seville, september de 2015.

[2] D. Guerrero and other members of Electronic Technology Deparment, University of Seville. “Computador simple 2010”. Apuntes de “Estructura de computadores” de los Grados en Ingeniería Informática. 2010.

[3] D. Green: "Modern Logic Design". Addison-Wesley. 1986.