Indian Journal of Science and Technology, Vol 10(22), DOI: 10.17485/ijst/2017/v10i22/114827, June 2017
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Implementation of an 8-Bit Softcore Microcontroller on Xilinx Spartan FPGA Family

Julio Enrique Rodríguez Prieto, Edwar Jacinto Gómez and Fernando Martínez Santa*

Technological Faculty, Universidad Distrital Francisco José de Caldas, Bogotá D.C., Colombia;
juerodriguezp@correo.udistrital.edu.co, ejacintog@udistrital.edu.co, fmartinezs@udistrital.edu.co

Abstract

Objectives: To design an 8-bit RISC microcontroller in VHDL and implement it on an FPGA, taking as a reference the mid-range microcontroller core of Microchip 16FXXX. **Methods/Analysis:** This microcontroller has been designed using modular blocks and correspond Harvard-type architecture. It has a stack of eight levels and an Arithmetic Logic Unit (ALU) for operations with 8-bit data and bit-oriented instructions. Also, it has a program memory of 8Kx14-bit and a data memory of 512x8-bit. The number of storage locations of these memory modules is easily scalable and/or modified as needed by the designer due to high abstraction level which has been used in its description. Similarly, the other blocks (as well as the microcontroller in general) are easily modified, providing a modular system resulting in excellent versatility to be adapted to be additional modules including: input and output ports, timers, blocks for communication among others. **Findings:** The proposed device has an instruction OPCODE correspondent to PIC16FXX processor core, so the system can be programmed with the same machine language. It also presents a high support for compilers such as MPASM (Microchip Assembler) and CCS PIC-C (C language). The compatibility with the C compiler is quite interesting, making easy the development of large-scale digital designs, thanks to the existence of extensive libraries in this programming language. **Novelty/ Improvements:** The system provides the enough user support. A document was generated with all data and instructions that the designer needs to take into account for executing or modifying the microcontroller design. This support, along with the open source firmware, was published on the Internet, so that communities can access it, adopt it and/or modify it.

Keywords: Embedded Microcontroller, FPGA, Harvard Architecture, SoftCore, VHDL

1. Introduction

Generally, in digital designs, there are different design methods, mainly such as the ones that use sequential and parallel systems features[1,2]. A system based on simultaneous functional features or of parallel type, implies an increment in the amount of hardware used but an impressive performance in the processing speed; as there is an increment in the hardware use, a final block is obtained with a higher economic spending[3,4]. A microcontroller implemented on a FPGA and not made directly as ASIC, carries low cost, due to it is possible to take advantage of the FPGA as development platform or as part of the final prototype. On an ASIC, making a prototype must do in mass and the final result is irreversible; also, its flexibility is low as to the amount and peripheric availability.

Microcontrollers are totally sequential devices, but at the same time they offer low resource spending, lower design and development time than the prototypes based on program able logic devices; which is better for applications where the flow control is irregular. Also, the use of FPGA offers the option of including integrated system that combines microcontrollers and additional hardware for concurrent working[2-5].

The use of microcontrollers in FPGA-based designs, guaranties the optimization in fast systems which need to attend low speed tasks such as slow communication channel reading, keyboards, displays and peripheric management[6,7].

Nowadays there are some works about processors design, mainly the ones that have a didactic focus[8-11]. On the other hand, there are some microprocessor core

*Author for correspondence

versions published on internet[12-21]. Few of those VHDL and Verilog designs have been done keeping correlation with Microchip PIC microcontrollers[18-21]. However, there are interesting proposals that try to establish compatibility, but have some testing errors. An example is the design named risc16f84[18]–a microcontroller described on Verilog and published inwww.opencores.org, which has not the enough support by the author to debug the proposed system.

The idea of this project was born when working with C compilers[16,17] for Microchip microcontrollers (specifically CCS Pic C Compiler®), and used them to program core microcontrollers implemented on FPGAs. That is why a microcontroller with the most of features of a PIC microcontroller of the16fXXX family is wanted to do, to be used on a FPGA.

2. Architecture

It is important to notice, that in order to imitate the architecture and the features of the core of PIC16FXXX, a correspondent analysis was done to understand its working just having as a reference the datasheet of this microcontrollers. Although the concept diagram of the designed device is not the same, it is possible to notice a lot of similarities with PIC microcontrollers. Figure 1 shows the block diagram of all the implemented system.

Harvard architecture was defined for the microcontroller design[22-24], after each module was described, starting by the ALU, following by the registers, RAM, ROM and finishing with the Control Unit. Unidirectional buses were used in the inner data flow, which produce a structure with high working speed and low resource consume cost.

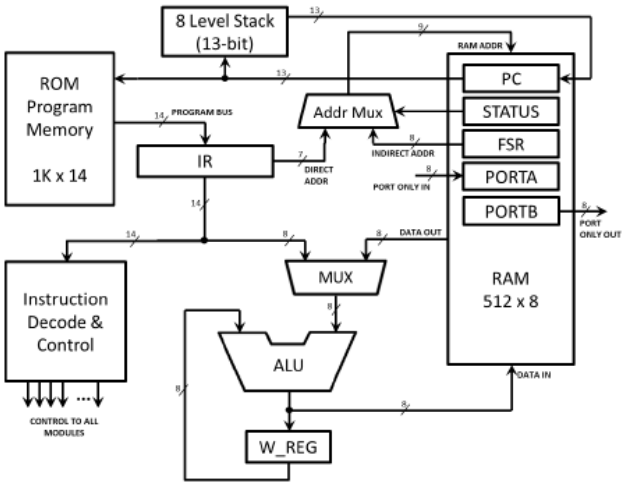

Figure 1. Block diagram of the designed SoftCore microcontroller.

The program counter is implicit in the data memory then que the registers that can be affected by this module do it immediately, likewise other specific purpose registers are located within the RAM as: FSR, STATUS, PORTA, PORTB and the availability to refer new peripherals or additional modules.

It has two non-configurable ports by means of the assembler or C programming, but modifiable from the VHDL description: by default, the device initially has the only input port A (PORTA) and the only output port B (PORTB).

2.1 The ALU

The Arithmetic-Logic Unit was the first designed block. It works doing 8-bit and bit-oriented operations between the accumulator register W and another data that can come from of RAM or the instruction register. It has a carry (c) input for arithmetic and rotating operations, a 4-bit selector for the ALU to decode 18 operations; output flags: carry (c), digit carry (dc) and zero (z), which go to the STATUS register.

2.2 Accumulator W

The accumulator register (Working Register) is a block which loads the8-bitdata from the ALU output and hold them for in the next clock cycle of the same ALU operates that data with a new one from the RAM or the IR.

2.3 Data Multiplexer

It is a combinational circuit that selects one of two possible inputs according to a selection input bit. It selects between data from the IR or the RAM and gives the result to the ALU.

2.4 Instructions Register

It is a register like the Accumulator, but it is a 14-bit register. It receives the instructions from the ROM and hold them to be used by different blocks as the data multiplexer MUX, the address multiplexer ADDR MUX, the ALU and the Control Unit.

2.5 Address Multiplexer

The ADDR MUX selects if a direct or indirect addressing will be done to the data memory. If the addressing is

direct, 7 bits from the OPCODE (Operation code) given by the IR are taken and concatenated with RP1 and RP0 (bits number 6 and 5 respectively of the STATUS register) in order to give an address of 9 bits for the RAM. If on the contrary, the addressing is indirect, 8-bitfrom the FSR register are taken and concatenated with IRP (bit number 7 of the STATUS register) for giving 9-bits to the RAM.

2.6 ROM

The program memory is ad only memory, but it cannot be implemented as Flash memory, due to the inability of the FPGA to support non-volatile memory systems. Wherever, in the ROM 14-bit instruction codes are loaded, these correspond to the instructions of the machine language. The memory is of 8K words like the one of the PIC 16F877 and can be scalable it means its storage capacity can be increased or decreased modifying the VHDL description of the block, reaching the desired storage capacity by the designer.

2.7 RAM

The data memory is of 512 scalable positions, with a data width of 8 bits. Exactly 7 memory positions are reserved for Specific Purpose Registers (SPRs) and the rest can be used as general purpose storage positions. The specific purpose registers keep their original positions in order to be keeping the comb ability even if new modules are included to the microcontroller design. The RAM is a block that contains integrated the program counter along with the rest of the Specific Purpose Registers (SPRs). In fact, the program counter being a 13-bitregister is composed by two 8-bit registers located within the RAM. The chosen method to describe all the integrated memory system, guaranteed update the data in the SPR registers immediately. As shown in Figure 2 the described RAM is composed by additional outputs and inputs, correspondent to the registers and the PC.

It has control inputs (PC_control) and PC data when executing a CALL or GOTO instructions (PC_in) also of the STACK input. The ALU_flags and ALU_loads modify the STATUS register and the PORTA_inis the input peripheral of the microcontroller. Therefore, PORTB is the output port. PC_OUT, STATUS_OUT and FSR_OUT has independent outputs, and the rest of block is a regular RAM with clock input, data input, addressing, and data output.

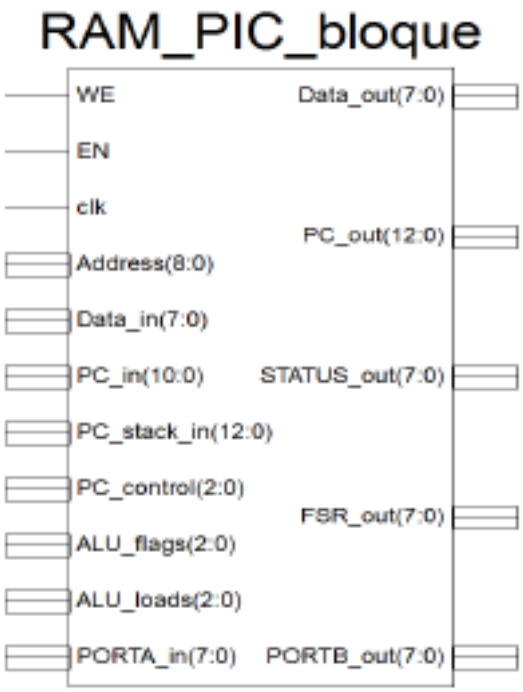

Figure 2. VHDL block of the RAM.

When a device like the PIC 16F84 is implemented using the proposed design, the memory capacity is limited to just 256 bytes that is not even the half of the total capacity, but it is possible to scale to a device like the PIC 16F877 that has implemented 512bytes and thus to take advantage of the data memory capacity. The data memory organization is shown in Figure 3.

The non-assigned registers are available for the addition of new modules by the designer, thanks to the high abstraction level of the description of this module, and thus to keep the programming compatibility with Assembler and C languages. Following each register is described.

- IND: The register 00h (IND) in the same way as a PIC microcontroller is a reference register to call the FSR register and do indirect addressing.
- PCL: Keeps the8 least significant bits of the program counter.
- PCLATH: stores the remaining 5 most significant bits of the program counter.
- STATUS: contains the state of the ALU flags and the configuration bits of direct and indirect addressing. Thus, the bit number 7 of the STATUS, is IRP, bit used in indirect addressing; the bit number 6 and 5, RP1 and RP0 respectively are used in direct addressing;

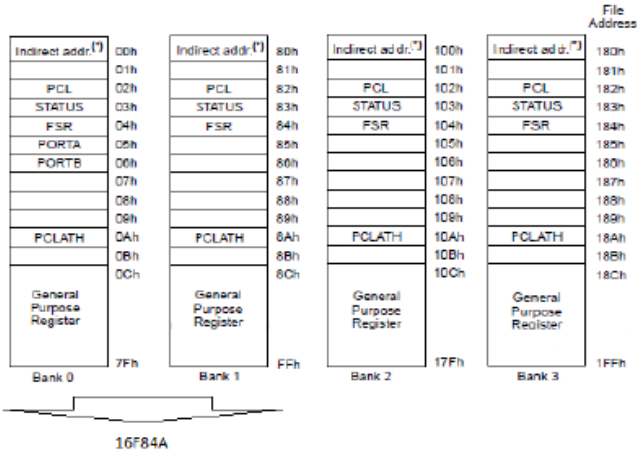

Figure 3. RAM memory organization based on PIC16F84A[25].

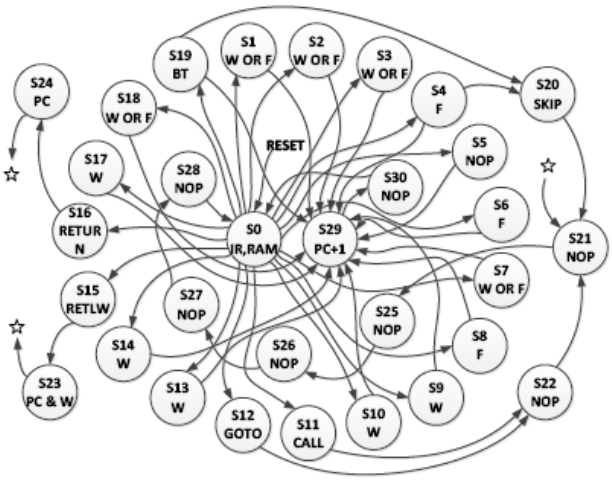

Figure 4. State diagram of the control unit.

the bit 2 corresponds to the Z flag of the ALU; the bit 1 corresponds to DC of the ALU and the bit 0 is the carry flag of the ALU. The bit number 4 and 3 are not used.

- PORTA: Is the register in charge of storing data of the port, for this case (initial setup) is only input.
- PORTB: Is the register in charge of managing data of the output port of the microcontroller (initial setup).

2.8 Stack

The STACK of the program counter is basically a RAM memory of type LIFO (Last In First Out) of 8 positions, for 13-bit data. It stores the value of the program counter when a subroutine is called by means of the instruction CALL and it is red with the instructions RETURN or RETLW.

2.9 Control Unit

The Control Unit is a big Finite State Machine (FSM) that decodes the 14-bit operation code. It controls all the blocks of the system in order to execute the micro-instructions of each instruction. The Figure 4 shows the state diagram of the Control Unit.

2.10 Instructions Set

This system implements a RISC architecture type [15] because it has a reduced number of instructions. Of the total of 35 instructions, 3 are not implemented yet. The microcontroller accepts the same instructions set of the PIC16fXXX microcontrollers family [17], but it does not execute the same action with the instructions: CLRWDT, RETFIE and SLEEP. Due to the absence of blocks dedicated to these instructions, the device does a "not operation" instruction instead, it means, that these instructions are equivalent to the instruction NOP. Following in the Table 1, the accepted instructions by the Soft-core microcontroller are described.

3. Results

According to the state sequence (Figure 4) for the instructions execution, it is possible to notice that the number of micro-instructions in each cycle is 4 in the most of instructions, except in the jump ones which executes 8 states, an equivalent to two work cycles. The most of instructions are executed in 3 states and executes a fourth one where no operation are done; this was done in that way in order to keep the times and being totally compatible with PIC microcontrollers, specifically with the CPU of the 16fXXX family. Then there is ease in the calculus of the delay times and in the general processing.

Finding in this project the expected compatibility both in the processing times and in the structure, it is possible to do applications that use integrated development environments. One of these is CCS PIC-C which is quite versatile thanks to the huge amount of libraries for the managing of external peripherals. These libraries reduce the development time to the designers.

After the VHDL description of each module, these are integrated and interconnected in order to form the final core of the microcontroller (Figure 5). After that, it was

Table 1. Instructions set of the softcore microcontroller, based on PIC16F84A[25].

Instruction Syntax		Data type	Operation	Affected Flags
ADDLW	k	O<=k<=255	(W) + k -> (W)	C, DC, Z
ADDWF	f,d	0<=f<=127; d[0,1]	(W) + (f) -> (destination)	C, DC, Z
ANDLW	k	O<=k<=255	(W) AND (k) -> (W)	Z
ANDWF	f,d	0<=f<=127; d[0,1]	(W) AND (f) -> (destination)	Z
BCF	f,b	0<=f<=127; 0<=b<=7	0 -> (f<b>)	
BSF	f,b	0<=f<=127; 0<=b<=7	1 -> (f<b>)	
BTFSS	f,b	0<=f<=127; 0<=b<=7	PC += 2, if (f<b>)==1	
BTFSC	f,b	0<=f<=127; 0<=b<=7	PC += 2, if (f<b>)==0	
CALL	k	O<=k<=2047	(PC) + 1 ->TOS;k -> (PC<10:O>); (PCLATH<4:3>) -> PC<12:11>	
CLRF	f	0<=f<=127	00h -> (f);1 -> Z	Z
CLRW			00h -> (W);1 -> Z	Z
CLRWDT			Not operation	
COMF	f,d	0<=f<=127; d[0,1]	NOT(f) -> (destination)	Z
DECF	f,d	0<=f<=127; d[0,1]	(f) - 1 -> (destination)	Z
DECFSZ	f,d	0<=f<=127; d[0,1]	(f) - 1 -> (destination); pc += 2, if result == 0	
GOTO	k	O<=k<=2047	k -> PC<10:O>; PCLATH<4:3> -> PC<12:11>	
INCF	f,d	0<=f<=127; d[0,1]	(f) + 1 -> (destination)	Z
INCFSZ	f,d	0<=f<=127; d[0,1]	(f) + 1 -> (destination); pc += 2, if result == 0	
IORLW	k	O<=k<=255	(W) OR k -> (W)	Z
IORWF	f,d	0<=f<=127; d[0,1]	(W) OR (f) -> (destination)	Z
MOVF	f,d	0<=f<=127; d[0,1]	(f) -> (destination)	Z
MOVLW	k	O<=k<=255	k -> (W)	
MOVWF	f	o sf 5 127	(W) -> (f)	
NOP			Not operation	
RETFIE			Not operation	
RETLW	k	O<=k<=255	k -> (W);TOS -> PC	
RETURN			TOS -> PC	
RLF	f,d	0<=f<=127; d[0,1]	(f<6:0>& (carry)) -> (destination); f<7> -> (carry)	C
RRF	f,d	0<=f<=127; d[0,1]	((carry) & f<7:1>) -> (destination); f<0> -> (carry)	C
SLEEP			Not operation	
SUBLW	k	O<=k<=255	k - (W) -> (W)	C, DC, Z
SUBWF	f,d	0<=f<=127; d[0,1]	(f) - (W) -> (destination)	C, DC, Z
SWAPF	f,d	0<=f<=127; d[0,1]	(f<3:0>) -> (destination<7:4>); (f<7:4>) -> (destination<3:0>)	
XORLW	k	O<=k<=255	(W) XOR k -> (W)	Z
XORWF	f,d	0<=f<=127; d[0,1]	(W) XOR (f) -> (destination)	Z

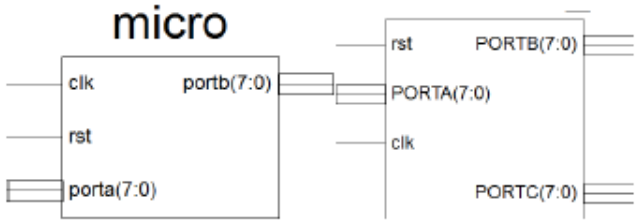

Figure 5. Main schematic of the microcontroller (left) integrated with a peripheral output device (right).

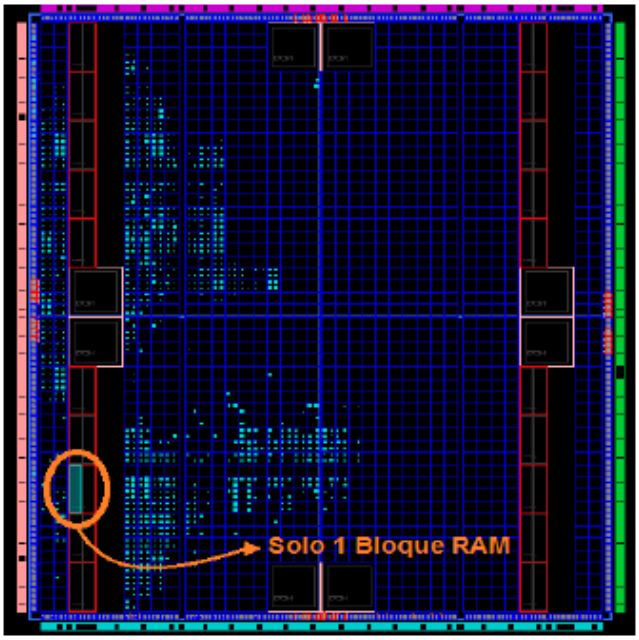

Figure 6. Report of used resources (only 1 block of RAM was used).

implemented in software using a FPGA of the Spartan 3AN family, specifically XC3S700AN. The following features were reached according to the report of Xilinx ISE ver. 12.1, software: 1% of Flip Flops (148 of 11776), 13% of Slices (790 of 5888), 5% of RAM Blocks (1 of 20), Maximum delay: 1070000ns equivalent toa maximum work speed for the microcontroller of 900MHz.Figure 6 shows the graphical report of the used resources within the FPGA according with the tool Net list Design Plan Ahead 12.1.

4. Conclusion

There is in this project, without a doubt, an exploitation of the standard architecture that promises a large compatibility and the use of robust compilers that includes a lot of devices and libraries widely tested and certified. Additionally, communities on internet gives support to these kind of projects, thanks to the use of open code and free distribution of microcontroller design projects. Particularly, this project along with all its source code and documentation was shared in opencores.org website in order to be used or modified by anybody who needs it.

The design of this type of architecture gives the versatility of managing the memory capacity because the dedicated blocks are easily scalable and modifiable from the same VHDL description.

On the other hand, the obtained result spends less resource within the FPGA, thanks to an optimization done developing combinatory blocks as the ALU, which has standard features. Like-wise, as the program memory as the data memory of the microcontroller have been mapped, in only one RAM block of hardware, predefined in the FPGA.

The methodology of this microcontroller guaranties a parallel working with another type of microcontrollers or hardware modules of the same type, in order to be able to use in multitasking mode. This feature plus its reduced size, makes it possible to implement a lot of different microcontrollers in the same FPGA. According to the amount of logic gates of current FPGAs, one design can contain even hundreds of these microcontrollers.

Under no circumstances, the objective of this project has been to replace a microcontroller; instead of this, this offers to the designer some design advantages of using Soft Core microcontrollers (under known and tested architectures), and encourage them to include these devices in their FPGA designs.

At an academic level, this project can be reused as tool for teaching some of the themes of the basic courses or even in advanced levels of subjects related with the digital design. With this system, the students can reach to comprehend of didactic way the behavior of the computational systems and like-wise include them in their own practice designs, taking advantage of FPGA development boards, which have well capacities and integrated peripherals. In the case of designing prototypes, adapted FPGAs can be used, with versatile connection interfaces.

5. References

1. Angelov V, Lindenstruth V. The Educational Processor Sweet-16. In: 2009 International Conference on Field

Programmable Logic and Applications. IEEE; 2009. p. 555–9. Crossref
2. Jaquenod GA. Dise-o de Unmicrocontrolador MC6805 Usandológicaprogramable FLEX de ALTERA. NUEVA Telegr Electron. ARIEL ARBO EDITOR SRL; 2001. p. 130–8.
3. Jaquenod GA, Villagarcía Wanza HA, De Giusti MR, Bria ON. Adaptación Del Núcleo IP De Unprocesadortipo MC6805 Para Operar en un Ambientemultiprocesador y Multitarea. In: VII Congreso Argentino de Ciencias de la Computación; 2001, p. 1–10.
4. Jaquenod GA, De Giusti MR. Dise-o de Microcontroladoresempotradosmedianteprocesamiento Serial: análisisusando FLEX10K para sintetizarunmicrocontroladortipo COP8SAx. In: VII Workshop IBERCHIP Uruguay, 2001, p. 1-11.
5. Zavala HA, Nieto CO, Ruelas HJA, Dominguez CAR. Design of a General Purpose 8-bit RISC Processor for Computer Architecture Learning, Comput y Sist. 2015 Jun 29; 19(2):371–85.
6. Ria-o J, Ladino C, Martínez F. Implementación de la transformada FFT Sobreuna FPGA Orientada a Suaplicación en Convertidoreselectrónicos de Potencia, Tekhnê. 2012; 9:21–32.
7. Ruge IAR, Alvarado JD. Sistema Basado en FPGA Para la Evaluación de Redesneuronalesorientadas al Reconocimiento de Imágenes, Rev Tecnura. 2013; 17(36):87–95. Crossref
8. Lopez Presa JL, Perez Calle E. MMP16 a 16-bit Didactic Micro-Programmed Micro-Processor. In: 2011 3rd International Conference on Computer Research and Development, IEEE, 2011, p. 61–66.
9. Costa RV, Fernandes S, Casilo L, Soares A, Freire D. SICXE: Improving Experience with Didactic Processors. In: 2012 Brazilian Symposium on Computing System Engineering, IEEE, 2012, p. 83–89. Crossref PMid:22266531.
10. Casillo LA, Silva IS. Adapting a Low Complexity Data path to MIPS-1. In: 2012 VIII Southern Conference on Programmable Logic, IEEE, 2012, p. 1–6. Crossref
11. Jansen D, Dusch B. Every Student Makes his Own Microprocessor. In: 10th European Workshop on Microelectronics Education (EWME), IEEE, 2014, p. 97–101. Crossref
12. Guzman F. Natalius 8 bit RISC: Overview. Date accessed: 08/06/2012. Crossref
13. Pepelyashev D. 8-bit Microcontroller with Extended Peripheral Set: Overview. Open Cores. Date accessed: 10/08/2008. Crossref
14. Riedel U. Tiny 8: Overview. Open Cores. Date accessed: 11/02/2007. Crossref
15. Hays K, Jshamlet, Open 8 uRISC: Overview, Open Cores. Date accessed: 20/06/2016. Crossref
16. Godinho D. 1664 Microprocessor: Overview, Open Cores. Date accessed: 26/03/2010. Crossref
17. Tan S. aeMB: Overview. Open Cores. Date accessed: 20/12/2009. Crossref
18. Clayton J. risc16f84. Overview. Open Cores. Date accessed: 28/06/2014. Crossref
19. Usselmann R. Mini-Risc Core: Overview Open Cores. Date accessed: 10/03/2009. Crossref
20. Wei L. Clai RISC - Runs 12 Bit Opcode PIC Family: Overview. Open Cores. Date accessed: 13/02/2009. Crossref
21. Mikej. RISC5x: Overview. Open Cores. Date accessed: 09/09/2011. Crossref
22. Prado DFG. Embedded Microcontrollers and FPGAs Soft-Cores, Electrónica-UNMSM. 2006; 18:3–14.
23. Yue-li H, Jia-lin C, Feng R, Zhi-jian L. Design of a High Performance Microcontroller. In: High Density Micro System Design and Packaging and Component Failure Analysis, 2004 HDP'04 Proceeding of the Sixth IEEE CPMT Conference on. IEEE, 2004, p. 25–33.
24. Ortega VHG, Savedra JCS, Ortega S, Tovar RH. Microprocesadordidáctico de Arquitectura RISC Implementado en un FPGA, e-Gnosis. 2009; 7:1–9.
25. PIC16F84A Data Sheet. Microchip Technology Data Sheets. Date accessed: 29/11/2012. Crossref