Indian Journal of Science and Technology, Vol 10(14), DOI: 10.17485/ijst/2017/v10i14/111328, April 2017
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Minimalist 4-bit Processor Focused on Processors Theory Teaching

Fernando Martinez Santa*, Fredy H. Martinez Sarmiento and Holman Montiel Ariza

Technological Faculty, District University Francisco Jose de Caldas, Bogota D. C., Colombia;
fmartinezs@udistrital.edu.co, fhmartinezs@udistrital.edu.co, hmontiela@udistrital.edu.co

Abstract

Objectives: To design and implement a totally-functional 4-bit didactic processor, in order to be applied as a tool for improving the learning process of students of microprocessors courses, and to be used as a base for future applications. **Methods/Analysis:** This work was reached through the application of a final course project in microprocessor and digital circuit courses, where it was given to the students via web platform, just the basic block diagram (to use Bottom-Up methodology) and they had to complete or design collaboratively the rest of details or blocks which is necessary to work. This processor was implemented using hardware description languages such as: Very fast speed Hardware Description Language (VHDL) and Verilog, on Complex Programmable Logic Device (CPLD) and Field Programmable Gate Array (FPGA). **Findings:** The present one is used as a teaching tool for digital circuits and microprocessor courses in the Technology Faculty of District University (Bogotá, Colombia). This minimalist design of the processor was done to reduce the amount of resultant digital gates, in order to implement inside a small digital programmable device. As a result, it was obtained as a simple processor to start with the basic concepts in the learning process of micro-processors and a little improvement of final exams results due to the involving of students into real problems as collaborative designers. **Novelty/Improvements:** Reduced design is capable to implement on small CPLD or to be implemented sometimes on FPGA to do parallel applications. Additionally, its reduced size helps to easily understand it and implementing by the students.

Keywords: 4-bit Processor, Didactic Processor, Finite State Machine, Harvard, RISC, VHDL, Verilog, Processor Design

1. Introduction

Learning process of specific concepts of computing and microprocessors theory uses to be very difficult for students due to most of the cases, that process is based on a too complex and complete processor[1–3]. The number of different concepts such as addressing modes, data flow, data buses, micro-instructions, architecture and others, use to be confusing, even though if the complexity of the real processor were low. This is the reason of the different tries of creating a minimalist and didactic processor[4–9] in which its architecture makes easier the learning of those basic concepts.

Most of the didactic processors have been implemented through simulation platforms[10]. hardware designs are used in digital programmable devices such as CPLDs and FPGAs using VHDL or Verilog as hardware description language[11–15]. That approach is so interesting due to the design could be modified, simulated and implemented by the students, in order to reach a better comprehension about the processor and all of its fundamental concepts. Additional to processor theory learning, those designs can be used to teach advanced concepts of digital circuits and its applications taking advantage of their small size and functionality.

Doing a processor design, the concepts what the students need to learn about processors theory, is obtained at least a medium complexity processor which anyway is too complex for beginner students[2,3]. In order to make easier the starting into the processor world, it is proposed a design with a huge reduction of the components than a real small processor[14], for this, it is necessary to include

*Author for correspondence

some features and restrict the final functionality of the processor. Additionally, this minimalist approach makes easier the hardware implementation and modification.

At Technological Faculty of District University some thesis about implementations using logical programmable devices have been developed, using mainly FPGAs[16–19], but only a few amount of students can reach the needed knowledge level to write a paper, or even though to approve digital circuits and microprocessor courses. Then, there is a high disapproval percentage of microprocessors course by electrical engineering students; that is why we motivated to do this work. ARMOS research group had a rising interest of designing its own basic processors instead of using a design already done. For this specific work, not all the complete design was given to the students, in order to encourage them to redesign or modify it, under their own criteria. This approach allows involving the students into the problem and thus improves their learning process.

This paper is organized as follows. In Section 2 design description of the didactic processor is shown, Section 3 describes some results obtained. Section 4 shows some considerations about the results and future work. Finally, conclusions are shown in section 5.

2. Design Description

The processor designed is a Reduced Instruction Set Computer (RISC) and accumulator-based one with a basic Harvard architecture, all its design was thinking of being a minimalist processor and therefore reducing the amount of resultant digital gates needed to implement it. In order to be minimized its data bus was reduced to just 4 bits and its storage capability was reduced to the minimum. This consists of a logical arithmetic unit Arithmetic Logic unit (*ALU*), a 4-bit word selector o multiplexer (*MUX*), a Program Counter (*PC*), an Instruction Register (*IR*), an Accumulator register (*ACC*), a data memory, a program memory and a control unit. Processor includes 4 input ports and 4 output ports of 4 bits each one. The schematic design is shown in Figure 1. It is described that each component of designed processor. The *ALU* has a Carry output (*C*) for the carry of each operation and a Zero output (*Z*) to indicate when an operation gives a result with all bit on zero. Program counter *PC* is an 8-bit register with a parallel load input of 6 bits and 3 control bits. Instruction register *IR* is another 8-bit register with input and output of 8 bits.

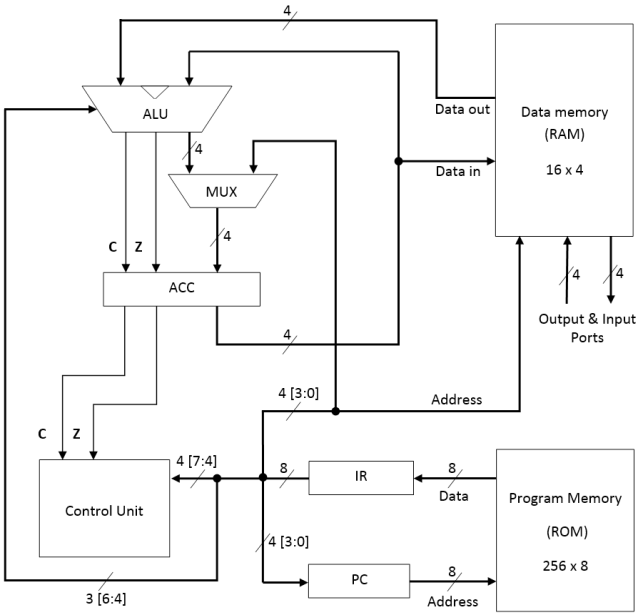

Figure 1. Processor block diagram.

Accumulator *ACC* is a 6-bit register with a 4-bit input from *ALU* and other two input bits for *Z* and *C* signals, also it is possible to reset *Z* and *C* values by a control bit. Data memory (RAM) is a bank of 16 register of 4 bits each one, it has a 4-bit address bus and two different 4-bit data buses for input and output. Memory addresses from 0 to 7 are internal registers, addresses from 8 to 11 work as output ports and addresses from 12 to 15 works as input ports. Program memory is a Read Only Memory (ROM) of 256 positions of 8 bits each one and it stores the corresponding values of instructions to be executed by the processor. Those instructions are programmed directly on the description of program memory language using VHDL and Verilog. This memory is totally asynchronous and has two 8-bit buses address bus and output data bus. Unit control is a Finite State Machine (FSM), which does each necessary step (micro-instruction) to execute the programmed instructions; this one has the instructions decoder inside. This block receives *Z* and *C* signals and a 4-bit from *IR* Most Significant Bits (MSB), in order to get the instruction operation code. Control unit has output signal to control: loading of *ACC*, increment and addition of *PC*, writing and reading of memories and selection of *MUX*.

2.1 Instruction Set

The processor has an instruction set of just 14simple instructions. This is the first feature that makes the processor minimalist and didactic, due to this amount of

Table 1. Instruction set of processor

Instruction	Function	Description
ADD Address	ACC <= DataMem[Address] + ACC	Addition
SUB Address	ACC <= DataMem[Address] – ACC	Subtraction
AND Address	ACC <= DataMem[Address] and ACC	And
OR Address	ACC <= DataMem[Address] or ACC	Or
XOR Address	ACC <= DataMem[Address] xor ACC	Exclusive or
NOT	ACC <= not ACC	1´s Complement
LOAD Address	ACC <= DataMem[Address]	ACC load
STORE Address	DataMem[Address] <= ACC	RAM storage
LOADK Constant	ACC <= Constant	ACC constant load
GOUP Unsigned	PC <= PC – Unsigned	Go up (Jump backward)
GODW Unsigned	PC <= PC + Unsigned	Go down (Jump forward)
NOP	ACC <= ACC	Not operation
JFIC Unsigned	if(carry = '1')then PC <= PC + *Unsigned*	Jump forward if Carry
JFIZ Unsigned	if(zero = '1')then PC <= PC + *Unsigned*	Jump forward if Zero

instructions is easily memorized. The complete instruction is shown in Table 1. These instructions can be classified on 2 arithmetical, 4 logical, 3 transferring memory ones, 2 unconditional jumps, 2 conditional jumps and a not operation instruction. Those instructions use direct, immediate and inherent addressing modes.

2.2 Instruction Coding

The processor uses an 8-bit instruction coding that is composed of 4-bit operation code and a 4-bit operand. The Most Significant Bits (MSB) is the opcode and the Least Significant Bits (LSB) is the operand. Table 2 shows the instruction coding.

For the *ALU* instructions such as: signed Addition (ADD), Subtraction (SUB), bitwise and (AND), bitwise Or (OR), bitwise Exclusive Or (XOR) and Load register (LOAD). There is a specific opcode for each one and 4 bits A3, A2, A1 and A0 that relate to a memory address to access. Negation instruction (NOT) does not require operand, thus the last four bits are ignored. Load constant instruction (LOADK) has an operand K3, K2, K1 and K0 bits that refer to 4-bit data to be load to the accumulator register *ACC*. The conditional and unconditional jump instructions have as operand J3, J2, J1 and J0 that refer to the value of the jump to do (the value to add or subtract to *PC*), that allows to do jumps of 16 memory positions.

Table 2. Instruction coding for processor

Instruction	Opcode	Operand
ADD Address	0 0 0 0	A3 A2 A1 A0
SUB Address	0 0 0 1	A3 A2 A1 A0
AND Address	0 0 1 0	A3 A2 A1 A0
OR Address	0 0 1 1	A3 A2 A1 A0
XOR Address	0 1 0 0	A3 A2 A1 A0
NOT xxxx	0 1 0 1	X X X X
LOAD Address	0 1 1 0	A3 A2 A1 A0
NOP xxxx	0 1 1 1	X X X X
STORE Address	1 0 0 0	A3 A2 A1 A0
LOADK Constant	1 0 0 1	K3 K2 K1 K0
GOUP Unsigned	1 0 1 0	U3 U2 U1 U0
GODW Unsigned	1 0 1 1	U3 U2 U1 U0
JFIC Unsigned	1 1 0 0	U3 U2 U1 U0
JFIZ Unsigned	1 1 0 1	U3 U2 U1 U0

2.3 Arithmetic Logic Unit ALU

ALU does arithmetic operations such as signed addition (ADD) and subtraction (SUB), and logic operations such as NOT, AND, OR and exclusive-OR (XOR). This unit has couple of 4-bit data inputs (A and B named from right to left) and a data output, also its selection input has 3 bits. Table 3 shows all the possible operation of *ALU*.

Table 3. Operations of the 4-bit ALU

Selection *IR*[6:4	Output
0 0 0	A + B
0 0 1	A - B
0 1 0	A and B
0 1 1	A or B
1 0 0	A xor B
1 0 1	not B
1 1 0	A
1 1 1	B

3. Results

Usually the instruction decoder is the processor components that use the amount of logic gates. This design attacks the problem in two different ways, reducing the amount of instructions and matches the opcodes with the *ALU* operations. The *ALU* operations match with the bits from 4 to 6 position of opcode of arithmetic and logic instructions (LOAD and NOP are *ALU* operations). Therefore, it is not necessary to do any de-codification for arithmetic and logic instructions, for the rest it is implemented direct de-codification through the FSM of the control unit, merging all *ALU* instructions as an only one state, as shown in Figure 2.

It was reached a finite state machine just with 10 state including reset states, in order to minimize the total size of the obtained processor and of course it is the enough simple to be understood very fast.

It was obtained an executing rate of just 4 clock cycles for all the instructions; this is the most important features for a RISC processor. This rate was done inside the state "Decode" the comparison of processor flags (Carry and Zero). Different hardware descriptions using VHDL and Verilog languages were designed. First one was done using the Top-Down methodology, it means that each functional block was described separately and merged at the end. This approach is useful for students to understand much better the internal working of processor. Additionally, using both languages it describes the complete processor as an only one block, in order to be easier, the use of this, on a specific application, once the students have understood its basic behavior.

The description of data memory is so particular, because it includes the input and output ports itself, this was done, in order to manage those ports as a simple memory positions and avoid creating specific instructions for them.

For the student of electrical engineering at Technological Faculty of District University was done compared the results of final exam of micro-processors course, before and after first term of 2011 when it was started using of the minimalist processor as a didactic tool. The average scores improve after that in approximately 6% until second term of 2014, according with faculty data.

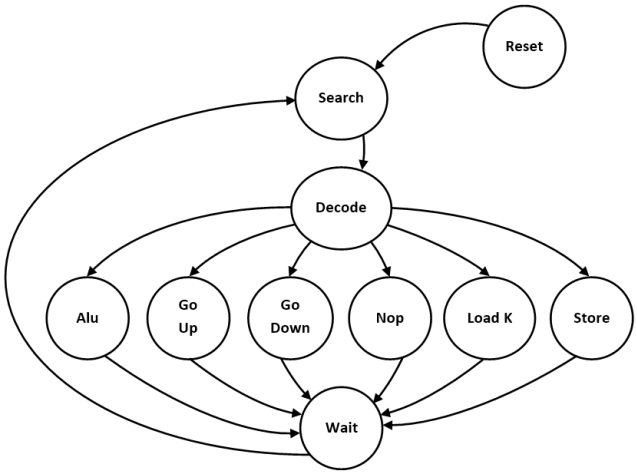

Figure 2. Finite state machine of Control unit.

Approximately 92% of asked students, think of this didactic processor is a good tool to teach the micro-processors concepts and additionally it helps to approve the exams not only it is simple and easy to understand but the collaborative way to design it is good. This value was obtained through polls applied in each term, to all the students of the course from 2012.

Starting from a basic block diagram, each semester the students reached similar but different designs that were enriching the current one. The obtained design is the result of continuous changes done by the teachers of ARMOS researching group after apply the processor as final project and some design ideas given by the students through semesters.

4. Limitations and Future Research

This design has a lot of limitations of reducing its size. First, it is an accumulator-based 4-bit processor which

limits its compute capacity itself. The processor has not interrupt sources; it can be sequentially programmed. There is not an instruction for subroutine calling. The amount of program and data memory is too limited to be applied to most of real applications. This processor has not any type of indirect addressing which is not necessary due to the low amount of registers of Random Access Memory (RAM). The instruction set does not include arithmetic instructions with carry, division and multiplication, and rotation and shift instructions. All jumps are relative and limited to 16 memory positions. There is no way to do constant vectors or tables in ROM memory.

The effectiveness of the processor as a didactic tool is too difficult to determine accurately, this design is being improved through the addition of more instructions such as addition and subtraction with carry and right-left rotation. Also, the conditional and unconditional jumps will be enlarged taking advantage of the unused bits of instructions coding.

ARMOS researching group is designed with an 8-bit processor in this work. The approach in this case is not only to reach a small and optimized design but to have the enough functionality to be used for real applications.

5. Conclusion

It is possible to obtain a minimalist but functional processor design, using standard hardware description languages and using web collaborative platforms, in order to improve the learning process of processors theory. Due to the reached design is very simple and totally modifiable; the students obtain a better comprehension of the architecture and the basics concepts making small or big changes to the processor. We expect that the knowledge acquired by the students through this direct and collaborative interaction with the processor architecture makes easier the learning process of any new processor architecture or even encourages them by studying about processors and computing theory.

6. Acknowledgments

This work was supported by the District University Francisco José de Caldas specifically through CIDC. The views expressed in this paper are not necessarily endorsed by District University. The authors thank the research group ARMOS for the evaluation carried out on designs and tests.

7. References

1. Golze U. VLSI chip design with the hardware description language VERILOG: An introduction based on a large RISC processor design. Springer Science & Business Media; 2013. p. 360.
2. Fischer V, Drutarovský M. Scalable RSA processor in reconfigurable hardware - a SoC building block. Proceedings of XVI Conference on Design of Circuits and Integrated Systems - DCIS; 2001.
3. Casillo LA, Silva IS. Adapting a low complexity datapath to MIPS-1. VIII Southern Conference on Programmable Logic, IEEE; 2012 Mar. p. 1–6.
4. Zavala AH, Nieto OC, Ruelas JAH, Dominguez ARC. Design of a general purpose 8-bit RISC processor for computer architecture learning. Computacion y Sistemas. 2015 Jun; 19(2):1–15.
5. Presa JLL, Calle EP. MMP16 a 16-bit didactic microprogrammed micro-processor. 2011 3rd International Conference on Computer Research and Development; 2011. p. 61–5.
6. Carracedo J, Núñez MA, Pastor E. MICROSIMPLEX: A didactic microcomputer specially designed for computer education. Microprocess Micro-programming. 1983 Oct; 12(3–4):217–21.
7. Ferlin EP, Junior VP. Microprocessors: from theory to practice, a didactic experience. 34th Annual Frontiers in Education; 2004 Oct. p. 971–4.
8. Costa RV, Fernandes S, Casilo L, Soares A, Freire D. SICXE: Improving experience with didactic processors. 2012 Brazilian Symposium on Computing System Engineering; 2012 Nov. p. 83–6.
9. Martins CAPS, Correa JBT, Goes LFW, Ramos LES, Medeiros TH. A new learning method of microprocessor architecture. 32nd Annual Frontiers in Education; 2002 Nov. p. 16–21.
10. de Freitas HC, Martins CAPS. Didactic architectures and simulator for network processor learning. WCAE '03 Proceedings of the 2003 workshop on Computer architecture education: Held in conjunction with the 30th International Symposium on Computer Architecture; 2003. p. 14.
11. Morales-Velazquez L, Osornio-Rios RA, Romero-Troncoso RJ. FPGA embedded single-cycle 16-bit microprocessor and tools. 2012 International Conference on Reconfigurable Computing and FPGAs; 2012. p. 1–6.
12. Park H, Ko Y-W, So J, Lee J-G. Synthesizable manycore processor designs with FPGA in teaching computer architecture. International Journal of Control Automation. 2013 Oct; 6(5):429–38.
13. Shi Q, Xiang L, Chen T, Hu W. FPGA-based embedded system education. 2009 First International Workshop on

Education Technology and Computer Science; 2009 Mar. p. 123–7.
14. Pereira MC, Viera PV, Raabe ALA, Zeferino CA. A basic processor for teaching digital circuits and systems design with FPGA. 2012 VIII Southern Conference on Programmable Logic; 2012 Mar. p. 1–6.
15. Jansen D, Dusch B. Every student makes his own microprocessor. 10th European Workshop on Microelectronics Education (EWME); 2014. p. 97–101.
16. Jacinto E, Plazas DCP, Restrepo MFR. Voltimetro true-rms sobre fpga basado en algoritmo cordic. Tecnura. 2015; 19:129–36.
17. Garzón VAB, Bareño JJN, Jacinto E. Diseño e implementación de un codec digital de audio con FPGA, en formato PCM, de 2 canales con interfaz para usuario. Tecnura Tecnol y Cult Afirmando el Conoc. Universidad Distrital Francisco José de Caldas. 2010; 14(26):56–68.
18. Giral D, Romero R, Martinez S. F. Procesamiento paralelo en FPGA para convolución de imágenes usando Matlab. Tecnura. 2015 Jan; 19(43):119.
19. Riano J, Ladino C, Martiez F. Implementación de la transformada FFT sobre una FPGA orientada a su aplicación en convertidores electrónicos de potencia. Tekhne. 2012; 9:21–32.