Hardware description of a simplified 4-bit softcore processor with bcd capabilities

Fernando Martínez Santa, Edwar Jacinto, Holman Montiel
ARMOS Research Group, Technology Faculty, Universidad Distrital, Colombia

Article Info

Article history:

Received Apr 8, 2019
Revised Oct 25, 2019
Accepted Nov 6, 2019

Keywords:

Didactic processor
FPGA
Hardware description
Softcore
Verilog

ABSTRACT

The objective of the work reported in this paper is to improve a 4-bit softcore processor previously designed in Verilog language, keeping its compact size. This processor was thought to be used as academic and didactic tool for teaching as computers architecture subject as digital circuits subject in the technology faculty of the Universidad Distrital. The new features include arithmetic instruction with input carry, BCD operations enabling, rotating instructions, implementation of input and output register banks, increase of the number of general purpose registers of the data memory, and the reduction of the execution clock cycles per instruction. Additionally, the assembler software was enabled to support macro-instructions to make easy the comprehension of some composed functions. As result, a very compact softcore processor was obtained, by means of a Verilog description done in a single file. This implementation occupies only the 2% of the medium-size FPGA used for the application, reaching a maximum possible working clock frequency of 929 Mhz.

Corresponding Author:

Fernando Martínez Santa,
Facultad Tecnológica,
Universidad Distrital,
Cl. 68d Bis ASur #49F - 70, Bogotá, Colombia.
Email: fmartinezs@udistrital.edu.co

1. INTRODUCTION

The digital circuits and processor theory learning are one of the most important subjects for the electric and electronic engineering students, but also they are the ones that presents more amount of students who do not approve, that is why a lot of different didactic strategies have been developed [1], mainly in the practical area. Due to the invention of hardware description languages like VHDL and Verilog and programmable logic devices like CPLDs, FPGAs and SOCs, since a pair of decades the learning focused on the design has been highly facilitated [2]. Different processor designs have been developed mainly with academic purposes, for teaching digital circuits [3-6], computer architecture [7-10] and embedded systems [11-14]. Most of these designs has been described using VHDL language [15-17], Verilog language [6, 18], or even both. Those designs can be classified taken into account the number of bits of their main data bus, basically there are 16-bit processors [8, 9, 13], 8-bit processors [17, 19, 20] and 4-bit processor [18]. These referenced implementations are as diverse as the universities and institutes where were designed, some have Von Neumann architecture, some other Harvard architecture, some are consider RISC, some other CISC, etc.

In the Universidad Distrital from Bogotá, Colombia, specifically at technology faculty, some different designs about this topic have been developed. The first approach, was a very simple 4-bit processor [18] described in VHDL and Verilog languages, which was born from the need of creating a final project of the Digital Circuits course for electrical engineering students. At that time, the electronics laboratory (belongs to electric engineering) just had small Xilinx® CPLDs, that was the reason because a highly reduced design was developed. After that, along with the ARMOS research group, other two designs were done, both were 8-bit softcore processors. The first one was inspired on a Microchip's PIC® microcontroller [20], which has as main objective, to take advantage of the advantage of the compilation and debugging tools existing for those microcontrollers. The second one, was a totally new design [19] that used a pair of twins accumulators and some other non-explored features inspired in part by PicoBlaze microprocessor and its multiple application [21-25].

The main goal of the first 4-bit processor designed [18] is to offer to the students another practical tool to strengthen the knowledge acquired in the classroom. For the students of the digital circuits subject, the processor is focused as final project of the course, where the professor gives the block diagram and the description of some of the functional blocks preformed in Verilog language, and they have to describe the rest of the blocks and integrate all of them in order to do the design totally functional, implementing a simple application. On the other hand, for the students of the computers architecture subject, the 4-bit processor is focused as the tool to learn in a global way, how a computer works, so it is used only in the first part of the course. In order to keep the learning process of the students at the courses of digital circuits and computers architecture (at Universidad Distrital, Bogotá, Colombia), as simple as possible, the first 4-bit processor designed has been preferred instead of the two 8-bit processor after designed. Then, since that time, some changes and improvements have been developed to the original description, but always trying to keep its simpleness. This paper shows the changes and improvements implemented and the respective comparison between the original processor and its second version.

2. RESEARCH METHOD

The previous processor design had only 14 instructions [18], including arithmetic, logic, register transfer and branch ones. Instead of being totally functional, after apply it in several therms, we realized that processor lacks some basic functionalities such as: carry arithmetic instructions, an easy to understand jump or branch instructions, decimal adjust for BCD operations, rolling or shifting operations and finally a better way to get in and get out data to the outside. In this article, the improvements implemented on those features are shown, first the detection of those weak points and after the proposed improvements and their implementations. All of the proposed changes were thought to avoid increase the amount of the program and data memories.

2.1. Instructions without direct-carry functionality

The original design has only 2 arithmetic instructions: ADD and SUB, both without input carry, which make difficult to chain operations for instance to do 8-bit additions. An example of an 8-bit addition using the original processor, is shown as follows, where R1:R0 compose the first 8-bit operand, R3:R2 the second one, and R5:R4 the destiny.

```
LOAD    R3      ;addition of the most
ADD     R1      ;significant nibbles
STORE   R5

LOAD    R0      ;addition of the least
ADD     R2      ;significant nibbles
STORE   R4

JFIC    2       ;carry test
GODW    4

LOADK   1       ;increment the most
ADD     R5      ;significant nibble of
STORE   R5      ;the destiny, if carry

NOP
```

As shown in the previous assembly code, an 8-bit addition can be performed with at least 11 instructions. If an addition with carry operation existed, the same operation would be performed since 6 or 7 instructions.

2.2. Input an output operations

The existing design has not direct input or output registers, instead of that it has 4 input and 4 output registers mapped directly into the data memory in fixed position. This has the advantage each instruction can affect directly the ports, but at the same time make difficult de description of the data memory as a standard RAM. This makes the comprehension too hard specially to the students of the digital circuits subject. If separated I/O banks existed, the manage of inputs and outputs would be easier and all the 16 memory positions of the RAM could be used as general purpose registers.

2.3. Rotating or shifting operations

There are no rolling or shifting instruction in the original design of the 4-bit processor. That kind of instructions are very important for manage of independent bits of a register or in communication applications.

2.4. BCD operations functionality

Being the design a 4-bit processor, it highlights for its lack of BCD operations. Most of the functional 4-bit processors are focused on performing BCD operations, because its number of bits make easier to do this kind of processing, like the processors used in old calculators. It is very important to give to the design some BCD options in order to increase its functionality.

2.5. More understandable jump instructions

Jump or branch instruction in the original design produced some confusing to the students, because there are to different instruction to jump. GOUP "go up" and GODW "go down", depending on the direction of the jump. If there was only one branch instruction, the written code would be clear.

3. RESULTS AND ANALYSIS

The new proposed processor design is an accumulator machine with a Harvard-RISC architecture. This is a non-pipelining processor and it has not interruptions. The processor consists of: a RAM data memory of 16x4 bits, a ROM program memory of 256x8 bits, an 8-bit instruction register IR, an 8-bit program counter PC, a 4-bit accumulator ACC, 1-bit flags of zero Z, carry C and digit carry DC, two 4-bit 4-input selectors, a 4-bit ALU and finally a bank of 4 input registers and another of 4 input registers, as shown in the block diagram of Figure 1. The main changes looking at the block diagram is the inclusion of the digit carry DC flag and the input and output register banks. The digit carry flag was implemented to enable BCD operations, due to this one is activated only when the result of the ACC after any arithmetic instruction is grater than 10. On the other hand, the input and output ports were implemented to separate them from the data memory.

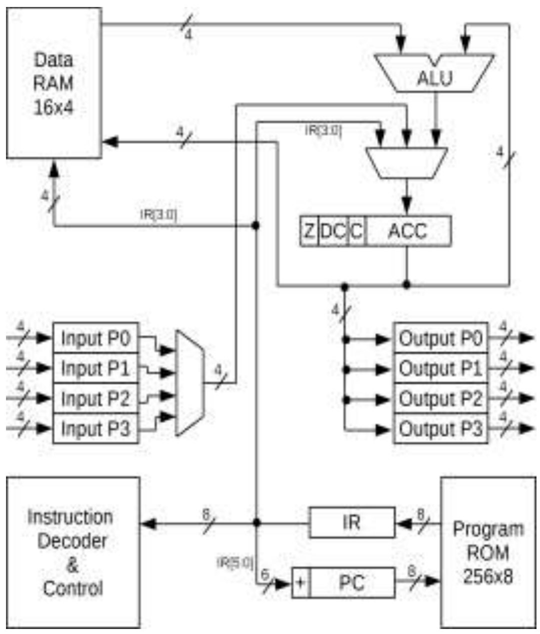

Figure 1. Block diagram of the designed softcore processor

3.1. Instruction set

The proposed design has a total of 20 instructions: 3 arithmetic, 4 logic, 2 rotation, 3 of flags management, 5 register transferring, 1 branch and 2 testing instructions, increasing in 6 the previous instruction set. The Table 1 shows each instruction and its description.

Table 1. Instruction set summary

Instruction	Description
ADDDC	addition with carry or digit carry
SUBDC	subtraction with carry or digit carry
AND	and operation
OR	or operation
XOR	exclusive or operation
NOT	not operation to the accumulator
DA	decimal adjust to the accumulator
ROL	rotate left the accumulator
ROR	rotate right the accumulator
SETC	set carry flag
CLRC	clear carry and digit carry flags
CLRDC	clear digit carry flag only
IN	load the accumulator with an input port
LOAD	load the accumulator with a memory position
LOADK	load the accumulator with a constant
JUMP	relative jump
JFIDC	jump forward if carry or digit carry
JFIZ	jump forward if zero
STORE	store accumulator in a memory position
OUT	load an output port with the accumulator

The main changes on the instruction set were: the instruction of not operation NOP was deleted, the arithmetic instructions ADD and SUB were changed to support input carry, a decimal adjust DA instruction was crated to make easy BCD operations, rotate left and right instructions ROL and ROR were created, flag management instructions SETC, CLRC and CLRDC were created, I/O instructions IN and OUT were created, branch instructions GOUP and GODW were replaced by JUMP, and finally the test instruction JFIC were replaced by JFIDC which also test the digit carry. All the instructions of the processor have only 1 of the 2 available addressing modes, direct addressing or inherent addressing.

3.2. Instruction codification

The main idea was to increase the number of instructions and thus its functionality, but at the same time keep its reduced space occupied and its simplicity. In order to not increase the size and number of bits of the program memory, the previous codification scheme [18] of 4-bit instruction and 4-bit operand was reformulated, doing that the opcode has a variable number of bits. In the new codification scheme the opcode can variate from 2 to the complete 8 bits of the instruction, the JUMP instruction is codified only with 2 bits, the I/O instructions are codified with 6 bits, NOT, DA, the rotating instructions and the flag management instructions are codified with 8 bits, and the rest of instruction are codified using 4 bits, as shown in Table 2. The instructions are classified in two groups, the ones that update the accumulator and the ones that do not, those groups are differentiated by means of the first bit (Table 2).

Table 2. Instruction codification in 8 bits

Instruction		Opcode	
ADDDC	*Address*	0 0 0 0	A3 A2 A1 A0
SUBDC	*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		0 1 0 1 0 0 0 0	
DA		0 1 0 1 0 0 0 1	
ROL		0 1 0 1 0 0 1 0	
ROR		0 1 0 1 0 0 1 1	
SETC		0 1 0 1 0 1 0 0	
CLRC		0 1 0 1 0 1 0 1	
CLRDC		0 1 0 1 0 1 1 0	
IN	*Port*	0 1 0 1 1 1	P1 P0
LOAD	*Address*	0 1 1 0	A3 A2 A1 A0
LOADK	*Constant*	0 1 1 1	K3 K2 K1 K0
JUMP	*Signed*	1 0	S5 S4 S3 S2 S1 S0
JFIDC	*Unsigned*	1 1 0 0	U3 U2 U1 U0
JFIZ	*Unsigned*	1 1 0 1	U3 U2 U1 U0
STORE	*Address*	1 1 1 0	A3 A2 A1 A0
OUT	*Port*	1 1 1 1 x x	P1 P0

It was taken advantage of that 7 of the 20 instructions use inherent addressing, this means those instruction do not need any operand, then they can use all the 8 bits of the instruction. The binary code "01010111" was not used, so it is possible to implement for instance the NOP instruction using this code. On the other hand, the out really only need 4 bits for the codification, this means than the other 4 bits can be used as port address, increasing them from 4 to a total of 16 output registers.

3.3. Instructions with input carry

In order to give to the *ADD* and *SUB* instructions the possibility to work with input carry, the instructions *ADDDC* and *SUBDC* were created. Due to the limited number of codification bits (only 8), it was not possible to include the respective instructions without input carry. An example of an addition of two 8-bit numbers, is shown as follows.

```
LOAD     R0       ;addition of the least
CLRC              ;significant nibbles
ADDDC    R2
STORE    R4

LOAD     R1       ;addition of the most
ADDDC    R3       ;significant nibbles
STORE    R5
```

A reduction from 11 instructions to only 7 was obtained, in comparation with the original design. This same architecture can easily perform BCD addition only adding the *DA* instruction. An example of an addition of 2 BCD numbers of 2 digits each one (right), is shown as follows:

```
         LOAD     R0       ;addition of the least
         CLRC              ;significant BCD digits
         ADDDC    R2       ;with decimal adjust
         DA
         STORE    R4

         LOAD     R1       ;addition of the most
         ADDDC    R3       ;significant BCD digits
STORE    R5
```

3.4. Control unit

For the control unit, a FSM (Finite State Machine) was implemented following the non-pipelining *Fetch-Decode-Execute* scheme. The state diagram is shown in Figure 2. The *Wait* state presented in the previous design of the state machine [18] was used to increment de PC register before the next *Search* state. For this design, this state was eliminated, and the increment of the PC is directly done in each execution state. This last, reduced from 4 to only 3 states (thus only 3 clock cycles) the execution of each instruction.

3.5. Macro-instructions

As shown in the code examples of the previous sub-section, the simple additions needs and additional *CLRC* instruction, but this can be confusing for the students at the beginning, that is why some macro-instructions were implemented under the assembler software, in order to write instruction combinations as only one. The Table 3 shows the implemented macro-instructions and their translation to assembly language.

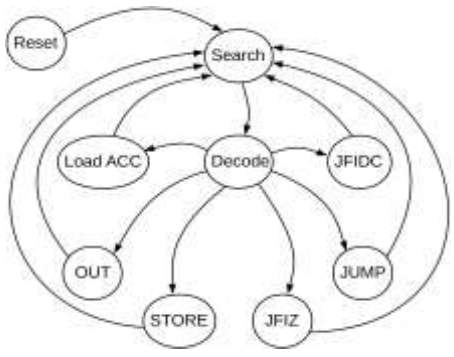

Figure 2. State diagram of the control unit

Table 3. Macro-instructions and their meaning

Macro Instruction	Assembly	Description
ADD Address	CLRC ADDDC *Address*	addition
ADDC Address	CLRDC ADDDC *Address*	addition with carry
SUB Address	CLRC SUBDC *Address*	subtraction
SUBC Address	CLRDC SUBDC *Address*	subtraction with carry (borrow)
SHL	CLRC ROL	shift left
SHR	CLRC ROR	shift right
NOP	NOT NOT	not operation

3.6. Processor implementation size

The design was implemented and tested on a FPGA Spartan 3AN xc3s700an-4fgg484 by Xilinx®. This new design was compared with the previous implementation on the same device, obtaining that it uses less flip-flops (from 98 to 81) but more LUTs (Look Up Tables) and slices (basic units that group flip-flops and combinational logic). Specifically the new design uses more than the double of LUTs (from 114 to 238). Regarding the maximum possible frequency the difference in minimal (from 938Mhz to 929Mhz). The Figure 3 summarizes all of these results.

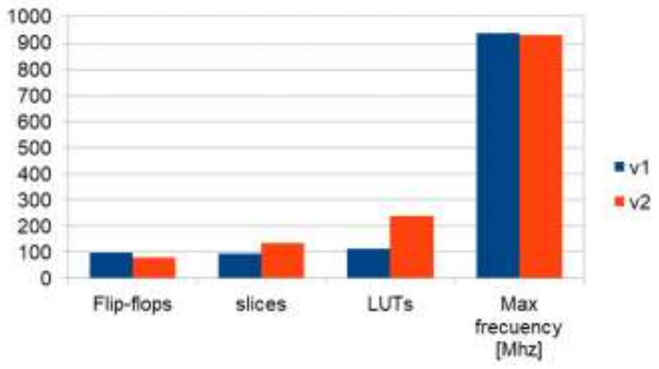

Figure 3. Results comparison of the hardware implementation

4. CONCLUSION

The previous design was improved in some many ways, firstly the number of instruction was increased from 14 to 20, and also 7 macro-instructions were implemented. Some instruction were improved or replaced and other ones were included, but the program memory kept its original size. Secondly, a 4-register input bank and a 4-register output bank were implemented, making all the registers of the data memory are general purpose registers. Also, operations with input carry and the possibility to implement BCD operations were enabled by means of implementing addition and subtraction instructions with input carry and input digit carry, as well as the decimal adjust instruction. On the other hand, the modification implemented on the FSM of the control unit, reduced the number of execution clock cycles from 4 to 3. In addition, the maximum frequency was only reduced to the 99.1% of the frequency reached with the original processor. Finally, the total size occupied by the processor increased, but the functionality was improved too, also making more standard the Verilog descriptions and anyway keeping the design as reduced as possible, since the design only occupied the 2% of the total resources of the device selected for the implementation.

ACKNOWLEDGEMENTS

This work was supported by the Universidad Distrital Francisco José de Caldas, specifically by the Technological Faculty. The views expressed in this paper are not necessarily endorsed by District University. The authors thank all the students and researchers of the research group ARMOS for their support in the development of this work.

REFERENCES

[1] M. Benítez Mendivelso, "La valoración de los applets, por parte del docente, posibilita la utilización de las TIC en el aula", *#ashtag*, n.º 4&5, pp. 106-115, nov. 2014.
[2] J. Riaño, C. Ladino, and F. Martínez, "Implementación de la transformada FFT sobre una FPGA orientada a su aplicación en convertidores electrónicos de potencia," *Tekhnê*, vol. 9, pp. 21–32, 2012.
[3] M. C. Pereira, P. V. Viera, A. L. A. Raabe, and C. A. Zeferino, "A basic processor for teaching digital circuits and systems design with FPGA," in *2012 VIII Southern Conference on Programmable Logic*, pp. 1–6, 2012.
[4] C. D. Cunțan, I. Baciu, and M. Osaci, "Studies on the Necessity to Integrate the FPGA (Field Programmable Gate Array) Circuits in the Digital Electronics Lab Didactic Activity," *Int. J. Mod. Educ. Comput. Sci.*, vol. 7, no. 6, 2015.
[5] D. Jansen and B. Dusch, "Every student makes his own microprocessor," in 10th *European Workshop on Microelectronics Education* (*EWME*), pp. 97–101, 2014
[6] U. Golze, VLSI Chip Design with the Hardware Description Language VERILOG: "An Introduction Based on a Large RISC Processor Design", vol. 2013. *Springer Science & Business Media*, 2013.
[7] L. Ribas-Xirgo, "Yet Another Simple Processor (YASP) for Introductory Courses on Computer Architecture," *IEEE Trans. Ind. Electron.*, vol. 57, no. 10, pp. 3317–3323, Oct. 2010.
[8] V. Angelov and V. Lindenstruth, "The educational processor Sweet-16," *in 2009 International Conference on Field Programmable Logic and Applications*, pp. 555–559, 2009.
[9] J. L. Lopez Presa and E. Perez Calle, "MMP16 a 16-bit Didactic Micro-Programmed Micro-Processor," *in 2011 3rd International Conference on Computer Research and Development*, vol. 1, pp. 61–65, 2011.
[10] R. V. Costa, S. Fernandes, L. Casilo, A. Soares, and D. Freire, "SICXE: Improving Experience with Didactic Processors," *in 2012 Brazilian Symposium on Computing System Engineering*, pp. 83–86, 2012.
[11] Q. Shi, L. Xiang, T. Chen, and W. Hu, "FPGA-Based Embedded System Education," in 2009 *First International Workshop on Education Technology and Computer Science*, vol. 1, pp. 123–127, 2009.
[12] J. Ruiz, D. Guerrero, I. Gomez, and J. Viejo, "Implementation of a hardware and software framework for a simple academic processor," *in 2012 Technologies Applied to Electronics Teaching (TAEE)*, pp. 48–53, 2012.
[13] L. Morales-Velazquez, R. A. Osornio-Rios, and R. J. Romero-Troncoso, "FPGA embedded single-cycle 16-bit microprocessor and tools," *in 2012 International Conference on Reconfigurable Computing and FPGAs*, pp. 1–6. 2012
[14] S. N. Soares and F. R. Wagner, "T&D-Bench—Innovative Combined Support for Education and Research in Computer Architecture and Embedded Systems," *IEEE Trans. Educ.*, vol. 54, no. 4, pp. 675–682, Nov. 2011.
[15] R. Sanz Fernández, "Síntesis de un procesador en VHDL para su posterior volcado en una FPGA," 2017.
[16] H. Park, Y.-W. Ko, J. So, and J.-G. Lee, "Synthesizable Manycore Processor Designs with FPGA in Teaching Computer Architecture," *Int. J. Control Autom.*, vol. 6, no. 5, pp. 429–438, Oct. 2013.
[17] A. Hernandez Zavala, O. Camacho Nieto, J. A. Huerta Ruelas, C. Domínguez, and R. Arodí, "Design of a general purpose 8-bit RISC processor for computer architecture learning," *Comput. y Sist.*, vol. 19, no. 2, pp. 371–385, 2015,
[18] F. M. Santa, F. H. M. Sarmiento, and H. M. Ariza, "Minimalist 4-bit Processor Focused on Processors Theory Teaching," *Indian J. Sci. Technol.*, vol. 10, no. 14, 2017.
[19] W. Sáenz Rodríguez, F. Rivera Sánchez, and F. Martínez Santa, "8-bit softcore microprocessor with dual accumulator designed to be used in FPGA," *Tecnura*, vol. 22, no. 56, pp. 40–50, 2018.
[20] J. E. R. Prieto, E. J. Gómez, and F. M. Santa, "Implementation of an 8-Bit Softcore Microcontroller on Xilinx Spartan FPGA Family," *Indian J. Sci. Technol.*, vol. 10, no. 22, 2017.
[21] Xilinx, "PicoBlaze 8-bit Embedded Microcontroller User Guide," IP documentation, [_Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf, 2011.
[22] Ivanov, Vladimir N. "Using a PicoBlaze Processor to Traffic Light Control". *Cybernetics and Information Technologies*, vol. 15, no 5, p. 131-139, 2015.
[23] Bencheva, N.; Kostadinov, N.; Ruseva, Y. "On Teaching Hardware/Software Co-design using FPGA". *Elektronika ir Elektrotechnika*, vol. 102, no 6, p. 91-94, 2015.
[24] Lung, Claudiu; Sabou, Sebastian; Buchman, Attila. "Wireless sensor networks as part of emergency situations management system". *En 2016 IEEE 22nd International Symposium for Design and Technology in Electronic Packaging (SIITME). IEEE*. p. 240-243. 2016.
[25] Afzal, Shiraz, et al. "Single Chip Embedded System Solution: Efficient Resource Utilization by Interfacing LCD through Softcore Processor in Xilinx FPGA". *International Journal of Information Engineering and Electronic Business*, vol. 7, no 6, p. 23, 2015.