Research Journal of Applied Sciences, Engineering and Technology 15(1): 31-39, 2018
DOI:10.19026/rjaset.15.5289
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: July 28, 2017 Accepted: September 11, 2017 Published: January 15, 2018

Research Article

Design and Simulation of an 8-bit Educational Microprocessor Architecture using Very High Speed Integrated Circuit Hardware Description Language

Jihene Mallek
Electronic and Communications Group, LETI-Laboratory, National School of Engineers of Sfax, University of Sfax, Tunisia

Abstract: The aim of this study is the design and simulation of a simple 8-bit microprocessor architecture dedicated for educational purposes, using Very High Speed Integrated Circuit Hardware Description Language (VHDL). In fact, the existing commercial microprocessors are provided as black box units, with which users are unable to monitor internal signals and operation process, neither can they modify the original structure. So they are unsuitable for education. In order to solve this problem, the present paper involves the design method of a simple microprocessor, from analysis to programming, passing by the choice and the wiring of the microprocessor components. Moreover, the processor contains a number of the basic modules. These modules are a sequencer, an Arithmetic Logic Unit (ALU) and registers, which are studied and designed. All of these units or modules are assembled together and communicate through an 8-bit data bus. The proposed educational microprocessor worked successfully since it was verified using VHDL simulation results.

Keywords: Arithmetic logic unit, 8-bit data bus, educational microprocessor, hardware level, sequencer, VHDL

INTRODUCTION

Any hardware design can be described in terms of its operations at different levels of abstraction, from the system through to logic gates. At each level of this hierarchy the overall inputs and outputs remain the same, but the functionality of distinct sections, becomes clearer with the help of detailed schematics (Gupta *et al*., 2014). The motivation is to link digital electronics with microprocessor architecture and, therefore, show the students how a microprocessor can be built using the simple components they already know (Presa and Calle, 2011). In addition to that, it is not possible to observe the events taking place in the microprocessor during experimental studies (Morimoto and Tsutsumi, 2009). Overcoming these problems is only possible through simulators (Gorgunoglu *et al*., 2012). Nowadays, VHDL has become one of the most popular hardware languages (Alaer *et al*., 2007).

The educational microprocessor architecture is considered through the study of the microprocessor program controlled registers, data types, instruction formats, addressing modes, instruction set and interrupt mechanism (Ackovska and Ristov, 2013). Design and simulation of sequencer and Arithmetic logic unit, using VHDL, are considered in detail.

ARCHITECTURE DESCRIPTION OF THE EDUCATIONAL MICROPROCESSOR

The microprocessor is the brain of the computer. It is composed of several parts, like data path, control path and memory units. At each clock cycle, the control unit is needed to generate the control signals automatically for operating the data path (Chadha *et al*., 2011; Tiejun and Fang, 2008). Moreover, the sequencer represents the main block of the control unit. The second part deals with the Arithmetic Logic Unit, which perform arithmetic computations such as addition, subtraction, multiplication, division etc. (Hwang, 2004). The educational microprocessor consists of an 8-bit data bus and address bus. Figure 1 depicts microprocessor architecture.

Registers are used to store the value of instructions during processing. It acts as a memory unit for temporary storage of data. The number of registers available and their size leads to the determination of the power and speed of microprocessor (Daghooghi, 2013; Kamble and Mhala, 2012). In fact, key registers include: Address Register (A.R), Data Register (D.R), Instruction Register (I.R), AL and BL registers which are the ALU accumulators, Code Condition Register (CCR) and the Program Counter (PC) (Goyal, 2014). In addition, the microprocessor concept can be

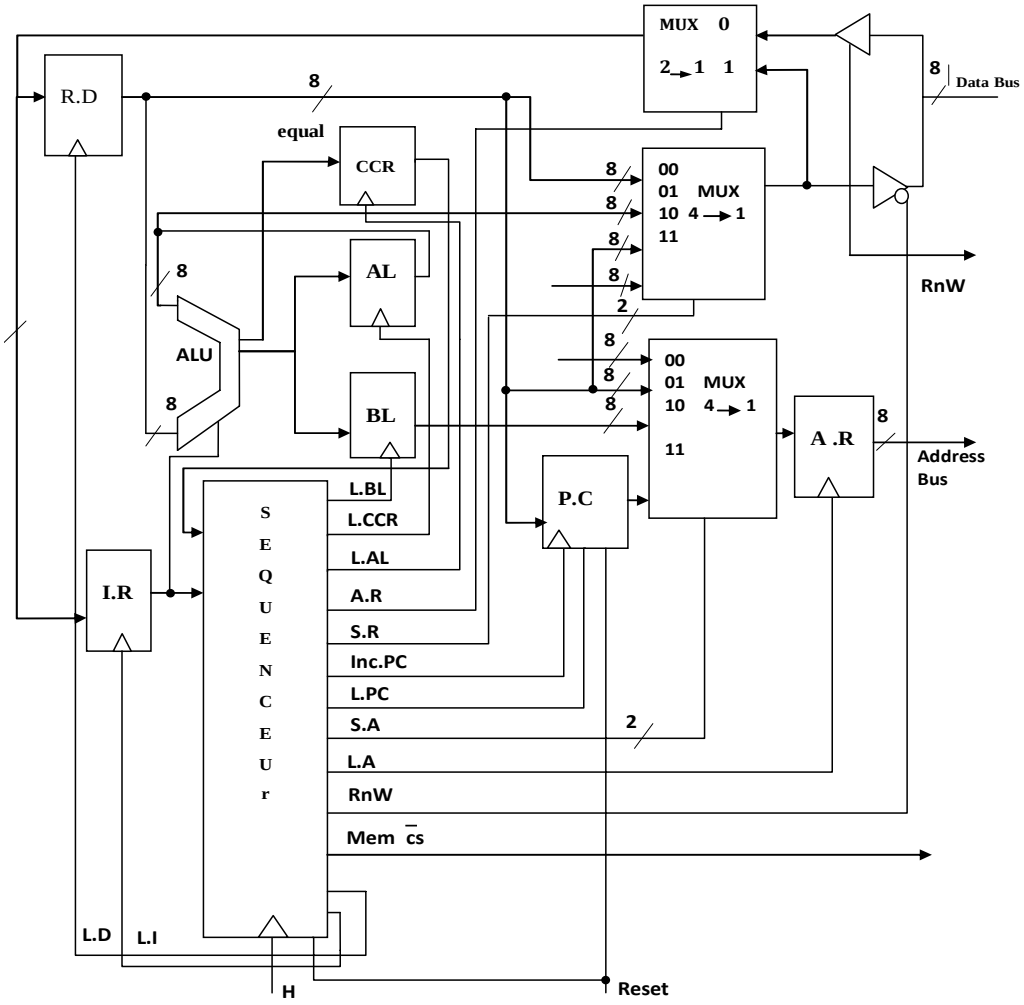

Fig. 1: Block diagram of the 8-bit microprocessor

summarized in instruction steps which begins from his address serial transfer by the program counter to the address decoder. Thereafter, the selected memory cell will be loaded by the microprocessor in one or two cycles and treated by the ALU according to the code given by the sequencer.

Given the above overview of the proposed educational microprocessor, we can easily examine its various blocks in details in the following sections. In particular, we presented the design of the sequencer and Arithmetic logic unit using VHDL.

Table 1: Signals description

Signals	Description
L.BL	Load BL
L.AL	Load AL
L.CCR	Load CCR
S.R	Select register
Inc.PC	Increment Pointer Counter
L.PC	Load Pointer Counter
S.A	Select Address
L.A	Load Address
R\W	Read or write
mem cs	memory chip select
Inc .BC	Load Block Counter
L.D	Load data
L.I	Load instruction

SEQUENCER DESIGN METHOD

This study was conducted In 2016 at the university of Sfax.

The sequencer repeats indefinitely an operation sequence which will be stopped by the reset signal. The sequencer operations are such as the operation code loading, the addressing mode processing and the instruction processing associated to the operation code and the addressing mode. Figure 2 depicts sequencer architecture.

Sequencer behavioral study: The sequencer can be defined by Petri net which represents a sequential set of actions by a symbolic graphic as shown in Fig. 3. In order to accomplish these operations, the sequencer must send control signals to all participating blocks to the instruction execution. Different signals are described in Table 1.

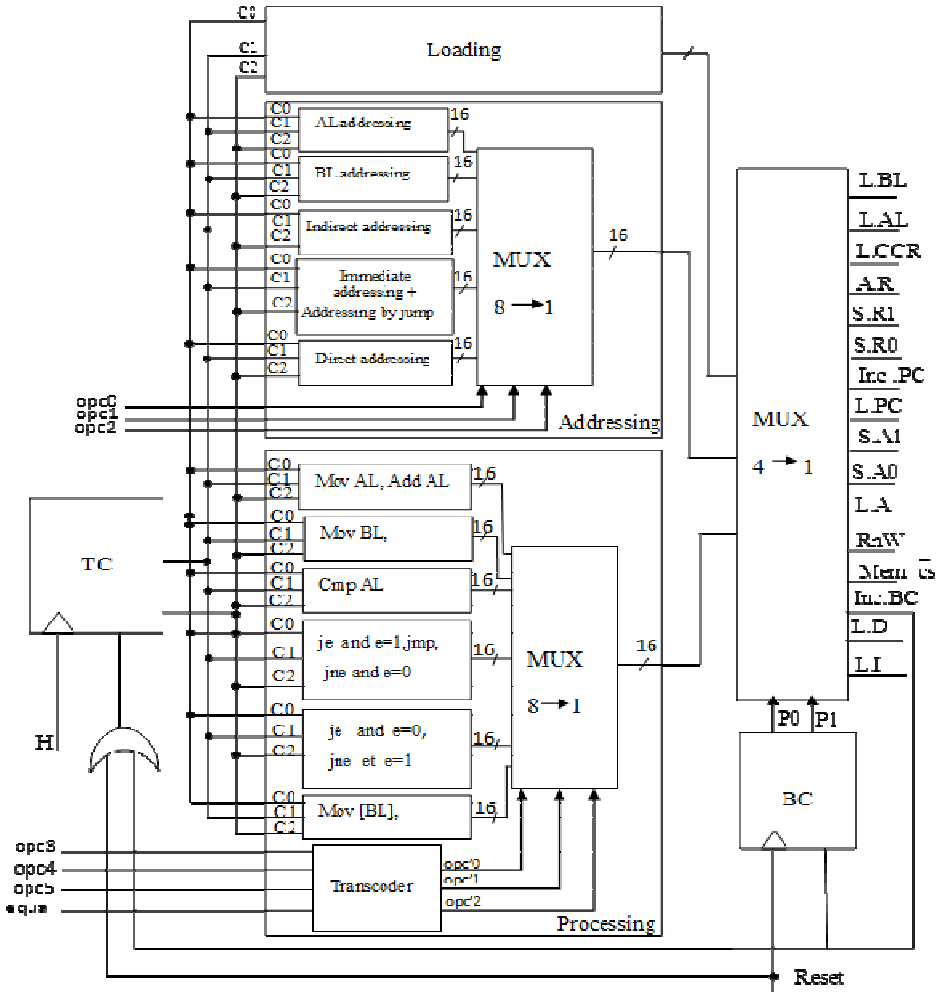

Fig. 2: Block diagram of the sequencer

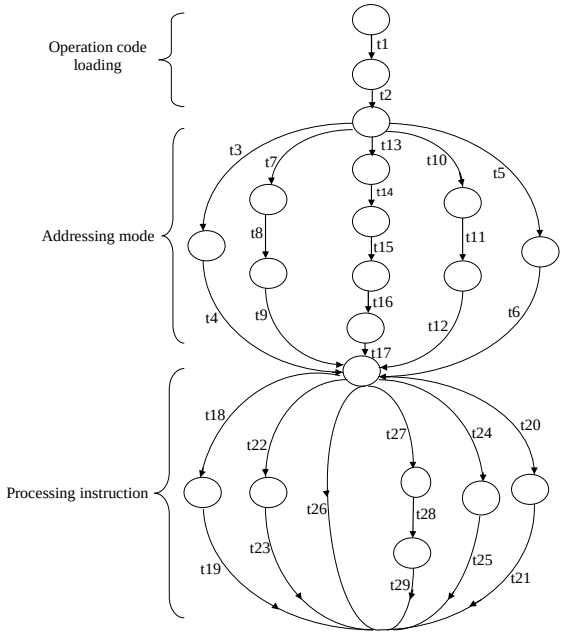

Fig. 3: Symbolic graphic of sequencer

Table 2: Loading block transitions

t	Condition	Action	t	Condition	Action
1	C0=0	L.BL=0	2	C0=1	L.BL=0
	C1=0	L.AL=0		C1=0	L.AL=0
	C2=0	L.CCR=0		C2=0	L.CCR=0
		A.R=0			A.R=0
		S.R=00			S.R=00
		Inc.PC=0			Inc.PC=0
		L.PC=0			L.PC=0
		S.A=00			S.A=00
		L.A=1			L.A=0
		R\W=1			R\W=0
		mem cs=0			mem cs=1
		Inc. .BC=0			Inc.BC=1
		L.D=0			L.D=0
		L.I=1			L.I=1

We distinguish two sequencers types: hardwired and micro-programmed. For our application, we have chosen the hardwired sequencer which is implemented with gates and other digital circuits. It has the advantage that it can be optimized to produce a fast mode of operation (Al-Haija *et al*., 2013). Moreover, the sequencer dedicated for educational microprocessor has a low complexity architecture. In addition, each sequencer transition is associated with some actions presented in binary form which constitutes a sub-instruction. The loading block, the address block and the processing block transitions are given in Table 2 to 4 respectively.

Table 3: Address block transitions

t	Condition	Action	t	Condition	Action	t	Condition	Action
3	C0=0	L.BL=0	4	C0=1	L.BL=0	5	C0=0	L.BL=0
	C1=0	L.AL=0		C1=0	L.AL=0		C1=0	L.AL=0
	C2=0	L.CCR=0		C2=0	L.CCR=0		C2=0	L.CCR=0
		A.R=1			A.R=1			A.R=1
		S.R=01			S.R=01			S.R=10
		Inc.PC=0			Inc.PC=1			Inc.PC=0
		L.PC=0			L.PC=0			L.PC=0
		S.A=00			S.A=00			S.A=00
		L.A=0			L.A=0			L.A=0
		R\W=0			R\W=0			R\W=0
		mem cs=1			mem cs=1			mem cs=1
		Inc. .BC=0			Inc. .BC=1			Inc. .BC=0
		L.D=0			L.D=1			L.D=0
		L.I=0			L.I=0			L.I=0
6	C0=1	L.BL=0	7	C0=0	L.BL=0	8	C0=1	L.BL=0
	C1=0	L.AL=0		C1=0	L.AL=0		C1=0	L.AL=0
	C2=0	L.CCR=0		C2=0	L.CCR=0		C2=0	L.CCR=0
		A.R=1			A.R=0			A.R=0
		S.R=10			S.R=00			S.R=00
		Inc.PC=1			Inc.PC=0			Inc.PC=0
		L.PC=0			L.PC=0			L.PC=0
		S.A=00			S.A=01			S.A=01
		L.A=0			L.A=1			L.A=0
		R\W=0			R\W=1			R\W=1
		mem cs=1			mem cs=0			mem cs=0
		Inc. .BC=1			Inc .BC=0			Inc .BC=0
		L.D=1			L.D=0			L.D=1
		L.I=0			L.I=0			L.I=0
9	C0=0	L.BL=0	10	C0=0	L.BL=0	11	C0=1	L.BL=0
	C1=1	L.AL=0		C1=0	L.AL=0		C1=0	L.AL=0
	C2=0	L.CCR=0		C2=0	L.CCR=0		C2=0	L.CCR=0
		A.R=0			A.R=0			A.R=0
		S.R=00			S.R=00			S.R=00
		Inc.PC =1			Inc.PC =1			Inc.PC=0
		L.PC=0			L.PC=0			L.PC=0
		S.A=00			S.A=00			S.A=00
		L.A=0			L.A=1			L.A=0
		R\W=0			R\W=1			R\W=1
		mem cs=1			mem cs=0			mem cs=0
		Inc .BC=1			Inc .BC=0			Inc .BC=0
		L.D=0			L.D=0			L.D=1
		L.I=0			L.I=0			L.I=0
12	C0=0	L.BL=0	13	C0=0	L.BL=0	14	C0=1	L.BL=0
	C1=1	L.AL=0		C1=0	L.AL=0		C1=0	L.AL=0
	C2=0	L.CCR=0		C2=0	L.CCR=0		C2=0	L.CCR=0
		A.R=0			A.R=0			A.R=0
		S.R=00			S.R=00			S.R=00
		Inc.PC=1			Inc.PC=1			Inc.PC=0
		L.PC=0			L.PC=0			L.PC=0
		S.A=00			S.A=00			S.A=00
		L.A=1			L.A=0			L.A=0
		R\W=0			R\W=1			R\W=1
		mem cs=1			mem cs=0			mem cs=0

Table 3: Continue

		Inc .BC=1			Inc .BC=0			Inc .BC=0
		L.D=0			L.D=0			L.D=1
		L.I=0			L.I=0			L.I=0
15	C0=0	L.BL=0	16	C0=1	L.BL=0	17	C0=0	L.BL=0
	C1=1	L.AL=0		C1=1	L.AL=0		C1=0	L.AL=0
	C2=0	L.CCR=0		C2=0	L.CCR=0		C2=1	L.CCR=0
		A.R=0			A.R=0			A.R=0
		S.R=00			S.R=00			S.R=00
		Inc.PC=0			Inc.PC=0			Inc.PC=1
		L.PC=0			L.PC=0			L.PC=0
		S.A=10			S.A=10			S.A=00
		L.A=1			L.A=0			L.A=0
		R\W=1			R\W=1			R\W=0
		mem cs=0			mem cs=0			mem cs=1
		Inc .BC=0			Inc .BC=0			Inc .BC=1
		L.D=0			L.D=1			L.D=0
		L.I=0			L.I=0			L.I=0

Table 4: Processing block transitions

t	Condition	Action	t	Condition	Action	t	Condition	Action
18	C0=0	L.BL=0	19	C0=1	L.BL=0	20	C0=0	L.BL=1
	C1=0	L.AL=1		C1=0	L.AL=0		C1=0	L.AL=0
	C2=0	L.CCR=0		C2=0	L.CCR=0		C2=0	L.CCR=0
		A.R=0			A.R=0			A.R=0
		S.R=00			S.R=00			S.R=00
		Inc.PC=0			Inc.PC=0			Inc.PC=0
		L.PC=0			L.PC=0			L.PC=0
		S.A=00			S.A=00			S.A=00
		L.A=0			L.A=0			L.A=0
		R\W=0			R\W=0			R\W=0
		mem cs=1			mem cs=1			mem cs=1
		Inc .BC=0			Inc .BC=1			Inc .BC=0
		L.D=0			L.D=0			L.D=0
		L.I=0			L.I=0			L.I=0
21	C0=1	L.BL=0	22	C0=0	L.BL=0	23	C0=1	L.BL=0
	C1=0	L.AL=0		C1=0	L.AL=0		C1=0	L.AL=0
	C2=0	L.CCR=0		C2=0	L.CCR=1		C2=0	L.CCR=0
		A.R=0			A.R=0			A.R=0
		S.R=00			S.R=00			S.R=00
		Inc.PC=0			Inc.PC=0			Inc.PC=0
		L.PC=0			L.PC=0			L.PC=0
		S.A=00			S.A=00			S.A=00
		L.A=0			L.A=0			L.A=0
		R\W=0			R\W=0			R\W=0
		mem cs=1			mem cs=1			mem cs=1
		Inc .BC=1			Inc .BC=0			Inc .BC=1
		L.D=0			L.D=0			L.D=0
		L.I=0			L.I=0			L.I=0
24	C0=0	L.BL=0	25	C0=1	L.BL=0	26	C0=0	L.BL=0
	C1=0	L.AL=0		C1=0	L.AL=0		C1=0	L.AL=0
	C2=0	L.CCR=0		C2=0	L.CCR=0		C2=0	L.CCR=0
		A.R=0			A.R=0			A.R=0
		S.R=00			S.R=00			S.R=00
		Inc.PC=0			Inc.PC=0			Inc.PC=0
		L.PC=1			L.PC=0			L.PC=0
		S.A=00			S.A=00			S.A=00
		L.A=0			L.A=0			L.A=0
		R\W=0			R\W=0			R\W=0
		mem cs=1			mem cs=1			mem cs=1
		Inc .BC=0			Inc .BC=1			Inc .BC=1
		L.D=0			L.D=0			L.D=0
		L.I=0			L.I=0			L.I=0
27	C0=0	L.BL=0	28	C0=1	L.BL=0	29	C0=0	L.BL=0
	C1=0	L.AL=0		C1=0	L.AL=0		C1=1	L.AL=0
	C2=0	L.CCR=0		C2=0	L.CCR=0		C2=0	L.CCR=0
		A.R=0			A.R=0			A.R=0
		S.R=00			S.R=00			S.R=00
		Inc.PC=0			Inc.PC=0			Inc.PC=0
		L.PC=0			L.PC=0			L.PC=0
		S.A=01			S.A=00			S.A=00
		L.A=1			L.A=0			L.A=0
		R\W=0			R\W=0			R\W=0

Table 4: Continue

	mem $\overline{cs}$=1 Inc .BC=0 L.D=0 L.I=0	mem $\overline{cs}$=0 Inc .BC=0 L.D=0 L.I=0	mem $\overline{cs}$=1 Inc .BC=1 L.D=0 L.I=0

Table 5: Addressing types

opc (2, 1,0)	Syntax	Description
000	AL	AL addressing
001	BL	BL addressing
010	[BL]	Indirect addressing
011	constant	Immediate addressing
100	constant	Direct addressing

Table 6: Processing types

opc(5,4,3)	Syntax	Description	Example
000	mov AL,	AL loading	mov AL, [100]
001	mov BL,	BL loading	mov BL, 0
010	add AL,	Addition on AL	add AL, [BL]
011	cmp AL,	AL comparison	cmp AL, 10
100	je	Jump if equal	je 100
101	jne	Jump if different	jne 100
110	jmp	Jump	jmp 100
111	mov [BL]	Memory backup	mov [BL], AL

Table 7: Truth table of loading block

C2	C1	C0	mem $\overline{cs}$	R\W	L.A	L.I	Inc. BC
0	0	0	0	1	1	1	0
0	0	1	1	0	0	1	1

Table 8: Processing transcoder operation codes

opc'2, opc'1, opc'0	Instruction type
000	mov AL, Add AL
001	mov BL
010	cmp AL
011	je and e=1, jne and e=0, jmp
100	je and e=0, jne and e=1
101	mov [BL]

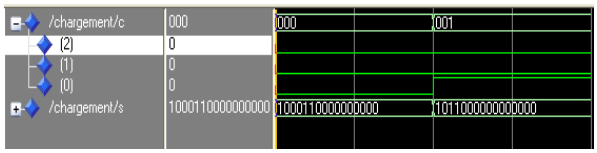

Fig. 4: Loading block simulation

Microprocessor instruction set: A machine instruction occupying one or two bytes must provide the microprocessor all the information to initiate an elementary operation. In fact, the machine instruction must contain in the first field an operation code (opc) and an address or a constant depending on the addressing type in the second field (Hayne, 2011). In our application, the operation code occupies 6 bits such as 3 bits are reserved for the instruction type, the other 3 bits serve to define the loading addressing mode. We chose a well-defined coding, with the instruction set syntax close to the Pentium. The addressing and processing types are represented by Table 5 and 6 respectively.

Different blocks design: Design and simulation of the sequencer sub-circuits follow the steps of configured sub-instructions in the transition tables. For example, the truth table of loading block is given by Table 7. Moreover, the loading block equations are presented as follows:

$$mem\ \overline{cs} = In.BC = \overline{C}2.\overline{C}1.C0 \quad (1)$$

$$L.I = \overline{C}2.\overline{C}1 \quad (2)$$

$$R/W = L.A = \overline{C}2.\overline{C}1.\overline{C}0 \quad (3)$$

where, mem $\overline{cs}$, Inc. BC, L.I, R\W and L.A are the loading block signals. In addition, C0, C1 and C2 are the loading block inputs. We validate these equations by simulation as presented in Fig. 4.

We used a processing transcoder in order to minimize the processing sub-circuits. In fact, we tried to regroup instructions that require the same treatment approach and we have associated new operation codes (opc'2, opc'1, opc'0). Table 8 illustrates the processing transcoder operation codes.

DESIGN METHOD OF ARITHMETIC LOGIC UNIT

The arithmetic logic unit is the most essential block in microprocessor since it is concerned with arithmetic, logical and decision, making operations such as AND, OR, NOT, NAND, etc. (Singh *et al*., 2011; Zhang *et al*., 2012). The ALU internal architecture (Fig. 5) can be described as follows. The 8-bit comparator has as inputs the AL and RD registers outputs and as output the 'equal' signal which is equal to '1' if the result is equal to '0'. Moreover, we have used a follower logic gate in the ALU to treat the 'mov' instruction. In fact, this instruction directs the data register output to store it in one of the AL or BL registers, according to the instruction addressing mode. In addition, the transcoder encodes the operation code 3 bits (5, 4 and 3) in one bit which define the multiplexer selection input.

We need a single selection signal for the 2-to-1 MUX. For this, we chose the bit '0' for the 'mov' instruction and '1' for the 'add' instruction. The truth table of UAL transcoder is given by Table 9. The output equation of UAL transcoder (S) is presented as follows. Moreover, We validate these equations by simulation as presented in Fig. 6:

$$S = opc5.\overline{opc4}.opc3 \quad (4)$$

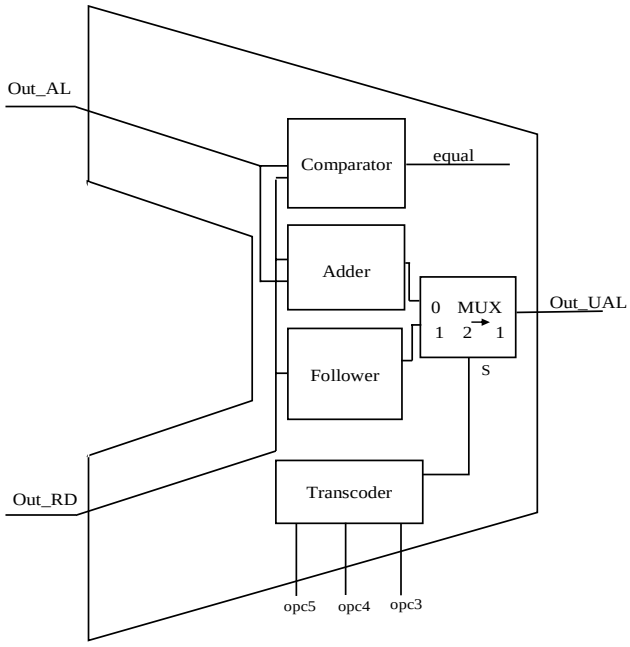

Fig. 5: ALU internal architecture

Fig. 6: UAL transcoder simulation

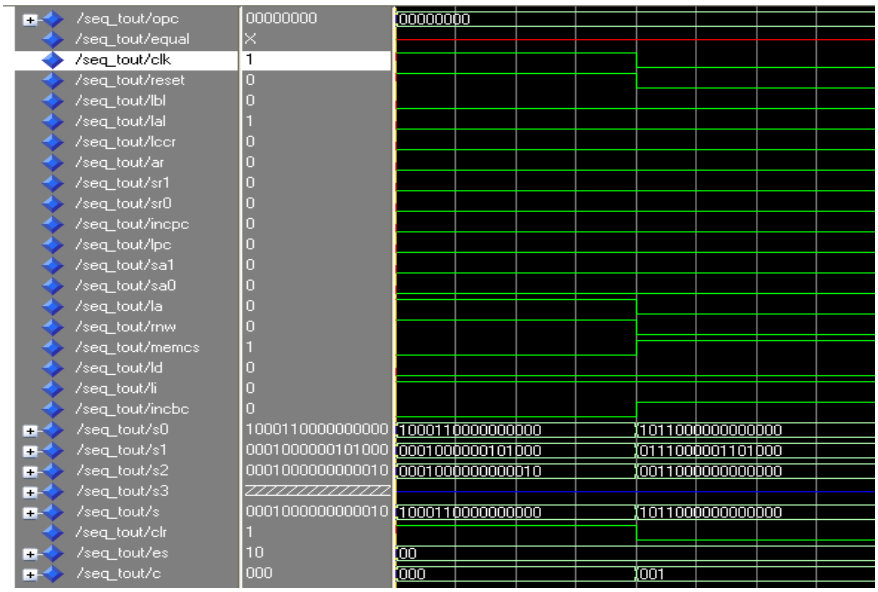

Fig. 7: Simulation results of sequencer

RESULTS AND DISCUSSION

We used the following signals list to make the link between different sequencer internal circuits:

- C: 3-bit vector which designates the transition counter output.
- S0, S1, S2: 16-bit vectors which designate the output signals of the loading, addressing and processing blocks respectively. S3 is a high impedance input in the 4-to-1 Multiplexer (MUX).
- es: input selection of the 4-to-1 MUX on 2 bits.
- S: 4-to-1 MUX output of 16 bits.
- Clr: OR gate output, which enters the transition counter

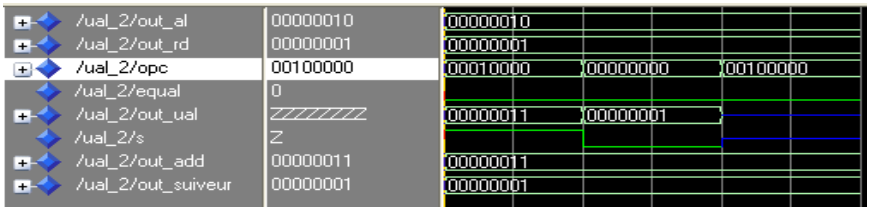

Fig. 8: Simulation result for arithmetic logic unit

Table 9: Truth table of UAL transcoder

Opc5	Opc4	Opc3	S
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	H
1	0	0	H
1	0	1	H
1	1	0	H
1	1	1	0

In Fig. 7, we deduce that for an operation code equal to '00000000' (mov or Add AL, AL addressing) and at each falling edge of the clock, one of the blocks signals appears similar to the transition table. We performed at each sequencer output a clue 'S' (example: L.BL <= S(0)).

We observed from the simulation results (Fig. 8) that the ALU implemented by the above description, worked successfully for all the input combinations and the select operation codes according to the given specification in Table 9. Where out_ual, out_add and out_follower are the outputs of the ALU, the adder and the follower respectively.

CONCLUSION

In this study, the design and simulation of 8-bit educational microprocessor architecture using VHDL were achieved. In fact, the motivation is to link digital electronics with microprocessor architecture and, therefore, show the students how a microprocessor can be built using the simple components they already know. In addition, the microprocessor is the brain of the computer. It is composed of several parts, like data path, control path and memory units. At each clock cycle, the control unit is needed to generate the control signals automatically for operating the data path. Moreover, the sequencer represents the main block of the control unit. The second part deals with the Arithmetic Logic Unit, which perform arithmetic computations. Then, we presented the design and simulation of the sequencer and Arithmetic logic unit using VHDL. The results obtained are satisfactory and are in accordance with theoretical expectations.

CONFLICT OF INTEREST

This study was supported by the Higher Institute of Computer Sciences and Multimedia of Sfax, university of Sfax, Tunisia.

REFERENCES

Ackovska, N. and S. Ristov, 2013. Hands-on improvements for efficient teaching computer science students about hardware. Proceeding of the IEEE Global Engineering Education Conference (EDUCON), Germany, pp: 295-302.

Alaer, E., A. Tangel and M. Yakut, 2007. "MIB-16" FPGA based design and implementation of a 16-bit microprocessor for educational use. Proceeding of the 6th WSEAS International Conference on Circuits, Systems, Electronics, Control and Signal Processing. Cairo, Egypt, pp: 284-288.

Al-Haija, Q.A., H. Al-Amri, M. Al-Nashri and S. Al-Muhaisen, 2013. An engineering design of 4-bit special purpose microprogrammed processor. Proc. Comput. Sci., 21: 512-516.

Chadha, A., D. Jyoti and M.G. Bhatia, 2011. Design and simulation of an 8-bit dedicated processor for calculating the sine and cosine of an angle using the CORDIC algorithm. Proceeding of the IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pp: 1-6.

Daghooghi, T., 2013. Design and development MIPS processor based on a high performance and low power architecture on FPGA. Int. J. Mod. Educ. Comp. Sci., 5(5): 49-59.

Gorgunoglu, S., M. Peker, B. Sen and A. Cavusoglu, 2012. An efficient Pseudo microprocessor for engineering education. Proc. Technol., 1: 36-43.

Goyal, S., 2014. 16 bit microprocessor -design and simulations in VHDL. Int. J. Innov. Res. Dev., 3(1): 153-156.

Gupta, N., P. Gupta, H. Bajpai, R. Singh and S. Saxena, 2014. Analysis of 16 bit microprocessor architecture on FPGA using VHDL. Int. J. Adv. Res. Elect. Electr. Instrum. Eng., 3(4): 8979-8986.

Hayne, R.J., 2011. An instructional processor design using VHDL and an FPGA. Proceeding of the American Society for Engineering Education Annual Conference and Exposition, Vancouver, BC.

Hwang, E.O., 2004. Microprocessor Design Principles and Practices with VHDL. Ebook, Brooks-Cole. Retrieved from: https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxzaXRpb2ZwZ2F8Z3g6NzViNGU3MjM5Zjk5NTBmNw.

Kamble, S. and N.N. Mhala, 2012. VHDL implementation of 8-bit ALU. IOSR J. Electron. Commun. Eng., 1(1): 7-11.

Morimoto, T. and T. Tsutsumi, 2009. Development of instruction analysis tool for microprocessor design education. Proceeding of the 17th International Conference on Computers in Education, Hong Kong.

Presa, J.L.L. and E.P. Calle, 2011. MMP16 a 16-bit didactic micro-programmed micro-processor. Proceeding of the 3rdInternational Conference on Computer Research and Development (ICCRD), Shanghai, China.

Singh, R.R., A. Tiwari, V.K. Singh and G.S. Tomar, 2011. VHDL environment for floating point arithmetic logic unit-ALU design and simulation. Proceeding of the IEEE International Conference on Communication Systems and Network Technologies (CSNT). Katra, Jammu, India.

Tiejun, X. and L. Fang, 2008. 16-bit teaching microprocessor design and application. Proceeding of the IEEE International Symposium on IT in Medicine and Education. Xiamen, China.

Zhang, H., Z.Q. Wang, W. Liu and Z. Tan, 2012. The design of arithmetic logic unit based on ALM. Proc. Eng., 29: 1969-1973.