"MIB-16" FPGA Based Design and Implementation of a 16-Bit Microprocessor for Educational Use

ESMA ALAER, ALI TANGEL, MEHMET YAKUT
Electronics and Communication Engineering
Kocaeli University
Muhendislik Fakultesi Veziroglu Yerleskesi 41040, Izmit
TURKEY
esma-esen@hotmail.com atangel@kou.edu.tr myakut@kou.edu.tr
http://mf.kou.edu.tr/elohab

Abstract: - This paper presents a design and FPGA implementation of a 16-bit microprocessor core, so called "MIB-16" using VHDL. The microprocessor can directly access to the memory which consists of 16-bit words, addressed by a 16-bit word-address. Instructions are all multiples of 16-bit words, and are stored in this memory. There are 16 general purpose registers (R0–R15), a program counter (PC) and a condition code register (CC). The microprocessor can execute 16 instructions such as add, subtract, multiply, divide, load and store. The complete design is realized and verified on Xilinx Spartan-3 Evaluation Board. "MIB-16" is suitable especially for educational purposes and for FPGA based industrial digital system-on-chip ASIC solutions as being an easy to use basic microprocessor core. Whereas, professional VHDL or Verilog based microprocessor core libraries offered by the FPGA vendors are expensive, more complex to be modified, and consume too many resources in an FPGA chip.

Key-Words: - Microprocessors, FPGA, VHDL, Digital System Design

1 Introduction

The increased complexity in electronic systems requires the development of new design methodologies. For this reason, traditional methods of "use pencil and paper to design the circuit" and "implement to do experiments" have been replaced by "define and synthesis" methods [1].

These new methods have resulted in development of Hardware Description Languages (HDLs). Nowadays, VHDL (Very High Speed Integrated Circuit Hardware Description Language) has become one of the most popular hardware languages [2]. The source capacity (intensity) and the maximum signal frequency of the reconfigurable systems have been increased in parallel with the developments in technology [3].

After 1990s, field programmable gate arrays (FPGAs) have also been popular in custom ASIC design world due to having the fastest time to market property. They also allow designer to combine macro cell designs to form digital system-on-chip solutions. Nowadays, there are different design methods for a system implementation using FPGA architectures. HDLs are the most preferable methods among others due to resulting in reduced design period and cost [4]. FPGAs have especially led to the development of designs in high level description languages like VHDL or Verilog, which allow the designer to conceive the design at the level of RTL without reference to the final technology or vendor used for the final implementation [5].

The earliest studies on microprocessor designs go back to the invention of transistor in 1948s, and it has still been continuing nowadays. The Intel's first child, 4004 in 1971, was able to run at 740 KHz performance. However, recent microprocessor designs have reached the performance of over 3 Giga Hertz.

Several microprocessor designs based on FPGAs are reported in the literature [3], [6], [7], [8]. In this study, a complete design of an FPGA based 16-bit microprocessor is presented especially for educational purposes. This paper is the extended version of the authors' earlier work, [9].

2 FPGA Based Processor Design Steps and Instruction Set

Fig.1 shows the FPGA design flow in general for the FPGA ASIC solutions [10]. Fig.2 shows the architecture of the designed microprocessor in this study. The processor has 16-bit address bus and 16-bit data bus. In addition, it has 16 general purpose registers, a program counter, and a 3-bit status register. Every word has 16-bit word length.

3-bit status register is updated after every arithmetic and logic operation. Z (zero) flag indicates if the result of an operation is zero. Similarly, N flag is for negative result. V flag is for the indication of overflow situation. Fig.3 shows the register structures of the designed processor. Fig.4 shows the schematic representation of the behavioral models of each VHDL core library elements specifically designed for the targetted microprocessor.

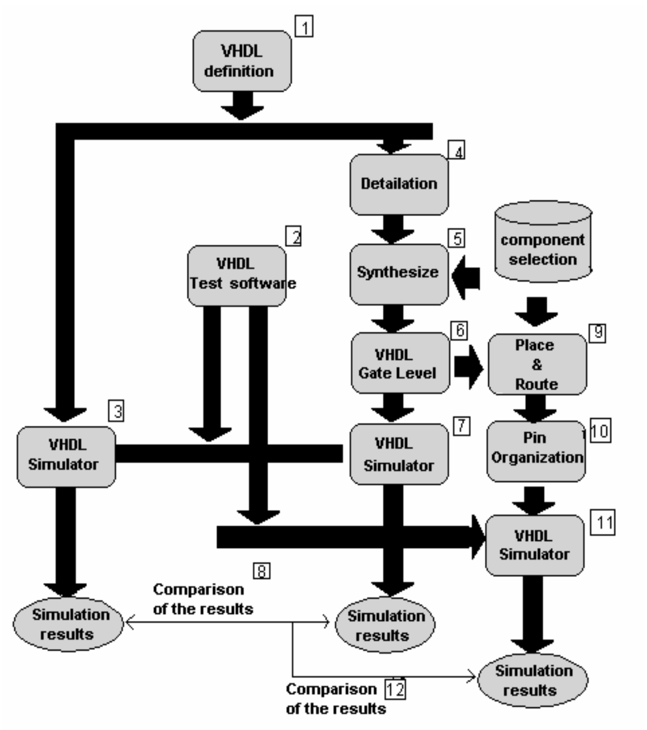


Fig.1 FPGA Design Cycle in general

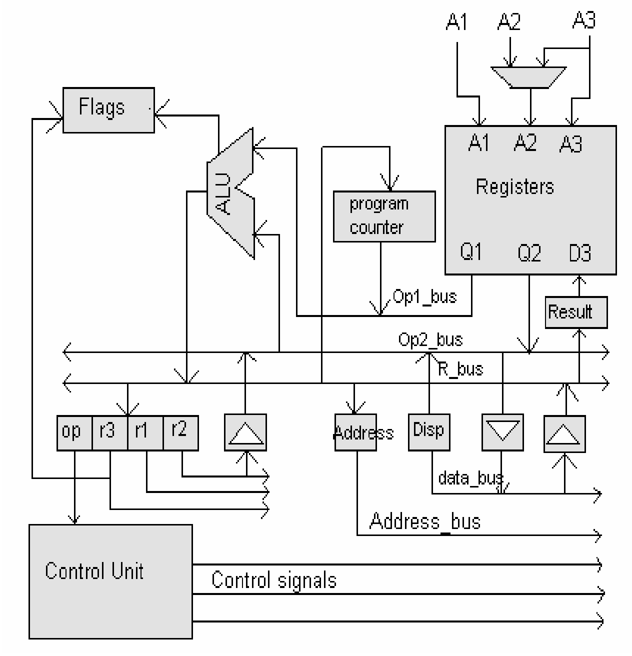


Fig.2 The architecture implemented

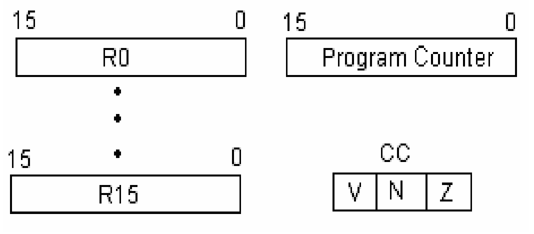


Fig.3 Register Structures

The microprocessor has an external memory, which has 16 bit word-length and 16 bit address bus to store the instructions. All instructions have 16-bit length. The PC register contains the address of the next instruction to be executed. After each instruction word is fetched, the PC is incremented by one to point to the next word. The arithmetic and logic instructions are listed in Table 1.

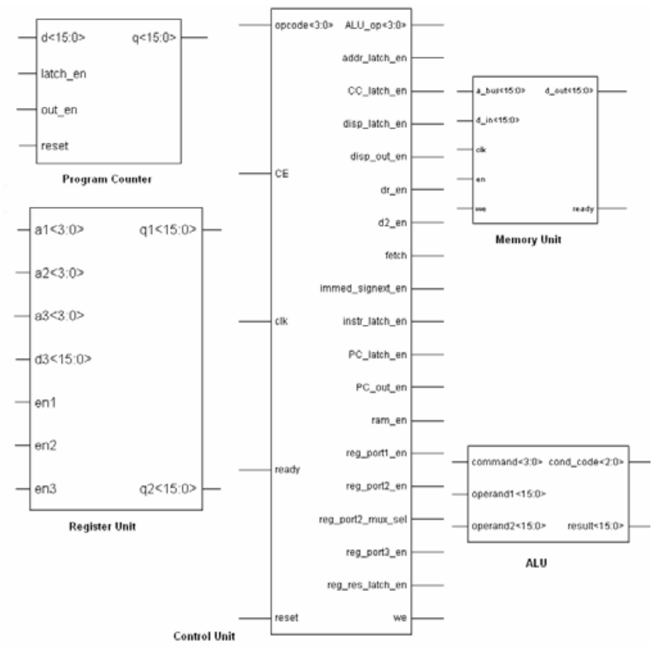


Fig.4. Schematic representation of the designed VHDL core library behavioral models

Instruction set is divided into four sections which are 4-bit each as shown in Fig. 5. The first section is for opcode. r3 shows the address of the location where the result is stored. r1 and r2 are source register addresses, and i8 is an immediate two-compliment integer operand.

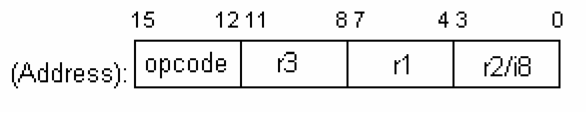


Fig.5. Format for the arithmetic and logic instructions

Table-1. Arithmetic and logic instructions

Instruction	Name	Function	Opcode
Add	Add	r3 ← r1+r2	0000
Sub	Subtract	r3 ← r1-r2	0001
Mul	Multiply	r3 ← r1*r2	0010
Div	divide	r3 ← r1/r2	0011
Addq	Add quick	r3 ← r1+i8	0100
Subq	Subtract quick	r3 ← r1-i8	0101
Mulq	Multiply quick	r3 ← r1*i8	0110
Divq	Divide quick	r3 ← r1/i8	0111
Land	LogicalAND	r3 ← r1&r2	1000
Lor	Logical OR	r3 ← r1!r2	1001
Lxor	Logical XOR	r3 ← r1+r2	1010
Brq	Branch quick	if cond then PC ← PC+i8	1011

Table 2 shows the load and store instructions. *Load from memory* and *store into memory* instructions have two format depending on the length of the displacement address. The format for the long and short displacement are shown in Fig. 6-a and 6-b, respectively.

The op field is the op-code, r3 specifies the register to be loaded or stored, r1 is used as an index register, disp is a long immediate displacement, and i8 is a short immediate displacement.

Table 2. Load and Store Instructions

Instructions	Name	Function	Opcode
Ld	Load	r3 ←M[r1+disp16]	1100
St	Store	M[r1+disp16] ← r3	1101
Ldq	Load quick	r3 ← M[r1+i8]	1110
Stq	Store quick	M[r1+i8] ←r3	1111

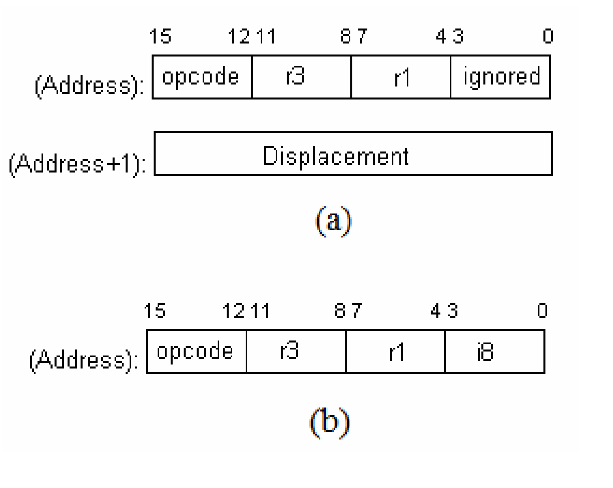


Fig.6 Format for load and store instructions a) long displacement b) short displacement

3 Instruction Executions

The I/O pin configuration for the microprocessor is shown in Fig.7. Firstly, the processor puts the address information of the data to be reached in the memory to the address bus for the READ operation. WE is kept at low (logic “0”) in this case. If the data to be read is an instruction information, the FETCH is set to active. Secondly, the information is transferred to the data bus. If the read operation is finished, READY signal is set to active. Otherwise, READY signal is remained in passive mode until the read operation is completed.

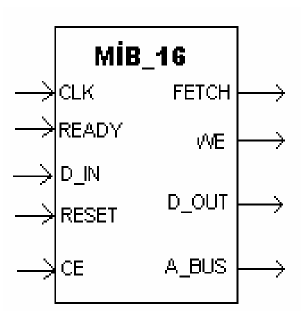


Fig.7 Pin configuration of the designed processor

Fig.8 shows the signal waveforms for a READ operation. The clock frequency is set to 50 MHz which is the value on Spartan-3 Eval-board, on which the processor is implemented. For the WRITE operation, the memory address information of the data to be written is firstly transferred to the address bus. The FETCH signal is set to passive mode and WE is tied to logic high. Secondly, the data to be written is carried to the data bus; then the WRITE operation starts. After the WRITE operation is completed, the READY is set to active. Fig.9 shows the signal waveforms for the WRITE operation.

If the ADD operation is taken as an example of the arithmetic operations, the process executes as follows:
The microprocessor first reads the instruction for ADD operation from the opcode. Two data to be added are received from the defined registers, and then added. The resultant information is written into the defined register. The result is related to the flags accordingly. Fig.10 shows the Model Sim simulation code execution of the ADD operation.

Here, *a_bus* shows the address of the next instruction to be executed. *d_in* shows the input data bus of the microprocessor, which includes the instruction data. *op1_bus* and *op2_bus* indicates the data to be added, reg_result indicates the result. The flags are kept in *alu_cc*.

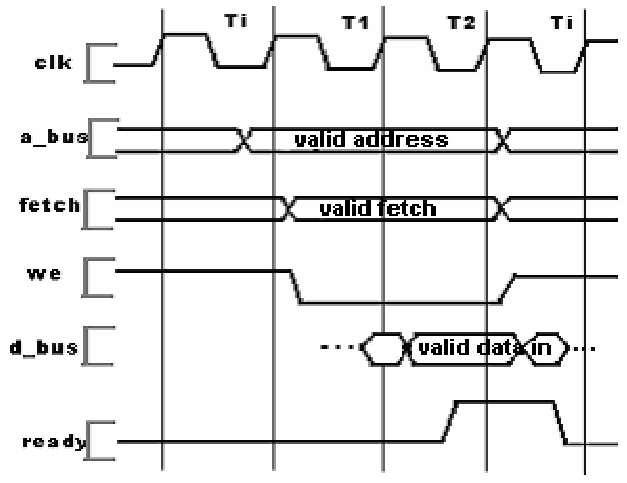


Fig.8 Signals for the READ operation

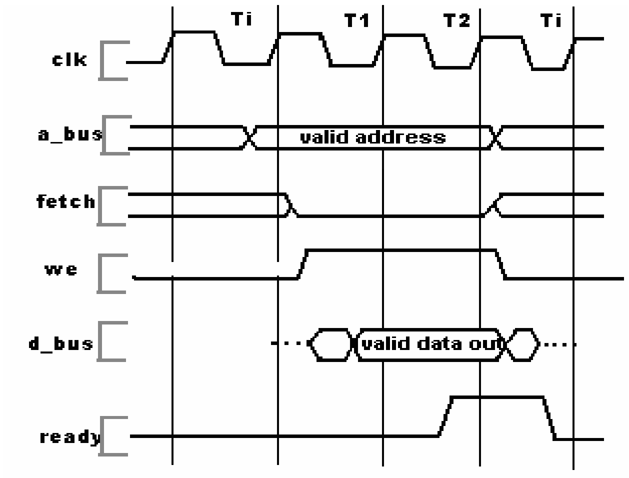
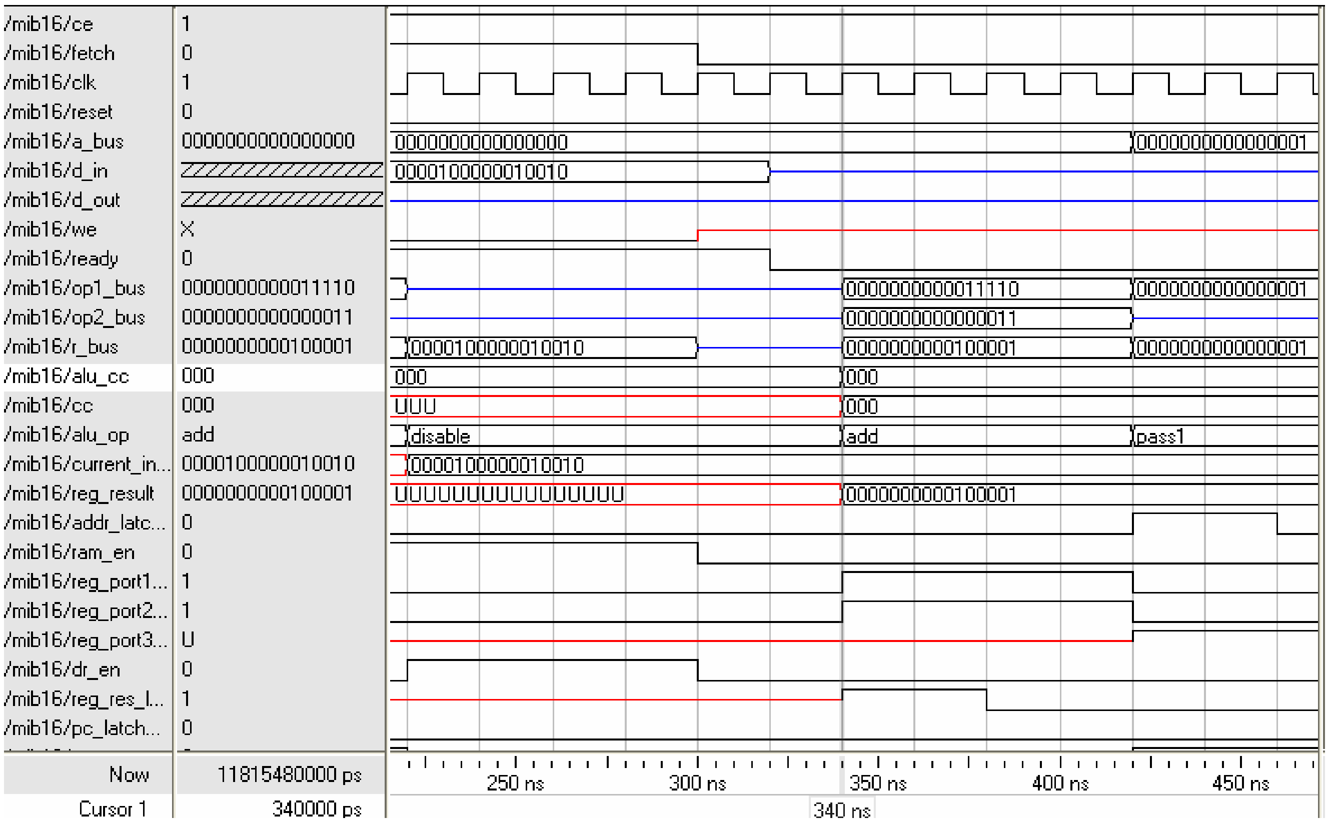


Fig.9 Signals for the WRITE operation



/mib16/ce	1
/mib16/fetch	0
/mib16/clk	1
/mib16/reset	0
/mib16/a_bus	0000000000000000
/mib16/d_in	//////////////////
/mib16/d_out	//////////////////
/mib16/we	X
/mib16/ready	0
/mib16/op1_bus	0000000000011110
/mib16/op2_bus	0000000000000011
/mib16/r_bus	0000000000100001
/mib16/alu_cc	000
/mib16/cc	000
/mib16/alu_op	add
/mib16/current_in...	0000100000010010
/mib16/reg_result	0000000000100001
/mib16/addr_latc...	0
/mib16/ram_en	0
/mib16/reg_port1...	1
/mib16/reg_port2...	1
/mib16/reg_port3...	U
/mib16/dr_en	0
/mib16/reg_res_l...	1
/mib16/pc_latch...	0
Now	11815480000 ps
Cursor 1	340000 ps

Fig.10 Modelsim simulation of code execution for ADD operation

4 Conclusions

A 16-bit microprocessor so called “MİB_16” is designed using VHDL, and also implemented on Xilinx Spartan-3 Evaluation board. Simulation and implementation tools used are Xilinx ISE and Modelsim. There were some limitations encountered when the program uploading and testing due to the evaluation board limitations.

Multiplication operation could be implemented for the cases when the resulting number doesn’t exceed 16-bit. Division operation is achieved only for the cases when resulting numbers are without residue. The arithmetic and logic operations are performed in about 340 ns, load and store commands are performed in 660 ns, quick load and quick store commands are performed in 440 ns. In other words, the performance of the microprocessor realized is 3 MHz for the arithmetic and logic operations; 1.5MHz for load and store operations; and 2.3 MHz for quick load and store operations.

Table-3. Design Summary

Number of errors:	0
Number of warnings:	19
Logic Utilization:	
Total Number Slice Registers:	453 out of 3,840 11%
Number used as Flip Flops:	87
Number used as Latches:	366
Number of 4 input LUTs:	2,260 out of 3,840 58%
Logic Distribution:	
Number of occupied Slices:	1,227 out of 1,920 63%
Number of Slices containing only related logic:	1,227 out of 1,227 100%
Number of Slices containing unrelated logic:	0 out of 1,227 0%
*See NOTES below for an explanation of the effects of unrelated logic	
Total Number 4 input LUTs:	2,281 out of 3,840 59%
Number used as logic:	2,260
Number used as a route-thru:	21
Number of bonded IOBs:	34 out of 173 19%
IOB Flip Flops:	4
Number of Block RAMs:	1 out of 12 8%
Number of MULT18X18s:	1 out of 12 8%
Number of GCLKs:	2 out of 8 25%
Total equivalent gate count for design:	89,497
Additional JTAG gate count for IOBs:	1,632
Peak Memory Usage:	86 MB

The clock frequency was set to 50 MHz, which is the evaluation board value. In fact the processor can work at higher clock frequencies. Another limitation was the capacity of the RAM available on the Eval Board, which resulted in reduction of the address bus from 16-bit to 10-bit for the testing purposes only.The RAM with the largest capacity uses 10-bit address bus. The switches, surface mounted LEDs and LCDs on the board are used for different purposes during the verification of the complete microprocessor. The design summary is given in Table-3.

It is believed that this processor core can also be adapted into low-speed FPGA-based System On Chip Industrial ASIC solutions beside its educational use. Students can easily modify the microprocessor such as replacing different instruction commands, and they can use it in their any FPGA implementation of digital system design projects requiring a basic core processor inside.

References:

[1] Bezarra, E. A. Gough, M.P. “A Guide to Migrating from Microprocessor to FPGA Coping the Support Tool Limitations”, *ELSEVIER Microprocessor and Microsystems* 23,1999, pp. 561-572.

[2] Herman, H. S., Srihari, C., Matthew, M., “Pipeline Reconfigurable FPGAs”, *Journal of VLSI Signal Processing Systems”,*2000*,* pp. *24,* 129-146.

[3] Borgatti, M., Lertora, F., Foret, B., Cali L., “A Reconfigurable System Featuring Dynamically Extensible Embedded Microprocessor, FPGA and Customizable I/O”, *IEEE Custom Integrated Circuits Conference,* 2002, pp. 13-16.

[4] Janiszewski, I., Baraniecki, R., Siekierska, K. “A reusable microcontroller core’s design”, *IEEE, VHDL International Users Forum Fall Workshop (VIUF ’99),* 1999, pp. 14-21.

[5] Jurado-Carmona, F.J., Tombs, J., Aguirre, M.A., Torralba, A., “Implementation of a fully pipelined ARM compatible microprocessor core” *XVII Design on Circuits and Integrated Systems Conference,* 2002, pp. 559-563.

[6] Davidson, J. “FPGA Implementation of a Reconfigurable Microprocessor” *IEEE Custom Integrated Circuits Conference*, 1993, pp. 3.2.1-3.2.4

[7] Sueyoshi, T., Kuga, M., and Shibamura, H., “KITE Microprocessor and CAE for Computer Science”, *Systems and Computers in Japan*, Vol. 33, No. 8, 2002, pp.64-74.

[8] Pastor, J. S., Gonzalez, I., Lopez, J., Arribas, F.G, Martinez, J. “A Remote Laboratory for Debugging FPGA-Based Microprocessor Prototypes”, *Proceedings of the IEEE International Conference on Advanced Learning Technologies (ICALT’04),*2004.

[9] Alaer, E., Tangel, A., Yakut, M. "MIB-16 FPGA based Design and Implementation of a 16 Bit Microprocessor for Educational Use", *6th WSEAS International Conf. on Circuits, Systems, Electronics, Control&Signal processing*, *Cairo-Egypt*, 2007, pp. 284-288.

[10] Cakıroglu, M. “*Gerçek Zaman Sayma Birimi Iceren SAU80C51 Mikro denetleyicisinin FPGA Mimarileri Kullanilarak Geliştirilmesi*”, MsC Thesis, Sakarya University, Turkey, pp. 29-32, (2003).