"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

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 frequency of the microprocessor is only 3 MHz for an operand such as add, subtract, multiply and divide and approximately 1.5 MHz for the operands, load and store due to the restrictions of the evaluation board, on which the system is implemented. 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 available basic processor core whenever needed.
Key-Words: - Microprocessors, FPGA, VHDL, Digital Systems

1 Introduction

The increased complexity in electronic systems requires the development of design methodologies. For this reason, traditional methods of "use pencil and paper to design the circuit", and "implement to do experiments" have been replaced with "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 paralel 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 continiuing nowadays. The Intel's first child, 4004 in 1971, was able to run at 740 KHz performance. However, recent microprocessor designs has reached the performance of over 3 Giga Hertz.

Several microprocessor designs based on FPGA are reported in the literature [3], [7], [8], [9]. In this study, a complete design of an FPGA based 16-bit microprocessor is presented especially for educational purposes.

2 FPGA Based Processor Design Steps and Instruction Set

Fig.1 shows the FPGA design flow in general for the FPGA ASIC solutions [6]. Fig.2 shows the architecture of the designed microcomputer 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 if any. Fig.3 shows the register structures of the designed processor.

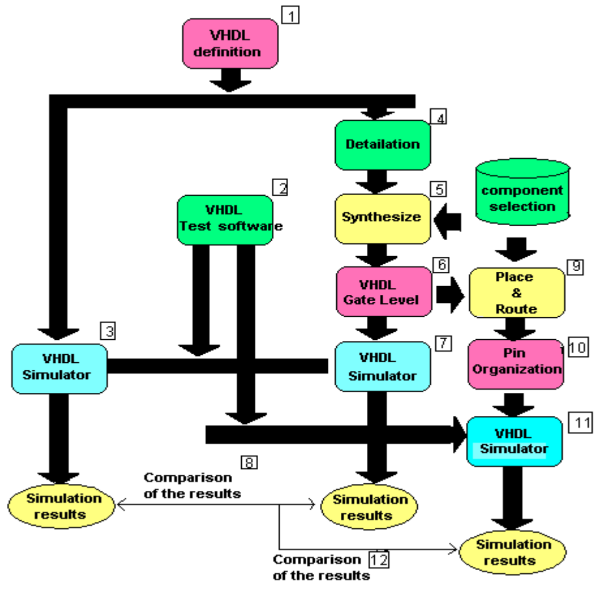

Fig.1 FPGA Design Cycle in general

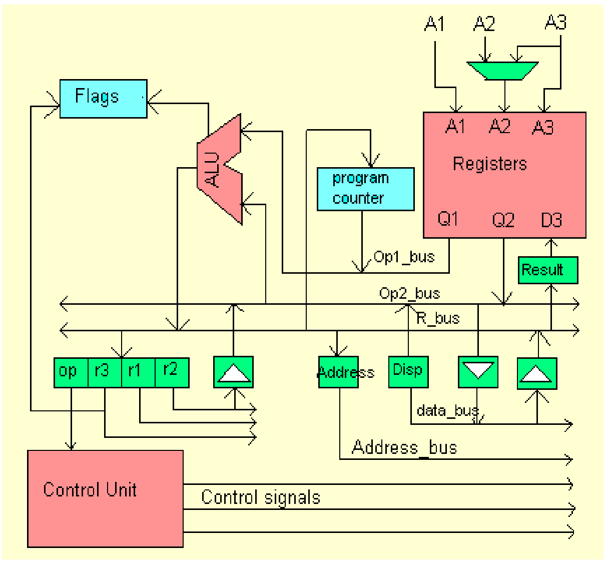

Fig.2 The Architecture Implemented

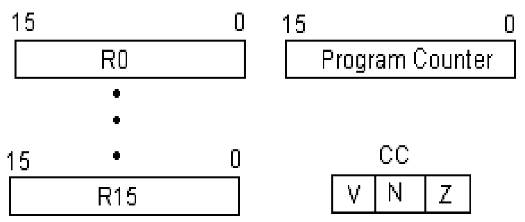

Fig.3 Register Structures

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.

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	Logical AND	r3←r1&r2	1000
Lor	Logical OR	r3 ← r1!r2	1001
Lxor	Logical XOR	r3← r1+r2	1010
Lmask	Logic Mask	r3←r1&-r2	1011

Instruction set is divided into four sections which are 4-bit each. 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.

	15 12	11 8	7 4	3 0
Address:	opcode	r3	r1	r2/i8

Table 2 shows the load and store instructions. *Load from memory* and *store into memory* instructions have two formats depending on the length of the displacement address. The format for the long displacement is:

	15 12	11 8	7 4	3 0
Address :	opcode	r3	r1	ignored
(Addres+1):	Displacement			

The format for the short displacement is as follows:

	15 12	11 8	7 4	3 0
(Address) :	opcode	r3	r1	i8

The op field is the op-code, r3 specifies the register to be loaded or stored, r1is 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

3 Instruction Executions

The I/O pin configuration for the microprocessor is shown in Fig.4 Firstly, the processor puts the address information of the data to be reached in the memory to the address bus for 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 data bus. If the read operation is finished, READY signal is set to active. Otherwise READY signal is remained in passive mode until read operation is completed.

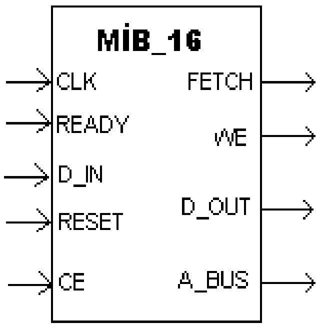

Fig.4 Pin configuration of the designed microprocessor

Fig.5 shows the signal waveforms for the READ operation. The clock frequency is set to 50 MHz which is the value on Spartan-3 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 high (logic "1"). Secondly, the data to be written is carried out to the data bus, then the WRITE operation starts. After write is completed, the READY is set to active. Fig.6 shows the signal waveforms for the WRITE operation.

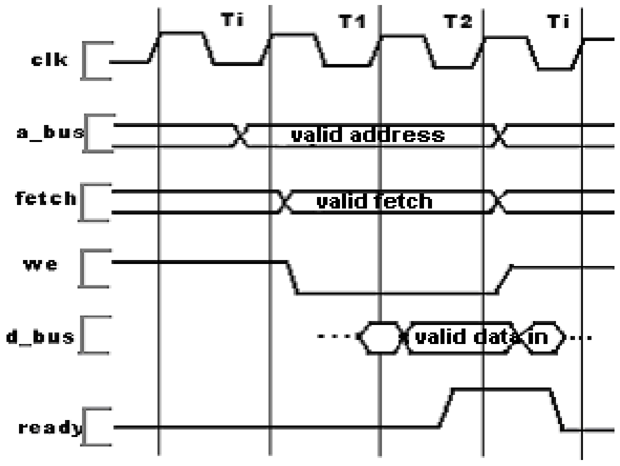

Fig.5 Signals for the READ operation

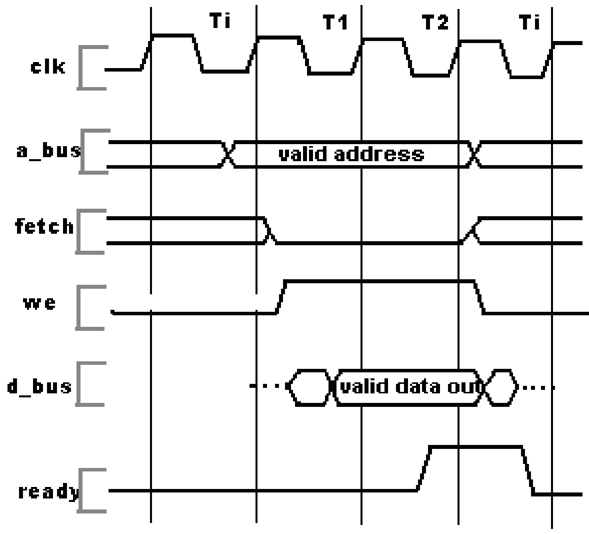

Fig.6 Signals for the WRITE operation

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 in the defined register. The result is related to the flags accordingly. Fig.7 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*.

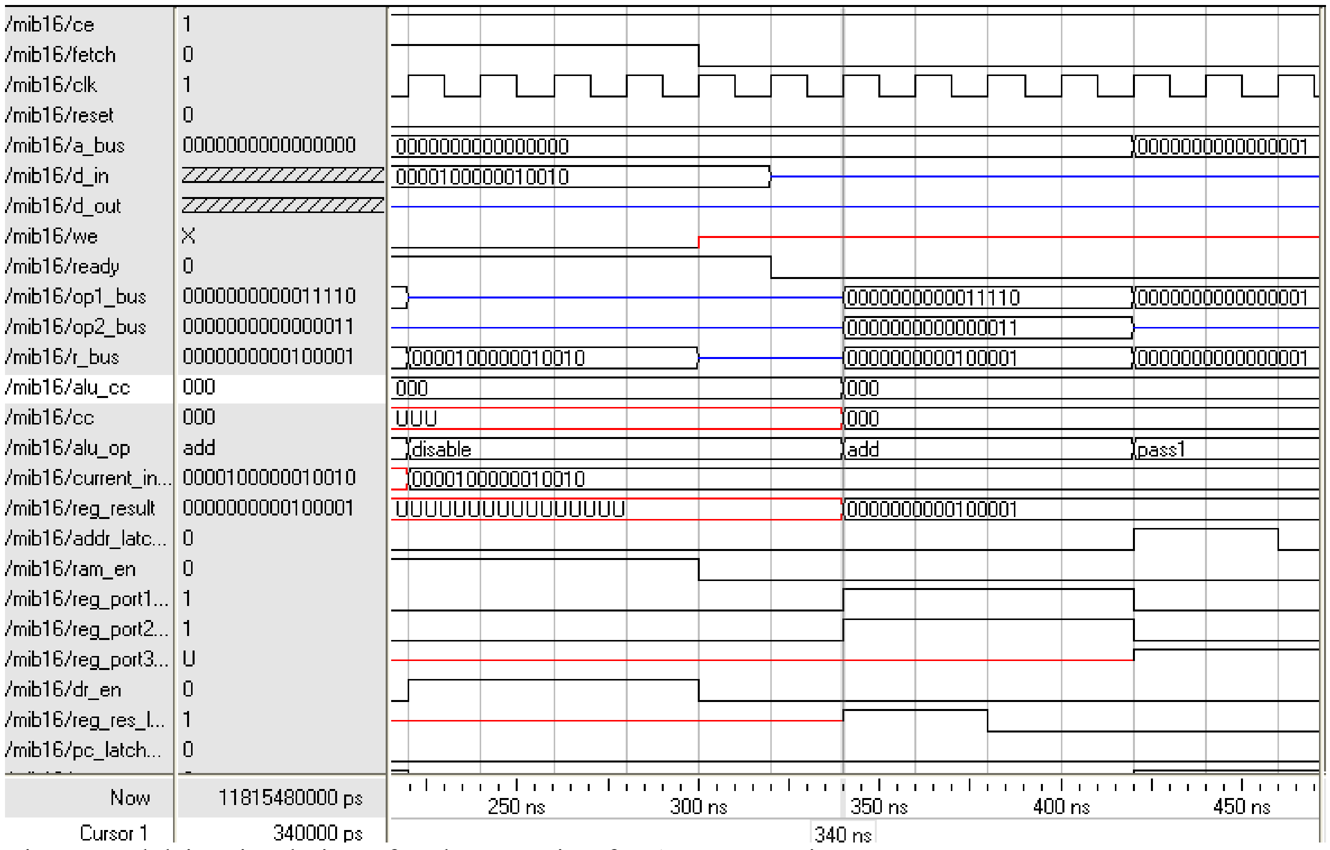

Fig.7 Modelsim simulation of code execution for ADD operation

4 Results and Conclusion

A 16-bit microprocessor so called MİB_16 is designed using VHDL and also implemented on Xilinx Spartan-3 Evaluation board as shown in Fig.8. Simulation and imlementation tools used are Xilinx ISE 6.3i and Model Sim 6.0. There were some limitations encountered during the uploading and testing of the processor due to evaluation board capacity and limitations.

Multiplication operation could only be implemented for the resulting number of not exceeding 16-bit. Division operation is achieved only for the resulting numbers 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, and 1.5 MHz for load and store operations, and 2.3 MHz for the quick load and store operations.

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 test board. Resulting a conversion on the size of the address bus from 16-bit to 10-bit for the testing purposes only. Because, 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 for the verification of the complete microprocessor.

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.

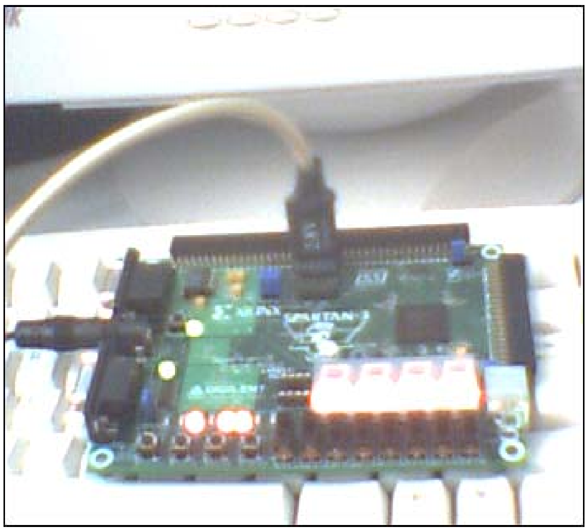
Fig.8 Experimental Setup

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,* 561-572, (1999)

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

[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,* pp. 13-16, (2002).

[4] Ireneusz Janiszewski, Robert Baraniecki, Krystyna Siekierska, "A reusable microcontroller core's design", *IEEE, VHDL International Users Forum Fall Workshop (VIUF '99),* pp. 14-21, (1999).

[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, DCIS'02*, pp. 559-563. 2002

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

[7] J. Davidson " FPGA Implementation of a Reconfigurable Microprocessor" *IEEE Custom Integrated Circuits Conference, 1993.*

[8] T. Sueyoshi, M. Kuga, and H. Shibamura, "KITE Microprocessor and CAE for Computer Science", *Systems and Computers in Japan*, Vol. 33, No. 8, 2002

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