International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - NEWS 2016 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

Custom 8 Bit Microprocessor Designing and Implementation on FPGA Board

Gautam R. Gare, Kenneth Peterand Amaresh L. R.

Abstract: ***The day to day activities of every person is highly influenced and dependent on Micro-processors. Since their invention, they have revolutionised the world. The design and developments of such processor has become the centre of attraction in VLSI technology. To address the need of speedy development, Field-programmable gate array (FPGA)are being used for testing, as they are reprogrammable. In this paper, we describe the designing of a very simple 8 bit micro-processor based on Von-Neumann architecture using VHDL language, capable of carrying out 16 instructions and its implementation on Spartan 6 FPGA Board. The entire design is implemented using basic gates so to check for optimal performance on the FPGA Board. 190 (5%) of the slices were used. A maximum frequency of 375.094MHz was reached with a minimum period of 2.666ns.***

Keywords: Field-Programmable Gate Array (FPGA), Micro-processor (µp), Register (Reg), Data, Instruction (Inst), Opcode, Control Unit (CU), Arithmetic Logic Unit (ALU), Memory.

I. INTRODUCTION

With the birth of Embedded System, Micro-processor (µp) design has been in the lime light. With its complexity and design increasing exponentially, it has revolutionized our World. From computers to cell phones, satellites and watches, all are based on the Micro-processor.

FPGA as a design tool is very reliable and fast to implement. It is a great platform for testing designs as it is easily reprogrammable. Further these designs can be mass produced as ASIC (Application Specific Integrated Circuit) chips and released to the market. Thus FPGA enables rapid transition from lab designing to market implementation.

In this paper we design a basic 8 bit Micro-processor using just basic gates and implement it on a Spartan 6 FPGA Board. Here we use Von-Neumann architecture based design with a 16 byte RAM, 5 Registers, a Control Unit and an 8 bit ALU capable of carrying out any 16 custom designed instructions.

II. RELATED WORK

The ease of implementing and testing designs on FPGA has led to many significant works based on FPGA. But only a few deal with the actual design of a processor.

In [1], the authors discuss the ease of implementation of a simple processor, with very basic and generic processor design.

In [2], the cost and configurability benefits of implementing a floating point processor array for a high precision dot product using FPGA is described. In [3], the use of FPGA as a computer simulator were discussed with rapidity, accuracy, re-configurability, and cost factors under consideration. In this paper, we present the design and implementation of an8 bit micro-processor on a Spartan 6 FPGA Board using Xilinx ISE foundation 14.2 and VHDL language.

III. INSTRUCTION SET

Any design of a micro-processor begins with an Instruction set, which decides the processing capabilities of the micro-processor. According to one's requirements and intended use of the micro-processor, its instruction set is decided. The 8 bit Micro-processor that we have designed has 16 instructions, each with their corresponding opcodes.

Table 1. Opcodes With Corresponding Instructions

Opcode	Instruction
0000	HLT
0001	LOAD A
0010	LOAD B
0011	OUTPUT
0100	NOP
0101	JUMP
0110	SLL
0111	SLR
1000	ADD
1001	A and B
1010	A or B
1011	not B
1100	SUB
1101	not A and B
1110	not A or B
1111	not B

International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - NEWS 2016 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

In our design the 4 MSB bits represent the opcode of the instruction (inst) and the 4 LSB bits represent the address at which this instruction is to be carried about.

Opcode				Address			
8	7	6	5	4	3	2	1

The HLT inst is used to halt the execution of the program. LOAD A (or B) inst is used to load the data from the corresponding Reg to Reg A (or B). NOP inst is used as a delay were no new inst is processed in that cycle. JUMP inst is used to change the point of execution of the code. SLR(or SLL) inst is used to shift the data to the right (or left) by 1 bit. OUTPUT inst is used to display the value stored in that Reg. The rest of the inst which has a 1 in MSB of the opcode are ALU operations we perform, such as addition, subtraction, and, or & not operations.

IV. MICROPROCESSOR DESIGN

Our Micro-processor design consists of:

- Functional blocks – ALU, CU, RAM, CLK and PC.
- Temporary Reg – A, B, MAR, IR & OR.

These components are in turn made of ring counters, latches, flip-flops, multiplexer, demultiplexer, buffer and basic gates.

In our architecture we use an 8 bit common BUS that connects all the various components and acts as a mean for Data and Inst transfer.

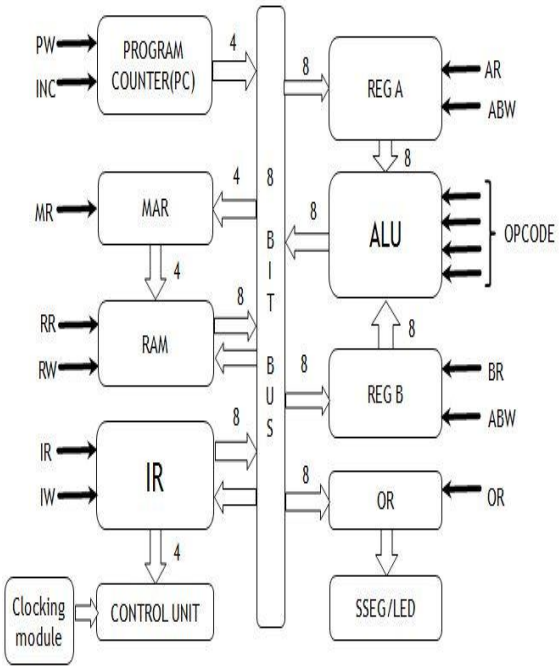

Fig. 1. Block Diagram of 8 Bit Micro-Processor

A. Control Unit

CU is the module that co-ordinates the entire process of the micro-processor. Our CU is designed using basic gates, unlike the usual finite state machine (FSM) based design. It consists of a 6 state ring counter, which divides a machine cycle into 6 states. The first 3 states are used to fetch the inst and next 3 states are used to execute the inst.

Based on the opcode provided by the IR reg, CU sets the various control bits in sequence depending on which state of execution it is in and the opcode.

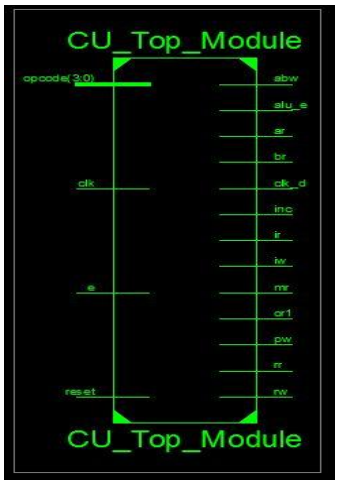

Fig. 2. Control Unit Module With Inputs and Outputs

B. Arithmetic Logic Unit (ALU)

The ALU is used to carry out arithmetic operations such as Addition & Subtraction and Logical operations such as AND, OR, NOT, SRL (Shift right logical) and SLL (Shift left logical).

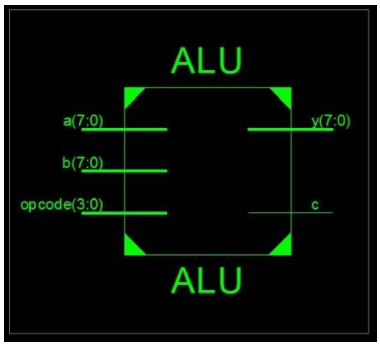

Fig. 3. ALU Module With Inputs and Outputs

C. Memory (RAM & Registers)

The memory unit of the micro-processor consists of a 16 byte RAM and 5 Temporary Reg.

The RAM is used to store both Data and Inst. As only 16 byte of memory is available, we need to fit the entire

International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - NEWS 2016 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

program along with the data in this memory space. This is a constraint which can be resolved by using additional memory in future designs.

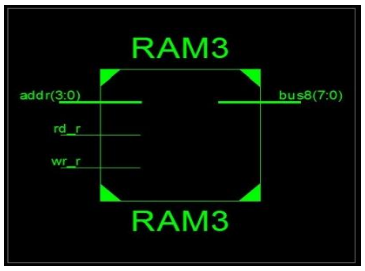

Fig. 4. RAM Module With Inputs and Outputs

The 5 temporary Reg are used as buffer memory while carrying out the execution of the program. The temporary Reg are:

Reg MAR (Memory Access Register) is a 4 bit register, which acts as a buffer Reg to store the address the RAM has to point to.

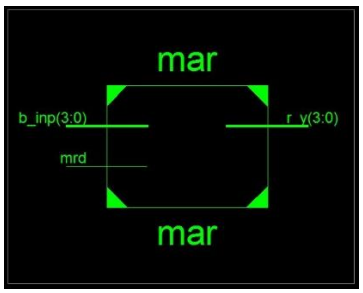

Fig. 5. MAR Reg Module With Inputs and Outputs

*Reg IR (Instuction Register)*is an 8 bit register which holds the current inst being executed. It provides opcode to the CU and the ALU. It also provide the address to the MAR on which the current inst is to be performed.

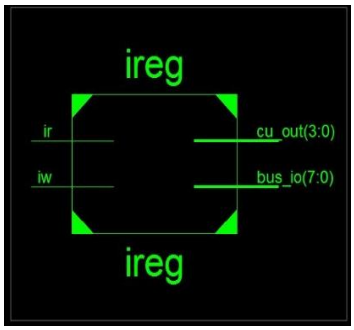

Fig. 6. IR Reg Module With Inputs and Outputs

Reg OR (Output Register) is an 8 bit register which holds the value to be displayed.

Reg A & B are 8 bit registers which hold the value on which the ALU performs the operation.

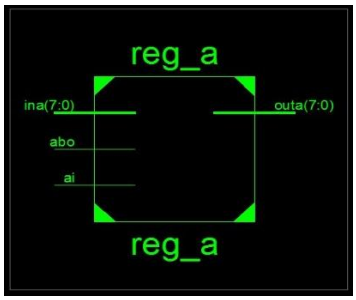

Fig. 7. A (B/OR) Reg Module With Inputs and Outputs

D. Clock

CLK module is used to provide clock for the PC, ring counter and a D flip-flop (used to prevent debounce). It also performs clock division on the internal clock of Spartan 6 Board which has a frequency of 100MHz to provide the appropriate clock frequency for the design.

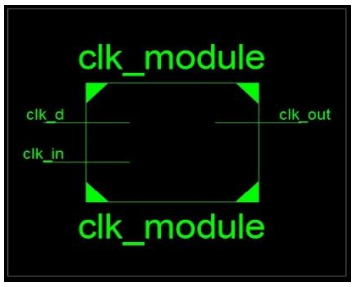

Fig. 8. CLK Module With Inputs and Outputs

E. Program Counter (PC)

PC is the Program counter which is nothing but a 4 bit up-counter made of D flip-flops, buffer and basic gates. It provides the address of the next inst to be executed to the MAR.

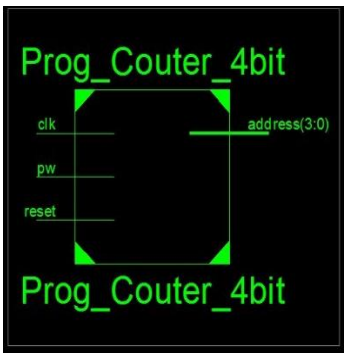

Fig. 9. PC Module With Inputs and Outputs

Apart from the above mentioned blocks we also require an additional functional block, which is used to change the mode of operation of the micro-processor, i.e. from program mode to execution mode and visa-versa.

International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - NEWS 2016 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

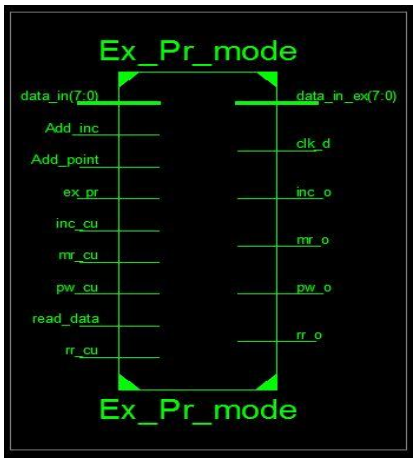

Fig. 10. Execute/Program Mode Module With Inputs and Outputs

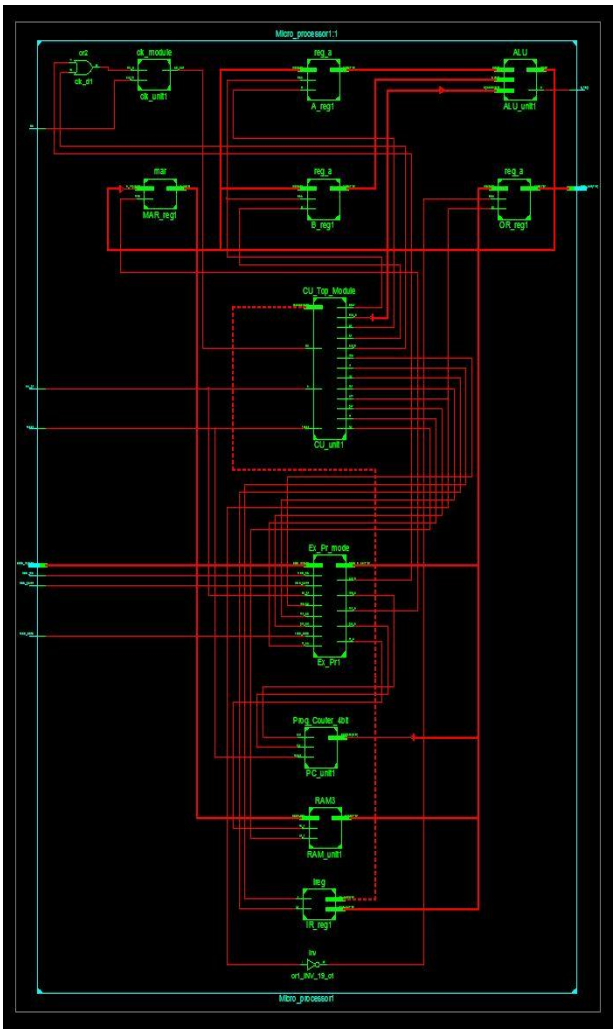

Fig. 11. Micro-Processor Top Integration Module With Inputs and Outputs

A Micro-processor top module is also needed, where in all the fuctional blocks and temporary registers are interconnected.

V. WORKING OF MICROPROCESSOR

A program to be executed by our micro-processor is first uploaded onto the RAM using various input pins provided on the Spartan 6 FPGA Board, keeping the micro-processor in program mode in which clock is disabled. Then we set the micro-processor into execution mode in which the clock is enabled.

First PC gives the address of the 1st inst to be executed which is read by the MAR thus setting the RAM to point to this address. Next PC is incremented. The inst stored in this address is loaded into IR reg. This is called the Fetch Cycle which constitutes the first 3 states of a cycle. This is how each and every inst is loaded for execution.

The next 3 states of the cycle are for the execution of the inst. Let the inst to be executed be "00110111". This corresponds to output inst (as opcodei.e MSB 4bits are '0011'), in which data stored in address '0111' (i.e LSB 4bits) is to be outputted. Here the CU first sets the RAM to point to the address '0111' by storing the 4 LSB bits of IR reg into MAR. Then CU loads the 8bit data stored at this address in the RAM into the OR reg.

This is how CU coordinates the various functional block for the execution of an inst.

VI. IMPLEMENTATION AND RESULTS

The entire micro-processor was designed one component at a time using VHDL language, with each component tested and evaluated individually. The final Architecture is shown here:

Then the entire design was tested and evaluated for different programs. A simulation waveform is shown here, where a test program performs NOT operation on the binary data input "00110011" (data_in) and the result is "11001100" (data_out) as seen in the figure:

International Journal of Electrical Electronics & Computer Science Engineering
Special Issue - NEWS 2016 | E-ISSN : 2348-2273 | P-ISSN : 2454-1222
Available Online at www.ijeecse.com

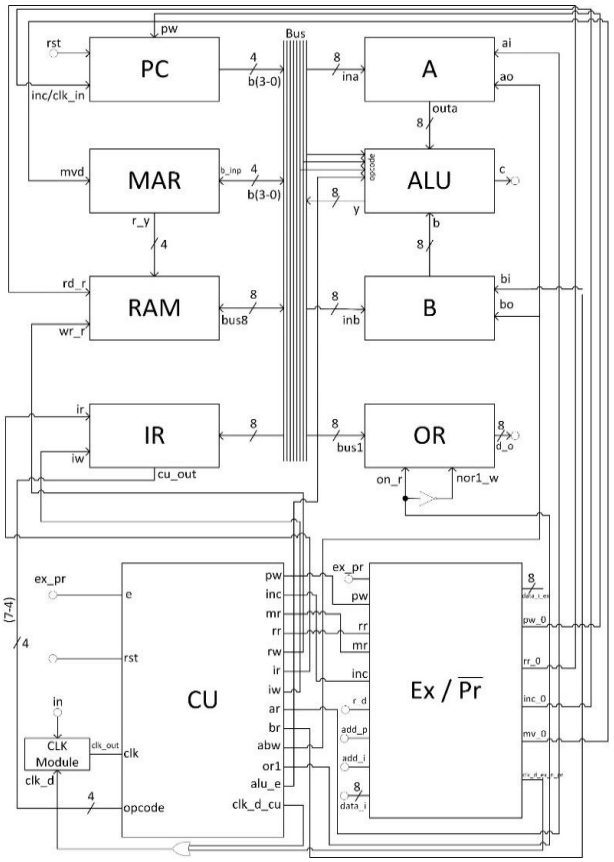

Fig. 12. Final Architecture

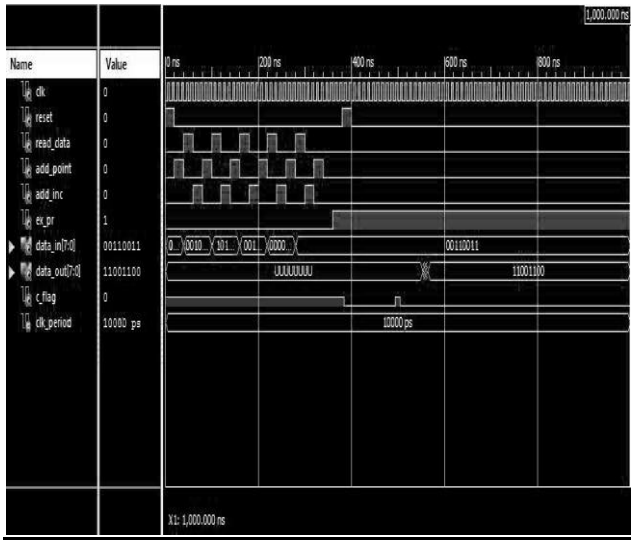

Fig. 13. Test-Bench Waveform to NOT '00110011'

The micro-processor design was implemented on the Spartan 6 Board and various programs were tested.

The results of a test program performing NOT operation on the binary data input "00110011" and the obtained result "11001100" is shown in the figure:

Fig. 14. Input '00110011' (led on-1; off-0)

Fig. 15. Output '11001100' (led on-1; off-0)

VII. Conclusion

The micro-processor design implemented on the Spartan 6 Board worked successfully for the various programs tested.

The micro-processor design used 190 number of slices (5 %).The minimum clock frequency for the design is 375.094MHz(Minimum period limit: 2.666ns). So we can conclude that the processor design implemented is computationally fast and efficient, due to the use of basic gates in the design on comparison to the design implemented in [1], where the processor has a maximum frequency of 95.364 MHz and 132 slices were utilized.

VIII. References

[1] E. Ayeh, K. Agbedanu, Y. Morita, O. Adamo, P. Guturu, "FPGA Implementation of an 8-bit Simple Processor" Region 5 Conference on Apr 2008.

[2] F. Mayer-Lindenberg, V. Beller, "An FPGA-based floating-point processor array supporting a high-precision dot product" Field Programmable Technology, 2006. FPT 2006. IEEE International Conference on Dec. 2006 Pages: 317 – 320.

[3] D. Chiou, H. Sanjeliwala, D. Sunwoo, J. Z. Xu, and N. Patil, "FPGA-based Fast, Cycle-Accurate, Full-System Simulators," in Proceedings of the second Workshop on Architecture Research using FPGA Platforms, held in conjunction with HPCA-12, Austin, TX, Feb. 2006.