Volume 2, Issue 8, August 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

VHDL Implementation of a MIPS RISC Processor

Anjana R[1] & Krunal Gandhi[2]
[1]Ph.D Scholar (ECE)
[2]Lecturer (ECE)
Laxmi Institute of Technology
Gujrat, India

Abstract: The implementation of 32 bit RISC processor with microprocessor without interlocked pipeline stages (MIPS) is presented. It was implemented in VHDL so as to reduce the instruction set present in the programmable memory. As the result the processor will contain the necessary logics for the implementation that requires fewer gates to be synthesized in the programmable matrix and has the capability to increase the speed of the target processor.

Keywords: MIPS, VHDL, Instruction Set, ASIC, FPGA

I. INTRODUCTION

Nowadays, embedded microprocessors are used in a variety of electronic gadgets such as personal computers, cell phones, and robots. These microprocessor includes millons of components which does the memory transfer from one to the other. The microprocessor transforms the logic design into software design which is the reason of the inclusion in the design.

With the ever-increasing size and less cost of FPGA devices, it is now possible to implement a complete system on a single chip s(SOC). Today the largest FPGA from XILINX is the VirtexE. It contains 150 million transistors, resulting in 3 million equivalent gates. SOC is of special interest for Information Appliances such as PDAs and Mobile Phones.

The idea of this project was to create a microprocessor without interlocked pipeline stages (MIPS) as a building block in which can be later included in larger design. This can be implemented to the system where a problem is easy to solve in software but hard to solve with control logic. A Finite state machine is designed in such a way to reduce the computation problems. However, at a high level of complexity it is easier to implement the function in software.

FPGAs are usually slower than ASICs but have the ability to be re-programmed in the field where the errors need to be corrected and upgraded, flexibility, and low-cost. Therefore, they combine many advantages of ASICs and DSPs. The use of hardware description languages (HDLs) allows FPGAs to be more suitable for different types of designs where errors and components failures can be limited.

II. LITERTURE REVIEW

The processor or Central Processing Unit (CPU) is the heart of the computer. It will determine the speed and the capabilities that computer can have. As designers we have a choice between using digital signal processors (DSPs), FPGAs, or application specific integrated circuits (ASICs) in our designing process. In this paper, we implemented an FPGA-based processor. As the FPGA combines the feature of ASIC design in cost reduction and DSP in fast speed, many engineers are now towards FPGA based implementation of the system.

In [2], the author spoke about high performance of embedded systems and real time computing can be obtained through the use of FPGA technology. In [3], the author dealt with the reliability of FPGA platforms in the design of embedded systems. The methods for power reduction, area, performance is presented as the solution. The authors in [4] described a system that can enable FPGAs to generate machine code for various CPUs. The system will perform a conversion of an intermediate to a CPU's native code for applications in virtual machines such as the Java Virtual Machine. In [5], the implementation of a floating point processor array for a high precision dot product is described. The authors mentioned the cost and reconfigurability benefits that can be obtained through the use of FPGA. The authors in [6] used FPGA as a practical experimentation and verification platform for emulation of hardware. The use of FPGA as a design tool can provide not only cost but also performance benefits. In [7], the accuracy, rapidity, (re)configurability, transparency and the cost of FPGA as computer simulator were discussed. In this paper, we present the FPGA implementation of a simple microprocessor without interlocked pipeline stages (MIPS) with limited resources. The processor was implemented on the Xilinx Spartan 3 xc3s200FT256 using ISE foundation 13.2 and VHDL.

III. IMPLEMENTATION

The RISC processor presented in this paper consists of three components as shown in Figure .1, these components are, the Control Unit (CU), the Data path unit, and the memory. The sub-components are: a program counter, an instruction memory, multiplexers (MUX), Adder. The figure below represents the block diagram of the simple processor.

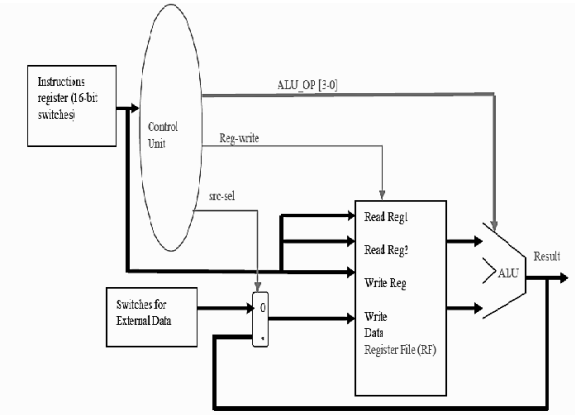

Figure 1: Simple Processor [1]

Design of Control Unit: The control unit is designed as FSM (Finite State Machine) which makes the processor to run at one clock. At the start FSM is in reset condition for initializing the registers and internal memory. Once the enable signal is given, processor allows the instruction to be read from the memory and to decode the read out instruction. The decoding state will select the required instruction from the memory and CU jumps to the appropriate state based on the instruction. Once the read and decode instruction is done, fetching takes place which allows the processor to go to next instruction in the program. Figure .2 shows the state diagram for the control unit.

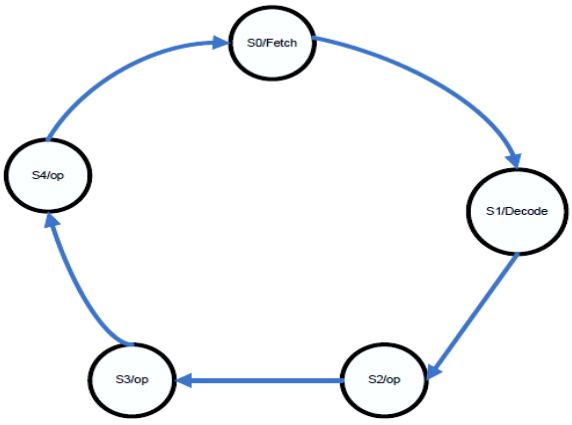

Figure 2: Control FSM [2]

Design of Datapath: The Data Path includes arithmetic and logic operations. The data path is has four section to perform this arithmetic operations. They are (i) Instruction Fetch (ii) Instruction Decode (iii) Execution (iv)Write back. The functionality of each block is tested separately and then combined to test the overall performance.

a) **Instruction Fetch Unit:** The Instruction fetch block contains the following sub units (i) PC – to increment the count and keep track of the instruction that is being fetched (ii) Instruction register – to store the current instruction that is being fetched (iii) the Memory Data Register (MDR) stores the memory address from where the data is fetched. The main function of Instruction fetch is to store the instruction that is currently fetched or decoded from instruction memory and increment the program counter by four.

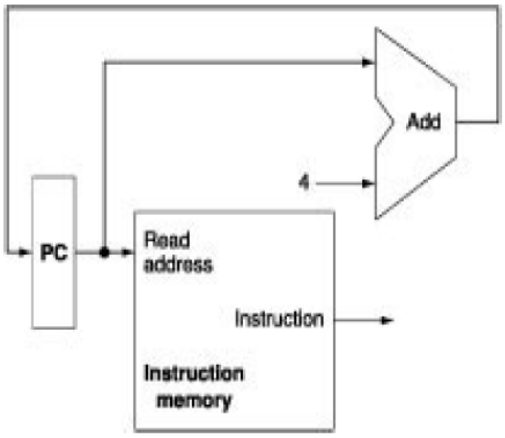

Figure 3: Instruction Fetch Unit [1]

a) **Instruction Decode Unit:** The instruction decode block gets the instruction from instruction register and decodes the operand for load and store operation. This unit is sign extended to 32-bits to read two registers given by 16 bit immediate field. All these values are stored in this unit along with the incremented value of program counter. The opcode fetched form the memory is being decoded in this stage.

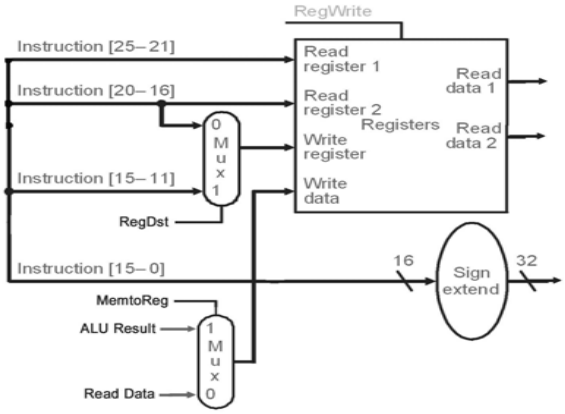

Figure 4: Instruction Decode Unit [1]

Steps:

1. Load the address to PC
2. Load PC content to MAR
3. Update PC contents
4. Load data required for MAR
5. Finally load IR

Transfer notation:

MAR ← [PC]
MDR ← [MAR address];
PC ← [PC] + 1;
IR ← [MDR];

b) **Execution unit:** The Execution unit has ALU as main element and finds the desired result. It also computes branch instruction for the targeted address and provides the same for Memory Write back Block. The multiplexer chooses which operand to be fetched.

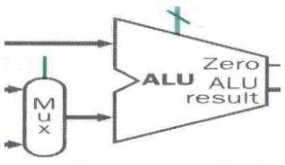

Figure 5: Execution unit [1]

c) **Memory write back unit:** The Memory Write back Block consists of the ALU register and a multiplexer with source signal. The result is fed to either to memory or to the register multiplexer. The result of which is again fed back to the program counter which the next instruction is to be fed.

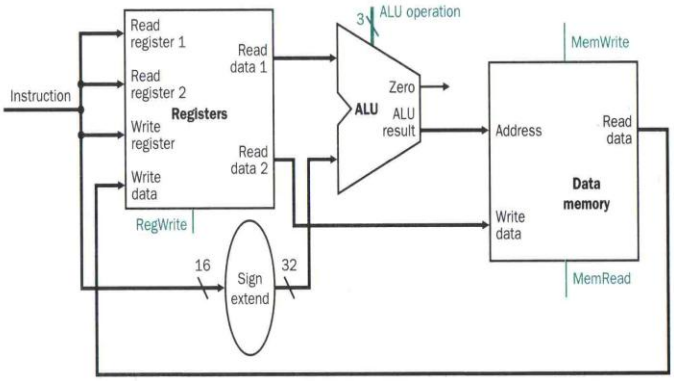

Figure 6: Memory write back unit[1]

Design of memory unit: The data memory unit is accessed only by the lw- and sw- instruction. They compute the memory address by adding a register value to the 16-bit offset field contained in the instruction. The instruction field of load and store instruction is shown in the figure.

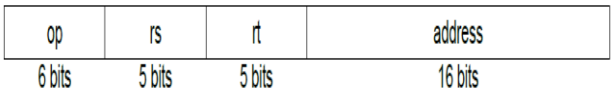

Figure 7: Instruction Format [1]

IV. SIMULATION RESULTS

Implementation is done using XILINX 13.2 RTL schematic and synthesized using modelsim. To ensure the quality, all the modules were designed and tested separately. Once each module is functionally verified they are combined together and tested again. The result shows overall power is reduced by 6%.

ALU:

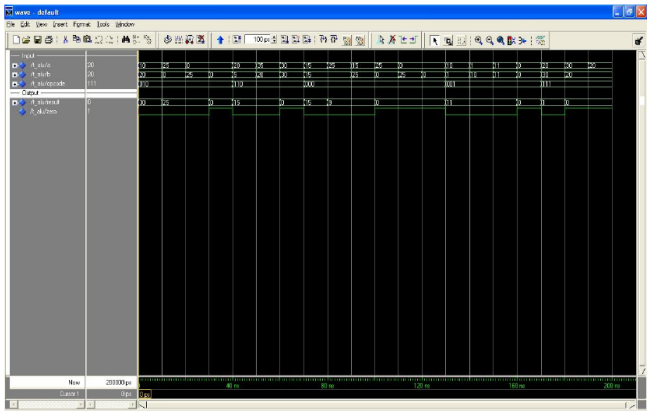

Figure 8: Waveform of ALU

Memory:

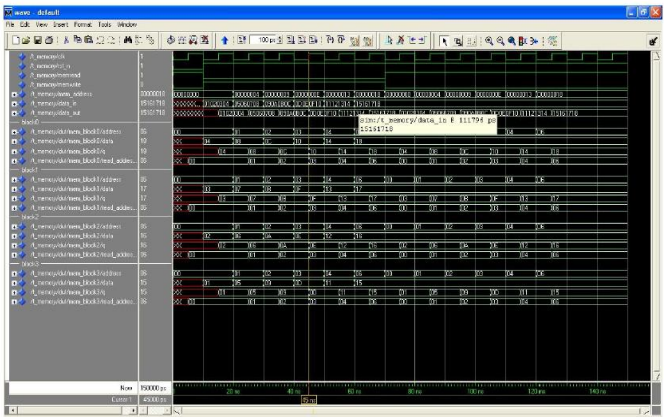

Figure 9: Waveform of Memory

Control:

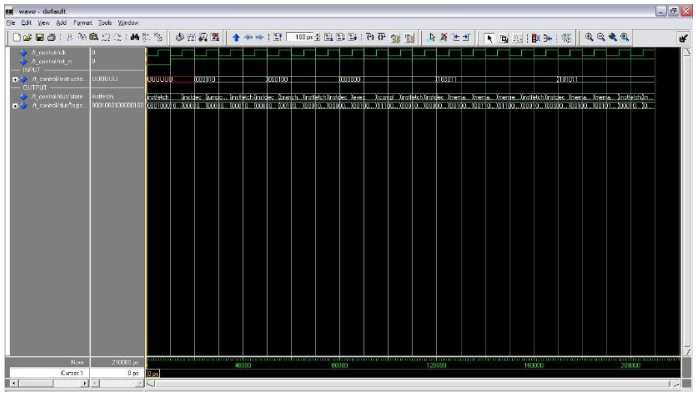

Figure 10: Waveform of control

MIPS and Memory

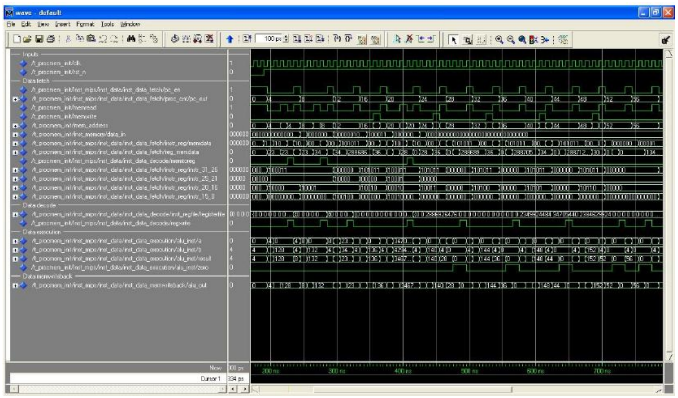

Figure 11: Waveform of MIPS and Memory

V. Conclusion:

The MIPS processor is implemented using VHDL and synthesized using Xilinx ISE. The goal was achieved and the processor is found to work successfully. Future work is to implement the processor using clock gating technique which mainly reduces the dynamic power consumption of the processor.

References:

[1] David A. Patterson, John L. Hennessy: „Computer Organization and Design - The Hardware/SoftwareInterface"Second Editon (1998) Morgan Kaufmann Publisher, Inc.

[2] R. Fryer, FPGA based CPU instrumentation for hard realtime embedded system testing. *SIGBED Rev.* 2, 2 (Apr.2005), 39-42.

[3] P. Yiannacouras, Rose, J., and Steffan, J. G. 2005. Themicro architecture of FPGA-based soft processors. InProceedings of *the 2005 international Conference onCompilers, Architectures and Synthesis For EmbeddedSystems* (San Francisco, California, USA, September 24 -27, 2005). CASES '05. ACM, New York, NY, 202-212.

[4] G. Achery, C. Trinitis, R. Buchty, "CPU-independent assembler in an FPGA," *Field Programmable Logic and Applications, 2005. International Conference*, vol., no.,pp. 519-522, 24-26 Aug. 2005

[5] F. Mayer-Lindenberg, V. Beller, "An FPGA-basedfloating-point processor array supporting a high-precisiondot product" *Field Programmable Technology, 2006.FPT 2006. IEEE International Conference on Dec. 2006*Pages: 317 – 320.

[6] Y Nagaonkar and M. L. Manwaring. "An FPGA-based Experiment Platform for Hardware-Software Codesign and Hardware Emulation" In proceedings of *The 2006World Congress in Computer Science, ComputerEngineering, and Applied Computing* Pages: 169-174.

[7] 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 onArchitecture Research using FPGA Platforms, held in conjunction with HPCA-12, Austin, TX*, Feb. 2006.

Anjana R received her Master's in VLSI Design from Vellore institute of Technology, Vellore and Bachelors from Anna University, Chennai. She has an IT experience of 2 years in eLearning Domain and currently working has Assistant Professor in Electronics and communication Dept. in Laxmi institute of technology, Gujarat. Her area of Interests include Low power VLSI, Domino Logics, Digital circuits.

Krunal Gandhi received his Bachelor's in ECE from VNSGU, Surat. He has the industry experience of two year in 2G and 3G domain and currently working has a lecturer in Laxmi institute of technology, Gujarat. His area of interest include communication systems, wireless communication and in 4G systems.

