A VLSI Design approach for RISC based MIPS architecture

Munmun Ghosal
Research Scholar: Dept. of Electronics Engineering,
G. H. Raisoni College of Engineering, Digdoh Hills,
Nagpur, India 440016.
Email: munmun_ghosal@yahoo.co.in

Dr. A. Y. Deshmukh
Professor: Dept. of Electronics Engineering
G. H. Raisoni College of Engineering, Digdoh Hills,
Nagpur, India 440016.
Email: aydeshmukh@gmail.com

Abstract: **This paper describes the design and analysis of the functional units of RISC based MIPS architecture. The functional units includes the Instruction fetch unit, instruction decode unit, execution unit, data memory and control unit. The functions of these modules are implemented by pipeline without any interlocks and are simulated successfully on Modelsim 6.3f and Xilinx 9.2i. It also attempts to achieve high performance with the use of a simplified instruction set.**

Keywords- MIPS, RISC, Pipelining, Memory.

I. INTRODUCTION

MIPS stands for microprocessor without interlocked pipeline stages. It is a reduced instruction set computer (RISC) instruction set architecture (ISA) [1] and is now a performance leader within the embedded industry. MIPS is a load/store architecture in which all operations are performed on operands held in the processor registers.

RISC CPU has advantages such as faster speed, simplified structure & easier implementation. The decision to include a microprocessor in a design is that it transforms the design effort from a logic design into a software design. With the ever-increasing size and reductions in cost of FPGA devices, it is now possible to implement a complete system on one device, a System-On-Chip (SOC).

In this paper the design of various functional units of RISC based MIPS processor using VHDL has been presented. The goal of this work was to evaluate the performance of various units of the MIPS processor.

II. SALIENT FEATURES OF MIPS ARCHITECTURE

A. Instructions Set Architecture

The MIPS Architecture defines thirty-two [6]; 32-bit general purpose registers (GPRs).

All instructions of MIPS microprocessor are 32 bit and are available in three formats: R-type, I-type and J-type [4].MIPS instructions are three address operations taking two sources and one destination.

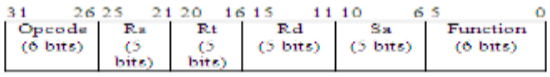

Figure 1: MIPS (R-Type) CPU Instruction format

Figure 1 explains the format of R-Type CPU instruction format which is meant for performing arithmetic operations. It allows a range of register to register operations.

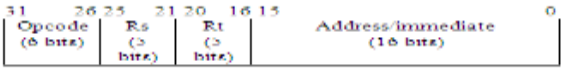

Figure 2: MIPS Immediate (I-Type) CPU Instruction format

Figure.2 explains the I-type instructions format that allows a 16 bit immediate to replace one of the operands and is also used for memory accesses and for conditional branches.

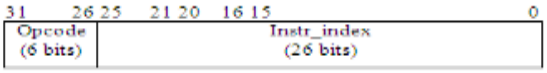

Figure 3: MIPS Jump (J-Type) CPU Instruction format

Figure.3 explains the J-type format with a 26 bit immediate field. The only instruction to use this format is a jump which places the value in the bottom 26 bits of the program counter.

B. Register

A MIPS microprocessor has 32 addressable registers. The registers are preceded by $ in assembly language instruction. Two formats for addressing are used i.e., using register numbers ($0 through $31) or using equivalent names ($t1, $sp).Special registers Lo and Hi used to store result of multiplication and division. The stack of MIPS grows from high memory to low memory [3].

C. Memory

Memory access instructions are included in the I-type format. The source register (RS) is added to the immediate to create an effective address which is used to reference the memory. The second register (RT) is either used as the destination in a memory load or as a source in a memory store. The memory is byte addressed but is 32 bit wide so all word loads and stores have to be word aligned. Half word accesses have to be aligned to half word boundaries. To help with unaligned loads and stores there are two more memory access instructions. Load Word Left (LWL) and Load Word Right (LWR) in combination allow word loads from unaligned addresses.

D. Pipeline Interlocking

In the MIPS microprocessor this means that some instructions have an implicit delay before their effect takes place [1].The general philosophy is to construct the hardware as simply as possible and, if a result is not ready for use in the next instruction then not to stop the whole processor but use the software to insert instructions into the space. The two main delays in the MIPS microprocessor are branch shadows and memory load delays.
Pipelining is a standard feature in RISC processors which is used to improve both clock speed and overall performance. It allows a processor to work on different steps of the instruction at the same time, thus more instruction can be executed in a shorter period of time.

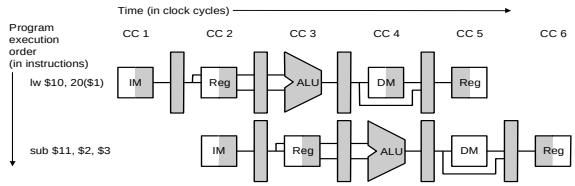

Figure 4: Five Stage Pipelining

E. Conditions

There are no condition flags but instead all branches are conditional on the values of the registers in the main register bank. Each conditional branch instruction specifies two registers (RS and RT) to be fetched and tested. A branch is conditional on the results of two tests. The first is compare the two registers together to test whether they are equal (RS=RT). The other test is simply to look at the sign value (bit 31) of the first register (RS<0). By choosing the second register to be R0 (RT=0) it becomes possible to test RS for less than greater or equal to zero or any combination of the three. For an unconditional branch the Branch if Greater or Equal to Zero instruction (BGEZ) is used with R0 as an operand. This condition will always be true.

III. MIPS FUNCTIONAL UNITS

A. Instruction Fetch Unit

The function of the instruction fetch unit is to obtain an instruction from the instruction memory using the current value of the PC and increment the PC value for the next instruction. The block diagram for the Instruction fetch unit is shown in Figure 5.

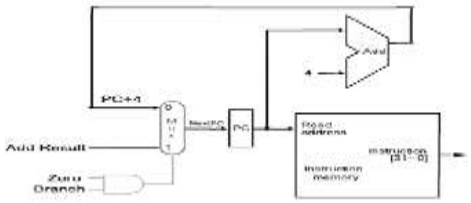

Figure 5: Instruction Fetch unit

B. Instruction Decode Unit

The main function of the instruction decode unit is to use the 32-bit instruction provided from the previous instruction fetch unit to index the register file and obtain the register data values. The block diagram for the Instruction decode unit is shown in Figure 6.

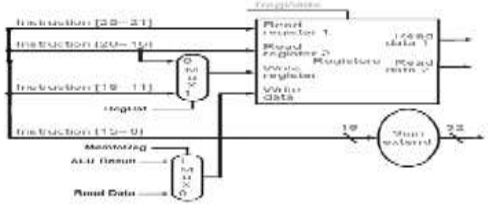

Figure 6: MIPS Instruction Decode Unit

C. The Control Unit

The control unit examines the instruction opcode bits [31 – 26] and decodes the instruction to generate nine control signals to be used in the additional modules. The block diagram for the Control unit is shown in Figure 7.

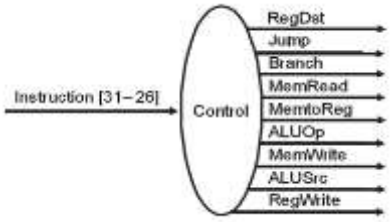

Figure 7: MIPS Control Unit

D. Execution Unit

The execution unit contains the arithmetic logic unit (ALU) .The branch address is calculated by adding the PC+4 to the sign extended immediate field shifted left 2 bits by a separate adder. The logic elements include a MUX, an adder, the ALU and the ALU control. The block diagram for the Execution unit is shown in Figure 8.

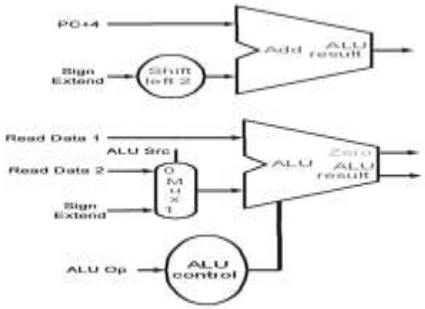

Figure 8: MIPS Execution Unit:

E. Data Memory Unit

It is only accessed by the load and store instructions. The load instruction asserts the MemRead signal and uses the ALU Result value as an address to index the data memory. The read output data is then subsequently written into the register file. A store instruction asserts the MemWrite signal and writes the data value previously read from a register into the computed memory address.

V. SYNTHESIS AND SIMULATION RESULTS

A. Instruction Fetch Unit

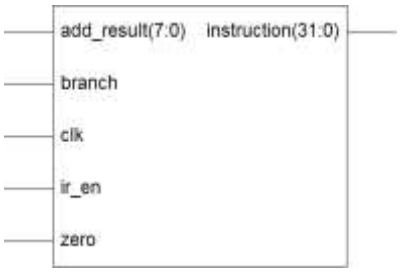

Figure 9: Block diagram of MIPS Instruction fetch Unit obtained by Synthesis using Xilinx 9.1.

Figure 9 explains the block diagram of Instruction fetch unit showing the input and output ports. The inputs to the unit are the instruction from the instruction memory given as add_result, zero and branch. The unit is also fed by the global clock and enable inputs. The output of the unit is a 32 bit instruction format which is obtained only when the enable input is high.

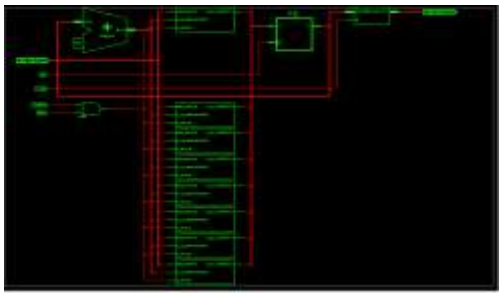

Figure 10: RTL Schematic of MIPS Instruction fetch Unit obtained by Synthesis using Xilinx 9.2i.

Figure 10 explains the RTL Schematic of MIPS Instruction fetch Unit which gives a detailed structure of the unit consisting of adder, AND gate, Program counter and Instruction memory. The Instruction memory consists of a series of latches which holds the instruction.

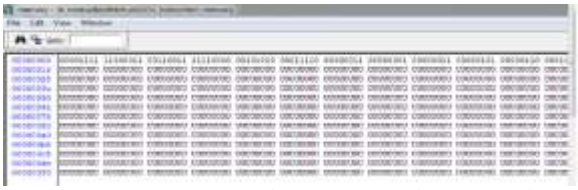

Figure 11: Memory assigned for the analysis of Instruction fetch Unit in Modelsim 6.3f.

Figure 11 explains the status of the memory obtained after the simulation of the Instruction Fetch Unit in which the instruction is written into the Instruction Memory.

Figure 12: Waveforms of MIPS Instruction fetch Unit obtained by Simulation using Modelsim 6.3f.

Figure 12 explains the simulation results of the Instruction fetch Unit. The waveforms show the various possible combinations of inputs and its corresponding outputs.

B. Instruction Decode Unit

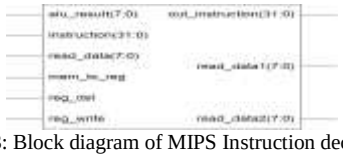

Figure 13: Block diagram of MIPS Instruction decode Unit obtained by Synthesis using Xilinx 9.1.

Figure 13 explains the block diagram of Instruction decode unit showing the input and output ports. The inputs to the unit are the outputs from the instruction fetch unit. It decodes the instruction obtained from the previous instruction fetch unit.

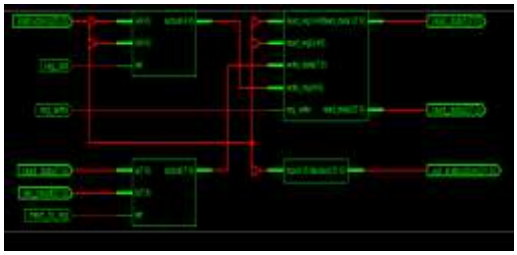

Figure 14: RTL Schematic of MIPS Instruction Decode Unit obtained by Synthesis using Xilinx 9.2i.

Figure 14 explains the RTL Schematic of MIPS Instruction Decode Unit which gives a detailed structure of the unit consisting of registers, MUX and sign extend unit.

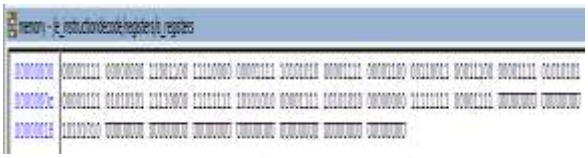

Figure 15: Memory assigned for the analysis of Instruction decode Unit in Modelsim 6.3f

Figure 15 explains the contents of the memory obtained after the simulation of the Instruction Decode Unit.

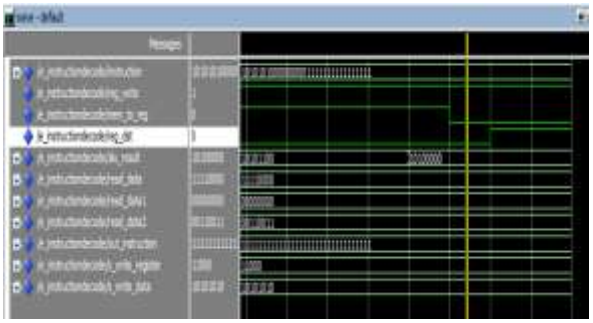

Figure 16: Waveforms of MIPS Instruction Decode Unit obtained by Simulation using Modelsim 6.3f.

Figure 16 explains the simulation results of the Instruction decode Unit. The waveforms show the various possible combinations of inputs and its corresponding outputs whereby depending on the state of MemtoReg signal, the ALUresult or the readData is written into the registers. The RegDat signal determines if instruction bits [20-16] or [15-11] is provided to the write registers.

C. Control Unit

Figure 17: Block diagram of MIPS Control Unit obtained by Synthesis using Xilinx 9.1.

Figure 17 explains the block diagram of control unit showing the input as a 32 bit instruction and outputs as various control signals which are used to execute a given instruction and hence the given program.

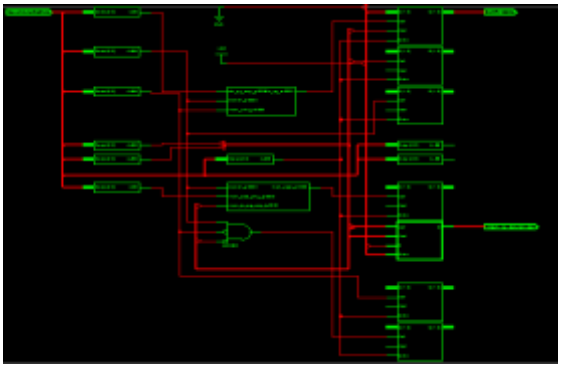

Figure 18: RTL Schematic of MIPS Control Unit obtained by Synthesis using Xilinx 9.2i.

Figure 18 explains the RTL Schematic of MIPS Control Unit which gives a detailed structure responsible for the generation on control signals.

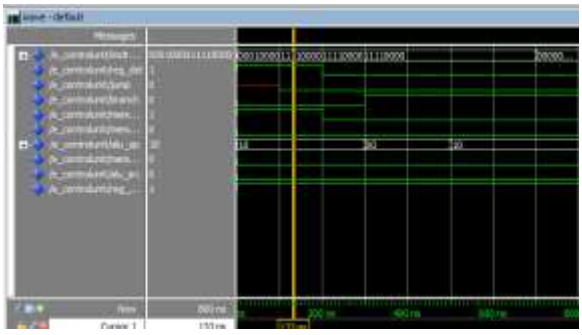

Figure 19: Waveforms of MIPS Control Unit obtained by Simulation using Modelsim 6.3f.

Figure 19 explains the simulation results of the Control Unit which shows various possible values of the given Instruction and the corresponding values of the control signals which are alu_op, alu_src, branch, jump, mem_read, mem_to_reg, mem_write, reg_dst and reg_write .

Execution Unit

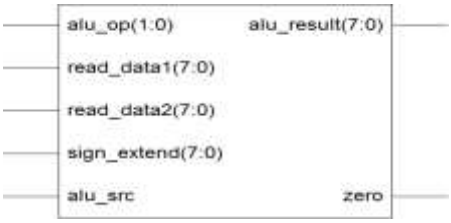

Figure 20: Block diagram of MIPS Execution Unit obtained by Synthesis using Xilinx 9.1.

Figure 20 explains the block diagram of Execution unit. The inputs to the unit are the data from decode unit and the ALU operation which is to be performed. The outputs consist of the result obtained on the desired operation and a zero signal which indicates if the result is zero or not.

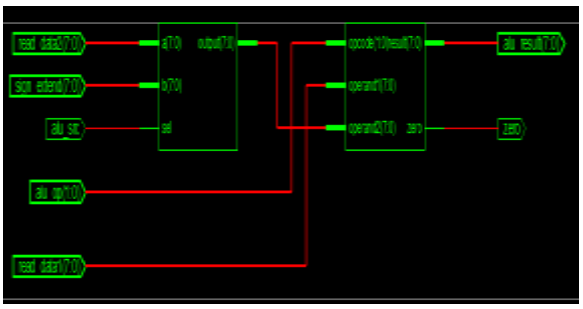

Figure 21: RTL Schematic of MIPS Execution Unit obtained by Synthesis using Xilinx 9.2i.

Figure 21 explains the RTL Schematic of MIPS Execution Unit which gives a detailed structure consisting of Arithmetic and logical Unit together with multiplexers, shift registers and sign extend unit.

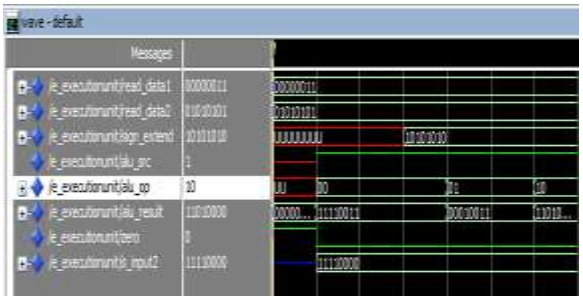

Figure 22: Waveforms of MIPS Execution Unit obtained by Simulation using Modelsim 6.3f.

Figure 22 explains the simulation result of the Execution Unit whereby the output alu_result is obtained with various possible combinations of inputs i.e. read_data1 and read_data2 and the opcode which specifies the operation to be performed.

CONCLUSION

A complete realistic, parameterized, synthesizable, modular, single clock and multiple clock multicore architecture of RISC based MIPS is studied. MIPS is a fully pipelined architecture having an efficient instruction scheduling. The functionality of the instruction fetch unit, Instruction decode unit, Control unit and the execution unit has been synthesized and verified using Modelsim 6.3f and Xilinx 9.2i.

REFERENCES

[1] Charles Brej.,"A MIPS R3000 microprocessor on an FPGA", 13 February 2002.

[2] MIPS Technologies, Inc. MIPS32™ Architecture For Programmers Volume II: The MIPS32™ Instruction Set, June 9, 2003.

[3] MIPS Technologies, Inc. MIPS32^{R} Architecture For Programmers Volume I:Introduction to the MIPS32 Architecture, June 25, 2008.

[4] GauthamP,ParthasarathyR,KarthiBalasubramanian,Department of Electronics and Communication, Amrita School of Engineering,Amrita Vishwavidyapeetham. "Low-Power Pipelined MIPS Processor Design", ISIC 2009.

[5] Zulkifli, Yudhanto, Soetharyo and Adiono, "Reduced Stall MIPS Architecture using Pre-Fetching Accelerator", International Conference on Electrical Engineering and Informatic5-7 August 2009, Selangor, Malaysia.

[6] Kui YI Department of Computer Science and Information Engineer, WuHan Polytechnic University Wuhan, HuBei Province 430023,China,Yue-Hua DING Department of Computer Science and Information Engineer, WuHan Polytechnic University Wuhan, HuBei Province 430023, China. , "32-bit RISC CPU Based on MIPS Instruction Fetch Module Design" 2009 International Joint Conference on Artificial Intelligence.

[7] Tyson, Aisar Labibi Romas, Rd. Siti Intan P, and Trio Adiono, Ph. D, Bandung Institute of Technology, Jl. Ganesha 10 Bandung – Indonesia, "A Pipelined Double-Issue MIPS Based Processor Architecture", 2009 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2009) December 7-9, 2009.

[8] Mamun Bin Ibne Reaz, MEEE, Md. Shabiul Islam, MEEE, Mohd. **S.** Sulaiman, "**A** Single Clock Cycle MIPS RISC Processor Design using VHDL", MEEE Faculty of Engineering, Multimedia University, 63 100 Cybejaya, Selangor, Malaysia, ICSE2002 Proc. 2002, Penang, Malaysia.

[9] Xizhi Li The CKC honored School of Zhejiang University, P.R. China,Tiecai Li Department of Electrical Engineering Harbin Institute of Technology, "ECOMIPS: An Economic MIPS CPU Design on FPGA", Proceedings of the 4th IEEE International Workshop on System-on-Chip for Real-Time Applications (IWSOC'04)0-7695-2182-7/04 IEEE.

[10] Asghar Bashteen, Ivy Lui, Jill Mullan."A Superpipeline Approach to the MIPS Architecture",MIPS Computer Systems,950 DeGuigne Drive,Sunnyvale, CA 94086.

[11] Smith Douglas J. 1996. HDL Chip Design: a Practical Guide for Designing, Synthesizing and Simulating ASICs and FPGAs using VHDL or Verilog. Madison: DoonPublications.