Design of 32 Bit MIPS RISC Processor Based on Soc

Pranjali S. Kelgaonkar
Department of Electronics and Communication Engineering
MIT, Aurangabad,Maharashtra,India

Prof. Shilpa Kodgire
Department of Electronics and Communication Engineering
MIT, Aurangabad,Maharashtra,India

Abstract-This paper concerned with the design and implementation of a 32bit Reduced Instruction Set Computer (RISC) processor on a Field Programmable Gate Arrays (FPGAs). The processor has been designed with Verilog HDL, synthesized using Xilinx ISE 14.2 and simulated using QuestaSim 6.4C simulator, and then will implement on ZedBoard_HW_UG_v1_9.

The project involves design of a 32bit RISC processor and simulating it. A Reduced Instruction Set compiler (RISC) is a microprocessor that had been designed to perform a small set of instructions, with the aim of increasing the overall speed of the processor . We use pipeline design process to reduce the execution time of instruction successfully, which involves instruction fetch (IF), instruction decoder (ID), execution (EXE), data memory (MEM), write back (WB) modules of 32-bit RISC processor.

Keywords – RISC, MIPS, Simulation Synthesis ,Instruction Set,MODELSIM.s,SOC.

I. INTRODUCTION

The MIPS processor, designed in 1984 by researchers at Stanford University, is a RISC (Reduced Instruction Set Computer) processor. Compared with their CISC (Complex Instruction Set Computer) counterparts(such as the Intel Pentium processors), RISC processors typically support fewer and much simpler instructions. RISC processor can be made much faster than a CISC processor because of its simpler design. These days, it is generally accepted that RISC processors are more efficient than CISC processors; and even the only popular CISC processor that is still around (Intel Pentium) internally translates the CISC instructions into RISC instructions before they are executed [1].

RISC processors typically have a load store architecture. This means there are two instructions for accessing memory: a load instruction to load data from memory and a store (s) instruction to write data to memory. It also means that none of the other instructions can access memory directly.

Processors are much faster than memories. For example, a processor clocked at 100 MHz would like to access memory in 10 nanoseconds, the period of its 100 MHz clock. Unfortunately, the memory interfaced to the processor might require 60 nanoseconds for an access. So, the processor ends up waiting during each memory access, wasting execution cycles.

II. RELATED WORK

Harvard architecture is used which has distinct program memory space and data memory space. Low power consumption helps to reduce the heat dissipation, lengthen battery life and increase device reliability. To minimize the power of RISC Core, clock gating technique is used in the architectural level which is an efficient low power technique.[1].In previous work VHDL language is used to implement the 32bit processor[2]. A 32 – bit RISC (Reduced Instruction Set Computer) processor using XILINX VIRTEX4 Tool for embedded and portable applications.[3]. With a single instruction , more executions scheme implemented to achieve high throughput.[4]. RISC MIPS Processor technique sends the machine code to the instruction memory of the soft-core from the software tool through UART.[5].Another more important advantage is that RISC chips [1] [5] require fewer transistors for cheaper design and produce.

III. MIPS PROCESSOR ARCHITECTURE

A. *MIPS Processor Architecture--*

The MIPS instruction set architecture (ISA) is a RISC based microprocessor architecture that was developed by MIPS Computer Systems Inc. in the early 1980s. MIPS is now an industry standard and the performance leader within the embedded industry. Their designs can be found in Canon digital cameras, Windows CE devices, Cisco Routers, Sony Play Station 2 game consoles, and many more products used in our everyday lives.

By the late 1990s it was estimated that all RISC chips produced was a MIPS-based design. Architecture of MIPS RISC microprocessor includes, fix-length straightforward decoded instruction format, memory accesses limited to load and store instructions, hardwired control unit, a large general purpose register file, and all operations are done within the registers of the processor.

The 32 bit Risc MIPS processor has 5 stages:

1. Instruction fetch (IF),
2. Instruction decoder (ID),
3. Execution (EXE),
4. Data memory (MEM),
5. Write back (WB)

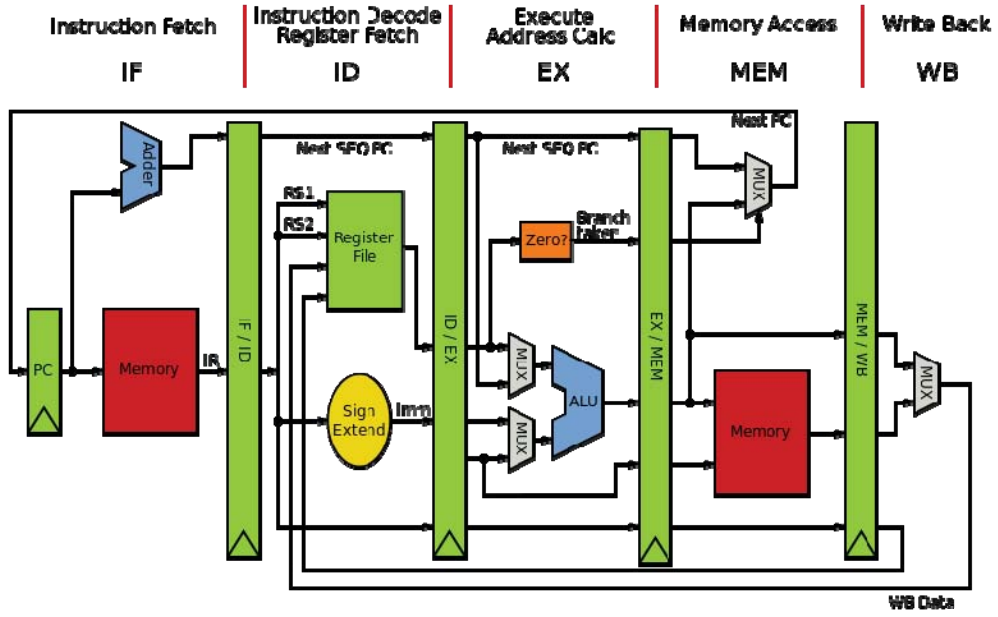

Figure1. Architecture of MIPS RISC Processor

The MIPS single-cycle processor performs the tasks of instruction fetch, instruction decode, execution, memory access and write-back all in one clock cycle. First the PC value is used as an address to index the instruction memory which supplies a 32-bit value of the next instruction to be executed.

1. IF : The Instruction Fetch stage fetches the next instruction from memory using the address in the PC (Program Counter) register and stores this instruction in the IR (Instruction Register)
2. ID : The Instruction Decode stage decodes the instruction in the IR, calculates the next PC, and reads any operands required from the register file.
3. EX : The Execute stage "executes" the instruction. In fact, all ALU operations are done in this stage. (The ALU is the Arithmetic and Logic Unit and performs operations such as addition, subtraction, shifts left and right, etc.)
4. MA : The Memory Access stage performs any memory access required by the current instruction, So, for loads, it would load an operand from memory. For stores, it would store an operand into memory. For all other instructions, it would do nothing.

5. WB : For instructions that have a result (a destination register), the Write Back writes this result back to the register file. This includes nearly all instructions, except nops (a nop or no-operation instruction simply does nothing) and s (stores).

The instructions opcode field bits [31-26] are sent to a control unit to determine the type of instruction to execute. The type of instruction then determines which control signals are to be asserted and what function the ALU is to perform, thus decoding the instruction. The instruction register address fields $RS bits [25 - 21], $RT bits [20 - 16], and $RD bits [15-11] are used to address the register file. The register file supports two independent register reads and one register write in one clock cycle. The register file reads in the requested addresses and outputs the data values contained in these registers. These data values can then be operated on by the ALU whose operation is determined by the control unit to either compute a memory address (e.g. load or store), compute an arithmetic result (e.g. add, and or slt), or perform a compare (e.g. branch). If the instruction decoded is arithmetic, the ALU result must be written to a register. If the instruction decoded is a load or a store, the ALU result is then used to address the data memory. The final step writes the ALU result or memory value back to the register file.

B. MIPS Instruction Format—

1. ADD <RS>, <RT>, <RD> - This instruction reads RT and RD, adds their contents and puts it into RS. RS, RT, RD are the register addresses from 0 through 31 (since we have 32 registers).Each register field is 5 bits long.
2. SUB <RS>, <RT>, <RD> - This instruction reads RT and RD, computes RT – RD and stored it in RS.
3. LW <RS>, IDX(<RT>) - This instruction reads the contents of RT, computes RT + IDX. Fetches the contents of the memory location (RT + IDX) and stores it into the register RS. IDX is 5 bits wide.
4. SW <RS>, IDX(<RT>) - This instructions computes RT + IDX and stores the contents of RS in the memory location (RT+IDX).
5. BEQ <RS>, <RT>, BRANCH - Checks if RS = RT. If true branches to PC + BRANCH and continues execution from there.
6. LI <RS>, IMM - Loads IMM into the register RS.

C. Software Implamentation—

Verilog HDL has evolved as a standard hardware description language. A hardware descriptive language is a language used to describe a digital system. It means that by using HDL one can describe any hardware at any level. Verilog HDL is one of the two most common Hardware Description Languages (HDL) used by integrated circuit (IC) designers.

For this Project, Verilog is used as an input for synthesis programs which will generate a gate-level description for the circuit. To synthesize the circuit, it need to write a test bench for getting different output taking out at different input and different interval. The simulator which is used for the language is Xilinx ISE 14.2 and Questasim 6.4C.

In this project Pipelining is used. Pipeline is a standard feature in RISC processors, a technique used to improve both clock speed and overall performance. Pipelining 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. For example in the VHDL MIPS single-cycle implementation above, the datapath is divided into different modules, where each module must wait for the previous one to finish before it can execute, thereby completing one instruction in one long clock cycle. When the MIPS processor is pipelined, during a single clock cycle each one of those modules or stages is in use at exactly the same time executing on different instructions in parallel. The MIPS pipelined processor involves five steps; the division of an instruction into five stages implies a five-stage pipeline. The key to pipelining the single-cycle implementation of the MIPS processor is the introduction of pipeline registers that are used to separate the datapath into the five sections IF, ID, EX, MEM and WB. Pipeline registers are used to store the values used by an instruction as it proceeds through the subsequent stages. The MIPS pipelined registers are labeled according to the stages they separate. (e.g. IF/ID, ID/EX, EX/MEM, MEM/WB)

II. PERFORMANCE ANALYSIS

1. Ouptut of IF stage

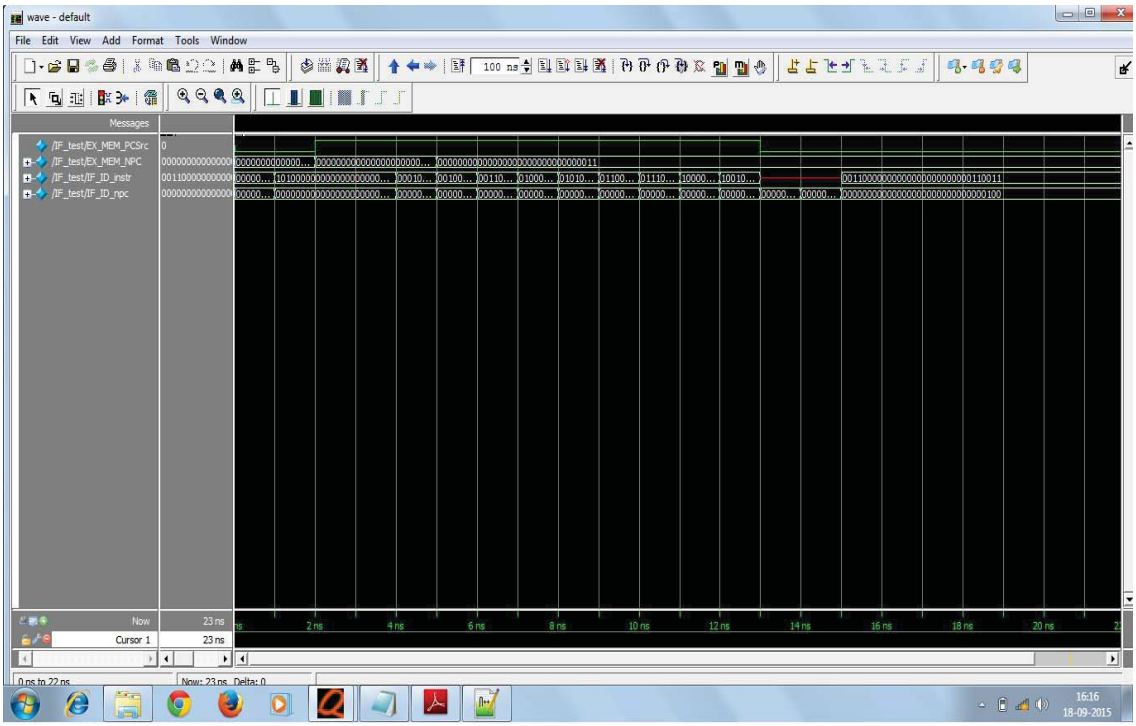

Figure 2. Output of IF stage

2.Output of 32bit processor

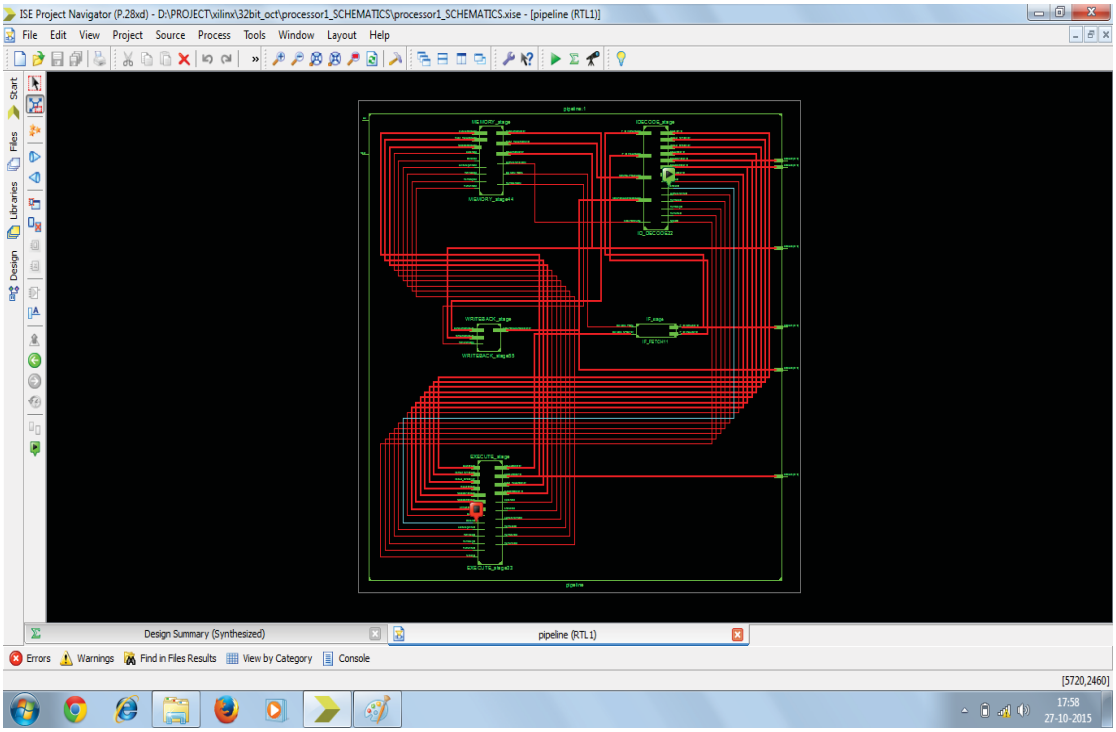

Figure 3. RTL Schematics of 32bit processor

IV. FUTURE DEVELOPMENT

A Field-programmable Gate Array (FPGA) is an integrated circuit designed to be configured by the customer or designer after manufacturing—hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) . FPGAs can be used to implement any logical function that an ASIC could perform.

Now my focus is on the implementation of the verilog code using FPGAs simulated, synthesized and implemented using ZedBoard_HW_UG_v1_9.

IV. CONCLUSION

32 bit RISC Processor has been designed and simulated on Xilinx 14.2. The design has been achieved using Verilog HDL and simulated with Questasim 6.4C.Most of the goal is achieved and simulation shows that the processor is working perfectly. Future work will be added by increasing the number of instructions and will be implemented on ZedBoard_HW_UG_v1_9,with less clock cycles per instructions.

REFERENCES

[1] Preetam Bhosale, Hari Krishna Murti, "FPGA Implementation of Low Power Pipelined 32-bit RISC Processor" *IJITEE Volume-1,Issue-3, August 2012.*

[2] N.Alekya ,P.Ganesh Kumar, " Design of 32-Bit Risc CPU Based on MIPS" JGRCS Volume 2,No 9,September 2011.

[3] Galani Tina G.,Riya Saini and R.D.Daruwala," Design and Implementation of 32 – bit RISC Processor using Xilinx," *IJITEE Volume-5,Issue-1, August 2013.*

[4] J.Poornima, G.V.Ganesh, M.jyothi,M.Sahithi,A.jhansi RaniB. Raghu Kanth, " Design and Implementation of Pipelined 32-bit Advanced RISC Processor for Various D.S.P Applications", (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 3 (1) , 2012,3208-3213.

[5] Marri Mounika,Aleti Shankar , " Design & Implementation Of 32-Bit Risc (MIPS) Processor", *International Journal of Engineering Trends and Technology (IJETT) – Volume 4 Issue 10 - Oct 2013*.

[6] Navneet kaur, Adesh Kumar, Lipika Gupta," VHDL Design and Synthesis of 64 bit RISC Processor System on Chip (SoC) ", *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 3, Issue 5 (Nov. – Dec. 2013), PP 31-42 e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197 www.iosrjournals.org*.

[7] R. Uma," Design and Performance Analysis of 8-bit RISC Processor using Xilinx Tool", International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 2, Mar-Apr 2012, pp.053-058.

[8] Sagar Bhavsar, Akhil Rao, Abhishek Sen, Rohan Joshi ," A 16-bit MIPS Based Instruction Set Architecture for RISC Processor", International Journal of Scientific and Research Publications, Volume 3, Issue 4, April 2013.

[9] Prof. Bruce Jacob," The RiSC-16 Instruction-Set Architecture," ENEE 446: Digital Computer Design, Fall 2000.

[10] Anand Nandakumar Shardul, "16-Bit RISC Processor Design for Convolution Application", International Journal of Advancements in Research & Technology, Volume 2, Issue 9, September-2013.

[11] Digital Design – Morris Mano 5E.

[12] Computer Organization and design – David A. Patterson,John L.Hennessy.