International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 02 Issue: 07 | Oct-2015 www.irjet.net p-ISSN: 2395-0072

OVERVIEW OF RISC PROCESSOR USING VHDL

Sangita Mohite[1], Aarti Tirmare[2], Priyadarshani Mali[3]

[123] *Assistant Professor, Electronics & Telecommunication, BVCOEK, Kolhapur, Maharashtra, India.*

Abstract - ***This paper describes a design methodology of a single clock cycle RISC Processor using VHDL to ease the description, verification, simulation and hardware realization. Known for their flexibility, Field Programmable Gate Arrays (FPGA) are widely used for ASIC emulation, as a solution for applications with high volatility. FPGAs facilitate quick time to market, and their incredible power of re-programmability often makes them the heart of a system. This paper will present the design of a Reduced Instruction Set Computer (RISC) system described using VHDL and the results of researching the implementation of this system in an FPGA. The RISC processor is separated into five stages: instruction fetch, instruction decode, execution, data memory and write back. The control unit controls the operations performed in these stages. The top-level module connects all the stages into a higher level. Once detecting the particular approaches for input, output, main block and different modules, the VHDL descriptions are run through a VHDL simulator, followed by the timing analysis for the validation, functionality and performance of the designated design that demonstrate the effectiveness of the design. This RISC is a 8-bit processor with high general-purpose register (GPR) orthogonality and communicates to peripheral devices via a serial bus.***

Key Words: *Arithmetic Logic Unit (ALU), Central Processing Unit (CPU), Control Unit (CU),Field Programmable Gate Array (FPGA),Program Counter (PC), Reduced Instruction Set Computer (RISC).*

1. INTRODUCTION

RISC stands for "reduced instruction set computer." It is a type of microprocessor architecture that utilizes a small, highly optimized set of instructions rather than more specialized set of instructions often found in other types of architecture. Reduced Instruction Set Computer (RISC) systems are a dramatic departure from the historical trend in Central Processing Unit (CPU) architecture. An analysis of the RISC architecture brings into focus many important issues in computer organization and architecture. Reduced Instruction Set Computer (RISC) is a design philosophy that becomes mainstream in the last few years, as the quest for raw speed has dominated the highly competitive computer industry. There is a desire to increase the processor's speed and simplify the hardware for reasons of cost. RISC design resulted in computers that execute instructions faster than other computers built of the same technology [2].

There are three basic levels: 1) All instructions will be executed in a single cycle.2) Memory will only be accessed via load and store instruction.3) All execution units will be hardwired with no micro-coding. Instruction set is the "hardware language" in which software tells the processor what to do. The vacated area of chip can be used in a ways that accelerated the performance of more common instruction. It becomes easier to optimize the design.

2. MATERIAL AND METHODS

Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here. Paragraph comes content here.

2.1 RISC Processor data path

The basic data path of a RISC processor is shown below in Figure 1. The Instruction Decoder loads the instruction pointed to by the program counter (PC) from processor memory. The Instruction Decoder then generates the appropriate control signals for the Execute unit, which performs the desired function (arithmetic, logic, etc.) on the data. The Write back unit then updates the memory with any new values.

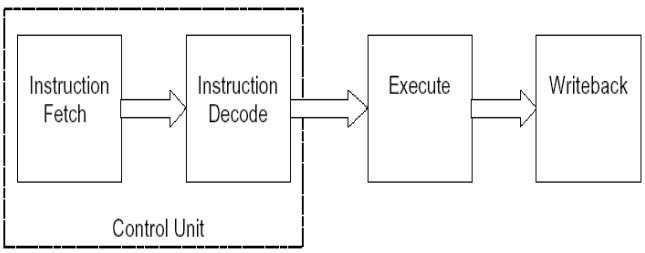

Figure 1 RISC processor data path

2.2 System architecture

RISC processor consist three components, these are CU, Data path and ROM.

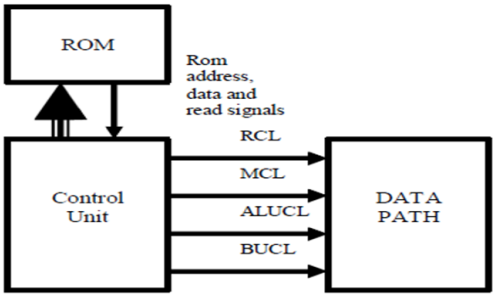

Figure 2. System architecture

The control unit design is based on FSM. We design it in a way that allows each state to run at one clock cycle, the first stage is reset which is initializes the CPU, Internal registers and variables. The machine goes to the reset state by enabling the reset signal for a certain number of clocks.

Following the reset state would be the instruction fetching and decoding state which will enable the appropriate signals for reading instructions data from the ROM then decoding the parts of the instructions. The decoding stage will also select next stage depending on the instruction, since every instruction has its own set of state the control unit will jump to the correct state based on instruction given. After all states of a running instruction are finished, the last one will return to the fetch state which will allows us to process the next instruction in program.

2.3 System level Block diagram

A more detailed look at the layout of the RISC processor is shown below in Figure 3. The process starts out at the branch selector, which loads the program counter with either the next sequential address or the address of a program branch depending on the value of the branch select signal. In the case of an interrupt, the branch address input would contain the address of the appropriate interrupt handler. The instruction is then fetched from program memory and sent into the instruction decoder, which loads the operand and function select buses and generates control signals for the rest of the processor. The execute stage performs an operation or interacts with the data memory. After the execute stage the result is written to the processor registers if applicable.

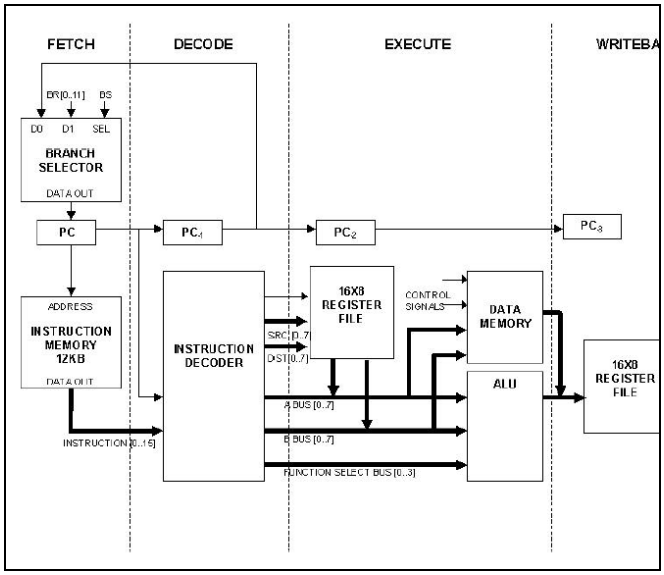

Figure 3. System level Block diagram

2.4 Sub-System Block diagram

The focus of this project is the control unit of the processor that is the section responsible for the Fetch and Decode stages. A closer look at the control unit is shown below in Figure 4.

A) Instruction Multiplexer Inputs*:*

1) BR [0...11] (Branch Address)*:* Contains the target address (PC + offset) for a branch. In the case of a program branch, this address will point to the appropriate instruction.
2) PC+1 (Incremented Program Counter): Unless a program branch occurs the next instruction executed is the sequential instruction in program memory
3) BS (Branch Select): If this bit is set then the instruction address Corresponding to a program branch is selected, otherwise the sequential (PC+1) instruction is selected.

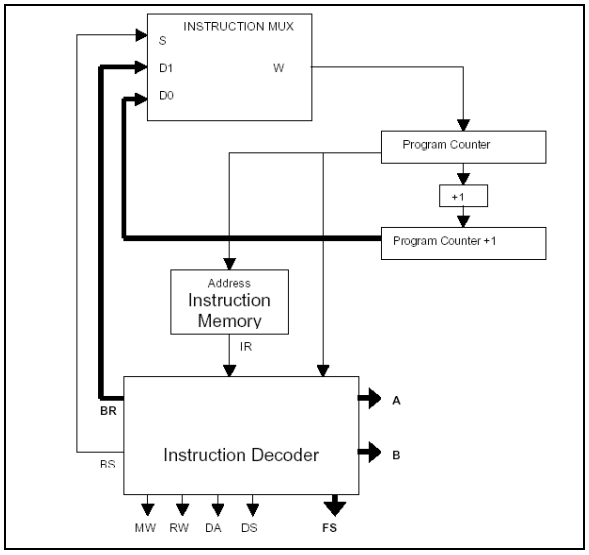

Figure 4. Sub-System Block diagram

B) Instruction Multiplexer Outputs:

1) W (Instruction Address): The address of the next location in program memory to be loaded into the program counter.

C) Instruction Decoder Inputs:

2) IR (Instruction Register): The actual instruction, containing the opcode, target offset in the case of a branch, and possibly some immediate data
3) PC (Program Counter): In case of a branch the target offset is added to This value

D) Instruction Decoder Outputs:

4) MW (Memory Write): Allows data to be written to the processor registers
5) RW (Read-after-write*):* Allows read of processor register
6) DA (Data Address*):* Address of processor register data to read/write
7) DS (Data Select): Select Function output, data output, or target offset
8) **FS** (Function Select Bus): Word which contains the op-code which will tell the ALU what type of operation to perform
9) **A** (A Data Bus): Word which contains data for the A input of ALU
10) **B** (B Data Bus): Word which contains data for the B input of ALU

E) Programmers Model:

From the programmer's viewpoint the processor provides a simple direct addressing mode and a relatively large address space for a 8-bit processor

3. VHDL & FPGA

A) VHDL:

VHDL stands for **V**HSIC (Very High Speed Integrated Circuits) **H**ardware **D**escription **L**anguage. It has become now one of industry's standard languages used to describe digital systems. The other widely used hardware description language is Verilog. Both are powerful languages that allow you to describe and simulate complex digital systems. A third HDL language is ABEL (Advanced Boolean Equation Language) which was specifically designed for Programmable Logic Devices (PLD). ABEL is less powerful than the other two languages and is less popular in industry.

Simulation and synthesis are the two main kinds of tools which operate on the VHDL language. The Language Reference Manual does not define a simulator, but unambiguously defines what each simulator must do with each part of the language.

VHDL does not constrain the user to one style of description. VHDL allows designs to be described using any methodology - top down, bottom up or middle out! VHDL can be used to describe hardware at the gate level or in a more abstract way. Successful high level design requires a language, a tool set and a suitable methodology.

Benefits of VHDL:

- Executable specification
- Validate spec in system context (Subcontract)
- Functionality separated from implementation
- Simulate early and fast (Manage complexity)
- Explore design alternatives
- Get feedback (Produce better designs)
- Automatic synthesis and test generation (ATPG for ASICs)
- Increase productivity (Shorten time-to-market)
- Technology and tool independence (though FPGA features may be unexploited)
- Portable design data(protect investment)

B) FPGA:

FPGA stands for "FIELD PROGRAMMABLE GATE ARRAY". This is chip which consists of programmable hardware which can be used to design any digital circuit into it. It consist of lot of programmable hardware in it like

multiplexer, lookup table, gates and flip flops which all can be connected using fuses and switches.

The fuses are basically programmable so that we configure any type of hardware in FPGA within less designed time.

Advantages of FPGA:

1. It is used due to its flexibility. It facilitate quick time to market and there incredible power at reprogram ability often makes them heart of system.

4. HAZARDS:

1] Code Quality:

- The performance of a RISC processor depends greatly on the code that it is executing. If the programmer (or compiler) does a poor job of instruction scheduling, the processor can spend quite a bit of time stalling.
- Time stalling: Waiting for the result of one instruction before it can proceed with a subsequent instruction.

2] Code expansion:

- Since CISC machines perform complex actions with a single instruction, where RISC machines may require multiple instructions for the same action, code expansion can be a problem.
- Code expansion refers to the increase in size that you get when you take a program that had been compiled for a CISC machine and re-compile it for a RISC machine. The exact expansion depends primarily on the quality of the compiler and the nature of the machine's instruction set

3] System Design:

- Another problem that faces RISC machines is that they require very fast memory systems to feed them instructions. RISC-based systems typically contain large memory caches, usually on the chip itself. This is known as a first-level cache.

5. CONCLUSION

In conjunction with other groups the VHSIC Hardware Description Language (VHDL) will be used to design and implement a functional 8-bit Reduced Instruction Set Computer (RISC) Processor. This group is responsible for constructing an instruction set architecture, determining the structure of the internal registers and memory, and designing the control unit of the processor. The control unit of the processor reads an instruction from memory, generates the appropriate control signals for the rest of the processor, and handles program branches. Our VHDL design and simulation will be done using Mentor Graphics ModelSim software

ACKNOWLEDGMENT

Sangita Mohite indebted to Principal Dr. S. R. Chougule, for giving permission for sending the paper to the journal. She is also thankful to the H.O.D. and colleagues for support.

REFERENCES

[1]. John L. Hennessy, and David A. Patterson, “Computer Architecture A Quantitative Approach”, 4th Edition; 2006.

[2] Charles E. Gimarc, Veljko M. Mhtinovic, "RISC Principles, Architecture, and Design", Computer Science Press Inc., 1989.

[3] "Design ware Components Datebook", Synopsys Inc., Version 1997.08. [Hennessy, Patterson "Computer Architecture - A Quantitative Approach", Znd edition 1996, Morgan Kaufman Publishers Inc.

[4] Nurmi, **J.,** Takala, J.: "A New Generation of Parameterized and Extensible DSP Cores", *1997 IEEE Workshop on Signal Processing Systems,* SIPS97, Leicester, U.K., November 1997, pp.320-329.