DESIGN AND ANALYSIS OF HIGH PERFORMANCE RISC PROCESSOR USING HYPERPIPELINING TECHNIQUE

Charu Sharma[1], Gurpreet Singh Saini[2]

PG Scholar[1], Assistant Professor[2]

EEE Department[1], ECE Department[2]

Arni University

H.P, India

ABSTRACT:

The paper proposes RISC processor with floating point arithmetic for high speed and low power consumption .It is having five stage pipelining which is designed using VHDL. Number of instruction which are designed for this processors. We use 5-stage pipelining which involves instruction fetch module, instruction decode, module, execution module, memory i/o and write block.

KEYWORDS: Complex instruction set computing (CISC), Instruction Set Architecture (ISA),Reduced instruction set computing (RISC), *processors.*

1. INTRODUCTION

1.1 Processor

A processor is the logic circuitry that responds to and processes the basic instructions that drive a computer. The term processor has generally replaced the term central processing unit (CPU). The processor in a personal computer or embedded in small devices is often called a microprocessor. Types of processors according to the Instruction set:-

1)CISC (Complex instruction set computing): The CISC concept is an approach to the Instruction Set Architecture (ISA) design that emphasizes doing more with each instruction using a wide variety of addressing modes, variable number of operands in various locations in its Instruction Set. As a result, the instructions are of widely varying lengths and execution times thus demanding a very complex Control Unit, which occupies a large real estate on Chip [9].

2)RISC (Reduced instruction set computing) : The RISC Processor have reduced number of Instructions, fixed instruction length, more general purpose registers, load-store architecture and simplified addressing modes which makes individual instructions execute faster, achieve a net gain in performance and an overall simpler design with less silicon consumption as compared to CISC.

1.2RISC features

The main features of RISC processor are the instruction set can be improved to speed instruction execution. No microcode is needed for single cycle execution. All instructions are fixed bit in length. This simplifies the instruction fetch mechanism since the location of instruction boundaries is not a function of the instruction type. The processor has small number of addressing modes. Only load and store instructions access memory, load/store instructions operate between memory and a register. The machine cycle time is minimized. The fixed size of the instructions allows the instructions to be easily piped. RISC provides a flexible and

expandable architecture that maximizes performance from any given semiconductor technology. RISC includes extensions to RISC concepts that help achieve given levels of performance at significantly lower cost than other systems. The design process is very fast and cost effective. In this design, most instructions are of uniform length and similar structure, arithmetic operations are restricted to CPU registers and only separate *load* and *store* instructions access memory. The Instruction cycle consists of four stages namely fetch, decode, execute and Store. After every instruction fetch, Control Unit generate signals for the selected Instruction. The architecture supports 8 instructions to support Arithmetic, Logical, Shifting and Load-store operations.

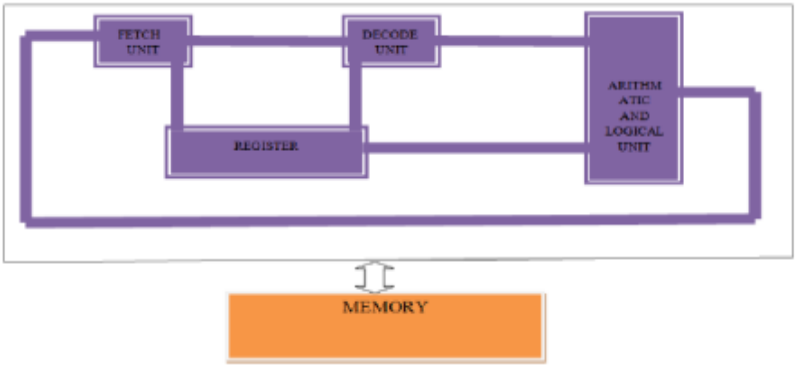

Fig 1: Block diagram of RISC Processor

The RISC processor architecture consists of Arithmetic Logic Unit (ALU) (including arithmetic and logical units, barrel shifter, booth multiplier), control unit (CU), Memory and Register File. RISC processor is designed with load/store architecture, meaning that all operations are performed on operands held in the processor registers and the main memory can only be accessed through the load and store instructions.

2. PIPELINING CONCEPT

Pipelining is a technique of decomposing a sequential process into sub-operations, with each sub process being executed in a special dedicated segment that operates concurrently with all other segments.

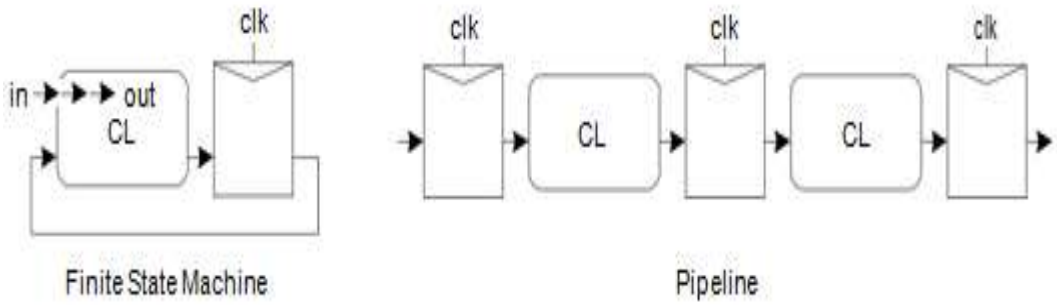

Fig 2. Comparison of FSM with Pipelined sequential structure

2.1 Different types of pipelining techniques

1) Super pipelining: Pipelining is the concept of overlapping of multiple instructions during execution time. Pipeline splits one task into multiple subtasks. These subtasks of two or more different tasks are executed parallel by hardware unit. The concept of pipeline can be same as the water tab. The amount of water coming out of tab in equal to amount of water enters at pipe of tab. suppose there are five tasks to be executed. Further assume that each task can be divided into four subtasks so that each of these subtasks is executed by one state of hardware. The execution of these five tasks is super pipelining. But the Non-pipelined system

takes a time equal to time taken to complete each task for the same operation. The total time required for n tasks is nt. The speed up of a pipeline processing over an equivalent non-pipeline processing is defined by the ratio.

2) Super Scalar Processor/Machine: The scalar machine executes one instruction one set of operands at a time. The super scalar architecture allows on the execution of multiple instruction, at the same time in different pipelined hardware. Here multiple processing elements are used for different instruction at the same time. Pipeline is also implemented in each processing elements.

3) Hyper Pipelining: In this paper, a method is discussed that how the functionality of a core can be multiplied by adding registers to the core. It does not only provide the less area usage compared to its individual instantiations, but it can also provide impact on the system performance as a whole. This method is called “hyper pipelining” here the hyper pipelined complex RISC core (OR1200 from Open Cores) is discussed. Hyper pipelining is a technique where the Core Multiplier multiplies the functionality of cores, bus-systems or complete sub designs. It implements registers (called pipes) in the design to create CMF independent designs, whereas CMF can be any number greater than 1. Since only registers are inserted, the resulting area is much less than duplicating the complete design. The result is a much smaller than ASIC or lesser than FPGA size. The applied method is also called "C-slow Retiming".

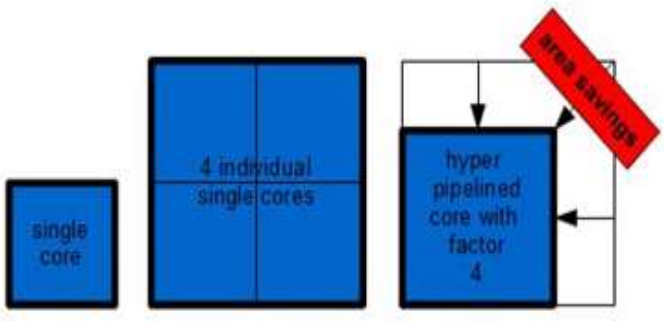

Fig 3: Hyper pipelined RISC core with CMF=4(Core Multiplication Factor)

Figure 3 shows the simplified structure of sequential logic. Inputs and sequential elements clocked by clk1 drive the combinational logic. The combinational logic drives the outputs and the data inputs of the registers. In Figure 3 each sequential element is duplicated with an intermediate register clocked by a second clock clk2. If clk2 is synchronous to clk1, but not edge-aligned, and if the timing is right (no setup or hold time violation between clk1 and clk2 registers), the behaviour of the sequential logic doesn't change. Here some sequential logic diagrams are shown in the following figures

3. HARDWARE IMPLEMENTATION OF RISC PROCESSOR

The figure shown below provides a quick tour of the key highlights and capabilities of the ISE Design Suite. Logic Edition and how it is used in typical design scenarios. The figure explains the main steps to getting a design through the entire tool chain from HDL entry to place and route and all the way through to bit stream generation. Bit stream generation covers common tasks like assigning pins and specifying constraints. It explains the most relevant places to analyze and visualize results.

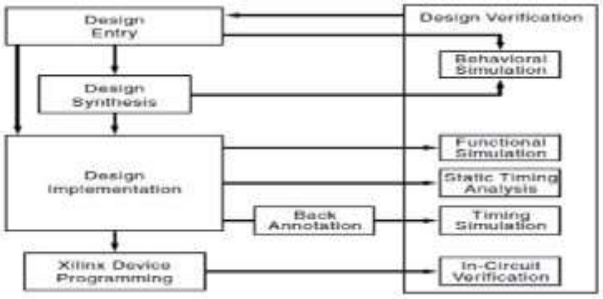

Fig 4: The key highlights and capabilities of the ISE Design Suite

4. SIMULATION RESULTS AND VALIDATION

ModelSim is used for ASIC and FPGA design. Mentor Graphics was the first to combine single kernel simulator (SKS) technology with a unified debug environment for Verilog, VHDL, and System C. The combination of industry-leading, native SKS performance with the best integrated debug and analysis environment make ModelSim the simulator of choice for both ASIC and FPGA designs. The best standards and platform support in the industry make it easy to adopt in the majority of process and tool flows.

4.1 Different steps for Compiling a Design

ModelSim is a very good approach for compiling a design. Modelsim Mentor Graphics was the first to combine single kernel simulator (SKS) technology with a unified debug environment for Verilog, VHDL, and System C. Different steps for compiling a design are:

1) Code Your Design
2) Start ModelSim
3) Set the Working Directory
4) Create a Work library
5) Compile a Design

B. Different steps for Simulating a Design

Modelsim is also very good tool to simulate a design. In the Modelsim for simulating a design first we have create a library and compile the source code into that library. Different steps for simulating a design are:

1) Code the Testbench
2) Compile the Testbench
3) Load the Testbench
4) Display Waveforms
5) Run the Simulation

C. Proposed RISC Simulations

To display the simulation waveforms after loading the testbench we go for the vsim command in the command prompt then from the object window we select the signals to addd into the waveform like "Add > To Wave >All selected signals into the region". Then in the waveform window click on “All Run”. Here according to this procedure the different timing Waveforms are shown with the explanation. Here in figure 3.3 executions of 200 Instruction Cycles by the processor is shown. In this figure data is which is entered for the different main signal is given below for OR1200 Testbench.. The waveform shown for the following signals as Cnt (cycle count)=200, System CLK=1, RST=0

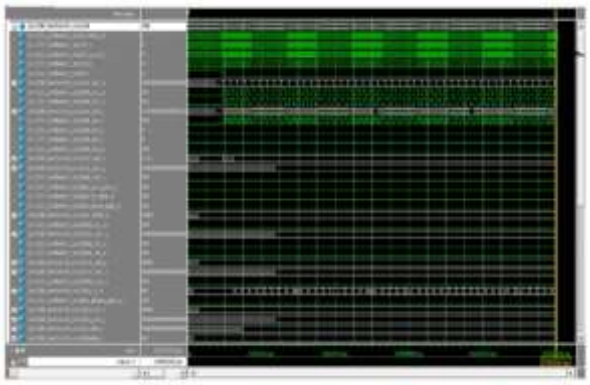

Fig 5: Execution of 200 Instruction Cycles by the processor

In fig 6 here the execution of the 24000 cycles is shown with the condition is that first the processor is Halted, Resetted and then again started depending on the Logic Low and Logic High condition on the Reset Control signal. When the processor is started again the address will be shown on the interface of Wishbone bus select line & the data will be received on the interference of wishbone date bus.

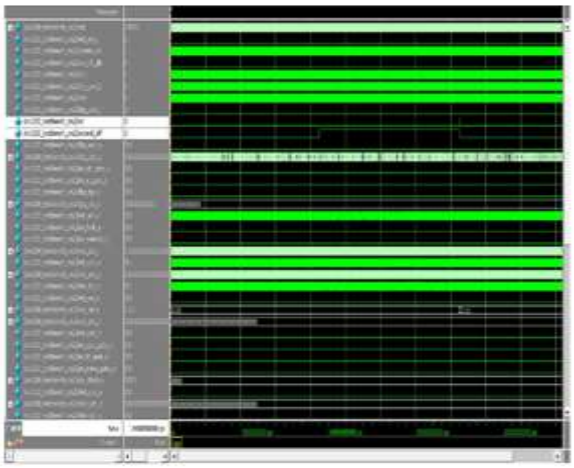

Fig 6: Execution of 24000-Simulation Cycle by the Processor-Halted, Resetted and then Started again

Here in fig 7 the processor execution is carried out for twenty cycles. In this timing waveform it is shown that when the reset is deactivated and system clock is activated the first data is received which is received from the memory and memory address is generated randomly through Random Instruction Generator.

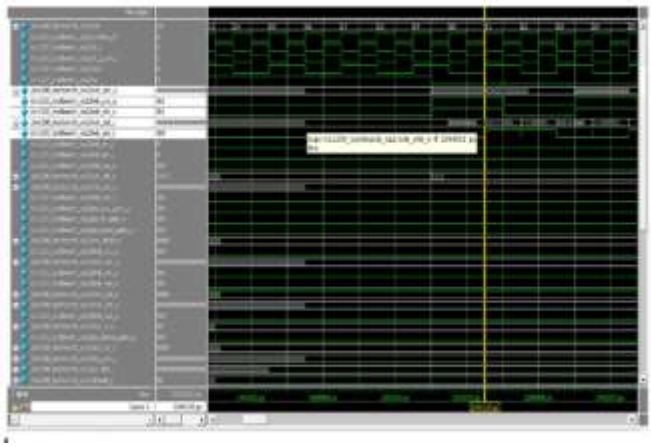

Fig 7: First Data is received by the Processor

In fig 8 the processor execution is carried out for nineteen cycles. In this the timing waveform it is shown that when the reset is deactivated and system clock is activated the first data is fetched from the memory and the memory address is generated randomly through Random Instruction Generator.

Fig 8: First Instruction fetches operation Started

In fig 9 the processor execution is carried out for nineteen cycles. In this the timing waveform it is shown that firstly logic high is given to the reset signal. Here system clock (clk) &clk_i are inverted to each other.

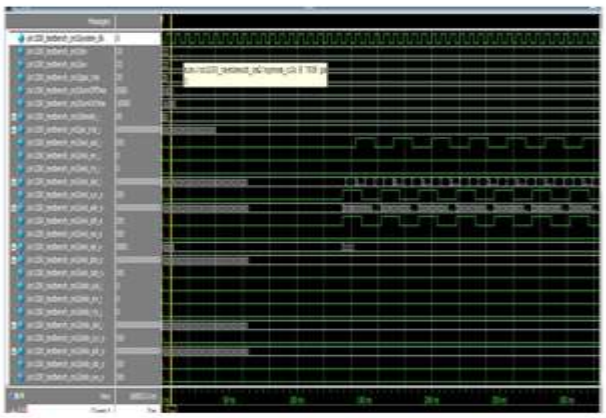

Fig 9:Reset is De-asserted; the Instruction bus is showing no address

5. CONCLUSION

The concept of hyper pipelining for RISC open cores 1200 HP was drawn from the Hyper Pipelined Technology used for Pentium series processors to increase the speed. By multiplying the clock frequency, the performance of the design is the same as the sum of the individual implementations. It is important to notice, that each "new" core works totally independent of the others.

In many cases, FPGAs work in conjunction with a conventional DSP as integrating pre and post processing functions, along with high performance signal processing. There is a constant requirement for efficient use of FPGA resources where occupying less hardware for a given system that can yield significant cost-related benefits. The expected outcome of this thesis work will be the development of RISC processor (reduced instruction set computer) i.e. a low power embedded processor -Enhance speed, Area saving as comparison to the actual area if implemented individuallyUsing hyper pipelining technique based on the open cores or core multipliers or core multiplication factor.

ACKNOWLEDGMENT

This paper work is made possible through the help and support from significant advisors and contributors of Arni University. First and foremost, I would like to deeply thank to my supervisor Er. Gurpreet Singh Saini, Assistant Professor & Head, Department of Electronics and Communication Engineering, Arni University for guiding and correcting at various stages of my paper work with attention and care. I thank to all my friends who help me at many stages in this work.

REFERENCES

[1] Reduced instruction set computing - Wikipedia, the free encyclopediaen.wikipedia.org/wiki/ Reduced _instruction_set_computing.

[2] Sharda P. Katke, G.P. Jain "Design and Implementation of 5 Stages Pipelined Architecture in 32 Bit RISC Processor", International Journal of Emerging Technology and Advanced Engineering , Vol. 2, Issue No.4, pp. 340-346, April 2012.

[3] Introduction to RISC Processors by ni logic Pvt.Ltd, pp. 1-42

[4] S. P. Ritpurkar, M. N. Thakare, G. D. Korde, "Design and Simulation of 32-Bit RISC Architecture based on MIPS using VHDL", IEEE, International Conference on Advanced Computing and Communication Systems (ICACCS -2015),Coimbatore, INDIA, January 2015.

[5] Sharda P. Katke, G.P. Jain, "Design and Implementation of 5 Stages Pipelined Architecture in 32 Bit RISC Processor", International Journal of Emerging Technology and Advanced Engineering, ISSN: 2250-2459, Vol. 2, Issue 4, April 2012.

[6] Pranjali S. Kelgaonkar, ShilpaKodgire, "Pipelined 32bit RISC MIPS Processor on Spartan-6 FPGA", International Journal of Science, Engineering and Technology Research (IJSETR), ISSN: 2278-7798, Vol. 5, Issue 8, August 2016.

[7] Samiappa Sakthikumaran,S.Salivahanan and V.S.Kaanchana Bhaaskaran , June 2011, "16-Bit RISC Processor Design For Convolution Application ",IEEE International Conference on Recent Trends In Information Technology, pp.394-397.

[8] Rohit Sharma, Vivek Kumar Sehgal, Nitin Nitin1, Pranav Bhasker, Ishita Verma , 2009, "Design And Implementation Of 64-Bit RISC

Processor Using VHDL", UKSim : 11th International Conference on Computer Modeling And Simulation, pp. 568 – 573.

[9] V. B. Saambhavi and V. S. Kanchana Bhaaskaran, "A 16-BIT RISC MICROPROCESSOR USING DCPAL CIRCUITS", International Journal of Advanced Engineering Technology (IJAET), Vol. 2, no. 1, pp.1-9, 2011.

[10] Samanta, S. "A Contemporary Review of Adiabatic Computing," IEEE Symp. Computers and Devices for Communication, 2009.CODEC 2009.

[11] Yasuhiro Takahashi, Member, IACSIT, Toshikazu Sekine, and Michio Yokoyama, "Design of a 16-bit Non-pipelined RISC CPU in a Two Phase Drive Adiabatic Dynamic CMOS Logic", International Journal of Computer and Electrical Engineering, Vol. 1, No. 1, April 2009.