International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 2, Issue 3, March 2015)

64-Bit RISC Processor Design using Verilog

Manjula H.R
M.Tech Student, VLSI Design & Embedded System,
Appa Institute of Engineering & Technology
Gulbarga, Karnataka, India
hrmanjula21@gmail.com

Associate Prof.Rekha S
Department of Electronics & Communication Engineering
Appa Institute of Engineering & Technology
Gulbarga, Karnataka, India
rekha_nl@yahoo.co.in

***Abstract*- The Reduced Instruction Set Computer or RISC is a microprocessor design principle that favours a smaller and simpler set of instructions that all take same amount of time to execute. RISC architecture is used across a wide range of platforms from cellular phones to super-computers. In this paper the behavioural design and functional characteristics of 64- bit RISC processor is proposed, which utilizes minimum functional units without compromising in performance. The instruction word length is 17-bit wide. The processor supports 23 instructions. It has 12 general purpose registers. Each register can store 64-bit data. The processor has 64-bit ALU capable of performing arithmetical and logical operations. The processor also incorporates a flag register which indicates carry, zero and parity status of the result. All the modules in the design are coded in verilog. The design entry and synthesis is done using Xilinx ISE 13.2 tool.**

***Keywords*- RISC, Instruction set, ALU, Verilog, Xilinx ISE 13.2 tool.**

I. INTRODUCTION

RISC-The reduced instruction set computer, or RISC, is a microprocessor CPU design philosophy that favors a smaller and simpler set of instructions that all take about the same amount of time to execute. The most common RISC microprocessors are ARM, DEC Alpha, PA-RISC, SPARC, MIPS, and IBM's PowerPC. The idea was inspired by the discovery that many of the features that were included in traditional CPU designs to facilitate coding were being ignored by the programs that were running on them. Also these more complex features took several processor cycles to be performed. Additionally, the performance gap between the processor and main memory was increasing. This led to a number of techniques to streamline processing within the CPU, while at the same time attempting to reduce the total number of memory accesses. When the controller design become more complex in CISC and the performance was also not up to expectations, people started looking on some other alternatives. It had been found that when a processor talks to the memory the speed gets killed. So the one improvement on CPI was to keep the instruction set very simple. Simple in not the way it works but the way it looks. That's why we have very few instructions in any typical RISC architecture where processor asks data from memory probably not other than Load and Store. We avoid keeping such addressing modes. The complexity of controller design has been overcome with the help of operands and op-code bits fixed in instruction register. At the end the pipelining added a new dimension in the speed just with the help of some additional registers. Now what pipeline does is it increases throughput by reducing CPI. The instruction can be executed effectively in one clock cycle. The pipelining in any kind of architecture took birth from the inherent parallelism and the idle states of components.

Features which are generally found in RISC designs are:

- **Uniform instruction encoding** (for example the op-code is always in the same bit position in each instruction, which is always one word long), which allows faster decoding;
- A **homogeneous register set**, allowing any register to be used in any context and simplifying compiler design.
- **Simple addressing modes** (complex addressing modes are replaced by sequences of simple arithmetic instructions);
- **Few data types** supported in hardware (for example, some CISC machines had instructions for dealing with byte strings. Others had support for polynomials and complex numbers. Such instructions are unlikely to be found on a RISC machine).

Over many years, RISC instruction sets have tended to grow in size. Thus, some have started using the term "load-store" to describe RISC processors, since this is the key element of all such designs. Instead of the CPU itself handling many addressing modes, load-store architecture uses a separate unit dedicated to handling very simple forms of load and store operations. CISC processors are then termed "register-memory" or "memory-memory".
Today RISC CPUs (and microcontrollers) represent the vast majority of all CPUs in use. The RISC design technique offers power in even small sizes, and thus has come to completely dominate the market for low-power "embedded" CPUs. Embedded CPUs are by far the largest market for processors. RISC had also completely taken over the market for larger workstations for much of the 90s. After the release of the Sun SPARC station the other vendors rushed to compete with RISC based solutions of their own. Even the mainframe world is now completely RISC based.

International Journal of Ethics in Engineering & Management Education
Website: www.ijeee.in (ISSN: 2348-4748, Volume 2, Issue 3, March 2015)

II. PROPOSED PROCESSOR ARCHITECTURE

The architecture of proposed 64-bit Processor is shown in Fig.1

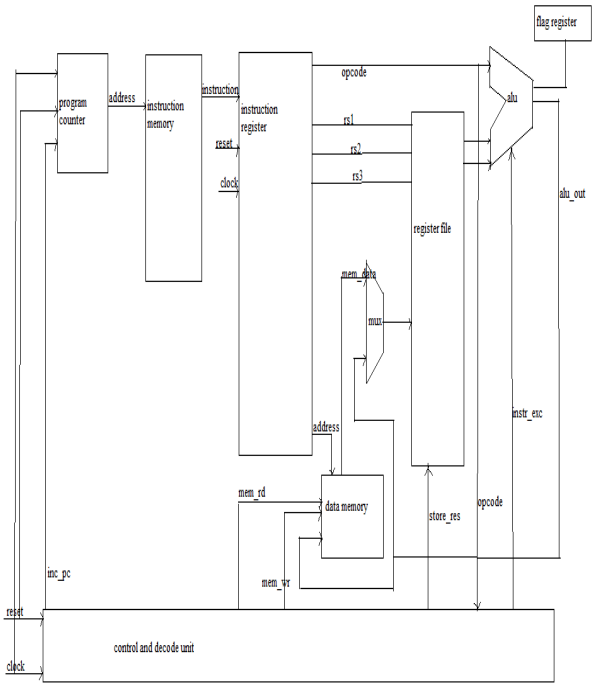
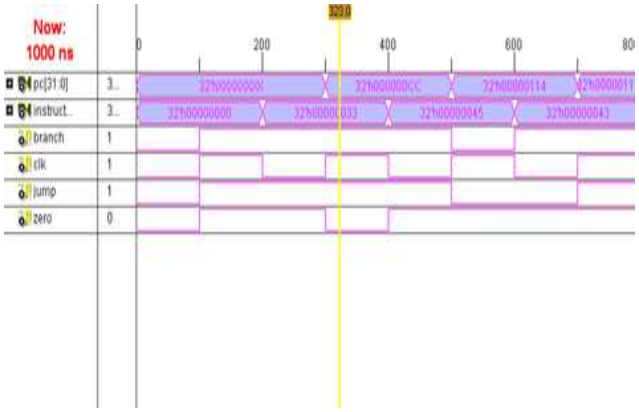


Fig 1. 64-Bit processor architecture

The processor works at positive edge of clock and when reset is high. When clock and reset inputs are given to processor, the instruction from instruction memory is loaded into instruction register. The address of the instruction memory, from where the instruction to be loaded, is given by the program counter. After reset the content of program counter (PC) is 0. So program will start from the address 00000000.After loading the instruction into instruction register, the program counter is incremented by 1 or some other value that depends on the previously executed instruction. Next the instruction from instruction register is decoded as opcode, source register, destination register. According to source registers, the control and decode unit will load the operands from either data memory or internal register. If the operand to be loaded is from data memory then control and decode unit will set the signals mem_rd to 1 and mem_wr to 0 indicating that memory read operation is going on. Then the two operands are loaded into the ALU unit where the execution of operation will be done. At that time the control and decode unit will tell which operation we want to do using OpCode.

After doing ALU operation that output can be stored in either internal registers or memory. If we want to store the data in the internal registers then the control and decoder unit will select the destination register using the operand destination address. And if we want to store the data in the memory then the control and decoder unit will set the mem_wr to 1 and mem_rd to 0 so that we can write the data into the memory.

III. SIMULATION AND SYNTHESIS RESULTS



HDL Synthesis Report

Macro Statistics	
# Multipliers	: 1
64x64-bit multiplier	: 1
# Adders/Subtractors	: 3
64-bit adder carry out	: 1
64-bit addsub	: 1
65-bit subtractor	: 1
# Registers	:24
1-bit register	: 8
64-bit register	:16
# Comparators	: 1
64-bit comparator equal	: 1
# Multiplexers	: 2
64-bit 16-to-1 multiplexer	: 2
# Logic shifters	: 4
64-bit shifter logical left	: 1
64-bit shifter logical right	: 1
65-bit shifter logical left	: 1
65-bit shifter logical right	: 1
# Xors	: 1
64-bit xor2	: 1
Design-Statistics	
# IOs	:331
Cell-Usage	
# BELS	: 1
# GND	: 1
# IO Buffers	:137
# OBUF	:137

IV. CONCLUSION

From the overall design of RISC processor, it is concluded that i have implemented the above processor architecture by different instruction. Here arithmetic, logical and shifting programs are created in verilog using Xilinx 13.2 Software. Simulated output results are compared with the expected results which we considered at the timing of design.

ACKNOWLEDGEMENT

I am thankful to Holiness of Poojya Dr. Sharanabaswappa Appa, Mahadasoha Peethadhipati, Sharanabasaweshwar Samsthana, Gulbarga. President, Sharanabasaweshwar Vidhya Vardhak Sangha, Gulbarga. And also thankful to Principal, Dean AIET, Gulbarga.

REFERENCES

[1]. Preetam Bhosle, Hari Krishna Moorthy , "FPGA IMPLEMENTATION OF LOW POWER PIPELINED 32-BIT RISC PROCESSOR" International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-1, Issue-3, August 2012.

[2]. 2. Mrs. Rajitha Gullapalli et al, "DESIGINING OF INSTRUCTION FETCHING IN 64-BIT RISC PROCESSOR USING VERILOG" international journal of innovative technologies, VOL. 02, ISSUE 05, MAY 2014 ISSN 2321 –8665.

[3]. 3. Y.Shekar et al, "Cryptographic Algorithms Implementation on RISC Processor" International Journal of Inventive Engineering and Sciences (IJIES) ISSN: 2319–9598, Volume-1, Issue-12, November 2013.

[4]. 4. Indu.M & Arun Kumar.M, "Design of Low Power Pipelined RISC Processor" International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (ISO 3297: 2007 Certified Organization) Vol. 2, Issue 8, August 2013.

[5]. 5. Surendra Bajia et al, "Power Performance of Core Hardware modeling of 64-bit Pipeline MIPS RISC Processor Using VHDL" International Journal of Advances in Electrical and Electronics Engineering.