IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)
e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 8, Issue 4 (Nov. - Dec. 2013), PP 01-06
www.iosrjournals.org

Implementation of RISC-Based Architecture for Low power applications

Shaik Afroz [1], M.Sumalatha[2]
[1] *PG Student (M. Tech), Dept. of ECE, Chirala Engineering College, Chirala, A.P, India.*
[2] *Associate Professor, Dept. of ECE, Chirala Engineering College, Chirala, A.P, India.*

Abstract: *This paper explains the design of a simple RISC processor and procedure for 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 along with reducing the power consuming while executing the instruction. The RISC architecture follows the philosophy that one instruction should be performed every cycle.*

This work presents the design and implementation of a 32 bit RISC soft core processor intended for computer architecture introduction considered to be an effective solution for computer comprehension. The idea of this project was to create a microprocessor as a building block in VHDL than later easily can be included in a larger design. It will be useful in systems where a problem is easy to solve in software but hard to solve with control logic. A state machine dedicated to the function can of course replace the microprocessor and associated software. However at a high level of complexity it is easier to implement the function in software. In this project Modelsim is used for logical verification and further synthesizing is carried on Xilinx-ISE tool. To perform the power analysis the Xilinx xpower analyzer is used.

Keywords: *Low Power, Speed, RISC, Processor, Xilinx.*

I. Introduction

Processors are regarded as one of the most important devices in our everyday machines called computers. Before we start, we need to understand what exactly processors are and their appropriate implementations. Processor is an electronic circuit that functions as the central processing unit (CPU) of a computer, providing computational control. Processors are also used in other advanced electronic systems, such as computer printers, automobiles, and jet airliners, Calculators and etc.

Typical processors incorporate arithmetic and logic functional units as well as the associated control logic, instruction processing circuitry, and a portion of the memory hierarchy. Portions of the interface logic for the input/output (I/O) and memory subsystems may also be infused, allowing cheaper overall systems. While many processors and single-chip designs, some high-performance designs rely on a few chips to provide multiple functional units and relatively large caches.

Processors have been described in many different ways. They have been compared with the brain and the heart of humans. Their operation has been likened to a switched board, and to the nervous system in an animal. They have often been called microcomputers. The original purpose of the processor was to control memory. That is what they were originally designed to do, and that is what they do today. Specifically, a processor is "a component that implements memory."

The traditional approach to develop a digital system was to use a set of interconnected digital integrated circuits like counters, buffers, logic gates and memory. That task required lots of analysis, testing and the need to adapt the design to the hardware's inherent limitations (speed, response time, power consumption, etc.) which resulted in capped headroom for development.

Also, every design change implied a whole analysis but sometimes the prototyping hardware wouldn't allow any expansion without a considerable –and most times expensive- upgrade.

At present, technological advance has brought new options like programmable logic as Complex Programmable Logic Devices (CPLD) or Field Programmable Gate Arrays (FPGA) with more sophisticated simulation and design verification environments, which enable engineers to reach new levels of complexity and robustness, while greatly reducing the time between development and implementation.

Also, those advances let engineers focus on the application needs, rather than to fit the system onto existing hardware. This way, one can develop a system that can be optimized for manufacturing on a single chip, with the capacity to add or remove modules according to the requirements in the future.

Modern processor design sometimes reduce the implementation effort by acquiring some of these elements as Intellectual Property (IP) or through implementation techniques to build the other components, like VHDL or Verilog language and a proprietary synthesizer depending on each hardware vendor.

There exist different approaches in order to obtain high comprehension when explaining computer architecture, one approach uses simulation of processors that allow interaction with each module that composes a computer as on [1]-[4]; another option is to use FPGA devices and VHDL language to construct a simple computer [5]-[6], but there is the disadvantage that by programming there is a lack of understanding in dataflow. There is even an approach that uses MSI digital components such as TTL in order to construct a computer [7], it is a good exercise to realize interconnections between components, but sometimes using wires to build those connections is excessively time consuming.

On the other hand, there is an option to construct a simple computer by using a GUI that allow to design at gate level by placing components in a spreadsheet, it is easy to use and easy to understand dataflow between components even is more comprehensive if we can create all of the computer components from logical gates.

In this present work, the design of an 8-bit data width Reduced Instruction Set Computer (RISC) processor is presented; it was developed with simplicity and implementation efficiency in mind. It has a complete instruction set, program and data memories, general purpose registers and a simple Arithmetical Logical Unit (ALU) for basic operations. It operates following a multi-cycle execution nature and is implemented on a Xilinx Spartan-3E FPGA.

It is an advantageous Processor because of its Speed, Simpler hardware, shorter design cycle and User (programmer) friendly nature.

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.

To reduce the number of accesses to main memory, designers added instruction and data cache to the processors. A cache is a special type of high speed RAM where data and the address of the data are stored. Whenever the processor tries to read data from main memory, the cache is examined first. If one of the addresses stored in the cache matches the address being used for the memory read (called a hit), the cache will supply the data instead. Cache is commonly ten times faster than main memory, so you can see the advantage of getting data in 10 nanoseconds instead of 60 nanoseconds. Only when we miss (i.e., do not find the required data in the cache), does it take the full access time of 60 nanoseconds. But this can only happen once. Since a copy of the new data is written into the cache after a miss. The data will be there the next time we need it. Instruction cache is used to store frequently used instructions. Data cache is used to store frequently used data.
Implementing fewer instructions and addressing modes on silicon reduces the complexity of the instruction decoder, the addressing logic, and the execution unit. This allows the machine to be clocked at a faster speed, since less work needs to be done each clock period.

RISC typically has large set of registers. The number of registers available in a processor can affect performance the same way a memory access does. A complex calculation may require the use of several data values. If the data values all reside in memory during the calculations, many memory accesses must be used to utilize them. If the data values are stored in the internal registers of the processor instead, their access during calculations will be much faster. It is good then to have lot of internal registers.
CISC is an acronym for complex instruction set computer. The reasons to have the large instruction set are:

1. CISC systems are complex as the architecture and the instructions are modified as per the technology and hence there is a requirement for more number of instructions and therefore will have many addressing modes.
2. The main function of any microprocessor is to fetch the data and process it. The chip as such doesn't have any provision for memory i.e., memory is external. As process needs to access data from a memory or from I/O peripherals there is a necessary for more instructions.

Drawbacks behind having the large instruction set:

- The large instruction set is responsible for a very complicated control logic that generates a host of control signals and interrupts that are important for the working of the processor.
- This large instruction set creates problems for the compiler as it has to write code for each individual instruction. It becomes difficult to debug if there is any mistake in the code of the compiler.
- The Large instruction set has a very complicated decoding logic which is responsible for a more number of stages or for an increased propagation delay of the control logic. This is because an increase in the number of stages leads to proportionate increase in the number of gates. Transistor has a rise time, a fall time contributing to the propagation delay of the signal. So at every gate the signal gets delayed and this cumulative delay in propagation adds up, to decrease the processing speed of the data.
- They have a variety of instruction formats like one byte, two byte, three byte. They have a large number of addressing modes.

- In general the pipelining is used to increase the speed of memory operations with separate control over the bus. They are difficult to handle if there is a considerable difference between instruction lengths and execution cycle of different instructions then the pipeline design will be much more complicated.

RISC (Reduced Instruction Set Computer):
The advantages of instruction set of RISC are as follows
- In the RISC based system the frequently used instructions are hardware realized and it is therefore minimizes the memory access time and the decoding time.
- It is easy to write code for fixed instruction set, so the compiler supports for efficient translation of high level language programs into machine language programs.
- As the architecture supports pipeline in different segments, it may be possible to execute number of instructions in a single instruction cycle.
- Memory access is limited to load accumulator and store accumulator instructions only.

II. Design and Development of the Processor

The designed processor is based on the Harvard architecture, in which the size of the instructions is not related to the size of the data, and therefore it can be optimized in a way that any instruction occupies a single position of program memory, thus obtaining greater speed and a minor program length. Also, the access time to the instructions can be superposed with the one of the data, obtaining a greater speed in each operation.
The processor presented includes a RISC instruction set and uses a Single Instruction – Single Data (SISD) execution order.

Its main characteristics are:
• Eight 8-bit general purpose registers
• 256 allocations of 16-bit wide ROM program memory
• 256 allocations of 8-bit wide RAM data memory
• ALU with basic arithmetic and logical operations

Operational Blocks:
Arithmetic-Logic Unit (ALU)

The Arithmetic-Logic Unit has 8 operations; each one of them was created and converted into a symbol, then, a multiplexor was placed in order to obtain a 3-bit selector. Also, it has 3 flags: carry (C), half carry (H) and zero (Z), which indicate if there is a carry, a half carry or a zero after any ALU operation see Fig. 1.

RAM Data Memory

The RAM memory is a data storage block, there the stack is handled and other data are kept as variables. It is built with 32 memory blocks of 8 bits each one. The address input is divided in 2 parts.
The first part has 3 bits select the memory block to read or write using a decoder. The second has 5 bits that select the memory location between 0 and 31 of the previously specified portion by the first part. In Fig. 1 appears the schematic diagram of the RAM memory.

Program Counter (PC)

The program counter produces the address to read instructions from the program memory. It has to be capable of loading a random address if the program requires so (i.e. loops or branches), and should be able to wait while the other functional parts complete their tasks (i.e. while the ALU gets the sum of 2 registers). An 8-bit parallel-load counter was used [9].

Control Unit

The control unit operates as a state machine. States 0, 1 and 3 remain equal, whereas the second state can change depending on the instruction read by the decoder. In this way: state 0 represents a reset stage, state 1 represents the fetch/write back stage and finally during state 3 the program counter is incremented.

ROM Program Memory

The program memory –as its name describes- stores instructions to be executed. It has to be non-volatile and fast. It was decided to use internal ROM as program memory, because it was the fastest option and eliminated the need for external storage. This memory was built with 256x1 bit ROM blocks, with a total of 16 blocks to store 256 16-bit instructions.

Instruction Registers and Decoder: Instruction registers store the instruction read from the program memory, and keep it as an output for the decoder, which separates the operation code and operands to execute the command.
Two 8-bit registers (D-type flip-flops) were used, one for the high byte and one for the low byte (16-bit instruction).

The instruction decoder deals with the raw data stored in the instruction registers, and separates it in parts: Operation code (OPCODE), Rd, Rs and A/K, so that these values can be sent to the corresponding component, like the ALU, General Purpose Register block, etc. This is achieved simply by using a set of buffers inside a block to sort the signals to separate buses.

General Purpose Registers (GPRs) store and save operands and results during program execution. ALU and memories must be able to write/read those registers, so a set of eight 8-bit registers were used, along with multiplexers and a decoder to control which register is read or written. Two registers can be read at a time but only one register can be written at a time see Fig. 1
The RTL Schematic of the Designed model is shown in Fig. 2.

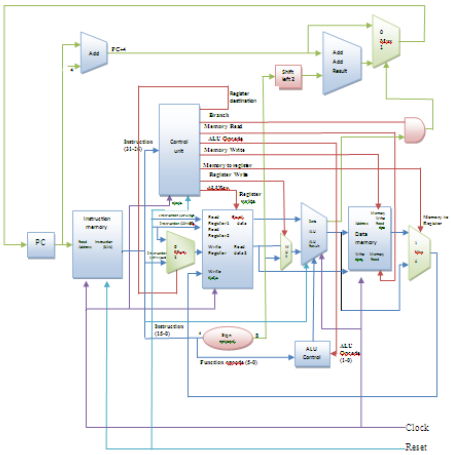

Figure 1 Block diagram of the RISC Processor

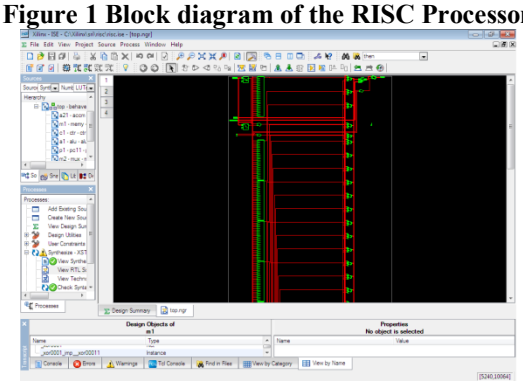

Figure 2 RTL Schematic of the Implemented Processor

III. Results and Conclusions

In this project it is observed that the RISC based system is simulated using VHDL. The overall system is simulated and synthesized. After synthesizing the system we could get a statistical data about the number of input-output buffers, the number of registers, number of flip-flops and latches were used .The modules simulated are Accumulator, Buffer, , Instruction Register, Multiplexer, Program Counter, Control Logic Decoder, Arithmetic Logic Unit and the overall system. Few instructions were executed and their timing sequences were analyzed. It shows that the different operations of the instruction including the decoding and execution come in the overall system. Therefore we conclude that the behavior shows, the system is working as RISC as instruction will be executed within a single clock cycle. The power consumed for the processor is 132mw.when compare to the previous architecture it became half in terms of power consumption .So based on the power report the proposed architecture is efficient in terms of power parameter.

For the complete CPU implementation, a Xilinx Spartan3 XC3S1000-5FG320 FPGA was used. For design purpose, XilinxISE Web pack tools were used under schematic mode to allow direct gate level designs. Simulation is performed on the included XilinxISE Simulator, designed to work with files generated by

XilinxISE. In order to simplify the conversion of CPU test programs, a specific translator was designed in MATLAB language, that reads a Microsoft Excel archive (*.xls) which contains the program in assembly language, and translates it to machine code, giving the proper format to load the instructions to the program memory.

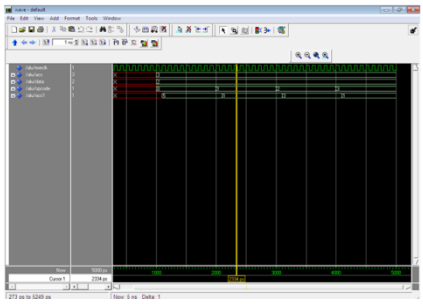

Figure 3 Simulation Result for ALU Module

In Fig.3,Every signal reacts at the clock rising edge only. Here in the input (acc and data) the given data is 3 and 2 . Based on the opcode we are getting the output acc1.Here we are applied opcode as 0,1,2,3.so the outputs we are getting as 5,1,3,1 which are obtained by performing addition, subtraction, data increment and data decrement .In the same way we are applying the remaining opcode also.

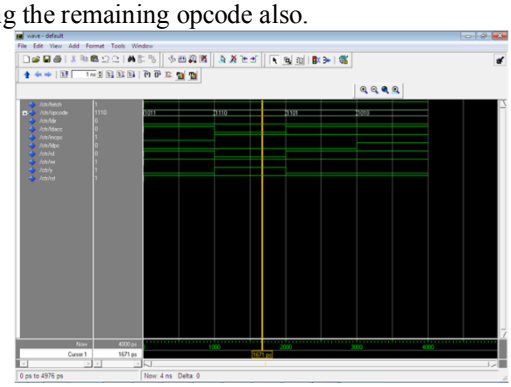

Figure 4 Simulation Result of Control Unit

From Fig 4, based on the opcode generated, Control unit generates necessary control signals as outputs whose combination, results in different actions. Different outputs like incpc, ldacc, ldir, ldpc, rd, rst, wr, and y are generated. Here, for opcode, ”0001” different outputs are obtained as seen above.

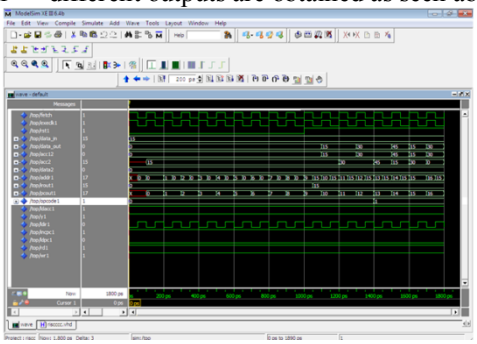

Figure 5 Simulation Result for the TOP Module

Here the instructions are given through opcode signal and the data is given through data_in signal. Based on the opcode and data_in the data_out is generated which shows the functionality of the design. It is shown in Fig. 5.

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality and presentation of this paper.

References:

[1] Pan-Song, Huang-JiYe, SOPC Technology Utility Tutorial, Tsinghua University Press, 2006.
[2] Zheng-WeiMin, Tang-ZhiZhong. Computer System Structure (The second edition), Tsinghua University Press, 2006.
[3] Ramdas, T. Li-Minn Ang and Egan, G., "FPGA implementation of an integer MIPS processor in Handel-C and its application to human face detection," in Proc. of IEEE Region 10 Conference, Vol. 1, pp. 36-39, 2004.
[4] Xizhi Li and Tiecai Li, "ECOMIPS: an economic MIPS CPU design on FPGA," in Proc. of the 4th IEEE International Workshop on System-on- Chip for Real-Time Applications, pp. 291-294, 2004.
[5] D. M. Harris and S. L. Harris, Digital Design and Computer Architecture,1st edition, Morgan Kaufmann, 2007, USA.

[6] Balpande, R.S. and Keote, R.S., "Design of FPGA based Instruction Fetch & Decode Module of 32-bit RISC (MIPS) Processor," in Proc. of International Conference on Communication Systems and Network Technologies, pp. 409-413, 2011.
[7] MIPS Technologies, Inc. MIPS32™ Architecture For Programmers Volume II: The MIPS32™ Instruction Set June 9, 2003.
[8] H. ElAarag, "A complete design of a RISC processor for pedagogical purposes," Journal of Computing Sciences in Colleges, vol. 25, Issue 2, pp. 205-213, 2009.
[9] Pattersson, Hennessy, Computer Organization and Design, The hardware/software interface, 2nd Edition, Morgan Kaufmann, 1998. [9] Tocci, Widmer, Moss, "Sistemas Digitales, Principios y aplicaciones," 10ª Edición, Pearson, 2007.

Authors Profile:

SHAIK AFROZ is Pursuing his M. Tech from Chirala Engineering College, Chirala in the department of Electronics & Communications Engineering (ECE) with specialization in VLSI & Embedded Systems.

M.SUMALATHA is working as an Associate Professor in the department of Electronics & Communication Engineering in Chirala Engineering College,Chirala. She has 7 years of teaching experience.