International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 09 | Sep 2018 www.irjet.net p-ISSN: 2395-0072

Design, Analysis and Processing of Efficient RISC Processor

Ramareddy[1], M.N.Pradeep[2]

[1]M-Tech., VLSI D& Embedded Systems, Dept of E&CE, Dayananda Sagar College of Engineering, Bangalore. Karnataka, India
[2]Professor, Electronics and Communication Engineering, Dayananda Sagar College of Engineering, Bangalore. Karnataka, India

---

Abstract - Due to the advancement in the modern technology, the processors in digital signal world are playing major role in almost all the electronic gadgets. RISC processor takes a highest level compared to other processors in terms of speed and area acquisition. Hence in this paper we have implemented 32-bit RISC processor architecture which is better when compared with other existing works. The main feature of a RISC processor is to fetch the instruction with only one clock cycle. In this proposed architecture we have succeeded to implement the concept of fetch, decode and execute more efficiently using the VHDL for the design implementation and simulated on Xilinx ISE 14.5 and synthesized on Spartan 6. The results show that the proposed 32-bit RISC processor architecture is more efficient than the other existing works and can be implemented on the hardware.

Key Words: VHDL, FPGA, Digilent Atlys, Xilinx ISE, Operating frequency, etc.

1. INTRODUCTION

New society is highly dependent upon technological things such as gadgets. The main requirements of any gadgets are that it should be small in size, high processing speed and good configuration. The configurability issues are the main reason behind the rapid growth in the used of embedded system on various electronic based consumer product.

We know most of the embedded systems used a microprocessor or microcontroller as operating core in the total architecture which acts as a brain or controlling unit and processing unit of the system. The microcontroller based architectures are used in the system where the applications do not have various different kinds of complex operations. These types of architectures are mainly used for small and medium range applications mainly. Also its configurability is not so good. That is the main reason higher level embedded systems contain microprocessor based architecture. The microprocessor based embedded system having more re-configurabilities in both software and hardware with respect to microcontroller based embedded system.

The microprocessor is an integrated circuit (IC) which is designed using VLSI technique. The architecture of the microprocessor has greater effect on the various performance parameters of the total embedded system. As a result to design any efficient embedded system the processor architecture also is efficient.

In this project we have proposed an efficient VLSI architecture for RISC based microprocessor. The proposed architecture is coded using standard VHDL language and simulated using Xilinx ISE 14.5 tool. The comparison results show that the proposed technique is better with respect to existing techniques.

1.1. LITERATURE SURVEY

Uma [1] presented a new design for 8-bit microprocessors. The presented architecture is implemented on Spartan-3E FPGA board. To code architecture in FPGA the author use standard HDL language. To check the functionality and generate the bit-stream to the corresponding FPGA chip Xilinx ISE tool is used. In this case the author implemented most of the essential instruction set effectively using structural modeling. This allows the author to reduce the hardware requirement. The performance of the architecture is improved due to this reason. Also the implementation cost of the architecture reduces drastically due to this reason. Moreover to increase throughput of the architecture the author use some degrees of pipelining. ShahlaGul et al.

[2] Made a comparison between RISC and CISC microprocessor architectures in this paper in a wide way. Also they have showed that the types of operations are performed by both RISC and CISC architectures. Like CISC architecture divides the total algorithm into a various simplifier instructions which executes step by step. This simplifies the program having smaller architecture but for larger or complex architecture this method is difficult. In such case RISC architecture is helpful. Because the RISC architecture embedded many instruction set. Also in the design prospective the design of RISC architecture is harder than the CISC architecture. This is mainly due to the complexity of the controller which is used to cascade instruction. Mrudul S. Ghaturle et al.

2. PROPOSED 32-BIT RISC ARCHITECTURE

Reduced Instruction Set Computer (RISC) Architecture is very highly optimized architecture to reduce the execution time. The execution time is reduced and the speed of the processor is increased due to the implementation of pipelining and parallelism technique. The pipelining technique produces different outputs at the different unit blocks for every single clock cycle like fetch, decode and execute. Hence the number of clock cycles used by a set of instructions decreases for their execution. In order to make this possible it is mandatory that the instructions should also require a single clock cycle for their execution. Hence, only a few numbers of common and basic instructions are used and complex instructions are performed by the iterative execution of the basic instructions. The block diagram is shown in Fig. 1.

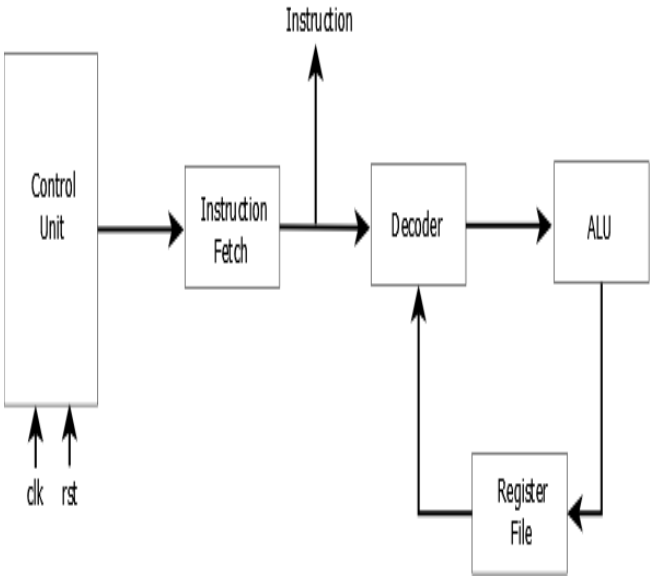

Fig-1: 32-bit RISC Processor Architecture

2.1. CONTROL UNIT

The program to be run by the processor is stored in the Block RAM. These instruction sets are to be processed one at a time for every clock cycle. Therefore control unit comes first in the processing part which plays an important role in feeding the instruction set to be processed with respect to clock signal. There are two main registers, in which one is used to store the instruction set that is to be executed currently and the other one is used to store the next instruction set that is to be processed. So the main role of this block is to generate the address of the instruction that is to be interpreted and fed to the processing. The block diagram is shown in the Fig. 2.

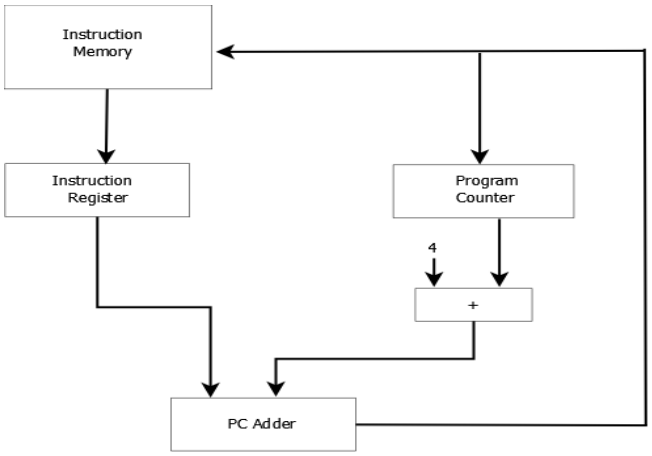

Fig-2: Control Unit

2.2. PROGRAM COUNTER

The Program Counter also called as Instruction Pointer or Instruction Address Register is an important part in the processor to make it work more efficient than the other general processor architectures. The program counter always points to the next instruction to be fetched. For example, if instruction A is being fetched, the program counter will be pointing to the B instruction, which will be in the sequence of the program instruction sets. By this we can tell that the program counter is mainly used for the pipelining concept, further to satisfy the parallelism concept.

Usually the processors fetch instructions in a sequential pattern which is controlled by the program counter but some instructions like branch, jump, subroutine calls, value returns, etc., replace the sequence. The next instruction is fetched from other memory when branch instruction is fetched. The previous program counter contents are saved by the subroutine calls. When the subroutine call is over, the program counter contents are retrieved and the sequential instruction fetching is resumed. The block diagram is shown in the Fig. 3.

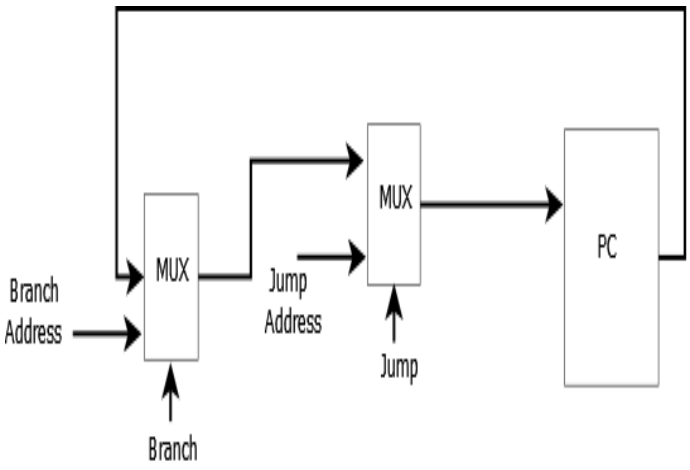

Fig-3: Program Counter

2.3. INSTRUCTION FETCH

Fetch the word means taking the data. The address generated by the program counter is sent to the instruction fetch unit and program counter points to the next instruction incrementing the program counter value by four. The address given to the fetch unit checks in the instruction memory and takes the instruction data set to be decoded.

The instruction set fetched from the memory contains all the data is lightly coded, which is to be decoded to produce the control signals, data path and the data form the fetched instruction bits. For example, the instruction MOV A, B should generate the control signals read to the register A and write to register B and execute a data path between A and B registers. The block diagram is shown in the Fig. 4.

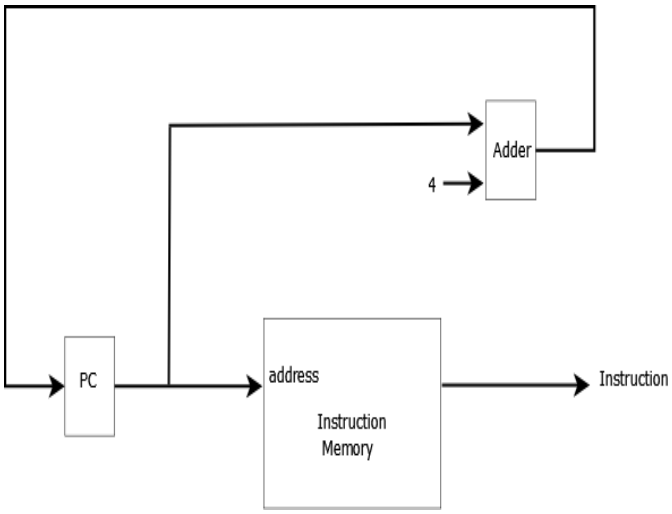

Fig-4: instruction Fetch

2.4. ALU

Arithmetic Logic Unit is the execution unit of arithmetic and logical operations like addition, subtraction, etc. and OR, AND, XOR, etc. Here in this proposed ALU architecture, we have designed a 16-bit multiplication operation, which is designed using Vedic multiplication technique. The operation of the ALU is given in the Table 1.

Table-1: ALU Operation

Opcode	Operation
000	AND
001	OR
010	Addition
011	Subtraction
100	NAND
101	NOR
110	Multiplication
111	No Operation

RESULTS

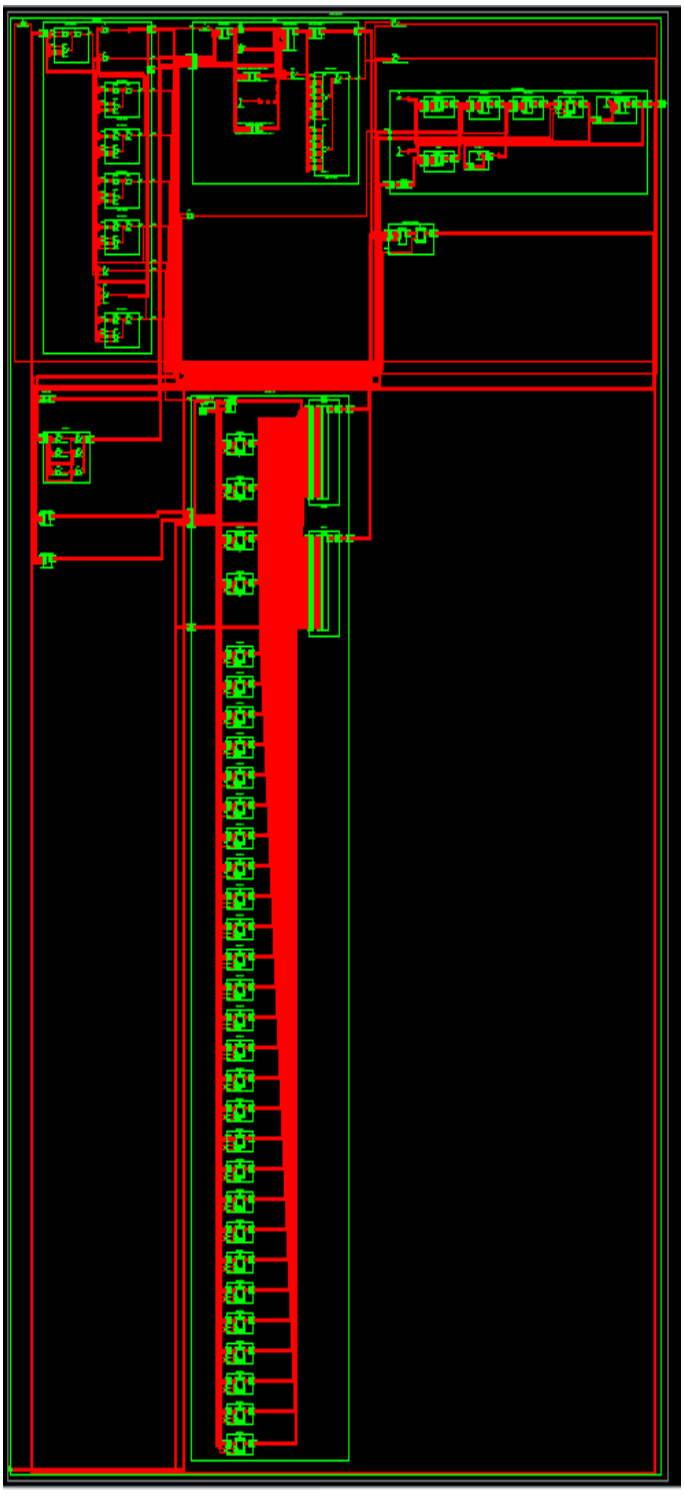

Fig-5: RTL Schematic

The Technology Schematic is shown in the Fig.6.

Fig-6: Technology Schematic

The Simulation Results are shown in the Fig. 7.

Fig-7: Simulation Results

3. CONCLUSIONS

In this paper we have proposed a new efficient architecture of 32-bit RISC microprocessor. The architecture is the modified version of normal multi instruction per cycle architecture. The main advantage of this architecture is that the reduction of the hardware requirement by considering looping architecture. This will allow the user to perform very complex mathematical computations present in most of the real time algorithms. This will allow the user to use PC interface through some specific software packages where all binary opcode converted into some alphabetic opcode.

At the time of designing the whole architecture we have modified each sub blocks manually which reduces the overall hardware requirement of the total architecture and increases the overall frequency. This can be seen in the comparison section.

In this project we designed and tested 32-bit RISC microprocessor manually from FPGA by giving manual inputs. In future we will develop the software interface. Software where we can write some real time algorithm for our designed microprocessor in the similar way we execute normal microprocessor instruction using MASM/TASM software. Moreover in future we will not try to optimize the proposed design up to next extent and also increase the bit size to compare our proposed architecture with really available microprocessors.

REFERENCES

1) Rafael C Gonzalez and Richard E Woods, “Digital Image Processing”, 3rd Edition, Prentice Hall, 2008.

2) Mohamed Nasir Bin Mohamed Shukor, Lo HaiHiung, Patrick Sebastian, “Implementation of Real-time Simple Edge Detection on FPGA”, International Conference on Intelligent and Advanced Systems, 2007.

3) Zhengyang Guo Wenbo Xu and Zhilei Chai, “Image Edge Detection Based on FPGA”, Ninth International Symposium on Distributed Computing and Applications to Business Engineering and Science, 2010.

4) FerdousHossain, Mithun Kumar P.K. and Mohammad Abu Yousuf, “Hardware Design and Implementation of Adaptive Canny Edge Detection Algorithm”, International Journal of Computer Applications, Vol. 128, No. 9, pp. 31-38, 2015.

5) T. Sridevi B. Poornima and Y. Ramadevi, “Threshold Based Edge Detection Algorithm”, International Journal of Computer Science & Information Technology, Vol. 2, No. 6, pp. 153-161, 2010.

6) Mina Asaduzzaman, Md. Armanur Rahman, Mohammad Abu Yousuf and Ferdous Hossain, "Dynamic Thresholding based Adaptive Canny Edge Detection", International Journal of Computer Applications, Vol. 135, No. 4, pp. 37-41, 2016.

7) Neethu P. R, "Cancer cell detection using distributed canny edge detector", International Research

8) Journal of Engineering and Technology, Vol. 2, pp. 1224-1226, 2015.

9) R. Tourki, T. Saidani, M. Atri, D. Dia and W. Elhamzi, "Hardware Co-simulation For Video Processing Using Xilinx System Generator", Proceedings of the World Congress on Engineering, 2009.

10) WeibinRong, Zhanjing Li, Wei Zhang and Lining Sun, "An Improved Canny Edge Detection Algorithm", International Conference on Mechatronics and Automation, pp.577-782, 2014.