Hardware Implementation High Speed RISC Processor for Convolution

Mr. Prashant Bhirange
Department of Electronics Engineering, S.D.C.O College of Engineering, Wardha
Maharashtra, India
prashantbhirange@gmail.com

Abstract-Many algorithms have been design in order to accomplish an improved the performance of the filters by using the convolution design. The architecture of the proposed RISC CPU is a uniform 32-bit instruction format, single cycle non-pipelined processor. It has load/store architecture, where the operations will only be performed on registers, and not on memory locations. It follows the classical von-Neumann architecture with just one common memory bus for both instructions and data. A total of 27 instructions are designed in initial development step of the processor. The instruction set consists of Logical, Immediate, Jump, Load, store and HALT type of instruction. The combined advantages RISC processor such as high speed, low power, area efficient and operation-specific design possibilities have been analyzed.

In this paper we have implemented 32 bit RISC processor to perform circular convolution at different modules of RISC processor like execute unit along with ALU, Instruction fetch along with instruction memory, decode unit, resistor unit, data memory has been implemented.

The execution time to perform 4 bit circular convolution is found to be 270 ns. The execution time to execute the instruction is found to be 5 ns. In our paper, 3200 LUTs and 320 logic elements are used to implement 32 bit RISC processor which is found to be area efficient as compared to various designs.

Keywords- *Circular Convolution, RISC Processor.*

*****

I. INTRODUCTION

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 sparcstation the other vendors rushed to compete with RISC based solutions of their own. Even the mainframe world is now completely RISC based The reasons have been the advantages, such as its simple, flexible and fixed instruction format and hardwired control logic, which paves for higher clock speed, by eliminating the need for microprogramming. The combined advantages of high speed, low power, area efficient and operation-specific design possibilities have made the RISC processor universal [9].

Convolution is one of the basic and most common operations in both analog and digital domain signal processing. Often times, it is desirable to modulate a given signal in order to enhance or extract important information contained in it. This signal modulation is generally known as filtering[9].

II. LITERATURE REVIEW

Reference [1] proposed a design methodology of a single clock cycle MIPS RISC Processor using VHDL to ease the description, verification, simulation and hardware realization. The RISC processor has fixed-length of 32-bit instructions based on three different format R-format, I-format and J-format, and 32-bit general-purpose registers with memory word of 32-bit. The MIPS 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. All the modules in the design are coded in VHDL, as it is very useful tool with its concept of concurrency to cope with the parallelism of digital hardware. 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 & performance of the designated design that demonstrate the effectiveness of the design.

Reference [2] proposed processor which has Harvard architecture and consists of 24-bit address, 5-stage pipeline instruction execution, and internal debug logic. ADPCM vocoder and SOLA algorithm are successfully carried out on the processor made with FPGA.

Reference [3] works on design and verification of a 32-bit general purpose microprocessor which is compatible with ARM7 RISC Core, is described. In the architectural point of view, the processor has 3-stage pipeline, 6 register banks, 32-bit ALU, and 4-cycle MAC. The core described here was designed by latch base for low power and low complexity. Its functional operation was verified by comparison the results of logic simulation with those of the commercial simulator.

Each instruction and its random combinations were tested. The designed core was emulated to check its proper operation for various applications, such as ADPCM, SOLA (voice speed variation), MP3 decoding. Finally it was implemented in 0.5µm CMOS process and it carried out successfully those algorithms.

Reference [4] has proposed a 32 bit RISC Processor for embedded application is presented. With respect to limitation of power and area in embedded system, RISC Processor is deliberately designed.

Dual-issue technology is adopted to improve the performance; the complex logic of the dynamic scheduling algorithm is allocated into different pipeline stage to improve the frequency.

Lower power design method is used to decrease the whole power. The processor is implemented by SMIC 0.18 um CMOS technology. It contains almost 5 million transistors; the core frequency is 266 MHz and the power is about 1.3 w under it. The embedded VxWorks OS can run on it stably. The performance analysis of the RISC processor is also provided. According to the embedded benchmark program, the average IPC of the RISC processor is nearly 1.5.

Reference [5] has described the architecture and design of the pipelined execution unit of a 32-bit RISC processor. Organization of the blocks in different stages of pipeline is done in such a way that pipeline can be clocked at high frequency. Control and forward of `data flow' among the stages are taken care by dedicated hardware logic. Different blocks of the execution unit and dependency among themselves are explained in details with the help of relevant block diagrams. The design has been modeled in verilog HDL and functional verification policies adopted for it have been described thoroughly. Synthesis of the design is carried out at 0.13-micron standard cell technology and for slow timing library the reported frequency of operation is 714 MHz at synthesis level.

Reference [6] proposed the architecture and implementation of a programmable video signal processor dedicated as building block of a multiple instruction multiple data (MIMD)-based bus-connected multiprocessor system is presented. This system can either be constructed from several single processor chips, or it can be integrated on a large area integrated circuit containing several processors.

The processor allows an efficient implementation of different video coding standards like H.261, H.263, and MPEG-1&MPEG-2.

It consists of a RISC processor supplemented by a coprocessor for computation intensive convolution-like tasks, which provides a peak performance of more than 1 giga-arithmetic operations per second (GOPS). The large area integrated circuit integrating 9 processor elements (PE's) on an area of 16.6 cm/sups2 has been designed. Due to yield considerations redundancy concepts have been implemented, that-even in the presence of production defects-result in working chips utilizing a lower number of PE's. Each PE has built-in self-test (BIST) capabilities, which allow for an independent test of itself under the control of its integrated fault-tolerant BIST controller. Defective PE's are switched off. Only the PE's passing the BIST is used for video processing tasks. Prototypes have been fabricated in a 0.8 /spl mu/m complementary metal-oxide-semiconductor (CMOS) process structured by masks using wafer stepping with overlapping exposures. Employing redundancy, up to 6 PE's per chip were functional at 66 MHz, thus providing a peak arithmetic performance of up to 6 GOPS.

Reference [7] presents a parallel MAC (multiply-accumulation) architecture is designed for DSP applications on a 200Mhz, 1.6-GOPS Multimedia RISC Processor. The data path architecture of the processor is designed to realize parallel execution of a data transfer and SIMD parallel arithmetic operations. SIMD parallel 16-bit MAC instructions are introduced with a symmetric rounding scheme which maximizes the accuracy of the 18-bit accumulation. This parallel 16-bit MAC instruction on a 64-bit data path is shown to be efficiently utilized for DSP applications such as convolution in the multimedia RISC processor. By using the parallel MAC instruction with the symmetric rounding scheme, the two-dimensional inverse discrete cosine transform (2D-IDCT) which satisfies IEEE 1180 can be implemented in 202 cycles.

Reference [8]discusses a VLSI based multiprocessor architecture for real time processing of video coding applications. The architecture consists of multiple identical processing elements and is characterized as MIMD (multiple instruction multiple data). The architecture of a processing element is based on a standard processor core, e.g., a RISC processor, and a low-level coprocessor. The low-level coprocessor is adapted to parallel processing of convolution like operations. The performance of the architecture is discussed with respect to the processing time for hybrid coding algorithms as well as to the required silicon area.

In this work we are implemented high speed, area efficient 32 bit RISC processor to perform circular convolution on an FPGA.Implementation of processor that consists of several blocks such as execute unit along with ALU, instruction fetch along with instruction memory, decoding logic, control unit, data memory and program memory. Our objective is to designed processor architecture for RISC type of instruction set with one cycle operation which is modified for convolution operation.

The rest of the paper is organized as follows. Section III Present design of the RISC CPU. Section IV present design RISC Processor For circular convolution. Section V gives Simulation result of design RISC Processor for circular convolution. Section VI gives comparison result with other references section. Sections VII conclude.

III. DESIGN OF 32 BIT RISC CPU

A. Architecture

The architecture of the proposed RISC CPU is a uniform 16-bit instruction format, single cycle non-pipelined processor shown in figure 1. It has a load/store architecture, where the operations will only be performed on registers, and not on memory locations. It follows the classical von-Neumann architecture with just one common memory bus for both instructions and data. A total of 27 instructions are designed as a first step in the process of development of the processor. The instruction set consists of Logical, Immediate, Jump, Load, store and HALT type of instructions [9].

B. Detail of Logical Blocks

Figure 1 illustrates the block diagram of the 32-bit RISC CPU. The proposed RISC CPU consists of five blocks, namely, Arithmetic and Logical Unit (ALU), Program Counter (PC), Register file (REG), Instruction Decoder Unit (IDU) and Clock Control Unit (CCU). The data-path of the proposed CPU in Fig. 1 is explained as follows [9].

1) Program Counter:

The Program Counter (PC) is a 32-bit latch that holds the memory address of location, from which the next machine language instruction will be fetched by the processor. It is a 6-bit pointer to indicate the instruction memory. It additionally uses a 6-bit pointer to point to the data memory, which will be

used only when a Load/Store instruction is encountered for execution.

2) Arithmetic and Logic unit:

The arithmetic and logic unit (ALU) performs arithmetic and logic operations. It also performs the bit operations such as rotate and shift by a defined number of bit positions. The proposed ALU contains three sub-modules, viz. arithmetic, logic and shift modules. The arithmetic unit involves the execution of addition and multiplication operations and generates Sign flag and Zero flag. The shift module is used for executing instructions such as rotation and shift operations[9].

3) Register File:

The register file consists of 8 general purpose registers of 32-bits capacity each. These register files are utilized during the execution of arithmetic and data-centric instructions. The load instruction is used to load the values into the registers and store instruction is used to retrieve the values back to the memory to obtain the processed outputs back from the processor.

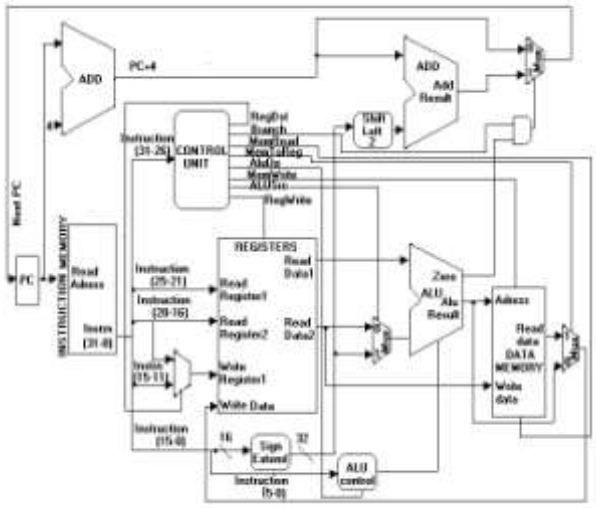

Fig. 1: Processor Architecture

4) Instruction fetch unit

The function of the instruction fetch unit is to obtain an instruction from the instruction memory using the current value of the PC and increment the PC value for the next instruction as shown in Figure 1.1 Since this design uses an 8-bit data width we had to implement byte addressing to access the registers and word address to access the instruction memory.

The instruction fetch component contains the following logic elements that are implemented in VHDL: 8-bit program counter (PC) register, an adder to increment the PC by four, the instruction memory, a multiplexor, and an AND gate used to select the value of the next PC. Appendix C contains the VHDL code used to create the instruction fetch unit of the MIPS single-cycle processor [10].

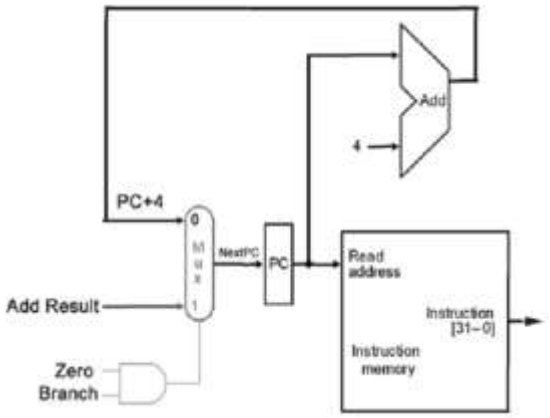

Fig 1.1 MIPS Instruction Fetch Unit (Source [10])

5) Instruction decode unit

The main function of the instruction decode unit is to use the 32-bit instruction provided from the previous instruction fetch unit to index the register file and obtain the register data values as seen in Figure 1.2 This unit also sign extends instruction bits [15-0] to 32-bit. However with our design of 8-bit data width, our implementation uses the instruction bits [7–0] bits instead of sign extending the value. The logic elements to be implemented in VHDL include several multiplexors and the register file[10].

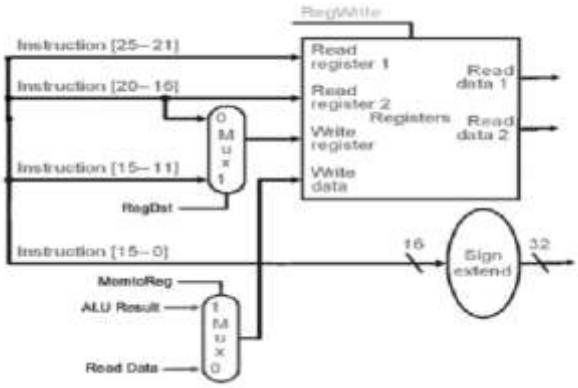

Fig 1.2 MIPS Instruction Decode Unit (Source [10])

6) The control unit

The control unit of the MIPS single-cycle processor examines the instruction opcode bits [31–26] and decodes the instruction to generate nine control signals to be used in the additional modules as shown in Figure 1.3. The RegDst control signal determines which register is written to the register file. The Jump control signal selects the jump address to be sent to the PC.

The Branch control signal is used to select the branch address to be sent to the PC. The MemRead control signal is asserted during a load instruction when the data memory is read to load a register with its memory contents. The MemtoReg control signal determines if the ALU result or the data memory output is written to the register file. The ALUOp control signals determine the function the ALU performs. (e.g. and, or, add, sbu, slt) The MemWrite control signal is asserted when during a store instruction when a registers value is stored in the data memory. The ALUSrc control signal determines if the ALU second operand comes from the register file or the sign extend [10].

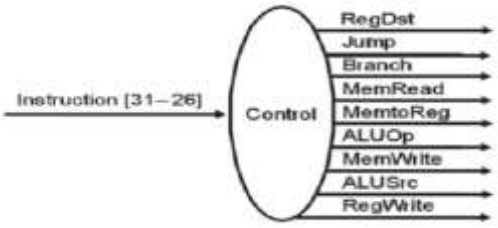

Fig. 1.3: MIPS Control Unit

7) Execution unit

The execution unit of the MIPS processor contains the arithmetic logic unit (ALU) which performs the operation determined by the ALUop signal. The branch address is calculated by adding the PC+4 to the sign extended immediate field shifted left 2 bits by a separate adder. The logic elements to be implemented in VHDL include a multiplexor, an adder, the ALU and the ALU control as shown in Figure 1.4[10].

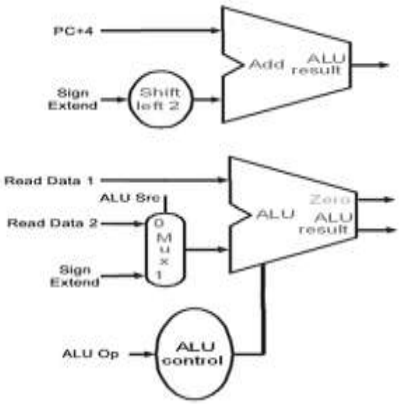

Fig 1.4: MIPS execution unit (Source [10])

8) Data Memory Unit

The data memory unit is only accessed by the load and store instructions. The load instruction asserts the MemRead signal and uses the ALU Result value as an address to index the data memory. The read output data is then subsequently written into the register file. A store instruction asserts the MemWrite signal and writes the data value previously read from a register into the computed memory-address [10].

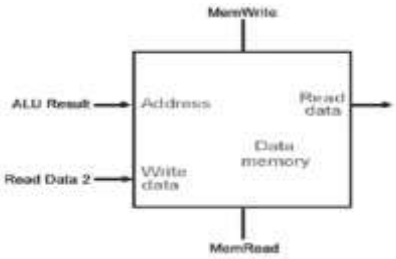

Fig 1.5 MIPS Data Memory unit

IV. IMPLEMENTATION OF CIRCULAR CONVOLUTION

Unlike the FPGA implementation care must be taken about the edge of a frame. Min and max operations are used to ensure that a memory address doesn't exceed the edges of a frame hence leave the address range. However one can execute different portions of the code depending on current pixel locations, to reduce this overhead. The general structure is again similar to that for the FPGA.

For actual implementation we have gone through following three processes.

1. Rotate and Shifting

In this process the data stream is shifted by the shift instruction from the processor. The data stream is shifted from one digit as shown in following matrix.

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 2 \\ 2 & 1 & 1 & 0 \\ 0 & 2 & 1 & 1 \\ 1 & 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

Let $x_1(n) = \{2,3,4,5\}$ and $h_1(n) = \{1,2,0,1\}$
Then according to the formula of circular convolution

$$f(n) = \sum_{n=0}^{N-1} x_1(n).h_1(n-k)$$

2. Multiplying

In this process the data is multiplied by the multiply instruction from the processor. The data is multiplied by each row to column from the above matrix. That is

$$\begin{bmatrix} 1 & 1 & 0 & 2 \\ 2 & 1 & 1 & 0 \\ 0 & 2 & 1 & 1 \\ 1 & 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 1*2+1*3+0*4+2*5 \\ 2*2+1*3+1*4+0*5 \\ 0*2+2*3+1*4+1*5 \\ 1*2+0*3+2*4+1*5 \end{bmatrix} \text{ ---------- (1)}$$

$$\begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix} = \begin{bmatrix} 15 \\ 11 \\ 15 \\ 15 \end{bmatrix}$$

3. Adding

In this process the data is added bit by bit by the add instruction from the processor. The data is adding as shown in the above matrix (1). Result is shown from the simulation of circular convolution process.

IV. Simulation result of design risc processor for circulation convolution.

The RISC processor architecture was simulated on Altera Quartus-II Version 7.0. The simulation results are found as following

I. RTL VIEWS AND SIMULATION RESULTS OF RISC PROCESSOR

1) Control unit

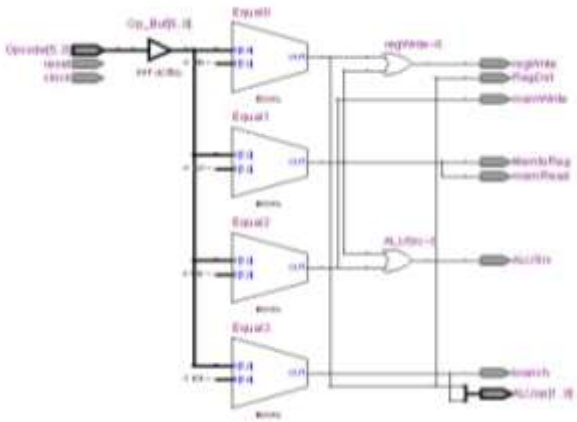

Fig 2.1 RTL view of control unit

ISSN: 2321-8169
471 - 476

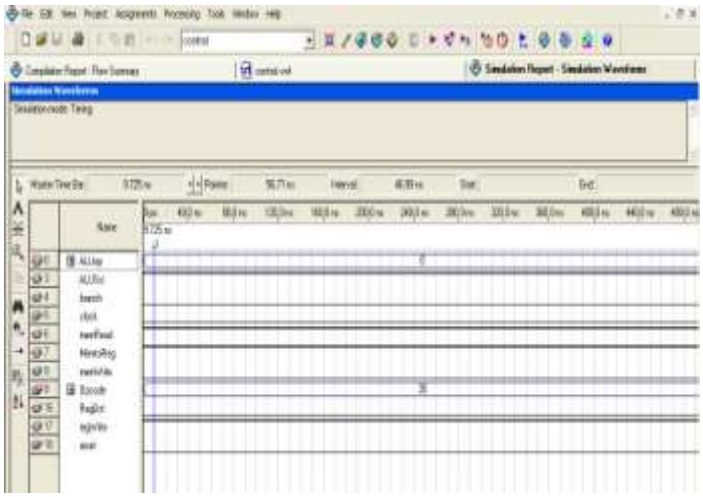

Fig 2.2 Simulation of control unit

2) Data memory

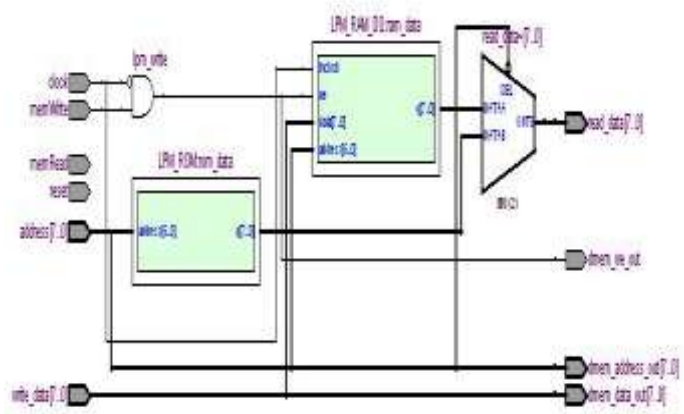

Fig 2.3RTL view data memory unit

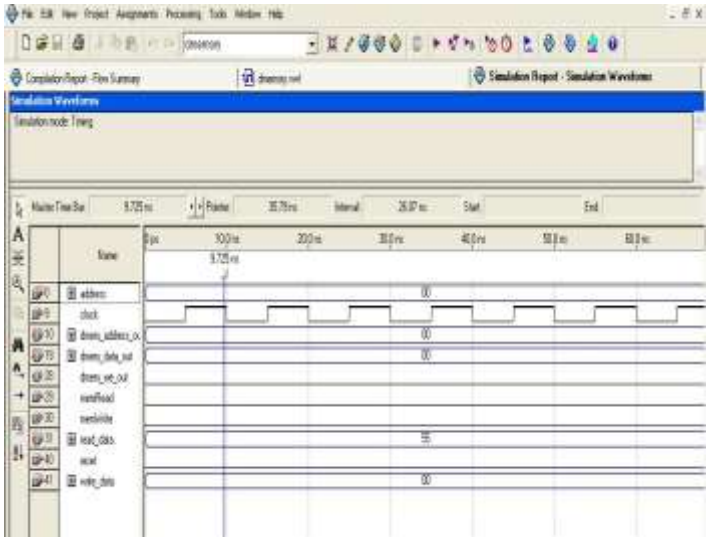

Fig 2.4 Simulation of data memory unit

3) Instruction unit

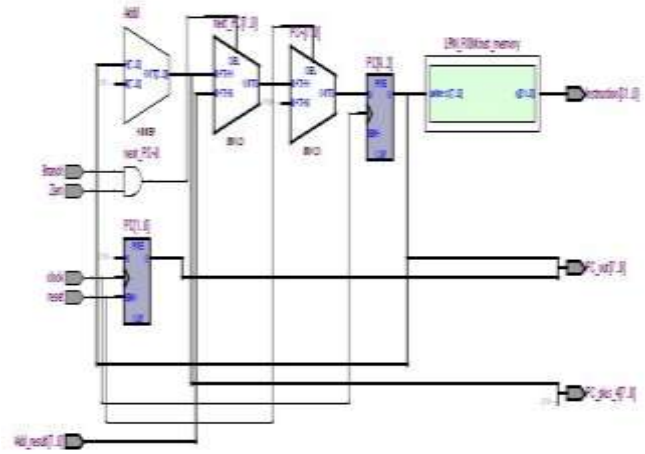

Fig 2.5 RTL view Instruction fetch unit

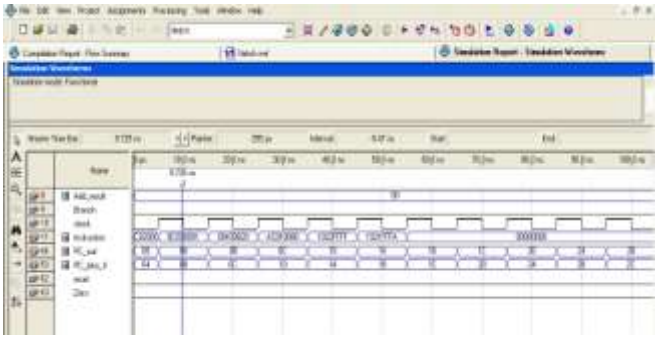

Fig 2.6 Simulation of Instruction Fetch unit

V. *SIMULATION RESULTS OF RISC PROCESSOR WITH CONVOLUTION*

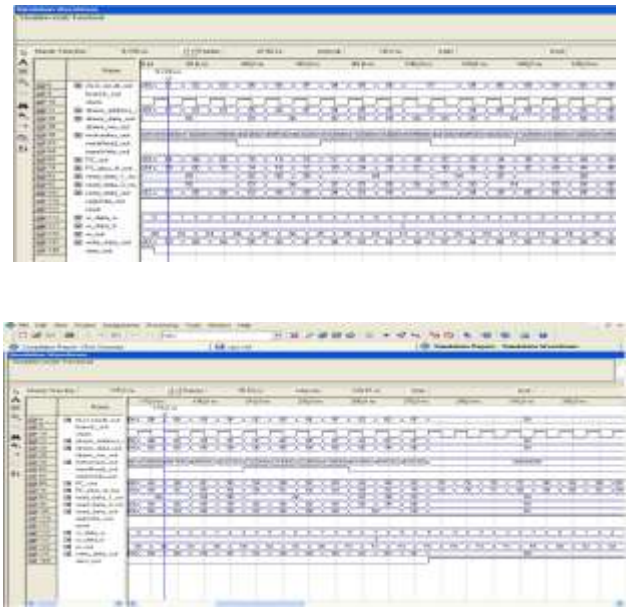

Fig 2.7 Simulation *results of RISC PROCESSOR with Convolution*

VI. COMPARISON TABLE

Table I: Comparison of referred paper with our work done

Parameters	**Ref. [1]**	**Ref. [2]**	**Ref. [3]**	**Ref. [4]**	**Ref. [5]**	**Our work**
Implementation	FPGA	FPGA	FPGA	FPGA	VLSI	FPGA
Frequency	200 MHz	50 MHz	50 MHz	69 MHz	16 MHz	250 MHz
Area	-	-	78 CLBs and 6 DSP elements	2234 CLBs 1164 FFs 3829 LUTs	65012 nm^2	3200 LUTs 320 Total Logic Elements

VII. CONCLUSION

we have implemented 32 bit RISC processor to perform circular convolution at different modules of RISC processor like execute unit along with ALU, Instruction fetch along with instruction memory, decode unit, resistor unit, data memory has been implemented. We have simulated the instructions of 32 bit RISC processor. We have also simulated the result of circular convolution on 32 bit RISC processor.

The execution time to perform 4 bit circular convolution is found to be 270 ns. The execution time to execute the instruction is found to be 5 ns.

In our work, 3200 LUTs and 320 logic elements are used to implement 32 bit RISC processor which is found to be area efficient as compared to various designs given in the literature.

Future scope

In the future 32 bit RISC processor can be used to perform 2D linear and circular convolution which is widely used for filtering in digital image processing (DIP).

Acknowledgment

We place our gratitude on record to the Department of Electronics and Communication Engineering, SD College of Engineering, wardha for the support rendered to us in carrying out this work.

References

[1] Reaz, M.B.I., Islam, M.S.; Sulaiman, M.S.; "A single clock cycle MIPS RISC processor design using VHDL", Semiconductor Electronics, 2002. Proceedings. ICSE 2002. IEEE International Conference. ISBN: 0-7803-7578-5; Issue Date: 19-21 Dec. 2002

[2] Seung Pyo Jung; Jingzhe Xu; Donghoon Lee; Ju Sung Park; Kang-joo Kim; Koon-shik Cho; "Design & verification of 16 bit RISC processor", SoC Design Conference, 2008. ISOCC '08. International; ISBN: 978-1-4244-2598-3; Issue Date: 24-25 Nov. 2008

[3] Geun-young Jeong; Ju-sung Park; Hyun-woo Jo; Byung-woo Yoon; Myung-jin Lee; "ARM7 compatible 32-bit RISC processor design and verification", Science and Technology, 2005. KORUS 2005. Proceedings. The 9th Russian-Korean International Symposium on; July 2005.

[4] Xiaoping Huang; Xiaoya Fan; Shengbing Zhang, Solid-State and Integrated-Circuit Technology, 2008. ICSICT 2008. 9th International Conference.

[5] Shofiqul Islam; Debanjan Chattopadhyay; Manoja Kumar Das; V Neelima; Rahul Sarkar; "Design of High-Speed-Pipelined Execution Unit of 32-bit RISC Processor"; India Conference, 2006 Annual IEEE.

[6] Herrmann, K.; Otterstedt, J.; Jeschke, H.; Kuboschek, M. "A MIMD-based video signal processing architecture suitable for large area integration and a 16.6-cm/sup 2/ monolithic implementation", Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, Volume: 6 , Issue: 2.

[7] Kuroda, I.; Murata, E.; Nadehara, K.; Suzuki, K.; Arai, T.; Okamura, A.; "A 16-bit parallel MAC architecture for a multimedia RISC processor", Signal Processing Systems, 1998. SIPS 98. 1998 IEEE Workshop on, ISBN: 0-7803-4997-0, Date: 8-10 Oct 1998.

[8] Jeschke, H.; "A VLSI based multiprocessor architecture for video signal processing", Circuits and Systems, 1992. ISCAS '92. Proceedings., 1992 IEEE International Symposium on, ISBN: 0-7803-0593-0, Date : 3-6 May 1992

[9] Samiappa Sakthikumaran, et al, "16-Bit RISC Processor Design for Convolution Application", in IEEE-International Conference on Recent Trends in Information Technology, (ICRTIT 2011), MIT, Anna University, Chennai. June 3-5, 2011