*International Journal of Engineering Research and Development*
*e-ISSN : 2278-067X, p-ISSN : 2278-800X, www.ijerd.com*
*Volume 2, Issue 6 (August 2012), PP. 51-55*

# Processor Realization for Application of Convolution

Prashant D Bhirange[1], V. G. Nasre[2], M. A. Gaikwad[3]
[1]*Department of Electronics Engineering, Acharya Shrimannarayan Polytechnic, Wardha.*
[2]*P. G. Department of Electronics Engineering, B. D. College of Engineering, Sevagram, Wardha.*
[3]*Dean R & D, B. D. College of Engineering, Sevagram, Wardha.*

***Abstract—Convolution is very important in signal and image processing applications which is used in filter designing. Many algorithms have been proposed 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.***

***Keywords—CISC, RISC, Convolution, Wallace Tree Multiplier***

## I. INTRODUCTION

The trend in the past shows the RISC processors clearly outsmarting the earlier CISC processor architectures. The reasons have been the advantages, such as simplicity, flexibility. paves for higher clock speed, by eliminating the need for microprogramming through fixed instruction format and hardwired control logic. The combined advantages of high speed, low power, area efficient and operation-specific design possibilities have made the RISC processor universal. The main feature of the RISC processor is its ability to support single cycle operation, meaning that the instruction is fetched from the instruction memory at the maximum speed from the memory. RISC processors are designed to achieve this by pipelining, where there is a possibility of stalling of clock cycles due to wrong instruction fetch when jump type instructions are encountered. This reduces the efficiency of the processors. This paper describes a RISC architecture in which, single cycle operation is obtained without using a pipelined design. It averts possible stalling of clock cycles in effect. The development of CMOS technology provides very high density and high performance integrated circuits. The performance provided by the existing devices has created a never-ending greed for increasingly better performing devices. This predicts the use of a whole RISC processor as a basic device by the year 2020. However, as the density of IC increases, the power consumption becomes a major threatening issue along with the complexity of the circuits. Hence, it becomes necessary to implement less complex, low power processor designs. In order to employ the processor for signal processing applications, a modified Wallace tree multiplier that uses compressor circuits to achieve low power, high speed operation, in the ALU [1]. Literature suggests that it is possible to achieve the high speed, low power and area efficient operations by reducing the stronger operations such as multiplication, at the cost of increasing the weaker operations such as addition. Convolution is an important signal processing application used in filter design. Many algorithms have been proposed in order to achieve an optimized performance of the filters by optimizing the convolution design. Modified Winograd algorithm is notable among them. They require 4 multiplication operations only, when compared to 6 for a 3*2 normal convolution methodology with an additional rise in the number of adders to 9 from 4. These operations are expensive computation, and perform sluggishly when implemented on microprocessors. Part of the poor performance is due to the serial nature of microprocessors, while the operations of convolution and correlation are inherently parallel. One approach to implementing these operations in parallel is to build them in hardware using application specific integrated circuits (ASICs). Another approach is to use Field Programmable Gate Arrays (FPGAs) and reconfigurable computing [2]. In order to seek an alternative design that allows the rapid development of real time image processing systems, an unified hardware architecture for some image filtering algorithms in space domain, such as windowing-based operations, which are implemented on FPGAs (Field Programmable Gate Arrays) [3]. For achieving this, six different filters have been implemented in a parallel approach, separating them in simple hardware structures, allowing the algorithms to explore their parallel capabilities by using a simple systolic architecture. Both image processing and synthesis results have demonstrated the feasibility of FPGAs for implementing the proposed filtering algorithms in a full parallel approach. Literature presents a high-speed 2-dimensional hardware convolution architecture based on VLSl systolic arrays for image processing applications. Architecture of "Parallel processing by the 2-D convolution". First a VLSl convolution chip was designed to accommodate various convolution window sizes. Three number coefficients computing elements (processors) of are used in such processors to develop on one VLSl chip. Signed coefficients and unsigned data of 8-bits are supported. All processing and inter-processor communications are performed bit-serially [5]. Convolution is an important signal processing application which is used in filter design. Convolution operation is also applied in image processing application such as spatial filters for edge detection, blurring and sharpening. Correlation operation can also be implemented using convolution operation.

## II. DESIGN OF 32-BIT RISC CPU FOR CONVOLUTION OPERATION

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 [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. 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. 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. Instruction Decoder Unit: The decoder units decodes the instruction and gives out the 3-bit source and destination addresses respectively, depending on the op-code's operation and it also decides whether the write back circuit has to be enabled or not. instructions, such as, MOV, AND, OR, XOR, ADD, SUB, SL (Shift Left), RL (Rotate Right), SR (Shift Right), RR (Right Rotate), SWAP and Multiply instructions. LHI (Load 8-bit value into the eight higher significant bits of the given register), LLI (Load 8-bit value into given register's 8 least significant bits), ANDI, ORI, XORI, ADDI, SUBI, instruction for JUMP, JZ (Jump if Zero), JNZ (Jump if not Zero), JP (Jump if positive), JN (Jump if Negative) instructions. Instruction format is shown in figure 2. Clock Control Unit: Efficient phase scheduling and clock gating is required to optimize the throughput and the energy consumption of the processor.

Features:

- RISC Controller
- Low Power and High Speed
- 32 bit processor
- 32 bit address lines
- 32 x 8 bit register bank
- Internal program memory to store program
- Data memory
- 32 bit ALU
- Instruction decoder and control unit (instruction set).

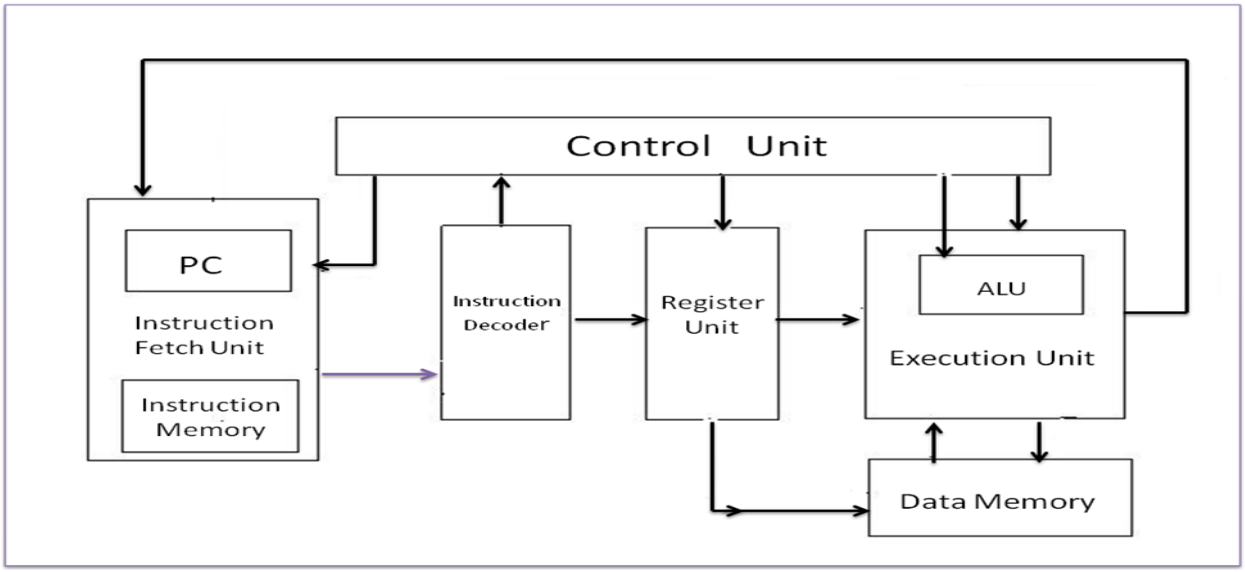


***Figure 1*** Block diagram of RISC Processor

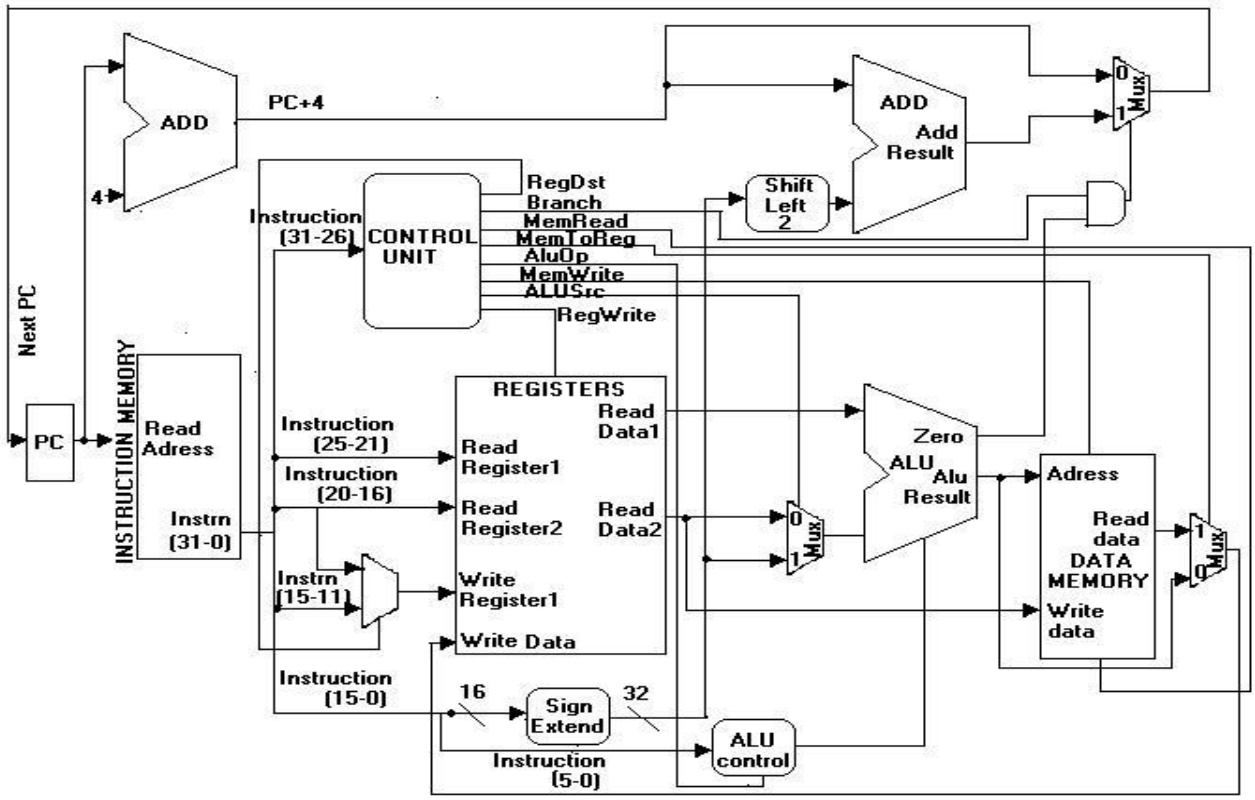


***Figure 2:*** Processor Architecture

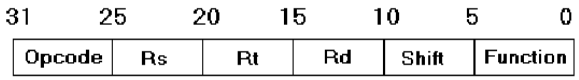


***Figure 3:*** R-Format

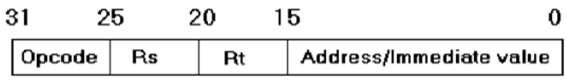


***Figure 4:*** I-Format

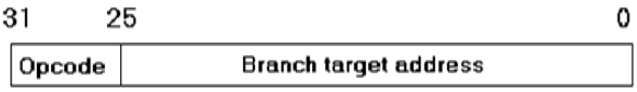


***Figure 5***: J-Format

## III. IMPLEMENTATION

Tools for Synthesis & Simulations Xilinx (VHDL) and ModelSim can be used & controller can be implemented. Xilinx ISE Design Tool is the ideal downloadable solution for FPGA and CPLD design offering HDL synthesis and simulation, implementation, device fitting, and JTAG programming. Following are the steps of designing digital circuits in FPGA using VHDL.

1. Use a schematic or a Hardware Description Language (HDL) to design the logic block.
2. Use the tool to synthesis the logic block.
3. Use Timing Analyzer to find and optimize the critical path(s).
4. Use Floor planner and FPGA Editor to optimize the area and routing length of the block.
5. Use FPGA Editor to map the function block into a hard macro.
6. Delay-matching hard macro, which has a comparative delay and dimension with the function block.

## IV. SIMULATION RESULTS

The simulation results of proposed processor are shown below:

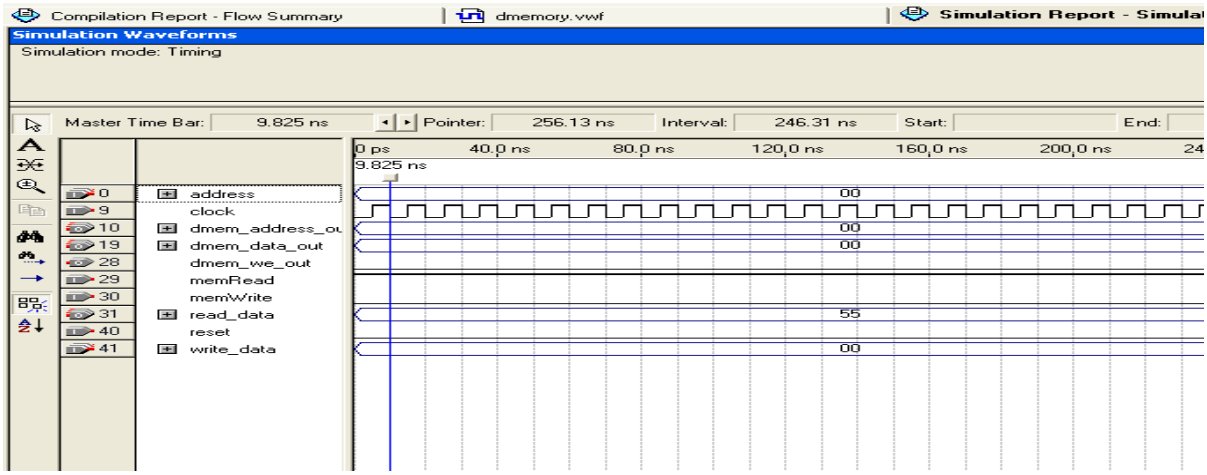


***Figure 6*:** Simulation of Data-Memory unit (opcode 00)

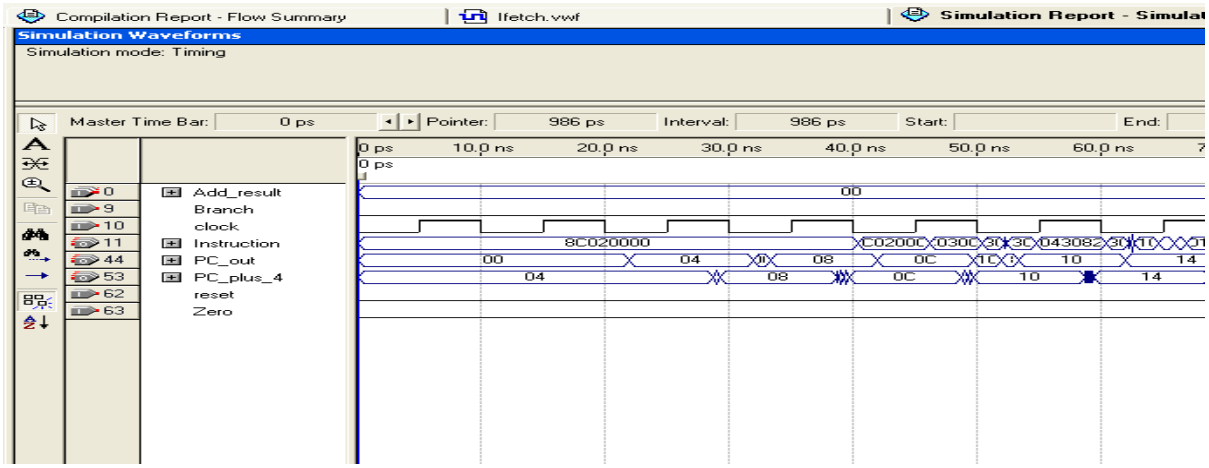


***Figure 7*:** Simulation of Ifetch Unit Opcode(00)

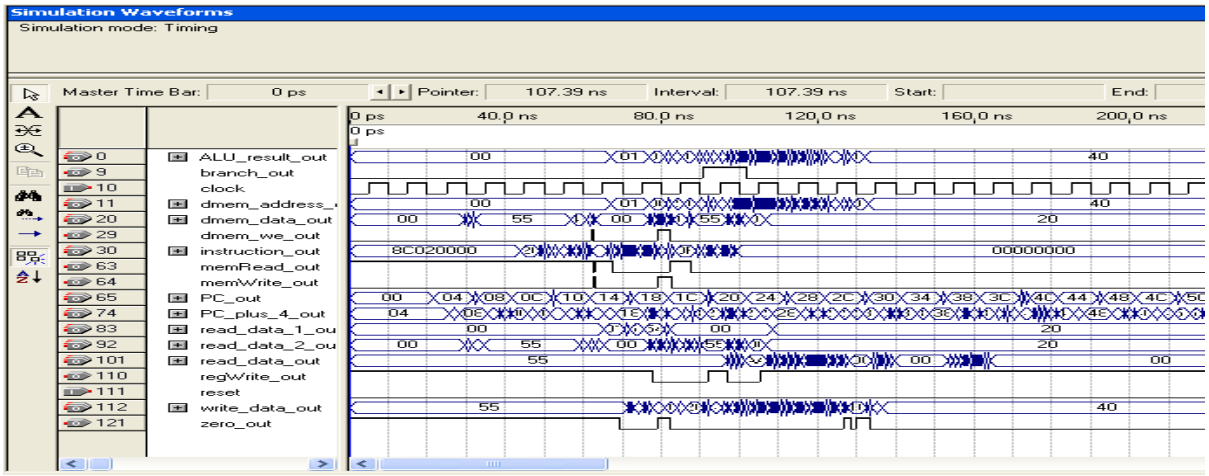


***Figure 8*:** Simulation Result of processor

## V. CONCLUSION

The realization of processor includes several blocks such as ALU, instruction fetch and decoding logic, control unit, data memory and program memory. The proposed architecture of processor is suitable for convolution operation that can be used for signal and image processing algorithms. Convolution is an important in signal processing application which is used in filter design. Convolution operation can also be applied in image processing application such as spatial filters for

edge detection, blurring and sharpening. Using convolution operation Correlation operation can also be implemented. We can apply a clock-gating scheme in order to achieve low power consumption.

## REFERENCE


[1]. 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.

[2]. Sebastien C. Wong, et al, "Fast 2D Convolution using Reconfigurable Computing," in IEEE International Conference on Signal Processing, 2005.

[3]. Jones Y. Mori, et al, "An Unified Approach for Convolution based Image Filtering on Reconfigurable Systems," in IEEE Workshop on Signal Processing Systems, 2011.

[4]. Ming Z. Zhang, et al, "An efficient multiplier-less architecture for 2-D convolution with quadrant symmetric kernels," in Elsevier Integration, the VLSI Journal, vol. 40, 2007, pp 490 – 502.

[5]. D.D. Haule, et al, "High-speed 2-D Hardware Convolution Architecture Based on VLSl Systolic Arrays," in IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, June 1st - 2nd, 1989.

[6]. D. L. Perry, " VHDL", Tata Mcgraw Hill Edition, 4th Edition, 2002.

[7]. C. Maxfiled, "The Design Warriors Guide to FPGAs", Elsevier, 2004.