2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA), Benghazi, Libya 21-23 May 2023.

Design and Implementation of Single Precision Floating-point Arithmetic Logic Unit for RISC Processor on FPGA

1st Noor Alhuda Saad Adela
Department of Computer Engineering
University of Tripoli
Tripoli, Libya
n.adela@uot.edu.ly

2nd Amani Najeeb Ben Yousuf
Department of Computer Engineering
University of Tripoli
Tripoli, Libya
a.yousuf@uot.edu.ly

3rd Mohamed Muftah Eljhani
Department of Computer Engineering
University of Tripoli
Tripoli, Libya
m.eljhani@uot.edu.ly

2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA) | 979-8-3503-1989-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/MI-STA57575.2023.10169623

***Abstract*— The main purpose of conducting this research is to design and implement a single precision floating-point arithmetic logic unit (ALU) that considered as a part of the math coprocessor. The main advantage of floating-point representation is that it can support more values than fixed-point and integer representations. Summation, Subtraction, multiplication and division are arithmetic functions in these calculations. In this floating-point unit, input must be provided in IEEE-754 format, which is 32 single precision floating point values. The application of This arithmetic unit is located in the math coprocessor. Commonly referred to as reduced instruction set computation (RISC) processor. In this processor, for a signal processing, a value with high accuracy is required and as it is an iterative process, the calculation should be as fast as possible. A fixed-point and integer central processing unit (CPU) can't meet the requirements. The floating-point representation can calculate very large or very small process quickly and accurately. The system designed, verified and implemented with Verilog hardware description language using Intel Altera software tools.**

Keywords—: Floating point, IEEE-754, ALU, FPGA Design, Verilog Hardware Description Language

I. INTRODUCTION

RISC is a microprocessor design approach that emphasizes simplicity and ease of use. While it may not be as efficient for complex calculations that require a single, complicated instruction to execute, it excels at tasks that can be broken down into multiple simple operations. This makes RISC processors ideal for use in pipelining applications[1] [2]. With the increasing complexity of devices, integrated microprocessors must provide high performance, while still maintaining low power consumption and a small form factor. This has become essential for daily activities, as computers and mobile phones have become indispensable tools. However, with the growing complexity of these products, there is a greater need for processing power while still maintaining battery life. Recent trends have shown that RISC processors have overtaken CISC processor architecture. The advantage of RISC is that it can execute instructions quickly due to the simplicity of the instructions [3]. A RISC microprocessor is characterized by its limited number of instructions and ability to perform millions of instructions per second. This article discusses a CPLD which is based on a 32-bit general-purpose four-stage pipeline processor equipped with a floating-point unit. The processor includes a comprehensive collection of instructions, programs, and data memory, general-purpose registers, and an arithmetic logic unit that can perform single-precision floating-point arithmetic calculations [4]. Designing high-performance arithmetic hardware has always been a sought-after challenge because microprocessors and signal processors are widely used. The Arithmetic and Logic Unit, which controls the speed of processor operations, is a crucial component of microprocessors. Simple arithmetic calculations are performed by standalone circuitry on modern CPUs. Adding on-chip memory or cache to fast arithmetic hardware allows processors to reduce latency associated with data access from main memory, resulting in a significant boost in performance [5]. CPUs and GPUs nowadays come equipped with processors that can accommodate multiple and robust ALUs. Typically, the ALU is responsible for performing various mathematical, logical, and decision-making computations as part of the final processing stage by the processor. Arithmetic operations such as addition, subtraction, and incrementing, as well as logic operations like AND, OR, XOR, and NOT, are all executed by the ALU[6]Floating point operations are widely utilized in various sectors due to their wide dynamic range, simple operation rules, and high precision. There is an increasing need for high-speed hardware floating point arithmetic units to fulfill the demand for high-speed data signal processing and scientific procedures. Additionally, the use of floating-point arithmetic operations is increasing in commercial, financial, and internet-based applications [7]. The floating point representation is a widely used method to represent real numbers in scientific notation. It uses a sliding window mechanism to adjust the precision according to the number's scale, making it capable of representing extremely large or small numbers, ranging from 1,000,000,000,000 to 0.000000000000001. To implement floating point arithmetic on reconfigurable hardware, such as FPGAs, is challenging due to the complex algorithms. Therefore, we aim to construct a floating-point arithmetic unit or DSP processor for high-accuracy scientific tasks using Verilog HDL and Altera Quartus II. The module includes arithmetic operations such as addition, subtraction, multiplication, and division [8].

II. METHODOLOGY

The focus of this paper is on planned to follow a Bottom to up approach while designing this ALU. Firstly, we implemented simple logic blocks such as Comparator, Adder, Shifter, etc. in Verilog. Then after verifying their performance in the ModelSim simulator, with respect to generated signal patterns as inputs, we implemented the whole design.

A. Design Block

The design comprises of 5 basic sections as shown in "Fig. 1".

1. Opcode Decoder
2. Arithmetic Block
3. Logical Block
4. Comparator Block
5. Shifter Block

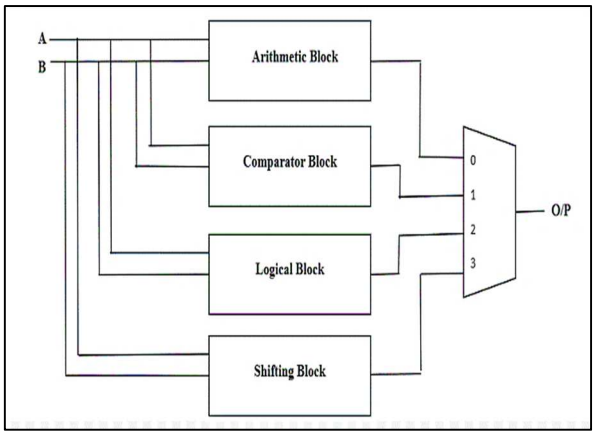

Fig. 1. ALU Block Diagram

B. Opcode Decoder

The proposed design consists of a main opcode decoder that activates the corresponding function block based on the instruction opcode executed by the processor and passed to the ALU block. The corresponding output lines will provide the activation currents of the respective function blocks. The decoder is 6 to 43 bits, it takes a 6-bit opcode as input and it turn activates the corresponding function block to perform the desired operation.

C. Selected Instructions

The ALU that was designed requires a clock signal, two 32-bit operands, and a 6-bit opcode as its inputs. The instructions that have been implemented in the ALU can be found in TABLE I.

"Table. I". show the complete list of the implemented instructions, and "Table. II". show the Floating-Point: Special Cases.

.

TABLE. I. Complete List of The Implemented Instructions

SELECT OPCODE	ACTIVE-HIGH DATA				
	ARITHMETIC S4 S4=00		COMPARRATOR S4 S4=01	LOGIC S4 S4=10	SHIFTING S4 S4=11
S0 S1 S2 S3	Cin=0	Cin=1			
0 0 0 0	OUT=A	OUT=A+1	OUT=A>B	OUT=~A	OUT= shl A
0 0 0 1	OUT=A+B	OUT=A+B+1	OUT=A<B	OUT=~(A^B)	OUT=shl B
0 0 1 0	OUT=A+`B	OUT=A+`B+1	OUT=A≠B	OUT=~A ⱽB	OUT=shr A
0 0 1 1	OUT=-1	OUT=0		OUT=0	OUT=shr B
0 1 0 0	OUT=B	OUT=B+1		OUT=~(AⱽB)	OUT=rol A
0 1 0 1	OUT=`B	OUT=`B+1		OUT=~B	OUT=rol B
0 1 1 0	OUT=A-B	OUT=A-B+1		OUT=A	OUT=ror A
0 1 1 1	OUT=B-A	OUT=B-A+1		OUT=A^~B	OUT=ror B
1 0 0 0	OUT=1	OUT=2		OUT=AⱽB	
1 0 0 1	OUT=0	OUT=1		OUT=~A^B	
1 0 1 0	OUT=A-1	OUT=A		OUT=B	
1 0 1 1	OUT=A+A	OUT=A+A+1		OUT=A^B	
1 1 0 0	OUT=B-1	OUT=B		OUT=1	
1 1 0 1				OUT=Aⱽ~B	

By combining these operations, it is possible to implement any logic operation.

D. Arthimetic Block

This block is used to perform arithmetic operations such as addition, subtraction and multiplication. The accumulator and auxiliary register b provide input to this block, as shown in "Fig. 2".

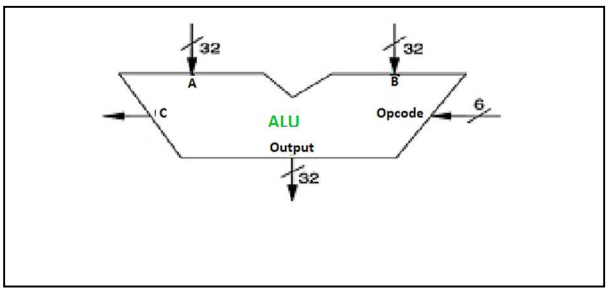

Fig. 2. Pipelined RISC Architecture

1. Integer Addition/Subtraction

Fast addition is crucial in various digital systems, and it's important to note that subtraction and addition are essentially the same operation. The only difference is that in subtraction, the second operand's sign is reversed. With this understanding, it's easy to execute integer subtraction using any integer adder. To make the adder compatible with both operations, slight modifications are necessary to perform addition and subtraction of integers, a logic must be set up. If subtraction is selected, the B operand needs to undergo a two's complement conversion, while for addition, B should remain the same. There are two ways to achieve this: using an inverter and a per-bit multiplexer or by using an XOR gate. In the XOR gate method, one bit is associated with control, and the other is linked with the corresponding bit of B.

Identify applicable funding agency here. If none, delete this text box.

2. Integer Multiplication

Multiplication is a commonly used operation that has various applications, including scientific calculations and signal processing. It is also essential for performing division. Integer coefficients can be implemented through different methods, with typical implementations being change and add. When performing multiplication, both registers are utilized to produce a 64-bit output. as in "Fig.3"

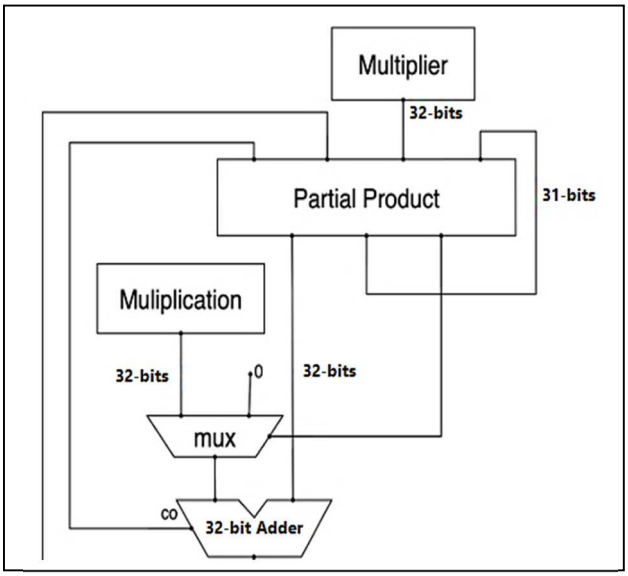

Fig. 3. Shift and Addition Multiplier Circuit

E. Comparator Block

This block includes a combinatorial HDL code that performs bit matching and comparison. The corresponding outputs can be used to derive instructions based on the comparison. Likewise, numerical greater or less than indices can be used on the respective operands.

F. Logical Block

This block contains fundamental logic gates like AND, OR, NOT, and XOR responsible for performing logic operations on data operands. The outputs are stored in their respective latches. These logic gates perform standard logical operations on bit operands. If both Ai and Bi are 1, the AND operation produces 1 in the i output bit. If either Ai or Bi is 1 but not both, the OR operation produces 1 in the i output bit. The NOR operation is the opposite of the OR operation, with a 0 inserted in the bit position if the conditions are not met. The ENV operation reverses the bits of A and B.

G. Shifter/ Rotat Block

This block contains simple shifters like the barrel shifter and end cap turning mechanisms that can be used to shift data to the right. Two methods are used for this shift - arithmetic and logical. In arithmetic shifting, the operation is expanded as it is shifted by adding the shift symbol. In logical shifting, the empty bit positions in the shift are filled with zeros like in "Fig.4" [9].

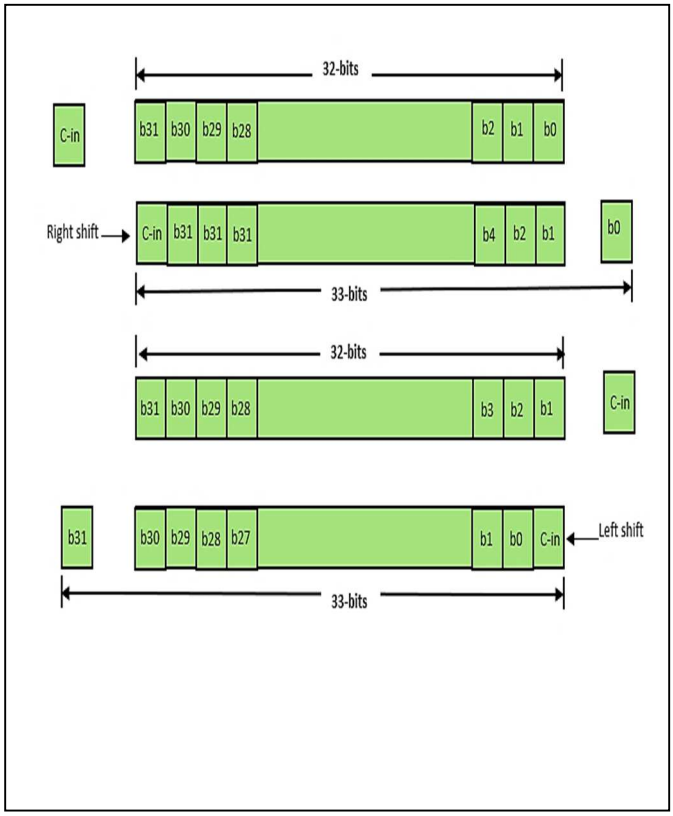

Fig. 4. 32 bit scaled shifter block

H. Floating Point Unit

Floating point units (FPUs) are specialized hardware components within the CPU that are responsible for processing decimal operations. FPUs perform basic arithmetic operations such as addition, subtraction, multiplication, and division. However, unlike the Arithmetic Logic Unit (ALU) which handles integer values represented in binary numbers, the FPU handles both integer and fractional parts of numbers. Since bit vectors cannot distinguish between integer and fractional parts of numbers, the operands must be separated into the sign, exponent, and mantissa parts of the number. This allows for more accurate decimal calculation within the CPU.

I. IEEE 754 Standred

The standard includes support for both 32-bit single-precision and 64-bit double precision numbers. Double precision has a wider range and higher precision than single precision due to its 11 bits of exponent and 52 bits of fraction, compared to 8 bits of exponent and 23 bits of fraction for single precision shown in "Fig 5". Since this article only focuses on 32-bit inputs, we will only explore the single-precision format, which operates in the same way as double precision as shown in TABLE. II.

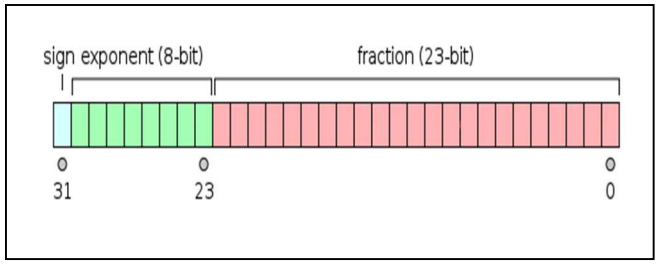

Fig. 5. Single-Precision IEEE 754 Format

The standard accommodates 32-bit single-precision numbers and 64-bit double precision numbers. It is noteworthy that the double precision category offers a broader range, considering that it has 11 bits of exponent as opposed to the 8 bits of single-precision. Furthermore, double precision provides higher precision, thanks to its 52 bits of fraction in comparison to the 23 bits available in single-precision as shown in "Fig 5", and TABLE. II.

TABLE. II. Floating-Point: Special Cases

Number	Sign	Exponent (e)	Fraction (f)
0	X	00000000	00000000000000000000000
∞	0	11111111	00000000000000000000000
−∞	1	11111111	00000000000000000000000
NAN	X	11111111	nonzero

J. FPU Instruction

The operations supported by the floating-point unit include addition, subtraction, and multiplication as shows in TABLE. VI.

TABLE. VI. FPU Operations

Operation	OP Code
Addition	00
Subtraction	01
Multiplication	10

1. Floating-Point Adder/Subtractor

The most commonly employed floating-point operation is FP addition/subtraction, analogous to the widespread use of integer addition/subtraction in the ALU.

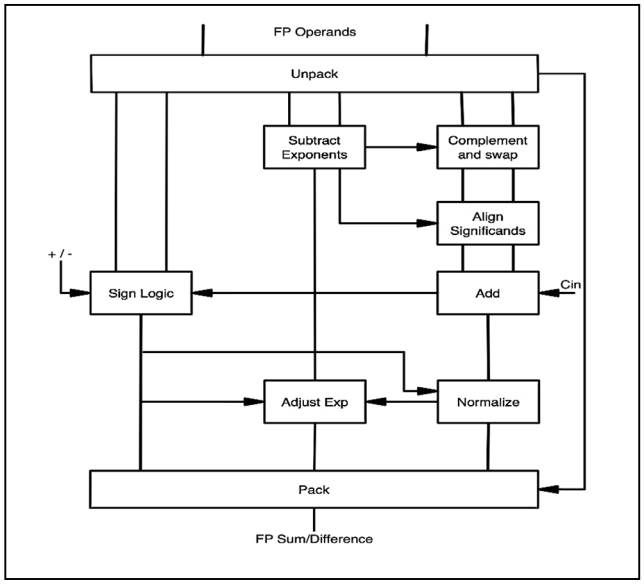

Fig. 6. Block Diagram of Floating-Point Adder/Subtractor

In order to align the smaller operand with the larger operand, one needs to shift the smaller operand to the right by an amount determined by the difference between their exponents as shows in "Fig. 6".

2. Floating-Point Multiplication

Efficient multiplication design is crucial as floating-point multiplication is widely used in various applications, comparable to floating-point addition/subtraction as shows in "Fig. 7".

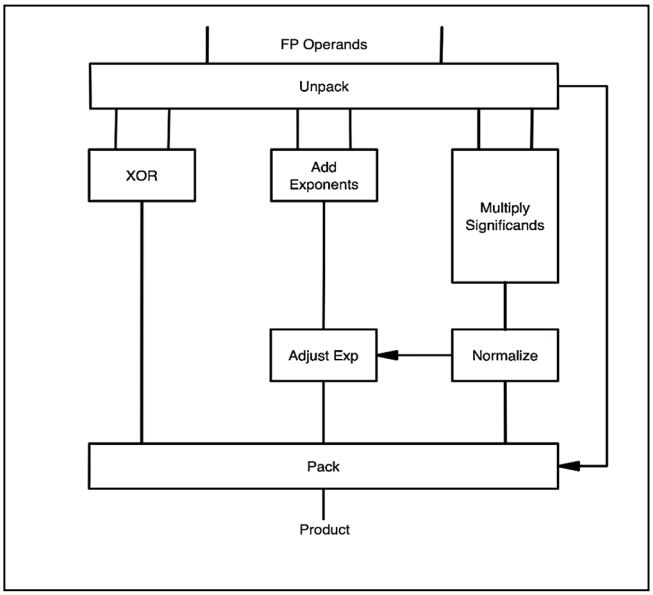

Fig. 7. Generic FP Multiplier Block Diagram

K. *Hardware Test Environment*

Our team has developed a comprehensive EDA (Electronic Design Automation) printed circuit board system to enable efficient and thorough testing and validation of the floating point ALU in hardware. This system is intended to facilitate the use of more advanced FPGA development tools as shows in "Fig.8".

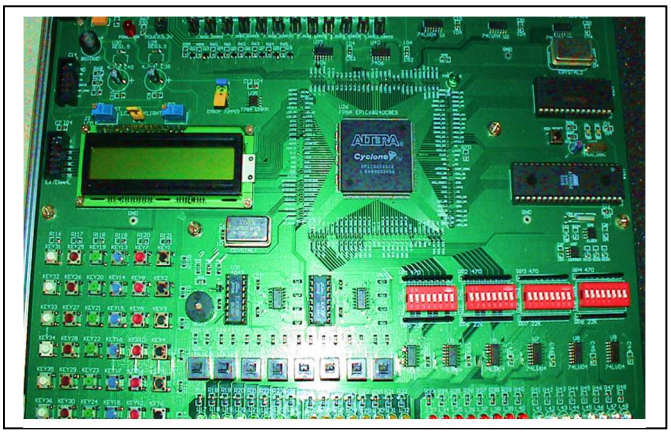

Fig.8. PCB with Cyclone FPGA

III. SIMULATION AND RESULTS

To confirm the proper functioning of each arithmetic block and floating-point implementation, they were programmed using Verilog hardware descriptive language. In ModelSim, test benches were created and executed to validate their performance as shows in "Fig.9", "Fig.10", and "Fig.11".

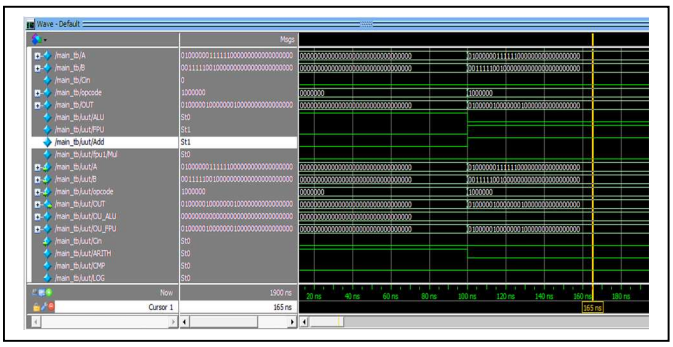

Fig.9. FPU Adder

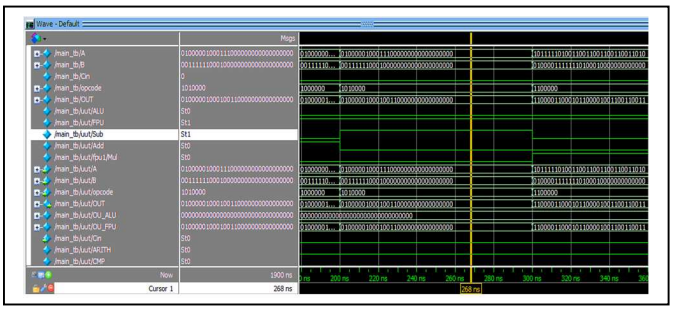

Fig.10 .FPU Subtractor

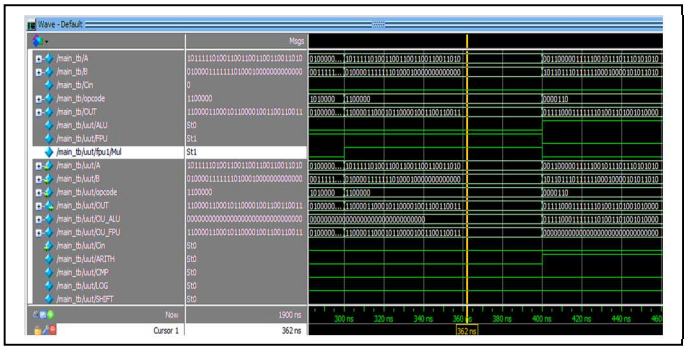

Fig.11. FPU Multiplier

IV. CONCLUSION

The goal of this project is to create an architecture for an arithmetic logic unit and floating-point unit that can handle all arithmetic and logical operations for a RISC processor. The design should be flexible enough to scale up to 32 bits. The project uses Verilog language and utilizes software tools like ModelSim 10.1 and Quartus II 12.1 for simulation and testing. The techniques utilized in this project provide a distinct approach to design, test, and verify the system. The design was done using electronic design automation tools.

REFERENCES

[1] C. Hamacher, Z. Vranesic and S. Zaky, Computer Organizationn fifth edition, 2002.

[2] Mohamed M. Eljhani, Veton Z. Kepuska, Reduced Instruction Set Computer Design on FPGA. 2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA, 25-27 May 2021, Tripoli-Libya

[3] J. Poornima, G. V. Ganesh, M. Jyothi, M. Shanti and A. Jhansi, "Design and implementation of pipelined 32-bit Advanced RISC processor for various D.S.P Application," *Proceedings of International Journal of Computer Science And Information Technology,* June 2012.

[4] R. P. Colwell, C. Y. Hitchcock, E. D. Jensen, H. M. Brinkley Sprung and C. P. Kollar, "Instruction sets and beyond: computers, complexity, and controversy,," *IEEE Computer,* 1986.

[5] A. Athihrii, M. Stephen and S. Kumar, "Design and implementation of 32-bit ALU using verilog," 02 June 2016.

[6] D. Malik and R. S. Rathore, "32-bit Arithmetic Logical Unit (ALU) using VHDL," vol. 1, no. 1, 26 NOV 2013.

[7] R. Cherian, N. Thomas and Y. Shyju, "Implementation of Binary to Floating Point Converter using HDL," no. 461-64, 2013.

[8] R. Payal, "Simulation and Synthesis Model for the Addition of Single Precision Floating Point Numbers Using Verilog," vol. 02, no. 09, Sep 2013.

[9] K. Garg, M. Garg, S. Mehta and S. Vashist, "n-Bit Arithmetic & Logical Unit (ALU)," 2015.