Design & Verification of Single Precision Floating Point ALU

Nirmala S O [1], Leela G H[2], G S Sunitha[3] , Vanishree H V[4]
[1234]*Dept., of E&CE BIET , Davangere, Karnataka*

Abstract— A single precision floating-point arithmetic logic unit (ALU) is considered as a part of the math coprocessor. 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-bit single precision floating point values. The application of this arithmetic unit is located in the math coprocessor. In a Reduced Instruction Set Computation (RISC) processor, for signal processing, a value with high accuracy is required and as it is an iterative process, the calculation should be as fast as possible. The floating-point representation can calculate very large or very small process quickly and accurately. The system will be designed, verified and implemented with Verilog hardware description language using Cadence software tool.

Index Terms— floating point, IEEE-754, Single Precision, synthesis

I. INTRODUCTION

FPU is a math coprocessor which is designed specially to carry out operations on floating point numbers. The ALU can perform the arithmetic operation: addition, subtraction, multiplication and division. It can also perform some logical operations such as logical NOT, logical NAND and shift right. Main function of FPUs can execute different functions such as exponential or trigonometric calculations. The logical method for Addition and Subtraction operation is designed for getting better performance which is required in signal computation applications and Floating point ALUs are used for high precision computing. The design of floating point ALU is used to get the aim of small area. This ALU uses 32 bit numbers, which is the common computer word length. The numbers are represented in IEEE 754 standard. This standard is widely used in floating point arithmetic.

Single precision number format consists of 32 bits and the double precision number format consists of 64 bits. The single-precision number format is 32 bits, including 1-bit sign value, 8-bit exponent value, and 23-bit fraction value. For many algorithms that require millions of calculations per second, floating-point numbers are used because of their wide and dynamic range. In applications with floating-point numbers, bit lengths that are overused can lead to more usage and lower speed.

In modern computing, floating-point arithmetic is crucial for a wide range of applications, including scientific simulations, graphics rendering, financial calculations, and more. Efficient and accurate floating-point operations are essential for achieving reliable results in various computational domains.

II. LITERATURE SURVEY

The paper [1] provides the representation of floating point number, IEEE Single precision format and IEEE double precision format. It shows the advantages and applications of floating point representation. The proposed floating point arithmetic unit that supports four arithmetic operations: Add, Subtract, Multiply and Divide. All the basic mathematical arithmetic operations have been carried out in four separate modules one for addition, one for subtraction, one for multiplication and one for division. This is a review paper which employs different proposed techniques for implementation of floating point ALU. The paper [2] presents the floating point unit according to IEEE 754 Standard. Resolved floating point representation concept which have large range of values as well as accuracy. Hence hardware necessity is reduced, thereby reducing power consumption and delay. So, designing of power efficient 32-bit single precision Floating point unit (FPU) based on IEEE-754 standard based on FPGA is better. The paper [3] presents the Design of 32-Bit floating point Arithmetic logic unit. The methods of Addition, Subtraction, Multiplication and Division are simulated by Verilog HDL using Xilinx Software,14.7 Version. The logical method for Addition and Subtraction operation is expanded in order to decrease the no.of gates used. The results are seen in the RTL view and Synthesis reports. The paper [4] presents an implementation of addition and subtraction for IEEE single precision floating point numbers and their pipeline design. They implemented trade-off between area and speed for accuracy.

Implementation of an adder and subtractor as a bit-parallel adder. The algorithms are designed in VHDL language and implemented on FPGA kit by use of Xilinx ISE compiler. Floating point adder and subtractor unit design using pipelining provides high performance and increase the speed. It is used to execute multiple instructions simultaneously.

III. METHODOLOGY

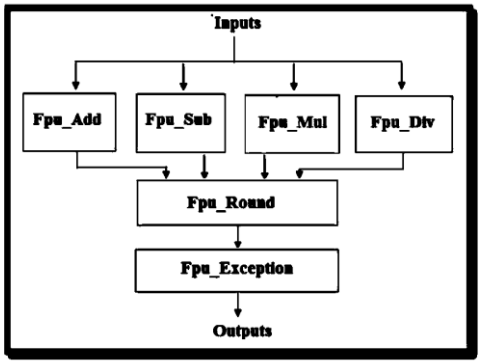

Fig1. Block Diagram

The components in the Floating Point ALU are

- Add/Sub: This block performs addition and subtraction operations two single precision floating point numbers. It handles the exponent alignment and fraction addition or subtraction.
- Multiply: This component handles multiplication of two single precision floating point numbers. It multiplies the fractions and add the exponents.
- Round: This component rounds the result of arithmetic operations according to the specified rounding mode.
- Exception Handling: This part detects and handles exceptional cases such as overflow, underflow, division by zero and invalid operation. It generates appropriate flags or signals to indicate these conditions.
- Output: The final result of the arithmetic operation is provided as output, along with any flags indicating exceptions or overflow/underflow conditions.

Floating point addition/ subtraction

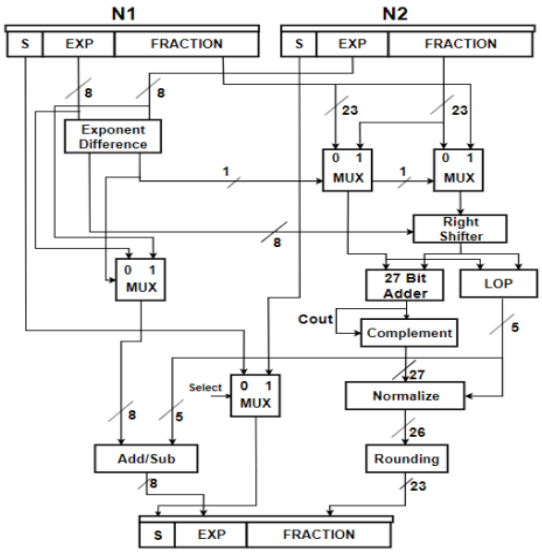

Fig.2 Architecture of floating point addition and subtraction

Algorithm for ADD/SUB

1.Determine if either exponent (E1 or E2) is all ones, indicating NaN or infinity. Set the Exception flag accordingly.

2.Determine if either exponent (E1 or E2) is all zeros, indicating denormalized numbers. Set appropriate flags.

3. Calculate the difference between the exponents (E1 - E2) and handle the sign of the result.

4. Select the larger exponent (E) to ensure correct alignment of mantissas.

5. Shift the smaller mantissa (M2) to align with the larger one (M1) based on the exponent difference.

6. Determine the operation to be performed (addition or subtraction) based on the sub input and the signs of the operands.

7. Perform addition or subtraction of the mantissas (M1 and M2) considering the operation and carry.

8. Adjust the sign of the result based on the operation and the carry.

9. Increment the exponent by 1 if addition is performed and a carry is generated.

10. Determine if the result is negative and calculate its two's complement if necessary.

11. Normalize the result and calculate the shift amount.

12. Subtract the shift amount from the exponent to perform normalization.

13. Check for overflow and underflow conditions based on the modified exponent.

14. Combine the sign, exponent, and mantissa to get the final result.

Floating point multiplication

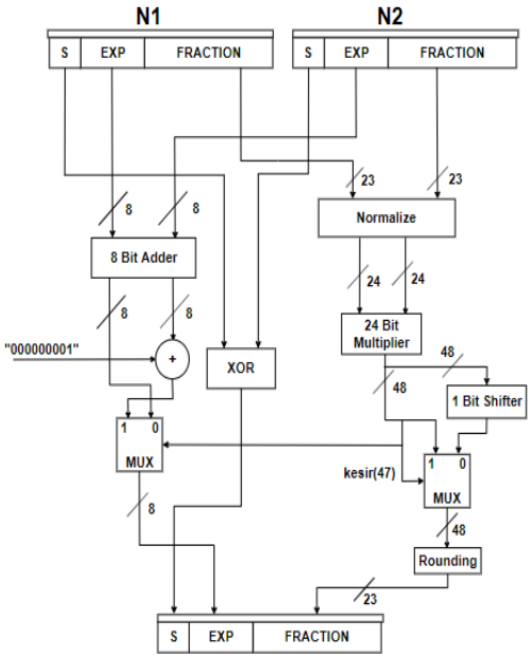

Fig.3 Architecture of floating point Multiplication

Algorithm for Multiplication:

1. Calculate the exponent part of both input numbers n1 and n2.

2. Check if either exponent is all ones (indicating NaN or infinity), in which case set Exception flag.

3.Calculate the sign of the result (final_sign) by performing XOR operation on the signs of n1 and n2.

4. Determine if either input number is denormalized by checking if all exponent bits are zero.

5. Multiply the mantissas of n1 and n2 using a 24-bit multiplier (Multiplier24bit).

6. Determine if the product is normalized or denormalized.

7. Calculate the rounding bit based on the product and whether it needs to be rounded.

8. Normalize the product by shifting bits if necessary.

9. Add the exponents of n1 and n2 and subtract the bias (127) to get the final exponent (final_E).

10. Adjust the exponent if necessary based on the product's normalization.

11. Check for overflow and underflow conditions based on the final exponent.

12.Concatenate the sign, exponent, and mantissa to get the final result.

Floating point Division

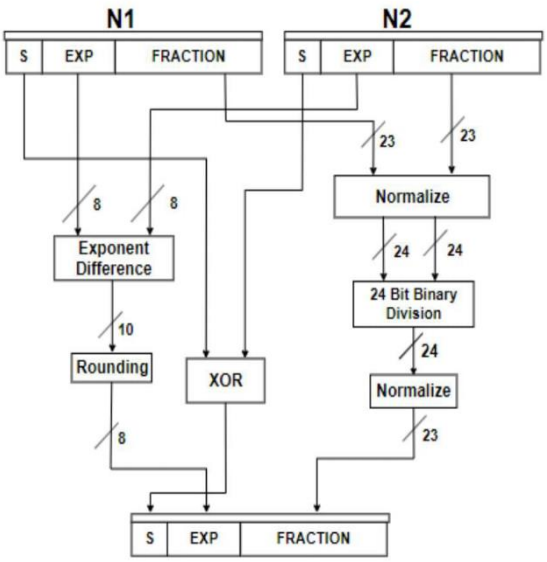

Fig.4 Architecture of floating point Division

Algorithm for Division:

1. It is checked whether N1 and N2 have a special case of being zero or infinity. If such a condition exists, the result is sent directly to the output.

2. Two numbers in 32-bit floating-point format are input into the designed module.

3. The fraction values of numbers are normalized. Thus, the new fraction lengths are 24 bits.

4. 24-bit fraction values obtained as a result of normalization are divided.

5. The 24-bit number obtained from division is rounded and the fraction value of the result is generated.

6. To calculate sign bit of the result, the sign bits of the two numbers taken as input are passed through xor gate.

7. The exponent values of N1 and N2 are subtracted, 127 (bias value) is added to the difference. Exponent, fraction, and sign values are placed in the required 32-bit result, and the result is sent.

IV. RESULTS

The results obtained for the different operations on two set of data are as shown in Fig. 5 and 6.

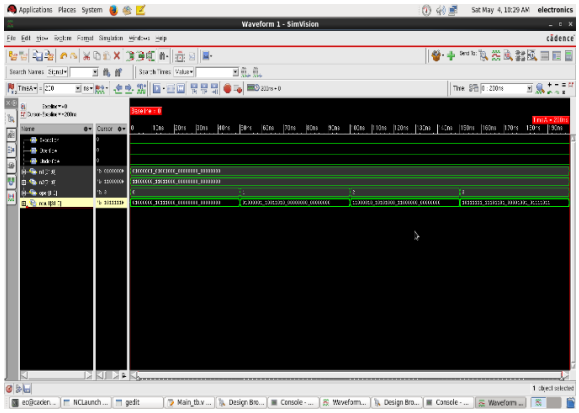

Fig.5 Result of floating points 12.5 and -6.75

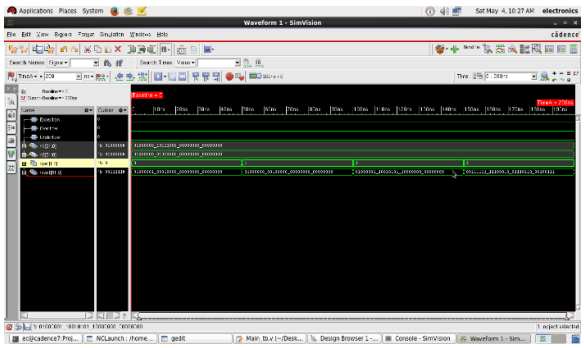

Fig.6 Result of floating points 5.75 and 3.25

Also the design were synthesized and area , timing and power reports were generated

V. CONCLUSION

Floating point arithmetic unit has been designed to perform arithmetic operations such as addition, subtraction, multiplication and division on floating point numbers. The unit has been coded in Verilog HDL. Code has been synthesized using cadence tool and verified on the software successfully. Improvement is observed in terms of speed and power.

REFERENCES

[1] Naresh Kumar, Onkar Singh, Harjit Singh, “Design and Analysis of Floating Point Arithmetic Unit: A review” International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 07 Issue: 10, Oct 2020.

[2] Mr. Ankit Trivedi, Mr. Apoorv Verma “A Review on Single Precision Floating Point Arithmetic Unit of 32 bit Number” International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 06 Issue: 04 | Apr 2019.

[3] M. Bhavani, G. Sirisha, K. Siva Kumari, B. Lalitha Rani, A.Charishma “Design of Single Precision Floating Point Arithmetic Logic Unit” M. Bhavani, et. al. International Journal of Engineering Research and Applications www.ijera.com ISSN: 2248-9622, Vol. 11, Issue 8, (Series-III) August 2021, pp. 18-24.

[4] Hariom Kumar , G. Sankara Rao “Design And Implementation of Single Precision Floating Point ALU” International Journal of VLSI System Design and Communication Systems ISSN 2322-0929 Volume-07, Jan-Dec-2019, Pages:19-23.