JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

FPGA-SUPPORTED HDL APPROACH TO IMPLEMENT REVERSIBLE LOGIC GATE-BASED ALU

M SUMALATHA (ASSISTANT PROFESSOR)[1], T PRATHYUSHA (UG STUDENT)[2], S GOVINDA RAJU (UG STUDENT)[3], T SRAVANI (UG STUDENT)[4], P CHANDANA (UG STUDENT)[5]

DEPARTMENT OF E.C.E, N.B.K.R. INSTITUTE OF SCIENCE AND TECHNOLOGY, TIRUPATHI DISTRICT, ANDHRA PRADESH, INDIA.

❖ ABSTRACT:

This manuscript banks on the design of reversible gates and implementation of an ArithmeticLogic Unit – 32 bit (ALU) utilizing Verilog with Xilinx ISE 14.7, Spartan 6 FPGA kit. The same functionality is compared with a basic logic gate-based ALU. Reversible gates can produce a distinct output vector from each input vector, and the opposite is also possible.

Circuits with irreversible gates suffer from data erosion. Power loss results from a circuit's loss of data. In conclusion, gates with reversible logic are preferable over irreversible counterparts. A library of reversible gates, comprising of AND, OR, NAND, NOR, and XOR, using Verification Logic Hardware Description Language (HDL) is developed, which in turn contributes to the designing of arithmetic and combinational logic like full adder, decoder (2:4),decoder (3:8), multiplier, full subtractor, and comparator.

KEYWORDS:

Verilog, Xilinx ISE, Spartan 6.

❖ INTRODUCTION:

In the newest technology, Precision plays a major part in more applications like Digital signal processing. Floating point numbers are used to represent non-integer fractional numbers and are used in most engineering and technical calculations. The most commonly used floating point standard is the IEEE Standard. According to this standard, floating point numbers are represented with 32bits (single precision). Floating point numbers are used in more applications are such as telecommunications, medical, imagining, radars etc. In this paper the operations are executed on 32-bit floating point numbers and the logical method for Addition Subtraction multiplication and division operations is designed to getting better performance which is required in signal computation applications. The design of floating point ALU is used to get the aim of small area. Then we use Verilog hardware description language (VHDL). It is a user defined language. In VHDL we use two approaches to get better performance., we use topdown approach. Topdown approach means stepwise design. The HDL is used to respect to speed and area. In the ALU the main block of central processing unit (CPU) that handles all format have 32 bits to represent a floating arithmetic operations, logical operations etc. The IEEE 754 floating point standard format has dividing into three main parts. They are Sign (1 bit), Mantissa (8 bits), exponent (23 bits).

IEEE 754 floating point format:

The IEEE 754 floating point format have 32 bits for representing a floating point number.

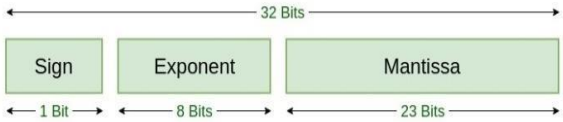

Fig 1: IEEE 754 floating point format

The standard format to represent floating point numbers which has three major parts as shown in figure1. They are sign, mantissa and exponent. The sign bit carries 1-bit where '1' and '0'represents a positive and negative number. The mantissa is also known as floating point number or significant, it carries much no.of bits. which carries 23 bits. It represents precision bits of numbers.

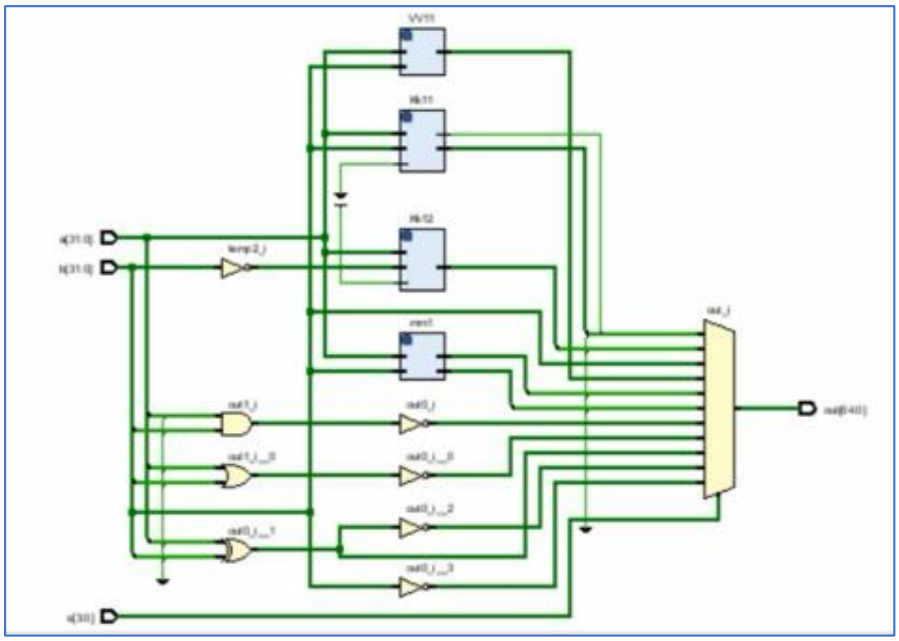

Fig 2: Block diagram of floating point ALU

1. Architecture

Arithmetic unit performs Arithmetic operations on floating point numbers consist of addition, subtraction, multiplication and division. The operations are done with algorithms similar to those used on sign magnitude integers (because of the similarity of representation) — example, only add numbers of the same sign. If the numbers are of opposite sign, must do subtraction.

The Addition, Subtraction, Multiplication and division have been executed by the 32-bit floating point ALU as shown in figure 2. Pre Normalization blocks have been used. First block is used for the addition/subtraction and other for multiplication/division operations. The Mantissa part has been normalised by the post normalization unit. Then the final result is obtained. Two IEEE 754 32 bit operands have been taken to do the arithmetic operations. Eventually the exceptions occurred have been detected and handled by using handling. The Exception handling are of two types. They are one is Over flow and another one is Underflow. First one is overflow that occurs in Addition and underflow occurs in Subtraction.

Overflow is said to occur when the true result of an arithmetic operation is finite is said to occur when the true result of an arithmetic operation is finite but larger in magnitude than the largest floating point number which can be stored using the given precision. Underflow is said to occur when the true result of an arithmetic operation is smaller in magnitude (infinitesimal) than the smallest normalized floating point number which can be stored. Overflow can't be ignored in calculations whereas underflow can effectively be replaced by zero. These exceptions are handled by using exception handling block. If there are no exceptions then we get final result. In such cases, the result must be rounded to fit into the available number of M positions. The extra bits that are used in intermediate calculations to improve the precision of the result are called guard

bits. It is only a tradeoff of hardware cost (keeping extra bits) and speed versus accumulated rounding error, because finally these extra bits have to be rounded off to conform to the IEEE standard.

1.1 Addition and Subtraction

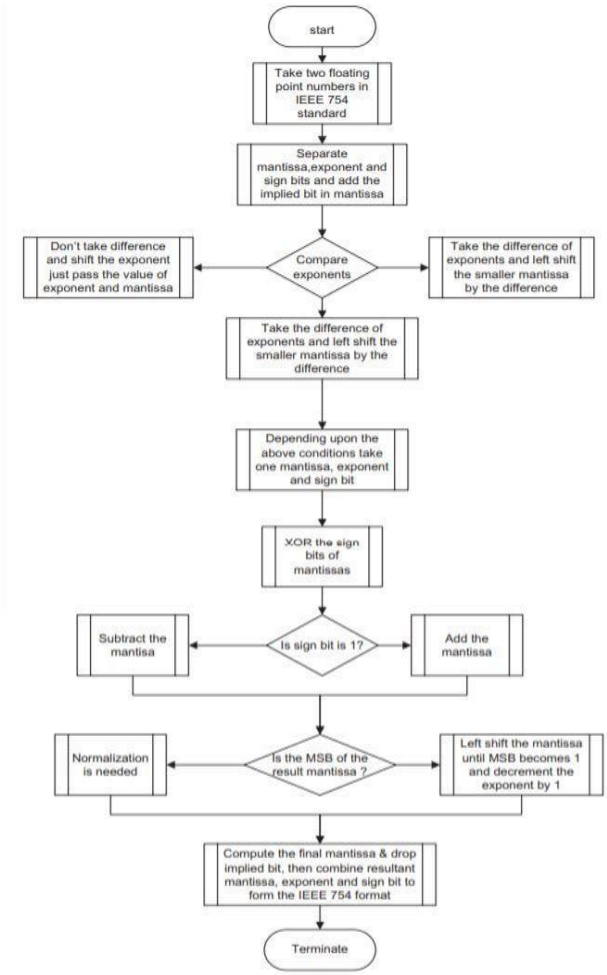

Fig 1.1: Flow chart for Addition and Subtraction

The flow chart for Addition and Subtraction as shown in figure 1.1. Here duplet scopes be allowed transpire occupy measure. One and other ivalues exist matching indication, Both MSB bit perhaps to '1' or '0' (Balnace of contrasy forces).Although the pair of ivalues exist non identical indications, especially be MSB of only integer added is '1' (positive) and the MSB of other is '0' (negative).In the first place require to inspect effective indication of twain numbers if the signal of pair digits possess are non identical execute two's complement for the MSB number having '1'.Behind Adding performance takes place carry out one and the other digits.

First step is to take two floating point numbers using IEEE 754 standards. Now we separate mantissa, exponent and sign bits. Next we go to exponent compare circuit. It performs the comparision operation of two exponents. If we have a exponent difference we perform left shift operation, Otherwise just pass the values of exponent and mantissa. There are two possibilities that may occur while computing two 32-bit floating point numbers. By depending on sign bit, we perform arithmetic operation. If both operands are same sign, Their MSB could be either '1' or '0' (Positive or Negative). When both operands are of different signs, that is MSB of one operand is '1' (Positive) and the MSB of other is '0' (negative).We need to check the sign of two floating point numbers. If the sign of both numbers are different, then we perform one's complement plus one operation for having MSB is '1'. After Addition operation is executed. If we get any exceptions like overflow or underflow .Then overcome the exceptions by left shift the mantissa until MSB becomes '1' and decrement the exponent by 1. Finally then combine the resultant mantissa, exponent and sign bit to from the IEEE 754 format.

1.2 Multiplication

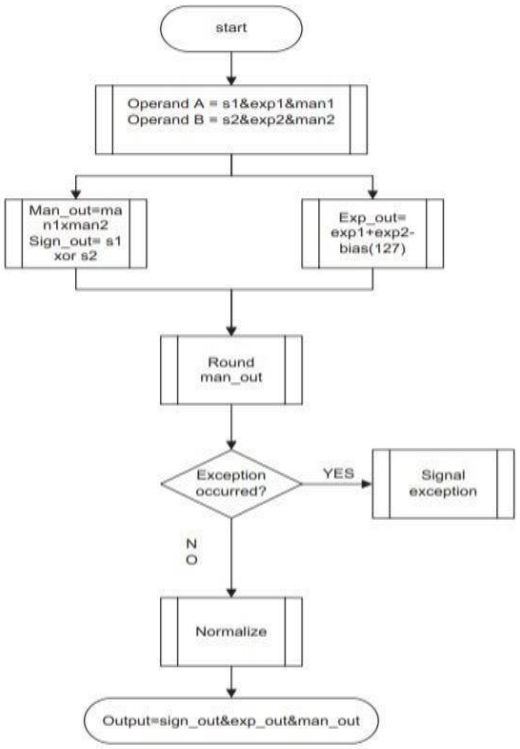

Fig 1.2: Flowchart for Multiplication

The algorithm of floating point multiplication is shown in fig 1.2. First step is to take the two floating point numbers using IEEE 754 standard. We have to check whether the multiplicand, multipliers and or zeros or not. We perform Addition and Subtraction operations, and then align the mantissas. In such a way that both the exponents are equal. In multiplication algorithm, there is no need to align the mantissas. To get exponent output, add the two exponents and the bias 127 is subtracted. The ex- or operation is performed on the MSB bits to get the sign bit. Multiply the mantissa parts of the two numbers. We used fixed point signed magnitude multiplication algorithm. Check the exception flags are overflow, underflow have been determined. Overflow means the corresponding register cannot store the additional bit. After normalisation we got the final output.

1.3 Division

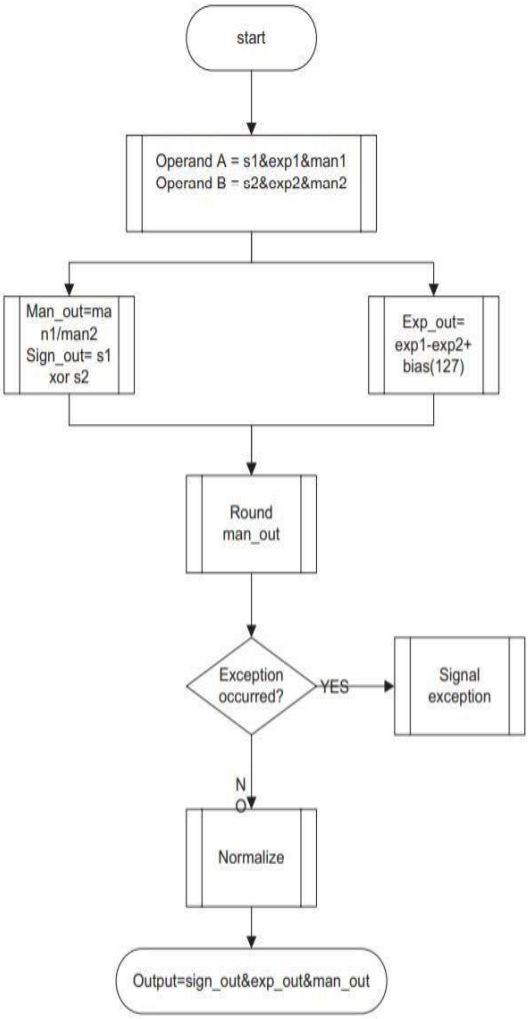

Fig 1.3 : Flowchart for division

The flowchart for division as shown in above figure 1.3.First step is to take the two floating point numbers using IEEE 754 standard. We check zeros or not. If BR is zero then we cannot possible to perform division. If divisor is zero, we get division by zero problem. Report the error message as divide by zero. If BR is not equal to zero, that means divisor is zero. Now check the value of dividend. It is present in Ac register. Suppose dividend value is zero, then result is also zero. No need to continue the process. Otherwise perform the operation. Here QS is nothing but the size of quotient and depends upon dividend sign and divisor sign as same or different. It performs exclusive or operation on dividend sign and divisor sign. If they are same then the result is zero and they are different, result will be one. Next we have to check any overflow or underflow. So for that purpose we have to subtract division from the dividend. In order to perform the subtraction operation, we need to add two's complement of the division to dividend. To overcome the overflow, we perform shift right operation. To get exponent output, Subtract the two exponents and the bias 127 is added. The ex-or operation is performed on the MSB bits to get the sign bit. Check the exception flags and overflow or underflow have been determined. Divide the mantissa parts of the two numbers. We have to follow restoring algorithm. After normalisation we got the final output.

❖ SIMULATION RESULTS AND PERFORMANCE ANALYSIS

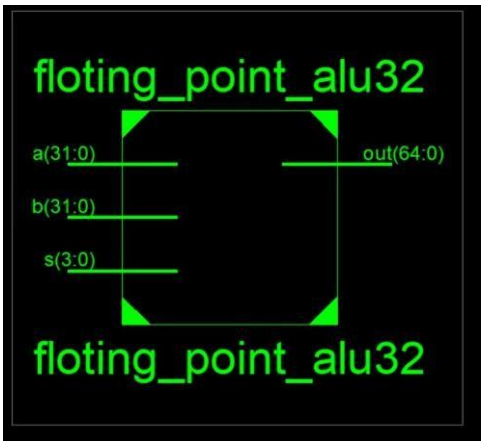

Fig 3 : Basic ALU Block diagram

Fig 4 : Simulation Outputs for 32-bit floating point ALU

The above figure 4 shows the simulation output for 32- bit flaoting point ALU, is represented in binary and unsigned decimal form. By depending on selection lines to find the output.

Fig 5: Device utilisation of 32-bit floating Point ALU.

Utilization | Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization %
LUT	4483	274080	1.64
DSP	3	2520	0.12
IO	133	328	40.55

Fig 6: Power and Delay analysis of 32 bit Floating point ALU

The above figure 5 shows the Device utilisation of 32-bit floating point ALU, it is presented that how much area is occupied . It takes lesser area compared to older designs. Mainly this figure 5 shows the no.of LUTs are 4483. It is the main aim to use Floating point ALU.

The above figure 6 shows the power and Delay analysis of 32- bit floating point ALU.

❖ CONCLUSION

The implementation of a 32-bit floating point arithmetic logic unit is done and efficient algorithm for addition, subtraction, multiplication and division is implemented in order to reduce the no.of gate used.

❖ REFERENCES

[1]. Shanthala. N1, Nayana. M, Chandrashekar.C, Dr. Siva Yella mp al li "Basic operation performed on Arithmetic Logic Unit (ALU) For 32-Bit Floating Point Numbers", *International Journal of Applied Engineering Research* ISSN 0973-4562 Volume 12, Number 12 (2017) pp. 3248- 3252 Research india Publications. http://www.ripublication.com

[2]. Swathi.A, G.Srinivasulu "ASIC implementation of a High speed double Presicion(64) floating point unit using verilog", International journal and magazine of engineering, technology, management and research ISSN 2348-4845

[3]. Dave Omkar R, Aarthy M, "ASIC implementation of 32 and 64 Bit floating point ALU using Pipelining", International Journal of computer applications(0975- 8887) volume 94-No. 17,May 2014

[4]. H.H. Saleh, ―H.Fused Floating-Point Arithmetic for DSP,‖ PhD dissertation,Univ. of Texas, 2008.

[5]. Jorge Tonfat, Ricardo Reis, ―Improved Fused Floating Point Add-Subtract and Multiply-Add Unit for FFT Implementation‖, in 2014 2nd International Conference on Devices, Circuits and Systems (ICDCS).

[6]. Ushasree G, R Dhanabal, Dr Srat Kumar sahoo, "VLSI implementation of a High Speed Single Precision Floating Point Unit Using Verilog", Proceedings of 2013 IEEE conference on information and communication technologies(ICT 2013)

[7]. Naresh Grover, M,K Soni, "Design of FPGA based 32-bit Floating Point Arithmetic Unit And verification of its VHDL code using MATLAB", I.J Information Engineering and Electronics Buisiness, 2014,1,1-14 published Online February in MECS

[8]. Sayali A. Bawankar, Prof. Girish. D. Korde," Design and Simulation of Floating Point Adder,Subtractor & 24-bit Vedic Multiplier", International Jornal for Research in Applied Science &Multiplier", International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887 Volume 5 Issue V11, July 2017-Available at www.ijraset.com .