International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)
Volume 3, Issue 11, November 2014

Implementation Of Floating-point Adder Unit

Gayathiri K[1], Prakash S P[2], Valarmathy S[3], Jaibalaji K R[4]

Abstract— **Floating point is the most important part of many modern microprocessor. It is complex while using butterfly structure for large bit of floating point representations. For a radix implementation, it consists of complex butterfly structure followed by complex part of addition and subtraction. Thus, these complex butterfly operations can be implemented with a fused add subtract unit. IEEE 754 standard specifies the methods for binary and decimal floating point arithmetic unit. Moreover, single precision addition and subtraction is performed by normalization. The numerical result of the fused butterfly unit is more accurate because only fewer rounding operations are used compared to conventional floating point. In this paper, add/subtract unit is implemented using vhdl language and simulation are verified.**

Index Terms— **add/subtract unit, IEEE 754, normalization, single precision.**

I. INTRODUCTION

For the application-specific system and for designing the digital processor, the digital arithmetic operations plays an very important role. Arithmetic circuits play an important role in digital systems. With the previous development in the very large scale integration (VLSI) circuit technology, many complex circuits are easily implemented today. In general floating-point representations provide a wide dynamic range, free from scaling and overflow/underflow that appear with fixed-point representations. Algorithms are seemed to be impossible to implement very large circuits in the future. This means that not only the conventional computer arithmetic methods, but also the unconventional ones are worth investigation in new designs. The notion of real numbers in mathematics is convenient for hand computations and formula manipulations.

However, real numbers are not well-suited for general purpose computation, because their numeric representation as a string of digits expressed in base 10 can be very long or even infinitely long. Examples include 2/3, π and e. In practice, computers store numbers with finite precision. Arithmetic and numbers used in scientific computation should meet a few general criteria.

- Storage requirements for the numbers should be low.
- Arithmetic operations should be efficient to carry out
- A level of standardization, or portability, is desirable– results obtained on one computer should closely

match the results of the same computation on other computer.

II. FRACTION REPRESENTATION

Beside integer representation, there is a need to represent the fraction part also. In general, the radix part are divided into two parts, an integer part(left of the radix)using positive powers and an fraction part(right of the radix)using negative powers. Hence, fixed and floating point representation are used to handle the fraction parts in computers.

A. Fixed Point Numbers

Fixed number is simple and easy way to represent the fraction part using fixed number of bits. The total number of bits for the fractional part number is the precision of the fixed number. The dynamic range of the fixed point is the ratio of the largest represent able number to the smallest non zero number.

Hence, for more precision we have less dynamic range. The main drawback in fixed point number is the lack of dynamic range because, very large number and very small number cannot be represented in the same representation.

B. Floating Point Numbers

The decimal point is floating when there is no fixed number of digits before and after the decimal point, and such point is said to be floating point numbers. But, the floating point representations are slower and less accurate than fixed point representations, and also they can handle larger range of numbers. Floating Point Numbers is a numbers which presents fractional part. For e.g. following numbers are the 95, -102.5, 1/4,2E-5 etc.

For past years, floating-point arithmetic is considered to be tough subject but now-a-days it is surprised to be found in computers. There are many language to support floating-point data ;They are converted by computers from PC's to supercomputers ;and most compilers will compile floating point from time to time; and most important is each and every operating system must respond to IEEE standard exceptions such as underflow and overflow.

The main advantage in floating point is its large dynamic range. This is because the largest represent able number is approximately equal to the base raised to the power of maximum positive exponent and the smallest non-zero number is approximately equal to the base raised to the power of the maximum negative exponent. The another important factor of floating point is its precision. It is equal to the number of bits used to represent the mantissa.

In general, a floating-point number consists of three main parts: sign (S), mantissa (M) and exponent (E). Its value can be given by $(-1)^s \times M \times base^E$ the base is one of the most important aspects needed to know when using floating point representation. It is equal 2 for binary, 10 for decimal and 16 for hexadecimal numbers.

ISSN: 2278 – 909X All Rights Reserved © 2014 IJARECE

C. Floating Point Unit

Floating-point units (FPU) is one of the math coprocessor which is designed specially to carry out operations on floating Point number. Basically the Unit can handle operations like addition, subtraction, multiplication and division. In this paper, we design a single precision IEEE 754 FPU. This can handle not only basic floating point operations like addition, subtraction, multiplication and division but can also handle operations like shifting, square root determination and other transcendental functions like sine, cosine and tangential function.

III. IEEE 754 STANDARDS

IEEE 754 is a IEEE standard for the Floating-point arithmetic developed for binary floating point arithmetic. The standard specifies the extended floating-point number formats, arithmetic operations, rounding algorithms, and its exceptions.

A. Formats

The standard defines basically five formats according to their base and the number of bits used to encode. In this there are three binary formats(32,64 and 128)and two decimal formats(64 and128).The first two binary format are said to be single precision and double Precision formats and the third is often called quad. The decimal formats are similarly said to be double and quad.

B. Single Precision

The 32 bit of IEEE 754 standard are said to be Single precision. In this, the most significant starts from left exponent and mantissa. The storage layout for sign, exponent and mantissa are shown below.

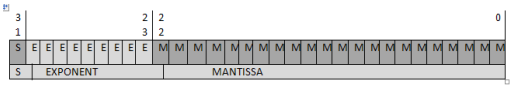

Figure1:format of single precision.

The sign bit (s) is either 0 for positive number or 1 for negative numbers. The exponent field represents both positive and negative exponents. To do this, a bias is added to the actual exponent. For IEEE single-precision format, this value is 127. The mantissa part is composed of implicit leading bit and the fraction bits, and mainly represents the precision bits of the number.

C. Normalized and De-Normalized Numbers

The number with biased exponent in its exponent field other than 0's and 1's is said to be normalized number. The hidden bit is appeared just left to decimal point is considered to be 1 and it is not in the representation.

The number with biased exponent in its exponent field are all 0's is said to be normalized number. The hidden bit is appeared just left to decimal point is always considered to be 0 and it is not in the representation.

D. Rounding modes

To make precision finite, some rounding modes are necessary. The standard mainly consists of five rounding modes. They are

- *Round towards* -∞*:*It round towards negative infinity and also called as round down. For example 5.2 is rounded to 5 and -4.2 is rounded to -5.
- *Round towards* +∞*:*It round towards positive infinity and also called as round up. For example 5.2 is rounded towards 6 and -4.2 is rounded towards -4.
- *Round towards zero:* It round toward nearest floating point number.
- *Round to nearest even:* It round towards near value with least significant bit.

E. Exceptions

In general, the standards consists of different types of exception that when exist one bit status flag is raised. They are overflow, underflow, invalid, divide by zero and inexact.

F. Guard digits

One method of computing the difference between two floating point numbers is to compute the difference exactly and then round it to the nearest floating point number. This is very expensive if the operands differ greatly in size.

IV. FUSED ADD-SUBTRACT UNIT

In many DSP algorithm and in some other fields, both addition and subtraction of two consecutive pair of operands are needed for processing. some other frequently used operations is to calculate the sum of products of two pairs of operands. These operations are needed in computing FFT and DFT butterfly structures.

In floating-point hardware unit, the DSP operations may be done in a serial , which reduces the throughput. other than, they may be perform in parallel, but which is expensive. Fused add-subtract unit(FAS) unit is introduced and normally it performs addition and subtraction is done in same unit.FAS unit performs the following operations such as

$$C = a + b \text{ and } D = a - b$$

The layout for the add-subtract is shown below.

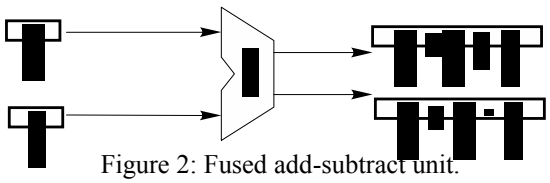

Figure 2: Fused add-subtract unit.

The standard specifies:

- Basic and extended floating-point number formats.
- Add, subtract, multiply, divide, square root, remainder, and compare operations
- Conversions between integer and floating-point formats.
- Conversions between different floating-point formats.
- Conversions between basic format floating-point

ISSN: 2278 – 909X All Rights Reserved © 2014 IJARECE

numbers and decimal strings.

• Floating-point exceptions and their handling, including non numbers when it comes to their precision and width in bits, the standard defines two groups: basic and extended format.

A. *Software used*

In this work VHDL implementation of IEEE 754 standard is used. VHDL is a versatile language in which we can effectively done the coding for all arithmetic operations, pack, unpack and rounding and the synthesis result are taken using ISE DESIGN SUITE 14.3. In this work coding for 32 bit addition is done for two operands.

V. RESULT

In this work two 32 bit is split into sign, exponent and mantissa part. If the sign bits are of 0 the addition operations take place otherwise subtraction is takes place. The two 8 bit exponent is compared and smallest exponent is calculated. The two mantissa part is concatenated with one guard bit and two rounding bit. The mantissa part is normalized by using shifter. The results are then obtained by adding two mantissa part. The addition of single precision is shown below.

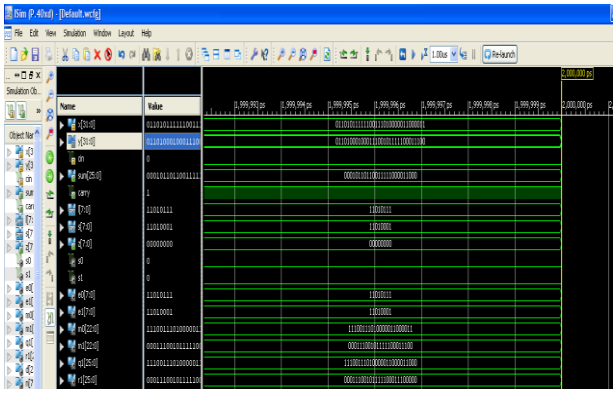

Figure 3: single precision addition.

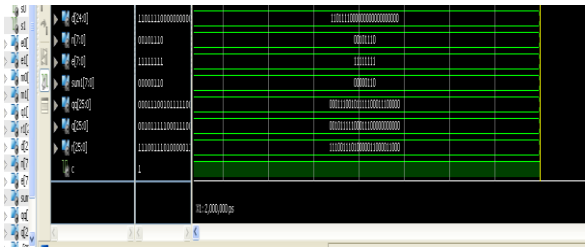

Figure 4:single precision addition.

VI. CONCLUSION

Arithmetic unit has been designed to perform rounding arithmetic operations, on floating point umbers. IEEE 754 standard based floating point representation has been used. The unit has been coded in VHDL. Code has been synthesized on xilinx 4.3 and has been implemented successfully. This standard provides a method for computation with floating-point numbers that will yield the same result whether the processing is done in hardware, software, or a combination of the two. The results of the computation will be identical, independent of implementation, given the same input data. Errors, and error conditions, in the mathematical processing will be reported in a consistent manner regardless of implementation.

VII. FUTURE SCOPE

The designed arithmetic unit operates on 32-bit operands. It can be designed for 64- bit operands to enhance precision. It can be extended to have more mathematical operations like addition, subtraction, division, square root etc.

REFERENCES

[1] Hani H. Saleh, Earl E. Swartzlander, Jr, "A Floating-Point Fused Dot-Product Unit," IEEE . PP 427–431, April- 2008.

[2] J.D. Bruguera, T. Lang. Floating—Point Multiply—Add Fused: Latency Reduction for Floating—Point Addition. Internal Report. University of Santiagode Compostela (2004).

[3] James, E., Michael J., "A Combined Two's Complement and Floating Point Comparator" Proceedings of the IEEE International Symposium on Circuits and Systems,vol.1,May2005,pages:89–92.

[4] Stehlé, D., Lefèvre, V., and Zimmermann, P., "Searching worst cases of a one-variable function", IEEE Transactions on Computers, 54(3), pp.340–346, 2005.

[5] Overton, M. L., Numerical Computing with IEEE Floating Point Arithmetic, ISBN 0-89871-571-7, Society for Industrial and Applied Mathematics 2001

[6] Muller, J.-M., Elementary Functions: Algorithms and Implementation, 2nd edition, Chapter 10, ISBN 0-8176-4372-9, Birkhäuser, 2006.

[7] The Unicode Standard, Version 5.0, The Unicode Consortium, Addison Wesley Professional,27October 2006,ISBN0-321-48091-0.54.

[8] Z. Zhao and M. Leeser, "Precision Modeling and Bit-width Optimization of Floating-Point Applications," MIT HPEC, 2003.

[9] Hani Saleh and Earl Swartzlander, Jr., "A Floating-Point Fused Dot Product Unit," IEEE International Conference on Computer Design (ICCD),Lake Tahoe,CA,pp.427-431,2008.

[10] The Unicode Standard, Version 5.0, The Unicode Consortium, AddisonWesley Professional, 27October 2006, ISBN 0-321-48091-0.54

[11] Quinnel, et al., Swartzlander, E., and Lemonds, C., "Floating-Point FusedMultiply-Add Architectures." Proceeding of the 41th Asilomar Conference on Signals, Systems and Computers, November 2007, Pages: 42 – 51.

[1]M.E. Applied Electronics, Bannari Amman Institute Of Technology.
[2]Assistant Professor, Dept.of ECE, Bannari Amman Institute Of Technology.
[3] Professor & Head, Dept.of ECE, Bannari Amman Institute Of Technology.
[4]M.E. VLSI Design, Bannari Amman Institute Of Technology.

ISSN: 2278 – 909X All Rights Reserved © 2014 IJARECE