A Review on Design Trade off Analysis of Floating Point Arithmetic Adder

Neelam Sharma[1] Anil Khandelwal[2]
[1]M.Tech Scholar
[1,2]VNS Group of Institution, Bhopal, India

Abstract— Most of the today's processor's require fast arithmetic operation with greater accuracy and relies on floating point arithmetic for numerical calculation. Addition and multiplication are the keys of floating point arithmetic realization at various levels in literature which is surveyed here. When we consider floating point unit, addition is at the top level in term of operation intricacy. The floating point unit provides with certain amount of delay with significant area. Many researchers have worked to reduce the overall latency. In order to design an efficient floating point adder the major requisite is improvement in performance, area and latency. Multiplication is the most important operation in the field of signal processing. Multiplication includes add and shift operation which requires large computation time. The main intent behind the design of floating point multiplier unit is to increase the speed so as to improve the performance of the signal processing application. A different design paradigm of floating point unit is compared here using Xilinx ISE 14.1.

Key words: Floating, Fused, Adder, Multiplier, CSLA, MUX, Power, Area

I. INTRODUCTION

Design of area- and power-efficient high- speed data path logic systems are one of the most substantial areas of research in VLSI system design. An arithmetic processor comprising: an arithmetic logic unit having a plurality of arithmetic circuits each for performing a group of associated arithmetic operations, such as finite field operations, or modular integer operations. Many fields of science, engineering and finance require manipulating real numbers efficiently. Since the first computers appeared, many different ways of approximating real numbers on it have been introduced One of them, the floating point arithmetic, is clearly the most efficient way of representing real numbers in computers. Representing an infinite, continuous set (real numbers) with a finite set (machine numbers) is not an easy task: some compromises must be found between speed, accuracy, ease of use and implementation and memory cost. Floating Point Arithmetic represent a very good compromise for most numerical applications.

II. STANDARD FLOATING POINT ADDER ALGORITHM

The standard architecture is the prototype algorithm for floating-point addition in any kind of hardware and software design [1]. Micro-architecture of the floating-point adder is shown in Fig. 1. The first step, not shown in Fig. 1, is to check whether the inputs are deformalized, infinity, or zero. These numbers are defined by special formats and standards, and VHDL comparators are used to identify them. The results are used to identify exceptions and are common to all algorithms. Next, the exponents are subtracted from one another to compute the absolute difference and identify the larger exponent. The mantissa of the number with the smaller exponent is right-shifted by the exponent difference and extended by 3 bits, to be used later for rounding; then the two mantissas are added using a two-complement adder. The next step is to detect the leading number of zeros before the first 1 in the result; this step is done by the module known as the leading-one detector (LOD). Using this value, the result is left-shifted by means of a left-shifter. When the result is negative and the operation is subtraction, the result is right shifted by 1. The last 5 bits are used to detect whether rounding is needed, and another adder is used to add a 1, yielding the resulting mantissa. The resulting exponent is computed by subtracting the leading-zeros amount from the larger exponent and adding a 1 when there is a right shift. The standard floating-point adder consists of five variable size integer adders and one right-shifter which can extend the result by 3 bits, named the guard (g), round (r), and sticky (s) bits. For post normalization, we need an LOD and a left shifter. All these modules add significant delay to the overall latency of the adder. Many scientific applications require high performance to carry out mathematical computation, almost all digital signal processor. These applications rely upon floating-point (FP) arithmetic for numerical calculations. Floating point formula is represented in equation (1). It gives real number to its estimate so that it the trade-off between range and precision get supported. Floating point representation having certain advantages over fix point representation. Floating-point number representations 1 render a ample dynamic range as compared to fix point and provide an exemption to DSP designers from overflow/underflow concerns that arises with fixedpoint representations [1]. IEEE 754 standard is the most widely used for computation of floating point arithmetic.:

III. PROPOSED ALGORITHM

The addition of floating point operation is difficult to perform because the signs are expressed in the sign magnitude format. Depending on sign of two operand addition or subtraction operation is performed. In case of adding a carry-out may be generated in such a case the result will be de-normalized. In subtraction, a negative result may be obtained which signifies that both the sign bit and the signs need to be inverted [3]. Floating point addition/subtraction algorithm:

A. Algorithm

The two operands, N1 and N2 are read in and compared for demoralization and infinity.

1) If numbers are deformalized, set the implicit bit to 0 otherwise it is set to 1. At this point, the fraction part is extended to 24 bits.
2) The two exponents, e1 and e2 are compared using 8-bit subtraction. If e1 is less than e2, N1 and N2 are swapped i.e. previous f2 will now be referred to as f1 and vice versa.

3) The smaller fraction, f2 is shifted right by the absolute difference result of the two exponents' subtraction. Now both the numbers have the same exponent.
4) The two signs are used to see whether the operation is a subtraction or an addition.
5) If the operation is a subtraction, the bits of the f2 are inverted.
6) Now the two fractions are added using a 2's complement adder.
7) If the result sum is a negative number, it has to be inverted and a 1 has to be added to the result.
8) The result is then passed through a leading one detector or leading zero counter. This is the first step in the normalization step.
9) Using the results from the leading one detector, the result is then shifted left to be normalized. In some cases, 1-bit right shift is needed.
10) The result is then rounded towards nearest even, the default rounding mode.
11) If the carry out from the rounding adder is 1, the result is left shifted by one.
12) Using the results from the leading one detector, the exponent is adjusted. The sign is computed and after overflow and underflow check, the result is registered.

IV. DESIGN PARADIGMS

Floating point arithmetic getting more popularized from the last few decades, due to its wide dynamic range and excellent strength against quantization error ability. This representation provides higher resolution and accuracy. A universal standard specified by IEEE for Floating-Point Arithmetic is ANSI/IEEE Standard 754-2008, New York:" IEEE, Inc. [2], 2008 describes interchange and arithmetic 0330 methods and formats for binary and decimal floating-point arithmetic in computer programming environments.

V. RELATED WORK

Amir Kaivani, et.al,[1] in their paper, "Floating Point Butterfly Architecture Based On Binary Signed Digit Representation" have proposed a fast FP (floating point) butterfly architecture, which is relatively quick than the previous work. The reason for the speed improvement is due to the FDPA unit suggested in this paper. Thus, by eliminating extra LZD, normalization and rounding units this high speed is achieved[1].

Jr., Hani et.al[2] in their paper, "FFT Implementation With Fused Floating Point Operations " have used the contrived idea of two new fused floating-point arithmetic units. Both the fused dot product unit and the fused add-subtract unit are smaller than collateral executions constructed with discrete floating point adders and multipliers[2]

Jongwook Sohn, et.al.3],in their paper "Improved Architectures for a Floating Point Fused Dot Product Unit" have introduced the better architecture designs and implementations for the floating-point fused two-term dot product unit. Overall performance of the circuit have been improved due to a new coalition scheme, early normalization and fast rounding and four-input LZA. The speed of proposed floating-point fused dot product unit have been improved through 15% as compared to the conventional floating-point fused dot product unit[3].

Yao Tao et.al.[4] in their paper, "Three Operand Floating Point Adder" have used Compound adder and LZA to get the most out of the architecture. An OR-logic network is implemented rather than the compared to observe the ruinous cancellation. The principle to select the internal width is given and the realignment method is used to obviate more than one stickybit rendered [4].

Jae Hong Et.al[5] Swartzlander, in their paper, "A Floating Point Fused FFT Butterfly Arithmetic Unit With Merged Multiple-Constant Multipliers" demonstrared reduced power and advanced speed floating point fused FFT architecture. The new MMCM butterfly architecture brings down the relative implicit error by 65% with marginally increased power consumption compared to the MCM implementation

Hani Saleh Et.al[6] in their paper, "A Floating-Point Fused Add- Subtract Unit" have used the fused add-subtract unit execute both of the ADD and SUB operations at about the same speed as a floatingpoint adder with less than 40% area overhead.

Alexandre F.Tenca Et.al[7]in his paper, "Multi – Operand Floating Point Addition" concentrated to a greater extent to accurate output than FPADD2 and that devolving on number of input operands, adder can have issue of area and delay. The experimental results show that the delays can be minimized or done shorter in 3-input FP adder design than the network of 2-input FP adders, with equal or better area, and much accuracy

Hani H. Saleh, Et.al[4]. in their paper, "Fused Floating Point Arithmetic For DSP" have concluded that the fused dot product is mediate in the area between the conventional serial and collateral approaches. Its latency is about 80% with respect to that of the formal parallel approach and about half that of the formal serial approach[8]

G. Jaberipur, B. Parhami, Et.al[4]. in their paper, "Efficient realisation of arithmetic algorithms with weighted collection of posibits and negabits" showed advantages which involve intermingling posibits and negabits as components of weighted bit lot, in order to use in symbolizing signed digit lot or for direct symbolizing of integers in a coveted range. Thus, the benefits of such implementation reduces the design effort and improves the design parametric quantities such as cost, speed and/or power usage [9].

Ishan A. Patil, Et.al[4]., in their paper, "Floating Point-based Universal Fused Add Subtract Unit" generated FP adder, which is indeed absolute modified version acquired by studying traditional floating point adder. Use of verilog design programs have been preferred in this paper. Less number of LUTs, reduced delay, less use of IOBs are the main highlights of this paper

REFERENCES

[1] Amir Kaivani and Seokbum Ko "Floating-Point Architecture Based on Binary Signed-Digit Representation" IEEE Transaction on VLSI Systems, volume 24., no.3, pp.1208-1211, March 2016
[2] E. E. Swartzlander, Jr. and H. H. Saleh "FFT Implementation with Fused Floating-Point Operations"

IEEE Transaction On Computers, volume 61., no. 2, pp. 284-288, Feb. 2012

[3] J. Sohn and E. E. Swartzlander, Jr. "Improved Architectures for a Floating-Point Fused Dot Product Unit" in Proc. IEEE 21st Symposium On Computer Arithmetic, pp.41-48 , Apr. 2013.

[4] Y.Tao, G. Deyuan, F. Xiaoya, and R. Xianglong "Three-operand floating-point adder" in Proc. 12th IEEE International Conference of Computer Inf. Technology, October 2012,pp. 192-196

[5] J. H. Min, S.-W. Kim and E. E. Swartzlander, Jr. "A Floating-Point Fused FFT Butterfly Arithmetic Unit with Merged Multiple-Constant Multipliers" in Proc. 45th Asilomar Conference on Signals, Systems and Computers, pp. 520–524, Nov. 2011

[6] Hani Saleh and Earl E. Swartzlander, Jr., "A Floating-Point Fused Add- Subtract Unit" Proc. IEEE Midwest Symp. Circuits and Systems (MWSCAS), pp. 519-522, 2008

[7] A. F. Tenca, "Multi-operand floating-point addition" in Proc. 19th IEEE Symp. Comput. Arithmetic, Jun. 2009, pp. 161-168.

[8] H. H. Saleh, "Fused Floating-Point Arithmetic for DSP", PhD dissertation, Univ. of Texas, 2008.

[9] G. Jaberipur, B. Parhami, "Efficient realisation of arithmetic
algorithms with weighted collection of posibits and negabits"IET Comput. Digit. Tech. 2012, Vol. 6, ISS. 5, pp. 259-268, Jan. 2012

[10] Ishan A. Patil, Prasanna Palsodkar, Ajay Gurjar, "Floating Point-based Universal Fused Add-Subtract Unit" Proc. Of Second International Conference on Soft Computing for Problem Solving, pp. 259-270, December 28-30 2012.

