p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 08
July 2017

Design and Analysis of Area and Delay Efficient Double Precision Floating -Point Adder

V.Suresh [1] Mr. G.Malyadri [2]

1. M. Tech Student, Department of ECE, KKR & KSR Institute of Technology & Sciences (KITS), Guntur.

2. Associate Professor, Department of ECE, KKR & KSR Institute of Technology & Sciences (KITS), Guntur.

1 sureshvangipurapu@gmail.com 2 malyadri_gmr@yahoo.co.in

Abstract

In the fields of scientific applications because of dynamic representation capabilities and large spectrum of numbers can be represented with limited number of bits, floating-point numbers are being widely adapted. A floating-point arithmetic unit is specifically designed to carry out on floating-point numbers and in the area of binary applications it is one of the most common parts of any computing system. In signal processing and embedded platforms the critically important components are floating-point adders and Floating-point additions which are the most frequent floating-point operations. The survey of related works of different algorithms/techniques which are important for implementation of double precision floating point adder with reduced delay based on FPGAs are presented in this paper. In this paper, by approximately designing of an exponent subtractor and mantissa adder, an area and delay efficient floating-point adder is proposed. Related operations such as normalization and rounding are also dealt with in terms of inexact computing.

Keywords: Double Precision, floating-point adders, area Efficient.

1. Introduction

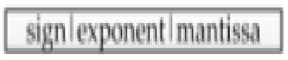
Lot of research has been done to get the accurate answers from the past two decades in many numerical computations. This work is dedicated to get a maximum of accuracy which is developed on FPGA. Of course many numerical applications utilized double precision format. Several research works are already done in floating Point operations. For better accuracy and precision, the highest precision is to be taken but even then in the double precision format also will show the error. So in floating point arithmetic operations maintaining the accuracy is difficult in previous adders. Many serial components like right shifter or left shifter and the floating point addition (FPA) have been taken a longer latency. The Floating point (FP) adder required to have to be speedy in order to match with the increasing clock rate demand. In general the conventional floating point adders perform the computation in a single clock cycle. For that cause clock rates would be lower and lower. Thus in order to perform a sequential summations or sequential computations the conventional adder is incompetent. So pipelining is the technique is necessary to overcome this limitation on clock frequency. Pipelining means the instructions are executed sequentially, finally, the last output will come one by one. The speed of the operation will achieve by increasing the pipelines. The Pipelining techniques are applied in FPA in order to speed up the numerical arithmetic operation and to increase the through put. Now- a -days everybody try to increase the clock speed this means that according to, Moore's Law feature size scaling is increasing exponentially in transistor per integrated chip. So the industry is now might be reaches that end point, the focus is now transfer to enhances with parallelism in computations more willingly than clock speed.

Each IEEE-754 standard floating-point number system has a specific precision like single precision or double precision and quadruple precision which is comprises of sign bit, exponent and significant or fractional bits. But the bits will vary depending on the precision. In single precision one sign bit, 8-exponent bits, 23-significand bits with implied one. In case of double precision, one sign bit, 11-exponent bits, 53- significand bits including one hidden bit by the standard. Exponent must be greater than zero and less than 1023.In this paper, we focus on the problem of summing two double-precision FP values, but when we rearrange the order of numbers sum would produce a miscellaneous result. This is an alternative to be deprived of parallelism in order to increase the clock speed.

The sequential summation operation will always give the similar result, but it may still be an erroneous one in case of shuffled order. In this case parallelizations and accuracy may get failed. When we know the error we have a freedom to correct the answer. Based on for FP addition using pipelining technique we made an effort still guaranteeing an ideal result and of course acceptably rounded deterministic result in this paper. The design of the accurate floating point unit or FPA unit to speed up the computations for many scientific applications is thus of interest in this domain. In residue preserving addition most of the algorithms such as [2]– [4] rely on the similar basic building block that is studied in detail by Kornerup et al.

2. Background

The FP format typically contains a sign bit, the exponent and the mantissa fields (commonly represented as a string from left to right). It offers a higher dynamic range than a fixed-point format to represent real numbers. However, the FP hardware is both more complex and consumes significant power. The most commonly used standard for the FP format is the IEEE 754-2008. There are basic and extended types that are supported by this standard: half precision (16 bits), single precision (32 bits), double precision (64 bits), extended precision (80 bits) and quad precision (128 bits). A general IEEE FP format is shown in Fig. 1. The exponent part has a bias of 2E-1-1, where E is the number of exponent bits. The single precision and double precision formats are mostly used in today’s computers.



$$FP\ No. = (-1)^{s} \times 2^{\text{exponent}-\text{bias}} \times (1 + \text{mantissa})$$

Fig. 1 General IEEE 754 FP format

p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 08
July 2017

3. Existing Floating-Point Adder Architecture

A generic FP adder architecture includes hardware blocks for exponent comparison, mantissa alignment, mantissa addition, normalization and rounding of the mantissa (shown in Fig. 2 and detailed in [7]). Two operands are first unpacked from the FP format, and each mantissa is added to the hidden '1' bit. The addition of FP numbers involves comparing the two exponents and adding the two mantissas; the exponents are first evaluated to find the larger number. The mantissas are then swapped according to the exponent comparison; they are then aligned to have an equal exponent prior to the addition in the mantissa adder. Following the addition, normalization shifts are required to restore the result to the IEEE standard format. The normalization is completed by left shifting with a number of leading zeros; therefore, leading zero detection is a key step for normalization. Rounding the normalized result is the last step before storing back the result; special cases (such as overflow, underflow, and not a number) are also detected and represented by flags.

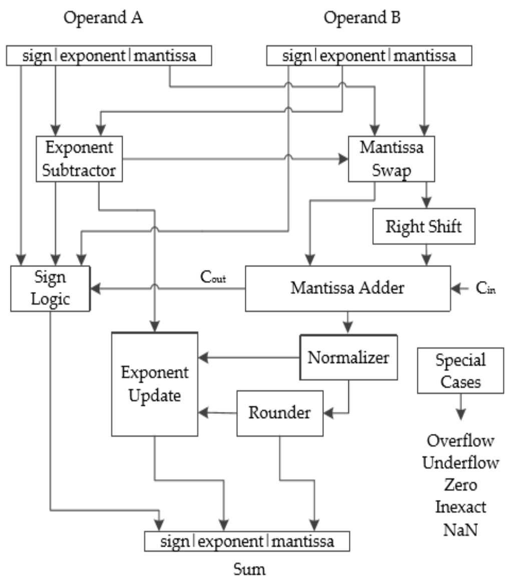


Fig. 2 The accurate FP adder architecture

4. Design of Inexact Floating-Point Adders

The inexact design of an FP adder originates at an architectural level (Fig. 2). It consists of designing both the mantissa adder and exponent subtractor by using approximate fixed-point adders. At the same time, related logic including the normalizer and the rounder should also be considered according to the inexact mantissa and exponent parts. The circuit level inexact designs are discussed in detail in the following subsections. 3.1 Exponent Subtractor The exponent subtractor is used for exponent comparison and can be implemented as an adder. An inexact fixed-point adder has been extensively studied and can be used in the exponent adder; inexact adders such as lower-part-OR adders (LOA) [8], approximate mirror adders [9], approximate XOR/XNOR-based adders [10], and equal segmentation adders [11] [12] can be found in the literature. For a fast FP adder, a revised LOA adder is used, because it significantly reduces the critical path by ignoring the lower carry bits.

A k-bit LOA consists of two parts, i.e., an m-bit exact adder and an n-bit inexact adder (Fig. 3). The m-bit adder is used for the m most significant bits of the sum, while the n-bit adder consists of OR gates to compute the addition of the least significant n bits (i.e., the lower n-bit adder is an array of n 2-input OR gates).

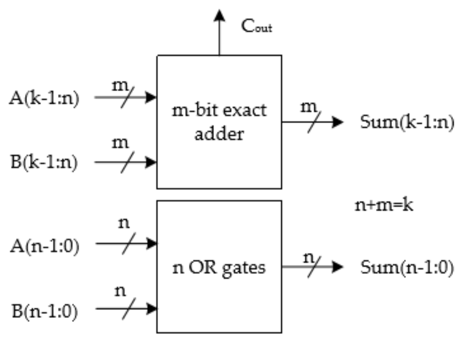


Fig. 3 The revised LOA adder structure

In the original LOA design, an additional AND gate is used for generating the most significant carry bit of the n-bit adder; in this work, all carry bits in the n-bit inexact adder are ignored to further reduce the critical path.

The exponent is dominant in the FP format, because it determines the dynamic range. The approximate design of the exponent subtractor must be carefully considered due to its importance in the number format. The results of the addition are significantly affected by applying an approximate design to only a few of the least significant bits of the exponent subtractor under a small data range.

4.1 Mantissa Adder

The revised LOA adder can also be used in the mantissa adder for an inexact design. Compared to an exponent subtractor, the mantissa adder offers a larger design space for inexact design, because the number of bits inthe mantissa adder is significantly larger than the exponent subtractor. As shown in Table I, the number of mantissa bits is larger than the number of exponent bits. For the IEEE single precision format, the exponent subtractor is an 8-bit adder, while the mantissa adder is a 25-bit adder (for two 24-bit significances).

Table I NO. Of Exponent and Mantissa Bits for the IEEE 754 Basic and Extended FP Types.

Type	Sign Bit	Exp. Bits	Mant. Bits	Total	Mant. Bits/Total
Half	1	5	10	16	62.5%
Single	1	8	23	32	71.9%
Double	1	11	52	64	81.2%
Extended	1	15	64	80	80.0%
Quad	1	15	112	128	87.5%

Furthermore, the inexact design in the mantissa adder has a lower impact on the error than its exponent counterpart in the lower data range, because the mantissa part is less significant than the exponent part.

4.2 Normalizer

Normalization is required to ensure that the addition results fall in the correct range; the sum or difference may be too small and a multi-bit left shift process may be required. A reduction of the exponent is also necessary. The normalization is performed by a leading zeros counter that determines the required number of left shifts. As the mantissa adder is already not exact for the n least significant bits, the detection of the leading zeros can also be simplified in the inexact design, i.e., approximate leading zero counting logic can be used.

4.3 Rounder

A rounding mode is required to accommodate the inexact number that an FP format can represent. A proper rounding maintains three extra bits (i.e., guard bit, round bit and sticky bit). The adder may require a further normalization and exponent adjustment after the rounding step, therefore the hardware for rounding is significant.

However, it does not affect the results of the inexact addition as the lower significant n bits are already inexact. Therefore, rounding can be ignored in the inexact design of an FP adder.

4.4 Overall Inexact FP Adder Architecture

Based on the previous discussion, an inexact FP adder can be designed by using approximate adders in the exponent subtractor and mantissa adders, an approximate leading zero counter in the normalizer and by ignoring the rounder. The inexact FP adder architecture is shown in Fig. 4.

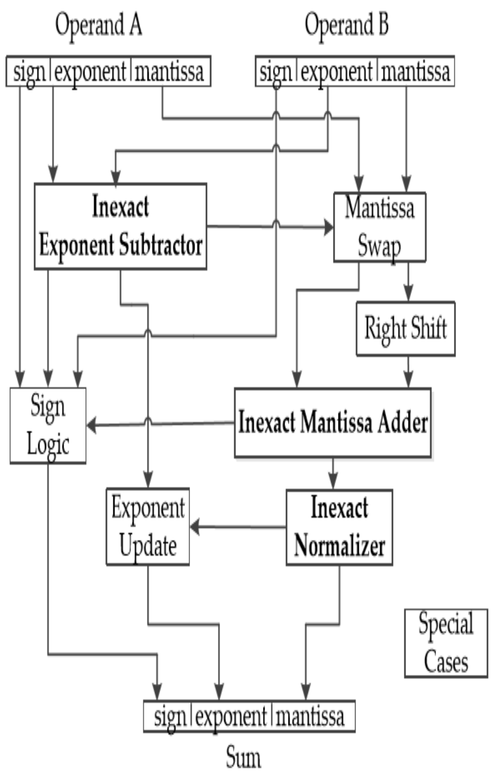


Fig. 4 The Inexact FP adder architecture

5. Results

5.1 RTL Schematic Diagram

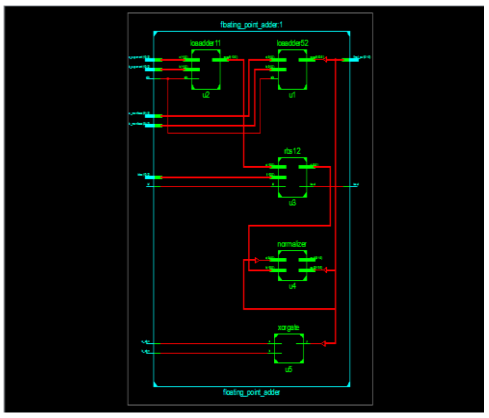


Fig 5 RTL Schematic of Inexact Double Precision Floating Point Adder

5.2 Comparison Table

Table II Comparison of Delay and area for Conventional and Proposed Double Precision Floating Point adder units

Architecture	LUT's	Area (Kb)	Delay (ns)
Existing	140	193448	54.696
Proposed	108	193256	31.236

Conclusion

This project presents the implementation of double precision inexact floating point adder. The whole design was captured in Verilog HDL, tested in simulation using Model Tech's Modelsim, placed and routed on a Spartern 3E FPGA from Xilinx 13.2. Two extreme cases for the inexact design of FP adders have been studied. The first design uses an all-bit inexact mantissa adder; the second design uses an inexact LSB in the exponent subtraction. Thesecond designtakes a small area and less delay and offer higher performance than the firstdesign. As such thisis suitable for high dynamic image applications. It has been shown that the exponent part is a dominant part in the FP number format; however it has a smaller design space for an inexact design compared to the mantissa adder.

References

[1]. M. V. Manoukian and G. A. Constantinides, "Accurate Floating point arithmetic through hardware error-free transformations," in Proc. Intl. Conf. on Reconf. Comp. Springer-Verlag, 2011, pp. 94–101.

[2]. I. J. Anderson, "A distillation algorithm for floatingpoint summation," SIAM J. Sci. Comput, vol. 20, pp. 1797–1806, 1999.

[3] Y.-K. Zhu and W. B. Hayes, "Correct rounding and a hybrid approach to exact floating-point summation," SIAM J. Sci.Comput., vol. 31, no. 4, pp. 2981–3001, July 2009.

[4] S. M. Rump, "Ultimately fast accurate summation," SIAM J.Sci. Comput., vol. 31, no. 5, pp. 3466–3502, September 2009.

[5]. P. Kornerup, V. Lef`evre, N. Louvet, and J.-M. Muller, "On the computation of correctly rounded sums," IEEE Trans.Comput., vol. 61, no. 3, pp. 289–298, March 2012.

[6].IEEE Standard for Floating-Point Arithmetic,"IEEE Std 7542008, Aug. 29 2008,doi:10.1109/IEEESTD.2008.4610935.

[7] B. Parhami, Computer arithmetic: algorithms and hardware designs. Oxford University Press, Inc., 2009.

[8] H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, ‚Bioinspired imprecise computational blocks for efficient VLSI implementation of soft-computing

applications,‛ IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, pp. 850-862, 2010.

[9] V. Gupta, D. Mohapatra, S. Park, A. Raghunathan, and K. Roy,IMPACT: IMPrecise Adders for Low-Power Approximate Computing,‛ Proc. Int. Symp. Low Power Electronics and Design (ISLPED), pp. 1-3, 2011.

[10] Z. Yang, A. Jain, J. Liang, J. Han and F. Lombardi, ‚Approximate XOR XNOR-based Adders for Inexact Computing‛, Proc. 13rd IEEE Conf. Nanotechnol. (IEEE-NANO), pp. 690-693, 2013.

[11] D. Mohapatra, V. Chippa, A. Raghunathan, and K. Roy, ‚Design of voltage-scalable meta-functions for approximate computing‛, Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1-6, 2011

[12] C. Liu, J. Han, F. Lombardi, ‚An Analytical Framework for Evaluating the Error Characteristics of Approximate Adders‛ IEEE Trans. Computers, preprint, 14Apr.2014,doi:10.1109/TC.2014.231718 0.