Design and Implementation of Low Power Inexact Floating Point Adder

Kamlesh Pedraj, Jayendra Kumar
Dept. of ECE, National Institute of
Technology Jamshedpur, India

ABSTRACT
Floating-point applications are a growing trend in the FPGA community. In nanoscale integrated circuits design as the demand for mobile computing & higher integration density is increasing power is becoming a very important constraint. Low-power is an imperative requirement for portable multimedia devices employing various signal processing algorithms and architectures. For some applications where error is in tolerable range an inexact circuit offers reduction in both static and dynamic power .In this paper, an inexact floating-point adder is designed by approximating exponent sub tractor and mantissa adder. Related operations such as normalization and rounding are also dealt with in terms of inexact computing. It is then observed that it greatly reduced the power consumption and hence increased the reliability.

Keywords
Floating-point adders, low power, high dynamic range image, inexact circuits, error analysis.

1. INTRODUCTION
In digital integrated circuits as the technology is getting advanced and innovative day by day, power consumption is also increasing dramatically, saving power is high in demand as it will reduce the overall cost for mobile computing and higher integration density. Present designs offers fully accurate computing to all of its application but there are some error tolerant applications including human intervention (for example image processing) in which error can be tolerated, does not require full accuracy [1].

So, in such applications we can perform the computations with inexact circuits, as inexact computing is the recommended way to save power area and hence cost of implementation with better results in terms of performance as compared to the exact computing. A processor's core is its arithmetic unit and power consumed by it is the big percentage of the power consumed by the whole processor. According to a recent study on inexact adders, inexact processing hardware with error in the tolerable range of relative error of 7.58 % can be approximately 15 times more efficient than an exact chip in terms of speed, energy product & area [2]. Inexact chips are smaller, faster and consume less energy. For inexact computing, fixed point arithmetic circuits have been already studied[2],[3],[4],[5],[6],[7],[8] as mentioned in the literature floating-point (FP) circuits consumes significantly more power due to its more complexity they have not been recommended for inexact computing. In computationally intensive applications FP format offers a large range dynamically. In DSP systems FP multipliers & adders have extensive uses. However, it is not used in embedded system because of its limitation of high power consumption. A design of a low power FP multiplier was investigated by Tong et al [10] this design includes truncation of hardware & a reduction of the bit width representation of the FP data. A probable FP multiplier design was given by Gupta et al [11]. as an energy efficient design. According to the authors knowledge there is very less research on inexact FP adders.A lightweight FP design flow using bit-width optimization was proposed for low power signal processing applications [12]. In this paper, adder designs are studied with this different inexact adders are also studied and several inexact adder designs are proposed and assessed. The adder is then implemented in the software and then dumped on FPGA the results are then analyzed.

2. FIXED POINT AND FLOATING POINT REPRESENTATIONS
Every real number has an integer part and a fraction part a radix point is used to differentiate between them the number of binary digits assigned to the integer part may be different to the number of digits assigned to the fractional part. A generic binary representation with decimal conversion is shown in Figure 1.

	Integer Part					Binary Point	Fraction Part			
Binary	---	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}	---
Decimal		8	4	2	1		½	¼	⅛	

Figure1: Binary representation and conversion to decimal of a numeric

2.1 Basic Format
There are two basic formats described in IEEE 754 format, double-precision using 64-bits and single-precision using 32-bits. Table 1 shows the comparison between the important aspects of the two representations.

Table 1: Single and double precision format summary

Format	Precision (*p*)	E_{max}	E_{min}	Exponent bias	Exponent width	Format width
Single	24	+127	-126	127	8	32
Double	53	+1023	-1022	1023	11	64

To evaluate different adder algorithms, we are only interested in single precision format. Single-precision format uses 1-bit for sign bit, 8-bitsfor exponent and 23-bits to represent the fraction as shown in Figure 2.

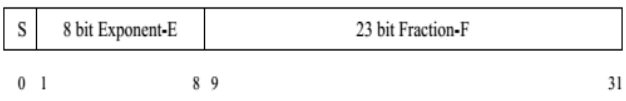

Figure 2: IEEE 754 single precision format.

The single- precision floating-point number is calculated as $(-1)^s \times (1+F) \times 2^{(e-127)}$. The sign bit is either 0 for non-negative 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, for example, a stored value

of 200 indicates an exponent of (200-127), or 73. The mantissa or significand is composed of an implicit leading bit and the fraction bits, and represents the precision bits of the number. Exponent values (hexadecimal) of 0xFF and 0x00 are reserved to encode special numbers such as zero, denormalized numbers, infinity, and NaNs. The mapping from an encoding of a single-precision floating-point number to the number's value is summarized in Table 2.

Table 2: IEEE 754 single precision floating-point encoding

Sign	Exponent	Fraction	Value	Description
S	0xFF	0x00000000	$(-1)^S \infty$	Infinity
S	0xFF	F≠0	*NaN*	Not a Number
S	0x00	0x00000000	0	Zero
S	0x00	F≠0	$(-1)^S \times 0.F \times 2^{(E-126)}$	Denormalized Number
S	0x00 < E < 0xFF	F	$(-1)^S \times 1.F \times 2^{(E-127)}$	Normalized Number

2.2 Standard Floating Point Addition Algorithm

This section will review the standard floating point algorithm architecture, and the hardware modules designed as part of this algorithm, including their function, structure, and use. The standard architecture is the baseline algorithm for floating-point addition in any kind of hardware and software design.

3. BACKGROUND

An inherent problem of binary floating-point arithmetic used in financial calculations is that most decimal floating point numbers cannot be represented exactly in binary floating-point formats, and errors that are not acceptable may occur in the course of the computation. Decimal floating-point arithmetic addresses this problem, but a degradation in performance will occur compared to binary floating-point operations implemented in hardware. Despite its performance disadvantage, decimal floating-point arithmetic is required by certain applications that need results identical to those calculated by hand.

This is true for currency conversion, banking, billing, and other financial applications. Sometimes, these requirements are mandated by law; other times, they are necessary to avoid large accounting discrepancies. Because of the importance of this problem a number of decimal solutions exist, both hardware and software. Software solutions include C#, COBOL, and XML, which provide decimal operations and datatypes. Also, Java and C/C++ both have packages, called Big Decimal and decimal Number, respectively. Hardware solutions were more prominent earlier in the computer age with the ENIAC and UNIVAC. However, more recent examples include the CADAC, IBM's z900 and z9 architectures, and numerous other proposed hardware implementations. More hardware examples can be found, and a more in-depth discussion is found in Wang's work.

4. DESIGN TRADEOFF ANALYSIS OF FLOATING-POINT ADDER IN FPGA

Field Programmable Gate Arrays (FPGA) are increasingly being used to design high end computationally intense microprocessors capable of handling both fixed and floating point mathematical operations. Addition is the most complex operation in a floating-point unit and offers major delay while taking significant area. Over the years, the VLSI community has developed many floating-point adder algorithms mainly aimed to reduce the overall latency. An efficient design of floating-point adder onto an FPGA offers major area and performance overheads. With the recent advancement in FPGA architecture and area density, latency has been the main focus of attention in order to improve performance. Our research was oriented towards studying and implementing standard, LOA adder, and far and close data-path floating-point addition algorithms. Each algorithm has complex sub-operations which lead significantly to overall latency of the design. Each of the sub-operation is researched for different implementations and then synthesized onto a Xilinx Virtex2p FPGA device to be chosen for best performance.

This paper discusses in detail the best possible FPGA implementation for algorithm and will act as an important design resource. The performance criterion is latency in all the cases. The algorithms are compared for overall latency, area, and levels of logic and analyzed specifically for Virtex2p architecture, one of the latest FPGA architectures provided by Xilinx. According tour results standard algorithm is the best implementation with respect to area but has overall large latency of 27.059 ns while occupying 541 slices. LOP algorithm improves latency by 6.5% on added expense of 38% area compared to standard algorithm.

5. PROPOSED SYSTEM

5.1 Design of Inexact Floating-Point Adders:

The design of an inexact floating point adder originates at its architectural level. It includes designing of both mantissa adder & inexact exponent subtractor by using inexact fixed point adders. On the other hand, other related blocks including rounder and normalizer should be designed in consideration with inexact mantissa and exponent part. Detail design of inexact circuit is explained in depth in the sections below.

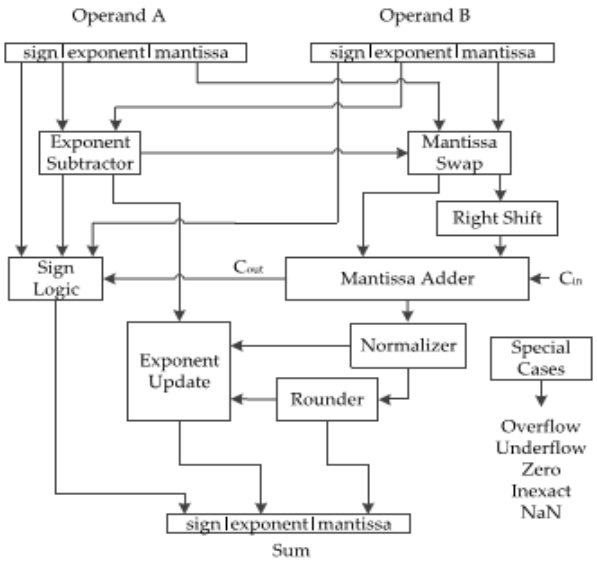

Fig.3. The accurate FP adder architecture

5.2 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) [3], approximate mirror adders[4], approximate XOR/XNOR-based adders, and equal segmentation adders[6], [7] can be found in the literature [1,2,3]. 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. The m-bit adder is used for the m most significant bits of the sum, while then-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 two-input OR gates) [6]. In the original LOA design, an additional AND gate is used for generating the most significant carry bit of then-bit adder; in this work, all carry bits in then-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.

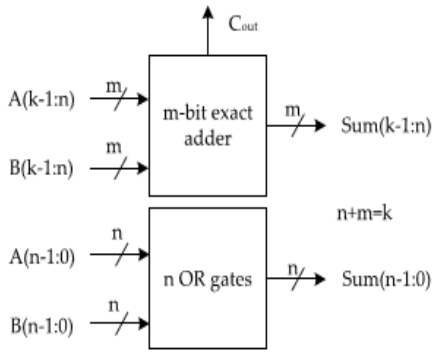

Fig4.The revised LOA adder structure.

5.2 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 in the mantissa adder is significantly larger than the exponent subtractor. As shown in Table 1, the number of mantissa bits is larger than the number of exponent bits [7] .For the IEEE single precision format, the exponent subtractor is an 8-bit adder, while the mantissa adder is a 25-bit adder (for two24-bit significances). 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. Therefore, an inexact design of a mantissa adder is more appropriate. A detailed analysis of errors introduced by each part is further discussed in the next section.

5.3 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 then 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.

5.4 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.

5.5 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. 5.

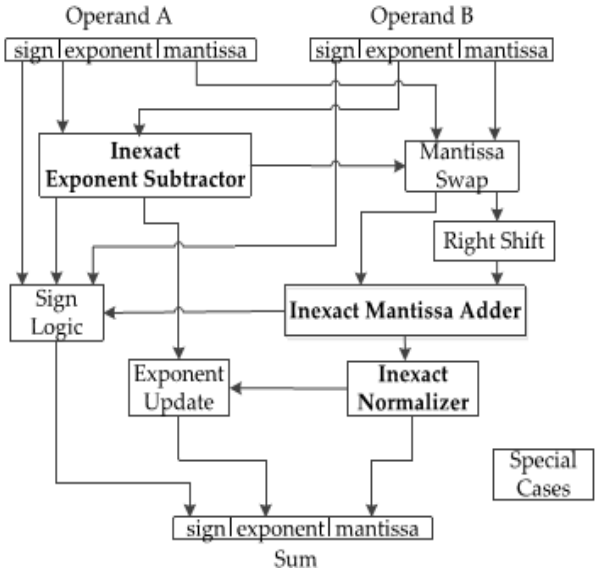

Fig5. The inexact FP adder architecture.

6. RESULTS

The floating point adder Verilog HDL Modules have successfully simulated and verified using Modelsim6.4b and synthesized using Xilinxise10.1.

6.1 Simulation Result:

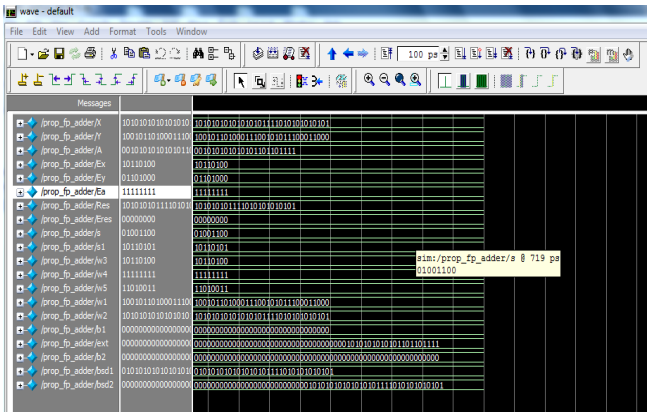

Fig 6 simulation results of inexact floating point adder

6.2 Synthesis Results

6.2.1 RTL Schematic:

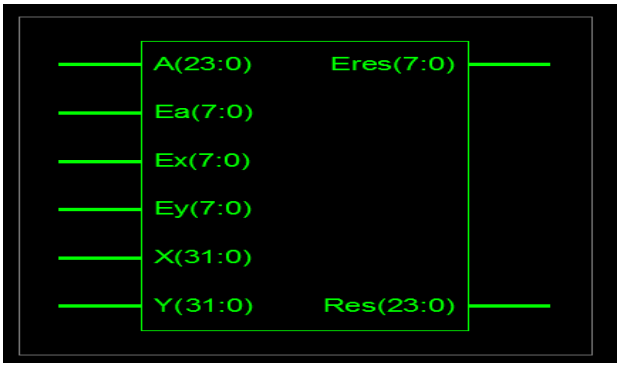

Fig 7. RTL schematic of inexact floating point adder

6.2.2 Technology Schematic:

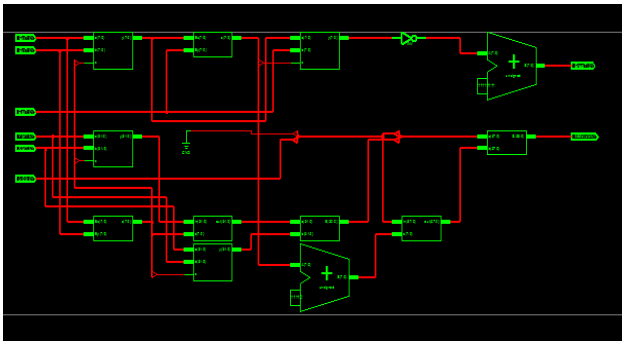

Fig 8. Technology schematic diagram of inexact floating point adder

6.3 Design Summary

Device utilization summary is shown in the table given below:

Logic utilization	Used	Available	Utilization (%)
Number of slices	189	3584	5
Number of four input LUTs	331	7168	4
Number of bonded IOBs	128	221	57

Fig 9. Design summary of inexact floating point adder

7. CONCLUSION

Design of inexact floating point adder has been studied in this paper. Approximate designs of the mantissa and exponent adders have been proposed and consideration has been given to normalization and rounding. 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. The results have shown that both inexact FP adders are very low power designs. These designs require a small area and offer higher performance than their equivalent exact designs. As such they are 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.

8. REFERENCES

[1] Liu, Weiqiang, et al. "Design and analysis of inexact floating-point adders." IEEE Transactions on Computers 65.1 (2016): 308-314.

[2] K. Palem and A. Lingamneni, “Ten years of building broken chips: The physics and engineering of inexact computing,”ACM Trans. Embedded Comput. Syst., vol. 12, no. 2, article 87, 2013.

[3] A. Lingamneni, K. Muntimadugu, C. Enz, R. Karp, K. Palem, and C. Piguet,“Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling,” inProc. ACM Int. Conf. Comput. Frontiers, 2012,pp. 3–12.

[4] H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, “Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft computing applications,”IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57,no. 4, pp. 850–862, Apr. 2010.

[5] V. Gupta, D. Mohapatra, S. Park, A. Raghunathan, and K. Roy, “IMPACT: Imprecise adders for low-power approximate computing,” in Proc. Int.Symp. Low Power Electron. Des., 2011, pp. 1–3.

[6] Z. Yang, A. Jain, J. Liang, J. Han and F. Lombardi, “Approximate XORXNOR-based adders for inexact computing,” inProc. 13rd IEEE Conf. Nanotechnol., 2013, pp. 690–693.

[7] D. Mohapatra, V. Chippa, A. Raghunathan, and K. Roy, “Design of voltage scalable meta-functions for approximate computing,” inProc. Des.,Autom.Test Eur. Conf. Exhib., 2011, pp. 1–6.

[8] C. Liu, J. Han, and F. Lombardi, “An analytical framework for evaluating the error characteristics of approximate adders,” IEEE Trans. Comput.,vol. 64, no. 5, pp. 1268–1281, May 2015.

[9] C. Liu, J. Han, and F. Lombardi, “A low-power, high-performance approximate multiplier with configurable partial error recovery,” inProc. Design,Autom. Test Eur. Conf. Exhib., 2014, pp. 1–4.

[10] J. Y. Tong, D. Nagle, and R. Rutenbar, “Reducing power by optimizing then necessary precision/range of floating-point arithmetic, ”IEEE Trans. Very Large Scale Integer. Syst., vol. 8, no. 3, pp. 273–286, Jun. 2000.

[11] A. Gupta, S. Mandavalli, V. Mooney, K. Ling, A. Basu, H. Johan, and B.Tandianus, “Low power probabilistic floating-point multiplier design,” in Proc. IEEE Comput. Soc. Annu. Symp. VLSI, 2011, pp. 182–187.

[12] F. Fang, T. Chen, and R. Rutenbar, “Floating-point bit-width optimizationfor low-power signal processing applications,” in Proc. IEEE Int. Conf.Acoust., Speech, Signal Process., 2002, vol. 3, pp. 3208–3211.

[13] W. Liu, L, Chen, C. Wang, M. O’Neill, and F. Lombardi, “Inexact floatingpoint adder for dynamic image processing,” in Proc. 14th IEEE Conf. Nanotechnol., 2014, pp. 239–243.

[14] IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2008, Aug. 29,2008.

[15] J. Liang, J. Han, and F. Lombardi, “New metrics for the reliability of approximate and probabilistic adders,” IEEE Trans.Comput., vol. 62, no. 9, pp. 1760–1771, Sep. 2013.