IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 5, May 2017

Design and Study of Floating Point Adders using Parallel Prefix Adders

Asif Hussain C V[1], R Dharmalingam[2]

ME VLSI Department, Maharaja Institute of Technology, Coimbatore[1]

ME VLSI, Head of the Department, Maharaja Institute of Technology, Coimbatore[2]

Abstract: In this paper, we propose 32 bit types of Brent-Kung, Kogge-Stone and a Ladner-Fischer parallel prefix adders. In general N-bit adders like Ripple Carry Adders (very inefficient processes of adders compare to other adders), and Carry Look Ahead adders (area consuming adders) are used in earlier days. As existing methods of an emerging computational paradigm, an inexact circuit offers a promising method to significantly reductions of the both dynamic and static power dissipation for error-tolerant applications. an inexact floating-point adder is by approximately designing an exponent sub-tractor and mantissa adder. But now the most integrated circuits are using parallel prefix adders because of their advantages compare to other adders. Parallel prefix adders are faster and area efficient. We simulate and synthesis different types of 32-bit prefix adders using Xilinx ISE 10.1i tool. We using these systems out performances, we noted the performance parameters like number of LUTs and delay.

Keywords: Inexact circuits, floating-point adders prefix adder, carry operator.

I. INTRODUCTION

In highly integrated nano scale designs, accuracy problems resulting from PVT (process level, voltage and temperature) variations, aging effects and soft errors have become major barrier for leveraging the benefits of a lower device scaling; moreover, wastage of power and static power are significant concerns for the high power expenditures encountered at such high density. A potential solution to lower power bender is to employ almost circuit designs.

Arithmetic circuits are the ones which perform mathematical operations like addition, subtraction, multiplication, division, parity calculation. Most of the time, making and modeling these circuits are the same as architecture bases muxers, encoders and decoders. In digital electron devices, an adder or summer is a digitalized made of circuits that produce addition of numbers. In many computers and other types of system processors, adders are other parts of the processor, many digital systems and other types of processors the super computers, where they are used to work out addresses, table and similar. The binary circuits of an adder is the one type of element in most digital circuit designs that are including digital signal processors (DSP) and micro signal processor data way of units. Therefore very speed and exact operation of digital system depends on the performance of adders. Hence increasing the results of outperformance of an adder is the main area of research in VLSI system design. The Conventional adders discussed in section.

Floating-Point Adder and Architecture

The inputs and output of the floating-point adder are assumed to be encoded in the IEEE 754 single precision format. The IEEE format requires 32 bits, i.e. 1-bit sign, 8-bit exponent and 23-bit mantissa. A generic floating-point adder architecture includes exponent comparison, mantissa swap and alignment, mantissa addition, normalization and rounding of the mantissa, as shown in Fig. (and detailed in [8]).

Two operands are first unpacked from the comparison, mantissa swap and alignment, mantissa addition, normalization and rounding of the mantissa, as shown in Fig. 2 (as detailed in [8]). Two operands are first unpacked from the IEEE 754 format, in which a 23-bit mantissa is added to the hidden bit. Addition of floating-point numbers involves comparing two 8-bit exponents and adding two 24-bit mantissas; the exponents are first evaluated to find the larger number.

The mantissas are then swapped related to the exponent comparison and aligned to have an equal exponent before they are added in the mantissa adder. Following the addition, normalization shifts are required to restore the results into the IEEE standard form. The normalization is completed by left shifting with the number of leading zeros. Therefore, leading zero detection is a key step of normalization. Rounding the normalized result is the last step before storing it; special cases (such as overflow, underflow, zero) are also detected and represented by flags.

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

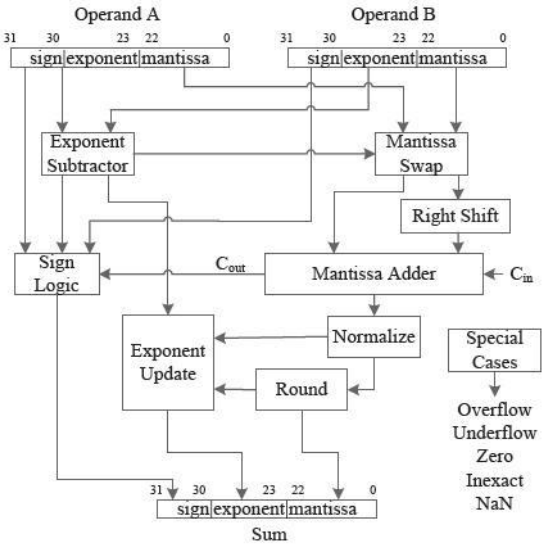
International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 5, May 2017

DESIGN OF INEXACT FLOATING-POINT ADDERS

A simple design of an inexact floating-point adder involves replacing the mantissa adder and exponent adder with approximate adders**.**



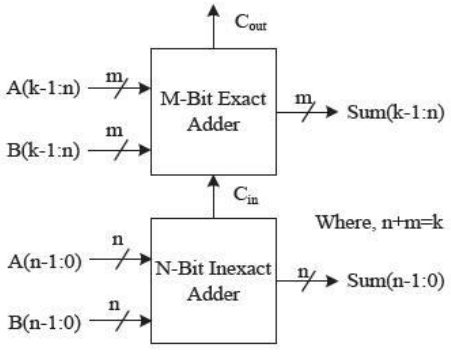
However, as the floating-point adder architecture is substantially different from a fixed-point adder, related logic (addition of an inclusion the normalizer and the rounder) should also be changed.

A. Exponent Subtractor

The exponent is dominant in the floating-point format, because it determines the dynamic range. Due to its importance in the number format, an approximate design of the exponent subtractor must be carefully considered. The results can be significantly affected by applying the approximate design to only the least significant bit (LSB) of the exponent subtractor. (as demonstrated in the following section).

B. Mantissa Adder

The mantissa is less significant than the exponent; therefore, it is more appropriate to consider using an inexact mantissa adder in an inexact floating-point adder. The mantissa adder is also larger than the exponent subtractor, so it offers a larger design space. The mantissa adder is modified to an inexact architecture in the proposed approach. Different inexact fixed-point adders have been proposed and can be used in the types of an mantissa adder, such as lower-part-OR adders (LOA), approximate mirror adders and approximate XOR/XNOR-based adders. In this paper, LOA is applied for the inexact mantissa adder due to its low complexity; a k-bit LOA consists of two parts, that has to be m-bit of an exact adder and an n-bit type of an inexact adder. The m-bit adder is used for the m MSB most level of significant bits of the sum, while the n-bit adder consists of OR gates to compute the addition of the LSB of an least significant n bits. So, the lower n-bit adder is an array of n 2-input OR gates. In the original LOA architecture, an additional AND gate is used for generating the most significant carry bit of the n-bit adder; in this research, all carry bits in the n-bit inexact adder are ignored.



DOI 10.17148/IJIREEICE.2017.5553

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 5, May 2017

C. Normalization and Leading Zero Counting

Normalization is necessary to ensure that the addition results are in the exact range; the addition or subtraction may be too small and a multi-bit left shift may be required. A reduction of the exponent is also mandatory. The normalization is performed based on a leading zeros/ones counter that determines the wanted number of left shifts. As the mantissa adder is not exact for the least significant n bits, the detection of the leading zeros can also be simplified in an approximate design; therefore, an approximate leading zero detection logic can be used.

D. Rounding

A rounding mode is required to accommodate the inexact number that an IEEE 754 format can represent. A proper rounding maintains three extra bits (i.e. guard bit, round bit and sticky bit); however, the adder may require another normalization and exponent adjustment after the rounding step. It is clear that the hardware overhead for rounding is significant. However, it does not affect the results of the inexact addition as the lower significant n bits are already inexact.

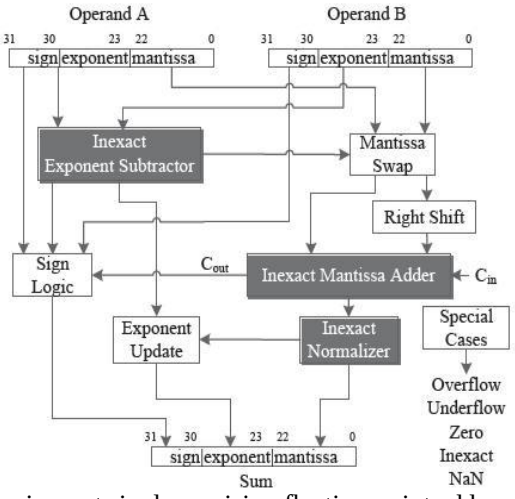


Figure: The inexact single precision floating-point adder architecture.

II. PROPOSED STRUCTURE

PARALLEL PREFIX ADDERS

The Parallel Prefix Adder is similar to a Carry Look Ahead Adder. The construction of the carriers the prefix adders can be designed in much different ways of approaches based on the different needs. We use tree structure form to increase the speed of mathematical operation. Parallel prefix adders are faster adders and these are very much speeded types of adders and used forhigh performance arithmetic architecture in industries. in 3 steps we used to did a parallel prefix addition is done.

1. Pre-processing stage
2. A Carry generation stage of network
3. Post processing stage

Pre-Processing Stage

In this stage we work out, the execute and show out a signals are used to generate carry input of each adder. the inputs are A and B. These signals are given by the equation 1and 2 are given below.

$$Pi=Ai + Bi \quad .....(1)$$

$$GI=Ai \cdot Bi \quad .....(2)$$

A Carry Execution Network

In this level we work out the carries corresponding to each bit. An executed Generation is done in parallel. After the estimation of carries in parallel they are sub divisional into smaller pieces. A carry operator consists a two AND gates , one OR gate. The below equations are 3&4. It uses to propagate and generate as an intermediate signals which are given below.

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 5, May 2017

P(i:k)=P(i:j)**.**P(j1:k)…….....(3)

G(i:k)=G(i:j)+(G(j1:k)**.**P(i:j))……..….(4)

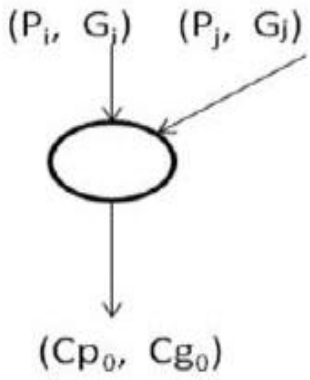


Figure: Carry operator.

This above processes are involved in this figure are given as.

Post Processing Stage
This is the final stage to execute the addition of input bits. it is similar for all adders and sum bit equation given
Si=Pi+Ci……...…..(5)

Ci+1=(Pi**.**C0)+Gi……(6)

III. PARALLEL PREFIX ADDER

NOTATION BACKGROUND
Prefix: The outcome of the operation depends on the initial level of input data's.

Parallel: It involves the execution of an operation in parallel. This is did by segmentation results into smaller pieces that are executed in parallel.

Operation:
Any arbitrary primitive operator "○"that is related to its parallelizable it is very speed because the flow is proficient in a parallel fashion.

3 Types of Parallel Prefix Adders
- 1.Kogge- Stone Adder
- 2.Brent-Kung Adder
- 3.Ladner-Fischer Adder

KOGGE - STONE ADDER
Kogge-Stone adder is a parallel prefix adder is shows a carry look ahead adder. The Kogge-Stone adder was constructed by peter M. Kogge and Harold S. Stone which that they released in 1973. Kogge-Stone prefix adder is a very speed and fast adder design. KS adder has best results in VLSI operations. Kogge-Stone adder has a very big area with minimum fan-out. The Kogge-Stone adder is popularly well known method as a parallel prefix adder that operates fast logical addition. Kogge-Stone adder applications is used for wide adders since of it executions that the less delay among other architectures. In fig, each vertical stage produce reproduces and execution bits. Generate bits are produced in the last level and these bits are XORed with the initial generate after the input to gives the sum bits. The Proposed Kogge- Stone adder figures shown below.

The Kogge-Stone adder Consists:
- Low level of depth
- High nodes of count (implies more area).
- Minimal fan-by all of 1 at each node (implies faster than performance).

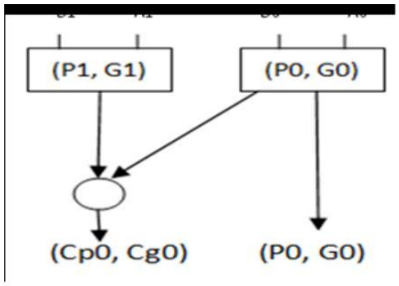
IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 5, May 2017



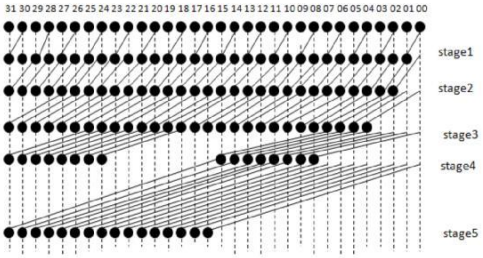


Figure: Kogge-Stone Adder

IV. SIMULATION RESULTS AND ITS DESCRIPTION

Comparisons for Inexact Floating Point and Modified Parallel Prefix Adders Outputs

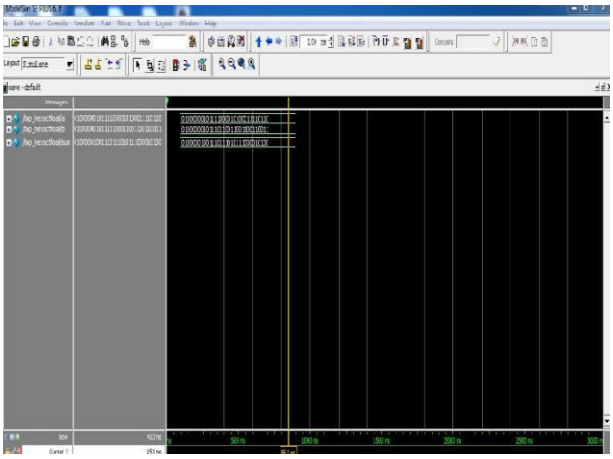


Figure 1: Simulated Output of inexact floating point adders

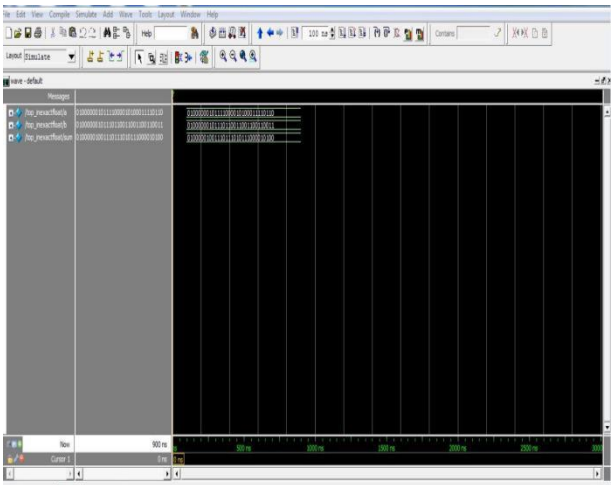


Figure 2: Simulated Output of modified adder (Parallel prefix Adder types in KOGGE -STONE ADDER.)

IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in
Electrical, Electronics, Instrumentation and Control Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 5, May 2017

In modified result, in the mantissa adder of inexact floating point adder. We have replaced the normal adder in the addition of significant bits by parallel prefix adder (kogge-stone-adder) and we obtained the results of delay time has been decreased much better than inexact floating point normal adder types.

V. CONCLUSION

Various adders were designed using Verilog language in Xilinx ISE Navigator 10.1 and all the operations are performed using Modelsim 6.5e simulator. The performance of proposed adders are worked and compared. In this proposed design, the implementation code for 32-bit Kogge-Stone is developed and appropriate values of delay and area were observed. Figure 1 & 2 shows the trade-off between different topologies. The simulated outputs of modified inexact floating point adder in replacements of parallel prefix proposed adders are shown in comparison figures 1 & 2.

REFERENCES

[1] K. Palem, and A. Lingamneni, "Ten years of building broken chips: the physics and engineering of inexact computing." ACM Trans. Embedded Computing Systems, vol. 12, no, 2, article 87, 2013.
[2] Lingamneni, K. Muntimadugu, C. Enz, R. Karp, K. Palem and C. Piguet, "Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling." Proc. ACM Int. Conf. Computing Frontiers, pp. 3-12, 2012.
[3] 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.
[4] 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.
[5] 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
[6] 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
[7] C. Liu, J. Han, F. Lombardi, An Analytical Framework for Evaluating the Error Characteristics of Approximate Adders' IEEE Trans. Computers, preprint, 14 Apr. 2014, doi: 10.1109/TC.2014.2317180.
[8] C. Liu, J. Han, and F. Lombardi, ‚A low-power, high performance approximate multiplier with configurable partial error recovery,' in Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), 4 pp., DOI: 10.7873/ DATE.2014.108, 2014.
[9] J. Y. Tong, D. Nagle, and R. Rutenbar, Reducing power by optimizing the necessary precision/range of floating-point arithmetic', IEEE Trans. Very Large Scale Integr. Syst., vol. 8, pp. 273-286, 2000.
[10] Gupta, S. Mandavalli, V. Mooney, K. Ling, A. Basu, H. Johan, and B. Tandianus, Low power probabilistic floatingpoint multiplier design', Proc. IEEE Comput. Soc. Annu. Symp. VLSI, pp. 182-187, 2011.
[11] F. Fang, T. Chen, and R. Rutenbar, ‚Floating-point bit-width optimization for low-power signal processing applications', Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 3, pp. 3208-3211, 2002.
[12] J. Eilert, A. Ehliar, and D. Liu, ‚Using low precision floating mpoint numbers to reduce memory cost for MP3 decoding', Proc. 6 th IEEE Workshop on Multimedia Signal Processing, pp. 119-122, 2004.
[13] W. Liu, L, Chen, C. Wang, M. O'Neill and F. Lombardi, ‚Inexact Floating-Point Adder for Dynamic Image Processing', Proc. 14th IEEE Conf. Nanotechnol (IEEE-NANO), pp. 239-243, 2014.
[14] IEEE Standard for Floating-Point Arithmetic," IEEE Std 754-2008, Aug. 29 2008, doi: 10.1109/IEEESTD.2008.4610935.
[15] B. Parhami, Computer arithmetic: algorithms and hardware designs. Oxford University Press, Inc., 2009
[16] J. Liang, J. Han, and F. Lombardi, ‚New metrics for the reliability of approximate and probabilistic adders'. IEEE Trans. Comput., vol. 62, pp. 1760-1771, 2013.
[17] Industrial Light & Magic, http://www.openexr.com/, 2014.
[18] R. Mantiuk, K. Kim, A. Rempel, and W. Heidrich, ‚HDR-VDP-2: a calibrated visual metric for visibility and quality predictions in all luminance conditions'. ACM Trans. Graphics, vol. 30, pp. 40, 2011.