IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834, p- ISSN: 2278-8735.
PP 91-98
www.iosrjournals.org

Floating- Point Three-Term Adder

Sruthy K Pillai[1], Anoop T R[2]
[1]*(Electronics and Communication Department, Sree Narayana Gurukulam College of Engineering, India)*
[2]*((Electronics and Communication Department, Sree Narayana Gurukulam College of Engineering, India)*

Abstract: *The fused floating-point three-term adder performs two additions in a single unit to achieve better performance and better accuracy compared to a network of traditional floating-point two-term adders, which is referred to as a discrete design. Here are several critical design issues for the fused floating-point three-term adder: 1) Complex exponent processing and significand alignment, 2) Complementation after the significand addition, 3) Large precision significand adder, 4) Massive cancellation management, and 5) Complex round processing. In order to further improve the performance of the three-term adder, several optimization techniques are applied including a new exponent compare and significand alignment, dual-reduction, early normalization, three-input leading zero anticipation, compound addition/rounding and pipelining. The proposed design is implemented for single precision. This paper presents improved architectures for a multi-operand floating-point three-term adder. The multi-operand floating point three term adder is modified by replacing the adder block with CB(Common Boolean logic full adder) adder in order to reduce the area and power consumption and to reduce the latency when comparing with a discrete floating-point three-term adder. The proposed design performs the significand addition and rounding simultaneously so that the latency is significantly reduced. Finally, the shifters for the alignment and normalization are overlapped with the exponent difference computation and LZD logic, respectively so that only the last level of the shifter is in the critical path.*
Keywords *- Floating-point arithmetic, CB adder, LZD, three-term adder*

I. Introduction

Addition is the most frequently used operation in many algorithms and applications. In case of the additions in series, however, a network of the two term adders loses accuracy due to the multiple rounding—one after each addition. Mainly floating point three term adders does not require a special data path or register file. Several issues for the design of the floating point three term adder are **: -** 1) Complex exponent processing and significand alignment, 2) Complementation after the significand addition, 3) Large precision significand addition, 4) Massive cancellation management and 5) Complex round processing. The proposed fused floating-point three-term adder takes three normalized operands and executes two additions (or subtractions) as **S=A ±B ± C.** Several algorithms and optimization techniques are applied not only to resolve the design issues but also to improve the performance:-

1) A new exponent compare and significand alignment scheme is proposed. The three exponent differences are computed in parallel and the differences are used for the significand alignment. By shifting the significands with the partial difference results, the exponent difference computation and the significand alignment can be overlapped. The control logic determines the largest exponent and three aligned significands. This approach reduces the latency by performing the exponent processing and significand alignment simultaneously.
2) Dual-reduction is used to handle both cases that the result of the significand addition is positive and negative. Two reduction trees generate both the positive and negative significand pairs and the positive significand pair is selected based on the significand comparison. The selected significand pair produces a positive sum so that the complementation after the significand addition can be skipped, which reduces the latency.
3) Early normalization is applied to reduce the significand addition size. By performing the normalization prior to the significand addition, the adder size is reduced by half and the rest of bits are covered by the rounding, which significantly reduces the latency.
4) Since the normalization is performed prior to the significand addition, the Leading Zero Anticipation (LZA) and normalization shift are on the critical path. In order to reduce the latency, a three-input LZA is proposed, which hides the delay of the 3:2 reduction trees.
5) Compound addition is used for fast rounding. The compound addition produces the rounded and unrounded sums together, and then the correct one is selected by the round logic so that the delay of the

rounding is hidden.

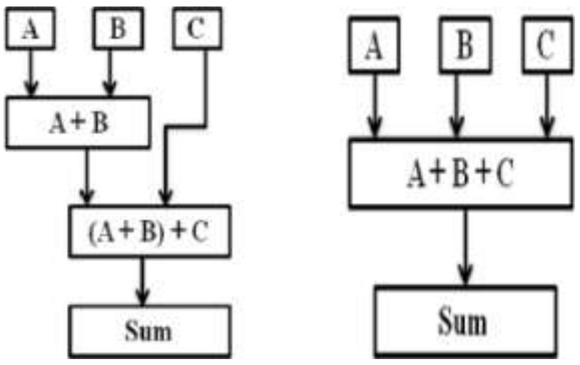


Fig. 1 Discrete vs. fused floating-point three-term adders [1]

In order to increase the throughput, pipelining can be applied. Based on the data flow analysis, the proposed fused floating point three-term adder is split into three stages. Since the latencies of three stages are fairly well balanced, the throughput can be improved. The multi-operand floating point three term adder is made to improve with its throughput and power consumption part by replacing a full adder used in addition block with a parallel prefix adder full adder with Common Boolean logic. This can imply the adder with better performance. This adder provides with less congestion complexity and area consumption. For further performance improvement, pipelining can be applied

II. Literature Survey

Although the floating-point adders are well-optimized for two-term addition, the discrete design for the three-term adder takes twice the area, latency and power consumption of a single floating-point adder. In order to reduce the overhead, a fused floating-point three-term adders have been proposed, which performs two additions in a single unit [2], [3]. Fig. 1 shows a high level comparison of the discrete design and a representative fused design. The fused floating-point three-term adder shares common logic to reduce the area, power consumption and latency.

In [1] the design of a component to perform parallel addition of multiple floating-point (FP) operands is explored in this work. In particular, a 3-input FP adder is discussed in more detail, but the main concepts and ideas presented in this work are valid for FP adders with more inputs. The proposed design is more accurate than conventional FP addition using a network of 2-operand FP adders and it may have competitive area and delay depending on the number of input operands. Implementation results of a 3-operand FP adder are presented to compare its performance to a network of 2-input FP adders. The experimental results int he paper show that the 3-input FP adder design can be synthesized to reach shorter or similar delays than the network of 2-input FP adders, with comparable or better area, and more accuracy. This paper depicts the addition algorithm using a carry select adder. Even though for multi-operand addition, mainly focus in three term addition and depicting the results are better than the two term adder.

[2] Explains an improved 3-operand floating-point (FP) adder has been presented. Firstly, the internal width of the adder has been given which is compatible with IEEE-Std754. Secondly, a realignment method processing sticky bits is used to make the architecture has the same accuracy with a FP adder which has a infinite internal width. Thirdly, a low cost method to detect catastrophic cancellation has been employed. Several sophisticated techniques, such as compound adder and Leading zero anticipation (LZA), are utilized to optimize the architecture. The implementation results show that the proposed architecture has a competitive area and delay.

[3] This paper presents improved architectures for a fused floating-point add–subtract unit. To improve the performance of the fused floating-point add–subtract unit, a dual-path algorithm and pipelining are employed. The proposed designs are implemented for both single and double precision. The fused floating-point add–subtract unit saves 40% of the area and power consumption compared to a discrete floating-point add–subtract unit. The proposed dual-path design reduces the latency by 30% compared to the discrete design with area and

power consumption between that of the discrete and fused designs. Based on a data flow analysis, the proposed fused dual-path floating-point add–subtract unit can be split into two pipeline stages. Since the latencies of two pipeline stages are fairly well balanced, the throughput is increased by 80% compared to the non- pipelined dual-path design.

III. Methodology

A high level algorithm to perform multi-operand addition is as:

1. Unpacking each of the FP input values fi to obtain sign bit(1bit)MSB, exponent(8 bit), and significand(23 bit+ 1 bit hidden)
2. determine the maximum exponent emax = max(e1,e2,e3) and compute the exponent differences diffi= emax - ei
3. align the values of 1.mi based on diffi and determine the values of stki (sticky bits). The number of output bits in the alignment unit depends on the internal precision of the adder (p bits). Thus, aligned significand will be the shifted value of the significand with difference
4. determine the value stk to be used during addition of aligned significands based on stki and fi
5. perform addition of aligned mantissas with stk
6. detect catastrophic cancellation of the largest fi's adjust the addition result and emax value accordingly
7. perform normalization and rounding
8. detect for special cases and fix the Floating point

A) Alignment of significands

The alignment of significand involves also the computation of the sticky bit used for addition. The alignment of operands requires the detection of the maximum exponent and the computation of the relative difference between exponents. Once the differences between exponents are computed, the alignment is done using variable shifters. An important parameter for this block is the bit width of the variable shifter outputs (k). They depend on the width of the internal adder for aligned signific and Most of the time, the bits of one aligned significand may have a weight that is completely different than the weight of bits in another aligned significand. It is practically impossible (for usual FP formats) to have enough internal precision to keep all the aligned significand bits of all operands. Therefore, a decision must be made regarding the precision of the internal adder for significands, and the degree of truncation that must happen as a consequence of this decision.

B) Catastrophic cancellation

Catastrophic cancellation of the largest aligned significands, resulting in a complete loss of significant bits if the exponent difference between the two largest FP numbers and the next FP number is greater than the precision of the internal adder. This problem cannot be solved with a reasonable adder size. The best alternative is to use detectors of catastrophic cancellations and a correction step that takes the next group of aligned significands

C) 3-operand FP adder

A simple way to implement a floating-point three-term adder is to concatenate two identical floating-point adders, which is referred to as a discrete floating-point three-term adder. One of the high-performance floating-point adders [7]–[8] can be used for the discrete design. The traditional floating-point three-term adder takes three operands and performs the two additions at once.

The procedure of the traditional fused floating-point tree-term adder is the exponent compare logic determines the largest of the three exponents and computes the differences between the largest exponent and each exponent, the effective operations are determined based on the three sign bits and the two op codes. The aligned significands are inverted if the corresponding operations are subtraction. Then, the significands are passed to the 3:2 reduction tree. Carry Save Adders (CSA) are used to perform the reduction, the significand addition is performed and the sum is complemented if it is negative. The LZA is performed in parallel with the significand addition and the significand sum is shifted by the amount of the LZA result, the sign logic determines the sign bit of the sum. The sign bit is passed to the round logic. The normalized significand is rounded and post-normalized. The exponent is adjusted with the carry-out of the significand addition and the shift amount from the LZA.

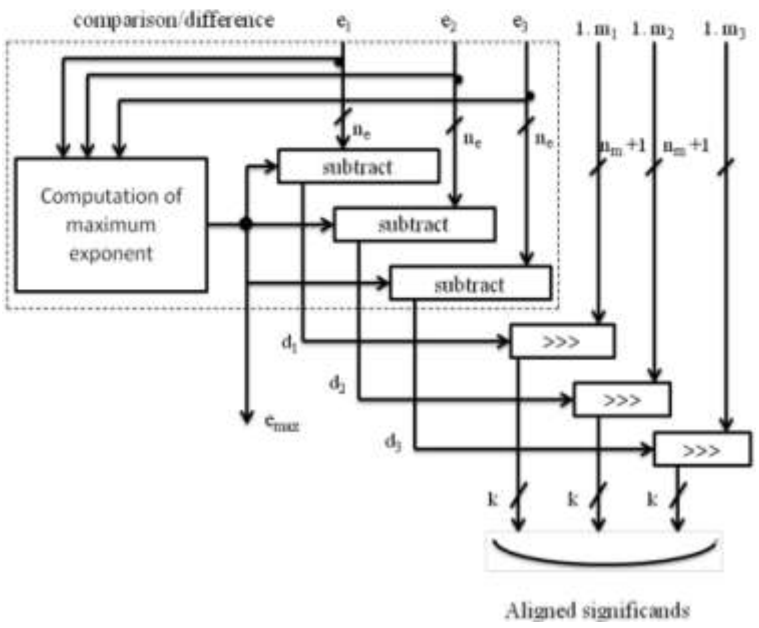


Fig 2. Alignment unit for a 3-input FP adder

D) Detection and processing of sticky bits

When performing multi-operand addition there is a possibility of having more than one aligned significand out of range (completely shifted out of range during alignment) or partially out of range. In any case, the sticky bit set for each significand after alignment phase must be used to define the sticky bit to be incorporated to the addition process. The decision about which sticky bit should be kept is decisive to generate an accurate result. The solution is trivial when the operands getting out of range (completely or partially) have the same sign, we just pick any one of them. The critical situation happens when the sticky bits area associated to operands with different signs.

signstk = sign($\sum^n$ i=1 stki ∗ fi)

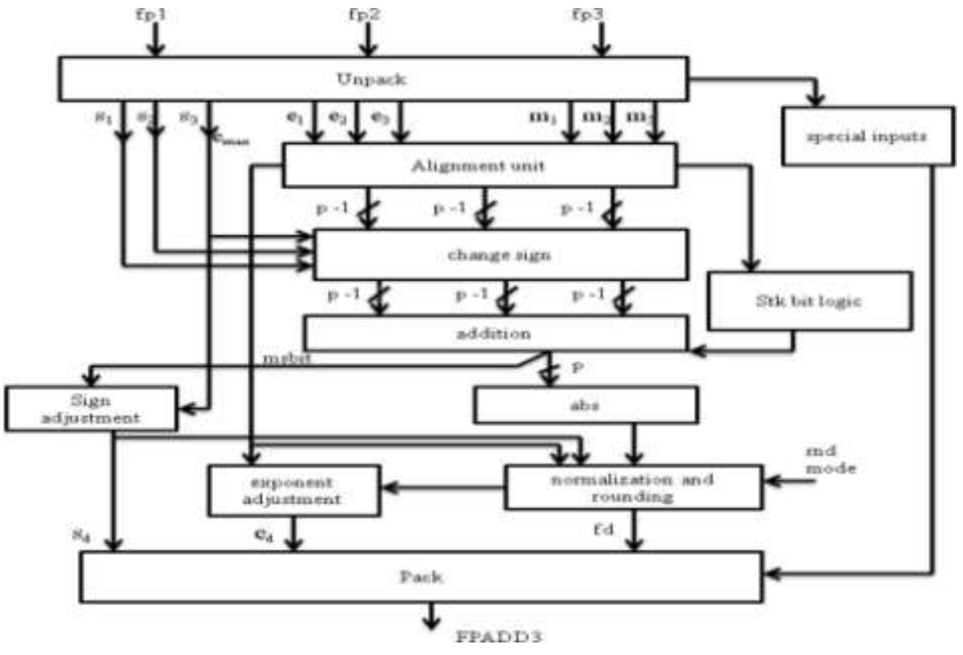


Fig 3 Block diagram for the 3-input FP Adder

The change of sign block adjusts the sign of significands based on the floating-point input *f*3 (anchor). The sign of *f*3, called *s*3 is passed to the sign adjustment logic to compute the final result sign. The sticky bit logic process the *stki* values coming from the alignment unit and let only one of them be used during addition. The blocks that follow the internal addition of aligned significands are similar to the blocks in a two-input FP adder.

In order to have exact rounding, the internal adder used in the FPADD3 must have $2f + 5$ bits of precision. The catastrophic cancellation detection is done in the special inputs block. This block is also responsible for the detection of other input conditions, such as multiple zeros and infinities, the presence of NaNs, and combinations of special values. Comparators are used to identify the catastrophic cancellation cases during addition and control a bypass circuit in the pack block to forward the smallest FP input to the output

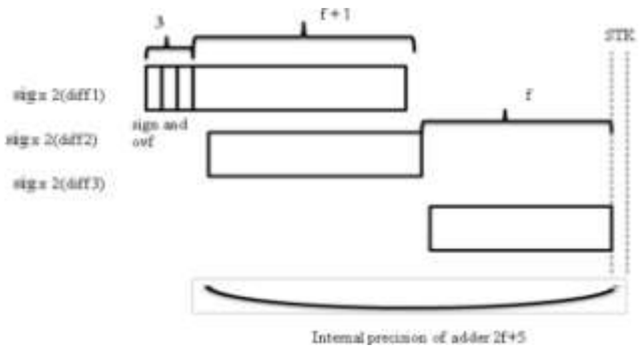
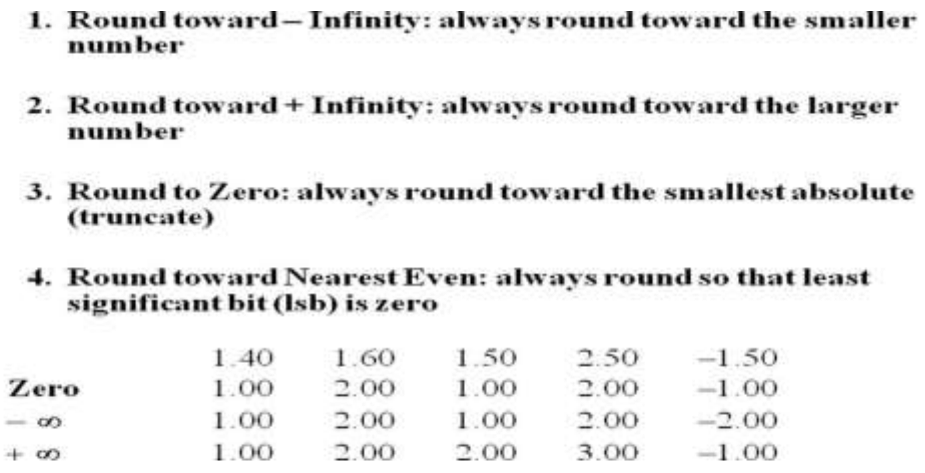


Fig 4 Minimum adder size when partial cancellation of largest operands occurs

The FPADD3 generates a perfectly rounded result. This means that its output is computed as it would be done with infinite precision, and then rounded to the finite FP format. The design enforces commutativity in the operation by treating all the inputs the same way. This characteristic cannot be obtained with a network of FPADD2s. One potential optimization to reduce area in the implementation of the FPADD3 would be to use only two alignment shifters instead of three since only 2-out-of-3 significands are aligned. However, in order to do this type of processing, the inputs should be sorted before they were applied to the alignment shifters, which would increase the critical path.

E) Rounding

Floating point multipliers perform the computation in two phases. In the first phase, an addition tree reduces the partial products to a carry-save encoded digit string that represents the exact product. In the second phase, a binary string representing the rounded product is computed from the carry-save encoded string.



1. **Round toward – Infinity: always round toward the smaller number**
2. **Round toward + Infinity: always round toward the larger number**
3. **Round to Zero: always round toward the smallest absolute (truncate)**
4. **Round toward Nearest Even: always round so that least significant bit (lsb) is zero**

	1.40	1.60	1.50	2.50	–1.50
Zero	1.00	2.00	1.00	2.00	–1.00
– ∞	1.00	2.00	1.00	2.00	–2.00
+ ∞	1.00	2.00	2.00	3.00	–1.00

Fig 5 Common rounding algorithms

Eventhough all the four algorithms are efficient, round to zero mode is used in the paper. Many other rounding algorithms are in experiment such as algorithm of Quach et al, which denoted as the QTF algorithm; 2) the algorithm of Yu and Zyne, which we denote as the YZ algorithm; and 3) a new algorithm that is based on injection-based rounding , which we denote as the ES algorithm.

F) Addition

In the design of Integrated Circuits, area occupancy plays a vital role because of increasing the necessity of portable systems. Carry Select Adder (CSLA) is one of the fastest adders used in many data-processing processors to perform fast arithmetic functions. In the paper, an area-efficient carry select adder by sharing the common Boolean logic term (CBL) is replaced for full adder. After logic simplification and sharing partial circuit, only one XOR gate and one inverter gate in each summation operation as well as one AND gate and one inverter gate in each carry-out operation are needed. Through the multiplexer, the correct output is selected according to the logic states of the carry in signal. This has reduced area and delay as compared with the regular full adder architecture. The result analysis shows that the CBL structure is better than the normal full adder.[8]

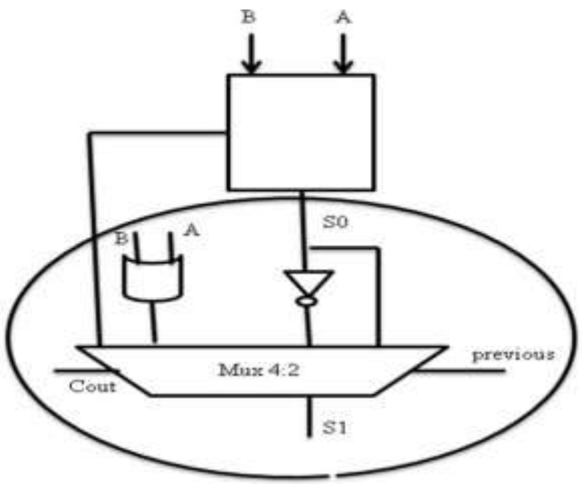


Fig 6. Single bit Full adder with CB

The summation and carry signal for full adder which has Cin=1, generate by INV and OR gate. Through the multiplexer, the correct output result is selected according to the logic state of carry-in signal. The internal structure of the group3 of proposed CSLA is shown in Fig. 8. One input to the mux goes from ripple carry adder block with Cin=0 and other input from the Common Boolean logic.

IV. Implementation Results And Analysis

The design of the 3-operand FP adder (FPADD3) was described in Verilog. The 3-input FP adder is superior in terms of speed, but it is not as small as the network of 2 DW fp add components for more relaxed time constraint. However, the 3-input FP adder is much more accurate than the network of small adders. It has an area of 29,323 and delay of 2.7ns when synthesized for very tight time constraint (without pipeline stages). Two of those adders in a network would add up to an area of almost 51,200 (considering an overall area reduction of 17% when two FP adders are combined) with a delay of 5.4ns

Table I shows the results for the four designs. Since the fused floating-point add–subtract unit shares much of the logic, it saves area over the traditional discrete floating-point add–subtract unit. Also, the fused floating-point add–subtract unit performs only one sign and operation decision at the end of the entire logic,while the traditional floating-point adder requires sign and operation decision logic for each addition, subtraction and exponent adjustment. As a result, the fused floating-point add– subtract unit shows less latency than the traditional discrete floating-point add–subtract unit. The dual -path fused floating-point add–subtract unit requires more area due to the three parallel additions, subtractions and LZAs for the close path. However, the dual -path design reduces the latency. The proposed pipelined fused floating-point add–subtract unit contains two stages. Each stage requires latches as many data and control signals are passed from the first stage to the next.

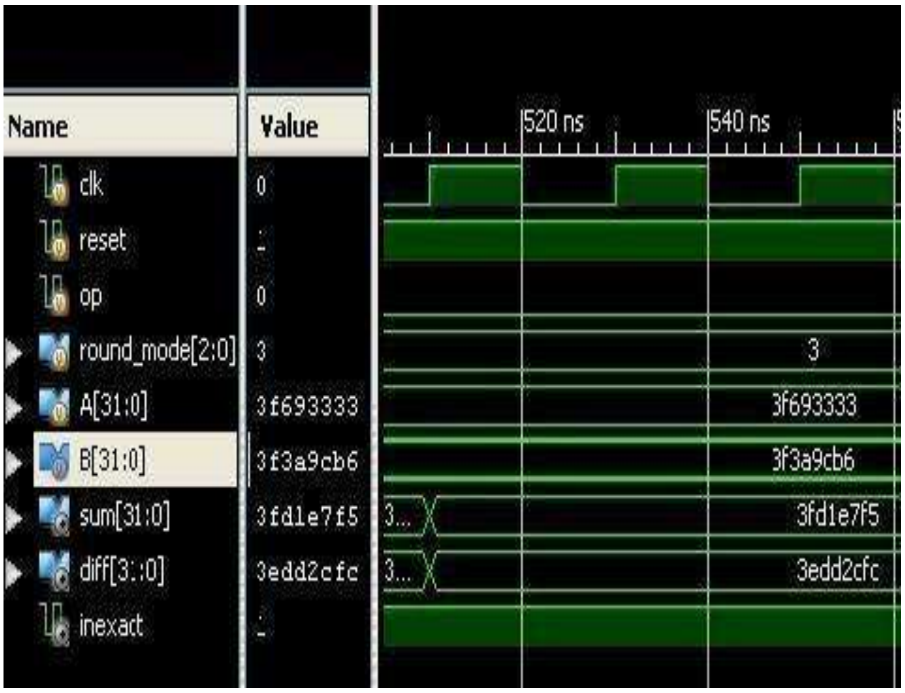


Fig 7 Simulation results of a 3-input adder

Table I
Design comparison

	Design	
parameter	Discrete	Improved
Area(gates)	40,147	20,507
Delay(ns)	12.12	11.04
Throughput(1/ns)	0.083	0.091

V. Conclusion

Improved architectures for the design and implementation of a fused floating-point add–subtract unit are presented. The floating-point add–subtract unit is useful for digital signal processing applications such as FFT and DCT butterfly operations. This paper presents improved architectures which apply the dual-path algorithm and pipelining to the fused floating-point add– subtract unit and compares the area, latency, throughput with the traditional parallel implementation.

The fused floating-point add–subtract unit saves area compared to the traditional discrete floating-point add– subtract unit by sharing the common logic. Also, the fused floating-point add–subtract unit reduces the latency due to its simplified control logic. The dualpath fused floating-point add–subtract unit reduces the latency compared to the discrete design by performing several add–subtract operations for each case in parallel. Additionally, a pipelined implementation to increase the throughput of the dual-path fused floating-point add–subtract unit is described. It uses two pipeline stages and the latencies are well balanced so that the throughput is increased compared to the non-pipelined dual-path design.

Acknowledgements

It is my immense pleasure to acknowledge with gratitude the help guidance and encouragement rendered to me by a lot of people to whom I owe a substantial measure for the successful completion of this desertion. I wish to express my sincere thanks to the project guide Mr. ANOOP T R, Assistant Professor ECE dept.

References

Transaction Papers:

[1] T. Lang and J. D. Bruguera, "Floating-point fused multiply-add with reduced latency," IEEE Trans. Comput., vol. 53, no. 8, pp. 988–1003, Aug. 2004.

[2] E. E. Swartzlander, Jr. and H. H. Saleh, "FFT implementation with fused floating-point operations," *IEEE Trans. Comput.*, vol. 61, no. 2,pp. 284–288, Feb. 2012.

[3] P. M. Seidel and G. Even, "Delay-optimized implementation of IEEE floating-point addition," *IEEE Trans. Comput.*, vol. 53, no. 2, pp.97– 113, Feb. 2004.

[4] J. D. Bruguera and T. Lang, "Leading—One prediction with concurrent position correction," *IEEE Trans. Comput.*, vol. 48, no. 10, pp. 1083–1097, Oct. 1999.

[5] R. Ji, Z. Ling, X. Zeng, B. Sui, L. Chen, J. Zhang, Y. Feng, and G. Luo, "Comments on 'Leading one prediction with concurrent position correction'," *IEEE Trans. Comput.*, vol. 58, no. 12, pp. 1726– 1727,Dec. 2009.

[6] G. Even and P.M. Seidel, "A comparison of three rounding algorithms for IEEE floating-point multiplication," *IEEE Trans. Comput.*, vol. 49,pp. 638–650, 2000.

Proceedings Papers:

[7] H. H. Saleh and E. E. Swartzlander, Jr., "A floating-point fused add–subtract unit," in *Proc. 51st IEEE Midwest Symp. Circuits Syst.*, 2008, pp. 519–522.

[8] K. T. Lee and K. J. Nowka, "1 GHz leading zero anticipator using independent sign-bit determination logic," in *Proc. Dig. Tech. Papers VLSI Circuits Symp.*, 2000, pp. 194–195

Theses:

[9] Damarla Paradhasaradhi, Prof. K. Anusudha ''An Area Efficient Enhanced SQRT Carry Select Adder'', D Paradhasaradhi et al Int. Journal of Engineering Research and Applications ISSN : 2248-9622, Vol. 3, Issue 6, Nov-Dec 2013, pp.876-880

[10] Mr.Kunapareddy S Nagendra , Mr.D.Suresh **,"** Improved Fused Floating Point Add-Subtract Unit**",** International Journal of Research in Computer and Communication Technology, Vol 2, Issue 9, September -2013