ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

Improved Architecture for Floating Point Addition

Meera K A

M. Tech Scholar, Dept. of Electronics, IIET, M G University, Kerala, India

ABSTRACT: This paper presents improved architectures for floating point addition. The improved architecture performs two additions in a single unit to achieve better performance and better accuracy. To improve the performance of the three term adder, several optimization techniques are applied. Including exponent compare and significand alignment, dual-reduction, early normalization, three input leading zero anticipation, compound addition and rounding. The proposed system reduce the area, delay and power consumption. In addition, the improved floating point three term adder rounding only once, which improves the accuracy.

KEYWORDS: Floating point arithmetic; optimization techniques for addition; three term addition; performance improvement

I. INTRODUCTION

Most of the general purpose and application specific processors using floating point addition. Floating point arithmetic differentiated from fixed point arithmetic by its constant relative precision over a wide dynamic range. Floating point operations require complex processes, including alignment, normalization and rounding, which increases the area, power consumption and latency. To reduce the overhead, improved floating point units have been proposed, which execute several operations in a single unit to reduce the area, power consumption and latency.

Traditional floating point two term adders are discussed in the previous work [1-3]. A network of the two term adders loses accuracy due to the multiple rounding one after each addition. Several issues for the design of the floating point three term adder are discussed in the previous work [4], [5]. They 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. Improved architecture, those issues are reduced by using several optimization techniques.

Optimization techniques are applied not only to resolve the design issues but also to improve the performance: A new exponent compare and significand alignment scheme is proposed. The three exponent differences are computed in parallel and the significand alignment can be done using them. Simultaneous operations reduce the latency. Dual reduction used to generate positive and negative significand pairs. This will avoid the complementation after the significand addition and also reduce the latency.

Significand addition size can be minimized using early normalization.It can possible to reduce the size of adder by half. By the use of Leading Zero Anticipation (LZA) hides the delay of 3:2 reduction trees. Compound addition used for fast rounding, with the help of round logic so that the delay is hidden.

II. RELATED WORK

A traditional fused floating point three term adder [4], [5]. The traditional floating point three term adder performs the two additions at once. The procedure of the traditional fused floating point tree term adder is:

1. The exponent compare logic determines the largest exponent from three operands. Computes the differences between the largest exponent and each exponent. The significands are shifted by the amount of the corresponding exponent differences.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

2. The aligned significands are inverted if the corresponding operations are subtraction. Then, the significands are passed to the 3:2 Carry Save Adders (CSA) are used to perform the reduction, which reduces the three significands to two.
3. 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 carry out of the significand addition is passed to the sign logic and the exponent adjustment logic.
4. 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 carry out of the significand addition and the shift amount from the LZA are used for exponent adjustment.

III. PROPOSED SYSTEM

Proposed system introducing optimizations techniques that can be applied to improve the performance of floating point addition. The modified design for an improved fused floating point three term adder shown in fig.1 . In this section, five optimizations for the improved fused floating point three term adder are described: 1) A new exponent compare and significand alignment scheme, 2) Dual reduction to avoid the need for complementation after the significand addition, 3) Early normalization, 4) Three input LZA, and 5) Compound addition and rounding.

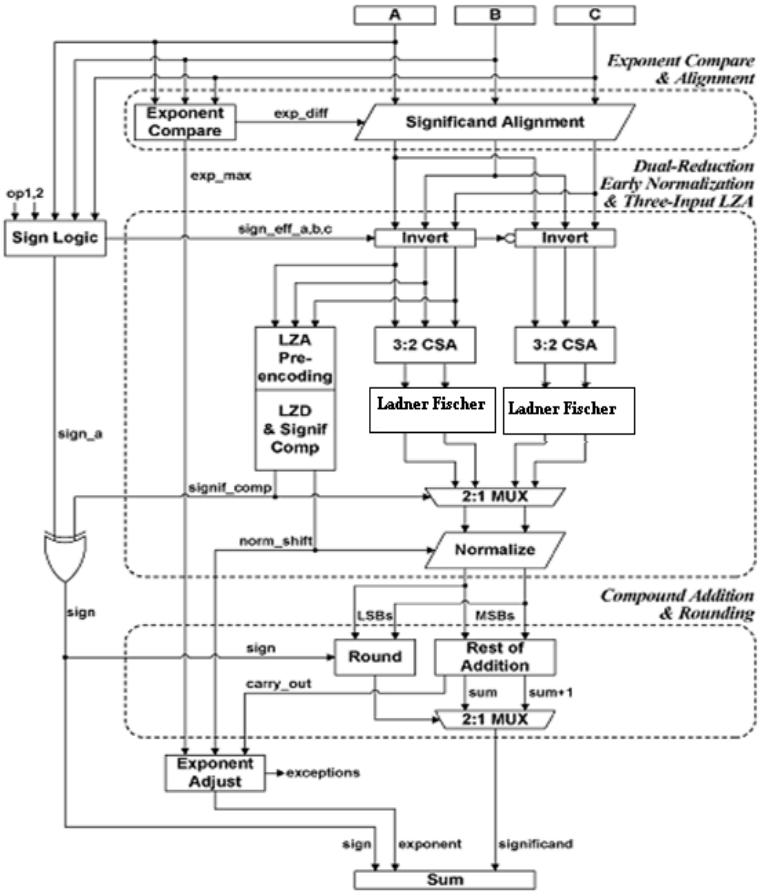

Fig. 1.Improved Architecture for Floating Point Addition

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

A. *Exponent Compare and Significand Alignment:*

In order to reduce the latency, a new exponent compare and significand alignment logic is proposed as shown in Fig.2 . Exponent differences are computed using six subtraction exp_a-exp_b , exp_b-exp_a , exp_b-exp_c , exp_c-exp_b , exp_c-exp_a , exp_a-exp_c .

Positive differences are selected from exponent differences pair, that skipping the complementation after subtraction. Exponent difference used for significand shifter so that the exponent compare and significand alignment are overlapped. By this method delay can reduced.

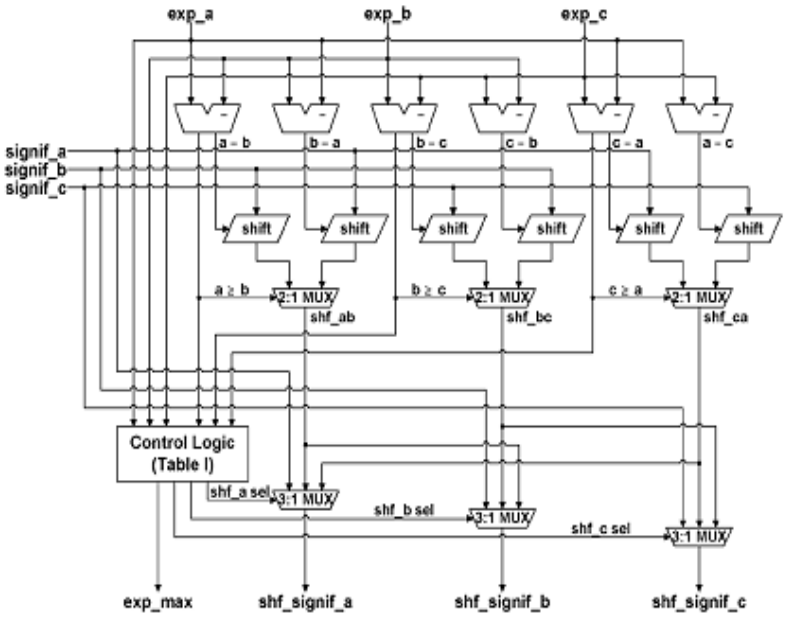

Fig. 2.Exponent Compare and Significand Alignment

The sticky logic is performed during the alignment to determine the guard, round and sticky bits. First and second bits under the LSBs are guard and round bits. Sticky bit can calculated by using OR tree. The largest exponent and aligned significands are determined by control logic shown result in Table 1. The aligned significands become 2f+ 6 bits wide including two overflow bits, and guard, round and sticky bits, where f is the number of the significand. The exponent compare and significand alignment logic reduce the latency due to both operations done simultaneously.

a ≥ b	b ≥ c	c ≥ a	exp_max	shf_a_sel	shf_b_sel	shf_c_sel
0	0	0	N/A	N/A	N/A	N/A
0	0	1	exp_c	shf_ca	shf_bc	signif_c
0	1	0	exp_b	shf_ab	signif_b	shf_bc
0	1	1	exp_b	shf_ab	signif_b	shf_bc
1	0	0	exp_a	signif_a	shf_ab	shf_ca
1	0	1	exp_c	shf_ca	shf_bc	signif_c
1	1	0	exp_a	signif_a	shf_ab	shf_ca
1	1	1	any	signif_a	signif_b	signif_c

Table.1 .Exponent Compare Control Logic

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

B. *Invert & Dual-Reduction:*

Sign logic determines the effective sign bit using three sign bits and two opcodes. Opcodes are the first and second sign bits respectively.

Sign_eff_a=sign_a
Sign_eff_b=sign_a ⊕ (sign_b ⊕ op1)
Sign_eff_c=sign_a ⊕ (sign_c ⊕ op2)

To avoid the increments after the inverters, 2 bits are extended to the LSB of the significands that are propagated to the significand addition to handle the cases that one or two significands are inverted by effectively adding 1 or 2, respectively. Table 2 shows the 2 bit extended LSBs based on the effective sign bits.

s_eff_a	s_eff_b	s_eff_b	$a_{-1}a_{-2}$	$b_{-1}b_{-2}$	$c_{-1}c_{-2}$	sum_0
0	0	0	00	00	00	0
0	0	1	10	00	10	1
0	1	0	00	10	10	1
0	1	1	10	11	11	2
1	0	0	10	10	00	1
1	0	1	11	10	11	2
1	1	0	11	11	10	2
1	1	1	00	00	00	0

Table.2 . Two Bit Extended LSBs for Complementation

Dual reduction avoids the complementation after significant addition. The two 3:2 CSAs produce the two reduced significand pairs. Between the two significand pairs, the positive pair is selected based on the sign of the significand sum. The part of addition in dual reduction carried out by the Ladner Ficher adder and which performs

$p_i=a_i \oplus b_i$
$g_i=a_i b_i$
$P_i^j=p_i p_{i-1} \ldots\ldots\ldots\ldots p_{i+(2^j-1)}$
$G_i^j=g_i+g_{i-1}p_i+\ldots\ldots\ldots+g_{i-(2^j-1)}p_i p_{i-1}\ldots\ldots\ldots p_{i-(2^j-2)}$
a_i and b_i are the i th bits of significands

C. *Early Normalization:*

Large significands require a large significand addition and normalization, which is the biggest bottleneck of the floating point three term adder. Early normalization is applied to reduce this problem. The normalization is performed prior to the significand addition so that the significand adder size is reduced to f+1. The rest of lower bits f+7 are passed to rounding. By normalizing the significand pair prior to the significand addition, the round position is fixed so that the significand addition and rounding can be performed in parallel, which significantly reduces the latency of the critical path.

D. *Three Input LZA and Significand Comparison*

The three-input LZA encodes the three inputs at once to skip the delay of the 3:2 CSA. The three-input LZA can be implemented by extending the two input LZA [6]. The three input LZA consists of two parts: 1) Pre encoding indicator vectors and 2) Leading Zero Detection (LZD) logic for generating the leading zero count.

The pre encoder generate the W vector as

$W=A+B+C$
$W_i=a_i+b_i+c_i$
$W_i \in \{0,1,2,3\}$
a_i, b_i, c_i are i th bit of significands

The W vector can be represented by four elements

$P_i^0=1$ if $w_i=0$

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

$P_i^1 = 1$ if $w_i = 1$
$P_i^2 = 1$ if $wi = 2$
$P_i^3 = 1$ if $w_i = 3$

The W Vector Is Pre-encoded Into Three Symbols

$$z_i = p_i^0(p^0_{i-1} + p^1_{i-1}) + p^3_i(p^2_{i-1} + p^3_{i-1})$$
$$t_i = (p_i^0 + p_i^2)(p_{i-1}^2 + p_{i-1}^3) + p_i^1 + p_i^3)(p_{i-1}^0 + p_{i-1})$$
$$g_i = p_i^1(p_{i-1}^2 + p_{i-1}^3) + p_i^2(p_{i-1}^0 + p_{i-1}^1)$$

F factor computation

$$f_i = t_{i+1}(g_i z_{i-1} + z_i g_{i-1}) + t_{i+1}(z_i z_{i-1} + g_i g_{i-1})$$

The F vector is passed to the LZD logic. The LZD produces the leading zero count which becomes the shift amount of the normalization. LZD logic producing the MSBs of the shift amount first is selected so that the LZD logic and the normalization shifter are overlapped [4]. Fig. 3 shows the 64 bit LZD tree, which used for the single precision. Lower levels of LZD logic overlap the delay for higher bits.

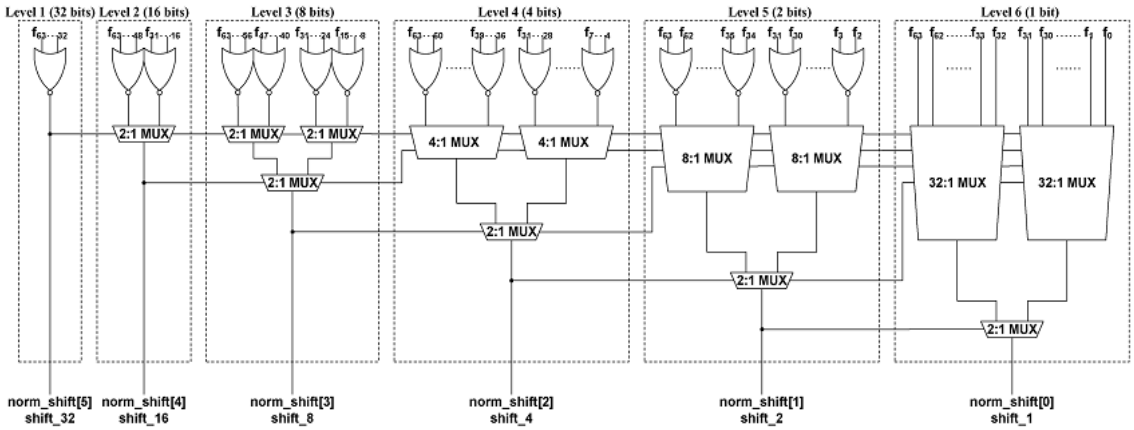

Fig. 3.64 Bit LZD Tree

To find out the sign of significand sum significand comparison is used. In order to reduce the overhead, LZA pre-encoded bits can be used for the comparison tree[7].

$$signif\_comp = z_n + t_n z_{n-1} + t_n t_{n-1} z_{n-2} \ldots\ldots + t_n t_{n-1} t_{n2} \ldots\ldots t_1 z_0 + t_n t_{n-1} t_{n-2} \ldots\ldots t_0$$

n is the MSB position of significands.

Significand comparison used for sign decision.

$sign = sign\_a \oplus signif\_comp$

E. *Compound Addition and Rounding*

The three term adder have two aligned significands, which causes carry propagation from the lower part. Then overflow can occur up to 2 bits. The significand addition result needs to be right shifted ,which changes the rounding position. Fig. 4 shows the overflows of the significand addition depends on the significand result selection and the rounding position.

Compound addition and rounding depends on [8] and compound addition determines the upper f bits including possible two overflow bits, and the rounding determines the rest of three LSBs and the round decision. The round logic requires computing the LSB, carry, guard, round and sticky bits (L, C, G, R and S) to determine if the significand sum is rounded up or not.In Fig.4 shows the rounding position depends on overflow bits.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

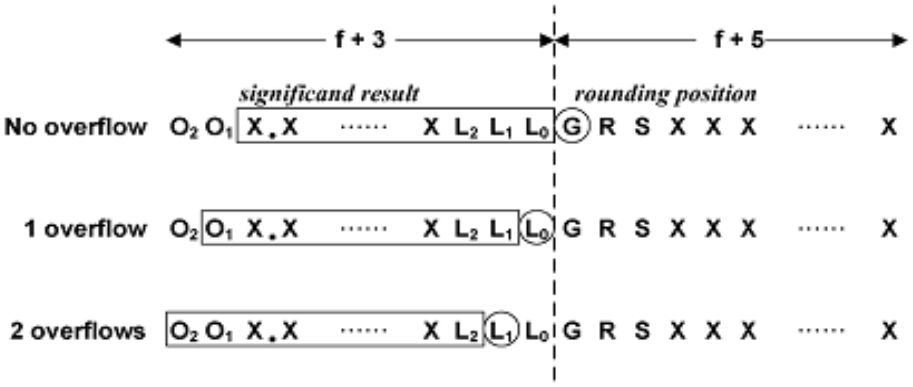

Fig. 4.Significand Selection and Rounding

Compound addition and rounding shown in Fig.5. determines the upper f bits including possible two overflow bits, and the rounding determines the rest of three LSBs and the round decision. Fig. 6 shows the round logic, which determines the round up bit with certain L, G, R and S bits.

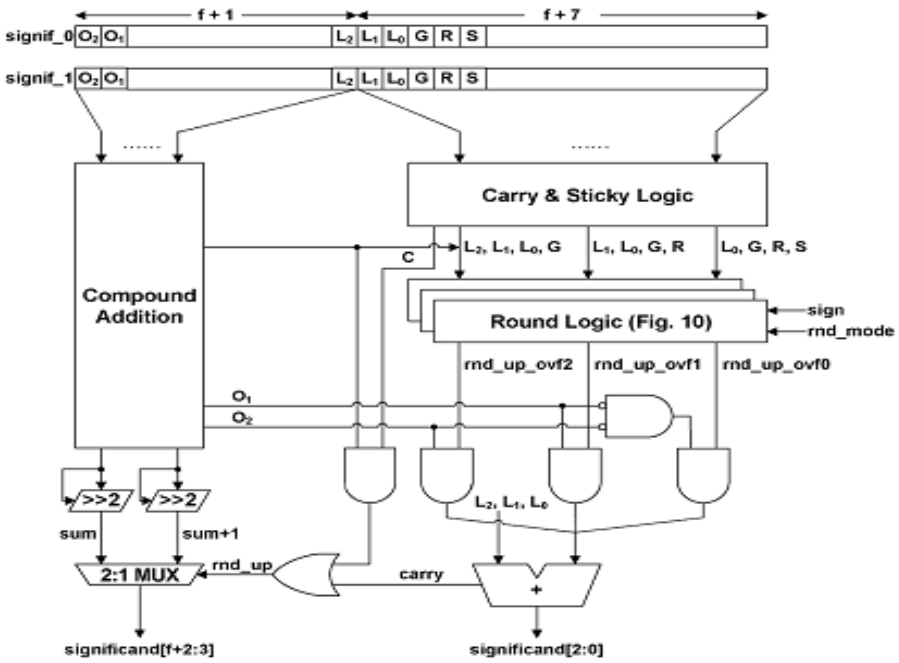

Fig. 5.Compound Addition and Rounding

The largest exponent determined by the exponent compare logic is adjusted by subtracting the shift amount from the LZA and adding the carry-out of the significand addition as shown in Fig. 7. The three significands generate overflow up to 2 bits, two carry-out bits are used for the adjustment.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

L2,L1 and L0 are find out LSBs of significand result, then three bit adder sum round up operands. The carry out of the addition is used for the final round up bit, which determines the upper significand result between sum and sum + 1. Round logic using five round modes including round to zero, round to positive, round to negetive, round near even, round near away.

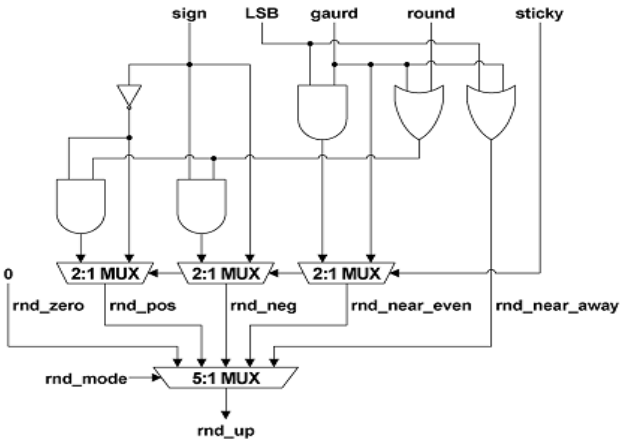

Fig. 6.Round Logic

The three excep-tion cases specified in IEEE-754 Standard [1] are detected as

$$overflow = \begin{cases} 1 & if\ exp \geq max\_exp \\ 0 & otherwise \end{cases}$$

$$underflow = \begin{cases} 1 & if\ exp \leq 0 \\ 0 & otherwise \end{cases}$$

$$inexact = round\_up\ overflow | underflow$$

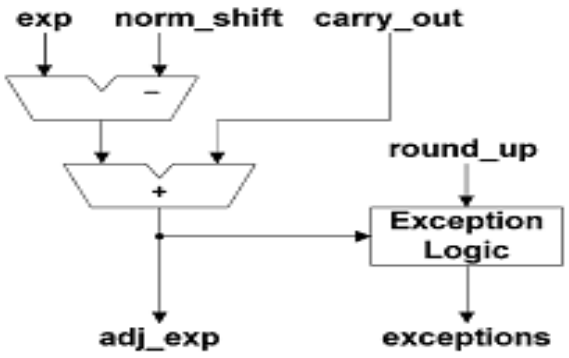

Fig. 7. Exponent Adjust Logic

IV. RESULT COMPARISON

The dual reduction stage including a part of addition module. We can use any type of prefix adder suitable for that. In this system using Ladner Ficher adder which is provide better performance than Kogge Stone adder. In Table 3 shows the comparison result of their logic level, fan out and wiring track.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 9, September 2015

Topology	Logic Level	Fan-out	Wiring Track
Kogge-Stone	LOW	LOW	HIGH
Ladner-Fischer	HIGH	LOW	LOW

Table 3.Comparison Result

The proposed design performs a smaller significand addition compared to the Kogge Stone adder designs. Delay reduction can possible without complementation by using the dual reduction. Also, 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.

Table 4 shows simulation difference between two adders. It shows requirement of LUTs, area, power. This result shows Ladner Ficher require less area and power as compared to Kogge Stone adder.

Adder	No of LUTs	Area(um^2)	Power(uW)
Ladner Ficher	87	335.87	21.17
Kogge Stone	204	502.16	27.35

Table 4.Simulation Difference

V. CONCLUSION AND FUTURE WORK

The improved architecture design and implementation for a fused floating point three term adder has been presented. There are several 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. To resolve those issues, several algorithms and optimization techniques are applied: 1) A new exponent compare and significand alignment 2) Dual-reduction, 3) Early normalization, 4) Three-input LZA and 5) Compound addition and rounding. The improved fused floating point three term adder reduces the area, power consumption. The algorithms and optimizations described in this paper can be also extended to fusedfloating point multi term adders with more than three operands. The improved fused floating point three term adder will contribute to the next generation of floating point arithmetic unit design.

REFERENCES

1. R. K. Montoye, E. Hokenek, and S. L. Runyon, "Design of the IBM RISC system/6000 floating-point execution unit," IBM J. Res. Develop., vol. 34, pp. 59–70, 1990.
2. S.F. Oberman, H. Al Twaijry, and M. J. Flynn, "The SNAP project: Design of floating point arithmetic units," in Proc. 14th IEEE Symp. Computer Arithmetic, pp. 156–165,1997.
3. P. M. Seidel and G. Even, "Delay-optimized implementation of IEEE floating-point addition, "IEEE Trans. Computers, vol. 53, no. 2, pp. 97–113, Feb. 2004.
4. A. Tenca, "Multi operand floating point addition," in Proc. 21st Symp. Computer Arithmetic, 2009, pp. 161–168, 2009.
5. Y. Tao, G. Deyuan, F. Xiaoya, and R. Xianglong, "Three-operand floating-point adder," in Pro. 12th IEEE Int. Conf. Comput. Inf. Technol., pp. 192–196,2012.
6. E. Hokenek, R. K. Montoye, and P. W. Cook, "Second-generation RISC floating point with multiply-add fused," IEEE J. Solid-State Circuits, vol. 25, pp. 1207–1213, 1990.
7. K. T. Lee and K. J. Nowka, "1 GHz leading zero anticipator using independent sign-bit determination logic," inProc. Dig. Tech. Papers VLSI Circuits Symp., pp. 194–195,2000.
8. G. Even and P. M. Seidel, "A comparison of three rounding algorithms for IEEEfloating-point multiplication,"IEEE Trans. Comput., vol. 49, pp. 638–650, 2000.
9. P. Kornerup, "Correcting the normalization shift of redundant binary representations," IEEE Trans. Computers, vol. 58, pp. 1435–1439, 2009.

BIOGRAPHY

Meera K A is a M tech Student in the VLSI & Embedded Department, College of IIET, M G University. She received B tech degree in 2013 from ICET, M G University in Kerala.