Parallelism Effects in Higher Term Dot Product Floating Point Computation

Prasanna Palsodkar, Prachi Palsodkar

Abstract*: *In complex signal processing applications, Floating Point (FP) arithmetic is a complex, but extremely accurate representation, which needs to be optimizing by architectural modification. This paper describes discrete to fused arithmetic implementation with two, three and four operand FP methodology. Parameters like Area, Power and Delay (APD) are considered for analysis. Exhaustive analysis is carried out here from basic FP component to complete structuring of Four Term Dot Product FP (FTDPFP). Analysis shows that FTDPFP computation improves speed by 89-91% compared to three term and two term computation. Area wise overheads increases in FTDPFP and it is optimized by using new exponent, dual reduction, early normalization, Leading zero participator (LZA), rounding and compounding techniques. Power consumption is optimized with same competency of Two and Three Term Dot Product Floating Point (TTDPFP).

Keywords: Floating point, two term, dot product, three operand, four term

I. INTRODUCTION

Complex arithmetic for 3D graphics, multimedia applications etc. needs to perform for today's Digital Signal Processing (DSP) System. To speed up DSP systems, FP number represented in IEEE 754 format is advantageous to use as it covers a large range of number for 32 bit, varies from maximum to minimum number [1]. Major maneuvers required in FP calculation are exponent adjustment, significant alignment, normalization and rounding unit [2]. Adding parallelism in operation by eliminating operation redundancy is one of the ways to optimize overheads in FP unit design. Various fused architectures proposed to improve calculation efficiency in [8] [6][21][7][22][34]. The fused architectures are namely Fused Add-Subtract (FAS), Fused Multiply-Add (FMA), Fused two term dot product (FTDP), Fused three term adder (FTTA) [37], Fused TTDPFP (FTTDPFP), and fused four operand adders (FFA) and FFTDPFP. In addition to fused architectures internal modifications also suggested in Data Path (DP) by [21][22], in adder structuring by [36][35][15][20][11] and in error correction by [24][27][38][18][26].

A different design of FP units is discussed here with their comparative analysis. FP arithmetic mainly deals with FP adder, subtractor, multipliers and division. A generic FP adder includes hardware block of exponent compare (EC), mantissa alignment (MA), mantissa addition, normalization and rounding discussed in section II. Floating point adders are broadly categorized on the basis of operands and fused architectures discussed in section III. Discrete FP units require more area and consume more power. Latency of a fused system is more compared to discrete unit. Multi-operand implementation increases speed at the cost of increase in hardware due to parallelism. Early normalization, DP optimization, modification of internal adders, LZA modification leads to improvement in FP unit performance. Section II discuss about discrete FP with respect to individual computation APD briefing. Improvement in APD effect shown in section III due to fusing effect. Section IV shows effect of three term optimization and improvement in APD due to modified adder structure. In Section V, FTDPFP unit in detail and its final its comparative analysis with all basic structures. Section VI ends with conclusion. User requirement depending upon application choose appropriate architecture.

II. DISCRETE FP UNIT

Table I: Area Analysis Of Basic Component On Vertex 5

All Basic Component	AREA(LUT)
Validity Checker	3
Comparator	33
Mantissa selection	64
Exponent Adjust	113
Mantissa Addition	73
Mantissa Subtraction	73
Mantissa multiplication	2 DSP48E
Exponent Addition	17
Normalization	110
Rounding	81
Output block	Nil
Multiplication adjustment	122

Revised Manuscript Received on April 15, 2020.

Prachi Palsodkar*, Department of Electronics Engg, Yeshwantrao Chavan College of Engineering, Nagpur, India. Email: prachi.palsodkar@gmail.com

Prasanna Palsodkar, Department of Electronics Engg, Yeshwantrao Chavan College of Engineering, Nagpur, India. Email: palsodkar.prasanna@gmail.com

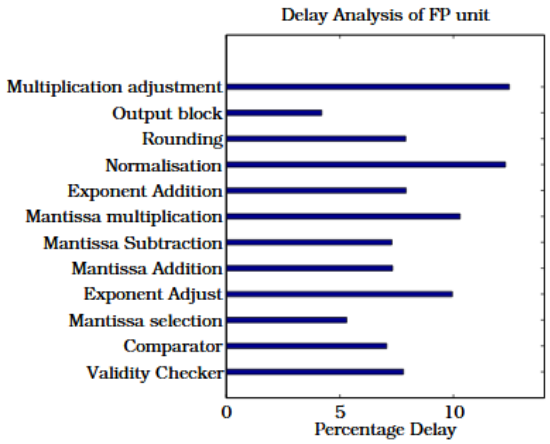

Fig. 1: Delay Analysis for Discrete FP Unit

The FP unit consists of sign, exponent and significant parts, along with normalization block. Fig.1 and TABLEI shows the Delay and Area analysis on vertex 5 for different FP unit. Area consumed is in terms of Look Up Table (LUT) of all FP components. It shows that exponent, mantissa multiplication-add unit and normalization block consumes more area and delay. In Multi-operand or fused designs Exponent adjust, comparator, validity checker, normalization blocks get shared thus reducing overall area and delay of the unit compared to discrete operation.

III. FUSED FP UNITS

DSP requires huge addition and multiplication commutatively. Discrete multiplication and addition consumes more area and power. Multiplication and addition per formed in fused way reduces critical path, adds parallelism in operation. Thus overall throughput of system increases.

A. Design Paradigm for fused FP unit

FMA FP unit performs multiplication followed by addition and replaces a processors FP adder, multipliers and reduces the latency. For embedded SP and graphics applications FMA is becoming integral part of processors like IBM, Intel and HP [15], [4], [3]. Similarly FAS gives addition and subtraction in a single unit [20]. FFPAS modules can be implemented using two different approaches. These are 1) Parallel implementation where two adders operate in parallel 2) Serial implementation where a single adder is used twice with the same operands. The structure of the FTDP FP module is derived from the discrete FP add and multiplier module [16]. Input1 and Input2 multiply and Input3 and Input4 multiply after adjusting result of multiplication. In fused structure the difference of exponent, significant shift and exponent adjustment functions needs be performed once with a single set of hardware and the results shared by both the multiply and the add operations.

TABLE II shows the area reduction for fused FP module over the discrete FP (vertex 5). It is observed that the fused FP modules are efficient as compared to the discrete FP implementation.FAS reduces cost by 17.93%. Fused FMA (FFMA) cost reduction over discrete unit is 18.52% and FTTDP calculation saves cost by 24.1 %. TABLE III shows 44.27%, 45.12 % and 61.86% reduction in delay for fused FFAS, FFMA and FTTDP unit over discrete FP implementation. Similarly TABLE IV shows for fused implementation of FPAS, FMA, and FP dot product unit is 50.28 %, 23.29% and 47.86% respectively. Section 3.2 will show effect of fused implementation on FFT module.

B. Comparative analysis for Discrete FP and Fused FP for Fast Fourier Transform(FFT)

Application of FP for Radix-2 Butterfly structure using fused as well as discrete FP unit is shown in TABLEV. It shows that the fused model of Radix 2 has 27.09% decrease in number of Look Up Table (LUTs), 17.10% reduction in delay, 11.00% reduction in power and 26.22% reduction in energy as compared to the discrete Radix-2.

Table II: Comparison Of Lut For Discrete And Fused Fpa& Fpm Models

FP unit	Discrete	Fused	Reduction %
One Add &One Sub FPA	1188	975	17.93%
One Mult &One Add FMA	907	739	18.52%
Two Mult & One add FMA	1220	926	24.10%

Table III: Delay (ns) Comparison of Discrete and Fused FPA & FPM Model

FP unit	Discrete	Fused	Reduction %
One Add & One Sub FPA	1291.76	719.94	44.27%
One Multi & One Add FMA	1261.44	692.27	45.12%
2 Mult & One Add FMA	1876.55	715.68	61.86%

Table IV: Power (mW) Comparison of discrete and Fused FPA and FPM

Design	Discrete	Fused	Reduction %
One Add& One Sub FPA	58.372	29.02	50.28%
One Mult& One Add FMA	52.094	39.96	23.29%
Two Mult & One Add FMA	274.802	39	47.86%

Table V: Comparison of Fused and Discrete FP Modules for Radix 2FFT

Design Radix2	LUT	DELAY (ns)	POWER (mW)	ENERGY (nJ)
Discrete	4799	3.089	1138.37	83.20
Fused-Module	3499	60.594	1013.15	61.39

IV. THREE OPERAND FP

This section describes improvement over discrete FP as well as FTDP computation using three terms. In this structure one more level of parallelism increased. In addition to this, internal modification in individual unit like adder helps in improvement of APD. Architecture is discussed in this section with redundant adder for three term addition [37], [30].

A. Fused Three Term Adder Substractor (FTTAS) using Redundant Adder Substractor

Fig. 2 shows detail architecture of FTTA using redundant Binary Signed Digit (BSD) adder. Shared units like exponent computation, Alignment scheme, BSD Adder and normalization is used. It reduces overall area and data path compared to discrete FP. Special BSD computation tends to increase in speed of computation.

Network of substractor and comparator calculates the maximum exponent and the shifting amount for mantissa [2], [37]. Maximum exponent calculation and alignment unit finds largest exponent and aligned mantissa. The aligned mantissas of first two operand are used as the input to BSD adder-subtractor. This operation is followed by sign logic block, it takes decision of addition or subtraction to be performed by BSD adder. Next to that, first BSD unit output propagates to the input of second BSD unit along with third operand for three operand processing. The output of BSD unit is followed by normalization and rounding.

BSD Operation

A= 1100, B=1010 Aplus=1100, Aminus=0000 B=1010

For Addition

Splus=11100, Sminus=1010,Sum=10110

For Substraction

Splus=1011, Sminus=1001, DIfference=0010

The LZA unit is used to normalize the result. Rounding operation is performed after normalization. The discarded bits generated during alignment of mantissa gives guard bit, round bit and sticky bits through estimation using these bits rounding is performed. From the discarded bits, the first MSB bit is considered as guard bit, next to guard bit is the round bit and ORing of all discarded bits is the sticky bits [38].

e.g Discarded bits= 1101,
Guard Bit =1, MSB
Round Bit= 1 Next to MSB
Sticky Bit= or(1101)=1

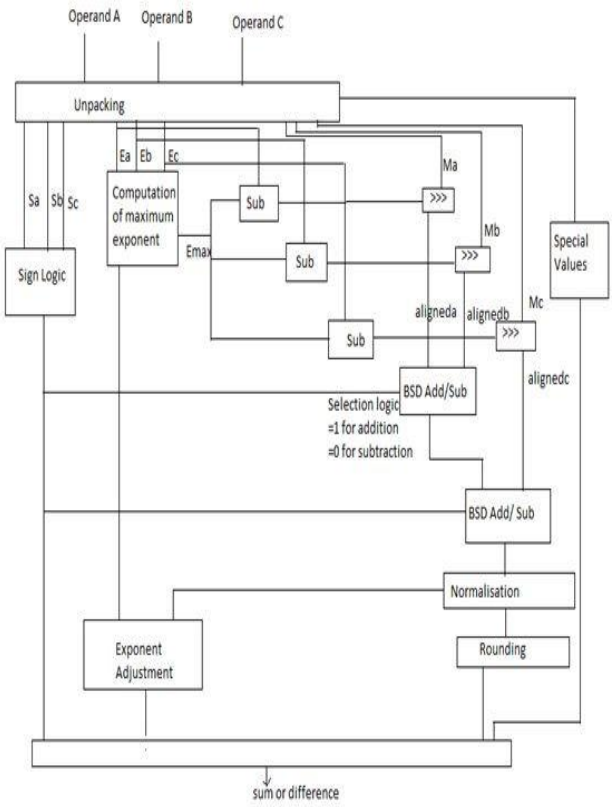

Fig. 2: Three-Operand Adder Substractor

B. Analysis of Three Term FFP(TTFFP) Computation

TTFFP design using BSD adder/substractor [33] achieves high speed with reduction in latency. TABLE VI shows the comparison of three different architectures of three term adders. It shows architectural modification also helps in computation and speed improvement. It shows that designed architecture using Fig. 2 reduces the critical path and shows the capability to perform fused operation (addition and subtraction). Three term calculation provides improvement by reducing delay by 25 % and area by 14.34 % over discrete FP structure. Fig. 3 shows that APD reduces due to parallelism. Fig. 2shows that the three term adder uses additional redundant arithmetic for adder, reduces APD of the Design compared to traditional FTTA, FAS and discrete FP adder

TABLE VI: Comparison of Three-Operand Addition

Constraint	Conventional [30]	Reference [31]	Fig.2
Exponent comparison and alignment	3(8 bit Sub) + 3 MUX + 3 Shifter Combinational logic	2 (8 bit Sub) 4MUX+ 2Shifter +30 block	3 (8 bit Sub) + 2 comparator + 2 shifter
Significant addition	CSA+ CPA	2 BSD adder	2 Fused BSD adder- subtractor
Critical path	Sub-MUX-Shifter-Comb -CSA-CPA	Sub-MUX-Sub-(+30) -Shifter- BSD adder	Sub-Shifter -BSD adder-Subtractor
Sign logic	Required	Not Required	Required

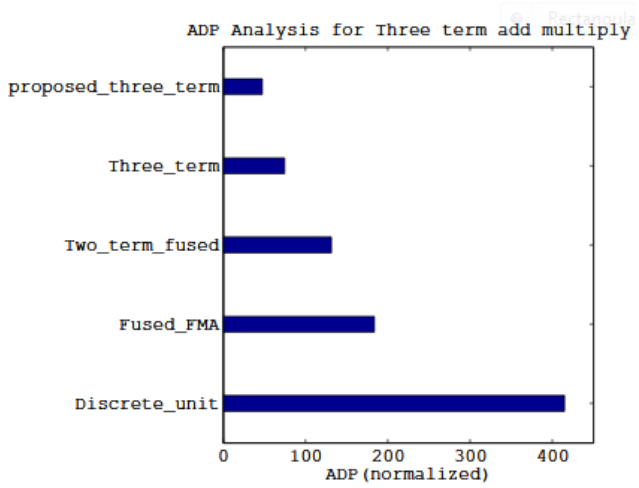

Fig. 3: APD Analysis for Three Term Add-Multiply

V. FTDPFP UNIT

The FTDPFP [16] unit described by Fig.4 applies several optimization techniques to improve the performance and results in reduction in the area and power consumption. It shows the FTDP designed in this section consist of the following stages:

1. Unpacking
2. Multiplier tree and exponent compare
3. Significant alignment and dual reduction
4. LZA and Normalization
5. Significant addition, rounding, exponent adjust and sign logic

A. Implementation Details

Four term computation consist of eight inputs, e.g A=0.4, B=5, C=0.3, D=0.8, E=-3, F=0.2, G=0.2, H=3. In binary, it's illustrated as

A=00111110110011001100110011001101
B=01000000101000000000000000000000
C=00111110100110011001100110011010
D=00111111010011001100110011001101,
E=11000000010000000000000000000000
F=00111110010011001100110011001101,
G=00111110010011001100110011001101
H=01000000010000000000000000000000

Next stage is to determine four terms

AB=A+B=01111101 + 10000001 =11111110,
CD=C+D=01111101 + 01111110=11111011
EF =E+F= 10000000 + 01111100=11111100,
GH=G+H=01111100 +10000000=11111100

By comparing resultant exponent AB, CD, EF, GH, it is required to calculate the largest exponent among all four exponents. Here Exp max=AB= 11111110 is the largest exponent.

An comparison of exponent and significant alignment logic is used to avoid the operand sorting and effectively handles the eight operands together. The exponent compare logic simplifies the exponent processing by using control logic to determine the largest exponent and the shift amounts for the significant alignment simultaneously. Then Dual-reduction is used to handle both the positive and negative cases. Two sets of carry save adder (CSA) reduction trees produce two significant pairs. The positive significant pair is selected based on the significant comparison so that the significant sum does not need to be complemented, it reduces the latency. Third stage follows early normalization to reduce the latency of the critical path by reducing data path. In four operand LZA encoding of four input is done at once which reduces critical path. Normalization consists of following steps. Let W vector is generated,

$$W = A + B + C + D$$

$$w_i = a_i + b_i + c_i + d_i \quad w_i \in \{0,1,2,3,4\}$$

Where a_i, b_i, c_i, d_i are the bits of four significant. W vector can be represented by one of the five element, p_i^0, p_i^1, p_i^2, p_i^3 and p_i^4 and its corresponding w_i vector is equals to 0,1,2,3 and 4 respectively.

$$p_i^0 = 1 \text{ if } w_i = 0$$

$$p_i^1 = 1 \text{ if } w_i = 1$$

$$p_i^2 = 1 \text{ if } w_i = 2$$

$$p_i^3 = 1 \text{ if } w_i = 3$$

$$p_i^4 = 1 \text{ if } w_i = 4$$

Using this five elements the W vector is pre encoded into three symbol g_i, t_i and z_i as

$$g_i = p_i^0 p_{i-1}^4 + p_i^1 (p_{i-1}^2 + p_{i-1}^3) + p_i^2 (p_{i-1}^0 + p_{i-1}^1) + p_i^4 p_{i-1}^4$$

$$t_i = p_i^0 (p_{i-1}^2 + p_{i-1}^3) + p_i^1 (p_{i-1}^0 + p_{i-1}^1 + p_{i-1}^4) + p_i^2 (p_{i-1}^2 + p_{i-1}^3) + p_i^3 (p_{i-1}^0 + p_{i-1}^1 + p_{i-1}^4) + p_i^4 (p_{i-1}^2 + p_{i-1}^3)$$

$$z_i = p_i^0 (p_{i-1}^0 + p_{i-1}^1) + p_i^2 p_{i-1}^4 + p_i^3 (p_{i-1}^2 + p_{i-1}^3) + p_i^4 (p_{i-1}^0 + p_{i-1}^1)$$

The F vector is generated with the three symbols as

$$f_i = t_{i+1}(g_i \bar{z}_{i-1} + z_i \bar{g}_{i-1}) + \bar{t}_{i+1}(z_i \bar{z}_{i-1} + g_i \bar{g}_{i-1})$$

f_i is the normalization term obtained followed by addition and rounding. Normalization and sticky logic operation prior to the significant addition helps in the reduction of the significant pair, which reduces the significant addition size by 50%. The sign bit detection before addition operation and then execution of addition along with round in g reduces latency significantly. In order to reduce the latency, a four-input LZA is used, which predicts the normalization shift amount faster by encoding the four significant at once. The traditional FFP FTD P unit is having four multiplier and three adders but due to the large significant swapping, addition, normalization and the rounding process in sequence which requires large area, Latency and power consumption.

Rounding Modes

Default Modes: Round to nearest even when tie

Directed modes:

Round toward plus infinity
Round toward minus infinity
Round toward 0 (truncate)
Round up (add RND to position L)

Round up (add RND to position L)
{ If G = 1 and R and T are not both zero, RND = G(R + T)
{ If G = 1 and R = T = 0 then RND = G(R + T)0L { tie case
Combining both cases,
RND = G(R T) + G(R T)0L = G(L R T)
L 1 1 0 1 1 1 G=1, R=1, T = 1 ---------- RND = 1
L 1 0 0 0 0 0 G=1, R=0, T=1 ----------RND = 1 (tie case)
L 0 x x x x x G=0 ----------RND = 0
After rounding exponent adjust produce exponent and exception flags.
Overflow= 1 if exp>=exp_max
0 otherwise
underflow=1 if exp<=0
0 otherwise
Inexact=rnd_up|overflow|underflow
where rnd_up is the rounding decision of significant addition.

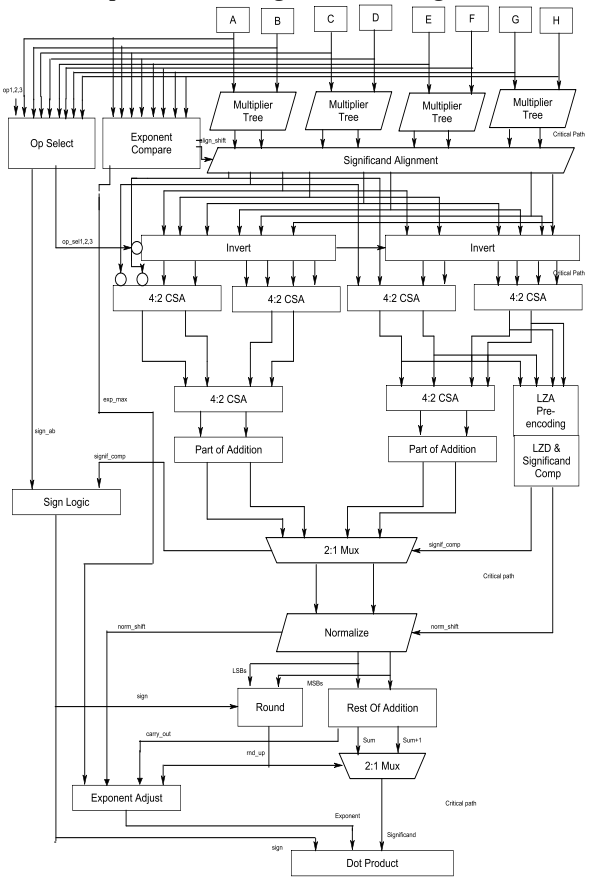

Fig. 4: Floating point four term dot product unit[16]

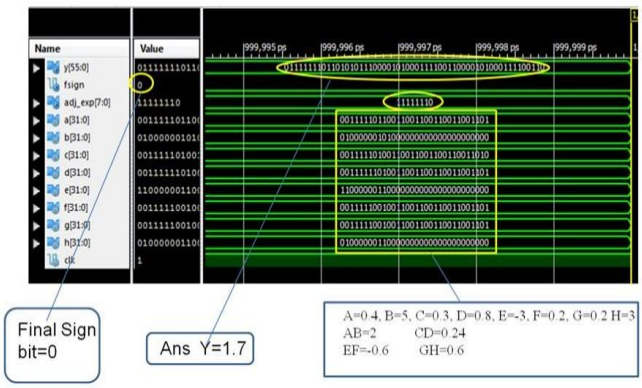

Fig. 5: Four Term Floating Point Dot Product for Valid Input

Fig 5 shows sample test results and Fig. 6 shows test setup for the experimentation. Table VII shows that critical path of Fig 4 design is less as compare to conventional design therefore speed of the this design is improved as compared to conventional design.

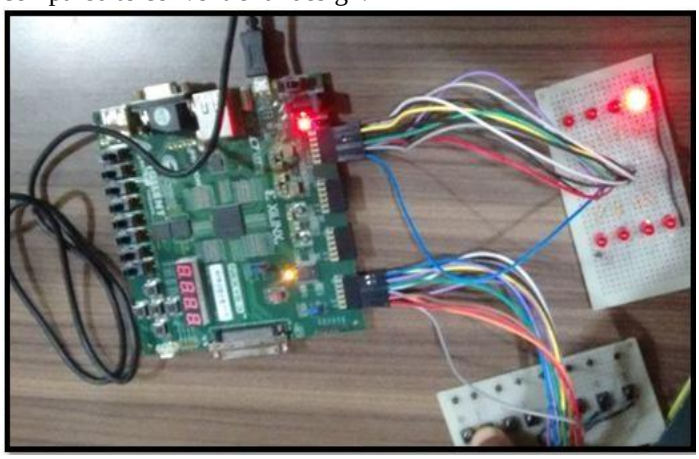

Fig. 6: Experimental Setup

Table Vii: Comparison Result Of Proposed Design With Existing Design (Vertex-5 Xc5vlx110t)

Design	Area (LUT)	Delay(ns)	Power(W)
Four-term Dot product unit	7494	12.917	2.278
Three Term Adder + one discrete Add-Multiply unit	1931+2 DSP48E	124.921	NA
Two Term Dot Product Unit	2446+8DSP8E	107.38	2.076
Two Term Fused Dot product unit	2072+4DSP48E	109.30	2.030
Discrete Design	3034+8DSP48E	178.99	4.398

VI. CONCLUSION

Discrete to FTDPFP implementation shows area overheads are increasing with reduction in delay. Constraint of design varies from application to application. If application has area as a constraint, FTDPFP will be effective. Area, power and speed decide the selection of FP unit. Fused architectures avoid redundancy in architecture and gives optimum solution in terms of APD. Four term computing reduces area and critical path. FTDPFP applies an optimized exponent compare and significant alignment, dual-reduction, early normalization, four-input LZA, and compound addition and rounding reduced area overhead and increases speed compared to previous architectures.

REFERENCES

1. Muller, Jean-Michel, Nicolas Brisebarre, Florent De Dinechin, Claude-Pierre Jeannerod, Vincent Lefevre, Guillaume Melquiond, Nathalie Revol, DamienStehl, and Serge Torres(2010), Handbook of floating-point arithmetic, Springer
2. Montoye, Robert K. and Hokenek, Erdem and Runyon, Stephen L.(1990), Design of the IBM RISC System/6000 floating-point execution unit, IBM Journal of research and development, vol. 34,number 1, pages 59-70, IBM
3. Swartzlander, Earl E and Saleh, Hani HM(2012), FFT implementation with fused floating-point operations, IEEE transactions on computers, volume 61,number 2, pages 284-288, IEEE
4. Swartzlander, Earl E and Saleh, Hani H,Fused floating-point arithmetic for DSP(2008), 42nd Asilomar Conference on Signals, Systems and Computers, pages 767-771,IEEE
5. Saleh, Hani and Swartzlander, Earl E(2008),A floating-point fused add-subtract unit,51st Midwest Symposium on Circuits and Systems, MWSCAS 2008, pages 519-522,IEEE
6. Hokenek, Erdem and Montoye, Robert K and Cook, Peter W(1990),Second-generation RISC floating point with multiply- add fused, IEEE Journal of Solid-State Circuits, volume 25,number 5, pages 1207-1213, IEEE
7. Lang, Tomas and Bruguera, Javier D(2002), Floating-point fused multiply-add with reduced latency, IEEE International Conference on Computer Design: VLSI in Computers and Processors Proceedings 2002, pages 145-150, IEEE
8. Bruguera, Javier D and Lang, Tom(2005), Floating-point fused multiply-add: reduced latency for floating-point addition,17th IEEE Symposium on Computer Arithmetic ARITH-17 2005, pages 42-51,IEEE
9. Roesler, Eric and Nelson, Brent(2002), Novel optimizations for hardware floating-point units in a modern FPGA architecture, Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream, pages 323-345, Springer
10. Roesler, Eric and Nelson, Brent(2002), Novel optimizations for hardware floating-point units in a modern FPGA architecture, Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream, pages 323-345,Springer
11. Sohn, Jongwook and Swartzlander, Earl E(2014), A fused floating-point three-term adder, IEEE Transactions on Circuits and Systems I: Regular Papers, volume 61, number 10, pages 2842-2850,IEEE
12. Kaivani, Amir and Ko, Seokbum(2016), Floating-point butterfly architecture based on binary signed-digit representation, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, volume 24, number 3, pages 1208-1211, IEEE
13. Takahashi, Daisuke(2003),A radix-16 FFT algorithm suitable for multiply-add instruction based on Goedeckermethod,International Conference on Multimedia and Expo, 2003. ICME'03.Proceedings. 2003, volume 2, pages II-845, IEEE
14. Sohn, Jongwook and Swartzlander, Earl E(2012),Improved architectures for a fused floating-point add-subtract unit, IEEE Transactions on Circuits and Systems I: Regular Papers, volume 59, number 10, pages 2285-2291, year 2012, IEEE
15. Saleh, Hani H and Swartzlander, Earl E(2008), A floating-point fused dot-product unit, IEEE International Conference on Computer Design, ICCD 2008, pages 427-431, IEEE
16. Swartzlander Jr, Earl and Saleh, Hani(2012),Floating-point fused dot-product unit, apr 24,Google Patents, US Patent 8,166,091
17. Swartzlander, Earl E and Sohn, Jongwook (2014), Dual-path fused floating-point two-term dot product unit, jan 7, Google Patents, US Patent 8,626,813
18. New, Bernard J(1990), Floating point add/subtract and multiplying assemblies sharing common normalization, rounding and exponential apparatus, jul 24,Google Patents, US Patent 4,943,940
19. Tenca, AlexandreF(2009), Multi-operand floating-point addition,19th IEEE Symposium on Computer Arithmetic, ARITH 2009, pages 161-168, IEEE
20. Sohn, Jongwook and Swartzlander, Earl E(2012), Improved architectures for a fused floating-point add-subtract unit, IEEE Transactions on Circuits and Systems I: Regular Papers, volume 59, number 10, pages 2285-2291,IEEE
21. Jessani, Romesh M. and Putrino, Michael(1998), Comparison of single- and dual-pass multiply-add fused floating-point units, IEEE Transactions on Computers, volume 47, number 9, pages 927-937,IEEE
22. Min, Jae Hong and Sohn, Jongwook and Swartzlander, Earl E(2012), A low-power dual-path floating-point fused add-subtract unit, 2012 Conference Record of the Forty Sixth Asilomar Conference on Signals,Systems and Computers (ASILOMAR),pages 998-1002, IEEE
23. Anane Mohamed and Bessalah Hamid and Issad Mohamed and Anane Nadjia and Salhi Hassen(2008), Higher radix and redundancy factor for floating point SRT division, IEEE Transactions on very large scale Integration (VLSI) systems, volume 16, number 6, pages 774-779, IEEE
24. Bruguera Javier D and Lang Tomas(1999), Leading-one prediction with concurrent position correction, IEEE Transactions on Computers, volume 48, number 10, pages 1083-1097, IEEE
25. Reddy B Naveen Kumar and Sekhar M Chandra and Veeramachaneni Sreehari and Srinivas MB(2014), A novel low power error detection logic for inexact leading zero anticipator in floating point units, 27th Interna- tional Conference on VLSI Design and 13th International Conference on Embedded Systems, 2014, pages 128-132, IEEE
26. Zhang Ge and Qi Zichu and Hu Weiwu(2005), A novel design of leading zero anticipation circuit with parallel error detection, IEEE International Symposium on Circuits and Systems, ISCAS 2005, pages 676-679, IEEE
27. Dimitrak opoulos Giorgos and Galanopoulos Kostas and Mavroke falidis, Christos and Nikolos, Dimitris(2008), Low-power leading-zero counting and anticipation logic for high-speed floating point units, IEEE transactions on very large scale integration (VLSI) systems, volume 16, number 7, pages 837-850, IEEE
28. Schmookler Martin S and Nowka Kevin J(2001), Leading zero anticipation and detection-a comparison of methods,15th IEEE Symposium on Computer Arithmetic Proceedings, pages 7-12, IEEE
29. Kaivani Amir and KoSeokbum(2016), Floating-point butterfly architecture based on binary signed-digit representation, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, volume 24, number 3, pages 1208-1211, IEEE
30. Tao Yao and Deyuan Gao and Xiaoya Fan and Xianglong Ren(2012), Three-operand floating-point adder, IEEE 12th International Conference on Computer and Information Technology (CIT), 2012, pages 192-196, IEEE
31. Kaivani Amir and Ko Seokbum(2016), Floating-point butterfly architecture based on binary signed-digit representation, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, volume 24, number 3, pages 1208-1211, IEEE
32. Tenca AlexandreF(2009), Multi-operand floating-point addition, 19th IEEE Symposium on Computer Arithmetic, ARITH 2009, pages 161-168, IEEE
33. Keshab K Parhi(2007), VLSI digital signal processing systems: design and implementation, John Wiley & Sons
34. Palsodkar Prasanna and Gurjar Ajay(2014), Improved fused floating point add-subtract and multiply-add unit for FFT implementation, 2nd International Conference on Devices, Circuits and Systems (ICDCS), 2014, pages 1-5, IEEE
35. Popalghat Mahesh and Palsodkar Prasanna(2016), Implementation of fused floating point three term adder unit, International Conference on Communication and Signal Processing (ICCSP), 2016 pages 1343-1346, IEEE
36. Narule Omshree and Palsodkar Prasanna(2016), Implementation of three operand floating point adder, International Conference on Communication and Signal Processing (ICCSP), pages 1037-1040, IEEE
37. Prachi Palsodkar, Prasanna Palsodkar(2017),Three operand fused floating point add-subtract unit using redundant adder, IEEE Region 10 Conference TENCON 2017, pages 1343-1346, IEEE
38. Khobragade Preeti and Palsodkar Prachi(2016), Floating point unit using error correction scheme and modified anticipator, Online International Conference on Green Engineering and Technologies (IC-GET), pages 1-5, IEEE
39. Patil Ishan and Palsodkar Prasanna and Gurjar Ajay(2014), Floating Point-based Universal Fused Add-Subtract Unit, Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), pages 259-270, Springer

AUTHORS PROFILE

Dr. Prasanna Palsodkar is an Electronics Engineer-Post graduated in 2007 & awarded a doctorate in 2017. He is currently Assistant Professor of Yeshwantrao Chavan College of Engineering, Nagpur. He is an active volunteer of IEEE. He is mentoring IEEE YCCE Student Branch. He is IEEE

Senior member. He has published more than 20 research paper in reputed international journals and conferences.

Dr. Prachi Palsodkar is an Electronics Engineer-Post graduated in 2007 & awarded a doctorate in 2016. She is currently Assistant Professor of Yeshwantrao Chavan College of Engineering, Nagpur. She is an active volunteer of IEEE. She has Published one book chapter, 11 Journals and 18 International Conferences (01 SCI , 20 Scopus) with more than 150 citation index. She has filed two patents and published three copyrights. She has delivered more than 20 technical talks delivered for faculty as well as for students. She is mentoring IEEE YCCE Student Branch is IEEE Senior member.