Received: December 20, 2016

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Effective Implementation of Dual Path Fused Floating-Point Add-Subtract Unit for Reconfigurable Architectures

Anitha Arumalla[1]* **Madhavi Latha Makkena[2]**

[1]Velagapudi Ramakrishna Siddhartha Engineering College, Vijayawada, India
[2]Jawaharlal Nehru Technological University Hyderabad, Hyderabad, India
** anithaarumalla83@gmail.com*

Abstract: Reconfigurable architectures have provided a low cost, fast turnaround platform for the development and deployment of designs in communication and signal processing applications. The floating point operations are used in most of the signal processing applications that require high precision and good accuracy. In this paper, an effective implementation of Fused Floating-point Add-Subtract (FFAS) unit with a modification in dual path design is presented. To enhance the performance of FFAS unit for reconfigurable architectures, a dual path unit with a modification in close path design is proposed. The proposed design is targeted on a Xilinx Virtex-6 device and implemented on ML605 Evaluation board for single, double and double extended precision. When compared to discrete floating point adder design, the FFAS unit reduces area requirement and power dissipation as the later shares common logic. A Dual Path FFAS (DPFFAS) unit has reduced latency when compared with FFAS unit. The latency is further reduced with the proposed modified DPFFAS when compared with DPFFAS for reconfigurable architectures.

Keywords: Discrete floating-point design, Dual path algorithm, Floating-point arithmetic, Fused floating-point operation, Leading zero anticipation

1. Introduction

In recent processors, the computer arithmetic units accomplish advanced computations such as scientific calculations, high performance graphics, and multimedia signal processing that need complex arithmetic. The binary fixed-point number system cannot provide adequate precision to handle such complex computations. As specified in IEEE Standard -754 [1], floating-point notation represents an extensive numbers range from trivial fractional numbers to extremely large numbers in converse to that of fixed-point number system.

The floating-point notation consists of three parts - sign, significand and exponent. Hence, the floating-point operations may require complex procedures. For example, the operations frequently require the normalization, which causes an increased logic delay. Therefore, improving the performance of floating-point operations has been a research topic in the computer arithmetic field.

Custom designs targeting Application Specific Integrated Circuits (ASICs) were designed if extremely high performance, with respect to area, speed, and power, is demanded for the designs. However, the design time and development cost of ASICs is huge. To address this limitation, reconfigurable architectures like the Field-Programmable Gate Arrays (FPGA) are extensively used. When compared to ASICs, FPGAs are becoming a solution for applications that are developed for small volumes. Many communication and signal processing systems are developed using reconfigurable architectures owing to their faster design turnaround time, lower design cost and availability of high performance devices from vendors like Xilinx, Altera. While the design for custom ICs cannot give similar performance as the design targeted for a generic reconfigurable architecture, there should be designs specific to the target technology. Hence, this paper presents an effective implementation of DPFFAS unit for reconfigurable architectures.

Many Digital Signal Processing (DSP) applications can benefit from the FFAS unit. The FFAS unit generates sum and difference in parallel from two normalized floating point operands as inputs. This unit reduces area when compared to the discrete design. A dual-path algorithm can improve the computational speed of FFAS. The algorithm consists of close and far paths and the path is chosen basing on the difference in exponents. In far path, rounding, addition and subtraction are accomplished in parallel. The close path is divided into three cases depending on exponent difference. For each of the three cases, LZA, addition, and subtraction are executed simultaneously and rounding is not necessary. This latency of DPFFAS can be reduced further if the design is customized to suit the internal architecture of FPGA. The normalization stage in the close path contributes at a larger degree to the overall latency of DPFFAS. The design when implemented for generic reconfigurable architectures like FPGA's, should be customized depending on design datawidth and target architecture datapath. Hence a modified DPFFAS algorithm with a Leading Zero Detector (LZD) in the close path is employed by investigating latencies of LZD and LZA for different significand data width. This algorithm reduces the additional latency introduced by LZA due to fixed finite datapath of FPGAs. A 28% reduction in the latency is observed in modified DPFFAS algorithm when compared with DPFFAS with LZA.

In the rest of the paper, the following content is presented. A brief literature survey is presented under the Section 2. Discrete and fused floating-point add-subtract units are introduced in Section 3. DPFFAS unit architecture is discussed under Section 4. Modified DPFFAS Unit design is presented in Section 5. Results of DPFFAS in comparison with other architectures are listed under Section 6 and conclusion is derived in the subsequent section.

2. Previous Architectures

Floating-point numbers make an effort to represent real numbers with highest accuracy. IEEE Standard 754 has defined different floating point number representation formats, exceptions and error conditions to support diverse precision requirements [2]. Several researchers [3]–[5] have implemented various architectures for Floating-Point Adders (FPA). The approach outlined in [3] allows for the construction of floating-point units with parameter selection like throughput, latency, and area. LOP (Leading One Predictor) algorithm, and 2-path algorithm are introduced in this work. A 2-path adder is designed to trade area for latency and LOP algorithm is introduced to reduce overall critical path delay. But the algorithms proposed in [3] [4]are confined only to unsigned operations with exception handling limited to overflow.

In [4], authors worked to analyze area and maximum achievable throughput of pipelined structures of FPA and multiplier. Area and throughput are traded against the number of pipeline stages as a parameter. The standard three stage algorithm with de-normalization/pre-shifting, significand addition/subtraction and rounding/ normalization is used to increase operating frequency of the design. Although the IEEE-754 formats are used, for the single and double precision implementations, demoralization and Not a Number (NaN) representations were not supported.

In [5], a IEEE standard -754 FPA with double precision is implemented including all rounding modes. A revised version of a FPA is proposed that can achieve accumulation in 2-cycles including some additional hardware which also engages flagged prefix addition to enhance branch resolution times. However, the design in [5] is not supported with performance analysis results. A Parameterized library [6] of floating point open cores considers precision as a significant parameter. Apart from considering precision as a parameter, the number of pipeline stages should be considered as a significant parameter as it affects the throughput of the design.

The fused add-subtract unit design is first introduced in [7] to support application, requiring concurrent addition and subtraction, like radix N butterfly FFT structures. The FAS unit has occupied 40% large area when compared with conventional FPA. Maximum frequency of operation of FAS unit is comparable with that of conventional FPA. Two different fused floating point operations are used for implementing radix-2 and radix-4 butterfly FFT structures [8]. The FFT structure using fused multiply add and fused add-subtract have reported 30% lower area and 15% more speed when compared with conventional FFT structures. The individual FAS however exhibited more delay in comparison with conventional add-sub unit.

Two-stage pipeline architecture for dual path algorithm is presented in [9]. The design has shown significant advantage in area and latency. But the design cannot be readily adapted to reconfigurable architectures. Leading Zero Anticipator (LZA) [10],

[11] is used to replace Leading Zero Detector in normalization step to reduce the latency of FFAS. LZA is used in parallel to addition and subtraction operations to predict the most significant zero in the result of subtraction operation. Though an overall performance is increased, the area requirement is high for LZA algorithm.

A multi threshold voltage technique [12] is used to implement ASIC implementation of floating point addition, subtraction and multiplication operations for reducing power dissipation. A hybrid floating-point (FP) implementation [13] is used to improve performance without incurring the area overhead of full hardware FP units. Add, Subtract, Multiply, sqare-root and division software kernels are integrated into small fixed-point processors with a RISC-like architecture.

A multifunctional floating point unit [14] is designed that includes multiplication above fused add-sub unit producing a hardware efficient implementation of fused floating point arithmetic. In [15], fused arithmetic units including two-term dot product unit and add-subtract unit are designed using radix-16 booth multiplier.

Many of the designs in the literature have considered algorithms for single precision floating point arithmetic. The designs are not evaluated for higher precision arithmetic. Also, reconfigurable architectures have been increasingly used to develop signal processing and communication application. Therefore there is a need to develop architectures specific to reconfigurable architectures by considering the generic structures of the target device. Hence the paper describes an IEEE-754 standard FFAS for signed floating point operations, including all exceptions detailed in the standard. The design is evaluated for single, double and extended double precisions.

3. Discrete and Fused Floating-point Add-Subtract Units

The Floating-point Add-Subtract (FAS) operation can be implemented using two identical FFASs in parallel [9]. Two adders are used to carry out addition and subtraction operations, so as to compute the sum and difference concurrently. A traditional FFAS can be used for each operation. But, most of the logic like alignment, exponent compare and significand swap in both the FFASs is almost similar for the two operations.

In order to decrease area overhead, the common logic is combined for two operations in the FFAS unit. The design of a FFAS unit [9] is as shown in Fig. 1. The FFAS unit computes the sum and difference results concurrently by sharing the common logic. Though different signed combinations are possible for both the operands, only one significant subtraction and addition are performed in the FFAS unit. The appropriate sign for the result is evaluated by sign logic. To reduce the number of addition/subtraction operations required by different signed combinations of the operands, exponent comparison and significand swap logic are used to adjust the significands such that larger operand is made as first operand during subtraction.

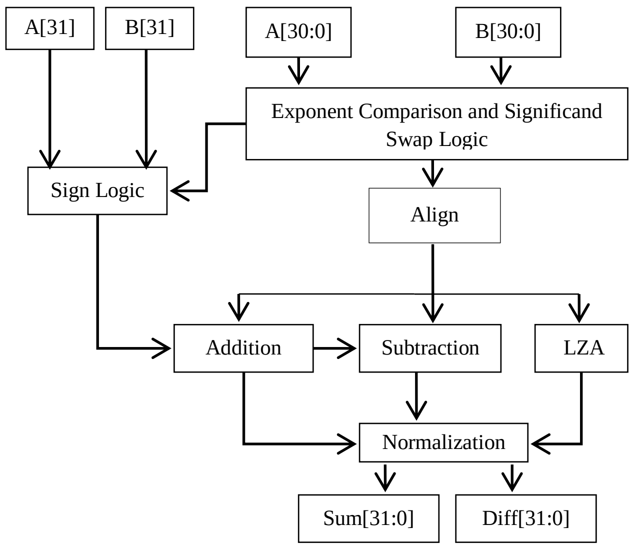


Figure.1 Fused floating-point add-subtract unit

For the sum and difference results, the two operations are explicitly performed. Hence the addition and subtraction blocks are positioned separately and only single LZA [16] and normalization (for the subtraction) are required. LZA generates the number of most significant leading zeros during the subtraction so that its result is immediately normalized. LZA gives the amount of shift required to the exponent adjust logic in normalization stage. The LZA calculation from the significands of subtraction operands is shown in the following equations.

$$sum_i = \left(A_i \text{ xor } B_i\right) = P_i \tag{1}$$

$$carry_i = A_{i+1} \text{ and } B_{i+1} = G_{i+1} \tag{2}$$

$$xor_i = sum_i \text{ xor } carry_i = PG_i \tag{3}$$

$$LZA = PG_1 \text{ and } PG_2 \ldots\ldots \text{ and } PG_i \tag{4}$$

In the normalization stage, exception flag is asserted if any one of underflow, overflow and inexact exceptions arise in the exponent. As common logic is being shared between addition and subtraction operations, the number of control signals required for differentiating the signs and final results are reduced. Thus, the FFAS unit attains lower area and higher speed when compared with discrete FAS units.

4. Dual Path Fused Floating-Point Add-Subtract Unit

A dual-path approach [9] can improve the performance of FFAS unit. The DPFFAS algorithm is used in most high-speed FFASs which are meant for designing custom ICs [5], [17] . Fig. 2 shows the DPFFAS unit. In FFAS unit, the normalization performed after subtraction introduces a large delay. The DPFFAS algorithm eliminates the normalization step if the exponent difference is large. Hence the DPFFAS design enhances the performance when compared with FFAS design.

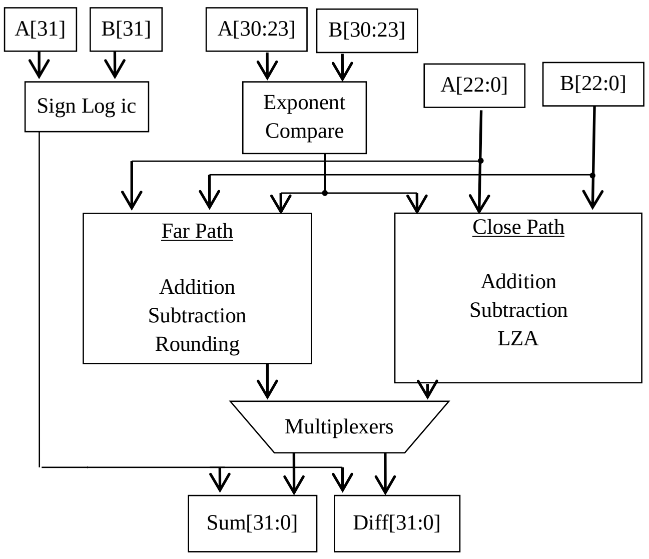


Figure. 2 DPFFAS unit

The dual-path approach is composed of close path and far path logic. If the difference in numbers is large, i.e difference of the exponents is more than ‘1’, then the operands choose far path [9], shown in Fig. 3. In this case, the difference in normalized significands will be large. Hence, huge elimination does not occur during the subtraction and LZA [10], [11] is not necessary. To generate normalized significands, the significands are adjusted by appending ‘1’ to the significand MSB. The operand, whose significand is small, is considered and shifted right by the measure of the exponent difference and a ‘1’ is appended to the MSB. These adder and subtraction blocks perform rounding operation concurrently and produce the relevant results [9]. For both addition and subtraction, the far path eliminates the use of large normalization process as it requires at most one-bit normalization shift.

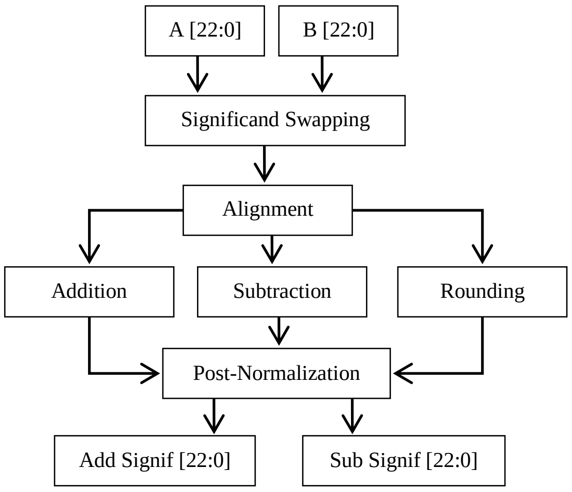


Figure. 3 Far path logic for the DPFFAS unit

The close path[9], shown in Fig. 4, chooses the significands if the exponent difference of operands is either ‘0’ or ‘1’. Three cases are present for the close path alignment relying upon the difference of the exponents:

$$Sig\,A\,[23{:}-1] = \begin{cases} (1,A[22{:}0],0)\ if\ Ex\,A - Ex\,B = 1 \\ (1,A[22{:}0],0)\ if\ Ex\,A - Ex\,B = 0 \\ (01,A[22{:}0])\ if\ Ex\,A - Ex\,B = -1 \end{cases} \quad (5)$$

$$Sig\,B\,[23{:}-1] = \begin{cases} (01,B[22{:}0])\ if\ Ex\,A - Ex\,B = 1 \\ (1,B[22{:}0],0)\ if\ Ex\,A - Ex\,B = 0 \\ (1,B[22{:}0],0)\ if\ Ex\,A - Ex\,B = -1 \end{cases} \quad (6)$$

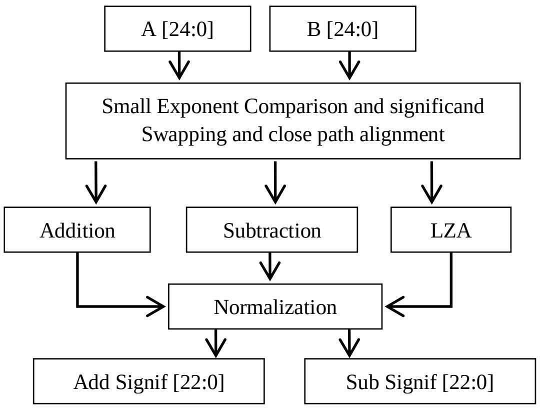


Figure. 4 Close path logic for the DPFFAS

For each case LZA, addition and subtraction are carried out concurrently. In close path there is no explicit alignment done initially. Hence,

- If exponent difference is ‘1’, i.e., A>B, then B must be shifted by 1-bit and a ‘0’ is aligned at MSB.
- If exponent difference is ‘0’, i.e., A=B, no shifting is done.
- If exponent difference is ‘-1’, i.e., A<B, then A must be shifted by 1-bit. As a result a ‘0’ is placed at MSB.

One of the above close path alignment results is selected depending on the exponent difference, computed from the two least significant bits of the exponents. Since the significands in the close path are mis-aligned at most by 1-bit, rounding is not needed.

5. Modified Dual Path Fused Floating-Point Add-Subtract Unit

The DPFFAS exhibits greater critical path delay when targeted to generic architectures like FPGAs, as the logic should be implemented using 6 input LUT. The increase in delay is because of the presence of LZA in the close path, when targeted for FPGAs. The LZA is used to predict the leading zeros in the result of the significand simultaneously while addition and subtraction operations are performed. As the number of significand bits are increased, the advantage of predicting the leading zeros ahead of subtraction operation diminishes. The subtraction operation performed is a two’s complement addition, using fast carry chains in FPGA CLBs, while LZA could not be speeded up, as it also needs a recursive operation.

The dual-path design with LZA can be considered efficient only when the delay of LZA is less than the delays combined with adder block and LZD together. But the delay obtained for single precision DPFFAS by LZA is 12.5ns, which is greater than the delay of adder (3.6ns) and LZD (4.5ns) blocks together being 8.1ns. The latency of LZA and subtraction with LZD are shown in Fig. 5.

From the figure it is evident that the advantage of LZA can be seen if the number of the significand bits is less than 20. But for higher precision designs LZA increases the overall latency. Instead of using LZA in close path, LZD can be used to reduce latency to a great extent which produces a modified DPFFAS. The modification in close path algorithm is presented in Fig. 6.

By using this approach, area is also reduced compared to the dual-path with an LZA in close path (which uses additional hardware). Hence a design with LZD can be used in FPGAs to achieve better performance. The design supports all five IEEE Standard-754 [1] rounding modes and exceptions. Hence for reconfigurable architectures such as Xilinx Virtex-6 devices, dual path design with LZA in close path increases the latency. This is because of the presence of fast look-ahead carry logic in configurable logic blocks which effectively synthesize adder blocks [18].

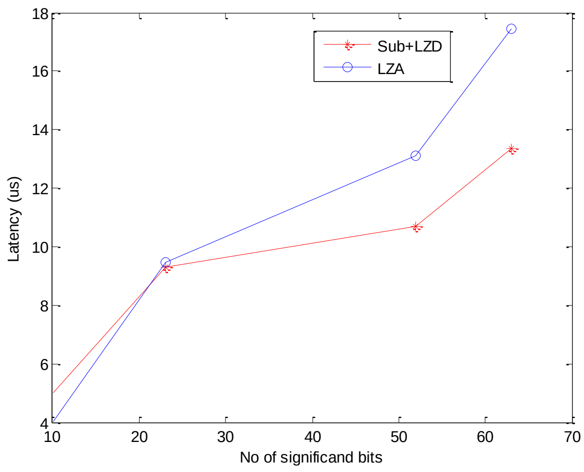


Figure. 5 Latency of Subtraction + LZD and LZA for various number of significand bits.

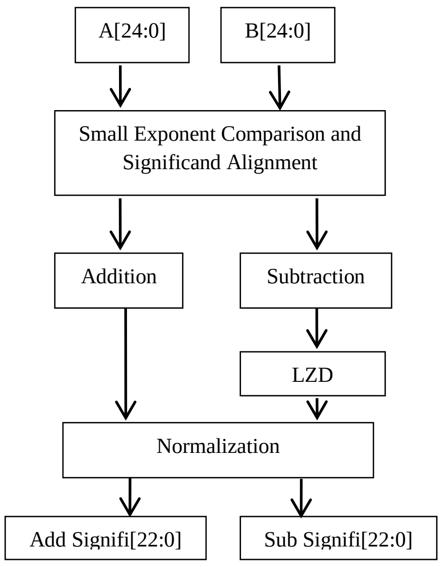
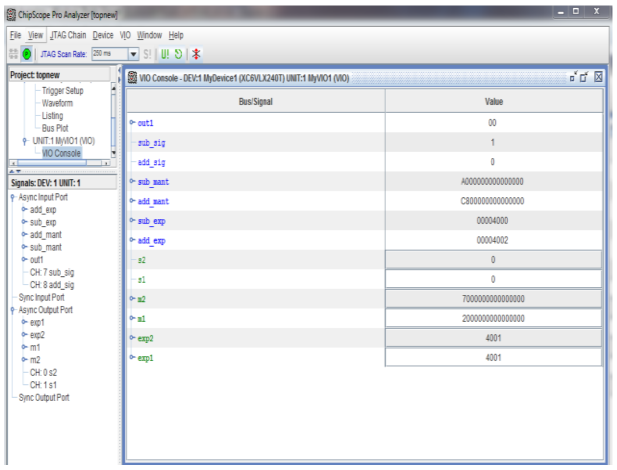


Figure. 6 Modified close path logic for DPFFAS.

In a modified DPFFAS with an LZD in the close path, the critical path latency is reduced proportional to the increase in number of significand bits. For many purposes the accuracy offered by the single precision format is inadequate. Greater accuracy can be achieved by using double precision format which uses 64 bits to store each floating-point value. Even greater accuracy may be achieved from the double extended precision format, which uses 80 bits of information [2]. Hence the proposed design is intended to implement floating point arithmetic for double extended precision. However the results for single precision, double precision and extended double precision floating point implementations were also presented.

6. Results

The floating-point designs are implemented in single, double and double extended precisions to evaluate their performance. The implementations of double and double extended precisions can be done by extending the single precision implementation. For simplicity, only single precision designs are described in the architectures.



Bus/Signal	Value
out1	00
sub_sig	1
add_sig	0
sub_mant	A000000000000000
add_mant	C800000000000000
sub_exp	00004000
add_exp	00004002
s2	0
s1	0
m2	7000000000000000
m1	2000000000000000
exp2	4001
exp1	4001

Figure. 7 Chipscope debug for modified DPFFAS with LZD

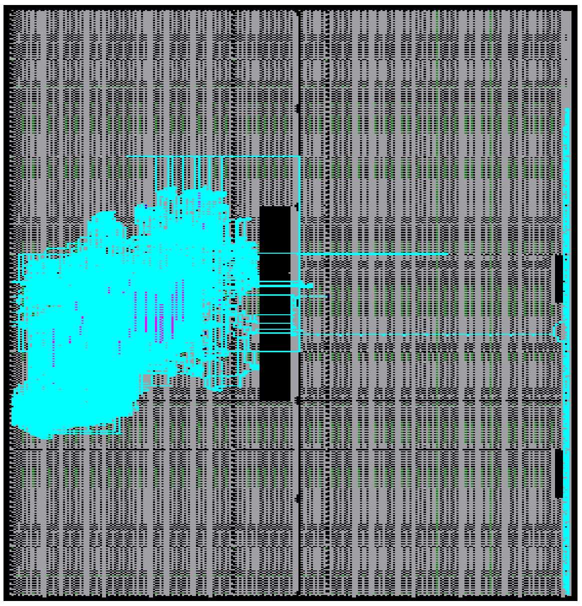


Figure. 8 Routing floorplan of modified DPFFAS with LZD

Each design is implemented using Verilog-HDL targeted on a Xilinx Virtex-6 device. The proposed algorithm is verified with random test vectors for 100% code coverage using QuestaSim tool. Prototype validation of the algorithm of the proposed architecture is performed on ML605 Virtex 6 FPGA Evaluation board using Xilinx chipscope. The experimental results of modified DPFFAS implemented on ML605 Virtex 6 FPGA Evaluation board and observed using chipscope are shown in Fig. 7. The routing floorplan of modified DPFFAS with LZD for double precision floating point is shown in Fig. 8.

Table 1. Fused Floating-Point Add-Subtract Design Comparison

Single Precision				
	Discrete [3]	Fused [7]	Fused + Dual path (using LZA)[9]	Fused + Dual path (using LZD)
No. of slice LUTs	1,473	1,063	2,504	1,834
Latency(ns)	24.309	25.582	20.1	17.359
Throughput(1/ns)	0.041	0.039	0.049	0.057
Power(mW)	1.99	1.45	2.09	1.61
Double Precision				
	Discrete [3]	Fused [7]	Fused + Dual path (using LZA) [9]	Fused + Dual path (using LZD)
No. of slice LUTs	3,383	2,407	5,448	4,320
Latency(ns)	31.311	32.113	29.333	19.867
Throughput(1/ns)	0.0319	0.0311	0.034	0.050
Power(mW)	2.49	2.07	4.72	3.75
Double Extended Precision				
	Discrete [3]	Fused [7]	Fused + Dual path (using LZA) [9]	Fused + Dual path (using LZD)
No. of slice LUTs	7,089	4,420	8,134	6,574
Latency(ns)	37.403	39.812	36.282	24.886
Throughput(1/ns)	0.026	0.025	0.027	0.040
Power(mW)	3.50	2.33	5.20	4.47

In order to evaluate the design performance, area, latency, throughput, and power consumption are compared with three other architectures. The discrete floating point add-subtract unit [3], FFAS [7], DPFFAS with LZA [9] are re-implemented for a Xilinx Virtex 6 target device to have valid comparision between the architectures. The results for the four designs in single precision, double precision and double extended precision implementations are shown in Table I.

Since the FFAS unit shares much of the common logic, it saves more than 27% of the area over the discrete FAS unit. The existing dual-path design (using LZA in close path) adds parallelism to the design but because of additional hardware and significant routing delays it does not significantly improve overall latency. When compared to an LZD in the close path, the close path using LZA is not an effective design for FPGAs, because of the requirement of additional slices for LZA and adds significant routing and gate delay [19]. Close path with LZA design requires long carry chain logic to be implemented in FPGAs. The carry chain length is proportional to significand width. While the design with LZD is implemented using a zero comparator that is free from carry chain and hence has less latency. The modified DPFFAS unit requires more area than the traditional and fused floating-point units due to the presence of three parallel additions, subtractions for close path. However, the modified DPFFAS design reduces the latency by 28% when compared to discrete FAS unit. The area and latency of modified DPFFAS design with LZD are reduced by 19% and 14% respectively, when compared to existing dual-path design with three LZAs.

7. Conclusion

An effective implementation of DPFFAS unit for reconfigurable architectures is presented in this paper. The architecture, though aimed for high throughput implementation, offers a significant reduction in area and power. The increase in throughput is due to LZD that can be easily implemented with fine grained target technology. But LZA, whose area increases linearly with significand width, exhibits higher throughput only for ASIC implementation. However, the choice of architecture always depends on the implementation technology and the design constraints. The modified DPFFAS unit can improve the speed of computation of digital signal processing applications, such as DCT and FFT butterfly operations, that require concurrent floating point addition subtraction operations when targeted for reconfigurable architectures. As the large number of floating point computation is required, the proposed design can significantly reduce area and power dissipation of the entire application. This paper demonstrates an effective architecture for DPFFAS algorithm to perform concurrent floating-point addition and subtraction operation. Area, latency, throughput, and power consumption are compared with the traditional parallel implementation, FAS, DFFAS algorithm. The modified DFFAS algorithm can be extended to develop pipelined FFAS unit. Development of signal transformation architectures for FFT, DCT can be done using modified DFFAS algorithm.

References

[1] M. Standards Committee of the IEEE Computer Society, "IEEE Std 754^{TM}-2008 (Revision of IEEE Std 754-1985), IEEE Standard for Floating-Point Arithmetic," 2008.

[2] S. B. Furber, *ARM system architecture*. Addison-Wesley, 1996.

[3] J. Liang, R. Tessier, and O. Mencer, "Floating point unit generation and evaluation for FPGAs," in *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2003, pp. 185–194.

[4] G. Govindu, Ling Zhuo, Seonil Choi, and V. Prasanna, "Analysis of high-performance floating-point arithmetic on FPGAs," in *Proceedings of 18th International Parallel and Distributed Processing Symposium.*, 2004, pp. 149–156.

[5] A. Beaumont-Smith, N. Burgess, S. Lefrere, and C. C. Lim, "Reduced latency IEEE floating-point standard adder architectures," in *Proceedings of 14th IEEE Symposium on Computer Arithmetic*, 1999, pp. 35–42.

[6] G. Govindu, G. Govindu, R. Scrofano, and V. K. Prasanna, "A library of parameterizable floating-point cores for FPGAs and their application to scientific computing," in *in Proc. of International Conference on Engineering Reconfigurable Systems and Algorithms*, 2005, pp. 137--148.

[7] H. Saleh and E. E. Swartzlander, "A floating-point fused add-subtract unit," in *Proc. of 51st Midwest Symposium on Circuits and Systems*, 2008, pp. 519–522.

[8] E. E. Swartzlander and H. H. M. Saleh, "FFT Implementation with Fused Floating-Point Operations," *IEEE Trans. Comput.*, vol. 61, no. 2, pp. 284–288, Feb. 2012.

[9] J. Sohn and E. E. Swartzlander, "Improved Architectures for a Fused Floating-Point Add-Subtract Unit," *IEEE Trans. Circuits Syst. I*

Regul. Pap., vol. 59, no. 10, pp. 2285–2291, Oct. 2012.

[10] H. Suzuki, H. Morinaka, H. Makino, Y. Nakase, K. Mashiko, and T. Sumi, "Leading-zero anticipatory logic for high-speed floating point addition," *IEEE J. Solid-State Circuits*, vol. 31, no. 8, pp. 1157–1164, 1996.

[11] M. S. Schmookler and K. J. Nowka, "Leading zero anticipation and detection-a comparison of methods," in *Proceedings of 15th IEEE Symposium on Computer Arithmetic*, 2001, pp. 7–12.

[12] S. Kukati, D. . Sujana, S. Udaykumar, P. Jayakrishnan, and R. Dhanabal, "Design and implementation of low power floating point arithmetic unit," in *International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, 2013, pp. 205–208.

[13] J. J. Pimentel, B. Bohnenstiehl, and B. M. Baas, "Hybrid Hardware/Software Floating-Point Implementations for Optimized Area and Throughput Tradeoffs," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. PP, no. 99, pp. 1–14, 2016.

[14] J. Sharma, P. Tarun, S. Satishkumar, and S. Sivanantham, "Fused floating-point add and subtract unit," in *Proc. of IEEE International Conference on Green Engineering and Technologies (IC-GET)*, 2015, pp. 1–5.

[15] E. Prabhu, H. Mangalam, and S. Karthick, "Design of area and power efficient Radix-4 DIT FFT butterfly unit using floating point fused arithmetic," *J. Cent. South Univ.*, vol. 23, no. 7, pp. 1669–1681, Jul. 2016.

[16] M. S. Schmookler and D. G. Mikan Jr., "Two state leading zero/one anticipator (LZA)," U.S. Patent No. 5493520, 1996.

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

[18] Xiinx, "UG364: Virtex-6 FPGA Configurable Logic Block," 2012.

[19] A. Malik, Dongdong Chen, Younhee Choi, Moon Lee, and Seok-Bum Ko, "Design tradeoff analysis of floating-point adders in FPGAs," *Can. J. Electr. Comput. Eng.*, vol. 33, no. 3/4, pp. 169–175, 2008.