http://www.jctjournals.com

ISSN(ONLINE):2278 - 3814

Volume-12, Issue-07, July 2023

JOURNAL OF COMPUTING TECHNOLOGIES (JCT)

International Journal

Page Number: 01-04

Floating-Point Butterfly Architecture Based On Binary Signed-Digit Representation :A Review

[1]Rahul Nath Tiwari, [2]Prof. Nishi Pandey,[3]Prof. Abhishek Agwekar

[1]M. Tech Scholar, [2]Asst.professor, [3]Head of Department,

[1,2,3] TIEIT, Bhopal

[1]rahulnathtiwari007@gmail.com,[2] nishi.pandey@trubainstitute.ac.in, [3]abhiagwekar@gmail.com

***Abstract*— Fast Fourier transform (FFT) coprocessor, having a significant impact on the performance of communication systems, has been a hot topic of research for many years. The FFT function consists of consecutive multiply add operations over complex numbers, dubbed as butter fly units. Fast Fourier transform (FFT) is one amongst the most necessary tools in digital signal processing in addition as communication system as a result of transforming time domain to S-plane is very convenient using FFT. Applying floating-point(FP) arithmetic to FFT architectures, specifically butterfly units, has become additional common recently. However, the main drawback of FP butterfly is its slowness as compared with its fixed point counterpart.**

***Keywords*— FET, DSP, Signal, IOT, API and OS.**

I. INTRODUCTION

In recent times, a great deal of importance within the field of big information and its investigation has rise, mainly driven from extensive range of analysis challenges strappingly related to bonafide applications, like modeling, processing, querying, mining, and distributing Large-scale repositories. Floating-point (FP) implementation of fast Fourier transforms (FFT) units has recently become a popular research topic. Providing a wide dynamic range is the main advantage of FP over fixed point arithmetic. On the other hand, FP implementations require more area and are relatively slower than fixed-point architectures. Given that FFT architecture consists of multiple consecutive arithmetic operations, merging these operations leads to higher overall performance and lower area cost. Multimedia processing has been finding additional and more applications in mobile devices. A lot of effort should be spent to manage the complexness, power consumption, and time to market of the modern multimedia system system-on-chip (SoC) designs. However, multimedia system algorithms are computationally intensive, made in expensive FP arithmetic operations instead of easy logic. FP arithmetic hardware offers a large dynamic vary and high computation precision, however occupies giant fractions of total chip space and energy budget.

1.2 Floating-Point Arithmetic :-Floating-point arithmetic is being used, and preferred over fixed-point, in many applications due to the fact that it provides a large range of numbers and a high degree of precision. It is also common to be used in a variety of Digital Signal Processing (DSP) applications because it relieves the designer of numerical issues e.g., scaling, overflow, and underflow. A floating-point number, as represented in Eqn. 1.1, consists of four components; namely, sign, significant, base and exponent.

$$(-1)^{sign} \mathbf{x} \text{significant} \mathbf{x} (\text{base})^{exponent} \quad \ldots\ldots\ldots\ldots\ldots\ldots (1.1)$$

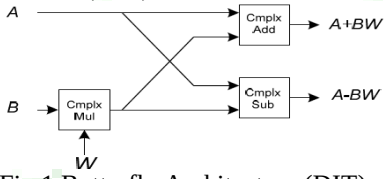

Fig.1 Butterfly Architecture (DIT)

There are two methods to implement a butterfly unit: 1) conventional 2) Golub's approach.Fig. 1.2 shows the implementation of a DIT butterfly with expanded complex numbers using the conventional approach. Accordingly, it consists of four multipliers and six adders/subtracters. Itshould be noted that, given the constant values of twiddle factors (W), the multipliers are constant and can be implemented via a series of shifters and adders in lieu of the multiplier tree.

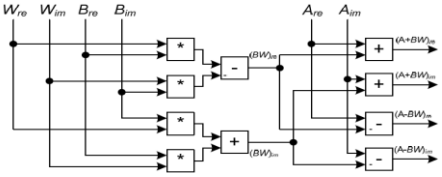

Fig. 2 DIT Butterfly architecture using conventional approach

II. LITERATURE REVIEW

Amir Kaivani et.al [1] Flotating-Point Butterfly Architecture Based on Binary Signed-Digit Representation in this paper planned a high-speed FP butterfly design that is faster thanprevious works however at the value of higher space. The reason for this speed improvement is twofold: 1) BSD illustration of the significant that eliminates carry propagation and 2) the new FDPA unit planned during this brief. This unit combines multiplications and additions needed inFP butterfly; therefore higher speed is achieved by eliminating additional LZD, normalization,and rounding units. More analysis is also envisaged on applying dual-path FP design to the threeoperandFP adder and using different redundant FP representations. Moreover, use of improvedtechniques within the termination phase of the planning (i.e., redundant LZD, standardization,and rounding) would lead to faster architectures, although higher area prices are expected.

Earl E. Swartzlander Jr-Point this paper describes the planning of 2 new fused floating-point arithmetic units and their application to the implementation of FFT butterfly operations. Though the fused add- subtract unit is particular to FFT applications, the fused dot product is applicable to a wide style of signal process applications. Each the fused real unit and also the fused add-subtract unit are smaller than parallel implementations created with discrete floating-point adders and multipliers. The fused dot product is faster than the standard implementation, since rounding and normalization isn't needed as a part of every multiplication. Due to longer interconnections, the fused add-subtract unit is slightly slower than the discrete implementation. The area of the fused radix-2 butterfly is 35 % smaller and also the latency is 15 % less than the discrete radix-2 FFT butterfly parallel implementation. The area of the fused radix-4 butterfly is 26 % smaller and also the latency is 13 % less than the discrete radix-4 FFT butterfly parallel implementation. Each fused butterflies use fewer rounding operations resulting in a lot of accurate results than the discrete approaches. The errors for a 64K purpose FFT are about 25 % less for the fused implementations.

Alexandre F. TencaMulti-operand Floating-point Addition in this paper demonstrated the feasibleness of implementing multi- operand floating-point adders to induce additional accurate operations than equivalent networks of FPADD2s. The work was focused on 3-input FP adders however the discussion concerning design problems and various solutions is additionally applicable to adders for additional operands. The experimental results show that the 3-input FP adder design may be synthesized to reach shorter or similar delays than the network of 2-input FP adders, with comparable or better area, and additional accuracy. the capability of this component to get outputs honoring the commutative property for its inputs is very advantageous to eliminate the ordering drawback (non-associative behavior) imposed at the algorithmic rule level on networks of FPADD2s.

Yao Tao Three-Operand Floating-Point Adder in this paper projected an improved design of 3- operand FP adder. The principle to choose the internal width is given and also the realignment technique is used to avoid quite one sticky bit generated, which may reduce price and avoid the lost of accuracy. In addition, an OR-logic network rather than the comparer to observe the catastrophic cancellation is used within the improved one. Many sophisticated techniques, like compound adder and LZA, are used to optimize the design. The experiment shows that our design has a competitive area and delays by comparison with each a basic 3- operand FP adder and a network of 2-operand FP adders. The perform verification shows that our design has a similar accuracy as a full precision FP adder.

Asger Munk Niels-Point Adder that Conforms withthe Pipelined Packet- An addition algorithmic rule conforming with the packet-forwarding pipeline paradigm is conferred. The adder accepts one operand within the standard format and therefore the second operand during a packet-forwarding floating-point format [8]. Note that a standard format quantity is trivially translated to conform to the packet- forwarding format and, therefore, the idea on the second operand doesn't restrict inputing 2 normal format operands. The sum is output within the packet-forwarding floating-point format starting at the end of the second cycle, similarly as within the standard format at the end of the fourth cycle. This algorithmic rule rearranges and avoids non essential 2:1-addition steps of the normal floating-point addition implementations. In fact, the process of a forwarded result that's output within the packet-forwarding format doesn't contain a 2:1- addition of the length of thesignificant; the only non-redundant addition is that the exponent subtraction. This permits outputting of a nearly complete significant termed the principal part packet at the tip of the second cycle. The rounding, that takes place during the third and fourth cycles [8], produces the carry-round packet at the end of the third cycle and also the normal format sum at the end of the fourth cycle. The worth encoded by the packet-forwarding floating-point format output is similar to the quality IEEE 754 rounded output. By these suggests that, only one 2:1-addition is performed and it's deferred to the latter portion of the rounding part, wherever it contributes no delay to the info forwarding method. All the additions within the initial 2 cycles are 4:2- and 3:2- redundant additions.

Jae Hong Min-Point Fused FFT Butterfly Arithmetic Unit with Merged Multiple-Constant Multipli in this paper presents a low-power and high-speed floating purpose fused FFT design. A butterfly unit with Multiple- Constant Multiplication has lower power consumption. However, it usually has lower precision. To increase the precision, a new butterfly architecture using merged MCMs was introduced. The new MMCM butterfly design reduces the relative absolute error by 65th with slightly increased power consumption compared to the MCM implementation. The mapping logic and further additions cause a small power consumption increase of about 12.5%.

Jongwook Sohn Improved Architectures for a Floating-Point Fused Dot Product Unitfloating-point fused two-term

dot product unit. Several optimizations are applied to improve the performance: 1) A new alignment scheme, 2) Early normalization and fast rounding, and 3) Four-input LZA. The enhanced floating-point fused dot product unit reduces the area and power consumption by 25% and improves the speed by 15% compared to the traditional floating-point fused dot product unit. Further improvement is achieved by the dual-path algorithm. The dual- path floating-point fused dot product unit consists of a far path and a close path and one path is selected based on the exponent difference. Since the dual-path design eliminates unnecessary logic in each path so that the latency is reduced by about 10% compared to the enhanced single path design. The pipelining is applied to improve the throughput. Based on the data flow analysis, the dual-path floating-point fused dot product unit can be split into three stages. Sincethe latencies of the three stages are fairly well balanced, the throughput is 2.8 times that of the non-pipelined design.

Fang Fang-Point Arithmetic: Case Study of Inverse Discretearithmetic, focusing on the most critical arithmetic operators (addition, multiplication), and the most common parameterizations useful for multimedia tasks (bit-width, rounding modes, exception handling, radix). With these libraries, we can easily translate a standard FP program to a lightweight FP program, and explore the system numerical performance versus hardware complexity tradeoff. An H.263 video codec is chosen to be our benchmark. Such media applications do not need a wide dynamic range and high precision in computations, so the lightweight FP can be applied efficiently. By examining the histogram information of FP numbers and relationship between PSNR and bit-width, we demonstrate that our video codec have almost no quality degradation when more than half of the bit-width in standard FP is reduced. Other features specified in the standard FP, such as rounding modes, exception handling and the radix choice are also discussed for this particular application. Such optimization offers huge reduction in hardware cost. In the hardware implementation of IDCT, we combined two FP adders in a butterfly (basic component of IDCT), which further reduced the hardware cost. At last, we show that power consumption of the lightweight FP IDCT is only 10.5% of the standard FP IDCT, and comparable to the fixed-point IDCT.

Amir Kaivani et -point FFT butterfly architectures based on multi-consists of multiple consecutive arithmetic operations over complex numbers. Applying floating- point arithmetic to FFT coprocessors leads to a wider dynamic range and allows the coprocessor to collaborate with general purpose processors via the standard floating-point arithmetic. This offloads compute-intensive tasks from the primary processor and overcomes floating-point concerns such as scaling and overflow/underflow detection. The downside, however, is that floating-point units are slower than the fixed-point counterparts. One of the popular ways to improve the speed of floating-point FFT units is to merge the arithmetic operations inside the butterfly units of FFT architecture. This leads to a butterfly architecture based on multioperand adders. Butterfly units are designed, in two of the most recent works, using three-operand and four-operand adders. However, the work reported here by the present authors goes further and a butterfly architecture based on a five-operand adder is proposed. Simulation results demonstrate that the proposed butterfly architecture is 50% smaller than the fastest previous work with about 17% latency overhead. Compared with the smallest previous work, the proposed design is 47% smaller and 8% faster.

III. PROPOSED METHODOLOGY

The FFT could be implemented in hardware based on an efficient algorithm in which the N-input FFT computation is simplified to the computation of two (N/2)-input FFT. We are also implementing proposed butterfly unit in 8 point FFT. The Continuing this decomposition leads to 2-input FFT block, also known as butterfly unit. The proposed butterfly unit is actually a complex fused-multiply add with FP operands. Expanding the complex numbers, Fig. 3.1 shows the required modules.

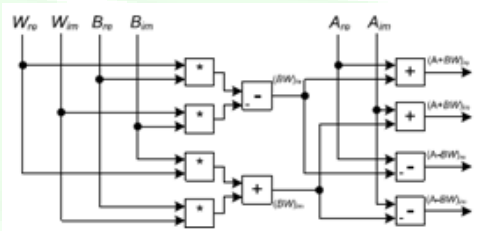

Fig.3 FFT butterfly architecture with expanded complex numbers

AccordingtoFig.3.1,the constituent operations for butter fly unitarea dot-product(e.g.,$B_{re}W_{im}$ +$B_{im}W_{re}$) followed by an addition/subtraction which leads to the proposed FDPA operation (e.g.,$B_{re}W_{im}$ +$B_{im}W_{re}$+A_{im}).Realization information of FDPA, over FP operands ,is discussed below. The exponents of all the inputs are assumed and represented in subtracting the bias), while the significant of Are, Aim, Bre, and Bim are represented in BSD. Within this representation every binary position takes values by one negative-weighted bit (negabit) and one positive-weighted bit (posibit). The carry-limited addition circuitry for BSD numbers is shown in Fig. 3.2, where capital (small) letters symbolizesnegab its (prohibits). The critical path delay of this adder consists of three full-adders. The projected FDPA consist of an unnecessary FP multiplier tag on by an unneeded FP three-operand adder.

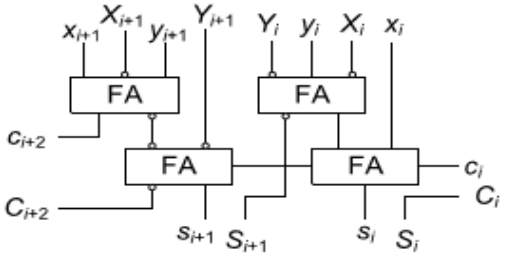

Fig.4 BSD adder (two-digit slice)

IV. IMPLEMENTTATION

Computer arithmetic is concerned with the hardware realization of mathematical formulas, algorithms, and complex models from a theoretical world. Hardware functions calculate -point and scientific notations (floating-point).In computing, a fixed-point number representation is a real data type for a number that has a fixed number of digits after (and sometimes before) the radix point. Fixed-point number representations are much less complicated (and less computationally demanding) than floating point number representations [6]. Fixed-point numbers are useful for representing fractional values, usually in base 2, when the executing processor has no floating point unit (FPU) or if fixed-point provides improved performance or accuracy for the application at hand [7]. A fixed-point number may be written as I.F, where I represents the integer part, '.' is the radix point, and F represents the fractional part. In binary fixed-point numbers, each magnitude bit represents a power of two, while each fractional bit represents an inverse power of two.

V. CONCLUSION

In this research work proposed an improved architecture of a high-speed FP butterfly, which is faster than previous works but at the cost of higher area. In this proposed work proposes a fast FP butterfly unit using a devised FP fused-dot product-add (FDPA) unit. In this research is applying dual-path FP architecture to the three-operand FP adder and using other redundant FP representations. Moreover, use of improved techniques in the termination phase of the design (i.e., redundant LZD, normalization, and rounding) would lead to faster architectures, though higher area costs are expected. In this proposed work Area is reduced. In the proposed work the design of two new fused floating-point arithmetic units and their application to the implementation of FFT butterfly operations. In this research work 32 point FFT architecture is used. In this Combiner and splitter are used. In this proposed work floating point arithmetic is used. In this proposed work area is reduced than previous research work and power consumption is less. This proposed work describes two fused floating-point operations and applies them to the implementation of fast Fourier transform (FFT) processors.

References

[1] Thamizharasan .V Et all "An Efficient VLSI Architecture for FIR Filter using Computation Sharing Multiplier on International Journal of Computer Applications (0975 – 8887) Volume 54– No.14, September 2012.

[2] Deepak Kumar Patel, Raksha Chouksey and Dr. Minal Saxena, "Design of Fast FIR Filter Using Compressor and Carry Select Adder", 2016 3rd International

[3] Conference on Signal Processing and Integrated Networks (SPIN).

[4] Basant Kumar Mohanty, and Pramod Kumar Meher, "High-Performance FIR Filter Architecture for Fixed and Reconfigurable Applications", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 78, No.06, April 2016.

[5] K. Durga and Mrs. A. Sivagam, "Efficient Adaptive RLFIR Filter based on Distributed Arithmetic Logic Using Reversible gates", International Conference on IEEE 2016.

[6] Indranil Hatai, Indrajit Chakrabarti, and Swapna Banerjee, "An Efficient VLSI Architecture of a Reconfigurable Pulse-Shaping FIR Interpolation Filter for Multi-standard DUC", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 23, No. 6, June 2015.

[7] S. Padmapriya and Lakshmi Prabha V., "Design of a power optimal reversible FIR filter for speech signal processing", International Conference, pp. 01-06, ICCCI 2015.

[8] Kiran Joy and Binu K Mathew, "Implementation of a FIR Filter Model using Reversible Fredkin Gate", Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on IEEE Xplore**,** 22 December 2014.

[9] Ravi H Bailmare, S. J. Honale And Pravin V Kinge, "Design And Implementation of Adaptive FIR Filter using Systolic Architecture", In International Journal of

[10] Current Engineering And Technology , Vol.4, No.3, June 2014.

[11] M. Usha, R. Ramadoss, "An Efficient Adaptive Fir Filter Based On Distributed Arithmetic", International Journal of Engineering Science Invention, Vol. 3, Issue. 4, pp. 15-20, April 2014.

[12] Sang Yoon Park and Pramod Kumar Meher, "Low power, High-throughput And Low- Area Adaptive FIR Filter Based on Distributed Arithmetic", in IEEE Transactions On Circuits And Systems-ii, Vol. 60, No. 6, pp. 346- 350, 2013.

[13] Basant K. Mohanty, And Pramod Kumar Meher, "A High-Performance Energyefficient Architecture For FIR Adaptive Filter Based On New Distributed Arithmetic Formulation Of Block LMS Algorithm", In IEEE Transactions On Signal Processing, Vol. 61, No. 4, February, 2013.

[14] Pallavi Saxena, Urvashi Purohit, Priyanka Joshi, "Analysis of Low Power, Area Efficient and High Speed Fast Adder", In International Journal Of Advanced

[15] Research In Computer And Communication Engineering, Vol. 2, Issue 9, September 2013.

[16] H. Thapliyal, N. Ran-Ganathan and S. Kotiyal, "Design of Testable Reversible Sequential Circuits", IEEE Transactions on VLSI, pp. 1-9, 2012.

[17] M. Chuang and C. Wang, "Reversible sequential element designs", Proceedings of the IEEE Asia and South Pacific Design Automation Conference, pp. 420-425, 2007.

[18] G. Challa Ram and D.Sudha Rani, "Area Efficient Modified Vedic Multiplier", 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT).

