International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 07
March 2018

Realization of Building Blocks of Floating Point Butterfly Architecture

Bharathi.R & Dr.M.Ramana Reddy
[1]PG Scholar, RGMCET, Nandyal, Kurnool, AP, India.
[2]Professor,Dept of ECE, RGMCET, Nandyal, Kurnool, AP

Abstract: *Fast Fourier transform (FFT) coprocessor, having a noteworthy crash on the performance of communication systems. The FFT function consists of uninterrupted multiply add operations over complex statistics, dubbed as butterfly units. By applying floating-point (FP) arithmetic to FFT architectures, expressly butterfly units, has become more popular recently. It off-load compute-intensive errands from general-purpose processors by dismissing FP (e.g., scaling and overflow, underflow etc). However, the key downside of FP butterfly is its slowness in contrast with its fixed-point equal. This reveals the spur to develop a high-speed FP butterfly architecture to moderate FP slowness. This brief presents fast FP butterfly unit using a devised FP fused-dot-product-add (FDPA) unit, based on carry select adder (CSA) in existing. This brief proposes a fused floating-point operations and applies them to the implementation of fast Fourier transform (FFT) processors. The fused operations are a two-term dot product and an add-subtract unit, FP fused-dot product-add (FDPA) unit, based on binary signed-digit (BSD) representation and compared with CSA representation. In this brief different blocks used in floating point Butterfly Architecture are designed. Simulation results are observed using Cadence tool.*

Index Terms— Binary-signed digit (BSD) representation, butterfly unit, complex number system, fast Fourier transform (FFT), floating-point (FP).

I. INTRODUCTION

Fixed point Representation.

A fixed-point representation of a number may be thought to consist of 3 parts: the sign field, integer field, and fractional field. One way to store a number using a 32-bit format is to reserve 1 bit for the sign, 15 bits for the integer part and 16 bits for the fractional part. A number whose representation exceeds 32 bits would have to be stored inexactly. On a computer, 0 is used to represent + and 1 is used to represent

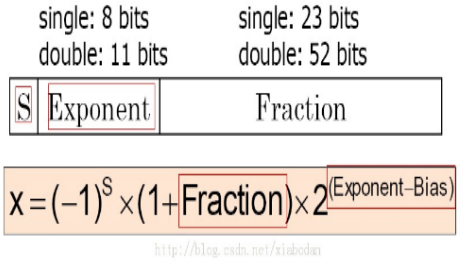
Example. The 32-bit string 1 | 000000000101011 | 1010000000000000

represents (−101011.101)2 = −43.625.

The fixed point notation, although not without virtues, is usually inadequate for numerical analysis as it does not allow enough numbers and accuracy.

Floating Point Representation.

FLOATING-POINT arithmetic provides a wide dynamic range, freeing special purpose processor designers from the scaling and overflow/underflow concerns that arise with fixed-point arithmetic. Use of the IEEE-754 standard 32-bit floating-point format also facilitates using the fast Fourier transform (FFT) processors as coprocessors in collaboration with general purpose processors.



- Exponent: excess representation: actual exponent + Bias
  - Ensures exponent is unsigned
  - Single precision: Bias = 127;
  - Double precision: Bias = 1203

Fast Fourier transform (FFT) circuitry consists of several consecutive multipliers and adders over complex numbers; hence an appropriate number representation must be chosen wisely. Most of the FFT architectures have been using fixed-point arithmetic, until recently that FFTs based on floating-point (FP) operations grow. Floating-point arithmetic provides a wide dynamic range, freeing special purpose processor designers from the scaling and

overflow/underflow concerns that arise with fixed-point arithmetic. Use of the IEEE-754 standard 32-bit floating-point format also facilitates using the fast Fourier transform (FFT) processors as coprocessors in collaboration with general purpose processors. This offloads compute-intensive tasks from the processors and leads to higher performance.

The rest of the paper deals with fused dot product unit and blocks of floating point used in the butterfly architecture. In this brief different blocks used in floating point Butterfly Architecture are designed.

II. TRADITIONAL FUSED DOT PRODUCT UNIT

The floating-point dot product unit can be simply implemented by using two floating-point multipliers and a floating-point adder. However, such a discrete version requires large area, power consumption and latency. Moreover, since rounding is performed three times (after each of the multiplications and after the addition), the accuracy is decreased. In order to reduce the area and latency, and increase the accuracy, the floating-point fused dot product unit is introduced. It shares the common logic such as exponent compare, significand addition, exponent adjust and sign logic so that the area, power consumption and latency are reduced. Also, the floating-point fused dot product unit performs only a single rounding so that the accuracy increases. The steps to execute the floating-point fused dot product are

1) Four floating-point numbers are unpacked into their signs, exponents and significands.

2) Two multiplier trees are used to produce two pairs of sums and carries (a total of four numbers). In parallel, two sums of exponents are computed and compared to determine the greater product and the difference is computed. Also, the operation (addition or subtraction) is selected using the sign bits and op code.

3) One sum and carry pair is aligned based on the exponent difference result and inverted if the operation is subtraction. The two pairs of significands are passed to a 4:2 reduction tree. Carry save adders are used to form the reduction tree, which reduces the four significands to two.

4) The two significands are summed and complemented if the sum is negative. The significand comparison result is passed to the sign logic so that the sign is determined.

All the significants in the design of Floating point butterfly are represented in Carry Select Adder (CSA) format and the corresponding carry limited (which avoids carry propagation from one stage to next stage) adder is designed. CSA adder is designed using eight full adders and five 2:1 multiplexers. The block diagram of 4 bit CSA designed is as shown in the figure 1.

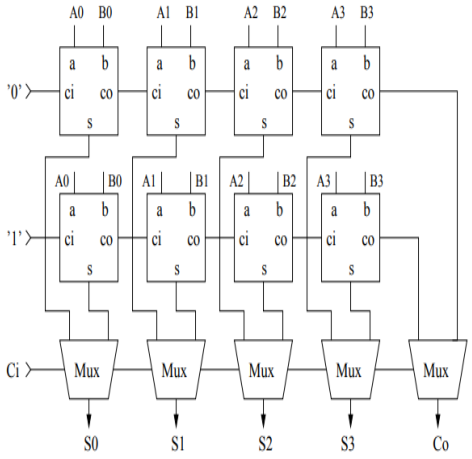


Fig.1. Block Diagram of CSA

Fused dot product unit blocks are designed using the CSA adder in the conventional system and compared with the proposed system.

III. PROPOSED BUTTERFLY ARCHITECTURE

In this section we presents a butterfly architecture using redundant Floating Point arithmetic, which is useful for FP FFT coprocessors and contributes to the digital signal processing applications and also in Orthogonal frequency division multiplexing applications. Although there are other existing algorithms works on the use of redundant Floating point number systems, which are not optimized and which requires more delay. In butterfly architecture in which both redundant FP multiplier and adder are required. The novelties FDPA and BSD adder techniques used in the proposed design include the following.

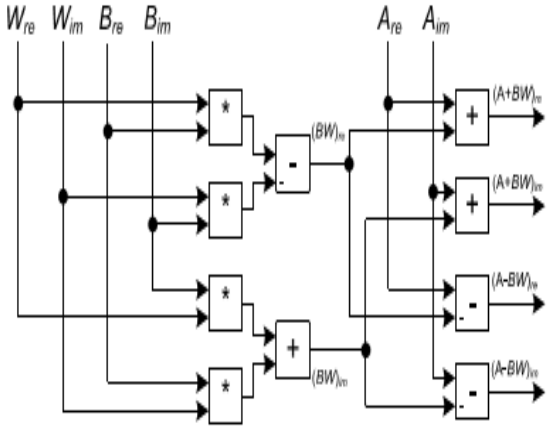


Fig.2.

FFT butterfly architecture

1) All the significants in the design of Floating point butterfly are represented in binary signed digit (BSD) format and the corresponding carry limited (which avoids carry propagation from one stage to next stage) adder is designed.

2) Design of the FP constant multipliers for operands with BSD significant.

The different blocks designed in the project are FP multiplier with BSD adder.

In the FP multiplier Partial products are generated using 2:1 mux.

Proposed Redundant Floating-Point Multiplier:

The proposed multiplier, is similar to that of other parallel multipliers, consists of two major steps, namely, partial product generation (PPG) and Partial product reduction (PPR). However, contrary to the conventional multipliers, the proposed multiplier keeps the product in redundant format and hence there is no need for the final carry-propagating adder. There is no carry propagation from one stage to the next stage. The exponents of the input operands are taken care of in the same way as is done in the conventional FP multipliers, however, normalization, deletion and rounding are left to be done in the next block of the butterfly architecture (i.e., three operand adder).

1.Partial Product Generation: The Partial product generator step of the proposed multiplier is completely different from that of the conventional multiplier one because of the representation of the input operand bits (B, W , Br ,W r).Moreover, given that Wre and Wim are constants, the multiplications (over significant) can be computed through a series of shifters ,adders.

2. Partial Product Reduction: The main advantage of the PPR step is the proposed in this section in which the carry-limited addition over the operands represented in BSD format. This carry-limited addition circuitry is shown in fig.4 (two-digit slice). Since each PP(PPi) is(n+1)-digit(n,...,0)which is either B (n−1,...,0) or 2B(n,...,1),the length of the final product may be more than2n.This is possible by use of BSD adders in the intermediate stages.

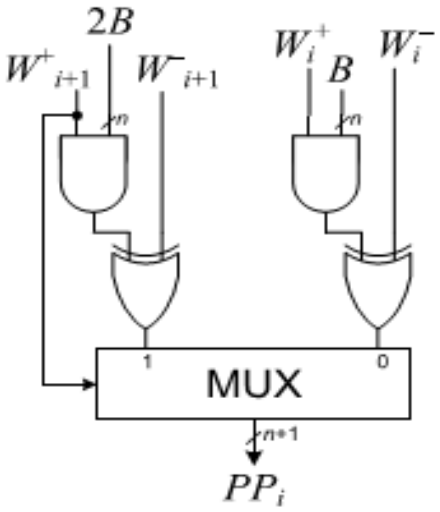


Fig.3: Generation of the ith PP

The generated partial products are then reduced using partial production and then these are added using BSD adder.

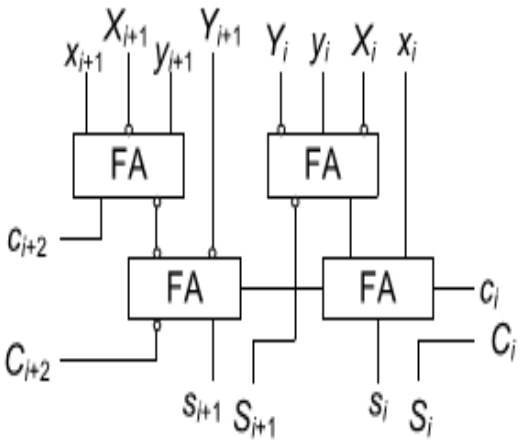


Fig.4: BSD adder (two-digit slice)

Proposed redundant FP multiplier is then added using the BSD adder. Basic block diagram how the BSD adder is used in the floating point addition is as shown in the figure. The blocks used

in this are separately designed and are compared with the existing blocks.

Barrel Shifter.

A barrel shifter is a digital circuit that can shift a data word by a specified number of bits. It can be implemented as a sequence of multiplexers. In this implementation, the output of one MUX is connected to the input of the next MUX in a way that depends on the shift distance. The number of multiplexers required is n*log2(n), for an n bit word. Four common word sizes and the number of multiplexers needed are listed below:

- 64-bit — 64 * log2(64) = 64 * 6 = 384
- 32-bit — 32 * log2(32) = 32 * 5 = 160
- 16-bit — 16 * log2(16) = 16 * 4 = 64
- 8-bit — 8 * log2(8) = 8 * 3 = 24

Basically, a barrel shifter works to shift data by incremental stages which avoids extra clocks to the register and reduces the time spent shifting or rotating data (the specified number of bits are moved/shifted/rotated the desired number of bit positions in a single clock cycle). A barrel shifter is commonly used in computer-intensive applications, such as Digital Signal Processing (DSP), and is useful for most applications that shift data left or right - a normal style for C programming code. Rotation (right) is similar to shifting in that it moves bits to the left. With rotation, however, bits which "fall off" the left side get tacked back on the right side as lower order bits, while in shifting the empty space in the lower order bits after shifting is filled with zeros. Data shifting is required in many key computer operations from address decoding to computer arithmetic. Full barrel shifters are often on the critical path, which has led most research to be directed toward speed optimizations. With the advent of mobile computing, power has become as important as speed for circuit designs. In this project we present a range of 32-bit barrel shifters that vary at the gate, architecture, and environment levels**.**

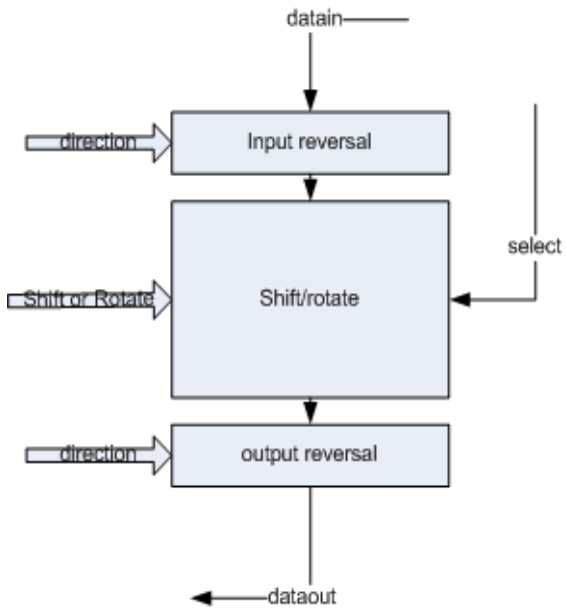


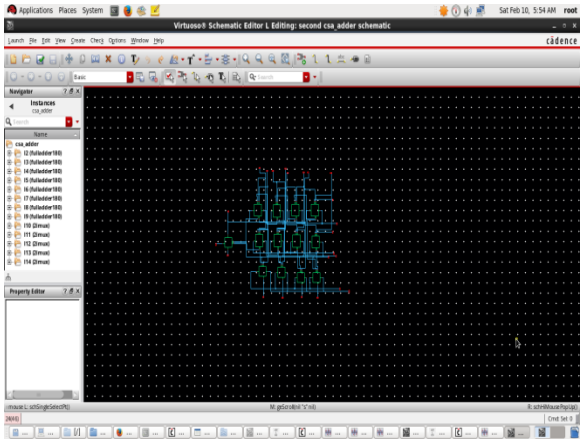
Fig.5 Design specification of circuit

Barrel shifter functionality.

The Barrel shifter component is applicable for cases where an efficient logical shift or rotate with a selectable shift amount is required. The component supports either shift or rotate operations depending on the ROTATION parameter. When the ROTATION parameter is set to 1, the barrel shifter performs rotation and when it is set to 0, a logical shift operation is performed, shifting logical 0 in. the DIRECTION parameter determines if the barrel shifter performs a left or right shift. Setting the DIRECTION parameter to 0 would result in a left shift and setting it to 2 would result in a right shift.

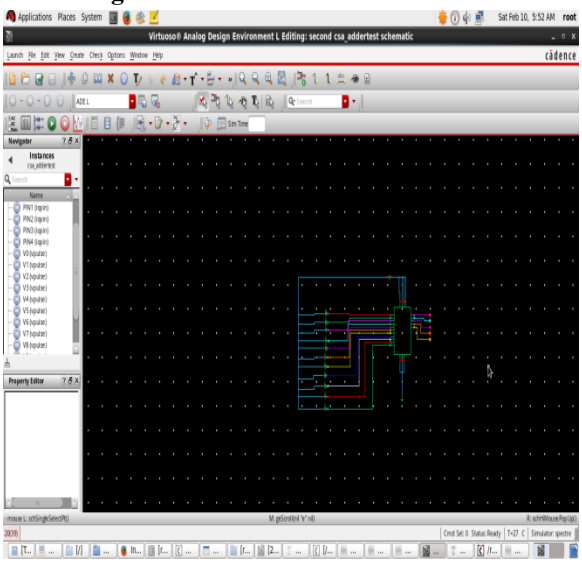
IV. EXPERIMENTAL RESULTS

Circuit Diagram for CSA Adder

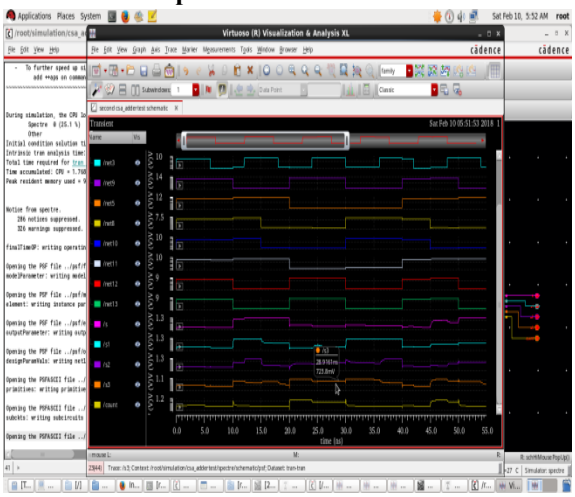


International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 07
March 2018

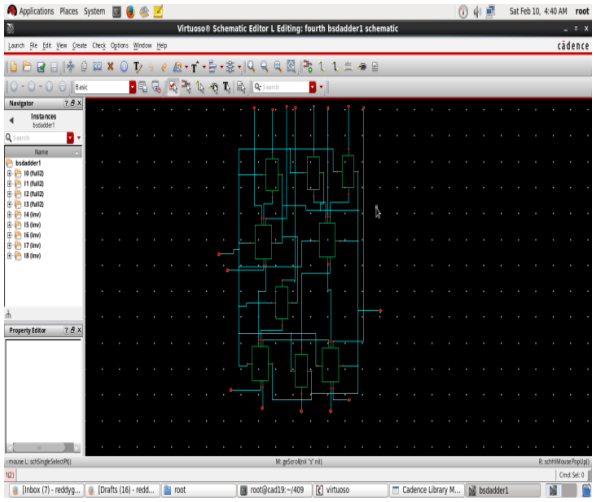
Block Diagram CSA adder



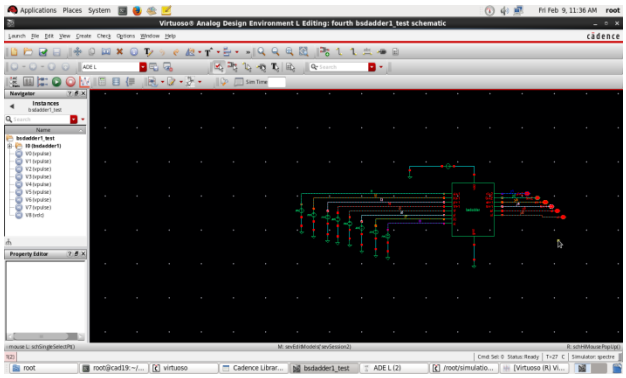
CSA Adder output



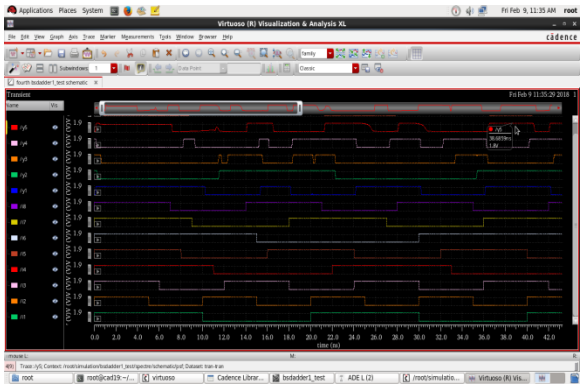
Circuit Diagram For BSD Adder



Block diagram For BSD adder



BSD Adder output



Comparison table.

V. CONCLUSION

Different blocks like FP multiplier and adder used in the floating point Butterfly Architecture are designed. Simulation and Synthesis of the design is observed using Cadence tool. The results of the proposed multiplier with BSD adder is compared with the blocks of FP multiplier when it used CSA adder which show much better performance.

DESIGN	Adders used	Power	Delay
CSA Adder	8 Full Adders and 5 2:1 mux	195.2E-6	10.08E-9
BSD Adder	4 full adders and 5 inverters	367.5E-9	609.9E-12

REFERENCES

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

[2] J. Sohn and E. E. Swartzlander, Jr., "Improved architectures for a floating-point fused dot product unit," in Proc. IEEE 21st Symp. Comput. Arithmetic, Apr. 2013, pp. 41–48.

[3] IEEE Standard for Floating-Point Arithmetic, IEEE Standard 754-2008, Aug. 2008, pp. 1–58.

[4] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, 2nd ed. New York, NY, USA: Oxford Univ. Press, 2010.

[5] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," Math. Comput., vol. 19, no. 90, pp. 297–301, Apr. 1965.

[6] A. F. Tenca, "Multi-operand floating-point addition," in Proc. 19th IEEE Symp. Comput. Arithmetic, Jun. 2009, pp. 161–168.

[7] Y. Tao, G. Deyuan, F. Xiaoya, and R. Xianglong, "Three-operand floating-point adder," in Proc. 12th IEEE Int. Conf. Comput. Inf. Technol., Oct. 2012, pp. 192–196.

[8] A. M. Nielsen, D. W. Matula, C. N. Lyu, and G. Even, "An IEEE compliant floating-point adder that conforms with the pipeline packetforwarding paradigm," IEEE Trans. Comput., vol. 49, no. 1, pp. 33–47, Jan. 2000.

[9] P. Kornerup, "Correcting the normalization shift of redundant binary representations," IEEE Trans. Comput., vol. 58, no. 10, pp. 1435–1439, Oct. 2009. [10] 90 nm CMOS090 Design Platform, STMicroelectronics, Geneva, Switzerland, 2007.

[11] J. H. Min, S.-W. Kim, and E. E. Swartzlander, Jr., "A floating-point fused FFT butterfly arithmetic unit with merged multiple-constant multipliers," in Proc. 45th Asilomar Conf. Signals, Syst. Comput., Nov. 2011, pp. 520–524.

IEEE Standard for Verilog Hardware Description Language, IEEE 1364-1995. [10]. IEEE Standard for Verilog Language, IEEE 1364-2001.

[12]. http://www.xilinx.com/FPGA_series[Online] [13]. Lecture notes - Chapter 7 - Floating Point Arithmetic[Online] http://pages.cs.wisc.edu/~smoler/x86text/lect.notes/arith.flpt.html

[14]. The FFT Demystified, Engineering Productivity Tools Ltd., [Online]: Available: http://www.engineeringproductivitytools.com/

[15]. IEEE 754 Standard - Binary Floating Point Number Calculator Convert a 32 Bit Word to Decimal Value [Online] http://www.ajdesigner.com/fl_ieee_754_word/ieee_32_bit_word.php.