International Journal of Engineering In Advanced Research Science and Technology ISSN: 2278-256

November 2016
VOLUME -2
ISSUE-4
Page:8012-17

# DESIGN RADIX-2 FFT SYSTEM USING AN EFFICIENT AND FUSED FLOATING POINT BUTTERFLY

1P.surekha, 2A.Srinivas Rao, 3B.V.Ramana
1.PG Scholar, Dept of ECE, BonamVenkataChalamayya institute of technology & science, amalapuram.
2Assistant Professor, Dept of ECE, BonamVenkataChalamayya institute of technology & science, amalapuram.
3. Associate Professor, Dept of ECE, BonamVenkataChalamayya institute of technology & science, amalapuram.

**ABSTRACT:**
A new VLSI architecture for real-time pipeline FFT processor is proposed in this project. This concept introduces a software reconfigurable OFDM system using a programmable fused-point DSP. In this project, both radix-2 floating point butterflies are implemented more efficiently with the two fused floating-point operations. The fused operations are a two-term dot product and add-subtract unit along with modified booth encoding algorithm. Both discrete and fused radix processors are implemented; compared in regarded with efficiency wise.

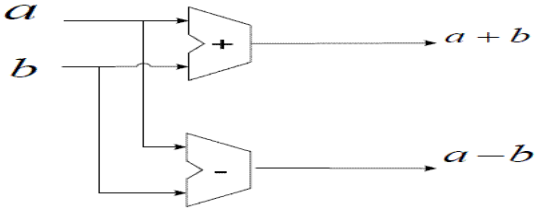
**INTRODUCTION**:
Pipeline FFT processor is a specified class of processors for DFT computation utilizing fast algorithms. It is characterized with real-time, non-stopping processing as the data sequence passing the processor. It is an AT2 with AT2=O(N3), since the area lower bound isO(N). However, as it has been speculated that for real time processing whether a new metric should be introduced since it is necessarily non optimal given the time complexity of O(N). Although asymptotically almost all the feasible architectures have reached the area lower bound, the class of pipeline FFT processors has probably the smallest constant factor Among the approaches that meet the time requirement, due to its least number, O (logN) of Arithmetic Elements (AE).The difference comes from the fact that an AE, especially the multiplier, takes much larger area than a register in digital VLSI implementation.

**IMPLEMENTATION:**
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 paper is concerned with the efficient floating-point implementation of the butterfly units that perform the computations in FFT processors. Since many signal processing applications need high throughput more than low latency, the primary focus of the fused elements is to reduce the circuit area with secondary attention to reducing the delay. Two fused floating-point primitive operations have been developed recently to reduce the delay and area of FFT computation units. The first of the fused operations is a fused floating-point two-term dot product unit (Fused DP). The Fused DP is an extension of the fused multiply-add (FMA) operation that was developed initially for the IBM RS/6000 processor and recently added to IEEE Std-754. The floating-point FMA has several advantages over discrete floating-point adders and multipliers in a general purpose processor. The FMA reduces the latency of a multiplication followed by an addition. Also, a single FMA may be used to replace the floating-point adder and the floating-point multiplier in a system. Some DSP algorithms have been rewritten to take advantage of the presence of FMA units. The second of the fused operations is a fused add subtract unit (Fused AS). In FFT algorithms, both the sum and the difference of a pair of operands are needed frequently. In a fused implementation, the sum and the difference operations can share a substantial amount of operand alignment logic with the benefit of reducing the circuit area.

FUSEDFLOATING-POINTADD-SUBTRACTUNIT
The floating-point fused add-subtract unit (Fused AS) performs an addition and a subtraction in parallel on the same pair of data

$X = A + B$ and

$Y = A - B.$

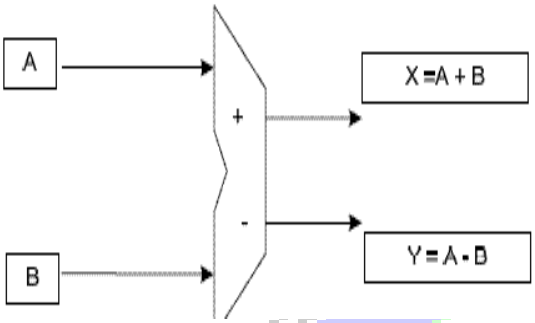



Parallel ADD-SUB unit

International Journal of Engineering In Advanced Research Science and Technology ISSN: 2278-256

November 2016
VOLUME -2
ISSUE-4
Page:8012-17

The fused add-subtract unit is based on a conventional floating-point adder [8]. Although higher speed adder designs are available, the basic design shown here serves to demonstrate the concept. A block diagram of the fused add-subtract unit is shown in below Figure.



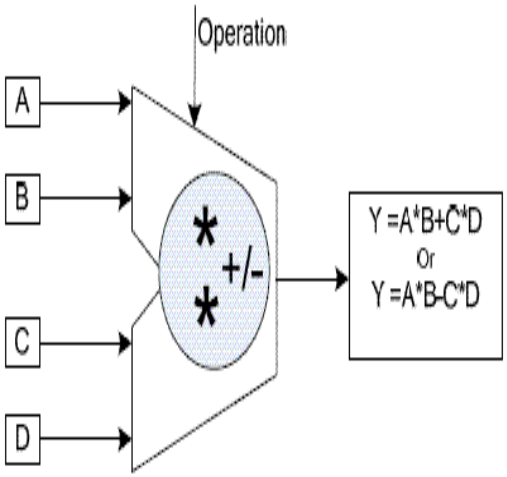

Some details, such as the LZA and normalization logic are omitted here to simplify the figure. The exponent difference calculation, significant swapping, and the significant shifting for both the add and the subtract operations are performed with a single set of hardware and the results are shared by both the operations. This significantly reduces the required circuit area. The significant swapping and shifting is done based solely on the values of the exponents (i.e., without comparing the significants). As a result, if the exponents are equal, the smaller significand may be misidentified as the larger operand.

FUSEDFLOATING-POINTTWO-TERMDOTPRODUCT UNIT

The floating-point two-term fused dot product (Fused DP) unit computes a two-term dot product

$$X = AB \pm CD.$$

Although a conventional dot product adds the two products exist, the Fused DP unit also allows forming the difference of the two products, which is useful in implementing complex multiplication. The Fused dot product unit is shown in below figure. It adds a second multiplier tree, and a revised exponent comparison circuit to the FMA. From the carry save adder onward the FDP and FMA are identical. There is a significant area reduction compared to a conventional parallel discrete implementation of two multipliers and an adder, since the rounding and normalization logic of both of the multipliers are eliminated. Designs of a discrete two-term dot product unit constructed with two floating-point multipliers .



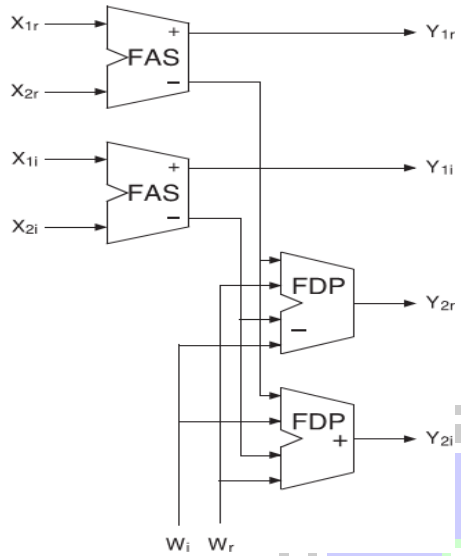

In addition, the Fused DP unit provides slightly more accurate results because only one rounding operation is performed compared to the three rounding operations (one at the output of each multiplier and the other at the output of the adder) of the discrete implementation.

RADIX-2 FFT BUTTERFLY:

To illustrate the utility of the Fused DP and Fused AS units for FFT implementation, FFT butterfly unit designs using both the discrete and the fused units have been made. First, a radix-2 decimation in frequency FFT butterfly was designed. All lines carry complex pairs of 32-bit IEEE-754 numbers and all operations are complex. The complex add, subtract, and multiply operations can be realized with a discrete implementation that uses two real adders to perform the complex add or subtract and four real multipliers and two real adders to perform the complex multiply. The complete butterfly consists of six real adders and four real multipliers as shown on the figure. In this and the following figure, all lines are 32-bits wide for the IEEE-754 single-precision data. The complex add and subtract can be performed with two fused add-subtract units (marked as FAS in the figure) and the complex multiplication can be realized with two fused dot product units (marked as FDP). These results are obtained from having done a complete layout for the butterfly unit. Thus, the areas are greater than the sum of the parts (i.e., four multipliers and six adders for the discrete version or two Fused DP and two Fused AS units for the fused version).

International Journal of Engineering In Advanced Research Science and Technology ISSN: 2278-256

November 2016
VOLUME -2
ISSUE-4
Page:8012-17




The disparity is due to the clock distribution and I/O circuits. Similarly, the delays are less than the sum of the delays of the parts. This is because the conditions that produce the worst-case delay for one part are different from the conditions that produce the worst-case delay for other parts. In comparing the discrete and fused radix-2 butterfly units, the fused version requires about one-third less area and is about15 percent faster than the discrete implementation.

**MODIFIED BOOTH ENCODING:**

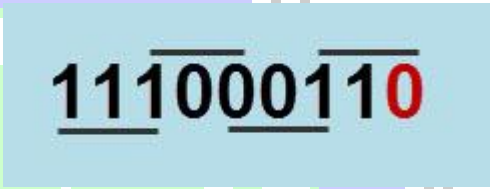
**Modified Booth Algorithm**

Booth multiplication algorithm consists of three major steps as shown in the structure of booth algorithm figure that includes generation of partial product called as recoding, reducing the partial product in two rows, and addition that gives final product. For a better understanding of modified booth algorithm & for multiplication, we must know about each block of booth algorithm for multiplication process.

**Modified Booth Algorithm Encoder**

This modified booth multiplier is used to perform high-speed multiplications using modified booth algorithm. This modified booth multiplier's computation time and the logarithm of the word length of operands are proportional to each other. We can reduce half the number of partial product. Radix-4 booth algorithm used here increases the speed of multiplier and reduces the area of multiplier circuit. In this algorithm, every second column is taken and multiplied by 0 or +1 or +2 or -1 or -2 instead of multiplying with 0 or 1 after shifting and adding of every column of the booth multiplier. Thus, half of the partial product can be reduced using this booth algorithm. Based on the multiplier bits, the process of encoding the multiplicand is performed by radix-4 booth encoder.

The overlapping is used for comparing three bits at a time. This grouping is started from least significant bit (LSB), in which only two bits of the booth multiplier are used by the first block and a zero is assumed as third bit as shown in the figure.




The figure shows the functional operation of the radix-4 booth encoder that consists of eight different types of states. The outcomes or multiplication of multiplicand with 0, -1, and -2 are consecutively obtained during these eight states.

Booth recoding table for radix-4

| Multiplier Bits Block | | | Recoded 1-bit pair | | 2 bit booth | |
|---|---|---|---|---|---|---|
| i+1 | i | i-1 | i+1 | i | Multiplier Value | Partial Product |
| 0 | 0 | 0 | 0 | 0 | 0 | Mx0 |
| 0 | 0 | 1 | 0 | 1 | 1 | Mx1 |
| 0 | 1 | 0 | 1 | -1 | 1 | Mx1 |
| 0 | 1 | 0 | 1 | 0 | 2 | Mx2 |
| 1 | 0 | 0 | -1 | 0 | -2 | Mx-2 |
| 1 | 0 | 1 | -1 | 1 | -1 | Mx-1 |
| 1 | 1 | 0 | 0 | -1 | -1 | Mx-1 |
| 1 | 1 | 0 | 0 | 0 | 0 | Mx0 |

Booth Recoding Table for Radix-4

Modified booth multiplier's (Z) digits can be defined with the following equation:

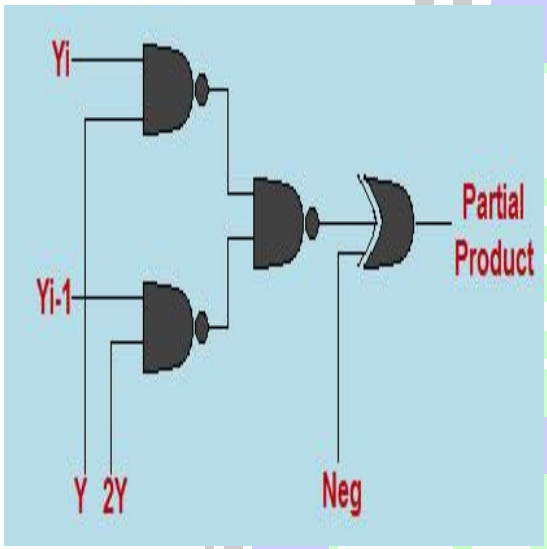
International Journal of Engineering In Advanced Research Science and Technology ISSN: 2278-256

November 2016
VOLUME -2
ISSUE-4
Page:8012-17

Zj = q2j + q2j-1 -2q2j+1 with q-1 = 0

The figure shows the modified booth algorithm encoder circuit. Now, the product of any digit of Z with multiplicand Y may be -2y, -y, 0, y, 2y. But, by performing left shift operation at partial products generation stage, 2y may be generated. By taking 1′s complement to this 2y, negation is done, and then one is added in appropriate 4-2 compressor. One booth encoder shown in the figure generates three output signals by taking three consecutive bit inputs so as to represent all five possibilities -2X, -X, 0, X, 2X.

***Partial Product Generator***

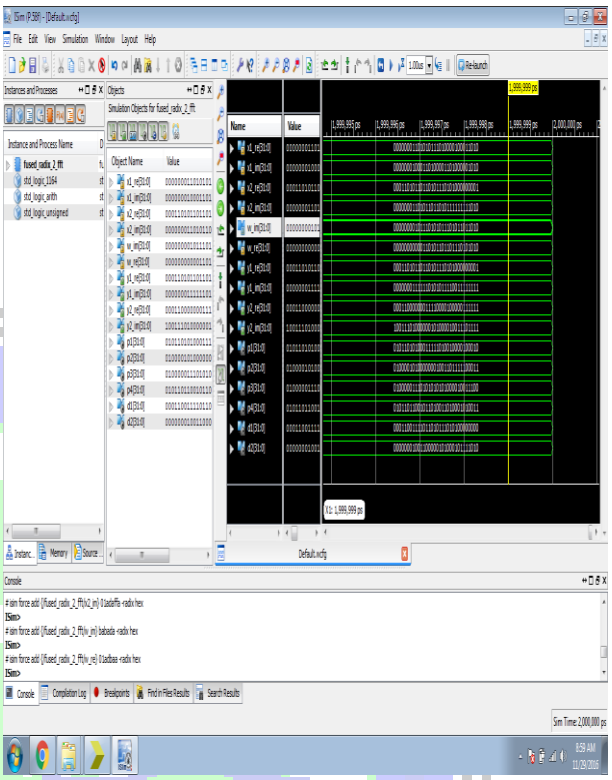



**Partial Product Generator**

If we take the partial product as -2y, -y, 0, y, 2y then, we have to modify the general partial product generator. Now, every partial product point consists of two inputs (consecutive bits) from multiplicand and, based on the requirement, the output will be generated and its complements also generated in case if required. The figure shows the partial product generator circuit.

The 2's complement is taken for negative values of y. There are different types of adders such as conventional adders, ripple-carry adders, carry-look-ahead adders, and carry select adders. The carry select adders (CSLA) and carry-look-ahead adders are considered as fastest adders and are frequently used. The multiplication of y is done by after performing shift operation on y – that is – y is shifted to the left by one bit.

**RESULT:**



| PARAMTER | TIME |
|---|---|
| EXISTING | 100354 ns |
| PROPOSED | 85986 ns |

**CONCLUSION:** Finally, This paper describes the design of two new fused floating-point arithmetic units and their application to the implementation of FFT butterfly operations. Although the fused add-subtract unit is specific to FFT applications, the fused dot product is applicable to a wide variety of signal processing applications. Both the fused dot product unit and the fused add-subtract unit are smaller than parallel implementations constructed with discrete floating-point adders and multipliers. The fused dot product is faster than the conventional implementation, since rounding and normalization is not required as a part of each 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 percent smaller and the latency is 15 percent less than the discrete radix-2 FFT butterfly parallel implementation.

REFERENCES:

1. A.M. Despain. Fourier transform computer using CORDIC iterations. IEEE Trans. Comput., C-23(10):993–1001, Oct. 1974.
2. E. E. Swartzlander, V. K. Jain, and H. Hikawa. A radix 8 wafer scale FFT processor. J. VLSI Signal Processing, 4(2,3):165– 176, May 1992.
3. G. Bi and E. V. Jones. A pipelined FFT processor for word-sequential data.IEEE Trans. Acoust., Speech, Signal Process-ing, 37(12):1982–1985, Dec. 1989.
4. A.M. Despain. Very fast Fourier transform algorithms hard-ware for implementation.IEEE Trans. Comput., C-28(5):333– 341, May 1979.
5. R. Storn. Radix-2 FFT-pipeline architecture with raduced noise-to-signal ratio. IEE Proc.-Vis. Image Signal Process., 141(2):81–86, Apr. 1994
6. D. Takahashi, "A Radix-16 FFT Algorithm Suitable for Multiply-Add Instruction Based on Goedecker Method," Proc. Int'l Conf. Multimedia and Expo,vol. 2, pp. II-845-II-848, July 2003.
7. J.H. McClellan and R.J. Purdy, "Applications of Digital Signal Processing to Radar,"Applications of Digital Signal Processing,A.V. Oppenheim, ed., pp. 239-329, Prentice-Hall, 1978.
8. B. Gold and T. Bially, "Parallelism in Fast Fourier Transform Hardware," IEEE Trans. Audio and Electroacoustics,vol. AU-21, no. 1, pp. 5-16, Feb. 1973.
9. H.H. Saleh and E.E. Swartzlander, Jr., "A Floating-Point Fused Dot-Product Unit,"Proc. IEEE Int'l Conf. Computer Design (ICCD),pp. 427-431, 2008.
10. M.P. Farmwald, "On the Design of High-Performance Digital Arithmetic Units," PhD thesis, Stanford Univ., 1981