A Hybrid and Memory Efficient Multiplier and Accumulator Design Using Radix -4 Algorithm

Sagar Krishna Sivvam, Solomon Gotham

Abstract- ***In this paper we proposed a new architecture for high speed MAC operation. By combining multiplication and addition and devising a hybrid type of Carry save adder, the performance was improved. The proposed CSA uses 1's complement based radix-2 booth algorithm The multiplication and accumulation unit provides high speed multiplication along with accumulative addition. And for final addition some final such as CLA, Kogge stone adder and then adders compare their performance characteristics. The one most effective way to increase the speed of a multiplier is to reduce the number of the partial products. Although the number of partial products can be reduced with a higher radix booth encoder, but the number of hard multiples that are expensive to generate also increases simultaneously. To increase the speed and performance, many parallel MAC architectures have been proposed.***

The design was implemented on Xilinx Xc3s500E fpga and the device utilized 13% of the total LUT's and the total power utilization was 0.041mW.

Index Terms— Radix-4 Booth multiplier, CLA, multiplier and- accumulator (MAC).

I. INTRODUCTION

In this paper, we study the various parallel MAC architectures and then implement a design of parallel MAC based on some booth encodings such as radix-2 booth encoder and some final adders such as CLA, Kogge stone adder and then compare their performance characteristics. In general, a multiplier uses Booth algorithm and an array of full adders, this multiplier mainly consists of three parts Wallace tree, to add partial products, booth encoder and final adder. A Digital multiplier is the fundamental component in general purpose microprocessor and in DSP [1]. Most of the DSP methods use discrete cosine transformations in discrete wavelet transformations. Compared with many other arithmetic operations multiplication is time consuming and power hungry. Thus enhancing the performance and reducing the power dissipation are the most important design challenges for all applications in which multiplier unit dominate the system performance and power dissipation. The one most effective way to increase the speed of a multiplier is to reduce the number of the partial products. Although the number of partial products can be reduced with a higher radix booth encoder, but the number of hard multiples that are expensive to generate also increases simultaneously. To increase the speed and performance, many parallel MAC architectures have been proposed. Parallelism in obtaining partial products is the most common technique used in this architecture. There are two common approaches that make use of parallelism to enhance the multiplication performance. The first one is reducing the number of partial product rows and second one is the carry-save-tree technique to reduce multiple partial product rows as two "carry-save" redundant forms. An architecture was proposed in [2] to provide the tact to merge the final adder block to the accumulator register in the MAC operator to provide the possibility of using two separate N/2-bit adders instead of one N-bit adder to accumulate the N–bit MAC results. The most advanced types of MAC has been proposed by Elguibaly in which accumulation has been combined with the carry save adder (CSA) tree that compresses partial products and thus reduces the critical path. While it has better performance as compared to the previous MAC architectures. Later on a new architecture for a high-speed MAC is proposed by Seo and Kim. The difference between the two is that the latest one carries out the accumulation by feeding back the final CSA output rather than the final adder results.

Revised Manuscript Received on September, 2012.

Sagar Sivam, M.Tech ECE Department,JNTU Kakinada University/ Kaushik College of Engineering/Visakhapatnam, India.

Solomon Gotham,.Professor & Head, Dept. of ECE, Kaushik College of Engineering /visakhapatnam, India.

II. OVERVIEW OF MAC

In this section, basic MAC operation is introduced. A multiplier can be divided into three operational steps.

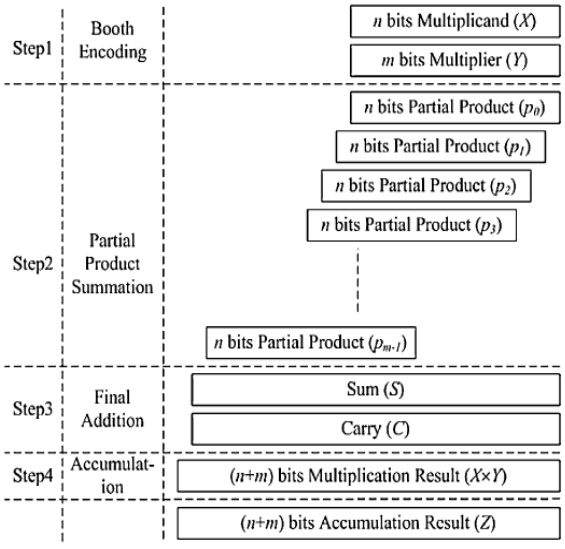

Fig. 1 Basic arithmetic steps of multiplication and accumulation

2.1 Hardware architecture of general MAC:

General hardware architecture of this MAC is shown in Fig. 2. It executes the multiplication operation by multiplying the input multiplier X and the multiplicand Y.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

This is added to the previous multiplication result Z as the accumulation step. The N -bit 2's complement binary number X can be expressed as

$$X = -2^{N-1}x_{N-1} + \sum_{i=0}^{N-2} x_i 2^i, \qquad x_i \in 0, 1.$$

If (1) is expressed in base-4 type redundant sign digit form in order to apply the radix-2 Booth's algorithm, it would be [7].

$$X = \sum_{i=0}^{N/2-1} d_i 4_i$$

$$d_i = -2x_{2i+1} + x_{2i} + x_{2i-1}.$$

If (2) is used, multiplication can be expressed as

$$X \times Y = \sum_{i=0}^{N/2-1} d_i 2^{2i} Y.$$

If these equations are used, the afore-mentioned multiplication–accumulation results can be expressed as

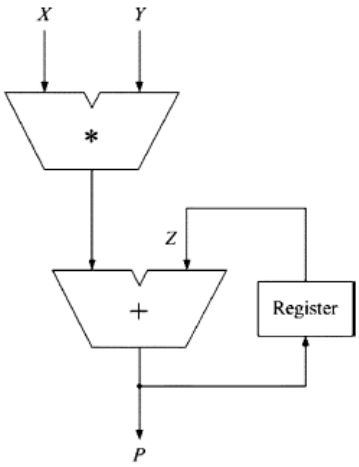

Fig.2. Hardware architecture of general MCA.

$$P = X \times Y + Z = \sum_{i=0}^{N/2-1} d_i 2^i Y + \sum_{j=0}^{2N-1} z_i 2^i.$$

2.2 Booth Encoding:

The effective way to increase the speed of the multiplier is to reduce the number of partial products. For high speed multiplication radix-2 Booth encoding is used in which a partial product is generated from the multiplicand (X) and the multiplier (Y).

2.3 Partial Products Summation:

The second is adder array or partial product compression to add all partial products and convert them into the form of sum and carry.

2.4 Final Addition:

The last is the final addition in which the final multiplication result is produced by adding the sum and the carry.

2.5 Accumulation:

If the process to accumulate the multiplied results is included, a MAC consists of four steps, as shown in Fig. 1, which shows the operational steps explicitly.

III. PROPOSED MAC ARCHITECTRE

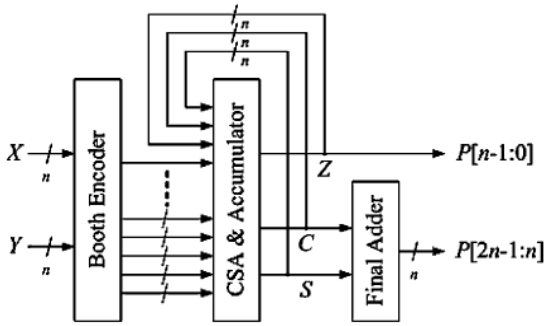

Fig. 4. Hardware architecture of the proposed MCA.

The n-bit MAC inputs, and , are converted into an (n+1) -bit partial product by passing through the Booth encoder. In the CSA and accumulator, accumulation is carried out along with the addition of the partial products. As a result, n-bit S,C, and Z (the result from adding the lower bits of the sum and carry) are generated. These three values are fed back and used for the next accumulation. If the final result for the MAC is needed, P[2n-1:n] is generated by adding S and C in the final adder and combined with that was already generated.

IV. BOOTH"S ALGORITHM

A.D. Booth proposed Booth encoding technique for the reduction of the number of partial products [3]. This algorithm is also called as Radix-2 Booth's Recoding Algorithm. Here the multiplier bits are recoded as Zi for every ith bit Yi with reference to Yi-1 .This is based on the fact that fewer partial products are generated for groups of consecutive zeros and ones. For a group of consecutive zeros in the multiplier there is no need to generate any new partial product. We only need to shift previously accumulated group partial product one bit position to the right for every 0 in the multiplier.

The radix-2 algorithms results in these observations [4]:

(a) Booth observed that whenever there was a large number of consecutive ones, the corresponding additions could be replaced by a single addition and a subtraction 2j + 2j-1 + ……+ 2i+1 + 2i = 2j+1 – 2i

(b) The longer the sequence of ones, the greater the savings.

(c) The effect of this translation is to change a binary number with digit set [0, 1] to a binary signed-digit number with digit set [-1, 1].

In this algorithm the current bit is Yi and the previous bit is Yi-1 of the multiplier Yn-1Yn-2…… Y1 Y0 are examined in order to generate the ith bit Zi of the recoded multiplier Zn-1 Zn-2…….Z1 Z2. The previous bit Yi-1 serves only as the reference bit. The recoding of the multiplier bits need not be done in any predetermined order and can be even done in parallel for all bit positions. The observations obtained from the radix-2 Booth recoding is listed below:

☐ It reduces the number of partial products which in turn reduces the hardware and delay required to sum the partial products.

- ☐ It works well for serial multiplication that can tolerate variable latency operations by reducing the number of serial additions required for the multiplication.
- ☐ The number of serial additions depends on the data (multiplicand).
- ☐ Worst case 8-bit multiplicand requires 8 additions.
- ☐ 01010101 ⇔ 1 -1 1 -1 1 -1 1 -1
- ☐ Parallel systems generally are designed for worst case hardware and latency requirements. Booth-2 algorithm does not significantly reduce the worst case number of partial products.
- ☐ Radix-2 Booth recoding is not directly applied in modern arithmetic circuits; however, it does help in understanding the higher radix versions of Booth's recoding. It doesn't have consecutive 1s or -1s. The disadvantages of the radix-2 Booth algorithm canbe overcome by using Modified Booth algorithm.

V. MODIFIED BOOTH ALGORITHM

Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets – high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation. The common multiplication method is "add and shift" algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, Modified Booth algorithm is one of the most popular algorithms.

It is a powerful algorithm for signed-number multiplication, which treats both positive and negative numbers uniformly. For the standard add-shift operation, each multiplier bit generates one multiple of the multiplicand to be added to the partial product. If the multiplier is very large, then a large number of multiplicands have to be added. In this case the delay of multiplier is determined mainly by the number of additions to be performed. If there is a way to reduce the number of the additions, the performance will get better. Booth algorithm is a method that will reduce the number of multiplicand multiples. For a given range of numbers to be represented, a higher representation radix leads to fewer digits. Since a k-bit binary number can be interpreted as K/2-digit radix-4 number, a K/3-digit radix-8 number, and so on, it can deal with more than one bit of the multiplier in each cycle by using high radix multiplication.

Multiplicand, A = ● ● ● ●

Multiplier, B = (●●)(●●)

Partial product bits ● ● ● ● (B1B0)2 $A4^0$

● ● ● ● (B3B2)2 $A4^1$

Product, P = ● ● ● ● ● ● ● ●

VI. RADIX-4 BOOTH ENCODING

The radix-2 disadvantages can be eliminated by examining three bits of Y at a time rather than two. The modified Booth algorithm is performed with recoded multiplier which multiplies only + a and +2a of the multiplicand, which can be obtained easily by shifting and/or complementation. The truth table for modified Booth recoding is shown below:

TABLE:1

Yi+1	Yi	Yi-1	Zi+1	Zi	Zi/2	Explanation
0	0	0	0	0	0	No string of 1's in sight
0	0	1	0	1	1	End of strings of 1
0	1	0	0	1	1	Isolated 1
0	1	1	1	0	2	End of strings of 1
1	0	0	-1	0	-2	Beginning of strings of 1
1	0	1	-1	1	-1	End a string begin a new 1
1	1	0	0	-1	-1	Beginning of strings of 1
1	1	1	0	0	0	continuation of strings of 1

The following gives the algorithm for performing signand unsigned multiplication operations by using radix-4 Booth recoding.

Algorithm: (for unsigned numbers)

- Pad the LSB with one zero
- Pad the MSB with two zeros if n is even and one zero if n is odd
- Divide the multiplier into overlapping groups of 3-bits
- Determine partial product scale factor from modifiedBooth-2 encoding table
- Compute the multiplicand multiplies
- Sum partial products Algorithm: (for signed numbers)
- Pad the LSB with one zero If n is even don"t pad the MSB (n/2 PP"s)
- Divide the multiplier into overlapping groups of 3-bits
- Determine partial product scale factor from modified Booth-2 encoding table
- Compute the multiplicand multiplies
- Sum partial products Booth recoding is fully parallel and carry free. It can be applied to design a tree and array multiplier, where all the multiples are needed at once. Radix-4 Booth recoding system works perfectly for both signed and unsigned.

VII. PROPOSED CSA ARCHITECTURE

The architecture of the hybrid-type CSA that complies with the operation of the proposed MAC is shown in Fig. 4, which performs 8x 8-bit operation. It was formed based on (12). In Fig. 4, S_i is to simplify the sign expansion and N_i is to compensate 1's complement number into 2's complement number. S[i] and C[i] correspond to the i th bit of the feedback sum and carry. Z_i is the i th bit of the sum of the lower bits for each partial product that were added in advance and Z[i] is the previous result.

In addition, $P_j[i]$ corresponds to the i th bit of the j th partial product. Since the multiplier is for 8 bits, totally four partial products ($P_o[7:0]$~$P_3[7:0]$) are generated from the Booth encoder. In (11), d_0Y and $d_{N/2-1}2^{N-2}Y$ correspond to$P_0[7:0]$ and$P_3[7:0]$, respectively. This CSA requires at least four rows of FAs for the four partial products. Thus, totally five FA rows are necessary since one more level of rows are needed for accumulation. For an -bit MAC operation, the level of CSA is . The white square in Fig. 4 represents an FA and the gray square is a half adder (HA). The rectangular symbol with five inputs is a 2-bit CLA with a carry input. The critical path in this CSA is determined by the 2-bit CLA. It is also possible to use FAs to implement the CSA without CLA. However, if the lower bits of the previously generated partial product are not processed in advance by the CLAs, the number of bits for the final adder will increase. When the entire multiplier or MAC is considered, it degrades the performance. In Table I, the characteristics of the proposed CSA architecture have been summarized and briefly compared with other architectures. For the number system, the proposed CSA uses 1's complement, but ours uses a modified CSA array without sign extension. The biggest difference between ours and the others is the type of values that is fed back for accumulation. Ours has the smallest number of inputs to the final adder. Speed of operation and thus can be used in high performance systems. This work can be utilized in any of the following such as in DSP applications, Numerical co-processor, Calculators (pocket, graphic etc), Filtering, Modulation & Demodulation etc.

As the summation networks and partial product generation logic results in higher delay and most of the area of a MAC. Thus in future, some more techniques and advancement needs to be done which further improves the performance of MAC. Also some measures should be taken which minimize the area consumption.

The multiplier block was implemented in a Xilinx Spartan 3E xc3s200E fpga and the device utilization summary was mentioned in table

Table:2

Logic Utilization	Used	Available	Utilization
Number of 4 input LUTs	533	3,840	13%
Number of occupied Slices	279	1,920	14%
Number of Slices containing only related logic	279	279	100%
Number of Slices containing unrelated logic	0	279	0%
Total Number of 4 input LUTs	533	3,840	13%
Number of bonded IOBs	64	173	36%
Average Fanout of Non-Clock Nets	4.67		

REFERENCES

1. J. J. F. Cavanagh, "Digital Computer Arithmetic", New York: McGraw-Hill, 2004.
2. K. Hwang. Computer Arithmetic – Principles, Architecture and Design. School of Electrical Engineering 1979.
3. Chung Nan Lyu & David W. Matula Redundant Binary Booth Recoding Proceedings of the 12th Symposium on Computer Arithmetic (ARITH '95) 1995 IEEE.
4. B.Cherkauer and E. Friedman. A Hybrid Radix-4/Radix-8 Low Power, High Speed Multiplier Architecture for Wide Bit Widths. In IEEE International Symposium on Circuits and Systems, volume 4, pages 53–56, 1996.
5. Yajuan He and Chip-Hong Chang, "A New redundant binary booth encoding for fast 2n-bit multiplier design", IEEE Transaction on circuit and systems, Vol. 56, No.6, June 2009.
6. A. R. Omondi, Computer Arithmetic Systems . Englewood Cliffs, NJ: Prentice-Hall, 1994.
7. A. D. Booth, "A signed binary multiplication technique," Quart. J. Math., vol. IV, pp. 236–240, 1952.
8. C. S. Wallace, "A suggestion for a fast multiplier,"IEEE Trans. Electron Comput. , vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
9. A. R. Cooper, "Parallel architecture modified Booth multiplier," Proc.Inst. Electr. Eng. G , vol. 135, pp. 125–128, 1988.
10. N. R. Shanbag and P. Juneja, "Parallel implementation of a 4 4-bit multiplier using modified Booth's algorithm,"IEEE J. Solid-State Circuits , vol. 23, no. 4, pp. 1010–1013, Aug. 1988.
11. G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54 54 regularstructured tree multiplier," IEEE J. Solid-State Circuits, vol. 27, no. 9,pp. 1229–1236, Sep. 1992.
12. J. Fadavi-Ardekani, "M N Booth encoded multiplier generator using optimized Wallace trees,"IEEE Trans. Very Large Scale Integr. (VLSI) Syst. , vol. 1, no. 2, pp. 120–125, Jun. 1993.
13. N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki, and Y. Nakagome, "A 4.4 ns CMOS 54 54 multiplier using pass-transistor multiplexer," IEEE J. Solid-State Circuits, vol. 30, no. 3, pp. 251–257, Mar. 1995.
14. A. Tawfik, F. Elguibaly, and P. Agathoklis, "New realization and implementation of fixed-point IIR digital filters," J. Circuits, Syst., Comput., vol. 7, no. 3, pp. 191–209, 1997.
15. A. Tawfik, F. Elguibaly, M. N. Fahmi, E. Abdel-Raheem, and P. Agathoklis, "High-speed area-efficient inner-product processor," Can. J. Electr. Comput. Eng. , vol. 19, pp. 187–191, 1994.
16. F. Elguibaly and A. Rayhan, "Overflow handling in inner-product processors," in Proc. IEEE Pacific Rim Conf. Commun., Comput., Signal Process., Aug. 1997, pp. 117–120. IEEE J. Solid-State Circuits, vol. 25, no. 2, pp. 584–594, Feb. 1990.