Parallel MAC Based On Radix-4 & Radix-8 Booth Encodings

SHANKEY GOEL
Student, Department of Electronics & Communication Engineering
National Institute of Technology, Kurukshetra
er.shankey@gmail.com

R.K. SHARMA
Faculty, Department of Electronics & Communication Engineering
National Institute of Technology, Kurukshetra
mail2drrks@gmail.com

ABSTRACT- Parallel MAC is frequently used in digital signal processing and video/graphics applications. The MAC provides high speed multiplication and multiplication with accumulative addition. This paper presents a combined process of multiplication and accumulation based on radix-4 & radix-8 booth encodings. In this Paper, we investigate the method of implementing the Parallel MAC with the smallest possible delay. Enhancing the speed of operation of the parallel MAC is a major design issue. This has been achieved by developing a new Radix-5 Kogge stone adder for parallel MAC.

Keywords: Radix-4 and Radix-8 based Booth multiplier; carry save adder (CSA) tree; computer arithmetic; digital signal processing (DSP); multiplier and- accumulator (MAC); Adders.

1. 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-4 & radix-8 booth encoder and some final adders such as CLA, Kogge stone adder and then compare their performance characteristics. A Digital multiplier is the fundamental component in general purpose microprocessor and in DSP [1], [2]. 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 the above implemented 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 [4] 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 [8] in which accumulation has been combined with the carry save adder (CSA) tree that compresses partial products and thus reduces the critical path. Later on a new architecture for a high-speed MAC is proposed by Seo and Kim [9]. 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.

The rest of the paper is organized as follows. In section second, an introduction to the general MAC is given along with basic MAC algorithms. In section third, entire process of parallel MAC based on radix-4 and radix-8 booth encodings [7] is explained. Section four shows implementation result and the characteristics of parallel MAC based on both of the booth encodings. Finally, the conclusion will be given in section five in which we provide summary of our proposed approach and discuss scope of future extensions.

2. GENERAL MAC STRUCTURE

In this section, we discuss basic MAC operation. Basically, multiplier operation can be divided into three operational steps. The first one is booth encoding to generate the partial products. The second one is adder array or partial product compression and the last one is final addition in which final multiplication result is produced.

If the multiplication process is extended to accumulate the multiplied result, then MAC consists of four steps. General hardware architecture for MAC is shown in Figure 1. It executes the multiplication operation by multiplying input multiplier X and input multiplicand Y. After that current multiplication result is added to the previous multiplication result Z as accumulation step.

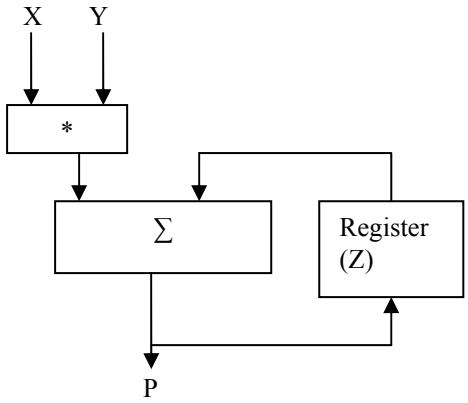

Fig. 1. Hardware architecture of general MAC

The N-bit 2's complement binary number X can be expressed as

$$Y = -2^{N-1} y_{N-1} + \sum_{i=0}^{N-2} y_i 2^i \quad (1)$$

If "Eq. (1)" is expressed in base-8 type redundant sign digit form in order to apply the radix-8 booth's algorithm, it would be

$$Y = \sum_{i=0}^{((N+1/3)-2)} d_i 8^i \quad (2)$$

$$di \in \{\pm 4, \pm 3, \pm 2, \pm 1, 0\} \quad (3)$$

If "Eq. (1)" is expressed in base-4 type redundant sign digit form in order to apply the radix-4 booth's algorithm, it would be

$$Y = \sum_{i=0}^{(N/2-1)} d_i 4^i \quad (4)$$

$$di \in \{\pm 2, \pm 1, 0\} \quad (5)$$

Here d_i refers to the select signals for the partial product. If "Eq. (2)" is used, then multiplication can be expressed as

$$X \times Y = \sum_{i=0}^{((N+1/3)-2)} d_i 2^{3i} Y \quad (6)$$

If these equations are used, then multiplication accumulation result can be expressed as

$$P = X \times Y + Z = \sum_{i=0}^{((N+1/3)-2)} d_i 2^{3i} X + \sum_{j=0}^{2N-1} z_i 2^i \quad (7)$$

Here the MAC architecture implemented by "Eq. (7)" is called standard design.

3. MAC ARCHITECTURE

Now in this section we will derive the expression for new MAC By using the standard design, and also with the help of above mentioned results, VLSI architecture for the new MAC will be proposed.

3.1 DERIVATION OF MAC ARITHMETIC

If an operation of multiplication of two N-bit numbers and accumulating them into a 2N-bit number is considered, then the critical path is totally determined by accumulation operation. If pipeline scheme is applied for each step, then the delay of last accumulator must be reduced by combining the accumulation operation with the CSA function. The aforementioned concept is applied to "Eq. (7)" to express the proposed MAC arithmetic. First "Eq. (6)" is decomposed and rearranged, it becomes

$$X \times Y = d_0 2X + d_1 2^3 X + d_2 2^6 X + \ldots\ldots\ldots + d_{((N+1/3)-1)} 2^{N-2} X \quad (8)$$

"Eq. (8)" can be re-expressed into equation (9) by dividing it into first partial product, sum of the middle partial products and the last partial product. Thus

$$X \times Y = \left(d_0 2X + \sum_{i=1}^{((N+1/3)-2)} d_i 2^{3i} X + d_{((N+1/3)-1)} 2^{N-2} X \right) \quad (9)$$

Now "Eq. (9)" is applied "Eq. (7)" in which Z is divided into upper and lower bits and rearranged "Eq. (10)" and "Eq. (11)" will be derived.

$$Z = \sum_{i=0}^{N-1} z_i 2^i + \sum_{i=N}^{2N-1} z_i 2^i \quad (10)$$

The second term can be further separated into sum and carry terms, thus we get

$$\sum_{i=N}^{2N-1} z_i 2^i = \sum_{i=0}^{N-1} z_{N+i} 2^i 2^N = \sum_{i=0}^{N-2} (c_i + s_i) 2^i 2^N \quad (11)$$

If equations "Eq. (9)" and "Eq. (11)" are used, then the MAC arithmetic in "Eq. (7)" can be expressed as

$$P = \left(d_0 2X + \sum_{i=1}^{((N+1/3)-2)} d_i 2^{3i} X + d_{((N+1/3)-1)} 2^{N-2} X \right) + \left(\sum_{i=0}^{N-1} z_i 2^i + \sum_{i=0}^{N-2} s_i 2^i 2^N + \sum_{i=0}^{N-2} c_i 2^i 2^N \right) \quad (12)$$

If each term of "Eq. (12)" is matched to its bit position and rearranged, it can be expressed as "Eq. (13)", which is the ultimate equation for the proposed MAC. Thus, we get

$$P = \left(d_0 2X + \sum_{i=0}^{N-1} z_i 2^i \right) + \left(\sum_{i=1}^{((N+1/3)-2)} d_i 2^{3i} X + \sum_{i=0}^{N-2} c_i 2^i 2^N \right) + \left(d_{((N+1/3)-1)} 2^{N-2} X + \sum_{i=0}^{N-2} s_i 2^i 2^N \right) \quad (13)$$

3.2 MAC ARCHITECTURE

The hardware architecture of the proposed MAC satisfying the aforementioned equations is shown in Figure 2. The n-bit MAC inputs, X and Y, are converted into group of partial products by passing through the booth encoders based on radix-4 and radix-8 booth algorithms.

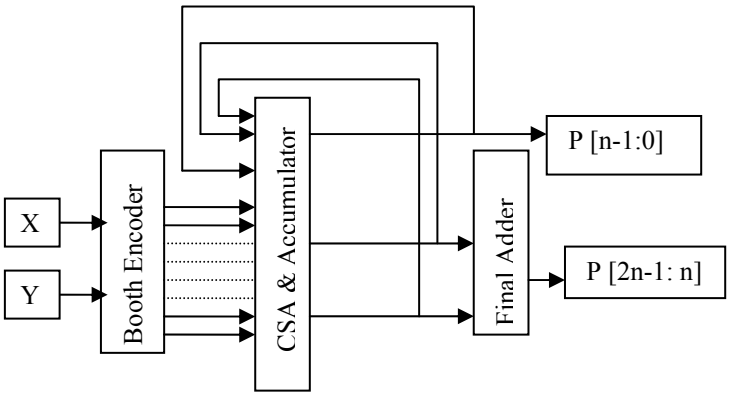

Fig. 2. Hardware architecture of proposed MAC

After that accumulation is carried out along with addition of partial products in CSA and Accumulator. As a result n-bit S, C and, Z are generated. These three values are fed back and used for accumulation operation. P [2n-1: n] is generated by adding S and C in the final adder which would be either CLA (carry look ahead adder) or RCA (ripple carry adder) or Kogge stone adder. At last, by combining P [2n-1: n] with P [n-1: 0] we get the final result.

3.3 BOOTH ENCODERS

The modified Booth's algorithm based on a radix-4, generally called Booth-2 [7] is the most popular approach for implementing fast multipliers using parallel encoding [1]. It uses a digit set {0, ±1, ±2} to reduce the number of the partial products to n'= [(n+1)/ 2]. Radix- 4 encoding start by appending a zero to the right of multiplier LSB. Triplets are taken beginning at position x_{-1} and continuing to the MSB with one bit overlapping between adjacent triplets.

Table 1. Radix-4 booth encoding

Y_{n+1}	Y_n	Y_{n-1}	Z_n	Partial Product
0	0	0	0	0
0	0	1	1	1× Multiplicand
0	1	0	1	1× Multiplicand
0	1	1	2	2× Multiplicand
1	0	0	-2	-2× Multiplicand
1	0	1	-1	1× Multiplicand
1	1	0	-1	1× Multiplicand
1	1	1	0	0

This recoding scheme applied to a parallel multiplier halves the number of partial products so the multiplication time and the hardware requirements decrease. Radix-8 recoding [5], [7] applies the same algorithm as radix-4, but now in this we take quartets of bits instead of triplets. The Booth-3 scheme is based on a radix-8 encoding to reduce this number to n' = [(n+1)/3]. All digit sets {0, ±1, ±2, ±3, ±4} are obtained by simple shifting and complementary operations, except generation of the multiple 3X, which is computed by an adding and shifting operation, 3X = 2X+X and -3X can be generated by complement 3X. Radix-4 booth encoding that generates partial products is shown in Table 1 respectively.

3.4 PROPOSED CSA AND ACCUMULATOR ARCHITECTURE

The architecture of CSA and accumulator based on radix-4 modified booth encoder is shown in Figure 3 which performs 8×8 bit MAC operation. In Figure 3, N_i is to compensate the 1′s complement number into 2′s complement, S_i is to specify the sign expansion of the corresponding partial product and ($\sim S_i$) is 1′s complement of S_i. S[i] and C[i] corresponds to the *i*th bit of the feedback sum and carry whereas Z[i] is the *i*th bit of the sum of lower bits for each partial product. S′[i], C′[i] and Z′[i] are the previous result for accumulation. In addition, P_j[i] is the *i*th bit of *j*th partial product.

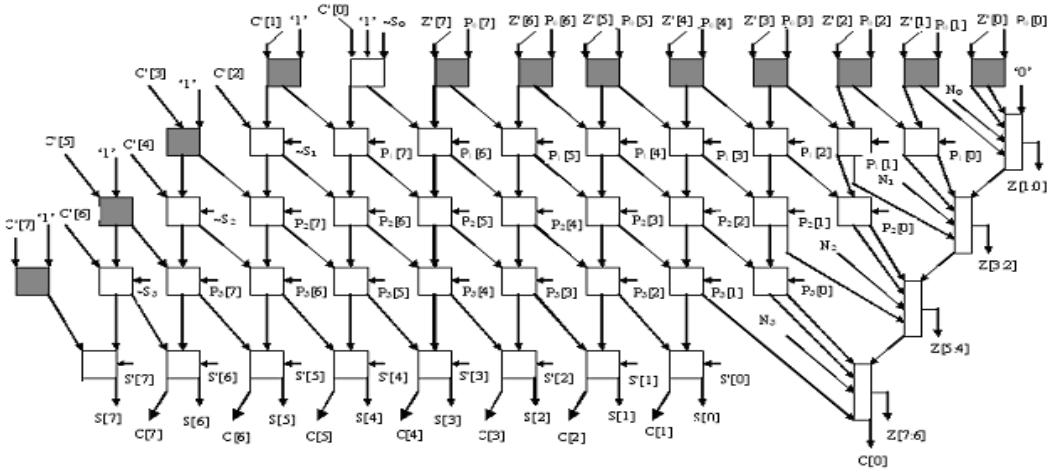

Fig. 3. CSA architecture for 8-bit MAC based on radix-4 booth encoder

Proposed CSA consists of full adders, half adders and carry look ahead adders as shown in Figure 3. In this, series of CLA adders are used to generate the lower 8-bits of the final result. The white square in Figure 3 represents an FA and the black square is a half adder (HA). The rectangular symbol with five inputs is a 2-bit CLA with a carry input.

3.5 FINAL ADDER

Final adders are digital circuits that compute the addition of variable binary strings of equivalent or different size. Two types of adder are used for the addition of upper sum and carry bits. These are CLA and radix-4 Kogge stone adder (KSA) [10]. In carry look-ahead adder, carry signal is calculated in advance based on input signals. The result is a reduced carry propagation time. The second adder i.e. radix-5 Kogge stone adder is a parallel prefix form carry look ahead adder. In KSA, carries are computed fast by computing them in parallel at the cost of increased area but reduced delay. A Parallel Prefix Adder (PPA) is equivalent to the CLA adder. The two differ in the way their carry generation block is implemented. Rdaix-5 Kogge stone adder structure and their symbol representation are shown in figure 4 and 5.

When large numbers of groups are combined then that in valency 5 group logic the equations for PG logic are:

$$P_{i:j} = P_{i:k} \bullet P_{k-1:l} \bullet P_{l-1:m} \bullet P_{m-1:n} \bullet P_{n-1:j} \tag{14}$$

$$G_{i:j} = G_{i:k} \text{ or } P_{i:k}\,(G_{k\text{-}l:l} \text{ or } P_{k\text{-}l:l}\,(G_{l\text{-}1:m} \text{ or } P_{l\text{-}1:m}\,(G_{m\text{-}1:n} \text{ or } P_{m\text{-}1:n} \text{ and } G_{n\text{-}1:j}))) \quad \text{where } (i > k > l > m > n > j) \tag{15}$$

The sum can therefore be computes using the following equation:

$$S_i = P_i \oplus G_{i-1:0} \tag{16}$$

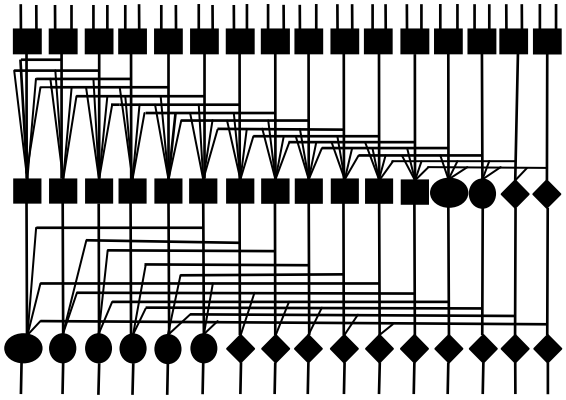

Fig. 4. Radix-5 Kogge stone adder structure

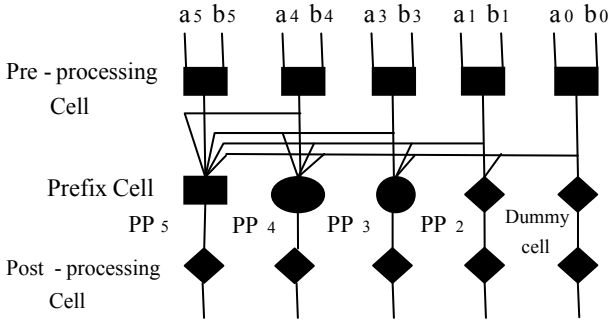

Fig. 5. Symbols used in Radix-5 Kogge stone adder

4. EXPERIMENTAL RESULT

The 16×16 bit parallel MAC based on both the booth encodings (i.e. Radix-4 booth encoding and Radix-8 booth encoding) and some final adders (such as CLA adder and Radix-5 Kogge stone adder) is designed in Verilog and the functionalities of the algorithms are verified by XILINX ISE 8.2i using Virtex 2p with XC2VP7 family with FF896 package & -7 speed grade [11].

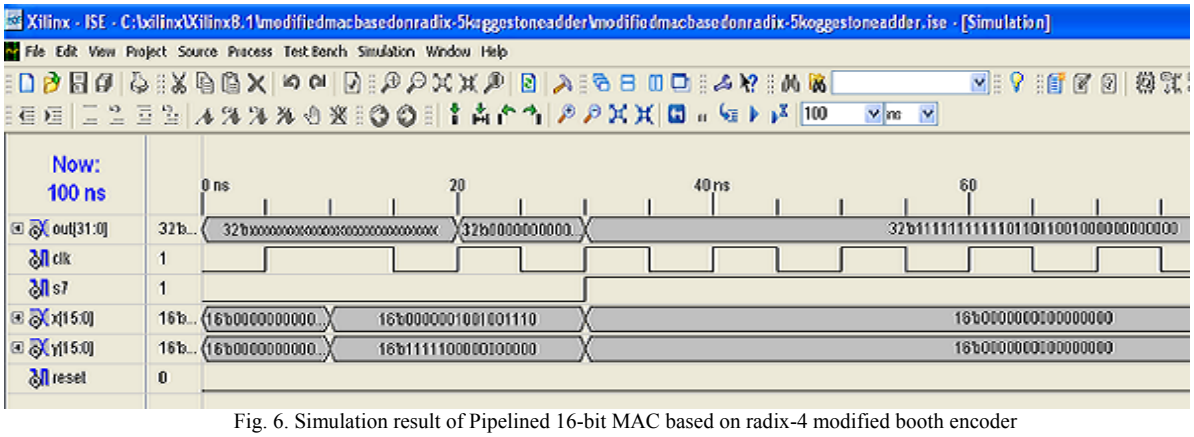

Fig. 6. Simulation result of Pipelined 16-bit MAC based on radix-4 modified booth encoder

Simulation result for the 16-bit parallel MAC based on radix-4 booth encoder and using Radix-5 Kogge stone as a final adder is shown in Figure 6 respectively and performance characteristics in terms of speed and area are shown in table I. It could be seen from table that the parallel MAC based on Radix-5 Kogge stone adder is having more area as compared to others but having less delay. Higher would be the no. of slices, higher would be area. Thus, MAC based on Radix-5 Kogge stone adder shows higher performance than others but at the cost of area.

Table 3. Comparison result of 16-bit parallel MAC

Performance Parameters and their comparison	MAC using Radix-4 Booth encoding & adder		MAC using Radix-8 Booth encoding & adder	
	CLA adder	Radix-5 Kogge stone adder	CLA adder	Radix-5 Kogge stone adder
Max. Operating Frequency(MHz)	111.495	**126.231**	100.553	**112.04**
Delay (ns)	5.112	**4.264**	7.990	**6.789**
Number of occupied Slices (4,928)	**404 (8%)**	423 (8%)	**784 (15%)**	814 (16%)
Total equivalent gate count	**5,868**	6,096	**10,938**	11,223

5. CONCLUSION

The primary objective of this thesis has been to present a new type of Parallel MAC using Radix-5 Kogge stone adder, to reduce the implementation to practice, and to show through simulation and design that this algorithm is competitive with other more commonly used algorithms when used for high performance implementations. Secondarily, this thesis has shown that algorithms based upon the Radix-4 Booth partial product method are distinctly superior in performance when compared to Radix-8 Booth encoded method. From above thesis, we conclude that Parallel MAC based on Radix-4 booth encoder and using Radix-5 Kogge stone adder has higher 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.

REFERENCES

[1] J. J. F. Cavanagh, "Digital Computer Arithmetic", New York: McGraw- Hill, 1984.
[2] O. L. MacSorley, "High speed arithmetic in binary computers," Proc. IRE, vol. 49, pp. 67–91, Jan. 1961.
[3] C. S. Wallace, "A suggestion for a fast multiplier", IEEE Trans. Electron Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
[4] Fayed and M. Bayoumi, "A merged multiplier-accumulator for high speed signal processing application", Proc. ICASSP, vol. 3, pp. 3212–3215, 2002.
[5] Yajuan He and Chip-Hong Chang, "A New redundant binary booth encoding for fast 2^n-bit multiplier design", IEEE Transaction on circuit and systems, Vol. 56, No.6, June 2009.
[6] R. Cooper, "Parallel architecture modified Booth multiplier", Proc. Inst. Electr. Eng. G, vol. 135, pp. 125–128, 1988.
[7] Marc Hunger and Daniel Marienfeld, "New self checking booth multiplier", Int. J. Appl. Math, Comput. Sci., Vol.18, No.3, 319–328, 2008.
[8] F. Elguibaly, "A fast parallel multiplier–accumulator using the modified Booth algorithm", IEEE Trans. Circuits Syst., vol. 27, no. 9, pp. 902–908, Sep. 2000.
[9] Young-Ho Seo and Dong-Wook Kim, "A New VLSI Architecture of Parallel Multiplier Accumulator Based on Radix-2 Modified Booth Algorithm", IEEE trans. on VLSI Systems, Vol.18 No. 2, Feb. 2010. R.
[10] E. Ladner and M. J. Fischer, "Parallel Prefix Computation", Journal of the ACM, vol.27, No.4, October 1980, 831-838.
[11] Virtex generation configuration user guide, Virtex-II Pro™ FPGA families, DS083-1 (v1.0) January 31, 2002, available at www.xilinx.com.