RESEARCH ARTICLE OPEN ACCESS

DESIGN AND ANALYSIS OF RADIX-16 BOOTH PARTIAL PRODUCT GENERATOR FOR 64-BIT BINARY MULTIPLIERS

K.Deepthi[1], Dr.T.Lalith Kumar[2]

[1]PG Scholar,Dept. Of ECE,Annamacharya Institute Of Technology And Sciences , Kadapa, AP, India.
[2] Associate professor, Annamacharya Institute Of Technology And Sciences ,Kadapa, AP ,India.

Abstract:

Redundant Binary Partial Product Generator technique are used to reduce by one row the maximum height of the partial product array generated by a radix16 Modified Booth Encoded multiplier, without any raise in the delay of the partial product creation Block. In this paper, we describe an optimization for binary radix-16 (modified) Booth recoded multipliers to reduce the maximum height of the partial product columns to [n/4] for n = 64-bit unsigned operands. This is in contrast to the conventional maximum height of [(n + 1)/4]. Therefore, a reduction of one unit in the maximum height is achieved. This Arithmetic multipliers increase the performance of ALU and Processors. We evaluate the proposed approach by comparison with Normal Booth Multiplier. Logic synthesis showed its efficiency in terms of delay and power consumption when the word length of each operand in the multiplier is 64bits.

Index Terms: Compound adder,Recoders ,Multiplexers ,Carry save adder ,Carry propogateadder.

I. INTRODUCTION

Digital multipliers are widely used in arithmetic units of microprocessors, multimedia and digital signal processors. Many algorithms and architectures have been proposed to design high-speed and low power multipliers. A normal binary (NB) multiplication by digital circuits includes three steps. In the first step, partial products are generated; in the second step, all partial products are added by a partial product reduction tree until two partial product rows remain. In the third step, the two partial product rows are added by a fast carry propagation adder. Two methods have been used to perform the second step for the partial product reduction. A first method uses 4-2 compressors, while a second method uses redundant binary (RB) numbers. Both methods allow the partial product reduction tree to be reduced at a rate of 2:1.

The redundant binary number representation has been introduced by Avizienis to perform signeddigit arithmetic; the RB number has the capability to be represented in different ways.

II. EXISTING METHOD

The architecture of the basic radix-16 Booth multiplier is shown in Fig.1. For sake of simplicity, but without loss of generality, we consider unsigned operands with n = 64. Let us denote with X the multiplicand operand with bit components xi (i = 0 to n − 1, with the least-significant bit, LSB, at position 0) and with Y the multiplier operand and bit components yi.

The first step is the recoding of the multiplier operand: groups of four bits with relative values in the set {0, 1,……,14, 15} are recoded to digits in the set {−8,−7,.. ……..,0,…….,7, 8}. This recoding is done with the help of a transfer digit ti and an interim digit wi.

The recoded digit zi is the sum of the interim and transfer digits zi = wi + ti. That is

$$0 \le vi < 8 : ti+1 = 0 \; wi = vi \; wi \in [0, 7]$$
$$8 \le vi \le 15 : ti+1 = 1 \; wi = -(16 - vi)$$
$$wi \in [-8,-1].$$

The transfer digit corresponds to the most-significant bit (MSB) of the four-bit group, since this bit determines if the radix-16 digit

is greater than or equal to 8. The final logical step is to add the interim digits and the transfer digits (0 or 1) from the radix-16 digit position to the right. Since the transfer digit is either 1 or 0, the addition of the interim digit and the transfer digit results in a final digit in the set {−8,−7, . . . , 0, . . . , 7, 8}. After recoding, the partial products are generated by digit multiplication of the recoded digits times the multiplicand X.

For the set of digits {−8,−7, . . . , 0, . . . , 7, 8}, the multiples 1X, 2X, 4X, and 8X are easy to compute, since they are obtained by simple logic shifts. The negative versions of these multiples are obtained by bit inversion and addition of a 1 in the corresponding position in the bit array of the partial products. The generation of 3X, 5X, and 7X (odd multiples) requires carry-propagate adders (the negative versions of these multiples are obtained as before). Finally, 6X is obtained by a simple one bit left shift of 3X.

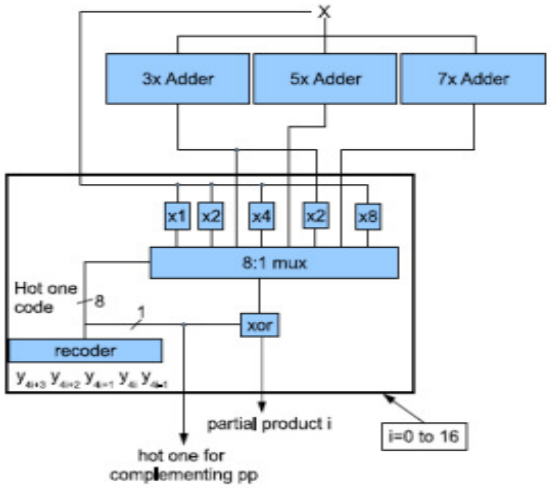


Fig. 1: Partial product generation.

After the generation of the partial product bit array, the reduction (multi operand addition) from a maximum height of 17 (for n = 64) to 2 is performed. A maximum height of 17 leads to different approaches that may increase the delay and/or require to use arrays of 3:2 carry-save adders interconnected to minimize delay [20]. After the reduction to two operands, a carry-propagate addition is performed. This addition may take advantage of the specific signal arrival times from the partial product reduction step.

III. PROPOSED METHOD

To reduce the maximum height of the partial product bit array we perform a short carry-propagate addition in parallel to the regular partial product generation. This short addition reduces the maximum height by one row and it is faster than the regular partial product generation. Fig. 2(b) shows the elements of the bit array to be added by the short adder. Fig. 2(c) shows the resulting partial product bit array after the short addition. Comparing both figures, we observe that the maximum height is reduced from 17 to 16 for n = 64.

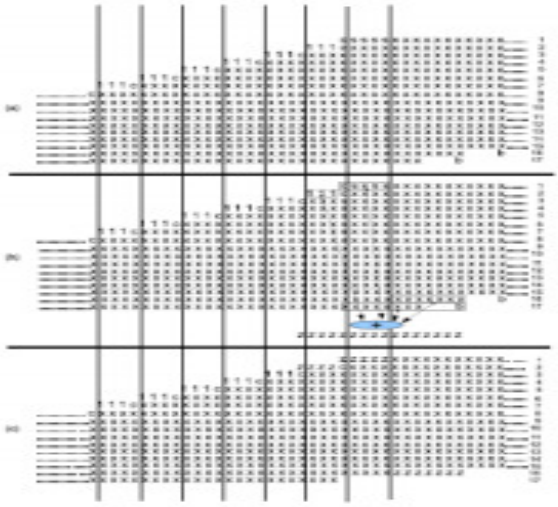


Fig. 2: Radix-16 partial product reduction array.

Since the partial products are left-shifted four bit positions with respect to each other, a costly sign extension would be necessary. However, the sign extension is simplified by concatenation of some bits to each partial

product (S is the sign bit of the partial product and C is S complemented): CSSS for the first partial product and 111C for the rest of partial products (except the partial product at the bottom that is non- negative since the corresponding multiplier digit is 0 or 1). The bits denoted by b in Fig. 2 corresponds to the logic 1 that is added for the two's complement for negative partial products.

We perform the computation in two concurrent parts A and B as indicated in Fig. 3. The elements of the partA are generated faster than the elements of part B. Specifically the elements of part A are obtained from the sign of the first partial product: this is directly obtained from bit y3 since there is no transfer digit from a previous radix-16 digit;

Therefore, we decided to implement part A as a speculative addition, by computing two results, a result with carry-in = 0 and a result with carry-in = 1. This can be computed efficiently with a compound adder.

Fig. 3 shows the implementation of part A. The compound adder determines speculatively the two possible results. Once the carry-in is obtained (from part B), the correct result is selected by a multiplexer. Note that the compound adder is of only five bits, since the propagation of the carry through the most significant three ones is straightforward. The computation of part B is more complicated. The main issue is that we need the 7 least-significant bits of partial product 15. Of course waiting for the generation of partial product 15 is not an option since we want to hide the short addition delay out of the critical path.

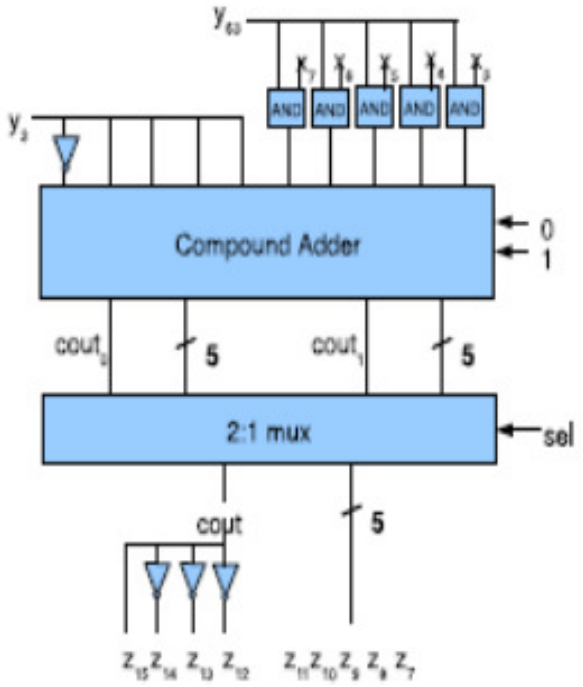


Fig. 3: Speculative addition of part A.

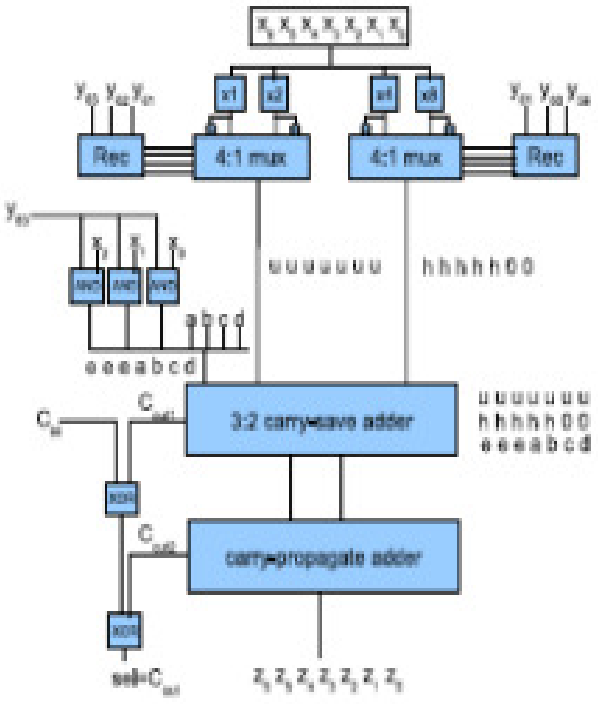


Fig. 4: Computation of part B.

Fig. 5 shows the recoding and partial product generation stage including the high level view of the hardware scheme proposed.

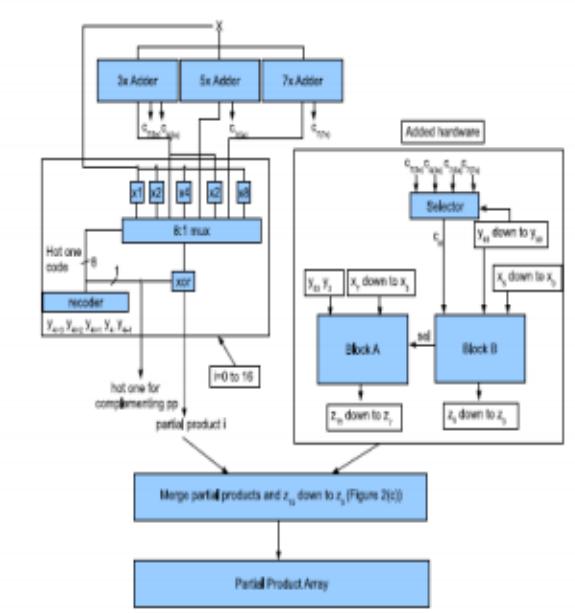
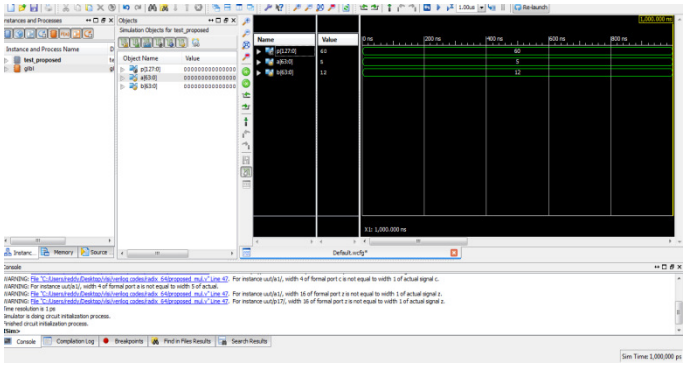


Fig. 5: High level view of the recoding and partial product generation stage including our proposed scheme.

The way we compute part B may still lead to an inconsistency with the computation of the most significant part of partial product 15. Specifically, when partial product 15 is the result of an odd multiple, a possible carry from the 7 least-significant bits is already incorporated in the most significant part of the partial product. During the computation of part B we should not produce again this carry. This issue is solved as follows. Let us consider first the case of positive odd multiples. Fig. 4 shows that the computation of part B may generate two carry outs: the first from the 3:2 carry-save adder (Cout1), and the second from the carry-propagate adder (Cout2). To avoid inconsistencies, we detect the carry propagated to the most significant part of the partial product 15 (we call this CM) and subtract it from the two carries generated in part B.

IV. RESULTS



Simulation result of the proposed system

Table 1: comparison table

	EXISTING	PROPOSED
NUMBER OF SLICE LUT'S USED	9637	157
DELAY	18.466ns	10.860ns

V. CONCLUSION

The multiplier using the proposed algorithm achieves better power-delay analysis than those achieved by conventional Booth multipliers. Here, we have presented a method to reduce by one the maximum height of the partial product array for 64-bit,128-bit radix-16 Booth recoded magnitude multipliers. This reduction may allow more flexibility in the design of the reduction tree of the pipelined multiplier and achieved with no extra delay for $n \geq 32$ for a cell-based design. We believe that the proposed Booth algorithm can be broadly utilized in general processors as well as digital signal processors, mobile application processors, and various arithmetic units that use Booth encoding.

REFERENCES

[1] S. Kuang, J. Wang, and C. Guo, "Modified booth multipliers with a regular partial product array," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 56, no. 5, pp. 404–408, May 2009.

[2] F. Lamberti et al., "Reducing the computation time in (short bit-width) twos complement multipliers," IEEE Trans. Comput., vol. 60, no. 2, pp. 148–156, Feb. 2011.

[3] N. Petra et al., "Design of fixed-width multipliers with linear compensation function," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 5, pp. 947– 960, May 2011.

[4] S. Galal et al., "FPU generator for design space exploration," in Proc. 21st IEEE Symp. Comput. Arithmetic (ARITH), Apr. 2013, pp. 25–34.

[5] K. Tsoumanis et al., "An optimized modified booth recoder for efficient design of the add-multiply operator," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 4, pp. 1133–1143, Apr. 2014.

[6] A. Cilardo et al., "High speed speculative multipliers based on speculative carry-save tree," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 12, pp. 3426–3435, Dec. 2014.

[7] M. Ercegovac and T. Lang, Digital Arithmetic. Burlington, MA, USA: Morgan Kaufmann, 2004.

[8] S. Vassiliadis, E. Schwarz, and D. Hanrahan, "A general proof for overlapped multiple-bit scanning multiplications," IEEE Trans. Comput., vol. 38, no. 2, pp. 172–183, Feb. 1989.

[9] "Binary Multibit Multiplier," Patent 4 745 570 A, 1986.