ISSN 2278-3091
International Journal of Advanced Trends in Computer Science and Engineering, Vol.5 , No.1, Pages : 65 -69 (2016)
Special Issue of ICACEC 2016 - Held during 23-24 January, 2016 in Institute of Aeronautical Engineering, Quthbullapur, Telangana-43, India

AN HIGH SPEED TWO'S COMPLEMENT MULTIPLIER REALIZATION IN FPGA

Srihari C[1], Anil Reddy G[2], Sruthi L[3]

Abstract— This paper focuses Two's complement multipliers with Short Bit-Width were used without any increase in the delay of the partial product generation stage. This was done by reducing one row the maximum height of the partial product array generated by a radix-4 Modified Booth Encoded multiplier, this reduction may allow for a faster compression of the partial product array and regular layouts. This technique is of particular interest in all multiplier designs, but especially in short bit-width two's complement multipliers for high-performance embedded cores.By implementing this method , it will reduce the Computation Time in Two's Complement multipliers by Short Bit-Width concept. This method is general and can be extended to higher radix encodings, as well as to any size square and m x n rectangular multipliers.

Index Terms— Multiplication, Modified Booth Encoding, partial product array.

INTRODUCTION

High processing performance and low power dissipation are the most important objectives in many multimedia and digital signal processing (DSP) systems, where multipliers are always the fundamental arithmetic unit and significantly influence the system's performance and power dissipation. To achieve high performance, the modified Booth encoding [3] which reduces the number of partial products by a factor of two through performing the multiplier recoding has been widely adopted in parallel multipliers.

METHODOLOGY

The basic algorithm for multiplication is based on the well-known paper and pencil approach [1] and passes through three main phases: 1) partial product (PP) generation, 2) PP reduction, and the 3)final (carry-propagated) addition. During PP generation, a set of rows is generated where final (carry-propagated) addition. During PP generation, a set of rows is generated where each one is the result of the product of one bit of the multiplier by the multiplicand.

For example, if we consider the multiplication X x Y with both X and Y on bits and of the form $x_{n-1} \ldots x_0$ and $y_{n-1} \ldots y_0$, then the i^{th} row is, in general, a proper left shifting of yi x X, i.e., either a string of all zeros when yi = 0, or the multiplicand X itself when yi = 1. In this case, the number of PP rows generated during the first phase is clearly n. a number of strategies for preventing sign extension have been developed. The array resulting from the application of the sign extension prevention technique in [1] to the partial product array of a 8 x 8 MBE multiplier

MODIFIED BOOTH MULTIPLIER

Let us consider the multiplication operation of two bit signed numbers (multiplicand) and (multiplier). Modified Booth Encoding (MBE) [3] is a technique that has been introduced to reduce the number of PP rows, still keeping the generation process of each row both simple and fast enough. One of the most commonly used schemes is radix-4 MBE, for a number of reasons, the most important being that it allows for the reduction of the size of the partial product array by almost half, and it is very simple to generate the multiples of the multiplicand.

More specifically, the classic two's complement n x n bit multiplier using the radix- 4 MBE scheme, generates a PP array with a maximum

ISSN 2278-3091
International Journal of Advanced Trends in Computer Science and Engineering, Vol.5 , No.1, Pages : 65 -69 (2016)
Special Issue of ICACEC 2016 - Held during 23-24 January, 2016 in Institute of Aeronautical Engineering, Quthbullapur, Telangana-43, India

height of (n/2)+1 rows, each row before the last one being one of the following possible values: all zeros, ±X , ±2X. The last row, which is due to the negative encoding, can be kept very simple by using specific techniques integrating two's complement and sign extension prevention [1]. The PP reduction is the process of adding all PP rows by using a compression tree [4], [5]. Since the knowledge of intermediate addition values is not important, the outcome of this phase is a result represented in redundant carry save form, i.e., as two rows, which allows for much faster implementations. The final (carry-propagated) addition has the task of adding these two rows and of presenting the final result in a non redundant form, i.e., as a single row.

In this work, an idea is introduced to overlap, to some extent, the PP generation and the PP reduction phases. Here, the aim is to produce a PP array with a maximum height of n/2 rows that is then reduced by the compressor tree stage.

y_{2i+1}	y_{2i}	y_{2i-1}	Generated partial products
0	0	0	$0 \times X$
0	0	1	$1 \times X$
0	1	0	$1 \times X$
0	1	1	$2 \times X$
1	0	0	$(-2) \times X$
1	0	1	$(-1) \times X$
1	1	0	$(-1) \times X$
1	1	1	$0 \times X$

TABLE 1
Modified Booth Encoding (Radix-4)

As the above reduction can lead to an implementation where the delay of the compressor tree is reduced by one XOR2 gate keeping a regular layout. Since focusing on small values of n and fast single-cycle units, this reduction might be important in cases where, for example, a high computation performance through the assembly of a large number of small processing units with limited computation capabilities is required, such as 8 x 8 or 16 x 16 multipliers.

As mentioned above, radix-4 MBE is particularly of interest since, for radix-4, it is easy to create the multiples of the multiplicand 0; ±X; ±2X. In particular, ±2X can be simply obtained by single left shifting of the corresponding terms ±X. It is clear that the MBE can be extended to higher radices, but the advantage of getting a higher reduction in the number of rows is paid for by the need to generate more multiples of X. In this paper, the attention is on radix-4 MBE only. From an operational point of view, it is well known that the radix-4 MBE scheme consists of scanning the multiplier operand with a three-bit window and a stride of two bits (radix-4). For each group of three bits (y_{2i+1}, y_{2i}, y_{2i-1}), only one partial product row is generated according to the encoding in Table 1. A possible implementation of the radix-4 MBE and of the corresponding partial product generation is shown in Fig. 1a produces the one, two, and neg signals. These signals are then exploited by the logic in Fig. 1b, along with the appropriate bits of the multiplicand, in order to generate the whole partial product array. Other alternatives for the implementation of the recoding and partial product generation can be found in among others.

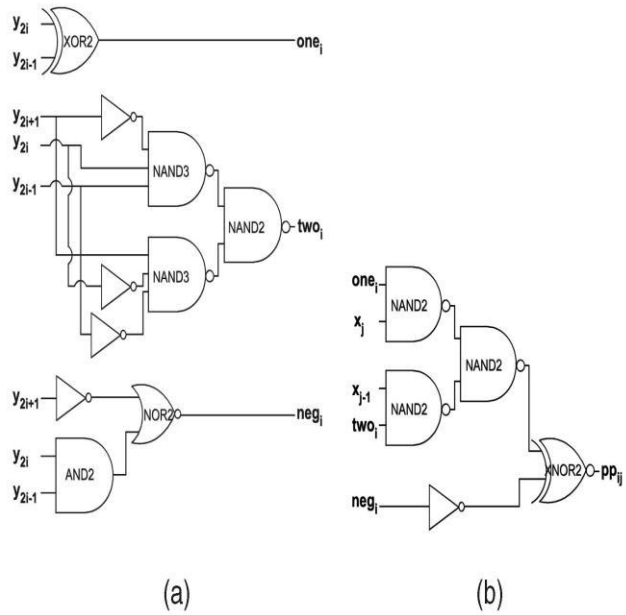

Fig. 1. Gate-level diagram for partial product generation using MBE(a) MBE signals generation. (b) Partial product generation.

As introduced previously, the use of radix-4 MBE allows for the (theoretical) reduction of the PP rows to n/2, with the possibility for each row to host a multiple of yi x X, with yi € {0, ±1, ±2}. While it

ISSN 2278-3091
International Journal of Advanced Trends in Computer Science and Engineering, Vol.5 , No.1, Pages : 65 -69 (2016)
Special Issue of ICACEC 2016 - Held during 23-24 January, 2016 in Institute of Aeronautical Engineering, Quthbullapur, Telangana-43, India

is straightforward to generate the positive terms 0, X, and 2X at least through a left shift of X, some attention is required to generate the terms -X and -2X which, as observed in Table 1, can arise from three configurations of the y_{2i+1}, y_{2i}, y_{2i-1} bits. To avoid computing negative encodings, i.e., -X and -2X, the two's complement of the multiplicand is generally used. From a mathematical point of view, the use of two's complement requires extension of the sign to the leftmost part of each partial product row, with the consequence of an extra area overhead. Thus, a number of strategies for preventing sign extension have been developed. The array resulting from the application of the sign extension prevention technique in [1] to the partial product array of a 8 x 8 MBE multiplier [3] is shown in Fig. 2.

The use of two's complement requires a neg signal (e.g., neg0, neg1, neg2, and neg3 in Fig. 2) to be added in the LSB position of each partial product row for generating the two's complement, as needed. Thus, although for a n x n multiplier, only n/2 partial products are generated, the maximum height the maximum height of partial product array is (n/2) + 1.

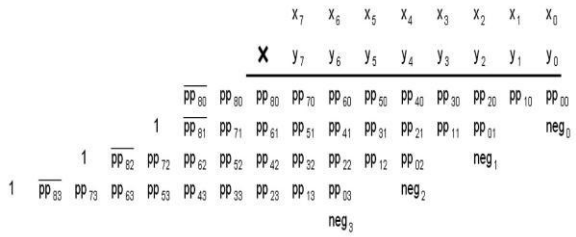

Fig. 2. Application of the sign extension prevention measure [1] on the partial product array of a 8 x 8 radix-4 MBE multiplier.

When 4-to-2 compressors are used, which is a widely used option because of the high regularity of the resultant circuit layout for n power of two, the reduction of the extra row may require an additional delay of two XOR2 gates. By properly connecting partial product rows and using a Wallace reduction tree [5], the extra delay can be further reduced to one XOR2.

However, the reduction still requires additional hardware, roughly a row of n half adders. This issue is of special interest when n is a power of two, which is by far a very common case, and the multiplier's critical path has to fit within the clock period of a high performance processor. For instance, in the design presented in [2], for n = 16, the maximum column height of the partial product array is nine, with an equivalent delay for the reduction of six XOR2 gates. For a maximum height of the partial product array of 8, the delay of the reduction tree would be reduced by one XOR2 gate. Alternatively, with a maximum height of eight, it would be possible to use 4 to 2 adders, with a delay of the reduction tree of six XOR2 gates, but with a very regular layout.To achieve the goal of eliminating the extra row before the PP reduction phase is based on computing the two's complement of the last partial product, thus eliminating the need for the last neg signal, in a logarithmic time complexity.

A special tree structure is used in order to produce the two's complement (Fig. 3), by decoding the MBE signals through a 3-5 decoder (Fig. 4a). Finally, a row of 4-1 multiplexers with implicit zero output1 is used (Fig. 4b) to produce the last partial product row directly in two's complement, without the need for the neg signal. The goal is to produce the two's complement in parallel with the computation of the partial products of the other rows with maximum overlap.

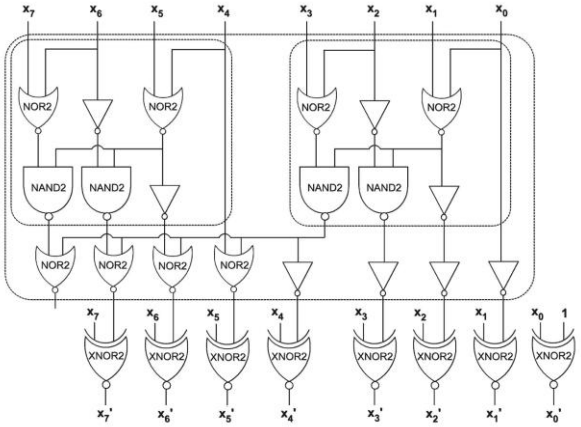

Fig. 3. Two's complement computation

ISSN 2278-3091
International Journal of Advanced Trends in Computer Science and Engineering, Vol.5 , No.1, Pages : 65 -69 (2016)
Special Issue of ICACEC 2016 - Held during 23-24 January, 2016 in Institute of Aeronautical Engineering, Quthbullapur, Telangana-43, India

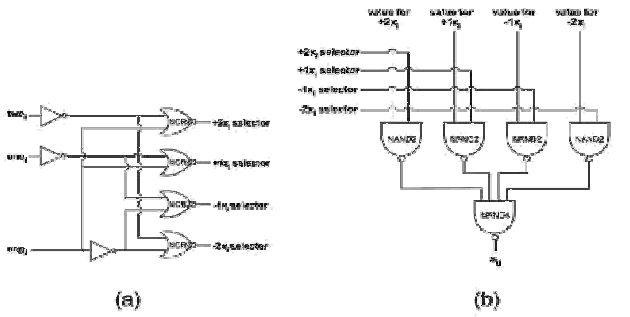

Fig. 4. Gate-level diagram for the generation of two's complement partial product rows (a) 3-5 decoder. (b) 4-1 multiplexer.

SQUARE MULTIPLIERS

For all the partial product rows except the first one. As depicted in Fig. 5a, the first row is temporarily considered as being split into two sub rows, the first one containing the partial product bits (from right to left) from pp_{00} to pp_{80} and the second one with two bits set at "one" in positions 9 and 8. Then, the bit neg3 related to the fourth partial product row, is moved to become a part of the second sub row. The key point of this "graphical" transformation is that the second sub row containing also the bit neg3, can now be easily added to the first sub row, with a constant short carry propagation of three positions (further denoted as "3-bits addition"), a value which is easily shown to be general, i.e., independent of the length of the operands, for square multipliers. In fact, with reference to the notation of Fig. 5, we have that qq_{90} qq_{90} qq_{80} qq_{70} qq_{60} = 0 0 pp_{80} pp $_{70}$ pp_{60} + 0 1 1 0 neg3.

As introduced above, due to the particular value of the second operand, i.e., 0 1 1 0 neg3, have observed that it requires a carry propagation only across the least-significant three positions, a fact that can also be seen by the implementation shown in Fig. 6. It is worth observing that, in order not to have delay penalizations, it is necessary that the generation of the other rows is done in parallel with the generation of the first row cascaded by the computation of the bits qq_{90} qq_{90} qq_{80} qq_{70} qq_{60} in Fig. 5b. In order to achieve this, simplify and differentiate the generation of the first row with respect to the other rows.

In particular, by direct comparison of Figs. 1 and 7, the generation of the MBE signals for the first row is simpler, and theoretically allows for the saving of the delay of one NAND3 gate. In addition, the implementation in Fig. 8 has a delay that is smaller than the two parts of Fig. 7, although it could require a small amount of additional area, since this extra hardware is used only for the three most significant bits of the first row, and not for all the other bits of the array.

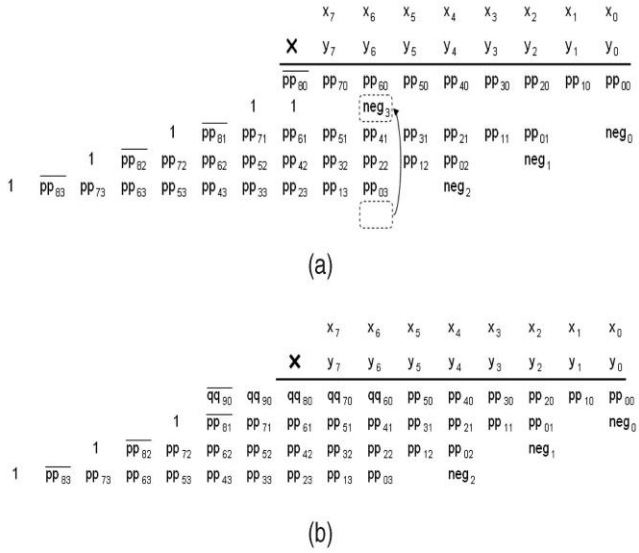

Fig. 5. Partial product array after adding the last neg bit to the first row (a) Basic idea. (b) Resulting array.

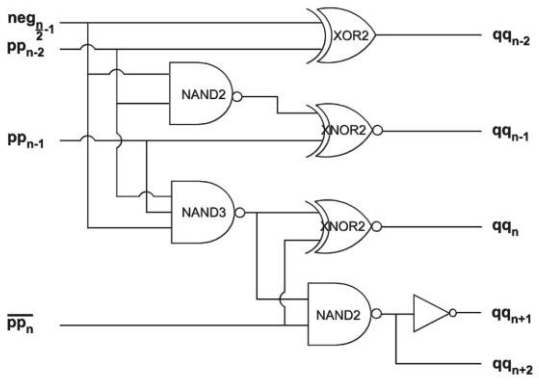

Fig. 6. Gate-level diagram of the proposed method for adding the last neg bit in the first row.

ISSN 2278-3091
International Journal of Advanced Trends in Computer Science and Engineering, Vol.5 , No.1, Pages : 65 -69 (2016)
Special Issue of ICACEC 2016 - Held during 23-24 January, 2016 in Institute of Aeronautical Engineering, Quthbullapur, Telangana-43, India

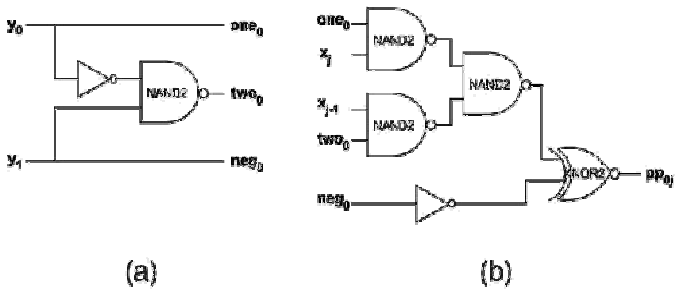

Fig. 7. Gate-level diagram for first row partial product generation.(a)MBE signals generation. (b) Partial product generation

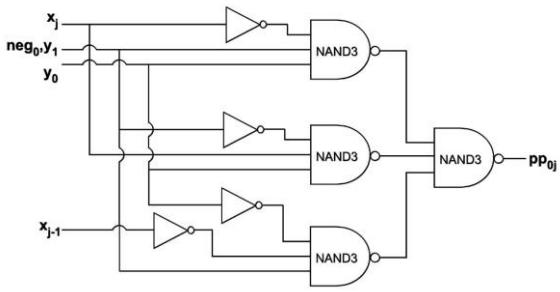

Fig. 8. Combined MBE signals and partial product generation for the first row (improved for speed).

1. generation of the three most significant bit weights of the first row, plus addition of the last neg bit: possible implementations can use a replication of three times the circuit of Fig. 8 (each for the three most significant bits of the first row), cascaded by the circuit of Fig. 6 to add the neg signal.

2. parallel generation of the other bits of the first row: possible implementations can use instances of the circuitry depicted in Fig. 7, for each bit of the first row, except for the three most significant

3. parallel generation of the bits of the other rows: possible implementations can use the circuitry of Fig. 1, replicated for each bit of the other rows.

All items 1 to 3 are independent, and therefore can be executed in parallel. Clearly if, as assumed and expected, item 1 is not the bottleneck (i.e., the critical path), then the implementation of the proposed idea has reached the goal of not introducing time penalties.

CONCLUSIONS

Two's complement n x n multipliers using radix-4 Modified Booth Encoding produce n/2 partial products but due to the sign handling, partial product array has a maximum height of (n/2)+1. Here the partial product array is reduced with a maximum height of n/2 , without introducing any extra delay in the partial product generation stage. With the extra hardware of a (short) 3-bit addition, and the simpler generation of the first partial product row, a delay for the proposed scheme within the bound of the delay of a standard partial product row generation is achieved. The outcome of the above is that the reduction of the maximum height of the partial product array by one unit may simplify the partial product reduction tree, both in terms of delay and regularity of the layout. This is of special interest for all multipliers, and especially for single-cycle short bit-width multipliers for high performance embedded cores, where short bit-width multiplications are common operations.By using this multiplication MAC application is developed.

REFERENCES

[1] Fabrizio Lamberti, Nikos Andrikos, Elisardo Antelo, and Paolo Montuschi, " Reducing the Computation Time in (Short Bit-Width) Two's Complement Multipliers," IEEE transactions on computers, vol. 60, no. 2, Feb. 2011.

[2] S.K. Hsu, S.K. Mathew, M.A. Anders, B.R. Zeydel, V.G.Oklobdzija, R.K. Krishnamurthy, and S.Y. Borkar, "A 110GOPS/W 16-Bit Multiplier and Reconfigurable PLA Loop in 90-nm CMOS," IEEE J. Solid State Circuits, vol. 41, no. 1, pp. 256-264, Jan.2006.

[3] O.L. MacSorley, "High Speed Arithmetic in Binary Computers," Proc. IRE, vol. 49, pp. 67-91, Jan. 1961.

[4] L. Dadda, "Some Schemes for Parallel Multipliers," Alta Frequenza, vol. 34, pp. 349-356, May 1965.

[5] C.S. Wallace, "A Suggestion for a Fast Multiplier," IEEE Trans.Electronic Computers, vol. EC-13, no. 1, pp. 14-17, Feb. 1964.