# An FPGA Implementation of Faster Compression of the Partial Product Array in Two's Complement Multiplier

[#1]P.Bala kishore, [#2] Mrs. P. Pushpa latha
[#1] *PG Student, Department of ECE, University College of Engineering, Kakinada, A.P, India*
[#2] *Assistant Professor, Department of ECE, University College of Engineering, Kakinada, A.P, India*
*pushpalatha86@gmail.com,*
pblakishore@yahoo.com

*Abstract* - **All multipliers are important for a wide range of applications. This technique is particular interest in all multiplier designs, but especially in short bit-width two's complement multipliers for high-performance embedded cores. With the extra hardware of a (short) 3-bit Addition, and the simpler generation of the first partial product row, we have been able to achieve a delay for the proposed scheme within the bound of the delay of a standard partial product row generation. This present a technique to reduce by one row the maximum height of the partial product array generated by a radix-4 Modified Booth Encoded multiplier, without any increase in the delay of the partial product generation stage. This reduction may allow for a faster compression of the partial product array and regular layouts. This method is used for higher radices encoding for any size of mxn multiplications. We evaluated the proposed approach by comparison with some other possible solutions; modest improvements in area (about 15%) and power (about 25%) over more conventional algorithms have been shown using this algorithm.**

*Keywords* --- **Multiplication, Radix-4, Modified Booth Encoding, partial product array.**

## 1 INTRODUCTION

In signal processing applications performance strongly depends on the effectiveness of the hardware used for computing multiplications. The high interest in this field is witnessed by the large amount of algorithm and implementations of the multiplication operations. In this short bit width (8-16bits) two's complement multipliers with single-cycle throughput and latency have emerged to be important building blocks for high performance embedded processors and DSP execution cores. Applications for short bit-width multipliers are the design of SIMD units supporting different data formats. The basic algorithm for multiplication is based three main phases:1) partial product (PP) generation, 2) PP reduction, and 3) 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.

Modified Booth Encoding (MBE) is a technique that has been introduced to reduce the number of PP rows, still keeping the generation process of 2 each row both simple and 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.

The PP reduction is the process of adding all PP rows by using a compression tree 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 addition has the task to sum these two rows and to present the final result in non redundant form i.e., as a single row.

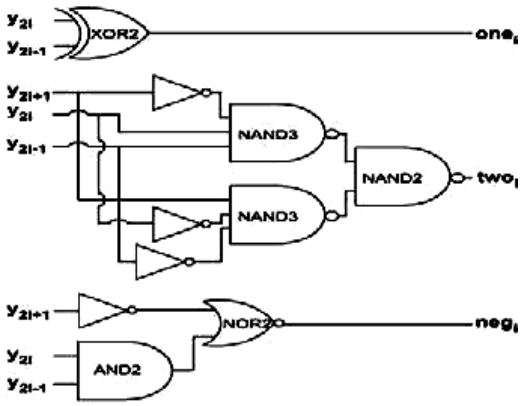
## 2 MODIFIED BOOTH ENCODING (RADIX-4)

| $Y_{2i+1}$ | $Y_{2i}$ | $Y_{2i-1}$ | Generated Partial Product |
|---|---|---|---|
| 0 | 0 | 0 | 0 x X |
| 0 | 0 | 1 | 1 x X |
| 0 | 1 | 0 | 1 x X |
| 0 | 1 | 1 | 2 x X |
| 1 | 0 | 0 | (-2) x X |
| 1 | 0 | 1 | (-1) x X |
| 1 | 1 | 0 | (-1) x X |
| 1 | 1 | 1 | 0 x X |

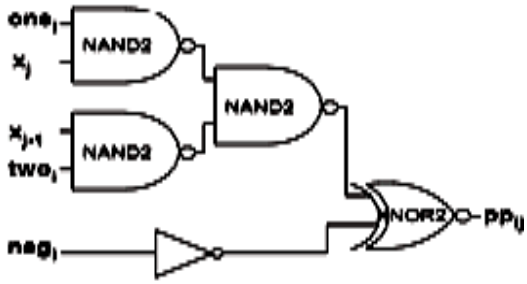
Table. 1 Modified Booth Encoding (RADIX-4)

The method is to compute the product of a multiplicand *X* and a multiplier *Y*, is to produce the partial product array by generating one row for each bit of the multiplier *Y*. This methodology produces n rows, where n is the size of the multiplier. In general, a radix-B *=2b* MBE leads to a reduction of the number of rows to about [*n/b*] while, on the other hand, it introduces the need to generate all the multiples of the multiplicand *X*, at least from - *B/2 x X* to *B/2 x X*. R adix-4 is easy to create the multiples of the multiplicand *0; ±X; ±2X*. *±2X* can be simply obtained by single left shifting of the corresponding terms *±X*.

© 2013 IJAIR. ALL RIGHTS RESERVED 




(a) MBE signals generation




(b) Partial product generation

Fig.1. Gate level diagram for partial product generation using MBE (adapted from [8]).

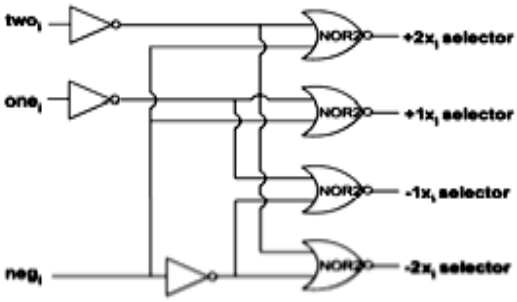
Radix-4 MBE scheme consists of scanning the multiplier operand with a three-bit window and a stride of two bits. For each group of three bits ($y_{2i+1}$, $y_{2i}$, $y_{2i-1}$), only one partial product row is generated according to the encoding in Table 1. For each partial product row, Fig. 1a produces the one, two, and negative 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. 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 $Y_i \times X$, with $Y_i \in \{0; \pm 1; \pm 2\}$

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 $y_{2i+1}$ , $y_{2i}$ , an *y* bits. To avoid computing negative encodings, i.e., *-X* and *- 2X*, the two's complement of the multiplicand is generally used. 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. For 2's complement it requires a negative signal to be added in the LSB position of each partial product row. For nxn multiplier, only [*n*/2] partial products are generated, the maximum height of the partial product array is [*n*/2] +1.

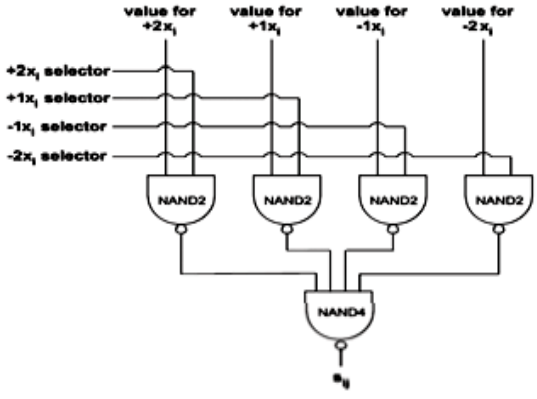
When 4-to-2 compressors are used the reduction of the extra row may require an additional delay of two XOR2 gates. By properly connecting partial product rows and using a Wallace reduction tree, the extra delay can be further reduced to one XOR2. However, the reduction still requires additional hardware, roughly a row of n half adders. This issue is of special interest when n is a power of 2, 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.

## 3 Related Works

This approach is based on computing the two's complement of the last partial product, thus eliminating the need for the last negative signal, in a logarithmic time complexity. A special tree structure is used in order to produce the two's complement by decoding the MBE signals through a 3-5 decoder (Fig. 2a). Finally, a row of 4-1 multiplexers with implicit zero output1 is used (Fig. 2b) to produce the last partial product row directly in two's complement, without the need for the negative 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. In such a case, it is expected to have no or a small time penalization in the critical path. An example of the partial product array produced using the above method is depicted in Fig. 2




a) 3-5 decoder




b) 4-1 multiplexer

Fig.2. Gate level diagram for the generation of two's complement partial product rows.

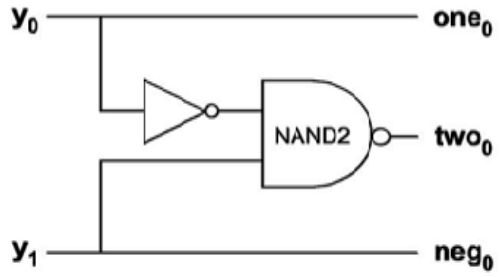
© 2013 IJAIR. ALL RIGHTS RESERVED 

## 4 BASIC IDEA

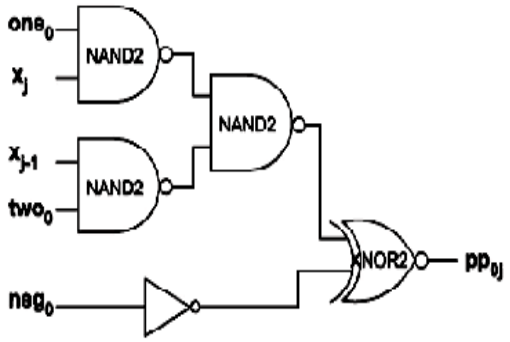
### *4.1 Square Multipliers*

1. Generation of most significant three bit weights of the first row, plus addition of the last negative bit. Possible implementations can use a replication of three times the circuit of (each for the three most significant bits of the first row), cascaded by the circuit of to add the negative signal;
2. Parallel generation of the other bits of the first row: possible implementations can use instances of the circuitry depicted in below Fig, for each bit of the first row, except for the three most significant.
3. Parallel generation of the bits of the other row 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.




(a) MBE signals generation.




(b) Partial product Generation.

Fig.3. Gate-level diagram for first row partial product generation.

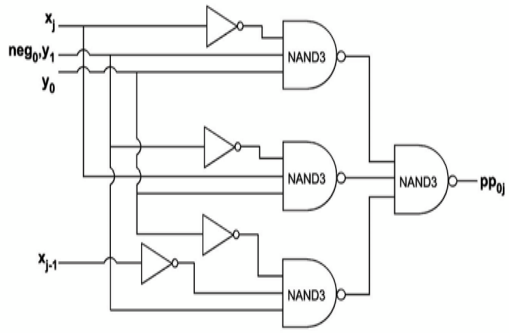



Fig.4. Combined MBE signals and partial product generation for the first row (improved for speed).

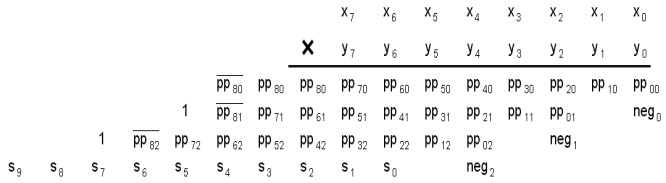
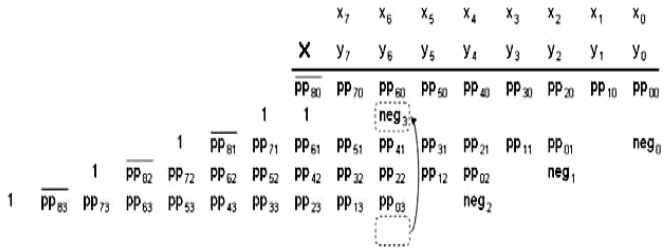
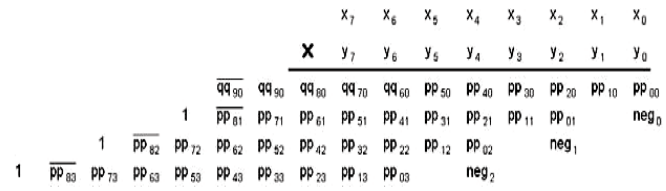



Fig. 5 Partial product array by applying the two's complement computation method in to the last row.




(a) Basic idea




(b) Resulting array

Although we have explicitly focused our attention to radix-4 MBE, the proposed method can be easily extended to any radix-B MBE. It is easily observed, by redrawing the equivalent of Fig. 6(a) for another radix-B MBE, that the negative signal of the last row can be included in the first row by using a simple 3-bit adder for a $n \times n$ multiplier and by using a $(m0 + 3)$-bit adder for the more general case of a $(n + m0) \times n$ adder. The use of a combined multiplier with accumulation is also common. In this case, the proposed approach can also be used and does not lose the benefit to reduce by one the number of rows to be added. Although this reduction does not necessarily lead to a reduction in the computation time for some values of $n$, it still remains interesting and useful since it is very reasonable to expect

© 2013 IJAIR. ALL RIGHTS RESERVED 

some savings and more regularity in the compression tree. We have not evaluated such potential reductions, because their impact could be strongly dependent on the value of *n* and on the strategy which has been identified to design the compression tree.

## 5 Modified Booth Fixed Multiplier

Modified Booth encoding is popular for reducing the number of partial products. The *2L*-bit product *P* can be expressed in two's complement representation as follows: $P=X \times Y$

Booth encoding, where *L* is an even number Taking $8 \times 8$ Booth multiplier as an example, due to the two's complement computation, $n_i$ is equal to *1* when is $y_i$ negative; otherwise, $n_i$ is equal to *0*. The fixed-width Booth multiplier design [2], some of products is truncated by using a rounding operator to hold the data length fixed in L-bit. Therefore, an extra one binary bit 1 added into the most significant column of truncation part (TP) in Fig. 2, which indicates the rounding off operation of the *P-T* Booth multipliers.

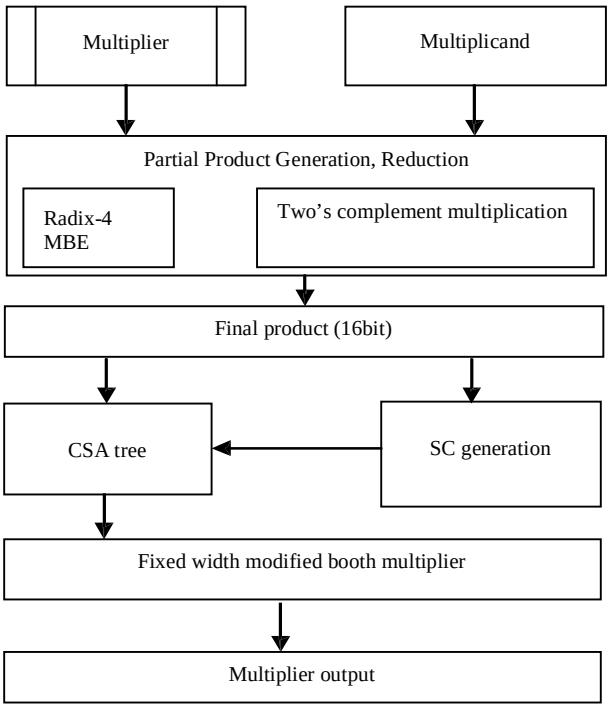



Fig. 6 fast compression of partial product in Two's complement multiplier

## 6 Experimental Results

The proposed booth multiplier to perform multi operand multiplication has been simulated and the synthesis report can be obtained by using Xilinx ISE 10.1i. The various parameters which have been noted are shown in the table.

| Multiplier Types | Power (mW) | Delay (ns) | Area (Gate count) |
|---|---|---|---|
| Before PP Reduction (Conventional Multiplier) | 200 | 7.199 | 1613 |
| After PP Reduction (Short bit-width) | 132 | 7.000 | 1277 |
| Fixed-width MBE | 109 | 6.700 | 1033 |

Table 2: Output for Power consumption and delay

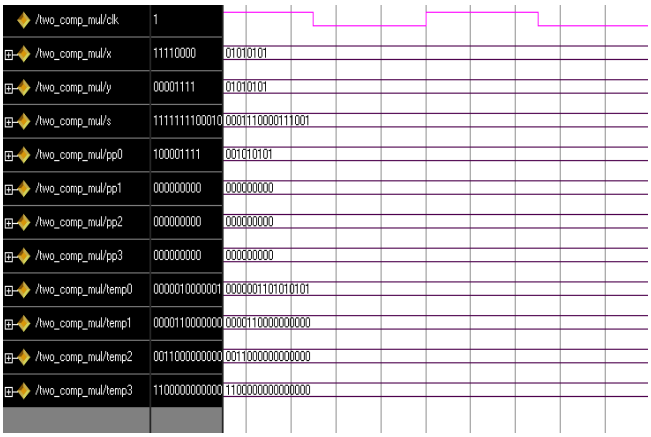



Fig7: simulation output for after reduction of the pp row

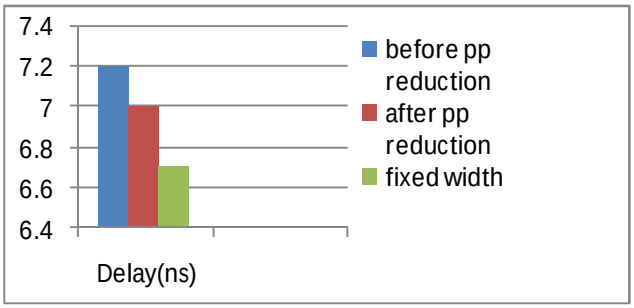



Chart 1: Result for delay

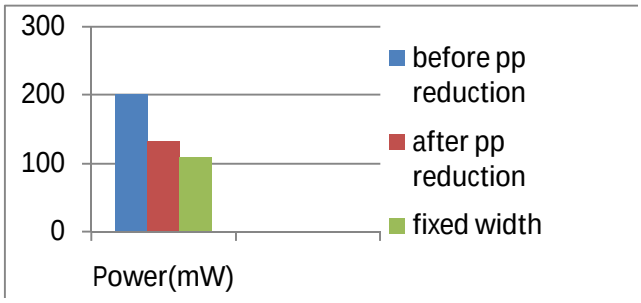



Chart 2: Result for Power

© 2013 IJAIR. ALL RIGHTS RESERVED 

## CONCLUSIONS

Two's complement $n \times n$ bit multipliers using $n$ radix-4 Modified Booth Encoding produce [n/2] partial products but due the sign handling, the partial product array has a maximum height of [$n$/2] +1.We presented a scheme that produces a partial product array with a maximum height of [$n$/2], without introducing extra delay in the partial product generation stage for $m \times n$ bit multipliers. With the extra hardware of a (short) 3-bit addition, and the simpler generation of the first partial product, we have been allowed to achieve a delay for the proposed scheme within the bound of the delay of a standard partial product generation.

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, in terms of delay and regularity of the layout. This is of special interest for all multipliers but especially for short bit-width multipliers for high performance embedded cores, where short but fast bit-width multiplications could be common operations.

## REFERENCES

[1] M. D. Ercegovac and T. Lang, "Digital Arithmetic". Los Altos, CA, USA: *Morgan Kaufmann Publishers,* 2003.
[2] S. K. Hsu, S. K. Mathew, M. A. Anders, B. R. Zeydel, V. G. Ok- lobdzija, R. K. Krishnamurthy, and S. Y. Borkar, "A 110gops/w 16-bit multiplier and reconfigurable pla loop in 90-nm cmos," *IEEE Journal of Solid State Circuits, vol. 41, pp. 256–264,* Jan. 2006.
[3] M. S. Schmookler, M. Putrino, A. Mather, J. Tyler, H. V. Nguyen, C. Roth, M. Sharma, M. N Pham, and J. Lent, "A low-power, high-speed implementation of a power pc(tm) microprocessor vector extension," Proceedings of the 14th *IEEE Symposium on Computer Arithmetic*, p. 12, 1999.
[4] A. D. Booth, "A signed multiplication technique," Quarterly J. Mech. Appl. Math., vol. 4, 1951.
[5] L. Dadda, "Some schemes for parallel msssultipliers," Alta Fre- quenza, vol. 34, May 1965.
[6] O. L. MacSorley, "High speed arithmetic in binary computers," Proceedings IRE, vol. 49, pp. 67–91, Jan. 1961.
[7] J.-Y. Kang and J.-L. Gaudiot, "A simple high-speed multiplier design," IEEE Transactions on Computers, vol. 55, no. 10, pp. 1253–1258, Oct. 2006.
[8] "A fast and well-structured multiplier," *Proceedings of Euromicro Symposium on Digital System Design,* pp. 508–515, Sept.2004.

© 2013 IJAIR. ALL RIGHTS RESERVED 