ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/290929092

Performance Enhancement and Reduced Area Parallel Multiplier

Chapter · January 2015

CITATIONS
2

READS
3,805

4 authors:

Sohiful Anuar Zainol Murad
Universiti Malaysia Perlis
170 PUBLICATIONS **738** CITATIONS
SEE PROFILE

R.C. Ismail
Universiti Malaysia Perlis
125 PUBLICATIONS **599** CITATIONS
SEE PROFILE

Siti Zarina Md Naziri
Universiti Malaysia Perlis
29 PUBLICATIONS **90** CITATIONS
SEE PROFILE

mohd nazrin md isa
Universiti Malaysia Perlis
99 PUBLICATIONS **337** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Internet of Things Applications in Agriculture View project

Project Development of ultra low power RF front-end design for wireless sensor network View project

All content following this page was uploaded by Sohiful Anuar Zainol Murad on 18 January 2016.

The user has requested enhancement of the downloaded file.

CHAPTER 2

Performance Enhancement and Reduced Area Parallel Multiplier

R.C. Ismail, S.Z.M. Naziri, S.A.Z Murad and M.N.M Isa

1. Introduction

Arithmetic operations normally dominate the execution time of most DSP algorithms and currently the time it takes to execute a multiplication operation is still the dominating factor in determining the instruction cycle time of a DSP chip and Reduced Instruction Set Computers (RISC) [1]. Therefore, there has been much work done on advanced multiplication algorithms and designs [2][3][17][18].

In any multiplication process, there are three major steps required. First step, the partial products are generated. Then, the additions of partial products are performed until they are reduced to one row of final sums and one row of carries. Finally, these two rows are added together to generate the result. Modified Booth Encoding (MBE) is the most famous design technique use in the first step [4-6] because of its capability to reduce the number of partial product rows in half. In the second step, the Wallace tree structure [6, 7] is the most widely used architecture to rapidly reduce the number of partial products to the final two rows. In performing the addition process for the final step, the fast adders such as carry look-ahead, carry-select or carry save addition adders are well suited for the implementation [8, 9].

In this paper, we will focus on the first step which consists in reducing the number of partial product rows by performing the two's complement operation even before applying partial products reduction techniques. By having fewer partial product rows, the reduction tree can be smaller in size and faster in speed. The design is structured for 8-bit words as it is one of the most commonly used word sizes in the kernels of most multimedia applications [10].

Further description of the approach is detailed into several sections in this paper. In Section 2, we present the algorithm of multiplication. Meanwhile, the design of architecture is discussed in Section 3. Simulation and synthesis details are presented in Section 4 and conclusion in Section 5.

2. Algorithm of Multiplication

There are many existing methods that can be used in performing the multiplication process. Among them are shift and add, Booth multiplication [2], signed number [11] and redundant binary arithmetic techniques [12]. Booth multiplication is one of the proven techniques that allows smaller, faster multiplication circuits, by recoding the numbers that are multiplied [13]. It is also the standard technique employed in FPGA design and provides significant improvements over the "long multiplication" technique.

2.1 Radix-4 Modified Booth Algorithm

The main reason of choosing Radix-4 Modified Booth Algorithm is because of its ability to reduce the number of partial products by half. The basic ideas is that, instead of shifting and adding using shift and add technique, the multiplier bits are clustered in groups as shown in Fig. 1 and basically based on a window size of 3-bit and a stride of 2 [13]. The multiplier (Y) is segmented into groups of three bits (y_{2i+1}, y_{2i}, y_{2i-1}) and each such group of bits is associated with its own partial product row by using Table 1 [14]. In this grouping, consider $y_{-1} = 0$.

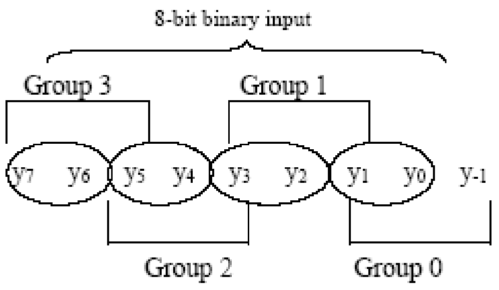


Fig. 1: Multiplier bits grouping according to Booth recording for 8-bit input [13].

Table 1: Radix-4 Modified Booth Recording [14].

y_{2i+1}	y_{2i}	y_{2i-1}	Generated Partial Products
0	0	0	0 * Multiplicand
0	0	1	1 * Multiplicand
0	1	0	1 * Multiplicand
0	1	1	2 * Multiplicand
1	0	0	-2 * Multiplicand
1	0	1	-1 * Multiplicand
1	1	0	-1 * Multiplicand
1	1	1	0 * Multiplicand

By implementing this technique, the numbers of partial product rows to be accumulated can be reduced from η to $\eta/2$. For example, when we are designing an 8×8-bit multiplication using Radix-4 Modified Booth Algorithm, only four ($\eta/2=4$) partial products are generated as shown in Fig. 2 compared with using shift and add technique whereby need to generate eight partial product rows. This is important in circuit design as it relates to the propagation delay in the running of the circuit and the complexity and power consumptions of its implementation.

However, to be clear, there are actually $(\eta/2) + 1$ partial product rows generated rather than $\eta/2$. This is because of the last negation signals are needed due to Radix-4 Modified Booth Algorithm may generate a negative encoding. Due to that, one additional carry save addition stage is required to perform the reduction process and this may lead to the overhead while implementing the negative encodings.

Therefore, the goals is to remove all the two's complement error correction signals which would then prevent the extra partial product row and thus save the time of an additional carry save adding stage and the hardware required for the additional carry save adding.

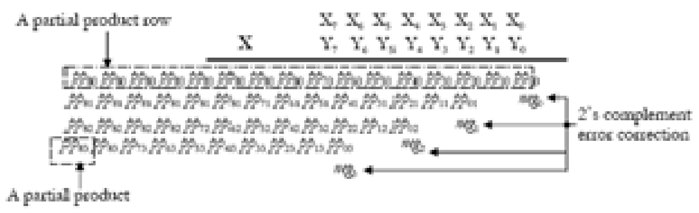


Fig. 2: The array of partial product rows for signed multiplication using Radix-4 Modified Booth Algorithm conventional technique.

2.2 Prevention of Signed Extension

Based on the conventional technique for signed multiplication, the sign bit of partial product row would have to be extended all the way to the MSB position (as shown in Fig. 2) which would then require the sign bit to drive that many output loads (each bit position until the MSB should have the same value as sign). As a result, the partial product rows will be unequal in length. In order to avoid this situation, Ercegovac [15] has proposed a new method whereby the sign extension can be removed as shown in Fig. 3. Therefore, the sign extension prevention method proposed by Ercegovac has been adopted.

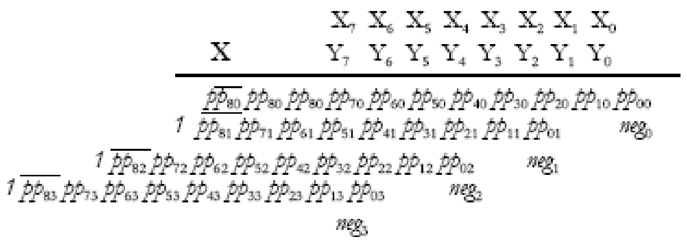


Fig. 3: Sign extension prevention method [15].

3. Design of Architecture

However, the sign extension prevention method introduced by Ercegovac still has the problem of having the last negation signal and this may leads to generate another carry save adder delay in order to perform the final accumulation process.

Therefore, the architecture introduced in this paper will help to prevent the extra partial product row thus save the time of one additional carry save adding stage and the hardware required for the additional carry save adding. We noticed that by performing the two's complement operation even before applying partial products reduction techniques will ensure there is none extra partial product row required. As a result, an easy and efficient design methodology in generating partial product rows is presented and shown in Fig. 4.

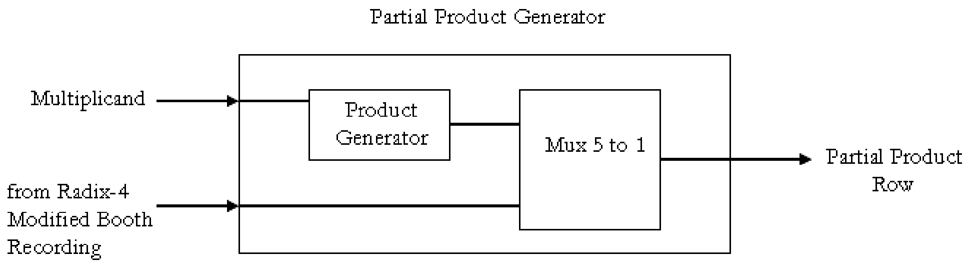

Partial Product Generator
Multiplicand
from Radix-4 Modified Booth Recording
Product Generator
Mux 5 to 1
Partial Product Row

Fig. 4: The block diagram of partial product generator.

3.1 Partial Product Generator

Based on the conventional technique for signed multiplication, the sign bit of partial product row would have to be extended all the way to the MSB position (as shown in Fig. 2) which would then require the sign bit to drive that many output loads (each bit position until the MSB should have the same value as sign). As a result, the partial product rows will be unequal in length. In order to avoid this situation, Ercegovac [15] has proposed a new method whereby the sign extension can be removed as shown in Fig. 3. Therefore, the sign extension prevention method proposed by Ercegovac has been adopted.

For any multiplication of signed numbers using Booth algorithm technique, two's complement operation is needed due to the possibility of generating negative encodings. Generally, the two's complement of an integer is obtained by complementing individual bits and then adding one to the number. Based on the conventional technique, two's complement numbers may be accommodated by simply inverting the negative operands or by so-called post-complementation of the negative results. Then, the error is adjusted using a correction factor applied at the Wallace tree structure on each of partial product rows. As a result, one additional carry save adding stage is needed to perform that operation.

Different with the approach introduced here, the product generator circuit has been designed (using VHDL code) in such a way that it will perform directly two's complement operations as shown in Fig. 5. Therefore, this would prevent the extra partial product row due to unused of correction circuits to accommodate the negative encodings. From gate-level point of view, this circuit consists of adders, inverters and shift register components.

Multiplexers are used to make a selection whether the number for each of rows are from either -1*Multiplicand, 2*Multiplicand, -2*Multiplicand, 1*Multiplicand or all zeroes which based upon the output driven from Radix-4 Modified Booth Recording. There are four multiplexers generated to accommodate the 8-bit multiplication process. The outputs from each of the multiplexers are then connected to Wallace tree structure for performing the addition operation.

By implementing this technique, the partial product rows will no longer required two's complement error correction circuits as well as last negation signal as represented in Fig. 6.

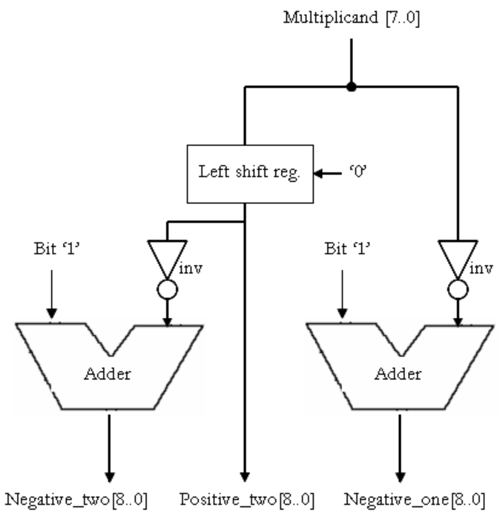


Fig. 5: Details of product generator circuit.

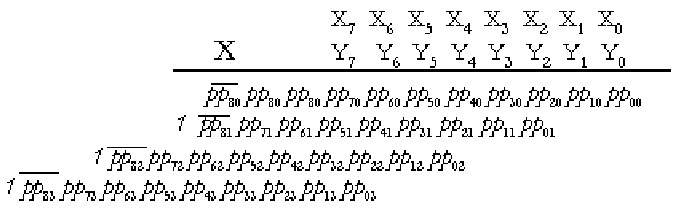
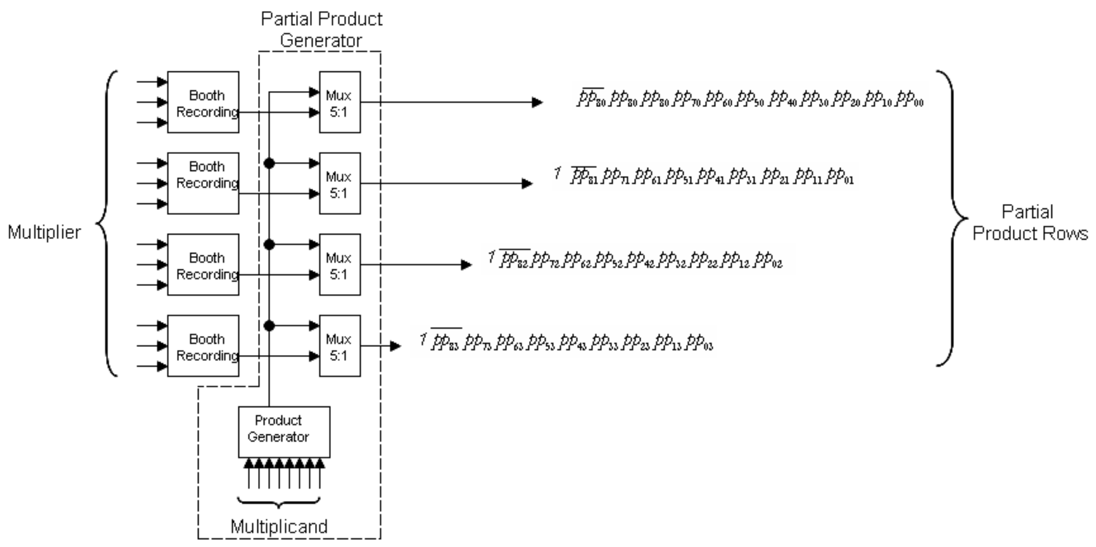


Fig. 6: The addition structure of 8x8-bit signed multiplication.

3.2 Multiplier Architecture

By applying the method that was just described for generating partial product rows, the multiplication can have a smaller critical path. The critical path column which initially with (η/2) + 1 elements (Fig. 3), now have only η/2 elements (Fig. 6). This could directly influences the speed of the multiplication as well as the area of the circuit. Fig. 7 shows the multiplier architecture in generating the partial products for 8x8-bit signed multiplication operation.

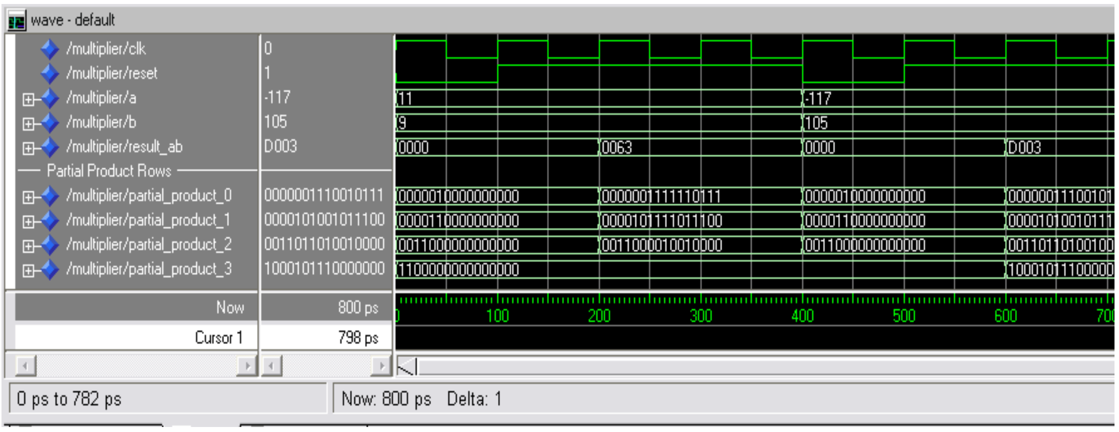


Partial Product Generator
Booth Recording
Booth Recording
Booth Recording
Booth Recording
Mux 5:1
Mux 5:1
Mux 5:1
Mux 5:1
Multiplier
$\overline{pp_{80}}\,pp_{80}\,pp_{80}\,pp_{70}\,pp_{60}\,pp_{50}\,pp_{40}\,pp_{30}\,pp_{20}\,pp_{10}\,pp_{00}$
$1\,\overline{pp_{81}}\,pp_{71}\,pp_{61}\,pp_{51}\,pp_{41}\,pp_{31}\,pp_{21}\,pp_{11}\,pp_{01}$
$1\,\overline{pp_{82}}\,pp_{72}\,pp_{62}\,pp_{52}\,pp_{42}\,pp_{32}\,pp_{22}\,pp_{12}\,pp_{02}$
$1\,\overline{pp_{83}}\,pp_{73}\,pp_{63}\,pp_{53}\,pp_{43}\,pp_{33}\,pp_{23}\,pp_{13}\,pp_{03}$
Partial Product Rows
Product Generator
Multiplicand

Fig. 7: Multiplier architecture.

4. Simulation and Synthesis

The VHDL codes have been successfully simulated using ModelSim XE II (ver. 5.8c) as depicted in Fig. 8, and have been synthesized using Xilinx Project Navigator (ver. 5.2i). As a proof of design concept, the multiplier architecture has been implemented on Xilinx Virtex-II Pro FPGA development board and shown working properly (Fig. 9). The target chip used is Xilinx Virtex-II Pro device number 2VP7FF672 with speed grade set to -6. The maximum operating frequency of the system is capable to achieve up to 729MHz and on its own consumed only 18% of total areas on that FPGA board.



```
wave - default
/multiplier/clk                0
/multiplier/reset              1
/multiplier/a                  -117               11                                                          -117
/multiplier/b                  105                9                                                           105
/multiplier/result_ab          D003               0000                   0063                                 0000                   D003
Partial Product Rows
/multiplier/partial_product_0  0000001110010111   0000010000000000       0000001111110111                     0000010000000000       00000011100101
/multiplier/partial_product_1  0000101001011100   0000110000000000       0000101111011100                     0000110000000000       00001010010111
/multiplier/partial_product_2  0011011010010000   0011000000000000       0011000010010000                     0011000000000000       00110110100100
/multiplier/partial_product_3  1000101110000000   1100000000000000                                                                   10001011100000
                         Now   800 ps             0      100     200     300     400     500     600     70
                    Cursor 1   798 ps
0 ps to 782 ps                 Now: 800 ps   Delta: 1
```

Fig. 8 Simulation results.

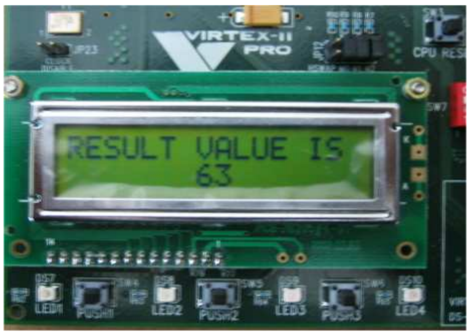


Fig. 9 Output displayed on FPGA board.

5. Conclusion

This paper have presented the partial product generator architecture which is capable to reduce the number of partial product rows from (η/2) + 1 to η/2 at the first step of a multiplication process. By doing so, the structure of partial product becomes easier to be added for the final accumulation. Furthermore, by having fewer partial products, it will not only reduce the area of the circuit but also increase the speed of the system concurrently due to the major source of delay in adders which is consumed from the carry numbers [16].

In order to completely validate this method, designing fast adders are recommended as a future work which can perform the final accumulation process based on generated partial product rows. From this, a complete design of multiplier can be produced and hence the performance could be evaluated and compared with other design approaches.

References

[1] S. Y. Kung, VLSI Array Processors, *Prentice Hall, 1998*.

[2] A. D. Booth, A Signed Binary Multiplication Technique, *Quartely J. Mechanical and Applied Math., 1951*, pp. 236-240.

[3] C. S. Wallace, A Suggestion for a Fast Multiplier, *IEEE Transactions on Computers*, *1964*, pp. 14-17.

[4] B. M. P. E. Madrid, E. E. Swartzlander, Modified Booth Algorithm for High-Radix Fixed Point Multiplication, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *1993*, **1**(2), pp. 164-167.

[5] J. H. J. Kenneth Lin, N. Tang, A High-Performance 32-bit Parallel Multiplier Using Modified Booth Algorithm and Sign Deduction Algorithm, *IEEE Proceedings*, *2003*, pp. 1281-1284.

[6] A. M. A. Lakshmanan, M. Othman, High Performance Parallel Multiplier Using Wallace-Booth Algorithm, *ICSE2002 Proc.*, *2002*, Penang, pp. 433-436.

[7] Y. H. Niichi Itoh, A 600 MHz 54x54-bit Multiplier with Rectangular Styled Wallace Tree, *IEEE Journal of Solid State Circuits*, *2001*, **36(**2), pp. 249-257.

[8] A. G. D. Oscar Gustafsson, Lars Wanhammar, Multipliers Block Using Carry Save Adders, Proc. *IEEE Int. Symp. Circuits Syst.*, *2001*, pp. 472-476.

[9] T. K. William Jao, S. Tjiang, Circuit Optimization Using Carry Save-Adder Cells, *IEEE Transactions on Computers-Aided Design of Integrated Circuits and Systems*, *1998*, **17**, pp. 974-984.

[10] A. J. S. N. Slingerland, Measuring the Performance of Multimedia Instruction Sets, *IEEE Transactions on Computers*, *2002*, **51**(11), pp. 1317-1332.

[11] J. E. Robertson, Two's Complement Multiplication in Binary Parallel Digital Computers, *IRE Trans.*, *1955*, **4**, pp. 118-119.

[12] B. S. S. K. Y. Shin, A Complex Multiplier Architecture Based on Redundant Binary Arithmetic, *IEEE International Symposium on Circuits and Systems*, *1997*, pp. 1944-1947.

[13] J. L. G. J.Y. Kang, A Fast and Well-Structured Multiplier, *EUROMICRO Systems on Digital System Design*, *2004*, pp. 692-701.

[14] O. L. MacSorely, High Speed Arithmetic in Binary Computation, *IEEE Proceedings*, *1990*, **49**, pp. 67-91.

[15] T. L. M. D. Ercegovac, Digital Arithmetic, *Morgan Kaufmann Publishers*, *2003*.

[16] S. Winograd, On the Time Required to Perfom Addition, *Journal of the ACM*, *1965*, **12**(2), pp. 277-285.

[17] Sowjanya, K.; Balivada, Leela Kumari, “Design and Performance Analysis of 32 and 64 Point FFT using Multiple Radix Algorithms”, *International Journal of Computer Applications*, Vol. 78, 2013, pp. 25-29.

[18] L.P Thakare, A.Y Deshmukh, G.D Kandale, “VHDL Implementation of Complex Number Multiplier Using Vedic Mathematics”, *Proceedings of International Conference on Soft Computing Techniques and Engineering Application Advances in Intelligent Systems and Computing* Volume 250, 2014, pp. 403-410.

View publication stats