A High Speed and Well-Structured Partial Product Generator for Parallel Multiplier

R. CHE ISMAIL[1], BECKETT P.[2]

[1]School of Microelectronics Engineering, Univerisiti Malaysia Perlis (UniMAP), Pusat Pengajian Jejawi,02600 Arau, Perlis, Malaysia.
[2]School of Electrical and Computer Eng., Royal Melbourne Institute of Technology (RMIT), Swanston St., Melbourne, 3001 Victoria, Australia.
[1]**rizalafande@unimap.edu.my**

ABSTRACT

Previously reported multiplication algorithms mainly focus on rapidly reducing the partial product rows down to final sums and carries used for the final accumulation. In this paper, an efficient approach for partial product generator is presented. The approach focuses on reducing the number of partial product rows by performing the two's complement operation even before applying partial products reduction techniques. Consequently, this directly influences the speed of the multiplication as well as the area of the circuits.

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 the determining the instruction cycle time of a DSP chip and Reduced Instruction Set Computers (RISC) [8]. Therefore, there has been much work done on advanced multiplication algorithms and designs [2],[15].

In any multiplication process, there are three major steps required. In first step, partial product rows 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 [3,6,10] because of its capability to reduce the number of partial product rows in half. In the second step, the Wallace tree structure [10], [5] 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 [1, 14].

The basic goal of this paper is to propose an efficient approach for generating the partial product rows based on the partial product generator which then suitable for high performance parallel multiplier. The proposed technique is done by performing the two's complement operation even before applying partial products reduction procedures. 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 [13]. Based on the performance evaluation, the speed of the system can be greatly increased by 6-8% as compared to the conventional and Kang's method. On top of that, the total area usage can also be

optimized by more than 30% when it is implemented in the final architecture of the parallel multiplier.

The paper is divided into 5 sections. In the following section (Section 2), we present the algorithm of multiplication. In Section 3, the design of architecture is discussed. Simulation and synthesis details are presented in Section 4 and conclusions in Section 5.

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 for smaller, faster multiplication circuits, by recoding the numbers that are multiplied [17]. It is also the standard technique employed in FPGA design and provides significant improvements over the "long multiplication" technique.

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 grouping as shown in Figure 1 and basically based on a window size of 3-bit and a stride of 2 [17]. 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 [9]. In this grouping, considered $y_{-1} = 0$.

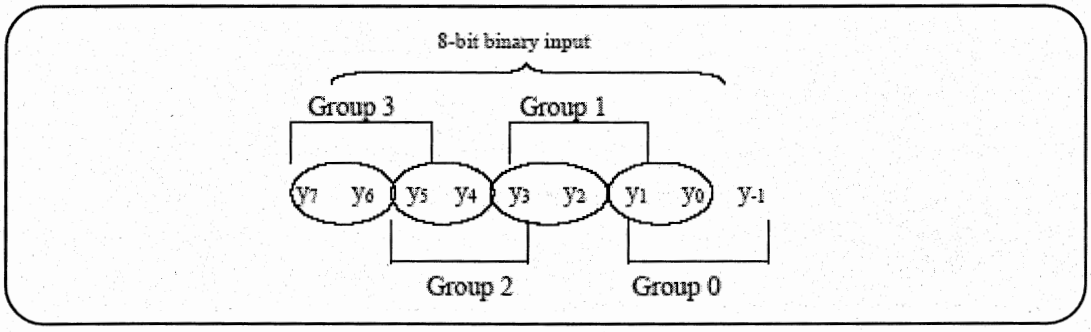

Figure 1. Multiplier bits grouping according to Booth recording for 8-bit input [13].

Table 1. Radix-4 Modified Booth Recording [9].

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 n to n/2. For example, when we are designing an 8x8-bit multiplication using Radix-4 Modified Booth Algorithm, only four (n/2=4) partial products are generated as shown in Figure 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 (n/2) + 1 partial product rows generated rather than n/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.

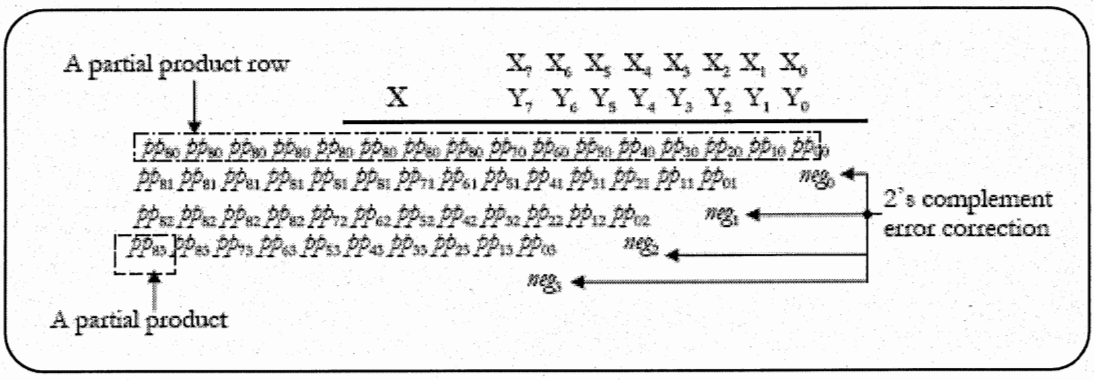

Figure 2. The array of partial product rows for signed multiplication using Radix-4 Modified Booth Algorithm conventional technique.

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 Figure 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 Figure 3. Therefore, the sign extension prevention method proposed by Ercegovac has been adopted.

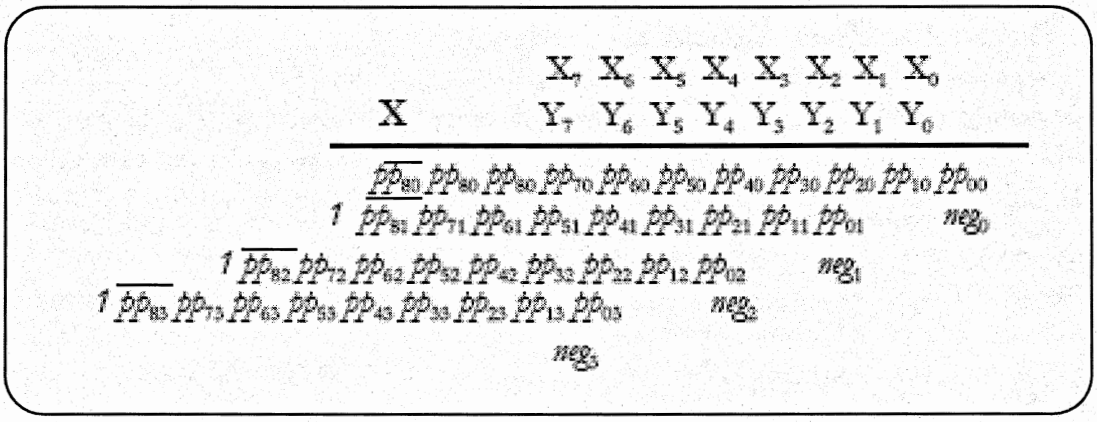

Figure 3. Sign extension prevention method [15].

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 as shown in Figure 4.

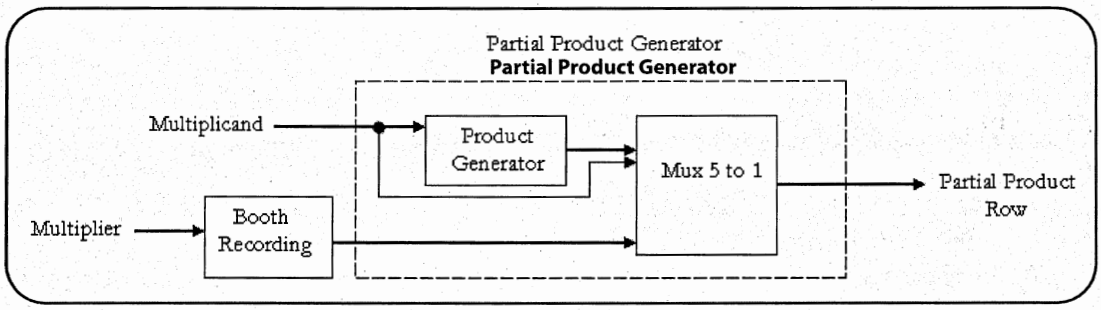

Figure 4. The block diagram of partial product generator.

Partial Product Generator

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. Therefore, this would prevent the extra partial product row due to unused of correction circuits to accommodate the negative encodings.

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 zeros 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 shown in Figure 5.

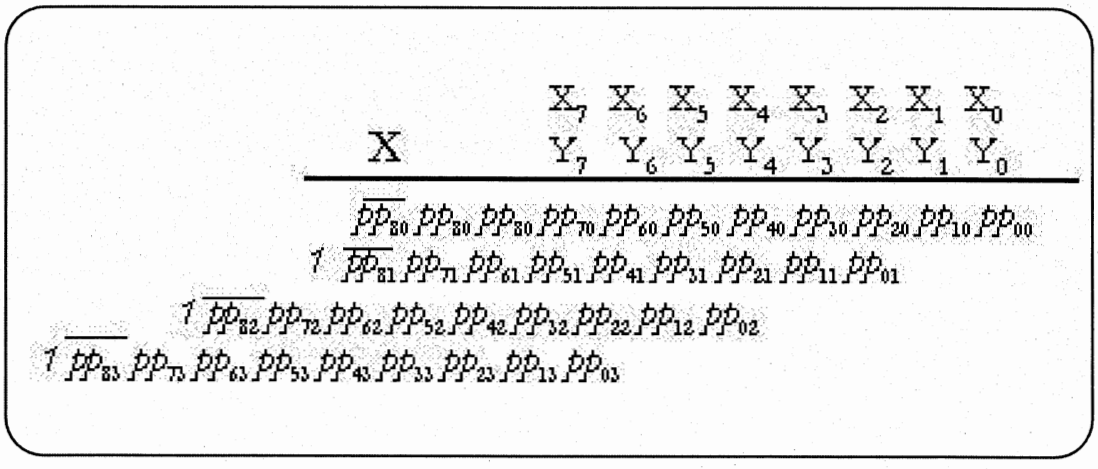

Figure 5. The addition structure of 8x8-bit signed multiplication.

Multiplier Architecture

By applying the method we just described for generating partial product rows, the multiplication can have a smaller critical path. The critical path column which initially with (n/2) + 1 elements (Figure 3), now have only n/2 elements (Figure 5). This directly influences the speed of the multiplication as well as the area of the circuit. Figure 6 shows the multiplier architecture in generating the partial products for 8x8-bit signed multiplication operation.

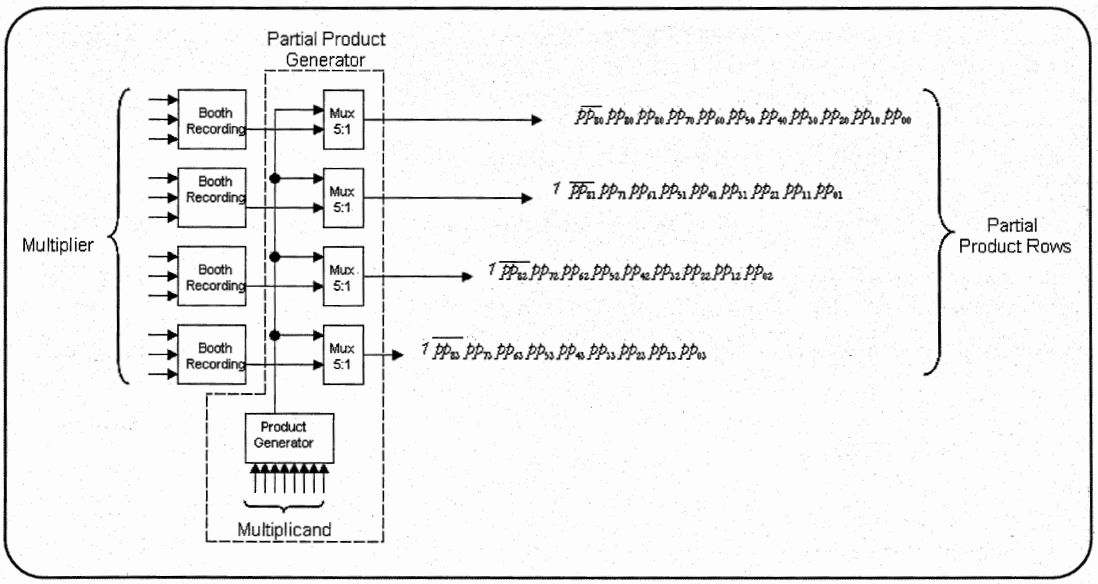

Figure 6. Multiplier architecture.

PERFORMANCE EVALUATION

In this section, the performance evaluation of the proposed design is done by comparing it with the two existing design technique in generating partial product rows as described in [7]. The investigation focused on the critical path delay and FPGA area utilization of using the proposed architecture as compared to previous methods. Table 2 depicts the simulation parameters used in this study.

Table 2. Simulation parameters.

CAD tools	: Xilinx ISE version 5.2i
FPGA Target Device	: Xilinx Virtex II Pro
Device ID	: 2vp2ff672-6
Optimization criteria	: Speed
Speed Grade	: -6

Critical Path

Table 3 shows the critical path delay for each of the related components in generating the partial product rows based on the conventional method, Kang's technique in [7] as well as the proposed architecture. In generating the partial product rows based on the conventional method, the maximum critical path delay required is about 16.423 ns whereby two main components involved which are MBE and product generator. However, the proposed design technique done by Kang in [7] will take about 17.126 ns to select the correct value for the last partial product row for up to 17-bit and this obviously more than the conventional method. Kang had not considered the delay for the MBE and 3-5 decoder because it is incurred in parallel with the delay of two's complementation (and is faster than it) and therefore masked by it.

For the proposed architecture described in this paper, there are three components known as MBE, product generator and Mux 5-1 selector required in generating the partial product. Based on the simulation results, the proposed architecture takes approximately 15.813 ns (MBE + Mux 5-1 selector) which is 0.61 ns less than conventional technique and faster 1.313 ns than the Kang's technique. Note that the delay for product generator has not been considered due to it is incurred simultaneously with the delay of MBE and in fact it is faster than it. Therefore, it has been masked by the MBE circuit. Figure 7, 8 and 9 depict the gate-level circuits for each of the components.

Table 3. Critical path delay in generating partial product rows.

Critical Path Delay Components	Design Techniques		
	Conventional Method	Kang's Method	Proposed Architecture
MBE	8.406ns	8.406ns	8.406ns
3-5 Decoder	-	7.969ns	-
Product Generator	8.017ns	-	5.727ns
Two's Complement	-	9.719ns	-
Mux 5-1 Selector	-	7.407ns	7.407ns
Total	16.423ns	17.126ns	15.813ns

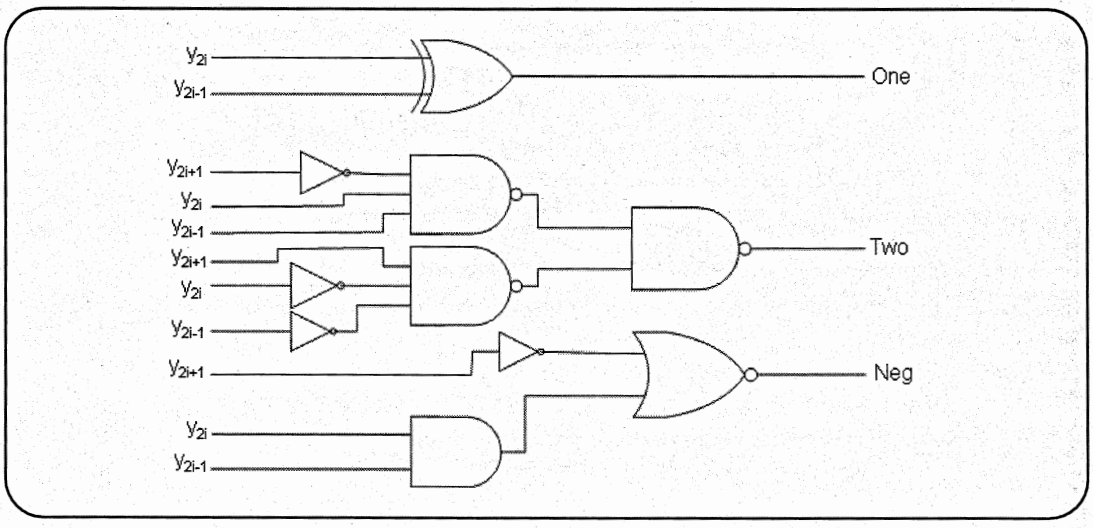

Figure 7. Gate-level diagram of MBE.

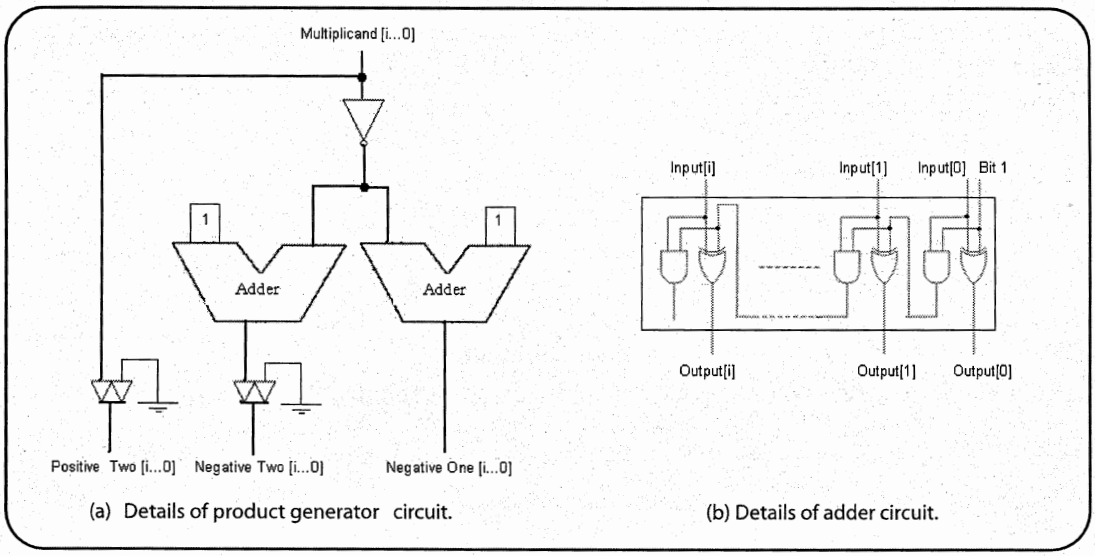

Figure 8. Gate-level diagram of product generator.

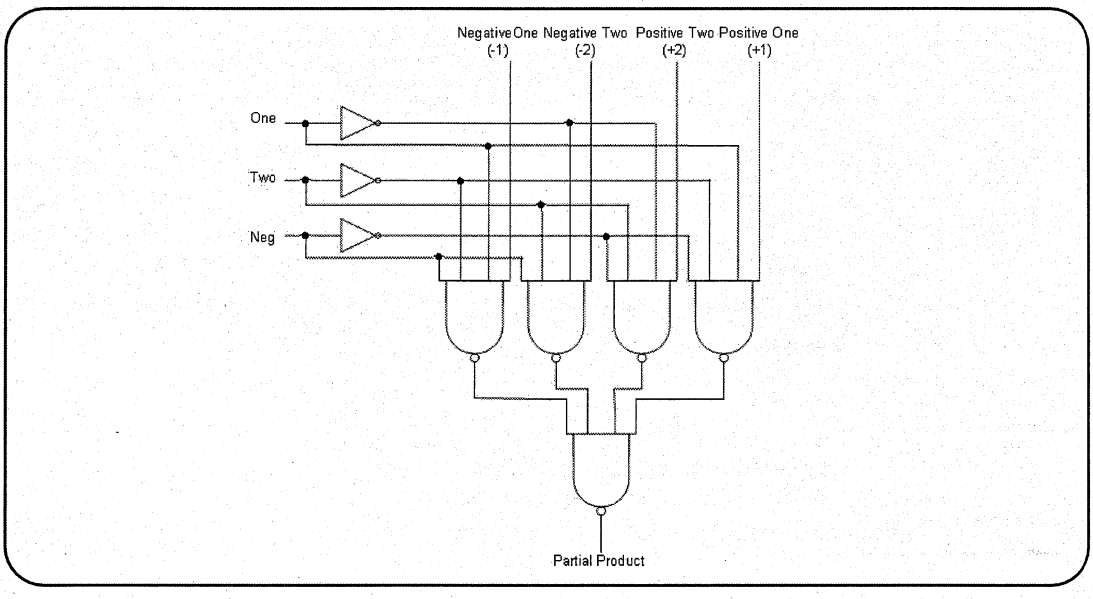

Figure 9. Gate-level diagram of Mux 5-1 selector.

Area Utilization

The evaluation of hardware requirements will be in terms of the total cell usage on Xilinx Virtex II Pro FPGA board adopted from the synthesis report using simulation parameters described in Table 2. Table 4 shows the total cell usage obtained in generating the partial product rows for each of the design technique.

Total Cell Usage	Design Techniques		
	Conventional Method	Kang's Method	Proposed Architecture
MBE	16 cells	16 cells	16 cells
3-5 Decoder	-	14 cells	-
Product Generator	10 cells	-	14 cells
Two's Complement	-	38 cells	-
Mux 5-1 Selector	-	14 cells	14 cells
Total	26 cells	82 cells	44 cells

Table 4. Total cell usage in generating the partial product rows.

It turns out that the proposed method requires fewer cells in total compared with the Kang's method in generating partial product rows. Although it consumes more cells than the conventional method, it still considered less area usage due to the proposed method does not required an additional partial product row whilst performing the addition operation whereby the conventional method still required one extra partial product row. For that particular extra row, it is actually consumes the same amount of hardware of two full-adder circuits. Thus, by implementing the proposed system, not only the speed advantage but also area utilization may be optimized and well balanced for any multiplication sizes. As a conclusion, having one additional partial product row as well as the use of error correction circuits not only brings to performance degradation but also more hardware usage.

CONCLUSION

In this paper, we have presented the partial product generator architecture which capable to reduce the number of partial product rows from $(n/2) + 1$ to $n/2$ 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, its not only can reduced the area of the circuit but also at the same time increase the speed of the system due to the major source of delay in adders is consumed from the carries numbers [16].

In order to completely validate this method, future work would be to design the fast adders which can perform the final accumulation process based on the partial product rows generated. From this, a complete design of multiplier can be produced and hence could be evaluate its performance and compare it with other design approaches.

REFERENCES

1. A. G. D. Oscar Gustafsson, L. W. (2001). Multipliers Block Using Carry Save Adders. *Proc. IEEE Int. Symp. Circuits Syst.*, 472-476.

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

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

4. Ercegovac, T. L. M. D. (2003). *Digital Arithmetic.* California, USA: Morgan Kaufmann Publishers.

5. Itoh, Y. H. N. (2001). A 600 MHz 54x54-bit Multiplier with Rectangluar Styled Wallace Tree. *IEEE Journal of Solid State Circuits*, vol. 36(No. 2), 249-257.

6. J. H. J. Kenneth Lin, N. T. (2003). A High-Performance 32-bit Parallel Multiplier Using Modified Booth Algorithm and Sign Deduction Algorithm. *IEEE Proceedings, vol. 0-7803-7889-X,* 1281-1284.

7. Kang, J. L. G. J. Y. (2004). A Fast and Well-Structured Multiplier. *EUROMICRO Systems on Digital System Design,* 692-701.

8. Kung, S. Y. (1998). *VLSI Array Processors.* New York: Prentice Hall.

9. MacSorely, O. L. (1990). High Speed Arithmetic in Binary Computation. *IEEE Proceedings, vol.* 49, 67-91.

10. Othman, A. M. A. L. M. (2002). High Performance Parallel Multiplier Using Wallace-Booth Algorithm. *ICSE2002 Proc. Penang,* 433-436.

11. Robertson, J. E. (1955). Two's Complement Multiplication in Binary Parallel Digital Computers. *IRE Trans., vol. EC-4,* 118-119.

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

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

14. T. K. William Jao, S. T. (1998). Circuit Optimization Using Carry Save-Adder Cells. *IEEE Transactions on Computers-Aided Design of Integrated Circuits and Systems, vol. 17,* 974-984.

15. Wallace, C. S. (1964). A Suggestion for a Fast Multiplier. *IEEE Transactions on Computers,* 14-17.

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