Reducing the Computation Time in Two's Complement Multipliers

A. Hari Priya[1]
[1]Assistant Professor, Dept. of ECE,
Indur Institute of Engineering. And Technology, Siddipet, Medak (D)India.

Abstract

To reduce the area of partial product array size and improve the speed which is generated by a radix-4Modified Booth Encoded Multiplier is used. This reduction is possible without any increase in the delay of the partialproduct generation stage. This reduction provides faster compression of the partial product array and regular layoutsin two's complement multiplier. The proposed method is that the Radix-4 (Fixed-Width) Modified Booth Multipliersare used to achieve the low power and increase the speed by modifying the partial product matrix size. The Multiplierdesign implemented using Xilinx. The results based on a rough theoretical analysis and on logic synthesis showed itsefficiency in terms of both area and delay. It is compared with Radix-4 (short bit-width) Modified booth encoded Multiplier.

Keywords: *Multiplication, Modified Booth encoding, partial product array.*

I. INTRODUCTION

The speed of multiplication operation is of prodigioussignificance in digital signal processing also in the general purpose processors today. In the past multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition produces a partial product. In utmost computers, the operand generallycomprises the same number of bits. Modified Booth Encoding (MBE) 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 regularly used schemes is radix- 4 MBE, for a number of whys and wherefores, the most important actuality that it allows for the reduction of the size of the partial product array by almost half, and it is very simple to produce the multiples of the multiplicand. More explicitly, the classic two's complement n×n bit multiplier using the radix- 4 MBE scheme, generates a PP array with a maximum height of [n/2]+1rows, each row before the last one being one of the succeeding possible values: all zeros, ±X; ±2X.

Digital multiplication comprises of three basic steps these are:-

1. Generation of Partial Product Array
2. Reduction of Partial Product Array
3. Final Addition

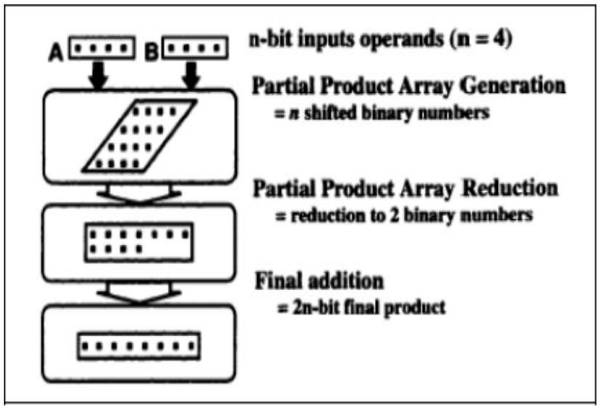

Fig 1. Multiplication Flow

II. LITERATURE SURVEY

The mainmoto of a good multiplier is to deliver a physically compact, good speed and low power consuming chip. Previously lot of research doings has been stated on the multiplier. Each and every multiplier technique has their exceptional methodologies and implementations. Among them some of the multiplier techniques and their implementations are discussed in this literature survey.In High-Speed Multiplier Design, an algorithm to accomplish fast multiplication in two's complement representation is presented. Fewer partial products rows generate rather than concentrating on reducing the partial products rows down to final sums and carries. Consecutively, this influences the speed of the multiplication, even before applying partial products reduction techniques. Fewer partial products rows are produced, thereby lowering the overall operation time. Furthermore to the speed improvement, the algorithm results in a real diamond-shape for the partial product tree, which is more resourceful in terms of implementation. [1].

The multiplier of an S/390 CMOS microprocessor is defined. It is implemented in an critical static CMOS technology with 0.20 pm effective channel length. The multiplier has been proved in a single-image shared-memory multiprocessor atfrequencies up to 400 MHz. The multiplier requires three machine cycles for a total latency of 7.5 ns. However, the design can sustenance a latency of 4.0 ns if the latches are removed. The design goal was to implement a versa- tile S/390 multiplier with sensible performance at a very aggressive cycle time. The multiplier implements a radix-8 Booth algorithm and

is capable of supporting S/390 floating-point and fixed-point multiplications and also division and square root. Logic design and physical design issues are debated relating to the Booth decode and counter tree implementations. [2].

In high speed Booth encoded parallel multiplier, for partial product generation, a new modified Booth encoding (MBE) scheme is proposed to improve the performance of outdated MBE schemes. For final addition, a new algorithm is established to construct multiple-level conditional-sum adder (MLCSMA). The proposed algorithm can optimize final adder according to the given cell properties and input delay profile. Related with a binary tree based conditional- sum adder, the speed performance improvement is up to 25percent. On average, the design developed here in reduces the total delay by 8 percent for parallel multiplier. The whole design has been verified by gate level simulation. [3]

A high-performance low-power design of linear array multipliers is based on a combination of the following techniques: signal flow optimization in [3:2] adder array for partial product reduction, left-to-right leap frog (LRLF) signal flow, and splitting of the reduction array in to upper/lower parts. The resulting upper/lower LRLF (ULLRLF) multiplier is compared with tree multipliers. The automatic layout experiments results ULLRLF multipliers having similar power, delay, and area as tree multipliers for n<=32. With more regularity and inherently shorter interconnects, the ULLRLF structure grants a inexpensive alternative to tree structures in the design of fast low-power multipliers implemented in deep submicron VLSI technology.[4]

In the data path synthesis from RTL, data paths are extracted into largest possible sum-of-product (SOP) blocks, thus making extensive use of carry-save intermediate results and reducing the number of expensive carry-Propagations to a minimum. The sum-of-product blocks are then implemented by constraint- and technology driven generation of partial products, carry-save adder tree and carry-propagate adder.

A smart generation feature selects the best among alternative implementation variants. Special data path library cells are used where available and beneficial. All these measures translate into better performing circuits for simple and complex data paths in cell-based design. [5].A parallel multiplier, which is optimized for speed is valid to any multiplier size and adjustable to any technology for which speed parameters are known. Most significantly, it is easy to include this method in silicon compilation or logic synthesis tools. The parallel multiplier produced by the proposed method outdoes other schemes used for. It uses the minimal number of cells in the partial product reduction tree. These findings are tested on design examples simulated in LP CMOS ASIC technology. [6].

III. PROPOSED ALGORITHM

A. Modified Booth Multiplier

Modified Booth Encoding (MBE) 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 sufficient. In this the bits can be encoded by looking at three bits at a time. One of the most normally 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 modest to generate the multiples of the multiplicand. Further specifically, the classic two's complement n * n bit multiplier using the radix-4 MBE scheme, generates a PP array with a maximum 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 is due to the negative encoding.

In Modified Booth technique, Booth Recording table is used. By this we can be able to add multiplicands -2, -1, 0, 1 and 2 times and have got rid of 3's hence generating partial products is simple.

i+1	i	i-1	Partial product
0	0	0	0*M
0	0	1	1*M
0	1	0	1*M
0	1	1	2*M
1	0	0	–2*M
1	0	1	–1*M
1	1	0	–1*M
1	1	1	0*M

Table 1. Modified Booth Encoding Table

The steps involved in the generation of partial products using Modified Booth algorithm are given below:

1. Pad the LSB with one zero.
2. Pad the MSB with 2 zeros if n is even and 1 zero if n is odd.
3. Divide the multiplier into overlapping groups of 3-bits.
4. Determine partial product scale factor from modified booth 2 encoding table.
5. Compute the Multiplicand Multiples
6. Sum Partial Products

MBE is efficient technique in reducing the partial products. However, it is important to note that there are two unavoidable consequences when using MBE: sign extension prevention and negative encoding. The combination of these two unavoidable consequences results in the formation of one additional partial product row and of course, this additional partial product row requires more hardware but more importantly increases delay. To overhead the formation of one additional partial product row a technique called short bit width two's complement multiplier is designed.

First of all pad a bit 0 in LSB and 0 or 1 in MSB depending upon the n*n multiplier. If n is even pad two 0's otherwise pad one 0. In this project 8*8 multiplier is used, so pad two 0's in MSB. For the generation of partial products using MBE signals we take 3 bits (y_{2i+1}, y_{2i}, y_{2i+1}) at a time and stride two bits each time.In the partial product generation, except for the first row all the three rows are generated in same way. This is done due to two reasons. One is to reduce a row for sign handling (neg3 added to first row) and other is the y_{-1} bit used by the MBE is always equal to zero.

The first row is split into two parts, one containing partial product bits from pp00 to pp80 bar and the second one with two bits set at "one" in positions 9 and 8 as shown in below fig.2. This is done so that the bit neg3 related to fourth partial product row can be moved to become a part of the second sub row and by this computation time can be reduced.In these way four partial products rows can be generated.

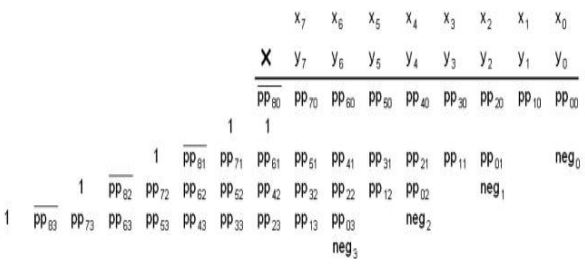

Fig.2 Representation of partial products and neg bits.

The detailed view of these partial product generations is given below.

B. Partial Products Generation

The partial products are generated using gate level logic unlike in previous techniques. Though this increase the area, it reduces the computation time. The partial products are generated in the following way.

Step 1: Generation of the three most significant bit weights of the first row, plus addition of the last neg bit

Step 2: Generation of the other bits of the first row
Step 3: Generation of the bits of the other rows

The above three steps are explained in detail:

STEP 1: The three most significant bits are generated by using gate level diagram shown in fig.3. Here x_j,x_{j-1}, y_0 and neg_0 bit are given as inputsand the output pp_{0j} is obtained. By using this gate level diagram bits pp_{06}, pp_{07}, pp_{08} bar are individually generated in three iterations.

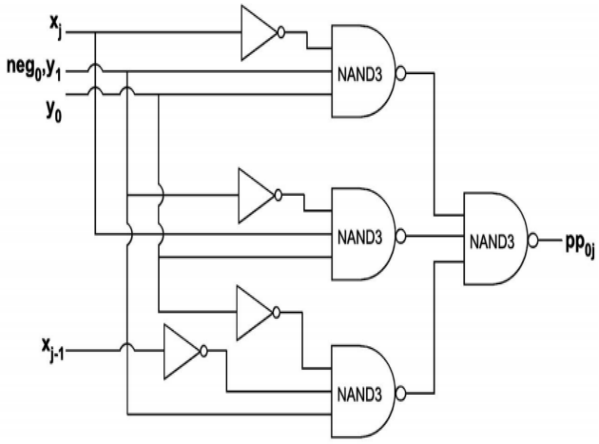

Fig 3. Gate-level diagram for generation of three MSB's

The addition of last neg bit to the second sub part of first row is shown in the fig.4. The bit neg3 related to the fourth partial product row, is moved to become a part of the second sub row. The second sub row can be easily added to the first sub row, with a constant short carry propagation of three. In fact, with reference to the notation of fig.5 it can be given as:

$$\overline{qq90}qq90\ qq80\ qq70\ qq60 = 00\ \overline{pp80}\ pp70\ pp60 + neg3$$

As introduced above, due to the particular value of the second operand, i.e., 0 1 1 0 neg3 requires a carry propagation only across the least-significant three positions. By this the five most significant bits of resulting array in the first row are obtained

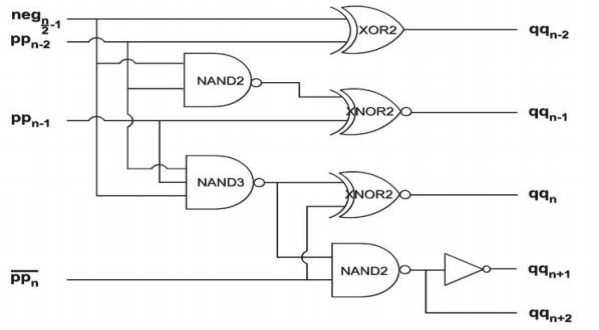

Fig.4 Gate-level diagram for adding last neg bit in the first row

The graphical transformation is that the second sub row containing also the bit neg3 is shown in fig 5 (a) and the resulting array of partial product is shown in fig 5 (b).

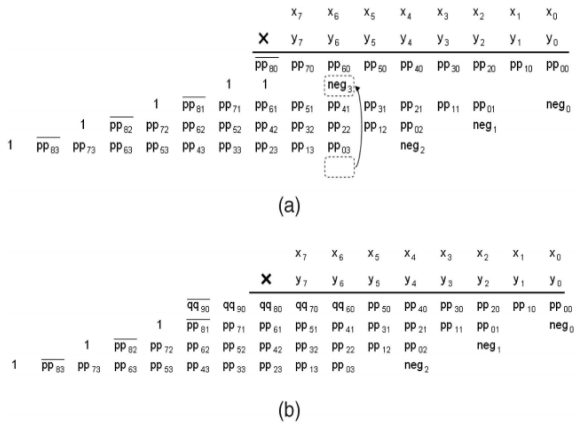

Fig.5 Partial product array after adding last neg bit to the first row (a) Basic idea (b) resulting array

3.1.2 STEP 2: The other bits of the first row are generated by using fig 6.By using fig 6 (a) one, two, neg signals can be generated. These signals are then exploited by the logic in fig 6 (b) along with appropriate bits of multiplicand, in order to generate whole partial product array. Hence by this partial product bits from pp_{00} to pp_{50} are produced

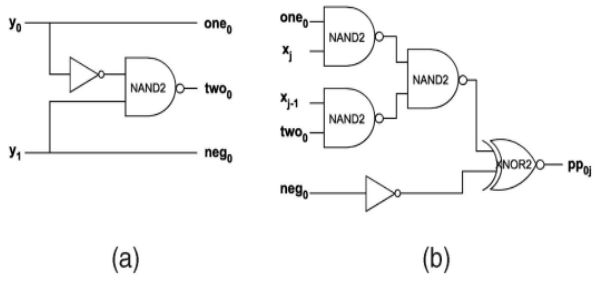

Fig 6 Gate-level diagram for first row partial product generation (a) MBE signal generation (b) partial product generation

STEP 3: 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$\overline{qq90}$ $qq90$ $qq80$ $qq70$ $qq60$.In order to achieve this, we must simplify and differentiate the generation of the first row with respect to the other rows. This can be done by using fig 7.For each of the partial product row, fig 7(a) produces the one, two, and neg signals. These signals are then exploited by the logic in fig 7(b), along with the appropriate bits of the multiplicand, in order to generate the whole partial product array.

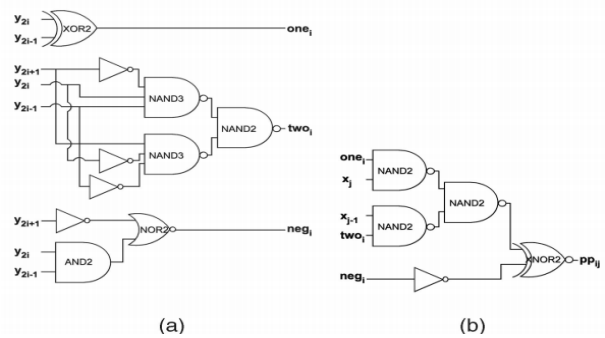

Fig 7 Gate-level diagram for partial product generation using MBE (a) MBE signal generation (b) partial product generation

By direct comparison of figs 7 and 6, 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. The above three step are independent to each other and implemented in parallel. By this way computation time can be reduced to a great extent.

IV. RESULT AND DISCUSSION

RTL SCHEMATIC: The RTL Diagram of the two's compliment multiplier was given in fig8.The inputs M_r and M_d are the multiplier and the multiplicand with input data of 8 bits. k_1, k_2, k_3, k_4 are partial product arrays and k_m is final output.

Fig8 RTL Diagram of the two's compliment multiplier

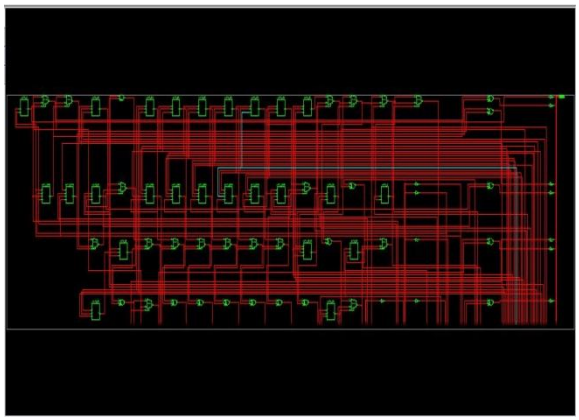

Fig 9 RTL Schematic

The final output of short bit width multiplier is K_m and k_1, k_2, k_3 and k_4 partial product rows.

Simulation result of short bit width two's complement multiplier is shown in fig 10.

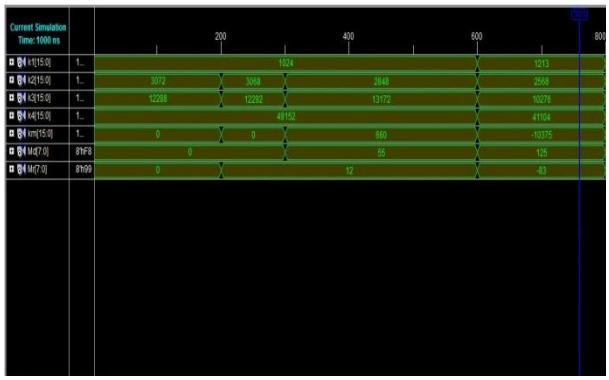

Fig 10Simulation result of short bit width multiplier

A. Final synthesis report

Device utilization summary

Selected Device: 3s500ecp132-5

Number of Slices	59 out of	4656	1%
Number of 4 input LUTs	104 out of	9312	1%
Number of IOs	96		
Number of bonded IOBs	96 out of	92	104% (*)

Timing Summary

Speed Grade: -5

Minimum period: No path found

Minimum input arrival time before clock: No path found

Maximum output required time after clock: No path found

Maximum combinational path delay: 14.939ns

Cell usage

Total memory usage is 188976 kilobytes.

V. CONCLUSION

Two's complement n×n multipliers using radix-4 Modified Booth Encoding produce [n/2] partial products but due to the sign handling, the partial product array has a maximum height of [n/2] + 1. To evade this and yield a partial product array with a maximum height of [n/2], without introducing any extra delay in the partial product generation stage short bit width technique is employed. With the additional hardware of a (short) 3-bit addition, and the simpler generation of the first partial product row, the delay is reduced which is within the bound of the delay of a standard partial product row 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, both in terms of delay and uniformity of the layout.

REFERENCES

[1] J.-Y. Kang and J.-L. Gaudiot, "A Simple High-Speed Multiplier Design," IEEE Trans. Computers, vol. 55, no. 10, pp. 1253-1258, Oct.2006.

[2] E.M. Schwarz, R.M. Averill III, and L.J. Sigal, "A Radix-8 CMOS S/390 Multiplier," Proc. 13th IEEE Symp. Computer Arithmetic, pp. 2-9, 1997.

[3] W.-C. Yeh and C.-W. Jen, "High-Speed Booth Encoded Parallel Multiplier Design," IEEE Trans. Computers, vol. 49, no. 7, pp. 692-701, July 2000.

[4] Z. Huang and M.D. Ercegovac, "High-Performance Low-Power Left-to-Right Array Multiplier Design," IEEE Trans. Computers, vol. 54, no. 3, pp. 272-283, Mar. 2005.

[5] R. Zimmermann and D.Q. Tran, "Optimized Synthesis of Sum-of-Products," Proc. Conf. Record of the 37th Asilomar Conf. Signals, Systems and Computers, vol. 1, pp. 867-872, 2003.

[6] V.G. Oklobdzija, D. Villeger, and S.S. Liu, "A Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach," IEEE Trans. Computers, vol. 45, no. 3, pp. 294-306, Mar. 1996.

[7] P.F. Stelling, C.U. Martel, V.G. Oklobdzija, and R. Ravi, "Optimal Circuits for Parallel Multipliers," IEEE Trans. Computers, vol. 47, no. 3, pp. 273-285, Mar. 1998.

[8] R. Hashemian and C.P. Chen, "A New Parallel Technique for Design of Decrement/Increment and Two's Complement Circuits," Proc. 34th Midwest Symp. Circuits and Systems, vol. 2, pp. 887-890, 1991.