Design and Implementation of High Performance Two's Complement Multiplier

Sandeep Reddy D
M.Tech VLSI II YEAR
VIT University, Vellore, Tamilnadu, India

Venkatesh J
M.Tech VLSI II YEAR
VIT University, Vellore, Tamilnadu, India

Karthik P
M.Tech VLSI II YEAR
VIT University, Vellore, Tamilnadu, India

Abstract- The multiplication operation is present in many parts of a digital system or digital computer, most notably in signal processing, graphics and scientific computation. With advances in technology, various techniques have been proposed to design multipliers, which offer high speed, low power consumption and lesser area. Thus making them suitable for various high speeds, low power compact VLSI implementations. These three parameters i.e. power, area and speed are always traded off. In this paper high performance two's compliment square multiplier (number of bits in multiplier and multiplicand are equal) algorithm is presented to reduce the total number of partial product (PP) rows generated, almost by half to achieve fast multiplication. By using sign extension prevention techniques the width of each partial product is kept under control which helps in reducing the total hardware implementation required. In this paper we also compared this technique with other multipliers in terms of area, delay and power..

Keywords – Multiplication, Modified Booth Encoding, Partial Product array.

I. INTRODUCTION

Multiplier performance is very important in many multimedia applications, signal processing systems, 3D graphics as more number of multipliers is involved in these environments. More specifically multipliers that have single cycle throughput became building blocks of high performance digital processors [1]. To meet its requirements multipliers should be fast enough to give output in the required cycle time.

Three major steps are involved in any multiplication [2]. In the first step partial products are produced. In phase of PP generation, a set of rows is generated where each product in the row is the result of the product of one bit of the multiplier by multiplicand. For example, consider the multiplication of X × Y with both X and Y having n bits which is of the form $x_{n-1}...x_0$ and $y_{n-1}...y_0$, then the i^{th} row in general is proper left shifting of $y_i \times X$, i.e., either a string of all zeros when $y_i = 0$, or the multiplicand X itself when $y_i = 1$. In this case, the number of Partial Product rows generated during the first phase is n. In second step partial product reduction is done. Using compression tree we can add all the partial product rows thus reducing number of Partial product rows [3] [4]. In last step final sums and carries are added to generate the result. Modified Booth Encoding (MBE) is a technique used to reduce total number of partial products generated by almost half [5] [6]. Radix-4 MBE is widely used because it reduces total number of partial product row by almost half while still keeping the generation of each partial product row fast and simple. If 'n' is the total number of bits in multiplier, radix-4 MBE technique reduces these partial product rows by "$\lceil \frac{n}{2} \rceil$ +1" [10]. All the partial products except the last one can take any of the following values: all zeroes, ±X, ±2X where X is multiplicand..

II. MODIFIED BOOTH ENCODED MULTIPLIER

Radix-B=2^b MBE reduces the number of PP rows to $\lceil \frac{n}{b} \rceil$ but it needs to generate multiples of multiplicand (X) from (- [$\frac{B}{2}$] X to + [$\frac{B}{2}$] X). For radix-4=2^2 reduces PP rows to $\lceil \frac{n}{2} \rceil$ and requires ±X, ±2X in the generation of PP rows. The value of 2X can easily be obtained just by left shifting X by one bit. By using higher radix MBE we can still reduce the number of PP rows but more multiples of X are need to be generated which makes it difficult to get each PP row. In radix-4 MBE multiplier is scanned by three bit window and stride off two bits. Each group of three bits (Y_{2i+1}, Y_{2i}, Y_{2i-1}) is associated with only one partial product row by using Table 1.

Table - 1 Modified Booth Encoding (radix-4)

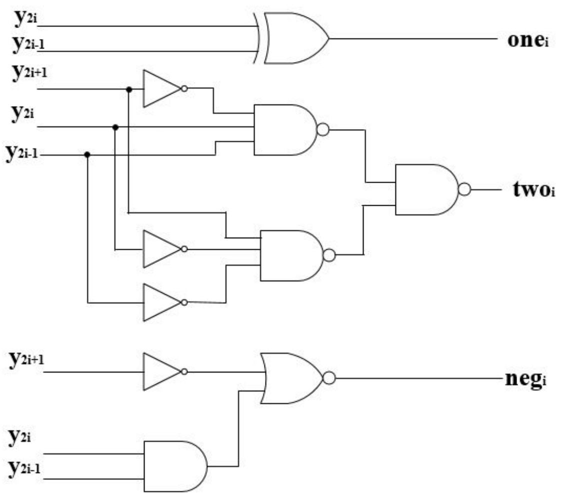
Y_{2i+1}	Y_{2i}	Y_{2i-1}	General partial products
0	0	0	0 ×X
0	0	1	1 × X
0	1	0	1 × X
0	1	1	2 × X
1	0	0	(-2) × X
1	0	1	(-1) × X
1	1	0	(-1) × X
1	1	1	0 × X

A possible implementation of MBE signals and generation of PP rows from these MBE signals are shown in Figure 1[9]. For each PP row signals one, two, negative signals are generated shown in Fig.1 (a). These signals along with appropriate bits of multiplicand which is shown in Fig.1 (b) are used to generate whole PP array.

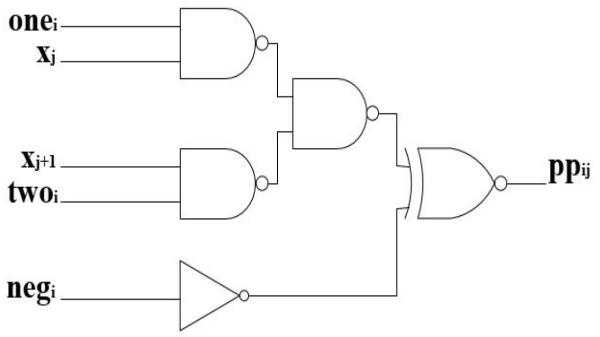
MBE generates only $\lceil \frac{n}{2} \rceil$ PP rows. However a total of $\lceil \frac{n}{2} \rceil$ +1 PP rows are generated due to last neg signal. To avoid negative encoding -X and -2X two's compliment form is used. Here we first generate one's compliment of X and 2X and then add negative signal to generate two's compliment signals. Two's complement required for negative (neg) signals should be added in Least Significant Bit of each PP row to produce two's complement [5] which is shown in Figure 2.

A. Sign extension and prevention-

The use of two's compliment requires extension of sign bit to left most part of PP row with the consequence of extra overhead. Also Sign Extension makes the length of each partial product row unequal. For 8×8 multiplier the length of first PP row are 16 bits, for second row it is 14 bits, the third row 12 and fourth 10. A number of techniques are provided to prevent sign extension. One such method is based on the observation as [-p= (1-p)-1=p'-1] [2] [6].



(a)



(b)

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

$$\begin{array}{r}
x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0 \\
*\ \ y_7\ y_6\ y_5\ y_4\ y_3\ y_2\ y_1\ y_0 \\
\hline
\overline{pp_{80}}\ pp_{80}\ pp_{80}\ pp_{70}\ pp_{60}\ pp_{50}\ pp_{40}\ pp_{30}\ pp_{20}\ pp_{10}\ pp_{00}\qquad\qquad \\
1\ \ \overline{pp_{81}}\ pp_{71}\ pp_{61}\ pp_{51}\ pp_{41}\ pp_{31}\ pp_{21}\ pp_{11}\ pp_{01}\quad\ \ neg_0\quad\ \ \\
1\ \ \overline{pp_{82}}\ pp_{72}\ pp_{62}\ pp_{52}\ pp_{42}\ pp_{32}\ pp_{22}\ pp_{12}\ pp_{02}\quad\ \ neg_1\qquad\qquad\qquad \\
1\,\overline{pp_{83}}\ \ pp_{73}\ \ pp_{63}\ pp_{53}\ pp_{43}\ pp_{33}\ pp_{23}\ pp_{13}\,pp_{03}\quad\ \ neg_2\qquad\qquad\qquad\qquad\qquad\qquad \\
neg_3\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad
\end{array}$$

Figure 2. Application of the sign extension prevention measure on the partial product array of an 8×8 radix-4 MBE multiplier

B. Reduction of additional partial product row –

Inclusion of last negative signal adds one more partial product which adds an additional delay of one carry save adder before getting sum and is carried out just before final accumulation. This additional delay of one carry save adder is more critical for multipliers of small words than with longer words because of relatively higher delay effect that this additional row brings.

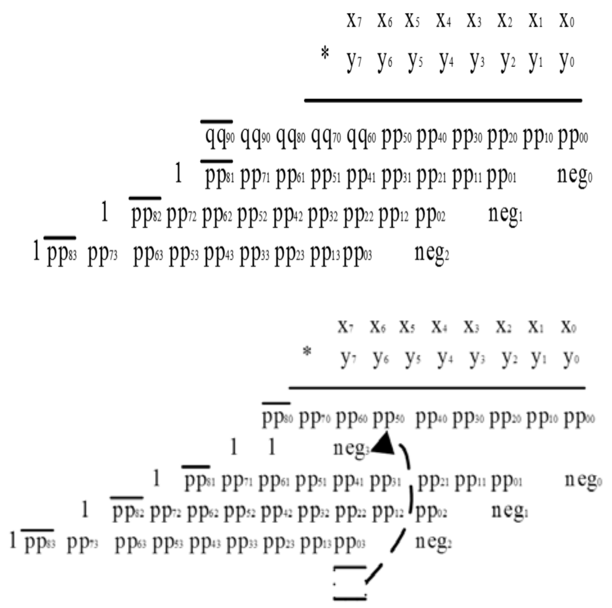
Therefore our goal is to remove last negative signal responsible for additional Partial Product Row which is responsible for additional delay. Somehow if we could directly produce two's compliment of multiplicand while producing PP rows then there will not be any need of last negative signal as it is already applied when generating twos compliment of operand as shown in Figure. 3[6].

$$\begin{array}{r}
x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0 \\
*\ y_7\ y_6\ y_5\ y_4\ y_3\ y_2\ y_1\ y_0 \\
\hline
\overline{pp_{80}}\ pp_{80}\ pp_{80}\ pp_{70}\ pp_{60}\ pp_{50}\ pp_{40}\ pp_{30}\ pp_{20}\ pp_{10}\ pp_{00} \\
1\ \ \overline{pp_{81}}\ pp_{71}\ pp_{61}\ pp_{51}\ pp_{41}\ pp_{31}\ pp_{21}\ pp_{11}\ pp_{01}\quad\ \ neg_0 \\
1\ \ \overline{pp_{82}}\ pp_{72}\ pp_{62}\ pp_{52}\ pp_{42}\ pp_{32}\ pp_{22}\ pp_{12}\ pp_{02}\quad\ \ neg_{10}\qquad\quad \\
s_9\ s_8\ s_7\ \ s_6\ \ \ s_5\ \ s_4\ \ \ s_3\ \ \ s_2\ \ \ s_1\ \ s_0\quad\ \ neg_2\qquad\qquad\qquad\quad
\end{array}$$

Figure 3. Partial Product array by applying the two's complement computation method in to the last row.

Therefore a fast method to calculate two's compliment of a number is needed. Two's complementation compliments only the bits after right most one in the word while keeping the other words as it is. Therefore in two's compliment only selectively complimenting the bits after some bits is enough.

Another approach [9] to reduce $\lceil \frac{n}{2} \rceil$ +1 row is to temporarily considering the first row split into two rows, the first one containing actual first row partial product bits obtained from MBE (pp_0 to pp_8) with last bit in complimented form and the second row containing ones in positions 8 and 9. Then the negative signal related to the last partial product is bought to position 6 of second row as shown in Figure. 4 (a). The two rows are then added together which involves addition of only four most significant bits. The result obtained replaces the actual first row partial product as shown in Figure. 4 (b). Gate-level diagram of the proposed method for adding the last negative bit in the first row is shown in the Figure. 5.



(a) (b)

Figure 4. Partial Product arrays after adding the last negative bit to the first row. (a) Basic idea, (b) Resulting array

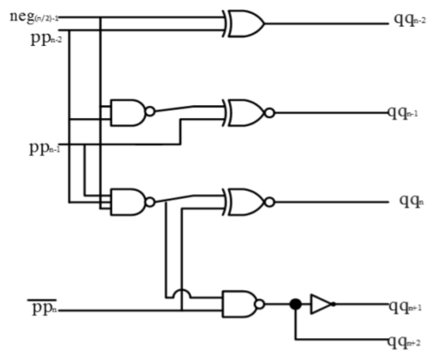


Figure 5. Gate-level diagram of the proposed method for adding the last negative bit in the first row.

III. EVALUIATONS AND COMPARISIONS

In this section, the proposed method based on the addition of the last negative signal to the first row is first evaluated. The designed architecture is then compared with an implementation based on the computation of the two's complement of the last row (referred "Two's complement" method). In this analysis, the standard MBE implementations for the first and for partial product row are also taken into account as summarized in Table 2.

Table 2. Designs for the generation of PP rows considered in the evaluation

Type of Implementation	Total Power (uW)	Area (um^2)	Delay (ns)
Ordinary multiplier	20.817	398	4.07
Modified booth multiplier	32.873	427	3.43
Proposed method	34.78	402	3.13

IV. RESULTS

In order to check the validity of our estimations in implementation technology, we implemented the designs in Table 3 through logic synthesis and technology mapping to an industrial standard 45nm cell library. To perform the evaluation power, area and delay of various methods of multiplication are compared. Proposed method of multiplication produced minimum delay with considerably less area.

Table 3. Comparative Evaluation of various methods of multiplication

Implementation	**Description**	**Motivation**
Standard multiplier	**Standard implementation of MBE: Signals Zero, One, negative are generated first and are used in PP rows generation.**	**The delay to produce PP rows and their reduction limits the speed of any multiplier. By reducing number of PP rows by still keeping the delay to produce each product low, speed of implementation can be increased.**
Proposed method	**Generation of first PP row**	**The main aim is to reduce number of PP rows from $\lceil \frac{n}{2} \rceil$ +1 to $\lceil \frac{n}{2} \rceil$. By reducing PP rows reduction hardware can be smaller in size and faster. This is achieved by including the effect of last negative signal into the first row.**
Two's Compliment	**Direct computation of last partial product in two's compliment form while computing other PP rows in parallel.**	**This method avoids extra partial product row and its delay has to be within the delay requirements of standard PP rows. The above goal is achieved by directly implementing last PP row in two's complimented format eliminating the need for last negative signal.**

Simulation results for the multiplier as shown in Figure.6. After performing synthesis for the RTL code of proposed multiplier, giving netlist(.v) file, Synopsys Design Constraint(.sdc) file, technology library, Capacitance (Cap) table file and I/O pin assignment file are given as input files to design system on Chip (Back End).

In the process of checking timing report, for Clock Tree Synthesis (CTS) there should be no timing violations in the design, if there are violations remove those violations by changing the Verilog code or rectifying those violations using reports in the summary report. After completing the entire steps final back end design chip layout is as shown in Figure. 7.

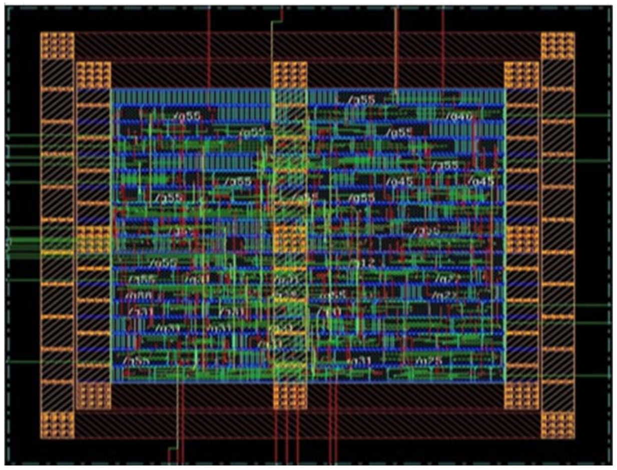
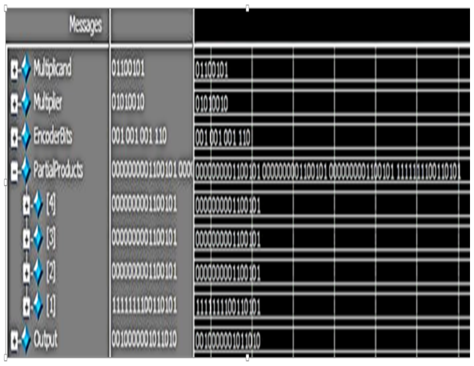


Figure 6. Simulation results



```
Messages
Multipicand     01100101                  01100101
Multipier       01010010                  01010010
EncoderBits     001 001 001 110           001 001 001 110
PartialProducts 0000000001100101 0000     0000000001100101 0000000001100101 0000000001100101 1111111100110101
  [4]           0000000001100101          0000000001100101
  [3]           0000000001100101          0000000001100101
  [2]           0000000001100101          0000000001100101
  [1]           1111111100110101          1111111100110101
Output          0010000001011010          0010000001011010
```

Figure 7. Physical layouts for MBE Radix-4 multiplier design

REFERENCES

[1] SS.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.

[2] M.D. Ercegovac and T. Lang, "*Digital Arithmetic"*. Morgan Kaufmann Publishers, 2003.

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

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

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

[6] A. D. Booth, "*A signed multiplication technique*," Quarterly J. Mech. Appl. Math., vol. 4, 1951.

[7] D. P. Agrawal and T. R. N. Rao. "*On multiple operand addition of signed binary numbers,*". IEEE Transactions on Computers, C-27:1068–1070, November 1978.

[8] 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.

[9] F. Lamberti, N. Andrikos, E. Antelo, and P. Montuschi,
"*Speeding-Up Booth Encoded Multipliers by Reducing the Size of Partial Product Array*," internal report, http://arith.polito.it/ir_mbe.pdf, pp. 1-14, 2009.

[10] 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.