# General Algorithms for a Simplified Addition of 2's Complement Numbers

O. Salomon, J.-M. Green, and H. Klar

*Abstract*—**Two algorithms for both a simplified carry save and carry ripple addition of 2's complement numbers are presented. The algorithms form the partial products so that they exclusively have positive coefficients which eliminates the need for the common sign bit extension. This results in a reduction of circuit area by up to six full adders per row of adders when partial products are added in an $N/2$ or Wallace tree. Furthermore, the capacitive load of the intermediate sum and carry sign bit signals decreases by up to a factor of seven which leads to an appropriate reduction of delay. Although the algorithms are derived for multipliers they can always be applied to appropriate adder circuits.**

$$
\begin{array}{rccccccc}
 & a_5 & a_5 & a_5 & \ldots a_1 & a_0 & & \\
+ & b_5 & b_5 & b_4 & \ldots b_0 & & & \\
\end{array}
\quad \Leftrightarrow \quad
\begin{array}{rccccccc}
 & 1 & \overline{a_5} & a_5 & \ldots a_1 & a_0 & & \\
+ & & 1 & & & & & \\
+ & 1 & \overline{b_5} & b_4 & \ldots b_0 & & & \\
+ & & 1 & & & & & \\
\hline
 & & \overline{a_5} & a_5 & \ldots a_1 & a_0 & & \\
+ & 1 & \overline{b_5} & b_4 & \ldots b_0 & & & \\
\hline
\end{array}
$$

Fig. 1. Example of the addition of two numbers $A$ and $B$ with $k = 6$ and $S = 1$.

## I. Introduction

MULTIPLIERS are one of the most important components in digital signal processing applications. The add and shift algorithm in connection with the 2's complement number representation is a common multiplication method which performs a sequential addition of partial products according to their bit significances.

The demand for high-speed multipliers has lead to the development of several algorithms speeding up the multiplication delay [1]–[4], [6], [7]. To reduce the number of partial products to be added the modified Booth algorithm can be employed which reduces the number of partial products by a factor of two [7]. Further speed improvements are possible by reducing the number of sequential adding stages as it is done in parallel adding trees like the $N/2$ [3], [4] or the Wallace tree [1]. With increasing parallelism, the amount of shifts between the partial products and intermediate sums to be added tends to increase. Two problems arise. Firstly a less regular layout is achieved, secondly and even worse, when obeying the rules of 2's complement arithmetic the sign bits of the numbers to be added have to be extended up to the most significant bit of the expected sum. This leads to a multiple and redundant addition of the most significant bits in each row of adders which further on leads to additional full adders per row. This slows down the circuit's speed and increases the layout area.

In this paper, two new algorithms for adder circuits are presented so that the common multiple sign bit addition does not have to be performed when either a sign bit extension is added in half adders or when two sign bit extensions are added in half or full adders at the same bit order. When partial products are added in carry save arithmetic the circuit simplification fully eliminates the redundant addition of the most significant bit so that all adder cells of the multiplier have the same load. This leads to an optimal transistor sizing and to an appropriate area reduction. In carry ripple additions one full adder per row is eliminated.

After an introduction of several multiplier algorithms in the following section a modified 2's complement number representation is derived in Section III. In Section IV the general algorithms for the simplified addition of 2's complement numbers are shown for carry ripple and carry save arithmetic, respectively. The simplifications are demonstrated for a 6 × 6 b carry ripple, a 6 × 6 b carry save and an 8 × 8 b Booth multiplier. These results are summarized in Section V.

## II. Additions in Array Multipliers

Most digital multiplier circuits are based on the add and shift algorithm. Each partial product is calculated by the multiplication of the multiplicand with one multiplier bit. The partial products are shifted according to their bit orders and then added. In the carry ripple technique (carry propagate principle) the carry signal of each adder in one row is passed on to the next adder of the same row (Fig. 6). A more common addition is the carry save arithmetic in which every carry and sum signal are passed to adders of the next stage (Fig. 8). The binary result is achieved in a final adder. These conventional binary multiplications require the addition of as many partial products as there are multiplier bits.

The most popular technique to reduce the number of partial products to be added is the modified Booth algorithm [7]. When employing this recoding scheme to the multiplier bits the number of partial products is reduced by a factor of two which leads to a corresponding reduction in the number of adder stages (Fig. 10).

Another speed-up of the circuit can be achieved by two parallel carry save adder arrays working concurrently (see Fig. 12; also see [3] and [4]). Their results are grouped together before the final addition. This structure called $N/2$ tree offers a good compromise between speed and density but is slower than fully parallel tree algorithms like the Wallace tree [1].

Manuscript received November 28, 1994; revised March 20, 1995.

O. Salomon is with the Siemens AG, ON TN ET D 31, 81359 München, Germany.

J.-M. Green and H. Klar are with the Institute for Microelectronics, Technical University of Berlin, 10623 Berlin, Germany.

IEEE Log Number 9412008.

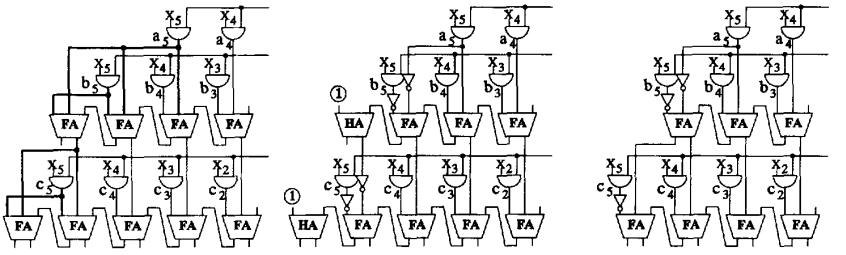


Fig. 2. Simplification of the carry ripple field multiplier.

Because both multiplier and multiplicand may have positive and negative values, the multiplications as well as the partial product additions have to handle signed numbers. The 2's complement representation is the most widespread method for representing both the numbers. It has been preferred since addition and subtraction can be treated equally. When performing 2's complement additions, the rules of 2's complement arithmetic demand a sign bit extension. The width of this sign bit extension depends on the chosen adding method.

When 2's complement partial products are added in carry save arithmetic all numbers to be added in one adder stage have to be of equal bit length. Therefore, the sign bits of the partial product(s) in the first row and the sum and carry signals of each adder row are extended up to the most significant sign bit of the number with the largest absolute value to be added in this stage (Fig. 8).

In carry ripple adders the sign bits of the numbers to be added have to be extended up to the most significant bit of the largest possible absolute value of the sum of this stage (Fig. 6).

In both the carry save and carry ripple additions the sign bit extension results in a higher capacitive load (fan out) of the sign bit signals compared to the load of other signals and accordingly slows down the speed of the circuit.

In the following section a modified 2's complement number representation is derived which in connection with the algorithms of Section IV can eliminate the need of the common sign bit extension in additions.

## III. Modified 2's Complement Number Representation

The value of a 2's complement number A with bit significances from $2^0$ to $2^k$ is calculated by

$$A = -a_k \cdot 2^k + \sum_{i=0}^{k-1} a_i \cdot 2^i \tag{1}$$

where the most significant bit $a_k$ indicates the number's sign (sign bit). $a_k = 0$ denotes $A$ is positive (or 0), whereas if $a_k$ is equal to 1 then $A$ is negative. As described in Section II, when performing additions, the rules of 2's complement arithmetic demand a proper sign bit extension. In this case, the 2's complement number can be divided into three parts, the most significant bit, the extension of the sign bit by $S$ bits and the number's value:

$$A = -a_k \cdot 2^{k+S} + \sum_{i=k}^{k+S-1} a_k \cdot 2^i + \sum_{i=0}^{k-1} a_i \cdot 2^i. \tag{2}$$

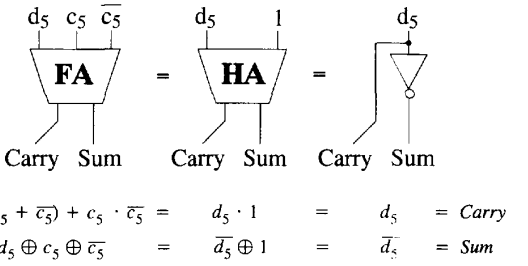


Fig. 3. Addition of a variable with two complementary signals in a full adder or with a "1" in a half adder.

We define $A_{\text{ext}}$ as the sign bit plus the $S$ extended bits. $A_{\text{ext}}$ itself can be represented as a 2's complement number with a word length of $S+1$ bits and a bit order from $2^k$ to $2^{k+S}$

$$\begin{aligned} A_{\text{ext}} &= -a_k \cdot 2^{k+S} + \sum_{i=k}^{k+S-1} a_k \cdot 2^i \\ &= -a_k \cdot 2^{k+S} + a_k \cdot \sum_{i=k}^{k+S-1} 2^i \\ &= a_k \cdot (-2^{k+S} + 2^{k+S} - 2^k) \\ &= -a_k \cdot 2^k, \quad \text{with} \sum_{i=j}^{n-1} 2^i = 2^n - 2^j. \end{aligned} \tag{3}$$

Equation (3) is extended by $(-2^k + 2^k)$ and by $2^{k+S+1}$ which exceeds the bit range of $A_{\text{ext}}$

$$\begin{aligned} A_{\text{ext}} &= (2^{k+S+1} - 2^k + 2^k) - a_k \cdot 2^k \\ &= \sum_{i=k}^{k+S} \cdot 2^i + 2^k - a_k \cdot 2^k \\ &= \sum_{i=k}^{k+S} \cdot 2^i + (1 - a_k) \cdot 2^k. \end{aligned} \tag{4}$$

With the modulo-2 substraction of $(1 - i) = \bar{i}$, (4) results in

$$A_{\text{ext}} = \sum_{i=k}^{k+S} \cdot 2^i + \overline{a_k} \cdot 2^k. \tag{5}$$

Hence, the number $A$ consists of exclusively positive coefficients. It follows for a 2's complement addition that instead of the multiple sign bit addition in (3), a fixed "1" can be added at every bit significance of $A_{\text{ext}}$ and the negated sign bit $a_k$

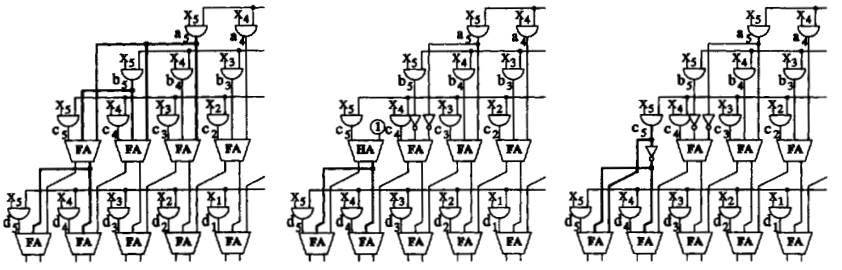


Fig. 4. Simplification of the field multiplier in carry save arithmetic.

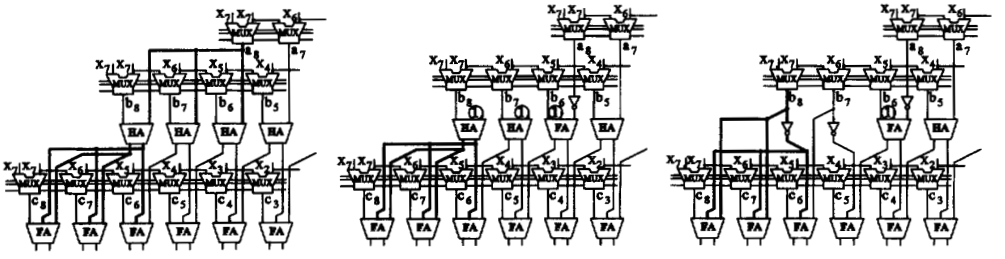


Fig. 5. Simplification of the Booth multiplier in carry save arithmetic.

added at bit significance $2^k$. In the case that the multiplier is an unsigned binary number i.e., only two quadrant multiplication is performed, this scheme is equal to a subsection of the algorithm already derived by Baugh–Wooley [6].

In the following, the addition of two sign bit extension numbers $A_{\text{ext}}$ and $B_{\text{ext}}$ will be simplified. The sum of $A_{\text{ext}}$ and $B_{\text{ext}}$ when both numbers are added at the same bit order from $2^k$ to $2^{k+S}$ is equal to

$$A_{\text{ext}} + B_{\text{ext}} = -a_k \cdot 2^{k+S} + \sum_{i=k}^{k+S-1} a_k \cdot 2^i - b_k \cdot 2^{k+S} + \sum_{i=k}^{k+S-1} b_k \cdot 2^i. \quad (6)$$

Using (5) leads to

$$A_{\text{ext}} + B_{\text{ext}} = 2 \cdot \left( \sum_{i=k}^{k+S} \cdot 2^i \right) + \overline{a_k} \cdot 2^k + \overline{b_k} \cdot 2^k = \sum_{i=k+1}^{k+S+1} \cdot 2^i + (\overline{a_k} + \overline{b_k}) \cdot 2^k. \quad (7)$$

The upper range $k + S + 1$ of the sum in (7) identifies a bit significance $2^{k+S+1}$ which exceeds the bit range of the sum of $A_{\text{ext}}$ and $B_{\text{ext}}$. Therefore the sum's upper range in (7) can be limited to $k + S$

$$A_{\text{ext}} + B_{\text{ext}} = \sum_{i=k+1}^{k+S} \cdot 2^i + (\overline{a_k} + \overline{b_k}) \cdot 2^k. \quad (8)$$

This algebraic expression again consists of only positive coefficients. It follows for this 2's complement addition that the sum of the sign bit extension numbers $A_{\text{ext}}$ and $B_{\text{ext}}$ each having a bit range from $2^k$ to $2^{k+S}$ can be replaced by an addition of "1" between the bit values of $2^{k+1}$ to $2^{k+S}$, and the negated values of $a_k$ and $b_k$ added at the order $2^k$.

An example for the simplified addition of two numbers $A$ and $B$ according to (8) with $k = 6$ and $S = 1$ is shown in Fig. 1. The point of this transformation is that the twofold employment of (5) at the bit order from $2^k$ to $2^{k+S}$ eliminates the additional "1"s at the bit order $2^k$.

## IV. Algorithms for Simplified Additions

### A. Carry Ripple Addition

The first algorithm is derived for a carry ripple addition of 2's complement numbers in successive orders in which each number is shifted by $S$ bits. In the first row of full adders (FA) two numbers are added. The derivation is done sequentially in two steps for a field multiplier (Fig. 6).

*Step I:* In every row of the carry ripple field, the sums of the sign bit extensions are simplified. These substitutions are done using (8). In case of the 6 × 6 b multiplier ($S = 1$, $k = 6$), shown in Fig. 6, and as a subsection in Fig. 2(a), the changes lead to the wiring shown in Fig. 2(b).

*Step II:* In the derived circuit the first half adder (HA) in each row of adders only adds one bit and "1." As shown in Fig. 3, these half adders can be replaced by inverters (Fig. 2(c)). Hence, in every row the adder which added the two sign bits is eliminated. Instead of its sum bit the next less significant carry bit of the same row is added in the following row. In case of the carry ripple addition no further step is necessary. All these changes result in the circuit shown in Fig. 7.

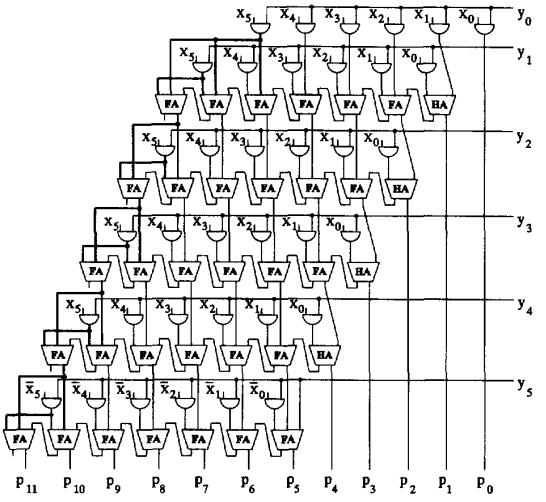

Fig. 6. Conventional 6 × 6 b carry ripple field multiplier.

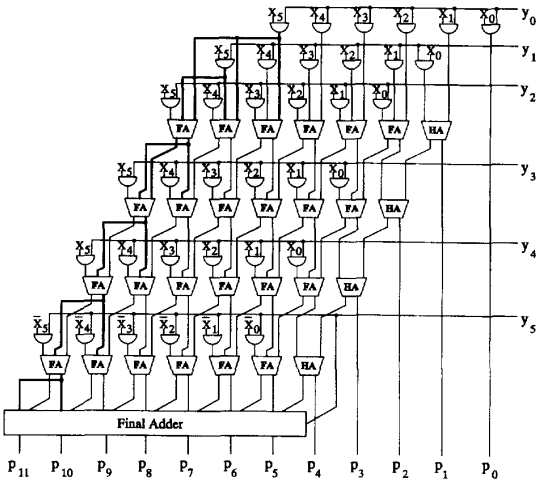


Fig. 8. Conventional 6 × 6 b field multiplier in carry save arithmetic.

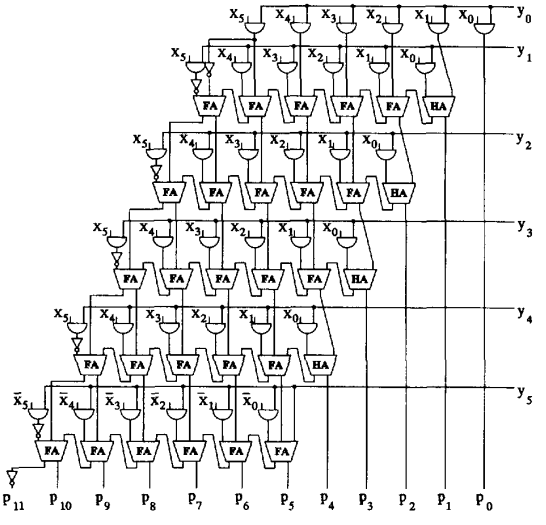

Fig. 7. Optimized 6 × 6 b carry ripple field multiplier.

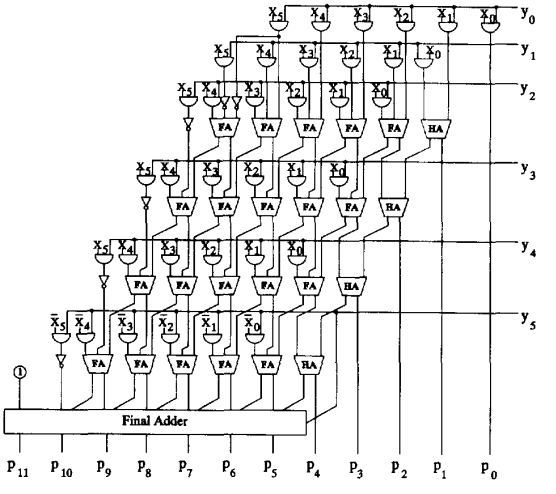


Fig. 9. Optimized 6 × 6 b field multiplier in carry save arithmetic.

### B. Carry Save Addition

The second algorithm is derived for a carry save addition of 2's complement numbers in successive orders, in which each number is shifted by $S$ bits compared to the preceeding number. Furthermore, in the first row of half (HA) or full adders (FA) either two or three numbers are added. The derivation is done sequentially in three steps for a field (Fig. 8) and a Booth multiplier (Fig. 10), respectively.

*Step I:* The sign bit extension(s) of the least significant partial products to be added in the first row of adders is simplified first. In the case of the Booth multiplier where two partial products are added in the first row of half adders the modifications are done using (5). The changes for an 8 × 8 b Booth multiplier ($k = 8$, $S = 2$), shown in Fig. 10 and as a subsection in Fig. 5(a), lead to the wiring in Fig. 5(b). In case of a field multiplier ($k = 6$, $S = 1$) three partial products can be added simultaneously first. Applying (8) to the 6 × 6 b field multiplier in Fig. 8 and as a subsection in Fig. 4(a), results in the wiring in Fig. 4(b).

*Step II:* In the derived circuits shown in Figs. 4(b) and Fig. 5(b) $S$ half adders (HA) in the first row only add a single bit and "1." As shown in Fig. 3 these half adders can be replaced by inverters (see Figs. 4(c) and Fig. 5(c)).

*Step III:* In the second row of carry save adders $S$ full adders add one bit to complementary bits generated in the previous row. The addition of two complementary bits can be replaced by a single addition of "1" (Fig. 3) so that these $S$ full adders can be replaced by inverters, too. Step III is successively repeated in the following rows of adders up to the final adder where again the addition of two complementary signals is replaced by a "1." All changes result in the circuits of Figs. 9 and 11 for the field and Booth multiplier, respectively [5].

The same modifications are applied to a 16 × 16 b Booth multiplier with $N/2$ tree, shown in Fig. 12.

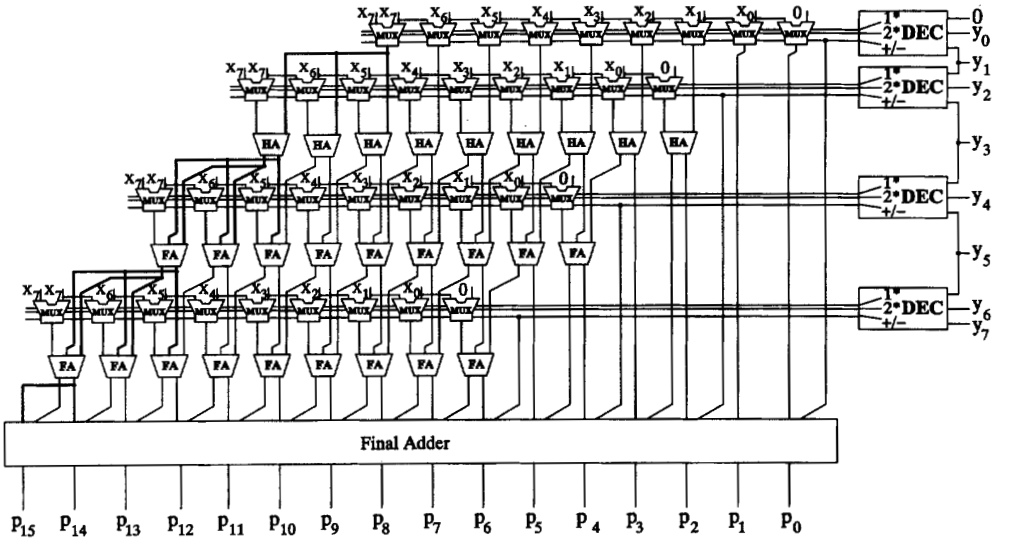


Fig. 10. Conventional 8 × 8 b Booth multiplier in carry save arithmetic.

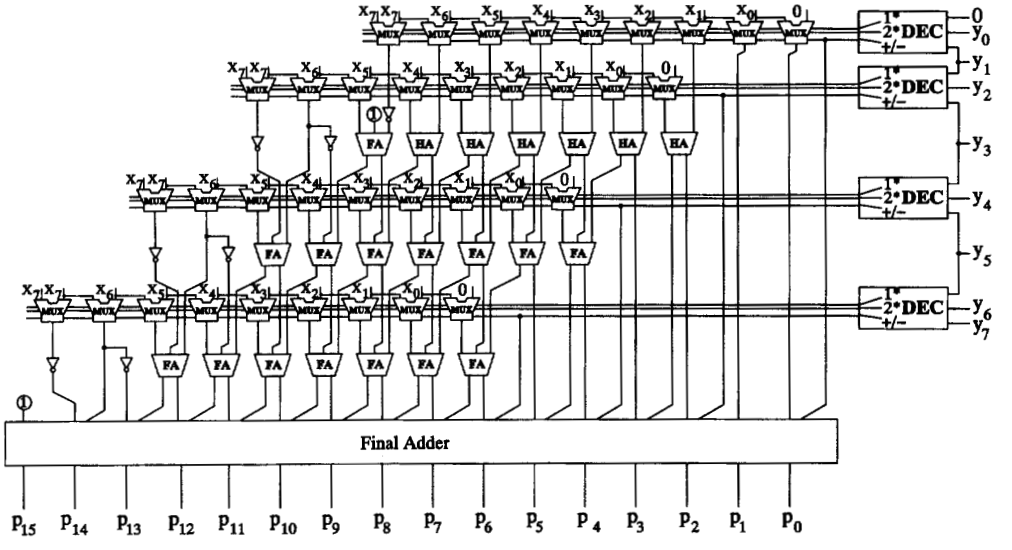


Fig. 11. Optimized 8 × 8 b Booth multiplier in carry save arithmetic.

## V. General Use and Results of the Presented Algorithms

The derived algorithms simplify the addition of 2's complement numbers if either one sign bit extension is added in half adders or when two sign bit extensions are added at the same bit order in half or full adders. The adder circuits are examined with respect to these distinguishing features and are modified in accordance with the algorithms of Section IV. This also works for mixed carry ripple and carry save addition principles.

When partial products are added in carry ripple adders the circuit simplification always leads to the elimination of one full adder per row of adders (see Fig. 7).

In the case of the carry save arithmetic with a shift $S$ between the 2's complement numbers to be added in a row of adders, $S$ of these adders are replaced by inverters. Thus, in common carry save field or Booth multipliers one or two full adders are replaced, respectively (see Figs. 9 and 11). In the case of the addition of partial products in an $N/2$ or a Wallace tree even up to six full adders per row of adders can be replaced by inverters which reduces the number of full adders in a 16 × 16 b Booth multiplier with $N/2$ tree by 25 (15%) (see Fig. 12). Furthermore, the capacitive load of the 2's complement sign bit signals decreases for the field (Fig. 9), Booth (Fig. 11) and $N/2$ tree multiplier (Fig. 12) by up to a factor of two, three and seven, respectively. In fact,

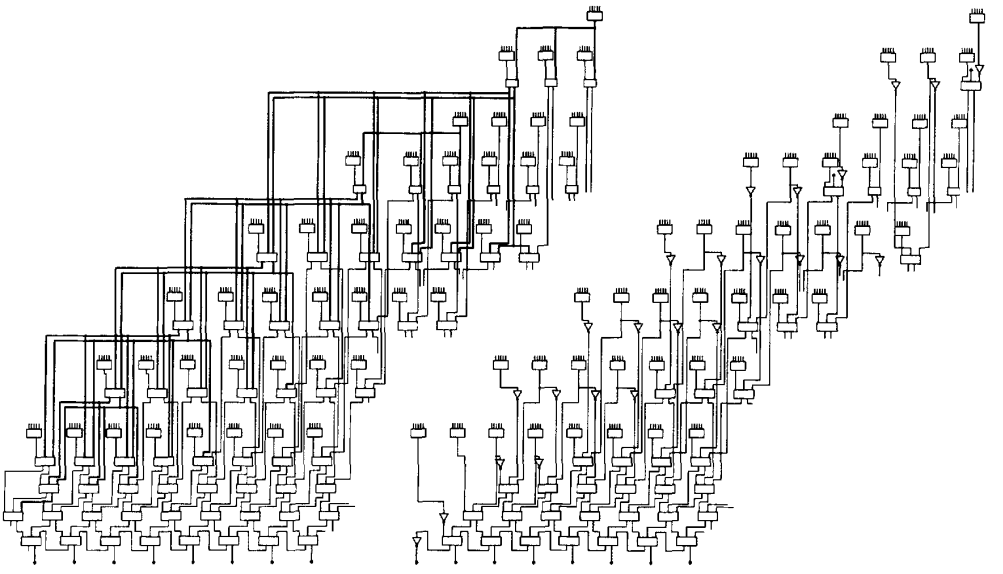

Fig. 12. Subsections of conventional and optimized 16 × 16 b Booth multipliers in carry save arithmetic ($N/2$ tree).

TABLE I
COMPARISON FOR DIFFERENT MULTIPLIERS

| | number of bits of the multiplicand | max. number of adders per row before / after simplification | max. capacitive load before / after simplification |
|---|---|---|---|
| field multiplier | m | m+1 / m | 2 / 1 |
| Booth multiplier | m | m+1 / m-1 | 3 / 1 |
| Booth multiplier with N/2 tree | m | m+5 / m+1 | 7 / 1 |

in any carry save addition the second new algorithm results in approximately the same load for every gate. This allows an optimal transistor sizing and leads to an appropriate delay reduction of the carry save addition.

Thus, the circuit simplifications lead to a smaller area and power consumption, faster speed and simpler layout. In conventional carry save adders both disadvantages, the big sign bit load and the higher number of full adders per row can be reduced to a minimum (see Figs. 8–12 and Table I) which leads to optimal adder circuits.

When only two partial products instead of three as possible are added in a first row of half adders and when assuming the multiplier to be an unsigned binary number without a sign bit then the presented second algorithm with use of (5) leads to a field multiplier which is equal to an implemented subsection of the multiplication algorithm derived by Baugh–Wooley [6].

## VI. SUMMARY

Two new algorithms were presented which simplify any addition of 2's complement numbers if either one sign bit extension is added in half adders or two sign bit extensions are added at the same bit order in either half or full adders. The algorithms form the partial products in such a way that they exclusively have positive coefficients. When they are added in carry ripple adders the circuit simplification always leads to the elimination of one full adder per row of adders. In carry save arithmetic the redundant addition of the common sign bit extension can be fully eliminated. This results in a reduction of the circuit area in common field or Booth multipliers by one or two full adders per row of adders, respectively, whereas in an $N/2$ or Wallace tree up to six full adders per row can be saved. Furthermore, the capacitive load of the intermediate sum and carry sign bit signals decreases by up to a factor of seven which leads to an appropriate reduction of delay.

## REFERENCES

[1] C. S. Wallace, "A suggestion for fast multipliers," *IEEE Trans. Electron. Comput.*, vol. EC-13, pp. 14–17, Feb. 1964.
[2] D. Henlin, M. Fertsch, M. Mazin, and E. Lewis, "A 16 bit * 16 bit pipelined multiplier macrocell," *IEEE J. Solid-State Circuits*, vol. SC-20, pp. 542–547, Oct. 1985.
[3] Y. Oowaki, *et al.*, "A sub-10 ns 16 * 16 multiplier using 0.6 $\mu$m CMOS technology," *IEEE J. Solid-State Circuits*, vol. SC-22, pp. 762–767, Oct. 1987.
[4] M. Belleville, *et al.*, "A 16 * 16 bits multiplier in 0.5 $\mu$m CMOS technology," in *Proc. ESSCIRC'91*, Milan/Italy, Sept. 1991, pp. 149–152.
[5] O. Salomon, J.-M. Green, and H. Klar, "General algorithm for a simplified addition of 2's complement numbers in multipliers," in *Proc. ESSCIRC'94*, Ulm/Germany, Sept. 1994, pp. 208–211.
[6] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. Comput.*, vol. C-22, pp. 1045–1047, Dec. 1973.
[7] L. P. Rubinfield, "A proof of the modified Booth's algorithm for multiplication," *IEEE Trans. Comput.*, vol. C-24, pp. 1014–1015, Oct. 1975.