Parallel encrypted array multipliers

by S. Vassiliadis
M. Putrino
E. M. Schwarz

An algorithm for direct two's-complement and sign-magnitude parallel multiplication is described. The partial product matrix representing the multiplication is converted to an equivalent matrix by encryption. Its reduction, producing the final result, needs no specialized adders and can be added with any parallel array addition technique. It contains no negative terms and no extra "correction" rows; in addition, it produces the multiplication with fewer than the minimal number of rows required for a direct multiplication process.

1. Introduction

The realization of multipliers for digital computers has been considered by several scientists and engineers. Many multiplication techniques and algorithms have been developed and proposed in the past, and some have been implemented in actual hardware. The conventional interactive add-shift methods for multiplication are inexpensive to implement in terms of hardware, but their resulting execution times are too slow to satisfy the increasing demand for speed. Given that circuit density and speed have increased tremendously while hardware costs have decreased, parallel-multiplication schemes for multiplier designs can be implemented that do meet high speed requirements. A variety of parallel-multiplication techniques and algorithms have been proposed in the past. Among the major classes of multiplication techniques are multi-bit overlapped scanning, parallel multiplication, and direct multiplication for array-connected matrices. The overlapped scanning technique has been described for two-bit overlapping by A. D. Booth [1], and has been extended to three- and four-bit overlapping by O. L. MacSorley [2]. The correctness of the three-bit overlapped scanning technique has been proven by L. P. Rubinfield [3], and the generalized proof for multi-bit overlapped scanning can be found in [4]. The parallel-multiplication technique has been introduced by L. Dadda [5], and parallel-multiplier algorithms can be found in [6–8]. A variety of direct-multiplication algorithms have been proposed in the past [9–12] that can be implemented in array-connected configurations.

Any proposed multiplication scheme for the realization of a fast multiplier in essence produces a matrix that contains rows equivalent to the partial products or modified partial products. The resulting matrix that represents the multiplication process has to be added to produce the final product, and some kind of counter is used to satisfy this purpose. The most commonly used, at present, are the 3/2 counters, otherwise called carry-save adders (CSAs). However, there are schemes that propose m/n counters, with m, n being some natural numbers greater than 3 and 2, respectively [5], as for example 5/3, 7/3, 15/4, etc., and generalized counters that receive serial successive weighted input columns to produce their sum and carry elements, taking into account the proper weighting [13, 14], as for example 5/5/4. The counters are placed according to some convenient scheme, e.g., those proposed by Dadda [5] and Wallace [15], and are sometimes connected in a special

manner, as described for instance in [16], to reduce delay. A summary of the major classes of multipliers can be found in [17] and [18].

The direct two's-complement and sign-magnitude parallel multipliers are of particular interest since those representations are used extensively in digital computer architectures. When the two's-complement representation is considered, additional complexity arises with respect to other representations using direct-multiplication algorithms or techniques. This is due to the fact that the sign of the number cannot be separated from its absolute value, as is the case with the sign-magnitude repesentation. The implication is that it is not possible to produce the multiplication by multiplying the two absolute values and then appending the separately computed sign bit to the result. The proposed direct two's-complement algorithms must account for the sign of the number in some way. This is because negative elements will be present when the partial product matrix is created. Those elements are the result of the multiplication of the sign bit of Y (Y being the multiplier) with every bit of X excluding the sign bit (X being the multiplicand) and vice versa. Therefore, element subtraction and addition must be performed during the reduction of the matrix. The addition and subtraction of the elements of the matrix can be achieved by using generalized types of counters, namely counters that can add either positive or negative elements or a combination of both, such as the 3-to-2 counters described in [18]. Alternatively, the partial product matrix can be "conveniently" manipulated to eliminate the "corrections" due to the signs. However, the resulting matrix containing modified partial products with only positive elements may require additional rows. Element addition and subtraction may result not only because of the nature of the two's-complement notation, but also because of the way that an algorithm or technique creates the partial or modified partial product matrix. Again, unless generalized adders are used, the matrix must be manipulated to exclude the negative elements.

This paper introduces a direct algorithm to realize the two's-complement and sign-magnitude multiplications most suitable for array-connected configurations. The partial product matrix produced by the algorithm is independent of the notation used and contains no negative elements, thus allowing the product to be formed using array addition schemes with no generalized counter cells. The partial product matrix is created by using three extra bits on most of the rows, resulting in the encryption of the negative elements, due to consideration of the two's-complement notation. The additional elements need no complex logical cells for their computation. In addition, because of the encryption of the sign, no extra rows need be added for the "corrections" due to the sign of the two's-complement-represented numbers. On the contrary, one row has been eliminated, with a negligible addition of complexity to the few extra elements of the matrix. The resulting matrix is highly regular, and therefore suitable for VLSI designs and implementations.

2. Two's-complement and sign-magnitude multiplication

For two's-complement multiplications, the $2n$-bit product P, represented in binary as $(P_{2n-1}, P_{2n-2}, \cdots, P_0)$, is formed by multiplying the n-bit multiplicand, X, represented in binary as $(X_{n-1}, X_{n-2}, \cdots, X_0)$, and the n-bit multiplier, Y, represented in binary as $(Y_{n-1}, Y_{n-2}, \cdots, Y_0)$. By assuming X and Y to be two-integer numbers such that X_{n-1} and Y_{n-1} are the sign bits, X and Y can be written as

$$X = -X_{n-1}2^{n-1} + \sum_{i=0}^{n-2} X_i 2^i,$$

$$Y = -Y_{n-1}2^{n-1} + \sum_{j=0}^{n-2} Y_j 2^j,$$

and the product XY is equal to

$$\begin{aligned} P &= \Pi_s + \Pi_{ij} - X_{n-1}2^{n-1}\sum_{j=0}^{n-2} Y_j 2^j - Y_{n-1}2^{n-1}\sum_{i=0}^{n-2} X_i 2^i \\ &= \Pi_s + \Pi_{ij} - X_{n-1}Y_0 2^{n-1} - Y_{n-1}X_{n-2}2^{2(n-1)-1} \\ &\quad - \left(X_{n-1}\sum_{j=1}^{n-2} Y_j 2^j + Y_{n-1}\sum_{i=0}^{n-3} X_i 2^i\right)2^{n-1}, \end{aligned} \tag{1}$$

where

$$\Pi_s = X_{n-1}Y_{n-1}2^{2(n-1)},$$

$$\Pi_{ij} = \sum_{i=0}^{n-2}\sum_{j=0}^{n-2} X_i Y_j 2^{i+j}.$$

The multiplication can be produced by adding the matrix in **Figure 1**, which represents the product P as presented in Equation (1).

An immediate observation is that some elements have to be added and some must be subtracted. The implication is that a generalized type of adder must be used, as described for the direct two's-complement array multiplier by Pesaris [9]. Alternatively, subtraction elements can be eliminated, usually with additional rows and consequent manipulation of the multiplication matrix, as described for example in [10] (see **Figure 2**).

For sign-magnitude multiplications, the $2n$-bit product P, represented in binary as $(P_{2n-1}, P_{2n-2}, \cdots, P_0)$, is formed by mutliplying the n-bit multiplicand, X, represented in binary as $(X_{n-1}, X_{n-2}, \cdots, X_0)$, and the n-bit multiplier, Y, represented in binary as $(Y_{n-1}, Y_{n-2}, \cdots, Y_0)$. By assuming X and Y to be two-integer numbers such that X_{n-1} and Y_{n-1} are the sign bits, X and Y can be written as

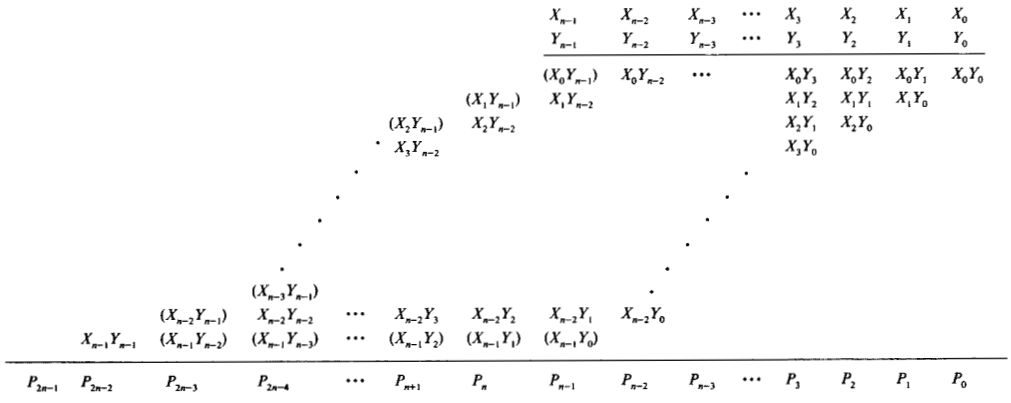

Figure 1

The multiplication matrix corresponding to Equation (1). Parentheses, as in $(X_{n-1}Y_{n-2})$, denote a negative term.

$$X = (-1)^{X_{n-1}} \sum_{i=0}^{n-2} X_i 2^i,$$

$$Y = (-1)^{Y_{n-1}} \sum_{j=0}^{n-2} Y_j 2^j,$$

and

$$XY = (-1)^{X_{n-1}}(-1)^{Y_{n-1}} \sum_{i=0}^{n-2} \sum_{j=0}^{n-2} X_i Y_j 2^{i+j}.$$

The computation of the multiplication can be achieved by separately computing the sign of the final result, which is equal to

$$\text{SIGN} = X_{n-1} \oplus Y_{n-1},$$

where $\oplus$ is exclusive-OR, and multiplying the two absolute values,

$$\sum_{i=0}^{n-2} \sum_{j=0}^{n-2} X_i Y_j 2^{i+j}.$$

If it is desired to use the two's-complement multiplier to compute the multiplication for both representations, the sign-magnitude number representation must be changed to resemble the two's-complement notation. The changing of the sign-magnitude notation to the two's-complement notation, with respect to the multiplication, is trivial. It can be achieved as follows.

Given that sign-magnitude multiplication involves the two absolute numbers and the sign is computed separately, X and Y can be represented as

$$X = -X_{n-1}2^{n-1} + \sum_{i=0}^{n-2} X_i 2^i,$$

$$Y = -Y_{n-1}2^{n-1} + \sum_{j=0}^{n-2} Y_j 2^j,$$

where $X_{n-1} = Y_{n-1} = 0$.

When the operands X, Y are presented as inputs, the two's-complement multiplier will produce the correct answer, except for the sign bit of the result. However, the sign bit can be computed separately and appended at the sign position of the final result, disregarding the resulting sign bit of the two's-complement multiplication. Thus, it can be assumed that the hardware of a two's-complement multiplier will produce both multiplications with one change to accommodate the sign bit, which can be computed by

$$\text{SIGN} = [(X_{n-1} \oplus Y_{n-1}) \cdot S] \oplus [P_{2n-1} \cdot \bar{S}],$$

with $S = 1$ iff sign magnitude multiplication is being considered and $\cdot$ indicating the logical AND function.

3. Encrypted array multiplication

This section is dedicated to the derivation and description of an algorithm to produce the two's-complement multiplication function which, when the observations and sign calculations described previously are implemented, will produce the sign-magnitude multiplication as well.

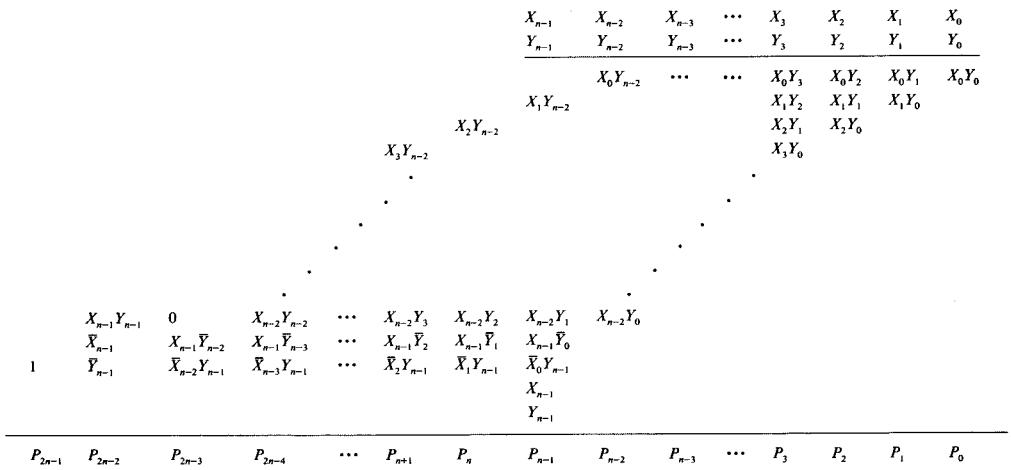

Figure 2

The Baugh–Wooley multiplication matrix.

The multiplication equation is transformed to an equivalent one by encryption of the negative terms. This transformation is achieved by proper element extension and overflow deletion on every row corresponding to each negative element that is present in a partial product. Then, encrypted elements are formed to replace each pair of extended element rows, respecting the binary addition of those rows. Consequently, all of the negative encrypted elements are eliminated, and finally, constants resulting from the encryption are added into the matrix to eliminate extra rows. The transformation of the multiplication matrix is achieved as follows.

Assuming the multiplication as described by Equation (1), the following holds true: For every j, it can be stated that

$$-X_{n-1}Y_j 2^{n-1+j} = X_{n-1}Y_j 2^{n-1+j} \sum_{k=0}^{n} 2^k.$$

This is because

if $X_{n-1}Y_j = 0$

then it is trivially true;

and if $X_{n-1}Y_j = 1$,

then

$$\begin{aligned} -2^{n-1+j} &= 2^{n-1+j} \sum_{k=0}^{n} 2^k \\ &= 2^{n-1+j}(2^0 + 2^1 + \cdots + 2^n) \\ &= 2^{n-1+j}(2^{n+1} - 2^0) \\ &= 2^{2n+j} - 2^{n-1+j}. \end{aligned}$$

Given that $0 \le j \le n-2$, then $2n + j \ge 2n$. The portion 2^{2n+j} is not involved with the multiplication and need not be considered.

Similarly,

$$-Y_{n-1}X_i 2^{n-1+i} = Y_{n-1}X_i 2^{n-1+i} \sum_{k=0}^{n} 2^k.$$

Thus, the multiplication described in Equation (1) is equivalent to

$$\begin{aligned} P = \Pi_s + \Pi_{ij} &- X_{n-1}Y_0 2^{n-1} - Y_{n-1}X_{n-2}2^{2(n-1)-1} \\ &+ \sum_{j=1}^{n-2} X_{n-1}Y_j 2^{n-1+j} \sum_{k=0}^{n} 2^k \\ &+ \sum_{i=0}^{n-3} Y_{n-1}X_i 2^{n-1+i} \sum_{k=0}^{n} 2^k. \end{aligned}$$

The implication is that the multiplication can be written as

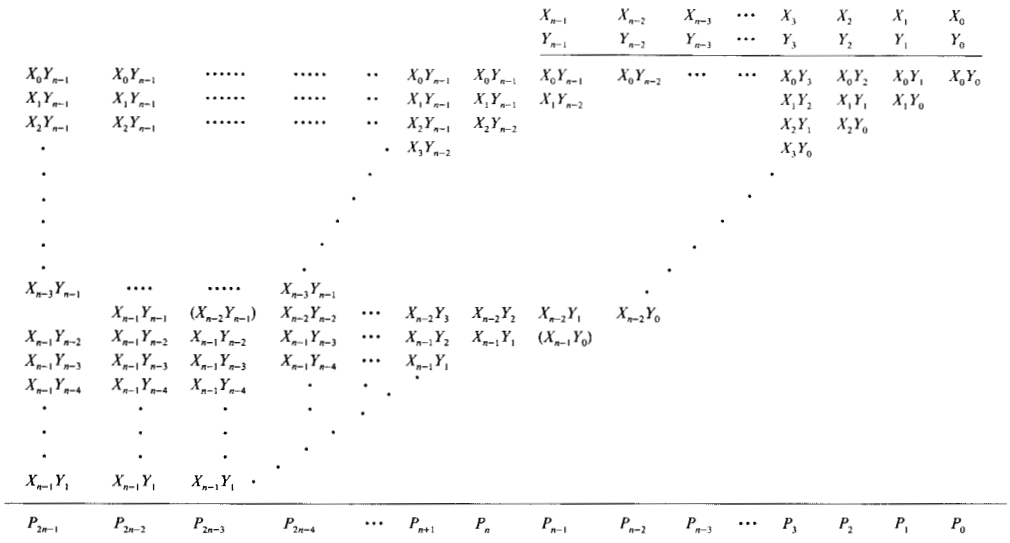

Figure 3

Modified multiplication matrix. $(X_{n-2}Y_{n-1})$ and $(X_{n-1}Y_0)$ denote the only negative terms.

Table 1 Encryption table.

A_j	B_j	*Encryption*		
		$n+j+1$	$n+j$	$n-1+j$
0	0		0	0
0	1		0	−1
1	0		−1	0
1	1	−1	0	1

$$
\begin{aligned}
P &= \Pi_s + \Pi_{ij} - X_{n-1}Y_0 2^{n-1} - Y_{n-1}X_{n-2}2^{2(n-1)-1} \\
&\quad + \sum_{j=0}^{n-3} X_{n-1}Y_{j+1}2^{n+j} \sum_{k=0}^{n} 2^k \\
&\quad + \sum_{i=0}^{n-3} Y_{n-1}X_i 2^{n-1+i} \sum_{k=0}^{n} 2^k \\
&= \Pi_s + \Pi_{ij} - X_{n-1}Y_0 2^{n-1} - Y_{n-1}X_{n-2}2^{2(n-1)-1} \\
&\quad + \sum_{j=0}^{n-3} \left(X_{n-1}Y_{j+1}2^{n+j} \sum_{k=0}^{n} 2^k \right. \\
&\quad \left. + Y_{n-1}X_j 2^{n-1+j} \sum_{k=0}^{n} 2^k \right). \qquad (2)
\end{aligned}
$$

The multiplication matrix described by the previous equation is shown in **Figure 3**.

It can be observed that while the matrix corresponds to the multiplication equation, it contains a few negative elements which can easily be eliminated with "element extension." An attempt to implement the matrix will result in high cost in terms of hardware cells and delay. However, this equation representing the equivalence to the multiplication can be used as a starting point to produce encryption of the negative elements, consequently reducing the number of elements and rows in the multiplication matrix described in Equation (2). The encryption is derived as follows.

Theorem 1

The multiplication matrix is equivalent to

$$
\begin{aligned}
P &= \Pi_s + \Pi_{ij} - X_{n-1}Y_0 2^{n-1} - Y_{n-1}X_{n-2}2^{2(n-1)-1} \\
&\quad + \sum_{j=0}^{n-3} [(A_j \cdot B_j)(2^{n+j-1} - 2^{n+j+1}) \\
&\quad - (\bar{A}_j \cdot B_j)2^{n-1+j} - (A_j \cdot \bar{B}_j)2^{n+j}],
\end{aligned}
$$

where

$A_j = X_{n-1}Y_{j+1}$,

$B_j = Y_{n-1}X_j$,

$\cdot$ is the logical AND, and $\bar{}$ is the logical NOT.

Proof

It must be proven that

$$\sum_{j=0}^{n-3}\left(X_{n-1}Y_{j+1}2^{n+j}\sum_{k=0}^{n}2^k + Y_{n-1}X_j 2^{n-1+j}\sum_{k=0}^{n}2^k\right)$$
$$= \sum_{j=0}^{n-3}[(A_j \cdot B_j)(2^{n+j-1} - 2^{n+j+1})$$
$$-(\bar{A}_j \cdot B_j)2^{n-1+j} - (A_j \cdot \bar{B}_j)2^{n+j}].$$

For any given j, the following holds true.

Case 1 $X_{n-1}Y_{j+1} = Y_{n-1}X_j = 1$.

$$2^{n+j}\sum_{k=0}^{n}2^k + 2^{n-1+j}\sum_{k=0}^{n}2^k$$
$$= 2^{n+j}(2^{n+1} - 2^0) + 2^{n-1+j}(2^{n+1} - 2^0)$$
$$= 2^{2n+j+1} + 2^{2n+j} - 2^{n+j} - 2^{n+j-1}$$
$$= -2^{n+j} - 2^{n+j-1}$$
$$= -2^{n+j+1} + 2^{n+j} - 2^{n+j} + 2^{n+j-1}$$
$$= 2^{n+j-1} - 2^{n+j+1}.$$

Case 2 $X_{n-1}Y_{j+1} = 0$ and $Y_{n-1}X_j = 1$.

$$2^{n-1+j}\sum_{k=0}^{n}2^k = 2^{n-1+j}(2^{n+1} - 2^0)$$
$$= 2^{2n+j} - 2^{n-1+j} = -2^{n-1+j}.$$

Case 3 $X_{n-1}Y_{j+1} = 1$ and $Y_{n-1}X_j = 0$.

$$2^{n+j}(2^{n+1} - 2^0) = -2^{n+j}.$$

Case 4 $X_{n-1}Y_{j+1} = X_{n-1}Y_j = 0$;

then trivially true. Thus, Theorem 1 holds true.

It is evident that depending on A_j and B_j, for every row beginning at the row enumerated as 0 and ending at the row enumerated as $n-3$ and for every row involving positions 2^{n+j+1}, 2^{n+j}, and 2^{n-1+j}, an encryption will result, as represented in **Table 1**.

Given that there are at most three bit positions in the encryption, for uniformity, the following can be proven.

Theorem 2

The multiplication XY is equivalent to

$$P = \Pi_s + \Pi_{ij} - X_{n-1}Y_0 2^{n-1} - Y_{n-1}X_{n-2}2^{2(n-1)-1}$$
$$+ \sum_{j=0}^{n-3}[(A_j \cdot B_j)(2^{n+j-1} - 2^{n+j+1})$$
$$+ (\bar{A}_j \cdot B_j)(2^{n-1+j} + 2^{n+j} - 2^{n+j+1})$$
$$+ (A_j \cdot \bar{B}_j)(2^{n+j} - 2^{n+j+1})],$$

with $\cdot$ and $\bar{}$ as previously defined.

Table 2 Uniform encryption table.

A_j	B_j	*Encryption*		
		$n+j+1$	$n+j$	$n-1+j$
0	0	0	0	0
0	1	−1	1	1
1	0	−1	1	0
1	1	−1	0	1

Proof

Trivial, with geometric series properties applied to the terms containing A_j and B_j.

The implication is that three bits have been added to all of the rows from 0 to $n-3$ of the matrix such that for any given A_j, B_j, **Table 2** will produce the right encryption for the added bit positions.

Theorem 3

The encryption can be transformed into the following equivalence:

$$\sum_{j=0}^{n-3}[(A_j \cdot B_j)(2^{n+j-1} - 2^{n+j+1})$$
$$+ (\bar{A}_j \cdot B_j)(2^{n-1+j} + 2^{n+j} - 2^{n+j+1})$$
$$+ (A_j \cdot \bar{B}_j)(2^{n+j} - 2^{n+j+1})]$$
$$= \sum_{j=0}^{n-3}[B_j 2^{n-1+j} + (A_j \oplus B_j)2^{n+j}]$$
$$- \sum_{j=0}^{n-3}(A_j \mid B_j)2^{n+j+1},$$

where $\cdot$ and $\bar{}$ are as previously defined, $\mid$ is the logical OR, and $\oplus$ is the logical exclusive-OR.

Proof

$$\sum_{j=0}^{n-3}[(A_j \cdot B_j)(2^{n+j-1} - 2^{n+j+1})$$
$$+ (\bar{A}_j \cdot B_j)(2^{n-1+j} + 2^{n+j} - 2^{n+j+1})$$
$$+ (A_j \cdot \bar{B}_j)(2^{n+j} - 2^{n+j+1})]$$
$$= \sum_{j=0}^{n-3}[(A_j \cdot B_j) + (\bar{A}_j \cdot B_j)]2^{n-1+j}$$
$$+ \sum_{j=0}^{n-3}[(A_j \cdot \bar{B}_j) + (\bar{A}_j \cdot B_j)]2^{n+j}$$
$$- \sum_{j=0}^{n-3}[(A_j \cdot B_j) + (\bar{A}_j \cdot B_j) + (A_j \cdot \bar{B}_j)]2^{n+j+1}.$$

If Q, R, and S are one of {0, 1}, then

$$Q + R = (Q \oplus R) + (Q \cdot R)2^1,$$

$$Q + R + S = (Q \oplus R \oplus S) + [(Q \cdot R) | (Q \cdot S) | (R \cdot S)]2^1,$$

$$(A_j \cdot B_j) + (\bar{A}_j \cdot B_j) = (A_j \cdot B_j) \oplus (\bar{A}_j \cdot B_j) = B_j,$$

$$(\bar{A}_j \cdot B_j) + (A_j \cdot \bar{B}_j) = A_j \oplus B_j,$$

and

$$\begin{aligned}(A_j \cdot B_j) + (\bar{A}_j \cdot B_j) + (A_j \cdot \bar{B}_j) &= (A_j \cdot B_j) \oplus (\bar{A}_j \cdot B_j) \oplus (A_j \cdot \bar{B}_j) \\ &= B_j \oplus (A_j \cdot \bar{B}_j) = B_j | (A_j \cdot \bar{B}_j) = B_j | A_j.\end{aligned}$$

Thus, Theorem 3 holds true.

Theorem 3 implies that for all rows, starting at row 0 and ending at row $n-3$, the encryption can be coded with three additional bits, in a unique way, with simple logical functions. Depending on j, i.e., the enumeration of the rows, the encryption can be computed at position 2^{n-1+j} by B_j, at position 2^{n+j} by $A_j \oplus B_j$, and at position 2^{n+j} by $A_j \cdot B_j$.

An inconvenience occurs in producing the encryption for every row in position 2^{n+j+1}, where an element will result that indicates subtraction. Given that the production of a matrix with no special adders is of interest, elements that imply subtraction have to be eliminated. The elimination of those elements can be achieved as follows.

Theorem 4

$$-\sum_{j=0}^{n-3} (A_j | B_j)2^{n+j+1} = \sum_{j=0}^{n-3} (\overline{A_j | B_j})2^{n+j+1} + 2^{n+1} + 2^{2n-1}.$$

Proof

$$\begin{aligned}&\sum_{j=0}^{n-3} (\overline{A_j | B_j})2^{n+j+1} + 2^{n+1} + 2^{2n-1} \\ &\quad = \sum_{j=0}^{n-3} [(\overline{A_j | B_j})2^{n+j+1} + (A_j | B_j)2^{n+j+1} \\ &\qquad - (A_j | B_j)2^{n+j+1}] + 2^{n+1} + 2^{2n-1},\end{aligned}$$

given that

$$\begin{aligned}(\overline{A_j | B_j}) + (A_j | B_j) &= (\bar{A}_j \cdot \bar{B}_j) + (A_j | B_j) \\ &= (\bar{A}_j \cdot \bar{B}_j) \oplus (A_j | B_j) \\ &= (\bar{A}_j \cdot \bar{B}_j) \oplus A_j \oplus B_j \oplus (A_j \cdot B_j) \\ &= (\bar{A}_j \cdot \bar{B}_j) \oplus A_j \oplus (\bar{A}_j \cdot B_j) \\ &= \bar{A}_j \oplus A_j = 1.\end{aligned}$$

Thus,

$$\begin{aligned}&\sum_{j=0}^{n-3} (\overline{A_j | B_j})2^{n+j+1} + 2^{n+1} + 2^{2n-1} \\ &\quad = \sum_{j=0}^{n-3} 2^{n+j+1} - \sum_{j=0}^{n-3} (A_j | B_j)2^{n+j+1} + 2^{n+1} + 2^{2n-1} \\ &\quad = 2^{n+1}(2^0 + 2^1 + \cdots + 2^{n-3}) \\ &\qquad - \sum_{j=0}^{n-3} (A_j | B_j)2^{n+j+1} + 2^{n+1} + 2^{2n-1} \\ &\quad = 2^{n+1}(2^{n-2} - 2^0) - \sum_{j=0}^{n-3} (A_j | B_j)2^{n+j+1} + 2^{n+1} + 2^{2n-1} \\ &\quad = 2^{2n-1} - 2^{n+1} - \sum_{j=0}^{n-3} (A_j | B_j)2^{n+j+1} + 2^{n+1} + 2^{2n-1} \\ &\quad = 2^{2n} - 2^{n+1} - \sum_{j=0}^{n-3} (A_j | B_j)2^{n+j+1} + 2^{n+1} \\ &\quad = 2^{2n} - \sum_{j=0}^{n-3} (A_j | B_j)2^{n+j+1}.\end{aligned}$$

Because 2^{2n} is not involved in the multiplication, it can be excluded, and Theorem 4 holds true.

Theorem 4 can be viewed as an adaptation, to preserve equivalence with multiplication and to transform negative elements into positive elements, of the following theorem.

Theorem 4a

$$-\sum_{j=i}^{i+m} \phi(.)_j 2^{k+j} = \sum_{j=i}^{i+m} \bar{\phi}(.)_j 2^{k+j} + 2^{i+k} - 2^{i+k+m+1},$$

such that

$\phi(.)_j$ and $\bar{\phi}(.)_j$ are one of {0, 1}, with

$\phi(.)_j$ and $\bar{\phi}(.)_j$ mutually exclusive.

Proof

$$\begin{aligned}&\sum_{j=i}^{i+m} \bar{\phi}(.)_j 2^{j+k} + 2^{i+k} - 2^{i+k+m+1} \\ &\quad = \sum_{j=i}^{i+m} [\bar{\phi}(.)_j + \phi(.)_j - \phi(.)_j]2^{j+k} + 2^{i+k} - 2^{i+k+m+1} \\ &\quad = \sum_{j=i}^{i+m} 2^{j+k} - \sum_{j=i}^{i+m} \phi(.)_j 2^{j+k} + 2^{i+k} - 2^{i+k+m+1} \\ &\quad = [2^{i+k} + 2^{i+k+1} + \cdots + 2^{i+k+m}] \\ &\qquad - \sum_{j=i}^{i+m} \phi(.)_j 2^{j+k} + 2^{i+k} - 2^{i+k+m+1} \\ &\quad = 2^{i+k}[2^0 + 2^1 + \cdots + 2^m] \\ &\qquad - \sum_{j=i}^{i+m} \phi(.)_j 2^{j+k} + 2^{i+k} - 2^{i+k+m+1} \\ &\quad = 2^{i+k+m+1} - 2^{i+k} - \sum_{j=i}^{i+m} \phi(.)_j 2^{j+k} + 2^{i+k} - 2^{i+k+m+1} \\ &\quad = -\sum_{j=i}^{i+m} \phi(.)_j 2^{j+k}.\end{aligned}$$

The theorem implies that the negative summation can be transformed to a positive one with the addition and subtraction of a 1 at positions 2^{i+k} and $2^{i+k+m+1}$ respectively; i.e., all negative elements but one have been transformed to positive ones. Regarding finite machine representation, the negative element can be easily transformed to a positive one via proper element extension and overflow elimination. In the multiplication, as described previously, it is achieved as follows.

Theorem 4a can be seen as equivalent to

$$-\sum_{j=i}^{i+m} \phi(\,.\,)_j 2^{j+k} = \sum_{j=i}^{i+m} \bar{\phi}(\,.\,)_j 2^{j+k} + 2^{i+k} + 2^{i+k+m+1}. \quad (3)$$

This is evident because

$$\sum_{j=i}^{i+m} \bar{\phi}(\,.\,)_j 2^{j+k} + 2^{i+k} + 2^{i+k+m+1} = 2^{i+k+m+2} - \sum_{j=i}^{i+m} \phi(\,.\,)_j 2^{j+k}, \quad (4)$$

and if $2^{i+k+m+2}$ need not be considered, it can be concluded that Equation (3) holds true.

In the case that it is not true that $2^{i+k+m+2}$ need not be considered, the elimination can be achieved as follows.

Let $i+m+k+1=a$; if it is assumed that 2^{a+b} is the first position indicating overflow, b being some integer, then

$$2^{a+b-1} + 2^{a+b-2} + \cdots + 2^{a+1} + 2^a = 2^{a+b} - 2^a.$$

Thus, by excluding 2^{a+b}, it can be stated that

$$2^{a+b-1} + 2^{a+b-2} + \cdots + 2^{a+1} + 2^a = -2^a.$$

Then, Theorem 4a can be written as

$$-\sum_{j=i}^{i+m} \phi(\,.\,)_j 2^{j+k} = \sum_{j=i}^{i+m} \bar{\phi}(\,.\,)_j 2^{j+k} + 2^{i+k} + 2^a + 2^{a+1} + \cdots + 2^{a+b-1},$$

with $a=i+m+k+1$ and b, some integer that guarantees the element elimination due to overflow, i.e., the exclusion of 2^{a+b}.

Theorem 4 implies that the multiplication XY is equivalent to

$$P = \Pi_s + \Pi_{ij} - X_{n-1}Y_0 2^{n-1} - Y_{n-1}X_{n-2}2^{2(n-1)-1} + \sum_{j=0}^{n-3} (B_j 2^{n-1+j} + (A_j \oplus B_j)2^{n+j}) + \sum_{j=0}^{n-3} (\overline{A_j \mid B_j})2^{n+j+1} + 2^{n+1} + 2^{2n-1}.$$

The equation still contains two negative terms, $Y_{n-1}X_{n-2}$, which correspond to the multiplication matrix at position $2^{2(n-1)-1}$ and $X_{n-1}Y_0$ at position 2^{n-1}. It can be observed that $-Y_{n-1}X_{n-2}$ can easily be eliminated, because

$$-Y_{n-1}X_{n-2}2^{2(n-1)-1} = Y_{n-1}X_{n-2}(2^{2n-1} + 2^{2n-2} + 2^{2n-3}).$$

The previous equation holds true because terms not involved with the multiplication need not be considered and because $-Y_{n-1}X_{n-2}2^{2(n-1)-1}$ can be written as

$$Y_{n-1}X_{n-2}(2^{2n-1+x} + 2^{2n-1+x-1} + \cdots + 2^{2n-1} + 2^{2n-2} + 2^{2(n-1)-1})$$
$$= Y_{n-1}X_{n-2}[2^{2n-1+x+1} - 2^{2(n-1)-1}]$$
$$= Y_{n-1}X_{n-2}2^{2n+x} - Y_{n-1}X_{n-2}2^{2n-3} = -Y_{n-1}X_{n-2}2^{2n-3},$$

with x being some integer ≥ 0.

The negative $X_{n-1}Y_0$ can be eliminated with proper element extension, described previously, or by the following. Given that the multiplication is equivalent to

$$P = \Pi_s + \Pi_{ij} - Y_{n-1}X_{n-2}2^{2(n-1)-1} + \sum_{j=1}^{n-3} [B_j 2^{n-1+j} + (A_j \oplus B_j)2^{n+j} + (\overline{A_j \mid B_j})2^{n+j+1}] - C2^{n-1} + B_0 2^{n-1} + (A_0 \oplus B_0)2^n + (\overline{A_0 \mid B_0})2^{n+1} + 2^{n+1} + 2^{2n-1},$$

where $C = X_{n-1}Y_0$ and all the logical operators are as previously defined, and because

$$-2^{n-1} = 2^{n-1} + 2^n + 2^{n+1} - 2^{n+2},$$

it can be stated that

$$-C2^{n-1} + B_0 2^{n-1} + (A_0 \oplus B_0)2^n + (\overline{A_0 \mid B_0})2^{n+1} = (B_0 + C)2^{n-1} + [(A_0 \oplus B_0) + C]2^n + [(\overline{A_0 \mid B_0}) + C]2^{n+1} - C2^{n+2},$$

and therefore

$$P = \Pi_s + \Pi_{ij} - Y_{n-1}X_{n-2}2^{2(n-1)-1} + \sum_{j=1}^{n-3} [B_j 2^{n-1+j} + (A_j \oplus B_j)2^{n+j} + (\bar{A}_j \cdot \bar{B}_j)2^{n+j+1}] + (B_0 + C)2^{n-1} + [(A_0 \oplus B_0) + C]2^n + [(\bar{A}_0 \cdot \bar{B}_0) + C]2^{n+1} - C2^{n+2} + 2^{n+1} + 2^{2n-1}.$$

The negative term $-C2^{n+2}$ can be eliminated as follows.

Theorem 5

$$(B_0 + C)2^{n-1} + [(A_0 \oplus B_0) + C]2^n + [(\bar{A}_0 \cdot \bar{B}_0) + C]2^{n+1} - C2^{n+2} = (C \oplus B_0)2^{n-1} + [A_0 \oplus (B_0 \mid C)]2^n + (\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C})2^{n+1}.$$

Proof

$$
\begin{aligned}
&(B_0 + C)2^{n-1} + [(A_0 \oplus B_0) + C]2^n \\
&\quad + [(\bar{A}_0 \cdot \bar{B}_0) + C]2^{n+1} - C2^{n+2} \\
&= (B_0 \oplus C)2^{n-1} + (B_0 \cdot C)2^n \\
&\quad + (A_0 \oplus B_0 \oplus C)2^n + [(A_0 \oplus B_0) \cdot C]2^{n+1} \\
&\quad + [(\bar{A}_0 \cdot \bar{B}_0) \oplus C]2^{n+1} + (\bar{A}_0 \cdot \bar{B}_0 \cdot C)2^{n+2} - C2^{n+2} \\
&= (B_0 \oplus C)2^{n-1} + [(B_0 \cdot C) \oplus A_0 \oplus B_0 \oplus C]2^n \\
&\quad + [B_0 \cdot C \cdot (A_0 \oplus B_0 \oplus C)]2^{n+1} \\
&\quad + [((A_0 \oplus B_0) \cdot C) \oplus (\bar{A}_0 \cdot \bar{B}_0) \oplus C]2^{n+1} \\
&\quad + [(A_0 \oplus B_0) \cdot C \cdot ((\bar{A}_0 \cdot \bar{B}_0) \oplus C)]2^{n+2} \\
&\quad + (\bar{A}_0 \cdot \bar{B}_0 \cdot C)2^{n+2} - C2^{n+2}.
\end{aligned}
$$

But

$$
\begin{aligned}
&(B_0 \cdot C) \oplus A_0 \oplus B_0 \oplus C \\
&\quad = (C \cdot \bar{B}_0) \oplus A_0 \oplus B_0 \\
&\quad = A_0 \oplus (C \mid B_0);
\end{aligned}
$$

$$
\begin{aligned}
&(B_0 \cdot C \cdot A_0) \oplus (B_0 \cdot C) \oplus (B_0 \cdot C) \\
&\quad = B_0 \cdot A_0 \cdot C;
\end{aligned}
$$

$$
\begin{aligned}
&[(A_0 \oplus B_0) \cdot C] \oplus (\bar{A}_0 \cdot \bar{B}_0) \oplus C \\
&\quad = [C \cdot (\overline{A_0 \oplus B_0})] \oplus (\bar{A}_0 \cdot \bar{B}_0);
\end{aligned}
$$

$$
\begin{aligned}
&[(C \cdot A_0) \oplus (B_0 \cdot C)] \cdot [(\bar{A}_0 \cdot \bar{B}_0) \oplus C] \\
&\quad = C \cdot (A_0 \oplus B_0).
\end{aligned}
$$

Thus,

$$
\begin{aligned}
&(B_0 + C)2^{n-1} + [(A_0 \oplus B_0) + C]2^n \\
&\quad + [(\bar{A}_0 \cdot \bar{B}_0) + C]2^{n+1} - C2^{n+2} \\
&= (B_0 \oplus C)2^{n-1} + [A_0 \oplus (C \mid B_0)]2^n + (B_0 \cdot A_0 \cdot C)2^{n+1} \\
&\quad + [(\bar{A}_0 \cdot \bar{B}_0) \oplus (C \cdot (\overline{A_0 \oplus B_0}))]2^{n+1} \\
&\quad + [C \cdot (A_0 \oplus B_0)]2^{n+2} + (\bar{A}_0 \cdot \bar{B}_0 \cdot C)2^{n+2} - C2^{n+2} \\
&= (B_0 \oplus C)2^{n-1} + [A_0 \oplus (C \mid B_0)]2^n \\
&\quad + [(B_0 \cdot A_0 \cdot C) \oplus (\bar{A}_0 \cdot \bar{B}_0) \oplus (C \cdot (\overline{A_0 \oplus B_0}))]2^{n+1} \\
&\quad + [B_0 \cdot A_0 \cdot C \cdot ((\bar{A}_0 \cdot \bar{B}_0) \oplus (C \cdot (\overline{A_0 \oplus B_0})))]2^{n+2} \\
&\quad + [C \cdot (A_0 \oplus B_0)]2^{n+2} + (\bar{A}_0 \cdot \bar{B}_0 \cdot C)2^{n+2} - C2^{n+2}.
\end{aligned}
$$

But

$$
\begin{aligned}
&(B_0 \cdot A_0 \cdot C) \oplus (\bar{A}_0 \cdot \bar{B}_0) \oplus [C \cdot (\overline{A_0 \oplus B_0})] \\
&\quad = C \cdot [(A_0 \cdot B_0) \oplus \bar{A}_0 \oplus B_0] \oplus (\bar{A}_0 \cdot \bar{B}_0) \\
&\quad = C \cdot [(\bar{A}_0 \cdot B_0) \oplus \bar{A}_0] \oplus (\bar{A}_0 \cdot \bar{B}_0) \\
&\quad = C \cdot (\bar{A}_0 \cdot \bar{B}_0) \oplus (\bar{A}_0 \cdot \bar{B}_0) = \bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C}
\end{aligned}
$$

and

$$
\begin{aligned}
&B_0 \cdot A_0 \cdot C \cdot [(\bar{A}_0 \cdot \bar{B}_0) \oplus (C \cdot \bar{A}_0) \oplus (C \cdot B_0)] \\
&\quad = A_0 \cdot B_0 \cdot C.
\end{aligned}
$$

Thus,

$$
\begin{aligned}
&(B_0 + C)2^{n-1} + [(A_0 \oplus B_0) + C]2^n \\
&\quad + [(\bar{A}_0 \cdot \bar{B}_0) + C]2^{n+1} - C2^{n+2} \\
&= (B_0 \oplus C)2^{n-1} + [A_0 \oplus (C \mid B_0)]2^n + (\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C})2^{n+1} \\
&\quad + (A_0 \cdot B_0 \cdot C)2^{n+2} + [C \cdot (A_0 \oplus B_0)]2^{n+2} \\
&\quad + (\bar{A}_0 \cdot \bar{B}_0 \cdot C)2^{n+2} - C2^{n+2},
\end{aligned}
$$

given that

$$
\begin{aligned}
&[(A_0 \cdot B_0 \cdot C) + (C \cdot (A_0 \oplus B_0))]2^{n+2} \\
&\quad = [(A_0 \cdot B_0 \cdot C) \oplus (C \cdot A_0) \oplus (C \cdot B_0)]2^{n+2} + 0 \\
&\quad = [(A_0 \cdot \bar{B}_0 \cdot C) \oplus (C \cdot B_0)]2^{n+2},
\end{aligned}
$$

and

$$
\begin{aligned}
&(\bar{A}_0 \cdot \bar{B}_0 \cdot C)2^{n+2} + [(A_0 \cdot \bar{B}_0 \cdot C) \oplus (C \cdot B_0)]2^{n+2} \\
&\quad = [(\bar{A}_0 \cdot \bar{B}_0 \cdot C) \oplus (A_0 \cdot \bar{B}_0 \cdot C) \oplus (C \cdot B_0)]2^{n+2} + 0;
\end{aligned}
$$

$$
C \cdot [(\bar{A}_0 \cdot \bar{B}_0) \oplus (A_0 \cdot \bar{B}_0) \oplus B_0] = C \cdot (\bar{B}_0 \oplus B_0) = C.
$$

Thus, it can be concluded that

$$
\begin{aligned}
&(B_0 \oplus C)2^{n-1} + [A_0 \oplus (C \mid B_0)]2^n \\
&\quad + (\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C})2^{n+1} + C2^{n+2} - C2^{n+2} \\
&= (B_0 \oplus C)2^{n-1} + [A_0 \oplus (C \mid B_0)]2^n + (\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C})2^{n+1},
\end{aligned}
$$

and Theorem 5 holds true.

From Theorem 5, it follows that the multiplication XY is equivalent to

$$
\begin{aligned}
P = \Pi_s + \Pi_{ij} &- Y_{n-1}X_{n-2}2^{2(n-1)-1} \\
&+ \sum_{j=1}^{n-3} [B_j 2^{n-1+j} + (A_j \oplus B_j)2^{n+j} \\
&+ (\bar{A}_j \cdot \bar{B}_j)2^{n+j+1}] \\
&+ (B_0 \oplus C)2^{n-1} + [A_0 \oplus (B_0 \mid C)]2^n \\
&+ (\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C})2^{n+1} + 2^{n+1} + 2^{2n-1}, \qquad (5)
\end{aligned}
$$

which corresponds to the matrix in **Figure 4**. Such a matrix can be transformed, taking into account the elimination of the element $-Y_{n-1}X_{n-2}2^{2(n-1)-1}$ and the addition

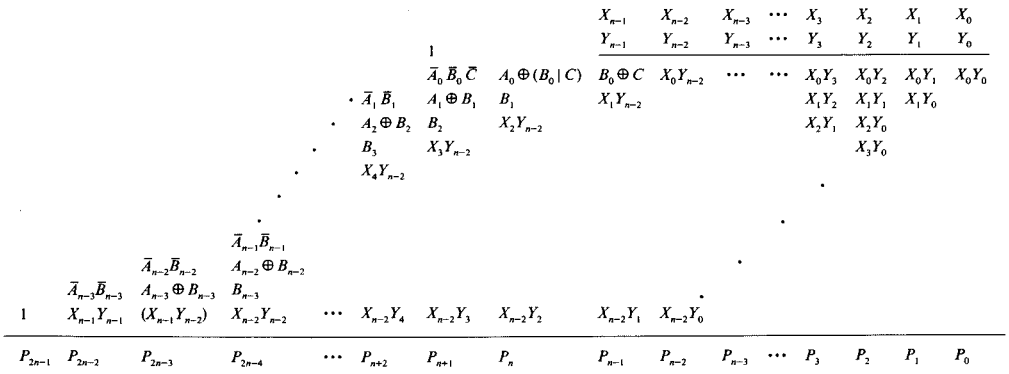

Figure 4

The multiplication matrix corresponding to Equation (5). ($X_{n-1}Y_{n-2}$) denotes the only negative term.

$(\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C} + 1)2^{n+1}$,

as follows:

$$
\begin{aligned}
P = {} & \Pi_s + \Pi_{ij} + Y_{n-1}X_{n-2}2^{2(n-1)-1} \\
& + Y_{n-1}X_{n-2}2^{2(n-1)} + Y_{n-1}X_{n-2}2^{2n-1} \\
& + \sum_{j=1}^{n-3} [B_j 2^{n-1+j} + (A_j \oplus B_j)2^{n+j} + (\bar{A}_j \cdot \bar{B}_j)2^{n+j+1}] \\
& + (B_0 \oplus C)2^{n-1} + [A_0 \oplus (B_0 \mid C)]2^n \\
& + (A_0 \mid B_0 \mid C)2^{n+1} + (\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C})2^{n+2} + 2^{2n-1},
\end{aligned}
$$

which corresponds to the multiplication matrix containing no negative terms shown in **Figure 5**.

The matrix as presented in Figure 5 contains $n-1$ rows instead of n, as normally required in the direct computation of the multiplication. It contains three additional bits on every row starting at the row enumerated as 1 (in effect, the second row) and ending at the row enumerated $n-3$ (in effect, the $n-2$ row). For every $1 \le j \le n-3$, the three bits are added at position 2^{n-1+j} computed by the logical AND of Y_{n-1} and X_j; at position 2^{n+j} computed by the logical AND of Y_{n-1} and X_j, the logical AND of X_{n-1} and Y_{j+1}, and the exclusive-OR of the resulting logical ANDs; and, finally, at position 2^{n+j+1} computed by the logical AND of Y_{n-1} and X_j, the logical AND of X_{n-1} and Y_{j+1}, and then the logical NOR of their result.

At the first row and starting at position 2^{n-1} and ending at position 2^{n+2}, four elements are added and can be computed by the logical cells shown in **Figure 6**.

An observation worth exploring is that while the rows for the multiplication matrix have been reduced by one, there are some elements that require more complex logical cells than the simple logical AND. It can also be observed that those elements are very few, especially when large operands are considered, and their calculation should definitely not add delay to an array-connected multiplier. This is because they require less delay than a CSA adder cell. Thus, even if no special attention is paid to the layout, they will not penalize the addition of one full stage. In addition, those quantities can be carefully calculated during the repowering of the first stage.

A more important consideration is that the complex elements can be added so that there is no need for any of the above. This is because, in an array-connected multiplier, they can be positioned at the end of the configuration, thus producing no extra delay when added.

Thus, it can be concluded that a row can be eliminated from the usual number of rows needed to produce an $n \times n$ multiplication.

In the Appendix, it can be found that more rows can be eliminated from the matrix configuration with the addition of more complexity for the computation of a few elements. Given that the time of the computation is of importance, and the number of elements requiring complex calculations is limited, it is worth doing so.

Another observation worth noting is that the matrix contains two uniformities, one belonging to the encryption bits and one belonging to the rest of the matrix. Thus, it can be concluded that the layout of the multiplier achieves a high degree of regularity, making it suitable for VLSI design.

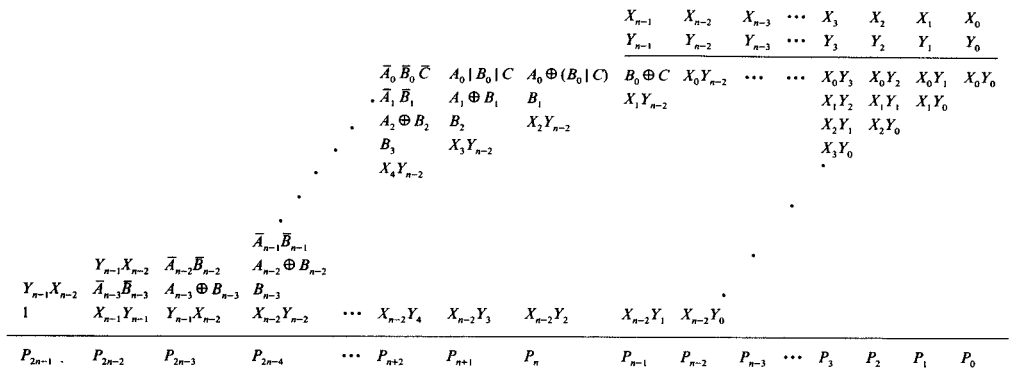

Figure 5

The final multiplication matrix.

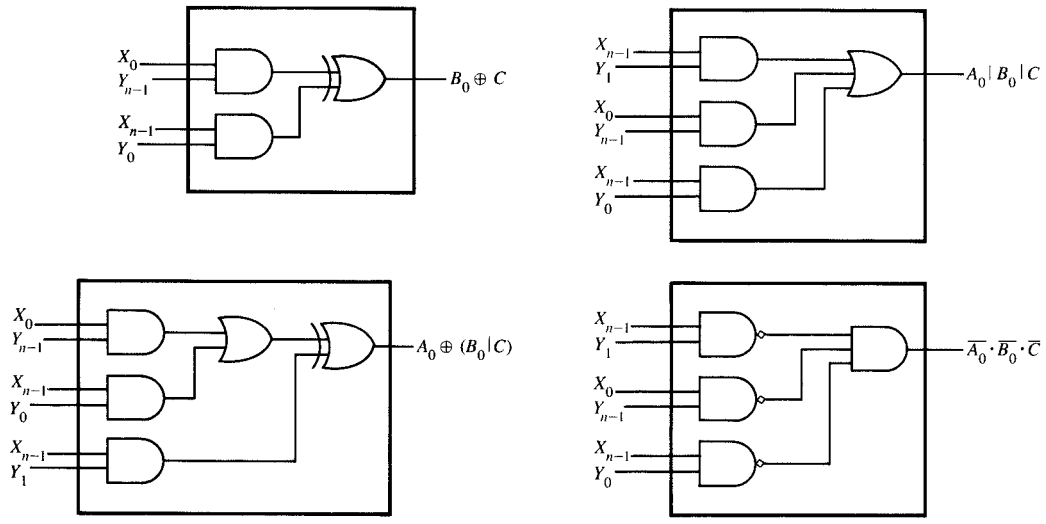

Figure 6

Logic cells for the encryption elements.

4. Conclusions

The encryption algorithm proposed above is an approach for the realization of two's-complement multipliers with direct multiplication. The negative terms present in two's-complement multiplication, produced by the fact that the sign and the absolute value are inseparable, are imbedded into the matrix as positive quantities with the addition of three bits in the high-order position. The implication is that there is no need for specialized adders that take into account element addition and subtraction, and furthermore, that no additional "correction" rows are needed to perform the multiplication. Additionally, there is no need for complete sign extension to preserve the equivalence of the matrix if negative terms are to be excluded.

The resulting matrix can be seen as uniform, with most of the initial matrix unaltered (i.e., each element is computed by the logical AND of the quantities involved), and with the addition of the encryption which consists of elements that are more difficult to implement than the rest of the matrix.

Two out of three bit positions, for every row but the first requiring encryption, can be computed with the logical exclusive-OR or with the logical NOR of the required $X_i Y_j$. However, except for the minimal amount of complexity added to produce the matrix, the element realizations will not result in increasing the delay needed to produce the multiplication because they are not required immediately.

On the first row, the four required elements that are not simple ANDs can be computed with simple logical cells.

A disadvantage of the encrypted multiplier is that some of the encrypted elements require more difficult functions than a simple AND to be computed, but it can be observed that those elements can be swapped with elements that are computed by ANDs and are further down in the matrix in the same column, so that they are available when required. Thus, in an array-connected multiplier, they will not penalize the total multiplication delay.

In addition, the number of rows required to produce the two's-complement multiplier is equal to $n-1$ instead of n, and the difficult elements of the last rows are of a lesser degree of complexity to compute than that of a three-way exclusive-OR. Thus, it can be stated that the multiplier will require less delay if compared to a direct two's-complement multiplier with no encryption.

Given its uniformity, the encrypted matrix results in a high degree of regularity, thus making it suitable for VLSI array-connected layouts.

Finally, it can be observed that it is possible to decrease the number of rows of the multiplier with the addition of complexity for the computations of some limited number of elements, as described in the Appendix.

The algorithm will also produce multipliers that can handle sign-magnitude notation, given that sign-magnitude notation, with respect to multiplication, can be seen as two's-complement with a minimal change regarding the sign correction. The resulting multiplication matrix with proper additional circuitry can also accommodate one's complement and unsigned notations for universal types of operations.

Appendix A: Further row reduction

The encrypted matrix can be reduced as shown below. Equation (5) can be transformed easily, so that the multiplication can be expressed as follows:

$$P = \Pi_s + \Pi_{ij} - Y_{n-1}X_{n-2}2^{2(n-1)-1} + \sum_{j=2}^{n-3}[B_j 2^{n-1+j} + (A_j \oplus B_j)2^{n+j} + (\bar{A}_j \cdot \bar{B}_j)2^{n+j+1}] + (B_0 \oplus C)2^{n-1} + B_1 2^n + [A_0 \oplus (B_0 \mid C)]2^n + (A_1 \oplus B_1)2^{n+1} + (\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C})2^{n+1} + (\bar{A}_1 \cdot \bar{B}_1)2^{n+2} + 2^{n+1} + 2^{2n-1}.$$

Given that

$$[B_1 + A_0 \oplus (B_0 \mid C)]2^n = [B_1 \oplus A_0 \oplus (B_0 \mid C)]2^n + B_1 \cdot [A_0 \oplus (B_0 \mid C)]2^{n+1},$$

and because

$$\Pi_{ij} = \sum_{j=0}^{n-2} X_0 Y_j 2^j + \sum_{j=0}^{n-3} X_1 Y_j 2^{j+1} + \sum_{i=2}^{n-2}\sum_{j=0}^{n-2} X_i Y_j 2^{i+j} + X_1 Y_{n-2} 2^{n-1} = \Pi_{ij}^* + X_1 Y_{n-2} 2^{n-1}$$

by its definition, and

$$[E + (B_0 \oplus C)]2^{n-1} = (E \oplus B_0 \oplus C)2^{n-1} + [E \cdot (B_0 \oplus C)]2^n$$

when

$$E = X_1 Y_{n-2},$$

the multiplication matrix is equivalent to

$$P = \Pi_s + \Pi_{ij}^* - Y_{n-1}X_{n-2}2^{2(n-1)-1} + \sum_{j=2}^{n-3}[B_j 2^{n-1+j} + (A_j \oplus B_j)2^{n+j} + (\bar{A}_j \cdot \bar{B}_j)2^{n+j+1}] + (\bar{A}_1 \cdot \bar{B}_1)2^{n+2} + (A_1 \oplus B_1)2^{n+1} + (\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C})2^{n+1} + [B_1 \cdot (A_0 \oplus (B_0 \mid C))]2^{n+1} + [E \cdot (B_0 \oplus C)]2^n + [B_1 \oplus A_0 \oplus (B_0 \mid C)]2^n + (E \oplus B_0 \oplus C)2^{n-1} + 2^{n+1} + 2^{2n-1}.$$

Since

$$[1 + (A_1 \oplus B_1)]2^{n+1} = (\bar{A}_1 \oplus B_1)2^{n+1} + (A_1 \oplus B_1)2^{n+2},$$

then

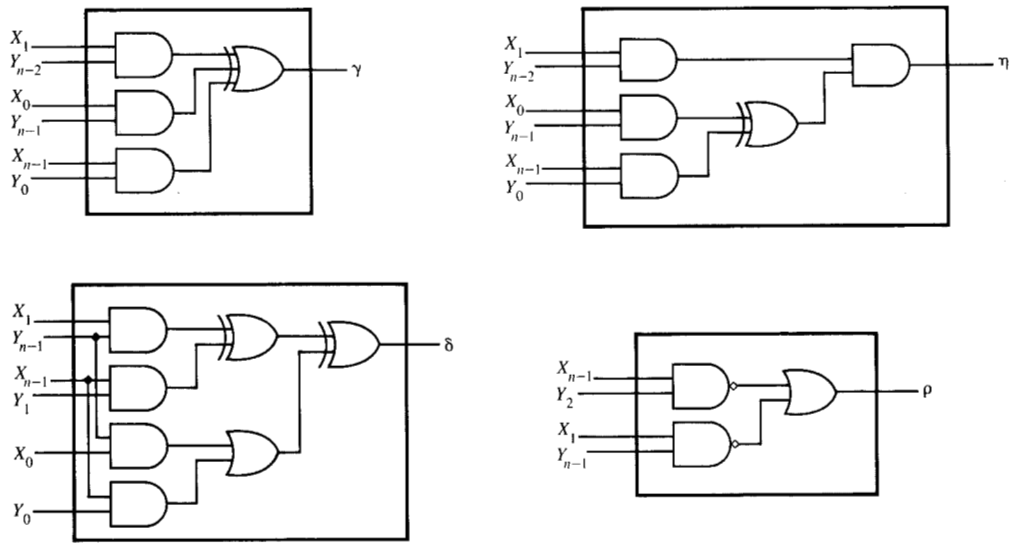

Figure 7

Logic cells for γ, η, δ, and ρ.

$$[(\bar{A}_1 \cdot \bar{B}_1) + (A_1 \oplus B_1)]2^{n+2}$$
$$= [(\bar{A}_1 \cdot \bar{B}_1) \oplus A_1 \oplus B_1]2^{n+2} + 0$$
$$= [(\bar{A}_1 \mid B_1) \oplus A_1]2^{n+2} = (\bar{A}_1 \mid \bar{B}_1)2^{n+2},$$

and

$$[(\bar{A}_1 \oplus B_1) + (\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C})]2^{n+1}$$
$$= [(\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C}) \oplus \bar{A}_1 \oplus B_1]2^{n+1}$$
$$+ [\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C} \cdot (\bar{A}_1 \oplus B_1)]2^{n+2}.$$

Thus, the multiplication can be expressed as follows:

$$P = \Pi_s + \Pi_{ij}^{*} + Y_{n-1}X_{n-2}2^{2(n-1)-1}$$
$$+ Y_{n-1}X_{n-2}2^{2(n-1)} + Y_{n-1}X_{n-2}2^{2n-1}$$
$$+ \sum_{j=2}^{n-3} [B_j 2^{n-1+j} + (A_j \oplus B_j)2^{n+j} + (\bar{A}_j \cdot \bar{B}_j)2^{n+j+1}]$$
$$+ \gamma 2^{n-1} + \delta 2^n + \eta 2^n + \lambda 2^{n+1} + \mu 2^{n+1}$$
$$+ \rho 2^{n+2} + \psi 2^{n+2} + 2^{2n-1}, \quad (6)$$

such that

$$\gamma = E \oplus B_0 \oplus C,$$
$$\delta = B_1 \oplus A_0 \oplus (B_0 \mid C),$$
$$\eta = E \cdot (B_0 \oplus C),$$
$$\lambda = B_1 \cdot [A_0 \oplus (B_0 \mid C)],$$
$$\mu = (\bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C}) \oplus \bar{A}_1 \oplus B_1,$$
$$\rho = \bar{A}_1 \mid \bar{B}_1,$$

and

$$\psi = \bar{A}_0 \cdot \bar{B}_0 \cdot \bar{C} \cdot (\bar{A}_1 \oplus B_1),$$

which can be computed by the logic cells in **Figures 7** and **8**.

A representation of the matrix is reported in **Figure 9**. The matrix still contains $n - 1$ rows. However, it can be transformed, as in **Figure 10**, with

$$\alpha = X_{n-2}Y_0 \oplus X_{n-1}Y_1 \quad \text{and} \quad \beta = X_{n-2}Y_0 \cdot X_{n-1}Y_1.$$

The $(n - 1)$th row contains four elements starting at position $n - 1$ and ending at position $n + 2$. In addition, the $(n - 2)$th row contains a zero at position $n + 3$. Thus, with proper addition, the four elements of the $(n - 1)$th row can be eliminated. Therefore, the multiplication can be represented with a matrix that contains $n - 2$ rows.

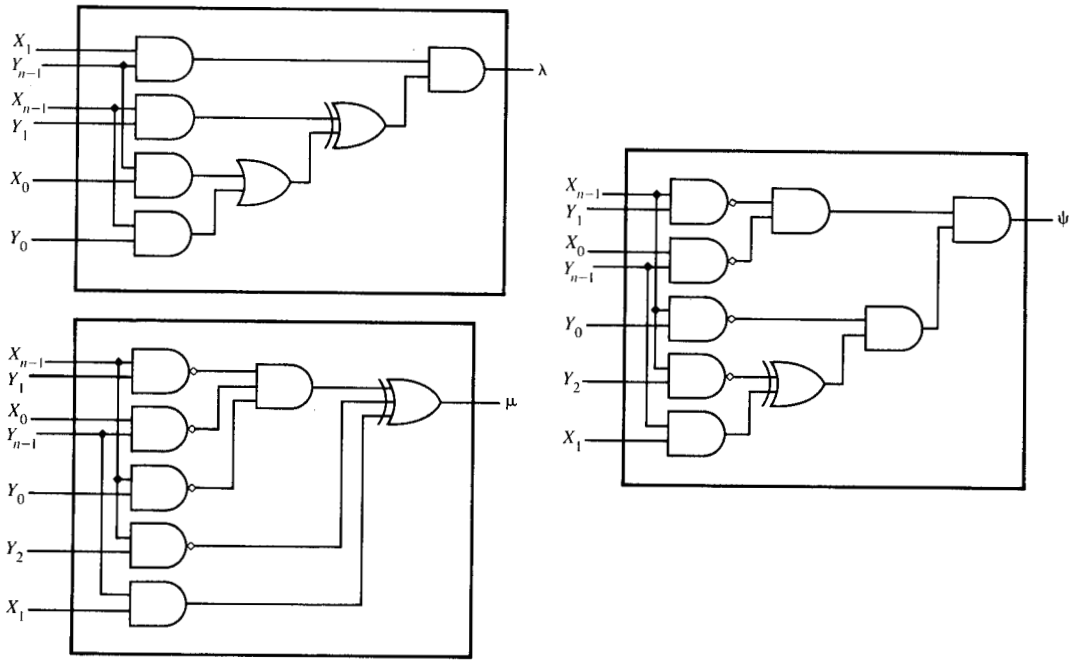

Figure 8

Logic cells for λ, μ, and ψ.

$$
\begin{array}{cccccccccccccccc}
 & & & & & & & & X_{n-1} & X_{n-2} & X_{n-3} & \cdots & X_3 & X_2 & X_1 & X_0 \\
 & & & & & & & & Y_{n-1} & Y_{n-2} & Y_{n-3} & \cdots & Y_3 & Y_2 & Y_1 & Y_0 \\
 & & & & & \psi & \lambda & \eta & & X_0Y_{n-2} & \cdots & \cdots & X_0Y_3 & X_0Y_2 & X_0Y_1 & X_0Y_0 \\
 & & & & & \rho & \mu & \delta & \gamma & & & & X_1Y_2 & X_1Y_1 & X_1Y_0 & \\
 & & & & \bar{A}_2\bar{B}_2 & A_2 \oplus B_2 & B_2 & X_2Y_{n-2} & & & & & X_2Y_1 & X_2Y_0 & & \\
 & & & & \cdot & B_3 & X_3Y_{n-2} & & & & & & X_3Y_0 & & & \\
 & & & & & X_4Y_{n-2} & & & & & & & \cdot & & & \\
 & & & \cdot & & & & & & & & \cdot & & & & \\
 & & & \bar{A}_{n-1}\bar{B}_{n-1} & & & & & & & & & & & & \\
 & Y_{n-1}X_{n-2} & \bar{A}_{n-2}\bar{B}_{n-2} & A_{n-2} \oplus B_{n-2} & & & & & & & \cdot & & & & & \\
Y_{n-1}X_{n-2} & \bar{A}_{n-3}\bar{B}_{n-3} & A_{n-3} \oplus B_{n-3} & B_{n-3} & & & & & & & & & & & & \\
1 & X_{n-1}Y_{n-1} & Y_{n-1}X_{n-2} & X_{n-2}Y_{n-2} & \cdots & X_{n-2}Y_4 & X_{n-2}Y_3 & X_{n-2}Y_2 & X_{n-2}Y_1 & X_{n-2}Y_0 & & & & & & \\
\hline
P_{2n-1} & P_{2n-2} & P_{2n-3} & P_{2n-4} & \cdots & P_{n+2} & P_{n+1} & P_n & P_{n-1} & P_{n-2} & P_{n-3} & \cdots & P_3 & P_2 & P_1 & P_0
\end{array}
$$

Figure 9

The multipication matrix corresponding to Equation (6).

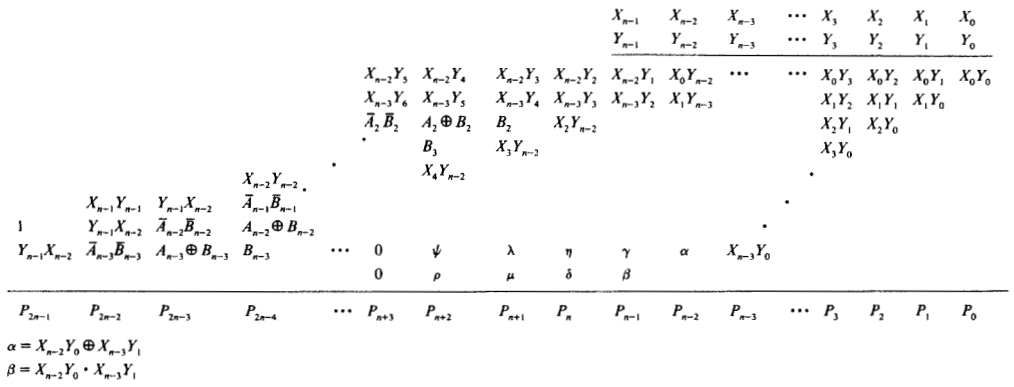

Figure 10

The transformed multiplication matrix corresponding to Equation (6).

The same process can be applied to further reduce the number of rows. However, depending on the length of the operands, the number of "difficult" elements and the complexity of producing them, and the additional complexity added to the layout of the multiplier, the purpose of such reductions becomes defeated at a certain point, and no further reductions should be considered.

References

1. A. D. Booth, "A Signed Multiplication Technique" (Part 2), *Quart. J. Mech. & Appl. Math.* **4,** 236–240 (1951).
2. O. L. MacSorley, "High-Speed Arithmetic in Binary Computers," *Proc. IRE* **99,** 67–91 (January 1961).
3. L. P. Rubinfield, "A Proof of the Modified Booth's Algorithm for Multiplication," *IEEE Trans. Computers* **C-24,** 1014–1015 (October 1975).
4. S. Vassiliadis, E. M. Schwarz, and D. J. Hanrahan, "A General Proof for Overlapped Multiple-Bit Scanning Multiplications," to appear in *IEEE Trans. Computers*, accepted for publication November 1987.
5. L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza* **34,** 349–356 (May 1965).
6. D. Ferrari and R. Stefanelli, "Some New Schemes for Parallel Multipliers," *Alta Frequenza* **38,** 843–852 (November 1969).
7. L. Dadda, "On Parallel Digital Multipliers," *Alta Frequenza* **45,** 574–580 (October 1976).
8. L. Dadda and D. Ferrari, "Digital Multipliers: A Unified Approach," *Alta Frequenza* **37,** 1079–1086 (November 1968).
9. S. D. Pesaris, "A 40-ns 17-Bit Array Multiplier," *IEEE Trans. Computers* **C-20,** 442–447 (April 1971).
10. C. R. Baugh and B. A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm," *IEEE Trans. Computers* **C-22,** 1045–1047 (December 1973).
11. R. De Mori and A. Serra, "A Parallel Structure for Sign Number Multiplication and Addition," *IEEE Trans. Computers* **C-21,** 1453–1454 (December 1972).
12. J. C. Majithia and R. Kita, "An Interactive Array for Multiplications of Signed Magnitude Numbers," *IEEE Trans. Computers* **C-20,** 214–216 (February 1971).
13. W. J. Stenzel, W. J. Kubitz, and G. H. Garcia, "Compact High-Speed Parallel Multiplication Scheme," *IEEE Trans. Computers* **C-26,** 948–957 (October 1977).
14. L. Dadda, "Composite Parallel Counters," *IEEE Trans. Computers* **C-29,** 942–946 (October 1980).
15. C. S. Wallace, "A Suggestion for Parallel Multipliers," *IEEE Trans. Electron Computers* **EC-13,** 14–17 (February 1964).
16. B. R. Mercy, "Multiplier Speed Improvement by Skipping Carry Save Adders," U.S. Patent 4,556,948, December 3, 1985.
17. S. Waser and M. Flynn, *Introduction to Arithmetic for Digital System Designers,* CBS College Publishing, New York, 1982, Ch. 4.
18. K. Hwang, *Computer Arithmetic Principles, Architecture, and Design*, John Wiley & Sons, Inc., New York, 1979.

Received March 31, 1987; accepted for publication February 10, 1988

Stamatis Vassiliadis *IBM System Products Division, P.O. Box 6, Endicott, New York 13760.* Dr. Vassiliadis received the Dr.Eng. degree in electronic engineering from the Politecnico di Milano, Milan, Italy, in 1978. He was most recently a member of the task force that defined the high-level design for a new computer system, for which he is currently the technical leader of the floating-point unit development group. Before his current assignment he participated in the design of the IBM 9370 Model 60. Since joining IBM he has received the First Invention Filed Award, the first and second levels of the Author's Recognition Award, and the first level of the IBM Invention Achievement Award. His research interests include computer arithmetic, computer architecture and hardware design, error detection and fault isolation for hardware implementations, pipelined computers, and vector and parallel processors. Dr. Vassiliadis is an adjunct professor at the Watson School of Engineering, State University of New York at Binghamton, and a member of the Computer Society of the Institute of Electrical and Electronics Engineers.

Michael Putrino *IBM System Products Division, P.O. Box 6, Endicott, New York 13760.* Mr. Putrino received the B.S. degree (cum laude) in electrical engineering from Syracuse University in 1977. He also received an A.S. degree in electrical technology from Broome Community College, Binghamton, New York, in 1975. Mr. Putrino joined IBM in 1977; he is currently a staff engineer and the design coordinator for the execution unit of a high-performance computer processor. His main interests lie within the scope of computer arithmetic, computer architecture and hardware design, error detection and fault isolation for hardware implementations, pipelined computers, and parallel processors. He has received the first level of the Author's Recognition Award and the First Patent Filed Award. Mr. Putrino is a member of the Institute of Electrical and Electronics Engineers and the Computer Society of the IEEE.

Eric M. Schwarz *IBM System Products Division, P.O. Box 6, Endicott, New York 13760.* Mr. Schwarz received the B.S.E.Sc. degree from The Pennsylvania State University in 1983, and the M.S.E.E. degree from Ohio University in 1984 under a Stocker Fellowship. Since he joined IBM in 1984, his interest has been in minicomputers, with emphasis on computer arithmetic and parallel and pipelined architectures. He has received the First Patent Filed Award and the first level of the Author's Recognition Award, and has filed three patents.