VLSI Architectures for the Multiplication of Integers Modulo a Fermat Number

J. J. Chang
Electronic Parts Reliability Section

T. K. Truong
Communications Systems Research Section

I. S. Reed and I. S. Hsu
University of Southern California

Multiplication is central in the implementation of Fermat number transforms and other residue number algorithms. There is need for a good multiplication algorithm that can be realized easily on a VLSI chip. In this article, the Leibowitz multiplier is modified to realize multiplication in the ring of integers modulo a Fermat number. This new algorithm requires only a sequence of cyclic shifts and additions. The designs developed for this new multiplier are regular, simple, expandable, and, therefore, suitable for VLSI implementation.

I. Introduction

Fermat number transforms (FNTs) were developed to compute cyclic convolutions (Refs. 1 and 2) and to encode and decode a (256, 224) Reed-Solomon (R-S) code in $GF(F_3)$, where $F_3 = 2^{2^3} + 1 = 257$ (Ref. 3). Recently Leibowitz (Ref. 4) proposed the diminished -$1$ representation for binary arithmetic operations modulo $F_p$. Based on the ideas in Ref. 4, a new algorithm is developed to realize a multiplier over the ring of integers modulo a Fermat number. This algorithm requires only cyclic shifts and additions. An example, illustrating both the pipeline and systolic array aspects of this structure, is given for a multiplier in the field of $GF(2^{2^2} + 1)$.

II. Multiplication Modulo $F_t$

In this section, a new algorithm is developed for multiplication in the ring of integers modulo $F_t = 2^{2^t} + 1$. This new algorithm is illustrated by the example for $t = 2$. The same structure clearly extends to more general multiplication algorithms over $F_t$.

To perform efficiently, the binary arithmetic operations modulo $F_p$, the diminished -$1$ representation proposed by Leibowitz (Ref. 4), are used. Table 1 shows the correspondence of elements in $GF(2^{2^2} + 1)$ with their decimal equivalents in a normal binary representation and with their values in the
diminished -1 representation. In the Leibowitz diminished -1 representation, the most significant bit (MSB) can be viewed as the zero-detection bit.

To realize this new algorithm for multiplication, let \( A, B \) be two binary integers and \( A - 1 \), \( B - 1 \) their diminished -1 representations. It is now desired to perform a multiplication of the two positive numbers \( A - 1 \) and \( B - 1 \) in their diminished -1 representation. Note that if the MSB of either \( A - 1 \) or \( B - 1 \) is one, then the multiplication is inhibited, and the product is zero.

To accomplish this new multiplication process, one first translates \( B - 1 \) from the diminished -1 representation to the normal binary representation, \( B \). Then the multiplication of \( A - 1 \) and \( B \) is performed as follows:

\[
(A - 1) \cdot B = (A - 1) \left( \sum_{k=0}^{4} b_k 2^k \right) - \sum_{k=0}^{4} b_k 2^k \cdot (A - 1)
\]

(1)

Multiplying the diminished -1 numbers by the different powers of two in Eq. (1) so that the result is a diminished -1 number. This is achieved by the identity \( 2^k (A - 1) + (2^k - 1) = 2^k A - 1 \) for \( 0 \leq k \leq 4 \). Thus when considered as a sum of diminished numbers, Eq. (1) becomes

\[
(A - 1) B = \sum_{k=0}^{4} b_k \cdot (2^k A - 1) = \sum_{k=0}^{4} (d_k - 1)
\]

(2)

where \( d_k - 1 = b_k \cdot (2^k A - 1) \) for \( 0 \leq k \leq 4 \).

For \( d_k - 1 \) with \( 0 \leq k \leq 4 \) to be diminished -1 numbers, they must add correctly pairwise in accordance with the formula,

\[
(d_i - 1) + (d_j - 1) + 1 = (d_i + d_j - 1)
\]

(3)

It is readily verified that this holds in all four cases of \( (b_i, b_j) \). Hence the diminished -1 addition in Eq. (2) can be performed recursively in the following manner:

\[
(A - 1) \cdot B = (((((d_0 + d_1 - 1) + d_2 - 1) + d_3 - 1) + d_4 - 1) + 1)
\]

(4)

A substitution of Eq. (3) into Eq. (4) yields

\[
(A - 1) \cdot B = ((((d_0 + (d_1 - 1) + 1) + (d_2 - 1) + 1) + (d_3 - 1) + 1) + (d_4 - 1) + 1)
\]

(5)

where \( d_k - 1 = b_k (2^k A - 1) \).

From Eq. (5), the following identity is obtained for the product \( A \cdot B \) diminished by 1:

\[
(A - 1) \cdot B = b_0 \cdot (A - 1) + b_1 \cdot (2A - 1) + b_2 \cdot (2^2 A - 1) + b_3 \cdot (2^3 A - 1) + b_4 \cdot (2^4 A - 1) + 4
\]

\[= A \cdot B - S + 4 = (A \cdot B - 1) - S + 5\]

(6)

where \( S = b_0 + b_1 + b_2 + b_3 + b_4 \).

It follows from Eq. (6) that \( A \cdot B \) in diminished -1 notation is

\[
(A \cdot B - 1) = (A - 1) \cdot B - (4 - S) - 2 + 1
\]

(7)

Now let

\[
D = 4 - S
\]

(8)

where \( 0 \leq D \leq 4 \). Since \( 2^4 + 1 = 0 \) mod \( F_2 \), one has \( D = 2^4 + 1 - D = 2^4 - D - 1 = \overline{D} \), the binary one's complement of \( D \). Hence, by Eq. (7), the diminished -1 representation of \( A \cdot B \) becomes

\[
C = (A \cdot B - 1) = (A - 1) \cdot B + \overline{D} + 1
\]

(9)

where \( \overline{D} \) is the one’s complement of \( D \) in Eq. (8). A substitution of Eq. (4) into Eq. (9) yields

\[
(A \cdot B - 1) = (((((\overline{D} + b_0 \cdot (2^2 A - 1) + 1) + b_1 \cdot (2^1 A - 1) + 1) + b_2 \cdot (2^2 A - 1) + 1) + b_3 \cdot (2^3 A - 1) + 1) + b_4 \cdot (2^4 A - 1) + 1)
\]

(10)

as the new multiplication algorithm.

Let \( C_0 = \overline{D} \). Then the multiplication algorithm in Eq. (10) can be put into the following recursive form:

\[
C_{k+1} = C_k + b_k \cdot (2^{k+1} \cdot A - 1) + 1 \text{ for } 0 \leq k \leq 4
\]

(11a)

If one successively computes \( C_{k+1} \) in Eq. (11a) for \( 0 \leq k \leq 4 \), then the required result is obtained as follows:
\[ C_6 = C_4 + b_4 (2^4A - 1) + 1 = (A \cdot B - 1) = C \]  
(11b)

**Example 1:** A recursive diminished -1 multiplication algorithm: Let \( A = 10101 \), \( B = 00100 \); compute \( C = (A \cdot B - 1) = 01010 \times 00101 \mod 2^4 + 1 \).

To compute \( C \), one first translates \( B - 1 \) to \( B \). That is, \( B = B - 1 + 1 = 001101 + 1 = 001100 = b_4 b_3 b_2 b_1 b_0 \). From Eq. (10), the sequence of computations for \( 01010 \times 00101 \) is then as follows:

\[
\begin{align*}
01101 & \quad C_0 = \overline{D} = 01101 \\
+00000 & \quad b_0 \cdot (2^0A - 1) = 0(01010) = 00000 \\
01101 & \quad +1 \\
01110 & \quad C_1 = C_0 + b_0 \cdot (2^0A - 1) + 1 \\
+00100 & \quad b_1 \cdot (2^1A - 1) = 1(00100) = 00100 \\
10010 & \quad +1 \\
00010 & \quad C_2 = C_1 + b_1 \cdot (2^1A - 1) + 1 \\
+01001 & \quad b_2 \cdot (2^2A - 1) = 1(01001) = 01001 \\
01011 & \quad +1 \\
01100 & \quad C_3 = C_2 + b_2 \cdot (2^2A - 1) + 1 \\
+00000 & \quad b_3 \cdot (2^3A - 1) = 0(00100) = 00000 \\
01100 & \quad +1 \\
01101 & \quad C_4 = C_3 + b_3 \cdot (2^3A - 1) + 1 \\
+00000 & \quad b_4 \cdot (2^4A - 1) = 0(01001) = 00000 \\
01101 & \quad +1 \\
01110 & \quad C_5 = C_4 + b_4 \cdot (2^4A - 1) + 1 = C
\end{align*}
\]

Thus \( C = 01110 \) is the desired result of \( 01010 \) times \( 00101 \), modulo \( 2^4 + 1 \) in diminished -1 notation.

### III. A VLSI Structure for Implementing Multiplication Modulo \( F_1 \)

Example 1 of the new diminished -1 multiplication algorithm in the previous section shows that diminished -1 additions require the addition of the complement of an end around carry to its sum. A considerable speed improvement can be obtained by performing this operation simultaneously with the summation. A modified algorithm with this simultaneous addition is given for the previous example as follows:

**Example 2:** Modified recursive diminished -1 multiplication:

\[
\begin{align*}
11101 & \quad C_0 = 10000 + \overline{D} = 11101 \\
00000 & \quad b_0 \cdot (2^0A - 1) = 0(01010) = 00000 \\
+1 & \quad +0 \\
01101 & \quad C_1 = C_0 + b_0 \cdot (2^0A - 1) + 1 \\
00100 & \quad b_1(2^1A - 1) = 1(00100) = 00100 \\
+1 & \quad +1 \\
10010 & \quad C_2 = C_1 + b_1 \cdot (2^1A - 1) + 1 \\
01001 & \quad b_2(2^2A - 1) = 1(01001) = 01001 \\
+0 & \quad +0 \\
01011 & \quad C_3 = C_2 + b_2 \cdot (2^2A - 1) + 1 \\
00000 & \quad b_3(2^3A - 1) = 0(00100) = 00000 \\
+1 & \quad +1 \\
01100 & \quad C_4 - C_3 + b_3(2^3A - 1) + 1 \\
00000 & \quad b_4(2^4A - 1) = 0(01001) = 00000 \\
+1 & \quad +1 \\
01101 & \quad C_5 = C_4 + b_4(2^4A - 1) + 1 \\
00000 & \quad (2^4A - 1) = 0(01011) = 00000 \\
+1 & \quad +1 \\
01110 & \quad C = C_5 + 1
\end{align*}
\]

A possible VLSI structure for Example 2 is presented in Fig. 1. In Fig. 1, A, B, and C are 4-bit, 6-bit, and 5-bit registers, respectively. Initially, registers A, B, and C contain the multiplicand in the diminished -1 representation, the multiplier in normal representation, and \( 2^4 + 1 \), respectively. At the very same moment, \( C_{K+1} = C_K + b_K(2^KA - 1) \) is computed and
loaded into the C register. Simultaneously, the diminished -1 multiplication of \((A - 1)\) by 2 is performed first by a left cyclic shift of the four least-significant bits of the register A with the \(A_3\)-bit circulated into the first significant bit complemented. Also, at the same time, register B is shifted right by one bit. These operations are continued repetitively until the MSB of the register B is shifted out. The desired final result of 01110 after 5 iterations is obtained in register C.

The layout of the structure in Fig. 1 has been completed with the use of the CAESAR design tool (Ref. 5). The final layout of the multiplication chip is shown in Fig. 2.

In the future, logic and circuit simulations will be performed. These will be followed by chip fabrication and final testing. The total number of transistors in this chip is about 300. The area of the chip is estimated to be about 12,000 \(\lambda^2\).

References


Table 1. The correspondence among decimal numbers, their values in the normal binary representation, and in the diminished $-1$ representation

<table>
<thead>
<tr>
<th>Decimal number</th>
<th>Normal binary representation</th>
<th>Diminished $-1$ representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0 0 0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 0 1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 1 0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 1 1</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>0 0 1 0 0</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>0 0 1 0 1</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>0 0 1 1 0</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>0 0 1 1 1</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>0 1 0 0 0</td>
<td>9 (-8)</td>
</tr>
<tr>
<td>9 (-8)</td>
<td>0 1 0 0 1</td>
<td>10 (-7)</td>
</tr>
<tr>
<td>10 (-7)</td>
<td>0 1 0 1 0</td>
<td>11 (-6)</td>
</tr>
<tr>
<td>11 (-6)</td>
<td>0 1 0 1 1</td>
<td>12 (-5)</td>
</tr>
<tr>
<td>12 (-5)</td>
<td>0 1 1 0 0</td>
<td>13 (-4)</td>
</tr>
<tr>
<td>13 (-4)</td>
<td>0 1 1 0 1</td>
<td>14 (-3)</td>
</tr>
<tr>
<td>14 (-3)</td>
<td>0 1 1 1 0</td>
<td>15 (-2)</td>
</tr>
<tr>
<td>15 (-2)</td>
<td>0 1 1 1 1</td>
<td>16 (-1)</td>
</tr>
<tr>
<td>16 (-1)</td>
<td>1 0 0 0 0</td>
<td>0</td>
</tr>
</tbody>
</table>
Fig. 1. The pipeline architecture for the implementation of multiplication modulo the Fermat number $2^4 + 1$ using diminished $-1$ number representations.

Fig. 2. The VLSI nmos technology layout for multiplication modulo the Fermat number $2^4 + 1$, using diminished $-1$ number representation.