High-Speed Polynomial Basis Multipliers Over $GF(2^m)$ for Special Pentanomials
José L. Imaña

Abstract—Efficient hardware implementations of arithmetic operations in the Galois field $GF(2^m)$ are highly desirable for several applications, such as coding theory, computer algebra and cryptography. Among these operations, multiplication is of special interest because it is considered the most important building block. Therefore, high-speed algorithms and hardware architectures for computing multiplication are highly required. In this paper, bit-parallel polynomial basis multipliers over the binary field $GF(2^m)$ generated using type II irreducible pentanomials are considered. The multiplier here presented has the lowest time complexity known to date for similar multipliers based on this type of irreducible pentanomials.

Index Terms—Bit-parallel multipliers, finite field, $GF(2^m)$, irreducible pentanomials, polynomial basis.

I. INTRODUCTION

BINARY GALOIS field arithmetic is a widely studied subject due to its use in several important applications. $GF(2^m)$ arithmetic only requires AND and XOR gates for its implementation. XOR-based logic functions have been studied since the 1960s [1] due to their use in coding theory [2], digital signal processing, cryptography and telecommunication circuits. These applications frequently require efficient very large scale integration (VLSI) implementations of high speed $GF(2^m)$ multipliers [3]–[9]. For this reason, several bit-parallel polynomial basis (PB) multipliers have been proposed. Polynomial basis is the most widely used, although normal [10] or dual [11] basis can also be considered. The complexity of the multiplier depends on the generating irreducible polynomial $f(y)$ selected for the finite field. For hardware implementation of $GF(2^m)$ multiplication, low Hamming weight irreducible polynomials, such as trinomials and pentanomials, are usually used. For irreducible trinomials, multipliers with low area and time complexities can be implemented [12]–[14]. Unfortunately, there are 468 values of $m$ in the interval [2,1024] such that irreducible trinomials of degree $m$ do not exist. For each of the other values of $m$ in the same range, where no such irreducible trinomial exist, an irreducible pentanomial can be found. Thus, the design of multipliers using irreducible pentanomials is needed. Polynomial basis multiplication requires a polynomial multiplication followed by a modular reduction. An efficient bit-parallel multiplier was proposed by Mastrovito [15] in which a product matrix is introduced to combine the above two steps together. The entries in this matrix can be computed efficiently by sharing common items, known as subexpression sharing [16]. Mastrovito multipliers using special irreducible pentanomials have been widely studied due to their low-complexity implementations [12], [13], [17], [18]. All these works exploit subexpression sharing in order to find efficient architectures. Other methods use the divide-and-conquer approach for polynomial multiplication in order to reduce the complexity of the multiplier [19], [21]. In [9], a new PB multiplication method was used. This method is based on the introduction of a product matrix that can be decomposed as a sum of matrices depending on the selected irreducible polynomial. Matrix decomposition was already used in similar $GF(2^m)$ multiplication approaches that exploit subexpression sharing [12], [13], [17], [18]. The method in [9] introduced the functions $S_i$ and $T_i$ given by the raw sum of terms $x_i = (a_i b_i)$ and $x_{i,j} = (a_i b_j + a_j b_i)$, where $a_i, b_i \in GF(2)$ are the coefficients of two $GF(2^m)$ elements $A$ and $B$, respectively. The coefficients of the product of two field elements can be computed as the sum of that functions. One of the problems of the above method is related with the monolithic construction of the $S_i$ and $T_i$ functions. For example, for $GF(2^9)$ the functions $T_2 = x_2 + x_{3,1} = a_2 b_4 + (a_3 b_5 + a_5 b_3)$ and $T_4 = x_5 - a_5 b_3$ are defined. The sum of these two functions $T_2 + T_4 = ((a_2 b_4 + (a_3 b_5 + a_5 b_3)) + a_5 b_3)$ would result in a 3-level (with depth 3) binary tree of XOR gates. However, the sum of $T_2 + T_4$ involves the addition of four product terms $(a_2 b_4, a_3 b_5, a_5 b_3)$ and $a_5 b_3$ and it could be done with a 2-level complete binary tree of XOR gates.

In this work, a new bit-parallel PB multiplier is presented by considering the functions $S_i$ and $T_i$ as a sum of $S_{i,r}$ and $T_{i,1}$ terms, respectively, in such a way that $S_i = s^1_i s^0_i + \ldots + s^1_i s^1_i + s^0_i s^0_i$ and $T_i = t^1_i t^0_i + \ldots + t^1_i t^1_i + t^0_i t^0_i$ for a given finite field $GF(2^m)$, where $s^1_i, t^1_i \in GF(2)$ and $\rho = |log_2 m|$. The terms $S_{i,r}$ and $T_{i,j}$ represent the addition of $2^j$ products $a_k b_l$ and therefore can be implemented as a $j$-level complete binary tree of XOR gates. In this way, the addition of terms $S_{i,r}$ and $T_{i,j}$ with the same superscript $j$ would result in a $j + 1$-level complete binary tree. If the sum of the functions $S_i$ and $T_i$ is performed by grouping the additions of terms with the same $j$-level $S_{i,j}$ and $T_{i,j}$, then the number of XOR levels needed to compute the product coefficients can be reduced. Furthermore, the coefficients $(s^1_i, \ldots, s^1_i, s^0_i)$ and $(t^1_i, \ldots, t^1_i, t^0_i)$ are given by the binary representations of the subindex $i$ for $S_i$ and of the value $m - 1 - i$ for $T_i$, respectively. In this contribution, the new multiplication approach is applied to type II irreducible pentanomials [18] $f(y) = y^m + y^{m+n} + y^{m+n+1} + y^n + 1$, with $1 < n < \lfloor m/2 \rfloor - 1$. These pentanomials are important because they are abundant (there are 597 values of $m$ in the interval [2,1024]).
such that these type of irreducible pentanomials of degree \(m\) exist) and because all five binary fields recommended by NIST for ECDSA, i.e., \(m \in \{163, 233, 253, 409, 571\}\), can be constructed using such irreducible polynomials.

The paper is organized as follows. Notation and mathematical background are presented in Section II, where PB multiplication for type II irreducible pentanomials given in [9] is also reviewed. The new multiplication approach is presented in Section III, where an example of multiplication and the complexity analysis are also given. In Section IV comparisons with other similar multipliers are done. Finally, concluding remarks are made in Section V.

II. NOTATION AND PRELIMINARIES

Let \(f(y) = \sum_{i=0}^{n-1} f_i y^i\) be an irreducible polynomial of degree \(m\) over \(GF(2)\). All elements of the binary finite field \(GF(2^n) = GF(2)[y]/(f(y))\) can be represented in the polynomial basis \(\{1, y, \ldots, y^{m-1}\}\), where \(x\) is a root of the polynomial \(f(y)\). For example, an element \(A \in GF(2^m)\) is represented in PB as \(A = \sum_{i=0}^{m-1} a_i x^i = (1, x, \ldots, x^{m-1}) \cdot (a_0, a_1, \ldots, a_{m-1})^T\), with \(a_i \in GF(2)\). The vector of the coefficients of \(A\) in PB can be represented by \(A = (a_0, a_1, \ldots, a_{m-1})^T\). Let \(A, B, C \in GF(2^m)\) and \(A, B, C\) be their coefficient vectors, respectively. Using the method given in [9], the product \(C = A \cdot B\) can be computed as \(C^{R} = (c_{m-1}, \ldots, c_0) = A^T \cdot K\), where \(C^R\) is the vector of reversed coefficients of \(C\) and \(K\) is the product or Mastrovito matrix that depends on \(f(y)\) and on the coefficients \(b_i\) of \(B\). In order to compute the \(C\) coefficients, a new notation was given in [9]. These coefficients consist of sum-of-products (SOP) given by the inner products of \(A\) and \(B\). An inner product can be represented by the permutation given by the subscripts of the coefficients of \(A\) and \(B\), respectively, in the SOP. From this permutation, 1-cycles and 2-cycles, can be found and associated with the terms \(x_k\) and \(z_{i,j}\), respectively. For example, the SOP \(a_0b_1 + a_2b_3 + a_3b_2 + a_0b_0\) can be represented by the cycles \((0,4,1,3,2)\). In [9], the sum of the \(x_k\) and \(z_{i,j}\) represented by the 1-cycles \((k)\) and 2-cycles \((i,j)\) were carried out by the functions \(S_k\) and \(T_{i,j}\). These functions are implemented as binary trees of 2-input XOR gates with a lower level of 2-input AND gates (corresponding to the products of the coefficients of \(A\) and \(B\)). The product \(C = A \cdot B\) can be computed as the sum of these functions. The expression for \(S_k(1 \leq i \leq m)\) with \(\xi = [i/2]\), is [9]:

\[
S_k = x_k + \sum_{h=0}^{i-1} z_{h,i-h-1},
\]

where \(x_k - a_k b_k\) only appears for \(i\) odd. The expression for \(T_{i,j}(0 \leq i \leq m-2)\) with \(\gamma = \lceil m/2 \rceil + \lfloor i/2 \rfloor\) is as follows:

\[
T_{i,j} = x_{\gamma} + \sum_{j=1}^{\gamma} z_{i+j-1,m-j},
\]

where the term \(x_{\gamma}\) only appears for \((m \text{ and } i \text{ even})\) or for \((m \text{ and } i \text{ odd})\). In this case, \(\eta = \gamma\). Otherwise, i.e., for \((m \text{ even and } i \text{ odd})\) or for \((m \text{ odd and } i \text{ even})\), the term \(x_{\gamma}\) does not appear and the value of \(\eta = \lceil m/2 \rceil + \lfloor i/2 \rfloor\). For example, for \(GF(2^5)\) the terms \(S_k\) and \(T_{i,j}\) are as follows:

\[
\begin{align*}
S_0 & = S_1 + T_0 + T_{-2} + T_{-1} + T_2, \\
S_1 & = S_2 + T_1 + T_{-1} + T_1 + T_3, \\
S_2 & = S_3 + T_2 + T_{-1} + T_0 + T_2, \\
S_3 & = S_4 + T_3 + T_{-1} + T_{-2} + T_1 + T_2, \\
S_4 & = S_5 + T_4 + T_{-1} + T_{-2} + T_{-1} + T_1 + T_3 + T_{-3}, \\
\cdots
\end{align*}
\]

The paper is organized as follows. Notation and mathematical background are presented in Section II, where PB multiplication for type II irreducible pentanomials given in [9] is also reviewed. The new multiplier approach is presented in Section III, where an example of multiplication and the complexity analysis are also given. In Section IV comparisons with other similar multipliers are done. Finally, concluding remarks are made in Section V.

III. NEW MULTIPLIER FOR TYPE II IRREDUCIBLE PENTANOMIALS

The functions \(S_k\) and \(T_{i,j}\) presented in (1), (2) are given by a raw sum of terms \(x_k - (a_k b_k)\) and \(z_{i,j} - (a_k b_j + a_j b_i)\). The coefficients of the product of two field elements represented in PB can be computed as the sum of that functions, as given in Table I. One of the problems of the above method is related to the delay in the computation of the \(S_k\) term that appears in the coefficient \(c_{m-1}\) included in the section with the maximum number of terms, then \(c_{m-1}\) is the coefficient with the highest delay of the multiplier. The following section, a new scheme for multiplication is given.
with the monolithic construction of the $S_1$ and $T_1$ functions. For example, for $GF(2^8)$ the functions $T_2 = x_4 + x_{3,5} = (a_4 b_4 + (a_3 b_5 + a_5 b_3))$ and $T_4 = x_5 = a_5 b_5$ are defined. The sum of these two functions $T_2 + T_4 = ((a_4 b_4 + (a_3 b_5 + a_5 b_3)) + a_5 b_5)$, where the terms in brackets point out that they must be added (XOR) previously to the XOR with the other terms, would result in a 3-level (with depth 3) binary tree of XOR gates. However, the sum of $T_2 + T_4$ involves the addition of four product terms ($a_4 b_4$, $a_3 b_5$, $a_5 b_3$ and $a_5 b_5$) so it could be done with a 2-level complete binary tree of XOR gates if the involved additions could be performed in a separate way, i.e., if the product $a_4 b_4$ could be first added with the term $a_5 b_5$ and then perform the addition with ($a_3 b_5 + a_5 b_3$) in the form $T_2 + T_4 = ((a_4 b_4 + a_3 b_5) + (a_5 b_3 + a_5 b_5))$.

Algorithm 1 Computation of initial $S_1^j$ terms of $S_1$.

for $i = 1$ to $m$ do
   if odd $i$ then
      $S_1^i = x_{[i/2]}$;
   else
      $S_1^i = \varnothing$;
   end if
end for

for $k = 1$ to $\lfloor \log_2 m \rfloor$ do
   if $\lfloor i/2^k \rfloor \mod 2 - 1 \neq S_1^k$ at level $k$ then
      $S_1^k = \sum_{h=1}^{\pi} z_{h, \pi} + \sum_{h=1}^{\pi} z_{h+1, \pi} + l - \pi + 1$;
   else
      $S_1^k = \varnothing$;
   end if
end for

Algorithm 2 Computation of initial $T_1^j$ terms of $T_1$.

for $i = 0$ to $m - 2$ do
   if (even $m$ and $i$) or (odd $m$ and $i$) then
      $T_1^i = x_{[m/2] + [i/2]}$;
   else
      $T_1^i = \varnothing$;
   end if
   $i = i + 1$;
end for

for $k = 1$ to $\lfloor \log_2 m \rfloor$ do
   if $(m - i)/2^k \mod 2 - 1 \neq T_1^k$ at level $k$ then
      $T_1^k = \sum_{h=1}^{\pi} z_{h, \pi} + l - \pi + 1$;
   else
      $T_1^k = \varnothing$;
   end if
end for

In this paper, a new bit-parallel PB multiplier is presented by considering the functions $S_1$ and $T_1$ as a sum of $S_1^j$ and $T_1^j$ terms, respectively, in such a way that $S_1 = s_{t_1}^j S_1^j + \ldots + s_{t_1}^j S_1^j + t_0^j S_1^j$ and $T_1 = t_{t_0}^j T_1^j + \ldots + t_{t_0}^j T_1^j + t_{t_0}^j T_1^j$ for a given finite field $GF(2^n)$, where $s_{t_1}^j, t_{t_0}^j \in GF(2^n)$ and $\rho = \lfloor \log_2 m \rfloor$. The initial terms $S_1^j$ and $T_1^j$ represent the addition of $2^j$ products $a_k b_k$ and therefore can be implemented as a j-level complete binary tree of XOR gates. In this way, the addition of two terms $S_1^j$ and $T_1^j$ with the same superscript $j$ would result in a new XOR in the level $j + 1$ (i.e., a new $j + 1$-level term) that represents a $j + 1$-level complete binary tree. If the sum of the functions $S_1$ and $T_1$ is performed by grouping the additions of terms with the same $j$-level $S_1^j$ and $T_1^j$, starting with the lower levels, then the number of XOR levels needed to compute the product coefficients can be reduced. In this way, the $0$-level initial terms $S_1^0$ and $T_1^0$ should be first added in pairs to give rise to a new XOR in the level 1 (i.e., a new 1-level binary tree term), that in turn should be added with other 1-level term to give rise to a new 2-level complete binary tree and so on. If there is only one $j$-level term (or there is an unpaired $j$-level term), then it should be added with an immediately above $(j + 1)$-level term in order to have a new $(j + 2)$-level tree. If no such a $(j + 1)$-level term exists, then it should be added with a $(j + 2)$-level term, and so on.

From (1), (2), the computations of the initial terms $S_1^i$ and $T_1^i$ of $S_1$ and $T_1$ are given in Algorithm 1 and Algorithm 2, respectively, where the term $\pi = l = (2^{k - 1}) - 1$ has been used. In these algorithms, the condition $|i/2^k| \mod 2 = 1$ in the inner for loop determines if the $S_1$ or $T_1$ terms have an initial term $S_1^j$ or $T_1^j$ at level $k$. This condition will be further explained in Section III-B.

A characteristic of the previous representation is that the coefficients $(s_{t_1}^j, s_{t_1}^j, t_0^j, \ldots, t_{t_0}^j)$ are given by the binary representations of the subindex $i$ for $S_1$ and of the value $m - 1 - i$ for $T_1$, respectively. This can be deduced from the expressions of $S_1$ and $T_1$ defined in (1) and (2). For example, from (1) it can be observed that $S_1$ is given by the sum of $i$ product terms $a_k b_k$. As any number $i$ can be given as a sum of powers of 2, then $S_1$ can also be given as a sum of powers of 2 of product terms $a_k b_k$. The addition of $2^j$ products was previously denoted as $S_1^j$. Therefore, in the notation $S_1 = s_{t_1}^j S_1^j + \ldots + s_0^j S_1^j$, the coefficients $(s_{t_1}^j, \ldots, s_0^j)$ correspond with the binary representation of $i$. A similar reasoning can be done for $T_1$ considering that $T_1$ is given by the sum of $m - 1 - i$ product terms $a_k b_k$. Furthermore, in order to reduce the number of XORs needed for the computation of the product, common terms appearing in several coefficients can also be shared. These common terms correspond to the addition of consecutive $S_1$ and $T_1$ terms, i.e., $(S_1 + S_{i+1})$ and $(T_1 + T_{i+1})$, that lead to the addition of terms $(S_1^j + S_1^j)$ and $(T_1^j + T_1^j)$, respectively, for different levels $l$ determined by the binary representations of the subindex $i$ (for $S_1$).
and \( m - 1 - i \) (for \( T_1 \)). The addition of any pair of terms in level \( l \) creates a new term in level \( l + 1 \). From Table I, it can be noted that for the coefficients of the multiplier only common additions \( (T_i + T_{i+1}) \) can be found. Using the binary representations of \( m - 1 - i \), it can be observed that for even \( m \), the common sums are \( (T_i + T_{i+1}) \) for \( i = 0, \ldots, m - 4 \), while that for odd \( m \), the common terms are \( (T_i + T_{i+1}) \) for \( i = 1, \ldots, m - 4 \). The occurrence of these common groups in the coefficients of the product is studied in Appendix B.

The algorithm for the computation of the new proposed multiplication is given in Algorithm 3. In the first for loop, the common terms to be shared \( (T_i + T_{i+1}) \) are created, where \( \text{consecutive}^* \) refers to that for even \( m \), the subindex \( i \) ranges from 0 to \( m - 4 \), while that for odd \( m \), \( i \) ranges from 1 to \( m - 4 \). For each coefficient in Table I, the outer for loop processes (for each level \( l = 0, \ldots, \left\lfloor \log_2 m \right\rfloor \)) the initial \( S_i \) and \( T_i \) terms, creating new \((l+1)\)-level terms and sharing common terms (if any). The while loop processes terms from level \( \left\lfloor \log_2 m \right\rfloor + 1 \) to a level \( L \) with only two terms, in such a way that the maximum level will be \( L + 1 \) for the given coefficient. The execution of the algorithm will provide the coefficients of the product. The above new method of multiplication is clarified with the following example.

Algorithm 3 Computation of the product for \( GF(2^m) \).

```
compute \( S_i \) and \( T_i \) terms (using Algorithms 1 and 2).
for \( l = 0 \) to \( \left\lfloor \log_2 m \right\rfloor \) do
  find \( \text{consecutive}^* \) \( l \)-level terms \( T_i \) and \( T_{i+1} \);
  create common terms \( (T_i + T_{i+1}) \) in level \( l + 1 \);
end for
for each coefficient \( c_i \) in Table I do
  for \( l = 0 \) to \( \left\lfloor \log_2 m \right\rfloor \) do
    if \( \exists \ \text{consecutive}^* \) \( l \)-level terms \( T_i \) and \( T_{i+1} \) then
      share common \((l+1)\)-level terms \( (T_i + T_{i+1}) \);
      end if
    for the remaining \( l \)-level terms do
      sum \( l \)-level terms in pairs to create \((l+1)\)-level terms;
      if \( \nexists \) a non-paired \( l \)-level term then
        consider the term as a \((l+1)\)-level term
      end if
    end for
  end for
  \( l = l + 1 \);
  while the number of \( l - \) level terms \( \geq 2 \) do
    sum \( l \)-level terms in pairs to create \((l+1)\)-level terms;
    if \( \nexists \) a non-paired \( l \)-level term then
      consider the term as a \((l+1)\)-level term
    end if
  end while
end for
```

A. Multiplication Example over \( GF(2^{14}) \)

Let us consider the product \( C \) of two elements \( A \) and \( B \) in \( GF(2^{14}) \) generated by the type II irreducible pentanomial \( f(y) = y^{14} + y^6 + y^5 + y^4 + 1 \). The \( S_i \) and \( T_i \) functions can be computed using (1), (2) and are given in Table II. In this table, the \( S_i \) and \( T_i \) functions are the sum of the \( x_k \) and \( z_{-1-i} \) terms in the \( i \)-th row given in the second column. This column is divided into four subcolumns labeled as \( 2^0 \), \( 2^1 \), \( 2^2 \), and \( 2^3 \) that represent the number of product terms \( a_k b_i \) involved in each subcolumn. For example, \( S_{13} = x_6 + (x_{0,12} + x_{1,11}) + (x_{2,10} + x_{3,8} + x_{4,8} + x_{5,7}) \), where \( x_k \) involves \( 1 = 2 \) product term \((a_6 b_6),(x_{0,12} + x_{1,11})\) and \((x_{2,10} + x_{3,8} + x_{4,8} + x_{5,7})\) is the sum of \( s = 2^3 \) product terms \((a_6 b_8 + a_6 b_9) + (a_6 b_{11} + a_6 b_{12})\) and \((x_{2,10} + x_{3,8} + x_{4,8} + x_{5,7})\). The term \( S_{13} \) can then be represented in the form \( S_{13} = S_{0,13} + S_{1,13} + S_{2,13} + S_{3,13} \) where \( S_{0,13}, S_{1,13}, S_{2,13} \) and \( S_{3,13} \) stand for the terms with \( 2^0, 2^1, 2^2 \) and \( 2^3 \) product terms, respectively. In this case, \( S_{13} \) has not the term \( S_{3,13} \) because it is associated with the binary coefficient 0.

The four columns in Table II includes these binary representations. For example, the term \( S_{13} \) corresponds with the binary vector (1101) while that \( T_{2} \) corresponds with (1011) that is the binary representation of the value 11 (in this example with \( m = 14 \)). It can be observed that the \( S_i \) and \( T_i \) terms given in Table II can also be computed using Algorithm 1 and Algorithm 2.

In order to reduce the space complexity of the multiplier, common terms appearing in several coefficients can also be shared. It can be observed in Table II that consecutive \( S_i \) and \( T_i \) terms have \( S_i \) and \( T_i \) terms with the same level \( j \). For example, \( S_{10} \) and \( S_{11} \) have 1-level terms \( S_{10} \) and \( S_{11} \) and 3-level terms \( S_{10} \) and \( S_{11} \), respectively. The addition of \( S_{10} \) and \( S_{11} \) then implies the sums \( S_{10} + S_{11} \) and \( S_{10} + S_{11} \) that give rise to 2-level and 4-level complete binary trees of XOR gates, respectively. Therefore, the group given by the addition of these two functions \( S_{10} \) and \( S_{11} \) can reduce the complexity. In Table II the groups that can be found are represented by shadowed cells with the same color. The \( S \) groups are \( \{S_0, S_3\} \), \( \{S_4, S_5\} \), \( \{S_6, S_7\} \), \( \{S_8, S_9\} \), \( \{S_{10}, S_{11}\} \) and \( \{S_{12}, S_{13}\} \). While the \( T \) groups are \( \{T_0, T_1\} \), \( \{T_2, T_3\} \), \( \{T_4, T_5\} \), \( \{T_6, T_7\} \), \( \{T_{8}, T_{9}\} \) and \( \{T_{10}, T_{11}\} \). The sum of the \( S_i \) and \( T_i \) terms that appear in the coefficients of the product must be done using the above groups in order to optimize the implementation.

The coefficients of the product are given in Table III using Table I for this \( GF(2^{14}) \) irreducible pentanomial. The previous
TABLE II

<table>
<thead>
<tr>
<th>$S_i$</th>
<th>$T_i$</th>
<th>$S_{i}^0$</th>
<th>$S_{i}^1$</th>
<th>$S_{i}^2$</th>
<th>$S_{i}^3$</th>
<th>$T_{i}^0$</th>
<th>$T_{i}^1$</th>
<th>$T_{i}^2$</th>
<th>$T_{i}^3$</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_2$</td>
<td>$x_0, 1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0010</td>
</tr>
<tr>
<td>$S_3$</td>
<td>$x_1, 2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0011</td>
</tr>
<tr>
<td>$S_4$</td>
<td>$(x_0, 3, x_2)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0100</td>
</tr>
<tr>
<td>$S_5$</td>
<td>$(x_0, 4, x_1)$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0101</td>
</tr>
<tr>
<td>$S_6$</td>
<td>$(x_2, 5, x_2)$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0110</td>
</tr>
<tr>
<td>$S_7$</td>
<td>$(x_3, 6, x_2)$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0111</td>
</tr>
<tr>
<td>$S_8$</td>
<td>$(x_2, 7, x_0, 1)$</td>
<td>$(x_2, 8, x_0, 1, 2)$</td>
<td>$(x_2, 9, x_0, 1, 2, 3)$</td>
<td>$(x_2, 10, x_0, 1, 2, 3, 4)$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0000</td>
</tr>
<tr>
<td>$S_9$</td>
<td>$(x_2, 11, x_0, 1, 2)$</td>
<td>$(x_2, 12, x_0, 1, 2, 3)$</td>
<td>$(x_2, 13, x_0, 1, 2, 3, 4)$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0100</td>
<td></td>
</tr>
<tr>
<td>$S_10$</td>
<td>$(x_2, 14, x_0, 1, 2)$</td>
<td>$(x_2, 15, x_0, 1, 2, 3)$</td>
<td>$(x_2, 16, x_0, 1, 2, 3, 4)$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0110</td>
<td></td>
</tr>
<tr>
<td>$S_11$</td>
<td>$(x_2, 17, x_0, 1, 2)$</td>
<td>$(x_2, 18, x_0, 1, 2, 3)$</td>
<td>$(x_2, 19, x_0, 1, 2, 3, 4)$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0111</td>
<td></td>
</tr>
<tr>
<td>$S_12$</td>
<td>$(x_2, 21, x_0, 1, 2)$</td>
<td>$(x_2, 22, x_0, 1, 2, 3)$</td>
<td>$(x_2, 23, x_0, 1, 2, 3, 4)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>$S_13$</td>
<td>$(x_2, 24, x_0, 1, 2)$</td>
<td>$(x_2, 25, x_0, 1, 2, 3)$</td>
<td>$(x_2, 26, x_0, 1, 2, 3, 4)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>$S_14$</td>
<td>$(x_2, 27, x_0, 1, 2)$</td>
<td>$(x_2, 28, x_0, 1, 2, 3)$</td>
<td>$(x_2, 29, x_0, 1, 2, 3, 4)$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1000</td>
<td></td>
</tr>
</tbody>
</table>

$S$ and $T$ groups found in the product coefficients are shadowed in Table III. It can be observed that only one $S_i$ term appears in each coefficient, so only the $T$ groups are used. The group $(T_8, T_9)$ appears in three coefficients $(c_0, c_4)$ and $(c_1)$ and the groups $(T_0, T_1), (T_2, T_3), (T_4, T_5), (T_6, T_7)$ and $(T_{10}, T_{11})$ appear in two coefficients. This means that only one of each of the above groups must be implemented and therefore the other occurrences of the groups must not be implemented. The number of XOR gates that can be reduced is given by the number of $T_i^j$ terms in each group. Using Table II it can be observed that the group $(T_0, T_1)$ involves the sum of two terms $T_0^j + T_1^j$ and $T_0^j + T_1^j$ and therefore it requires 2 XOR gates. In the same way, the groups $(T_2, T_3), (T_4, T_5), (T_6, T_7), (T_8, T_9)$ and $(T_{10}, T_{11})$ require 2, 1, 2, 1, and 1 XOR gates, respectively. Furthermore, as $(T_8, T_9)$ appears in three coefficients, then the number of XOR gates that can be reduced will be 2 times the number of XOR gates required, i.e., 2 · 1 = 2. Therefore, the number of XOR gates that will be reduced for the above groups will be, respectively, $2 + 2 + 2 + 2 + 1 = 10$ XOR. In Section III-B, general expressions will be given in order to compute the number of XOR gates that can be reduced due to the groups.

Using the $S_i^j$ and $T_i^j$ terms given in Table II for $S_i$ and $T_i$, respectively, the coefficients of the product are shown in Table IV. In this table, the sum of terms is accomplished using the rule previously given, i.e., the sum of the functions $S_i$ and $T_i$ is performed by grouping the sums of terms with the same $j$-level $S_i^j$ and $T_i^j$, starting with the lower levels. In this way, the 0-level initial terms $S_i^0$ and $T_i^0$ should be first added in pairs to give rise to new 1-level binary trees (1-level terms), that in turn should be added in pairs with other 1-level terms to give rise to new 2-level complete binary trees and so on. If there is only one $j$-level term (or there is an unpaired $j$-level term), then it should be added with an immediately above $(j + 1)$-level term in order to have a new $(j + 2)$-level tree. If no such a $(j + 1)$-level term exists, then it should be added with a $(j + 2)$-level term, and so on. The order of the additions of the terms in Table IV is represented by means of parenthesis. To reduce the number of XORs needed for the computation of the product, common terms appearing in several coefficients can also be shared. This is represented with shadowed boxes that correspond with the previously stated groups.

In order to illustrate the method, the implementation of the coefficient $c_6$ is given in Fig. 1. This coefficient requires the addition of 8 terms, so it is the most complex coefficient and determine the maximum delay of the multiplier (in fact, the $c_{6,2}$ coefficient of a $GF(2^m)$ multiplier given by Type II pentanomial is the most complex one, so it is used to determine the maximum delay complexity). In this figure, the $S_i$ and $T_i$ terms are represented by filled circles. These circles correspond to the initial $S_i^j$ and $T_i^j$ terms given in Table II in such a way that...
TABLE IV

<table>
<thead>
<tr>
<th>Coefficients $c_i$ of the Product for $GF(2^{14})$ With $S_i^x$ and $T_i^y$ Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_0 = \left{ \left( S_0^y + T_0^y \right) + \left( T_0^y + T_1^y \right) + \left( T_1^y + T_2^y \right) + 1 \right}$</td>
</tr>
<tr>
<td>$c_1 = \left{ \left( S_1^y + T_0^y \right) + \left( T_1^y + T_1^y \right) + \left( T_2^y + T_2^y \right) + 1 \right}$</td>
</tr>
<tr>
<td>$c_2 = \left{ \left( S_2^y + T_0^y \right) + \left( T_0^y + T_1^y \right) + \left( T_2^y + T_2^y \right) + 1 \right}$</td>
</tr>
<tr>
<td>$c_3 = \left{ \left( S_3^y + T_0^y \right) + \left( T_1^y + T_2^y \right) \right}$</td>
</tr>
<tr>
<td>$c_4 = \left{ \left( S_0^y + T_0^y \right) + \left( T_0^y + T_1^y \right) + \left( T_2^y + T_3^y \right) \right}$</td>
</tr>
<tr>
<td>$c_5 = \left{ \left( S_0^y + T_0^y \right) + \left( T_1^y + T_2^y \right) \right}$</td>
</tr>
<tr>
<td>$c_6 = \left{ \left( S_0^y + T_0^y \right) + \left( T_2^y + T_3^y \right) \right}$</td>
</tr>
<tr>
<td>$c_7 = \left{ \left( S_0^y + T_0^y \right) + \left( T_1^y + T_2^y \right) \right}$</td>
</tr>
<tr>
<td>$c_8 = \left{ \left( S_0^y + T_0^y \right) + \left( T_1^y + T_2^y \right) \right}$</td>
</tr>
<tr>
<td>$c_9 = \left{ \left( S_0^y + T_0^y \right) + \left( T_1^y + T_2^y \right) \right}$</td>
</tr>
<tr>
<td>$c_{10} = \left{ \left( S_0^y + T_0^y \right) + \left( T_1^y + T_2^y \right) \right}$</td>
</tr>
<tr>
<td>$c_{11} = \left{ \left( S_0^y + T_0^y \right) + \left( T_1^y + T_2^y \right) \right}$</td>
</tr>
<tr>
<td>$c_{12} = \left{ \left( S_0^y + T_0^y \right) + \left( T_1^y + T_2^y \right) \right}$</td>
</tr>
<tr>
<td>$c_{13} = \left{ \left( S_0^y + T_0^y \right) + \left( T_1^y + T_2^y \right) \right}$</td>
</tr>
</tbody>
</table>

$S_7 - S_0^y + S_1^y + S_2^y + T_0^y + T_1^y + T_2^y + T_3^y, T_0 - T_0^y + T_0^y + T_3^y, T_1 - T_1^y + T_3^y, T_2 = T_0^y + T_1^y + T_3^y, T_3 = T_0^y + T_3^y + T_3^y, T_4 = T_0^y + T_0^y + T_0^y, T_5 = T_0^y + T_0^y + T_0^y, T_6 = T_0^y + T_0^y + T_0^y, T_7 = T_0^y + T_0^y + T_0^y$. Vertical dashed lines represent the level of XOR binary trees. For example, the term $S_7^y$ in line 2 represent the 2-level binary tree $S_7^y = (s_1 + s_2) - (a_1b_5 + a_2b_1) + (a_2b_4 + a_1b_2)$. Circles enclosed within ellipses represent the terms of the corresponding $S_i$ and $T_i$ functions. For example, the $T_0$ function is given by the three initial terms $T_0^0, T_0^2, T_0^3$. Furthermore, the gray color XOR trees represent the group ($T_0, T_1$) that can be shared in several coefficients (in this case, the group correspond with the additions $T_0^2 + T_1^2$ and $T_0^3 + T_1^3$). It can be observed in Fig. 1 that the addition of terms follows the rule previously given, starting with 0-level terms and ascending in the construction of binary XOR trees. For example, the addition of the initial 0-terms $S_0^y$ and $T_0^0$ gives rise to a new 1-level term (a new XOR in level 1) that in turn is added with the initial 1-term $S_1^y$ to give rise to a new 2-level XOR term and so on. In this example, $c_4$ can be constructed with a 6-level binary XOR tree so the delay complexity of the multiplier is given by $T_A + 6T_X$, where $T_A$ and $T_X$ represent the delay of 2-input AND and XOR gates, respectively. The $T_A$ delay corresponds to the 0-level $a_i b_j$ products of the coefficients of $A$ and $B$. It must be noted that the best delay complexity for this multiplier given by other similar methods in the literature is $T_A + 7T_X$.

The space complexity (number of AND and XOR gates) can also be computed. The number of AND gates is given by all the different products $a_i b_j$, with $i, j \in [0, \ldots, m - 1]$. This number can be computed using (1), (2) and for this $GF(2^{14})$ example is given as 196 AND gates (see also Table II). It is proved in Appendix B that the number of AND gates for a $GF(2^m)$ multiplier is $m^2$. The XOR gates can be computed as the sum of XOR gates in the initial $S_i^x$ and $T_i^y$ terms (as given in Table II) plus the number of new XOR gates generated in the coefficients (as given in Table IV) minus the number of XOR gates due to the groups shared among coefficients. The $S_i^x$ and $T_i^y$ terms perform the XOR of $2^i$ product terms, therefore the number of XORs is $2^i - 1$. In this example, there are 7 $S_i^x$, $T_i^x$, $S_i^y$ and $T_i^y$ terms each. Therefore the number of XOR gates in the initial $S_i^x$ terms will be $7 \cdot (2^6 - 1) + 7 \cdot (2^4 - 1) + 7 \cdot (2^2 - 1) + 7 \cdot (2^3 - 1) = 77$ XOR. There are also 7 $T_i^y$ terms and 6 $T_i^x$, $T_i^y$ and $T_{2i}^y$ terms each, so the number of XOR gates in the initial $T_i^y$ terms is $7 \cdot (2^6 - 1) + 6 \cdot (2^4 - 1) + 6 \cdot (2^2 - 1) + 6 \cdot (2^3 - 1) = 66$ XOR. The number of new XOR gates generated in the coefficients for the sum of $S_i^x$ and $T_i^x$ terms can be found in Table IV and in this
case is 134 XOR. Finally, the number of XORs due to the groups shared among coefficients were previously computed and it was found to be 10 XOR. Therefore the total number of XOR of this multiplier is 77 + 66 + 134 − 10 = 267 XOR.

B. Complexity Analysis

General expressions for time and space complexities for the GF(2^m) multiplier are given in this subsection.

1) Time Complexity: The coefficients in Table I have been divided into seven sections, depending on the number of S_j and T_i terms in the sums. The first section A (from c_0 to c_{n-2}) has 5 terms; section B with c_{n-1} = c_0 and c_{n+1} has 4, 7 and 6 terms, respectively; section C (c_{n+2} to c_{2n-2}) has 8 terms; sections D (c_{2n-1}, c_{2n}) and E (c_{2n+1}, c_{2n+2}) have 7 and 6 terms, respectively; section F (c_{2n+3} to c_{m-2}) has 5 terms; and finally section G (c_{m-1}) has 4 terms. It can be observed in Table II that the term T_{0} has the highest complexity among T_j terms. The coefficient c_{n+2}, that is included in section G with the maximum number of terms (8), includes this complex term T_0. Therefore, c_{n+2} is the most complex coefficient of a GF(2^m) multiplier given by Type II irreducible pentanomials and it will be used to determine the highest delay of the multiplier.

In order to do that, the complexity of the S_j and T_i terms must be determined. Their complexity depends on the number of the initial S_j and T_i terms they have. These terms can be represented as S_j = s_1^j S_{0}^j + ... + s_t^j S_{t}^j and T_i = t_1^i T_{0}^i + ... + t_t^i T_{t}^i for a given finite field GF(2^m), where s_1^j, t_1^i ∈ GF(2) and p = [log_2 m]. Therefore, the coefficients s_t^j, t_t^i determine if the corresponding S_j, T_i appear in S_j and T_i, respectively. As previously proved, the coefficients (s_1, s_2, ..., s_t^j) and (t_1, t_2, ..., t_t^i) are given by the binary representations of the subindex i for S_j and by the value m − 1 − i for T_i, respectively. Therefore the study of the number of T_i terms in T_j can be reduced to the study of S_j terms in S_j using the equivalence (only in relation to the number of terms) T_j = S_{m-1-j}. For the most complex coefficient c_{n+2}, this equivalence results in that T_{n+2} = S_m, T_{0} = S_{m-1}, T_{x} = S_{n+1}, T_{z} = S_{n+2}. The binary representation of n + 3, z − 3, m − 1, n + 1, n − 1, m − 2, m − 3 and n − 3 must then be determined. The binary configuration of a number x can be given by the expression

\[ x - \sum_{i=0}^{\lfloor \log_2 x \rfloor} \left( \frac{x}{2^i} \mod 2 \right) \cdot 2^i. \]  

(3)

The value \( \lfloor x/2^i \rfloor \mod 2 \) determines if the binary representation of x has a 1 in the position with weight 2^i in such a way that if \( \lfloor x/2^i \rfloor \) has an even value, then x has a 0 while that if \( \lfloor x/2^i \rfloor \) has an odd value, then x has a 1. A 1 in the position with weight 2^i for the binary representation of x will represent that the terms S_x, T_{m-1-x} have a term S_x, T_{m-1-x} that is the sum of 2^i product terms and that is implemented by means of a binary XOR tree with depth i (an i-level binary tree).

In order to compute the depth of the binary tree of XOR gates in GF(2^m) given by the coefficient c_{n+2}, the number of total terms in the \( [\log_2 m] \)-level must first be determined. The initial levels for a given m are [0, 1, 2, ..., [log_2 m]]. For a given level i, the number of new XOR terms that will result in level i + 1 due to the addition in pairs of the i-level terms is given by \( \lfloor n_i \rfloor = \frac{\lfloor n - i \rfloor}{2} \). For example, in Fig. 1 there are seven 0-level terms (S_{0}^6, T_{0}^5, T_{6}^0, T_{2}^6, T_{6}^2, T_{10}^6, T_{12}^0) whose sum gives rise to the four 1-level XOR terms (S_{0}^6 + T_{0}^6, T_{2}^2 + T_{6}^2, T_{6}^6 + T_{12}^6, T_{10}^6 + T_{12}^0) besides the term T_{12}^0, that can also be considered as a 1-level term (in order to be added to T_{10}^6 and result in a new 2-level XOR term).

Let \( \mu_j \) be the number of initial terms S_j and T_i in level j. This number will be given by the terms S_j in the previously computed equivalent expression (in relation to the number of terms) \( c_{n+2} \equiv S_{n+3} + S_{n-3} + S_{m-1} + S_{n+1} + S_{n-1} + S_{m-2} + S_{m-3} + S_m \). In order to do that, the binary representation of n + 3, z − 3, m − 1, n + 1, n − 1, m − 2, m − 3, and n − 3 must then be determined. Using (3), the value \( \lfloor x/2^i \rfloor mod 2 \) determines if the term S_j has an initial term S_j in the position with weight 2^i, i.e., in level j. Representing \( \lfloor x/2^i \rfloor mod 2 \) as \( \lfloor x/2^i \rfloor \), the number of initial terms S_j and T_i in level j (i.e., \( \mu_j \)) can be computed as follows:

\[
\frac{m - 1}{2^i} \ast \frac{m - 2}{2^i} + \frac{m - 3}{2^i} + \frac{n + 3}{2^i} + \frac{n + 1}{2^i} \ast \frac{n - 3}{2^i} + \frac{n - 1}{2^i} + \frac{z - 3}{2^i} + \frac{z - 3}{2^i}.
\]  

(4)

It must be noted that in (4) the fourth addend \( \lfloor n - 3 \rfloor / 2^i \) corresponds with the real \( S_{n+3} \) term, while the rest of addends corresponds with the equivalence previously given \( S_{m-1} \equiv T_i \). Using (4), the number of initial terms S_j and T_i for the coefficient c_0 given in the example in Section III-A can be computed. The number of initial terms in level 3, for example, will be \( \mu_3 = [3/4]^i + [1/2]^i + [11/4]^i + [7/4]^i + [5/4]^i + [1/4]^i + [3/4]^i + [7/4]^i \) = 1 + 1 + 0 + 1 + 1 + 0 + 0 + 1 = 5 corresponding with the initial terms \( S_{7}^0 = T_3^0, S_{7}^1 = T_1^0, S_{7}^2 = T_2^0, S_{7}^3 = T_1^1, S_{7}^4 = T_2^1, S_{7}^5 = T_1^2, S_{7}^6 = T_2^2 \), respectively, that are represented in Fig. 1 as filled (black and gray) circles.

If \( \mu_0, \mu_1, \ldots, \mu_{[log_2 m]} \) denote the number of initial terms S_j and T_i in levels 0, 1, ... , [log_2 m], respectively, then the total number of terms in the \( [log_2 m] \)-level (denoted by \( M_{[log_2 m]} \)) will be the addition of the initial terms in that level \( \mu_{[log_2 m]} \) plus the terms created due to the addition of terms in lower levels. In Section A of Appendix A, it is proved that these terms created in level \( [log_2 m] \) due to the addition of terms in levels 0, 1, ... , [log_2 m] − 1 is given by the expression:

\[
\left( \mu_0 + 2\mu_1 + 2^2\mu_2 + \ldots + 2^{[log_2 m] - 1}\mu_{[log_2 m] - 1} \right) \frac{4m + 3n - 9}{2^{[log_2 m]}}.
\]  

(5)

Therefore, the total number \( M_{[log_2 m]} \) of terms in the \( [log_2 m] \)-level will be the sum of \( \mu_{[log_2 m]} \) plus the expression in (5). In Appendix A it is proved that this addition is:

\[
M_{[log_2 m]} = \frac{4m + 3n - 9}{2^{[log_2 m]}}.
\]  

(6)

The sum in pairs of the terms \( M_{[log_2 m]} \) determined in (6) will determine the final level reached to compute the c_{n+2} coefficient. Therefore the number of XOR levels needed to compute this coefficient will be \( [log_2 m] + [log_2 M_{[log_2 m]}] \). Finally, the highest delay of the GF(2^m) multiplier based on type II pentanomials given by the c_{n+2} coefficient is:

\[
T_A + \left( [log_2 m] + [log_2 \left( \frac{4m + 3n - 9}{2^{[log_2 m]}} \right)] \right) T_X.
\]  

(7)
In order to compare this time complexity with other multipliers found in the literature, in Section B of Appendix A the following upper bound is derived for the XOR delay of the multiplier:

\[
\log_2 m + \log_2 M \log_2 m \leq 3 + \log_2 (m + 1) \tag{8}
\]

2) Area Complexity: In order to determine the area complexity of the PB multiplier given in Table I, the number of AND and XOR gates of the \(S_i\) and \(T_i\) terms must be known. In this work, these terms have been considered as a sum of \(S_i^j\) and \(T_i^j\) terms, in such a way that \(S_i = s_i^1 S_i^1 + \ldots + s_i^n S_i^n\) and \(T_i = t_i^1 T_i^1 + \ldots + t_i^n T_i^n\) for a given finite field \(GF(2^m)\), where \(s_i^j, t_i^j \in GF(2)\) and \(\rho = \log_2 m\). In Table I, the coefficients of the product are given as sums of \(S_i\) and \(T_i\) terms where their corresponding components \(S_i^j\) and \(T_i^j\) are considered as individual terms when performing the sum. It can be observed that the \(S_i\) terms, \(i = 1, 2, \ldots, m\), appears only once while that the \(T_i\) terms, \(i = 0, 1, \ldots, m - 2\), appear several times.

One way to determine the number of AND and XOR gates of the \(S_i\) and \(T_i\) terms is to count the number of AND and XOR gates given by the sum of terms \(x_k z_{i,j}\) in (1), (2). In this way, we compute the total number of AND gates of the multiplier, the XORs of the \(S_i^j\) and \(T_i^j\) terms, the XORs needed for the sum of all the \(S_i^j\) terms of \(S_i\) and the XORs needed for one sum of the \(T_i^j\) terms of \(T_i\), i.e., we count the XORs due to the contribution of all the \(S_i\) terms and of one occurrence of the \(T_i\) terms. If a term \(T_i\) appears \(p_i\) times in the additions given for the coefficients in Table I, then the other \(p_i - 1\) occurrences are taken into account by computing the number of XORs needed for the sum of the \(T_i^j\) terms of \(T_i\) and multiplying it by \(p_i - 1\). This must be done for each \(T_i\), \(i = 0, 1, \ldots, m - 2\). To determine the area complexity of the PB multiplier, the number of XOR gates needed for the sum of the \(S_i\) and \(T_i\) terms in the product coefficients of Table I and the number of shared groups \((T_i, T_j)\) that appear in the product coefficients should also be computed. This number of groups must be subtracted from the previous XOR gates computed. Therefore, the following figures must be computed to obtain the ORs of the multiplier:

1. The number of XOR gates given by \(S_i\) and \(T_i\) in (1), (2).
2. The number of XOR gates needed for the sum of the \(S_i\) and \(T_i\) terms in the product coefficients.
3. For each \(T_i\), the number \(p_i\) of times that \(T_i\) appears in Table I and the number \(\Theta_i\) of XORs needed for the sum of the \(T_i^j\) terms of \(T_i\) must be determined. Then the XOR gates given by \(\sum_{i=0}^{m-2} (p_i - 1) \cdot \Theta_i\) must be computed.
4. The number of XOR gates given by the shared groups \((T_i, T_j)\) that appear in the product coefficients.

The XOR gates of the multiplier will be \(1 \oplus 2 \oplus 3 \oplus 4 \oplus \). In Appendix B the following values have been computed:

- The total number of AND gates of the multiplier is \(m^2\).
- The number \(1\) of XOR gates given by \(S_i\) and \(T_i\) in (1), (2) is \((m - 1)^2\).
- The number \(2\) of XOR gates needed for the sum of the \(S_i\) and \(T_i\) terms in the product coefficients is \(4m + 3m - 2\).
- The number \(3\) of XOR gates can be computed by \(\sum_{i=0}^{m-2} \Theta_i \cdot 3 \cdot \gamma_{m-1} + 3 \cdot \gamma_{n-1} - 3\cdot \gamma_{m-2}\) where the number of XOR gates \(\Psi_n\) needed for the sum of the \(S_i\) terms of \(S_n\) is given by:

\[
\Psi_n = \left( n - \sum_{j=1}^{\log_2 m} \left\lceil \frac{n}{2^j} \right\rceil \right) - 1 - H_n - 1 \tag{9}
\]

where \(H_n\) is the Hamming Weight of \(n\) and where \(\gamma_h = \sum_{i=1}^{h} \Psi_i\) is given as:

\[
\gamma_h = \sum_{i=1}^{h} \Psi_i - \frac{h(h - 1)}{2} - \sum_{i=1}^{h} \left\lceil \frac{|\log_2 i|}{2} \right\rceil \tag{10}
\]

- The number of XOR gates \(\gamma_j\) given by the shared groups \((T_i, T_j)\) that appear in the product coefficients is:

\[
\sum_{i=2,4,6,\ldots} \gamma_i + \gamma_2 \cdot \gamma_{p_i-1} - 3 \cdot \gamma_{m-1} \cdot \gamma_{m-2} \tag{11}
\]

A more compact expression for (12) could not be found. The functions \(\gamma_m\) and \(\gamma_m - \sum_{i=2,4,6,\ldots} \gamma_i\) could be computed for any value of \(m\) using Maple. In Table VII the values of these functions for \(m \in [8, 100]\) are given. Using Table VII, it can be observed that for the example given in Section III-A with \(m = 14, k = 4\), the values \(\gamma_{13} = 12\) and \(\gamma_{14} = 9\). In this example, the values \(\gamma_{5} = 2\) and \(\gamma_{2} = 1\) can also be computed.

Applying the above values to (12) we have \(196 + 28 + 12 + 3 \cdot (12 + 2) - 1 - 9 - 1 = 267\) XOR gates, matching the result given in Section III-A.

IV. COMPARISON WITH OTHER PB MULTIPLIERS

In Table V the theoretical complexities obtained by the approach here proposed are compared with the best results known to date for bit-parallel polynomial basis multipliers over \(GF(2^m)\) generated by type II irreducible pentanomials. In (8) it was proved that \(\log_2 m + \log_2 M \leq 3 + \log_2 (m + 1)\). It can also be observed that \(3 + \log_2 (m + 1) > 3 + \log_2 (m - 2)\), where \(3 + \log_2 (m - 2)\) is the best XOR delay found in the literature for this type of bit-parallel multipliers [9]. Simulations have been done using Maple that have proved that the delay of our multiplier is less than or equal to the delay in [9], i.e., \(\log_2 m + \log_2 M \leq 3 + \log_2 (m - 2)\). From the simulation results, it was found that for the 593 different values of \(m\) in the interval \(m \in [8, 100]\) for which an irreducible type II pentanomial exists, the proposed multiplier has the smallest delay in 465 different values of the field size \(m\). More specifically, among the type II irreducible pentanomials existing in \(m \in [8, 100]\), there are 477 and 1162 different combinations of \((m, n)\) for which the proposed multiplier has equal and less delay, respectively, than the multiplier in [9]. With respect to area complexity, it was found that the proposed multiplier presents equal number of AND gates in comparison with the other similar multipliers existing in the literature (except for the approach presented in [21]) and a higher number of XOR gates in comparison with the other multipliers. This increased
TABLE V
COMPLEXITIES OF BIT-PARALLEL PB MULTIPLIERS FOR $f(x) = x^m + x^{n+2} + x^{n+1} + 1$

<table>
<thead>
<tr>
<th>#AND</th>
<th>#XOR</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>[13]</td>
<td>$m^2$</td>
<td>$m^2 + 2m - 3$</td>
</tr>
<tr>
<td>[12]</td>
<td>$m^2$</td>
<td>$m^2 + 2m - 3$</td>
</tr>
<tr>
<td>[18]</td>
<td>$m^2$</td>
<td>$m^2 + 2m - 3$</td>
</tr>
<tr>
<td>[6]</td>
<td>$m^2$</td>
<td>$m^2 + (m + m^2) - k$ with $k \in [2, 3]$</td>
</tr>
<tr>
<td>[9]</td>
<td>$m^2$</td>
<td>$m^2 + (m^2 + 2m - 3)$</td>
</tr>
<tr>
<td>[21]</td>
<td>$3m^2 + 2m - 1$</td>
<td>$3m^2 + 2m - 1$</td>
</tr>
</tbody>
</table>

This work $m^2$ $m^2 + 2m + 3n + 3(Y_{m-1} + Y_{n+1}) - H_n - Y_m - H_{m-1}$

TABLE VI
COMPLEXITIES OF BIT-PARALLEL PB MULTIPLIERS USING TYPE II PENTANOMIALS FOR THE FIVE RECOMMENDED NIST FIELDS

<table>
<thead>
<tr>
<th>#AND</th>
<th>#XOR</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x) = x^{2^{33}} + x^{2^{31}} + x^{2^{19}} + 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[13]</td>
<td>$54289$</td>
<td>$54289 + 14T_Y$</td>
</tr>
<tr>
<td>[12]</td>
<td>$54289$</td>
<td>$54289 + 14T_Y$</td>
</tr>
<tr>
<td>[18]</td>
<td>$54289$</td>
<td>$54289 + 14T_Y$</td>
</tr>
<tr>
<td>[6]</td>
<td>$54289$</td>
<td>$54289 + 14T_Y$</td>
</tr>
<tr>
<td>[9]</td>
<td>$54289$</td>
<td>$54289 + 14T_Y$</td>
</tr>
<tr>
<td>$f(x) = x^{2^{14}} + x^{2^{7}} + x^{2^{5}} + 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[13]</td>
<td>$80089$</td>
<td>$80089 + 14T_Y$</td>
</tr>
<tr>
<td>[12]</td>
<td>$80089$</td>
<td>$80089 + 14T_Y$</td>
</tr>
<tr>
<td>[18]</td>
<td>$80089$</td>
<td>$80089 + 14T_Y$</td>
</tr>
<tr>
<td>[6]</td>
<td>$80089$</td>
<td>$80089 + 14T_Y$</td>
</tr>
<tr>
<td>[9]</td>
<td>$80089$</td>
<td>$80089 + 14T_Y$</td>
</tr>
<tr>
<td>$f(x) = x^{2^{67}} + x^{2^{65}} + x^{2^{19}} + 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[13]</td>
<td>$167281$</td>
<td>$167281 + 14T_Y$</td>
</tr>
<tr>
<td>[12]</td>
<td>$167281$</td>
<td>$167281 + 14T_Y$</td>
</tr>
<tr>
<td>[18]</td>
<td>$167281$</td>
<td>$167281 + 14T_Y$</td>
</tr>
<tr>
<td>[6]</td>
<td>$167281$</td>
<td>$167281 + 14T_Y$</td>
</tr>
<tr>
<td>[9]</td>
<td>$167281$</td>
<td>$167281 + 14T_Y$</td>
</tr>
</tbody>
</table>

TABLE VII
COMPUTED VALUES FOR $Y_m$ AND $\Sigma_m$, $m = 8 \ldots 100$

<table>
<thead>
<tr>
<th>$m$</th>
<th>$Y_m$</th>
<th>$\Sigma_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>17</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>21</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>22</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>23</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>24</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>25</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>26</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>27</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>28</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>29</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>31</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>37</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>38</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

V. CONCLUSIONS

High-speed algorithms and hardware architectures for computing $GF(2^m)$ multiplication are highly required in several applications, such as coding theory, computer algebra and cryptography. In this paper, a new bit-parallel $GF(2^m)$ polynomial basis multiplier for type II irreducible pentanomials with reduced time-complexity has been presented. The coefficients of the multiplier are computed as a sum of $S_1$ and $T_1$ functions given by the addition of product terms of the coefficients of the two operands to be multiplied. In the new approach here proposed, the sum of products in the $S_1$ and $T_1$ functions are separated into sums of product terms (corresponding to the initial $S_1$ and $T_1$ terms) that can be implemented as binary trees of XOR gates with depth $j$. The sum in pairs of binary trees with the same depth, starting with the lower levels, leads to a reduction of the time complexity of the multiplier. In this paper, a complete multiplication example has been presented. The theoretical complexity analysis has shown that the proposed bit-parallel multiplier presents the lowest delay among the best
results known to date for similar polynomial basis multipliers based on irreducible pentanomials. Simulations have been done that have proved that for the 593 different values of $m$ in the interval $m \in [8,1000]$ for which an irreducible type II pentanomial exists, the proposed multiplier has the smallest delay in 465 different values of the field size $m$. Furthermore, for the five binary fields recommended by NIST for ECDSA, i.e., $m \in \{163,233,283,409,571\}$, the multiplier here proposed presents the lowest delay except for $GF(2^{233})$, that matches the best delay given in the literature.

**APPENDIX A**

**TIME COMPLEXITY**

A. Total Number of Terms in $\log_{2^m}$-level

Let $\mu_0, \mu_1, \ldots, \mu_{\log_{2^m}}$ denote the number of initial $S_i$ and $T_i$ terms in levels 0, 1, $\ldots$, $\log_{2^m}$, respectively. As previously stated, for a given level $i$, the number of new XOR terms that will result in level $i+1$ due to the addition in pairs of the $i$-level terms is given by $\lceil \text{number of } i \text{-level terms}/2 \rceil$. Starting in level 0, then the new terms created in level 1 due to the sum in pairs of the initial terms in level 0, $\mu_0$, will be $\lceil \mu_0/2 \rceil$. The total number of terms in level 1, denoted by $M_1$, will now be $M_1 = \mu_1 + \lceil \mu_0/2 \rceil$. Using the property of modulo operation $[x] = [x + n]$ for $n$ integer, then we have that $M_1 = \mu_1 + \mu_0/2 - \lfloor \mu_0/2 \rfloor - \mu_1/2$. Next the new terms created in level 2 due to the sum in pairs of the terms in level 1, $M_1$, will be $\lceil M_1/2 \rceil$. Using the property of modulo operation $[x/m]/n = [x/(m \cdot n)]$ for positive integers $m, n$ and arbitrary real number $x$, then the new XOR terms created in level 2 will be $M_2 = \lceil (\mu_0 + 2\mu_1)/2 \rceil = \lceil \mu_0/2 \rceil + \lceil \mu_1/2 \rceil$. The total number of terms in level 2, denoted by $M_2$, will now be $M_2 = \mu_2 + \mu_0/2 + \mu_1/2$. Proceeding in the same way we will have that the new XOR terms created in level $\log_{2^m}$ due to the sum in pairs of the terms in level $\log_{2^m}-1$, $M_{\log_{2^m}-1}$, will be $\lceil \mu_2 + \mu_1/2 + \mu_0/2 \rceil$. Finally, the total number $M_{\log_{2^m}}$ of terms in the $\log_{2^m}$-level will be the sum of $\mu_{\log_{2^m}}$ plus the expression in (5), that is:

$$M_{\log_{2^m}} = \mu_0/2 + \mu_1 + \mu_2 + \ldots + \mu_{\log_{2^m}} + \lceil \mu_0/2 \rceil = \frac{(n-1) + (m-n-3) - 2\Gamma_2}{2\log_{2^m}} - (4m + 3n - 9) - 2\Gamma_2.$$

Now (4) can be used to simplify (13). In (4), the number of initial $S_i$ and $T_i$ terms in level $j$ is given, where $[x/2^j]$ represents $[x/2^j] \bmod 2$. The modulo operator is defined by the expression $x \bmod y = x - y \lfloor x/y \rfloor$, for real $x, y \neq 0$. Therefore $[x/2^j]\bmod 2 = [x/2^2] - \lfloor x/2^2 \rfloor = [x/2^{2j} - 2 \cdot [x/2^{2j}]].$ Then $\mu_0$ in (4) can be rewritten as $\mu_0 = (m-1)/2^j + (m-3)/2^j + \ldots + (m-n)/2^j$, $\mu_1 = (m-1)/2^j + (m-3)/2^j + \ldots + (m-n)/2^j$, $\mu_2 = (m-1)/2^j + (m-3)/2^j + \ldots + (m-n)/2^j$, $\mu_{\log_{2^m}} = (m-1)/2^j + (m-3)/2^j + \ldots + (m-n)/2^j$, where we have denoted the sum of the first eight terms as $\Gamma_2$, and the sum of the eight terms into the parenthesis as $\Gamma_{2^j}$. For even $m$, we have $\mu_0 = \Gamma_2 - 2\Gamma_{2^j} = (m-1) + (m-2) + (m-3) + (m-4) + (m-5) + (m-6) + (m-7) + (m-8)$

### B. Upper Bound for $T_X$ Delay

The delay of the multiplier given in (7) is the following:

$$T_A + \left( \left\lceil \log_{2^{m+1}} \frac{4m + 3n - 9}{2\log_{2^m}} \right\rceil \right) T_X$$

For type II irreducible pentanomials, $n \leq \lfloor m/2 \rfloor - 1$, so for even $m$ we have that $n \leq \lfloor m/2 \rfloor - 1$ while that for odd $m$ we have $n \leq (\lfloor m/2 \rfloor - 1)/2 - 1$. The following operations can be done:

- **Even $m$.** We have $4m + 3n - 9 \leq 4m + 3\lfloor (m/2) - 1 \rfloor - 9 = (4m + 3m/2) - 12 = 11m/2 - 12 < 11m/2$. Substituting this expression in the quotient in $M_{\log_{2^m}}$, and using the fact that $2\log_{2^m+1} > m$, then we have $\left(4m + 3n - 9\right)/2\log_{2^m} \leq 11m/2 - 12$. Therefore we have that $\left(4m + 3n - 9\right)/2\log_{2^m} \leq 11m/2 - 12$. Finally we will have that $\left\lceil \log_{2^m} \frac{4m + 3n - 9}{2\log_{2^m}} \right\rceil \leq \log_{2^m} \left\lceil \frac{11m}{2} \right\rceil = 4m + 3n - 9. \quad (20)$

**Odd $m$.** We have $4m + 3n - 9 \leq 4m + 3\left(\left\lfloor (m/2) - 1 \right\rfloor - 1 \right) - 9 = 11m/2 - 27/2 < 11m/2$ and then we get the same results as in the previous case for even $m$.

Using the result given in (20), then we have $\left\lceil \log_{2^m} \right\rceil + \left\lceil \log_{2^m} \left(4m + 3n - 9\right)/2\log_{2^m} \right\rceil \leq \log_{2^m} \left(4m + 3n - 9\right)/2\log_{2^m} + 4$ and using the property $1 + \log_{2^m} = \left\lfloor \log_{2^m}(m+1) \right\rfloor$ we have finally

$$\mu_0 + \mu_1 + \mu_2 + \ldots + \mu_{\log_{2^m}} = (4m + 3n - 9) - 2\Gamma_2 + \Gamma_{2^j} + \mu_{\log_{2^m}} = (4m + 3n - 9) - 2\Gamma_2 + \Gamma_{2^j} + \mu_{\log_{2^m}} + 1$$

Using (14)–(16), the numerator of (13) can be simplified as follows:

$$\mu_0 + \mu_1 + \mu_2 + \ldots + \mu_{\log_{2^m}} = (4m + 3n - 9) - 2\Gamma_2 + \Gamma_{2^j} + \mu_{\log_{2^m}} + 1$$

The term $\Gamma_{2^{\log_{2^m}+1}}$ in (17) can be computed using the definition previously given. According to that definition, it can be observed that $\Gamma_{2^{\log_{2^m}+1}}$ is the sum of the eight floor functions of eight quotients with the same denominator $2\log_{2^m}+1$ and where the numerators are integers smaller than $m$. However, the value of $2\log_{2^m}+1$ is always greater than $m$, so the quotients are less than unity and all the floor functions are always zero. Therefore, the term $\Gamma_{2^{\log_{2^m}+1}}$ is 0 and (17) will be $\mu_0 + \mu_1 + \mu_2 + \ldots + \mu_{\log_{2^m}} = (4m + 3n - 9)$. Using this result and applying it to (13), it follows (18), that matches (6).

$$M_{\log_{2^m}} = \left\lceil \log_{2^{m+1}} \frac{4m + 3n - 9}{2\log_{2^m}} \right\rceil \quad (18)$$
that the XOR delay of the multiplier can be upper bounded as follows, matching (8):

\[ |\log_2 m| + \left\lfloor \log_2 \left( \frac{4m + 3n - 9}{2^{1+2|\log_2 m|}} \right) \right\rfloor \leq |\log_2 (m + 1)| + 3 \]

(21)

**APPENDIX B**

**AREA COMPLEXITY**

The XOR gates of the multiplier will be \( \sum_{i=1}^h \Psi_i \) if they appear \( h \) times. This is determined as follows:

1. The number of AND gates needed for the sum of the terms \( \Psi_i \) for \( i \leq h \) can be computed as

\[ \sum_{i=1}^h \Psi_i \]

2. The number of XOR gates needed for the sum of the terms \( \Psi_i \) for \( i > h \) can be found that for \( i \geq h \), the term with lowest subindex gives the number of XOR gates to be shared.

\[ \Psi_i \sim \left( i - \sum_{j=1}^{\log_2 i} \left\lfloor \frac{i}{2^j} \right\rfloor \right) \sim H_i - 1, \]

(22)

that matches (9). Using (22), then \( \Psi_i \) can be computed:

\[ \Psi_i \sim \left( i - \sum_{j=1}^{\log_2 i} \left\lfloor \frac{i}{2^j} \right\rfloor \right) \sim H_i - 1, \]

(23)

**that matches (10).**

4. The number of XOR gates needed for the sum of the terms \( \Psi_i \) for \( i \geq h \) can be found that for \( i \geq h \), the term with lowest subindex gives the number of XOR gates to be shared.

\[ \Psi_i \sim \left( i - \sum_{j=1}^{\log_2 i} \left\lfloor \frac{i}{2^j} \right\rfloor \right) \sim H_i - 1, \]

(24)

that matches (10).
\[
\sum_{t=2,4,6,\ldots}^{\infty} H_t + H_{n-t}/(n-1)^t = \Sigma m + H_{\ell}^t \tag{24}
\]
that matches (11). In (24), \(\ell\) represents the limit \((m - 2)\) of the summatory for even \(m\), \(\kappa\) represents the limit \((m - 3)\); for odd \(m\), \(\ell\) represents the Hamming Weight of \(n\) to be computed for even \(n\) and \(\kappa\) the Hamming Weight of \((n-1)\) for odd \(n\).

REFERENCES


José L. Imaña received the M.Sc. and Ph.D. degrees in physics from Complutense University, Madrid, Spain, in 1989 and 2003, respectively. He was an Electronic Design Engineer at the Madrid Institute of Technology, Spain. He is currently with the Department of Computer Architecture and Systems Engineering at Complutense University, where he was promoted to an Associate Professor with tenure in 2006. He has been the promoter and cofounder of the International Workshop on the Arithmetic of Finite Fields (WAIFI). His research interests include algorithms and VLSI architectures for computations in finite fields, cryptography, computer arithmetic, and reconfigurable computing.