Reconfigurable implementation of $GF(2^m)$ bit-parallel multipliers

José L. Imaña
Department of Computer Architecture and Automation
Faculty of Physics, Complutense University
28040 Madrid, Spain
Email: jluimana@ucm.es

***Abstract*—Hardware implementations of arithmetic operations over binary finite fields $GF(2^m)$ are widely used in several important applications, such as cryptography, digital signal processing and error-control codes. In this paper, efficient reconfigurable implementations of bit-parallel canonical basis multipliers over binary fields generated by type II irreducible pentanomials $f(y) = y^m + y^{n+2} + y^{n+1} + y^n + 1$ are presented. These pentanomials are important because all five binary fields recommended by NIST for ECDSA can be constructed using such polynomials. In this work, a new approach for $GF(2^m)$ multiplication based on type II pentanomials is given and several post-place and route implementation results in Xilinx Artix-7 FPGA are reported. Experimental results show that the proposed multiplier implementations improve the $area{\times}time$ parameter when compared with similar multipliers found in the literature.**

I. Introduction

Galois or finite fields $GF(2^m)$ have been widely studied due to their use in important applications, such as cryptography, digital signal processing and error control codes. These applications frequently require efficient hardware implementations of $GF(2^m)$ arithmetic operations, of which *multiplication* is the most complex and important one. Efficient multiplication methods and architectures have been proposed for different representation bases, where *canonical* or *polynomial* basis is the most widely used. Apart from the basis selection, the complexity of the multiplication also depends on the defining irreducible polynomial $f(y)$ selected for the field. For $GF(2^m)$ hardware implementation, irreducible trinomials and pentanomials are normally used.

Two-step classic polynomial basis multiplication in $GF(2^m)$ involves a polynomial multiplication followed by a reduction modulo an irreducible polynomial. An efficient bit-parallel canonical multiplier was proposed by Mastrovito [1] in which a *product matrix* combines the above two steps together [2],[3],[4],[5]. A new polynomial basis multiplication scheme was proposed in [6], where the decomposition of a product matrix led to the introduction of $\mathbf{S_i}$ and $\mathbf{T_i}$ functions given by the sum of product terms. The addition of these functions is used for the computation of the product of two $GF(2^m)$ elements. The sum of products in $\mathbf{S_i}$ and $\mathbf{T_i}$ were split in [7] into sums of 2^j product terms implemented as binary trees of XOR gates with depth j. The addition in pairs of binary trees with the same depth leads to a reduction of the multiplication delay.

In this paper, efficient Xilinx FPGA implementations of $GF(2^m)$ bit-parallel canonical basis multipliers based on type II pentanomials $f(y) = y^m + y^{n+2} + y^{n+1} + y^n + 1$ are presented. In order to optimize the synthesis, a new approach for the product is considered in which the splitting of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms is performed, but where the restriction imposed by the addition in pairs of binary trees with the same depth has been removed. In such a case, Xilinx XST tool has freedom to optimize the synthesis of the multiplier. Several $GF(2^m)$ multipliers, including the specific field $GF(2^8)$, have been described in VHDL and post-place and route implementation results in Xilinx Artix-7 have been reported. The field $GF(2^8)$ is specially important because it has been standardized for space communication by NASA and ESA and used in CD players and Advanced Encryption Standard (AES). Experimental results show that the proposed approach for multiplication improves the $area{\times}time$ complexity when compared with similar multiplication methods found in the literature. Furthermore, the new approach also presents the lowest delay for most of the here implemented multipliers.

II. Background

Any element A of the binary field $GF(2^m)$ can be represented in the canonical or polynomial basis $\{1, x, \ldots, x^{m-1}\}$ as $A = \sum_{i=0}^{m-1} a_i x^i$, with $a_i \in GF(2)$, where x is a root of the irreducible polynomial $f(y) = \sum_{i=0}^{m} f_i y^i$. Canonical basis multiplication in $GF(2^m)$ involves a polynomial multiplication followed by a reduction modulo the irreducible polynomial. An efficient bit-parallel canonical multiplication method was proposed by Mastrovito [1] in which a *product matrix* combines the above two steps together. In [6], a new $GF(2^m)$ polynomial basis multiplication approach was used. In order to compute the product $C = A \cdot B$, this method introduced the functions $\mathbf{S_i}$ and $\mathbf{T_i}$ given by the addition of terms $x_k = (a_k b_k)$ and $z_i^j = (a_i b_j + a_j b_i)$, with $a_i, b_i \in GF(2)$ being the coordinates of $A, B \in GF(2^m)$, respectively. These functions are implemented as binary trees of 2-input XOR gates with a lower level of 2-input AND gates (corresponding to the $a_i b_j$ products). The expressions for $\mathbf{S_i}$ $(1 \leq i \leq m)$ and $\mathbf{T_i}$ $(0 \leq i \leq m-2)$ are [6]:

$$\mathbf{S_i} = x_p + \sum_{h=0}^{p-1} z_h^{i-h-1}, \qquad \mathbf{T_i} = x_q + \sum_{j=1}^{r-(i+1)} z_{i+j}^{m-j} \qquad (1)$$

where $p = \lfloor i/2 \rfloor$ and $q = (\lceil m/2 \rceil + \lfloor i/2 \rfloor)$. In (1), the term $x_p = a_p b_p$ only appears for i *odd* and x_q only appears for (m and i *even*) or for (m and i *odd*). In this case, $r = q$. Otherwise, i.e., for (m *even* and i *odd*) or for (m *odd* and i *even*), the term x_q does not appear and $r = (\lceil m/2 \rceil + \lceil i/2 \rceil)$.

For example, using (1), the terms $\mathbf{S_i}$ and $\mathbf{T_i}$ for $GF(2^8)$ are: $\mathbf{S_1} = x_0 = a_0b_0$, $\mathbf{S_2} = z_0^1 = (a_0b_1 + a_1b_0)$, $\mathbf{S_3} = x_1 + z_0^2 = a_1b_1 + (a_0b_2 + a_2b_0)$, $\mathbf{S_4} = z_0^3 + z_1^2 = (a_0b_3 + a_3b_0) + (a_1b_2 + a_2b_1)$, $\mathbf{S_5} = x_2 + z_0^4 + z_1^3 = a_2b_2 + (a_0b_4 + a_4b_0) + (a_1b_3 + a_3b_1)$, $\mathbf{S_6} = z_0^5 + z_1^4 + z_2^3 = (a_0b_5 + a_5b_0) + (a_1b_4 + a_4b_1) + (a_2b_3 + a_3b_2)$, $\mathbf{S_7} = x_3 + z_0^6 + z_1^5 + z_2^4 = a_3b_3 + (a_0b_6 + a_6b_0) + (a_1b_5 + a_5b_1) + (a_2b_4 + a_4b_2)$, $\mathbf{S_8} = z_0^7 + z_1^6 + z_2^5 + z_3^4 = (a_0b_7 + a_7b_0) + (a_1b_6 + a_6b_1) + (a_2b_5 + a_5b_2) + (a_3b_4 + a_4b_3)$, and $\mathbf{T_0} = x_4 + z_1^7 + z_2^6 + z_3^5 = a_4b_4 + (a_1b_7 + a_7b_1) + (a_2b_6 + a_6b_2) + (a_3b_5 + a_5b_3)$, $\mathbf{T_1} = z_2^7 + z_3^6 + z_4^5 = (a_2b_7 + a_7b_2) + (a_3b_6 + a_6b_3) + (a_4b_5 + a_5b_4)$, $\mathbf{T_2} = x_5 + z_3^7 + z_4^6 = a_5b_5 + (a_3b_7 + a_7b_3) + (a_4b_6 + a_6b_4)$, $\mathbf{T_3} = z_4^7 + z_5^6 = (a_4b_7 + a_7b_4) + (a_5b_6 + a_6b_5)$, $\mathbf{T_4} = x_6 + z_5^7 = a_6b_6 + (a_5b_7 + a_7b_5)$, $\mathbf{T_5} = z_6^7 = (a_6b_7 + a_7b_6)$, $\mathbf{T_6} = x_7 = a_7b_7$. The product $C = A \cdot B$ can then be computed as the addition of these terms.

For hardware implementation of $GF(2^m)$ multiplication, low Hamming weight irreducible polynomials, such as trinomials and pentanomials, are normally used. *Type II irreducible pentanomials* [5] $f(y) = y^m + y^{n+2} + y^{n+1} + y^n + 1$, with $2 \le n \le \lfloor m/2 \rfloor - 1$, are important because they are abundant and all five binary fields recommended by NIST for ECDSA can be constructed using such irreducible polynomials. Canonical basis multiplication for these pentanomials was studied in [6], where expressions for the coefficients of the product were given using $\mathbf{S_i}$ and $\mathbf{T_i}$ terms. For the specific field $GF(2^8)$ generated by the polynomial $f(y) = y^8 + y^4 + y^3 + y^2 + 1$, with $(m, n) = (8, 2)$, the coefficients computed as in [6] are given in Table I.

TABLE I
COEFFICIENTS OF THE PRODUCT FOR $GF(2^8)$ WITH $(m, n) = (8, 2)$.

c_0	=	$\mathbf{S_1} + \mathbf{T_0} + \mathbf{T_4} + \mathbf{T_5} + \mathbf{T_6}$;
c_1	=	$\mathbf{S_2} + \mathbf{T_1} + \mathbf{T_5} + \mathbf{T_6}$;
c_2	=	$\mathbf{S_3} + \mathbf{T_0} + \mathbf{T_2} + \mathbf{T_4} + \mathbf{T_5}$;
c_3	=	$\mathbf{S_4} + \mathbf{T_0} + \mathbf{T_1} + \mathbf{T_3} + \mathbf{T_4}$;
c_4	=	$\mathbf{S_5} + \mathbf{T_0} + \mathbf{T_1} + \mathbf{T_2} + \mathbf{T_6}$;
c_5	=	$\mathbf{S_6} + \mathbf{T_1} + \mathbf{T_2} + \mathbf{T_3}$;
c_6	=	$\mathbf{S_7} + \mathbf{T_2} + \mathbf{T_3} + \mathbf{T_4}$;
c_7	=	$\mathbf{S_8} + \mathbf{T_3} + \mathbf{T_4} + \mathbf{T_5}$;

One of the problems to reduce the delay of the product is due to the monolithic construction of $\mathbf{S_i}$ and $\mathbf{T_i}$ functions, given by a sum of terms x_k and z_i^j. For example, for $GF(2^8)$ the addition of $\mathbf{S_1} + \mathbf{T_4} = a_0b_0 + (a_6b_6 + (a_5b_7 + a_7b_5))$ would result in a 3-level binary tree of XOR gates. However, the sum of $\mathbf{S_1} + \mathbf{T_4}$ involves the addition of four product terms a_0b_0, a_6b_6, a_5b_7 and a_7b_5, so it could be done with a 2-level complete binary tree of XOR gates if the product a_0b_0 could be first added with a_6b_6 and then perform the addition with $(a_5b_7 + a_7b_5)$, in such a way that $\mathbf{S_1} + \mathbf{T_4} = (a_0b_0 + a_6b_6) + (a_5b_7 + a_7b_5)$. This idea was used in [7] for $GF(2^m)$ polynomial multiplication based on type II irreducible pentanomials. Functions $\mathbf{S_i}$ and $\mathbf{T_i}$ were split in the form $\mathbf{S_i} = s_\rho^i \mathbf{S_i^\rho} + \ldots + s_1^i \mathbf{S_i^1} + s_0^i \mathbf{S_i^0}$ and $\mathbf{T_i} = t_\rho^i \mathbf{T_i^\rho} + \ldots + t_1^i \mathbf{T_i^1} + t_0^i \mathbf{T_i^0}$, with $s_j^i, t_j^i \in GF(2)$ and $\rho = \lfloor log_2 m \rfloor$. The terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ represent the addition of 2^j products a_kb_l and therefore can be implemented as a j-level complete binary tree of XOR gates. The sum of any of these j-level terms results in a new XOR in the level $j+1$ representing a $(j+1)$-level complete binary tree of XOR gates. If the sum of $\mathbf{S_i}$ and $\mathbf{T_i}$ functions in the coordinates of the product is done by grouping the additions of $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms with the same level, starting with the lower ones, then the number of XOR levels needed to compute the product of two field elements can be reduced.

For $GF(2^8)$, the expressions for the $\mathbf{S_i}$ and $\mathbf{T_i}$ functions given as the addition of $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms are as follows [7]: $\mathbf{S_1} = 0 \cdot \mathbf{S_1^3} + 0 \cdot \mathbf{S_1^2} + 0 \cdot \mathbf{S_1^1} + 1 \cdot \mathbf{S_1^0} = \mathbf{S_1^0}$, $\mathbf{S_2} = 0 \cdot \mathbf{S_2^3} + 0 \cdot \mathbf{S_2^2} + 1 \cdot \mathbf{S_2^1} + 0 \cdot \mathbf{S_2^0} = \mathbf{S_2^1}$, $\mathbf{S_3} = \mathbf{S_3^1} + \mathbf{S_3^0}$, $\mathbf{S_4} = \mathbf{S_4^2}$, $\mathbf{S_5} = \mathbf{S_5^2} + \mathbf{S_5^0}$, $\mathbf{S_6} = \mathbf{S_6^2} + \mathbf{S_6^1}$, $\mathbf{S_7} = \mathbf{S_7^2} + \mathbf{S_7^1} + \mathbf{S_7^0}$, $\mathbf{S_8} = \mathbf{S_8^3}$, and $\mathbf{T_0} = \mathbf{T_0^2} + \mathbf{T_0^1} + \mathbf{T_0^0}$, $\mathbf{T_1} = \mathbf{T_1^2} + \mathbf{T_1^1}$, $\mathbf{T_2} = \mathbf{T_2^2} + \mathbf{T_2^0}$, $\mathbf{T_3} = \mathbf{T_3^2}$, $\mathbf{T_4} = \mathbf{T_4^1} + \mathbf{T_4^0}$, $\mathbf{T_5} = \mathbf{T_5^1}$, $\mathbf{T_6} = \mathbf{T_6^0}$. The expressions of the corresponding $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms are given in Table II.

TABLE II
TERMS $\mathbf{S_i^j}$ AND $\mathbf{T_i^j}$ FOR $GF(2^8)$.

$\mathbf{S_1^0}$	=	x_0	$\mathbf{T_0^0}$	=	x_4
$\mathbf{S_2^1}$	=	z_0^1	$\mathbf{T_0^1}$	=	z_1^7
$\mathbf{S_3^0}$	=	x_1	$\mathbf{T_0^2}$	=	$(z_2^6 + z_3^5)$
$\mathbf{S_3^1}$	=	z_0^2	$\mathbf{T_1^1}$	=	z_2^7
$\mathbf{S_4^2}$	=	$(z_0^3 + z_1^2)$	$\mathbf{T_1^2}$	=	$(z_3^6 + z_4^5)$
$\mathbf{S_5^0}$	=	x_2	$\mathbf{T_2^0}$	=	x_5
$\mathbf{S_5^2}$	=	$(z_0^4 + z_1^3)$	$\mathbf{T_2^2}$	=	$(z_3^7 + z_4^6)$
$\mathbf{S_6^1}$	=	z_0^5	$\mathbf{T_3^2}$	=	$(z_4^7 + z_5^6)$
$\mathbf{S_6^2}$	=	$(z_1^4 + z_2^3)$	$\mathbf{T_4^0}$	=	x_6
$\mathbf{S_7^0}$	=	x_3	$\mathbf{T_4^1}$	=	z_5^7
$\mathbf{S_7^1}$	=	z_0^6	$\mathbf{T_5^1}$	=	z_6^7
$\mathbf{S_7^2}$	=	$(z_1^5 + z_2^4)$	$\mathbf{T_6^0}$	=	x_7
$\mathbf{S_8^3}$	=	$(z_0^7 + z_1^6 + z_2^5 + z_3^4)$			

From Table II, it can be observed that the terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ perform the addition of 2^j products so they can be implemented as j-level complete binary trees of XOR gates. Using these terms, expressions for the $GF(2^8)$ multiplier based on the splitting method introduced in [7] for type II irreducible pentanomial are given in Table III, where the notations $\mathbf{T_{i,j}^{k+1}} = \mathbf{T_i^k} + \mathbf{T_j^k}$ and $\mathbf{ST_{i,j}^{k+1}} = \mathbf{S_i^k} + \mathbf{T_j^k}$ have been used. Terms in parenthesis indicate that they must be XORed previously to the XOR with the other terms in order to reduce the delay. Furthermore, terms that appear in more than one coefficient could be shared, therefore reducing the space requirements (for example, the term $\mathbf{T_{0,4}^1} = \mathbf{T_0^0} + \mathbf{T_4^0}$ in Table III, that appears in the coefficients c_0 and c_2).

Theoretical complexities of bit-parallel multipliers based on type II pentanomials were given in [7]. For the $GF(2^8)$ multiplier shown in table III, it can be found that the delay complexity is $T_\text{A} + 5T_\text{X}$, with T_A and T_X representing the

TABLE III
COEFFICIENTS OF THE PRODUCT FOR $GF(2^8)$ WITH SPLITTING.

$$
\begin{aligned}
c_0 &= ((\mathbf{S}^{\mathbf{0}}_{\mathbf{1}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{0,4}}) + \mathbf{T}^{\mathbf{2}}_{\mathbf{0}}) + (\mathbf{T}^{\mathbf{2}}_{\mathbf{0,4}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{5,6}});\\
c_1 &= (\mathbf{ST}^{\mathbf{2}}_{\mathbf{2,1}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{1}}) + \mathbf{T}^{\mathbf{2}}_{\mathbf{5,6}};\\
c_2 &= ((\mathbf{ST}^{\mathbf{1}}_{\mathbf{3,2}} + \mathbf{S}^{\mathbf{1}}_{\mathbf{3}}) + \mathbf{T}^{\mathbf{2}}_{\mathbf{0}}) + ((\mathbf{T}^{\mathbf{1}}_{\mathbf{0,4}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{5}}) + (\mathbf{T}^{\mathbf{2}}_{\mathbf{0,4}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{2}}));\\
c_3 &= ((\mathbf{T}^{\mathbf{2}}_{\mathbf{0,1}} + \mathbf{S}^{\mathbf{2}}_{\mathbf{4}}) + \mathbf{T}^{\mathbf{3}}_{\mathbf{0,1}}) + ((\mathbf{T}^{\mathbf{1}}_{\mathbf{0,4}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{4}}) + \mathbf{T}^{\mathbf{2}}_{\mathbf{3}});\\
c_4 &= (((\mathbf{ST}^{\mathbf{1}}_{\mathbf{5,0}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{2,6}}) + \mathbf{S}^{\mathbf{2}}_{\mathbf{5}}) + \mathbf{T}^{\mathbf{3}}_{\mathbf{0,1}}) + (\mathbf{T}^{\mathbf{2}}_{\mathbf{0,1}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{2}});\\
c_5 &= \mathbf{ST}^{\mathbf{3}}_{\mathbf{6,1}} + ((\mathbf{ST}^{\mathbf{2}}_{\mathbf{6,1}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{2}}) + \mathbf{T}^{\mathbf{3}}_{\mathbf{2,3}});\\
c_6 &= ((\mathbf{ST}^{\mathbf{1}}_{\mathbf{7,2}} + \mathbf{S}^{\mathbf{1}}_{\mathbf{7}}) + \mathbf{S}^{\mathbf{2}}_{\mathbf{7}}) + (\mathbf{T}^{\mathbf{3}}_{\mathbf{2,3}} + (\mathbf{T}^{\mathbf{0}}_{\mathbf{4}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{4}}));\\
c_7 &= \mathbf{S}^{\mathbf{3}}_{\mathbf{8}} + (\mathbf{T}^{\mathbf{2}}_{\mathbf{3}} + (\mathbf{T}^{\mathbf{2}}_{\mathbf{4,5}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{4}}));
\end{aligned}
$$

delay of 2-input AND and XOR gates, respectively. This theoretical delay is the lowest one among similar $GF(2^8)$ multipliers, such as those given in [6] and [3], with delays $T_{\mathrm{A}} + 6T_{\mathrm{X}}$ and $T_{\mathrm{A}} + 7T_{\mathrm{X}}$, respectively. The space complexity (number of 2-input AND and XOR gates) of the multiplier given in table III was found to be 64 AND and 87 XOR gates. In this case, the theoretical number of XOR gates is greater than those found in [6] and [3], with 80 and 77 XOR gates, respectively, while that the number of 2-input AND gates is the same in all approaches.

III. FPGA EFFICIENT $GF(2^8)$ POLYNOMIAL BASIS MULTIPLIER

Expressions given in Table III for the coefficients of the $GF(2^8)$ polynomial basis multiplier impose hard restrictions (given by the parenthesis) for the addition of the different terms in order to reduce the number of XOR levels and therefore reduce the delay of the multiplier. However, these restrictions could not be efficient for a synthesis tool in order to map that expressions into FPGA's logic blocks. In such a case, more freedom should be given to the synthesizer to find an optimized implementation of the multiplier.

TABLE IV
NEW COEFFICIENTS OF THE PRODUCT FOR TYPE II $GF(2^8)$.

$$
\begin{aligned}
c_0 &= \mathbf{S}^{\mathbf{0}}_{\mathbf{1}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{4}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{4}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{5}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{6}};\\
c_1 &= \mathbf{S}^{\mathbf{1}}_{\mathbf{2}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{1}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{1}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{5}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{6}};\\
c_2 &= \mathbf{S}^{\mathbf{1}}_{\mathbf{3}} + \mathbf{S}^{\mathbf{0}}_{\mathbf{3}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{2}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{2}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{4}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{4}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{5}};\\
c_3 &= \mathbf{S}^{\mathbf{2}}_{\mathbf{4}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{1}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{1}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{3}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{4}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{4}};\\
c_4 &= \mathbf{S}^{\mathbf{2}}_{\mathbf{5}} + \mathbf{S}^{\mathbf{0}}_{\mathbf{5}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{0}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{1}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{1}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{2}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{2}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{6}};\\
c_5 &= \mathbf{S}^{\mathbf{2}}_{\mathbf{6}} + \mathbf{S}^{\mathbf{1}}_{\mathbf{6}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{1}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{1}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{2}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{2}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{3}};\\
c_6 &= \mathbf{S}^{\mathbf{2}}_{\mathbf{7}} + \mathbf{S}^{\mathbf{1}}_{\mathbf{7}} + \mathbf{S}^{\mathbf{0}}_{\mathbf{7}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{2}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{2}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{3}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{4}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{4}};\\
c_7 &= \mathbf{S}^{\mathbf{3}}_{\mathbf{8}} + \mathbf{T}^{\mathbf{2}}_{\mathbf{3}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{4}} + \mathbf{T}^{\mathbf{0}}_{\mathbf{4}} + \mathbf{T}^{\mathbf{1}}_{\mathbf{5}};
\end{aligned}
$$

In Table IV, new expressions for the coefficients of the $GF(2^8)$ polynomial basis multiplier are given. The splitting of the $\mathbf{S_i}$ and $\mathbf{T_i}$ functions as the addition of $\mathbf{S^j_i}$ and $\mathbf{T^j_i}$ terms (given in Table II) has been used, but the restriction imposed in the product by the parenthesized addition of terms with the same j-level has been removed. The coefficients of the product are then given as sums of $\mathbf{S^j_i}$ and $\mathbf{T^j_i}$ individual terms and the synthesis tool is free to perform an optimized implementation of the multiplier.

IV. FPGA IMPLEMENTATION RESULTS

The $GF(2^8)$ polynomial basis multiplier given in Table IV has been implemented in Xilinx Artix-7 XC7A200T-FFG1156. The design entry has been behavioral VHDL and the experimental results are those reported by Xilinx ISE 14.7 using XST synthesizer. Furthermore, same pin assignments and *speed high* optimizations have been part of the design methodology. In order to compare the proposed $GF(2^8)$ multiplier with other similar approaches, VHDL descriptions of different multipliers have also been implemented. The methods used for description and comparison have been the Mastrovito approaches given in [2] and [3], the bit-parallel version of the multiplier presented in [8], the multiplier given in [6] that introduced the $\mathbf{S_i}$ and $\mathbf{T_i}$ functions, and the method with splitting $\mathbf{S_i}/\mathbf{T_i}$ functions and hard parenthesized restrictions presented in [7].

Experimental post-place and route results obtained for $GF(2^8)$ multipliers are given in Table V, where the area complexity is expressed in terms of the number of LUTs and Slices used. Time results (in nanoseconds) represent the critical path of the $GF(2^m)$ multipliers. The A$\times$T metrics express time delay by area in $LUTs \times ns$ in order to compare the area and delay (less is better). From Table V, it can be observed that the proposed multiplier exhibits the lowest number of LUTs used and the lowest A$\times$T metrics among the different approaches, while the lowest number of slices and delay correspond to the works given in [2] and [8], respectively. In comparison with the splitting method with parenthesized restrictions given in Table III, it can be observed that the new approach is more area and time efficient.

The new approach used for the $GF(2^8)$ multiplier has been applied to the implementation of several type II irreducible polynomial basis multipliers. Same design methodology and methods used for comparison in $GF(2^8)$ have been considered. The post-place and route results are also given in Table V for type II multipliers with values (m, n) = (64,23), (113,4), (113,34), (122,49), (139,59), (148,72), (163,66) and (163,68), where binary fields $GF(2^{113})$ are recommended by SECG (Standards for Efficient Cryptography Group) [9] and $GF(2^{163})$ are recommended by NIST for ECDSA. From the experimental results, it can be observed that the new approach here proposed exhibits the best $area \times time$ values among the different methods implemented for most of the binary fields considered (except for NIST (163,68) and SECG (113,34), where the multiplier given in [3] obtains the best values). Furthermore, the new approach also presents the lowest delays for most of the fields implemented (except for NIST (163,66) and SECG (113,34), where the method introduced in [6] gets lowest delays). With respect to area complexity, the multiplier given in [3] presents the lowest number of LUTs in most cases. There is not an specific method getting the lowest number of slices. In comparison with the splitting method with hard parenthesized restrictions given in [7], it can be observed that the new approach is more area and time efficient in all implemented fields. Therefore, the hard restrictions imposed by the parenthesis for the addition of the different terms in

[7] in order to reduce the number of XOR levels made the synthesizer could not perform an optimized mapping into the FPGA's logic blocks. This optimization could be done in the non-parenthesized implementations given in Table V that offer the synthesis tool more freedom to find an optimized implementation of the $GF(2^m)$ multiplier.

TABLE V
COMPARISON OF $GF(2^m)$ MULTIPLIERS.

	LUTs	Slices	Time (ns)	A×T	(m, n)
[2]	34	**11**	9.86	335.24	
[8]	35	14	**9.62**	336.70	
[3]	35	13	10.10	353.50	(8,2)
[6]	37	14	9.68	358.16	
[7]	40	13	9.90	396.00	
This work	**33**	12	9.77	**322.41**	
[2]	1836	586	22.63	41548.68	
[8]	1794	585	20.37	36543.78	
[3]	**1749**	566	20.91	36571.59	(64,23)
[6]	1825	580	20.21	36883.25	
[7]	1854	642	21.28	39453.12	
This work	1769	**541**	**20.18**	**35698.42**	
[2]	5747	2672	21.39	122928.33	
[8]	5501	2864	23.29	128118.29	
[3]	5424	2637	21.77	118080.48	(113,4)
[6]	5778	2469	21.28	122955.84	SECG
[7]	5944	**2115**	21.30	126607.20	
This work	**5420**	2571	**20.94**	**113494.80**	
[2]	5560	2849	23.58	131104.80	
[8]	5505	2682	23.38	128706.90	
[3]	**5445**	2563	20.84	**113473.80**	(113,34)
[6]	5813	2361	**20.36**	118352.68	SECG
[7]	5909	**2073**	21.73	128402.57	
This work	5474	2507	21.59	118183.66	
[2]	6487	3122	23.47	152249.89	
[8]	6420	3045	23.75	152475.00	
[3]	**6305**	2024	21.15	133350.75	(122,49)
[6]	6834	2287	21.83	149186.22	
[7]	6858	1992	21.86	149915.88	
This work	6361	**1951**	**20.95**	**133262.95**	
[2]	8370	3511	23.54	197029.80	
[8]	8301	3915	23.77	197314.77	
[3]	**8139**	2657	21.63	176046.57	(139,59)
[6]	8900	2960	22.29	198381.00	
[7]	8998	3031	21.55	193906.90	
This work	8222	**2543**	**21.35**	**175539.70**	
[2]	9466	3888	25.27	239205.82	
[8]	9406	3804	23.91	224897.46	
[3]	**9252**	3156	21.98	203358.96	(148,72)
[6]	9996	3329	22.40	223910.40	
[7]	9943	3112	22.31	221828.33	
This work	9314	**3104**	**21.76**	**202672.64**	
[2]	11425	4053	25.20	287910.00	
[8]	11379	4433	23.52	267634.08	
[3]	**11179**	**3361**	23.66	264495.14	(163,66)
[6]	12155	4056	**22.48**	273244.40	NIST
[7]	12293	4015	22.95	282124.35	
This work	11295	3621	22.77	**257187.15**	
[2]	11422	4205	24.20	276412.40	
[8]	11379	4349	24.01	273209.79	
[3]	**11172**	**3105**	22.40	**250252.80**	(163,68)
[6]	12187	3876	22.83	278229.91	NIST
[7]	12334	4430	23.82	293795.88	
This work	11330	3697	**22.39**	253678.70	

V. CONCLUSION

In this work, efficient Xilinx Artix-7 FPGA implementations of $GF(2^m)$ bit-parallel canonical basis multiplier based on type II irreducible pentanomials have been presented. These pentanomials are important because they are abundant and all five binary fields recommended by NIST for ECDSA can be constructed using such irreducible polynomials.

A new approach for the computation of the product coefficients has been considered. It is based on the use of $\mathbf{S_i}$ and $\mathbf{T_i}$ functions that can be split into sums of product terms implemented as complete binary trees of XOR gates with different depths. The addition of binary trees with the same depth can reduce the delay of the multiplier. However, this restriction could not be efficient for a synthesis tool to map that expressions into FPGA's logic blocks. In order to optimize the synthesis of the multiplier, in this work the splitting of $\mathbf{S_i}$ and $\mathbf{T_i}$ functions has been used, but the restriction imposed by the addition of binary trees with the same depth has been removed. In this case, Xilinx XST tool had freedom to optimize the synthesis of binary field polynomial basis multipliers.

In order to illustrate the new approach, a specific example for $GF(2^8)$ has been given. Furthermore, several $GF(2^m)$ multipliers have been described in VHDL and post-place and route implementation results in Artix-7 have been reported. Experimental results have shown that the proposed $GF(2^m)$ multiplier implementations improve the $area \times time$ parameter when compared with similar multipliers found in the literature. Furthermore, the new approach also presents the lowest delay for most of the binary fields used for implementation.

ACKNOWLEDGMENT

This work has been supported by the EU (FEDER) and the Spanish MINECO, under grants TIN 2015-65277-R and TIN2012-32180.

REFERENCES

[1] E.D. Mastrovito, "VLSI Architectures for Multiplication Over Finite Fields $GF(2^m)$", *Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes, Proc. Sixth Int'l Conf., AAECC-6*, New York: Springer-Verlag, Rome, pp. 297-309, July 1988.

[2] C. Paar, "Efficient VLSI Architectures for Bit Parallel Computation in Galois Fields", PhD Thesis, Universität GH Essen, 1994.

[3] A. Reyhani-Masoleh and M.A. Hasan, "Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over $GF(2^m)$", *IEEE Trans. Computers*, vol. 53, no. 8, pp. 945-959, August 2004.

[4] T. Zhang and K.K. Parhi, "Systematic Design of Original and Modified Mastrovito Multipliers for General Irreducible Polynomials", *IEEE Trans. Computers*, vol. 50, no. 7, pp. 734-749, July 2001.

[5] F. Rodríguez-Henríquez and Ç.K. Koç, "Parallel Multipliers Based on Special Irreducible Pentanomials", *IEEE Trans. Computers*, vol. 52, no. 12, pp. 1535-1542, December 2003.

[6] J.L. Imaña, "Efficient polynomial basis multipliers for Type II irreducible pentanomials", *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 11, pp. 795-799, November 2012.

[7] J.L. Imaña: 'High-Speed Polynomial Basis Multipliers over $GF(2^m)$ for Special Pentanomials', *IEEE Trans. Circuits and Systems I-Regular Papers*, vol. 63, no. 1, pp. 58-69, January 2016.

[8] B. Rashidi, R.R. Farashahi and S.M. Sayedi, "Efficient Implementation of Low Time Complexity and Pipelined Bit-Parallel Polynomial Basis Multiplier over Binary Finite Fields", *Int. Journal of Information Security*, vol. 7, no. 2, pp. 101-114, July 2015.

[9] SEC 2. Standards for Efficient Cryptography Group, "Recommended Elliptic Curve Domain Parameters". Version 1.0, 2000.