FPGA and ASIC Implementations of the η_T Pairing in Characteristic Three

Jean-Luc Beuchat, Hiroshi Doi, Kaoru Fujita, Atsuo Inomata, Piseth Ith, Akira Kanaoka, Masayoshi Katouno, Masahiro Mambo, Eiji Okamoto, Takeshi Okamoto, Takaaki Shiga, Masaaki Shirase, Ryuji Soga, Tsuyoshi Takagi, Ananda Vithanage, and Hiroyasu Yamamoto

***Abstract*—Since their introduction in constructive cryptographic applications, pairings over (hyper)elliptic curves are at the heart of an ever increasing number of protocols. As they rely critically on efficient implementations of pairing primitives, the study of hardware accelerators has become an active research area.**

In this paper, we propose two coprocessors for the reduced η_T pairing introduced by Barreto *et al.* as an alternative means of computing the Tate pairing on supersingular elliptic curves. We prototyped our architectures on FPGAs. According to our place-and-route results, our coprocessors compare favorably with other solutions described in the open literature. We also present the first ASIC implementation of the reduced η_T pairing.

***Index Terms*—Tate pairing, η_T pairing, elliptic curve cryptography, finite field arithmetic, hardware accelerator.**

I. Introduction

In the mid-nineties, Menezes, Okamoto & Vanstone [2] and Frey & Rück [3] introduced the Weil and Tate pairings in cryptography as a tool to attack the discrete logarithm problem on some classes of elliptic curves defined over finite fields. A few years later, Mitsunari, Sakai & Kasahara [4], Sakai, Oghishi & Kasahara [5], and Joux [6] discovered constructive properties of pairings. Their respective works initiated an extensive study of pairing-based cryptography, and an ever increasing number of protocols based on the Weil or the Tate pairing have appeared in the literature: identity-based encryption [7], short signature [8], and efficient broadcast encryption [9] to mention but a few. As noticed by Dutta, Barua & Sarkar [10], such protocols rely critically on efficient algorithms and implementations of pairing primitives.

According to [11], [12], when dealing with general curves providing common levels of security, the Tate pairing seems to be more efficiently computable than the Weil pairing. In 1986, Miller described the first iterative algorithm to compute the Tate pairing [13], [14]. Significant improvements were independently proposed by Barreto *et al.* [15] and Galbraith *et al.* [16] in 2002. One year later, Duursma & Lee gave a closed formula in the case of characteristic three [17]. In 2004, Barreto *et al.* [18] introduced the η_T approach, which further shortens the loop of Miller's algorithm.

This paper describes the design of two hardware accelerators for the η_T pairing in characteristic three. Section II provides the reader with a brief overview of pairing computation. As detailed in that section, the considered pairing algorithm relies heavily on arithmetic over $\mathbb{F}_{3^{6m}}$, a degree-6 extension of the base field of the curve. However, thanks to a tower field representation, all operations over $\mathbb{F}_{3^{6m}}$ can be replaced by arithmetic over $\mathbb{F}_{3^m}$. We describe hardware arithmetic operators over $\mathbb{F}_{3^m}$ and explain how to take advantage of the tower field in Section III. We then propose two hardware accelerators for the η_T pairing (Section IV). We have prototyped our architectures on FPGA, and propose the first ASIC implementation of the η_T pairing in characteristic three. Section V summarizes our implementation results on FPGA and ASIC, and provides the reader with a comprehensive comparison with previously published architectures.

II. Computation of the Modified Tate Pairing in Characteristic Three

Given a positive integer m coprime to 6, we consider a supersingular[1] elliptic curve E over $\mathbb{F}_{3^m}$, defined by the equation $y^2 = x^3 - x + b$, with $b \in \{-1, 1\}$. According to [18], there is no loss of generality from considering this case since these curves offer the same level of security for pairing applications as any supersingular elliptic curve over $\mathbb{F}_{3^m}$. The number N of rational points of E over the finite field $\mathbb{F}_{3^m}$ is given by $N = \#E(\mathbb{F}_{3^m}) = 3^m + 1 + \mu b 3^{\frac{m+1}{2}}$, with

$$\mu = \begin{cases} +1 & \text{if } m \equiv 1, 11 \pmod{12}, \text{ or} \\ -1 & \text{if } m \equiv 5, 7 \pmod{12}. \end{cases}$$

J.-L. Beuchat, A. Kanaoka, P. Ith, M. Mambo, and E. Okamoto are with the Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan.

H. Doi is with the Graduate School of Information Security, Institute of Information Security, 2-14-1 Tsuruya-cho Kanagawa-ku, Yokohama 221-0835, Japan.

K. Fujita, M. Katouno, R. Soga, and H. Yamamoto are with FDK Module System Technology Corporation, 1 Kamanomae, Kamiyunagaya-machi, Jyoban, Iwaki-shi, Japan.

A. Inomata is with the Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara, 630-0192, Japan.

T. Okamoto is with the Department of Computer Science, Tsukuba University of Technology, 4-12-7 Kasuga, Tsukuba, Ibaraki, 305-8521, Japan.

T. Shiga and A. Vithanage are with FDK Corporation, 1 Kamanomae, Kamiyunagaya-machi, Jyoban, Iwaki-shi, Japan.

M. Shirase and T. Takagi are with the School of Systems Information Science, Future University-Hakodate, 116-2 Kamedanakano-cho, Hakodate, Hokkaido, 041-8655, Japan.

This work was supported by the New Energy and Industrial Technology Development Organization (NEDO), Japan. This paper is an extended version of [1].

[1] See for instance Theorem V.3.1 in [19] for a definition.

A. Modified Tate Pairing

Let ℓ be the largest prime factor of N. $E(\mathbb{F}_{3^m})[\ell]$ denotes the ℓ-torsion subgroup of $E(\mathbb{F}_{3^m})$, *i.e.* the set of points $P \in E(\mathbb{F}_{3^m})$ such that $[\ell]P = \mathcal{O}$, where $\mathcal{O}$ is the point at infinity of the elliptic curve E. The modified Tate pairing is a function that takes as input two points of $E(\mathbb{F}_{3^m})[\ell]$ and outputs an element of the group of ℓth roots of unity $\mu_\ell = \{R \in \mathbb{F}^*_{3^{km}} : R^\ell = 1\}$.

The *embedding degree* or *security multiplier* is the least positive integer k for which μ_ℓ is contained in the multiplicative group $\mathbb{F}^*_{3^{km}}$ (*i.e.* k is the smallest integer such that ℓ divides $3^{km} - 1$). The considered curve has an embedding degree of $k = 6$, which is the maximum value possible for supersingular elliptic curves, and hence seems to be an attractive choice for pairing implementation.

The modified Tate pairing of order ℓ is then the map

$$\hat{e}(\cdot, \cdot) : E(\mathbb{F}_{3^m})[\ell] \times E(\mathbb{F}_{3^m})[\ell] \to \mathbb{F}^*_{3^{6m}}$$

given by

$$\hat{e}(P, Q) = f_{\ell,P}(\psi(Q))^{(3^{6m}-1)/\ell},$$

where

- ψ is a distortion map (the concept of a distortion map was introduced in [20]) from $E(\mathbb{F}_{3^m})[\ell]$ to $E(\mathbb{F}_{3^{6m}})[\ell] \setminus E(\mathbb{F}_{3^m})[\ell]$ defined as $\psi(x_Q, y_Q) = (\rho - x_Q, y\sigma)$ for all $Q = (x_Q, y_Q) \in E(\mathbb{F}_{3^m})[\ell]$, where ρ and σ are elements of $\mathbb{F}_{3^{6m}}$ satisfying the equations $\rho^3 - \rho - b = 0$ and $\sigma^2 + 1 = 0$ [15]. Note that $\{1, \sigma, \rho, \sigma\rho, \rho^2, \sigma\rho^2\}$ is a basis of $\mathbb{F}_{3^{6m}}$ over $\mathbb{F}_{3^m}$. We will therefore represent an element $R \in \mathbb{F}_{3^{6m}}$ as $R = r_0 + r_1\sigma + r_2\rho + r_3\sigma\rho + r_4\rho^2 + r_5\sigma\rho^2$, where the r_i's belong to $\mathbb{F}_{3^m}$.
- $f_{n,P}$, for $n \in \mathbb{N}$ and $P \in E(\mathbb{F}_{3^m})[\ell]$ is a rational function defined over $E(\mathbb{F}_{3^{6m}})[\ell]$ with divisor $(f_{n,P}) = n(P) - ([n]P) - (n-1)(\mathcal{O})$ (see [19] or [21] for an account of divisors). We consider here the definition proposed by Barreto *et al.* [15], where $f_{n,P}$ is evaluated on a point rather than on a divisor.
- $f_{\ell,P}(\psi(Q))$ is only defined up to ℓth powers, which is undesirable in most of the cryptographic applications. The powering by $(3^{6m}-1)/\ell$, referred to as *final exponentiation*, allows one to obtain a unique value in a multiplicative subgroup of $\mathbb{F}^*_{3^{6m}}$.

Choosing an order of low Hamming weight provides computational savings in Miller's algorithm. However, ℓ being a quotient of N by a small cofactor, it does not have a small Hamming weight. Galbraith *et al.* [16] noted that one can compute the modified Tate pairing of order ℓ with respect to the group order N (note that N divides $3^{6m} - 1$):

$$f_{\ell,P}(\psi(Q))^{(3^{6m}-1)/\ell} = f_{N,P}(\psi(Q))^{(3^{6m}-1)/N}.$$

In the following, M denotes the final exponent of the modified Tate pairing of order N:

$$M = \frac{3^{6m}-1}{N} = \left(3^{3m}-1\right)\left(3^m+1\right)\left(3^m + 1 - \mu b 3^{\frac{m+1}{2}}\right).$$

The modified Tate pairing satisfies the following properties:

- *Bilinearity*. For all A, B, $C \in \mathbb{F}_{3^m}[\ell]$,
$$\hat{e}(A+B, C) = \hat{e}(A, C)\hat{e}(B, C) \quad \text{and}$$
$$\hat{e}(A, B+C) = \hat{e}(A, B)\hat{e}(A, C).$$
- *Non-degeneracy.* $\hat{e}(P, P) \neq 1$, for all $P \neq \mathcal{O}$.
- *Computability.* $\hat{e}$ can be efficiently computed.

B. The Duursma-Lee Approach

Duursma & Lee [17] proposed to compute the order $3^{3m}+1$ modified Tate pairing. This approach simplifies both Miller's algorithm and the final exponentiation[2]. Furthermore, Duursma & Lee showed that the number of iterations of Miller's algorithm can be reduced from $3m$ to m iterations [17].

C. The η_T Approach

Barreto *et al.* [18] introduced the η_T pairing as "an alternative means of computing the Tate pairing on certain supersingluar curves" [22, page 108]. They suggest to compute $\hat{e}(P, Q)$ using an order $T \in \mathbb{Z}$ that is smaller than N. Their main result is a lemma which gives a method to select T such that $\eta_T(P, Q)^M$ is a non-degenerate bilinear pairing [18]. In characteristic three they choose $T = 3^m - N = -\mu b 3^{\frac{m+1}{2}} - 1$ and show that their method gives a further halving of the length of the loop compared to the Duursma & Lee approach. The η_T pairing is defined as follows:

$$\eta_T(P, Q) = \left\{ \begin{array}{rl} f_{T,P}(\psi(Q)) & \text{if } T > 0\text{, or} \\ f_{-T,-P}(\psi(Q)) & \text{if } T < 0. \end{array} \right. \tag{1}$$

Defining $T' = -\mu b T = 3^{\frac{m+1}{2}} + \mu b$ and $P' = [-\mu b]P$, we rewrite Equation (1) as $\eta_T(P, Q)^M = f_{T',P'}(\psi(Q))^M$. Then, the techniques proposed by Duursma & Lee [17] allow one to simplify the computation of $f_{n,P}$ in Miller's algorithm:

$$f_{T',P'}(\psi(Q)) = \left(\prod_{i=0}^{\frac{m-1}{2}} g_{[3^i]P'}(\psi(Q))^{3^{\frac{m-1}{2}-i}} \right) l_{P'}(\psi(Q)),$$

where

- g_V is the rational function introduced by Duursma & Lee [17], defined over $E(\mathbb{F}_{3^{6m}})[\ell]$, and having divisor $(g_V) = 3(V) + ([-3]V) - 4(\mathcal{O})$. For all $V = (x_V, y_V) \in E(\mathbb{F}_{3^m})[\ell]$ and $(x, y) \in E(\mathbb{F}_{3^{6m}})[\ell]$, it is defined as:
$$g_V(x, y) = y_V^3 y - (x_V^3 - x + b)^2.$$
- l_V, for all $V = (x_V, y_V) \in E(\mathbb{F}_{3^m})[\ell]$, is the equation of the line corresponding to the addition of $\left[3^{\frac{m+1}{2}}\right] V$ with $[\mu b]V$. It is defined for all $(x, y) \in E(\mathbb{F}_{3^{6m}})[\ell]$:
$$l_V(x, y) = y - (-1)^{\frac{m+1}{2}} y_V (x - x_V) - \mu b y_V.$$

As pointed out by Barreto *et al.* [18], the computation of $f_{T',P'}(\psi(Q))$ requires cubings over $\mathbb{F}_{3^{6m}}$ because of the exponent $3^{\frac{m-1}{2}-i}$ inside the main product. They suggested to bring the powering into the formulae as a Frobenius action, or to compute the product in reverse. Both approaches allow one to replace two cubings over $\mathbb{F}_{3^m}$ and one cubing over

[2] The exponent is $(3^{6m}-1)/(3^{3m}+1) = 3^{3m} - 1$.

$\mathbb{F}_{3^{6m}}$ by two cube roots over $\mathbb{F}_{3^m}$ at each iteration. However, the second one turns out to be slightly more effective since it also saves three multiplications over $\mathbb{F}_{3^m}$ when multiplying by $l_{P'}(\psi(Q))$ (see [23] for further details). Note that the Duursma-Lee algorithm also comes in two flavors: the original one involves cube roots and Kwon proposed a cube root-free version in [24].

Algorithm 1 Cube-root-free reversed-loop algorithm for computing the η_T pairing [23].

```
Input: $P, Q \in E(\mathbb{F}_{3^m})[\ell]$. The algorithm involves a local
    variable $t \in \mathbb{F}_{3^m}$, and two local variables $R$ and $S \in \mathbb{F}_{3^{6m}}$.
Output: $\eta_T(P,Q)^M \in \mathbb{F}^*_{3^{6m}}$.
1.  $x_P \leftarrow x_P + b$;
2.  $y_P \leftarrow -\mu b y_P$;
3.  $x_Q \leftarrow x_Q^3$;   $y_Q \leftarrow y_Q^3$;
4.  $t \leftarrow x_P + x_Q$;
5.  $R \leftarrow (y_P t - y_Q\sigma - y_P\rho) \cdot (-t^2 + y_P y_Q\sigma - t\rho - \rho^2)$;
6.  for $j \leftarrow 1$ to $\frac{m-1}{2}$ do
7.      $R \leftarrow R^3$;
8.      $x_Q \leftarrow x_Q^9 - b$;   $y_Q \leftarrow -y_Q^9$;
9.      $t \leftarrow x_P + x_Q$;   $u \leftarrow y_P y_Q$;
10.     $S \leftarrow -t^2 + u\sigma - t\rho - \rho^2$;
11.     $R \leftarrow R \cdot S$;
12. end for
13. return $R^M$;
```

Fong *et al.* showed that extracting a square root in $\mathbb{F}_{2^m}$ requires approximately the time of a field multiplication and proposed an improved scheme for trinomials [25]. Barreto extended this approach to cube root in characteristic three [26]: if $\mathbb{F}_{3^m}$ admits an irreducible trinomial $x^m + f_n x^n + f_0$ (f_n, $f_0 \in \{-1, 1\}$) with the property $n \equiv m \pmod 3$, then five shifts and five additions allow one to implement this operation. Nevertheless, even if computing a cube root is not a difficult operation, it requires specific hardware and a slightly more complex control and datapath. In this work, we decided to minimize the area of the Arithmetic and Logic Unit (ALU) and considered a cube root-free version of the reversed-loop approach described by Algorithm 1. Consider the operand $S \in \mathbb{F}_{3^{6m}}$ (line 10) and note that it is sparse (*i.e.* some of its terms are trivial). This property will allow us to optimize the computation of $R \cdot S$ in Section III-B2.

The relationship between the modified Tate pairing and the reduced η_T pairing is given by [27]:

$$\hat{e}(P,Q)^M = \eta_T\left([-\mu b]\,P, \left[3^{\frac{3m-1}{2}}\right]Q\right)^M,$$

where $[-\mu b]\,P = (x_P, -\mu b y_P)$ and $\left[3^{\frac{3m-1}{2}}\right]Q = \left(\sqrt[3]{x_Q} - b, (-1)^{\frac{m+1}{2}}\sqrt[3]{y_Q}\right)$. We can modify Algorithm 1 as follows to obtain $\hat{e}(P,Q)^M$:

- Since we compute the pairing with $(x_p, -\mu b y_P)$, line 2 becomes $y_p \leftarrow -\mu b \cdot (-\mu b y_P) = y_p$ and can be discarded.
- It is no longer necessary to compute the cube of x_Q and y_Q (line 3). We have now $x_Q \leftarrow x_Q - b$.
- Let $x'_P = x_P + b$ and $x'_Q = x_Q - b$. Since $t = x'_P + x'_Q = x_P + x_Q$ (line 4), we can actually remove lines 1 and 3.

It is worth noticing that we obtain a cube root-free algorithm and that the modified Tate pairing requires less operations than the reduced η_T pairing in this case.

D. Final Exponentiation

Fermat's little theorem provides us with an effective way to perform the final exponentiation of the reduced η_T pairing. As pointed out by Barreto *et al.*, "the result of raising to $3^{3m} - 1$ produces an element of order $3^{3m} + 1$, so that any further inversion reduces to a simple conjugation" [18, page 248]. The main loop of Algorithm 1 returns $R = \eta_T(P,Q) \in \mathbb{F}^*_{3^{6m}}$. Writing $R = R_0 + R_1\sigma$, where R_0 and $R_1 \in \mathbb{F}^*_{3^{3m}}$, we obtain:

$$V = R^{3^{3m}-1} = \frac{(R_0^2 - R_1^2) + R_0R_1\sigma}{R_0^2 + R_1^2},$$

Algorithm 2 summarizes the computation of the final exponentiation. When $\mu b = -1$, the computation of $W' = W^{-\mu b}$ on line 4 is a dummy operation. Let us write $W = W_0 + W_1\sigma$, where W_0 and $W_1 \in \mathbb{F}^*_{3^{3m}}$. Since W is an element of order $3^{3m} + 1$ [18], the inversion is completely free when $\mu b = 1$:

$$\begin{aligned} W' = W^{-1} = W^{3^{3m}} &= (W_0 + \sigma W_1)^{3^{3m}} \\ &= W_0^{3^{3m}} + \sigma^{3^{3m}} W_1^{3^{3m}} = W_0 - \sigma W_1. \end{aligned}$$

It suffices to propagate the sign corrections in the product $V \cdot W'$. Whereas the computation of $\eta_T(P,Q)$ involves only sparse multiplications over $\mathbb{F}_{3^{6m}}$ (Algorithm 1, line 11), the final exponentiation requires a full multiplication over $\mathbb{F}_{3^{6m}}$ (Algorithm 2, line 6). Note that the computation of V and W involves only operations over $\mathbb{F}_{3^{3m}}$. Algorithms to compute $R^{3^{3m}-1}$ and V^{3^m+1} are for instance detailed in [23].

Algorithm 2 Final exponentiation of the reduced η_T pairing.

```
Input: $R = \eta_T(P,Q) \in \mathbb{F}^*_{3^{6m}}$.
Output: $R^M \in \mathbb{F}^*_{3^{6m}}$.
1. $V \leftarrow R^{3^{3m}-1}$;
2. $V \leftarrow V^{3^m+1}$;
3. $W \leftarrow V^{3^{\frac{m+1}{2}}}$;
4. $W' \leftarrow W^{-\mu b}$;
5. $V \leftarrow V^{3^m+1}$;
6. return $V \cdot W'$;
```

III. Arithmetic over $\mathbb{F}_{3^m}$ and $\mathbb{F}_{3^{6m}}$

Thanks to the tower field representation, all operations over $\mathbb{F}_{3^{6m}}$ and $\mathbb{F}_{3^{3m}}$ in Algorithms 1 and 2 can be replaced by arithmetic over $\mathbb{F}_{3^m}$. For instance, 12 multiplications, 11 additions, and a single inversion over $\mathbb{F}_{3^m}$ allow one to carry out the inversion over $\mathbb{F}_{3^{3m}}$ involved in the computation of $V = R^{3^{3m}-1}$. We describe here the hardware operators we designed for arithmetic over $\mathbb{F}_{3^m}$ (Section III-A) and the algorithms for sparse multiplication and cubing over $\mathbb{F}_{3^{6m}}$ (Section III-B). We refer the reader to [23] for further details about other operations.

A. Arithmetic over $\mathbb{F}_{3^m}$

In the following, elements of $\mathbb{F}_{3^m}$ are encoded using a polynomial basis. Given a degree-m irreducible polynomial $f(x) \in \mathbb{F}_3[x]$, we have $\mathbb{F}_{3^m} \cong \mathbb{F}_3[x]/(f(x))$. Consequently, each element of $\mathbb{F}_{3^m}$ is represented as a polynomial of degree less than m with coefficients in $\mathbb{F}_3$.

1) Addition and Subtraction over $\mathbb{F}_{3^m}$: Since they are performed component-wise, addition and subtraction over $\mathbb{F}_{3^m}$ are rather straightforward operations. Each element of $\mathbb{F}_3$ being encoded by two bits, the addition of a_i and $b_i \in \mathbb{F}_3$ on most of Altera or Xilinx FPGAs requires two 4-input LUTs.

2) Multiplication over $\mathbb{F}_{3^m}$: Among the many modular multipliers described in the open literature (see for instance [28]–[30]), we selected a Most Significant Element (MSE) first array multiplier based on Song & Parhi's work [31] to carry out $a(x)b(x) \bmod f(x)$. At step i we compute a degree-$(m + D - 2)$ polynomial $t(x)$ which is the sum of D partial products: $t(x) = \sum_{j=0}^{D-1} a_{Di+j}x^j b(x)$. A degree-$(m + D - 1)$ polynomial $s(x)$, updated according to the celebrated Horner's rule, allows us to accumulate the partial products:

$$s(x) \leftarrow t(x) + x^D \cdot (s(x) \bmod f(x)).$$

Thus, after $\lceil m/D \rceil$ steps, this algorithm returns a degree-$(m+D-1)$ polynomial $s(x)$ which is congruent to $a(x)b(x)$ modulo $f(x)$. The circuit described by Song & Parhi requires dedicated hardware to compute $p(x) = s(x) \bmod f(x)$ [31]. We suggest to achieve the final modulo $f(x)$ reduction by performing an additional iteration with $a_{-j} = 0$, $1 \le j \le D$. Since $t(x)$ is now equal to zero, we have: $s(x) = x^D \cdot (a(x)b(x) \bmod f(x))$. Therefore, it suffices to consider the m most significant coefficients of $s(x)$ to get the result (i.e. $p(x) = s(x)/x^D$). Algorithm 3 summarizes this multiplication scheme. Figure 1 describes the architecture of an array multiplier processing $D = 3$ coefficients at each clock cycle.

Algorithm 3 MSE multiplication over $\mathbb{F}_{3^m}$.

Input: A degree-m irreducible monic polynomial $f(x) = x^m + f_{m-1}x^{m-1} + \ldots + f_1x + f_0$, two degree-$(m-1)$ polynomials $a(x)$, and $b(x)$. We assume that $a_{-j} = 0$, $1 \le j \le D$. The algorithm requires a degree-$(m+D-1)$ polynomial $s(x)$ as well as a degree-$(m + D - 2)$ polynomial $t(x)$ for intermediate computations.

Output: $p(x) = a(x)b(x) \bmod f(x)$.

```
1. $s(x) \leftarrow 0$;
2. for $i$ in $\lceil m/D \rceil - 1$ downto $-1$ do
3.     $t(x) \leftarrow \sum_{j=0}^{D-1} a_{Di+j}x^j b(x)$;
4.     $s(x) \leftarrow t(x) + x^D \cdot (s(x) \bmod f(x))$;
5. end for
6. $p(x) \leftarrow s(x)/x^D$;
```

The cost of the modular reduction (line 4) depends on D and $f(x)$. Assume that $f(x)$ is an irreducible trinomial such that $f(x) = x^m + f_nx^n + f_0$, where f_0 and $f_n \in \mathbb{F}_3$, and $0 < n < m$. We have:

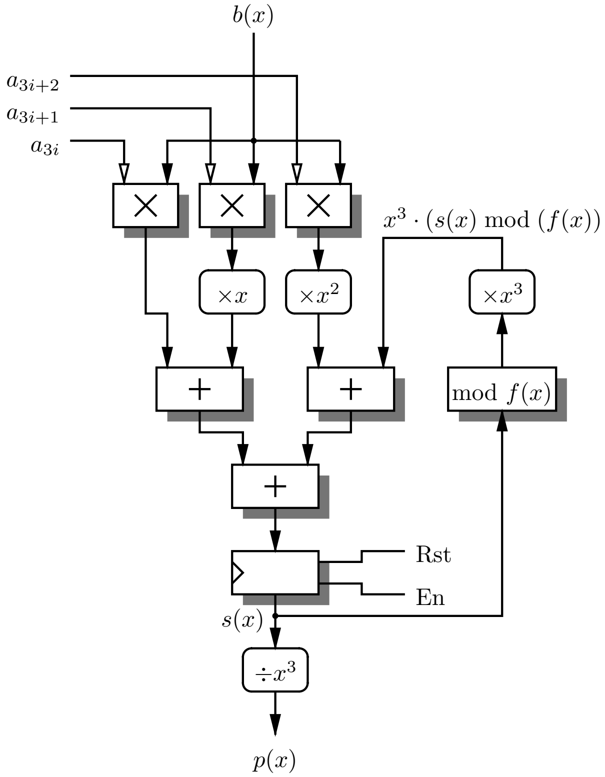


Fig. 1. MSE array multipliers processing $D = 3$ coefficients at each clock cycle. Boxes with rounded corners involve only wiring.

$$s(x) \bmod f(x) = \left(\sum_{i=0}^{D-1} s_{m+i}x^{m+i} + \sum_{i=0}^{m-1} s_ix^i \right) \bmod f(x).$$

Since $x^m \equiv -f_nx^n - f_0 \pmod{f(x)}$, we note that:

$$s_{m+i}x^{m+i} \equiv s_{m+i}(-f_nx^n - f_0)x^i \pmod{f(x)}.$$

In the following, we assume that $D \le m - n$ to ensure that the degree of $s_{m+i}(-f_nx^n - f_0)x^i$, $0 \le i \le D-1$, is at most equal to $m - 1$. Thus, we obtain:

$$\begin{aligned} s(x) \bmod f(x) &= \sum_{i=0}^{D-1} s_{m+i}(-f_nx^n - f_0)x^i + \sum_{i=0}^{m-1} s_ix^i \\ &= -\sum_{i=0}^{D-1} s_{m+i}f_nx^{n+i} - \sum_{i=0}^{D-1} s_{m+i}f_0x^i \\ &\quad + \sum_{i=0}^{m-1} s_ix^i, \end{aligned}$$

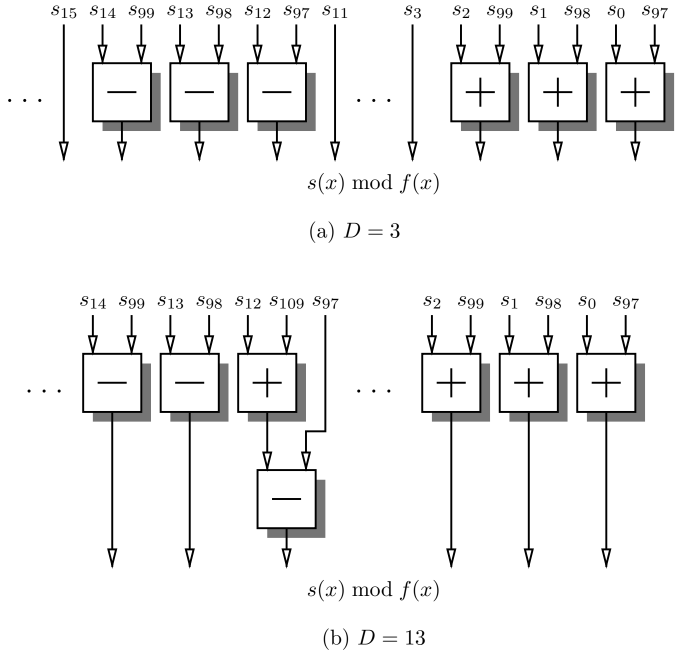
and the modular reduction involves $2D$ additions (or subtractions) over $\mathbb{F}_3$. When $D \le n$, the degree of x^i, $0 \le i \le D-1$, is always smaller than the one of x^{n+i} and the modular reduction requires a single stage of 2-input adders (or subtracters) over $\mathbb{F}_3$. Thus, selecting the parameter D such that $D \le \min(n, m-n)$ allows one to achieve the shortest critical path in the case of an irreducible trinomial.

Let us consider for instance the irreducible trinomial $f(x) = x^{97} + x^{12} + 2$ (*i.e.* $m = 97$, $n = 12$, $f_0 = 2$, and $f_{12} = 1$).

Since -2 is congruent to 1 modulo 3, we have:

$$s(x) \bmod f(x) = -\sum_{i=0}^{D-1} s_{97+i}x^{i+12} + \sum_{i=0}^{D-1} s_{97+i}x^{i} + \sum_{i=0}^{96} s_i x^i.$$

Figures 2a and 2b describe the circuits performing the modular reduction when $D = 3$ and $D = 13$, respectively. In the first case, a single stage of 2-input adders allows one to carry out $s(x) \bmod f(x)$. However, in the second case, a 2-input adder and a 2-input subtracter are required to compute $s_{13} + s_{109} - s_{97}$.


s_{15} s_{14} s_{99} s_{13} s_{98} s_{12} s_{97} s_{11} s_3 s_2 s_{99} s_1 s_{98} s_0 s_{97}
$s(x) \bmod f(x)$
s_{14} s_{99} s_{13} s_{98} s_{12} s_{109} s_{97} s_2 s_{99} s_1 s_{98} s_0 s_{97}
$s(x) \bmod f(x)$

(a) $D = 3$

(b) $D = 13$

Fig. 2. Computation of $s(x) \bmod f(x)$ when $f(x) = x^{97} + x^{12} + 2$ for (a) $D = 3$ and (b) $D = 13$.

3) Cubing over $\mathbb{F}_{3^m}$*:* Let us now consider the computation of $b(x) = a(x)^3$ over $\mathbb{F}_{3^m}$. Cubing over $\mathbb{F}_{3^m}$ consists of reducing the following expression modulo $f(x)$:

$$b(x) = a(x)^3 = \left(\sum_{i=0}^{m-1} a_i x^{3i}\right) \bmod f(x).$$

A formal reduction allows us to express each coefficient b_i of the result as a linear combination of the coefficient of $a(x)$. Therefore, a cubing operator mainly consists of a D'-operand adder and some extra wiring to permute the coefficients of $a(x)$. The main challenge here is to find an irreducible polynomial minimizing D'.

Let us consider again the irreducible trinomial $f(x) = x^{97} + x^{12} + 2$. Reducing $a(x)^3$ modulo $f(x)$, we obtain:

$$\begin{aligned} b_0 &= a_{93} + a_{89} + a_0, & b_2 &= a_{33}, \\ b_1 &= a_{65} - a_{61}, & b_3 &= a_{94} + a_{90} + a_1, \\ \ldots &= \ldots, & b_{96} &= a_{32}. \end{aligned}$$

The most complex operation involved here is the addition of $D' = 3$ elements of $\mathbb{F}_3$. Since we consider a cube root-free η_T pairing algorithm, $f(x) = x^{97} + x^{12} + 2$ is a good candidate: it has a simple cubing formula and allows one to perform the modulo $f(x)$ reduction involved in the multiplication over $\mathbb{F}_{3^m}$ by means of a single stage of 2-input adders as long as $D \leq 12$. However, if one intends to implement a pairing algorithm with cube roots, one should consider a further constraint to select an irreducible trinomial. Barreto noticed that the cost of computing cube roots in $\mathbb{F}_{3^m}$ is only $O(m)$ if $m \equiv n \pmod 3$ [26]. Despite of a slightly more complex cubing formula, $f(x) = x^{97} + x^{16} + 2$ is for instance a better choice in this case.

4) Inversion over $\mathbb{F}_{3^m}$*:* Since the computation of the reduced η_T pairing involves a single inversion over $\mathbb{F}_{3^m}$ in the final exponentiation, we perform this operation according to Fermat's little theorem and Itoh & Tsujii's algorithm [32]. Thus, inversion over $\mathbb{F}_{3^m}$ is carried out by means of cubings and multiplications over $\mathbb{F}_{3^m}$ and does not require specific hardware resources.

B. Arithmetic over $\mathbb{F}_{3^{6m}}$

1) Cubing over $\mathbb{F}_{3^{6m}}$*:* When we compute the η_T pairing according to Algorithm 1, we raise $R = r_0 + r_1\sigma + r_2\rho + r_3\sigma\rho + r_4\rho^2 + r_5\sigma\rho^2 \in \mathbb{F}_{3^{6m}}$ to the cube at each iteration of the main loop. Since $\rho^3 = \rho + b$ and $\sigma^3 = -\sigma$, we obtain:

$$\begin{aligned} R^3 \quad = \quad & \left(r_0^3 + br_2^3 + r_4^3\right) + \left(-r_1^3 - br_3^3 - r_5^3\right)\sigma \\ & + \left(r_2^3 - br_4^3\right)\rho + \left(-r_3^3 + br_5^3\right)\sigma\rho + r_4^3\rho^2 - r_5^3\sigma\rho^2. \end{aligned}$$

This operation involves six cubings and six additions (or subtractions) over $\mathbb{F}_{3^m}$.

2) Multiplication over $\mathbb{F}_{3^{6m}}$*:*

a) Full Multiplication over $\mathbb{F}_{3^{6m}}$*.:* Karatsuba-Ofman's algorithm allows one to compute the product of two polynomials belonging to $\mathbb{F}_{3^{6m}}$ by means of 18 multiplications and 58 additions (or subtractions) over $\mathbb{F}_{3^m}$ (see for instance [33]). An improvement was recently proposed by Gorla *et al.* [34]: they represented elements of $\mathbb{F}_{3^{6m}}$ as degree-2 polynomials with coefficients in $\mathbb{F}_{3^{2m}}$ and took advantage of Lagrange interpolation to compute a product over $\mathbb{F}_{3^{6m}}$ by means of 5 multiplications over $\mathbb{F}_{3^{2m}}$. Each of these multiplications is then carried out according to Karatsuba-Ofman's scheme, and the total cost of a multiplication over $\mathbb{F}_{3^{6m}}$ is equal to 15 multiplications and 67 additions (or subtractions) over $\mathbb{F}_{3^m}$.

b) Sparse Multiplication over $\mathbb{F}_{3^{6m}}$*.:* Consider now the computation of the reduced η_T pairing (Algorithm 1), where each iteration of the loop requires a sparse multiplication over $\mathbb{F}_{3^{6m}}$. As pointed out by Bertoni *et al.* [35] and Granger *et al.* [36], the product $R \cdot S$ (line 11) can be computed by means of 13 multiplications and 50 additions (or subtractions) over $\mathbb{F}_{3^m}$ according to Karatsuba-Ofman's scheme. Again, the approach introduced by Gorla *et al.* allows one to further reduce the cost of this operation to 12 multiplications and 51 additions (or subtractions) over $\mathbb{F}_{3^m}$ (see [23] for details). Two further multiplications are needed to compute $y_P y_Q$ as well as t^2.

In this paper, we focus on parallel architectures featuring several multipliers. In this context, it seems more interesting to find a good trade-off between the number of multiplications and additions, to share registers between multipliers, and to reduce the number of accesses to memory. Let $R = r_0 +$

$r_1\sigma + r_2\rho + r_3\sigma\rho + r_4\rho^2 + r_5\sigma\rho^2$ and $C = c_0 + c_1\sigma + c_2\rho + c_3\sigma\rho + c_4\rho^2 + c_5\sigma\rho^2$ be two elements of $\mathbb{F}_{3^{6m}}$. We write each coefficient c_i as the sum of two elements $c_i^{(0)}$ and $c_i^{(1)} \in \mathbb{F}_{3^m}$. Thanks to this notation we define the product $C = R \cdot (-t^2 + y_P y_Q \sigma - t\rho - \rho^2)$ as follows, where $b \in \{-1, 1\}$ is a parameter of the elliptic curve:

$$\begin{aligned}
c_0^{(0)} &= -br_4 t - br_2, & c_0^{(1)} &= -r_0 t^2 - r_1 y_P y_Q,\\
c_1^{(0)} &= -br_5 t - br_3, & c_1^{(1)} &= r_0 y_P y_Q - r_1 t^2,\\
c_2^{(0)} &= -r_0 t - br_4 + bc_0^{(0)}, & c_2^{(1)} &= -r_2 t^2 - r_3 y_P y_Q,\\
c_3^{(0)} &= -r_1 t - br_5 + bc_1^{(0)}, & c_3^{(1)} &= r_2 y_P y_Q - r_3 t^2,\\
c_4^{(0)} &= -r_2 t - r_0 - r_4, & c_4^{(1)} &= -r_4 t^2 - r_5 y_P y_Q,\\
c_5^{(0)} &= -r_3 t - r_1 - r_5, & c_5^{(1)} &= r_4 y_P y_Q - r_5 t^2.
\end{aligned}$$

Note that the computation of the $c_i^{(0)}$'s, $0 \le i \le 5$, requires six multiplications over $\mathbb{F}_{3^m}$ and depends neither on t^2 nor on $y_P y_Q$. Thus, we can perform eight multiplications over $\mathbb{F}_{3^m}$ in parallel (t^2, $y_P y_Q$, and $r_i t$, $0 \le i \le 5$). Consider now $c_0^{(1)}$ and $c_1^{(1)}$ and assume that $(r_0 + r_1)$ and $(y_P y_Q - t^2)$ are stored in registers. Karatsuba-Ofman's algorithm allows one to compute $c_0^{(1)}$ and $c_1^{(1)}$ by means of three multiplications and three additions over $\mathbb{F}_{3^m}$:

$$\begin{aligned}
c_0^{(1)} &= -r_0 t^2 - r_1 y_P y_Q,\\
c_1^{(1)} &= (r_0 + r_1)(y_P y_Q - t^2) + r_0 t^2 - r_1 y_P y_Q\\
&= r_0 y_P y_Q - r_1 t^2.
\end{aligned}$$

Therefore, the computation of the $c_i^{(1)}$'s involves nine multiplications over $\mathbb{F}_{3^m}$, which can be carried out in parallel. Algorithm 4 summarizes this multiplication scheme involving 17 multiplications and 29 additions (or subtractions) over $\mathbb{F}_{3^m}$.

Since the computation of the nine products p_i, $8 \le i \le 16$, depends on p_6 and p_7, we can not perform the 17 multiplications over $\mathbb{F}_{3^m}$ in parallel and have to proceed in two steps (Algorithm 4, lines 1 and 3). Therefore, we suggest to design a coprocessor embedding nine multipliers over $\mathbb{F}_{3^m}$, denoted by M_i, $0 \le i \le 8$, in the following. A control unit will contain the instructions required to implement the sparse multiplication over $\mathbb{F}_{3^{6m}}$ on such an architecture.

A careful scheduling allows one to share operands between up to three multipliers, thus saving hardware resources (Table I): during the first step (9 multiplications over $\mathbb{F}_{3^m}$), M_0, M_1, and M_2 respectively compute $r_0 t$, $r_2 t$, and $r_4 t$. The MSE multiplier described in Section III-A2 stores its first operand in a shift register, and its second operand in a standard register. Since a shift register is more complex (an operand is loaded in parallel, and then shifted), we load the common operand t in this component. At the end of these multiplications, the three registers still contain r_0, r_2, and r_4. Therefore it suffices to load t^2 in the shift register before starting the second step (9 multiplications over $\mathbb{F}_{3^m}$). Figure 3a describes the operator we designed to perform three multiplications with a common operand. The same architecture allows for computing $r_1 t$, $r_3 t$, $r_5 t$, $r_1 y_P y_Q$, $r_3 y_P y_Q$, and $r_5 y_P y_Q$. The five remaining multiplications involve a slightly more complex component (Figure 3b): two shift registers are required to compute t^2 and $y_P y_Q$ since there is no common operand. At the end of the first multiplication cycle, a dedicated subtracter computes $y_P y_Q - t^2$ and stores the result in the shift registers.

TABLE I
SPARSE MULTIPLICATION OVER $\mathbb{F}_{3^{6m}}$: SCHEDULING.

	1st step: 8 multiplications over $\mathbb{F}_{3^m}$	**2nd step: 9 multiplications over $\mathbb{F}_{3^m}$**
M$_0$	$p_0 = r_0 \cdot t$	$p_8 = r_0 \cdot t^2$
M$_1$	$p_2 = r_2 \cdot t$	$p_{11} = r_2 \cdot t^2$
M$_2$	$p_4 = r_4 \cdot t$	$p_{14} = r_4 \cdot t^2$
M$_3$	$p_1 = r_1 \cdot t$	$p_9 = r_1 \cdot y_P y_Q$
M$_4$	$p_3 = r_3 \cdot t$	$p_{12} = r_3 \cdot y_P y_Q$
M$_5$	$p_5 = r_5 \cdot t$	$p_{15} = r_5 \cdot y_P y_Q$
M$_6$	$p_6 = t \cdot t$	$p_{10} = (r_0 + r_1) \cdot (y_P y_Q - t^2)$
M$_7$	$p_7 = y_P \cdot y_Q$	$p_{13} = (r_2 + r_3) \cdot (y_P y_Q - t^2)$
M$_8$	–	$p_{16} = (r_4 + r_5) \cdot (y_P y_Q - t^2)$

Consider the additions occurring in the fourth step of Algorithm 4. Interestingly enough, they involve at most one result of each block of three multipliers (Figure 3). Instead of a large multiplexer selecting the output of one multiplier among nine, we include a multiplexer in each block and connect a 3-operand adder to the outputs of our multiplication units. In order to also take advantage of these adders while performing a multiplication, each block of three multipliers has an additional input D1 that allows for bypassing the multipliers.

IV. HARDWARE IMPLEMENTATION

In this section, we propose two architectures to compute the reduced η_T pairing for the field $\mathbb{F}_3[x]/(x^{97} + x^{12} + 2)$ and the curve $y^2 = x^3 - x + 1$ (*i.e.* $b = 1$). This choice of parameters allows us to easily compare our work against the many pairing accelerators for $m = 97$ described in the open literature. It is nonetheless important to note that the architectures and algorithms presented here can be easily adapted to different parameters.

A. Hardware Accelerator for the Reduced η_T Pairing

Figure 4 describes the architecture of our hardware accelerator for the η_T pairing calculation (Algorithm 1). The ALU and the datapath are strongly related to the pairing algorithm and our sparse multiplication over $\mathbb{F}_{3^m}$ scheme. Nine multipliers over $\mathbb{F}_{3^m}$ sharing shift registers allow us to carry out the products p_i, $0 \le i \le 16$, of our sparse multiplication scheme (Algorithm 4) in two steps, according to the scheduling summarized in Table I. The 3-operand adder/subtracter allows for computing the c_i's. Recall that we raise the result of a sparse multiplication to the cube at the beginning of each iteration of the considered η_T pairing algorithm. This operation consists of six cubings and six additions over $\mathbb{F}_{3^m}$ (Section III-B1). Therefore, we connected the output of the 3-operand adder/subtracter to a cubing operator. This approach allows us to bypass the register file and to save clock cycles when raising to the cube over $\mathbb{F}_{3^{6m}}$. Inputs and outputs, as well as intermediate results, are stored in a dual-ported RAM (DPRAM) implemented using embedded memory blocks available in the FPGA. The control unit mainly consists of a ROM containing the microcode of Algorithms 1

Algorithm 4 Sparse multiplication over $\mathbb{F}_{3^{6m}}$.

```
Input: $R = r_0 + r_1\sigma + r_2\rho + r_3\sigma\rho + r_4\rho^2 + r_5\sigma\rho^2 \in \mathbb{F}_{3^{6m}}$; $t$, $y_P$, and $y_Q \in \mathbb{F}_{3^m}$; the parameter $b \in \{-1, 1\}$ of the
    supersingular elliptic curve.
Output: $C = R \cdot (-t^2 + y_P y_Q \sigma - r_0\rho - \rho^2)$.
1. Compute in parallel (8 multiplications and 3 additions over $\mathbb{F}_{3^m}$):
        $p_i \leftarrow r_i \cdot t,\ 0 \le i \le 5$;    $p_6 \leftarrow t \cdot t$;    $p_7 \leftarrow y_P \cdot y_Q$;
        $s_0 \leftarrow r_0 + r_1$;    $s_1 \leftarrow r_2 + r_3$;    $s_2 \leftarrow r_4 + r_5$;
2. Compute in parallel (7 additions over $\mathbb{F}_{3^m}$):
        $s_3 \leftarrow p_7 - p_6$;  // $y_P y_Q - t^2$      $c_2 \leftarrow br_4 + p_0$;  // $br_4 + r_0 t$      $c_4 \leftarrow r_0 + p_2$;  // $r_0 + r_2 t$
        $c_0 \leftarrow br_2 + bp_4$;  // $br_2 + br_4 t$    $c_3 \leftarrow br_5 + p_1$;  // $br_5 + r_1 t$      $c_5 \leftarrow r_1 + p_3$;  // $r_1 + r_3 t$
        $c_1 \leftarrow br_3 + bp_5$;  // $br_3 + br_5 t$
3. Compute in parallel (9 multiplications and 4 additions over $\mathbb{F}_{3^m}$):
        $p_8 \leftarrow r_0 \cdot p_6$;  // $r_0 t^2$                          $p_{13} \leftarrow s_1 \cdot s_3$;  // $(r_2 + r_3)(y_P y_Q - t^2)$    $c_2 \leftarrow c_2 + bc_0$;
        $p_9 \leftarrow r_1 \cdot p_7$;  // $r_1 y_P y_Q$                       $p_{14} \leftarrow r_4 \cdot p_6$;  // $r_4 t^2$                      $c_3 \leftarrow c_3 + bc_1$;
        $p_{10} \leftarrow s_0 \cdot s_3$;  // $(r_0 + r_1)(y_P y_Q - t^2)$    $p_{15} \leftarrow r_5 \cdot p_7$;  // $r_5 y_P y_Q$                  $c_4 \leftarrow c_4 + r_4$;
        $p_{11} \leftarrow r_2 \cdot p_6$;  // $r_2 t^2$                       $p_{16} \leftarrow s_2 \cdot s_3$;  // $(r_4 + r_5)(y_P y_Q - t^2)$    $c_5 \leftarrow c_5 + r_5$;
        $p_{12} \leftarrow r_3 \cdot p_7$;  // $r_3 y_P y_Q$
4. Compute in parallel (15 additions over $\mathbb{F}_{3^m}$):
        $c_0 \leftarrow -c_0 - p_8 - p_9$;    $c_2 \leftarrow -c_2 - p_{11} - p_{12}$;    $c_4 \leftarrow -c_4 - p_{14} - p_{15}$;
        $c_1 \leftarrow -c_1 + p_{10} + p_8 - p_9$;    $c_3 \leftarrow -c_3 + p_{13} + p_{11} - p_{12}$;    $c_5 \leftarrow -c_5 + p_{16} + p_{14} - p_{15}$;
```

and 4. When $m = 97$ and $D = 3$, we need 4849 clock cycles to compute $\eta_T(P, Q))$ according to Algorithm 1.

Since algorithms for multiplication over $\mathbb{F}_{3^{3m}}$ and $\mathbb{F}_{3^{6m}}$ do not share operands between several multipliers, it turns out to be impossible to take advantage of the full parallelism of our architecture when performing the final exponentiation (Algorithm 2). Thus, it seems attractive to supplement the η_T pairing accelerator with dedicated hardware to raise $\eta_T(P, Q)$ to the Mth power. Beuchat *et al.* [37] proposed a unified arithmetic operator performing addition, subtraction, accumulation, cubing, and multiplication over $\mathbb{F}_{3^m}$. When $m = 97$ and $D = 3$, this coprocessor performs the final exponentiation in 4082 clock cycles. We can therefore pipeline the computation of the η_T pairing and the final exponentiation. In the following, we assume that we keep the pipeline busy and that we obtain a new result after 4849 clock cycles (*i.e.* we neglect the overhead introduced by our approach to get the first result). This coprocessor for the final exponentiation requires 64 registers to store elements of $\mathbb{F}_{3^m}$. On FPGA, they are efficiently implemented using the embedded memory blocks.

B. A Coprocessor for Arithmetic over $\mathbb{F}_{3^m}$

We also investigated a second architecture based on a coprocessor for arithmetic over $\mathbb{F}_{3^m}$ embedding nine multipliers, an addition unit (able to carry out addition, subtraction, and accumulation), and a cubing unit (Figure 5). Since we implement the main loop of the η_T pairing (Algorithm 1) and the final exponentiation (Algorithm 2) on the same hardware, each multiplier must have two input registers and we cannot share shift registers between up to three multipliers over $\mathbb{F}_{3^m}$ anymore.

The sparse multiplications over $\mathbb{F}_{3^{6m}}$ are carried out according to Algorithm 4. Since performing 15 or 18 multiplications over $\mathbb{F}_{3^m}$ requires the same number of clock cycles on our coprocessor, we implemented the multiplication over $\mathbb{F}_{3^{6m}}$ of the final exponentiation according to Karatsuba-Ofman's scheme in order to minimize the number of additions over $\mathbb{F}_{3^m}$. When $m = 97$ and $D = 3$, the computation of $\eta_T(P, Q)$ and the final exponentiation require 6560 clock cycles and 2527 clock cycles, respectively.

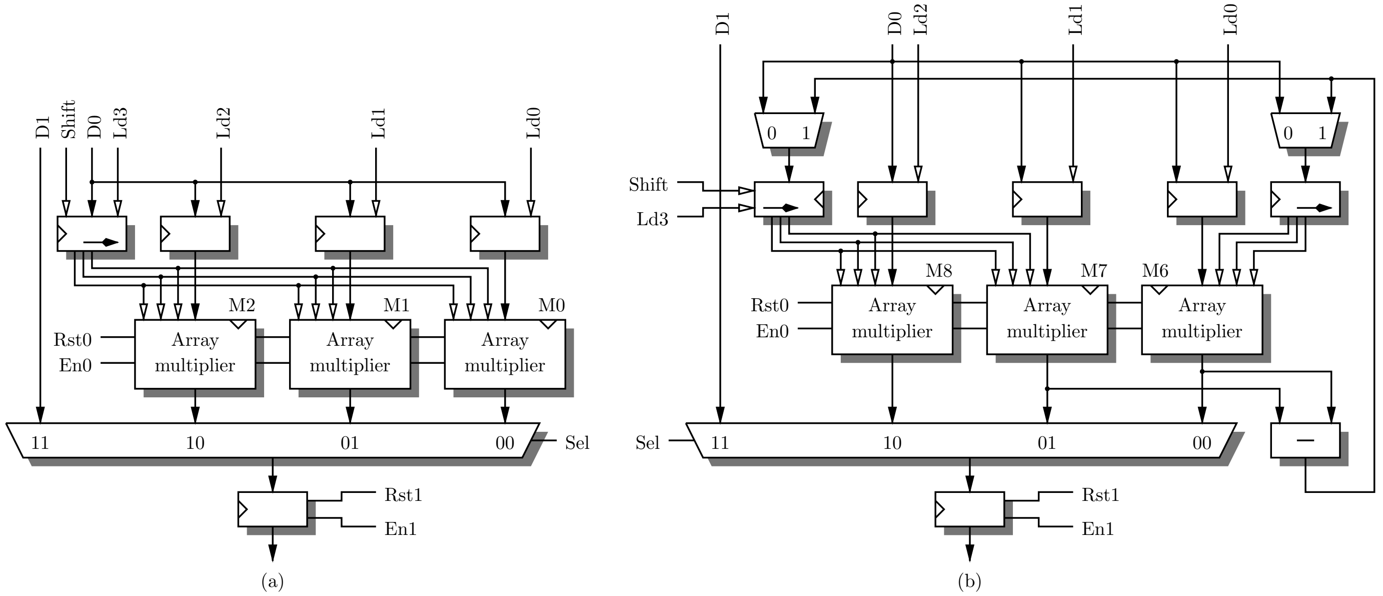
This coprocessor for arithmetic over $\mathbb{F}_{3^m}$ is of course slower than the architecture described in the previous section when considering the computation of the η_T pairing (Algorithm 1). However, it is much more versatile and allows for the implementation of a wider range of algorithms: besides pairing computation, it is for instance possible to perform a scalar multiplication, which is a crucial operation in pairing-based cryptography.

V. Results and Comparisons

A. FPGA Implementation

Our reduced η_T pairing accelerator and the coprocessor for arithmetic over $\mathbb{F}_{3^m}$ were captured in the VHDL language and prototyped on Altera Cyclone II and Xilinx Virtex-II Pro FPGAs. Table II summarizes our place-and-route results.

Several processors for the reduced η_T pairing (Table II) and the modified Tate pairing (Table III) have already been



(a) (b)

Fig. 3. Building blocks for sparse multiplication over $\mathbb{F}_{3^{6m}}$. (a) Three multipliers with a common operand. (b) Two multipliers with a common operand.

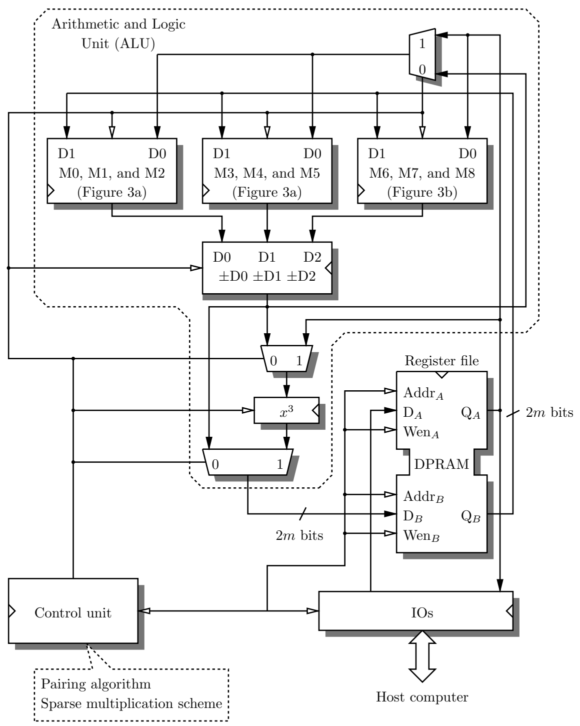


Fig. 4. Architecture of the coprocessor for the η_T pairing calculation. The ALU embeds the building blocks for sparse multiplication over $\mathbb{F}_{3^{6m}}$ described by Figure 3.

published. Since $\hat{e}(P,Q)$ can be computed from $\eta_T(P,Q)^M$ at almost no extra cost (Section II-C), we can compare our architectures against all these results. Note that the hardware accelerators proposed by other researchers are always implemented on Xilinx FPGAs. Therefore, we decided to compute the Area-Time (AT) product in terms of slices to provide the reader with a fair comparison (each slice of a Virtex-II, Virtex-II Pro, or Virtex-4 embeds two 4-input function generators and two storage elements). Note that register files implemented in memory blocks are not included in the AT product.

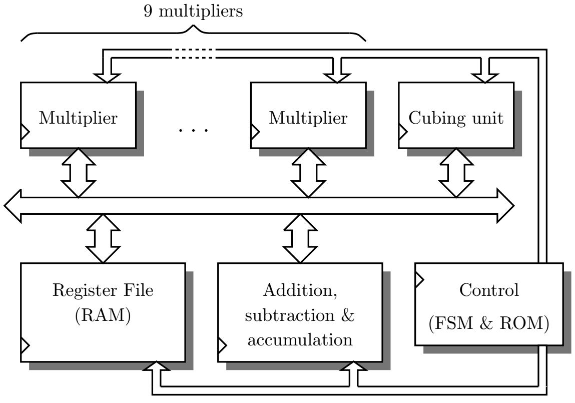


Fig. 5. Coprocessor for arithmetic over $\mathbb{F}_{3^m}$ amenable for pairing computation.

To our best knowledge, Jiang [40] designed the fastest η_T pairing core (Table II). However, our processors achieves a better area-time trade-off. Additionally, our approach allows for reaching higher levels of security without risking to exhaust the FPGA resources. Jiang's coprocessor already requires one of the largest FPGAs available now.

In order to easily study the trade-off between calculation time and circuit area, Ronan *et al.* [39] wrote a C program which automatically generates a VHDL description of a coprocessor and its control according to the number of multipliers to be included and D. The ALU also embeds an adder, a subtracter, a cubing unit, and an inversion unit. Their fastest architecture embeds 8 multipliers ($D = 4$) and is very similar to the hardware accelerator for the reduced η_T pairing proposed in Section IV-B. However, since our multipliers process $D = 3$ coefficients at each clock cycles and the inversion over $\mathbb{F}_{3^m}$ is performed according to Fermat's

TABLE II
HARDWARE ACCELERATORS FOR THE REDUCED η_T PAIRING (POST-PLACE-AND-ROUTE FIGURES). THE PARAMETER D REFERS TO THE NUMBER OF COEFFICIENTS PROCESSED AT EACH CLOCK CYCLE BY A MULTIPLIER.

	Curve	**Technology**	**# mult.**	**Area**	**Freq. [MHz]**	**Calc. time [μs]**	**AT product**
Ronan *et al.* [38]	$C(\mathbb{F}_{2^{103}})$	Virtex-II Pro 100	20 ($D = 4$)	21021 slices	51	206	4.33
			20 ($D = 8$)	24290 slices	46	152	3.79
			20 ($D = 16$)	30464 slices	41	132	4.02
Ronan *et al.* [39]	$E(\mathbb{F}_{3^{97}})$	Virtex-II Pro 100	5 ($D = 4$)	10540 slices	84.8	187	1.97
			8 ($D = 4$)	15401 slices	84.8	183	2.81
Beuchat *et al.* [27]	$E(\mathbb{F}_{3^{97}})$	Virtex-II Pro 20	1 ($D = 3$)	1896 slices	156	178	0.34
			1 ($D = 7$)	2711 slices	128	117	0.32
			1 ($D = 15$)	4455 slices	105	92	0.41
	$E(\mathbb{F}_{2^{239}})$	Virtex-II Pro 20	1 ($D = 7$)	2366 slices	199	196	0.46
			1 ($D = 15$)	2736 slices	165	127	0.35
			1 ($D = 31$)	4557 slices	123	107	0.49
Jiang [40]	$E(\mathbb{F}_{3^{97}})$	Virtex-4 LX 200	Not specified	74105 slices	77.7	20.9	1.55
Coprocessor for the η_T pairing & coprocessor for the final exponentiation							
	$E(\mathbb{F}_{3^{97}})$	Cyclone II EP2C35	9 ($D = 3$)	18000 LEs	149	33	–
	$E(\mathbb{F}_{3^{97}})$	Virtex-II Pro 30	9 ($D = 3$)	10897 slices	147	33	0.36
Coprocessor for arithmetic over $\mathbb{F}_{3^m}$ – PairingLite							
FPGA	$E(\mathbb{F}_{3^{97}})$	Virtex-II Pro 30	9 ($D = 3$)	10262 slices	142	64	0.66
		Cyclone II EP2C70	9 ($D = 3$)	15293 LEs	240	39.6	–
ASIC	$E(\mathbb{F}_{3^{97}})$	0.18μm CMOS	9 ($D = 3$)	193765 NAND	200	46.7	–

TABLE III
HARDWARE ACCELERATORS FOR THE TATE PAIRING (POST-PLACE-AND-ROUTE FIGURES). THE PARAMETER D REFERS TO THE NUMBER OF COEFFICIENTS PROCESSED AT EACH CLOCK CYCLE BY A MULTIPLIER. THE ARCHITECTURE PROPOSED BY KÖMÜRCÜ & SAVAS [41] DOES NOT IMPLEMENT THE FINAL EXPONENTIATION. BARENGHI *et al.* [42] COMPUTE THE TATE PAIRING OVER $\mathbb{F}_p$, WHERE p IS A 512-BIT PRIME NUMBER.

	Curve	**Technology**	**# mult.**	**Area**	**Freq. [MHz]**	**Calc. time [μs]**	**AT product**
Keller *et al.* [43]	$E(\mathbb{F}_{2^{251}})$	Virtex-II 6000	1 ($D = 6$)	3788 slices	40	4900	18.56
			3 ($D = 6$)	6181 slices	40	3200	19.78
			9 ($D = 6$)	13387 slices	40	2600	34.81
Keller *et al.* [44]	$E(\mathbb{F}_{2^{251}})$	Virtex-II 6000	13 ($D = 1$)	16621 slices	50	6440	107.04
			13 ($D = 6$)	21955 slices	43	2580	56.64
			13 ($D = 10$)	27725 slices	40	2370	65.71
Kerins *et al.* [33]	$E(\mathbb{F}_{3^{97}})$	Virtex-II Pro 125	18 ($D = 4$)	55616 slices	15	850	47.27
Li *et al.* [45]	$E(\mathbb{F}_{2^{283}})$	Virtex-4 FX 140	12 ($D = 32$)	55844 slices	159.8	590	32.95
Kömürcü & Savas [41]	$E(\mathbb{F}_{3^{97}})$	Virtex-II Pro 4	20 ($D = 1$)	14267 slices	77.3	250.7	3.58
		0.25μm CMOS	20 ($D = 1$)	10 mm^2	78	250	–
Grabher & Page [46]	$E(\mathbb{F}_{3^{97}})$	Virtex-II Pro 4	1 ($D = 4$)	4481 slices	150	432.3	1.94
Ronan *et al.* [47]	$E(\mathbb{F}_{2^{313}})$	Virtex-II Pro 100	14 ($D = 4$)	34675 slices	55	203	7.04
			14 ($D = 8$)	41078 slices	50	124	5.09
			14 ($D = 12$)	44060 slices	33	146	6.43
Shu *et al.* [48]	$E(\mathbb{F}_{2^{239}})$	Virtex-II Pro 100	6 ($D = 16$), 1 ($D = 4$), 1 ($D = 2$), and 1 ($D = 1$)	25287 slices	84	41	1.04
Barenghi *et al.* [42]	$E(\mathbb{F}_p)$	Virtex-II 8000	4 (Montgomery)	33857 slices	135	1610	54.51

little theorem, we achieve a smaller area. Furthermore, thanks to our sparse multiplication algorithm, we compute the η_T pairing in 6560 clock cycles, whereas Ronan *et al.* need 10089 clock cycles to complete the same task. They unrolled the exponent M and grouped the inversions together. Their final exponentiation is therefore much more expensive than ours: 5440 clock cycles against 2527.

Grabher and Page designed a coprocessor dealing with $\mathbb{F}_{3^m}$ arithmetic, which is controlled by a general purpose processor [46]. Their hardware accelerator embeds a single multiplier over $\mathbb{F}_{3^m}$. Our architectures requires roughly 2.5 times as many slices, while performing up to nine multiplications in parallel.

B. ASIC Implementation

We designed the first ASIC implementation of the reduced η_T pairing (0.18μm CMOS technology). Our two hardware accelerators require roughly the same number of slices on Xilinx FPGAs. However, the architecture based on a coprocessor for the η_T pairing and a coprocessor for the final exponentiation involves two register files. Since they are implemented using the numerous memory blocks available in modern FPGAs, they are not taken into account in our area measurement. We decided to minimize the area of the chip and selected the coprocessor for arithmetic over $\mathbb{F}_{3^m}$ with $D = 3$. Furthermore, this architecture is more versatile than the η_T pairing accelerator described in Section IV-A. A simple modification of the control unit would allow us to support scalar multiplication in a new version of the ASIC. Table IV summarizes our place-and-route results. The PairingLite chip computes the reduced η_T pairing (Algorithms 1 and 2) in 46.7μs. This timing includes the 52 and 78 clock cycles required to write the coordinates P and Q in the register file and to read the result, respectively.

TABLE IV
ASIC IMPLEMENTATION OF THE REDUCED η_T PAIRING (PLACE-AND-ROUTE FIGURES).

Process	TSMC CL018G (0.18μm CMOS)
Area	193765 2NAND gates
Frequency	200 MHz
Calculation time	46.7μs
Core size	3849.6μm×3849.6μm
Package	TSMC CQFP 100 pin
Operating voltage	VDD CORE: 1.8V, VDD IO: 3.3V
Power consumption	Total power: 671.739mW
Consumption current	Total current: 373.188mA
Temperature conditions	25^{o}C
Output terminal	Drive capability 4 mA

Figures 6 and 7 describe the evaluation board we designed to test the PairingLite ASIC (on the left on Figure 6). We also included a Cyclone II device (on the right on Figure 6) to test our FPGA architectures, and a true random number generator manufactured by FDK corporation to produce secret keys. A USB port allows one to connect the board to a computer. The figures reported in Table IV were measured using this board.

In order to check that it is possible to correctly compute the reduced η_T pairing, we implemented the BLS short signature scheme [8]. The map-to-point function is computed in software. Then, the two pairings involved in the verification are performed in hardware on our evaluation board and in software on a desktop computer. We compare the results returned by the ASIC, the FPGA and the software. It takes 0.8ms to send the coordinates of points P and Q, compute the pairing on the ASIC, and read the result. Communications are clearly a bottleneck here, however, recall that the only purpose of our board is to serve as a prototype.

VI. Conclusion

We proposed two parallel architectures to compute the reduced η_T pairing in characteristic three and reported the first ASIC implementation of a pairing accelerator. Our coprocessors take advantage of a novel sparse multiplication algorithm over $\mathbb{F}_{3^{6m}}$. Instead of minimizing the number of multiplications over $\mathbb{F}_{3^m}$, we tried to find a good trade-off between the number of multiplications and additions over $\mathbb{F}_{3^m}$. Our method also allows for sharing operands between up to three multipliers and reduces the number of accesses to memory compared to other algorithms.

Our next challenge is to design a pairing accelerator providing the level of security of AES-128. We plan to make a thorough comparison between supersingular curves over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$. We will consider several architectures: small processors based on a single unified operator [23], accelerators embedding several parallel-serial multipliers, and massively parallel architectures based on a Karatsuba-Ofman multiplier. The study of the Ate pairing [49] would also be of interest, for it presents a large speedup when compared to the Tate pairing and also supports non-supersingular curves. Once the best curve and architecture will be defined, we'd like to design a coprocessor for pairing-based cryptography supporting the most widely used primitives (e.g. pairing, random number generation, scalar multiplication, hashing, etc).

Acknowledgment

The authors would like to thank the anonymous referees for their valuable comments. This work was supported by the New Energy and Industrial Technology Development Organization (NEDO), Japan.

References

[1] J.-L. Beuchat, M. Shirase, T. Takagi, and E. Okamoto, "An algorithm for the η_T pairing calculation in characteristic three and its hardware implementation," in *Proceedings of the 18th IEEE Symposium on Computer Arithmetic*, P. Kornerup and J.-M. Muller, Eds. IEEE Computer Society, 2007, pp. 97–104.

[2] A. Menezes, T. Okamoto, and S. A. Vanstone, "Reducing elliptic curves logarithms to logarithms in a finite field," *IEEE Transactions on Information Theory*, vol. 39, no. 5, pp. 1639–1646, Sep. 1993.

[3] G. Frey and H.-G. Rück, "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves," *Mathematics of Computation*, vol. 62, no. 206, pp. 865–874, Apr. 1994.

[4] S. Mitsunari, R. Sakai, and M. Kasahara, "A new traitor tracing," *IEICE Trans. Fundamentals*, vol. E85-A, no. 2, pp. 481–484, Feb 2002.

[5] R. Sakai, K. Ohgishi, and M. Kasahara, "Cryptosystems based on pairing," in *2000 Symposium on Cryptography and Information Security (SCIS2000), Okinawa, Japan*, Jan. 2000, pp. 26–28.

[6] A. Joux, "A one round protocol for tripartite Diffie-Hellman," in *Algorithmic Number Theory – ANTS IV*, ser. Lecture Notes in Computer Science, W. Bosma, Ed., no. 1838. Springer, 2000, pp. 385–394.

[7] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology – CRYPTO 2001*, ser. Lecture Notes in Computer Science, J. Kilian, Ed., no. 2139. Springer, 2001, pp. 213–229.

[8] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Advances in Cryptology – ASIACRYPT 2001*, ser. Lecture Notes in Computer Science, C. Boyd, Ed., no. 2248. Springer, 2001, pp. 514–532.

[9] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *Advances in Cryptology – CRYPTO 2005*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., no. 3621. Springer, 2005, pp. 258–275.

[10] R. Dutta, R. Barua, and P. Sarkar, "Pairing-based cryptographic protocols: A survey," 2004, cryptology ePrint Archive, Report 2004/64.

[11] R. Granger, D. Page, and N. P. Smart, "High security pairing-based cryptography revisited," in *Algorithmic Number Theory – ANTS VII*, ser. Lecture Notes in Computer Science, F. Hess, S. Pauli, and M. Pohst, Eds., no. 4076. Springer, 2006, pp. 480–494.

[12] N. Koblitz and A. Menezes, "Pairing-based cryptography at high security levels," in *Cryptography and Coding*, ser. Lecture Notes in Computer Science, N. P. Smart, Ed., no. 3796. Springer, 2005, pp. 13–36.

[13] V. S. Miller, "Short programs for functions on curves," 1986, available at http://crypto.stanford.edu/miller.

[14] ——, "The Weil pairing, and its efficient calculation," *Journal of Cryptology*, vol. 17, no. 4, pp. 235–261, 2004.

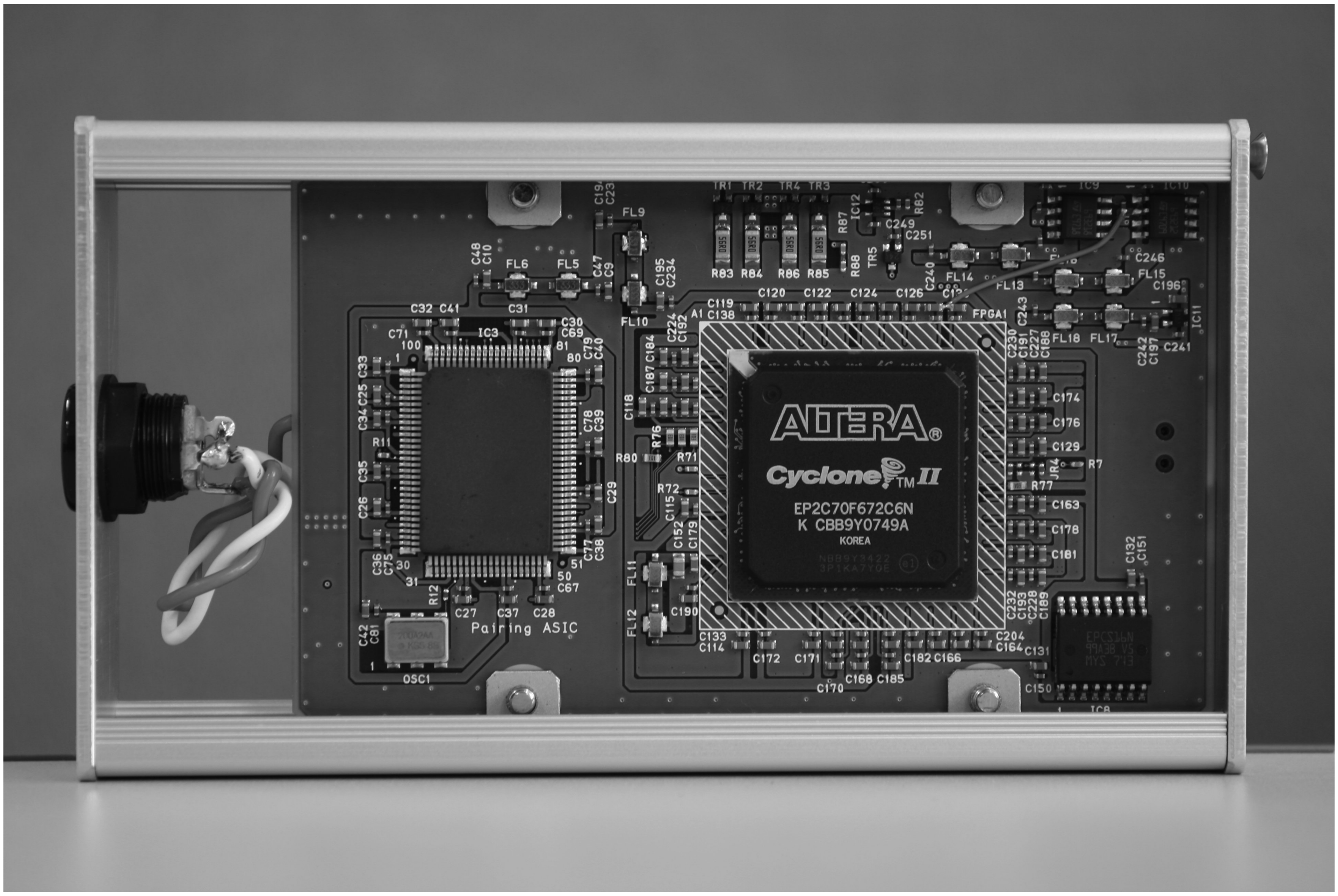


Fig. 6. Evaluation board for the PairingLite chip.

[15] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *Advances in Cryptology – CRYPTO 2002*, ser. Lecture Notes in Computer Science, M. Yung, Ed., no. 2442. Springer, 2002, pp. 354–368.

[16] S. D. Galbraith, K. Harrison, and D. Soldera, "Implementing the Tate pairing," in *Algorithmic Number Theory – ANTS V*, ser. Lecture Notes in Computer Science, C. Fieker and D. Kohel, Eds., no. 2369. Springer, 2002, pp. 324–337.

[17] I. Duursma and H. S. Lee, "Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$," in *Advances in Cryptology – ASIACRYPT 2003*, ser. Lecture Notes in Computer Science, C. S. Laih, Ed., no. 2894. Springer, 2003, pp. 111–123.

[18] P. S. L. M. Barreto, S. D. Galbraith, C. Ó hÉigeartaigh, and M. Scott, "Efficient pairing computation on supersingular Abelian varieties," *Designs, Codes and Cryptography*, vol. 42, pp. 239–271, 2007.

[19] J. H. Silverman, *The Arithmetic of Elliptic Curves*, ser. Graduate Texts in Mathematics. Springer-Verlag, 1986, no. 106.

[20] E. R. Verheul, "Evidence that XTR is more secure than supersingular elliptic curve cryptosystems," *Journal of Cryptology*, vol. 17, no. 4, pp. 277–296, 2004.

[21] L. C. Washington, *Elliptic Curves – Number Theory and Cryptography*, 2nd ed. CRC Press, 2008.

[22] C. Ó hÉigeartaigh, "Pairing computation on hyperelliptic curves of genus 2," Ph.D. dissertation, Dublin City University, 2006.

[23] J.-L. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto, M. Shirase, and T. Takagi, "Algorithms and arithmetic operators for computing the η_T pairing in characteristic three," *IEEE Transactions on Computers*, vol. 57, no. 11, pp. 1454–1468, Nov. 2008.

[24] S. Kwon, "Efficient Tate pairing computation for elliptic curves over binary fields," in *Information Security and Privacy – ACISP 2005*, ser. Lecture Notes in Computer Science, C. Boyd and J. M. González Nieto, Eds., vol. 3574. Springer, 2005, pp. 134–145.

[25] K. Fong, D. Hankerson, J. López, and A. Menezes, "Field inversion and point halving revisited," *IEEE Transactions on Computers*, vol. 53, no. 8, pp. 1047–1059, Aug. 2004.

[26] P. S. L. M. Barreto, "A note on efficient computation of cube roots in characteristic 3," 2004, cryptology ePrint Archive, Report 2004/305.

[27] J.-L. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto, and F. Rodríguez-Henríquez, "A comparison between hardware accelerators for the modified Tate pairing over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$," in *Proceedings of Pairing 2008*, ser. Lecture Notes in Computer Science. Springer, 2008, to appear. An extended version is available as Report 2008/115 of the Cryptology ePrint Archive.

[28] J. Guajardo, T. Güneysu, S. Kumar, C. Paar, and J. Pelzl, "Efficient hardware implementation of finite fields with applications to cryptography," *Acta Applicandae Mathematicae*, vol. 93, no. 1–3, pp. 75–118, Sep. 2006.

[29] J.-L. Beuchat, T. Miyoshi, J.-M. Muller, and E. Okamoto, "Horner's rule-based multiplication over GF(p) and GF(p^n): A survey," *International Journal of Electronics*, 2008, to appear.

[30] S. E. Erdem, T. Yamk, and Ç. K. Koç, "Polynomial basis multiplication over GF(2^m)," *Acta Applicandae Mathematicae*, vol. 93, no. 1–3, pp. 33–55, Sep. 2006.

[31] L. Song and K. K. Parhi, "Low energy digit-serial/parallel finite field multipliers," *Journal of VLSI Signal Processing*, vol. 19, no. 2, pp. 149–166, Jul. 1998.

[32] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases," *Information and Computation*, vol. 78, pp. 171–177, 1988.

[33] T. Kerins, W. P. Marnane, E. M. Popovici, and P. S. L. M. Barreto, "Efficient hardware for the Tate pairing calculation in characteristic three," in *Cryptographic Hardware and Embedded Systems – CHES 2005*, ser. Lecture Notes in Computer Science, J. R. Rao and B. Sunar, Eds., no. 3659. Springer, 2005, pp. 412–426.

[34] E. Gorla, C. Puttmann, and J. Shokrollahi, "Explicit formulas for efficient multiplication in $\mathbb{F}_{3^{6m}}$," in *Selected Areas in Cryptography –*

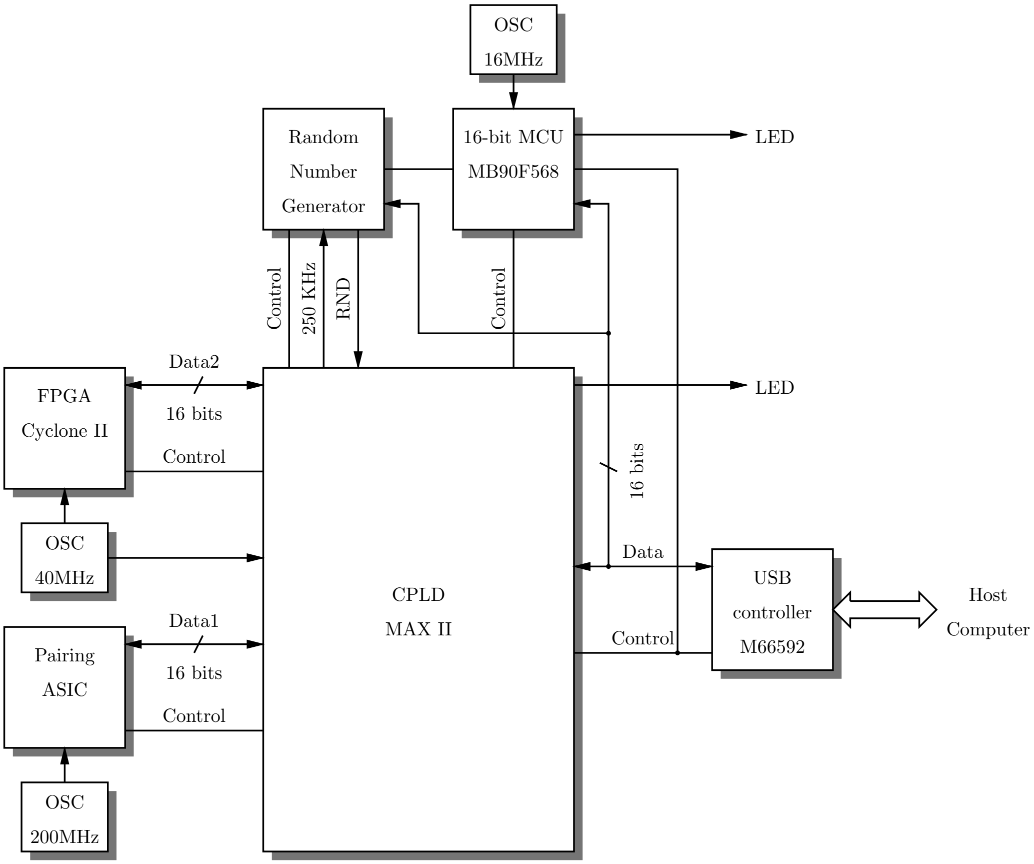


Fig. 7. Architecture of the evaluation board.

SAC 2007, ser. Lecture Notes in Computer Science, C. Adams, A. Miri, and M. Wiener, Eds., no. 4876. Springer, 2007, pp. 173–183.

[35] G. Bertoni, L. Breveglieri, P. Fragneto, and G. Pelosi, "Parallel hardware architectures for the cryptographic Tate pairing," in *Proceedings of the Third International Conference on Information Technology: New Generations (ITNG'06)*. IEEE Computer Society, 2006.

[36] R. Granger, D. Page, and M. Stam, "On small characteristic algebraic tori in pairing-based cryptography," *LMS Journal of Computation and Mathematics*, vol. 9, pp. 64–85, Mar. 2006.

[37] J.-L. Beuchat, N. Brisebarre, M. Shirase, T. Takagi, and E. Okamoto, "A coprocessor for the final exponentiation of the η_T pairing in characteristic three," in *Proceedings of Waifi 2007*, ser. Lecture Notes in Computer Science, C. Carlet and B. Sunar, Eds., no. 4547. Springer, 2007, pp. 25–39.

[38] R. Ronan, C. Ó hÉigeartaigh, C. Murphy, M. Scott, and T. Kerins, "Hardware acceleration of the Tate pairing on a genus 2 hyperelliptic curve," *Journal of Systems Architecture*, vol. 53, pp. 85–98, 2007.

[39] R. Ronan, C. Murphy, T. Kerins, C. Ó hÉigeartaigh, and P. S. L. M. Barreto, "A flexible processor for the characteristic 3 η_T pairing," *Int. J. High Performance Systems Architecture*, vol. 1, no. 2, pp. 79–88, 2007.

[40] J. Jiang, "Bilinear pairing (Eta_T Pairing) IP core," City University of Hong Kong – Department of Computer Science, Tech. Rep., May 2007.

[41] G. Kömürcü and E. Savas, "An efficient hardware implementation of the Tate pairing in characteristic three," in *Proceedings of the Third International Conference on Systems – ICONS 2008*, E. P.-F. rland and M. Popescu, Eds. IEEE Computer Society, 2008, pp. 23–28.

[42] A. Barenghi, G. Bertoni, L. Breveglieri, and G. Pelosi, "A FPGA coprocessor for the cryptographic Tate pairing over $\mathbb{F}_p$," in *Proceedings of the Fourth International Conference on Information Technology: New Generations (ITNG'08)*. IEEE Computer Society, 2008.

[43] M. Keller, R. Ronan, W. P. Marnane, and C. Murphy, "Hardware architectures for the Tate pairing over GF(2^m)," *Computers and Electrical Engineering*, vol. 33, no. 5–6, pp. 392–406, 2007.

[44] M. Keller, T. Kerins, F. Crowe, and W. P. Marnane, "FPGA implementation of a GF(2^m) Tate pairing architecture," in *International Workshop on Applied Reconfigurable Computing (ARC 2006)*, ser. Lecture Notes in Computer Science, K. Bertels, J. Cardoso, and S. Vassiliadis, Eds., no. 3985. Springer, 2006, pp. 358–369.

[45] H. Li, J. Huang, P. Sweany, and D. Huang, "FPGA implementations of elliptic curve cryptography and Tate pairing over a binary field," *Journal of Systems Architecture*, vol. 54, pp. 1077–1088, 2008.

[46] P. Grabher and D. Page, "Hardware acceleration of the Tate pairing in characteristic three," in *Cryptographic Hardware and Embedded Systems – CHES 2005*, ser. Lecture Notes in Computer Science, J. R. Rao and B. Sunar, Eds., no. 3659. Springer, 2005, pp. 398–411.

[47] R. Ronan, C. Ó hÉigeartaigh, C. Murphy, M. Scott, and T. Kerins, "FPGA acceleration of the Tate pairing in characteristic 2," in *Proceedings of the IEEE International Conference on Field Programmable Technology – FPT 2006*. IEEE, 2006, pp. 213–220.

[48] C. Shu, S. Kwon, and K. Gaj, "FPGA accelerated Tate pairing based cryptosystem over binary fields," in *Proceedings of the IEEE International Conference on Field Programmable Technology – FPT 2006*. IEEE, 2006, pp. 173–180.

[49] F. Hess, N. Smart, and F. Vercauteren, "The Eta pairing revisited," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4595–4602, Oct. 2006.