# Towards Faster and Greener Cryptoprocessor for Eta Pairing on Supersingular Elliptic Curve over $\mathbb{F}_{2^{1223}}$

Jithra Adikari[1], M. Anwar Hasan[2], and Christophe Negre[3,4,5]

[1] Elliptic Technologies Inc., Ottawa ON, Canada
[2] Department of Electrical and Computer Engineering, University of Waterloo, Canada
[3] Team DALI, Université de Perpignan, France
[4] LIRMM, UMR 5506, Université Montpellier 2, France
[5] LIRMM, UMR 5506, CNRS, France

**Abstract.** At the CHES workshop last year, Ghosh *et al.* presented an FPGA based cryptoprocessor, which for the first time ever makes it possible to compute an eta pairing at the 128-bit security level in less than one milli-second. The high performance of their cryptoprocessor comes largely from the use of the Karatsuba method for field multiplication. In this article, for the same type of pairing we propose hybrid sequential/parallel multipliers based on the Toeplitz matrix-vector products and present some optimizations for the final exponentiation, resulting in high performance cryptoprocessors. On the same kind of FPGA devices, our cryptoprocessor performs pairing faster than that of [12] while requiring less hardware resources. We also present ASIC implementations and report that the three-way split multiplier based cryptoprocessor consumes less energy than the two-way.

## 1 Introduction

Since 2001, cryptographic *pairing* has been used extensively to develop various security protocols, including the well known identity based encryption [3] and the short signature scheme [4]. For such protocols, pairing is by far the most computation intensive operation. A pairing algorithm typically requires thousands of additions and multiplications followed by a final exponentiation over very large finite fields. From the implementation point of view, pairing is thus very challenging; in fact it is computationally far more demanding than classical cryptographic schemes such as elliptic curve cryptography.

In this paper, we consider hardware implementation of a type of pairing known as the $\eta_T$ pairing [17] on elliptic curves defined over extended binary fields. Specifically, we focus on the 128-bit security level. During the past few years several pairing implementations for 128-bit security level have been published for various field characteristics [13,6,7,12,1,15,10,11]. Here we consider pairing over elliptic curve $E(\mathbb{F}_{2^{1223}})$, for which we also need to deal computations over $\mathbb{F}_{2^{4\cdot 1223}}$. In CHES 2011, Ghosh *et al.* [12] have proposed a cryptoprocessor architecture

L.R. Knudsen and H. Wu (Eds.): SAC 2012, LNCS 7707, pp. 166–183, 2013.

for computing such $\eta_T$ pairing at the 128-bit security level, and reported its implementation results based on field programmable gate arrays (FPGA). The cryptoprocessor has a hybrid sequential/parallel architecture for multiplication in $\mathbb{F}_{2^{1223}}$ and performs the inversion of a non-zero element of $\mathbb{F}_{2^{4\cdot 1223}}$ using linear algebra. More specifically, using the Karatsuba formula, a multiplication in $\mathbb{F}_{2^{1223}}$ is broken down into nine separate multiplications of polynomials of size 306 bits each. This allows the cryptoprocessor perform one $\mathbb{F}_{2^{1223}}$ multiplication in ten clock cycles, i.e., nine cycles are used for nine 306-bit multiplications and one cycle for the reconstruction and the reduction of the product. For Xilinx Virtex6 FPGA, the cryptoprocessor of Ghosh *et al.* [12] takes 190 $\mu$s only, making it the fastest 128-bit secure $\eta_T$ pairing unit available up until now (also see [1] for a more recent comparison).

**Our Work:** In this paper, we propose a new cryptoprocessor for the 128-bit security level $\eta_T$ pairing on the same supersingular elliptic curve used in [12]. The proposed cryptoprocessor is different than that in [12] in a number of ways, and when implemented on the same type of FPGA devices, it performs the pairing in much less time. The primary difference, which is also the main source of improvements, lies in the multiplier over $\mathbb{F}_{2^{1223}}$, which is typically the most area consuming component of such a cryptoprocessor. We use an asymptotically better, namely Toeplitz-matrix vector product (TMVP) based approach for multiplication in $\mathbb{F}_{2^{1223}}$. To the best of our knowledge, this is the first time that TMVP based multipliers are used in the implementation of pairing. The two-way split and the three-way split TMVP formulas of [8] result in multipliers which are more efficient in area and time compared to those based on the corresponding Karatsuba formulas. For example, the two-way TMVP formula enables us perform one $\mathbb{F}_{2^{1223}}$ multiplication in nine clock cycles (instead of ten cycles using the Karatsuba), and the three-way formula does it only in six clock cycles without a proportional increase in area.

In our work, we also improve the final exponentiation operation for the pairing cryptoprocessor. Typically, this exponentiation is performed via costly operations including several multiplications over $\mathbb{F}_{2^{1223}}$ along with an inversion and an exponentiation to the power of $2^{612}$ over $\mathbb{F}_{2^{4\cdot 1223}}$. We reduce the complexity of the inversion by adapting the norm based approach [5] over the tower fields $\mathbb{F}_{2^{1223}} \subset \mathbb{F}_{2^{2\cdot 1223}} \subset \mathbb{F}_{2^{4\cdot 1223}}$ We find that a square root can take a considerably fewer number of bit operations than a squaring operation in $\mathbb{F}_{2^{1223}}$, and following [2], we use this feature to reduce the computational cost of the exponentiation to the power $2^{612}$ by replacing the sequence of squaring by a sequence of square root operations.

In terms of hardware realization, we report both FPGA and application specific integrated circuit (ASIC) implementations of the proposed cryptoprocessor. To the best of our knowledge, these are the first ASIC implementations for $\eta_T$ pairing at the 128-bit security level using binary supersingular curves. Based on the ASIC results, we find that the three-way split multiplier based cryptoprocessor is a greener choice as it consumes less energy than its two-way counterpart.

**Organization:** The remainder of this paper is organized as follows: in Section 2 we briefly review $\eta_T$ pairing and analyze different operations involved in it. In Section 3 we briefly review two approaches to multiply elements of $\mathbb{F}_{2^{1223}}$ based on Toeplitz-matrix vector products and provide their respective implementation results. Then in Section 4, we present several improvements to the final exponentiation in $\eta_T$ pairing. In Section 5 we present the overall architecture for the $\eta_T$ pairing and implementation results. We compare our schemes with best known results and wind up the paper with some concluding remarks in Section 6.

## 2 Pairing Algorithm

**Pairing:** We consider supersingular elliptic curve $E$ defined by the following equation $Y^2 + Y = X^3 + X$ over the finite field $\mathbb{F}_q$, where $q = 2^{1223}$. As stated in [17], we have $\#E(\mathbb{F}_{2^{1223}}) = 5r$ where $r = (2^{1223}+2^{612}+1)/5$ is a 1221-bit prime divisor and the curve $E$ has an embedding degree $k = 4$. We construct the field $\mathbb{F}_{q^k} = \mathbb{F}_{2^{4\cdot 1223}}$ through two extensions of degree two $\mathbb{F}_{2^{2\cdot 1223}} = \mathbb{F}_{2^{1223}}[u]/(u^2 + u + 1)$ and $\mathbb{F}_{2^{4\cdot 1223}} = \mathbb{F}_{2^{2\cdot 1223}}[u]/(v^2 + v + u)$. Now if we denote $\mu_r$ the subgroup of order $r$ of $\mathbb{F}^*_{q^k} = \mathbb{F}^*_{2^{4\cdot 1223}}$ and if we pick an element $P \in E(\mathbb{F}_{2^{1223}})$ of order $r$, we can define the $\eta_T$ pairing

$$\eta_T \colon \langle P \rangle \times \langle P \rangle \longrightarrow \mu_r$$

as $\eta_T(P_1, P_2) = e(P_1, \psi(P_2))$ where $e$ is the Tate paring and $\psi(x, y) = (x + u^2, y + xu + v)$. The security level of this pairing is equal to 128 bits. In [17] the authors have proposed Algorithm 1 for the computation of the $\eta_T$ pairing.

**Algorithm 1.** $\eta_T$ pairing [17]

**Require:** $P_2 = (x_1, y_1)$ and $P_2 = (x_2, y_2) \in E(\mathbb{F}_{2^{1223}})[r]$
**Ensure:** $\eta_T(P_1, P_2)$
  $T \leftarrow x_1 + 1$
  $f \leftarrow T \cdot (x_1 + x_2 + 1) + y_1 + y_2 + (T + x_2)u + v$
  **for** $i = 1$ **to** 612 **do**
    $T \leftarrow x_1$, $x_1 \leftarrow \sqrt{x_1}$, $y_1 \leftarrow \sqrt{y_1}$
    $g \leftarrow T \cdot (x_1 + x_2) + y_1 + y_2 + x_1 + 1 + (T + x_2)u + v$
    $f \leftarrow f \cdot g$
    $x_2 \leftarrow x_2^2$, $y_2 \leftarrow y_2^2$
  **end for**
  **return**$(f^{(2^{2\cdot 1223}-1)(2^{1223}-2^{612}+1)})$

The **for** loop in Algorithm 1 is a re-expression of the Miller's loop of the Tate pairing for the special curve $E$ and the $\eta_T$ pairing considered here. We remark that the main operations performed in Algorithm 1 are two square roots in $\mathbb{F}_{2^{1223}}$ in the first step of the **for** loop, one multiplication $T \cdot (x_1 + x_2)$ in $\mathbb{F}_{2^{1223}}$ plus several additions for the computation of $g$, one special multiplication

$f \cdot g$ in $\mathbb{F}_{2^{4\cdot 1223}}$ for the computation of $f$, two squarings in $\mathbb{F}_{2^{1223}}$ and the final exponentiation $f^{(2^{2\cdot 1223}-1)(2^{1223}-2^{612}+1)}$ in $\mathbb{F}_{2^{4\cdot 1223}}$.

Addition of two elements of a binary field corresponds to bit-wise XOR operations and, for field $\mathbb{F}_{2^{1223}}$, the bit-parallel implementation of an adder requires 1223 two-input XOR gates. Below we briefly describe squaring and square root operations for elements of $\mathbb{F}_{2^{1223}}$. The other operations are discussed in subsequent sections.

**Square and Square Root:** The field $\mathbb{F}_{2^{1223}}$ is constructed as $\mathbb{F}_{2^{1223}} = \mathbb{F}_2[x]/(x^{1223} + x^{255} + 1)$. The squaring of $A = \sum_{i=0}^{1223} a_i x^i$ in $\mathbb{F}_{2^{1223}}$ can, in this situation, be performed as follows

$$\begin{aligned} A^2 &= \textstyle\sum_{i=0}^{1222} a_i x^{2i} \mod (x^{1223} + x^{255} + 1) \\ &= \textstyle\sum_{i=0}^{127} a_i x^{2i} + \sum_{i=128}^{254} (a_i + a_{i+612-128} + a_{i+1224-256}) x^{2i} \\ &\quad + \textstyle\sum_{i=255}^{611} (a_i + a_{i+612-128}) x^{2i} + \sum_{i=0}^{126} (a_{i+612} + a_{i+1224-128}) x^{2i+1} \\ &\quad + \textstyle\sum_{i=127}^{610} a_{i+612} x^{2i+1}. \end{aligned}$$

Based on the aforementioned expression, the squaring in $\mathbb{F}_{2^{1223}}$ can be implemented with 738 XOR gates and a delay of $2D_X$.

The square root of an element $A = \sum_{i=0}^{1223} a_i x^i$ in $\mathbb{F}_{2^{1223}}$ can be expressed as

$$\begin{aligned} \sqrt{A} &= \sqrt{\textstyle\sum_{i=0}^{611} a_{2i} x^{2i}} + \sqrt{\textstyle\sum_{i=0}^{610} a_{2i+1} x^{2i+1}} \\ &= \left(\textstyle\sum_{i=0}^{611} a_{2i} x^i\right) + \sqrt{x} \left(\textstyle\sum_{i=0}^{610} a_{2i+1} x^i\right). \end{aligned}$$

Since $x = x^{256} + x^{1224} \mod (1 + x^{255} + x^{1223})$, we have $\sqrt{x} = x^{128} + x^{612} \mod (1 + x^{255} + x^{1223})$, and, after replacing $\sqrt{x}$ with $x^{256} + x^{1224}$, we obtain that $\sqrt{A}$ can be computed as

$$\begin{aligned} \sqrt{A} = &\textstyle\sum_{i=0}^{127} a_{2i} x^i + \sum_{i=128}^{611} (a_{2i} + a_{2i-256+1}) x^i \\ &+ \textstyle\sum_{i=0}^{126} (a_{2i+1} + a_{2(i+612-128)+1}) x^{i+612} + \sum_{i=127}^{610} a_{2i+1} x^{i+612}. \end{aligned}$$

Hence a square root can be implemented with 611 XOR gates and a delay of $D_X$. Consequently, the number of bit operations is lower in a square root than squaring in $\mathbb{F}_{2^{1223}}$ defined by $x^{1223} + x^{255} + 1$. We take advantage of this feature in the final exponentiation of pairing.

Unlike squaring and square root operations, the various multiplications appearing in the $\eta_T$ paring algorithm are more difficult to implement. In the next section we present two multiplier architectures used in our proposal for $\eta_T$ pairing cryptoprocessor.

## 3 Multiplier Architectures

In the $\eta_T$ pairing algorithm, the main operations include multiplications with inputs in one of the three fields $\mathbb{F}_{2^{1223}} = \mathbb{F}_2[x]/(x^{1223} + x^{255} + 1)$, $\mathbb{F}_{2^{2\cdot 1223}} = \mathbb{F}_{2^{1223}}[u]/(u^2 + u + 1)$ and $\mathbb{F}_{2^{4\cdot 1223}} = \mathbb{F}_{2^{2\cdot 1223}}[v]/(v^2 + v + u)$. The authors in [12]

have designed a multiplier architecture for $\mathbb{F}_{2^{1223}}$ and perform multiplications in the extended fields $\mathbb{F}_{2^{2\cdot 1223}}$ and $\mathbb{F}_{2^{4\cdot 1223}}$ through a sequence of multiplications in $\mathbb{F}_{2^{1223}}$. This method for the multiplication over $\mathbb{F}_{2^{2\cdot 1223}}$ and $\mathbb{F}_{2^{4\cdot 1223}}$ is reviewed later in Subsection 3.3. For the multiplication in $\mathbb{F}_{2^{1223}}$ the authors in [12] use a hybrid sequential and parallel recursion of the Karatsuba formula for polynomial multiplication. In the next subsection we investigate an alternative approach, which is based on the formulation of multiplication over $\mathbb{F}_{2^{1223}}$ as Toeplitz matrix vector products.

### 3.1 Multiplication in $\mathbb{F}_{2^{1223}}$ through Toeplitz Matrix Vector Products

The field $\mathbb{F}_{2^{1223}}$ is the set of binary polynomials of degree $< 1223$ modulo the irreducible trinomial $x^{1223}+x^{255}+1$. For two given elements $A = \sum_{i=0}^{1222} a_i x^i$ and $B = \sum_{i=0}^{1222} b_i x^i$, the product of $A$ and $B$ can be computed by first performing a polynomial multiplication and then reduce it modulo $x^{1223} + x^{255} + 1$. As stated in [16], this multiplication and reduction can be re-expressed as follows

$$\begin{aligned} A \times B \mod (x^{1223} + x^{255} + 1) &= A \times (\textstyle\sum_{i=0}^{1222} b_i x^i) \mod (x^{1223} + x^{255} + 1) \\ &= \textstyle\sum_{i=0}^{1222} A^{(i)} b_i \end{aligned}$$

where $A^{(i)} = (x^i \times A) \mod (x^{1223} + x^{255} + 1)$. This means that the product in $\mathbb{F}_{2^{1223}}$ can be seen as a matrix-vector product $M_A \times B$ where $M_A = \left[A^{(0)}\ A^{(1)}\ \cdots\ A^{(1222)}\right]$. We can further arrange this matrix-vector product. Indeed if we define the following $1223 \times 1223$ circulant matrix

$$U = \begin{bmatrix} 0 & I_{968\times 968} \\ I_{255\times 255} & 0 \end{bmatrix},$$

then matrix $T_A = U \cdot M_A$ is obtained by removing the top 255 rows and placing them below the other 968 rows of $M_A$. The resulting matrix $T_A$ has the following Toeplitz structure:

$$\begin{bmatrix} a_{255} & a_{254}+a_{1222} & \cdots & a_0+a_{968} & a_{1222}+a_{967} & \cdots & a_{510}+a_{255} & \cdots & a_{257}+a_2+a_{970} & a_{256}+a_1+a_{969} \\ a_{256} & a_{255} & \cdots & a_1+a_{969} & a_0+a_{968} & \cdots & a_{511}+a_{256} & \cdots & a_{258}+a_3+a_{971} & a_{257}+a_2+a_{970} \\ \vdots & & & & & & & & & \vdots \\ a_{1221} & a_{1220} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{1222}+a_{967} \\ a_{1222} & a_{1221} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_0+a_{968} \\ a_0 & a_{1222} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_1+a_{969} \\ \vdots & & & & & & & & & \vdots \\ a_{253} & a_{252} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{254}+a_{1222} \\ a_{254} & a_{253} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{255} \end{bmatrix}.$$

The product $C = A \times B \mod (x^{1223} + x^{255} + 1)$ is obtained by first performing this Toeplitz matrix vector product $C' = T_A \cdot B$ and then by switching the top 968 coefficients of $C'$ with its other 255 coefficients.

We take advantage of the Toeplitz matrix vector product (TMVP) expression of a multiplication in $\mathbb{F}_{2^{1223}}$ since a TMVP can be performed using a subquadratic

method [19,8]. This subquadratic method is obtained by recursively applying two-way or three-way split formulas. Assuming that $T$ is an $n \times n$ Toeplitz matrix and $V$ is a column vector with $n$ rows, the two-way split formula reduces a TMVP of size $n$ to three TMVPs of size $n/2$ each as follows:

$$T \cdot V = \begin{bmatrix} T_1 \; T_0 \\ T_2 \; T_1 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} = \begin{bmatrix} P_0 + P_1 \\ P_2 + P_1 \end{bmatrix}, \text{ where } \begin{cases} P_0 = (T_0 + T_1) \cdot V_1, \\ P_1 = T_1 \cdot (V_0 + V_1), \\ P_2 = (T_1 + T_2) \cdot V_0. \end{cases} \quad (1)$$

In [8] we can also find a three-way split formula which reduces a TMVP of size $n$ to six TMVPs of size $n/3$ as follows:

$$T \cdot V = \begin{bmatrix} T_2 \; T_1 \; T_0 \\ T_3 \; T_2 \; T_1 \\ T_4 \; T_3 \; T_2 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} P_0 + P_3 + P_4 \\ P_1 + P_3 + P_5 \\ P_2 + P_4 + P_5 \end{bmatrix}, \text{ where } \begin{cases} P_0 = (T_0 + T_1 + T_2) \cdot V_2, \\ P_1 = (T_1 + T_2 + T_3) \cdot V_1, \\ P_2 = (T_2 + T_3 + T_4) \cdot V_0, \\ P_3 = T_1 \cdot (V_1 + V_2), \\ P_4 = T_2 \cdot (V_0 + V_2), \\ P_5 = T_3 \cdot (V_0 + V_1). \end{cases} \quad (2)$$

When applied recursively, the above formulas provide a multiplication with subquadratic complexity.The complexities of the two- and three-way split formulas are reported in Table 1. For details on the evaluation of these complexities the reader may refer to [8]. For comparison purposes, we also provide the complexities of the two-way and the three-way polynomial approaches presented in [18] and [9]. The complexities in Table 1 show that, in each type of splits, TMVP approaches outperform polynomial approaches and are thus more suitable to design finite field multipliers.

**Table 1.** Area and time complexities of two-way and three-way split polynomial multiplication and TMVP

| Formula type | Method | #AND | #XOR | Delay |
|---|---|---|---|---|
| Two-way | Poly. mult. with Karatsuba ([18]) | $n^{\log_2(3)}$ | $6n^{\log_2(3)} - 8n + 2$ | $D_A + 3\log_2(n)D_X$ |
| | Poly. mult. with [9] | $n^{\log_2(3)}$ | $6n^{\log_2(3)} - 8n + 2$ | $D_A + 2\log_2(n)D_X$ |
| | TMVP with [8] | $n^{\log_2(3)}$ | $5.5n^{\log_2(3)} - 7n + 1.5$ | $D_A + 2\log_2(n)D_X$ |
| Three-way | Poly. mult. with [18] | $n^{\log_3(6)}$ | $5.33n^{\log_3(6)} - 7.33n + 2$ | $D_A + 4\log_3(n)D_X$ |
| | Poly. mult. with [9] | $n^{\log_3(6)}$ | $5.33n^{\log_3(6)} - 7.33n + 2$ | $D_A + 3\log_3(n)D_X$ |
| | TMVP with [8] | $n^{\log_3(6)}$ | $4.8n^{\log_3(6)} - 5n + 0.2$ | $D_A + 3\log_3(n)D_X$ |

### 3.2 Hybrid Sequential/Parallel TMVP Multiplier

In order to use the TMVP formulas (1) and (2) in our design of a multiplier for the field $\mathbb{F}_{2^{1223}}$, we begin with two important issues. First, in order to apply the TMVP formula, the size of the considered TMVP must be divisible by 2 or 3. The solution we adopt here is to extend the matrices (by extending the diagonal)

and the vectors (by padding 0) up to a size divisible by 2 or 3. For example for $n = 1223$ we can extend it to 1224, which is divisible by 2 and 3. If needed, we can repeat this strategy during the recursion.

The second and more challenging issue is that a fully parallel multiplier for the field $\mathbb{F}_{2^{1223}}$ is too large to fit in common FPGA devices. To this end, we adopt the approach used in [12] in the case of polynomial multiplication with the Karatsuba formula. Specifically, instead of performing all the computations in parallel, we process the recursion of the TMVP formulas through a hybrid sequential/parallel process. Below we describe this approach by applying *one* recursion of the three-way split formula. We have done also an hybrid sequential/parallel multiplier based on *two* recursions of the two-way split TMVP formula (1). Because of lack of space, we cannot present this case in details in this paper. The reader may refer to the forthcomming technical report extending the current paper for these details.

**Three-Way Hybrid Sequential/parallel TMVP Multiplier for $n = 1224$.** The first hybrid sequential/parallel TMVP multiplier for $n = 1224$ includes hardware for splitting the associated Toeplitz matrices and vectors of size 1224 in three ways. The splitting leads to six TMVP instances of size 408 each. One fully parallel hardware unit performs these six TMVPs one after another. Some additional hardware is used to temporarily store and combine the resulting 408-bit outputs into a 1224-bit product.

The overall architecture of this multiplier is depicted in Fig. 1 and a brief operational description follows.

Referring to Fig. 1, the entries of the Toeplitz matrix and the vector are stored it two registers, namely $T_{in}$ and $R_{in}$. We note that the Toeplitz matrix is completely defined by its top row and left most column. Following (2), the entries are split as follows:

$$T \cdot V = \begin{bmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix}$$

where matrices $T_i, i = 0, 1, \ldots, 4$ and vectors $V_i, i = 0, 1, 2$ are of size 408. The *CONVERTOR* block computes, through from the 1223 bit stored in $T_{in}$ the coefficients of the first column and the first row of $T$. Then the $T_{net}$ block of Fig. 1 sequentially generates the six matrices

$$(T_2 + T_1 + T_0),\ (T_3 + T_2 + T_1),\ (T_4 + T_3 + T_2),\ T_1,\ T_2,\ T_3,$$

involved in the six products $P_i, i = 0, \ldots, 5$ of (2). In parallel to the formation of the matrices, the $V_{net}$ block of Fig. 1 generates the following six corresponding vectors

$$V_2,\ V_1,\ V_0,\ (V_1 + V_2),\ (V_0 + V_2),\ (V_0 + V_1).$$

The (matrix and vector) outputs of $T_{net}$ and $V_{net}$ are input to a fully parallel unit which computes TMVP of size 408 each. The parallel unit consists of three

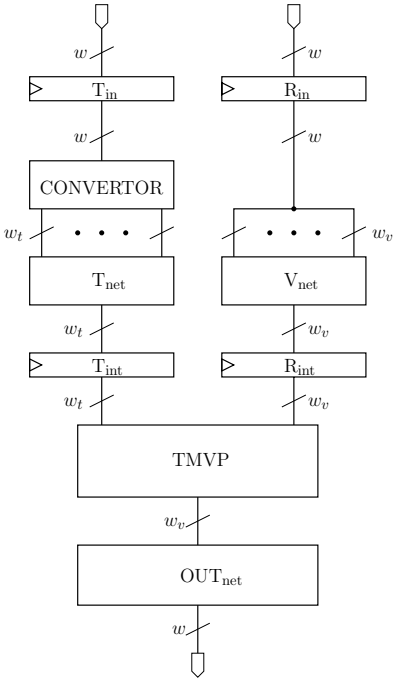



**Fig. 1.** Multiplier architecture

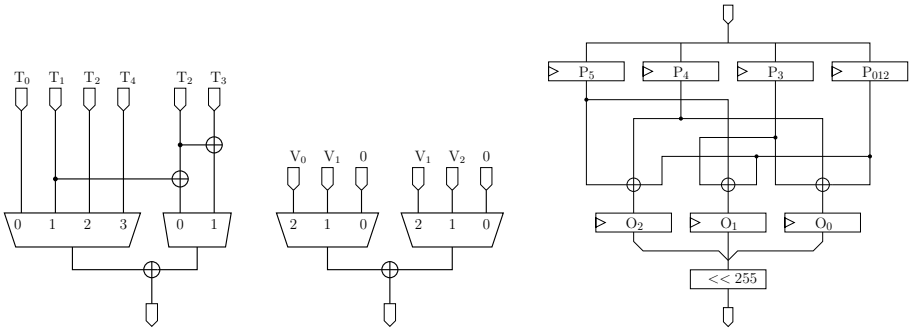
recursions of three-way split TMVP formula and one two-way split TMVP formula: $408 \rightarrow 136 \rightarrow 46 \rightarrow 16 \rightarrow 8$, the TMVPs of size 8 are performed with a quadratic method.

This parallel unit sequentially outputs the six products $P_0, P_1, \ldots, P_5$ defined in (2). These products $P_i, i = 0, \ldots, 5$, are accumulated in the block labeled as $OUT_{net}$ of Fig. 1 to form the result $W = T \cdot V$. Some additional details of blocks $T_{net}$, $V_{net}$ and $OUT_{net}$ are depicted in Fig. 2.

**Implementation Results of the Multiplier.** We have implemented in FPGA and ASIC the two-way and three-way hybrid sequential/parallel multipliers presented in the previous subsections. Our main goal has been to optimize the speed. Below we present the FPGA implementations; for lack of space we omit the ASIC implementations for the multipliers, but we do report the ASIC implementations of the complete cryptoprocessor later in the paper.

The multiplier designs have been placed and routed on Xilinx `xc6vlx365t-3` FPGA by executing Xilinx Integrated Software Environment (ISE$^{\text{TM}}$) version `12.4`. The synthesis tool for FPGA estimated LUT counts and operating frequencies for our three- and two-way hybrid sequential/parallel multipliers. In Table 2 we have reported these results along with the results of Ghosh *et al.* [12].

Table 2 shows that the proposed two-way hybrid sequential/parallel multiplier is better than the multiplier presented in [12], considering both LUTs and




(a) $T_{net}$ (b) $V_{net}$ (c) $OUT_{net}$

**Fig. 2.** Blocks of the hybrid three-way sequential/parallel multiplier

**Table 2.** FPGA Place & Route implementation synthesis results and comparisons for $\mathbb{F}_{2^{1223}}$ multipliers

| Multiplier architecture | LUTs | Freq. (MHz) | # Clock Cycles per mult | Latency (ns) | Area × time (LUTs) (ms) |
|---|---|---|---|---|---|
| Sequential use of 306 bit parallel Karatsuba Mult [12] | 30,148 | 250 | 10 | 40.0 | 1.21 |
| Two-way 306-bit TMVP Mult | 19,721 | 271 | 9 | 33.2 | 0.65 |
| Three-way 408-bit TMVP Mult (Subsec. 3.2) | 33,546 | 267 | 6 | 22.5 | 0.75 |

frequency. The improvement in the number of LUT is mainly due to the use of TMVP approach for finite field multiplication. The reduction of the delay is partly explained by the use of the TMVP approach, but is also due to the reduced number of clock cycle (9 instead of 10) needed for a multiplication. The proposed three-way hybrid sequential/parallel multiplier present an alternative to the proposed two-way multiplier and the multiplier of [12]. Specifically, it has the largest number of LUTs, but offers the highest frequency and smallest serial use number (i.e., 6 vs. 9 and 10). This results in a multiplier which is less area efficient but faster than the proposed two-way multiplier.

### 3.3 Multiplication in Fields $\mathbb{F}_{2^{2\cdot 1223}}$ and $\mathbb{F}_{2^{4\cdot 1223}}$

In this section we review the method used to multiply two elements in $\mathbb{F}_{2^{2\cdot 1223}}$ and to multiply two elements in $\mathbb{F}_{2^{4\cdot 1223}}$. This method uses one recursion (resp. two recursions) of Karatsuba to reduce the multiplication in $\mathbb{F}_{2^{2\cdot 1223}}$ (resp. $\mathbb{F}_{2^{4\cdot 1223}}$) to several multiplications in $\mathbb{F}_{2^{1223}}$. These multiplications are performed through one of the two multipliers presented in the previous subsection.

*Multiplication in* $\mathbb{F}_{2^{2\cdot 1223}}$*:* The elements of $\mathbb{F}_{2^{2\cdot 1223}} = \mathbb{F}_{2^{1223}}[u]/(u^2+u+1)$ are considered as degree one polynomial in $u$. We can thus reduce one multiplication in $\mathbb{F}_{2^{2\cdot 1223}}$ to three multiplications in $\mathbb{F}_{2^{1223}}$ using the Karatsuba formula. Indeed, let $A = A_0 + A_1 u$ and $B = B_0 + B_1 u$ be two elements of $\mathbb{F}_{2^{2\cdot 1223}}$. We perform these three products $P_0 = A_0 B_0$, $P_1 = (A_0 + A_1)(B_0 + B_1)$ and $P_2 = A_1 B_1$. We then reconstruct and reduce modulo $u^2 + u + 1$ the product $C = A \times B$ as follows

$$\begin{aligned} C &= P_0 + (P_1 + P_0 + P_2)u + P_2 u^2 \\ &= P_0 + P_2 + (P_1 + P_0)u \mod (u^2 + u + 1) \end{aligned}$$

The cost of this method is equal to 3 multiplications and 4 additions in the field $\mathbb{F}_{2^{1223}}$.

*Multiplication in* $\mathbb{F}_{2^{4\cdot 1223}}$. The elements of $\mathbb{F}_{2^{4\cdot 1223}} = \mathbb{F}_{2^{2\cdot 1223}}[v]/(v^2+v+u)$ are considered as degree one polynomial in $v$. We reduce one multiplication in $\mathbb{F}_{2^{4\cdot 1223}}$ to three multiplications in $\mathbb{F}_{2^{2\cdot 1223}}$ by applying the Karatsuba formula. Indeed, let $A = (A_0 + A_1 u) + (A_2 + A_3 u)v$ and $B = B_0 + B_1 u + (B_2 + B_3 u)v$ be two elements of $\mathbb{F}_{2^{4\cdot 1223}}$. We first perform three products $P_0 + P_1 u = (A_0 + A_1 u)(B_0 + B_1 u), P_2 + P_3 u = (A_0 + A_2 + (A_1 + A_3)u)(B_0 + B_2 + (B_1 + B_3)u)$ and $P_4 + P_5 u = (A_2 + A_3 u)(B_2 + B_3 u)$ in $\mathbb{F}_{2^{2\cdot 1223}}$ and then reconstruct $C = A \times B$ modulo $v^2 + v + u$ as follows

$$\begin{aligned} C &= (P_0 + P_1 u) + (P_2 + P_3 u + P_0 + P_1 u + P_4 + P_5 u)v + (P_4 + P_5 u)v^2 \\ &= (P_0 + P_5 + (P_1 + P_4 + P_5)u) \\ &\quad + (P_2 + P_0 + (P_3 + P_1)u)v \mod (v^2 + v + u, u^2 + u + 1). \end{aligned}$$

The cost of this approach is equal to 3 multiplications in $\mathbb{F}_{2^{2\cdot 1223}}$ plus 9 additions in $\mathbb{F}_{2^{1223}}$. Using the cost of one multiplication in $\mathbb{F}_{2^{2\cdot 1223}}$ previously evaluated, we obtain a cost of 9 multiplications and 21 additions in $\mathbb{F}_{2^{1223}}$.

*Cost of* $f \cdot g$ *in Algorithm 1:* The most costly operation in the Miller's loop of Algorithm 1 is the multiplication $f \cdot g$. Thanks to the special form of $g = g_0 + g_1 u + v$, this multiplication can be reduced to two multiplications in $\mathbb{F}_{2^{2\cdot 1223}}$ plus a few additions. Indeed, if we write $f = f_0 + f_1 u + f_2 v + f_3 uv$, we have the following:

$$\begin{aligned} fg &= (f_0 + f_1 u)(g_0 + g_1 u) + (f_2 + f_3 u)(g_0 + g_1 u)v + (f_0 + f_1 u + f_2 v + f_3 uv)v \\ &= ((f_0 + f_1 u)(g_0 + g_1 u) + f_2 + f_3 + f_3 u) \\ &\quad + ((f_2 + f_3 u)(g_0 + g_1 u) + f_0 + f_3 + (f_1 + f_3)u)\, v \end{aligned}$$

This expression requires two multiplications, namely $(f_0 + f_1 u)(g_0 + g_1 u)$ and $(f_2 + f_3 u)(g_0 + g_1 u)$ in $\mathbb{F}_{2^{2\cdot 1223}}$, plus the additions of the resulting terms to $f_2 + f_3 + f_3 u$ and $f_0 + f_3 + (f_1 + f_3)u$. Consequently, the cost of $f \cdot g$ in Algorithm 1 is 6 multiplications and 15 additions in $\mathbb{F}_{2^{1223}}$.

## 4 Final Exponentiation

In this section, we focus on the final operation of the $\eta_T$ pairing (Algorithm 1). This operation is to compute $(f^{(2^{2\cdot 1223}-1)(2^{1223}-2^{612}+1)})$ for a given $f \in \mathbb{F}_{2^{4\cdot 1223}}$.

We begin with the strategy presented in [12]: we first raise $f$ to the power $2^{2\cdot 1223}-1$ and then raise $f'=f^{(2^{2\cdot 1223}-1)}$ to the power $(2^{1223}-2^{612}+1)$. This method is restated in Algorithm 2.

**Algorithm 2.** Final exponentiation

**Require:** $f \in \mathbb{F}_{2^{4\cdot 1223}}$
**Ensure:** $f^{(2^{2\cdot 1223}-1)(2^{1223}-2^{612}+1)}$
Step 1. $S \leftarrow f^{2^{2\cdot 1223}}$
Step 2. $T \leftarrow f^{-1}$
Step 3. $S \leftarrow S \times T$ // *S is equal to* $f^{(2^{2^{2\cdot 1223}}-1)}$
Step 4. $T \leftarrow S^{2^{2\cdot 1223}}$ // *T is equal to* $(f^{2^{2^{2\cdot 1223}}-1})^{2^{2\cdot 1223}} = f^{1-2^{2^{2\cdot 1223}}}$
Step 5. $T \leftarrow T^{2^{612}}$ // *T is then equal to* $f^{(2^{2^{2\cdot 1223}}-1)(-2^{612})}$
Step 6. $R \leftarrow S^{2^{1223}}$ // *R is equal to* $f^{(2^{2^{2\cdot 1223}}-1)(2^{1223})}$
Step 7. $R \leftarrow R\cdot T\cdot S$ // *R is finally equal to* $f^{(2^{2^{2\cdot 1223}}-1)(2^{1223}-2^{612}+1)}$
Step 8. **return**(R)

Note that in Step 4, we have used the fact that $f^{2^{4\cdot 1223}}=f$, since $f\in\mathbb{F}_{2^{4\cdot 1223}}$, to derive the expression $f^{1-2^{2^{2\cdot 1223}}}$ of $T$. In Algorithm 2 a number of operation are performed including

- Powering to some 2 power exponents, like the power $2^{612}$ in Step 5 and the powers $2^{1223}$ and $2^{2\cdot 1223}$ in Step 1, Step 4 and Step 6.
- Multiplication in the field $\mathbb{F}_{2^{4\cdot 1223}}$ in Step 3 and Step 7.
- Inversion in $\mathbb{F}_{2^{4\cdot 1223}}$ in Step 2.

We now specify how we perform the above operations. A multiplication in $\mathbb{F}_{2^{4\cdot 1223}}$ is performed using the method given in Subsection 3.3. For the powering to $2^{1223}$ and $2^{2\cdot 1223}$, we use the formula stated in the following lemma.

**Lemma 1.** *Let* $f=f_0+f_1u+f_2v+f_3uv\in\mathbb{F}_{4\cdot 1223}$ *, then the following identities hold*

*(i)* $u^4=u$ *and* $v^{16}=v$.
*(ii)* $f^{2^{1223}}=(f_0+f_1+f_2)+(f_1+f_2+f_3)u+(f_2+f_3)v+f_3uv$.
*(iii)* $f^{2^{2\cdot 1223}}=(f_0+f_2)+(f_1+f_3)u+f_2v+f_3uv$.

The proof is not difficult: *i*) is the direct consequence of the definition of $u$ and $v$, and *ii*) and *iii*) are consequences of the little Fermat theorem and of *i*). Due to lack of space we omit the proof of this lemma.

For the other operations, i.e., inversion in $\mathbb{F}_{2^{4\cdot 1223}}$ and exponentiation to the power $2^{612}$, we propose some new optimizations. These optimizations are given in the following two subsections.

### 4.1 Inversion in $\mathbb{F}_{2^{4\cdot 1223}}$

To perform the inversion in $\mathbb{F}_{2^{4\cdot 1223}}$ we use an approach similar to the one used for some implementations of AES Sbox [5]. The proposed inversion is based on the following well known expression $A^{-1} = A^{r-1} \times (A^r)^{-1}$, which can be used to compute an inverse in an extension of degree two $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$ by taking $r = q$. This reduces the computation of an inversion in $\mathbb{F}_{q^2}$ to an inversion in $\mathbb{F}_q$ as follows

$$A^{-1} = A^q (A^{1+q})^{-1},$$

since $A^{1+q}$ is in $\mathbb{F}_q$. As $A^{1+q}$ is a *norm* relative to the field extension $\mathbb{F}_{q^2}$ over $\mathbb{F}_q$, this approach is often referred to as the *norm approach* for field inversion. In our situation we apply this approach twice: first to reduce the inversion in $\mathbb{F}_{2^{4\cdot 1223}}$ to an inversion in $\mathbb{F}_{2^{2\cdot 1223}}$ with $q = 2^{2\cdot 1223}$ and then to reduce the latter inversion in $\mathbb{F}_{2^{2\cdot 1223}}$ to an inversion in $\mathbb{F}_{2^{1223}}$ with $q' = 2^{1223}$. This method is detailed in Algorithm 3.

**Algorithm 3.** Inversion in $\mathbb{F}_{2^{4\cdot 1223}}$

**Require:** $A = A_0 + A_1 u + A_2 v + A_3 uv$
**Ensure:** $A^{-1}$
Step 1. $R_0 + R_1 u + R_2 v + R_3 uv \leftarrow (A_0 + A_2) + (A_1 + A_3)u + A_2 v + A_3 uv$ // $R = A^{2^{2\cdot 1223}}$
Step 2. $S_0 + uS_1 \leftarrow (A_1 + uA_u)(R_1 + uR_u) + u(R_2 + uR_3)^2$ // $S = A \times A^{2^{2\cdot 1223}}$
Step 3. $T_0 + T_1 u \leftarrow S_0 + S_1 + S_1 u$ // $T = S^{2^{1223}}$
Step 4. $U \leftarrow S_0 T_0 + S_1 T_1$ // $U = S \times S^{2^{1223}}$
Step 5. $V \leftarrow Inv_{\mathbb{F}_{2^{1223}}}(U)$ // $V$ *the inverse of* $U$
Step 6. $W_0 \leftarrow T_0 V + T_1 V u$ // $W = T \times V$
Step 7. $(Z_0 + Z_1 u) \leftarrow (R_0 + uR_1)(W_0 + W_1 u),\ (Z_2 + uZ_3) \leftarrow (R_2 + R_3 u)(W_0 + W_1 u)$ // $Z = R \times W$
**return**$(Z = Z_0 + Z_1 u + Z_2 v + Z_3 uv)$

We now evaluate the cost of Algorithm 3 in terms of the operations in $\mathbb{F}_{2^{1223}}$. Specifically, we count the number of additions ($Add$), squarings ($Squ$), multiplications ($Mul$) and inversion ($Inv$) in $\mathbb{F}_{2^{1223}}$. It is easy to see that Step 1 requires $2Add$ and Step 3 only $1Add$. Step 4 requires $1Mul$, $1Squ$ plus one $Add$ (we have a squaring since $S_1 = T_1$), Step 5 requires one $Inv$ and Step 6 requires $2Mul$. Finally, Step 2 and Step 7 contribute three multiplications in $\mathbb{F}_{2^{2\cdot 1223}}$, which leads to $9Mul + 9Add$ in $\mathbb{F}_{2^{1223}}$. The terms $u(R_2 + uR_3)^2 = R_2^2 u + R_3^2 \mod (u^2 + u + 1)$ in Step 2 contributes to $2Squ + 2Add$. We finally obtain the overall cost of Algorithm 3: $14Add + 3Squ + 10Mul + 1Inv$.

In our proposed architecture, the additions and the squaring operations are performed through dedicated fully parallel adder and squarer. The multiplications are done through one of the $\mathbb{F}_{2^{1223}}$ multipliers presented in Section 3. We perform the inversion in $\mathbb{F}_{2^{1223}}$ using the algorithm of Itoh-Tsujii [14]. This method expresses the inversion as a sequence of squarings and multiplications specified by an addition chain. There exist several addition chains suitable to

**Table 3.** Complexity of the proposed approach for the final exponentiation

| | $\#Add$ | $\#Squ/SquRoot$ | $\#Mult$ |
|---|---|---|---|
| Step 1. | 2 | 0 | 0 |
| Step 2. | 14 | 1,225 | 24 |
| Step 3. | 21 | 0 | 9 |
| Step 4. | 2 | 0 | 0 |
| Step 5. | 2 | 2,444 | 0 |
| Step 6. | 6 | 0 | 0 |
| Step 7. | 42 | 0 | 18 |
| Total | 89 | 3669 | 51 |

perform an inversion efficiently. The addition chain we use to compute the inverse of $a$ is as follows

$$s \leftarrow a, s \leftarrow s^2 \cdot s,\ r \leftarrow s,\ s \leftarrow s^{(2^2)} \cdot s, s \leftarrow s^{(2^2)}, r \leftarrow r \cdot s, s \leftarrow s^{(2^4)} \cdot s,$$
$$s \leftarrow s^{(2^8)} \cdot s, s \leftarrow s^{(2^{16})} \cdot s, s \leftarrow s^{(2^{32})} \cdot s, s \leftarrow s^{(2^4)}, r \leftarrow r \cdot s, s \leftarrow s^{(2^{64})} \cdot s,$$
$$s \leftarrow s^{(2^{64})}, r \leftarrow r \cdot s, s \leftarrow s^{(2^{128})} \cdot s, s \leftarrow s^{(2^{256})} \cdot s, s \leftarrow s^{(2^{512})} \cdot s, s \leftarrow s^{(2^{128})},$$
$$r \leftarrow r \cdot s, r \leftarrow r^2,$$

and the last $r$ satisfies $r = a^{-1}$. The resulting complexity of the inversion is equal to $14Mul$ and $1222Squ$ in $\mathbb{F}_{2^{1223}}$.

This means that the total cost in terms of additions, squarings and multiplications of the proposed inversion in $\mathbb{F}_{2^{4\cdot 1223}}$ is

$$14Add + 1225Squ + 24Mul.$$

In [12], the same inversion operation is performed by solving a system of equations and it incurs a cost of $57Add + 1230Squ + 50Mul$.

### 4.2 Complexity Evaluation and Comparison of the Final Exponentiation

Using the following expression given [2] of the exponentiation to the power $2^{612}$ in $\mathbb{F}_{2^{4\cdot 1223}}$ $(f_0+f_1u+f_2v+f_3uv)^{2^{612}} = (f_0^{2^{-611}})+(f_1^{2^{-611}})u+(f_2^{2^{-611}})+(f_3^{2^{-611}})uv$. this exponentiation is reduced to four independent sequences of 611 square roots, and these four sequences can be performed in parallel. Based on this and the cost of inversion in $\mathbb{F}_{2^{4\cdot 1223}}$ from the previous subsection, in Table 3 we list the cost of each step of Algorithm 2 using the complexity results stated in Subsection 3.3, Lemma 1 and Subsection 4.1. The cost of each step is then added to obtain the overall cost of the proposed approach for the final exponentiation. We now compare the proposed approach with that of [12]. To this end, first we correct a small error in [12] which reports the 612 squarings for Step 5 as squarings in $\mathbb{F}_{2^{1223}}$ instead of squarings in $\mathbb{F}_{2^{4\cdot 1223}}$. Since a squaring in $\mathbb{F}_{2^{4\cdot 1223}}$ has a cost of $4Squ + 4Add$ in $\mathbb{F}_{2^{1223}}$, the actual complexity of the method of [12] is $3083Add + 3872Squ + 98Mul$. We then remark that the proposed approach reduces the number of additions, squarings and multiplications compared to that of [12].

## 5 Proposed Cryptoprocessor for $\eta_T$ Pairing and Implementation Results

The final architecture for the $\eta_T$ pairing is depicted in Fig. 3. There are three main blocks in our pairing-based cryptoprocessor, namely binary arithmetic unit (BAU), input and squaring unit (ISU) and data handling unit (DHU). The different computations involved in Algorithm 1, i.e., the operations of the Miller's loop and the operations of the final exponentiation, are distributed in the three blocks BAU, ISU and DHU. Specifically, BAU has five 1223-bit registers, one binary field multiplier, one adder and two squaring units. Both squaring units are connected in series to compute two squarings ($X^{2^2}$) in a single clock cycle during the computation of an inversion in $\mathbb{F}_{2^{1223}}$. Any of the registers $R_0$, $R_1$, $R_2$, $R_3$ and $R_X$ can be added through the field adder by setting $R+1$ signal to high. $R_2$ is set to one at the beginning of computation as required by Algorithm 1. Note that our multiplier operates in steady state when the Miller's loop is computed. Hence the multiplication costs are only six and nine clock cycles in three-way split and two-way split multiplier structures, respectively. The ISU block has five 1223-bit registers to store the intermediate values $x_1$, $x_2$, $y_1$, $y_2$ and $T$ values during the Miller's loop computation (cf. Algorithm 1). In final exponentiation, the computation $T^{2^{612}} \in \mathbb{F}_{2^{4\cdot 1223}}$ is performed through four sequences of 611 square roots in the ISU block. When extended field multiplication is performed in Miller's loop and final exponentiation, intermediate values are stored in eight 1223-bit registers in DHU. Powering to the exponent $2^{1223}$ during the final exponentiation are also computed in DHU. Furthermore, DHU handles transferring data from BAU to ISU and vice versa. There are two variants of this architecture: the three-way variant which uses the three-way hybrid sequential/parallel multiplier and the two-way variant which uses the two-way hybrid sequential/parallel multiplier.

In Table 4, we present the FPGA synthesis results of the two variants of the cryptoprocessor. We have run several test vectors through both implementations and both have been verified for correctness. In our design special attention was given to handle input and outputs in a more manageable way. This is because we have four 1223-bit inputs and four 1223-bit outputs. Current FPGA devices cannot handle such large amount of input/output in parallel. As a result, our VHDL codes were written for FPGA devices to support 306-bit words for input and output, i.e., an 1223-bit input is accepted in four steps. The results presented here do not include input and output timings and are purely on the computational time required to perform a single pairing.

The frequencies do not change compared to the frequencies of the multiplier (Table 2). Indeed the the critical path of the cryptoprocessor is part of the multiplier in $\mathbb{F}_{2^{1223}}$. The other parts of the architecture does not vary in the two considered variants of the cryptoprocessor, i.e., two-way and three-way variants. A consequence is that the difference in LUTs between the two architectures is roughly the same as the difference in LUTs of the multiplier in $\mathbb{F}_{2^{1223}}$ (cf. Table 2). We remark also that $\cong 80\%$ of the computation time is consumed by the Miller's loop and $\cong 20\%$ is consumed by the final exponentiation.

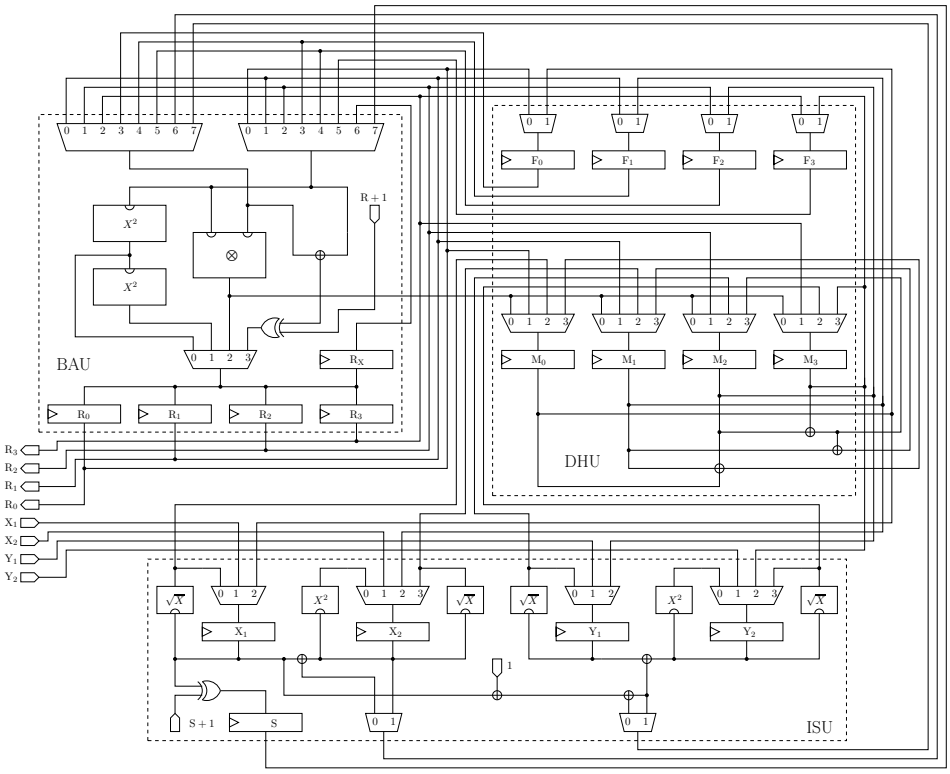



**Fig. 3.** Proposed $\eta_T$ pairing architecture

**Table 4.** Implementation results for FPGA (Virtex-6)

| Cryptoprocessor Design | Max. Freq. (MHz) | Processor (LUTs) | #CC per pairing | Multiplier (LUTs) | Time ($\mu$s) | #CC for all Mults. | #CC/ per Inverse | #CC per Ext. Mult. |
|---|---|---|---|---|---|---|---|---|
| Two-way way hybrid seq/para | 271 | 50,179 | 40,320 | 19,721 | 148.78 | 38,562 | 794 | 89 |
| Three-way hybrid seq/para | 267 | 63,103 | 27,308 | 33,546 | 102.40 | 25,710 | 752 | 62 |

#CC denotes number of clock cycles.

In Table 5 we present our results for ASIC. The VHDL code is fed into Synopsys Design Compiler Version `E-2010.12` for synthesizing with TSMC 65 nm library `TCBN65GPLUS` at best case corner. At 500 MHz the architecture which uses the two-way sequential/parallel multiplier needs 80.64 $\mu$s and the architecture

**Table 5.** Implementation results for ASIC

| Architecture | Gates | Area ($\mu m^2$) | Time ($\mu s$) | Power (mW) | Energy ($\mu J/calc$) | $calc.s/J$ |
|---|---|---|---|---|---|---|
| Two-way split | 497,417 | 716,281 | 80.640 | 389.139 | 31.38 | 31,867 |
| Three-way split | 548,453 | 789,773 | 54.616 | 486.300 | 26.56 | 37,651 |

using the three-way sequential/parallel multiplier needs $54.616\,\mu s$ per calculation. Energy consumption is calculated as $power \times time$ for a single computation and reported in Table 5. Obviously the architecture using the three-way sequential/parallel multiplier consumes more power, whereas a pairing based computation needs less energy in the same circuit, implying that the three-way approach is greener than the two-way.

## 6 Comparison and Conclusion

As the pairing algorithm used in the paper is heavily dominated by more than four thousand multiplications over $\mathbb{F}_{2^{1223}}$, it has thus been crucial to deploy high performance multiplier(s). To this end, the use of the Toeplitz matrix-vector product approach has enabled us to reduce the area and time requirements considerably. In order to explore area-time trade-offs for designs dealing with a large field like $\mathbb{F}_{2^{1223}}$, we have implemented both two-way and three-way split based multipliers in a hybrid sequential/parallel manner.

Using our two-way and three-way split multipliers, we have implemented the resulting $\eta_T$ pairing cryptoprocessors in FPGA and ASIC. In Table 6 we have reported some known implementation results for pairing at the 128-bit security level. As it can be seen from Table 6, the previous best results is the one

**Table 6.** Comparison results for FPGA and ASIC

| Design | Curve | FPGA | Area (Slices/DSP) | Freq. | Time ($\mu s$) | $A$ (slice) $\cdot T$ (sec) |
|---|---|---|---|---|---|---|
| Fan *et al.* [11] | $E/\mathbb{F}_{p256}$ | xc6vx240-3 | 4,014/42 | 210 | 1,170 | – |
| Ghosh *et al.* [13] | $E/\mathbb{F}_{p256}$ | xc4vlx200-12 | 5,2000 | 50 | 16,400 | 852.8 |
| Estibals [7] | $E/\mathbb{F}_{3^{5\cdot 97}}$ | xc4vlx200-11 | 4,755 | 192 | 2,227 | 10.6 |
| Aranha *et al.* [1] | $Co/\mathbb{F}_{2^{367}}$ | xc4vlx200-11 | 4,518 | 220 | 3,518 | 15.9 |
| Ghosh *et al.* [12] | $E/\mathbb{F}_{2^{1223}}$ | xc4vlx200-11 | 35,458 | 168 | 286 | 10.1 |
| Ghosh *et al.* [12] | $E/\mathbb{F}_{2^{1223}}$ | xc6vlx130t-3 | 15,167 | 250 | 190 | 2.9 |
| **This work**/2-way | $E/\mathbb{F}_{2^{1223}}$ | xc6vlx365t-3 | 13,596 | 271 | 148 | 2.0 |
| **This work**/3-way | $E/\mathbb{F}_{2^{1223}}$ | xc6vlx365t-3 | 16,403 | 267 | 102 | 1.7 |
| Design | Curve | ASIC Tech. | Area (Gates) | Freq. | Time ($\mu s$) | $A$ (gates) $\cdot T$ (sec) |
| Kammler *et al.*[15] | $E/\mathbb{F}_{p256}$ | 130 nm CMOS | 97,000 | 338 | 15,800 | 1,532.6 |
| Fan*et al.* [10] | $E/\mathbb{F}_{p256}$ | 130 nm CMOS | 183,000 | 204 | 2,900 | 530.7 |
| **This work**/2-way | $E/\mathbb{F}_{2^{1223}}$ | 65 nm CMOS | 497,417 | 500 | 80.64 | 40.1 |
| **This work**/3-way | $E/\mathbb{F}_{2^{1223}}$ | 65 nm CMOS | 548,453 | 500 | 54.616 | 29.9 |

presented in [12] which requires 190 $\mu$s to complete a single computation of the $\eta_T$ pairing. The work presented here requires 102 $\mu$s (∼45% improvement) for the architecture using the three-way-sequential/parallel multiplier and 148 $\mu$s (∼22% improvement) for the architecture using a two-way sequential/parallel multiplier. Our FPGA implementations also offer the best area-time products. Our ASIC implementations are faster and have better area-time products than those of the previous best implementations.

**Acknowledgements.** This work was supported in part by NSERC grants awarded to Dr. Hasan. The bulk of the work was done while Drs. Adikari and Negre were at the University of Waterloo.

## References

1. Aranha, D.F., Beuchat, J.-L., Detrey, J., Estibals, N.: Optimal Eta Pairing on Supersingular Genus-2 Binary Hyperelliptic Curves. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 98–115. Springer, Heidelberg (2012)
2. Beuchat, J.-L., Detrey, J., Estibals, N., Okamoto, E., Rodríguez-Henríquez, F.: Fast Architectures for the $\eta_T$ Pairing over Small-Characteristic Supersingular Elliptic Curves. IEEE Transactions on Computers 60(2), 266–281 (2011)
3. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. SIAM Journal on Computing 32(3), 586–615 (2003)
4. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. Journal of Cryptology 17(4), 297–319 (2004)
5. Canright, D.: A very compact Rijndael S-box. Technical Report NPS-MA-04-001, Naval Postgraduate School (2004)
6. Cheung, R.C.C., Duquesne, S., Fan, J., Guillermin, N., Verbauwhede, I., Yao, G.X.: FPGA Implementation of Pairings Using Residue Number System and Lazy Reduction. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 421–441. Springer, Heidelberg (2011)
7. Estibals, N.: Compact Hardware for Computing the Tate Pairing over 128-Bit-Security Supersingular Curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 397–416. Springer, Heidelberg (2010)
8. Fan, H., Hasan, M.A.: A New Approach to Sub-quadratic Space Complexity Parallel Multipliers for Extended Binary Fields. IEEE Transactions on Computers 56(2), 224–233 (2007)
9. Fan, H., Sun, J., Gu, M., Lam, K.-Y.: Overlap-free Karatsuba-Ofman polynomial multiplication algorithms. Information Security, IET 4, 8–14 (2010)
10. Fan, J., Vercauteren, F., Verbauwhede, I.: Faster $\mathbb{F}_p$-Arithmetic for Cryptographic Pairings on Barreto-Naehrig Curves. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 240–253. Springer, Heidelberg (2009)
11. Fan, J., Vercauteren, F., Verbauwhede, I.: Efficient Hardware Implementation of $\mathbb{F}_p$-Arithmetic for Pairing-Friendly Curves. IEEE Transactions on Computers 61(5), 676–685 (2012)
12. Ghosh, S., Roychowdhury, D., Das, A.: High Speed Cryptoprocessor for $\eta_T$ Pairing on 128-bit Secure Supersingular Elliptic Curves over Characteristic Two Fields. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 442–458. Springer, Heidelberg (2011)

13. Ghosh, S., Mukhopadhyay, D., Roychowdhury, D.: High Speed Flexible Pairing Cryptoprocessor on FPGA Platform. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 450–466. Springer, Heidelberg (2010)
14. Itoh, T., Tsujii, S.: A Fast Algorithm for Computing Multiplicative Inverses in GF($2^m$) Using Normal Bases. Inf. Comput. 78(3), 171–177 (1988)
15. Kammler, D., Zhang, D., Schwabe, P., Scharwaechter, H., Langenberg, M., Auras, D., Ascheid, G., Mathar, R.: Designing an ASIP for Cryptographic Pairings over Barreto-Naehrig Curves. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 254–271. Springer, Heidelberg (2009)
16. Mastrovito, E.D.: VLSI Architectures for Computation in Galois Fields. PhD thesis, Linkoping University, Department of Electrical Engineering, Linkoping, Sweden (1991)
17. Barreto, P.S.L.M., Galbraith, S.D., O'Eigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular Abelian varieties. Designs, Codes and Cryptography 42(3), 239–271 (2007)
18. Sunar, B.: A Generalized Method for Constructing Subquadratic Complexity GF($2^k$) Multipliers. IEEE Transactions on Computers 53, 1097–1105 (2004)
19. Winograd, S.: Arithmetic Complexity of Computations. Society For Industrial & Applied Mathematics, U.S. (1980)