High Speed Cryptoprocessor for η_T Pairing on 128-bit Secure Supersingular Elliptic Curves over Characteristic Two Fields

Santosh Ghosh, Dipanwita Roychowdhury, and Abhijit Das

Computer Science and Engineering
Indian Institute of Technology Khaargpur
WB, India, 721302
{santosh,drc,abhij}@cse.iitkgp.ernet.in

Abstract. This paper presents an efficient architecture for computing cryptographic η_T pairing for providing 128-bit security. A cryptoprocessor is proposed for Miller's Algorithm with a new 1223-bit Karatsuba multiplier that exploits parallelism. To the best of our knowledge this is the first hardware implementation of 128-bit secure η_T pairing on supersingular elliptic curves over characteristic two fields. The design has been implemented on Xilinx FPGAs. The place-and-route results show that the proposed design takes only $190\mu s$ to complete an 128-bit secure η_T pairing on a Virtex-6 FPGA. The proposed cryptoprocessor achieves eight times speedup compared to the best known existing design. It also outperforms the previous designs with respect to $area \times time$ product.

Keywords: Pairing, Supersingular curves, characteristic two fields, FPGA, Karatsuba multiplier.

1 Introduction

Since 2000, pairing is used in cryptography for developing security schemes for various applications. It is well suited for identity based cryptography [8] which has gained lot of importance in recent times. As a natural consequence, implementations of pairings are also extremely important. The implementations should be cost effective, both in terms of time and space requirement. In practice, pairing could be implemented either as a software library executed on general purpose processors or as a dedicated cryptoprocessor. However, the later one is favored due to huge mathematical operations required for pairing computation [5]. This paper broadly addresses design techniques of a pairing cryptoprocessor for high security level.

Pairing for cryptographic applications are computed on elliptic or hyperelliptic curves defined over suitably large finite fields and having small embedding degree [19,13]. The security of a pairing depends on the underlying algebraic curves and respective field types. For example, 128-bit symmetric security could be achieved by computing η_T pairing [3,18] on a supersingular elliptic curve

B. Preneel and T. Takagi (Eds.): CHES 2011, LNCS 6917, pp. 442–458, 2011.

defined over $\mathbb{F}_{2^{1223}}$ and having embedding degree $k = 4$. As per NIST recommendation, 128-bit symmetric security is essential beyond 2030 [2]. Therefore, it is of importance to explore the efficient implementation techniques of 128-bit secure pairings on different platforms.

Hardware implementation of 128-bit secure pairings was introduced in 2009, individually by Kammler *et al.* [21] and Fan *et al.* [12]. Both of them described hardware implementation techniques for computing 128-bit secure pairings over Barreto-Naehrig curves (BN curves) [4]. These CMOS based designs take $15.8ms$ and $2.9ms$ for computing an optimal-ate pairing, respectively. Thereafter, designs in [10,14,1,9] are appeared in literature, which computes 128-bit secure pairings in $2.3ms$, $16.4ms$, $3.5ms$, and $1.07ms$ respectively. However, to the best of our knowledge there is no hardware implementation results available in the literature which computes 128-bit secure pairings below one ms time limit. High-speed software implementations reported in [6,7] compute 128-bit secure pairings in $0.832ms$ and $1.87ms$. The work proposed by Beuchat *et al.* [5] describes design architectures for η_T pairings on supersingular elliptic curves over characteristic two and three fields for a maximum of 105-bit and 109-bit security, respectively. However, to the best of the authors' knowledge no hardware architectures are available for computing η_T pairing on 128-bit secure supersingular elliptic curves over binary fields.

Contribution. This paper explores the hardware design techniques for η_T pairing on 128-bit secure supersingular elliptic curves over characteristic two fields. It first designs cost-effective and time-efficient hybrid architectures for Karatsuba multiplication over $\mathbb{F}_{2^{1223}}$ field, on which the respective supersingular elliptic curve is defined. The major contributions of the paper are highlighted here.

- The paper explores area-time tradeoff designs of hybrid Karatsuba multiplier over $\mathbb{F}_{2^{1223}}$ field.
- It further explores high speed architectures for computing η_T pairing on supersingular elliptic curves based on the proposed hybrid multiplier.
- It provides the first hardware implementation result of an 128-bit secure pairing on elliptic curves over characteristic two fields.
- The proposed design is the first one which computes an 128-bit secure pairing in less than one ms.

The proposed design of hybrid multiplier and parallelism techniques result in the high speed cryptoprocessor which achieves significant improvement on the performance of 128-bit secure η_T pairing on supersingular elliptic curves over small characteristic fields.

Organization of the Paper. Section 2 of the paper proposes design techniques of Karatsuba multipliers for $\mathbb{F}_{2^{1223}}$ field. Section 3 describes the proposed pairing cryptoprocessor. Results and comparisons are provided in Section 4. Finally, the paper is concluded in Section 5.

2 The $\mathbb{F}_{2^{1223}}$-Multiplier

Multiplication is the key operation of a pairing computation. The 128-bit secure η_T pairing could be computed on a supersingular elliptic curve defined over 1223-bit characteristic-two fields. Therefore, the multiplication in $\mathbb{F}_{2^{1223}}$ field is an essential operation in this context. Karatsuba multiplication [22] is one of the most efficient and popular techniques for fields like $\mathbb{F}_{q^m}$. This technique is based on divide-and-conquer algorithm, where a full m-bit multiplication is divided recursively into several m/k-bit multiplications with small $k \in \{2, 3\}$. It then accumulates the results of smaller multiplications for the generating final result. Karatsuba technique for $k = 2$ computes product $a \cdot b$ of two elements $a, b \in \mathbb{F}_{q^m}$ by the following way:

$$\begin{aligned} a \cdot b &= (a_1 x^{\lceil m/2 \rceil} + a_0)(b_1 x^{\lceil m/2 \rceil} + b_0) \\ &= a_1 b_1 x^m + [(a_1 + a_0)(b_1 + b_0) - a_1 b_1 - a_0 b_0]\, x^{\lceil m/2 \rceil} + a_0 b_0. \end{aligned} \tag{1}$$

Hence, an m-bit multiplication can be performed by three $m/2$-bit multiplications along with four m-bit and two $m/2$-bit addition/subtraction operations. Generalization of Karatsuba multiplication is provided in [29]. We refer to the reader [27,17] for getting idea about implementation techniques of Karatsuba multiplication. Efficient implementation of Karatsuba multiplication is challenging– mainly for larger field sizes like $m = 1223$. It is more challenging on resource-constrained environments like an FPGA platform where the number of logic cells are limited. We may follow several ways for making trade-off between the multiplication latency and hardware resources for developing a multiplier for $\mathbb{F}_{2^{1223}}$ field. Fig. 1 shows the decomposition of a 1223-bit operand for Karatsuba multiplication with $k = 2$. The operand is decomposed recursively up to their 19-bit or 20-bit levels as it gives the most optimum design [27].

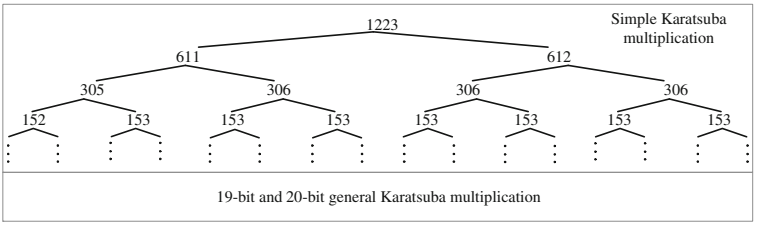

Fig. 1. The decomposition of an 1223-bit operand for Karatsuba multiplication

Fully Parallel Multiplier for $\mathbb{F}_{2^{1223}}$. A fully parallel Karatsuba multiplier can be designed for $\mathbb{F}_{2^{1223}}$ field by following the decomposition as shown in Fig. 1. After implementing it by Verilog (HDL) we synthesize the design by ISE tool for a Virtex-4 FPGA. The synthesis tool estimates 324342 LUTs for an 1223-bit fully parallel Karatsuba multiplier, which makes it infeasible to implement on a single Virtex-4 FPGA device.

Serial Use of 612-bit Parallel Multiplier. As an alternative to area-time tradeoff we take a fully parallel 612-bit Karatsuba multiplier on which three multiplications are performed in serial for computing multiplication in $\mathbb{F}_{2^{1223}}$. After synthesizing by ISE synthesis tool, it demands 95324 LUTs. Thus, it could be useful to implement a high throughput $\mathbb{F}_{2^{1223}}$ multiplier on a high-end single FPGA device. However, a pairing cryptoprocessor demands more circuits along with multipliers, which may not be put together on a single FPGA device.

2.1 Serial Use of 306-bit Parallel Multiplier

It is shown that the fully parallel multiplier as well as serial use of 612-bit parallel multiplier for $\mathbb{F}_{2^{1223}}$ are infeasible to implement a respective η_T pairing cryptoprocessor. Here we propose a serial use of 306-bit parallel Karatsuba multiplier for $\mathbb{F}_{2^{1223}}$ field. The current multiplier is based on a 306-bit fully parallel Karatsuba multiplier on which top two levels of Fig. 1 are performed in serial. The proposed architecture for computing 1223-bit multiplication based on this serial-parallel hybridization is shown in Fig. 2. The architecture follows the exact steps and nomenclatures of variables that are described in Algorithm 2, Appendix A.

The proposed architecture works as follows. During the initialization stage (Algorithm 2, step 1 to step 7) it breaks the operands a, and b into four parts by following two repeated Karatsuba decompositions. The smaller operands are generated by following way:

$$\begin{aligned} a \cdot b &= (a_1 x^{612} + a_0)(b_1 x^{612} + b_0) \\ &= a_1 b_1 x^{1222} + [(a_1 + a_0)(b_1 + b_0) - a_1 b_1 - a_0 b_0]\, x^{612} + a_0 b_0. \end{aligned}$$

The 1223-bit multiplication is performed by three 612-bit multiplications[1], $a_0 \cdot b_0$, $a_1 \cdot b_1$, and $(a_1 + a_0) \cdot (b_1 + b_0)$, which are further decomposed by following way.

$$\begin{aligned} a_0 \cdot b_0 &= (a_{01} x^{306} + a_{00})(b_{01} x^{306} + b_{00}) \\ &= a_{01} b_{01} x^{612} + [(a_{01} + a_{00})(b_{01} + b_{00}) - a_{01} b_{01} - a_{00} b_{00}]\, x^{306} + a_{00} b_{00} \\ &= a_{01} b_{01} x^{612} + [g_0 h_0 - a_{01} b_{01} - a_{00} b_{00}]\, x^{306} + a_{00} b_{00}, \end{aligned} \tag{2}$$

where, $g_0 = a_{01} + a_{00}$ and $h_0 = b_{01} + b_{00}$. Similarly, the second 612-bit multiplication is performed by following equation.

$$\begin{aligned} a_1 \cdot b_1 &= (a_{11} x^{306} + a_{10})(b_{11} x^{306} + b_{10}) \\ &= a_{11} b_{11} x^{611} + [(a_{11} + a_{10})(b_{11} + b_{10}) - a_{11} b_{11} - a_{10} b_{10}]\, x^{306} + a_{10} b_{10} \\ &= a_{11} b_{11} x^{611} + [g_1 h_1 - a_{11} b_{11} - a_{10} b_{10}]\, x^{306} + a_{10} b_{10}, \end{aligned} \tag{3}$$

where, $g_1 = a_{11} + a_{10}$ and $h_1 = b_{11} + b_{10}$. The third 612-bit multiplication is performed as:

[1] More accurately, two 612-bit multiplications and one 611-bit multiplication. For simplicity we say three 612-bit multiplications.

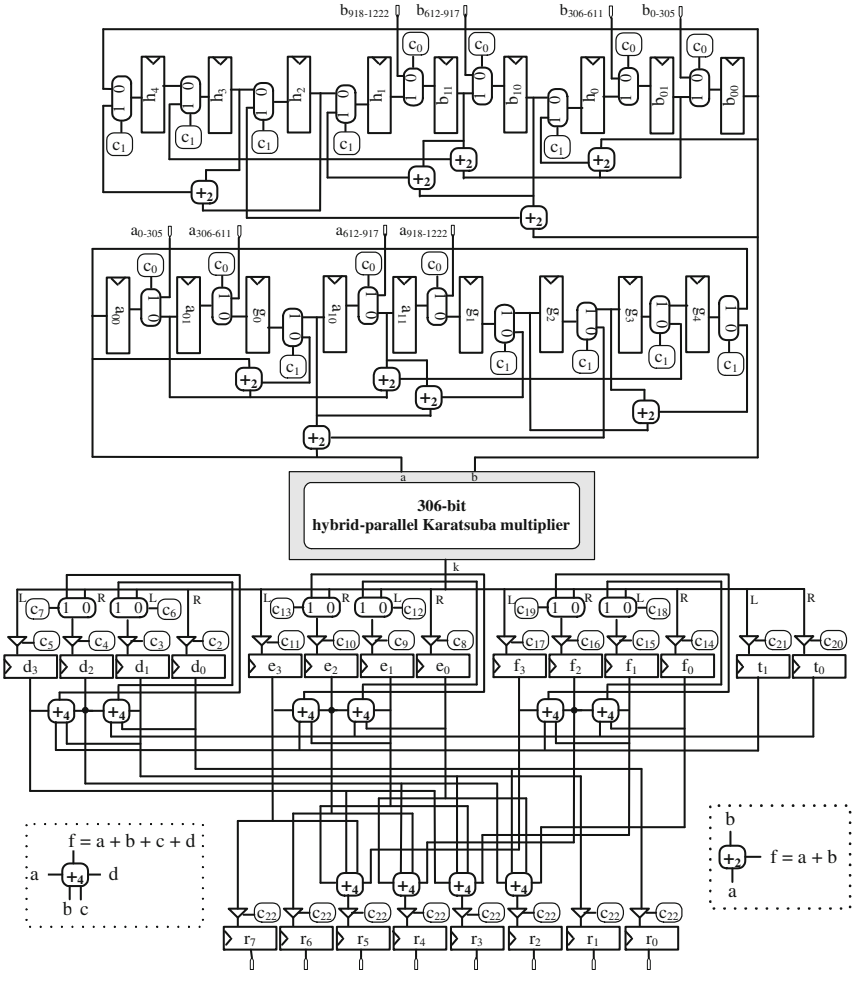

Fig. 2. The architecture of $\mathbb{F}_{2^{1222}}$ multiplier unit

$$
\begin{aligned}
g_2 &= a_{10} + a_{00}; \quad g_3 \;=\; a_{11} + a_{01} \\
h_2 &= b_{10} + b_{00}; \quad h_3 \;=\; b_{11} + b_{01} \\
(a_1 + a_0) \cdot (b_1 + b_0) &= (g_3 x^{306} + g_2)(h_3 x^{306} + h_2) \\
&= g_3 h_3 x^{612} + \left[(g_3 + g_2)(h_3 + h_2) - g_3 h_3 - g_2 h_2\right] x^{306} + g_2 h_2 \\
&= g_3 h_3 x^{612} + \left[g_4 h_4 - g_3 h_3 - g_2 h_2\right] x^{306} + g_2 h_2, \qquad (4)
\end{aligned}
$$

where, $g_4 = g_3 + g_2$ and $h_4 = h_3 + h_2$. Therefore, one 1223-bit multiplication is performed by nine 306-bit multiplications. In our proposed architecture (Fig. 2), the operands of these nine multiplications are stored into nine 306-bit parallel shift registers. These registers are automatically reloaded by synchronous shift

operations so that the two correct operands of 306-bit parallel multiplier are available into a_{00} and b_{00} registers, respectively, at every clock. The 306-bit parallel Karatsuba multiplier takes only one clock cycle to compute one respective multiplication. The strategy of shift register is adopted for avoiding two complex 9-to-1 multiplexers to the multiplier input ports. The first three 309-bit multiplication results (Algorithm 2, step 8 to step 13) are combined to generate the intermediate result of first 612-bit multiplication $a_0 \cdot b_0$. The final result of $a_0 \cdot b_0$ as defined in Eq. 2 (Algorithm 2, step 14) is computed by means of two 306-bit 4-input parallel adders (4-input XORs in this case) and it is stored into the registers $d_i,\ 0 \leq i \leq 3$. Similarly, the result of the second 612-bit multiplication as defined in Eq. 3 (Algorithm 2, step 15 to step 21) is stored into the registers $e_i,\ 0 \leq i \leq 3$, and for the third one, Eq. 4, (Algorithm 2, step 22 to step 28) is stored into the registers $f_i,\ 0 \leq i \leq 3$. Finally, in steps 29 to 31, the algorithm combines the final result of 1223-bit multiplication and stores into the registers $r_i,\ 0 \leq i \leq 7$. The proposed architecture (Fig. 2) takes 10 clock cycles for completing one multiplication in the respective base field $\mathbb{F}_{2^{1223}}$.

Implementation Results on FPGA Platforms. The synthesis tool estimates 34325 LUTs on a Virtex-4 FPGA for implementing the proposed serial use of 306-bit parallel multiplier for $\mathbb{F}_{2^{1223}}$. In this paper, we are looking for a pairing cryptoprocessor on a medium-range FPGA device. The place-and-route results as summarized in Table 1 ensure that this multiplier is suitable for designing our target cryptoprocessor.

Table 1. Cost and time of 1223-bit multipliers on FPGA platforms

Multiplier type	**FPGA family**	**LUTs**	**Frequency** $[MHz]$	**Serial use**	**Multiplication latency** $[ns]$	$(A \cdot T)^{\S}$
Serial use of	Virtex-2	34 547	125	10	80.0	2.76
306-bit parallel	Virtex-4	34 325	168	10	60.0	2.06
multiplier	Virtex-6	30 148	250	10	40.0	1.21
§ : $(A \cdot T)$ represents product of *area* in LUTs and *time* in milliseconds.						

However, designer may opt for *serial use of 153-bit parallel multiplier* with low resources. But, it requires 27 serial use, which slows down the multiplication. On a Virtex-4 FPGA one such multiplier takes 16231 LUTs and achieves maximum 185 MHz clock frequency. Therefore, this multiplier with lower resource requires $151ns$ for completing one 1223-bit multiplication which is 2.5 times slower than *serial use of 306-bit parallel multiplier*. The respective $A \cdot T$ value (2.46) of this design is 1.2 times higher than the same for the design with 306-bit parallel multiplier. Thus, *serial use of 306-bit parallel multiplier* provides the most optimized design with respect to the feasibility of implementation as well as $area \times time$ product.

3 The η_T Pairing Cryptoprocessor over $F_{2^{1223}}$

In this section, we present a high-speed cryptoprocessor for computing the η_T pairing over a large characteristic-two field $\mathbb{F}_{2^{1223}}$. The proposed architecture is depicted in Fig. 3. The pairing computation consists of two major operations – the non-reduced pairing (Miller's algorithm) and the final exponentiation. Beuchat *et al.* in [5] proposed two separate coprocessors on which these two tasks are pipelined. Two separate coprocessors in pipeline helps to reduce the computation time. But, at the same time it needs larger area. In case of a large field like $\mathbb{F}_{2^{1223}}$ it is important to take care of the overall area requirement for pairing computation as most of its applications demand area-constrained devices. It is observed that almost 50% datapath of both the above coprocessors are consumed by the base-field multipliers. This paper attempts to optimize the area of an η_T pairing cryptoprocessor. We propose here a common datapath for computing both the Miller's algorithm and the final exponentiation. Adequate parallelism is also applied in the datapath to achieve a high-speed cryptoprocessor. The supersingular elliptic curves, the representation of the fields, and the η_T pairing algorithm that are used in this paper are described in Appendix B.

The η_T pairing computation in characteristic-two field is described in [16]. We rewrite it, specifically for $\mathbb{F}_{2^{1223}}$, in Algorithm 1 with parenthesized indices in superscript in order to emphasize the intrinsic dependency as well as parallelism of the pairing computation. Two interdependent operations in the Miller's algorithm, namely, the computation of the $G^{(i)}$ (step 7 to step 10) and the sparse multiplication[2] $F^{(i-1)} \cdot G^{(i)}$ over $\mathbb{F}_{(2^{1223})^4}$ along with the computation of $x_2^{(i)}$, $y_2^{(i)}$ for next iteration (step 11 and step 12) are performed in serial, whereas we apply the parallelism within each of these two operations.

3.1 Computation of Miller's Loop

The proposed cryptoprocessor as shown in Fig. 3 first computes the non-reduced pairing based on Algorithm 1. It breaks this computation in three sub-parts as described here. We use the same nomenclature of Algorithm 1 for representing the intermediate results in the architecture.

Initialization. The registers $x_1^{(i)}$, $y_1^{(i)}$, $x_2^{(i-1)}$, $y_2^{(i-1)}$, and $s^{(i)}$ are initialized according to step 1 and step 2 of Algorithm 1 [Fig. 3]. During the initialization of $s^{(i)}$, the operation x_1+1 is performed simply by inverting the least significant bit of x_1 as $x_1 \in F_2[x]$. The variables $t_0^{(i)}$ and $t_1^{(i)}$ are initialized by two sets of 2-input XORs,[3] which perform $s^{(0)}+x_2^{(0)}$ and $y_1^{(0)}+y_2^{(0)}$, respectively. These two operations are performed on the fly. As defined in step 4, the initialization of

[2] An operand in $\mathbb{F}_{(2^{1223})^4}$ is sparse when some of its coefficients are trivial (i.e., either zero or one).

[3] The addition in $\mathbb{F}_2[x]$ is performed by simple bit-wise XOR. Therefore, addition and XOR are used with same meaning in this paper.

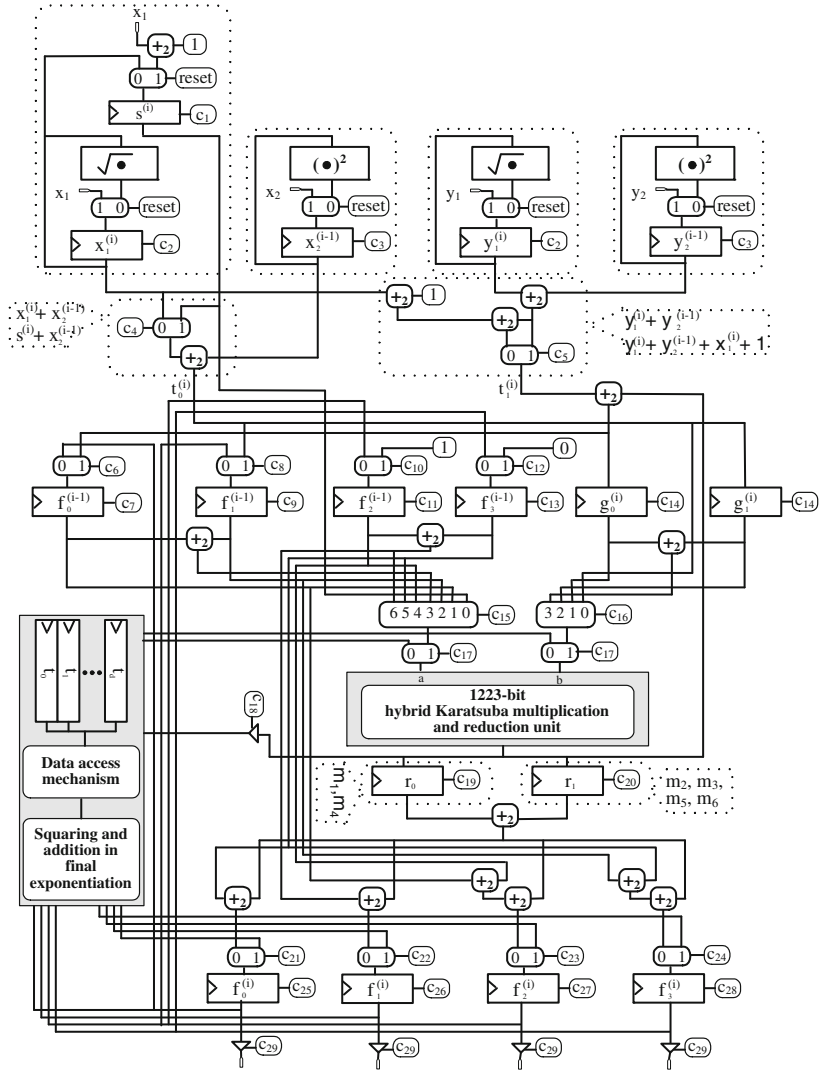

Fig. 3. The η_T pairing cryptoprocessor over $F_{2^{1223}}$

register $f_0^{(i-1)}$ is done by means of the output of a multiplication followed by a 2-input addition. Similarly, the initialization of $f_1^{(i-1)}$ requires a 2-input addition, whereas the same for $f_2^{(i-1)}$ and $f_3^{(i-1)}$ are trivial. In total, the initialization part of Miller's algorithm takes only 12 clock cycles in our proposed cryptoprocessor.

Computation of $G^{(i)}$. We represent $G^{(i)} \in \mathbb{F}_{(2^{1223})^4}$ in $\{1, u, v, uv\}$ basis. However, throughout Miller's loop $G^{(i)}$ contains sparse value which is represented as

Algorithm 1 . Computing the η_T pairing on $E/\mathbb{F}_{2^{1223}}$. Intermediate variables in uppercase belong to $\mathbb{F}_{(2^{1223})^4}$, whereas those in lowercase to $\mathbb{F}_{2^{1223}}$.

Input: $P(x_1, y_1)$ and $Q(x_2, y_2) \in E(\mathbb{F}_{2^{1223}})[r]$.
Output: $\eta_T(P, Q)$.

1. $x_1^{(0)} \leftarrow x_1$; $y_1^{(0)} \leftarrow y_1$; $x_2^{(0)} \leftarrow x_2$; $y_2^{(0)} \leftarrow y_2$;
2. $s^{(0)} \leftarrow x_1 + 1$;
3. $t_0^{(0)} \leftarrow s^{(0)} + x_2^{(0)}$; $t_1^{(0)} \leftarrow y_1^{(0)} + y_2^{(0)}$;
4. $f_0^{(0)} \leftarrow s^{(0)} \cdot t_0^{(0)} + t_1^{(0)}$; $f_1^{(0)} \leftarrow s^{(0)} + x_2^{(0)}$; $f_2^{(0)} \leftarrow 1$; $f_3^{(0)} \leftarrow 0$;
5. $F^{(0)} \leftarrow f_0^{(0)} + f_1^{(0)}u + f_2^{(0)}v + f_3^{(0)}uv$;
6. **for** i *from* 1 *to* 612 **do**
7. $s^{(i)} \leftarrow x_1^{(i-1)}$, $x_1^{(i)} \leftarrow \sqrt{x_1^{(i-1)}}$; $y_1^{(i)} \leftarrow \sqrt{y_1^{(i-1)}}$;
8. $t_0^{(i)} \leftarrow x_1^{(i)} + x_2^{(i-1)}$; $t_1^{(i)} \leftarrow y_1^{(i)} + y_2^{(i-1)} + x_1^{(i)} + 1$;
9. $g_0^{(i)} \leftarrow s^{(i)} \cdot t_0^{(i)} + t_1^{(i)}$; $g_1^{(i)} \leftarrow s^{(i)} + x_2^{(i-1)}$;
10. $G^{(i)} \leftarrow g_0^{(i)} + g_1^{(i)}u + v$;
11. $F^{(i)} \leftarrow F^{(i-1)} \cdot G^{(i)}$;
12. $x_2^{(i)} \leftarrow (x_2^{(i-1)})^2$; $y_2^{(i)} \leftarrow (y_2^{(i-1)})^2$;
13. **end for**
14. **return** $(F^{(612)})^{(2^{2446}-1)(2^{1223}-2^{612}+1)}$.

$g_0^{(i)} + g_1^{(i)}u + 1$. The computation of $g_0^{(i)}$ (Algorithm 1, step 9) is performed by means of one multiplication in $\mathbb{F}_{2^{1223}}$ followed by one 2-input addition. The operands of above multiplication $s^{(i)}$ and $t_0^{(i)}$ are generated on the fly after computing two square-root operations (in step 7) in parallel. Current cryptoprocessor computes the square-root operations inexpensively by means of simple shift and XOR operations. Let, $a = \sum a_i x^i \in \mathbb{F}_{2^{1223}}$, then $\sqrt{a} = \sum a_{2j} x^j + (x^{612} + x^{128}) \sum a_{2j+1} x^j$, which is computed in one clock. Therefore, in the proposed cryptoprocessor (Fig. 3) the control signal c_2 is activated only in that respective clock cycle during the execution of each iteration of Miller's algorithm. After computing $x_1^{(i)}$ and $y_1^{(i)}$ at the next clock the cryptoprocessor starts the multiplication $s^{(i)} \cdot t_0^{(i)}$. The multiplication in $\mathbb{F}_{2^{1223}}$ takes 10 clock cycles and immediately at the next clock the cryptoprocessor updates $g_0^{(i)}$ and $g_1^{(i)}$ registers. Therefore, in total the computation of $G^{(i)}$ takes 12 clock cycles by the proposed cryptoprocessor.

Sparse Multiplication over $\mathbb{F}_{(2^{1223})^4}$. The operation $F^{(i-1)} \cdot G^{(i)}$ in Algorithm 1, step 11 is identified as sparse multiplication in $\mathbb{F}_{(2^{1223})^4}$ as $G^{(i)}$ consists only two non-trivial coefficients. The computation of this sparse multiplication is much easier than a full multiplication in the above extension field. The computation procedure on our proposed cryptoprocessor is described in Table 2.

In the proposed cryptoprocessor multiplications $m_i, 1 \leq i \leq 6$, are performed in serial on a single $\mathbb{F}_{2^{1223}}$ multiplier core. The registers r_0 and r_1 (in Fig. 3) are alternatively used to hold the multiplication outputs. After completing m_1 and m_2 we start m_3 at the next clock when in parallel the value of $f_0^{(i)}$ is computed

Table 2. Computation of $F^{(i-1)} \cdot G^{(i)}$

$m_1 : \ r_0 \leftarrow f_0^{(i-1)} \cdot g_0^{(i)}$;	$m_4 : \ r_0 \leftarrow f_2^{(i-1)} \cdot g_2^{(i)}$;
$m_2 : \ r_1 \leftarrow f_1^{(i-1)} \cdot g_1^{(i)}$;	$m_5 : \ r_1 \leftarrow f_3^{(i-1)} \cdot g_3^{(i)}$;
$x_4^{(1)}: \ f_0^{(i)} \leftarrow (r_0 + r_1) + f_4^{(i-1)}$;	$x_4^{(3)}: \ f_2^{(i)} \leftarrow (r_0 + r_1) + (f_1^{(i-1)} + f_3^{(i-1)})$;
$m_3 : \ r_1 \leftarrow (f_0^{(i-1)} + f_1^{(i-1)}) \cdot (g_0^{(i)} + g_1^{(i)})$;	$m_6 : \ r_1 \leftarrow (f_2^{(i-1)} + f_3^{(i-1)}) \cdot (g_0^{(i)} + g_1^{(i)})$;
$x_4^{(2)}: \ f_1^{(i)} \leftarrow (r_0 + r_1) + (f_3^{(i-1)} + f_4^{(i-1)})$;	$x_4^{(4)}: \ f_1^{(i)} \leftarrow (r_0 + r_1) + (f_2^{(i-1)} + f_4^{(i-1)})$;

by two sets of 2-input XORs as defined by $x_4^{(1)}$ in Table 2. Similarly, we perform m_4 and $x_4^{(2)}$ in parallel and also do m_6 and $x_4^{(3)}$. Finally, after m_6 we execute $x_4^{(4)}$ for computing $f_3^{(i)}$ at the next clock cycle. Therefore, the computation of sparse multiplication $F^{(i-1)} \cdot G^{(i)}$ takes 61 clock cycles in the proposed cryptoprocessor.

Computation of $x_2^{(i)}$ and $y_2^{(i)}$. Squaring over $\mathbb{F}_2[x]$ is free. Let $a = \sum a_i x^i \in \mathbb{F}_{2^{1223}}$ then $a^2 = \sum a_i x^{2i}$. However, the reduction after squaring requires some XOR operations, which are performed in parallel in only one clock cycle. The computation of $x_2^{(i)}$ and $y_2^{(i)}$ are independent of the last step of sparse multiplication (step $x_4^{(4)}$ of Table 2). Therefore, they are computed in parallel with $x_4^{(4)}$ which does not take any additional time.

Computation Cost of Miller's Algorithm. One iteration of Miller's algorithm is performed by following three parts. The computation of $G^{(i)}$ which takes 12 clock cycles, the computation of $F^{(i-1)} \cdot G^{(i)}$ which takes 61 clock cycles, and the computation of $x_2^{(i)}, y_2^{(i)}$ which is free. Thus, in total, each iteration of Algorithm 1 takes 73 clock cycles, which incurs 44688 clock cycles for computing whole Miller's algorithm including initialization.

3.2 Computation of Final Exponentiation

The output $F^{(612)} \in \mathbb{F}_{(2^{1223})^4}$ of the Miller's algorithm is raised to the power $(2^{2446} - 1)(2^{1223} - 2^{612} + 1)$. The 2^{1223}-th powering an element $G = g_0 + g_1 u + g_2 v + g_3 uv$ in $\mathbb{F}_{(2^{1223})^4}$ is easily computed by following equation.

$$G^{2^{1223}} = (g_0 + g_1 + g_2) + (g_1 + g_2 + g_3)u + (g_2 + g_3)v + g_3 uv, \tag{5}$$

which is computed by three additions (one 2-input and two 3-input additions). Thus, two clock cycles are taken for computing $(F^{(612)})^{2^{2446}}$ by the current cryptoprocessor. Further we perform one inversion followed by one multiplication in $\mathbb{F}_{(2^{1223})^4}$ for computing $(F^{(612)})^{2^{2446}-1}$.

The Inversion in $\mathbb{F}_{(2^{1223})^4}$. Let $G = g_0 + g_1 u + g_2 v + g_3 uv$ and $H = G^{-1} = h_0 + h_1 u + h_2 v + h_3 uv$ then $(g_0 + g_1 u + g_2 v + g_3 uv)(h_0 + h_1 u + h_2 v + h_3 uv) = 1$. This could follow the matrix representation :

$$\begin{bmatrix} g_0 & g_1 & g_3 & g_2+g_3 \\ g_1 & g_0+g_1 & g_2+g_3 & g_2 \\ g_2 & g_3 & g_0+g_2 & g_1+g_3 \\ g_3 & g_2+g_3 & g_1+g_3 & g_0+g_1+g_2+g_3 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

From which the value of h_1, h_2, h_3, and h_4 could be solved by $(I + 36M + 8S + 57A)$, where I, M, S, A stand for inversion, multiplication, squaring, and addition in the base field $\mathbb{F}_{2^{1223}}$. Operation I is performed by Itoh-Tsujii algorithm [20], which requires $(14M+1222S)$. Thus the cost for computing inversion in $\mathbb{F}_{(2^{1223})^4}$ is $(50M + 1230S + 57A)$. Thereafter, a multiplication in $\mathbb{F}_{(2^{1223})^4}$ is performed by $(16M + 22A)$ operations. Thus, in total, the computation of $(F^{(612)})^{2^{2446}-1}$ requires $(66M + 1230S + 82A)$ operations.

The Exponentiation by $(2^{1223} - 2^{612} + 1)$. The second part of the exponent $(2^{1223} - 2^{612} + 1)$ is raised to the power of $(F^{(612)})^{2^{2446}-1}$ by means of $(32M + 612S + 53A)$. This is possible as the inverse of $(F^{(612)})^{2^{2446}-1}$ in $\mathbb{F}_{(2^{1223})^4}$ is performed by computing $((F^{(612)})^{2^{2446}-1})^{2^{2446}}$, which is easy as shown in Eq. 5. Thus, major operations in the second part are two multiplications in $\mathbb{F}_{(2^{1223})^4}$.

Computation Costs of Final Exponentiation and η_T Pairing. The final exponentiation is performed by means of $(98M + 1842S + 135A)$ operations. In our proposed cryptoprocessor (Fig. 3) the multiplier unit is shared by both Miller's algorithm and the final exponentiation. The control signal c_{17} selects operands from one of these two operations. The squaring and additions of final exponentiation are performed separately from the Miller's algorithm. Some of squaring and additions are performed in parallel. The proposed cryptoprocessor computes final exponentiation in 2922 clock cycles, which is much less than the cycle count for computing Miller's algorithm. Total clock cycle count for computing an 128-bit secure η_T pairing is 47610 on our proposed architecture.

4 Results

The whole design has been done in Verilog (HDL). All results have been obtained from the place-and-route report of Xilinx ISE Design Suit. Table 3 shows the implementation results. The critical path of the design is formed in between the input and the output of the hybrid 306-bit Karatsuba multiplier (in Fig. 2). We produce the results for fair comparison, observing the performance of the proposed cryptoprocessor on different FPGA platforms. The Virtex-6 is the latest FPGA family of Xilinx, on which the proposed design runs at a maximum frequency of $250MHz$. In total, it uses 15167 logic slices including whole data path (for Miller's algorithm and for final exponentiation), the controller logic, and registers on the Virtex-6 FPGA, where it finishes computation of one 128-bit secure η_T pairing in $190\mu s$.

Table 3. Implementation results of the η_T pairing cryptoprocessor

Platform	**Slice**	**LUT**	**Frequency** $[MHz]$	**Clock Cycles**	**Security** $[bit]$	**Times** $[\mu s]$
Virtex-2	36534	69367	125	47610	128	381
Virtex-4	35458	69367	168			286
Virtex-6‡	15167	54681	250			190
‡ : One Virtex-6 slice consists of four LUTs and eight flip-flops.						

4.1 Comparison with Existing Designs

Two aspects of the proposed design are considered when it is compared with the existing designs. First, we compare it with the existing η_T pairing processors over characteristic-two fields as summarized in Table 4. We consider only the design results with maximum security level provided by the respective authors. To the best of the authors' knowledge no hardware implementation is available

Table 4. Hardware designs for the η_T pairing

Designs	**Curve**	**Security** $[bit]$	**FPGA**	**Area** $[Slices]$	**Frequency** $[MHz]$	**Times** $[\mu s]$
Shu *et al.* [28]	$E/\mathbb{F}_{2^{557}}$	96	xc4vlx200-10	37931	66	675.5
Beuchat *et al.* [5]	$E/\mathbb{F}_{2^{691}}$	105	xc4vlx200-11	78874	130	18.8
This work	$E/\mathbb{F}_{2^{1223}}$	128	xc4vlx200-11	35458	168	286.0
This work	$E/\mathbb{F}_{2^{1223}}$	128	xc6vlx130t-3	15167	250	190.0

for computing 128-bit secure η_T pairing on supersingular elliptic curves over characteristic-two fields. The existing designs in this respect are for a maximum of 105-bit secure design over $\mathbb{F}_{2^{691}}$ field, which is proposed by Beuchat *et al.* in [5]. The design proposed in [5] computes 105-bit secure η_T pairing and achieves a very good speed of $18.8\mu s$. However, compared to the respective design in [5] our design with higher security level demands much lesser, less than half, number of slices on the same FPGA family. As a result, the proposed design could be implemented on a medium-range Virtex-4 device, whereas the existing one's demand a high-range device in the same FPGA family. This makes our design more useful in resource-constrained identity-aware devices.

The second aspect of the design is considered on the fact of 128-bit secure pairing computation irrespective of underlying curve and field types. Table 5 summarizes the comparative studies of related designs. The proposed design is the first one which computes an 128-bit secure pairing in less than one millisecond ($190\mu s$ on a Virtex-6 FPGA) on a dedicated hardware.

All the existing designs except [1] are based on elliptic curves. The design of [1] computes optimal-eta pairing on 128-bit secure supersingular Genus-2 binary hyperelliptic curves. The compact design proposed in [10] computes η_T pairing on supersingular elliptic curves over $\mathbb{F}_{3^{m'}}$ fields. Due to its low area the

Table 5. Hardware designs for 128-bit secure pairings

Designs	**Curve**	**FPGA**	**Area**	**Freq.** $[MHz]$	**Times** $[\mu s]$	$A \cdot T$†
Duquesne et al. [9]§	$E/\mathbb{F}_{p256}$	Stratix III	4233 A‡	165	1070	-
Fan et al. [11]	$E/\mathbb{F}_{p256}$	xc6vlx240-3	4014 Slices, 42 DSP	210	1170	-
Kammler *et al.* [21]	$E/\mathbb{F}_{p256}$	130nm CMOS	97000 Gates	338	15800	-
Fan *et al.* [12]	$E/\mathbb{F}_{p256}$	130nm CMOS	183000 Gates	204	2900	-
Ghosh *et al.* [14]	$E/\mathbb{F}_{p256}$	xc4vlx200-12	52000 Slices	50	16400	852.8
Estibals [10]	$E/\mathbb{F}_{3^{5\cdot 97}}$	xc4vlx200-11	4755 Slices	192	2227	10.6
Aranha *et al.* [1]	$Co/\mathbb{F}_{2^{367}}$	xc4vlx25-11	4518 Slices	220	3518	15.9
This work	$E/\mathbb{F}_{2^{1223}}$	xc4vlx200-11	35458 Slices	168	286	10.1
This work	$E/\mathbb{F}_{2^{1223}}$	xc6vlx130t-3	15167 Slices	250	190	2.9

† $A \cdot T$ represents product of *area* in slices and *time* in seconds.
§ It provides 126-bit security instead of 128-bit.
‡ It has 8 Rowers, each consisting of two 36x36 DSP blocks and one 9x9 multiplier.

Table 6. Software for 128-bit secure pairings

Reference	**Platform**	**Pairing**	**Curve**	**Frequency** $[MHz]$	**Times** $[ms]$
Beuchat et al. [7]	core i7 2.8GHz	modified Tate	$E/\mathbb{F}_{3^{509}}$	2800	1.87
			$E/\mathbb{F}_{2^{1223}}$	2800	3.08
Naehrig et al. [26]	core2 Q6600	optimal-ate	$E/\mathbb{F}_{p256}$	2394	1.86
Beuchat et al. [6]	core i7 2.8GHz	optimal-ate	$E/\mathbb{F}_{p256}$	2800	0.83
Hankerson et al. [16]	64-bit core2	optimal-ate	$E/\mathbb{F}_{p256}$	2400	6.25
		η_T	$E/\mathbb{F}_{2^{1223}}$	2400	16.25
		η_T	$E/\mathbb{F}_{3^{509}}$	2400	13.75
Grabher et al. [15]	64-bit core2	ate	$E/\mathbb{F}_{p256}$	2400	6.01

design of [10] is useful to resource constrained applications. It is analyzed in Section 5.2, [5] that the number of base-field multiplications required for computing an η_T pairing with high security level over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^{m'}}$ are almost same. This is also true for other fields with 128-bit security. For example, the *optimal-ate* pairing on $E/\mathbb{F}_{p256}$ reported in [9,11,12,14,21] requires 15093 multiplications in the base field [16]. On the other hand, the η_T pairing on $E/\mathbb{F}_{2^{1223}}$ requires 4566 multiplications in the base field, which is only 1/3 of *optimal-ate* pairing. Furthermore, the base field size of $\mathbb{F}_{2^{1223}}$ is 1223 bits which is 4.8 times longer than the size of $\mathbb{F}_{p256}$. Thus, the operation complexities for computing both the pairings are almost same. To sum up, the proposed design achieves a significant performance improvement for computing 128-bit secure pairings on hardware platforms. With respect to the $A \cdot T$ product too, the proposed design gives the best results compared to all existing designs. The software implementation results of 128-bit secure pairings computed over different elliptic curves are enlisted in Table 6. The most efficient software for computing 128-bit pairings

on supersingular elliptic curves over $\mathbb{F}_{2^{1223}}$ is proposed in [7]. It takes $3.08ms$ on eight parallel cores of a *core i7 2.8GHz* processor.

5 Conclusion

In this paper we have proposed an area and time optimized hybrid Karatsuba multiplier for $\mathbb{F}_{2^{1223}}$. Sufficient parallelism has been employed in the architecture for which we have achieved a high-speed η_T pairing cryptoprocessor. A common datapath for both non-reduced pairing and final exponentiation has been shared which reduces the overall logic cells in its FPGA implementation. The proposed design achieves a significant improvement with respect to two aspects of the design. It computes η_T pairing in characteristic-two field with higher security (128:105) in half area. On the other hand, it achieves eight times speedup and also provides the best $area \times time$ product among existing designs for computing 128-bit secure pairings.

References

1. Aranha, D.F., Beuchat, J.L., Detrey, J., Estibals, N.: Optimal Eta pairing on supersingular genus-2 binary hyperelliptic curves. Cryptology ePrint Archive, Report 2010/559, `http://eprint.iacr.org/`
2. Barke, E., Barker, W., Burr, W., Polk, W., Smid, M.: Recommendation for key management part 1: General (revised). National Institute of Standards and Technology, NIST Special Publication 800-57 (2007)
3. Barreto, P.S.L.M., Galbraith, S.D., ÓhÉigeartaigh, C., Scott, M.: Efficient pairing computation on supersingular Abelian varieties. Designs, Codes and Cryptography 42, 239–271 (2007)
4. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
5. Beuchat, J.L., Detrey, J., Estibals, N., Okamoto, E., Henríguez, F.R.: Fast architectures for the η_T pairing over small-characteristic supersingular elliptic curves. IEEE Transactions on Computers 60(2) (2011)
6. Beuchat, J.-L., González-Díaz, J.E., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 21–39. Springer, Heidelberg (2010)
7. Beuchat, J.L., Trejo, E.L., Ramos, L.M., Mitsunari, S., Henríquez, F.R.: Multi-core Implementation of the Tate Pairing over Supersingular Elliptic Curves. Cryptology ePrint Archive, Report 2009/276 (2009), `http://eprint.iacr.org/`
8. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
9. Duquesne, S., Guillermin, N.: A FPGA pairing implementation using the residue number system. Cryptology ePrint Archive, Report 2011/176 (2011), `http://eprint.iacr.org/`

10. 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)
11. Fan, J., Vercauteren, F., Verbauwhede, I.: Efficient Hardware Implementation of $\mathbb{F}_p$-arithmetic for Pairing-Friendly Curves. IEEE Trasaction on Computers (to appear, 2011)
12. 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)
13. Galbraith, S.: Pairings. In: Blake, I.F., Seroussi, G., Smart, N.P. (eds.) Advances in Elliptic Curve Cryptography. London Mathematical Society Lecture Note Series, vol. ch. IX, Cambridge University Press, Cambridge (2005)
14. 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)
15. Grabher, P., Großschädl, J., Page, D.: On software parallel implementation of cryptographic pairings. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 35–50. Springer, Heidelberg (2009)
16. Hankerson, D., Menezes, A., Scott, M.: Software implementation of pairings. Cryptology and Information Security Series, ch. 12, pp. 188–206. IOS Press, Amsterdam (2009)
17. Henríquez, F.R., Koç, Ç.K.: On fully parallel Karatsuba multipliers for $GF(2^m)$. In: International Conference on Computer Science and Technology CST 2003, pp. 405–410 (2003)
18. Hess, F., Smart, N.P., Vercauteren, F.: The Eta pairing revisited. IEEE Transactions on Information Theory 52(10), 4595–4602 (2006)
19. Hoffstein, J., Pipher, J., Silverman, J.H.: An introduction to mathematical cryptography. Springer, Heidelberg (2008)
20. 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)
21. 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)
22. Karatsuba, A., Ofman, Y.: Multiplication of Multidigit Numbers on Automata. Soviet Physics Doklady (English Translation) 7(7), 595–596 (1963)
23. Lee, E., Lee, H.S., Park, C.M.: Efficient and generalized pairing computation on abelian varieties. Cryptology ePrint Archive, Report 2009/040 (2009), `http://eprint.iacr.org/`
24. Lenstra, A.: Unbelievable security: Matching AES security using public key systems. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 67–86. Springer, Heidelberg (2001)
25. Miller, V.S.: The Weil pairing, and its efficient calculation. Journal of Cryptology 17, 235–261 (2004)
26. Naehrig, M., Niederhagen, R., Schwabe, P.: New software speed records for cryptographic pairings. Cryptology ePrint Archive, Report 2010/186, `http://eprint.iacr.org/`
27. Rebeiro, C., Mukhopadhyay, D.: High speed compact elliptic curve cryptoprocessor for FPGA platforms. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 376–388. Springer, Heidelberg (2008)

28. Shu, C., Kwon, S., Gaj, K.: Reconfigurable computing approach for tate pairing cryptosystems over binary fields. IEEE Transactions on Computers 58(9), 1221–1237 (2009)
29. Weimerskirch, A., Paar, C.: Generalizations of the Karatsuba algorithm for efficient implementations. Cryptology ePrint Archive, Report 2006/224 (2006), http://eprint.iacr.org/

Appendix A

We describe 1223-bit multiplication based on *the serial use of 306-bit parallel multiplier* in Algorithm 2. The variable names of the algorithm are similar to the registers and intermediate results computed by the proposed 1223-bit multiplier as shown in Fig. 2.

Appendix B

The η_T Pairing on Supersingular Elliptic Curves over $\mathbb{F}_{2^{1223}}$. This paper considers the η_T pairing computed over characteristic two field $\mathbb{F}_{2^{1223}}$, which is represented as $\mathbb{F}_2[x]/(x^{1223} + x^{255} + 1)$ in the polynomial basis with irreducible polynomial $(x^{1223}+x^{255}+1)$. The supersingular elliptic curve E over above field is defined as:

$$E/\mathbb{F}_{2^{1223}} : Y^2 + Y = X^3 + X, \tag{6}$$

which has embedding degree $k = 4$. It forms a large subgroup with prime order $r = (2^{1223}+2^{612}+1)/5$. The η_T pairing on $E/\mathbb{F}_{2^{1223}}$ attains 128-bit security level because Pollard's rho method for computing discrete logarithms in above order-r subgroup has running time at least 2^{128}, as do the index-calculus algorithms for computing discrete logarithms in the extension field $\mathbb{F}_{(2^{1223})^4}$. We refer the reader to [3,18,24] for more details about the computation techniques of η_T pairing and its respective security. We represent the extension field $\mathbb{F}_{(2^{1223})^4}$ using tower field extensions $\mathbb{F}_{(2^{1223})^2} = \mathbb{F}_{2^{1223}}[u]/(u^2+u+1)$ and $\mathbb{F}_{(2^{1223})^4} = \mathbb{F}_{(2^{1223})^2}[v]/(v^2+v+u)$, where a basis for $\mathbb{F}_{(2^{1223})^4}$ over $\mathbb{F}_{2^{1223}}$ is [1, u, v, uv].

Algorithm 2 . The 1223-bit multiplication based on Karatsuba technique[†].

Input: $a = \sum_{i=0}^{1222} a_i x^i$ and $b = \sum_{i=0}^{1222} b_i x^i$.
Output: $a \cdot b$.

1. $a_{00} \leftarrow \sum_{i=0}^{305} a_i x^i$; $a_{01} \leftarrow \sum_{i=306}^{611} a_i x^i$;
2. $a_{10} \leftarrow \sum_{i=612}^{917} a_i x^i$; $a_{11} \leftarrow \sum_{i=918}^{1222} a_i x^i$;
3. $b_{00} \leftarrow \sum_{i=0}^{305} b_i x^i$; $b_{01} \leftarrow \sum_{i=306}^{611} b_i x^i$;
4. $b_{10} \leftarrow \sum_{i=612}^{917} b_i x^i$; $b_{11} \leftarrow \sum_{i=918}^{1222} b_i x^i$;
5. $g_0 \leftarrow a_{00} + a_{01}$; $g_1 \leftarrow a_{10} + a_{11}$; $g_2 \leftarrow a_{00} + a_{10}$; $g_3 \leftarrow a_{01} + a_{11}$;
6. $h_0 \leftarrow b_{00} + b_{01}$; $h_1 \leftarrow b_{10} + b_{11}$; $h_2 \leftarrow b_{00} + b_{10}$; $h_3 \leftarrow b_{01} + b_{11}$;
7. $g_4 \leftarrow g_2 + g_3$; $h_4 \leftarrow h_2 + h_3$;
8. $k \leftarrow a_{00} \cdot b_{00}$;
9. $d_1 \leftarrow k_L$; $d_0 \leftarrow k_R$;
10. $k \leftarrow a_{01} \cdot b_{01}$;
11. $d_3 \leftarrow k_L$; $d_2 \leftarrow k_R$;
12. $k \leftarrow g_0 \cdot h_0$;
13. $t_1 \leftarrow k_L$; $t_0 \leftarrow k_R$;
14. $d_1 \leftarrow d_1 + d_0 + d_2 + t_0$; $d_2 \leftarrow d_2 + d_1 + d_3 + t_1$;
15. $k \leftarrow a_{10} \cdot b_{10}$;
16. $e_1 \leftarrow k_L$; $e_0 \leftarrow k_R$;
17. $k \leftarrow a_{11} \cdot b_{11}$;
18. $e_3 \leftarrow k_L$; $e_2 \leftarrow k_R$;
19. $k \leftarrow g_1 \cdot h_1$;
20. $t_1 \leftarrow k_L$; $t_0 \leftarrow k_R$;
21. $e_1 \leftarrow e_1 + e_0 + e_2 + t_0$; $e_2 \leftarrow e_2 + e_1 + e_3 + t_1$;
22. $k \leftarrow g_2 \cdot h_2$;
23. $f_1 \leftarrow k_L$; $f_0 \leftarrow k_R$;
24. $k \leftarrow g_3 \cdot h_3$;
25. $f_3 \leftarrow k_L$; $f_2 \leftarrow k_R$;
26. $k \leftarrow g_4 \cdot h_4$;
27. $t_1 \leftarrow k_L$; $t_0 \leftarrow k_R$;
28. $f_1 \leftarrow f_1 + f_0 + f_2 + t_0$; $f_2 \leftarrow f_2 + f_1 + f_3 + t_1$;
29. $r_0 \leftarrow d_0$; $r_1 \leftarrow d_1$; $r_2 \leftarrow d_2 + d_0 + e_0 + f_0$;
30. $r_3 \leftarrow d_3 + d_1 + e_1 + f_1$; $r_4 \leftarrow e_0 + d_2 + e_2 + f_2$;
31. $r_5 \leftarrow e_1 + d_3 + e_3 + f_3$; $r_6 \leftarrow e_2$; $r_7 \leftarrow e_3$;
32. **return** $(r_7 \cdot x^{2142} + r_6 \cdot x^{1836} + r_5 \cdot x^{1530} + r_4 \cdot x^{1224} + r_3 \cdot x^{918} + r_2 \cdot x^{612} + r_1 \cdot x^{306} + r_0)$.

[†] In the algorithm, k_R represents the least significant m bits of $(2m-1)$-bit result k, and k_L represents the most significant $m-1$ bits of k.