Flexible and Parallel Architectures for Optimal Ate pairing on FPGA

Oussama Azzouzi *

Ecole nationale Supérieure d'Informatique,
Laboratoire des Méthodes de Conception des Système,
BP 68M, 16309, Oued-Smar, Alger, Algérie
o_azzouzi@esi.dz

Mohamed Anane

Ecole nationale Supérieure d'Informatique,
Laboratoire des Méthodes de Conception des Systèmes,
BP 68M, 16309, Oued-Smar, Alger, Algérie.
m_anane@esi.dz

Mouloud Koudil

Ecole nationale Supérieure d'Informatique,
Laboratoire des Méthodes de Conception des Systèmes,
BP 68M, 16309, Oued-Smar, Alger, Algérie.
m_anane@esi.dz

Mohamed Issad

Department of System and Multimedia Architecture,
Centre de Développement des Technologies Avancées,
BP. 17 Cite 20 Aout 1956
Baba Hassen, 16081, Algiers, Algeria
missad@cdta.dz

Abstract. This paper presents three approaches for the implementation of Optimal Ate pairing based on Jacobean coordinates, over Barreto-Naehrig curves, targeting the 128 bits security level, in Genesys board. The first approach is a fully software implementation using MicroBlaze processor. The second approach is software/hardware implementation, in which the most useful operations in F_p and F_{p^2} are coded as intellectual property cores around the Microblaze. The third approach is based on the second in which we exploit the parallelism that exists to compute Optimal Ate pairing. The integration of multi-MicroBlaze processor in single architecture allows not only the flexibility of the overall system but also the parallelism to speed up pairing. Various techniques and parameters are used and combined to compute Optimal Ate in efficient way, namely: Montgomery modular multiplication, Karatsuba method, Jacobean coordinate, Complex method for squaring, Sparse multiplication, squaring in the cyclotomic subgroup $G_{\phi 6}(F_{p^{12}})$ and addition chain method. Our flexible and parallel systems are dedicated for restricted environment resources with a reasonable execution time.

Keywords- Flexible design, Optimal Ate, Montgomery modular multiplication, Karatsuba method, Virtex-5, MicroBlaze.

1. Introduction

Pairing functions are mathematical tools introduced by André Weil in 1948. The use of bilinear pairings on elliptic curves was introduced for the first time in cryptography for the purpose of cryptanalysis. The property of bilinearity allows the transfer of the discrete logarithm problem (DLP) from an elliptic curve E to a finite field F_p. This introduces the MOV attack (Menezes, Okamoto and Vanstone) [18] and Frey-Rück attack [19]. However, the constrictive use of pairings in cryptography appears from the 2000s. Joux [20] proposed the tripartite key exchange scheme for Diffie-Hellman. After that, cryptologists show that the pairings could be used to invent new protocols. Indeed, the Identity Based Encryption (IBE) [21] proposed by Boneh and Franklin is without doubt the famous application used for pairings.

In general, the construction of pairing functions is based on Miller Loop [6] plus Final Exponentiation, whose performances depend on the arithmetic used in the basic field F_p and its extensions F_{p^k}. To execute pairings in an efficient way, several curves were discovered for providing better pairing computation as well as for achieving better security. We refer to Freeman and al. [22] for a taxonomy of pairing friendly curves. In current days, one of the best choices in terms of computational efficiency and security is Barreto-Naehrig (BN) curves [1]. Many articles propose protocols using pairings [20,21], others improve the computation of pairings [3,8] and few articles propose FPGA implementations to compute pairing functions [13,14]. Recently, researchers have shown an increased interest in the implementation of cryptographic pairings. In general, hardware implementations have been proved better approaches compared with the software developments. In 2010, the authors [15] presented the first hardware implementation of pairing functions over BN-curves achieving 128 bits security level. In 2011, Cheung et al. [23] gave two designs using the Residue Number System to enhance the execution time of optimal Ate pairing at 126-bit security. In 2012, Fan et al. [20] presented a hardware implementation of F_p-arithmetic for pairing. Then in 2013, Ghosh et al. [13] presented a fully hardware implementation of Ate and Optimal Ate pairing where all the F_{p^k}-arithmetic are coded in hardware. In 2015, Duquesne et al. [25] developed a new variant for computing the hard part of the final exponentiation with minimum consumption of resources. In 2018, Sghaier et al. [27] presented a novel high speed and efficient area optimal Ate pairing processor implementation over BN and Barreto-Lynn-Scott (BLS12) curves on FPGA.

In this paper, we propose three approaches for the implementation of Optimal Ate pairing based on Jacobean coordinates over BN-curves achieving 128 bits security level, using Virtex-5 circuit. The first approach is a fully software implementation on FPGA, using MicroBlaze processor. This kind of approaches represents the best option when flexibility is strongly desired. It is based generally on embedded software or hardware processors. The second approach is software/hardware implementation, in which we add an intellectual property (IP) core, coded in VHDL, around the Microblaze. This approach offers the trade-off between flexibility, area and speed. The flexibility constraint provides the possibility of an easier design modification due to the system update functionality, while leaving the hardware architecture fixed. The third approach is based on the second one in which we exploit the parallelism that exists at the level of critical operations in the purpose to speed up pairing computations. Various FPGA-based pairing implementations have been proposed in the literature. The key issues of such works are the enhancement of execution time. In order to achieve the best trade-off between flexibility, timing execution and area occupation, our paper shows flexible FPGA implementations that support various methods and parameter settings of pairings in which Montgomery modular multiplication, Karatsuba method, Complex method for squaring, Sparse multiplication, squaring in the cyclotomic subgroup $G_{\phi 6}(F_{p^{12}})$, Jacobean coordinate and addition chain method are combinatorially applied. Our main objective is the implementation of flexible and parallel pairing systems targeting minimum resource consumption with a reasonable execution time. The efficiency of our implementations is achieved by the combination of the mixed SW/HW approach and exploiting the parallelism in Optimal Ate pairing.

The paper is organized as follows. Section 2 gives a brief overview of Optimal Ate pairing over BN-curves with specific parameter selection. Section 3 presents the IP cores developed in VHDL. Section 4 describes three approaches for the implementation of Optimal Ate pairing as embedded cryptosystem on FPGA. The implementation results are analyzed and compared to previous work in Section 5. Finally, section 6 concludes the paper.

2. Optimal Ate Pairing over Barreto-Naerig Curves

Let G_1 and G_2 be two cyclic additive groups and let G_3 be a multiplicative group. A pairing $e(P,Q): G_1 \times G_2 \to G_3$ is a mathematic function that maps two points P and Q given on an elliptic curve E into an element of an extension field F_{p^k}. This function should satisfy the property of bilinearity and non-degeneracy. The most useful property derived from the bilinearity is: for $P \in G_1$, $Q \in G_2$, we have:

$$\forall j \in \mathbb{N} : e([j]P, Q) = e(P,Q)^j = e(P,[j]Q)$$

Barreto-Naehrig presented in [1] a method to generate pairing friendly ordinary elliptic curves over a prime field F_p. These curves are the most important family of curves for achieving 128 bit security level and providing better pairing computation. They are defined by the equation:

$$E:\ y^2 = x^3 + b \text{ where } b \neq 0$$

The BN-curves have an embedding degree $k = 12$. Moreover, the characteristic p of the prime field, the group order r and the trace of Frobenius t_r of these curves are defined by:

$$p(t) = 36t^4 + 36t^3 + 24t^2 + 6t + 1$$
$$r(t) = 36t^4 + 36t^3 + 18t^2 + 6t + 1$$
$$t_r(t) = 6t^2 + 1, \text{where } t \in \mathbb{Z}$$

The selection of such parameters has an important impact on the security and the performance of the pairing. The independent variable t is chosen such that both p and r are prime numbers. In addition, t must be large enough to achieve higher security level. For a security level equivalent to AES 128 bits, we should select t such that $log_2(r(t)) \geq 256$ and $3000 \leq k.log_2(p(t)) \leq 5000$ which corresponds to the recommendations of National Institute of Standards and Technology (NIST) [2]. Therefore, t should have roughly 64 bits.

Let $E[r]$ denote the r-torsion subgroup of E and π_p be the Frobenius endomorphism $\pi_p: E \to E$ given by $\pi_p(x,y) = (x^p, y^p)$. We defined: $G_1 = E(F_p)$, $G_2 \subseteq E(F_{p^{12}})$ and $G_3 = \mu_r \subset F^*_{p^{12}}$. The Optimal Ate pairing on the BN-curves can be shown by the map:

$$\begin{aligned} e_{opt}:\ & G_2 \times G_1 \to G_3 \\ (Q,P)\ & \mapsto \left(f_{s,Q}(P) \cdot f_{[s]Q,\pi_p(Q)}(P) \cdot f_{[s]Q+\pi_p(Q),-\pi_p^2(Q)}(P)\right)^{\frac{p^{12}-1}{r}} \end{aligned}$$

where $s = 6t + 2$. Algorithm 1 shows the overall procedure of Optimal Ate pairing [3] implemented in this design. This Algorithm uses the non-adjacent form (NAF representation). There are three major steps in this algorithm. The Miller Loop (lines 3-10) generates the value of $f_{s,Q}(P)$. Point additions with the Frobenius map of point Q is computed in (line 11-13). Finally, the Final Exponentiation is executed in line 14.

Algorithm 1. Optimal Ate pairing over BN-curves.

Input: $P \in G_1$ and $Q \in G_2$
Output: $a_{opt}(Q,P)$

```
1. Write $s = 6t + 2$ as $s = \sum_{i=0}^{L-1} s_i 2^i$, where $s_i \in \{-1,0,1\}$; $L = bitlength(s)$
2. $T \leftarrow Q, f \leftarrow 1$;
3. for $i = L - 2$ to 0 do
4.     $f \leftarrow f^2 . l_{T,T}(P); T \leftarrow 2T$;
5.     if $s_i = -1$ then
```

```
6.          $f \leftarrow f.l_{T,-Q}(P); T \leftarrow T - Q;$
7.      else if $s_i = 1$ then
8.          $f \leftarrow f.l_{T,Q}(P); T \leftarrow T + Q;$
9.      end if
10. end for
11. $Q_1 \leftarrow \pi_p(Q);\ Q_2 \leftarrow \pi_{p^2}(Q);$
12. $f \leftarrow f.l_{T,Q_1}(P);\ T \leftarrow T + Q_1;$
13. $f \leftarrow f.l_{T,-Q_2}(P);\ T \leftarrow T - Q_2;$
14. $f \leftarrow f^{(p^{12}-1)/r};$
15. return $f;$
```

The major operations used in Optimal Ate pairing are: Doubling and Addition steps (line 4, 6, 8, 12 and 13), Sparse multiplication [4] (line 4, 6, 8, 12 and 13) which represents a multiplication in $F_{p^{12}}$ with the second operand has half of the coefficients equal to zero, Frobenius operation (line 11), Squaring in the cyclotomic subgroup $G_{\phi 6}(F_{p^2})$ (line 14) and Final Exponentiation (line 14). Doubling and Addition steps are performed in F_{p^2}, and most of the operations are performed in $F_{p^{12}}$.

As we have: $k = 12 = 2^2.3$, we can construct the arithmetic in $F_{p^{12}}$, step by step in smaller extensions fields, with the polynomial irreducible $X^k - \beta$. Several advanced techniques can compute these extended field operations with much lower number of multiplications and squaring. Therefore, $F_{p^{12}} = F_p[X]/(X^k - \beta)$, it can be considered as a tower of extensions of degree 2 (quadratic extension) and 3 (cubic extension). Indeed, we represent $F_{p^{12}}$ using the same tower extension of [5], namely, we construct a quadratic extension, which is followed by a cubic extension and then by a quadratic one, using the following irreducible binomials:

$$F_{p^{12}} \longrightarrow^2 F_{p^6} \longrightarrow^3 F_{p^2} \longrightarrow^2 F_p$$

$$\begin{cases} F_{p^2} = F_p[u]/(u^2 - \beta),\ \ where\ \beta = -5 \\ F_{p^6} = F_{p^2}[v]/(v^3 - \xi),\ \ where\ \xi = u \\ F_{p^{12}} = F_{p^6}[w]/(w^2 - v) \end{cases}$$

An element $f \in F_{p^{12}}$ can be represented as $f = g + hw$, with $g, h \in F_{p^6}$, such that, $g = g_0 + g_1 v + g_2 v^2$ and $h = h_0 + h_1 v + h_2 v^2$ where $g_i, h_i \in F_{p^2}$ for $i = 0, 1, 2$. This towering scheme helps us to speed up pairing.

2.1. ***Miller Loop***

The most famous pairings such as Weil, Tate, Ate and Optimal Ate pairing are based on Miller algorithm [6]. This latter builds a rational function $f_{r,P}$ associated to the point P and evaluated at the point Q, where P and Q are two points of an elliptic curve E, with an iterative process using the "double and addition" method. The function $f_{r,P}$ is defined by its divisor:

$$Div(f_{r,P}) = r(P) - ([r]P) - (r-1)(P_\infty)$$

where r is an integer and P_∞ denotes the point at infinity. The computation of this function is done thanks to the equality of Miller:

$$f_{[i+j],P} = f_{[i]P}\, f_{[j]P}\, \frac{l_{[i]P,[j]P}}{v_{[i+j]P}}$$

where $l_{[i]P,[j]P}$ is the line passing through $[i]P$ and $[j]P$, and $v_{[i+j]P}$ is the vertical to E at $[i+j]P$. The efficiency of Miller Loop depends certainly on the bit length of s and also on its Hamming.

2.1.1. Doubling and tangent line equations

The formulas for $T = 2Q = (X_T, Y_T, Z_T)$ in Jacobian coordinates are defined as follows:

$$X_R = 9X_T^4 - 8Y_T Y_T^2$$
$$Y_R = 3X_T^2(4X_T Y_T - X_R) - 8Y_T^4$$
$$Z_R = 4X_T Y_T$$

Let $P \in E(F_p)$ be given in affine coordinates as $P = (x_p, y_p)$. The tangent equation line at T evaluated at P can be calculated as follows:

$$l_{T,T}(P) = (4Z_R Z_T^2 y_P) - (6X_T^2 Z_T^2 x_p)w + (6X_T^3 - 4Y_T^2)w^2 \in F_p^{12}$$

2.1.2. Addition and line equations

The formulas for addition $R = T + Q = (X_R, Y_R, Z_R)$ are defined as follows:

$$X_R = (2Y_Q Z_T^3 - 2Y_T)^2 - 4\left(X_Q Z_T^2 - X_T\right)^3 - 8(X_Q Z_T^2 - X_T)^2 X_T$$
$$Y_R = \left(2Y_Q Z_T^3 - 2Y_T\right)\left(4\left(X_Q Z_T^2 - X_T\right)^2 X_T - X_R\right) - 8Y_T(X_Q Z_T^2 - X_T)^3$$
$$Z_R = 2Z_T(X_Q Z_T^2 - X_T)$$

The line through T and Q evaluated at the point P is given by:

$$l_{T,Q}(P) = \left(4Z_T\left(X_Q Z_T^2 - X_T\right)y_p\right) - \left(4x_p\left(Y_Q Z_T^3 + Y_T\right)\right)w + (4X_Q\left(Y_Q Z_T^2 X_Q - Y_T\right) - 4Y_Q Z_T(X_Q Z_T^2 - X_T))w^2 \in F_p^{12}$$

After computing the Miller Loop, we must carry out an extra step which is called the Final Exponentiation, where the Miller Loop result must be risen to the power $\frac{p^k - 1}{r}$.

2.2. Final Exponentiation

Several methods can be used to compute the Final Exponentiation of algorithm 1. The most traditional way is to use the "square and multiply" method. This one takes a lot of time because the exponent $e = \frac{p^{12}-1}{r}$ is too large. For that, this exponent can be break down as:

$$e = \frac{p^{12} - 1}{r} = (p^6 - 1) . (p^2 + 1) . \frac{p^4 - p^2 + 1}{r}$$

To calculate the first part $f^{(p^6-1)(p^2+1)} \in F_{p^{12}}$ which is the easy part, we must rise f to the power p^6 and p^2 which represent simple conjugation and Frobenius operations. After that, the result becomes an element of the cyclotomic subgroup $G_{\phi 6}(F_{p^2})$. In literature, many methods are developed to calculate the hard part of the Final Exponentiation. In 2009 Scott and al. [8] proposed a new approach based on addition chain for computing the hard part. During this part, all the elements involved are in the cyclotomic subgroup $G_{\phi 6}(F_{p^{12}})$. This simplifies computations enormously, namely, the operation count of f^2 computation [7], and any future inversion $f^{-1} = f^{p^6+1} = g - hw$ can be implemented as a simple conjugation [4].

The addition chain method takes advantage of the fact that p and r have a polynomial representation in t. This can be used effectively to break down the hard part. This method describes a smart procedure that requires the calculation of ten temporary values, namely:

$$f^t, f^{t^2}, f^{t^3}, f^p, f^{p^2}, f^{p^3}, f^{(tp)}, f^{(t^2p)}, f^{(t^3p)}, f^{(t^2p^2)}$$

These necessary elements are used for the construction of a multiplication chain whose evaluation produces the Final Exponentiation f^e, by using this equation:

$$[f^p . f^{p^2} . f^{p^3}] . [1/f\ ]^2 . \left[\left(f^{t^2}\right)^{p^2}\right]^6 . [1/(f^t)^p\ ]^6 . \left[1/\left(f^t . \left(f^{t^2}\right)^p\right)\ \right]^{18} . \left[1/f^{t^2}\right]^{30} . \left[1/\left(f^{t^3} . f^{t^3}\right)^p\right)\ \right]^{36}$$

To rise an element to the power p, we can compute it by the application of Frobenius operation. Also, to rise an element to the power t, which takes a lot of time, we can use the “square and multiply” method. Finally, we use Fermat’s little theorem to perform modular inversion in F_p, by using this expression:

$$A^{-1} \equiv A^{p-2}\ mod\ p$$

3. IP Cores on FPGA

The computational costs of each operation to compute Optimal Ate pairing developed in this work are given in table 1. We use the following notations: $\{a, m, s, i : F_p\}$ and $\{a_2, m_2, s_2, i_2 : F_{p^2}\}$ for modular addition/ subtraction, multiplication, squaring and inversion. m_β: multiplication by constant in F_p.

Table 1. Computational costs of Optimal Ate pairing operations

	add/sub	Multiplication	squaring	Inversion
F_p	a	m	s	i
F_{p^2}	$a_2 = 2a$	$m_2 = 3m + m_\beta + 5a$	$s_2 = 2m + 2m_\beta + 5a$	$i_2 = 4m + m_\beta + 2a + i$
F_{p^6}	$3a_2$	$6m_2 + 2m_\beta + 15a_2$	$2m_2 + 3s_2 + 2m_\beta + 10a_2$	$9m_2 + 3s_2 + 4m_\beta + 5a_2 + i_2$
$F_{p^{12}}$	$6a_2$ (-)	$18m_2 + 7m_\beta + 60a_2$ (**21**)	$12m_2 + 6m_\beta + 45a_2$ (**64**)	$25m_2 + 9s_2 + 13m_\beta + 61a_2 + i_2$ (**1**)
$G_{\emptyset_6}(F_{p^2})$	$6a_2$ (-)	$18m_2 + 7m_\beta + 60a_2$ (-)	$6m_2 + 6m_\beta + 39a_2$ (**190**)	Conjugation (**6**)
Sparse multiplication	$13m_2 + 3m_\beta + 28a_2$ (**72**)			
Doubling and tangent line step	$3m_2 + 8s_2 + 25a_2 + 4m$ (**64**)			
Addition and line step	$7m_2 + 8s_2 + 25a_2 + 4m$ (**8**)			
Miller	$1978m_2 + 568s_2 + 606m_\beta + 6755a_2 + 296m$ (**1**)			
Exp « t »	$408m_2 + 386m_\beta + 3541a_2$ (**3**)			
Final Exponentiation	$1543m_2 + 9s_2 + 1300m_\beta + 11776a_2 + 80m + i_2$ (**1**)			
Optimal Ate	$3521m_2 + 577s_2 + 1906m_\beta + 18531a_2 + 376m + i_2$			

Most of pairing functions are based on Miller Loop and Final Exponentiation, which need F_{p^k} arithmetic operations. The number of required operations to compute Optimal Ate pairing is given in bold between two parentheses. This number is calculated using our software implementation of Optimal Ate by Java language [(1)].

In this work, all operations that need arithmetic in F_{p^6} and $F_{p^{12}}$ are transferred to the arithmetic in F_p and F_{p^2}, as it shown in table 1. Therefore, our reflection avoids the problem of routing where operations in F_{p^6} and $F_{p^{12}}$ are coded in a hardware way. For that, we choose to work in F_p and F_{p^2} in the purpose to have a minimum consumption of resources and to make our system more flexible. In other way, modular operations in both fields F_p and F_{p^2} are developed in VHDL as IP cores and piloted by MicoBlaze(s). In addition, any curves that need F_p and F_{p^2} arithmetic could use our IP cores, by configuring only the software part.

[1] https://github.com/oussamaEsi/Ate-and-Optimal-Ate-on-JAVA

3.1. MMM IP Core

Multiplication in base field F_p is the most important operation for computing a cryptographic pairing. It can be executed by several techniques. In this paper, we use the Montgomery modular multiplication (MMM) which represents an efficient algorithm to perform modular multiplication. This algorithm avoids the division by transforming the modulus reduction to a series of additions and right shifts. The MMM based on High Radix-r ($r = 2^n$) is defined by the following expression:

$$S_e = Mont(A, B) = (A \times B \times R^{-1})\ mod\ p$$

R is the Montgomery constant. Algorithm 2 represents the Montgomery modular multiplication in Radix-2^{32} [9]. This algorithm contains two nested loops (i) and (j). The outer loop (i) allows the computation of the q_i digits. The inner loop (j) introduces the digits $B[j]$ and $p[j]$ to compute the digits of the intermediate result $S[j-1]$. The final result S_e is obtained for $i = j = e$.

```
Algorithm 2. Radix-2^32 Montgomery modular multiplication
Inputs : $A = \sum_{i=0}^{e} A[i] \times 2^{i\times32},\ B = \sum_{i=0}^{e} B[i] \times 2^{i\times32}, p = \sum_{i=0}^{e} p[i] \times 2^{i\times32},$
Variables: $H_i = \sum_{j=0}^{e} H[j]_i \times 2^{j\times32}, H1_i = \sum_{j=0}^{e} H1[j]_i \times 2^{j\times32},$
           $C1_i = \sum_{j=0}^{e} C1[j]_i \times 2^{j\times32},\ H2_i = \sum_{j=0}^{e} H2[j]_i \times 2^{j\times32},$
           $C2_i = \sum_{j=0}^{e} C2[j]_i \times 2^{j\times32},\ c1_j = c2_j = c3_j = c4_j$
Pre-computed: $p' = -p[0]^{-1}\ mod\ 2^{32}$
Output : $S_e = \sum_{j=0}^{e} S[j]_e \times 2^{j\times32} = (A \times B \times R^{-1}) mod\ p$
Begin
1.  $S_0 = \sum_{j=0}^{e} S[j]_0 \times 2^{j\times32} = 0$
2.  $For\ i\ from\ 0\ to\ e$
3.      $C1[-1]_i = 0;\ C2[-1]_i = 0;$
4.      $c1_{-1} = c2_{-1} = c3_{-1} = c4_{-1} = 0;$
5.      $H[0]_i = S[0]_i + A[i] \times B[0]$
6.      $q_i = (H[0]_i \times p')\ mod\ 2^{32}$
7.      $For\ j\ from\ 0\ to\ e$
8.          $(C1[j]_i 2^{32}, H1[j]_i) = A[i] \times B[j]$
9.          $(c2_j 2^{32}, c1_j 2^{32}, H[j]_i) = H1[j]_i + C1[j-1]_i + S[j]_i\ c1_{j-1} + c2_{j-1}$
10.         $(C2[j]_i 2^{32}, H2[j]_i) = q_i \times p[i]$
11.         $(c4_j 2^{32}, c3_j 2^{32}, S[j-1]_i) = H[j]_i + H2[j]_i + C2[j-1]_i + c3_{j-1} + c4_{j-1}$
        $end\ for$
12.     $S[e]_i = c1_e + c2_e + c3_e + c4_e + c1[e]_i + c2[e]_i$
    $end\ for$
13. $return\ S_e$
End
```

In practice, it is necessary to convert each operand to its equivalent Montgomery form which costs another MM operation because each multiplication is computed with additional factor R^{-1}. However, for repeated multiplications used in a pairing computation, the operands are converted once at the beginning and converted back at the end.

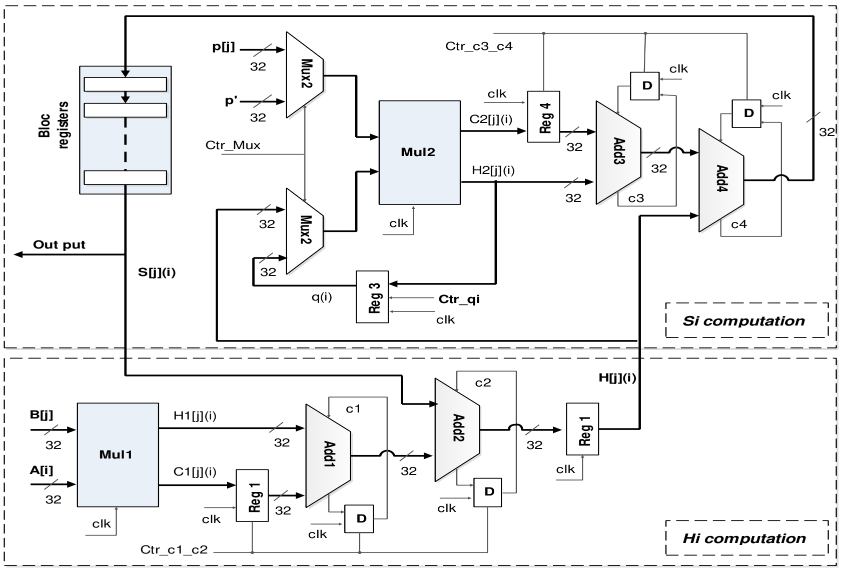

Fig. 1. Hardware architecture of MMM on FPGA

The hardware architecture of our MMM is shown in figure 1. It performs the arithmetic operations of algorithm 2. It contains two 32×32 bits multipliers (Mul1 and Mul2), four carry propagate adders (Add1, Add2, Add3 and Add4), four registers (Reg1, Reg2, Reg3, Reg4), four D Flip-flops, two multiplexers (Mux1, Mux2) and one bloc registers. The execution of the MMM core requires the storage of the operands A, B and p in memories. The intermediate results digits $S[j]_i$ of algorithm 2 are stored in the block register as queue. MMM receives four control signals, namely Ctr_Mux, Ctr_q_i, Ctr_c1_c2 and Ctr_c1_c2.

Our MMM performs each iteration (i) of algorithm 2 in three steps. In the first step, the process begins by the execution of lines 5 and 6 for computing the digit q_i. Its value is stored in Reg3 which is controlled by Ctr_q_i. Reg3 holds the value of q_i constant during the execution of the iterations (j) of algorithm 2. The second and the third steps consist of performing the computations of the multiplications of lines (8, 9) and (10, 11) of algorithm 2, respectively. They allows the computations of the digits $H[j]_i$ and $S[j]_i$. The multiplier Mul2 is shared between the execution of the multiplications of lines 6 and 10 of algorithm 2.

3.2 KARATSUBA IP Core

Arithmetic in F_{p^2} contains modular addition, subtraction, multiplication, squaring, multiplication by a constant, reduction and inversion, where all operators are presented by two numbers in F_p. Using traditional method, modular multiplication in F_{p^2} costs at least four multiplications and five additions/subtractions in F_p. It can be optimized by parallel computation. For that, we proposed the algorithm 3. This later costs only two multiplications plus two additions/subtractions in F_p.

Algorithm 3. Parallel Karatsuba method in F_{p^2}

```
Inputs : $A = a_0 + a_1 u, B = b_0 + b_1 u$
Output : $C = c_0 + c_1 u$

Begin
1. $t_0 \leftarrow a_0 * b_0,\ \ t_1 \leftarrow a_1 * b_1,\ \ tmp \leftarrow b_0 + b_1,\ \ c_1 \leftarrow a_0 + a_1$
2. $c_1 \leftarrow c_1 * tmp,\ \ c_0 \leftarrow t_1 * redFp$
3. $c_0 \leftarrow c_0 - t_0,\ \ c_1 \leftarrow c_1 - t_0$
4. $c_1 \leftarrow c_1 - t_1$
End
```

The second IP core developed in this work is called KARATSUBA. It allows us to carry out modular operations in F_{p^2} which are multiplication, squaring, multiplication by a constant and reduction. In addition, it can be used to perform Montgomery modular multiplication in F_p. The hardware architecture of KARATSUBA is shown in figure 2. It is performed in four steps. For each step, the appropriate IP(s) are selected using the control circuit. It contains two MMM and ADD/SUB IP VHDL cores. This latter is used to calculate the modular addition/subtraction in F_p.

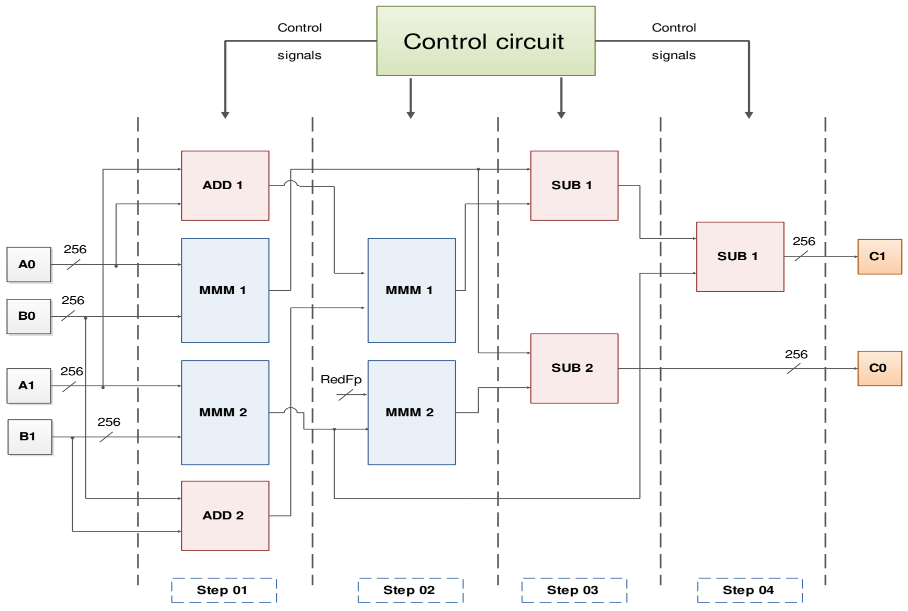

Fig. 2. Hardware architecture of KARATSUBA on FPGA

4. Proposed architectures for Optimal Ate pairing

In this work, three approaches for the hardware architectures of Optimal Ate pairing as embedded system on FPGA are proposed. In the following, we present our hardware architectures.

4.1. Signal MicroBlaze-based software implementation

The first approach represents a fully software implementation of Optimal Ate on Genesys board. In this approach, all functions and operations to calculate Optimal Ate are stored in memory (BRAM) and executed in sequential way by the MicroBlaze.

The MicroBlaze processor is Xilinx's 32-bit RISC soft core, optimized for embedded applications. It can be used on any Xilinx or Xilinx's Partner development board. It offers basic operations such as addition, subtraction and multiplication. All operands used to perform Optimal Ate pairing are coded in packets of 32 bits.

In order to perform Optimal Ate pairing, MicroBlaze executes a C program, using SDK tools. The software architecture is designed in four abstraction levels as it shown in figure 3. The top level comprises the pairing function. The second level represents the Miller algorithm and Final Exponentiation. The third level consists of Doubling step, Addition step and Frobenius function. Finally, Frobenius operations, finite field addition, subtraction, multiplication, division and exponentiation in F_p, F_{p^2}, F_{p^6} and $F_{p^{12}}$ are in the fourth level.

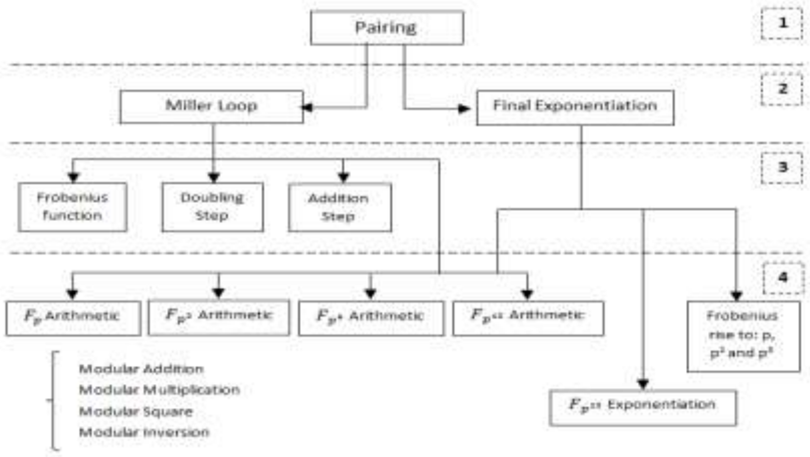

Fig. 3. Optimal Ate pairing implementation hierarchy

Most of operations used in the towering scheme $F_{((p^2)^2)^3}$ are performed either in quadratic (F_{p^2}) and cubic (F_{p^3}) extension fields. Various techniques for computing multiplication and squaring in such fields are explained in [10]. In this work, we use Karatsuba method for multiplication and complex method for squaring in F_{p^2}. Whereas, we follow the Karatsuba method for computing both multiplication and squaring in F_{p^3}.

Figure 4 shows the hardware architecture for the software implementation of Optimal Ate on Virtex-5 circuit around the MicroBlaze processor. This architecture contains the following components: MicroBlaze, BRAM memory, Local Memory Buses (ILMB, DLMB) which allow the processor to control the BRAM, Timer to evaluate the execution time and Universal Asynchronous Receiver Transmitter (UART) to exchange the inputs and the outputs data with the serial port.

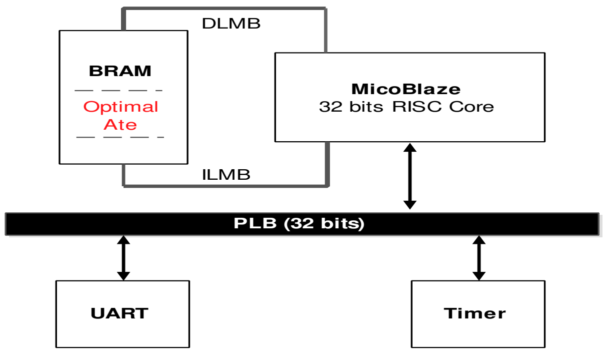

Fig. 4. Hardware architecture of 1Mb approach

4.2. Single MicroBlaze-based SW/HW implementation

The second approach presents a software/hardware design of Optimal Ate pairing, over BN-curves on Virtex-5 circuit. An accelerator IP core is implemented around the MicroBlaze in order to speed up the overall execution time.

The first design based on this approach use an MMM core to perform all the required MMM operations, e.g., an optimal Ate pairing defined on a 256-bit BN curve requires around 10^4 modular multiplications [26]. Figure 5 presents the hardware architecture of the corresponding embedded system.

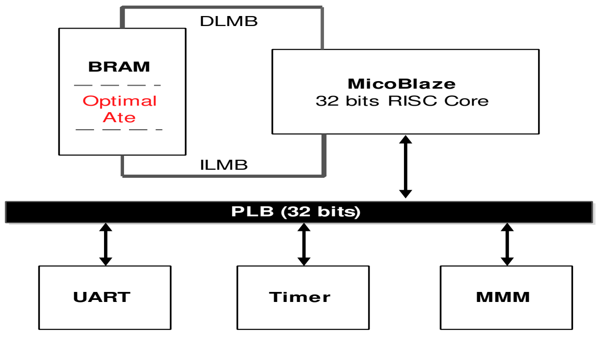

Fig. 5. Hardware architecture of 1Mb/1MMM approach

The second design based on the hardware/software approach use a KARATSUBA core around the MicroBlaze to perform all the required operations in F_p and F_{p^2}. Figure 6 presents the hardware architecture of the corresponding embedded system.

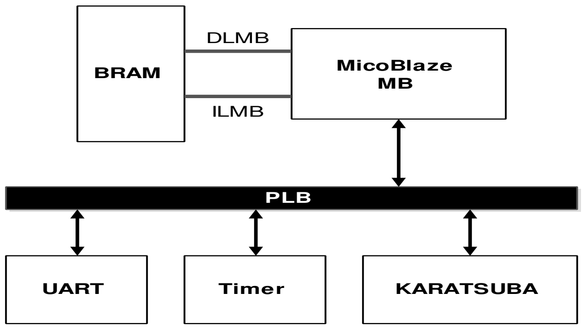

Fig. 6. Hardware architecture of 1Mb/1KARATSUBA approach

The proposed software/hardware partitioning allows not only the execution time improvement, but also the overall embedded system flexibility since that the upper level functions are implemented in software way. However, the overall execution time is also influenced by an essential parameter which is the data transfer time between the MicroBlaze and the IP core.

The global architecture of our IP cores for its integration around the processor MicroBlaze is shown in figure 7. The IP is connected to Microblaze through Xilinx PLB Bus. This bus allows the data/instruction exchanging between Microblaze and the IP core. The overall architecture is composed by the Xilinx's Intellectual Property InterFace (IPIF) and User_Logic blocks. The communication between the two blocks is achieved by a standard back-end interface IP InterConnect (IPIC) [11].

The IPIF interface aims to decode the communication protocol of the PLB system bus. It is configured with three registers: Ins_reg, DataIn_reg and DataOut_reg. Microblaze control the IP core using a set instruction codes through Ins_reg instruction register. User_Logic is the part that describes the logic of the circuit. It is composed by three units, Memory Unit (MU), Control Unit (CU) and the IP core. CU retrieves the instructions from Ins_reg and manages MU and IP cores.

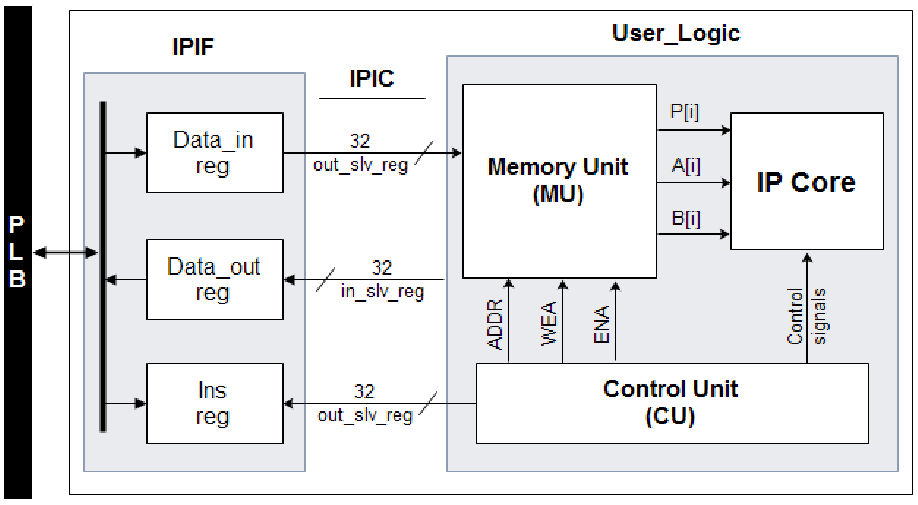

Fig. 7. Hardware architecture of our IP cores

4.3. Dual MicroBlaze-based SW/HW implementation

The third approach consists of exploiting the parallelism that exists at the level of critical operations such as modular multiplication in F_{p^6} and $F_{p^{12}}$, Sparce multiplication, Squaring in the cyclotomic subgroup $G_{\phi 6}(F_{p^2})$, Doubling and Addition steps.

Several configurations of software/hardware architectures for the implementation of Optimal Ate pairing could be performed. In this work, we try to choose the one that ensures our initial hypothesis, i.e. the design that have a minimum consumption of memory with a reasonable execution time. Figure 8 presents the hardware architecture for a parallel and flexible implementation of Optimal Ate pairing. It is composed by two MicroBlaze processors and IP KARATSUBA core. The processors MB_0 and MB_1 are connected through FSL Bus. MB_0 is considered as master processor and MB_1 acts as slave processor. This later is connected to KARATSUBA core through the PLB bus and is responsible for all operations in F_p and F_{p^2}. The data transfer time between two processors through the FSL link is very fast. That means the transfer time through the FSL link is negligible compared to the one used through the PLB bus. For this reason, we favorite the addition of embedded processors with FSL links instead of adding IP cores through the PLB bus.

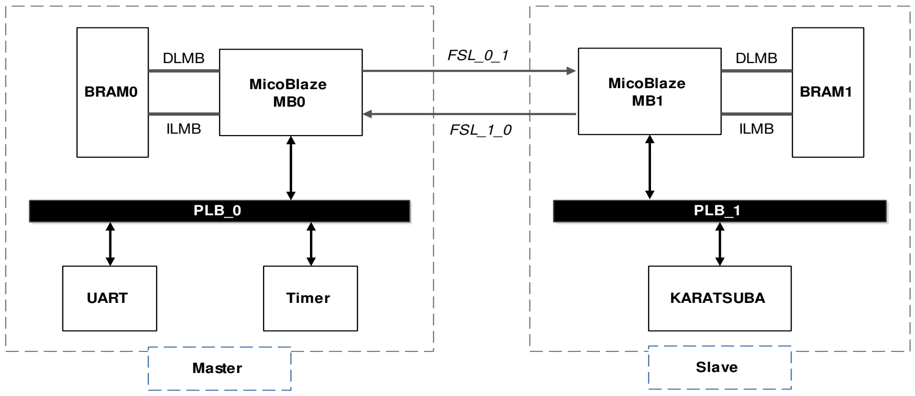

Fig. 8. Hardware architecture of 2Mb/1KARATSUBA approach

5. Implementation and Result Analysis

The whole design of the proposed embedded systems for Optimal Ate pairing computation were implemented on Virtex-5 “XC5VLX50T” Genesys development Xilinx board [12] using XPS 13.2 (Xilinx Platform Studio) environment. Our MMM and KARATSUBA were coded in VHDL language. Then, the code was simulated using the Modelsim SE 64 10.0c and synthesized using Xilinx ISE Design Suite 13.2. The DSP48E cores and the RAM blocks of the IPs were generated by Core Generator tool of ISE. High level arithmetic of Optimal Ate pairings were developed by C language in SDK.

We choose to use the parameter $t = 2^{62} - 2^{54} + 2^{44}$ [4] together with the BN-curve $E: y^2 = x^3 + 5$ to guarantee the proposed design 128 bit security level. Therefore, the exponent number t and the parameter $s = 6t + 2$ in Miller Loop have only 63 and 65 signed bit length, respectively.

Table 2 presents a comparison of our results based on four hardware architectures with recent implementations of Optimal Ate pairing based on BN-curves, achieving 128 bits security level in different platforms. The comparison includes the execution time, the required hardware and the design efficiency. This later is calculated by the expression:

$$efficiency = \frac{datapath\ (bit)}{occupied\ area\ (slice) * execution\ time(s)}$$

The occupied area is calculated based on the following note: assume that the height of a DSP48E is the same as a five configurable logic blocks (CLBs) and also match the height of one block RAM. The CLB is composed by four slices.

The software design executed by MicroBlaze has the higher speed comparing to the other designs. The SW/HW design comes to improve the execution time with an augmentation of area. However, the data transfer time between the MicroBlaze and the IP core influence on the overall execution time. Exploiting the parallelism that exists in Optimal Ate pairing presents an excellent idea in the purpose to improve the global execution time with a minimum consumption of area.

Table 2. Optimal Ate pairing implementation results comparison

Ref.	**Platform**	**Freq. MHz**	**Design**	**Area**			**Cycles**	**Times** (ms)	**efficiency**
				Slices	**DSP**	**RAM**			
Our	Virtex-5	125	**SW** Mb [(2)]	1063	3	32	262445486	2099.56	0.07
		100	**SW/HW** 1Mb/1MMM [(3)]	1558	11	35	26551593	265.51	0.38
		100	**SW/HW** 1Mb/1KARATSUBA [(4)]	3027	31	38	12642367	126.42	0.45
		100	**SW/HW** 2Mb/1KARATSUBA	4029	34	40	4035686	40.35	1.15
[17]	Virtex-5	125	**SW**	1063	3	32	348793411	2790.34	0.05
[15]	Virtex-4	50	**HW**	52000	-	-	821000	16.42	0.29
[13]	Viretx-6	145	**HW**	23000	-	-	821000	5.66	1.96
[14]	Virtex-5	125	**HW**	10592	51	-	283111	2.26	9.75

In our previous work [17], we have proposed a fully software implementation of Weil, Tate, Ate and Optimal Ate using Jacobean coordinate, over BN-curve on Virtex-5 and Zynq platform. This work has a significant improvement in terms of execution time, about 25% compared to the previous. This is due to the use of Sparse multiplication method and squaring in $G_{\phi 6}F_{p^{12}}$.

[2] https://github.com/oussamaEsi/Weil-Tate-Ate-and-Optimal-Ate-on-Virtex-5

[3] https://github.com/oussamaEsi/1MB-1MMM

[4] https://github.com/oussamaEsi/KARATSUBA-

In [15], the authors present the first hardware implementation of pairing functions over BN-curves achieving 128 bits security level. They use Blakley's algorithm to compute modular multiplication. They have the higher area occupation without using DSP and RAM cores. Our SW/HW designs achieve better consumption of slices. In addition, they provide less efficiency.

Ref [13] present a fully hardware implementation of Ate and Optimal Ate pairing where all the F_{p^k}-arithmetic are coded in hardware. This design uses $23k$ logic slices without any use of DSP and RAM blocks. Compared to 2Mb/1KARATSUBA design, the proposed design is 7.2 times faster in terms of time performance. However, it consumes 4.17 times more slices. This system presents a higher area occupation and isn't suitable for restricted environments.

The authors from [14] propose a hardware cryptoprocessor of Optimal Ate pairing. The proposed architecture consists of two processing engines which compute F_p-arithmetic in parallel. Each engine use Montgomery algorithm to compute modular multiplication with specific add-subtraction unit. This design has a reasonable increase of area with higher number of DSP blocks. This constraint limits the integration possibility of the proposed processor only to the boards with high resources FPGAs, whereas our designs could be highly exploited in large FPGA circuits. Although, their processor provides better efficiency than our implementations, but it lacks flexibility.

6. Conclusion

In this paper, we have presented three approaches for the implementation of Optimal Ate pairing based on Jacobean coordinates over BN-curves at the 128-bits security level, as embedded cryptosystem exploring FPGA features devices. The first approach is completely software implementation where all operations are performed by MicroBlaze processors. The second approach is based on software/hardware codesign, where the useful operation in F_p and F_{p^2} have implemented in hardware around the MicroBlaze. To improve the execution time with minimum area consumption, the third approach consists of exploiting the parallelism at the level of critical operations in Optimal Ate pairing. Various methods and techniques are used and combined to implement flexible pairing systems such as Montgomery modular multiplication, Karatsuba method, Complex method for squaring, Sparse multiplication, squaring in the cyclotomic subgroup $G_{\phi 6}(F_{p^{12}})$, Jacobean coordinate and addition chain method. Our flexible and parallel systems can be used for restricted environment with a reasonable execution time.

Data availability statement

All data generated or analysed during this study are included in this published article. All data and source codes for pairing functions have been deposited in GitHub (https://github.com/oussamaEsi).

Conflicts of interest statement

The authors certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

[1] Barreto PSLM, Naehrig M (2006) Pairing-friendly elliptic curves of prime order. in SAC'05 LNCS 3897. pp 319–331.

[2] Barker E, Barker W, Burr W, Polk W, Smid M (2007) Recommendation for key management part 1: General (revised). In: Published as NIST Special Publication 800-57

[3] Vercauteren F (2010) Optimal pairings. IEEE Trans Inf Theory vol 56, no 1, pp 455–461

[4] Beuchat JL, Díaz JEG, Mitsunari S, Okamoto E, Henríquez FR, Teruya T (2010) High-speed software implementation of the optimal ate pairing over Barreto–Naehrig Curves. In: International Conference on Pairing-Based Cryptography, pp 21–39

[5] Hankerson D, Menezes A, Scott M (2008) Software implementation of pairings. In Identity-Based Cryptography, M Joye and G Neven, Eds Amsterdam, The Netherlands: IOS Press

[6] Miller V (2004) The Weil pairing and its efficient calculation. J Cryptology, vol 17, pp 235–261

[7] Granger R, Scott M (2010) Faster squaring in the cyclotomic subgroup of sixth degree extensions. PKC '10, LNCS 6056, pp 209–223

[8] Scott M, Benger N, Charlemagne M, Dominguez PLJ, Kachisa EJ (2009) On the final exponentiation for calculating pairings on ordinary elliptic curves. Pairing '09, LNCS 5671, pp 78–88

[9] Issad M, Boudraa B, Anane M, Anane N (2014) Software/Hardware Co-Design of Modular Exponentiation for Efficient RSA Cryptosystem. Journal of Circuits, Systems and Computers, vol 23, no 3

[10] Devegili A, OhEigeartaigh C, Scott M, Dahab R (2006) Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, Tech Rep 2006/471

[11] PLB IPIF (v2.02a) DS448, 2005

[12] Genesys™ Board, Reference Manual, Revision, 2013

[13] Ghosh S, Mukhopadhyay D, Roychowdhury D (2013) Secure dual-core cryptoprocessor for pairings over Barreto-Naehrig Curves on FPGA Platform. IEEE Trans. Very Large Scale Integr. Syst, vol 21, no 3

[14] Hao Z, Guo W, Wei J, Sun D (2016) Dual Processing Engine Architecture to Speed Up Optimal Ate Pairing on FPGA Platform. IEEE Trustcom/BigDataSE/ISPA, Tianjin, pp 584-589

[15] Ghosh S, Mukhopadhyay D, Chowdhury D R (2010) High Speed Flexible Pairing Cryptoprocessor on FPGA Platform, presented at the DASIP'10, Scotland U.K

[16] Fan J, Vercauteren F, Verbauwhede I (2009) Faster IFp-Arithmetic for Cryptographic Pairings on Barreto-Naehrig Curves. CHES '09, LNCS 5747, pp 240–253

[17] Azzouzi O, Anane M, Haddam N (2018) Software Implementation of Pairing Based Cryptography on FPGA. CSA'18, Springer LNNS

[18] Menezes A, Okamoto T, Vanstone S (1991) Reducing elliptic curve logarithms to logarithms in a finite field. in Proc STOC'91, pp 80-89

[19] Frey G, Rück H G (1994) A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of groups. In Mathematics of Computation, vol 62, pp 865-874

[20] Joux A (2000) A one round protocol for tripartite Diffie-Hellman. in Proc ANTS, pp 385–394

[21] Boneh D, Franklin M K (2001) Identity-Based Encryption from the Weil Pairing. in CRYPTO 2001, LNCS 2139, pp 213–229

[22] Freeman D, Scott M, Teske E (2010) A taxonomy of pairing-friendly elliptic curves. In Jour-nal of Cryptology 23, pp 224-280

[23] Cheung R, Duquesne S, Fan J, Guillermin N, Verbauwhede I, Yao G (2011) Fpga implementation of pairings using residue number system and lazy reduction. Proc13th Int Workshop CHES, pp 421–441

[24] Fan J, Verbauwhede FVI (2012) Efficient hardware implementation of f_p-arithmetic for pairing-friendly curves. IEEE Trans. Comput. 61 (5) 676–685

[25] Duquesne S, Ghammam L (2016) Memory-saving computation of the pairing final exponentiation on bn curves. Groups Complexity Cryptol. 8 (1) 75–90

[26] Aranha D, Karabina K, Longa P, Gebotys CH, Lóopez J (2011) Faster explicit formulas for computing pairings over ordinary curves. Advances in Cryptology EUROCRYPT of LNCS Springer Heidelberg 6632, pp 48–68

[27] Sghaier A, Zeghid M, Ghammam L, Duquesne S, Machhout M, Hassan YA (2018) High speed and efficient area optimal ate pairing processor implementation over BN and BLS12 curves on FPGA. Microprocessors and Microsystems, v 61, pp 227-241