High-performance Public-key Cryptoprocessor for Wireless Mobile Applications

Kazuo Sakiyama · Lejla Batina · Bart Preneel ·
Ingrid Verbauwhede

Published online: 3 October 2007
© Springer Science + Business Media, LLC 2007

Abstract We present a high-speed public-key cryptoprocessor that exploits three-level parallelism in Elliptic Curve Cryptography (ECC) over $GF(2^n)$. The proposed cryptoprocessor employs a Parallelized Modular Arithmetic Logic Unit (P-MALU) that exploits two types of different parallelism for accelerating modular operations. The sequence of scalar multiplications is also accelerated by exploiting Instruction-Level Parallelism (ILP) and processing multiple P-MALU instructions in parallel. The system is programmable and hence independent of the type of the elliptic curves and scalar multiplication algorithms. The synthesis results show that scalar multiplication of ECC over $GF(2^{163})$ on a generic curve can be computed in 20 and 16 μ s respectively for the binary NAF (Non-Adjacent Form) and the Montgomery method. The performance can be accelerated furthermore on a Koblitz curve and reach scalar multiplication of 12 μ s with the TNAF (τ -adic NAF) method. This fast performance allows us to perform over 80,000 scalar multiplications per second and to enhance security in wireless mobile applications.

Keywords public-key cryptosystem · elliptic curve cryptography(ECC) · binary fields · parallel processing

1 Introduction

It is challenging to implement high-performance Public-Key Cryptosystems (PKCs) in embedded devices such as mobile phones and portable RFID readers because they have limited silicon resources and a limited power budget. The idea of PKC was introduced in the mid 70's [1]. PKCs ensure a secure data transfer via an insecure channel without prior key exchanges in order to exchange some confidential data. The most widely-accepted PKCs are RSA [2] and ECC [3, 4]. Due to the fact that RSA requires a large integer for its private key (at least 1,024 bits), ECC is often preferred for a compact hardware implementation because much shorter key-lengths are needed. Blake, Seroussi and Smart argue in [5] that 160-bit ECC has the same security level as 1,024-bit RSA.

The performance of ECC (i.e. scalar multiplication) is several orders of magnitude lower than that of a secret-key cryptography such as AES [6]. However, some fast results are reported with ECC implementations recently, which shows a great potential for wider range of application using ECC, especially in the field of secure networks. For instance, RFID tags with ECC (ECC-RFID tags) are proposed by [7–9] to enhance the privacy in physical distribution management systems [10]. Such systems require a local network server that can authenticate thousands (or even millions) of the ECC-RFID tags quickly. The position of ECC in mobile network applications is illustrated in Fig. 1a.

K. Sakiyama (✉) · L. Batina · B. Preneel · I. Verbauwhede
Department Electrical Engineering—ESAT/SCD-COSIC,
Katholieke Universiteit Leuven/IBBT,
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium
e-mail: ksakiyam@esat.kuleuven.be

L. Batina
e-mail: lbatina@esat.kuleuven.be

B. Preneel
e-mail: preneel@esat.kuleuven.be

I. Verbauwhede
e-mail: iverbauw@esat.kuleuven.be

Figure 1 **a** Position of ECC in mobile network applications. **b** Various options in hardware ECC implementations

As other examples of potential applications using a high-speed PKC, electronic voting [11, 12] and private searching on streaming data [13] are presented for the purpose of preserving private information.

In comparison with the relatively simple RSA implementation, ECC has many options in designing the hardware architecture of the system. The main operation in ECC is the elliptic curve (EC) scalar multiplication (kP). It is performed by a sequence of point doublings and point additions which are generally implemented as a controller block (e.g. finite state machine: FSM) in a hardware design or as software in hardware/software co-design as shown in Fig. 1b. The computation sequence of scalar multiplication varies according to the type of the curve. For instance, Smart [14] showed that up to three field operations could be executed in parallel for the Hessian form of an elliptic curve. Several computation sequences are also introduced including the recommendation in IEEE P1363 [15] and FIPS 186-2 [16]. The selection of the sequences has a great impact on cost and performance because the number of modular operations and the required memory size differ according to the sequence.

ECC can be defined over $GF(p)$ or $GF(2^n)$. Namely, 160-bit ECC over $GF(p)$ is determined by a prime number p that satisfies $p \geq 2^{160}$ and ECC over $GF(2^n)$ is specified by an irreducible polynomial $P(x)$ whose degree, n is a prime that satisfies $n > 160$ (e.g. $n = 163$). The field size and the type of the underlying finite field are important parameters that determine the architecture of the data processing block (see Fig. 1b). Especially, finite fields of characteristic 2 or $GF(2^n)$ operations are suitable for hardware implementations due to the carry-free arithmetic.

Many interesting computation sequences for ECC over binary fields exist including the following sequences: (1) the binary NAF method [5], (2) the Montgomery method [17] and (3) the τ -NAF or TNAF method on a Koblitz curve [18]. In this sense, a domain specific programmable architecture is an attractive choice for an Elliptic Curve (EC) cryptoprocessor because it offers the equivalent performance as ASIC while maintaining the flexibility to support the wide range of options for EC scalar multiplication. As summarized in Table 1, we denote computation sequences of scalar multiplication as ECC, ECC_M and ECC_KC depending on the type of the curve and the scalar multiplication algorithm. Although further performance improvement is possible by applying the window method [5] for ECC and ECC_KC, this is out of scope of this paper. We discuss this in Section 5.

Our contribution in this paper deals with a cryptoprocessor architecture with three-level parallelism for ECC over a binary field. More precisely, Modular Arithmetic Logic Unit (MALU) is based on a digit-serial multiplier that can operate n -bit $\times$ d -bit modular multiplication in one cycle, where n is the field size and d is the digit size. The critical path delay is only dependent on the digit size and hence this can be considered as the first level parallelism in modular multiplications. Second parallelism is in the Parallelized MALU (P-MALU) that uses two MALUs in parallel and the third is in the instructions determined by the computation sequence of EC scalar multiplication. Namely, the proposed cryptoprocessor exploits ILP in an EC scalar multiplication algorithm and executes instructions on several parallelized datapaths, i.e. the P-MALUs. The design is synthesized with a 0.13- μ m CMOS technology to evaluate the performance of scalar multiplication. This research is of interest because a constant progress in public-key cryptography can remove performance bottleneck and enhance security in wireless mobile applications.

This paper is organized as follows. Section 2 lists some relevant previous work. Some mathematical background is explained in Section 3. The proposed architecture of our cryptoprocessor is given in Section 4. The main contribution of our work, i.e. a parallelized datapath and ILP are explained in detail. The

Table 1 Various methods to compute EC scalar multiplication

	Type of Curve	Scalar Multiplication Algorithm
ECC	Generic Curves	Binary NAF
ECC_M	Generic Curves	Montgomery method
ECC_KC	Koblitz Curves	TNAF method

implementation results are discussed in section 5. Section 6 concludes the paper.

2 Related work

This section lists some relevant previous work. As already mentioned, there is a considerable amount of work done on hardware implementations on FPGAs as well as ASIC implementations for ECC since Agnew et al. [19] reported the first result for performing the elliptic curve operations on hardware in 1989. The majority of hardware ECC implementations have been over binary fields. Gao et al. proposed an elliptic curve cryptosystem coprocessor with variable key sizes, which utilizes the internal SRAM/registers in an FPGA in [20]. In 2000 Orlando and Paar proposed a scalable elliptic curve processor architecture which operates over finite fields $\text{GF}(2^n)$ in [21]. Gura et al. [22] have introduced a programmable hardware accelerator for ECC over $\text{GF}(2^n)$, which can handle arbitrary field sizes up to 255. Satoh and Takano [23] present a dual field multiplier with high performance in both binary and prime fields. The throughput of an elliptic curve scalar multiplication is maximized by use of the Montgomery modular multiplier and an on-the-fly redundant binary converter. The biggest advantage of their design is in scalability in operand size and also flexibility between speed and hardware area. The work by Tenca and Koç [24] also introduces a scalable architecture for the computation of modular multiplication, based on the Montgomery multiplier. Their proposed multiplier works with any precision of the input operands, limited only by memory or control constraints. Andres et al. [25] improve the version of the Tenca-Koç Montgomery multiplier and achieve half the latency and half the queue memory requirement. Recently significantly fast implementations are proposed by [26, 27].

Our previous work [28] employs a superscalar architecture to exploit ILP in ECC over $\text{GF}(2^n)$. The cryptoprocessor is scalable in field size and programmable for various EC scalar multiplication algorithms. The architecture in this paper is also based on it and explores the fastest possible performance of ECC by fixing the field size to 163 bits and using the irreducible polynomial, $x^{163} + x^7 + x^6 + x^3 + 1$.

3 Curve-based cryptography

Here, we consider some background information for curve-based cryptography over a binary field. We mention the basic algorithms and the structure of the

Table 2 Address map of memory in the cryptoprocessor

Address[3:0]	Variables	Descriptions
0,1	value 0 and 1	Constant values
2	$P(x) - x^n = \sum_{i=0}^{n-1} p_i x^i$	Irreducible polynomial
3,4	$a, c = \sqrt[4]{b}$ or x, b	Curve parameters
5,6,7	$X_1, Y_1, Z_1 (= 1)$	Initial point P_1
8,9,10	X_2, Y_2, Z_2	Intermediate point P_2
10~15	$t_1 \sim t_6$	Intermediate variables

operations. Good references for the mathematical background are [5, 29–31]. In addition, we give the detailed computation sequences for point addition and doubling in projective coordinates for the operation form $A(B + D) + C$ that the proposed P-MALU can handle as explained in section 4. We assume that the memory has a 4-bit address space and stores n -bit data as shown in Table 2.

3.1 ECC over a binary field

ECC relies on a group structure induced on an elliptic curve. A set of points on an elliptic curve (with one special point added, the so-called point at infinity $\mathcal{O}$) together with a point addition as a binary operation has the structure of an abelian group. As we consider a finite field of characteristic 2, i.e. $\text{GF}(2^n)$, a non-supersingular elliptic curve E over $\text{GF}(2^n)$ is defined as the set of solutions $(x, y) \in \text{GF}(2^n) \times \text{GF}(2^n)$ of the equation,

$$y^2 + xy = x^3 + a_0x^2 + b_0 \quad (1)$$

where $a_0, b_0, x, y \in \text{GF}(2^n)$, $b_0 \neq 0$, together with $\mathcal{O}$. The inverse of the point $P_1 = (x_1, y_1)$ is $-P_1 = (x_1, x_1 + y_1)$. The points on the curve and the point at infinity $\mathcal{O}$ form an abelian group. The sum $P + Q$ of the points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ ($P_1, P_2 \neq \mathcal{O}$, and $P_1 \neq \pm P_2$) is the point $P_3 = (x_3, y_3)$ where

$$\begin{aligned} \lambda &= \frac{y_1 + y_2}{x_1 + x_2} \\ x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 &= (x_1 + x_3)\lambda + x_3 + y_1. \end{aligned} \quad (2)$$

This operation is called point addition. For $P = Q$, point doubling formulae are

$$\begin{aligned} \lambda &= \frac{y_1}{x_1} + x_1 \\ x_3 &= \lambda^2 + \lambda + a \\ y_3 &= (x_1 + x_3)\lambda + x_3 + y_1. \end{aligned} \quad (3)$$

The point at infinity $\mathcal{O}$ is the neutral element, similar to the number 0 in ordinary addition. Thus, $P + \mathcal{O} = P$ and $P + (-P) = \mathcal{O}$ for all points P .

We use weighted projective coordinates where an affine point (x, y) is converted to a projective point (X, Y, Z) by computing $x = X/Z$, $y = Y/Z^2$. The projective curve equation corresponding to the affine equation (1) is given by

$$Y^2 + XYZ = X^3 + a_1 X^2 Z^2 + b_1 Z^6. \quad (4)$$

Suppose that point addition is computed by $P_2 = P_2 + P_1$ ($P_1 \neq P_2$), i.e., storing the result of $P_2 + P_1$ ($P_1 \neq P_2$) in the register of P_2 , in order to save memory space. The computation sequences for point addition on the projective curve is presented in (5) for the special case $Z_1 = 1$.

$$\begin{aligned} t_2 &= Z_2 Z_2, & t_4 &= t_2 X_1 + X_2, & t_2 &= t_2 Z_2, \\ t_1 &= t_2 Y_1 + Y_2, & t_3 &= t_4 Y_2, & t_3 &= t_1 X_2 + t_3, \\ Z_2 &= t_4 Z_2, & t_5 &= t_1(t_1 + Z_2), & Y_2 &= t_4 t_4, \\ t_2 &= t_4 Y_2 + t_5, & X_2 &= a Z_2, & X_2 &= X_2 Z_2 + t_2, \\ t_4 &= Y_2 t_3, & Y_2 &= X_2(t_1 + Z_2) + t_4. \end{aligned} \quad (5)$$

The computation sequence for point subtraction is obtained simply by using

$$t_1 = t_2(X_1 + Y_1) + Y_2 \quad (6)$$

in place of 4th Step of (5). For point doubling $P_2 = 2P_2$, we use

$$\begin{aligned} t_1 &= X_2 X_2, & t_2 &= t_1 t_1, & t_4 &= Y_2 Z_2 + t_1 \\ t_3 &= Z_2 Z_2, & Z_2 &= X_2 t_3, & t_5 &= c t_3 + X_2 \\ t_3 &= t_5 t_5, & X_2 &= t_3 t_3, & t_1 &= X_2(Z_2 + t_4) \\ Y_2 &= t_2 Z_2 + t_1 \end{aligned} \quad (7)$$

where $c = \sqrt[3]{b}$. We assume that this curve parameter is pre-computed and stored in memory.

Many techniques for recoding the scalar k have been proposed in the literature. Here we mention the signed-digit representation. Consider an integer representation of the form $k = \sum_{i=0}^l k_i 2^i$, where $k_i \in \{-1, 0, 1\}$. This is called the (binary) signed digit (SD) representation (see Menezes et al. [5, 31]). If an SD representation has no adjacent non-zero digits, it is called a non-adjacent form (NAF). Every integer k has a unique NAF which has the minimum weight of any signed digit representation of k . Algorithm I shows the binary NAF method for scalar multiplication.

Algorithm 1 Algorithm for point multiplication: binary NAF method [31].

Require: a point P

an positive integer $k = \sum_{i=0}^l k_i 2^i$, $k_i \in \{-1, 0, 1\}$

Ensure: kP

$P_1 \leftarrow P$, $P_2 \leftarrow \mathcal{O}$

for i from l downto 0 do

$P_2 \leftarrow 2P_2$;

if $k_i = 1$ then

$P_2 \leftarrow P_2 + P_1$;

if $k_i = -1$ then

$P_2 \leftarrow P_2 - P_1$;

end for

return P_2 ;

3.2 Montgomery's powering ladder

The EC scalar multiplication can be operated with the method of Montgomery that maintains the relationship $P_2 - P_1$ as invariant (Algorithm II) [17]. All computations are performed on the x -coordinate only in affine coordinates and hence the point multiplication can be performed with less computation sequences.

Algorithm 2 Algorithm for point multiplication: Montgomery method [17].

Require: a point P

an positive integer $k = (k_{l-1} \cdots k_1 k_0)_2$, $k_{l-1} = 1$

Ensure: kP

$P_1 \leftarrow P$, $P_2 \leftarrow 2P$

for i from $l-2$ downto 0 do

if $k_i = 1$ then

$P_1 \leftarrow P_1 + P_2$, $P_2 \leftarrow 2P_2$;

else

$P_2 \leftarrow P_1 + P_2$, $P_1 \leftarrow 2P_1$;

end for

return P_1 ;

If $P_1 \neq P_2$, the x -coordinate of point addition ($P_3 = P_2 + P_1$) is computed as

$$x_3 = x + \left(\frac{x_1}{x_1 + x_2} \right)^2 + \frac{x_1}{x_1 + x_2} \quad (8)$$

where

$$x = \frac{x_1 y_2 + x_2 y_1}{(x_1 + x_2)^2} + \frac{x_1 x_2}{(x_1 + x_2)^2} + \frac{x_1 x_2}{(x_1 + x_2)}. \quad (9)$$

If $P_1 = P_2$,

$$x_3 = x_1^2 + \frac{b}{x_1^2}. \quad (10)$$

For the purpose of escaping computationally expensive modular inversions, López and Dahab applied the method to the projective coordinates [32]. In this algorithm, all necessary computations are performed on the X and Z coordinates in projective representation. The computation sequences for point addition ($P_2 = P_1 + P_2$) and point doubling ($P_1 = 2P_1$) are described in (11) and (12).

$$\begin{aligned} X_2 &= X_2 Z_1, & Z_2 &= X_1 Z_2, & t_1 &= X_2 Z_2 \\ Z_2 &= X_2 + Z_2, & Z_2 &= Z_2 Z_2, & X_2 &= x Z_2 + t_1. \end{aligned} \quad (11)$$

$$\begin{aligned} t_2 &= X_1 X_1, & t_3 &= Z_1 Z_1, & Z_1 &= t_2 t_3 \\ t_2 &= t_2 t_2, & t_3 &= t_3 t_3, & X_1 &= b t_3 + t_2. \end{aligned} \quad (12)$$

3.3 Koblitz curve

A Koblitz curve is an elliptic curve defined over $GF(2)$ that is given by the equation

$$E_a : y^2 + xy = x^3 + ax^2 + 1 \quad (13)$$

where $a \in \{0, 1\}$. Koblitz curves are of interest because point doublings in the binary method can be replaced with computationally cheaper operations. Namely, consider the Frobenius map $\tau : E_a \rightarrow E_a$ defined by $\tau(x, y) = (x^2, y^2)$ for all points on a curve E_a except $\tau(\infty) = \infty$. This map can be easily computed since it relies on squaring operation in $GF(2^n)$. It can be shown from the point addition operation that three points (x, y) , (x^2, y^2) and (x^4, y^4) satisfy the following:

$$(x^4, y^4) + 2(x, y) = (-1)^{1-a}(x^2, y^2). \quad (14)$$

From the definition of the Frobenius map we get

$$(\tau^2 + 2)P = \mu\tau(P) \quad (15)$$

where $\mu = (-1)^{1-a}$.

So, the Frobenius map can be viewed as a complex number τ that satisfies $\tau^2 + 2 = \mu\tau$, for which we choose $\tau = (\mu + \sqrt{-7})/2$. Let $\mathbb{Z}[\tau]$ denote the ring of polynomials in τ with coefficients from $\mathbb{Z}$. Then we write:

$$\begin{aligned} (u_{t-1}\tau^{t-1} + \dots + u_1\tau + u_0)P \\ = u_{t-1}\tau^{t-1}(P) + \dots + u_1\tau(P) + u_0(P) \end{aligned} \quad (16)$$

Therefore, one has to find a decomposition of a scalar k in the following form $k = u_{t-1}\tau^{t-1} + \dots + u_1\tau + u_0$ (so-called τ -adic expansion) and then use the previous equation to compute kP . This scalar multiplication is called TNAF method and it is computed

Table 3 Computation cost of scalar multiplication

	ECC	ECC_M	ECC_KC
Point addition	14	6	14
Point doubling (τ multiplication)	10	6	2
Scalar multiplication	2,391	1,956	1,087

with point additions/subtractions and τ multiplications (τP) (see Algorithm III). Solinas proposed an efficient τ -adic expansion of k by recoding the scalar in this form [18].

Algorithm 3 Algorithm for point multiplication: TNAF method [31].

Require: a point P

an positive integer $\rho' = \sum_{i=0}^{l-1} u_i \tau^i$, $u_i \in \{-1, 0, 1\}$

Ensure: kP

$P_1 \leftarrow P$, $P_2 \leftarrow \mathcal{O}$

for i from $l-1$ downto 0 do

$P_2 \leftarrow \tau P_2$;

if $u_i = 1$ then

$P_2 \leftarrow P_2 + P_1$;

if $u_i = -1$ then

$P_2 \leftarrow P_2 - P_1$;

end for

return P_2 ;

Table 3 summarizes the cost of the three different computation sequences for scalar multiplication. Figures in the table indicate the number of operations by the P-MALU. The cost of scalar multiplication is estimated with 163-bit key.

4 Cryptoprocessor architecture

The proposed architecture of the cryptoprocessor is composed of the main controller, several P-MALUs and the register file that shares intermediate variables between the P-MALUs (i.e. the so-called shared memory). The block diagram of the cryptoprocessor is illustrated in Fig. 2. Although conventional hardware architectures for ECC employ several different data processing units that accelerates specific operations, (e.g. modular squaring and inversions), our cryptoprocessor executes a single operation form, $A(B + D) + C$ on the P-MALU. As modular inversions are performed with a chain of modular additions and multiplications [33], one can perform all necessary modular operations for ECC with this operation form. This architecture facilitates parallel processing for EC scalar multiplication.

Figure 2 Block diagram for programmable EC cryptoprocessor that embeds multiple parallelized MALUs

The main CPU communicates with the cryptoprocessor through memory-mapped I/O (e.g. SRAM interface) and has three types of 32-bit inputs and outputs; one of them is a signal that tells the main CPU to stop sending instructions when the instruction buffer is full

(IQB full in Fig. 2). A 32-bit input/output passes data back and forward between the main CPU and the cryptoprocessor and a 32-bit output is used to send instructions. The data transfer between the main CPU and the cryptoprocessor is controlled by a data bus controller (DBC). If using SRAM attached to the main CPU for storing intermediate variables during ECC operations, the cryptoprocessor can be constructed without use of the register file in the cryptoprocessor. Alternatively, for the purpose of reducing the I/O transfer overhead, the register file can be embedded in the cryptoprocessor. In this case, the path through the DBC is only activated when an initial point and the parameters of an elliptic curve are sent to the RAM, or when the result is retrieved. The details about a instruction bus controller (IBC) will be discussed in Section 4.3 including our strategy for ILP.

4.1 Parallelized modular arithmetic logic unit (P-MALU)

Our proposed architecture for the P-MALU is based on the implementation presented in [34]. The datapath of the P-MALU deals with a polynomial basis

Figure 3 Parallelized datapath for polynomial-basis $GF(2^n)$ operation. MSB-first digit-serial multiplier (left). Montgomery modular multiplier (right). Most of the XORs in the gray blocks can be removed when fixing the irreducible polynomial, e.g. $P(x) = x^{163} + x^7 + x^6 + x^3 + 1$

$(1, \alpha, \alpha^2, \dots, \alpha^{n-1})$, where α is a root of an irreducible polynomial $P(x)$ of degree n over $\text{GF}(2)$. It is composed of a conventional bit-serial MSB-first multiplier and a Montgomery's multiplier over $\text{GF}(2^n)$ as illustrated in Fig. 3 (the XOR chain in the left and right side, respectively). This datapath implementation computes

$$A(x)B(x) \cdot x^{-N} + C(x) \pmod{P(x)}, \quad (17)$$

where $N = \lceil n/2 \rceil$, $A(x) = \sum_{i=0}^{n-1} a_i x^i$, $B(x) = \sum_{i=0}^{n-1} b_i x^i$, $C(x) = \sum_{i=0}^{n-1} c_i x^i$ and $P(x) = x^n + \sum_{i=0}^{n-1} p_i x^i$. This is explained as follows. First, the MSB-first multiplier performs modular multiplication by providing the coefficients of $A(x)$ from the MSB up to N bits. Thus we obtain the partial modular multiplication result as

$$\sum_{i=N}^{n-1} a_i x^{i-N} \cdot B(x) \pmod{P(x)}. \quad (18)$$

Second, the Montgomery's multiplication is executed by providing the coefficients of $A(x)$ from the LSB as follows.

$$\sum_{i=0}^{N-1} a_i x^i \cdot B(x) \cdot x^{-N} \pmod{P(x)}. \quad (19)$$

Lastly, above two multiplications and $C(x)$ are added, and (17) is obtained. The advantage of this method lies in the fact that modular multiplication can be accelerated by computing two different multiplications in parallel. As the operation can be completed in N cycles, the latency of the modular multiplication can be improved compared to the case of using individual multipliers that take n cycles.

Although $B(x)$ can be divided into different bit sizes, it is simple to separate it in half (i.e. $\lceil n/2 \rceil$ bits) so that the computation in (17) is distributed equally to two different type of multipliers. The paper also employs this strategy. For instance, $n = 163$ can be used for $\text{GF}(2^{163})$. Then a 163-bit $\times$ 163-bit modular multiplication is divided into two 82-bit $\times$ 163-bit multiplications.

For convenience of repeated usage of (17), the modified Montgomery form or $\tilde{A}(x) = A(x) \cdot x^N \pmod{P(x)}$ is applied because the output is in the Montgomery form as well.

$$\begin{aligned} \tilde{A}(x)\tilde{B}(x) \cdot x^{-N} + \tilde{C}(x) \pmod{P(x)} \\ = (A(x)B(x) + C(x)) \cdot x^N \pmod{P(x)}. \end{aligned} \quad (20)$$

4.2 MSB-first modular multiplier and Montgomery modular multiplier

Here, the digit-serial implementation of two multipliers is explained in detail. The MSB-first multiplier sums up

three inputs that are $a_i B(x)$, $m_i P(x)$, and $T(x)$, and then outputs the intermediate result, $T_{next}(x)$ by computing

$$T_{next}(x) = (T(x) + a_i B(x) + m_i P(x))x \quad (21)$$

where $m_i = t_n$. By providing $A(x)$ by d bits from the MSB-side and $T_{next}(x)$ as the next-cycle input $T(x)$, one can obtain the result of (18) in $\lceil N/d \rceil$ cycles. The detailed explanation is also discussed in [35].

As for the Montgomery multiplier, $A(x)$ is sent to the datapath from the LSB. In this way,

$$U_{next}(x) = (U(x) + a_i B(x) + w_i P(x))/x \quad (22)$$

is computed where $w_i = u_0 + a_i b_0$. The result of (19) can be obtained in $\lceil N/d \rceil$ cycles for the case of d -bit digit-serial implementation. Those results of $T(x)$ and $U(x)$ are XORed with $C(x)$ at the last step in multiplication.

Figure 4 Datapath for MSB-first digit-serial polynomial-basis $\text{GF}(2^n)$ operation. Most of the XORs in the gray blocks can be removed when fixing the irreducible polynomial

Table 4 Example of parallelized out-of-order instruction sequence for three consecutive point doublings of ECC with two-way executions (*i.e.* $P \Leftarrow 2^3 P$, where $P(X_1, Y_1, Z_1)$)

The ECDBs in italic are instructions from preceding and succeeding point doublings

	Original sequence						Parallelized out-of-order sequence
	Address:	<i>R</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	
ECDB01:	(	t_1 ,	X_1 ,	X_1 ,	0,	0	) <i>ECDB08</i> & ECDB04
ECDB02:	(	t_2 ,	t_1 ,	t_1 ,	0,	0	) <i>ECDB09</i> & ECDB06
ECDB03:	(	t_4 ,	Y_1 ,	Z_1 ,	t_1 ,	0	) <i>ECDB10</i> & ECDB01
ECDB04:	(	t_3 ,	Z_1 ,	Z_1 ,	0,	0	) ECDB02 & ECDB03
ECDB05:	(	Z_1 ,	X_1 ,	t_3 ,	0,	0	) ECDB05 & ECDB07
ECDB06:	(	t_5 ,	d_6 ,	t_3 ,	X_1 ,	0	) ECDB08 & <i>ECDB04</i>
ECDB07:	(	t_3 ,	t_5 ,	t_5 ,	0,	0	) ECDB09 & <i>ECDB06</i>
ECDB08:	(	X_1 ,	t_3 ,	t_3 ,	0,	0	) ECDB10 & <i>ECDB01</i>
ECDB09:	(	t_1 ,	X_1 ,	Z_1 ,	0,	t_4	) <i>ECDB02</i> & <i>ECDB03</i>
ECDB10:	(	Y_1 ,	t_2 ,	Z_1 ,	t_1 ,	0	) <i>ECDB05</i> & <i>ECDB07</i>

The proposed datapath is scalable in the digit size d which can be decided by exploring the best combination of system performance and cost. The field size n is determined by the key-length. Figure 3 illustrates the block diagram of the digit-serial P-MALU.

For the purpose of checking the improvement in area-performance trade-off by using the P-MALU, we also implement a compact modular arithmetic unit that has only MSB-first multiplier as shown in Fig. 4. We denote this multiplier as MALU. The MALU performs

$$A(x)B(x) + C(x) \pmod{P(x)} \quad (23)$$

in $\lceil n/d \rceil$ cycles which is almost double comparing to the P-MALU.

4.3 Instruction-level parallelism

Instructions are sent to the MALU or P-MALU either from the main CPU or from pre-set micro codes in the μ -code RAM. When the main CPU is in charge of dispatching instructions, the IBC block can be detached from the cryptoprocessor. In this case, it occurs that the throughput of issuing instructions is not high enough for the MALU(s) or P-MALU(s) to be utilized effectively. On the contrary, when the μ -code RAM is used for assisting the main CPU, the instruction bus controller (IBC) can handle one instruction per cycle. For instance, the sequence of point doubling is stored in the μ -code RAM and the main CPU calls it as an instruction. Thus multiple P-MALUs can be activated in parallel without any instruction stalls. This is another level of parallelism in the proposed cryptoprocessor. During point multiplication, the IBC keeps on reading instructions from the μ -code RAM and stores them to an instruction queue buffer (IQB) unless the IQB is full. The IBC checks if there is instruction-level parallelism (ILP) by checking the data-dependency of instructions in the IQB and forwards them to the

P-MALU(s) including *out-of-order* execution. Table 4 shows an example of the mechanism of ILP for three consecutive point doublings of ECC. The first two instructions have a RAW (read-after-write) dependency with t_1 . ECDB04 has no RAW dependency upon the first three instructions in in-order and out-of-order execution, and therefore it can be issued prior to the first three instructions.

5 Results

As discussed in the previous section, the proposed cryptoprocessor is oriented to maximize its performance

Figure 5 Degree of parallelism and required clock cycles for different hardware configurations of ECC and ECC_M

Figure 6 Performance of ECC for area complexity. The figures labeled on plots indicate the number of P-MALU or MALU

by deploying hardware that supports multi-level parallelism in ECC as well as using an optimized algorithm for scalar multiplication. The cryptoprocessor is synthesized with 0.13- μ m CMOS technology. In this section, cost and performance trade-offs are discussed.

5.1 Performance improvement by three-level parallel processing

Our proposed cryptoprocessor has three-level parallelism; the first is in the datapath of the MALU, the second is in modular operations performed in the P-MALU and the third is in the instructions determined by the computation sequence of EC scalar multiplication. Figure 5 illustrates performance improvement of the proposed cryptoprocessor for different hardware configurations and different scalar multiplication algorithms to see the effect of each parallelism. Although double performance cannot be obtained by introducing the P-MALU due to the memory accesses, the performance of scalar multiplication is improved by a factor of 1.8 for all three scalar multiplication algorithms. The performance also improves as the number of the P-MALU or MALU increases because of the parallel execution of instructions. In order to see the

Table 5 Performance comparison of ECC hardware implementations over a binary field

Ref. design	Technology/ FPGA	$f_{\max}$ (MHz)	Area (slices/gates)	Galois field	Field size /irreducible polynomial	Performance (μ s)	Comments
This work (Fastest)	0.13- μ m CMOS	555.6	154K gates	$GF(2^{163})$	163 bits/ Fixed: $x^{163} + x^7 + x^6 + x^3 + 1$	20	Binary NAF method.
						12	TNAF method.
						16	Montgomery method.
This work (Best trade-off)	0.13- μ m CMOS	555.6	108K gates	$GF(2^{163})$	163 bits/ Fixed: $x^{163} + x^7 + x^6 + x^3 + 1$	27	Binary NAF method.
			131K gates			14	TNAF method.
			90K gates [‡]			17	Montgomery method.
Sozzani et al. [27]	0.13- μ m CMOS	416.7	113K gates [‡]	$GF(2^{163})$	163 bits/Arbitrary	209	Binary method. Multiplier, divider and squarer.
Satoh and Takano [23]	0.13- μ m CMOS	510.2	117.5K gates	$GF(2^{163})$	163 bits/Arbitrary	30	Montgomery method. Two multipliers, divider and squarer.
Smyth et al. [36]	0.13- μ m CMOS	166	90.9K gates 32K-bit RAM	$GF(2^{163})$	Arbitrary/Arbitrary	190	Support of $GF(2^m)$ and $GF(p)$. 64-bit $\times$ 64-bit multiplier.
					Arbitrary/Arbitrary	3,720	Support of $GF(2^m)$ and $GF(p)$ up to 2,016-bit fields.

[‡]: Authors' estimation based on their results

Figure 7 Performance of ECC_M for area complexity. The figures labeled on plots indicate the number of P-MALU or MALU

effectiveness of this parallelism, we define the *degree of parallelism* as

$$\frac{\sum_{i=1}^{l_{\max}} M_i \cdot i}{\sum_{i=1}^{l_{\max}} M_i} \quad (l_{\max} = 1, 2, 3, 4) \quad (24)$$

where M_i is the number of instructions executed in i -way parallel processing and $l_{\max}$ is the number of P-MALUs or MALUs. For instance, the degree of parallelism becomes 2.0 for ECC_M with two copies of the datapath since almost all instructions are processed in two-way parallel executions. The degree of parallelism is almost proportional to the number of the processing elements up to three-way parallelism for the case of ECC_M. In this way, we obtain 2.7 for degree of parallelism in ECC_M with three copies of the datapath. However, effective performance improvement cannot be observed even with four P-MALUs or MALUs. For the degree of parallelism in ECC and ECC_KC, we obtain 2.1 and 2.2 with three copies of the datapath, respectively. This results indicate that the computation sequence in ECC and ECC_KC cannot utilize more than two processing elements effectively.

5.2 Synthesis results

For the purpose of observing the fastest implementation possible, the cryptoprocessor is synthesized with

0.13- μm CMOS technology with the P-MALU configuration of $d = 12$. As summarized in Table 5, our results are faster than any other previous ASIC work to our best knowledge. Work by Sozzani et al. [27] showed several different ECC implementations. Although their design of ECC_M shows comparable performance, the result of ECC is much lower than ours. Satoh and Takano proposed a scalable dual-field cryptoprocessor [23] that could support ECC over $\text{GF}(2^m)$ and $\text{GF}(p)$. Another scalable dual-field ECC processor was designed by Smyth et al. [36]. Their implementations offer greater flexibility and support of various ECC. Their performances of ECC over $\text{GF}(2^{163})$, however, are much lower than ours. Figure 6 illustrates area and performance trade-offs for the previous work and our cryptoprocessor with various implementation options.

5.3 Cost-performance trade-offs

So far, we focused on reducing the time for one scalar multiplication. More precisely, our optimization in the cryptoprocessor is for reducing the latency of one scalar multiplication. This type of improvement is important for applications that perform scalar multiplications sequentially. When an ECC-RFID reader needs to verify the identity of an ECC-RFID tag, the critical operation

Figure 8 Performance of ECC_KC for area complexity. The figures labeled on plots indicate the number of P-MALU or MALU

Figure 9 Performance of ECC, ECC_M and ECC_KC with the P-MALU for the number of channels. The window method is also applied for ECC and ECC_KC (width-4). The figures labeled on plots indicate the number of channels

is scalar multiplication. For speeding up the protocol, scalar multiplication in the reader should be as fast as possible since tags are normally heavily constrained in area and performance and the communication speed between tags and the reader is determined by the carrier frequency.

On the other hand, we can assume other kinds of PKC applications that need scalar multiplications in parallel. If computing two independent scalar multiplications (e.g. $k_1 P_1$ and $k_2 P_2$), the performance can be doubled by using two EC cryptoprocessors (denoted as two channels) in parallel. Namely, this type of improvement is for increasing the throughput.

Figure 7 and 8 show trade-offs between area and performance respectively for ECC_M and ECC_KC with up to four channels. The figures marked at the plot indicate the number of P-MALUs or MALUs. One channel of the cryptoprocessor with one MALU is the smallest hardware configuration in our study, and the gate size is 78K gates. By increasing the number of MALUs, the performance improves up to 38,200 and 48,600 kP/s respectively for ECC_M and ECC_KC. When using the P-MALUs in place of the MALUs, the performance achieves 82,000 and 62,600 kP/s respectively for ECC_M and ECC_KC with four P-MALUs.

As discussed in Section 5.1, the best cost-performance trade-off is obtained from a cryptoprocessor with two P-MALUs for ECC and ECC_KC, and with three P-MALUs for ECC_M. Figure 9 shows the estimated cost-performance curve when preparing up to ten channels of ECC each of which has two P-MALUs for ECC and ECC_KC, and three P-MALUs for ECC_M. As can be seen from this figure, our 163-bit

ECC implementation offers performances of 689, 446 and 350 kP/s per 1 K gate, respectively for ECC_KC, ECC_M and ECC on the same cyptoprocessor.

The result of the NAF method with windows of width 4 (NAF₄) for ECC and the TNAF method with windows of width 4 (TNAF₄) for ECC_KC are also plotted to compare to the binary method and the TNAF method, respectively. As a result, NAF₄ and TNAF₄ show worse area-performance trade-offs in our implementation. This is because we use a flip-flop-based multi-port register file, and hence additional memory space for pre-computed points is expensive considering the performance gain.

6 Conclusions

This paper introduced an EC cryptoprocessor that exploits three-level parallelism in ECC. The implementation results on a generic curve showed that scalar multiplication of ECC over $GF(2^{163})$ were performed in 16 and 20 μs , respectively with the binary NAF method and the Montgomery method. Further more, the scalar multiplication was accelerated up to 12 μs on a Koblitz curve by using the TNAF method. This speed-up was achieved by thoroughly exploiting parallelism in the system architecture of ECC.

Acknowledgements Kazuo Sakiyama and Lejla Batina are funded by a research grant of the Katholieke Universiteit Leuven and FWO projects (G.0450.04, G.0475.05). This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by the EU IST FP6 projects (SESOC and ECRYPT), by the K.U. Leuven, and by the IBBT-QoE project of the IBBT.

References

1. Diffie W, Hellman ME (1976) New directions in cryptography. *IEEE Trans Inf Theor* 22:644–654
2. Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. *Commun ACM* 21(2):120–126
3. Koblitz N (1987) Elliptic curve cryptosystem. *Math Comput* 48:203–209
4. Miller V (1985) Uses of elliptic curves in cryptography. In: Williams HC (ed) *Advances in cryptology: Proceedings of CRYPTO'85*, number 218 in lecture notes in computer science. Springer, pp 417–426
5. Blake I, Seroussi G, Smart NP (1999) *Elliptic curves in cryptography*. London mathematical society lecture note series. Cambridge Univ. Press
6. Hodjat A, Verbaudhede I (2006) Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors. *IEEE Trans Comput* 55(4):366–372, April

7. Wolkerstorfer J (2005) Scaling ECC hardware to a minimum. In: ECRYPT workshop—cryptographic advances in secure hardware—CRASH 2005, September 6–7 2005 (invited talk)
8. Kumar S, Paar C (2006) Are standards compliant elliptic curve cryptosystems feasible on RFID? In: Proceedings of Workshop on RFID Security
9. Batina L, Mentens N, Sakiyama K, Preneel B, Verbauwhede I (2006) Low-cost elliptic curve cryptography for wireless sensor networks. In: Proceedings of Third European Workshop on Security and Privacy in Ad Hoc and Sensor Networks, number 4357 in lecture notes in computer science. Springer, pp 415–429
10. Juels A (2006) RFID security and privacy: a research survey. *IEEE J Sel Areas Commun* 24(2):381–394, February
11. Abe M (1998) Universally verifiable mix-net with verification work independent of the number of mix-servers. In: Nyberg N (ed) *Advances in cryptology: proceedings of EUROCRYPT'98*, number 1403 in lecture notes in computer science. Springer, pp 437–447
12. Neff CA (2001) A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001), pp 116–125, November
13. Ostrovsky R, We Skeith III (2005) Private searching on streaming data. In: Shoup V (ed) *Advances in cryptology: proceedings of CRYPTO'05*, number 3621 in lecture notes in computer science. Springer, pp 223–240
14. Smart NP (2001) The Hessian form of an elliptic curve. In: Koç ÇK, Naccache D, Paar C (eds) *Proceedings of 3rd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 2162 in lecture notes in computer science. Springer, Paris, France, pp 121–128, May 14–16
15. IEEE P1363 / D13 (Draft Version 13) (1999) Standard specifications for public key cryptography, November
16. US Department of Commerce and National Institute of Standards and Technology (2000) Digital signature standard (DSS), FIPS PUB 186-2, January
17. Montgomery P (1987) Speeding the Pollard and elliptic curve methods of factorization. *Math Comput* 48(177):243–264
18. Solinas J (2000) Efficient arithmetic on koblitz curves. *Designs Codes Cryptogr* 19:195–249
19. Agnew GB, Mullin RC, Vanstone SA (1989) A fast elliptic curve cryptosystem. In: Quisquater J-J, Vandewalle J (eds) *Advances in cryptology: proceedings of EUROCRYPT'89*, number 434 in lecture notes in computer science. Springer, pp 706–708
20. Gao L, Shrivastava S, Lee H, Sobelman GE (1999) A compact fast variable key size elliptic curve cryptosystem coprocessor. In: Proceedings of the seventh annual IEEE symposium on field-programmable custom computing machines (FCCM), pp 304–305
21. Orlando G, Paar C (2000) A high-performance reconfigurable elliptic curve processor for $GF(2^m)$. In: Koç ÇK, Paar C (eds) *Proceedings of 2nd International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 1965 in lecture notes in computer science. Springer-Verlag, pp 41–56
22. Eberle H, Gura N, Shantz SC (2003) Cryptographic processor for arbitrary elliptic curves over $GF(2^m)$. In: Schulte M, Bhattacharyya S, Burgess N, Schreiber R (eds) *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP)*, pp 444–454, June 24–26
23. Satoh A, Takano K (2003) A scalable dual-field elliptic curve cryptographic processor. *IEEE Transactions on Computers*, special issue on cryptographic hardware and embedded systems 52(4):449–460, April
24. Tenca AF, Koç ÇK (2003) A scalable architecture for modular multiplication based on Montgomery's algorithm. *IEEE Trans Comput* 52(9):1215–1221, September
25. Harris D, Krishnamurthy R, Andres M, Mathew S, Hsu S (2005) An improved unified scalable radix-2 montgomery multiplier. In: *Proceedings of 14th IEEE Symposium on Computer Arithmetic (ARITH-17)*, pp 172–178
26. Saqib NA, Rodríguez-Henruez F, Díaz-Prez, A (2005) A reconfigurable processor for high speed point multiplication in elliptic curves. *Int J of Embed Syst* 2005 1(3/4):237–249
27. Sozzani F, Bertoni G, Turcato S, Breveglieri L (2005) A parallelized design for an elliptic curve cryptosystem coprocessor. In: *International symposium on information technology: Coding and computing (ITCC 2005)*, pp 626–630
28. Sakiyama K, Batina L, Preneel B, Verbauwhede I (2006) Superscalar coprocessor for high-speed curve-based cryptography. In: Goubelin L, Matsui M (eds) *Proceedings of 8th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, number 4249 in lecture notes in computer science. Springer, pp 415–429
29. Koblitz N (1998) *Algebraic aspects of cryptography*, 1st edn. Springer
30. Menezes A, Wu Y-H, Zuccherato R (1998) An elementary introduction to hyperelliptic curves—appendix. In: Koblitz N (ed) *Algebraic aspects of cryptography*. Springer, pp 155–178
31. Hankerson D, Menezes A, Vanstone S (2004) *Guide to elliptic curves cryptography*. Springer
32. López J, Dahab R (1999) Fast multiplication on elliptic curves over $GF(2^m)$. In: Koç ÇK, Paar C (eds) *Proceedings of 1st International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, vol 1717 of lecture notes in computer science. Springer, pp 316–327
33. Itoh T, Tsujii S (1988) Effective recursive algorithm for computing multiplicative inverses in $GF(2^m)$. *Electron Lett* 24(6):334–335
34. Batina L, Mentens N, Preneel B, Örs SB (2004) Serial multiplier architectures over $GF(2^n)$ for elliptic curve cryptosystems. In: *The 12th IEEE Mediterranean Electrotechnical Conference (MELECON 2004)*, pp 779–782
35. Sakiyama K, Preneel B, Verbauwhede I (2006) A fast dual-field modular arithmetic logic unit and its hardware implementation. In: *Proceedings of IEEE International Symposium On Circuits and Systems (ISCAS'06)*, pp 787–790
36. Smyth N, McLoone M, McCanny JV (2006) An adaptable and scalable asymmetric cryptographic processor. In: *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, And Processors (ASAP)*. IEEE Computer Society, pp 341–346

Kazuo Sakiyama obtained the B.Eng. and M.Eng. degrees in electrical engineering from Osaka University, Japan in 1994 and 1996, respectively. From 1996 to 2004, he was with the Semiconductor and IC Division of Hitachi, Ltd. (now Renesas Technology Corp.). During this time, he received the M.S. degree in electrical engineering from the University of California, Los Angeles (UCLA). He is currently working on a Ph.D. program at the Katholieke Universiteit Leuven (K.U. Leuven), Belgium. His main research interest is efficient and secure embedded system architectures and design methodologies.

Lejla Batina Lejla Batina is a post-doctoral researcher at the research group COSIC in the Electrical Engineering Department of the Katholieke Universiteit Leuven, Belgium. She received her M.Sc. degree in Mathematics from the University of Zagreb, Croatia in 1995 and Ph.D. degree in engineering from the K.U. Leuven in 2005. She also studied and worked as a research assistant at the Technical University of Eindhoven, The Netherlands from 1999 to 2001. Her research interests include efficient arithmetic for cryptographic algorithms, secure implementations of cryptographic algorithms, side-channel security, lightweight cryptography e.g. crypto for RFIDs, sensor net works etc.

Bart Preneel received the Electrical Engineering degree and the Doctorate in Applied Sciences from the Katholieke Universiteit Leuven (Belgium). He is currently full professor at the Katholieke Universiteit Leuven. He was visiting professor at several universities in Europe. His main research interests are cryptography and information security.

He has authored and co-authored more than 200 scientific publications. He is vice president of the IACR (International Association for Cryptologic Research) and a member of the Editorial Board of the Journal of Cryptology and of the IEEE Transactions on Forensics and Information Security. He has participated to 20 research projects sponsored by the European Commission, for four of these as project manager. He has been program chair of nine international conferences (including Eurocrypt 2000, SAC 2005 and ISC 2006) and he has been invited speaker at 25 conferences.

In 2003, he has received the European Information Security Award in the area of academic research, and he received an honorary Certified Information Security Manager (CISM) designation by the Information Systems Audit and Control Association (ISACA). He is president of L-SEC vzw. (Leuven Security Excellence Consortium), an association of 45 companies and research institutions in the area of e-security. <http://homes.esat.kuleuven.be/~preneel>

Ingrid Verbauwhede's interest include circuits, processor architectures and design methodologies for real-time embedded systems for security, cryptography, digital signal processing and wireless communications. This includes the influence of new technologies and new circuit solutions on the design of next generation systems-on-chip.

She received her Electrical Engineering degree and Ph.D. degree from the K.U. Leuven in Belgium. She is currently a professor at the K.U. Leuven and an adjunct associate professor

at UCLA. At K.U. Leuven, she is co-director of the COSIC (Computer Security and Industrial Cryptography) lab. She was a lecturer and visiting research engineer at UC Berkeley from 1992 to 1994. From 1994 to 1998 she was a principal engineer first with TCSI and then with Atmel in Berkeley, CA. She joined UCLA in 1998 as an associate professor and the K.U. Leuven in 2003.

She is active in many conferences: she was the program chair in 2002 and the general chair for ISLPED 2003. She was a member of the executive committee of the 42nd and 43th DAC as the design community chair. She is the program chair of the 2007 CHES (Cryptographic Hardware and Embedded Systems) conference. More information on her research can be found by going to: www.emsec.ee.ucla.edu or www.esat.kuleuven.be/cosic.