

Parallel and High-Speed Computations of Elliptic Curve Cryptography Using Hybrid-Double Multipliers

Reza Azarderakhsh, *Member, IEEE*, and Arash Reyhani-Masoleh, *Member, IEEE*

Abstract—High-performance and fast implementation of point multiplication is crucial for elliptic curve cryptographic systems. Recently, considerable research has investigated the implementation of point multiplication on different curves over binary extension fields. In this paper, we propose efficient and high speed architectures to implement point multiplication on binary Edwards and generalized Hessian curves. We perform a data-flow analysis and investigate maximum number of parallel multipliers to be employed to reduce the latency of point multiplication on these curves. Then, we modify the addition and doubling formulations and employ a newly proposed digit-level hybrid-double Gaussian normal basis multiplier to remove the data dependencies and hence reduce the latency of point multiplication. To the best of our knowledge, this is the first time that one employs hybrid-double multiplication technique to reduce the computation time of point multiplication. Moreover, we have implemented our proposed architectures for point multiplication on FPGA and obtained the results of timing and area. Our results indicate that the proposed scheme is one step forward to improve the performance of point multiplication on binary Edward and generalized Hessian curves.

Index Terms—Elliptic curve cryptography (ECC), double-hybrid multiplier, binary Edwards curves, generalized Hessian curves, Gaussian normal basis

1 INTRODUCTION

It has been shown by Miller [1] and Koblitz [2] independently that a group of points on an elliptic curve over finite fields can be used for elliptic curve cryptography (ECC) as a public-key cryptography method. In comparison to the RSA, ECC offers the same level of security employing smaller key size [3]. Therefore, efficient implementation of ECC in terms of time-area trade-offs is crucial. For server applications, high speed implementations are required while for small and embedded devices area usage is the main concern that needs to be considered.

There are several implementations available in the literature; see for example, [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], and [17]. In all of these implementations, various hardware platforms such as field-programmable gate arrays (FPGAs) and application-specific integrated circuit (ASIC) have been utilized. Binary Edwards and generalized Hessian curves have recently been introduced in [18] and [19]. It has been shown that all generic elliptic curves over binary fields can be written in Edwards and generalized Hessian form to obtain complete and unified addition formulas which work for all pairs of inputs [18]. However, few works in the literature have considered the implementation

of point multiplication on these curves [20], and [17]. In [20], an ASIC implementation of point multiplication on binary Edwards curves (BECs) is presented for resource-constrained applications. In [17], lower level parallelization in finite field arithmetic as well as parallelization in higher level of curve formulations have been investigated to evaluate the time-area trade-offs. Curve level parallelization results indicate that although point multiplication on these curves is unified and complete, but they are slow in comparison to the point multiplication results on binary generic curves proposed in [21] and [4]. This is due to the data dependency in computing combined point addition (PA) and doubling which require three levels of multiplication employing (maximum) four parallel finite field multipliers. Note that point multiplication on binary generic curves require only two levels of multiplication incorporating three or more parallel multipliers as pointed out in [4] and [22].

Highly-parallel and fast computations of the widely-used cryptographic algorithm ECC is required for high-performance applications such as secure web servers and systems using big data. However, a challenge to cope with is that most applications for which parallelism is essential, have significantly large scale that is not commonly supported by today's cryptographic algorithms. Therefore, new techniques and architectures are required to investigate parallelization and scalability in all levels of computations of ECC. The main purpose of this paper is to provide new architectures to reduce the latency of the computation of cryptographic primitives specifically point multiplication of ECC targeting high-performance applications.

In this paper, we propose a new scheme to reduce the latency of point multiplication on binary Edwards and generalized Hessian curves using a hybrid-double multiplication technique proposed in [23]. The digit-level

- R. Azarderakhsh is with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, New York. E-mail: rxaec@rit.edu.
- A. Reyhani-Masoleh is with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON, Canada, N6A 5B9. E-mail: areyhani@uwo.ca.

Manuscript received 12 Nov. 2013; revised 21 Mar. 2014; accepted 6 May 2014. Date of publication 11 May 2014; date of current version 8 May 2015.

Recommended for acceptance by H. Jin.

For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.

Digital Object Identifier no. 10.1109/TPDS.2014.2323062

hybrid-double multiplier of [23] computes two consecutive multiplications using the same latency as required for a single traditional digit-level multiplier. Using this scheme reduces the latency of point multiplication on these curves to only two levels of hybrid-double multiplication and hence increases the speed of point multiplication on binary Edwards and generalized Hessian curves. It should be noted that the scheme proposed in this paper is actually more than just merging the previous works presented in [23] and [17].

We first perform data-flow analysis for ECC computations to understand how data has to move between the different logic and computational elements such as field multipliers, adders, and squarers over binary fields $GF(2^m)$. Then, we perform a latency analysis to determine where there are double field multiplications and hence we can employ the hybrid-double multiplier (with small modifications in the PA and PD formulae). The results indicate that the speed of point multiplication on these curves are competitive with binary generic curves as well as providing completeness. To evaluate the practical performance of the proposed scheme, we design two crypto-processors using the hybrid-double multiplier and code it using VHDL and implement it on Xilinx Virtex-4 and Virtex-7 FPGAs.

The rest of the paper is organized as follows. In Section 2, preliminaries of Gaussian normal basis (GNB) representation and hybrid-double multiplication are presented. Also, binary Edwards and generalized Hessian curves are briefly explained in this section. In Section 3, a latency reduction scheme for point multiplication is presented. In Section 4, the proposed ECC crypto-processors are explained. In Section 5, we implement the crypto-processors on two FPGAs and their time and area results are presented. Finally, we conclude the paper in Section 6.

2 PRELIMINARIES

2.1 Gaussian Normal Basis

The finite fields of characteristic two, $GF(2^m)$ can be constructed by a normal basis $N = \{\beta, \beta^2, \beta^{2^2}, \dots, \beta^{2^{m-1}}\}$ where $\beta \in GF(2^m)$ is called a normal element of $GF(2^m)$. Then, any element of $GF(2^m)$, say, $A = (a_0, a_1, \dots, a_{m-1})$, can be represented as $A = \sum_{i=0}^{m-1} a_i \beta^{2^i}$, where $a_i \in GF(2)$ [24]. In normal basis, squaring can be achieved by simple right cyclic shift of A , i.e., $A^2 = \sum_{i=0}^{m-1} a_i \beta^{2^{i+1}} = (a_{m-1}, a_0, a_1, \dots, a_{m-2})$ [3]. Note that this operation is fast without any cost if it is implemented in hardware. Gaussian normal basis is a special class of normal basis which is included in the IEEE 1363 [25] and NIST [3] standards for elliptic curve digital signature algorithm (ECDSA) and exists for every $m > 1$ that is not divisible by eight. The complexities of type T , GNB multiplier in terms of time and area depend on $T > 1$. For the five binary fields recommended by NIST, i.e., $m = 163, 233, 283, 409$, and 571 , the values of T are even, and are 4, 2, 6, 4, and 10, respectively.

2.2 Single Multiplication

Single multiplication is the computation of $C = A \times B$ in $GF(2^m)$ and several research in the literature has been conducted on its efficient computation and implementation [28], [27], and [26].

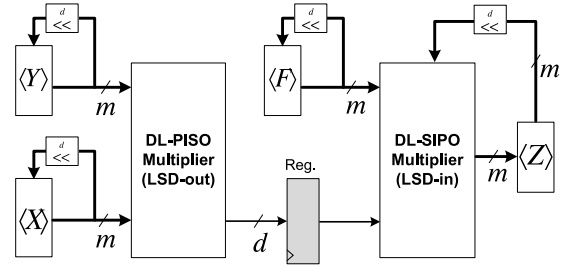


Fig. 1. The architecture of digit-level hybrid-double multiplier of [23].

GNB multiplication is based on a multiplication matrix $R_{(m-1) \times T}$ [26]. Let A and B be two field elements represented by GNB over $GF(2^m)$. Then, their product in $GF(2^m)$ is [26]

$$C = (A \odot (B \ll 1)) \oplus \sum_{i=1}^{m-1} (A \ll i) \odot S(i, B), \quad (1)$$

where

$$S(i, B) = (B \ll R(i, 1)) \oplus (B \ll R(i, 2)) \oplus \dots \oplus (B \ll R(i, T)), 1 \leq i \leq m-1, \quad (2)$$

and $(X \ll i)$ is the i -fold left cyclic shift of $X \in GF(2^m)$ and T is the type of GNB over $GF(2^m)$. Also, $X \odot Y = (x_0 y_0, \dots, x_{m-1} y_{m-1})$ and $X \oplus Y = (x_0 + y_0, \dots, x_{m-1} + y_{m-1})$ denote bit-wise AND (\odot) and XOR (\oplus) operations between coordinates of X and Y , respectively. $R(i, j)$, $1 \leq j \leq T$ denotes the entries of column i and row j of multiplication matrix $R_{(m-1) \times T}$. Equation (1) can be implemented in bit-level, digit-level, and bit-parallel, depending on the available resources. In this paper, we choose digit-level parallel-in parallel-out (DL-PIPO) architecture to perform field level multiplications of ECC. The time complexity of the employed digit-level GNB multiplier is $T_A + (\lceil \log_2 T \rceil + \lceil \log_2(d+1) \rceil)T_X$, and its area complexity is dm AND gates and $\leq \frac{d(m-1)}{2}(T-1) + dm$ XOR gates. Note that T_A and T_X are the delays of an AND gate and a XOR gate, respectively.

2.3 Hybrid-Double Multiplication

Let A , B , and C be three field elements of $GF(2^m)$ represented by GNB. Double multiplication is the computation of $D = A \times B \times C$, $D \in GF(2^m)$. In [29], the idea of hybrid double multiplication which requires the same number of clock cycles as the one for one multiplication is initially proposed. Then, in [23], a digit-level multiplier architecture using GNB has been proposed which computes D with the latency as the one required for one multiplication. The structure of the hybrid-double multiplier is depicted in Fig. 1. As one can see, it is composed of a digit-level parallel-in serial-out (DL-PISO) GNB multiplier and a digit-level serial-in parallel-out (DL-SIPO) GNB multiplier. For the operation of the hybrid-double multiplier, the registers $\langle X \rangle$, $\langle Y \rangle$, and $\langle F \rangle$ should be pre-loaded with the coordinates of operands A , B , and C , respectively, and the register $\langle Z \rangle$ should be cleared to $0 \in GF(2^m)$. The entire hybrid-double multiplier performs two multiplications simultaneously, where the results are available in parallel after $M = \lceil \frac{m}{d} \rceil + 1$ clock cycles, where d is the digit-size and one clock cycle is used for storing and loading from the middle d -bit

TABLE 1

Comparison of the Upper Bound of the Space and Time Complexities of Different Digit-Level Type T GNB Multipliers over $GF(2^m)$

Multiplier Architecture	digit size	Latency $q = \lceil \frac{m}{d} \rceil$	# AND gates	# XOR ¹ gates	# Reg.	Critical-Path delay	Output
DL-PISO [26]	d	q	dm	$\leq (d(m-1) - \frac{d(d-1)}{2}) \times (T-1) + d(m-1)$	$2m$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil) T_X$	Serial
DL-PIPO [26]	d	q	dm	$\leq \frac{d(m-1)}{2} \times (T-1) + dm$	$3m$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2(d+1) \rceil) T_X$	Parallel
DL-SIPO [23]	d	q	dm	$\leq (d(m-1) - \frac{d(d-1)}{2}) \times (T-1) + dm$	$2m$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2(d+1) \rceil) T_X$	Parallel
DL-PISO [26]	$2d$	$\lceil \frac{q}{2} \rceil$	$2dm$	$\leq (2d(m-1) - d(2d-1)) \times (T-1) + 2d(m-1)$	$2m$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil) T_X$	Serial
DL-PIPO [26]	$2d$	$\lceil \frac{q}{2} \rceil$	$2dm$	$\leq d(m-1) \times (T-1) + 2dm$	$3m$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2(2d+1) \rceil) T_X$	Parallel
DL-SIPO [27]	$2d$	$\lceil \frac{q}{2} \rceil$	$2dm$	$\leq (2d(m-1) - d(2d-1)) \times (T-1) + 2dm$	$2m$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2(2d+1) \rceil) T_X$	Parallel
DL-HD [23]	d	$q+1$	$2dm$	$\leq 2(d(m-1) - \frac{d(d-1)}{2}) \times (T-1) + 2dm - d$	$4m+d$	$T_A + (\lceil \log_2 T \rceil + \lceil \log_2 m \rceil) T_X$	Parallel

register. The critical-path delay of the hybrid-double multiplier is equal to the maximum of the delays for the DL-PISO (t_s) and DL-SIPO (t_p) multipliers i.e., $t_{CP} = \max\{t_p, t_s\}$. Then, one can obtain the time of multiplication as $T = t_{CP} \times M$ [23].

In Table 1, we compare the time and space complexities of different single digit-level multipliers (including DL-PIPO, DL-PISO, and DL-SIPO) with the digit size d and $2d$ with the hybrid-double multiplier with the digit size d over GNB. As one can see, the presented hybrid-double multiplier has smaller space complexity in comparison to the DL-SIPO and DL-PISO multipliers and is comparable to the DL-PIPO multiplier employing $2d$ digit size. Also, its time complexity is equal to the DL-PISO multiplier and is almost similar to the DL-PIPO and DL-SIPO multipliers depending to the digit size. We note that for practical implementations, increasing the digit size drastically reduces the maximum operating clock frequencies and hence employing hybrid-double multipliers seems to be better choice to point multiplication on elliptic curves. Therefore, in the next sections we employ the presented hybrid-double multiplier to reduce the latency of point multiplication on binary elliptic curves.

2.4 Trace and Quadratic Equation Solution

The trace function $\text{Tr}: GF(2^m) \rightarrow GF(2)$ is a linear map and for an element $A = (a_0, a_1, \dots, a_{m-1}) \in GF(2^m)$ is defined as $\text{Tr}(A) = \sum_{i=0}^{m-1} A^{2^i} \in \{0, 1\}$. For normal basis, the trace of element A can be computed as $\text{Tr}(A) = \sum_{i=0}^{m-1} a_i$, which is bit-wise XOR operation of all bits of vector A .

The quadratic equation $X^2 + X = A$ for $X = (x_0, x_1, \dots, x_{m-1}) \in GF(2^m)$ has a solution if and only if $\text{Tr}(A) = 0$, and hence if X is a solution, then $X + 1$ is a solution. In normal basis the solution can be found bit-wise. However, in polynomial basis it is complicated and needs half-trace computations which requires $m-1$ squarings and $(m-1)/2$ additions [30]. The cost of solving quadratic equation using normal basis is only $m-2$ additions.

2.5 Arithmetic over Binary Edwards and Generalized Hessian Curves

It is well known that a non-supersingular binary generic (short Weierstraß) elliptic curve can be defined by a set of

points (x, y) and a special point at infinity \mathcal{O} (group identity) that satisfy the following equation

$$E_{a,b}/GF(2^m) : y^2 + xy = x^3 + ax^2 + b, \quad (3)$$

where $a, b \in GF(2^m)$ and $b \neq 0$ [30]. Binary Edwards curves belong to a special class of generic elliptic curves defined over binary field when $m \geq 3$ [18]. The merit of binary Edwards curves over generic curves is that their point addition formulas are complete and their implementations are comparable with the traditional ones of [21].

Definition 1 ([18]). Let \mathbb{K} be a finite field of characteristic two, i.e., $\text{char}(\mathbb{K}) = 2$ and d_1 and d_2 be the elements of \mathbb{K} with $d_1 \neq 0$ and $d_2 \neq d_1^2 + d_1$. The binary Edwards curve with coefficients d_1 and d_2 is the affine curve

$$E_{B,d_1,d_2}/GF(2^m) : d_1(x+y) + d_2(x^2+y^2) = xy + xy(x+y) + x^2y^2, \quad (4)$$

where $d_1, d_2 \in GF(2^m)$.

Given a point $P = (x, y)$, its negation, $-P$, is obtained as (y, x) which has no cost [18]. The point $(0, 0)$ is the neutral element and $(1, 1)$ has order 2 [18]. The binary Edwards curves are complete if $\text{Tr}(d_2) = 1$, i.e., d_2 cannot be written as $e^2 + e$ for any e in \mathbb{K} , where Tr is the absolute trace of $GF(2^m)$ over $GF(2)$ [18].

Definition 2 ([19]). Let c and d be elements of \mathbb{K} such that $c \neq 0$ and $d^3 \neq 27c$. The generalized Hessian curve (GHCs) $H_{c,d}$ over \mathbb{K} is defined by the equation

$$H_{c,d}/GF(2^m) : x^3 + y^3 + c = dxy, \quad (5)$$

where $c = 1$ results in a Hessian curve, i.e., H_d . Note that the GHCs are complete if and only if c is not a cube in \mathbb{K} .

The standard formulas on generic curves in [21] and [31] fail in computing addition of two points on curves if one of the points or their addition is at infinity. These possibilities should be tested before designing an elliptic curve cryptosystem. Note that point addition and doubling formulas on binary Edwards and generalized Hessian curves work for all input pairs, this characteristic is called *completeness*.

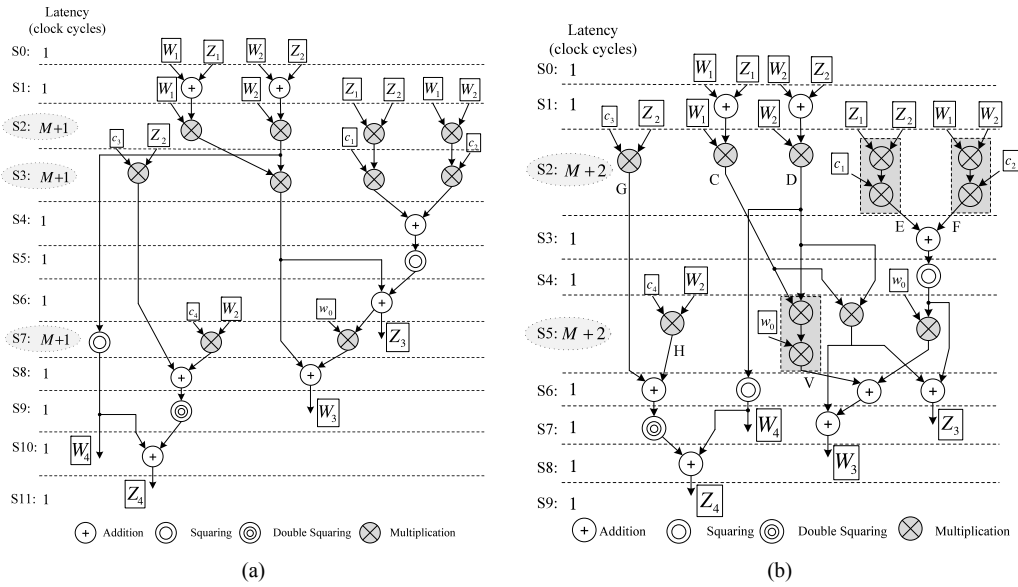


Fig. 2. Data dependency graph for fast computation of combined PA and PD for binary Edwards curves (a): employing four DL-PIPO multipliers. (b): employing three DL-PIPO and two hybrid-double multipliers. Note that $c_1 = \sqrt{d_1}$, $c_2 = \sqrt{d_2/d_1 + 1}$, $c_3 = \sqrt{c_1}$, and $c_4 = \sqrt{c_2}$.

3 NEW TECHNIQUE FOR LATENCY REDUCTION IN POINT MULTIPLICATION

The efficiency of point multiplication, i.e., $Q = k \cdot P$, depends on finding the minimum number of steps to reach kP from a given point P . Point multiplication on binary Edwards and generalized Hessian curves is efficient when one uses w -coordinate differential addition and doubling formulas and Montgomery's ladder. In this case, for every bit of scalar k , one needs to perform combined point addition and doubling. In [17], parallel computation of point multiplication on binary Edwards and generalized Hessian curves have been investigated employing two finite field multipliers considering area-time trade-offs. In [17], the maximum parallelization over those curves is not considered for the purpose of high speed implementations. This idea can be explored deeply to investigate the area-time trade-offs for different applications. Another idea is to optimize the size of the finite field multipliers in terms of different digit sizes. In this effect, one can employ different digit sizes (some multipliers with smaller digit sizes and some with larger digit sizes) and further reduce the occupied area. We note that in theory one may keep the latency unchanged and reduce the required area but in practice the multipliers with the larger digit sizes dominate the critical path and reduce the maximum operating clock frequencies.

In this section, we first study the maximum number of multipliers to achieve high speed computations of point multiplication. Then, we employ a new hybrid-double multiplier presented in Section 2.3. This reduces the overall latency of point multiplication on BECs and GHCs.

3.1 Binary Edwards Curves

In binary Edwards curves, mixed w -coordinate has been incorporated to compute mixed differential point addition and point doubling (PD) for Montgomery point multiplication [18]. Differential addition [32] is the computation of $Q + P$, given points of Q , P , and $Q - P$. Let us assume w to

be a linear and symmetric function in terms of the coordinates x and y of the point P and is defined as $w_i = x_i + y_i$, where $w(P) = w(-P)$. Bernstein et al. [18] have defined w -coordinate differential addition for computing $w(Q + P)$ given $w(Q)$, $w(P)$, and $w(Q - P)$. Similarly, the w -coordinates differential doubling is the computation of $w(2P)$ given $w(P)$. Therefore, using w -coordinates of differential addition and doubling formulas, $w((2n + 1)P)$ and $w(2nP)$ can be computed given $w(nP)$ and $w((n + 1)P)$, recursively [18]. The combined point addition and doubling formula with $d_1 \neq d_2$ is given as

$$\begin{aligned} C &= W_1 \cdot (Z_1 + W_1), D = W_2 \cdot (Z_2 + W_2), E = Z_1 \cdot Z_2, \\ F &= W_1 \cdot W_2, V = C \cdot D, W_3 = V + w_0 \cdot Z_3, \\ Z_3 &= V + (c_1 \cdot E + c_2 \cdot F)^2, W_4 = D^2, \\ Z_4 &= W_4 + ((c_3 \cdot Z_2 + c_4 \cdot W_2)^2)^2, \end{aligned} \quad (6)$$

where $w_0 = x_0 + y_0$, $c_1 = \sqrt{d_1}$, $c_2 = \sqrt{d_2/d_1 + 1}$, $c_3 = \sqrt{c_1}$, and $c_4 = \sqrt{c_2}$. Also, $P_1 = (W_1, Z_1)$, $P_2 = (W_2, Z_2)$, $P_3 = P_1 + P_2$, and $P_4 = 2P_2$. As seen from the above formulations, the cost of the combined PA and PD operations is $10M$, where M is the cost of a multiplication. For achieving highest degree of parallelization, we employ maximum number of parallel multipliers. The data dependency graph is depicted in Fig. 2a employing four DL-PIPO multipliers. In Steps S2 and S3 of Fig. 2a, four DL-PIPO multipliers operate in parallel and in S7, only two multipliers perform the operation. Therefore, the multiplier utilization is $\frac{(4+4+2)}{3 \times 4} \times 100 = 83.33\%$. As one can see, the smallest latency for the combined PA and PD is achieved by employing four multipliers as $3M + 12$. Note that employing more than four multipliers does not reduce the latency due to data dependencies.

For binary Edwards curves, we modify the combined PA and PD formulations in (6) in such a way to incorporate the proposed hybrid-double multiplier and remove the data dependencies to further reduce the number of multipliers in

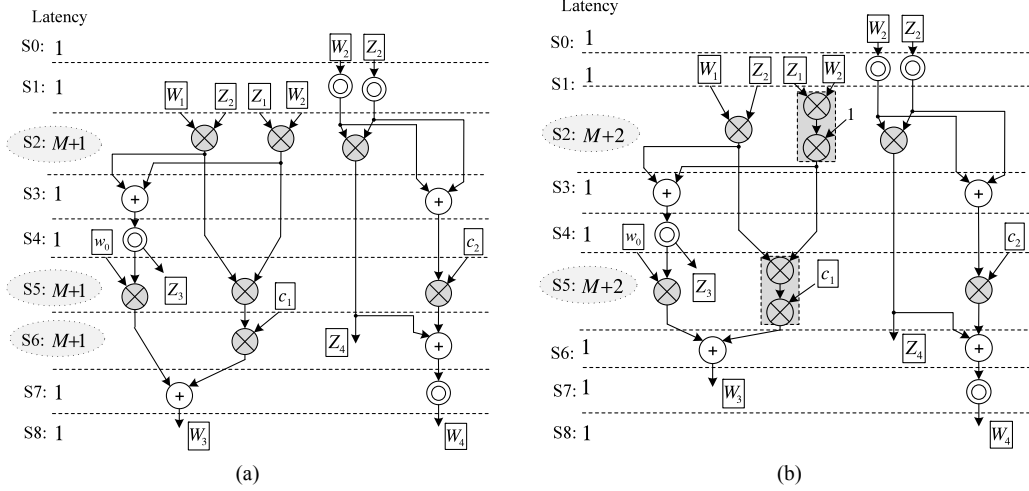


Fig. 3. Data dependency graph for generalized Hessian curves with $c_1 = d^3$, and $c_2 = \frac{1}{\sqrt{d^3}}$, (a) employing three single DL-PIPO multipliers and (b) employing two single DL-PIPO multipliers and one hybrid-double multiplier.

the data path (i.e., reduce the latency). The modified formulations are as follows:

$$\begin{aligned}
 C &= W_1 \cdot (Z_1 + W_1), \quad D = W_2 \cdot (Z_2 + W_2), \\
 E &= Z_1 \cdot Z_2 \cdot c_1, \quad F = W_1 \cdot W_2 \cdot c_2, \\
 G &= c_3 \cdot Z_2, \quad V = C \cdot D \cdot w_0, \\
 Z_3 &= C \cdot D + (E + F)^2, \quad H = c_4 \cdot W_2, \\
 W_3 &= V + (E + F)^2 \cdot w_0 + CD, \\
 W_4 &= D^2, \quad Z_4 = W_4 + ((G + H)^2)^2.
 \end{aligned} \tag{7}$$

The corresponding data dependency graph for the modified formulations for the combined PA and PD is illustrated in Fig. 2b for binary Edwards curves. As shown in this figure, we employ the proposed hybrid-double multiplier in Steps S2 and S5. In Step S2, we combined computation of field multiplications by constants (c_1 and c_2) and performed them in one step with the latency of $M + 2$ using two hybrid-double multipliers. Three traditional digit-level PIPO multipliers are also operating in this step. In Step S5, we modified formulation of the PA operation in computing (W_3 and Z_3) to take the advantage of the hybrid-double multiplier as much as possible. As one can see, in this step the computation of $V = C \cdot D \cdot w_0$ is done using one hybrid-double multiplier with the latency of $M + 2$. As a result, the latency of the overall point multiplication over binary Edwards curves is reduced to $2M + 12$. Therefore, applying the proposed technique reduces the latency of computation of combined PA and PD by about 33 percent. We further note that the proposed approach is a new method to reduce the latency of point multiplication while parallelization fails due to data dependency. Therefore, one can achieve higher speeds in computing of point multiplication for high speed applications.

3.2 Generalized Hessian Curves

Similar to BECs, mixed w -coordinate has been incorporated to compute mixed differential PA and PD for Montgomery

point multiplication as follows [19]:

$$\begin{aligned}
 A &= W_1 \cdot Z_2, \quad B = W_2 \cdot Z_1, \quad Z_4 = W_2^2 \cdot Z_2^2, \\
 Z_3 &= (A + B)^2, \quad D = W_2^2 + Z_2^2, \quad E = w_0 \cdot Z_3, \\
 F &= (A \cdot B), \quad G = D \cdot c_2, \quad H = F \cdot c_1, \\
 W_3 &= E + H, \quad W_4 = (Z_4 + G)^2,
 \end{aligned} \tag{8}$$

where $c_1 = d^3$, and $c_2 = \frac{1}{\sqrt{d^3}}$. As one can figure out the cost of combined PA and PD is $7M$. In Fig. 3a, the data dependency graph for combined PA and PD is depicted employing three parallel multipliers. As illustrated in this figure, the latency is $3M + 9$ and employing more than three multipliers will not reduce the latency. This is the maximum possible number of parallel multipliers that can be used to accelerate the computation of combined PA and PD. In Fig. 3b, we employ a hybrid-double multiplier to reduce the latency. First, as one can see in Step S5 one should compute $A \cdot B \cdot c_1$ and employing a hybrid multiplier can compute it in $M + 2$ clock cycles. Moreover, in this step we need two parallel multipliers as well. In Step S2, we employ a hybrid multiplier having one of its inputs to be 1 to compute a single multiplication. This increases the latency 1 clock cycle but results in reducing the area. Therefore, the utilization factor for single and hybrid-double multipliers is 100 percent. Further, we reduce the latency to $2M + 11$ clock cycles as shown in Fig. 3b. Hence, the architecture of point multiplication on generalized Hessian curves requires two single multipliers and a hybrid-double multiplier. In Table 2, we summarize the cost of combined point addition and doubling for binary Edwards and generalized Hessian curves. As one can see, the costs are given in terms of number of single digit-level and hybrid-double multipliers. The cost (latency in terms of number of multipliers in the critical-path) reduction for both binary Edwards and generalized Hessian curves is about 33 percent.

TABLE 2

Comparison of the Cost of Combined Point Addition and Doubling on BEC and GHC with and without Employing Hybrid Multipliers, where \mathcal{M} is the Number Single Digit-lezel Multipliers and \mathcal{H} is the Number of Hybrid-Double Multipliers

Curve	# of Mults.	Figure	Multiplier Utilization	Cost of PA and PD
BEC	$\mathcal{M} = 4,$ $\mathcal{H} = 0$	Fig. 2a	$\frac{(4+4+2)}{3 \times 4} \times 100$ $= 83.33\%$	$3M$
	$\mathcal{M} = 3,$ $\mathcal{H} = 2$	Fig. 2b	$\frac{(5+4)}{10} \times 100$ $= 90\%$	$2M$
GHC	$\mathcal{M} = 3,$ $\mathcal{H} = 0$	Fig. 3a	$\frac{(3+3+1)}{3 \times 3} \times 100$ $= 78\%$	$3M$
	$\mathcal{M} = 2,$ $\mathcal{H} = 1$	Fig. 3b	$\frac{(3+3)}{3 \times 2} \times 100$ $= 100\%$	$2M$

It should be noted that the values of $M = \lceil \frac{m}{d} \rceil + 1$ depend to the digit-size and should be chosen carefully. For smaller ds , the hybrid-double multiplier is not efficient (as shown in [23], Table 4). On the other hand, for larger ds the latency reduction for the computation of PA and PD in the main loop of point multiplication is not significant. Therefore, one needs to do a trade-off between the latency reduction for multiplier and the whole architecture of ECC choosing larger digit sizes. It is worth mentioning that the occupied area and maximum operating clock frequency should be considered when one chooses the digit-size.

4 PROPOSED CRYPTO-PROCESSORS FOR POINT MULTIPLICATION

The crypto-processors for point multiplication are composed of four main units including finite field arithmetic unit (FAU), control unit, register file, and conversion unit. In FAU of BEC, we employ three single digit-level parallel-in parallel-out GNB multipliers and two hybrid-double multipliers ($\mathcal{M} = 3$ and $\mathcal{H} = 2$) to perform double multiplications as explained in Section 3. In FAU of GHC, two single DL-PIPO multipliers and a hybrid-double multiplier ($\mathcal{M} = 2$ and $\mathcal{H} = 1$) are employed. The architectures of the proposed crypto-processors are depicted in Fig. 4. In FAU, squaring is simply rewiring without any time and area cost in normal basis. Addition is bit-wise XORing of the coordinates of the two operands. This requires only one clock cycle to store the results in the registers. In a digit-level parallel-in parallel-out GNB multiplier both input operands, A and B should be

present through multiplication process and the results will be available in parallel after $M = q + 1$, $q = \lceil \frac{m}{d} \rceil$, $1 \leq d \leq m$, clock cycles. Where one clock cycles is due to pipelining the multiplier. For the given field size $m = 163$, digit-size d is chosen in such a way to reduce the latency while increasing d . Therefore, we choose the digit sizes from the set $d \in \{11, 21, 33\}$ for $m = 163$. As explained in Section 2, the hybrid-double multiplier computes a double multiplication with the same number of clock cycles as the one requires for a single multiplication. The single and hybrid-double multipliers are efficiently pipelined (one level) to achieve maximum operating clock frequencies. The architectures of proposed crypto-processors for different curves are different mainly in terms of number of employed single and hybrid-double multipliers, register file, and the control unit.

4.1 Control Unit

The control units are designed with finite state machine (FSM) to perform the point multiplication with other units. Scheduling the computation tasks is done by generating the signals and switching the operands for FAU. The intermediate results are stored in the register file. This eliminates the overhead of communication between memory and FAU. Several multiplexers are employed to choose appropriate registers and connect to the FAU. Also, for binary Edwards and generalized Hessian curves we obtained the affine coordinates employing linear half-trace and Itoh-Tsujii's scheme [33] which its implementation is very efficient using Gaussian normal basis as stated in Section 2.4.

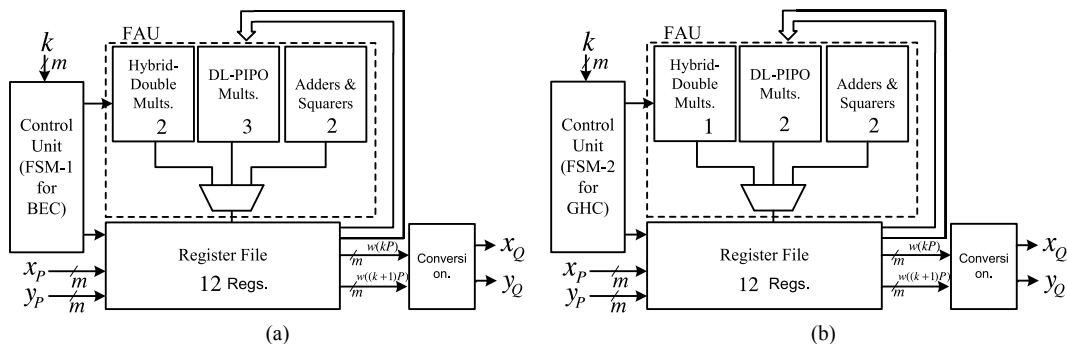


Fig. 4. The architecture of proposed crypto-processors for point multiplication on (a) binary Edwards and (b) generalized Hessian curves.

TABLE 3

Implementation Results of Point Multiplication on BECs (with $M = 3$ and $\mathcal{H} = 2$) and GHCs (with $M = 2$ and $\mathcal{H} = 1$) over $GF(2^{163})$ and $GF(2^{233})$ in Xilinx Virtex-4 XC4VLX160 FPGA

Implementation Results for BECs ($d_1 \neq d_2$) [This work]									
Field size	d	$q + 3$	Latency (L_{PM})	f_{max} (MHz)	# Slices	Area # LUTs	# FFs	Point Mult. [μs]	AT Area \times Time
$m = 163$	11	18	7,278	221.1	9,456	19,866	17,164	32.9	0.311
	21	11	4,849	218.3	16,806	38,464	26,481	22.2	0.373
	33	8	3,808	217.2	27,778	57,432	35,052	17.5	0.486
$m = 233$	13	21	11,532	212.8	15,832	25,358	25,881	54.1	0.857
	26	12	7,212	198.4	29,252	48,577	44,880	36.3	1.063
Implementation Results for GHCs [This work]									
Field size	d	$q + 3$	Latency (L_{PM})	f_{max} (MHz)	# Slices	Area # LUTs	# FFs	Point Mult. [μs]	AT Area \times Time
$m = 163$	11	18	6,871	223.6	5,526	10,991	10,588	30.8	0.170
	21	11	4,491	220.3	9,718	21,542	15,832	20.4	0.198
	33	8	3,471	218.2	15,992	31,814	20,573	15.9	0.254
$m = 233$	13	21	11,103	215.7	9,201	14,483	15,942	51.6	0.47
	26	12	6,791	205.1	16,940	28,145	26,068	33.1	0.56

4.2 Register File

The register file includes 12 m -bit registers to store inputs, outputs, curve parameters, and intermediate results for binary Edwards and generalized Hessian curves, respectively. For instance for binary Edwards curves, we have five intermediate results that need to be stored in the registers file, four registers to store internal input parameters (i.e., W_1, W_2, Z_1, Z_2), two registers to store coordinates of affine input point (i.e., x and y), and a shift register to store scalar k . The curve parameters are stored in a separate registers as to be fixed including d_1, d_2, c_1, c_2, c_3 and c_4 . Our proposed data dependency graphs are efficiently hand-optimized to ensure we occupy optimal number of registers. We employed flip-flops available in each slice to construct register files as they are available as a decentralized resources. These flip-flops are put close to their usage and hence we reduced the routing constraints. For instance the employed DL-PIPO and Hybrid-double multipliers require sequential computations (with shift and accumulate operations) and hence we were able to place the registers close to these operators. We note that these flip-flops are available without additional area requirements. The register file is composed of several multiplexers to chose appropriate registers and connect them to the arithmetic unit.

5 FPGA IMPLEMENTATIONS RESULTS AND COMPARISONS

The proposed architecture for point multiplication on BECs and GHCs are implemented on FPGA. To be able to make fair comparisons with the previous work available in the literature, we have chosen same platform for our implementations. The proposed architectures for BECs and GHCs (in Figs. 4a and 4b) are modeled in VHDL [34] and are synthesized using XST of Xilinx ISE version 12.1 design software and are implemented on Xilinx Virtex-4 XC4VLX160 FPGA. In terms of available resources it contains 67,584 Slices (135,168 LUTs and 135,168 FFs). The implementation results for different digit sizes d , $d \in \{11, 21, 33\}$ and $d \in \{13, 26\}$ are reported in Table 3 after place and route (PAR) for the proposed architectures over

$GF(2^{163})$ and $GF(2^{233})$. As one can see, for $m = 163$ the fastest point multiplications take 17.5 μs and 15.9 μs , and require 27,778 Slices and 15,992 Slices for BEC and GHC, respectively. For $m = 233$ the fastest point multiplication is computed in 36.3 and 33.1 μs occupying 29,252 Slices and 16,940 Slices, respectively. In Table 3, we also provide the results of area-time products as a parameter to measure the efficiency of the proposed architectures for practical applications. Further, as illustrated in Fig. 5 we plot the area-time for different digit sizes and compare it to the previous work presented in [17].

It should be noted that for any particular application the digit-size should be chosen in such a way to achieve highest performance considering the time-area trade-offs. As explained in the above sections, the novelty of the proposed architectures for point multiplication in this paper is based on employing a new hybrid-double multiplier architecture proposed in [23]. This scheme reduces the latency of point multiplication on previous work which parallelization fails due to data dependencies.

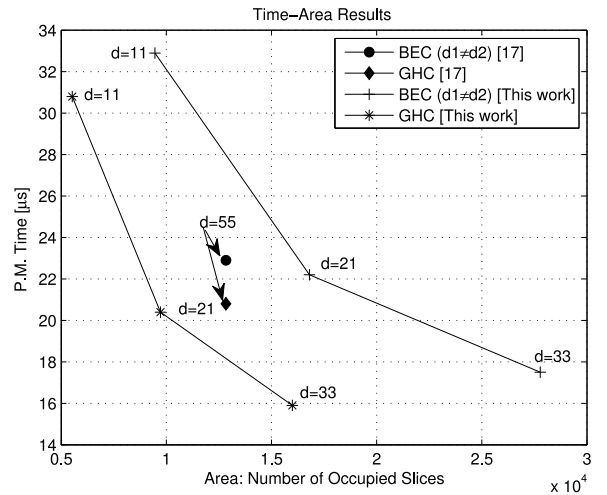


Fig. 5. Comparison of time-area results for the point multiplication on binary Edwards and generalized Hessian curves over $GF(2^{163})$ on Xilinx Virtex-4 XC4VLX160 FPGA.

TABLE 4
Comparison of ECC Implementations on FPGA over $GF(2^{163})$ and $GF(2^{233})$

Work ¹	Device	Basis	Curve	digit size (d)	Area	Time [μ s]
$GF(2^{163})$						
BKC [35]	Stratix II	NB	Koblitz	17	23,580 ALMs	9.48
BKC [15]	Stratix II	NB	Koblitz	24	26,381 ALMs	13.38
BGC [36]	Stratix II	NB	Generic	41	18,489 ALMs	51.56
BKC [36]	Stratix II	NB	Koblitz	41	19,498 ALMs	35.1
BKC [16]	Stratix II	GNB	Koblitz	41	23,084 ALMs	9.15
BGC [4]	Virtex-4	GNB	Generic	55	24,363 Slices	10.11
BEC ($d_1 \neq d_2$) [17]	Virtex-4	GNB	Edwards	55	12,834 Slices	23.3
GHC [17]	Virtex-4	GNB	Hessian	55	12,834 Slices	20.8
This work BEC ($d_1 \neq d_2$)	Virtex-4	GNB	Edwards	33	27,778 Slices	17.5
This work GHC	Virtex-4	GNB	Hessian	33	15,992 Slices	15.9
This work BEC ($d_1 \neq d_2$)	Virtex-7	GNB	Edwards	33	10,569 Slices	12.2
This work GHC	Virtex-7	GNB	Hessian	33	6,042 Slices	11.1
over $GF(2^{233})$						
BEC [12]	Virtex-4	PB	Edwards	–	21,816 Slices	190
This work BEC ($d_1 \neq d_2$)	Virtex-4	GNB	Edwards	26	29,252 Slices	36.3
This work GHC	Virtex-4	GNB	Hessian	26	16,940 Slices	33.1

1. BGC: binary generic curve, BKC: binary Koblitz curve, BEC: binary Edwards curve, GHC: generalized Hessian curve.

Note that the results presented here are comparable to the ones based on complete addition formulae i.e., BEC and GHC.

5.1 Discussion and Comparison

In this section, the implementation results of the proposed crypto-processors for point multiplication in this paper are compared to the counterparts in the literature as illustrated in Table 4. As explained in the previous sections, the novelty of the proposed architectures for point multiplication in this paper is based on employing a new hybrid-double multiplier architecture proposed in [23]. This scheme reduces the latency of point multiplication on previous work which parallelization fails due to data dependencies. The work presented in [17] and [12], are the only available works in the open literature on BECs and GHCs and hence we mainly compare our results to them on the same platform and field sizes. Comparisons to the other works which have not employed unified and complete formulae (i.e., implementations on BGCs and BKC) is presented only in terms of performance results.

The point multiplication architecture presented in [17], employs two finite field multipliers and investigates the time-area trade-offs and multiplier utilization. As reported in [17], the point multiplication takes 23.3 and 20.8 μ s employing two parallel finite field multipliers for binary Edwards and generalized Hessian curves, respectively.

In [36], Järvinen and Skyttä have employed four parallel normal basis multipliers for point multiplication on binary generic and Koblitz curves. The point multiplication requires 51.56 and 35.1 μ s for binary generic and Koblitz curves, respectively. Also, in [35], a new scheme has been proposed employing interleaving which reduces the time of point multiplication on Koblitz curves to 9.48 μ s on Altera StratixII. However, in our proposed architecture for point multiplication on Koblitz curves requires 8.6 μ s on the same platform. We note that one can reach higher speeds by combining our scheme with the one proposed in [35] utilizing data interleaving. In [13], Roy et al. proposed a parallel architecture for scalar multiplication on Koblitz

curves. They have employed a hybrid Karatsuba multiplier for field multiplications and have implemented their parallel architecture on Xilinx Virtex-5 FPGA. The point multiplication requires 12.1 μ s and occupies 12,430 Slices over $GF(2^{163})$. In [14], Rebeiro et al. proposed a high speed architecture based on pipelined hybrid Karatsuba multiplier (using polynomial basis) for computing point multiplication on binary generic curves. Similar to [17], they employed a LUT-based pipelining technique to achieve maximum operating clock frequency. Their architecture is coming with a careful data scheduling for computing differential point addition and doubling in the main loop of computing point multiplication. They have implemented their proposed architecture on different FPGA platforms. For instance, their architecture requires 9.7 μ s and occupies 8,070 Slices on Xilinx Virtex-4 over $GF(2^{163})$. In [11], Chatterjee and Sengupta. proposed a similar architecture for computing point multiplication on binary Huff curves which have been recently proposed by Devigne and Joye [37]. Their proposed architecture is efficiently implemented over $GF(2^{233})$ which is unified and complete. They have also investigated side channel resistivity of point multiplication architecture on binary Edwards curves implemented on Xilinx Virtex-4 over $GF(2^{233})$ in [12]. Their architecture requires 21,816 Slices and computes a single point multiplication in 190 μ s. This architecture operates in 47.3 MHz clock frequency which is fairly low due to the usage of Karatsuba multiplier and requires about 9,006 clock cycles in total. In addition, one should note that binary Edwards curves requires 10 multiplication operations and it seems that parallelization is more efficient as we have employed in our proposed scheme. We note that their proposed method is one step forward in investigating the side channel analysis of proposed architectures for point multiplication on binary Edwards curves which motivates more research in this direction for other curves such as generalized Hessian curves.

The point multiplication scheme proposed in [4] has been performed on binary generic curves over $GF(2^{163})$ employing three digit-serial GNB multipliers. The computation of point multiplication takes $10.11 \mu\text{s}$ on Xilinx Virtex-4 platform [4].

As summarized in Table 3, we have employed smaller digit sizes and reduced the latency of point multiplication based on the newly proposed crypto-processors. As a result, we have increased the speed of point multiplication computation on binary Edwards and generalized Hessian curves about 25 percent in comparison to the ones presented in [17]. As one can see in Fig. 5, our results for binary Edwards curves are faster than the previous work on the same platform and are more efficient in terms of time-area products for GNCs.

Note that the idea of employing hybrid-double multiplier to speed up point multiplication on binary curves can also be employed for the case when field elements are represented by polynomial basis. Therefore, similar improvements will be achieved for increasing the speed of point multiplication using any basis. One should note that the point multiplication on BECs and GNCs are complete and work for all pairs of inputs. However, for the other curves one needs to check points at infinity and avoid them for practical applications. Therefore, it is not fair to compare our implementation results to the counterparts using generic and Koblitz curves quantitatively. In addition, if security against side-channel attacks plays the main role, uniform formulas certainly do have an advantage which we have investigated in this work. Further, in this paper, our target is high-performance applications as we speed up the time of point multiplication at the cost of increasing the area. We stress that one could have used Karatsuba method for multiplication over $GF(2^m)$ for computing point multiplication, but as indicated in [38], for smaller field sizes digit-serial normal/polynomial basis multipliers can operate in higher clock frequencies occupying similar area in comparison to the Karatsuba multiplier.

6 CONCLUSIONS

In this paper, we have proposed a new scheme to speed up the point multiplication on binary Edwards and generalized Hessian curves employing a digit-level hybrid-double multiplier of [23] which performs two multiplications with the latency as the one requires for one multiplication. We have performed a data-flow analysis for point multiplication to understand how data has to move between the different logic and computational elements such as field multipliers, adders, and squarers. Then, we have investigated how the hybrid-double multiplier can be employed to reduce the latency of point multiplication. We have obtained that employing hybrid-double multiplier can reduce the latency of point multiplication on binary Edwards and generalized Hessian curves and speed up the computation time. We have evaluated the performance of the proposed crypto-processors for different digit sizes with implementing on FPGA and have reported the area and timing results. Our proposed architecture performs a point multiplication on binary Edwards and generalized Hessian curves 25 percent faster than the previous fastest results appearing in [17].

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their constructive comments. This work has been supported in part by a Natural Sciences and Engineering Council (NSERC) grants awarded to Arash Reyhani-Masoleh. Also, the first author would like to thank A. Ava-Azarderakhsh for kind support.

REFERENCES

- [1] V. S. Miller, "Use of elliptic curves in cryptography," in *Proc. Adv. Cryptol.*, 1986, pp. 417–426.
- [2] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, pp. 203–209, 1987.
- [3] U.S. Department of Commerce/NIST, "National Institute of Standards and Technology," Digital Signature Standard, FIPS Publications 186-2, Jan. 2000.
- [4] C. H. Kim, S. Kwon, and C. P. Hong, "FPGA implementation of high performance elliptic curve cryptographic processor over $GF(2^{163})$," *J. Syst. Arch.*, vol. 54, no. 10, pp. 893–900, 2008.
- [5] Y. K. Lee, K. Sakiyama, L. Batina, and I. Verbauwhede, "Elliptic-curve-based security processor for RFID," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1514–1527, Nov. 2008.
- [6] J. Adikari, V. S. Dimitrov, and L. Imbert, "Hybrid binary-ternary number system for elliptic curve cryptosystems," *IEEE Trans. Comput.*, vol. 60, no. 2, pp. 254–265, Feb. 2011.
- [7] W. Chelton and M. Benaissa, "Fast elliptic curve cryptography on FPGA," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 2, pp. 198–205, Feb. 2008.
- [8] J. Adikari, V. Dimitrov, and K. Jarvinen, "A fast hardware architecture for integer to τ -NAF conversion for koblitz curves," *IEEE Trans. Comput.*, vol. 61, no. 5, pp. 732–737, May 2011.
- [9] M. Keller, A. Byrne, and W. P. Marnane, "Elliptic curve cryptography on FPGA for low-power applications," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 1–20, 2009.
- [10] G. Sutter and J. Deschamps and J. Imana, "Efficient elliptic curve point multiplication using digit serial binary field operations," *IEEE Trans. Ind. Electron.*, vol. 60, no. 1, pp. 217–225, Jan. 2013.
- [11] A. Chatterjee and I. Sengupta, "High-speed unified elliptic curve cryptosystem on FPGAs using binary huff curves," in *Proc. 16th Int. Symp. Prog. VLSI Des. Test*, 2012, pp. 243–251.
- [12] A. Chatterjee and I. Sengupta, "Design of a high performance binary Edwards curve based processor secured against side channel analysis," *Integr., VLSI J.*, vol. 45, no. 3, pp. 331–340, 2012.
- [13] S. Roy, C. Rebeiro, and D. Mukhopadhyay, "A parallel architecture for Koblitz curve scalar multiplications on FPGA platforms," in *Proc. 15th Euromicro Conf. Digit. Syst. Des.*, 2012, pp. 553–559.
- [14] C. Rebeiro, S. S. Roy, and D. Mukhopadhyay, "Pushing the limits of high-speed $GF(2^m)$ elliptic curve scalar multiplication on FPGAs," in *Proc. Cryptographic Hardware Embedded Syst.*, 2012, pp. 494–511.
- [15] V. S. Dimitrov, K. U. Jarvinen, M. J. Jacobson Jr, W. F. Chan, and Z. Huang, "Provably sublinear point multiplication on koblitz curves and its hardware implementation," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1469–1481, Nov. 2008.
- [16] R. Azarderakhsh and A. Reyhani-Masoleh, "High-performance implementation of point multiplication on koblitz curves," *IEEE Trans. Circuits Syst. II*, vol. 60-II, no. 1, pp. 41–45, Jan. 2013.
- [17] R. Azarderakhsh and A. Reyhani-Masoleh, "Efficient FPGA implementations of point multiplication on binary Edwards and generalized Hessian curves using Gaussian normal basis," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 8, pp. 1453–1466, Aug. 2012.
- [18] D. Bernstein, T. Lange, and R. Farashahi, "Binary Edwards curves," in *Proc. Workshop Cryptographic Hardware Embedded Syst.*, 2008, vol. 5154, pp. 244–265.
- [19] R. Farashahi and M. Joye, "Efficient arithmetic on Hessian curves," in *Proc. 13th Int. Conf. Pract. Theory Public Key Cryptography*, 2010, pp. 243–260.
- [20] U. Kocabas, J. Fan, and I. Verbauwhede, "Implementation of binary Edwards curves for very-constrained devices," in *Proc. 21st Int. Conf. Appl.-Specific Syst. Arch. Processors*, 2010, pp. 185–191.
- [21] J. López and R. Dahab, "Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation," in *Proc. Workshop Cryptographic Hardware Embedded Syst.*, 1999, pp. 316–327.
- [22] R. Cheung, N. Telle, W. Luk, and P. Cheung, "Customizable elliptic curve cryptosystems," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 13, no. 9, pp. 1048–1059, Sep. 2005.

- [23] R. Azarderakhsh and A. Reyhani-Masoleh, "Low complexity multiplier architectures for single and hybrid-double multiplications in Gaussian normal bases," *IEEE Trans. Comput.*, vol. 62, no. 4, pp. 744–775, Apr. 2013.
- [24] A. Menezes, I. Blake, S. Gao, R. Mullin, S. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*. Norwell, MA, USA: Kluwer, 1993.
- [25] *IEEE Standard Specifications for Public-Key Cryptography*, IEEE Std 1363-2000, Jan. 2000.
- [26] A. Reyhani-Masoleh, "Efficient algorithms and architectures for field multiplication using Gaussian normal bases," *IEEE Trans. Comput.*, vol. 55, no. 1, pp. 34–47, Jan. 2006.
- [27] R. Azarderakhsh and A. Reyhani-Masoleh, "A modified low complexity digit-level Gaussian normal basis multiplier," in *Proc. 3rd Int. Workshop Arithmetic Finite Fields*, 2010, vol. 6087, pp. 25–40.
- [28] J. Massey and J. Omura, "Computational method and apparatus for finite arithmetic," US Patent 4 587 627, 1986.
- [29] A. Reyhani-Masoleh, "A new bit-serial architecture for field multiplication using polynomial bases," in *Proc. 10th Int. Workshop Cryptographic Hardware Embedded Syst.*, 2008, vol. 5154, pp. 300–314.
- [30] D. Hankerson, S. Vanstone, and A. Menezes, *Guide to Elliptic Curve Cryptography*. New York, NY, USA: Springer-Verlag, 2004.
- [31] M. Stam, "On montgomery-like representations for elliptic curves over $GF(2^k)$," in *Proc. 3rd Int. Conf. Pract. Theory Public Key Cryptography*, 2003, pp. 240–254.
- [32] P. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Math. Comput.*, vol. 48, pp. 243–264, 1987.
- [33] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases," *Inform. Comput.*, vol. 78, no. 3, pp. 171–177, 1988.
- [34] F. J. Azcondo, A. de Castro, and C. Branas, "Course on digital electronics oriented to describing systems in VHDL," *IEEE Trans. Ind. Electron.*, vol. 57, no. 10, pp. 3308–3316, Oct. 2010.
- [35] K. Järvinen and J. Skyttä, "Fast point multiplication on koblitz curves: Parallelization method and implementations," *Microprocessors Microsyst.*, vol. 33, no. 2, pp. 106–116, 2009.
- [36] K. Järvinen and J. Skyttä, "On parallelization of high-speed processors for elliptic curve cryptography," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 16, no. 9, pp. 1162–1175, Sep. 2008.
- [37] J. Devigne and M. Joye, "Binary huff curves," in *Proc. Topics Cryptol. Cryptographers' Track RSA Conf.*, 2011, pp. 340–355.
- [38] M. Morales-Sandoval, C. F. Uribe, R. Cumplido, and I. Algreto-Badillo, "An area/performance trade-off analysis of a $GF(2^{m1})$ multiplier architecture for elliptic curve cryptography," *Comput. Elect. Eng.*, vol. 35, no. 1, pp. 54–58, 2009.



Reza Azarderakhsh received the BSc degree in electrical and electronic engineering in 2002, the MSc degree in computer engineering from the Sharif University of Technology in 2005, and the PhD degree in electrical and computer engineering from the University of Western Ontario in 2011. He was with the Department of Electrical and Computer Engineering, University of Western Ontario, as a Limited Duties Instructor, in 2011. He was a recipient of the Natural Sciences and Engineering Research Council of Canada (NSERC) Post-Doctoral Research Fellowship in 2012. He has been an NSERC post-doctoral research fellow with the Center for Applied Cryptographic Research at the Department of Combinatorics and Optimization, University of Waterloo. Currently, he is a faculty member with the Department of Computer Engineering at Rochester Institute of Technology. His current research interests include finite field and its application, high-performance computation, elliptic curve cryptography, and pairing based cryptography. He is a member of the IEEE.



Arash Reyhani-Masoleh received the BSc degree in electrical and electronic engineering from the Iran University of Science and Technology in 1989, the MSc degree in electrical and electronic engineering from the University of Tehran in 1991, both with the first rank, and the PhD degree in electrical and computer engineering from the University of Waterloo in 2001. From 1991 to 1997, he was with the Department of Electrical Engineering, Iran University of Science and Technology. From June 2001 to September 2004, he

was with the Centre for Applied Cryptographic Research, University of Waterloo, where he was awarded a Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowship in 2002. In October 2004, he joined the Department of Electrical and Computer Engineering, University of Western Ontario, London, Canada, where he is currently a tenured associate professor. His current research interests include fault-tolerant computing, algorithms and VLSI architectures for computations in finite fields, cryptography, and error control coding. He has been awarded a NSERC Discovery Accelerator Supplement (DAS) in 2010. Currently, he serves as an associate editor for *Integration, the VLSI Journal* (Elsevier). He is a member of the IEEE and the IEEE Computer Society.

▷ For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.