Abstract

In this paper a processor that supports elliptic curve cryptographic applications over $GF(2^m)$ is proposed. The proposed structure is capable of calculating point multiplication and addition using a single coordinate to contain the point information. This compression allows for a better usage of the bandwidth resources. For the point multiplication procedure, all coordinate pre-calculations are completely avoided. This design was successful prototyped on a reconfigurable device for the field $GF(2^{163})$. Experimental results suggest that point multiplication can be performed in 144 $\mu$s and point affine addition in 1.02 $\mu$s. Comparing with the related work, a 5 times speedup is obtained for point addition and multiplication. The presented design offers a well balanced area-time performance when compared with existent elliptic curve point multiplication specific processors.

1. Introduction

Since Elliptic Curve Cryptography (ECC) was proposed it has been gathering increasing importance. Other algorithms, such as RSA, have reached a higher popularity and are now widely used. However, there is a new set of circumstances which motivates the need to research a different cryptosystem, namely the increasing use of portable devices, with constrained computing and power resources, and the increasingly more valuable bandwidth and memory availability, which demands lower sized keys. ECC appears as an interesting alternative to RSA, since its arithmetic is computationally more efficient and presents higher security per key bit.

The finite field $GF(2^m)$ has been leading to more efficient ECC hardware solutions, due to its mathematical properties. Our design differs from the related state of the art in the sense that only one coordinate of an elliptic point is used to compute point multiplication. This method allows for higher performance than the related art. This increased performance is achieved by the use of projective coordinates and a careful operation scheduling. The proposed processor also implements point affine addition starting from only one coordinate, supporting the computation of digital signature algorithms and cipher/decipher procedures.

The implementation of the proposed elliptic curve processor suggests that ECC point multiplication and addition can be performed in 144 $\mu$s and 1.02 $\mu$s, respectively, for the field $GF(2^{163})$, using 10,488 slices. Higher performance, regarding, the state of the art, for equivalent FPGA platforms is achieved.

The following sections are organized as follows. Section 2 presents the ECC processor characteristics. In Section 3 implementation and experimental results are presented and compared with the related state of the art. Concluding remarks on the proposed ECC design over $GF(2^m)$ are presented in Section 4.

2. ECC Processor Implementation

The steps taken by two individuals in order to communicate a secret through an unsecured channel using ECC can be found in [7]. In this protocol both parts perform the same computation apart from the signal of the point multiplication. To encrypt the message $M$, $C = M + k_1(k_2G)$ is computed and to decrypt $M = C - k_2(k_1G)$ is computed, with $k_i$ the private keys and $k_iG$ the public keys of the two individuals. To implement this and other protocols, like digital signatures, the elliptic curve point addition is used [1]. In this work, the following elliptic curve is used:

$$y^2 + xy = x^3 + ax^2 + b, \quad a, b \in GF(2^m).$$

The following describes the proposed structure, capable of performing elliptic curve point multiplication and addition. The design has been oriented towards the Xilinx VIRTEX 4 reconfigurable technology. An overview of the proposed structure is depicted in Figure 1. The correspondent operation schedule is presented in Table 1. The developed structure is able to perform point multiplication and addition with only one coordinate, as described in [1]. The presented structure was implemented for the field $GF(2^{163})$, supported over a polynomial basis. It can however be gen-
eralized for any odd field size. The computation was parallelized in order to perform three multiplications simultaneously, increasing the point multiplication performance. Parameters $a$ and $b$ from the elliptic curve described as in (1) are used, as well as a pre-computed parameter $d$ such that $d^3 = b$. Squaring units are also used, designated by $w^2$. The second most significant bit of the register $Scalar$ as designated by $k$. When transmitting/receiving a secret that is mapped in more than one elliptic curve point, the point addition initialization steps in Table 1 are performed only once, after the multiplication of the local private key with the other part public key. There is a bit that controls the complementation of this last operation trace $T(y_R/x_R)$. This bit also controls the calculus of the $y_R$ coordinate of a point or the $y$ coordinate of its inverse, the later used in the decryption process. The following presents each individual unit.

**Field multiplication unit:** This unit ($mult_i$) is obtained from the parallelization of the method presented in [5]. In order to achieve a good compromise between the available area and performance, a 4 level parallelization is used.

\[
Z_i(x) = b_0 A(x) + b_4 x^4 A(x) + \ldots + b_{160} x^{160} A(x);
\]

\[
Z_1(x) = x \left[ b_1 A(x) + b_5 x^4 A(x) + \ldots + b_{161} x^{160} A(x) \right];
\]

\[
Z_2(x) = x^2 \left[ b_2 A(x) + b_6 x^4 A(x) + \ldots + b_{162} x^{160} A(x) \right];
\]

\[
Z_3(x) = x^3 \left[ b_4 A(x) + b_7 x^4 A(x) + \ldots + b_{159} x^{156} A(x) \right];
\]

with $Z(x) = Z_0(x) + Z_1(x) + Z_2(x) + Z_3(x)$.

**Field division unit:** The division/inversion unit ($div$) is implemented directly from the Brunn algorithm discussed in [3]. The resulting structure is depicted in Figure 2.

**Trace of multiplication unit:** This unit calculates the trace of a multiplication from the input operands, without calculating the product itself. The trace vector presented in [1] is sparse if a polynomial basis is being used. For the field parameters used in this implementation only two non zero bits exist. Thus, this unit only needs to calculate this two bits of the multiplication result, which index correspond
to the non zero trace vector’s index. The computation of the dependencies of those two bits is performed taking into account that:

$$Z(x) = \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} a_k b_l x^{k+l}$$

(3)

Particularly in this implementation, the non zero index of the trace vector are 0 and 157. Thus the dependencies for these two bits are the values $k, l$ such that $x^{k+l}$ would have any term $x^0$ or $x^{157}$. If the trace vector is not sparse, this unit result can be alternatively calculated with a multiplier followed by the trace calculation unit.

**Root calculation unit:** This unit allows to calculate the value of $y/x$ from $w(x) = x + a + b/x^2$ and the trace $T(y/x)$ by solving a quadratic equation as described in [1]. After calculating $y/x$, the value of the $y$ coordinate can be obtained by multiplying $y/x$ with the $x$ coordinate. The resulting structure is depicted in Figure 3. The control signal implements a conditional addition operation with the associated term. This is useful when implementing the design for fields $GF(2^m)$ such that $m \equiv 3 \mod 4$. This means that there is an odd number of terms collected from the serial $w^2 - w^4 - w^2$ units. For our implementation, with $m = 163$, this conditional addition is used.

3 Implementation Results

The proposed ECC core was successfully implemented and thoroughly tested on a Xilinx VIRTEX 4 (XC4VSX35) prototyping platform. Table 2 depicts the area occupation and performance for the implementation results, after Place&Route, considering the field size $m = 163$. These results were obtained from a VHDL description of the design, using Synplify Premier (version 8.6.2) and Xilinx ISE (version 9.2.04i) tools. Considering the computation time of the field division, field multiplication, and the root calculation units, 410 clock cycles are required to perform a $y$ coordinate calculation. This computation is described in Table 1 as the point addition initialization. It is computed only once when a secret is mapped in more than one elliptic curve point.

In order to compare the related state of the art with the work herein proposed, we consider time and area inversely proportional towards the field size. A compensation factor of $163/m$ is used to obtain compensated area and time values, with $m$ as the compared field size. The comparison is made using the compensated values. In Table 3 the point multiplication characteristics of the related work are presented and compared with the results obtained for the proposed structure. Virtex-E and Spartan 3 implementations were also realized, in order to properly compare with the related work.

Many of the proposed ECC structures use Lopez-Dahab coordinates [8, 4, 2], and require 10 and 4 multiplications to compute point addition point doubling, respectively. To perform simultaneously point addition and doubling our design requires 6 multiplications.

In [8], the use of Non-Adjacent Forms (NAF) scalar recoding is proposed in order to reduce the required point additions in the traditional double and add algorithms. Since our design has 5 times speedup, we expect that only if this design would avoid 4 additions in every 5 additions, it would achieve the same time performance as the design herein proposed, considering identical field multiplication performance. Furthermore, this design does not implement affine point addition or projective to affine point conversion.

To improve the performance of projective to affine conversion, a mixed coordinate representation is proposed in [2]. Jacobian coordinates applied on the Montgomery ECC multiplication are also used in [2], resulting in a better performance using the Massey-Omura field multiplier supported over an optimal normal base. Nevertheless, perfor-
Table 3. ECC point multiplication state of the art comparison table.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Device</th>
<th>Field</th>
<th>Slices</th>
<th>Max. Freq [MHz]</th>
<th>Total time [µs]</th>
<th>area (\times) (10^2)</th>
<th>speedup (\times) (10^3)</th>
<th>(Slices (\times) time) (^{-1})</th>
<th>(Slices (\times) time) (^{-1}) improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>[4]</td>
<td>XC4085XLA</td>
<td>(GF(2^{46}))</td>
<td>2,634</td>
<td>32</td>
<td>1,610</td>
<td>28%</td>
<td>0.21</td>
<td>0.32</td>
<td>0.89</td>
</tr>
<tr>
<td>[2]</td>
<td>XCV1000</td>
<td>(GF(2^{46}))</td>
<td>29,766</td>
<td>36</td>
<td>270</td>
<td>316%</td>
<td>1.27</td>
<td>0.17</td>
<td>0.47</td>
</tr>
<tr>
<td>[6]</td>
<td>XCV3200E</td>
<td>(GF(2^{46}))</td>
<td>18,314</td>
<td>9.99</td>
<td>56</td>
<td>194%</td>
<td>6.11</td>
<td>1.34</td>
<td>3.69</td>
</tr>
<tr>
<td>Ours</td>
<td>XCV3200E</td>
<td>(GF(2^{46}))</td>
<td>9,432</td>
<td>49</td>
<td>292</td>
<td>100%</td>
<td>1</td>
<td>0.36</td>
<td>1</td>
</tr>
<tr>
<td>[8]</td>
<td>XC3S1000</td>
<td>(GF(2^{163}))</td>
<td>n.a.</td>
<td>80</td>
<td>2280</td>
<td>-</td>
<td>0.20</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td>XC3S2000</td>
<td>(GF(2^{163}))</td>
<td>10,379</td>
<td>44</td>
<td>325</td>
<td>100%</td>
<td>1</td>
<td>0.30</td>
<td>1</td>
</tr>
<tr>
<td>Ours</td>
<td>XC4VSX35</td>
<td>(GF(2^{163}))</td>
<td>10,488</td>
<td>99</td>
<td>144</td>
<td>100%</td>
<td>1</td>
<td>0.66</td>
<td>1</td>
</tr>
</tbody>
</table>

Table for ECC point multiplication state of the art comparison table.

In [4] an Itoh-Tsujii inverter is implemented using multiplications recurrently. In our design, an inversion takes approximately the same time as 8 multiplications. Since, in [4] more than 8 multiplications are needed to perform the inversion, our design is able to achieve a higher performance. Furthermore, with a dedicated divider architecture it is possible to parallelize the division and multiplication procedures, which is not possible in the approach used in [4].

The design proposed in [6], using two field Karatsuba-Offman multipliers, is able to achieve the computation of point multiplication about 6 times faster than the structure herein proposed. Part of this improvement is obtained from a more aggressive parallelization, which results in an area 94% higher. This design does not support the computation of point addition or projective to affine coordinates conversion. Moreover, in [6] 24 embedded memories (BRAM) are used but are not considered in the area metric.

As a concluding remark, it should be noted that none of these designs implements coordinate \(y\) collapsing, as is the case of the proposed structure. This coordinate collapse allows to reduce by half the required bandwidth.

References


Performance comes at the higher expense of circuit area. Moreover, no parallelization of the ECC multiplication algorithm can be exploited. Our design suggests better performance, since parallelization techniques can be used in order to obtain a more area efficient solution. A \((\text{slice} \times \text{gain})^{-1}\) metric of 0.47 is obtained.

In [4] an Itoh-Tsujii inverter is implemented using multiplications recurrently. In our design, an inversion takes approximately the same time as 8 multiplications. Since, in [4] more than 8 multiplications are needed to perform the inversion, our design is able to achieve a higher performance. Furthermore, with a dedicated divider architecture it is possible to parallelize the division and multiplication procedures, which is not possible in the approach used in [4].

The design proposed in [6], using two field Karatsuba-Offman multipliers, is able to achieve the computation of point multiplication about 6 times faster than the structure herein proposed. Part of this improvement is obtained from a more aggressive parallelization, which results in an area 94% higher. This design does not support the computation of point addition or projective to affine coordinates conversion. Moreover, in [6] 24 embedded memories (BRAM) are used but are not considered in the area metric.

As a concluding remark, it should be noted that none of these designs implements coordinate \(y\) collapsing, as is the case of the proposed structure. This coordinate collapse allows to reduce by half the required bandwidth.

4 Conclusions

In this paper a complete elliptic curve processor capable of computing point multiplication and addition is proposed. The proposed structure is also able of collapsing both point coordinates into a single coordinate and performs scalar multiplications without having to calculate both input/output coordinates. This is an advantage in the sense that we have only to transmit half the data for the same information without compromising the performance. The proposed ECC processor design supports operations over field \(GF(2^m)\), and was implemented for a field with \(m = 163\), on a Xilinx Virtex4. It requires 10,488 slices and computes each point multiplication in 144.374µs and each point addition in 1.024µs. Results suggest that significant performance gains can be achieved in comparison with most of the related state of the art. Performance gains in the order of 5 times faster can be achieved. When compared with the related work with better results, there is a proposed structure which performs the point multiplication 6 times faster at the expense of 94% more area. Furthermore, the design herein proposed also computes coordinate collapsing, allowing for a better usage of the available transmission bandwidth.