High-Speed Low-Complexity Elliptic Curve Cryptographic Processor

Tuy Tan Nguyen and Hanho Lee
Dept. of Information and Communication Engineering
Inha University, Incheon, Korea
hhlee@inha.ac.kr

Abstract—**In this paper, we present a novel modified algorithm and architecture to compute Elliptic Curve Cryptographic point multiplication. The proposed algorithm significantly reduces the number of point addition operations. Furthermore, the field multiplication operations are scheduled to perform simultaneously to reduce the latency. As a result, our algorithm and architecture reduce the hardware complexity compared to others, while the required time for point multiplication is kept at a reasonable value. The simulation result on Xilinx Virtex-7 FPGA shows that the proposed architecture offers an improvement in hardware complexity up to 68% and much better efficiency compared to others.**

Keywords-bit-serial; elliptic curve cryptography; architecture; point multiplication

I. INTRODUCTION

Elliptic curve cryptography (ECC) is an interesting research topic that has attracted great attention in recent years. Compared with other public key cryptography, it requires less number of key bits to achieve the same security level [1], [2]. An elliptic curve E over Galois-field (GF) is the set of solution to the equation $y + xy = x^3 + ax^2 + b$. The underlying operation in elliptic curve cryptosystems is scalar point multiplication, $Q = kP$, the multiplication of an elliptic curve point $P(x_P, y_P)$ by a scalar k to give the resultant point Q.

II. BACKGROUND

A. Galois-Field Arithmetic

A set of elements denoted as $GF(2^m)$ is a Galois-field. The operations over $GF(2^m)$ includes addition and multiplication. Subtraction of field elements can be defined in terms of addition: for $a, b \in GF(2^m)$, $a - b = a + (-b)$, where $-b$ is the negative of b. Similarly, division of field elements is defined in terms of multiplication: for $a, b \in GF(2^m)$ with $b \neq 0$, $a/b = a \cdot b^{-1}$ where b^{-1} is the unique inverse element in $GF(2^m)$. An element $a(x)$ in $GF(2^m)$ can be expressed as $a(x) = a_0 + a_1x + \dots + a_{m-1}x^{m-1}$

B. ECC Point Multiplication Algorithm

In this section, we introduce the Lopez-Dahab (LD) algorithm, which is used in many previous works [1], [3]. A point $P(x_P, y_P)$ on elliptic curve E. The LD scalar point multiplication calculate the point $Q(x_Q, y_Q)$ so that $Q = kP$,

Algorithm 1. *LD scalar point multiplication [1]*

```
Input: $k = (k_{m-1}...k_1k_0)$, $P(x_P, y_P)$. Output: $Q = kP$
 1. $A = 0; B = P;$
 2. for $i = 1$ to $i = m$ do
      if( $k_{m-1} = 0$)
        $B = A + B;\ A = 2A;$
      else
        $A = A + B;\ B = 2B;$
      end if
    end for
 3. Return $Q = A;$
```

where k is the security key. To perform this multiplication, the point addition and point doubling are efficiently combined.

As can be seen in Algorithm 1, the point A and B are initially assigned to 0 and P, respectively. When the key bit is "1", the result of point addition is assigned to A, followed by the assignment of doubling operation results to B. After finishing all operation, the result is returned to Q. The advantage of this algorithm is that the point addition and point multiplication operations are performed alternately. However, we can reduce the number of point addition operations to calculate one point multiplication. As a result, the hardware requirement and the latency are reduced.

III. PROPOSED ALGORITHM AND ARCHITECTURE FOR ECC POINT MULTIPLICATION

In this section, we present a novel modified algorithm and corresponding architecture to calculate the ECC point multiplication in projective coordinate. Initially, the assignments $X_2 = x_P$ and $Z_2 = 1$ are performed. The initial values of X_1 and Z_1 depend on the value of the least significant bit of the key k_0. If $k_0 = 1$, X_1 and Z_1 are assigned to x_P and 1, respectively. Otherwise, both X_1 and Z_1 are assigned to 0. As we can see in the Algorithm 2 and Fig. 1, the point doubling is executed each iteration. To perform the point multiplication, the field squaring (X_2^4, Z_2^4, T^2), field multiplication ($T^2Z^2_2$) and field addition ($X_2^4 + bZ_2^4$) are required. When the key bit $k_i = 1$, the point addition block is enabled. The result from the point double block is added with the current value of point $P_1(X_1, Z_1)$. The field operation includes point multiplication (X_1Z_2, X_2Z_1, x_PZ_1, $X_1Z_2TZ_2$), field addition ($X_1Z_2+X_2$, Z_1 $x_PZ_1+X_1X_2TZ_2$) and field squaring $(X_1Z_2+X_2Z_1)^2$.

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC support program (IITP-2015-H8501-15-1019) supervised by the IITP, and IDEC (EDA Tool, MPW).

Algorithm 2. *Modified LD algorithm*

```
Input: $k = (k_{m-1}...k_1k_0)$, $P(x_P, y_P)$. Output: $Q = kP$
 1. From affine coordinate to projective coordinate
   $X_2 = x_P, Z_2 = 1;$
   if $(k_0 = 1)$ $X_1 = x_P, Z_1 = 1$ else $X_1 = 0, Z_1 = 0;$ end if
 2. Calculate in projective coordinate
   for $i = 1$ to $i = m - 1$ do
     $T = X_2, X_2 = X_2^4 + bZ_2^4, Z_2 = T^2Z_2^2$
     if$(k_i = 1)$
         $T = Z_1, Z_1 = (X_1Z_2 + X_2Z_1)^2, X_1 = x_PZ_1 + X_1X_2TZ_2;$ end if
   end for
 3. Convert from projective to affine coordinate
   $x_Q = X_1/Z_1$
   $y_Q = (x_P + x_Q)[(X_1 + x_PZ_1)(X_2 + x_PZ_2)+(x_P^2 + y_P)Z_1Z_2)](x_PZ_1Z_2)^{-1} + y_P$
 4. Return$(x_Q, y_Q)$
```

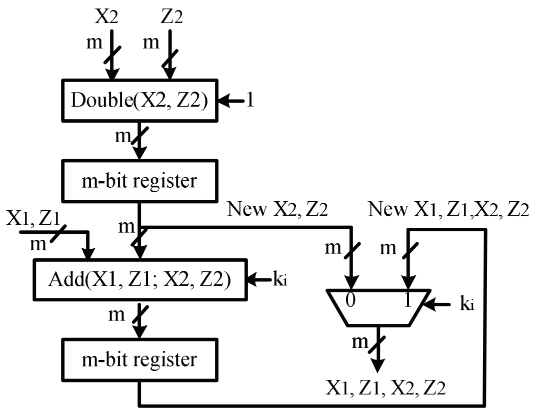

Figure 1. ECC point multiplication architecture.

The result from point addition is fed to the MUX, where the current key bit plays a role of enable signal. The output value of this MUX is updated for the next iteration. This process repeats until $i = m - 1$.

IV. Implementation and Performance Comparison

The proposed algorithm is implemented using Visual C++ to check the function and calculate the number of field operations. Fig. 2 shows the comparison in the number of field calculations between our algorithm and the LD algorithm in [1]. As can be seen, our algorithm significantly reduces the number of calculations, including the squaring, multiplication and addition operations compared to the traditional LD algorithm. From Fig. 2, in case of 50% of bit "0" in the key, the proposed algorithm can reduce 35% number of field multiplication operations compared to this in the traditional LD algorithm.

Table 1 show the performance comparison between our proposed algorithm and the existing ones. To perform the point multiplication, the proposed architecture requires only 3,303 LUTs, which is 35% and 68% less than the architecture in [1] and [4], respectively. Moreover, the efficiency of our work is 1.8 times and 17 times compared with that of [1] and [4], respectively. In addition, the throughput of our design is higher than that of the previous bit-serial structure in [1] and [4].

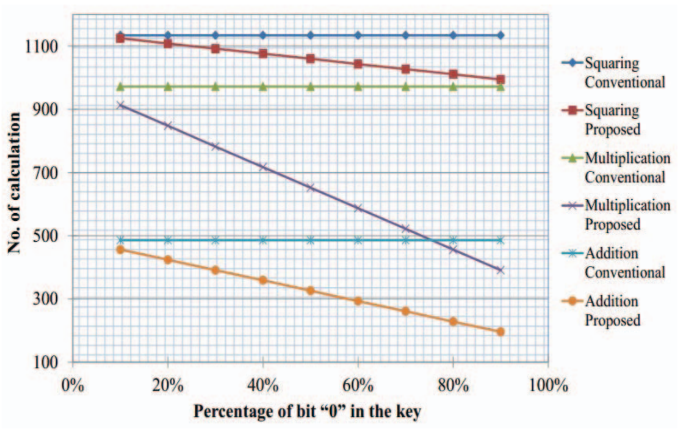

Figure 2. Comparison in the number of field operations.

TABLE I. Performance comparison for ECC processors.

	Proposed	*[1]*	*[4]*
FPGA Device	*Virtex-7*	*Virtex-5*	*Virtex-5*
Mult. operation	Bit-serial	Bit-serial	Bit-serial
LUTs	3,303	5,100	10,584
Slices	5,238	-	-
Clock rate (MHz)	800	550	271
# of clock cycles	68,255	54,943	124,660
Time (µs)	85.3	98.1	460
Throughput (Mbps)	1.91	1.63	0.35
Efficiency[a]	578	313	33

a. Efficiency = Throughput/Area [2]

V. Conclusion

This paper presents a novel algorithm and architecture to calculate the ECC point multiplication. Synthesis result shows that the proposed architecture over-performs others in term of number of field calculation, hardware cost and efficiency. This work is suitable for applications in which resources are strictly considered.

References

[1] G. D. Sutter, J.-P. Deschamps, and J. L. Imaña, "Efficient Elliptic Curve Point Multiplication Using Digit-Serial Binary Field Operations," *IEEE Trans. on Industrial Electronics*, vol. 60, no.1, pp. 217-225, Jan. 2013.

[2] N. Koblitz, A. Menezes, and S. Vanstone, "The state of elliptic curve cryptography," *Des. Codes Cryptography*, vol. 19, no. 2–3, pp. 173–193, Mar. 2000.

[3] H. Mahdizadeh and M. Masoumi, "Novel Architecture for Efficient FPGA Implementation of Elliptic Curve Cryptographic Processor Over GF(2^{163})," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 12, pp. 2330-2333, Dec. 2013.

[4] A. P. Fournaris. J. Zafeirakis and O. Koufopavlou, "Designing and Evaluating High Speed Elliptic Curve Point Multipliers" *17th Euromicro Conference on Digital System Design*, pp. 169-174, Aug. 2014.