High-performance Pipelined FPGA Implementation of the Elliptic Curve Cryptography over GF (2^n)

Salah Harb[a], M. Omair Ahmad[b] and M. N. S. Swamy[c]
Electrical and Computer Engineering Department, Concordia University, 1440 De maisonnueve, Montreal, Canada

Keywords: Cryptography, Elliptic Curve Cryptography, FPGA, Pipelining Architecture, Finite Field Operations, Field Multiplications, Projective Coordination, Efficiency.

Abstract: In this paper, a high-performance area-efficient hardware design for the Elliptic Curve Cryptography (ECC) is presented, targeting the area-constrained high-bandwidth embedded applications. The high-speed design is implemented using pipelining architecture. The applied architecture is performed using n-bit data path of the finite field GF(2^n). For the finite field operations, the implementation in the ECC uses the bit-parallel recursive Karatsuba-Ofman algorithm for multiplication and Itoh-Tsuji for inversion. A modified efficient montgomery ladder algorithm is utilized for the scalar multiplication of a point. The pipelined registers are inserted in ideal locations, where balanced-execution paths among computing components are guaranteed. A Memory-less finite state machine model is developed to control the instructions of computing the finite field operations efficiently. The high-performance design has been implemented using Xilinx Virtex, Kintex and Artix FPGA devices. It can perform a single scalar multiplication in 226 clock cycles within 0.63μs using 2780 slices and 360Mhz working frequency on Virtex-7 over GF (2^{163}). In GF (2^{233}) and GF (2^{571}), a scalar multiplication can be computed in 327 and 674 clock cycles within 1.05μs and 2.32μs, respectively. Comparing with previous works, our design requires less number of clock cycles, and operates using less FPGA resources with competitive high working frequencies. Therefore, the proposed design is well suited in the resources-constrained real time cryptosystems like those in online banking services, wearable smart devices and network attached storages.

1 INTRODUCTION

Elliptic curve cryptosystem (ECC) is a public-key cryptography, which was first proposed by Neal Koblitz and Victor Miller in the 1980s (Kocher et al., 1999) , (Miller, 1985). Since then, many studies have been conducted to explore its security levels against other public-key cryptosystems such as El-Gamal, RSA and Digital Signature Algorithm (DSA) (ElGamal, 1985), (Rivest et al., 1978), which are based on either the integer factorization or discrete logarithm problems (McGrew et al., 2011). Equivalent security levels with smaller sizes of keys, ease to implement, and resource savings, are reasons that give the ECC to be very appealing and more dominant between the hardware reconfigurable implementations. Moreover, ECC is well suited to be implemented in such resource-constrained embedded systems, since it provides same security levels as in RSA using small keys. ECC has been standardized by IEEE and the National Institute of Standard and Technology (NIST) as a scheme in digital signature and key agreement protocols (for Standardization (ISO), 2000).

Generally, most of cryptographic algorithms are implemented in software platforms. Performing an algorithm on a general purpose processor (e.g. CPU) will require most of its resources to compute results of intensive operations because of the large operands used in these very accurate computations. Moreover, CPU is not suitable in performing such these algorithms that having the parallel architecture in nature. These issues prove that software implementation of encryption algorithms does not provide the required performance. Due to the diversity in the applications, the trade-off between area, speed and power is required. Some applications, such as RFID cards, nodes of wireless sensor networks and cell phones, need a small area and power. Other applications, such as web servers, large bandwidth networks

[a] https://orcid.org/0000-0002-5975-6537
[b] https://orcid.org/0000-0002-2924-6659
[c] https://orcid.org/0000-0002-3989-5476

Harb, S., Ahmad, M. and Swamy, M.
High-performance Pipelined FPGA Implementation of the Elliptic Curve Cryptography over GF (2n).
DOI: 10.5220/0007772800150024
In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019)*, pages 15-24
ISBN: 978-989-758-378-0

and satellite broadcast require very high throughputs. To cover the issues of software implementation and meet trade-offs in numerous applications, the hardware platforms have been utilized for implementing the cryptographic algorithms, where high efficiency to perform tasks is achieved in different applications.

Field Programmable Gate Array (FPGA) is one of the preferable reconfigurable hardware platforms (Xilinx, 2018a) which offers flexible and more customizable methods for performing and evaluating different hardware implementations. Because of this fact and since FPGAs have been employed by most of the previous hardware implementations to evaluate their performances; the presented ECC hardware implementation in this paper have been performed using Xilinx FPGA devices (Xilinx, 2018b). Scalar point multiplication (SPM) is the main point operation in ECC cryptosystems or protocols such as Elliptic Curve Diffie-Hellman (ECDH) (Diffie and Hellman, 1976) for key agreements and Elliptic Curve Digital Signature (ECDS) for digital signatures.

SPM can be implemented over many finite fields under either prime or polynomial fields. Finite fields named also Galois Fields (GF), where GF (p) is a prime field and GF (2^n) is the polynomial field. SPM has two point operations, doubling and adding points. Each operation consists finite field operations such as square, addition, multiplier. Figure 1 presents the hierarchical implementation of the ECC protocol. Polynomial fields are more suited and efficient to implement on a customizable platform such FPGAs (Wenger and Hutter, 2011a).

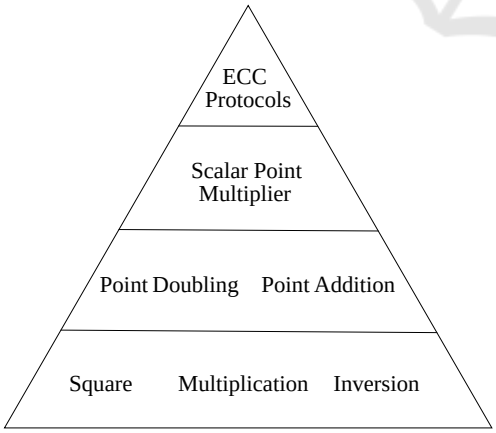

Figure 1: ECC Cryptosystem Hierarchy.

To gain high performance in today's high loaded communication networks, utilization of hardware accelerators for physical security has created a great demand for efficient and high-speed implementations of ECC. Based on this fact, many FPGA implementations of the ECC have been published in the literature, where various ranges of latencies and number of clock cycles are achieved targeting applications that require high/low throughputs. Providing high performance as well as utilizing efficient area, is a challenge to achieve it in FPGA's ECC implementations.

In this paper, a high-speed area-efficient Xilinx FPGA implementation of the ECC over GF (2^n) using the pipelining architecture is proposed. The main target of our work is to develop high performance design that targets systems that have constrained resources such as wearable smart devices, processing engines in image steganographic systems (Dalal and Juneja, 2018), (Amirtharajan, 2014) and Internet of Things (IoTs) network processors. This paper is organized as follow: section 2 describes the arithmetic operations of the ECC; section 3 describes our high-performance hardware implementation core for ECC over GF (2^n); section 4 shows the results and comparisons; and finally, section 5 concludes this paper.

2 ELLIPTIC CURVE CRYPTOGRAPHY (ECC)

Elliptic Curves (ECs) are formulated by the so called Weiestrass equations, which can be performed over by normal or polynomial basis. In this paper, we work on the polynomial basis in GF (2^n) for its efficiency on the hardware platforms (Wenger and Hutter, 2011a). Equation 1 represents the general form of the nonesingular curve over GF (2^n) (Hankerson et al., 2006).

$$y^2 + xy = x^3 + ax^2 + b \tag{1}$$

where a, b $\in$ GF (2^n) and b $\neq$ 0. A set of affine points (x, y) satisfying the curve forms a group (Hankerson et al., 2006) with an identity point of that group. There are two fundamental elliptic curve operations, doubling and adding points. Doubling point is denoted as P_1=2P_0, where P_1 is (x_1, y_1) and P_0 is (x_0, y_0) while point addition is denoted as P_2 = P_0+P_1, where P_2 is (x_2, y_2), and $P_1 \neq P_0$. All points in the selected curve are represented in affine coordinates. Finite fields operations are involved in the ECC point operations such as addition, square, multiplication and inversion. Dealing with affine coordinates requires an inversion field operation. Due to the complexity in the inversion operation, a projective coordinate is utilized to avoid it by mapping points in affine (x, y) to be represented in (X, Y, Z) form. Scalar Point Multiplication (SPM) is the main important operation that dominants the ECC-based cryptosystems. SPM is process of adding a point k times, where k is a positive integer and P is a point on a

selected curve. SPM is based on the implementation of the underlying point operations, where a series of doubling and adding points is performed by scanning the binary sequence of the k scalar, where $k = k_n k_{n-1} \dots k_1 k_0$. There are several methods and techniques to implement k.P efficiently. Binary Add-and-Double Left-Right, Binary Add-and-Double Right-Left as the work presented in (Harb and Jarrah, 2019), Non-Adjacent-Form (NAF) (Moon, 2006) and montgomery ladder method (Montgomery, 1987). Affine coordinate system is the default representation that some SPM methods use it to introduce the points on the curves, while other SPMs utilize alternative projective coordinate systems for representing the points. The reason of applying different projective systems is to avoid the time-resource consuming inversion field operation. However, it increases the number of field multiplications. Generally, projective systems tend to provide efficient ECC cryptosystem's designs in terms of area/latency.

Lopez-Dahab (LD) (López and Dahab, 1998) is one of the most efficient projective coordinate systems. Points in affine system are mapped to LD projective system, where P at (x, y) coordinate is equal to P at (X = x, Y = y, Z = 1) coordinate. In any SPM, the point multiplication algorithm must be selected first for performing the k.P. The next step is to define the projective coordinate system for representing the points. The last step, the algorithm that is used in the finite field operations, mainly, for the field multiplication and inversion operations. Figure 2 illustrates the main steps of constructing SPM at our ECC cryptosystem. In the presented work, montgomery ladder algorithm is selected as point multiplication algorithm and LD coordinate system is defined to represents the points. For the field multiplication and inversion operations, the Karatsuba-Ofman algorithm and Itoh-Tsuji algorithm are implemented, respectively.

2.1 Finite Fields GF (2^n)

ECC over GF (2^n) is more suitable and efficient in hardware implementations because the arithmetic in the polynomial fields is carry-less. In GF (p), finding points on an elliptic curve requires performing a square root algorithm (Harb and Jarrah, 2017) (Wenger and Hutter, 2011b), while the process is much easier over GF (2^n), where the points can be found using generators for polynomials. There are 2n-1 elements in GF (2^n), which can be represented as binary polynomials. For example, the element in GF (2^n) has a polynomial representation:

$$a_{n-1}x^{n-1} + a_{n-1}x^{n-2} + \dots + ax + 1 \qquad (2)$$

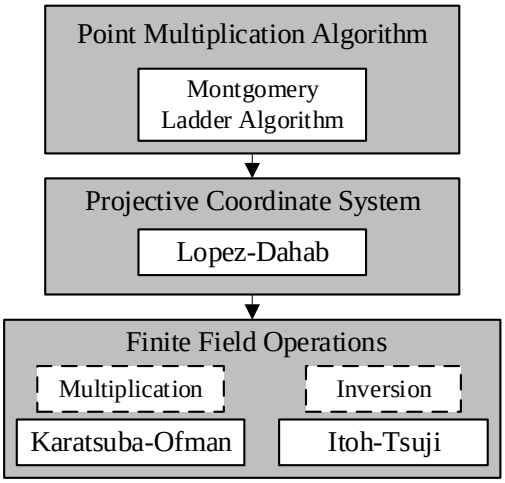

Figure 2: Main Components of SPM.

where x_i is the location of the ith term, and $a_i \in$ [0,1] is the coefficient of the ith term. Arithmetic operations are applied over these elements such as addition, multiplication, squaring and division (i.e. inversion and multiplication). Adding two elements C = A + B is done using the logic XOR. Squaring an element A^2 is computed by padding 0s between two adjacent bits of the element. Multiplying two elements C = A · B is much harder and slower operation than addition and square.

Result of squaring an element or multiplying two elements would be out of the field. To reduce it, an irreducible polynomial of degree n ($f(x)$) is used by applying the reduction (modular) step, such as C = (A^2) mod $f(x)$ or C = (A · B) mod $f(x)$. Inversion operation is the most complex operation in terms of time and resources. Obtaining the inverse of an element A is the process of finding another element B that satisfies A · B $\equiv$ 1 mod $f(x)$. Note that, the $f(x)$ has a major role in the performance of these operations. In this paper, all curves and irreducible polynomials are chosen based on the recommendation of the NIST (Gallagher, 2013).

3 HIGH-SPEED CORE FOR ECC OVER GF (2^n)

In this section, we introduce our proposed architecture of the high-speed ECC core over GF (2^n). Karatsuba-Ofman and Itoh-Tsuji algorithms are implemented for performing both field multiplication and inversion operations, respectively. The LD coordinate system is used as projective coordinate system to present the points. For the SPM, a modified-pipelined montgomery ladder method is implemented

in such a way that an efficient number of pipelined stages are inserted. Next subsections present more details about these algorithms.

3.1 Montgomery Ladder Scalar Point Multilocation Algorithm

At present, montgomery ladder is one of the most popular multiplication algorithms to perform k.P, where k is an integer. It can be implemented in both affine and projective coordinate systems. Doubling and adding point operations are computed in an efficient way for every bit in the sequence of k = k = $k_{t-1}k_{t-2}\ldots k_1k_0$. Algorithm 1 shows the projective coordinate version of montgomery ladder method (López and Dahab, 1998). As it's shown, the algorithm is based on performing point operations recursively using the x affine coordinate, which leads to reduce the number of field multiplications. The y coordinate is used at the post-process affine, which is required to recover the affine coordinates from the LD coordinates. This algorithm has been implemented in many hardware implementations that provides high performance (Ansari and Hasan, 2008) (Roy et al., 2013) (Mahdizadeh and Masoumi, 2013) due to its speed, parallelism capability, resource-constrained systems and power analysis resistance.

3.2 Area-constrained and High-performance Tradeoff

High-performance ECC hardware implementations are achieved by considering optimization techniques such as short-critical path, required resources and number of clock cycles to perform a single SPM. The architectural pipelining optimization aims to optimize the long-critical path of the design through break it into stages. Number of clock cycles can be improved by adopting the parallel field multiplication operations. All these optimization techniques impact on the resources that are required for performing the SPM. In pipelining, inserting registers to minimize the critical path delay results in an increase on the number of the clock cycles (latencies) and resources. Obtaining an efficient pipelined design is done by determining the number of the stages. More stages yield higher working frequencies but higher latencies. Balancing this tradeoff can be achieved by considering an efficient field multiplier, independency levels among point operations, and finite-state machines that control these operations effectively.

Algorithm 1: Montgomery Ladder Scalar Multiplier (k.P).

Input: $k = k_{t-1}k_{t-2}\ldots k_1k_0$, Point P($x$,$y$) on Elliptic Curve.
Output: A point Q = kP
$X_1 \leftarrow x$; $Z_1 \leftarrow 1$; $X_2 \leftarrow x^4 + b$; $Z_2 \leftarrow x^2$;
if $k == 0 \parallel x == 0$ **then**
Q $\leftarrow$ (0, 0);
end if
Stop;
for $i = t-2$ to 0 **do**
if $k_i == 1$ **then**
Q (X_1,Z_1) $\leftarrow$ add(X_1, Z_1, X_2, Z_2); $\Rightarrow$ {
$Z_1 \leftarrow X_2 \cdot Z_1$; $X_1 \leftarrow X_1 \cdot Z_2$; $T \leftarrow X_1 + Z_1$;
$X_1 \leftarrow X_1 \cdot Z_1$; $Z_1 \leftarrow T^2$; $T \leftarrow x \cdot Z_1$;
$X_1 \leftarrow X_1 + T$; }
Q (X_2,Z_2) $\leftarrow$ double(X_2, Z_2); $\Rightarrow$ {
$Z_2 \leftarrow Z_2^2$; $T \leftarrow Z_2^2$; $T \leftarrow b \cdot T$;
$X_2 \leftarrow X_2^2$; $Z_2 \leftarrow X_2 \cdot Z_2$; $X_2 \leftarrow X_2 + T$; }
else
Q (X_2,Z_2) $\leftarrow$ add(X_2, Z_2, X_1, Z_1); $\Rightarrow$ {
$Z_2 \leftarrow X_1 \cdot Z_2$; $X_2 \leftarrow X_2 \cdot Z_1$; $T \leftarrow X_2 + Z_2$;
$X_2 \leftarrow X_2 \cdot Z_2$; $Z_2 \leftarrow T^2$; $T \leftarrow x \cdot Z_2$;
$X_2 \leftarrow X_2 + T$; }
Q (X_1,Z_1) $\leftarrow$ double(X_1, Z_1); $\Rightarrow$ {
$Z_2 \leftarrow Z_1^2$; $T \leftarrow Z_1^2$; $T \leftarrow b \cdot T$;
$X_1 \leftarrow X_1^2$; $Z_1 \leftarrow X_1 \cdot Z_1$; $X_1 \leftarrow X_1 + T$; }
end if
end for
Return Q (X_3,Z_3) = affine(X_1, Z_1, X_2, Z_2); $\Rightarrow$ {
$x_3 \leftarrow X_1/Z_1$;
$y_3 \leftarrow (x + X_1/Z_1)[(X_1 + x \cdot Z_1)\ (X_2 + x \cdot Z_2) + (x^2 + y)(Z_1 \cdot Z_2)]$
$(x \cdot Z_1 \cdot Z_2^{-1}) + y$; }

3.3 Proposed High-speed Area-efficient ECC Core over GF (2^n)

Algorithm 1 states that for performing single point multiplication, three stages are covered; 1) map the affine coordinates to LD coordinates, 2) perform doubling and adding point operations recursively, and 3) recover the point from LD to affine coordinates. Each iteration performs the same point operations with swapping between input and output registers (X and Z registers) depending on the current bit of the k_i if it is 1 or 0. From this recursive manner, the authors in (Mahdizadeh and Masoumi, 2013) noticed that the initialization step of Algorithm 1 can be merged into the main loop of the algorithm. the registers are before main loop: X_1 = 1, Z_1 = 0, X_2 = x, Z_2 =1. This eliminates the need of precomputed values to be obtained before starting the main loop.

However, it requires extra clock cycles for the merged initialization step. The design in (Ansari and Hasan, 2008) addressed the swapping feature

in Algorithm 1 and proposed a merged-improved montgomery ladder point multiplication algorithm as shown in Algorithm 2.

Algorithm 2 : Merged-Improved Montgomery Ladder Scalar Multiplier (k.P).

Input: $k = k_{t-1}k_{t-2}...k_1k_0$, Point P($x$,$y$) on Elliptic Curve.
Output: A point Q = kP
$X_1 \leftarrow 1; Z_1 \leftarrow 0; X_2 \leftarrow x; Z_2 \leftarrow 1;$
if $k == 0 \parallel x == 0$ **then**
Q $\leftarrow (0, 0)$;
end if
Stop;
for $i = t-1$ to 0 **do**
$T \leftarrow Z_1; Z_1 \leftarrow (X_1 \cdot Z_2 + X_2 \cdot Z_1)^2;$
$X_1 \leftarrow x \cdot Z_1 + X_1 \cdot X_2 \cdot T \cdot Z_2;$
$T \leftarrow X_1$;
$X_2 \leftarrow X_2^4 + b \cdot Z_2^4;$
$Z_2 \leftarrow T^2 \cdot Z_2^2;$
if $(i \neq 1 \& k_i \neq k_{i-1}) \parallel (i == 0 \& k_i == 0)$ **then**
swap(X_1, X_2, Z_1, Z_2); $\Rightarrow \{$
$T_1 \leftarrow X_1; X_1 \leftarrow X_2$;
$X_2 \leftarrow T_1$;
$T_2 \leftarrow Z_1; Z_1 \leftarrow Z_2$;
$Z_2 \leftarrow T_2; \}$
end if
end for
Return Q (X_3,Z_3) = affine(X_1, Z_1, X_2, Z_2); $\Rightarrow \{$
$x_3 \leftarrow X_1/Z_1;$
$y_3 \leftarrow (x + X_1/Z_1)[(X_1 + x \cdot Z_1)\ (X_2 + x \cdot Z_2) + (x^2 + y)(Z_1 \cdot Z_2)]\ (x \cdot Z_1 \cdot Z_2^{-1}) + y; \}$

The proposed high-speed area-efficient hardware design is shown in Figure 3. The general architecture has three units, optimized finite-state machine control unit, GF (2^n) arithmetic unit contains Karatsuba-Ofman and square, and control signal unit. Next, further details are given for these main unites in the high-speed ECC core.

3.3.1 Finite-state Machine for the Proposed ECC Core

Finite-State Machine (FSM) is a control model which is developed to control any hardware design that has many components such registers, addresses and flags. Controlling design includes maintaining the proper transitions between these components.
Each transition in FSM affects the state (i.e. current values) of the registers and flags. FSM of the proposed high-performance core is implemented in an efficient way, where minimum number of states have been utilized. Figure 4 represents the main FSM

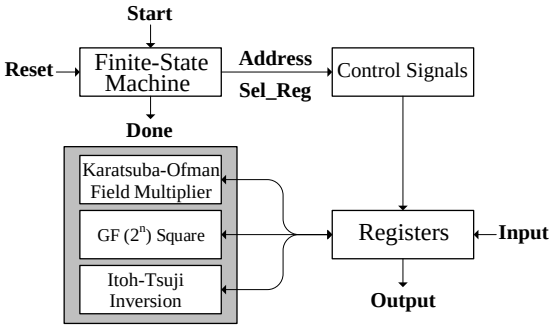

Figure 3: The Proposed High-speed Area-Efficient Hardware Design.

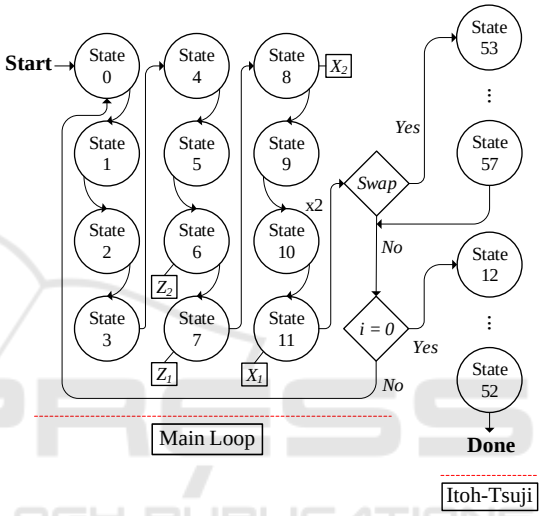

Figure 4: State Machine of the Proposed SPM.

of the proposed high-speed core. As it's shown, the main loop of the merged-improved montgomery ladder SPM starts from state 0 and ends at state 11. At state 11, the condition (the second if-statement in Algorithm 2) of whether swapping registers must be done or go back to state 0 for the next ki. The swap process is done in a routine which starts from state 53 to state 57. Once the i is equal to zero, the results are ready to be mapped back to the affine coordinates. Itoh-Tsuji inversion algorithm is used to achieve that, and it starts from state 12 to state 52. At state 52, a done signal is asserted to indicate that mapping process is done, and affine coordinates are obtained.

3.3.2 Computation Schedule for Montgomery Ladder SPM

In this subsection, we introduce a new efficient scheduling for performing a single scalar point based on the merged-improved version as shown in Algorithm 2. Free-idle cycles schedule is achieved by performing the doubling and adding points in less-

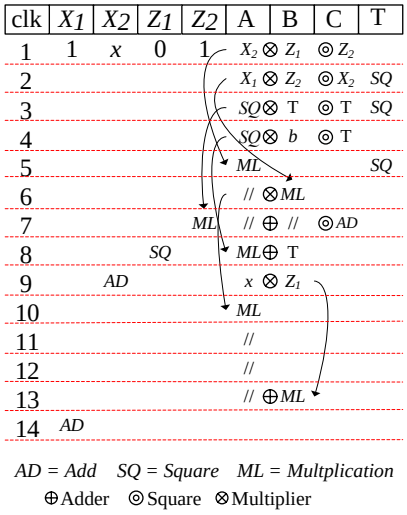

Figure 5: State Machine of the Proposed SPM.

dependent way. This is done by parallelizing the field operations of the point operations, such as squaring and multiplying same or different operands, simultaneously. Figure 5 illustrates our new schedule, where 8 registers are utilized to perform a single loop in Algorithm 2. A and B registers are operands that are connected to the field multiplier, while register C is operand of the square field operation. Register T is a temporary register which is used to hold intermediate values. This zero-idle schedule can perform a single SPM iteration in 14 clock cycles using the three pipelined stages Karatsuba-Ofman multiplier, where 3 clock cycles are required to obtain field multiplication results. One clock cycle for the square one clock cycle for square. Four subsequent field multiplications and squares are performed independently and simultaneously from clock number 1 to 4. Each square result is stored at register T.

At 5, results of first multiplication ($X_1 \cdot Z_2$) is obtained to multiply with the next multiplication result ($X_2 \cdot Z_1$) at clock 6 and added at clock 7. Result of the third multiplication ($T^2 \cdot Z_2^2$) is stored in Z_2 at clock 7. The fourth multiplication ($b \cdot Z_2^4$) is obtained at clock 8 and added with register T which contains X_2^4. Result of ($X_1 \cdot X_2 \cdot T \cdot Z_2$) is obtained at clock 10 which is added to the result of ($x \cdot Z_1$) at clock 13. The total number of clock cycles for performing a single SPM consists of: initialization step, SPM process and LD to affine routine. In our proposed ECC core, there are 9 clock cycles for initialization step, ($\lfloor \log_2(k)+1 \rfloor$) x 14MUL for SPM process, and (9 ‖ 10 ‖ 13) x 3MUL + (162 ‖ 232 ‖ 570) for LD to affine routine.

For example, if the scalar k is 10 and GF (2^{163}), then the total number of clock cycles for performing a single SPM is equal to: 9 + ($\lfloor \log_2(10) + 1 \rfloor$) x 14MUL + 9x3 + 162 = 254 clock cycles. Note that, Itoh-Tsuji algorithm requires n-1 squares and $\lfloor \log_2(n-1) \rfloor + HW(n-1) - 1$ multiplications, where the HW is the hamming weight of the integer (n-1). For example: in GF (2^{233}), 232 squares and $\lfloor \log_2(232)+1 \rfloor + HW(232) - 1 = 7+4-1 = 10$ multiplications.

3.3.3 Karatsuba-Ofman Field Multiplier

Field multiplication consists of two steps, first, compute as C' = A · B, the second is reducing the C' by using the mod operation as C = (C') mod *f(x)*. This kind of multipliers is called as two-step classic field multiplier. The interleaved field multiplier is one of the classical field multipliers (Großschädl, 2001) that apply the two steps as shift and add operations in iterations. Few resources are utilized to implement the interleaved multipliers, which makes it a very attractive one to the constrained-resources systems. However, this type of multipliers has a very long critical path due to the dependency between the iterations. The Karatsuba-Ofman field multiplication is a recursive algorithm that performs polynomial GF (2^n) multiplications in large finite fields efficiently (Karatsuba and Ofman, 1962). Karatsuba-Ofman is given and defined as follows: Let A and B be two arbitrary elements in GF (2^n). Result of C' = A · B is a product of a 2n-2 degree polynomial. Both A and B can be represented as two split parts:

$$A = x^{n/2}(x^{n/2-1} \cdot a_{n-1} + \cdots + a_{n/2}) + (x^{n/2-1} \cdot a_{n/2-1} + \cdots + a_0) = x^{n/2} \cdot A_H + A_L$$

$$B = x^{n/2}(x^{n/2-1} \cdot b_{n-1} + \cdots + b_{n/2}) + (x^{n/2-1} \cdot b_{n/2-1} + \cdots + b_0) = x^{n/2} \cdot B_H + B_L$$

The polynomial of the product C' is:

$$C' = x^n \cdot A_H \cdot B_H + x^{n/2}(A_H \cdot B_L + A_L \cdot B_H) + A_L \cdot B_L$$

The sub-products are defined as auxiliary polynomials as follows:

$$C'_0 = A_L \cdot B_L$$
$$C'_1 = (A_L + A_H)(B_L + B_H)$$
$$C'_2 = A_H \cdot B_H$$

Then the product C' can be obtained by:

$$C' = x^n \cdot C'_2 + x^{n/2}(C'_0 + C'_1 + C'_2) + C'_0$$

This field multiplication can be recursive if we split the auxiliary polynomials again with new auxiliaries are generated. More recursions yield in an increased

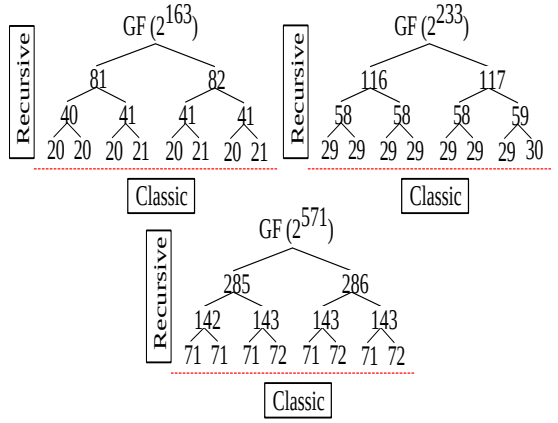

Figure 6: Recursive Splits over Different Finite Fields.

delay for the Karatsuba-Ofman multiplier (Peter and LangendOorfer, 2007). So, this recursion ends after the threshold q splits, where it ends with a classical field multiplier. Number of splits is optimum when splitting reaches the balance between area utilized and delay. The work in (Zhou et al., 2010) have discussed this trade off in details. The best split for the GF (2^{163}), GF (2^{233}) and GF (2^{571}) fields, is shown in Figure 6. The optimum split is coming from the used FPGA technology in term of the lookup tables LUTs (Zhou et al., 2010).

There are two technologies have been released: 4-input LUT (i.e. old FPGA devices) and 6-input LUT (i.e. new FPGA devices) (Percey, 2007), (Specification, 2006). For GF (2^{163}), at first recursive, we get three auxiliaries: 2 of 82-bit multipliers and 1 of 81-bit multiplier. Second recursive, 2 of 41-bit and 1 40-bit multipliers for 81-bit multiplier, and 3 of 41-bit multipliers for 82-bit multiplier. Third recursive, 2 of 21-bit and 1 20-bit multipliers for 41-bit multiplier, and 3 of 20-bit multipliers for 40-bit multiplier. The multiplier used after the recursive split is a single-step (no mod operation) classic multiplier which is used for all three fields. Figure 7 shows the logic gate implementation for the classic multiplier. The critical path of Karatsuba-Ofman multiplier is long due to the recursive nature in its hierarchy. Applying an architectural improvement such as inserting pipelining registers between recursive splits improves the long critical path and provides a higher working frequency.

Efficient pipelined Karatsuba-Ofman multiplier can be achieved when the critical path is the shortest. In FPGA, the shortest critical path implies that the delay-to-area ratio is the minimum in time and utilized area. For achieving that, different pipelined stages have been inserted in the Karatsuba-Ofman multiplier. As shown in Figure 8, the efficient balance in delay-to-area ratio over GF (2^{163}) is achieved by inserting exactly three pipelined stages, where 2121

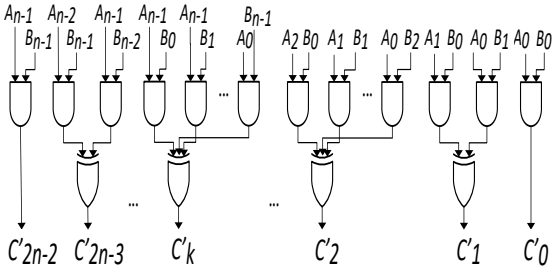

Figure 7: Logic Gate Implementation for the Classic Multiplier.

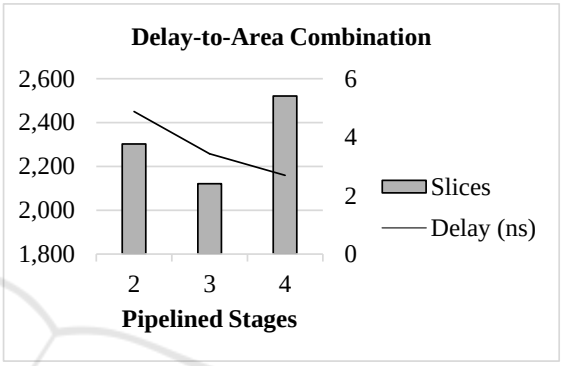

Figure 8: Delay-to-Area Ratio using Pipelined Stages.

slices are used and the maximum delay between two registers is 3.4 ns. The first pipelined stage is inserted after the classic multiplier, while the second stage is located after combining all 40-bit, 41-bit, 81-bit, 82-bit of recursive splits of the Karatsuba-Ofman multiplier. Note that, the FPGA technology that have been used in Figure 8 is a 6-input LUT.

4 FPGA IMPLEMENTATION: RESULTS AND COMPARISONS

The pipelining architectural approach is applied to the proposed ECC core for higher speed with efficient utilized area in terms of both working frequencies and slices. Elliptic curve doubling and adding point operations are performed using a merged-improved montgomery ladder scalar point multiplication algorithm. The proposed ECC core doesn't require any precomputed values or any memories for calculations, which provides an efficient design with less slices and clock cycles. An effective FSM is developed to control the main ECC components, where a minimum number of states are used for performing the merged-improved montgomery ladder SPM. To verify performance of the proposed SPM, our high-speed ECC core is implemented over three finite fields, GF (2^{163}), GF (2^{233}) and GF (2^{571}) using many FPGA devices which are

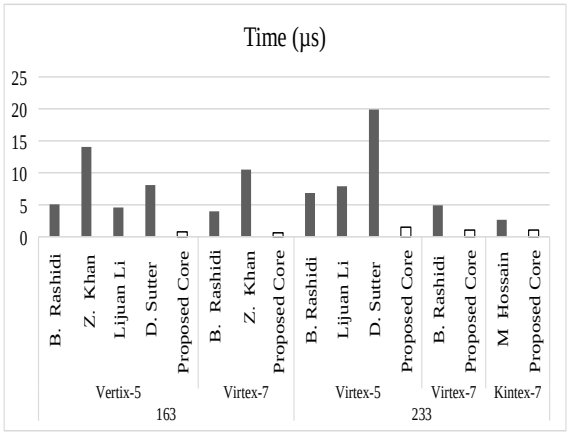

Figure 9: Time Comparison between the Proposed High-Speed Core and the Others.

provided by Xilinx (Przybus, 2010). Virtex-5, Virtex-7, Kintex-7 and Artix-7 FPGA families are used to implement the high-speed core.

The high-speed core has been synthesized, placed and routed using Xilinx ISE 14.4 design suite (Xilinx, 2012). The optimization goal has been set to the balance strategy. A time constraint is applied for all results to achieve better area-speed ratio with zero timing error. Table 1 presents the place and route results for the proposed high-speed ECC core. Table 2 includes our design compared with other previous ECC hardware implementations. The efficiency is defined as follows:

$$Efficiency = \frac{Numberofbits}{Time \cdot Slices} \quad (3)$$

The proposed high-speed core provides higher speed in both Virtex-7 and Kintex-7 FPGA devices, since they are fabricated and optimized at 28nm technology (Przybus, 2010). Artix-7 device consumes less resources which makes it well suited for the battery-powered cell phones, automotive, commercial digital cameras and IP cores of SoCs. The graphical representation in Figure 9 represents the time comparison between our proposed high-speed core and other designs. As seen in Figure 9, our design has the lowest execution time for performing a single successful SPM. As shown in Table 2, the efficiency of our proposed high-speed core outperforms the other ECC hardware implementations. The design in (Rashidi et al., 2016) provides higher frequencies but consumes about twice the resources of the proposed core over GF (2^{163}) and using the same FPGA device.

The area-efficient hardware implementation in (Khan and Benaissa, 2015) consumes less area than the proposed core but requires 95% more clock cycles. This large number of clock cycles comes from writing/reading of the distributed RAM-based memory and register shift operations. In (Li and Li, 2016), a pipelined architecture is applied to the SPM, which achieves high performance in terms of the area and working frequencies. However, it requires 84% extra clock cycles than our proposed clock cycles using the same device and field. Our high-speed core achieves better efficiency than the design in (Sutter et al., 2013). Our design has 88% less clock cycles, 33% higher frequency, 44% less slices than (Sutter et al., 2013). Using Kintex-7 FPGA device, the design in (Hossain et al., 2015) performs on less slices than the proposed core by 39% over GF (2^{233}). Although, large number of clock cycles is required for performing single SPM. In (Hossain et al., 2015), an iterative-based architecture is adopted by all main SPM operations, where the binary (left-to-right) algorithm is used for scalar multiplication, an interleaved field multiplier is implemented for the multiplication operations, and a modified Extended-Euclidian is applied for the inversion operation.

To sum up the comparisons, shifting registers, segmented multipliers, or memory-based implementations results in large latency (clock cycle) as in (Khan and Benaissa, 2015) and (Sutter et al., 2013). Iterative architecture is not the efficient way for achieving higher speeds as the work in (Hossain et al., 2015). Pipelining architecture is more practical to apply for achieving higher performance and maintaining the balance between speed and area, as works (Rashidi et al., 2016) and (Li and Li, 2016) do.

The balance in the speed-area ratio can be applied when the optimal number of pipelined stages are inserted. Our high-performance ECC core uses few slices with small latencies and high working frequencies. This high-speed area-efficient ECC core makes it very suitable to be used in different kinds of real-time embedded systems such as cellphone banking services, health-care monitoring using smart watches, and accessing office networks and storage devices while abroad.

5 CONCLUSIONS AND FUTURE WORK

In this paper, a high-speed area-efficient ECC core over GF (2^n) is proposed. Xilinx FPGA devices are used to implement the core, where the pipelining architecture is applied for achieving higher working frequencies. A merged-improved montgomery ladder scalar point method is developed for performing scalar multiplications (kP). Karatsuba-Ofman algorithm is used for performing field multiplication op-

Table 1: FPGA Results of the Proposed High-Speed ECC Core: After Place and Route.

GF (2^n)	Device	LUTs	F.Fs	Slices	Clock Cycles	Freq (MHz)	Time (μs)	Efficiency
163	Virtex-5	8,900	3,044	2,814	226	291	0.78	74584.42
233		15,672	4,333	4,585	327	217	1.51	33723.19
571		72,259	10,259	21,720	674	198	3.4	7722.93
163	Virtex-7	8,409	3,023	2,780	226	360	0.63	93397.85
233		16,003	4,323	4,762	327	310	1.05	46385.32
571		71,180	10,384	18,178	674	290	2.32	13515.38
163	Artix-7	8,417	3,016	2,693	226	191	1.18	51153.6
233		15,236	4,489	3,977	327	209	1.56	37445.44
163	Kintex-7	8,437	3,023	3,030	226	311	0.73	74028.16
233		15,221	4,483	4,915	327	309	1.06	44796.41

Table 2: Comparison between our High-Speed ECC Core and other Works over GF (2^n).

	GF (2^n)	Device	LUTs	F.Fs	Slices	Clock Cycles	Freq (MHz)	Time (μs)	Efficiency
Rashidi	163	Vertix-5	-	-	5,768	-	343	5.08	5562.87
	233		-	-	10,601	-	359	6.84	3213.32
	163	Virtex-7	-	-	5,575	-	437	3.97	7364.66
	233		-	-	10,528	-	496	4.913	4504.68
Khan	163	Virtex-5	3,958	1,522	1,089	4168	296	14.06	10645.71
		Virtex-7	4,721	1,886	1,476	4168	397	10.51	64.47
Lijuan	163	Vertix-5	9,470	4,526	3,041	1,363	294	4.6	11652.35
	233		15,296	6,559	4,762	1,926	244	7.9	6193.55
Sutter	163	Vertix-5	22,039	-	6,059	1,591	200	8.1	3321.26
	233		28,683	-	8,134	2,889	145	19.9	1439.46
	571		32,432	-	11,640	44,047	126	348	140.97
Hossain	233	Kintex-7	9,151	9,407	3,016	679776	255.66	2.66	29043.1
Proposed Core	163	Vertix-5	8,900	3,044	2,814	226	291	0.78	74584.42
	233		15,672	4,333	4,585	327	217	1.51	33723.19
	571		72,259	10,259	21,720	674	198	3.4	7722.93
	163	Virtex-7	8,409	3,023	2,780	226	360	0.63	93397.85
	233		16,003	4,323	4,762	327	310	1.05	46385.32
	571		71,180	10,384	18,178	674	290	2.32	13515.38
	163	Kintex-7	8,437	3,023	3,030	226	311	0.73	74028.16
	233		15,221	4,483	4,915	327	309	1.06	44796.41

eration. Itoh-Tsuji method is applied for mapping the LD coordinates back to the affine coordinates. In GF (2^{163}), A single scalar multiplication can be done in 0.63μs at 360Mhz working frequency in Virtex-7 FPGA devices using 2780 slices, which is the fastest area-efficient hardware implementation result. The proposed ECC core was developed and evaluated using Xilinx ISE 14.4. Place and route results show our implemented ECC core provide best performance in terms of latency and utilized area compared to other existing designs. The proposed ECC core would be suitable for platforms that require efficiency in terms of area/speed. Platforms deal with the public key cryptosystems such as key exchange agreements in ECDH and signing certificates in ECDS. On other hand, our proposed ECC core can be integrated and embedded with applications that have a security layer in its implementations, such as image steganographic engines. Design a separable secure image steganographic cryptosystem would be our next step in future.

ACKNOWLEDGEMENTS

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada and in part by the Regroupement Strategique en Microelectronique du Quebec (ReSMiQ).

REFERENCES

Amirtharajan, R. (2014). Dual cellular automata on fpga: An image encryptors chip. *Research Journal of Information Technology*, 6(3):223–236.

Ansari, B. and Hasan, M. A. (2008). High-performance architecture of elliptic curve scalar multiplication. *IEEE Transactions on Computers*, 57(11):1443–1453.

Dalal, M. and Juneja, M. (2018). A robust and imperceptible steganography technique for sd and hd videos. *Multimedia Tools and Applications*, pages 1–21.

Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654.

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472.

for Standardization (ISO), I. O. (2000). Cryptographic techniques based on elliptic curves.

Gallagher, P. (2013). Digital signature standard (dss). *Federal Information Processing Standards Publications, volume FIPS*, pages 186–183.

Großschädl, J. (2001). A bit-serial unified multiplier architecture for finite fields gf (p) and gf (2 m). In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 202–219. Springer.

Hankerson, D., Menezes, A. J., and Vanstone, S. (2006). *Guide to elliptic curve cryptography*. Springer Science and Business Media.

Harb, S. and Jarrah, M. (2017). Accelerating square root computations over large gf (2m). In *SECRYPT*, pages 229–236.

Harb, S. and Jarrah, M. (2019). Fpga implementation of the ecc over gf (2 m) for small embedded applications. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(2):17.

Hossain, M. S., Saeedi, E., and Kong, Y. (2015). High-speed, area-efficient, fpga-based elliptic curve cryptographic processor over nist binary fields. In *Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on*, pages 175–181. IEEE.

Karatsuba, A. A. and Ofman, Y. P. (1962). Multiplication of many-digital numbers by automatic computers. In *Doklady Akademii Nauk*, volume 145, pages 293–294. Russian Academy of Sciences.

Khan, Z. U. and Benaissa, M. (2015). Throughput/area-efficient ecc processor using montgomery point multiplication on fpga. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(11):1078–1082.

Kocher, P., Jaffe, J., and Jun, B. (1999). Differential power analysis. In *Annual International Cryptology Conference*, pages 388–397. Springer.

Li, L. and Li, S. (2016). High-performance pipelined architecture of elliptic curve scalar multiplication over gf (2^m). *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(4):1223–1232.

López, J. and Dahab, R. (1998). Improved algorithms for elliptic curve arithmetic in gf (2 n). In *International Workshop on Selected Areas in Cryptography*, pages 201–212. Springer.

Mahdizadeh, H. and Masoumi, M. (2013). Novel architecture for efficient fpga implementation of elliptic curve cryptographic processor over gf(2^{163}). *IEEE transactions on very large scale integration (VLSI) systems*, 21(12):2330–2333.

McGrew, D., Igoe, K., and Salter, M. (2011). Fundamental elliptic curve cryptography algorithms. Technical Report 2018, Internet Engineering Task Force (IETF).

Miller, V. S. (1985). Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer.

Montgomery, P. L. (1987). Speeding the pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177):243–264.

Moon, S. (2006). A binary redundant scalar point multiplication in secure elliptic curve cryptosystems. *IJ Network Security*, 3(2):132–137.

Percey, A. (2007). Advantages of the virtex-5 fpga 6-input lut architecture.

Peter, S. and LangendOorfer, P. (2007). An efficient polynomial multiplier in gf (2m) and its application to ecc designs. In *Design, Automation and Test in Europe Conference and Exhibition, 2007. DATE'07*, pages 1–6. IEEE.

Przybus, B. (2010). Xilinx redefines power, performance, and design productivity with three new 28 nm fpga families: Virtex-7, kintex-7, and artix-7 devices. *Xilinx White Paper*.

Rashidi, B., Sayedi, S. M., and Farashahi, R. R. (2016). High-speed hardware architecture of scalar multiplication for binary elliptic curve cryptosystems. *Microelectronics Journal*, 52:49–65.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.

Roy, S. S., Rebeiro, C., and Mukhopadhyay, D. (2013). Theoretical modeling of elliptic curve scalar multiplier on lut-based fpgas for area and speed. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(5):901–909.

Specification, P. (2006). Virtex-5 family overview.

Sutter, G. D., Deschamps, J.-P., and Imaña, J. L. (2013). Efficient elliptic curve point multiplication using digit-serial binary field operations. *IEEE Transactions on Industrial Electronics*, 60(1):217–225.

Wenger, E. and Hutter, M. (2011a). Exploring the design space of prime field vs. binary field ecc-hardware implementations. In *Nordic Conference on Secure IT Systems*, pages 256–271. Springer.

Wenger, E. and Hutter, M. (2011b). Exploring the design space of prime field vs. binary field ecc-hardware implementations. In *Nordic Conference on Secure IT Systems*, pages 256–271. Springer.

Xilinx, I. (2018a). Xilinx - adaptive and intelligent.

Xilinx, I. (2018b). Xilinx fpga devices, virtex, kintex, artix.

Xilinx, I. (April 24, 2012). Ise in-depth tutorial, complete guide (ug695). Technical Report 14.1, Xilinx, Inc.

Zhou, G., Michalik, H., and Hinsenkamp, L. (2010). Complexity analysis and efficient implementations of bit parallel finite field multipliers based on karatsuba-ofman algorithm on fpgas. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 18(7):1057–1066.