sciendo Journal of Electrical Engineering, Vol. 76, No. 3, 2025, pp. 241-255

A low-latency 163-bit ECC processor for IoT applications

Sumit Singh Dhanda[1], Savita Kumari[2], Vinod Kumar[3], Sachin Kumar Gupta[3]

Extensive research is being carried out on elliptic curve cryptography and its applications in resource-constrained environments such as IoT. Yet, critical gaps, such as high computational overhead in ECC operations, scalar multiplication and computationally expensive inversion, remain unanswered. Existing ECC implementations focus on improving either multiplication or inversion separately, but an integrated, optimized approach is lacking. There is a need for a processor that maintains cryptographic strength while significantly reducing computational latency. This research aims to bridge these gaps by designing a low-resource and low-latency 163-bit ECC processor that implements a hybrid Karatsuba multiplier and Quad-Itoh-Tsuji inversion for enhanced efficiency. It balances low latency, high security, and resource efficiency for IoT applications. Provides comparative analysis against existing ECC implementations to validate its performance. The processor is synthesized with PlanAhead software and implemented on several Virtex FPGAs. The most significant achievement of the design is its ability to minimize the latency to 991 cycles. In comparison to other designs, the implemented design is the most suited for Virtex-4 and 5. It is the smallest on Virtex-4 compared to other designs in the literature. In comparison to existing designs, the implemented approach saves 40-50% of resources on Virtex-4 and Virtex-7 implementations. Although the design's maximum frequency is lower than other designs, its area-time product yields good results. It also shows satisfactory values for the area-time product compared to previous designs.

Keywords: Elliptic Curve Cryptography (ECC), Internet of Things (IoT), Field Programmable Gate Arrays (FPGA), Karatsuba multiplier, information security

1 Introduction

Although the trend has started to shift towards network-on-chip (NoC) communication, the traditional communication method, with encryption to maintain confidentiality, will still remain the main method of communication. Asymmetric ciphers are inherently more versatile than symmetric ones [1]. Elliptic Curve Cryptography (ECC) is a popular asymmetric primitive with many protocols, such as 6LoWPAN, CoAP, etc., utilizing it to provide security [2-3]. This cipher's size, time, and power needs have been optimized after extensive research [4-5]. It delivers higher security per key bit than RSA, which in turn reduces the overhead. ECC has many operations involved in encryption. Scalar multiplication is one of the most expensive procedures.

Moreover, curve selection is also a significant consideration for encryption security and ease of implementation. Many different curves can be employed in ECC systems. The National Institute of Standards and Technology (NIST) provides details on these curves and parameters. IEEE-1363 [6] is an American National Standards Institute (ANSI) standard. The basic classification for the curves is prime curves and extension field curves. Due to the similarity in presentation, the hardware implementation is done mainly with binary extension curves. On the other hand, software implementation utilizes prime ones. Different coordinate systems can be chosen for these implementations. The Lopez-Dahab coordinate system cut down the inversions necessary for the calculation.

The underlying problem defines the strength of the cryptosystem. For ECC, the Elliptic Curve Discrete Logarithm Problem (ECDLP) is the problem to be solved. It is much harder for the general discrete logarithm problem or integer factorization theorem. The length of the key directly relates to the strength of the cryptosystem. The non-availability of such an algorithm that can compute the ECDLP is less than exponential time, making it secure in comparison to other similar ciphers. As a result, the key sizes necessary to guarantee adequate security are significantly smaller than other public key cryptosystems.

Elliptic Curve Cryptography (ECC) is a widely accepted asymmetric cryptographic technique, particularly beneficial for resource-constrained environments such as the Internet of Things (IoT). However, the high computational cost associated with scalar multiplication and inversion operations remains a significant challenge. While existing research has optimized ECC implementations for specific use cases, there is still a need for a low-latency, resource-efficient ECC processor that balances security and performance. This study addresses the following key research questions:

[1] School of Computer Science and Engineering, IILM University, Greater Noida, 201306, India
[2] School of Computer Science and Engineering, Galgotias University, Greater Noida, 201306, India
[3] Department of Electronics and Communication Engineering, Central University Jammu, Samba-Jammu (UT of J&K), 181143, India
dhandasumit@gmail.com, sachin.ece@cujammu.ac.in

https://doi.org/10.2478/jee-2025-0025, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

1. How can the latency of ECC computations be minimized without compromising security in IoT applications?
2. What architectural enhancements can be introduced to optimize multipliers and inversion units for efficient ECC performance?
3. How does the proposed ECC processor compare to existing implementations in terms of computational efficiency, power consumption, and security robustness?

The primary focus of this research is to ensure the security of IoT applications. These devices are constrained in terms of resources. The 163-bit NIST field is a well-balanced choice for secure, efficient, and scalable cryptographic implementations in IoT environments, ensuring strong encryption with minimal resource consumption.

In the Internet of Things (IoT), many security services are delivered by ECC. Inversion and Scalar Multiplication (SM) are two expensive processes involved in encryption. These consume the most of the resources. If the SM is optimized in a way that suits the needs of small devices, it will also make ECC lighter. One such attempt was presented in [7]. In [8], the authors used the Galois Field of 191 bits to implement elliptic curve point multiplication in a very efficient way. The multiplier has been adapted to propose a modified Karatsuba-Ofman multiplication. To reduce resource consumption in ECC, [9] offered an architecture that leveraged pipelining and parallelism, as well as efficient LUT placement and routing. Pipelining was further used to propose a high-speed FPGA implementation of ECC [10]. A similar attempt to create a fast ECC processor has been presented in [11]. Apart from optimizing calculation time, the slice consumption is another important parameter. In [12], 256- and 1024-bit modular multipliers have been used for implementing area and time-efficient Montgomery modular multipliers. In [13], a full assessment of lightweight cryptographic approaches for securing IoT networks is offered, including discussions on asymmetric cryptography. Energy efficiency is another important consideration. Algorithms that can minimize energy consumption will be given priority in such devices. Device-to-device (D2D) communication is one such application. An energy-efficient system has been designed using ECC for D2D communication. It is designed with smart city applications in mind [14].

Considering resource-constrained devices, a 256-bit compact Montgomery multiplier has been designed that uses a 256-bit adder cum subtractor [15]. It has achieved not only a small time but also an area. In [16], a state-of-the-art review regarding the application of ECC in the IoT context has been provided. It discusses the recent works on using ECC in different areas, such as smart grids, smart home systems, etc. Combining the Diffie-Hellman algorithm for key exchange with hybrid ECC is utilized to strengthen the secrecy of multi-hop IoT networks [17].

Despite the extensive research on Elliptic Curve Cryptography (ECC) and its applications in resource-constrained environments such as IoT, several critical gaps remain unaddressed:

1. High computational overhead in ECC operations
   - Scalar multiplication and inversion are computationally expensive, particularly for IoT devices with limited processing power.
   - Existing ECC implementations focus on improving either multiplication or inversion separately, but an integrated, optimized approach is lacking.
2. Latency vs. security trade-off
   - Many ECC architectures prioritize speed at the cost of security robustness, making them susceptible to side-channel attacks.
   - There is a need for a processor that maintains cryptographic strength while significantly reducing computational latency.
3. Resource-efficient hardware architectures
   - Current ECC processors often require high power and area consumption, limiting their applicability in energy-constrained devices.
   - Few studies explore hybrid techniques (e.g., Karatsuba multipliers and Quad-Itoh-Tsuji inversion) to optimize both power efficiency and processing speed.
4. Lack of benchmarking against modern ECC standards
   - Most existing research does not comprehensively compare ECC architectures against standardized cryptographic protocols.
   - There is limited empirical evaluation of ECC performance in real-world IoT environments.

Addressing the gaps:

This research aims to bridge these gaps by designing a 163-bit ECC processor that:

- Implements a hybrid Karatsuba multiplier and Quad-Itoh-Tsuji inversion for enhanced efficiency.
- Balances low latency, high security, and resource efficiency for IoT applications.
- Provides comparative analysis against existing ECC implementations to validate its performance.

The rest of the paper is organized as follows: the rationale behind coordinate selection, basic operations, and the proposed design, with integral parts, are discussed in Section 2. Implementation setup and details, along with the results and their analysis, are discussed in Section 3. A conclusion is drawn, and future work based on the current shortcomings of the proposed design is presented in Section 4.

2 Projective coordinates and processor design

A particular field type utilizes a specific equation for point generation in the field. Weierstrass equation is used in a prime field GF (p). It is written as: $E_{a,b}^{W}: y^2 = x^3 + ax + b$ with $4a^3 + 27b^2 \neq 0$, but in the binary extension field, GF (2n) will be written as

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

with $b \neq 0$

To ensure the safety of the ECC implementation, the field's order should be bigger than 2160. We investigated the binary extension field GF (2163) for the ECC implementation. Though the Montgomery approach is efficient and fast, here, scalar multiplication is performed using the double and add algorithms. However, because there are more inversion operations in affine coordinates, the procedure takes longer.

Scalar multiplication is the most time-consuming operation on an elliptic curve. It is carried out via a series of additions. As a result, the process is completed by adding and doubling points.

Scalar multiplication uses two important operations: addition and doubling. These are presented in Eqns. (2) through (13). When two points, P1 and P2, are given, the procedure uses the chord and tangent rule to determine the third point. In this scenario, there are two possibilities: either the two points (P_1 and P_2) are distinct, or they are identical. In the first situation, point addition is utilized, whereas in the second case, point doubling is used to get the curve's third point.

It makes use of a simple notion known as coordinate geometry. The variable 'γ' represents the slope of the line connecting two points. The calculation is as follows:

$$\gamma = \frac{y_2 - y_1}{x_2 - x_1} \tag{2}$$

Point addition is carried out in the case when $P_1 \neq P_2$ and P3 is calculated as follows:

$$x_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \left(\frac{y_1 + y_2}{x_1 + x_2}\right) + x_1 + x_2 + a \tag{3}$$

$$y_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right) + (x_1 + x_3) + x_3 + y_1 \tag{4}$$

Point doubling is employed when P_1=P_2. The slope value is now calculated using Eqn. (5). Because the calculation is in the GF(2m) field, the slope equation changes to (6).

$$\gamma_1 = \frac{3x_1^2 + 2ax_1 + y_1}{2y_1 + x_1} \tag{5}$$

$$\gamma_1 = x_1 + \frac{y_1}{x_1} \tag{6}$$

$$x_3 = x_1^2 + \frac{b}{x_1^2} \tag{7}$$

$$y_3 = x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right)x_3 + x_3 \tag{8}$$

If $P_1 \neq P_2$, then equations can be further simplified as

$$x_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \left(\frac{y_1 + y_2}{x_1 + x_2}\right) + x_1 + x_2 + a \tag{9}$$

$$x_3 = \frac{x_1y_2 + x_2y_1 + x_1x_2^2 + x_2x_1^2}{(x_1 + x_2)^2} \tag{10}$$

Here $P = (x_0, y_0)$, and as $P = P_2 - P_1$, then

$$x = \frac{x_1y_2 + x_2(x_1 + y_1) + x_1x_2^2 + x_2x_1^2}{(x_1 + x_2)^2} \tag{11}$$

By combining Eqns. (10) and (11)

$$x_3 = x_1 + \left(\frac{x_1}{x_1 + x_2}\right)^2 + \left(\frac{x_1}{x_1 + x_2}\right) \tag{12}$$

Finally, we have

$$x_3 = \begin{cases} x_1 + \left(\frac{x_1}{x_1 + x_2}\right)^2 + \left(\frac{x_1}{x_1 + x_2}\right) \text{ if } P_1 \neq P_2 \\ x_1^2 + \frac{b}{x_1^2} \text{ if } P_1 = P_2 \end{cases} \tag{13}$$

This simplification helps to eliminate the y-coordinate from the total calculation. Hence, it can be calculated using the x-coordinate at the final stage. Then, $P_2 = P_1 + P$ can be calculated using Eqns. (9) and (10).

$$x_2 = \frac{x_1y + xy_1 + x_1x^2 + xx_1^2}{(x_1 + x)^2} \tag{14}$$

$$y_1 = \left(\frac{x_1}{x} + 1\right)\{(x_1 + x)(x_2 + x) + x^2 + y\} + x \tag{15}$$

As a result, the loop process requires only the x coordinate. It helps to reduce the number of finite field operations. Scalar multiplication, when calculated using affine coordinates, the number of finite field inversions is very large. In standard projective co-ordinates, when $Z \neq 0$, then

$$(X, Y, Z) \leftrightarrow (X/Z\,, Y/Z) \tag{16}$$

By substituting $(X/Z\,, Y/Z)$ in Eqn. (16), the projective form of the EC is as follows:

$$ZY^2 + XYZ = X^3 + aX^2Z + bZ^3 \tag{17}$$

Lopez-Dahab is the most suitable system for minimizing the number of inversions in finite fields. In the Lopez-Dahab system, a point on $E(GF(2^m))$can also be written as (X, Y, Z). The relation with affine coordinates can be expressed as $(X, Y, Z) \leftrightarrow (X/Z\,, Y/Z^2)$. Hence, the elliptic curve from Eqn. (1) becomes

$$E: Y^2 + XYZ = X^3 + aX^2Z^2 + bZ^4 \tag{18}$$

The use of Lopez-Dahab coordinates helps in optimizing the Montgomery method, as Eqn. (13) can now be written as If $P_1 = P_2$, then

$$\begin{aligned} X_3 &= X_1^2 + bZ_1^4 \\ Z_3 &= Z_1^2 X_1^2 \\ Y_3 &= b.Z_1^4.Z_3 + X_3.(a.Z_3 + Y_1^2 + b.Z_1^4) \end{aligned} \tag{19}$$

It can be seen that the doubling now needs 5 FFMs, and inversion is not required. Similarly, if $P_1 \neq P_2$, then

$$\begin{aligned} X_3 &= xZ_3 + (X_1Z_2)(X_2Z_1) \\ Z_3 &= (X_1Z_2 + X_2Z_1)^2 \end{aligned} \tag{20}$$

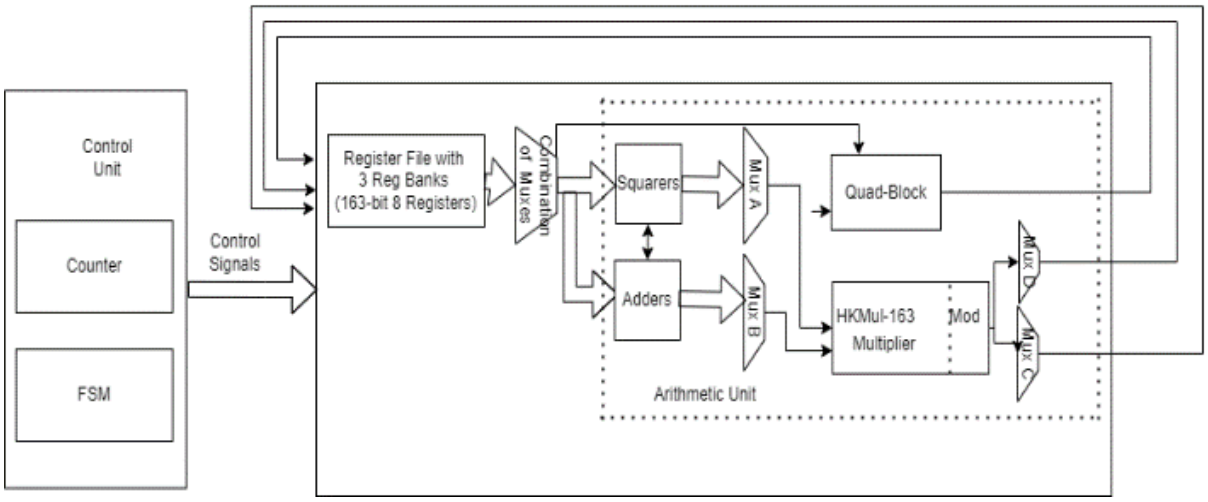

Fig. 1. Architectural blocks of the proposed design

Figure 1 displays the proposed design of ECCP-163. It consists of a control unit with a counter and a Finite State Machine (FSM) to perform all operations. The second component is a register file which contains three register banks comprised of eight 163-bit registers. This register file is coupled to the Arithmetic Unit (AU) via several multiplexers that aid in the selection of variables to be provided to AU. Inside AU, squaring and adding operations are performed on the input variables (a total of five), which are then selected by multiplexers A and B before being given to HKMul-163. The flowchart for the complete process of the FSM is provided below.

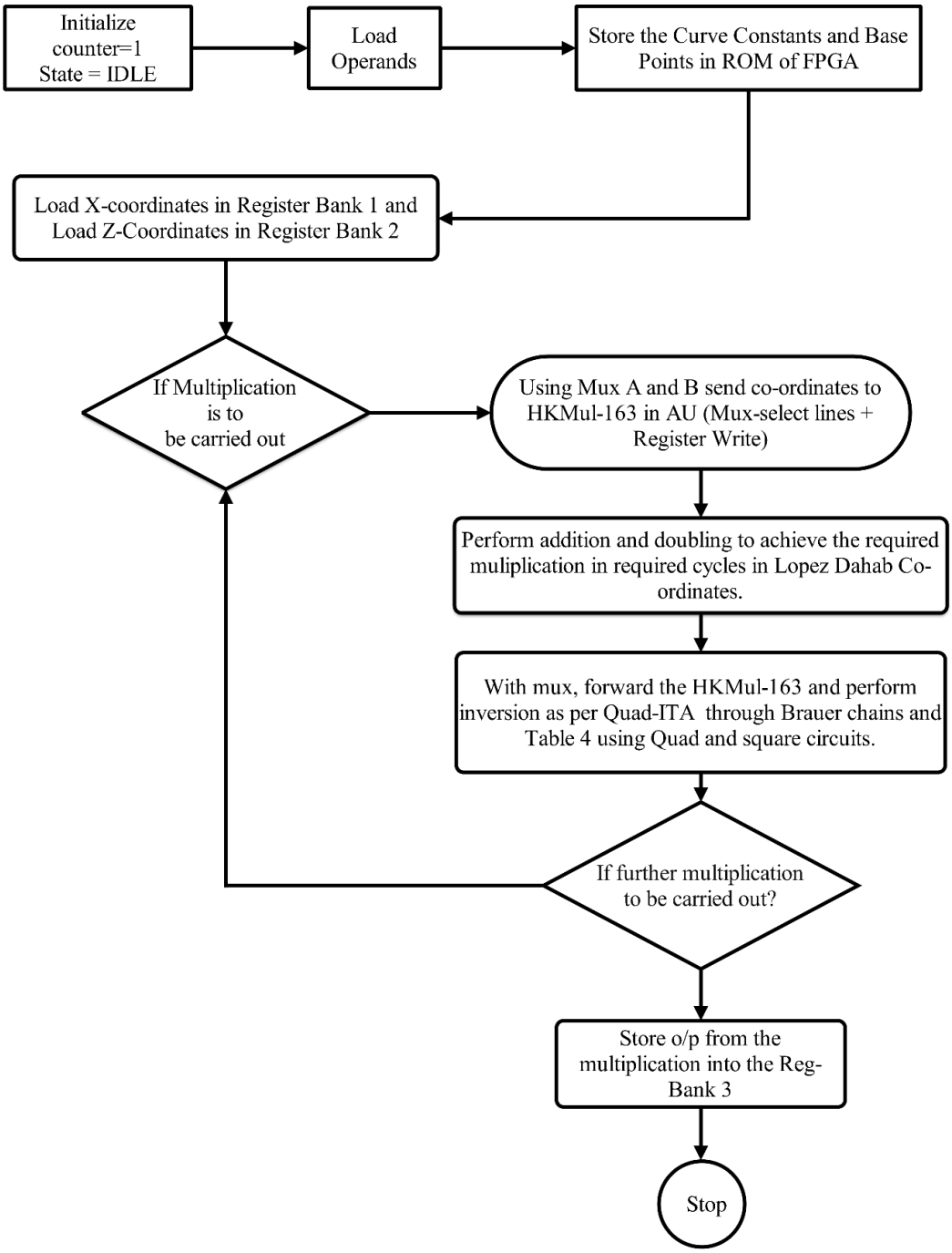

Fig. 2. Flowchart for the ECCP-163

At the beginning of each cycle, the AU receives operands from the register bank, which contains eight registers of 163 bits in size. At the end, the register bank stores the calculated results. Figure 2 describes the steps in details. It follows the process mentioned as per the details of units in coming sections, like inversion, which follows the process as per Tab. 4.

Multiplier and inversion algorithms are an integral part of AU. The Lopez-Dahab coordinate system is used to implement it. To minimize the resource requirement the field multiplier is being utilized by Quad-ITA for the inversion. All of these complex operations are controlled and synchronized by the control unit. The rationale behind choosing the 163-bit field is explained below in Tab. 1.

The 163-bit NIST field is a well-balanced choice for secure, efficient, and scalable cryptographic implementations in IoT environments, ensuring strong encryption with minimal resource consumption.

Table 1. Factors for selecting 163-bit field

	Factor	Justification
1	Optimized security with lower bit-size	• *Meets Modern Cryptographic Standards:* The 163-bit NIST ECC field provides security equivalent to 1024-bit RSA, offering strong encryption with a much smaller key size. • *Resistant to Quantum Attacks:* While post-quantum cryptography is still evolving, ECC over binary fields is considered more resilient than RSA against potential quantum threats.
2	Computational efficiency for IoT devices	• *Lower Power Consumption:* Since IoT devices often run on batteries or energy-harvesting systems, reducing computational overhead is crucial. A 163-bit ECC processor consumes significantly less power compared to RSA and higher-bit ECC fields. • *Faster Computations:* Compared to higher-bit ECC curves (e.g., 256-bit, 384-bit), a 163-bit field enables faster scalar multiplications and point operations, making it ideal for time-sensitive IoT applications.
3	Memory and storage optimization	• *Smaller Key Size = Lower Storage Requirements:* IoT devices have limited RAM and storage. The 163-bit field provides strong encryption with reduced key storage overhead, which is essential for embedded systems. • *Efficient Hardware Implementations:* The use of binary fields allows for efficient hardware accelerators and custom processor designs, such as those leveraging hybrid Karatsuba multipliers for optimized performance.
4	Suitability for low-latency communication	• *Ideal for Real-Time Secure Communication:* IoT applications in smart homes, healthcare, and industrial automation require fast cryptographic operations. The 163-bit ECC field provides a balanced trade-off between security and real-time performance. • *Efficient Error Handling in Wireless Networks:* The binary nature of GF(2^p) arithmetic allows simpler and more efficient error correction methods in wireless sensor networks.
5	Compliance with standardized security protocols	• *Adopted in IoT Security Standards:* The NIST-recommended ECC-163 curve is widely supported in TLS/SSL, secure boot, and IoT authentication protocols. • *Interoperability with Existing Systems:* Since many cryptographic libraries and protocols (e.g., TLS 1.2, IPSec, and Zigbee security) support 163-bit ECC, integrating it into IoT ecosystems is seamless.

2.1 Hybrid Karatsuba multiplier

Multipliers are the most resource-consuming part of an ECC processor. Karatsuba multipliers can help reduce the resources and time needed for calculation. Table 2 above provides a detailed comparison among the different type of multipliers based on three different factors. It also underlines the need of such multipliers that can provide a trade-off between area and latency.

Table 2. Comparison of different multipliers

Multiplier type	Space complexity	Time complexity	Latency
Montgomery modular multiplier	It requires additional registers and logic for moderate area utilization	Reduced time complexity	Streamlined computation results in low-latency
Booth multiplier	Higher area requirements	Reduces the time complexity	Moderate latency: Additional delays can be there
Quadratic multiplier	Higher area consumption due to the replication components	Low time complexity due to parallel processing	Achieves minimal latency at the cost of resources
Karatsuba multiplier	Increased memory usage and area due to the recursion	Reduced time complexity, approximately O(n^1.58), compared to the traditional O(n^2) complexity	Reduced latency for large operand, large overhead for smaller sizes
Combinational multiplier	Increased area requirements due to storage needs	Reduced time complexity	Immediate precomputed results leads to low latency

In [18], a multiplier with sub-quadratic complexity was designed for a Galois Field of 191 bits. It was a combination of two Karatsuba multipliers. Hence, the name hybrid Karatsuba multiplier (HKMul) was given. On the same design, with some changes, a hybrid-karatsuba multiplier (HKMul) is designed for the GF (2^{163}) field with $x^{163}+x^7+x^6+x^3+1$ as the irreducible polynomials. The basic polynomial product can be expressed as:

$$C = x^m A^H B^H + (A^H B^L + A^L B^H) x^{\frac{m}{2}} + A^L B^L \quad (21)$$

Equation (21) can be re-arranged as

$$C = x^m A^H B^H + A^L B^L + \left(A^H B^H + A^L B^L + (A^H + A^L) \times (B^H + B^L)\right) x^{\frac{m}{2}} \quad (22)$$

This is the Karatsuba-Ofman algorithm which can be written as

$$C = x^m C^H + C^L \quad (23)$$

Here, the *C* is divided into two parts, i.e., two polynomial products are there in the final equation below

$$C^H = [c_{2m-2}, c_{2m-1}, \ldots\ldots, c_{m+1}, c_m] \quad (24)$$

$$C^L = [c_{m-1}, c_{m-2}, \ldots\ldots, c_1, c_0] \quad (25)$$

The Hybrid Karatsuba-Ofman multiplier employed here is composed of the general and simple-karatsuba multipliers. The primary reason for this design is to maximize the use of the FPGA architecture. According to [19], while the general karatsuba multiplier consumes the most gates, it consumes fewer slices due to the presence of input gates and the size of the LUT. FPGAs can function similarly to conventional K-O multipliers. The multiplication has five levels of computation. The first level consists of a 163-bit multiplier. The second computation level is comprised of 82- and 81-bit multipliers. The third level contains 41-bit and 40-bit multipliers, followed by 21- and 20-bit multipliers on the fourth level. The fourth level's multipliers are universal Karatsuba multipliers, whereas the remaining levels use simple Karatsuba multipliers.

2.2 Quad-Itoh-Tsuji inversion

Inversion is the second most costly operation in ECC processors. Quad-Itoh-Tsuji Inversion [20] is used in this design. It is an improved version of the Itoh-Tsuji algorithm (ITA). ITA performs better than the Extended Euclidean Algorithm (EEA), Montgomery Inversion, Binary Euclidean (BEA), etc. in terms of computation time. The use of Brauer-chains in ITA in combination with exponentiation circuits helps in faster and less expensive calculation of the inverses. It happens as a result of the reduction in the size of the Brauer-chain from *n* to (*n*–1)/2. Tables 3 and 4 show the details of the Quad-Itoh Tsuji algorithm for this design.

The quad-block contains 14 quad circuits, which are used to raise the input power cyclically. To calculate α^(4^28), a quad-block calculates α^(4^14) in a single step and returns the output to the input after passing through the circuit once more. 'Bsel' selects 9 as the output for α^(4^9), using four control lines. However, the Quad-ITA will use the 22 cycles to execute the inversion individually. However, because it is part of the ECCP, it will make use of the inner circuitry, completing the conversion from LD-projective to affine coordinates in 20 cycles.

Table 3. Itoh-Tsuji inversion algorithms with squares

i	μ_i	$\left[\beta_{\mu_{i_1}}(a)\right]^{2^{\mu_{i_2}}} \times \beta_{\mu_{i_2}}(a)$	$\beta_{\mu_i}(a) = a^{2^{\mu_i}-1}$
0	1	-	$\beta_{\mu_0}(a) = a^{2^1-1}$
1	2	$\left[\beta_{\mu_0}(a)\right]^{2^{\mu_0}} \times \beta_{\mu_0}(a)$	$\beta_{\mu_1}(a) = a^{2^2-1}$
2	3	$\left[\beta_{\mu_1}(a)\right]^{2^{\mu_0}} \times \beta_{\mu_0}(a)$	$\beta_{\mu_2}(a) = a^{2^3-1}$
3	5	$\left[\beta_{\mu_2}(a)\right]^{2^{\mu_1}} \times \beta_{\mu_1}(a)$	$\beta_{\mu_3}(a) = a^{2^5-1}$
4	10	$\left[\beta_{\mu_3}(a)\right]^{2^{\mu_3}} \times \beta_{\mu_0}(a)$	$\beta_{\mu_4}(a) = a^{2^{10}-1}$
5	20	$\left[\beta_{\mu_4}(a)\right]^{2^{\mu_4}} \times \beta_{\mu_3}(a)$	$\beta_{\mu_5}(a) = a^{2^{20}-1}$
6	40	$\left[\beta_{\mu_5}(a)\right]^{2^{\mu_5}} \times \beta_{\mu_4}(a)$	$\beta_{\mu_6}(a) = a^{2^{40}-1}$
7	80	$\left[\beta_{\mu_5}(a)\right]^{2^{\mu_5}} \times \beta_{\mu_5}(a)$	$\beta_{\mu_7}(a) = a^{2^{80}-1}$
8	81	$\left[\beta_{\mu_7}(a)\right]^{2^{\mu_6}} \times \beta_{\mu_6}(a)$	$\beta_{\mu_8}(a) = a^{2^{81}-1}$
9	162	$\left[\beta_{\mu_8}(a)\right]^{2^{\mu_8}} \times \beta_{\mu_8}(a)$	$\beta_{\mu_9}(a) = a^{2^{162}-1}$

Table 4. Itoh-Tsuji inversion algorithms with Quad-circuits

	Calculate	C1	C2	C3	QB	B_{sel}	Clock cycle
1	$\alpha_1(a)$	0 0	0 2	X X	X X	1 1	1 2
2	$\alpha_2(a)$	X 1	X 1	0 X	1 X	0 1	3 4
3	$\alpha_4(a)$	X 2	X 1	0 X	2 X	0 1	5 6
4	$\alpha_5(a)$	X 1	X 1	0 X	1 X	0 1	7 8
5	$\alpha_{10}(a)$	X 2	X 1	0 X	5 X	0 1	9 10
6	$\alpha_{20}(a)$	X 2	X 1	0 X	10 X	0 1	11 12
7	$\alpha_{40}(a)$	X X 2	X X 1	0 1 X	14 6 X	0 0 1	13 14 15
8	$\alpha_{80}(a)$	X X X 2	X X X 1	0 1 1 X	14 14 12 X	0 0 0 1	16 17 18 19
9	$\alpha_{81}(a)$	X 1	X 1	0 X	1 X	0 1	20 21
10	a^{-1}	2	2	X	X	1	22

QB is the Quad-block selection signal, B_{sel} is the control signal for selection of output.

The quad-block comprises of different circuits that generates fourth power of supplied input. In total 14 circuits are cascaded. When input is supplied, quad-block raises its power to in^4, in^{4^2}, in^{4^3}, ..., $in^{4^{14}}$. These powers are selected with the help of control lines C1 C2 and C3 of different multiplexers. B_{sel} ensures the right output is transferred. The buffers are used to store the output from multiplier and quad-block. In one clock cycle either of the two is active, but not together. The register bank can be used for holding of the result of inversion algorithm if it needs to be used in further calculation. The controller in FSM generates the requisite signals to carry out all the within the quad-block. Control signals are generated for the multiplexer selection lines, buffer enables, and access signals to the register bank at each clock cycle. The calculations for inversion are presented in Tab. 3. The control signals produced by the controller are depicted in Tab. 4. The total cycles required for a^{-1} are 22.

2.3 Finite state machine / control unit

To coordinate these units and synchronize them to work efficiently, a complex control mechanism is required. This is achieved by the finite state machine. It is designed in such a way that HKMul-163 can be used both for multiplication and inversion to save the resources required for the implementation. This control unit completes the process in 38 cycles. A total of 33 control signals are generated to coordinate these operations. Out of these, 10 control signals are required for the multiplier's inputs and two outputs. The inversion unit needs 5 control signals. The details of these signals concerning the calculated exponent are shown in Tab. 4. The remaining 18 signals are used in the read and write operations of 3 register banks.

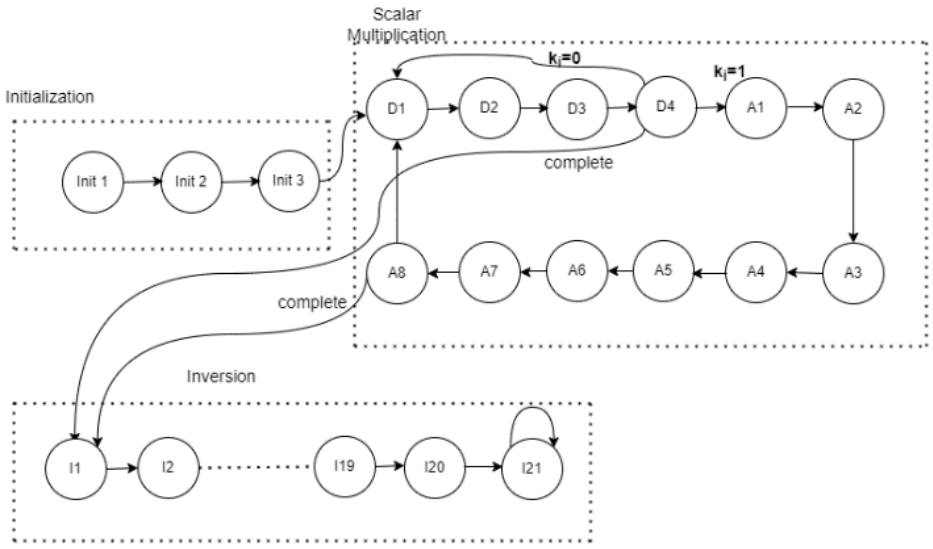

Fig. 3. Finite state machine for ECCP-163

There are 3 phases in the computation, as shown in Fig. 3. In the first step, the process is initialized over three clock cycles. The second phase involves multiplication in projective coordinates, addition, and doubling in GF(2^{163}). This product must be brought back in affine coordinates. Hence, inversion is carried out in the third phase. It takes 21 cycles. The total amount of cycles necessary for the processor [19] to calculate the whole multiplication are

$$\#CC = 3 + 12(h-1) + 4(l-h) + 20$$
$$= 11 + 8h + 4l$$

If scalar multiplication is represented as (k*P) and k is the scalar, then 'l' represents its length in bits, and 'h' is the hamming weight.

3 Results and discussion

The proposed design has been coded and implemented in Verilog. After synthesizing with Xilinx's PlanAhead Integrated Development Environment (IDE), it is implemented on different Xilinx Field Programmable Gate Arrays (FPGAs). The number of ports required for the design was the deciding factor in the board's selection. The findings were acquired and compared to the existing designs (Tabs. 5, 6, and 7). This demonstrates a significant improvement in resource consumption (LUTs) over previous works. When the small gadget is used, resources become critical. This 163-bit ECC processor is synthesized using PlanAhead software and implemented on Virtex-4, 5, 6, and 7, as well as the Spartan-3 FPGA.

Due to space constraints, the design used 12162 slices and 23551 LUTs on the Spartan-3 FPGA, denoted by S-3 in Tab. 5 and Fig. 4. Similarly, Virtex-4, -5, and -7 are designated as V-4, V-5, and V-7, respectively. The Virtex-4 implementation utilized 23330 LUTs and 12010 slices. The Virtex-5 implementation required 6949 slices and 19317 LUTs. The design used 5511 slices and 16813 LUTs on the Virtex-7 FPGA. The architecture required 991 cycles for scalar multiplication (k.P). The S-3, V-4, V-5, and V-7 have maximum clock frequencies of 87.5, 107.5, 124.5, and 160.8 MHz, respectively.

The design performance is compared against several implementations accessible in the literature. Tables 5, 6, and 7 compare the design based on its details for Virtex-4, 5, and 7. While some implementations on Virtex-2 and E were compared to Spartan-3 implementations. There are four primary metrics for comparison: resource consumption in terms of LUTs and slices, latency in terms of clock cycles used to compute the scalar multiplication, calculation time required for the scalar multiplication, and area-time product for the scalar multiplication. The A×T product is used for extensive comparisons in the literature.

First and foremost, the comparison focuses on resource use. Table 5 compares the implementation specifics to the Virtex-2 and -E-based designs shown in Fig 4. The comparison is based on resource use. Here, the proposed design is compared to [25–27]. Design outperforms [25] by 8.58% in terms of LUT usage. It utilizes 17% more LUTs than [27]. While increasing slice consumption of design [26] by 35%.

Figure 5 shows a resource comparison of the design with Virtex-4-based implementations seen in the literature. It indicates that the design uses 11.5% fewer LUTs and 26% fewer slices than the design [10]. Many implementations additionally report the flip-flops utilized in the designs, which are likewise a significant resource in FPGA. Our concept employs only 87 flip-flops, compared to 6683 in design [28], 7962 in design [10], and 3077 in design [11]. These designs are resource-intensive. While the percentage improvement over [11] is only 1.34% in terms of LUTs, it is 6.18% in slices.

Figure 5 shows that our design is the most limited of all the designs offered. Compared to the designs of [28, 29] and [23, 24], it improves slices by 4.14%, 32%, 41.44%, and 50%, respectively. When it comes to LUTs, there is a 63% improvement over [28]. Though [29] utilizes 2.22% fewer LUTs, it comes at the cost of 6683 F/F, resulting in a 4% increase in slice consumption.

When measured in terms of clock cycles, the proposed design has the lowest latency of all the designs in the table. In terms of calculation time, the suggested design takes the third-lowest time to complete scalar multiplication after design [11] and [24], with 5.32 μs and 7.7 μs, respectively.

Table 5. Comparison of Virtex-4/E/2 implementation

Architecture	LUTs / FFs	Slices	Frequency (MHz)	CC	Time (μs)
Prop. V-4 Work S-3	23330 23351	12010 12162	107.5 87.3	991 991	9.21 11.35
[11] V-4	23648/3077	12964	210	1119	5.32
[24] V-4	-	20807	185	1428	7.7
[23] V-4	-	24363	143	1446	10
[27] XCV 2000E-7	19508	-	150		143
[26] XC2V 6000	-	18079	100		106
[25] XCV 400E	25763	-	76.7		48
[10] V-4	26364/7962	16209	154	3010	19.55
[35] V-4	24363	-	143	1446	10
[29] V-4	33414	17929	250	2751	9.60
[28] V-4	22815 6683	12834	196	3372	17.20
[21] V-4	27889		133	2128	16.00

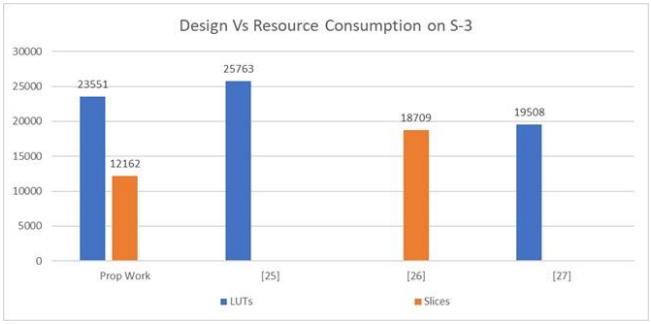

Fig. 4. Comparison of resource consumption of designs with Proposed one on Virtex-E and Virtex-2 for ECCP-163

X-axis=Design Name
Y-axis=Number of LUTs/ Slices/F/Fs

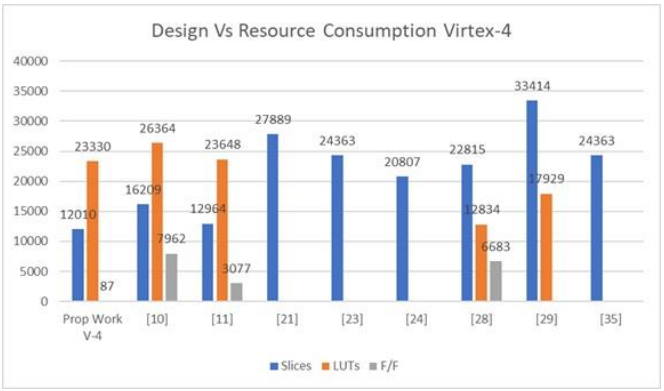

Fig. 5. Comparison of resource consumption of designs with Proposed one on Virtex-4 for ECCP-163

X-axis=Design Name
Y-axis=Number of LUTs/ Slices/F/Fs

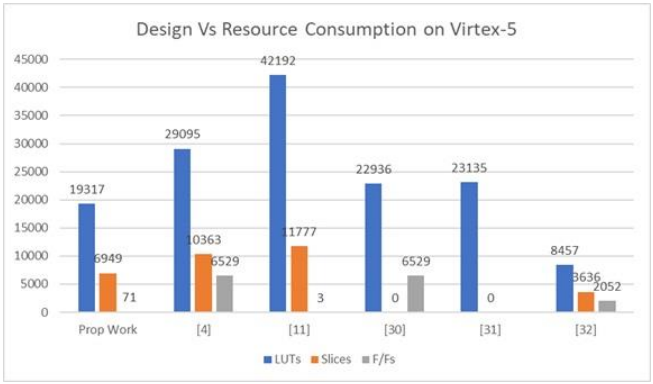

Fig. 6. Comparison of resource consumption of designs with Proposed one on Virtex-5 for ECCP-163

X-axis=Design Name
Y-axis=Number of LUTs/ Slices/ F/Fs

Table 6 compares the design to available Virtex-5 implementations. In terms of resource use, this design is the smallest of all the designs. Figure 6 represents the results from Tab. 6. The design saves 33.6% and 54% on LUTs compared to [4] and [11], respectively. Meanwhile, the savings in slices are 33% and 41%, respectively. Though the design [32] appears smaller than the proposed one but it uses 2052 F/Fs, which makes it appears good by using the available resources on the slice. But the cost is paid in terms of time, which is 305 % slower in time and consumes 2435 cycles more. The proposed design is the smallest of all the designs, and processors pay the price in terms of performance when computing scalar multiplication, where the design is the slowest of all the designs. This can be enhanced by implementing buffer registers and pipelining.

Table 7 shows the implementation details and comparisons of the designs on the Virtex-7 FPGA. It is represented in Fig 7. The proposed design is the fourth smallest of all the designs in Table 7. It requires around 40% more resources than [33] and [34].

Table 6. Comparison of Virtex-5 implementation

Architecture	LUTs /FF	Slices	Frequency (MHz)	CC	Time (µs)
Proposed	19317/71	6949	124.5	991	7.96
[4]	29095/6529	10363	153	780	5.10
[11]	42192	11777	159	450	2.83
[31]	22936/ 6529 ff	-	250	1371	5.48
[30]	23135	-	145	-	2.22
[32]	8457/2052	3636	106	3426	32.3

Table 7. Comparison of Virtex-7 implementation

Architecture	LUTs /FF	Slices	Frequency (MHz)	CC	Time (µs)
Proposed	16813 71	5511	160.78	991	6.163
[11]	41090 7969	11657	159	450	2.83
[4]	27105 6529	8736	223	780	3.50
[11]-b	14202 3747	4150	352	1119	3.18
[33]	9965		369	3959	10.73
[34]	10128		135	3415	25.3

While 11-b consumes 15.5% less LUTs, this is due to the design of 3747 F/Fs. However, its slice consumption is 25% lower than the planned design. [11-b]'s highly pipelined architecture also contributes to its faster computing times.

The design is by 40% and 59% smaller than [4] and [11]. However, it is slower than these designs. The pipelined architecture allows for faster performance. Now, the comparison will be done in terms of area-time product. Again, there will be three separate tables for V-4, V-5, and V-7 implementations. This area-time product has been estimated in two ways in the literature for FPGA implementations. The first is the LUTs-Time product presented in [22], and the second is the Slices-Time product described in [11]. Both calculations are shown in each table. Table 8 shows the area-time product for Virtex-4 implementations using the proposed design. It demonstrates that the proposed design is second best to the design [11].

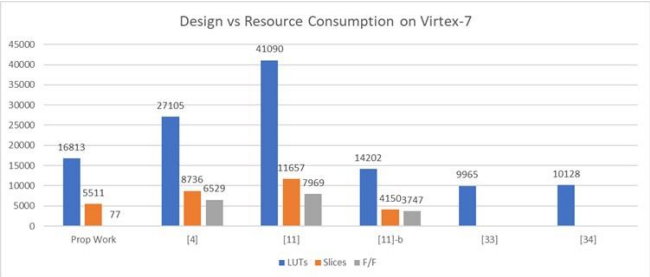

Fig. 7. Comparison of resource consumption of designs with Proposed one on Virtex-7 for ECCP-163
X-axis=Design Name
Y-axis=Number of LUTs/ Slices/ F/Fs

Table 8. Area-Time product comparison for Virtex-4 implementation

Design	$A \times T \times 10^{-3}$ (LUTs × Time)	$A \times T \times 10^{-3}$ (Slices × Time)
Proposed S-3 Work V-4	267.342 215.070	138.039 110.715
[10]	515.416	316.886
[11]	125.807	68.968
[21]	446.224	-
[23]	-	243.630
[24]	-	160.213
[25]	1236.624	-
[26]	-	1916.374
[27]	-	2789.644
[28]	360.145	--
[29]	320.774	-
[35]	243.630	-

It reaches 215.070 and 110.715 for the Virtex-4 design in the LUT × Time and Slices × Time parameters, respectively. As a result, the whole design is more effective. Table 9 shows the identical calculations for the Virtex-5 implementations. The suggested design records for the two goods are 153.760 and 55.314, respectively. It is the second lowest among all the designs, indicating that the design's frequency performance must be improved.

Table 10 shows the findings for the Virtex-7 implementations of the area-time product. The design generates 103.63 LUTs ×Time and 33.964 slices × Time. It ranks third in terms of LUTs-Time products and is comparable to [4] and [11]. So, the whole design produces satisfactory results while pointing to improved frequency performance.

Table 9. Area×Time product comparison for Virtex-5 implementation

Design	A×T × 10^{-3} (LUTs × Time)	A×T × 10^{-3} (Slices × Time)
Proposed work	153.760	55.314
[4]	148.384	52.10
[11]	119.403	33.329
[31]	125.689	
[30]	51.359	

Table 10. Area×Time product comparison for Virtex-7 implementation

Design	A×T × 10^{-3} (LUTs × Time)	A×T × 10^{-3} (Slices × Time)
Proposed work	103.630	33.964
[4]	94.867	32.989
[11]	116.284	30.576
[11-b]	45.162	-
[33]	106.924	-
[34]	258.515	-

The complete analysis depicts a picture where design [11] appears comparatively better. However, the proposed design is efficient. There are two reasons behind terming this design efficient. 40-50% reduction in resources as compared to [11]. It also achieves a low Area-Time product. Therefore, its design is able to provide low resource consumption, which provides a suitable trade-off between latency and resource consumption. Therefore, it can be useful in applications that need fewer resources but moderate throughput.

4 Conclusion

This paper presents a low-latency 163-bit ECC processor optimized for IoT applications, addressing key challenges such as high computational overhead and resource constraints. By employing a hybrid Karatsuba multiplier with low space complexity alongside a quad-Itoh-Tsuji inversion technique, the proposed design achieves significant reductions in computation time while optimizing hardware utilization. Synthesis and FPGA implementation results validate the processor's efficiency, demonstrating its suitability for real-time, resource-constrained environments. The most significant achievement of the design is its ability to minimize the latency to 991 cycles. Compared to others, the proposed architecture is the most suited for Virtex-4 and 5. Although its maximum frequency is lower than that of other designs, its area-time product yields good results.

The frequency of performance is a matter of concern, and we shall target this in the future. The frequency of operation can be improved through efficient pipelining and scheduling. It will, in turn, improve the timing and area-time product of the design. Looking ahead, future work will focus on further optimizing the architecture for ultra-low-power applications, making it more energy-efficient for battery-operated IoT devices. Additionally, expanding support for higher-bit ECC curves and investigating hardware acceleration techniques using AI-driven optimizations could further enhance performance and security. Integrating post-quantum cryptographic mechanisms is also a promising direction to ensure long-term resilience against emerging cryptographic threats.

Acknowledgment

This work was supported in part by Agency ABCBA under contract number 12/34/2000.

References

[1] S. Sankar, R. Gupta, J. Jose, S. Nandi, *Sec-NoC: A Lightweight Secure Communication System for On-Chip Interconnects*, IEEE Embedded Systems Letters, Vol. 16, No. 2, pp. 214-217, 2024, doi: 10.1109/LES.2023.3333561.

[2] D. Owens, R. E. Khatib, M. Bisheh-Niasar, R. Azarderakhsh, M. M. Kermani, *Efficient and Side-Channel Resistant Ed25519 on ARM Cortex-M4*, IEEE Transactions on Circuits and Systems I: Regular Papers, Vol. 71, No. 6, pp. 2674-2686, 2024, doi: 10.1109/TCSI.2024.3384414.

[3] N. Mishra, S. K. Hafizul Islam, S. Zeadally, *A survey on security and cryptographic perspective of Industrial-Internet-of-Things", Internet of Things*, Vol. 25, 101037, pp. 1-28, 2024, doi.org/10.1016/j.iot.2023.101037.

[4] Z. U. A. Khan, M. Benaissa, *High speed ECC implementation on FPGA over GF(2^m)*, 25th Int. Conf. Field-Program. Logic Appl. (FPL), 1–6, Sep. 2015, doi: 10.1109/FPL.2015.7293951.

[5] K. C. Cinnati Loi, Seok-Bum Ko, *Parallelization of scalable elliptic curve cryptosystem processors in GF (2^m)*, Microprocessors and Microsystems 45, Part A, pp. 10–22, 2016, https://doi.org/10.1016/j.micpro.2016.02.013.

[6] K Sowjanya, M. Dasgupta, S. Ray, *Elliptic curve cryptography-based authentication scheme for Internet of Medical Things*, Journal of Information Security and Applications, Vol. 58, pp 102761, 2021, doi.org/10.1016/j.jisa.2021.10276.

[7] C. A. Lara-Nino, A. Diaz-Perez, M. Morales-Sandoval, *Lightweight elliptic curve cryptography accelerator for internet of things applications*, Ad Hoc Networks, Vol. 103, 102159, 2020, doi.org/10.1016/j.adhoc.2020.102159.

[8] S. M. Shohdy, A. b. El-sisi, N. Ismail, *FPGA Implementation of Elliptic Curve Point Multiplication over GF (2^{191})*, In J.H. Park et al. (Eds.): ISA 2009, LNCS 5576, pp. 619–634, Springer-Verlag Berlin Heidelberg 2009, https://doi.org/10.1007/978-3-642-02617-1_63.

[9] A. P. Fournaris, J. Zafeirakis, O. Koufopavlou, *Designing and evaluating high speed elliptic curve point multipliers*, in Proc. 17th Euromicro Conf. Digit. Syst. Design (DSD), 169–174, 2014, doi: 10.1109/DSD.2014.104.

[10] W. N. Chelton, M. Benaissa, *Fast elliptic curve cryptography on FPGA*, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., Vol. 16, No. 2, pp. 198–205, 2008, doi: 10.1109/TVLSI.2007.912228.

[11] Z. U. A. Khan, M. Benaissa, *High-Speed and Low-Latency ECC Processor Implementation Over GF(2^m) On FPGA*, IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, Vol. 25, No. 1, 165-176, 2017, doi: 10.1109/TVLSI.2016.2574620.

[12] A. A. H. Abd-Elkader, M. Rashdan, El-S. A. M. Hasaneen, H. F. A. Hamed, *Efficient implementation of Montgomery modular multiplier on FPGA*, Computers and Electrical Engineering, Vol. 97, 107585, 2022, doi.org/10.1016/j.compeleceng.2021.107585.

[13] V. Kumar, N. Badal, R. Mishra, *Body sensor networks architecture and security issues in healthcare application*, IOP conference Series: Materials science and Engineering, Vol. 1022, No. 1, 012075, 2021, doi:10.1088/1757-899X/1022/1/012075.

[14] T. K. Dang, C. D. Pham, T. L. P. Nguyen, *A pragmatic elliptic curve cryptography-based extension for energy-efficient device to device communication in smart cities*, Smart Cities and Societies, Vol. 56, 102097, 2020, https://doi.org/10.1016/j.scs.2020.102097.

[15] J. Ye, Z. Yang, *An ECC with error detection and against side channel attacks for resource constrained devices*, Journal of King Saud University-Computer and Information Sciences, Vol. 36, No. 4, 102019, 2024, https://doi.org/10.1016/j.jksuci.2024.102019.

[16] A. E. Adeniyi, R. G. Jimoh, J. B. Awotunde, *A systematic review on elliptic curve cryptography algorithm for internet of things: Categorization, application areas, and security*, Computers and Electrical Engineering, Vol. 118, Part A, 109330, 2024, 10.1016/j.compeleceng.2024.109330.

[17] H. S. Bhagappa, N. Divyashree, N. Avinash, B.N. Manjunatha, J. Vishesh, M. Mamatha, *Enhancing secrecy using hybrid elliptic curve cryptography and Diffie Hellman key exchange approach and Young's double slit experiment optimizer based optimized cross layer in multihop wireless network*, Measurement: Sensors, Vol. 31, 100967, 2024, https://doi.org/10.1016/j.measen.2023.100967.

[18] F. Rodríguez-Henríquez, N. A. Saqib, A. Díaz-Pérez, *A fast parallel Implementation of Elliptic Curve point multiplication over GF(2^m)*, Microprocessors and Microsystems, Vol. 28, No. 5-6, pp. 329–339, 2004, https://doi.org/10.1016/j.micpro.2004.03.003.

[19] D. Panwar, S. S. Dhanda, K. S. Kaswan, P. Singh, S. Kumari, *A 233-Bit Elliptic Curve Processor for IoT Applications*, Lecture Notes in Electrical Engineering Emergent Converging Technologies and Biomedical Systems, Springer Nature Singapore, 61-69, 2024, https://doi.org/10.1007/978-981-99-8646-0_6

[20] C. Rebeiro, S. S. Roy, D. S. Reddy, D. Mukhopadhyay, *Revisiting the Itoh-Tsuji Inversion algorithm for FPGA platform*, IEEE Transactions on VLSI Systems, Vol. 19, No. 8, 1508 - 1512, 2011. 10.1109/TVLSI.2010.2051343

[21] A. P. Fournaris, J. Zafeirakis, O. Koufopavlou, *Designing and evaluating high speed elliptic curve point multipliers*, in Proc. 17th Euromicro Conf. Digit. Syst. Design (DSD), 169–174, 2014, doi: 10.1109/DSD.2014.104.

[22] N. P. Kumar, C. Shirisha, *An area-efficient ECC architecture over GF(2^m) for resource constrained applications*, Int. J. Electronics and Communications (AEU) Vol. 125, pp. 153383, 2020, 10.1016/j.aeue.2020.153383.

[23] C. H. Kim, S. Kwon, C. P. Hong, *FPGA implementation of high-performance elliptic curve cryptographic processor over GF (2^{163})*, Journal of System Architecture, Vol 54, No. 10, pp. 893-900, 2008, 10.1016/j.sysarc.2008.03.005.

[24] Y. Zhang, D. Chen, Y. Choi, L. Chen, S.-B. Ko, *A high performance ECC hardware implementation with instruction-level parallelism over GF (2^{163})*, Microprocessors and Microsystems, Vol 34, pp. 228-236, 2010, https://doi.org/10.1016/j.micpro.2010.04.006.

[25] G. Orlando, C. Paar, *A high performance reconfigurable elliptic curve processor for GF(2^m)*, CHES 2000, Lecture Notes in Computer Science, Vol. 1965, pp. 41-56, 2000, https://doi.org/10.1007/3-540-44499-8_3

[26] C. Grabbe, M. Bednara, J. V. Z. Gathen, J. Shokrollahi, J. Teich, *A high performance vliw processor for finite field arithmetic*, Reconfigurable architecture workshop (RAW), 2003, doi: 10.1109/IPDPS.2003.1213351.

[27] M. Benaissa, W. M. Lim, *Design of flexible GF (2m) elliptic curve cryptography processors*, IEEE transactions on VLSI System, Vol. 14, No. 6, pp. 59-662, 2006, https://doi.ieeecomputersociety.org/10.1109/TVLSI.2006.878235.

[28] R. Azarderaksh, A. Reyani-Masoleh, *Efficient FPGA implementations of point multiplication on binary Edwards and generalized Hessian curves using Gaussian normal basis*, IEEE Transaction on VLSI Systems, Vol. 20, No. 8, 1453-1466, 2012, doi: 10.1109/TVLSI.2011.2158595.

[29] H. Mahdizadeh, M. Masoumi, *Novel architecture for efficient FPGA implementation of elliptic curve cryptographic processor over GF (2^{163})*, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., Vol. 21, No. 12, pp. 2330–2333, 2013, doi: 10.1109/TVLSI.2012.2230410.

[30] R. Salarifard, B. S. Sarmadi, H. M. Boorani, *A low-latency and low-complexity point-multiplication in ECC*, IEEE Transaction on Circuits and Systems-I Regular Papers, Vol. 65, No. 9, 2869-2877, 2018, doi: 10.1109/TCSI.2018.2801118.

[31] G. D. Sutter, J. Deschamps, J. L, Imana, *Efficient elliptic curve point multiplication using digit-serial binary field-operations*, IEEE Transactions on Industrial Electronics Vol. 60, No. 1, 217-225, 2013, 10.1109/TIE.2012.2186104.

[32] M. Imran, M. Rashid and I. Shafi, "Lopez Dahab based elliptic crypto processor (ECP) over GF(2163) for low-area applications on FPGA," *2018 International Conference on Engineering and Emerging Technologies (ICEET)*, Lahore, Pakistan, 2018, pp. 1-6, doi: 10.1109/ICEET1.2018.8338645.

[33] M. Imran, M. Rashid, A, R, Jafri, M. Kashif, *Throughput/area optimized piplined architecture for elliptic curve crypto processor*, IET Computer Digital Tech, Vol. 13, No. 5, 1-8, 2019, https://doi.org/10.1049/iet-cdt.2018.5056.

[34] M. Imran, I. Shafi, A. R. Jafri and M. Rashid, "Hardware design and implementation of ECC based crypto processor for low-area-applications on FPGA," *2017 International Conference on Open Source Systems & Technologies (ICOSST)*, Lahore, Pakistan, 2017, pp. 54-59, doi: 10.1109/ICOSST.2017.8279005.

[35] H. M. Choi, C. P. Hong, C. H. Kim, *High Performance Elliptic Curve Cryptographic Processor over GF(2^{163})*, 4th IEEE Int. Symp. Elect. Design Test and Appl. (DELTA), 290-295, 2008, doi: 10.1109/DELTA.2008.118.

Dr. Sumit Singh Dhanda currently works as associate professor in CSE in IILM University, Greater Noida. He has received B. Tech and M. Tech degrees in ECE from Kurukshetra University, Kurukshetra in 2005 and 2011, respectively. He has completed his PhD from NIT Kurukshetra. He has total experience of 18+ years of teaching, industry and research. The current area of research includes machine learning in the field of hardware security, 6G, and Internet of Medical Things.

Dr. Savita Kumari works as an Assistant Professor in Galgotias University, Greater Noida, India. Savita Kumari received Bachelor and Master Degree in Electronics and Communication Engineering from Maharshi Dayanand University, Haryana and Ph.D. degree from NIT Kurukshetra, India. She has more than 7 years of teaching and research experience. Her research interest is wireless communication, and soft computing.

Dr. Vinod Kumar works as an associate professor in CSE Department of Galgotias University. Also, he works on DST sponsored research project. His area of research interests is IoT, machine learning, cloud computing, and security. He has done Ph.D. in CSE, M. Tech ICT in software engineering and B. Tech in information technology. He has worked in CAIR-DRDO research laboratory and also at many reputed universities.

Dr. Sachin Kumar Gupta currently works as an associate professor in the DoECE (IIST–ISRO), Central University of Jammu, India since 8th September 2023, He has received his B. Tech in Electronics and Telecomm. Engg. from NIT Raipur, India in 2008 and M. Tech & Ph.D. in Systems Engineering from IIT (BHU) Varanasi, India in 2011 & 2016, respectively. He was a former research fellow in the Mobile Computing and Broadband Networking Lab, NCTU, Hsinchu, Taiwan. He has also served as assistant professor in SoECE, SMVDU, Katra, India during 01/01/2015 to 07/09/2023. His research interests include UAV, IoT, Networking and Security issues, etc. He has published more than 120 research articles in reputed international/national journals and prestigious conference proceedings and is also an author and editor of many books.

Received 12 January 2025