Towards High-Performance FPGA Implementation of ECDSA for Koblitz Curve: An Instruction-Set Approach

Phu Nguyen
Department of Electronics, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam | Vietnam National University-Ho Chi Minh City, Ho Chi Minh City, Vietnam
phunguyen200175@hcmut.edu.vn

Hung Nguyen
Department of Electronics, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam | Vietnam National University-Ho Chi Minh City, Ho Chi Minh City, Vietnam
ngthung@hcmut.edu.vn

Kim Anh Phan Vo
Department of Electronics, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam | Vietnam National University-Ho Chi Minh City, Ho Chi Minh City, Vietnam
pvkanh@hcmut.edu.vn

Linh Tran
Department of Electronics, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam | Vietnam National University-Ho Chi Minh City, Ho Chi Minh City, Vietnam
linhtran@hcmut.edu.vn (corresponding author)

Received: 19 March 2025 | Revised: 5 April 2025 | Accepted: 23 April 2025

 DOI: https://doi.org/10.48084/etasr.11040

ABSTRACT

This paper presents a novel instruction-set-based hardware implementation of the Elliptic Curve Digital Signature Algorithm (ECDSA) for a 256-bit Koblitz curve on FPGA. The research contribution under consideration utilizes the integration of Koggle-Stone Adders (KSAs) into the modified structure of modular multiplication and inversion units, thereby enabling high-speed performance in modular computation architecture. Furthermore, by employing an instruction-set-based approach for the control unit instead of the conventional finite state machine for the implementations of ECDSA and point multiplications, we can complete a scalar multiplication operation in less than 2 ms. Our design achieved 110.44 MHz in clock speed on Xilinx Artix-7, occupying 6.4K slices in resource utilization. The modified algorithms employed are constant time, thereby preventing timing attacks. The design is efficient in terms of speed, area, and throughput.

Keywords-elliptic curve digital signature algorithm; Koblitz curve; koggle-stone adders; instruction-set approach

I. INTRODUCTION

Elliptic Curve Cryptography (ECC), a public-key cryptography form based on elliptic curves over finite fields, was proposed by Koblitz [1] and Mille [2] in 1985. The efficiency of ECC, its robust security, and its minor key sizes in comparison to conventional cryptographic schemes such as RSA make it particularly well-suited for resource-constrained applications [3].

Koblitz curves, defined over prime fields, allow for faster and more efficient cryptographic operations in ECC due to their unique properties, making them suitable for secure communications, digital signatures, and key exchange protocols [1, 4]. While software ECC implementations are easier to develop, they are vulnerable to security threats such as side-channel and fault injection attacks [5]. In contrast, hardware implementations provide better performance, lower power consumption, and increased security [6].

Several studies have explored efficient hardware implementations of Koblitz ECC [7]. Authors in [8] address performance and security in their ECC processor design, whereas authors in [9] examine the integration of Koblitz ECC with other cryptographic protocols. Authors in [10] and [11] focus on balancing performance, security, and resource utilization in their ECC designs. Lastly, authors in [12] provide a comprehensive survey of secure ECC implementations and associated countermeasures. Authors in [13] provide a recent survey on techniques to implement ECC point multiplication on hardware.

Our study focuses on the Elliptic Curve Digital Signature Algorithm (ECDSA) rather than other public-key cryptosystems like RSA and DSA [14], because ECDSA offers a shorter signature and better performance for the same security level. We choose the curve secp256k1 because it has the simple formula ($y^2 = x^3 + 7$), which helps minimize the complexity of the calculation, but ensures the equivalent level of security compared to other curves.

Recent studies have further improved the performance of ECC on the prime field across various hardware platforms, using higher-performance modular units as well as new structures that utilize parallel computation or enhancement on point doubling and point-adding modules [15]. Authors in [16] proposed a multi-functional resource-constrained elliptic curve cryptographic processor with low-resource usage and a unified structure, utilized for different curve parameters. Authors in [17] presented their low-hardware architecture on ASIC for various bit lengths. Authors in [18] proposed a similar lightweight architecture for elliptic curve scalar multiplication over similar 256-bit prime field curves. Weierstrass curves are also chosen to be optimized on modulus operation in [19]. Authors in [20] present their research on elliptic curve scalar multiplication on both FPGAs and ASICs.

In our study, we further address the limitations of existing methods and present our solution. The main contributions of this paper include the following aspects:

- The development of an extensively optimized modular operation compute core modified with Koggle-Stone Adders (KSAs).
- The deployment of an instruction-set-based controller instead of the conventional usage of a finite state machine to compute the ECDSA signature on the Koblitz curve secp256k1.

II. FUNDAMENTALS AND METHODS

A. Fundamentals

1) Properties of the Chosen Koblitz Curves

The primary property of Koblitz curves that makes them attractive for ECC is the existence of the Frobenius endomorphism [21]. This endomorphism, denoted by *T*, maps points on the curve to other points on the same curve and enables fast scalar multiplication using a combination of point addition, point doubling, and *T* operations. This property significantly reduces the computational complexity of scalar multiplication compared to general elliptic curves.

The curve chosen for this research, secp256k1, has been widely used in recent years due to its usage in the Bitcoin ECDSA algorithm. Table I shows the parameters of the Koblitz secp256k1 256-bit prime field elliptic curve.

TABLE I. ELLIPTIC CURVE SECP256K1 DOMAIN PARAMETERS

Parameter	Value
p	$2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$
a	0
b	7
xG	0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
yG	0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8
n	0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141
h	0×1

2) Mathematical Operations in Koblitz Elliptic Curve Cryptography

Given two points P and Q on a Koblitz curve, point addition is the operation that finds a third point $R = P + Q$ on the same curve. The point addition formula for Koblitz curves is derived from the curve equation and involves arithmetic operations in the prime field $\mathbb{F}_p$ [4].

For a point P on a Koblitz curve, point doubling is the operation that computes $2P$, another point on the curve. The point-doubling formula for Koblitz curves is more straightforward than for general elliptic curves, requiring fewer arithmetic operations in $\mathbb{F}_p$ [4].

Scalar multiplication, the most critical operation in ECC, computes kP for a point P on the curve and an integer k. The primary method for scalar multiplication on Koblitz curves is based on the T-adic Non-Adjacent Form (TNAF) representation of the scalar k. It uses the Frobenius endomorphism to reduce the required point additions and doublings [21].

The point addition and doubling algorithms are originally calculated in the affine coordinates (x, y). Let $P(x_p, y_p)$, $Q(x_p, y_p)$ be two points located on the curve secp256k1. $R(x_r, y_r) = P + Q$ and y_r are computed as follows:

$$\begin{cases} x_r = (s^2 - x_q - x_p) \bmod p \\ y_r = \left(-y_p + s(x_p - x_r)\right) \bmod p \end{cases} \quad (1)$$

where $s = (y_q - y_p)(x_q - x_p)^{-1} \bmod p$.

In the case of point doubling computation, let $P(x_p, y_p)$ and $R(x_r, y_r) = 2P$. The coordinates of R are calculated as follows:

$$\begin{cases} x_r = (s^2 - 2x_p) \bmod p \\ y_r = \left(-y_p + s(x_p - x_r)\right) \bmod p \end{cases} \quad (2)$$

where $s = (3x_p)^2 (2y_p)^{-1} \bmod p$.

With the point addition and doubling algorithms, to avoid the use of inversion during computation, we transform points in affine coordinates (x, y) to Jacobian coordinates $(X:Y:Z)$ where $x = \frac{X}{Z^2}$; $y = \frac{X}{Z^3}$.

Let $P(X_1:Y_1:Z_1), Q(X_2:Y_2:Z_2)$ where $Q \neq \pm P$ and $Q \neq O$. Here, O represents the point at infinity $(X:Y:0)$.

We have $R(X_3:Y_3:Z_3) = P + Q$, where X_3, Y_3, and Z_3 are computed as follows:

$$\begin{cases} X_3 = R^2 + G - 2 * V \\ Y_3 = R * (V - X_3) - S_1 * G \\ Z_3 = Z_1 * Z_2 * H \end{cases} \quad (3)$$

where $R = S_1 - S_2$, $G = H^3$, $V = U_1 H^2$, $S_1 = Y_1 Z_2^3$, $S_2 = Y_2 Z_1^3$, $H = U_1 - U_2$, ; $U_1 = X_1 Z_2^2$, and $U_2 = X_2 Z_1^2$.

For point doubling operations, given a point $P(X_1:Y_1:Z_1)$, we need to compute $R(X_3:Y_3:Z_3) = 2P$, which is performed as follows:

$$\begin{cases} X_3 = M^2 - 2S \\ Y_3 = M(S - X_3) - 8T \\ Z_3 = 2Y_1 Z_1 \end{cases} \quad (4)$$

where $M = 3X_1^2 + a_4 Z_1$, $T = Y_1^4$, and $S = 4X_1 Y_1^2$.

3) Montgomery Ladder Algorithm

With Co-Z coordination, we use the Montgomery ladder algorithm for efficient computation of the multiplication of a point by a scalar to implement in our hardware controller [22]. Algorithm 1 shows the Montgomery ladder algorithm utilized in this paper.

```
Algorithm 1: Montgomery ladder algorithm
Input: P ∈ E(F_p), k = (1, k_{n-2}, ..., k_1, k_0)
Output: Q = k · P
Set R_0 ← P
Set R_1 ← 2P
For i ← (n − 2) down to 0 do
   If k_i = 1 then
      R_0 ← R_0 + R_1
      R_1 ← 2R_1
   Else
      R_1 ← R_0 + R_1
      R_0 ← 2R_0
Return Q = R_0
```

B. Design the Hardware Implementation

1) Building the Instruction Set and Memory

Based on the ECDSA and Montgomery ladder algorithms, we propose an instruction set suitable for computing the point-adding, doubling, and digital signature generation steps. Figure 1 shows the visualized ECDSA algorithm. The instruction set aims to provide an efficient method for data exchange between the ECDSA compute core and the Data Memory (DMEM) memory that holds the primary data.

Table II shows a list of custom instructions developed for our instruction-set-based ECDSA design. Each instruction corresponds to a particular mathematical operation or function in the modular compute core. These specialized instructions are essential to the efficient and accurate execution of our ECDSA core. Additionally, we use a set of four predefined constants for the Koblitz curve K-256.

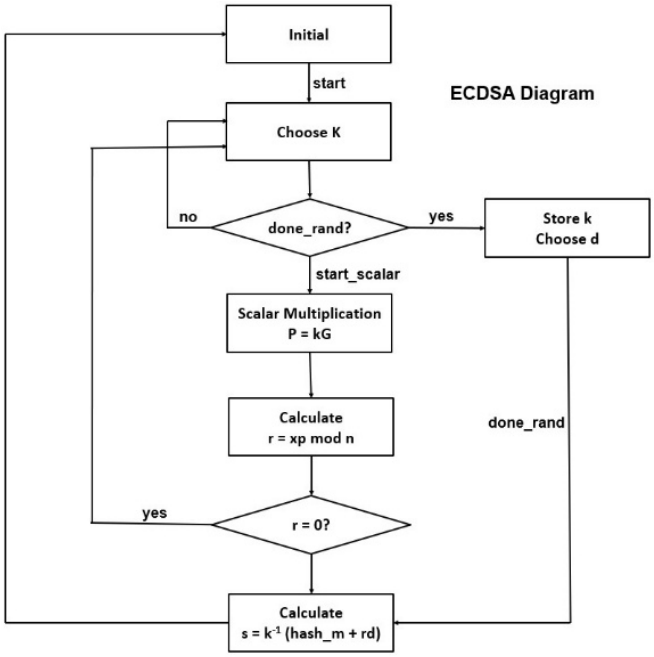

Fig. 1. Visualized ECDSA algorithm.

TABLE II. INSTRUCTION DETAILS

Instruction	Description
ADDP	(*a* + *b*) mod *p*
ADDN	(*a* + *b*) mod *n*
SUBP	(*a* - *b*) mod *p*
MULP	(*a* * *b*) mod *p*
MULN	(*a* * *b*) mod *n*
MUL_CNST	(const * *a*) mod *p*
MUL_PRV_KEY	(*r* * *d*) mod *n*
INVP	*r*^-1 mod *p*
INVN	*k*^-1 mod *n*
LOAD	Store point *G* coordinates in memory
MULP_R_SQR	(*a* * *n1*) mod *p*
MULN_R_SQR	(*a* * *n2*) mod *n*
MUL_PRE_INVP	(*a* * *n3*) mod *p*
MUL_PRE_INVN	(*a* * *n4*) mod *n*

These instructions were created because we want to handle complex operations (Montgomery ladder algorithm) with simple modulo calculations. The instructions are 19-bit in length, where bits [18:15] contain the description for the opcode part, [14:10] is the address of the destination register to write data into DMEM, and [9:5] and [4:0] are the addresses for the two source registers, containing the data used for the computation in the modular compute core. The instructions are shorter in bit length because we omit the elliptic curve's bit length in the instruction set. Furthermore, this opens the

scalability of this instruction set, as curves with longer bit lengths could easily be reused in the same architecture with minor adjustments in the memories and the modular compute core.

2) General Architecture

Figure 2 illustrates the architecture of our design. We build the general architecture of the hardware design based on the proposed instruction set, making the development of the control unit's Finite State Machines (FSMs) more efficient.

The custom instructions for the top module are stored in the FPGA-based Instruction Memory (IMEM). It has 2^{10} addresses and a 19-bit data length for the instructions. The register file (REGFILE) is a dual-read, single-write memory with 256-bit data length and 2^5 address depth.

Fundamentally, d and k are values that come from the random generator, and $hash_m$ is the hashed message. $data_{in}$ is the coordinate of the curve's *G* point (*xG*, *yG*, *zG*). This *G* point is represented in the Jacobian coordinate. After that, we start computing the digital signature (r, s). This step contains two phases: scalar multiplication and signature computation. We designed our instructions for the scalar multiplication phase based on the Montgomery ladder theory, which can compute the point multiplication within a fixed amount of time. The signature computation phase concludes with the calculations for generating the digital signature (r, s) of the ECDSA operation. After the calculation, the digital signature (r, s) will appear at the output (r, s) attached to the REGFILE.

Figure 2 can be divided into two parts: CONTROL UNIT and DATA PATH. The CONTROL UNIT is responsible for detecting the instructions that control the operation of the data path. In the DATA PATH, the PC acts as a pointer to the instructions that must be read from the IMEM. IMEM and DMEM store the instructions and the calculated data. Our architecture also has an Imm_gen block for extending the constant value stored in the instruction, because, during our calculation, there are cases in which we need to compute our data with a constant number. The modular compute core calculates modular addition, modular multiplication, and modular inversion.

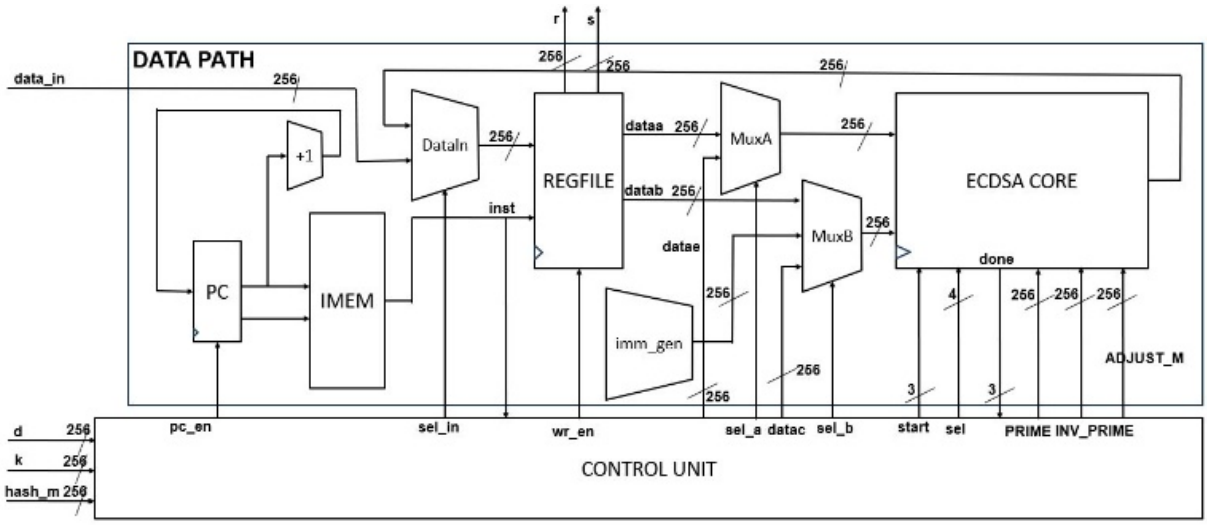

Fig. 2. General architecture of the hardware design.

3) Modular Compute Core

The modular operation core consists of three sub-modules: modular addition, modular multiplication, and modular inversion. These modules are designed using constant-time algorithms to protect against specific side-channel attacks.

As illustrated in Figure 3, the ECDSA core structure includes inputs a_i and b_i for computation, along with PRIME (the prime number of the curve) and INV PRIME (the inversion of PRIME).

For modular multiplication, we utilize the radix-2 Montgomery modular multiplication, proposed by authors in [23], but modified with KSAs for high performance. It completes the multiplication result after 256 loops. Figure 4 presents the modified modular multiplication structure.

Figure 5 shows the modified modular inversion structure. We design the modular inversion block based on the constant-time binary extended Euclidean algorithm proposed by authors in [24], modified with KSAs and optimized controllers. The INV CONTROL block is an FSM used to control the operations of the DENTA UV, DENTA RS, and SIGMA blocks. The algorithm computes the modular inversion result within 512 iterations, with the calculation time depending on the length of our prime value. For the secp256k1 prime data of 256 used in this paper, we need to iterate 512 times.

For this modular inversion module, we design the dual-purpose modular addition and subtraction unit based on the high-radix parallel prefix network modular adder/subtractor proposed by authors [25]. This Koggle–Stone parallel prefix network adder/subtractor is appropriate for optimizing operating frequency and pipelining. Figure 6 provides the dual-purpose structure packed with a micro finite state machine.

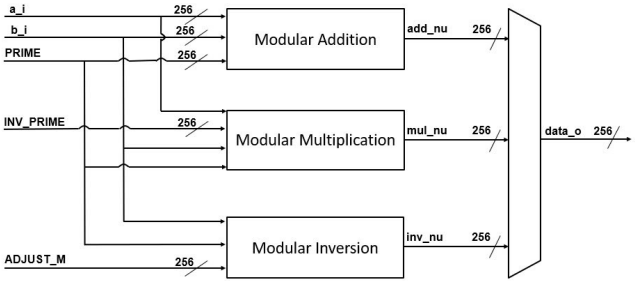

Fig. 3. Modular compute core hardware diagram.

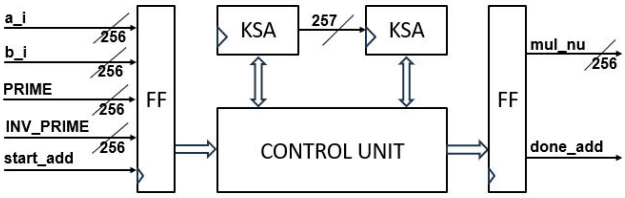

Fig. 4. Modified modular multiplication structure.

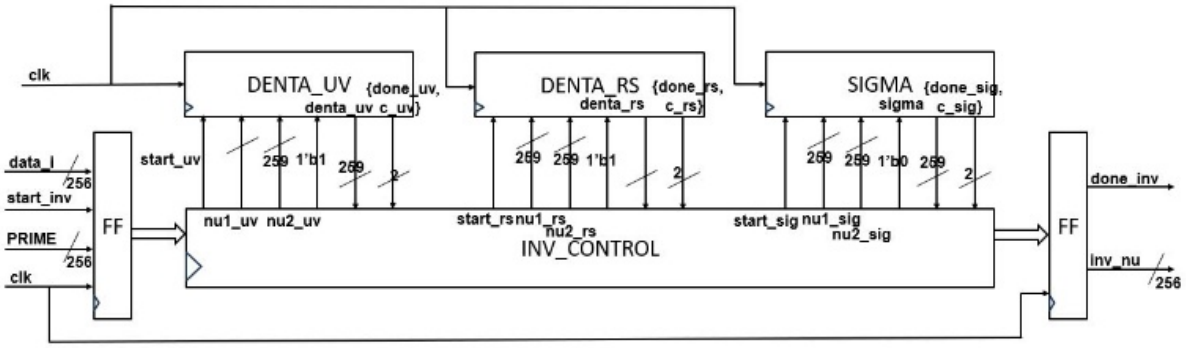

Fig. 5. Modified modular inversion structure.

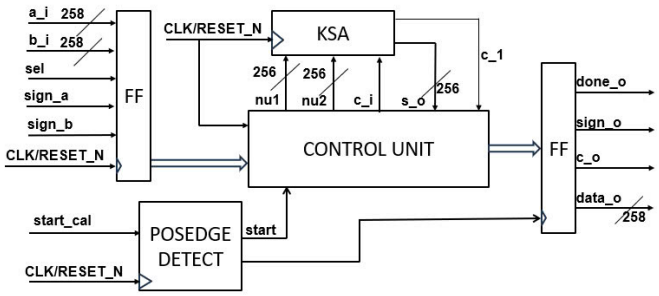

Fig. 6. The dual purpose modular addition and subtraction unit.

III. RESULTS

We use Intel Questasim to simulate our design. We verified our result with a well-known standard Python software implementation. The implementation uses AMD Vivado on the Artix-7 FPGA platform. The design occupies 6471 slices, 23187 Look-Up Tables (LUTs), 12878 Flip-Flops (FF), and 8 Block Memories (BRAMs). Our design achieved a maximal clock frequency of 110.44 MHz. The point-adding and doubling operations require 95359 and 67630 clock cycles, respectively, translating to 0.871 ms and 0.615 ms. A single-point multiplication operation takes 1.48 ms. Generating an elliptic curve digital signature on the Koblitz curve takes 5.92 ms, producing the signature pair (r, s). The throughput of our 256-bit prime field ECC implementation is calculated using the formula of (5):

$$Throughput\ (kbps) = \frac{bit{-}length\ (bits)}{Point\ Multiplication\ op\ (ms)} \quad (5)$$

IV. DISCUSSION

This section provides a comparison of our FPGA-based ECDSA implementation on the secp256k1 Koblitz curve with recent ECC hardware designs. The analysis encompasses throughput, execution time, clock frequency, resource utilization, and the area-time product (AxT), using data from Table III and visuals from Figures 7 and 8. Our design, built on the Xilinx Artix-7 FPGA, runs at 110.44 MHz, completes a single-point multiplication in 1.48 ms, and achieves a throughput of 173 kbps. It uses 6,471 slices and has an AxT of 9, the lowest among the works compared, showing a strong mix of speed and efficiency.

TABLE III. PERFORMANCE ANALYSIS FOR OUR PROPOSED HARDWARE IMPLEMENTATION

Reference	Slices	Frequency (MHz)	Time (ms)	Throughput (kbps)	AxT
This work	6471	110.4	1.48	173	9
[15] 2019,V7	8873	177.7	1.48	173	13
[15] 2019, V6	9246	161.1	1.63	157	15
[18] 2022, K7	6477	156.3	1.73	147.9	11
[18] 2022, V7	6397	158.7	1.7	150.6	10
[18] 2022, Z	6351	149.7	1.8	142.1	11
[20] 2020, K7	7400	122.8	2.44	104.9	18
[20] 2020, V7	5500	122.8	2.44	104.9	13
[20] 2020, V6	6600	105.9	2.83	90.45	18

a. [15], 2019, V6, V7 indicates the design implemented on Virtex-6 and Virtex-7, respectively.

b. [18], 2022, Z, V7, K7 denotes the design implemented on Zynq, Virtex-7, and Kintex-7, respectively.

c. [20], 2020, V6, V7, K7 refers to the design implemented on Virtex-6, Virtex-7, and Kintex-7, respectively.

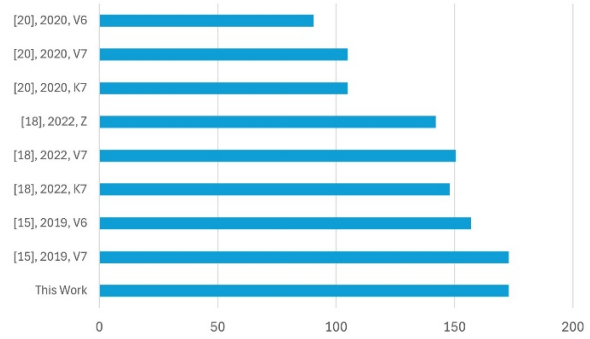

Fig. 7. Performance comparison of throughput between implementations (higher is better).

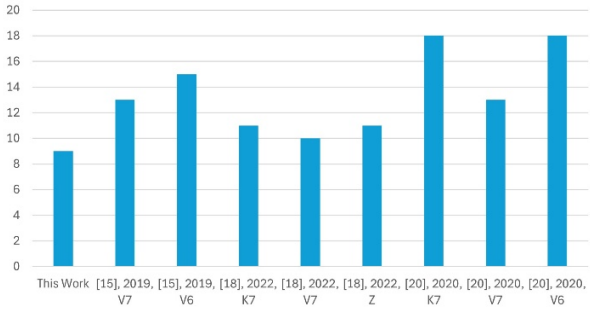

Fig. 8. Performance comparison on the AxT product between implementations (lower is better).

As illustrated in Table III, authors in [15] achieve throughput levels comparable to ours in certain scenarios on Virtex-7 (177.7 MHz, 1.48 ms, 173 kbps, 8,873 slices, AxT 13) and Virtex-6 (161.1 MHz, 1.63 ms, 157 kbps, 9,246 slices, AxT 15). However, they utilize more resources and operate at higher frequencies. Authors in [18] report 142.1–150.6 kbps, 1.7–1.8 ms, and AxT 10–11 on Kintex-7, Virtex-7, and Zynq (6,351–6,477 slices), whereas authors in [20] report 90.45–104.9 kbps, 2.44–2.83 ms, and AxT 13–18 (5,500–7,400 slices). Figure 7 shows our 173 kbps as the top value, equal to the study in [15] on Virtex-7, but beating their 157 kbps on Virtex-6, and clearly ahead of the study in [18] with a best of 150.6 kbps and the study in [20] with a highest of 104.9 kbps. The top bar in Figure 7 is due to our instruction-set control unit and KSAs that improve modular operations. Figure 8 shows our AxT of 9 as the shortest bar, implying better efficiency than the 10–11 AxT of [18], the 13–15 AxT of [15], and the 13–18 AxT of [20], where higher bars imply worse tradeoffs.

Our resource usage is in the middle with 6,471 slices – more than the range of the study in [18], but less than the ranges of the studies in [15] and [20]. This shows that we trade some space for better performance with KSAs and custom instructions, supported by 8 BRAMs and 23,187 LUTs. Compared to the study in [16], our 5.92 ms signature time beats their 10 ms, even though their design works with more curves. Compared to ASIC designs such as in the study in [17], our FPGA uses more resources but offers more flexibility.

Our strengths are high throughput and speed (top bar in Figure 7), good efficiency (shortest bar in Figure 8), and security (constant-time algorithms to prevent timing attacks). Weaknesses include average resource usage, focus on a single curve, and a lower clock speed, indicating areas for improvement. Differences in platforms and curves (e.g., secp256k1 vs. Ed25519 or NIST P-256) affect comparisons, but our Artix-7 setup proves cost-effective for embedded systems.

V. CONCLUSION

This paper introduces a high-performance FPGA implementation of the Elliptic Curve Digital Signature Algorithm (ECDSA) utilizing the Montgomery ladder technique, achieving a throughput of 173 kbps and an area–time product (AxT) of 9, which reflects superior efficiency in terms of speed and hardware utilization compared to existing works. The novelty of the design lies in its low-complexity, pipeline-friendly scalar multiplication architecture, which enhances parallelism while optimizing the balance between area and delay, rendering it particularly suitable for embedded and resource-constrained environments. Future research will aim to extend this design to accommodate multiple elliptic curves, bolster resistance against side-channel attacks, and investigate adaptive reconfiguration to meet dynamic security requirements, alongside integrating the implementation into end-to-end secure systems and assessing its efficacy within real-world cryptographic protocols.

ACKNOWLEDGMENT

We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

REFERENCES

[1] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987, https://doi.org/10.1090/S0025-5718-1987-0866109-5.

[2] V. S. Miller, "Use of Elliptic Curves in Cryptography," in *Advances in Cryptology — CRYPTO '85 Proceedings*, Santa Barbara, CA, USA, 1985, pp. 417–426, https://doi.org/10.1007/3-540-39799-X_31.

[3] H. C. A. Tilborg and S. Jajodia, *Encyclopedia of Cryptography and Security*, 2nd ed. New York, NY, USA: Springer, 2011, https://doi.org/10.1007/978-1-4419-5906-5.

[4] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 1996.

[5] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, 1st ed. New York, NY, USA: Springer, 2007, https://doi.org/10.1007/978-0-387-38162-6.

[6] L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, Eds., *Fault Diagnosis and Tolerance in Cryptography: Third International Workshop, FDTC 2006, Yokohama, Japan, October 10, 2006. Proceedings*, 1st ed. Berlin, Heidelberg, Germany: Springer, 2006, https://doi.org/10.1007/11889700.

[7] K. Jarvinen and J. Skytta, "On Parallelization of High-Speed Processors for Elliptic Curve Cryptography," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 9, pp. 1162–1175, Sep. 2008, https://doi.org/10.1109/TVLSI.2008.2000728.

[8] H. M. Choi, C. P. Hong, and C. H. Kim, "High Performance Elliptic Curve Cryptographic Processor Over GF(2^163)," in *4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)*, Hong Kong, China, 2008, pp. 290–295, https://doi.org/10.1109/DELTA.2008.118.

[9] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of Data Encryption Standard," in

2004 International Conference on Test, Charlotte, NC, USA, 2004, pp. 339–344, https://doi.org/10.1109/TEST.2004.1386969.

[10] R. Azarderakhsh, K. U. Järvinen, and M. Mozaffari-Kermani, "Efficient algorithm and architecture for elliptic curve cryptography for extremely constrained secure applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 4, pp. 1144–1155, Apr. 2014, https://doi.org/10.1109/TCSI.2013.2283691.

[11] T. Oliveira, J. López, and F. Rodríguez-Henríquez, "Software Implementation of Koblitz Curves over Quadratic Fields," in *18th International Conference on Cryptographic Hardware and Embedded Systems*, Santa Barbara, CA, USA, 2016, pp. 259–279, https://doi.org/10.1007/978-3-662-53140-2_13.

[12] J. Fan, X. Guo, E. De Mulder, P. Schaumont, B. Preneel, and I. Verbauwhede, "State-of-the-art of secure ECC implementations: a survey on known side-channel attacks and countermeasures," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust*, Anaheim, CA, USA, 2010, pp. 76–87, https://doi.org/10.1109/HST.2010.5513110.

[13] A. Verri Lucca, G. A. Mariano Sborz, V. R. Q. Leithardt, M. Beko, C. Albenes Zeferino, and W. D. Parreira, "A Review of Techniques for Implementing Elliptic Curve Point Multiplication on Hardware," *Journal of Sensor and Actuator Networks*, vol. 10, no. 1, Mar. 2021, Art. no. 3, https://doi.org/10.3390/jsan10010003.

[14] E. S. I. Harba, "Secure Data Encryption Through a Combination of AES, RSA and HMAC," *Engineering, Technology & Applied Science Research*, vol. 7, no. 4, pp. 1781–1785, Aug. 2017, https://doi.org/10.48084/etasr.1272.

[15] Md. M. Islam, Md. S. Hossain, Moh. K. Hasan, Md. Shahjalal, and Y. M. Jang, "FPGA Implementation of High-Speed Area-Efficient Processor for Elliptic Curve Point Multiplication Over Prime Field," *IEEE Access*, vol. 7, pp. 178811–178826, 2019, https://doi.org/10.1109/ACCESS.2019.2958491.

[16] B. K. Do-Nguyen, C. Pham-Quoc, N.-T. Tran, C.-K. Pham, and T.-T. Hoang, "Multi-Functional Resource-Constrained Elliptic Curve Cryptographic Processor," *IEEE Access*, vol. 11, pp. 4879–4894, 2023, https://doi.org/10.1109/ACCESS.2023.3236406.

[17] X. Hu, X. Zheng, S. Zhang, S. Cai, and X. Xiong, "A Low Hardware Consumption Elliptic Curve Cryptographic Architecture over GF(p) in Embedded Application," *Electronics*, vol. 7, no. 7, Jul. 2018, Art. no. 104, https://doi.org/10.3390/electronics7070104.

[18] Y. Hao *et al.*, "Lightweight Architecture for Elliptic Curve Scalar Multiplication over Prime Field," *Electronics*, vol. 11, no. 14, Jul. 2022, Art. no. 2234, https://doi.org/10.3390/electronics11142234.

[19] M. A. Javed, E. Ben Hamida, and W. Znaidi, "Security in Intelligent Transport Systems for Smart Cities: From Theory to Practice," *Sensors*, vol. 16, no. 6, Jun. 2016, Art. no. 879, https://doi.org/10.3390/s16060879.

[20] T. Kudithi and S. R, "An efficient hardware implementation of the elliptic curve cryptographic processor over prime field," *International Journal of Circuit Theory and Applications*, vol. 48, no. 8, pp. 1256–1273, Mar. 2020, https://doi.org/10.1002/cta.2759.

[21] T. Kudithi and J. A. Solinas, "Generalized Mersenne Numbers," Center for Applied Cryptographic Research, University of Waterloo, Technical report CORR-99-39, 1999. [Online]. Available: https://cacr.uwaterloo.ca/techreports/1999/corr99-39.pdf.

[22] P. L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," *Mathematics of Computation*, vol. 48, no. 177, pp. 243–264, 1987, https://doi.org/10.1090/S0025-5718-1987-0866113-7.

[23] H. Xiao, S. Yu, B. Cheng, and G. Liu, "FPGA-based high-throughput Montgomery modular multipliers for RSA cryptosystems," *IEICE Electronics Express*, vol. 19, no. 9, pp. 20220101–20220101, 2022, https://doi.org/10.1587/elex.19.20220101.

[24] E. Savaş and Ç. K. Koç, "Montgomery inversion," *Journal of Cryptographic Engineering*, vol. 8, no. 3, pp. 201–210, Sep. 2018, https://doi.org/10.1007/s13389-017-0161-x.

[25] M. Rogawski, E. Homsirikamol, and K. Gaj, "A novel modular adder for one thousand bits and more using fast carry chains of modern FPGAs," in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, Munich, Germany, 2014, pp. 1–8, https://doi.org/10.1109/FPL.2014.6927493.

AUTHORS PROFILE

Phu Nguyen is a master student who also received his B.S. degree in Electronics and Telecommunications Engineering from Ho Chi Minh City University of Technology (HCMUT), VNU HCM, in 2022. His research interests include hardware design, cryptography, and computer architecture.

Hung Nguyen received the B.S. and M.S. degrees in Electronics and Telecommunications Engineering from Ho Chi Minh City University of Technology (HCMUT), VNU-HCM (2019, 2022). He is currently a lecturer at the Faculty of Electrical-Electronics Engineering, Ho Chi Minh City University of Technology (HCMUT), VNU-HCM. His research interests include hardware-software co-design, efficient algorithms, hardware design, and cryptography.

Kim-Anh Phan Vo received the B.S. and M.S. degrees in Electronics and Telecommunications Engineering from Ho Chi MinhCityUniversity of Technology (HCMUT),VNU-HCM (2002, 2012). She is currently a lecturer at the Faculty of Electrical-Electronics Engineering, Ho Chi Minh City Uni versity of Technology (HCMUT), VNU-HCM. Her research interests include memristor logic synthesis, memristor circuit, and analog IC design.

Linh Tran received the B.S. degree in Electrical and Computer Engineering from the University of Illinois, Urbana – Champaign (2005), M.S. and Ph.D. degrees in Computer Engineering from Portland State University (2006, 2015). Currently, he is working as a lecturer at the Faculty of Electrical-Electronics Engineering, Ho Chi Minh City University of Technology – VNU HCM. His research interests include quantum/reversible logic synthesis, computer architecture, hardware-software co-design, efficient algorithms and hardware design targeting FPGAs, and deep learning.