

Power-Efficient FPGA-Based Elliptic Curve Cryptography: A Comparative Study of Multiplier Architectures

Dr.K.Muralidharan,

Assistant Professor,
Department of ECE,

Coimbatore Institute of Technology, Coimbatore, India,

Dr.G.Christina,

Professor,
Dept.of EETE,

St.Joseph University in Tanzania, Tanzania

Dr.T.Yathavi.,

Assistant Professor,
Department of ECE,

Coimbatore Institute of Technology, Coimbatore, India,

Dr.K.Harini

Assistant Professor,
Department of ECE,

Coimbatore Institute of Technology, Coimbatore, India,

Abstract—Elliptic Curve Cryptography (ECC) is widely adopted in resource-constrained and mobile environments because it provides RSA-equivalent security at substantially smaller key sizes, reducing computation, memory, and power overhead. Since scalar point multiplication is the dominant cost in ECC and is itself built on modular multiplication, the choice of multiplier architecture directly determines the energy efficiency of an ECC hardware core. This paper investigates the effect of four multiplier architectures — Montgomery, Vedic, Baugh-Wooley, and Booth — on the power consumption of an FPGA-based ECC implementation. Each multiplier is first characterized in isolation, and the Montgomery multiplier is then integrated into a complete ECC point-multiplication datapath and implemented on [FPGA device/family] using Xilinx Vivado. Simulation results show that the Montgomery-based design consumes 338.739 W total on-chip power, while achieving correct encryption/decryption behaviour verified against a representative test input. Integration challenges encountered with the Vedic, Baugh-Wooley, and Booth multipliers within the ECC datapath are also reported, along with the standalone power profile of each. These results identify Montgomery multiplication as the most power-efficient choice for lightweight ECC hardware among the architectures evaluated and outline the remaining integration work needed for a complete four-way comparison.

Keywords—*Elliptic Curve Cryptography (ECC), Field Programmable Gate Array (FPGA), Montgomery multiplier, Vedic multiplier, Baugh-Wooley multiplier, Booth multiplier, power consumption, public key, private key.*

I. INTRODUCTION

Elliptic Curve Cryptography (ECC) is a public-key cryptosystem built on the algebraic structure of elliptic curves defined over finite fields. Its security rests on the Elliptic Curve Discrete Logarithm Problem (ECDLP), which has no known sub-exponential solution, allowing ECC to achieve security equivalent to RSA at a fraction of the key size. This property makes ECC well suited to resource-constrained environments such as embedded systems and IoT devices, where key length directly determines memory, bandwidth, and computation cost. At the arithmetic level, ECC point operations reduce to repeated modular multiplications, so the efficiency of the underlying multiplier — for example, Montgomery Modular Multiplication (MMM), which chooses the reduction constant R as a power of two to avoid explicit division — has a direct and measurable impact on the power and area of the resulting hardware.

ECC extends its influence across encryption, digital signature generation, and secure key exchange. One of its most pronounced strengths lies in the swift generation of cryptographic keys, enabling rapid setup of secure communication channels, and it excels in encryption and decryption for applications where computational resources are at a premium. ECC's impact reverberates throughout secure communication protocols, epitomized by its role in HTTPS, email encryption, and digital signature processes, and its standardization by NIST underscores its adoption at the highest levels of data security.

The modular multiplication operation is the most time-consuming operation for number-theoretic cryptographic algorithms involving large integers. In RSA — the first public-key encryption and digital-signature algorithm, in use since 1978 — implementations reveal that more than 75 percent of the running time is spent in modular multiplication for moduli larger than 1,024 bits. Such designs are large in area and comparatively inefficient: the very first RSA implementations in the early 1980s assumed a 512-bit modulus would be sufficient, but within a decade advances in factorization pushed this to 1,024 bits, and by the 2010s even that was questioned. Many implementations now use a 2,048-bit modulus, while NIST recommends 3,072- or 4,096-bit moduli for the near future, and larger key sizes translate directly into longer processing time and higher hardware cost. Because RSA's modular exponentiation is itself computed via repeated modular multiplications, the performance of the underlying multiplier has a direct impact on RSA efficiency — motivating the search, in this paper, for the most power-efficient multiplier for the ECC analogue of this problem.

This paper's objective is therefore to reduce the power consumption associated with ECC operations by replacing the conventional multiplier with, and comparing power consumption across, the Montgomery, Vedic, Booth, and Baugh-Wooley multipliers.

II. RELATED WORK AND BACKGROUND

Several recent works address low-power ECC hardware from different angles. Lee et al. [1] proposed an ECC processor using algorithm-architecture co-implementation techniques, reporting a detailed area-delay-product comparison against competing Montgomery point-multiplication (PM) designs. Li et al. [3] introduced a dual-

field PM architecture completing point multiplication in six clock cycles by maximizing Karatsuba multiplier utilization and reordering the PM schedule, while Coliban [5] proposed a radix-2 Montgomery multiplier that replaces the conventional pair of adders with a single three-input (ternary) adder to reduce critical-path delay. Fotovvat et al. [4] examined ECC as a faster and more secure alternative to RSA across the full range of asymmetric primitives, including digital signatures and key agreement, motivated by the growing throughput demands of high-performance authentication servers. More recently, Abirami et al. [12] optimized Montgomery modular multiplication for FPGA using the Karatsuba algorithm, reporting reduced delay and resource usage relative to a direct-method baseline, and Awaludin et al. [15] applied a related Karatsuba-based digit-multiplier variant to a Curve448 ECC processor, reporting throughput gains over prior Curve448 hardware. Ifrim et al. [14], in a broader systematic review of FPGA- and GPU-accelerated ECC for blockchain applications, likewise find Montgomery-based point multiplication to be the dominant architectural choice across the surveyed literature, corroborating its prevalence in the ECC-specific works above.

On the multiplier-comparison side, Leelavathi et al. [11] is, to the authors' knowledge, the closest prior work to the present study: it implements a $GF(2^m)$ ECC processor that combines Montgomery multiplication with Vedic and Booth-encoded multipliers for the underlying finite-field arithmetic, reporting delay, LUT utilization, and static/dynamic power for a wireless-sensor-network target. That work, however, builds a single combined datapath rather than isolating and comparing each multiplier's standalone power profile before integration — the gap the present paper targets directly. Separately, Kalaiselvi and Sabeenian [13] combined Booth encoding with the Vedic (Urdhva-Tiryakbhyam) multiplication sutra to reduce area and improve the area-delay product on both FPGA and 45 nm ASIC, confirming that hybrid Booth-Vedic techniques remain competitive for low-power, low-area multiplication outside the ECC context specifically.

Beyond ECC-specific designs, a separate body of work has optimized the Montgomery multiplier itself as a general-purpose arithmetic block: Abd-Elkader et al. [16] proposed an FPGA-optimized Montgomery multiplier design targeting reduced latency and resource usage, Wu et al. [17] improved high-radix $GF(p)$ Montgomery reduction through deeper reuse of the underlying multiplier hardware, and Kuang et al. [18] presented a scalable radix-4 Montgomery architecture adaptable to varying operand widths. These general-purpose optimizations are complementary to the ECC-level integration studied here and suggest directions for further reducing the 21.436 W standalone figure measured for the Montgomery multiplier in Section V.

A number of other FPGA-based ECC processors report complementary design points without directly comparing multiplier architectures. Saju and S. K. N. [7] designed a Verilog-HDL ECC processor intended to be adaptable across curve parameters and FPGA families. Al-Khaleel et al. [8] implemented an ECC processor over Edwards curves combined with a DFT-based modular multiplication scheme,

illustrating that the choice of curve form is a further design axis alongside the multiplier architecture studied here. Wang et al. [9] targeted IoT applications directly, reporting a low-delay, high-efficiency ECC circuit and algorithm co-design, while Sivajyothi and D. T. [10] combined ECC with AES in a hybrid scheme aimed at reducing encryption time for cloud data protection. More broadly, Sagar and Chourasia [2] proposed a lightweight-cipher VLSI architecture for high-speed FPGA applications, reflecting the same area-power motivation that drives the multiplier choices compared in this paper.

Together, these studies establish Montgomery point multiplication as a recurring architectural choice for efficient ECC hardware and confirm that Vedic- and Booth-based techniques are independently competitive for low-power, low-area multiplication. However, with the partial exception of [11], none of them systematically compares Montgomery multiplication against non-Montgomery multiplier architectures (Vedic, Baugh-Wooley, Booth) under a common ECC datapath on a single FPGA toolchain — the gap this paper addresses.

III. ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic Curve Cryptography (ECC) is a public-key cryptosystem that relies on the mathematical properties of elliptic curves over finite fields to secure communications and data. A pair of keys — a public key and a private key — is used for encryption and decryption, and the security of ECC rests on the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is computationally infeasible to solve. ECC offers strong security with relatively short key lengths: a 256-bit ECC key is often considered on par with a 3,072-bit RSA key, and equivalent RSA security otherwise requires keys of 1,024 to 4,096 bits. This makes ECC well suited to resource-constrained environments such as mobile devices and the IoT, where reduced key size translates into faster key operations, lower bandwidth and storage demands, and enhanced forward secrecy. ECC is used in secure web browsing (HTTPS), email encryption, digital signatures, and related applications where data security and privacy are critical.

A. Equations for ECC

Equation (1) represents the general elliptic curve, where X, Y are points on the curve and a, b are the constants that define it.

$$Y^2 = X^3 + aX + b \quad (1)$$

Let $P(x_1, y_1)$ and $Q(x_2, y_2)$ be points on the curve, and let $R(x_3, y_3) = P + Q$:

$$R(x_3, y_3) = P + Q \quad (2)$$

$$x_3 = \lambda^2 - a - x_1 - x_2 \quad (3)$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \quad (4)$$

where $\lambda = (y_2 - y_1)/(x_2 - x_1)$ when $P \neq Q$, and, for the point doubling $P = Q$:

$$\lambda = (3x_1^2 + 2ax_1 + 1)/(b \cdot y_1) \quad (5)$$

with the doubled point $P(x_1, y_1) \rightarrow (x_4, y_4)$ given by:

$$x_4 = \lambda^2 - a - 2x_1 \quad (6)$$

$$y_4 = \lambda(x_1 - x_4) - y_1 \quad (7)$$

B. Elliptic Curve Cryptography Algorithm

Fig. 1 shows the process involved in the ECC algorithm. ECC does not have a physical architecture like a computer or a network; it is a mathematical and algorithmic framework used for cryptographic purposes, based on sequential steps of key generation, encryption, and decryption.

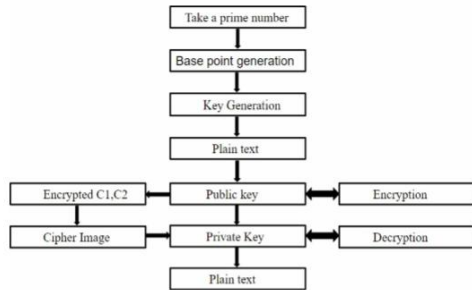


Fig. 1. Process involved in ECC.

Key generation follows $Q = d \cdot G$, where G is the base point on the curve, d is the private key (a randomly chosen integer within a predefined range), and Q is the resulting public key, obtained by point-multiplying G by d .

For encryption, the sender chooses a random integer k and, using the recipient's public key Q , computes the ciphertext pair $(C1, C2)$:

$$C1 = k \cdot G \quad (8)$$

$$C2 = P + (k \cdot Q) \quad (9)$$

where P is the point representation of the plaintext message. To decrypt, the recipient uses the private key d and the ciphertext pair $(C1, C2)$:

$$S = d \cdot C1 \quad (10)$$

$$P = C2 - S \quad (11)$$

Because S computed from $d \cdot C1$ equals $k \cdot Q$ (the same shared secret used at encryption time), the recipient recovers P and, from it, the original message.

IV. MULTIPLIER ARCHITECTURES FOR THE ECC DATAPATH

Multipliers are the fundamental arithmetic components used to compute the product of two operands, and in ECC hardware they are integral to scalar point multiplication: the private key (scalar) is repeatedly combined with a base point on the curve through point doubling and addition, both of which reduce to modular multiplications. The choice of multiplier architecture therefore directly determines the computational efficiency, area, and power of the ECC core. This section describes the four multiplier architectures compared in this work.

A. Montgomery Multiplier

Montgomery multiplication avoids the explicit division required by conventional modular multiplication $(A \times B) \bmod N$ by working in a transformed “Montgomery representation” with respect to a modulus N and a radix R chosen as a power of two. Fig. 2 shows the components of the architecture used in this work. Given operands A and B in Montgomery form,

modulus N , and precomputed constants R and R^{-1} , the multiplier computes $T = A \times B$, then $U = T \times R^{-1} \bmod N$, and forms $V = (T + U \times N)/R$; V is reduced by subtracting N whenever $V \geq N$, and the final V is the Montgomery-domain result of $A \times B \bmod N$, which can be converted back to standard representation if required.

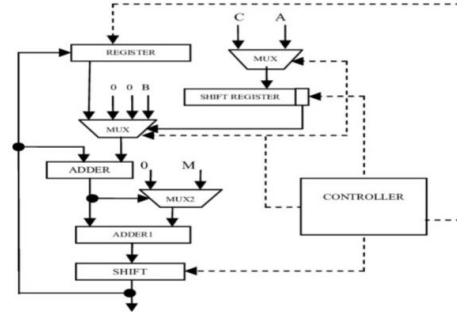


Fig. 2. Architecture of the Montgomery multiplier.

B. Baugh-Wooley Multiplier

The Baugh-Wooley technique targets signed (two's-complement) multiplication. It first forms an $n \times n$ matrix of partial products by multiplying every bit of the multiplier A against every bit of the multiplicand B , then exploits the structural overlap of these partial products to reduce the number of adders needed, folding the sign-correction terms into the final column summation rather than treating them as a separate step. Power-efficient FPGA realizations of this structure have been reported both at the level of basic building blocks [20] and through pipelined, reconfigurable fixed-width variants that apply gated-clock and zero-input techniques to cut dynamic power [21]. Fig. 3 shows the resulting architecture.

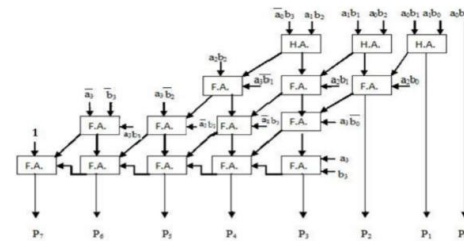


Fig. 3. Architecture of the Baugh-Wooley multiplier.

Column-wise summation of the partial-product matrix, with carry propagation between columns, yields a $2n$ -bit product; the most significant bit of the upper half is checked to detect overflow before the final result is presented.

C. Vedic Multiplier

The Vedic multiplier implemented here is based on the Urdhva-Tiryagbhyam (“vertically and crosswise”) sutra from ancient Vedic mathematics. The operands are loaded into input registers and segmented by bit position; each segment of A is cross-multiplied against each segment of B , and the resulting partial products are accumulated through a tree of binary adders with carry propagation handled at each stage.

FPGA realizations using resource-efficient adder trees, such as the SPST-adder-based design of Gowreesrinivas et al. [19], have reported reduced area and power relative to conventional ripple-carry accumulation for the same sutra. Fig. 4 shows the architecture.

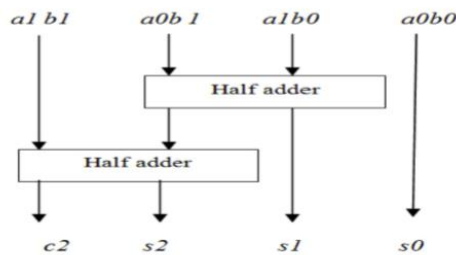


Fig. 4. Architecture of the Vedic multiplier.

Because all partial products are generated and summed in effectively one structural step, the Vedic multiplier is well suited to parallel, FPGA-friendly implementation, at the cost of a regular but resource-intensive adder tree.

D. Booth's Multiplier

Booth's algorithm encodes the multiplier M in overlapping 2-bit windows (patterns '00', '01', and '10') to decide, at each step, whether to add M to, subtract M from, or leave unchanged an accumulator A , followed by an arithmetic right shift of the combined (A, Q) register pair. Fig. 5 shows the architecture used in this work.

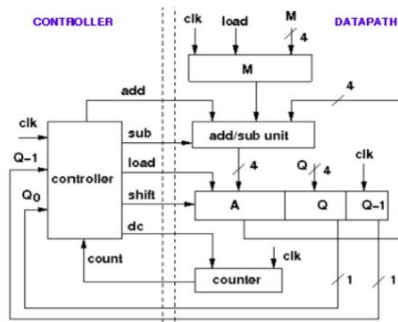


Fig. 5. Architecture of Booth's multiplier.

Booth recoding reduces the number of partial products that must be added relative to a naive shift-and-add multiplier — the same baseline that area-efficiency optimizations such as Pathan et al. [6] target directly — which is the main source of its area and power advantage in the results reported in Section V.

V. SIMULATION AND HARDWARE RESULTS

Each multiplier was first characterized in isolation on Xilinx Vivado for a common 8-bit input pair (7 and 7), and the Montgomery multiplier was then integrated into a complete ECC point-multiplication datapath. This section reports the schematic, simulated device design, functional simulation, and power analysis for each case.

A. Montgomery Multiplier

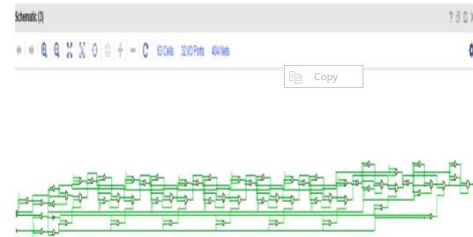


Fig. 6. Schematic of the Montgomery multiplier.

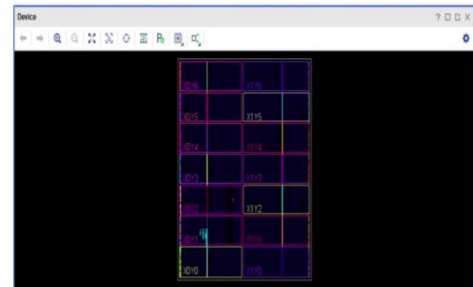


Fig. 7. Simulated device design of the Montgomery multiplier.

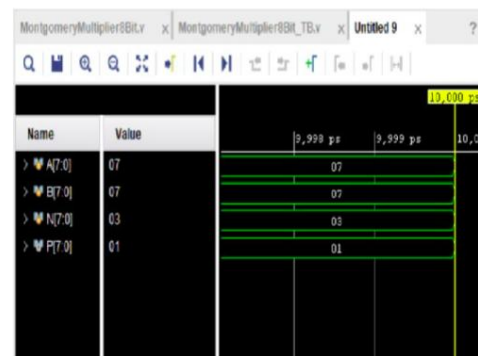


Fig. 8. Simulation results of the Montgomery multiplier.

Fig. 8 shows the simulation result of the Montgomery Modular Multiplier for inputs 7 and 7 with $N = 3$; the product 49_{10} reduces to an output of '1' after the modulo operation.

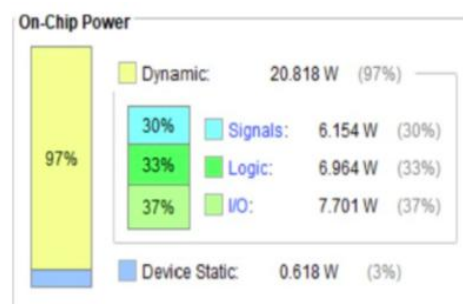


Fig. 9. Power report of the Montgomery multiplier.

Fig. 9 shows the power analysis of the Montgomery multiplier module for the same inputs. The static and dynamic power consumptions are 0.618 W and 20.818 W respectively, for a total average power of 21.436 W.

B. Baugh-Wooley Multiplier

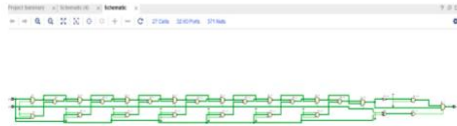


Fig. 10. Schematic of the Baugh-Wooley multiplier.

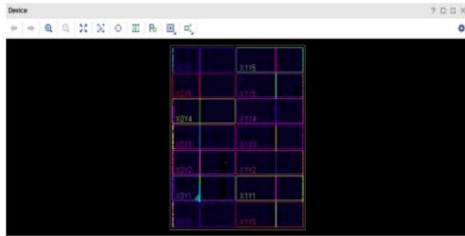


Fig. 11. Simulated device design of the Baugh-Wooley multiplier.

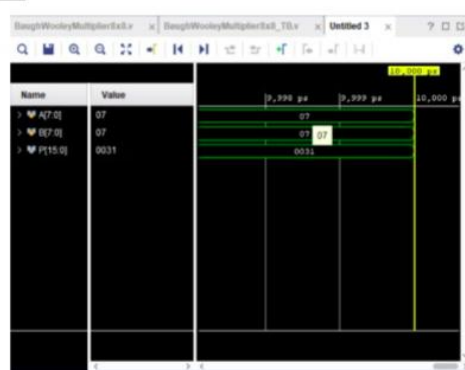


Fig. 12. Simulation results of the Baugh-Wooley multiplier.

Fig. 12 shows the simulation result of the Baugh-Wooley multiplier for inputs 7 and 7; the output obtained is 31h.

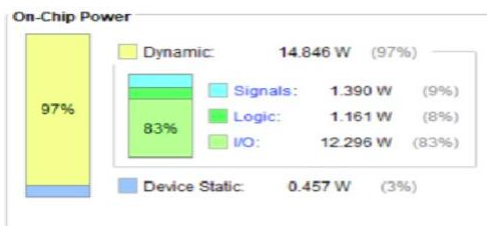


Fig. 13. Power report of the Baugh-Wooley multiplier.

Fig. 13 shows the power analysis of the Baugh-Wooley multiplier module. The static and dynamic power consumptions are 0.457 W and 14.846 W respectively, for a total average power of 15.303 W.

C. Vedic Multiplier

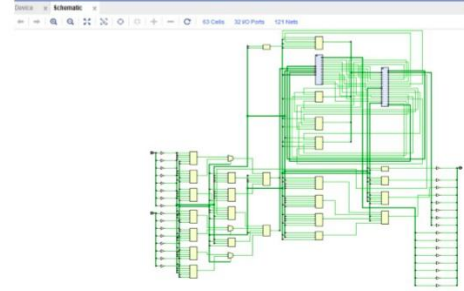


Fig. 14. Schematic of the Vedic multiplier.

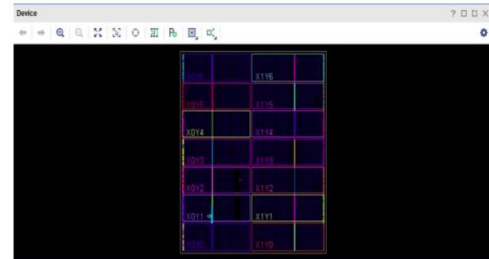


Fig. 15. Simulated device design of the Vedic multiplier.

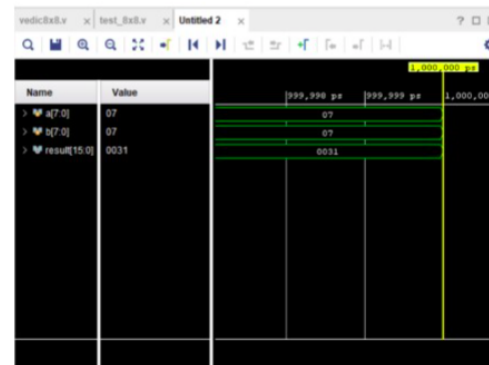


Fig. 16. Simulation results of the Vedic multiplier.

Fig. 16 shows the simulation result of the Vedic multiplier for inputs 7 and 7; the output obtained is 31h.

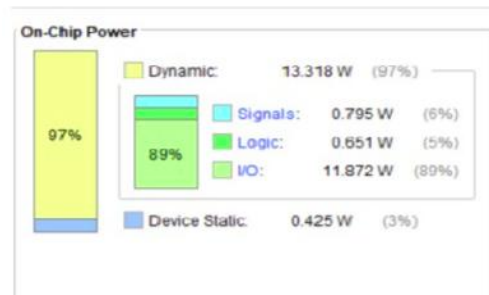


Fig. 17. Power report of the Vedic multiplier.

Fig. 17 shows the power analysis of the Vedic multiplier module. The static and dynamic power consumptions are 0.425 W and 13.318 W respectively, for a total average power of 13.743 W.

D. Booth's Multiplier

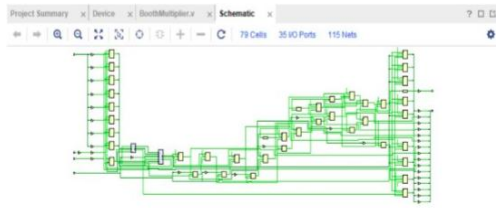


Fig. 18. Schematic of Booth's multiplier.

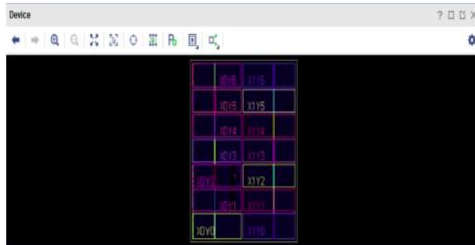
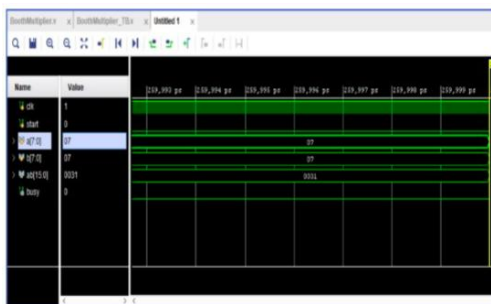


Fig. 19. Simulated device design of Booth's multiplier.



F. Power Comparison

TABLE I. POWER CONSUMPTION OF THE FOUR STANDALONE MULTIPLIERS

Multiplier	Static Power (W)	Dynamic Power (W)	Total Average Power (W)
Montgomery	0.618	20.818	21.436
Baugh-Wooley	0.457	14.846	15.303
Vedic	0.425	13.318	13.743
Booth's	0.315	6.652	6.967

Table I compares the static power, dynamic power, and total average power of the Montgomery, Baugh-Wooley, Vedic, and Booth multipliers, all measured under the same input pair and Vivado settings.

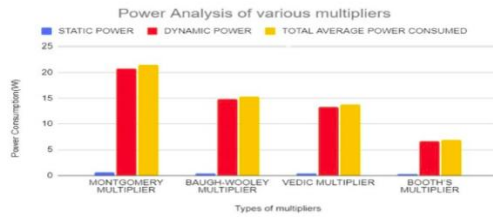


Fig. 26. Graphical comparison of standalone multiplier power consumption.

Fig. 26 plots the same standalone results graphically: Booth's multiplier draws the least total power of the four (6.967 W), followed by Vedic (13.743 W), Baugh-Wooley (15.303 W), and Montgomery (21.436 W) — the reverse of the ranking that would be expected purely from area, and a reminder that the standalone comparison and the system-level comparison in Table II are not interchangeable.

TABLE II. POWER CONSUMPTION OF THE IMPLEMENTED ECC CORE USING THE MONTGOMERY MULTIPLIER

Design	Static Power (W)	Dynamic Power (W)	Total Power (W)
ECC using Montgomery multiplier	6.632	332.107	338.739

Table II reports the corresponding static, dynamic, and total on-chip power for the complete ECC point-multiplication datapath built around the Montgomery multiplier — the only one of the four architectures integrated into a full ECC core in this study.

VI. CONCLUSION

This paper examined the effect of multiplier architecture on the power consumption of FPGA-based Elliptic Curve Cryptography (ECC) hardware, comparing Montgomery, Vedic, Baugh-Wooley, and Booth multipliers. Each multiplier was first characterized in isolation on Xilinx Vivado, giving a total average power of 21.436 W (Montgomery), 13.743 W (Vedic), 15.303 W (Baugh-Wooley), and 6.967 W (Booth) for a common input pair. The Montgomery multiplier was then integrated into a complete ECC point-multiplication datapath, yielding a static power of 6.632 W, a dynamic power of 332.107 W, and a total on-chip power of 338.739 W, confirming that Montgomery-based

modular reduction is a workable basis for ECC hardware on this platform.

Integrating the Vedic, Baugh-Wooley, and Booth multipliers into the same ECC datapath was not completed in this work due to synthesis errors, so the comparison presented here covers the multipliers' standalone power profiles rather than their behaviour inside a full ECC core. This is the principal limitation of the present study: the standalone results indicate relative power efficiency among the four multipliers in isolation, but only the Montgomery-based design has been validated at the ECC system level, and — as Fig. 26 shows — the standalone ranking (Booth lowest, Montgomery highest) should not be assumed to carry over once each multiplier is embedded in a full point-multiplication schedule with its associated control and routing overhead.

Future work will resolve the integration issues to complete the four-way ECC-level comparison, extend validation from the current toy-sized operands to standard curve parameters (e.g., NIST P-192/P-256) with multiple test vectors, and report area (LUT/slice/DSP utilization), maximum operating frequency, and throughput alongside power, enabling a complete area-delay-power characterization of lightweight ECC hardware — following the direction of [11] and [12], which integrate alternative multipliers directly into an ECC/Montgomery datapath rather than benchmarking them standalone.

REFERENCES

- [1] C.-Y. Lee, M. Zeghid, A. Sghaier, H. Y. Ahmed, and J. Xie, "Efficient Hardware Implementation of Large Field-Size Elliptic Curve Cryptographic Processor," IEEE Access, vol. 10, pp. 7926–7936, 2022.
- [2] Sagar and B. Chourasia, "VLSI Architecture of Lightweight Cipher Cryptography Technique for High Speed FPGA Applications," Int. J. Emerging Technologies and Innovative Research, vol. 9, no. 10, 2022.
- [3] J. Li, W. Wang, J. Zhang, Y. Luo, and S. Ren, "Innovative Dual-Binary-Field Architecture for Point Multiplication of Elliptic Curve Cryptography," IEEE Access, vol. 9, pp. 12405–12419, 2021.
- [4] A. Fotovvat, G. M. E. Rahman, S. S. Vedaeei, and K. A. Wahid, "Comparative Performance Analysis of Lightweight Cryptography Algorithms for IoT Sensor Nodes," IEEE Internet of Things Journal, vol. 8, no. 10, pp. 8279–8290, 2021.
- [5] R.-M. Coliban, "FPGA Implementation of Interleaved Modular Multiplier with Improved Area-time Efficiency," in Proc. 11th IEEE Int. Conf. Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), Cracow, Poland, 2021, pp. 895–898.
- [6] A. Pathan, A. H. Chandio, and R. Aziz, "An Optimization in Conventional Shift & Add Multiplier for Area-Efficient Implementation on FPGA," in Proc. Int. Conf. Emerging Technologies in Electronics, Computing and Communication (ICETECC), Jamshoro, Pakistan, 2022, pp. 1–6.
- [7] N. S. Sajju and S. K. N., "Design and Execution of Highly Adaptable Elliptic Curve Cryptographic Processor and Algorithm on FPGA using Verilog HDL," in Proc. Int. Conf. Communication, Control and Information Sciences (ICCISc), Idukki, India, 2021, pp. 1–6.
- [8] O. Al-Khaleel, S. Bakır, and A. Kıpçık, "FPGA Implementation of an ECC Processor Using Edwards Curves and DFT Modular Multiplication," in Proc. 12th Int. Conf. Information and Communication Systems (ICICS), Valencia, Spain, 2021, pp. 344–351.

- [9] Wang, D. Lin, Y. Hu, J. Zhang, and C. Zhong, "FPGA Implementation for Elliptic Curve Cryptography Algorithm and Circuit with High Efficiency and Low Delay for IoT Applications," *Micromachines*, vol. 14, p. 1037, 2023.
- [10] M. Sivajyothi and D. T., "Analysis of Elliptic Curve Cryptography with AES for Protecting Data in Cloud with Improved Time Efficiency," in *Proc. 2nd Int. Conf. Innovative Practices in Technology and Management (ICIPTM)*, Gautam Buddha Nagar, India, 2022, pp. 573–577.
- [11] G. Leelavathi, K. Shaila, and K. R. Venugopal, "Elliptic Curve Crypto Processor on FPGA using Montgomery Multiplication with Vedic and Encoded Multiplier over $GF(2^m)$ for Nodes in Wireless Sensor Networks," in *Proc. 2018 IEEE 13th Int. Conf. Industrial and Information Systems (ICIIS)*, 2018, pp. 207–210.
- [12] T. Abirami, S. Saravanan, A. Rajeshkumar, and K. M. Santhosh, "FPGA-Based Optimized Design of Montgomery Modular Multiplier Using Karatsuba Algorithm," in *Proc. 2023 Int. Conf. Electronics and Renewable Systems (ICEARS)*, IEEE, 2023, pp. 131–135.
- [13] C. M. Kalaiselvi and R. S. Sabeenian, "A modular technique of Booth encoding and Vedic multiplier for low-area and high-speed applications," *Scientific Reports*, vol. 13, art. 22379, 2023.
- [14] R. Ifrim, D. Loghin, and D. Popescu, "A Systematic Review of Fast, Scalable, and Efficient Hardware Implementations of Elliptic Curve Cryptography for Blockchain," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 17, no. 4, pp. 1–33, 2024.
- [15] A. M. Awaludin, J. Park, R. W. Wardhani, and H. Kim, "A High-Performance ECC Processor over Curve448 Based on a Novel Variant of the Karatsuba Formula for Asymmetric Digit Multiplier," *IEEE Access*, vol. 10, pp. 67470–67481, 2022.
- [16] A. A. H. Abd-Elkader, M. Rashdan, E.-S. A. M. Hasaneen, and H. F. A. Hamed, "FPGA-Based Optimized Design of Montgomery Modular Multiplier," *IEEE Trans. Circuits Syst. II*, vol. 68, no. 6, pp. 2137–2141, 2021.
- [17] R. Wu, M. Xu, Y. Yang, G. Tian, P. Yu, Y. Zhao, B. Lian, and L. Ma, "Efficient High-Radix $GF(p)$ Montgomery Modular Multiplication via Deep Use of Multipliers," *IEEE Trans. Circuits Syst. II*, vol. 69, no. 12, pp. 5099–5103, 2022.
- [18] S.-R. Kuang, C.-Y. Liang, and C.-C. Chen, "An Efficient Radix-4 Scalable Architecture for Montgomery Modular Multiplication," *IEEE Trans. Circuits Syst. II*, vol. 63, no. 6, pp. 568–572, 2016.
- [19] K. V. Gowreesrinivas, S. Srinivas, and P. Samundiswary, "FPGA Implementation of a Resource Efficient Vedic Multiplier Using SPST Adders," *Eng. Technol. Appl. Sci. Res.*, vol. 13, no. 3, pp. 10698–10702, 2023.
- [20] M. H. Rais, B. M. Al-Harhi, S. I. Al-Askar, and F. K. Al-Hussein, "Design and Field Programmable Gate Array Implementation of Basic Building Blocks for Power-Efficient Baugh-Wooley Multipliers," *Am. J. Eng. Appl. Sci.*, vol. 3, no. 2, pp. 307–311, 2010.
- [21] J.-H. Tu and L.-D. Van, "Power-Efficient Pipelined Reconfigurable Fixed-Width Baugh-Wooley Multipliers," *IEEE Trans. Comput.*, vol. 58, no. 10, pp. 1346–1355, Oct. 2009.