

Article

Low-Power Design Implementation of AES-128-CCM Coprocessor for Secure Chip

Jian-Qiang Wang ¹, Yu-Chun Li ¹, Wei (David) Zhang ^{1,2} and Hong-Liang Lu ^{1,2,*} 

¹ School of Microelectronics, Fudan University, Shanghai 200433, China; 20112020121@fudan.edu.cn (J.-Q.W.); yuchun_li@fudan.edu.cn (Y.-C.L.); dwzhang@fudan.edu.cn (W.Z.)

² Jia-Shan Fudan Institute, Jiading 201300, China

* Correspondence: honglianglu@fudan.edu.cn

Abstract

This paper presents a low-power hardware implementation of an AES-CCM coprocessor for secure chips in embedded systems. The proposed design performs key expansion only once and stores the round keys in an on-chip RAM to avoid redundant computations. Meanwhile, the S-box module is shared between the key expansion and encryption to reduce hardware overhead. A dual-mode architecture supporting parallel (two-core) and serial (single-core) operations is implemented to adapt to high-throughput and low-power scenarios. The design supports AES-128, with a 1.25 Kb RAM used to store the 10 round keys. Experimental results using TSMC 40 nm technology show that the parallel mode achieves a 5.4% power reduction at the cost of 12.8% area overhead compared with the reference design. The energy efficiency reaches 2.11 pJ/bit in the parallel mode and 2.17 pJ/bit in the serial mode.

Keywords: AES-128-CCM; low-power design; security chip

1. Introduction

With the rapid proliferation of embedded IoT terminals, wearable equipment, smart metering chips and automotive secure controllers, embedded security chips have become indispensable core components for data confidentiality and anti-tampering protection. Miniature embedded hardware is constrained by limited battery capacity and on-chip resources, imposing strict concurrent requirements on cipher coprocessors for power consumption, silicon area and real-time throughput. The Advanced Encryption Standard Counter with Cipher Block Chaining Message Authentication Code (AES-CCM) [1] encryption protocol was released by the IEEE Wireless Standards Group in 2004 as a security standard for wireless local area networks. AES-CCM is an encryption protocol that combines the AES Cipher Block Chaining (CBC) mode and Counter (CTR) mode to provide both data encryption and integrity protection functionalities. However, its hardware implementation on secure chips faces a fundamental challenge: the need to perform both CBC and CTR operations while maintaining low-power consumption and a small silicon area, without sacrificing throughput when required.

Existing academic research has proposed various AES-CCM optimization schemes from the perspectives of S-box logic compression, multi-core parallelism, a serialized data path and peripheral power management perspectives. However, a comprehensive literature review reveals three critical research gaps that existing published AES-CCM hardware



Academic Editor: Taeshik Shon

Received: 25 April 2026

Revised: 23 June 2026

Accepted: 23 June 2026

Published: 25 June 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

implementations fail to resolve in a unified architecture. First, almost all reported AES-CCM implementations execute key expansion repeatedly for each new encryption packet. Although storing pre-expanded round keys in on-chip RAM has been exploited in some AES implementations, no prior AES-CCM hardware design integrates this technique with unified S-box sharing and runtime-configurable dual-mode (parallel/serial) operation.

Second, most existing S-box optimization works only optimize standalone encryption or key-exp modules separately; few designs realize the dynamic time-sharing reuse of S-box resources between the key expansion and round encryption, leading to a duplicate S-box layout and extra silicon area cost.

Third, prior architectures are mostly fixed as either fully parallel multi-core for high speed or single serial core for low power. There is a lack of configurable dual-mode hardware that can dynamically switch between parallel/serial operation at runtime according to task requirements, so a single chip cannot simultaneously balance high-throughput transmission and battery-saving, low-power working conditions.

To fill these gaps, this paper proposes a low-power AES-128-CCM coprocessor with the following main contributions:

- A one-shot key expansion scheme with the round keys stored in a RAM, eliminating repeated key expansion.
- Unified S-box sharing between the key expansion and encryption, reducing the hardware resource count.
- A dual-mode (parallel/serial) architecture supporting both high-throughput and ultra-low-power applications.

To the best of our knowledge, little prior work has simultaneously integrated one-shot key expansion with round-key storage in a RAM, dynamic time-sharing S-box reuse between the key expansion and encryption cores, and runtime-configurable parallel/serial dual-mode operation in a single low-power AES-128-CCM coprocessor for embedded low-power scenarios. This paper presents such an architecture that synergistically combines all three optimizations.

The remainder of this paper is organized as follows. Section 2 describes the existing related work. Section 3 analyzes the AES-CCM protocol and motivates the optimizations. Section 4 details the proposed coprocessor architecture. Section 5 presents the implementation results and comparisons. Section 6 concludes the paper.

2. Related Work

AES-CCM is the dominant authenticated encryption algorithm for resource-limited IoT embedded systems. Over the past two decades, abundant hardware acceleration solutions have been proposed to optimize its area and power performance. This section classifies existing research into four mainstream technical branches, conducts an in-depth critical analysis of the inherent defects of each category, extends the original comparison table with comprehensive qualitative evaluation metrics, and explicitly highlights the distinct novelty and research positioning of the proposed design against the identified research limitations.

2.1. S-Box Combinational Logic Optimization Schemes

Morioka et al. simplified the logical expressions between different domains and adjacent transformations, proposing new S-box-circuit implementation methods that reduce power consumption by minimizing the number of logic gates and optimizing signal paths [2–4]. Additionally, studies have shown that S-boxes based on combinational logic are more efficient than LUT-based implementations, enabling low-area and low-power designs [5–7].

The surveyed schemes in this category treat the key expansion and round encryption as independent computing branches, deploying completely separate S-box arrays for the two stages. Every new data packet triggers a full key expansion, which repeatedly invokes dedicated S-box hardware and generates continuous redundant switching power. The optimization scope is limited to individual S-box components, lacking global data-flow scheduling. Even if a single S-box achieves a compact layout, duplicated S-box groups for key generation and encryption usually introduce extra silicon overhead, and the iterative key expansion operation remains a persistent power bottleneck without any round-key caching mechanism. These designs lack a cross-module hardware-reuse approach and cannot achieve system-level energy optimization.

2.2. Multi-Core Parallel High-Throughput Architectures

In 2013, Nguyen et al. designed a dual-AES-encryption-core hardware architecture by parallelizing CBC and CTR modes, achieving a throughput of 2.69 Gbps at 269 MHz [8]. In 2019, Pamu's team proposed a nine-core parallel scheme, optimizing the checksum generation process using finite field $GF(2^8)$ multiplication, increasing the data processing rate to 8.32 Gbps [9].

Multi-core architectures sacrifice power efficiency for throughput performance. Each independent AES core integrates a complete set of key expansion and S-box circuits, resulting in substantial static power consumption. Most of these parallel implementations lack on-chip RAM to store the precomputed round keys, so the key expansion is recalculated for each frame during long-term, continuous encryption, leading to significant energy waste. More importantly, the hardware structure is fixed without reconfiguration capability; the multi-core circuit cannot be shut down partially under low-traffic, idle states, making it incompatible with battery-powered wearable and smart-meter IoT devices that require long-standby, low-power operation. No dynamic mode-switching logic is embedded in these fixed parallel designs.

2.3. Lightweight Serial Single-Core Hardware

Hoang et al. introduced an ultra-low-power 8-bit single-core AES-CCM implementation, sacrificing performance for reduced power consumption [10]. In 2020, Shahbazi et al. proposed a low-power, low-area, 8-bit, lightweight data path accelerator specifically designed for resource-constrained mobile SoCs and devices [11].

Serial designs suffer from obvious throughput degradation when processing large data blocks. Similar to their parallel counterparts, they execute repeated key expansion for every packet and cannot share S-box hardware between key expansion and encryption. The circuit is hardwired exclusively for the serial mode and lacks parallel acceleration branches, which prevents it from satisfying the burst high-speed transmission demands of industrial IoT terminals. Such single-mode hardware has a narrow range of application scenarios and cannot support multi-scenario dynamic switching of power and performance levels.

2.4. Partial Module and Peripheral Optimization Architectures

In 2021, Pandey et al. analyzed the power consumption of different FPGA I/Os, selecting the lowest-power I/O and further reducing the AES module power consumption through frequency scaling, thermal scaling, and clock gating [12]. Cui et al. proposed dual-core parallel and ping-pong serial modes, sharing the key expansion module and reusing S-box resources, achieving a 12.34% reduction in power consumption compared to traditional implementations while maintaining performance [13]. However, their AES-CCM encryption cycle still required continuous key expansion operations, resulting in relatively high power consumption. In 2022, Nanni Pieri et al. proposed a configurable AES hardware accelerator architecture to minimize logic resources and reduce module power

consumption [14]. In 2023, Sai et al. merged row-shift operations with state register updates, implemented column-mixing transformations through substructure sharing, and applied clock gating to achieve low-power, high-speed devices [15]. Kala et al. adopted a low-power key expansion mechanism for iterative stages to minimize hardware consumption [16].

These works only carry out scattered local optimizations instead of a global collaborative improvement covering key derivation, S-box resource allocation and operation mode scheduling. Cui's design only reuses partial intermediate registers rather than the core S-box arrays and still recalculates the round keys frame by frame. Clock gating and I/O tuning belong to peripheral-level optimizations with limited energy-saving gains, which cannot eliminate the internal redundant power loss caused by duplicated key expansion. To the best of our knowledge, no prior scheme has simultaneously integrated all three optimizations, so they cannot fundamentally resolve the trade-off conflict between throughput and power consumption.

2.5. Common Limitations of Existing Research and Positioning of This Work

Based on the above analysis, three common limitations exist in current AES-CCM hardware implementation research. Table 1 provides a qualitative comparison of mainstream schemes across five dimensions.

Table 1. Qualitative comparison of state-of-the-art AES-CCM hardware implementations.

References	Repeated Key Expansion	Shared S-Box	Configurable Dual Mode	Core Optimization Direction	Main Limitation
Refs. [2–7]	Yes	No	Single mode	S-box combinational optimization	Cannot reduce key-exp redundant power
Refs. [8,9]	Yes	No	Parallel only	Multi-core parallel acceleration	High static and dynamic power
Refs. [10,11]	Yes	No	Serial only	Lightweight serialized path	Poor throughput performance
Ref. [12]	Yes	Partial	Single mode	Clock and I/O power optimization	No inner circuit optimization
Ref. [13]	Yes	Partial	Dual but repeated key-exp	Partial module reuse	Repeated key expansion
Refs. [14–16]	Yes	Partial	Single mode	Operation fusion/iterative optimization	No one-shot key pre-storage
This Work	No	Yes	Dual switchable	One-shot key expansion + S-box sharing	Slight area increase

Limitation 1. *Repeated key expansion computations. Nearly all reported schemes re-execute key expansion for each new encryption packet. This paper proposes a one-shot key expansion with the round keys stored in a RAM, eliminating these redundant computations.*

Limitation 2. *Redundant S-box resource layout. Most designs configure independent S-box instances for key expansion and round encryption, causing area waste. This paper achieves the dynamic, time-sharing reuse of S-box resources between the key expansion and encryption cores, reducing the number of S-boxes from 40 to 16 in the dual-core parallel scheme.*

Limitation 3. *Fixed operating modes. Existing architectures are fixed as either fully parallel or fully serial, lacking a dynamic switching capability. This paper designs a dual-mode reconfigurable architecture supporting runtime-configurable parallel/serial switching.*

Table 1 provides a qualitative comparison of mainstream published AES-CCM hardware implementations across five dimensions: key expansion mode, S-box sharing, configurable working mode, core optimization direction, and main limitations. As shown in the

table, existing research leaves gaps in all three of the aforementioned dimensions that have not been adequately addressed.

Different from scattered local optimizations in the previous literature, our design realizes systematic collaborative optimization from key generation, core arithmetic resource allocation to task-level performance-power scheduling. Besides the three hardware optimization innovations, this work embeds first-order algebraic masking S-box countermeasures to resist DPA/CPA side-channel attacks, which most existing AES-CCM hardware modules neglect.

3. AES-CCM Protocol Analysis and Optimization Motivation

In the CCM mode of the AES algorithm, CTR mode is responsible for encrypting the original data to ensure data confidentiality, while CBC mode is used to compress the plaintext data and generate a Message Authentication Code (MAC) to guarantee data integrity and authenticity [1]. The system appends the generated MAC value to the ciphertext, transmitting both to the receiving end. At the receiving end, the decryption process uses the same key as the encryption process: first, CTR mode decrypts the received ciphertext to restore the original plaintext data; then, CBC mode calculates the MAC value from the decrypted plaintext. By comparing the computed MAC value with the received MAC value, if they match, it confirms that the data has not been tampered with during transmission, thereby ensuring data reliability.

The hardware implementation of the AES-CCM encryption protocol primarily adopts two architectures: parallel implementation and serial implementation. The parallel implementation scheme involves two AES encryption cores, where one core implements CBC mode and the other implements CTR mode, operating in parallel. The serial implementation scheme uses only one AES encryption core, alternating between CBC mode and CTR mode. The advantage of the parallel scheme is its high performance, making it suitable for applications requiring high-speed processing, but at the cost of higher power consumption. The serial scheme, on the other hand, offers lower power consumption but slower speed, making it suitable for applications with lower performance requirements but higher demands for power efficiency.

Regarding the AES-CCM protocol, we observe that:

- In the AES-CCM encryption protocol, except for the first packet of data in CBC mode used to generate the MAC value, the remaining CBC mode and CTR mode operations can be executed in parallel.
- The entire AES-CCM encryption protocol uses only one key, meaning the same key is shared between CBC mode and CTR mode.
- Throughout the AES-CCM protocol, only AES encryption is used for both encryption during transmission and decryption during reception; AES decryption is not required.

Based on the above analysis, this paper designs and implements a low-power AES-CCM coprocessor. This coprocessor can dynamically adjust its working mode according to application requirements: in high-speed application scenarios, it adopts the AES parallel implementation method; in low-power application scenarios, it switches to the AES serial implementation method.

4. Proposed AES-CCM Coprocessor Architecture

4.1. Key Expansion Optimization with RAM Storage

This paper proposes an optimized solution: in the entire AES-CCM encryption protocol, the key expansion calculation is performed only once. The expanded round keys are stored in a 1.25 Kb RAM, with each round key being 128 bits and a total of 10 rounds, as shown in Figure 1.

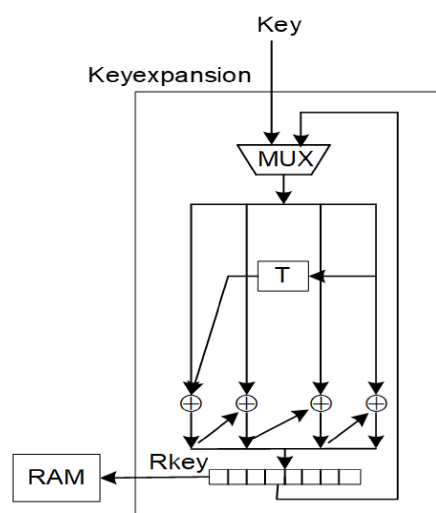


Figure 1. Implementation of AES parallel optimization and key expansion storage in AES-CCM encryption protocol.

In the parallel computation process of the AES-CCM encryption protocol, there is no need to repeatedly perform key expansion operations. By storing the round keys in a RAM, the round keys can be directly read from the RAM for AES-parallel computation, effectively reducing execution power consumption. The specific implementation is shown in Figure 2.

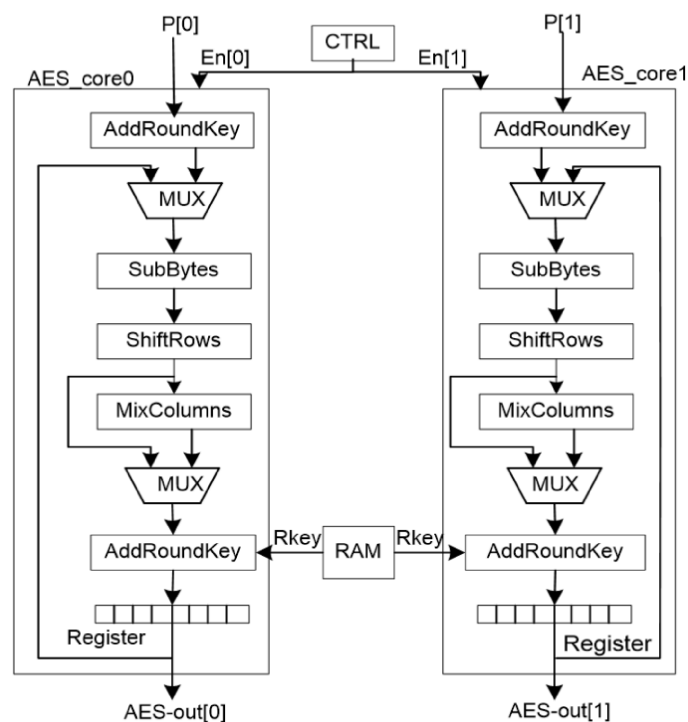


Figure 2. AES Parallel Optimization in AES-CCM Encryption Protocol.

By reusing the pre-generated round key design, the overhead of re-performing key expansion during each encryption computation is avoided, reducing computational overhead and significantly lowering dynamic power. Additionally, reading the round keys from RAM enhances the efficiency of the encryption process, enabling it to better meet the dual requirements of high performance and low-power consumption.

To ensure the continuity of AES parallel computation in the AES-CCM encryption protocol, the key expansion operation is integrated into the initialization module of the overall design. This approach ensures that the processor only needs to perform an initialization

operation once before executing the AES-CCM encryption protocol. For scenarios where the AES serial implementation is chosen in low-power applications, the key expansion operation is similarly moved to the initialization phase, and the pre-generated round keys are reused during the encryption process, avoiding the overhead of repeated computations.

The proposed design supports AES-128 only. To achieve compatibility with AES-192 and AES-256, a full redesign of the RAM architecture is required. These two algorithms need 12 and 14 round keys and larger RAM resources, and such hardware adjustments are inconsistent with our low-power design objectives.

Key expansion is performed only once during initialization. All subsequent encryption cycles read the round keys directly from RAM, avoiding redundant computations and reducing power consumption significantly.

4.2. S-Box Sharing Mechanism

The AES encryption algorithm consists of two computational components: key expansion and round encryption. Both the key expansion and round encryption stages require S-box byte substitution operations. Typically, the key expansion employs four parallel S-box byte substitution operations, while round encryption requires 16 parallel S-box byte substitution operations.

Through optimized design, the S-box is reused: the same S-box is utilized in both the key expansion and round encryption stages. This design eliminates the need for four parallel S-boxes during the key expansion phase. In addition, the time-division reuse of eight S-boxes is also adopted in the round function of each AES core: only eight S-box operations are executed per clock cycle, so one full round takes two clock cycles to complete. This reduces the silicon area by eight S-box instances at the cost of one additional cycle per round.

The specific optimized structure is shown in Figure 3.

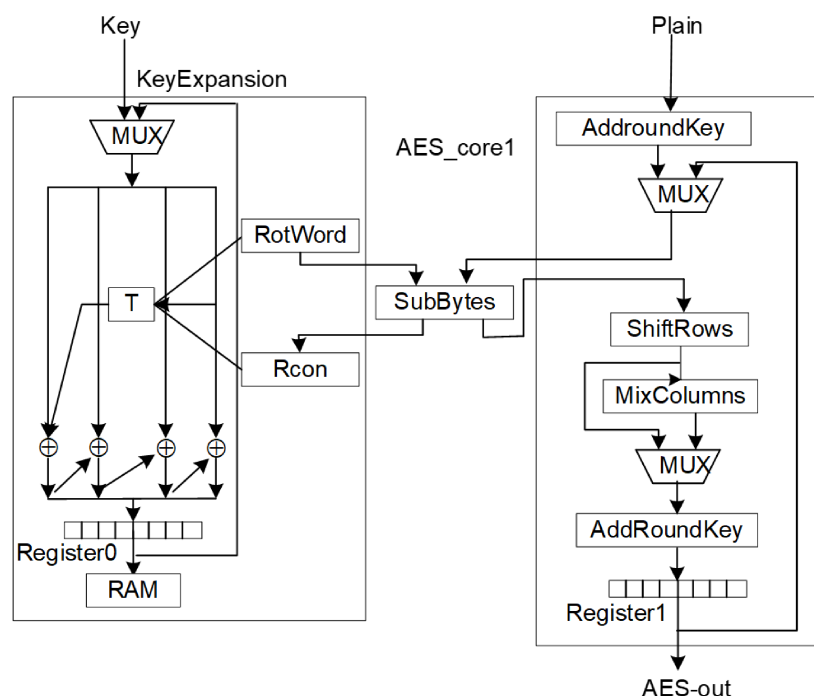


Figure 3. AES S-box reuse design.

4.3. Dual-Mode Architecture

The proposed architecture supports two working modes:

- Parallel mode: two AES cores, high throughput, for high-performance scenarios.

- Serial mode: one AES core, ultra-low power, for battery-powered embedded devices.

In the parallel mode, both En [0] and En [1] are asserted. The coprocessor instantiates two independent AES encryption cores, dedicated to the CBC mode and the CTR mode, respectively. The CBC core computes the message authentication code for the first data block, while the CTR core simultaneously performs encryption of subsequent plaintext blocks.

In the serial mode, only En [0] is asserted. The coprocessor uses a single AES encryption core that time-multiplexes the CBC and CTR modes. The data flow follows a sequential procedure: first, the CBC computation is performed on the initial authentication block; then, the same core reconfigures its data path to execute CTR mode encryption on each plaintext block. The round keys are still pre-expanded during initialization and stored in the RAM, but now they are supplied exclusively to the single core.

4.4. Advanced High-Performance Bus (AHB) Interface Integration and Data Flow Control

The AES-CCM coprocessor is designed as an AHB slave module, seamlessly integrated into the on-chip AHB matrix to facilitate efficient communication with the system processor and other peripherals. The AHB interface adopts a simplified slave protocol, supporting single-burst transactions and fixed-length burst transfers. This simplified protocol is sufficient for the data block sizes required by the AES-CCM operations. The coprocessor occupies a dedicated address region within the system memory map, enabling the processor to configure encryption parameters, initiate operations, and retrieve results through standard memory-mapped register accesses.

The AHB slave interface implements a 32-bit data width, matching the system bus specification. Since each AES-CCM data block is 128 bits wide, the input plaintext and output ciphertext are transferred across the AHB through four consecutive 32-bit write or read transactions. This design choice balances bus bandwidth utilization with hardware simplicity, avoiding the need for a wider 128-bit bus interface that would increase routing complexity and power consumption. The coprocessor internally buffers these four 32-bit words and assembles them into a complete 128-bit block before feeding the data into the AES encryption core.

To minimize processor intervention and reduce system-level power consumption, the coprocessor implements a dedicated Direct Memory Access (DMA) handshake interface that operates in conjunction with the system DMA controller. The DMA engine autonomously fetches plaintext data from system memory, delivers it to the AES-CCM core for encryption, and writes the resulting ciphertext back to memory without requiring continuous processor involvement.

The integration of the AHB interface with the internal RAM-based round key storage follows a strict access isolation policy. While the processor can trigger key expansion through the AHB control register, it cannot directly read the contents of the 1.25 Kb RAM that stores the expanded round keys. The RAM is exclusively accessible by the AES encryption cores through internal dedicated buses, physically decoupled from the AHB interface. This isolation ensures that sensitive cryptographic material never appears on the external system bus, mitigating the risk of bus-sniffing attacks or unintended information leakage through memory-mapped debug interfaces. The RAM write operations are performed solely by the key expansion module during initialization, and read operations are conducted exclusively by the AES cores during encryption, with no external visibility into these transactions.

4.5. Clock Gating and Dynamic Power Management

The AES-CCM coprocessor incorporates a hierarchical clock-gating strategy to achieve dynamic power reduction without compromising functional integrity. At the top level, a

global clock gate controls the entire coprocessor module, allowing the system to completely power down the cryptographic engine when no encryption tasks are pending. This coarse-grained gate is managed by the system power controller through the AHB interface, enabling software to power off the coprocessor during extended idle periods typical in battery-operated IoT devices.

The RAM storing the round keys implements a read-activated clock gate that activates only when the AES cores request key material. Since each round key is read exactly once per block encryption across ten rounds, the RAM clock duty cycle is inherently low. The clock gate leverages this temporal sparsity to suppress unnecessary toggling during the intervals between round key fetches, yielding measurable power savings, particularly evident in serial mode where encryption throughput is lower and idle intervals between rounds are proportionally longer.

5. Implementation and Experimental Results

The design is described in Verilog RTL, verified with NC-Verilog, and synthesized with Design Compiler under a TSMC 40 nm process at 200 MHz.

5.1. Implementation of the Computation Module

5.1.1. Parallel Implementation Scheme of the AES-CCM Encryption Module

The parallel implementation scheme of the AES-CCM encryption module includes one key expansion module, two AES encryption modules, and a 1.25 Kb RAM. During the initialization phase of the AES-CCM encryption protocol, the key expansion module completes the key expansion operation and stores the generated 10-round round keys in the RAM. In subsequent parallel encryption operations, the two AES encryption cores directly read and share the round keys from the RAM.

Additionally, a reuse design is applied to the S-box operations in the key expansion module and the encryption cores: the original four independent S-box operations in the key expansion module are eliminated, and the four S-box operations in the AES encryption cores are reused instead. Specifically, during the key expansion phase, the key expansion module occupies the four S-box resources; during non-key expansion phases, these S-box resources are released for use by the AES encryption module. With this S-box time-sharing mechanism, the total number of S-boxes in the dual-core parallel design is reduced from 40 to 16. Each core requires only eight S-boxes per round (enabled by reusing the same eight instances over two cycles), leading to a total of 16 S-boxes for both cores combined.

5.1.2. Serial Implementation Scheme of the AES-CCM Encryption Module

The serial implementation scheme of the AES-CCM encryption module includes one key expansion module, one AES encryption module, and a 1.25 Kb RAM. During the encryption process, the round keys stored in the RAM are provided exclusively to the single AES encryption core. The serial scheme also adopts a reuse design for the S-box operations in the key expansion module and the AES encryption core: the original four independent S-box operations in the key expansion module are eliminated, and the four S-box operations in the AES encryption core are reused. During the key expansion phase, the key expansion module occupies the four S-box resources; during non-key expansion phases, these S-box resources are released for use by the AES encryption module.

Through this design, the total number of S-boxes required for the overall computation in the serial implementation scheme is reduced from 20 to 8.

5.2. Security Implementation

5.2.1. Security Design

To protect the AES S-box against side-channel attacks, algebraic masking methods are adopted [17], while the compact composite-field S-box arithmetic basis adopted for masking hardware is derived from the low-area S-box design of Canright [18]. First-order algebraic masking methods re-express the S-box in a composite field so that the inversion operation can be masked step-by-step using smaller masked multipliers and adders. The masking arithmetic offers theoretically provable security against first-order DPA under ideal hardware assumptions. This algebraic scheme reduces the hardware area and random number overhead, at the cost of increased combinational logic depth and longer critical paths [19]. To enhance the resistance against side-channel attacks, a fresh random mask is applied for each round of AES encryption. Furthermore, the key expansion module adopts the same masking scheme as the data path. As a result, the round keys stored in the RAM are masked values instead of the original secret keys.

In our implementation, the RAM is placed inside the AES core. The external system bus has no direct access to the RAM. All writes to and reads from the RAM are performed directly by the AES core through dedicated internal interfaces. This physical isolation ensures that masked round keys never leave the core's trusted boundary.

Given our focus on low-power secure integrated circuits for IoT applications, higher-order attacks are considered out of scope. The significant power and area overhead associated with higher-order countermeasures does not align with the design constraints of resource-constrained IoT devices.

In the target embedded secure chip scenario, the RAM is placed inside the security perimeter with physical protection, voltage/clock tamper detection, and isolation [20]. Therefore, under the intended deployment environment, the key storage does not introduce obvious additional security risks. While our design does not include dedicated fault-injection countermeasures, the per-round random masking provides inherent resistance against basic fault attacks. Dedicated fault resilience mechanisms remain a direction for future research targeting high-security application domains.

5.2.2. Security Analysis

We assume an adversary with physical access to the chip. The attacker can:

- Collect power or electromagnetic traces during AES encryption.
- Perform differential power analysis (DPA) or correlation power analysis (CPA) attacks.

However, the following protections are in place:

- The RAM that stores the round keys resides inside the AES core. The external bus cannot access this RAM; all data transfers to/from the RAM are handled directly by the AES core via internal interfaces. Therefore, an attacker cannot directly read the masked round keys through external bus probing.
- Under the assumed threat model and within the secure perimeter, the key never appears in an unmasked form on any external bus or unprotected memory. Internal masked operations are executed only inside the core with verified first-order DPA resistance.
- We deploy first-order Boolean masking with random shares using the algebraic masking scheme in Ref. [17], which is secure against first-order DPA.
- The secure chip includes voltage/clock sensors and a dedicated security controller that erases all key material upon tamper detection [20].

Thus, in the intended deployment environment, the key storage does not introduce additional security risks beyond those already mitigated by the masking scheme and physical isolation.

To provide an initial, simulation-based security assessment, the Test Vector Leakage Assessment (TVLA) [21] was conducted on the proposed AES-128-CCM coprocessor RTL implementation featuring algebraic masking countermeasures, 8-S-box reuse architecture, and one-shot key expansion with RAM storage. A total of 100,000 simulated power traces were collected using a Hamming-weight power model under the non-specific fixed-versus-random Test Vector Leakage Assessment (TVLA) methodology, with 50,000 traces per group.

The *t*-statistic heatmap revealed the spatial distribution of leakage across all 20 clock cycles and 10 samples per cycle. The maximum absolute *t*-value was 4.2. This value remains below the standard detection threshold for side-channel leakage of 4.5 [22]. Based on simulated TVLA results, the algebraic masking scheme shows promising indications of first-order side-channel resistance, pending physical silicon validation. No sample points exceed the $|t| > 4.5$ threshold, indicating preliminary passing results under simulation-based TVLA evaluation.

The CPA using Hamming weight prediction model shows maximum correlation coefficients of 0.019 between predicted and measured power consumption—well below the 0.05 exploitation threshold and significantly lower than typical unmasked AES implementations (0.3–0.8). The uniform distribution of correlation across all 16-byte positions (0.014–0.019) demonstrates balanced masking application. Again, these are simulation-based results that require physical silicon validation to confirm actual resistance.

All side-channel evaluations in this work are based on simulation-derived power traces using a Hamming-weight model. We emphasize that the above TVLA and CPA results are simulation-based indications of first-order leakage resistance under a specific leakage model and do not constitute a validated security certification. Physical TVLA measurements on fabricated silicon, which are essential for conclusive side-channel resistance evaluation, are left for future work.

5.3. Functional Verification

The design adopts the Verilog hardware description language for RTL (Register Transfer Level) implementation of the overall scheme. The AES-CCM module is interfaced with the on-chip Advanced High-performance Bus (AHB), and data transmission is implemented via Direct Memory Access (DMA). NC-Verilog is utilized for functional simulation of the entire hardware module to verify the functionality of each component. The process for module functional verification is as follows:

- Configure the AES encryption mode to serial encryption mode, encrypt two 128-bit data blocks, and input the encryption key as 0xC0C1C2C3C4C5C6C7C8C9CACBCCDCECF. Wait for the key expansion operation to complete and store the generated round keys in the RAM. The simulation waveforms are shown in Figure 4a.
- Input the plaintext as data0 = 0xEB955546710A51AE25190A2DFE4B90D6 and data1 = 0x0100000003020100A0A1A2A3A4A50001. In serial encryption mode, the results are output sequentially as cipher0 = 0xCDB6411E3CDC9B4FF0915D9258B69EE7 and cipher1 = 0x50859D916DCB6DDDE077C2D1D4EC9F97. The simulation waveforms are shown in Figure 4b.
- Configure the AES encryption mode to parallel encryption mode, encrypt two 128-bit data blocks, and input the encryption key as 0xC0C1C2C3C4C5C6C7C8C9CACBCCDCECF. Wait for the key expansion operation to complete and store the generated round keys in the RAM. The simulation waveforms are shown in Figure 4a.

- Input the plaintext as data0 = 0xEB955546710A51AE25190A2DFE4B90D6 and data1 = 0x0100000003020100A0A1A2A3A4A50001. In parallel encryption mode, the results are output simultaneously as cipher0 = 0xCDB6411E3CDC9B4FF0915D9258B69EE7 and cipher1 = 0x50859D916DCB6DDDE077C2D1D4EC9F97. The simulation waveforms are shown in Figure 4c.

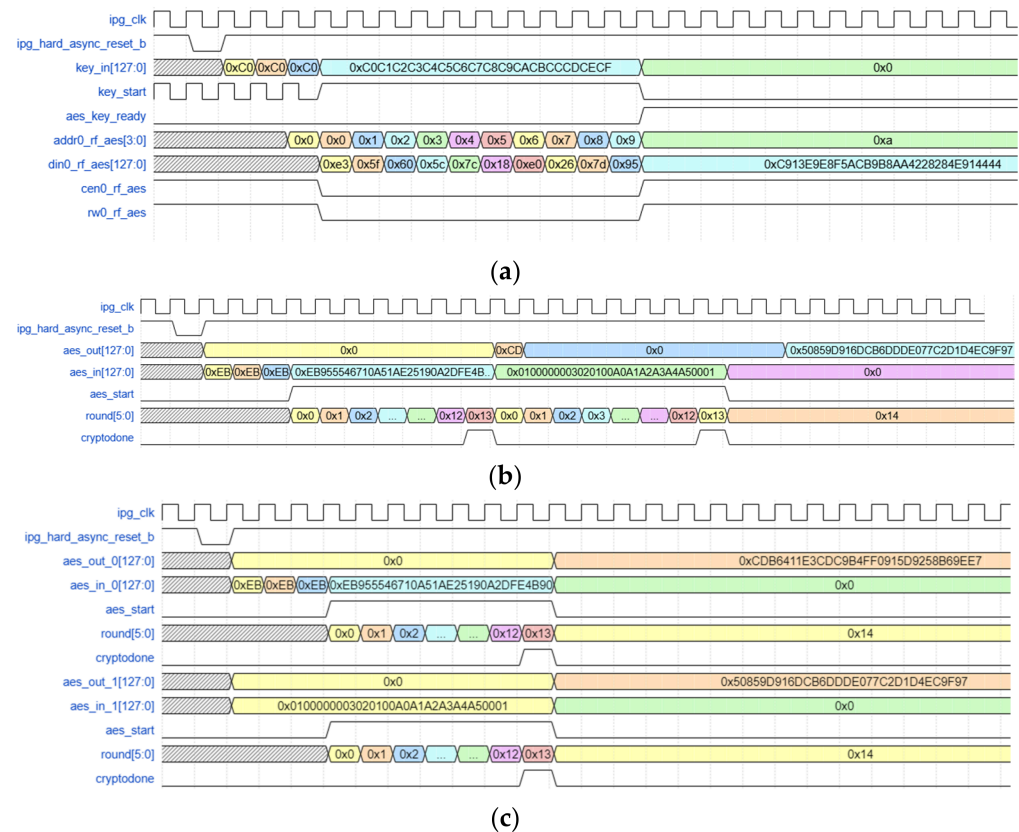


Figure 4. Experimental results of AES serial and parallel encryption. (a) Simulation results of storing round keys in RAM for AES key expansion. (b) Simulation results of serial AES encryption scheme. (c) Simulation results of parallel AES encryption scheme.

The initialization phase consists of key expansion and RAM write operations, which last for 14 clock cycles in total. Among them, the input of the 128-bit primary key occupies four clock cycles. In practical embedded secure applications, rekeying occurs rarely. Thus, the overhead is negligible and can be amortized over long data packets. Even in scenarios with frequent rekeying, the performance impact remains low.

To thoroughly verify the correctness of our AES-128-CCM coprocessor, we performed the following tests:

- NIST KATs: All 128-bit AES-128 cipher and inverse cipher test vectors from NIST SP 800-38A [23] and SP 800-38C [24] were simulated. The RTL outputs matched the expected vectors in all 128 test cases.
- Boundary cases: An all-zero key + all-zero plaintext, an all-one key + an all-one plaintext, and alternating patterns (0x55, 0xAA) were tested. All passed.
- RFC 3610 CCM test vectors: All test vectors for AES-128-CCM defined in RFC 3610 [25] were applied to our RTL model. For each vector, the computed ciphertext and MAC exactly matched the RFC values.
- Random test against OpenSSL: We generated 100,000 random (key, nonce, plaintext, associated data) sets. Each set was encrypted by both our RTL simulator and OpenSSL 3.0.0 [26]. The results were compared byte-by-byte; all 100,000 tests passed.

- Code coverage: A total of 100,000 groups of random test vectors were used for cross-check with OpenSSL-3.0.0. The verification covers CBC, CTR and full CCM modes with complete functional coverage. The RTL statement coverage and branch coverage both reached 99.7% via NC-Verilog simulation."

5.4. Synthesis Results and Performance Comparison

Additionally, the standard AES-CCM algorithm from the open-source software OpenSSL-3.0.0 was utilized to conduct extensive comparisons with random test data for the AES-CCM hardware module, verifying the correctness of the overall module. Meanwhile, the AES-CCM module was synthesized using Design Compiler software in the TSMC 40 nm eFlash process. At a clock frequency of 200 MHz, Cui et al.'s parallel implementation of the AES-CCM scheme resulted in a circuit area of 23.45 KGE with a power consumption of 4.05 mW. The parallel implementation of the AES-CCM scheme described in this proposal, after synthesis, resulted in a circuit area of 26.45 KGE with an average power consumption of 1.92 mW. For the serial implementation, Cui et al.'s scheme yielded a circuit area of 16.45 KGE with a power consumption of 2.03 mW, whereas the AES-CCM scheme described in this proposal resulted in a circuit area of 19.35 KGE with a power consumption of 1.15 mW. The RAM leakage power is 0.11 μ W, which is already accounted for in the overall average power of our proposed solution.

For the dual-core parallel scheme, each 128-bit plaintext block requires four clock cycles for input via the 32-bit system bus, and another four clock cycles for 128-bit ciphertext output. AES-128 involves 10 rounds of encryption operations. Each round takes two clock cycles owing to the reuse of eight S-boxes, so the core cryptographic computation consumes 20 clock cycles. Taking data input and output latency into account, the total number of clock cycles for processing one 128-bit data block is 28. Operating at a clock frequency of 200 MHz, the achievable throughput is 0.91 Gbps. For the single-core serial scheme, AES-CCM needs to perform both CBC and CTR operations for a single 128-bit data block. The overall processing latency is 48 clock cycles, including four cycles for data input, four cycles for data output, and 40 cycles for core encryption operations. Correspondingly, the throughput is 0.53 Gbps. Since Ref. [13] does not provide the throughput, it is impossible to calculate the energy efficiency. The energy efficiency of our design is 2.11 pJ/bit (parallel) and 2.17 pJ/bit (serial). A comparison of the area and power consumption data for both schemes is presented in Table 2.

Table 2. Data comparison between our proposal and traditional approaches in implementing AES-CCM encryption protocol.

Design	AES (Core(s))	Speed (MHz)	Throughput (Gbps)	Platform	Area (KGE)	Power (mW)
Ref. [8] Parallel two AES	2	264	2.69	90 nm	N/A	N/A
Ref. [13] Parallel two AES	2	200	N/A	65 nm	23.45	4.05
Ref. [13] Serial one AES	1	200	N/A	65 nm	16.45	2.03
Ref. [27] Parallel four AES	4	200	4.23	65 nm	46.51	8.62
Ref. [27] Parallel five AES	5	200	4.23	65 nm	39.59	6.89
Ref. [28] Parallel two AES	2	10	0.054	250 nm	14.9	4.37
Our design of parallel two AES	2	200	0.91	40 nm	26.45	1.92
Our design of serial one AES	1	200	0.53	40 nm	19.35	1.15

To ensure a fair comparison, we normalize the dynamic power to the same process node. According to the quasi-constant electric field (QCE) scaling rule [29], the dynamic power reduction from 65 nm to 40 nm is approximately 50% [30], so we adopt a scaling factor of 0.5 for normalization. The normalized power consumption of Ref. [13] is 2.03 mW (parallel) and 1.02 mW (serial). With additional algorithm-level security countermeasures integrated, our parallel scheme consumes 5.4% less power than the design in Ref. [13], while the serial scheme consumes 12.7% more power, despite integrating explicit side-channel countermeasures (masking). Our design is the only one among the compared works with such side-channel protection, as listed in Table 3. We adopt a dynamic power scaling factor of 0.031 for designs migrated from the 250 nm process to the 40 nm process [30].

Table 3. Normalized data comparison between our proposal and traditional approaches in implementing AES-CCM encryption protocol.

Design	Throughput (Gbps)	SCA Counter-Measure	Norm. Area (μm^2)	Norm. Power (mW)	Norm. Throughput (Gbps/KGE)	Norm. Energy Efficiency (pJ/bit)
Ref. [8] Parallel two AES	2.69	None	N/A	N/A	N/A	N/A
Ref. [13] Parallel two AES	N/A	None	16,649.5	2.03	N/A	N/A
Ref. [13] Serial one AES	N/A	None	11,679.5	1.02	N/A	N/A
Ref. [27] Parallel four AES	4.23	None	33,022.1	4.31	0.091	1.02
Ref. [27] Parallel five AES	4.23	None	28,108.9	3.45	0.107	0.82
Ref. [28] Parallel two AES	0.054	None	10,579	0.14	0.004	2.59
Our design of parallel two AES	0.91	Masking	18,779.5	1.92	0.034	2.11
Our design of serial one AES	0.53	Masking	13,738.5	1.15	0.027	2.17

It should be noted that the power normalization between different technology nodes in this work follows the quasi-constant electric field scaling rule, which assumes ideal Dennard scaling. However, Esmaeilzadeh et al., in their seminal work on “Dark Silicon,” explicitly pointed out that the reduced rate of voltage scaling, coupled with reduced transistor capacitance scaling, poses a severe power challenge for continued multicore scaling [31]. Their study further demonstrated that due to the inability of voltage to scale proportionally with transistor dimensions, Dennard scaling has effectively broken down, making power constraints a fundamental bottleneck for multicore scaling. This trend implies that when scaling from 65 nm to more advanced nodes such as 40 nm, the actual reduction in dynamic power is significantly lower than the 50% predicted by ideal Dennard scaling. The stagnation of voltage scaling directly limits the reduction in dynamic power rendering traditional scaling factors incapable of accurately reflecting actual power variations. Therefore, this work adopts a scaling factor of 0.5 from 65 nm to 40 nm as a conservative first-order engineering approximation; the actual power reduction is likely to be considerably smaller than this ideal value.

To fairly compare hardware overhead across different technology nodes, all gate counts must be normalized to a unified reference process by multiplying the original KGE value with the NAND2 area of the target reference process [32]. According to the standard cell library data provided by the foundry for the 40nm technology node, the NAND2 gate area is $0.71 \mu\text{m}^2$. Based on the NAND2 gate area data provided, we can normalize the area to the 40 nm technology node, as presented in Table 3.

To provide further insight into the power distribution, Table 4 details the power consumption breakdown across the major building blocks of the coprocessor. The AES encryption cores, including S-box and data path logic, dominate the total power. The power

data for the logic blocks are obtained from logic synthesis using Design Compiler, whereas the RAM power is derived from the memory compiler provided by the foundry under specific configurations. The combined results are presented in Table 4.

Table 4. Power breakdown of the proposed AES-128-CCM coprocessor.

Component	Parallel Mode (mW)	Serial Mode (mW)
AES Encryption Cores	1.31	0.76
Masking Overhead	0.35	0.18
RAM	0.09	0.09
AHB Interface and DMA Logic	0.17	0.12
Total Power	1.92	1.15

Our design occupies an area of 3.0 KGE more than Ref. [13] for the parallel mode and 2.9 KGE more for the serial mode, corresponding to an area increment of 12.8% and 17.6%, respectively. This increase is primarily due to the 1.25 Kb RAM used for one-time key expansion. Under TSMC 40 nm, this RAM occupies approximately 4.2 KGE (based on memory compiler data). Excluding the RAM, the logic area of our design is 22.25 KGE (parallel) and 15.15 KGE (serial), which are 5.1% and 7.9% smaller than the corresponding logic areas in Ref. [13]. Thus, the expected area scaling from 65 nm to 40 nm is indeed achieved for the logic core.

To address the initialization amortization argument, a packet size breakeven analysis is provided based on the measured synthesis data. The one-time initialization phase consisting of key expansion and RAM write operations consumes 14 clock cycles, which corresponds to 70 ns at the operating frequency of 200 MHz. Assuming an active power of 1.92 mW in parallel mode, the one-time initialization energy cost is approximately 134.4 pJ. According to the synthesis data, our design reduces dynamic power consumption by 0.48 mW during encryption by removing repetitive key expansion operations. For a single 128-bit block taking 28 cycles and 140 ns, the dynamic energy saved is 67.2 pJ. Thus, the breakeven point occurs after processing just two data blocks, where the cumulative dynamic energy savings of 134.4 pJ surpass the one-time initialization cost. For larger payloads, the benefits multiply. Taking a typical 4 KB payload as an example, the parallel mode takes 36 μ s to process and saves 17.3 nJ of dynamic energy, while the serial mode takes 61.8 μ s and saves 29.7 nJ. In contrast, the additional static leakage energy introduced by the 1.25 Kb RAM with a leakage power of 0.11 μ W during the same 4 KB encryption periods is only 3.96 pJ (parallel) and 6.80 pJ (serial). Therefore, the dynamic energy savings are more than three orders of magnitude larger than the additional leakage energy, confirming that the initialization overhead is rapidly amortized even for moderate packet sizes, yielding substantial net energy savings for continuous IoT secure communication.

Compared with Ref. [27], although it achieves higher normalized throughput and better normalized energy efficiency, its normalized power consumption is larger than that of our design. Therefore, our scheme demonstrates a strong competitive advantage in low-power application scenarios. For Ref. [28], it features lower normalized power, while its normalized throughput and normalized energy efficiency are obviously inferior to those of the proposed design.

6. Conclusions

This paper leverages the characteristic of the AES-CCM encryption protocol where the same key is used for both data checksum generation and data confidentiality and authentication. As a result, the key expansion operation needs to be performed only once

during the initialization phase, rather than being repeated for every encryption block. Additionally, based on the structure of the AES algorithm, the S-box hardware circuit reuse mechanism is adopted. This paper presents a low-power coprocessor implementation scheme for AES-128-CCM, which includes both parallel and serial dual-module AES configurations. In the parallel implementation scheme, the number of S-boxes is reduced by 24 compared with traditional schemes, and repeated key expansion operations are removed. In the serial implementation scheme, the number of S-boxes is reduced by 12, and the key expansion module computation is also performed only once. This design reduces the power consumption of the AES-CCM module while maintaining high performance for demanding scenarios, providing an efficient hardware solution for the AES-CCM protocol.

Design Limitations: Although the proposed coprocessor achieves notable improvements in low-power design, the following limitations remain to be addressed in future work. First, the proposed design supports AES-128 only. Supporting longer key lengths would require a complete redesign of the RAM architecture, since AES-192 requires 12 round keys while AES-256 requires 14 round keys, which would increase storage overhead and conflict with our low-power design objectives. Second, the algebraic masking scheme adopted in this work is first-order only, providing effective resistance against first-order DPA and CPA attacks but insufficient protection against higher-order differential power analysis. While higher-order masking can provide stronger security guarantees, its significant area and power overhead make it difficult to deploy directly in resource-constrained IoT devices. Third, the side-channel security evaluations in this work, including both TVLA and CPA tests, are based entirely on simulated power traces using a Hamming-weight model and have not yet been validated on fabricated silicon. Although the simulation results provide an initial security indication, they cannot fully replace physical measurements on actual hardware. Fourth, no dedicated countermeasures against fault-injection attacks, including clock glitches, voltage glitches, and laser illumination, are integrated, although the per-round random masking provides a certain degree of inherent resistance; however, security risks may still exist when facing attackers with physical access capabilities.

Future Work Directions: To address the above limitations, future work will focus on the following directions. First, silicon fabrication and physical validation are the most important steps. After tape-out in the TSMC 40 nm process, comprehensive side-channel security evaluations will be conducted in a real physical environment, including TVLA tests and CPA attacks based on actual power traces, to confirm the validity of the simulation results and calibrate the effectiveness of the masking scheme on real silicon. Second, we will explore reconfigurable architectures supporting AES-192 and AES-256, investigating how to support multiple key lengths through optimized RAM organization and control logic without significantly increasing power and area. Third, we will investigate lightweight implementation techniques for higher-order masking targeting resource-constrained IoT devices, exploring strategies such as composite-field mask reuse and random number sharing to reduce the overhead of higher-order masking, aiming for a better balance between security and power-area trade-offs. Fourth, we plan to integrate lightweight sensors, including voltage monitors and frequency monitors, together with fault detection logic to enhance resistance against fault-injection attacks without significantly increasing power consumption. Finally, we will extend the low-power optimization methodology developed in this work to other authenticated encryption modes such as AES-GCM to verify its generality and reusability.

Author Contributions: Conceptualization, J.-Q.W.; Methodology, J.-Q.W.; Validation, J.-Q.W., Y.-C.L. and W.Z.; Formal analysis, J.-Q.W., Y.-C.L. and W.Z.; Investigation, J.-Q.W., Y.-C.L. and W.Z.; Resources, J.-Q.W.; Data curation, J.-Q.W., Y.-C.L. and W.Z.; Writing—original draft, J.-Q.W.; Writing—review & editing, J.-Q.W. and H.-L.L.; Visualization, H.-L.L.; Supervision, H.-L.L.; Project administration, H.-L.L.; Funding acquisition, H.-L.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. IEEE Std 802.11i-2004; IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Computer Society: New York, NY, USA, 2004.
2. Morioka, S.; Satoh, A. An optimized S-box circuit architecture for low power AES design. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2523, pp. 172–186.
3. Satoh, A.; Morioka, S.; Takano, K.; Munetoh, S. A compact Rijndael hardware architecture with S-box optimization. In *Advances in Cryptology: ASIACRYPT 2001*; Boyd, C., Ed.; Springer: Berlin, Germany, 2001; Volume 2248, pp. 239–254.
4. Wolkerstorfer, J.; Oswald, E.; Lamberger, M. An ASIC implementation of the AES SBoxes. In *Topics in Cryptology: CT-RSA 2002*; Preneel, B., Ed.; Springer: Berlin, Germany, 2002; Volume 2271, pp. 67–78.
5. Ahmad, N.; Hasan, R.; Jubadi, W.M. Design of AES S-box using combinational logic optimization. In *2010 IEEE Symposium on Industrial Electronics and Applications (ISIEA)*; IEEE: New York, NY, USA, 2010; pp. 696–699.
6. Mui, E.N. *Practical Implementation of Rijndael S-Box Using Combinational Logic*; Custom R&D Engineer, Texco Enterprise Pvt. Ltd.: Delhi, India, 2007.
7. Boyar, J.; Peralta, R. A new combinational logic minimization technique with applications to cryptology. In *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA)*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6049, pp. 178–189.
8. Nguyen, K.; Lanante, L.; Nagao, Y.; Kurosaki, M.; Ochi, H. Implementation of 2.6 Gbps super-high speed AES-CCM security protocol for IEEE 802.11i. In *Proceedings of the 13th International Symposium on Communications and Information Technologies (ISCIT)*; IEEE: New York, NY, USA, 2013; pp. 669–673.
9. Pammu, A.A.; Ho, W.G.; Lwin, N.K.Z.; Chong, K.S.; Gwee, B.H. A high throughput and secure authentication-encryption AES-CCM algorithm on asynchronous multicore processor. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1023–1036. [[CrossRef](#)]
10. Hoang, V.P.; Phan, T.T.; Dao, V.L.; Pham, C.K. A compact, ultra-low power AES-CCM IP core for wireless body area networks. In *Proceedings of the IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*; IEEE: New York, NY, USA, 2016; pp. 1–4.
11. Shahbazi, K.; Ko, S.B. Area-efficient nano-AES implementation for internet-of-things devices. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2021**, *29*, 136–148. [[CrossRef](#)]
12. Pandey, B.; Bisht, V.; Hussain, D.M.A.; Jamil, M.; Hasan, M.K. Energy-efficient implementation of AES algorithm on 16nm FPGA. In *Proceedings of the 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)*; IEEE: New York, NY, USA, 2021; pp. 740–744.
13. Cui, C.; Zhao, Y.; Xiao, Y.; Lin, W.B.; Zeng, Y.G. Optimized design and implementation of a universal AES-CCM coprocessor. *J. Cryptologic Res.* **2021**, *8*, 834–843.
14. Nannipieri, P.; Marino, F.; Fanucci, L.; Saponara, S. CRFlex: A flexible and configurable cryptographic hardware accelerator for AES block cipher modes. In *Applications in Electronics Pervading Industry, Environment and Society (ApplePies 2021)*; Saponara, S., De Gloria, A., Eds.; Springer: Cham, Switzerland, 2022; pp. 59–74.
15. Sai, T.V.; Balasubramanian, K.; Yamuna, B. Design of a high-speed and low-power AES architecture. In *Proceedings of the 4th International Conference on Communication, Computing and Electronics Systems*; Bindhu, V., Tavares, J.M.R.S., Vuppapapati, C., Eds.; Springer: Singapore, 2023; pp. 421–430.
16. Kala, J.; Panda, J.; Tanwar, L. FPGA implementation of a high throughput low power advanced encryption standard (AES-128) cipher. In *Proceedings of the 10th International Conference on Signal Processing and Integrated Networks (SPIN)*; IEEE: New York, NY, USA, 2023; pp. 367–372.
17. Rivain, M.; Prouff, E. Provably secure higher-order masking of AES. In *Cryptographic Hardware and Embedded Systems: CHES 2010*; Mangard, S., Standaert, F.-X., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6225, pp. 413–427.

18. Canright, D. A very compact S-box for AES. In *Cryptographic Hardware and Embedded Systems: CHES 2005*; Rao, J.R., Sunar, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3659, pp. 441–455.
19. Coron, J.-S. Higher order masking of look-up tables. In *Advances in Cryptology: EUROCRYPT 2014*; Nguyen, P.Q., Oswald, E., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8441, pp. 441–458.
20. *FIPS PUB 140-2*; Security Requirements for Cryptographic Modules. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2002. Available online: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf> (accessed on 6 August 2025).
21. Goodwill, G.; Jun, B.; Jaffe, J.; Rohatgi, P. A testing methodology for side-channel resistance validation. In *NIST Non-Invasive Attack Testing Workshop*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2011; pp. 115–136.
22. Schneider, T.; Moradi, A. Leakage assessment methodology: A clear roadmap for side-channel evaluations. In *Cryptographic Hardware and Embedded Systems: CHES 2016*; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9813, pp. 495–513.
23. *NIST Special Publication 800-38A*; Recommendation for Block Cipher Modes of Operation: Methods and Techniques. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001.
24. *NIST Special Publication 800-38C*; Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. National Institute of Standards and Technology: Gaithersburg, MD, USA, 2004.
25. Whiting, D.; Housley, R.; Ferguson, N. Counter with CBC-MAC (CCM). RFC 3610, Internet Engineering Task Force. September 2003. Available online: <https://datatracker.ietf.org/doc/html/rfc3610> (accessed on 6 August 2025).
26. Hudson, M.; Henson, S.; Levitte, S.; Salz, R.; Gal, V. OpenSSL 3.0.0: Open source cryptography toolkit. The OpenSSL Project, Release 3.0.0. September 2021. Available online: <https://www.openssl.org/source/openssl-3.0.0.tar.gz> (accessed on 6 August 2025).
27. Cui, C.; Zhao, Y.; Xiao, Y.; Lin, W.B.; Xu, D. High-throughput and low-power AES-CCM hardware implementation. *J. Cryptologic Res.* **2021**, *8*, 365–375.
28. Huai, L.; Zou, X.; Liu, Z.; Han, Y. An energy-efficient AES-CCM implementation for IEEE802.15.4 wireless sensor networks. In *Proceedings of the International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC)*; IEEE Computer Society: Los Alamitos, CA, USA, 2009; pp. 394–397.
29. Dennard, R.H.; Gaensslen, F.H.; Rideout, V.L.; Bassous, E.; LeBlanc, A.R. Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE J. Solid-State Circuits* **1974**, *9*, 256–268. [[CrossRef](#)]
30. Rabaey, J.M.; Chandrakasan, A.; Nikolic, B. *Digital Integrated Circuits: A Design Perspective*, 2nd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2003.
31. Esmaeilzadeh, H.; Blem, E.; St. Amant, R.; Sankaralingam, K.; Burger, D. Dark silicon and the end of multicore scaling. In *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA)*; IEEE: Piscataway, NJ, USA, 2011; pp. 365–376.
32. Kaeslin, H. *Digital Integrated Circuit Design: From VLSI Architectures to CMOS Fabrication*; Cambridge University Press: Cambridge, UK, 2008.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.