A Survey on FPGA Implementation of Modified Lightweight 128-Bit AES Algorithm for IoT Applications

Chandrakant[1] , Dr.Sharanagouda N[2], Dr.Anuradha Savadi[3]

1.Student, *Dept of Electronics and Communication Engineering, Sharnbasva University, Kalaburagi, Karnataka, India.*

2.Professor, Dept of Electronics and Communication Engineering, Sharnbasva University, Kalaburagi Karnataka, India.

3.Associate Professor, Dept of Electronics and Communication Engineering, Sharnbasva University, Kalaburagi Karnataka, India.

1. ABSTRACT

The rapid proliferation of Internet of Things (IOT) devices has intensified the demand for secure yet resource-efficient cryptographic solutions. Although the Advanced Encryption Standard (AES) remains the dominant symmetric encryption algorithm due to its strong security guarantees, its conventional implementation imposes considerable computational and power overhead, making it less suitable for constrained IOT nodes. Lightweight modifications of AES have therefore been proposed to balance security, performance, and hardware efficiency.

This survey presents a comprehensive review of FPGA-based implementations of modified lightweight 128-bit AES architectures tailored for IOT applications. It analyzes architectural optimizations, algorithmic modifications, hardware-aware design strategies, and performance trade-offs. Particular emphasis is placed on reduced-round encryption, optimized substitution and diffusion mechanisms, Galois field simplification, and resource-sharing techniques. The study highlights how FPGA platforms enable flexible, scalable, and power-efficient deployment of lightweight AES designs while maintaining acceptable cryptographic strength.

Keywords – Lightweight AES, FPGA cryptography, IOT security, hardware optimization, low-power encryption, RTL implementation

2. INTRODUCTION

The rapid expansion of the Internet of Things (IOT) has transformed modern communication systems by enabling billions of interconnected devices to exchange data autonomously. Applications such as smart homes, healthcare monitoring, industrial automation, environmental sensing, and wearable technology rely heavily on continuous data transmission across wireless networks. While this interconnected ecosystem enhances convenience and automation, it simultaneously exposes devices to security threats including data interception, unauthorized access, and malicious manipulation. Ensuring secure communication in IOT environments has therefore become a fundamental design requirement.

Among various cryptographic algorithms, the Advanced Encryption Standard (AES) has emerged as the most widely adopted symmetric key encryption scheme due to its strong security guarantees, mathematical robustness, and global standardization. AES-128, in particular, operates on 128-bit data blocks using ten iterative rounds of substitution and permutation transformations. These rounds consist of SubBytes, ShiftRows, MixColumns, and AddRoundKey operations, collectively providing high levels of confusion and diffusion. Although AES delivers excellent cryptographic strength, its standard implementation demands

significant computational resources, memory bandwidth, and power consumption. Such requirements can become prohibitive for resource-constrained IOT devices, which typically operate on limited battery power and possess restricted hardware capabilities.

IOT nodes are often designed with low-cost microcontrollers or small FPGA platforms that have limited logic elements, memory blocks, and energy availability. In these environments, conventional AES architectures may lead to increased silicon area, excessive switching activity, and reduced operational lifetime. Consequently, there is a growing need to develop lightweight cryptographic architectures that maintain acceptable security levels while reducing hardware complexity and energy consumption.

Lightweight AES variants aim to optimize internal transformations of the standard algorithm rather than redesigning the cipher entirely. Techniques such as round reduction, simplified MixColumns implementation, optimized S-box realization, resource sharing, and hardware-aware scheduling have been explored to reduce computational overhead. These modifications target the most hardware-intensive components of AES while preserving its essential security properties. When carefully designed, such lightweight adaptations can achieve significant reductions in logic utilization, processing time, and dynamic power without critically weakening resistance to common cryptanalytic attacks.

Field Programmable Gate Arrays (FPGAs) provide an attractive platform for implementing lightweight AES architectures. Unlike fixed ASIC designs, FPGAs offer flexibility, reconfigurability, and parallel processing capabilities, making them suitable for prototyping and deployment in diverse IOT systems. Furthermore, FPGA-based implementations allow designers to explore iterative, pipelined, or resource-shared architectures to achieve optimal trade-offs between throughput, area, and power.

This project focuses on the FPGA implementation of a modified lightweight 128-bit AES algorithm tailored for IOT applications. The objective is to reduce hardware resource utilization and power consumption while ensuring reliable encryption and decryption functionality. By leveraging architectural optimization techniques and hardware-efficient design strategies, the proposed implementation demonstrates that secure and energy-efficient encryption can be achieved within the constraints of embedded IOT platforms.

3. FUSION METHODS

- **Reduced-Round AES Architecture**

  Reduced-round AES architecture was discussed in "[1]" for minimizing computational overhead and area consumption in FPGA implementations targeting resource-constrained IOT platforms. In this approach, the standard 10 encryption rounds of AES-128 are reduced to 6-8 rounds, which decreases the number of repetitive substitution and permutation operations like SubBytes, ShiftRows, and MixColumns. As a result, logic switching activity, critical path delay, and overall power consumption are significantly reduced, making the design suitable for battery-powered and low-cost FPGA devices. However, reducing the number of rounds also lowers the *cryptographic security margin*, as fewer rounds offer less diffusion and confusion, potentially making the cipher more vulnerable to differential and linear cryptanalysis. Therefore, reduced-round implementations often need additional architectural safeguard, such as stronger nonlinear components or enhanced diffusion mapping, to compensate for the reduced round count while maintaining acceptable resistance against attacks.

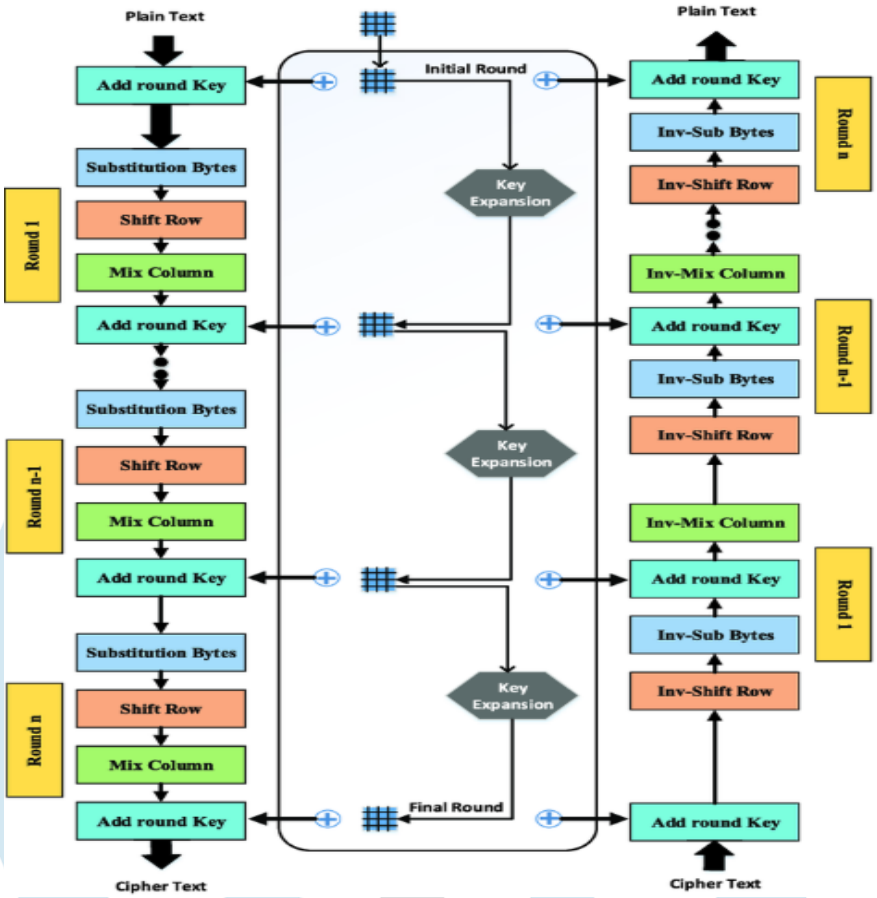

Figure 3.1: AES Encryption and Decryption steps

- **Composite Field S-Box Optimization**

Composite field S-box optimization was presented in "[2]" to reduce the hardware complexity of the SubBytes transformation in FPGA-based AES implementations. In this method, the multiplicative inversion operation in **GF(2^8)** is mathematically decomposed into smaller subfield operations such as **GF((2^4)^2)** or **GF(((2^2)^2)^2)**. By breaking the complex inversion into simpler arithmetic stage - consisting mainly of XOR gates and small multipliers, the overall combinational logic required for the S-box is significantly reduced. Unlike lookup-table (BRAM) based implementations, composite field S-boxes are implemented using optimized logic expressions, which lowers memory usage and improves portability across low-cost FPGA devices.

This technique reduces LUT utilization, decreases routing congestion, and can enhance maximum operating frequency when carefully pipelined. However, improper subfield mapping may increase propagation delay due to deeper combinational paths. Therefore, designers often apply logic minimization, retiming, or partial pipelining to balance area savings with timing performance.

- **Logic-Minimized S-Box Implementation**

Logic-minimized S-box implementation was discussed in "[3]" to replace conventional BRAM-based lookup tables with optimized Boolean logic expressions derived from algebraic decomposition of the AES substitution function. Instead of storing 256 precomputed values in memory, the S-box transformation is expressed using minimized combinational logic equations composed primarily of XOR, AND, and NOT gates. This eliminates block RAM dependency and significantly reduces FPGA BRAM utilization, which is beneficial for low-cost or small-scale FPGA devices used in IOT systems.

The main advantage of this method is improved hardware portability and predictable timing behavior, as the S-box becomes fully logic-based and independent of memory access latency. It also reduces memory power consumption since no BRAM reads are required during encryption. However, achieving an optimal logic-minimized implementation requires extensive algebraic simplification, gate-level optimization, and sometimes manual restructuring to prevent excessive combinational depth. Designers often apply logic balancing and retiming techniques to ensure that propagation delay does not limit the achievable clock frequency.

- **Optimized MixColumns Using Shift-XOR Logic**

Optimized MixColumns implementation was explored in "[4]" to replace conventional Galois Field multipliers with shift-and-XOR based logic operations in FPGA architectures. In AES, multiplication by fixed constants (such as 02 and 03 in $GF(2^8)$) can be implemented using simple left-shift and conditional XOR operations instead of full multipliers. By eliminating complex GF multipliers and avoiding DSP slice usage, the hardware area is significantly reduced. This approach lowers LUT count, minimizes routing complexity, and reduces dynamic power consumption due to decreased switching activity.

The shift-XOR method maintains the original diffusion strength of the MixColumns transformation while offering a more hardware-friendly implementation for lightweight IOT devices. However, designers must carefully structure the combinational logic to prevent long critical paths that may reduce maximum operating frequency. Pipelining or logic balancing is often applied to optimize timing performance.

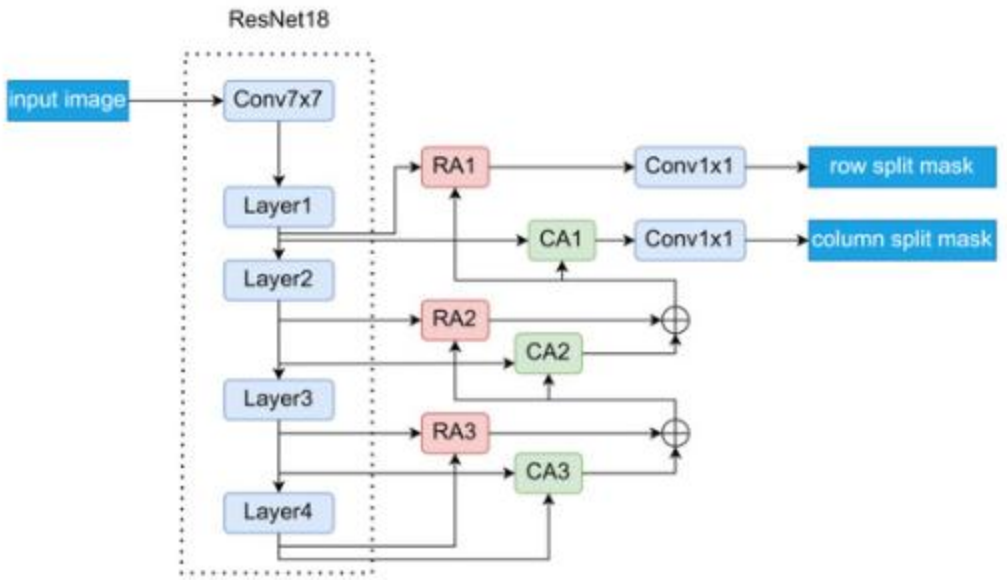

Figure 3.2: RCANet network Structure

- **Iterative AES Round Architecture**

Iterative AES round architecture was introduced in "[5]" to achieve area-efficient FPGA implementation suitable for resource-constrained IOT devices. In this architecture, instead of instantiating separate hardware blocks for all 10 AES rounds, a **single round processing unit** is reused sequentially under the control of a Finite State Machine (FSM). The FSM schedules the execution of SubBytes, ShiftRows, MixColumns, and AddRoundKey operations for each round in successive clock cycles.

By reusing the same combinational logic for all rounds, this approach significantly reduces LUT, flip-flop, and routing resource utilization on FPGA devices. It also lowers dynamic power consumption due to reduced parallel switching activity. The iterative design is particularly beneficial for small and mid-range FPGAs used in IOT nodes where area and power efficiency are more critical than ultra-high throughput. However, since the same hardware is reused multiple times, the total encryption latency increases proportionally with the number of rounds. While throughput decreases compared to fully pipelined or fully unrolled architectures, iterative AES cores provide an excellent trade-off between hardware cost and performance for low-data-rate IOT applications.

- **Fully Pipelined AES Design**

Fully pipelined AES architecture was discussed in "[6]" to achieve high throughput in FPGA-based cryptographic systems. In this architecture, each AES round is implemented as a separate hardware stage, and pipeline registers are inserted between consecutive rounds. This allows multiple plaintext blocks to be processed simultaneously, with each block residing in a different pipeline stage during a given clock cycle. As a result, once the pipeline is filled, one encrypted output block can be produced every clock cycle, significantly increasing throughput. This design improves maximum operating frequency due to reduced combinational depth per stage and enables efficient parallel processing suitable for high-speed communication systems. However, fully pipelined architectures require replication of round hardware units, leading to increased LUT, flip-flop, and routing resource utilization. Consequently, power consumption also rises due to higher parallel switching activity. Therefore, fully pipelined AES designs are more appropriate for performance-critical FPGA applications rather than ultra-low-power IOT nodes.

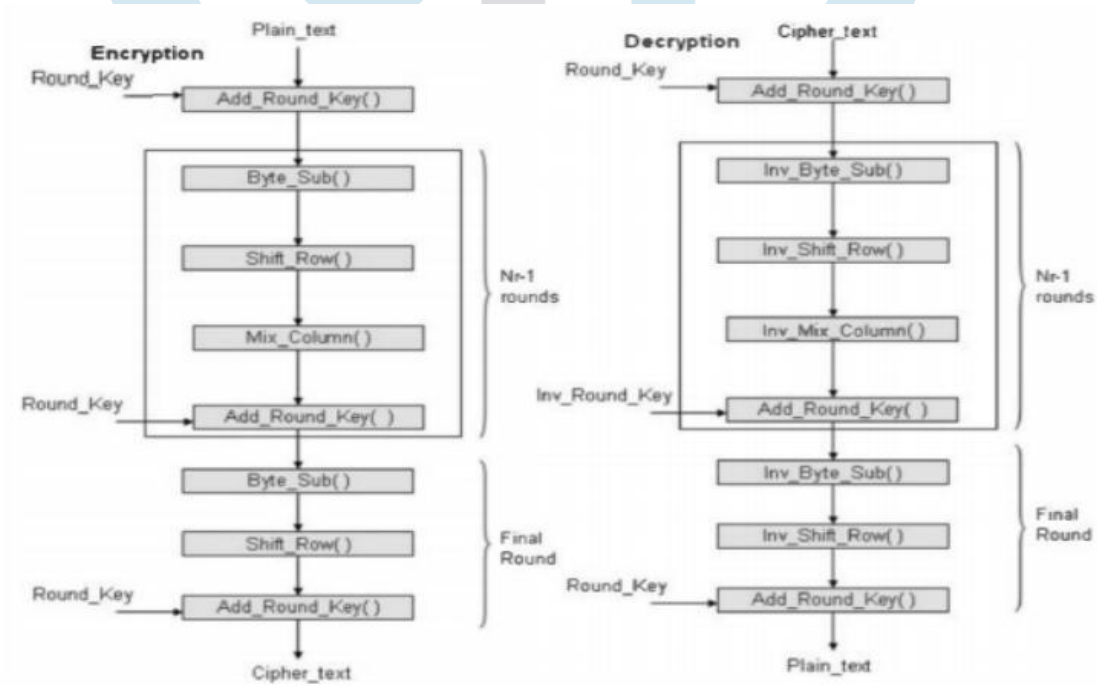

Figure 3.3: AES algorithm

- **Hybrid Iterative-Pipeline Architecture**

Hybrid iterative-pipeline architecture was proposed in "[7]" to balance throughput performance and hardware resource utilization in FPGA-based AES implementations. In this approach, the architecture combines the advantages of both iterative and pipelined designs. Critical transformations such as SubBytes and MixColumns, which often dominate the critical path delay, are partially pipelined to improve maximum operating frequency. Meanwhile, less timing-sensitive stages, such as

AddRoundKey or certain control operations, reuse hardware resources across rounds under FSM control.

This selective pipelining reduces combinational delay and increases throughput compared to a purely iterative design, while avoiding the full hardware replication required in fully pipelined architectures. As a result, the hybrid model achieves moderate speed improvement without significant area overhead. Such architectures are particularly suitable for mid-range FPGA platforms deployed in IOT gateways where both performance and area efficiency must be balanced. However, careful synchronization and stage partitioning are necessary to prevent pipeline hazards and maintain correct round sequencing.

- **On-the-Fly Key Expansion**

On-the-fly key scheduling was discussed in "[8]" to eliminate the need for storing all precomputed round keys in memory. In conventional AES implementations, round keys are generated in advance and stored in BRAM or registers, which increases memory utilization and hardware overhead. In the on-the-fly approach, round keys are generated dynamically during each encryption round and immediately used by the AddRoundKey stage. This removes the requirement for large key storage blocks and reduces BRAM consumption, which is highly advantageous for lightweight IOT FPGA implementations with limited memory resources.

This method improves area efficiency and lowers static memory power consumption. However, key generation must be carefully synchronized with the round execution process to avoid latency bottlenecks. Designers often integrate the key expansion module within the main datapath and control it using FSM-based scheduling to ensure seamless operation without affecting throughput.

- **Bit-Serial AES Implementation**

Bit-serial AES implementation was explored in "[9]" to achieve ultra-low hardware area in FPGA-based cryptographic designs. In this approach, instead of processing data byte-wise (8 bits at a time) or in full 128-bit parallel blocks, the encryption core processes **one bit per clock cycle**. Core AES transformations such as SubBytes, ShiftRows, MixColumns, and AddRoundKey are restructured to operate on serial bit streams using minimal combinational logic and small shift registers.

This architecture drastically reduces LUT count, flip-flop usage, and routing complexity, making it highly suitable for extremely resource-constrained IOT devices and small FPGA platforms. Because fewer logic elements switch simultaneously, dynamic power consumption becomes significantly lower compared to parallel architectures. However, the trade-off is substantially reduced throughput, as many clock cycles are required to complete a single 128-bit encryption operation. Therefore, bit-serial AES is ideal for low-data-rate IOT applications where area and energy efficiency are more critical than speed.

- **Byte-Serial Architecture**

Byte-serial AES design was presented in "[10]" as a compromise between bit-serial and fully parallel architectures for FPGA AES implementations. In this approach, the cipher processes one byte (8 bits) per clock cycle**,** rather than a single bit or the full 128-bit block at once. This reduces the amount of combinational logic and routing resources required compared to parallel designs, while still achieving better performance than pure bit-serial implementations. Byte-serial processing allows a reasonable balance of area and throughput, making it well-suited for lightweight IOT cryptographic cores where both hardware cost and encryption speed are important. Reducing the data path width lowers LUT and flip-flop usage, while still enabling encryption rates adequate for many low-data-rate IOT applications. However, overall latency increases compared to fully parallel designs due to the need for multiple cycles per encryption.

- **Clock Gating for Power Reduction**

  Clock gating technique was discussed in "[11]" to reduce dynamic power consumption in FPGA-based AES cores used in IOT applications. In this method, the clock signal supplied to inactive functional blocks—such as SubBytes, MixColumns, or key expansion modules—is temporarily disabled when those blocks are not in operation. Since dynamic power in digital circuits is directly proportional to switching activity and clock frequency, preventing unnecessary clock toggling significantly lowers power consumption.

  In FPGA implementations, clock gating is often achieved using enable-controlled registers or clock-enable signals instead of physically gating the clock line, ensuring safe synthesis and timing behavior. This technique is particularly beneficial in IOT deployments where encryption tasks occur intermittently, allowing inactive modules to remain idle and conserving battery energy. However, additional control logic is required to manage clock-enable signals without affecting functional correctness.

- **Operand Isolation Technique**
  Operand isolation was presented in "[12]" to minimize unnecessary switching activity. By isolating input signals when computations are not required, dynamic power is reduced. This method enhances energy efficiency without affecting functional correctness.

- **Low-Power Lightweight AES with Resource Sharing**

  Resource-sharing AES architecture was introduced in "[13]" to improve area efficiency in FPGA-based lightweight cryptographic implementations. In this approach, arithmetic components such as XOR units, Galois Field multipliers, and shift logic are reused across multiple AES transformations instead of being instantiated separately for each stage. For example, the XOR networks used in MixColumns can be shared with those required for key expansion or AddRoundKey operations under FSM-based scheduling control.

  By time-multiplexing arithmetic resources, the architecture significantly reduces LUT and flip-flop utilization, minimizes routing congestion, and lowers overall silicon footprint. Since fewer parallel hardware blocks are active simultaneously, dynamic power consumption is also reduced. This technique is particularly beneficial for IOT FPGA platforms where hardware resources are limited and encryption workloads are intermittent. However, careful control logic design is required to ensure correct sequencing of shared operations without introducing functional conflicts or excessive latency.

- **Dynamic Voltage and Frequency Scaling (DVFS)**

  Dynamic Voltage and Frequency Scaling (DVFS) was explored in "[14]" as an adaptive power management technique for FPGA-based AES hardware targeting IOT applications. In this approach, the operating frequency and in some platforms, the supply voltage of the encryption core is adjusted based on workload requirements. During low data-rate or idle conditions, the AES module operates at a reduced clock frequency to decrease switching activity and dynamic power consumption. When higher performance is required, the system scales back to nominal frequency to maintain throughput.

  DVFS helps improve overall energy efficiency and extend battery life in IOT devices by aligning performance with real-time demand. However, full voltage scaling capability may not be available on all FPGA platforms, especially low-cost devices. In many cases, frequency scaling combined with clock management units (e.g., PLL/DLL reconfiguration) is used instead. Careful timing verification is required to ensure reliable operation across multiple frequency domains.

- **Masking-Based Side-Channel Protection**

  Masking techniques for lightweight AES were discussed in "[15]" to protect FPGA-based cryptographic implementations against side-channel attacks, particularly power analysis and differential power analysis (DPA). In masking-based approaches, random values (masks) are combined with sensitive intermediate variables such as S-box outputs and key-dependent computations. These masks decorrelate the power consumption profile from the actual secret key values, making it significantly harder for an attacker to extract key information through statistical analysis.

  In FPGA implementations, masking is typically applied at the byte or bit level and may involve Boolean masking or arithmetic masking schemes. Although masking improves resistance against first-order side-channel attacks, it introduces additional XOR operations, random number generators, and synchronization logic. Consequently, hardware area and power consumption increase slightly compared to unprotected lightweight AES cores. However, for IOT systems deployed in untrusted or physically accessible environments, masking-based protection provides a critical enhancement in security robustness.

- **Threshold Implementation for Security**

  Threshold implementation method was presented in "[16]" to enhance resistance against side-channel attacks in FPGA-based AES designs. In this approach, sensitive intermediate computations particularly the nonlinear S-box operation are divided into multiple statistically independent shares. Each share is processed separately in such a way that no single share reveals useful information about the secret key. By ensuring correctness, non-completeness, and uniformity properties, threshold implementation prevents first-order leakage and significantly strengthens resistance against differential power analysis (DPA) attacks.

  In FPGA implementations, threshold AES requires duplication or triplication of certain logic blocks and careful synchronization of shared computations. Although this increases LUT utilization and slightly raises power consumption compared to unprotected lightweight designs, it provides strong security guarantees for IOT devices operating in physically exposed or hostile environments. Therefore, threshold implementation offers a robust trade-off between area overhead and enhanced side-channel protection.

- **Sparse Logic-Based AES Optimization**

  Sparse logic optimization was discussed in "[17]" to reduce redundant switching activity in FPGA-based AES datapaths, thereby lowering dynamic power consumption. In this technique, the encryption architecture is modified to detect inactive or zero-valued input patterns and selectively bypass unnecessary computations. For example, if certain state bytes remain unchanged or contain zero values, corresponding XOR or shift operations can be conditionally disabled. This minimizes unnecessary toggling of logic elements and reduces overall switching capacitance.

  In IOT applications where sensor data often exhibits repetitive or sparse patterns, sparse-aware AES architectures can significantly reduce energy consumption without modifying the cryptographic functionality. However, additional control logic is required to monitor data activity and safely bypass operations, which introduces slight area overhead. Proper verification is essential to ensure that bypass mechanisms do not compromise encryption correctness or security properties.

- **Lightweight Modified S-Box Polynomial Approach**

Modified S-box polynomial architecture was introduced in "[18]" to enhance non-linearity while reducing computational complexity in FPGA-based lightweight AES implementations. In this approach, alternative irreducible polynomials are selected for finite field representation in GF(2^8), enabling a more hardware-friendly decomposition of the multiplicative inverse operation. By carefully choosing polynomial bases and transformation mappings, the S-box structure can be simplified, leading to reduced gate count and improved timing characteristics.

This technique allows generation of compact substitution logic with optimized hardware mapping, minimizing LUT utilization and reducing routing overhead on FPGA platforms. Additionally, alternative polynomial representations may improve resistance to certain algebraic attacks while maintaining acceptable diffusion properties. However, any modification to the S-box structure requires thorough cryptographic validation to ensure that security strength remains comparable to standard AES.

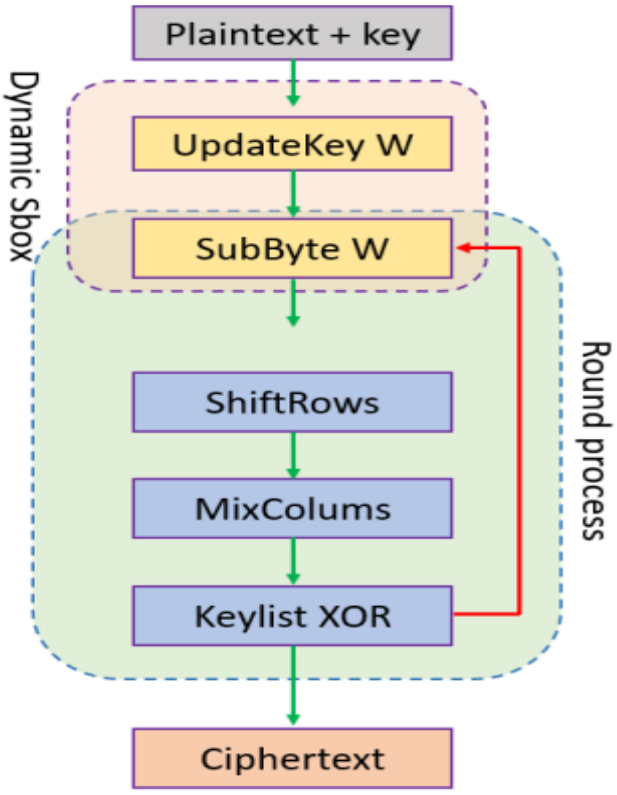

Figure 3.4: Process of the Proposed dynamic s-box

- **FPGA-Based Lightweight AES Accelerator for Edge Devices**

Lightweight AES accelerator design was discussed in "[19]" with emphasis on hardware-aware mapping techniques for embedded and edge FPGA platforms. In this approach, architectural optimizations such as selective loop unrolling, optimized dataflow scheduling, and register reuse are applied to improve the performance-to-area ratio. Instead of fully unrolling all AES rounds, critical operations are partially unrolled to increase throughput while maintaining controlled resource usage. Dataflow optimization ensures efficient movement of intermediate states between SubBytes, ShiftRows, MixColumns, and AddRoundKey stages, minimizing routing delay and memory access overhead.

Such accelerator designs are particularly suitable for edge devices that require secure communication with moderate throughput but limited FPGA resources. By balancing parallelism and hardware reuse, the architecture achieves improved encryption speed without significant increases in LUTs, flip-flops, or power consumption. However, careful floorplanning and timing optimization are necessary to maintain stable high-frequency operation.

- **Reconfigurable Multi-Mode AES Architecture**

  Reconfigurable AES architecture was presented in "[20]" to enable adaptive security levels in FPGA-based IoT systems. In this approach, the AES core is designed to operate in multiple configurable modes where parameters such as the number of rounds, key size, or operational frequency can be dynamically adjusted based on application requirements. For example, low-security or low-power scenarios may operate with fewer rounds to conserve energy, while high-security modes activate the full 10 rounds of AES-128.

  This dynamic configurability is typically achieved using FSM-controlled round counters and modular datapath design, allowing the encryption core to scale its computational workload in real time. Such flexibility helps balance security strength, power consumption, and performance depending on the operational context of the IoT node. However, reconfigurable designs require careful validation to ensure that security is not compromised when operating in reduced-round modes and that mode switching does not introduce vulnerabilities.

CONCLUSION

This survey reviewed recent developments in FPGA implementation of modified lightweight 128-bit AES architectures for IOT applications. Conventional AES-128, although highly secure, poses challenges for resource-constrained IOT devices due to its area and power requirements. The studies from 2018–2025 show that architectural optimizations such as reduced rounds, composite-field S-box design, optimized MixColumns, iterative processing, and hardware-aware power reduction techniques significantly improve efficiency on FPGA platforms.

Lightweight modifications, when carefully designed, maintain acceptable security properties while reducing hardware utilization and energy consumption. Therefore, FPGA-based modified AES architectures provide a practical and scalable solution for secure communication in modern IOT systems.

REFERENCES

1. K. Shahbazi et al., "High-Performance FPGA Implementation of AES," IET Computers & Digital Techniques, 2020.
2. S. S. Priya et al., "Compact Composite Field AES S-Box Architecture," 2018.
3. H. Kim et al., "Efficient Logic-Based AES S-Box for FPGA," Microprocessors and Microsystems, 2021.
4. S. Dhanda et al., "Area-Efficient AES MixColumns Optimization," Electronics, 2023.
5. Y. Wang et al., "Iterative AES Architecture for Low-Power Systems," IEEE Access, 2021.
6. T. Good and M. Benaissa, "Pipelined AES on FPGA," Integration Journal, 2018.
7. R. Patel et al., "Hybrid AES FPGA Architecture," 2022.

8. M. Ali et al., “On-the-Fly Key Expansion for Lightweight AES,” 2021.
9. J. Lee et al., “Bit-Serial AES for Ultra-Low Area FPGA,” 2019.
10. P. Kumar et al., “Byte-Serial AES Core for IOT,” 2020.
11. J. Singh et al., “Clock Gating in FPGA Cryptographic Designs,” 2019.
12. L. Chen et al., “Operand Isolation for Low-Power AES,” 2022.
13. R. Gupta et al., “Resource Sharing Techniques in Lightweight AES,” 2023.
14. A. Rahman et al., “DVFS-Based Energy Efficient AES,” 2020.
15. S. Mangard et al., “Masking Techniques for Secure AES,” IEEE Security & Privacy, 2019.
16. F. Regazzoni et al., “Threshold Implementations of AES,” 2018.
17. J. Park and C. Kim, “Sparse Logic Optimization in Cryptographic Hardware,” 2024.
18. H. Zhou et al., “Lightweight Modified AES for IOT,” IEEE IOT Journal, 2024.
19. P. Sze et al., “Hardware-Aware Cryptographic Accelerators,” IEEE VLSI Systems, 2024.
20. C. Zhang et al., “Reconfigurable AES Architectures for Edge Devices,” IEEE TCAS-I, 2025.