Journal of Systems Architecture 181 (2026) 104028

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

AES hardware architectures and non-standard Rijndael extensions up to 1024-bit keys: An open-source FPGA and 45-nm ASIC evaluation☆

Dinis Santos [a,c,*], Joao Cabacinho [a,c], Joao Casaleiro [b,c], Valderi R.Q. Leithardt [d], Luis Bica Oliveira [a,c]

[a] Department of Electrical and Computer Engineering, NOVA School of Science and Technology, Caparica, 2829-516, Portugal
[b] Department of Electronic Engineering and Telecommunications, Instituto Superior de Engenharia de Lisboa, Lisbon, 1959-007, Portugal
[c] CTS-UNINOVA & LASI, Portugal
[d] Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR, Lisboa, Portugal

ARTICLE INFO

Dataset link: https://github.com/Denishes/crypto-hardware-benchmark

Keywords:
AES
ASIC
FPGA
Hardware cryptography
Open-source EDA

ABSTRACT

Recent advances in embedded and IoT systems demand scalable and efficient cryptographic hardware capable of balancing performance, area, and energy constraints.

This paper presents a unified hardware evaluation of the standardized AES-128, AES-192, and AES-256 configurations together with two non-standard, author-defined extended Rijndael configurations using 512 and 1024-bit keys, targeting both application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs). Six micro-architectures are considered: fully iterative, two round partially unrolled, fully pipelined and their composite-field S-box variants. All designs share a 128-bit datapath and differ only in round reuse, enabling a controlled study of how the number of rounds and the key schedule depth affect area, maximum clock frequency, throughput, latency and area efficiency (Gbps/mm^2). The ASIC flow uses an open-source RTL to gates methodology in a 45 nm standard-cell library. The same RTL is mapped to a ECP5 FPGA to compare LUT usage and achievable frequency under realistic routing constraints. All RTL sources, synthesis scripts and configuration files used in this work are compatible with publicly available open-source EDA tools, enabling full reproducibility of the presented results.

1. Introduction

The Advanced Encryption Standard (AES) is the one of the most widely deployed symmetric encryption algorithm for secure communication in modern digital systems. Since its standardization by NIST in 2001 [1] and its formal specification in the Rijndael design framework [2], has become the dominant cryptographic primitive in applications ranging from secure storage and wireless communication to embedded and cloud computing platforms [1,3]. Due to the computational intensity of its substitution and permutation operations, hardware implementations of AES are often preferred when high throughput, low latency, or strict energy constraints are required.

In recent years, the rapid expansion of Internet-of-Things (IoT) devices has further increased the demand for efficient hardware cryptographic accelerators [4]. In these systems, encryption operations frequently occur during short active intervals between low-power sleep states. During wake-up events, the device typically performs several background tasks such as establishing wireless connectivity, authenticating with a network and transmitting encrypted data, reducing the execution time of cryptographic operations will allow the system to return to deep-sleep states faster. These events can occur thousands of times per year, making small reductions in processing latency significantly extend the battery lifetime of the device.

Hardware implementations of AES offer a large architectural design space, where performance and resource utilization are strongly influenced by the selected design methodology. Consequently, several architectural approaches have been proposed in the literature [5], ranging from iterative designs that reuse a single round unit to partially unrolled and fully pipelined architectures capable of producing one encrypted block per clock cycle. Each approach offers different trade-offs between area, throughput, latency and energy efficiency depending on the target application and hardware platform.

☆ This work was supported by the national funds through the Fundação para a Ciência e a Tecnologia (FCT) under the CTS multiannual funding program UID/00066/2025, the Lisbon 2030 Program under the code LISBOA2030-FEDER-00816400 and by the European Union through the European Regional Development Fund (ERDF), with DOI:10.54499/2023.16583.ICDT.

* Corresponding author.
E-mail address: dj.santos@uninova.pt (D. Santos).

https://doi.org/10.1016/j.sysarc.2026.104028
Received 4 June 2026; Received in revised form 25 September 2026; Accepted 25 September 2026
Available online 30 September 2026
1383-7621/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Several ASIC implementations of AES have been reported using advanced technology nodes. Sayilar and Chiou [6] focused on maximizing throughput for secure networking systems through extensive parallelism and architectural optimization, achieving significantly higher data rates at the expense of increased hardware resources. In contrast, Dong et al. [7] targeted throughput-per-area efficiency, employing architectural optimizations to reduce silicon cost while maintaining competitive performance. These studies collectively demonstrate the wide range of design trade-offs available in AES hardware implementations, although most focus on specific key sizes or architectural configurations rather than providing a unified solution supporting multiple standard AES and extended Rijndael configurations.

In this work, a unified evaluation of hardware architectures covering the standard AES-128, AES-192, and AES-256 configurations and two author-defined non-standard extended Rijndael configurations with 512 and 1024-bit keys will be presented. Multiple architectural approaches are implemented using a common RTL framework and evaluated under identical synthesis conditions. The designs are synthesized using an open-source RTL to gates flow based on Yosys [8] and OpenSTA, targeting the Nangate 45 nm Open Cell Library [9]. Physical implementation is performed using the OpenROAD toolchain [10,11] and are additionally mapped to a Lattice ECP5 FPGA [12]. This methodology enables a consistent comparison of area, clock frequency, throughput, latency and area efficiency across multiple architectures and key sizes while demonstrating the viability of open-source electronic design automation tools for cryptographic hardware research.

Unlike prior studies that typically focus on a single architecture, a limited key configuration or a single implementation platform, this work integrates multi-architecture exploration, extended key-size scaling, cross-platform ASIC and FPGA evaluation and fully open-source RTL to layout reproducibility within a unified framework, enabling transparent and physically grounded design-space exploration while providing a reusable foundation for future research in scalable cryptographic hardware.

A preliminary version of this research explored iterative and unrolled AES-128 architectures synthesized using an open-source 45 nm ASIC flow [13]. The present work substantially extends this previous study through the inclusion of non-standard extended Rijndael configurations with 512 and 1024-bit keys, physical layout generation and post-layout evaluation using the OpenROAD toolchain, expanded cross-platform FPGA and ASIC analysis and a broader exploration of scalable AES architectural design trade-offs.

The iterative, partially unrolled and fully pipelined organizations evaluated in this work, as well as the LUT-based and composite-field S-box implementations, are established AES hardware techniques. Accordingly, the contribution of this work is methodological and empirical rather than the proposal of a new AES microarchitecture. The study provides a controlled and reproducible benchmarking framework in which these established techniques are implemented using a common RTL structure and evaluated under consistent ASIC and FPGA flows.

The main contributions of this work are as follows:

- a common RTL benchmarking framework covering six established architectural and S-box implementation combinations.
- a controlled scalability evaluation across AES-128, AES-192, AES-256 and the non-standard ER-512 and ER-1024 hardware test cases.
- post-synthesis and post-route ASIC characterization under a common open-source 45 nm flow, including area, timing, throughput, power, utilization and critical-path analysis.
- platform-specific FPGA evaluation of the same RTL designs on the ECP5 architecture using an open-source toolchain.
- a reproducible verification methodology based on official AES known-answer vectors and independent Python reference models for the non-standard extended configurations.

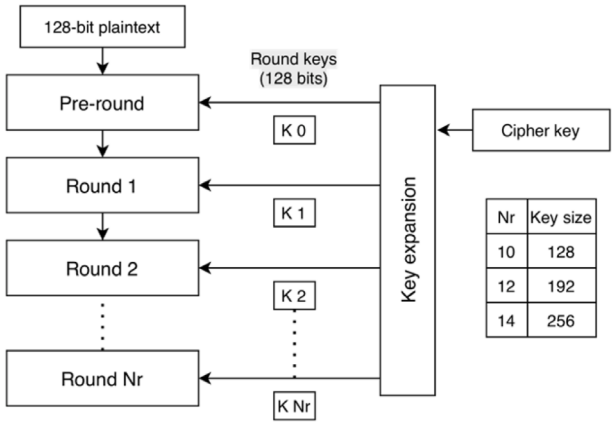


Fig. 1. AES encryption process.

Table 1
Standard AES configurations and corresponding round counts.

AES version	Number of rounds	Key size (bits)	Number of keys
AES-128	10	128	11
AES-192	12	192	13
AES-256	14	256	15

The data that support the findings of this study are available from the corresponding author upon reasonable request. The complete open-source RTL implementation flow, verification files, synthesis and physical-design experimental results, are publicly available at https://github.com/Denishes/crypto-hardware-benchmark [14].

2. Theoretical background and related work

2.1. AES algorithm overview

The Advanced Encryption Standard (AES) is a symmetric block cipher standardized by the National Institute of Standards and Technology (NIST) in [1]. AES operates on fixed-size 128-bit data blocks and supports key sizes of 128, 192 and 256 bits (see Fig. 1). Its cipher is based on a substitution permutation network composed of multiple transformation rounds that iteratively update a 4 × 4-byte state matrix [2]. Each encryption round consists of four main transformations applied to the internal state. The SubBytes operation performs a nonlinear byte substitution using a fixed substitution box (S-box). The ShiftRows transformation cyclically permutes the rows of the state matrix to provide inter-byte diffusion. The MixColumns operation applies a linear transformation over the finite field GF(2^8) (Galois Field) to each column of the state, increasing diffusion across the data block. Finally, AddRoundKey performs a bitwise exclusive OR between the current state and the round key derived from the key schedule, the number of encryption rounds depends on the cipher key size and Table 1 summarizes the standard AES configurations defined in [1].

The AES key expansion algorithm generates $N_r + 1$ round keys, where the first key is applied during the initial AddRoundKey step and the remaining keys are used in the subsequent encryption rounds.

On the other hand, the nonlinear SubBytes transformation is typically the most hardware intensive component of AES. It can be implemented using either lookup tables or composite-field arithmetic over GF($(2^4)^2$) that significantly reduce logic complexity by decomposing the inversion operation into smaller field operations, making them attractive for area-efficient hardware implementations.

2.2. Hardware architecture styles

AES hardware accelerators can be implemented using different architectural strategies depending on the desired trade-off between silicon area, throughput and latency, such architectural classifications are widely adopted in AES VLSI implementations [3,7]. These are some examples:

2.2.1. Iterative architectures

Iterative architectures reuse a single round implementation across multiple clock cycles. In this approach, one AES round is computed per cycle and the internal state is stored in registers between iterations, this minimizes hardware resources since only a single round datapath is required.

However, this type of implementation suffers from relatively high latency because all rounds must be executed sequentially, since the total number of cycles required for encryption equals the number of rounds plus additional cycles for key expansion or initialization.

2.2.2. Fully pipelined architectures

Fully pipelined implementations instantiate a dedicated hardware stage for every AES round. Once the pipeline is filled, one encrypted block can be produced every clock cycle. This approach achieves the highest possible throughput but requires significantly larger silicon area due to the replication of round logic.

2.2.3. Partially unrolled architectures

Partially unrolled designs represent a compromise between iterative and fully pipelined architectures. In this approach, multiple AES rounds are implemented within a single pipeline stage, reducing the number of cycles required to complete encryption while avoiding the full area cost of a completely unrolled pipeline.

For example, a two-round unrolled architecture processes two AES rounds per clock cycle, reducing the number of cycles required for encryption by approximately half, this allows for a favorable trade-off between area efficiency and throughput.

2.3. S-Box implementation techniques

The AES S-box can be implemented using several hardware techniques. The most straightforward and fastest approach is a lookup-table (LUT) implementation, where the nonlinear substitution values are stored in memory and accessed directly during the SubBytes operation, this method provides very low latency and simple control logic, it requires significant storage resources when multiple S-box instances are replicated, its simplicity and speed make it common in high-throughput or fully unrolled architectures.

An alternative approach uses composite-field arithmetic, where the multiplicative inversion in $GF(2^8)$ is decomposed into operations over smaller fields such as $GF((2^4)^2)$ [5,15]. By exploiting algebraic transformations and isomorphic mappings between fields, the inversion can be implemented using a network of smaller logic operations rather than a full lookup table, helping reduce logic complexity and silicon area, making it attractive for resource-constrained hardware implementations.

Composite-field S-boxes are particularly advantageous in architectures where the S-box must be replicated multiple times, such as deeply pipelined AES implementations, where minimizing the area of each substitution unit becomes critical for achieving efficient hardware utilization. Consequently, many area-optimized AES implementations adopt composite-field S-boxes, while high-performance designs often rely on LUT-based implementations to minimize critical path delay.

2.4. Related work

Numerous hardware implementations of the Advanced Encryption Standard (AES) have targeted a wide range of performance, area and power requirements, with architectural explorations focusing on either throughput-optimized spatial designs or compact iterative implementations [3,7,16]. These implementations explore architectural trade-offs between iterative, partially unrolled and fully pipelined datapaths, each offering different balances between silicon cost and achievable throughput, thereby establishing the foundation for subsequent research on AES architectural optimization.

Early high-performance AES implementations primarily focused on maximizing throughput through spatially expanded datapaths and aggressive pipelining. Mathew et al. [3] demonstrated that deeply pipelined AES and extended Rijndael architectures implemented in 45 nm CMOS can achieve multi-gigabit throughput while maintaining competitive area efficiency. Following a similar design philosophy, Sayilar and Chiou [6] targeted networking and data-center applications by employing extensive spatial parallelism to further increase throughput, accepting a larger silicon footprint in exchange for higher performance.

On the complete opposite, compact implementations have also been explored to minimize hardware cost, providing valuable insight into the performance and area trade-offs associated with different architectural approaches, where Dong et al. [7] proposed a compact AES architecture that improves throughput-per-area efficiency through a careful balance between datapath replication and logic reuse, illustrating the impact of parallelism and hardware resource utilization.

Another line of research has investigated non-standard hardware configurations beyond the AES key-size range. Cho et al. [17] evaluated author-defined configurations using 512- and 1024-bit cipher keys in a 45 nm CMOS technology. Their work extrapolated the Rijndael round-count relationship, resulting in 22 rounds for the 512-bit configuration and 38 rounds for the 1024-bit configuration. The experimental results showed that power consumption increases approximately linearly with key size, while silicon area grows more rapidly because of the additional key-expansion logic and encryption rounds. However, the study considered only a single iterative architecture and did not explore alternative hardware organizations.

Beyond conventional cryptographic accelerators, recent hardware-security-oriented studies have explored entropy generation and device-specific key derivation mechanisms based on ring oscillators, TRNGs and PUFs. A self-biased ring oscillator architecture implemented in 28 nm CMOS was proposed to improve robustness against process, voltage and temperature (PVT) variations while maintaining low area and power overheads [18]. Related work also demonstrated an FPGA-based true random number generator (TRNG) using ring oscillator jitter as an entropy source, achieving lightweight integration while successfully passing NIST statistical validation tests [19]. More recently, a systematic methodology for evaluating Physically Unclonable Functions (PUFs) was introduced, focusing on metrics such as uniqueness, reliability and randomness for hardware-based key generation [20]. These approaches are particularly relevant for future secure cryptographic systems, where ring oscillator structures may provide entropy sources, TRNGs may generate nonces and initialization vectors, and PUF-based architectures may derive device-unique cryptographic keys without permanent key storage.

Kang et al. [21] provided a comprehensive review of FPGA and ASIC-based accelerators for transformer inference, motivated by the growing challenge of deploying increasingly complex transformer models on resource-constrained hardware platforms. The survey examines how large computational demands of operations such as multi-head self-attention, together with substantial memory requirements and expensive non-linear functions, create significant bottlenecks for efficient hardware implementation. To address these challenges, the authors

analyze techniques such as quantization, pruning, sparsity-aware computation, approximation methods, pipelining, memory-efficient architectures, and hardware-software co-design approaches. For this purpose, the authors compare recent accelerator implementations in terms of throughput, power consumption, area, and resource utilization. Although focused on transformer acceleration, the survey is highly relevant to this work as it demonstrates how different hardware architectures can be systematically compared using metrics such as throughput, area, power consumption, and resource utilization.

Recent studies have highlighted the importance of light weight and computationally efficient cryptographic mechanisms for resource-constrained IoT environments, where security requirements must be balanced against strict hardware limitations [22]. Beyond data encryption, recent research has also explored secure boot and firmware authentication mechanisms based on post-quantum cryptography, Zhou et al. [23] proposed a layer-specific SPHINCS+ scheme that provides quantum-resistant firmware authentication while ensuring device integrity from power-up. Although such approaches significantly strengthen system security, their substantial computational and memory overheads reinforce the need for highly efficient symmetric cryptographic accelerators to protect data during normal system operation.

Recent research has highlighted that secure IoT systems depend not only on cryptographic strength but also on reliable detection and entropy-generation mechanisms. With this is mind, Pasikhani et al. [24] proposed an intrusion detection framework for 6LoWPAN networks capable of mitigating multiple routing attacks, while Shepherd and Hurley [25] demonstrated that mobile sensor data can provide substantially less entropy than commonly assumed. These findings reinforce the importance of jointly designing cryptographic accelerators, entropy sources and system-level security mechanisms for constrained IoT platforms, where efficient hardware implementations of cryptographic primitives such as AES play a central role in achieving both security and energy efficiency.

The PRISEC II and PRISEC III frameworks [26–28] propose adaptive cryptographic architectures that dynamically select algorithms such as AES, ChaCha20, Blowfish and RSA according to application requirements, device constraints and data sensitivity. Although these frameworks operate primarily at the system level, the computational overhead introduced by their layered security mechanisms highlights the value of hardware acceleration for reducing latency and energy consumption in constrained IoT devices while supporting more demanding security configurations in high-throughput applications.

Although these works provide valuable insights into AES hardware design, most prior studies focus either on architectural optimizations for standard AES configurations or on non-standard key-size scaling using a single hardware structure. A systematic evaluation of multiple hardware microarchitectures across standard AES and non-standard extended configurations under identical synthesis conditions remains largely unexplored.

In this work, a unified evaluation of scalable hardware architectures is presented, covering AES-128, AES-192, AES-256 and the non-standard ER-512 and ER-1024 hardware-scaling cases. Multiple architectural styles, including iterative, partially unrolled and fully pipelined implementations, are implemented using a common RTL framework and evaluated using an open-source ASIC flow targeting a 45 nm standard-cell library. The same RTL implementations are also mapped separately to an FPGA platform. This methodology enables a reproducible exploration of architectural scalability across the standardized AES and non-standard ER configurations while providing consistent comparisons among implementation styles within each target flow.

Prior work has already established the principal architectural trade-offs between iterative, unrolled and pipelined AES implementations, as well as the area benefits and timing characteristics of composite-field S-boxes [3,5,15]. Moreover, Cho et al. [17] previously investigated hardware scaling using 512- and 1024-bit key configurations. Accordingly, this work does not claim conceptual novelty in these individual architectures, S-box techniques or extended-key scaling rules. Its contribution is instead a controlled and reproducible benchmark that integrates a broader set of architectural configurations within a common RTL framework and evaluates them using consistent verification, post-synthesis, post-route ASIC and platform-specific FPGA flows [29].

Table 2
Non-standard extended Rijndael (ER) configurations used as hardware scalability test cases.

Configuration	Number of rounds	Key size (bits)	Round keys
ER-512	22	512	23
ER-1024	38	1024	39

2.5. Extended Rijndael configurations

FIPS 197 defines AES with a fixed 128-bit block and key sizes of 128, 192, and 256 bits, corresponding to 10, 12, and 14 encryption rounds, respectively [1]. Consequently, the 512- and 1024-bit key configurations considered in this work do not constitute AES variants. These key lengths also lie outside the parameter range specified for Rijndael [2].

Within the specified Rijndael parameter range, the number of encryption rounds is determined by

$$N_r = \max(N_b, N_k) + 6, \tag{1}$$

where N_b represents the block size in 32-bit words and N_k represents the key size in 32-bit words. For the standardized AES configurations, $N_b = 4$ and $N_k \in \{4, 6, 8\}$. For example, AES-256 uses $N_k = 8$ and therefore requires $N_r = 14$ rounds.

For the experimental configurations evaluated in this work, the block size remains fixed at 128 bits ($N_b = 4$), while Eq. (1) is extrapolated to $N_k = 16$ and $N_k = 32$. This produces configurations with 22 and 38 rounds, respectively. The Rijndael round transformations are retained, and the key-expansion structure is extended to support the larger keys.

These configurations are denoted through this work as ER-512 and ER-1024. They are non-standard extended Rijndael configurations defined exclusively as hardware-scalability test cases, they are not standardized AES variants or parameter sets specified by Rijndael. Cryptanalytic security evaluation is outside the scope of this work and was not performed for these configurations. Consequently, no claim of standard compliance, enhanced security, or suitability for practical deployment is made. Previous work by Cho et al. [17] investigated comparable 512- and 1024-bit configurations in hardware. The parameters adopted in this work are summarized in Table 2.

For ER-512 and ER-1024, the master key is divided into $N_k = 16$ and $N_k = 32$ consecutive 32-bit words, respectively, using the byte ordering defined in FIPS 197. The standard AES word-oriented key-expansion recurrence is then extrapolated to these larger values of N_k. Let

$$G(w, j) = \mathrm{SubWord}(\mathrm{RotWord}(w)) \oplus \mathrm{Rcon}_j\,. \tag{2}$$

Because both extended configurations satisfy $N_k > 6$, each subsequent word is calculated as

$$w_i = \begin{cases} w_{i-N_k} \oplus G(w_{i-1}, i/N_k), & i \bmod N_k = 0, \\ w_{i-N_k} \oplus \mathrm{SubWord}(w_{i-1}), & i \bmod N_k = 4, \\ w_{i-N_k} \oplus w_{i-1}, & \text{otherwise.} \end{cases} \tag{3}$$

The round constants are generated using the standard Rijndael recurrence, with $\mathrm{RC}_1 = 01$ and each subsequent value obtained by multiplication by 02 in GF(2^8), using the reduction polynomial 0x11B. The corresponding Rcon_j word contains RC_j as its most significant byte and zeros in the remaining three bytes.

The expansion continues until $4(N_r+1)$ words have been generated. Round key r is formed by concatenating words w_{4r}, w_{4r+1}, w_{4r+2} and

w_{4r+3}. Consequently, ER-512 generates 23 round keys for $N_r = 22$, whereas ER-1024 generates 39 round keys for $N_r = 38$. This extrapolation is part of the author-defined experimental configurations and is not specified by FIPS 197.

3. Hardware architecture

This section describes the hardware implementation of the standard AES configurations and the non-standard extended Rijndael scaling cases evaluated in this work. The designs were developed in Verilog HDL and implemented in three architectural configurations: iterative, partially unrolled and fully pipelined. All architectures share a common modular structure and differ primarily in the degree of round replication and register placement.

The objective of this design exploration is to evaluate the trade-offs between area, operating frequency and throughput across different architectural organizations in a consistent 128-bit datapath.

3.1. Architecture overview

Each encryption round consists of four transformations applied sequentially: SubBytes, which performs nonlinear byte substitution using the AES S-box; ShiftRows, which cyclically permutes the rows of the state matrix; MixColumns, which applies a linear transformation over GF(2^8); and AddRoundKey, which performs a bitwise XOR between the state and the corresponding round key. The final round omits the MixColumns transformation.

These round transformations and the fixed 128-bit state are shared by the standard AES configurations and the non-standard extended Rijndael configurations evaluated in this work. The configurations differ in their key sizes, key-expansion depth and number of encryption rounds.

Both encryption and decryption datapaths share the same hierarchical organization and key-expansion module, differing only in the order and type of round transformations. Encryption rounds use SubBytes, ShiftRows, MixColumns and AddRoundKey, while decryption rounds employ InvSubBytes, InvShiftRows, InvMixColumns and AddRoundKey [1,2].

To facilitate verification and synthesis, each transformation is implemented as an independent Verilog module. The datapath is therefore composed of reusable blocks corresponding to the Rijndael round transformations.

Full byte-level parallelism is achieved by instantiating sixteen S-box units per round, corresponding to the sixteen bytes of the fixed 128-bit state. Verilog's `generate` construct is used to replicate the substitution units within each round module.

Two S-box implementations are explored:

- lookup-table-based implementation, optimized for FPGA synthesis;
- composite-field implementation, optimized for ASIC area efficiency.

These alternatives enable a direct evaluation of the impact of S-box implementation style on hardware resource utilization and timing performance.

3.2. Register-transfer datapaths

All evaluated architectures use a 128-bit state datapath. Let P denote the input plaintext, K_r the 128-bit round key used in round r, and S_r the state after round r. The initial state is produced by the initial AddRoundKey operation:

$$S_0 = P \oplus K_0. \tag{4}$$

For rounds $1 \le r < N_r$, the round transformation is

$$S_r = \mathrm{MC}\big(\mathrm{SR}\big(\mathrm{SB}(S_{r-1})\big)\big) \oplus K_r, \tag{5}$$

where SB, SR and MC denote SubBytes, ShiftRows and MixColumns, respectively. The final round omits MixColumns:

$$S_{N_r} = \mathrm{SR}\Big(\mathrm{SB}(S_{N_r-1})\Big) \oplus K_{N_r}. \tag{6}$$

For compactness, let $\mathcal{R}_r(S, K_r)$ denote the transformation performed by round r, including AddRoundKey and the omission of MixColumns when $r = N_r$.

In the iterative architecture, a single 128-bit state register stores the intermediate state. During the first encryption cycle, an input multiplexer selects $P \oplus K_0$, the first-round transformation is evaluated combinationally and its result is captured by the state register. During cycles 2 through N_r, the multiplexer selects the registered feedback state, while the round counter selects the corresponding round key. The round result is written back into the same state register at every rising clock edge. After N_r register updates, the state-register output contains the ciphertext. Therefore, the initial AddRoundKey operation does not require a separate clock cycle.

In the two-round partially unrolled architecture, one state register is placed around a combinational datapath containing two consecutive round modules. During processing step j, the key-selection logic supplies K_{2j+1} and K_{2j+2}, and the state-register update is

$$S^{(j+1)} = \mathcal{R}_{2j+2}\left(\mathcal{R}_{2j+1}\left(S^{(j)}, K_{2j+1}\right), K_{2j+2}\right). \tag{7}$$

Here, $S^{(0)} = P \oplus K_0$, $0 \le j < N_r/2$, and $S^{(j)}$ denotes the state-register contents after processing step j. Thus, two rounds are completed between consecutive state-register updates. All evaluated round counts are even, resulting in $N_r/2$ register-transfer steps per block.

In the fully pipelined architecture, each encryption round has its own combinational round module followed by a dedicated 128-bit state register. The initial AddRoundKey operation is evaluated before the first round stage and does not introduce an additional registered pipeline stage. If $Q_r[c]$ denotes the contents of the register following round r during clock cycle c, the register transfer is

$$Q_r[c+1] = \mathcal{R}_r\left(Q_{r-1}[c], K_r\right), \qquad 1 \le r \le N_r, \tag{8}$$

with $Q_0[c] = P[c] \oplus K_0$. At every rising clock edge, each stage captures the round result produced by the preceding stage. Consequently, a block accepted during cycle c advances by one registered round stage per cycle and appears at Q_{N_r} after N_r cycles. A new plaintext block can be accepted during every cycle from the start of operation. After the initial N_r-cycle pipeline-fill period, one ciphertext block is produced during every clock cycle.

The combinational key-expansion chain makes all N_r+1 round keys available after settling. During fixed-key operation, each pipeline stage receives the stable round key corresponding to its round index. The reported steady-state throughput assumes continuous block processing after key initialization.

The decryption datapaths use the same register placement and transfer schedules, replacing the encryption transformations with their inverse counterparts and applying the corresponding round keys in reverse order.

3.3. Parallelization and throughput

Parallelism is exploited at two complementary levels: byte-level parallelism and round-level parallelism. At the byte level, each round module instantiates sixteen S-box units, enabling all sixteen bytes of the 128-bit state to be processed concurrently (see Fig. 2).

At the round level, the architectures differ in the degree of round replication and in their ability to overlap the processing of consecutive

Pipeline Timing Diagram-10 Parallel Stages						
Clock	Stage 1	Stage 2	Stage 3	...	Stage 10	Output
1	Block A (1)			...		
2	Block B (1)	Block A (2)		...		
3	Block C (1)	Block B (2)	Block A (3)	...		
...	...	...	...	...	...	...
10	Block J (1)	Block I (2)	Block H (3)	...	Block A (10)	
11	Block k (1)	Block J (2)	Block I (3)	...	Block B (10)	Output A

Fig. 2. Register-transfer timing of the fully pipelined architecture AES-128 with $r = 10$. Each block advances by one registered round stage per cycle, resulting in a latency of N_r cycles and an initiation interval of one cycle.

input blocks. To distinguish these timing characteristics, the latency L, pipeline fill time and initiation interval II are defined separately.

The latency L is the number of clock cycles between the acceptance of a plaintext block and the availability of its corresponding ciphertext. For a pipelined architecture, the pipeline fill time is the interval between the acceptance of the first plaintext block and the production of the first valid ciphertext. In the fully pipelined implementations evaluated here, the fill time is therefore equal to the latency. The initiation interval II is the minimum number of clock cycles between the acceptance of two consecutive independent plaintext blocks.

For a sequence of B blocks, the number of elapsed clock cycles between the acceptance of the first input and the production of the final ciphertext is

$$N_{\text{cycles}}(B) = L + (B-1)II. \tag{9}$$

The corresponding average throughput for this finite sequence is

$$T(B) = \frac{128Bf_{\text{clk}}}{L + (B-1)II}, \tag{10}$$

where f_{clk} is the operating clock frequency and 128 is the block size in bits. As the number of continuously processed blocks increases, the effect of the initial latency becomes negligible. The steady-state throughput is therefore

$$T_{\text{ss}} = \lim_{B\to\infty} T(B) = \frac{128f_{\text{clk}}}{II}. \tag{11}$$

When f_{clk} is expressed in MHz, division by 10^3 gives throughput in Gbit/s. For the post-synthesis ASIC results, $f_{\text{clk}} = f_{\text{pre}}$, where f_{pre} is the pre-layout timing estimate. For the physical-design results, $f_{\text{clk}} = f_{\text{target}}$, where f_{target} is the highest target frequency at which the corresponding implementation achieved timing closure and the frequency used for power estimation. The resulting post-layout value is therefore reported as the nominal steady-state throughput $T_{\text{nom,ss}}$.

The iterative architecture instantiates one round module and reuses it until all N_r rounds have been completed. Because the processing of consecutive blocks is not overlapped, both its latency and initiation interval are N_r cycles.

The two-round partially unrolled architecture instantiates two round modules and evaluates two encryption rounds during each clock cycle. Its latency and initiation interval are therefore both $\lceil N_r/2 \rceil$ cycles. As with the iterative architecture, there is no separate pipeline-fill phase because a new block is accepted only after the current block has completed.

The fully pipelined architecture instantiates a dedicated hardware stage for each encryption round and inserts registers between consecutive stages. In the implementation evaluated here, its latency and initial fill time are both N_r cycles. However, because the processing of different blocks is overlapped across the pipeline stages, its initiation interval is one cycle. After the initial fill period, the pipeline accepts one new plaintext block and produces one ciphertext block during every clock cycle [3,6].

The timing characteristics of the three architectural organizations are summarized in Table 3.

Table 3
Latency, fill time and initiation interval of the evaluated architectures.

Architecture	Latency L (cycles)	Fill time (cycles)	II (cycles)
Iterative	N_r	–	N_r
Partially unrolled	$\lceil N_r/2 \rceil$	–	$\lceil N_r/2 \rceil$
Pipeline	N_r	N_r	1

The same timing organization is applied to AES-128, AES-192, AES-256, ER-512 and ER-1024, which use 10, 12, 14, 22 and 38 rounds, respectively. Consequently, the iterative implementations have latency and initiation-interval values of 10, 12, 14, 22 and 38 cycles, while the partially unrolled implementations have corresponding values of 5, 6, 7, 11 and 19 cycles.

For example, the AES-128 fully pipelined architecture has a latency and initial fill time of ten cycles, but an initiation interval of one cycle. Its first ciphertext is produced after the ten-cycle fill period, after which one ciphertext is produced per clock cycle. Therefore, the value of ten reported in the latency column describes the time experienced by an individual block and is not used as the divisor when calculating steady-state throughput. In contrast, the AES-128 iterative and partially unrolled architectures have initiation intervals of ten and five cycles, respectively.

3.4. Key expansion module

For AES-128, AES-192 and AES-256, the key-expansion procedure follows FIPS 197 [1]. For example, AES-128 derives eleven 128-bit round keys from its 128-bit master key. The standardized key-expansion procedure includes the following operations:

- RotWord – cyclic rotation of a 32-bit word.
- SubWord – byte substitution using the AES S-Box.
- Rcon addition – XOR with the round constant.

For ER-512 and ER-1024, the RTL extends this word-oriented key-expansion structure to $N_k = 16$ and $N_k = 32$, respectively. The extended procedure generates $N_r + 1$ round keys of 128 bits, corresponding to 23 round keys for ER-512 and 39 round keys for ER-1024. This extension forms part of the experimental definition used in this work and is not specified by FIPS 197.

Each evaluated core includes the complete key-expansion circuitry required to derive all $N_r + 1$ round keys from the master key. The expansion stages form a cascaded combinational chain, and SubWord uses the same S-box implementation style as the encryption datapath. The reported area, FPGA resource usage and total-power estimates cover the complete implemented cores, including their key-expansion circuitry.

Post-layout timing, block latency and steady-state throughput are evaluated under fixed-key operation, with the master key preloaded and the derived round keys stable before block processing begins. The master key remains unchanged until all blocks using it have completed. Key loading and key-expansion settling are initialization overheads outside the reported steady-state performance metrics. The corresponding timing constraints are described in Section 4.5.

3.5. Functional verification

Functional verification was performed at the RTL level for all evaluated configurations and microarchitectures. The standardized AES-128, AES-192 and AES-256 implementations were verified using the official known-answer example vectors associated with FIPS 197 [1]. For each key size, the same plaintext and key were applied to the pipeline, iterative and partially unrolled architectures using both LUT-based and composite-field S-box implementations. The encryption outputs were

compared bit for bit with the corresponding NIST reference ciphertexts. The reference ciphertexts were subsequently applied to the decryption datapaths, which were required to recover the original plaintexts. All 18 standard AES implementations produced the expected encryption and decryption results.

Because ER-512 and ER-1024 are author-defined non-standard configurations, no standardized reference vectors are available for these cases. Therefore, separate bit-accurate encryption and decryption reference models were implemented in Python. The Python models use the same 128-bit state representation and Rijndael round transformations as the standard configurations, together with the extended key-expansion rules defined in Eq. (3). The Python models were developed independently of the Verilog implementations and served as the functional references for the extended configurations.

A fixed set of plaintext–key pairs was assembled for ER-512 and ER-1024. The keys were randomly generated, and the exact plaintexts, keys and expected ciphertexts were retained in the released test-vector files. Each pair was processed by the corresponding Python reference model to generate a ciphertext, which was compared bit for bit with the outputs of the Verilog microarchitectures. The ciphertexts were also passed to the decryption datapaths to confirm recovery of the original plaintexts. All comparisons and round-trip tests produced identical results. The Python reference model and test vectors are included in the accompanying repository [14].

3.6. Open-source implementation flow

All AES and non-standard extended Rijndael designs were implemented in Verilog HDL using a common open-source flow. For ASIC evaluation, Yosys [8] synthesized the designs using the Nangate 45 nm Open Cell Library [9]. OpenSTA provided post-synthesis area and timing estimates, while OpenROAD was used for placement, clock-tree synthesis, routing and post-route analysis [10].

Post-layout power was estimated with OpenSTA within OpenROAD using each design's final routed OpenDB database, SDC constraints and SPEF-extracted parasitics. The Nangate45 typical Liberty corner at 1.1 V was used. No application-specific input vectors or VCD/SAIF activity traces were applied. Instead, a common vectorless model assigned 0.1 transitions per clock cycle and a static probability of 0.5 to all non-clock pins. Clock activity was derived from the SDC. Power was evaluated at each design's reported target frequency. Only total power, comprising internal-cell, net-switching and leakage power, is reported and used to calculate throughput per watt. The vectorless power model is applied separately from the fixed-key timing constraints. The reported power and derived energy metrics use the stated activity assumptions; the energy associated with individual key-initialization events is not evaluated. These values are comparative estimates, not workload-specific or measured-silicon power.

The same RTL designs were also mapped to a Lattice ECP5 FPGA [12] to evaluate LUT usage and mapped clock frequency. ASIC and FPGA results characterize the designs on their respective platforms and are not used for a direct cross-technology performance ranking.

4. Results and discussion

This section presents the ASIC and FPGA implementation results for the six evaluated micro-architectures: iterative, two-round partially unrolled and fully pipelined, each using either LUT-based or composite-field S-boxes. Each micro-architecture is evaluated for the standard AES-128, AES-192 and AES-256 configurations and for the non-standard ER-512 and ER-1024 scaling cases. All configurations maintain a fixed 128-bit datapath.

Table 4 consolidates the post-synthesis ASIC metrics obtained with the Nangate 45 nm standard-cell library and the FPGA mapping results obtained for the ECP5 device. Throughput is derived from the maximum operating frequency and the initiation interval of each architecture, while ASIC area efficiency is reported in Gb/s/mm^2. For fully pipelined designs, steady-state throughput corresponds to one 128-bit block per clock cycle after pipeline fill. Iterative and partially unrolled designs require multiple cycles per block but use less replicated round hardware.

4.1. ASIC synthesis results

Table 4 reports the ASIC synthesis results targeting the Nangate 45 nm standard-cell library. The fully pipelined architectures achieve the highest steady-state throughput across all evaluated configuration. In AES-128, the LUT-based pipelined core reaches 113 Gb/s at 881 MHz with an area of 0.109 mm^2, while the composite-field pipelined design reduces silicon area to 0.052 mm^2 while sustaining 86 Gb/s at 671 MHz. Consequently, the composite-field pipelined core achieves the highest AES-128 area efficiency (1652 Gb/s/mm^2), demonstrating that arithmetic optimization of the S-box significantly improves throughput density in spatial architectures.

In contrast, the iterative LUT-based design occupies 0.033 mm^2 with 3.8 Gb/s throughput, while the composite-field variant reduces area further to 0.016 mm^2 with similar throughput. The 2-round partially unrolled designs occupy an intermediate position in the design space, reducing cycles per block relative to iterative implementations while avoiding full replication of all round stages.

Across the evaluated configurations, the fully pipelined architectures maintain approximately constant steady-state throughput because their initiation interval remains one clock cycle. However, their pipeline latency and silicon area increase with the number of encryption rounds because additional round stages are instantiated. In contrast, the iterative architectures reuse the same round datapath and therefore remain comparatively compact, but require more clock cycles per block as the round count increases.

4.2. FPGA mapping results

FPGA mapping results in Table 4 report ECP5 maximum frequency and total LUT usage for the same RTL architectures.

As expected, LUT usage increases with architectural replication. For AES-128, the fully pipelined LUT-based design uses 38,033 LUTs, while the composite-field pipelined variant reduces this to 24,426 LUTs. Iterative designs require substantially fewer resources due to round reuse, using 5925 LUTs (LUT-based) and 5306 LUTs (composite-field).

A second observable trend is that iterative and 2-round partially unrolled designs exhibit nearly constant LUT usage and mapped frequency across the evaluated configurations. This occurs because these architectures reuse a fixed 128-bit round datapath and scale to larger keys primarily by increasing the number of round iterations (i.e., cycles per block), rather than by instantiating additional round hardware. Consequently, the dominant LUT cost is determined by the round datapath (S-boxes, MixColumns and XOR networks), while key size-dependent control logic contributes only marginal overhead that is often absorbed by FPGA packing.

Since the critical path remains dominated by the same per-cycle round logic, the mapped f_{max} remains relatively stable (approximately 80 MHz for iterative designs and 50 MHz for partially unrolled designs).

In contrast, fully pipelined architectures replicate round hardware as the number of rounds increases, leading to a strong growth in LUT usage and routing congestion. The resulting increase in interconnect delay and fanout reduces achievable frequency relative to more compact architectures. The ECP5 logic fabric, organized into PFU blocks with efficient packing capabilities, partially mitigates this growth but does not eliminate the scaling limitation.

Notably, the fully pipelined ER-1024 LUT-based design could not be mapped onto the selected ECP5 device because its resource requirement of 133,619 LUTs exceeds the approximately 85k-LUT capacity of the device. This result illustrates the practical resource limitations of deeply pipelined extended-round architectures on a mid-range FPGA.

Table 4
ASIC (45 nm) and FPGA (ECP5) implementation results for AES and non-standard extended Rijndael (ER) configurations.

Config.	Design	Area (mm^2)	CLK (MHz)	T_{ss} (Gb/s)	Latency L (cycles)	Area Eff. (Gb/s/mm^2)	Max Freq. (MHz)	LUT usage (total)
AES-128	Pipeline LUT	0.109	881	113	10	1035	54.66	38 033
	Pipeline GF	0.052	671	86	10	1652	64.37	24 426
	Iterative LUT	0.033	293	3.8	10	114	79.43	5925
	Iterative GF	0.016	309	4.0	10	247	67.76	5306
	Partially unrolled LUT	0.040	284	7.3	5	182	50.16	11 469
	Partially unrolled GF	0.022	220	5.6	5	256	36.05	9872
AES-192	Pipeline LUT	0.130	886	113	12	872	57.72	45 247
	Pipeline GF	0.064	671	86	12	1342	62.80	30 302
	Iterative LUT	0.029	523	5.6	12	192	83.82	5788
	Iterative GF	0.017	512	5.5	12	321	67.91	5263
	Partially unrolled LUT	0.045	284	6.1	6	135	50.23	13 048
	Partially unrolled GF	0.025	221	4.7	6	189	35.67	9898
AES-256	Pipeline	0.150	883	113	14	753	50.76	51 040
	Pipeline GF	0.073	666	85	14	1168	60.37	30 505
	Iterative LUT	0.040	595	5.4	14	136	78.65	4892
	Iterative GF	0.024	591	5.4	14	225	68.64	4401
	Partially unrolled LUT	0.078	244	4.5	7	57	51.06	10 446
	Partially unrolled GF	0.040	214	3.9	7	198	36.37	8909
ER-512	Pipeline LUT	0.218	880	113	22	517	52.36	78 007
	Pipeline GF	0.104	662	85	22	815	63.90	46 438
	Iterative LUT	0.036	324	1.9	22	52	82.40	4630
	Iterative GF	0.022	322	1.9	22	85	67.55	3965
	Partially unrolled LUT	0.114	359	4.2	11	37	50.89	10 483
	Partially unrolled GF	0.059	195	1.3	11	38	36.70	8350
ER-1024	Pipeline LUT	0.359	879	113	38	313	–	133 619
	Pipeline GF	0.170	662	85	38	499	62.87	75 733
	Iterative LUT	0.038	427	1.4	38	38	81.83	4658
	Iterative GF	0.025	423	1.4	38	58	70.65	4053
	Partially unrolled LUT	0.185	271	1.8	19	10	50.45	11 386
	Partially unrolled GF	0.092	188	1.3	19	14	36.00	9482

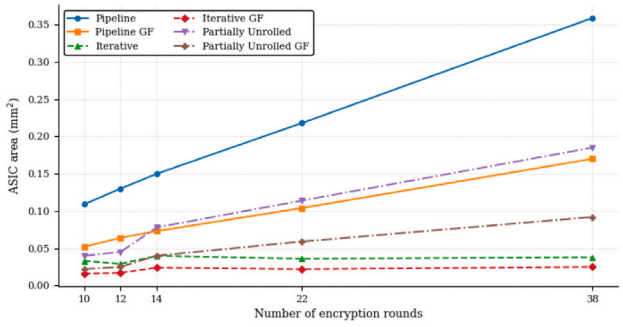


Fig. 3. ASIC area scaling as a function of the number of encryption rounds for the evaluated architectures synthesized using the Nangate 45 nm standard-cell library.

4.3. Area scaling analysis

To further investigate architectural scalability, Fig. 3 presents the synthesized silicon area as a function of the number of encryption rounds for the six evaluated micro-architectures. The results illustrate how hardware cost changes across the standard AES configurations and the non-standard ER-512 and ER-1024 scaling cases.

As expected, the fully pipelined architectures exhibit the highest area consumption due to the replication of round logic required to maximize throughput. The growth becomes more pronounced as the number of encryption rounds increases, reflecting the direct relationship between pipeline depth and hardware resources. In contrast, the iterative architectures maintain nearly constant area across all key sizes because the same round hardware is reused over multiple clock cycles.

A consistent reduction in silicon area can be observed when composite-field arithmetic is employed. For all evaluated key sizes, the GF-based implementations require significantly fewer resources than their conventional counterparts. This reduction is primarily attributed to the lower complexity of the composite-field S-box implementation, which replaces the traditional lookup-table-based realization with arithmetic operations over smaller Galois fields.

The partially unrolled architectures occupy an intermediate position between the iterative and fully pipelined approaches, providing a compromise between hardware cost and throughput. As the number of rounds increases, their area grows approximately linearly, but at a lower rate than that observed for the fully pipelined designs.

Overall, the results show that composite-field arithmetic reduces hardware area across the standard AES and non-standard extended Rijndael configurations, with this relative area reduction remaining evident as the number of encryption rounds increases. Furthermore, the observed scaling behavior illustrates the expected trade-off between hardware area and steady-state throughput: iterative architectures minimize area, whereas fully pipelined architectures maximize steady-state throughput.

4.4. Physical layout evaluation

To evaluate physical realizability, all thirty ASIC configurations were processed through the complete OpenROAD physical-design flow targeting the Nangate 45 nm standard-cell library at 1.1 V. The evaluated set comprises six micro-architectures for AES-128, AES-192, AES-256, ER-512 and ER-1024. The physical-design flow included floorplanning, standard-cell placement, clock-tree synthesis, global routing, detailed routing and post-route analysis.

Fig. 4 shows the AES-128 iterative GF and Pipeline GF layouts as representative examples of the round-reuse and round-replication extremes. The layouts are presented at the same scale to illustrate their relative core dimensions and routing densities. Although only these representative layouts are shown, all evaluated ASIC configurations were processed through the same physical-design flow.

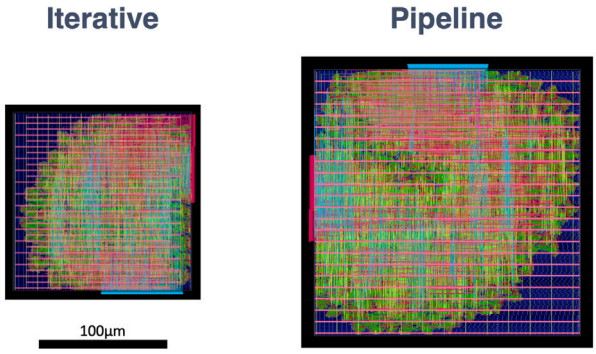


Fig. 4. Representative physical layouts of the AES-128 composite-field implementations: iterative and fully pipelined.

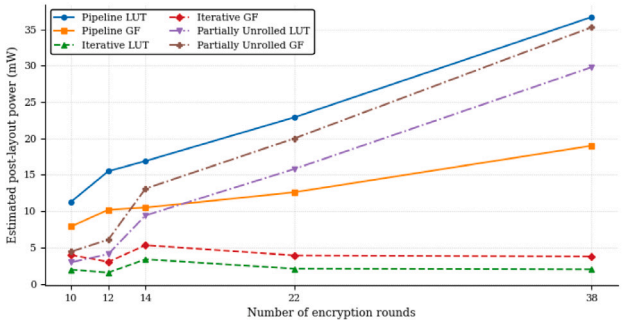


Fig. 5. Estimated post-layout total power versus number of rounds.

Table 5 summarizes the post-layout core area, target clock frequency, latency, initiation interval, target-based nominal throughput, estimated power and core utilization. The reported target frequency, f_{target}, is the clock constraint applied during physical implementation and should not be interpreted as the achieved post-route maximum frequency when timing closure is not obtained. The nominal throughput, T_{nom}, is calculated at f_{target} using

$$T_{\text{nom}} = \frac{128 f_{\text{target}}}{\text{II}}, \tag{12}$$

where II denotes the initiation interval. For fully pipelined architectures, the pipeline-fill time is equal to the reported latency and the initiation interval is one clock cycle. For iterative and two-round partially unrolled architectures, the initiation interval corresponds to the number of cycles required to complete one block.

The energy efficiency of each implementation is expressed as throughput per watt:

$$\eta_{\text{E}} = \frac{1000\, T_{\text{nom,ss}}}{P_{\text{total}}}, \tag{13}$$

where $T_{\text{nom,ss}}$ is expressed in Gbit/s, P_{total} is expressed in mW and η_{E} is obtained in Gbit/J. The corresponding energy required to encrypt one 128-bit block is

$$E_{\text{block}} = P_{\text{total}} \frac{II}{f_{\text{target}}}, \tag{14}$$

where II is the initiation interval. When P_{total} is expressed in mW and f_{target} in MHz, E_{block} is obtained directly in nJ per block. Equivalently, the energy per encrypted bit is

$$E_{\text{bit}}\,[\text{pJ/bit}] = \frac{P_{\text{total}}\,[\text{mW}]}{T_{\text{nom,ss}}\,[\text{Gbit/s}]}. \tag{15}$$

The reported power and energy values are post-layout estimates evaluated at f_{target}; they are not measured-silicon results.

The post-layout results show that core area increases with round replication and pipeline depth. The largest core is the ER-1024 fully pipelined LUT-based implementation, whereas the iterative implementations remain comparatively compact because they reuse the same round datapath. Composite-field S-boxes reduce post-layout core area across all evaluated configurations, with the most pronounced reductions occurring in the fully pipelined architectures where the S-box logic is replicated at every round stage.

Core utilization remains between 34% and 42% across the evaluated implementations. The reported post-layout core areas are larger than the corresponding pre-layout synthesis estimates because the two metrics represent different stages of implementation. Pre-layout area primarily represents the estimated area of the synthesized standard cells, whereas post-layout core area includes the floorplan whitespace reserved for placement and routing, together with clock-tree buffers, timing-repair cells, physical-only cells and routing-related spacing.

For example, the AES-128 Pipeline LUT area increases from 0.109 mm^2 before layout to 0.150 mm^2 after layout, corresponding to an increase of approximately 37.5%. Its Pipeline GF counterpart increases from 0.052 mm^2 to 0.079 mm^2, or approximately 52.2%. For ER-1024, the corresponding increases are approximately 26.9% for Pipeline LUT and 40.4% for Pipeline GF. These differences should therefore not be interpreted solely as growth in functional logic, but primarily as the additional physical footprint required to obtain a routable implementation. The lower utilization of some deeply pipelined configurations reflects the additional whitespace required for routing, buffering and clock-tree distribution as the number of replicated round stages increases.

As shown in Fig. 5, estimated power increases with the number of round stages in the fully pipelined architectures. This behavior results from the replication of round logic, pipeline registers and clock-distribution resources. Within this architectural class, the composite-field implementation consistently consumes less power than the LUT-based implementation. The estimated reduction increases from approximately 30% for AES-128 to approximately 48% for ER-1024, indicating that the power benefit of the composite-field S-box becomes more pronounced when the round logic is extensively replicated.

A different trend is observed for the iterative and partially unrolled architectures. Although their composite-field implementations occupy less core area, they exhibit higher estimated power than their LUT-based counterparts. Therefore, the area reduction obtained through composite-field arithmetic does not necessarily translate into lower power for architectures that reuse round hardware. This result highlights the importance of evaluating area and power independently rather than assuming that the smallest implementation is also the least power-consuming.

The iterative architectures exhibit the lowest absolute power because they contain less replicated round logic. However, their throughput is also substantially lower than that of the fully pipelined architectures. Consequently, absolute power alone is insufficient for comparing energy efficiency.

The throughput-per-watt results reported in Table 5 normalize power consumption by processing rate and show that the fully pipelined architectures, particularly the Pipeline GF implementations, provide the highest energy efficiency across the evaluated configurations. From AES-128 to ER-1024, post-layout iterative-core area increases from 0.047 to 0.053 mm^2 for the LUT design and from 0.031 to 0.040 mm^2 for the GF design, despite the round count increasing from 10 to 38. This comparatively modest total-area growth is consistent with round-datapath reuse. A separate key-schedule area breakdown was not performed, so its individual contribution cannot be quantified.

Because each implementation is evaluated at its respective target frequency, Fig. 5 should be interpreted together with the frequency and throughput results in Table 5. The figure characterizes power under the

Table 5
Post-layout ASIC results for AES and non-standard extended Rijndael (ER) configurations. Timing, block latency and throughput assume fixed-key operation after key initialization. Total power and throughput per watt are vectorless OpenSTA estimates from routed designs, not measurements from fabricated chips.

Config.	Design	Area (mm^2)	f_{pre} (MHz)	f_{target} (MHz)	Δf (%)	Latency L (cycles)	$T_{nom,ss}$ (Gbit/s)	P_{total} (mW)	Thr./W (Gbit/J)	Util. (%)
AES-128	Pipeline LUT	0.150	881	881	0.0	10	112.8	11.30	9979	42
	Pipeline GF	0.079	671	671	0.0	10	85.9	7.90	10 872	42
	Iterative LUT	0.047	293	293	0.0	10	3.8	1.97	1904	38
	Iterative GF	0.031	309	309	0.0	10	4.0	3.99	991	39
	Partially unrolled LUT	0.058	284	284	0.0	5	7.3	2.97	2448	38
	Partially unrolled GF	0.036	220	220	0.0	5	5.6	4.45	1266	40
AES-192	Pipeline LUT	0.179	886	886	0.0	12	113.4	15.50	7317	42
	Pipeline GF	0.098	671	671	0.0	12	85.9	10.20	8420	42
	Iterative LUT	0.041	523	523	0.0	12	5.6	1.56	3576	38
	Iterative GF	0.026	512	512	0.0	12	5.5	3.03	1802	42
	Partially unrolled LUT	0.065	284	284	0.0	6	6.1	4.11	1474	38
	Partially unrolled GF	0.042	221	221	0.0	6	4.7	6.08	773	39
AES-256	Pipeline LUT	0.205	883	880	−0.3	14	112.6	16.90	6665	42
	Pipeline GF	0.106	666	600	−9.9	14	76.8	10.50	7314	41
	Iterative LUT	0.059	595	595	0.0	14	5.4	3.39	1605	37
	Iterative GF	0.039	591	591	0.0	14	5.4	5.33	1014	39
	Partially unrolled LUT	0.114	244	244	0.0	7	4.5	9.42	474	37
	Partially unrolled GF	0.076	214	214	0.0	7	3.9	13.10	299	39
ER-512	Pipeline LUT	0.287	880	850	−3.4	22	108.8	22.90	4751	41
	Pipeline GF	0.149	662	590	−10.9	22	75.5	12.60	5994	41
	Iterative LUT	0.050	324	324	0.0	22	1.9	2.10	898	38
	Iterative GF	0.037	322	322	0.0	22	1.9	3.93	477	40
	Partially unrolled LUT	0.166	359	359	0.0	11	4.2	15.80	264	36
	Partially unrolled GF	0.112	195	195	0.0	11	2.3	20.00	113	39
ER-1024	Pipeline LUT	0.455	879	840	−4.4	38	107.5	36.70	2930	40
	Pipeline GF	0.239	662	580	−12.4	38	74.2	19.00	3907	41
	Iterative LUT	0.053	427	427	0.0	38	1.4	2.00	719	39
	Iterative GF	0.040	423	423	0.0	38	1.4	3.79	376	40
	Partially unrolled LUT	0.283	271	271	0.0	19	1.8	29.80	61	34
	Partially unrolled GF	0.187	188	188	0.0	19	1.3	35.30	36	39

reported implementation conditions rather than power at a common operating frequency.

Most evaluated architectures retained the pre-layout frequency as their post-layout implementation target. The only exceptions were six deeply pipelined configurations: the AES-256, ER-512 and ER-1024 Pipeline LUT designs and their corresponding Pipeline GF variants. For the Pipeline LUT designs, the target frequency was reduced relative to the pre-layout estimate by 0.3%, 3.4% and 4.4% for AES-256, ER-512 and ER-1024, respectively. The corresponding Pipeline GF reductions were more pronounced, reaching 9.9%, 10.9% and 12.4%.

These six designs achieved physical implementation at the reduced target frequencies reported in Table 5. All iterative and partially unrolled configurations, together with the AES-128 and AES-192 pipelined configurations, retained their respective pre-layout frequencies. The increasing frequency reduction observed in the deeper pipelines indicates that routing delay, clock-tree complexity and interconnect loading become more significant as the number of replicated round stages increases.

4.5. Critical-path analysis

Table 6 summarizes representative worst post-route setup paths for the AES-128 and ER-1024 implementations at the Nangate45 typical corner (1.10 V, 25 °C). The analysis used the final routed OpenROAD databases, extracted SPEF parasitics and final timing constraints. Each path was traced through the synthesized hierarchy, and its data-path delay was decomposed into cell-arc and extracted data-net contributions. This decomposition excludes launch and capture clock insertion, clock-reconvergence pessimism removal, setup time and external input delay.

Cell-arc delay dominates the reported paths, with the data-net fractions listed in Table 6 ranging from 0.3% to 4.8%. Routing also affects cell delay indirectly through load capacitance and signal transition time.

The critical-path structure varies with the architecture. Pipelined paths lie between adjacent round registers, while AES-128 iterative and partially unrolled paths traverse one and two rounds, respectively. In ER-1024, the LUT-based iterative path passes through round-key/state selection, and the partially unrolled path passes through key/output selection. Their GF counterparts retain one- and two-round data cones. These results suggest that larger key and control selectors can become timing bottlenecks as the round count increases.

Within the representative GF paths, major cell-delay contributors include the composite-field S-box and subsequent round logic. Technology mapping combines portions of the S-box, MixColumns and AddRoundKey logic, preventing reliable attribution of delay percentages to individual RTL transformations.

Post-layout timing analysis uses the complete implemented cores under the fixed-key operating assumption. Paths originating at `key_in` are constrained as false paths because the master key and derived round keys are assumed stable during block processing. The reported critical paths therefore characterize operation after key initialization. The key-expansion circuitry remains within the implementation boundary used for area and power estimation.

4.6. Comparison with prior work

Table 7 summarizes representative ASIC results reported in the literature using the values pre-layouts from Table 4, while Table 8 provides contextual references for FPGA implementations. Because the FPGA results were obtained using heterogeneous devices and implementation flows, they are not used for direct performance ranking.

From the ASIC perspective, the evaluated composite-field pipelined architecture shows the best balance between throughput and silicon cost. As shown in Table 7, the AES-128 implementation reaches 86 Gb/s while occupying 0.052 mm^2 corresponding to an area efficiency of approximately 1652 Gb/s/mm^2, exceeding the values reported by several

Table 6
Post-route static timing estimates at the Nangate45 typical corner. The net fraction is $100d_{net}/(d_{cell} + d_{net})$, expressed as a percentage.

Config.	Design	d_{cell} (ns)	d_{net} (ns)	Net (%)	Setup slack (ns)	Critical timed cone
AES-128	Pipeline LUT	1.094	0.018	1.6	0.061	Mapped LUT round cone
AES-128	Pipeline GF	1.402	0.008	0.6	0.008	GF S-box and round logic
AES-128	Iterative LUT	1.183	0.018	1.5	2.268	One-round state feedback
AES-128	Iterative GF	1.913	0.005	0.3	1.282	One-round state feedback
AES-128	Partially unrolled LUT	2.094	0.045	2.1	1.343	Two-round data cone
AES-128	Partially unrolled GF	3.229	0.019	0.6	0.237	Two-round data cone
ER-1024	Pipeline LUT	1.043	0.026	2.4	0.045	Mapped LUT round cone
ER-1024	Pipeline GF	1.647	0.011	0.7	0.013	GF S-box and round logic
ER-1024	Iterative LUT	1.678	0.032	1.9	0.570	Round-key/state selection
ER-1024	Iterative GF	1.834	0.009	0.5	0.457	One-round state feedback
ER-1024	Partially unrolled LUT	2.563	0.129	4.8	0.957	Key/output selection
ER-1024	Partially unrolled GF	3.445	0.029	0.8	1.808	Two-round data cone

Table 7
Pre-layout comparison with representative AES ASIC implementations across multiple key sizes and technology nodes.

AES key	Work	Tech (nm)	Area (mm^2)	Freq. (MHz)	Throughput (Gb/s)	Area Eff. ($Gb/s/mm^2$)
AES-128						
128	Hodjat et al. [30]	180	0.790	330	3.84	4.86
128	Wang et al. [31]	250	6.290	66	0.845	0.13
128	Mathew et al. [3]	45	0.150	2100	53	353
128	Sayilar and Chiou [6]	45	6.320	1000	128	20
128	Dong et al. [7]	45	0.130	870	111	854
128	Dong et al. [32]	45	1.040	800	1024	983
128	Lee et al. [33]	32	0.024	1111	–	–
128	This work: Pipeline GF	45	0.052	671	86	1652
AES-192/AES-256						
192	Wang et al. [31]	250	6.290	66	0.704	0.11
192	Mathew et al. [3]	45	0.150	2100	44	293
192	Lee et al. [33]	32	0.032	1099	–	–
192	This work: Pipeline GF	45	0.064	671	86	1342
256	Wang et al. [31]	250	6.290	66	0.603	0.10
256	Mathew et al. [3]	45	0.150	2100	38	253
256	Lee et al. [33]	32	0.035	1087	–	–
256	This work: Pipeline GF	45	0.073	666	85	1168

Table 8
Contextual AES-128 FPGA results. Literature values use different devices, tools and timing constraints and are not directly comparable, this work's results use a common ECP5 device and tool flow.

Reference	FPGA device	Reported freq. (MHz)	LUTs	Reported throughput (Gbit/s)
Qu et al. [34]	XC5VLX85	576.07	22 994	73.73
Elsayed et al. [35]	XC4VLX200	250.00	86 806	32.00
Fan et al. [36]	XC5VLX110T	272.59	11 677	34.89
Guzman et al. [37]	XC2VP30	405.28	6361	108.59
Liu et al. [38]	XC7VX690T	516.80	3436	66.10
Haroldo et al. [39]	XC7Z020	540.54	10 102	69.19
This work: Pipeline GF	ECP5-85F	64.37	24 426	8.24
This work: Iterative GF	ECP5-85F	67.76	5306	0.87

prior implementations. Some designs achieve higher absolute throughput, such as the multi-core architecture presented by Dong et al. [32]. However, these gains are primarily obtained through the replication of multiple AES processing cores operating in parallel, which increases the overall silicon area almost proportionally to the throughput improvement, which makes the throughput efficiency almost constant. In contrast, the results obtained in this work demonstrate that the combination of composite-field arithmetic and pipelining can improve throughput while maintaining a relatively compact hardware footprint.

The FPGA results in Table 8 are presented exclusively as contextual references and not as a normalized performance comparison. The reported implementations target different FPGA manufacturers, device families, generations and speed grades and were produced using different synthesis, placement and routing toolchains. They may also use different timing constraints, architectural organizations and resource-reporting conventions. In particular, the logic cells, routing resources and dedicated hardware available in the Lattice ECP5 device differ substantially from those of the high-end Xilinx devices used by several referenced studies. Consequently, LUT count, maximum frequency and throughput cannot be compared directly across rows, and the table is not used to establish a performance or efficiency ranking.

The literature values are reproduced as reported by their respective authors and indicate only the range of results obtained on different FPGA platforms. Comparisons among the implementations developed in this work remain meaningful because they use the same ECP5-85F device, open-source toolchain and implementation constraints.

Within this controlled environment, the results illustrate the effects of architectural organization and S-box implementation on frequency and resource utilization.

Accordingly, the FPGA contribution of this work is the demonstration of successful mapping and reproducible evaluation using an entirely open-source flow. The ECP5 results provide a platform-specific baseline for future experiments and should not be interpreted as being directly faster, slower or more resource-efficient than implementations targeting different FPGA technologies.

Overall, the ASIC and FPGA evaluations serve complementary purposes. The ASIC results quantify architectural trade-offs under a common 45 nm implementation flow, whereas the ECP5 results establish a reproducible, platform-specific FPGA baseline using open-source tools. The FPGA values collected from prior studies are retained solely to provide context regarding previously reported implementations; no cross-device performance or resource-efficiency ranking is inferred from them.

5. Conclusion

This work presented a unified and reproducible benchmark of established hardware architectures for the standardized AES-128, AES-192 and AES-256 configurations and the non-standard ER-512 and ER-1024 hardware-scaling cases across ASIC and FPGA targets. Six architecture and S-box implementation combinations were evaluated using a common RTL framework and consistent open-source implementation flows, enabling a controlled analysis of architectural replication and round-scaling effects.

The results demonstrate that architectural organization strongly influences hardware efficiency. Fully pipelined round replication largely preserves steady-state throughput as the number of rounds increases, at the cost of approximately proportional area growth. In contrast, iterative architectures maintain a comparatively compact silicon area while their steady-state throughput generally decreases as the initiation interval increases with the number of rounds. Partially unrolled architectures occupy an intermediate region of the design space, providing a compromise between performance and implementation cost.

Across the non-standard extended Rijndael configurations, the deeply pipelined architectures maintained broadly similar steady-state throughput while silicon area increased with the number of replicated round stages. Qualitatively similar scaling trends were observed in the ASIC and FPGA evaluations, suggesting that these architectural trade-offs persist across the two evaluated targets, although their absolute area and performance values remain dependent on the target technology and implementation flow.

5.1. Scientific contributions

The contribution of this work is a reproducible hardware benchmarking methodology rather than a new AES round transformation, S-box construction or microarchitecture. Iterative, partially unrolled and fully pipelined datapaths, together with LUT-based and composite-field S-boxes, are established implementation techniques. Their use within a common parameterized RTL framework enables their architectural trade-offs to be evaluated under controlled and consistent conditions.

The study contributes a unified evaluation spanning the standardized AES-128, AES-192 and AES-256 configurations and the non-standard ER-512 and ER-1024 hardware-scaling cases. The same architectural descriptions are characterized using post-synthesis and post-route ASIC flows and are mapped separately to an ECP5 FPGA. This provides a consistent dataset for examining the effects of round reuse, round replication, S-box implementation and increasing round count on area, timing, throughput, power and implementation efficiency.

A further contribution is the use of a reproducible open-source methodology covering functional verification, RTL synthesis, static timing analysis, physical implementation and FPGA mapping. The resulting framework and implementation data provide a baseline that can be reproduced, inspected and extended in future hardware-architecture studies.

5.2. Challenges and limitations

The FPGA implementations support functional verification, rapid prototyping and architectural validation. Their operating frequencies and resource requirements depend on the selected device and its programmable logic and routing resources. These results therefore support comparisons within the evaluated FPGA platform but cannot be directly extrapolated to ASIC performance or silicon area.

The ASIC evaluation is constrained by the selected technology and implementation flow. Open-source EDA tools improve reproducibility and accessibility, but the availability of public libraries and process design kits limits the range of technologies that can be evaluated. Results obtained with the selected 45 nm library should therefore be interpreted as technology-dependent reference points, without assuming equivalent performance in advanced commercial processes.

The evaluated cores are unprotected architectural baselines. No differential or correlation power analysis, electromagnetic assessment, leakage-detection testing or fault-injection experiments were performed. Although LUT-based and composite-field S-boxes are functionally equivalent, differences in gate structure, logic depth, switching activity, glitch propagation and routing may affect their leakage and fault-propagation characteristics. Aggregate post-layout power estimates alone cannot establish resistance to these attacks.

Dedicated countermeasures, including masking, hiding, temporal or spatial redundancy and fault detection, were not implemented. Their area, power and performance overheads are consequently excluded from the reported results. Security-critical deployment would require protected implementations supported by dedicated leakage measurements and fault-injection evaluation.

Finally, cryptanalytic assessment of the two extended architectures was not performed, as it lies outside the scope of this hardware study. Their functional verification checks implementation consistency but does not establish resistance to attack. The reported results therefore provide neither security certification nor practical-deployment guidance for these configurations.

5.3. Future work

Future research will focus on extending the proposed open-source hardware evaluation methodology to other cryptographic algorithms, including modern lightweight ciphers and stream ciphers suitable for resource-constrained systems.

A particularly promising direction is the development of a multi-layer hardware security architecture inspired by frameworks such as PRISEC, where multiple cryptographic primitives can be dynamically combined according to different security requirements. Such approaches have primarily been explored at the software and system levels, a fully hardware implementation at ASIC level remains largely unexplored and could provide significant improvements in performance, energy efficiency and security assurance for IoT or high throughput applications.

Future work will also investigate the integration of hardware entropy sources, including TRNGs and PUF-based key generation mechanisms, enabling complete hardware security infrastructures capable of secure key generation, storage and cryptographic processing.

Finally, the adopted open-source workflow can be extended to more advanced technology nodes, additional FPGA families and emerging open-source physical design frameworks, further improving reproducibility and accessibility within the hardware security research community.

CRediT authorship contribution statement

Dinis Santos: Writing – original draft, Software, Investigation. **Joao Cabacinho:** Validation. **Joao Casaleiro:** Validation. **Valderi R.Q. Leithardt:** Supervision, Software. **Luis Bica Oliveira:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request. And the complete implementation flow and experimental results are publicly available in the accompanying repository https://github.com/Denishes/crypto-hardware-benchmark.

References

[1] National Institute of Standards and Technology, Advanced Encryption Standard (AES), Tech. Rep. FIPS 197-upd1, National Institute of Standards and Technology, 2023, http://dx.doi.org/10.6028/NIST.FIPS.197-upd1.

[2] J. Daemen, V. Rijmen, The rijndael block cipher: AES proposal, 2003, (Accessed: 27 April 2026) http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf.

[3] S.K. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S.K. Hsu, H. Kaul, M.A. Anders, R.K. Krishnamurthy, 53 Gbps native GF$(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors, IEEE J. Solid-State Circuits 46 (4) (2011) 767–776, http://dx.doi.org/10.1109/JSSC.2011.2108131.

[4] M. Adam, M. Hammoudeh, R. Alrawashdeh, B. Alsulaimy, A survey on security, privacy, trust, and architectural challenges in IoT systems, IEEE Access 12 (2024) 57128–57149, http://dx.doi.org/10.1109/ACCESS.2024.3382709.

[5] A. Satoh, S. Morioka, K. Takano, S. Munetoh, A compact rijndael hardware architecture with S-box optimization, in: Advances in Cryptology – ASIACRYPT 2001, Springer, 2001, pp. 239–254, http://dx.doi.org/10.1007/3-540-45682-1_15.

[6] G. Sayilar, D. Chiou, Cryptoraptor: High throughput reconfigurable cryptographic processor, in: 2014 IEEE/ACM International Conference on Computer-Aided Design, 2014, pp. 155–161, http://dx.doi.org/10.1109/ICCAD.2014.7001346.

[7] P.-K. Dong, H.K. Nguyen, X.-T. Tran, A 45nm high-throughput and low latency AES encryption for real-time applications, in: 2019 19th International Symposium on Communications and Information Technologies, 2019, pp. 196–200, http://dx.doi.org/10.1109/ISCIT.2019.8905235.

[8] C. Wolf, J. Glaser, Yosys – a free verilog synthesis suite, Proc. Austrochip (2013) URL https://yosyshq.net/yosys/files/yosys-austrochip2013.pdf (Accessed: 27 April 2026) (Accessed: 27 April 2026).

[9] Silicon Integration Initiative, Open cell and free PDK libraries, 2024, (Accessed: 27 April 2026) https://si2.org/open-cell-library.

[10] T. Ajayi, D. Blaauw, R. Dreslinski, et al., Openroad: Toward a self-driving, open-source digital layout implementation tool chain, in: Government Microcircuit Applications and Critical Technology Conference (GOMACTech), 2019, pp. 1105–1110, URL https://theopenroadproject.org/publications/ (Accessed: 27 April 2026).

[11] The OpenROAD Project, The openroad project, 2024, (Accessed: 27 April 2026) https://theopenroadproject.org/.

[12] Lattice Semiconductor, ECP5 / ECP5-5G FPGAs, 2024, (Accessed: 27 April 2026) https://www.latticesemi.com/en/Products/FPGAandCPLD/ECP5.

[13] D. Santos, J. Cabacinho, J. Casaleiro, L.B. Oliveira, Architectural evaluation of iterative and unrolled aes-128 in 45 nm using an open-source flow, in: 2026 33rd International Conference on Mixed Design of Integrated Circuits and System (MIXDES), MIXDES, 2026, pp. 59–43, http://dx.doi.org/10.23919/MIXDES69535.2026.11617228.

[14] D. Santos, Crypto hardware benchmark: Reproducible FPGA and ASIC evaluation framework, 2026, GitHub repository, accessed 2026 https://github.com/Denishes/crypto-hardware-benchmark.

[15] D. Canright, A very compact S-box for AES, in: Cryptographic Hardware and Embedded Systems – CHES 2005, Springer, 2005, pp. 441–455, http://dx.doi.org/10.1007/11545262_32.

[16] B. Sarkar, A. Saha, D. Dutta, G. De Sarkar, K. Karmakar, A survey on the advanced encryption standard (AES): A pillar of modern cryptography, Int. J. Comput. Sci. Mob. Comput. 13 (4) (2024) 68–87, http://dx.doi.org/10.47760/ijcsmc.2024.v13i04.008.

[17] J. Cho, S. Soekamtoputra, K. Choi, J. Moon, Power dissipation and area comparison of 512-bit and 1024-bit key AES, Comput. Math. Appl. 65 (9) (2013) 1378–1383, http://dx.doi.org/10.1016/j.camwa.2012.01.035.

[18] J. Cabacinho, J. Casaleiro, L. Oliveira, Design of a 28nm CMOS self-biased ring oscillator for intrinsically robust PVT TRNG, in: Proceedings of the 18th International Conference on Ph.D Research in Microelectronics and Electronics, PRIME 2023, IEEE, 2023, pp. 225–228, http://dx.doi.org/10.1109/PRIME58259.2023.10161976.

[19] R. Oliveira, J. Cabacinho, L. Oliveira, J. Casaleiro, True random number generator prototype implemented in an FPGA, in: Proceedings of the 2023 International Young Engineers Forum in Electrical and Computer Engineering, YEF-ECE, IEEE, 2023, pp. 95–99, http://dx.doi.org/10.1109/YEF-ECE58420.2023.10209297.

[20] J. Cabacinho, J. Casaleiro, L.B. Oliveira, A physically unclonable function systematic performance analysis methodology, in: IFIP Advances in Information and Communication Technology, Vol. 759, Springer, 2025, pp. 395–409, http://dx.doi.org/10.1007/978-3-031-97051-1_28.

[21] B.J. Kang, H.I. Lee, S.K. Yoon, Y.C. Kim, S.B. Jeong, S.J. O, H. Kim, A survey of FPGA and ASIC designs for transformer inference acceleration and optimization, J. Syst. Archit. 155 (2024) 103247, http://dx.doi.org/10.1016/j.sysarc.2024.103247, URL https://www.sciencedirect.com/science/article/pii/S138376212400184X.

[22] Z. Mishra, B. Acharya, High throughput novel architectures of TEA family for high speed IoT and RFID applications, J. Inf. Secur. Appl. 61 (2021) 102906, http://dx.doi.org/10.1016/j.jisa.2021.102906.

[23] L. Zhou, M. Zhao, X. Lin, H. Chen, W. Wang, Y. Han, A hash-based signature scheme with layer-specific configuration for secure boot in IoT devices, J. Inf. Secur. Appl. 100 (2026) 104461, http://dx.doi.org/10.1016/j.jisa.2026.104461.

[24] A.M. Pasikhan, J.A. Clark, P. Gope, Incremental hybrid intrusion detection for 6lowpan, Comput. Secur. 135 (2023) 103447, http://dx.doi.org/10.1016/j.cose.2023.103447.

[25] C. Shepherd, E.A.J. Hurley, Entropy collapse in mobile sensors: The hidden risks of sensor-based security, J. Inf. Secur. Appl. 95 (2025) 104272, http://dx.doi.org/10.1016/j.jisa.2025.104272.

[26] P. da Costa, D. Noetzold, R.G. Ovejero, R.M. Esteban, V.R.Q. Leithardt, Prisec II: A comprehensive model for IoT security, in: Smart Systems Design, Applications, and Challenges, Springer, 2025, pp. 147–158, http://dx.doi.org/10.1007/978-3-031-99474-6_13.

[27] H. Sohail, V.R.Q. Leithardt, A. Trigo, PRISEC III: Cryptographic techniques for enhanced security, in: Proceedings of the 2025 25th International Conference on Control Systems and Computer Science, CSCS, IEEE, 2025, pp. 354–361, http://dx.doi.org/10.1109/CSCS66924.2025.00059.

[28] H. Sohail, D. Noetzold, V.R.Q. Leithardt, PRISEC III: Dynamic cryptographic adaptation for balancing performance and security, J. Internet Serv. Appl. 17 (1) (2026) 174–191, http://dx.doi.org/10.5753/jisa.2026.6502, URL https://journals-sol.sbc.org.br/index.php/jisa/article/view/6502.

[29] E. Konstantopoulou, G. Athanasiou, N. Sklavos, Review and analysis of FPGA and ASIC implementations of nist lightweight cryptography finalists, ACM Comput. Surv. 57 (10) (2025) 246:1–246:35, http://dx.doi.org/10.1145/3721122.

[30] A. Hodjat, D.D. Hwang, B. Lai, K. Tiri, I. Verbauwhede, A 3.84 gbits/s AES crypto coprocessor with modes of operation in a 0.18-μ m CMOS technology, in: Proceedings of the 15th ACM Great Lakes Symposium on VLSI, GLSVLSI '05, Association for Computing Machinery, New York, NY, USA, 2005, pp. 60–63, http://dx.doi.org/10.1145/1057661.1057677, URL https://doi.org/10.1145/1057661.1057677.

[31] M.-Y. Wang, C.-P. Su, C.-L. Horng, C.-T. Huang, Single- and multi-core configurable AES architectures for flexible security, Very Large Scale Integr. (VLSI) Syst. IEEE Trans. on 18 (2010) 541–552, http://dx.doi.org/10.1109/TVLSI.2009.2013231.

[32] P.K. Dong, H.K. Nguyen, F.A. Hussin, X.-T. Tran, Ultra-high-throughput multi-core AES encryption hardware architecture, VNU J. Sci.: Comput. Sci. Commun. Eng. 37 (2) (2021) 32–46, http://dx.doi.org/10.25073/2588-1086/vnucsce.290.

[33] D. Lee, M. Kwak, J. Lee, B. Kim, Y. Kim, A light-weight AES design using LFSR-based S-box for IoT applications, IEIE Trans. Smart Process. Comput. 11 (2) (2022) 140–148, http://dx.doi.org/10.5573/IEIESPC.2022.11.2.140.

[34] S. Qu, G. Shou, Y. Hu, Z. Guo, Z. Qian, High throughput, pipelined implementation of AES on FPGA, in: 2009 International Symposium on Information Engineering and Electronic Commerce, IEEE, 2009, pp. 542–545, http://dx.doi.org/10.1109/IEEC.2009.141.

[35] G. Elsayed, S. Elramly, K. Shehata, An efficient implementation of CBC mode rijndeal AES on an FPGA, ISBN: 978-977-5031-95-2, 2008, pp. 1–8, http://dx.doi.org/10.1109/NRSC.2008.4542373.

[36] C.-P. Fan, J.-K. Hwang, Implementations of high throughput sequential and fully pipelined aes processors on FPGA, in: Proceedings of the 2007 International Symposium on Intelligent Signal Processing and Communication Systems, 2007, pp. 353–356, http://dx.doi.org/10.1109/ISPACS.2007.4445896.

[37] I.C. Guzman, R.D. Nieto, A.B. Norena, FPGA implementation of the AES-128 algorithm in non-feedback modes of operation, DYNA 83 (198) (2016) 37–43, http://dx.doi.org/10.15446/dyna.v83n198.55251.

[38] Q. Liu, Z. Xu, Y. Yuan, A 66.1 gbps single-pipeline AES on FPGA, in: FPT 2013 - Proceedings of the 2013 International Conference on Field Programmable Technology, 2013, pp. 378–381, http://dx.doi.org/10.1109/FPT.2013.6718392.

[39] C.M. Haroldo, N.C. David, M. Madani, E.-B. Bourennane, FPGA implementation of AES-based on optimized dynamic S-box, in: Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024), SCITEPRESS, 2024, pp. 730–737, http://dx.doi.org/10.5220/0012780300003767.