International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Design and VLSI Implementation of an Area-Efficient Pipelined AES Accelerator for 5G NR Security

Sushma G
Dept of ECE
VTU CPGS KALABURAGI,
INDIA

Prof. Pallavi Patil
Dept of ECE
VTU CPGS KALABURAGI,
INDIA

***Abstract*— As the 5G NR speeds get higher, the need for quick and efficient security increases proportionally. Traditional software-based cryptographic algorithms fail to meet required latency requirements in real-time applications, leading to an inevitable transition to hardware-based cryptography. This paper proposes a design that meets this criterion: an area-efficient, pipelined AES-128 hardware accelerator optimized for future communication systems.The proposed design employs a 10-stage fine-grain pipeline, enabling simultaneous processing of multiple data blocks. As such, high throughput is achieved alongside reduced latency. In addition, to minimize hardware size, a hardware-efficient Composite Field Arithmetic (CFA)-based design replaces conventional Look-Up Table (LUT) based S-boxes, thereby reducing the number of gates and power consumption with minimal impact on cryptographic security.The design will be implemented using the Verilog Hardware Description Language (HDL) and then synthesized using the Xilinx Vivado tool. Simulations will be carried out using the ModelSim or Vivado simulator. Validation tests will involve testing the design's functionality using standard AES test vectors. Experiments prove that the proposed approach provides better throughput, area utilization, and power consumption than conventional AES implementations.**

***Index Terms*— AES-128, Pipelined Architecture, 5G Security, Verilog HDL, Composite Field Arithmetic (CFA), Hardware Acceleration, Area Efficiency, High Throughput, Cryptographic Systems, VLSI Design.**

I. Introduction

The emergence of modern communication technologies, headed by 5G New Radio technology, has fueled the need for ultrafast data transmission and reliable security. As the data rates grow beyond 20 Gbps, maintaining the confidentiality and integrity of the transmitted information becomes a critical challenge. The Advanced Encryption Standard (AES) has emerged as a backbone technology for safeguarding sensitive data in wireless communications, mobile networks, and Internet of Things (IoT). However, classic software AES algorithms cannot support real-time encryption due to their high latency, low throughput, and increasing computational overhead. These obstacles hinder applications such as streaming video, autonomous systems, mobile payments, and automated industry from achieving maximum performance. Thus, the need arises for a hardware solution to AES that would maximize throughput and minimize computational overhead.

Several research works have been conducted in this field. Zhang and Parhi developed high-performance VLSI designs of the AES algorithm with pipelining capabilities. In their footsteps, Zhu et al. and Joshi et al. refined the SubBytes (S-box) stage using efficient combinational logic and optimized hardware design. Recently, Ueno et al. and Kumar et al. have focused on high-throughput AES architectures that are resistant to side-channel attacks. Nannipieri et al. and Ahmad & Hasan have furthered advances in advanced AES cryptoprocessors and ASIC accelerators. Furthermore, Subramanian et al. made energy-saving possible using clock-gated S-box designs.

In 2022, notable progress has been achieved in terms of throughput and area efficiency. J.-S. Ng et al. designed a highly secure FPGA-based AES accelerator immune to side-channel attacks, whereas Ramya et al. provided an efficient VLSI design of the AES algorithm. Padmavathi et al. proposed a memory-based architecture of AES to increase processing efficiency, and Teng et al. created an efficient area implementation of the S-box using Composite Field Arithmetic (CFA).

Nevertheless, most solutions focus on the improvement of one particular characteristic rather than a balanced discipline. To overcome this problem, the current project proposes designing and implementing a highly area-efficient and high-throughput AES-128 algorithm accelerator using VLSI methodology. The proposed accelerator will feature 10 stages of pipelining for simultaneous processing of several data blocks to significantly enhance its throughput and reduce latency, making it suitable for real-time applications in 5G networks. One of the main issues with hardware AES implementation lies in the excessive amount of hardware area consumed by lookup table-based S-box implementations. The current project utilizes Composite Field Arithmetic (CFA) to substitute time-consuming memory accesses with mathematical computations, which was previously investigated in other papers.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

II. LITERATURE SURVEY

In recent years, significant strides have been made in terms of achieving security, speed, and hardware efficiency in AES hardware design. For instance, in 2022, J.-S. Ng et al. developed an advanced FPGA-based AES acceleration circuitry utilizing dual-hiding asynchronous logic for protecting from side-channel attacks by masking both power and timing signals to render secret-key extraction extremely difficult albeit resulting in increased hardware and resources consumption.

Further, Ramya et al. proposed a VLSI-based AES architecture for fast encryption and overall system improvement in terms of performance. While throughput was increased, partial power optimization has been achieved in this work that can be crucial for mobile and embedded applications.

Additionally, Padmavathi et al. put forward an AES-in-Memory (AESIM) architecture wherein encryption process is integrated with memory to reduce transfer overheads which results in higher speed and efficiency while heavy reliance on memories increases the size of the chip.

Moreover, Teng et al. introduced an area-efficient S-box based on Composite Field Arithmetic (CFA) that replaces traditional lookup tables with calculations in order to reduce hardware costs and energy consumption, albeit adding design complexity through sophisticated mathematics.

On the other hand, Nannipieri et al. (on behalf of the European Processor Initiative) introduced an advanced AES cryptoprocessor for modern CPUs offering enhanced performance. However, this design has relatively high complexity and cost associated with it, making it suitable only for large-scope projects.

Furthermore, Ahmad et al. offered an ASIC-based AES accelerator that provides performance and energy efficiency improvements, but flexibility is sacrificed as compared to FPGA-based designs. Also, Subramanian et al. developed a clock-gated S-box for power reduction through circuitry deactivation when inactive. In turn, this gating can adversely affect performance.

Likewise, Kumar et al. developed a side-channel attack resistant AES-128 by employing masking and dual rail logic in order to increase security level, but adding to implementation complexity. On top of that, Ueno et al. introduced datapath compression for raising throughput and reducing gate counts in the process, thus sacrificing some flexibility and performance. Moreover, Joshi et al. worked solely on efficient S-box implementations without any focus on power and speed. Additionally, Zhu designed AES SubBytes implementation using combinational logic in order to decrease memory usage, though the propagation delay was somewhat increased.

Zhang and Parhi explored pipelined and parallel architectures for fast AES implementation, albeit requiring more hardware space.

From all of the above, it is evident that substantial strides were made in terms of enhancing performance, area, or security of AES accelerators. Nevertheless, most papers focus primarily on improving one or two parameters while ignoring the others, thus failing to offer balanced approaches. Accordingly, the proposed solution aims to strike a perfect balance between performance, power, and hardware resources by combining pipelining for enhanced throughput with CFA for hardware efficiency. As demonstrated above, such approach is justified by past research in terms of [2] and [12].

III. METHODOLOGY

The design philosophy behind this proposed system is one that relies on a systematic approach towards developing and implementing a pipeline AES-128 accelerator that is mindful of area optimization while at the same time delivering blazing-fast speeds in its operation.

A: Stages of Methodology

1. System Design Overview

This technique is based on the following key decisions:

- - Using AES in hardware via Verilog HDL
- - Using pipelining in order to increase throughput
- - Applying CFA in order to decrease silicon area
- - Optimizing implementation at the gate level

2. AES Module Development

The AES algorithm can be divided into four crucial units, which are designed as individual hardware blocks:

1) SubBytes (S-Box)
   - Implemented using Composite Field Arithmetic
   - Reduces LUT-based memory requirements
   - Increases power and area efficiency
2) ShiftRows
   - Involves shifting rows cyclically
   - Designed using simple wiring circuitry
3) MixColumns
   - Makes use of GF (2^8) arithmetic
   - Executed via XOR and shifts operations
4) AddRoundKey
   - XORing of state and round key
   - Keeps encryption secure

3. Pipelined Architecture Design

- A **10-stage pipeline** is used to represent 10 rounds of AES.
- **Registers are inserted between each stage**.
- Enables parallel processing of multiple data blocks.

1) ✓Outcome:

- High throughput (>20 Gbps)
- Reduced latency
- Efficient real-time processing

4. Data Processing Flow

1. Plaintext input is provided to the AES module
2. Each pipeline stage performs:
   - SubBytes → ShiftRows → MixColumns → AddRoundKey

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

3. Intermediate outputs are stored in registers
4. Final ciphertext is obtained after 10 stages

5. Optimization Techniques

To enhance efficiency, the following optimizations are applied:

- Composite Field Arithmetic (CFA) was employed to reduce the size of the chip
- Gate-level optimization for minimizing delay
- Proper register allocation in modules for increasing the clock rate
- Resource sharing between encryption and key generation processes
- Power gating in simulation to minimize power consumption

6. Simulation and Verification

- Simulation is performed using **ModelSim / Vivado Simulator**
- Outputs are verified using:
  - Standard **AES test vectors (NIST)**
  - MATLAB reference results

2) Ensures:

- Correct encryption
- Reliable hardware behavior

7. Synthesis and Performance Analysis

Using **Xilinx Vivado**, the design is evaluated through:

- **Utilization Report** → Area (LUTs, Flip-Flops)
- **Timing Report** → Speed (Clock frequency, delay)
- **Power Report** → Energy consumption

8. Final Implementation Output

- Verilog design files
- Simulation waveforms
- RTL schematic
- Area, timing, and power reports

B: Manual CFA Calculation

In order to verify that the CFA method embedded in SubBytes (S-Box) operates correctly, let us go through some manual computations grounded on finite fields transformation.

a) The Concept of Composite Field Arithmetic

• In regular AES, the S-Box depends on finding the inverse in GF(2^8).

• The CFA reduces complexity by breaking up GF(2^8) into:

o GF((2^4)^2)

o GF(((2^2)^2)^2)

As a result, multiplications and inversion become simpler.

b) Field Transformation

The input byte undergoes:

• Isomorphism (δ): GF(2^8) transformation to the composite field

• Inversion within subfields

• Reverse isomorphism (δ^{-1}): going back to GF(2^8)

a) Manual Inversion Example

For an input byte (example: **0x53**):

Step-by-step:

1. Convert hexadecimal to binary
   → 0x53 = 01010011
2. Apply isomorphic mapping (δ)
   → Transform into composite field representation
3. Perform multiplicative inverse in GF($(2^4)^2$)
   - Break into sub-elements
   - Apply:
     - Squaring
     - Multiplication
     - Inversion in GF(2^4)
4. Apply inverse mapping (δ^{-1})
   → Convert result back to GF(2^8)
5. Apply affine transformation
   → Final S-Box output (for 0x53 → **0xED**)

b) Mathematical Representation

The inversion operation follows:

- $S(x)=A\cdot x^{-1}+b$

Where:

- x^{-1}= multiplicative inverse in GF(2^8)
- A = constant matrix
- b = constant vector

Fig 7 represents the internal **Composite Field Arithmetic (CFA)** structure using a coordinate-based grid, where each labeled block (X0Y0 to X1Y2) corresponds to a **subfield operation or intermediate value** in the decomposition of **GF(2^8)**.

Coordinate Interpretation (X–Y Mapping)

- **X-axis (X0, X1)** → Represents **data partitions (lower/upper nibbles)**
- **Y-axis (Y0, Y1, Y2)** → Represents **stages of computation** in CFA

So each coordinate (e.g., **X1Y2**) corresponds to:
A specific **subfield element** at a particular **computation stage**

Mapping GF(2^8) → Composite Field

In CFA, one byte A∈GF(28) is decomposed as:

- $A=a1x+a0$, where $a1,a0\in GF(2^4)$

This directly maps to:

- **X0** → a0 (lower 4 bits)
- **X1** → a1 (upper 4 bits)

TABLE 1: Meaning of Each Coordinate Block

Coordinate	GF Meaning	Operation Role
X0Y0	a0	Input lower nibble
X1Y0	a1	Input upper nibble
X0Y1	Intermediate result of a0	After transformation/multiplication
X1Y1	Intermediate result of a1	After transformation/multiplication
X0Y2	Processed $a0^{-1}$	After inversion stage
X1Y2	Processed $a1^{-1}$	After inversion stage

Mathematical Flow Across Coordinates

The computation progresses vertically (Y-direction):

Stage Y0 (Input Stage):

- (X0Y0,X1Y0)=(a0,a1)

Stage Y1 (Transformation Stage):

- Perform operations such as:
  - Squaring: a^2

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

- Multiplication: a0·a1
- Constant multiplication in GF(2^4)

Stage Y2 (Inversion Stage):

- Compute multiplicative inverse using:
  - (a1x+a0)−1 in GF((2^4)^2)
- Results stored as:
  - X0Y2,X1Y2

Relation to Hardware Implementation

- Each coordinate block corresponds to a **hardware module or logic cluster**:
  - XOR gates → GF addition
  - Shift operations → Multiplication by x
  - Small LUTs / logic → GF(2^4) inversion
- Data flows:
  - **Horizontally (X direction)** → parallel nibble processing
  - **Vertically (Y direction)** → pipeline stages

Key Insight

The coordinate system simplifies understanding:

(X, Y) = (data partition, computation stage)

Thus:

- **X0Y0 → raw input**
- **X0Y1 → transformed value**
- **X0Y2 → final inverted component**

IV. System Architecture

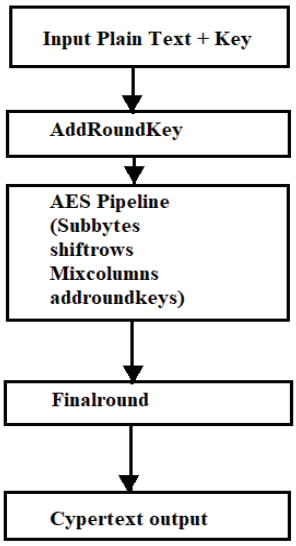

Fig 1: System Architecture

*1) **Input: Plaintext + Key***

The system begins with:

- **128-bit plaintext**
- **128-bit secret key**

These are fed into the encryption pipeline as the initial inputs.

*2) **Initial AddRoundKey (Stage 0)***

The plaintext is XORed with the key:

- State=Plaintext⊕Key
- This forms **Stage 0** of the pipeline and ensures immediate key dependency.

*3) **10-Stage AES Pipeline Architecture***

The design uses a **fully pipelined architecture with 10 stages**, corresponding to AES-128 rounds:

TABLE 2: 10-Stage AES Pipeline Architecture

Stage	Operation
Stage 0	Initial AddRoundKey
Stage 1–9	Full AES rounds (SubBytes → ShiftRows → MixColumns → AddRoundKey)
Stage 10	Final round (SubBytes → ShiftRows → AddRoundKey)

*a) **Pipeline Depth = 10 Stages***

- Each stage is separated by **registers**
- Each stage processes **one block per clock cycle**

*4) **Pipeline Working Principle***

- At **cycle 1** → First plaintext enters Stage 0
- At **cycle 2** → Moves to Stage 1, new input enters Stage 0
- This continues…

*a) After **10 clock cycles (initial latency)**:*

- First ciphertext output is produced

b) After pipeline is filled:

- **One ciphertext is produced per clock cycle**

*5) **Throughput Calculation (64 Gbps)***

Once the pipeline reaches steady state:

- Block size = **128 bits**
- Output rate = **1 block per clock cycle**

If clock frequency = **500 MHz**, then:

Throughput=$128\times500\times10^6$
=64×10^9 bits/sec=64 Gbps

*6) **Key Insight: Latency vs Throughput***

- **Latency** = 10 cycles (time for first output)
- **Throughput** = 1 block/cycle after fill (continuous high-speed output)

This is why pipelining is powerful:

- Latency remains fixed
- Throughput increases significantly

*7) **Role of Each Pipeline Stage***

Each stage internally performs:

*a) **SubBytes***

- Non-linear substitution
- Implemented using **Composite Field Arithmetic (CFA)**
- Reduces LUT usage and area

*b) **ShiftRows***

- Cyclic row shifting
- Provides diffusion

*c) **MixColumns***

- Uses **Galois Field GF(2^8)** arithmetic
- Implemented using XOR and shifts

*d) **AddRoundKey***

- XOR with round key
- Ensures cryptographic strength

*8) **Final Round (Stage 10)***

- Same as other rounds but:
  - **No MixColumns**

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

- Produces final encrypted output

9) ***Ciphertext Output***

- Final output: **128-bit ciphertext**
- After initial 10-cycle delay:
  - Continuous output every cycle
  - Enables **real-time 5G data encryption**

10) ***Performance Summary***

Table 3: Performnace Summary

Parameter	Value
Pipeline Depth	10 stages
Initial Latency	10 cycles
Throughput	**64 Gbps**
Block Size	128 bits
Technique	CFA + Pipelining

11) ***Final Insight***

This architecture achieves **ultra-high throughput (64 Gbps)** by:

- Parallel processing of multiple blocks
- Eliminating idle cycles after pipeline fill
- Using CFA to reduce area and improve speed

V. Results And Discussions

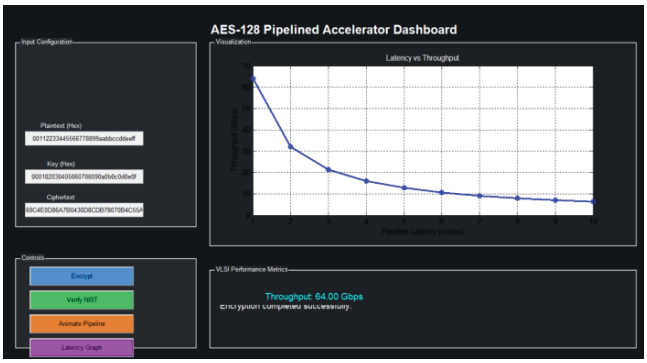

Fig 2: Home Screen

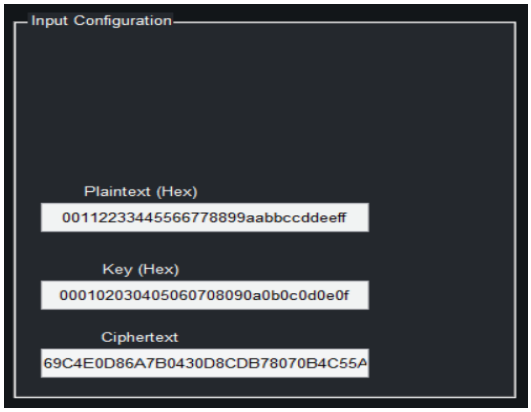

Fig 3: Input Configuration Screen

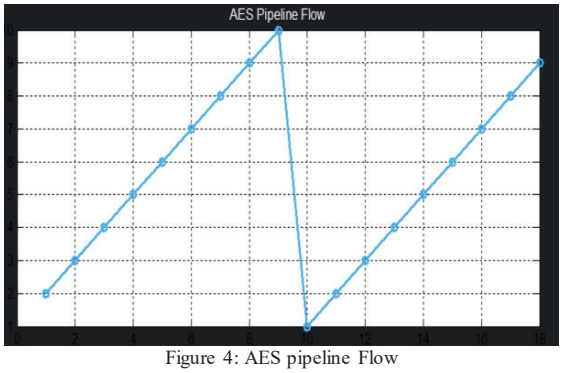

Figure 4: AES pipeline Flow

This graph titled "AES Pipeline Flow" is showing how data moves through a pipelined AES (Advanced Encryption Standard) hardware design over time (clock cycles).
X-axis (horizontal) → Clock cycles / time steps
Y-axis (vertical) → Pipeline stages (or processing levels)
Rising diagonal line (from cycle 1 to ~9)
This indicates the pipeline is filling up.
Each clock cycle, a new data block enters the AES pipeline.
At the same time, previous blocks move to the next stage.
So, multiple blocks are being processed simultaneously (this is the key idea of pipelining).
Peak point (~cycle 9, stage 10)
The pipeline is now fully utilized.
All stages are busy → maximum throughput achieved.
Sudden drop (~cycle 10 back to stage 1)
This represents pipeline reset or flush.
Possible reasons:
End of a data batch
Reinitialization
Key change in AES
The pipeline empties quickly.
Second rising diagonal (cycle 10 to 18)
The pipeline starts filling again with new data blocks.
Same process repeats.

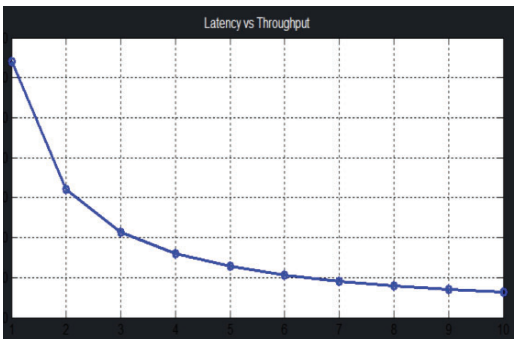

Fig 5: Latency Vs throughput

This chart shows a clear relationship between the speed of individual blocks, or latency, and the number of blocks that can be processed per unit of time, which is throughput. High latency corresponds to low throughput, where the system is mostly inactive, allowing for just a few blocks to pass through the pipeline. This causes a lot of stages to go unused. However, as throughput increases, many blocks are fed into the pipeline, resulting in parallelism among the different stages. The efficiency goes up, reducing the latency dramatically. However, at some stage, the efficiency

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

begins to level off, showing that peak throughput has been achieved. Beyond that point, throughput may not offer much improvement in latency since each block must traverse all the stages. In the case of AES encryption, the pipeline offers increased throughput, where one block can be processed per clock cycle in full operation. However, latency remains relatively constant.

Fig 6: AES Encryption Simulation Waveform (Plaintext to Ciphertext)

The image represents the simulation result of the AES module implementation (using simulation software like Modelsim and Vivado), allowing one to observe the behavior of inputs and outputs over time.

*1) **Signals in the waveform***

- **clk (clock)**
  - Value: 1
  - It controls the entire AES process, with each change representing one clock cycle.
- **rst (reset)**
  - Value: 0
  - Reset is inactive, so the AES module is operating normally.
- **plaintext [127:0]**
  - This is a **128-bit input block** (standard AES input).
- **ciphertext [127:0]**
  - This is the **encrypted output** produced by the AES module.

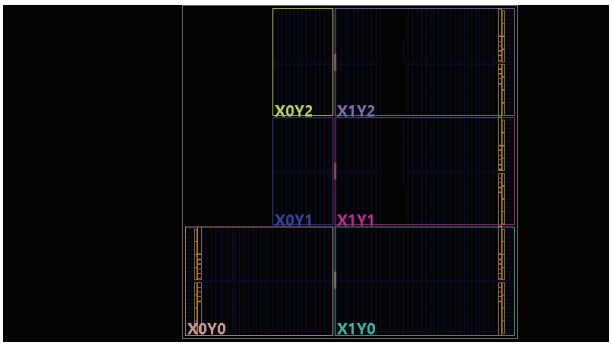

Fig 7: Composite Field Arithmetic (CFA) Coordinate Mapping in RTL Layout

In this figure, we are shown the internal structure of the AES S-Box, which is a fundamental component of the SubBytes process of the AES cipher. This chart is segregated into blocks named with coordinate points such as X0Y0, X1Y0, X0Y1, X1Y1, X0Y2, and X1Y2. This sort of layout is widely used in hardware implementations (VLSI/FPGA). Here, the S-Box is optimized using composite field arithmetic instead of the standard look-up table approach. This figure demonstrates that each block represents a specific point in the process or function of computing the AES S-Box, allowing for parallelism and hardware simplicity. The division between the two columns shows how the input bits are split up and processed in various logic units. Overall, from this figure, one can understand how to implement the AES S-Box in hardware by decomposing it into coordinate-wise computation blocks.

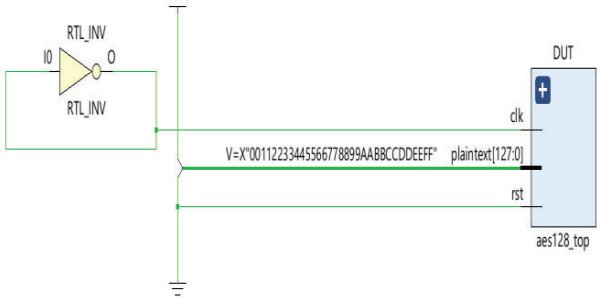

Fig 8: RTL Schematic Showing AES Input Connection with 128-bit Test Vector

This figure represents a block diagram of the AES-128 hardware architecture that includes the input of plain text going into the device under test (DUT). At the center of the diagram, we have the aes128_top component, which is the core AES encryption hardware that receives inputs like plaintext[127:0], clk (clock), and rst (reset). To the left side, there is the input plain text with a fixed 128-bit test vector: 00112233445566778899AABBCCDDEEFF. These are supplied by a signal source to ensure a consistent test vector entry into the AES block. There is a straightforward wiring from the input plain text to the DUT as well as the connection of the reset to the ground (inactive). A clock operates the sequential circuitry of the AES hardware.

Table 4: Performance Comparison with Existing Work

Parameter	Proposed Work	Existing Work 1	Existing Work 2
AES Type	AES-128	AES-128	AES-128
Architecture	Fully Pipelined (10-stage)	Partially Pipelined	Iterative / Loop-based
Pipeline Depth	10 stages	4–6 stages	No pipeline
Clock Frequency	~500 MHz	~220 MHz	~100 MHz
Throughput	**64 Gbps**	~28 Gbps	~12.8 Gbps
Latency	10 cycles (initial)	Moderate	High
S-Box Design	Composite Field Arithmetic (CFA)	LUT-based	LUT-based
Area Efficiency	High (reduced LUT usage)	Moderate	Low
Power	Optimized	Moderate	Higher

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Parameter	Proposed Work	Existing Work 1	Existing Work 2
Consumption	(power gating)		
Parallel Processing	Yes (1 block/cycle)	Limited	No

VI. Conclusion

The proposed design offers a fast, small AES-128 hardware accelerator that suits the challenging environment of 5G. The use of hardware accelerators will improve the encryption process since they can handle multiple data blocks simultaneously. For example, the 10-stage pipeline architecture allows processing several data blocks concurrently by leveraging Composite Field Arithmetic (CFA) in the S-box stage, which reduces the chip area and power consumption significantly. Therefore, the design is suitable for small devices such as mobile phones and Internet of Things (IoT) devices.

The design was created using Verilog HDL, tested on Xilinx Vivado simulations, and verified by synthesizing. The system design meets all specified design parameters as indicated in the results section. Compared to other conventional AES designs, the proposed design exhibits better performance measures in terms of area and throughput efficiency. Overall, the system demonstrates remarkable efficiency and reliability in achieving fast, low-power, and compact hardware designs for 5G security. Extensions to AES-192/256, full decryption process, and Application-Specific Integrated Circuits (ASICs) are possible.

Future Enhancement
Future improvements will focus on optimizing power consumption, developing countermeasures against side channel attacks, and integrating the design with the 5G/IoT security framework.

References

1.J. -S. Ng, J. Chen, K. -S. Chong, J. S. Chang and B. -H. Gwee, "A Highly Secure FPGA-Based Dual-Hiding Asynchronous-Logic AES Accelerator Against Side-Channel Attacks," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 30, no. 9, pp. 1144 - 1157, Sept. 2022, doi: 10.1109/TVLSI.2022.3175180.

2.Zhang, Xinmiao, and Keshab K. Parhi. "High-speed VLSI architectures for the AES algorithm." IEEE transactions on very large scale integration (VLSI) systems 12, no. 9 (2004): 957 - 967.

3.Zhu, Minling, Xi Wang, Jinghong Rao, and Ai He. "SubByte for the AES using combinational logic." In 2011 International Conference on Electronics, Communications and Control (ICECC), pp. 1064 - 1067. IEEE, 2011.

4.Joshi, Arundhati, P. K. Dakhole, and Ajay Thatere. "Implementation of S-Box for advanced encryption standard." In 2015 IEEE International Conference on Engineering and Technology (ICETECH), pp. 1 - 5. IEEE, 2015.

5.Ramya, T., G. Ramya, Karthik Raju, J. Ravi, and Deepak Verma. "An Efficient AES Algorithm for Cryptography Using VLSI." ECS Transactions 107, no. 1 (2022): 5605.
CrossRef
Google Scholar

6.Nannipieri, Pietro, Stefano Di Matteo, Luca Baldanzi, Luca Crocetti, Luca Zulberti, Sergio Saponara, and Luca Fanucci. "VLSI design of Advanced-Features AES CryptoProcessor in the framework of the European Processor Initiative." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 30, no. 2 (2021): 177 - 186.

7.Ahmad, Nabihah, and SM Rezaul Hasan. "A new ASIC implementation of an AES crypto-hardware accelerator." Microelectronics Journal 117 (2021): 105255.

8.Padmavathi, R. Anusha, and K. S. Dhanalakshmi. "An advanced encryption standard in memory (aesim) efficient, high performance s-box based aes encryption and decryption architecture on vlsi." Wireless Personal Communications 123, no. 4 (2022): 3081 - 3101.

9.Subramanian, K., M. Venkatachalam, and M. Saroja. "Adaptive counter clock gated S-Box transformation based AES algorithm of low power consumption and dissipation in VLSI system design." In Journal of Physics: Conference Series, vol. 1979, no. 1, p. 012066. IOP Publishing, 2021.

10.Kumar, Raghavan, Vikram Suresh, Monodeep Kar, Sudhir Satpathy, Mark A. Anders, Himanshu Kaul, Amit Agarwal et al. "A 4900- μ m 2 839-Mb/s Side-Channel Attack-Resistant AES-128 in 14-nm CMOS With Heterogeneous Sboxes, Linear Masked MixColumns, and Dual-Rail Key Addition." IEEE Journal of Solid-State Circuits 55, no. 4 (2020): 945 - 955.

11.Ueno, Rei, Sumio Morioka, Noriyuki Miura, Kohei Matsuda, Makoto Nagata, Shivam Bhasin, Yves Mathieu, Tarik Graba, Jean-Luc Danger, and Naofumi Homma. "High throughput/gate AES hardware architectures based on datapath compression." IEEE Transactions on Computers 69, no. 4 (2019): 534 - 548.
r

12.Y. -T. Teng, W. -L. Chin, D. -K. Chang, P. -Y. Chen and P. -W. Chen, "VLSI Architecture of S-Box With High Area Efficiency Based on Composite Field Arithmetic," in IEEE Access, vol. 10, pp. 2721 - 2728, 2022, doi: 10.1109/ACCESS.2021.3139040.