International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 08 Issue: 06 | June - 2024 SJIF Rating: 8.448 ISSN: 2582-3930

Hardware Accelerated AES: RTL to GDS

Rakshan Kulkarni
Dept. of Electronics and Instrumentation
RV College of Engineering

Dr. Rachana S. Akki
Dept. of Electronics and Instrumentation
RV College of Engineering)

Abstract—This paper presents a hardware-accelerated Advanced Encryption Standard (AES) implementation featuring an 11-stage pipelined architecture and an optimized subkey generation scheme for enhanced performance. This design is implemented in Verilog and verified through waveform analysis. Operating at a maximum clock frequency of 100 MHz, our implementation achieves significant efficiency. The final design occupies an area of 1.44 mm^2 and consists of 32,424 standard cells, with a power consumption of 32 mW at the typical corner. Furthermore, the entire design flow, from synthesis to layout
(GDSII), is achieved using OpenLane, an open-source Electronic Design Automation (EDA) tool. This democratization of hardware design empowers a wider range of participants, particularly students and researchers, to efficiently implement their designs and gain hands-on experience without the constraints of expensive commercial tools. This fosters innovation and accelerates progress by allowing a broader community to contribute to technological advancements.

Index Terms—OpenLane, Open-source , EDA, Pipelined AES,
RTL to GDSII

I. INTRODUCTION

In today's digital landscape, encryption has become an indispensable tool for protecting sensitive information. It is is a fundamental aspect of modern digital security, ensuring the confidentiality and integrity of sensitive information in various applications, from personal data protection to secure communication in business and government. It acts as a shield, transforming data into an unreadable format, ensuring confidentiality, and preventing unauthorized access. The Advanced Encryption Standard (AES) stands out as a widely adopted encryption algorithm due to its robustness and efficiency. As a symmetric-key algorithm, it utilizes a single shared key for both encryption and decryption. AES offers a high level of security, making it the preferred choice for various applications[1]. From securing communication channels and protecting data at rest to implementing access control mechanisms, AES plays a vital role across numerous industries, including government, finance, defense, and healthcare[6]. Its importance is further understood by its use in numerous security protocols and systems, including SSL/TLS for internet security, VPNs for secure remote access, and encrypted storage solutions. The strength and versatility of AES make it a cornerstone of data protection in the digital age[14]. As network speeds soar into the gigabits-per-second (Gbps) range, software implementations of cryptographic algorithms struggle to keep pace[15]. These traditional methods can't handle the high data volumes efficiently. In [2], a comparative study shows that the hardware implementation of the AES algorithm is significantly faster when compared to its software implementation. In recent years, many hardware-based implementations have been proposed in literature[11-23]. Papers [3-5], implementing the said algorithm on different FPGA boards, successfully obtaining high throughput as high as around 30Gbps. Furthermore, hardware implementations offer an additional layer of security. By physically encapsulating the cryptographic algorithms and key generation within a chip, these solutions make it significantly harder for external attackers to tamper with or steal sensitive information. This enhanced physical security is a valuable advantage for protecting critical data.

II. AES ALGORITHM

The AES algorithm is a symmetric-key cipher, in which both the sender and the receiver use a single key for encryption and decryption. The AES encryption comes in three flavors, with three different key lengths of 128bits, 192bits, or 256bits. The data block length is fixed to be 128 bits, while the key length can be 128, 192, or 256 bits, respectively. The AES algorithm is an iterative algorithm, in which each iteration is named as a round and the total number of rounds can be either 10, 12 or 14 rounds depending on whether the key size is of 128bits, 192bits, or 256bits respectively. The 128-bit data block is divided

into 16 bytes. These bytes are mapped to a 4X4 array called the State, and all the internal operations of the AES algorithm are performed on the State[7]. The basic encryption and decryption block diagram of the AES algorithm is shown in Figure 1.

It takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data. A Key Schedule algorithm is used to calculate all the round keys from the key. So the initial key is used to create many different round keys which will be used in the corresponding round of the encryption. More on this is explained in the further sections. The steps involved in the algorithm of Encryption and Decryption are as follows:

A. SubBytes

In this step, each byte of the plain text array is replaced with a SubByte using an 8-bit Substitution box.It is replaced in hexadecimal bytes using substitution table/Inv Subtution Table for Encryption/Decryption Process respectively. It is the only

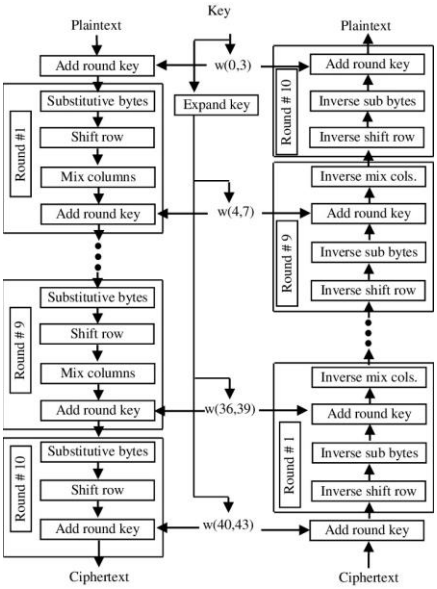

Fig. 1. Block Diagram of AES

Non-linear process in the algorithm. This is shown in Figure. 2.

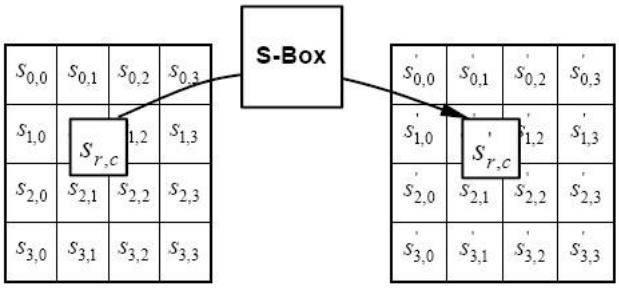

Fig. 2. SubBytes function

B. ShiftRow

In this step, the rows in the state arrays are shifted left cyclically by a certain offset. The first row remains unchanged as the offset is zero. The second, third, and fourth rows are shifted by offset of 1,2 and 3 respectively. The columns of output which is obtained after this step is composed of bytes from each column from the input state. In case of Decryption, the rows are shifted cyclically right with respective offset values. The importance of this step is to avoid the columns being encrypted independently, in which case AES degenerates into four independent block ciphers. Implementation of this transformation is shown in Figure.3

C. MixColumns

This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result. This step is skipped in the last round.

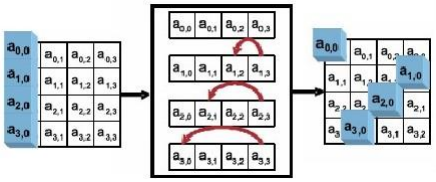

Fig. 3. ShiftRows Transformation

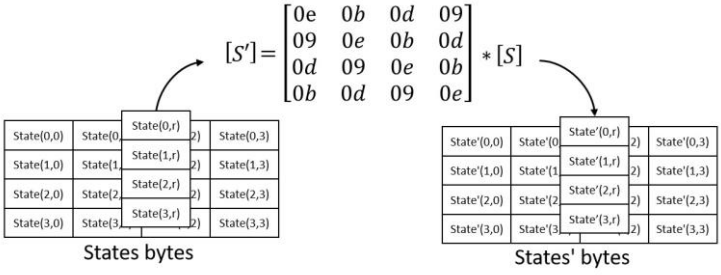

Fig. 4. MixColumns Operation

D. Add Round Key

Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes is not considered as a grid but just as 128 bits of data. This operation is demonstrated in Figure. 5.

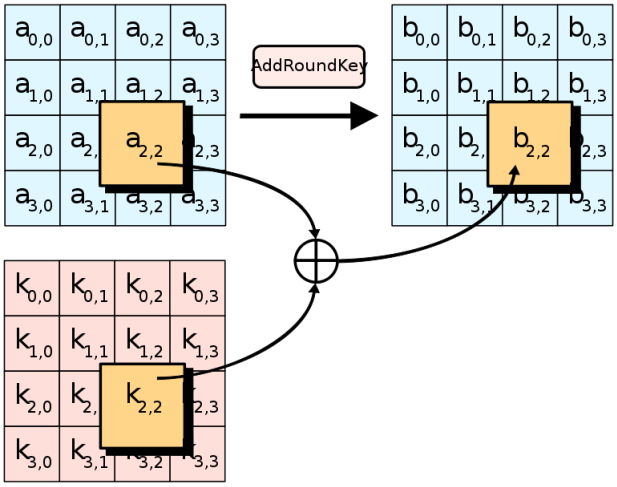

Fig. 5. Add Round Key Step

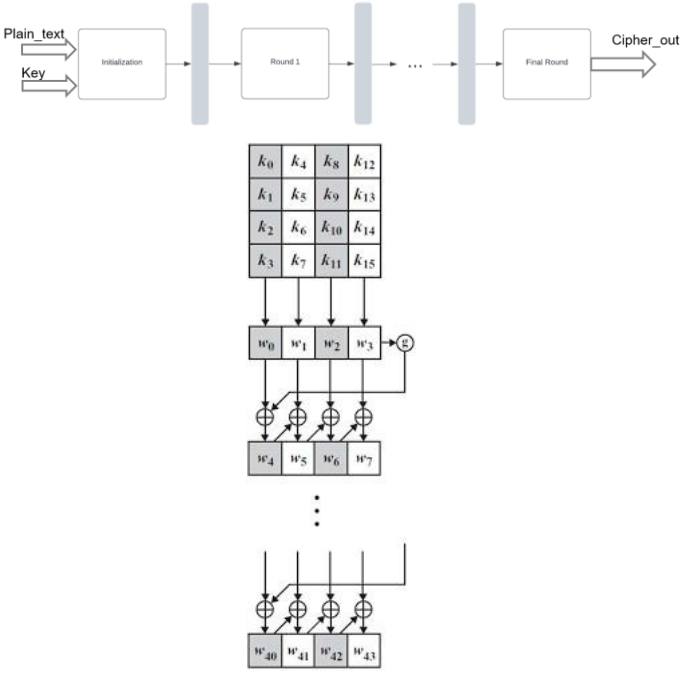

Fig. 6. Key Expansion

After all these rounds 128 bits of encrypted data is given back as output. This process is repeated until all the data to be encrypted undergoes this process. The stages in the rounds can be easily undone as these stages have an opposite to it which when performed reverts the changes.Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

E. Key Expansion

The initial key is expanded and a series of Round Keys are genrated for each round of the encryption. It takes as input a 4-word (16-byte) key and produces a linear array of 44 words (176 bytes). This is sufficient to provide a 4-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher. In the Existing Architecture $4(N_r + 1)$ words are obtained from the intial main key, W_0,W_1,W_2,W_3. This can be visualized from the Figure. 6.

III. PROPOSED ARCHITECTURE

From the previous sections, we observed that each of the rounds in the AES architecture takes place sequentially. This introduces a significant amount of latency and reduces the throughput considerably. There are many ways in which one can improve the performance of the AES algorithm. [8] proposes 4 methods: loop architectures, unrolled architecture, pipelining and sub-pipelining. The first two methods do not utilize much hardware, but the throughput offered by them is quite low. The latter two methods, eat up a lot of hardware resources but also provide good throughput.

A. Outer Round Pipeline

We have implemented an 11-stage outer round-only pipeline architecture. In this proposed method, only the rounds as a whole are pipelined with the help of pipeline registers that are inserted between each round. This allows us to proceed to the computation of each round faster. Each round does not need to wait for the entire encryption to finish before starting the required round of computations of the next plain text. At each clock cycle or after regular intervals, each round passes its encrypted data to the next round with the help of the registers. Another way to improve the performance is to sub-pipeline the various

computations present inside each round. However this will increase the design complexity and consume huge amounts of hardware resources. Figure 7 clearly articulates our proposed architecture.

Fig. 7. Pipeline Architecture

B. Key Expansion Optimization

From 6, we understand that the generation of each key completely depends on the key generated for the previous round. This sequential flow, once again creates a bottleneck when we are trying to optimize the AES encryption process. In our architecture, the sixth subkey is generated from the main key instead of the fifth key. Because of this modification, the sixth key will be generated at the same time the first subkey is generated. Hence, both blocks can generate subkeys in parallel. Figure 8 helps us visualize this architecture. The key expansion process is implemented using parallel architecture. This increases the amount of hardware required, but provides us with a significant boost in the time it takes to produce the Subkeys for each round.

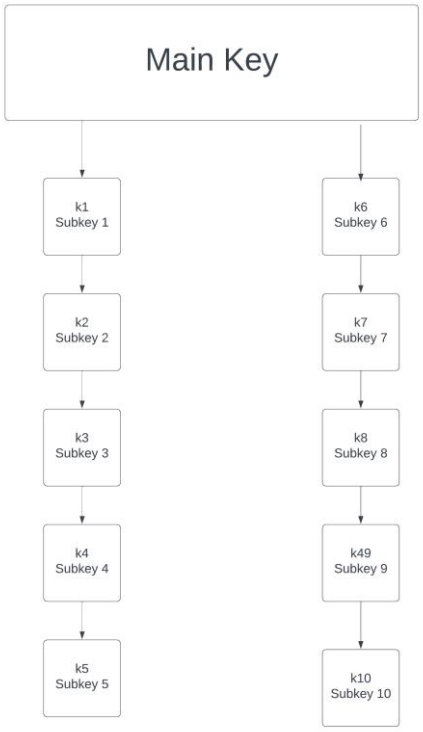

Fig. 8. Optimized Key Expansion

IV. METHODOLOGY

The methodology encompasses implementing AES encryption in Verilog and optimizing it for performance. Synthesis generates a gate-level netlist, further optimized for timing, area, and power. Floorplanning optimizes chip area partitioning, addressing routing congestion and signal integrity. Power planning distributes and optimizes power networks to meet consumption targets. Placement optimizes logic cell and macro locations for timing and congestion, while clock optimization minimizes skew and power in clock networks. Routing establishes physical paths between components, iteratively refining for timing and congestion. Static timing analysis validates timing constraints throughout the flow. Each step is crucial for achieving efficient hardware AES encryption, ensuring performance, area efficiency, and reliability.

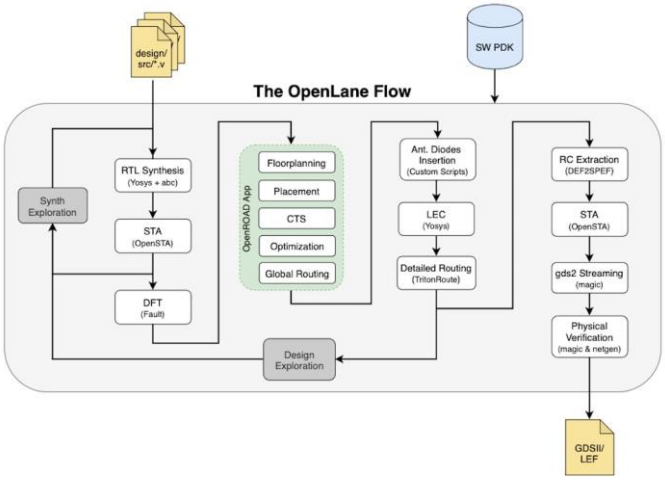

Fig. 9. Methodology

V. RESULTS

A. Verilog Implementation

The Verilog implementation of the AES algorithm was designed with a focus on modularity and efficiency. Each component of the AES encryption process, such as SubBytes, ShiftRows, MixColumns, and AddRoundKey, was implemented as separate Verilog modules. This modular approach facilitated easier debugging and verification, allowing for reusable and scalable design.

The implementation process began with designing and verifying the standard AES-128 algorithm using Verilog. Each component, including SubBytes, ShiftRows, MixColumns, and AddRoundKey, was carefully developed and integrated into the AES core module. The standard AES-128 design was then rigorously tested and verified through simulation, using waveform analysis to ensure correct functionality and timing. Following the successful

verification of the standard AES128, our proposed modifications were introduced to enhance the performance and efficiency of the encryption process. The waveform for this is shown in Figure 10.

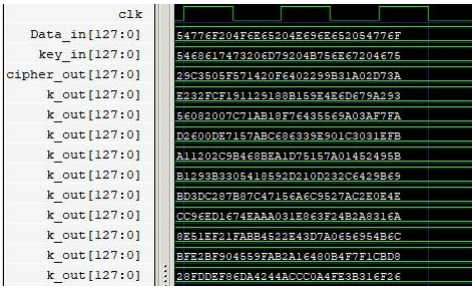

Fig. 10. Standard AES-128

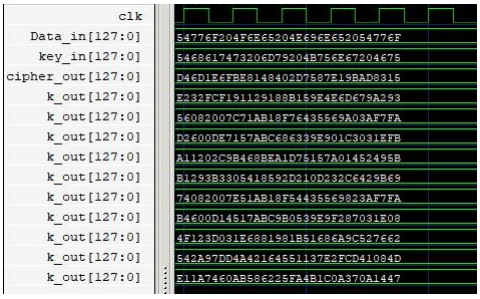

Fig. 11. Modified AES-128

Specifically, we implemented an 11-stage pipeline architecture and modified the key generation process to enable parallel generation of the first and sixth subkeys. These modifications were also implemented in Verilog and subjected to the same rigorous simulation and verification process. The proposed design demonstrated improved throughput and reduced key expansion time, confirming the effectiveness of our enhancements. The waveform analysis for this is depicted in Figure 11.

Figures 10 and 11 illustrate the implementation of the standard AES-128 and our modified AES-128, respectively. A comparative analysis of these figures reveals that the key generation outputs remain identical up to the fifth key. This consistency confirms that the initial stages of our modified implementation align perfectly with the standard AES-128. However, differences emerge in the key generation process beyond the fifth key, where our modifications take effect. These changes validate the correctness of our approach, demonstrating that the modified design adheres to the original algorithm's principles while introducing enhancements to improve performance and efficiency. The successful parallel generation of the first and sixth subkeys in our modified implementation underscores the effectiveness of our proposed optimizations. This modification significantly reduces the time required for the key generation process, potentially reducing it by up to 50%, thereby enhancing the encryption module's overall efficiency. All verilog coding and waveform analysis have been done using the Xilinx Vivado environment.

B. Physical Implementation

The physical implementation of the hardware-accelerated AES module involved translating the verified RTL design into a manufacturable layout. This process encompassed several critical stages, from synthesis to final verification, ensuring that the design meets all performance, area, and power requirements. The PDK used to implement the design is the Skywater 130nm library.

The Verilog RTL code was synthesized using Xilinx Vivado and Yosys, converting the high-level design into a gate-level netlist. This process involved logic optimization and implementation of gate sizing techniques to minimize gate count and improve performance. The final synthesized netlist consisted of 32,424 logic cells. Post-synthesis static timing analysis (STA) was performed to ensure the design met the necessary timing constraints, and any critical paths were identified and optimized.

The gate-level netlist was imported into the OpenLane flow for floorplanning. Floorplanning involves defining the chip's layout, including the placement of major functional blocks and I/O pads. At this stage, we also fix the location of the physicalonly cells like De-Coupling capacitors, Endcap cells, Tap cells, etc. The aspect ratio of the chip was set to 1 and the utilization factor set to 70%.

A robust power distribution network was designed to ensure adequate power delivery to all parts of the chip and to maintain continuous voltage and ground supply to all the cells. Power rings and straps were created to distribute the power evenly and mitigate IR drop issues. Special care was taken to design power grids that can handle the high current demands of the pipelined AES module, ensuring stable and reliable operation. During floor planning, a grid of multiple Vdd and Vss rails is laid down and each cell is given power from a specific rail. The next step involved placing standard cells within the defined floorplan. Using

OpenLane's automated placement tools, the cells were arranged to optimize for performance and area. The placement process ensured that critical paths were minimized, and signal integrity was maintained. Figure 12 shows the placement step's view in the Magic tool. Another round of STA is performed to make sure there are no setup violations in the design.

Clock tree synthesis was performed to create a balanced clock distribution network. The goal was to minimize clock skew and ensure that all parts of the design received the clock signal simultaneously. The clock tree was optimized to reduce clock latency and improve the overall timing performance of the design. During the process of optimizations, a total of 592 buffers were added to the design, to ensure good timing performance by avoiding setup and hold violations. The global routing phase established the primary paths for all interconnections, while detailed routing refined these paths to meet design rules and optimize performance. The routing process ensured that all signal nets were correctly connected, minimizing crosstalk and ensuring signal integrity. After routing, post-route static timing analysis was conducted to verify

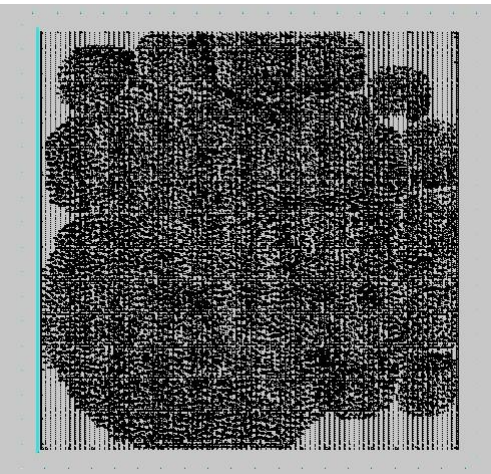

Fig. 12. Placement

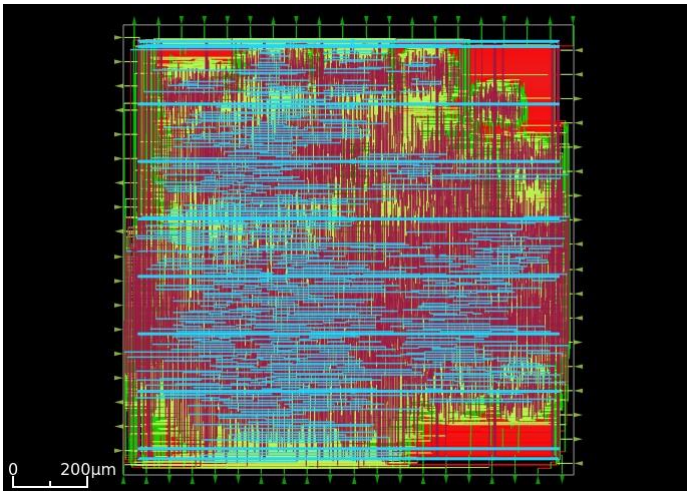

Fig. 13. Final View after routing

that the design met all timing constraints. Any timing violations were identified and corrected. The STA process ensured that the design operated correctly at the target clock frequency without timing errors. Figure 13. is the final view of the chip after all the phases are completed. This is a view from the OpenROAD GUI.

Design rule checks (DRC) were performed to ensure that the layout adheres to the manufacturing process rules of the SkyWater 130nm PDK. Layout versus schematic (LVS) checks verified that the physical layout matched the original schematic design, ensuring design accuracy and completeness. Once the design passed all verification steps, it was prepared for final sign-off. This involved generating the GDSII file, which is the standard format for IC layout data. The final design was ready for fabrication, ensuring that all performance, area, and power specifications were met.

VI. CONCLUSION

The implementation of a hardware-accelerated AES with an 11-stage pipelined architecture and an optimized subkey generation scheme has demonstrated significant performance and efficiency improvements. The Verilog-based design, verified through waveform analysis and synthesized to layout using the OpenLane EDA tool, achieved a peak clock frequency of 100 MHz, an area utilization of 1.44 mm^2, and power consumption of 35 mW, comprising 32,424 standard cells.

The successful use of OpenLane showcases the capability of open-source EDA tools to handle

sophisticated cryptographic hardware designs, thereby reducing the barrier to entry for hardware design and innovation. This not only allows for greater accessibility for students and researchers but also fosters a more inclusive environment for technological advancements in hardware security.

Moving forward, our work opens up new avenues for enhancing the power efficiency and scalability of cryptographic hardware. Future research will aim to refine the power consumption metrics further and explore the implementation of other cryptographic algorithms using similar design methodologies. The demonstrated success of this approach underlines its potential to drive forward the next generation of secure and efficient hardware solutions

REFERENCES

[1] Nation Institute of Standards and Technology (NIST), Data Encryption Standard (DES), National Technical Information Service, Sprinfgield, VA 22161, Oct. 1999

[2] Yulin Zhang and Xinggang Wang, ”Pipelined implementation of AES encryption based on FPGA”, 2010 IEEE International Conference on Information Theory and Information Security, pp. 170-173, 2010.

[3] Vilas V Deotare, Dinesh V Padole, and Ashok S. Wakode, ”Performance Evaluation of AES using Hardware and Software Codesign,” International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC), vol. 2, no. 6, pp. 1638-1643, Jun. 2014.

[4] M. Natheera Banu, ”FPGA Based Hardware Implementation of Encryption Algorithm,” International Journal of Engineering and Advanced Technology (IJEAT), vol. 3, no. 4, pp. 271-277, Apr. 2014.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] M. Santhanalakshmi, M. Lakshana and M. S. GM, ”Enhanced AES-256 cipher round algorithm for IoT applications”, The Scientific Temper, vol. 14, no. 01, pp. 184-190, 2023.

[7] Xinmiao Zhang, Student Member, IEEE, and Keshab K. Parhi, Fellow, IEEE, ”High-Speed VLSI Architectures for the AES Algorithm,” IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, vol. 12, no. 9, pp. 957-967, Sep. 2004.

[8] F.X.Standaert,G.Rouvroy and J.D.Legat, “Efficient Imple- mentation of Rijndael Encryption in Reconfigurable Hardware,”Improvements and Design Tradeoffs, pp. 334-350, 2003.

[9] M. Shalan and T. Edwards, “Building OpenLANE: A 130nm OpenROAD-basedTapeout-Proven Flow: Invited Paper,” 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), San Diego, CA, USA, 2020, pp. 1-6

[10] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

[11] Shih-Hsiang Lin, Jun-Yi Lee, Chia-Chou Chuang, Narn-Yih Lee, PeiYin Chen and Wen-Long Chin, ”Hardware Implementation of HighThroughput S-Box in AES for Information Security,” IEEE Access, vol. 11, no. 6, pp. 9, 2023

[12] M. S. Dontha, ”AES-128 Algorithm Design with Verilog,” *Int. J. Res. Eng., Sci. Manag.*, vol. 5, no. 8, Aug. 2022.

[13] Shet, Ganesh Gopal, Jamuna V, Shravani S, Nayana H G, and Pramod Kumar S. ”Implementation of AES Algorithm Using Verilog.” JNNCE Journal of Engineering & Management 4, no. 1 (January–June 2020): 1-18. [ISSN 2582-0079]

[14] D. Le, J. Chang, X. Gou, A. Zhang and C. Lu, ”Parallel AES algorithm for fast data encryption on GPU” in 2010 2nd international conference on computer engineering and technology, IEEE, vol. 6, pp. V6.1-V6.6, 2010.

[15] 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,” IEEE Access, vol. 10, pp. 2721–2728, 2022, doi:10.1109/ACCESS.2021.3139040.

[16] B. Rashidi, “Compact and efficient structure of 8-bit S-box for lightweight cryptography,” Integr., VLSI J., vol. 76, pp. 172–182, Jan. 2021.

[17] Abdullah, A. M. (2017), “Advanced encryption standard (AES) algorithm to encrypt and decrypt data.”, Cryptography and Network Security, 16(1), 11.

[18] Pravin B. Ghewari, Mrs. Jaymala K. Patil, and Amit B. Chougule, ”Efficient Hardware Design and Implementation of AES Cryptosystem,” International

Journal of Engineering Science and Technology (IJEST), vol. 2, no. 3, pp. 213-219, 2010.

[19] Shanxin Qu,Guochu Shou and Yihong Hu, "High Throughput, Pipelined Implementation of AES on FPGA,"Information Engineering and Electronic Commerce(IEEC) IEEE Press, 2009, pp. 542-545,doi: 10.1109/IEEC.2009.120.

[20] M.R.M.Rizk, S.Member, M.Morsy, "Optimized Area and Optimized "Speed Hardware Implementations of AES on FPGA," Design and Test Workshop, IEEE Press, 2007, pp. 207-217, doi: 10.1109/IDT.2007.4437462.

[21] Archana garg et al, "Implementation of Advanced Encryption Standard Algorithm using VHDL", International Journal of Engineering Trends and Technology, Volume 4 Issue 9, pp. 3956- 3961, September 2013.

[22] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," IEEE Transactions on Very Large Scale Integration Systems, vol. 12, no. 9, pp. 95 967, Sep. 2004.

[23] Reyhani-Masoleh, M. Taha, and D. Ashmawy, "Smashing the implementation records of AES S-box," IACR Trans. Cryptograph. Hardw. Embedded Syst., vol. 2018, no. 2, pp. 298–336, May 2018.