JOURNAL OF SEMICONDUCTOR TECHNOLOGY AND SCIENCE, VOL. 25, NO. 4, AUGUST, 2025
https://doi.org/10.5573/JSTS.2025.25.4.406
ISSN(Print) 1598-1657
ISSN(Online) 2233-4866

A Lightweight AES-256 Accelerator Design through Processing Order Optimization for Low-cost Hardware Security

Yuseong Lee[1*], Jaehak Kang[1*], and Jongmin Lee[1,2]

***Abstract*:** In the era of AI and IoT, securing sensitive data is paramount, especially for resource-constrained edge devices. This paper presents a novel lightweight Advanced Encryption Standard (AES)-256 accelerator design that optimizes encryption and decryption processes for low-cost hardware security. By reordering the processing steps in the AES round function and introducing a pre-processing technique in the key expansion process, the proposed architecture enhances both throughput and resource efficiency. Our work results demonstrate that the proposed architecture achieves a throughput of 42.667 Gbps with an area efficiency of 233.69 Kbps/GE. This makes the proposed design an ideal solution for modern IoT devices, where high-speed secure data processing and hardware efficiency are critical.

***Index terms*:** Advanced encryption standard (AES), hardware security, cost-effectiveness, processing order optimization

I. INTRODUCTION

As an increasing number of devices connected to networks collect personal information to enhance user convenience, vast amounts of sensitive data are being transmitted over these networks. To ensure the rapid and secure protection of such personal information, not only the application of encryption algorithms but also the correlation of encryption and decryption processes is essential. Currently, the Advanced Encryption Standard (AES), a symmetric-key encryption algorithm established by the National Institute of Standards and Technology (NIST) in 2001, is widely used for data encryption and decryption [1]. AES offers high levels of data security and stability, making it applicable across diverse fields, including secure data communications and databases.

Manuscript received Jan. 24, 2025; revised May 25, 2025; accepted Jun. 3, 2025
[1]Department of Electrical and Computer Engineering, Ajou University, Suwon 16499, Korea
[2]Department of Intelligence Semiconductor Engineering, Ajou University, Suwon 16499, Korea
* Equally Contributed Authors
E-mail : jongmin@ajou.ac.kr

However, recent advancements in quantum computing pose significant threats to existing security systems. It has been demonstrated that the Grover's algorithm can reduce the time complexity of brute-force attacks from $O(n)$ to $O(\sqrt{n})$, potentially increasing the vulnerability of symmetric-key encryption algorithms [2]. To maintain sufficient security levels for AES, the key size must be doubled. However, increasing the key size also leads to a corresponding increase in computational overhead, which highlights the importance of the acceleration and lightweight implementation of encryption and decryption operations.

Various attempts have been made to reduce the hardware footprint of AES implementations. Studies such as [3-5], which focus on optimizing the hardware implementation of functional units used in AES operations, have significantly contributed to advancements in this field - and we actively incorporated their ideas into our design. Additionally, there have been ongoing efforts to design accelerators with specialized purposes based on novel architecture. For instance, [6] proposed a round-based AES accelerator optimized for low area and high throughput by applying techniques such as tower-field S-box implemen-

tation and on-the-fly key expansion. In [7], an Authenticated Encryptions with associated data (AEAD) core that integrates multiple AES-based encryption algorithms was introduced, while [8] applied dual-rail flush logic (DRFL) to the AES core to defend against differential power analysis (DPA) attacks. Finally, [9] presented an 8-bit round-based AES-128 encryption accelerator optimized for low area and low power consumption.

This paper aims to optimize two critical operations in the AES encryption/decryption process: the round function responsible for encryption and the Key Expansion process for key generation. In the proposed AES-256 architecture, we introduce an optimized execution order for the round algorithm and a pre-processing algorithm to speed up the round key expansion. These optimization strategies improve the area-throughput trade-off of AES-256 by 92.89% compared to [1] and achieve an 8.76% enhancement over [3-5].

The remainder of this paper is organized as follows: Section II discusses optimization methods for the AES round function and improvements in overall round function operations. Section III addresses the acceleration of AES-256 Key Expansion through pre-processing techniques. Section IV summarizes the implementation results of the proposed approach and compares our fully-pipelined AES-256 accelerator with the architectures presented in [6-9]. Finally, Section V summarizes this research.

II. Processing Order Optimization of AES-256 Round Functions

1. Previous Round Function Optimization Methods

A previous research attempted to optimize such stage of AES operation for efficient hardware implementation [3]. Traditionally, AES has implemented separate modules for the S-box and inverse S-box used in SubBytes and InvSubBytes operations. However, [3] leveraged the fact that, under normal conditions, encryption and decryption are not performed simultaneously, and proposed sharing the inverse multiplicative table between the two operations.

Specifically, as shown in Fig. 1, SubBytes or InvSubBytes operations can be selectively performed based on the EC signal. In this structure, when the EC signal is 1, the SubBytes operation for encryption is performed, and when it is 0, the InvSubBytes operation for decryption is selected. By sharing the inverse multiplicative table in this way, the design achieved a 43% area reduction compared to implementing separate modules for SubBytes and InvSubBytes.

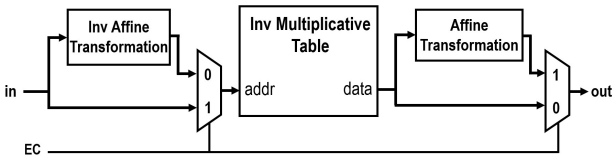


Fig. 1. . Architecture of the inverse-optional S-box module.

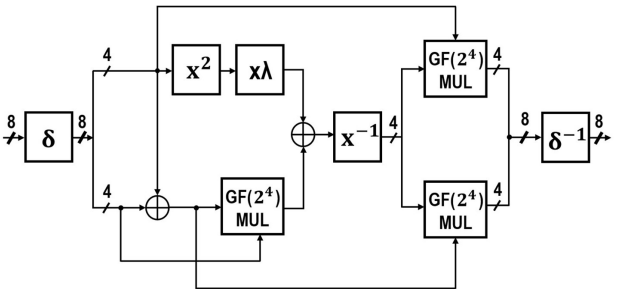


Fig. 2. Multiplicative inversion module for the S-box.

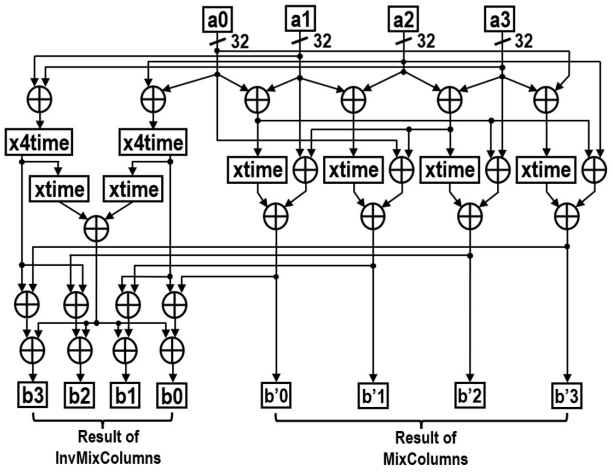
A pipelined structure using only combinational logic gates was proposed to overcome limitation of low area efficiency caused by constructing the inverse multiplicative table for SubBytes and InvSubBytes operations using LUTs [4].

In this structure, as shown in Fig. 2, the inverse multiplicative table is generated over a Galois Field. The structure in Fig. 2 is based on composite Galois field arithmetic, where the complex inverse multiplication in $GF(2^8)$ is decomposed into operations over lower-degree fields such as $GF(2^4)$ and $GF(2^2)$, with pipelining applied.

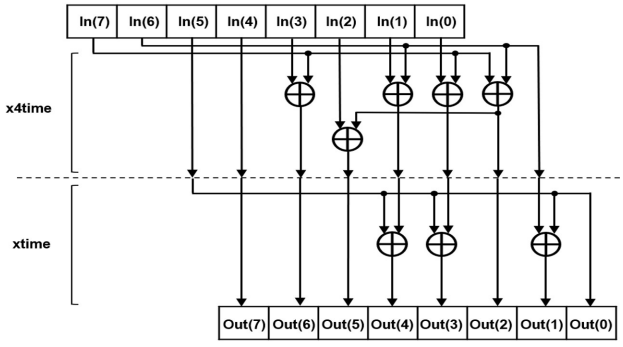
The module consists of blocks for squaring, multiplication, inversion, and isomorphic transformations, all of which can be implemented using simple logic gates. This approach reduces circuit area and power consumption while improving computational speed.

In [5], the MixColumns and InvMixColumns operations of the AES algorithm were integrated, as illustrated in Fig. 3(a), to maximize resource sharing. [5] merged the MixColumns and InvMixColumns operations into a single module to perform the operations quickly while minimizing the area used.

To perform these operations, as shown on the left side of Fig. 3(a) for the InvMixColumns operation, *xtime* (multiplication by 0x02 followed by a modulo operation with 0x1B) and *x4time* (two consecutive *xtime* operations) are executed sequentially, requiring delays of 1 t_{XOR} and



(a)



(b)

Fig. 3. (a) Architecture of MixColumns/InvMixColumns module. (b) cascaded "*xtime*" and "*x4time*" function for resource sharing.

2 t_{XOR}, respectively. However, when arranged consecutively, as illustrated in Fig. 3(b), the delay is reduced to just 2 XOR gates. This plays a key role in reducing the critical path delay of the module in Fig. 3(a) to 5 t_{XOR}.

2. Proposed Round Function Optimization Methods

Optimization techniques targeting individual steps of the AES round function were proposed in [3-5], achieving partial optimization of the function and accelerating AES operations. However, such approaches were limited from the perspective of optimizing the entire round function, presenting challenges in maximizing the encryption and decryption speeds of AES.

The encryption and decryption round functions of AES can be implemented using a single architecture, as shown in Fig. 4. When the EC signal is set to 1, the architecture encrypts input data in order of SubBytes, ShiftRows,

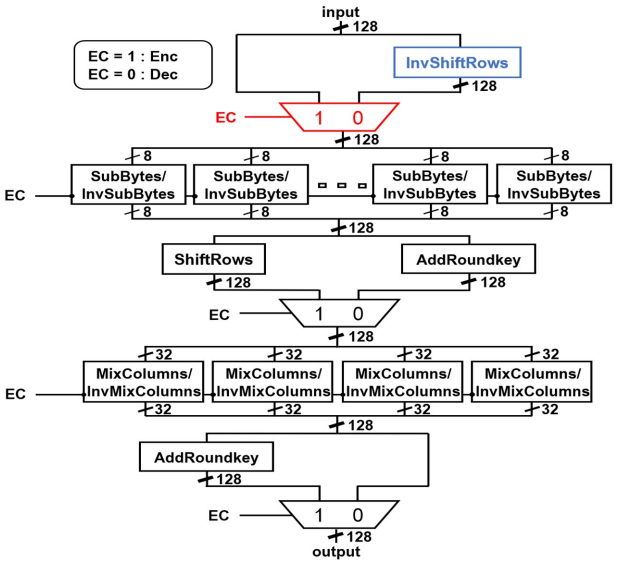


Fig. 4. The architecture of the AES round function after incorporating the concepts proposed in [3-5].

MixColumns, and AddRoundKey. Conversely, when the EC signal is set to 0, it performs decryption in order of InvShiftRows, InvSubBytes, AddRoundKey, and InvMixColumns. In the structure depicted in Fig. 4, it is essential to selectively implement encryption and decryption paths using multiplexers (MUXs) for resource sharing. However, this approach has drawbacks, including area overhead and processing delays caused by the unnecessary overhead of passing through the multiplexers. This inclusion of additional MUXs can lead to increased area and delays.

To mitigate these delays, processing order optimization was introduced to reduce the latency of the round function. The InvShiftRows operation, as indicated by the blue-colored box in Fig. 4, was moved to the position after InvSubBytes, indicated as a green-colored box in Fig. 5. This reordering is feasible because InvShiftRows shifts the data on a byte basis, while InvSubBytes substitutes data on the same byte basis, ensuring that the reordering does not affect the final result. This approach improves both performance and area efficiency by eliminating unnecessary MUXs, which is red-marked in Fig. 4.

The AES Round function includes the AddRoundKey operation, which performs a bitwise XOR with the round key after the MixColumns operation. The MixColumns and InvMixColumns operation utilizes the unified structure proposed in [5]. However, using the structure [5], the MixColumns operation requires a delay of $3t_{XOR}$, while the InvMixColumns operation demands a

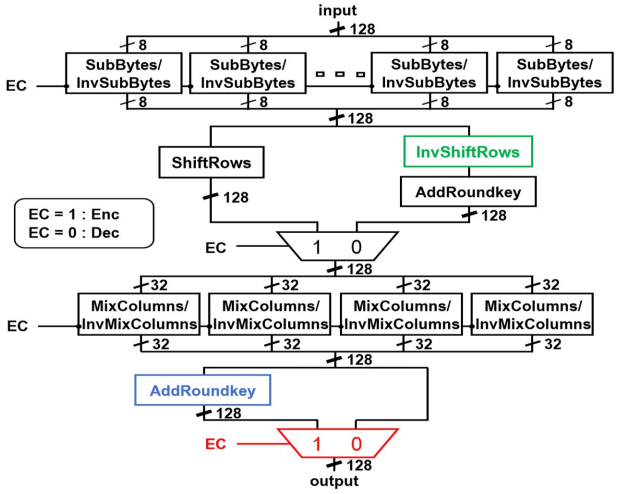


Fig. 5. The architecture after reordering the SubBytes and ShiftRows operations.

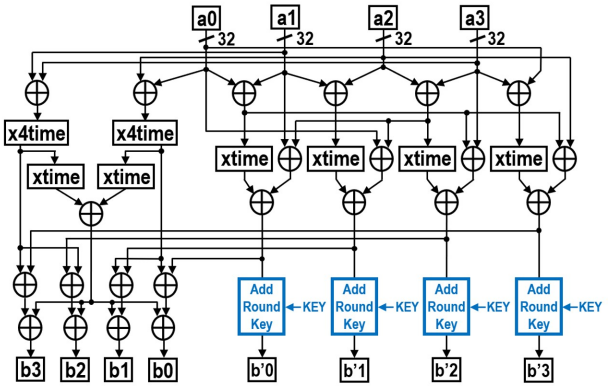


Fig. 6. Architecture of the MixColumns/InvMixColumns after AddRoundKey is added after MixColumns process.

total delay of $5t_{XOR}$. Therefore, MixColumns operation has $2t_{XOR}$ shorter delay compared to InvMixColunms. Thanks to the shorter delay, the proposed structure is able to integrate the AddRoundkey operation, right after the MixColumns operation during encryption, represented by blue-highlighted boxes in Fig. 6. This integration eliminates the need for a 2-to-1 MUX, as depicted by the red-marked area in Fig. 5, and reduces the total delay by $t_{XOR}+t_{MUX}$. Through this optimization, the delay from MixColumns, InvMixColumns, and AddRoundKey functions was improved, as indicated by the green-colored box in Fig. 7.

In the SubBytes and InvSubBytes operations, as shown in Fig. 1, the shared multiplicative table is flanked by the IAT and AT, which are selectively utilized in encryption and decryption operations. In particular, during decryption, the operation proceeds through the multiplicative table following the IAT, making AT operation unnecessary. Consequently, the critical path of the AT limits performance improvement in the decryption process.

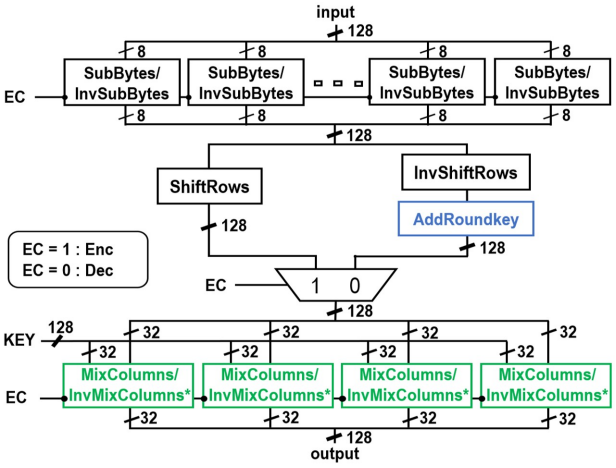


Fig. 7. Architecture of after changing the order of AddRoundKey for encryption path.

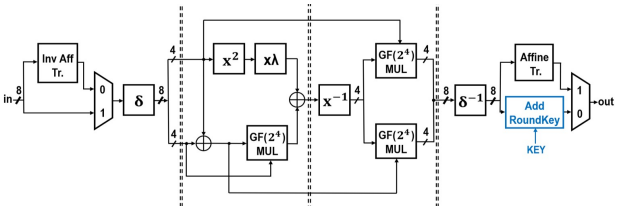


Fig. 8. Architecture of SubBytes/InvSubBytes after ARK is added after InvSubBytes process.

To address this challenge, the proposed structure takes advantage the fact that the AddRoundKey and InvShiftRows operations are performed on a byte-by-byte basis. This allows the AddRoundKey operation to be moved before the InvShiftRows operation without affecting the computation results. By leveraging this characteristic, the AddRoundKey operation was repositioned after the InvSubBytes operation in the decryption process, as illustrated by the blue highlights in Fig. 8, effectively eliminating the delay caused by AddRoundKey.

As a result, the integration of the SubBytes and AddRoundKey operations minimized the overall path delay of the round function as illustrated by the green highlights in Fig. 9.

III. Processing Order Optimization of AES-256 Key Expansion

The conventional AES key expansion process sequentially expands a 256-bit key, as shown on the left side of

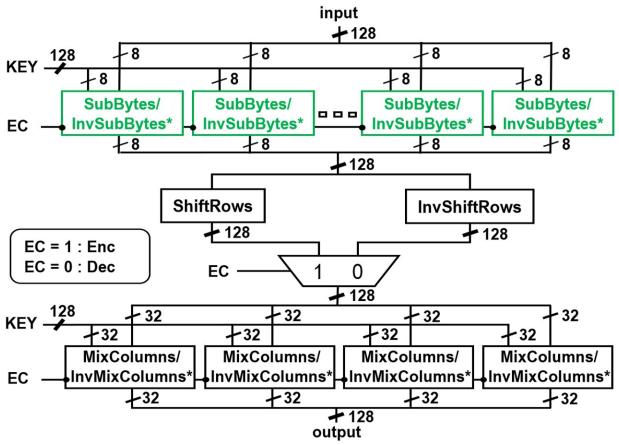


Fig. 9. Final Architecture of the proposed round function optimization module.

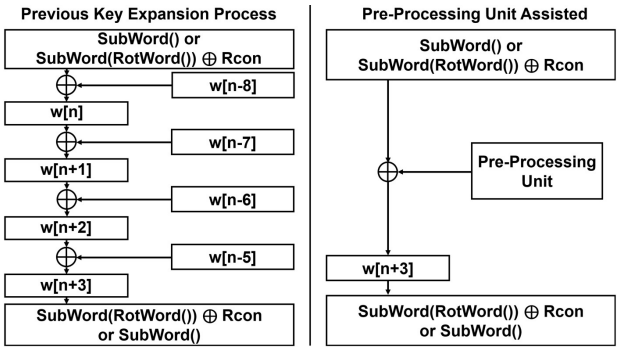


Fig. 10. Basic concept of pre-processing key expansion.

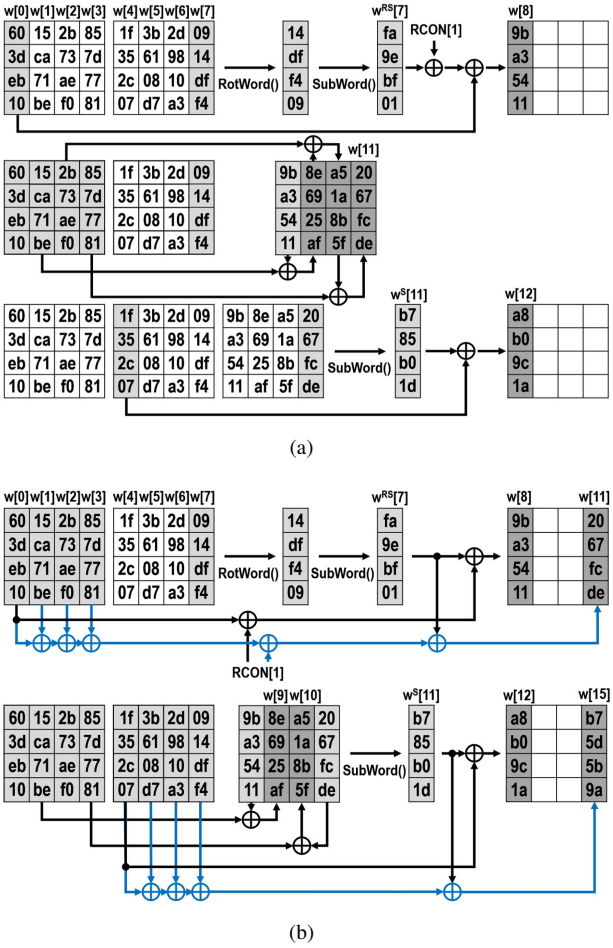


Fig. 11. (a) Conventional key expansion mechanism. (b) Pre-processing unit assisted key expansion.

Fig. 10. However, in this approach, it is necessary to wait until the final word of the current stage key is calculated to generate the key for the next stage as shown in Fig. 11(a). This dependency causes delays, slowing down the overall key expansion process.

To overcome the limitations of this sequential key expansion process, a high-speed key expansion structure utilizing a pre-processing unit is proposed as shown on the right side of Fig. 10. In this structure, as depicted in Fig. 11(b), the final word of the n-th stage key, $w[4n+3]$, is prioritized for computation by leveraging the commutative property of XOR operations. Specifically, the three XOR operations typically performed after the SubWord operation, optimizing the computation sequence.

Through this approach, the word required for the next SubWord operation can be quickly calculated within t_{XOR} after the completion of the previous SubWord operation. As a result, compared to the traditional key expansion process on the left side of Fig. 10, the computation time was reduced by $3t_{XOR}$.

To enhance the speed of the pre-processing operation, the results expressing the final word of each stage key in terms of the previous stage's values are summarized in Table 1. The analysis confirms that the final word of each stage key can be derived based on the final word of the previous stage and the input key. Notably, the computation of the w^{RS} (or w^{S}) key expansion round function for each round can be performed in parallel with the XOR operations for the remaining data. This parallelism reduces the calculation time of $w[4n+3]$ to the sum of the w^{RS} (or w^{S}) function delay and t_{XOR}.

The proposed key expansion structure incorporating the pre-processing technique is shown in Fig. 12. The pre-processing unit in the proposed architecture receives the AES key and the pre-computed $w[11]$, $w[15]$, $w[19]$, $w[23]$, $w[27]$ values from the key schedule handler. Based on this input, it performs pre-processing operations, excluding the key expansion round functions specified in Table 1.

Table 1. Elements that consist SubWord() logic input.

Result	Comb. logic	Result	Comb. logic
w[11]	w^{RS}[7]⊕ w[0]⊕w[1]⊕w[2]⊕w[3]	w[39]	w^{S}[35]⊕w[7]
w[15]	w^{S}[11]⊕ w[4]⊕w[5]⊕w[6]⊕w[7]	w[43]	w^{RS}[39]⊕w[11]
w[19]	w^{RS}[15]⊕w[1]⊕w[3]	w[47]	w^{S}[43]⊕w[15]
w[23]	w^{S}[19]⊕w[5]⊕w[7]	w[51]	w^{RS}[47]⊕w[19]
w[27]	w^{RS}[23]⊕w[2]⊕w[3]	w[55]	w^{S}[51]⊕w[23]
w[31]	w^{S}[27]⊕w[6]⊕w[7]	w[59]	w^{RS}[55]⊕w[27]
w[35]	w^{RS}[31]⊕w[3]		

w^{RS}[k]=SubWord(RotWord(w[k]))⊕Rcon

w^{S}[k]=SubWord(w[k])

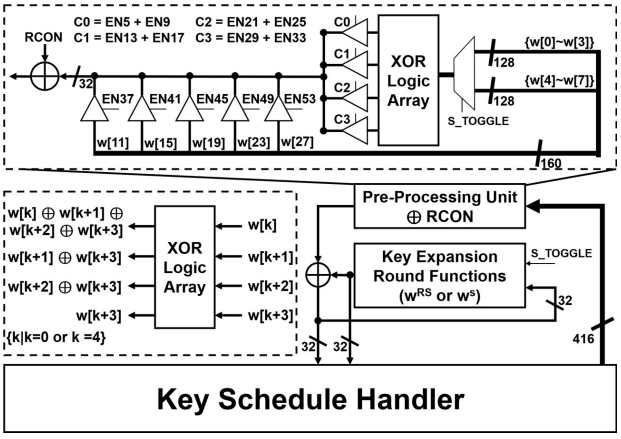


Fig. 12. Architecture of key expansion logic including pre-processing unit.

While the pre-processing operation is executed, the key expansion round functions are processed concurrently. Subsequently, the pre-processing results and the output of the key expansion round function are XORed to compute $w[4n+3]$. This structure effectively minimizes the long computation time required by the conventional design shown in Fig. 11(a), as demonstrated in the optimized structured depicted in Fig. 11(b).

IV. Implementation Results

The proposed AES-256 structure was implemented using a 180 nm CMOS process. To minimize the impact of implemented process and ensure fairness, implementations using the methods from [1,3-5] were performed using the same process. The implementation results are summarized in Table 2, which compares area, encryption/decryption speed, and area efficiency across different maximum operating frequencies.

Table 2. Comparison of the optimization results of the round function implemented in a 180nm CMOS process.

	Proposed	[3-5]	[5]	[3,4]	[1]
Operation frequency (MHz)	416.67	400	500	384.62	555.56
Area (mm^2)	0.191	0.197	0.514	0.204	0.625
NAND2-equivalent gate count (GE[1)])	13064.0	13473.5	35126.0	13943.3	42714.3
# of Cycles (cycle)	5	5	5	5	5
Enc/Dec speed (MEnc/s or MDec/s)	416.67	400	500	384.62	555.56
Enc/Dec efficiency[2)] (GEnc/s/mm^2 or GDec/s/mm^2)	2.1783	2.0283	0.9724	1.8832	0.8886

[1)] GE: Gate equivalent

[2)] Enc/Dec efficiency are calculated @steady state

Table 3. Comparison of the optimization results of the key expansion function implemented in a 180nm CMOS process.

	Proposed	[1]
Operation frequency (MHz)	454.54	400
Area (mm^2)	0.414	0.440
NAND2-equivalent gate count (GE)	28299.5	28716.5
# of Cycles (cycle)	55	55
KE[1)] completion Time (ns)	121	137.5
KE efficiency (MKE/s/mm^2)	19.94442	16.53165

[1)] KE: Key expansion

As a result, applying the proposed structure has 25% slower encryption/decryption speed compared to [1], but the area was reduced by 69.4%. Additionally, when compared to [3-5], the speed improved by 4% and the area was reduced by 3%, respectively.

Notably, in terms of encryption/decryption area efficiency, the proposed structure achieved an improvement of 145.14% over [1] and 7.4% over [3-5]. These results demonstrate that the proposed structure is an efficient design capable of optimizing encryption/decryption performance within a limited area.

The design results of the key expansion pre-procession unit are summarized in Table 3. By utilizing the pre-processing unit, the operating frequency was increased by 13.6%, while the resource-sharing technique minimized trade-offs in terms of area efficiency.

As a result, the total time required for key expansion was reduced by 12% compared to [1], and the computation throughput per unit area was improved by 20.6%. These

Table 4. Comparison of the AES-256 implementation in a 180nm CMOS process.

	Proposed	[1,3-5]	[1]
Operation frequency (MHz)	333.33	312.5	333.33
Area (mm^2)	2.672	2.724	5.154
NAND2-equivalent gate count (GE)	182577.8	186162.7	352154.5
# of Cycles (cycle)	69	69	69
Power consumption (mW)	0.842	0.842	0.850
Enc/Dec efficiency (MEnc/s/mm^2 or MDec/s/mm^2)	124.748979	114.700775	64.67482

results demonstrate that the proposed pre-processing unit design enhances both the performance and area efficiency of key expansion.

The implementation results of the entire AES-256 architecture are summarized in Table 4. By applying the proposed structure, the same operational speed as [1] was maintained while reducing the area by 51.15%. Furthermore, compared to [1,3-5], a 6.67% improvement in speed and a 2.23% reduction in area were achieved. As a result, in terms of encryption/decryption efficiency, the proposed design demonstrated a 92.89% improvement over [1] and an 8.76% improvement over [1,3-5]. These results exhibits that the proposed design enhances both area efficiency and performance in AES-256 implementations.

Table 5 presents a comprehensive comparison of various AES accelerator designs, including the proposed design and several state-of-the-art implementations. The table includes detailed information on each design's technology node, supply voltage, operating frequency, area, power consumption, throughput, and efficiency metrics. In this table, our study demonstrated superior area efficiency and energy efficiency while maintaining high throughput, which is the strength of the fully-pipelined AES accelerator.

V. Conclusions

A novel approach to optimize AES-256 was proposed in this paper, focusing on the processing order optimization of both round and key expansion functions. To overcome the inefficiencies of conventional designs, innovative architectures and pre-processing techniques were introduced, achieving notable improvements in performance and area efficiency.

By optimizing the round function, unnecessary delays caused by multiplexer overheads were eliminated, reducing critical path delays and enhancing resource sharing. For the key expansion process, a pre-processing unit was designed to reorder computational sequences, enabling faster key generation.

The implementation of AES-256 incorporating these advancements maintained the operational speed of prior

Table 5. AES accelerator implementation summary and comparison with previous works.

	Proposed	VLSI '16 [9]		A-SSCC '17 [8]		TCAS-II '20 [7]	IEEE TOC '20 [6]		[1,3-5]	[1]
Type	Fully-pipelined AES	8bit round-based AES-128 encryptor		DRFL applied AES (for DPA attack resistance)		AEAD core	Round-based AES		Fully-pipelined AES	Fully-pipelined AES
							Area opt.	Area-speed opt.		
Technology	180 nm	40 nm		65 nm		45nm	45 nm		180 nm	180 nm
Voltage (V)	1.8	0.9	0.47	1	0.4	-	-		1.8	1.8
Frequency (MHz)	333.33	1300	122	430	10	568.2	571.43	694.44	312.5	333.33
Area (mm^2)	2.672	0.00429		0.048		-	-	-	2.724	5.154
NAND2-equivalent gate count[1)] (GE)	182578	-	-	-	-	392541	16418	17369	186163	352155
Power (mW)	0.842	4.39	0.1	19.5	0.08	-	-	-	0.842	0.850
Throughput (Gb/s)	42.667	0.494	0.046	2.752	0.064	72.7	7.31	8.89	40	42.667
Area efficiency (Kbps/GE)	233.69	-	-	-	-	185.204	445.5	511.78	214.866	121.159
Energy efficiency (pJ/b)	1.99E-2	8.85	2.24	7.09	1.25	-	-	-	2.11E-2	1.97E-2

[1)] Synthesis results

designs while achieving a 51.15% reduction in area compared to [1]. Furthermore, the proposed design demonstrated a 6.67% improvement in speed and a 2.23% reduction in area compared to [1,3-5]. These improvements led to encryption and decryption efficiency gains of 92.89% over [1] and 8.76% over [1,3-5].

Altogether, the findings highlight that the proposed design provides a practical and effective solution for implementing AES-256, significantly enhancing performance and resource utilization. These contributions are particularly impactful for resource-constrained applications, such as edge devices, where both performance and efficiency are critical.

ACKNOWLEDGMENTS

The EDA Tool was supported by the IC Design Education Center (IDEC), Korea.

REFERENCES

[1] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback, and J. Dray, *Advanced Encryption Standard (AES)*, Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, 2001.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," *Proc. of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pp. 212-219, 1996.

[3] C.-C. Lu and S.-Y. Tseng, "Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter," *Proc. of IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, San Jose, CA, USA, pp. 277-285, 2002.

[4] B. Rashidi and B. Rashidi, "Implementation of an optimized and pipelined combinational logic Rijndael S-box on FPGA," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 5, no. 1, pp. 41-48, 2013.

[5] C. -Y. Li, C. -F. Chien, J. -H. Hong and T. -Y. Chang, "An Efficient Area-Delay Product Design for MixColumns/InvMixColumns in AES," *Proc. of IEEE Computer Society Annual Symposium on VLSI*, Montpellier, France, pp. 503-506, 2008.

[6] R. Ueno et al., "High throughput/gate AES hardware architectures based on datapath compression," *IEEE Transactions on Computers*, vol. 69, no. 4, pp. 534-548, April 2020.

[7] S. Sawataishi, R. Ueno, and N. Homma, "Unified hardware for high-throughput AES-based authenticated encryptions," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 9, pp. 1604-1608, September 2020.

[8] S. Lu, et al., "A 1.25pJ/bit 0.048mm^2 AES core with DPA resistance for IoT devices," *Proc. of IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pp. 65-68, 2017.

[9] Y. Zhang, K. Yang, M. Saligane, D. Blaauw, and D. Sylvester, "A compact 446 Gbps/W AES accelerator for mobile SoC and IoT in 40 nm," *Proc. of IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, Honolulu, HI, pp. 1-2, 2016.

Yuseong Lee received his B.S. degree from the Department of Electrical and Computer Engineering from Ajou University, Korea, in 2025. His research interests include digital integrated circuits and hardware security circuits.

Jaehak Kang received his B.S. degree from the Department of Electrical and Computer Engineering from Ajou University, Korea, in 2024. His research interests include digital integrated circuits and hardware security circuits.

Jongmin Lee received his B.S. degree in semiconductor systems engineering and a Ph.D. degree in electrical and computer engineering from Sungkyunkwan University, Suwon, Korea, in 2017 and 2022, respectively. From 2022 to 2023, Dr. Lee was affiliated with Samsung Electronics as an Engineer. In 2023, he joined Ajou University, Suwon, Korea, as an Assistant Professor for the Department of Intelligence Semiconductor Engineering. His research interests include hardware security, post-quantum cryptography accelerators, and low power digital circuits and systems.