Area Optimization of AES Key Expansion on FPGA-Based Cryptographic Systems

Swagata Pradhan
Dept. of Electronics and Communication Engineering Institute of Engineering & Management, Kolkata School of University of Engineering & Management, Kolkata
Kolkata, India
swagata.chakravyuha@gmail.com

Swastika Pradhan
Dept. of Electronics and Communication Engineering Institute of Engineering & Management, Kolkata School of University of Engineering & Management, Kolkata
Kolkata, India
swastika.chakravyuha@gmail.com

Mihir Lal Saha
Dept. of Electronics and Communication Engineering Institute of Engineering & Management, Kolkata School of University of Engineering & Management, Kolkata
Kolkata, India
mlsahaece@gmail.com

2026 9th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech) | 979-8-3315-6339-4/26/$31.00 ©2026 IEEE | DOI: 10.1109/IEMENTECH202669403.2026.11434413

***Abstract*— The Advanced Encryption Standard (AES) is popularly used in modern cryptography systems and the hardware realization of this algorithm plays a central role in defining the overall system performance. The key-expansion unit, which is one of the constituent units of AES, is a major area-consumer of silicon as it relies on the substitution and permutation processes. This paper focuses on reducing area consumed by AES key-expanding module in the cryptographic application of field-programmable gate array (FPGA)-based systems. A modified key-expansion architecture is proposed and its performance is compared with the existing AES implementation by using the synthesis-based analysis. The synthesis of both designs is done under the same FPGA hardware. The optimized architecture resulted in a reduction of 42.3% in the number of cells, about 66.5% in the number of LUTs, and an 87.5% decrease in the number of instances S-boxes. Moreover, the key-expansion block analysis in itself demonstrates that the use of slice-LUT is significantly reduced. This evidence shows that the proposed method is effective to reduce hardware area and still be compatible with the standard AES key schedule, which makes it a resource-saving implementation of cryptographic applications on a limited hardware platform of the FPGA based systems.**

Keywords— AES, Cryptography, Hardware Security, Block Cipher, FPGA

I. Introduction

In the current era where large amounts of data are continuously generated, stored and processed [1], the use of cryptographic algorithms[2] has become crucial to safeguard these data and maintain security in systems and environments dealing with data transmission and storage. AES [3], also known as the Advanced Encryption Standard, is a classical symmetric encryption algorithm published by NIST [4] in 2001. In 2000, NIST selected three members of the Rijndael family of ciphers[5] to replace DES, a less secure block cipher, to form this encryption algorithm known as AES.

AES has a block size of 128 bits and is widely known for its effectiveness and high security. It is recognized as one of the most secure algorithms. With its varying key size of 128, 192 or 256 bits, AES is a highly flexible block cipher and can be used widely because it can be adapted to meet unique security requirements[6]. AES can be of three types- AES-128 having 10 rounds, AES-192 having 12 rounds and AES-256 with 14 rounds depending on the key size[7]. The two foundational principles of grouped cipher- diffusion and confusion are adopted in AES which mostly include the different transformational operations. The different transformations of this algorithm are SubBytes, ShiftRows, MixColumns and AddRoundKey[8] where substitution of bytes or SubBytes is the only non-linear operation. Thus, the AES algorithm's performance is primarily enhanced by the byte substitution transformation [9]. The subkeys for each round are obtained by the Key Expansion algorithm which is a procedure where the input seed key is expanded through multiple rounds. Although the traditional key expansion algorithm is fast, the co-relation between each round-key makes it easy for attackers to derive other rounds keys or the master key. The key generation algorithm of AES is prone to several key-breaking attacks[10]. Thus, the security of AES algorithm is affected by this. Following the discovery of this issue, ample research has been performed to bring about cryptographic improvements. Being the most used and secure_encryption algorithm, AES has faced multiple attacks like differential, side-channel and power analysis attacks. Strength analysis conducted by fellow cryptanalysts on AES have revealed that a reduced-round AES (where only 8 rounds are used instead of 10) are less secure and can be broken by brute force attack due to the increasing computing power. This calls for further exploration to enhance the security of this algorithm.

This paper proposes the idea of a modified key expansion algorithm of AES by changing the G-function of the key expansion algorithm.

II. Standard AES Algorithm

Advanced Encryption Standard (AES) comprises of a 128 bits plaintext which is represented as a 4 x 4 state matrix. This acts as the initial state and all the round transformations are performed on this state. Fig.1 shows the standard architecture of AES.

A. Initial transformation:

The input 4x4 state matrix is made to undergo xor operation with the master key (Round Key 0) also taken in the form of 4x4 matrix. The output matrix is then used as input for all the subsequent round operations.

B. Round Transformations:

The number of round transformation in AES depends on the key size. It is 10, 12 and 14 respectively for key sizes 128 bits, 192 bits and 256 bits respectively [11]. Each round transformation (except the last round) includes four different operations namely SubBytes, ShiftRows, MixColumns and

979-8-3315-6339-4/26/$31.00 ©2026 IEEE

AddRoundKey respectively. In the last round the MixColumns operation is omitted.

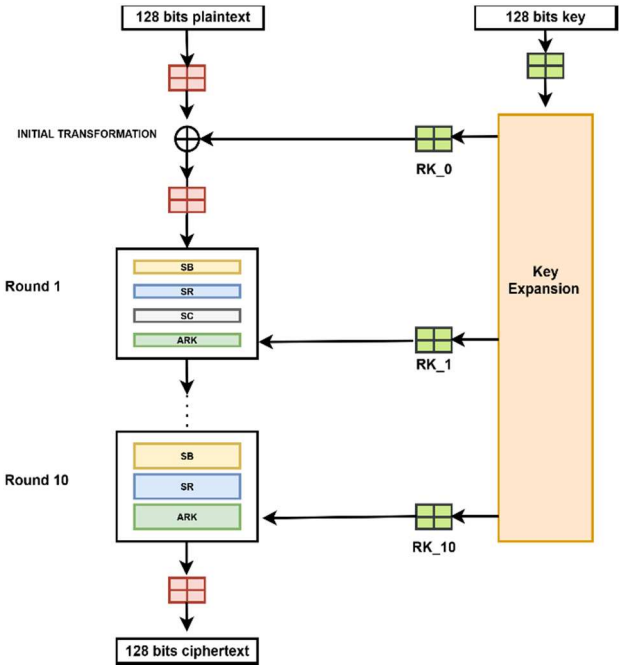

Fig.1 Block Diagram of the standard architecture of AES

SubBytes: This is a nonlinear byte substitution operation in which every byte is mapped on a fixed 8-bit lookup table (s-box). Each byte from the input matrix is taken with the lower nibble being mapped onto the column of the s-box and the upper nibble onto the row. The value at the intersection of this row and column provides the output byte. SubBytes operation provides nonlinearity.

ShiftRows: This is a permutation operation where the rows of the input state matrix are shifted by various offsets. The first row undergoes no shifting, the second row undergoes cyclic left shift by 1 byte, the third row by 2 bytes and ultimately the fourth row is shifted by 3 bytes. This operation guarantees that a byte from a column is spread across several columns in the subsequent rounds.

MixColumns: This is a linear transformation operation performed over Galois Field ($GF(2^8)$) where the 4 bytes of the column in the matrix is made to perform multiplication and xor additions with the constants (1, 2, 3) . This operation enhances diffusion.

AddRoundKey: This is the last operation performed in a round where the state matrix outputted after the MixCoulmns layer is made to perform xor addition with a fixed Round key which is unique for each round and is obtained with the help of the AES Key Expansion. This round introduces key material into the state making the strength of AES dependent on the unpredictability of each round key.

C. AES Key Expansion:

AES Key Expansion performs the operation of generation round keys for each rounds [12]. The purpose of this key Scheduling Algorithm is to ensure that each round key is cryptographically independent and unpredictable and to guarantee that the master key propagates small changes across all the rounds. For AES-128 the master key of 128 bits is divided into 4 words (w[i]) each of 32 bits for a particular round. A total of 44 words is used across all the 10 rounds as depicted in Fig.2.

The first 4 words of Round Key 0 in case of AES-128 corresponds to the original master key. The subsequent words which are used for round keys are generated as: w[i]=w[i-4]⊕G(w[i-1]) where G(w[1-i]) represents either the output of the g function or the direct output word depending on the word index. The next four words generated are used as Round Key 1 for first round and the process goes on till generation of all the round keys.

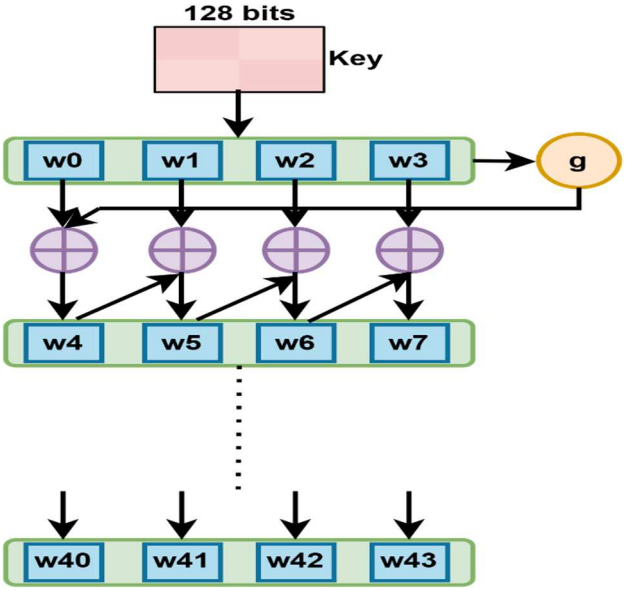

Fig.2 Block Diagram of the Key Scheduling Algorithm of AES

g-function: Fig.3 shows the g-function which is the only nonlinear component of the key scheduling algorithm which introduces nonlinearity and round dependence into the key algorithm thereby making it secure against cryptanalytic exploitation. The g-function is applied to the key periodically , i.e. in every 4th word (w[3], w[7], w[11], w[15],…w[39])

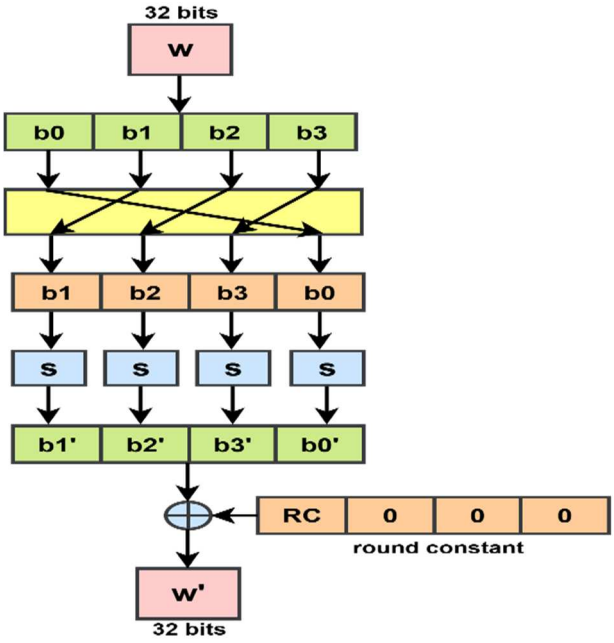

Fig.3 Block Diagram of the g-function

and the output of the g-function is then made to perform xor operation with the first word of a round key to form the first word of next round key. The g-function undergoes four operations: RotWord, SubWord and Round constant addition operation.

In RotWord operation the word is divided into 4 bytes and made to undergo circular left shift by 1 byte. In SubWord each byte undergoes the 8 bit substitution whose output is xor-ed with the round constant. For each round there exists a unique round constant which introduces essential nonlinearity. It is expressed as: G(w)=SubWord(RotWord(w)) $\oplus$ Rcon[r] where r is the round index.

For AES-128, 192 and 256 total 11, 13 and 15 round keys are used respectively.

Fig.4 shows the modified key expansion that has been proposed with the modification being mainly focused on the g-function.

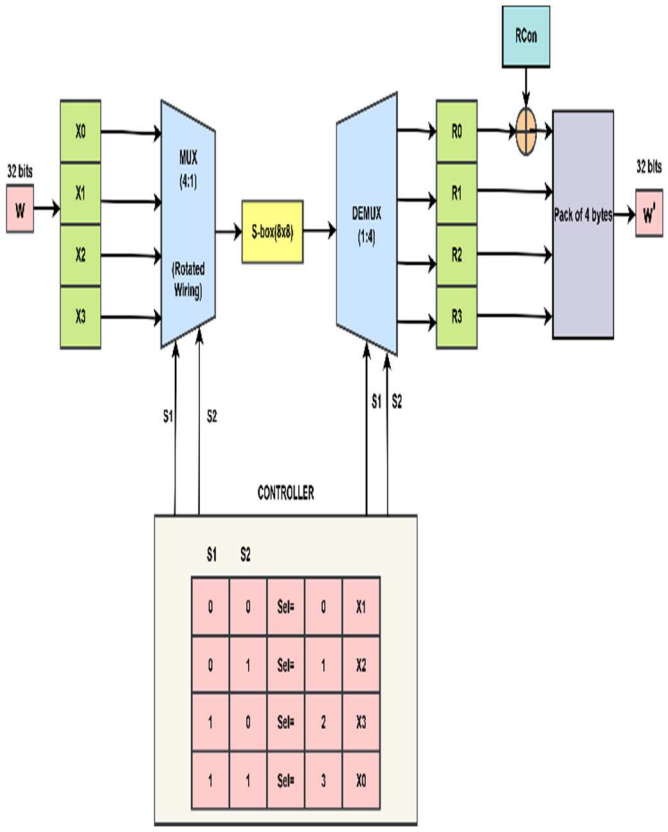

Fig.4 Block Diagram of the modified g-function

III. Proposed Modification Idea

The standard AES Key Expansion introduces nonlinearity and diffusion with the help of the g-function. Conventionally g-function operates on a 32 bit word through cyclic left shift followed by four parallel substitution operations and an xor addition of round constant on the most significant byte. This process incurs significant hardware resources due to separate rotation layer and several s-boxes.

The proposed modified g-function aims to mitigate these restrictions. The separate rotation logic has been replaced with a controlled byte selection process. The 32 bit word is divided into 4 bytes and taken as input through a 4:1 multiplexer. Instead of rotating the word physically and separately, the multiplexer which is controlled by a controller consisting of two select lines, selects the byte to be outputted in the order of the equivalent rotation order. Subsequently the rotation is achieved without performing wide explicit rotation logic. The output byte from the multiplexer is passed through the s-box.

Another structural optimization proposed is the usage of one single 8-bit s-box serially instead of using 4 parallel s-boxes simultaneously. In contrast to the original g-function where each rotated byte is passed through an s-box of 8-bits concurrently, the modified g-function uses a single 8 bit s-box where each of the 4 bytes are processed iteratively with each byte per cycle. This serialized reuse of s-box reduces the hardware resource complexity.

The output of the substitution layer is the passed through a 1:4 demultiplexer again controlled by a controller. The select lines in the controller performs 4 small sub cycles the output of which determines in which slot the output byte is to be placed in the demultiplexer. This ensures the order of the output bytes is maintained even though the internal operations are being performed serially.

The 4 bytes outputted are stored in a register. The Round constat addition layer is performed in accordance to the original g-function with the round constant being xor-ed with the most significant byte. The remaining bytes are not required to perform this operation thus making the output analogous to the original g-function.

The entire process is driven by a controller which controls which output is to be produced by the multiplexer, enables the s-box and the round constant operation for the particular byte and ensures that positional integrity is maintained by the demultiplexer. This key expansion modification as shown in Fig.5 with the modification focused on the g-function aims at addressing the limitation of hardware complexity experienced by the AES key scheduling while keeping the functional behaviour of g-function unchanged and compatibility with the original key expansion preserved. This makes the design particularly suitable for usage.

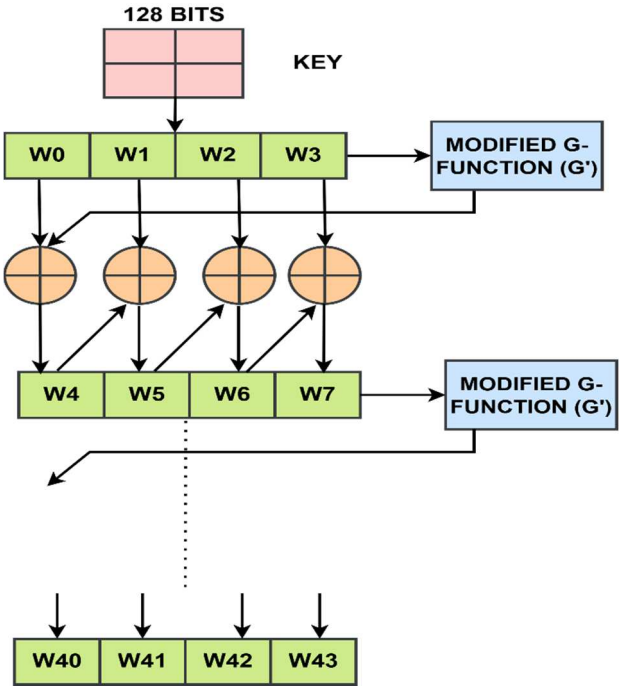

Fig.5 Block Diagram of Modified Key Expansion

IV. Result and Discussion

The proposed AES modification focusing on the key scheduling algorithm was implemented and synthesised on Vivado v2018.2 EDA tools on Artix-7 xc7a100tcsg324-1 FPGA platform in order to evaluate its hardware efficiency in comparison to the original AES key scheduling architecture. The hardware synthesis results clearly indicates that the modification incorporated in the key has led to noticeable decrease in certain FPGA resource utilizations.

As summarized in Table 1, the total number of cells has significantly reduced from 660 in the original AES Key Expansion to 381 in the modified design thereby achieving an overall reduction by 42.3%. A similar improvement has been observed in the total number of LUTs. In contrast to the original AES Key which requires 337 LUTs spanning from LUT3 to LUT6 configuration, the modified AES structure only consumes 113 LUTs spanning from LUT2 to LUT5 configuration thereby introducing a reduction of 66.5%. The total number of 8-bit s-boxes utilized by original AES key expansion is 8 whereas the proposed design employs only 1 8-bit s-box iteratively. Thus, a notable decrease of 87.5% has been observed. Since s-box is a critical component which determines the overall area of AES, this optimization plays a great role in area reduction. Furthermore, slices utilization has also indicated improvement by reducing the slice utilization by 65.2% from 325 in the original key architecture to 113 in the proposed one.

LUTs and slices forms fundamental building blocks of FPGA implementation, reduction in which directly results in improved area efficiency. The proposed design results in much more compact hardware utilization as compared to the original one making it suitable and useful in areas where FPGA implementation is required. Fig.6 shows the RTL schematic of the proposed Key Expansion unit.

TABLE 1. QUANTITATIVE COMPARISON (ORIGINAL VS MODIFIED AES KEY EXPANSION)

Metric	**Original AES Key Expansion**	**Proposed Modified Key Expansion**	**Improvement**
Total Cells	660	381	42.3% reduction
Total LUTs	337 (LUT3–LUT6)	113 (LUT2–LUT5)	≈66.5% reduction
S-box instances	8 × (256×8)	1 × (256×8)	87.5% reduction
Slice (LUTs)	325	113	≈65.2%

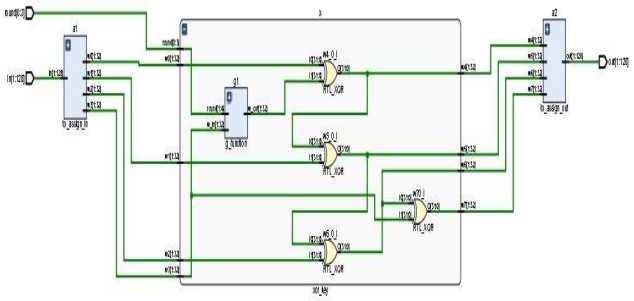

Fig.6 RTL Schematic of Proposed Key Expansion unit

V. Conclusion

This paper outlines an AES key-expansion architecture that is area-optimized to implement cryptography systems on FPGA. The proposed design will result in significant cost savings in the use of hardware resources by re-arranging the internal order of the key-expansion block and removing unnecessary hardware elements. The results of the synthesis indicate that the overall number of cells decreased by 42.3 percent, LUTs usage diminished by approximately 66.5 percent, and S -boxes instances fell by 87.5 percent compared to traditional AES key -expansion. A further examination of the key-expansion block proves a significant reduction in the use of slice-LUT. These results demonstrates how effective the suggested methodology in reducing the size of the hardware, but at the same time keeping the compatibility with the standard AES algorithm. This means that the optimized architecture is ideal in those applications where the efficient use of FPGA resources is one of the main design considerations.

References

[1] S. Ahmed *et al*., "Lightweight AES Design for IoT Applications: Optimizations in FPGA and ASIC With DFA Countermeasure Strategies," in *IEEE Access*, vol. 13, pp. 22489-22509, 2025, doi: 10.1109/ACCESS.2025.3533611.

[2] R. S. Salman, A. K. Farhan and A. Shakir, "Lightweight Modifications in the Advanced Encryption Standard (AES) for IoT Applications: A Comparative Survey," *2022 International Conference on Computer Science and Software Engineering (CSASE)*, Duhok, Iraq, 2022, pp. 325-330, doi: 10.1109/CSASE51777.2022.9759828.

[3] A. Rathod and K. Patel, "Internal Re-keying based modified AES," Indian Journal of Science and Technology, vol. 18, no. 1, pp. 85–94, Jan. 2025, doi: 10.17485/ijst/v18i1.2012.

[4] R. Parthasarathy, P. Saravanan, S. R. Srivatsav, and C. M. Manisha, "Lightweight implementation of AES for resource constrained environment," Analog Integrated Circuits and Signal Processing, vol. 123, no. 1, Mar. 2025, doi: 10.1007/s10470-025-02343-x.

[5] C. Boura, P. Derbez, and M. Funk, "Alternative key schedules for the AES," in Lecture notes in computer science, 2024, pp. 485–506. doi: 10.1007/978-3-031-54773-7_19.

[6] R. Sudharshan, Y. P and A. Prathiba, "Enhancing AES Security and Performance: A Novel 8-bit Architecture with Unique Key Expansion for IoT Applications," *2024 IEEE Silchar Subsection Conference (SILCON 2024)*, Agartala, India, 2024, pp. 1-6, doi: 10.1109/SILCON63976.2024.10910811.

[7] G. Leurent and C. Pernot, "New representations of the AES key schedule," J. Cryptology, vol. 38, no. 1, Nov. 2024, doi:10.1007/s00145-024-09522-5.

[8] S. Liu, Y. Li and Z. Jin, "Research on Enhanced AES Algorithm Based on Key Operations," *2023 IEEE 5th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, Dali, China, 2023, pp. 318-322, doi: 10.1109/ICCASIT58768.2023.10351719.

[9] Z. Cao, G. Yi, B. Wu, J. Li and D. Xiao, "Analysis And Improvement of AES Key Expansion Algorithm," *2022 International Conference on*

Artificial Intelligence and Computer Information Technology (AICIT), Yichang, China, 2022, pp. 1-5, doi: 10.1109/AICIT55386.2022.9930239.

[10] Z. Rahman, X. Yi, M. Billah, M. Sumi, and A. Anwar, "Enhancing AES Using Chaos and Logistic Map-Based Key Generation Technique for Securing IoT-Based Smart Home," Electronics, vol. 11, no. 7, art. No. 1083, 2022, doi: 10.3390/electronics11071083.

[11] National Institute of Standards and Technology, Advanced Encryption Standard (AES), Federal Information Processing Standards Publication (FIPS) 197, NIST FIPS 196-upd1, May 2023. doi: 10.6028/NIST.FIPS.197-upd1.

[12] J. Daemen and V. Rijmen, "AES Proposal: Rijndael," in Proceedings of *the First Advanced Encryption Standard (AES) Candidate Conference*, National Institute of Standards and Technology (NIST), 1999.