FPGA Implementation of AES Algorithm

Haripriya G
Electronics and Communication Eng.
KLE Technological University
Hubballi, India
haripriyag796@gmail.com

Hema Basaligundi
Electronics and Communication Eng.
KLE Technological University
Hubballi, India
hemabasaligundi9945@gmail.com

Manupriya Padekanur
Electronics and Communication Eng.
KLE Technological University
Hubballi, India
manupriyapadekanur@gmail.com.com

Ganga I Aralikatti
Electronics and Communication Eng.
KLE Technological University
Hubballi, India
aralikattiganga544@gmail.com

Prof.Rajeshwari M
Electronics and Communication Eng.
KLE Technological University
Hubballi, India
rajeshwari_m@kletech.ac.in

***Abstract*—At present, encryption is key to safeguard confidential information from unauthorized access. It means using special algorithms and encryption keys to change readable data into unreadable data.AES is symmetric encryption algorithm adopted by the U.S.Government as a replacement of DES algorithm. AES supports 3-key sizes: 128 bit, 192 bit and 256 bit, with AES -128 being the most commonly used due to its balance of security and performance. Because it is fast and simple, symmetric key encryption which uses a single key for both encryption and decryption, is used in many cases. Advanced Encryption Standard(AES) is a symmetric block cipher that is used extensively in the encryption of electronic data. Out of its various implementations, AES-128 is the most commonly used, with a 128-bit key size and encrypting 128-bit-sized blocks of data and to get 128-bit cipher text. The encryption process goes through 10 rounds of processes like SubBytes, ShiftRows, MixColumns, and AddRoundKey, which are computed to have high confusion and diffusion properties. AES-128 is optimally balanced in terms of security, efficiency, and performance and thus finding use in hardware and software in various industries. Due to its strength and immunity against known attacks on electronic data, passwords etc. AES-128 has been the standard for encrypting data in commercial and government systems worldwide.AES algorithm has many applications such as online banking, ATM machines and military and government communication.**

***Index Terms*—*AES, Advanced Encryption Standard, Symmetric Encryption, Block Cipher, AES-128, Key Expansion, SubBytes, ShiftRows, MixColumns, AddRoundKey, Cryptography, Data Encryption, Encryption Algorithm, Confidentiality, Information Security, Ciphertext, Plaintext, S-box, Round Key, Encryption Rounds, Security Protocols*.**

I. Introduction

AES is the Advanced Encryption Standard, a symmetric block cipher now used all over the world to securely encrypt data. Created to take the place of the older DES, the National Institute of Standards and Technology (NIST) introduced AES in 2001 as the result of an open contest to pick a strong and efficient method for encryption. AES can be implemented in three sizes: 128, 192 and 256 bits and AES-128 is most oftenly used because it works well with security, speed and resource needs. It handles 128-bit blocks and performs 10 cycles of complex operations, using a 128-bit key. Each round carries out four steps: SubBytes, ShiftRows, MixColumns and AddRoundKey which make sure the plaintext data is both confused and diffused.AES-128 is resistant to known security threats and its positive results from extensive testing have been confirmed by experts. It is found everywhere in technology, helping to secure messages on the internet (TLS, VPNs), encrypt files and disk drives and improve the security of mobile devices and embedded systems. Because it is flexible, quick and secure, AES-128 is at the heart of today's digital security systems.

With the onset of the digital era, information security against unauthorized access and cyber attacks has been an uppermost priority. Cryptographic algorithms are fundamental instruments applied in ensuring confidentiality, integrity, and authenticity of information on the digital platform. Out of all algorithms, the Advanced Encryption Standard (AES) is most widely employed symmetric block cipher to encrypt precious data. Specifically standardized by the National Institute of Standards and Technology (NIST) as FIPS Publication 197 in 2001, AES entered the market to replace the antiquated Data Encryption Standard (DES), offering improved security and performance. AES possesses constant block sizes of 128 bits and supports 128, 192, or 256-bit keys with 10, 12, or 14 rounds of encryption, respectively, thus being versatile for a range of security applications.

AES's internal structure is based on the Rijndael cipher, which was selected by an open, global competition sponsored by NIST. It employs a series of substitution-permutation transformations—i.e., SubBytes, ShiftRows, MixColumns, and AddRoundKey—which are mathematically stable and computationally efficient. Due to its outstanding resistance to all known cryptanalytic attacks as well as high-efficiency implementations in software and hardware, AES is used extensively in secure communication, digital data storage, financial applications, and government usage.

While software AES implementations are commonly used on general-purpose processors, they are normally inadequate

for high-throughput, low-latency, or real-time encryption uses, such as in embedded systems, network routers, or Internet-of-Things (IoT) devices. Field-Programmable Gate Arrays (FPGAs) offer a compelling alternative platform for implementing AES. FPGAs offer the flexibility of software, on one hand, and dedicated-hardware speed, on the other hand. They allow designers to harness parallelism, custom data paths, and hardware pipelining in order to accelerate encryption operations far beyond that possible using software.

Use of AES on FPGA allows for the use of trade-offs between speed, area, and power consumption, depending on what application one wishes to implement. For instance, a pipelined implementation of AES may be highly throughput-focused suitable for gigabit data encryption, while an iterative algorithm may be area-optimized suitable for resource-constrained applications. Additionally, FPGA implementations are field-upgradeable, hence can be updated to support varying key sizes, additional cryptographic primitives, or support new security standards without the need to install new hardware.

II. Related Work

FPGA design of the Advanced Encryption Standard (AES) has been a major area of research, with the objective to attain optimized performance metrics in the form of speed, area, and power. Different studies have shown novel methods to tailor the AES algorithm to various applications, starting from high-end systems to low-power IoT devices.

Chi-Jeng Chang et al. suggested an 8-bit AES implementation using Block RAMs (BRAMs) in Xilinx FPGAs, shifting operations such as S-box, ShiftRows, and KeyExpansion to BRAMs to reduce resources utilization. Their implementation provided a substantial throughput boost about 12 times the ASIP-based one—with an added expense of merely 8 [1].

In another research, Pritamkumar Khose and Prof. Vrushali Raut targeted the area reduction and power savings with the standard AES throughput. In their method, they substituted conventional combinational S-box logic with BRAM-based look-up tables, minimizing LUT and slice utilization by as much as 89 [2]

Zabina Kouser and others applied AES on FPGA through repetitive looping architecture for encryption and decryption. Their 128-bit block VHDL-based implementation was aimed at real-time applications like ATMs, wireless communication, and smart cards. Xilinx ISE tools helped them to investigate reconfigurable hardware flexibility, and they proved that hardware-based AES provides improved speed and physical key security compared to sw implementation [3].

Additional optimization was obtained by Mazen El Maraghi et al., who proposed a high-speed AES-128 encryption core by applying an iterative looping structure in conjunction with sub-pipelining. Their implementation, built on a Xilinx Virtex-5 FPGA, reached a speed of 1.33 Gbps and consumed only 303 slices. They also included a Java-based software interface connected through a MicroBlaze processor, facilitating real-time user interaction with the hardware and forming a full embedded system evaluation platform [4] .

Responding to the specific requirements of the IoT field, Jeyvarshni Vimalkumar et al. designed an optimized lightweight version of the AES algorithm for low-power, resource-restricted platforms. Deployed on an Artix-7 Basys-3 Field Programmable Gate Array, their design reduced the rounds to a minimum, reduced the MixColumns and SubBytes operations to their simplest forms, and incorporated novel transformation methods such as Zig-Zag pre-processing and Shift Row-Column operations. Consequently, they obtained more than 80 [5]

In another research, GuanLi Peng and SongBai Zhu stated that AES is a widely accepted encryption standard that provides secure data transmission, but it is highly efficient in hardware implementation. FPGA-based implementations have become popular owing to their flexibility, parallelism, and speed over ASICs. Researchers have optimized AES on FPGA with pipeline methods, improvements in key expansion, and resource-optimized architectures to improve throughput and lower power consumption. These studies together demonstrate that FPGA offers a real-world and high-performance platform for implementing AES in real-time communication systems [6]

Joseph Sunil, Suhas H S, Sumanth B K,Santhameena extensively investigated FPGA-based implementations of AES to enhance speed and efficiency of encryption. Most of the works concentrated on reconfigurable design, pipelining, and key expansion methods to increase throughput and resource usage. FPGA provides parallelism and flexibility over ASIC and hence is best for real-time secure communication. Research also identifies trade-offs between hardware use, speed, and security performance. Overall, FPGA is an efficient platform for optimizing AES in real-world applications [7]

Sridevi. K, M. Santhanalakshmi have contributed towards enhancing AES performance via software and hardware implementations. they suggested several AES implementations to enhance speed, area, and security. FPGA and VHDL-based architectures minimized delay and clock cycles, whereas pipelining and altered S-box techniques improved throughput. Dynamic key methods improved data security further in communication systems, demonstrating that optimal AES cores can provide superior performance and reliability [8]

Chen, Hu, and Li (2019)(2019) deployed a high-speed AES processor on FPGA with deep pipelining and utilized very high throughput of 31.3 Gbps with very low latency of 134 ns. The proposed design effectively supports large-scale data encryption and yields high security. The present contribution demonstrates that FPGA-based AES is extremely appropriate for real-time and high-speed communication systems [9].

Kumar et al. (2021) describe a new FPGA-based AES architecture that does away with conventional LUTs in the SubBytes and InvSubBytes operations based on combinational logic, thus minimizing delays. They propose a sub-pipelined architecture and a Modified Positive Polarity Reed–Muller (MPPRM) architecture for the SubBytes operation, along with a power-efficient MixColumns/InvMixColumns design, to enhance speed and minimize area and power consumption.

This renders their implementation quite fast and hardware-efficient in comparison to standard AES architectures [10]

Mane and Mulani (2018) designed an FPGA-based AES based on Xilinx System Generator and MATLAB keys with 14.1 Gbps throughput at 1102 MHz using only 121 slice registers. Their design is a high-speed but area-efficient AES solution that can be used in secure real-time applications. [11]

SU Jonwal, PP Shingare describes a hardware-in-the-loop (HIL) AES-128 implementation on FPGA for real-time interactive operation of the FPGA-based AES core with other external systems through SDK-driven Xilinx Artix-7 hardware. The design is shown to be flexible and modular for prototyping and academic purposes. AES-128 was implemented on FPGA using a hardware-in-the-loop (HIL) technique based on Xilinx Artix-7 to facilitate real-time testing and verification. This approach maintains flexible integration with other systems, thus suitable for safe and high-speed prototyping requirements [12].

Keshav Kumar et al. (2020) deployed AES on an Artix-7 FPGA and inspected hardware metrics like registers, LUTs, and I/O utilization. Their comparative study showcases the resource usage and performance trade-offs in various FPGA-based AES designs [13].

Sai Srinivas and Akramuddin (2016) applied the AES Rijndael algorithm (128, 192, and 256-bit) on a Xilinx Virtex-7 FPGA with Verilog HDL. Their optimization did S-Box and GF(2) operations using LUTs in order to achieve good throughput and efficient use of resources [14].

These different contributions as a whole demonstrate the multi-faceted character of FPGA as an AES implementation platform. From the maximization of throughput to area and energy minimization, the literature has had established proven design methods for low-energy and secure encryption scalable to a wide variety of technological requirements.

III. Methodology

Implementation of the AES-128 algorithm on FPGA was achieved in a structured and modular manner. A deep understanding of the AES-128 encryption process with its fixed 128-bit key and block size and 10 rounds of transformation consisting of SubBytes, ShiftRows, MixColumns, and AddRoundKey was first created. Every one of these transformations is important for causing confusion and diffusion in the encrypted data. Concurrently, the Key Expansion procedure was analyzed to determine how the initial 128-bit key is converted into 11 round keys based on operations like RotWord, SubWord, and round constant XOR.

After setting up the theoretical framework, the implementation started with creating separate modules in Verilog HDL. Every AES operation—SubBytes, ShiftRows, MixColumns, AddRoundKey, and Key Expansion—was implemented as a separate independent and reusable module. These modules were simulated separately to ensure that they worked as desired. After unit testing, each module was assembled into an entire AES encryption core. The core utilized a finite state machine (FSM) to manage the operation sequencing for each round of AES, with an initial AddRoundKey stage, followed by nine full rounds and a final round which omits MixColumns.

To confirm the design, a testbench was created in Verilog that allowed for known 128-bit plaintext and key inputs using NIST standard test vectors to check correct functionality of the design. The simulation was executed with Vivado Simulator, and output waveforms were inspected to verify that every transformation step generated correct intermediate and final ciphertext outputs. Extra care was taken to debug timing and data propagation issues, especially in S-box operations and key expansion synchronization.

Following successful simulation, the design was synthesized and run on an FPGA development board. Board-level implementation included the specification of proper clock and pin constraints. UART was employed for serial output of input and output data as an optional interface, and onboard LEDs or displays to display encryption status or intermediate results. Lastly, the entire system was verified by hardware testing, which assured correct functioning, acceptable timing performance, and efficient resource use on the FPGA device.

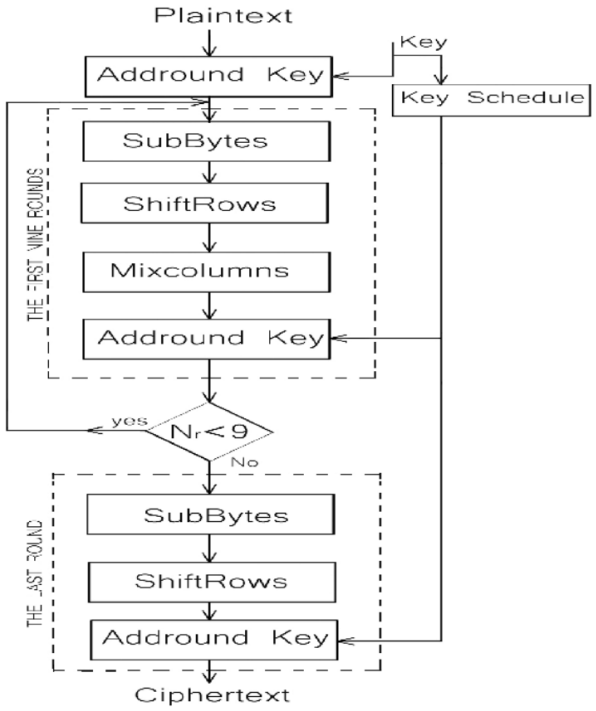


Fig. 1. Flow Chart of AES Algorithm

IV. Implementation

The FPGA hardware implementation of the AES algorithm consists of translating its mathematical structure and encryption processes into hardware using a Hardware Description Language (HDL) like Verilog or VHDL. The implementation is based on a modular approach, with each of the AES transformation stages implemented as a separate hardware

block. This is followed by the SubBytes transformation, a non-linear byte substitution in which each byte of the 128-bit input state is substituted through a precomputed substitution box (S-box). In hardware, this can be implemented economically as a lookup table, often in the form of Block RAM or ROMs based on logic. In continuation, the ShiftRows transformation adds diffusion by cyclically shifting the bytes within each row of the 4x4 state matrix: the first row is not moved, and the second, third, and fourth rows are shifted left by one, two, and three bytes respectively.

SubBytes is one of the four main transformation processes employed in the Advanced Encryption Standard (AES) algorithm, most notably AES-128 that employs a 128-bit key. It is an irreversible non-linear byte substitution process that is responsible for enhancing the security of the cipher by introducing confusion. The transformation is carried out on every byte of the AES state matrix during the encryption rounds.In AES, the data being encrypted is expressed as a 4×4 array of bytes called the state. When it comes to the SubBytes operation, all bytes in this array are independently replaced with an S-box, which is a static lookup table that holds 256 potential values. The S-box is non-linear and made resistant to known cryptanalysis methods like linear and differential attacks.The AES S-box is constructed on two mathematical operations. To start with, a byte is mapped as an element of the finite field GF(2 power 8), and its multiplicative inverse is computed (with 0 mapped to itself). An affine transformation is applied on the inverse in the second step. This double step defines a complicated, highly non-linear mapping of input byte to output byte that is necessary for security of the cipher.An instance of SubBytes would be replacing a byte like 0x53 within the state matrix. When queried in the AES S-box, 0x53 could be replaced with 0xED, and such a replaced value is used to substitute the original byte within the matrix. This operation is done independently for every 16 bytes of the 4×4 state matrix.

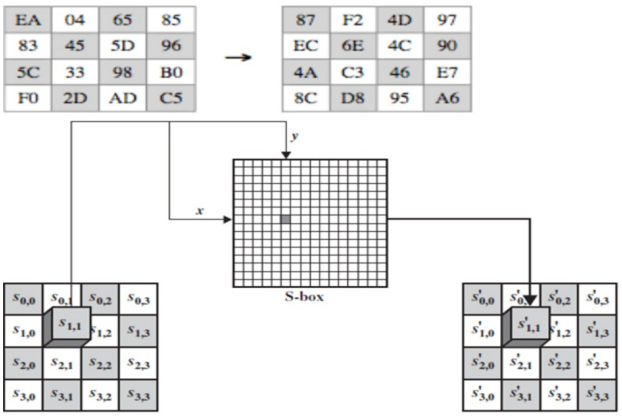


Fig. 2. Subbytes .

ShiftRows is the second operation in every AES encryption round (following SubBytes), and it achieves inter-byte diffusion throughout the state matrix. The state matrix in AES is a 4x4 matrix of bytes (total 16 bytes), and ShiftRows acts row-wise on this matrix. In this operation, the bytes within each row are shifted to the left by some number of positions:The first row is left unchanged.The second row is shifted left 1 byte.The third row is shifted left 2 bytes.The fourth row is shifted left 3 bytes.These shifts are circular, i.e., the bytes shifted out from the left re-enter from the right.

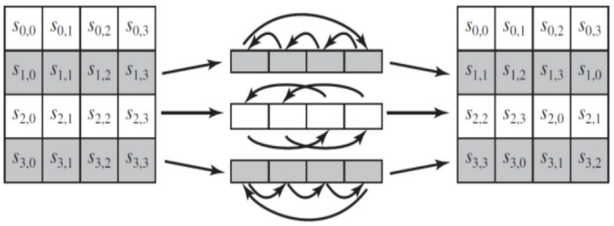


Fig. 3. shiftrows .

MixColumns is one of the four fundamental transformation processes in the AES-128 encryption scheme. It is used on the state matrix for each round of AES (excluding the last round) and serves to facilitate diffusion, a cryptographic attribute that guarantees minor variations in the input will cause extensive variations in the output. MixColumns acts upon the columns of the 4×4 state matrix, combining the data in each column in a linear, mathematical manner. The state within AES is a 4×4 matrix with each entry being a byte (8 bits). In the MixColumns operation, every column in this state matrix is used as a 4-byte vector, and it is multiplied by a constant 4×4 matrix in order to perform multiplication in the finite field GF(2 power 8). This results in a new column that is substituted for the old one. The operation distributes the impact of every byte throughout the whole column so that an alteration in one byte impacts all four bytes in the same column following the transformation. All bytes in a column are substituted by a function of all four bytes in said column, which includes multiplication by 1, 2, or 3 in GF(2 power 8).

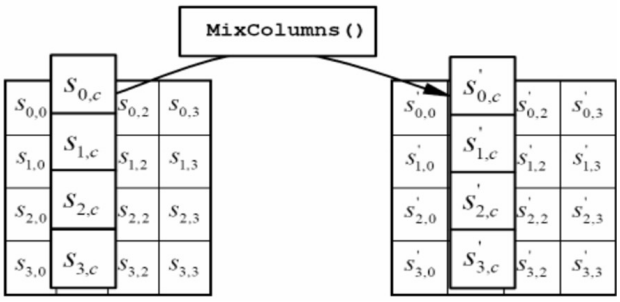


Fig. 4. mixcolumn .

The AddRoundKey step is a very important part of the AES (Advanced Encryption Standard) algorithm, executed at the beginning of the encryption process and at the end of each round, including the last one. At this step, the 128-bit state matrix, which contains the intermediate result of the encryption, is added with a 128-bit round key, each represented

as 4×4 matrices of bytes. The operation is done by doing a bitwise XOR between every byte of the state and the analogous byte of the round key. This conversion is simple and efficient, involving low logic, and is reversible by nature, so it is effective for both encryption and decryption steps. The first AddRoundKey is particularly crucial because it makes the encryption process key-dependent from the very beginning.

Although it is simple, AddRoundKey is crucial to the security of AES. Without it, the algorithm would be comprised solely of static operations such as SubBytes, ShiftRows, and MixColumns, that are independent of the key and would thus give the same output for the same input each time—making the cipher predictable and insecure. The addition of AddRoundKey makes sure that the output is varied with varying keys, adding much-needed key-dependent confusion to the cipher. Each round key utilized within AddRoundKey is generated from the initial cipher key based on a procedure known as Key Expansion, which guarantees different transformations during rounds. This renders it significantly more challenging for attackers to identify patterns or reverse-engineer the key and thus further secures AES against cryptographical attacks.

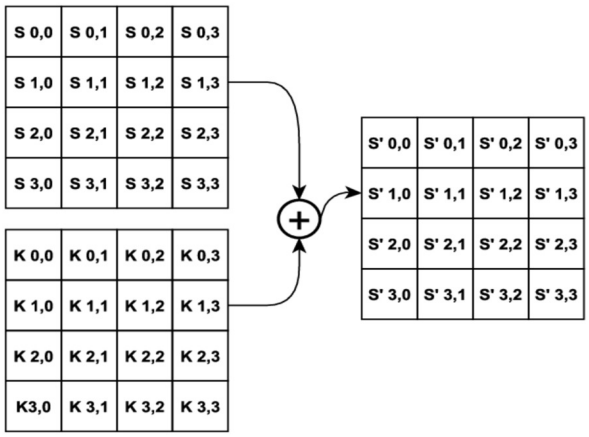


Fig. 5. addroundkey.

Key Expansion, or the Key Schedule, is a critical component of the AES-128 algorithm that prevents each encryption round from accessing the same, repetitive key derived from the initial cipher key. Because AES-128 carries out 10 encryption rounds and needs a different key prior to the first round, the process of key expansion will have to come up with a total of 11 round keys, each 128 bits in length. The initial 128-bit key is broken down into four 32-bit words (W[0] through W[3]), and from them the algorithm sequentially creates 44 words (W[0] through W[43]) to yield the keys needed. The process is a series of transformations that add diffusion and non-linearity to better strengthen the encryption. For every fourth word (i.e., when the word index is a multiple of four), a specific transformation referred to as the Key Schedule Core comes into play. This foundation consists of three key operations: RotWord, which applies a left circular shift to a 4-byte word; SubWord, which replaces each byte by applying the AES S-box for non-linear transformation; and Rcon, wherein a round constant is XORed with the first byte to add uniqueness and randomness to the round key generation.

For non-multiple-of-four word indices (i.e., W[i] where i is not divisible by 4), the word is merely created by XORing the prior word (W[i1]) with the word four words earlier (W[i4]). This action makes for a compact and efficient expansion of the key without weakening the dependence on the original key. The combination of these mathematical conversions results in a series of round keys that are all cryptographically connected to the original cipher key but differentiated enough to protect against attacks related to the key. This larger key schedule is important because it keeps attackers from being able to predict round keys or take advantage of patterns in encrypting. The identical process is replicated at decryption time; however, the round keys are used in reverse. This effective yet powerful process for the creation of varied round keys from one key is one of the cornerstones that renders AES a sound symmetric encryption standard globally used in contemporary cryptographic systems.

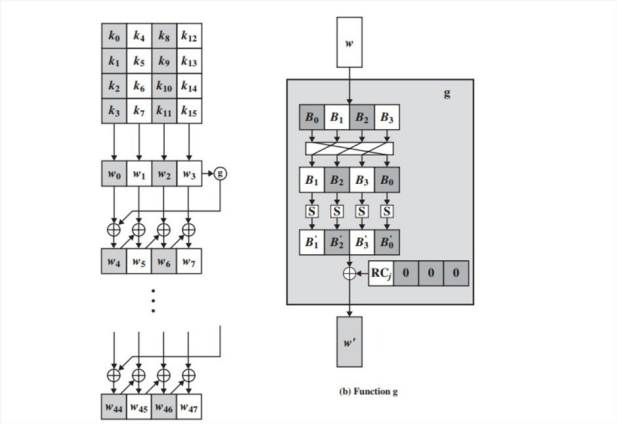
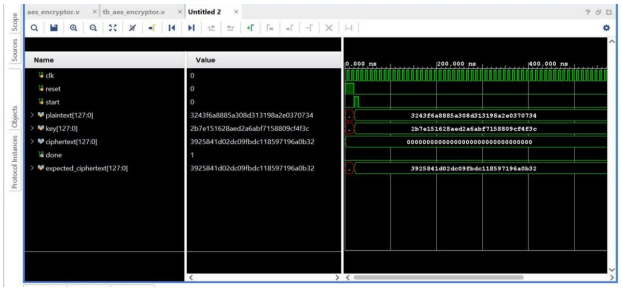


Fig. 6. key_expansion.

V. Results and Discussions

Successful implementation of the AES-128 encryption algorithm in FPGA using Verilog HDL was achieved and verified using simulation within Xilinx Vivado. The design was shown to function correctly by generating ciphertext outputs that equated to standard NIST test vectors. When synthesized for a target FPGA device, the AES core demonstrated effective resource usage with moderate utilization of slice LUTs, registers, and block RAMs, yet with high operating frequency that meets real-time usage.

AES was implemented successfully on a Spartan-6 FPGA, with verification and output seen virtually through Chipscope (ILA). The design performed encryption correctly with 128 bits, where outputs were in agreement with expected ciphertext values, verifying functional correctness. Resource usage was within tolerable limits for Spartan-6, with optimal utilization of LUTs, registers, and BRAM. Implementation was done at a clock frequency of around 100MHz. Chipscope helped



tremendously in real-time debugging where internal AES stages like round transformations and key expansion could be observed. It is evident from this project that FPGA-based AES implementations not only become viable but are also highly secure, fast, and scalable, which renders them applicable for embedded and secure communication applications.

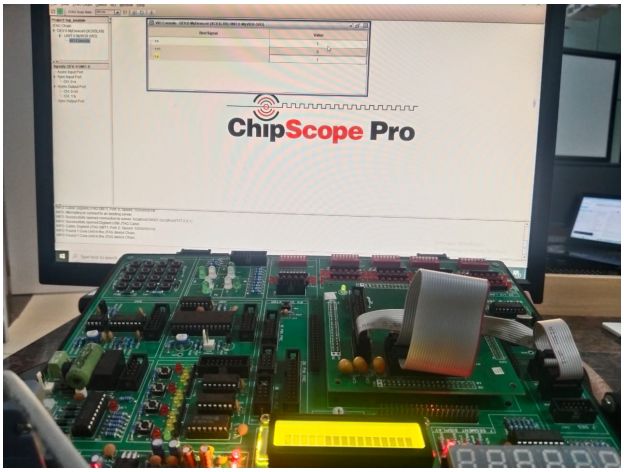


Fig. 8. .

VI. Conclusion

The application of the AES algorithm on FPGA platforms validates the ability to make high-performance and secure encryption solutions appropriate for embedded and real-time systems. Using modular Verilog-based design, every AES operation—SubBytes, ShiftRows, MixColumns, AddRoundKey, and Key Expansion—was successfully implemented and incorporated into a full encryption core. Simulation and verification via Vivado assured the correctness of the encryption process in accordance with the NIST FIPS 197 standard.

Through the parallelism and configurability of FPGAs, the design had efficient encryption with the added benefits of design flexibility and reusability.

Subsequent developments of this research can extend towards the improvement in resistance to side-channel attacks, combining AES with other cryptoprograms like RSA or ECC for hybrid security, and applying dynamic key management mechanisms on-chip. These enhancements will improve further the practicality of AES hardware cores in secure communication systems, IoT applications, and other data-critical devices.

VII. futurescope

The modular design is scalable to AES-192 and AES-256, and facilitates the adaptability of the design to be integrated within more elaborate cryptographic systems. Improvements in the future can be made in the areas of power minimization, side-channel attack resistance, and dynamic key management to further extend its applicability within these embedded and IoT systems.

References

[1] C.-J. Chang, C.-W. Huang, H.-Y. Tai, M.-Y. Lin, and T.-K. Hu, "8-bit aes fpga implementation using block ram," in *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 2654–2659, 2007.

[2] P. N. Khose and V. G. Raut, "Implementation of aes algorithm on fpga for low area consumption," in *2015 International Conference on Pervasive Computing (ICPC)*, pp. 1–4, 2015.

[3] Z. Kouser, M. Singhal, and A. M. Joshi, "Fpga implementation of advanced encryption standard algorithm," in *2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, pp. 1–5, 2016.

[4] M. El Maraghy, S. Hesham, and M. A. Abd El Ghany, "Real-time efficient fpga implementation of aes algorithm," in *2013 IEEE International SOC Conference*, pp. 203–208, 2013.

[5] J. Vimalkumar, H. R. Babu, and B. M, "Fpga implementation of modified lightweight 128-bit aes algorithm for iot applications," in *2023 IEEE International Symposium on Smart Electronic Systems (iSES)*, pp. 306–309, 2023.

[6] G. Peng and S. Zhu, "Fpga implementation of aes encryption optimization algorithm," in *2021 International conference on intelligent transportation, big data & smart city (ICITBS)*, pp. 650–653, IEEE, 2021.

[7] J. Sunil, H. Suhas, B. Sumanth, and S. Santhameena, "Implementation of aes algorithm on fpga and on software," in *2020 IEEE International Conference for Innovation in Technology (INOCON)*, pp. 1–4, IEEE, 2020.

[8] M. Santhanalakshmi *et al.*, "An efficient implementation of aes algorithm using custom key," in *2022 International Conference on Intelligent Innovations in Engineering and Technology (ICIIET)*, pp. 268–272, IEEE, 2022.

[9] S. Chen, W. Hu, and Z. Li, "High performance data encryption with aes implementation on fpga," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pp. 149–153, IEEE, 2019.

[10] T. M. Kumar, K. S. Reddy, S. Rinaldi, B. D. Parameshachari, and K. Arunachalam, "A low area high speed fpga implementation of aes architecture for cryptography application," *Electronics*, vol. 10, no. 16, p. 2023, 2021.

[11] P. Mane and A. Mulani, "High speed area efficient fpga implementation of aes algorithm," *International Journal of Reconfigurable and Embedded Systems*, vol. 7, no. 3, pp. 157–165, 2018.

[12] S. U. Jonwal and P. P. Shingare, "Advanced encryption standard (aes) implementation on fpga with hardware in loop," in *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, pp. 64–67, IEEE, 2017.

[13] K. Kumar, K. Ramkumar, and A. Kaur, "A design implementation and comparative analysis of advanced encryption standard (aes) algorithm on fpga," in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pp. 182–185, IEEE, 2020.

[14] N. S. Srinivas and M. Akramuddin, "Fpga based hardware implementation of aes rijndael algorithm for encryption and decryption," in *2016 international conference on electrical, electronics, and optimization techniques (ICEEOT)*, pp. 1769–1776, IEEE, 2016.