Report

by 01fe22 bec303

Submission date: 25-Jun-2025 03:13PM (UTC+0530)
Submission ID: 2705775215
File name: Minor_Project_3__removed_removed.pdf (3.09M)
Word count: 3141
Character count: 16866

Chapter 1

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.

1.1 Motivation

As digital communication and data storage have grown at a breakneck pace, protecting sensitive data's confidentiality and integrity has turned into an imperative. Encryption is one of the primary methods utilized to do so, and of the many encryption standards on the market, the Advanced Encryption Standard (AES) is the most widely used and trusted.In particular, the AES-128 algorithm provides an excellent balance of security, performance, and resource utilization. It employs a 128-bit key and encrypts data in 128-bit blocks in a sequence of well-delineated

transformation rounds and is thus very secure against brute-force and known cryptanalysis attacks. AES-128 is NIST-standardized and is utilized worldwide in applications including secure communication, banking, cloud storage, and embedded systems.The reason for implementing AES-128 is because of its established cryptographic robustness, simplicity of implementation in software and hardware, and its performance that renders it very efficient for real-time encryption in resource-limited environments. From embedded devices to IoT systems and cloud services, AES-128 is now the cornerstone of contemporary cryptographic solutions.

1.2 Objectives

The AES-128 algorithm's goals are oriented towards data security. The primary goal is confidentiality-ensuring the information remains confidential by transforming it into a way that it cannot be read without the proper key. The second is security-through robust encryption with a 128-bit key that does not allow for easy guessing and brute-force cracking, as well as other forms of cracking, of the encryption. Third, AES-128 itself does not offer data integrity, but is generally employed in conjunction with other methods in order to confirm the data has not been altered. Fourth, efficiency-is a main goal, as AES-128 is streamlined to execute rapidly and draw upon few resources, making it accessible from big servers down to tiny devices such as smartphones. Fifth, it provides simplicity and flexibility, with an uncluttered structure that is simple to use and accommodates various key sizes. Finally, standardization and compatibility are key goals—AES-128 conforms to global standards, so it functions the same way on various systems and is used worldwide for secure communication.

1.3 Literature Survey

Implementation of AES Algorithm on FPGA and on Software

This paper gives a comparative study of the AES-128 encryption scheme applied in hardware utilizing FPGA (Field Programmable Gate Array) and in software utilizing Python. The primary aim is to represent the difference in performance, particularly speed and efficiency, between these two methodologies of encryption.The authors synthesized AES-128 in Verilog HDL in Xilinx Vivado 2017.4 for the Artix-7 FPGA. Simultaneously, an AES software implementation using Python was done. Hardware setup involved modular form with UART-based

receiver and transmitter modules for serial communication between a PC and the FPGA board. A state machine had to be created to handle user input of plaintext and keys, handle encryption, and output data formatting.The AES Top encryption module was implemented with a hard-coded 128-bit key for ease of use. Intshift and Stringoutshift supporting modules were utilized to handle 8-bit packing and unpacking of data into 128-bit blocks. UART was used to send the encrypted data back to the PC. Tera Term software was employed on the PC for monitoring and interaction with the FPGA.To compare the design, simulation and real-time testing were performed. The simulation showed the propagation delay of 40.26 ns for encryption and 80.36 ns for decryption, while the software implementation took 51 ms for encryption and 7 ms for decryption. The decryption was slower than encryption because it involves a more complex Inverse MixColumns operation and the fact that one must wait for the final round key during key expansion.Additionally, the system was scaled up to a board-to-board model where one FPGA was responsible for encryption and another for decryption. The UART interface provided secure real-time encrypted data transfer between the boards, highlighting efficient hardware-based cryptographic communication[4].

FPGA Implementation of Advanced Encryption Standard Algorithm

This paper gives a hardware-oriented method of the implementation of the AES-128 encryption algorithm in VHDL on FPGA chips, based on the main reason of satisfying the growing need for secure and high-speed data communication in digital systems.The AES-128 algorithm, which is a symmetric block cipher that has a fixed block length of 128 bits and a key length of 128 bits, is developed based on Xilinx ISE 14.4 and synthesized on Spartan-6 and Virtex-5 FPGA devices. Authors highlight the superiority of FPGA over software: improved speed, reconfigurability, physical security of the key, and lower latency.The following modules are included in the architecture: Controller: Is responsible for sequencing the operations and handling the signals.KeyRounds: Handles AES key expansion and scheduling. State Block: Where the core transformations—SubBytes, ShiftRows, MixColumns, and AddRoundKey—are executed 10 rounds (R0 to R10). The first round (R0) begins with AddRoundKey transformation, then nine normal rounds and a last round that skips the MixColumns step. Implementation employs a looping method with repetition, which reduces control flow complexity and saves area. This improves scalability and ease of modification for potential future additions such as AES-192 and AES-256[2].

FPGA Implementation of AES Encryption Optimization Algorithm

This paper describes an FPGA implementation of the AES encryption algorithm for better data security and performance improvement in encryption/decryption. The authors utilize FPGA's parallelism, flexibility, and reconfigurability to concentrate on pipeline optimization, enhanced key expansion, and bit-width conversion for bus compatibility.AES-128 is utilized, utilizing a 128-bit key and plaintext, based on a state matrix and several transformation rounds. Efficient pipeline design, minimizing redundant logic, and optimizing the S-box for encryption as well as decryption are placed high priority due to the system's balance of performance and efficiency of resources while retaining distinct logic paths based on differences in encryption and decryption key schedules.The design was deployed in Quartus II and an Altera Cyclone IV FPGA. Both simulation and hardware results validated accurate encryption and decryption of 128-bit messages, with iterative round outputs being verified for accuracy. Actual testing through packet capture tools further proved the reliability and efficiency of the system[3].

8-bit AES FPGA Implementation using Block RAM

The paper describes a low-resource FPGA execution of 8-bit AES decryption and encryption for embedded applications such as PDAs and wireless systems. Shifting resource-hungry blocks (S-box, ShiftRows, KeyExpansion) to BRAM, the design reduces logic slice utilization. It delivers 27 Mbps throughput with only 130 slices and 4 BRAMs, leaving earlier 8-bit implementations far behind. The architecture employs an 8-bit datapath taking 160 clock cycles for a complete AES encryption with efficient BRAM-based control of ShiftRows and MixColumns. Key expansion is managed by dual BRAMs with minimum hardware. The design was implemented on both Spartan-2 and Virtex-4 FPGAs successfully, with high throughput at low area usage, which makes it suitable for cost-conscious applications[1].

1.4 Problem Statement

"FPGA Implementation of AES Algorithm"

1.5 Application in Societal Context

The AES-128 algorithm serves a vital purpose in society by keeping information in our everyday life secure. In the present digital era, we share and keep personal and sensitive information with us all the time—like bank account information, passwords, medical histories, and confidential messages. AES-128 keeps such information secure and confidential even if intercepted. For instance, when we make online purchases, AES-128 is employed in encrypting our payment details so that other people cannot access them during the process of transmission. It is also applied extensively in mobile applications, secure messaging services, government networks, and cloud storage in order to avoid data breaches. In a wider social context, AES-128 facilitates trust in digital systems that are necessary to ensure the seamless operation of e-commerce, digital banking, health-care systems, and national security infrastructure. It enables individuals, organizations, and governments to communicate and store information with the confidence that it cannot be pilfered or misused. Also, its effectiveness and capacity to perform even on low-power devices assist in bridging the digital gap by supporting secure services in distant or underdeveloped areas. Through strong and stable encryption, AES-128 supports data privacy, digital freedom, and general cyber security in society.

1.6 Organization of the report

The report is mainly divided into the following parts.

CHAPTER 1

Introduction contains the information about the entire problem statement. It gives an insight about the possible ways for handling the error

CHAPTER 2

System design: Describe the block diagram of the system, alternative solutions, and the final solution.

CHAPTER 3

Implementation: Details describe the specifications, final system architecture, algorithm, flowchart.

CHAPTER 4

Result and discussion: Contains the discussion on the results obtained from the system.

CHAPTER 5

Conclusion and future scope.

Chapter 2

System Design

2.1 Substitute Bytes

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(2^8), 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.

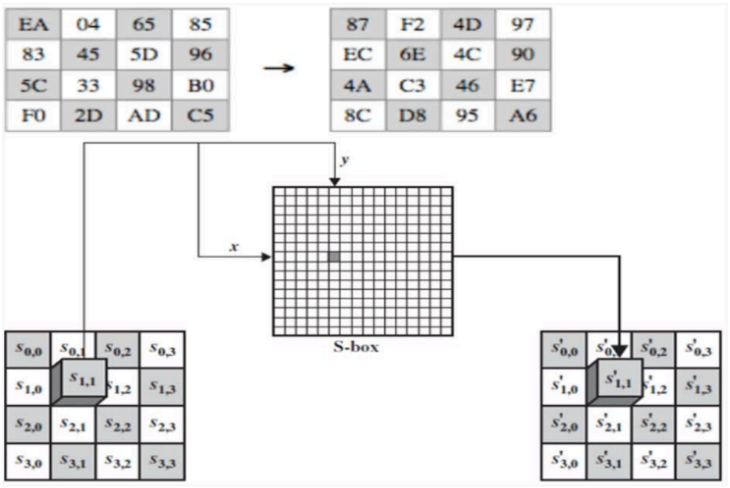

Figure 2.1: Substitute Bytes

2.2 Shift Rows

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.

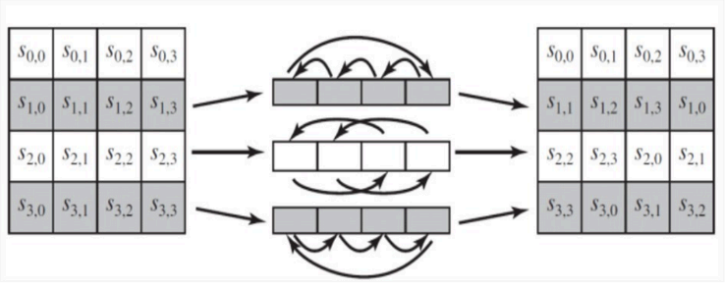

Figure 2.2: Shift Rows

2.3 Mix Columns

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(2^8). This results in a new column that is substituted for the old one. 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(2^8). Multiplication within this field employs modulo arithmetic over the AES irreducible polynomial $x^8 + x^4 + x^3 + x + 1$.

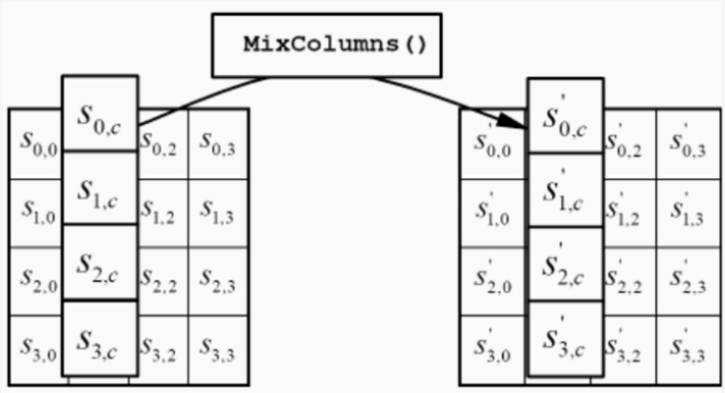

Figure 2.3: Mix Columns

2.4 Add Round Key

The operation of Add Round Key is the last operation of every AES round, also the first round before proceeding with the other transformations. It is an uncomplicated yet critical operation where the state matrix and the round key are added together using a bitwise XOR operation. The state and the round key are both 128-bit values presented as 4x4 matrices of bytes.Here, every byte of the state is XORed with the matching byte of the present round key. This step adds the key-dependent transformation to the AES process, resulting in the encryption being distinct and secure for every key. Without this step, all the AES substitutions and permutations would be fixed and deterministic.

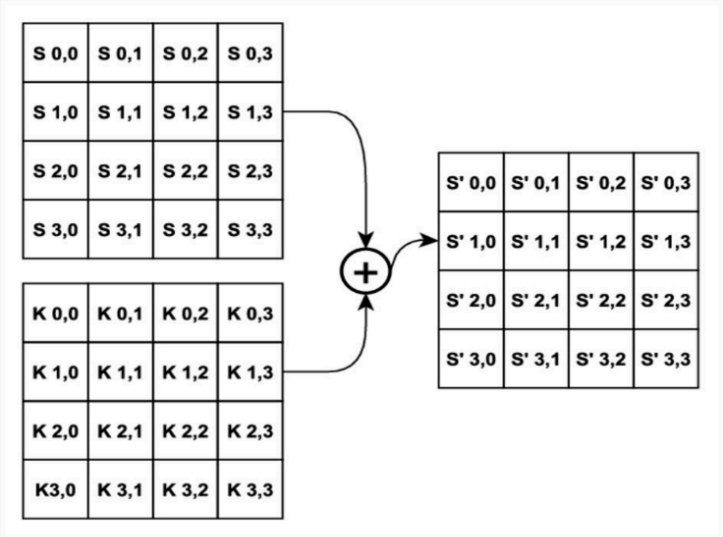

Figure 2.4: Add Round Key

2.5 Key Expansion

Key Expansion (also referred to as Key Schedule) in AES-128 is an important process that produces several round keys from the initial 128-bit encryption key. As AES-128 executes 10 rounds of encryption, and one more round key is utilized prior to the first round, the key expansion algorithm generates 11 round keys, each of 128 bits. The initial 128-bit key is broken into four 32-bit words (W[0] through W[3]). To produce the complete 44 words (W[0] through W[43]), the algorithm expands the key through the application of mathematical conversions. Each fourth word (W[4], W[8], etc.) is subjected to a unique transformation known as the Key Schedule Core, which consists of three steps: 1. RotWord – a cyclic left rotation of the 4-byte word by a single byte. 2. SubWord – replaces each byte with the AES S-box (the same applied in SubBytes). 3. Rcon (Round Constant) – a predetermined constant is XORed with the first byte to bring in the non-linearity and uniqueness of each round. The result of the Key Schedule Core is then XORed with the word four positions prior to create the new word. For other

words (not 4 multiples), each new word is produced by XORing with the preceding word. This extended key schedule makes each round of AES employ a unique 128-bit key that is generated from the initial key. This diversity in round keys enhances the security and randomness of the AES algorithm. The same key expansion routine is also employed in decryption, but with the round keys being used in a reverse direction

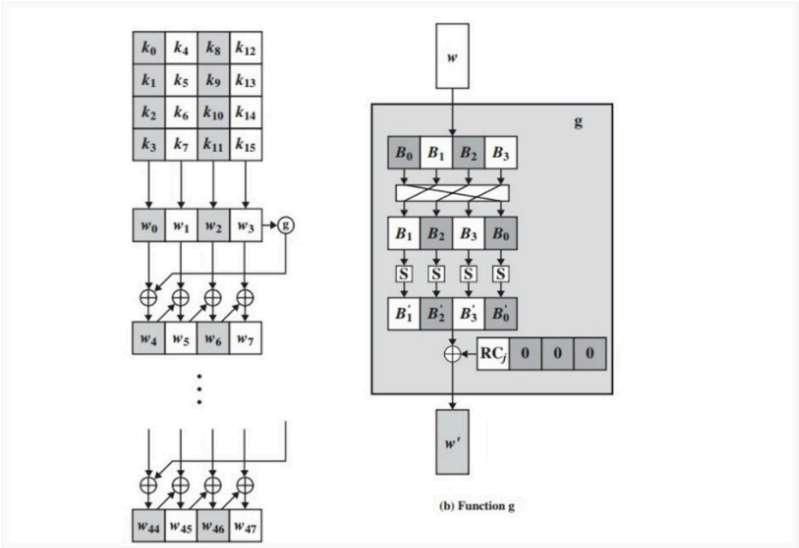

Figure 2.5: Key Expansion

2.6 AES Block Diagram

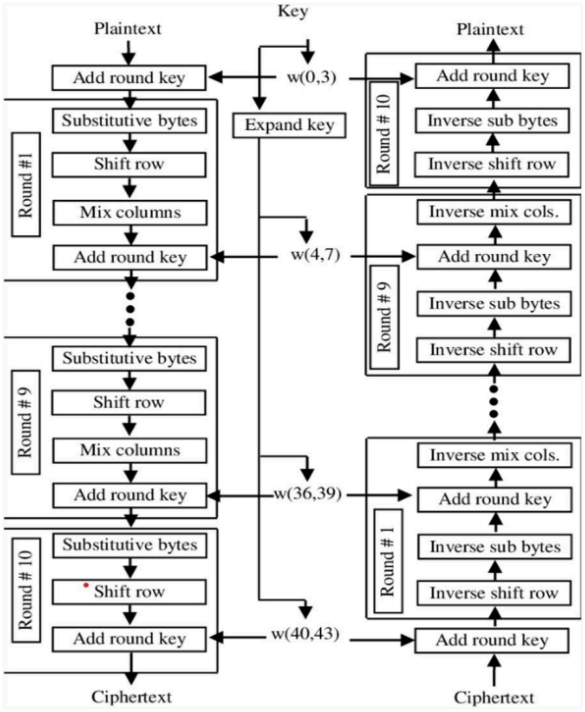

Figure 2.6: AES Block

Chapter 3

Implementation Details

3.1 AES Algorithm

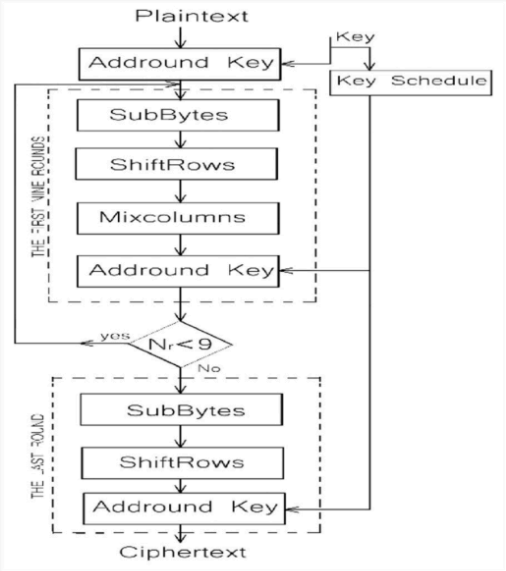

Figure 3.1: AES Algorithm

- **Plaintext to State Array:** The 128-bit plaintext is divided into a 4x4 matrix (state array) of bytes (8-bit values).
- **Key Expansion:** The initial key is expanded into a set of round keys, one for each round of encryption.
- **Rounds (Encryption):**
  1. **Sub Bytes:** Each byte in the state array is replaced with its corresponding value from an S-box (a 16x16 lookup table).
  2. **Shift Rows:** Each row of the state array is circularly shifted by a certain number of bytes.
     - The first row remains unchanged.
     - The second row is shifted one byte.
     - The third row is shifted two bytes.
     - The fourth row is shifted three bytes.
  3. **Mix Columns:** Each column of the state array is multiplied by a fixed matrix in a Galois field ($GF(2^8)$). This operation mixes the bits within each column. In the final round, the Mix Columns step is omitted.
  4. **Add Round Key:** The state array is XORed with the corresponding round key.

Chapter 4

Results and Discussions

4.1 Result Analysis

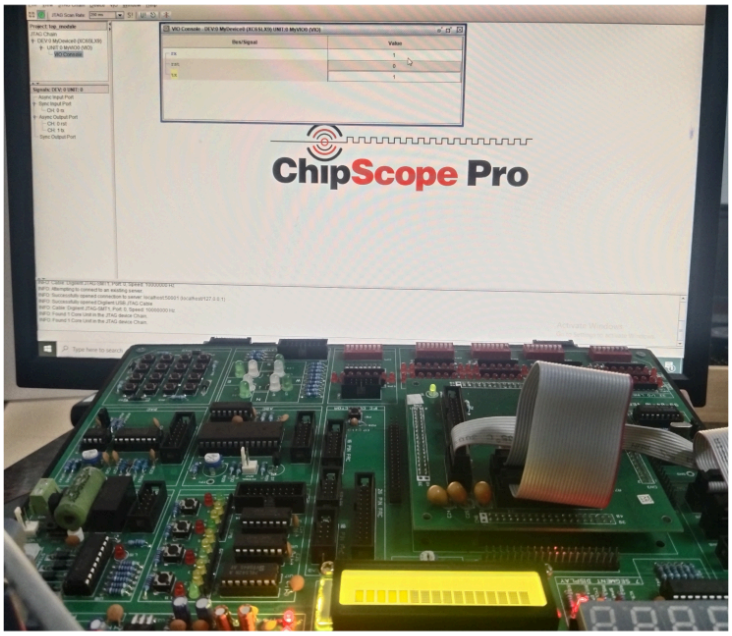

Figure 4.1: Results

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.

4.2 Simulation Waveform

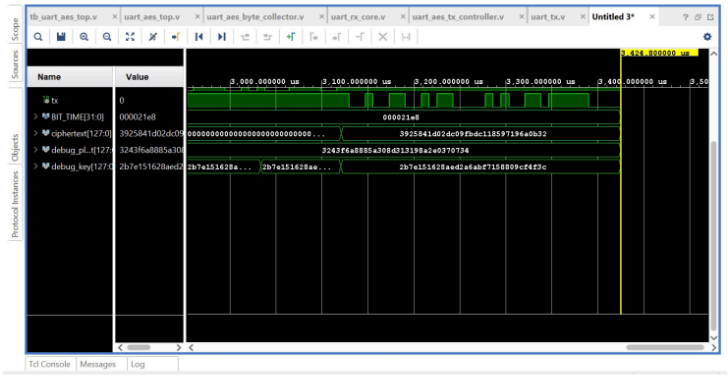

Figure 4.2: Simulation Waveform

Chapter 5

Conclusion and Futurescope

5.1 Conclusion

The project successfully implemented AES-128 encryption on an FPGA with Verilog HDL, and it showed the viability of hardware-based encryption for secure and efficient data protection. The fundamental AES operations—SubBytes, ShiftRows, MixColumns, AddRoundKey, and Key Expansion—were correctly modeled and added to a working encryption pipeline. One of the most important components of the work was real-time analysis of the system with ChipScope, which validated by monitoring signals that data was properly encrypted, sent, and received successfully, thus ensuring overall functioning of the system. Utilizing the FPGA's low-latency and parallel processing capabilities, the design was both high performance and secure in nature, hence applicable for real-time and resource-limited systems like embedded systems and IoT applications. This project also underscores the validity of AES-128 as a widely accepted and universally relied-upon encryption technique for secure data protection on digital systems.

5.2 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.

Report

ORIGINALITY REPORT

13%	8%	10%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	tolumichael.com Internet Source	3%
2	codedinsights.com Internet Source	1%
3	Liakot Ali, Ishak Aris, Fakir Sharif Hossain, Niranjan Roy. "Design of an ultra high speed AES processor for next generation IT security", Computers & Electrical Engineering, 2011 Publication	1%
4	Umesh Kumar Lilhore, Sarita Simaiya, Surjeet Dalal, Yogesh Kumar Sharma, Shilpi Tomar, Arshad Hashmi. "Secure WSN Architecture Utilizing Hybrid Encryption with DKM to Ensure Consistent IoV Communication", Wireless Personal Communications, 2024 Publication	1%
5	www.ijirset.com Internet Source	1%
6	docplayer.net Internet Source	1%
7	ijritcc.org Internet Source	1%
8	ijact.in Internet Source	1%

9	A. Brokalakis, A.P. Kakarountas, C.E. Goutis. "A high-throughput area efficient FPGA implementation of AES-128 encryption", IEEE Workshop on Signal Processing Systems Design and Implementation, 2005., 2005 Publication	<1 %
10	prr.hec.gov.pk Internet Source	<1 %
11	Debdeep Mukhopadhyay, Rajat Subhra Chakraborty. "Hardware Security - Design, Threats, and Safeguards", Chapman and Hall/CRC, 2019 Publication	<1 %
12	Chetna Sangwan. "VLSI Implementation of Advanced Encryption Standard", 2012 Second International Conference on Advanced Computing & Communication Technologies, 01/2012 Publication	<1 %
13	dokumen.pub Internet Source	<1 %
14	Hossein Kouzehzar, Meisam Nesary Moghadam, Pooya Torkzadeh. "A High Data Rate Pipelined Architecture of AES Encryption/Decryption in Storage Area Networks", Electrical Engineering (ICEE), Iranian Conference on, 2018 Publication	<1 %
15	Lavanya R, Karpagam M. "Enhancing the security of AES through small scale confusion operations for data communication", Microprocessors and Microsystems, 2020 Publication	<1 %

16	braincoke.fr Internet Source	<1%
17	"The 2021 International Conference on Smart Technologies and Systems for Internet of Things", Springer Science and Business Media LLC, 2023 Publication	<1%
18	Farashahi, Reza Rezaeian, Bahram Rashidi, and Sayed Masoud Sayedi. "FPGA based fast and high-throughput 2-slow retiming 128-bit AES encryption algorithm", Microelectronics Journal, 2014. Publication	<1%
19	Trang Hoang. "An Efficient FPGA Implementation of the Advanced Encryption Standard Algorithm", 2012 IEEE RIVF International Conference on Computing & Communication Technologies Research Innovation and Vision for the Future, 02/2012 Publication	<1%
20	cyberleninka.org Internet Source	<1%
21	eprint.iacr.org Internet Source	<1%
22	www.facweb.iitkgp.ernet.in Internet Source	<1%
23	"International Conference on Computer Networks and Communication Technologies", Springer Science and Business Media LLC, 2019 Publication	<1%

24	"Intelligent Data Engineering and Analytics", Springer Science and Business Media LLC, 2025 Publication	<1%
25	Simarpreet Singh Chawla, Nidhi Goel. "FPGA implementation of an 8-bit AES architecture: A pipelined and masked approach", 2015 Annual IEEE India Conference (INDICON), 2015 Publication	<1%
26	Simarpreet Singh Chawla, Nidhi Goel. "FPGA implementation of an 8-bit AES architecture: A rolled and masked S-Box approach", 2015 Annual IEEE India Conference (INDICON), 2015 Publication	<1%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		