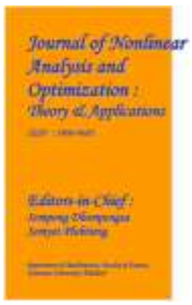
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 15, Issue. 1 : 2024
ISSN : **1906-9685**



ENHANCING HARDWARE SECURITY: OPTIMIZED 256-BIT AES IMPLEMENTATION FOR AREA AND POWER EFFICIENCY IN FPGA

[1] Yarragunta Divya, Department of Electronics and Communication Engineering, DVR & Dr.HS MIC College of Technology, Kanchikacherla , Andhra Pradesh, Email yarraguntadivya2002@gmail.com
[2] M. Nageswara RAO, Assistant Professor, Department of Electronics and Communication Engineering, DVR & Dr.HS MIC College of Technology, Kanchikacherla , Andhra Pradesh, Email nagesh.malisetti@gmail.com
[3] Badisa Upendra, Department of Electronics and Communication Engineering, DVR & Dr.HS MIC College of Technology, Kanchikacherla , Andhra Pradesh, Email upendrabadisa7275@gmail.com
[4] Banavathu Janaki, Department of Electronics and Communication Engineering, DVR & Dr.HS MIC College of Technology, Kanchikacherla , Andhra Pradesh, Email janakibanavathu26@gmail.com
[5] Chinthalapati Hemanth, Department of Electronics and Communication Engineering, DVR & Dr.HS MIC College of Technology, Kanchikacherla , Andhra Pradesh, Email chinthalapatihemanth@gmail.com

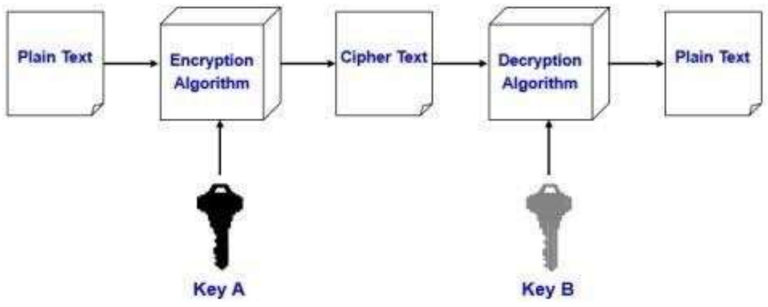
Abstract
Hardware security is paramount in various applications such as net banking, e-commerce, military operations, satellite communications, wireless communications, electronic gadgets, and digital image processing. Cryptography plays a crucial role in securing data by converting plain text into unintelligible text and vice versa. Three primary cryptographic techniques are widely used: Symmetric key cryptography, Hash functions, and Public key cryptography. Symmetric key algorithms like Advanced Encryption Standard (AES) and Data Encryption Standard utilize the same key for both encryption and decryption processes. They offer faster execution, ease of implementation, and require less processing power. This paper proposes an optimized 256-bit AES algorithm focusing on enhancing key schedule and sub bytes blocks for improved area and power efficiency. The proposed optimization leverages a novel approach where internal operations are performed using 32-bit operations, as opposed to 128-bit operations. This facilitates the reuse of hardware in a pipelined fashion, leading to area reduction through the efficient utilization of slice registers, slice LUTs, and LUT-FF Pairs. Consequently, the FPGA implementation exhibits a reduction in power consumption. Through experimentation utilizing a BASYS FPGA, the proposed implementation demonstrates enhanced throughput (Mbps), indicating its efficacy in real-world applications. This optimization not only enhances security but also contributes to improved resource utilization and power efficiency in hardware-based cryptographic systems.

Keywords: Hardware Security, Cryptography, Symmetric Key Algorithms, Advanced Encryption Standard (AES), Key Schedule Optimization, Sub Bytes Blocks, 256-bit AES, FPGA Implementation.

1 Introduction

Secure communication is vital in today's interconnected world, demanding meticulous attention to several key requirements. Firstly, confidentiality, or secrecy, must be ensured to safeguard sensitive information from prying eyes. Encryption techniques like the Advanced Encryption Standard (AES) play a pivotal role here, scrambling data into an unreadable format that only authorized parties can decipher. Secondly, authentication serves to verify the identities of both the sender and receiver, thwarting potential impersonation attempts. Public key cryptography enables secure identity verification without relying on shared secrets, bolstering trust in communication channels. Thirdly,

integrity guarantees that transmitted data remains unaltered during transit, accomplished through mechanisms like digital signatures and hash functions to detect and deter unauthorized modifications. Moreover, non-repudiation strengthens accountability by preventing individuals from disowning their actions or messages. This is facilitated by digital signatures, which uniquely tie identities to transmitted data, ensuring accountability. Finally, effective key management is imperative to safeguard cryptographic keys, the linchpin of secure communication. Proper key generation, distribution, and revocation protocols are essential to thwart potential security breaches. In sum, by meticulously addressing these requirements, secure communication systems can fortify data protection, establish trust among users, and uphold the integrity of digital transactions in an increasingly interconnected world.
Cipher: Cipher is a method for encrypting messages.



Encryption algorithms are standardized & publishedThe key which is an input to the algorithm is secret

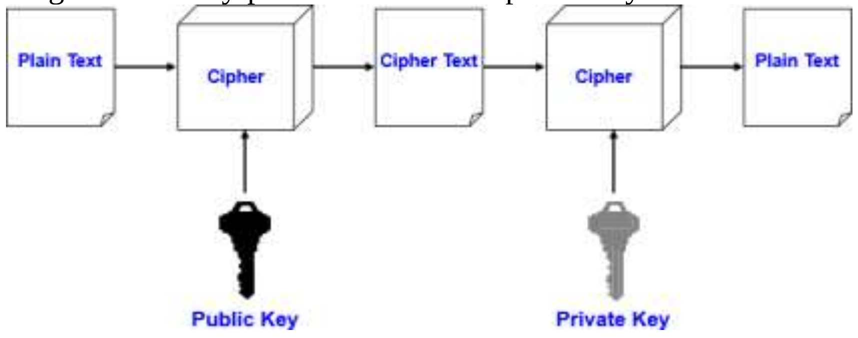
– Key is a string of numbers or characters
– If same key is used for encryption & decryption the algorithm is called symmetric
– If different keys are used for encryption & decryption the algorithm is calledasymmetric.

Uses a pair of keys for encryption

– Public key for encryption
– Private key for decryption

Messages encoded using public key can only be decoded by the private key

– Secret transmission of key for decryption is not required
– Every entity can generate a key pair and release its public key



Symmetric algorithms:

Algorithms in which the key for encryption and decryption are the same are SymmetricExample: Caesar Cipher
Types:

Block Ciphers

- Encrypt data one block at a time (typically 64 bits, or 128 bits)

- Used for a single messageStream Ciphers
- Encrypt data one bit or one byte at a time
- Used if data is a constant stream of information

In modern-day block ciphers, the role of substitution-boxes is to transform the plaintext data nonlinearly to generate cipher-text data with sufficient confusion. It has been well-confirmedthat the robustness and security of such block ciphers heavily based on the cryptographic strength of the underlying substitution-boxes. Reason being, they are the only components that are held responsible to bring required nonlinearity and complexity into the security system which can frustrate the attackers. Accordingly, a number of different concepts have been explored to construct strong S-boxes. To move forward with the same aim, a novel simple modular approach, the very first time, is investigated to construct nonlinear S-box in this paper. The new modular approach consists of three operations such as new transformation, modular inverses, and permutation. A number of highly nonlinear S-boxes can be easily constructed with slight changes in the novel transformation parameters. An example S-box is presented whose critical performance assessment against some benchmarking criterions such as high nonlinearity, absence of fixed points, fulfillment of SAC and BIC properties, low differentialuniformity and linear approximation probability and comparison with recent S-boxes demonstrate its upright cryptographic potentiality. In addition, an image encryption algorithmis also proposed wherein the generated S-box is applied to perform the pixels shuffling and substitution for strong statistical and differential encryption performance.

2Literature Survey

Rijmen et.al [1] The implementation of a combined memory-less S-Box and Inv S-Box in AES architecture represents a significant advancement, offering superior hardware efficiency compared to traditional LUT-based approaches. By consolidating both functionalities onto the same hardware, notable reductions in gate count, area footprint, and power consumption are achieved. This integration facilitates resource sharing, enhancing efficiency further. AES, widely applied in banking, digital video recorders, web servers, ATMs, and cellular phones, benefits from its hardware implementation, providing enhanced security and processing speed compared to software approaches. Notably, AES's departure from the Feistel structure contributes to its efficiency and versatility, making it a preferred choice for diverse encryption needs.

Lin et.al [2] The paper introduces compact architectures for AES MixColumn and its inverse, aimed at minimizing the area cost in AES implementations. Traditional hardware implementations of AES encounter challenges due to the resource-intensive MixColumn/Inverse MixColumn transformations, impacting critical path delay and throughput. The proposed designs leverage byte and bit-level decomposition, resulting in two architecture types that facilitate deeper resource sharing and rearrangement of output terms to better suit FPGA architectures at the bit level. Experimental validation on an FPGA platform demonstrates a 40% reduction in reconfigurable logic area compared to separate MixColumn implementations, along with a 9% decrease in path delay. These findings highlight the significant area cost reduction achievable with the proposed architectures, outperforming previous implementations.

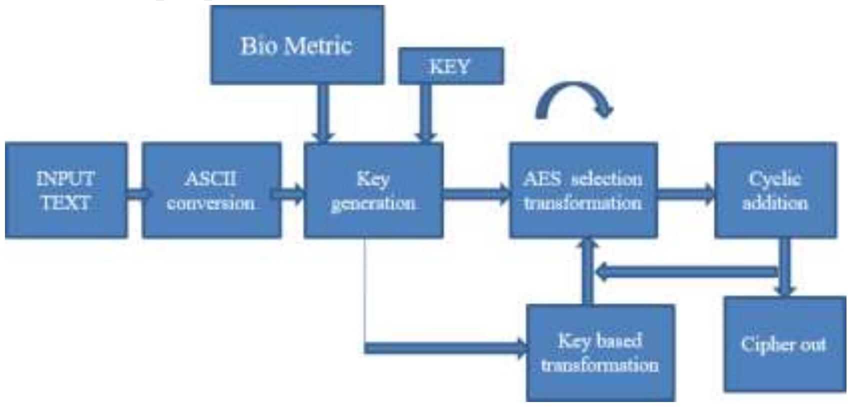
Zhang et.al [3] This paper proposes a pipelined architecture tailored for FPGA-based AES-128 encryption, optimizing Block RAM usage for storing S-box values and reducing critical delay by combining operations within a single round. With network speeds reaching gigabits per second, hardware-based implementations offer superior throughput and faster key generation compared to software-based approaches. Moreover, by encapsulating cryptographic processes and key generation within chips, hardware solutions enhance physical security. While FPGA-based implementations offer flexibility, ASIC-based solutions, despite high development costs and long cycles, also contribute to hardware-based encryption advancements.

Sivakumar et.al [4] This paper presents a high-speed, non-pipelined FPGA implementation of the AES-CCMP cipher for wireless LAN, focusing on the IEEE 802.11i standard. Utilizing Xilinx development tools and Virtex-It Pro FPGA circuits, the AES CCMP core is designed in Verilog 2001

to achieve both speed and security. Operating at 194/148MHz for encryption/decryption respectively, the implementation achieves throughput rates of 2.257 Gbits/sec and 1.722 Gbits/sec. By migrating to hardware from software, the design offers enhanced security and faster encryption speeds. The AES S-box, crucial for encryption, employs a 256-entry table consisting of multiplicative inverse and affine transformations over GF(2^8). The paper includes a comparative analysis with existing implementations to highlight the efficacy of the proposed design.
Bhargav et.al [5] This paper introduces a hardware implementation of the Advanced Encryption Standard (AES) algorithm using Xilinx Virtex-5 Field Programmable Gate Array (FPGA). With the increasing demand for data protection in computer networks, hardware implementations offer superior security compared to software solutions. The design focuses on optimizing performance by implementing key AES operations, such as Sub Byte, Inverse Sub Byte, Mix Column, and Inverse Mix Column, using Look-Up Tables (LUTs) and Read Only Memories (ROMs).
Deshpande et.al [6] This paper presents a hardware implementation of the AES Rijndael Encryption and Decryption Algorithm using Xilinx Virtex-7 FPGA. The design relies on pre-calculated look-up tables (LUTs) to simplify the architecture, resulting in high throughput and low latency. The implementation covers all three AES formats (128, 192, and 256 bits), utilizing Verilog-HDL for efficient encryption and decryption block designs. Synthesis on the Virtex-7 XC7VX690T chip optimizes speed, area, and power using the Xilinx ISE Design Suite-14.7 Tool. Power analysis conducted with Xilinx XPower Analyzer highlights the efficiency of the proposed architecture, showcasing favorable latency, throughput, speed/delay, area, and power characteristics.

3 Methodology

Fig 3 Pipelined structure of proposed method



The diagram of a key-based transformation process for biometric key generation and a multi-round AES cryptographic system. The process involves taking biometric data as input, converting it into ASCII text, and then generating a key from the ASCII text. The key is then used in a cyclic AES transformation and a key-based transformation. The final output is a cipher text.
1 Biometric Input: The process starts with biometric data, which can be a fingerprint, iris scan, or any other unique physical characteristic of a person.
2 ASCII Conversion: The biometric data is converted into a string of ASCII characters. ASCII (American Standard Code for Information Interchange) is a character encoding standard that assigns a unique number to every character. This allows the biometric data to be represented as a digital signal.
3 Key Generation: A key is generated from the ASCII text derived from the biometric data. The method for generating the key is not shown in the diagram, but it is likely to involve a cryptographic hash function. A cryptographic hash function is a mathematical function that takes an input of any size and produces a fixed-size output. The output, called a hash, is a unique fingerprint of the input data. Any change to the input data will result in a different hash value.
4 AES Selection: The key is then used in an AES transformation. AES (Advanced Encryption Standard) is a symmetric block cipher that is widely used for secure communication. A symmetric block cipher is a type of encryption algorithm that uses the same key for both encryption and

decryption. In the AES transformation, the key is used to select a specific AES algorithm from a set of possible algorithms.

5 Cyclic Transformation: The output of the AES transformation is then fed into a cyclic transformation. The nature of the cyclic transformation is not shown in the diagram, but it is likely to involve some form of permutation or substitution operation. A permutation is a mathematical operation that rearranges the order of a set of elements. A substitution is a mathematical operation that replaces one element with another.

6 Key-Based Transformation: The output of the cyclic transformation is then used in a key-based transformation. The key used in this transformation is likely to be the same key that was generated from the biometric data. The nature of the key-based transformation is not shown in the diagram, but it is likely to involve some form of mixing operation. A mixing operation is a cryptographic operation that combines two or more data streams in a way that makes it difficult to recover the original data streams.

7 Cipher Text: The final output of the process is a cipher text. The cipher text is unintelligible without the corresponding key.

This process is designed to be secure because the biometric data is never stored in its original form. Instead, it is converted into a key that is used to encrypt the data. Even if an attacker were to gain access to the cipher text, they would not be able to decrypt it without the key. The key is generated from the biometric data, which is unique to each individual. This means that even if an attacker were to obtain the key for one person, they would not be able to use it to decrypt the data of another person.

Results

RTL schematic

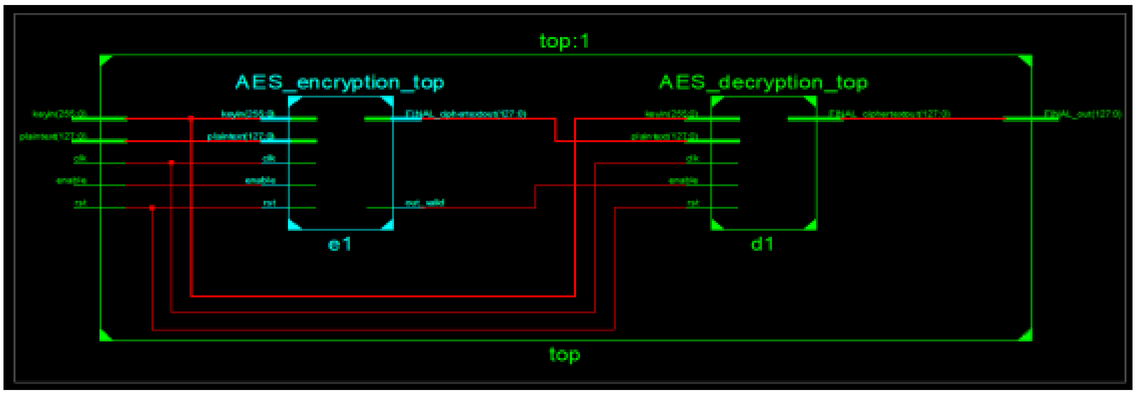
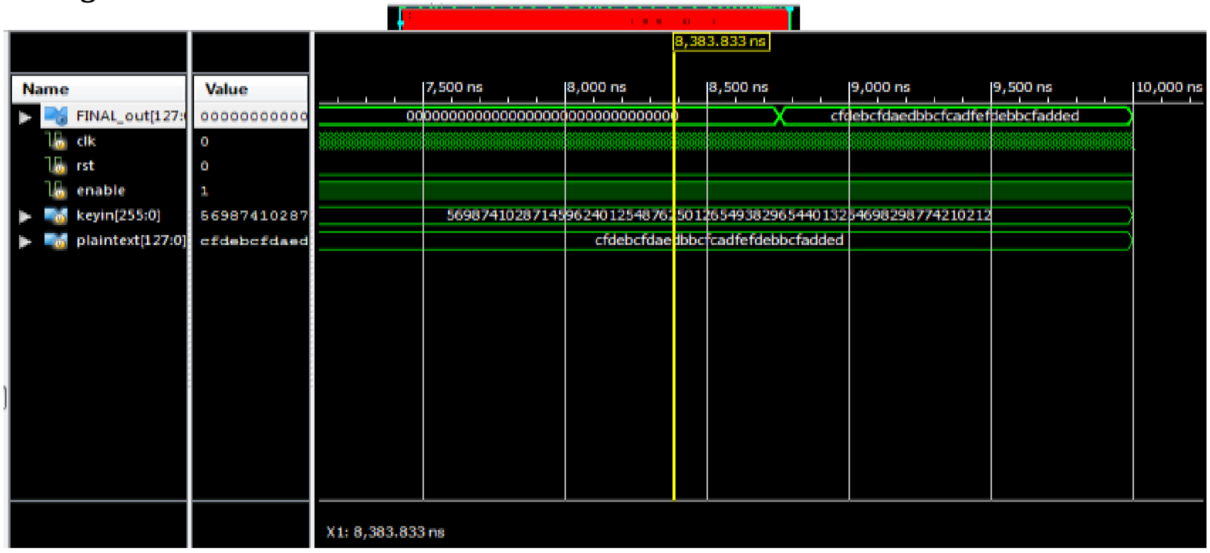
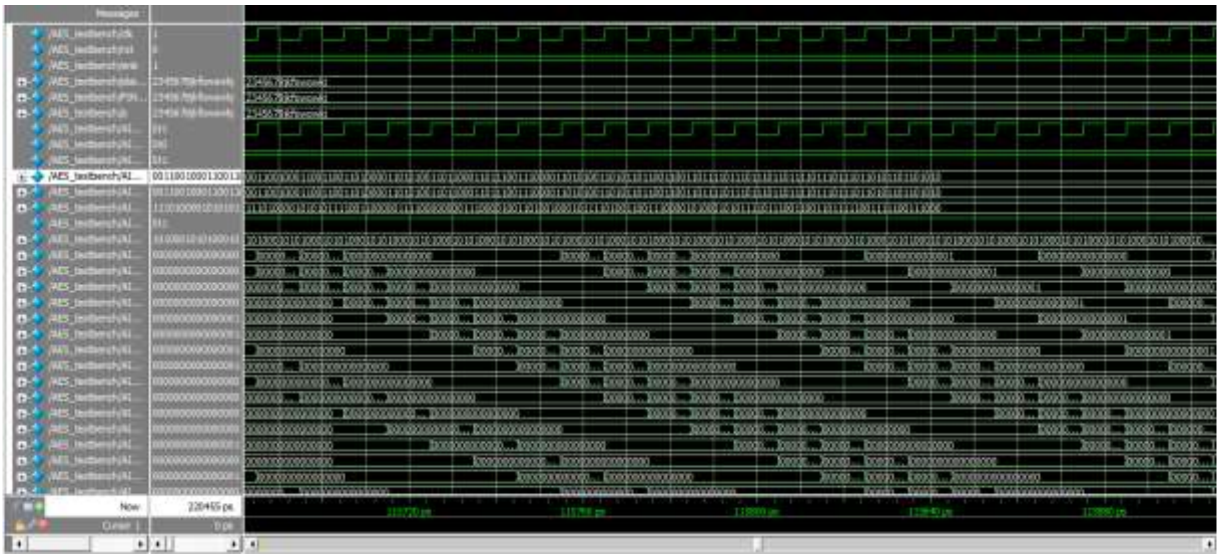


Fig 4 Technology Schematic

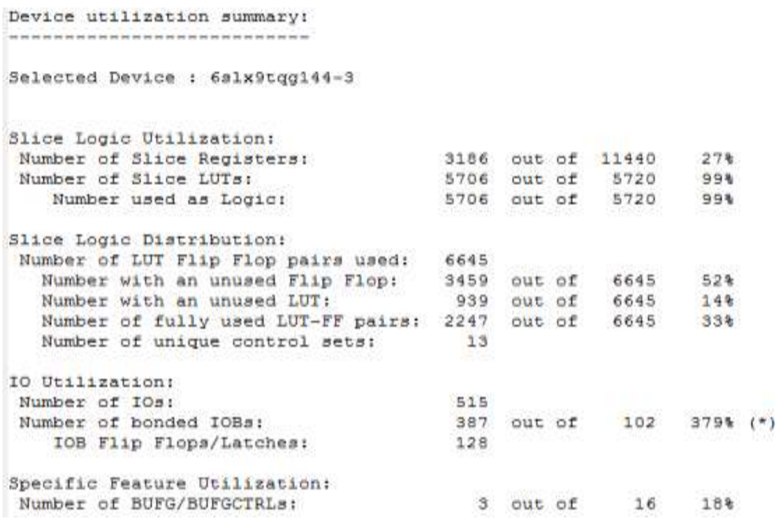
Fig 5 Simulation results



```
                                8,383.833 ns
Name              Value          |7,500 ns   |8,000 ns   |8,500 ns   |9,000 ns   |9,500 ns   |10,000 ns
FINAL_out[127:    00000000000    00000000000000000000000000000000  X  cfdebcfdaedbbcfcadfefdebbcfadded
clk               0
rst               0
enable            1
keyin[255:0]      56987410287    5698741028714596240125487625012654938296544013254698298774210212
plaintext[127:0]  cfdebcfdaed    cfdebcfdaedbbcfcadfefdebbcfadded

X1: 8,383.833 ns
```

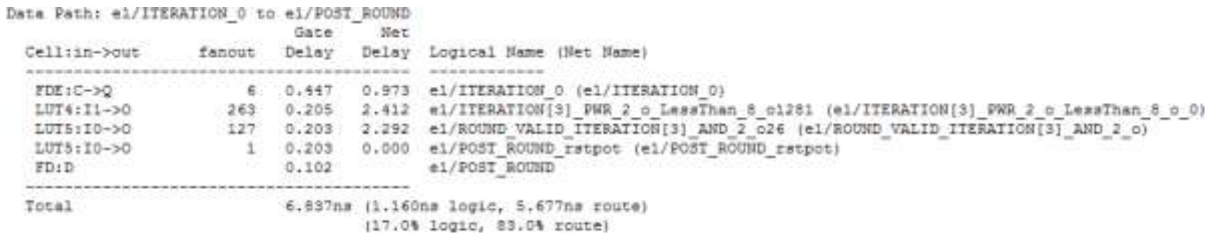


Area

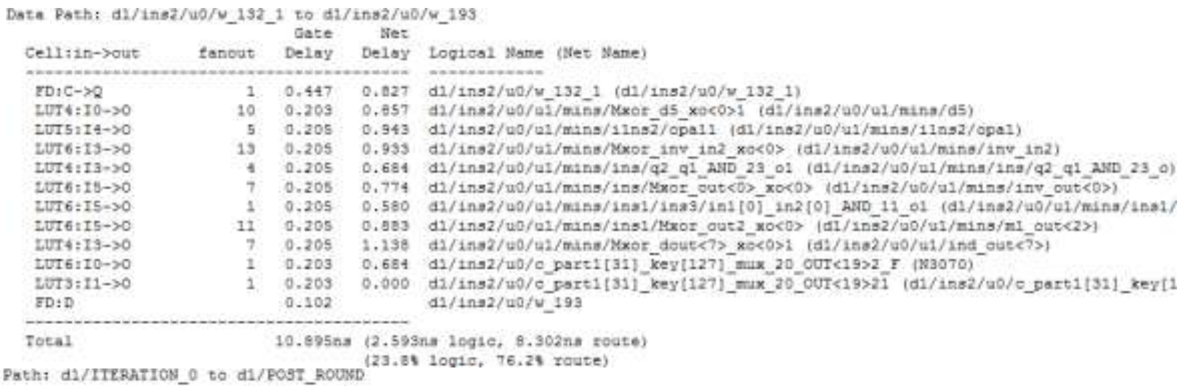


```
Device utilization summary:
---------------------------

Selected Device : 6slx9tqg144-3


Slice Logic Utilization:
 Number of Slice Registers:             3186  out of  11440    27%
 Number of Slice LUTs:                  5706  out of   5720    99%
    Number used as Logic:               5706  out of   5720    99%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    6645
   Number with an unused Flip Flop:     3459  out of   6645    52%
   Number with an unused LUT:            939  out of   6645    14%
   Number of fully used LUT-FF pairs:   2247  out of   6645    33%
   Number of unique control sets:         13

IO Utilization:
 Number of IOs:                          515
 Number of bonded IOBs:                  387  out of    102   379% (*)
    IOB Flip Flops/Latches:              128

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:                 3  out of     16    18%
```

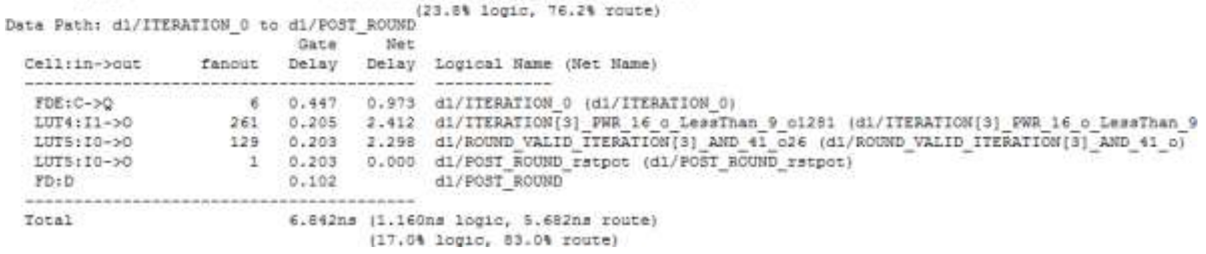
Delay



```
Data Path: e1/ITERATION_0 to e1/POST_ROUND
                                Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   FDE:C->Q              6   0.447   0.973  e1/ITERATION_0 (e1/ITERATION_0)
   LUT4:I1->O          263   0.205   2.412  e1/ITERATION[3]_PWR_2_o_LessThan_8_o1281 (e1/ITERATION[3]_PWR_2_o_LessThan_8_o_0)
   LUT5:I0->O          127   0.203   2.292  e1/ROUND_VALID_ITERATION[3]_AND_2_o26 (e1/ROUND_VALID_ITERATION[3]_AND_2_o)
   LUT5:I0->O            1   0.203   0.000  e1/POST_ROUND_rstpot (e1/POST_ROUND_rstpot)
   FD:D                      0.102          e1/POST_ROUND
  ----------------------------------------
  Total                      6.837ns (1.160ns logic, 5.677ns route)
                                     (17.0% logic, 83.0% route)
```



```
Data Path: d1/ins2/u0/w_132_1 to d1/ins2/u0/w_193
                                Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   FD:C->Q               1   0.447   0.827  d1/ins2/u0/w_132_1 (d1/ins2/u0/w_132_1)
   LUT4:I0->O           10   0.203   0.857  d1/ins2/u0/u1/mins/Mxor_d5_xo<0>1 (d1/ins2/u0/u1/mins/d5)
   LUT5:I4->O            5   0.205   0.943  d1/ins2/u0/u1/mins/i1ns2/opal1 (d1/ins2/u0/u1/mins/i1ns2/opal)
   LUT6:I3->O           13   0.205   0.933  d1/ins2/u0/u1/mins/Mxor_inv_in2_xo<0> (d1/ins2/u0/u1/mins/inv_in2)
   LUT4:I3->O            4   0.205   0.684  d1/ins2/u0/u1/mins/ins/q2_q1_AND_23_o1 (d1/ins2/u0/u1/mins/ins/q2_q1_AND_23_o)
   LUT6:I5->O            7   0.205   0.774  d1/ins2/u0/u1/mins/ins/Mxor_out<0>_xo<0> (d1/ins2/u0/u1/mins/inv_out<0>)
   LUT6:I5->O            1   0.205   0.580  d1/ins2/u0/u1/mins/ins1/ins3/in1[0]_in2[0]_AND_11_o1 (d1/ins2/u0/u1/mins/ins1/
   LUT6:I5->O           11   0.205   0.883  d1/ins2/u0/u1/mins/ins1/Mxor_out2_xo<0> (d1/ins2/u0/u1/mins/m1_out<2>)
   LUT4:I3->O            7   0.205   1.138  d1/ins2/u0/u1/mins/Mxor_dout<7>_xo<0>1 (d1/ins2/u0/u1/ind_out<7>)
   LUT6:I0->O            1   0.203   0.684  d1/ins2/u0/c_part1[31]_key[127]_mux_20_OUT<19>2_F (N3070)
   LUT3:I1->O            1   0.203   0.000  d1/ins2/u0/c_part1[31]_key[127]_mux_20_OUT<19>21 (d1/ins2/u0/c_part1[31]_key[1
   FD:D                      0.102          d1/ins2/u0/w_193
  ----------------------------------------
  Total                     10.895ns (2.593ns logic, 8.302ns route)
                                     (23.8% logic, 76.2% route)
```



```
Data Path: d1/ITERATION_0 to d1/POST_ROUND
                                Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   FDE:C->Q              6   0.447   0.973  d1/ITERATION_0 (d1/ITERATION_0)
   LUT4:I1->O          261   0.205   2.412  d1/ITERATION[3]_PWR_16_o_LessThan_9_o1281 (d1/ITERATION[3]_PWR_16_o_LessThan_9
   LUT5:I0->O          129   0.203   2.298  d1/ROUND_VALID_ITERATION[3]_AND_41_o26 (d1/ROUND_VALID_ITERATION[3]_AND_41_o)
   LUT5:I0->O            1   0.203   0.000  d1/POST_ROUND_rstpot (d1/POST_ROUND_rstpot)
   FD:D                      0.102          d1/POST_ROUND
  ----------------------------------------
  Total                      6.842ns (1.160ns logic, 5.682ns route)
                                     (17.0% logic, 83.0% route)
```

Evaluation table for Area, Delay:

	Area (LUT's)	**Delay (ns)**
Top	5706	24.574

Evaluation Table:

	Area (LUT's)	**Delay (ns)**
TOP	**5286**	**14.377**

Conclusion

In conclusion, our project introduces a highly optimized 256-bit AES algorithm tailored for 128-bit data, focusing on key schedule and SubBytes blocks to achieve efficiency in terms of area and power. Key optimizations include reusing the S-box block and leveraging 32-bit SubBytes and MixColumn blocks for 128-bit data. The proposed method is versatile, accommodating key sizes of 128, 196, and

256 bits, as well as word sizes of 16, 32, and 64 bits. Additionally, we integrated multi-level confusion matrices to strengthen key selection and introduced a practical implementation of AES with biometric-based key extraction, enhancing security compared to static keys. Functionally verified using Xilinx 14.7, our proposed AES design demonstrates reduced area and delay compared to existing approaches. Overall, our project offers a comprehensive and efficient solution for AES encryption and decryption, showcasing improvements in both performance and security.

Feature Scope

The feature scope of the project encompasses the development of a highly optimized 256-bit AES algorithm tailored for encrypting 128-bit data, with a focus on optimizing the key schedule and SubBytes blocks to enhance efficiency in terms of area and power consumption. The project aims to achieve this optimization by reusing the S-box block, accommodating key sizes of 128, 196, and 256 bits, as well as word sizes of 16, 32, and 64 bits. Additionally, the implementation includes enhanced security measures such as multi-level confusion matrices for key strengthening and introduces practical biometric-based key extraction methods. Functional verification using Xilinx 14.7 tools ensures reliability, while performance comparison against existing implementations demonstrates improvements in area usage and delay. Overall, the project aims to deliver a comprehensive and efficient AES encryption and decryption solution addressing both performance and security requirements.

References

[1] Rijmen, V., & Daemen, J. (2001). Advanced encryption standard. Proceedings of federal information processing standards publications, national institute of standards and technology, 19, 22..

[2] Lin, T. F., Su, C. P., Huang, C. T., & Wu, C. W. (2002, August). A high-throughput low-cost AES cipher chip. In Proceedings. IEEE Asia-Pacific Conference on ASIC, (pp. 85-88). IEEE.

[3] Zhang, X., & Parhi, K. K. (2004). High-speed VLSI architectures for the AES algorithm. IEEE transactions on very large scale integration (VLSI) systems, 12(9), 957-967

[4] Sivakumar, C., & Velmurugan, A. (2007, February). High speed VLSI design CCMP AES cipher for WLAN (IEEE 802.11 i). In 2007 International Conference on Signal Processing, Communications and Networking (pp. 398-403). IEEE.

[5] Bhargav, S., Chen, L., Majumdar, A., & Ramudit, S. (2008). 128-bit AES decryption. Project report, CSEE.

[6] Deshpande, A. M., Deshpande, M. S., & Kayatanavar, D. N. (2009, June). FPGA implementation of AES encryption and decryption. In 2009 international conference on control, automation, communication and energy conservation (pp. 1-6). IEEE.

[7] Iyer, N. C., Anandmohan, P. V., & Poornaiah, D. V. (2010, July). Mix/InvMixColumn decomposition and resource sharing in AES. In 2010 5th International Conference on Industrial and Information Systems (pp. 166-171). IEEE.

[8] Zhang, Y., & Wang, X. (2010, December). Pipelined implementation of AES encryption based on FPGA. In 2010 IEEE International Conference on Information Theory and Information Security (pp. 170-173). IEEE.

[9] Borkar, A. M., Kshirsagar, R. V., & Vyawahare, M. V. (2011, April). FPGA implementation of AES algorithm. In 2011 3rd International Conference on Electronics Computer Technology (Vol. 3, pp. 401-405). IEEE.

[10] Wang, W., Chen, J., & Xu, F. (2012, May). An implementation of AES algorithm Based on FPGA. In 2012 9th International conference on fuzzy systems and knowledge discovery (pp. 1615-1617). IEEE.