ICCSE'21 International Conference on Contemporary approach on revolutionary Science and Engineering, April 9-10, 2021 ISBN: 978-81-910765-1-6

ESTIMATION OF OPTIMIZED PARAMETERS IN MIXED COLUMN ARCHITECTURE FOR ADVANCED ENCRYPTION STANDARD CRYPTOGRAPHY

Nandhini M
Associate Professor: Dept of ECE
Indra Ganesan College of Engineering Trichy, Tamil Nadu
smnandhini123@gmail.com
Kokila D
Assistant Professor: Dept of ECE
Indra Ganesan College of Engineering, Trichy, Tamil Nadu
ksathiya1882019@gmail.com
Vel Rajeswari M R
Indra Ganesan College of Engineering, Trichy,Tamil Nadu
m.r.raajee1996@gmail.com

Abstract— Advanced Encryption Standard (AES) is a specification for electronic data encryption. This standard has become one of the most widely used encryption method and has been implemented in both software and hardware. AES has excellent resistance against linear and differential cryptanalysis. Although the standard itself is algorithmically secure, based on the implementation, it can be vulnerable to attackers through side channels. In order to achieve the same, this paper discusses significant and novel modifications to the existing hardware architecture of the mix column step of the AES algorithm. This paper presents an efficient hardware based 128-bit AES design using a Proposed Mixed Column Operation. This Mixed Column design is implemented in FPGA XC3S 200 TQ-144 using Verilog HDL and simulated by Modelsim 6.4 c and Synthesized by Xilinx tool.

Keywords— : Mixed Column, Advanced Encryption Standard (AES), Data encryption

I. INTRODUCTION

Cryptography, often called encryption, is the practice of creating and using a cryptosystem or cipher to prevent all but the intended recipient(s) from reading or using the information or application encrypted. A cryptosystem is a technique used to encode a message. The recipient can view the encrypted message only by decoding it with the correct algorithm and keys. Cryptography is used primarily for communicating sensitive material across computer networks. The process of encryption takes a clear-text document and applies a key and a mathematical algorithm to it, converting it into crypto-text. In crypto-text, the document is unreadable unless the reader possesses the key that can undo the encryption. The AES encryption algorithm is a block cipher that uses an encryption key and a several rounds of encryption. A block cipher is an encryption algorithm that works on a single block of data at a time. In the case of standard AES encryption the block is 128 bits, or 16 bytes, in length. The term "rounds" refers to the way in which the encryption algorithm mixes the data re-encrypting it ten to fourteen times depending on the length of the key. AES encryption uses a single key as a part of the encryption process. The key can be 128 bits (16 bytes), 192 bits (24 bytes), or 256 bits (32 bytes) in length. The term 128-bit encryption refers to the use of a128-bit encryption key. With AES both the encryption and the decryption are performed using the same key. This is called a symmetric encryption algorithm. Encryption algorithms that use two different keys, a public and a private key, are called asymmetric encryption algorithms.

There are several techniques which can greatly frustrate side channel attacks. 1) Avoid use of arrays. Compute values of SBOX and RCon to avoid timing attacks. 2) Design algorithms and devices to work with constant time intervals. 3) Use same memory throughout, remember, cache is faster than DRAM. 4) Compute Key Expansion on the fly. Don't compute the Key Expansion and then reference it from memory. 5) Utilize pipelining to stabilize CPU power consumption. 6) Use specialized chips whenever possible, right now they are significantly faster than CPU's and require extremely expensive equipment for side channel attack measurements.

II LITERATURE SURVEY

Although cryptosystem designers frequently assume that secret parameters will be manipulated in closed reliable computing environments, Kocher et al. reported in 1998 that microchips leak information correlated with the data handled and introduced a new kind of attacks which were radically different from software and algorithmic attacks. These attacks use leaking or side-channel information, like power consumption data, electromagnetic emanations or

 ISBN: 978-81-910765-1-6

computing time to recover the secret key. While FPGAs are becoming increasingly popular for cryptographic applications, there are only a few articles that assess their vulnerability to such attacks. This paper describes the principles of differential power analysis (DPA) attack and also illustrates a practical and successful implementation of this attack against an FPGA implementation of the advanced encryption standard (AES) algorithm. The results obtained in this work clearly demonstrate that DPA is a serious threat against realization of encryption algorithms on SRAM-based FPGAs without effective countermeasures.

This article examines vulnerabilities to power analysis attacks between software and hardware implementations of cryptographic algorithms. A simulation-based experimental environment is built to acquire power data, and single-bit differential power analysis (DPA), and multi-bit DPA and correlation power analysis (CPA) attacks are conducted on two implementations respectively. The experimental results show that the hardware implementation has less data-dependent power leakages to resist power attacks. Furthermore, an improved DPA approach is proposed. It adopts hamming distance of intermediate results as power model and arranges plaintext inputs to differentiate power traces to the maximal probability. Compared with the original power attacks, our improved DPA performs a successful attack on AES hardware implementations with acceptable power measurements and fewer computations.

III EXISTING SYSTEM

The popular method to perform the mix column multiplication is by utilizing pre calculated multiplication look up tables (LUT). They created two LUT's a logarithmic L table and an exponential E Table, which can be used together to find the product of two numbers in the Galois field. Since the values are pre-computed, this method has an upper edge with respect to calculation time in comparison with other methods. However, in an area-on-chip perspective the method proves to be futile due to its enormous memory requirement overhead.

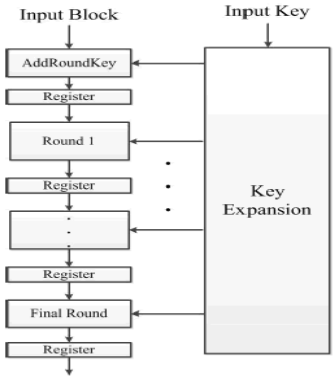
DISADVANTAGE

- Complex Key Generation Process
- Manual Key Process
- A conventional masked AES engine requires a large lookup table
- Complexed Design

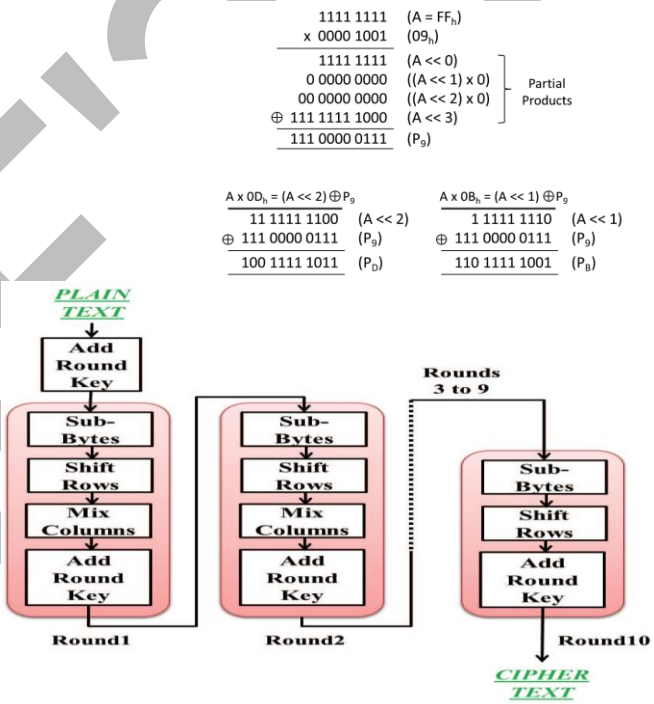
IV PROPOSED METHOD

The main aim of the work carried out in this paper is to improvise the speed efficiency of the AES algorithm and also to minimize the on-chip area occupied as well. In order to achieve the same, two major optimizations were identified and are elaborated below. The first deals with the optimization of the multiplication unit of the mix column step whilst the second method addresses improving the efficiency of the stage which reduces the multiplier output to Galois field – 2^8. The AES implementation consists of the AES core with optimized Mix column and a Key expansion for Encryption & Decryption.

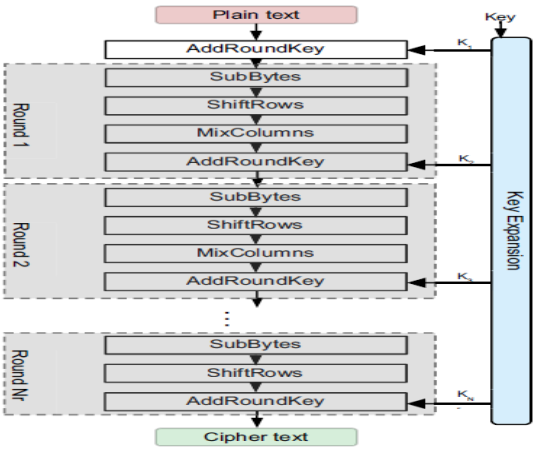
A.BLOCKDIAGRAM



B PROPOSED SYSTEM PROCESS



C ALGORITHM



Cryptographic architectures provide protection for sensitive and smart infrastructures such as secure healthcare, smart grid, fabric, and home. Cryptography is closely related to the disciplines of cryptology and

 ISBN: 978-81-910765-1-6

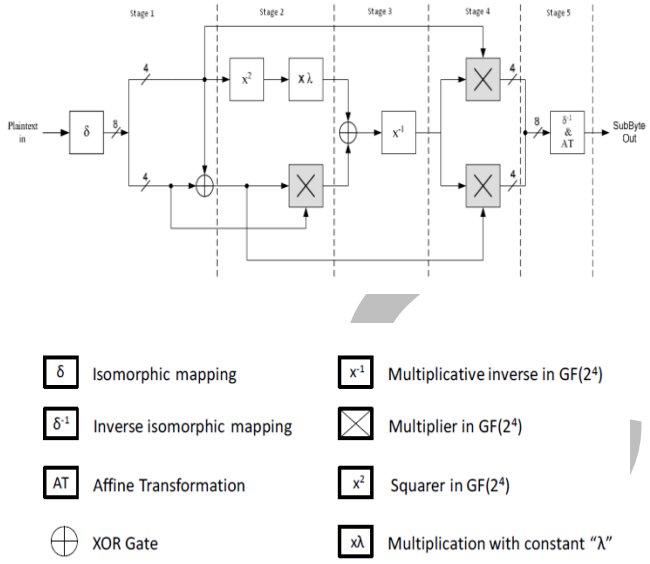
cryptanalysis. Cryptography includes techniques such as microdots, merging words with images, and other ways to hide information in storage or transit. However, in today's computer-centric world, cryptography is most often associated with scrambling plaintext (ordinary text, sometimes referred to as clear text) into cipher text (a process called encryption), then back again (known as decryption). Individuals who practice this field are known as cryptographers. Extensive research has been done for detecting such faults in the cryptographic algorithms such as elliptic curve cryptography and the Advanced Encryption Standard. Design for reliability and fault immunity ensures that with the presence of faults, reliability is provided for the aforementioned sensitive cryptographic architectures.

D MODULE DESCRIPTION

Substitution Box (S- Box)

The Sub Bytes operation is a nonlinear byte substitution. Each byte from the input state is replaced by another byte according to the substitution box (called the S-box). The S-box is computed based on a multiplicative inverse in the finite field GF (2^8) and a bitwise affine transformation.

In this module the implementation of the composite field S-BOX is accomplished using combinational logic circuits rather than using pre-stored S-BOX values.

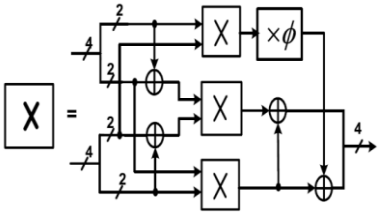


ADDITION IN GF (2^4)

Addition of 2 elements in Galois Field can be translated to simple bitwise XOR operation Addition of 2 elements in Galois Field can be translated to simple bitwise XOR operation.

GF (2^4) MULTIPLIER

Sub Bytes is a nonlinear transformation that uses 16 byte substitution tables (S-Boxes). An S-Box is the multiplicative inverse of a Galois field GF (2^4) followed by an affine transformation**.** Although two Galois Fields of the same order are isomorphic, the complexity of the field operations may heavily depend on the representations of the field elements. Three multipliers in GF (2^4) are required as a part of finding the multiplicative inverse in GF (2^8). Fig. shows the GF (2^4) multiplier circuit. As can be seen from the figure the GF (2^4) multipliers consist of 3 GF (2^2) multipliers with 4 XOR Gates and with constant multiplier θ. This constant multiplier which has 2 bits input extracts the lower bit output as the higher bit input, while the higher output bit will be the result of XOR operation between the 2 input bits. Full derivation of this multiplier circuit can be found.

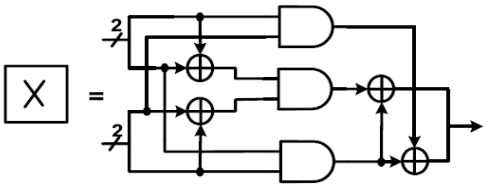


GF (2^4) Multiplier

GF (2^2) MULTIPLIER

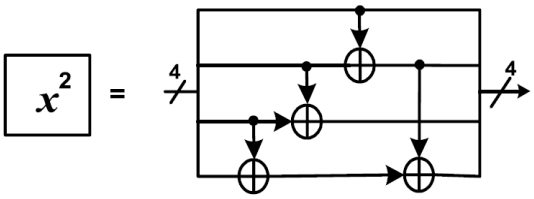
While each finite field is itself not infinite, there are infinitely many different finite fields; their number of elements (which is also called cardinality) is necessarily of the form p^n where p is a prime number and n is a positive integer.

IMPLEMENTATION OF THE GF (2^2) MULTIPLIER



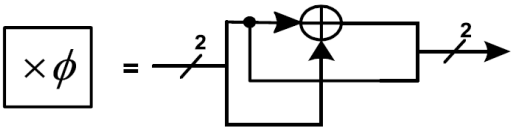
GF (2^4) SQUARER

It consists of bitwise xor operation. A bitwise operation operates on one or more bit atterns or binary numerals at the level of their individual bits. It is a fast, primitive action directly supported by the processor, and is used to manipulate values for comparisons and calculations. On simple low-cost processors, typically, bitwise operations are substantially faster than division, several times faster than multiplication, and sometimes significantly faster than addition. While modern high-performance processors usually perform addition and multiplication as fast as bitwise operations, the latter may still be optimal for overall power/performance due to lower resource use.



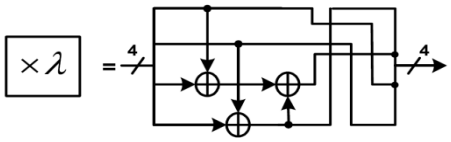
Hardware diagram for Squarer in GF (2^4)

CONSTANT MULTIPLIER (X)



HARDWARE IMPLEMENTATION OF MULTIPLICATION WITH CONSTANT Φ

CONSTANT MULTIPLIER (Lambda)

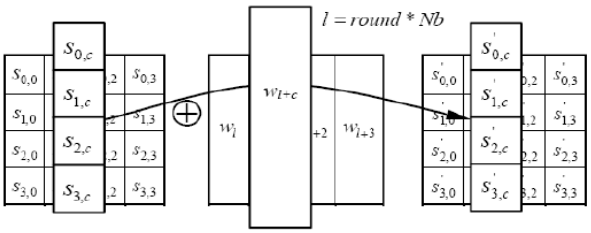


 ISBN: 978-81-910765-1-6

HARDWARE DIAGRAM FOR MULTIPLICATION WITH CONSTANT

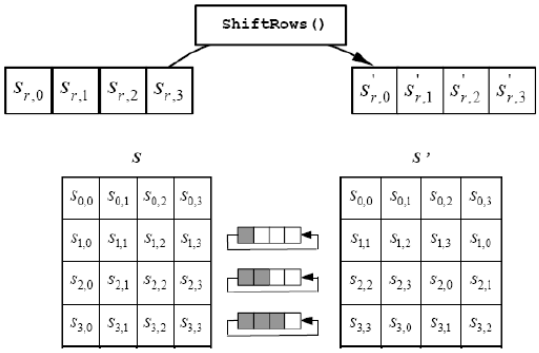
ADDROUNDKEY TRANSFORMATION

In the AddRoundKey transformation, a round key is added to the state by Bitwise Exclusive-OR (XOR) operation. Figure below illustrates the AddRoundKey. This transformation is the same for both encryption and decryption.



SHIFT ROWS TRANSFORMATION (INV SHIFT ROWS)

Shift Rows is a cyclic shift operation in each row of the State. In this operation, the bytes in the first row of the state do not change. The second, third, and fourth rows shift cyclically to the left one byte, two bytes, three bytes, respectively, as illustrated in Figure. The reverse process, inv Shift Row, operates in reverse order to Shift Rows.



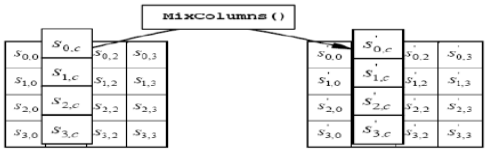
MIX COLUMN TRANSFORMATION (INV MIX COLUMN)

The Mix Column transformation is performed independently on the state Column-by-column. Each column is considered as four term polynomial over GF (2^8) and multiplied by $a(x)$ modulo $(x4 + 1)$ where $a(x) = \{03\}x3 + \{01\}x2 + \{01\}x + \{02\}$

This transformation can be expressed in matrix form

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} \{02\} & \{03\} & \{01\} & \{01\} \\ \{01\} & \{02\} & \{03\} & \{01\} \\ \{01\} & \{01\} & \{02\} & \{03\} \\ \{03\} & \{01\} & \{01\} & \{02\} \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

For *invMixColumn()*, replace $a(x) = \{0E\}x^3 + \{09\}x^2 + \{0D\}x + \{0B\}$.



as

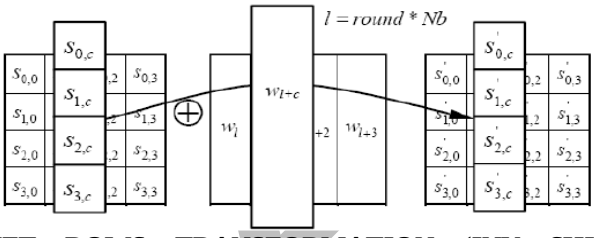
TRANSFORMATION MATRIX & INVERSE TRANSFORMATION MATRIX

A transformation matrix (M) transforms the elements in the binary field to the composite field GF $((2^3)^3)$. Then, the operations are done in composite fields to achieve the inverse which is then retransformed to binary field using an inverse transformation matrix (M^{-1}). Eventually, the two most and least significant bits are discarded to get to the uneven structure of the substitution box.

$$\beta \times q = \begin{pmatrix} 1&0&1&0&0&0&0&0 \\ 1&1&0&1&1&1&1&0 \\ 1&0&1&0&1&1&0&0 \\ 1&0&1&0&1&1&1&0 \\ 1&1&0&0&0&1&1&0 \\ 1&0&0&1&1&1&1&0 \\ 0&1&0&1&0&0&1&0 \\ 0&1&0&0&0&0&1&1 \end{pmatrix} \times \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix} \quad \delta^{-1} \times q = \begin{pmatrix} 1&1&1&0&0&0&1&0 \\ 0&1&0&0&0&1&0&0 \\ 0&1&1&0&0&0&1&0 \\ 0&1&1&1&0&1&1&0 \\ 0&0&1&1&1&1&1&0 \\ 1&0&0&1&1&1&1&0 \\ 0&0&1&1&0&0&0&0 \\ 0&1&1&1&0&1&0&1 \end{pmatrix} \times \begin{pmatrix} q_7 \\ q_6 \\ q_5 \\ q_4 \\ q_3 \\ q_2 \\ q_1 \\ q_0 \end{pmatrix}$$

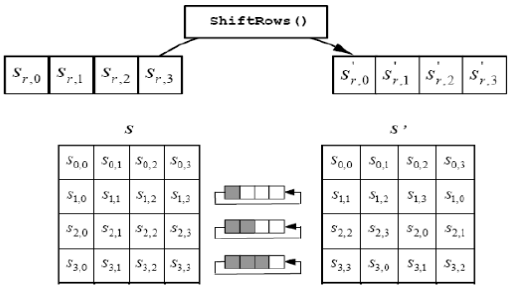
ADDROUNDKEY TRANSFORMATION

In the Add Round Key transformation, a round key is added to the state by Bitwise Exclusive-OR (XOR) operation. Figure below illustrates the AddRoundKey. This transformation is the same for both encryption and decryption.



SHIFT ROWS TRANSFORMATION (INV SHIFT ROWS)

Shift Rows is a cyclic shift operation in each row of the State. In this operation, the bytes in the first row of the state do not change. The second, third, and fourth rows shift cyclically to the left one byte, two bytes, three bytes, respectively, as illustrated in Figure. The reverse process, inv Shift Row, operates in reverse order to Shift Rows.



The state register is organized so that after loading the input data and the input key, the encryption is done by shifting the data 32 b in each clock cycle. The state register consists of sixteen 8-b registers which are further divided into four 4-stage shift registers. AES standard specifies that Shift Row is a permutation operation on the rows of the state matrix, while Mix Colum is an operation on the columns.

MIX COLUMN TRANSFORMATION (INV MIX COLUMN)

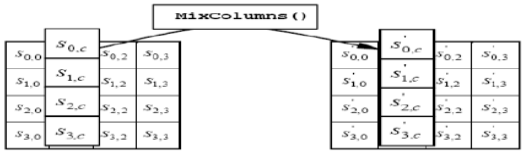
The Mix Column transformation is performed independently on the state Column-by-column. Each column is considered as four term polynomial over GF (2^8) and multiplied by $a(x)$ modulo $(x4 + 1)$ where $a(x) = \{03\}x3 + \{01\}x2 + \{01\}x + \{02\}$

This transformation can be expressed in matrix form as

 ISBN: 978-81-910765-1-6

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} \{02\} & \{03\} & \{01\} & \{01\} \\ \{01\} & \{02\} & \{03\} & \{01\} \\ \{01\} & \{01\} & \{02\} & \{03\} \\ \{03\} & \{01\} & \{01\} & \{02\} \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

For *invMixColumn()*, replace $a(x) = \{0E\}x^3 + \{09\}x^2 + \{0D\}x + \{0B\}$.



ROTCON, ROT WORD

These S-boxes are enabled for the first cycle of a round for 128- and 256-b keys and every six cycles for 192-b key. After that, they remain inactive. This leads to 30% reduction in power consumption of the S-boxes in the key expansion. RotWord step can be removed because it exchanges the position of bytes in a 32-b signal. The XOR of the RCON with the output of the S-boxes after RotWord is minimized by XORing only the necessary bits. The clock gating technique is also applied in the key expansion to save power consumption. During the idle state, the key register and the S-boxes will not create any activities.

AES ALGORITHM

AES is a symmetric block cipher. This means that it uses the same key for both encryption and decryption. The algorithm Rijndael allows for a variety of block and key sizes and not just the 64 and 56 bits of DES' block and key size. The block and key can in fact be chosen independently from 128, 160, 192, 224, 256 bits and need not be the same. However, the AES standard states that the algorithm can only accept a block size of 128 bits and a choice of three keys - 128, 192, 256 bits. Depending on which version is used, the name of the standard is modified to AES-128, AES-192 or AES- 256 respectively. As well as these differences AES differs from DES in that it is not a feistel structure. Pipelined decryption comprises the inverse of the transformations that encryption uses, performed in reverse order. Decryption commences with the inverse of the final round, followed by the inverses of the rounds, and finishes with the initial data/key addition, which is its own inverse. AES performs all its computations on bytes rather than bits. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key. A number of AES parameters depend on the key length. For example, if the key size used is 128 then the number of rounds is 10 whereas it is 12 and 14 for 192 and 256 bits respectively. At present the most common key size likely to be used is the 128 bit key. This description of the AES algorithm therefore describes this particular implementation.

Rijndael was designed to have the following characteristics:

• Resistance against all known attacks.

• Speed and code compactness on a wide range of platforms.

• Design Simplicity

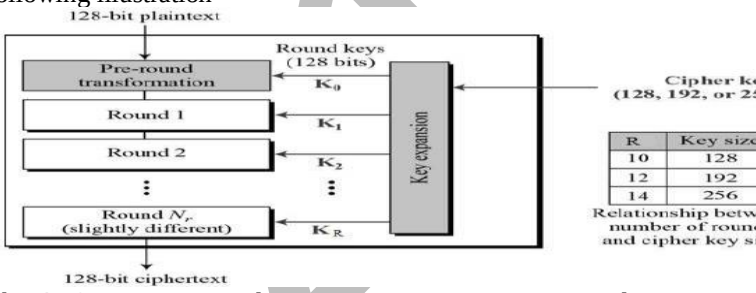
The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of it's counterpart in the encryption algorithm. The four stages are as follows:

1. Substitute bytes
2. Shift rows
3. Mix Columns
4. Add Round Key

The tenth round simply leaves out the Mix Columns stage. The first nine rounds of the decryption algorithm consist of the following:

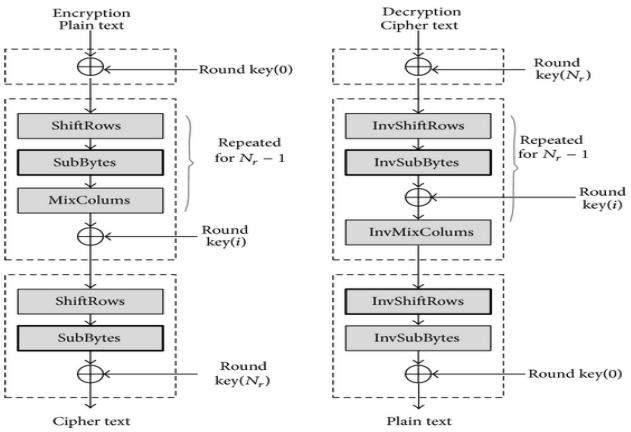
1. Inverse Shift rows
2. Inverse Substitute bytes
3. Inverse Add Round Key
4. Inverse Mix Columns

The schematic of AES structure is given in the following illustration



The AES Encryption and Decryption process is mentioned in bellow Diagram. The process of decryption of an AES cipher text is similar to the encryption process in the reverse order. Each round consists of the four processes

- Add round key
- Mix columns
- Shift rows
- Byte substitution

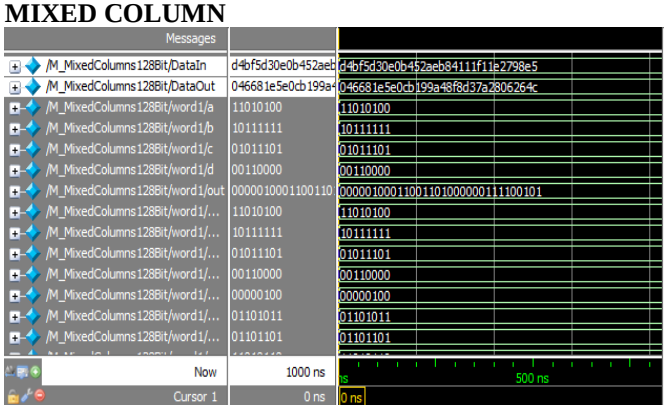


V SIMULATION RESULTS

Its performed simulation using ModelSim 6.4a. . The RTL schematic view has been viewed by Xilinx Design 13.2. The proposed architecture uses data encryption and decryption , Security system overhead that the existing system technique. Overall, the design presented remarkably improve area, power, delay.

 ISBN: 978-81-910765-1-6

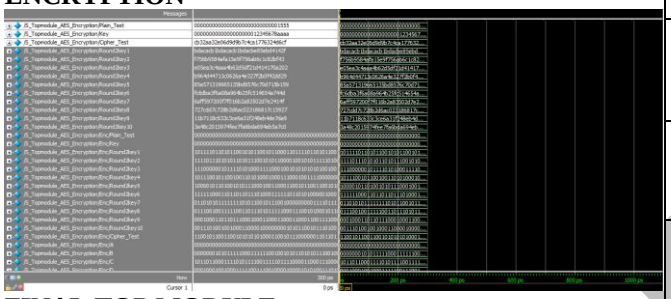
MIXED COLUMN



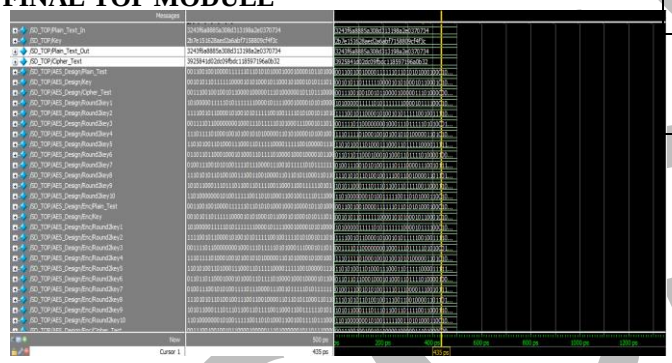
Mix column

Here, figure shows the output of mixed column 128 bit / data in and mixed coloumn 128 bit/ data out

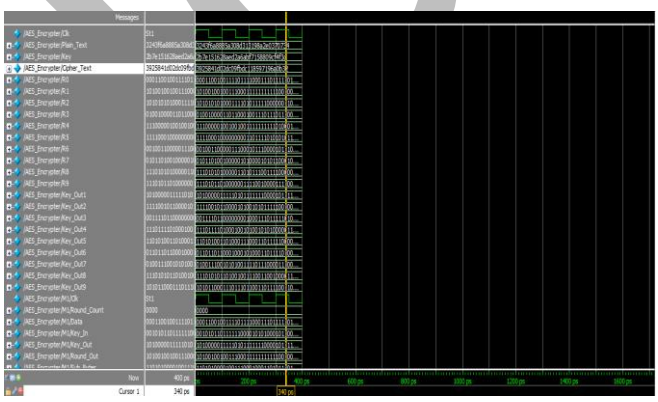
ENCRYPTION



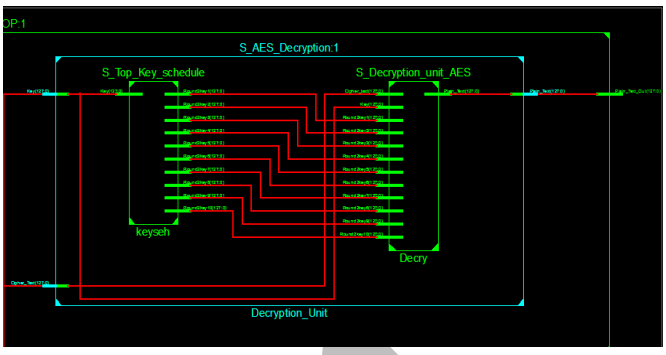
FINAL TOP MODULE



AES Encrypter



RTL SCHEMATIC VIEW

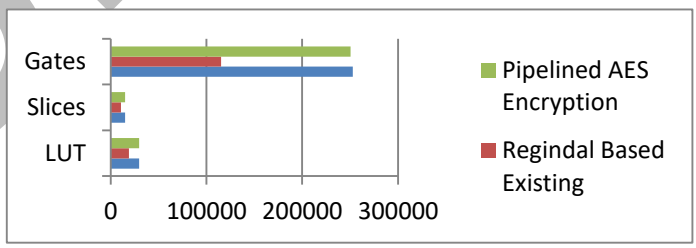


DESIGN UTILIZATION SUMMARY

AREA AND DELAY COMPARISON TABLE

Method Name	Area in Number of LUT			Delay		
Spartan 3 XC 3S 5000-4FG1156	LUT	Slices	Gates	Delay	Gate or Logic Delay	Path or Route Delay
Pipelined Based Encryption	29672	15012	250455	106.865 ns	43.155ns	63.710ns
Existing Rejindal Based Encryption	18878	9906	115446	298.073	198.209 ns	184.329ns
LUT based Encryption	29832	15045	252696	110.375 ns	45.245ns	65.130ns

AREA GRAPH:



DELAY GRAPH:

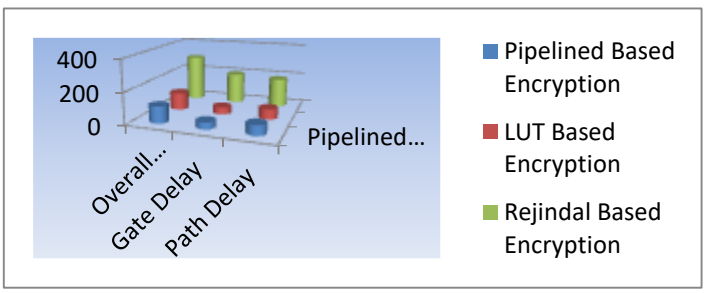


Fig 9: Energy consumption x-graph

 ISBN: 978-81-910765-1-6

VI CONCLUSION

A novel Pipelined AES Design implementation with High Security Constrains. The Implementation is based on mathematical properties of Pipelined algorithm and remains, Another contribution of this paper is that it designs Encryption Design using Shit rows, Mixed Column, Add Round Key and We Will Design a Decryption Part also. Finally with the help of Key Generation is used for Encryption and Decryption Process. The new design permits the construction of efficient area and speed characteristics, while still keeping a very high protection level. We conducted relevant AES Implementation with for Key Generation Method. Nearly all the algorithms embedded in Cryptographer have been designed to resist at high level to linear, differential and high order differential attacks, whereas nothing has been done to make them inherently resistant attacks. However, this work will be implemented in the FPGA. It is possible to design algorithms, so when the next generation of cryptographic.

REFERENCES

1. K. Fu and J. Blum, "Controlling for cybersecurity risks of medical device software," Commun. ACM, vol. 56, no. 10, pp. 35–37, Oct. 2013.
2. M. Rostami, W. Burleson, A. Jules, and F. Koushanfar, "Balancing security and utility in medical devices?" in Proc. 50th ACM/EDAC/IEEE Int. Conf. Design Autom., May/Jun. 2013, pp. 1–6.
3. M. Zhang, A. Raghunathan, and N. K. Jha, "Trustworthiness of medical devices and body area networks," Proc. IEEE, vol. 102, no. 8, pp. 1174–1188, Aug. 2014.
4. H. Khurana, M. Hadley, N. Lu, and D. A. Frincke, "Smart-grid security issues," IEEE Security Privacy, vol. 8, no. 1, pp. 81–85, Jan./Feb. 2010.
5. M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A low-power high performance concurrent fault detection approach for the composite field S-box and inverse S-box," IEEE Trans. Comput., vol. 60, no. 9, pp. 1327–1340, Sep. 2011.
6. M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A lightweight high performance fault detection scheme for the Advanced Encryption Standard using composite fields," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 1, pp. 85–91, Jan. 2011.
7. A. Satoh, T. Sugawara, N. Homma, and T. Aoki, "High-performance concurrent error detection scheme for AES hardware," in Proc. 10th Int. Workshop CHES, Aug. 2008, pp. 100–112.
8. X. Guo and R. Karri, "Recomputing with permuted operands: A concurrent error detection approach," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 32, no. 10, pp. 1595–1608, Oct. 2013.
9. M. Mozaffari-Kermani and R. Azarderakhsh, "Efficient fault diagnosis schemes for reliable lightweight cryptographic ISO/IEC standard CLEFIA benchmarked on ASIC and FPGA," IEEE Trans. Ind. Electron., vol. 60, no. 12, pp. 5925–5932, Dec. 2013.