International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 7, July 2015, PP 187-195
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Implementation of AES Algorithm for Area and Power Optimization by the Application of Reversible Logic and Null Conventional Logic

Poojitha Avuta[1], Anitha Patibandla[2], Gayathri Aouta[3]

[1]Department of ECE, MRCET, Hyderabad, India (PG Scholar)
[2]Department of ECE, MRCET, Hyderabad, India (Associate Professor)
[3]Department of CSE, UCE, OU, Hyderabad, India (Assistant Professor)

ABSTRACT

Advanced Encryption Standard (AES) is a symmetric key generation for encryption, it contains three different key sizes like 128bit, 192bit and 256bit; in this paper the 128 bit encryption uses reversible logic and null conventional logic. Reversible logic gates are fredkin gate, cnot gate. Proposed design contains null conventional logic with dual rail encoding using this we design NCL SBOX based on NCL XOR gate and NCL AND gate. The128 bit encryption using reversible Logic and null conventional logic is simulated and synthesized using VHDL Xilinx 13.2 version and verified using Spartan 3e kit.

Keywords: reversible logic, null conventional logic and encryption.

INTRODUCTION

Reversible logic circuits have attracted the attention of researchers in recent years for mainly two reasons. Firstly, Landauer [8] showed that during logic computation, every bit of information loss generates KT ln2 joules of heat energy, where K is the Boltzmann's constant and T is the absolute temperature of environment. And, according to Ben net [2], for theoretically zero energy dissipation, computations have to be reversible in nature. Secondly, quantum computations which are the basis of quantum computers are reversible in nature.

In the field of cryptography, there has been many works that propose hardware implementations of cryptographic primitives [13]. Some of these implementations use optimized architectures for high-speed operations, some for area-efficiency targeted to low-cost implementations where speed is not the major concern, some more general-purpose with limited capabilities of reconfigurability, while some optimized for low-power applications. By their very design, some of the cryptographic primitives like encryption and decryption are reversible in nature. However, to the best of the knowledge of the authors, no complete reversible logic implementations of such algorithms have been reported. However, some works on the reversible implementations of specific subsystems of a cryptographic processor, namely the Montgomery multiplier. Another motivation for studying reversible logic implementation of cryptographic algorithms results from the fact that side-channels in hardware implementations of such algorithms have been widely studied in recent times [7]. Side-channel attack is considered to be a very cost-effective alternative to attacking traditional cryptographic algorithms, and designers use various countermeasures in this regard. Power analysis attack is one of the easiest attack to mount, and is based on the variations in power dissipation during a computation. Since reversible logic circuits are expected to consume must less energy as compared to traditional CMOS logic, variations in power consumptions will be less and hence side-channel

Address for correspondence:

poojitha.avuta@gmail.com

attacks will be more difficult to mount.

With this motivation, this paper reports the results of re-eversible logic implementation of a state-of-the-art block cipher, the 128-bit Advanced Encryption Standard (AES). The rest of the paper is organized as follows. Section 2 introduces some basic concepts in reversible logic synthesis. Section 3 serves dual purpose; it gives brief introductions to the various steps in AES encryption process, and also discusses the reversible logic implementations of the same. Section 4 discusses the synthesis framework, and presents the experimental results. Section 5 summarizes the paper and identifies a few areas for future work.

REVERSIBLE LOGIC AND REVERSIBLE GATES

Preliminaries

A Boolean function f: $B^n \rightarrow B^n$ is said to be reversible if it is objective. In other words every input vector is uniquely mapped to an output vector. The problem of synthesis is to determine a reversible circuit that realizes a given function f.In this paper, for the purpose of synthesis we consider the gate library consisting of multiple-control Toffoli (MCT) gates. An n-input MCT gate with inputs (x_1, x_2. . . x_n) pass the first (n − 1) inputs unchanged, and complements the last input if all the remaining (n − 1) inputs are at 1. Figure 1 shows an n-input MCT gate. A simple NOT (n = 1) and controlled-NOT or CNOT (n = 2) are special cases of the MCT gate.Any reversible function can be implemented as a cascade of reversible gates, without any fan out or feedback. To estimate the cost of an implementation, several metrics are used,

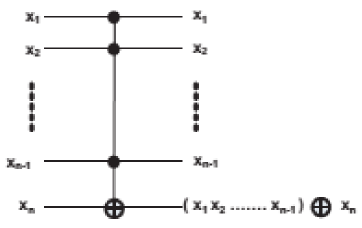


Fig. 1. *n*-input MCT gate

Namely, number of gates, number of equivalent MOS transistors, and number of equivalent basic quantum operations called the quantum cost [1]. There are standard ways of computing the quantum cost from a given gate net list [3]. Some works also try to reduce the number of Garbage Outputs, which are the outputs that are don't cares for all possible input conditions.

AES ALGORITHM AND ITS IMPLEMENTATION

The top-level structure of the AES encryption process is shown in Figure 2, which takes as input an 128-bit plaintext P and an 128-bit key K, and produces as output an 128-bit cipher text C. The basic steps in the encryption process are shown in Figure 3, which is divided into ten iterations or rounds. There are four distinct operations that are carried out in a specific order: AddRoundKey (ARK), Byte Substitution (BS), Shift Row (SR) and Mix Columns (MC). The 128-bit data blocks are divided into groups of 16 bytes each, and organized in the form of a 4 × 4 State Matrix. After an initial ARK step, nine rounds are performed, each consisting of a sequence of four operations {BS, SR, MC, ARK}. In the tenth round, only three steps {BS, SR, ARK} are carried out. The ARK step also takes another 128-bit input, the (transformed) key, which is generated by a separate Key Scheduler module as shown in Figure 4. The first ARK step takes the User Key, while the following ten rounds use transformed keys

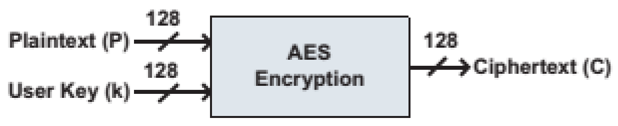


Fig. 2. Top-level schematic of AES encryption

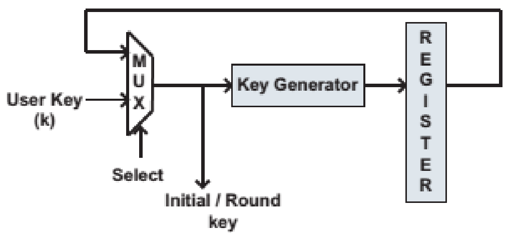


Fig. 4. Iterative key generation

Implementation of Byte Substitution

This step in the AES algorithm carries out a non-linear transformation on each byte of the State matrix independently. The transformation is carried out using the S-box, which basically implements a permutation of 8-bit integers (in the range 0 to 255).

Two alternate schemes for implementing an S-box using reversible logic gates is depicted in Figure 5. Figure 5(a) shows the block diagram of an implementation that does not require any garbage output lines, while Figure 5(b) shows the block diagram that uses eight garbage lines.

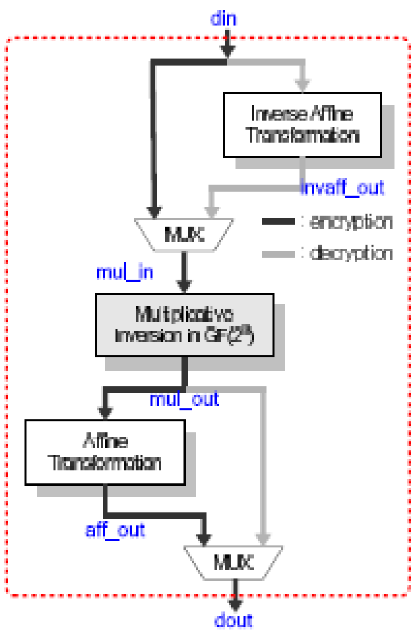
NCL AES S-Box design

AES algorithm consists of a number of rounds that are dependent on the key size. For both cipher and decipher of AES algorithm, each round consists of linear operation (i.e., AddRoundKey, Shift Rows, and Mix Columns steps) and non-linear operation (i.e., Sub Bytes step). Sub Bytes step is the first step of AES round. Each byte in the array is updated by an 8-bit substitution box (S-Box), derived from the multiplicative inverse over GF. AES S-Box is constructed by combining the inverse function with an invertible affine transformation in order to avoid attacks based on mathematics. The S-Box is one the of most critical components in the implementation of AES hardwares. It consumes the majority of power and is also the most vulnerable component to SCAs. A block diagram of AES S-Box is shown in Figure The block diagram of multiplicative inversion over GF component where MM modular multiplication is, and XOR is exclusive-or operation. The hardware implementation of AES S-Box follows the combinational logic circuit architecture, but uses NCL gates

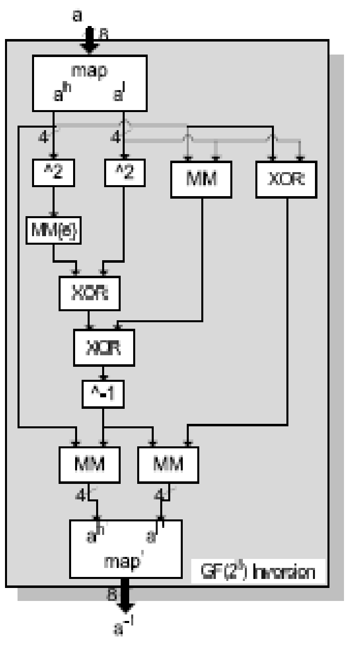
Block Diagram of a Combinational S-Box with Encryption and Decryption Data Paths

Instead of Boolean logic gates. The affine transformation and inverse affine transformation components follow a series of Boolean equations .where I and q represents the 8-bit input and output, respectively. Both transformations re- quire many XOR gates. The multiplicative inversion in GF follows the procedure shown in Figure part (b). 1) Map operation converts the 8-bit input into elements of GF (i.e. ah and al); 2) Calculate the square of ah and al. It should be noticed that multiplication in GF is done by multiplying the polynomial ah(x) ah(x) followed by a modular reduction; 3) A series of multiplication and XOR operations were implemented to extend the field

GF to the field GF. To implement this conventional S-Box using NCL, the XOR, AND, and MUX operation in dual-rail NCL gates are required. NCL has a total of 27 threshold gates to realize various logic functions. In order to achieve the input-completeness and observability, it is important to choose appropriate threshold gates. For example, in the design of a two-to-one multiplexer, according to the Karnaugh map in



(a)



(b)

Boolean Equations for Affine Transformation and Inverse Affine Transformation Components

$q = aff\_trans(i)$	$q = aff\_trans^{-1}(i)$
$q_0 = (i_0 \oplus i_4) \oplus (i_5 \oplus i_6) \oplus (i_7 \oplus 1)$	$q_0 = i_2 \oplus i_5 \oplus i_7 \oplus 1$
$q_1 = i_1 \oplus i_5 \oplus i_6 \oplus i_7 \oplus i_0 \oplus 1$	$q_1 = i_0 \oplus i_3 \oplus i_6$
$q_2 = i_2 \oplus i_6 \oplus i_7 \oplus i_0 \oplus i_1$	$q_2 = i_1 \oplus i_4 \oplus i_7 \oplus 1$
$q_3 = i_3 \oplus i_7 \oplus i_0 \oplus i_1 \oplus i_2$	$q_3 = i_2 \oplus i_5 \oplus i_0$
$q_4 = i_4 \oplus i_0 \oplus i_1 \oplus i_2 \oplus i_3$	$q_4 = i_1 \oplus i_3 \oplus i_6$
$q_5 = i_1 \oplus i_5 \oplus i_2 \oplus i_3 \oplus i_4 \oplus 1$	$q_5 = i_2 \oplus i_4 \oplus i_7$
$q_6 = i_6 \oplus i_2 \oplus i_3 \oplus i_4 \oplus i_5 \oplus 1$	$q_6 = i_0 \oplus i_3 \oplus i_5 \oplus 1$
$q_7 = i_7 \oplus i_3 \oplus i_4 \oplus i_5 \oplus i_6$	$q_7 = i_1 \oplus i_4 \oplus i_6$

$$Z^0 = A^0S^0 + S^1B^0; \tag{8}$$

$$Z^1 = A^1S^0 + S^1B^1; \tag{9}$$

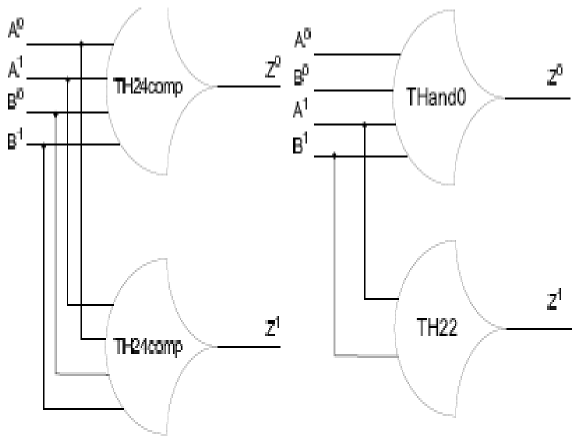
After modifying both functions for input-completeness, new SOP functions are obtained:

$$Z^0 = A^0S^0(A^0 + A^1)(B^0 + B^1) + S^1B^0(A^0 + A^1)(B^0 + B^1); \tag{10}$$

$$Z^1 = A^1S^0(A^0 + A^1)(B^0 + B^1) + S^1B^1(A^0 + A^1)(B^0 + B^1); \tag{11}$$

And both of them can be mapped to a NCL circuit with a TH24comp gate, a THand0 gate, and a TH22 gate. The finalized NCL MUX logic diagram is shown in Figure 7.4 (b). Likewise, two TH24comp gates can be used to implement an XOR logic function. A THand0 and a TH22 gate are used to implement an AND logic function. The logic diagrams.

Complete NCL XOR (left) and NCL AND (right) functions



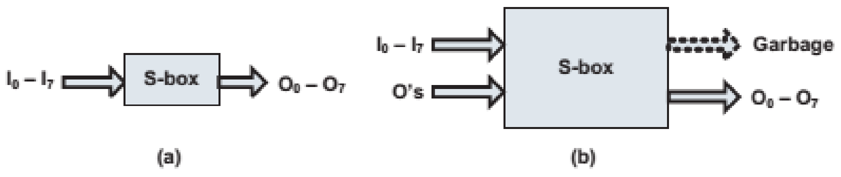


Fig. 5. Alternate reversible designs of the S-box

Since there are 16 bytes of the State matrix, to carry out the transformation in parallel, we need 16 S-boxes.

Implementation of Shift Rows

This step basically implements fixed cyclic shift operations on the rows of the State matrix, and as such can be implemented by permuting the input bits to get the output bits. No gates or hardware components are required for this step.

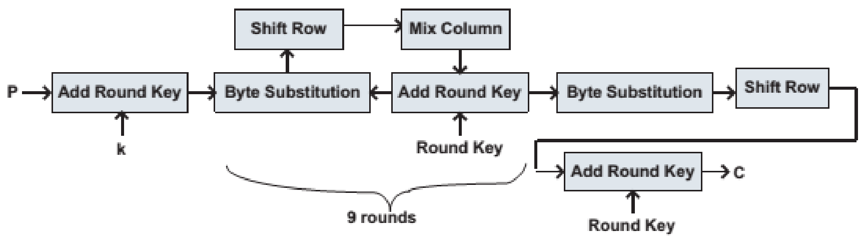


Fig. 3. Steps of AES encryption

Implementation of Mix Columns

In this step, each column of the State matrix is treated as a polynomial over GF (2^8), and is multiplied by a predefined polynomial $03.x^3 + 01; x^2 + 01.x + 01$ modulo ($x^4 + 1$). This can be formulated using matrix multiplication as follows:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ O_4 \end{bmatrix}$$

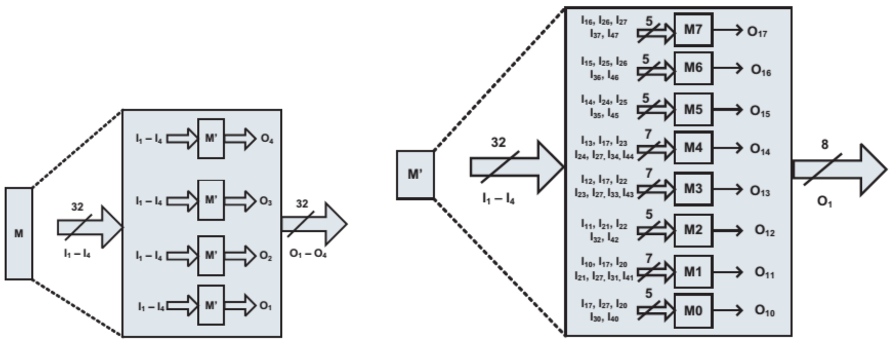


Fig. 6. Reversible schematic of *MixColumns*

There are four identical Mix Column boxes in every round. The inputs and outputs to these boxes are 32-bits long. We represent the inputs as I_1-I_4 and the outputs as O_1-O_4, where each I_i and O_i is 8-bits long; the block level schematic for the reversible implementation of Mix Columns is shown in Figure 6.

Implementation of AddRoundKeys

In this step, the output from the Mix Columns step is XOR-ed with the corresponding round key. The step can be efficiently implemented in reversible logic using a CNOT gate for every bit. The i^{th} CNOT gate will have the i^{th} bit of key as control input, and i^{th} bit of Mix Columns output as target. A total of 128 CNOT gates are required for realizing this step. This is shown in Figure 7.

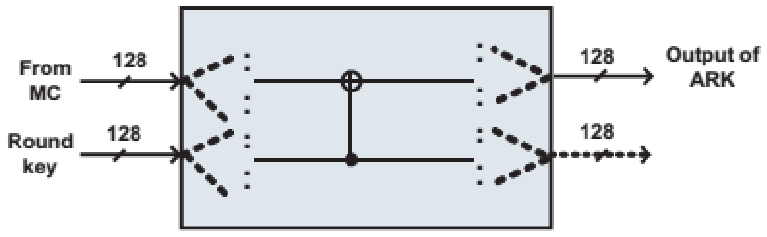


Fig. 7. Reversible implementation of *AddRoundKey*

Implementation of Key Scheduler

The Key Scheduler is responsible for generating the round keys to be used in the AddRoundKey steps. As illustrated in Figure 4, it uses a Key Generator module in a repetitive fashion to generate the successive round keys. The Key Generator module has three essential steps:

a) A rotate left one word step that can be implemented by wiring alone.

b) The Byte Substitution step, where the same S-boxes as used in the main encryption flow are used.

c) An 128-bit XOR step, which can again be easily implemented using 128 CNOT gates.

Pipelined Implementation of AES Encryption

In order to have high throughput, we can overlap successive block encryption processes by implementing the AES encryptor as a pipeline. Since the overlapped encryption processes may use different keys, the Key Scheduler also needs to be pipelined to generate the round keys for successive encryption processes in an overlapped fashion. A block level diagram of the pipelined implementation is shown in Figure 8.

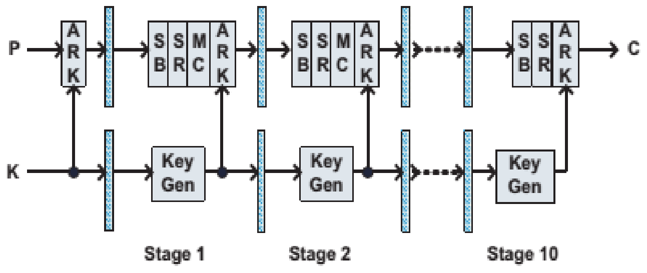


Fig. 8. Reversible pipelined implementation of AES

In Figure 8, each of the blocks SB, SR, MC, ARK and KeyGen are implemented using reversible logic gates. The registers that serve to isolate the pipeline stages are all 128-bits wide, and are implemented in a reversible way as shown in Figure 9

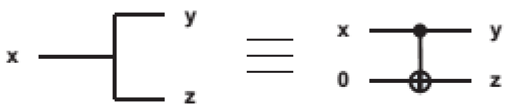


Fig. 10. 128-bit reversible register [14]

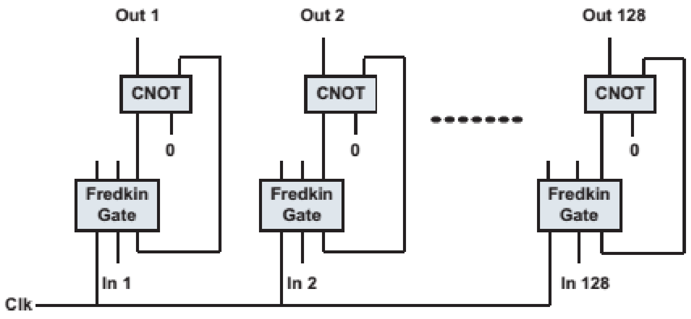
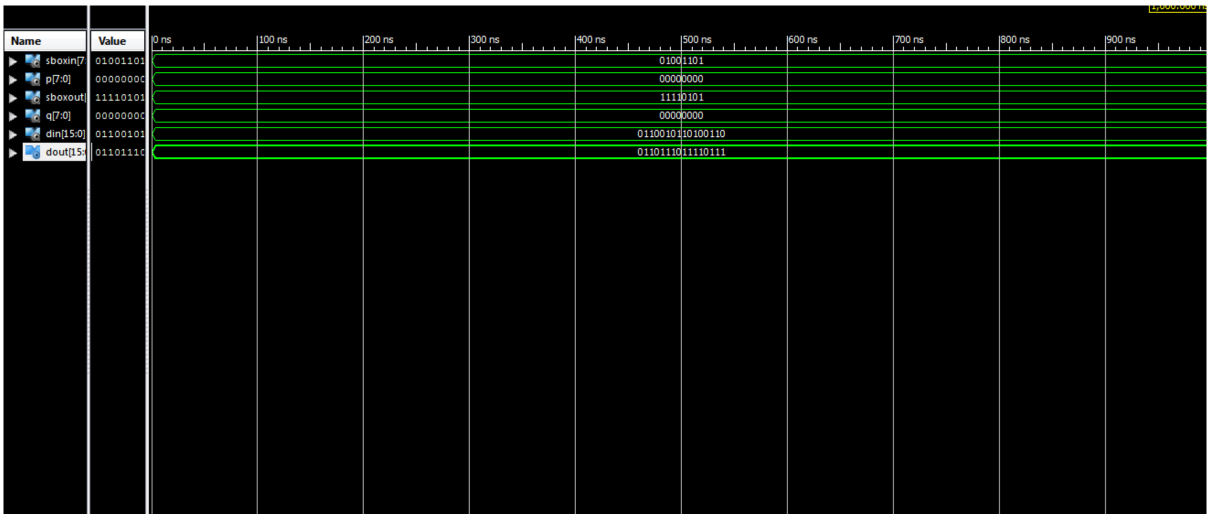


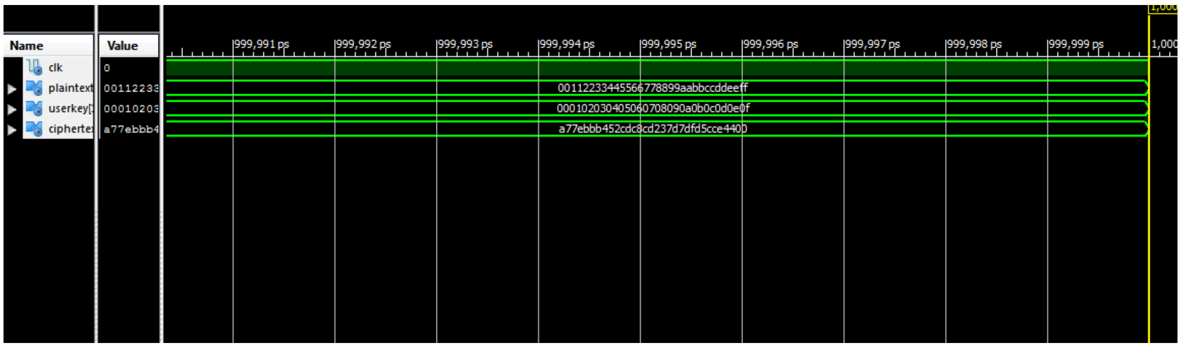
Fig. 9. 128-bit reversible register [14]

SIMULATION WAVEFORMS

NCLSBOX



Encryption



```
clk          0
plaintext    00112233   00112233445566778899aabbccddeeff
userkey[     00010203   000102030405060708090a0b0c0d0e0f
ciphertex    a77ebbb4   a77ebbb452cdc8cd237d7dfd5cce4400
```

CONCLUSION

The reversible logic and null conventional logic implementation of 128 bit AES encryption algorithm discussed in this paper. The sbox using null conventional logic and pipelining concept using reversible logic gates. The reversible logic and null conventional logic using Aes algorithm of synthesizable vhdl code is developed for the implementation of 128 data encryption using Xilinx 13.2 simulator.

REFERENCES

[1] A. Barenco, H. H. Bennett, R. Cleve, D. P. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. Physical Review A (Atomic, Molecular, and Optical Physics), 52(5):3457–3467, 1995.

[2] C. H. Bennett. Logical reversibility of computation. Journal of IBM Research and Development, 17:525–532, 1961.

[3] R. Drechsler, A. Finder, and R. Wille. Improving ESOP-based synthesis of reversible logic using evolutionary algorithms. In Proceedings of Intl. Conference on Applications of Evolutionary Computation (Part II), pages 151–161, 2011.

[4] K. Fazel, M. A. Thornton, and J. Rice. ESOP-based Toffoli gate cascade generation. In Proceedings of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pages 206–209, 2007.

[5] D. Grosse, R. Wille, G. W. Dueck, and R. Drechsler. Exact multiple control Toffoli network synthesis with SAT techniques. IEEE Trans. on CAD of Integrated Circuits and Systems, 28(5):703–715, May 2009.

[6] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski. Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. IEEE Trans. on CAD of Integrated Circuits and Systems, 25(9):1652–1663, September 2006.

[7] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In

[8] Proceedings of Advances in Cryptology (CRYPTO '99), LNCS Vol. 1666, pages 388–397, 1999.

[9] R. Landauer. Irreversibility and heat generation in computing process.

[10] Journal of IBM Research and Development, 5:183–191, 1961.

[11] D. M. Miller, D. Maslov, and G. W. Dueck. A transformation based algorithm for reversible logic synthesis. In Proceedings of Design Automation Conference, pages 318–323, 2003.

[12] A. Mishchenko and M. Perkowski. Fast heuristic minimization of exclusive-sums-of-products. In Proceedings of 6th Reed-Muller Work-shop, pages 242–250, 2001.

[13] N. Nayeem, L. Jamal, and H. Babu. Efficient reversible Montgomery multiplier and its

applications to hardware cryptography. Journal of Computer Science, 5(1):49–56, January 2009.

[14] N. Nayeem and J. E. Rice. A shared-cube approach to ESOP-based

[15] synthesis of reversible logic. Facta Universitatis of NiE, Elec. Energ., 24(3):385–402, 2011.

[16] F. Rodriguez-Henriquez, N. Saqib, A. Perez, and C. Koc. Cryptographic Algorithms on Reconfigurable Hardware. Springer: Series on Signals and Communication Technology, New York, 2006.

[17] H. Thapliyal and M. Zwolinski. Reversible logic to cryptographic hardware: a new paradigm. In Proceedings of 49th Midwest Symposium on Circuits and Systems (MWSCAS '06), pages 342–346, 2006.

[18] R. Wille and R. Drechsler. BDD-based synthesis of reversible logic for large functions. In Proceedings of Design Automation Conference, pages 270–275, 2009

AUTHORS' BIOGRAPHY

Poojitha Avuta, has completed her B.TECH in Electronics and Communication Engineering from BHOJ REDDY ENGINEERING COLLEGE FOR WOMEN, J.N.T.U.H affiliated college in 2013.She is pursuing her M.Tech in VLSI AND EMBEDDED SYSTEMS from MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY , J.N.T.U.H affiliated college.

Anitha Patibandla, is an Associate Professor at MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY, Hyderabad. She received her Bachelor degree in Electronis and Communication Engineering from OSMANIA UNIVERSITY COLLEGE OF ENGINEERING, Hyderabad, and M.Tech in V.L.S.I System Design from J.N.T.U.H affiliated college. She is presently pursuing Ph.D at JNTUH. Her research interest is Low power VLSI Design.

Gayathri A, is an Assistant Professor (C) at Department of Computer Science & Engineering, University College of Engineering, Osmania University, and Hyderabad. She received her Bachelor degree in Computer Science & Engineering from Sree Visvesvaraya Institute of Technology & Science, Mahabunnagar, J.N.T.U.H affiliated Engineering College, M.Tech, Computer Science & Engineering from Technology, Hyderabad, and J.N.T.U.H affiliated college. She is presently teaching Network Security to M.Tech Students of University College of Engineering, Osmania University and Hyderabad.