International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 12, December 2015, PP 138-145
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Implementation of Pipelined Architecture for AES Algorithm using Reversible Logic

Dasari Nagaveni[1], C.H Pushpalatha[2]

[1]*Department of ECE, Gonna Institute of Technology, Vishakhapatnam, India (Pg Scholar)*
[2]*Department of ECE, Gonna Institute of Technology, Vishakhapatnam, India (Associate Professor)*

ABSTRACT

Advanced Encryption Standard (AES) is highest Bit size and highly secured algorithm in this algorithm in this paper first time implementation of block cipher with reversible logic we Have 128 bit encryption and decryption using reversible Logic Is designed and synthesized using Vhdlcode This algorithm is implemented in Xilinx 13.2 version and verified using Spartan 3e kit.

Keywords: Reversible Logic, Encryption, Decryption.

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 attacks will be more difficult to mount.

Address for correspondence:
nagaveni.kadapa@gmail.com

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, 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.

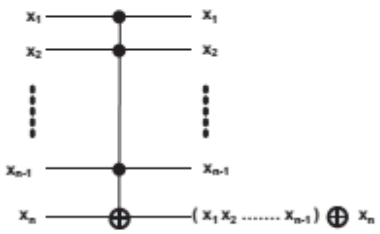


Fig1. *N-input MCT gate*

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.

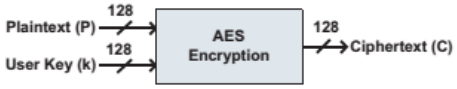


Fig2. *Top-level schematic of AES encryption*

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.

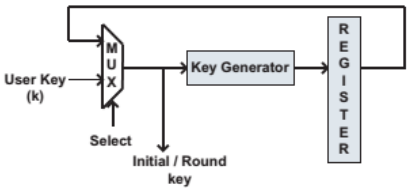


Fig3. *Iterative Key Generation*

Implementation of Byte Substitution

This step in the AES algorithm carries out a non-linear bijective 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.

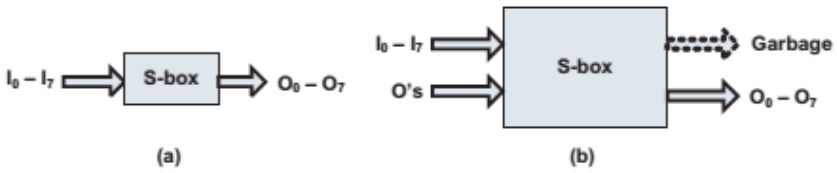


Fig4. *Alternate Reversible Designs of the S-Box*

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.

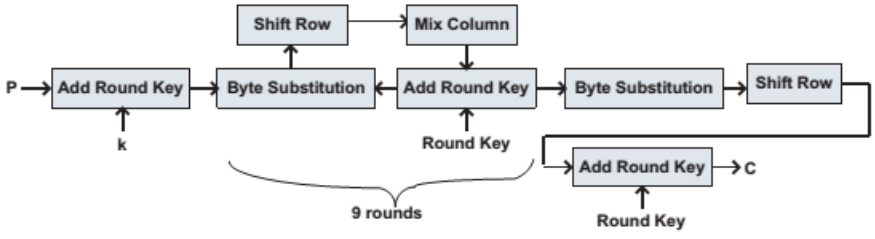


Fig5. *Steps of AES encryption*

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:

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

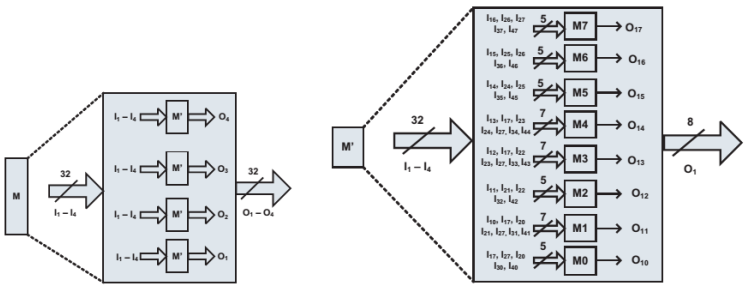


Fig6. *Reversible schematic of mix columns*

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 Add Round Keys

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.

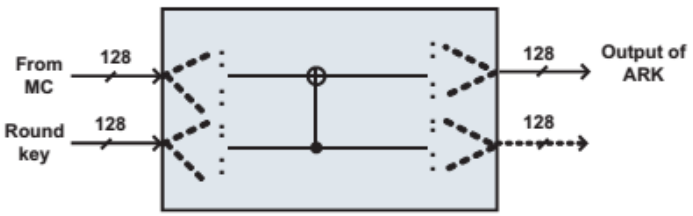


Fig7. *Reversible implementation of add round key*

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 rotate left one word step that can be implemented by wiring alone.
- The Byte Substitution step, where the same S-boxes as used in the main encryption flow are used.
- 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.

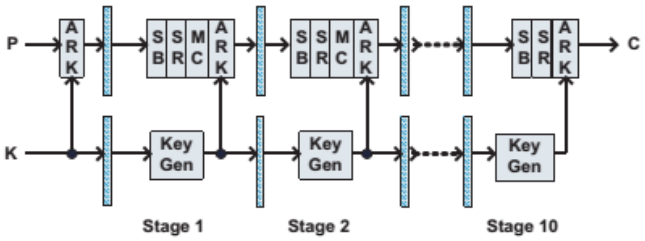


Fig8. *Reversible pipelined implementation of AES*

In Figure 8, each of the blocks SB, SR, MC, ARK and Key Gen 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

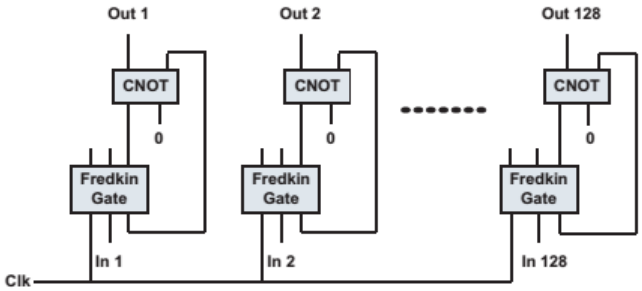


Fig9. *128-bit reversible register [14]*

Fig10. *128-bit reversible register [14]*

AES DECRYPTION

For decryption, the same process occurs simply in reverse order – taking the 128-bit block of cipher text and converting it to plaintext by the application of the inverse of the four operations. Add Round Key is the same for both encryption and decryption. However the three other functions have inverses used in the decryption process: Inverse Sub Bytes, Inverse Shift Rows, and InverseMixColumns.

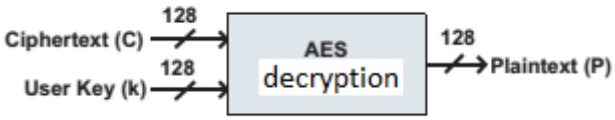


Fig11. *Top-level schematic of AES encryption*

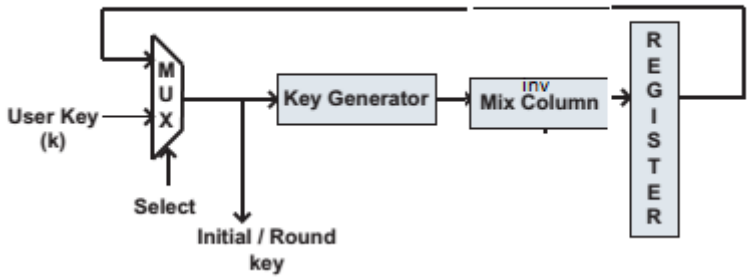


Fig12. *Inverse key generation*

Implementation of inverse Byte Substitution

This step in the AES algorithm carries out a non-linear bijective 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 inv S-box using reversible logic gates is depicted in Figure 13. Figure 13(a) shows the block diagram of an implementation that does not require any garbage output lines, while Figure 13(b) shows the block diagram that uses eight garbage lines.

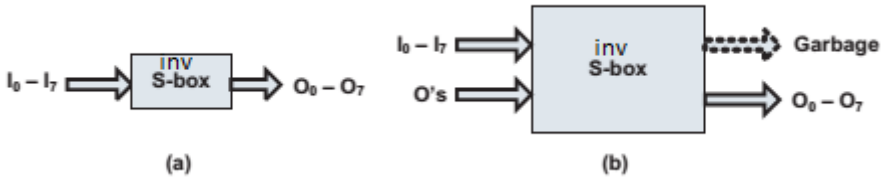


Fig13. *Alternate Reversible Designs of the S-Box*

Implementation of inverse Shift Rows

This step basically implements fixed cyclic right shift operations on the rows of the State matrix

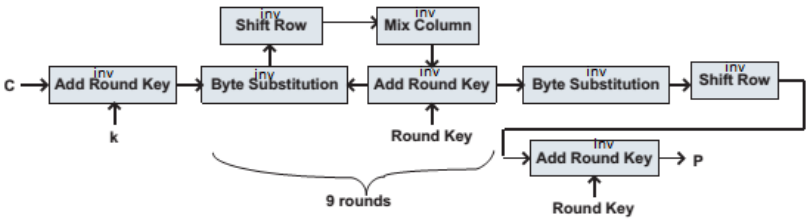


Fig14. *Steps of AES decryption*

Implementation of inverse Mix Columns

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

$$s'_{0,c} = (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$

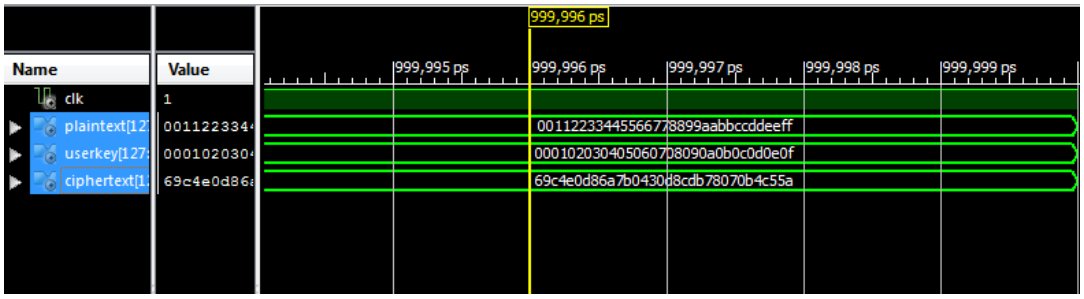
$$s'_{1,c} = (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c})$$

$$s'_{2,c} = (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c})$$

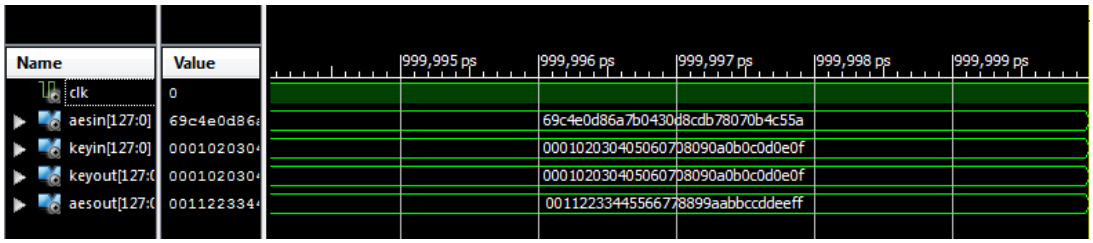
$$s'_{3,c} = (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c})$$

SIMULATION WAVEFORMS

Encryption



Decryption



CONCLUSION

The AES algorithm is an private key symmetric algorithm that can process 128bit plaintext and different sizes of keys based on different rounds The reversible logic of 128-bit AES encryption and decryption is presented in this paper optimized and synthesizable vhdl code is developed for the implementation of both 128 data encryption and decryption 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 ˆ

[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

DasariNagaveni, has received her B. Tech Degree in 2008 Electronics and communication engineering from K.S.R.M College, Kadapa and pursuing M.Tech in VLSI DESIGN from GONNA INSTITUTE OF TECHNOLOGY Vishakhapatnam. Her current area of research includes REVERSIBLE LOGIC.

C.H Pushpalatha, has completed her B.Tech in 2011 Electronics and communication engineering from NIE Guntur and completed her M.Tech in AVANTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY Vishakhapatnam. Now working as an associate professor in GONNA INSTITUTE OF TECHNOLOGY Vishakhapatnam.