# Design of High-throughput and Area efficient hardware using AES Algorithm

Rajalakshmi A and Ashok kumar A

**Abstract**— The Advanced Encryption Standard is a replacement for Data Encryption Standard, which is not only has comparable security strength, but also achieves significant improvement of energy efficiency for both software and hardware implementation. In encryption process, the AES accepts a plaintext input, which is limited to 128 bits and a key that can be specified to be 128 bits to generate the Cipher text and in similar manner of decryption process. The AES can be implemented in different granularities and task parallelism as a onetime one Processor (OTOP), Small Encryption, Parallel Mixed Column and Full Parallelism. We map these implementations using Modelsim. The proposed design occupies less area and small delay by reducing the number of cores and also achieves the higher throughput performance per chip area.

**Index Terms**— Advanced Encryption Standard (AES), Decryption process, Fine grain many core system, Full parallelism, Model sim, Parallel processor, Synchronous Dataflow, Model sim

## 1 Introduction

Cryptography is a technique used for creating and using a cryptosystem or cipher to prevent all the data and information and for communicating sensitive material across computer networks

In 1997 the National Institute of Standards and Technology (NIST), a branch of the US government, started a process to identify a replacement for the Data Encryption Standard (DES). It was generally recognized that DES was not secure because of advances in computer processing power. The NIST invited cryptography and data security specialists from around the world to participate in the discussion and selection process. In 2001, the NIST selected the Rijndael algorithm also called Advanced Encryption Standard (AES) which is in common use today. The AES has been used various applications such as, military application, banking sectors, automated teller machines, government information transactions, secure communication, Radio frequency identifier (RFID), smart cards, digital audio and video recorders and so on.

The first AES implementation is a combination of the SubBytes, ShiftRows, and MixColumns phases in the AES algorithm and the AES system can achieves the throughput per chip area through parallelism. These common techniques used to enhance the performance of a system In general, the hardware implementations of AES offer higher throughput and better energy efficiency than software designs but it is time consuming. The AES can be implemented in different granularities and task parallelism as a OTOP, Small Encryption, Parallel Mixed Column and Full Parallelism using inverse transformation techniques of decryption process.

While designing the system, the area should be major concern, the area is represented by the number of cores required to implement applications. The proposed design occupies smaller area translates into fewer used cores and leaves more opportunities for dealing with other applications on the same platform simultaneously.

## 2 Advanced Encryption Standard

The AES encryption algorithm is a symmetric block cipher that uses an encryption key for several rounds and 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. The AES algorithm is not a computer program or computer source code but also a mathematical description of a process of obscuring data.

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 a 128-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. An encryption key is simply a binary string of data used in the encryption process. Because the same encryption key is used to encrypt and decrypt data, it is important to keep the encryption key a secret and to use keys that are hard to guess. Some keys are generated by software used for this specific task. Another method is to derive a key from a pass phrase. Good encryption systems never use a pass phrase alone as an encryption key.

Side channel Attacks are attacks on the implementation of AES, not on the input or the AES cipher text. It attempts to correlate various measurements of the encrypting tool with time in an attempt to guess the key. The algorithm on the Pentium III running FreeBSD 4.8 and by measuring time delays

- *Rajalakshmi A is currently pursuing master degree program in Applied electronics, Arunai Engineering College, Thiruvannamalai, Tamilnadu, India, PH-+91-9597145035. E-mail: rajlak89@mail.com*
- *Ashok kumar A is currently pursuing master degree program in VLSI Design, S.K.P Engineering College, Thiruvannamalai, Tamilnadu. India. PH-+918939667910. E-mail: anbuazok@gmail.com*

between the CPU and memory was able to successfully guess the key in under 100 minutes.

## 3 OPERATION OF AES

The AES also called as Rijndael which 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 its 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

These different modules of operation can be implemented using AES for analyzing the area throughput tradeoffs on the Full-parallelism.

### 3.1 SUB BYTES

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 using Figure 1.1

The implementation of the composite field S-BOX is accomplished using combinational logic circuits rather than using pre-stored S-BOX values. S-BOX substitution starts by finding the multiplicative inverse of the number in

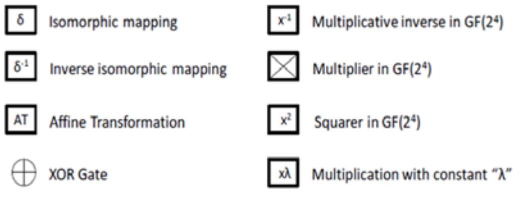

Figure 1.1 Internal Blocks

#### 3.1.1 SUB 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

#### 3.1.2 MULTIPLIER IN GF (2^4)

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. Composite field arithmetic can be employed to reduce the hardware complexity.

Three multipliers in GF($2^4$) are required as a part of finding the multiplicative inverse in GF(2^8). Figure 1.2 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.

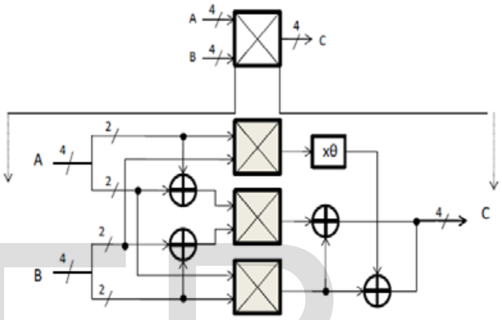

Figure 1.2 GF($2^4$) Multiplier

### 3.2 ADDROUND KEY TRANSFORMATION

In the AddRoundKeytransformation, a round key is added to the state by Bitwise Exclusive-OR (XOR) operation. Figure 2 illustrates the AddRoundKey. This transformation is the same for both encryption and decryption.

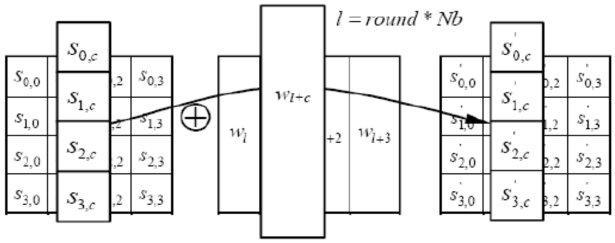

Figure 2. Add Round Key

### 3.3 SHIFT ROWS TRANSFORMATION

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 3. The reverse process, inv Shift Row, operates in reverse order to Shift Rows.

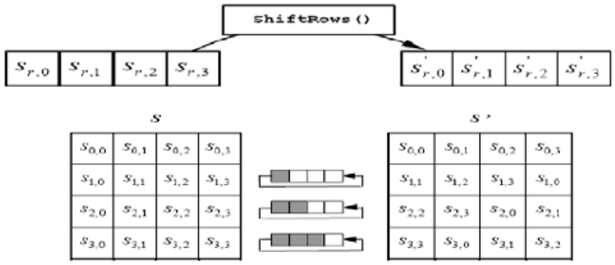

Figure 3. Shift Row and its inverse transformation

### 3.4 MIX COLUMN TRANSFORMATION

The Mix Column transformation is performed independently on the state Column-by-column. Each column is considered as four term polynomial over GF ($2^8$) using Figure 4. and multiplied by

*a*(*x*) modulo (*x*4 + 1) where *a*(*x*) = {03}*x*3 + {01}*x*2 + {01}*x* + {02}This transformation can be expressed in matrix form as

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

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

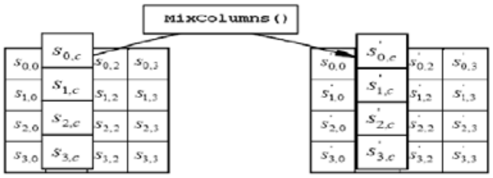

Figure 4. Mixcolumns and its inverse transformation

## 4 PROPOSED WORK OF FULL-PARALLELISM

The Full-parallelism model, shows in Figure 5. illustrates the MixColumns-4 processors are the throughput bottlenecks which determine the performance of the cipher. Therefore, parallelizing the SubBytes process with more than four processors would only increase the area and power overhead without any performance improvement.

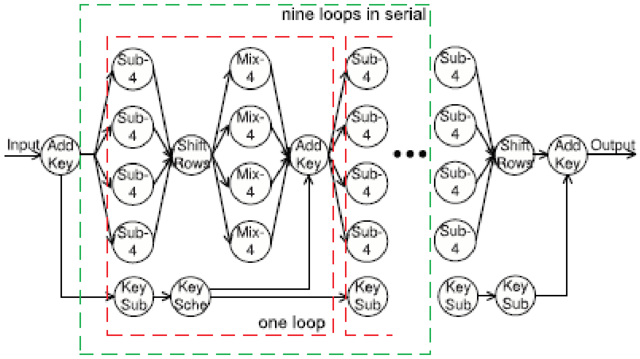

Figure 5.Dataflow diagram of full-parallelism

## 5 BASE PLATFORM

ModelSim provides seamless, scalable performance and capabilities. Through the use of a single compiler and library system for all ModelSim configurations, employing the right ModelSim configuration for project needs is as simple as pointing your environment to the appropriate installation directory.

ModelSim also supports very fast time-tenet-simulation turnarounds while maintaining high performance with its new black box use model, known as bbox. With bbox, non-changing elements can be compiled and optimized once and reused when running a modified version of the test bench. bbox delivers dramatic throughput improvements of up to 3X when running a large suite of test cases.

TABLE 1
COMPARISON OF THROUGHPUT AND NUMBER OF CORES REQUIRED BY DIFFERENT IMPLEMENTATIONS

| Implementation | Area | Delay | Route Delay | Throughput (cycles/byte) |
|---|---|---|---|---|
| OTOP | 21647 | 307.909ns | 202.047ns | 223.875 |
| Parallel Mixcolumn | 21647 | 232.742ns | 115.912ns | 136.250 |
| Fullparallelism Decryption | 4.375 | 232.245ns | 115.562ns | 4.375 |

Table 1 shows the comparison of area and delay reduction achievement in full parallelism decryption process

## CONCLUSION

This paper has been presented the full Parallelism on a fine-grained many-core system and the software implementation exploits both encryption and decryption process of different levels of data and task parallelism. The proposed design requires reduced number of cores by applying parallel processing technique and the approximation of 18 percent of area reduction can be verified in decryption process using Model sim.

## FUTURE WORK

Since AES implementation is not only fit for software implementations, but also suitable for efficient hardware designs. An efficient area analysis on software comparison of decryption process and the full parallelism techniques can be with one time one processor and parallel mix column implementation.

## ACKNOWLEDGMENT

I avail this golden opportunity to express my gratitude to all those people who have helped me to a great deal in preparing my project on "Design of high throughput and area efficient hardware using AES algorithm". First and foremost, I am indebted to the almighty God for providing me with ample strength and poise required to work persistently on this project.

My project would not have been a real fulfilment without the backing and cooperation from various individuals through various means. First of all, I am highly grateful to Mr.P.Sureshkumar HOD-PG(ECE) and Prof Mr.S.Elango HOD(ECE) for Arunai Engineering college,Thiruvannamalai.

I would like to extend my sincere thanks to Mr.Arivumani samson, my project co-ordinator and Mr.Manikandan C, my project guide who gaves me valuable advice and were good enough to find time for fruitful discussions.

Last of all, I am heartily thankful to all my family members for providing me moral support and patience.

## REFERENCES

[1] B.M. Baas, and Z. Yu"A Low-Area Multi-Link Interconnect Architecture for GALS Chip Multiprocessors," *IEEE Trans. Very Large Scale Integration (VLSI) Systems,* vol. 18, no. 5, pp. 750-762, May 2010.

[2] M. Butler, "AMD Bulldozer Core—A New Approach to Multithreaded Compute Performance for Maximum Efficiency and Throughput," Proc. *IEEE HotChips Symp. High-Performance Chips (HotChips '10),* Aug. 2010.

[3] S. Borkar, "Thousand Core Chips: A Technology Perspective," Proc. 44th Ann. Design Automation Conf., pp. 746-749, 2007

[4] D. Bernstein and P. Schwabe, "New AES Software Speed Records," Proc. INDOCRYPT '08: Ninth Int'l Conf. Cryptology in India: Progress in Cryptology, pp. 322-336, 2008.

[5] E. Biham, "A Fast New DES Implementation in Software," Proc. Fourth Int'l Workshop Fast Software Encryption, pp. 260-272, 1997.

[6] J. Daemen and V. Rijmen, "The Design of Rijndael", Springer-Verlag,2002

[7] S. Gueron, "Intel Advanced Encryption Standard (AES) Instructions Set," Jan. 2010.

[8] J. Granado-Criado, M. Vega-Rodriguez, J. Sanchez-Perez, and J. Gomez-Pulido, "A New Methodology to Implement the AES Algorithm Using Partial and Dynamic Reconfiguration," Integration, the VLSI J., vol. 43, no. 1, pp. 72-80, 2010.

[9] A. Hodjat and I. Verbauwhede, "Area-Throughput Trade-Offs for Fully Pipelined 30 to 70 Gbits/s AES Processors," *IEEE Trans. Computers*, vol. 55, no. 4, pp. 366-372, Apr. 2006.

[10] A. Hodjat and I. Verbauwhede, "A 21.54 gbits/s Fully Pipelined AES Processor on FPGA," *Proc. IEEE 12th Ann. Symp. Field-Programmable Custom Computing Machines*, pp. 308- 309, Apr. 2004

[11] E. Kasper and P. Schwabe, "Faster and Timing-Attack Resistant AES-GCM," Proc. 11th Int'l Workshop Cryptographic Hardware and Embedded Systems (CHES '09), pp. 1-17, 2009

[12] S.K. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S.K. Hsu, H. Kaul, M.A. Anders, and R.K. Krishnamurthy, "53 gbps Native GF(ð24Þ2) Composite-Field AES Encrypt/Decrypt Accelerator for Content-Protection in 45 nm High-Performance Microprocessors," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 767-776, Apr. 2011.

[13] S. Morioka and A. Satoh, "A 10-gbps full-AES Crypto Design with a Twisted BDD s-Box Architecture,*" IEEE Trans. Very Large Scale Integration Systems,* vol. 12, no. 7, pp. 686-691, July 2004.

[14] Morsy.M, Rizk.M.R "Optimized Area and Optimized Speed Hardware Implementations of AES on FPGA",Dec 2007

[15] D. Mukhopadhyay and D. RoyChowdhury, "An Efficient end to End Design of Rijndael Cryptosystem in 0:18_m CMOS," Proc. 18th Int'l Conf. VLSI Design, pp. 405-410, Jan. 2005.

[16] S. Qu, G. Shou, Y. Hu, Z. Guo, and Z. Qian, "High Throughput, Pipelined Implementation of AES on FPGA," Proc. Int'l Symp. Information Eng. and Electronic Commerce, pp. 542-545, May 2009.

[17] T. Wollinger, M. Wang, J. Cuajardo, and C. Paar, "How Well are High-end DSPs Suited for the AES Algorithm?," Proc. Third AES Candidate Conf., pp. 94-105, Apr. 2000.