International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Fully Pipelined High Throughput Cost Effective FPGA Based Implementation of AES Algorithm

Athira Das A J[1], Ajith Kumar B P[2]

[1]Student, Dept. of Electronics and Communication, Karavali Institute of Technology, Neermarga, Mangalore, Karnataka

[2]Assistant Professor, Dept. of Electronics and Communication, Karavali Institute of Technology, Neermarga, Mangalore, Karnataka

Abstract: ***This proposes a fully pipelined high-throughput cost effective implementation of Advanced Encryption Standard (AES) supporting encryption and decryption with 128-, 192-, and 256-bit cipher key. AES is the most secure symmetric encryption technique that used for wireless communication. The AES based on the Rijndael Algorithm is an efficient cryptographic technique that includes generation of ciphers for encryption and inverse ciphers for decryption. A high speed security algorithm is always necessary and important for wired/wireless communication. The symmetric block cipher key plays a major role in the bulk data encryption. One of the best existing symmetric security algorithms to provide data security is advanced encryption standard (AES). FPGA-based implementation of the Advanced Encryption Standard (AES) algorithm is presented in this paper. The design has been coded by Very high speed integrated circuit Hardware Descriptive Language. All the results are synthesized and simulated using Xilinx ISE and ModelSim software respectively. This implementation is compared with other works to show the efficiency. The design uses an iterative looping approach with block of 128 bits, lookup table implementation of S-box. This gives low complexity architecture and easily achieves low latency as well as high throughput.***

Keywords: Encryption, Decryption, Rijndael, AES, VLSI

1. Introduction

Cryptography is playing an important role in the security of data transmission with the rapid growing number of Internet and wireless communication users. It enables us to store sensitive information or transmit it across insecure networks so that unauthorized persons cannot read it. The urgency for secure exchange of digital data resulted in large quantities of different encryption algorithms which can be classified into two groups: asymmetric encryption algorithms (with public key algorithms) and symmetric encryption algorithms (with private key algorithms). Symmetric key algorithms are in general much faster to execute electronically than asymmetric key algorithms.

In cryptography, the AES, also known as Rijndael, is a block cipher adopted as an encryption standard by the US government, which specifies an encryption algorithm capable of protecting sensitive information. AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.

- To achieve a high throughput up to tens of Gbps, pipelining, sub-pipelining and loop-unrolling architectures have been explored.
- This project adopts iterative architecture to achieve small area, but the structure can be applied to pipelining and sub-pipelining easily to get a higher throughput for no LUTs or memory are used so that no unbreakable delay is introduced in the architecture.
- Composite field arithmetic has been employed in SubBytes and InvSubBytes to reduce the area requirement, and presented 16 ways to construct the composite field GF ((((2)^2)^2)^2) from 16 sets of different irreducible polynomial coefficients.
- MixColumn/InvMixColumn operations are also optimized. This got a smaller area of 672 XOR gates and 8 XOR gates in the critical path by applying serial InvMixcolumn decomposition.
- This paper uses an architecture that supports three kinds of keys and both encryption and decryption can be handled, which can generate a 128-bit key in one cycle.
- The hardware implementation of the Rijndael algorithm can provide high performance and low cost for specific applications. At backbone communication channels or heavily loaded servers it is not possible to lose processing speed, which drops the efficiency of the overall system while running cryptography algorithms in software.

2. System Architecture

2.1 Architecture Overview

The AES algorithm is also known as the Rijndael algorithm which is a symmetric block cipher that processes data blocks of 128 bits using the cipher key of length 128, 192, or 256 bits. Each data block consists of a 4×4 array of bytes called the State, on which the basic operations of the AES algorithm are performed.

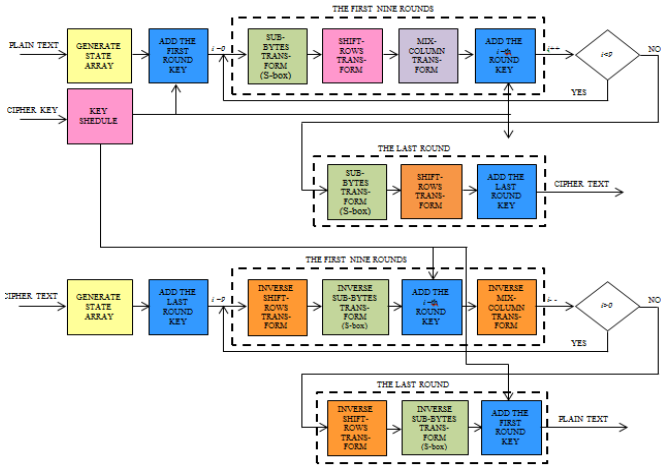

Figure 1: Structure of AES Algorithm

After an initial round key addition, a round function consisting of four different transformations—SubBytes(), ShiftRows(), MixColumns() and AddRoundKey() is applied to the data block, i.e., the State array.

2.2 Glossary of Terms and Acronyms

Cipher: Series of transformations that converts plaintext to cipher text using the Cipher Key.

Cipher Key: Secret, cryptographic key that is used by the Key Expansion routine to generate a set of Round Keys; can be pictured as a rectangular array of bytes, having four rows and **Nk** columns.

Cipher text: Data output from the Cipher or input to the Inverse Cipher.

Inverse Cipher: Series of transformations that converts cipher text to plaintext using the Cipher Key.

Key Expansion: Routine used to generate a series of Round Keys from the Cipher Key.

Plain text: Data input to the Cipher or output from the Inverse Cipher.

Rijndael: Cryptographic algorithm specified in this Advanced Encryption Standard (AES).

Round Key: Round keys are values derived from the Cipher Key using the Key Expansion routine; they are applied to the State in the Cipher and Inverse Cipher.

State: Intermediate Cipher result that can be pictured as a rectangular array of bytes, having four rows and Nb columns.

S-box: Non-linear substitution table used in several byte substitution transformations and in the Key Expansion routine to perform a one for one substitution of a byte value.

Word: A group of 32 bits that is treated either as a single entity or as an array of 4 bytes.

2.3 Architecture of AES

AES operates on a 4×4 array of bytes, termed the state (versions of Rijndael with a larger block size have additional columns state). For encryption, each round of AES consists of four stages;

A. AddRoundKey –Key Expansion(Rijndael key)
B. SubBytes
C. ShiftRows
D. MixColumns

The final round replaces the MixColumns stage with another instance of AddRoundKey. The round function is performed iteratively for 10, 12, or 14 times, depending on the key length. In the last round, MixColumns() does not applied. The four transformations are described briefly as follows;

1. SubBytes(): a non-linear byte substitution that operates independently on each byte of the State using a substitution table (called the S-box).
2. ShiftRows(): a circular shifting operation on the rows of the State with different numbers of bytes (offsets).
3. MixColumns(): the operation that mixes the bytes in each column by the multiplication of the State with a fixed polynomial modulo $x^4 + 1$.
4. AddRoundKey(): an XOR operation that adds a round key to the State in each iteration, where the round keys are generated during the key expansion phase.

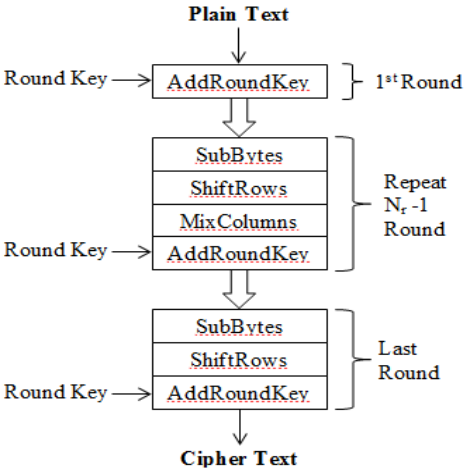

Figure 2: AES Encryption architecture

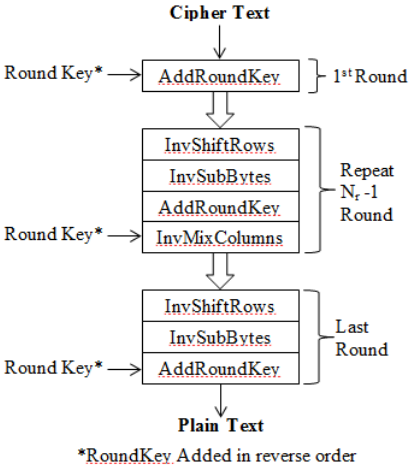

Figure 3: AES Decryption Architecture

Equivalent decryption procedure rearranges the order so that the order of transformations in the decryption procedure keeps consistent with that in the encryption procedure. Thus, resource sharing will be enabled. However, in the decryption procedure, the modified round keys should be applied to the original generated roundkeys using InvMixColumn transformation. Fig.2 and 3 shows the architecture adopted in this project. SubBytes and InvSubBytes transformations are merged using composite field arithmetic. ShiftRows and InvShiftRows are simple shifting transformations. MixColumn and InvMixColumn transformations are optimized and merged.

In the decryption procedure the round key is added to the state first and then the result is applied with the InvMixColumn transformation. In this way modified roundkeys mentioned above can share with the state using just one InvMixColumn. The SubBytes() (S-box) transformation, which consists of a multiplicative inversion over $GF(2^8)$ and an affine transform, is the most critical part in the AES algorithm, so far as computational complexity is concerned.

The S-box operation is required both for the encryption and key expansion. The S-box dominates the hardware complexity of the AES circuit. Conventionally, the coefficients of the S-box and inverse S-box are stored in LUTs or a hard-wired multiplicative inverter over $GF(2^8)$ can be used, together with the affine transform function. The dedicated inverter, however, has a high area overhead.

The Sub Bytes transformation consists of two steps: compute the multiplicative inverse of each byte in $GF(2^8)$ and then apply an affine transformation. Denoting each byte by S, S is an element of the Galois field $GF(2^8)$ and the Sub Bytes can be described by

$$S'=MS-1+c \quad (1)$$

where M is an 8X8 matrix and C is an 8-bit vector. In traditional look up table (LUT) approaches, the unbreakable delay is longer than the total delay of the rest of operations in each round. Speed up by pipelining and sub pipelining will be unachievable for this feature. Furthermore, the LUT approach is not suitable for resource constrained use for it costs a large area. Composite field arithmetic has been introduced to solve the problem. The multiplicative inverse in $GF(2^8)$ is very complicated by direct computation. However, two fields of the same order are isomorphic. This gives us the possibility to use an isomorphic transform to convert

$GF(2) = GF(2^2)$: $P_o(x)=x^2+x+1$
$GF(2^2) = GF((2^2)^2)$: $P_1(x)= x^2+x+ \phi$, $\phi \varepsilon GF(2^2)$
$GF((2^2)^2) =GF(((2^2)^2)^2)$: $P_2(x) =x^2+x+\lambda$, $\lambda \varepsilon GF((2^2)^2)$

Where the values of ϕ and λ satisfy $P_1(x)$ and $P_2(x)$ are irreducible over $GF(2^2)$ and $GF((2^2)^2)$ respectively.

There are two options for ϕ and for each t ϕ there are 8 options for λ to satisfy the requirement above. Different polynomial coefficients will severely affect the complexities of operations in the field. Furthermore, for a fixed set of polynomial coefficients, there exist 8 isomorphic mappings. Thus, polynomial coefficients and isomorphic mappings should all be taken into consideration to minimize the gate count and short path. The isomorphic mappings that convert between $GF(2^8)$ and $GF(((2^2)^2)^2)$ can be merged into one 16 X 8 matrix instead of traditional two 8 X 8 matrices. This gives us more freedom to use substructure sharing and can get smaller area.

Optimal irreducible polynomial coefficients and isomorphic mappings are selected using standard and normal bases respectively. A 16X 8 matrix is used instead of two 8 X 8 matrices as in previous work to obtain more freedom to use substructure sharing.

The MixColumn transformation operates on the four bytes of each column of the state matrix. The columns are considered as polynomials over $GF(2^8)$ and multiplied modulo $X^4 + 1$ with a fixed polynomial a(x), given by (3)
$a(x)=\{03\}_{16}X^3+\{0l\}_{16}x^2+\{0l\}_{16}X+\{02\}_{16}$ (3)

In the InvMixColumn transformation the fixed polynomial $a^{-1}(x)$ is given by
$a^{-1}(x)=\{0b\}_{16}X^3+\{0d\}_{16}X^2+\{09\}_{16}X+\{0e\}_{16}$ (4)

To minimize gate count and consider that the complexity of InvMixColumn is higher than that of MixColumn, the InvMixColumn transformation can be decomposed to share resource with MixColumn. The polynomial $a^{-1}(x)$ can be decomposed as in (5)
$a^{-1}(x)=a^3(x)=a(x)\cdot d(x)$ (5)
$d(x)=a^2(x)=\{04\}_{16}X^2+\{05\}_{16}=\{04\}_{16}(X^2+1)+\{0l\}_{16}$

The on-the-fly key expansion can generate a round key per clock without additional memory to store the keys. AES with 128-, 192- and 256-bit key can all be handled, and both encryption and decryption are supported. To cooperate with the AES architecture, one 128-bit round key should be generated in each clock. According to AES key expansion algorithm, we can get a 192- or 256-bit round key per clock, however, it is incompatible with the width of AES data block. A data shuffling multiplexer is introduced to rearrange the round key to generate the next round key. The control signal sel0,….. ,sel6 is produced using a finite state machine (FSM).

3. Project Objectives

The main objectives of the proposed system are,

- To implement AES algorithm supporting both encryption and decryption.
- Doing encryption and decryption using three different cipher keys (128-, 192,256-bit).
- Implementation with minimum gate count in Sub Bytes/InvSub Bytes with the help of the optimum irreducible polynomial coefficients.
- To improve throughput by applying a novel on-the-fly key expansion structure.
- Obtain a cost effective implementation of AES with enough security.

4. Project Scope

- For security and fast transmission of data over an insecure path, cryptographic method can be used. Here AES implementation gives an efficient architecture with shorter critical path and smaller area. The AES algorithm has broad applications, including smartcard, cellular phone, WWW servers, automated teller machine (ATMs) and digital video recorders.
- Compared to the software implementation, hardware implementations of the AES algorithm more physical security as well as higher speed. There are a number of areas seeking even lower area designs for block cipher such as the AES in consumer electronics, for example mobile communications, which require modest data rates of the order of 1 Mbps.
- A low cost and small design of AES algorithm can be used in smart card applications, which allow a wide range of equipment to operate securely.

5. Result analysis

The simulation results are carried out for fully pipelined AES to show the efficiency and throughput. Result synthesized and simulated using Xilinx ISE8.2i and ModelSim6.3f software's. Efficiency and throughput of fully pipelined architecture is given in below table.

Table 1: Execution time and Throughput

Function	Average Execution Time (µs)	Throughput (Mbit/s)
Key Expansion Cipher	4.71	27.18
Cipher	16.50	**7.76**
Inverse Cipher	20.56	6.23

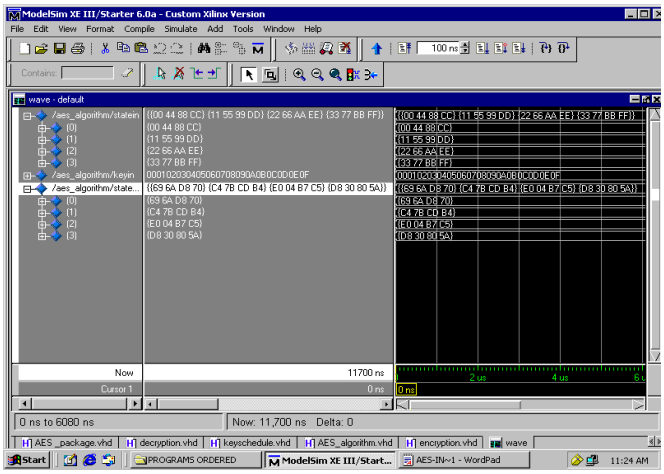

Figure 4: Encryption of 128 AES key

Figure 5: Decryption of 128 AES key

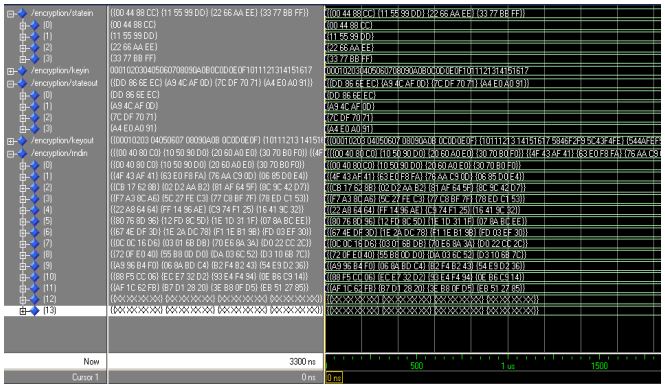

Figure 6: Encryption of 192 AES key

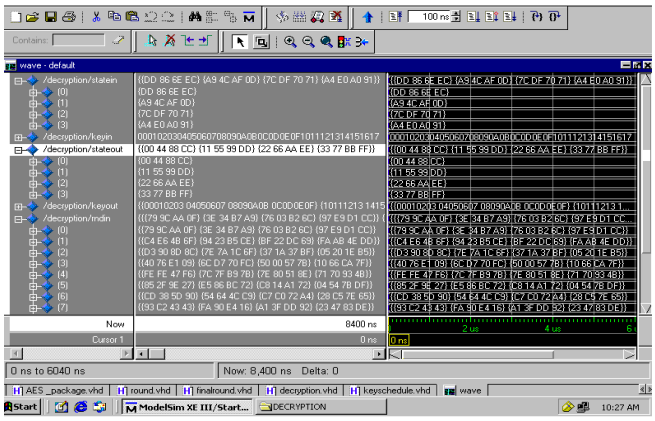

Figure7: Decryption of 192 AES key

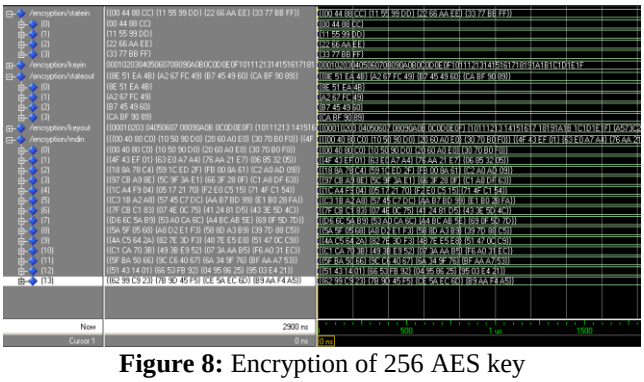

Figure 8: Encryption of 256 AES key

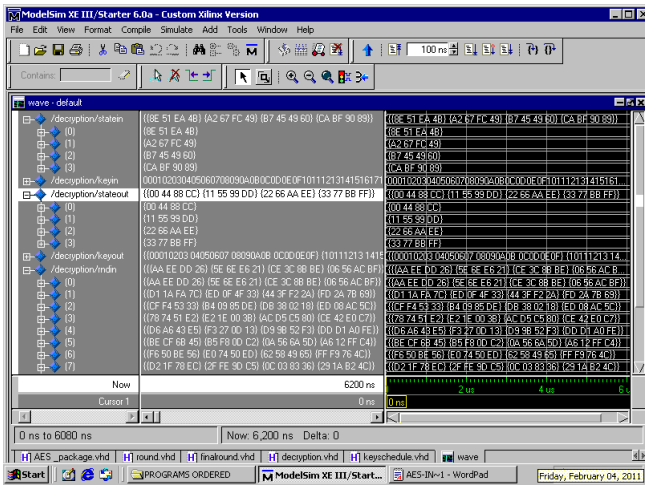

Figure 9: Decryption of 256 AES key

6. Future Scope

In this project, there are 16 ways to construct the composite field GF from 16 sets of different irreducible polynomials with standard and normal bases. If we can make availability of these polynomial more and differently, we can reduce the area further. This architecture can easily applied using pipelining or sub pipelining to get higher throughput. If we can optimize the architecture of on-the-fly key with three different keys and both encryption and decryption can handled, which can generate more than a 128-bit key in one cycle.

7. Conclusion

The proposed design is an implementation with a high throughput per kilo gates parameter which can perform encryption and decryption and support all the key sizes. A new architecture is proposed that only one MixColumn/InvMixColumn copy is used instead of two copies in prior works. The SubBytes is smaller than prior works. Here get a smallest MixColumn/InvMixColumn as already know. Furthermore, the architecture can be easily applied using pipelining or sub pipelining to get higher throughput. The design is quite efficient for it has the highest parameter throughput per kilo gate count as know with the same process.

References

[1] Advanced Encryption Standard (AES), FIPS PUB 197, Nov. 26, 2001, Federal Information Processing Standards publication 197.
[2] X. Zhang , K. K. Parhi, "High-speed VLSI architectures for the AES algorithm", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.12 n.9, p.957-967, September 2004.
[3] X. Zhang, K. K. Parhi, "On the Optimum Constructions of composite field for the AES Algorithm", IEEE Transactions on Circuits. & Systems-II, Vo1.53, No.10, Oct.2006.
[4] D. Canright, "A very compact S-box for AES," in Proc. Cryptographic Hardware and Embedded Syst., Edinburgh, U.K., Sep. 2005, pp. 441455.
[5] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with S-box optimization," in Proc. ASIACRYPT, Gold Coast, Australia, Dec. 2000, pp. 239-254.
[6] Y. Huang, Y. Lin, K. Hung and K. Lin, "Efficient Implementation of AES IP," Circuits and Systems2006, APCCAS IEEE Conference, pp. 1418-1421,2006.
[7] M. Alam, S. Ray, D. Mukhopadhyay, S. Ghosh, D. Roychowdhury and I. Sengupta: "An Area Optimized Reconfigurable Encryptor for AES Rijndael", in the proceeding of Design, Automation and Test in Europe (DATE 2007), pp.1116-1121, April 16-21, Nice, France.
[8] N.Mentens, L.Batina, B.Preneel, I.Verbauwhede. "A systematic evaluation of compact hardware implementations for the Rijndael Sbox", in CT-RSA 2005, LNCS 3376, pp. 323-333, 2005.
[9] J.Wolkerstorfer, E.Oswald, M.Lamberger. "An ASIC Implementation of the AES SBoxes", Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology, p.67-78, February 18-22, 2002.
[10] I. Verbauwhede, P. Schaumont, and H. Kuo, "Design and Performance Testing of a 2.29 Gb/s Rijndael Processor," IEEE 1. Solid-State Circuits (JSSC), Mar. 2003.
[11] H. Kuo, and I. Verbauwhede."Architecture optimization for a 1.82Gbit/s VLSI implementation of the AES Rijndael algorithm".Proc. 3rd Int.CHES 2001, May 2001, pp. 51 64.
[12] H. Shim, D. W. Kim, Y. K. Kang, T. W. Kwon, 1. R.. Choi, "A Rijndael crypto processor using shared on-the-fly key scheduler", IEEE Asia-Pacific Conference on ASIC, pp. 89-92, Aug. 6-8,2002.
[13] Kenneth Stevens, Otmane Ait Mohamed, "Single-chip FPGA Implementation of a Pipelined, Memory-Based AES Rijndael Encryption Design", IEEE ECE2005,pp. 1296-1299 Paper.
[14] X. Zhang and K. K. Parhi, "Implementation approaches for the advanced encryption standard algorithm," IEEE Circuits Syst. Mag., vol. 2, no. 4, pp. 24–46, 2002.
[15] A. Aziz and N. Ikram, "Memory efficient implementation of AES S-boxes on FPGA", Journal of Circuits, Systems, and Computers, Vol. 16, No. 4, pp. 603-611, 2007.