M.Gnanambika, S.Adilakshmi, Dr.Fazal Noorbasha / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 2, March -April 2013, pp.166-169

AES-128 Bit Algorithm Using Fully Pipelined Architecture for Secret Communication

M.Gnanambika*, S.Adilakshmi,Dr.Fazal Noorbasha*****,**
*(M.Tech VLSI Student, Department ECE, KL University, Guntur, A.P, India)
** (Assistant Professor ,Department of ECE , KL University, Guntur, A.P, India)
*** (Associate Professor, Department of ECE, KL University, Guntur, A.P, India)

ABSTRACT

In this paper, an efficient method for high speed hardware implementation of AES algorithm is presented. So far, many implementations of AES have been proposed, for various goals that effect the Sub Byte transformation in various ways. These methods of implementation are based on combinational logic and are done in polynomial bases. In the proposed architecture, it is done by using composite field arithmetic in normal bases. In addition, efficient key expansion architecture suitable for 6 sub pipelined round units is also presented. These designs were described using VerilogHDL, simulated using Modelsim.

Keywords - AES, VLSI Cryptosystems, Encryption, Decryption, Block Cipher, Encipher, Decipher.

I. INTRODUCTION

Cryptography plays an important role in the security of data transmission. The development of computing technology imposes stronger requirements on the cryptography schemes. The Data Encryption Standard (DES) has been the U.S. government standard since 1977. However, now, it can be cracked quickly and inexpensively. In 2000, the Advanced Encryption Standard (AES) replaced the DES to meet the ever-increasing requirements for security. In cryptography, the AES, also called 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 .The Rijndael algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data into 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 supports keys length of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits ,thus the name AES-128, AES-192 and AES-256 respectively. The hardware implementation of the AES algorithm can provide high performance, low cost for specific applications and reliability compared to its software counterparts.

The AES algorithm has broad applications, such as smart cards and cell phones, WWW servers, fiber network, satellite communication, and digital video recorders. Considering the variety of presented application, different requirement and limitation of each of them, implementation of different structure are noticed by researchers. Regarding this, numerous architectures have been proposed for the hardware implementations of the AES algorithm [3] – [7]. Among these architectures, the design in can achieve the highest speed while it is more efficient than the prior designs. The key idea in is to employ composite field arithmetic in the computation of the multiplicative inversion in the *SubBytes*/*Inv SubBytes* transformation of the AES algorithm. As a result, deep sub pipelining is enabled, and hardware complexity is reduced. In the design in [9] $GF(2^8)$ is decomposed into $GF((2^4)^2)$, and composite field arithmetic is applied to all the transformations in the AES algorithm. The optimum construction scheme for $GF((2^4)^2)$ is selected based on minimizing the total gate count in the implementation of all transformations. However, it is more efficient to apply composite field arithmetic only in the computation of the multiplicative inversion in the *SubBytes* and *InvSubBytes* transformations. In this case, the construction scheme selected in is no longer optimum. In $GF(2^8)$ is decomposed into $GF((2^4)^2)$, while in [10], $GF(2^8)$ is decomposed into $GF(((2^2)^2)^2)$. Nevertheless, each of them proposed only one possible way to construct the composite field. There exist other construction schemes with smaller gate counts and shorter critical paths. In applications that the rate of receive and transmit data are very high, using of high speed hardware encryptors are usual. In these encryptors, three techniques of loop-unrolling, pipelining and subpipelining are used for increase of output bitrates. From these techniques, the third one has the maximum influence on the increment of encryptor throughput and causes the more optimization in throughput/area ratio. In this paper, subpipelining method used with resource sharing technique are used for area and throughput optimization of encryptor hardware. In prior designs, the implementation of *Sub Byte* transformation is done in polynomial bases but in this design normal bases form is used for this purpose.

II. RIJNDAEL: THE ADVACED ENCRYPTION STANDARD

The Rijndael algorithm is composed of *Nr* times repetition of four transformations *Sub Byte*, *Shift Row*, *Mix Column*, *AddRoundKey* in each round of algorithm except the last one. Fig. 1 shows the AES encryption process for a 128- bit secret key. The value of *Nr* for key length of 128-bit is equal to ten. Each round and the initial stage requires a 128- bit round key, and thus 11 sets of round keys are generated from the secret key. In the AES algorithm, the plain text data string is divided into blocks of 128 bits. Each block is divided into 16 bytes (4 × 4 matrix of bytes), and each byte is considered as an element of $GF(2^8)$.

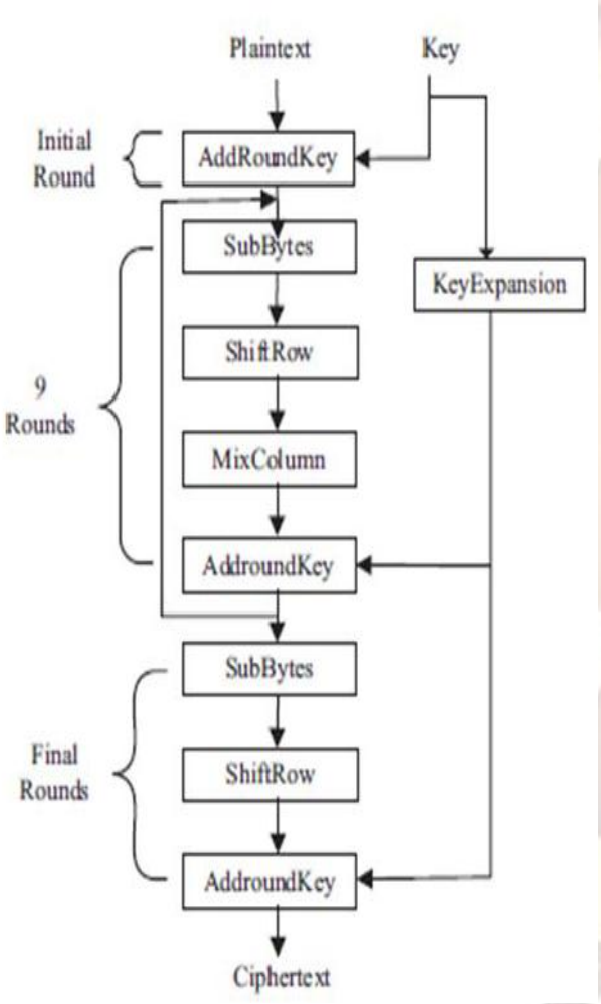

FIG 1.The AES-128 Encryption Algorithm

Although different irreducible polynomials can be used to construct $GF(28)$, the one specified for the AES algorithm is shown in (1).

$$m(x) = x^8 + x^4 + x^3 + x + 1 \qquad (1)$$

The primitive functions SubBytes, ShiftRows and Mix- Columns are based on byte-oriented arithmetic, and AddRoundKey is a simple 128-bitwise XOR operation. *SubBytes* is a nonlinear transformation that uses 16 byte substitution tables (S-Boxes). An S-Box is the multiplicative inverse of a Galois field $GF(2^8)$ followed by an affine transformation. In the decryption process, the affine transformation is executed prior to the inversion. $m(x)$ is the irreducible polynomial used by a Rijndael S-Box. *Shift Rows* is a cyclic shift operation of the last three rows by different offsets. *Mix Columns* treats the 4-byte data in each column as coefficients of a 4-term polynomial, and multiplies the data modulo $x^4 + 1$ with the fixed polynomial given by

$$c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \qquad (2)$$

In the decryption process, *Inv Mix Columns* multiplies each
column with the polynomial

$$c{-}1(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\} \qquad (3)$$

and *Inv Shift Rows* shifts the last three rows in the opposite direction from *Shift Rows*.

III. IMPLEMENTATION OF FULL PIPELINE AES

The key step in the implementation of this algorithm is *SubByte* transformation, the only nonlinear step in each round of encryption algorithm. Consideration on high delay and required gates of this transformation, different implementation methods are presented. The most of traditionally hardware implementation of this transformation are done with using of Look Up Tables (LUT) [3] – [5] . Using of LUTs in high speed applications has two main problems. The first one is the high volume of required gates for implementation of this transformation. The second one is inherent and unbreakable existent delay in this LUT structures. This delay is longer than the total delay of the rest of the transformations in each round unit and prohibits each round unit from being divided into more than two substages to achieve any further speedup. Non-LUT-based approaches, which employ combinational logic only, can be used to avoid the unbreakable delay of LUTs. However, these approaches involve inversion in Galois Field $GF(2^8)$, which may have high hardware complexities. Therefore, composite field arithmetic is presented, in which the field operations are implemented in lower order fields such as $GF(2^4)$ or $GF((2^2)^2)$ and by lower cost subfield operations. In the *SubByte* transformation is implemented with using composite field arithmetic and in polynomial bases. In , this implementation is done by using Composite field arithmetic in normal bases. The main advantages of the new bases than to older one

is that it doesn't require any hardware for implementation of square operation.

The subpipelined architecture can achieve maximum speed up if each round unit is divided into more substages with equal delay. By increment the number of these substages, the critical path and clock pulse width of system can be decreased and as a result the throughput is increased. Fig. 2 shows a full pipeline AES encryptor with 6 subpipeline stages and key length of 128-bit.

IV. IMPLEMENTATION OF KEY EXPANSION

The round keys that are used in each encryption round unit, can be either generated beforehand and stored in memory or generated on the fly. In applications that key variation is required during encryption operation, the second approach is suitable. If we want the round keys generation to be done simultaneously with encryption operation, it must be considered that the key expansion architecture is divided the same as the number of existent sub stages in encryption round unit . Fig. 3 shows the key expansion architecture suitable for 6 substages subpipelined AES algorithm with the key length of 128-bit.

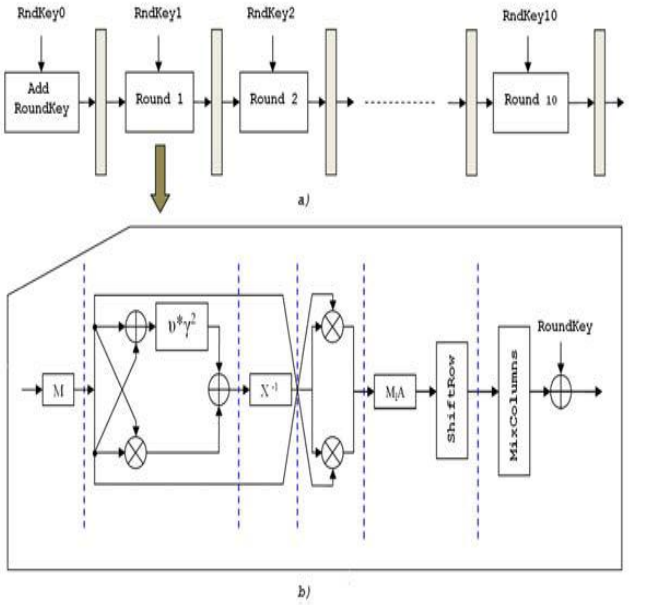

FIG. 2. The AES algorithm a) Fully pipelined architecture b) The subpipelined composite field architecture of round unit

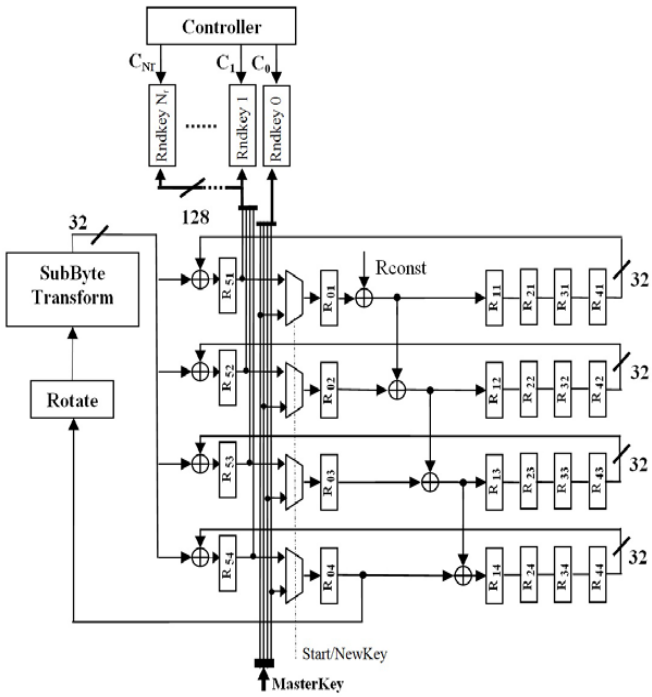

FIG. 3. The key expansion architecture for 6 substages subpipelined AES algorithm with 128-bit key

Assuming the same subpipelined *SubByte* transformation is used in the key expansion unit, the remaining path of this unit has one 2-to-1 Mux and 1 Xor gate. The 2-to-1 Mux operation is executed simultaneously with first substage of encryption round unit and the Xor operation [12] is executed simultaneously with 6th substage of encryption round unit after *MIA* operation. The proposed key expansion architecture is capable of generating all required $Nr + 1$ round keys after $6 \times Nr$ clock cycle. The roundkey i is available at output of $R5J$ -column ($(1 \leq J \leq 4)$) registers at clock cycle $6 \times I$ ($0 \leq i \leq Nr$) and at the moment, with activation Ci signal, it is loaded into $RndKey(i)$ registers. These round keys are held there for using in the entire encryption process. Also, the Xor operation of *Rconst* with each row of input state matrix are executed simultaneously with second sub stage of key expansion unit.

V. SIMULATION RESULTS

The simulation results are shown in figures 4 and 5. Here ,in the iterative method the output is obtained in single clock run. These are simulated in Xilinx ISE 9.1i. The outputs are verified in behavioral model. In this the same key is given in encryption and decryption. The cipher text of encryption is given to plain text in decryption.

The simulation results of encryption and decryption is shown below as Fig 4 & 5.

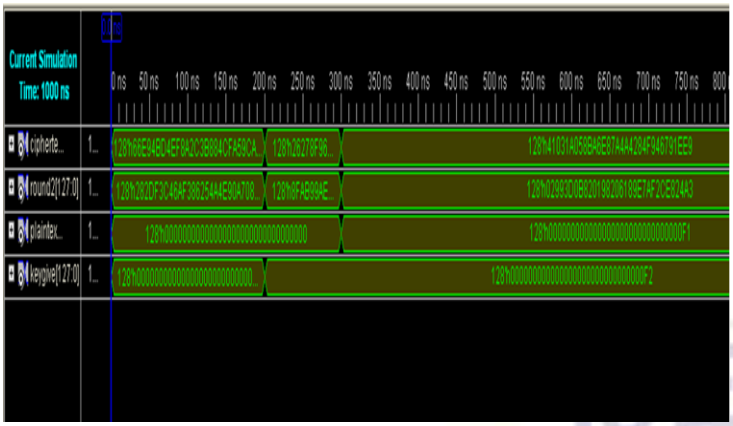

FIG 4 Encryption Result

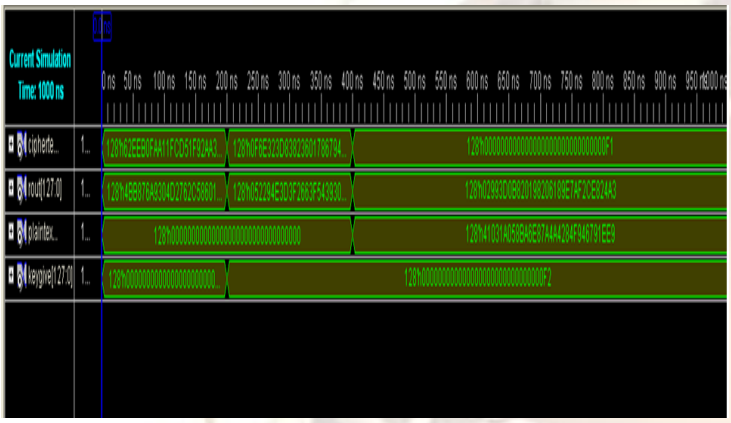

FIG 5 Decryption Result

VI . CONCLUSION

In this paper, an efficient subpipelined architecture of AES algorithm with its key expansion unit is presented. The key expansion architecture is suitable for 6 substages sub pipelined AES architecture Against prior implementations, this architecture uses composite field arithmetic in normal bases representation to reduce the required hardware.

REFERENCES

[1] National Institute of Standards and Technology (NIST), Information Technology Laboratory (ITL), *Advanced Encryption Standard (AES)*, Federal Information Processing Standards (FIPS) Publication 197, November 2001

[2] X. Zhang and K. K. Parhi, *On the Optimum Constructions of Composite Field for the AES Algorithm*, IEEE Transactions on Circuits and Systems-II: Express Briefs, VOL. 53, NO. 10, OCTOBER 2006.

[3] V. Fischer and M. Drutarovsky, *Two methods of Rijndael mplementation in reconfigurable hardware*, in Proc. CHES 2001, Paris, France, May 2001, pp. 77-92.

[4] H. Kuo and I. Verbauwhede, *Architectural optimization for a 1.82 Gbits/sec VLSI implementation of the AES Rijndael algorithm*, in Proc. CHES 2001, Paris, France, May 2001, pp. 51-64.

[5] M. McLoone and J. V. McCanny, *Rijndael FPGA implementation utilizing look-up tables*, in IEEEWorkshop on Signal Processing Systems,Sept. 2001, pp. 349-360.

[6] V. Rijmen, *Efficient Implementation of the Rijndael S-box*, 2000. Available online at www.iaik.tugraz.at/RESEARCH/krypto /AES/old / rijmen/ rijndael/sbox.pdf.

[7] A. Satoh, S. Morioka, K. Takano, S. Munetoh, *A Compact Rijndael Hardware Architecture with S-Box Optimization*, Proceedings of ASIACRYPT 2001, LNCS Vol.2248, pp. 239 - 254, Springer-Verlag, December 2001.

[8] N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, *A Systematic Evaluation of Compact Hardware Implementations for the Rijndael SBox*, In Alfred Menezes, editor, CT-RSA, volume 3376 of LNCS, pages323-333. Springer, 2005.

[9] A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi, *Efficient implementation of Rijndael encryption with composite field arithmetic*, in Proc. CHES, Paris, France, May 2001, pp. 171-184.

[10] J.Wolkerstorfer, E. Oswald, and M. Lamberger, *An ASIC implementation of the AES S-boxes*, in Proc. RSA Conf., San Jose, CA, Feb.2002, pp. 67-78.

[11] X. Zhang, K. K. Parhi, *High-speed VLSI architectures for the AES algorithm*, IEEE Trans. VLSI Systems, Vol. 12, Iss. 9, pp. 957 - 967, Sept. 2004.

[12] G. P. Saggese, A. Mazzeo, N. Mazocca, and A. G. M. Strollo, , *An FPGA based performance analysis of the unrolling, tiling and pipelining of the AES algorithm*, Proc. FPL 2003, Portugal, Sept. 2003.