International Conference of High Performance Computing (HiPC 2005)

Design of an Efficient Architecture for Advanced Encryption Standard Algorithm Using Systolic Structures

[1]Suresh Sharma, [2]T S B Sudarshan
[1]Student, Computer Science & Engineering, IIT, Khragpur
[2]Assistant Professor, Computer Science & Information Systems, BITS, Pilani.

Abstract:
This Paper presents a systolic architecture for Advanced Encryption standard (AES). Use of systolic architecture has improved the hardware complexity and the rate of encryption/decryption. Similarities of encryption and decryption are used to provide a high performance using an efficient architecture. The efficiency of the design is quite high due to use of short and balanced combinational paths in the design. The encryption or decryption rate is 3.2-Bits per clock-cycle and due to the use of pipelined systolic architecture and balanced combinational paths maximum clock frequency for the design is quite high.

Index terms:- Advanced Encryption standard (AES), Systolic Architecture, processing elements, regularity, combinational paths.

1. Introduction:-

Advanced Encryption standard (AES) [1,12,13] is successor of Data Encryption Standard (DES) [2,12,13]. A symmetric block cipher Rijndael [3] was standardized by National Institute of Standards and Technology (NIST) as AES in November 2001. Due to practical importance of hardware implementation, the different candidates are implemented on FPGAs [4,5,6,9] and on ASICs [7,8,10]. This paper presents a simple and regular hardware architecture based on systolic architecture [11] to provide a throughput of 3.2 Bits per clock cycle for AES-128 encryption and decryption.
Systolic Architecture is used for constructing high-speed, special- purpose devices. In a systolic system, data flows from the computer memory in a rhythmic fashion, passing through many processing elements before it returns to memory .A systolic system consists of a set of interconnected cells each capable of performing some simple operations, cells in a systolic system are typically connected to form a systolic array or a systolic tree. Information in a systolic system flows between cells in a pipelined fashion and communication with the memory or external devices occurs only through the boundary cells.
The architecture uses similarities of encryption and decryption to provide a high level of performance while keeping the chip size small. The architecture is highly regular and scalable. The key size can easily be changed from 128 to 192 or 256 bits by making very few changes in the design.
The paper is organized as follows. Section 2 gives a brief overview of the AES algorithm. In section 3, a summary of available AES architecture is given and the proposed AES hardware architecture is described. The performance of the architecture is compared with other implementations in section 4. Concluding remarks are given in section 5.

2. AES Algorithm:

The AES takes a 128-bits data block as input and performs several transformations to encrypt or decrypt the data. In case of encryption, the input block is called plaintext and the returned block is called ciphertext. All intermediate blocks are called states. These are represented as two-dimensional array of bytes. AES encryption and decryption are based on four transformations that are performed repeatedly in a certain sequence. The number of repetitions is based on the key size.

2.1. AddRoundKey transformation:-

In the AddRoundKey transformation, a Round Key is added to the State by a simple bitwise XOR operation. The AddRoundKey transformation is self-inverting.

2.2. SubBytes transformation:-

The SubBytes() transformation is a non-linear byte substitution that operates independently on each byte of the state using a substitution table (S-box). This S-box, which is invertible, is constructed by composing two transformations:

1. Take the multiplicative inverse in the Galois Field $GF(2^8)$ with the irreducible polynomial $m(x)= x^8+x^4+x^3+x+1$. The element {00} is mapped to itself.
2. Apply the affine transformation (over GF(2)):

The inverse of SubBytes transformation, which is needed for decryption, is the inverse of the affine transformation followed by the same inversion as the SubBytes transformation

2.3. ShiftRows transformation:-

The ShiftRows transformation rotates each row of the input state to the left; the offset of the rotation corresponds to the row number. The inverse of this transformation is computed by performing the corresponding rotations to the right.

2.4. MixColumns transformation:-

The MixColumns transformation operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over $GF(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial a(x), given by

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} .$$

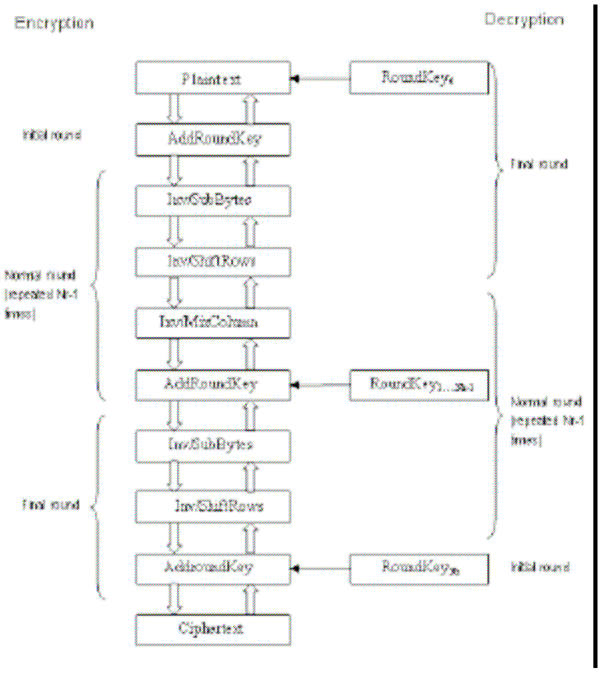

Fig.1: Sequence of the execution of different transformations used in AES

The coefficients of a(x) are also elements of $GF(2^8)$ and are represented by the hexadecimal values in this equation. The inverse MixColumn Transformation is the multiplication of each column with

$$a^{-1}(x) = \{0B\}x^3 + \{0D\}x^2 +\{09\}x +\{0E\} \text{ modulo } x^4 +1$$

These transformations are applied in a certain sequence to perform encryption or decryption as shown in Fig.1. In both cases, the transformations are grouped to so-called rounds. The number of rounds depends on the key size. There are three different kinds of rounds: the initial round, the normal round and the final round.
For the decryption, the same round keys are used in the reversed order.

3. Proposed AES Hardware Architecture:-

Due to practical importance of hardware implementation, the different candidates are implemented on FGPAs [4,5,6,9] and on ASICs [7,8,10]. In [17] a cryptographic hardware module for an AES algorithm is described that provides complete protection against first order differential power analysis by embedding a data masking countermeasure at a hardware level. It uses a fully unfolded implementation that executes one round of encryption/decryption per cycle. In [8] a compact and high-speed hardware architectures and logic optimization methods for the AES algorithm Rijndael are described. Encryption and decryption data paths are combined and all arithmetic components are reused. By introducing a new composite field, the S-Box structure is also optimized.
In [16] all the transformations are made by data unit whereas for generating keys a separate unit is used. Also resource sharing is done between encryption and decryption, and also between key expansion and the computations of AES rounds.
The AES hardware module consists of the following four components: -

3.1 I/O Interface

The I/O Interface is used to make the data communication with the external devices or memory. It communicates 32-bit words in a clock cycle with the other components and to the external devices using the bus.

3.2 Data Unit

Data Unit performs all the AES encryption or decryption rounds. There is no need to change the design of the data unit for different key sizes.

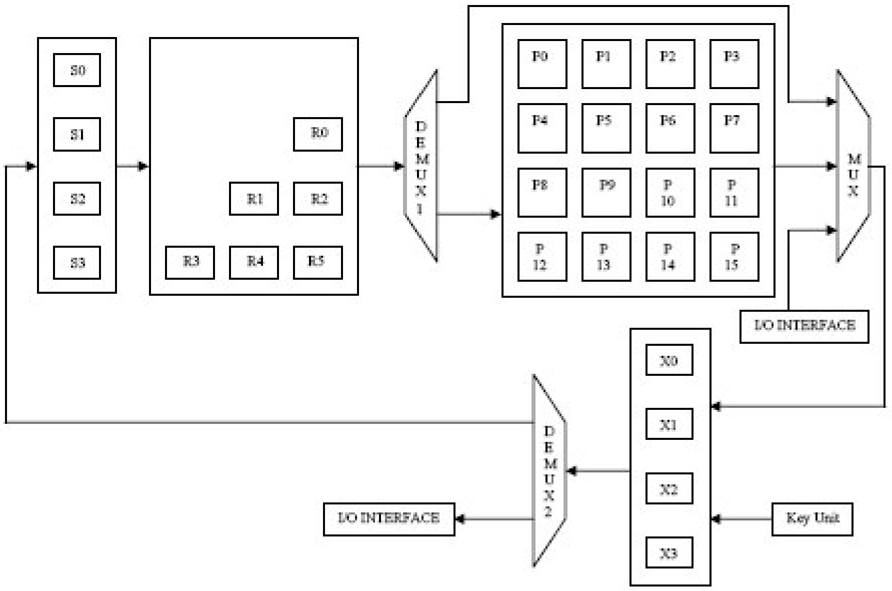

Fig.2: Design for AES Data Unit

The design of data unit is to perform all four transformations. The detailed circuit for data unit is given in the Fig 2. Also the detailed design for all the transformations has been explained below:

3.2.1 SubBytes Transformation:

In the proposed architecture, a pipelined implementation of AES S-Box is used [15]. The architecture of the AES S-Box is shown in fig 3. When it gets the input it is stored into the register Res and then in next clock cycle the transformation is performed. The fact used for designing combinational logic for S-Box is that GF(2^8) can be seen as a quadratic extension of the field GF(2^4). In this case, an element a in GF(2^8) is represented as a linear polynomial with coefficients in GF(2^4), and will be denoted by the pair [a_h, a_l].

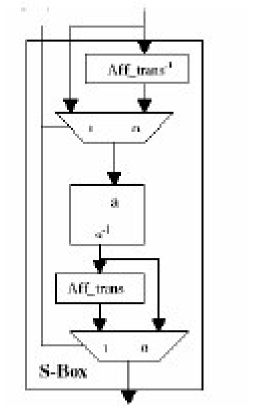

Fig.3: Design for AES S-Box

Both coefficients of such a polynomial have four bits. All mathematical operations applied to elements of GF(2^8) can also be computed in this representation which we call two-term polynomials.

$$a = a_h x + a_l$$

Multiplication and inversion of two-term polynomials require a modular reduction step to ensure that the result is a two-term polynomial too. The irreducible polynomial needed for the modular reduction is given by

$$n(x) = x^2 + \{1\}x + \{e\}.$$

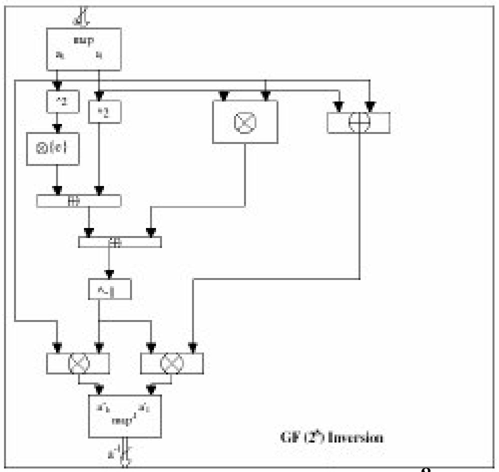

Fig.4: Inversion in GF (2^8)

The coefficients of n(x) are elements in GF (2^4) and are written in hexadecimal notation. Their particular values are chosen to optimize the finite field arithmetic. The design for inversion of two-term polynomials is Fig 4:

3.2.2. Shift Rows Transformation:

For shift rows transformation no separate design have been implemented. In Shift Row Transformation maximum of three shifts are required and by setting the control signals accordingly and using registers R0-R5 we can provide data for MixColumn Transformation in required sequence as shown in Fig-5.
In this representation no separate design is used for the Inverse Shift Rows Transformation because in Inv/shift rows transformation the content of data don’t changes and in SubBytes Transformation order of data remains the same only substitution takes place so sequence for both of these can be changed without effecting the algorithm as
InvShiftRows(InvSubBytes(a)) is equivalent to InvSubBytes(InvShiftRows(a)) .

3.2.3. Mix Column Transformation:

For mix column transformation systolic architecture is used for matrix-matrix multiplication [11] for array size of 4×4. In each processing element a combinational logic for multiplication in GF (2^8) is used and for addition XOR gates are used. Else than this each cell has one register which contains the values required corresponding to the encryption and decryption and one register is used to store the result of the process computation.
In case of systolic design the data flows in the horizontal direction whereas the result of the each process computation flows in the vertical direction and XORed with the previous result. The procedure for entry of data in the systolic design for MixColumn Transformation is shown in the Fig.5, where blank boxes show the delay part of the circuit.

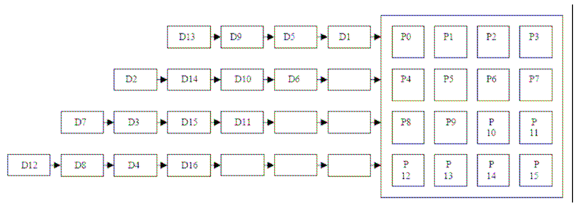

Fig.5: Systolic Matrix-Matrix Multiplication

3.2.4. Add Round key transformation:

For add round key transformation a combination of XOR gates are used.
For flow of data according to the different rounds of the encryption and decryption certain number of MUX and DEMUX are used as shown in the Fig 3.

3.3. Key Unit

The key unit stores the cipher keys and also calculates the round keys. Making small changes in the key unit design it is possible to use it for multiple key sizes. For calculation of AddRound keys no separate S-Boxes are given in the key unit only the S-Boxes of the data unit are used to save the area of and cost of the design.
As data unit don’t use S-Boxes while the MixColumn and AddRoundKey transformation are executed, the S-Boxes of the data unit are reused by the key unit during this clock cycle this save a significant amount of area. The cipher keys are loaded and the round keys are calculated iteratively. For decryption, first a key setup to be done as a decryption uses roundkey10 in the initial round and the cipher key is required in the final. During a key setup, the cipher key is expanded to roundkey10 and roundkey9 to roundkey0 can be obtained iteratively for the decryption.

3.4. Control Unit

For better efficiency of the design a vertical micro programmed control unit is used. The width of the control word will be equal to the total number of the control signals in the circuit. For using the same design only slight changes are to be made in the micro program written into the ROM.

4. Performance:-

Using the above implementation, four clock cycles are required to enter the 128-bit cipher key into the key unit and after that four clock cycles are required for each round of encryption or decryption. Thus using the above implementation, the minimal number of clock cycles required performing encryption or decryption for each data block of 128-bit is 40. Thus the throughput of the design is 3.2-Bits per clock cycle.

4.1 Relative Work:

In the architecture presented [17] a fully unfolded implementation is used, although it increases the speed of the architecture but due to use of 16 Inv/S-Boxes and a large amount of MUX the area as well as the cost of the design increases a lot. For key expansion also separate S-Boxes are used. In [8] the architecture has very unbalanced combinational paths and requires a time and area consuming selector function, which is not part of the actual AES algorithm. The SubBytes, the MixColumn and the AddRoundKey transformation are done for one column within one clock cycle. Additionally, in the same clock cycle, the passes the selector function, which seems to be another major cause of delay. In [16] as 16 different data cells are used to perform transformations so the number of controls increases by a large amount that also increases the overall cost and area.

Taking care of all these in the presented architecture only 4 S-Boxes are used and resource sharing is done between encryption and decryption, and between key expansion and computation of the AES rounds. Combinational paths used are short and balanced. A simple design is used for ShiftRows Transformation so the complexity and time is reduced.

4.2. Comparison in terms of speed

Among other published AES hardware implementation the design of Satoh [8] requires 54 clock cycles to perform an encryption for 128 bits, which leads to a theoretical throughput of 2.37 Bits per clock cycle. The design presented in [16] requires 64 clock cycles to perform an encryption for 128-bits, so the throughput is 2 Bits per clock cycle. Compared to these the encryption or decryption for the present implementation requires 40 clock cycles per128-bits, which leads to theoretical throughput of 3.2 bits per clock cycle. The comparison wrt the number of cycles and throughput of various implementations and our Systolic implementation is as shown in Table-1.

Table-1 Comparison of Systolic Implementation and other implementations to encrypt 128-bit block

Implementation	No of Cycles	Throughput (bits-per-cycle)
Mangard[16]	64	2
Satoh[8]	54	2.37
Systolic	40	3.2

5. Conclusion:

In this paper a highly regular and pipelined AES hardware architecture is presented using the systolic architecture. There is a high level of sharing between encryption and decryption, as well as between the key expansion and the computation of the AES rounds. The used cipher key size and performance of the architecture can be further improved by suitable changes in the design. As shown in Fig 7 and 8 the efficiency is high and also area requirement is less for the present architecture as the combinational paths used are small and balanced, which reduces the delay and glitches in the design.

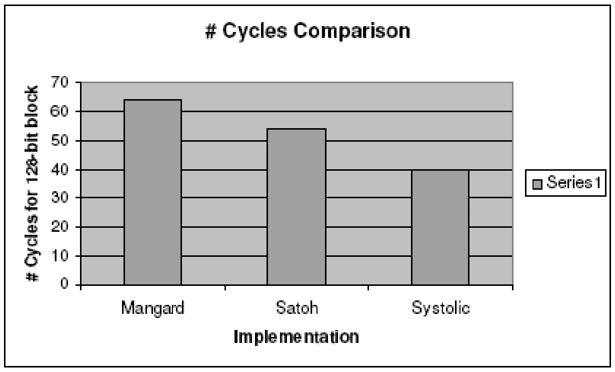

Fig. 7: Comparison of number of cycles.

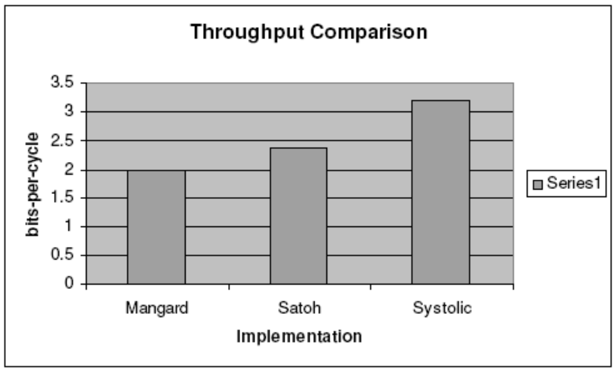

Fig. 8: Comparison of Throughput.

References:

1. National Institute of Standards and Technology (NIST), " Federal Information Processing Standard 197, The Advanced Encryption standard (AES)", http://csrc.nist.gov/publications/fips/fips197/fip s-197.pdf ,2001
2. National Institute of Standards and Technology (NIST), " Federal Information Processing Standard 46-3, Data Encryption Standard (DES)", http://csrc.nist.gov/publications/fips/fips46 ,1999
3. J. Daemen and V. Rijmen, The Design of Rijndael. Springer-Verlag,2002
4. A.J. Elbwirt, W. Yip, B. Chetwynd and C. Paar, "An FGPA Implementation And Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists," Proc. Third Advanced Encryption Standard Candidate Conf., pp. 13-27,2000
5. K. Gaj and P. Chodowic, "Comparison of the Hardware Performance of the AES Candidate Using Reconfigurable Hardware," Proc. Third Advanced Encryption Standard Candidate Conf., pp. 40-56,2000
6. N. weaver and J. wawrzynek, "A Comparison of the AES Candidates Amenability to FPGA Implementation," Proc. Third Advanced Encryption Standard Candidate Conf., pp. ,2000
7. B. Weeks, M. Bean, T. Rozylowicz and C. Ficke, "Hardware Performance Simulation of Round 2 Advanced Encryption standard Algorithms, http://csrc.nist.gov/encryption/aes/round2/NSA-AESfinalreport.pdf ,2000
8. A. Satoh, S.Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," Proc. Advances in Cryptology-ASIACRYPT 2001, pp 239-254, 2001
9. M. McLoone and J.V. McCanny, "High Performance Single-Chip FPGA Rijndael Algorithm Implementation," Proc. Workshop Cryptographic Hardware and Embedded Systems-CHES 2001, pp. 65-76,2001
10. A. Rudra, P.K. Dubey, C.S. Jutla, V. Kumar, J.R. Rao and P. Rastogi, "Efficient Rijndael Encryption Implementation with Composite Field Arithmetic," Proc. Workshop Cryptographic Hardware and Embedded Systems-CHES 2001, pp. 171-184, 2001
11. S.Y. Kung, "VLSI Array Processors," Prentice Hall Inc, 1988
12. William Stallings, "Cryptography and Network Security," Pearson Education, 2004
13. C. Kaufman, R. Perlman, and M. Speciner, "Network Security," Pearson Education, 2002
14. V. Rijmen, "Efficient Implementation of the Rijndael S-Box," http://www.esat.kuleuven.ac.be/rijmen/rijndael/sbox.pdf ,2000
15. J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An ASIC Implementation of the AES S-Boxes," Topic in Cryptology-CT-RSA 2002, Proc. RSA Conf. 2002, Feb. 2002.
16. S. Mangard, M. Aigner, and S. Dominikus, "A Highly Regular and Scalable AES Hardware Architecture," IEEE Transactions on Computers, volume 52 pp.483-491, April 2003
17. Elena Trichina and Tymur Korkishko, "Secure AES Hardware Module for resource Constrained Devices", Lecture Notes in Computer Science, Vol no. 3313 pp 216-230.