Efficient Implementation of AES IP

Yu-Jung Huang, Yang-Shih Lin, Kuang-Yu Hung, Kuo-Chen Lin
Department of Electronic Engineering, I-Shou University
Kaohsiung, Taiwan 80424, ROC
E-mail: yjhuang@isu.edu.tw

Abstract—**The AES IP with different architectures implemented using ASIC and FPGA are presented in this paper. For ASIC design, the performance of the AES IP has been evaluated by comparing its area/power/delay, synthesized with TSMC 0.35 um cell library and TSMC 0.18 um cell library. The performance estimation in FPGA implementation with Altera and Xilinx platforms are also presented. The hardware implementation results of the proposed architecture with Mixcolumn/preprocess InvMixcolumn to perform Mixcolumn/InvMixcolumn transformation has less area cost as compared with previous relevant architecture. Due to the I/O bottlenecks between host processor and a stand alone AES module, a reconfigurable bandwidth sharing architecture is proposed to enhance the system performance.**

Keywords—**Advanced encryption standard, Mixcolumn, FPGA, Galois Field, Cell-Based library**

I. INTRODUCTION

For today's electronic data transactions, digital cryptography has become a de facto standard for many applications such as secure electronic information to protect economic interests, prevent fraud, and guarantee the privacy of individuals. The Rijndael Advanced Encryption Standard (AES) is a block cipher adopted as an encryption standard by the US government and National Institute of Standards and Technology (NIST) as US FIPS PUB 197 in November 2001 after a 5-year standardisation process [1, 2].

Cryptographic transformations can be implemented in both software and hardware [3] – [6]. In software approach, the physical security and achievable speed are limited and cannot acceptable for real applications. In contrast, hardware implementations are considered more physical secure and offer superior performance with higher throughput. The drawback of traditional hardware implementation is lack of flexibility. Field Programmable Gate Arrays (FPGA) devices are highly attractive options for hardware implementations, as they provide reconfigurable flexbility, physical security, and potentially much higher performance than software solutions.

In terms of hardware implementation performance, it is necessary to optimize these AES main operations both in encryption and decryption process. In addition, the host processor which operates together with the AES operations has a significant impact on the hardware performance evaluation. For example, 2x more data per instruction can be achieved in a 64-bit data path than a 32-bit processor. Because of different data path in processor architecture, it results the potential bottlenecks getting the data on/off the AES chip which is basically implemented based on the 128-bit AES algorithm. A reconfigurable bandwidth sharing AES module to further enhance the performance is proposed in this paper.

This paper is organized as follows. In Section 2, the AES algorithm is briefly described. In Section 3, the efficient hardware implementation is introduced. The implementation results of integrated encryptor/decryptor and associated performances and comparisons are given in Section 4, and Section 5 concludes the paper.

II. AES BACKGROUND REVIEW

The original design of Rijndael has a block cipher with a variable block length and a variable key length. The block length and the key length can be independently specified to 128, 192 or 256 bits. The number of rounds in AES depends on the key length. For instance, the proposed number of rounds in AES is 10, 12 and 14 when the key length is 128 bits, 192 bits and 256 bits, respectively [1, 2]. AES uses a 4 x 4 byte representation of the data block for each step performed in a round. There are four distinct invertible transformations in one round , namely: the ByteSub transformation, the ShiftRow transformation, the MixColumn transformation, and the AddRoundKey transformation. Initially, incoming data and key are added together in the AddRoundKey module. After the initial roundkey addition, Nr rounds of encryption are performed. The intermediate transformation result during the round operation is called the state. The state information is then retrieved and the ByteSub, ShiftRow, MixColumn and AddRoundKey are performed on it in the specified order. The first Nr – 1 rounds are the same. Each round involves these four steps, except the last round which skips the MixColumn operation. All four transformations described above have corresponding inverse operations such that the decryption is simply the reverse order operations of these inverse transformations.

The ByteSub Transformation is a byte substitution operated on each of the State bytes using a look-up table (S-box) independently. The operation of the S-box is constructed of the compositions of two transformations: multiplicative inverse in the Galois field GF(2^8) with the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$ and an affine mapping

1-4244-0387-1/06/$20.00 ©2006 IEEE

over GF(2) transformation. The affine transformation can be expressed as

$$b_i^{'} = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus \{63\}_{16} \quad (1)$$

The inverse of ByteSub is a byte substitution using the inverse table. The inverse affine transformation is

$$b_i^{'} = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus \{5\}_{16} \quad (2)$$

where b_i represent the ith bit of one byte. The ShiftRow transformation shifts the rows of the State cyclically by a row dependent amount. Depending on the row location, offset of left shift varies from zero, one, two and three bytes for row 1, 2, 3, and 4 in the corresponding S-box, respectively. In the InvShiftRows, the rows are cyclically shifted to the right by the same offset as that in the ShiftRows. The MixColumn transformation operates on the four bytes in each column of the State. Each column is considered as the coefficients of a polynomial over GF(2^8) and multiplied by modulo x^4+1 with a fixed polynomial p(x), where

$$p(x) = \{03\}_{16} \bullet x^3 + \{01\}_{16} \bullet x^2 + \{01\}_{16} \bullet x + \{02\}_{16} \quad (3)$$

In the InvMixColumns transformation, every column is transformed by multiplying it with polynomial $p^{-1}(x)$, where

$$p^{-1}(x) = \{0b\}_{16} \bullet x^3 + \{0d\}_{16} \bullet x^2 + \{09\}_{16} \bullet x + \{0e\}_{16} \quad (4)$$

In Round Key addition, a Round Key is applied to the State by a bitwise XOR operation

III. AES MAIN OPERATION

As shown in Eqs. (1) and (2), the affine and inverse affine transformation are series of XOR operation, the ByteSub and InvByteSub can be integrated to reduce the hardware expense. The architecture of the integrated ByteSub and InvByteSub is shown in Fig. 1. The S-box is shared for both encryption and decryption. The effectively utilizing the ByteSub operation saves substantial amount of area.

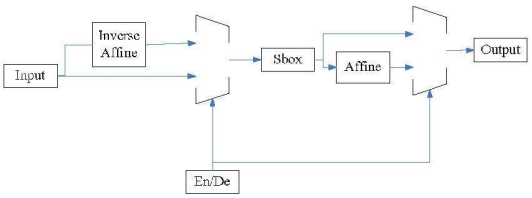

Fig. 1. Integrated ByteSub and InvByteSub architecture

The coefficients of $p^{-1}(x)$ in Eq. (4) are more complex than those of p(x) in Eq. (3). It indicate that the hardware complexity of InvMixColumn transformation is higher than that of MixColumn transformation. In order to reduce design complexity, the InvMixColumn can be decomposed to share hardware resources with MixColumn. For parallel InvMixColumn decomposition:

$$p^{-1}(x) = P(x) + e(x) \quad (5)$$

$$e(x) = \{08\}_{16} \bullet (x^3 + x) + \{0c\}_{16} \bullet (x^2 + 1) \quad (6)$$

For Serial InvMixColumn Decomposition:

$$p^{-1}(x) = p(x) \bullet d(x) \quad (7)$$

$$d(x) = p^{-2}(x) = \{04\}_{16} \bullet x^2 + \{05\}_{16} \quad (8)$$

As shown in the above equations, serial decomposition leads to an efficient multiplier implementation because two of its coefficients in d(x) are zero.

The multiplication process by a term x of the polynomial is a one-bit shift to the left, and an addition by the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$, if the degree of the new polynomial is greater than 7. For some $A \in GF(2^8)$ denotes by the bit string $B_7B_6B_5B_4B_3B_2B_1B_0$, the resulting multiplication can be expressed as

$$\begin{aligned} &Xtime(A) = A \bullet x \bmod m(x) \\ &= A_7x^8 + A_6x^7 + A_5x^6 + A_4x^5 + A_3x^4 + A_2x^3 + A_1x^2 + A_0x \\ &\quad \bmod A_7x^8 + A_7x^4 + A_7x^3 + A_7x + A_7 \\ &= A_6x^7 + A_5x^6 + A_4x^5 + (A_3 \oplus A_7)x^4 + (A_2 \oplus A_7)x^3 + A_1x^2 \\ &\quad + (A_0 \oplus A_7)x + A_7 \\ &= B_7x^7 + B_6x^6 + B_5x^5 + B_4x^4 + B_3x^3 + B_2x^2 + B_1x + B_0 \end{aligned} \quad (9)$$

The basic Xtime module is shown in Fig. 2.

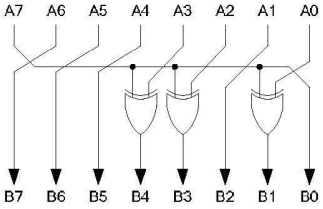

Fig. 2. The basic module of multiplication process

For a 32-bit input word $W = W_3W_2W_1W_0$ where each byte W_i is 8-bits, the MixColumn transformation based on Eq. (3) is given as

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} W_3 \\ W_2 \\ W_1 \\ W_0 \end{bmatrix} = \begin{bmatrix} C_3 \\ C_2 \\ C_1 \\ C_0 \end{bmatrix} \quad (10)$$

The first byte of the matrix operation can be expressed as

$$\begin{aligned} C_3 &= \{02\} \bullet x^3 + \{02\} \bullet x^2 + \{01\} \bullet x^2 + \{01\} \bullet x + \{01\} \bullet 1 \\ &= \{02\} \bullet (x^3 + x^2) + \{01\} \bullet (x+1) + \bullet \{01\} \bullet x^2 \end{aligned} \quad (11)$$

Based on Eqs. (9) - (11), the implementation architecture of MixColumn transformation is shown in Fig. 3.

For InvMixColumn transformation, the Eq. (8) can be rewritten as

$$d(x) = \{04\}_{16} \bullet (x^2 + 1) + \{01\}_{16} \quad (12)$$

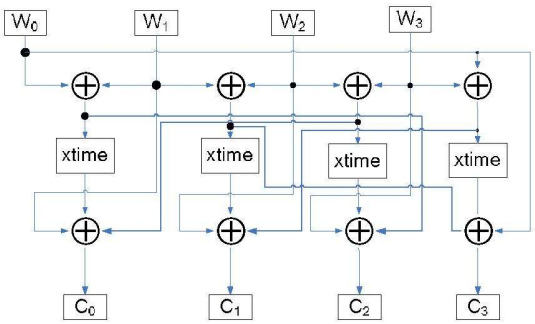

Fig. 3. implementation for mixcolumn transformation

The architecture of the preprocess InvMixColumn transformation based on Eq. (12) is shown in Fig. 4. The functional block of the integrated MixColumns and InvMixColumns transformation is shown in Fig. 5.

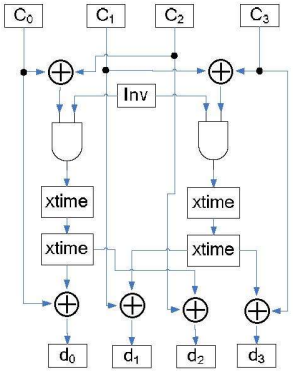

Fig. 4. Architecture of preprocess Invmixcolumn transformation

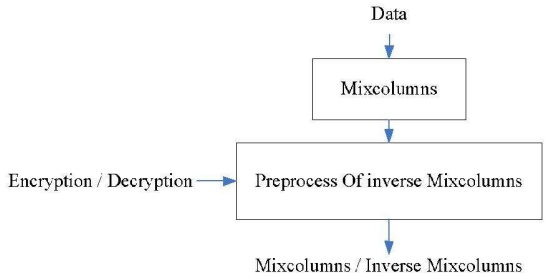

Fig. 5. Functional block of the integrated MixColumns and InvMixColumns transformation

A proposed reconfigurable bandwidth sharing module to enhance the AES performance is shown in Fig. 6. A bandwidth control circuit being arranged for loading data from system bus before addround key transformation. The bandwidth control can be set to match the processor which operates together with the AES module. The main advantage of the proposed architecture is the system performance due to I/O bottlenecks between host processor/AES module can be enhanced. These features allow the AES module to be optimized the I/O data load interface.

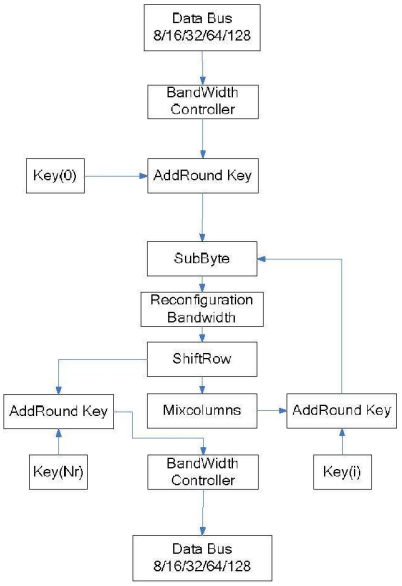

Fig. 6. A proposal reconfigurable bandwidth sharing architecture

IV. IMPLEMENTATION RESULTS

The RTL Model of these IP modules is synthesized to gate-level using Synopsys Design Vision with TSMC 0.35 μm cell library and TSMC 0.18 μm cell libraries [6]. The synthesized verilog codes are then simulated in ModelSim SE 6.0c together with the simulation model of data memory and instruction memory. The switching activities obtained are used by Synopsys PrimePower for power calculations. For FPGA implementation, the AES modules have been implemented using Xilinx VirtexE and Altera Stratix GX. The verification platform for AES module including an ARM7TDMI processor together with Xilinx VirtexE is conducted in this work. The key schedule contains keyExpansion and Round key selection. A software implementation of the keyExpansion transformation including XOR, SubByte transformation and Rocn operation is executed through ARM7TDMI processor. The generated roundkeys are then stored in memory. For AES modules, two different architectures are tested in our study. AES-1 architecture performs encryption and decryption operations without the functional integration. In AES-2 design, the S-box and inverse S-box are integrated for SubBytes and InvSubBytes transformation. The integrated S-box module performs the functions of both S-box and inverse S-box with only one look-up table. Also, the Mixcolumn and preprocess InvMixcolumn are integrated together to perform Mixcolumn/InvMixcolumn transformation.

The experimental results are summarized in Table I for ASIC implementation, and Table II for FPGA implementation, respectively. The differences in cell type, area, total, dynamic,

leakage, and switching powers obtained from Synopsys PrimePower analysis is summarized in Table I. Table II summarizes the FPGA implementation including number of the slices, look up tables, logic elements, total power calculated from Xpower, and the throughput. Results collected in Table III illustrate the comparison among different architectures for the implementation of Mixcolumn/InvMixcolumn transformation

Table I ASIC Implementation Results

Desgin	AES1		AES2	
Tech. (um)	0.18	0.35	0.18	0.35
Area (um2)	408242.4	1970378	285867.5	1578903
Total Power (mW)	11.87	105.1	20.34	192.5
Dynamic Power (mW)	11.86	105.1	20.33	192.5
Leakage Power (nW)	8339	20.78	8820	16.21
Switching Power (mW)	5.275	53.11	8.243	101.7
Critical Path Delay(nS)	20.59	18.13	20.41	20.44
Throughput (Mb/s)	565(48MHz)	641(55MHz)	570(48MHz)	569(48MHz)

Table II FPGA Implementation Results

FPGA	Xilinx	
Design	AES1	AES2
Slice Flip Flop	908	453
4 input LUTs	8932	6338
Total Power (mW)	1950	1329
Delay (ns)	22.222	28.571
Throughput (Mb/s)	523 (45MHz)	407 (35MHz)
FPGA	Altera	
Design	AES1	AES2
Logic Element (%)	10110	6813
Total Power (mW)	5137.53	3017.97
Delay (ns)	14.286	16.667
Throughput (Mb/s)	814 (70MHz)	698 (60MHz)

Table III Comparison of different MC/IMC Design

Level	Architecture-Level	Gate-Level
Methods	Area (AXOR)	Area (gates)
(1) Direct realization [6]	1888	2896
(2) Shared Xtime [7]	1216	2902
(3) IMC decomposition [8]	1696	3928
(4) IMC decomposition [9]	780	2080
(5) Shared Xtime [4]	772	2059
(6) CSE method [6]	772	2059
(7) Hua Li method [5]	1168	
(8) Viktor Fischer [10]	768	
(9) Ours	672	1924

V. SUMMARY AND CONCLUSION

For AES modules, it is difficult to accurately explain the implementation results because the different mapping algorithms built for different platform. However, from area performance point of view, it is clear that the AES with functional integration has the advantage over the one without functional integration. For instance, In AES module with TSMC 0.18 μm cell library, the amount of hardware for implementation with encryption and decryption operations integration has a significant decrease of 29% for AES-2, as compared with the hardware requirement without the functional integration for AES-1. For Xilinx VirtexE/Altera Stratix GX implementation, AES-2 has almost 50% slice saving, 30% reduction in LUTs, and 33% reduction in LEs as compare with the area efficiency of AES-1. The implementation results indicates that the proposed architectures with Mixcolumn and preprocess InvMixcolumn integrated together to perform Mixcolumn/InvMixcolumn transformation have more area efficiency than previous related work.

ACKNOWLEDGMENT

This research was partially supported by National Science Council, R. O. C., under Grant NSC95-2622-E-214-005-CC3.

REFERENCES

[1] J. Daemen and R. Rijmen. (1999) AES Proposal: Rijndael. http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf

[2] (2001, Nov.) Advanced Encryption Standard (AES). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[3] C. P. Su, T. F. Lin, C. T. Huang, and C. W. Wu, "A high-throughput low-cost AES processor," IEEE Commun. Mag., vol. 41, pp. 86–91, Dec. 2003.

[4] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for AES algorithm," IEEE Trans. Very Large Scale Integr. Syst.(VLSI), vol. 12, no. 9, pp. 957–967, Sep. 2004.

[5] Hua Li and Zac Friggstad, "An efficient Architecture for the AES MixColumns Operation", Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on 23-26 May 2005 Page(s):4637 – 4640, 2005

[6] SF. Hsiao, MC Chen, and CS Tu, "Memory-Free Low-Cost Designs of Advanced Encryption Standard Using Common Subexpression Elimination for Subfunctions in Transformations," IEEE Transactions on circuits and systems, vol. 53, no. 3, pp. 615 – 626, 2006.

[7] C. C. Lu and S. Y. Tseng, "Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter," in Proc. Application-Specific Syst., Architectures Processors, Jul. 2002, pp. 277–285.

[8] J.Wolkerstorfer, "An ASIC implementation of the AES MixColumn-operation,"in Proc. Austrochip, Vienna, Austria, Oct. 2001, pp. 129–132.

[9] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with S-box optimization," in Proc. Advanced in Crytopgraphy, 2001, pp. 239–254.

[10] Viktor Fischer, Milo˜s Drutarovsky, Paweł Chodowiec, and Francois Gramain "InvMixcolumn Decomposition and Multilevel Resource Sharing in AES Implementations", in IEEE Transactions on very large scale integration (VLSI) systems, vol.13, NO.8, August 2005, pp.989-992