A New CAM Based S/S^{-1}-Box Look-up Table in AES

Hua Li
Department of Mathematics and Computer Science
University of Lethbridge
Canada T1K 3M4

***Abstract*— In this paper, an efficient s/s^{-1}-box look-up table is proposed based on Content Addressable Memory (CAM) which can be used in both encryption and decryption of AES. The memory complexity is dramatically reduced compared to the SRAM based s-box and s^{-1}-box look-up tables.**

Keywords: CAM, AES, S-Box, cryptography

I. Introduction

AES is a symmetric block cipher which performs multiple rounds of operations on the plain text. These operations are key addition, substitute bytes, shift rows and mix columns [1]. Many AES designs have shown the s-boxes for the substitute bytes operation to be the most expensive module in terms of critical path length in the architecture for AES. There are two types of s-box implementation. One is using finite field inverter [2]. Another is using the look-up table [3]. When using the look-up table approach, usually two tables are required in the architecture, i.e., s-box table and inverse s-box table.

Content Addressable Memory (CAM) has an inherent parallelism that allows parallel access and comparison of the words in the memory [4], [5], [6]. In this paper, we propose a CAM based look-up table to implement both s-box and inverse s-box operation. The comparison indicates that the complexity of combined s/s^{-1} look-up table will be dramatically reduced compared to the two separate s-box and s^{-1}-box look-up tables.

This paper is organised in the following way. Section 2 and 3 briefly introduce the algorithm of AES and the structure of CAM. Section 4 presents an architecture of CAM based s/s^{-1}-box look-up table, and compares this design with single implementation of s-box and s^{-1}-box look-up tables. Section 5 concludes this paper.

II. AES Operations

The AES is a symmetric block cipher that performs four operations on a block of data. These operations are add_round_key, shift_rows, mix_columns and substitute_bytes. Each block of data in the AES cryptosystem is 128 bits long.

The add_round_key operation is only a logical XOR of the current state with a round key that is generated by the key scheduler. Each round key is determined by a series of transformations applied to the initial input key. Representing the block by matrix, the shift_rows operation is the circular shifting of bytes to the left on encryption or to the right on decryption.

The mix_columns operation is a multiplication of a matrix of $GF(2^8)$ elements with each column of the current state to produce another column. The equations of the forward and inverse mix columns operation are given below.

$$\begin{pmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{pmatrix} \tag{1}$$

and

$$\begin{pmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{pmatrix} \begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix} \begin{pmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{pmatrix} \tag{2}$$

where $0 \leq c < 4$.

The substitue_bytes operation is performed on each individual byte in the block according to the s-box table.

III. Content Addressable Memory

Content Addressable Memory (CAM) is a parallel pattern matching circuit. It can search all the content in the memory simutaniously and return the address which stores the matched value. In some applications, it can be regarded as a parallel processor. The architecture based on CAM has been used to accelerate many applications such as image processing [7] and database searching [8].

Conventional memories like SRAMs associate data from address, i.e., the data can be read or written in the corresponding address. CAM has all the functions like SRAM, such as read or write, but it has an additional comparison function which can parallel compare the data in the CAM with the content in the data lines (i.e., the compared data is connected with data lines), and return the corresponding address which satisfied the condition of comparison.

In the following, we design an 8-bit CAM which can be applied to the s-box and s^{-1}-box look-up table. Fig. 1 illustrates the logic circuit of the i-th bit of CAM cell ($0 \leq i \leq 7$). Before searching the content, we reset 'match' line to 1. In the figure, $\bar{B}_i$ is connected to the data line of b_0, and

B_i is connected to the data line of b_1. It can be figured out that $C_i = A_i \oplus B_i$, i.e., $C_i = 1$ if $A_i \neq B_i$, then the match line is connected to ground, and $match = 0$. Otherwise, we can see that $A = B$, and $match = 1$. When doing the write operation, address line is active, and the data is written to the cell (the data is stored at the point of A_i, and $\bar{A}_i$) through data line of b_0 and b_1. When reading the content, we select the corresponding address line, and obtain the memory content through data lines.

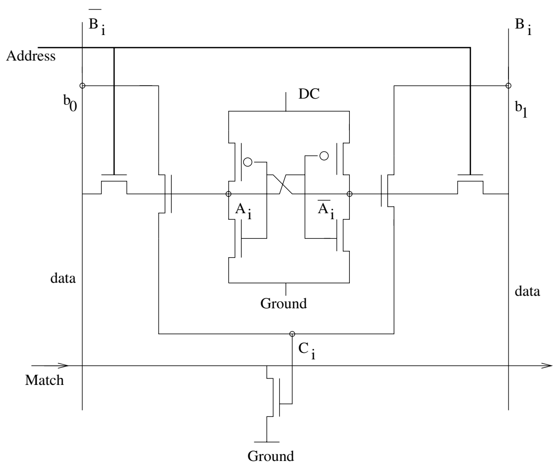

Fig. 1. Logic circuit of a bit cell in CAM

IV. CAM BASED S/S^{-1}-BOX LOOK-UP TABLE

Substitute bytes transformation, called SubBytes, is a non-linear substitution operation (also called s-box). AES defines an s-box table, that contains a permutation of all possible 256 8-bit values. An inverse s-box table is also defined used for decryption. In this section, the architecture of a CAM based s/s^{-1}-box look-up table is proposed.

In the CAM cell array, we use the address line to denote the input of s-box, and store the value of output to the CAM array where the length of the each row is 8bits, and there are 256 rows in the CAM architecture. The initial value of the table are loaded into the CAM at configuration time. When doing the s-box operation, we select the corresponding address line and read the output by the data lines. For example, suppose we want to find the substitution of byte $\{00\}$ (hexadecimal number), the address line of 00 is active, then we can obtain the output of byte $\{63\}$ (hexadecimal number) by reading the content from the data lines. When doing the inverse s-box operation, the searching function of CAM is applied. For instance, in order to find the inverse substitution of $\{63\}$, we put 63 to the comparand register, and begin a parallel search. The CAM structure will match the value of 63, and return the corresponding address which is $\{00\}$, thus we can use this address value as the inverse substitution result of $\{63\}$. Thus, the tables of s-box and inverse s-box can be implemented in one table.

Fig. 2 illustrates the architecture of the proposed CAM based s/s^{-1}-box look-up table. In Fig. 2, there is an 8bit comparand register which stores the compared data. The data in the comparand register is connected with the CAM through the data lines (Fig. 1). Please note that the compared data are inverted when connecting to the data line of b_0 because of the logic structure of the CAM cell. If the data is matched in the CAM, the corresponding match line becomes '1' and the match flag is on. The corresponding address of the CAM can be obtained by a address encoder which returns an 8-bit address value. Because the property of the s-box and s^{-1}-box substitution, there are no redundant data in the table.

Memory	Number of transistors in one cell	Total number of transistors
SRAM s-box and s^{-1}-box	6	24,576
CAM s/s^{-1}-box	9	18,432

TABLE I
COMPLEXITY COMPARISON OF DIFFERENT S-BOX AND S^{-1}-BOX LOOK-UP TABLE

Compared with using two look-up tables for the s-box and inverse s-box substitution, the complexity of the proposed CAM based s/s^{-1}-box look-up table is dramatically reduced. Table I compares the complexity of memory space in different implementations. It can be seen that we only use one CAM based look-up table in both encryption and decryption procedures, and 25% of transistors are reduced compared with SRAM implementation of the s-box and s^{-1}-box look-up tables.

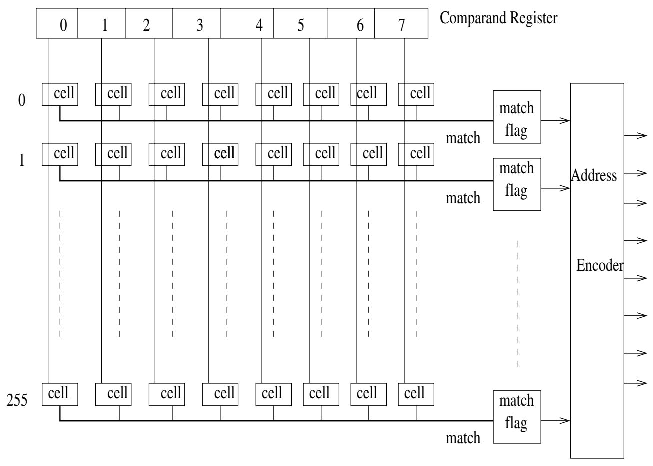

Fig. 2. The architecture of CAM based s/s^{-1}-box look up table

V. CONCLUSION

In this paper, a content addressable memory (CAM) based AES s/s^{-1}-box implementation is proposed. We combine the s-box and s^{-1}-box substitution into one look-up table which is used for both encryption and decryption. Comparing to the two separate SRAM based s-box and s^{-1}-box look-up tables, 25% of transistors are reduced in the proposed one CAM based s/s^{-1}-box look-up table.

Acknowledgment

This research work is supported by the funding of Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] NIST, "Federal Information Processing Standard 197, The Advanced Encryption Standard (AES)", "http://csrc.nist.gov/publications/fips/fips197/fips197.pdf, 2001

[2] A. Satoh and S. Morioka and K. Takano and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," Proc. Advances in Cryptology, ASIACRYPT, pp. 239–254, 2001.

[3] H. Kuo and I. Verbauwhede, "Architectural Optimization for a 1.82Gbits/sec VLSI Implementation of the AES Rijndael Algorithm," CHES 2001, LNCS 2162, pp. 51-64, 2001.

[4] Potter, J. L., "Associative Computing," *Plenum Publishing*, New York, 1992.

[5] Krikelis, A. and Weems, CC., "Associative Processing and Processors," *IEEE Computer Society,* 1997.

[6] Walker, R. A. Potter, J, Wang, Y. and Wu, M., "Implementing Associative Processing: Rethinking Earlier Architectural Decisions," First Workshop on Massively Parallel Processing (WMPP), 27 April 2001 at IPDPS'01.

[7] Naganuma, J. and Ogura, T., "CAM-based Prolog machine and its performance evaluation," *Trans. Inst. Electron. Inf. Commun. Eng. D-I,* 73D-I(11), pp. 856-63, 1990.

[8] Wade, J. P. and Sodini, C. G., "A ternary content addressable search engine," *IEEE J. Solid-State Circuits,* vol.24 n.4, pp.1003-13, 1989.