IEICE TRANS. INF. & SYST., VOL.E94–D, NO.9 SEPTEMBER 2011

PAPER

Software-Based Parallel Cryptographic Solution with Massive-Parallel Memory-Embedded SIMD Matrix Architecture for Data-Storage Systems

Takeshi KUMAKI[†a)], **Tetsushi KOIDE**[††], **Hans Jürgen MATTAUSCH**[††], ***Members***, **Masaharu TAGAMI**[††], ***and*** **Masakatsu ISHIZAKI**[††], ***Nonmembers***

SUMMARY This paper presents a software-based parallel cryptographic solution with a massive-parallel memory-embedded SIMD matrix (MTX) for data-storage systems. MTX can have up to 2,048 2-bit processing elements, which are connected by a flexible switching network, and supports 2-bit 2,048-way bit-serial and word-parallel operations with a single command. Furthermore, a next-generation SIMD matrix called MX-2 has been developed by expanding processing-element capability of MTX from 2-bit to 4-bit processing. These SIMD matrix architectures are verified to be a better alternative for processing repeated-arithmetic and logical-operations in multimedia applications with low power consumption. Moreover, we have proposed combining Content Addressable Memory (CAM) technology with the massive-parallel memory-embedded SIMD matrix architecture to enable fast pipelined table-lookup coding. Since both arithmetic logical operation and table-lookup coding execute extremely fast on these architectures, efficient execution of encryption and decryption algorithms can be realized. Evaluation results of the CAM-less and CAM-enhanced massive-parallel SIMD matrix processor for the example of the Advanced Encryption Standard (AES), which is a widely-used cryptographic algorithm, show that a throughput of up to 2.19 Gbps becomes possible. This means that several standard data-storage transfer specifications, such as SD, CF (Compact Flash), USB (Universal Serial Bus) and SATA (Serial Advanced Technology Attachment) can be covered. Consequently, the massive-parallel SIMD matrix architecture is very suitable for private information protection in several data-storage media. A further advantage of the software based solution is the flexible update possibility of the implemented-cryptographic algorithm to a safer future algorithm. The massive-parallel memory-embedded SIMD matrix architecture (MTX and MX-2) is therefore a promising solution for integrated realization of real-time cryptographic algorithms with low power dissipation and small Si-area consumption.

key words: *matrix-processing architecture, SIMD, bit-serial and word-parallel, CAM, table-lookup coding, cryptographic algorithm, AES*

1. Introduction

The rapid development of semiconductor-integration and VLSI-implementation technology is leading to a continuous improvement in our mobile computing environment. There has been a spread across society of mobile applications like notebook or netbook PC, digital still camera, mobile phone and many forms of flash-memory storage, such as USB flash memory, SD card and so on. Almost every person is now carrying huge volumes of personal data, such as pictures, mail addresses or business documents. On the other hand, since publication of the OECD's eight principles document in 1980 [1], which contains guidelines for the protection of privacy and transborder flows of personal data, the end-user is increasingly interested in the protection of personal data. The Japanese government also enforces the private information protection law in April 2005. Thus, mobile appliances need to apply private information security technology, especially cipher algorithm processing, for protection against the leakage of personal information. Additionally, low power dissipation and small hardware area are also required-capability for mobile equipment.

Presently, several USB flash memories are supported in their security-function capability [2], [3], which can prevent accidental leaking of the private information. Thus, hardware encryption ASICs have been implemented directly on the main board of USB flash memories and can allow fast transfer of the encrypted or decrypted data, when the end-user deals with his personal data. Several hard disk drives [4], [5] also adopt hardware-based full disk encryption. Above security solutions have begun to adopt the 256-bit AES (Advanced Encryption Standard) algorithm for robust encrypted data. On the other hand, previous standard cryptographic algorithms, such as DES (Data Encryption Standard), RSA (Rivest Shamir Adleman), SHA (Secure Hash Algorithm) etc, are no longer considered to provide the data security that is needed to protect personal information. This fact is obviously caused by the cipher-related year-2010 problems [6]–[9]. Similarly, the 256-bit AES algorithm is undeniably also exposed to a hazard against private information in the feature. Thus, an ASIC-based cryptography method is not completely safe against leaking of private information, because the implemented algorithm may become corruptible in future. Consequently, the encryption algorithm, which is implemented in a cryptography chip, may have to be changed to a safer algorithm at some point in time. However, an ASIC-based implementation of the cryptographic algorithm cannot be updated or modified. As a result, above cipher-related problems make heavy demands on the end-user's cost burden for keeping a high safety level. On the other hand, software-based general-purpose CPU or media processor solutions tend to increase power con-

Manuscript received July 13, 2010.
Manuscript revised March 6, 2011.

[†]The author is with the Department of VLSI System Design, Ritsumeikan University, Kusatsu-shi, 525–8577 Japan.
[††]The authors are with Research Institute for Nanodevice and Bio Systems, Hiroshima University, Higashi-hiroshima-shi, 739–8527 Japan.
a) E-mail: kumaki@fc.ritsumei.ac.jp

DOI: 10.1587/transinf.E94.D.1742

sumption and hardware cost, and are also unsuitable for the bit-oriented data operations of the cryptographic processing [10].

For overcoming these security-related problems, this paper presents a software-based parallel cryptographic processing solution with a massive-parallel memory-embedded SIMD matrix for a flexible and robust data-security system. The massive-parallel memory-embedded SIMD matrix has been proposed as a novel SIMD multimedia processor, which provides a better way for processing several types of multimedia applications [11]–[16]. It achieves highly parallel processing with low power consumption, and can thus target the mobile product applications. Moreover, since the massive-parallel memory-embedded SIMD matrix uses a software-based architecture for several algorithm implementations, it is programmable for all processing functions required by multimedia VLSI chips.

For increasing the degree of parallel processing and flexibility, the massive-parallel SIMD matrix applies bit-serial and word-parallel mode of operation and a close connection between many small processing elements and SRAM arrays. Then, the matrix architecture can enable effective processing of multimedia contents with a large data amount and decrease data-transfer power consumption. Furthermore, for improving the processing efficiency of multimedia data, a Content Addressable Memory (CAM)-enhanced massive-parallel SIMD matrix has also been proposed, which can realize simultaneously fast repeated arithmetic operations and pipelined table-lookup coding and achieves optimized performance of multimedia applications [15], [17]. For optimizing the security-related processing capability in mobile data storage devices, such as USB flash memory, SDD and so on, we propose an effective parallel-implementation method of the cryptographic algorithm on massive-parallel SIMD matrix processors. As an example, from the many cryptographic algorithms known to the public, the Advanced Encryption Standard (AES) algorithm is selected and implemented. The number of encryption clock cycles for the AES algorithm is then determined with the CAM-less and CAM-enhanced massive-parallel SIMD matrix architectures. Furthermore, several standard data-storage-transfer specifications are compared with reference to the throughput of the massive-parallel SIMD matrix processors.

This paper is organized as follows. Section 2 describes the massive-parallel memory-embedded SIMD matrix processor architecture in detail. Section 3 explains the AES algorithm and its flow-chart. Section 4 proposes a detailed processing method for the AES algorithm using the massive-parallel memory-embedded SIMD matrix architecture. Section 5 describes the developing environment for the massive-parallel memory-embedded SIMD matrix and compares the performance of the AES implementation with the requirements of recent data-transfer standards. Finally, Sect. 6 concludes this paper.

2. Massive-Parallel Memory-Embedded SIMD Matrix Architecture

In this section, the developed multimedia SIMD processor architecture and the CAM-enhancement of this architecture are described to facilitate the understanding of the later sections. These two architectures have also been verified to allow effective processing of both repeated arithmetic operations and table-lookup coding operations in several multimedia applications.

For processing multimedia applications efficiently, various SIMD architectures have been reported previously [18], [19]. However, the number of Processing Elements (PEs), which can be implemented, is mostly quite small.

We have developed a massive parallel processor based on a SRAM-embedded matrix architecture [11]–[15], [20], [21], which overcomes the limitations in parallelism of previous architectures. This massive-parallel SIMD matrix architecture achieves for example 40 GOPS performance for 16-bit additions at 200 MHz clock frequency and 250 mW power dissipation in a 90 nm CMOS technology [11]. It is furthermore programmable for all processing functions required for multimedia VLSI chips. The block diagram of the SIMD matrix architecture is shown in Fig. 1. 1 M-bit SRAM is provided for data registers and 2,048 2-bit processing elements (PEs), connected by a flexible switching network, are integrated e.g. on 3.1 mm^2 in a 90 nm low power CMOS technology. A Vertical channel (V-ch) connects the PEs, while a Horizontal channel (H-ch) connects SRAM-register space and PEs. Through V-ch a communication path between a PE and another PE, being a distance of a power of 2 rows away, is generated and operated in one cycle. As for H-ch, 3 register accesses (2 read, 1 write) in the form of a read-modify-write operation are achieved in one cycle by

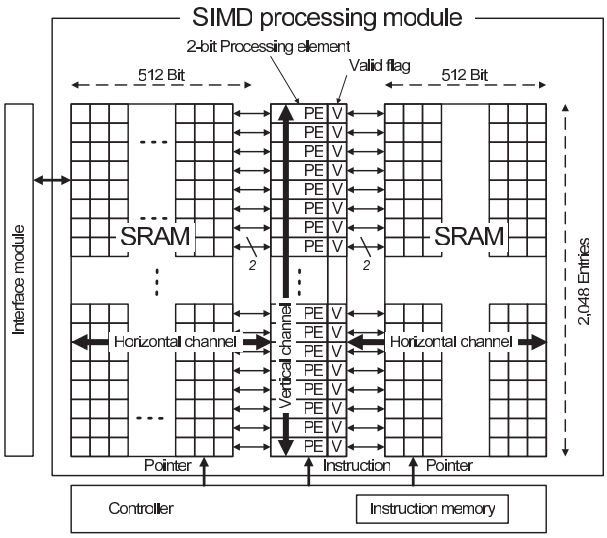


Fig. 1 Block diagram of the massive-parallel memory-embedded SIMD matrix architecture.

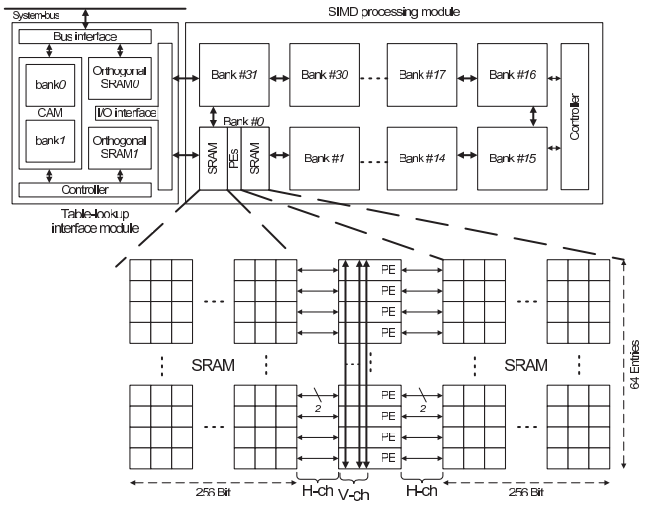


Fig. 2 Physical implementation image of the CAM-enhanced massive-parallel SIMD matrix architecture.

the SRAM division into 2 simultaneously accessible parts. Since the massive-parallel SIMD matrix consists of a simple SRAM-based architecture, the processed-data width and the magnitude of parallelism can be changed and optimized flexibly according to application needs.

Generally, most conventional processors execute with bit-parallel and word-serial operation. In the example of a DSP, multimedia data is processed sequentially in a pipelined form. Thus, these conventional DSP architectures can be regarded as realizing parallel processing in the time domain. On the contrary, our developed massive-parallel SIMD matrix supports 2,048-way bit-serial and word-parallel operations. Moreover, the 2,048 PEs can synchronize with a single command and process all stored data, thus realizing a highly parallel SIMD architecture. Therefore, the massive-parallel SIMD matrix can be regarded as realizing a space domain architecture for parallel processing.

Besides an effective parallel processing for repeated arithmetic operations, we utilized our previous works [22], [23] on effective and fast processing of the table-lookup coding operation when executed by a CAM-exploiting architecture, for developing a still more versatile multimedia processor. This resulted in a CAM-enhanced massive-parallel SIMD matrix, which has a physical implementation image is shown in Fig. 2. The considered architecture implementation has 32 matrix banks, a controller circuit, and an interface module enhanced with a CAM-based table-lookup function. Each bank consists of two SRAM-cell-based data registers with 64×256 bit and 64 2-bit PEs, which directly connect with the data registers. Again, a V-ch connects the PEs, while a H-ch connects memory cells and PEs. The block diagram of the table-lookup-enhanced interface module, which can take over the role of table-lookup coding operations during data input to or output from for the SIMD matrix, is shown in Fig. 3. It is composed of two CAM banks, two orthogonal SRAMs, a controller, a bus interface,

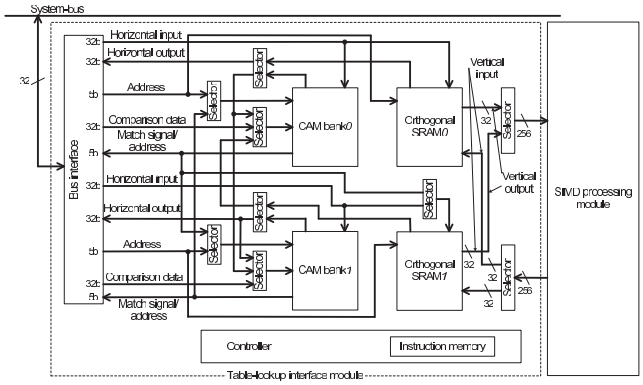


Fig. 3 Block diagram of the interface module enhanced with the CAM-based table-lookup function.

and nine selectors.

For efficiently realizing above integration architecture to allow effective multimedia data processing, sufficient consideration of the differences in data handling between SIMD architecture and CAM is needed. While the massive-parallel SIMD matrix is optimized for bit-serial and word-parallel operation, the CAM has to execute bit-parallel and word-serial operations for achieving best performance. Therefore, the two CAM components have their optimum location in the interface module, which performs an orthogonal transformation of the data direction, for enabling the bit-serial processing of the SIMD matrix. As a result, the CAM processes the role of processing the table-lookup coding operations at the word-parallel end within the interface module. In this way, the CAM-enhanced massive-parallel SIMD matrix architecture can efficiently exploit the fast search operation provided by the CAM function for faster execution of all multimedia algorithms which require the CAM capabilities.

For the JPEG application, studied as an example, the clock cycle number required by the proposed CAM-enhanced processor is up to 86% smaller than with a conventional mobile DSP. Furthermore, its efficiency in Mpixel/mm^2 is up to 3.3 times and 4.4 times higher than that of a CAM-less massive-parallel SIMD matrix and a conventional mobile DSP, respectively [15].

3. Advanced Encryption Standard (AES) Algorithm

The Advanced Encryption Standard (AES) algorithm is a well-known cryptographic algorithm which has been announced as replacement to the Data Encryption Standard (DES) by the National Institute of Standards and Technology (NIST) [24] in 2000 and 2001. It is therefore a FIPS (Federal Information Processing Standards) approved cryptographic applications and is described in detail in [24]. The ASE algorithm uses a cryptographic key called “Round key” and its flow chart is shown in Fig. 4. The AES is categorized as a block-cipher algorithm, which processes typically blocks of 128 bits from the original message by using Round Keys with a data length of 128, 192, or 256 bits. The encryp-

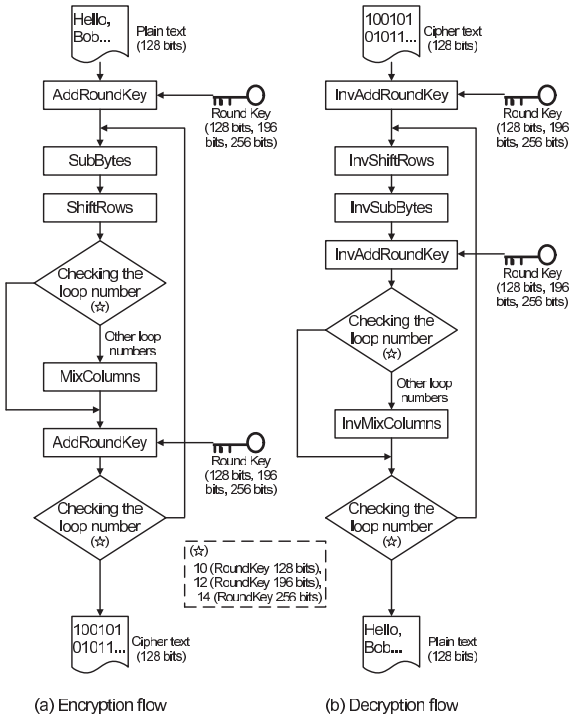


Fig. 4 Processing flow diagram of the Advanced Encryption Standard (AES) algorithm: (a) Encryption flow, (b) Decryption flow.

tion flow (Fig. 4-(a)) uses four elementary transformations called SubBytes, ShiftRows, MixColumns, and AddRoundKey, while the decoding flow (Fig. 4-(b)) uses the inverse of these four elementary transformations called InvSubBytes, InvShiftRows, InvMixColumns, and InvAddRoundKey.

The main loops of the encoding/decoding flow are repeated 10, 12, or 14 times depending on the number of Round Key bit-length (128, 196 and 256 bit). For the basic processing of the AES algorithm, blocks of 128 bits from the original message are arranged into arrays of 4 columns by 4 rows containing 8-bit elements called bytes. The SubBytes and the InvSubBytes transformations are nonlinear byte substitutions that operate independently on each byte of the 4×4 array by using a substitution table called S-Box or S^{-1}-Box (inverse S-Box). The ShiftRows or the InvShiftRows transformations are circular shifting operations on the rows of the 4×4 arrays by different numbers of bytes. The MixColumns or the InvMixColumns transformations replace elements of each column by carrying out a matrix-multiplication with a defined matrix. The AddRoundKey or the InvAddRoundKey transformations calculate XOR operations between the data block and the Round Key.

4. Efficient AES Processing with SIMD Matrix Processor

The massive-parallel memory-embedded SIMD matrix can accelerate efficiently the block-cipher processing by using its high parallelism, flexible data-width, flexible data channel and fast table-lookup architecture. Generally, as shown

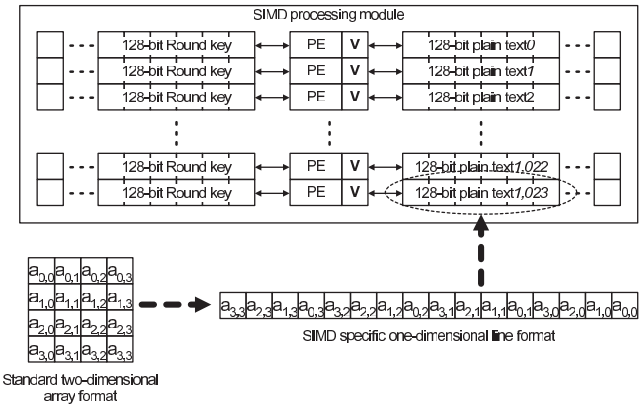


Fig. 5 Storage scheme of plain texts in the massive-parallel memory-embedded SIMD matrix (example of 128-bit plain texts and a 128-bit round key).

in Fig. 5, the plain text of the block-cipher algorithm is normally represented in two-dimensional array format [24], because conventional processors are limited to a single-access data width, such as 16 or 32 bit. On the other hand, the proposed SIMD matrix processing module has a flexible data width architecture up to 256 or 512 bits [11], [16]. The massive-parallel memory-embedded SIMD matrix can therefore adopt one-dimensional line format to take advantage of its highly parallelism.

4.1 AddRoundKey and InvAddRoundKey Transformations

In the AddRoundKey and the InvAddRoundKey processing, which is applied several times in the flows of Fig. 4, parallel bit-serial XOR operations are calculated between the encoded data and the 128 bits of the round key. As an Fig. 6 shows the procedure for implementing the AddRoundKey transformation for parallel-encoding of 2,048 blocks with 128 bit each. The InvAddRoundKey can also be executed according to Fig. 6 procedures without modification.

Four steps AR-1 to AR-4 of AddRoundKey transformation processing on the massive-parallel SIMD matrix processor can be represented in the following four items:

AR-1 The controller downloads the 128-bit AddRoundKey data from SRAM through the system bus into its registers and broadcasts the 128-bit AddRoundKey to the left-side SRAM registers in the SIMD matrix processor.
AR-2 2,048 PEs in the SIMD matrix processor can execute XOR operation in parallel.
AR-3 The bit-serial XOR operation is continuously executed in the right-side SRAM registers.
AR-4 Encoded data in the 2,048 blocks are stored in right-side SRAM registers.

As a result, the massive-parallel memory-embedded SIMD matrix architecture can use 2,048 PEs for the Ad-

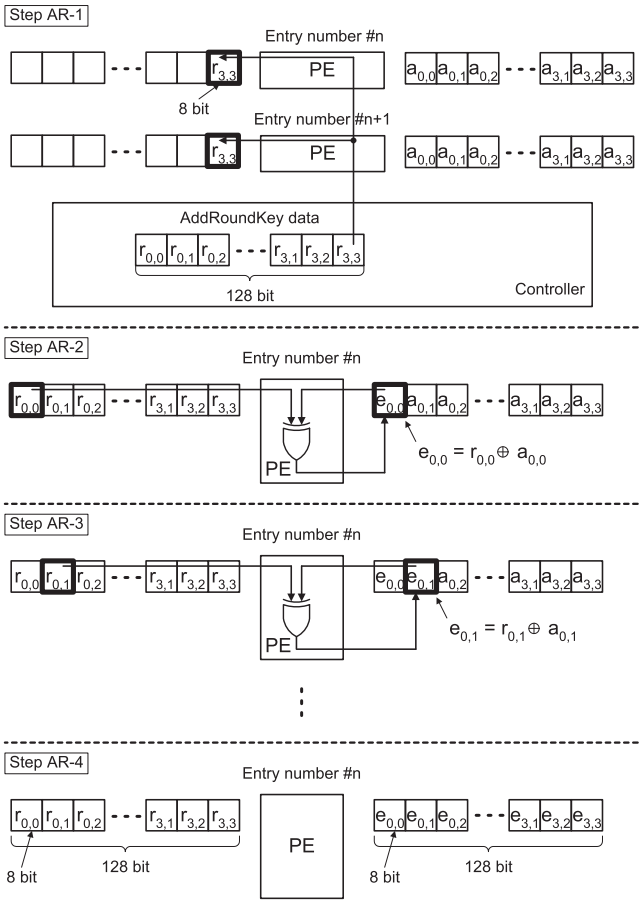


Fig. 6 Detailed implementation procedures of the AddRoundKey transformation on the massive-parallel memory-embedded SIMD matrix: (Step AR-1) Copy AddRoundKey to all rows of the SIMD-processor, (Step AR-2) Execute the XOR operation for bit 1 - 8, (Step AR-3) XOR operation for bit 9 - 16, (Step AR-4) XOR operation for all 128 bits.

dRoundKey and the InvAddRoundKey transformation as efficiently as possible.

4.2 SubBytes and InvSubBytes Transformations

The SubBytes and InvSubBytes transformations always follow after finishing an AddRoundKey and InvShiftRows transformations, respectively. They need to perform table-lookup operations according to the S-Box or the S^{-1}-Box code-word tables, which can be done with a one-by-one substitution of 8-bit data parts from the 128 bit data blocks. Although, the table-lookup coding operation is known as less suitable for a SIMD processing architecture [15], it can be implemented on the massive-parallel memory-embedded SIMD matrix, as described in the following.

For explanation of the substitution process according to the S-Box table, Fig. 7 shows the example of transforming the 8-bit data $\{FF\}_{16}$ and $\{FE\}_{16}$ into $\{16\}_{16}$ and $\{BB\}_{16}$, respectively. The transformation procedure starts with addition operations of 1 to all 8-bit data parts, e.g. $\{FF\}_{16}$ becomes $\{00\}_{16}$ due to an overflow, and uses the ALU, the valid flag and the carry flag in each processing element (PE).

Five processing steps SB-1 to SB-5, as explained below, are sufficient for implementing the SubBytes transformation and can be represented in the following five items:

SB-1 The SIMD processing module stores up to 2,048 post-AddRoundKey data blocks and sets an incremental value for the addition operation in parallel.

SB-2 The substitution value $\{16\}_{16}$, which is determined by the intersection of the row with index $\{F\}_{16}$ and the column with index $\{F\}_{16}$ in S-Box table [24] is loaded to a register in the controller. Then, the substitution value is broadcasted to the area of S-Box table value.

SB-3 For the valid past-substitution values in S-Box table area, the post-AddRoundKey data are incremented in parallel. Consequently, all rows, which have the value of $\{FF\}_{16}$, generate overflow bits. Overflow bits are copied into carry flag registers.

SB-4 Overflow bits in carry flag registers are inverted and copied into the valid flag register. Thereby, rows with an overflow, for which the S-Box transformation is finished, are deactivated for the subsequent processing.

SB-5 The value $\{BB\}_{16}$, which substitutes the input data $\{FE\}_{16}$ according to the S-Box code-word table, overwrites the "S-Box table value field" in the rows remaining valid. The S-Box transformation continues with steps SB-3 to SB-5 until all S-Box table values have been processed.

In the described algorithm, the massive-parallel memory-embedded SIMD matrix has to execute steps SB-3 to SB-5 for all possible input values from $\{FF\}_{16}$ to $\{00\}_{16}$, regardless of the actual distribution the input value. Thus, the performance tends to be low due to the large number of necessary clock cycles.

On the other hand, the CAM-enhanced massive-parallel memory-embedded SIMD matrix [15], which is verified to allow effective processing of both repeated arithmetic or logic operations and table-lookup coding operations in several multimedia applications, is more efficiently applied to generating substitution values in the SubBytes or the InvSubBytes transformation. Here, the CAM, which is included in table-lookup interface module (see Fig. 3), can realize a fast SubBytes transformation. Moreover, a CAM-based table-lookup coding architecture simplifies the implementation of the InvSubBytes transformation. Figure 8 shows the concept of a CAM-based transformation processing. The SubBytes transformation utilizes the CAM function for substitution coding, as shown in Fig. 8-(a), by storing the possible input data in such a way that the corresponding substitution data is represented by the matching address. The InvSubBytes transformation can then be simply realized by RAM function using the input data as the address under which the substitution data is stored, as shown in Fig. 8-(b).

For pipeline processing of the S-box or the S^{-1}-box transformation in the interface module of the CAM-enhanced massive-parallel memory-embedded SIMD matrix, both CAM bank0 and CAM bank1 are used to store the possible post-AddRoundKey data, as shown in Fig. 9. Both

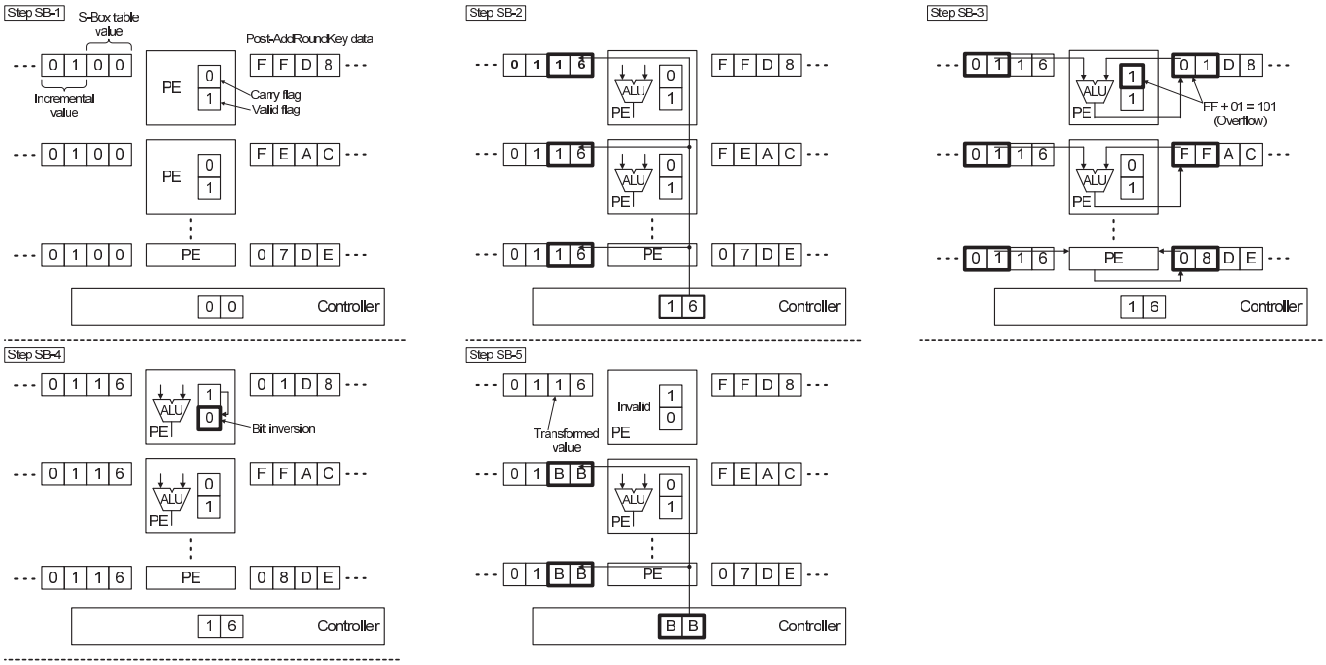


Fig. 7 Processing flow diagram of table-lookup procedure, required for the SubBytes transformation of the advanced encryption standard (AES) algorithm, on the massive-parallel SIMD matrix processor: (Step SB-1) Default setting, (Step SB-2) Broadcast operation of first table value (corresponding to substitution for FF), (Step SB-3) Addition operation, (Step SB-4) Invert valid-flag operation for rows with overflow(carry flag became 1), (Step SB-5) Broadcasting of 2nd table value (corresponding to substitution for FE) to valid rows.

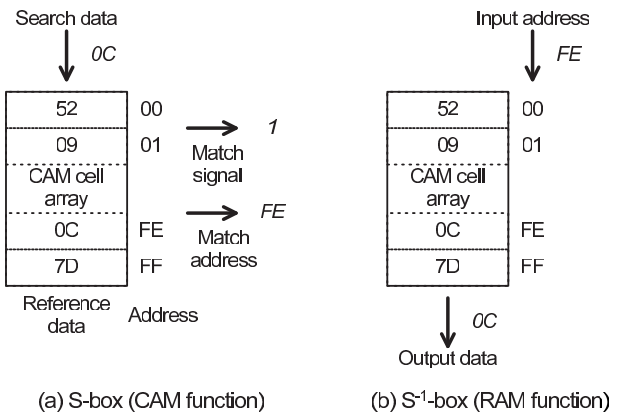


Fig. 8 Concept of the CAM-based SubBytes or InvSubBytes table-lookup transformation processing.

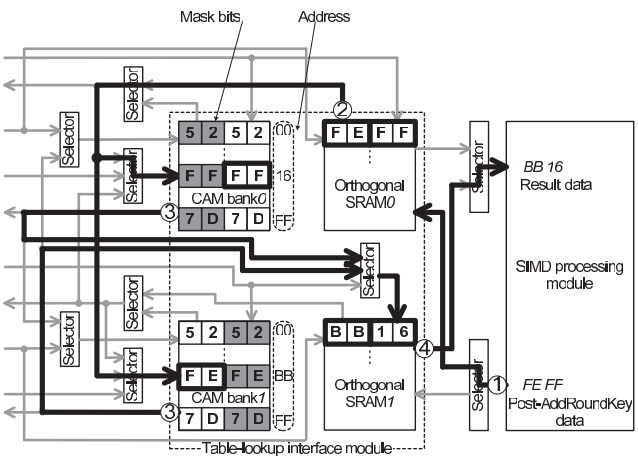


Fig. 9 Fast CAM-based SubBytes transformation flow within the table-lookup interface module: (Step SBC-1 (①)) Store post-AddRoundKey data, (Step SBC-2 (②)) Send stored data to two CAM banks, (Step SBC-3 (③)) Generate matching addresses for SubBytes transformation, (Step SBC-4 (④)) Output result data.

CAMs have this data two times under in each address. The detailed pipelined-coding procedure is described by the four processing steps SBC-1 to SBC-4 as explained below. The mapping of four steps onto the pipeline stages of the hardware is also indicated in Fig. 9. Steps SBC-1 to SBC-4 are represented by ① to ④, respectively.

SubBytes transformation processing steps on the CAM-enhanced massive-parallel SIMD matrix processor can be represented in the following four items:

SBC-1 The SIMD processing module outputs 8 bit data (e.g. $\{FE\}_{16}$ and $\{FF\}_{16}$), after finishing the AddRoundKey transformation, sequentially bit column after bit column of the data words. The orthogonal SRAM0 stores these post-AddRoundKey data in vertical direction. ⟶ ① in Fig. 9.

SBC-2 The orthogonal SRAM0 then outputs the post-AddRoundKey data in horizontal direction i.e. word by word in descending order of the storing address. CAM

bank0 and CAM bank1 receive outputted data and set left-half and right-half mask bits, respectively. ⟶ ② in Fig. 9.

SBC-3 The CAM bank0 and CAM bank1 compare input data, such as $\{FF\}_{16}$ and $\{FE\}_{16}$ with all possible input symbols of the post-AddRoundKey in parallel. Two matching addresses from the CAM bank0 and the CAM bank1, which are corresponding two substitution values (e.g. $\{16\}_{16}$ and $\{BB\}_{16}$), are then combined and sent to the orthogonal SRAM1. ⟶ ③ in Fig. 9.

SBC-4 After steps SBC-2 - SBC-3 have been carried out for all post-AddRoundKey data in SRAM0, the orthogonal SRAM1 then seamlessly outputs its substitution data in vertical direction bit column by bit column to the SIMD processing modules. ⟶ ④ in Fig. 9.

Above CAM-exploiting processing steps can realize fast pipelined SubBytes operations. Furthermore, fast InvSubBytes operations are realized by using the input data as an address and by exploiting the RAM function of the CAM.

4.3 ShiftRows and InvShiftRows Transformation

The processing of the ShiftRows and the InvShiftRows operations can be realized without executing actual changes to the data obtained from the after previous processing steps. Since the one-dimensional line format (Fig. 5) is adopted for storing intermediate data, the massive-parallel memory-embedded SIMD matrix can write and read up to 2,048 obtained data at all entries indicated by a pointer at the same time. Therefore, the ShiftRows and the InvShiftRows operations can be achieved by only changing the pointer of each entry during read-out of the results from the previous AES-algorithm steps of the in processing flow shown in Fig. 4 and can be combined with the SubBytes and the InvMixColumns processing, respectively.

4.4 MixColumns and InvMixColumns Transformation

The MixColumns and the InvMixColumns transformation is treated by a four-term polynomial as described in [24]. The required operations can be written in XOR form with the following matrix:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} : MixColumns$$

$$\begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix} : InvMixColumns$$

These are 7 different matrix elements, namely $\{01\}_{16}$, $\{02\}_{16}$, $\{03\}_{16}$, $\{09\}_{16}$, $\{0b\}_{16}$, $\{0d\}_{16}$ and $\{0e\}_{16}$, which can be calculated from the two elements $\{01\}_{16}$ and $\{02\}_{16}$. For example, $\{03\}_{16}$ and $\{09\}_{16}$ can be represented in the following equations:

$$\{03\}_{16} = \{02\}_{16} + \{01\}_{16}$$
$$\{09\}_{16} = \{02\}_{16} + \{02\}_{16} + \{02\}_{16} + \{01\}_{16}$$

The massive-parallel memory-embedded SIMD matrix enables flexible bit-oriented shift operations in parallel. Therefore, above sample equations can be executed by the SIMD processing module in the following way:

$$\{03\}_{16} = \{1bit\_shift\} + \{addition\_of\_\{01\}_{16}\}$$
$$\{09\}_{16} = \{3bit\_shift\} + \{addition\_of\_\{01\}_{16}\}$$

As a result, the MixColumns and InvMixColumns operations, which need some complex product-sum calculation, are realized by simple shift operations and an addition on the SIMD processing module.

Figure 10 describes the eight processing steps MC-1 to MC-8, which are represented in the following eight items:

MC-1 The SIMD processing module stores up to 2,048 blocks of 128 bit data, each represented by following 16 elements:
$a_{0,0}$, $a_{1,0}$, $a_{2,0}$, $a_{3,0}$, $a_{0,1}$, $a_{1,1}$, $a_{2,1}$, $a_{3,1}$, $a_{0,2}$, $a_{1,2}$, $a_{2,2}$, $a_{3,2}$, $a_{0,3}$, $a_{1,3}$, $a_{2,3}$, $a_{3,3}$.
These 16 elements are obtained after finishing the ShiftRows transformation in case of AES encoding.

MC-2 All PEs output $\{02\}_{16} \cdot a_{0,0}, \{02\}_{16} \cdot a_{1,0}, \{02\}_{16} \cdot a_{2,0}, \{02\}_{16} \cdot a_{3,0}, \cdots, \{02\}_{16} \cdot a_{3,3}$ by 1 bit-left-shifting of the values $a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, \cdots, a_{3,3}$.

MC-3 If processed data overflows in step MC-2 as a result of the shift-operation, a rounding operation is needed. For searching the overflowing cases, all PEs copy the Most Significant Bit (MSB) of the elements $a_{0,0}, a_{1,0}, a_{2,0}, a_{3,0}, \cdots, a_{3,3}$ to their carry flag registers.

MC-4 The MSB data in the carry flag registers are copied to the valid flag registers. With these valid flags, the cases which need a rounding operation are identified.

MC-5 The results of the rounding operation for the valid rows are obtained by XOR operations between the shifted-data and $\{1b\}_{16}$.

MC-6 Repeating steps MC-3 to MC-5, the massive-parallel memory-embedded SIMD matrix architecture can calculate rounding data in parallel.

MC-7 All PEs execute XOR operations between $a_{0,0}, a_{1,0}$, $a_{2,0}, a_{3,0}, \cdots, a_{3,3}$ and the results of step MC-6, respectively.

MC-8 By repeating step MC-7, the massive-parallel memory-embedded SIMD matrix architecture can obtain easily $\{03\}_{16}$ times plain text data in parallel.

As a result, the massive-parallel memory-embedded SIMD matrix architecture can represent all elements in MixColumns and InvMixColumns matrices by using only shift operation and addition. Due to the storage of seven different matrix elements in the same row, the actual matrix multiplication can then be carried out by applying the normal addition and multiplication operations of the SIMD processor, without data communication clock cycles by using the vertical channel (V-ch).

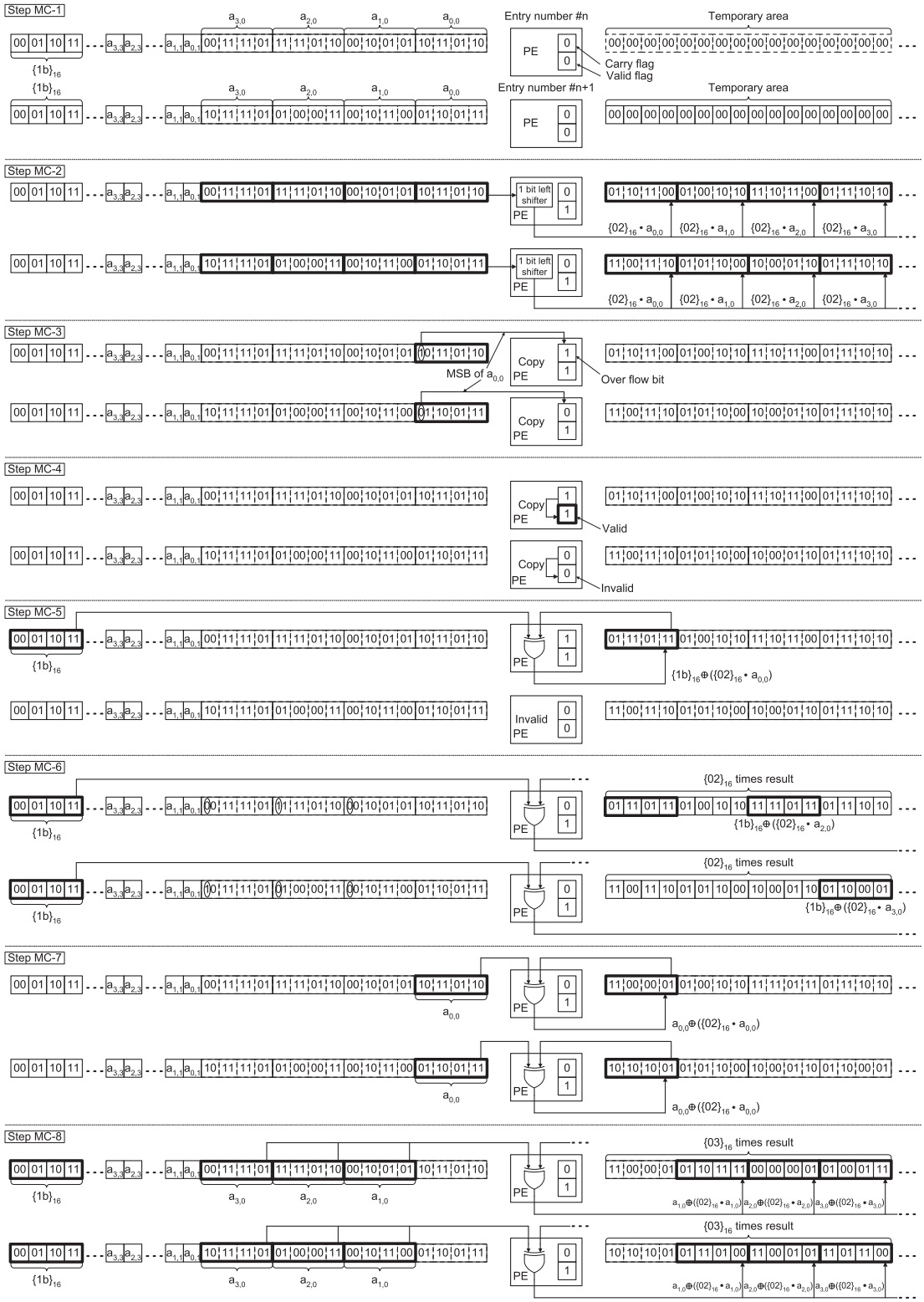


Fig. 10 Detailed procedures of the shift and rounding operations in the MixColumns transformation when expected by the massive-parallel memory-embedded SIMD matrix: (Step MC-1) Default setting, (Step MC-2) 1 bit shift operation for $a_{0,0}$ to $a_{3,0}$ (matrix multiplication 02), (Step MC-3) Copy MSB to carry flag register for all rows, (Step MC-4) Copy carry-flag bit to valid-flag bit for all rows (valid rows become 1), (Step MC-5) Execute XOR operation between $\{02\}_{16} \cdot a_{0,0}$ and $\{1b\}_{16}$ for rounding operation in valid rows (rounding operation), (Step MC-6) Repeat three steps MC-3, MC-4 and MC-5, (Step MC-7) Execute XOR operation between step MC-6 result and $a_{0,0}$ (rounding operation), (Step MC-8) Repeat step MC-7.

5. Experimental Results for AES Encryption Processing

In this section, several experimental results for the security application of implementing the cryptographic AES algorithm with the massive-parallel SIMD matrix architecture are reported. For obtaining the number of processing clock cycles, on the SIMD matrix processor, the AES encrypting algorithm has been implemented with a C language-based embedded program and tested by SIMD-matrix-processor hardware on an evaluation board. Figure 11-(a) and (b) shows a photograph and the system block diagram of the evaluation board. Main components are the massive-parallel SIMD matrix processor, a host CPU, a DMA (Direct Memory Access controller) and two SDRAMs. Maximum clock frequencies of the SIMD matrix processor, host CPU and SDRAM are 162 MHz, 81 MHz, and 81 MHz, respectively. The specifications of the massive-parallel SIMD matrix processor are: 1,024-parallelism, 512-bit word-length, 2-bit-serial processing and 150 mW power dissipation. The host CPU dispatches several tasks to the massive-parallel SIMD matrix and the DMA, and executes serial operations. The DMA transfers plain text data between SDRAMs and the massive-parallel SIMD matrix through the system bus.

In principle, the execution of the SubBytes operation method using the host CPU is a technically possible alternative. However, this possibility has several drawbacks which lead to a long effective processing time for the SubBytes operation. In case of executing the SubBytes transformation with the host CPU, the host CPU needs to move post-AddRoundKey data from the massive-parallel SIMD matrix to the SDRAMs through the system bus before the SubBytes transformation can be performed. The reason is that the host CPU cannot use cache memory to process the post-AddRoundKey data between the CPU register and the orthogonal SRAM in the interface module, directly. Moreover, the shared system bus may be occupied by other processes at the time of the data transfer and may therefore not be fully available. Consequently, a large number of additional unproductive data-transfer clock cycles will be needed for a CPU-based SubBytes transformation. On the other hand, the CAM-based table-lookup operation can be executed on the fly in a pipelined highly parallel manner within the interface module and does not need the system bus. Thus, the CAM-based SubBytes operation becomes possible with the number of clock cycles (or even less clock cycles) which would otherwise be needed for only transferring the data between the SDRAMs and the SIMD matrix.

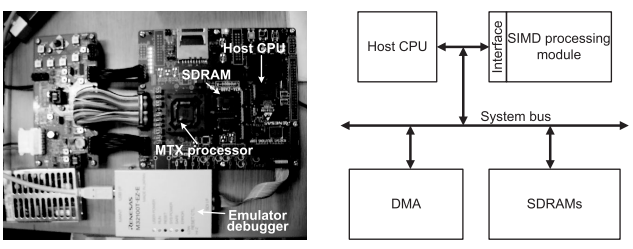


Fig. 11 Photograph and system architecture diagram of the evaluation board.

Figure 12 shows the determined number of clock cycles for the AES encrypting algorithm for encoding of 1,024 128-bit data blocks. The two discussed architectures, namely the massive-parallel SIMD matrix (CAM-less MTX) and the CAM-enhanced massive-parallel SIMD matrix (CAM-enhanced MTX) are indicated on the vertical axis and the accumulated number of clock cycles is given on the horizontal axis. Since the AddRoundkey transformation, the MixColumns transformation, the ShiftRows transformation and data input/output are very suitable for parallel processing, both architectures are able to decrease the number of clock cycles in comparison to a conventional DSP. Especially, the ShiftRows transformation needs no additional clock cycles, because this transformation can be executed only by a suitable pointer operation during read-out of the data (The detailed explanation has been given in Sect. 4.3). On the other hand, since the SubBytes transformation involves essentially sequential operations, the CAM-less massive-parallel SIMD matrix processor needs a large number of clock cycles for this part of the AES encoding. To overcome this problem, we have proposed the CAM-enhanced massive-parallel SIMD matrix architecture [15], which is highly effective for both, repeated logical operations and the table-lookup coding, needed for the SubBytes operation, as described in Sect. 2. Thus, the CAM-enhanced massive-parallel SIMD matrix processor, which adds a the table-lookup interface module to the massive-parallel SIMD matrix (Fig. 2), is verified in Fig. 12 to realize a significant further reduction of the number of SubBytes-transformation clock cycles. The number of clock cycles is about 97% smaller than that of the CAM-less massive-parallel SIMD matrix processor.

For comparison to the processing speed requirements

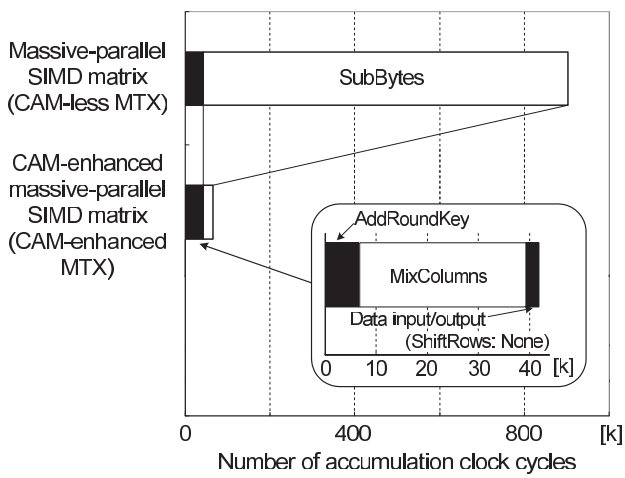
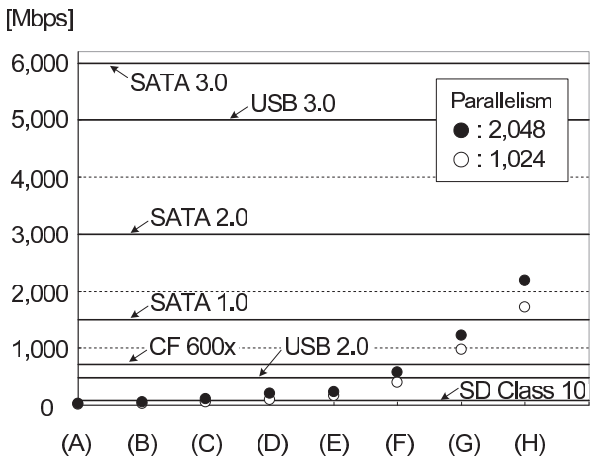


Fig. 12 AES algorithm implementation results for the CAM-less massive-parallel SIMD matrix and the CAM-enhanced massive-parallel SIMD matrix.

	Architecture	Operation frequency [MHz]	Throughput [Mbps]	
			1,024 PEs	2,048 PEs
(A)	MTX (Evaluation board)	81	12	24
(B)	MTX	200	29	58
(C)	MX-2 (Normal frequency mode)	300	55	110
(D)	MX-2 (Double frequency mode)	560	102	205
(E)	CAM-enhanced MTX	81	160	234
(F)	CAM-enhanced MTX	200	396	579
(G)	CAM-enhanced MX-2 (Normal frequency mode)	300	981	1,221
(H)	CAM-enhanced MX-2 (Double frequency mode)	560	1,721	2,192

(a) AES processing throughput of the massive-parallel SIMD matrix architectures



(b) Bit transfer rate for the massive-parallel SIMD matrix architectures and data storage standards

Fig. 13 Comparison of AES algorithm processing capability: (a) Processing throughput of the massive-parallel SIMD matrix architectures, (b) Bit transfer rate for the massive-parallel SIMD matrix architectures and data storage standards.

of several data storage transfer specifications, the necessary values of encryption throughput for the AES algorithm are calculated by eight implementation conditions of the SIMD matrix (MTX) architectures, namely: (A) an evaluation board-implemented MTX (81 MHz), (B) an MTX operated at normal frequency [11] (200 MHz), (C) a normal frequency mode MX-2 [16] (300 MHz), which expands the PE capability to 4-bits per clock, (D) a double frequency mode MX-2 [16] (560 MHz), (E) a CAM-enhanced MTX [15] (81 MHz), (F) a CAM-enhanced normal frequency MTX [15] (200 MHz), (G) a normal frequency mode CAM-enhanced MX-2 (300 MHz) and (H) a double frequency mode CAM-enhanced MX-2 (560 MHz). The table of Fig. 13-(a) shows the encryption throughput efficiency expressed in Mega bit per second (Mbps) for 1,024 and 2,048 parallelism of above eight architectures. As expected, the 2,048 PEs double the performance in comparison to 1,024 PEs ((A) - (D)). Figure 13-(b) shows the performance data of the massive-parallel SIMD matrix architectures, in comparison to selected-specifications of four recent data-transfer standards indicated by horizontal lines. Selected standards are: SD class 10 (SD association [25]–[27]), CF 600x (Compact Flash Association [28]–[30]), SATA 1.0 (SATA-IO [31]) and USB 2.0 (USB-IF [32]). The CAM-less SIMD

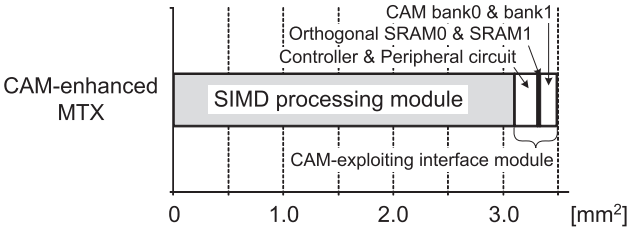


Fig. 14 Result for implementation area of the CAM-enhanced massive-parallel SIMD matrix as soft/hard macro in 90 nm CMOS technology.

matrix-processors are just able to cover the SD class 10 standard ((A) - (D)), while the CAM-enhanced SIMD matrix processors ((E) - (H)) can satisfy AES-encoding/decoding for data-transfer standards up to the level of SATA 1.0, as illustrated in Fig. 13-(b). Since many peripheral equipments and storage devices, such as HDD (Hard Disk Drive) or SSD (Solid State Drive), adopt above standards for their input/output interface, hardware encryption for private data is increasingly important commercially. Recently, next-generation standards, such as SATA2.0, SATA 3.0 and USB 3.0 are being discussed. As the CAM-less/enhanced massive-parallel SIMD matrix architecture has a freely scalable parallel processing structure, its parallelism can be easily extended by increasing the number of PEs and the SRAM storage area. Thus, the massive-parallel SIMD matrix architecture can also be adopted to future higher encrypting or decrypting speeds. On the other hand, the extended-parallelism of the massive-parallel SIMD matrix leads to more latency of the encryption and therefore requires larger buffer memories. This higher-parallelism-related drawback may be solved by interleaved processing with two or more SIMD matrix processors. This allows some SIMD matrix processors to receive new data for encoding from the system bus while other massive-parallel SIMD matrix processors are executing the AES algorithm for previous data.

Figure 14 shows the example of the total area consumption of a CAM-enhanced MTX design in 90 nm CMOS technology. The area amount of the SIMD processing module is about 3.1 mm^2 as described in Sect. 2. On the other hand, the total area consumption of the CAM-exploiting interface module in this semicustom design study becomes about 0.39 mm^2. The storage capacities for orthogonal SRAM bank and CAM bank are only 32 words with 32 bits word-length and 256 words with 32 bits word-length, respectively. These memory blocks are implemented by hard-macro VLSI designs. The area amount of the controller with peripheral circuits is estimated by soft-macro implementation. In comparison to the interface circuit without the CAM enhancement, the CAM-exploiting interface module is only increased by 0.16 mm^2. The total area of the CAM-enhanced MTX would become 3.49 mm^2, meaning only about 5% increased area overhead in comparison to the CAM-less MTX in this semicustom design study.

The power dissipation of the CAM-exploiting interface module is estimated at about 32 mW which is about 20% (or more less) of the total SIMD module without CAM en-

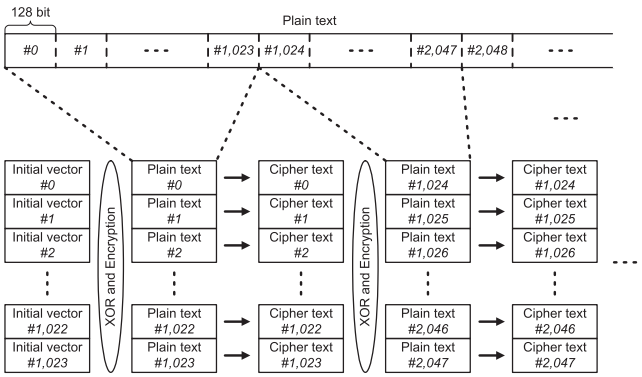


Fig. 15 Basic concept of interleaved CBC mode processing with the massive-parallel SIMD matrix (The number of interleaved positions N = 1,024).

hancement(150 mW or more). This shows that the power dissipation of the additional CAM part is much smaller than that of the SIMD module without CAM enhancement. Furthermore, the SIMD matrix processing and the CAM-interface processing never occur in parallel, so that the actual power-dissipation increase is reduced even more. Therefore, the power consumption of the CAM-enhanced massive parallel SIMD matrix is expected to increase only by a small amount in comparison to the CAM-less massive parallel SIMD matrix.

Consequently, the additional costs with respect to both silicon area and power dissipation are relatively small for a CAM-enhanced version of the SIMD matrix processor.

Up to now, we have reported in detail description about the AES-ECB (Electronic CodeBook) operation mode of the AES standard. Generally, the AES algorithm has the possibility of five block cipher modes, such as ECB mode, CBC (Cipher Block Chaining) mode, CFB (Cipher-FeedBack) mode, OFB (Output-FeedBack) mode and CTR (CounTeR) mode, which can improve the security of the encryption. Since the massive-parallel SIMD matrix architecture has flexible programming ability for various multimedia algorithms, each of the AES modes can be easily implemented in the massive-parallel SIMD matrix processor without additional hardware resources. We verified this for the CBC mode and the CTR mode by using a C language-based MTX simulator. While the CTR mode is directly suitable for parallel processing, the CBC mode consists of essentially a sequential chain of operations and is more difficult to adapt for parallel processing. For improving the processing speed of the CBC mode, an interleaved CBC mode [33], [34] has been developed which adopts ANSI X3.106 and ISO 10116. The interleaved CBC mode can execute parallel encryption processing and can be implemented in the massive-parallel SIMD matrix, as shown in Fig. 15. All SRAM entries of the massive-parallel SIMD matrix store N blocks of plain text, which are located at N positions with N different initial vectors. These N blocks are encrypted independently by the ECB encryption algorithm. The encryption of the next N blocks can start as soon as the previous blocks from N positions have been encrypted. With these refinements the same processing performance as in ECB mode can be achieved. Consequently, the massive-parallel SIMD matrix is clearly effective for fast parallel execution of the practically important CBC and CTR modes of the AES encryption processing.

The operating power consumption of the MTX and the MX-2 architecture have been reported to be 250 mW [11] and 450 mW [16], respectively. In comparison, the USB 2.0 standard, for example, has requirements for maximum power consumptions from 500 to 4,500 mW, and is used for many storage devices. Therefore, the MTX or the MX-2 architectures can be smoothly applied in many data storage devices.

As a result from above evaluations, the massive-parallel SIMD matrix architecture can realize a high-throughput encryption/decryption solution for protecting binary data in private storage devices. In particular, the CAM-enhanced massive-parallel SIMD matrix architecture is very suitable as an efficient cryptographic processor.

6. Conclusion

In this paper, a software-based parallel cryptographic solution with a CAM-less and a CAM-enhanced massive-parallel memory-embedded SIMD matrix processor is proposed, which can realize efficient encrypting and decrypting cryptographic algorithms. For the AES algorithm, which is studied as an example, an encryption throughput of up to 2.19 Gbps can be achieved. This performance enables the architecture to cover several standard data-storage-transfer specifications. Consequently, the massive-parallel SIMD matrix architecture is very effective for private information protection in several data-storage media, and is a promising solution for low power dissipation and small Si-area consumption of integrated real-time realizations for cryptographic algorithms.

Acknowledgements

The reported results are obtained as part research collaboration with Renesas Electronics Co. We are deeply thankful to T. Gyohten, H.Noda, Y. Okuno and K. Arimoto of Renesas Technology Co. for their support. Part of this work has been supported by the program “Interdisciplinary Research on Integration of Semiconductor and Biotechnology” for “Creation of Innovation Centers for Advanced Interdisciplinary Research Areas”, a Grant-in-Aid for Young Scientists (B) (No.16700184), Ministry of Education, Culture, Sports, Science and Technology, Japanese government and a Grant-in-Aid for JSPS Fellows, 175303, 2005.

References

[1] http://www.oecd.org/document/18/0,3343,en_2649_34255_1815186_1_1_1_1,00.html
[2] http://www.kingston.com/flash/DataTravelers_enterprise.asp
[3] http://www.iodata.jp/product/usbmemory/easydisk/ (in Japanese).

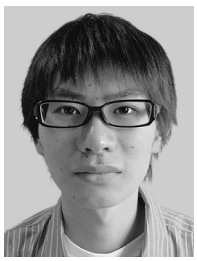
[4] http://sdd.toshiba.com/main.aspx?Path=StorageSolutions/PCNotebookHardDrives/MJA2xxxCHSeries
[5] http://www.seagate.com/ww/v/index.jsp?locale=en-US&name=dn_sec_intro_fde&vgnextoid=1831bb5f5ed93110VgnVCM100000f5ee0a0aRCRD
[6] "Announcing approval of the withdrawal of Federal Information Processing Standard (FIPS) 46-3, Data Encryption Standard (DES); fips 74, guidelines for implementing and using the NBS data encription standard; and FIPS 81, DES modes of operation," National Intstitute of Standards and Technology (NIST), Commerce, vol.70, no.90, pp.28907–28908, May 2005.
[7] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management — part 1: General (revised)," National Intstitute of Standards and Technology (NIST), Special Publication, no.SP 800-57, March 2007.
[8] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management — part 2: Best practices for key management organization," National Intstitute of Standards and Technology (NIST), Special Publication, no.SP 800-57, March 2007.
[9] E. Barker, W. Burr, A. Jones, T. Polk, S. Rose, M. Smid, and Q. Dang, "Recommendation for key management — part 3: Application-specific key management guidance," National Intstitute of Standards and Technology (NIST), Special Publication, no.SP 800-57, March 2007.
[10] Y. Hirose, M. Saito, and W.E. Xouzijn, "Embedded micro processor with reconfigurable functional unit," IEICE Trans. Electron. (Japanese Edition), vol.J76-C, no.8, pp.808–816, Aug. 2003.
[11] M. Nakajima, H. Noda, K. Dosaka, K. Nakata, M. Higashida, O. Yamamoto, K. Mizumoto, H. Kondo, Y. Shimazu, K. Arimoto, K. Saitoh, and T. Shimizu, "A 40 GOPS 250 mW massively parallel processor based on matrix architecture," ISSCC Dig. Tech. Papers, pp.410–412, Feb. 2006.
[12] T. Tanizaki, T. Gyohten, H. Noda, M. Nakajima, K. Mizumoto, and K. Dosaka, "A super parallel SIMD processor with time/space conversion bus bridge on the matrix architecture," IEICE Technical Report, ICD2006-79, Aug. 2006 (in Japanese).
[13] H. Noda, T. Tanizaki, T. Gyohten, K. Dosaka, M. Nakajima, K. Mizumoto, K. Yoshida, T. Iwao, T. Nishijima, Y. Okuno, and K. Arimoto, "The circuits and robust design methodology of the massively parallel processor based on the matrix architecture," Symp. VLSI Circuits Dig. Tech. Papers, pp.260–261, June 2006.
[14] T. Kumaki, M. Ishizaki, T. Koide, H.J. Mattausch, Y. Kuroda, H. Noda, K. Dosaka, K. Arimoto, and K. Saito, "Acceleration of DCT processing with massive-parallel memory-embedded SIMD matrix processor," IEICE Trans. Inf. & Syst., vol.E90-D, no.8, pp.1312–1315, Aug. 2007.
[15] T. Kumaki, T. Koide, H.J. Mattausch, Y. Kuroda, T. Gyohten, H. Noda, K. Dosaka, K. Arimoto, and K. Saito, "Integration architecture of content addressable memory and massive-parallel memory-embedded SIMD matrix for versatile multimedia processor," IEICE Trans. Electron., vol.E91-C, no.9, pp.1409–1418, Sept. 2008.
[16] T. Kurafuji, M. Haraguchi, M. Nakajima, T. Gyoten, T. Nishijima, H. Yamasaki, Y. Imai, M. Ishizaki, T. Kumaki, Y. Okuno, T. Koide, H.J. Mattausch, and K. Arimoto, "A scalable massive parallel processor for real-time image processing," ISSCC Dig. Tech. Papers, pp.15–17, Feb. 2010.
[17] M. Tagami, M. Ishizaki, T. Kumaki, Y. Kono, T. Koide, H.J. Mattausch, T. Gyohten, H. Noda, K. Dosaka, K. Arimoto, and K. Saito, "Acceleration of advanced excryption standard (AES) processing on a CAM enhanced super parallel SIMD processor," Proc. 14th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI2007), pp.74–80, Oct. 2007.
[18] I. Kuroda and S. Kyo, "Media processing LSI architectures for automotives —Challenges and future trends—," IEICE Trans. Electron., vol.E90-C, no.10, pp.1850–1857, Oct. 2007.
[19] H. Amano, "A survey on dynamically reconfigurable processors," IEICE Trans. Commun., vol.E89-B, no.12, pp.3179–3187, Dec. 2006.
[20] Y. Kono, T. Kumaki, M. Ishizaki, M. Tagami, T. Koide, H.J. Mattausch, T. Gyohten, H. Noda, Y. Kuroda, K. Dosaka, K. Arimoto, and K. Saito, "Super parallel SIMD processor with CAM based high-speed pattern matching capability," IEICE Technical Report, ICD2006-116, Oct. 2006 (in Japanese).
[21] T. Kumaki, Y. Kono, M. Ishizaki, M. Tagami, T. Koide, H.J. Mattausch, T. Gyohten, H. Noda, Y. Kuroda, K. Dosaka, K. Arimoto, and K. Saito, "CAM enhanced super parallel SIMD processor with high-speed pattern matching capability," Proc. IEEE International Midwest Symposium on Circuits And Systems (MWSCAS'07), pp.803–806, Aug. 2007.
[22] T. Kumaki, Y. Kuroda, M. Ishizaki, T. Koide, H.J. Mattausch, H. Noda, K. Dosaka, K. Arimoto, and K. Saito, "Real-Time Huffman encoder with pipelined CAM-based data path and code-word-table optimizer," IEICE Trans. Inf. & Syst., vol.E90-D, no.1, pp.334–345, Jan. 2007.
[23] T. Kumaki, Y. Kono, M. Ishizaki, T. Koide, and H.J. Mattausch, "Scalable FPGA/ASIC implementation architecture for parallel table-lookup-coding using multi-ported content addressable memory," IEICE Trans. Inf. & Syst., vol.E90-D, no.1, pp.346–354, Jan. 2007.
[24] "Announcing the advanced encryption standard (AES)," Federal Information Processing Standards Publications 197 (FIPS 197), National Intstitute of Standards and Technology (NIST), Nov. 2001.
[25] http://www.sdcard.org/developers/tech/speed_class/
[26] http://shop.panasonic.co.uk/invt/rpsdw32ge1k
[27] http://www.sandisk.com/products/imaging/sandisk-extreme-sdhc-cards-
[28] http://www.compactflash.org/
[29] https://w1.broadserver.jp/~gaaum000/products/products_0.html
[30] http://www.transcendusa.com/Products/ModDetail.asp?ModNo=252&LangNo=0&Func1No=&Func2No=
[31] http://www.serialata.org/index.asp
[32] http://www.usb.org/home
[33] B. Schneier, Applied cryptography, 2nd edition, John Wiley & Sons, pp.210–211, 1996.
[34] J. Viega and M. Messier, "Secure programming cookbook for C and C++," O'Reilly Media, pp.208–211, July 2003.

Takeshi Kumaki received a B.S. degree from the Department of mathematics, Faculty of Science and completed the first half of the M.E. program in Information Mathematics from National Defence Academy, Kanagawa, Japan in 1998 and 2003, respectively, and Ph.D., degree in electric engineering from Hiroshima University, Hiroshima, Japan in 2006. From 2003 to 2004, he was affiliated with the Japan Air Self-Defence Force Electric Experimentation Group. From 2005 to 2009, he joined the Research Center for Nanodevices and Systems (RCNS) and the Research Institute for Nanodevice and Bio Systems (RNBS), Hiroshima University, Japan, where he has engaged in the system design and architecture research. Since 2010, he has been an Assistant Professor in the department of VLSI system design, Ritsumeikan University. He is interested in content addressable memory, SIMD processing architecture and these applications. Dr. Kumaki is a member of the Institute of Electrical and Electronics Engineers (IEEE).

Tetsushi Koide received the B.E. degree in Physical Electronics, the M.E. and the Ph.D. degrees in Systems Engineering from Hiroshima University in 1990, 1992, and 1998, respectively. He was a Research Associate and an Associate Professor in the Faculty of Engineering at Hiroshima University in 1992 - 1999 and 1999, respectively. From 1999 to 2001 he was with the VLSI Design and Education Center (VDEC), The University of Tokyo as an Associate Professor. From 2001 to 2008 he was an Associate Professor in the Research Center for Nanodevices and Systems, Hiroshima University. Since 2008 he has been an Associate Professor in the Research Institute for Nanodevice and Bio Systems (RNBS) and Graduate School of Advanced Sciences of Matter, Hiroshima University. His research interests include system design and architecture issues for memory-based systems, real-time image processing, VLSI CAD/DA, genetic algorithms, and combinatorial optimization. Dr. Koide is a member of the Institute of Electrical and Electronics Engineers, the Association for Computing Machinery, and the Information Processing Society of Japan.

Hans Jürgen Mattausch is a Professor at the Research Institute for Nanodevice and Bio Systems and the Graduate School for Advanced Sciences of Matter, Hiroshima University, Higashi-hiroshima, Japan. He received the Dr. rer. nat. degree from the University of Stuttgart, Germany in 1981. From 1982 to 1995 he was with the Research Laboratories of Siemens AG in Munich, Germany, where he was involved in the development of CMOS technology, memory and telecommunication circuits, power semiconductor devices and compact modeling. From 1995 to 1996 he was with the Siemens Semiconductor Group as Department Head for Product Analysis and Improvement in the Chip-Card IC Division. Since 1996 he is with Hiroshima University. Dr. Mattausch is a senior member of IEEE (Institute of Electrical and Electronics Engineers).

Masaharu Tagami was born in Miyazaki, Japan in 1985. He received his M.S. degree in Advanced Sciences of Matter from Hiroshima University, Japan in 2009. He joined Renesas Technology Corp., Japan, in 2009 and transferred to Renesas Electronics Corp., Japan, in 2010.



Masakatsu Ishizaki received the B.E. and M.E. degrees in Electronic Engineering from Hiroshima University, Japan in 2006 and 2008 respectively. In 2008, he joined the System Core Development Division, Renesas Technology Corp., Hyogo, Japan. In 2010, he transferred to Renesas Electronics Corp. He is engaged in the development of CPU.