A High Throughput/Gate AES Hardware Architecture by Compressing Encryption and Decryption Datapaths

— Toward Efficient CBC-Mode Implementation

Rei Ueno[1(✉)], Sumio Morioka[2], Naofumi Homma[1], and Takafumi Aoki[1]

[1] Tohoku University, Aramaki Aza Aoba 6–6–05, Aoba-ku, Sendai-shi 980-8579, Japan
ueno@aoki.ecei.tohoku.ac.jp

[2] Central Research Laboratories, NEC Corporation, Athene, Odyssey Business Park, West End Road, South Ruislip, Middlesex HA4 6QE, UK

Abstract. This paper proposes a highly efficient AES hardware architecture that supports both encryption and decryption for the CBC mode. Some conventional AES architectures employ pipelining techniques to enhance the throughput and efficiency. However, such pipelined architectures are frequently unfit because many practical cryptographic applications work in the CBC mode, where block-wise parallelism is not available for encryption. In this paper, we present an efficient AES encryption/decryption hardware design suitable for such block-chaining modes. In particular, new operation-reordering and register-retiming techniques allow us to unify the inversion circuits for encryption and decryption (i.e., SubBytes and InvSubBytes) without any delay overhead. A new unification technique for linear mappings further reduces both the area and critical delay in total. Our design employs a common loop architecture and can therefore efficiently perform even in the CBC mode. We also present a shared key scheduling datapath that can work on-the-fly in the proposed architecture. To the best of our knowledge, the proposed architecture has the shortest critical path delay and is the most efficient in terms of throughput per area among conventional AES encryption/decryption architectures with tower-field S-boxes. We evaluate the performance of the proposed and some conventional datapaths by logic synthesis results with the TSMC 65-nm standard-cell library and NanGate 45- and 15-nm open-cell libraries. As a result, we confirm that our proposed architecture achieves approximately 53–72 % higher efficiency (i.e., a higher bps/GE) than any other conventional counterpart.

Keywords: AES · Hardware architectures · Unified encryption/decryption architecture · CBC mode

B. Gierlichs and A.Y. Poschmann (Eds.): CHES 2016, LNCS 9813, pp. 538–558, 2016.
DOI: 10.1007/978-3-662-53140-2_26

1 Introduction

Cryptographic applications have been essential for many systems with secure communications, authentication, and digital signatures. In accordance with the rapid increase in Internet of Things (IoT) applications, many cryptographic algorithms are required to be implemented in resource-constrained devices and embedded systems with a high throughput and efficiency. Since 2001, many hardware implementations for AES have been proposed and evaluated for CMOS logic technologies. Studies of AES design are important from both practical and academic perspectives since AES employs an SPN structure and the major components (i.e., an 8-bit S-box and permutation used in ShiftRows and MixColumns) followed by many other security primitives.

AES encryption and decryption are commonly used in block-chaining modes such as CBC, CMAC, and CCM (e.g., for SSL/TLS, IEEE802.11 wireless LAN, and IEEE802.15.4 wireless sensor networks). Therefore, AES architectures that efficiently perform both encryption and decryption in the above block-chaining modes are highly demanded. However, many conventional architectures employ pipelining techniques to enhance the throughput and efficiency [13,15,17], although such block-wise parallelism is not available in the above block-chaining modes. For example, the highest throughput of 53 Gbps was achieved in the previous best encryption/decryption architecture [17], but it only worked in the ECB mode. In addition, these previous studies assumed offline key scheduling owing to the difficulty of on-the-fly scheduling. On-the-fly key scheduling should be implemented in most resource-constrained devices because an offline key scheduling implementation requires additional memory to store expanded round keys. Thus, it is valuable to investigate an efficient AES architecture with on-the-fly key scheduling without any pipelining technique.

In this paper, we present a new round-based AES architecture for both encryption and decryption with on-the-fly key scheduling, which achieves the lowest critical path delay (the least number of serially connected gates in the critical path) with less area overhead compared to conventional architectures with tower-field S-boxes. Our architecture employs new operation-reordering and register-retiming techniques to unify the inversion circuits for encryption and decryption without any selectors. In addition, these techniques make it possible to unify the affine transformation and linear mappings (i.e., the isomorphism and constant multiplications) to reduce the total number of logic gates. The proposed and conventional AES encryption/decryption datapaths are synthesized and evaluated with the TSMC standard-cell and NanGate open-cell libraries. The evaluation results show that our architecture can perform both (CBC-) encryption and decryption more efficiently. For example, the throughput per gate of the proposed architecture in the NanGate 15-nm process is 72 % larger than that of the best conventional architecture.

The rest of this paper is organized as follows: Sect. 2 introduces related works on AES hardware architectures, especially those with round-based encryption and decryption. Section 3 presents a new AES hardware architecture based on our operation-reordering, register-retiming, and affine-transformation unification

techniques. Section 4 evaluates the proposed datapath by the logic synthesis compared with conventional round-based datapaths. Section 5 discusses variations of the proposed architecture. Finally, Sect. 6 contains our conclusion.

2 Related Works

2.1 Unified AES Datapath for Encryption and Decryption

Architectures that perform one round of encryption or decryption per clock cycle without pipelining are the most typical for AES design and are called round-based architectures in this paper. Round-based architectures can be implemented more efficiently in terms of throughput per area than other architectures by utilizing the inherent parallelism of symmetric key ciphers. For example, the byte-serial architecture [16,18] is intended for the most compact and low-power implementations such as in RFID but is not intended for the high throughput and efficiency. In contrast, round-based architectures are suitable for a high throughput per gate, which leads to a low-energy implementation [29].

To design such round-based encryption/decryption architectures in an efficient manner, we consider how to unify the resource-consuming components such as the inversion circuits in SubBytes/InvSubBytes for the encryption and decryption datapaths. There are two conventional approaches for designing such unified datapaths. The first approach is to place two distinct datapaths for encryption and decryption and select one of the datapaths with multiplexers as in [15]. Figure 1 shows an overview of the datapath flow in [15], where the inversion circuit is shared by both paths, and additional multiplexers are used at the input and output of the encryption and decryption paths. In [15], a reordered decryption operation was introduced as shown in Fig. 2. The intermediate value is stored in a register after InvMixColumns instead of AddRoundKey. Such register retiming was suitable for pipelined architectures. The main drawbacks of such approaches are the false critical path delay and the required area and delay overheads caused by three multiplexers. The critical path of the datapath in Fig. 1 is denoted in bold, which would never be active because it passes from the decryption path to the encryption path. This false critical path reduces the maximum operation frequency owing to logic synthesis due to the false longest logic chain. The overhead caused by the multiplexers is also nonnegligible for common standard-cell-based designs.

The second approach is to unify the circuits of the functions SubBytes, ShiftRows, and MixColumns with their inverse functions, respectively. Figure 3 shows the datapath in [29] where encryption and decryption paths are combined using the second approach, where the reordering technique is given in Fig. 4. The order of the decryption operations is changed to be the same as that of the encryption operations. Note that the order of (Inv)SubBytes and (Inv)ShiftRows can be changed without any overhead, and the datapath in [29] changes the order of SubBytes and ShiftRows in the encryption. The reordering of AddRoundKey and InvMixColumns utilizes the linearity of InvMixColumns as follows: $MC^{-1}(M_r + K_r) = MC^{-1}(M_r) + MC^{-1}(K_r)$, where MC^{-1} is

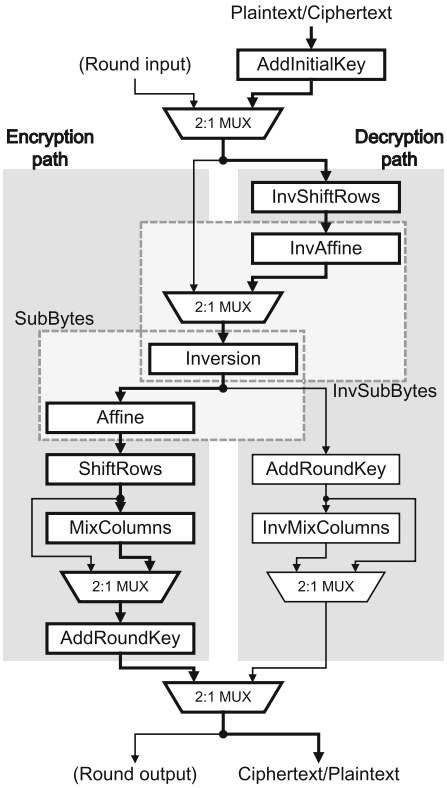

Fig. 1. Conventional parallel datapath in [15].

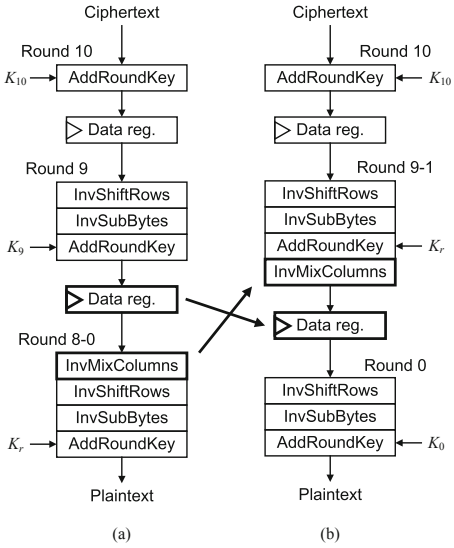

Fig. 2. Register-retiming techniques in [15]: (a) original and (b) resulting decryption flows.

the function InvMixColumns, and M_r and K_r are the intermediate value after InvShiftRows and the round key at the r-th round, respectively. Here, InvMixColumns requires the round keys, whereas MixColumns and InvMixColumns can be unified to reduce the area. Therefore, this type of architecture requires an additional InvMixColumns to compute $MC^{-1}(K_r)$ for decryption. In addition, the false path and multiplexer overhead exist because each function and its inverse function are implemented in a partially serial manner with multiplexers like SubBytes and InvSubBytes in Fig. 1, where the critical path consists of Affine, Inversion, InvAffine, and an additional multiplexer.

The architecture in [17] employs a reordering technique similar to [29]. The major difference is the intermediate value stored in the register. The architecture in [14] also employs the same approach that combines the encryption and decryption datapaths, but does not change the order of AddRoundKey and InvMixColumns to remove InvMixColumns to compute $MC^{-1}(K_r)$. As a result, an additional selector is required to unify MixColumns and InvMixColumns.

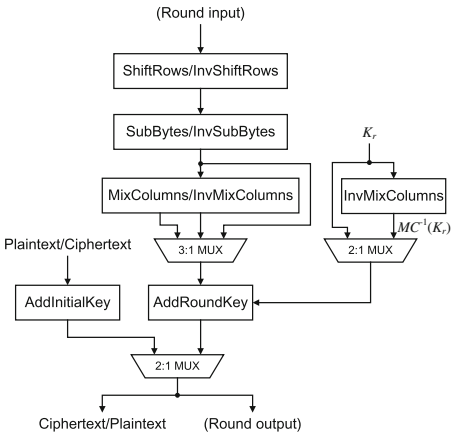

Fig. 3. Conventional datapath in [29], where encryption and decryption paths are combined.

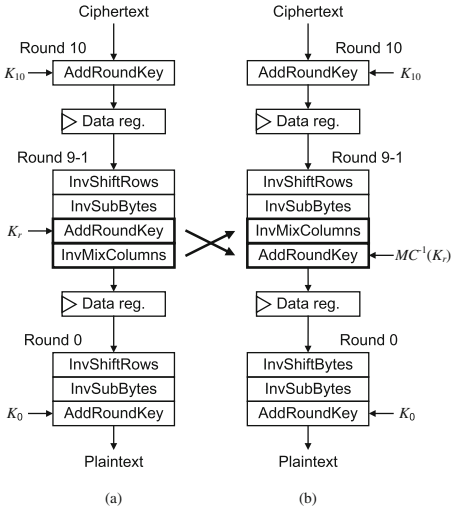

Fig. 4. Reordering technique in [29]: decryption flows (a) before and (b) after reordering.

As described above, sharing inversion circuits is essential for designing efficient AES hardware. Although a hardware T-box architecture such as that in [20] is also useful for a high-throughput implementation, it is not applicable to the above shared datapath owing to the lack of sharable components between the encryption and decryption paths.

2.2 Inversion Circuit Design and Tower-Field Arithmetic

The design of the inversion circuit used in (Inv)SubBytes has a significant impact on the performance of AES implementations. Many inversion circuit designs have been proposed. There are two major approaches using direct mapping and tower-field arithmetic. Inversion circuits based on direct mapping such as table-lookup, Binary Decision Diagram (BDD), and Positive-Polarity Reed-Muller (PPRM) [15,19,20] are faster but larger than those based on a tower field. On the other hand, tower-field arithmetic enable us to design more compact and more area-time efficient inversion circuits in comparison with direct mapping. Therefore, we focus on inversion circuits based on tower-field arithmetic in this paper.

The performance of tower-field-based inversion circuits varies with the field towering and Galois field (GF) representation. After the introduction of tower-field inversion over $GF(((2^2)^2)^2)$ based on a polynomial basis (PB) by Satoh et al. [29], Canright reduced the gate count using a normal basis-(NB-)based $GF(((2^2)^2)^2)$, which has been known as the smallest for a long time [7], Nogami et al. showed that a mixture of a PB and an NB was useful for a more efficient design [23]. On the other hand, Rudra et al., Joen et al., and Mathew et al.

designed inversion circuits using PB-based $GF((2^4)^2)$, which have a smaller critical path delay than those based on $GF(((2^2)^2)^2)$ [12,17,27]. Nekado et al. showed that a redundantly represented basis (RRB) was useful for an efficient design [21]. Recently, Ueno et al. designed an inversion circuit based on the combination of an NB, an RRB, and a polynomial ring representation (PRR), which is known as the most area-time efficient inversion [31]. In addition, a logic minimization technique was applied to Canright's S-box, which resulted in a more compact S-box [6].

To embed such a tower-field-based inversion circuit in AES hardware, an isomorphic mapping between the AES field and the tower field is required because the inversion and MixColumns are performed over the AES field (i.e., PB-based $GF(2^8)$ with an irreducible polynomial $x^8+x^4+x^3+x+1$). Typically, the input into the inversion circuit (in the AES field) is initially mapped to the tower field by the isomorphic mapping. After the inversion operation over the tower field, an inverse isomorphic mapping (and affine transformation) are applied [29]. On the other hand, some architectures perform all of the AES subfunctions (i.e., SubBytes as well as ShiftRows, MixColumns, and AddRoundKey) over the tower field, where isomorphic mapping and its inverse mappings are performed at the timings of the data (i.e., plaintext and ciphertext) input and output, respectively [10,16–18,27]. In other words, the cost of field conversion is suppressed when the conversion is performed only once during encryption or decryption. However, the cost of constant multiplications in MixColumns over a tower field is worse than that over the AES field while inversion is efficiently performed over the tower field. More precisely, in tower-field architectures, such linear mappings including constant multiplications usually require $3T_{XOR}$ delay, where T_{XOR} indicates the delay of an XOR gate [21]. The XOR gate count used in (Inv)MixColumns over a tower field is also worse than that over AES field.

3 Proposed Architecture

This section presents a new round-based AES architecture that unifies the encryption and decryption paths in an efficient manner. The key ideas for reducing the critical path delay are summarized as follows: (1) to merge linear mappings such as MixColumns and isomorphic mappings as much as possible by reordering subfunctions, (2) to minimize the number of selectors to unify the encryption and decryption paths by the above merging and a register retiming, and (3) to perform isomorphic mapping and its inverse mappings only once in the pre- and post-round datapaths. We can reduce the number of linear mappings to at most one for each round operation as the effect of (1). Moreover, we can reduce the number of selectors to only one (4-to-1 multiplexer) in the unified datapath as the effect of (2) while the inversion circuit is shared by the encryption and decryption paths. From the idea of (3), we can remove the isomorphic mapping and its inverse mappings from the critical path. Figure 5 shows the overall architecture that consists of the round function and key scheduling parts. Our architecture performs all of the subfunctions over a tower field for

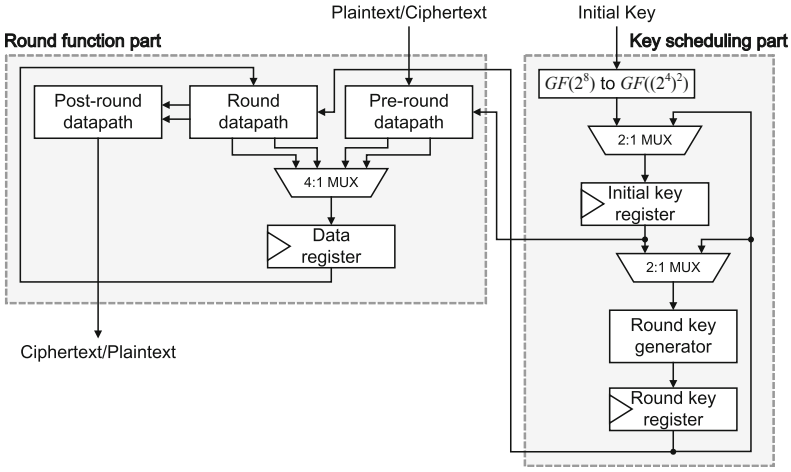

Fig. 5. Overall architecture of proposed AES hardware.

both the round function and key scheduling parts and therefore applies isomorphic mappings between the AES and tower fields in the datapaths of the pre- and post-round operations, which are represented as the blocks "Pre-round datapath" and "Post-round datapath" in Fig. 5. "Round datapath" performs one round operation for either encryption or decryption.

3.1 Round Function Part

The proposed architecture employs a unified datapath for encryption and decryption as in [15] and applies new operation-reordering and register-retiming techniques to address the conventional issues of a false critical path and additional multiplexers. Using our operation-reordering technique and then merging linear mappings, we can reduce the number of linear mappings on the critical path of the round datapath to at most one. Our reordering technique also allows to unify the linear mappings and affine transformation in a round. The unification of these mappings can drastically reduce the critical path delay and the XOR-gate count of linear mappings, even in a tower-field architecture.

The new operation reordering is derived as follows. First, the original round operation of AES encryption is represented by the following equation:

$$\begin{aligned} m_{i,j}^{(r+1)} &= u_{-i}S(m_{0,i+j}^{(r)}) + u_{1-i}S(m_{1,i+j}^{(r)}) + u_{2-i}S(m_{2,i+j}^{(r)}) + u_{3-i}S(m_{3,i+j}^{(r)}) + k_{i,j}^{(r)} \\ &= \sum_{e=0}^{3}(u_{e-i}S(m_{e,i+j}^{(r)})) + k_{i,j}^{(r)}, \end{aligned} \tag{1}$$

where $m_{i,j}^{(r)}$ and $k_{i,j}^{(r)}$ are the i-th row and j-th column intermediate value and round key at the r-th round, except for the final round. Note that the subscripts of each variable are a member of $\mathbb{Z}/4\mathbb{Z}$. The function S indicates the 8-bit S-box, and u_0, u_1, u_2, and u_3 are the coefficients of the matrix of MixColumns

and respectively given by $\beta, \beta+1, 1$, and 1, where β is the indeterminate of $GF(2^8)$ satisfying $\beta^8+\beta^4+\beta^3+\beta+1=0$. We can rewrite Eq. (1) by decomposing S into inversion and affine transformation as follows:

$$m_{i,j}^{(r+1)} = \sum_{e=0}^{3}(u_{e-i}(A(\left(m_{e,i+j}^{(r)}\right)^{-1}) + c)) + k_{i,j}^{(r)}, \tag{2}$$

where A is the linear mapping of the affine transformation, and c ($=\beta^6+\beta^5+\beta+1$) is a constant. In the case of tower-field architectures, Eq. (2) is represented by

$$m_{i,j}^{(r+1)} = \sum_{e=0}^{3}(u_{e-i}(A(\Delta'(\left(\Delta(m_{e,i+j}^{(r)})\right)^{-1})) + c)) + k_{i,j}^{(r)}, \tag{3}$$

where Δ is the isomorphic mapping from the AES field to a tower field, and Δ' is the inverse isomorphic mapping.

The linear mappings, which include an isomorphism and constant multiplications over the GF, are performed by the constant multiplication of the corresponding matrix over $GF(2)$. Therefore, we can merge such mappings to reduce the critical path delay and the number of XOR gates. In addition, we consider the variable $d_{i,j}^{(r)}$ of the tower field derived from $m_{i,j}^{(r)}$. Substituting $m_{i,j}^{(r)}$ with $\Delta'(d_{i,j}^{(r)})$ ($= m_{i,j}^{(r)}$), we can merge the linear mappings as follows:

$$d_{i,j}^{(r+1)} = \sum_{e=0}^{3}(U_{e-i}(\left(d_{e,i+j}^{(r)}\right)^{-1})) + \Delta(c) + \Delta(k_{i,j}^{(r)}), \tag{4}$$

where $U_e(x) = \Delta(u_e(A(\Delta'(x))))$. Note that an arbitrary linear mapping L satisfies $L(a+b) = L(a)+L(b)$. Thus, the linear mappings of a round in Eq. (4) can be merged into at most one, even with a tower-field S-box, whereas the linear mappings in Eq. (3) cannot be.

On the other hand, the corresponding equation for AES decryption with tower-field arithmetic is given by

$$d_{i,j}^{(r-1)} = \sum_{e=0}^{3}(\Delta(v_{e-i}(\Delta'(\left(\Delta(A'(\Delta'(d_{e,j-i}^{(r)}))) + \Delta(c')\right)^{-1} + \Delta(k_{e,j-i}^{(r)}))))), \tag{5}$$

where A' indicates the linear mapping of the inverse affine transformation. The coefficients v_0, v_1, v_2, and v_3 are respectively given by $\beta^3+\beta^2+\beta, \beta^3+\beta+1, \beta^3+\beta^2+1$, and β^3+1, and c' ($=\beta^2+1$) is a constant. Here, the linear mappings cannot be merged into one because they are performed both before and after the inversion operation. In addition, if we construct an encryption/decryption datapath based on Eqs. (4) and (5), the inversion circuit cannot be shared by encryption and decryption without a selector because the timings of the inversion operations are different from each other. Therefore, we consider a register retiming to store the intermediate value $s_{i,j}^{(r)}$ given after the inverse affine transformation over the

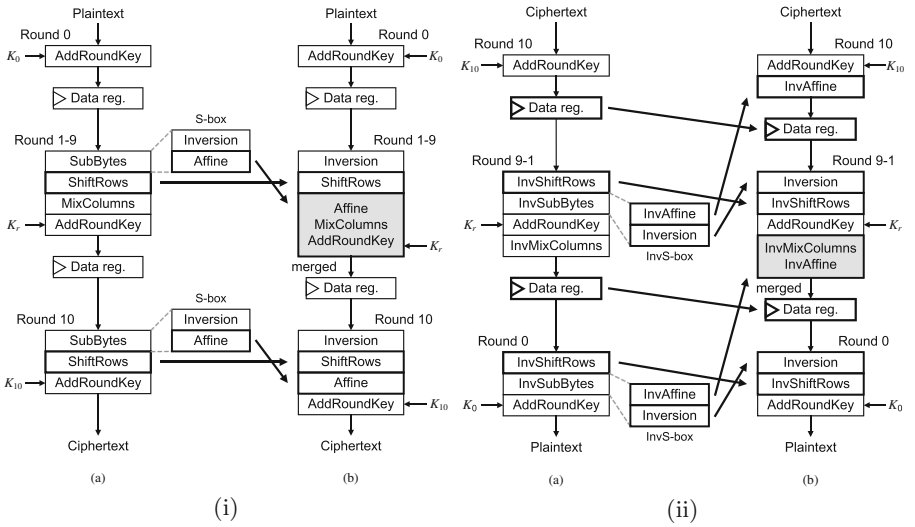

Fig. 6. Proposed (i) encryption and (ii) decryption flows (a) before and (b) after reordering and register-retiming.

tower-field. Here, $s_{i,j}^{(r)}$ is given by $s_{i,j}^{(r)} = \Delta(A'(\Delta'(d_{i,j}^{(r)}))) + \Delta(c')$. In the decryption, we store $s_{i,j}^{(r)}$ in the data register instead of $d_{i,j}^{(r)}$. Using $s_{i,j}^{(r)}$ and $s_{i,j}^{(r-1)}$, we rewrite Eq. (5) as follows:

$$s_{i,j}^{(r-1)} = \sum_{e=0}^{3} (V_{e-i}(\left(s_{e,j-i}^{(r)}\right)^{-1} + \Delta(k_{e,j-i}^{(r)}))) + \Delta(c'), \tag{6}$$

where $V_e(x) = \Delta(A'(v_e(\Delta'(x))))$.

Our round datapath is constructed with a minimal critical path delay according to Eqs. (4) and (6). Here, we further reorder the sequence of operations (i.e., subfunctions) to share inversion circuits without additional selectors and to unify the linear mappings. Figure 6 shows the proposed reordering technique. We first decompose SubBytes into the inversion and (Inv)Affine. In the encryption, Affine, MixColumns, and AddRoundKey can be merged by exchanging Affine and ShiftRows. In the decryption, the inversion circuit is located at the beginning of the round by exchanging the inversion and InvShiftRows. Thus, additional selectors for sharing the inversion circuit are not required thanks to the operation-reordering and register-retiming techniques. This is because both inversion operations are performed at the beginning of the round, which means that the data register output can be directly connected to the inversion circuit.

Figure 7 illustrates the proposed round function datapath with the unification of linear mappings. Our architecture employs only one 128-bit 4-in-1 multiplexer, whereas conventional ones employ several 128-bit multiplexers. For example, the

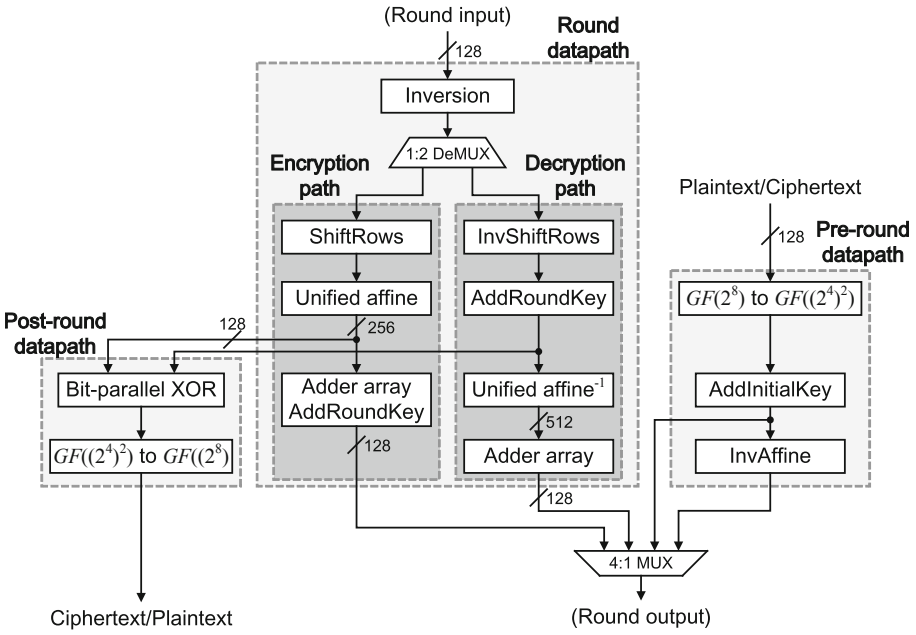

Fig. 7. Proposed round function part.

datapath in [14] employs seven 128-bit multiplexers[1]. Fewer selectors can reduce the critical path delay and circuit area and solve the false critical path problem. Unified affine and Unified affine^{-1} in Fig. 7 perform the unified linear mappings (i.e., $U_0, \ldots, U_3$ and $V_0, \ldots, V_3$) and constant addition. The number of linear mappings on the critical path is at most one in our architecture, whereas that of the conventional architectures is not. We can also suppress the overhead of constant multiplication over the tower field by the unification. Adder arrays in Fig. 7 consist of four 4-input 8-bit adders in MixColumns or InvMixColumns. In the encryption, the factoring technique for MixColumns and AddRoundKey [21] is available for Unified affine, which makes the circuit area smaller without a delay overhead. As a result, the data width between Unified affine and Adder array in Encryption path is reduced from 512 to 256 bits because the calculations of U_1 and U_3 are not performed in Encryption path. In addition, Adder array and AddRoundKey are unified in Encryption path because both of them are composed of 8-bit adders[2]. On the other hand, since there is no factoring technique for InvMixColumns without delay overheads, the data width from Unified affine^{-1} to Adder array in Decryption path is 512 bits. Finally, an inactive path can be disabled using a demultiplexer since our datapath is fully parallel after the inversion circuit. Thanks to the disabling, a multiplexer and AddRoundKey

[1] The selectors in SubBytes/InvSubBytes are included in the seven multiplexers.

[2] Some architectures such as [14,29] unify AddInitialKey and AddRoundKeys. We did not unify them to avoid increasing the number of selectors.

are unified as Bit-parallel XOR. (The addition of $\Delta(c)$ in Unified affine should be active only when encryption.) In addition, the demultiplexer would suppress power consumption due to a dynamic hazard. Although tower-field inversion circuits are known to be power-consuming owing to dynamic hazards [19], these hazards can be terminated at the input of the inactive path.

Our datapath employs the inversion circuit presented in [31] because it has the highest area-time efficiency among inversion circuits including one using a logic minimization technique [6]. We can merge the isomorphic mappings in order to reduce the linear function on the round datapath to only one, even if the inversion circuit has different GF representations at the input and output. Since the output is given by an RRB, the data width from Inversion to Unified affine (or Unified affine^{-1}) is given by 160 bits. However, AddRoundKey in the decryption path and Bit-parallel XOR in the post-round datapath are implemented respectively by only 128 XOR gates because the NB used as the input is equal to the reduced version of the RRB. In addition, a 1:2 DeMUX is implemented with NOR gates thanks to the redundancy, whereas nonredundant representations require AND gates.

3.2 Key Scheduling Part

The on-the-fly key scheduling part is shared by the encryption and decryption processes. For the encryption, the key scheduling part first stores the initial key in the initial key register (in Fig. 5) and then generates the round keys during the following clock cycles. For the decryption, the final round key should be calculated from the initial key and stored in the initial key register in advance. The key scheduling part then generates the round keys in the reverse order by the round key generator (in Fig. 5). However, conventional key scheduling datapaths such those as in [14,29] are not applicable to our round datapath because they have a loop with a false path and/or a longer true critical path than our datapath.

To address the above issue, we introduce a new architecture for the key scheduling datapath. For on-the-fly implementation, the subkeys are calculated for each of the four subkeys (i.e., 128 bits) in a clock cycle. Therefore, the on-the-fly key scheduling for the encryption is expressed as

$$\begin{cases} k_0^{(r+1)} = k_0^{(r)} + KeyEx(k_3^{(r)}) \\ k_1^{(r+1)} = k_0^{(r)} + k_1^{(r)} + KeyEx(k_3^{(r)}) \\ k_2^{(r+1)} = k_0^{(r)} + k_1^{(r)} + k_2^{(r)} + KeyEx(k_3^{(r)}) \\ k_3^{(r+1)} = k_0^{(r)} + k_1^{(r)} + k_2^{(r)} + k_3^{(r)} + KeyEx(k_3^{(r)}) \end{cases}, \tag{7}$$

where $k_0^{(r)}, k_1^{(r)}, k_2^{(r)}$, and $k_3^{(r)}$ are a 32-bit subkey at the r-th round and $KeyEx$ is the key expansion function that consists of a round constant addition, RotWord, and SubWord. The inverse key scheduling for the decryption is represented by

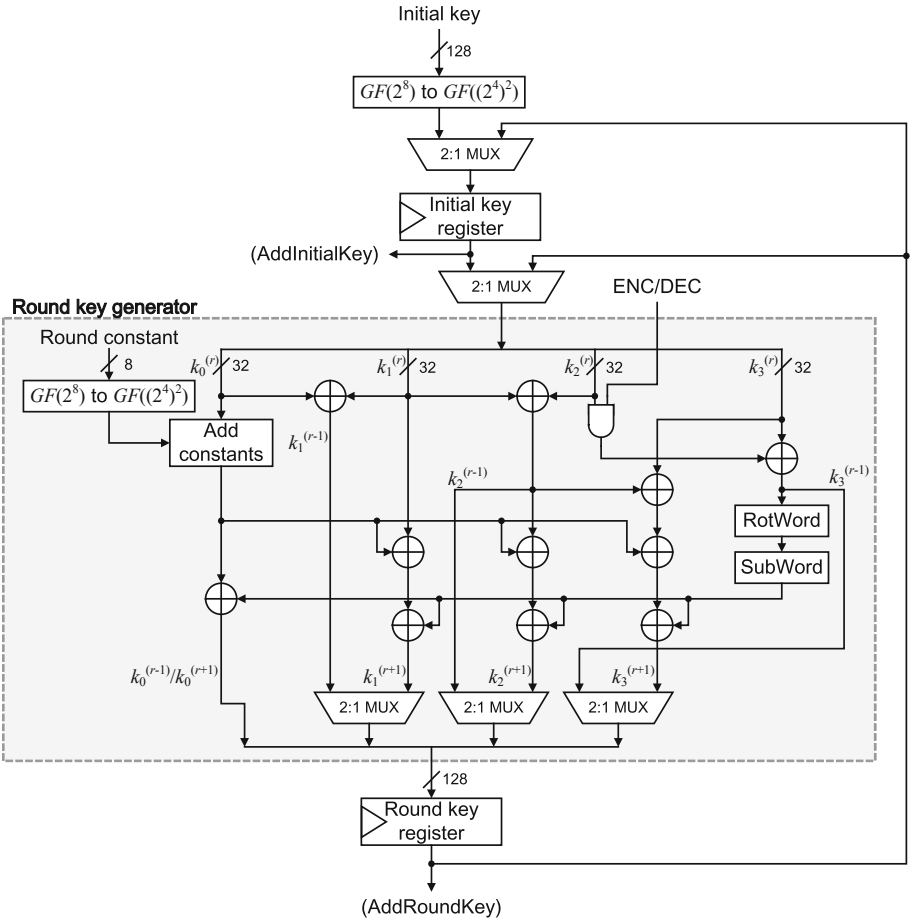

Fig. 8. Proposed key scheduling part.

$$\begin{cases} k_0^{(r-1)} = k_0^{(r)} + KeyEx(k_2^{(r)} + k_3^{(r)}) \\ k_1^{(r-1)} = k_0^{(r)} + k_1^{(r)} \\ k_2^{(r-1)} = k_1^{(r)} + k_2^{(r)} \\ k_3^{(r-1)} = k_2^{(r)} + k_3^{(r)} \end{cases} . \tag{8}$$

Figure 8 shows the proposed key scheduling datapath architecture, where the $KeyEx$ components are unified for encryption and decryption. Note here that most of adders (i.e., XOR gates) for computing $k_1^{(r+1)}$, $k_2^{(r+1)}$, and $k_3^{(r+1)}$ should be non-integrated to make the critical path shorter than that of the round function part. The input key is initially mapped to the tower field, and all of the computations (including AddRoundKey) are performed over the tower field. The ENC/DEC signal controls the input to RotWord and SubWord using a 32-bit AND gate. The upper 2-in-1 multiplexer selects an initial key or a final round key as the input to

Initial key register, the middle 2-in-1 multiplexer selects a key stored in Initial key register or a round key as the input to Round key generator, and the lower 2-in-1 multiplexers select encryption or decryption path. The round constant addition is performed separately from RotWord and SubWord to reduce the critical path delay. As a result, the critical path delay of the key scheduling part becomes shorter than that of the round function part.

4 Performance Evaluation

Tables 1 and 2 summarize the synthesis results of the proposed AES encryption/decryption architecture by Synopsys Design Compiler (Version D2010-3) with the TSMC 65-nm and NanGate 45- and 15-nm standard-cell libraries [2,3] under the worst-case conditions, where Area indicates the circuit area estimated on the basis of a two-way NAND equivalent gate size (i.e., gate equivalents (GEs)); Latency indicates the latency for encryption, which is estimated by the circuit path delay of the datapath under the worst low condition; Max. freq. indicates the maximum operation frequency obtained from the critical path delay; Throughput indicates the throughput at the maximum operation frequency; and Efficiency indicates the throughput per area, which corresponds to the product of the area and latency in this nonpipelined design[3]. To perform a practical performance comparison, an area optimization (which maximizes the effort of minimizing the number of gates without flattening the description) was applied in Table 1, and an area-speed optimization (where an asymptotical search with a set of timing constraints was performed after the area optimization) was applied in Table 2.

In these tables, the conventional representative datapaths [14,15,17,29] were also synthesized using the same optimization conditions. The source codes for these syntheses were described by the authors referring to [14,15,17,29], except for the source codes of Satoh's and Canright's S-boxes in [7,29] that can be obtained from their websites [1,8]. For a fair comparison, the datapaths of [15,17] were adjusted to the round-based nonpipelined architecture corresponding to the proposed datapath. Note that only the inversion circuit over a PB-based $GF((2^4)^2)$ in [17] was not described faithfully according to the paper[4]. Latency and Throughput were calculated assuming that the datapath of [15] requires 10 clock cycles to perform each encryption or decryption and the others require 11

[3] Design Compiler generated a static power consumption report for each architecture. However, the report dose not consider the effect of glitches while tower-field inversion circuits are known to include non-trivial glitches [19]. Therefore, we did not mention the power consumption report to avoid misleading.

[4] According to [17], the $GF(2^4)$ inversion in the circuit can be implemented with a $T_{XOR} + 3T_{NAND}$ delay, where T_{XOR} and T_{NAND} are the delays of the XOR and NAND gates, respectively. However, there is no detailed description to realize such a circuit. Therefore, using the best of our knowledge, we described the circuit by a direct mapping based on the PPRM expansion, which is an algebraic normal form frequently used for designing GF arithmetic circuits [19,28].

Table 1. Synthesis results for proposed and conventional AES hardware architectures with area optimization

	Area (GE)	Latency (ns)	Max. freq. (MHz)	Throughput (Gbps)	Efficiency (Kbps/GE)
TSMC 65-nm					
Satoh et al. [29]	13,671.75	78.10	140.85	1.64	119.88
Lutz et al. [15]	20,380.50	68.50	145.99	1.87	91.69
Liu et al. [14]	12,538.75	85.25	129.03	1.50	119.75
Mathew et al. [17]	20,639.50	97.68	112.61	1.31	63.49
This work	15,242.75	46.97	234.19	2.73	178.78
NanGate 45-nm					
Satoh et al. [29]	12,560.99	31.57	348.43	4.05	322.78
Lutz et al. [15]	20,000.66	20.30	492.61	6.31	315.26
Liu et al. [14]	11,829.34	34.43	319.49	3.72	314.28
Mathew et al. [17]	17,573.33	41.80	263.16	3.06	174.25
This work	13,814.69	16.94	649.35	7.56	546.96
NanGate 15-nm					
Satoh et al. [29]	14,526.01	4.36	2,524.17	29.37	2,022.04
Lutz et al. [15]	23,391.49	4.57	2,185.84	25.44	1,087.37
Liu et al. [14]	13,847.25	4.74	2,321.05	27.01	1,950.46
Mathew et al. [17]	21,361.00	5.32	2,066.93	24.05	1,125.95
This work	15,468.97	2.65	4,144.22	48.22	3,117.44

clock cycles. This is because the initial key addition and first-round computation are performed with one clock cycle for [15]. Area was calculated without the initial key, round key, and data registers to compare the datapaths more clearly. Note also that the key scheduling parts of [15,17] were implemented with the one presented in this paper because there is no description for the key scheduling parts. (For [15], the isomorphic mapping from $GF(2^8)$ to $GF((2^4)^2)$ was removed for applying to the round function part.)

The results in Table 1 show that our datapath achieves the lowest latency (i.e., highest throughput) compared with the conventional ones with tower-field inversion circuits owing to the lower critical path delay. Moreover, the circuit area is not the largest owing to fewer selectors. Note that the latency is consistent with the throughput because these circuits are not pipelined. Although all operations are translated to the tower field in our architecture, the area and delay overheads of MixColumns and InvMixColumns are suppressed by the unification technique. In addition, even with a tower-field S-box, our architecture has an advantage with regard to the latency over Lutz's one with table-lookup-based inversion, as indicated in Table 2. As a result, our architecture is more efficient in terms of the throughput per area than any conventional architecture. More precisely, the proposed datapath is approximately 53–72 % more efficient than any conventional architecture under the conditions of the three CMOS processes. The results also suggest that the proposed architecture would perform an AES encryption or decryption with the smallest energy. Moreover, the cutoff of an inactive path by a demultiplexer would further reduce the power

Table 2. Synthesis results for proposed and conventional AES hardware architectures with area-speed optimization

	Area (GE)	Latency (ns)	Max. freq. (MHz)	Throughput (Gbps)	Efficiency (Kbps/GE)
TSMC 65-nm					
Satoh et al. [29]	14,516.50	56.87	193.42	2.25	155.05
Lutz et al. [15]	22,883.25	33.90	294.99	3.78	165.00
Liu et al. [14]	13,970.50	60.17	182.82	2.13	152.27
Mathew et al. [17]	23,298.49	65.45	168.07	1.96	83.94
This work	15,807.00	34.10	322.58	3.75	237.47
NanGate 45-nm					
Satoh et al. [29]	13,386.67	24.42	450.45	5.24	391.55
Lutz et al. [15]	22,417.01	14.40	694.44	8.89	396.52
Liu et al. [14]	12,443.66	28.27	389.11	4.53	363.86
Mathew et al. [17]	19,243.67	31.90	344.83	4.01	208.51
This work	14,582.99	13.53	813.01	9.46	648.73
NanGate 15-nm					
Satoh et al. [29]	16,924.74	3.31	3,322.26	38.66	2,284.17
Lutz et al. [15]	25,692.49	2.08	4,799.85	61.44	2,391.28
Liu et al. [14]	15,768.43	3.65	3,014.14	35.07	2,224.29
Mathew et al. [17]	23,789.48	4.03	2,729.18	31.76	1,334.95
This work	17,232.00	1.80	6,117.70	71.19	4,131.14

consumption caused by a dynamic hazard, but this could not be evaluated by the logic synthesis and still remains for the future study.

The performance of the architecture in [17] was relatively lower for our experimental conditions because its critical path includes InvMixColumns to compute $MC^{-1}(K_r)$ and therefore becomes longer than those of other designs. In addition, InvMixColumns over a tower-field is more area-consuming than that over an AES field. This suggests that the architecture in [17] is not suitable for an on-the-fly key scheduling implementation. The architectures in [14,29] have smaller areas than the proposed architecture; however, our architecture has a higher throughput. The increasing ratio of the throughput is larger than that of the circuit area because the architectures in [14,29] use InvMixColumns to compute $MC^{-1}(K_r)$ and require several additional selectors, respectively.

The above comparative evaluation was done with the proposed and some conventional but representative datapaths. There are other previous works focusing on efficiency (i.e., throughput per gate) by round-based architectures. However, such previous works do not provide a concrete implementation and/or exhibit better performance than the abovementioned conventional datapaths. For example, a hardware AES implementation with a short critical path was presented in [21], which employed an RRB to reduce the critical path delay of SubBytes/InvSubBytes and MixColumns/InvMixColumns. However, we could not evaluate the efficiency by ourselves because of the lack of a detailed description. Another AES encryption/decryption architecture with a high throughput was presented in [14]. However, the architecture had a lower throughput/area

efficiency compared to the architecture in [29] according to that paper. Moreover, AES architectures that support either encryption or decryption such as in [20,32] are not evaluated in this paper.

5 Discussion

The proposed design employs a round-based architecture without block-wise parallelism such as pipelining. The modes of operations with block-wise parallelism (e.g., the ECB and CTR modes) are also available owing to the trade-off between the area and the throughput by pipelining [11]. A simple way to obtain a pipelined version of the proposed architecture is to unroll the rounds and insert pipeline registers between them. The datapath can be further pipelined by inserting registers into the round datapath. The proposed datapath can be efficiently pipelined by placing the pipeline register at the output of the inversion with a good delay balance between the inversion and the following circuit. For example, the synthesis results for the proposed datapath using the area-speed optimization with the NanGate 45-nm standard-cell library indicated that the inversion circuit had a delay of 0.63 ns, and the remainder had a delay of 0.67 ns. As a result, pipelining would achieve a throughput of 17.37 Gbps, which is nearly twice that without pipelining. Thus, the proposed datapath is also suitable for such a pipelined implementation.

Another discussion point is how the proposed architecture can be resistant to side-channel attacks. A masking countermeasure would be based on a masked tower-field inversion circuit [9,25] such as that in [24]. The major features of the countermeasure are to replace the inversion with a masked inversion and to duplicate other linear operations. Such a countermeasure can also be applied to the proposed datapath. In addition, hiding countermeasures, such as WDDL [30], which replaces the logic gates with a complementary logic style, would also be applicable, and the hardware efficiency would be proportionally lower with respect to the results in Tables 1 and 2.

More sophisticated countermeasures such as threshold implementation (TI) and generalized masking schemes (GMSs) [4,5,18,22,26] would also be applicable to the proposed datapath in principle in the same manner as other conventional ones. On the other hand, such countermeasures, especially against higher-order DPAs, require a considerable area overhead and more random bits compared with the aforementioned countermeasures. When applying such countermeasures, the area overhead would be critical for some applications. In addition, TI- and GMS-based inversion circuits should be pipelined to reduce the resulting circuit area (i.e., the number of shares). To divide the circuit delay equally, it would be better to insert pipeline register at the middle of Encryption and Decryption path in Fig. 7.

6 Conclusion

This paper presented a new efficient round-based AES architecture that supports both encryption and decryption. An efficient AES datapath with a lower latency (or higher throughput per gate) is suitable for some practical modes of operation, such as CBC and CCM, because pipelined parallelism cannot be applied to such modes. The proposed datapath utilizes new operation-reordering and register-retiming techniques to unify critical components (i.e., inversion and linear matrix operations) with fewer additional selectors. As a result, our datapath has the lowest critical path delay compared to conventional ones with tower-field S-boxes. The proposed and conventional AES hardware were designed on the basis of compatible round-based architectures and evaluated using logic synthesis with TSMC 65-nm and NanGate 45- and 15-nm CMOS standard-cell libraries under the worst-case conditions. The synthesis results suggested that the proposed architecture was approximately 53–72 % more efficient than the best conventional architecture in terms of the throughput per area, which would also indicate that the proposed architecture can perform encryption/decryption with the lowest energy.

The performance evaluation was performed at the design stage of the logic synthesis; therefore, the power consumption and latency considering place and route were not evaluated. A detailed evaluation after the place and route is planned as future work. However, the post-synthesis results would be proportional to the presented synthesis results because the proposed and conventional architectures employ the same or similar hardware algorithms (e.g., tower-field inversion) and do not have any extra global wires that have an impact on the critical path. The design of efficient and side-channel-resistant AES hardware based on the proposed datapath is also planned for future work.

Acknowledgment. This work has been supported by JSPS KAKENHI Grant No. 25240006.

Appendix: An Example Set of Linear Mappings and a Unified Affine

This appendix provides an example set of matrices for linear operations, i.e., an isomorphic mapping, an inverse isomorphic mapping, an affine transformation over the tower field, inverse affine transformation over the tower field, $U_0, U_1, U_2, U_3, V_0, V_1, V_2$, and V_3. In this study, we employ the tower-field inversion circuit in [31]. In the following formulae, the least-significant bits are in the upper-left corner.

The conversion matrices of the isomorphic mapping and its inverse mapping (denoted by δ and δ', respectively) are given by

$$\delta = \begin{pmatrix} 0\,1\,0\,1\,1\,1\,0\,0 \\ 1\,0\,1\,0\,0\,0\,1\,1 \\ 1\,0\,0\,1\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0\,1\,0\,0 \\ 0\,1\,1\,0\,1\,1\,0\,0 \\ 1\,0\,1\,0\,1\,0\,0\,0 \\ 1\,1\,1\,0\,0\,0\,0\,1 \\ 0\,0\,1\,1\,0\,0\,0\,1 \end{pmatrix}, \quad \delta' = \begin{pmatrix} 1\,1\,0\,1\,1\,0\,0\,1\,1\,0 \\ 0\,1\,0\,1\,0\,0\,1\,0\,1\,0 \\ 0\,1\,0\,0\,1\,1\,0\,1\,1\,1 \\ 1\,0\,0\,0\,1\,0\,1\,1\,1\,1 \\ 1\,0\,0\,1\,0\,0\,0\,1\,0\,1 \\ 1\,0\,0\,0\,1\,0\,0\,0\,0\,0 \\ 1\,1\,1\,1\,0\,1\,1\,0\,0\,0 \\ 1\,1\,0\,0\,0\,0\,1\,0\,0\,1 \end{pmatrix}. \tag{9}$$

The isomorphic mapping using δ performs conversion from the AES field to the tower field used in [31] (i.e., an NB-based $GF((2^4)^2)$). The inverse isomorphic mapping using δ' performs conversion from the RRB-based $GF((2^4)^2)$ to the AES field. The affine and inverse affine matrices over the tower field (denoted by ϕ and ϕ', respectively) are given by

$$\phi = \begin{pmatrix} 1\,1\,1\,0\,1\,0\,0\,1\,1\,0 \\ 1\,0\,0\,0\,1\,0\,0\,1\,1\,0 \\ 1\,1\,0\,1\,1\,1\,0\,1\,0\,0 \\ 1\,0\,0\,0\,1\,1\,0\,1\,1\,1 \\ 1\,0\,0\,1\,0\,1\,0\,0\,0\,1 \\ 1\,1\,0\,1\,1\,0\,1\,0\,0\,1 \\ 1\,0\,0\,1\,0\,1\,1\,1\,1\,0 \\ 1\,1\,0\,1\,1\,0\,1\,1\,0\,0 \end{pmatrix}, \quad \phi' = \begin{pmatrix} 0\,0\,0\,1\,0\,1\,1\,0 \\ 1\,1\,0\,1\,0\,1\,1\,0 \\ 0\,1\,0\,1\,1\,0\,0\,0 \\ 0\,0\,1\,1\,1\,0\,1\,1 \\ 0\,0\,1\,0\,0\,0\,0\,1 \\ 0\,1\,0\,1\,0\,1\,0\,1 \\ 0\,0\,1\,0\,1\,1\,1\,0 \\ 0\,1\,0\,1\,0\,0\,0\,0 \end{pmatrix}. \tag{10}$$

The input and output of the linear mapping represented by ϕ are given by the RRB- and NB-based $GF((2^4)^2)$, respectively. The input and output of the linear mapping represented by ϕ' are given by the NB-based $GF((2^4)^2)$. The constants $\Delta(c)$ and $\Delta(c')$ are given by $\beta^5 + \beta^3 + \beta^2$ and $\beta^7 + \beta^4 + \beta^2$, respectively. Let ψ_e and ψ'_e be the matrices representing U_e and V_e, respectively ($0 \leq e \leq 3$). The matrices ψ_0, ψ_1, ψ_2, and ψ_3 are given by

$$\psi_0 = \begin{pmatrix} 1\,1\,1\,1\,0\,0\,1\,1\,1\,1 \\ 0\,0\,1\,1\,0\,1\,0\,1\,0\,0 \\ 1\,1\,0\,1\,1\,0\,1\,1\,1\,1 \\ 1\,1\,0\,1\,1\,1\,0\,0\,0\,1 \\ 1\,0\,0\,1\,0\,0\,0\,0\,1\,1 \\ 1\,0\,1\,1\,1\,0\,0\,0\,0\,0 \\ 1\,1\,1\,0\,1\,0\,1\,0\,1\,0 \\ 0\,1\,0\,0\,1\,0\,1\,0\,0\,1 \end{pmatrix}, \quad \psi_1 = \begin{pmatrix} 0\,0\,0\,1\,1\,0\,1\,0\,0\,1 \\ 1\,0\,1\,1\,1\,1\,1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1\,1\,0\,1\,1 \\ 0\,1\,0\,1\,0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,1\,0 \\ 0\,1\,1\,0\,0\,0\,1\,0\,0\,1 \\ 0\,1\,1\,1\,1\,1\,0\,1\,0\,0 \\ 1\,0\,0\,1\,0\,0\,0\,1\,0\,1 \end{pmatrix}, \tag{11}$$

$$\psi_2 = \psi_3 = \phi. \tag{12}$$

respectively. The matrices $\psi'_0, \psi'_1, \psi'_2$, and ψ'_3 are given by

$$\psi'_0 = \begin{pmatrix} 0&0&0&0&0&0&1&1&0&0 \\ 0&0&1&0&1&0&0&1&0&1 \\ 0&1&0&0&1&1&1&0&1&1 \\ 1&0&0&0&1&1&1&0&1&1 \\ 1&1&0&0&0&0&0&1&0&1 \\ 0&0&1&0&1&0&0&0&1&1 \\ 1&1&0&1&1&0&0&0&1&1 \\ 1&1&0&1&1&1&1&1&1&0 \end{pmatrix}, \quad \psi'_1 = \begin{pmatrix} 0&0&0&0&0&1&1&0&1&1 \\ 0&0&0&1&1&0&0&0&1&1 \\ 1&1&0&0&0&0&1&0&0&1 \\ 0&1&1&1&1&0&1&0&0&1 \\ 1&0&1&1&1&0&0&0&1&1 \\ 0&0&0&1&1&1&1&1&1&0 \\ 0&1&0&0&1&1&1&1&1&0 \\ 0&1&0&0&1&0&1&0&1&0 \end{pmatrix}, \tag{13}$$

$$\psi'_2 = \begin{pmatrix} 1&0&1&1&1&1&1&1&1&0 \\ 0&1&0&1&0&1&1&1&1&0 \\ 1&0&0&0&1&1&1&0&1&1 \\ 0&1&1&1&1&1&0&1&0&0 \\ 1&1&0&0&0&1&0&1&1&1 \\ 1&0&0&0&1&1&0&0&0&1 \\ 1&1&0&0&0&0&0&1&0&1 \\ 1&0&0&0&1&0&0&0&0&0 \end{pmatrix}, \quad \psi'_3 = \begin{pmatrix} 0&0&1&1&0&0&1&1&1&1 \\ 1&0&0&0&1&1&1&1&1&0 \\ 0&0&1&0&1&1&0&0&0&1 \\ 1&0&0&1&0&1&1&1&0&1 \\ 0&0&1&0&1&0&0&0&0&0 \\ 1&0&0&1&0&0&1&0&0&1 \\ 1&1&0&0&0&0&0&1&1&0 \\ 0&0&1&1&0&1&0&1&0&0 \end{pmatrix}. \tag{14}$$

References

1. Cryptographic hardware project. http://www.aoki.ecei.tohoku.ac.jp/crypto/
2. NanGate FreePDK15 open cell library, January 2016. http://www.nangate.com/?page_id=2328
3. NanGate FreePDK45 open cell library, January 2016. http://www.nangate.com/?page_id=2325
4. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: Higher-order threshold implementations. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 326–343. Springer, Heidelberg (2014)
5. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: Trade-offs for threshold implementations illustrated on AES. IEEE Trans. Comput. Aided Des. Integr. Syst. **34**(7), 1188–1200 (2015)
6. Boyer, J., Matthews, P., Peralta, P.: Logic minimization techniques with applications to cryptology. J. Cryptology **47**, 280–312 (2013)
7. Canright, D.: A very compact S-box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005)
8. Canright, D.: http://faculty.nps.edu/drcanrig/
9. Canright, D., Batina, L.: A very compact "Perfectly Masked" S-Box for AES. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 446–459. Springer, Heidelberg (2008)
10. Hammad, I., El-Sankary, K., El-Masry, E.: High-speed AES encryptor with efficient merging techniques. IEEE Embed. Syst. Lett. **2**, 67–71 (2010)
11. Hodjat, A., Verbauwhede, I.: Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors. IEEE Trans. Comput. **50**(4), 366–372 (2006)
12. Jeon, Y., Kim, Y., Lee, D.: A compact memory-free architecture for the AES algorithm using resource sharing methods. J. Circ. Syst. Comput. **19**(5), 1109–1130 (2010)

13. Lin, S.Y., Huang, C.T.: A high-throughput low-power AES cipher for network applications. In: The 12th Asia and South Pacific Design Automation Conference (ASP-DAC 2007), pp. 595–600. IEEE (2007)
14. Liu, P.C., Chang, H.C., Lee, C.Y.: A 1.69 Gb/s area-efficient AES crypto core with compact on-the-fly key expansion unit. In: 41st European Solid-State Circuits Conference (ESSCIRC 2009), pp. 404–407. IEEE (2009)
15. Lutz, A., Treichler, J., Gürkaynak, F., Kaeslin, H., Basler, G., Erni, A., Reichmuth, S., Rommens, P., Oetiker, P., Fichtner, W.: 2Gbit/s hardware realizations of RIJNDAEL and SERPENT: a comparative analysis. In: Kaliski, B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 144–158. Springer, Heidelberg (2002)
16. Mathew, S., Satpathy, S., Suresh, V., Anders, M., Himanshu, K., Amit, A., Hsu, S., Chen, G., Krishnamurthy, R.K.: 340 mV-1.1V, 289 Gbps/W, 2090-gate nanoAES hardware accelerator with area-optimized encrypt/decrypt $GF(2^4)^2$ polynomials in 22 nm tri-gate CMOS. IEEE J. Solid-State Circ. **50**, 1048–1058 (2015)
17. Mathew, S.K., Sheikh, F., Kounavis, M.E., Gueron, S., Agarwal, A., Hsu, S.K., Himanshu, K., Anders, M.A., Krishnamurthy, R.K.: 53 Gbps native $GF(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors. IEEE J. Solid-State Circ. **46**, 767–776 (2011)
18. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: a very compact and a threshold implementation of AES. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011)
19. Morioka, S., Satoh, A.: An optimized S-Box circuit architecture for low power AES design. In: Kaliski, B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 172–186. Springer, Heidelberg (2002)
20. Morioka, S., Satoh, A.: A 10 Gbps full-AES crypto design with a twisted-BDD S-box architecture. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **12**, 686–691 (2004)
21. Nekado, K., Nogami, Y., Iokibe, K.: Very short critical path implementation of AES with direct logic gates. In: Hanaoka, G., Yamauchi, T. (eds.) IWSEC 2012. LNCS, vol. 7631, pp. 51–68. Springer, Heidelberg (2012)
22. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glithces. J. Cryptology **24**, 292–321 (2011)
23. Nogami, Y., Nekado, K., Toyota, T., Hongo, N., Morikawa, Y.: Mixed bases for efficient inversion in $\mathbb{F}_{((2^2)^2)^2}$ and conversion matrices of SubBytes of AES. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 234–247. Springer, Heidelberg (2010)
24. Okamoto, K., Homma, N., Aoki, T., Morioka, S.: A hierarchical formal approach to verifying side-channel resistant cryptographic processors. In: Hardware-Oriented Security and Trust (HOST), pp. 76–79. IEEE (2014)
25. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A side-channel analysis resistant description of the AES S-Box. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 413–423. Springer, Heidelberg (2005)
26. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 764–783. Springer, Heidelberg (2015)
27. Rudra, A., Dubey, P.K., Jutla, C.S., Kumar, V., Rao, J.R., Rohatgi, P.: Efficient Rijndael encryption implementation with composite field arithmetic. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 171–184. Springer, Heidelberg (2001)

28. Sasao, T.: AND-EXOR expressions and their optimization. In: Sasao, T. (ed.) Logic Synthesis and Optimization. The Kluwer International Series in Engineering and Computer Science, vol. 212, pp. 287–312. Kluwer Academic Publishers (1993)
29. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A compact Rijndael hardware architecture with S-Box optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
30. Tiri, K., Verbauwhede, I.: A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In: Design, Automation and Test in Europe Conference and Exhibition (DATE), vol. 1, pp. 246–251 (2004)
31. Ueno, R., Homma, N., Sugawara, Y., Nogami, Y., Aoki, T.: Highly efficient $GF(2^8)$ inversion circuit based on redundant GF arithmetic and its application to AES design. In: Güneysu, T., Handschuh, H. (eds.) CHES 2015. LNCS, vol. 9293, pp. 63–80. Springer, Heidelberg (2015)
32. Verbauwhede, I., Schaumont, P., Kuo, H.: Design and performance testing of a 2.29-GB/s Rijndael processor. IEEE J. Solid-State Circ. **38**, 569–572 (2003)