Journal of Cryptographic Engineering (2019) 9:101–113
https://doi.org/10.1007/s13389-018-0187-8

REGULAR PAPER

# Highly efficient $GF(2^8)$ inversion circuit based on hybrid GF representations

**Rei Ueno[1] · Naofumi Homma[1] · Yasuyuki Nogami[2] · Takafumi Aoki[1]**

Received: 6 June 2017 / Accepted: 10 March 2018 / Published online: 20 March 2018


**Abstract**
This paper proposes a compact and highly efficient $GF(2^8)$ inversion circuit design based on a combination of non-redundant and redundant Galois field (GF) (or finite field) arithmetic. The proposed design utilizes an optimal normal basis and redundant GF representations, called polynomial ring representation and redundantly represented basis, to implement $GF(2^8)$ inversion using a tower field $GF((2^4)^2)$. The flexibility of the redundant representations provides efficient mappings from/to the $GF(2^8)$. This paper evaluates the efficacy of the proposed circuit by gate counts and logic synthesis with a 65-nm CMOS standard cell library in comparison with conventional circuits. Consequently, we show that the proposed circuit achieves approximately 25% higher area–time efficiency than the conventional best inversion circuit in our environment. We also demonstrate that AES S-Box with the proposed circuit achieves the best area–time efficiency.

**Keywords** Hardware implementation · $GF(2^8)$ inversion circuit · S-Box · AES

## 1 Introduction

Cryptography based on Galois field (GF) (or finite field) arithmetic has been widely utilized for secure communications, authentication, and digital signatures in many systems. For modern ciphers, the substitution function is one of the most integral parts to be resistant against major cryptanalytic techniques such as differential and linear cryptanalyses [1,2]. Inversion functions over $GF(2^m)$ are known as a useful component for $m$-bit substitution functions [3]. Many ISO/IEC standard ciphers (e.g., AES and Camellia) employ an inversion function over $GF(2^8)$ in substitution functions [4,5]. For example, SubBytes of AES consists of an inversion over $GF(2^8)$ (i.e., S-Box) and an affine transformation over $GF(2)$. The hardware performance of such ciphers heavily depends on the inversion circuits used. As a result of the explosive increase in resource-constrained devices in the context of Internet of things (IoT) applications, there is currently substantial demand for lightweight implementation of inversion functions [3,6].

So far, many approaches to reducing the hardware cost of $GF(2^8)$ inversion circuits have been proposed. While it has been shown that direct mapping-based approaches (e.g., table lookup, PPRM, and BDD [7–9]) are useful for low-latency implementation, the tower field approach, which calculates $a^{-1}$ $(= a^{254})$ $(a \in GF(2^8))$ using the equivalent tower field, is a promising approach for achieving the compact and efficient implementation. This technique converts the original field $GF(2^8)$ into an isomorphic tower field such as $GF(((2^2)^2)^2)$ and $GF((2^4)^2)$ in the middle of the inversion. Researchers have previously shown that the tower field approach is efficient because the subfields $GF((2^2)^2)$ and $GF(2^4)$ operations are designed more compactly than the original field operations. Satoh et al. [10] were the first to present a compact implementation of the AES S-Box by the tower field $GF(((2^2)^2)^2)$ represented by polynomial bases (PB). Canright [11] further reduced the gate count of the AES S-Box by using normal bases (NB) and optimizing the change-of-basis. Canright's implementation was the smallest for a long time. Nogami et al. [12] recently mixed polynomial and normal bases (MB) to achieve the most efficient implementation. They showed that the product of gate count and critical delay for the inversion circuit could be reduced by the MB. Some implementations using $GF((2^4)^2)$ have

✉ Rei Ueno
ueno@riec.tohoku.ac.jp

[1] Tohoku University, 6-6-05, Aramaki Aza Aoba, Aoba-ku, Sendai-shi 980-8579, Japan

[2] Okayama University, Tsushima-naka, Kita-ward, Okayama-shi 700-8530, Japan

also been proposed by researchers such as Jeon et al. [13], who presented PB-based $GF((2^4)^2)$ inversion circuit design. These results suggest that such field representations have a significant impact on hardware performance.

The above bases (i.e., PB, NB, and MB) represent each element of $GF(2^m)$ using $m$ bits in a non-redundant manner. However, there are two redundant representations, namely polynomial ring representation (PRR) and redundantly represented basis (RRB), which use $n$ $(> m)$ bits to represent each element of $GF(2^m)$. The defining polynomial of these redundant representations is given by a reducible polynomial of degree $n$, whereas that of non-redundant representations is given by an irreducible polynomial of degree $m$. This means that redundant representations provide a wider variety of polynomials that can be selected as a defining polynomial than non-redundant representations. Drolet [14] showed that the use of PRR makes it possible to select a binomial $x^n+1$ as a defining polynomial, which can lead to the design of small-complexity arithmetic circuits. Wu et al. [15] and Nekado et al. [16] showed that RRB-based designs were useful for designing efficient inversion circuits.

This paper presents a technique in which non-redundant and redundant GF arithmetic are combined to achieve a compact and efficient $GF(2^8)$ inversion circuit design. The key idea underlying the proposed circuit is calculation of the inversion of the tower field $GF((2^4)^2)$ by the NB, PRR, and RRB combination. The former part for the 16th and 17th powers of the input is calculated by an NB with a symmetric property. This is followed by calculation of the latter parts for $GF(2^4)$ inversion and $GF(2^4)$ multiplication by PRR and RRB, respectively. The mapping from NB to PRR/RRB is efficiently implemented by the symmetric property of the NB. The efficacy of the proposed circuit is evaluated by means of gate counts and logic synthesis results using a TSMC 65-nm CMOS standard cell library. The proposed circuit has approximately 25% higher efficiency (i.e., area–time product) excluding the change-of-basis than any other conventional circuits, including those with the tower field $GF(((2^2)^2)^2)$. In addition, the flexibility of redundant representations in the proposed circuit enables it to have the best efficiency even including the change-of-basis from/to $GF(2^8)$. To the best of our knowledge, the proposed circuit is the most efficient tower field arithmetic-based implementation for the AES S-Box.

While a preliminary version [17] presented and evaluated the above circuit based on the combination of non-redundant and redundant GF arithmetic, this paper further enhances our circuit by two new techniques. First, we present a variation of the new conceptual circuit with an additional unification of inner operations, which achieves the shortest critical path among tower field inversion circuits including that in [17]. The unification also results in a more area–time efficient inversion/S-Box circuit than ever before. We then present a more efficient unified S-Box design which supports both encryption and decryption by a new optimization technique for change-of-basis. The new designs are evaluated by the same manner as the preliminary version [17]. As a result, we confirm that our new designs have the highest efficiency in terms of area–time product even where both encryption and decryption are supported.

The remainder of this paper is organized as follows. Section 2 introduces preliminary and related work associated with the design of $GF(2^8)$ inversion circuits. The redundant GF representations introduced in the proposed circuit are also described. Section 3 presents the proposed $GF(2^8)$ inversion circuit. In addition, this section evaluates the proposed circuit by the results of gate count and logic synthesis. Section 4 presents the AES S-Box design that incorporates the proposed inversion circuit and its change-of-basis, and shows their evaluation results in the same manner as Sect. 3. Finally, Sect. 5 presents concluding remarks.

## 2 Preliminaries and related works

### 2.1 Inversion circuits by tower fields

This section briefly describes previous work on the design of $GF(2^8)$ inversion circuits based on tower field arithmetic. The inverse element of $a \in GF(2^8)$ is given by

$$a^{254} = \begin{cases} a^{-1} & \text{for } a \neq 0 \\ 0 & \text{if } a = 0 \end{cases},$$

because any element of $GF(2^8)$ satisfies $a = a^{256}$. The basic idea underlying the tower field approach is reduction in hardware cost by exploiting smaller arithmetic operations over subfield $GF((2^2)^2)$ or $GF(2^4)$ instead of $GF(2^8)$. There is a one-to-one mapping (i.e., change-of-basis) between the elements of $GF(2^8)$ and those of the tower field. This GF inversion over a tower field is efficiently implemented in the Itoh–Tsujii Algorithm (ITA) [18].

Figure 1 illustrates a $GF(2^8)$ inversion circuit presented in [11], where the data path is divided into upper and lower 4 bits and each component denotes an arithmetic circuit over subfield $GF((2^2)^2)$. Let $a \in GF(((2^2)^2)^2)$ be the input given by $h\alpha^{16} + l\alpha$ in an NB $\{\alpha^{16}, \alpha\}$, where $h$ and $l$ $(\in GF((2^2)^2))$ are, respectively, the upper and lower 4 bits of $a$, and $\alpha$ is a root of an irreducible polynomial of degree 2 over $GF((2^2)^2)$ (i.e., a defining polynomial for extending $GF((2^2)^2)$ to $GF(((2^2)^2)^2)$). The inversion of $a$ is calculated in the following three stages: (1) calculation of the 16th and 17th powers, (2) subfield inversion, and (3) final multiplication. Note that the above $GF((2^2)^2)$ operators are replaced with the $GF(2^4)$ operators in the case of the tower field $GF((2^4)^2)$.

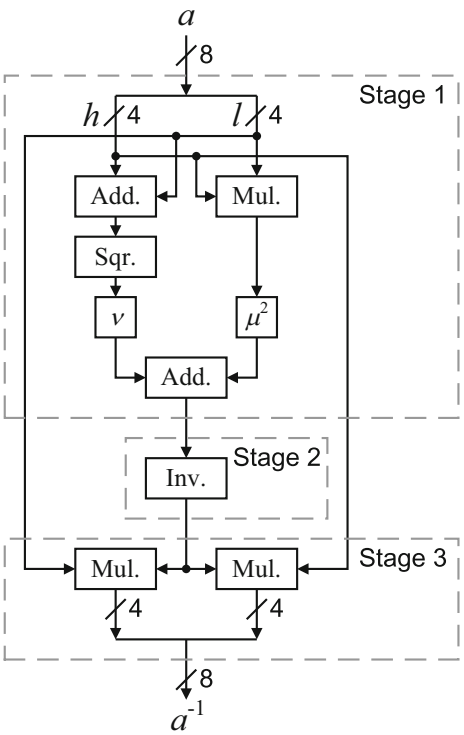


**Fig. 1** Inversion circuit over $GF(((2^2)^2)^2)$ in [11]

The performance of this inversion circuit depends on the tower field and its basis representation. Three of the best known circuit structures are based on the tower field of $GF(((2^2)^2)^2)$. Satoh et al. first designed this kind of $GF(((2^2)^2)^2)$ inversion circuit using PB [10]. Canright then designed a more compact circuit based on NB [11]. The hardware cost of inversion and exponentiation operations can be reduced by NB because the squaring operation is performed solely by wiring. Nogami et al. presented the possibility of MB, which employs both polynomial and normal bases for the input and output data, respectively [12]. Their method exhibited improved performance in the product of gate count and critical delay for the $GF(((2^2)^2)^2)$ inversion circuit and the AES S-Box, including change-of-basis. In addition to $GF(((2^2)^2)^2)$, it is possible to design efficient inversion circuits using another tower field of $GF((2^4)^2)$. Jeon et al. [13] designed $GF((2^4)^2)$ inversion circuit based on PB with smaller critical delay than those of $GF(((2^2)^2)^2)$ inversion circuits.

## 2.2 Redundant representations for Galois fields

*Polynomial ring representation (PRR)* is a redundant representation of GF [14]. An extension field $GF(2^m)$ based on a PB has a set of elements (i.e., polynomials) whose degrees are at most $m-1$ (i.e., $m$ bits). Elements of an NB-based $GF(2^m)$ are also represented by $m$ bits. On the other hand, an extension field $GF(2^m)$ based on PRR has a set of polynomials whose degrees are up to $n-1$ (i.e., $n$ bits), where $n > m$ [14]. In other words, whereas a PB- or an NB-based $GF(2^m)$ is defined as an $m$-dimensional linear space over $GF(2)$, a PRR-based $GF(2^m)$ is defined as an $m$-dimensional subspace of an $n$-dimensional linear space. PRR is also equivalent to cyclic redundancy code (CRC), a kind of error-correction code.

Let $x$ and $H(x)$ be an indeterminate element and an irreducible polynomial of degree $m$ over $GF(2)$, respectively. Let $G(x)$ be a polynomial of degree $n-m$, which is relatively prime to $H(x)$, and is satisfied with $G(0) \neq 0$. Let $P(x)$ be a polynomial (of degree $n$) given by the product of $G(x)$ and $H(x)$. A set of polynomials of degrees less than or equal to $n-1$, where each polynomial is divisible by $G(x)$, together with modulo $P(x)$ arithmetic is isomorphic to $GF(2^m)$. Note here that $n = m + \deg G(x)$. The representation of $GF(2^m)$ using such a residue ring is called PRR. A PRR can be constructed from any PB-based $GF(2^m)$.

As an example of PRR, we present the construction of a PRR-based $GF(2^4)$, where the defining polynomial $P(x)$ is given by $x^5+1$ and its factors $G(x)$ and $H(x)$ are given by

$$G(x) = x + 1,$$
$$H(x) = x^4 + x^3 + x^2 + x + 1.$$

We first compute the multiplicative unit element $E(x)$, which is given by

$$E(x) = U(x)G(x) = 1 - V(x)H(x),$$

where $U(x)$ and $V(x)$ are polynomials satisfying $U(x)G(x) + V(x)H(x) = 1$. Hence, $E(x) = (x^3+x)(x+1) = x^4 + x^3 + x^2 + x$. Let $C_i(x) = (x^i \bmod H(x)) \times E(x) \bmod P(x)$ $(0 \leq i \leq 2^4 - 2)$. Here, $C_i(x)$ corresponds to $\beta^i$, where $\beta$ is a root of $H(x)$. In addition, zero is mapped to zero. Thus, we can derive a correspondence between a PB-based $GF(2^4)$ with the irreducible polynomial $H(x)$ and the PRR-based $GF(2^4)$. Table 1 shows the correspondence between the PB- and PRR-based $GF(2^4)$, which shows an example of one-to-one mapping between PB- and PRR-based elements. Here, the PB-based $GF(2^4)$ represents elements by polynomials of degrees at most 3 (i.e., 4 bits), whereas the PRR-based $GF(2^4)$ represents elements by polynomials of degrees up to 4 (i.e., 5 bits). In addition, the PRR-based $GF(2^4)$ consists of only polynomials dividable by $G(x) = x + 1$, which means that the PRR-based $GF(2^4)$ is equivalent to a cyclic code with the generator polynomial $G(x)$. See [20] for details about the construction method.

It is known that the performance of the GF circuit generally improves as the number of terms in the modular polynomial decreases [19]. Here, a binomial $x^n + 1$ is available for the modular polynomial of PRR-based $GF(2^m)$, whereas it is unavailable for GFs based on non-redundant representations.

**Table 1** Example of correspondence between PB- and PRR-based $GF(2^4)$

| PB where $\beta^4+\beta^3+\beta^2+\beta+1=0$ | | PRR where $P(x)=x^5+1$ | |
|---|---|---|---|
| Polynomial | Vector repr. | Polynomial | Vector repr. |
| 0 | 0000 | 0 | 00000 |
| 1 | 0001 | $x^4+x^3+x^2+x$ | 11110 |
| $\beta^2+\beta$ | 0110 | $x^2+x$ | 00110 |
| $\beta^3+\beta+1$ | 1011 | $x^4+x^2$ | 10100 |
| $\beta^2$ | 0100 | $x^4+x^3+x+1$ | 11011 |
| $\beta^2+\beta+1$ | 0111 | $x^4+x^3$ | 11000 |
| $\beta^3+\beta^2+1$ | 1101 | $x^4+x$ | 10010 |
| $\beta^3+\beta^2+\beta+1$ | 1111 | $x^3+x^2+x+1$ | 01111 |
| $\beta+1$ | 0011 | $x+1$ | 00011 |
| $\beta^3+\beta$ | 1010 | $x^3+x$ | 01010 |
| $\beta$ | 0010 | $x^4+x^3+x^2+1$ | 11101 |
| $\beta^3+\beta^2$ | 1100 | $x^3+x^2$ | 01100 |
| $\beta^3+1$ | 1001 | $x^3+1$ | 01001 |
| $\beta^3$ | 1000 | $x^4+x^2+x+1$ | 10111 |
| $\beta^3+\beta^2+\beta$ | 1110 | $x^4+1$ | 10001 |
| $\beta^2+1$ | 0101 | $x^2+1$ | 00101 |

This is because the modular polynomial $P(x)$ is given by a reducible polynomial (i.e., $G(x)\times H(x)$). Thus, the performance of PRR-based GF arithmetic circuits can be better than those of PB- and NB-based arithmetic circuits. For example, we can use $x^{m+1}+1$ for $P(x)$ if the $m$th degree all one polynomial (AOP) is irreducible according to the following formula over $GF(2)$:

$$x^{m+1}+1=(x+1)(x^m+x^{m-1}+\cdots+1),$$

where the polynomial $x^m+x^{m-1}+\cdots+1$ is called the AOP of degree $m$.

The major advantages of using the binomial are as follows: (i) Parallel multiplication can be given as the discrete time Wiener–Hopf equation and (ii) squaring and a part of constant multiplication are performed only by bit-wise permutation (i.e., wiring). In (i), reduction by $P(x)$ in multiplication can be performed only by bit-wise permutation while multiplication based on non-redundant representation requires some addition over $GF(2)$ for the reduction. In (ii), squaring and a part of constant multiplication can be performed without any logic gate since they are the special case of multiplication. Accordingly, the PRR-based design can be more efficient than conventional designs.

*Redundantly represented basis (RRB)* is another redundant representation of GF [16]. Each element is represented by a primitive $n$th root where $n$ is the minimal integer such that $GF(2^m)$ can be embedded into $GF(2^n)$.

RRB is available when the AOP of degree $m$ is irreducible. Let $\beta$ be a root of the AOP. The $m$ elements (i.e., bases) $\beta^{m-1}, \beta^{m-2}, \ldots,$ and $\beta^0$ are linearly independent and then compose a PB. In contrast, RRB employs a binomial $\beta^{m+1}-1$ as the defining polynomial, which is satisfied with the following equation:

$$\beta^{m+1}+1=(\beta+1)(\beta^m+\beta^{m-1}+\cdots+1)=0.$$

The set $\{\beta^m, \beta^{m-1}, \ldots, \beta^0\}$ is called RRB. Because the degree of the binomial is $m+1$, each element is represented by a linear combination of $\beta^m, \beta^{m-1}, \ldots,$ and $\beta^0$. Note that the elements of such an RRB-based $GF(2^m)$ are represented in a non-unique manner because $\beta^m, \beta^{m-1}, \ldots,$ and $\beta^0$ are linearly dependent in contrast to PRR[1].

RRB-based $GF(2^m)$ squaring can be performed by bit-wise permutation, as is the case with NB. This is because RRB is equivalent to an extended (Type I) optimal normal basis (ONB). We can derive RRB by adding a base $\{\beta^0\}$ to ONB. This means that an efficient multiplication method for ONB, called the cyclic vector multiplication algorithm (CVMA) [21], is also available for RRB. Thus, we can design more compact and efficient multipliers by combining RRB and CVMA. As an example, let us consider a $GF(2^4)$ multiplier based on RRB. Let $s$ and $t$ ($\in GF(2^4)$) be the inputs and $u$ ($\in GF(2^4)$) be the output. Let $\beta$ be a root of the AOP of degree 4. In RRB, $s$ is given by $s_4\beta^4+s_3\beta^3+\cdots+s_0$,

[1] While PRR and RRB can use the same defining polynomial, the major difference between PRR and RRB is the uniqueness of representation for each element. For example, $GF(2^2)$ can be represented redundantly with a modular polynomial $x^3+1$. Here, a PRR-based $GF(2^2)$ consists of four elements: $0, x+1, x^2+1,$ and $x^2+x$ (which is equivalent to a cyclic code with a generator polynomial $x+1$) while an RRB-based $GF(2^2)$ consists of eight elements: $0, 1, \gamma, \gamma+1, \gamma^2, \gamma^2+1, \gamma^2+\gamma,$ and $\gamma^2+\gamma+1$ where $\gamma$ denotes a root of $x^2+x+1$.

where $s_0, s_1, \ldots,$ and $s_4 \in GF(2)$. Also, $t$ and $u$ are given in the same manner. The multiplication is represented by

$$u = s \times t = u_4\beta^4 + u_3\beta^3 + u_2\beta^2 + u_1\beta + u_0,$$

where

$$u_0 = (s_1 + s_4)(t_1 + t_4) + (s_2 + s_3)(t_2 + t_3), \quad (1)$$
$$u_1 = (s_0 + s_1)(t_0 + t_1) + (s_2 + s_4)(t_2 + t_4), \quad (2)$$
$$u_2 = (s_0 + s_2)(t_0 + t_2) + (s_3 + s_4)(t_3 + t_4), \quad (3)$$
$$u_3 = (s_0 + s_3)(t_0 + t_3) + (s_1 + s_2)(t_1 + t_2), \quad (4)$$
$$u_4 = (s_0 + s_4)(t_0 + t_4) + (s_1 + s_3)(t_1 + t_3). \quad (5)$$

The critical delay of the RRB-based $GF(2^4)$ multiplier is $T_A + 2T_X$, while those of multipliers based on non-redundant representations are $T_A + 3T_X$ [16]. The gate count of the RRB-based multiplier requires only 10 AND and 25 XOR gates [16], whereas that of a PRR-based multiplier requires 25 AND and 20 XOR gates [14].

Nekado et al. [16] designed a more efficient $GF((2^4)^2)$ inversion circuit based on RRB by utilizing the above advantage. Figure 2 shows the block diagram of $GF((2^4)^2)$ inversion circuit in [16], where $GF(2^4)$ is given by RRB and $GF((2^4)^2)$ (i.e., the quadratic extension of $GF(2^4)$) is given by NB. In the circuit, RRB-based multiplications are divided into two parts denoted by Mul0 and Mul1 in order to reduce circuit area. Mul0 performs bit-wise XOR operations, and Mul1 performs bit-wise AND operations followed by XOR operations. This circuit achieved lower latency than the previous ones thanks to the efficient multiplications based on RRB.

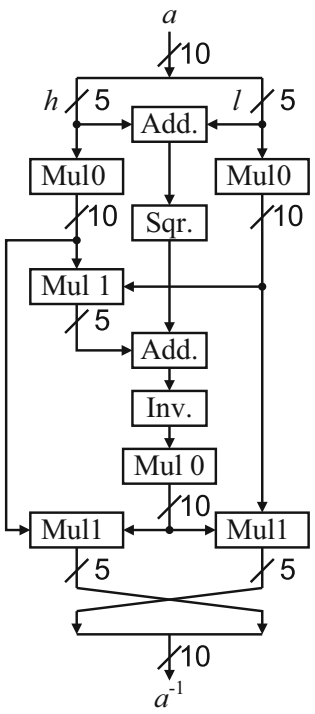


**Fig. 2** Inversion circuit over RRB-based $GF((2^4)^2)$ in [16]

## 3 Proposed $GF(2^8)$ inversion circuit

### 3.1 Circuit description

This section presents our proposed $GF(2^8)$ inversion circuit that takes full advantage of the above redundant GF arithmetic. The important ideas are to employ the tower field $GF((2^4)^2)$ inside the circuit and perform the subfield (i.e., $GF(2^4)$) operations using redundant GF arithmetic. We introduce PRR for the $GF(2^4)$ inversion because we can exploit a defining polynomial, $P(x) = x^5 + 1$, thanks to the irreducible AOP of degree 4. We also introduce RRB for the $GF(2^4)$ multiplication. In addition, we employ an NB for the input in order to exploit the Frobenius mapping feature, which performs the 16th power of input solely by wiring.

In accordance with ITA, our inversion circuit consists of three stages, as shown in Fig. 1. Here, we represent the inputs of Stages 1, 2, and 3 by NB, PRR, and RRB, respectively. In particular, we employ an NB that has a symmetric property, which makes it possible to convert the elements from NB to PRR without increasing the circuit delay.

Figure 3 shows a block diagram of our proposed circuit, where components $H$, $L$, and $F$, respectively, calculate $H_{i,j}$, $L_{i,j}$, and $F_{i',j'}$ described in the following. When input $a$ is represented by $h\alpha^{16} + l\alpha$, components NB2RRB convert $h$ and $l$ from NB to RRB solely by wiring. Note that $H$ and $L$ are shared with Stages 1 and 3. The stages in the proposed circuit are designed as follows:

#### 3.1.1 Calculation of the 16th and 17th powers

Stage 1 performs the 16th and 17th powers of input, where input $a$ is given by NB, and outputs $a^{16}$ and $a^{17}$ are given by RRB and PRR, respectively. Let $\alpha$ be a root of an irreducible polynomial of degree 2 over $GF(2^4)$. The irreducible polynomial is given by $\alpha^2 + \mu\alpha + \nu$, where $\mu$ and $\nu$ are constants of $GF(2^4)$. When input $a$ is represented by $a = h\alpha^{16} + l\alpha$ in an NB $\{\alpha^{16}, \alpha\}$, $a^{16}$ and $a^{17}$ are, respectively, given by

$$a^{16} = l\alpha^{16} + h\alpha, \quad (6)$$
$$a^{17} = hl\mu^2 + (h + l)^2\nu. \quad (7)$$

Equation (6) indicates that $a^{16}$ is performed by twisting wires.

The change-of-basis from NB to RRB does not require any additional gates because the NB (e.g., $\{\beta^4, \beta^3, \beta^2, \beta^1\}$) can be considered as a reduced version of RRB (e.g., $\{\beta^4,$

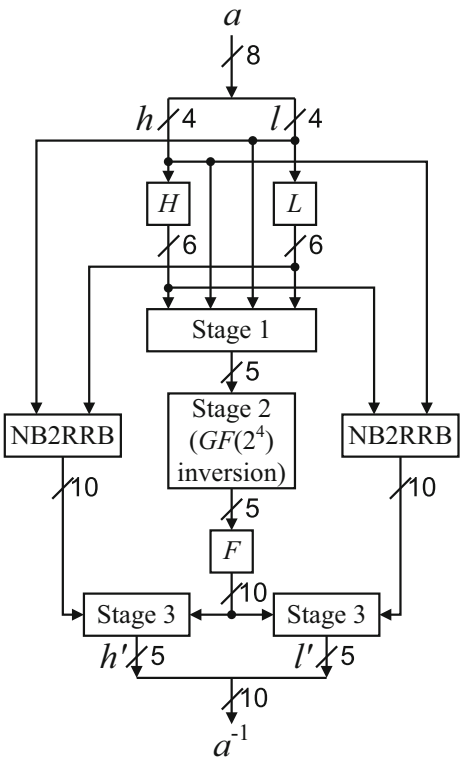


**Fig. 3** Proposed inversion circuit

$\beta^3, \beta^2, \beta^1, \beta^0\}$) with the same root of the AOP of degree 4. Conversely, the change-of-basis from NB to PRR requires some gates. However, the symmetric property of the NB used in our circuit provides a mapping that does not increase the circuit delay.

Let us now look at the change-of-basis from NB to PRR. Since the change-of-basis is given by an isomorphim represented by $z' = \Gamma(z)$, where an element $z$ in one GF representation is converted into an element $z'$ in another GF representation. In the binary vector form, the output $z'$ is obtained from the product of a conversion matrix $\gamma$ and the transposed input (i.e., $z' = \gamma z^T$) when the conversion matrix $\gamma$ represents the isomorphism $\Gamma$. The PRR-based $GF(2^4)$ is given with the defining polynomial $P(x) = x^5 + 1$ ($G(x) = x + 1$ and $H(x) = x^4 + x^3 + x^2 + x + 1$) and the conversion matrix from NB to PRR is as follows:

$$\phi = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix},$$

where the least significant bits are in the upper left corner. (See [20] for an explanation of how to obtain the matrix.) Let $d = d_4x^4 + d_3x^3 + \cdots + d_0$ be the output of Stage 1 (i.e., the 17th power of input in PRR), where $d_0, d_1, \ldots,$ and $d_4$ are elements of $GF(2)$. The output is provided by applying the change-of-basis $\Phi$ from NB to PRR to $a^{17}$ (i.e., the product of the conversion matrix $\phi$ and the transposed vector form of $a^{17}$). However, the multiplication of $\phi$ and the output of Eq. (7) requires an additional circuit with $2T_X$ delay if the multiplication is performed explicitly. To avoid such additional circuit, we derive another output equation from Eq. (7) as follows:

$$\begin{aligned} d &= \Phi(hl\mu^2 + (h+l)^2\nu) \\ &= \Phi(\mu^2(hl)) + \Phi(\nu((h+l)^2)) \\ &= \Phi'(hl) + \Phi''((h+l)^2), \end{aligned} \tag{8}$$

where $\Phi'$ and $\Phi''$ are the linear functions obtained by merging $\Phi$ with the constant multiplications of $\mu^2$ and $\nu$, respectively. Note that constant multiplications over GF can also be given as linear functions represented by conversion matrices. When $\mu = \beta^4 + \beta$ and $\nu = \beta$, the resulting matrices $\phi'$ and $\phi''$ representing, respectively, $\Phi'$ and $\Phi''$ are given as

$$\phi' = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}, \quad \phi'' = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix},$$

where the least significant bits are in the upper left corners. The resulting elements of PRR are shown in Table 1.

To design the circuit defined by Eq. (8), we exploit the ONB with the symmetric property that $h$ and $l$ are given by $h = h_4\beta^4 + h_3\beta^3 + h_2\beta^2 + h_1\beta$ and $l = l_4\beta^4 + l_3\beta^3 + l_2\beta^2 + l_1\beta$ with a common NB $\{\beta^4, \beta^3, \beta^2, \beta^1\}$, where $h_1, \ldots, h_4$ and $l_1, \ldots, l_4$ are the elements of $GF(2)$. Here, "symmetric" indicates that $h$ and $l$ is represented by the same NB while the circuit in [16] uses different (i.e., asymmetric) bases for $h$ and $l$ [2]. The usage of ONB makes it more efficient to compute $a^{17}$ and also makes it possible to perform the change-of-basis between the ONB, PRR, and RRB by solely bit-wise permutation because all of the used representations are defined by the AOP of degree 4. As a result, the outputs $d_0, d_1, \ldots,$ and $d_4$ are given by

$$\begin{aligned} d_0 = {} & (h_1l_2 + h_2l_1 + h_3l_4 + h_4l_3 + h_1l_1 + h_4l_4) \\ & + (h_1 + l_1 + h_3 + l_3 + h_4 + l_4), \end{aligned} \tag{9}$$

$$\begin{aligned} d_1 = {} & (h_1l_2 + h_2l_1 + h_1l_3 + h_3l_1 + h_2l_2 + h_4l_4) \\ & + (h_1 + l_1 + h_2 + l_2 + h_3 + l_3 + h_4 + l_4), \end{aligned} \tag{10}$$

[2] More precisely, we can construct five different bases using $\beta$, that is, four PBs $\{\beta^3, \beta^2, \beta^1, \beta^0\}$, $\{\beta^4, \beta^2, \beta^1, \beta^0\}$, $\{\beta^4, \beta^3, \beta^1, \beta^0\}$, and $\{\beta^4, \beta^3, \beta^2, \beta^0\}$ in addition to an ONB $\{\beta^4, \beta^3, \beta^2, \beta^1\}$. We use the ONB for $h$ and $l$ for efficient computation of Stage 1, while the previous work [16] used one of them for each $h$ and $l$ in order to construct efficient conversion matrices for the change-of-basis, MixColumns, and affine transformation.

$$d_2 = (h_1l_3 + h_3l_1 + h_1l_4 + h_4l_1 + h_2l_3 + h_3l_2 + h_2l_2) + (h_1 + l_1 + h_2 + l_2 + h_4 + l_4), \quad (11)$$

$$d_3 = (h_1l_4 + h_4l_1 + h_2l_3 + h_3l_2 + h_2l_4 + h_4l_2 + h_3l_3) + (h_2 + l_2 + h_3 + l_3 + h_4 + l_4), \quad (12)$$

$$d_4 = (h_2l_4 + h_4l_2 + h_3l_4 + h_4l_3 + h_1l_1 + h_3l_3) + (h_1 + l_1 + h_2 + l_2 + h_3 + l_3), \quad (13)$$

respectively. Here, the symmetric property enables us to factor Eqs. (9)–(13) as follows:

$$d_0 = H_{1,2} \vee L_{1,2} + H_{3,4} \vee L_{3,4} + h_2 \vee l_2 + h_3l_3, \quad (14)$$

$$d_1 = H_{1,2} \vee L_{1,2} + H_{1,3}L_{1,3} + h_3 \vee l_3 + h_4 \vee l_4, \quad (15)$$

$$d_2 = H_{1,3} \vee L_{1,3} + H_{1,4}L_{1,4} + H_{2,3} \vee L_{2,3} + h_4 \vee l_4, \quad (16)$$

$$d_3 = H_{1,4} \vee L_{1,4} + H_{2,3} \vee L_{2,3} + H_{2,4}L_{2,4} + h_1 \vee l_1, \quad (17)$$

$$d_4 = H_{2,4} \vee L_{2,4} + H_{3,4} \vee L_{3,4} + h_1 \vee l_1 + h_2l_2, \quad (18)$$

where $H_{i,j} = h_i + h_j$, $L_{i,j} = l_i + l_j$ $(1 \le i < j \le 4)$, and $\vee$ denotes the OR operator (i.e., $a \vee b = a + b + ab$). The component denoted by Stage 1 in Fig. 3 performs the computations corresponding to Eqs. (14)–(18). Therefore, the proposed Stage 1 is performed with only $T_A + 3T_X$ (or $T_O + 3T_X$) delay, whereas conventional $GF(((2^2)^2)^2)$ inversion implementations are performed with at least $6T_X$ delay, where $T_A$, $T_O$, and $T_X$ denote the delays of the AND, OR, and XOR gates, respectively. In addition, the proposed Stage 1 is implemented with 5 AND, 10 OR, and 27 XOR gates while the conventional design with $T_A + 3T_X$ delay requires 10 AND and 35 XOR gates [16].

#### 3.1.2 Subfield inversion

Stage 2 performs the inversion over the subfield $GF(2^4)$, where the input and output are given by PRR and RRB, respectively. We first describe the architecture of the PRR-based $GF(2^4)$ inversion and then show the change-of-basis from PRR to RRB below.

The inversion over $GF(2^4)$ is performed by the 14th power of the input. The input (i.e., the output of Stage 1) $d = d_4x^4 + d_3x^3 + \cdots + d_0$ is given as an element of the PRR-based $GF(2^4)$. The input is satisfied with the condition (called the linear recurrence relation) $d_0 + d_1 + d_2 + d_3 + d_4 = 0$ [3] because it is equivalent to the codeword of a CRC generated by $G(x) = x + 1$, which makes it possible to perform the exponentiation by bit-wise operations over the PRR-based $GF(2^4)$ in an efficient manner.

[3] The linear recurrence relation is used for error detection in CRC. A polynomial is a codeword of a CRC iff the relation is satisfied.

Let $e = e_4x^4 + e_3x^3 + \cdots + e_0$ be the inverse element of $d$ in PRR, where $e_0, e_1, \ldots,$ and $e_4$ are elements of $GF(2)$. Using the linear recurrence relation, we can derive $e_0, e_1, \ldots,$ and $e_4$ as follows:

$$e_0 = (d_1 \vee d_4)(d_2 \vee d_3), \quad (19)$$

$$e_1 = ((d_4 + 1)(d_1 + d_2)) \vee (d_0d_4(d_2 \vee d_3)), \quad (20)$$

$$e_2 = ((d_3 + 1)(d_2 + d_4)) \vee (d_0d_3(d_1 \vee d_4)), \quad (21)$$

$$e_3 = ((d_2 + 1)(d_1 + d_3)) \vee (d_0d_2(d_1 \vee d_4)), \quad (22)$$

$$e_4 = ((d_1 + 1)(d_3 + d_4)) \vee (d_0d_1(d_2 \vee d_3)). \quad (23)$$

According to Eqs. (19)–(23), the proposed Stage 2 requires $T_A + T_O + T_X$ delay, whereas the conventional structures [10–12,16] require at least $T_A + 3T_X$. Also, the proposed Stage 2 is implemented with 13 AND, 6 OR, and 4 XOR, and 4 NOT gates, whereas the conventional fastest Stage 2 [16] requires 12 AND, 11 XOR, and 2 XNOR gates. Note that when the multiplicative unit element $E(x)$ $(= x^4 + x^3 + x^2 + x)$ is given as the input, the output becomes not $E(x)$ but 1. However, the output is acceptable in Stage 3 (i.e., $GF(2^4)$ multiplication) because both $E(x)$ and 1 are the idempotent elements in the residue ring modulo $P(x)$.

Let us now look at the PRR-to-RRB mapping. To provide it uniquely, we focus on the definition of PRR in [14], in which the mapping $\Psi$ from PRR defined by $x$ to another representation defined by $\beta$ is isomorphism. According to [20], the change-of-basis from PRR can be performed by substituting a root of $H(x)$ (i.e., $\beta$) to elements. Let $f = f_4\beta^4 + f_3\beta^3 + \cdots + f_0$ be the output of Stage 2 in RRB, where $f_0, f_1, \ldots,$ and $f_4$ are elements of $GF(2)$. The output can be given by

$$f = \Psi(e) = e_4\beta^4 + e_3\beta^3 + e_2\beta^2 + e_1\beta + e_0.$$

This is because the RRB is defined using $H(x)$. This means that the PRR-to-RRB mapping is performed without any additional circuit, assuming that $f_0 = e_0, \ldots,$ and $f_4 = e_4$. As a result, the PRR-based design provides inversion and change-of-basis with fewer logic gates.

#### 3.1.3 Final multiplication

Stage 3 generates the final output using two $GF(2^4)$ multiplication operations, where both the inputs and output are given by RRB. Since Stage 3 solely consists of $GF(2^4)$ multiplication, Stage 3 can be efficiently implemented with the efficient RRB-based $GF(2^4)$ multipliers described in Sect. 2.2.

Let $h' = h'_4\beta^4 + h'_3\beta^3 + \cdots + h'_0$ be the upper 5 bits of the final output $a^{-1}$ in RRB, where $h'_0, h'_1, \ldots,$ and $h'_4$ are elements of $GF(2)$. Multiplying $f$ and $l$, we can calculate elements $h'_0, h'_1, \ldots,$ and $h'_4$ as follows:

$$h'_0 = L_{1,4}F_{1,4} + L_{2,3}F_{2,3}, \quad (24)$$

**Table 2** Critical delay and gate count of inversion circuits over tower fields

| | Field | Representation | | Gate count* | Critical delay |
|---|---|---|---|---|---|
| | | Tower field of $GF(2^8)$ | Intermediate field | ($g_0$, $g_1$, $g_2$, $g_3$, $g_4$, $g_5$, $g_6$) | |
| Satoh et al. [10] | $GF(((2^2)^2)^2)$ | PB | PB | (30, 0,96, 0, 0, 6, 0) | $4T_A + 17T_X$ |
| Canright [11] | $GF(((2^2)^2)^2)$ | NB | NB | (0, 0, 56,0, 0, 34, 6) | $4T_A + 15T_X$ |
| Nogami et al. [12] | $GF(((2^2)^2)^2)$ | PB and NB | PB and NB | (36, 0, 95, 0, 0, 0, 0) | $4T_A + 14T_X$ |
| Jeon et al. [13] | $GF((2^4)^2)$ | PB | PB | (58, 2,67, 0, 0, 0, 0) | $4T_A + 10T_X$ |
| Nekado et al. [16] | $GF((2^4)^2)$ | NB | RRB | (42, 0, 68,2, 0, 0, 0) | $4T_A + 7T_X$ |
| This work (compact) | $GF((2^4)^2)$ | NB | NB, PRR, and RRB | (38, 16, 51, 0, 4, 0, 0) | $3T_A + T_O + 6T_X$ |
| This work (high-speed) | $GF((2^4)^2)$ | NB | NB, PRR, and RRB | (45,10, 57, 0, 10, 0, 0) | $3T_A + 6T_X$ |

*$g_0$, $g_1$, $g_2$, $g_3$, $g_4$, $g_5$, and $g_6$ denote the number of AND, OR, XOR, XNOR, NOT, NAND and NOR gates, respectively

$$h'_1 = l_1 F_{0,1} + L_{2,4} F_{2,4}, \tag{25}$$
$$h'_2 = l_2 F_{0,2} + L_{3,4} F_{3,4}, \tag{26}$$
$$h'_3 = l_3 F_{0,3} + L_{1,2} F_{1,2}, \tag{27}$$
$$h'_4 = l_4 F_{0,4} + L_{1,3} F_{1,3}, \tag{28}$$

where $F_{i',j'}$ denotes $f_{i'} + f_{j'}$ $(0 \le i' < j' \le 4)$. The lower five bits of $a^{-1}$ (denoted by $l'$) are also obtained in the same manner as in Eqs. (24)–(28). The component denoted by Stage 3 in Fig. 3 performs the computations corresponding to Eqs. (24)–(28). Note here that the calculations for $F_{i',j'}$ can be shared within Stage 3. As a result, the number of circuit components for the two multipliers in Stage 3 is reduced.

The above inversion circuit achieves the shortest critical delay among tower field inversion circuits as evaluated in Sect. 4. In the following, we describe a variation of the proposed circuit with a smaller critical delay at the cost of a slight area overhead. We focus on Stage 2 and $F_{i',j'}$ computation prior to Stage 3. Since Stage 2 is given by one-to-one mapping and $F_{i',j'}$ is computed by XORing the output of Stage 2, we can unify Stage 2 and $F_{i',j'}$ computation by deriving $F_{i',j'}$ directly from $d_0, d_1, \ldots$, and $d_4$ as follows:

$$F_{0,1} = d_2(d_1 d_3 + 1) + d_0 d_1 + d_3 d_4, \tag{29}$$
$$F_{0,2} = d_4(d_1 d_2 + 1) + d_0 d_2 + d_1 d_3, \tag{30}$$
$$F_{0,3} = d_1(d_3 d_4 + 1) + d_0 d_3 + d_2 d_4, \tag{31}$$
$$F_{0,4} = d_3(d_2 d_4 + 1) + d_0 d_4 + d_1 d_2, \tag{32}$$
$$F_{1,2} = d_1(d_0 d_2 + 1) + d_0 d_4 + d_2 d_3, \tag{33}$$
$$F_{1,3} = d_3(d_0 d_1 + 1) + d_0 d_2 + d_1 d_4, \tag{34}$$
$$F_{1,4} = d_0(d_2 d_3 + 1) + d_1 d_3 + d_2 d_4, \tag{35}$$
$$F_{2,3} = d_0(d_1 d_4 + 1) + d_1 d_2 + d_3 d_4, \tag{36}$$
$$F_{2,4} = d_2(d_0 d_4 + 1) + d_0 d_3 + d_1 d_4, \tag{37}$$
$$F_{3,4} = d_4(d_0 d_3 + 1) + d_0 d_1 + d_2 d_3. \tag{38}$$

The above computation for each $F_{i',j'}$ requires $T_A + 2T_X$ delay while the critical delay of the original circuit in Fig. 3 requires $T_A + T_O + 2T_X$. On the other hand, we require 20 AND, 20 XOR, and 10 NOT gates to compute $F_{i',j'}$ based on Eqs. (29)–(38), whereas the original circuit requires 13 AND, 6 OR, and 14 XOR, and 4 NOT gates (i.e., 13 AND, 6 OR, and 4 XOR, and 4 NOT gates for computing Stage 2, and 10 XOR gates for computing $F_{i',j'}$). Thus, we can further reduce the critical delay of the inversion circuit at the expense of a few additional gates.

### 3.2 Implementation results

Table 2 shows the circuit delay and gate count of the proposed inversion circuit, where ($g_0$, $g_1$, $g_2$, $g_3$, $g_4$, $g_5$, $g_6$) in the gate count column, respectively, indicate the number of AND, OR, XOR, XNOR, NOT, NAND, and NOR gates, and representation indicates the GF representation(s) used in the circuit. In the representation column, "Tower field of $GF(2^8)$" denotes the representations for $GF(((2^2)^2)^2)$ or $GF((2^4)^2)$, and "Intermediate field" denotes $GF((2^2)^2)$ or $GF(2^4)$. "This work (compact)" denotes the inversion circuit in Fig. 3, and "This work (high-speed)" denotes the circuit where Stage 2 and $F_{i',j'}$ computation are unified as described in the last paragraph of Sect. 3. For comparison, Table 2 also shows those of the conventional inversion circuits. The critical delay of all the conventional ones were given by reference to [16]. On the other hand, the gate counts of the conventional ones were individually given because there were no single reference data covering all of them. The gate count of [11] was given from the original paper, that of [10] was given from a public source code by the authors [22], and those of [13,16] were given by reference to [16]. The gate count of [12] was given from a straightforward structure designed by us according to [12] since there were neither public data nor source code.

The critical paths of Stages 1, 2, and 3 in the proposed circuit require $T_A + 3T_X$, $T_A + T_O + T_X$, and $T_A + 2T_X$ delay, respectively. In our high-speed version, that of Stages 2 and 3 is at most $3T_A + 2T_X$. As a result, the total delay of our inversion circuit is $3T_A + T_O + 6T_X$ (or $3T_A + 6T_X$),

**Table 3** Performance evaluation of inversion circuits over tower fields

| | Area (GE) | Time (ns) | Area–time product |
|---|---|---|---|
| Satoh et al. [10] | 210.50 | 3.02 | 635.72 |
| Canright [11] | 178.00 | 2.92 | 519.75 |
| Nogami et al. [12] | 291.50 | 3.67 | 1,069.81 |
| Jeon et al. [13] | 221.25 | 2.19 | 474.54 |
| Nekado et al. [16] | 204.50 | 1.89 | 386.51 |
| This work (compact) | 174.75 | 1.81 | 316.30 |
| This work (high-speed) | 187.50 | 1.55 | 290.63 |

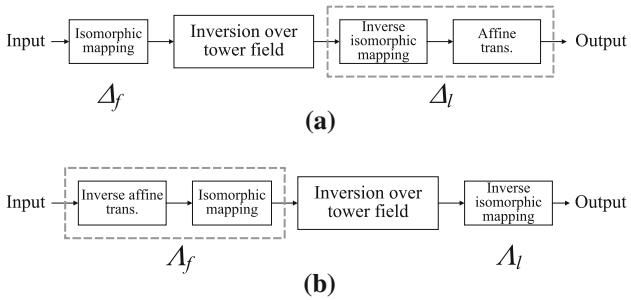


**Fig. 4** Overview of AES **a** S-Box and **b** inverse S-Box based on tower field arithmetic

which is the fastest compared with the other inversion circuits. The gate count in "This work (high-speed)" is smaller or comparable to the conventional ones because the number of additional XOR and XNOR gates due to unification is trivial. In total, the high-speed version newly designed in this paper is more efficient than any other circuits including our compact version.

To conduct a detailed evaluation, the above tower field $GF(2^8)$ inversion circuits were synthesized using Synopsys Design Compiler with a TSMC 65-nm CMOS standard cell library. Table 3 shows the synthesis results, where area indicates the circuit area estimated based on a two-way NAND equivalent gate size (i.e., gate equivalents (GE))[4], time indicates the circuit delay under the worst-case conditions, and area–time product indicates the product of area and time. For the best performance comparison, an area optimization option (which maximizes the effort of minimizing the number of gates without flattening the description) was applied. Note that the results were consistent even when the following speed optimization (which searches for the minimum timing without increasing the area obtained from the prior area optimization) options was applied. The conventional inversion circuits were also synthesized and evaluated using the same tools and options. The source codes of [10] and [11] were obtained from authors' Web sites [22,23], respectively. (Like them, we also applied gate-reduction techniques to our inversion circuit.) The source codes of [12], [13], and [16] were described by the authors according to the structures given in the papers.

Our compact inversion circuit achieved the smallest area although the total gate count of the proposed circuit was roughly the same as the conventional ones [11,16]. The less XOR gates in our circuit would lead to the smaller circuit area because an XOR (or XNOR) gate requires larger area than a NAND (or NOR) gate in standard cell libraries. Consequently, we confirmed that the compact version of the proposed circuit achieved the smallest area of 174.75 GE, the smaller circuit delay of 1.81 ns compared with conventional other circuits. In addition, we performed a dynamic gate-level timing simulation to estimate power consumption with Verilog-XL Simulator at the operation frequency of 100 MHz. As a result, we confirmed that the power consumption of our circuit was 38% lower than the conventional best in our environment. Moreover, the high-speed version of the proposed circuit achieved the smallest critical delay of 1.55 ns, and the area–time product was 24.8% smaller than that of the conventional best circuit, respectively.

[4] While we calculated GE values from synthesis results with an inverter cell information in the preliminary version [17], in this paper, we derived these values directly from a NAND cell in order to accommodate them to a result in [6]. Note that the GE values in this paper are consistent with those in the previous version [17].

## 4 Application to AES S-Box

### 4.1 Description of AES S-Box

The proposed inversion circuit was efficiently applied to the AES S-Box design. The AES S-Box consists of a $GF(2^8)$ inversion and an affine transformation over $GF(2)$. Here, the $GF(2^8)$ is represented in a PB with an irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. Therefore, a change-of-basis between $GF(2^8)$ and $GF((2^4)^2)$ is required if the inversion over $GF((2^4)^2)$ is applied. Figure 4 shows an overview of the AES S-Box with tower field arithmetic. In an S-Box (Fig. 4a), the input (in the PB-based $GF(2^8)$) is initially mapped to the tower field by applying an change-of-basis $\Delta_f$ which is given by an isomorphim. After the inversion operation over the tower field, the inverse mapping and affine transformation are finally performed in series. Here, we can merge the inverse mapping into the affine transformation because both of them are represented in the form of constant matrices over $GF(2)$. The merged mapping is denoted by $\Delta_l$. This merging reduces the delay and gate counts. On the other hand, in an inverse S-Box (Fig. 4b), the inverse affine transformation is performed prior to change-of-basis and tower field inversion. Hence, the inverse affine transformation and change-of-basis are unified as $\Lambda_f$, and the inverse change-of-basis $\Lambda_l$ is solely performed after the tower field inversion.

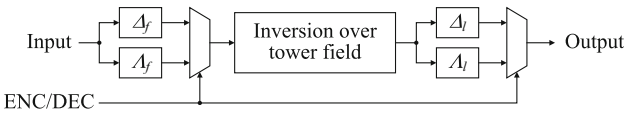


**Fig. 5** Typical architecture of unified S-Box

The matrices for the change-of-basis $\Delta_f$ and $\Delta_l$ ($\Lambda_f$ and $\Lambda_l$) have an impact on the performance of S-Box. When tower field $GF((2^4)^2)$ is used, the matrices are defined by the bases of $GF(2^4)$ and the defining polynomials for the extension of $GF(2^4)$ to $GF((2^4)^2)$. The efficiency of the two matrices for change-of-basis $\Delta_f$ and $\Delta_l$ ($\Lambda_f$ and $\Lambda_l$) is determined by the largest Hamming weight in the columns. For example, if the largest Hamming weight in the columns is four, the critical path becomes $2T_X$ delay. If it is five, the critical path becomes $3T_X$ delay. Therefore, the matrices should be selected with a view to minimizing the largest Hamming weight in the columns.

Some techniques for optimizing change-of-basis between AES field and $GF((2^4)^2)$ have been reported in [16,24,25]. In [24,25], an algorithm which searches all construction of isomorphic mappings from/to PB-based $GF((2^4)^2)$ was used. In addition, a technique in [16] used More Miscellaneously Mixed Bases (MMMB) to expand search space of the isomorphim for change-of-basis by utilizing asymmetric property of input of inversion circuit given by RRB. However, these technique cannot be applied to our inversion circuits because the irreducible polynomial of $GF(2^4)$ should be given by the AOP of degree 4 and the input of inversion circuit should be given by symmetric NB.

In this paper, we design a unified S-Box that supports both encryption and decryption shown in Fig. 5 in addition to the S-Box. There are efficient conversion matrices for either encryption or decryption in the proposed inversion circuit. However, we found that there is no efficient conversion matrix for both encryption and decryption in the structure of Fig. 5 due to the deficiency of search space of isomorphim for change-of-basis. Therefore, we introduce a new technique for expanding the search space. Figure 6 illustrates the AES S-Box structures with the proposed technique, where the component "constant multiplication" perform a multiplication over PB-based $GF(2^8)$ with a fixed value. The S-Box based on tower field arithmetic (denoted by $S$) is represented by $S(a) = A(\Theta'((\Theta(a))^{-1})) + 0\text{x}63$, where $\Theta$ is a change-of-basis from the PB-based $GF(2^8)$ to a tower field, $\Theta'$ is its inverse change-of-basis, and $A$ denotes the linear mapping of the affine transformation. We can rewrite the equation using a nonzero fixed value $c$ (in the PB-based $GF(2^8)$) as follows:

$$S(a) = A(c(\Theta'((\Theta(c(a)))^{-1}))) + 0\text{x}63.$$

Because the multiplication with $c$ is a linear mapping, we can unify $c$ and $\Theta$ as $\Delta_f$, and unify $\Theta'$, $c$, and $A$ as $\Delta_l$. The fixed value $c$ can take one of 255 elements over $GF(2^8)$. Thus, we can increase the variety of conversion matrices by 255 times. The proposed technique can be also applied to the inverse S-Box $S'$ as follows:

$$S'(a) = c'(\Theta'((((\Theta(c'(A'(a)))))^{-1})) + \Theta(c'(0\text{x}05)),$$

where $A'$ is the linear mapping of the inverse affine transformation and $c'$ is a nonzero fixed value for decryption. Note that $c$ and $c'$ do not need to be the same in general.

With the proposed technique, we successfully found efficient conversion matrices $\delta_f$, $\delta_l$, $\lambda_f$, and $\lambda_l$, respectively, for $\Delta_f$, $\Delta_l$, $\Lambda_f$, and $\Lambda_l$ when the $GF(2^4)$ elements of Stage 1 are represented in an NB $\{\beta^4, \beta^3, \beta^2, \beta^1\}$ and the defining polynomial for the extension is given by $\alpha^2 + (\beta^4 + \beta)\alpha + \beta$. As a concrete example, the matrices $\delta_f$, $\delta_l$, $\lambda_f$, and $\lambda_l$ are given by

$$\delta_f = \begin{pmatrix} 1&1&0&0&1&0&1&0 \\ 1&1&0&0&1&0&0&0 \\ 0&1&1&0&0&1&1&0 \\ 0&0&0&1&1&0&0&1 \\ 0&1&1&0&1&0&0&0 \\ 1&1&1&1&0&0&0&0 \\ 0&1&0&0&0&1&1&0 \\ 1&0&1&0&0&1&0&0 \end{pmatrix}, \delta_l = \begin{pmatrix} 1&0&1&1&1&0&1&1&1 \\ 1&0&1&0&0&1&0&0&1 \\ 0&0&1&1&0&1&0&1&1 \\ 0&0&0&0&0&1&1&1&0 \\ 0&1&0&0&1&0&0&1&0 \\ 1&0&1&0&0&1&0&1&0 \\ 1&0&0&1&0&0&1&1&1 \\ 1&0&1&0&0&1&0&0&0 \end{pmatrix},$$

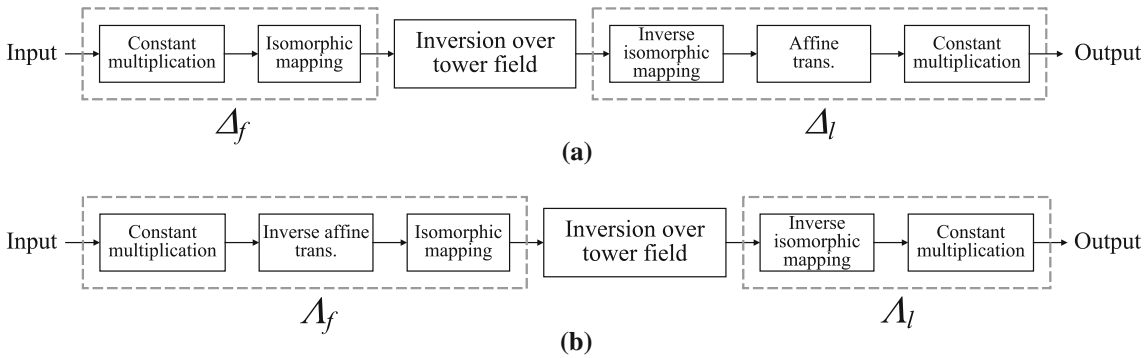


**Fig. 6** AES **a** S-Box and **b** inverse S-Box with proposed technique for optimizing linear mappings

**Table 4** Performance comparison of S-Boxes based on tower field arithmetic

| | Critical delay | | | Number of XOR gates | | Area | Time | Area–time |
|---|---|---|---|---|---|---|---|---|
| | $\Delta_f$ | Inversion | $\Delta_l$ | $\Delta_f$ | $\Delta_l$ | (GE) | (ns) | product |
| Satoh et al. [10] | $3T_X$ | $4T_A + 17T_X$ | $3T_X$ | 24 | 21 | 283.50 | 4.41 | 1,250.24 |
| Canright [11] | $3T_X$ | $4T_A + 15T_X$ | $3T_X$ | 24 (in total) | | 236.75 | 4.30 | 1,018.04 |
| Nogami et al. [12] | $2T_X$ | $4T_A + 14T_X$ | $2T_X$ | 20 | 19 | 392.00 | 4.78 | 1,873.77 |
| Jeon et al. [13] | $3T_X$ | $4T_A + 10T_X$ | $3T_X$ | 10 | 31 | 312.25 | 4.82 | 1,505.03 |
| Nekado et al. [16] | $2T_X$ | $4T_A + 7T_X$ | $3T_X$ | 17 | 36 | 289.50 | 3.29 | 952.46 |
| This work (compact) | $2T_X$ | $3T_A + T_O + 6T_X$ | $3T_X$ | 15 | 21 | 249.00 | 3.04 | 756.96 |
| This work (high-speed) | $2T_X$ | $3T_A + 6T_X$ | $3T_X$ | 15 | 21 | 261.50 | 2.78 | 726.98 |

**Table 5** Performance comparison of unified S-Boxes based on tower field arithmetic

| | Critical delay | | | Number of XOR gates | | Area | Time | Area–time |
|---|---|---|---|---|---|---|---|---|
| | $\Delta_f$ or $\Lambda_f$ | Inversion | $\Delta_l$ or $\Lambda_l$ | $\Delta_f + \Lambda_f$ | $\Delta_l + \Lambda_l$ | (GE) | (ns) | product |
| Satoh et al. [10] | $3T_X$ | $4T_A + 17T_X$ | $3T_X$ | 46 | 44 | 366.75 | 4.94 | 1,811.75 |
| Canright [11] | $3T_X$ | $4T_A + 15T_X$ | $3T_X$ | 20 | 18 | 311.25 | 4.97 | 1,546.91 |
| Jeon et al. [13] | $3T_X$ | $4T_A + 10T_X$ | $3T_X$ | 22 | 31 | 381.00 | 5.93 | 2,259.33 |
| Nekado et al. [16] | $2T_X$ | $4T_A + 7T_X$ | $3T_X$ | 32 | 68 | 372.50 | 3.66 | 1,363.36 |
| This work (compact) | $2T_X$ | $3T_A + T_O + 6T_X$ | $3T_X$ | 32 | 56 | 334.75 | 3.33 | 1,114.71 |
| This work (high-speed) | $2T_X$ | $3T_A + 6T_X$ | $3T_X$ | 32 | 56 | 347.50 | 3.05 | 1,059.87 |

$$\lambda_f = \begin{pmatrix} 0&0&0&0&1&1&0&0\\ 1&0&0&0&0&1&0&0\\ 1&1&1&0&1&0&0&0\\ 0&0&1&0&1&1&0&0\\ 0&1&0&1&0&0&1&1\\ 0&0&1&1&1&1&0&0\\ 0&0&1&1&1&0&0&0\\ 1&1&0&0&0&1&1&0 \end{pmatrix}, \lambda_l = \begin{pmatrix} 1&0&0&1&0&0&1&0&0&1\\ 0&0&1&0&1&0&1&0&1&0\\ 0&0&0&1&1&0&0&0&0&0\\ 0&1&1&1&1&1&1&0&0&0\\ 1&0&1&0&0&1&1&0&0&0\\ 1&0&1&0&0&0&1&1&0&0\\ 1&1&1&1&0&0&1&0&0&1\\ 0&0&0&0&0&0&0&1&0&1 \end{pmatrix},$$

where the least significant bits are in the upper left corners. The vector of the inverse affine transformation is given by $\Theta(c'(0\text{x}05)) = 0\text{x}29$. Here, the largest Hamming weight in each column of $\delta_f$ and $\lambda_f$ is at most four while that of $\delta_l$ and $\lambda_l$ is at most six. This means that the former and latter mappings are implemented with at most $2T_X$ and $3T_X$ delays, respectively.

## 4.2 Implementation results

Tables 4 and 5 show the critical delay and the number of XOR gates required for the change-of-basis of the proposed AES S-Box and unified S-Box compared with the conventional implementations, respectively. Here, the numbers of XOR gates for Canright's and Jeon's S-Boxes, which were achieved by factorizing XOR gates, were given according to their papers. We applied the similar factorization technique to our S-Boxes. On the other hand, those for other S-Boxes were derived directly from conversion matrices without any factorization. Our circuits achieve $3T_A+T_O+11T_X$ or $3T_A+11T_X$ delay, which is smaller than the conventional S-Boxes with tower field arithmetic[5]. Tables 4 and 5 also show the synthesis results (area and delay time) obtained from the same tool, synthesis options, and method as the above. The source codes were given from the same methods as Table 3. Note here that Canright's design in [11] supports both encryption and decryption, and we have slightly changed it to support only encryption to allow a fair comparison to our design (for Table 4). On the other hand, since Satoh's design supports either encryption or decryption, we have also changed it to support both encryption and decryption as described in Fig. 5 (for Table 5). As a result, the area–time products of our AES S-Boxes unified S-Boxes were, respectively, 28.1 and 31.5% better than Canright's ones, which had been the smallest for a long time, and were also, respectively, 23.2 and 22.3% better than Nekado's latest ones. The power consumption of our S-Boxes and unified S-Boxes, which were estimated in the same manner as Sect. 4, were, respectively, 40.8 and 42.1% better than Canright's ones, and were also, respectively, 33.2 and 31.2% better than Nekado's ones in our environment.

[5] According to [26,27], a logic minimization method can further reduce the total gates or critical delay of [11,12]. However, the same minimization can also be applied to other circuits including ours. Therefore, we did not apply the minimization in this paper. Note that, in our environment, the critical delay and area–time product of our S-Boxes without the minimization technique are smaller than those in [26,27].

Note that the results of inverse S-Boxes would be consistent with Table 4.

A further discussion point when applying the proposed method to cryptographic cores is the well-known side channel issue. In particular, the resource sharing of Stages 1 and 3 would cause glitches during the computation. To apply our method to masking-based countermeasures with pipelining such as threshold implementation (TI) [28,29] and generalized masking scheme (GMS) [30,31] to defeat sophisticated (higher-order) attacks utilizing glitches, we need to decompose shared resources, which results in the increase of 12 XOR gates for the compact version or 17 XOR gates for high-speed version in total. Note, however, that such increase would also happen in other works (e.g., [10,11,16]) using the similar optimization. In contrast, our method is more suitable for multiplexing-based countermeasures, such as WDDL [32], due to the high efficiency. A further and comprehensive study on the side channel security is definitely one of the important future topics for our method.

## 5 Conclusion

This paper proposed a new $GF(2^8)$ inversion circuit that utilizes a combination of non-redundant and redundant GF arithmetic. The proposed circuit, which is based on tower field arithmetic, was designed by utilizing PRR and RRB for the subfield inversion and multiplication, respectively. The flexibility of such redundant representations can provide efficient change-of-basis from/to $GF(2^8)$. The efficiency of our proposed inversion circuit and its AES S-Box was evaluated by gate count and logic synthesis results with a 65-nm CMOS standard cell library. Consequently, we confirmed that the newly proposed inversion circuit was approximately 47% faster than the conventional best $GF(((2^2)^2)^2)$ circuit with 5% area overhead. In addition, we proposed a new optimization technique for change-of-basis between AES and tower field. The proposed S-Boxes and unified S-Boxes achieved the best efficiency in comparison with any other existing S-Boxes based on tower field arithmetic.

Redundant GF representations, such as PRR and RRB, provide high flexibility for GF arithmetic circuit design. It is definitely possible to obtain efficient circuit structures using them because the search space of isomorphism for change-of-basis increases as a result of their flexibility. In addition, a combination of non-redundant and redundant GF representations has the potential to further improve GF circuits, as shown in this paper. Further research is being conducted to expand the application of the design methodology based on hybrid GF arithmetic.

**Acknowledgements** We are deeply grateful to Dr. Amir Moradi and Mr. Yukihiro Sugawara for their insightful and valuable advices. This work has been supported by JSPS KAKENHI Grant Nos. 16K12436, 17H00729, and 16J05711.

## References

1. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, Berlin (1993)
2. Matsui, M.: The first experimental cryptoanalysis of the data encryption standard. In: Advances in Cryptology—CRYPTO 1994. Volume 839 of Lecture Notes in Computer Science, pp. 1–11. Springer (1994)
3. Nyberg, K.: Differentially uniform mappings for cryptography. In: Advances in Cryptology—EUROCRYPT 1993. Volume 765 of Lecture Notes in Computer Science, pp. 55–64. Springer (1993)
4. National Institute of Standards and Technology (NIST): Advanced encryption standard (AES), vol. 197. FIPS Publication. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (November 2001)
5. Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: *Camallia*: A 128-bit block cipher suitable for multiple platforms—design and analysis. In: Selected Areas in Cryptography. Volume 2012 of Lecture Notes in Computer Science, pp. 39–56. Springer (2001)
6. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: a very compact and a threshold implementation of AES. In: Advances in Cryptology—EUROCRYPT 2011. Volume 6632 of Lecture Notes in Computer Science, pp. 59–88. Springer (2011)
7. Sasao, T.: And-Exor expressions and their optimization. In: Sasao, T. (ed.) Logic Synthesis and Optimization. Volume 212 of The Kluwer International Series in Engineering and Computer Science, pp. 287–312. Kluwer Academic Publishers, Dordrecht (1993)
8. Morioka, S., Satoh, A.: An optimized S-Box circuit architecture for low power AES design. In: Cryptographic Hardware and Embedded Systems (CHES). Volume 2523 of Lecture Notes in Computer Science, pp. 172–186. Springer (2002)
9. 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)
10. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A compact Rijndael hardware architecture with S-box optimization. In: Advances in Cryptology—ASIACRYPT 2001. Volume 2248 of Lecture Notes in Computer Science, pp. 239–254. Springer (2001)
11. Canright, D.: A very compact S-box for AES. In: Cryptographic Hardware and Embedded Systems (CHES). Volume 3659 of Lecture Notes in Computer Science, pp. 441–455. Springer (2005)
12. 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: Cryptographic Hardware and Embedded Systems (CHES). Volume 6225 of Lecture Notes in Computer Science, pp. 234–247. Springer (2010)
13. Jeon, Y., Kim, Y., Lee, D.: A compact memory-free architecture for the AES algorithm using resource sharing methods. J. Circuits Syst. Comput. **19**(5), 1109–1130 (2010)
14. Drolet, G.: A new representation of elements of finite fields $GF(2^m)$ yielding small complexity arithmetic circuits. IEEE Trans. Comput. **47**(9), 938–946 (1997)
15. Wu, H., Hasan, A., Blake, I.F.: Highly regular architectures for finite field computation using redundant basis. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES). Volume 1717 of Lecture Notes in Computer Science, pp. 269–279. Springer (1999)
16. Nekado, K., Nogami, Y., Iokibe, K.: Very short critical path implementation of AES with direct logic gates. In: Advances in Information and Computer Security. Volume 7631 of Lecture Notes in Computer Science, pp. 51–68. Springer (2012)

17. 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: International Workshop on Cryptographic Hardware and Embedded Systems (CHES). Volume 9293 of Lecture Notes in Computer Science, pp. 63–80. Springer (2015)
18. Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases. Inf. Comput. **78**, 171–177 (1988)
19. Wu, H.: Low complexity bit-parallel finite field arithmetic using polynomial basis. In: Workshop on Cryptographic Hardware and Embedded Systems (CHES). Volume 1717 of Lecture Notes in Computer Science, pp. 280–291. Springer (1999)
20. Hirotomo, M., Mouri, K., Morii, M.: Generalized polynomial ring representation over $GF(2^m)$ and its application. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **J89–A**(10), 790–800 (2006). (Japanese Edition)
21. Nogami, Y., Saito, A., Morikawa, Y.: Finite extension field with modulus of all-one polynomial field and representation of its elements of for fast arithmetnic operations. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E86–A**(9), 2376–2387 (2003)
22. Tohoku University: Cryptographic hardware project. http://www.aoki.ecei.tohoku.ac.jp/crypto/ May 2015
23. Canright, D.: http://faculty.nps.edu/drcanrig/ May (2015)
24. 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 Circuits **46**, 767–776 (2011)
25. 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 Circuits **50**, 1048–1058(2015)
26. Boyer, J., Matthews, P., Peralta, P.: Logic minimization techniques with applications to cryptology. J. Cryptol. **47**, 280–312 (2013)
27. Boyer, J., Peralta, R.: A small depth-16 circuit for the AES S-box. In: Information Security and Privacy Research. Volume 376 of IFIP Advances in Information and Communication Technology, pp. 287–298. Springer (2012)
28. Nikova, S., Rijmen, V., Schläffer, M.: Secure hardware implementation of nonlinear functions in the presence of glitches. J. Cryptol. **24**, 292–321 (2011)
29. Bilgin, B., Gierlichs, B., Nikov, V., Rijmen, V.: Higher-order threshold implementations. In: Advances in Cryptology—ASIACRYPT 2014. Volume 8874 of Lecture Notes in Computer Science, pp. 326–343. Springer (2014)
30. Reparaz, O., Bilgin, B., Nikova, S., Gierlichs, B., Verbauwhede, I.: Consolidating masking schemes. In: Advances in Cryptology—CRYPTO 2015. Volume 9215 of Lecture Notes in Computer Science, pp. 764–783. Springer (2015)
31. Cnudde, T.D., Reparaz, O., Bilgin, B., Nikova, S., Nikov, V., Rijmen, V.: Masking AES with $d+1$ shares in hardware. In: International Conference on Cryptographic Hardware and Embedded Systems (CHES). Volume 9813 of Lecture Notes in Computer Science, pp. 194–212. Springer (2016)
32. 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)