New Area Record for the AES Combined S-box/Inverse S-box

Arash Reyhani-Masoleh, Mostafa Taha and Doaa Ashmawy
Department of Electrical and Computer Engineering
Western University, London, Ontario, Canada
{areyhani,mtaha9,dashmawy}@uwo.ca

***Abstract*—The AES combined S-box/inverse S-box is a single construction that is shared between the encryption and decryption data paths of the AES. The currently most compact implementation of the AES combined S-box/inverse S-box is Canright's design, introduced back in 2005. Since then, the research community has introduced several optimizations over the S-box only, however the combined S-box/inverse S-box received little attention. In this paper, we propose a new AES combined S-box/inverse S-box design that is both smaller *and* faster than Canright's design. We achieve this goal by proposing to use new tower field and optimizing each and every block inside the combined architecture for this field. Our complexity analysis and ASIC implementation results in the CMOS STM 65nm and NanGate 15nm technologies show that our design outperforms the counterparts in terms of area and speed.**

I. Introduction

The implementation of the AES S-box is one of the most extensively studied areas of cryptography [1]. The S-box performs a non-linear mapping between an 8-bit input to an 8-bit output, by computing the multiplicative inverse in $GF(2^8)$, followed by an affine transformation and addition with a constant. The multiplicative inverse can be computed in the native $GF(2^8)$ representation [1], however the implementation cost (area and delay) would be higher than the cost of using composite/tower fields.

In 2001, Satoh et al. proposed to compute the S-box output by first converting the field to an isomorphic representation in $GF(((2^2)^2)^2)$, where the multiplicative inversion can be performed efficiently, then converting the output back to the corresponding representation in $GF(2^8)$ [2]. In 2005, Canright proposed an exhaustive search over all the possible representations in $GF(((2^2)^2)^2)$, expressed as polynomial basis as well as normal basis (a total of 432 representations), and proposed a very compact design for the S-box, the inverse S-box and the combined S-box/inverse S-box [3], [4]. These designs served as hardware benchmarks for low-area designs of AES.

Faster and/or more efficient S-boxes were proposed in [5]–[8], while faster and/or more efficient combined S-boxes/inverse S-box were proposed in [9]–[12]. In addition, an automated search for a field representation in $GF((2^4)^2)$ was conducted in [13], [14] to optimize the area of high speed AES cores to be suitable for memory encryption engines.

The research for a more compact S-box only was explored in a line of work by Boyar et al. [15]–[17] and Reyhani-Masoleh et al. [8]. They proposed to move all the linear operations in the input and the output of the composite/tower field inversion circuit to the input and output isomorphic mappings, respectively. Then, logic-minimization heuristics were proposed to reduce the number of XOR gates that is required to implement the extended isomorphic mappings. The currently smallest design of the AES S-box can be found in [8]. Note that [8] used the composite field over $GF((2^4)^2)$, whereas in this paper and [15]–[17], the tower field over $GF(((2^2)^2)^2)$ is considered.

To the best of our knowledge, the research for a combined S-boxes/inverse S-box received little attention. The line of research that uses extended isomorphic mappings cannot be ported to the combined S-boxes/inverse S-box due to the high number of multiplexers required between the isomorphic mappings and the inversion circuit. The contributions in [18], [19] used a combined S-box/inverse S-box that is based on the one introduced back in 2002 [20], which required more resources than the one proposed by Canright. Another compact design was proposed in [21] as an application for a new, non standard cell, XOR gate. Despite using two different implementation technologies, the hardware complexity (expressed in number of gates) shows that Canright design was more compact. In fact, a very recent contribution in [22] proposed an overall AES encryption/decryption engine using no other than Canright combined S-box/inverse S-box circuit. Protection against side-channel analysis was proposed in [23]. In this paper, we do not explicitly target side-channel protection. We will explore side-channel protection as a future research target.

In this paper, we propose a combined S-box/inverse S-box circuit using $GF(((2^2)^2)^2)$ tower field representation in normal basis, that is both smaller *and* faster than Canright design in [3], [4]. In order to achieve this result goal, we propose the following contributions:

- We consider using isomorphic mapping for the decryption path that is independent (different or the same) from the mapping for the encryption path. This results in 32 unique mapping circuits per field, whereas Canright design considered using the same mapping for both encryption and decryption paths resulting in only one mapping circuit per field. Since we have 16, all normal basis, field representations in the tower field $GF(((2^2)^2)^2)$, we study a total of $(16 \times 32 = 512)$ mapping circuits and use the most compact one (Sec. IV).
- For the new obtained tower field with the least complexities in its mappings, we combine several small blocks into one block, denoted as the exponentiation block, and propose new closed-form formulations that make use of optimum gate sharing. The exponentiation block is proposed in Sec. V.
- We propose new formulations for the subfield inversion and the output multipliers, resembling the most compact designs to date. This contribution is discussed in Sec. VI and Sec. VI.
- We code all the blocks in VHDL and provide their implementation results. Then, we compare the ASIC implementation results of our entire scheme with the one proposed by Canright in two CMOS technologies of STM 65nm and NanGate 15nm. Our complexity analysis and implementation results show that our design outperforms the counterparts. In addition, we validate each and every block in the design using Matlab® codes and HDL testbenches.

II. Preliminaries

A. S-box/inverse S-box Arithmetic Computations

The S-box computations are performed over the binary field $GF(2^8)$ generated by the irreducible polynomial $q(x) = x^8 + x^4 + x^3 + x + 1$. Let $\mathbf{g} = [g_7, \cdots, g_1, g_0]^{tr}$ and $\mathbf{s} = [s_7, \cdots, s_1, s_0]^{tr}$ be the vectors corresponding to the input and output of the S-box, respectively. Then, the S-box computation consists of computing of the multiplicative inverse of a non-zero input $g \in GF(2^8)$, denoted as $f = g^{-1} \in GF(2^8)$, $g \neq 0$ followed by the affine transformation, i.e.,

$$\mathbf{s} = \mathbf{M}\mathbf{f} \oplus \mathbf{h}, \tag{1}$$

where $\mathbf{f} = [f_7, \cdots, f_1, f_0]^{tr}$ is the vector corresponding to the field element $f \in GF(2^8)$ and $\oplus$ is the mod-2 addition (XOR). The inverse S-box reverses the S-box operation. From (1), the inverse S-box computes

$$\mathbf{f} = \mathbf{M}^{-1}(\mathbf{s} \oplus \mathbf{h}), \tag{2}$$

where

$$\mathbf{M} = \begin{bmatrix} 1\,1\,1\,1\,1\,0\,0\,0 \\ 0\,1\,1\,1\,1\,1\,0\,0 \\ 0\,0\,1\,1\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1\,1\,1 \\ 1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,1\,1 \\ 1\,1\,1\,1\,0\,0\,0\,1 \end{bmatrix}, \quad \mathbf{M}^{-1} = \begin{bmatrix} 0\,0\,1\,0\,0\,1\,0\,1 \\ 1\,0\,0\,1\,0\,0\,1\,0 \\ 0\,1\,0\,0\,1\,0\,0\,1 \\ 1\,0\,1\,0\,0\,1\,0\,0 \\ 0\,1\,0\,1\,0\,0\,1\,0 \\ 0\,0\,1\,0\,1\,0\,0\,1 \\ 1\,0\,0\,1\,0\,1\,0\,0 \\ 0\,1\,0\,0\,1\,0\,1\,0 \end{bmatrix},$$

and $\mathbf{h} = [01100011]^{tr}$. Then, the inverse S-box consists of adding the constant $\mathbf{h}$, followed by applying the inverse affine transformation $\mathbf{M}^{-1}$ and then the multiplicative inversion to find the inverse S-box output $g = f^{-1}$, $f \neq 0$.

B. Tower Field $GF(((2^2)^2)^2)$ Construction

There are two different sets of field architectures, namely the tower field $GF(((2^2)^2)^2)$ and the composite field $GF((2^4)^2)$ that can be used for the AES S-box computations. Also, the subfield elements can be represented in polynomial basis (PB) or normal basis (NB). In this paper, we use the tower field $GF(((2^2)^2)^2)$ over NB, similar to the Canright's scheme, but using different irreducible polynomials.

In our scheme, we convert a field element $g = (g_7, ..., g_0) \in GF(2^8)$ to an isomorphic tower field $GF(((2^2)^2)^2)$ defined using the irreducible polynomial over $GF((2^2)^2)$:

$$p(y) = y^2 + y + \nu = (y + \gamma)(y + \gamma^{16}), \tag{3}$$

where γ (its root) and ν are subfield elements in $GF((2^2)^2)$ that should be chosen so that this polynomial is irreducible over $GF((2^2)^2)$. Then, $\{\gamma, \gamma^{16}\}$ is the NB over $GF((2^2)^2)$ and every element g in $GF(2^8)$ can be mapped to its tower field representation over $GF((2^2)^2)$ as $g = A\gamma + B\gamma^{16}$, where A and B are subfield elements in $GF((2^2)^2)$.

Similarly, the subfield $GF((2^2)^2)$ is generated using the irreducible polynomial over $GF(2^2)$ of

$$q(z) = z^2 + z + \eta = (z + \alpha)(z + \alpha^4), \tag{4}$$

with its root α that generates the NB $\{\alpha, \alpha^4\}$ over $GF(2^2)$. In (4), $\eta \in GF(2^2)$ should be selected so that $q(z)$ is irreducible over $GF(2^2)$. Therefore, any subfield element $A \in GF((2^2)^2)$ can be represented with respect to the NB $\{\alpha, \alpha^4\}$ by $A = A_0\alpha + A_1\alpha^4$ where $A_0, A_1 \in GF(2^2)$. In parts of the circuit proposed in this paper, we use the redundant normal basis (RNB) $\{\alpha, \alpha^4, 1\}$ to represent field elements over $GF(2^2)$, where $\alpha + \alpha^4 = 1 \in GF(2^2)$. Here, we use the hat notation to represent the coordinates with respect to the RNB. Therefore, the element A can also be represented by $A = \hat{A}_0\alpha + \hat{A}_1\alpha^4 + \hat{A}_2$, where $\hat{A}_i \in GF(2^2)$, and $A_i = \hat{A}_i + \hat{A}_2$ for $i = 0, 1$ as $\alpha + \alpha^4 = 1$.

To construct the binary field $GF(2^2)$, the irreducible all-one-polynomial (AOP) with degree 2, i.e.,

$$r(t) = t^2 + t + 1 = (t + \omega)(t + \omega^2), \tag{5}$$

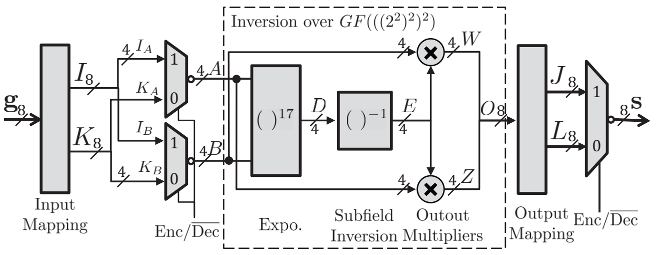


Fig. 1. The overall architecture for the proposed combined S-box/inverse S-box using $GF(((2^2)^2)^2)$ in the NB representation.

is used to generate the NB $\{\omega, \omega^2\}$ over $GF(2)$. Therefore, one can represent the field elements $A_0, A_1 \in GF(2^2)$ as $A_0 = a_0\omega + a_1\omega^2$ and $A_1 = a_2\omega + a_3\omega^2$, respectively. As a result, any subfield element $A = (a_0a_1a_2a_3) \in GF((2^2)^2)$ can be represented as

$$A = (a_0\omega + a_1\omega^2)\alpha + (a_2\omega + a_3\omega^2)\alpha^4, \tag{6}$$

where a_i are the binary coordinates. It is noted that we also use the RNB $\{\omega, \omega^2, 1\}$ to represent an element over $GF(2^2)$, where $\omega + \omega^2 = 1 \in GF(2)$. Similarly, we use the hat notation for the coordinates with respect to the RNB. Therefore, A_0 can also be represented as $A_0 = \hat{a}_0\omega + \hat{a}_1\omega^2 + \hat{a}_2$ where $\hat{a}_i \in GF(2)$, $i \in [0, 2]$ are the coordinates of A_0 with respect to the RNB. One can find $a_i = \hat{a}_i \oplus \hat{a}_2$ for $i = 0, 1$ as $\omega + \omega^2 = 1$.

C. Inversion over Tower Field

The multiplicative inverse of $g = A\gamma + B\gamma^{16}$ in the tower field $GF(((2^2)^2)^2)$ can be written as $g^{-1} = (g^{17})^{-1}g^{16}$ [24], [25]. Let us define $D = g^{17} = (A\gamma + B\gamma^{16})(B\gamma + A\gamma^{16})$, which can be simplified to

$$D = A \times B + (A + B)^2\nu. \tag{7}$$

Then, $g^{-1} = D^{-1}(B\gamma + A\gamma^{16})$. Assuming that $g^{-1} = W\gamma + Z\gamma^{16}$, where $W, Z \in GF((2^2)^2)$, the outputs of the inversion $g^{-1} \in GF(((2^2)^2)^2)$ can be found by computing

$$\begin{aligned} W &= B \times D^{-1} = B \times E \\ Z &= A \times D^{-1} = A \times E, \end{aligned} \tag{8}$$

where $E = D^{-1}$.

III. Proposed Architecture

In this section, we introduce the overall architecture for the combined proposed S-box/ inverse S-box. The proposed architecture is shown in Figure 1. In the beginning, the input 8-bit g is processed through the input isomorphic mapping, which generates two 8-bit outputs, one for the encryption path, denoted as I, and one for the decryption path, denoted as K. The input and output isomorphic mappings are introduced in Sec. IV. The input multiplexers select the inputs of the inversion circuit; A and B of 4-bit each. Here, we use inverting multiplexers because they are both cheaper and faster than the regular multiplexers.

Then, the inversion circuit is composed of three stages:

1) The exponentiation block (Sec. V) raises the input g to the power 17 in $GF(((2^2)^2)^2)$ to compute D in (7).
2) The subfield inversion block (Sec. VI) computes inversion over $GF((2^2)^2)$, to generate $E = D^{-1}$.
3) The output multipliers (Sec. VII) perform multiplication in $GF((2^2)^2)$, to compute the outputs W and Z in (8), while resembles $O = W\gamma + Z\gamma^{16}$.

Finally, the output isomorphic mapping converts the outputs (W and Z) into two corresponding outputs in the $GF(2^8)$ field, one for the encryption path, denoted as J, and one for the decryption path, denoted as L. The output inverting multiplexers select the overall output $s = (s_7, ..., s_0) \in GF(2^8)$.

In this paper, we adopt the $GF(((2^2)^2)^2)$ tower field with all NB representations for the underlying fields. To make $p(y)$ in (3) irreducible, the constant ν in (3) can take any of the following 8 different values: {[1000],[0100],[0010],[0001],[1110],[1101],[1011],[0111]}. Here, The 4-bits of each value represent the coefficients of $\nu_i, i \in [0,3]$ in $\nu = (\nu_0\omega+\nu_1\omega^2)\alpha+(\nu_2\omega+\nu_3\omega^2)\alpha^4$. Similarly, the constant η in (4) can take any of the following 2 values: {[10],[01]}, where the 2-bits represent the coefficients of $\eta = \eta_0\omega + \eta_1\omega^2$. In total, we explore $8 \times 2 = 16$ different field representations.

IV. Isomorphic Mappings

In this section, we explore all the possible isomorphic mappings that could be used to select the most compact ones. Isomorphic mapping performs a one-to-one mapping between every element of one field to a corresponding element in an isomorphic field. This mapping preserves additive as well as multiplicative isomorphisms and is mathematically represented by a matrix multiplication, where all the elements of the matrix are ones or zeros.

The matrix for isomorphic mapping, denoted as $\mathbf{X}^{-1}$, can be found by solving the following equations:

$$\phi_2^i = \mathbf{X}^{-1} \times \phi_1^i, i \in [0, n-2], \tag{9}$$

where ϕ_1 is a generator in the $GF(2^8)$ AES field, ϕ_2 is a generator in the new $GF(((2^2)^2)^2)$ field, and n is the size of the field (here, $n = 256$). Note that in order to preserve multiplicative isomorphism, the selected generator ϕ_2 must also be a root of the AES $GF(2^8)$ irreducible polynomial as evaluated under the $GF(((2^2)^2)^2)$ arithmetic. Once such a generator is found (ϕ_2), all its conjugate elements $\phi_2^{2^j}, j \in [1,7]$ can also be used to build different mappings, i.e., 8 possible mappings per field.

At the output of the S-box, we would apply the inverse mapping $\mathbf{X}$ so that: $\phi_1^i = \mathbf{X} \times \phi_2^i, i \in [0, n-2]$. Note that the $\mathbf{X}^{-1}$ and $\mathbf{X}$ mappings can be used at the input and output, respectively, of either the S-box, the inverse S-box, or both.

A. Asymmetric Isomorphic Mapping

In the proposed architecture (Figure 1), the affine transformation ($\mathbf{M}$) is multiplied by the isomorphic mappings ($\mathbf{X}_e$ or $\mathbf{X}_d$) at the output of the S-box as $\mathbf{MX}_e$ and the input of the inverse S-box as $(\mathbf{MX}_d)^{-1}$. Hence, the input isomorphic mapping will be represented by a 16×8 matrix denoted as $\mathbf{Tr}_{in}$:

$$\left[\begin{array}{c}\mathbf{I}\\ \hline \mathbf{K}\end{array}\right] = \mathbf{Tr}_{in} \times \mathbf{g} = \left[\begin{array}{c}\mathbf{X}_e^{-1}\\ \hline (\mathbf{MX}_d)^{-1}\end{array}\right] \times \mathbf{g} \tag{10}$$

Similarly, the output isomorphic mapping will be a 16×8 matrix denoted as $\mathbf{Tr}_{out}$:

$$\left[\begin{array}{c}\mathbf{J}\\ \hline \mathbf{L}\end{array}\right] = \mathbf{Tr}_{out} \times \mathbf{O} = \left[\begin{array}{c}\mathbf{MX}_e\\ \hline \mathbf{X}_d\end{array}\right] \times \mathbf{O} \tag{11}$$

It is noted in [3], the same mapping $\mathbf{X}$ is considered for both encryption and decryption, whereas we consider two independent mappings ($\mathbf{X}_e$ and $\mathbf{X}_d$) which could be different or the same. In [13], [14], the authors used asymmetric mapping to build two completely separate encryption and decryption engines over the same die, where no gate-sharing was used. However, using asymmetric mapping with gate sharing over a unified encryption/decryption data path is proposed here for the first time, to the best of our knowledge. Since we have 8 possible mappings per field in each data path, we can build a total of 64 different isomorphic mappings, each consisting of a set of $\mathbf{Tr}_{in}$ and a corresponding $\mathbf{Tr}_{out}$.

Note that, in the NB representation, if $\phi_2 = \phi_{2A}\gamma + \phi_{2B}\gamma^{16}$ is a generator, one of the conjugates generators would be: $\phi_2^{2^4} = \phi_{2B}\gamma + \phi_{2A}\gamma^{16}$. Hence, the isomorphic mappings that could be built using these two generators would be row-wise related in the input mapping $\mathbf{X}^{-1}$ and column-wise related in the output mapping $\mathbf{X}$. These two mappings are considered equivalent and can be realized using the same circuit with only a change in the output notations. In general, the mappings of $\phi_2^{2^i}, i \in [4,7]$ will be equivalent to the mappings of $\phi_2^{2^i}, i \in [0,3]$, respectively. Therefore, despite having 64 different isomorphic mappings, we end up having only 32 unique circuits. As a result, in the next subsection, we study a total of 16 (field representations) × 32 (mappings per field) = 512 unique isomorphic mapping circuits.

B. Logic-Minimization Algorithms

Efficient implementation of mapping matrices is performed using logic-minimization algorithms. The problem of finding the most compact circuit to implement a matrix of isomorphic mapping is called the Shortest Linear Program (SLP) problem and is known to be NP-hard [15], [17]. The optimum solution can only be found using exhaustive search. Indeed, Canright used exhaustive search in order to find the most compact circuit [4]. However, it was shown by Boyar et al. that the search conducted by Canright is cancellation-free [16], [17], where XOR gates are never used to cancel out common terms, i.e., Canright did not exhaust all the actually possible solutions. In this paper, we use the Focused-Search logic-minimization algorithm proposed in [8], which is a speed-optimized variant of exhaustive search and is not cancellation-free. Despite having a relatively long execution time, it is shown in [8] that Focused-Search is the currently best logic-minimization algorithm.

C. The Proposed Mappings

After searching across all the 512 mappings, we propose to use mappings that are obtained based on the tower field with $\nu = [0010] = \omega\alpha^4$ and $\eta = [01] = \omega^2$. It is noted that we use a tower field that is different from the one used by Canright [3]. Using (10) and (11), the proposed input and output transformation matrices, $\mathbf{Tr}_{in}$ and $\mathbf{Tr}_{out}$, are shown in Table I.

The exact formulations to implement $\mathbf{Tr}_{in}$ and $\mathbf{Tr}_{out}$ are shown in Table II. Note that these equations reflect the use of inverting multiplexer at the input and output of the combined S-box/inverse S-box because they require smaller area and lower delay than the regular multiplexers. Also, some XOR gates (denoted with $\oplus$) were replaced with XNOR gates (denoted with $\odot$) and NOT gates (denoted with $(\ )'$) were added to incorporate the addition with the constant $\mathbf{h}$ at the output of the encryption path and the input of the decryption path, as required by (1) and (2). The delay of each signal is also highlighted in the table. We provide the complexities of the mapping blocks as follows.

Proposition 1. *The input mapping $\mathbf{Tr}_{in}$ can be implemented using 19 gates (12 XOR2, 6 XNOR2, and 1 NOT), with a maximum delay of $5D_X$, where D_X is the delay of a 2-bit XOR2/XNOR2 gate. The output mapping $\mathbf{Tr}_{out}$ can be implemented using 18 gates (4 XOR2, 13 XNOR2, 1 NOT), with a maximum delay of $4D_X$.*

TABLE I
THE PROPOSED INPUT AND OUTPUT ISOMORPHIC MAPPINGS.

$$\mathbf{Tr}_{in} = \left[\begin{array}{c} 0\,1\,1\,1\,0\,0\,0\,1 \\ 1\,1\,1\,0\,0\,1\,1\,1 \\ 1\,1\,1\,0\,0\,0\,0\,1 \\ 0\,1\,1\,0\,0\,0\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,1 \\ 1\,0\,0\,1\,1\,0\,1\,1 \\ 0\,1\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,0\,0\,1 \\ \hline 0\,1\,0\,1\,0\,0\,1\,1 \\ 1\,0\,0\,1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,1\,0\,1\,1 \\ 0\,1\,0\,1\,0\,0\,0\,0 \\ 1\,0\,1\,0\,0\,1\,0\,0 \\ 1\,1\,0\,1\,0\,0\,0\,0 \\ 0\,1\,1\,1\,0\,0\,1\,1 \\ 0\,0\,0\,1\,1\,0\,0\,1 \end{array}\right], \quad \mathbf{Tr}_{out} = \left[\begin{array}{c} 0\,0\,0\,1\,0\,1\,0\,0 \\ 0\,1\,0\,0\,0\,1\,0\,0 \\ 1\,0\,0\,0\,0\,0\,1\,0 \\ 0\,1\,0\,1\,0\,1\,0\,0 \\ 1\,1\,1\,1\,0\,1\,0\,0 \\ 1\,0\,0\,1\,1\,1\,1\,0 \\ 0\,0\,1\,1\,0\,0\,0\,1 \\ 1\,0\,1\,0\,0\,0\,0\,1 \\ \hline 0\,0\,1\,0\,0\,0\,0\,1 \\ 1\,1\,0\,1\,0\,1\,1\,1 \\ 1\,1\,0\,1\,1\,1\,1\,0 \\ 1\,0\,0\,0\,0\,0\,0\,1 \\ 1\,0\,1\,1\,1\,1\,0\,1 \\ 0\,1\,1\,1\,0\,0\,0\,1 \\ 0\,0\,0\,1\,0\,0\,0\,1 \\ 0\,0\,0\,0\,1\,0\,0\,0 \end{array}\right].$$

TABLE II
EQUATIONS USED TO IMPLEMENT THE PROPOSED ISOMORPHIC MAPPINGS: (a) THE PROPOSED $\mathbf{Tr}_{in}$, (b) THE PROPOSED $\mathbf{Tr}_{out}$.

(a)

$i_{a0} = i_{b3} \oplus g_4$	$3D_X$	$i_{a1} = t_0 \oplus i_{a3}$	$4D_X$	$i_{a2} = i_{b3} \oplus g_7$	$3D_X$
$i_{a3} = i_{b3} \oplus g_1$	$3D_X$	$i_{b0} = (g_0)'$	$0D_X$	$i_{b1} = k_{b1} \odot k_{a2}$	$4D_X$
$i_{b2} = k_{a2} \odot g_2$	$4D_X$	$i_{b3} = t_1 \odot g_5$	$2D_X$	$k_{a0} = k_{b2} \odot g_5$	$5D_X$
$k_{a1} = g_7 \odot g_4$	$1D_X$	$k_{a2} = t_2 \oplus g_1$	$3D_X$	$k_{a3} = g_6 \oplus g_4$	$1D_X$
$k_{b0} = t_0 \oplus g_5$	$2D_X$	$k_{b1} = k_{a1} \odot g_6$	$2D_X$	$k_{b2} = i_{a0} \oplus g_1$	$4D_X$
$k_{b3} = k_{a3} \oplus t_2$	$3D_X$	$t_0 = g_7 \oplus g_2$	$1D_X$	$t_1 = g_6 \oplus g_0$	$1D_X$
$t_2 = t_1 \oplus g_3$	$2D_X$				

(b)

$j_7 = w_3 \odot z_1$	$1D_X$	$j_6 = w_1 \oplus z_1$	$1D_X$	$j_5 = w_0 \oplus z_2$	$1D_X$
$j_4 = j_7 \oplus w_1$	$2D_X$	$j_3 = j_0 \odot tt_1$	$4D_X$	$j_2 = j_5 \odot tt_0$	$3D_X$
$j_1 = l_7 \odot w_3$	$2D_X$	$j_0 = l_7 \odot w_0$	$2D_X$	$l_7 = w_2 \odot z_3$	$1D_X$
$l_6 = j_5 \odot tt_1$	$4D_X$	$l_5 = j_2 \oplus w_1$	$4D_X$	$l_4 = w_0 \odot z_3$	$1D_X$
$l_3 = j_0 \odot tt_0$	$3D_X$	$l_2 = j_1 \odot w_1$	$3D_X$	$l_1 = w_3 \odot z_3$	$1D_X$
$l_0 = (z_0)'$	$0D_X$	$tt_0 = j_7 \odot z_0$	$2D_X$	$tt_1 = j_4 \odot z_3$	$3D_X$

V. NEW EXPONENTIATION BLOCK

The exponentiation block in Figure 1 generates $D = g^{17}$ as required in (7). Since we selected $\nu = \omega\alpha^4$, (7) can be written as $D = A \times B + (A+B)^2\omega\alpha^4$ which requires multiplication, squaring and scaling over $GF((2^2)^2)$. In this section, we analyze these operations to find the coordinates of $D = (d_0d_1d_2d_3) = (d_0\omega + d_1\omega^2)\alpha + (d_2\omega + d_3\omega^2)\alpha^4$ for the new tower field.

A. Multiplication over $GF((2^2)^2)$

Let $A = A_0\alpha + A_1\alpha^4$ and $B = B_0\alpha + B_1\alpha^4$ be subfield elements over $GF((2^2)^2)$ where $A_i, B_i \in GF(2^2)$, $i = 0, 1$. Also, lets use the RNB representation to define the output of their multiplication as $C = A \times B = \hat{C}_0\alpha + \hat{C}_1\alpha^4 + \hat{C}_2$, where $\hat{C}_0, \hat{C}_1, \hat{C}_2 \in GF(2^2)$ can be computed as

$$\begin{aligned} \hat{C}_0 &= A_0B_0, \\ \hat{C}_1 &= A_1B_1, \\ \hat{C}_2 &= (A_0 + A_1)(B_0 + B_1)\omega^2. \end{aligned} \tag{12}$$

Here, we used $\alpha\alpha^4 = \eta$ as implied from (4), and $\eta = \omega^2$ following the mapping selection in Sec. IV-C. As seen from (12), multiplication in $GF((2^2)^2)$ requires three multiplications over $GF(2^2)$. Since the polynomial used to define the $GF(2^2)$ field in (5) is an AOP with degree 2 ($m = 2$) and is irreducible, we use the type-I optimal normal basis (ONB-I) multiplication scheme proposed in [26] for $m = 2$. Specifically, we use equation (36) to represent the multiplication operation as follows.

Lemma 1. *From [26], let $A_0 = a_0\omega + a_1\omega^2 \in GF(2^2)$ and $B_0 = b_0\omega + b_1\omega^2 \in GF(2^2)$ be represented in the ONB-I $\{\omega, \omega^2\}$. Then, the coordinates of their multiplication, represented in the redundant normal basis (RNB) $\{\omega, \omega^2, 1\}$, can be calculated as follows: $\hat{C}_0 = A_0B_0 = a_0b_0\omega + a_1b_1\omega^2 + a_{01}b_{01}$ where $a_{01} = a_0 \oplus a_1$ and $b_{01} = b_0 \oplus b_1$.*

Since $A_1 = a_2\omega + a_3\omega^2 \in GF(2^2)$ and $B_1 = b_2\omega + b_3\omega^2 \in GF(2^2)$, one can use Lemma 1 to find $(A_0 + A_1)(B_0 + B_1) = a_{02}b_{02}\omega + a_{13}b_{13}\omega^2 + a_pb_p$. As a result, (12) can be computed as

$$\begin{aligned} \hat{C}_0 &= a_0b_0\omega + a_1b_1\omega^2 + a_{01}b_{01} \\ \hat{C}_1 &= a_2b_2\omega + a_3b_3\omega^2 + a_{23}b_{23} \\ \hat{C}_2 &= a_{13}b_{13}\omega + a_pb_p\omega^2 + a_{02}b_{02}, \end{aligned} \tag{13}$$

where $a_{jk} = a_j \oplus a_k$ and $b_{jk} = b_j \oplus b_k$ for $0 \le j, k \le 3$, and $j \neq k$. Also, $a_p = a_{02} \oplus a_{13}$ and $b_p = b_{02} \oplus b_{13}$ which are the parities of A and B, respectively.

B. Squaring with Scaling

Lets denote $V = (A+B)^2\omega\alpha^4 = V_0\alpha + V_1\alpha^4$, with $V_0, V_1 \in GF((2^2)^2)$. Using (12), one can find $A^2 = A_0^2\alpha + A_1^2\alpha^4 + (A_0 + A_1)^2\omega^2$ and similar expression for B^2. Since $(A+B)^2 = A^2 + B^2$, we can simplify computation of the coefficients of V to:

$$\begin{aligned} V_0 &= (A_0 + A_1 + B_0 + B_1)^2, \\ V_1 &= (A_1 + B_1)^2\omega. \end{aligned} \tag{14}$$

Representing the equations in $GF(2^2)$, similar to (13), one can find that

$$\begin{aligned} V_0 &= (a_{13} \oplus b_{13})\omega + (a_{02} \oplus b_{02})\omega^2, \\ V_1 &= (a_2 \oplus b_2) + (a_3 \oplus b_3)\omega^2, \end{aligned} \tag{15}$$

where $a_{13} = a_1 \oplus a_3$, $a_{02} = a_0 \oplus a_2$ and $b_{13} = b_1 \oplus b_3$, $b_{02} = b_0 \oplus b_2$.

C. New Exponentiation Computation

The output of exponentiation block, i.e., D, can be computed by adding $C = \hat{C}_0\alpha + \hat{C}_1\alpha^4 + \hat{C}_2$ with $V = V_0\alpha + V_1\alpha^4$. Then, one can obtain the coefficients of D with respect to the RNB as $D = \hat{D}_0\alpha + \hat{D}_1\alpha^4 + \hat{D}_2$, where

$$\begin{aligned} \hat{D}_0 &= \hat{C}_0 + V_0 \\ \hat{D}_1 &= \hat{C}_1 + V_1 \\ \hat{D}_2 &= \hat{C}_2. \end{aligned} \tag{16}$$

Using (13) and (15), we can simplify the RNB coefficients of D to:

$$\begin{aligned} &\hat{D}_0 = (a_0b_0 \oplus a_{13} \oplus b_{13})\omega + (a_{02} \oplus b_{02} \oplus a_1b_1)\omega^2 + a_{01}b_{01} \\ &\hat{D}_1 = a_2b_2\omega + (a_3b_3 \oplus a_3 \oplus b_3)\omega^2 + (a_{23}b_{23} \oplus a_2 \oplus b_2) \\ &\hat{D}_2 = a_{13}b_{13}\omega + a_pb_p\omega^2 + a_{02}b_{02}. \end{aligned} \tag{17}$$

We can represent D in the NB as $D = D_0\alpha + D_1\alpha^4$, with $D_i = \hat{D}_i + \hat{D}_2$ for $i = 0, 1$. Then, the $GF(2^2)$ coefficients of $D = (d_0\omega + d_1\omega^2)\alpha + (d_2\omega + d_3\omega^2)\alpha^4$ can be found as:

$$\begin{aligned} &d_0 = (a_{01}b_{01} \oplus a_{02}b_{02} \oplus a_0b_0 \oplus (a_{13} \vee b_{13})), \\ &d_1 = (a_{01}b_{01} \oplus (a_{02} \vee b_{02}) \oplus a_1b_1 \oplus a_pb_p), \\ &d_2 = ((a_2 \vee b_2) \oplus a_{13}b_{13} \oplus a_{23}b_{23} \oplus a_{02}b_{02}), \\ &d_3 = (a_3b_3 \oplus a_pb_p \oplus (a_{23} \vee b_{23}) \oplus a_{02}b_{02}). \end{aligned} \tag{18}$$

Here, we use $a_2 \oplus b_2 \oplus a_2b_2 = a_2 \vee b_2$ to obtain d_2. For the computation in d_3, we first replace $a_2 \oplus a_3$ and $a_3 \oplus b_3$ with a_{23} and

b_{23}, respectively and then use them to simplify $a_{23} \oplus b_{23} \oplus a_{23}b_{23}$ as $a_{23} \vee b_{23}$. Also, for low cost implementation, we replace all AND and OR operations in (18) to NAND and NOR operations, respectively, without changing their functions.

Proposition 2. *The exponentiation computation block, with original formulations presented in (18), consists of 12+10=22 XOR2 (2-input XOR), 8 NAND2 (2-input NAND), and 4 NOR2 (2-input NOR) gates with the critical path delay of $4D_X + D_{NAND}$ where D_X and D_{NAND} are the delays of one XOR2 gate and one NAND2 gate, respectively.*

VI. New Subfield Inversions over $GF((2^2)^2)$

Let $D = (d_0d_1d_2d_3)$ be the input of the subfield inverter. In this section, we derive the formulations for the output of the subfield inverter, i.e., $E = D^{-1} = (e_0e_1e_2e_3) = E_0\alpha + E_1\alpha^4 \in GF((2^2)^2)$, where $E_0 = e_0\omega + e_1\omega^2$, $E_1 = e_2\omega + e_3\omega^2$ and $e_i \in GF(2)$, $0 \le i \le 3$, are its coordinates. In the following, we introduce two methods to calculate the coordinates of E.

A. Using Computations over $GF(2^2)$

To find the formulations for E, one can use an algebraic expression of $E = D^{-1} = (D^r)^{-1} \times D^{r-1}$, where $r = \frac{2^{2\times2}-1}{2^2-1} = 5$ with arithmetic operations performed over $GF(2^2)$. Hence, $E = (DD^4)^{-1} \times D^4$. Assuming that $U = (DD^4)^{-1} \in GF(2^2)$, then $E = UD_1\alpha + UD_0\alpha^4 = E_0\alpha + E_1\alpha^4$. Since the inversion over $GF(2^2)$ is equal to squaring, one can obtain that $U = (D_0D_1 + (D_0 + D_1)^2\omega^2)^2$. Representing U in the RNB, i.e., $U = \hat{u}_0\omega + \hat{u}_1\omega^2 + \hat{u}_2$, and using reduction techniques similar to the one presented in Sec. V, one can find that:

$$\begin{aligned} \hat{u}_0 &= d_1d_3 \\ \hat{u}_1 &= (d_0 \vee d_2) \\ \hat{u}_2 &= (d_{01}d_{23} \oplus d_{13}), \end{aligned} \tag{19}$$

where we use $d_0d_2 \oplus d_{02} = d_0 \vee d_2$. Converting U back to the NB results in $U = u_0\omega + u_1\omega^2$ with coefficients:

$$\begin{aligned} u_0 &= d_{01}d_{23} \oplus (d_1 \vee d_3) \\ u_1 &= d_{01}d_{23} \oplus d_{13} \oplus (d_0 \vee d_2). \end{aligned} \tag{20}$$

Using (20) and Lemma 1, one can find $E_0 = UD_1 = \hat{e}_0\omega + \hat{e}_1\omega^2 + \hat{e}_2$ with coefficients equal to:

$$\begin{aligned} \hat{e}_0 &= d_2(d_{01}d_{23} \oplus (d_1 \vee d_3)), \\ \hat{e}_1 &= d_3(d_{01}d_{23} \oplus d_{13} \oplus (d_0 \vee d_2)), \\ \hat{e}_2 &= u_{01}d_{23}, \end{aligned} \tag{21}$$

where $u_{01} = u_0 \oplus u_1 = \hat{u}_0 \oplus \hat{u}_1 = d_1d_3 \oplus (d_0 \vee d_2)$ and $d_{23} = d_2 \oplus d_3$. Then, the coordinates of $E_0 = e_0\omega + e_1\omega^2$ represented in the NB can be found as:

$$e_0 = \hat{e}_0 \oplus \hat{e}_2 \quad \text{and} \quad e_1 = \hat{e}_1 \oplus \hat{e}_2. \tag{22}$$

Similarly, the coordinates of $E_1 = UD_0 = \hat{e}_3\omega + \hat{e}_4\omega^2 + \hat{e}_5$ are calculated as:

$$\begin{aligned} \hat{e}_3 &= d_0(d_{01}d_{23} \oplus (d_1 \vee d_3)), \\ \hat{e}_4 &= d_1(d_{01}d_{23} \oplus d_{13} \oplus (d_0 \vee d_2)), \\ \hat{e}_5 &= u_{01}d_{01}, \end{aligned} \tag{23}$$

and the coordinates of $E_1 = e_2\omega + e_3\omega^2$ represented in NB can be found as:

$$e_2 = \hat{e}_3 \oplus \hat{e}_5 \quad \text{and} \quad e_3 = \hat{e}_4 \oplus \hat{e}_5. \tag{24}$$

TABLE III
The truth table of the inverter over $GF((2^2)^2)$.

$d_0d_1d_2d_3$	$e_0e_1e_2e_3$	$d_0d_1d_2d_3$	$e_0e_1e_2e_3$
0000	0000	1000	0011
0001	0100	1001	1011
0010	1100	1010	0101
0011	1000	1011	1001
0100	0001	1100	0010
0101	1010	1101	0111
0110	1110	1110	0110
0111	1101	1111	1111

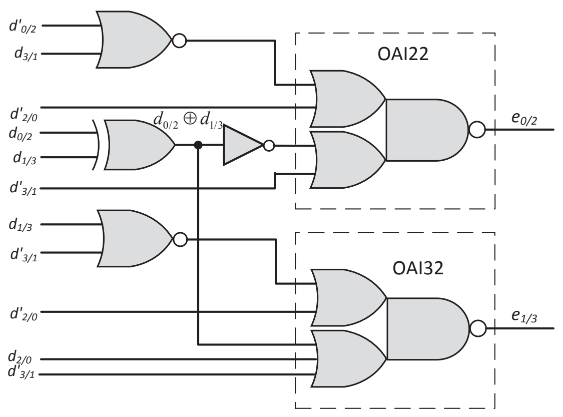


Fig. 2. The proposed subfield inverter block.

B. Using Combinational Circuit Design

The subfield inverter is a 4-bit input 4-bit output combinational circuit which can be designed using the systematic way of designing digital systems. The design starts with the truth table of the inverter. We used Matlab® to find the inversion truth table as shown in Table III. Using this table one can obtain the following.

Lemma 2. *Let $E = D^{-1}$ represented by $E = (e_0e_1e_2e_3) = E_0\alpha + E_1\alpha^4 \in GF((2^2)^2)$, where $E_0 = e_0\omega + e_1\omega^2$ and $E_1 = e_2\omega + e_3\omega^2$. Then, the coordinates of E_0 and E_1 are*

$$\begin{aligned} e_0 &= d_3(d_0 \oplus d_1) \vee d_2(d_0' \vee d_3) \\ e_1 &= d_2'd_3(d_0 \odot d_1) \vee d_2(d_1 \vee d_3'), \end{aligned} \tag{25}$$

and

$$\begin{aligned} e_2 &= d_1(d_2 \oplus d_3) \vee d_0(d_2' \vee d_1) \\ e_3 &= d_0'd_1(d_2 \odot d_3) \vee d_0(d_3 \vee d_1'), \end{aligned} \tag{26}$$

respectively.

We designed and implemented the above formulations and several other equivalent functions to obtain the optimum design with the least area and delay in the ASIC implementation. In order to reduce the area and improve the speed of the ASIC implementations of (25) and (26), we found that their implementations using compound OR-AND-Invert (OAI) gates, such as OAI22 and OAI32 gates, have better implementation results instead of using AND/OR as well as NAND/NOR gates. One can easily convert the formulations in Lemma 2 to the following using Boolean algebra and De Morgan's laws. As a result, we conclude the following with its realization as shown in Figure 2.

Corollary 1. *The coordinates of E_0 can be found from*

$$\begin{aligned} e_0 &= ((d_3' \vee (d_0 \oplus d_1)')(d_2' \vee (d_0' \vee d_3)'))' \\ e_1 &= ((d_2 \vee d_3' \vee (d_0 \oplus d_1))(d_2' \vee (d_1 \vee d_3')'))'. \end{aligned} \quad (27)$$

Similarly, the coordinates of E_1 can be found by switching all indices in (27) between 0 and 2, i.e., $0 \leftrightarrow 2$ and between 1 and 3, i.e., $1 \leftrightarrow 3$.

Using 4 NOT gates at the inputs of Figure 2, one can obtain the following regarding the time and space complexities of this subfield inverter.

Proposition 3. *The space complexity of the proposed subfield inverter block over $GF((2^2)^2)$ includes 4 NOR2, 2 XOR2, 2 OAI22, 2 OAI32 and 6 NOT gates. The time complexity due to gates for the proposed subfield inverter is $1D_{OAI22} + 1D_{XOR2} + 1D_{NOT}$.*

VII. New Output Multipliers

One can use the formulations presented in Subsection V-A to obtain the formulations for two output multipliers that generate $Z = A \times E$ and $W = B \times E$. In this section, we further optimize these formulations to make the output multipliers faster than the one using these formulations directly. Starting with the formulations for $Z = A \times E$, let A be represented as in (6) and similarly represent E as $E = (e_0\omega + e_1\omega^2)\alpha + (e_2\omega + e_3\omega^2)\alpha^4 \in GF((2^2)^2)$, where E is the field element generated by the subfield inverter, with coordinates of $e_i \in GF(2)$, $i \in [0, 3]$. Let us represent $Z = A \times E$ with respect to the RNB as $Z = A \times E = \hat{Z}_0\alpha + \hat{Z}_1\alpha^4 + \hat{Z}_2$. Then, using (13), one can obtain

$$\begin{aligned} \hat{Z}_0 &= a_0e_0\omega + a_1e_1\omega^2 + a_{01}e_{01} \\ \hat{Z}_1 &= a_2e_2\omega + a_3e_3\omega^2 + a_{23}e_{23}, \\ \hat{Z}_2 &= a_{13}e_{13}\omega + a_pe_p\omega^2 + a_{02}e_{02}. \end{aligned} \quad (28)$$

To implement (28), the a_{ij} and a_p signals are available from the implementation of exponentiation block. However, 5 additional XOR gates are required for the implementation of the e_{ij} and e_p signals used in (28). To reduce the critical path delay (CPD), we convert the representations presented in (28) from the RNB to the NB. Using $a_{01}e_{01} = a_{01}e_0 \oplus a_{01}e_1$, we can simplify $\hat{Z}_0$ to

$$\hat{Z}_0 = (a_1e_0 \oplus a_{01}e_1)\omega + (a_0e_1 \oplus a_{01}e_0)\omega^2. \quad (29)$$

Similarly, using $a_{23}e_{23} = a_{23}e_2 \oplus a_{23}e_3$, $\hat{Z}_1$ can be simplified to

$$\hat{Z}_1 = (a_3e_2 \oplus a_{23}e_3)\omega + (a_2e_3 \oplus a_{23}e_2)\omega^2. \quad (30)$$

The computation of e_p is in the longest path which reduces the speed of the output multiplier. To design a fast multiplier, we use $e_p = e_{02} \oplus e_{13}$ in the expression of $\hat{Z}_2$ in (28), so that:

$$\hat{Z}_2 = (a_{13}e_{13} \oplus a_{02}e_{02})\omega + (a_pe_{13} \oplus a_{13}e_{02})\omega^2. \quad (31)$$

One can write $Z = Z_0\alpha + Z_1\alpha^4 = (z_0\omega + z_1\omega^2)\alpha + (z_2\omega + z_3\omega^2)\alpha^4$, with $Z_i = \hat{Z}_i + \hat{Z}_2$ for $i = 0, 1$ and so the coordinates of Z are as follows:

$$\begin{aligned} z_0 &= a_1e_0 \oplus a_{01}e_1 \oplus z_4 \\ z_1 &= a_0e_1 \oplus a_{01}e_0 \oplus z_5 \\ z_2 &= a_3e_2 \oplus a_{23}e_3 \oplus z_4 \\ z_3 &= a_2e_3 \oplus a_{23}e_2 \oplus z_5, \end{aligned} \quad (32)$$

where

$$\begin{aligned} z_4 &= a_{13}e_{13} \oplus a_{02}e_{02} \\ z_5 &= a_pe_{13} \oplus a_{13}e_{02}. \end{aligned} \quad (33)$$

Similarly, one can optimize the formulations for $W = B \times E$ by replacing the coordinates of A by the ones of B to obtain the coordinates of $W = (w_0\omega + w_1\omega^2)\alpha + (w_2\omega + w_3\omega^2)\alpha^4$ as:

$$\begin{aligned} w_0 &= b_1e_0 \oplus b_{01}e_1 \oplus w_4 \\ w_1 &= b_0e_1 \oplus b_{01}e_0 \oplus w_5 \\ w_2 &= b_3e_2 \oplus b_{23}e_3 \oplus w_4 \\ w_3 &= b_2e_3 \oplus b_{23}e_2 \oplus w_5, \end{aligned} \quad (34)$$

where

$$\begin{aligned} w_4 &= b_{13}e_{13} \oplus b_{02}e_{02} \\ w_5 &= b_pe_{13} \oplus b_{13}e_{02}. \end{aligned} \quad (35)$$

Note that the two signals $e_{02} = e_0 \oplus e_2$ and $e_{13} = e_1 \oplus e_3$ are shared among the two multipliers.

Instead of using AND gates, one can design the logical circuit of these multipliers using NAND gates by simply replacing all AND operations to NAND. Such replacements do not change the multiplication operation because the two inputs of XOR gates are complemented which result in no change at the output of XOR gates. It is interesting to note that using NAND gates is cheaper and faster in the ASIC implementation. As a result, one can obtain the space and time complexities of the two output multipliers as follows.

Proposition 4. *The output multipliers consist of 22 XOR2 and 24 NAND2 gates with the longest propagation delay of $D_{NAND} + 3D_X$.*

VIII. Implementation Results and Comparisons

In this section, we evaluate the implementation results of all the proposed blocks along with the overall combined S-box/inverse S-box circuit.

A. Complexity Analysis

TABLE IV
Complexity comparison of different combined S-box/inverse S-box designs.

Design	HW Complexity	GEs*
Zhang [27]	154X + 36AD + 16M	385
Jeon [19]	116X + 58AD + 2OR + 10NT + 16M	347
Ahmad [21]	123X + 35AD + 16M	321.75
Canright [3]	94X + 34ND + 6NR + 2NT + 16MI	257.5
This Work	81X + 32ND + 8NT + 8NR + 2O2 + 2O3 + 16MI	243.5

*All GE values are estimated using STM 65 technology where X is XOR2/XNOR2 =2GEs, AD is AND2 = 1.25GEs, ND is NAND2 = 1GE, OR is OR2 = 1.5GEs, NR isNOR2 = 1GE, NT is NOT = 0.75GEs, O2 is OAI22 = 1.75GEs, O3 is OAI32 = 2GEs,M is MUX21 = 2GEs, and MI is MUXI21 = 1.75GEs.

Table IV compares the hardware complexity analysis of the proposed combined S-box/inverse S-box against the schemes proposed in the contributions [27], [19], [21] and [3]. The estimate implementation areas (in GEs) are computed based on the area of individual gates in the STM 65nm technology library. [14] proposed the implementation area of the overall AES encryption and/or decryption engines, but no complexity analysis of the underlying combined S-box/inverse S-box was reported. However, from the information provided in the paper, we derived the complexities of the exponentiation block and output multipliers. Each multiplier in [14] requires 16 NAND2 gates and 15 XOR2 gates with no gate sharing. Therefore, the two output multipliers requires 30 XOR2 and 32 NAND2 gates ($30\times2+32 = 92$ GEs) which has higher area than our design, i.e., 22 XOR2 and 24 NAND2 ($22\times2+24 = 68$ GEs). Based on the parameters used in [14] ($\alpha = c$ and $\beta = 2$), the scaling with α requires 1 XOR2 gate and the combined squaring with scaling with β requires 3 XOR2 gates. Also,

TABLE V
ASIC SYNTHESIS RESULTS FOR THE THREE BLOCKS OF THE INVERSION AND THE ENTIRE INVERSION OVER THE TOWER FIELD AT STM 65NM TECHNOLOGY.

	Code #	Imp.	Ref	Area		CPD	Pow.
				μm^2	GE	ns	μW
Exp.	1	Str.	(18)	**116.48**	**56**	0.208	6.96
	2	Beh.		120.12	57.75	0.206	6.02
Sub. Inv.	3	Str.	(22),(24)	66.56	32	0.214	3.03
	4	Str.	Fig. 2	**41.6**	**20**	0.070	1.58
	5	Beh.	(27)	44.72	21.5	0.083	1.26
Out. Mul.	6	Str.	(32),(34)	**141.44**	**68**	0.171	6.74
	7	Beh.		143.52	69	0.192	5.67
Cascading the Three Blocks							
Tower Inv.	8	Str.	#1,4,6	**299.52**	**144**	0.607	25.02
	9	Beh.	#2,5,7	309.4	148.75	0.762	23.22

adding the complexity of the two subfield adders ($2 \times 4 = 8$ XOR2) and one multiplier (16 NAND2 gates and 15 XOR2), the overall exponentiation block requires 27 XOR2 and 16 NAND2 (= 70 GEs) which has more area than ours, i.e., 22 XOR2 + 8 NAND2 + 4 NOR2 (= 56 GEs). Table IV shows that Canright combined S-box/inverse S-box is the smallest design among the previous work.

B. ASIC Implementation Results

We use VHDL coding as a design entry to the Synopsys Design Vision® for logic synthesis. All the individual blocks and the combined S-box/inverse S-box, as proposed here and the previous work of [3], are evaluated using the STM 65nm and the NanGate 15nm CMOS standard-cell libraries. Note that results are collected at conservative wire load models and the critical path delays (CPDs) are reported by the CAD tool when there is no load to the external output.

1) Individual Blocks: We code all blocks of the proposed combined S-box/inverse S-box in two different modeling methods using VHDL. The first type of modeling is the structural modeling where we code all the blocks exactly as presented in equations and/or figures, with no optimization in the gate selection by the CAD tool. Then, we use behavioral modeling by defining the input/output relationship of each block and allowing optimization by the CAD tool to select the most compact circuit. In Table V, we compare the results of structural modeling against behavioral modeling for each block of the proposed combined S-box/inverse S-box. In this table, the power consumptions are included as reported by the CAD tool at relaxed constraints using a clock frequency of 100 MHz. Note that the maximum clock frequency can be obtained from the CPD. Later, we evaluate the combined S-box/inverse S-box design under more tight constraints.

Table V lists synthesis results for the proposed exponentiation block (Sec. V), the proposed subfield inversion block (Sec. VI-A and Sec. VI-B) and the output multipliers (Sec. VII) using both structural modeling and behavioral modeling. The table shows that structural modeling of all the blocks results in a more compact design than behavioral modeling. As a result, we propose the overall $GF(((2^2)^2)^2)$ tower field inversion consisting of Codes #1, 4 and 6 for the exponentiation, subfield inversion and output multipliers blocks, respectively.

2) Overall Design: Due to code availability, in this section, we compare the actual ASIC implementations results of the proposed combined S-box/inverse S-box against the most compact previous work, as proposed in [3]. Table VI highlights the reported results. It is noted that when we compiled the Canright original code, the tool used 16 non-inverting MUX cells which have more area than the inverting ones (2×16 GEs as compared to 1.75×16 GEs in the STM 65nm). To have a fair comparison, we changed this code to structural modeling and used 16 AOI22 (1.75×16 GEs) and one NOT gate (0.75 GEs) for their selectors. As a result, we added one NOT gate to the design of Canright as well as our proposed design. Table VI also shows the CPD as reported by the CAD tool. As shown in Table VI, our proposed architecture of the combined S-box/inverse S-box is, not only more compact than the currently best design in the literature (in [3], [4]), but also faster and requires lower power consumption.

TABLE VI
ASIC SYNTHESIS RESULTS FOR THE COMBINED S-BOX/INVERSE S-BOX AT STM 65NM TECHNOLOGY.

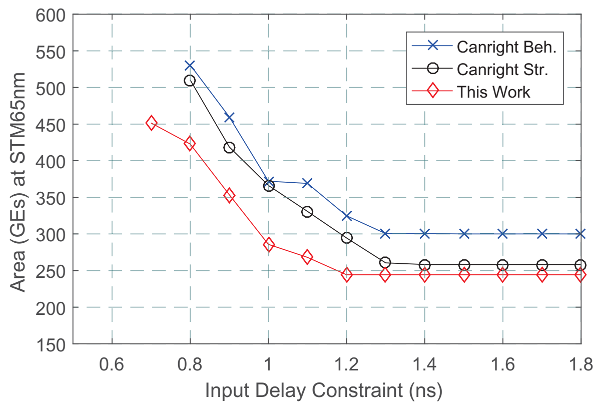
Design	Impl.	Area		Del.	M. Freq.	Pow.
		μm^2	GE	ns	MHz	μW
Canright [3]	Beh.	624	300	1.321	757.00	61.36
	CPD: $2D_{OAI22} + 1D_{XOR3} + 18D_{XOR2} + 4D_{NAND2}$					
Canright [3]	Str.	537.16	258.25*	1.304	766.87	56.86
	CPD: $2D_{AOI22} + 20D_{XOR2} + 3D_{NAND2} + 1D_{NOR2}$					
This Work	Str.	**508.04**	**244.25***	**1.159**	**862.81**	**55.00**
	CPD: $2D_{AOI22} + 1D_{OAI32} + 17D_{XOR2} + 2D_{NAND2}$					

*Compared to Table IV, one NOT gate (0.75 GEs) was added to use AOI22 gates instead of inverting MUX gates.

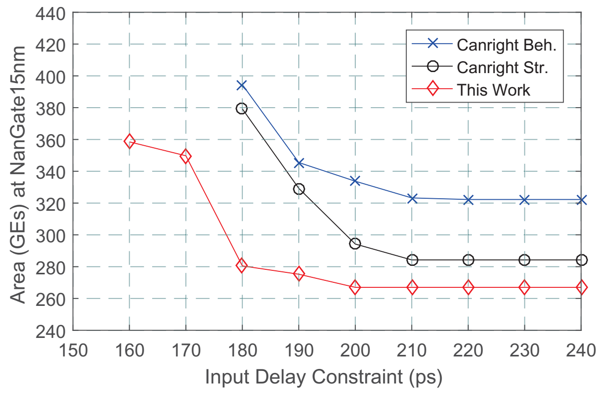
In Figure 3, we evaluate the proposed combined S-box/inverse S-box design against previous work, in [3], under different delay constraints as an input design requirement for the CAD tool. Here, we enforce using the exact same gate types, by using strict structural modeling, while allowing the CAD tool to use gates with different output strengths to improve the delay of each gate at a slight increase of the design area. Figure 3 shows that the proposed combined S-box/inverse S-box circuit results in a more compact design than the previous work, in [3], across all the targeted delay constraints. In fact, our design could be synthesized at tight delay constraints where Canright scheme became not synthesizable (slack violated). The highest speed of our design is 1.43 GHz (CPD = 0.7 nsec) in the STM 65nm technology and 6.25 GHz (CPD = 160 psec) in the NanGate 15nm technology at the corresponding areas of 451.5 GEs and 358.5 GEs, respectively.

IX. CONCLUSION

In this paper, we have proposed a new design for the AES combined S-box/inverse S-box. Using an exhaustive search, we have found a new tower field which results in more compact transformation blocks. For the new field, we have derived new formulations and designed corresponding circuits. We have designed several circuits for each block and selected the optimized ones. Moreover, we have verified our designs by extensive simulation codes and implemented our design along with the best one available in the literature. Our analysis and the ASIC implementation results show that our new combined S-box/inverse S-box outperforms the best scheme available in the literature in terms of the area and delay. To the best of our knowledge, these implementation results set a new record for the combined S-box/inverse S-box.



(a)



(b)

Fig. 3. Area in GEs of the proposed combined S-box/inverse S-box circuit as compared to previous work (in [3]) at different input delay constraints. (a) Synthesis targeting the STM 65nm library. (b) Synthesis targeting the NanGate 15nm library.

Acknowledgment

We would like to thank the reviewers of ARITH-25 for their constructive comments and the Canadian Microelectronics Corporation (CMC) Microsystems for providing the required infrastructure and the CAD tools used in this work. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under the Discovery and Discovery Accelerate Supplement (DAS) Grants awarded to A. Reyhani-Masoleh.

References

[1] J. Daemen and V. Rijmen, *The Design of Rijndaels: AES - The Advanced Encryption Standard*, ser. Information Security and Cryptography. Springer, 2002.

[2] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with S-box optimization," in *Advances in Cryptology - ASIACRYPT, Proceedings*, 2001, pp. 239–254.

[3] D. Canright, "A very compact S-box for AES," in *Cryptographic Hardware and Embedded Systems - CHES, Proceedings*, 2005, pp. 441–455.

[4] ——, "A very compact Rijndael S-box," Naval Postgraduate School Technical Report: NPS-MA-05-001, Tech. Rep., 2005.

[5] J. Boyar and R. Peralta, "A small depth-16 circuit for the AES S-box," in *Information Security and Privacy Conference - SEC, Proceedings*, 2012, pp. 287–298.

[6] J. Boyar, M. Find, and R. Peralta, "Low-depth, low-size circuits for cryptographic applications," in *Boolean Functions and their Applications - BFA, Proceedings*, 2017.

[7] R. Ueno, N. Homma, Y. Sugawara, Y. Nogami, and T. Aoki, "Highly efficient GF(2^8) inversion circuit based on redundant GF arithmetic and its application to AES design," in *Cryptographic Hardware and Embedded Systems - CHES, Proceedings*, 2015, pp. 63–80.

[8] A. Reyhani-Masoleh, M. Taha, and D. Ashmawy, "Smashing the implementation records of AES S-box," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018 (2), p. 39, 2018.

[9] P. C. Liu, H. C. Chang, and C. Y. Lee, "A 1.69 gb/s area-efficient AES crypto core with compact on-the-fly key expansion unit," in *European Solid-State Circuits Conference - ESSCIRC, Proceedings*, Sept 2009, pp. 404–407.

[10] S. K. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S. K. Hsu, H. Kaul, M. A. Anders, and R. K. Krishnamurthy, "53 gbps native $GF(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 4, pp. 767–776, 2011.

[11] K. Nekado, Y. Nogami, and K. Iokibe, "Very short critical path implementation of AES with direct logic gates," in *Advances in Information and Computer Security - IWSEC, Proceedings*, 2012, pp. 51–68.

[12] R. Ueno, S. Morioka, N. Homma, and T. Aoki, "A high throughput/gate AES hardware architecture by compressing encryption and decryption datapaths - toward efficient cbc-mode implementation," in *Cryptographic Hardware and Embedded Systems - CHES, Proceedings*, 2016, pp. 538–558.

[13] S. Mathew, S. Satpathy, V. Suresh, M. Anders, H. Kaul, A. Agarwal, S. Hsu, G. Chen, and R. Krishnamurthy, "340 mv-1.1 v, 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 Journal of Solid-State Circuits*, vol. 50, no. 4, pp. 1048–1058, 2015.

[14] S. Gueron and S. Mathew, "Hardware implementation of AES using area-optimal polynomials for composite-field representation GF$((2^4)^2)$ of GF(2^8))," in *23nd IEEE Symposium on Computer Arithmetic - ARITH, Proceedings*, 2016, pp. 112–117.

[15] J. Boyar, P. Matthews, and R. Peralta, "On the shortest linear straight-line program for computing linear forms," in *Mathematical Foundations of Computer Science - MFCS, Proceedings*, 2008, pp. 168–179.

[16] J. Boyar and R. Peralta, "A new combinational logic minimization technique with applications to cryptology," in *Experimental Algorithms, 9th International Symposium - SEA, Proceedings*, 2010, pp. 178–189.

[17] J. Boyar, P. Matthews, and R. Peralta, "Logic minimization techniques with applications to cryptology," *Journal of Cryptology*, vol. 26, no. 2, pp. 280–312, 2013.

[18] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "AES implementation on a grain of sand," *IEE Proceedings - Information Security*, vol. 152, pp. 13–20, October 2005.

[19] Y.-S. JEON, Y.-J. KIM, and D.-H. LEE, "A compact memory-free architecture for the AES algorithm using resource sharing methods," *Journal of Circuits, Systems and Computers*, vol. 19, no. 05, pp. 1109–1130, 2010.

[20] J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An asic implementation of the AES SBoxes," in *Topics in Cryptology — CT-RSA 2002*, 2002, pp. 67–78.

[21] N. Ahmad and S. R. Hasan, "Low-power compact composite field AES S-Box/inv S-Box design in 65nm CMOS using novel XOR gate," *Integration, the VLSI Journal*, vol. 46, no. 4, pp. 333 – 344, 2013.

[22] S. Banik, A. Bogdanov, and F. Regazzoni, "Compact circuits for combined AES encryption/decryption," *Journal of Cryptographic Engineering*, pp. 1–15, Oct 2017.

[23] D. Canright and L. Batina, "A very compact "perfectly masked" S-Box for AES," in *Applied Cryptography and Network Security*, 2008, pp. 446–459.

[24] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases," *Information and computation*, vol. 78, no. 3, pp. 171–177, 1988.

[25] C. Paar, "Efficient VLSI architectures for bit parallel computation in galios fields," Ph.D. dissertation, University of Duisburg-Essen, Germany, 1994.

[26] A. Reyhani-Masoleh and M. A. Hasan, "Efficient multiplication beyond optimal normal bases," *IEEE Trans. Computers*, vol. 52, no. 4, pp. 428–439, 2003.

[27] X. Zhang and K. K. Parhi, "High-speed vlsi architectures for the aes algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 9, pp. 957–967, 2004.