An Optimized S-Box Circuit Architecture for Low Power AES Design

Sumio Morioka and Akashi Satoh

IBM Research, Tokyo Research Laboratory, IBM Japan Ltd.,
1623-14 Shimotsuruma, Yamato-shi, Kanagawa-ken 242-8502, Japan
{e02716,akashi}@jp.ibm.com

Abstract. Reducing the power consumption of AES circuits is a critical problem when the circuits are used in low power embedded systems. We found the S-Boxes consume much of the total AES circuit power and the power for an S-Box is mostly determined by the number of dynamic hazards. In this paper, we propose a low-power S-Box circuit architecture: a multi-stage PPRM architecture over composite fields. In this S-Box, (i) the signal arrival times of gates are as close as possible if the depths of the gates from the primary inputs are the same, and (ii) the hazard-transparent XOR gates are located after the other gates that may block the hazards. A low power consumption of 29 μW at 10 MHz using 0.13 μm 1.5V CMOS technology was achieved, while the consumptions of the BDD, SOP, and composite field S-Boxes are 275, 95, and 136 μW, respectively.

1 Introduction

DES (Data Encryption Standard) has been used as a de facto standard cipher for more than 20 years. In 2001, NIST (National Institute of Standards and Technology) made Rijndael the new standard cipher AES [1,2].

Reducing the power consumption of AES (Advanced Encryption Standard) circuits is a critical problem when the circuits are used in embedded systems. Many circuit architectures for AES have been proposed recently and their performances have been evaluated by using ASIC libraries [3,4] and FPGAs [5,6]. However, most of them are simple implementations according to the AES specification, and there is no report of low-power AES implementations, as far as the authors know.

In this paper, we investigated a design methodology for a low-power AES. Although many power reduction techniques for IP cores are known for various design abstraction levels from the algorithm level to the transistor level [7], we focused on the logic level (gate level), because the power optimization techniques at this level can be applied in many applications. The optimizations at the other levels cannot be adopted as often, because the AES circuit is usually used as an IP core in a system, and changing the algorithm, operator scheduling, data-path architecture, and/or arrangements of transistors of the AES is difficult under given requirements for throughput, clock speed, and technology library.

By investigating the power consumption of each primitive component in AES circuits, we found the S-Box in the SubBytes component consumes much of the total

B.S. Kaliski Jr. et al. (Eds.): CHES 2002, LNCS 2523, pp. 172–186, 2003.

power (for instance, 75% is consumed in a 1 round/cycle loop architecture). The power consumption of the clock drivers is not large. When the circuit structure of the S-Box is changed, the power of the S-Box circuits can vary more than several-fold, due to the changes in the situations creating and propagating dynamic hazards, even though the total circuit size has less effect on the power consumption than expected. In fact, the power consumption of SOP (Sum of Products) S-Box is less than that of a composite field S-Box [8,9,10], while this circuit size is much larger.

We have developed a low-power S-Box architecture: a multi-stage PPRM (Positive Polarity Reed-Muller form [11]) architecture for compact S-Boxes. It is an improvement of the composite field S-Box, and in this S-Box, the gates are arranged so that: (i) the signal arrival times at the gates are as close as possible if the depths of the gates from the primary inputs are the same, to avoid generating dynamic hazards, and (ii) the hazard-transparent XOR gates are located after the other gates that may block the hazards, to avoid the propagation of dynamic hazards. The multi-stage PPRM S-Box archives the lowest power consumption of 29 μW at 10 MHz using 0.13 μm 1.5 V CMOS technology, and its circuit size is still much smaller than conventional S-Box implementations whose power consumptions are around 140 μW.

This paper is organized as follows. In Section 2, a standard AES circuit implementation is shown. In Section 3, the results of power analysis of the AES are described. In Section 4, the proposed S-Box architecture and its ASIC implementation results are explained.

2 AES Algorithm and Its Circuit Implementations

2.1 AES Algorithm

In the AES algorithm [1,2], a sequence of four primitive functions, SubBytes, ShiftRows, MixColumns, and AddRoundKey, are executed Nr-1 times. Each loop is called a round and the concrete value of Nr is 10, 12, or 14 depending on the key length. Prior to this main loop, AddRoundKey is executed for initialization. After executing the main loop, SubBytes, ShiftRows, and AddRoundKey is executed are executed once more as the final round. In the decryption process, the inverse operations of each primitive function are executed.

SubBytes is a one-byte input/output nonlinear transformation that uses 16-byte (128-bit) S-Boxes. Each S-Box is a multiplicative inversion on a Galois field GF(2^8) followed by an affine transformation. The irreducible polynomial used by the field is $m(x)=x^8+x^4+x^3+x+1$. ShiftRows is a cyclic shift operation in each row of four 4-byte data blocks using 0~3-byte offsets. MixColumns treats the 4-byte data blocks in each column as coefficients of a 4-term polynomial, and multiplies the data modulo x^4+1 with a fixed polynomial. AddRoundKey is a simple bit-wise XOR operation on the 128-bit round keys and the data.

In Fig. 1, a standard circuit implementation of AES, which executes 1 round per clock, is shown. A sequence of round operations is implemented as a combinational circuit and its input and output are connected to a 128-bit data register. The 3:1 selector before the AddRoundKey (XOR) is used to skip some operations in the first and last rounds.

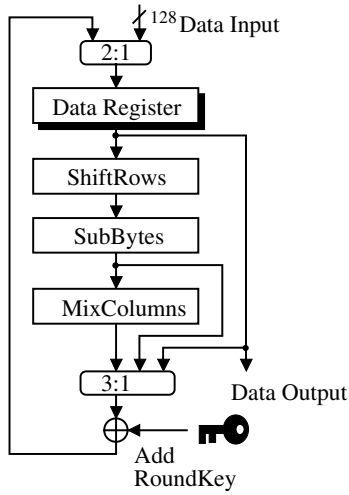

Fig. 1. A standard implementation of an AES encryption circuit (1 round/clock)

2.2 Various S-Box Circuit Implementations

There are two approaches for designing S-Box circuits:

(1) construct a multiplicative inversion circuit and an affine transformation circuit independently, and then connect these two circuits in serial,
(2) construct a single circuit directly whose input-output relation is equivalent to the S-Box.

In Method (1), circuit area reductions using mathematical theorems over Galois fields (GF) [12] are possible. Various methods for constructing compact inversion circuits over GF have been studied, based on Fermat's Little Theorem [14], the Low-latency algorithm of Itoh and Tsujii [13,14], the extended Euclid's Algorithm, and so on. In particular, the composite field (or tower field) inversion [8] is effective over $GF(2^8)$, and it can be used to create compact AES implementations [9,10]. The detail of the composite field technique will be explained in the next sub-section.

In Method (2), a fast implementation is possible. The S-Box circuit can be obtained from its truth table by using two-level logic such as SOP, POS (Product of Sums), PPRM [11], or by using decision diagrams such as BDD (Binary Decision Diagram) [15]. Our investigations' results in [16,17] show that the variable ordering of the BDD does not have much effect on the size and speed of the S-Box and GF inverter. Based on this research, we designed a very fast S-Box called twisted-BDD [17], which is 1.5 to 2 times faster than the other S-Box implementations.

Although in many AES implementations the table-lookup method is used, where the S-Box circuit is automatically synthesized using EDA tools, the performance of these synthesized circuits is usually close to that of SOP or BDD implementations.

2.3 A S-Box Implementation Based on Composite Field Technique

Fig. 2 shows the outline of an S-Box implementation using the composite field technique [9,10]. The most costly operation in the S-Box is the multiplicative inversion over a field A (the AES field), where A is extended from GF(2) with the irreducible polynomial $m(x)$ mentioned in Section 2.1. To reduce the cost of this operation, the following 3-stage method is adopted:

(Stage 1) Map all elements of the field A to a composite field B, using an isomorphism function δ

(Stage 2) Compute the multiplicative inverses over the field B.

(Stage 3) Re-map the computation results to A, using the function δ^{-1}.

The composite field B in Stage 2 is constructed not by applying a single degree-8 extension to GF(2), but by applying multiple extensions of smaller degrees. To reduce the cost of Stage 2 as much as possible, it is known to be efficient to construct the composite field B using repeated degree-2 extensions under a polynomial basis using these irreducible polynomials [10],

$$\begin{cases} \mathrm{GF}(2^2): & x^2+x+1 \\ \mathrm{GF}((2^2)^2): & x^2+x+\phi \\ \mathrm{GF}(((2^2)^2)^2): & x^2+x+\lambda \end{cases}$$

where $\phi=\{10\}_2$, $\lambda=\{1100\}_2$. The inverter over the field above has fewer GF(2) operators compared with the composite field used in [9],

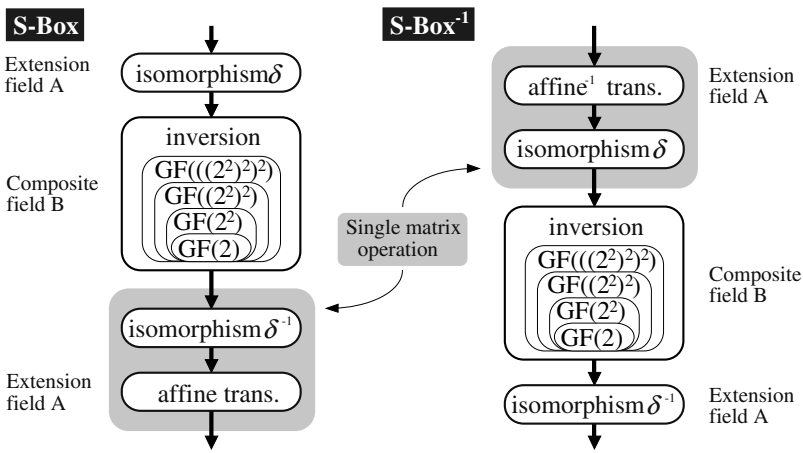

Fig. 2. Computation sequences of composite-field-based S-Box

$$\begin{cases} \mathrm{GF}(2^4): & x^4 + x + 1 \\ \mathrm{GF}((2^4)^2): & x^2 + x + \omega_{14} \end{cases}$$

where $\omega_{14}=\{1001\}_2$.

An implementation of Stage 2 is shown in Fig. 3. For any composite fields $\mathrm{GF}((2^m)^n)$, computing the multiplicative inverses can be done as a combination of operations over the sub-fields $\mathrm{GF}(2^n)$, using the following equation [8]:

$$P^{-1} = (P^r)^{-1} \cdot P^{r-1}, \quad \text{where} \quad r = (2^{nm}-1)/(2^m-1). \tag{1}$$

For AES ($n = 2$, $m = 2^2$), this equation becomes

$$P^{-1} = (P^{17})^{-1} \cdot P^{16}. \tag{2}$$

The circuit in Fig. 3 is an implementation of Equation (2), with additional optimizations. In the circuit, P^{16} is computed first and then P^{17} is obtained by multiplying P by P^{16} over $\mathrm{GF}(((2^2)^2)^2)$. Because P^{17} is always an element of $\mathrm{GF}((2^2)^2)$ (i.e., the upper 4 bits of P^{17} are always 0), computing the upper 4 bits of P^{17} is unnecessary [8]. The value of $(P^{17})^{-1}$ is computed recursively over $\mathrm{GF}((2^2)^2)$, then multiplied by P^{16} over $\mathrm{GF}(((2^2)^2)^2)$, and finally P^{-1} is obtained. This final multiplication requires fewer circuit resources than conventional multiplication over $\mathrm{GF}(2^8)$, because P^{17} is an element of $\mathrm{GF}((2^2)^2)$. Further gate reduction is possible by sharing circuit gates of the three $\mathrm{GF}((2^2)^2)$ multipliers in Fig. 3, where common inputs are used. Note that our multipliers and inverter over subfields $\mathrm{GF}((2^2)^2)$ and $\mathrm{GF}(2^2)$ are also small.

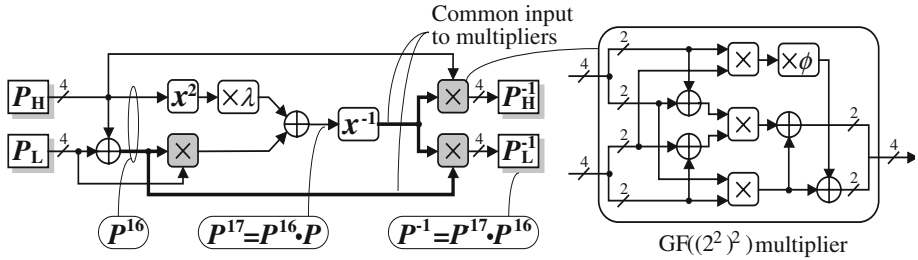

Fig. 3. Computation sequences of composite-field-based inverter

The isomorphism functions δ and δ^{-1} in Stages 1 and 3 were constructed as follows. First, search for a primitive element α in the field A and a primitive element β in the field B, where both α and β are roots of a same primitive irreducible polynomial. Any primitive irreducible polynomial can be used, and here we used $p(x)=x^8+x^4+x^3+x^2+1$. Once such elements α and β are found, the definition table of the isomorphism functions δ and δ^{-1} are immediately determined, where α^k is mapped to β^k (or β^k to α^k) for any $1 \le k \le 254$. The hardware implementation of these functions can be obtained by mapping the basis elements of A (or B) into B (or A), and these mappings are described as multiplications of constant matrices over GF(2). The concrete descriptions of δ and δ^{-1} are

$$\delta = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \quad \delta^{-1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix},$$

where the least significant bits are in the upper left corners. These matrices can be merged with affine transformations for circuit size reduction, as shown in Fig. 2. All of these isomorphism functions and the constant multipliers in the S-Boxes are implemented as XOR arrays, and their Boolean logic is compressed by applying a factoring technique based on a greedy algorithm [18].

3 Analysis of Power Consumption in AES Circuits

3.1 Power Analysis Method

For analyzing the power consumption of the AES circuits, a simulation-based analysis method was used. In this method, a timing simulation (or delay simulation) at the gate level is performed using a given set of test input data, and the switching activities of all internal gates are logged. Then the circuit power is computed from this simulation log and the cell information of the target ASIC library. Although this analysis method requires much more CPU time than the often used probabilistic method (a kind of static analysis method) [7], it is much more accurate because the effects of dynamic hazards can be reflected in the power estimation results.

3.2 Power Consumption of Various S-Box Architectures

The power estimation results for each AES component in a 0.13 μm ASIC standard cell library are shown in Table 1. Note that the estimation results depends on the test data, although we observed no large differences between the various test data sets we used. In this table, the SOP S-Box is used. Clearly, the SubBytes operations (16 S-Boxes) consume much more power than the other components.

In Tables 2 and 3, a performance comparison of various S-Boxes in 0.13μm and 0.18μm ASIC libraries, including the proposed method (which will be described in Section 4), is shown. We checked all patterns of primary input switching ($2^8 \times 2^8$ = 65,536 patterns) and the average power consumptions obtained are shown in the tables. Although the absolute values of circuit performance are different between the two ASIC libraries, the relative relationships within each architecture are almost the same.

Until these experimental results were obtained, we thought the power consumption of an AES with the composite field S-Box would be the lowest, because the circuit size of the composite field S-Box is very small. However, the estimated power consumption was unexpectedly large. In addition, the power consumption of the BDD S-Box was much larger than that of the SOP, although their speed and size are similar.

Table 1. Power consumption of each AES component
(0.13 μm 1.5 V CMOS standard cell, 1 gate = 2 way-NAND)

	Observed Max. Power (μW@10MHz)	Power ratio (%)
SubBytes (SOP S-Box $\times$16)	1,940	75
MixColumns	262	10
AddRoundKey	> 10	> 1
Data Selectors	> 10	> 1
FFs + Clock Drivers	400	15

Table 2. Comparison of various S-Box architectures
(0.13 μm 1.5 V CMOS standard cell, 1 gate = 2 way-NAND)

	Delay (ns)	Size (gate)	Average Power of S-Box (μW@10 MHz)
Itoh and Tsujii [14]	2.79	1,771	2,100
PPRM (1-stage)	1.14	2,241	343
BDD	0.69	1,399	275
Twisted-BDD [17]	0.43	2,818	272
Table lookup	0.68	2,623	144
Composite Field [10]	2.19	354	136
SOP (1-stage)	0.69	1,650	95
Proposed method (3-stage PPRM)	1.43	712	29

Table 3. Comparison of various S-Box architectures
(0.18 μm 1.8 V CMOS standard cell, 1 gate = 2 way-NAND)

	Delay (ns)	Size (gate)	Average Power of S-Box (μW@10 MHz)
Itoh and Tsujii [14]	4.11	1,540	3,490
PPRM (1-stage)	1.32	2,242	408
Twisted-BDD [17]	0.66	1,977	334
BDD	0.96	857	332
Table look-up	0.91	1,706	206
Composite Field [10]	3.01	305	166
SOP (1-stage)	0.97	1,142	138
Proposed method (3-stage PPRM)	1.86	701	51

3.3 Analysis

We determined that the power consumption of the S-Boxes was strongly influenced by the number of dynamic hazards. In fact, the following two characteristics A and B, which are the main reasons for creating or propagating dynamic hazards, are significantly different among the S-Boxes.

A. Differences of Signal Arrival Time at Each Gate

The dynamic hazards can occur when the signal arrival times are different between multiple inputs of a gate. If multiple gates are connected serially and some of the internal gates generate hazards, then the hazards propagate into the circuit path and some extra power is consumed.

Regarding S-Boxes, the composite field S-Box involves many crossing and/or branched signal paths. The signal arrival times of the internal gates are very different and as a result, the gates (or sub-operators) can switch many times per single transition of the primary inputs, as shown in Fig. 4. This is the main reason for its relatively large power consumption in spite of the small circuit size. The situation is the same with the BDD S-Box and S-Box based on the algorithm of Itoh and Tsujii. On the other hand, the signal arrival time of each gate is not much different in the S-Boxes based on two-level logic, such as the SOP and PPRM S-Box.

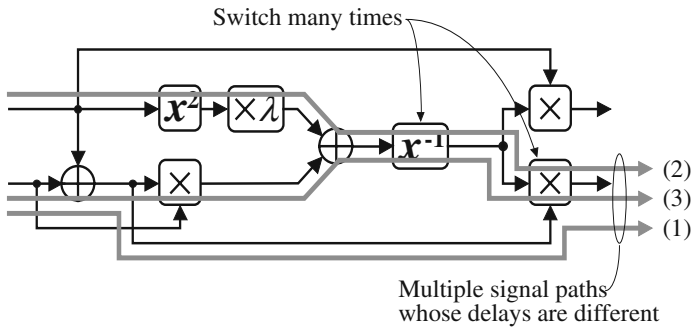

Fig. 4. Dynamic hazards caused by differences of signal arrival time

B. Propagation Probability of Signal Transitions

The kinds of the logic gates closely relates to the propagation of hazards. In particular, the use of XOR gates can increase the power consumption, because their probability of propagating a signal transition is 1, i.e. all hazards can be propagated from input ports to output port, while the probabilities are 0.5 in the other gates such as AND and OR. The use of many XORs is another main reason for the large power consumptions of the composite field S-Box and PPRM S-Box, compared to the SOP S-Box.

4 The Proposed Low Power S-Box Architecture and Its Evaluation Results

4.1 The Proposed Multi-stage PPRM Architecture

The complicated signal paths of the composite field S-Boxes, which is the main reason for their large power consumption, can be simplified by converting some parts of the S-Box logic into two-level logic. Although converting the entire circuit into a single two-level logic structure would increase the circuit size and lose an important advantage of the composite field S-Box, converting carefully selected sub-components into two-level logic can generate a much better S-Box, if an appropriate partitioning of the S-Box into sub-components is done.

In Fig. 5, the example S-Box circuit with the best performance in our trials is shown. Three sub-components of the composite field S-Box were converted into PPRM form: the pre-inversion section, the inversion section, and the post-inversion section (see Appendix for their complete description). Two signal paths not passing through the inversion section were connected to delay chains whose delay time is close to that of the inversion. In this circuit, the signal arrival times at the gates are almost equal, if the depths of the gates from primary inputs are the same. In addition, the power consumption of the pre-inversion part can be greatly reduced, because the XOR gates are located after the AND gates. The original pre-inversion part has the

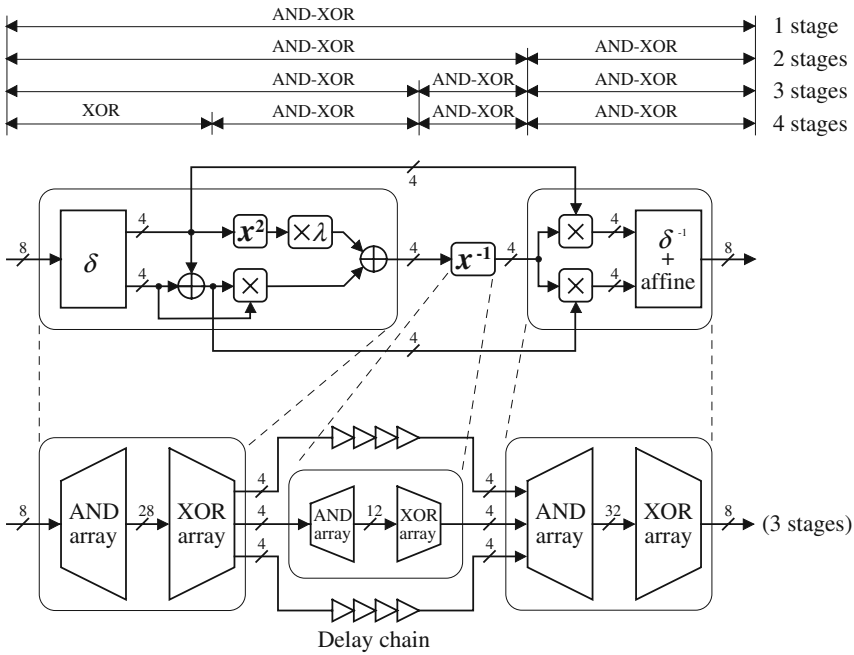

Fig. 5. A conventional composite field S-Box (top) and the proposed method called multi-stage PPRM (bottom)

opposite XOR-AND structure, and the XOR gates must always switch state on any change of the primary inputs.

As shown in Tables 2 and 3 in Section 3.2, the 3-stage PPRM S-Box achieves the lowest power consumption. Its circuit size is much smaller than conventional S-Box implementations. The circuit speed is not so fast, but it is still fast enough for embedded systems.

When other partitioning methods such as a 2-stage PPRM, a 4-stage (XOR)-AND-XOR etc. were used (Fig. 5), the power consumption was increased, as shown in Table 4. The original composite field S-Box has a 5-stage (XOR)-AND-XOR structure. Of those the authors tested, the 3-stage PPRM gives the lowest power consumption. Another 2-stage PPRM implementation besides the one shown in Fig. 5 (combined inversion and post-inversion sections) is possible, but its circuit size and power consumption are similar to those of the 1-stage PPRM implementation. In addition, when each section was implemented in a form other than PPRM (such as SOP), the circuit size and power consumption became significantly larger (see Table 4), because these sections contain many XOR gates.

Please note that computing over a composite field is highly recommended, although the proposed circuit architecture only requires the use of Equation (2) in Section 2.3 and the equation is independent of the field (i.e., the equation even holds if a composite field is not used). If the inversion is not done over a composite field, P^{17} in Equation (2) becomes an 8-bit vector and as a result, the circuit size must be much larger.

Table 4. Number of logic stages vs. S-Box performance
(0.13 μm 1.5 V CMOS standard cell, 1 gate = 2 way-NAND)

		Size (gate)	Average Power of S-Box (μW@10 MHz)
PPRM	1-stage	2,241	343
	2-stages	1,445	273
	3-stages	712	29
	4-stages	413	88
	5-stages (Composite Field [10])	354	136
SOP	1-stage	1,650	95
	2-stages	5,891	612
	3-stages	6,114	697

4.2 Circuit Sharing between S-Box and S-Box^{-1}

The proposed multi-stage PPRM architecture is suitable for implementing AES circuits that perform both encryption and decryption, because most gates of the S-Box (for encryption) and the S-Box^{-1} (for decryption) can be shared.

An example shared S-Box circuit is shown in Fig. 6. In this circuit, an inverter over the AES field is shared between the S-Box and the S-Box^{-1}. The power consumption of the shared S-Box is larger than that of the unshared 3-stage PPRM S-Box, but it is still smaller than most S-Box implementations (see Table 5). Similar circuit sharing is possible for the composite field S-Box, but the power consumption is very large. Note

that SOP and the other conventional S-Box implementations require two different circuits to support both encryption and decryption, and the total circuit size (S-Box + S-Box^{-1}) can exceed 2,000 gates.

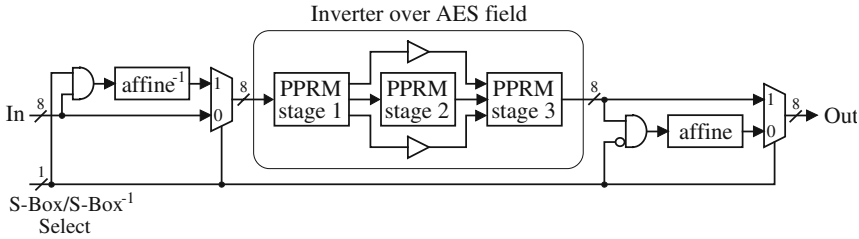

Fig. 6. Circuit sharing between S-Box and S-Box^{-1} under the multi-stage PPRM architecture

Table 5. Performance comparison of shared S-Box architectures (0.13 μm 1.5 V CMOS standard cell, 1 gate = 2 way-NAND)

	Delay (ns)	Size (gate)	Average Power (μW@10 MHz)
Composite Field [10]	2.53	381	179 (S-Box)
			189 (Inv S-Box)
Proposed Method (3-stage PPRM)	2.00	725	79 (S-Box)
			70 (Inv S-Box)

5 Conclusion

In this paper, we have developed a multi-stage PPRM architecture for low-power S-Box circuits, because the S-Boxes consume much of the total power of AES designs. The power consumption of S-Box circuits can be significantly reduced by avoiding the creation and propagation of dynamic hazards. A power consumption of only 29 μW at 10 MHz using a 0.13 μm 1.5V CMOS technology was achieved, while the power of the conventional BDD, SOP and composite field S-Boxes are 275, 95 and 136 μW, respectively.

References

1. J. Daemen and V. Rijmen, “AES Proposal: Rijndael,” http://csrc.nist.gov/ encryption/aes/rijndael/Rijndael.pdf.
2. National Institute of Standards and Technology (NIST), “Advanced Encryption Standard (AES)”, FIPS Publication 197, http://csrc.nist.gov/encryption/ aes/index.html, Nov. 2001.
3. H. Kuo et al., "Architectural Optimization for a 1.82 Gbits/sec VLSI Implementation of the AES Rijndael Algorithm," *Proc. CHES2001*, LNCS Vol. 2162, pp. 53–67, 2001.

4. B. Weeks et al., "Hardware Performance Simulation of Round 2 Advanced Encryption Standard Algorithm," http://csrc.nist.gov/encryption/aes/round2/ NSA-AESfinalreport.pdf.
5. M. McLoone et al., "High performance single-chip FPGA Rijndael algorithm implementations," *Proc. CHES2001*, LNCS Vol. 2162, pp. 68–80, 2001.
6. V. Fischer et al, "Two methods of Rijndael implementation in reconfigurable hardware," *Proc. CHES2001*, LNCS Vol. 2162, pp. 81–96, 2001.
7. A.P. Chandrakasan and R.W. Brodersen (eds.), *Low Power Digital CMOS Design*, Kluwer Academic Publishers, 1995.
8. J. Guajardo and C. Paar, "Efficient Algorithms for Elliptic Curve Cryptosystems," *CRYPTO'97*, LNCS Vol. 1294, pp. 342–356, 1997.
9. A. Rudra et al, "Efficient Rijndael encryption implementation with composite field arithmetic," *Proc. CHES2001*, LNCS Vol. 2162, pp. 175–188, 2001.
10. A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," Advances in Cryptology – *ASIACRYPT 2001*, LNCS Vol. 2248, pp. 239–254, 2001.
11. T. Sasao, "AND-EXOR expressions and their optimization", in Sasao, editor: *Logic Synthesis and Optimization*, Kluwer Academic Publishers, pp. 287–312, 1993.
12. I.F. Blake, X. Gao, R.C. Mullin, S.A. Vanstone and T. Yaghoobian, *Applications of Finite Fields*, Kluwer Academic Publishers. 1993.
13. 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.
14. S. Morioka and Y. Katayama, "O($\log_2 m$) Iterative Algorithm for Multiplicative Inverse in GF(2^m)," *IEEE Intl. Symp. On Info. Theory (ISIT2000)*, pp. 449 ff., 2000.
15. R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Trans. on Computers*, Vol. C-35, No. 8, pp. 677–691, 1986.
16. S. Morioka, Y. Katayama, and T. Yamane, "Towards Efficient Verification of Arithmetic Algorithms over Galois Fields GF(2^m)," *13th Conference on Computer Aided Verification (CAV'01)*, LNCS Vol. 2102, pp. 465–477, 2001.
17. S. Morioka and A. Satoh, "A 10 Gbps Full-AES Crypto Design with a Twisted-BDD S-Box Architecture," *2002 IEEE Intl. Conf. on Computer Design (ICCD2002)*, 2002.
18. S. Morioka and Y. Katayama, "Design Methodology for one-shot Reed-Solomon Encoder and Decoder," *1999 IEEE Intl. Conf. on Computer Design (ICCD'99)*, pp. 60–67, 1999.

Appendix: PPRM Representations of Each Stage (3-Stage PPRM)

Variables x_7-x_0 denote primary inputs of an S-Box/S-Box^{-1}/Inverter and y_7-y_0 denote primary outputs (x_7 and y_7 are MSB). The other variables such as a and b denote internal wires.

A-1. Stage 1 of S-Box

$a_3 = x_7 \text{ xor } x_5$
$a_2 = x_7 \text{ xor } x_6 \text{ xor } x_4 \text{ xor } x_3 \text{ xor } x_2 \text{ xor } x_1$
$a_1 = x_7 \text{ xor } x_5 \text{ xor } x_3 \text{ xor } x_2$
$a_0 = x_7 \text{ xor } x_5 \text{ xor } x_3 \text{ xor } x_2 \text{ xor } x_1$
$b_3 = x_5 \text{ xor } x_6 \text{ xor } x_2 \text{ xor } x_1$
$b_2 = x_6$
$b_1 = x_7 \text{ xor } x_5 \text{ xor } x_3 \text{ xor } x_2 \text{ xor } x_6 \text{ xor } x_4 \text{ xor } x_1$
$b_0 = x_7 \text{ xor } x_5 \text{ xor } x_3 \text{ xor } x_2 \text{ xor } x_6 \text{ xor } x_0$

$c_3 = (x_5 \text{ and } x_1) \text{ xor } (x_7 \text{ and } x_1) \text{ xor } (x_5 \text{ and } x_2) \text{ xor } (x_5 \text{ and } x_6) \text{ xor } (x_5 \text{ and } x_7) \text{ xor } (x_5 \text{ and } x_4) \text{ xor } (x_7 \text{ and } x_4) \text{ xor } (x_5 \text{ and } x_0) \text{ xor } (x_7 \text{ and } x_0) \text{ xor } (x_3 \text{ and } x_1) \text{ xor } (x_4 \text{ and } x_1) \text{ xor } (x_3 \text{ and } x_2) \text{ xor } (x_2 \text{ and } x_4) \text{ xor } (x_4 \text{ and } x_6) \text{ xor } (x_2 \text{ and } x_1) \text{ xor } (x_2 \text{ and } x_6) \text{ xor } (x_6 \text{ and } x_1)$

$c_2 = (x_6 \text{ and } x_1) \text{ xor } (x_2 \text{ and } x_6) \text{ xor } (x_3 \text{ and } x_6) \text{ xor } (x_7 \text{ and } x_6) \text{ xor } (x_1 \text{ and } x_0) \text{ xor } (x_2 \text{ and } x_0) \text{ xor } (x_3 \text{ and } x_0) \text{ xor } (x_4 \text{ and } x_0) \text{ xor } (x_6 \text{ and } x_0) \text{ xor } (x_7 \text{ and } x_0) \text{ xor } (x_5 \text{ and } x_2) \text{ xor } (x_5 \text{ and } x_3) \text{ xor } (x_2 \text{ and } x_4) \text{ xor } (x_3 \text{ and } x_4) \text{ xor } (x_5 \text{ and } x_7) \text{ xor } (x_7 \text{ and } x_2) \text{ xor } (x_5 \text{ and } x_6) \text{ xor } (x_3 \text{ and } x_2) \text{ xor } (x_7 \text{ and } x_3)$

$c_1 = (x_2 \text{ and } x_1) \text{ xor } (x_2 \text{ and } x_4) \text{ xor } (x_5 \text{ and } x_4) \text{ xor } (x_3 \text{ and } x_6) \text{ xor } (x_5 \text{ and } x_6) \text{ xor } (x_2 \text{ and } x_0) \text{ xor } (x_3 \text{ and } x_0) \text{ xor } (x_5 \text{ and } x_0) \text{ xor } (x_7 \text{ and } x_0) \text{ xor } x_1 \text{ xor } (x_5 \text{ and } x_2) \text{ xor } (x_7 \text{ and } x_2) \text{ xor } (x_5 \text{ and } x_3) \text{ xor } (x_5 \text{ and } x_7) \text{ xor } x_7 \text{ xor } x_2 \text{ xor } (x_3 \text{ and } x_2) \text{ xor } x_4 \text{ xor } x_5$

$c_0 = (x_1 \text{ and } x_0) \text{ xor } (x_2 \text{ and } x_0) \text{ xor } (x_3 \text{ and } x_0) \text{ xor } (x_5 \text{ and } x_0) \text{ xor } (x_7 \text{ and } x_0) \text{ xor } (x_3 \text{ and } x_1) \text{ xor } (x_6 \text{ and } x_1) \text{ xor } (x_3 \text{ and } x_6) \text{ xor } (x_5 \text{ and } x_6) \text{ xor } (x_7 \text{ and } x_6) \text{ xor } (x_3 \text{ and } x_4) \text{ xor } (x_7 \text{ and } x_4) \text{ xor } (x_5 \text{ and } x_3) \text{ xor } (x_4 \text{ and } x_1) \text{ xor } x_2 \text{ xor } (x_3 \text{ and } x_2) \text{ xor } (x_4 \text{ and } x_6) \text{ xor } x_6 \text{ xor } x_5 \text{ xor } x_3 \text{ xor } x_0$

A-2. Stage 2 of S-Box

$d_3 = (c_3 \text{ and } c_2 \text{ and } c_1) \text{ xor } (c_3 \text{ and } c_0) \text{ xor } c_3 \text{ xor } c_2$

$d_2 = (c_3 \text{ and } c_2 \text{ and } c_0) \text{ xor } (c_3 \text{ and } c_0) \text{ xor } (c_3 \text{ and } c_2 \text{ and } c_1) \text{ xor } (c_2 \text{ and } c_1) \text{ xor } c_2$

$d_1 = (c_3 \text{ and } c_2 \text{ and } c_1) \text{ xor } (c_3 \text{ and } c_1 \text{ and } c_0) \text{ xor } c_3 \text{ xor } (c_2 \text{ and } c_0) \text{ xor } c_2 \text{ xor } c_1$

$d_0 = (c_3 \text{ and } c_2 \text{ and } c_0) \text{ xor } (c_3 \text{ and } c_1 \text{ and } c_0) \text{ xor } (c_3 \text{ and } c_2 \text{ and } c_1) \text{ xor } (c_3 \text{ and } c_1) \text{ xor } (c_3 \text{ and } c_0) \text{ xor } (c_2 \text{ and } c_1 \text{ and } c_0) \text{ xor } c_2 \text{ xor } (c_2 \text{ and } c_1) \text{ xor } c_1 \text{ xor } c_0$

A-3. Stage 3 of S-Box

$y_7 = (d_3 \text{ and } a_0) \text{ xor } (d_2 \text{ and } a_1) \text{ xor } (d_1 \text{ and } a_2) \text{ xor } (d_0 \text{ and } a_3) \text{ xor } (b_2 \text{ and } d_3) \text{ xor } (b_3 \text{ and } d_2) \text{ xor } (b_2 \text{ and } d_2) \text{ xor } (d_3 \text{ and } a_3) \text{ xor } (d_3 \text{ and } a_1) \text{ xor } (d_1 \text{ and } a_3) \text{ xor } (b_0 \text{ and } d_2) \text{ xor } (b_2 \text{ and } d_0) \text{ xor } (d_3 \text{ and } a_2) \text{ xor } (d_2 \text{ and } a_3) \text{ xor } (b_0 \text{ and } d_3) \text{ xor } (b_1 \text{ and } d_2) \text{ xor } (b_2 \text{ and } d_1) \text{ xor } (b_3 \text{ and } d_0)$

$y_6 = \text{'1' xor } (a_0 \text{ and } d_2) \text{ xor } (a_2 \text{ and } d_0) \text{ xor } (d_3 \text{ and } a_3) \text{ xor } (a_0 \text{ and } d_1) \text{ xor } (a_1 \text{ and } d_0) \text{ xor } (d_3 \text{ and } a_2) \text{ xor } (d_2 \text{ and } a_3) \text{ xor } (a_0 \text{ and } d_0) \text{ xor } (d_3 \text{ and } a_0) \text{ xor } (d_2 \text{ and } a_1) \text{ xor } (d_1 \text{ and } a_2) \text{ xor } (d_0 \text{ and } a_3)$

$y_5 = \text{'1' xor } (d_3 \text{ and } a_3) \text{ xor } (d_3 \text{ and } a_1) \text{ xor } (d_1 \text{ and } a_3) \text{ xor } (d_3 \text{ and } a_2) \text{ xor } (d_2 \text{ and } a_3) \text{ xor } (b_2 \text{ and } d_2) \text{ xor } (b_0 \text{ and } d_2) \text{ xor } (b_2 \text{ and } d_0) \text{ xor } (b_3 \text{ and } d_3) \text{ xor } (b_1 \text{ and } d_3) \text{ xor } (b_3 \text{ and } d_1) \text{ xor } (d_3 \text{ and } a_0) \text{ xor } (d_2 \text{ and } a_1) \text{ xor } (d_1 \text{ and } a_2) \text{ xor } (d_0 \text{ and } a_3)$

$y_4 = (d_3 \text{ and } a_1) \text{ xor } (d_1 \text{ and } a_3) \text{ xor } (a_0 \text{ and } d_0) \text{ xor } (b_3 \text{ and } d_3) \text{ xor } (b_0 \text{ and } d_1) \text{ xor } (b_1 \text{ and } d_0) \text{ xor } (d_3 \text{ and } a_0) \text{ xor } (d_2 \text{ and } a_1) \text{ xor } (d_1 \text{ and } a_2) \text{ xor } (d_0 \text{ and } a_3) \text{ xor } (a_1 \text{ and } d_1) \text{ xor } (b_2 \text{ and } d_2) \text{ xor } (b_0 \text{ and } d_0)$

$y_3 = (b_0 \text{ and } d_1) \text{ xor } (b_1 \text{ and } d_0) \text{ xor } (b_0 \text{ and } d_2) \text{ xor } (b_2 \text{ and } d_0) \text{ xor } (b_1 \text{ and } d_3) \text{ xor } (b_3 \text{ and } d_1) \text{ xor } (b_0 \text{ and } d_0)$

$y_2 = (a_0 \text{ and } d_2) \text{ xor } (a_2 \text{ and } d_0) \text{ xor } (a_0 \text{ and } d_1) \text{ xor } (a_1 \text{ and } d_0) \text{ xor } (b_1 \text{ and } d_1) \text{ xor } (b_2 \text{ and } d_2) \text{ xor } (d_3 \text{ and } a_1) \text{ xor } (d_1 \text{ and } a_3) \text{ xor } (b_0 \text{ and } d_2) \text{ xor } (b_2 \text{ and } d_0) \text{ xor } (b_3 \text{ and } d_3) \text{ xor } (a_0 \text{ and } d_0) \text{ xor } (b_0 \text{ and } d_3) \text{ xor } (b_1 \text{ and } d_2) \text{ xor } (b_2 \text{ and } d_1) \text{ xor } (b_3 \text{ and } d_0) \text{ xor } (b_0 \text{ and } d_0)$

$y_1 = \text{'1' xor } (d_3 \text{ and } a_0) \text{ xor } (d_2 \text{ and } a_1) \text{ xor } (d_1 \text{ and } a_2) \text{ xor } (d_0 \text{ and } a_3) \text{ xor } (b_1 \text{ and } d_1) \text{ xor } (b_2 \text{ and } d_3) \text{ xor } (b_3 \text{ and } d_2) \text{ xor } (d_3 \text{ and } a_3) \text{ xor } (d_3 \text{ and } a_1) \text{ xor } (d_1 \text{ and } a_3) \text{ xor } (b_3 \text{ and } d_3) \text{ xor } (d_3 \text{ and } a_2) \text{ xor } (d_2 \text{ and } a_3) \text{ xor } (b_0 \text{ and } d_0)$

y_0 = '1' xor (d_3 and a_0) xor (d_2 and a_1) xor (d_1 and a_2) xor (d_0 and a_3) xor (a_0 and d_2) xor (a_2 and d_0) xor (b_0 and d_1) xor (b_1 and d_0) xor (d_2 and a_2) xor (b_0 and d_2) xor (b_2 and d_0) xor (b_1 and d_3) xor (b_3 and d_1) xor (d_3 and a_2) xor (d_2 and a_3) xor (b_0 and d_0)

B-1. Stage 1 of S-Box^{-1}

a_3 = x_6 xor x_1 xor x_7 xor x_2
a_2 = '1' xor x_6 xor x_1 xor x_3 xor x_0 xor x_2 xor x_7
a_1 = '1' xor x_6 xor x_4 xor x_5 xor x_0
a_0 = '1' xor x_4 xor x_5 xor x_3
b_3 = '1' xor x_6 xor x_1 xor x_2 xor x_5
b_2 = x_3 xor x_0 xor x_5
b_1 = '1' xor x_6 xor x_4 xor x_0 xor x_3 xor x_1
b_0 = x_4 xor x_5 xor x_3 xor x_6 xor x_7 xor x_2
c_3 = x_7 and x_1 xor (x_2 and x_1) xor (x_3 and x_1) xor x_0 xor (x_7 and x_0) xor (x_5 and x_3) xor (x_3 and x_0) xor x_4 xor (x_7 and x_4) xor (x_5 and x_4) xor (x_2 and x_0) xor (x_4 and x_2) xor (x_6 and x_2) xor (x_4 and x_1) xor (x_6 and x_0) xor (x_6 and x_4)
c_2 = '1' xor x_0 xor x_7 xor (x_7 and x_0) xor (x_7 and x_5) xor (x_6 and x_5) xor (x_1 and x_0) xor (x_3 and x_1) xor x_1 xor x_6 xor (x_7 and x_1) xor x_4 xor (x_4 and x_2) xor (x_4 and x_1) xor (x_6 and x_4) xor (x_7 and x_2)
c_1 = x_1 xor (x_5 and x_1) xor (x_6 and x_1) xor (x_3 and x_0) xor (x_5 and x_3) xor (x_5 and x_0) xor (x_6 and x_5) xor x_2 xor (x_4 and x_2) xor (x_6 and x_2) xor x_0 xor x_5 xor x_4 xor x_7 xor (x_7 and x_0) xor (x_7 and x_4) xor (x_6 and x_4) xor x_6 xor (x_7 and x_2) xor (x_7 and x_1) xor (x_3 and x_2)
c_0 = '1' xor (x_3 and x_2) xor (x_4 and x_2) xor x_3 xor x_4 xor (x_7 and x_4) xor (x_6 and x_5) xor (x_6 and x_4) xor (x_1 and x_0) xor (x_4 and x_1) xor (x_6 and x_1) xor (x_3 and x_0) xor (x_4 and x_3) xor (x_5 and x_4) xor (x_7 and x_0)

B-2. Stage 2 of S-Box^{-1}

Same as the stage 2 of the S-Box in A-2.

B-3. Stage 3 of S-Box^{-1}

y_7 = (d_3 and a_0) xor (d_2 and a_1) xor (d_1 and a_2) xor (d_0 and a_3) xor (a_0 and d_2) xor (a_2 and d_0) xor (a_1 and d_1) xor (a_0 and d_1) xor (a_1 and d_0) xor (b_1 and d_1) xor (b_0 and d_1) xor (b_1 and d_0) xor (b_2 and d_3) xor (b_3 and d_2) xor (b_2 and d_2)
y_6 = (d_2 and a_2) xor (a_0 and d_2) xor (a_2 and d_0) xor (d_3 and a_3) xor (d_3 and a_1) xor (d_1 and a_3) xor (b_2 and d_2) xor (b_0 and d_2) xor (b_2 and d_0) xor (b_3 and d_3) xor (b_1 and d_3) xor (b_3 and d_1)
y_5 = (a_0 and d_2) xor (a_2 and d_0) xor (d_3 and a_3) xor (d_3 and a_1) xor (d_1 and a_3) xor (a_1 and d_1) xor (a_0 and d_1) xor (a_1 and d_0) xor (d_3 and a_2) xor (d_2 and a_3) xor (b_1 and d_1) xor (b_0 and d_1) xor (b_1 and d_0) xor (b_2 and d_3) xor (b_3 and d_2) xor (b_2 and d_2)
y_4 = (a_0 and d_2) xor (a_2 and d_0) xor (d_3 and a_1) xor (d_1 and a_3) xor (a_0 and d_1) xor (a_1 and d_0) xor (a_0 and d_0) xor (b_0 and d_2) xor (b_2 and d_0) xor (b_3 and d_3) xor (b_1 and d_3)

xor (b_3 and d_1) xor (b_1 and d_1) xor (b_0 and d_1) xor (b_1 and d_0) xor (b_2 and d_3) xor (b_3 and d_2)

y_3 = (a_0 and d_1) xor (a_1 and d_0) xor (d_2 and a_2) xor (a_0 and d_0) xor (d_3 and a_3) xor (b_0 and d_3) xor (b_1 and d_2) xor (b_2 and d_1) xor (b_3 and d_0) xor (b_0 and d_2) xor (b_2 and d_0) xor (b_1 and d_1) xor (b_0 and d_1) xor (b_1 and d_0)

y_2 = (d_3 and a_1) xor (d_3 and a_0) xor (d_2 and a_1) xor (d_1 and a_3) xor (d_1 and a_2) xor (d_0 and a_3) xor (a_0 and d_0) xor (a_1 and d_1) xor (b_0 and d_3) xor (b_1 and d_2) xor (b_2 and d_1) xor (b_3 and d_0) xor (b_0 and d_2) xor (b_2 and d_0) xor (b_1 and d_1) xor (b_0 and d_1) xor (b_1 and d_0)

y_1 = (a_0 and d_1) xor (a_1 and d_0) xor (d_2 and a_2) xor (a_0 and d_0) xor (d_3 and a_3)

y_0 = (a_0 and d_2) xor (a_2 and d_0) xor (d_3 and a_1) xor (d_1 and a_3) xor (a_0 and d_1) xor (a_1 and d_0) xor (a_0 and d_0) xor (b_2 and d_2) xor (b_0 and d_2) xor (b_2 and d_0) xor (b_1 and d_3) xor (b_3 and d_1) xor (b_0 and d_0) xor (b_1 and d_1) xor (b_2 and d_3) xor (b_3 and d_2)

C-1. Stage 1 of Inverter over the AES Field

Same as the stage 1 of the S-Box in A-1.

C-2. Stage 2 of Inverter over the AES Field

Same as the stage 2 of the S-Box in A-2.

C-3. Stage 3 of Inverter over the AES Field

Same as the stage 3 of the S-Box^{-1} in B-3.