# Optimized Composite Field Based Hardware Architectures for AES S-box using Logic Decomposition Techniques

Ruby Mishra[0000−0002−7331−281X], Manish Okade[0000−0003−1500−2693], and Kamalakanta Mahapatra[0000−0003−4917−7088]

Department of Electronics and Communication Engineering,
National Institute of Technology, Rourkela, Odisha, India.
rubymishrabgr1@gmail.com, okadem@nitrkl.ac.in, kkm@nitrkl.ac.in

**Abstract.** This paper proposes optimized architectures for AES substitution boxes using functional decomposition techniques. Functional decomposition techniques are logic synthesis approaches useful in reducing the support size of complex Boolean functions with more literals. The functional decomposition techniques in this work are applied to the multiplicative inverse function of the AES S-box constructed using sub-field arithmetic based on a normal basis. Three architectures are proposed; the first is based on single-variable decomposition, while the second and third are based on double-variable decomposition techniques. The proposed architectures exhibit high throughput and less area than AES's existing architectures. It is observed that the best of our proposed designs has a reduction of around 18% in the number of slices and nearly 50% in power consumption compared with the state-of-the-art architectures for the FPGA platform. Again, for standard cell libraries, our proposed design exhibits a delay reduction of around 41%, making them useful for resource-constrained applications.

**Keywords:** composite field · AES · logic synthesis · Shannon's expansion · Boolean decomposition · normal basis

## 1 Introduction

Hardware and software optimization and implementations are necessary for algorithms and functions tailored to a particular application. The designer chooses the parameter to be optimized, which is purely based on the application where the design must be applied. RAM size, execution time, latency, etc., should be determined for software implementations of a design. In contrast, area, delay, power, throughput, etc., should be evaluated for hardware implementations to understand how efficient a design is compared to others. Several software tools and computer-aided design tools aid in achieving the design implementations along with optimizing the required design parameters in the background. However, this optimization imparted by the tools may not always meet the specifications. Therefore, the designers follow several techniques to optimize the metrics,

either at the algorithmic level, logic level, circuit level, etc., to successfully attain the design functionality and standards. Targeting algorithms for hardware implementations, among several techniques, logic synthesis is a step in the VLSI Design flow, which converts a high-level code to a gate-level netlist. Usually, this step is carried out by Electronic Design Automation (EDA) tools. However, several logic synthesis approaches can be applied to the algorithms before feeding them to the tools. This improves the design mapping with the tool libraries and increases resource utilisation efficiency. Apart from this, the performance of the designs is enhanced significantly. Another aspect of logic optimization is that the techniques perform differently when the designs are aimed at ASIC or FPGA platforms; care has to be taken by the designers while selecting or proposing such techniques.

Functional decomposition or Boolean decomposition is one of the logic synthesis methods that can be applied to logic designs constructed solely by Boolean functions. Logic tree-based decomposition and implementation using majority gates designed using XOR logic is proposed in [4]. In [7], it has been demonstrated how the number of literals is reduced after the application of decomposition techniques. An automated software-based tool popularly called ABC is proposed in [2], which automatically minimizes the Boolean functions. Another approach for synthesizing logic functions for optimum use of resources is given in [6]. The re-substitution of logic functions method [14] is also one of the approaches by which the area occupied by design can be minimized. Numerous implementations of AES (Advanced Encryption Standard) [5] S-box in software and hardware platforms aim for different design and performance objectives. One of the compact and high-speed implementations for AES S-box using composite fields (using polynomial basis) was given by Satoh et al. [13]. Another architecture for AES S-box in [9] targeted low-power applications with good results. Canright et al. [3] proposed an area-optimized S-box using both polynomial and normal basis for composite field architectures. Again, S-box architecture was designed in [17] for high-speed applications. AES S-boxes redesigned for reducing the number of LUTs of FPGA resources were described in [11]. Recently, novel linear transformations [10] have been proposed for the AES S-box for the encryption and decryption process. This work focuses on logic synthesis techniques applied to the composite field-based architecture of AES S-box. Its implementations are evaluated both for FPGA and ASIC platforms with the motivation to optimize the design and performance parameters.

The remainder of the paper is organized as follows. Section 2 imparts preliminary ideas on composite field arithmetic applied to the S-box architectures. The motivation and key contribution of this work are highlighted in Section 3. The proposed logic decomposition-based S-box architectures are illustrated in Section 4. Section 5 explains the results of the proposed architectures compared with the existing designs. The conclusions are drawn in Section 6.

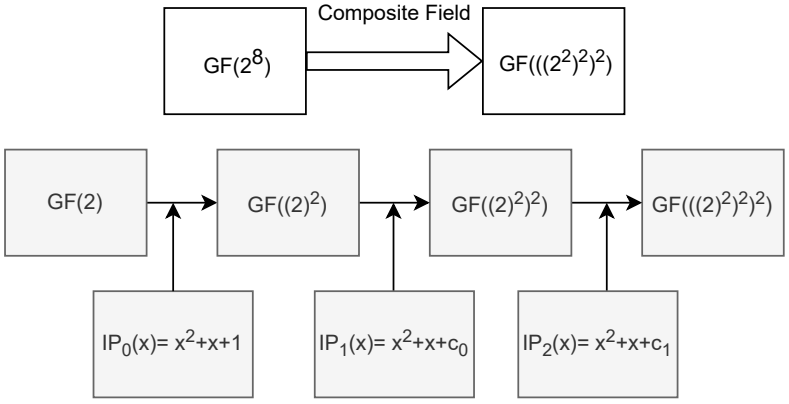



Fig. 1: Composite Fields for $GF(2^8)$.

## 2 Overview of Composite Fields

### 2.1 Finite Field Notations

A finite field, referred to as a Galois field ($GF$) [1], is included in field theory and has a finite number of elements. The Galois field is vital in cryptography since the data is represented here as a vector and can be permuted successfully. It is a set onto which mathematical operations, specifically binary operations, can be applied by predetermined rules. A finite field is said to be of order, $s = r^n$, where $r$ is a prime number and $n$ is supposed to be a positive integer number. For example, $GF(2)$ represents either a '0' or '1' element, whereas $GF(2^8)$ comprises '8' binary elements. So the generic representation of finite field can be denoted as $GF(2^n)$.

### 2.2 Composite Field Arithmetic applied to AES S-box Architectures

Data encryption in AES is carried out on blocks of bytes using $GF(2^8)$ representation with a particular polynomial as described in [3], [15], [16]. The I/O's of the AES S-box are 8-bit. The S-box function is applied to an 8-bit input data, and this substitution function has two sub-operations, i.e., the Multiplicative Inverse (MI) of the 8-bit input and then the Affine Transformation, i.e. mathematically, the S-box output function [16] is given by

$$f = Mf^{-1} \oplus b \tag{1}$$

where '$M$' is the binary matrix of dimension '$8 \times 8$', $f^{-1}$, is the MI of the 8-bit input, and '$b$' is an 8-bit array whose combined operation gives the S-box output. The MI unit may be effortlessly implemented using look-up tables (LUTs). However, the LUT method of implementation on hardware consumes more area. Therefore, composite arithmetic is used for compact implementations

of the S-box. The composite field for $GF(2^8)$ can be obtained as $GF((2^4)^2)$ or as $GF(((2^2)^2)^2)$, where $GF((2^4)$ and $GF((2^2)^2)$ are the subfields. These composite fields are produced iteratively from the lower-order fields using certain irreducible polynomials. Following are the equations Eq. (2-4) representing the construction of composite fields using the subfields [16]:

$$GF(2) \Rightarrow GF(2^2) \tag{2}$$

$$GF(2^2) \Rightarrow GF((2^2)^2) \tag{3}$$

$$GF((2^2)^2)) \Rightarrow GF(((2^2)^2)^2) \tag{4}$$

A general block diagram of this approach is illustrated in Fig. 1. In the figure, $IP_0$, $IP_1$, and $IP_2$ are the polynomials which are validated to be irreducible over the respective subfields depending on the coefficients '1', '$c_0$' and '$c_1$'. The details of the construction of these fields can be referred from [3] and [16]. This paper focuses on implementing MI in $GF(2^8)$ using normal basis. The MI module in $GF(2^8)$ consists of other submodules: field additions, multiplications, inverse, and modulo-2 additions that are XOR operations in the subfields. The modules follow the same hierarchy as in Fig. 1, i.e., $GF(2^8)$ can be decomposed to $GF(2^4)$ then to $GF(2^2)^2)$ and finally to $GF(2)$ which will serve as the basic module to execute all the operations required to obtain the MI. The substitution boxes of AES or any other encryption algorithm can be implemented via subfield arithmetic [3] and [13].

## 3 Motivation and Key Contributions

The conventional encryption algorithm AES (Advanced Encryption Standard) [5] is a symmetric block cipher that meets several applications' security and privacy requirements for generalized applications. However, the size of the input plaintext (128 bits) and the key (128/192/256 bits) is larger, affecting the architecture's area, power, and delay when targeted for resource-constrained applications. Therefore, there is a requirement to optimize these parameters, which is the primary motivation of the work reported in this paper. AES encryption comprises four operations: substituting input with corresponding bytes in the S-box (SubBytes), shifting of rows (ShiftRows), mixing of columns (MixColumns), and generation/addition of round keys (AddRoundKey) [16]. Among these stated operations, the byte substitution using S-boxes and inverse S-boxes are non-linear. They play a vital role in the implementation of AES, along with determining the security and performance of the architecture. The remaining operations are linear and can be implemented with fewer resources when hardware implementation is considered. Therefore, this work focuses on the substitution unit in detail, which is modified due to its higher resource utilization among the other functional units.

1. In this work, we propose efficient architectures based on functional decomposition approaches to the AES S-box constructed using subfield arithmetic.

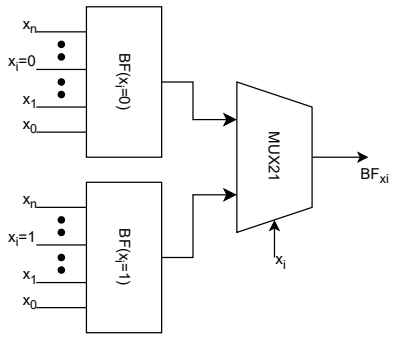



Fig. 2: Architectural Representation of Logic decomposition

To be specific, we modify the multiplier module and the multiply and scale module of the Multiplicative inverse function of the AES S-box in the $GF(2^2)$ subfield to achieve the field multiplications in $GF(2^8)$.

2. We have also evaluated and compared the proposed S-boxes with the existing architectures by constraining the design to operate at RFID frequency, i.e., 13.56 MHz on the FPGA platform, which enables the design to be compatible with communication and IoT applications. Such an application-based study, to the best of our knowledge, has not been carried out before. We have also evaluated the designs using standard cell libraries for the ASIC platform.

## 4 Logic Decomposition of AES Boolean Functions and Proposed Architectures

### 4.1 Logic Decomposition of AES Boolean Functions

Boolean functions having more variables and a large number of literals can be simplified by logic decomposition. This is achieved by the Shannon decomposition theorem, which helps represent a long Boolean function in terms of small functions, a literal, or a bit. Let us consider a Boolean function with '$n$' input variables, i.e., $BF(x_n, ..x_i, ..x_1, x_0)$ and then consider one of the variables, $x_i$ as constant. The resulting function can be expressed as

$$BF_{x_i} = x_i.BF(x_i = 1) + (x_i)'.BF(x_i = 0) \tag{5}$$

Here, the logic functions obtained for $x_i = 1$ are called positive cofactors, and those obtained for $x_i = 0$ are called negative cofactors. These cofactors are independent of the variable $x_i$. A common architecture for this type of decomposition is shown in Fig. 2. It is observed that the constant variable(s) serve as the select lines to the multiplexer structure. The variable to be kept fixed is decided according to the requirement and can be single or multiple.

In this work, we have chosen the composite field architecture using a normal basis for designing the AES S-box. The choice of the normal basis rather than

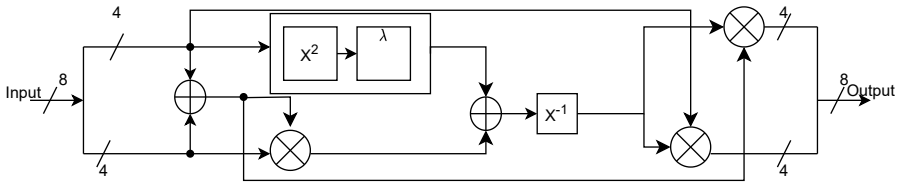
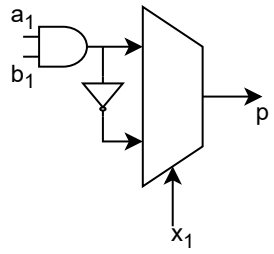



Fig. 3: Block Diagram for Multiplicative Inverse using Normal Basis

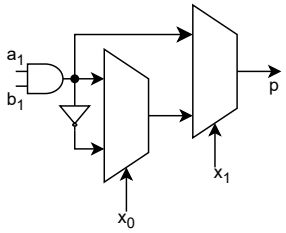
**Algorithm 1** Proposed Algorithm for Logic Decomposition of Boolean Functions

**Require:** ($x_i$ ). //Input Variables
**Ensure:** ( $BF_{x_i}$).//Output Boolean Functions
1: $n_l$= literal count
2: Assign $x_m = f_v$ //$x_m \epsilon x_i$, *fixed variable which can be* >= 1 *depending on SVD/DVD*
3: **for** All $i$ **do**
4: Assign $f_v$ = '0'
5: Compute $BF_{x_i'}$//*Substituting* $x_m$=*'0'*
6: Assign $f_v$ = '1'
7: Compute $BF_{x_i}$//*Substituting* $x_m$=*'1'*
8: **end for**
9: Repeat till $n_l = 0/1/2$ or $< i$
10: $BF_{x_i}$ is optimized.

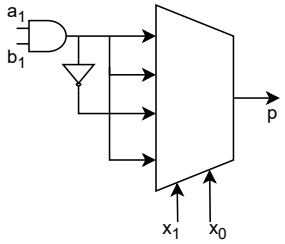
polynomial or mixed basis stems from the fact that the obtained architectures have compact representation due to more shared factors [3]. The overall architecture of the normal basis architecture used for Multiplicative Inverse (MI) implementation is shown in Fig. 3. The MI unit is interfaced with input and output transformation matrices to obtain the complete AES S-box. The input and output bits to this block diagram are 8 bits, which are further split into 4 bits and fed as input to the subsequent modules. The $\oplus$ and $\otimes$ denote the field additions and multiplications, respectively. In Galois field arithmetic, $\oplus$ are generally modulo-2 additions, which are XOR operations. The squaring, scaling, and inverter units are the $X^2$, $\lambda$, and $X^{-1}$. The proposed architectures are based on the algorithm proposed in Algorithm 1, which can be applied to any design utilizing Boolean functions. The technique proposed is utilized for the multiplier unit and the multiply and scale unit in $GF(2^2)$ of the complete architecture shown above. In contrast, the other blocks need less hardware, like one or two gates or only wiring, so the other blocks were implemented as it is. The Boolean functions are derived for the multiplier in $GF(2^2)$ using the Canright architectures [3]. This block $M_2$ has four inputs $a$, $b$, $x_0$, $x_1$, and one output $y$, where $a$, $b$, and $y$ are of two bits split as $a_1, a_0, b_1, b_0$, $p$ and $q$, respectively. While the other inputs $x_0$ and $x_1$ are 1-bit each. Eq. (6 - 7) are the equations for the multiplier block, which will be further utilized for decomposition steps and implementation in the overall architecture of the S-box.




(a) Architecture for $P_{svd}$




(b) Architecture for $P_{dvdM21}$




(c) Architecture for $P_{dvdM41}$

Fig. 4: Proposed $GF(2^2)$ Multiplier Architecture using Logic Decomposition

$$p = (a_1' + b_1') \oplus (x_1' + x_0') \tag{6}$$

$$q = (a_0' + b_0') \oplus (x_1' + x_0') \tag{7}$$

### 4.2 Proposed Architecture using Single Variable Decomposition (SVD), $P_{svd}$

In the proposed architecture, we first decompose the equation Eq. (6) using a single variable, i.e. $x_1$, since it is considered to be the MSB (Most Significant Bit). The MSB can be selected as the fixed variable for a set of more variables. So the reduced Boolean functions are derived as given in Eq. (8) and (9). The functions $p_{x_1'}$ and $p_{x_1}$ are the negative ($x_1 = 0$) and positive ($x_1 = 1$) cofactors which, when combined, give the total function as shown in Eq. (5). As a result of the decomposition, the number of functional input literals is reduced. While designing the architecture for this Boolean function, we can observe that the complement of Eq. (8) gives the first minterm of Eq. (9) that is shown in the architecture of Fig. 4a. The architecture utilizes one 2:1 MUX to realize the cofactor functions, one AND gate and a NOT gate. This reduces the number of AND gates required when counted for the complete design. Similarly, the

decomposed functions can be derived for $q$ where only $a_1$ is replaced by $a_0$. After determining $p$ and $q$, we concatenate the two bits to obtain the final output bit $y$ of the multiplier.

$$p_{x_1'} = (a_1' + b_1') \oplus (1 + x_0) = (a_1' + b_1') \oplus 1 = a_1.b_1 \tag{8}$$

$$p_{x_1} = (a_1' + b_1') \oplus (0 + x_0) = (a_1' + b_1') \oplus x_0' = (a_1.b_1)' \oplus x_0' \tag{9}$$

### 4.3 Proposed Architecture using Double Variable Decomposition (DVD), $P_{dvd}$

The approach here extends the idea of the previous $P_{svd}$ design by further decomposing Eq. (8) and (9) with two variables. In this case, $x_1$ and $x_0$ will be the fixed variables, but in any other complex Boolean functions, the first two MSB bits can be considered as the constant variables. The equations obtained after decomposition are depicted in Eq. (8-9). We also decompose the Boolean equations for obtaining the cofactors for $q$ and have designed their architectures. The architectures of these functions can be designed by either using only 2:1 MUX (MUX21) denoted as $P_{dvdM21}$ or by utilizing only 4:1 MUX (MUX41) denoted as $P_{dvdM41}$. This exploration aims to establish the unique utility of these MUXes for various requirements. Mapping these multiplexers with the FPGA resources varies accordingly, with varying results discussed in the next section.

$$p_{x_1'x_0'} = a_1.b_1 \tag{10}$$

$$p_{x_1'x_0} = a_1.b_1 \tag{11}$$

$$p_{x_1x_0'} = a_1.b_1 \tag{12}$$

$$p_{x_1x_0} = (a_1.b_1)' \tag{13}$$

The above architectures and Boolean equations were discussed for the multiplier unit of the overall architecture of the MI. We have also similarly designed the multiply and scale unit but have not illustrated it here in detail since it has equivalent architectures to the multiplier. The only difference is that the input variables are shuffled and rearranged. We have then included both the multiplier ($M_2$) and scaler ($M\lambda$) in $GF(2^2)$ subfield, which are further combined with the squarer ($X^2$) unit to build the inverter ($I_4$) and multiplier ($M_4$) unit in $GF(2^4)$ as shown in Fig. 5, where the complete hierarchy of the multiplicative inverse in $GF(2^8)$ as $I_8$ is illustrated. Then $I_4$, $M_4$, and the transformation matrices, designed using multiplexers, are combined to obtain the complete AES S-box for composite field architectures using a normal basis. The modified MI module, which plays a significant role in the design and performance aspects, is then substituted in the overall S-box design of the AES algorithm.

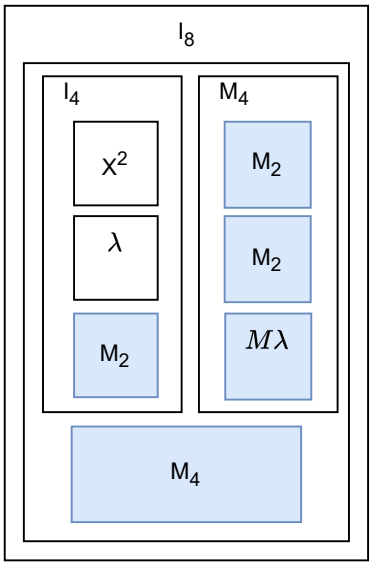



Fig. 5: Modules Hierarchy for Multiplicative Inverse of AES S-box in $GF(2^8)$

## 5 Results and Discussion

The architectures described above and the complete AES S-box are designed using Verilog HDL. Initially, to show the efficiency of the proposed architectures, we evaluated the multiplier module for SAED 90nm standard cell libraries simulated using the Synopsys Design Vision tool. The synthesis results are shown in Table. 1. It can be observed that the area evaluated in terms of gate equivalents (GEs) and power is more for our proposed designs due to the use of multiplexers. But our proposed architectures $P_{dvdM21}$ and $P_{dvdM41}$ exhibit minimized delay of about 41 % reduction, thereby increasing the throughput of the design. The delay referred to here is the critical path delay (CPD) of the design, which plays a significant role in the performance of the design, affecting both the energy and throughput performance parameters. In contrast, the architectures in [3] are designed using only logic gates. We observe similar metrics when calculated for the complete AES S-box when evaluated for 180nm standard libraries. Architectures designed in [8] and in [12] have fewer gate equivalents when compared to our proposed designs but have considerably high critical path delay. The best case for $P_{dvdM41}$ has a delay reduction of around 69% and 96% when compared to [12] and [8], respectively. Also, they have not evaluated the power consumption of their designs, which is an important performance parameter for resource-constrained applications. Again, the CPD is reduced by 5 % for $P_{svd}$ and 7 % for $P_{dvdM41}$ when compared with [3] due to the utilization of multiplexers, but $P_{dvdM21}$ gives similar CPD due to two-stage multiplexer architecture.

We then evaluated the complete AES S-box on the Nexys4 DDR FPGA platform, illustrating the design parameters, area of LUTs and slices, critical delay,

Table 1: Synthesis Results of Multiplier Architectures in $GF(2^2)$for 90nm Standard Cell Libraries

| Design | Area (GE) | Power (mW) | CPD (ns) |
|---|---|---|---|
| Canright et.al., [3] | 7.95 | 0.08 | 0.12 |
| $P_{svd}$ | 11.9 | 0.1 | 0.12 |
| $P_{dvdM21}$ | 15.96 | 0.2 | 0.08 |
| $P_{dvdM41}$ | 15.9 | 0.1 | 0.07 |

Table 2: Synthesis Results of AES S-box for 180nm Standard Cell Libraries

| Design | Area (GE) | Power (mW) | CPD (ns) |
|---|---|---|---|
| Canright et al., [3] | 639 | 2.84 | 0.43 |
| Mentens et al., [8] | 272 | - | 10 |
| Rashidi et al., [12] | 211 | - | 1.28 |
| $P_{svd}$ | 705 | 3.09 | 0.41 |
| $P_{dvdM21}$ | 694 | 3.02 | 0.43 |
| $P_{dvdM41}$ | 745 | 3.46 | 0.40 |

and power. This is shown in Table 3. The dynamic power is calculated only after constraining the S-box design with a 13.56 MHz clock frequency. This frequency is chosen because it is the standard frequency applicable to RFID applications in IoT environments. All the parameters were obtained by simulating the design in the Vivado tool, and the results were noted after the Place and Route step of the design implementation. It is observed that the number of slices of our proposed designs is reduced by around 18 % due to the efficient mapping of the multiplexers with the FPGA resources as compared to the logic gates-based design of [3]. The dynamic power consumption has been reduced by around 50 %. Also, the delay has been minimized by an average reduction of around 10 %, thereby increasing the maximum frequency of operation by 10 %, 6 %, and 19 % of $P_{svd}$, $P_{dvdM21}$, and $P_{dvdM41}$, respectively when compared to the state-of-the-art designs. This results in increasing the throughput of the proposed designs as well.

Table 3: AES S-box using Composite Field Architectures on Nexys4 DDR FPGA

| Design | LUT (#) | Slices (#) | Power (W) | Delay (ns) | Fmax (MHz) |
|---|---|---|---|---|---|
| Canright et al., [3] | 55 | 16 | 0.002 | 14.81 | 68 |
| $P_{svd}$ | 40 | 13 | 0.001 | 13.23 | 75 |
| $P_{dvdM21}$ | 48 | 13 | 0.001 | 13.86 | 72 |
| $P_{dvdM41}$ | 49 | 14 | 0.001 | 12.35 | 81 |

## 6 Conclusions

This work concludes with a note that the logic decomposition-based architecture for composite field S-box is unique, with the possibility of optimising more than one design metric. These techniques can be applied to any logic design utilizing complex Boolean functions. The proposed techniques reduce the number of literals in the Boolean equations used to implement the AES S-box. The techniques were applied to the multiplier and multiply and scale units in $GF(2^2)$ subfield and then substituted in the overall multiplicative inverse unit in $GF(2^8)$ and then in the complete AES S-box of Canright architecture using normal basis. Our proposed designs were designed using multiplexers, proving them best for FPGA-based applications, where the design metrics are optimized to their best. Also, the designs are suitable for standard cell libraries in ASIC platforms if high-speed applications are targeted. The proposed designs can be selectively chosen depending on the requirement of the target applications and the availability of the resources both for FPGA tools and standard cell libraries.

## References

1. Benvenuto, C.J.: Galois field in cryptography. University of Washington **1**(1), 1–11 (2012)
2. Brayton, R., Mishchenko, A.: Abc: An Academic Industrial-Strength Verification Tool. In: International Conference on Computer Aided Verification. pp. 24–40. Springer (2010)
3. Canright, D.: A Very Compact S-box for AES. In: Proceedings of the 7th International Conference on Cryptographic Hardware and Embedded Systems. p. 441–455. CHES'05, Springer-Verlag, Berlin, Heidelberg (2005)
4. Chu, Z., Soeken, M., Xia, Y., Wang, L., De Micheli, G.: Advanced Functional Decomposition Using Majority and Its Applications. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **39**(8), 1621–1634 (2019)
5. Dworkin, M., Barker, E., Nechvatal, J., Foti, J., Bassham, L., Roback, E., Dray, J.: Advanced Encryption Standard (AES) (2001-11-26 2001). https://doi.org/https://doi.org/10.6028/NIST.FIPS.197

6. Legl, C., Wurth, B., Eckl, K.: Computing Support−Minimal Subfunctions During Functional Decomposition. IEEE Transactions on Very Large Scale Integration (VLSI) systems **6**(3), 354–363 (1998)
7. Machado, L., Cortadella, J.: Support−Reducing Decomposition for FPGA Mapping. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **39**(1), 213–224 (2018)
8. Mentens, N., Batina, L., Preneel, B., Verbauwhede, I.: A systematic evaluation of compact hardware implementations for the Rijndael S-box. In: Topics in Cryptology–CT-RSA 2005: The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005. Proceedings. pp. 323–333. Springer (2005)
9. Morioka, S., Satoh, A.: An optimized S-Box circuit architecture for low power AES design. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 172–186. Springer (2002)
10. Nakashima, A., Ueno, R., Homma, N.: AES S-Box Hardware with Efficiency Improvement Based on Linear Mapping Optimization. IEEE Transactions on Circuits and Systems II: Express Briefs **69**(10), 3978–3982 (2022). https://doi.org/10.1109/TCSII.2022.3185632
11. Pradeep, A., Mohanty, V., Subramaniam, A.M., Rebeiro, C.: Revisiting AES SBox Composite Field Implementations for FPGAs. IEEE Embedded Systems Letters **11**(3), 85–88 (2019). https://doi.org/10.1109/LES.2019.2899113
12. Rashidi, B.: Compact and efficient structure of 8-bit S-box for lightweight cryptography. Integration **76**, 172–182 (2021)
13. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-Box Optimization. In: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology. p. 239–254. ASIACRYPT '01, Springer-Verlag, Berlin, Heidelberg (2001)
14. Sawada, H., Suyama, T., Nagoya, A.: Logic Synthesis for Look−Up Table based FPGAs using Functional Decomposition and Support Minimization. In: Proceedings of IEEE International Conference on Computer Aided Design (ICCAD). pp. 353–358. IEEE (1995)
15. Wong, M.M., Wong, M.D., Nandi, A.K., Hijazin, I.: Construction of optimum composite field architecture for compact high-throughput AES S-boxes. IEEE transactions on very large scale integration (VLSI) systems **20**(6), 1151–1155 (2011)
16. Zhang, X., Parhi, K.: On the Optimum Constructions of Composite Field for the AES Algorithm. IEEE Transactions on Circuits and Systems II: Express Briefs **53**(10), 1153–1157 (2006)
17. Zhang, X., Wu, N., Zhou, F., Ge, F.: Optimization of Area and Delay for Implementation of the Composite Field Advanced Encryption Standard S-Box. Journal of Circuits, Systems and Computers **25**(05), 1650037 (2016)