Resource efficient Implementation of Unified S-box Hardware for AES Algorithm using Composite Field Arithmetic

Raghavendra Mahalatkar B S, Dr Menka Yadav,
Department of Electronics and Communication,
Malaviya National Institute of Technology, Jaipur-302017, India
raghavendra.abd36@gmail.com, menka.ece@mnit.ac.in

***Abstract*—This paper introduces a resource efficient method for implementing the unified hardware for Substitution box for AES encryption and decryption by exploiting the concept of unique mapping between isomorphic fields, followed by combinational logic optimization in between at pre and post processing stages , by choosing the appropriate isomorphic and inverse isomorphic matrix described in the paper through which one can easily map element between GF(2^8) to GF((2^4)^2), compared to classical lookup-table based S-box this method provide better performance in-terms of Lookup-table count consumption and resource utilization with low power, Both design styles are synthesized and implemented for Artix-7 xa7a35tcpg236-1 series FPGA using Vivado tool, This architecture has better hardware utilization with 62 LUTs for FPGA implementation with total power consumption of 10.514 Watt which include static and dynamic.**

I. INTRODUCTION

National Institute of Standards and Technology replaced conventional Data encryption standard algorithm by Advanced encryption standard, it involves Rijndael algorithm, this was chosen finally for Federal Information processing Standard[1,2,9,10], Due to the evolvement in Digital communication technology and IoT application, There is need of robust cryptographic algorithm for digital data transmission over unsecured communication channels, Hardware based method is more suitable than software based methods for cryptographic implementations because of several advantages[3], S-box is the major hardware component in AES whose hardware architecture decides the overall performance of AES cipher interms of resource consumption for FPGA implementation, one approach is by LUTs and full combinational logic which is inefficient[4,6], [5,7,8] explains the unified hardware architecture for implementing the S-box using multiplexers with inversion circuit, calculating the S-box directly requires significant amount of hardware. Once it can be converted to its isomorphic field and composite field impacts the overall hardware requirement for inversion hardware significantly lesser than that of the existing counterparts, mapping between isomorphic fields can be explained in[11,12], the hardware is optimally designed by choosing the constants of the Galois field, [13,14] suggests the best choices for choosing the constants for optimal Hardware implementation of inversion in GF((2^4)^2) logic.

II. OVERVIEW OF ADVANCED ENCRYPTION STANDARD

The AES is the most widely used block cipher and symmetric-key cryptographic algorithm standardized in 2001. it supports key size of 128,192,256, 512 and so on as per the security requirement, all the AES variants operate on the plain text of size 128-bits. AES 128 , 192, 256 has total 10,12,14 rounds of data transformation respectively, each round except last round has 4 intermediate rounds ie Sub-bytes,Shift rows,Mix-column, Add round key but last round don't have mix-column operation the overall flow of AES cipher for encryption and decryption is shown in figure 1.

A. Sub Bytes and Inverse Sub Bytes

This is the first step in AES algorithm, here Byte by Byte substitution is made for the corresponding input 128-bit input element. This will add non-linearity to the algorithm making it resistant against crypt-analytic attacks through Substitutiopn Box(S-box), This step is mathematically visualized as inversion of a byte ie an element in $GF(2^8)$ under irreducible polynomial $P(x)=x^8+x^4+x^3+x+1$ followed by affine transformation and inverse affine transformation for the sub-byte and inverse sub-byte respectively.

B. Shift Rows and Inverse Shift Rows

Shift Rows operation is byte-wise shifting of the individual bytes of the state array, first rows remain unaffected, but 2nd ,3rd and 4th rows gets shifted by 2 , 3 , 4 positions cyclic left-shift during encryption, right shift during decryption respectively.

C. Mix- columns and Inverse Mix-columns

This step adds diffusion to the cipher text , each column is considered as 4-term polynomial $b(x)=b_3x^3+ b_2x^2+b_1x+b_0$ which are elements within the field $GF(2^8)$, The coefficients of the polynomial are elements within the prime sub-field $GF(2)$, Each column is multiplied with a fixed polynomial $a(x)=3x^3+x^2+x+2$ modulo x^4+1 and inverse of this polynomial $a^{-1}(x)=11x^3+13x^2+9x+14$ respectively.

D. Add-Round key

The input byte is transformed by Secret Key through bit-wise addition in finite fields (xored) with the state array at every round, both during encryption and decryption.

E. Key Expansion

AES takes a single key for the first round then passed through the key expansion algorithm to generate the sub-keys for the subsequent rounds.

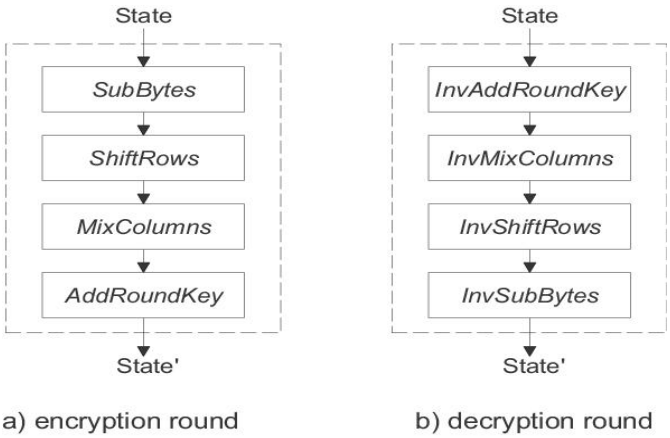

Figure 1. Complete AES flow for encryption and decryption

III. Binary Finite Field And Composite Field Arithmetic

A. Binary Finite Fields

Any polynomial $a(x)=a_m x^m+a_{m-1}x^{m-1}+\ldots\ldots+a_{m-1}$ is considered as a polynomial over Galois field, ie $GF(2^m)$ all the coefficients of polynomial are Binary in nature, since we operate on a binary fields of 1 byte of data in AES we should consider it as $GF(2^8)$ under some primitive polynomial or irreducible polynomial $x^8+x^4+x^3+x+1$(not factorizable) , any element in the $GF(2^8)$ can be generated by linear combination of the roots of this generator polynomial.

B. Composite fields and Isomorphic fields

Any element in $GF(2^8)$ can be uniquely mapped between its corresponding Isomorphic field by using the concept of Isomorphism and Composite field. This mathematical tool is advantageous for efficient Hardware implementation in AES cipher, The pair of fields $GF(2^n)$ and $GF(2^n)^m$ is said to be composite in nature if there exist a Irreducible polynomials Q(y) of degree 'n' and P(x) of degree 'm' used to extend an element from $GF(2)$ to $GF(2^n)$ and $GF(2^n)$ to $GF(2^n)^m$ respectively.

The composite field $GF(2^n)^m$ is isomorphic to the field $GF(2^k)$ where k=m×n, where arithmetic in the field $GF(2^k)$ can be done by using the primitive polynomial R(z) of degree 'k', where $R(z)=z^k+r_{k-1}z^{k-1}+r_{k-2}z^{k-2}+\ldots\ldots+1$, $\forall r_i \in GF(2)$, let γ be the root of R(z) then, all elements in $GF(2^k)$ can be expressed as a linear combination of the polynomial bases $(1,\gamma,\gamma^2\ldots\ldots\gamma^{k-1})$.

Finally Let R(z),Q(y),P(x) be the primitive polynomials in the fields $GF(2^k)$, $GF(2^n)$, $GF(2^n)^m$ respectively , let the element γ mapped α on its corresponding Isomorphic field if it satisfies the relation , $R(\gamma)=0$ and $R(\alpha)\ mod(P(x)Q(y))=0$.

C. Arithmetic in Composite fields for constructing S-box

Mathematically S-box for an AES is multiplicative inversion over $GF(2^8)$ followed by affine transformation which can be expressed as

$$S\text{-}box(x)=MS^{-1}+C \qquad (1)$$

where M is an 8×8 binary matrix and C is a binary column matric of order 8×1 , computing the multiplicative inversion for an element in $GF(2^8)$ consume significant amount of hardware and hence power. Upon converting this element into its corresponding composite fields it can be possible to implement it efficiently, Mapping from $GF(2)$ to $GF(2^2)$, $GF(2^2)$ to $GF((2^2)^2)$ and $GF((2^2)^2)$ to $GF(((2^2)^2)^2)$ can be done by defining the irreducible polynomials $Q_0(x)=x^2+x+1$, $Q_1(x)=x^2+x+\Phi$ and $Q_2(x)=x^2+x+\lambda$ respectively the constants Φ and λ are 2-bit, 4-bit respectively and must be chosen such that the polynomial remain irreducible out of which 4-combination of Φ and λ is optimum from hardware implementation point-of-view they are given in equation 2

$$\begin{aligned}&\Phi=\{10\}\ \lambda=\{1100\},\\ &\Phi=\{11\}\ \lambda=\{1000\},\\ &\Phi=\{10\}\ \lambda=\{1111\}\\ &\Phi=\{11\}\ \lambda=\{1010\}\end{aligned} \qquad (2)$$

The [14] suggests that choosing $\Phi=\{10\}\ \lambda=\{1100\}$ will give optimum hardware implementation for isomorphic mapping.

IV. Hardware Architecture And Mathematical Analysis

A. Hardware Architecture for unified S-box

The unified hardware architecture for S-box and inverse S-box is described in figure 2 below it has two 2:1 multiplexer, first mux will route one of the inputs from preprocessing module to the inversion in $GF((2^2)^2)$ hardware, second mux at the last will route one of the input from the post-processing module to the final output . The select lines for both muxes are controlled by the mode of operation which is S-box or inverse S-box, The inversion in $GF((2^2)^2)$ modules has multiple sub-modules which are $GF((2^2)^2)$ multiplication module, inversion module, squaring and multiplication with constant etc.. as described below in Fig 2.

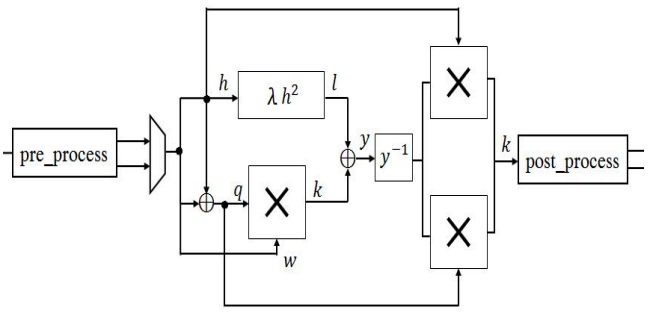

Figure 2: Unified Hardware architecture using Composite $GF(2^8)$ to $GF((2^4)^2)$ Field Arithmetic for S-box implementation

The above hardware is derived based on the mathematical identity for the inverse of the element in $GF(2^8)$, Any

element/Byte in GF(2^8) can be splitted as higher and lower nibble as menctioned in … and can be written as

$$(bx+c)^{-1}=b(b^2\lambda+c(b+c))^{-1}x+(c+b)(b^2\lambda+c(b+c))^{-1} \quad (3)$$

From the above equation 1-byte of message can be splitted to two-nibbles all the arithmetic operations involved for finding the inversion can be done for this splitted nibbles. The multiplication and addition operation can be done for these nibbles under $GF(2^4)$ and $GF((2^2)^2)$ arithmetic.

B. Multiplication in GF((2²)²)

Let k be the 4-bit product generated by multiplying two elements in $GF((2^2)^2)$ ie $'q'(=q_hx+q_l)$ and $'w'(=w_hx+w_l)$ under irreducible polynomial $x^2+x+\Phi$ with $\Phi=\{10\}$. The 4-bit vector $k=\{k_3k_2k_1k_0\}$ be the product of q and w, will be splited as $k_h=\{k_3k_2\}$ and $k_l=\{k_1k_0\}$ we can visualize the result k as

$$k=k_hx+k_l$$
$$k=qw=(q_h+q_l)(w_h+w_l) \quad (4)$$

after furthur reducing the term using the given irreducible polynomial we can write

$$k=qw=(q_h+q_l)(w_h+w_l)$$
$$k=(q_hw_h+q_hw_l+q_lw_h)x+q_hw_h\Phi+q_lw_l \quad (5)$$

upon considering the irreducible polynomial x^2+x+1 for $GF((2^2)^2)$ multiplication with $\Phi=\{10\}$ we can get finally the result 4-bit k in $GF((2^2)^2)$ multiplication as

$$k_3=(q_3+q_2)(w_3+w_2)+(q_3+q_2)(w_1+w_0)+(q_1+q_0)(w_3+w_2)+q_2w_0+q_0w_2+q_2w_2$$
$$k_2=q_3w_3+q_3w_1+q_1w_3+q_2w_2+q_2w_0 \quad (6)$$
$$k_1=(q_3+q_2)(w_3+w_2)+q_3w_3+(q_1+q_0)(w_1+w_0)+q_0w_0$$
$$k_0=(q_3+q_2)(w_3+w_2)+q_2w_2+q_1w_1+q_0w_0$$

The addition operation is bitwise xor in finite field arithmetic.There are terms $(q_3+q_2),(q_1+q_0),(w_3+w_2),(w_1+w_0),q_3w_3,q_2w_2,q_1w_1,q_0w_0$ and $(q_2w_0+q_0w_2)$ are needed to realize this 4-bit $GF((2^2)^2)$ multiplication can be reused for realizing the whole logic for this 4-bit multiplication.

C. Squaring and multiplication with constant λ in GF((2²)²)

The squaring the h followed by and multiplication with constant $\lambda=\{1100\}$ in $GF((2^2)^2)$ are unified and optimized to improve the area . Let h be an element in $GF((2^2)^2)$, its squaring and multiplication with constant λ with irreducible polynomial x^2+x+1 can be written as sh0wn in equation 7 and 8 and is given in figure 3

$$l=\lambda h^2$$
$$l=(\lambda_Hx+\lambda_L)(h_Hx+h_L)(h_Hx+h_L) \quad (7)$$
$$=(\lambda_Hh^2_H+\lambda_Hh^2_H\Phi+\lambda_Hh^2_L+\lambda_Lh^2_H)x+(\lambda_Hh^2_H\Phi+\lambda_Lh^2\Phi+\lambda_Lh^2_L)$$

Where

$$l_H=(\lambda_Hh^2_H+\lambda_Hh^2_H\Phi+\lambda_Hh^2_L+\lambda_Lh^2_H)$$
$$l_l=(\lambda_Hh^2_H\Phi+\lambda_Lh^2_H\Phi+\lambda_Lh^2_L)$$

finally we get

$$l_3=(h_2+h_1+h_0)$$
$$l_2=(h_3+h_0) \quad (8)$$
$$l_1=h_3$$
$$l_0=h_2+h_3$$

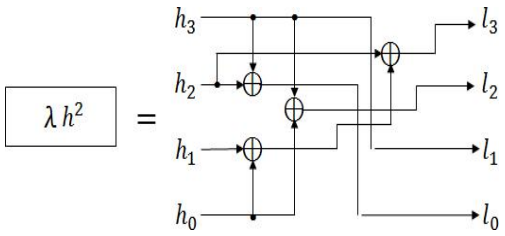

Figure 3: Squaring and Multiplication with constant λ

This approach needs only 4 -xor gates with 2-xor gates in the critical path.

D. Multiplicative inversion in GF((2²)²)

Ready-made truth table for each bit of multiplicative inversion for an element in $GF((2^2)^2)$ is given below, based on which boolean logic equations are developed the logic expression for individual bits are given in equation 9.

$y_3y_2y_1y_0$	$(y_3y_2y_1y_0)^{-1}$
0000	0000
0001	0001
0010	0011
0011	0010
0100	1111
0101	1100
0110	1001
0111	1011
1000	1010
1001	0110
1010	1000
1011	0111
1100	0101
1101	1110
1110	1101
1111	0100

Table 1. Truth table for inversion of element in GF((2²)²)

$$y_3^{-1}=\bar{y}_3y_2+y_2\bar{y}_1y_0+y_2y_1\bar{y}_0+y_3\bar{y}_2\bar{y}_0$$
$$y_2^{-1}=\bar{y}_3y_2\bar{y}_1+y_3y_2+y_3\bar{y}_2y_0 \quad (9)$$
$$y_1^{-1}=\bar{y}_3y_2\bar{y}_1\bar{y}_0+\bar{y}_3\bar{y}_2\bar{y}_1+y_3\bar{y}_1y_0+y_3\bar{y}_2y_0+\bar{y}_3\bar{y}_2y_1+\bar{y}_3y_1y_0$$
$$y_0^{-1}=\bar{y}_3y_1\bar{y}_0+\bar{y}_3y_2y_1+y_3\bar{y}_2y_1y_0+\bar{y}_3\bar{y}_2\bar{y}_1y_0+y_2\bar{y}_0$$

E. Design for Pre-processing and Post-processing Stages

At pre-processing stage during forward S-box calculation Isomeric mapping has to be done , during inverse S-box

calculation Inverse affine transform in cascaded with Isomeric transformation has to be done, but at the post processing stage inverse Isomeric mapping in cascaded with affine transform is done for forward S-box. For inverse S-box, the Inverse Isomeric mapping has to be done where some logic gates output is getting shared at both preprocessing and post processing stages, The matrix chosen for Isomeric, Inverse Isomeric is given below here addition is xor operation in finite field, Let w be the 8-bit input, t1 and t2 be the 8-bit intermediate outputs of preprocessing stage during S-box and inverse S-box calculation we can write the logical expressions for them as given below, logic equations r_1,r_2,r_3,r_4 are used for building the logic for other bits efficiently at Pre-processing Stage.

Pre-processing Design:

$$\begin{aligned} r_1&=w_7+w_3+w_2\\ r_2&=w_6+w_1\\ r_3&=w_7+w_2\\ r_4&=w_7+w_5 \end{aligned} \quad (10)$$

The matrix taken to calculate the Isomeric mapping is given below in equation 11 during S-box data path whereas for inverse S-box calculation, inverse affine transformation in cascaded with Isomorphic mapping is done.

$$\delta\times w=\begin{bmatrix}1&0&1&0&0&0&0&0\\1&1&0&1&1&1&1&0\\1&0&1&0&1&1&0&0\\1&0&1&0&1&1&1&0\\1&1&0&0&0&1&1&0\\1&0&0&1&1&1&1&0\\0&1&0&1&0&0&1&0\\0&1&0&0&0&0&1&1\end{bmatrix}\begin{bmatrix}w7\\w6\\w5\\w4\\w3\\w2\\w1\\w0\end{bmatrix} \quad (11)$$

Output of Isomeric mapping	Output of Inverse affine+Isomeric mapping
$t1_7=r_4$	$t2_7=r_2+r_3$
$t1_6=r_1+r_2+w_4$	$t2_6=r_1+r_2+w_0+1$
$t1_5=r_1+w_5$	$t2_5=w_6+w_5+w_4+w_0$
$t1_4=t1_5+w_1$	$t2_4=w_5+w_4+w_3$
$t1_3=r_3+r_2$	$t2_3=r_4+1$
$t1_2=r_1+w_4+w_1$	$t2_2=r_2+r_4+w_2+1$
$t1_1=r_2+w_4$	$t2_1=w_5+w_3+w_1$
$t1_0=w_0+r_2$	$t2_0=r_3+w_6+1$

Table 2. Truth table for Pre Processing logic

Post-processing Design:

Let u be the input and r and p be the intermediate outputs during S-box and inverse S-box calculation respectively with $t_7,t_6,t_5,t_4,t_3,t_2,t_1,t_0$ be the outputs used for building the logic for other bits efficiently one can write

$$\begin{aligned} t_8&=t_4+t_0\\ t_7&=u_7+u_0\\ t_6&=u_7+u_2\\ t_5&=u_7+u_6\\ t_4&=u_5+u_6\\ t_3&=u_5+u_4\\ t_2&=u_4+u_3\\ t_1&=u_2+u_1\\ t_0&=u_2+u_0 \end{aligned} \quad (11)$$

The matrix taken to calculate the inverse-Isomeric mapping is given below in equation 12.

$$\delta^{-1}\times u=\begin{bmatrix}1&1&1&0&0&0&1&0\\0&1&0&0&0&1&0&0\\0&1&1&0&0&0&1&0\\0&1&1&1&0&1&1&0\\0&0&1&1&1&1&1&0\\1&0&0&1&1&1&1&0\\0&0&1&1&0&0&0&0\\0&1&1&1&0&1&0&1\end{bmatrix}\begin{bmatrix}u7\\u6\\u5\\u4\\u3\\u2\\u1\\u0\end{bmatrix} \quad (12)$$

Output of inv-Isomeric mapping	Output of inverse-Isomeric mapping+affine mapping
$p_7=t_5+u_5+u_1$	$r_7=t_6+u_3$
$p_6=u_6+u_2$	$r_6=t_3+t_5+1$
$p_5=t_4+u_1$	$r_5=t_6+1$
$p_4=t_4+t_1+u_4$	$r_4=t_7+u_4+u_1$
$p_3=t_3+t_1+u_3$	$r_3=t_1+u_0$
$p_2=t_2+t_1+u_7$	$r_2=t_8+t_2$
$p_1=t_3$	$r_1=t_7+1$
$p_0=t_8+u_4$	$r_0=t_5+t_1+u_0+1$

Table 3. Truth table for Post Processing logic

Let r and p be the 8-bit intermediate outputs generated for calculating S-box and inverse S-box respectively shown in the above Table 3, inverse isomorphic mapping cascaded with affine transformation which is 8×8 matrix multiplication along with 1×8 vector addition can also be done at the post-processing stage itself.

V. HARDWARE IMPLEMENTATION AND RESULT ANALYSIS

Functional Simulation for the unified S-box and its Performance

The proposed hardware architecture for S-box works for both S-box and inverse S-box which is controlled by the multiplexer select line at pre-processing followed by inversion Hardware and finally post processing stages given in schematic in fig.4.

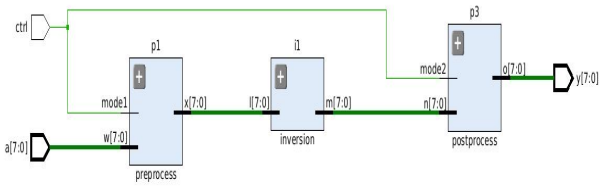

Figure 4: Schematic for the generated Unified S-box

Theoretical analysis of Composite field arithmetic is done and it is used for developing the RTL code for AES S-box hardware and is verified by giving the test vectors using xilinx vivado tool. The several design architectures for S-box is synthesized and implemented using Atrix-7-xa7a35tcpg236-1 FPGA, The performance of the three different S-box architecture is analyzed interms of resource consumption(LUT slices) and Power consumption is presented in Table below

Type of S-box Architecture	Number of LUT slices consumed	Static Power consumptio(Watt)	Switching Power Consumption(Watt)
Look up-table based approach	80	0.184	10.97
S-box with composite field Arithmetic	63	0.151	10.615
S-box with resource sharing	62	0.149	10.514

Table 4. Performance analysis of different S-box architectures for Atrix-7 FPGA implementation

Above table reveals that comparatively the S-box hardware with resource sharing approach can give better performance among three different unified S-box architectures.

VI. CONCLUSION

The new architecture for unified S-box and inverse S-box using Composite field arithmetic is developed along with mathematical analysis and all the individual sub-components which are Galois field $GF((2^2)^2)$, multiplier inversion in $GF((2^2)^2)$,squaring and multiplication with constant and pre and post-processing stages is optimized for efficient resource utilization. The above S-box architecture can be deployed for developing the actual AES hardware for varied key sizes(128, 192, 256) as per the need.

REFERENCES

[1] J. Daemen and V. Rijmen, The Design of Rijndael: AES—The Advanced Encryption Standard. New York, NY, USA: Springer, 2002.

[2] Specification for the Advanced Encryption Standard (AES), document FIPS PUB197, National Institute of Standards and Technology,Nov. 2001.

[3] A. Kchaou, W. E. H. Youssef, and R. Tourki, ‘‘Software implementation of AES algorithm on leon3 processor,’’ in Proc. 15th Int. Conf. Sci. Techn.

[4] R. Santhosh, R. Shashidhar, M. Mahalingaswamy, S. Praveen, and M. Roopa, ‘‘Design of high speed AES system for efficient data encryption and decryption system using FPGA,’’ in Proc. Int. Conf. Electr.,Electron., Commun., Comput., Optim. Techn. (ICEECCOT), Dec. 2018,pp. 1279–1282.

[5] A. Reyhani-Masoleh, M. Taha, and D. Ashmawy, ‘‘New low-area designs for the AES forward, inverse and combined S-Boxes,’’ IEEE Trans. Comput., vol. 69, no. 12, pp. 1757–1773, Dec. 2020.

[6] D. Canright, A Very Compact S-Box for AES, vol. 3659. Cham, Switzerland: Springer, 2005, pp. 441–455.

[7] Y. Liu, N. Wu, X. Zhang, F. Zhou, and F. Ge, ‘‘A compact implementation of AES S-box using evolutionary algorithm,’’ Chin. J. Electron., vol. 26, no. 4, pp. 688–695, Jul. 2017.

[8] S. S. Priya, K. G. Das, N. M. SivaMangai, and P. K. Kumar, ‘‘Multiplexer based high throughput S-box for AES application,’’ in Proc. 2nd Int. Conf. Electron. Commun. Syst. (ICECS), Feb. 2015, pp. 242–247.

[9] National Institute of Standards and Technology, FIPS PUB 46-3: Data Encryption Standard (DES). October 1999. supersedes FIPS 46-2.

[10] H.D. Zodpe, P.W. Wani, and R.R. Mehta, Design and implementation of algorithm for DES cryptanalysis, in 2012 12th International Conference on Hybrid Intelligent Systems (HIS),,pages 278–282, Dec 2012.

[11] R. Ueno, N. Homma, Y. Nogami, and T. Aoki,“Highly effiient $GF(2^8)$ inversion circuit based on hybrid GF representations,” J.Cryptographic Eng., vol. 9, no. 2, pp. 101–113, 2019.

[12] 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 Proc. IEEE 23nd Symp. Comput. Arithmetic, 2016, pp. 112–117.

[13] S. Sawataishi, R. Ueno, and N. Homma, ‘‘Unified hardware for highthroughput AES-based authenticated encryptions,’’ IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 67, no. 9, pp. 1604–1608, Sep. 2020.

[14] Gangadari, Bhoopal Rao, and Shaik Rafi Ahamed. "FPGA implementation of compact S-Box for AES algorithm using composite field arithmetic." In 2015 Annual IEEE India Conference (INDICON), pp. 1-5. IEEE, 2015.