LOW POWER IMPLEMENTATION OF OPTIMUM COMPOSITE FIELD ARCHITECTURE WITH MINIMAL AREA FOR HIGH-THROUGHPUT AES S-BOXES

Vasili Supraja and P. Mahesh Kannan, SRM university, kattankulathur

Abstract— In this work, we derive novel composite field arithmetic (CFA) Advanced Encryption Standard (AES) S-boxes of the field GF (\((2^8)^3\)). After a sequence of algorithmic and architectural optimization processes a best design is selected. The isomorphic mapping with minimal implementation area and cost functions chosen for implementation after the exploitation of new common sub expression elimination algorithm. Existing System Performs the 8-bit Galois Field inversion of the S-box using subfields of 4 bits and of 2 bits. This work describes a refinement of this approach that minimizes the circuitry, and hence the chip area, required for the S-box with high speed of operation for applications using larger chips. Through the exploitation we achieved 47.91% area improvement and 11.49% improvement in total power consumption.

Index Terms—AES(Advanced Encryption Standard), S-Box(Substitution Box), CFA(Composite Field Arithmetic), GF(Galois Field), ANF(Algebraic Normal Form).

I. INTRODUCTION

CRYPTOGRAPHY plays an important role in security of the data transmission. The Advanced Encryption Standard(AES) is an encryption standard chosen by the National Institute of Standards and Technology(NIST) in 2001. The AES algorithm has broad applications, including smart cards and cellular phones, WWW servers and automated teller machines (ATMs), and digital video recorders. The AES algorithm has a fixed block size of 128 bits and key length can be of 128,192 or 256 bits. It generates its key from using Key Expansion function involved for both transformation is as shown below.

\textbf{SubByte}: MultiplicativeInversion in GF(\((2^8)^3\)) \rightarrow Affine Transformation

\textbf{InvSubByte}: \rightarrow Inverse Affine Transformation \rightarrow Multiplicative Inversion in GF(\((2^8)^3\))

At first we will deduce and present new composite field AES S-box construction which is followed by exploring all of the eight possible isomorphic mappings for each composite field construction. In AES implementation, for the first time, we introduced the exploitation of subsharing optimization for binary matrix multiplication. To reduce the area required in the isomorphic mappings, precisely, we apply a new common subexpression elimination (CSE) algorithm. To cater the drawback of CFA, we proposed a novel architectural optimization scheme for achieving higher speed of implementation. In order to outperform the performance, we have applied fine-grained pipelining to the multiplier by using the existing pipelining schemes for CFA AES S-Box. CFA AES S-boxes are

\footnotesize

Manuscript received July 15, 2012.

Vasili Supraja, M.Tech – VLSI Design, ECE Department, SRM University, (e-mail: supraja.vasilia1@gmail.com), Chennai, India.

P. Mahesh Kannan, Assistant Professor(Sr. Grade), ECE Department, SRM University, Chennai, India. (e-mail: maheshkannanp@gmail.com).

ISSN: 2277 – 9043
International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE)
Volume 2, Issue 4, April 2013

405

All Rights Reserved © 2013 IJARCSEE
converted into direct computation modules which are expressed using algebraic normal form (ANF) representations, consisting of only AND, OR and XOR operations to the end of paper. The consequences of the paper can be organised as follows. At first in the section 2 we discuss some details on the AES algorithm. Where as Section 3 enlists hardware implementation of Rijndael and analytical comparison between our optimum CSE AES S-Boxes and selected previous works. Section 4 depicts the conclusion on the paper and outlines future work.

II. AES ALGORITHM

Rijndael has a variable block and key length which can be 128, 192 or 256 bits, the AES standard includes only block length of 128 bits. The level of security depends on the key length we choose for the corresponding encryption process. In this implementation we focus on the 128-bit key version of AES which has 10 rounds. In this case, each round and the initial stage require a 128-bit round key. In total 10 sets of round keys are generated from the secret key by using S-box. The input data is arranged as a table i.e., a matrix of bytes. Figure 1 outlines the basic structure of the algorithm. The round transformation consists of four different transformations: SubByte, ShiftRow, MixColumn and AddRoundKey. They are performed in order with exception of the final round which is slightly different. All transformations are based on byte-oriented arithmetic and AddRoundKey is a bitwise XOR operation. The transformations operate on the intermediate result, which is called the state. The SubByte transformation is a non-linear byte also called S-Box (Substitution table) The S-Box function of an input byte a is defined by two substeps:

Inverse: Let $c = a^{-1}$, the multiplicative inverse in GF$(2^8)$ field, modulo irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$ (except if $a = 0$ then $c = 0$).

Affine transformation: Then the 8 bit output is given as $S = M \cdot c \oplus b$, where $M$ is a specified 8x8 matrix of bits, $b$ is a specified byte, and the bytes $c$, $b$, $s$ are treated as vectors of bits.

$$
\begin{pmatrix}
    s_7 \\
    s_6 \\
    s_5 \\
    s_4 \\
    s_3 \\
    s_2 \\
    s_1 \\
    s_0 \\
\end{pmatrix}
= \begin{pmatrix}
    1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
    0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
    1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
    0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
    1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\
    1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
    1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
    1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
    c_7 \\
    c_6 \\
    c_5 \\
    c_4 \\
    c_3 \\
    c_2 \\
    c_1 \\
    c_0 \\
\end{pmatrix}
\oplus
\begin{pmatrix}
    0 \\
    1 \\
    0 \\
    0 \\
    0 \\
    0 \\
    0 \\
    1 \\
\end{pmatrix}
$$

Direct calculation of the inverse (modulo an eighth-degree polynomial) of a seventh-degree polynomial is not easy. In this paper we utilize a CSE algorithm[8] that combines both greedy algorithm[9] and exhaustive search[10] in an iterative two-term pattern selection. But calculation of the inverse (modulo a second-degree polynomial) of a first degree polynomial is relatively easy. This suggests the following changes of representation in GF$(2^8)$.

1) We take isomorphism between GF$(2^8)$ and GF$(2^8)/GF(2)$ to represent the general element ‘g’ of GF$(2^8)$ the degree ‘1’ or less by Review Stage

$$
g = \gamma y^\gamma_0
$$

the polynomial over GF$(2^8)$ that is in y.

Irreducible polynomials[11]

$$
r(y) = y^2 + ty + \gamma \quad ----- (a)
$$

all the coefficients are in GF$(2^8)$, the pair $[\gamma_1, \gamma_0]$ can be represented ‘g’ in terms of polynomial basis $[Y, 1]$, where Y is one root of $r(y)$. Same operation in Normal basis $[Y^{16}, Y]$

$$
r(y) = y^2 + ty + \gamma = (Y + Y^{16}) (y + Y^{16})
$$

so,

$$
\gamma = N F_{16}/F_{16}(Y) = \text{the Norm}
$$

2) Second GF$(2^8)/GF(2^8)$ it can represent GF$(2^4)$ in Z over GF$(2^2)$

$$
\gamma = \Gamma, z + \Gamma_0
$$

Irreducible polynomial

$$
s(z) = z^2 + Tz + N \quad ----- (b)
$$

all coefficients are in GF$(2^8)$ in polynomial basis $[Z, 1]$, Z is one root of $s(z)$. Same operation in Normal basis $[Z^2, Z]$.

$$
T = T F_{16}/F_{16}(z) \text{is the Trace}
$$

$$
N = N F_{16}/F_{16}(z) \text{is the Norm}
$$

3) GF$(2^8)/GF(2)$ to represent GF$(2^2)$ linear polynomials over GF$(2)$

$$
\Gamma = g_1 \omega + g_0
$$

Irreducible polynomial

$$
t(\omega) = \omega^2 + \omega + 1
$$

where $g_1, g_0 \in \{0, 1\}$ and in polynomial basis $[W, 1]$, where $W$ is either $\Omega$ or $\Psi$ and in Normal basis $[W^2, W]$.

Here in this S-Box value determination finding of inverse in terms of higher power is a complicated task and requires complex circuitry. Hence we are calculating those higher power elements inverse values in terms of low power elements[6] and [5]. To find the inverse GF$(2^8)$, i.e $g$ is d where

$$
g = \gamma_1 y^\gamma_0
$$

$$
d = \delta_1 y^\delta_0
$$

We already know that

$$
g x d = 1
$$

$$
gd = (\gamma_1 y^\gamma_0)(\delta_1 y^\delta_0) \text{mod } (y^2 + \gamma y + \gamma)
$$

$$
= \gamma_1 \delta_1 y^{\gamma_1 \gamma_0 + \delta_1 \delta_0 + \gamma_1 \delta_0 + \delta_1 \gamma_0 + \gamma_0 \delta_1 + \gamma \delta_0 + \gamma \delta_0 \gamma)
$$

We know

$$
\gamma_1 \delta_1 y^2 + \gamma_1 \delta_0 y^\gamma = 0
$$

(since XOR operation of same input means output is zero)

$$
= (\gamma_1 \delta_1 y^{\gamma_0 \delta_1 + \gamma_0 \delta_1 \gamma})(\gamma_0 \delta_1 + \gamma_0 \delta_1 \gamma)
$$

We get

$$
0 = \gamma_1 \delta_1 y^{\gamma_0 \delta_1 + \gamma_0 \delta_1 \gamma} = \text{0y+1}
$$

Ordering this equation

$$
0 = \gamma_1 \delta_1 y^{\gamma_0 \delta_1 + \gamma_0 \delta_1 \gamma} \quad (1)
$$

$$
1 = \gamma_0 \delta_1 y + \gamma_0 \delta_1 \gamma \quad (2)
$$

Multiplying by $\gamma_0$ on both sides of equation (1)

We get

$$
0 = \gamma_0 \gamma_1 \delta_1 y^{\gamma_0 \delta_1 + \gamma_0 \delta_1 \gamma} \quad (3)
$$

Multiplying by $\gamma_0$ on both sides of equation (2)

We get

$$
1 = \gamma_0 \gamma_1 \delta_1 \gamma \quad (4)
$$

adding equation (3) and (4) we get

$$
\gamma_1 = (\gamma_0 y + \gamma_1 \gamma)(\gamma_1 \gamma)(\gamma) \delta_1 \gamma \quad (5)
$$
From equation (3) we can get
\[ \gamma_1 \delta_0 = (\gamma_0 + \gamma_1 \tau) \delta_0 \quad \text{(6)} \]
Finally we want to find \( \delta_1, \delta_2 \) in equation (5)
\[ \gamma_1 = (\gamma_1 \gamma + \gamma_1 \tau \gamma + \gamma_0 \tau)^{-1} \]
in equation (6)
\[ \gamma_1 \delta_0 = (\gamma_0 + \gamma_1 \tau) \delta_1 \]
\[ \gamma_1 \delta_1 = (\gamma_0 + \gamma_1 \tau) (\gamma_1 \gamma + \gamma_1 \tau \gamma + \gamma_0 \tau)^{-1} \]
\[ \delta_0 = (\gamma_0 + \gamma_1 \tau) (\gamma_1 \gamma + \gamma_1 \tau \gamma + \gamma_0 \tau)^{-1} \]
finally we can write
\[ \delta_1 = (\gamma_1 \gamma + \gamma_1 \tau \gamma + \gamma_0 \tau)^{-1} \]
\[ \delta_0 = (\gamma_0 + \gamma_1 \tau) (\gamma_1 \gamma + \gamma_1 \tau \gamma + \gamma_0 \tau)^{-1} \]

2) Similarly inverse intermsGF(2^2)
Inverse of \( \gamma \) is \( \delta \), where
\[ \gamma = \gamma_1 \gamma + \gamma_0 \]
\[ \delta = \delta_1 \gamma + \delta_0 \]
Same operation doing finnally we are getting
\[ \Delta_1 = (\gamma_0 + \gamma_1 \tau) (\gamma_1 \gamma + \gamma_1 \tau \gamma + \gamma_0 \tau)^{-1} \]
\[ \Delta_0 = (\gamma_0 + \gamma_1 \tau) (\gamma_1 \gamma + \gamma_1 \tau \gamma + \gamma_0 \tau)^{-1} \]

3) Similarly inverse intermsGF(2^3)
Inverse of \( \gamma \) is \( \Delta \), where
\[ \gamma = \gamma_1 \gamma + \gamma_0 \gamma \]
\[ \delta = \delta_1 \gamma + \delta_0 \gamma \]
By doing same operation finnally we will get
\[ d_1 = g_1 (g_1 \gamma + g_0 \gamma + g_0 \gamma)^{-1} \]
\[ d_2 = (g_0 + g_1) (g_1 \gamma + g_0 \gamma + g_0 \gamma)^{-1} \]

4) For GF(2) Trace and Norm are \( t(\omega) \) and is 1
\[ d_1 = g_1 \]
\[ d_0 = g_0 + g_1 \]

Here if these inversion formulas were applied with both input as zeros means then inverse output values will also be zero only[5]. Here finding the result of these inverse operation module involves multiplication module, squarer module, scalar module, adder modules. These modules are best defined by using the different basis(norm and polynomials) of representation. Then these are defined in terms of GF (i.e in terms of addition and multiplication) by using XOR gates, OR gates and AND gates for minimal area of implementation with minimum power consumption and high throughput.

III. HARDWARE IMPLEMENTATION OF PROPOSED DESIGN

In this section we will try to optimise for the area, power and speed. Fig 3.1 describes the hardware overview of this S-Box implementation.

![Figure 3.1](image)

While finding the Substitution-Box values, calculation of inverse values of the inverse module of 8 bit plays the key role. The complexity of the circuitry increases with an increase in the bit values. Hence to decrease the complexity and the area of implementation we will opt for finding the inverse value of higher bits in terms of lower order bits i.e 8 bit inverse values in terms of 4 bit and 2 bit.

NIST has defined the Isomorphism, inverse isomorphism, affine transform and inverse affine transform matrices. While solving the algorithm we assumed values of \( \tau \) and \( \gamma \) as \( \tau = \{1100\} \) and \( \gamma = \{10\} \).

Architecture overview shown in fig 3.1 is implemented on Quartus Cyclone II EP2C5T144C6 FPGA. Technological schematic of 8bit inversion module of S-Box is as given below in figure 3.2 which takes an 8 bit value and provides 8 bit inverted value for the same which is calculated in terms of 4 bit and 2 bit values.

Here we proposed fine grained pipelining to achieve high speed of operation in terms of GF(2^4) and GF(2^2). Hence high speed of operation with minimal area of implementation is achieved in our paper. At first we need to convert the complicated circuit into several logical expressions without violating the functionality of the circuit’s property to ensure the efficiency of the pipelined construction. We can also say that, different isomorphism in different sub operation stages are now replaced with direct computation modules which are expressed in ANF consisting of only AND gates, OR gates and XOR gates. While converting from a CFA AES S-Box into logical expressions, has to be done in such a way that it should not induce excessive area. For making sure of minimal area we first translated all the sub operations in the main modules of S-Box i.e inversion module into logical expressions individually. Isomorphism function, the 4-bit adder, the square-scalar/square/scaler, the GF(2^2) and GF(2^3) multiplier, the GF(2^4) and GF(2^5) inverter and the inverse isomorphism function with affine transformation will include these sub operations. Before inserting pipeline stages we grouped and merged some of the sub-operations into several ANF modules. Figure 3.2 shows the main module of substitution box that is inverse module which takes 8bit value as input and gives 8bit inverted value. Inside that module inverse operation is done in terms 4bit and 2bit to reduce the complexity of the circuit. Figure 3.3 shows the inner module of inverse operation module in which 4bit inverse values are calculated which takes 4bit input and provides 4bit inverted output.

The another important consideration factor is the choice of implementing pipelining scheme. We have implemented the proposed AES-Box on the Quartus Cyclone II EP2C5T144C6 FPGA for this implementation. Table 3.1 provides the comparison with the previous work to our implementation.

According to the reports it requires a total number of 50 LE(Logical Elements), 16 pins on Quartus Cyclone II EP2C5T144C6 FPGA. It shows there is total of 47.91% of area improvement for the pipelined architecture of ANF-CFA AES S-Box architecture which is operated at a voltage of 1.2V. It consumes a total power of 30.8mW for providing the corresponding substitution values.
IV. RESULT AND DISCUSSION

We have presented, a detailed study on composite field construction for the S-box function in AES, in this paper. Here main concentration was on derivation of a new composite field AES S-box [1] to achieve an optimally balanced construction in terms of area optimization and frequency of operation, compared to the previous studies [4] and [3] in which we used 48 XOR gates, 34 AND gates and 6 OR gates for inverse operation in S-Box module (i.e, main module). In addition, we had explored all the possible isomorphic mappings for the composite field construction and employed an efficient field multiplication using isomorphic and inverse isomorphic mappings with affine transformation. The best architecture obtained with area improvement of 47.91% and 11.49% in power consumption. After the employment of several stages of fine-grained pipelining, the hardware architecture of S-Box to achieve high frequency of operation, it consumed total thermal power of 30.8 mW.

V. CONCLUSION

The methodologies proposed in this work are also applicable for the development of any similar cryptographic circuits that involve finite field arithmetic and part from AES S-box. The main focus on our future work will be on constructing a composite field with field polynomials in the multi-level representation. Future enhancement of this paper is to construct AES encryption and as well as the decryption CFA on both side of coding and decoding sides.

References
