DESIGN AND IMPLEMENTATION OF S-BOX ARCHITECTURES IN VLSI

PRIYA STALIN¹, PRIYANKA K²

¹ Assistant Professor, ² Student Scholar, Department of Electronics & Communication Engineering, Sree Sastha Institute of Engineering & Technology,

ABSTRACT

Three novel composite field arithmetic (CFA) Advanced Encryption constructions Standard (AES) S-boxes of the field GF (((2²)²)²) have been derived in this work. The best construction is selected after a sequence of algorithmic and architectural optimization processes. Furthermore, for each composite field, there exist eight possible isomorphic mappings. Therefore, after the exploitation of a new common sub expression elimination algorithm, the isomorphic mapping that results in the minimal implementation area cost is chosen. The technique used, shows less power consumption and shorter execution cycles in synthetic applications, with minimum mapping time overhead. Through the exploitation of both algebraic normal form and seven stages fine-grained pipelining, the best case can achieve a throughput 3.49 Gbps on a field-programmable gate array.

Keywords – Advanced Encryption Standard (AES), algebraic normal form (ANF), composite field arithmetic (CFA), S-box.

1. INTRODUCTION

The Advanced Encryption Standard (AES) is an encryption standard chosen by the National Institute of Standards and Technology (NIST) in 2001, which has its origin in the Rijndael block cipher. Several studies in the area had identified the nonlinear Sub Bytes transformation as the major bottleneck in achieving both small area and high speed VLSI AES implementations. This brief presents a methodology that is based on a pure combinatorial circuitry. In which, the Galois inverse of elements in is computed prior using the composite field arithmetic (CFA). To date, there are several successful composite field constructions reported for AES S-box implementations. The AES algorithm, also called the Rijndael algorithm, is a symmetric encryption algorithm; meaning encryption and decryption are performed by essentially the same steps [1]. It is a block cipher, where the data is encrypted / decrypted in blocks of 128 bits. (The original Randal algorithm allows other block sizes, but the Standard only permits 128-bit blocks.) Each data block is modified by several “rounds” of processing, where each round involves four steps. Three different key sizes are allowed: 128 bits, 192 bits, or 256 bits, and the corresponding number of rounds for each is 10 rounds, 12 rounds, or 14 rounds, respectively. From the original key, a different “round key” is computed for each of these rounds. For simplicity, the discussion below will use a key length of 128 bits and hence 10 rounds.

There are several different modes in which AES can be used. For some of these, such as Cipher Block Chaining (CBC), the result of encrypting one block is used in encrypting the next. These are called feedback modes, and the feedback effectively precludes pipelining (simultaneous processing of several blocks in the “pipeline”). Other modes, such as the “Electronic Code Book” mode or “Counter” modes do not require feedback. These non feedback modes may be pipelined for greater throughput. But for hardware implementations of AES, there is one drawback of the table look-up approach to the S-box function: each copy of the table requires 256 bytes of storage, along with the circuitry to address the table and fetch the results. Each of the 16 bytes in a block can go through the S-box function independently, the byte substitution step. This then effectively requires 16 copies of the S-box table for one round. To fully pipeline the encryption would entail “unrolling” the loop of 10 rounds into 10 sequential copies of the round calculation. This would require 160 copies of the S-box table, a significant allocation of hardware resources. In contrast, this work describes a direct calculation of the S-box function using sub-field arithmetic, similar to while the calculation is complicated to describe, the advantage is that the circuitry required to implement this in hardware is relatively simple, in terms of the number of logic gates required [5]. This type of S-box implementation is significantly smaller (less area) than the table it replaces, especially with the optimizations in this work. Furthermore, when chip area is limited, this compact implementation may allow parallelism in each round and/or unrolling of the round loop, for a significant gain in speed.

The AES algorithm, designed by Joan Daemen and Vincent Rijmen, has an SPN (Substitution Permutation Network) structure. Its use is mandatory for the encryption of sensitive but unclassified US government information; in 2003 the US government has announced that it can also be used for encrypting secret and top secret information (for the last category key lengths of at least 192 bits need to be used). AES is currently replacing the Data Encryption Standard as the worldwide standard algorithm. Design challenges for AES mainly lie in exploring all the options for the Sox design. The most common strategy to reduce the gate complexity consists of exploiting composite field arithmetic. By following that approach one still has several options to
represent the finite field GF (28). In this paper we represent GF (28) as the composite field GF (((2^2)^2)^2). In this way, we reduce the arithmetic in GF (28) to operations in smaller fields. There are many ways to represent GF (28) as GF (((2^2)^2)^2). Choices need to be made with respect to the irreducible polynomials that are used to create the extension fields and with respect to the transformation matrices that map elements from one representation to the other [9]. Exploring these two degrees of freedom we optimize the S-box of Satoh, which is to our knowledge the most compact implementation today. The existing system performs the 8-bit Galois field inversion of the S-box using subfields of 4 bits and of 2 bits. It describes a refinement of this approach that minimizes the circuitry, and hence the chip area, required for the S-box. It used polynomial bases at each level, and normal bases also, with arithmetic optimizations; altogether, 432 different cases were considered. The isomorphism bit matrices are fully optimized, improving on the “greedy algorithm.” The best case reduces the number of gates in the S-box by 20%.

2. COMPOSITE FIELD ARITHMETIC
Composite fields are frequently used in implementations of Galois Field arithmetic. In cases where arithmetic operations rely on table lookups, subfield arithmetic is used to reduce lookup-related costs. This technique has been used to obtain relatively efficient implementations for specific operations such as multiplication, inversion and exponentiation.

The actual hardware implementation of the proposed CFA AES S-box constructions is discussed. The implementations can be viewed as two-stage procedure. First, the circuit is manually coded using a hardware description language for the three proposed CFA AES S-boxes. Next, ANF representation is employed along with a strategic fine-grained pipeline registers insertion, in an attempt to validate the feasibility of the proposed compact CFA AES Boxes in achieving high throughput hardware implementations. Consequently, the fastest speed achievable is bounded by the time required for the most complex sub-operations, in this case, the Multiplication in GF (2^4). In order to overcome this constraint, the multiplication sub-operation is divided into two parts, i.e., fine-grained pipelining. To ensure the efficiency of the pipelined constructions, the complicated circuit is first converted into several logical expressions without violating the functionality of them circuit’s properties. In other words, the sub-operations over different isomorphism stages are now replaced with direct computation modules which are expressed in ANF, consisting of only AND gates and XOR gates.

Converting a CFA AES S-box into logical expressions has to be done in a way that it does not induce excessive area increase. To ensure this, all the sub-operations are first translated in GF (2^8) inversion into logical expressions individually. These sub-operations include the isomorphism function, the 4-bit adder, the squarer/scalar, the GF (2^4) multiplier, the GF (2^4) inverter and the inverse isomorphism function with affine transformation. Next, some of the sub-operations are grouped and merged into several ANF modules before inserting pipeline stages.

Different construction schemes for composite fields are proposed for the AES algorithm. In previous design, GF (2^8) is decomposed into, GF ((2^4)^2) and composite field arithmetic is applied to all the transformations in the AES algorithm. The optimum construction scheme for GF ((2^4)^2) is selected based on minimizing the total gate count in the implementation of all transformations. However, it is more efficient to apply composite field arithmetic only in the computation of the multiplicative inversion in the Sub Bytes and Inv Sub Bytes transformations. In this case, the construction scheme selected in is no longer optimum.

Fig. 2. ANF-CFA AES S-box with seven stages fine-grained pipelining for (a) Case I, (b) Case II, and (c) Case III.

3. MULTIPLICATIVE INVERSE IN S-BOX USING CFA
In AES, the encryption of the data is performed on blocks of byte, through the representation in GF (2^8) with the specified field polynomial \( q(x) = x^8 + x^4 + x^3 + x + 1 \). Every round in AES consists of four identical transformations, i.e., Sub Bytes, Shift Rows, Mix Columns, and Add Round Key. In a nutshell, the S-box
function, which is claimed to be most resource consuming operation, involves finding a multiplicative inverse over GF(2^8) and followed by an affine transformation.

Finding the multiplicative inverse of elements from the higher order field is a very tedious effort. Rijmen was the first to propose an alternative solutions by employing CFA, where one maps the element in GF(2^8) to its subfield that yields less complexity in the multiplicative inverse. A composite field can be built iteratively from its lower order fields; therefore the actual mathematical manipulation can be done in the lower fields rather in the original higher order field. The following summarizes the steps in performing multiplicative inversion using CFA:

1) Map all elements of field A to a composite field B using isomorphism function; b=ψ(a) =δ×a;
2) Compute the multiplicative inverse over B; x=ψ^(-1)(b) (except if b=0, then x = 0);
3) Remap the computation results to A, using the inverse isomorphism function; a=ϕ^(x) (x) =δ×(-1)×x.

Mapping Galois Field from GF(2^8) to GF(((2^2)^2)^2) requires three stages of isomorphism and field polynomials which are stated (in a general form) as follows:

\[ r(y) = y^2 + τy + υ \]  (isomorphism for GF(2^8)/GF(2^4))  \hspace{1cm} (3.1)
\[ s(z) = z^2 + Tz + N \]  (isomorphism for GF(2^4)/GF(2^2))  \hspace{1cm} (3.2)
\[ t(w) = w^2 + w + 1 \]  (isomorphism for GF(2^2)/GF(2))  \hspace{1cm} (3.3)

In this work, CFA for multiplicative inverse (in S-box algorithm) over the composite field GF((2^2)^2)^2) with respect to both polynomial basis and normal basis is presented. Based on the thorough reviews of these architectures, yet another new normal basis composite field AES S-box that uses a combination of norm and trace unities in different field polynomials has been derived. In total, the proposed three new constructions listed as follows.

CASE I: Using polynomial basis representation with field polynomials’ norms equal to unity (both υ and N in (3.1) and (3.2) equal to unity).
CASE II: Using normal basis representation with field polynomials’ norms equal to unity (both υ and N in (3.1) and (3.2) equal to unity).
CASE III: Using normal basis representation with τ in (3.1) and N in (3.2) equal to unity.

4. COMMON SUB EXPRESSION ELIMINATION ALGORITHM

In computer science, common sub-expression elimination (CSE) is a compiler optimization that searches for instances of identical expressions (i.e., they all evaluate to the same value), and analyses whether it is worthwhile replacing them with a single variable holding the computed value. The core of many VLSI design tasks is the multiplication of a variable by a set of constants (digital filtering, image processing, linear transforms, etc.). The optimization of these multiplications can lead to important improvements in various design parameters like area or power consumption. The goal of CSE is to identify the bit patterns that are present in the coefficient set more than once. Since it is sufficient to implement the calculation of the multiple identical expressions only once, the resources necessary for these operations can be shared.

In general, the goal of CSE can be defined as follows.

1) Identify multiple patterns in the coefficient set.
2) Remove these patterns and calculate them only once.

The binary 8*8 matrix-vector product in isomorphic mapping can be expressed as eight bit-level equations. Therefore, CSE would be useful in extracting common factors from these bit-level equations to reduce the area cost and the critical path. In this work, a CSE algorithm that combines both greedy algorithm and exhaustive search in an iterative two-term pattern selection is utilized.

To further enhance the performance of the proposed constructions, some optimization measures from the algorithmic and the architectural perspectives are performed. For algorithmic optimization, a precise selection of the field polynomials’ coefficients that result in minimal arithmetic complexity is to be performed. For architectural optimization, elimination of redundant common factor in the inverter can be conducted, followed by a merger of certain multipliers with its respective sub-operations.

4.1. ALGORITHMIC strength reduction

As a fore mentioned, there is a certain level of freedom in the choice of coefficients in the field polynomials r(y) and s (z). Each of the coefficient combinations will lead to a different level of complexities in CFA.

In GF(2^8)/GF(2^4), the inversion involved is as follows:

\[ δ1= [(y^2)^0+(y^2)^0(y^0)^0]-1y^0. \]  \hspace{1cm} (4.1)
\[ δ0= [(y^2)^0+(y^2)^0(y^0)^0]-1y^1 \]  \hspace{1cm} (4.2)

which is reduced to inversion and multiplications in the field of GF(2^4).

In this case, refer to (3.2) and (3.3), combine υ = Δ1Z4 + Δ0Z4 with the squaring of γ = Γ1Z4 + Γ0Z As such, only one specific circuit for \( γ^2 \) is needed. Squaring of \( γ = (Γ1Z4+Γ0Z) \) is deduced as

\[ γ^2 = (Γ1+Γ0T^2)Z4+[(Γ0+Γ1T^2)Z^2] \]  \hspace{1cm} (4.3)

and therefore \( γ_2 \) are deduced as follows:
\begin{equation}
\nu_2 = [(\Delta_1 \Gamma_1^2 T_2 + \Delta_0 (T_1 + T_0)] Z^4 + [(\Delta_0 \Gamma_0^2 T_2 + \Delta_1 (T_1 + T_0)] Z^4
\end{equation}

The \( \nu \) values that contribute to minimal complexity in this squarescaler are tabulated. The squaring-scaler operation requires sub-operations in GF \((2^2)\). Hence scaling of both \( T+1 \) are \( g_0 \) derived as follows:

\begin{equation}
(W)\hat{\Delta}(g_1 W^2 + g_0 W) = (g_0 \hat{\Delta} g_1) W^2 + (g_1) W
\end{equation}

\begin{equation}
(W_2)\hat{\Delta}(g_1 W^2 + \hat{g}_0 W) = (g_0) W^2 + (g_0 \hat{\Delta} g_1) W
\end{equation}

Meanwhile, the squaring of \( G = g_1 W^2 + g_0 W \) in GF \((2^2)\) is given by

\begin{equation}
G^2 = (g_1 W^2 + g_0 W) (g_1 W^2 + g_0 W) = (g_0 W^2 + g_1 W)
\end{equation}

On the other hand, a combination of scaling and squaring gives

\begin{equation}
W G^2 = W (g_0 W^2 + g_1 W) = (g_0 + g_1) W^2 + g_0 W
\end{equation}

For the normal basis, the squaring in GF\((2^2)\) is a free operation, which requires only one swapping.

### 4.2. Architectural Optimization

First, a potential sub sharing is readily available in the subfield multipliers. The sum of the higher and lower halves of each factor can be shared between two or more subfield multipliers which have the same input factor. A 2-bit factor shared by two GF \((2^2)\) multipliers saves one XOR addition while a 4-bit factor shared by two GF \((2^4)\) multipliers saves five XORs. Second, combining sub-operations in GF \((2^4)\) is capable of producing further area savings. To begin with, GF \((2^2)\) multiplier is combined with a scalar in a GF \((2^4)\) multiplier. Such approach results in saving three XORs in total gates and one XOR in the critical path for the GF \((2^4)\) multiplier. Then, another potential multiplier-scalar combination is available in GF \((2^4)\) inverter. Such merging gives one additional XOR reduction in both total gates and critical path. Last, combining the sum of higher and lower halves of the inputs (common factors with GF \((2^4)\) multiplier) and the following square scalar saves two XORs in GF \((2^8)\) inverter.

### 5. IMPLEMENTATION OF CFA AES S-BOXES

In this work, first two new composite field AES S-box constructions are deduced and presented. Thorough analysis of these two constructions and the existing works reported in the literatures lead to the development of yet another new composite field AES S box. Consequently, there are three new CFA AES S-boxes are proposed in this work. Second, of the eight possible isomorphic mappings for each composite field construction are explored. This work, for the first time, introduces the exploitation of sub sharing optimization for binary matrix multiplication in AES implementation. Isomorphism is studied in mathematics in order to extend insights from one phenomenon to others. If two objects are isomorphic, then any property that is preserved by an isomorphism and that is true of one of the objects is also true of the other. A named Isomorphism indicates which features are selected for this purpose. Thus, for example, two objects may be isomorphic, since the isomorphism selects the additional structure of the multiplicative operator. To construct isomorphic mappings for , we need to find eight base elements \(1, \beta, \beta^2, ..., \beta^7\), of GF\(((2^2)^2)^2)\) to which the base elements \(1, \alpha, \alpha^2, ..., \alpha^7\) are mapped. The isomorphic mapping \(\delta\) matrix is formed by taking the binary representation of \(\beta^j\) as the entries of its \((8-j)\)th column. The multiplicative inversion in involved in the Sub-Bytes/Inv Sub Bytes is a hardware demanding operation; it takes at least 620 gates to implement by repeat multiplications. This gate count can be reduced greatly by using composite field arithmetic. The multipliers in GF \((2^4)\) can be further decomposed into multipliers in GF\((2^2)\) and then to GF\((2)\) , in which a multiplication is simply an AND operation.

### 6. PERFORMANCE EVALUATION

<table>
<thead>
<tr>
<th>PARAMETERS</th>
<th>EXISTING METHOD(using only greedy algorithm)</th>
<th>PROPOSED METHOD(greedy algorithm &amp; exhaustive search)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Consumption</td>
<td>33.45mW</td>
<td>33.1mW</td>
</tr>
<tr>
<td>Throughput</td>
<td>0.98gbps</td>
<td>3.49gbps</td>
</tr>
</tbody>
</table>

### 7. SIMULATED RESULTS

![Fig.7.1 case 1 Polynomial S-box](image1)

![Fig.7.2 polynomial inverter](image2)
CONCLUSION

In this brief, a detailed study on composite field construction for the S-box function in AES was presented. The major contribution of our work was the derivation of a new composite field AES S-box that achieves an optimally balanced construction in terms of area of implementation and critical path, compared to the previous studies. Furthermore, we had explored all of the possible isomorphic mapping for each of the composite field construction and employed a new CSE algorithm to derive the most optimum isomorphic and inverse isomorphic mapping with affine transformation.

REFERENCES