Composite Field Decomposition for Higher Order S-Boxes

Christopher A. Wood

April 1, 2013

Agenda

1. Composite Fields
2. Common AES Composite Field Extensions
3. Tower Field Isomorphic Functions
4. AES Combinational Implementations
5. Hardware Designs
   - 8-Bit S-Boxes
6. Hardware Designs
   - 16-Bit S-Boxes

- *Q: Why use combinational logic?*
- A: Optimize hardware area of SubBytes operation on memory-constrained platforms (i.e. where LUTs are not acceptable implementations)
- *Q: How are composite fields useful?*
- A: The multiplicative inverse calculation is the most expensive operation - using composite fields helps us reduce the gate-level complexity of this operation.

Composite Fields

A *composite field* is a pair

$$\{GF(2^n), Q(y) = y^n + \sum_{i=0}^{n-1} q_i y^i, q_i \in GF(2)\}$$

$$\{GF((2^n)^m), P(x) = x^m + \sum_{i=0}^{m-1} p_i x^i, p_i \in GF(2^n)\},$$

where $GF(2^n)$ is constructed from $GF(2)$ by $Q(y)$, and $GF((2^n)^m)$ is constructed from $GF(2^n)$ by $P(x)$. Also, $GF((2^n)^m)$ is a degree m extension of $GF(2^n)$.

Research has systematically examined all composite field extensions for $GF(2^8)$, seeking to minimize the area...

$$GF(2^4) - GF((2^4)^2)$$
$$GF(2^2) - GF((2^2)^4)$$
$$GF(2^2) - GF((2^2)^2) - GF(((2^2)^2)^2)$$

- The irreducible polynomial cannot exceed the degree of the extension → degree two extensions have the smallest number of irreducible polynomials!
- A systematic evaluation must check all tower field extensions **and** all irreducible polynomials.
- **Thesis Work:** Generate all possible irreducible polynomials for all tower field extensions.

The isomorphic functions are constructed as follows (assume we're constructing a function for mapping $GF(2^{nm}) \rightarrow GF((2^n)^m)$):

- Find two generators α and β ($\alpha \in GF(2^{nm})$ and $\beta \in GF((2^n)^m)$), where α and β are roots of the same primitive irreducible polynomial.
- Map $\alpha^k \rightarrow \beta^k$ for $1 \leq k \leq 2^{nm}$ (mapping basis elements of $GF(2^{nm}) \rightarrow GF((2^n)^m)$). If the mapping doesn't hold group homomorphism, find the next generator β and repeat.

AES Combinational Implementations

Every element in a field $GF(2^{nm})$ can be represented by a polynomial with coefficients from $GF(2^n)$ using an irreducible polynomial of the form $x^2 + Ax + B$ (we assume $m = 2$). Thus, if $\alpha \in GF(2^{nm})$, and $\alpha = bx + c$, where $b, c \in GF(2^n)$, then:

$$\alpha^{-1} = (bx + c)^{-1} = b(b^2B + bcA + c^2)^{-1} + (c + bA)(b^2B + bcA + c^2)^{-1}$$

Now we compute the inverse over $GF(2^n)$! We can also play with A and B to simplify the isomorphic mapping.

Isomorphic Functions

Once an isomorphic function δ is defined it can be implemented in hardware using constraint matrices.

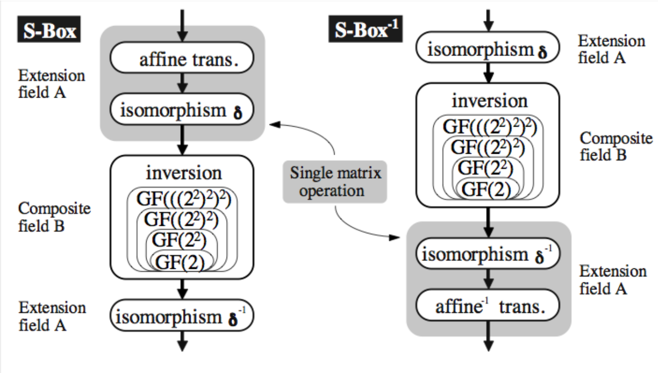
Example: A δ from $GF(2^8) \rightarrow GF((2^4)^2)$ (using the irreducible polynomial $p(x) = x^8 + x^4 + x^3 + x^2 + 1$) can be defined as follows:

$$\delta = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Of course, this just reduces to an array of XOR gates.

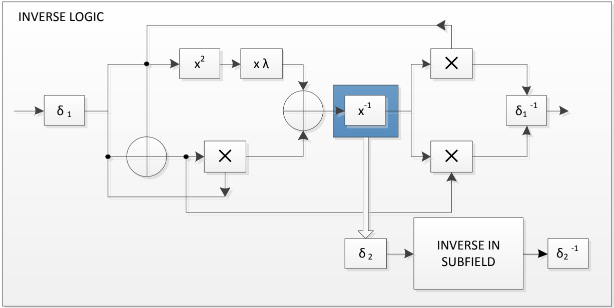
8-Bit S-Boxes

Satoh tower field design - $GF(((2^2)^2)^2)$



16-Bit S-Boxes

Proposed design - $GF((((2^2)^2)^2)^2)$



Research question: How deep should this recession go? What yields the minimal hardware area?

VHDL Model Overview

- **Top-level entity** - SBOX (16-bit input, 16-bit output)
  - Internal signals for the input and output of each of the operational blocks
- There will be one entity for each multiplicative inverse operation (with the appropriate width I/O signals)

What's next?

Action Items

- Hard deadline for VHDL design: **3/15/13** (sent via email)
- Hard deadline for software implementation of isomorphic mapping generation: **3/16/13** (sent via email) - Python (for simplicity)
- Finalized software and scripts to generate vectorial Boolean function representations of S-boxes: **3/17/13** (sent via email) - combination of C/Python

Next meeting: **?**