Sequential Verification over Galois Field based on Word-level Abstraction

Xiaojun Sun
Electrical and Computer Engineering
University of Utah
Salt Lake City, Utah
Email: xiaojun.sun@utah.edu

Priyank Kalla
Electrical and Computer Engineering
University of Utah
Salt Lake City, Utah
Email: kalla@ece.utah.edu

***Abstract*—This paper introduce a method to apply *Word-level Abstraction* to sequential arithmetic circuits verification. Details of word-level abstraction are concluded in Tim's IWLS and FMCAD paper. Here we discuss more on Sequential Multiplier with Parallel Output (SMPO) designed for normal basis multiplication, as well as *optimal normal basis theory*.**

I. Fundamentals on Mathematics

A. Normal Basis Theory

Let β be a element in the Galois field F_{2^n} constructed by primitive element α and minimal polynomial $f(\alpha)$. Then a basis in the form $\{\beta, \beta^2, \beta^4, \beta^8, ..., \beta^{2^{n-1}}\}$ is a *Normal Basis*; here β is called *Normal Element*.

All about normal basis theory can be found in *Gao's Thesis*. [1]

B. Normal Basis Multiplication

For arithmetic in Galois fields, multiplication (with modulus) can be greatly simplified if Normal Basis representation is adopted to represent operands, especially in element square and multiplication:

Example 1: Element square: In F_{2^n}, we have relation $(a+b)^2 = a^2 + b^2$. Then $(b_0\beta + b_1\beta^2 + b_2\beta^4 + \cdots + b_{n-1}\beta^{2^{n-1}})^2 = b_0^2\beta^2 + b_1^2\beta^4 + b_2^2\beta^8 + \cdots + b_{n-1}^2\beta = b_{n-1}^2\beta + b_0\beta^2 + b_1\beta^4 + \cdots + b_{n-2}\beta^{2^{n-1}}$, which means a simple right-cyclic rotation. One the other hand, standard basis representation do not have this benefit:
In F_{2^3} constructed by $\alpha^3 + \alpha + 1$, standard basis is $\{1, \alpha, \alpha^2\}$. Let $\beta = \alpha^3$, $N = \{\beta, \beta^2, \beta^4\}$ is a normal basis. Write down an element with both representations: $E = a_0 + a_1\alpha + a_2\alpha^2 = b_0\beta + b_1\beta^2 + b_2\beta^4$, and its square in standard basis: $E^2 = a_0 + a_1\alpha^2 + a_2\alpha^4 = a_0 + a_2\alpha + (a_1 + a_2)\alpha^2$. When it is written in normal basis: $E^2 = b_2\beta + b_0\beta^2 + b_1\beta^4$, no additional terms, just simple rotation.

Example 2: Normal basis multiplication: Suppose we have 2 binary vectors representing 2 operands in normal basis: $A = (a_0, a_1, \ldots, a_{n-1}), B = (b_0, b_1, \ldots, b_{n-1})$; similarly the product can also be written as: $C = A * B = (c_0, c_1, \ldots, c_{n-1})$.
Then the highest digit of product can be got with some function: $c_{n-1} = f(a_0, a_1, \ldots, a_{n-1}; b_0, b_1, \ldots, b_{n-1})$. Square both side: $C^2 = A^2 * B^2$, i.e. the second highest digit $c_{n-2} = f(a_{n-1}, a_0, a_1, \ldots, a_{n-2}; b_{n-1}, b_0, b_1, \ldots, b_{n-2})$. By this method it is easy to get all digits of product C.

Example 3: λ-Matrix: We can use a binary $n \times n$ matrix M to describe the "function" mentioned above: $c_{n-1} = f(A, B) = A \cdot M \cdot B^T$, B^T denotes vector transposition. More specifically, we can denote the matrix by *k-th λ-Matrix*: $c_k = A \cdot M^{(k)} \cdot B^T$. Then $c_{k-1} = A \cdot M^{(k-1)} \cdot B^T = rotate(A) \cdot M^{(k)} \cdot rotate(B)^T$, which means by right and down shifting $M^{(k-1)}$ we can get $M^{(k)}$.
In F_{2^3} constructed by $\alpha^3+\alpha+1$, let $\beta = \alpha^3$, $N = \{\beta, \beta^2, \beta^4\}$ is a normal basis. 0-th λ-Matrix

$$M^{(0)} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}. \tag{1}$$

i.e.,

$$c_0 = (a_0\ a_1\ a_2) \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix}. \tag{2}$$

λ-Matrix is defined with cross-product terms from multiplication. That is

$$ProductC = (\sum_{i=0}^{n-1} a_i\beta^{2^i})(\sum_{j=0}^{n-1} b_j\beta^{2^j}) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} a_ib_j\beta^{2^i}\beta^{2^j} \tag{3}$$

The expressions $\beta^{2^i}\beta^{2^j}$ are referred to as cross-product terms, and can be represented by normal basis, i.e.

$$\beta^{2^i}\beta^{2^j} = \sum_{k=0}^{n-1} \lambda_{ij}^{(k)}\beta^{2^k}, \quad \lambda_{ij}^{(k)} \in F_2. \tag{4}$$

Substitution yields, get the expression for k-th digit of product as showed in *Example 3*:

$$c_k = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} \lambda_{ij}^{(k)} a_ib_j \tag{5}$$

$\lambda_{ij}^{(k)}$ is the entry with coordinate (i, j) in k-th λ-Matrix.

From examples above we can see the advantages of using normal basis representation. Then how can we find out normal element based on primitive element? Meanwhile, how to compute λ-Matrix? In the following part we will figure out a kind of normal basis with specific properties, which will make λ-Matrix easy to calculate.

II. Optimal Normal Basis

A. Complexity & ONB Definition

The number of non-zero entries in λ-Matrix is known as *Complexity*(C_N).
To define optimal normal basis, it is necessary to find out the minimum number of non-zero terms in λ-matrix.
Theorem If N is a normal basis for F_{p^n} with λ-matrix $M^{(k)}$, then non-zero entries in matrix $C_N \geq 2n-1$.

Proof. Let $N = \{\beta, \beta^p, \beta^{p^2}, \ldots, \beta^{p^{n-1}}\}$, denote β^{p^i} by β_i. Then $\sum_{i=0}^{n-1} \beta_i = trace\ \beta$.
Denote $trace\ \beta$ with b, consider $n \times n$ matrix $M^{(0)}$. Then $b\beta_0 = \sum_{i=0}^{n-1} \beta\beta_i$.
Therefore, the sum of all rows in $M^{(0)}$ is an n-tuple with b as the first element and zeros elsewhere. So the first column must contain at least 1 nonzero element, and other columns must get a sum of zero while keeping every row linearly independent so there should be at least 2 non-zero elements each column. □

If there exists a set of normal basis satisfying $C_N = 2n-1$, this normal basis is named as *Optimal Normal Basis*.

Example 4: In F_{2^4} constructed with $\alpha^4 + \alpha + 1$, 2 sets of normal basis can be found: $\beta = \alpha^3, N = \{\alpha^3, \alpha^6, \alpha^{12}, \alpha^9\}$ and $\beta = \alpha^7, N = \{\alpha^7, \alpha^{14}, \alpha^{13}, \alpha^{11}\}$. Multiplication table for $\beta = \alpha^3$:

$$T_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{6}$$

Complexity C_N = 7, this is optimal. For $\beta = \alpha^7$:

$$T_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \tag{7}$$

Complexity C_N = 9, this is NOT optimal.

There should be an efficient method to construct optimal normal basis. Following contents mainly come from refernce book[2].

B. Type-I Optimal Normal Basis

Rules for creating Type-I ONB over F_{2^n} are:

- n+1 must be prime.
- 2 must be primitive in $\mathbb{Z}_{n+1}$.

Second rule means 2 raise to any power 0 $\sim$ n-1 (mod n+1) must cover every integer 1 $\sim$ n.

$\lambda_{ij}^{(k)}$ is the entry with coordinate (i, j) from k-th λ-Matrix. Then the crossproduct term can be written as

$$\beta^{2^i}\beta^{2^j} = \sum_{k=0}^{n-1} \lambda_{ij}^{(k)} \beta^{2^k} \tag{8}$$

Suppose we only care about k=0. So simplified to following equations:

$$\begin{cases} \beta^{2^i}\beta^{2^j} = \beta & (9) \\ \beta^{2^i}\beta^{2^j} = 1 \quad (if \quad 2^i \equiv 2^j \mod n+1) & (10) \end{cases}$$

solution (i, j) implies entries equal to "1" in matrix. If β is already an optimal normal element, β^{2^i} counts through all powers of β and generates our basis. So

$$\begin{cases} 2^i + 2^j = 1 \mod n+1 & (11) \\ 2^i + 2^j = 0 \mod n+1 & (12) \end{cases}$$

have the same solution for $M^{(0)}$. Note this method can be applied even both primitive polynomial and normal element are unkown (but existance and adoption of optimal normal basis must be guaranteed).
If do similar thing, equations for multiplication table can also be obtained. Based on this, a transformation relation between λ-Matrix and Multiplication table can be concluded. However, the proof is still a **problem**. *BTW, example 4 shows a type-I ONB over* F_{2^4}.

C. Type-II Optimal Normal Basis

Rules for creating Type-II ONB over F_{2^n} are:

- 2n+1 must be prime. And either
- 2 must be primitive in $\mathbb{Z}_{2n+1}$, OR
- $2n + 1 \equiv 3 \mod 4$ AND 2 generates the quadratic residues in $\mathbb{Z}_{2n+1}$

The last rule means: The last 2 bits are set in the binary representation of prime 2n+1; even if $2^k \mod 2n+1$ does not generate every element in 0 $\sim$ 2n, we can at least take the square root $\mod 2n+1$ of 2^k. (Need to learn more on quadratic residues)
To generate a Type-II ONB, pick an element γ of order 2n+1 in $F_{2^{2n}}$, then use this to find optimal normal element from F_{2^n}: $\beta = \gamma + \gamma^{-1}$. Crossproduct terms

$$\begin{aligned} \beta^{2^i}\beta^{2^j} &= (\gamma^{2^i} + \gamma^{-2^i})(\gamma^{2^j} + \gamma^{-2^j}) \\ &= (\gamma^{2^i+2^j} + \gamma^{-(2^i+2^j)}) + (\gamma^{2^i-2^j} + \gamma^{-(2^i-2^j)}) \\ &= \begin{cases} \beta^{2^k} + \beta^{2^{k\prime}} & if\ 2^i \neq 2^j \mod 2n+1 \\ \beta^{2^k} & if\ 2^i \equiv 2^j \mod 2n+1 \end{cases} \end{aligned} \tag{13}$$

k and $k\prime$ are the 2 possible solutions to multiplication of any 2 basis elements. That's what makes this normal basis optimal: it has the minimum number of possible terms. In the case of $2^i \neq 2^j \mod 2n+1$, at least one of these equations:

$$\begin{cases} 2^i + 2^j = 2^k \mod 2n+1 \\ 2^i + 2^j = -2^k \mod 2n+1 \end{cases} \tag{14}$$

will have a solution, and at least one of these equations:

$$\begin{cases} 2^i - 2^j = 2^{k\prime} \mod 2n+1 \\ 2^i - 2^j = -2^{k\prime} \mod 2n+1 \end{cases} \tag{15}$$

also has a solution.
In the case of $2^i \equiv \pm 2^j \mod 2n+1$, at least one of the following 4 equations has a solution:

$$\begin{cases} 2^i + 2^j = 2^k \mod 2n+1 \\ 2^i + 2^j = -2^k \mod 2n+1 \\ 2^i - 2^j = 2^k \mod 2n+1 \\ 2^i - 2^j = -2^k \mod 2n+1 \end{cases} \tag{16}$$

In the first set of equations, there are two possible solutions, and in the second set of equations, there is only one possible solution. It is easy to see that the equations are all similar, so instead of working with 2 different sets we can combine them and work with just one group of 4 equations. To build our λ-Matrix, we set k=0 and find solutions to:

$$\begin{cases} 2^i + 2^j = 1 \mod 2n+1 \\ 2^i + 2^j = -1 \mod 2n+1 \\ 2^i - 2^j = 1 \mod 2n+1 \\ 2^i - 2^j = -1 \mod 2n+1 \end{cases} \tag{17}$$

Example 5: Following this criteria, easy to find $M^{(0)}$ for F_{2^5}:

$$M^{(0)} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{18}$$

More details about proof can be found in [3], which is believed to be first paper on ONB theory.
Now from the simplified criteria we get, it is not expensive any more to compute a λ-Matrix. Moreover, one advantage adopting this criteria is that for multiplier designers, they can design without knowing specific normal element (althought this optimal normal element is unique when primitive polynomial is set). However, for our approach it is necessary to find out what optimal normal element we are using. So the problem is: If field extension and primitive polynomial (F_{2^n} and $f(\alpha)$) is given, and we already get the λ-Matrix (which means we know a type I or II optimal normal basis is adopted), is it possible to find an efficient way to compute corresponding optimal normal element?

III. Normal Basis Multiplier

A. SMPO Design

This part mainly comes from *ref2:SMPO*. [4]
Based on the normal basis multiplication in Example 2, Massey and Omura designed a Sequential Multiplier with Serial Output, get every bit of output by shifting operands 1 bit per clock.

For 5-bit SMSO, based on λ-Matrix we got from *Example 5*, consider following equation

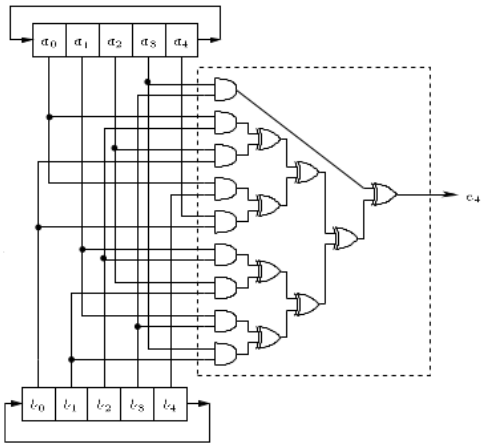

Fig. 1. Massey-Omura SMSO for m-bit

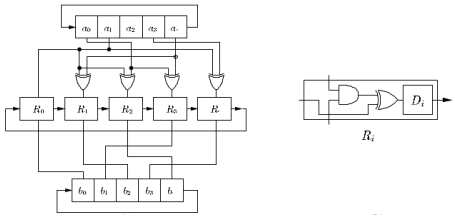

Fig. 2. 5-bit SMPO design

$c_i = b_i a_{i+1} + b_{i+1}(a_i + a_{i+3}) + b_{i+2}(a_{i+3} + a_{i+4}) + b_{i+3}(a_{i+1} + a_{i+2}) + b_{i+4}(a_{i+2} + a_{i+4}), 0 \le i \le 4$
It is possible to calculate every product term within one clock. Also utilize cyclic similarity, we have SMPO design:

Example 6: Sequential Multiplier Protocol:

- **Initial** $R_0 = R_1 = R_2 = R_3 = R_4 = 0$
- **Clock 1** $R_0 = a_1 b_0, R_1 = b_2(a_1 + a_4), R_2 = b_4(a_0 + a_1), R_3 = b_1(a_4 + a_0), R_4 = b_3(a_1 + a_3)$
- **Clock 2** $R_0 = b_3(a_1 + a_3) + a_0 b_4, R_1 = a_1 b_0 + b_1(a_0 + a_3), R_2 = b_2(a_1 + a_4) + b_3(a_4 + a_0), R_3 = b_4(a_0 + a_1) + b_0(a_3 + a_4), R_4 = b_1(a_4 + a_0) + b_2(a_0 + a_2)$
- ...
- **Clock 5** $R_0 = c_0, R_1 = c_1, R_2 = c_2, R_3 = c_3, R_4 = c_4$, i.e. $R = A \cdot B$.

B. Adopting ONB

Easy to find out complexity C_N is linearly dependent to terms in equation above, which determines the number of gates, i.e. the delay and area of multiplier circuit. So it's necessary to keep the complexity minimum for an optimal design.
There are 2 types of optimal normal basis (about optimal normal basis, refer to *ref3:ONB*)
Type I ONB: n+1 is prime and q is primitive in $\mathbb{Z}_{n+1}$, then the n nonunit (n+1)th roots of unity form ONB.

Type II ONB: 2n+1 is prime, if (1) 2 is primitive in $\mathbb{Z}_{2n+1}$, OR

(2) $2n+1 \equiv 3 \pmod 4$ and 2 generates the quadratic residues in $\mathbb{Z}_{2n+1}$.
Then $\beta = \gamma + \gamma^{-1}$ generates ONB, where γ is a primitive (2n+1)th root if unity.

Can we use these definitions to compute optimal normal element within $O(1)$?

IV. Methodology

- **Step 1** Calculate Multiplication table, using equation ((i,j) is coordinate in matrix)
  $2^i + 2^j \equiv 0 \text{ or } 1 \pmod{(2n+1)}$ (Type-I) OR
  $2^i \pm 2^j \equiv \pm 1 \pmod{(2n+1)}$; (Type-II)[5]
- **Step 2** Compute normal element $\beta = \alpha^k, 1 \leq k \leq 2^n$;
- **Step 3** Generate bit-level polynomials from multiplication table, word-level polynomials from normal element, and vanishing polynomials;
- **Step 4** Compose one ideal, input is current state r, operands A and B, arrange refined term ordering and compute Gröbner basis to abstract relation output $R + f(A, B)$;(problem here)
- **Step 5** Shift input operands A and b by 1 bit, feed output R to input r, repeat step 4 for n times, output is n-bit product $R + A * B$.

References

[1] Shuhong Gao. *Normal bases over finite fields.* Citeseer, 1993.

[2] Michael Rosing. *Implementing elliptic curve cryptography*. Manning New York, 1999.

[3] Ronald C Mullin, Ivan M Onyszchuk, Scott A Vanstone, and Richard M Wilson. Optimal normal bases in gf (p¡ sup¿ n¡/sup¿). *Discrete Applied Mathematics*, 22(2):149–161, 1989.

[4] Arash Reyhani-Masoleh and M Anwar Hasan. Low complexity word-level sequential normal basis multipliers. *Computers, IEEE Transactions on*, 54(2):98–110, 2005.

[5] Turki F Al-Somani and Alaaeldin Amin. Hardware implementations of gf (2m) arithmetic using normal basis. *Journal of Applied Sciences*, 6(6):1362–1372, 2006.