EFFICIENT GRÖBNER BASIS REDUCTIONS FOR FORMAL VERIFICATION OF GALOIS FIELD MULTIPLIERS

Jinpeng Lv and Priyank Kalla
Department of Electrical and Computer Eng.
University of Utah, Salt Lake City, UT-84112

Florian Enescu
Department of Mathematics and Statistics
Georgia State University, Atlanta, GA 30302-4038

Abstract - *Galois field arithmetic finds application in many areas, such as cryptography, error correction codes, signal processing, etc. Multiplication lies at the core of most Galois field computations. This paper addresses the problem of formal verification of hardware implementations of (modulo) multipliers over Galois fields of the type $\mathbb{F}_{2^k}$, using a computer-algebra/algebraic-geometry based approach. The multiplier circuit is modeled as a polynomial system in $\mathbb{F}_{2^k}[x_1,x_2,\cdots,x_d]$ and the verification problem is formulated as a membership test in a corresponding (radical) ideal. This requires the computation of a Gröbner basis, which can be computationally intensive. To overcome this limitation, we analyze the circuit topology and derive a term order to represent the polynomials. Subsequently, using the theory of Gröbner bases over Galois fields, we prove that this term order renders the set of polynomials itself a Gröbner basis of this ideal – thus significantly improving verification. Using our approach, we can verify the correctness of, and detect bugs in, upto 163-bit circuits in $\mathbb{F}_{2^{163}}$; whereas contemporary approaches are infeasible.*

I. INTRODUCTION

Finite (Galois) field theory is extensively applied in Elliptic Curve Cryptography (ECC), error correction codes, digital signal processing, etc. Therefore, dedicated hardware implementations of Galois field arithmetic abound. Multiplication lies at the core of most Galois field computations and is a high-complexity operation, particularly in cryptosystems where the datapath size can be very large. For example, $k = 163$ for a Galois field recommended for ECC by the U.S. National Institute for Standards and Technology (NIST). Therefore, this has lead researchers to derive efficient and sophisticated hardware implementations of multiplier circuits over Galois Fields [1] [2] [3].

The complicated and specialized nature of these arithmetic architectures requires custom design, raising the potential for errors and bugs in the implementation. Such errors not only cause unintended operation, but they also manifest themselves as security vulnerabilities which can be maliciously exploited [4]. Arithmetic bugs are especially catastrophic for cryptosystems, as incorrect (buggy) hardware multiplication can lead to full leakage of the secret key [5]. Therefore, it is of utmost importance to verify the correctness of hardware implementations of finite field multipliers residing at the core of such systems. This paper addresses formal verification of multiplier circuits over (binary) Galois fields of the type $\mathbb{F}_{2^k}$ using computer algebra techniques.

This work is sponsored in part by a grant from NSF #CCF-546859.

Problem Statement: We are given the following:

- Given a finite field $\mathbb{F}_{2^k}$, i.e. given k (circuit input size), along with the corresponding irreducible polynomial $P(x)$. Let $P(\alpha)=0$, i.e. α be the root of $P(x)$.
- Given a word-level multiplier specification polynomial $Z = A\cdot B \pmod{P(x)}$, where $A,B,Z \in \mathbb{F}_{2^k}$ (k-bit vectors).
- Given a gate-level combinational circuit C. The bit-level primary inputs of the circuit are $\{a_0,\ldots,a_{k-1},\ b_0,\ldots,b_{k-1}\}$, and $\{z_0,\ldots,z_{k-1}\}$ are the primary outputs; here $a_i,b_i,g_i \in \mathbb{F}_2, i=0,\ldots,k-1$. Therefore, $A = a_0 + a_1\alpha + \cdots + a_{k-1}\alpha^{k-1}$, $B = b_0 + b_1\alpha+\cdots+b_{k-1}\alpha^{k-1}$ and $Z = z_0+z_1\alpha+\cdots+z_{k-1}\alpha^{k-1}$.

Our *objective* is to formally prove that $\forall A,B \in \mathbb{F}_{2^k}$, the circuit C correctly computes the multiplication $Z = A\cdot B \pmod{P(x)}$ over $\mathbb{F}_{2^k}$. Otherwise, we have to produce a counter-example that excites the bug in the design.

We formulate and solve the verification problem using a computer-algebra and algebraic-geometry based approach. In particular, we apply the results of *Strong Nullstellensatz over Galois fields*, along with *Gröbner basis techniques*, to efficiently perform the equivalence verification between the specification polynomial and the multiplier circuit implementation. Our *approach and contributions* can be outlined as follows:

- We model the given circuit C as a polynomial system $\{f_1,\ldots,f_s\}$ in $\mathbb{F}_{2^k}[x_1,x_2,\ldots,x_d]$, and denote the generated ideal as $J = \langle f_1,\ldots,f_s\rangle$. The specification $Z = A\cdot B \pmod{P(x)}$ is also modeled as a polynomial $f : Z + A\cdot B$ in $\mathbb{F}_{2^k}[x_1,x_2,\ldots,x_d]$.
- Using the results of Strong Nullstellensatz over finite fields [6], we show that the verification test can be formulated as membership testing of f in the ideal $(J+J_0)$, where $J_0 = \langle x_1^{2^k}-x_1,\ldots,x_d^{2^k}-x_d\rangle$.
- For this ideal membership test, it is required to compute a Gröbner basis of $(J+J_0)$. Buchberger's algorithm [7], employed for Gröbner basis computation, can be computationally intensive, and its efficiency is highly susceptible to the term orderings used to represent and manipulate the polynomials. To overcome this complexity, we draw inspirations from [8], and derive a term order by analyzing the circuit topology to represent the polynomials.
- The work of [8] shows that this term order makes the set $\{f_1,\ldots,f_s\}$ a Gröbner basis of J. However, *using the theoretical concepts of Gröbner bases over Galois fields, we further prove that this term order also makes the set $\{f_1,\ldots,f_s,\ x_1^{2^k}-x_1,\ldots,x_d^{2^k}-x_d\}$ a Gröbner basis of $J+J_0$.* This significantly enhances the capacity of our

verification framework, enabling verification of custom implementations of 163-bit circuits – corresponding to the NIST recommended 163-bit elliptic curve.
- Not only can we verify the correctness of multiplier circuits, but we can also detect the presence of bugs using our approach. Moreover, the time required to prove correctness or to detect bugs is comparable. Our experiments show that none of the contemporary techniques based on BDDs, SAT, SMT-solvers etc. can prove correctness beyond 16-bit circuits.

Our approach is generic enough to verify the implementation of any Galois field arithmetic circuit against its given polynomial specification. However, for this paper, we concentrate only on multiplier circuits over Galois fields $\mathbb{F}_{2^k}$.

II. Galois Fields & Multiplication

We briefly describe the relevant finite field concepts (details can be found in the textbook [9]) and modular multiplier design over such fields [1] [2] [3].

A finite field, also called a Galois field, is a field with a finite number of elements. The number of elements q of the finite field is a power of a prime integer – i.e. $q = p^k$, where p is prime integer, and $k \geq 1$. Galois fields are denoted as $\mathbb{F}_q$ and also $GF(q = p^k)$. We are interested in fields where $p = 2$ and $k > 1$; i.e. *binary Galois extension fields* $\mathbb{F}_{2^k}$, as these are often employed in elliptic curve cryptography implementations.

To construct $\mathbb{F}_{2^k}$, we take the polynomial ring $\mathbb{F}_2[x]$ and an irreducible polynomial $P(x) \in \mathbb{F}_2[x]$ of degree k, and construct $\mathbb{F}_{2^k}$ as $\mathbb{F}_2[x] \pmod{P(x)}$. For example, $\mathbb{F}_8 = \mathbb{F}_2[x] \pmod{x^3 + x + 1}$. All the field operations are performed modulo the irreducible polynomial $P(x)$ and the coefficients are reduced modulo $p = 2$. Any element $A \in \mathbb{F}_{2^k}$ can be represented in polynomial form as $A = a_0 + a_1\alpha + \cdots + a_{k-1}\alpha^{k-1}$, where $a_i \in \mathbb{F}_2 = \{0,1\}$ and $P(\alpha) = 0$. For all elements $A \in \mathbb{F}_q, A^q = A$, and hence $A^q - A = 0, \forall A \in \mathbb{F}_q$.

Multiplier circuit design over $\mathbb{F}_{2^k}$ is illustrated below.

Example 2.1: Consider the field $\mathbb{F}_{2^4}$. We take as inputs: $A = a_0 + a_1 \cdot \alpha + a_2 \cdot \alpha^2 + a_3 \cdot \alpha^3$ and $B = b_0 + b_1 \cdot \alpha + b_2 \cdot \alpha^2 + b_3 \cdot \alpha^3$, along with the irreducible polynomial $P(x) = x^4 + x^3 + 1$. We have to perform the multiplication $Z = A \times B \pmod{P(x)}$. The coefficients of $A = \{a_0, \ldots, a_3\}, B = \{b_0, \ldots, b_3\}$ are in $\mathbb{F}_2 = \{0,1\}$. Multiplication can be performed as shown below:

			a_3	a_2	a_1	a_0
$\times$			b_3	b_2	b_1	b_0
			$a_3 \cdot b_0$	$a_2 \cdot b_0$	$a_1 \cdot b_0$	$a_0 \cdot b_0$
		$a_3 \cdot b_1$	$a_2 \cdot b_1$	$a_1 \cdot b_1$	$a_0 \cdot b_1$	
	$a_3 \cdot b_2$	$a_2 \cdot b_2$	$a_1 \cdot b_2$	$a_0 \cdot b_2$		
$a_3 \cdot b_3$	$a_2 \cdot b_3$	$a_1 \cdot b_3$	$a_0 \cdot b_3$			
s_6	s_5	s_4	s_3	s_2	s_1	s_0

The result $Sum = s_0 + s_1 \cdot \alpha + s_2 \cdot \alpha^2 + s_3 \cdot \alpha^3 + s_4 \cdot \alpha^4 + s_5 \cdot \alpha^5 + s_6 \cdot \alpha^6$, where, $s_0 = a_0 \cdot b_0$, $s_1 = a_0 \cdot b_1 + a_1 \cdot b_0$, $s_2 = a_0 \cdot b_2 + a_1 \cdot b_1 + a_2 \cdot b_0$, and so on. Here the multiply "$\cdot$" and add "+" operations are performed modulo 2, so they can be implemented in a circuit using AND and XOR gates. Note that unlike integer multipliers, there are no carry-chains in the design, as the coefficients are always reduced modulo $p = 2$. However, the result is yet to be reduced modulo the primitive polynomial $P(x) = x^4 + x^3 + 1$. This is shown below:

s_3	s_2	s_1	s_0	
s_4	0	0	s_4	$\Leftarrow s_4 \cdot \alpha^4 \pmod{P(\alpha)} = s_4 \cdot (\alpha^3 + 1)$
s_5	0	s_5	s_5	$\Leftarrow s_5 \cdot \alpha^5 \pmod{P(\alpha)} = s_5 \cdot (\alpha^3 + \alpha + 1)$
s_6	s_6	s_6	s_6	$\Leftarrow s_6 \cdot \alpha^6 \pmod{P(\alpha)} = s_6 \cdot (\alpha^3 + \alpha^2 + \alpha + 1)$
z_3	z_2	z_1	z_0	

The final result (output) of the circuit is: $Z = z_0 + z_1\alpha + z_2\alpha^2 + z_3\alpha^3$; where $z_0 = s_0 + s_4 + s_5 + s_6$; $z_1 = s_1 + s_5 + s_6$; $z_2 = s_2 + s_6$; $z_3 = s_3 + s_4 + s_5 + s_6$.

The above multiplier design is called the *Mastrovito multiplier* [1]. In cryptosystems, multiplication is often performed repeatedly – e.g., for exponentiation. For such applications, Montgomery multiplier architectures [2] [3] over Galois fields are employed for faster computation (at the expense of area). In our experiments, we verify custom implementations of both Mastrovito and Montgomery multipliers, where the circuits are given as gate-level (flattened) netlists.

III. Related Previous Work

Contemporary graph-based canonical DAG representations of Boolean functions such as BDDs [10], OKFDDs [11], BMDs [12] and MODDs [13], etc. are ill-suited for such modulo-multiplication applications, particularly over large finite fields. The work of [14] presents a DAG representation for synthesis and verification of multi-output polynomials over finite integer rings $\mathbb{Z}_{2^k}, k > 1$. Since their canonical reduction rules employ the theory of polynomial functions over finite integer rings [15], this approach is not directly applicable over finite fields $\mathbb{F}_{2^k}$. Similarly, the work of [16] is also only applicable for verification of *integer-modulo-arithmetic* over $\mathbb{Z}_{2^k}$ at word-level/RTL, and not over Galois field circuits.

This verification problem is also very hard for SAT solvers, due to the large circuit size, and the presence of AND-XOR computations. Contemporary Satisfiability Modulo Theory (SMT) solvers employ a mixture of theories for reasoning – however, none of them employ polynomial equation solving over Galois fields (which is itself a hard problem). As shown in out experiments, BDDs, SAT and SMT solvers cannot prove design correctness beyond 16-bit multipliers.

The theorem-proving approach of [17] also verifies a Galois Field $\mathbb{F}_{2^k}$ implementation against a given polynomial specification. They devise a decision procedure based on polynomial division, variable elimination, term re-writing, etc., and demonstrate a correctness proof of a sub-block of a Reed-Solomon decoder. The work of [18] solves similar problems as those of [17]. However, they make use of OKFDDs [11] to canonically represent the circuit constraints. Moreover, instead of verifying circuits over $\mathbb{F}_{2^k}$ directly, [18] verifies the circuits over its equivalent composite field $GF((2^m)^n)$ representation, where a *non-prime* $k = m \cdot n$. Their approach has no benefit if k is prime – say, when $k = 163$ for elliptic curves. Moreover, the size-explosion of FDDs limits their approach to 16-bit ($\mathbb{F}_{2^{16}}$) circuits, as shown in their experiments.

The work of [19] verifies a given composite field $GF((2^m)^n)$ multiplier against its specification using a Gröbner basis approach based on the *Weak Nullstellensatz* over $\mathbb{F}_q$ – by proving

that the "specification $\neq$ implementation" (miter) is infeasible. However, their approach has the limitation in that it requires the circuit decomposition hierarchy be made available to the verification engine. Moreover, when k in $\mathbb{F}_{2^k}$ is prime, this approach is inapplicable. Recent work of [20] formulates the verification problem in the similar style as [19]. They *heuristically* derive a variable order to represent the monomial terms. Using their heuristic, they are able to speed-up the Gröbner basis computation. However, in the presence of bugs, their heuristic is inefficient in that the Gröbner basis computation runs into memory explosion beyond 16-bit circuits.

The paper [8] addresses verification of finite precision integer datapath circuits using the concepts of Gröbner bases over the ring $\mathbb{Z}_{2^k}$. They model the circuit constraints by way of arithmetic-bit-level (ABL) polynomials ($\{G\}$), and formulate the verification test as an equivalent variety subset problem. To solve this, first they derive a term order that already makes $\{G\}$ a Gröbner basis. Then they compute a normal form f of the specification g w.r.t. $\{G\}$. If f is a vanishing polynomial over $\mathbb{Z}_{2^k}$ [16], circuit correctness is established. In [21], the authors further show that the vanishing polynomial test can be omitted by formulating the problem directly over $Q := \mathbb{Z}_{2^k}[X]/\langle x^2 - x : x \in X\rangle$.

Our overall problem formulation is similar to those of [8] [21]. However, *our application and computational domains are different*: we have to verify the circuits over the Galois field $\mathbb{F}_{2^k}$, as opposed to $\mathbb{Z}_{2^k}$ in [8] [21]. Therefore, working over Galois fields, using the Strong Nullstellensatz over $\mathbb{F}_q$, we show how to directly formulate the problem as an ideal membership test over the (radical) ideal $J+J_0$; where J corresponds to the ideal generated by the circuit constraints and J_0 denotes the ideal of all vanishing polynomials over $\mathbb{F}_{2^k}$. We apply the same term order from Proposition 2 in [8]; this of course makes the set of polynomials $\{f_1,\ldots,f_s\}$ extracted from the circuit a Gröbner basis of J. However, *we further prove that it also renders the set* $\{f_1,\ldots,f_s,\ x_1^q-x_1,\ldots,x_d^q-x_d\}$ *a Gröbner basis of* $J+J_0$ *over* $\mathbb{F}_q$ – which is a new contribution. Subsequently, the verification test can be carried out directly via Gröbner basis reduction.

IV. Ideals, Varieties & Nullstellensatz

Let $\mathbb{F}$ be any field and $\mathbb{F}[x_1,\ldots,x_d]$ the polynomial ring over $\mathbb{F}$ with indeterminates $x_1,\ldots,x_d$. An *ideal* generated by $f_1,\ldots,f_s \in \mathbb{F}[x_1,\ldots,x_d]$ is $\langle f_1,\ldots,f_s\rangle = \{h = \sum_{i=1}^{s} g_i \cdot f_i :\ g_i \in \mathbb{F}[x_1,\ldots,x_d]\}$. Let $\mathbf{a} \in \mathbb{F}^d$ be a point, and $f \in \mathbb{F}[x_1,\ldots,x_d]$ be a polynomial. We say that f *vanishes* on $\mathbf{a}$ if $f(\mathbf{a}) = 0$.

For any subset J of $\mathbb{F}[x_1,\ldots,x_d]$, the *affine variety* of J over $\mathbb{F}$ is $V(J) = \{\mathbf{a} \in \mathbb{F}^d : \forall f \in J, f(\mathbf{a}) = 0\}$.

Definition 4.1: For any subset V of $\mathbb{F}^d$, the ideal of polynomials that vanish on V, called the *vanishing ideal of* V, is defined as $I(V) = \{f \in \mathbb{F}[x_1,\ldots,x_d] : \forall \mathbf{a} \in V, f(\mathbf{a}) = 0\}$.

If a polynomial f vanishes on a variety V, then $f \in I(V)$.

Definition 4.2: Let $J \subset \mathbb{F}[x_1,\ldots,x_d]$ be an ideal. The *radical of* J is defined as $\sqrt{J} = \{f \in \mathbb{F}[x_1,\ldots,x_d] : \exists m \in \mathbb{N}, f^m \in J\}$.

When $J = \sqrt{J}$, J is said to be a *radical ideal* and $I(V)$ is a radical ideal. The strong Nullstellensatz establishes the correspondence between radical ideals and varieties over $\mathbb{F}_q$.

Theorem 4.1: (*Strong Nullstellensatz* [22]) Let $\mathbb{F}$ be an arbitrary field and J be an ideal in $\mathbb{F}[x_1,\ldots,x_d]$. Let $\overline{\mathbb{F}}$ denote the algebraic closure of $\mathbb{F}$, and let $V_{\overline{\mathbb{F}}}(J)$ denote the variety of J over $\overline{\mathbb{F}}$. Then $I(V_{\overline{\mathbb{F}}}(J)) = \sqrt{J}$.

The Nullstellensatz admits a special form over Galois fields. We state the following results of Nullstellensatz in Galois fields, proofs of which can be found in [6].

Lemma 4.1: [6] For any ideal $J \subseteq \mathbb{F}_q[x_1,\ldots,x_d]$, $J+J_0 = J+\langle x_1^q-x_1,\ldots,\ x_d^q-x_d\rangle$ is radical. In other words, $\sqrt{J+J_0} = J+J_0$.

Theorem 4.2: [6] For any Galois field $\mathbb{F}_q$, let $J \subseteq \mathbb{F}_q[x_1,\ldots,x_d]$ be an ideal. Let $V_{\mathbb{F}_q}(J)$ denote the variety of J over $\mathbb{F}_q$. Then, $I(V_{\mathbb{F}_q}(J)) = J+J_0 = J+\langle x_1^q-x_1,\ldots,\ x_d^q-x_d\rangle$.

V. Verification Problem Formulation

Notation: In the sequel, we denote the Galois field as $\mathbb{F}_q$, and its algebraic closure as $\overline{\mathbb{F}_q}$, where $q = 2^k$ for our application. Moreover, $V_{\mathbb{F}_q}(J)$ denotes the variety of J over the field $\mathbb{F}_q$, and $V_{\overline{\mathbb{F}_q}}(J)$ denotes the variety over it algebraic closure.

We are given a specification polynomial $f : Z + A \cdot B$ where $A = a_0 + a_1\alpha + \cdots + a_{k-1}\alpha^{k-1}, B = b_0 + b_1\alpha + \cdots + b_{k-1}\alpha^{k-1}$ and $Z = z_0 + z_1\alpha + \cdots + z_{k-1}\alpha^{k-1}$. We are also given a gate-level circuit C with $\{a_0,\ldots,a_{k-1},\ b_0,\ldots,b_{k-1}\}$ as primary inputs and $\{z_0,\ldots,z_{k-1}\}$ as the primary outputs. We have to check if the circuit C implements the function f over the given field $\mathbb{F}_q$.

We analyze the circuit and model the Boolean gate-level operators as polynomials over $\mathbb{F}_2$ ($\subset \mathbb{F}_q$), as shown in Example 2.1. Besides, we append the following three polynomials: $A + a_0 + a_1\alpha + \cdots + a_{k-1}\alpha^{k-1},\ B + b_0 + b_1\alpha + \ldots b_{k-1}\alpha^{k-1},\ Z + z_0 + z_1\alpha + \cdots + z_{k-1}\alpha^{k-1}$; these polynomials specify the correspondence between the bit-level ($\mathbb{F}_2$) and the word-level ($\mathbb{F}_q$) variables in the system. We denote these constraints as polynomials $\{f_1,\ldots,f_s\}$ over the ring $\mathbb{F}_q[x_1,\ldots,x_d]$, and denote the generated ideal as $J = \langle f_1,\ldots,f_s\rangle$. Similarly, the specification polynomial $f \in \mathbb{F}_q[x_1,\ldots,x_d]$. As opposed to integer coefficients in [8] ($\mathbb{Z}_{2^k}$), we have coefficients $(\alpha, \alpha^2, \ldots)$ from the field $\mathbb{F}_q$.

To prove that the specification (f) matches the implementation (J) at all points in the design space, we need to check *whether or not* f *vanishes on the variety* $V_{\mathbb{F}_q}(J)$. This is because for all $p \in V_{\mathbb{F}_q}(J)$, if $f(p) = 0$, then $Z + A \cdot B = 0 \implies Z = A \cdot B$. On the other hand, if $f(p) \neq 0$ for some point p, then p corresponds to the bug in the design. Now if f vanishes on $V_{\mathbb{F}_q}(J)$, we know that f should be a member of $I(V_{\mathbb{F}_q}(J))$. Strong Nullstellensatz over $\mathbb{F}_q$ (Theorem 4.2) tells us that $I(V_{\mathbb{F}_q}(J)) = J + J_0 = \langle f_1,\ldots,f_s,\ x_1^q-x_1,\ldots,x_d^q-x_d\rangle$. Therefore, *we need to test whether or not* f *is a member of the ideal* $J+J_0$. If $f \in (J+J_0)$, correctness of the circuit is proved. Otherwise, there is a bug in the design. To test if $f \in (J+J_0)$, it is required to compute a Gröbner basis of $J+J_0$.

Gröbner Basis of Ideals: Buchberger's algorithm [7], shown in Algorithm 1, is used to compute a Gröbner basis over a field. Given polynomials $F = \{f_1,\ldots,f_s\}$, the algorithm computes

another set of polynomials $G = \{g_1, \ldots, g_t\}$ such that ideal $I = \langle f_1, \ldots, f_s \rangle = \langle g_1, \ldots, g_t \rangle$. In the algorithm,

$$Spoly(f,g) = \frac{L}{lt(f)} \cdot f - \frac{L}{lt(g)} \cdot g$$

where $L = \text{LCM}(lm(f), lm(g))$, where $lm(f)$ is the leading monomial of f, and $lt(f)$ denotes the leading term of f.

The set G has nice properties that allow to solve many polynomial decision questions. For our context, Gröbner basis provides a decision procedure to test for membership in an ideal. The set G is a Gröbner basis of ideal I, if and only if for all $f \in I$, dividing f by polynomials of G gives 0 remainder: i.e. $G = GB(I) \iff \forall f \in I, f \xrightarrow{G}_+ 0$.

Input: : $F = \{f_1, \ldots, f_s\}$
Output: : $G = \{g_1, \ldots, g_t\}$
  $G := F$;
  REPEAT
    $G' := G$
    For each pair $\{f,g\}, f \neq g$ in G' DO
      $Spoly(f,g) \xrightarrow{G'}_+ r$
      IF $r \neq 0$ THEN $G := G \cup \{r\}$
  UNTIL $G = G'$

Algorithm 1: Buchberger's Algorithm

We now have a complete solution to our problem: We can take the polynomials $\{f_1, \ldots, f_s,\ x_1^q - x_1, \ldots, x_d^q - x_d\}$ and compute a Gröbner basis G for the ideal $J + J_0$. Then, we reduce the specification polynomial f w.r.t. G. If $f \xrightarrow{G}_+ 0$, then the circuit is correct, otherwise there is a bug.

Complexity of Gröbner basis: For our specific problem of computing a Gröbner basis for $J + J_0$ over $\mathbb{F}_q$, the following result is known [6]:

Theorem 5.1: Let $I = \langle f_1, \ldots, f_s,\ x_1^q - x_1, \ldots, x_d^q - x_d \rangle \subset \mathbb{F}_q[x_1, \ldots, x_d]$ be an ideal. The time and space complexity of Buchberger's algorithm to compute a Gröbner basis of I is bounded by $q^{O(d)}$ assuming that the length of input $f_1, \ldots, f_s$ is dominated by $q^{O(d)}$.

In our case $q = 2^k$, and when k and d are large, this complexity may make verification infeasible. In what follows, we show that a term order can be derived which makes $\{f_1, \ldots, f_s, x_1^q - x_1, \ldots, x_d^q - x_d\}$ a Gröbner basis – obviating the need to apply Buchberger's algorithm.

VI. Obviating Buchberger's Algorithm

To improve Buchberger's algorithm, variations of the chain and product criteria are applied.

Lemma 6.1: [Product Criterion [23]] Let $f, g \in \mathbb{F}[x_1, \cdots, x_d]$ be polynomials. If the equality $lm(f) \cdot lm(g) = LCM(lm(f), lm(g))$ holds, then $Spoly(f,g) \xrightarrow{G}_+ 0$.

The above result states that when the leading monomials of f, g are relatively prime, then $Spoly(f,g)$ always reduces to 0 modulo G. Thus $Spoly(f,g)$ need not be considered in Buchberger's algorithm. If we could analyze the given circuit and derive a term order such that every polynomial pair (f,g) in the generating set has relatively prime leading monomials, then $Spoly(f,g) \xrightarrow{G}_+ 0$ for all pairs f, g. Consequently, the polynomials $\{f_1, \ldots, f_s\}$ extracted from the circuit (corresponding ideal J) and represented using such a term order would itself constitute a Gröbner basis of J. In [8], the authors derive exactly such a term order, and a similar concept can be applied in our case.

Note that in our case: i) since the circuit constraints $\{f_1, \ldots, f_s\}$ are modeled as polynomials in $\mathbb{F}_2$, they contain only multi-linear monomial terms; ii) the output of a gate is uniquely computed, and it always appears as a "single variable term" in the polynomials; iii) the circuit is acyclic. Let x_i be the output variable of any gate H_i in the circuit, and let $x_{p1}, \ldots, x_{pj}$ denote variables that are the inputs to the gate H_i. If we can represent the polynomials f_i such that $x_i >$ every monomial in the variables $x_{p1}, \ldots, x_{pj}$, then all $(f_i, f_j), i \neq j$ have relatively prime leading monomials and $\{f_1, \ldots, f_s\}$ is a Gröbner basis.

Example 6.1: Consider a 2-bit multiplier over $\mathbb{F}_{2^2}$ given in Fig. 1. Variables a_0, a_1, b_0, b_1 are primary inputs, z_0, z_1 are primary outputs, and s_0, s_1, s_2, s_3, r_0 are intermediate variables.

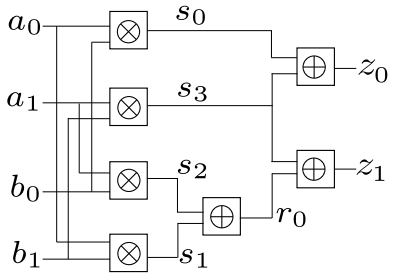

Fig. 1: A 2-bit Multiplier over $\mathbb{F}(2^2)$. The gate $\otimes$ corresponds to AND-gate, i.e. bit-level multiplication modulo 2. The gate $\oplus$ corresponds to XOR-gate, i.e. addition modulo 2.

We perform a "reverse topological traversal" of the circuit. Starting from the primary outputs, traverse the circuit to the primary inputs, and order the gates according to the their (reverse) topological levels. The primary outputs z_0, z_1 are both at level-0, variables r_0, s_0, s_3 are at level-1, s_1, s_2 are at level-2, and the primary inputs a_0, a_1, b_0, b_1 are at level-3. We order the variables $z_0 > z_1 > r_0 > s_0 > s_3 > s_1 > s_2 > a_0 > a_1 > b_0 > b_1$. Using this variable order, we impose a *lex* term order on the monomials. Then the polynomials extracted from the circuit all have relatively prime leading terms, as shown below:

$$s_0 + a_0 \cdot b_0,\ lm = s_0; \quad s_1 + a_0 \cdot b_1,\ lm = s_1$$
$$s_2 + a_1 \cdot b_0,\ lm = s_2; \quad s_3 + a_1 \cdot b_1,\ lm = s_3$$
$$r_0 + s_1 + s_2,\ lm = r_0; \quad z_0 + s_0 + s_3,\ lm = z_0$$
$$z_1 + r_0 + s_3,\ lm = z_1$$

In our overall problem formulation, we also have variables $A, B, S \in \mathbb{F}_q$. They can also be accommodated in this term order by imposing $Z > A > B > z_0 > z_1 > r_0 > s_0 > s_3 > s_1 > s_2 > a_0 > a_1 > b_0 > b_1$.

Thus, using the result of Proposition 2 [8], the set of polynomials $G_1 = \{f_1, \ldots, f_s\}$ is a Gröbner basis for J. Note that $G_0 = \{x_1^q - x_1, \ldots, x_d^q - x_d\}$ is a Gröbner basis for J_0. However, we have to compute a Gröbner basis of $J + J_0 = \langle f_1, \ldots, f_s,\ x_1^q - x_1, \ldots, x_d^q - x_d \rangle$. Not all polynomials pairs in $G_1 \cup G_0$ have relatively prime leading monomials.

Consider an arbitrary polynomial $f_i \in G_1$. Using our term order, we have $f_i = x_i + \text{tail}(f_i)$; i.e. the leading monomial of f_i is a single variable term x_i. Clearly, the pairs $(x_i + \text{tail}(f_i),\ x_i^q - x_i),\ f_i \in G_1,\ x_i^q - x_i \in G_0$ do not have relatively prime leading monomials. In fact, the pairs $(x_i + \text{tail}(f_i), x_i^q - x_i)$ are the only ones to be considered for Gröbner basis computation, as all other pairs have relatively prime leading terms. This motivated us to investigate further the question: "what is the result of the reduction $\text{Spoly}(x_i + \text{tail}(f_i), x_i^q - x_i) \xrightarrow{G_1,G_0}_+ r$?" We state and prove the following:

Theorem 6.1: Let $R = \mathbb{F}_q[x_1,\ldots,x_d]$ on which we have a monomial order $\leq$. Let I be a subset of $\{1,\ldots,d\}$. For all $i \in I$, let $f_i = x_i + P_i$ (where $P_i = \text{tail}(f_i)$) such that all indeterminates x_j that appear in P_i satisfy $x_i > x_j$. Then the set $G_1 \cup G_0 = \{f_i : i \in I\} \cup \{x_1^q - x_1,\ldots,x_d^q - x_d\}$ is a Gröbner basis.

Proof: According to Buchberger's Theorem (Theorem 1.7.4 in [22]), we need to show that for all $f,g \in G_1 \cup G_0$, $Spoly(f,g) \xrightarrow{G_1,G_0}_+ 0$. Lemma 6.1 shows that if f,g have relatively prime leading terms, then $Spoly(f,g) \xrightarrow{G_1,G_0}_+ 0$. So the only case where Lemma 6.1 does not apply is when $f = x_i + P_i$ and $g = x_i^q - x_i$. Then $Spoly(f,g) = x_i^{q-1} f - g = P_i x_i^{q-1} + x_i$. In what follows, it is important to note that the indeterminates appearing in P_i are all less than x_i.

First of all, $P_i x_i^{q-1} + x_i - P_i x_i^{q-2}(x_i + P_i) = P_i^2 x_i^{q-2} + x_i$, which shows that $P_i x_i^{q-1} + x_i \xrightarrow{x_i+P_i} P_i^2 x_i^{q-2} + x_i$.

Next, $P_i^2 x_i^{q-2} + x_i - P_i^2 x_i^{q-3}(x_i + P_i) = P_i^3 x_i^{q-3} + x_i$. Continuing in this fashion, we get $P_i^{q-1} x_i + x_i - P_i^{q-1}(x_i + P_i) = x_i + P_i^q$, and finally $x_i + P_i^q - (x_i + P_i) = P_i^q - P_i$. Hence,

$$P_i x_i^{q-1} + x_i \xrightarrow{x_i+P_i} P_i^2 x_i^{q-2} + x_i \xrightarrow{x_i+P_i} P_i^3 x_i^{q-3} + x_i \xrightarrow{x_i+P_i} \cdots$$

$$\cdots \xrightarrow{x_i+P_i} P_i^q + x_i \xrightarrow{x_i+P_i} P_i^q - P_i.$$

Over the Galois field $\mathbb{F}_q$, $P_i^q - P_i \in I(V(0)) = \langle x_1^q - x_1,\ldots,x_d^q - x_d\rangle$. By Lemma 6.1, $G_0 = \{x_1^q - x_1,\ldots,x_d^q - x_d\}$ is Gröbner basis. Therefore $P_i^q - P_i \xrightarrow{G_0}_+ 0$.

In conclusion, $\forall f,g \in G_1 \cup G_0,\ Spoly(f,g) \xrightarrow{G_1,G_0}_+ 0$ and hence $G_1 \cup G_0$ is a Gröbner basis. ■

We setup the verification problem in $\mathbb{F}_q[x_1,\ldots,x_d]$, on which we have the monomial order $\leq$ as specified above. We extract the set of polynomials $G_1 = \{f_1,\ldots,f_s\}$ from the circuit. We append the set $G_0 = \{x_1^q - x_1,\ldots,x_d^q - x_d\}$. Then the set $G_1 \cup G_2$ is a Gröbner basis of the ideal $J + J_0 = \langle f_1,\ldots,f_s, x_1^q - x_1,\ldots,x_d^q - x_d\rangle$. We take our specification polynomial f and compute $f \xrightarrow{G_1,G_0}_+ r$. If $r = 0$, then $f \in J + J_0$ and the circuit is correct; otherwise, if $r \neq 0$, then we have a bug in the design. Moreover, *if $r \neq 0$, then the monomial order ensures that r contains only the primary input variables*. To show this, assume that $r \neq 0$ and r contains either an intermediate or a primary output variable x_j. As there always exists a polynomial f_j in $G_1 \cup G_0$ with $lm(f_j) = x_j$, r can be further reduced by f_j. Continuing in this fashion, all the terms with non-primary-input (intermediate or primary output) variables can be eliminated. Finally, *in the presence of a bug, any assignment to the (primary-input) variables that makes $r \neq 0$, provides a counter-example for debugging* – and a SAT or SMT-solver can find such an assignment in no time. Our results therefore obviate the need to construct a Gröbner basis, and the verification can be performed only by reduction: $f \xrightarrow{G_1,G_0}_+ r$.

VII. Experimental Results

We have conducted experiments to verify custom implementations of Mastrovito and Montgomery Multipliers using SAT/SMT/BDDs and our proposed approach. We use the computer algebra tool SINGULAR [v. 3-1-3] [24] to conduct multivariate polynomial division for our approach. Our experiments are conducted on a desktop with 2.40GHz Intel Core(TM)2 Quad CPU and 8GB memory running 64-bit Linux. The initial designs are given in EQN format and then translated to different formats: CNF, SMTLIB, BLIF, Polynomials, as used by SAT, SMT, BDD and the SINGULAR tool, respectively.

Evaluation of SAT/SMT/BDD: Using the conventional equivalence checking approaches, we create a "miter" with the specification and the implementation, and use SAT/SMT/BDD methods to check if it is unsatisfiable. While the implementation is given as a circuit, the specification is given as a *word-level* polynomial $Z = A \cdot B \pmod{P(x)}$. For SAT and BDDs, we use a Mastrovito-style gate-level circuit as the golden model. For SMT experiments, the designs are modeled at bit-vector level using QF-BV theories, maintaining a BV-level abstraction whenever possible. Table I shows that none of BDDs, SAT or SMT solvers can verify the correctness of circuits beyond 16-bits.

TABLE I: Runtime for correct multiplier circuits over $\mathbb{F}_{2^k}$ for BDDs, SAT, SMT-solver based methods. TO = timeout of 10hrs. Time is givn in seconds.

	Word size of the operands k-bits		
Solver	8	12	16
MiniSAT	22.55	*TO*	*TO*
CryptoMiniSAT	7.17	16082.40	*TO*
PicoSAT	14.85	*TO*	*TO*
Yices	10.48	*TO*	*TO*
Beaver	6.31	*TO*	*TO*
CVC	*TO*	*TO*	*TO*
Z3	85.46	*TO*	*TO*
Boolector	5.03	*TO*	*TO*
SimplifyingSTP	14.66	*TO*	*TO*
ABC	242.78	*TO*	*TO*
BDD	0.10	14.14	1899.69

Conceptually, our approach requires to check whether the specification $f \in J + J_0$. Without deducing the result of Theorem 6.1, if we use Singular to compute a Gröbner basis for $G_1 \cup G_0$ using the proposed term order of [8], we can only verify the correctness upto 48-bit multipliers. Beyond that, the Gröbner basis engine runs into memory explosion. This is shown in Table II.

Evaluation of Our Approach: Our approach only requires the Gröbner basis reduction (division) for the verification test: $f \xrightarrow{G_1,G_0}_+ r$ and to check if $r = 0$? Results for verification of Mastrovito multipliers using only this Gröbner basis reduction (REDUCE command in SINGULAR) are shown in Table III.

TABLE II: Verification of correct circuits by computing $GB(J+J_0)$. *MO*=out of 8G memory. Time is givn in seconds.

Size	16	32	64	96	128	160	163
#variables	323	1155	4355	9603	16899	26243	27224
#polynomials	609	2241	8577	19009	33537	52161	54117
#terms	2415	9439	37311	83615	148351	231519	240261
Time	0.94	93.80	*MO*	*MO*	*MO*	*MO*	*MO*

With our approach, we can verify the correctness of upto 163-bit Mastrovito multipliers. We also experimented with bug-catching in incorrect designs; the bugs were introduced by arbitrarily swapping the wires (variables) x_i with x_j, for some $i \neq j$. In such cases, we obtained a non-zero r. We used a SAT-solver to find a SAT assignment to $r \neq 0$, and the counter-example was generated in no time. These run times are shown in Table III.

Results of the verification of Montgomery multipliers is shown in Table IV. Montgomery multipliers are significantly larger than Mastrovito multipliers. If we represent a polynomial for every gate in the design, then we create too many variables (d) in the system, exceeding SINGULAR'S capacity ($d \leq 32767$). For this reason, we have to partition the circuit, and construct the polynomials for each circuit partition – and we ensure that our term ordering constraint is not violated. With such efforts, we are able to verify Montgomery multipliers upto 128-bits, beyond which we still exceed SINGULAR'S capacity.

TABLE III: Runtime for verifying bug-free and buggy Mastrovito multipliers using our approach. Time is given in seconds.

Size	16	32	64	96	128	160	163
#variables	323	1155	4355	9603	16899	26243	27224
#polynomials	291	1091	4227	9411	16643	25923	26989
#terms	1793	7169	28673	64513	114689	179201	185984
Bug-free	0.04	1.41	112.13	758.82	3054	9361	16170
Bugs	0.04	1.43	114.86	788.65	3061	9384	16368

TABLE IV: Runtime for verifying bug-free and buggy Montgomery multipliers using our approach. Time is given in seconds.

Size	16	32	48	64	96	128
#variables	319	1194	2280	4395	6562	14122
#polynomials	287	1130	2184	4267	6370	13866
#terms	2262	10741	18199	40021	55512	134887
Bug-free	0.03	1.50	11.03	27.70	1802.75	10919.35
Bugs	0.03	1.52	11.10	28.18	1812.15	11047.10

VIII. CONCLUSIONS

This paper has presented a formal approach to model and verify multiplier circuits over Galois fields $\mathbb{F}_{2^k}$ using a computer-algebra based approach. We show how the verification test can be formulated as membership testing of the specification polynomial f in a (radical) ideal $\langle G_1, G_0 \rangle = \langle f_1, \ldots, f_s, x_1^q - x_1, \ldots, x_d^q - x_d \rangle$, where $G_1 = \{f_1, \ldots, f_s\}$ corresponds to the ideal generated by polynomials extracted from the circuit, and $G_0 = \{x_i^q - x_i\}$ corresponds to the ideal of vanishing polynomials of the field. By analyzing the circuit topology, we derive a monomial order that makes the set $G_1 \cup G_0$ itself a Gröbner basis of $J + J_0$. Subsequently, the verification can be formulated by simply carrying out the reduction $f \xrightarrow{G_1, G_0}_+ r$. Using our approach, we are able to verify the correctness of upto 163-bit multipliers over $\mathbb{F}_{2^{163}}$, whereas conventional techniques based on SAT/SMT/BDD solvers are infeasible.

REFERENCES

[1] E. D. Mastrovito, "VLSI Designs for Multiplication Over Finite Fields GF(2^m)," *Lecture Notes in Computer Science*, vol. 357, pp. 297–309, 1989.

[2] C. Koc and T. Acar, "Montgomery Multiplication in GF(2^k)," *Designs, Codes and Cryptography*, vol. 14, no. 1, pp. 57–69, Apr. 1998.

[3] H. Wu, "Montgomery Multiplier and Squarer for a Class of Finite Fields," *IEEE Transactions On Computers*, vol. 51, no. 5, May 2002.

[4] D. Babic and M. Musuvathi, "Modular Arithmetic Decision Procedure," Microsoft Research, Tech. Rep. TR-2005-114, 2005.

[5] E. Biham, Y. Carmeli, and A. Shamir, "Bug Attacks," in *Proc. Annual Conf. on Cryptology: Advances in Cryptology*, ser. CRYPTO 2008, 2008, pp. 221–240.

[6] S. Gao, "Counting Zeros over Finite Fields with Gröbner Bases," Master's thesis, Carnegie Mellon University, 2009.

[7] B. Buchberger, "Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal," Ph.D. dissertation, University of Innsbruck, 1965.

[8] O. Wienand, M. Wedler, D. Stoffel, W. Kunz, and G. Gruel, "An Algebraic Approach to Proving Data Correctness in Arithmetic Datapaths," in *Intl. Conf. on Computer-Aided Verification*, 2008.

[9] R. J. McEliece, *Finite Fields for Computer Scientists and Engineers*. Kluwer Academic Publishers, 1987.

[10] R. E. Bryant, "Graph Based Algorithms for Boolean Function Manipulation," *IEEE Trans. on Computers*, vol. C-35, pp. 677–691, August 1986.

[11] R. Drechsler and *et al*, "Efficient Representation and Manipulation of Switching Functions based on Ordered Kronecker Functional Decision Diagrams," in *DAC*, 1994, pp. 415–419.

[12] R. E. Bryant and Y.-A. Chen, "Verification of Arithmetic Functions with Binary Moment Diagrams," in *DAC*, 95.

[13] T. Rajaprabhu and *et al.*, "Modd for CF: A Compact Representation for Multiple Output Function," in *IEEE International High Level Design Validation and Test Works hop*, 2004.

[14] B. Alizadeh and M. Fujita, "Modular Datapath Optimization and Verification based on Modular-HED," *IEEE Trans. CAD*, pp. 1422–1435, Sept. 2010.

[15] N. Shekhar and *et al.*, "Equivalence Verification of Polynomial Datapaths with Fixed-Size Bit-Vectors using Finite Ring Algebra," in *ICCAD*, 2005.

[16] ——, "Equivalence Verification of Polynomial Datapaths using Ideal Membership Testing," *IEEE Trans. CAD*, pp. 1320–1330, July 2007.

[17] S. Morioka, Y. Katayama, and T. Yamane, "Towards efficient verification of arithmetic algorithms over Galois fields *GF*(2^m)," *Proc. Computer-Aided Verification*, vol. 2102, pp. 465–477, 2001.

[18] D. Mukhopadhyaya, G. Sengar, and D. Chowdhury, "Hierarchical Verification of Galois Field Circuits," *IEEE Trans. on CAD*, 2007.

[19] J. Lv, P. Kalla, and F. Enescu, "Verification of Composite Galois Field Multipliers over GF($(2^m)^n$) using Computer Algebra Techniques," in *IEEE High-Level Design Validation and Test Workshop (HLDVT 2011)*, 2011.

[20] ——, "Formal Verification of Galois Field Multipliers using Computer Algebra," in *25th IEEE International Conference on VLSI Design (VLSID 2012)*, 2012.

[21] E. Pavlenko and *et al.*, "STABLE: A new QBF-BV SMT Solver for hard Verification Problems combining Boolean Reasoning with Computer Algebra," in *Proc. Design Auto. & Test in Europe (DATE)*, 2011.

[22] W. W. Adams and P. Loustaunau, *An Introduction to Grobner Bases*. American Mathematical Society, 1994.

[23] B. Buchberger, "A criterion for detecting unnecessary reductions in the construction of a groebner bases," in *EUROSAM*, 1979.

[24] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, "SINGULAR 3-1-3 — A computer algebra system for polynomial computations," 2011, http://www.singular.uni-kl.de.