On the Rectifiability of Arithmetic Circuits using Craig Interpolants in Finite Fields

Utkarsh Gupta[1] Irina Ilioaea[2] Vikas Rao[1] Arpitha Srinath[1] Priyank Kalla[1] Florian Enescu[2]
[1]Electrical and Computer Engineering, University of Utah, Salt Lake City UT, USA
[2]Mathematics and Statistics, Georgia State University, Atlanta GA, USA

***Abstract*—When formal verification of arithmetic circuits identifies the presence of a bug in the design, the task of rectification needs to be performed to correct the function implemented by the circuit so that it matches the given specification. This paper addresses the problem of rectification of buggy finite field arithmetic circuits. The problems are formulated by means of a set of polynomials (ideals) and solutions are proposed using concepts from computational algebraic geometry. Single-fix rectification is addressed – i.e. the case where any (set of) bugs can be rectified at a single net (gate output). We determine if single-fix rectification is possible at a particular location, formulated as the Weak Nullstellensatz test. Subsequently, we introduce the concept of Craig interpolants in polynomial algebra over finite fields and show that the rectification function can be computed using algebraic interpolants. Experimental results demonstrate the superiority of our approach against SAT-based approaches.**

I. Introduction

Past few years have seen extensive investigations into formal verification of arithmetic circuits. Circuits that implement polynomial computations over large bit-vector operands are hard to verify using methods such as SAT/SMT-solvers, decision diagrams, etc. Recent techniques have investigated the use of polynomial algebra and algebraic geometry techniques for their verification. These include verification of integer arithmetic circuits [1] [2] [3] and also finite field circuits [4] [5]. While these are successful in proving correctness or detecting the presence of bugs, the problem of debugging and correction of arithmetic circuits has only just begun to be addressed [6], [7].

In this paper, we address the problem of rectification of buggy finite field arithmetic circuits. Our problem setup is as follows:

- A specification model (*Spec*) is given either as a polynomial description f_{spec} over a finite field, or as a golden model of a finite field arithmetic circuit. The finite field considered is the field of 2^k elements (denoted by $\mathbb{F}_{2^k}$), where k corresponds to the operand-width (bit-vector word length). An implementation circuit C is also given.
- Equivalence checking is performed between the *Spec* and the circuit C, and the presence of a bug is detected. No restrictions on the number, type, or locations of the bugs are assumed.
- We assume that error-diagnosis has been performed, and a subset X of the nets of the circuit is identified as *potential rectification locations.*

Given the *Spec*, the buggy implementation circuit C, the set X of potential rectifiable locations, our objective is to determine whether or not the buggy circuit can be rectified at *one particular net (location)* $x_i \in X$. This is called **single-fix rectification** in literature [8]. If a single-fix rectification does exist at net x_i in the buggy circuit, then our subsequent objective is to derive a polynomial function $U(X_{PI})$ in terms of the set of primary input variables X_{PI}. This polynomial can be translated (synthesized) into a logic subcircuit such that $x_i = U(X_{PI})$ acts as the rectification function for the buggy circuit C so that C matches the specification.

Another important contribution of our work is that we show that the rectification function $U(X_{PI})$ can be determined based on the concept of Craig interpolants [9] in algebraic geometry. While Craig interpolation is a well-studied concept in propositional and first-order logic theories, we recently showed in [10] that polynomial algebra in finite fields also admits Craig interpolation, and described algorithms to compute interpolants. Based on our results of [10], we show how to compute a rectification function using Craig interpolation in finite fields.

Our techniques and algorithms are based on symbolic computer algebra and algebraic geometry – particularly on the concepts of the Weak Nullstellensatz and Gröbner bases [11]. We show how to apply our techniques to rectify finite field arithmetic circuits, where conventional SAT-solver based rectification approaches are infeasible.

Review of the Previous Work: Automated diagnosis and rectification of digital circuits has been addressed in [12], [13]. The paper [14] presents algorithms for synthesizing Engineering Change Order (ECO) patches. The use of interpolation for ECO has been presented in [8], [15], [16]. The single-fix rectification function approach in [15], [16] has been extended in [8] to generate multiple partial-fix functions. As these approaches are SAT based, they work well for random logic circuits but are not efficient for arithmetic circuits. In contrast to these works, our work presents a word-level formulation for single-fix rectification. Computer algebra has been utilized for circuit debugging and rectification in [17], [6], [7]. These approaches rely heavily on the structure of the circuit for coefficient calculation. If the arithmetic circuit contains redundancies, the approach may not identify the buggy gate due to ambiguity in coefficient values. On the

This research is funded in part by the US National Science Foundation grants CCF-1619370 and CCF-1320385.

978-1-5386-4756-1/18/$31.00 ©2018 IEEE

other hand, our approach although more efficient for finite field arithmetic circuits, is applicable to any circuit in general.

The paper is organized as follows. The following section describes preliminary concepts in computer algebra and describe an equivalence checking framework using the Weak Nullstellensatz over finite fields. Section III presents our main theorem on identifying a rectification location and obtaining the correction function using Craig interpolants in finite fields. Section IV presents our experimental results and section V concludes the paper.

II. Preliminaries

Let $\mathbb{F}_q$ denote the finite field of q elements where $q = 2^k$ and k is the operand width. Let $R = \mathbb{F}_q[x_1,\dots,x_n]$ be the polynomial ring in n variables $x_1,\dots,x_n$, with coefficients from $\mathbb{F}_q$. A monomial is a power product of variables $x_1^{e_1}\cdot x_2^{e_2}\cdots x_n^{e_n}$, where $e_i \in \mathbb{Z}_{\geq 0}, i \in \{1,\dots,n\}$. A *polynomial* $f \in R$ is written as a finite sum of terms $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$, where $c_1,\dots,c_t$ are coefficients and $X_1,\dots,X_t$ are monomials. A monomial order $>$ (or a term order) is imposed on the ring so that the monomials of all polynomials $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$ are ordered w.r.t. $>$, such that $X_1 > X_2 > \cdots > X_t$, where $lm(f) = X_1$ is called the *leading monomial* of f. In this work, we employ lexicographic (lex) term orders (see Definition 1.4.3 in [11]).

We model the given circuit C by a set of multivariate polynomials $f_1,\dots,f_s \in \mathbb{F}_{2^k}[x_1,\dots,x_n]$; here $x_1,\dots,x_n$ denote the nets (signals) of the circuit. Every Boolean logic gate of C is represented by a polynomial in $\mathbb{F}_2$, as $\mathbb{F}_2 \subset \mathbb{F}_{2^k}$. This is shown below. Note that in $\mathbb{F}_{2^k}$, $-1 = +1$.

$$\begin{aligned} z &= \neg a \;\rightarrow\; z + a + 1 \pmod 2 \\ z &= a \wedge b \;\rightarrow\; z + a\cdot b \pmod 2 \\ z &= a \vee b \;\rightarrow\; z + a + b + a\cdot b \pmod 2 \\ z &= a \oplus b \;\rightarrow\; z + a + b \pmod 2 \end{aligned} \tag{1}$$

Given a set of polynomials $F = \{f_1,\dots,f_s\}$ in R, the *ideal* $J \subseteq R$ generated by them is:

$$J = \langle f_1,\dots,f_s\rangle = \{\sum_{i=1}^{s} h_i\cdot f_i : \; h_i \in R\}.$$

The polynomials $f_1,\dots,f_s$ form the *generators* of J.

Let $\boldsymbol{a} = (a_1,\dots,a_n) \in \mathbb{F}_q^n$ be a point in the affine space, and f a polynomial in R. If $f(\boldsymbol{a}) = 0$, we say that f *vanishes* on $\boldsymbol{a}$. In verification, we have to analyze the *set of all common zeros* of the polynomials of F that lie within the field $\mathbb{F}_q$. In other words, we need to analyze solutions to the system of polynomial equations $f_1 = f_2 = \cdots = f_s = 0$. This zero set is called the *variety*. It depends not just on the given set of polynomials but rather on the ideal generated by them. We denote it by $V(J) = V(f_1,\dots,f_s)$, where:

$$V(J) = V(f_1,\dots,f_s) = \{\boldsymbol{a} \in \mathbb{F}_q^n : \forall f \in J, f(\boldsymbol{a}) = 0\}.$$

We denote the complement of a variety, $\mathbb{F}_q^n \setminus V(J)$, by $\overline{V(J)}$.

Algebraic Miter for Equivalence Checking: Given f_{spec} as the specification polynomial, we need to construct an *algebraic miter* between f_{spec} and C. For equivalence checking, we need to prove that the miter is infeasible. Fig. 1 depicts how a word-level algebraic miter is setup. Suppose that $A = \{a_0,\dots,a_{k-1}\}$ and $Z = \{z_0 \dots, z_{k-1}\}$ denote the k-bit primary inputs and outputs of the finite field circuit. Then $A = \sum_{i=0}^{k-1} a_i\alpha^i, Z = \sum_{i=0}^{k-1} z_i\alpha^i$ correspond to the word-level polynomials for the inputs and outputs of the circuit. Here α is the primitive element of $\mathbb{F}_{2^k}$. Let Z_S be the word-level output for f_{spec}, which computes some polynomial function $\mathcal{F}(A)$ of A, so that $f_{spec} : Z_S + \mathcal{F}(A)$. The word-level outputs Z, Z_S are mitered to check if for all inputs, $Z \neq Z_S$ is infeasible.

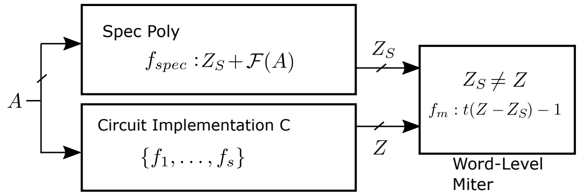

Fig. 1: Word-Level Miter

In finite fields, the disequality $Z \neq Z_S$ can be modeled as a single polynomial f_m, called the miter polynomial, where $f_m = t\cdot(Z - Z_S) - 1$, and t is introduced as a free variable. If $Z = Z_S$, $Z - Z_S = 0$. So $f_m : t\cdot 0 + 1 = 0$ has no solutions (miter is infeasible). Whereas if for some input A, $Z \neq Z_S$, then $Z - Z_S \neq 0$. Let $t^{-1} = (Z - Z_S) \neq 0$. Then $f_m : t\cdot t^{-1} - 1 = 0$ has a solution as t, t^{-1} become multiplicative inverses of each other. Thus the miter becomes feasible.

In this way, equivalence checking using the algebraic model is solved as follows: Construct an ideal $J = \langle f_{spec}, f_1,\dots,f_s, f_m\rangle$, as described above. Then determine if the variety $V(J) = \emptyset$? If $V(J) = \emptyset$, the miter is infeasible, and C implements f_{spec}. If $V(J) \neq \emptyset$, the miter is feasible, and there exists a bug in the design.

The Weak Nullstellensatz: To ascertain whether $V(J) = \emptyset$, we employ the Weak Nullstellensatz over $\mathbb{F}_q$, for which we use the following notations. Given two ideals $J_1 = \langle f_1,\dots,f_s\rangle, J_2 = \langle h_1,\dots,h_r\rangle$, the sum $J_1 + J_2 = \langle f_1,\dots,f_s,h_1\dots,h_r\rangle$, and $V(J_1 + J_2) = V(J_1) \cap V(J_2)$. Moreover, if $J_1 \subseteq J_2$ then $V(J_1) \supseteq V(J_2)$.

For all elements $\alpha \in \mathbb{F}_q, \alpha^q = \alpha$. Therefore, the polynomial $x^q - x$ vanishes everywhere in $\mathbb{F}_q$, and is called the vanishing polynomial of the field. Let $J_0 = \langle x_1^q - x_1,\dots,x_n^q - x_n\rangle$ be the ideal of all vanishing polynomials in R.

Theorem II.1 (*The Weak Nullstellensatz over finite fields (from Theorem 3.3 in [18])*)**.** *For a finite field $\mathbb{F}_q$ and the ring $R = \mathbb{F}_q[x_1,\dots,x_n]$, let $J = \langle f_1,\dots,f_s\rangle \subseteq R$, and let $J_0 = \langle x_1^q - x_1,\dots,x_n^q - x_n\rangle$ be the ideal of vanishing polynomials. Then $V(J) = \emptyset \iff 1 \in J + J_0$.*

To determine whether $V(J) = \emptyset$, we need to test whether or not the unit element 1 is a member of the ideal $J + J_0$. For this *ideal membership test, we need to compute a Gröbner basis of $J + J_0$.*

Gröbner Basis of Ideals: An ideal may have many different sets of generators: $J = \langle f_1,\dots,f_s\rangle = \cdots = \langle g_1,\dots,g_t\rangle$. Given a non-zero ideal J, a *Gröbner basis* (GB) for J is a finite set of polynomials $G = \{g_1,\dots,g_t\}$ satisfying $\langle\{lm(f) \mid f \in J\}\rangle = \langle lm(g_1),\dots,lm(g_t)\rangle$. Then $J = \langle G\rangle$ holds and so $G = GB(J)$ forms a basis for J. A GB G possesses important properties that allow to solve many polynomial computation and decision problems. The famous Buchberger's algorithm (see Alg. 1.7.1 in [11]) takes as input the set of polynomials $F = \{f_1, \dots, f_s\}$ and computes the GB $G = \{g_1, \dots, g_t\}$. A GB can be *reduced* to eliminate redundant polynomials from the basis. A reduced GB is a canonical representation of the ideal. When $1 \in J$, then $G = reduced\_GB(J) = \{1\}$.

Thus, for equivalence check, we compute a reduced GB $G = GB(J + J_0)$, and see if $G = \{1\}$. If so, $V(J) = \emptyset$ and the miter is infeasible. If $G \neq \{1\}$, then there exists a bug in the design.

Craig interpolation: The Weak Nullstellensatz is the polynomial analog of SAT/UNSAT checking. For UNSAT problems, the formal logic and verification communities have explored the notion of abstraction of functions by means of Craig interpolants, which has been applied to circuit rectification [8]. In propositional logic, the concept is defined as follows:

Definition II.1. Let (A,B) be a pair of CNF formulae (sets of clauses) such that $A \wedge B$ is unsatisfiable. Then there exists a formula I such that: (i) $A \implies I$; (ii) $I \wedge B$ is unsatisfiable; and (iii) I refers only to the common variables of A and B, i.e. $Var(I) \subseteq Var(A) \cap Var(B)$. The formula I is called the **interpolant** of (A,B).

Given the pair (A,B) and their refutation proof, a procedure called the *interpolation system* constructs the interpolant in linear time and space in the size of the proof. In our work [10], we have proposed the notion (theory and algorithms) of Craig interpolants in polynomial algebra over finite fields, based on the results of Nullstellensatz. These are presented and utilized in this paper for rectification of arithmetic circuits.

Elimination Ideals: We employ one more concept, that of elimination ideals.

Definition II.2. Given an ideal $J \subset \mathbb{F}_q[x_1, \dots, x_n]$, the l-th elimination ideal J_l is an ideal in R defined as $J_l = J \cap \mathbb{F}_q[x_{l+1}, \dots, x_n]$.

Theorem II.2 (*Elimination Theorem (from Theorem 2.3.4 [11])*)**.** Given an ideal $J \subset R$ and its GB G *w.r.t.* the lexicographical (lex) order on the variables where $x_1 > x_2 > \cdots > x_n$, then for every $0 \leq l \leq n$ we denote by G_l the GB of l-th elimination ideal of J and compute it as:

$$G_l = G \cap \mathbb{F}_q[x_{l+1}, \dots, x_n].$$

G_l is called the l-th elimination ideal as it eliminates the first l variables from J.

III. Theory

This section presents our main theorem on checking whether a buggy circuit is single fix rectifiable, and a procedure for computing a correction function using the theory and algorithms on Craig interpolants in finite fields [10].

After the verification of a circuit against the specification detects the presence of a bug in the design, we are provided with a list of potential gate-output nets x_i's. The circuit may or may not be rectified at a particular x_i. First we ascertain that the circuit can indeed be rectified at some x_i and then apply a correction function $U(X_{PI})$ as $x_i = U(X_{PI})$.

A. Single Fix Rectification

In this subsection, we formally set up the problem of single fix circuit rectification. Using the Weak Nullstellensatz (Theorem II.1), we formulate the test for rectifiability at a gate output x_i in the circuit. The following proposition will be used later in this subsection.

Proposition III.1. Given two ideals J_1 and J_2 over some finite field such that $V(J_1) \cap V(J_2) = \emptyset$, there exists a polynomial U which satisfies $V(J_1) \subseteq V(U) \subseteq \overline{V(J_2)}$.

Proof. Over finite fields, $V(J_1)$ and $V(J_2)$ are finite sets of points. There exists a set of points which contains $V(J_1)$ and does not intersect with $V(J_2)$. As every set of points in finite fields is a variety, let this variety be denoted by $V(J_I)$, where J_I is the corresponding ideal. Then $V(J_1) \subseteq V(J_I) \subseteq \overline{V(J_2)}$. In addition, we can construct a polynomial U whose roots are exactly the points in $V(J_I)$ by means of the Lagrange's interpolation formula. □

Now we present the theorem to check the circuit's rectifiability at some gate output. Let us assume that a potential rectifiable gate output is x_i (*i.e.* i^{th} gate) and a possible function in primary inputs that can be implemented is $x_i = U(X_{PI})$ so that the i^{th} gate is represented by a polynomial $f_i : x_i + U(X_{PI})$. The ideal constructed from the polynomials for the gates $f_1, \dots, f_s$ of the circuit, the specification polynomial f_{spec}, and the miter polynomial f_m, is denoted by J:

$$J = \langle f_{spec}, f_1, \dots, f_i : x_i + U(X_{PI}), \dots, f_s, f_m \rangle.$$

The following theorem checks whether the circuit is indeed rectifiable at gate with output net x_i.

Theorem III.1. Construct two ideals:

- $J_L = \langle f_{spec}, f_1, \dots, f_i : x_i + 1, \dots, f_s, f_m \rangle$ where $f_i : x_i + U(X_{PI})$ in J is replaced with $f_i : x_i + 1$.
- $J_H = \langle f_{spec}, f_1, \dots, f_i : x_i, \dots, f_s, f_m \rangle$ where $f_i : x_i + U(X_{PI})$ in J is replaced with $f_i : x_i$.

Compute $E_L = (J_L + J_0) \cap \mathbb{F}_{2^k}[X_{PI}]$ and $E_H = (J_H + J_0) \cap \mathbb{F}_{2^k}[X_{PI}]$ to be the respective elimination ideals, where all the non-primary input variables have been eliminated. Then the circuit can be rectified with a logic function at net x_i with the polynomial function $f_i : x_i + U(X_{PI})$ to implement the specification iff $1 \in E_L + E_H$.

Proof. We will first prove the *if* case of the theorem. Assume $1 \in E_L + E_H$, or equivalently $V_{X_{PI}}(E_L) \cap V_{X_{PI}}(E_H) = \emptyset$. Using Proposition III.1, we can find a polynomial $U(X_{PI})$ such that,

$$V_{X_{PI}}(E_L) \subseteq V_{X_{PI}}(U(X_{PI})) \subseteq \overline{V_{X_{PI}}(E_H)} \qquad (2)$$

where the universal set for computing $\overline{V_{X_{PI}}(E_H)}$ is $\mathbb{F}_{2^k}^{X_{PI}}$. Let us assume that a point $\boldsymbol{p}$ exists in $V(J)$. Point $\boldsymbol{p}$ is an assignment to every variable in J such that all the generators of J are satisfied. We denote by $\boldsymbol{a}$, the projection of $\boldsymbol{p}$ on the primary inputs (the primary input assignments under $\boldsymbol{p}$). There are only two possibilities for $U(X_{PI})$,

1) $U(\boldsymbol{a}) = 1$, or in other words $\boldsymbol{a} \notin V_{X_{PI}}(U(X_{PI}))$. It also implies that the value of x_i under $\boldsymbol{p}$ must be 1 because $x_i + U(X_{PI})$ needs to be satisfied. Since the generator f_i

of J_L also forces x_i to be 1 and all its other generators are exactly the same as that of J, $\boldsymbol{p}$ is also a point in $V(J_L)$. Moreover, E_L is the elimination ideal of J_L, and therefore, $\boldsymbol{a} \in V_{X_{PI}}(E_L)$. But this a contradiction to our assumption that $V_{X_{PI}}(E_L) \subseteq V_{X_{PI}}(U(X_{PI}))$ and such a point $\boldsymbol{a}$ (and $\boldsymbol{p}$) does not exist.

2) $U(\boldsymbol{a}) = 0$, or in other words $\boldsymbol{a} \in V_{X_{PI}}(U(X_{PI}))$. Using similar argument as the previous case, we can show that $\boldsymbol{a} \in V_{X_{PI}}(E_H)$. This is again a contradiction to our assumption $V_{X_{PI}}(U(X_{PI})) \subseteq \overline{V_{X_{PI}}(E_H)}$.

In conclusion, there exists no point in $V(J)$ (or the miter is infeasible) when $U(X_{PI})$ satisfies Eqn. 2, and therefore, circuit can be rectified at x_i.

Now we will prove the *only if* direction of the proof. We show that if $1 \notin E_L + E_H$, then there exists no polynomial $U(X_{PI})$ that can rectify the circuit. If $1 \notin E_L + E_H$, then E_L and E_H have a common zero. Let $\boldsymbol{a}$ be a point in $V_{X_{PI}}(E_L)$ and $V_{X_{PI}}(E_H)$. This point can be extended to some points $\boldsymbol{p'}$ and $\boldsymbol{p''}$ in $V(J_L)$ and $V(J_H)$, respectively. Notice that in point $\boldsymbol{p'}$ the value of x_i will be 1, and in $\boldsymbol{p''}$ x_i will be 0. Any polynomial $U(X_{PI})$ will either evaluate to 0 or 1 for the assignment $\boldsymbol{a}$ to the primary inputs. If it evaluates to 1, then we can say that $\boldsymbol{p'}$ is in $V(J)$ as f_i in J forces $x_i = 1$ and all other generators of J and J_L are same. This implies that $f_m(\boldsymbol{p'}) = 0$ (f_m: miter polynomial is feasible) and this choice of $U(X_{PI})$ will not rectify the circuit. If $U(X_{PI})$ evaluates to 0, then $\boldsymbol{p''}$ is a point in $V(J)$.

Therefore, no choice of $U(X_{PI})$ can rectify the circuit if $1 \notin E_L + E_H$. □

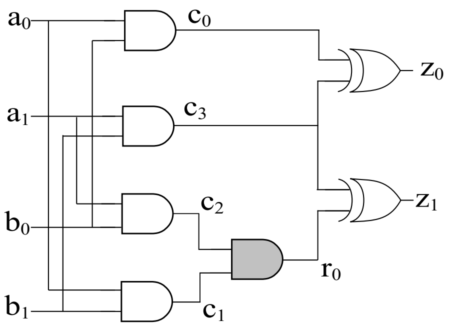

Fig. 2: A buggy 2-bit modulo multiplier circuit

Example III.1. *Consider the buggy modulo multiplier circuit in Fig. 2 where the gate output r_0 should have been the output of an XOR gate and the AND gate has been incorrectly implemented. We want to apply Thm. III.1 at r_0. The polynomials for the gates of the correct circuit implementation are,*

$$f_1 : c_0 + a_0 \cdot b_0; \quad f_2 : c_1 + a_0 \cdot b_1; \quad f_3 : c_2 + a_1 \cdot b_0;$$
$$f_4 : c_3 + a_1 \cdot b_1; \quad f_5 : r_0 + c_1 + c_2; \quad f_6 : z_0 + c_0 + c_3;$$
$$f_7 : z_1 + r_0 + c_3;$$

The problem is modeled over $\mathbb{F}_4$ and α is the primitive element of $\mathbb{F}_4$. The word-level polynomials are $f_8 : Z + z_0 + z_1\alpha$, $f_9 : A + a_0 + a_1\alpha$, and $f_{10} : B + b_0 + b_1\alpha$. The specification polynomial is $f_{spec} : Z_s + AB$. We create a miter polynomial against this specification as $f_m : t(Z - Z_s) - 1$.

The ideals J_L and J_H are as follows,

$$J_L = \langle f_{spec}, f_1, \dots, f_4, r_0 + 1, f_6, \dots, f_{10}, f_m \rangle$$
$$J_H = \langle f_{spec}, f_1, \dots, f_4, r_0, f_6, \dots, f_{10}, f_m \rangle$$

and the corresponding ideals E_L and E_H are as follows,

$$E_L = \langle a_0b_1 + a_1b_0, a_1b_0b_1 + a_1b_0, a_0a_1b_0 + a_1b_0 \rangle$$
$$E_H = \langle b_0b_1 + b_0 + b_1 + 1, a_1b_1 + a_1 + b_1 + 1, a_0b_1 + a_1b_0 + 1,$$
$$a_0b_0 + a_0 + b_0 + 1, a_0a_1 + a_0 + a_1 + 1 \rangle$$

If we compute a Gröbner basis of $E_L + E_H$, it results in $\{1\}$. Therefore, we can rectify this circuit at r_0.

B. Craig Interpolants in Finite Fields

If x_i is a feasible location for rectification, then the corresponding E_L and E_H satisfy $1 \in E_L + E_H$. We are now in a position to introduce the notion of Craig interpolants in finite fields which will help us in obtaining $U(X_{PI})$ from an *"ideal-interpolant"* J_I defined below.

Definition III.1 (*Interpolants in finite fields*)**.** Given two ideals $J_A \subset \mathbb{F}_q[A,C]$ and $J_B \subset \mathbb{F}_q[B,C]$ where A, B, C denote the three disjoint sets of variables such that $V_{A,B,C}(J_A) \cap V_{A,B,C}(J_B) = \emptyset$. Then there exists an ideal J_I satisfying the following properties:

1) $V_{A,B,C}(J_I) \supseteq V_{A,B,C}(J_A)$
2) $V_{A,B,C}(J_I) \cap V_{A,B,C}(J_B) = \emptyset$
3) Generators of J_I contain only the C-variables; or $J_I \subseteq \mathbb{F}_q[C]$.

We call $V_{A,B,C}(J_I)$ the **interpolant** in finite fields of the pair $(V_{A,B,C}(J_A), V_{A,B,C}(J_B))$, and the corresponding ideal J_I the **ideal-interpolant**.

Example III.2. *Consider the ring $R = \mathbb{F}_2[a,b,c,d,e]$, partition the variables as $A = \{a\}, B = \{e\}, C = \{b,c,d\}$. Let ideals*

$$J_A = \langle ab, bd, bc + c, cd, bd + b + d + 1 \rangle + J_{0,A,C}$$
$$J_B = \langle b, d, ec + e + c + 1, ec \rangle + J_{0,B,C}$$

where $J_{0,A,C}$ and $J_{0,B,C}$ are the corresponding ideals of vanishing polynomials. Then, we have

$$V_{A,B,C}(J_A) = \mathbb{F}_q^B \times V_{A,C}(J_A) = (abcde) :$$
$$\{01000, 00010, 01100, 10010, 01001, 00011, 01101, 10011\}$$
$$V_{A,B,C}(J_B) = \mathbb{F}_q^A \times V_{B,C}(J_B) = (abcde) :$$
$$\{00001, 00100, 10001, 10100\}$$

The ideals J_A, J_B have no common zeros as $V_{A,B,C}(J_A) \cap V_{A,B,C}(J_B) = \emptyset$. The pair (J_A, J_B) admits a total of 8 interpolants:

1) $V(J_S) = (bcd) : \{001, 100, 110\}$ $\quad J_S = \langle cd, b + d + 1 \rangle$
2) $V_C(J_1) = (bcd) : \{001, 100, 110, 101\}$
$J_1 = \langle cd, bd + b + d + 1, bc + cd + c \rangle$
3) $V_C(J_2) = (bcd) : \{001, 100, 110, 011\}$
$J_2 = \langle b + d + 1 \rangle$
4) $V_C(J_3) = (bcd) : \{001, 100, 110, 111\}$
$J_3 = \langle b + cd + d + 1 \rangle$
5) $V_C(J_4) = (bcd) : \{001, 100, 110, 011, 111\}$
$J_4 = \langle bd + b + d + 1, bc + b + cd + c + d + 1 \rangle$
6) $V_C(J_5) = (bcd) : \{001, 100, 110, 101, 111\}$
$J_5 = \langle bc + c, bd + b + d + 1 \rangle$

7) $V_C(J_6) = (bcd) : \{001,100,110,101,011\}$
$J_6 = \langle bd+b+d+1, bc+cd+c\rangle$
8) $V_C(J_L) = (bcd) : \{001,011,100,101,110,111\}$
$J_L = \langle bd+b+d+1\rangle$.

It is easy to check that all $V(J_I)$ satisfy the 3 conditions of Def. III.1. Note also that $V(J_S)$ is the smallest interpolant, contained in every other interpolant. Likewise, $V(J_L)$ contains all other interpolants and it is the largest. The other containment relationships are shown in the corresponding interpolant lattice in Fig. 3; $V_C(J_1) \subset V_C(J_5), V_C(J_1) \subset V_C(J_6)$, etc.

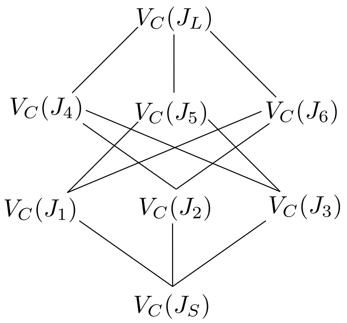

Fig. 3: Interpolant lattice for Example III.2

Theorem III.2. (from Theorem IV.1 in [10]) An ideal-interpolant J_I, and correspondingly the interpolant $V_{A,B,C}(J_I)$, as given in Def. III.1, always exists.

Another result from [10] (Theorem IV.2) that we make use of here is that the smallest interpolant can be computed as $J_I = J_A \cap \mathbb{F}_q[C]$.

Back to our formulation of single fix rectification, we have $1 \in E_L + E_H$ or $V(E_L) \cap V(E_H) = \emptyset$. E_L and E_H are elimination ideals containing only X_{PI} variables. As a result, the set of variables A, B, and C are primary inputs. Moreover, we want to compute an ideal J_I in X_{PI} such that $V_{X_{PI}}(E_L) \subseteq V_{X_{PI}}(J_I)$ and $V_{X_{PI}}(J_I) \cap V_{X_{PI}}(E_H) = \emptyset$. The *smallest ideal-interpolant* $J_I = E_L \cap \mathbb{F}_2[X_{PI}] = E_L$ itself. Therefore, we use E_L to compute the correction function $U(X_{PI})$.

C. Obtaining $U(X_{PI})$ from E_L

In finite fields, given an ideal J, it always possible to find a polynomial U such that $V(U) = V(J)$. The reason is that every ideal in a finite field has a finite variety and a polynomial with those points as its roots can always be constructed. Let the generators of J be denoted by $g_1, \dots, g_t$. We can compute U as,

$$U = (1+g_1)(1+g_2)\cdots(1+g_t)+1 \qquad (3)$$

It is easy to assert that $V(U) = V(J_I)$. Using the Eqn. 3, we can write a recursive procedure as presented in Algorithm 1 to compute U. In addition, at every recursive step we also reduce the intermediate sum by J_0 (line 6) to avoid large degree terms. In our setting, $U = U(X_{PI})$ and $J = E_L$, and therefore, we can find a correction function $x_i + U(X_{PI})$ which can be used to rectify the circuit.

Using this procedure for Example III.1, we have $U(X_{PI})$ as $a_0b_1 + a_1b_0$ and the correction function as $r_0 + a_0b_1 + a_1b_0$ which can be synthesized as $r_0 = (a_0 \wedge b_1) \oplus (a_1 \wedge b_0)$ (replacing the modulo 2 product and sum with Boolean AND and XOR, respectively).

Algorithm 1 Compute U from J_I such that $V(U) = V(J_I)$

```
1: procedure compute_U($J_I, J_0$) //$J_I = \{f_1, \dots, f_s\}$
2:     if $size(J_I) = 1$ then
3:         return $(1 + J_I[1])$
4:     $subsetJ = \{J_I[1], J[2], \dots, J_I[size(J_I)-1]\}$
5:     $poly\ S_1 = compute\_U(subsetJ, J_0)$
6:     Perform $S_1 \cdot J_I[size(J_I)] \xrightarrow{J_0}_+ S_2$
7:     return $S_1 + S_2$
```

IV. Experimental Results

We have performed experiments on finite field arithmetic circuits (used in cryptography) where the implementation is different from the specification due to exactly one gate. This is to ensure that a single fix rectification is feasible. We implement the procedure described in the previous section (Thm. III.1 and Algo. 1) using the SINGULAR symbolic algebra computation system [ver. 4-1-0][19]. The experiments were conducted on a desktop computer with a 3.5GHz Intel Core$^{\text{TM}}$ i7-4770K Quad-core CPU, 16 GB RAM, running 64-bit Linux OS.

We have performed experiments with three different types of finite field benchmarks. First two of them are Mastrovito and Montgomery multiplier circuits used for modular multiplication. Mastrovito multipliers compute $Z = A \times B \pmod{P(x)}$ where $P(x)$ is a given primitive polynomial for the datapath size k. Montgomery multipliers are preferred for exponentiation operations (often required in cryptosystems) over Mastrovito multipliers. The last set of benchmarks are circuits implementing point addition over elliptic curves used for encryption, decryption and authentication in elliptic curve cryptography.

TABLE I: Mastrovito multiplier rectification against Montgomery multiplier specification. Time in seconds; Time-out = 5400s; k: Operand width

k	# of Gates		SAT	Thm. III.1	Algo. 1	Mem
	Mas	Mont				
4	48	96	0.09	0.03	0.001	8.16 MB
8	292	319	158.34	0.41	0.006	20.36 MB
9	237	396	4,507	0.47	0.001	18.95 MB
10	285	480	TO	0.84	0.001	28.2 MB
16	1,836	1,152	TO	73.63	0.024	0.32 GB
32	5,482	4,352	TO	3621	0.043	2.4 GB

First we present the results for the case where the Thm. III.1 is applied at a gate location such that the circuit is completely rectifiable. Table I compares the execution time for SAT based approach [8] and our approach (Theorem III.1) for checking whether a buggy Mastrovito multiplier can be rectified at a certain location in the circuit against a Montgomery multiplier specification. We have implemented the SAT procedure using the *abc* tool [20]. We execute the command *inter* on the ON set and OFF set as described in [8]. The SAT based procedure is unable to perform the necessary unsatisfiability check for circuits beyond of 9 bit operand words. Using our approach, the polynomial $U(X_{PI})$ needed for rectification is computed from E_L and the time is reported in Table I in the Algo. 1 column. The last column in the table is the memory usage of our approach.

We can also perform the rectification when a polynomial specification is given instead of a specification circuit. Table II shows the result of checking whether the incorrect Mastrovito implementation can be rectified at a particular location against the word level specification polynomial $Z_S = AB$.

TABLE II: Mastrovito multiplier rectification against polynomial specification $Z_S = AB$. Time in seconds; Time-out = 5400s; k: Operand width

k	# of Gates	Thm. III.1	Algo. 1	Mem
4	48	0.01	0.001	7.24 MB
8	292	0.08	0.006	14.95 MB
16	1,836	4.83	0.038	0.2 GB
32	5,482	100.52	0.015	1.42 GB
64	21,813	4,989	0.117	12.25 GB

Point addition operation can be represented as polynomials because modern approaches represent the points in projective coordinate systems, *e.g.*, the López-Dahab (LD) projective coordinate [21]. Each of these polynomials can be implemented as a circuit. Table III shows the result for one of these blocks. For all the experiments, the most computationally expensive part is the computation of ideals E_L and E_H.

TABLE III: Point Addition circuit rectification against polynomial specification $D = B^2 \cdot (C + aZ_1^2)$. Time in seconds; Time-out = 5400s; k: Operand width

Field Size (k)	# of Gates	Thm. III.1	Algo. 1	Mem
8	243	0.05	0.022	9.73 MB
16	1,277	3.48	0.019	88.78 MB
32	3,918	86.75	0.028	0.47 GB
64	1,5305	4,923	0.053	7.13 GB

We also performed experiments where we apply Thm. III.1 at a gate output which cannot rectify the circuit. We used Montgomery circuit as the specification and Mastrovito as the implementation as we did for the experiments in Table I. For 4 and 8 bits size cases, the execution time was comparable for Thm. III.1 and SAT based approach and was ~ 0.1 seconds. When we tried the 16 bit case, the SAT based approach was able to complete in 0.11 seconds. On the other hand, Thm. III.1 formulation resulted in a memory explosion and consumed $\sim$ 30 GB of memory in 5-6 minutes. This is due to the fact when $1 \notin E_L + E_H$, then $GB(E_L + E_H)$ is not equal to $\{1\}$ and the Gröbner basis algorithm produces a very large generating set. To improve our approach we are working on term ordering heuristics so that our approach can perform efficiently in both cases. We also want to employ better data structures as SINGULAR's data structure is not very memory efficient and also has an upper limit on the number of variables (32,768) that can be accommodated in the system.

V. Conclusion

This paper considers the single-fix rectification of circuits after the verification has detected a bug in the design. A number of possible gate outputs are provided whose functionality can be changed so that circuit corresponds to the specification. We want to select one such (single) gate output so that by applying a correction function there, the circuit is rectified. We present a theorem that answers definitively whether a single fix rectification is feasible at a particular gate output. We also briefly describe the notion and definition of Craig interpolants in finite fields which is used to obtain a correction function. Experiments performed over finite field arithmetic circuits shows the efficiency of our approach and also points out the regions for improvements.

References

[1] D. Ritirc, A. Biere, and M. Kauers, "Column-Wise Verification of Multipliers Using Computer Algebra," in *Formal Methods in Computer-Aided Design (FMCAD)*, 2017, pp. 23–30.

[2] M. Ciesielski, C. Yu, W. Brown, D. Liu, and A. Rossi, "Verification of Gate-level Arithmetic Circuits by Function Extraction," in *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[3] A. Sayed-Ahmed, D. Große, U. Kühne, M. Soeken, and R. Drechsler, "Formal Verification of Integer Multipliers by Combining Gröbner Basis with Logic Reduction," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 1048–1053.

[4] J. Lv, P. Kalla, and F. Enescu, "Efficient Gröbner Basis Reductions for Formal Verification of Galois Field Arithmetic Circuits," in *IEEE Trans. on CAD*, vol. 32, no. 9, 2013, pp. 1409–1420.

[5] A. Lvov, L. Lastras-Montano, B. Trager, V. Paruthi, R. Shadowen, and A. El-Zein, "Verification of Galois field based circuits by formal reasoning based on computational algebraic geometry," *Formal Methods in System Design*, vol. 45, no. 2, pp. 189–212, Oct 2014.

[6] F. Farahmandi and P. Mishra, "Automated Debugging of Arithmetic Circuits Using Incremental Gröbner Basis Reduction," in *IEEE International Conference on Computer Design (ICCD)*, 2017.

[7] ——, "Automated Test Generation for Debugging Arithmetic Circuits," in *Des. Auto. Test in Eur. Conf. (DATE)*, 2016.

[8] K. F. Tang, C. A. Wu, P. K. Huang, and C. Y. Huang, "Interpolation-Based Incremental ECO Synthesis for Multi-Error Logic Rectification," in *Proc. Design Automation Conf. (DAC)*, 2011, pp. 146–151.

[9] W. Craig, "Linear reasoning: A new form of the Herbrand-Gentzen theorem," *Journal of Symbolic Logic*, vol. 22, no. 3, pp. 250–268, 1957.

[10] U. Gupta, I. Ilioaea, P. Kalla, F. Enescu, V. Rao, and A. Srinath, "Craig Interpolants in Finite Fields using Algebraic Geometry: Theory and Applications," in *Intl. Workshop on Logic and Synthesis (IWLS)*, June 2018.

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

[12] J. C. Madre, O. Coudert, and J. P. Billon, "Automating the Diagnosis and the Rectification of Design Errors with PRIAM," in *Proc. ICCAD*, 1989, pp. 30–33.

[13] H. T. Liaw, J. H. Tsaih, and C. S. Lin, "Efficient Automatic Diagnosis of Digital Circuits," in *Proc. ICCAD*, 1990, pp. 464–467.

[14] C. C. Lin, K. C. Chen, S. C. Chang, and M. Marek-Sadowska, "Logic Synthesis for Engineering Change," in *Proc. Design Automation Conf. (DAC)*, 1995, pp. 647–652.

[15] B. H. Wu, C. J. Yang, C. Y. Huang, and J. H. R. Jiang, "A Robust Functional ECO Engine by SAT Proof Minimization and Interpolation Techniques," in *Intl. Conf. on Comp. Aided Des.*, 2010, pp. 729–734.

[16] A. C. Ling, S. D. Brown, S. Safarpour, and J. Zhu, "Toward Automated ECOs in FPGAs," *IEEE Trans. CAD*, vol. 30, no. 1, pp. 18–30, 2011.

[17] S. Ghandali, C. Yu, D. Liu, W. Brown, and M. Ciesielski, "Logic Debugging of Arithmetic Circuits," in *IEEE Computer Society Annual Symposium on VLSI*, 2015.

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

[19] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, "Singular 4-1-0 — A computer algebra system for polynomial computations," http://www.singular.uni-kl.de, 2016.

[20] R. Brayton and A. Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool," in *Comp. Aid. Verif.*, vol. 6174, 2010, pp. 24–40.

[21] J. López and R. Dahab, "Improved Algorithms for Elliptic Curve Arithmetic in GF(2^n)," in *Proceedings of the Selected Areas in Cryptography*. London, UK, UK: Springer-Verlag, 1999, pp. 201–212.