Computing Rectification Functions for Integer Arithmetic Circuits using Algebraic Geometry

Abstract: *This paper addresses the problem of rectification of buggy integer arithmetic circuits using concepts from symbolic computer algebra and algebraic geometry. When formal verification identifies the presence of a bug in the implemented circuit, logic rectification needs to be performed at local nets so that the circuit implementation matches the given specification. Assuming that the circuit admits rectification at a single net, we show how to compute a rectification function at that net using an algebraic model. The approach is based on Gröbner basis techniques, which are used as a quantification procedure to compute a rectification function. We demonstrate the application to rectification of integer multiplier circuits, while presenting new theory and techniques that associate the algebraic computations to the presence of local don't care sets. This gives us a complete approach to compute Boolean functions that can rectify the buggy circuit. Experiments are performed over integer multipliers that demonstrate the efficacy of our approach.*

I. Introduction

Past few years have witnessed significant advances in automatic formal verification of arithmetic circuits. Given a polynomial specification (*Spec* f) and an implementation circuit C, techniques have been researched that formally prove that C implements f, or ascertain the presence of a bug in the design. These techniques use an *algebraic (i.e. polynomial algebra) model* as a core computational engine for verification, and have demonstrated successful verification for integer arithmetic circuits [1] [2] [3] [4] [5], integer modulo arithmetic circuits [6], and also for finite field circuits [7] [8] – all with large operand sizes.

When formal verification of arithmetic circuits detects the presence of a bug in the design, then it is required to perform post-verification debugging and **rectification** of the buggy implementation so that the rectified circuit matches the given specification. In such a setting, it is desirable to compute rectification functions at internal nets, without resynthesizing/redesigning the entire circuit – i.e. the problem of *partial synthesis*. The partial synthesis problem is more involved than that of verification, as the former is a quantification problem whereas the latter is a decision problem. Rectification requires to first identify a set of nets where the circuit can be rectified, and then to subsequently compute corresponding rectification functions at those nets. While this problem of has been addressed for random-logic circuits [9] [10] [11] [12], as well as recently for finite field circuits [13] [14], the problem has not been satisfactorily addressed for rectification of integer arithmetic circuits. This paper addresses the problem of *computing and synthesizing rectification functions for buggy integer arithmetic circuits*.

Problem Statement: We are given an integer arithmetic circuit C and a polynomial specification f. Formal verification is performed to verify C against f (say, using the techniques of [2]). We assume that verification determines that C is buggy, i.e. C does not implement f. No assumptions are made about the type, nature or the number of bugs in C. Assume further that post-verification debugging has been performed and *it has been determined that C can be rectified at a single-net x_i*. In other words, we assume that C admits *single-fix rectification* at net x_i. Our *objective* is to compute and synthesize a Boolean function $U(X_{PI})$ in terms of primary input variables of the circuit (X_{PI}) such that modifying the circuit at $x_i = U(X_{PI})$ rectifies C to match f.

Related Previous Work: Automated diagnosis and rectification of digital circuits was addressed in [15], and the technique was extended in [9] for synthesizing Engineering Change Order (ECO) patches - an analogous problem to rectification. Once rectification is deemed feasible, the problem of finding the rectification function has been considered as partial synthesis of an unknown component U. The approach [11] formulates the computation of U as a quantified Boolean formula (QBF), and solves it using incremental and iterative SAT solving. Similarly, the authors in [12] present a QBF formulation for answering whether a partial implementation (with block-boxes U) can be extended to a complete design that models a given specification. As Craig interpolation is an alternative to QBF solving, it has been presented in [10] for multi-fix rectification for ECO synthesis.

The concept of Craig interpolation was extended to polynomials in finite fields [13] [16] and application to rectification of finite field arithmetic circuits was demonstrated. As Craig interpolation in finite fields is a computationally hard problem, another rectification approach was presented for finite field circuits in [14] which uses the (extended) Gröbner basis algorithm as a quantification procedure to compute the rectification function. Rectification has been attempted for integer arithmetic circuits [17] [18]. Unfortunately, the technique of [17] was demonstrated to be incomplete in [14], and [18] is limited by the fault model which only addresses the gate misplacement fault.

Approach and Contributions: We model the rectification problem using concepts from algebraic geometry and use symbolic computer algebra algorithms to compute the rectification function $U(X_{PI})$. Using the model presented in [2], we describe the gates of C with a set of polynomials $F = \{f_1, \dots, f_s\}$ with integer coefficients. Subsequently, the problem of computing $U(X_{PI})$ is modeled as an extended ideal membership testing problem, which is solved using the Gröbner basis algorithm [19].

Our work draws inspirations from [1] [2] and [14]. In [1][2], the authors showed that the verification of integer multipliers could be formulated and solved over polynomial rings with coefficients from the field of fractions $\mathbb{Q}$, as opposed to solving over integers $\mathbb{Z}$. As most algebraic geometry results are valid over fields (and $\mathbb{Z} \subset \mathbb{Q}$ is not a field), their approach leveraged theory as well as efficient computational techniques over fields to solve the verification problem. Also, [14] shows how to compute single-fix rectification functions $x_i = U(X_{PI})$ over finite field circuits. *We use the polynomial model of [1] [2] to perform Gröbner basis computations over $\mathbb{Q}$, and use the ideal membership approach presented in Section VI in [14] to compute the rectification function.*

Our approach assumes that it has been determined that the net x_i in C admits single-fix rectification. This implies the *existence of a Boolean function* $U : \{0,1\}^{|X_{PI}|} \rightarrow \{0,1\}$, such that $x_i = U(X_{PI})$ rectifies the circuit. However, since we use a polynomial model over $\mathbb{Q}$, application of the techniques of [14] computes U as a polynomial function over $\mathbb{Q}$, i.e. it may evaluate to non-Boolean values in $\mathbb{Q}$. Therefore, one of the main contribution of our approach is that it investigates the nature of the rectification polynomial, its relationship to don't care conditions, and shows how to synthesize a rectification polynomial U for integer arithmetic circuits such that it maps as $\{0,1\}^{|X_{PI}|} \rightarrow \{0,1\}$. Experiments demonstrate the application of our approach to integer (array) multiplier circuits.

Finally, the problem of ascertaining single-fix rectifiability at net x_i is not described in this paper (mostly due to lack of space). It can be solved in a manner similar to Theorem V.1 in [14]. Similarly, multi-fix rectification is also beyond the scope of this paper.

Paper Organization: The paper is organized as follows: The following section covers preliminary concepts. Sec. III covers the verification setup, and Sec. IV describes the rectification theory and approach. Experiments are described in Sec. V, and Sec. VI concludes the paper.

II. Preliminaries: Notation and Background

Let $\mathbb{Q}$ be the set of rational numbers that forms a field. Let $R = \mathbb{Q}[x_1, \dots, x_n]$ be the polynomial ring in variables $x_1, \dots, x_n$ with coefficients in $\mathbb{Q}$. A polynomial $f \in R$ is written as a finite sum of terms $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$. Here $c_1, \dots, c_t$ are coefficients and $X_1, \dots, X_t$ are monomials, i.e. power products of the type $x_1^{e_1} \cdot x_2^{e_2} \cdots x_n^{e_n}$, $e_j \in \mathbb{Z}_{\geq 0}$. To systematically manipulate the polynomials, a monomial order $>$ (also called a term order) is imposed on the polynomial ring. Subject to $>$, $X_1 > X_2 > \cdots > X_t$, and $lt(f) = c_1X_1$, $lm(f) = X_1$, $lc(f) = c_1$, are the *leading term*, *leading monomial* and *leading coefficient* of f, respectively. Also, for a polynomial f, $tail(f) = f - lt(f)$. In this work, we are mostly concerned with *lexicographic* (lex) term orders.

Polynomial Reduction via division: Let f, g be polynomials. If $lm(f)$ is divisible by $lm(g)$, then we say that f *is reducible to* r modulo g, denoted $f \xrightarrow{g} r$, where $r = f - \frac{lt(f)}{lt(g)} \cdot g$. Similarly, f can be *reduced w.r.t. a set of polynomials* $F = \{f_1, \dots, f_s\}$ to obtain a remainder r. This reduction is denoted as $f \xrightarrow{F}_+ r$, where the remainder r is said to be *reduced* – i.e. no term in r is divisible by the leading term of any polynomial f_j in F. Algorithm 1.5.1 from [19] depicts a procedure for this reduction. Along with the remainder r, the algorithm also returns the set of quotients $\{u_1, \dots, u_s\}$ of division of f by $\{f_1, \dots, f_s\}$, respectively, such that $f = u_1 \cdot f_1 + \cdots + u_s \cdot f_s + r$.

Polynomial Ideals, Varieties and Gröbner Bases:

Definition II.1. Given a ring $R = \mathbb{Q}[x_1, \dots, x_n]$ and a set of polynomials $F = \{f_1, \dots, f_s\}$ from R, the *ideal* generated by F is $J = \langle F \rangle \subseteq R$:

$$J = \langle f_1, \dots, f_s \rangle = \{h_1 \cdot f_1 + \cdots + h_s \cdot f_s : h_1, \dots, h_s \in R\}. \quad (1)$$

The polynomials $f_1, \dots, f_s$ form the basis of ideal J.

Let $\boldsymbol{a} = (a_1, \dots, a_n) \in \mathbb{Q}^n$ be a point in the affine space, and f a polynomial in R. If $f(\boldsymbol{a}) = 0$, we say that f *vanishes* on $\boldsymbol{a}$. We have to analyze the *set of all common zeros* of the polynomials of F that lie within the field $\mathbb{Q}$. This zero set is called the *variety*. We denote it by $V_{\mathbb{Q}}(J)$, where: $V_{\mathbb{Q}}(J) = V_{\mathbb{Q}}(f_1, \dots, f_s) = \{\boldsymbol{a} \in \mathbb{Q}^n : \forall f \in J, f(\boldsymbol{a}) = 0\}$.

An ideal may have many different sets of generators, i.e. it is possible to have $J = \langle f_1, \dots, f_s \rangle = \langle g_1, \dots, g_t \rangle = \cdots = \langle h_1, \dots, h_r \rangle$, such that $V(f_1, \dots, f_s) = V(g_1, \dots, g_t) = \cdots = V(h_1 \dots, h_r)$. A Gröbner basis (GB) of an ideal is one such generating set $G = \{g_1, \dots, g_t\}$, that is a canonical representation of the ideal. Gröbner Basis is used to solve many polynomial decision problems.

The Gröbner basis for an ideal J can be computed using the Buchberger's algorithm [20], given in Algorithm 1.7.1 in [19]. The algorithm takes as input a set of polynomials $\{f_1, \dots, f_s\}$ and computes its GB $G = \{g_1, g_2, \cdots, g_t\}$.

Buchberger's algorithm can be easily extended to output not just the Gröbner basis $G = \{g_1, \dots, g_t\}$ but also a $t \times s$ matrix M with polynomial entries such that:

$$\begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_t \end{bmatrix} = M \cdot \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_s \end{bmatrix} \quad (2)$$

An important property of Gröbner bases is that as a decision procedure, they allow for membership testing of a polynomial in an ideal.

Proposition II.1. (*Ideal Membership Testing)* Let $G = GB(J) = \{g_1, \dots, g_t\}$ be the Gröbner basis of ideal J, and f be any polynomial. Then $f \in J \iff f \xrightarrow{G}_+ 0$.

Therefore, if $f \in J$, f can be written as a linear combination (with polynomial coefficients) of the elements of the Gröbner basis:

$$f = u_1g_1 + u_2g_2 + \cdots + u_tg_t, \quad (3)$$

where u_i's correspond to the quotients of division $f \xrightarrow{g_1, \dots, g_t}_+ 0$. Subsequently, Eqns. (3) and (2) can be combined to give f as combination of the original polynomials $f_1, \dots, f_s$:

$$f = v_1f_1 + \cdots + v_sf_s. \quad (4)$$

Given two ideals $J_1 = \langle f_1, \dots, f_s \rangle, J_2 = \langle h_1, \dots, h_r \rangle$, we use the notation and concept of the sum of ideals $J_1 + J_2 = \langle f_1, \dots, f_s,\ h_1 \dots, h_r \rangle$, and $V(J_1 + J_2) = V(J_1) \cap V(J_2)$.

Definition II.2. (Ideal of vanishing polynomials) Let $F_0 = \{x_l^2 - x_l : l = 1, \dots, n\}$ be a set of polynomials over the ring $\mathbb{Q}[x_1, \dots, x_n]$. Let J_0 be the ideal generated by F_0, $J_0 = \langle F_0 \rangle$. The variety of ideal J_0 is $V(J_0) = \{0,1\}^n$.

Consider the polynomial $f = x_l^2 - x_l \in \mathbb{Q}[x]$. The solution to this polynomial f over $\overline{\mathbb{Q}}$, and over $\mathbb{Q}$, is $\{0,1\}$. Therefore, $V(J_0) = V_{\mathbb{Q}}(J_0) = \{0,1\}^n$. Thus we can include ideal J_0 in our computations to restrict the solutions of circuit's polynomial constraints to over Boolean values. In other words, $V_{\mathbb{Q}}(J + J_0) \subset \{0,1\}^n$.

III. The Polynomial Model and Gröbner Basis Reduction

Let $R = \mathbb{Q}[x_1, \dots, x_n]$, where the set of variables $X = \{x_1, \dots, x_n\}$ denotes the nets in the circuit. Let f be a polynomial in R that acts as specification for a given (multiplier) circuit. Consider the (multiplier) circuit C with two k-bit vectors as inputs. Let input vectors be $a_0, \dots, a_{k-1}$ and $b_0, \dots, b_{k-1}$, i.e. $X_{PI} = \{a_0, \dots, a_{k-1}, b_0, \dots, b_{k-1}\}$. The output of the circuit C is a $2k$-bit vector $s_0, \dots, s_{2k-1}$.

The **multiplier specification** is:$f = \sum_{i=0}^{2k-1} 2^i s_i - \sum_{i=0}^{k-1} 2^i a_i \cdot \sum_{i=0}^{k-1} 2^i b_i$, where $a_i, b_i, s_i \in \{0,1\} \subset \mathbb{Q}$.

Implementation: Given a gate-level circuit netlist, we map the gate-level Boolean operators (AND, OR, NOT, XOR) to polynomials over $\mathbb{Q}$ using the following mapping:

$$\begin{aligned} u = \neg v &\implies u - 1 + v = 0 \\ u = v \wedge w &\implies u - vw = 0 \\ u = v \vee w &\implies u - v - w + vw = 0 \\ u = v \oplus w &\implies u - v - w + 2vw = 0 \end{aligned} \tag{5}$$

The polynomials $F = \{f_1, \dots, f_s\}$ model all the gates in the circuit and generate an ideal $J = \langle F \rangle$. Ideal $J_0 = \langle x_1^2 - x_1, \dots, x_n^2 - x_n \rangle$ is also constructed.

The verification problem can be solved as checking whether or not f is a member of the ideal $J + J_0$, for which we can compute $G = GB(J + J_0)$, and check of $f \xrightarrow{G}_+ 0$, i.e. if dividing f by a Gröbner basis of $J + J_0$ gives zero remainder. As the computation of a GB is potentially explosive, it has been shown in [7] [1] that a term order $>$ can be derived that renders the set of polynomials of $F \cup F_0$ themselves a Gröbner basis of their ideal $J + J_0$.

Fact III.1. Let C be any arbitrary combinational circuit. Let $\{x_1, \dots, x_n\}$ denote the set of all variables (signals) in the circuit, i.e. the primary input, intermediate and primary output variables. Perform a **reverse topological traversal** of the circuit and order the variables such that $x_i > x_j$ if x_i appears earlier in the reverse topological order. Impose a lexicographic term order to represent the Boolean expression for each gate as a polynomial f_i; then $f_i = x_i + tail(f_i)$. Such a term order is called **Reverse Topological Term Order (RTTO)**. Then the set of all polynomials $F = \{f_1, \dots, f_s\}$, together with the set of all vanishing polynomials only in the primary input variables $F_0 = \{x_l^2 - x_l : x_l \in X_{PI}\}$ forms a Gröbner basis: i.e. if $J = \langle F \rangle, J_0 = \langle F_0 \rangle$, then $F \cup F_0 = GB(J + J_0)$.

As a result, verification can be performed as $f \xrightarrow{F \cup F_0}_+ r$ under RTTO $>$ and checking iff $r = 0$. Also note that due to RTTO, the leading term of each polynomial $f_i, 1 \le i \le s$ is the net x_i, corresponding to the output of the f_i gate.

Example III.1. A bug is introduced in the 2-bit integer array multiplier circuit (redundant gates added to the circuit), shown in Fig. 1. The bug is introduced at a net s. The bug is introduced by replacing the AND gate at s by an OR gate. The polynomials of the circuit, written in RTTO $>$ are $F = \{f_1, f_2, f_{r1}, f_{r2}, f_3, f_4, f_5, f_{r3}, f_{r4}, f_{r5}, f_{b6}, f_7, f_8\}$:

$$\begin{aligned} &f_1 = z_0 - a_0 \cdot b_0 && f_2 = m - a_1 \cdot b_0 \\ &f_{r1} = g - 1 + b_0 && f_{r2} = h - a_0 - b_0 + a_0 \cdot b_0 \\ &f_3 = n - a_0 \cdot b_1 && f_4 = o - a_1 \cdot b_1 \\ &f_5 = z_1 - m - n + 2 \cdot m \cdot n && f_{r3} = i - h \cdot b_0 \\ &f_{r4} = j - i - g + 2 \cdot i \cdot g && f_{r5} = k - j \cdot m \\ &\mathbf{f_{b6} = s - k - n + k \cdot n} && f_7 = z_2 - s - o + 2 \cdot s \cdot o \\ &f_8 = -z_3 + s \cdot o && \end{aligned}$$

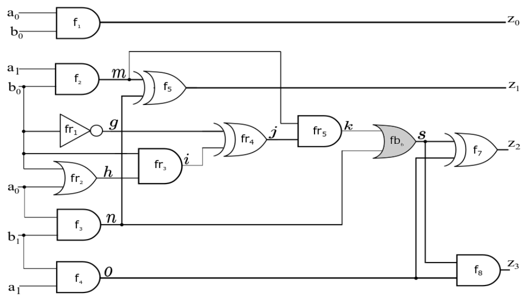

Fig. 1: 2-bit integer multiplier with a bug at s.

The *Spec* is: $f = \sum_{i=0}^{2n-1} 2^i z_i - \sum_{i=0}^{n-1} 2^i a_i \cdot \sum_{i=0}^{n-1} 2^i b_i$. Verification is done by performing Gröbner Basis reduction, $f \xrightarrow{J+J_0}_+ r$. We find $r = -4 \cdot a_0 \cdot a_1 \cdot b_0 \cdot b_1 + 4 \cdot a_0 \cdot b_1$. Since $r \neq 0$, verification detects the presence of a bug in the design. We will now attempt to rectify the circuit at an internal net (single-fix rectification is possible at net s itself!).

IV. Computing a rectification function

Assume that post-verification debugging has been performed and a net x_i in the circuit C has been identified that admits single-fix rectification. Therefore, our objective is to now compute a function $x_i = U(X_{PI})$ such that $U : \{0,1\}^{|X_{PI}|} \to \{0,1\}$. We will first show how to compute U as a polynomial function, and then how to synthesize it as a circuit. Polynomially, we will represent the hitherto unknown component U as $f_i = x_i - U(X_{PI})$, where x_i is the leading term of the corresponding polynomial.

Let $R = \mathbb{Q}[x_1, \dots, x_n]$ and let $F = \{f_1, \dots, f_i : x_i - U(X_{PI}), \dots, f_s\}$ the system of polynomials modeling the correct circuit and $U(X_{PI})$ is the unknown rectification polynomial to be computed. The polynomials are ordered according to RTTO, s.t. $lt(f_1) > lt(f_2) > \cdots > lt(f_i) : x_i > lt(f_{i+1}) > \cdots > lt(f_s)$. The desired rectification function $U(X_{PI})$ computed must satisfy the condition $f \in \langle f_1, \dots, f_i : x_i - U(X_{PI}), \dots, f_s, x_l^2 - x_l \rangle$, where $x_l \in X_{PI}$. This polynomial function $U(X_{PI})$ can be computed using the combination of extended Gröbner Basis and ideal membership testing. As the polynomial $f_i = x_i - U$ corrects the circuit, we can write:

$$f \in \langle f_1, \dots, f_{i-1}, \boldsymbol{f_i : x_i - U}, f_{i+1}, \dots, f_s, x_l^2 - x_l \rangle \tag{6}$$

The ideal membership relation of f can be written as:

$$\begin{aligned} f = &h_1 f_1 + h_2 f_2 + \cdots + \boldsymbol{h_i f_i} + \cdots + h_s f_s \\ &+ \sum_{x_l \in X_{PI}} H_l \cdot (x_l^2 - x_l) \end{aligned} \tag{7}$$

where $\{h_1, \dots, h_s, H_l\} \subset R$ are arbitrary polynomials. Substituting $f_i = x_i - U$:

$$\begin{aligned} f =& h_1 f_1 + h_2 f_2 + \cdots + \boldsymbol{h_i(x_i - U)} + \cdots + h_s f_s \\ &+ \sum_{x_l \in X_{PI}} H_l \cdot (x_l^2 - x_l) \end{aligned} \tag{8}$$

$$\begin{aligned} f =& h_1 f_1 + h_2 f_2 + \cdots + \boldsymbol{h_i x_i - h_i U} + \cdots + h_s f_s \\ &+ \sum_{x_l \in X_{PI}} H_l \cdot (x_l^2 - x_l) \end{aligned} \tag{9}$$

$$\begin{aligned} & f - h_1 f_1 - h_2 f_2 - \cdots - h_i x_i \\ &= -h_i U + \cdots + h_s f_s + \sum_{x_l \in X_{PI}} H_l \cdot (x_l^2 - x_l) \end{aligned} \tag{10}$$

Notice that on the L.H.S. of Eqn. (10), the polynomials $f, f_1, \dots, f_{i-1}$ and the monomial x_i are known polynomial expressions. Therefore, f can be divided by $f_1, \dots, f_{i-1}$ and x_i to obtain the respective quotients of the division $h_1, \dots, h_i$ and remainder r. The polynomial f can be written as

$$f = h_1 f_1 + h_2 f_2 + \cdots + h_i x_i + r \tag{11}$$

The remainder r can be computed as:

$$r = f - h_1 f_1 - h_2 f_2 - \cdots - h_i x_i \tag{12}$$

After h_i is computed (as quotient of division by x_i), the R.H.S. of Eqn. (10) consists of $h_i, f_{i+1}, \dots, f_s$ and polynomials $x_l^2 - x_l$ as known expressions. This implies:

$$f - h_1 f_1 - \cdots - h_i x_i \in \langle -h_i, f_{i+1}, \dots, f_s, x_l^2 - x_l \rangle \tag{13}$$

$$r \in \langle -h_i, f_{i+1}, \dots, f_s, x_l^2 - x_l \rangle \tag{14}$$

Note that RTTO renders the polynomial $f_{i+1}, \dots, f_s$ a Gröbner Basis. However, $\{-h_i, f_{i+1}, \dots, f_s\}$ is not. We compute its a Gröbner Basis $G = \{g_1, \dots, g_t\}$. We apply the extended Gröbner Basis theory explained in Eqn. (4) to G to find the ideal membership relation of r.

$$r = -h_i' h_i + h_{i+1}' f_{i+1} + \cdots + h_s' f_s + \sum_{x_l \in X_{PI}} H_l' (x_l^2 - x_l) \tag{15}$$

Then h_i' is a polynomial that satisfies the ideal membership relation in Eqn. (6). According to [14], the polynomial $h_i' = U$ rectifies the circuit such that the circuit matches the specification.

However, in our model, h_i' is computed over the field of rational numbers $\mathbb{Q}$. This polynomial may have fractional coefficients and it may also evaluate to any non-Boolean value in $\mathbb{Q}$ for some primary input vectors. We use the example of a 3-bit multiplier shown in Fig. 2 to explain this in more detail.

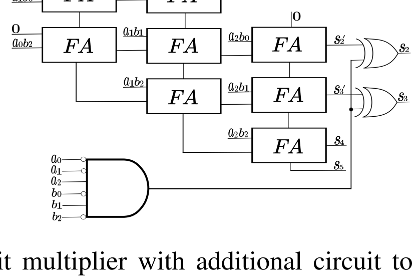

Fig. 2: 3-bit multiplier with additional circuit to introduce a bug

Example IV.1. The Fig. 2 shows a 3-bit array multiplier circuit. A bug in introduced in the circuit in the form of additional gates as shown. Gates are added to the circuit to modify the function of the circuit at two output bits for one input pattern $(a_0, a_1, a_2, b_0, b_1, b_2) = (0, 0, 1, 0, 1, 0)$. Single-fix rectification is feasible at a net n_{22} in the circuit. We apply Eqns. (6) to (15) and compute the rectification function h_i', as well as h_i, as shown in Eqn. (10).

$$\begin{aligned} h_i =& 8 \cdot a_0 \cdot a_1 \cdot a_2 \cdot b_0 + 16 \cdot a_0 \cdot a_1 \cdot a_2 \cdot b_1 - 12 \cdot a_0 \cdot a_1 \\ &- 8 \cdot a_0 \cdot a_2 \cdot b_0 - 16 \cdot a_0 \cdot a_2 \cdot b_1 + 12 \cdot a_0 - 8 \cdot a_1 \cdot a_2 \cdot b_0 \\ &- 16 \cdot a_1 \cdot a_2 \cdot b_1 + 12 \cdot a_1 + 8 \cdot a_2 \cdot b_0 + 16 \cdot a_2 \cdot b_1 - 12 \end{aligned}$$

$$\begin{aligned} h_i' =& -\frac{44}{3} \cdot a_0 \cdot a_1 \cdot a_2 \cdot b_0 \cdot b_1 \cdot b_2 + \frac{4}{3} \cdot a_0 \cdot a_1 \cdot a_2 \cdot b_0 \cdot b_1 \\ &+ 8 \cdot a_0 \cdot a_1 \cdot a_2 \cdot b_0 \cdot b_2 + \frac{8}{3} \cdot a_0 \cdot a_1 \cdot a_2 \cdot b_1 \cdot b_2 \\ &+ \frac{4}{3} \cdot a_0 \cdot a_1 \cdot b_0 \cdot b_1 \cdot b_2 - \frac{2}{3} \cdot a_0 \cdot a_1 \cdot b_0 \cdot b_1 \\ &+ \frac{4}{3} \cdot a_0 \cdot a_1 \cdot b_1 \cdot b_2 + \frac{8}{3} \cdot a_0 \cdot a_2 \cdot b_0 \cdot b_1 \cdot b_2 \\ &- 4 \cdot a_0 \cdot a_2 \cdot b_0 \cdot b_2 + \frac{28}{3} \cdot a_1 \cdot a_2 \cdot b_0 \cdot b_1 \cdot b_2 \\ &- \frac{14}{3} \cdot a_1 \cdot a_2 \cdot b_0 \cdot b_1 - \frac{8}{3} \cdot a_1 \cdot a_2 \cdot b_0 \cdot b_2 \\ &- \frac{20}{3} \cdot a_1 \cdot a_2 \cdot b_1 \cdot b_2 + \frac{8}{3} \cdot a_1 \cdot a_2 \cdot b_1 \\ &- \frac{2}{3} \cdot a_1 \cdot b_1 - \frac{4}{3} \cdot a_1 \cdot b_2 + 1 \end{aligned}$$

We evaluate the polynomial h_i and h_i' for all primary input patterns. The observations are recorded in the following table.

$a_0, a_1, a_2, b_0, b_1, b_2$	h_i	h_i'
0,0,0,0,0,0	-12	1
0,0,0,0,0,1	-12	1
0,0,0,0,1,0	-12	1
0,1,0,0,0,1	0	$-\frac{1}{3}$
0,1,0,0,1,0	0	$\frac{1}{3}$
0,1,0,0,1,1	0	-1

TABLE I: Evaluating h_i and h_i'

In order to conserve space, Table I shows the data for only some input patterns. The rectification polynomial h_i' has fractional coefficients. At points where $h_i = 0$, we see that h_i' can evaluate to any value in $\mathbb{Q}$. We found 48 such points where h_i evaluates to 0. At points where $h_i \neq 0$, we always observe that h_i' evaluates to a value in $\{0, 1\}$.

In order to understand the conditions under which the rectification polynomial h_i' computes to only Boolean values, we investigate at the variety of h_i. We make the following claims:

Fact IV.1. The polynomials h_i and h_i' are computed in terms of variables in the set X_j, where $X_j < x_i$ in RTTO.

We use Gröbner basis division to compute h_i. This division is shown in Eqn. (10). The order of division under RTTO ensures that h_i is a polynomial in those variables which are less than x_i in the term order. Similarly, the extended Gröbner basis algorithm used to compute h_i' in Eqn. (15), ensures that

h'_i is a polynomial in those variables which are less than x_i in the term order.

Fact IV.2.

$$h_i(U - h'_i) = \sum_{j=i+1}^{s} (h_j - h'_j)f_j + \sum_{x_l \in X_{PI}} (H_l - H'_l)(x_l^2 - x_l) \quad (16)$$

We get Eqn. (16) by combining Eqns. (10), (12) and (15).

Fact IV.3. Let $p \in \{0,1\}^{|X_{PI}|}$ be an assignment to the primary inputs of C. Let p' be the assignment to the all the nets in C under the application of p to X_{PI}; denote $p'|_{X_{PI}} = p$ as the projection of p' to primary inputs. Then, we have $\forall p \in \{0,1\}^{|X_{PI}|}$, $\exists p'$, such that $p'|_{X_{PI}} = p$ and $f_{i+1}(p') = \cdots = f_s(p') = 0$.

Clearly, for all primary input assignments, there exists an assignment to the nets that satisfies all the gates of the circuit. By using the three facts stated above, we state and prove the following results:

Theorem IV.1. Let there exist a polynomial U that maps from $\{0,1\}^{|X_j|} \to \{0,1\}$. Let p' be a point in the affine space such that $p'|_{X_{PI}} = p$ and $f_{i+1}(p') = \cdots = f_s(p') = 0$. If $h_i(p') \neq 0$, then $h'_i(p) = U(p')$.

Proof: Consider Eqn. (16) and point p' such that $p'|_{X_{PI}}$, and $f_{i+1}(p') = \cdots = f_s(p') = 0$. We know that $x_l^2 = x_l$ at any point $p \in \{0,1\}^{|X_{PI}|}$. Therefore, R.H.S. of Eqn. (16), when evaluated at p', is 0. Then, we have $h_i(p')(U(p') - h'_i(p')) = 0$. In L.H.S. of Eqn. (16), when $h_i(p') \neq 0$, $U(p') = h'_i(p')$.

Since we know (or have assumed) that there exists a (Boolean) rectification function U for C at x_i, we know that $U(p')$ evaluates to a value in $\{0,1\}$. Therefore, according to Theorem IV.1, h'_i evaluates to a Boolean value at those points where $h_i \neq 0$.

Corollary IV.1. Let point p be such that $p \in \{0,1\}^{|X_{PI}|}$. Let p' be a point such that $p'|_{X_{PI}} = p$. If $h_i(p') = 0$, the polynomial h'_i can be modified such that $h'_i(p')$ evaluates to a value in $\{0,1\}$.

Proof: Consider Eqn. (16). Consider a point p' such that $p'|_{X_{PI}}$ and $f_{i+1}(p') = \cdots = f_s(p') = 0$. We know that $x_l^2 = x_l$ at any point $p \in \{0,1\}^{|X_{PI}|}$. Therefore, R.H.S. of Eqn. (16) is 0.

$$h_i(U - h'_i) = 0. \quad (17)$$

In L.H.S. of Eqn. (17), $h_i(p') = 0$. Therefore, U can take any value in $\mathbb{Q}$ at p' to satisfy Eqn. (17).

The above results imply that $U(X_{PI})$ evaluate to Boolean values wherever $h_i(p') \neq 0$, and $U(X_{PI})$ may evaluate to non-Boolean values in $\mathbb{Q}$ where $h_i(p') = 0$. However, ideal membership Eqn. (10) is always satisfied when $h_i(p') = 0$ (irrespective of the value of $h'_i(p')$). Hence, we can modify the polynomial function $h'_i = U$ at those points p where $h_i(p) = 0$, to evaluate to any value in $\{0,1\}$, without affecting the ideal membership relation in Eqn (10). These points p where $h_i(p) = 0$ form the *don't care* set for the rectification function.

Based on the above discussion, the following claim can also be easily shown, and is stated without proof.

Corollary IV.2. If $\forall p \in \{0,1\}^{|X_{PI}|}$ and $h_i(p') \neq 0 \; \forall p'$ such that $p'|_{X_{PI}} = p$, then polynomial U is unique and can be computed as polynomial h'_i using Eqns. (6)-(15).

Overall Rectification Approach: i) Compute h_i and $U = h'_i$ as shown in Eqns. (6)-(15). ii) Compute $G = GB(h_i, F_0)$. If $G = \{1\}$, then it implies that h_i has no zeros, i.e. $h_i(p') \neq 0, \forall p' \in \{0,1\}^n$. Then $U = h'_i$ is a Boolean function and it can be uniquely computed with integral coefficients. U can then be reduced $\pmod 2$, where the products terms can be implemented as AND gates, and sum terms as XOR gates. iii) If $G = GB(h_i, F_0) \neq \{1\}$, then h_i evaluates to 0 at some points. Evaluate h_i for all points p', and where $h_i(p') = 0$, set $h'_i(p) = U(p)$ as *don't cares*, and generate a function table, which can be synthesized and attached to net x_i.

V. Experiments

We have performed experiments on buggy integer array multiplier circuits. We generated AIG files of array multipliers up to 64 bits by using the *gen -m* command from ABC [21]. The tool from the authors in [2] was used to input obtain *.SING files for the multiplier. We introduced bugs which could be corrected by a single-fix rectification. The complete single-fix rectification procedure is implemented using the SINGULAR symbolic algebra computation system [ver. 4-1-0][[22]]. The experiments were conducted on a desktop computer with a 3.5GHz Intel CoreTM i7-4770K Quad-core CPU, 16 GB RAM, running 64- bit Linux OS.

k	t_1	t_2	t_3	t_4
16	0.400	42.915	1.782	42.981
18	0.647	48.964	2.479	49.060
28	6.329	288.448	26.707	289.860
32	12.119	368.319	44.965	370.579
56	292.203	980.283	1221.654	1040.504
64	577.162	16.049	28.597	20.147

TABLE II: Single-fix rectification of integer multiplier against polynomial specification.

In the above table, t_1 is the time taken to perform verification of the buggy circuits. Times t_2 and t_3 indicate the time taken to find set of potentially rectifiable nets and identify a net x_i which admits single-fix rectification. The procedure to find potentially rectifiable nets and the rectification target net x_i is borrowed from [14]. Time t_4 is the time taken to compute a rectification polynomial using our approach.

An interesting observation in Table II is the difference in computation time between the 56-bit and 64-bit multiplier. Single-fix rectification in the 56-bit multiplier is attempted at a net x_i that is located deeper in the circuit, when compared to the placement of the rectification target in the 64-bit multiplier. This result shows that the efficiency of our approach depends on the location of the rectification target net.

Once we compute a rectification polynomial, we obtain an AND-XOR expression for $x_i = U$. We performed experiments to obtain synthesis results for rectified circuits, compared to the original correct circuit. The BLIF files were generated for the multipliers. The circuit was mapped to a logic library, *"cadence.genlib1"*, and area-delay stats recorded.

A bug b_1 was introduced in the circuit and rectification was attempted. We then used the command "$resyn$" in ABC to

k	32			56		
	b_1	b_2	b_3	b_1	b_2	b_3
$area_{(func)}$	9	41	312	9	87	TO
$delay_{(func)}$	4.00	8.00	17.00	4.00	11.00	TO
$area_{(rect)}$	6032	6186	6469	18608	18938	TO
$delay_{(rect)}$	184.00	184.00	184.00	328.00	328.00	TO
$penalty_{(area)}$	2.44%	5.06	9.86%	1.30%	3.1%	TO
$area_{(orig)}$	5888.00			18368.00		
$delay_{(orig)}$	184.00			328.00		

TABLE III: Synthesis Results. $area$ in sq. units. $delay =$ No. of gates.

re-synthesize the rectified circuit, which was mapped to the gates in the same logic library. The command *"print_stats"* was used to obtain area and delay properties of the rectified circuit. The above experiment was repeated for other bugs b_2 and b_3. The bug b_2 is located deeper in the circuit compared to b_1, and bug b_3 is located deeper in the circuit compared to b_2. The synthesis results are presented in Table III.

Here, $area_{(func)}$ and $delay_{(func)}$ are the area and delay numbers for the rectification function; $area_{(rect)}$ and $delay_{(rect)}$ are the area and delay numbers for the rectified circuit; $area_{(orig)}$ and $delay_{(orig)}$ are area and delay numbers for the original correct circuit. In all cases, the rectified circuit has a larger area compared to the original correct circuit. Arithmetic circuits are custom designed. However, we are synthesizing a sub-function to implement in the multiplier circuit and synthesizing the whole circuit. Logic synthesis for arithmetic circuits is not very efficient and that is reflected in these results.

Consider the bug b_3 in the 56-bit array multiplier. The bug is placed in the middle of the circuit and the Gröbner Basis reduction for verification times out. Therefore, a feasible rectification polynomial was not computed to rectify this bug. By observing the results shown in Table II and Table III, we can conclude that the location of the bug influences the efficiency of our approach. One of the limitations of our approach is that we witness the worst case of the complexity Gröbner Basis reduction based on the nature/location bugs.

VI. Conclusion

We model the rectification problem using concepts from algebraic geometry and use symbolic computer algebra algorithms to compute the rectification function $U(X_{PI})$. Using the model presented in [2], we obtain a system of polynomials $F = \{f_1, \ldots, f_s\}$ to describe the gates of the circuit over the field of rational numbers $\mathbb{Q}$. A net x_i in a buggy circuit is found to admit single-fix rectification, confirming the existence of a polynomial $x_i = U(X_{PI})$ in primary input variables that can rectify the circuit at net x_i. The problem of computing $U(X_{PI})$ is modeled as an extended ideal membership testing problem, which is solved using the Gröbner Basis algorithm. This paper investigates the nature of the rectification function, by stating the conditions under which the rectification polynomial we compute is a Boolean polynomial, i.e., it maps from $\{0,1\}^{|X_{PI}|} \rightarrow \{0,1\}$. We also introduce the concept of don't cares of the rectification function in the computer algebra setup. This paper describes the procedure to synthesize a logic circuit from the rectification polynomial we compute. By conducting experiments on integer array multipliers, we demonstrate the efficiency of our approach.

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] D. Kaufmann, A. Biere, and M. Kauers, "Incremental column-wise verification of arithmetic circuits using computer algebra," *Formal Methods in System Design*, Feb 2019. [Online]. Available: https://doi.org/10.1007/s10703-018-00329-2

[3] A. Mazhoon, D. Grosse, and R. Drechsler, "PolyCleaner: Clean your Polynomials before Backward Rewriting to Verify Million-Gate Multipliers," in *In Proc. ICCAD*, 2018.

[4] C. Yu, W. Brown, D. Liu, A. Rossi, and M. Ciesielski, "Formal Verification of Arithmetic Circuits by Function Extraction," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, 2016.

[5] C. Yu, M. Ciesielski, and A. Mishchenko, "Fast Algebraic Rewriting Based on And-Inverter-Graphs," *IEEE Trans. on CAD*, vol. 37, no. 9, pp. 1907–1911, Sept. 2018.

[6] O. Wienand, M. Wedler, D. Stoffel, W. Kunz, and G. Gruel, "An Algebraic Approach to Proving Data Correctness in Arithmetic Datapaths," in *Computer Aided Verification Conference*, 2008, pp. 473–486.

[7] 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.

[8] T. Pruss, P. Kalla, and F. Enescu, "Efficient Symbolic Computation for Word-Level Abstraction from Combinational Circuits for Verification over Finite Fields," *IEEE Trans. on CAD*, vol. 35, no. 7, pp. 1206–1218, July 2016.

[9] 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.

[10] 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.

[11] M. Fujita, "Toward Unification of Synthesis and Verification in Topologically Constrained Logic Design," *Proceedings of the IEEE*, 2015.

[12] K. Gitina, S. Reimer, M. Sauer, R. Wimmer, C. Scholl, and B. Becker, "Equivalence Checking of Partial Designs Using Dependency Quantified Boolean Formulae," in *IEEE International Conference on Computer Design (ICCD)*, 2013.

[13] U. Gupta, I. Ilioaea, V. Rao, A. Srinath, P. Kalla, and F. Enescu, "On the Rectifiability of Arithmetic Circuits using Craig Interpolants in Finite Fields," in *IFIP/IEEE International Conference on Very Large Scale Integration*, Oct 2018.

[14] V. Rao, U. Gupta, I. Ilioaea, A. Srinath, P. Kalla, and F. Enescu, "Post-verification debugging and rectification of finite field arithmetic circuits using computer algebra techniques," in *2018 Formal Methods in Computer Aided Design (FMCAD)*, Oct 2018, pp. 1–9.

[15] 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.

[16] U. Gupta, P. Kalla, I. Ilioaea, and F. Enescu, "Exploring algebraic interpolants for rectification of finite field arithmetic circuits using Gröbner bases," in *Proc. European Test Symposium (ETS)*, 2019 2019.

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

[18] A. Mahzoon, D. Grosse, and R. Drechsler, "Combining Symbolic Computer Algebra and Boolean Satisfiability for Automatic Debugging and Fixing of Complex Multipliers," in *The IEEE Symposium on VLSI (ISVLSI)*, 2018, pp. 351–356.

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

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

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

[22] 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.