Rectification of Integer Arithmetic Circuits using Computer Algebra Techniques

Vikas Rao Haden Ondricek Priyank Kalla
Department of Electrical & Computer Engineering
University of Utah, Salt Lake City, USA
vikas.k.rao@utah.edu haden.ondricek@utah.edu kalla@ece.utah.edu

Florian Enescu
Department of Mathematics & Statistics
Georgia State University, Atlanta, USA
fenescu@gsu.edu

2021 IEEE 39th International Conference on Computer Design (ICCD) | 978-1-6654-3219-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCD53106.2021.00039

***Abstract*—This paper proposes a symbolic algebra approach for multi-target rectification of integer arithmetic circuits. The circuit is represented as a system of polynomials and rectified against a polynomial specification with computations modeled over the field of rationals. Given a set of nets as potential rectification targets, we formulate a check to ascertain the existence of rectification functions at these targets. Upon confirmation, we compute the patch functions collectively for the targets. In this regard, we show how to synthesize a logic sub-circuit from polynomial artifacts generated over the field of rationals. We present new mathematical contributions and results to substantiate this synthesis process. We present two approaches for patch function computation: a greedy approach that resolves the rectification functions for the targets and an approach that explores a subset of don't care conditions for the targets. Our approach is implemented as custom software and utilizes the existing open-source symbolic algebra libraries for computations. We present experimental results of our approach on several integer multipliers benchmark and discuss the quality of the patch sub-circuits generated.**

***Index Terms*—Rectification, Debugging, Integer Arithmetic Circuits, Formal Methods, Symbolic Algebra.**

I. Introduction

Symbolic Computer Algebra (SCA) techniques are better suited for verification of arithmetic circuits as compared to conventional approaches based on Boolean satisfiability and decision diagrams [1], [2], [3]. The approaches based on decision diagrams are shown to exhibit exponential space complexity, whereas Boolean satisfiability techniques are proven to be not scalable. Methods based on theorem provers [4] require preservation of hierarchical information. On the other hand, the symbolic algebra techniques employing a *polynomial model* as the core computation engine has been shown to be the most effective approach for gate-level datapath verification. Such models verify the correctness of a circuit C by checking if the given polynomial specification $Spec$ is implied by a set of polynomials representing C. These algebraic models have been shown to be largely successful in verifying integer arithmetic circuits [1], [2], [3], [5], [6], [7], integer modulo arithmetic circuits [8], and also for finite field circuits [9], [10] – all with large operand sizes. Despite these advances in verification, automated debug and rectification of arithmetic circuits, especially multipliers, are still a significant challenge.

When formal verification detects the presence of a bug in the design, it is required to perform post-verification debug and *rectification* of the faulty implementation such that the rectified circuit matches the given specification. In such a setting, it is desirable to compute rectification functions at internal nets, without re-synthesizing/re-designing the entire circuit – a 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. Given a set of candidate nets as potential rectification targets, the rectification procedure requires i) to ascertain the rectifiability of the circuit at these targets, and ii) subsequently, compute a corresponding rectification function at each target.

While rectification has been addressed for random-logic circuits [11], [12], [13], [14], as well as recently for large operand finite field circuits [15], [16], the problem has not been satisfactorily addressed for integer arithmetic circuits. This paper addresses the problem of *ascertaining rectifiability and computing rectification functions for faulty integer arithmetic circuits at multiple targets*. While the concepts presented in this paper apply to any polynomial algebra-based model of integer arithmetic circuits, the application is demonstrated over integer multiplier benchmarks. For rectification, we assume that the candidate nets are provided beforehand, say, selected using contemporary signal selection heuristics [17], [13], [14].

Problem Statement: Given a gate-level integer arithmetic circuit C and a polynomial specification $Spec$ with integral coefficients, we model them over the field of rationals (denoted $\mathbb{Q}$). Formal verification is performed to verify C against $Spec$ (say, using the techniques of [18], [1], [3], [6]). Verification determines that C is buggy, i.e., C does not implement $Spec$. No assumptions are made about the type, nature, or number of bugs in C. A set of m candidate nets $\{w_1, \dots w_m\}$ from C are given as targets. Our *objective* is to ascertain if C is indeed rectifiable at the m targets, and if so, compute a set of individual *rectification polynomial functions* $U = \{u_1, \dots, u_m\}$ for the targets. And finally, we synthesize these polynomial functions into logic sub-circuit patches.

Related Previous Work: Once rectification is deemed feasible, the problem of finding the rectification function can be considered as a partial synthesis of an unknown component. The approach of [19] formulates the computation of the

This research is funded in part by the US National Science Foundation grants CCF-1911007 and CCF-1910368.

978-1-6654-3219-1/21/$31.00 ©2021 IEEE
DOI 10.1109/ICCD53106.2021.00039

unknown component as a quantified Boolean formula (QBF) and solves it using incremental and iterative SAT solving. Similarly, the authors in [20] present a QBF formulation for answering whether a partial implementation 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 [11] for multi-fix rectification for ECO synthesis.

The rectification problem has been addressed for finite field arithmetic circuits, particularly the ones that find applications in Elliptic Curve Cryptography. For this class of circuits, the problems are modeled and solved using polynomial algebra over finite fields. The concept of Craig interpolation was extended to polynomials in finite fields [15], [21] and application to rectification of finite field arithmetic circuits was demonstrated over large operand widths. As Craig interpolation in finite fields is a computationally challenging problem, another rectification approach was presented for finite field circuits in [16] which uses the (extended) Gröbner basis algorithm as a quantification procedure to compute the rectification function. Recently, [22] proposed an SCA-based approach *that decides m-target rectifiability* of finite field arithmetic circuits. Given a set of m-targets, the approach only ascertains whether *there exists* a set of patch functions at those targets, and cannot *compute* rectification functions.

In contrast, the approach presented in this paper addresses the rectification of integer arithmetic circuit designs, where problems are modeled over infinite fields. Rectification has been attempted for integer arithmetic circuits [23] [24]. Unfortunately, the technique of [23] was demonstrated to be incomplete in [16], and [24] is limited by the fault model, which only addresses a gate misplacement fault. In [25], a buggy integer arithmetic circuit is patched at all the primary outputs where the bug-effect is observable by implementing half-adders and carry-propagate logic at those outputs. However, the approach does not explore the possibility of rectification at internal nets and also does not explore don't care conditions for simplifying patch functions.

Approach and Contributions: We model the rectification problem using concepts from algebraic geometry and use symbolic computer algebra algorithms to determine rectifiability and to compute rectification functions. In [26], the authors showed that the *verification* of integer multipliers can be formulated and solved over polynomial rings with coefficients from $\mathbb{Q}$, as opposed to solving over integers $\mathbb{Z}$. As many algebraic geometry results are valid over fields, and $\mathbb{Z}$ is not a field, their approach leverages the fact that $\mathbb{Z} \subset \mathbb{Q}$, and it devises theory as well as computational techniques over fields to solve the verification problem.

Expanding upon the model presented in [26], we describe the gates of C with a set of polynomials with integer coefficients and perform all the algebraic computations required for rectification over $\mathbb{Q}$. Overall, our contributions are as follows:

1) We formulate a rectifiability check to ascertain that the circuit C is rectifiable at the given set of m targets.
   - This check implies the *existence of a polynomial function over* $\mathbb{Q}$ with mapping $u_i : \{0,1\}^{|X_{PI}|} \rightarrow \{0,1\}$, at individual targets w_i, where $1 \leq i \leq m$ and X_{PI} denotes the set of primary inputs of C. Substituting the polynomial function at the corresponding targets, $w_i = u_i[X_{PI}]$, rectifies the circuit.
2) We present two approaches for patch function computation, a heuristic that greedily resolves the rectification functions for the targets and a heuristic that explores a subset of *don't care conditions* for the targets.
3) Modeling rectification over $\mathbb{Q}$ poses new challenges because algebraic computations may result in polynomial functions with coefficients in $\mathbb{Q}$. We further propose a synthesis procedure that can translate such polynomial functions to Boolean rectification functions. We present new mathematical results and provide proof of soundness and completeness as part of this investigation.
4) We demonstrate the application of our approach by performing rectification of faulty integer multiplier circuits at multiple targets.

Paper Organization: The paper is organized as follows: The following section covers preliminary concepts and Sec. III covers the circuit modeling. Sec. IV describes the rectification check followed by rectification function computation in Sec. V. The synthesis of rectification functions is described in Sec. VI. Experiments are described in Sec. VII, and Sec. VIII concludes the paper.

II. Preliminaries

This section reviews basic concepts from symbolic algebra and associated algorithms that are utilized in this paper.

1) Background:

- Let $R = \mathbb{Q}[X]$ be the polynomial ring in variables $X = \{x_1, \ldots, x_d\}$ with coefficients in $\mathbb{Q}$ (field of rationals). Finite extensions of $\mathbb{Q}$ are called algebraic number fields, and the algebraic closure of $\mathbb{Q}$ is the field of algebraic numbers (denoted $\overline{\mathbb{Q}}$ in this work).
- A monomial is a power product over the variables X of the type $x_1^{e_1} \cdot x_2^{e_2} \cdots x_d^{e_d}$, $e_i \in \mathbb{Z}_{\geq 0}$.
- A polynomial $p \in R$ is written as a finite sum of terms $p = c_1X_1 + c_2X_2 + \cdots + c_tX_t$. Here $c_1, \ldots, c_t$ are coefficients $\in \mathbb{Q}$ and $X_1, \ldots, X_t$ are monomials.
- 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 $lt(p) = c_1X_1$, $lm(p) = X_1$, $lc(p) = c_1$, are the *leading term*, *leading monomial* and *leading coefficient* of p, respectively. Also, for a polynomial p, $tail(p) = p - lt(p)$.
   - In this work, we use *lexicographic* (lex) term orders.

2) Polynomial Ideals, Varieties, and Gröbner Basis: We are given a multivariate polynomial specification f in the ring R. The behavior of the gate-level circuit C is modeled as a set of polynomials $F = \{f_1, \ldots, f_s\}$ contained in R (Sec. III).

- Then $f \xrightarrow{F}_+ r$ denotes the *reduction* of f modulo the set of polynomials F, resulting in a remainder r. The terms in f are iteratively canceled by the leading terms of the polynomials in F using polynomial division (cf. Algorithm 1.5.1 [27]).

- The **ideal** generated by the set of polynomials F is defined as $J = \langle f_1, \dots, f_s \rangle = \{h_1 \cdot f_1 + \cdots + h_s \cdot f_s \mid h_1, \dots, h_s \in R\}$. The polynomials $f_1, .., f_s$ form the basis or generators of ideal J.
- Given an ideal $J \subseteq R$, the **radical** of J is defined as $\sqrt{J} = \{p \in R : \exists e \in \mathbb{N}, p^e \in J\}$.
  - When $\sqrt{J} = J$, J is said to be a *radical ideal.*
- An ideal $J \subset R$ is said to be **maximal** if $J \neq R$ and any ideal J' containing J is such that either $J' = J$ or $J' = R$.
- Let $\boldsymbol{a} = (a_1, \dots, a_d) \in \mathbb{Q}^d$ be a point in the affine space, and $p \in R$. If $p(\boldsymbol{a}) = 0$, we say that p *vanishes* on $\boldsymbol{a}$, or $\boldsymbol{a}$ is a *zero* of the polynomial p.
- We are interested in the *set of all common zeros (points)* of the polynomials of F that lie within the field $\mathbb{Q}$. This zero set is called the **variety**.
  - The variety depends not only on the set of polynomials F, but also on the ideal J generated by F. We denote the variety of J over $\mathbb{Q}$ as $V_{\mathbb{Q}}(J)$, where: $V_{\mathbb{Q}}(J) = V_{\mathbb{Q}}(\langle F \rangle) = \{\boldsymbol{a} \in \mathbb{Q}^d : \forall p \in J, p(\boldsymbol{a}) = 0\}$.

Boolean functions comprise a finite set of points, so we can model them as varieties. If a point is an element of a variety, then that point can be considered an on-set minterm of a corresponding Boolean function. The first two columns of Table I describe the correspondence between operations on Boolean functions and operations on varieties. We utilize this correspondence to formulate rectification check and to construct rectification functions by modeling the on-, off-, and DC-sets as varieties.

TABLE I: Correspondences between algebraic operations and Boolean operations. Here, $J_1 = \langle f \rangle$, and $J_2 = \langle g \rangle$.

Boolean Functions	*Varieties*	*Ideal operations*
$f \vee g$	$V(J_1) \cup V(J_2)$	$J_1 \cdot J_2$
$f \wedge g$	$V(J_1) \cap V(J_2)$	$J_1 + J_2$
$f - g$	$V(J_1) \setminus V(J_2)$	$J_1 : J_2$

Algebraic geometry analyzes ideals to reason about varieties without explicitly computing the varieties, which is infeasible in practice. In this paper, we compute the union, intersection, and set difference of functions, represented by varieties, by performing corresponding operations on ideals. In the third column, ” $\cdot$ ” , ” $+$ ” , ” $:$ ” represent the product, sum, and colon operations on ideals, respectively; these ideal operations are implemented in computer algebra tools, which we utilize.

- An ideal may have many different bases or generators. A *Gröbner basis* (GB) is a basis with properties that are useful in solving many polynomial decision and quantification problems. For example, a polynomial f is a member of ideal J if and only if $f \xrightarrow{GB(J)}_+ 0$. When $f \notin J$, division by $GB(J)$ results in a non-zero remainder $f \xrightarrow{GB(J)}_+ r$ that is unique/canonical.
- The Gröbner basis for an ideal J can be computed using the Buchberger's algorithm (Algorithm 1.7.1 in [27]). A Gröbner basis can be reduced. A reduced GB (redGB) is a canonical representation of an ideal. Thus, any operation on an ideal can be construed as being performed on its GB.

3) Variety over the Algebraic Closure of a Field: All variables representing the nets of a circuit only range over binary values, and thus satisfy the polynomial constraint relation $x^2 - x = 0$.

- Let $F_0 = \{x_l^2 - x_l : \forall x_l \in X\}$ denote the set of all such polynomials, and J_0 be the ideal generated by it, $J_0 = \langle F_0 \rangle$. The variety of J_0 is $V_{\mathbb{Q}}(J_0) = \{0, 1\}^d$.
- Consider the polynomial $g = x_l^2 - x_l \in \mathbb{Q}[X]$. The solution to $g = 0$ over $\overline{\mathbb{Q}}$, and also over $\mathbb{Q}$, is $\{0, 1\}$. Therefore, $V_{\mathbb{Q}}(J_0) = V_{\overline{\mathbb{Q}}}(J_0) = \{0, 1\}^d$. Here, $V_{\overline{\mathbb{Q}}}(J_0)$ denotes the variety of J_0 over the algebraic closure $\overline{\mathbb{Q}}$.
  - To enforce idempotency, we include the ideal J_0 in our computations. In other words, the circuit is modeled as an ideal $J + J_0$. We have that, $V_{\mathbb{Q}}(J + J_0) \subset \{0, 1\}^d$, and $V_{\mathbb{Q}}(J + J_0) = V_{\overline{\mathbb{Q}}}(J + J_0)$.
  - In this work, we simplify notation by writing $V(J)$ to mean $V_{\overline{\mathbb{Q}}}(J)$. However, we will use the notation $V_{\mathbb{Q}}$ and $V_{\overline{\mathbb{Q}}}$ when we need to specify which field the variety is considered over.

4) Quotient rings:

- Although circuit polynomials F and specification polynomial f contains integral coefficients; every polynomial has to satisfy the constraints imposed by the set F_0. Let $J = \langle F \rangle$ and $J_0 = \langle F_0 \rangle$. Thus, every computation is performed $\pmod{J_0}$ and effectively, we are working over the quotient ring $R/J_0 = \mathbb{Q}[X]/J_0$. Note that J_0 is a radical ideal in $\mathbb{Q}[X]$. This implies that R/J_0 is 0-dimensional and reduced, and hence R/J_0 is a *Von Neumann regular ring* [28], [29], which have special properties.
- There is a one-to-one correspondence between ideals $I \in R$ containing J_0 and ideals I' in R/J_0. The correspondence is given by $I \rightarrow I' = I/J_0$. As such, properties of $I' \in R/J_0$ lift to the corresponding ideal $I \in R$.
- Due to the fact that R/J_0 is a Von Neumann regular ring, it has the following properties:
  - Every ideal I' is radical in R/J_0. So, every ideal I of R such that $J_0 \subseteq I$ is radical in R, or equivalently $\sqrt{J + J_0} = J + J_0$, for any ideal J in R.
  - Every ideal I' is principal in R/J_0, which implies the ideal can be generated by a single element, i.e., $J + J_0 \pmod{J_0} = p \cdot R + J_0 \pmod{J_0}$ or $J + J_0 = p \cdot R + J_0$.
- This implies that we can compute a single rectification polynomial p by performing ideal computations on $J + J_0 \subset \mathbb{Q}[X]$.

III. Polynomial Modeling

Consider an n-bit gate-level integer multiplier circuit implementation C modeled over the ring $R = \mathbb{Q}[X]$. Let the two n-bit input vectors of C be $a_0, \dots, a_{n-1}$ and $b_0, \dots, b_{n-1}$, i.e. $X_{PI} = \{a_0, \dots, a_{n-1}, b_0, \dots, b_{n-1}\}$. Let $X_{PO} = \{s_0, \dots, s_{2n-1}\}$ denote the $2n$-bit output vector. Each internal gate is represented by a gate variable, and let $X_g = \{x_1, \dots, x_k\}$ denote the set of all such variables. Then, $X = X_{PO} \cup X_g \cup X_{PI}$.

Specification: Let f be a polynomial in R that acts as the specification for C. Given two n-bit input vectors a_i and b_i, f can be modeled as a polynomial in $\mathbb{Q}$ as:

$$f = \sum_{i=0}^{2n-1} 2^i s_i - \sum_{i=0}^{n-1} 2^i a_i \cdot \sum_{i=0}^{n-1} 2^i b_i \tag{1}$$

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 to polynomials over $\mathbb{Q}$. For example, the Boolean operators NOT, AND, OR, and XOR are represented as polynomials using the following relations:

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

Let $F = \{f_1, \dots, f_s\}$ represent the polynomial functions of all the gates in the circuit and $J = \langle F \rangle$ be the corresponding ideal. We also construct $J_0 = \langle F_0 \rangle = \langle x_l^2 - x_l : x_l \in X \rangle$. Then, ideal $J + J_0$ models the functionality of C. The verification problem can now be formulated as checking whether or not f is implied by the ideal $J + J_0$. In other words, $f \equiv C \iff f \xrightarrow{GB(J+J_0)}_+ 0$ [9], [26].

To systematically manipulate the polynomials, instead of imposing an arbitrary order, it is a standard practice to use a specialized term order called the *Reverse Topological Term Order (RTTO)*. RTTO is a lex term order with the circuit variables ordered reverse topologically from POs to PIs. RTTO has the property which ensures the polynomials of the circuit form a Gröbner basis themselves [9], [26]. As a consequence, $GB(J+J_0) = F \cup F_0$, and hence we can perform verification by checking whether the remainder r from the polynomial division $f \xrightarrow{F \cup F_0}_+ r$ equals zero. It was further shown [9] that in the ideal of vanishing polynomials J_0, it is sufficient to include vanishing polynomials in primary input variables ($X_{PI} \subset X$). Thus, ideal $J + J_0^{X_{PI}}$, where $J_0^{X_{PI}} = \langle F_0^{X_{PI}} \rangle = \{x_l^2 - x_l : \forall x_l \in X_{PI}\}$, models the functionality of the circuit.

Example III.1. *Fig. 1 represents a faulty implementation C of a two-bit integer multiplier circuit. A correct multiplier would have XOR gates in place of the AND gates at nets e_7 and z_1. The Spec polynomial for this two-bit multiplier is modeled according to III: $f = (8 \cdot z_3 + 4 \cdot z_2 + 2 \cdot z_1 + z_0) - ((2 \cdot a_1 + a_0) \cdot (2 \cdot b_1 + b_0))$. We impose RTTO $>$ as lex with: $\{z_0 > z_1 > z_2 > z_3\} > \{e_9 > e_5\} > \{e_8 > e_4\} > \{e_7 > e_1\} > \{e_6 > e_2\} > \{e_3\} > \{a_0 > a_1 > b_0 > b_1\}$.*

We map the Boolean gates of C to polynomials over Q:

$$\begin{aligned}
f_1 &: z_3 - (e_9 \cdot e_5); & f_8 &: e_4 - (a_0 \cdot b_1); \\
f_2 &: z_2 - (e_9 + e_5 - 2 \cdot e_9 \cdot e_5); & f_9 &: e_5 - (b_1 \cdot a_1); \\
f_3 &: z_1 - (e_4 \cdot e_1); & f_{10} &: e_6 - (e_3 \cdot b_0); \\
f_4 &: z_0 - (a_0 \cdot b_0); & f_{11} &: e_7 - (e_6 \cdot e_2); \\
f_5 &: e_1 - (a_1 \cdot b_0); & f_{12} &: e_8 - (e_7 \cdot e_1); \\
f_6 &: e_2 - (1 - b_0); & f_{13} &: e_9 - (e_8 \cdot e_4); \\
f_7 &: e_3 - (b_0 + a_0 - a_0 * b_0); & &
\end{aligned}$$

The polynomials highlighted in red specify the gates where bugs were introduced into the circuit. For simplicity, we select these two nets as targets for rectification ($w_1 = e_7, w_2 = z_1$). Let $J = \langle f_1, \cdots, f_{13} \rangle$ and $J_0^{X_{PI}} = \langle a_3^2 - a_3, \dots, b_0^2 - b_0 \rangle$.

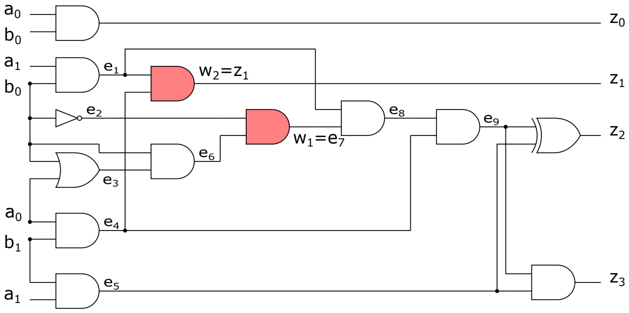

Fig. 1: Buggy two-bit multiplier circuit with added redundancy. Red gates were changed from XOR gates to AND gates. Labels w_1, w_2 indicate the two selected rectification targets.

To perform verification, we reduce the Spec polynomial by the ideal $J + J_0^{X_{PI}}$ which models C. We find that $f \xrightarrow{J+J_0^{X_{PI}}}_+ r = 2a_0a_1b_0b_1 - a_0b_1 - 2a_1b_0$, confirming that C is buggy.

IV. Rectification Check

If verification results in a non-zero remainder r, then the circuit is deemed faulty and rectification needs to be performed to fix the circuit. As a first step, it is required to derive the necessary and sufficient condition to confirm whether or not the circuit can be rectified at a given set of m targets. This condition implies the *existence of a polynomial function* with mapping $u_i : \{0,1\}^{|X_{PI}|} \rightarrow \{0,1\}$, at individual targets.

Let $W = (w_1, \dots, w_m) \subset \{X_g, X_{PO}\}$ denote a given set of m candidate targets. Let $W_c = \{(0,0,\dots,0), \dots, (1,1,\dots,1)\}$ represent the set of all $\{0,1\}$-assignments to targets W. Here, $|W| = m$ and $|W_c| = 2^m$, with each $W_c[l]$, $1 \le l \le 2^m$, representing one set of assignments to targets W ($|W_c(l)| = m$). In other words, the tail of gate polynomials representing the targets are replaced by the corresponding $\{0,1\}$ values from $W_c[l]$.

Theorem IV.1. A *Spec* polynomial f, a faulty circuit C, and a set of targets W from C are given. The circuit C is represented using the ideal $J + J_0 \subset R$, wherein the targets in set W are considered fan-in free or treated as pseudo primary inputs. RTTO $>$ is imposed on R.

The following ideals are constructed:

- $J_l = \langle F_l \rangle = \langle f_1, \dots, f_{w_1} : w_1 - W_c[l][1], \dots, f_{w_m} : w_m - W_c[l][m] \ , \dots, f_s \rangle, \forall l \in 1, \dots 2^m$.

Reduce f by $J_l + J_0$ to obtain remainders rem_l: $f \xrightarrow{J_l+J_0}_+ rem_l$, for $1 \leq l \leq 2^m$. Then, the faulty circuit C is rectifiable at the target set W ***if and only if***

$$\bigcup_{l=1}^{2^m} V(\langle rem_l \rangle + J_0^{X_{PI}}) = \{0,1\}^{|X_{PI}|} \qquad (2)$$

where, $J_0^{X_{PI}} = \langle F_0^{X_{PI}} \rangle = \{x_l^2 - x_l : \forall x_l \in X_{PI}\}$.

Proof IV.1. If rectification is feasible, then a rectification at the target-set W makes C match f. Consequently, f should be implied by the *patched ideal* $J + J_0$ representing the rectified circuit, or f should vanish on $V(J + J_0)$. Moreover, each rem_l comprises only X_{PI} variables. This is because RTTO $>$ ensures that each non-primary input variable (each gate output) appears as the leading term of some polynomial in F and all such leading terms are canceled in the reduction $f \xrightarrow{J_l+J_0}_+ rem_l$. Furthermore, as X_{PI} take values in $\{0,1\}$, $V(rem_l) \subseteq \{0,1\}^{|X_{PI}|}$. Thus, the rectification condition can be equivalently stated as: "f vanishes on $V(J + J_0) \iff \bigcup_{l=1}^{2^m} V(\langle rem_l \rangle + J_0^{X_{PI}}) = \{0,1\}^{|X_{PI}|}$".

(i) **To prove "$\Rightarrow$"**: Let $x_{PI} \in \{0,1\}^{|X_{PI}|}$ be an assignment to the primary input variables of C. Every assignment x_{PI} results in a corresponding assignment x_{int} to rest of the variables in C. For each such point $(x_{PI}, x_{int}) \in \{0,1\}^{|X|}$, the set of m targets W evaluate to one set of possible $\{0,1\}$ assignment from W_c. When the m-targets in W evaluate to $W_c[1] = (0, 0, \dots, 0)$, J_1 vanishes on the point (x_{PI}, x_{int}). Likewise, J_2 vanishes on (x_{PI}, x_{int}) when the targets evaluate to $W_c[2] = (0, 0, \dots, 1)$, and so on. Since $f \xrightarrow{J_l+J_0}_+ rem_l, 1 \leq l \leq 2^m$, and f vanishes on the point (x_{PI}, x_{int}) to begin with, we obtain that for every primary input assignment x_{PI}, one of the rem_l vanishes. This implies that $\bigcup_{l=1}^{2^m} V(\langle rem_l \rangle + J_0^{X_{PI}}) = \{0,1\}^{|X_{PI}|}$.

(ii) **To prove "$\Leftarrow$"**: Say there exists an assignment to the primary inputs $x_{PI} \in \{0,1\}^{|X_{PI}|}$ such that rem_1 vanishes on x_{PI}, i.e. $rem_1(x_{PI}) = 0$. For the given point x_{PI}, the rest of the variables of C get a corresponding assignment x_{int}. As $f \xrightarrow{J_1+J_0}_+ rem_1$, we have that f is a member of the ideal $J_1 + J_0 + \langle rem_1 \rangle$. Therefore, when $rem_1(x_{PI}) = 0$, the ideal J_1 also vanishes on $(x_{PI}, x_{int}) \in \{0,1\}^{|X|}$ because the point (x_{PI}, x_{int}) is a valid evaluation of the circuit. Further, J_0 by definition vanishes on all the points $\in \{0,1\}^{|X|}$. This implies that $f(x_{PI}, x_{int}) = 0$. The argument similarly holds for each rem_l vanishing on some x_{PI}. This proves that for all primary inputs, if any $rem_l : 1 \leq l \leq 2^m$ vanishes, then f vanishes too; and that completes the proof.

□

Intuitively, the above theorem can be elaborated as follows. The variety of rem_l, for any l, corresponds to the set of all assignments to primary inputs X_{PI} (minterms) where the *Spec* f agrees with the *Impl* C. Thus, the condition of Thm. IV.1 implies that every minterm in the input space is contained in the union of varieties of each rem_l. Thus, for every minterm from the input space, there exists an assignment $W_c[l]$ to W where f and C match. Consequently, there exists a set of individual rectification functions for the targets that can be computed to rectify every error minterm.

In the above check, the computation with the union of varieties $\bigcup_{l=1}^{2^m} V(\langle rem_l \rangle + J_0^{X_{PI}}) = \{0,1\}^{|X_{PI}|}$ can be performed as a product of ideals, i.e. by checking if $\prod_{l=1}^{2^m} rem_l \xrightarrow{J_0^{X_{PI}}}_+ 0$.

Example IV.1. *Continuing Ex. III.1, we demonstrate the rectification check for* $W = (w_1, w_2) = (e_7, z_1)$.

Constructing the J_l *ideals:*

- $J_1 = \langle F_1 \rangle$; $F_1[f_3 : z_1 - 0, \ f_{11} : e_7 - 0], (z_1 = 0, e_7 = 0)$
- $J_2 = \langle F_2 \rangle$; $F_2[f_3 : z_1 - 0, \ f_{11} : e_7 - 1], (z_1 = 0, e_7 = 1)$
- $J_3 = \langle F_3 \rangle$; $F_3[f_3 : z_1 - 1, \ f_{11} : e_7 - 0], (z_1 = 1, e_7 = 0)$
- $J_4 = \langle F_4 \rangle$; $F_4[f_3 : z_1 - 1, \ f_{11} : e_7 - 1], (z_1 = 1, e_7 = 1)$

Reducing the Spec$f$ *modulo these ideals results in:*

- $rem_1 = f \xrightarrow{J_1+J_0^{X_{PI}}}_+ -2a_0b_1 - 2a_1b_0$,
- $rem_2 = f \xrightarrow{J_2+J_0^{X_{PI}}}_+ -2a_0b_1 - 2a_1b_0 + 2$,
- $rem_3 = f \xrightarrow{J_3+J_0^{X_{PI}}}_+ 4a_0a_1b_0b_1 - 2a_0b_1 - 2a_1b_0$,
- $rem_4 = f \xrightarrow{J_4+J_0^{X_{PI}}}_+ 4a_0a_1b_0b_1 - 2a_0b_1 - 2a_1b_0 + 2$.

When we compute $\prod_{l=1}^{2^m} rem_l \xrightarrow{J_0^{X_{PI}}}_+$, *we obtain remainder 0, thus confirming that the target set W indeed admits correction.*

V. Computing Rectification Functions

For a given set of targets W, due to the presence of don't cares (DC), there may exist more than one set U of rectification functions which rectify the circuit. Exploring all the DC conditions for m targets might be computationally infeasible; we present two different approaches to overcome this. First, we present an approach to compute an on- and off-set for each rectification function by greedily resolving all the DC conditions. Following this, we present an approach to heuristically explore and compute a subset of the DC conditions, along with on- and off-sets, for each rectification function.

1) Greedy Approach for MFR: To illustrate the greedy approach, consider the case with $m = 2$ ($W = \{w_1, w_2\}$), where $W_c = \{(0,0), (0,1), (1,0), (1,1)\}$, and we must compute rectification functions u_1 and u_2 corresponding to targets w_1 and w_2, respectively. For brevity, let $V_{W_c[l]} = V(\langle rem_l \rangle + J_0^{X_{PI}})$, for $1 \leq l \leq 2^m$; in this case, $V_{W_c[1]} = V_{(0,0)} = V(\langle rem_1 \rangle + J_0^{X_{PI}}), V_{W_c[2]} = V_{(0,1)} = V(\langle rem_2 \rangle + J_0^{X_{PI}})$, and so on.

Recall that $V_{(0,0)}$ comprises the set of points where the *Spec* matches the *Impl* under the assignments $w_1 = w_2 = 0$ to the targets. This implies that at these points, the rectification functions u_1 and u_2 should evaluate to 0. Table II shows the required evaluations of u_1 and u_2 for the points in each variety, following the same reasoning, assuming each $V_{W_c[l]}$ is pairwise disjoint. The on (off)-set of the rectification function for a target corresponds to the union of the varieties where the function evaluates to 1 (0). In this case, the on- and off-sets of u_1 consist of the set of points in $V_{(1,0)} \cup V_{(1,1)}$ and $V_{(0,0)} \cup V_{(0,1)}$, respectively. Similarly, the on- and off-set of u_2 comprise points in $V_{(0,1)} \cup V_{(1,1)}$ and $V_{(0,0)} \cup V_{(1,0)}$. The functions u_1 and u_2 could be synthesized using these sets.

TABLE II: Required rectification function evaluations

Variety	u_1	u_2
$V_{(0,0)}$	0	0
$V_{(0,1)}$	0	1
$V_{(1,0)}$	1	0
$V_{(1,1)}$	1	1

However, the above argument is only correct when each $V_{W_c[l]}$ are pairwise disjoint, which may not be true in practice. For example, for a point contained in $V_{(0,0)} \cap V_{(0,1)}$, (u_1, u_2) may evaluate either to $(0, 0)$, or to $(0, 1)$ in order for the *Impl* to evaluate to the same value as the *Spec*; this point would be in both the on- and off-set of u_2 in the method previously described. A decision procedure is necessary to determine the evaluation of (u_1, u_2) at these intersections, unambiguously. *We present a greedy approach which resolves such ambiguities by imposing an order on the sets.*

An example of our greedy approach to evaluate (u_1, u_2) for an order $V_{(0,0)} > V_{(0,1)} > V_{(1,0)} > V_{(1,1)}$ is as follows: First, we place all the points from $V_{(0,0)}$ into the off-sets of (u_1, u_2). Next, we place all the points from $V_{(0,1)} \setminus V_{(0,0)}$ into the off-set of u_1 and the on-set of u_2. We perform the set difference to avoid placing the points in $V_{(0,0)} \cap V_{(0,1)}$ into both the on-set and off-set of u_2. Next, we place all the points from $V_{(1,0)} \setminus (V_{(0,0)} \cup V_{(0,1)})$ into the on-set of u_1, and the off-set of u_2. Finally, we place the remaining points from $V_{(1,1)} \setminus (V_{(0,0)} \cup V_{(0,1)} \cup V_{(1,0)})$ into the on-set of (u_1, u_2). The resulting on- and off-sets for u_1 and u_2 are shown below.

$$
\begin{aligned}
V(u_{1_{on}}) &= (V_{(1,1)} \setminus (V_{(0,0)} \cup V_{(0,1)} \cup V_{(1,0)})) \cup (V_{(1,0)} \setminus (V_{(0,0)} \cup V_{(0,1)})) \\
V(u_{1_{off}}) &= (V_{(0,0)}) \cup (V_{(0,1)} \setminus V_{(0,0)}) \\
V(u_{2_{on}}) &= (V_{(0,1)} \setminus V_{(0,0)}) \cup (V_{(1,1)} \setminus (V_{(0,0)} \cup V_{(0,1)} \cup V_{(1,0)})) \\
V(u_{2_{off}}) &= (V_{(0,0)}) \cup (V_{(1,0)} \setminus (V_{(0,0)} \cup V_{(0,1)}))
\end{aligned}
$$

This approach with the given order greedily places points into the off-sets of the rectification functions (u_1, u_2) where possible and only places points into the on-sets of the rectification functions when necessary. Subject to the given order, the on-sets of the rectification functions are thus minimized. For the experiments in this paper, we always use the order $V_{W_c[i]} > V_{W_c[j]}$ for $i < j$, as in the above example, though any order would yield valid rectification functions.

Generalizing our greedy approach for m targets, we first construct the following composite sets (varieties):

$$
S_l = \begin{cases} V_{W_c[1]}, & \text{if } l = 1 \\ V_{W_c[l]} \setminus (\bigcup_{j=1}^{l-1} V_{W_c[j]}), & 2 \leq l \leq 2^m \end{cases} \tag{3}
$$

The resulting on-set and off-set functions for each target i, where $1 \leq i \leq m$ are:

$$
\begin{aligned}
V(u_{i_{on}}) &= \bigcup S_l, \ \forall l \mid W_c[l][i] = 1 \\
V(u_{i_{off}}) &= \bigcup S_l, \ \forall l \mid W_c[l][i] = 0
\end{aligned} \tag{4}
$$

2) Don't Care Conditions for MFR: Let $U_d \subseteq U$ denote a subset of the target rectification functions. We are interested in the DC conditions which arise for these functions at points where they may evaluate to any value, for some fixed evaluation of the remaining functions in the set $\{U \setminus U_d\}$. For example, consider a point in $V_{(0,0)} \cap V_{(0,1)}$ for a circuit with two targets. As discussed previously, u_1 must evaluate to 0 at this point, but $U_d = \{u_2\}$ may evaluate either to 0 or to 1, so this is a DC point for u_2.

Not every intersection of varieties yields DC points which follow the conditions described above. Consider a point in $V_{(0,0)} \cap V_{(1,1)}$. Here, (u_1, u_2) must evaluate either to $(0, 0)$ or to $(1, 1)$. If this point were assigned to the DC set of u_2, for example, the *Spec* and *Impl* would only evaluate the same if u_1 evaluated to the same value as u_2. Thus, u_1 would become a function of u_2 at this point. This point cannot be placed into the on-set, off-set, or DC-set of u_1 before u_2 is evaluated. To avoid inter-dependencies between the rectification functions, we do not classify points in such intersections as DC points. We rely on our greedy heuristic to evaluate these points.

Finally, consider a point in $V_{(0,0)} \cap V_{(0,1)} \cap V_{(1,0)}$. This point cannot be a DC point for both targets simultaneously since the evaluation $(1, 1)$ here will result in an incorrect rectification function. However, because $V_{(0,0)} \cap V_{(0,1)} \cap V_{(1,0)} \subset V_{(0,0)} \cap V_{(0,1)}$, we could treat this point as a DC point for u_2 and evaluate u_1 to 0. Alternatively, because $V_{(0,0)} \cap V_{(0,1)} \cap V_{(1,0)} \subset V_{(0,0)} \cap V_{(1,0)}$, we could treat this point as a DC point for u_1 and evaluate u_2 to 0. Thus, we have a choice to place this point in the DC-set of either targets, but not both.

Finding every intersection containing DC points for every target can be very expensive for circuits with more than a few targets. We therefore propose an approach to compute a subset of the DC points by considering only the set of pairwise intersections of varieties which contain DC points for exactly one target, denoted as DC_{pair}.

Let $d(W_c[j], W_c[k])$ denote the Hamming distance between the two sets of assignments to the targets $W_c[j]$ and $W_c[k]$. We compute the set of varieties which contain DC points for one target, denoted DC_{pair}, from the equation below, where $1 \leq j, k \leq 2^m$.

$$
DC_{pair} = \{V_{W_c[j]} \cap V_{W_c[k]} \mid d(W_c[j], W_c[k]) = 1\} \tag{5}
$$

Since the Hamming distance $d = 1$ between the assignments $W_c[j]$ and $W_c[k]$ for each intersection of varieties in DC_{pair}, exactly one rectification function may evaluate either to 0 or to 1. The remaining rectification functions require fixed evaluations of 1 or 0. Therefore, each intersection of varieties in DC_{pair} yields DC points for exactly one rectification function in U, and either on- or off-set points for the remaining rectification functions in U. We use DC_{pair} to compute the DC points for each rectification function, as described below.

3) Computing Rectification Functions with Don't Cares: Once the set DC_{pair} has been found, a few steps remain to compute the on-, off-, and don't-care sets for each target. First, we follow an approach identical to the greedy approach to evaluate points outside of DC_{pair}. We construct new composite sets S_l^d for $1 \leq l \leq 2^m$, which are identical to the composite sets (varieties) created for the previous approach, except that all the points from DC_{pair} set are removed.

$$S_l^d = \begin{cases} V_{W_c[1]} \setminus DC_{pair}, & \text{if } l = 1 \\ V_{W_c[l]} \setminus ((\bigcup_{j=1}^{l-1} V_{W_c[j]}) \cup DC_{pair}), & 2 \leq l \leq 2^m \end{cases} \quad (6)$$

Points in these composite sets are assigned to the on- and off-set for each rectification function in the same way as Eqn. (4), substituting S_l with S_l^d. Next, we place the points in DC_{pair} in the on-, off-, or DC sets for each rectification function, by imposing an order on the intersections and resolving them as explained in the following example.

Given a circuit with two targets, $DC_{pair} = \{V_{(0,0)} \cap V_{(0,1)}, V_{(0,0)} \cap V_{(1,0)}, V_{(0,1)} \cap V_{(1,1)}, V_{(1,0)} \cap V_{(1,1)}\}$. We impose the order $V_{(0,0)} > V_{(0,1)} > V_{(1,0)} > V_{(1,1)}$. We place the points in $V_{(0,0)} \cap V_{(0,1)}$ into the off-set of u_1 and the DC set of u_2. We then place the points in $V_{(0,0)} \cap V_{(1,0)} \setminus V_{(0,0)} \cap V_{(0,1)}$ into the DC set of u_1 and the off-set of u_2. We place points in $V_{(0,1)} \cap V_{(1,1)} \setminus ((V_{(0,0)} \cap V_{(0,1)}) \cup (V_{(0,0)} \cap V_{(1,0)}))$ into the DC set of u_1 and the on-set of u_2. Finally, we place points in $V_{(1,0)} \cap V_{(1,1)} \setminus ((V_{(0,0)} \cap V_{(0,1)}) \cup (V_{(0,0)} \cap V_{(1,0)}) \cup (V_{(0,1)} \cap V_{(1,1)}))$ into the on-set of u_1 and the DC set of u_2. Following this approach, we calculate on- off- and DC sets for each rectification function.

VI. Synthesizing Rectification Functions

The above techniques show how to construct a rectification function by reasoning about the varieties of each rem_l. However, algebraically, these functions are computed using their corresponding ideals. We now show how the remainders computed in Sec.IV can be utilized for rectification function computation.

The rectification theorem IV.1 implies the *existence of a polynomial function* with mapping $u_i : \{0,1\}^{|X_{PI}|} \rightarrow \{0,1\}$, at individual targets. However, since we use a polynomial model over $\mathbb{Q}$, our algebraic rectification techniques compute a polynomial function over $\mathbb{Q}$, i.e., it may evaluate to constants in $\mathbb{Q}$, i.e., non-Boolean values. To overcome this, we present new mathematical contributions to transform a polynomial function of the form $\langle rem_l \rangle + J_0^{X_{PI}}$ to another polynomial rem_l' such that $V(\langle rem_l \rangle + J_0^{X_{PI}}) = V(\langle rem_l' \rangle + J_0^{X_{PI}})$, and rem_l' evaluates to only Boolean values $\{0,1\} \subset \mathbb{Q}$ and corresponds to the rectification function evaluation.

Consider the quotient ring $R/J_0 = \frac{\mathbb{Q}[X]}{J_0}$. Let $J \subseteq \mathbb{Q}[X]$ be an ideal with coefficients in $\mathbb{Q}$ and so $V(J + J_0) \subseteq \{0,1\}^{|X|}$. Moreover, $J + J_0 \pmod{J_0}$ is radical in R/J_0, and $J + J_0$ is radical in R. Given that ideals $J + J_0$ correspond to a finite variety, and that they are radical and principal, from the above discussion, we show the following results.

Fact VI.1. The ideal $J + J_0$ can be expressed as the intersection of maximal ideals. In $\mathbb{Q}[X]$ the intersection of maximal ideals is the same as their product: $J + J_0 = \bigcap_{i=1}^k M_i = \prod_{i=1}^k M_i$, where M_i are maximal in R. Furthermore, each such maximal ideal M_i is of the form $M = (x_1 - \epsilon_1, \ldots, x_d - \epsilon_d)$, where $\epsilon_j \in \{0,1\}, \forall 1 \leq j \leq d$.

Every radical ideal can always be decomposed into an intersection of maximal ideals. While the result is well known over algebraically closed fields, it is also true over $\mathbb{Q}[X]$.

Proposition VI.1. The ideal $\prod_{i=1}^k M_i$ can be expressed as generators $(f_1, \ldots, f_s, x^2 - x : \forall x \in X_{PI})$ such that $f_1, \ldots, f_s$ have coefficients only in $\{-1, +1\}$.

Since each M_i is of the form $(x_i - \epsilon)$, their intersection also have coefficients only in $\{-1, +1\}$. The above implies that any arbitrary polynomial $p \in R$, when combined with J_0, can be expressed as $\langle p \rangle + J_0 = \langle f_1, \ldots, f_s \rangle + J_0$, where $f_1, \ldots, f_s$ have coefficients in $\{-1, +1\}$.

Even though the remainders rem_l have coefficients in $\mathbb{Q}$ (higher field), their varieties are in $\{0,1\}^{|X_{PI}|}$ ($V(\langle rem_l \rangle) \subseteq \{0,1\}^{|X_{PI}|}$) as they correspond to bit-level assignments to X_{PI}.

Proposition VI.2. Compute the GB G_l for the ideals $(\langle rem_l \rangle + J_0^{X_{PI}}) \subset \mathbb{Q}[X]$, $1 \leq l \leq 2^m$, as $reducedGB(\langle rem_l \rangle + J_0^{X_{PI}})$, where rem_l's are the remainders generated in Thm. IV.1. Then the polynomials in G_l have coefficients in $\{-1, +1\}$ with variables in X_{PI}.

The above results imply that a reduced Gröbner basis of an ideal of the form $\langle f \rangle + J_0^{X_{PI}}$ for any arbitrary polynomial $f \in \mathbb{Q}[X_{PI}]$ will have generators $G = \{g_1, \ldots, g_t\}$ with coefficients from $\{+1, -1\}$. This enables the synthesis of Boolean rectification functions from these Gröbner bases.

Example VI.1. Let $p = 4/3 \cdot a_0 a_1 b_0 b_1 - 2 \cdot a_0 b_0 b_1 - 2/7 \cdot a_1 b_0$ be a polynomial in $R = \mathbb{Q}[a_0, a_1, b_0, b_1]$. Let $J_0 = \langle a_0^2 - a_0, \ldots, b_1^2 - b_1 \rangle$.

Computing $G = redGB(\langle p \rangle + J_0) = \{a_0 b_0 b_1, a_1 b_0\}$.

Note that the polynomials in G have coefficients in $\{-1, +1\}$.

Proposition VI.3. Given a set of generators $F = \{f_1, \ldots, f_s\}$ with coefficients only in $\{-1, +1\}$, a polynomial p can always be constructed as $p = ((1 + f_1)(1 + f_2) \ldots (1 + f_s) + 1) \pmod 2$, such that $V(\langle p \rangle + J_0) = V(\langle f_1, \ldots, f_s \rangle + J_0)$ and p evaluates to only binary values $\{0,1\} \subset \mathbb{Q}$.

We utilize the above facts and the reduced GB computation to construct a synthesizable Boolean function rem_l' from a polynomial function rem_l.

Overall, our procedure for the rectification of integer arithmetic circuits is as follows:

- Model the specification polynomial and the circuit polynomials over $\mathbb{Q}[X]$ using techniques described in Sec. III.
- Generate remainders rem_l and formulate the rectification check for a given set of m targets as $\prod_{l=1}^{2^m} rem_l \xrightarrow{J_0^{X_{PI}}}_+ 0$ (Thm. IV.1).
  - Here, each rem_l corresponds to a polynomial function mapping from $\{0,1\}^{|X_{PI}|} \rightarrow \mathbb{Q}^{|X_{PI}|}$.
- If the check confirms existence of rectification functions, then compute functions $u_i,\ i = 1, \dots, m$ corresponding to each target as follows:

1) Compute a reduced GB for each $(\langle rem_l \rangle + J_0^{X_{PI}})$ as shown in Prop. VI.2.
   - Here, the generators from each $redGB(\langle rem_l \rangle + J_0^{X_{PI}})$ will have coefficients only in $\{-1, +1\}$, and $V(\langle rem_l \rangle + J_0^{X_{PI}})$ corresponds to the minterms of the rectification function.
2) Construct a singleton polynomial rem_l' such that, $V(\langle rem_l' \rangle + J_0^{X_{PI}}) = V(\langle rem_l \rangle + J_0^{X_{PI}})$, as shown in Prop. VI.3.
   - Here, rem_l' is a polynomial function mapping from $\{0,1\}^{|X_{PI}|} \mapsto \{0,1\}$, such that it has the same variety.
3) Impose the order on the remainders: $rem_1' > \cdots > rem_l'$.
4) Greedy approach: Compute composite sets from Eqn. (3).
   - Assign points from composite sets to the on- or off-sets of the rectification functions (Sec. V-1).
5) DC-based approach: Compute DC_{pair} using Eqn. (5), and then compute the composite sets in Eqn. (6)
   - Assign the points in the composite sets and DC_{pair} to the DC-, on- or off-sets of the rectification functions (Sec. V-2).
6) Perform the union, intersection, and set difference of varieties using the respective ideal operations (Table I).
7) Translate the polynomials representing $u_{i_{DC}}$ and $u_{i_{on}}$ into Boolean functions $u_{i_{DC}}^{\mathbb{B}}$ and $u_{i_{on}}^{\mathbb{B}}$ by interpreting the algebraic product and sum as Boolean AND and XOR gates, respectively.
8) Optimize the on-set $u_{i_{on}}^{\mathbb{B}}$ w.r.t. to the DC-set $u_{i_{DC}}^{\mathbb{B}}$ for the DC-based approach using a logic synthesis tool.

Example VI.2. Continuing with Ex. IV.1, consider rem_3:

- Recall, $rem_3 = 4a_0a_1b_0b_1 - 2a_0b_1 - 2a_1b_0$.
- $redGB(\langle rem_3 \rangle + J_0) = \{a_1b_0b_1 - a_1b_0, a_0b_1 - a_1b_0, a_0a_1b_0 - a_1b_0\}$
  - Note, $V(\langle rem_3 \rangle + J_0) = V(redGB(\langle rem_3 \rangle + J_0))$
  - Compute a reduced GB for rem_1, rem_2, and rem_4.
- Impose the order $rem_1 > rem_2 > rem_3 > rem_4$.
- The rectification polynomials for the targets (e_7, z_1) computed using our greedy approach:

$$\begin{aligned}
u_{1_{on}} &= a_0a_1b_0b_1;\\
e_7 = u_1^{\mathbb{B}} &= (a_0 \wedge a_1 \wedge b_0 \wedge b_1);\\
u_{2_{on}} &= a_0b_1 + a_1b_0;\\
z_1 = u_2^{\mathbb{B}} &= (a_0 \wedge b_1) \oplus (a_1 \wedge b_0);
\end{aligned}$$

- The rectification polynomials for the targets (e_7, z_1) computed using our DC-based approach:

$$\begin{aligned}
u_{1_{on}} &= a_0a_1b_0b_1; \quad u_{1_{dc}} = a_0a_1b_0b_1 + 1;\\
e_7 &= u_1^{\mathbb{B}} = \mathbf{1};\ (after\ logic\ optimization)\\
u_{2_{on}} &= a_0b_1 + a_1b_0; \quad u_{2_{dc}} = 1;\\
z_1 &= u_2^{\mathbb{B}} = (a_0 \wedge b_1) \oplus (a_1 \wedge b_0);
\end{aligned}$$

Note that with DC conditions extracted using our approach, the synthesis tool was able to optimize the logic at net e_7 to tautology.

VII. Experiments

This section presents experimental results on performing rectification of faulty integer multiplier benchmarks using our approach. The multiplier benchmark circuits comprise three structural levels: the Partial Product Generator (PPG) stage, the Partial Product Accumulator (PPA) stage, and the Final Stage Adder (FSA) stage. We focus on two different multiplier structures, *sp-ar-rc* and *sp-wt-cl*. The naming convention of the multipliers follows the structure: "PPG-PPA-FSA", e.g., a *sp-ar-rc* multiplier indicates simple partial product for PPG, array structure for PPA, and ripple-carry adder for FSA. Similarly, the notation *wt* indicates a Wallace-tree structure, and *cl* indicates a Carry-look-ahead adder. The circuits are taken from [3] and are mapped using the *abc* tool with a library consisting of AND-XOR-OR-INV gates. We introduce gate and wiring faults in the netlists which affect multiple POs. The faults are placed at various topological levels across the multiplier stages; some faults are placed inside the PPG stage near the PIs, some are placed in the middle of the circuit in the PPA stage, and some inside the FSA stage near the POs. We select between two to five targets m, one near each bug location in the benchmark.

Our approach is implemented as a custom software using the programming language Python. The software utilizes the binary *revsca* [6], and the open-source libraries *amulet* [3], *Singular* [30], *abc*, and *sis*. The tools *amulet* and *revsca* use different approaches to generate the remainders rem_l. We run both tools to compare the performance of the two, and use the remainders from the first tool to complete. We utilize the symbolic computer algebra system *Singular* to perform the rectification check. *Singular* is also used to compute the reduced Gröbner basis for each rem_l and to construct a singleton polynomial rem_l' corresponding to each rem_l. Our custom software utilizes the polynomials rem_l' to compute rectification functions by performing all the algebraic

TABLE III: Rectification of faulty integer multipliers; Time in seconds, I = Row number, $n =$ Operand width, $m =$ Number of targets, $revsca$ = required time for remainder generation using [6], $amulet$ = required time for remainder generation using [3]. GBC = required time to perform rectification check and to compute the redGB for each rem_l using $Singular$, FC = required time to compute functions using the greedy approach and don't care based approach, PGC = synthesized patch sub-circuit using the greedy approach, PDC = synthesized patch sub-circuit using the DC-based approach, A = Area in terms of number of gates, D = longest topological delay, Time-Out (TO) = 10000s, NA = not applicable, OOM = out of memory

I	*n*	*m*	sp-ar-rc								sp-wt-cl							
			revsca	**amulet**	**GBC**	**FC**	**PGC**		**PDC**		***revsca***	**amulet**	**GBC**	**FC**	**PGC**		**PDC**	
							A	D	A	D					A	D	A	D
1	4	2	0.8	0.8	1.0	0.8	74	13	36	8	0.1	0.1	0.1	0.3	27	8	14	6
2	4	3	0.3	0.3	0.4	0.5	74	10	52	13	0.2	0.2	0.2	0.5	23	6	10	5
3	4	5	1.2	1.4	0.7	2.2	30	7	31	8	1	1	0.1	1.7	21	7	16	5
4	8	2	0.1	0.1	0.3	0.2	17	8	16	7	0.8	1.2	3.7	0.2	173	18	151	14
5	8	3	0.2	0.2	36	0.9	374	19	193	27	0.1	0.3	0.8	1.1	46	8	31	6
6	8	5	0.7	0.7	TO	NA	NA	NA	NA	NA	TO	TO	NA	NA	NA	NA	NA	NA
7	16	2	0.2	0.2	7696	1.3	823	17	114	17	0.4	0.8	1872	0.9	231	27	189	25
8	16	3	0.4	0.4	0.5	0.8	105	15	47	10	1.4	2.7	0.8	0.4	67	22	53	25
9	32	2	1.9	TO	0.2	0.4	53	9	39	8	4.6	6.5	193	0.6	92	16	83	17
10	32	3	3	3	3732	0.6	155	15	68	12	TO	TO	TO	NA	NA	NA	NA	NA
11	64	2	OOM	TO	NA	NA	NA	NA	NA	NA	7.3	TO	1940	1.2	137	22	94	19

computations described in Sec. V. The software uses the sis scripts *kernel extraction* and *full simplify* to optimize the rectification function using the DC-sets computed in the DC-based approach. Next, all the computed functions are optimized using abc. The optimized functions are then mapped using a library of AND-XOR gates and the synthesis results for $Area$ and $Delay$ are extracted. The experiments are conducted on a desktop computer with a 3.5GHz Intel CoreTM i7-4770K Quad-core CPU, 16 GB RAM, running 64-bit Linux OS.

Table III presents the results on performing rectification of faulty circuits for multiplier structures *sp-ar-rc* and *sp-wt-cl*. The columns denote the datapath size of the faulty benchmarks, the number of selected targets, the execution time for different stages of the approach, and the area and delay of the resulting patch sub-circuits after *abc* and *sis* optimization. As seen in these results, the main bottleneck of our approach is the execution time required to compute reduced Gröbner bases for each rem_l. This computation times out when performed on large remainders. The size of the remainders depend primarily on the factors: i) the number of bugs; ii) the number of targets; iii) the location of the bugs; iv) the location of the targets; and v) the size of the benchmark.

For example, consider the results for the structure *sp-ar-rc*, at row six of the table, the GBC execution times out while computing a reduced Gröbner basis due to large remainder sizes. However, the GBC computation completes relatively quickly for the same benchmark with fewer targets and fewer bugs at different locations as shown at row five of the table. The seventh row shows an example where the GBC computation completed after significant execution time due to large remainder sizes. Row eleven shows an example where neither $revsca$ nor $amulet$ were able to generate the remainders rem_l.

TABLE IV: Comparison of Area and Delay between original benchmark and patch-integrated rectified benchmarks. n = Datapath size, m = Number of targets, ORIG = Original faulty benchmark, Greedy = Benchmark rectified using patch from our greedy approach, DC-based = Benchmark rectified using patch from our DC-based approach, A = Area in terms of number of gates $\times 10^3$, D = Longest delay

n	*m*	sp-ar-rc						sp-wt-cl					
		ORIG		Greedy		DC-based		ORIG		Greedy		DC-based	
		A	*D*	*A*	*D*	*A*	*D*	*A*	*D*	*A*	*D*	*A*	*D*
4	2	127	31	126	20	119	20	130	22	116	20	96	20
4	3	134	31	130	20	141	24	137	22	96	20	96	20
4	5	148	35	97	19	90	21	151	25	98	22	99	22
8	2	647	67	534	62	538	55	755	32	731	29	714	30
8	3	654	70	815	64	674	72	762	37	723	34	698	36
16	2	1935	146	1908	135	1872	137	3831	50	3778	49	3753	51
32	3	12510	306	10090	270	10128	274	17501	84	17445	78	17312	79

Columns PGC and PDC denote the post-optimization synthesis results for the greedy approach and the DC-based approach, respectively. The synthesis results computed in PDC contain a smaller area and delay than the ones computed in PGC. This shows the efficacy of the DC-based approach in utilizing DC points for the logic optimization of the rectification functions. For these small patch sub-circuits, the difference in execution time between the greedy and DC-based approaches is negligible. Table IV compares the area and delay between the original benchmarks and the patch-integrated benchmarks for the relevant cases. As shown in the results, the area and delay of the patched benchmarks is comparable to the original faulty benchmark after resynthesis (using *abc*).

VIII. Conclusion

This paper presents an automated approach for the multi-fix rectification of integer arithmetic circuits using computer algebra techniques. Given a set of targets, we formulate a rectifiability check to ascertain the existence of rectification functions and present two approaches to compute them. One approach utilizes a greedy heuristic for quick patch computation, while the other finds a subset of DC points to better optimize the rectification functions. These approaches may result in polynomials with coefficients in the field of rationals. We present novel techniques to synthesize sub-circuit patches from these polynomials and demonstrate them on preliminary experimental results from array integer multiplier benchmarks.

Our approach requires computing reduced Gröbner bases, which is the main bottleneck in our experiments. As part of our future research, we are investigating alternatives to this expensive computation. We are also researching methods to compute the rectification functions in terms of internal nets of the circuit.

References

[1] M. Ciesielski, T. Su, A. Yasin, and C. Yu, "Understanding algebraic rewriting for arithmetic circuit verification: A bit-flow model," *IEEE TCAD of Integrated Circuits and Systems*, 2020.

[2] A. Mahzoon, D. Große, C. Scholl, and R. Drechsler, "Towards formal verification of optimized and industrial multipliers," in *DATE*, 2020.

[3] D. Kaufmann and A. Biere, "Amulet2.0 for verifying multiplier circuits," in *TACAS*, 2021.

[4] M. Temel, A. Slobodova, and W. A. Hunt, "Automated and scalable verification of integer multipliers," in *Computer Aided Verification*, 2020.

[5] D. Kaufmann, A. Biere, and M. Kauers, "Verifying large multipliers by combining sat and computer algebra," in *FMCAD*, 2019.

[6] A. Mahzoon, D. Große, and R. Drechsler, "Revsca: Using reverse engineering to bring light into backward rewriting for big and dirty multipliers," in *Design Automation Conference*, 2019.

[7] D. Kaufmann, "Formal verification of multiplier circuits using computer algebra," Ph.D. dissertation, Johannes Kepler University Linz, 2020.

[8] 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.

[9] 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.

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

[11] K. F. Tang, P. K. Huang, C. N. Chou, and C. Y. Huang, "Multi-patch Generation for Multi-error Logic Rectification by Interpolation with Cofactor Reduction," in *DATE*, 2012, pp. 1567–1572.

[12] H. T. Zhang and J. H. R. Jiang, "Cost-aware Patch Generation for Multi-target Function Rectification of Engineering Change Orders," in *Design Automation Conference (DAC)*, 2018, pp. 1–6.

[13] Y. Kimura, A. M. Gharehbaghi, and M. Fujita, "Signal Selection Methods for Efficient Multi-Target Correction," in *ISCAS*, 2019, pp. 1–5.

[14] V. N. Kravets, N. Lee, and J. R. Jiang, "Comprehensive Search for ECO Rectification Using Symbolic Sampling," in *DAC*, 2019, pp. 1–6.

[15] 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 *VLSI-SoC*, Oct 2018, pp. 49–54.

[16] V. Rao, U. Gupta, A. Srinath, I. Ilioaea, P. Kalla, and F. Enescu, "Post-Verification Debugging and Rectification of Finite Field Arithmetic Circuits using Computer Algebra Techniques," in *FMCAD*, 2018.

[17] A. Q. Dao, N.-Z. Lee, L.-C. Chen, M. P.-H. Lin, J.-H. R. Jiang, A. Mishchenko, and R. Brayton, "Efficient Computation of ECO Patch Functions," in *DAC*, 2018, pp. 51:1–51:6.

[18] 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.

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

[20] K. Gitina, S. Reimer, M. Sauer, R. Wimmer, C. Scholl, and B. Becker, "Equivalence Checking of Partial Designs Using Dependency Quantified Boolean Formulae," in *ICCD*, 2013.

[21] U. Gupta, P. Kalla, I. Ilioaea, and F. Enescu, "Exploring Algebraic Interpolants for Rectification of Finite Field Arithmetic Circuits with Groebner Bases," in *ETS*, May 2019, pp. 1–6.

[22] V. Rao, I. Ilioaea, H. Ondricek, P. Kalla, and F. Enescu, "Word-level multi-fix rectifiability of finite field arithmetic circuits," in *ISQED*, 2021.

[23] F. Farahmandi and P. Mishra, "Automated Debugging of Arithmetic Circuits Using Incremental Gröbner Basis Reduction," in *ICCD*, 2017.

[24] A. Mahzoon, D. Große, and R. Drechsler, "Combining Symbolic Computer Algebra and Boolean Satisfiability for Automatic Debugging and Fixing of Complex Multipliers," in *ISVLSI*, 2018, pp. 351–356.

[25] N. A. Sabbagh and B. Alizadeh, "Arithmetic Circuit Correction by Adding Optimized Correctors Based on Groebner Basis Computation," in *Proc. Eur. Test Symp. (ETS)*, 2021, pp. 1–6.

[26] D. Kaufmann, A. Biere, and M. Kauers, "Incremental column-wise verification of arithmetic circuits using computer algebra," *Formal Methods in System Design*, Feb 2019.

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

[28] J. von Neumann, "On regular rings," *Proceedings of the National Academy of Sciences*, vol. 22, no. 12, pp. 707–713, 1936.

[29] K. R. Goodearl., "Von neumann regular rings," *Monographs and Studies in Mathematics*, vol. 4, p. 369, 1979.

[30] 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.