An Algebraic Approach to Partial Synthesis of Arithmetic Circuits

Bhavani Sampathkumar[1], Ritaja Das[1], Bailey Martin[1], Florian Enescu[2] and Priyank Kalla[1]
[1]University of Utah, Salt Lake City, UT, USA
[2]Mathematics & Statistics, Georgia State University, Atlanta, GA, USA

ABSTRACT

We present an approach to partial logic synthesis of arithmetic circuits. Its targeted applications are rectification of buggy circuits, and computing care and don't care sets at internal nets of the circuit. The approach models the circuit by way of polynomial ideals in rings with coefficients in the field of rationals ($\mathbb{Q}$). Techniques from commutative algebra are applied to compute internal patch functions as polynomials over $\mathbb{Q}$. We describe how the care set and the don't care conditions manifest in the algebraic setting, and show how to generate corresponding Boolean functions from polynomials over $\mathbb{Q}$. Experiments are conducted over various integer multiplier architectures which demonstrate the efficacy of our approach, where SAT/interpolation based techniques are infeasible.

ACM Reference Format:
Bhavani Sampathkumar[1], Ritaja Das[1], Bailey Martin[1], Florian Enescu[2] and Priyank Kalla[1], [1]University of Utah, Salt Lake City, UT, USA, [2]Mathematics & Statistics, Georgia State University, Atlanta, GA, USA . 2025. An Algebraic Approach to Partial Synthesis of Arithmetic Circuits. In *30th Asia and South Pacific Design Automation Conference (ASPDAC '25), January 20–23, 2025, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3658617.3697724

1 INTRODUCTION

Partial Logic Synthesis (PLS) is a fundamental problem that finds application in many areas of computer-aided design (CAD), verification, test and debug of logic circuits. In PLS, the design's specification (*Spec*) is given — usually available as a Boolean function, a golden circuit model, or in the case of arithmetic circuits, the *Spec* might be given as a polynomial function over a multivariate polynomial ring. A *partial implementation* (*Impl*) of the circuit is also given as an interconnection of logic gates, along with some unfinished components given as *black-boxes*. Fig. 1 depicts a typical PLS setup, where a *miter* $\mathcal{M}$ between *Spec* and *Impl* is constructed. The primary inputs (X_{PI}) of *Spec* and *Impl* are connected together, and the implementation contains a block-box which may implement some unknown or unspecified function U at a *target net* x_i.

Then, *PLS* requires to solve the following problems: 1) To check if there exists a patch function U which can complete the partial implementation at target net x_i. If one exists, then subsequently compute the function U. This model can be applied to rectify buggy circuits at internal nets [7, 9, 22], or for rectification under engineering change orders (ECO) [16, 30]. 2) The function U may not be unique. There may be more than one admissible functions for the black-box. This corresponds to the *observability don't care (ODC)* set for the net x_i. The PLS model can be used to compute both the care-set and the ODC set for U, so that logic optimization can be performed to synthesize a patch function.

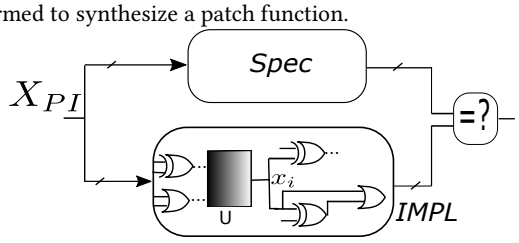

Figure 1: A PLS framework: Miter $\mathcal{M}$ between *Spec* and *Impl*, with unfinished component at target(s) x_i.

This paper describes an algebraic approach to the PLS problem *targeted to integer arithmetic circuits*. We demonstrate the application of our approach to two problems: 1) rectification of buggy integer arithmetic (multiplier) circuits, and 2) to compute care sets and ODC sets at a few heuristically selected internal target nets to perform logic optimization for various multiplier circuits.

The rectification problem setup and Objective: We are given a combinational integer arithmetic circuit C as an *Impl*. Also given is a multivariate polynomial (f_{spec}) as the *Spec*, where f_{spec} has integral coefficients and the variables take only Boolean values. We assume that equivalence checking has been performed between *Impl* and *Spec* (e.g., using the techniques of [2, 11, 20, 25]), and it is determined that *Impl does not match the Spec*, i.e. there are bugs in the design. We now debug the circuit and determine if the *Impl* C can be rectified at a *single* target net x_i, i.e. we solve single-fix rectification. If so, we compute a Boolean function U such that patching the circuit at x_i (with $x_i = U$) rectifies C. Moreover, we compute both the on-set U_{on} (or the off-set U_{off}) and the ODC set U_{dc} corresponding to the patch function for a simplified/optimal implementation as a logic sub-circuit.

For logic optimization: We are given the polynomial f_{spec} as above, along with a correct/bug-free gate-level implementation (*Impl*). Using a unate covering problem (UCP) formulation, we identify a set of target nets $\{x_i\}$ where we compute the ODC sets using our approach. These ODCs are then used to perform logic resynthesis to further optimize the given *Impl* circuit C.

Approach: We model the PLS problem using concepts from algebraic geometry and use symbolic computer algebra algorithms to compute rectification functions U_{on}, U_{dc}. We represent the *Impl* C using ideals in a multivariate polynomial ring. We identify a set of target nets $\{x_i\}$ in C and check if C can be patched at a target net.

ASP-DAC 2025, Jan. 20–23, 2025, Tokyo, Japan

ACM ISBN 979-8-4007-0635-6/25/01
https://doi.org/10.1145/3658617.3697724

For this, we use the Gröbner basis (GB) algorithm [1] as a decision procedure to check rectifiability of C at target x_i. Subsequently, we perform PLS by computing the patch function using the *extended* GB algorithm (cf. Sec 2.1 in [1]) as a quantification procedure. Thus, U_{on}, U_{dc} are computed as polynomials. Subsequently, we synthesize Boolean functions from such algebraically computed U_{on}, U_{dc}, and optimize the care-set w.r.t. the ODC set.

Rationale for the algebraic approach: Interpolation based PLS approaches rely on proving UNSAT of a miter and then computing a resolution proof. For arithmetic circuits, this is known to be infeasible [8] [24]. Research over the last decade [17] [25] [20] [8] [24] has amply demonstrated that *computer algebra* techniques are more appropriate to solve decision and quantification problems for datapath designs.

Contributions: We model the PLS problem for arithmetic circuits as ideal membership testing over *polynomial rings with coefficients in the field of rationals:* $\mathbb{Q}[x_1, \ldots, x_d]$. Here, the variables $x_1, \ldots, x_d$ take only binary values, and represent the nets of the circuit. The reason for modeling integer arithmetic circuits over $\mathbb{Q}$ (instead of over $\mathbb{Z}$) is that many algebraic geometry results are valid over fields (e.g. the *Nullstellensatz* [3]), and the set $\mathbb{Z}$ does not form a field. However, we leverage the fact that $\mathbb{Z} \subset \mathbb{Q}$, and compute polynomials U_{on}, U_{dc} using the extended GB algorithm in $\mathbb{Q}[x_1, \ldots, x_d]$. However, formulating PLS in $\mathbb{Q}[x_1, \ldots, x_d]$ poses a challenge: it results in U_{on}, U_{dc} as *polynomials with rational coefficients*, i.e. $U_{on}, U_{dc} \in \mathbb{Q}[x_1, \ldots, x_d]$. Thus, polynomials U may evaluate to rational (non-Boolean) values.

We describe the relationship of these rectification polynomials U_{on}, U_{dc} (with rational coefficients) to the desired Boolean patch functions. We show that the rectification polynomials and the Boolean functions should have the same zero-set (called a *variety*). Subsequently, we show how to construct a corresponding Boolean function $U^{\mathbb{B}}$ from the polynomial U, which preserves the variety of U and patches C. A logic synthesis tool is then used to optimize these Boolean functions w.r.t. their ODCs.

Experiments are performed over various integer multiplier architectures that rectify a buggy implementation. This rectification solves PLS and the corresponding ODC computation to generate an optimized patch function. As compared to contemporary interpolation based techniques, our approach is orders of magnitude faster, and also produces more optimized patch functions in terms of area and delay of the implementation. Finally, we address only the single-target PLS problem in the paper. Multi target PLS (with multiple black-boxes) is beyond the scope of this paper.

Paper Organization: The next section covers the notation and preliminary concepts. Section 3 reviews previous work in PLS and its limitations. Section 4 depicts the algebraic modeling of the PLS problem. Section 5 describes our approach to compute rectification functions and ODCs. Experiments are described in Section 6, and Section 7 concludes the paper.

2 PRELIMINARIES

Let $X = \{x_1, \ldots, x_d\}$ be the set of d variables and $R = \mathbb{Q}[X] = \mathbb{Q}[x_1, \ldots, x_d]$ be the multivariate polynomial ring with coefficients from $\mathbb{Q}$. We use $\neg, \wedge, \vee, \oplus$ to denote Boolean NOT, AND, OR, XOR operations and $+, \cdot$ for addition and multiplication in ring R, respectively. A monomial is a power product of the form $X_n = x_1^{e_1} \cdot x_2^{e_2} \cdots x_n^{e_n}$, where $e_i \in \mathbb{Z}_{\geq 0}, i \in \{1, \ldots, n\}$. A *polynomial* $f \in R$ is written as a finite sum of terms $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$, where $c_1, \ldots, c_t$ are coefficients and $X_1, \ldots, X_t$ are monomials. Impose a monomial order $>$ (a term order) on the ring. Then the monomials of all polynomials $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$ are ordered w.r.t. to $>$, such that $X_1 > X_2 > \cdots > X_t$. Subject to $>$, $lt(f) = c_1X_1$, $lm(f) = X_1$, $lc(f) = c_1$, are the *leading term*, *leading monomial* and *leading coefficient* of f, respectively. In this work, we consider mostly lexicographic (lex) term orders.

Ideals and Varieties: Given a set of polynomials $F = \{f_1, \ldots, f_s\}$ in R, the *ideal* $J \subseteq R$ generated by them is: $J = \langle f_1, \ldots, f_s \rangle = \{h_1f_1 + h_2f_2 + \cdots + h_sf_s : h_1, \ldots, h_s \in R\}$. The polynomials $f_1, \ldots, f_s$ form the *basis* or the *generators* of J.

In our work, the set X of variables corresponds to the nets of the given circuit C, and we represent the functionality of C using a set of polynomials $F = \{f_1, \ldots, f_s\}$ in $\mathbb{Q}[x_1, \ldots, x_d]$, which generates ideal $J = \langle F \rangle$. Given a specification polynomial $f_{spec} \in R$, $f_{spec} \xrightarrow{F}_+ r$ denotes the *reduction* of f_{spec} (mod F) resulting in the remainder r. This reduction is implemented using multivariate division (cf. Algorithm 1.5.1 [1]).

Let $\boldsymbol{a} = (a_1, \ldots, a_d) \in \mathbb{Q}^d$ be a point, 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. This zero set is called the **variety**. A variety depends not just on the given set of polynomials F, but rather on the ideal J generated by F. We denote the variety of J over $\mathbb{Q}$ as $V(J)$, where: $V(F) = V(J) = \{\boldsymbol{a} \in \mathbb{Q}^d : \forall p \in J, p(\boldsymbol{a}) = 0\}$.

Boolean Functions and Varieties: Boolean functions comprise a finite set of points (minterms), and every finite set of points is also a variety $V(J)$ of some ideal J. We construe the rectification patch functions and ODC-sets as varieties $V(J)$ and compute them in terms of the ideals J.

Given two ideals $J_1 = \langle f_1, \ldots, f_s \rangle, J_2 = \langle h_1, \ldots, h_r \rangle$, the sum $J_1 + J_2 = \langle f_1, \ldots, f_s, h_1 \ldots, h_r \rangle$, and $V(J_1 + J_2) = V(J_1) \cap V(J_2)$.

Gröbner Basis: An ideal may have many different sets of generators: $J = \langle f_1, \ldots, f_s \rangle = \cdots = \langle g_1, \ldots, g_t \rangle$. Given a non-zero ideal J, a *Gröbner basis* (GB) for J is a finite set of polynomials $G = \{g_1, \ldots, g_t\}$ satisfying $\langle \{lm(f) \mid f \in J\} \rangle = \langle lm(g_1), \ldots, lm(g_t) \rangle$. Then $J = \langle G \rangle$ holds and so $G = GB(J)$ forms a basis for J. The famous Buchberger's algorithm (see Alg. 1.7.1 in [1]) takes as input the set of polynomials $F = \{f_1, \ldots, f_s\}$ and computes the GB $G = \{g_1, \ldots, g_t\}$. A GB can be *reduced* to eliminate redundant polynomials from the basis. A reduced GB is a canonical representation of the ideal. Buchberger's algorithm can be easily extended to compute not just the Gröbner basis $G = \{g_1, \ldots, g_t\}$ but also a $t \times s$ matrix M with polynomial entries such that:

$$\begin{bmatrix} g_1 \\ \vdots \\ g_t \end{bmatrix} = M \cdot \begin{bmatrix} f_1 \\ \vdots \\ f_s \end{bmatrix}. \tag{1}$$

A GB G allows to solve the **ideal membership** problem which we make use of in this work: a polynomial f is a member of ideal J if and only if $f \xrightarrow{G=GB(J)}_+ 0$. When $f \in J$, f can be written as a linear combination (with polynomial coefficients) of the elements of the Gröbner basis:

$$f = u_1g_1 + u_2g_2 + \cdots + u_tg_t, \tag{2}$$

where u_i correspond to the quotients of division $f \xrightarrow{g_1,\ldots,g_t}_+ 0$. Subsequently, Eqns. (2) and (1) can be combined to give f as combination of the original polynomials $f_1, \ldots, f_s$:

$$f = v_1 f_1 + \cdots + v_s f_s. \tag{3}$$

Boolean idempotency and the bit-level vanishing polynomials: All the variables representing the nets of a circuit only take binary values, and thus satisfy the polynomial constraint $x^2 - x = 0$, i.e. $x^2 - x$ vanishes on $\{0, 1\}$. Let $F_0 = \{x_l^2 - x_l : \forall x_l \in X\}$ denote the set of all bit-level vanishing polynomials, and J_0 be the generated ideal s.t. $J_0 = \langle F_0 \rangle$. To enforce Boolean idempotency ($x \wedge x = x$), we include the ideal J_0 in our computations. In other words, the circuit is modeled as an ideal $J + J_0 = \langle F \cup F_0 \rangle$, where F is a set of polynomials derived from the logic gates of C, and F_0 is the set of bit-level vanishing polynomials.

3 LIMITATIONS OF PREVIOUS WORK

In the area of debugging and rectification, the early works of [18] [15] [16] [28] were reformulated using CI [30] [31] [29]. While successful for control-dominated applications (random logic circuits), these have not been effective in solving PLS for arithmetic datapaths. This is despite the recent developments that improve implementation costs [32] [10], or use symbolic sampling [14], etc.

Computer algebra techniques have been employed for formal verification (FV) of arithmetic circuits [11, 13, 17, 20, 25], as well as for their debugging and rectification [8, 9, 22–24]. The FV techniques use a GB-reduction (GBR) to check if the *Spec* polynomial reduces to 0 when divided by the polynomials of the circuit: $f_{spec} \xrightarrow{f_1,\ldots,f_s}_+ 0$. Efficient GBR engines were developed in [11, 20] to operate on multiplier circuits. We also make use of these GBR engines [11, 20] for computations in our work.

The works [8, 9, 22, 23] perform rectification of *finite field* arithmetic circuits, but are inapplicable to integer arithmetic circuits. Other attempts have been made to rectify buggy arithmetic circuits [19] [26] [6] [4], but these techniques are either incomplete [21] or do not address the PLS problem at internal nets.

The paper [24] attempts to rectify integer arithmetic circuits over the same polynomial ring $R = \mathbb{Q}[X]$. However, we have found an error in their approach. To compute a Boolean patch function, their approach relies on a GB computation for the ideal $\langle f, x_i^2 - x_i \rangle$ using any term order. In Prop. VI.2 in [24], they claim that the reduced GB $G = GB(f, x_i^2 - x_i)$ has polynomials with coefficients only as ± 1, and use this property for synthesis. However, this is incorrect, and here is a counter-example:

Example 3.1. Let $I = \langle x^2 - x, y^2 - y, z^2 - z, u^2 - u, (x + 2yz - y - z) \cdot (1 - u) + (x - 2yz + y + z - 1) \cdot u \rangle$ be an ideal in $\mathbb{Q}[x, y, z, u]$. Then with the degree-lexicographic order with $x > y > z > u$, the reduced Gröbner basis has rational coefficients:

$$x^2 - x, y^2 - y, z^2 - z, u^2 - u,$$
$$xu - yz - \frac{1}{2}x + \frac{1}{2}y + \frac{1}{2}z - \frac{1}{2}u,\ xz - yu - \frac{1}{2}x + \frac{1}{2}y - \frac{1}{2}z + \frac{1}{2}u,$$
$$xy - zu - \frac{1}{2}x - \frac{1}{2}y + \frac{1}{2}z + \frac{1}{2}u,\ yzu - \frac{1}{2}yz - \frac{1}{2}yu - \frac{1}{2}zu - \frac{1}{4}x + \frac{1}{4}y + \frac{1}{4}z + \frac{1}{4}u.$$

With rational coefficients in the Gröbner basis, [24] cannot solve the rectification problem. Our approach shows how to compute rectification polynomials $U_{on}, U_{dc} \in \mathbb{Q}[X]$. We also show that from these rational functions, the respective Boolean functions $U_{on}^{\mathbb{B}}, U_{dc}^{\mathbb{B}}$ can be obtained as follows: i) at those points (inputs) wherever the rational polynomials U_{on}, U_{dc} evaluate to 0, the corresponding Boolean functions $U_{on}^{\mathbb{B}}, U_{dc}^{\mathbb{B}}$ should also evaluate to 0; and ii) wherever the polynomials evaluate to non-zero (rational) values, the Boolean functions should evaluate to 1. In a recent work [27], the authors solve exactly the same theoretical problem as above. They recursively expand the polynomial U w.r.t. variable x_i at each recursion level using a positive-Davio decomposition. At terminal cases, they return a 0 or a 1 as desired, and generate the corresponding Boolean function $U^{\mathbb{B}}$ as an AND-XOR expression. We use the same technique of [27] and implement it using an OKFDD package to generate Boolean functions for the care ($U_{on}^{\mathbb{B}}$) and ODC ($U_{dc}^{\mathbb{B}}$) sets from corresponding rational polynomials for synthesis.

4 PROBLEM MODELING

We use the circuit C of Fig. 2 as a running example to demonstrate our approach. The *Spec* polynomial for C is given as

$$f_{spec} : z_0 + 2z_1 - 2a_0a_1b_0b_1 + 4a_0a_1b_1 - a_0b_0 - 2a_0b_1 + 4a_1b_0b_1 - 2a_1b_0 - 3a_1b_1, \tag{4}$$

with binary variables and integral coefficients.

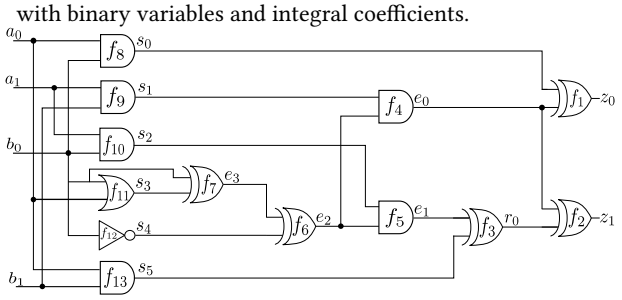

Figure 2: The circuit C doesn't match f_{spec}, to be rectified.

The logic gates of C (Boolean operators) can be modeled as polynomials over $\mathbb{Q}$ as follows [25]:

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

Using Eqns. (5), the gates of the circuit of Fig. 2 are represented by polynomials $F = \langle f_1, \ldots, f_{13} \rangle$ in R:

$$\begin{aligned} &f_1 : z_0 - (s_0 + e_0 - 2s_0e_0); && f_2 : z_1 - (e_0 + r_0 - 2 \cdot e_0 \cdot r_0); \\ &f_3 : r_0 - (e_1 + s_5 - 2e_1s_5); && f_4 : e_0 - (s_1 \cdot e_2); \\ &f_5 : e_1 - (s_2 \cdot e_2); && f_6 : e_2 - (e_3 + s_4 - 2e_3s_4); \\ &f_7 : e_3 - (b_0 + s_3 - 2b_0s_3); && f_8 : s_0 - (a_0 \cdot b_0); \\ &f_9 : s_1 - (b_1 \cdot a_1); && f_{10} : s_2 - (a_1 \cdot b_0); \\ &f_{11} : s_3 - (a_0 + b_0 - a_0b_0); && f_{12} : s_4 - (1 - b_0); \\ &f_{13} : s_5 - (a_0b_1); \end{aligned} \tag{6}$$

Then, ideal $J = \langle f_1, \ldots, f_{13} \rangle$. Generate the set of bit-level vanishing polynomials for all primary inputs $F_0^{PI} = \{x^2 - x, \forall x \in X_{PI}\}$, such that $J_0 = \langle F_0^{PI} \rangle$. Then ideal $J + J_0 = \langle F \cup F_0^{PI} \rangle$ models the functionality of C. Based on the work of [17, 20, 25], the verification problem can now be formulated as checking whether or not f_{spec} is

a member of the ideal $J + J_0$. This check is performed using Gröbner basis reduction: $f_{spec} \equiv C \iff f_{spec} \xrightarrow{GB(J+J_0)}_+ 0$.

Instead of imposing arbitrary term orders for the symbolic computation, the work of [17, 25] further showed that a specialized term order, called the *Reverse Topological Term Order (RTTO)* >, can be derived from the topology of the circuit. RTTO > is a lex term order with the circuit's variables ordered reverse topologically from outputs to inputs. RTTO > ensures that the polynomials in $F \cup F_0^{PI}$ themselves form a *minimal Gröbner basis of* $J + J_0$ [12, 17]. Thus, a GB computation is avoided, and ideal membership (verification) is performed just by dividing $f_{spec} \xrightarrow{F \cup F_0^{PI}}_+ 0$. We also use RTTO > based term orders in this work.

Example 4.1. For the circuit of Fig. 2, the polynomials $\{f_1, \ldots, f_{13}\}$ in Eqn. (6) are already described in RTTO >: lex order with $\{z_0 > z_1\} > \{r_0\} > \{e_0 > e_1\} > \{e_2\} > \{e_3\} > \{s_0 > s_1 > s_2 > s_3 > s_4 > s_5\} > \{a_0 > a_1 > b_0 > b_1\}$. Hence, $F = \{f_1, \ldots, f_{13}\} \cup F_0^{PI} = \{a_0^2 - a_0, \ldots, b_1^2 - b_1\}$ forms a minimal Gröbner basis of $J + J_0$. When verification is performed, we obtain $f_{spec} \xrightarrow{F \cup F_0^{PI}}_+ r = a_0a_1b_0b_1 + a_0a_1b_1 + a_1b_0b_1 - 2a_1b_0$. Since $r \neq 0$, C does not match f_{spec}; C is buggy and it needs to be rectified.

4.0.1 Rectification Target Selection. We now select target nets where we attempt single-fix rectification. When GBR produces a non-zero remainder r, then the assignments to the PIs in r that make $r \neq 0$ excite the bug in the design. We simulate C using those vectors v that make $r(v) \neq 0$, and find those POs where the bug is observable. Then we identify those nets that lie in the intersection of the fanin cones of all buggy outputs. These nets comprise the subset $\mathcal{N} \subseteq X$ as potential rectification targets. In Ex. 4.1 and Fig. 2, the bug is observable at both outputs z_0, z_1. Then $\mathcal{N} = \{s_4, s_3, s_1, e_3, e_2, e_0\}$ as potential rectifiable locations, as both outputs are reachable from each net in $\mathcal{N}$.

4.0.2 Rectification Check. The circuit may or maynot be rectifiable at all the nets in $\mathcal{N}$. Therefore, it is required to ascertain whether C indeed admits rectification at any target $x_i \in \mathcal{N}$. For this **rectification check**, an algebraic approach was presented in Thm. IV.1 in [24], which we utilize in our work.

Single-fix-rectification at target x_i means that there exists a polynomial function $x_i = U$ which patches C where $U : \{0,1\}^{|X_{PI}|} \rightarrow \{0,1\}$. For rectification, the set F is updated to $F = \{f_1, \ldots, f_{i-1}, f_i = x_i - U, f_{i+1}, \ldots, f_s\}$. The following theorem checks if such a patch function U exists.

Theorem 4.1 (Rectification Theorem [24])**.** Given f_{spec}, *Impl* C, derive RTTO > to represent the polynomials in F. Using RTTO >, construct two ideals:

- $J_L = \langle F_L \rangle, F_L = \{f_1, \ldots, f_{i-1},\ f_i = x_i - 1,\ f_{i+1}, \ldots, f_s\}$;
- $J_H = \langle F_H \rangle, F_H = \{f_1, \ldots, f_{i-1},\ f_i = x_i - 0,\ f_{i+1}, \ldots, f_s\}$;

where the polynomials $f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_s$ are the same as in the generators of ideal J (representing the circuit), and f_i is replaced with $f_i = x_i - 1$ in J_L and $f_i = x_i - 0$ in J_H, respectively. Perform the reductions: $f_{spec} \xrightarrow{F_L, F_0^{PI}}_+ r_L$ and $f_{spec} \xrightarrow{F_H, F_0^{PI}}_+ r_H$. Then C admits single-fix rectification at x_i **if and only if** $r_L \cdot r_H \xrightarrow{J_0}_+ 0$.

Example 4.2. We demonstrate the application of Thm. 4.1 on the nets e_3, and s_1 in Fig. 2. To check for rectifiability at e_3, replace f_7 (the function at net e_3) in Eqn. (6) with $f_7 : e_3 - 1$ in the ideal J_L and $f_7 : e_3 - 0$ in J_H. Reduction gives $f_{spec} \xrightarrow{J_L + J_0}_+ r_L = -4a_0a_1b_0b_1 + 4a_0a_1b_1 + 3a_1b_0b_1 - 3a_1b_1$, and $f_{spec} \xrightarrow{J_H + J_0}_+ r_H = 2a_0a_1b_0b_1 + a_1b_0b_1 - 2a_1b_0$. Then, we see that $r_L \cdot r_H \xrightarrow{J_0}_+ 0$, i.e. the circuit is rectifiable at e_3. However, the rectification test fails at net s_1. When polynomial f_9 at net s_1 is modified, we obtain $r_L \cdot r_H \xrightarrow{J_0}_+ -a_0a_1b_0b_1 + a_0a_1b_1 - 3a_1b_0b_1 + 4a_1b_0 \neq 0$, implying that C cannot be patched at s_1 to match f_{spec}.

5 COMPUTING RECTIFICATION FUNCTIONS

After ascertaining that C admits single-fix rectification at x_i, we demonstrate how a rectification function $x_i = U$ can be computed to patch the circuit. Let $F = \{f_1, \ldots, f_i : x_i - U, \ldots, f_s\}$ be the set of polynomials modeling the correct (patched) circuit. Since U patches the circuit, it must satisfy the ideal membership condition that $f_{spec} \in J + J_0^{X_{PI}} = \langle f_1, \ldots, f_i : x_i - U, \ldots, f_s,\ x_l^2 - x_l : x_l \in X_{PI} \rangle$. This polynomial function U can be computed using the combination of extended Gröbner basis and ideal membership testing. The ideal membership relation of f_{spec} can be written as:

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

where $h_1, \ldots, h_s, H_l$ are polynomials in $R = \mathbb{Q}[X]$. Substituting $f_i = x_i - U$ in the above equation:

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

$$\begin{aligned} f_{spec} - h_1 f_1 - h_2 f_2 - \cdots - h_{i-1} f_{i-1} - \boldsymbol{h_i x_i} \\ = \ -\boldsymbol{h_i U} + h_{i+1} f_{i+1} \cdots + h_s f_s + \sum_{x_l \in X_{PI}} H_l \cdot (x_l^2 - x_l) \end{aligned} \tag{9}$$

Notice that on the L.H.S. of Eqn. (9), the polynomials f_{spec}, $f_1, \ldots, f_{i-1}$, and the monomial x_i are known expressions. Therefore, f_{spec} can be divided by $f_1, \ldots, f_{i-1}$, and by x_i to obtain the respective quotients $(h_1, \ldots, h_i)$ of the division, and the remainder r, where $r = f_{spec} - h_1 f_1 - h_2 f_2 - \cdots - h_i x_i$. After h_i is computed (as the quotient of this division by x_i), the R.H.S. of Eqn. (9) consists of $h_i, f_{i+1}, \ldots, f_s$ and all the bit-level vanishing polynomials $x_l^2 - x_l, x_l \in X_{PI}$, as known expressions. This implies that:

$$f_{spec} - h_1 f_1 - \cdots - h_i x_i \ \in \langle h_i, f_{i+1}, \ldots, f_s,\ x_l^2 - x_l \rangle, \tag{10}$$

$$r \ \in \langle h_i, f_{i+1}, \ldots, f_s,\ x_l^2 - x_l \rangle, \tag{11}$$

where, $x_l \in X_{PI}$. This ideal membership further implies that r can be written as some polynomial combination of the generators $h_i, f_{i+1}, \ldots, f_s$, and $x_l^2 - x_l,\ \forall x_l \in X_{PI}$. This combination can be identified by first computing the Gröbner basis G of the ideal $\langle h_i, f_{i+1}, \ldots, f_s, x_l^2 - x_l \rangle$, and then performing the ideal membership test $r \xrightarrow{G}_+ 0$, while utilizing Eqns. (2) and (3). As a result, we

can write the following ideal membership relation:

$$r = h'_i h_i + h'_{i+1} f_{i+1} + \cdots + h'_s f_s + \sum_{x_l \in X_{PI}} H_l(x_l^2 - x_l) \qquad (12)$$

Then $U = h'_i$ is a rectification polynomial that forms the (mathematical) solution to the PLS (black-box) problem. However, in our model, h'_i in Eqn. (12) is computed using extended Gröbner basis over $\mathbb{Q}[X]$. This polynomial h'_i *may have fractional coefficients* and it may also evaluate to non-binary values in $\mathbb{Q}$ for some inputs.

Example 5.1. For the circuit C of Fig. 2, we compute a rectification polynomial at e_3 using the approach of Eqns. (7)-(12). The polynomial h_i of Eqn. (9) is computed to be $h_i = -6a_0a_1b_0b_1 + 4a_0a_1b_1 + 2a_1b_0b_1 + 2a_1b_0 - 3a_1b_1$. Similarly, the rectification polynomial $e_3 = h'_i$ from Eqn. (12) computes to $h'_i = 56/5a_0b_0b_1 - 56/5a_0b_0 - 56/5a_0b_1 + 56/5a_0 + b_0$. Clearly, the coefficients of h_i, h'_i are in $\mathbb{Q}$, and they evaluate to non-binary values for some assignments: e.g. $h'_i(a_0 = 1, b_0 = a_1 = b_1 = 0) = 56/5$.

What is remarkable is that if we use the computed h'_i as the "patch polynomial" at e_3, and replace the polynomial f_7 in Eqn. (6) with the patch $f_7 : e_3 - h'_i$, then the ideal membership test (verification test) passes, i.e. $f_{spec} \xrightarrow{f_1,\ldots,e_3-h'_i,\ldots f_s, J_0^{PI}}_{+} 0$. In other words, *according to our polynomial abstraction, the circuit is rectified. However, we have to resolve the non-binary evaluation of h'_i, and devise techniques to synthesize Boolean rectification functions.*

5.0.1 Rectification Polynomials and ODCs. Let us consider the varieties (zeros) of h_i and h'_i. In Eqn. (12), for any point in the input space (input vector) $\boldsymbol{a} \in \{0,1\}^{|X_{PI}|}$, when $h_i(\boldsymbol{a}) = 0$, then h'_i may evaluate to *any value* in $\mathbb{Q}$, *and yet satisfy the ideal membership relation.* This implies the existence of don't care conditions.

Example 5.2. Continuing from Ex. 5.1, we evaluate the polynomials h_i and h'_i for all primary input patterns. The evaluations are recorded in Table 1. At points where $h_i = 0$, we see that h'_i may evaluate to rational values. At other points where $h_i \neq 0$, we observe that h'_i always evaluates to a value in $\{0,1\}$ (Boolean value).

Lemma 5.1. Let the polynomials h_i and h'_i be computed in terms of variables in the set X_j, where $X_j < x_i$ in RTTO. Let there exist a polynomial U that maps from $\{0,1\}^{|X_j|} \rightarrow \{0,1\}$. Let $p \in \{0,1\}^{|X_{PI}|}$ be an assignment to the primary inputs of C. 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 5.1. Using Eqns. (9)-(12), we obain:

$$h_i(U - h'_i) = \sum_{j=i+1}^{s} ((h_j - h'_j)f_j) + \sum_{x_l \in X_{PI}} (H_l - H'_l)(x_l^2 - x_l) \qquad (13)$$

Consider Eqn. (13) and a point p' such that $p'|_{X_{PI}} = p$ 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. (13), when evaluated at p', is 0. Then, we have $h_i(p')(U(p') - h'_i(p')) = 0$. In L.H.S. of Eqn. (13), when $h_i(p') \neq 0, U(p') = h'_i(p')$.

Since we know from Thm.4.1 that there exists a (Boolean) rectification function U for C at x_i, we know that $U(p')$ evaluates to a value in $\{0,1\}$ where $h_i(p') \neq 0$. Thus our approach provides a method to compute not only the ODC-set, but also the off-set and on-set (care-set) of the patch function:

- Using Eqns. (7)-(12), compute polynomials h_i, h'_i.
- Observability don't care set U_{dc} of the patch function at x_i is the set of points in the variety $V(h_i, J_0)$, i.e. those points $\{\boldsymbol{a}\}$ where $h_i(\boldsymbol{a}) = 0$.
- The points $\{\boldsymbol{a}\}$ where $h_i(\boldsymbol{a}) \neq 0$ are the points which belong to the care set of the patch function.
  - The off-set of the patch function (U_{off}) at x_i corresponds to those points in the care set which are in the variety $V(h'_i, J_0)$, i.e. points $\{\boldsymbol{a}\}$ where $h_i(\boldsymbol{a}) \neq 0$ and $h'_i(\boldsymbol{a}) = 0$.
  - The on-set (U_{on}) of the patch function comprises the remaining points.

a_0, a_1, b_0, b_1	h_i	h'_i	a_0, a_1, b_0, b_1	h_i	h'_i
0,0,0,0	0	0	1,0,0,0	0	$\frac{56}{5}$
0,0,0,1	0	0	1,0,0,1	0	0
0,0,1,0	0	1	1,0,1,0	0	1
0,0,1,1	0	1	1,0,1,1	0	1
0,1,0,0	0	0	1,1,0,0	0	$\frac{56}{5}$
0,1,0,1	-3	0	1,1,0,1	1	0
0,1,1,0	2	1	1,1,1,0	1	1
0,1,1,1	1	1	1,1,1,1	-1	1

Table 1: Evaluating h_i and h'_i at all inputs.

Thus, using h_i, h'_i, we can compute a Boolean rectification function table (truth table) for $U^{\mathbb{B}}$ such that: (i) $U^{\mathbb{B}}$ evaluates to 0 at points $\boldsymbol{a} \in V(h'_i, J_0)$ (off-set); (ii) $U^{\mathbb{B}}$ can be restricted (forced, or changed) to 0 or 1 at points $\boldsymbol{a} \in V(h_i, J_0)$ (don't cares); and (iii) $U^{\mathbb{B}}$ evaluates to 1 elsewhere (on-set). Then, using a logic minimizer, the care-set of the patch function could be optimized w.r.t. the don't care set. Unfortunately, it is not feasible to "evaluate" h_i, h'_i at all points in the design space and construct a large truth table. Instead, we use the symbolic approach of [27] (Alg. 1 in [27]) that takes a polynomial $U(X) \in \mathbb{Q}[X]$, and computes a Boolean function $U^{\mathbb{B}}(X)$, such that: i) U and $U^{\mathbb{B}}$ have the same zeros; and ii) wherever U evaluates to non-zero (rational) values, $U^{\mathbb{B}}$ evaluates to 1. We implement the approach of [27] using an OKFDD package, and translate h_i, h'_i to their corresponding Boolean functions. Through them, we compute the on-set, and the dc-set, attach them to the net x_i in C and synthesize C to obtain an efficient implementation.

Example 5.3. When the approach of [27] is applied to Ex. 5.1, we obtain $U_{dc} = \overline{a_1} \vee \overline{b_0} \cdot \overline{b_1}$, and $h'^{\mathbb{B}}_i = b_0 \vee a_0 b_1$. When $h'^{\mathbb{B}}_i$ is simplified w.r.t. U_{dc}, we obtain $e_3 = b_0$ as the optimized patch function at e_3.

6 EXPERIMENTS

Table 2 presents experimental results for performing rectification of faulty integer multiplier benchmarks. 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. The naming convention is as: "PPG-PPA-FSA", e.g., a *sp-ar-rc* indicates simple partial product for PPG, array structure for PPA, and ripple-carry adder for FSA. Similarly, *wt* indicates Wallace-tree, *cl* indicates a carry-look-ahead adder and

Table 2: Single fix Rectification results of Integer Multipliers; n = operand width; *revsca* = time to generate remainders using [20]; *amulet* = time to generate remainders using [11]; RC = time to do rectification check; CPF = time to compute patch function; $TT = Totaltime = min(revsca, amulet) + RC + CPF$; SPC = Synthesized patch circuit; CI = Craig Interpolation; A = area (# gates); D = Delay (topological depth); Time-Out (TO) = 48000s; NA = not applicable.

Benchmarks	*n*	*target location*	**Algebraic**					*SAT/CI*	**SPC Singular**		**SPC CI**	
			revsca	*amulet*	*RC*	*CPF*	*TT*	*TT*	*A*	*D*	*A*	*D*
SP-AR-RC	4	n41	0.02	0	0.04	0.06	0.1	39.70	3	2	3	2
	8	n38	0.02	0	0	0.05	0.05	0.15	1	1	1	1
	16	n156	0.04	0.02	0.04	0.06	0.12	39.06	7	4	2512	84
	32	n277	0.26	0.06	0	0.05	0.11	1332.11	14	7	775191	8280
	64	n279	TO	TO	NA	NA	NA	888.87	NA	NA	1	1
SP-WT-CL	4	n43	0	0	0.04	0.05	0.09	0.06	5	3	5	3
	8	n51	0.12	TO	0.19	0.18	0.49	210.9	7	4	1188753	44955
	16	n98	613.46	TO	0.06	6377.56	6991.08	TO	20	8	NA	NA
BP-AR-RC	4	n22	0.02	0	0.04	0.05	0.09	0.25	2	2	7	3
	8	n67	0.02	0	0	0.09	0.09	419.36	23	8	1858717	139738
	16	n72	0.08	0.02	0.04	TO	NA	TO	NA	NA	NA	NA
	32	n140	0.87	0.14	0.04	0.05	0.23	TO	13	9	NA	NA
	64	n337	15.63	0.44	0.02	0.23	0.69	TO	4	4	NA	NA
BP-WT-CL	4	n48	0.02	0	0	0.08	0.10	0.18	21	10	9	4
	8	n84	0.06	TO	0.05	0.06	0.17	357.49	4	4	1858717	139738
	16	n116	1359.18	TO	0.06	0.06	1359.3	TO	4	4	NA	NA

bp indicates Booth product. The circuits are taken from [11] and mapped using the *abc* tool with the gate library cadence.genlib. Bugs are introduced by misconnecting wires or changing the gates in the circuit. Rectification targets are identified and a check is performed to see if the circuit is single-fix rectifiable.

We utilize *revsca* [20], *amulet* [11], to perform GBR to obtain remainders r_L and r_H. *Singular* [5] is used for rectification check and for computing h_i, h_i' using Gröbner bases, and *abc* and *sis* for logic synthesis. The rational polynomials are translated into corresponding Boolean functions using our custom implementation of [27]. The final implementation cost of the patch functions (area, delay) is displayed. 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.

The area and delay of the patch computed using the algebraic technique (SPC Singular) is often orders of magnitude smaller compared to the patch computed using the interpolation-based approach (SPC CI). For some architectures, SAT/CI based technique even fails to compute a rectification patch.

Table 3 presents results that depict the application of our approach to optimize (bug-free) integer multipliers by computing ODC-sets at a few internal nets in the circuit. Using a unate covering problem, we identify a minimum number of internal nets of C that cover all POs, given as "# targets" in the table. At these nets, ODCs are computed using our PLS approach, attached at these targets, and the entire circuit is synthesized using the tools *sis* (script.rugged) and *abc* (*dch, b, resyn2, map* commands). We report the (mapped) Area/Delay statistics for the original circuit netlists and the synthesized ones. Results demonstrate that there is further scope of optimization of multiplier architectures using don't cares, as in most cases, both area and delay can be improved upon.

Table 3: ODC based logic optimization of Integer Multiplier; n = operand width/word-length of benchmark multiplier designs; SCA = Synthesized circuit area; SCD = Synthesized circuit delay; OCA = Original circuit area; OCD = Original circuit delay.

Benchmarks	n	*# targets*	*SCA*	*SCD*	*OCA*	*OCD*
SP-AR-RC	4	4	86	8	76	9
	8	3	323	18	386	23
	16	3	1324	44	1866	51
	32	4	5051	93	7704	107
SP-WT-CL	4	4	58	10	78	8
	8	4	421	17	467	14
	16	3	1618	23	2240	21
	16	4	1668	23	2240	21
BP-AR-RC	16	3	1269	37	1338	45
	32	4	4732	70	4912	89
BP-WT-CL	8	4	347	15	439	16
	16	4	1638	22	1789	22

7 CONCLUSION

We present an approach to partial logic synthesis of arithmetic circuits using computer algebra techniques. We identify internal nets as potential rectification targets x_i. We algebraically perform rectification checks to ascertain the existence of rectification functions. Using Gröbner bases, we show how to compute rectification polynomials for the care and don't-care sets. The functions are computed as polynomials over the field of rationals, which are then appropriately synthesized to patch the circuits. Our approach also computes ODCs at internal nets, using which we show that integer multiplier architectures can be further simplified. Experiments demonstrate the efficacy of our approach over contemporary SAT/interpolation-based techniques.

REFERENCES

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

[2] M. Ciesielski, T. Su, A. Yasin, and C. Yu. 2020. Understanding Algebraic Rewriting for Arithmetic Circuit Verification: A Bit-Flow Model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2020).

[3] D. Cox, J. Little, and D. O'Shea. 2007. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra.* Springer.

[4] Jiteshri Dasari and Maciej Ciesielski. 2023. Efficient Formal Verification and Debugging of Arithmetic Divider Circuits. In *ICCAD*. 1–9.

[5] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. 2016. SINGULAR 4-1-0 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de.

[6] F. Farahmandi and P. Mishra. 2017. Automated Debugging of Arithmetic Circuits Using Incremental Gröbner Basis Reduction. In *International Conference on Computer Design (ICCD)*. 193–200.

[7] A. M. Gharehbaghi and M. Fujita. 2017. A new approach for selecting inputs of logic functions during debug. In *International Symposium on Quality Electronic Design (ISQED)*. 166–173.

[8] U. Gupta, I. Ilioaea, V. Rao, A. Srinath, P. Kalla, and F. Enescu. 2019. Rectification of Arithmetic Circuits with Craig Interpolants in Finite Fields. In *VLSI-SoC: Design and Engineering of Electronics Systems Based on New Computing Paradigms*. Vol. 561. Springer International Publishing, Chapter 5, 79–106.

[9] U. Gupta, P. Kalla, I. Ilioaea, and F. Enescu. 2019. Exploring Algebraic Interpolants for Rectification of Finite Field Arithmetic Circuits with Groebner Bases. In *24th IEEE European Test Symposium (ETS)*. 1–6.

[10] J. H. R. Jiang, V. N. Kravets, and N. Z. Lee. 2020. Engineering Change Order for Combinational and Sequential Design Rectification. In *Design, Automation Test in Europe Conference Exhibition (DATE)*. 726–731.

[11] Daniela Kaufmann and Armin Biere. 2021. AMulet2.0 for Verifying Multiplier Circuits. In *Tools and Algorithms for the Construction and Analysis of Systems*.

[12] Daniela Kaufmann, Armin Biere, and Manuel Kauers. 2019. Incremental column-wise verification of arithmetic circuits using computer algebra. *Formal Methods in System Design* (Feb 2019).

[13] Daniela Kaufmann, Armin Biere, and Manuel Kauers. 2019. Verifying Large Multipliers by Combining SAT and Computer Algebra. In *Formal Methods in Computer-Aided Design (FMCAD)*. 28–36.

[14] V. N. Kravets, N. Lee, and J. R. Jiang. 2019. Comprehensive Search for ECO Rectification Using Symbolic Sampling. In *Design Automation Conference (DAC)*. 1–6.

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

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

[17] J. Lv, P. Kalla, and F. Enescu. 2013. Efficient Gröbner Basis Reductions for Formal Verification of Galois Field Arithmetic Circuits. In *IEEE Trans. on CAD*, Vol. 32. 1409–1420.

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

[19] Alireza Mahzoon, Daniel Große, and Rolf Drechsler. 2018. Combining Symbolic Computer Algebra and Boolean Satisfiability for Automatic Debugging and Fixing of Complex Multipliers. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 351–356.

[20] Alireza Mahzoon, Daniel Große, and Rolf Drechsler. 2019. RevSCA: Using Reverse Engineering to Bring Light into Backward Rewriting for Big and Dirty Multipliers. In *Design Automation Conference*.

[21] Vikas Rao. 2018. Discussions on automatic debugging approaches. http://eng.utah.edu/~utkarshg/cex.pdf,. Internal Report.

[22] V. Rao, U. Gupta, A. Srinath, I. Ilioaea, P. Kalla, and F. Enescu. 2018. Post-Verification Debugging and Rectification of Finite Field Arithmetic Circuits using Computer Algebra Techniques. In *Formal Methods in Computer-Aided Design (FMCAD)*. 1–9.

[23] V. Rao, H. Ondricek, P. Kalla, and F. Enescu. 2021. Algebraic Techniques for Rectification of Finite Field Circuits. In *International Conference on Very Large Scale Integration (VLSI-SoC)*.

[24] V. Rao, H. Ondricek, P. Kalla, and F. Enescu. 2021. Rectification of Integer Arithmetic Circuits using Computer Algebra Techniques. In *IEEE International Conference on Computer Design (ICCD)*. 186–195.

[25] Daniela Ritirc, Armin Biere, and Manuel Kauers. 2017. Column-Wise Verification of Multipliers Using Computer Algebra. In *Formal Methods in Computer-Aided Design (FMCAD)*. 23–30.

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

[27] B. Sampathkumar, B. Martin, R. Das, P. Kalla, and F. Enescu. 2023. Logic Synthesis from Polynomials in the Field of Rationals. In *Proc. Intl. Symp. on Multi-Valued Logic*.

[28] C. Scholl and B. Becker. 2001. Checking Equivalence for Partial Implementations. In *Design Automation Conference (DAC)*.

[29] K. F. Tang, P. K. Huang, C. N. Chou, and C. Y. Huang. 2012. Multi-patch Generation for Multi-error Logic Rectification by Interpolation with Cofactor Reduction. In *Design, Automation Test in Europe Conference Exhibition (DATE)*. 1567–1572.

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

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

[32] H. T. Zhang and J. H. R. Jiang. 2018. Cost-Aware Patch Generation for Multi-Target Function Rectification of Engineering Change Orders. In *Design Automation Conference (DAC)*. 1–6.