Word-Level Multi-Fix Rectifiability of Finite Field Arithmetic Circuits

Vikas Rao[1], Irina Ilioaea[2], Haden Ondricek[1], Priyank Kalla[1], and Florian Enescu[3]
[1]Electrical & Computer Engineering, University of Utah
[2]Department of Mathematics, Louisiana State University Shreveport
[3]Mathematics & Statistics, Georgia State University

***Abstract*—Deciding whether a faulty circuit can be rectified at a given set of nets to match its intended specification constitutes a critical problem in post-verification debugging and rectification. Contemporary approaches which utilize Boolean SAT and Craig Interpolation techniques are infeasible in proving the rectifiability of arithmetic circuits. This paper presents a novel approach using symbolic computer algebra to prove the rectifiability of a faulty finite field arithmetic circuit at a given set of m nets. Our approach uses a word-level polynomial model and an application of a Gröbner basis decision procedure. The finite fields corresponding to the datapath word-length (n) and the patch word-length (m) may not be compatible. We make new mathematical and algorithmic contributions which resolve this disparity by modeling the problem in an appropriate composite field. Experiments demonstrate the efficacy of our word-level approach to ascertain multi-fix rectifiability compared to contemporary approaches.**

***Index Terms*—Debug, Rectification, Arithmetic Circuits, Gröbner Basis, Finite Fields**

I. Introduction

Formal verification checks whether a circuit implementation ($Impl$) conforms to its specification ($Spec$). In cases where verification detects the presence of bugs, debugging and rectification are performed. Rectification entails identifying candidate nets (targets) and determining whether the circuit can be patched at these targets. If the targets admit rectification, corresponding rectification functions are computed which make the $Impl$ conform to its $Spec$. This problem manifests itself in engineering change orders (ECO) – where a current $Impl$ needs to be rectified (preferably, with local modifications) to match the ECO-modified $Spec$. As a result, the problem has witnessed renewed interest by the logic synthesis, testing and verification communities [**?**], [**?**], [**?**], etc. These approaches are successful in rectifying random-logic circuits which are employed in control-dominated applications; However, they are infeasible for the rectification of arithmetic circuits.

This paper addresses the problem of ascertaining the rectifiability of buggy finite field arithmetic circuits at a given set of m targets against a polynomial function (Spec) over finite fields. Such circuits find application in cryptography and error-control codes. As arithmetic bugs may lead to security vulnerabilities [**?**], their rectification is of utmost importance. Our approach models the set of m targets as a bit-vector, enabling word-level reasoning, and utilizes techniques from Symbolic Computer Algebra (SCA) to determine rectifiability. The subsequent problem of computing rectification functions for these m targets is beyond the scope of this paper. *Word-level rectifiability checking* for arithmetic circuits is a challenging problem in its own right, and this manuscript covers its various facets.

Prior Work: Recent works attempt rectification using SCA techniques for finite field circuits [**?**], [**?**], and for integer arithmetic circuits [**?**], [**?**]. However, these algebraic approaches address only *single-fix* rectification – where rectification is attempted only at a *single net*. This is too restrictive, and depending on the nature of the bugs, the circuit may not admit single-fix rectification. In such cases, the correction has to be attempted at multiple targets. This is called *multi-fix* rectification in literature. The focus of this paper is m-target *multi-fix rectifiability* (MFR) of finite field circuits.

Contemporary approaches formulate the rectifiability checking problem inherently as part of a rectification procedure. They use quantified Boolean formula solving, Craig Interpolation, or iterative SAT solving [**?**], [**?**], [**?**]. The approach [**?**] considers multiple targets simultaneously and formulates the MFR check as a QBF. The QBF is then translated to a SAT problem and solved iteratively using a cofactor reduction technique. More recent techniques use this QBF formulation coupled with heuristics that improve resource awareness in patch generation [**?**], and the selection of effective targets [**?**]. The recent symbolic sampling approach [**?**] uses simulation to identify targets and ascertain rectifiability. However, models based on Boolean functions and SAT solvers are infeasible to perform rectifiability checking on arithmetic circuits. Our experiments show that contemporary SAT solvers fail to rectify finite field circuits beyond 16-bit operands.

Problem Statement: We are given the following: i) as the $Spec$, a multivariate polynomial f with coefficients in a finite field of 2^n elements (denoted $\mathbb{F}_{2^n}$), for a given $n \in \mathbb{Z}_{\geq 1}$; ii) a primitive polynomial $P_n(X)$ of degree n with coefficients in $\{0,1\}$ to construct $\mathbb{F}_{2^n}$; iii) an incorrect $Impl$ circuit C, with no assumptions on the number or the type of bugs present in C; and iv) a set of m targets from C, provided beforehand or selected using the heuristics proposed in [**?**], [**?**], [**?**]. The circuit may or may not be rectifiable at these m targets. *The objective is to check whether the given set of m targets collectively admit multi-fix rectification.* This MFR-check ascertains whether rectification functions exist that can patch C at these m-targets.

Approach: The given $Impl$ C, with operand word-length n, is modeled as a polynomial ideal in the multivariate poly-

nomial ring with coefficients in the finite field $\mathbb{F}_{2^n}$, denoted $\mathbb{F}_{2^n}[x_1, \ldots, x_d]$. The m targets are collected as an m-bit-vector word W, which evaluates in $\mathbb{F}_{2^m}$. For the rectification check, the unknown rectification function (U) is modeled as a (word-level) polynomial function in primary inputs (i.e. $W = U(X_{PI})$), which maps n-bit primary inputs X_{PI} to an m-bit word $W : \mathbb{F}_2^{|X_{PI}|} \rightarrow \mathbb{F}_{2^m}$. The rectifiability check is then formulated with algebraic geometry over finite fields and solved using Gröbner basis (GB) techniques [**?**].

Contributions: Our word-level algebraic approach enables efficient MFR checking of finite field arithmetic circuits. This word-level formulation poses new mathematical challenges: the field $\mathbb{F}_{2^m}$ might not be compatible with the field $\mathbb{F}_{2^n}$, which prevents us from performing algebraic operations in a unified domain. We overcome this problem by computing the smallest single field $\mathbb{F}_{2^k}$ containing both $\mathbb{F}_{2^m}, \mathbb{F}_{2^n}$. This requires the computation of a specific primitive polynomial $P_k(X)$ for $\mathbb{F}_{2^k}$. We present an approach to compute $P_k(X)$ using polynomial factorization and composite fields. We utilize the unified framework and derive an efficient GB-based word-level decision procedure for m-target MFR checking. Experiments conducted on various finite field benchmarks with different operand and patch word-lengths, n and m, respectively, corroborate the efficacy of our approach.

Paper Organization: The following section covers preliminary background. Section III reviews the polynomial modeling concepts. Section IV describes the setup of a unified computational framework for MFR checking. Word-level rectification check is described in Section **??**. Experimental results are described in Section **??**, and Section **??** concludes the paper.

II. Preliminaries: Notation and Background

This section reviews some relevant concepts from finite fields, symbolic computer algebra and associated algorithms that are utilized in this paper.

Finite Field and Primitive Polynomial: Let $\mathbb{F}_2 = \{0, 1\}$ be the field of 2 elements, and let $\mathbb{F}_q = \mathbb{F}_{2^n}$ denote the finite field of $q = 2^n$ elements, for a given n. $\mathbb{F}_{2^n}$ is the n-dimensional extension of $\mathbb{F}_2$, and it is constructed as $\mathbb{F}_{2^n} = \mathbb{F}_2[X] \pmod{P_n(X)}$. Here $P_n(X) \in \mathbb{F}_2[X]$ is a *primitive polynomial*, i.e. a degree-n polynomial, irreducible in $\mathbb{F}_2$, with a root γ ($P_n(\gamma) = 0$). Consequently, γ is called a primitive element (PE) of $\mathbb{F}_{2^n}$ and it generates the cyclic group: $\mathbb{F}^*_{2^n} = \{1 = \gamma^{2^n-1}, \gamma, \gamma^2, \ldots, \gamma^{2^n-2}\}$. Note: $\forall \gamma \in \mathbb{F}_{2^n}, \gamma^{2^n-1} = 1$, thus $\gamma^{2^n} = \gamma$.

Minimal Polynomial of γ: If $\gamma \in \mathbb{F}_{2^n}$ is a root of $P_n(X)$, then $\gamma^{2^l}, l \geq 0$, is also a root of $P_n(X)$. Elements γ^{2^l} are conjugates of each other. Let $e > 0$ be the smallest integer such that $\gamma^{2^e} = \gamma$. Then $P_n(X) = \prod_{i=0}^{i=e-1}(X + \gamma^{2^i})$ is called the *minimal polynomial* of γ. If γ is also a PE of $\mathbb{F}_{2^n}$, then the minimal polynomial of γ is also a primitive polynomial, and has degree $= n$. Hence, $\mathbb{F}_{2^n}$ is also called the γ-extension of $\mathbb{F}_2$, also denoted as $\mathbb{F}_2(\gamma)$. An element $A \in \mathbb{F}_{2^n}$ can be written as $A = a_0 + a_1 \cdot \gamma + \cdots + a_{n-1} \cdot \gamma^{n-1}$, where $a_0, \ldots, a_{n-1} \in \mathbb{F}_2$. In $\mathbb{F}_q$, the addition ($''+''$) and multiplication ($''\cdot''$) operations are performed in the base field $\mathbb{F}_2$ and reduced modulo the corresponding primitive polynomial $P_n(X)$. For $n, k \in \mathbb{Z}_{>0}$, if n divides k ($n \mid k$), then $\mathbb{F}_{2^n} \subset \mathbb{F}_{2^k}$. Thus, $\mathbb{F}_2 \subset \mathbb{F}_{2^k}, \forall k > 1$. The fields $\mathbb{F}_{2^n}$ have characteristic 2, and therefore $-1 = +1$ in $\mathbb{F}_{2^n}$.

Example II.1. *Consider the fields $\mathbb{F}_{2^4}$ and $\mathbb{F}_{2^2}$. Let $\mathbb{F}_{2^4}$ be constructed as $\mathbb{F}_2[X] \pmod{P_4(X) = X^4 + X^3 + 1}$, with $P_4(\gamma) = 0$. Let $\mathbb{F}_{2^2} = \mathbb{F}_2[X] \pmod{P_2(X) = X^2 + X + 1}$, with $P_2(\alpha) = 0$. Since $2 \mid 4$, $\mathbb{F}_{2^2} \subset \mathbb{F}_{2^4}$. We demonstrate this field containment in terms of their PEs.*

Compute γ^{2^l} for $l > 0$: for $l = 1, \gamma^{2^l} = \gamma^2$; $l = 2, \gamma^{2^l} = \gamma^4$; $l = 3$, $\gamma^{2^l} = \gamma^8$; $l = 4, \gamma^{2^l} = \gamma^{16} = \gamma$. Since $\gamma^{16} = \gamma$ (the elements repeat), we have that $\{\gamma, \gamma^2, \gamma^4, \gamma^8\}$ are conjugates. Thus the minimal polynomial of γ can be reconstructed as $(X + \gamma)(X + \gamma^2)(X + \gamma^4)(X + \gamma^8)$, which simplifies to $X^4 + X^3 + 1 = P_4(X)$ itself. Similarly, $\{\alpha, \alpha^2\} \in \mathbb{F}_{2^2}$ are conjugates, and $(X + \alpha)(X + \alpha^2) = P_2(X)$.

Now consider the element $\gamma^5 \in \mathbb{F}_{2^4}$. Then γ^{10} is its only conjugate, and $(X + \gamma^5)(X + \gamma^{10}) = X^2 + X + 1 = P_{2^2}(X)$. In other words, $\{0, 1, \alpha = \gamma^5, \alpha^2 = \gamma^{10}\}$ are the 4 elements of $\mathbb{F}_{2^2}$, also contained in $\mathbb{F}_{2^4}$; implying that $\mathbb{F}_{2^2} \subset \mathbb{F}_{2^4}$.

Let $R = \mathbb{F}_q[x_1, \ldots, x_d]$ be the polynomial ring in variables $x_1, \ldots, x_d$ with coefficients in $\mathbb{F}_q$. A polynomial $f \in R$ is written as a finite sum of terms $f = c_1 M_1 + \cdots + c_p M_p$, where c_i's are the coefficients and M_i's the monomials, and $c_i M_i$'s terms of f. To systematically manipulate the polynomials, a term order $>$ is imposed on all polynomials of R, such that subject to $>$, $c_1 M_1 > c_2 M_2 > \cdots > c_p M_p$. Then $lt(f) = c_1 M_1$ denotes the leading term of f.

Polynomial Reduction: Let $F = \{f_1, \ldots, f_s\}$ be a set of polynomials in R and $f \in R$ be another polynomial. Then $f \xrightarrow{F}_+ r$ denotes the *reduction* of f modulo F resulting in remainder r, which is obtained by iteratively canceling terms in f by $lt(f_j), f_j \in F$, via polynomial division (cf. Alg. 1.5.1 [**?**]). The remainder r is said to be *reduced* such that no term in r is divisible by the leading term of any $f_j \in F$.

Ideals: A set of polynomials $F = \{f_1, \ldots, f_s\}$ from R, generates the **ideal** $J = \langle F \rangle \subseteq R$, defined as $J = \langle f_1, \ldots, f_s \rangle = \{h_1 \cdot f_1 + \cdots + h_s \cdot f_s : h_1, \ldots, h_s \in R\}$. Polynomials $f_1, \ldots, f_s$ form the basis or generators of ideal J.

Varieties: Let $\boldsymbol{a} = (a_1, \ldots, a_d) \in \mathbb{F}_q^d$ be a point in the affine space, and f a polynomial in R. If $f(\boldsymbol{a}) = 0$, we say that f *vanishes* on $\boldsymbol{a}$. We have to analyze the *set of all common zeros* of the polynomials of $F = \{f_1, \ldots, f_s\}$ that lie within the field $\mathbb{F}_q$ – i.e. the set of all points $\boldsymbol{a} \in \mathbb{F}_q^d$ such that $f_1(\boldsymbol{a}) = \cdots = f_s(\boldsymbol{a}) = 0$. This zero set is called the **variety**, which depends not just on the given set of polynomials in F, but rather on the ideal generated by them. We denote the variety as $V(J)$, where: $V(J) = V_{\mathbb{F}_q}(J) = V_{\mathbb{F}_q}(f_1, \ldots, f_s) = \{\boldsymbol{a} \in \mathbb{F}_q^d : \forall f \in J, f(\boldsymbol{a}) = 0\}$.

Given two ideals $J_1 = \langle f_1, \ldots, f_s \rangle, J_2 = \langle h_1, \ldots, h_r \rangle$, we denote their **sum** $J_1 + J_2 = \langle f_1, \ldots, f_s, h_1 \ldots, h_r \rangle$, and their **product** $J_1 \cdot J_2 = \langle f_i \cdot h_j : 1 \leq i \leq s, 1 \leq j \leq r \rangle$. Ideals and varieties are dual concepts: $V(J_1 + J_2) = V(J_1) \cap V(J_2)$, and $V(J_1 \cdot J_2) = V(J_1) \cup V(J_2)$.

Gröbner Bases: An ideal J may have many different sets of generators/bases. A Gröbner basis (GB) is one such generating set $G = \{g_1, \ldots, g_t\}$ with special properties that alow to solve many polynomial decision problems.

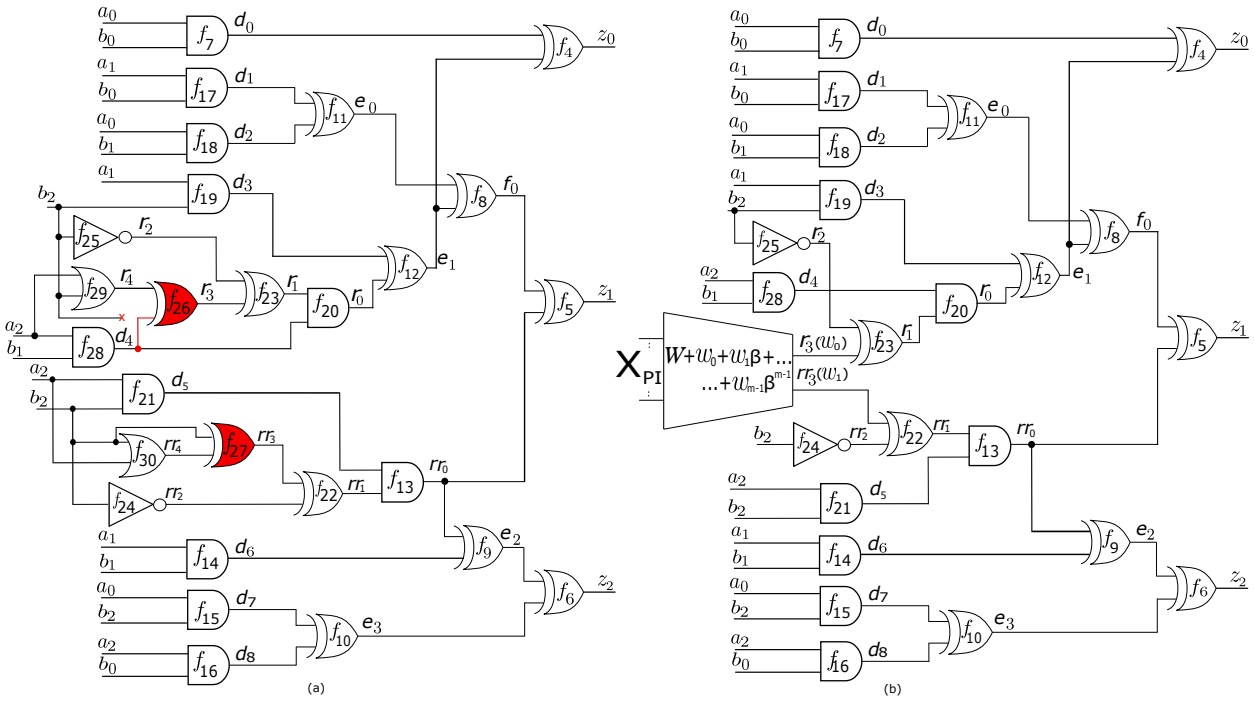

Fig. 1: (a) An incorrect implementation of a circuit C: a 3-bit finite field multiplier (n=3) with bugs introduced at net r_3 (AND gate replaced with an XOR gate and one of the inputs misconnected to d_4 instead of b_2) and net rr_3 (AND gate replaced with an XOR gate). (b) The patch function is modeled as a 2-bit-vector word (m=2), $f_W : W + r_3 + \beta \cdot rr_3$. Here, the rectification targets are $W = \{w_0, w_1\}$, where $w_0 = r_3, w_1 = rr_3$).

Definition II.1. [Gröbner Basis] [**?**]: For a monomial ordering $>$, a set of non-zero polynomials $G = \{g_1, \cdots, g_t\}$ contained in an ideal J, is called a GB of $J \iff \forall f \in J,\ f \xrightarrow{g_1,\ldots,g_t}_+ 0$.

The GB G for an ideal J can be computed using the Buchberger's algorithm (cf. Alg. 1.7.1 in [**?**]), which takes as input a set of polynomials $F = \{f_1, \ldots, f_s\}$ and computes its GB $G = \{g_1, \ldots, g_t\}$, such that $J = \langle F \rangle = \langle G \rangle$. Moreover, the polynomials of F and G have the same solution-set, i.e. $V(J) = V(F) = V(G)$.

Ideal Membership Test: The above definition inherently provides a decision procedure to test for membership of a polynomial in an ideal: f is a member of an ideal J *if and only if* $f \xrightarrow{GB(J)}_+ 0$. When $f \notin J$, then division by $GB(J)$ results in a non-zero remainder r that is unique/canonical.

For any element $\varphi \in \mathbb{F}_q$, $\varphi^q = \varphi$ holds. Therefore, the polynomial $x^q - x$ vanishes everywhere in $\mathbb{F}_q$, and we call it a *vanishing polynomial*. We denote by $F_0 = \{x_1^q - x_1, \ldots, x_d^q - x_d\}$ the set of all vanishing polynomials in R, and $J_0 = \langle F_0 \rangle$ as the ideal of vanishing polynomials. Further, $V_{\mathbb{F}_q}(J_0) = \mathbb{F}_q^d$, and for any ideal J, $V_{\mathbb{F}_q}(J) = V_{\mathbb{F}_q}(J + J_0)$.

III. Polynomial Modeling of the Circuit

A multivariate polynomial f over $\mathbb{F}_{2^n}$ is given as a specification, where n is the operand word-length (datapath size). A corresponding combinational circuit C is given as its implementation. The function implemented by C is modeled with a system of polynomials over $R = \mathbb{F}_{2^n}[Z, A, x_1, \ldots, x_d]$. Here $\{x_1, \ldots, x_d\}$ correspond to all the bit-level variables (nets) in the circuit, and Z, A the output and input words, respectively. The field $\mathbb{F}_{2^n}$ is constructed as $\mathbb{F}_{2^n} = \mathbb{F}_2[X] \pmod{P_n(X)}$, where $P_n(X)$ is the given primitive polynomial of degree n with γ as a root (PE), i.e. $P_n(\gamma) = 0$. As C comprises Boolean logic gates, the gates are represented by polynomials $\pmod 2$, i.e. over $\mathbb{F}_2 (\subset \mathbb{F}_{2^n})$, as the $\mathbb{B} \mapsto \mathbb{F}_2$ mapping:

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

Similarly, the primary output and input bits $z_i, a_i : i = 0, \ldots, n-1$ can be correlated to the corresponding operand words Z, A as follows:

$$\begin{aligned} f_1 &: Z + z_0 + \gamma \cdot z_1 + \cdots + \gamma^{n-1} \cdot z_{n-1} \\ f_2 &: A + a_0 + \gamma \cdot a_1 + \cdots + \gamma^{n-1} \cdot a_{n-1} \end{aligned} \tag{2}$$

Thus, the circuit is represented by a set of polynomials $F = \{f_1, \ldots, f_s\} \subset R$. Let $J = \langle F \rangle$ be the ideal generated by this set. Let $F_0 = \{x_i^2 - x_i, Y^{2^n} - Y : x_i \in \text{bit-level variables}, Y \in \text{word-level variables}\}$ be the set of all vanishing polynomials, and $J_0 = \langle F_0 \rangle$ be the corresponding ideal of all bit- and word-level vanishing polynomials. Then the ideal $J + J_0 = \langle F \cup F_0 \rangle$ models the functionality of *Impl* C.

In the manuscript, we use the circuit of Fig. 1 as a running example to demonstrate our algebraic approach for MFR.

Example III.1. *The circuit C in Fig. 1 (a) is an incorrect implementation of a 3-bit (n=3) Mastrovito multiplier. The*

field $\mathbb{F}_{2^3}$ is constructed using $P_3(X) = X^3 + X + 1$ and let γ be a PE of $\mathbb{F}_{2^3}$, s.t. $P_3(\gamma) = 0$. The Spec polynomial is $f : Z + A \cdot B$, where Z is the output word, and A, B the input words. Impose a lex term order with variable order:

$\{Z\} > \{A > B\} > \{z_0 > z_1 > z_2\} > \cdots > \{d_1 > d_2 > d_3 > r_0 > d_5 > rr_1\} > \{r_1 > rr_3 > rr_2\} > \{r_2 > r_3 > rr_4\} > \{r_4 > d_4\} > \{a_0 > a_1 > a_2 > b_0 > b_1 > b_2\}$.

Note that in this term order, the bit-level variables are ordered based on the topology of the circuit from POs to PIs, i.e. reverse topologically. Using this term order, the following polynomials represent C:

$$\begin{aligned} f_1 &: Z + z_0 + \gamma \cdot z_1 + \gamma^2 \cdot z_2; & f_{22} &: rr_1 + rr_3 + rr_2; \\ f_2 &: A + a_0 + \gamma \cdot a_1 + \gamma^2 \cdot a_2; & f_{23} &: r_1 + r_2 + r_3; \\ f_3 &: B + b_0 + \gamma \cdot b_1 + \gamma^2 \cdot b_2; & f_{26} &: r_3 + r_4 + d_4; \\ f_4 &: z_0 + d_0 + e_1; & f_{27} &: rr_3 + rr_4 + b_2; \\ f_5 &: z_1 + f_0 + rr_0; & \ldots & \\ \ldots & & f_{30} &: rr_4 + a_2 + b_2 + a_2 b_2; \end{aligned}$$

Then $F = \{f_1, \ldots, f_{30}\}$, $F_0 = \{a_0^2 - a_0, \ldots, z_2^2 - z_2, A^8 - A, B^8 - B, Z^8 - Z\}$. So, ideal $J + J_0 = \langle F \cup F_0 \rangle$ models C.

Since $\mathbb{F}_2 \subset \mathbb{F}_{2^n}$, the polynomials in Eqn. (1) can also be interpreted as polynomials over $\mathbb{F}_{2^n}$. The advantage of working over $\mathbb{F}_{2^n}$ is that we can represent and manipulate both the bit-level and n-bit word-level polynomials in one unified domain. However, we are given a patch word-length m which is modeled as an m-bit-vector word over field $\mathbb{F}_{2^m}$. Since the field $\mathbb{F}_{2^m}$ might not be compatible with the field $\mathbb{F}_{2^n}$, i.e. m may not divide n, not every m-bit-vector can be construed as an element in $\mathbb{F}_{2^n}$. To overcome this incompatibility, the following section presents techniques which enable the modeling of polynomial operations in a unified domain.

IV. A Unified Framework for MFR

Word-level Rectification Patch: Given a set of m-targets, let $W = \{w_0, w_1, \ldots, w_{m-1}\}$ denote the m-bit vector word, over which MFR check is to be performed. Here, $w_0, \ldots, w_{m-1} \in \{x_1, \ldots, x_d\}$ are internal nets of C. A rectification patch corresponds to a function mapping $f_W : \mathbb{F}_2^{|X_{PI}|} \rightarrow \mathbb{F}_{2^m}$, and is represented as a polynomial function $U(X_{PI}) \in \mathbb{F}_{2^m}[X_{PI}]$. Here, U is an *unknown* polynomial with variables from X_{PI}, and coefficients from $\mathbb{F}_{2^m}$. MFR then implies that, algebraically, $W = U(X_{PI})$ rectifies the circuit.

Such a setup *interprets* U as a polynomial function which evaluates to values in the field $\mathbb{F}_{2^m}$. To construct $\mathbb{F}_{2^m}$, we *select* a degree-m primitive polynomial $P_m(X) \in \mathbb{F}_2[X]$, with β as a root ($P_m(\beta) = 0$), such that $\mathbb{F}_{2^m} = \mathbb{F}_2[X] \pmod{P_m(X)}$. Note that any degree-$m$ primitive polynomial ($P_m(X)$) can be selected. This implies that β is a PE of $\mathbb{F}_{2^m}$. Thus, the rectification patch polynomial function can be represented as: $f_W : W + U(X_{PI}) = W + \sum_{i=0}^{m-1} \beta^i w_i$.

Composite Field Framework: The circuit C is modeled over $\mathbb{F}_{2^n}$ and the rectification patch is modeled over $\mathbb{F}_{2^m}$. However, the Gröbner basis computations for MFR can only be performed over a ring with coefficients from a single field. We select the field $\mathbb{F}_{2^k}$ of 2^k elements, where k is the least common multiple of m and n ($k = LCM(m, n)$). Thus $\mathbb{F}_{2^k}$ corresponds to the smallest field containing both $\mathbb{F}_{2^m}$ and $\mathbb{F}_{2^n}$.

To construct $\mathbb{F}_{2^k}$, we must now identify a degree-k primitive polynomial $P_k(X) \in \mathbb{F}_2[X]$, and a corresponding primitive element α such that $P_k(\alpha) = 0$. While there may exist many degree-k primitive polynomials, the selected $P_k(X)$ needs to conform to specific conditions imposed by the *given* $P_n(X)$, and the *selected* $P_m(X)$.

Given the primitive elements β, γ of the fields $\mathbb{F}_{2^m}, \mathbb{F}_{2^n}$, respectively, let α be a PE of $\mathbb{F}_{2^k}$, such that $\mathbb{F}_{2^k} \supseteq \mathbb{F}_{2^m}, \mathbb{F}_{2^n}$. For the desired unified computational framework, we represent β, γ in terms of α. As $\alpha \in \mathbb{F}_{2^k}, \alpha^{2^k-1} = 1$. Similarly, $\beta \in \mathbb{F}_{2^m}$ implies that $\beta^{2^m-1} = 1$, and $\gamma \in \mathbb{F}_{2^n}$ makes $\gamma^{2^n-1} = 1$. Equating $\beta^{2^m-1} = \gamma^{2^n-1} = \alpha^{2^k-1} = 1$, we obtain:

$$\begin{aligned} \beta &= \alpha^{(2^k-1)/(2^m-1)} = \alpha^{\mu} \\ \gamma &= \alpha^{(2^k-1)/(2^n-1)} = \alpha^{\lambda} \end{aligned} \tag{3}$$

With this setup, we show that the desired primitive polynomial $P_k(X)$ for $\mathbb{F}_2(\alpha)$ exists as a degree-k **common factor** of $P_n(X^\lambda)$ and $P_m(X^\mu)$, and it can be found by performing univariate polynomial factorizations (UPFs) of $P_n(X^\lambda)$ and $P_m(X^\mu)$ in the base field $\mathbb{F}_2[X]$. Efficient algorithms to perform UPFs in finite fields are well-known **[?]**, which can be employed for this purpose.

Theorem IV.1. Given primitive polynomials $P_n(X), P_m(X) \in \mathbb{F}_2[X]$ of degrees n, m, respectively, along with $P_n(\gamma) = P_m(\beta) = 0$. Let $k = LCM(m, n)$ and α be a PE of $\mathbb{F}_{2^k}$, such that $\beta = \alpha^\mu$ and $\gamma = \alpha^\lambda$, where $\mu = (2^k - 1)/(2^m - 1)$ and $\lambda = (2^k - 1)/(2^n - 1)$. Perform UPF of $P_n(X^\lambda)$ and $P_m(X^\mu)$ in $\mathbb{F}_2[X]$. Then there exists a polynomial $P_k(X) \in \mathbb{F}_2[X]$ as a common factor of $P_m(X^\mu)$ and $P_n(X^\lambda)$, such that $P_k(X)$ is a degree-k primitive polynomial with $P_k(\alpha) = 0$.

Proof. Since α is given as a PE of $\mathbb{F}_{2^k}$, the minimal polynomial of α is a primitive polynomial. Denote it as $P_k(X)$. Then $P_k(\alpha) = 0$, i.e. α is a root of $P_k(X)$. However, $P_m(\beta) = 0 \implies P_m(\alpha^\mu) = 0$. So $P_m(X^\mu)$ also has α as a root. This implies that $P_k(X)$ divides $P_m(X^\mu)$: $P_k(X) \mid P_m(X^\mu)$. Similarly, $P_k(X) \mid P_n(X^\lambda)$. So $P_k(X)$ is a common factor of both $P_m(X^\mu)$ and $P_n(X^\lambda)$. Since $P_m(X^\mu), P_n(X^\lambda) \in \mathbb{F}_2[X]$, and their UPFs are also performed in $\mathbb{F}_2[X]$, $P_k(X) \in \mathbb{F}_2[X]$. Moreover, as α is a PE of $\mathbb{F}_{2^k}$, and $P_k(X)$ is the minimal polynomial of α, we have that $P_k(X)$ is a primitive polynomial as $\mathbb{F}_{2^k} = \mathbb{F}_2(\alpha)$. Furthermore, by definition, as the minimal polynomial of α, $P_k(X)$ has degree $= k$. This completes the proof. □

Example IV.1. *Continuing with our rectification example of Fig. 1, assume that we are given $m = 2$ targets: $w_0 = r_3$, and $w_1 = rr_3$. The multi-fix patch is modeled as a 2-bit operand over $\mathbb{F}_{2^2} = \mathbb{F}_2[X] \pmod{P_2(X) = X^2 + X + 1}$ with $P_2(\beta) = 0$. It is given that the 3-bit circuit ($n = 3$) is designed over $\mathbb{F}_{2^3} = \mathbb{F}_2[X] \pmod{P_3(X) = X^3 + X + 1}$. With $k = LCM(m, n) = 6$, $P_6(X)$ is computed as follows:*

Using Eqn. (3), we have $\gamma = \alpha^9, \beta = \alpha^{21}$. Perform UPFs:

- $P_3(X^9) = (X^9)^3 + (X^9) + 1 = (X^6 + X^5 + X^2 + X + 1)(X^6 + X^5 + 1)$ $(X^6 + X^4 + X^3 + X + 1)(X^6 + X^4 + X^2 + X + 1)(X^3 + X + 1)$;

- $P_2(X^{21})=(X^{21})^2+(X^{21})+1=(X^6+X^5+X^2+X+1)$ $(X^6+X^5+1)(X^6+X^4+X^3+X+1)(X^6+X^5+X^3+X^2+1)$ $(X^6+X^5+X^4+X+1)(X^6+X+1)(X^6+X^3+1)$;

Among the degree-6 common factors of $P_3(X^9)$ and $P_2(X^{21})$, $X^6+X^5+X^2+X+1$ is omitted since it is non-primitive. We choose $P_6(X)=X^6+X^5+1$ as the required $P_k(X)$.

Note that if we (incorrectly) choose $P_6(X)=X^6+X^3+1$, then it implies that for its root α, we have:

$$\begin{aligned} \alpha^6+\alpha^3+1&=0\\ (\alpha^3)(\alpha^6+\alpha^3+1)&=0 \text{ (multiply by } \alpha^3)\\ \alpha^9+\alpha^6+\alpha^3&=0\\ \gamma+1&=0 \end{aligned} \tag{4}$$

as $\alpha^9=\gamma$ and $(\alpha^6+\alpha^3)=1$. However, $\gamma\neq 1$, as γ is a PE of $\mathbb{F}_{2^n}$. Therefore, selecting $P_k(X)$ that does not conform to Thm. IV.1 would lead to erroneous results.

V. Word-Level Rectification Check

The m-bit target word W is interpreted as an element in $\mathbb{F}_{2^m}$, such that $W=w_0+\beta\cdot w_1+\cdots+\beta^{m-1}\cdot w_{m-1}$, where β is a PE of $\mathbb{F}_{2^m}$. As $w_0,\ldots,w_{m-1}$ are bit-level variables, we first represent each w_i as a polynomial in terms of the word W, and then substitute their word-level expressions in generators of the ideal $J+J_0$.

Lemma V.1 (From [**?**]). Let $\theta_1,\ldots,\theta_t\in\mathbb{F}_{2^m}$. Then $(\theta_1+\theta_2+\cdots+\theta_t)^{2^i}=\theta_1^{2^i}+\theta_2^{2^i}+\cdots+\theta_t^{2^i}$, for $i\geq 1$.

By virtue of Lemma **??** and using the relation $W=\sum_{i=0}^{m-1}\beta^i\cdot w_i$, we compute $W^{2^j}, 0\leq j<m$:

$$\begin{aligned} W&=w_0+\cdots+\beta^{m-1}\cdot w_{m-1}\\ W^2&=w_0^2+\cdots+\beta^{2(m-1)}\cdot w_{m-1}^2\\ W^2&=w_0+\cdots+\beta^{2(m-1)}\cdot w_{m-1}\ (w_i^2=w_i)\\ &\cdots\\ W^{2^{m-1}}&=w_0+\cdots+\beta^{2^{m-1}(m-1)}\cdot w_{m-1} \end{aligned} \tag{5}$$

Considering $w_0,\ldots,w_{m-1}$ as m unknowns, and W and β as constants, we solve (using Gaussian elimination) the m linear equations of Eqn. (**??**) to obtain each w_i as a polynomial function in W,β: $w_i=\mathcal{F}_i(W,\beta)$. Each occurrence of w_i in the generators of $J+J_0$ can then be replaced with $\mathcal{F}_i(W,\beta)$.

Rectification setup: Using the aforementioned concepts, the rectification check is setup in $\mathbb{F}_{2^k}$ as follows:

1) Setup the polynomial ring as: $R'=\mathbb{F}_{2^k}[x_1,..,x_d,Z,A,W]$, by including the variable W, and constructing $\mathbb{F}_{2^k}$ using $P_k(X)$ (From Thm. IV.1), with $P_k(\alpha)=0$.
2) *Term order:* Derive a *variable order* on the nets of C such that each variable corresponding to the output of a gate appears earlier (lower index) than its immediate inputs. Moreover, include the word-level patch variable W in the variable order such that W is placed before the target net w_i which has the lowest index. Using such a variable order, impose a *lex* order on the monomials. We call the resulting term order the *Word-level Rectification Term Order (WRTO)* $>_R$.
3) Correspondingly update the set of polynomials F (describing the logic gates of C) to F' as follows: i) Begin by setting $F'=F$. Remove from F' the polynomials with w_i's as leading terms. ii) Substitute for w_i's the word-level polynomials $\mathcal{F}_i(W,\beta)$ as derived from Eqn. (**??**). iii) Add the polynomial $f_W: W+\sum_{i=0}^{m-1}\beta^i\cdot w_i$ to F'. iv) Substitute $\beta=\alpha^\mu,\gamma=\alpha^\lambda$ for the constants (Eqn. (3)). Denote ideal $J'=\langle F'\rangle\subset R'$.
4) Update the set of vanishing polynomials F_0 to F_0' to include the vanishing polynomial $W^{2^m}-W$. This restricts W to take values from $\mathbb{F}_{2^m}\subseteq\mathbb{F}_{2^k}$. Let $J_0'=\langle F_0'\rangle$.

The rectification check is then formulated by operating on the ideal $J'+J_0'=\langle F'\cup F_0'\rangle\subseteq R'$.

Example V.1. *Continuing with the running example of Fig. 1(b), let the target nets be $W=\{r_3,rr_3\}$, represented by the polynomial $f_W: W+r_3+\beta\cdot rr_3$. We begin with the set $F'=F$, and derive WRTO to be lex with variable order:*

$\{Z\}>\{A>B\}>\{z_0>z_1>z_2\}>\cdots>\{d_1>d_2>d_3>r_0>d_5>rr_1\}>\{r_1\}>\{\mathbf{W}\}>\{\mathbf{rr_3}>rr_2\}>\{r_2>\mathbf{r_3}>rr_4\}>\{r_4>d_4\}>\{a_0>a_1>a_2>b_0>b_1>b_2\}$.

*The polynomials f_{26},f_{27} are removed from F', as they correspond to the target net. Polynomials f_{22},f_{23} are updated to f_{22}',f_{23}' to represent rr_3,r_3 in terms of W. We use Eqn. (**??**) to represent these bit-level targets in terms of W : $rr_3=W^2+W$, and $r_3=\beta W^2+\beta^2 W$. These polynomials are updated as given below:*

$$\begin{aligned} &f_{22}': rr_1+(W^2+W)+rr_2\\ &f_{23}': r_1+r_2+(\beta W^2+\beta^2 W)\\ &f_W: W+r_3+\beta\cdot rr_3\\ &\beta=\alpha^{21} \text{ and } \gamma=\alpha^9\\ &F'=\{f_1,\ldots,f_{21},\ f_{22}',f_{23}',f_W,\ldots,f_{30}\}-\{f_{26},f_{27}\} \end{aligned}$$

We now state the *multi-fix rectification theorem* that checks for the existence of $W=U(X_{PI})$ as a MFR function.

Theorem V.1. [Multi-fix Rectification Theorem] Given the specification polynomial f, and the implementation C represented using the ideal $J'=\langle F'\rangle\subset R'$, where $F'=\{f_1,\ldots,\boldsymbol{f_W: W+\sum_{i=0}^{m-1}\beta^i\cdot w_i},\ldots,f_s\}$, and $J_0'=\langle F_0'\rangle$ is the corresponding ideal of vanishing polynomials in R'. Impose WRTO $>_R$ on R'. Construct the following ideals:

- $J_l'=\langle F_l'\rangle=\langle f_1,\ldots,f_W'=W+\delta[l],\ldots,f_s\rangle$, $1\leq l\leq 2^m$, where F_l' is obtained from F' by replacing $f_W\in F'$ with $f_W': W+\delta[l], 1\leq l\leq 2^m$, and $(\delta[1],\ldots,\delta[2^m])=(0,1,\beta,\ldots,\beta^{2^m-2})$.

Reduce f by $F_l'\cup F_0'$ to obtain the remainders rem_l: $f\xrightarrow{F_l'\cup F_0'}_+ rem_l$, for $1\leq l\leq 2^m$. Then, there exists a polynomial function $U(X_{PI}):\mathbb{F}_2^{|X_{PI}|}\rightarrow\mathbb{F}_{2^m}$, which, when implemented at the target W, rectifies C to match f *if and only if* $\bigcup_{l=1}^{2^m}V(rem_l)=\mathbb{F}_2^{|X_{PI}|}$.

A detailed proof of the above theorem can be devised using the Nullstellensatz, the circuit structure, and the ideal-variety correspondences. However, the proof is omitted for brevity.

TABLE I: Time is in seconds; *I* = Index, *n* = Datapath Size, *m* = target word size, *k* = composite field size (degree of $P_k(X)$), AM = Maximum resident memory utilization in Mega Bytes, #G = Number of gates $\times 10^3$, #BO = Number of faulty outputs, PBS = Required time for PolyBori setup (ring declaration/poly collection/spec collection), VMS = Required time for verification, polynomial factorization and computing $P_k(X)$, and MFR setup, RC = Required time for MFR check, TE = Required time for total execution

					Mastrovito						Montgomery						Point Addition					
I	***n***	***m***	***k***	**AM**	**#G**	**#BO**	**PBS**	**VMS**	**RC**	**TE**	**#G**	**#BO**	**PBS**	**VMS**	**RC**	**TE**	**#G**	**#BO**	**PBS**	**VMS**	**RC**	**TE**
1	**16**	**5**	**80**	100	0.8	6	0.04	0.06	0.12	0.22	0.9	16	0.04	0.56	35.6	36	0.9	7	0.06	0.11	1.73	1.9
2	**32**	**5**	**160**	120	2.8	8	0.13	0.12	0.4	0.65	2.8	32	0.13	0.57	27.6	28.3	2.9	13	0.18	0.8	134	135
3	**64**	**3**	**192**	160	11.2	5	0.57	0.45	227	228	9.6	47	0.52	0.32	1.79	2.63	10.6	64	0.84	0.56	58.1	59.5
4	**96**	**2**	**96**	240	24.5	5	1.47	0.26	0.83	2.56	21	96	1.36	1.27	13.3	16	24.8	96	2.46	0.64	14.9	18
5	**128**	**2**	**128**	370	43.2	5	3.23	0.5	2.03	5.76	35.8	128	2.8	1.4	64.2	68.4	43.2	128	6.45	1.55	73	81
6	**163**	**5**	**815**	550	69.8	6	6.04	3.36	11.9	21.3	57.5	128	5.2	6.8	262	274	71.6	22	15.7	4.7	15	35.4
7	**233**	**2**	**466**	750	119	3	13	1.2	0.01	14.2	112	233	11.5	3.5	360	375	122	233	19.2	2.15	0.15	21.5
8	**283**	**2**	**566**	1300	190	2	38	4.2	0.1	42.3	171	283	35	11	1503	1549	208	4	80.4	6.1	0.1	86.6
9	**409**	**2**	**818**	2400	384	2	190	5	0.1	195	340	409	134	10	4920*	5064	368	409	220	10	2007	2237
10	**571**	**2**	**1042**	5000	827	5	2150	12	0.1	2162	663	12	1313	82	0.2†	1395	813	5	2583	27	880	3490
11	**16**	**7**	**112**	100	0.8	11	0.04	0.17	4.96	5.14	0.9	13	0.05	2	228	230	0.9	12	0.05	0.55	33	33.6
12	**32**	**5**	**160**	120	2.8	8	0.13	0.09	0.81	1.03	2.8	32	0.13	0.9	100	101	2.9	13	0.18	0.8	244	245
13	**64**	**3**	**192**	160	11.2	5	0.58	0.23	1.64	2.45	9.6	47	0.51	0.6	10.4	11.4	10.6	5	0.8	0.2	4	5
14	**96**	**2**	**96**	240	24.5	5	1.48	0.25	0.04	1.77	21	96	1.34	2.16	87.5	91	24.8	96	2.44	0.66	35.5	38.6
15	**128**	**2**	**128**	370	43.2	5	3.21	0.53	0.1	3.84	35.8	128	2.7	1.3	66	70	43.2	128	6	2	73	81
16	**163**	**5**	**815**	550	69.8	6	6.3	3.4	12	21.7	57.5	128	5.3	7.7	524	537	71.6	22	16	4.6	37	57.6
17	**409**	**2**	**818**	2400	384	2	208	4	0.03	212	340	13	127	7.9	0.13	135	368	3	210	8	928	1146
18	**571**	**2**	**1042**	5000	827	5	2246	10	0.11	2256	663	427	1358	63.8	2.24	1424	813	5	2433	19	5	2457

Instead, we explain the intuition behind Thm. **??**, and show how it can be applied for MFR check.

Due to WRTO $>_R$, each rem_l comprises only X_{PI} variables. This is because under WRTO $>_R$, each gate output becomes a leading term of some polynomial in F'. However, as primary inputs cannot be gate outputs, they do not appear as a leading term of any polynomial. As a result, the computation $f \xrightarrow{F'_l \cup F'_0}_+ rem_l$ cancels all non primary input variables during polyomial division, and results in remainders composed of only primary inputs as their support. Hence, $V(rem_l) \subseteq \mathbb{F}_2^{|X_{PI}|}$. The variety of rem_l corresponds to the set of assignments to primary inputs X_{PI} (minterms) where the specification f agrees with the implementation C. Thus, the condition of Thm. **??** implies that the union of individual varieties of rem_l's comprises the entire input test set. Thus, for every minterm from the input test space, there exists an assignment $\delta[l]$ to W where f and C match. Consequently, there exists a function $U(X_{PI})$ that can be computed to rectify every error minterm.

Let $J_0^{X_{PI}} = \langle x^2 - x : x \in X_{PI} \rangle$. Then $V(J_0^{X_{PI}}) = \mathbb{F}_2^{|X_{PI}|}$. As the union of varieties corresponds to the variety of the product of corresponding ideals (cf. Section II), we have that: $\bigcup_{l=1}^{2^m} V(rem_l) = V_{\mathbb{F}_2}(\langle \prod_{l=1}^{2^m} rem_l \rangle + J_0^{X_{PI}})$. Therefore, to check for MFR at target W, the test "Is $\bigcup_{l=1}^{2^m} V(rem_l) = \mathbb{F}_2^{|X_{PI}|}$" can be performed by checking if $\prod_{l=1}^{2^m} rem_l \xrightarrow{J_0^{PI}}_+ 0$.

Example V.2. *Continuing on with the example from Ex.* **??**, *we demonstrate the rectification check at* $W = \{r_3, rr_3\}$.

Constructing the J'_l *ideals:*

- $J'_1 = \langle F'_1 \rangle$, *where* $F'_1[f'_W] = W + \delta[1] = W$
- $J'_2 = \langle F'_2 \rangle$, *where* $F'_2[f'_W] = W + \delta[2] = W + 1$
- $J'_3 = \langle F'_3 \rangle$, *where* $F'_3[f'_W] = W + \delta[3] = W + \beta = W + \alpha^{21}$
- $J'_4 = \langle F'_4 \rangle$, *where* $F'_4[f'_W] = W + \delta[4] = W + \beta^2 = W + \alpha^{42}$

Reducing the Spec $f : Z + A \cdot B$ *modulo these ideals, we get:*

- $rem_1 = f \xrightarrow{F'_1 \cup F'_0}_+ \alpha^{27}(a_2 b_1 b_2) + \alpha^{36}(a_2 b_2)$
- $rem_2 = f \xrightarrow{F'_2 \cup F'_0}_+ \alpha^{27}(a_2 b_1 b_2 + a_2 b_1) + \alpha^{36}(a_2 b_2)$
- $rem_3 = f \xrightarrow{F'_3 \cup F'_0}_+ \alpha^{27}(a_2 b_1 b_2)$
- $rem_4 = f \xrightarrow{F'_4 \cup F'_0}_+ \alpha^{27}(a_2 b_1 b_2 + a_2 b_1)$

When we compute the product $\prod_{l=1}^{4} rem_l$, *and reduce it by* $J_0^{X_{PI}}$ *in* $\mathbb{F}_{2^6}$ *with* $P_6(\alpha) = 0$, *we obtain the remainder 0, thus confirming that the target* $W = \{w_0 = r_3, w_1 = rr_3\}$ *indeed admits correction. Even though it is beyond the scope of this paper,* $W = a_2 b_1 b_2 + \beta \cdot a_2 b_2$ *is a polynomial which can be computed to rectify the circuit.*

VI. Experimental Results

We implement our approach as a custom software by integrating PolyBori [**?**] libraries, SINGULAR [**?**] libraries, and a custom high level finite field engine. We utilize PolyBori's reduction procedure to implement the division $f \xrightarrow{F'_l \cup F'_0}_+ rem_l$, $1 \leq l \leq 2^m$. The libraries from SINGULAR are used to compute $P_k(X)$ and to model composite field characteristics. The custom finite field engine is employed in modeling bit-vector and coefficient computations, and is utilized in the overall decision procedure. The experiments are performed on a 3.5GHz Intel(R) Core™ i7-4770K Quad-Core CPU with 32 GB RAM.

Table **??** presents the results of our approach when performing the MFR check on two modular multipliers (Mastrovito and Montgomery), and circuits that perform Point Addition over elliptic curves. The MFR check is performed against their respective polynomial specifications. These designs form the core components for performing encryption, decryption and authentication in Elliptic Curve Cryptography (ECC). These Benchmarks are taken from [**?**] and synthesized using the *abc* tool [**?**] with a two input gate library. We introduce bugs in the synthesized netlists by means of gate and wiring modifications at various topological levels: some closer to PIs, some in the middle of the circuit, and some near POs. The column BO for each benchmark corresponds to the total number of POs affected by the gate and wiring modifications.

In the results table, column 2 (n) denotes the datapath size for the corresponding benchmark, and the columns marked G denote the total number of gates for the respective synthesized netlists. In our experiments, we check the rectifiability of circuits, where the number of targets is chosen from $m = \{2, 3, 5, 7\}$. However, our approach doesn't constrain the number of targets to check for rectifiability. For cases where the targets admit rectification (Rows 1-10), the m targets are chosen to lie very close to the gate outputs where the bugs were introduced, while for the cases where the targets do not admit rectification (Rows 11-18), the m targets are chosen away from the cone of influence where bugs were introduced.

PBS denotes the time taken to build the ZDD models for the implementation and the specification of the corresponding benchmarks using the Polybori engine. Column AM denotes the maximum resident memory size recorded for these models, averaged across benchmarks for a given datapath size n. VMS denotes the time taken to perform verification, compute $P_k(X)$, and setup MFR check. The execution time required for RC (Thm. **??**) depends on the number (m) and selection of the targets in W, as well as the computed $P_k(X)$. These parameters result in different sizes of rem_l. The larger the size, the longer it takes to perform the rectification check. This is depicted in the table for Montgomery multipliers, where the rectification check for the 409-bit* circuit takes significantly longer than for the 571-bit† circuit.

We compare our approach against the cofactor reduction algorithm (Alg. 1) presented in [**?**]. We implemented the Alg. 1 of [**?**] using *abc/MiniSAT*. Empirical data shows that the rectification check for circuits beyond 16-bits timed out (3 hours). Specifically, the final UNSAT check on line 4 of the Alg. 1 is infeasible for the benchmarks presented in our results table, and hence the comparision is omitted.

VII. Conclusion

This paper presents an automated approach using techniques from symbolic computer algebra to determine multi-fix rectifiability of faulty finite field arithmetic circuits at a given set of targets. The underlying theory and algorithms are based on Gröbner basis reductions, the Strong Nullstellensatz principle, ideal membership testing, and finite fields. The efficiency of our approach is derived by interpreting the targets as a bit-vector and enabling word-level reasoning. We propose new mathematical insights to overcome the challenges associated with formulating the problem over a composite field. As future work, we are investigating subsequent computation of multi-fix rectification functions, also at the word-level. Further, we are also exploring techniques to improve the efficiency of rectification check implementation.

References

[1] 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.

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

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

[4] E. Biham, Y. Carmeli, and A. Shamir, "Bug Attacks," in *Proceedings on Advances in Cryptology*, 2008, pp. 221–240.

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

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

[11] J. von zur Gathen and D. Panario, "Factoring Polynomials over Finite Fields: A Survey," *J. Symbolic Computation*, vol. 31, pp. 3–17, 2001.

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

[13] M. Brickenstein and A. Dreyer, "Polybori: A framework for gröbner-basis computations with boolean polynomials," *J. Symb. Comput.*, 2009.

[14] 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.

[15] 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.

[16] R. Brayton and A. Mishchenko, "Abc: An Academic Industrial-strength Verification Tool," in *Proceedings of the 22Nd International Conference on Computer Aided Verification*, 2010.