I. Results from Prior NSF Support

• Awards CCF-1320335/CCF-1320385, *SHF:Small: Collaborative Proposal: Efficient Computer Algebra Techniques for Scalable Verification for Galois Field Arithmetic Circuits*. Investigators: Priyank Kalla and Florian Enescu. Award amounts: PI Kalla: \$205,071. PI Enescu: \$188,241. Award Duration: 8/1/2013 - 7/31/2016.

Intellectual Merit: The goal of this *recently concluded* project was to investigate algorithmic techniques for formal verification of *combinational arithmetic circuits* that implement computations over finite (Galois) fields of the type $\mathbb{F}_{2^k}$. Such circuits cater to applications in cryptography and error-correction codes. The operand-size k in such circuits is very large, ranging from $k = 163$ bits to 1024 bits and more. This makes verification extremely challenging; contemporary approaches do not scale beyond even 16-bit datapaths.

We have employed models from commutative algebra and finite field theory to verify such circuits. The functionality of the circuit is modeled as an ideal in polynomial rings $\mathbb{F}_{2^k}[x_1,\dots,x_n]$, and the Gröbner basis techniques are employed to reason about the correctness of (or bugs in) the design. Following are the intellectual contributions:

1) First, we addressed the problem of formal verification of Galois field circuits against a word-level polynomial specification. The problem is formulated as *ideal membership test* and solved using Gröbner basis techniques [1].
2) In another formulation, we have shown that the *geometry of elimination* – along with Gröbner basis techniques – can be employed efficiently to derive (or *abstract*) a canonical, word-level polynomial representation of the function implemented by a gate-level combinational circuit [2]. The application can be scaled to very large circuits by analyzing the circuit's topology [3].
3) The approach was further refined to verify bit-parallel finite-field circuits [4], based on k-cycle unrolling of the sequential circuit [5].

Broader Impacts: The PIs collaborations have positively impacted *theory, technology, application, as well as education*, in verification over finite fields using symbolic computation. The main challenge with Gröbner basis methods is to manage the computational complexity of the algorithms. Our work has shown that it is possible to overcome the high complexity of Gröbner basis techniques when applied to digital circuits. This can be achieved by extracting more information from the circuits under verification, and exploiting it to get more theoretical insights into the polynomial ideals, so as to improve the application of this approach. Following are the broader impacts:

1) (*Theory*) We have shown that by analyzing the topology of the given circuit, a *monomial order* can always be derived that renders the set of polynomials derived from the circuit already a Gröbner basis. This may obviate the need to compute a reduced Gröbner basis, thus avoiding an explosive computation [1].
2) (*Theory & Application*) Gröbner basis application on datapath circuits helps to identify the structure and symmetry inherent in the problem, which enhances the scalability of abstraction and verification [3].
3) (*Technology*) Gröbner basis also enables raising the abstraction level from bits to words; thus enabling hierarchy abstraction in verification [3] [4].
4) (*Technology & Application*) A Galois field datapath verification tool, along with datapath design benchmarks, based on these concepts has been released to our community [6].
5) (*Education*) PI Kalla was invited to present three tutorials on these topics: i) BIRS workshop on *Theoretical Foundations of Applied SAT Solving* 2014, covering Gröbner bases concepts viz-a-viz SAT and verification; ii) the joint session of FMCAD and SAT 2015 conferences [tutorial slides available at [7]]; and at iii) High-Level Design Validation & Test Workshop 2016. PI Kalla has also established course-work at his institution on this topic of Hardware Verification using Symbolic Computation.

• Award CCF-1619370, *SHF: Small: New Directions in Gröbner Basis based Verification using Logic Synthesis Techniques*. PI: Priyank Kalla; Award amount \$390,999; Award period 08/01/2016-07/31/2019.

In the previous project, we had researched algorithms for formal verification of finite field arithmetic circuits. This project explores efficient implementation of symbolic computing algorithms over *Boolean rings* using traditional logic synthesis techniques. Application of symbolic computing algorithms, particularly Gröbner basis reduction via polynomial division, on circuits requires computation over Boolean rings. These computations resemble *Boolean function decomposition of the AND-XOR variety*, so they can be implemented using: i) Zero-suppressed BDDs and the unate cube set algebra [8], [9]; and ii) And-Invert-Graphs (AIGs) and SAT solvers [10]. The project is currently underway, and one publication has been submitted for review [11].

PIs Kalla and Enescu have a long history of collaboration and have jointly coauthored the following other publications [12], [13], [14], [15], [16], [17], [18], [19], [20], [21].

This Proposal: This proposal investigates an altogether new concept – that of sequential circuit verification (sequential equivalence and model checking) applied *purely at the word-level*. For this purpose, we investigate techniques from algebraic geometry to *discover new word-level abstractions of the state-space of sequential circuits* including: *word-level reachability analysis, word-level Craig interpolants in algebraic geometry, word-level UNSAT cores of sets of polynomials,* etc. In this regard, this proposal is very different from the PI's past projects.

II. Introduction

With the increasing size of integrated circuits, sequential circuit designers face complicated problems of design errors in specification models and implementations. These errors are usually modeled as "bad" states, and the circuits/functional components are modeled as finite state machines (FSMs). Once state reachability is analyzed, the existence of errors can be identified by checking whether "bad" states are *reachable* from certain initial states. Temporal logic *model checking* (MC) formulations and solvers are often used for this purpose. Once the designs and specification models are validated using model checking, optimized implementations of sequential circuits are synthesized. A subsequent problem then needs to be solved to prove that the sequential circuit implementations are equivalent to the original specification models, i.e. *Sequential Equivalence Checking (SEC).*

Reachability analysis forms the backbone of most sequential verification techniques. As the state-space of FSMs increases, reachability analysis forms a fundamental bottleneck in sequential verification. Contemporary approaches employ various techniques to overcome this state-explosion problem: i) Bounded model checking [22] traverses the FSMs for a fixed number of steps k (k-BMC) to check whether a property violation can occur in k or fewer steps. ii) Analyze over-approximations (or *abstractions*) of the state-space. Abstraction proves properties on the system by first simplifying it, and when the abstraction does not satisfy the same properties as the original one, a process of refinement is needed. *Craig interpolants* [23] [24] are used as a means of abstraction and reachability analysis for linear temporal logic (LTL) model checking [25]. Similarly, counterexample guided abstraction refinement (CEGAR) [26] uses proofs of unsatisfiability (UNSAT cores) to refine the abstractions. iii) the recent breakthrough method of [27] where the set of over-approximations to forward reachable states are refined with inductive constraints – property directed reachability (PDR).

While the above techniques have made significant strides in sequential verification, numerous practical instances remain unsolved. One issue with all of the above techniques is that they mostly use bit-level constraints to model the transition relations and sets of states. We have access to instances from our industrial collaborators (Calypto Design Systems, now Mentor Graphics) where the designs are expressed at the level of bit-vector words (e.g. Matlab code, Verilog), and these word-level abstractions are rarely exploited in verification. The problem is further exacerbated when there are arithmetic operators on word-level operands embedded in the control logic. While attempts have been made towards word-level predicate abstraction [28] [29] [30], *using a purely word-level representation of the state-space, the properties and their abstractions have not been fully explored as another dimension in improving sequential verification.*

There are several advantages in exploiting word-level information for verification. A number of designs have their datapaths and/or system-level models described as word-level RTL models. Exploiting word-level instead of bit-level information is one way of abstraction – a key technique to reduce the state space of a FSM. It has the effect of combining sets of states with similar properties. During reachability analysis, if we use bit-level variables to represent the states, the representations may become too large to handle. However, when a "bundle" of bit-level variables are represented as *only one word-level variable*, the set of reachable states can be represented by a word-level constraint expression; which may lower verification complexity.

Overall Research Objective: In this proposal we investigate a set of new, promising approaches for *word-level representation, reachability analysis and abstraction* with applications to SEC, k-BMC and abstraction-based unbounded LTL model checking (A-UMC). Our techniques operate at the word-level and they are based largely on the concepts from *algebraic geometry*.

For word-level SEC, we are given two designs, or their corresponding Mealy/Moore FSMs $\mathcal{M}_1, \mathcal{M}_2$, along with their initial starting states S_0^1, S_0^2. We wish to prove the absence of a sequence of inputs (string) that distinguishes the initial states [31] [32]. Fundamentally, this requires the construction of a product machine; and the main research problem relates to that of performing *FSM traversal [33] but at the word-level*. Analogously, in the case of MC, the problem is setup w.r.t. a FSM $\mathcal{M}$, a set of initial states S_0 and a set of property states p. Techniques are to be researched that verify that there exist no sequence of transitions from an initial state to a non-property state ("bad" state). These problems have to be solved in the context of word-level verification – i.e. data representation, abstraction (Craig interpolation, UNSAT cores) and algorithm execution has to be carried out at word-level.

The Algebraic Geometry Model and Rationale: We will represent the FSMs – the transition relations – by means of a set of multi-variate polynomials with coefficients from the finite (Galois) field $\mathbb{F}_{2^k}$ of 2^k elements, i.e. polynomials in the ring $\mathbb{F}_{2^k}[x_1, \ldots, x_d]$. Each state of a FSM is identified with a Boolean assignment to a set of k-bit state register variables $S = \{s_0, \ldots, s_{k-1}\}$. Therefore, we can consider each (k-bit) state as a word-level element S of the finite field $\mathbb{F}_{2^k}$. Algorithms can directly operate on polynomials in word-level variable S.

Boolean functions with k-bit inputs and k-bit outputs $f : \mathbb{B}^k \to \mathbb{B}^k, \mathbb{B} = \{0,1\}$ can be construed as functions $f : \mathbb{F}_{2^k} \to \mathbb{F}_{2^k}$. It is well-known that over the finite field ($\mathbb{F}_q$) of q elements, *every function* $f : \mathbb{F}_q \to \mathbb{F}_q$ is a polynomial function [34]. Moreover, there exists a unique canonical polynomial $\mathcal{F}$ that describes f. This implies that one can derive a *canonical, polynomial abstraction* of the function as $Z = \mathcal{F}(A)$ where Z, A are word-level symbols representing k-bit operands. The concept also generalizes to functions with different input/output bit-vector sizes, i.e. functions of the type $f : \mathbb{B}^n \to \mathbb{B}^m$, modeled as polynomials over $f : \mathbb{F}_{2^k} \to \mathbb{F}_{2^k}$, where $k = LCM(n,m)$ [34]. This implies that the FSM's transition relations can be represented as polynomial functions (ideals) in $\mathbb{F}_{2^k}$, and values of state variables can be represented as solutions to these polynomials (variety of the ideal). Subsequently, the ideal-variety correspondences in algebraic geometry can be applied to implement symbolic reasoning about state reachability. Moreover, as $\mathbb{F}_2 \subset \mathbb{F}_{2^k}$, our model provides a single, unified and bit-precise representation for both bit-level ($\mathbb{F}_2$) and word-level ($\mathbb{F}_{2^k}$) constraints.

The decision and abstraction procedures in our setting will rely on the theory and technology of *Gröbner bases (GB)*. GB-based algebraic reasoning is very powerful; in fact it is known to be strictly stronger than resolution [35]. Therefore, in light of the above discussion, using concepts from algebraic geometry and Gröbner bases over $\mathbb{F}_{2^k}$, we can introduce another dimension of word-level abstraction to the techniques in sequential verification.

The core focus of this research: The use of algebraic geometry for sequential circuit verification and symbolic model checking has been presented before. Avrunin presented the concept of symbolic MC using algebraic geometry in [36]. Later, in [37], Vardi presented GB-algorithms for CTL, LTL and bounded MC over Boolean rings. However, these approaches are a straight-forward transformation of the problem to *bit-level* Boolean GB engines which are used in lieu of BDDs or SAT solvers. All the concepts of word-level reachability, abstraction-refinement using interpolation or UNSAT cores, etc., that we desire was not the focus of [36] [37].

In this research, we will focus on three specific research problems:

1) How to perform reachability analysis at the word-level? This of course forms the foundation for word-level SEC and MC.
2) What is the analog of Craig interpolation in an algebraic geometry setting (theoretical challenge), and how can we compute various interpolants using GB-engines efficiently (practical implementation)? This will enable word-level abstraction of the state-space.
3) Explore the concept, and the computation, of unsatisfiable (UNSAT) cores of a set of polynomials. Many abstraction-refinement techniques make use of UNSAT cores of CNF-formulas in MC. Therefore, our investigation of UNSAT cores of polynomial ideals will enable similar applications at word-level.

We assume that the reviewer is familiar with basic concepts of sequential equivalence [32] [31] [38] [39] [40] [41], model checking [42] [43] [44] [22], model checking with abstraction using interpolation [25] [29], and the various ways in which UNSAT cores/proofs are used for abstraction/refinement and verification [45] [26]. In the manuscript, we omit the description of these concepts. We concentrate only the aforementioned three problems, as the context in which they are applied should now be evident.

Proposal Organization: The rest of the proposal is organized as follows: The following Subsection describes the intellectual merit of the proposed work. Section III covers preliminary concepts and notation. Section IV describes the basic concept of word-level FSM traversal and the research problems to investigate. Section V explores the algebraic geometry analog of Craig Interpolants. Section VI describes algorithmic techniques to derive UNSAT cores of polynomial ideals. Section VII puts the whole research in perpective, by demonstrating with the help of an example how abstraction via unsat cores in algebraic geometry can simplify BMC. Section VIII addresses the broader impact of the proposed activity and concludes the proposal.

A. Intellectual Merit, Novelty and Challenge

This research should not be considered purely as an alternative to contemporary methods, but rather as one that adds a new dimension of word-level abstraction to SEC and MC. Drawing inspirations from the related work in the Boolean domain, we will develop the algebraic geometry analog of (word-level) FSM traversal, Craig interpolants and UNSAT cores. To the best of our knowledge, these concepts have not been developed yet. Subsequently, we will engineer efficient implementation of decision and abstraction procedures using Gröbner basis techniques. The intellectual merit and challenge of these techniques are described below:

- We wish to represent the FSMs – transition relations, sets of states (initial, reachable and property states) – using a set of word-level polynomial constraint expressions (i.e. polynomial ideals) in the rings $\mathbb{F}_{2^k}[x_1,\ldots,x_d]$. Using the ideal-variety correspondences in algebraic geometry, we will then discover new algorithms for *word-level reachability analysis*, and employ them in SEC and MC.
- As abstraction techniques will still be needed to overcome the state explosion problem, we will further investigate concepts related to *Craig interpolants in algebraic geometry* to produce over-approximate image operators in A-UMC. This problem requires that we study the geometry of elimination to devise algorithmic techniques for their computation. Interpolants may not be unique – so we wish to research GB-techniques to compute many different interpolants and possibly classify them as relatively "stronger" or "weaker" (details in the sequel).
- Abstraction-refinement techniques make use of resolution proofs and UNSAT cores to derive interpolants and/or to refine the abstractions in model checking. So we will also develop algorithms to identify *unsatisfiable cores of polynomial ideals* in the context of Nullstellensatz [46].
- *Gröbner basis* (GB) algorithms will be utilized as decision and abstraction procedures to solve the above problems. However, the GB problem exhibits high-complexity [47]; over finite fields ($\mathbb{F}_{2^k}[x_1,\ldots,x_d]$) it is bounded exponentially ($2^{k^{O(d)}}$) in the input data [48], [49]. An important challenge is to overcome the GB complexity. In our prior work [1], [3], we have shown that when applied to combinational circuits, domain-specific knowledge from the topology of the circuit

can be exploited to improve GB-based verification. Similar concepts will be researched to scale GB-based computation for sequential verification.

III. Preliminaries: Notation and Background Results

Finite fields and the Boolean domain: Let $\mathbb{B} = \{0,1\}$ denote the Boolean domain, $\mathbb{F}_2$ the finite field of 2 elements ($\mathbb{B} \equiv \mathbb{F}_2$), and $\mathbb{F}_q = \mathbb{F}_{2^k}$ the finite field of $q = 2^k$ elements. Let $R = \mathbb{F}_{2^k}[x_1,\dots,x_n]$ denote the polynomial ring over variables $x_1,\dots,x_n$ and coefficients in $\mathbb{F}_{2^k}$. The field $\mathbb{F}_{2^k}$ is constructed as $\mathbb{F}_{2^k} = \mathbb{F}_2[X] \pmod{P(X)}$, where $P(X)$ is an irreducible polynomial over $\mathbb{F}_2$. Let $P(\alpha) = 0$. Any element $A \in \mathbb{F}_{2^k}$ can be represented as $A = \sum_{i=0}^{k-1} a_i \alpha^i$, where $a_i \in \mathbb{F}_2$. Since $\mathbb{F}_{2^k}$ is a k-dimensional extension of $\mathbb{F}_2$, we have that $\mathbb{F}_{2^k} \supset \mathbb{F}_2$ and all $\mathbb{F}_{2^k}(k \geq 1)$ are of characteristic 2. We use $+, \cdot$ to denote addition and multiplication, respectively, in the ring R; $\vee$ and $\wedge$ to denote Boolean OR and AND operations; $\neg$ denotes Boolean complement.

Polynomials: A polynomial $f \in R$ is written as a finite sum of terms $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$. Here $c_1,\dots,c_t$ are coefficients and $X_1,\dots,X_t$ are monomials, i.e. power products of the type $x_1^{e_1} \cdot x_2^{e_2} \cdots x_n^{e_n}$, $e_i \in \mathbb{Z}_{\geq 0}$. To systematically manipulate the polynomials, a monomial order $>$ (also called a term order) is imposed on the ring 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. We also denote tail(f) = $f - lt(f) = c_2X_2 + \cdots + c_tX_t$.

When a circuit is modeled with polynomials, every Boolean logic gate operator in the circuit is mapped to a polynomial function over $\mathbb{F}_2$ $(\subset \mathbb{F}_{2^k})$:

$$\begin{aligned} &NOT: \ \neg a \rightarrow a+1 \pmod 2; \quad AND: \ a \wedge b \rightarrow a \cdot b \pmod 2; \\ &OR: \ a \vee b \rightarrow a+b+a\cdot b \pmod 2; \quad XOR: \ a \oplus b \rightarrow a+b \pmod 2 \end{aligned} \tag{1}$$

Using these mappings we can write Boolean functions as polynomials over $\mathbb{F}_{2^k}$.

Polynomial division: Let $F = \{f_1,\dots,f_s\}$ be a set of polynomials in R. A polynomial f can be reduced (divided) modulo F to obtain remainder r using a generalization of long division. This reduction is denoted $f \xrightarrow{F}_+ r$, and the remainder r has the property that no term in r is divisible by the leading term of any polynomial f_i in F.

Ideals, Varieties and Gröbner Basis: Given a set of polynomials $F = \{f_1,\dots,f_s\}$, denote the ideal J generated by F as $J = \langle F \rangle = \langle f_1,\dots,f_s \rangle = \{\sum_{i=1}^{s} h_i \cdot f_i : \ h_i \in R\}$.

We have to consider the set of solutions to the system of polynomials equations $f_1 = \cdots = f_s = 0$. The set of all solutions to a given system of polynomial equations $f_1 = \cdots = f_s = 0$ is called the *variety*, which depends upon the ideal $J = \langle f_1,\dots,f_s \rangle$ generated by the polynomials. The variety is denoted by $V(J) = V(f_1,\dots,f_s)$ and defined as:

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

where $\mathbf{a} = (a_1,\dots,a_n) \in \mathbb{F}_q^n$ denotes a point in the affine space.The ideal J may have many different generator polynomials, i.e. it is possible to have $J = \langle f_1,\dots,f_s \rangle = \langle g_1,\dots,g_t \rangle = \cdots = \langle h_1,\dots,h_r \rangle$. A Gröbner basis G of ideal J is one such set of polynomials $G = GB(J) = \{g_1,\dots,g_t\}$ that is a canonical representation of the ideal.

Definition 3.1: [**Gröbner Basis**] [46]: For a monomial ordering $>$, a set of non-zero polynomials $G = \{g_1, g_2, \cdots, g_t\}$ contained in an ideal J, is called a Gröbner basis for J iff $\forall f \in J$, $f \neq 0$ there exists $i \in \{1,\cdots,t\}$ such that $lm(g_i)$ divides $lm(f)$; i.e., $G = GB(J) \Leftrightarrow \ \forall f \in J : f \neq 0, \exists g_i \in G : lm(g_i) \mid lm(f)$.

The famous Buchberger's algorithm [50], which is now textbook knowledge [46], is used to compute a GB. Operating on input $F = \{f_1,\dots,f_s\}$, and subject to the imposed term order $>$, the algorithm computes $G = GB(J) = \{g_1,\dots,g_t\}$. The Gröbner basis also provides a decision procedure for ideal membership: To test if any polynomial $f \in J$, we compute $G = GB(J)$ and check if $f \xrightarrow{G}_+ 0$?

In this work, we will consider Gröbner bases as a quantification procedure for polynomial abstraction [49], as well as a decision procedure for ideal membership test. For this purpose, we introduce the concepts of *vanishing polynomials*, and *elimination ideals*.

Fermat's little theorem over $\mathbb{F}_q$: For any $\alpha \in \mathbb{F}_q, \alpha^q = \alpha$. Therefore, the polynomial $x^q - x$ vanishes $(= 0)$ over $\mathbb{F}_q$, and is called the vanishing polynomial of $\mathbb{F}_q$. Denote by $J_0 = \langle x_1^q - x_1, \dots, x_n^q - x_n \rangle$ the ideal of all vanishing polynomials in R.

Elimination Ideals: Gröbner bases can be used to *eliminate* (i.e. quantify) variables from an ideal. Given ideal $J = \langle f_1, \dots, f_s \rangle \subset R$, the l^{th} elimination ideal J_l is the ideal of $\mathbb{F}_q[x_{l+1}, \dots, x_n]$ defined by $J_l = J \cap \mathbb{F}_q[x_{l+1}, \dots, x_n]$. Variable elimination can be achieved by computing a Gröbner basis of J w.r.t. elimination orders:

Theorem 3.1: (Elimination Theorem [46]) Let $J \subset \mathbb{F}_{2^k}[x_1, \dots, x_n]$ be an ideal and let G be a Gröbner basis of J with respect to a lexicographic (LEX) ordering where $x_1 > x_2 > \cdots > x_n$. Then for every $0 \le l \le n$, the set $G_l = G \cap \mathbb{F}_{2^k}[x_{l+1}, \dots, x_n]$ is a Gröbner basis of the l-th elimination ideal J_l.

Example 3.1: Consider polynomials $f_1 : x^2 - y - z - 1;\ f_2 : x - y^2 - z - 1;\ f_3 : x - y - z^2 - 1$ and ideal $J = \langle f_1, f_2, f_3 \rangle \subset \mathbb{C}[x, y, z]$. Gröbner basis $G = GB(J)$ w.r.t. LEX term order equals to $g_1 : x - y - z^2 - 1;\ g_2 : y^2 - y - z^2 - z;\ g_3 : 2yz^2 - z^4 - z^2;\ g_4 : z^6 - 4z^4 - 4z^3 - z^2$. From observation, we find that the polynomial g_4 only contains variable z (x, y eliminated), and polynomials g_2, g_3, g_4 only contain variables y, z (x eliminated). According to theorem 3.1, $G_1 = G \cap \mathbb{C}[y, z] = \{g_2, g_3, g_4\}$ is the Gröbner basis of the 1^{st} elimination ideal of J and $G_2 = G \cap \mathbb{C}[z] = \{g_4\}$ is the 2^{nd} elimination ideal of J, respectively.

Application to word-level abstraction of the state-space: Assume that we can represent the transition relation of a machine using an ideal $J + J_0$ in $\mathbb{F}_{2^k}$, where J is the ideal generated by the polynomials corresponding to circuit gates and J_0 is the ideal of vanishing polynomials. Using an elimination order with *"bit-level variables"* $>$ *"present-state word S"* $>$ *"next-state word T"*, compute a Gröbner basis $G = GB(J + J_0)$. In the elimination ideals, we should find polynomials in word-level variables encoding the states of the machine. Based on this intuition, we are now ready to describe word-level reachability analysis.

IV. Research Problems: Word-Level Reachability Analysis of FSMs

In this section, we will describe the basic concepts of word-level FSM traversal using algebraic geometry. Based on our preliminary investigations, we show that this concept is feasible, and does present the opportunity to analyze the state-space at word-level. Subsequently, we describe the research problems that we need to investigate to make the approach practical.

Definition 4.1: A Mealy finite state machine is a n-tuple $\mathcal{M} = (\Sigma, O, S, S^0, \Delta, \Lambda)$ where: i) Σ is the input label, O is the output label; ii) S is the set of states, and $S^0 \subseteq S$ is the set of initial states; iii) $\Delta: \ S \times \Sigma \to S$ is the next state transition function; iv) $\Lambda: \ S \times \Sigma \to O$ is the output function.

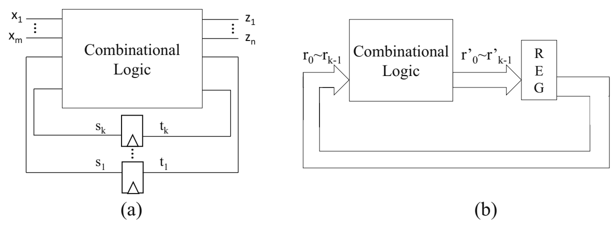

Fig. 1: FSM models of sequential circuits

Typical sequential circuits are depicted as in Fig.1(a). Primary inputs are denoted $x_1, \dots, x_m \in \Sigma$, and the primary outputs as $z_1, \dots, z_n \in O$. Signals $s_1, \dots, s_k$ are the present state (PS) variables, $t_1, \dots, t_k$ the next state (NS) variables. We can define 2 k-bit words denoting the PS/NS variables as there are k flip-flops in the datapath: $S = (s_1, \dots, s_k),\ T = (t_1, \dots, t_k)$. Transition function at the bit-level are defined as $\Delta_i : t_i = \Delta_i(s_1, \dots, s_k, x_1, \dots, x_m)$.

When the primary outputs of the FSM only depend on the present states, i.e. $\Lambda: S \to O$, then the FSM is called a Moore machine. For example, in some cases k-bit arithmetic computations are implemented as Moore machines, where input operands are loaded into register files R and the FSM is executed for k clock cycles. We can simplify these to the model in Fig.1(b). While we target both FSMs, we will consider only Mealy FSMs in the manuscript.

Conceptually, the state-space of a FSM is traversed in a breadth-first search (BFS) manner, as shown in Algorithm 1. The algorithm operates on the FSM $\mathcal{M} = (\sum, O, S, S^0, \Delta, \Lambda)$ underlying a sequential circuit. Starting from the initial state $from^i = S^0$, the algorithm computes the states reachable in 1-step from $from^i$ in each iteration. In line 4 of algorithm 1, the *image computation* is used to compute the reachable states in 1-step.

Algorithm 1: BFS Traversal for FSM Reachability

```
Input: Transition functions $\Delta$, initial state $S^0$
1 $from^0 = reached = S^0$;
2 repeat
3     $i \leftarrow i+1$;
4     $to^i \leftarrow \text{Img}(\Delta, from^{i-1})$;
5     $new^i \leftarrow to^i \cap \overline{reached}$;
6     $reached \leftarrow reached \cup new^i$;
7     $from^i \leftarrow new^i$;
8 until $new^i == 0$;
9 return $reached$
```

Let us now see how such a breadth-first traversal could be performed at word-level. We will describe the application on the FSM circuit of Fig. 2. In Line 1 of the BFS algorithm, assume that the initial state is S_3 in Fig.2(b), which is encoded as $S_3 = \{11\}$.

Our objective is to model the transition functions Δ as a polynomial ideal J, and to perform the image computations (Algorithm 1, line 4) using Gröbner bases over Galois fields. *This requires to perform quantifier elimination; which can be accomplished using the GB computation over $\mathbb{F}_{2^k}$ using elimination ideals* [49]. Finally, the set union, intersection and complement operations are also to be implemented in algebraic geometry.

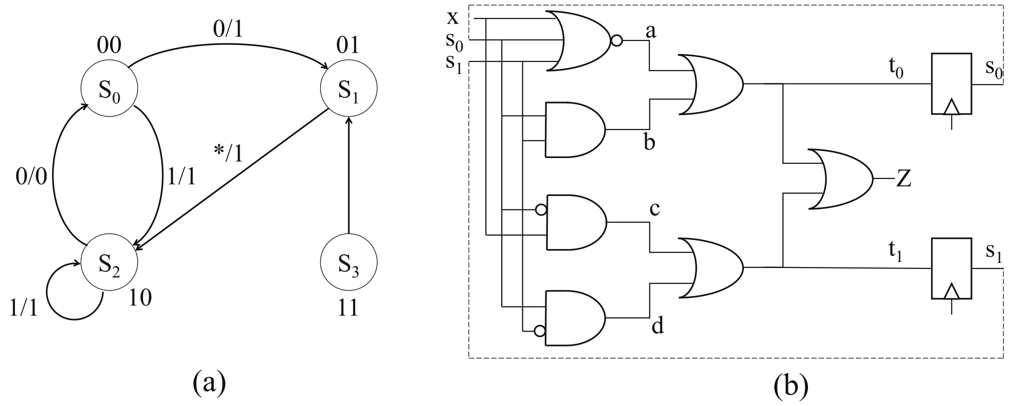

Fig. 2: The example FSM and the gate-level implementation.

FSM Traversal at word-level over $\mathbb{F}_{2^k}$: The state transition graph (STG) shown in Fig.2(a) uses a 2-bit Boolean vector to represent 4 states $\{S_0, S_1, S_2, S_3\}$. We map these states to elements in $\mathbb{F}_{2^2}$, where $S_0 = 0, S_1 = 1, S_2 = \alpha, S_3 = \alpha + 1$. Here, we take $P(X) = X^2 + X + 1$ as the irreducible polynomial to construct $\mathbb{F}_4$, and $P(\alpha) = 0$ so that $\alpha^2 + \alpha + 1 = 0$.

Initial state: In Line 1 of the algorithm, the initial state is specified by means of a corresponding polynomial $f = \mathcal{F}(S) = S - 1 - \alpha$. Notice that if we consider the ideal generated by the initial state polynomial, $I = \langle f \rangle$, then its variety $V(I) = 1 + \alpha$, corresponds to the state encoding $S_3 = \{11\} = 1 + \alpha$, where a polynomial in word-level variable S encodes the initial state.

Set operations: In Lines 5 and 6 of Alg. 1, we need **union**, **intersection** and **complement** of varieties over $\mathbb{F}_{2^k}$, for which we again use algebraic geometry concepts.

Definition 4.2: (**Sum/Product of Ideals** [46]) If $I = \langle f_1, \dots, f_r \rangle$ and $J = \langle g_1, \dots, g_s \rangle$ are ideals in R, then the **sum** of I and J is defined as $I + J = \langle f_1, \dots, f_r, g_1, \dots, g_s \rangle$. Similarly, the **product** of I and J is $I \cdot J = \langle f_i g_j \mid 1 \le i \le r, 1 \le j \le s \rangle$.

Theorem 4.1: If I and J are ideals in R, then $\mathbf{V}(I+J) = \mathbf{V}(I) \bigcap \mathbf{V}(J)$ and $\mathbf{V}(I \cdot J) = \mathbf{V}(I) \bigcup \mathbf{V}(J)$.

In Line 5 of Alg. 1, we need to compute the complement of a set of states. Assume that J denotes a polynomial ideal whose variety $V(J)$ denotes a set of states. We require the computation of another polynomial ideal J', such that $V(J') = \overline{V(J)}$. We show that this computation can be performed using the concept of **ideal quotient**:

Definition 4.3: (**Quotient of Ideals**) If I and J are ideals in a ring R, then $I : J$ is the set $\{f \in R \mid f \cdot g \in I, \forall g \in J\}$ and is called the **ideal quotient** of I by J.

We can now obtain the complement of a variety through the following results which are stated below:

Theorem 4.2: Let J be an ideal generated by a single univariate polynomial in variable x over $\mathbb{F}_{2^k}[x]$, and let the vanishing ideal $J_0 = \langle x^{2^k} - x \rangle$. Then

$$V(J_0 : J) = V(J_0) - V(J),$$

where all the varieties are considered over the field $\mathbb{F}_{2^k}$.

Let $J_0 = \langle x_1^{2^k} - x_1, \dots, x_n^{2^k} - x_n \rangle$ denote the ideal of all vanishing polynomials in $\mathbb{F}_{2^k}$. Then, we have $V(J_0) = (\mathbb{F}_{2^k})^n$; i.e. the variety of vanishing ideal contains all possible valuations of variables, so it constitutes the **universal set**. Subsequently, based on Theorem 4.2, the **absolute complement** $V(J')$ of a variety $V(J)$ can be computed as:

Corollary 4.1: Let J' be an ideal computed as $J' = J_0 : J$. Then

$$V(J') = \overline{V(J)} = V(J_0 : J)$$

With Corollary 4.1, we are ready to demonstrate the concept of word-level FSM traversal over $\mathbb{F}_{2^k}$ using algebraic geometry. The algorithm is given in Alg. 2. Note that in the algorithm, $from^i, to^i, new^i$ are *univariate polynomials in variables S or T* only, due to the fact that they are the result of a GB computation with elimination term order, where the bit-level internal variables are abstracted and quantified away.

Algorithm 2: Algebraic Geometry based Traversal Algorithm

```
Input: The circuit's characteristic polynomial ideal $J_{ckt}$, initial state polynomial $\mathcal{F}(S)$, and term order: bit-level variables $x, s, t$ > PS
       word $S$ > NS word $T$
1  $from^0 = reached = \mathcal{F}(S)$;
2  repeat
3      $i \leftarrow i + 1$;
4      $G \leftarrow$GB($\langle J_{ckt}, J_0, from^{i-1} \rangle$);
5      $\langle to^i \rangle \leftarrow G \cap \mathbb{F}_{2^k}[T]$;
6      $\langle new^i \rangle \leftarrow \langle to^i \rangle + (\langle T^{2^k} - T \rangle : \langle reached \rangle)$;
7      $\langle reached \rangle \leftarrow \langle reached \rangle \cdot \langle new^i \rangle$;
8      $from^i \leftarrow new^i(S \setminus T)$;
9  until $\langle new^i \rangle == \langle 1 \rangle$;
10 return $\langle reached \rangle$
```

Example 4.1: We apply Algorithm 2 to the example shown in Fig. 2 to execute the FSM traversal. Let the initial state $from^0 = \{00\}$ in $\mathbb{B}$ or $0 \in \mathbb{F}_4$. Polynomially, it is written as $from^0 = S - 0$. In the first iteration, we compose an elimination ideal $J_{ckt} = \langle f_1, f_2, f_3, f_4 \rangle$ where:

$$f_1 : t_0 - (xs_0s_1 + xs_0 + xs_1 + x + s_0 + s_1 + 1); \quad f_2 : t_1 - (xs_0 + x + s_0s_1 + s_0); \quad f_3 : S - s_0 - s_1\alpha; \quad f_4 : T - t_0 - t_1\alpha$$

and the vanishing polynomials $J_0 = \langle f_5, f_6, \dots, f_{11} \rangle$, where:

$$f_5 : x^2 - x; \quad f_6 : s_0^2 - s_0, \quad f_7 : s_1^2 - s_1$$

$$f_8 : t_0^2 - t_0, \quad f_9 : t_1^2 - t_1; \quad f_{10} : S^4 - S, \quad f_{11} : T^4 - T$$

Compute $G = GB(J)$ for $J = J_{ckt} + J_0 + \langle from^0 \rangle$, with an elimination term order

$$\underbrace{\{x, s_0, s_1, t_0, t_1\}}_{\text{all bit-level variables}} > \underbrace{S}_{\text{(PS word)}} > \underbrace{T}_{\text{(NS word)}} .$$

The resulting GB G contains a polynomial generator with only T as the variable. In Line 5, assign it to next state

$$to^1 = T^2 + (\alpha + 1)T + \alpha.$$

Note that the roots or variety of $T^2 + (\alpha + 1)T + \alpha$ is $\{1, \alpha\}$, denoting the states $\{01, 10\}$.

Since the formerly reached state "*reached* $= T$", its complement is computed using Corollary 4.1

$$\langle T^4 - T \rangle : \langle T \rangle = \langle T^3 + 1 \rangle.$$

$V(\langle T^3 + 1 \rangle) = \{1, \alpha, \alpha + 1\}$ denoting the states $\{01, 10, 11\}$. Then the newly reached state set in this iteration is

$$\langle T^3 + 1, T^2 + (\alpha + 1)T + \alpha \rangle = \langle T^2 + (\alpha + 1)T + \alpha \rangle$$

We add these states to formerly reached states: $reach = \langle T \cdot T^2 + (\alpha + 1)T + \alpha \rangle = \langle T^3 + (\alpha + 1)T^2 + \alpha T \rangle$, (i.e. states $\{00, 01, 10\}$). We update the present states for next iteration: $from^1 = S^2 + (\alpha + 1)S + \alpha$.

In the second iteration, we compute the reduced GB with the same term order for ideal $J = J_{ckt} + J_0 + \langle from^1 \rangle$. It includes a polynomial generator $to^2 = T^2 + \alpha T$ denoting states $\{00, 10\}$. The complement of *reached* is computed using ideal quotient:

$$\langle T^4 - T \rangle : \langle T^3 + (\alpha + 1)T^2 + \alpha T \rangle = \langle T + 1 + \alpha \rangle$$

(i.e. the state $\{11\}$). We compute the newly reached state: $\langle T^2 + \alpha T, T + 1 + \alpha \rangle = \langle 1 \rangle$.

Since the GB contains the unit ideal, it means the newly reached state set is empty, thus a fixpoint has been reached. The algorithm terminates and returns $reached = \langle T^3 + (\alpha + 1)T^2 + \alpha T \rangle$. As a Gröbner basis of the elimination ideal, *reached* canonically encodes the final reachable state set, as the variety of $\langle T^3 + (\alpha + 1)T^2 + \alpha T \rangle = \{00 = 0, 01 = 1, 10 = \alpha\}$.

Research Problem 4.1: *Algorithm 2 is our preliminary attempt at word-level BFS traversal. The correctness of the approach (Theorem 4.2 and Corollary 4.1) is restricted to elimination ideals consisting of univariate polynomials in one state variable word T. This is too restrictive, as many circuits may have multiple state-register words.*

The PIs will work together to generalize this approach to polynomials in multiple word-level variables.

Research Problem 4.2: *Once this theory is formalized, an efficient implementation of Alg. 2 is required. The approach also needs to be made scalable. The most computationally complex operation of the algorithm is the GB($\langle J_{ckt}, J_0, from^{i-1} \rangle$) computation in line 4. We have to investigate techniques to make this computation more efficient.*

We describe our line of investigation to address this issue. In [1], we had shown that for combinational circuits, it is possible to analyze the topology of the circuit to derive a term order $>$, which renders the set of polynomials themselves a Gröbner basis. This term order is derived by performing a reverse topological traversal of the circuit. It is thus named the "reverse topological order (RTO)." By ordering the variables in RTO, and imposing a LEX term order on the ring, the polynomials of the circuit already constitute a GB, and the expensive GB computation is avoided.

The effect of RTO on combinational circuits is that the output variable of each gate x_i becomes the leading term of the each polynomial f_i. Since each gate output is a unique variable, and the circuit is acyclic, all polynomials have relatively-prime leading terms. When all the polynomials in the generating set have relatively-prime leading terms, then they already constitute a Gröbner basis [51].

Unfortunately, for sequential circuits, we notice that the polynomials do not constitute a GB. Consider again the circuit of Fig. 2. Using RTO, i.e. LEX term order with: $(t_0,t_1) > (a,b,c,d) > T > (x,s_0,s_1)$, write the polynomial generators of J_{ckt}:

$$f_1 : a+xs_0s_1+xs_0+xs_1+x+s_0s_1+s_0+s_1+1 \quad f_2 : b+s_0s_1 \quad f_3 : c+x+xs_0;$$

$$f_4 : d+s_0s_1+s_0 \quad f_5 : t_0+ab+a+1 \quad f_6 : t_1+cd+c+d \quad f_7 : t_0+t_1\alpha+T$$

From observation, there is only one pair of polynomials (f_5, f_7) that have the same leading terms $= t_0$. All other polynomials have leading terms that are relatively prime. This means that we should be able to focus the GB computation around the S-polynomial reduction for the pair (f_5, f_7), i.e. $Spoly(f_5, f_7) \xrightarrow{J_{ckt}}_+ r$. As this will be a GB computation over a much smaller set of polynomials, it could be made scalable.

We will explore such term orderings derived from the topology/structure of the sequential circuits, and analyze their effect on the GB computations for sequential verification.

V. Research Problem: Craig Interpolants in Algebraic Geometry

The concept of Craig Interpolants (CI) and their existence comes from symbolic logic [23]; later, algorithms were presented to find the CI for Boolean formulae [24] [25]. Assume that Boolean formulae are represented in Clause Normal Form (CNF) as $f = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ where: i) Each clause C_i is a disjunction (Boolean OR, denoted $\vee$) of literals; ii) Each literal is a Boolean variable x_i or its complement $\overline{x_i}$. The Boolean satisfiability (SAT) problem requires that we find an assignment to the variables such that the formula f is satisfied (SAT), or otherwise prove that no such assignment exists (UNSAT). A CI is related to an UNSAT formula.

Definition 5.1: (From [25]) Let A and B be Boolean formulae given as sets of clauses such that $A \wedge B$ is unsatisfiable (UNSAT). Then there exists a formula P such that: i) A implies P (or $A \subseteq P$); ii) $P \wedge B$ is UNSAT; iii) P refers to only the common variables of A and B. The formula P represents an **interpolant** of A and B.

Given the pair (A,B) and their refutation proof, a procedure called interpolation system constructs an interpolant in linear time and space in the size of the proof [25] [24].

Example 5.1: Let $f = (\overline{d})(\overline{c})(\overline{a} \vee d)(a \vee b \vee c)(\overline{b})$ be a CNF formula. Let $f = A \wedge B = \emptyset$, where $A = (\overline{d})(\overline{c})(\overline{a} \vee d)$ and $B = (a \vee b \vee c)(\overline{b})$. Then $P = \overline{a} \wedge \overline{c}$ is an interpolant of (A,B).

CIs are used to derive abstractions to produce over-approximate image operators in model checking [25]. Since $A \implies P$, P contains A and is an *abstraction* of A. It also has fewer variables, so checking invariants on $P \wedge B$ is easier. The interpolant is derived through a resolution proof of the SAT problem. There can be many interpolants P_i for a pair (A,B); however, it is not feasible to explore a few or all of these interpolants by means of the resolution proof.

<u>Research Problem</u> 5.1: As we wish to perform MC with abstraction using algebraic geometry, we introduce the algebraic geometry analog of CI and explore algorithms to compute them. We conjecture that the concept of CI should be related to elimination ideals, so our line of investigation will focus on Gröbner basis computations with elimination term orders for their computation.

Definition 5.2: Let $F = \{f_1, \dots, f_s\}$ be a set of polynomials in the ring $R = \mathbb{F}_q[x_1, \dots, x_n]$. Let $F = F_A \cup F_B$ and ideals $J = \langle F \rangle, J_A = \langle F_A \rangle, J_B = \langle F_B \rangle$ be corresponding ideals in R such that $J = J_A + J_B$. Let it be known (say, due to application of Weak Nullstellensatz and Gröbner basis) that the varieties $V_{\mathbb{F}_q}(J) = V_{\mathbb{F}_q}(J_A) \cap V_{\mathbb{F}_q}(J_B) = V_{\mathbb{F}_q}(J_A + J_B) = \emptyset$. Also, let the set of variables $X = \{x_1, \dots, x_n\} = X_A \cup X_c \cup X_B$ where X_A, X_B are the set of variables present exclusively in the sets of polynomials F_A, F_B respectively. Only X_c is the set of variables that are common to both sets of polynomials F_A, F_B. Then, there exists a set of polynomials F_P and ideal $J_P = \langle F_P \rangle$ such that

- $V_{\mathbb{F}_q}(J_A) \subseteq V_{\mathbb{F}_q}(J_P)$
- $V_{\mathbb{F}_q}(J_P) \cap V_{\mathbb{F}_q}(J_B) = \emptyset$
- Polynomials of F_P contain the common variables (X_C) of F_A, F_B.

We call the ideal $J_P = \langle F_P \rangle$ the **algebraic interpolant** of $J_A + J_B$.

In the above definition $V_{\mathbb{F}_q}(J_A + J_B) = \emptyset$ implies that the ideal $J_A + J_B$ has an empty variety (UNSAT) problem.

Example 5.2: Based on Example 5.1, we translated the system over $\mathbb{F}_2[a,b,c,d]$. Let $F_A = \{f_1, f_2, f_3\}$ and $F_B = \{f_4, f_5\}$ where: $f_1 : d$; $f_2 : c$; $f_3 : a + da$; $f_4 : abc + ab + ac + bc + a + b + c + 1$; $f_5 : b$. The Boolean interpolant $\overline{a} \wedge \overline{c}$ from Example 5.1 translates to $\mathbb{F}_2$ as the polynomial $f_p = ac + a + c$, with its variety $V(f_p) = \{a = 0, c = 0\}$.

<u>Research Problem</u> *5.2: How can this algebraic interpolant f_p (or in the general case, the set of polynomials F_P) be computed? As the interpolant F_P contains only variables X_C that are common to the polynomial sets F_A, F_B, does this imply that the interpolants can be computed by means of Gröbner bases with an elimination term order $X_A > X_B > X_C$ with X_A, X_B eliminated from the problem?*

We believe that algebraic interpolation is strongly related to the GB computation with the elimination order $X_A > X_B > X_C$, and this relationship needs to be formally derived.

Conjecture 5.1: Computations of algebraic interpolants: Let J_0 denote the ideal of all vanishing polynomials in $\mathbb{F}_{2^k}[x_1, \dots, x_n]$.

- Compute a Gröbner basis $G_1 = GB(J_A + J_0)$ with the elimination order $X_A > X_B > X_C$, and select $F_{P1} = G_1 \cap \mathbb{F}_{2^k}[X_C]$. We conjecture that the Gröbner basis F_{P1} of the elimination ideal corresponds to an algebraic interpolant.
- Analogously, compute a Gröbner basis $G_2 = GB(J_B + J_0)$ with the elimination order $X_B > X_A > X_C$, and select $F_{P2} = G_2 \cap \mathbb{F}_{2^k}[X_C]$. Find an ideal F'_{P2} such that the variety $V(F'_{P2})$ is the complement of the variety $V(F_{P2})$. Then the set F'_{P2} gives another interpolant.

Example 5.3: Consider the polynomials $\{f_1, \dots, f_5\}$ from Example 5.2. Computing F_{P1} as described in Conjecture 5.1 produces $F_{P1} = \{a, c\}$, correctly giving us the desired variety $V(a = 0, c = 0)$. Similarly, when we compute F_{P2}, we find that the interpolant is $ac + a + c + 1$. Notice that the variety $V(ac + a + c + 1) = \{(0,1), (1,0), (1,1)\}$, which is exactly the complement of $V(F_{P1})$.

The aforementioned experiments in Example 5.3 do not invalidate our conjectures, therefore we will attempt to prove them. If we are able to prove our conjectures, then we will further study the proof to derive efficient algorithms to compute the algebraic interpolants. Moreover, the above experiment shows that there can be multiple ways of computing the interpolants. What can these Gröbner basis computations tell us about the number of valid algebraic interpolants for any given problem? Can they be classified as *weak or strong interpolants* based on the sparsity of the polynomials (power of abstraction)?

<u>Research Problem</u> *5.3:* As there can be many interpolants for an ideal-pair (J_A, J_B), we will further resolve these questions:

- Find the minimal interpolant: i.e. find the interpolant F_P such that $V(J_P)$ is the smallest variety larger than $V(J_A)$ such that $V(J_P) \cap V(J_B) = \emptyset$.
- Analogously, find the maximal interpolant.
- Over finite fields, the variety of an elimination ideal is exactly equal to the projection of the variety on the un-eliminated variables. Consider Fig. 3, where variety of the ideals J_A, J_B are respectively projected on the common variables $X_C = \{a, c\}$. Then, does computing F_{P1} (resp. F_{P2}) deliver the minimal (resp. maximal) interpolant?

Once these problems are understood and algorithms for the computation of algebraic interpolants derived, then we will integrate them with word-level reachability analysis for abstraction-based model checking.

VI. Research Problem: Finding UNSAT Cores of a Set of Polynomials using Gröbner Bases

For an unsatisfiable (unsat) Boolean formula C given in clause normal form (CNF), it is possible to identify a subset C_c of the original set of clauses C such that C_c is unsat too. This set C_c is called the unsat core of the problem. An unsat core

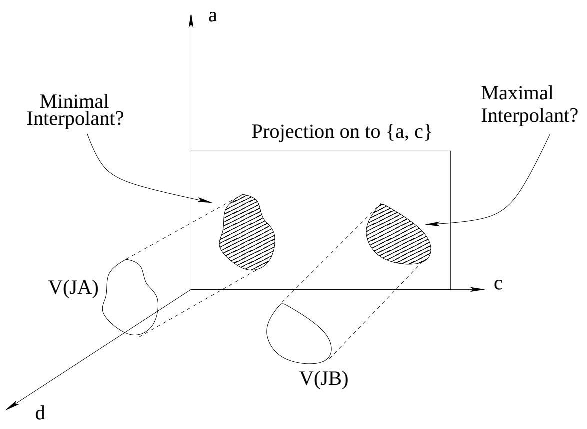

Fig. 3: Algebraic interpolant: Projection of varieties on common variables?

is a minimally unsatisfiable subformula (MUS) when any of its subsets is satisfiable. Unsat cores find application in many areas such as artificial intelligence [52], hardware synthesis [53], formal verification [26], among others [45], and unsat core extractors for CNF formulas are available (e.g. [54], [55]). In the context of MC, unsat cores are employed for abstraction [56], and also in the refinement phase in CEGAR-based MC [26]. Unsat cores have analogous manifestations in the algebraic geometry domain.

Research Problem 6.1: Suppose that we are given a set of polynomials $F = \{f_1,\dots,f_s\}$ with coefficients from $\mathbb{F}_{2^k}$. Suppose, further, that the system of polynomial equations $f_1 = f_2 = \cdots = f_s = 0$ has no common zero – i.e. the problem is unsat. Can we identify a subset $F_c \subseteq F$ such that the set of polynomials of F_c also has no common zeros?

As we explore algebraic geometry based word-level abstractions for MC, we need to develop algorithms to identify small (preferably minimal!) unsat cores of a set of polynomials and utilize them for abstraction/refinement. This problem can be modeled using the Weak Nullstellensatz, and the Gröbner basis algorithm can be extended to identify a unsat core.

Theorem 6.1 (The Weak Nullstellensatz over finite fields [46]): Let $J = \langle f_1,\dots,f_s\rangle$ be an ideal in the ring $R = \mathbb{F}_q[x_1,\dots,x_n]$ and $V_{\mathbb{F}_q}(J)$ be its variety over $\mathbb{F}_q$. Let $J_0 = \langle x_i^q - x_i : i = 1,\dots,n\rangle$ be the ideal of all vanishing polynomials in R. Then $V_{\mathbb{F}_q}(J) = \emptyset \iff 1 \in (J+J_0) \iff \text{reduced}GB(J+J_0) = \{1\}$.

The inconsistency of a set of polynomials can be checked using Nullstellensatz by testing whether the ideal $J+J_0$ contains the unit element. The Gröbner basis algorithm, when applied to this ideal membership test, can be extended to identify an unsat core $F_c \subseteq F$. In a recent publication [57], we presented an approach to identify a core by analyzing the $Spoly(f_i, f_j) \xrightarrow{F}_+ g_{ij}$ reductions in Buchberger's algorithm. Since F is itself an unsat core, definitely *there exists a sequence of Spoly reductions in Buchberger's algorithm where $Spoly(f_i, f_j) \xrightarrow{F}_+ 1$ is achieved.* This suggests that we should be able to identify a core by recording the *data* generated by Buchberger's algorithm — namely, the critical pairs(f_i, f_j) used in the Spoly computations, and the polynomials from F used to cancel terms in the reduction $Spoly(f_i, f_j) \xrightarrow{F}_+ 1$.

While our algorithm [57] did provide a solution to identify a core F_c, the algorithm by itself is unable to provide any information (or guarantees) on the minimality of F_c. As algebraic geometry is a "stronger" reasoning engine than SAT, we wish to further improve our core extraction algorithm by extracting smaller, and possibly minimal, cores. This is critical for abstraction/refinement in MC for faster convergence of verification. First, we give an overview of our approach of [57] using an example, and then we will describe the research problems to address to make the approach deliver a smaller core.

Example 6.1: Consider the following set of polynomials $F = \{f_1,\dots,f_9\}$: $f_1 : abc+ab+ac+bc+a+b+c+1$; $f_2 : b$; $f_3 : ac$; $f_4 : ac+a$ $f_5 : bc+c$; $f_6 : abd+ad+bd+d$; $f_7 : cd$; $f_8 : abd+ab+ad+bd+a+b+d+1$; $f_9 : abd+ab+bd+b$.

Assume $>_{DEGLEX}$ monomial ordering with $a > b > c > d$. Let $J = \langle F\rangle \subset \mathbb{F}_2[a,b,c,d]$. Then $V(J) = \emptyset$ as $GB(J) = 1$. The set F consists of 4 *minimal* cores: $F_{c1} = \{f_1, f_2, f_3, f_4, f_7, f_8\}, F_{c2} = \{f_2, f_4, f_5, f_6, f_8\}, F_{c3} = \{f_2, f_3, f_4, f_6, f_8\}$, and $F_{c4} =$

$\{f_1, f_2, f_4, f_5\}$.

Let us suppose that in the GB computation corresponding to Example 6.1, the first 3 critical *Spoly* pairs analyzed are $(f_1, f_2), (f_3, f_4)$ and (f_2, f_5). It turns out that the Spoly-reductions corresponding to these 3 pairs lead to the unit ideal. Recording the data corresponding to this sequence of reductions is depicted by means of a graph in Fig. 4. We call this graph a *refutation tree*. In the refutation tree, the nodes correspond to the polynomials used in the Spoly-reductions, and the edges e_{ij} denote that the polynomial at node i was generated using the one at node j.

To identify an $F_c \subset F$, we start from the refutation node "1", and traverse the graph in reverse, all the way up to the leaves. Then, all the leaves in the transitive fanin of "1" constitute an unsat core. The polynomials (nodes) that do not lie in the transitive fanin of "1" can be safely discarded from F_c. From Fig. 4, $F_c = \{f_1, f_2, f_3, f_4, f_5\}$ is identified as an unsat core of F.

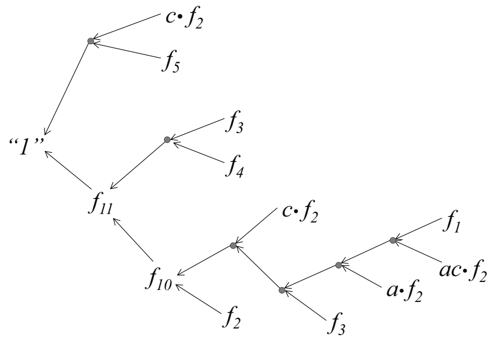

Fig. 4: Generating refutation trees to record unsat cores.

Limitations of [57]: The size and quality of the core obtained through our approach is dependent on the order of pairs of polynomials (f_i, f_j) chosen for the Spoly reduction: $Spoly(f_i, f_j) \xrightarrow{F}_+ 1$. Moreover, our approach is also not very efficient when a polynomial $f_k \in F$ can be represented using a combination of the rest of the polynomials of the core, e.g.: $f_k = \sum_{j \neq k} c'_j f_j$. Thus, f_k is redundant and can be dropped from the core. The refutation-tree based approach cannot easily identify such redundancies. For example, in the core $F_c = \{f_1, f_2, f_3, f_4, f_5\}$ derived from Fig. 4, the polynomial f_3 is redundant and can be represented as a combination of $\{f_1, f_2, f_4, f_5\}$, but our approach cannot identify this relationship. In our experiments, we had found many such cases; therefore we would like to investigate techniques that could identify such polynomials.

Research Problem *6.2: During the execution of Buchberger's algorithm, many critical pairs (f_i, f_j) do not add any new polynomials in the basis when $Spoly(f_i, f_j) \xrightarrow{F}_+ 0$ gives zero remainder. For the purpose of the GB computation, this data is discarded. We wish to investigate if the discarded data in Buchberger's algorithm corresponding to the computation $Spoly(f_i, f_j) \xrightarrow{F}_+ 0$ contains the information that characterizes the dependency of a polynomial $f_k \in \langle F - \{f_k\}\rangle$?*

The rationale for this investigation is as follows: Every $Spoly(f_i, f_j) \xrightarrow{F}_+ 0$ implies that some polynomial combination of $\{f_1, \dots, f_s\}$ vanishes: i.e. $c_1 f_1 + c_2 f_2 + \cdots + c_s f_s = 0$, for some $c_1, \dots, c_s$. For each $Spoly(f_i, f_j) \xrightarrow{F}_+ 0$ reduction, we could record this information as in Eqn. (2), also represented in matrix form in Eqn. (3):

$$\begin{cases} c_1^1 f_1 + c_2^1 f_2 + \cdots + c_s^1 f_s = 0 \\ c_1^2 f_1 + c_2^2 f_2 + \cdots + c_s^2 f_s = 0 \\ \quad\vdots \\ c_1^m f_1 + c_2^m f_2 + \cdots + c_s^m f_s = 0 \end{cases} \quad (2) \qquad \begin{bmatrix} c_1^1 & c_2^1 & \cdots & c_s^1 \\ c_1^2 & c_2^2 & \cdots & c_s^2 \\ \vdots & \vdots & \ddots & \vdots \\ c_1^m & c_2^m & \cdots & c_s^m \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_s \end{bmatrix} = 0 \quad (3)$$

We would like to investigate what information of inter-polynomial dependencies could be derived from the $m \times s$ matrix of Eqn. (3) for refining the core.

Research Problem 6.3: Deriving Craig Interpolants from Refutation Trees: In the CNF-SAT domain, Craig interpolants are derived from a resolution proof Π *of an Unsat problem.* Π *is a DAG whose vertices represent clauses and incoming edges to a vertex v_i relate how the resolvent clause at vertex v_i is generated from its predecessor clauses. The refutation trees employed in unsat core extraction (Fig. 4) also represents a similar relationship. This motivates us to investigate whether we can draw inspirations from McMillan's algorithm [25] of Boolean interpolant generation from resolution proofs to derive algebraic interpolants from refutation trees? Perhaps such an algorithm might be more efficient than GB computation over elimination ideals to derive the interpolant.*

This concludes the description of the theoretical and algorithmic research problems that both PIs and their students will investigate together as part of the project.

VII. The Research Plan: Putting it all together

Once the word-level FSM traversal, Craig interpolation and Unsat core related problems have been theoretically solved, they will be integrated as algorithmic implementations into sequential equivalence and model checkers. From our prior work, we have developed verification tools [6] written in C++ that are available as symbolic computation engines in $\mathbb{F}_{2^k}$. These tools have also been integrated in a verification flow involving the Singular [58] computer-algebra system. To this setup, we are going to integrate the new SEC and the abstraction based MC engines. The benchmarks that we will experiment with come from a variety of sources: i) Word-level RTL Verilog designs available to us from our industry collaborators; ii) ISCAS'91 and ITC'99 benchmarks; and iii) benchmarks from the hardware model checking repository. We already have conversion scripts that translate designs from these formats into our internal ALG file format which our tools take as input.

The interdisciplinary problems require tight collaborations between the PIs and their respective research groups. PI Kalla is requesting support for two graduate students, and PI Enescu plans to support one student through this project. In the past, both PIs have been visiting each other every year for research consultations. We plan to continue this interaction.

A demonstration of abstraction based *k*-BMC with UNSAT cores of polynomials:

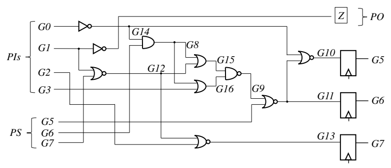

Fig. 5: Gate-level Schematic of Example Circuit

Example 7.1: Figure 5 shows a sequential circuit ("s27" from the ISCAS benchmark set) with 3-bit state registers PS = $\{G7, G6, G5\}$ and NS = $\{G13, G11, G10\}$. Its underlying FSM contains 8 states. On this FSM, define a LTL property $p = AG((\neg G13)\mathbf{U}(\neg Z))$ with given initial state $\{000\}$. MC on this machine requires traversal on the STG to search for an accepting trace. If we use k-BMC without abstraction-refinement, we need to unroll the machine to iteration $k = 3$. We show how abstraction will be applied using our setup.

Because the circuit has 3 latches, we model the problem over the field $\mathbb{F}_{2^3}$. First, let the bound $k = 0$. Generate the polynomial constraints for initial state ideal I and check $SAT(I \wedge \neg p)$ using Gröbner bases. The ideal $I = \langle G14+1+G0, G8+G14\cdot G6, G15+G12+G8+G12\cdot G8, G16+G3+G8+G3\cdot G8, G9+1+G16\cdot G15, G10+1+G14+G11+G14\cdot G11, G11+1+G5+G9+G5\cdot G9, G12+1+G1+G7+G1\cdot G7, G13+1+G2+G12+G2\cdot G12, Z+1+G1, G5, G6, G7(\text{Initial state } 000\ )\rangle$

Property $\neg p$ is also written as a polynomial in the first time-frame: $\neg p = Z\cdot G13+1$. As $I \wedge \neg p$ is unsat by Nullstellensatz, we extract an unsat core using our approach: $Core(I \wedge \neg p) = \{G12+1+G1+G7+G1\cdot G7, G13+1+G2+G12+G2\cdot G12, Z+1+G1, G7\}$.

The result shows that state variables $\{G5, G6, G10, G11\}$ are irrelevant when considering the violation of p. Thus, we can remove the latches $G10, G11$, and make $G5, G6$ primary inputs. In this way, the number of state variables is reduced to 1, such that the new machine only contains 2 states. Since the state space is greatly reduced, we can execute unbounded model checking on this abstracted machine with less cost. As a result, property p is not violated on the abstracted machine. Therefore, p is also a passing property of the original machine and the refined k-BMC algorithm terminates!

VIII. Broader Impacts

This proposal has described a set of research problems that aim to solve the SEC and MC problems at the word-level using algebraic geometry. The core focus of the research lies in discovering algebraic geometry analogs of the problems of reachability analysis, Craig interpolants and Unsat Cores – all performed at the level of polynomials in word-level variables. The problems are modeled over the finite field $\mathbb{F}_{2^k}$, where k corresponds to the state-register bits. The research requires investigations into new theoretical concepts and practical implementations to overcome the complexity of Gröbner basis engines.

Successful completion of this work will have the following impacts:

Impact on Knowledge in Verification and in Mathematics: The inter-disciplinary nature of this work brings together techniques from Gröbner bases, algebraic geometry, and sequential verification using abstractions. The targeted problems of abstractions, reachability and unsat cores now have to be studied analogously in the algebraic geometry domain. Conversely, Gröbner basis techniques also find application to SEC and MC. Our preliminary studies have provided us new theoretical mathematical problems, practical implementation issues related to scalability, as well as their engineering applications to investigate. The cross-fertilization of ideas would bring new knowledge to our respective areas of work.

Impact on Technology: Gröbner bases are a very important technology. They have applications in many areas both within and outside of computer engineering. It is imperative to develop new tools to support such applications. Unfortunately, EDA/CAD community's involvement in this area is really lacking. This research takes a big step in this direction by proposing the development of new Gröbner basis applications, and integration of symbolic computation engines in the sequential verification flow. This research will introduce automatic word-level algebraic reasoning as a new dimension for state-space abstraction. Knowledge gained from this research will lead to new and better EDA tools in this domain.

Impact on Education: Given the interdisciplinary nature of this work, our PhD students will have to learn a lot about the other fields: computer algebra, algebraic geometry, arithmetic circuit design and verification, and logic synthesis. "Hardware Verification using Computer Algebra" for EE & CS students. This course covers topics in Gröbner bases and their computation (from [46] [59]), Radical ideals & Nullstellensatz [46], Galois fields and their applications [60], and other selected topics in hardware/software verification using Gröbner basis techniques [36] [61] [21] [62] [35], etc. The PI is also writing a textbook covering the application of this material to hardware verification. The material is made freely available to the community at: http://www.ece.utah.edu/~kalla/index_6745.html. The textbook is not complete yet, but the completed chapters are also available http://www.ece.utah.edu/~kalla/ECE6745/book.pdf. All the knowledge gained from this project, will undoubtedly get incorporated in the course materials. The educational impact of this project is significant.

Recruitment of Minorities: PI Enescu has recently attracted a woman PhD student, Ms. Ilioaea, to this project and she has already published a paper with us [57]. PI Kalla is soliciting participation in this project through the Univ. of Utah's Undergraduate Research Opportunities program (UROP), particularly for combined BS/MS degree students. UROP has good outreach to minority students on campus, so we plan to leverage their resources in this endeavor.

Impact on Society: Formal verification of hardware designs is an imperative. Incomplete verification raises the potential for bugs in the design. Bug escapes can be expensive to rectify, which may raise the cost of hardware. Hardware bugs can also cause security vulnerabilities in cyber-infrastructure. As word-level abstraction improves the efficiency for sequential verification, this work will also enable security, protection and privacy of data, which indirectly impacts the safety and security of society.

REFERENCES

[1] 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.

[2] T. Pruss, P. Kalla, and F. Enescu, “Equivalence Verification of Large Galois Field Arithmetic Circuits using Word-Level Abstraction via Groebner Bases,” in *Design Automation Conf.*, 2014.

[3] ——, “Efficient Symbolic Computation for Word-Level Abstraction from Combinational Circuits for Verification over Finite Fields,” *IEEE Trans. on CAD*, vol. 35, no. 7, pp. 1206–1218, July 2016.

[4] X. Sun, P. Kalla, T. Pruss, and F. Enescu, “Formal verification of sequential galois field arithmetic circuits using algebraic geometry,” in *Proc. Design, Automation and Test in Europe*, 2015.

[5] X. Sun, P. Kalla, and F. Enescu, “Word-level Traversal of Finite State Machines using Algebraic Geometry,” in *Proc. High-Level Design Validation and Test*, 2016.

[6] T. Pruss and P. Kalla, “Word-level abstraction tool for formal verification,” [online] available at http://www.ece.utah.edu/~pruss/abstract.html.

[7] P. Kalla, “Formal verification of arithmetic datapaths using algebraic geometry and symbolic computation,” in *Proc. FMCAD*, 2015, p. 2, slides available online http://www.cs.utexas.edu/users/hunt/FMCAD/FMCAD15/slides/fmcad15-tutorial-kalla.pdf.

[8] S. Minato, “Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems,” in *DAC*, 93, pp. 272–277.

[9] ——, “Calculation of Unate Cube Set Algebra using Zero-Suppressed BDDs,” in *Proc. Design Automation Conference (DAC)*, 1994, pp. 420–424.

[10] R. Brayton and A. Mishchenko, “ABC: An Academic Industrial-Strength Verification Tool,” in *Computer Aided Verification*, vol. 6174. Springer, 2010, pp. 24–40.

[11] U. Gupta, P. Kalla, and V. Rao, “Gröbner Basis Reductions on Datapath Circuits using the Unate Cube Set Algebra,” in *(submitted) Design Auto. Conf.*, 2017, in review.

[12] N. Shekhar, P. Kalla, F. Enescu, and S. Gopalakrishnan, “Exploiting Vanishing Polynomials for Equivalence Verification of Fixed-Size Arithmetic Datapaths,” in *Intl. Conf. Computer Design*, 2005.

[13] ——, “Equivalence Verification of Polynomial Datapaths with Fixed-Size Bit-Vectors using Finite Ring Algebra,” in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2005.

[14] N. Shekhar, P. Kalla, and F. Enescu, “Equivalence Verification Arithmetic Datapaths with Multiple Word-length Operands,” in *Design Automation and Test in Europe (DATE)*, March 2006, pp. 6–10.

[15] N. Shekhar, P. Kalla, M. B. Meredith, and F. Enescu, “Simulation Bounds for Equivalence Verification of Arithmetic Datapaths with Finite Word-Length Operands,” in *Formal Methods in Computer Aided Design*, November 2006, pp. 179–186.

[16] N. Shekhar, P. Kalla, and F. Enescu, “Equivalence Verification of Polynomial Datapaths using Ideal Membership Testing,” *IEEE Transactions on CAD*, vol. 26, no. 7, pp. 1320–1330, July 2007.

[17] N. Shekhar, P. Kalla, M. B. Meredith, and F. Enescu, “Simulation Bounds for Equivalence Verification of Polynomial Datapaths using Finite Ring Algebra,” *IEEE Transactions VLSI*, vol. 16, no. 4, pp. 376–387, 2008.

[18] S. Gopalakrishnan, P. Kalla, and F. Enescu, “Optimization of Arithmetic Datapaths with Finite Word-Length Operands,” in *Asia/South-Pacific Design Automation Conference*, 2007, pp. 155–161.

[19] S. Gopalakrishnan, P. Kalla, B. Meredith, and F. Enescu, “Finding Linear Building-Blocks for RTL Synthesis of Polynomial Datapaths with Fixed-Size Bit-Vectors,” in *Proc. Intl. Conf. on CAD (ICCAD)*, 2007.

[20] J. Lv, P. Kalla, and F. Enescu, “Verification of Composite Galois Field Multipliers over GF($(2^m)^n$) using Computer Algebra Techniques,” in *IEEE High-Level Design Validation and Test Workshop*, 2011, pp. 136–143.

[21] ——, “Efficient Groebner Basis Reductions for Formal Verification of Galois Field Multipliers,” in *IEEE Design, Automation and Test in Europe*, 2012.

[22] E. C. A Biere, A Cimatti and Y. Yhu, “Symbolic model checking without bdds,” in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 1999, p. 193207.

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

[24] P. Pudlák, “Lower bounds for resolution and cutting plane proofs and monotone computations,” *J. Symbolic Logic*, vol. 62, no. 2, pp. 981–998, 1997.

[25] K. McMillan, “Interpolation and SAT-based model checking,” in *Proceedings of Computer Aided Verification*, ser. LNCS, vol. 2725. Springer, 2003, pp. 1–13.

[26] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, “Counterexample-Guided Abstraction Refinement for Symbolic Model Checking,” *Journal of ACM*, vol. 50, no. 5, pp. 752–794, Sept 2003.

[27] A. Bradley, “SAT-based model checking without unrolling,” in *Verification, Model Checking and Abstract Interpretation (VMCAI)*, 2011, pp. 70–87.

[28] H. Jain, D. Kroening, N. Sharygina, and E. Clarke, “Word Level Predicate Abstraction and Refinement for Verifying RTL Verilog,” in *in Des. Auto. Conf.* ACM/IEEE, 2005, pp. 445–450.

[29] K. McMillan, “Lazy abstraction with interpolants,” 2006, pp. 123–136.

[30] ——, "Lazy abstraction for program testing and verification," in *Computer-Aided Verification*, 2010, pp. 104–118.

[31] O. Coudert and J. Madre, "A Unified Framework for the Formal Verification of Sequential Circuits," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 1990, pp. 126–129.

[32] O. Coudert, C. Berthet, and J. C. Madre, "Verification of synchronous sequential machines based on symbolic execution," in *Automatic verification methods for finite state systems*. Springer, 1990, pp. 365–373.

[33] H. Touati, H. Savoj, B. Lin, R. Brayton, and A. Sangiovanni-Vincentelli, "Implicit State Enumeration of Finite State Machines using BDDs," in *Proc. ICCAD*, 1990, pp. 130–133.

[34] R. Lidl and H. Niederreiter, *Finite Fields*. Cambridge University Press, 1997.

[35] M. Clegg, J. Edmonds, and R. Impagliazzo, "Using the Gröbner Basis Algorithm to Find Proofs of Unsatisfiability," in *ACM Symposium on Theory of Computing*, 1996, pp. 174–183.

[36] G. Avrunin, "Symbolic Model Checking using Algebraic Geometry," in *Computer Aided Verification Conference*, 1996, pp. 26–37.

[37] M. Y. Vardi and Q. Tran, "Groebner Bases Computation in Boolean Rings for Symbolic Model Checking," in *IASTED*, 2007.

[38] C. Van Eijk, "Sequential equivalence checking without state space traversal," in *Design, Automation and Test in Europe, 1998., Proceedings*. IEEE, 1998, pp. 618–623.

[39] D. Stoffel and W. Kunz, "Record & play: A structural fixed point iteration for sequential circuit verification," in *Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*. IEEE Computer Society, 1997, pp. 394–399.

[40] Z. Khasidashvili, M. Skaba, D. Kaiss, and Z. Hanna, "Theoretical framework for compositional sequential hardware equivalence verification in presence of design constraints," in *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*. IEEE Computer Society, 2004, pp. 58–65.

[41] J. Baumgartner, H. Mony, V. Paruthi, R. Kanzelman, and G. Janssen, "Scalable sequential equivalence checking across arbitrary design transformations," in *Computer Design, 2006. ICCD 2006. International Conference on*. IEEE, 2007, pp. 259–266.

[42] J. R. Burch, E. M. Clarke, K. L. McMillan, and D. L. Dill, "Sequential circuit verification using symbolic model checking," in *Design Automation Conference, 1990. Proceedings., 27th ACM/IEEE*. IEEE, 1990, pp. 46–51.

[43] J. R. Burch, E. M. Clarke, and D. E. Long, "Representing circuits more efficiently in symbolic model checking," in *Proceedings of the 28th ACM/IEEE Design Automation Conference*. ACM, 1991, pp. 403–407.

[44] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu, "Symbolic Model Checking using SAT procedures instead of BDDs," in *In Proc. DAC*, 1999.

[45] J. Marques-Silva, "Minimal Unsatisfiability: Models, Algorithms and Applications," in *IEEE Intl. Symp. on Multi-Valued Logic*.

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

[47] T. W. Dube, "The Structure of Polynomial Ideals and Gröbner bases," *SIAM Journal of Computing*, vol. 19, no. 4, pp. 750–773, 1990.

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

[49] S. Gao, A. Platzer, and E. Clarke, "Quantifier Elimination over Finite Fields with Gröbner Bases," in *Intl. Conf. Algebraic Informatics*, 2011.

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

[51] ——, "A criterion for detecting unnecessary reductions in the construction of a groebner bases," in *EUROSAM*, 1979.

[52] J. L. de Siqueira and J.-F. Puget, "Explanation-based generalization of failures," in *Proc. European Conf. Artificial Intelligence*, 1988, pp. 339–344.

[53] J.-H. R. Jiang, C.-C. Lee, A. Mishchenko, and C.-Y. Huang, "To SAT or Not to SAT: Scalable Exploration of Functional Dependency," *IEEE Trans. Comp.*, vol. 59, no. 4, pp. 457–466, April 2010.

[54] A. Nadel, V. Ryvchin, and O. Strichman, "Accelerated deletion-based extraction of minimal unsatisfiable cores," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 9, pp. 27–51, 2014.

[55] A. Belov, I. Lynce, and J. Marques-Silva, "Towards efficient MUS extraction," *AI Communications*, vol. 25, no. 2, pp. 97–116, 2012.

[56] L. Zhang, "Design verification for sequential systems at various abstraction levels," Ph.D. dissertation, Virginia Polytechnic and State Univesity, 2005.

[57] X. Sun, P. Kalla, and F. Enescu, "Finding unsatisfiable cores of a set of polynomials using the Gröbner basis algorithm," in *Intl. Conf. Principles and Practice of Constraint Programming*, 2016.

[58] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, "SINGULAR 3-1-3 — A computer algebra system for polynomial computations," 2011, http://www.singular.uni-kl.de.

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

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

[61] 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.

[62] S. Sankaranarayanan, H. B. Sipma, and Z. Manna, "Non-linear Loop Invariant Generation using Grobner Bases," *SIGPLAN Not.*, vol. 39, no. 1, pp. 318–329, 2004.