Sequential Circuit Verification at Word Level using Algebraic Geometry

Xiaojun Sun

Ph.D Candidate
Electrical and Computer Engineering, University of Utah
xiaojuns@ece.utah.edu

Ph.D's Dissertation Proposal

Outline

- Introduction
- Sequential circuit verification requires reachability analysis
- From bit-level to word-level
- One step back: Sequential arithmetic circuit verification
- New inspiration: UNSAT core extraction using algebraic geometry
- The plan

Agenda

- Focus
  - Implicitly analyze the reachability of a sequential circuit at word level
  - Algebraic geometry to assist in sequential circuits verification and abstraction refinement
- Motivation
  - Data flow = word-level info
  - Conventional techniques are bit-level
  - Need bit-level to word-level abstraction
  - Verification $\hookleftarrow$ Reachability $\hookleftarrow$ Abstraction (at word level?)
- Target problems
  - Given: sequential circuit (FSM), with k-bit state variables, property for verification
    - Perform sequential equivalence/property checking at word level
    - Identify UNSAT cores in word-level problems

- Approach: **Algebraic geometry techniques**
  - Model system over $\mathbb{F}_{2^k}$ $\implies$ Represent at word-level; compatible with bit-level:$\mathbb{F}_2 \subset \mathbb{F}_{2^k}$
  - Gröbner basis methods + Elimination ordering + BFS traversal
- Challenge: Discover efficient algorithm to implement image computations, set operations, UNSAT proofs, etc. at word level
- Contributions:
  - Polynomial abstraction + algebraic geometry techniques applied to reachability analysis
  - A new algorithm based on Gröbner basis computation to extract UNSAT cores

Motivation

- Importance of reachability analysis in sequential circuits verification
  - Circuits $\rightarrow$ state machines; errors $\rightarrow$ bad states
  - Bad states are reachable *implies* errors affect circuit behavior
- Advantages exploiting word-level verification
  - Many circuit datapaths/system models are described at word level
  - Reduce state space, avoid "bit-blasting"
- Why use algebraic geometry?
  - A symbolic representation for bit-level & word-level: $\mathbb{F}_{2^k}$
  - Algebraic geometry: allows reasoning on solutions by analyzing polynomials, solution $\leftarrow$ states
  - Recent work [Tim, *Abstraction using GB*, DAC'14] [Lv, *Equivalence*, TCAD'13] [M. Brickenstein, *GB for Boolean Poly*, J.Symb.Comp.'09] shows it is practical to apply algebraic geometry to circuit verification

Outline

- Introduction
- Sequential circuit verification requires reachability analysis
- From bit-level to word-level
- One step back: Sequential arithmetic circuit verification
- New inspiration: UNSAT core extraction using algebraic geometry
- The plan

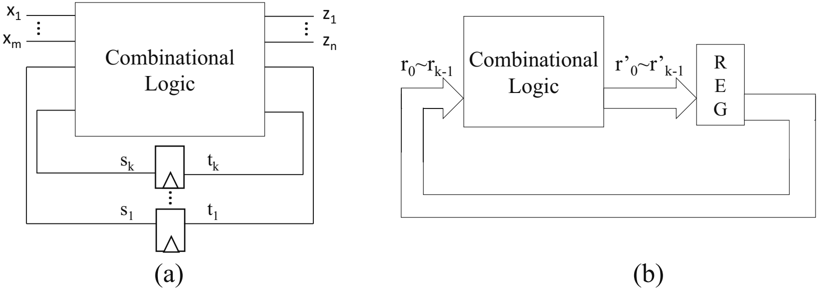


Figure : FSM models of sequential circuits

- (a) A typical model for sequential circuits
- (b) sequential datapath without primary inputs

State transition graph (STG) and breadth-first search (BFS) state space traversal

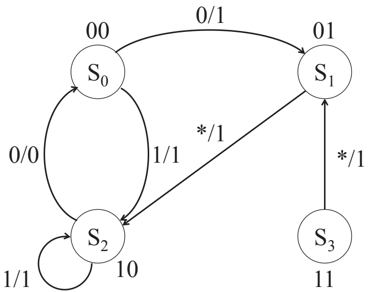


Figure : State Transition Graph

- Initial state: $\{00\}$
- Iteration 1:
  - Start from $\{00\}$
  - One-step transition: $\{01, 10\}$
  - Newly reached: $\{01, 10\}$
- Iteration 2:
  - Start from $\{01, 10\}$
  - One-step transition: $\{00, 10\}$
  - Newly reached: $\emptyset$
- All reachable states detected. Final reached states: $\{00, 01, 10\}$

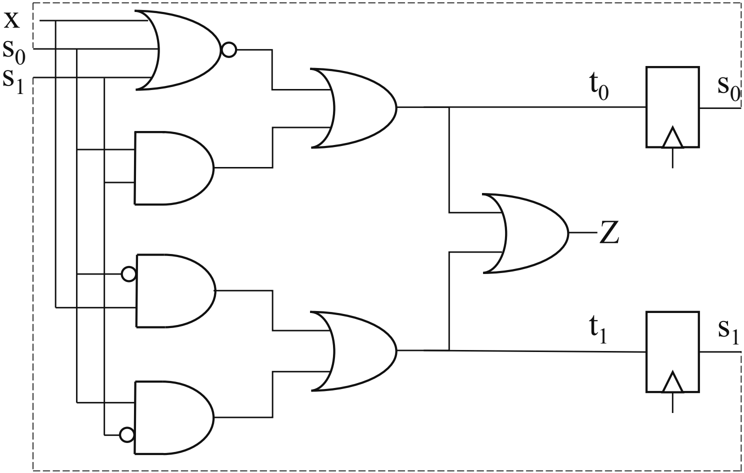


Figure : A Gate-level Circuit corresponding to last STG

Breadth-First Traversal Algorithm based on Boolean formulas

ALGORITHM 1: Breadth-first Traversal Algorithm

Input: Transition functions Δ, initial state S^0

$\mathit{from}^0 = \mathit{reached} = S^0$;
repeat
    $i \leftarrow i + 1$;
    $\mathit{to}^i \leftarrow \mathrm{Img}(\Delta, \mathit{from}^{i-1})$;
    $\mathit{new}^i \leftarrow \mathit{to}^i \cap \overline{\mathit{reached}}$;
    $\mathit{reached} \leftarrow \mathit{reached} \cup \mathit{new}^i$;
    $\mathit{from}^i \leftarrow \mathit{new}^i$;
until $\mathit{new}^i == 0$;
return *reached*

- Image function:
  $\mathrm{Img}(\Delta, \mathit{from}) = \exists_s \exists_x [T(s,x,t) \wedge \mathit{from}] = \exists_s \exists_x \bigwedge_{i=1}^{n} (t_i \overline{\oplus} \Delta_i) \wedge \mathit{from}$
- Initial state: $\mathit{from}^0 = \overline{s_0} \wedge \overline{s_1}$ $(\{00\})$
- Transition function:

$$\Delta_1 : t_0 \overline{\oplus} ((\overline{x \vee s_0 \vee s_1}) \ \vee \ s_0 \wedge s_1)$$
$$\Delta_2 : t_1 \overline{\oplus} (\overline{s_0} \wedge x \ \vee \ \overline{s_1} \wedge s_0)$$

- Iteration 1:
  - One-step transition
    $\mathit{to}^1 = \exists_{s_0, s_1, x} (\Delta_1 \wedge \Delta_2 \wedge \mathit{from}^0) = \overline{t_0} \wedge t_1 \ \vee \ t_0 \wedge \overline{t_1}$ $(\{01, 10\})$
  - Newly reached
    $\mathit{new}^1 = (\overline{t_0} \wedge t_1 \ \vee \ t_0 \wedge \overline{t_1}) \ \wedge \ \overline{(\overline{t_0} \wedge \overline{t_1})} = \overline{t_0} \wedge t_1 \ \vee \ t_0 \wedge \overline{t_1}$
- Iteration 2: same fashion
- Return value: $\mathit{reached} = \{\overline{t_0} \vee \overline{t_1}\}$ $(\{00, 01, 10\})$

Outline

- Introduction
- Sequential circuit verification requires reachability analysis
- From bit-level to word-level
  - All about math
  - Utilize the math tool: what does Tim get?
  - My approach
- One step back: Sequential arithmetic circuit verification
- New inspiration: UNSAT core extraction using algebraic geometry
- The plan

Galois field(GF) $\mathbb{F}_q$ is a finite field with q elements, $q = p^k$

- Commutative Ring with unity, associate, distributive laws
- Closure property: $+, -, \times$, inverse $(\div)$
- $\mathbb{F}_p \equiv (\mathbb{Z} \pmod p)$, where $p =$ prime, is a field
  - $\mathbb{Z}_7 = \{0, 1, 2, 3, 4, 5, 6\}$

Our interest: $\mathbb{F}_q = \mathbb{F}_{2^k}$, i.e. $q = 2^k$

- $\mathbb{F}_{2^k}$: k-dimensional extension of $\mathbb{F}_2$
  - k-bit bit-vector, AND/XOR arithmetic

To construct $\mathbb{F}_{2^k}$

- $\mathbb{F}_{2^k} \equiv \mathbb{F}_2[x] \pmod{P(x)}$
- $P(x) \in \mathbb{F}_2[x]$, irreducible polynomial of degree k

Field Elements: *e.g.* $\mathbb{F}_8$

Consider: $\mathbb{F}_{2^3} = \mathbb{F}_2[x] \pmod{x^3 + x + 1}$
$A \in \mathbb{F}_2[x]$
$A \pmod{x^3 + x + 1} = a_2 x^2 + a_1 x + a_0$. Let $P(\alpha) = 0$:

- $\langle a_2, a_1, a_0 \rangle = \langle 0, 0, 0 \rangle = 0$
- $\langle a_2, a_1, a_0 \rangle = \langle 0, 0, 1 \rangle = 1$
- $\langle a_2, a_1, a_0 \rangle = \langle 0, 1, 0 \rangle = \alpha$
- $\langle a_2, a_1, a_0 \rangle = \langle 0, 1, 1 \rangle = \alpha + 1$
- $\langle a_2, a_1, a_0 \rangle = \langle 1, 0, 0 \rangle = \alpha^2$
- $\langle a_2, a_1, a_0 \rangle = \langle 1, 0, 1 \rangle = \alpha^2 + 1$
- $\langle a_2, a_1, a_0 \rangle = \langle 1, 1, 0 \rangle = \alpha^2 + \alpha$
- $\langle a_2, a_1, a_0 \rangle = \langle 1, 1, 1 \rangle = \alpha^2 + \alpha + 1$

Polynomial functions in Galois field

- Polynomial functions over $\mathbb{F}_{2^k}$ $\Leftrightarrow$ Boolean functions on k-bit vectors

$$f : \mathbb{B}^k \rightarrow \mathbb{B}^k \Leftrightarrow \mathcal{F} : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_{2^k}$$

- Example: Lagrange's interpolation

$\{a_2a_1a_0\}$	A	$\rightarrow$	$\{z_2z_1z_0\}$	Z
000	0	$\rightarrow$	000	0
001	1	$\rightarrow$	000	0
010	α	$\rightarrow$	001	1
011	$\alpha+1$	$\rightarrow$	001	1
100	α^2	$\rightarrow$	010	α
101	α^2+1	$\rightarrow$	010	α
110	$\alpha^2+\alpha$	$\rightarrow$	011	$\alpha+1$
111	$\alpha^2+\alpha+1$	$\rightarrow$	011	$\alpha+1$

- 8 pairs of (A, Z), use Lagrange's interpolation to abstract a polynomial function

$$Z = \mathcal{F}(A) = \sum_{n=1}^{8} \left[\frac{\prod_{i \neq n}(A - A_i)}{\prod_{i \neq n}(A_n - A_i)} \cdot Z_n \right]$$

- Result $= (\alpha^2 + 1)A^4 + (\alpha^2 + 1)A^2$ *implies* polynomial representation of the function
- $P(\alpha) = \alpha^3 + \alpha + 1 = 0$

Let $\mathbb{F}_q = GF(2^k)$:

- $\mathbb{F}_q[x_1, \ldots, x_n]$: ring of all polynomials with coefficients in $\mathbb{F}_q$
- Given a set of polynomials:
  - $f, f_1, f_2, \ldots, f_s \in \mathbb{F}_q[x_1, \ldots, x_n]$
  - Find solutions to $f_1 = f_2 = \cdots = f_s = 0$
- Variety: Set of ALL solutions to a given system of polynomial equations: $V(f_1, \ldots, f_s)$
  - In $\mathbb{R}[x, y]$, $V(x^2 + y^2 - 1) = \{all\ points\ on\ circle : x^2 + y^2 - 1 = 0\}$
  - In $\mathbb{R}[x]$, $V(x^2 + 1) = \emptyset$
  - In $\mathbb{C}[x]$, $V(x^2 + 1) = \{(\pm i)\}$
- Variety depends on the ideal generated by the polynomials.
- Reason about the Variety by analyzing the Ideals

Definition

Ideals of Polynomials: Let $f_1, f_2, \ldots, f_s \in \mathbb{F}_q[x_1, \ldots, x_n]$. Let

$$J = \langle f_1, f_2 \ldots, f_s \rangle = \{f_1 h_1 + f_2 h_2 + \cdots + f_s h_s\}$$

$J = \langle f_1, f_2 \ldots, f_s \rangle$ is an ideal generated by $f_1, \ldots, f_s$ and the polynomials are called the generators.

- Different generators can generate the same ideal
- $\langle f_1, \cdots, f_s \rangle = \cdots = \langle g_1, \cdots, g_t \rangle$
- Some generators are a "better" representation of the ideal
- A (reduced)**Gröbner basis** is a "canonical" representation of an ideal

Given $F = \{f_1, f_2, \cdots, f_s\}$, Compute a Gröbner Basis (using Buchberger's algorithm) $G = \{g_1, g_2, \cdots, g_t\}$, such that $I = \langle F \rangle = \langle G \rangle$

$$V(F) = V(G)$$

Outline

- Introduction
- Sequential circuit verification requires reachability analysis
- From bit-level to word-level
  - All about math
  - Utilize the math tool: what does Tim get?
  - My approach
- One step back: Sequential arithmetic circuit verification
- New inspiration: UNSAT core extraction using algebraic geometry
- The plan

Elimination Theorem [Tim, *Abstraction using GB*, DAC'14]

- Let $J \subset \mathbb{F}_q[x_1, \ldots, x_d]$ be an ideal
- Let G be a Gröbner basis of J with respect to a lex ordering where $x_1 > x_2 > \cdots > x_d$.
- Then for every $0 \leq l \leq d$:
  - The set $G_l = G \cap \mathbb{F}_q[x_{l+1}, \ldots, x_d]$ is a Gröbner basis of the lth elimination ideal J_l.

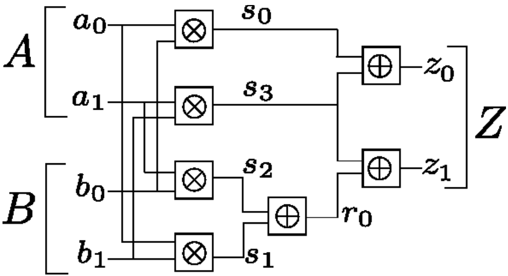
The lth elimination ideal does not contain variables $x_1, \ldots, x_l$, nor do the generators of it.

Elimination Term Ordering Example

- Let ideal $I = \langle f_1, f_2, f_3 \rangle$ where
  - $f_1 = x^2 + y + z - 1$
  - $f_2 = x + y^2 + z - 1$
  - $f_3 = x + y + z^2 - 1$
- The Gröbner basis of I with lex order $(x > y > z)$ is
  - $g_1 = x + y + z^2 - 1$
  - $g_2 = y^2 - y - z^2 + z$
  - $g_3 = 2yz^2 + z^4 - z^2$
  - $g_4 = z^6 - 4z^4 + 4z^3 - z^2$
- Notice that g_2 and g_3 only contain variables y and z
  - Eliminates variable $x \Leftrightarrow \exists_x$ in Boolean formula!
- Similarly, g_4 only contains the variable z and eliminates x and y

Derived from applying elimination theorem to our problem set

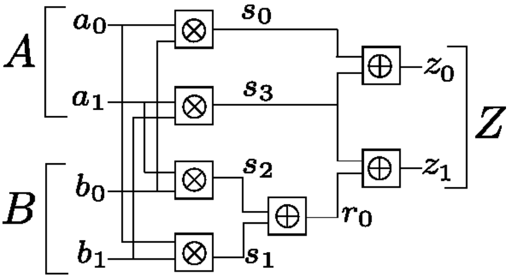
- Given a circuit C implementing $Z = \mathcal{F}(A)$ over $\mathbb{F}_q$
- Using the variable order $x_1 > x_2 > \cdots > x_d > Z > A$
  - $x_1, \ldots, x_d$ are the circuit variables
- Impose a lex term order $>$ on the polynomial ring $R = \mathbb{F}_q[x_1, \ldots, x_d, Z, A]$.
- This elimination term order $>$ is defined as the **Abstraction Term Order**.
- Compute a Gröbner basis G of ideal $(J + J_0)$ using $>$
  - G will contain a polynomial of the form $Z + \mathcal{F}(A)$ ($Z = \mathcal{F}(A)$)
  - $Z = \mathcal{F}(A)$ is a unique, canonical, polynomial representation of C over $\mathbb{F}_q$



$$(z_0 > z_1 > r_0 > s_0 > s_3 > s_1 > s_2 > a_0 > a_1 > b_0 > b_1 > Z > A > B)$$

$$f_1 : s_0 + a_0 \cdot b_0; \quad f_2 : s_1 + a_0 \cdot b_1; \quad f_3 : s_2 + a_1 \cdot b_0; \quad f_4 : s_3 + a_1 \cdot b_1$$
$$f_5 : r_0 + s_1 + s_2; \quad f_6 : z_0 + s_0 + s_3; \quad f_7 : z_1 + r_0 + s_3; \quad f_8 : a_0 + a_1\alpha + A$$
$$f_9 : b_0 + b_1\alpha + B; \quad f_{10} : z_0 + z_1\alpha + Z$$

$$J = \langle f_1, \ldots, f_{10} \rangle$$

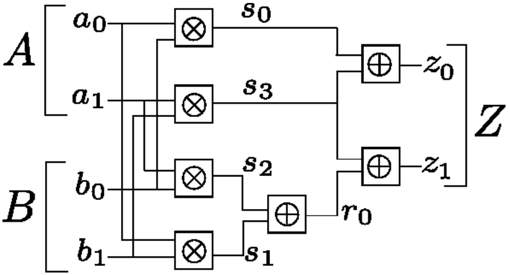


$(z_0 > z_1 > r_0 > s_0 > s_3 > s_1 > s_2 > a_0 > a_1 > b_0 > b_1 > Z > A > B)$

$$f_{11} : a_0^2 + a_0; \quad f_{12} : a_1^2 + a_1\alpha; \quad f_{13} : b_0^2 + b_0; \quad f_{14} : b_1^2 + b_1; \quad f_{15} : s_0^2 + s_0;$$
$$f_{16} : s_1^2 + s_1; \quad f_{17} : s_2^2 + s_2; f_{18} : s_3^2 + s_3; \quad f_{19} : r_0^2 + r_0; \quad f_{20} : z_0^2 + z_0$$
$$f_{21} : z_1^2 + z_1; \quad f_{22} : A^4 + A; \quad f_{23} : B^4 + B; \quad f_{24} : Z^4 + Z$$

$J_0 = \langle f_{11}, \ldots, f_{24} \rangle$

Abstraction Term Order Example

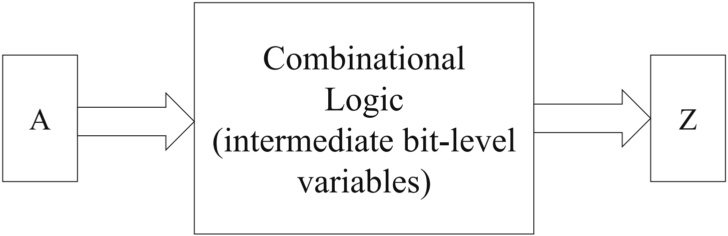


$(z_0 > z_1 > r_0 > s_0 > s_3 > s_1 > s_2 > a_0 > a_1 > b_0 > b_1 > Z > A > B)$
Compute the Gröbner basis, G, of $\{J + J_0\}$ with respect to abstraction term ordering $>$. $G = \{g_1, \ldots, g_{14}\}$

$$g_1 : B^4 + B; \quad g_2 : b_0 + b_1\alpha + B; \quad g_3 : a_0 + a_1\alpha + A; \quad g_4 : A^4 + A;$$
$$g_5 : s_0 + s_1\alpha + s_2(\alpha + 1) + Z; \quad g_6 : r_0 + s_1 + s_2; \quad g_7 : z_1 + r_0 + s_3$$
$$g_7 : z_0 + z_1\alpha + Z; \quad \mathbf{g_9 : Z + A * B}; \quad g_{10} : b_1 + B^2 + B; \quad g_{11} : a_1 + A^2 + A$$
$$g_{12} : s_3 + a_1 b_1; \quad g_{13} : s_2 + a_1 b_1\alpha + a_1 B; \quad g_{14} : s_1 + a_1 b_1\alpha + b_1 A$$

Outline

- Introduction
- Sequential circuit verification requires reachability analysis
- From bit-level to word-level
  - All about math
  - Utilize the math tool: what does Tim get?
  - My approach
- One step back: Sequential arithmetic circuit verification
- New inspiration: UNSAT core extraction using algebraic geometry
- The plan



- From Tim's work I learned:
  - Compute GB with term order *intermediate* $> Z > A$
  - Obtain $Z = \mathcal{F}(A)$
- Inspiration: what if we impose term order *intermediate* $> A > Z$?
- Proposed approach: add the evaluation of A into elimination ideal, then eliminate all but Z, we will get the evaluation of Z!
  - Present state $\leftarrow A$, next state $\leftarrow Z$
  - It is a way to do one-step reachability!

Recall: Breadth-First Traversal Algorithm

ALGORITHM 2: Breadth-first Traversal Algorithm

Input: Transition functions Δ, initial state S^0

$\mathit{from}^0 = \mathit{reached} = S^0$;
repeat
  $i \leftarrow i + 1$;
  $\mathit{to}^i \leftarrow \mathrm{Img}(\Delta, \mathit{from}^{i-1})$;
  $\mathit{new}^i \leftarrow \mathit{to}^i \cap \overline{\mathit{reached}}$;
  $\mathit{reached} \leftarrow \mathit{reached} \cup \mathit{new}^i$;
  $\mathit{from}^i \leftarrow \mathit{new}^i$;
until $\mathit{new}^i == 0$;
return *reached*

We want to implement this algorithm using algebraic geometry approach

Implement Image Function in Algebraic Geometry

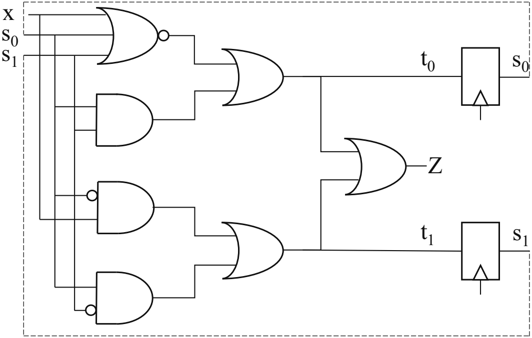
- State variables (word-level) S, T and sets of states such as $from^i$, to^i can always be represented as varieties of ideals.
- Boolean operators can always be converted to operations in $\mathbb{F}_2$

Boolean operator	operation in $\mathbb{F}_2$
$\overline{a}$	$1 + a$
a *and* b	ab
a *or* b	$a + b + ab$
$a \oplus b$	$a + b$

Table : Some Boolean operators and corresponding operations in $\mathbb{F}_2$

- An *elimination ideal* can be built from circuit gates, PS/NS word definition and vanishing polynomials

Example Formulation



Model circuit as polynomials in $\mathbb{F}_2 \subset \mathbb{F}_{2^k}$:

$$t_0 = (\overline{x} \textit{ and } \overline{s_0} \textit{ and } \overline{s_1}) \textit{ or } (s_0 \textit{ and } s_1)$$

$$\implies f_1 : t_0 - (xs_0s_1 + xs_0 + xs_1 + x + s_0 + s_1 + 1)$$

$$t_1 = (\overline{s_0} \textit{ and } x) \textit{ or } (s_0 \textit{ and } \overline{s_1})$$

$$\implies f_2 : t_1 - (xs_0 + x + s_0s_1 + s_0)$$

$$f_3 : S + s_0 + s_1\alpha, \quad f_4 : T + t_0 + t_1\alpha$$

Example Formulation(2)

- Elimination ideal to model Image function for example circuit:
  - Transition functions (bit-level): f_1, f_2
  - Word variable definitions: f_3, f_4
  - Vanishing polynomials: $f_6 : x^2 - x; f_7 : t_0^2 - t_0; f_8 : t_1^2 - t_1; f_9 : S^4 - S; f_{10} : s_0^2 - s_0; f_{11} : s_1^2 - s_1; f_{12} : T^4 - T$
- Add the present state (e.g. initial states in first iteration $f_5 : S$), compute Gröbner basis for ideal $J = \langle f_1, \ldots, f_{12} \rangle$ under elimination term order

$$\textit{intermediate bit-level signals} > \textit{bit-level PIs/POs} > S > T$$

- Result will include a univariate polynomial about *next states* T, e.g. $T^2 + (\alpha + 1)T + \alpha$

Recall: Breadth-First Traversal Algorithm

ALGORITHM 3: Breadth-first Traversal Algorithm

Input: Transition functions Δ, initial state S^0

$\mathit{from}^0 = \mathit{reached} = S^0$;
repeat
    $i \leftarrow i + 1$;
    $\mathit{to}^i \leftarrow \mathrm{Img}(\Delta, \mathit{from}^{i-1})$;
    $\mathit{new}^i \leftarrow \mathit{to}^i \cap \overline{\mathit{reached}}$;
    $\mathit{reached} \leftarrow \mathit{reached} \cup \mathit{new}^i$;
    $\mathit{from}^i \leftarrow \mathit{new}^i$;
until $\mathit{new}^i == 0$;
return *reached*

We want to implement this algorithm using algebraic geometry approach

Definition

(**Sum/Product of Ideals**) If $I = \langle f_1, \ldots, f_r \rangle$ and $J = \langle g_1, \ldots, g_s \rangle$ are ideals in $\mathbb{F}[x_1, \ldots, x_n]$, then the **sum** of I and J is defined as

$$I + J = \langle f_1, \ldots, f_r, g_1, \ldots, g_s \rangle$$

And the **product** of I and J is defined as

$$I \cdot J = \langle f_i g_j \mid 1 \leq i \leq r, 1 \leq j \leq s \rangle$$

Theorem

If I and J are ideals in $\mathbb{F}[x_1, \ldots, x_n]$, then $\mathbf{V}(I + J) = \mathbf{V}(I) \bigcap \mathbf{V}(J)$ and $\mathbf{V}(I \cdot J) = \mathbf{V}(I) \bigcup \mathbf{V}(J)$.

Definition

(**Quotient of Ideals**) If I and J are ideals in $\mathbb{F}[x_1, \ldots, x_n]$, then $I : J$ is the set

$$\{f \in \mathbb{F}[x_1, \ldots, x_n] \mid f \cdot g \in I, \forall g \in J\}$$

and is called the **ideal quotient** of I by J.

Theorem

Let I, J be ideals with vanishing polynomials over $\mathbb{F}_{2^k}[x_1, \ldots, x_n]$, then

$$\mathbf{V}(I : J) = \mathbf{V}(I) - \mathbf{V}(J)$$

Our proposed algorithm of BFS traversal based on algebraic geometry

ALGORITHM 4: Algebraic Geometry based Traversal Algorithm

Input: Input-output circuit characteristic polynomial ideal J_{ckt}, initial state polynomial $\mathcal{F}(S)$

$from^0 = reached = \mathcal{F}(S)$;
repeat
    $i \leftarrow i + 1$;
    $to^i \leftarrow$GB w/ elimination term order$\langle J_{ckt}, J_0, from^{i-1}\rangle$;
    $new^i \leftarrow$generator of $\langle to^i\rangle + (\langle T^4 - T\rangle : \langle reached\rangle)$;
    $reached \leftarrow$generator of $\langle reached\rangle \cdot \langle new^i\rangle$;
    $from^i \leftarrow new^i(S \setminus T)$;
until $new^i == 1$;
return $reached$

Go to example page 2

Go to example page 3

Full Blown traversal of example circuit using algebraic geometry

- Initial state $from^0 = S(\{00\})$
- **Iteration 1:** Compose an elimination ideal J

$$
\begin{aligned}
f_1 &: t_0 - (xs_0s_1 + xs_0 + xs_1 + x + s_0 + s_1 + 1) \\
f_2 &: t_1 - (xs_0 + x + s_0s_1 + s_0) \\
f_3 &: S - s_0 - s_1 \\
f_4 &: T - t_0 - t_1
\end{aligned}
$$

$J_{ckt} = \langle f_1, f_2, f_3, f_4 \rangle$

$$
\begin{aligned}
&f_5 : x^2 - x \\
&f_6 : s_0^2 - s_0, f_7 : s_1^2 - s_1 \\
&f_8 : t_0^2 - t_0, f_9 : t_1^2 - t_1 \\
&f_{10} : S^4 - S, f_{11} : T^4 - T
\end{aligned}
$$

$J_0 = \langle f_5, f_6, \ldots, f_{11} \rangle$

- Elimination term order:

$$\{x, s_0, s_1, t_0, t_1\}\ (all\ bits) > S\ (PS\ word) > T\ (NS\ word)$$

- Compute the reduced GB for $J = J_{ckt} + J_0 + \langle from^0 \rangle$
- Next state

$$to^1 = \langle T^2 + (\alpha + 1)T + \alpha \rangle\ (\{01, 10\})$$

- Complement of formerly reached state:

$$\langle T^4 - T \rangle : \langle T \rangle = \langle T^3 + 1 \rangle\ (\{01, 10, 11\})$$

- Newly reached state:

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

- Update current reached states

$$reach = \langle T \cdot T^2 + (\alpha + 1)T + \alpha \rangle = \langle T^3 + (\alpha + 1)T^2 + \alpha T \rangle\ (\{00, 01, 10\})$$

- Update the present states for next iteration

$$from^1 = \langle S^2 + (\alpha + 1)S + \alpha \rangle$$

The algorithm

- Iteration 2:
  - Next state: $to^2 = \langle T^2 + \alpha T \rangle$ $(\{00, 10\})$
  - The complement of *reached*:

$$\langle T^4 - T \rangle : \langle T^3 + (\alpha + 1)T^2 + \alpha T \rangle = \langle T + 1 + \alpha \rangle \ (\{11\})$$

  - Newly reached state:

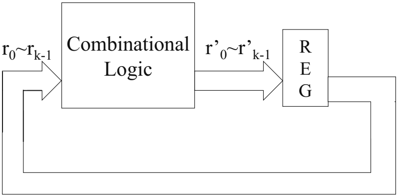
$$\langle T^2 + \alpha T, T + 1 + \alpha \rangle = \langle 1 \rangle$$

- Algorithm terminates
- Return value (final reachable states):

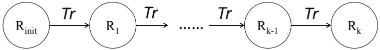
$$reached = \langle T^3 + (\alpha + 1)T^2 + \alpha T \rangle$$

The algorithm

Application on implicit unrolling



(a)

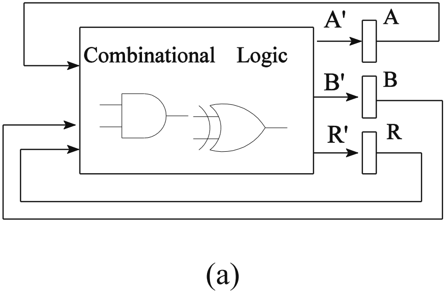


(b)

- State enumeration cannot address sequential arithmetic circuits verification
  - Need to consider various initial states
  - Exact reached states after k clock cycles directly implies desired arithmetic function
- State transitions on simplified model

$$R_k = Tr(R_{k-1}) = Tr(Tr(\cdots Tr(R_{init})\cdots)) = Tr^k(R_{init})$$

SPEC: $R = A_{init} \cdot B_{init} \pmod{P(\alpha)}$ after k clock cycles



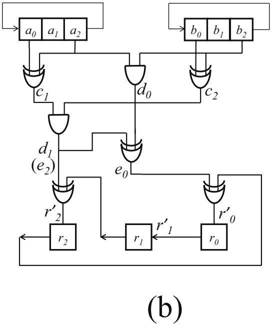
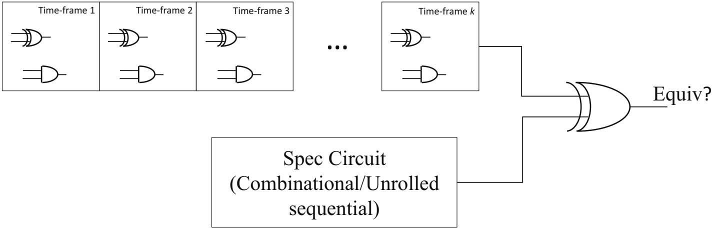


Figure : A 3-bit RH-SMPO and its Moore FSM model

Our approach vs Conventional approach



- Conventional: explicitly unroll k time-frames
  - Bit-blasting!
- New: Implicitly unroll
  - Keep only k polynomials ($R = \mathcal{F}(A, B)$) when unrolling
  - $R_1 = \mathcal{F}(A_{init}, B_{init})$, $R_2 = \mathcal{F}(A_1, B_1) = \mathcal{F}^2(A_{init}, B_{init})$, $\cdots$, $R_k = \mathcal{F}^k(A_{init}, B_{init}) = A_{init} \cdot B_{init}$

Experiment results

Table : Run-time (seconds) for verification of bug-free and buggy RH-SMPO using our approach

Operand size k	33	51	65	81	89	99
#variables	4785	11424	18265	28512	34354	42372
#polynomials	3630	8721	13910	21789	26255	32373
#terms	13629	32793	52845	82539	99591	122958
Runtime(bug-free)	112.6	1129	5243	20724	36096	67021
Runtime(buggy)	112.7	1129	5256	20684	36120	66929

* *Results from X. Sun, et al. "Formal Verification of Sequential Galois Field Arithmetic Circuits using Algebraic Geometry", to be presented in Grenoble, DATE'15*

What I have done for this experiment

- Sequential arithmetic circuits over normal basis
- Standard basis $\rightleftharpoons$ normal basis
- Optimal normal basis
- Design circuit(multiplier) over optimal normal basis
- Verify the function of the circuit

No more details!

- Normal basis representation: $A(a_0, \ldots, a_{k-1}) = \sum_{i=0}^{k-1} a_{n(i)} \beta^{2^i}$
- Normal element: $\beta = \alpha^t$
- Squaring of elements represented in normal bases can be implemented simply by a cyclic right-shift operation.

Example

For $a, b \in \mathbb{F}_{2^k}, (a+b)^2 = a^2 + b^2$. Applying this rule for element squaring:

$$\begin{aligned} B =& (b_0\beta + b_1\beta^2 + b_2\beta^4 + \cdots + b_{k-1}\beta^{2^{k-1}}) \\ B^2 =& b_0^2\beta^2 + b_1^2\beta^4 + b_2^2\beta^8 + \cdots + b_{k-1}^2\beta^{2^k} \\ =& b_{k-1}\beta + b_0\beta^2 + b_1\beta^4 + \cdots + b_{k-2}\beta^{2^{k-1}} \end{aligned}$$

as $\beta^{2^k} = \beta$ by applying Fermat's little theorem to $\mathbb{F}_{2^k}$, and $b_i^2 = b_i$.

- Let $R = \sum_{i=0}^{k-1} r_i \beta^{2^i}$, $A = \sum_{i=0}^{k-1} a_i \beta^{2^i}$, $B = \sum_{i=0}^{k-1} b_i \beta^{2^i}$, then

$$R = A \cdot B = (\sum_{i=0}^{k-1} a_i \beta^{2^i})(\sum_{j=0}^{k-1} b_j \beta^{2^j}) = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_i b_j \beta^{2^i} \beta^{2^j}$$

- Expressions $\beta^{2^i} \beta^{2^j}$ are called cross-product terms. Their normal basis representations are:

$$\beta^{2^i} \beta^{2^j} = \sum_{n=0}^{k-1} \lambda_{ij}^{(n)} \beta^{2^n}, \quad \lambda_{ij}^{(n)} \in \mathbb{F}_2.$$

- The expression for the n^{th} digit of product $R = (r_0, \ldots, r_n, \ldots r_{k-1})$ is:

$$r_n = \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} \lambda_{ij}^{(n)} a_i b_j = A \cdot M_n \cdot B^T, \quad 0 \leq n \leq k-1$$

- $M_n = (\lambda_{ij}^{(n)})$ is a binary $k \times k$ matrix over $\mathbb{F}_2$, and it is called the λ-matrix.
- λ-matrix is unique when k and β are given!

Modified algorithm to verify the function of sequential GF multipliers

ALGORITHM 5: Abstraction via implicit unrolling for Sequential GF circuit verification

Input: Circuit polynomial ideal J, vanishing ideal J_0, initial state ideal $R(=0), \mathcal{G}(A_{init}), \mathcal{H}(B_{init})$

$from_0(R, A, B) = \langle R, \mathcal{G}(A_{init}), \mathcal{H}(B_{init}) \rangle$;

$i = 0$;

repeat

    $i \leftarrow i + 1$;

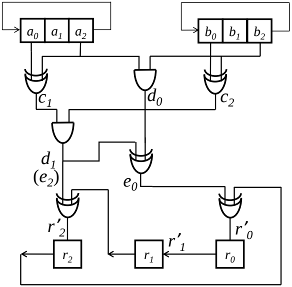
    $G \leftarrow$GB($\langle J + J_0 + from_{i-1}(R, A, B) \rangle$) with ATO;

    $to_i(R', A', B') \leftarrow G \cap \mathbb{F}_{2^k}[R', A', B', R, A, B]$;

    $from_i \leftarrow to_i(\{R, A, B\} \setminus \{R', A', B'\})$;

until $i == k$;

return $from_k(R_{final})$



- The elimination ideal (first iteration):

$$
\begin{aligned}
J = & d_0 + b_2 \cdot a_2, c_1 + a_0 + a_2, c_2 + b_0 + b_2, d_1 + c_1 \cdot c_2, \\
& e_0 + d_0 + d_1, e_2 + d_1, r'_0 + r_2 + e_0, r'_1 + r_0, r'_2 + r_1 + e_2, \\
& A + a_0\alpha^3 + a_1\alpha^6 + a_2\alpha^{12}, B + b_0\alpha^3 + b_1\alpha^6 + b_2\alpha^{12}, \\
& R + r_0\alpha^3 + r_1\alpha^6 + r_2\alpha^{12}, R' + r'_0\alpha^3 + r'_1\alpha^6 + r'_2\alpha^{12};
\end{aligned}
$$

- "J_0" is the ideal of vanishing polynomials in all bit-level variables (e.g. $a_0^2 - a_0$) and word-level variables (e.g. $A^8 - A$).
- $from_0 = \{R, A_{init} + a_0\alpha^3 + a_1\alpha^6 + a_2\alpha^{12}, B_{init} + b_0\alpha^3 + b_1\alpha^6 + b_2\alpha^{12}\}$
- $to_1 : R' + (\alpha^2)A_{init}^4 B_{init}^4 + (\alpha^2 + \alpha)A_{init}^4 B_{init}^2 + (\alpha^2 + \alpha)A_{init}^4 B_{init} + (\alpha^2 + \alpha)A_{init}^2 B_{init}^4 + (\alpha^2 + \alpha + 1)A_{init}^2 B_{init}^2 + (\alpha^2)A_{init}^2 B_{init} + (\alpha^2 + \alpha)A_{init} B_{init}^4 + (\alpha^2)A_{init} B_{init}^2$
- $from_1 = \{R' + (\alpha^2)A_{init}^4 B_{init}^4 + (\alpha^2 + \alpha)A_{init}^4 B_{init}^2 + (\alpha^2 + \alpha)A_{init}^4 B_{init} + (\alpha^2 + \alpha)A_{init}^2 B_{init}^4 + (\alpha^2 + \alpha + 1)A_{init}^2 B_{init}^2 + (\alpha^2)A_{init}^2 B_{init} + (\alpha^2 + \alpha)A_{init} B_{init}^4 + (\alpha^2)A_{init} B_{init}^2, A_{init} + a_2\alpha^3 + a_0\alpha^6 + a_1\alpha^{12}, B_{init} + b_2\alpha^3 + b_0\alpha^6 + b_1\alpha^{12}\}$
- $\cdots$
- After 3 iterations: $to_3 = \{R' + A_{init} B_{init},\ A_{init} + a_0'\alpha^3 + a_1'\alpha^6 + a_2'\alpha^{12},\ B_{init} + b_0'\alpha^3 + b_1'\alpha^6 + b_2'\alpha^{12}\}$

Outline

- Introduction
- Sequential circuit verification requires reachability analysis
- From bit-level to word-level
- One step back: Sequential arithmetic circuit verification
- New inspiration: UNSAT core extraction using algebraic geometry
- The plan

An example of abstraction refinement algorithm

ALGORITHM 6: *k*-BMC with Abstraction Refinement (L. Zhang Thesis)

Input: M is the original machine, p is the property to check, k is the number of steps in k-BMC

```
1  k = InitValue;
2  if k-BMC(M, p, k) is SAT then
3      return "Found error trace"
4  else
5      Extract UNSAT proof 𝒫 of k-BMC;
6      M′ = ABSTRACT(M, 𝒫);
7  end
8  if MODEL-CHECK(M′, p) returns PASS then
9      return "Passing property"
10 else
11     Increase bound k;
12     goto Line 2;
13 end
```

- k-BMC: unroll the machine for k times, represent reachable states with CNF formula to check property p

$$I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge \neg p$$

- UNSAT core: a subset of clauses that is still UNSAT
- State variables not in UNSAT core: no matter what their values are, p will NOT be violated
- In abstracted model, ignore these “irrelevant” variables

An example of abstraction refinement algorithm(3)

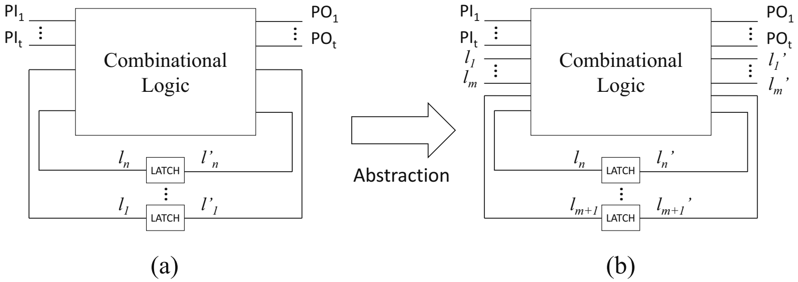


Figure : Abstraction by reducing latches

- Remove "irrelevant" latches, reduce state space
- Provide an over-approximation
- This algorithm requires UNSAT core extraction

Buchberger's Algorithm

INPUT : $F = \{f_1, \ldots, f_s\}$
OUTPUT : $G = \{g_1, \ldots, g_t\}$
$G := F;$
REPEAT
  $G' := G$
  For each pair $\{f, g\}, f \neq g$ in G' DO
    $S(f,g) \xrightarrow{G'}_{+} r$
    IF $r \neq 0$ THEN $G := G \cup \{r\}$
UNTIL $G = G'$

- S-poly: $S(f,g) = \frac{L}{lt(f)} \cdot f - \frac{L}{lt(g)} \cdot g$
  $L = \mathrm{LCM}(lm(f), lm(g))$, $lm(f)$: leading monomial of f
- Multivariate division: $f \xrightarrow{g} r$: $lm(g) | lm(f) \rightarrow r = f - \frac{lt(f)}{lt(g)} g$

- Translate CNF clauses to polynomials in $\mathbb{F}_2$

$$
\begin{array}{lcl}
c_1 : \bar{a} \vee \bar{b} & & f_1 : ab \\
c_2 : a \vee \bar{b} & & f_2 : ab + a \\
c_3 : \bar{a} \vee b & & f_3 : ab + b \\
c_4 : a \vee b & & f_4 : ab + a + b + 1 \\
c_5 : x \vee y & \Longrightarrow & f_5 : xy + y + x + 1 \\
c_6 : y \vee z & & f_6 : yz + y + z + 1 \\
c_7 : b \vee \neg y & & f_7 : by + y \\
c_8 : a \vee x \vee \neg z & & f_8 : axz + az + xz + z
\end{array}
$$

- Compute a GB using Buchberger's algorithm
- Term ordering: $a > b > x > y > z$
- 1. Compute $Spoly(f_1, f_2) \xrightarrow{F}_+ r_1 = a$
  2. Update $F = F \cup r_1$
  3. Compute $Spoly(f_1, f_3) \xrightarrow{F}_+ r_2 = b$
  4. Update $F = F \cup r_2$
  5. Use a directed acyclic graph (DAG) to represent the process to get r_1, r_2
  6. Compute $Spoly(f_1, f_4) = s_3 = a + b + 1$, $a + b + 1$ can be reduced r_1 , the intermediate remainder $r_3 = b + 1$. Then divided by r_2, and the remainder is "1", terminate the Buchberger's algorithm
  7. Draw a DAG depicting the process through which we obtain remainder "1". From leaf "1" we backtrace the graph to roots f_1, f_2, f_3, f_4.

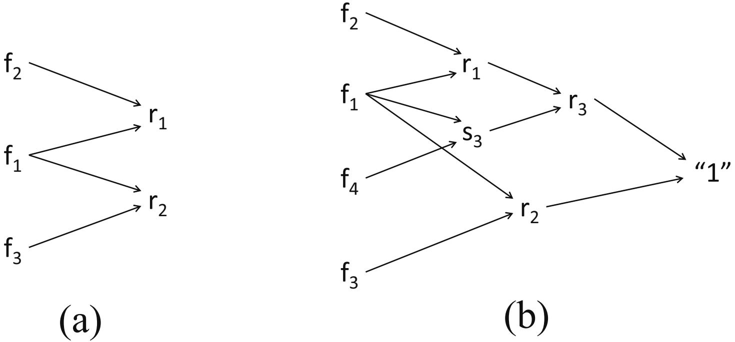


Figure : DAG representing Spoly computations and multivariate divisions

Observation when executing Buchberger's algorithm(3)

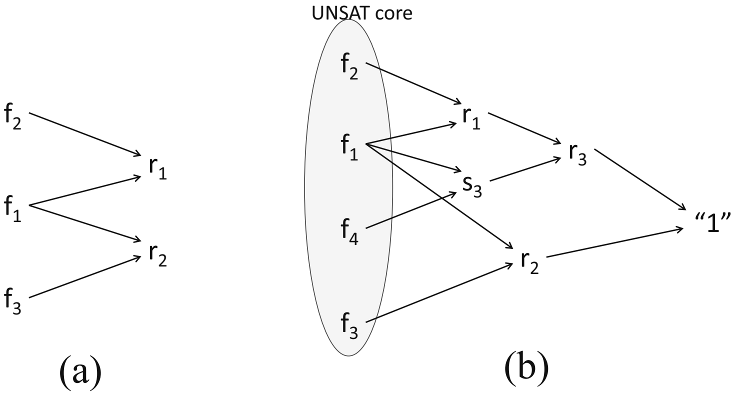


Figure : DAG representing Spoly computations and multivariate divisions

Proposed algorithm to extract UNSAT core

ALGORITHM 7: Extract UNSAT core using a variation of Buchberger's algorithm

Input: A set of polynomials $F = \{f_1, f_2, \ldots, f_s\}$
Output: An UNSAT core $\{f_{m_1}, f_{m_2}, \ldots, f_{m_t}\}$

repeat
    Pick a pair $f_i, f_j \in F$ that has never been computed S-poly;
    if $Spoly(f_i, f_j) \xrightarrow{F}_+ r_l \neq 0$ **then**
        $F = F \cup r_l$;
        Create a DAG G_l with f_i, f_j as roots, r_l as leaf, recording the S-poly, all intermediate remainders and $f_k \in F$ that cancel monomial terms in the S-poly;
    end
until $r_l == 1$;
Backward traverse the DAG for remainder "1", replace r_l with corresponding DAG G_l;
return *All roots*

Outline

- Introduction
- Sequential circuit verification requires reachability analysis
- From bit-level to word-level
- One step back: Sequential arithmetic circuit verification
- New inspiration: UNSAT core extraction using algebraic geometry
- The plan

Objectives

- Explore an implementation of algebraic geometry based reachability analysis
- CAD tool design
    - SINGULAR's data structure not optimized $\rightarrow$ design a standalone CAD tool implemented in C++
    - Specifically aims to solve word-level sequential verification problems
    - Lower complexity: borrow techniques from [T. Pruss, *Abstraction*, DAC'14], [J. Lv, *Equivalence*, TCAD'13] and [C. Eder, *F-4 reduction*, ISSAC'11]
- Fine-tune the tool for sequential GF arithmetic circuits verification
- Explore a new abstraction-refinement paradigm based on information from UNSAT cores

Time table

- Current status: Experiments have been performed to run implicit state enumeration on sequential circuits benchmarks such as ISCAS'89 circuits. Current problem is the algorithm spends too much time on multivariate division procedure;
- Spring 2015: Implement the tool which can efficiently do multivariate division and test the performance of our tool based on [Tim, *Abstraction*, DAC'14] approach;
- Summer 2015: Integrate the refined multivariate division routine into our verification tool, test its performance on circuits with various sizes;
- Fall 2015: Evaluate data and write the dissertation.