Hardware Datapath Verification using Commutative Algebra and Algebraic Geometry

Priyank Kalla

Associate Professor
Electrical and Computer Engineering, University of Utah
kalla@ece.utah.edu
http://www.ece.utah.edu/~kalla

A tutorial presented at the joint session of SAT, DIFTS and FMCAD 2015

Research funded in part by the US National Science Foundation

The Core Message of the Tutorial

- Modern Algebraic Geometry
  - Study of the *zeros* of multivariate polynomials
  - Infeasible to enumerate the solutions
  - Reason about various properties of the solution-sets
  - Employ techniques that lie at the cross-roads of number-theory, commutative algebra, geometry
- Use of Gröbner bases as a powerful reasoning engine
- Hardware datapaths possess structure and symmetry in the problem
- Gröbner bases help identify this structure/symmetry
- Exploit this structure/symmetry to engineer domain-specific implementations for datapath verification
- Enables verification of hard datapath verification problems

Tutorial Objective and Agenda

- Formal verification of datapath implementations (RTL)
  - Word-level abstractions from designs, symbolic techniques
  - Model bit-precise semantics at word-level
  - Applications: Cryptography, Error Control Circuits, Signal Processing

Tutorial Objective and Agenda

- Formal verification of datapath implementations (RTL)
  - Word-level abstractions from designs, symbolic techniques
  - Model bit-precise semantics at word-level
  - Applications: Cryptography, Error Control Circuits, Signal Processing
- Equivalence check: specification (*Spec*) vs implementation (*Impl*)
  - *Spec* and *Impl*: same function?
  - RTL: functions over k-bit vectors
    - k-bit vector $\mapsto$ Boolean domain $\mathbb{B}^k$
    - k-bit vector $\mapsto$ integers $\pmod{2^k} = \mathbb{Z}_{2^k}$
    - k-bit vector $\mapsto$ Galois (Finite) field $\mathbb{F}_{2^k}$

Tutorial Objective and Agenda

- Formal verification of datapath implementations (RTL)
  - Word-level abstractions from designs, symbolic techniques
  - Model bit-precise semantics at word-level
  - Applications: Cryptography, Error Control Circuits, Signal Processing
- Equivalence check: specification (*Spec*) vs implementation (*Impl*)
  - *Spec* and *Impl*: same function?
  - RTL: functions over *k*-bit vectors
    - k-bit vector $\mapsto$ Boolean domain $\mathbb{B}^k$
    - k-bit vector $\mapsto$ integers $\pmod{2^k} = \mathbb{Z}_{2^k}$
    - k-bit vector $\mapsto$ Galois (Finite) field $\mathbb{F}_{2^k}$
- Approach: **Computer Algebra Techniques**
  - Model: Polynomial functions over $f : \mathbb{Z}_{2^k} \rightarrow \mathbb{Z}_{2^k}$ or $f : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_{2^k}$
  - Devise decision procedures for polynomial function equivalence
  - Commutative algebra, algebraic geometry + contemporary verification

Verification of Galois field circuits

- Wide applications of Galois field (GF) circuits
  - **Cryptography**: RSA, Elliptic Curve Cryptography (ECC)
  - Error Correcting Codes, Digital Signal Processing, etc.

Verification of Galois field circuits

- Wide applications of Galois field (GF) circuits
  - **Cryptography**: RSA, Elliptic Curve Cryptography (ECC)
  - Error Correcting Codes, Digital Signal Processing, etc.
- Bugs in GF arithmetic circuits can leak secret keys
  - *Biham et al., "Bug Attacks", Crypto 2008* [1]

- Wide applications of Galois field (GF) circuits
  - **Cryptography**: RSA, Elliptic Curve Cryptography (ECC)
  - Error Correcting Codes, Digital Signal Processing, etc.
- Bugs in GF arithmetic circuits can leak secret keys
  - *Biham et al., "Bug Attacks", Crypto 2008* [1]
- Target problems
  - Given Galois field $\mathbb{F}_{2^k}$, polynomial f, and circuit C
  - Verify: circuit C implements f; or find the bug
  - Given circuit C, with k-bit inputs and outputs
    - Derive a polynomial representation for C over $f : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_{2^k}$
    - Word-level abstraction as a canonical polynomial representation

Verification of Galois field circuits

- Wide applications of Galois field (GF) circuits
  - **Cryptography**: RSA, Elliptic Curve Cryptography (ECC)
  - Error Correcting Codes, Digital Signal Processing, etc.
- Bugs in GF arithmetic circuits can leak secret keys
  - *Biham et al., "Bug Attacks", Crypto 2008* [1]
- Target problems
  - Given Galois field $\mathbb{F}_{2^k}$, polynomial f, and circuit C
  - Verify: circuit C implements f; or find the bug
  - Given circuit C, with k-bit inputs and outputs
    - Derive a polynomial representation for C over $f : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_{2^k}$
    - Word-level abstraction as a canonical polynomial representation
- Solutions employing Nullstellensatz over $\mathbb{F}_{2^k}$ + Gröbner Basis methods
  - Focus: Techniques and implementations to address scalability
  - Term-orders, custom F_4-style reduction

Galois Field Overview

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

- $0, 1$ elements, associate, commutative, distributive laws
- Closure property: $+, -, \times$, inverse $(\div)$

Our interest: $\mathbb{F}_q = \mathbb{F}_{2^k}$ $(q = 2^k)$

- $\mathbb{F}_{2^k}$: k-dimensional extension of $\mathbb{F}_2 = \{0, 1\}$
  - k-bit bit-vector, AND/XOR arithmetic
  - Efficient crypto-hardware implementations

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
- Operations performed $\pmod{P(x)}$ and coefficients reduced $\pmod{2}$

Example Field Construction: $\mathbb{F}_8$

Construct: $\mathbb{F}_{2^3} = \mathbb{F}_2[x] \pmod{P(x) = x^3 + x + 1}$
Consider any polynomial $A(x) \in \mathbb{F}_2[x]$
$A(x) \pmod{x^3 + x + 1} = a_2x^2 + a_1x + a_0$. 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 $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$

- Every function is a polynomial function over $\mathbb{F}_q$
- Consider 1-bit right-shift operation $Z[2:0] = A[2:0] >> 1$

$\{a_2a_1a_0\}$	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	$\alpha^2 + 1$	$\rightarrow$	010	α
110	$\alpha^2 + \alpha$	$\rightarrow$	011	$\alpha + 1$
111	$\alpha^2 + \alpha + 1$	$\rightarrow$	011	$\alpha + 1$

Polynomial Functions $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$

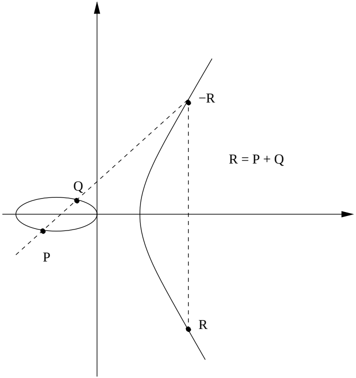
- Every function is a polynomial function over $\mathbb{F}_q$
- Consider 1-bit right-shift operation $Z[2:0] = A[2:0] >> 1$

$\{a_2a_1a_0\}$	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	$\alpha+1$

$Z = (\alpha^2+1)A^4 + (\alpha^2+1)A^2$ over $\mathbb{F}_{2^3}$ where $\alpha^3+\alpha+1=0$

Encryption, Decryption & Authentication using point addition: $P + Q = R$

$$y^2 + xy = x^3 + ax^2 + b \text{ over } \mathbb{F}_{2^k}$$



Compute Slope: $\dfrac{y_2 - y_1}{x_2 - x_1}$

Computation of inverses over $\mathbb{F}_{2^k}$ is expensive

Point addition using Projective Co-ordinates

- Curve: $Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4$ over $\mathbb{F}_{2^k}$
- Let $(X_3,\ Y_3,\ Z_3) = (X_1,\ Y_1,\ Z_1) + (X_2,\ Y_2,\ 1)$

$$
\begin{aligned}
A &= Y_2 \cdot Z_1^2 + Y_1 \\
B &= X_2 \cdot Z_1 + X_1 \\
C &= Z_1 \cdot B \\
D &= B^2 \cdot (C + aZ_1^2) \\
{\color{red}Z_3} &= C^2
\end{aligned}
\qquad
\begin{aligned}
E &= A \cdot C \\
{\color{red}X_3} &= A^2 + D + E \\
F &= X_3 + X_2 \cdot Z_3 \\
G &= X_3 + Y_2 \cdot Z_3 \\
{\color{red}Y_3} &= E \cdot F + Z_3 \cdot G
\end{aligned}
$$

Point addition using Projective Co-ordinates

- Curve: $Y^2 + XYZ = X^3 Z + aX^2 Z^2 + bZ^4$ over $\mathbb{F}_{2^k}$
- Let $(X_3,\ Y_3,\ Z_3) = (X_1,\ Y_1,\ Z_1) + (X_2,\ Y_2,\ 1)$

$$
\begin{aligned}
A &= Y_2 \cdot Z_1^2 + Y_1 & E &= A \cdot C \\
B &= X_2 \cdot Z_1 + X_1 & X_3 &= A^2 + D + E \\
C &= Z_1 \cdot B & F &= X_3 + X_2 \cdot Z_3 \\
D &= B^2 \cdot (C + aZ_1^2) & G &= X_3 + Y_2 \cdot Z_3 \\
Z_3 &= C^2 & Y_3 &= E \cdot F + Z_3 \cdot G
\end{aligned}
$$

- No inverses, just addition and multiplication
- Verify ECC hardware primitives: circuits for GF Multiplication and exponentiation
- Challenge: Large datapath size, from $k = 163$-bits to 1000+ bits

Field polynomials of $\mathbb{F}_q$

Theorem (Fermat's Little Theorem over $\mathbb{F}_q$)

For any element $\alpha \in \mathbb{F}_q$, then $\alpha^q = \alpha$.

Vanishing Polynomials

The polynomial $(x^q - x)$ vanishes $(= 0)$ on all points in $\mathbb{F}_q$. We call $(x^q - x)$ a vanishing polynomial of $\mathbb{F}_q$.

Computer Algebra Terminology

Let $\mathbb{F}_q = GF(2^k)$, and $\overline{\mathbb{F}_q}$ be its closure

- $\mathbb{F}_q[x_1, \dots, x_n]$: ring of all polynomials with coefficients in $\mathbb{F}_q$
- Polynomial $f = c_1 X_1 + c_2 X_2 + \cdots + c_t X_t$
  - Coefficients c_i, monomial $X = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}, \alpha_i \in Z_{\geq 0}$
  - A monomial ordering is imposed on the ring, so $f : X_1 > X_2 > \cdots > X_t$
  - Leading term $lt(f) = c_1 X_1, \quad tail(f) = c_2 X_2 + \cdots + c_t X_t$
  - Leading coefficient $lt(f) = c_1$ and leading monomial $lm(f) = X_1$

Let $\mathbb{F}_q = GF(2^k)$, and $\overline{\mathbb{F}_q}$ be its closure

- $\mathbb{F}_q[x_1, \dots, x_n]$: ring of all polynomials with coefficients in $\mathbb{F}_q$
- Polynomial $f = c_1 X_1 + c_2 X_2 + \cdots + c_t X_t$
  - Coefficients c_i, monomial $X = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}, \alpha_i \in Z_{\geq 0}$
  - A monomial ordering is imposed on the ring, so $f : X_1 > X_2 > \cdots > X_t$
  - Leading term $lt(f) = c_1 X_1, \quad tail(f) = c_2 X_2 + \cdots + c_t X_t$
  - Leading coefficient $lt(f) = c_1$ and leading monomial $lm(f) = X_1$
- Example: $f = 2x^2yz + 3xy^3 - 2x^3$
  - LEX with $x > y > z$: $\quad f = -2x^3 + 2x^2yz + 3xy^3$
  - DEGLEX with $x > y > z$: $\quad f = 2x^2yz + 3xy^3 - 2x^3$
  - DEGREVLEX with $x > y > z$: $\quad f = 3xy^3 + 2x^2yz - 2x^3$

Let $\mathbb{F}_q = GF(2^k)$, and $\overline{\mathbb{F}_q}$ be its closure

- $\mathbb{F}_q[x_1, \dots, x_n]$: ring of all polynomials with coefficients in $\mathbb{F}_q$
- Polynomial $f = c_1 X_1 + c_2 X_2 + \cdots + c_t X_t$
  - Coefficients c_i, monomial $X = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}, \alpha_i \in Z_{\geq 0}$
  - A monomial ordering is imposed on the ring, so $f : X_1 > X_2 > \cdots > X_t$
  - Leading term $lt(f) = c_1 X_1, \quad tail(f) = c_2 X_2 + \cdots + c_t X_t$
  - Leading coefficient $lt(f) = c_1$ and leading monomial $lm(f) = X_1$
- Example: $f = 2x^2yz + 3xy^3 - 2x^3$
  - LEX with $x > y > z: \quad f = -2x^3 + 2x^2yz + 3xy^3$
  - DEGLEX with $x > y > z: \quad f = 2x^2yz + 3xy^3 - 2x^3$
  - DEGREVLEX with $x > y > z: \quad f = 3xy^3 + 2x^2yz - 2x^3$
- Leading terms $lt(f)$ play an important role

Polynomial Division as Cancellation of Terms

Divide $f = x^3 - 2x^2 + 2x + 8$ by $g = 2x^2 + 3x + 1$

Polynomial Division as Cancellation of Terms

Divide $f = x^3 - 2x^2 + 2x + 8$ by $g = 2x^2 + 3x + 1$

$$
\begin{array}{r|rrrr}
\multicolumn{1}{r}{} & & & \frac{1}{2}x & -\frac{7}{4} \\
\cline{2-5}
2x^2 + 3x + 1 & x^3 & -2x^2 & +2x & +8 \\
 & -x^3 & -\frac{3}{2}x^2 & -\frac{1}{2}x & \\
\cline{2-4}
 & & -\frac{7}{2}x^2 & +\frac{3}{2}x & +8 \\
 & & \frac{7}{2}x^2 & +\frac{21}{4}x & +\frac{7}{4} \\
\cline{3-5}
 & & & \frac{27}{4}x & +\frac{39}{4}
\end{array}
$$

Polynomial Division as Cancellation of Terms

Divide $f = x^3 - 2x^2 + 2x + 8$ by $g = 2x^2 + 3x + 1$

$$
\begin{array}{rrrrr}
 & & & \frac{1}{2}x & -\frac{7}{4} \\ \cline{2-5}
2x^2+3x+1 \big) & x^3 & -2x^2 & +2x & +8 \\
 & -x^3 & -\frac{3}{2}x^2 & -\frac{1}{2}x & \\ \cline{2-4}
 & & -\frac{7}{2}x^2 & +\frac{3}{2}x & +8 \\
 & & \frac{7}{2}x^2 & +\frac{21}{4}x & +\frac{7}{4} \\ \cline{3-5}
 & & & \frac{27}{4}x & +\frac{39}{4}
\end{array}
$$

- The key step in division: $r = f - \frac{lt(f)}{lt(g)} \cdot g$, denoted $f \xrightarrow{g} r$
- Similarly divide f by a set of polynomials $F = \{f_1, \ldots, f_s\}$
- Denoted: $f \xrightarrow{f_1,\ldots,f_s}_+ r$
  - Remainder r is reduced: no term in r is divisible by $lt(f_i)$

Varieties

- We will model the circuit with a set of polynomials $F = \{f_1, \dots, f_s\}$
- In verification, we need solutions to the system of equations:

$$\begin{aligned} f_1 &= 0 \\ f_2 &= 0 \\ &\vdots \\ f_s &= 0 \end{aligned}$$

- Variety: Set of ALL solutions to a given system of polynomial equations: $V(f_1, \dots, f_s)$
- 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

$$J = \langle f_1, f_2 \ldots, f_s \rangle = \{f_1 h_1 + f_2 h_2 + \cdots + f_s h_s : h_i \in \mathbb{F}_q[x_1, \ldots, x_n]\}$$

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

Definition

Ideal Membership: Let $f, f_1, f_2, \ldots, f_s \in \mathbb{F}_q[x_1, \ldots, x_n]$. Let $J = \langle f_1, f_2 \ldots, f_s \rangle$ be an ideal $\subset \mathbb{F}_q[x_1, \ldots, x_n]$.

If $f = f_1 h_1 + f_2 h_2 + \cdots + f_s h_s$, then $f \in J$.

Let $f_1(\mathbf{a}) = f_2(\mathbf{a}) = \cdots = f_s(\mathbf{a}) = 0$; if $f \in \langle f_1, \ldots, f_s \rangle$ then $f(\mathbf{a}) = 0$

Ideal Membership Test Requires a Gröbner Basis

- Different generators can generate the same ideal
- $\langle f_1, \cdots, f_s \rangle = \cdots = \langle h_1, \ldots, h_r \rangle = \cdots = \langle g_1, \cdots, g_t \rangle$, such that $V(f_1, \ldots, f_s) = V(h_1, \ldots, h_r) = V(g_1, \ldots, g_t)$
- Some generators are a "better" representation of the ideal
- A **Gröbner basis** is a "canonical" representation of an ideal

Ideal Membership Test Requires a Gröbner Basis

- Different generators can generate the same ideal
- $\langle f_1, \cdots, f_s \rangle = \cdots = \langle h_1, \ldots, h_r \rangle = \cdots = \langle g_1, \cdots, g_t \rangle$, such that $V(f_1, \ldots, f_s) = V(h_1, \ldots, h_r) = V(g_1, \ldots, g_t)$
- Some generators are a "better" representation of the ideal
- A **Gröbner basis** is a "canonical" representation of an ideal

Definition (Gröbner Basis)

$$G = \{g_1, \ldots, g_t\} = GB(J) \iff \forall f \in J, \exists g_i \text{ s.t. } lm(g_i) \mid lm(f)$$

Definition (Gröbner Basis for Ideal Membership Test)

$$G = GB(J) \iff \forall f \in J, f \xrightarrow{g_1, g_2, \cdots, g_t}_{+} 0$$

Implies a "decision procedure" for ideal membership

Buchberger's Algorithm

INPUT : $F = \{f_1, \ldots, f_s\}$, and term order $>$

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 $G = G'$

$$S(f, g) = \frac{L}{lt(f)} \cdot f - \frac{L}{lt(g)} \cdot g$$

$L = \text{LCM}(lm(f), lm(g))$, $lm(f)$: leading monomial of f

Gröbner basis for Verification

Intuitively:

- Given a property to verify: f
- Polynomials corresponding to the circuit: $f_1, \ldots, f_s$
  - Generate ideal $J = \langle f_1, \ldots, f_s \rangle$
- Formulate verification test: Is $f \in J$?
- Compute Gröbner basis $G = GB(J) = \{g_1, \ldots, g_t\}$
- Test if $f \xrightarrow{g_1,\ldots,g_t}_+ 0$?

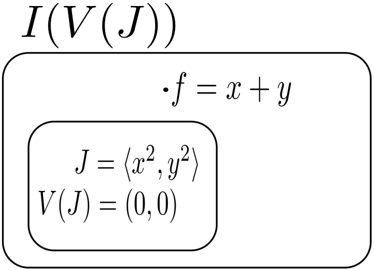
Gröbner basis for Verification

Intuitively:

- Given a property to verify: f
- Polynomials corresponding to the circuit: $f_1, \ldots, f_s$
  - Generate ideal $J = \langle f_1, \ldots, f_s \rangle$
- Formulate verification test: Is $f \in J$?
- Compute Gröbner basis $G = GB(J) = \{g_1, \ldots, g_t\}$
- Test if $f \xrightarrow{g_1, \ldots, g_t}_+ 0$?

However, it is not sufficient to analyze ideal J, but analyze ideal $I(V(J))$

Need to Analyze $I(V(J))$



- Consider ideal $J = \langle x^2, y^2 \rangle$ with $V(J) = (0, 0)$
- Let $f(x, y) = x + y$, then $f(0, 0) = 0$; i.e. f vanishes on $V(J)$
- But $f \notin J$, as no combination of x^2, y^2 can generate $x + y$
- So, $f \in I(V(J))$.

$I(V(J))$ and Nullstellensatz

Definition (Ideals of polynomials that vanish on V)

Given an ideal $J = \langle f_1, \ldots, f_s \rangle \subset R = \mathbb{F}_q[x_1, \ldots, x_n]$, let $V(J)$ be its variety. Then:
$I(V(J)) = \{f \in R : f(\mathbf{a}) = 0 \ \ \forall \mathbf{a} \in V(J)\}$

- If f vanishes on $V(J)$, then $f \in I(V(J))$
- Given ideal J, not easy to find $I(V(J))$ [unless operating over $\mathbb{F}_q$]

Theorem (Strong Nullstellensatz over $\mathbb{F}_q$)

Over Galois fields $\mathbb{F}_q$, $I(V_{\mathbb{F}_q}(J)) = J + J_0$, where:

- $J = \langle f_1, \ldots, f_s \rangle$ *is an arbitrary ideal*
- $J_0 = \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$ *is the ideal of vanishing polynomials in* $\mathbb{F}_q$

Proof: $I(V_{\mathbb{F}_q}(J)) = I(V_{\overline{\mathbb{F}_q}}(J + J_0)) = \sqrt{J + J_0} = J + J_0$

- Given specification polynomial: $f : Z = A \cdot B \pmod{P(x)}$ over $\mathbb{F}_{2^k}$, for given k, and given $P(x)$, s.t. $P(\alpha) = 0$
- Given circuit implementation C
  - Primary inputs: $A = \{a_0, \ldots, a_{k-1}\}, B = \{b_0, \ldots, b_{k-1}\}$
  - Primary Output $Z = \{z_0, \ldots, z_{k-1}\}$
  - $A = a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_{k-1}\alpha^{k-1}$
  - $B = b_0 + b_1\alpha + \cdots + b_{k-1}\alpha^{k-1}$, $Z = z_0 + z_1\alpha + \cdots + z_{k-1}\alpha^{k-1}$
- Does the circuit C implement f?

Mathematically:

- Model the circuit (gates) as polynomials: $f_1, \ldots, f_s$
  $J = \langle f_1, \ldots, f_s \rangle \subset \mathbb{F}_{2^k}[x_1, \ldots, x_n]$
- Does f agree with solutions to $f_1 = f_2 = \cdots = f_s = 0$?
- Does f vanish on the Variety $V_{\mathbb{F}_q}(J)$?
- Is $f \in I(V_{\mathbb{F}_q}(J)) = J + J_0$ or is $f \xrightarrow{GB(J+J_0)}_+ 0$?

Example Verification Formulation

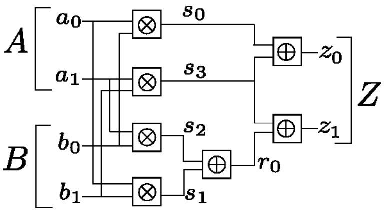


Figure: A GF multiplier over $\mathbb{F}_{2^2}$

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

$$
\begin{aligned}
z_0 = s_0 \oplus s_3; \quad &\mapsto \quad f_1 : z_0 + s_0 + s_3 \\
z_1 = r_0 \oplus s_3; \quad &\mapsto \quad f_2 : z_1 + r_0 + s_3 \\
&\vdots \\
s_0 = a_0 \wedge b_0; \quad &\mapsto \quad f_7 : s_0 + a_0 \cdot b_0 \\
A = a_0 + a_1\alpha; \quad &\mapsto f_8 : A + a_0 + a_1\alpha \\
B = b_0 + b_1\alpha; \quad &\mapsto f_9 : B + b_0 + b_1\alpha \\
Z = z_0 + z_1\alpha; \quad &\mapsto f_{10} : Z + z_0 + z_1\alpha
\end{aligned}
$$

Ideal $J_0 = \langle z_0^2 - z_0, s_0^2 - s_0, \dots, A^{2^k} - A, B^{2^k} - B, Z^{2^k} - Z \rangle$

Verification problem: Check if $f \xrightarrow{GB(J+J_0)}_+ 0$?

- In general, Complexity of Gröbner basis: doubly-exponential
  - Degree of polynomials in G is bounded by $2(\frac{1}{2}d^2 + d)^{2^{n-1}}$ [2]
- However, for zero dimensional ideals: single-exponential complexity
  - For $J \subset \mathbb{F}_q[x_1, \dots, x_n]$, Complexity $GB(J + J_0) : q^{O(n)}$
- $GB(J + J_0)$ computation explodes for 32-bit circuits
- GB complexity very sensitive to **term ordering**

Let $f = 2x^2yz + 3xy^3 - 2x^3$

- LEX $x > y > z$: $f = -\mathbf{2x^3} + 2x^2yz + 3xy^3$
- DEGLEX $x > y > z$: $f = \mathbf{2x^2yz} + 3xy^3 - 2x^3$
- DEGREVLEX $x > y > z$: $f = \mathbf{3xy^3} + 2x^2yz - 2x^3$

Recall, S-polynomial depends on term ordering:

$$S(f,g) = \frac{L}{lt(f)} \cdot f - \frac{L}{lt(g)} \cdot g; \qquad L = \text{LCM}(lm(f), lm(g))$$

Buchberger's Algorithm

INPUT : $F = \{f_1, \ldots, f_s\}$, and term order $>$

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 $G = G'$

$$S(f, g) = \frac{L}{lt(f)} \cdot f - \frac{L}{lt(g)} \cdot g$$

$L = \text{LCM}(lm(f), lm(g))$, $lm(f)$: leading monomial of f

Product Criteria

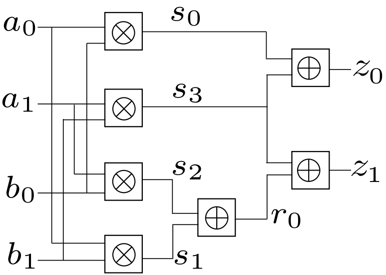
If $lm(f) \cdot lm(g) = LCM(lm(f), lm(g))$, then $S(f, g) \xrightarrow{G'}_+ 0$.

$lm(f) \cdot lm(g) = LCM(lm(f), lm(g))$, implies $lm(f), lm(g)$ are relatively prime

Our investigations...

Find a term order that makes ALL $\{lm(f),\ lm(g)\}$ relatively prime. Then: All $Spoly(f, g) \xrightarrow{G}_+ 0$ and we will already have a Gröbner basis!

For Circuits, such an order can be derived



$f_1 : s_0 + a_0 \cdot b_0;$ $f_2 : s_1 + a_0 \cdot b_1;$ $f_3 : s_2 + a_1 \cdot b_0;$

$f_4 : s_3 + a_1 \cdot b_1;$ $f_5 : r_0 + s_1 + s_2;$ $f_6 : z_0 + s_0 + s_3$

$f_7 : z_1 + r_0 + s_3;$ $f_8 : A + a_0 + a_1\alpha;$ $f_9 : B + b_0 + b_1\alpha$

$f_{10} : Z + z_0 + z_1\alpha;$

- Reverse Topological Traversal of the Circuit
- LEX with $Z > \{A > B\} > \{z_0 > z_1\} > \{r_0 > s_0 > s_3\} > \{s_1 > s_2\} > \{a_0 > a_1 > b_0 > b_1\}$

Using Our Topological Term Order:

- $F = \{f_1, \dots, f_s\}$ is a Gröbner Basis of $J = \langle f_1, \dots, f_s \rangle$
- $F_0 = \{x_1^q - x_1, \dots, x_n^q - x_n\}$ is also a Gröbner basis of J_0
- But we have to compute a Gröbner Basis of $J + J_0 = \langle f_1, f_2 \dots, f_s, \quad x_1^q - x_1, \dots, x_n^q - x_n \rangle$
- We show that $\{f_1, f_2 \dots, f_s, \quad x_1^q - x_1, \dots, x_n^q - x_n\}$ is a Gröbner basis!!
- From our circuit: $f_i = x_i + P$; Vanishing polynomials $x_i^q - x_i$
- Only pairs to consider: $S(f_i, \quad x_i^q - x_i)$ in Buchberger's Algorithm:

$$S(f_i = x_i + P, \quad x_i^q - x_i) = x_i^{q-1} P + x_i$$

$$x_i^{q-1} P + x_i \xrightarrow{x_i + P} x_i^{q-2} P^2 + x_i \xrightarrow{x_i + P} \dots \xrightarrow{x_i + P} P^q - P \xrightarrow{J_0}_+ 0$$

Our Overall Approach

- Given the circuit, perform reverse topological traversal
- Derive the term order to represent the polynomials for every gate
- The set: $\{F, F_0\} = \{f_1, \ldots, f_s, \quad x_1^q - x_1, \ldots, x_n^q - x_n\}$ is a Gröbner Basis
- Obtain: $f \xrightarrow{F,F_0}_+ r$
- If $r = 0$, the circuit is correct
- If $r \neq 0$, then r contains only the primary input variables
- Any SAT assignment to $r \neq 0$ generates a counter-example
- Counter-example found in no time as r is simplified by Gröbner basis reduction

Improve GB-reduction: F_4-style reduction

Our approach moves the "complexity' from $GB(J + J_0)$ to $f \xrightarrow{f_1,\ldots,f_s}_+ r$

Improve GB-reduction: F_4-style reduction

Our approach moves the "complexity' from $GB(J + J_0)$ to $f \xrightarrow{f_1,\dots,f_s}_+ r$

New algorithm to compute a Gröbner basis by J.C. Faugère: F_4

- Buchberger's algorithm $S(f, g) \xrightarrow{G}_+ r$
- Instead, compute a "set" of $S(f, g)$ in one-go
- Reduces them "simultaneously"
- Significant speed-up in computing a Gröbner basis
- Models the problem using sparse linear algebra
- Gaussian elimination on a matrix representation of the problem

Our term order: already a Gröbner basis. We only need F_4-style reduction: $f \xrightarrow{F, F_0}_+ r$

- Spec: $f : Z + A \cdot B$, compute $f \xrightarrow{f_1,\ldots,f_s}_+ r$
- Find a polynomial f_i that divides f, or "cancels" $LT(f)$
- $r = f - \frac{lt(f)}{lt(f_i)} \cdot f_i = f - \frac{lc(f)}{lc(f_i)} \cdot \frac{lm(f)}{lm(f_i)} \cdot f_i$
- Construct a matrix: rows = polynomials, columns = monomials, entries = coefficient of monomial present in the polynomial

$$
\begin{array}{c}
\\ f \\ f_3 \\ Bf_1 \\ a_0 f_2 \\ a_1 f_2 \\ f_5 \\ f_6 \\ f_4
\end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} Z & AB & Ba_0 & Ba_1 & z_0 & z_1 & r_0 & a_0 b_0 & a_0 b_1 & a_1 b_0 & a_1 b_1 \end{array} \\
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & \alpha \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{pmatrix}
\end{array}
$$

- Spec: $f : Z + A \cdot B$, compute $f \xrightarrow{f_1,\dots,f_s}_+ r$
- $f_3 : Z = z_0 + z_1\alpha$

$$
\begin{array}{c|c}
 & \begin{array}{ccccccccccc} Z & AB & Ba_0 & Ba_1 & z_0 & z_1 & r_0 & a_0b_0 & a_0b_1 & a_1b_0 & a_1b_1 \end{array} \\
\begin{array}{c} f \\ f_3 \\ Bf_1 \\ a_0f_2 \\ a_1f_2 \\ f_5 \\ f_6 \\ f_4 \end{array} &
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & \alpha \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{pmatrix}
\end{array}
$$

- To cancel the term AB
- $f_1 : A = a_0 + a_1\alpha$
- $Bf_1 : AB = Ba_0 + Ba_1\alpha$

$$
\begin{array}{c}
 \\ f \\ f_3 \\ Bf_1 \\ a_0 f_2 \\ a_1 f_2 \\ f_5 \\ f_6 \\ f_4
\end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} Z & AB & Ba_0 & Ba_1 & z_0 & z_1 & r_0 & a_0 b_0 & a_0 b_1 & a_1 b_0 & a_1 b_1 \end{array} \\
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & \alpha \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{pmatrix}
\end{array}
$$

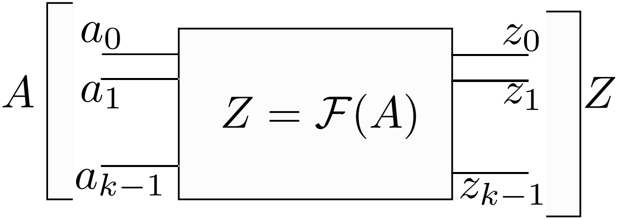
F_4-style reduction

- Construct the Matrix for polynomial reduction
- Apply Gaussian elimination on the matrix
- Last row = result of reduction = $\alpha^2 + \alpha + 1 = 0$

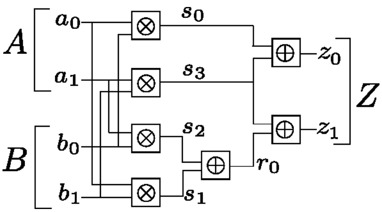
$$
\begin{array}{c}
\begin{array}{ccccccccccc}
Z & AB & Ba_0 & Ba_1 & z_0 & z_1 & r_0 & a_0b_0 & a_0b_1 & a_1b_0 & a_1b_1
\end{array}\\
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 1 & \alpha & 1 & \alpha & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & \alpha & 1 & \alpha & 0 & 1 & \alpha & 0 & 0\\
0 & 0 & 0 & 0 & 1 & \alpha & 0 & 1 & \alpha & \alpha & \alpha^2\\
0 & 0 & 0 & 0 & 0 & \alpha & 0 & 0 & \alpha & \alpha & \alpha^2+1\\
0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & \alpha & \alpha & \alpha^2+\alpha+1\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^2+\alpha+1
\end{pmatrix}
\end{array}
$$

See publication [Lv *et al*, TCAD 2013] [3] for more details

Problem 2: Polynomial Interpolation from Circuits



- Circuit: $f : \mathbb{B}^k \rightarrow \mathbb{B}^k$
- Model it as a polynomial function $f : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_{2^k}$
- Interpolate a word-level polynomial from the circuit: $Z = \mathcal{F}(A)$
- $A = a_0 + a_1\alpha + \dots a_{k-1}\alpha^{k-1}, \quad Z = z_0 + z_1\alpha + \dots z_{k-1}\alpha^{k-1}$
- Compute Gröbner basis of circuit polynomials with elimination (LEX) order: circuit-variables $> Z > A$
- Obtain $Z = \mathcal{F}(A)$ as a unique, canonical, polynomial representation from the circuit



$f_1 : z_0 + z_1\alpha + Z$; $f_2 : b_0 + b_1\alpha + B$; $f_3 : a_0 + a_1\alpha + A$; $f_4 : s_0 + a_0 \cdot b_0$; $f_5 : s_1 + a_0 \cdot b_1$; $f_6 : s_2 + a_1 \cdot b_0$; $f_7 : s_3 + a_1 \cdot b_1$; $f_8 : r_0 + s_1 + s_2$; $f_9 : z_0 + s_0 + s_3$; $f_{10} : z_1 + r_0 + s_3$. Ideal $J = \langle f_1, \ldots, f_{10} \rangle$.

Add J_0 and compute $GB(J + J_0)$ with $x_i > Z > A > B$, then G :

$g_1 : z_0 + z_1\alpha + Z$; $g_2 : b_0 + b_1\alpha + B$; $g_3 : a_0 + a_1\alpha + A$; $g_4 : s_3 + r_0 + z_1$; $g_5 : s_1 + s_2 + r_0$; $g_6 : s_0 + s_3 + z_0$; $g_7 : Z + AB$; $g_8 : a_1 b_1 + a_1 B + b_1 A + z_1$; $g_9 : r_0 + a_1 b_1 + z_1$; $g_{10} : s_2 + a_1 b_0$

A Proof Outline for this result

- Let unknown specification polynomial $f : Z + \mathcal{F}(A) \quad (Z = \mathcal{F}(A))$
- I have already shown that $f \in J + J_0$
- Let $G = \{g_1, \ldots, g_t\}$ be a reduced $GB(J + J_0)$ with LEX "circuit variables $> Z > A$
- Definition of GB: $\exists g_i$ such that $lm(g_i) \mid Z$
- So $g_i = Z + \mathcal{F}(A)$
- Play the same tricks with term-ordering and scale your verification
- For more details, see [4] [5].

For the algebraists....

In general, $\pi_l(V(J)) \subseteq V(J_l)$. However, over Galois fields $\mathbb{F}_q$, $\pi_l(V(J + J_0)) = V((J + J_0)_l)$.

Tool Development and Experimental Results

- Initial experiments with SINGULAR computer algebra tool [6]
- Developed a custom verification tool, written in C++
- GF library, ring operations $\mathbb{F}_q[x_1, \ldots, x_n]$, LEX order
- Euclidean algorithm, F_4-style reduction fine-tuned for circuits
- Solves verification & abstraction
- Tools and benchmarks can be obtained from:
  `http://www.ece.utah.edu/~pruss/abstract.html`

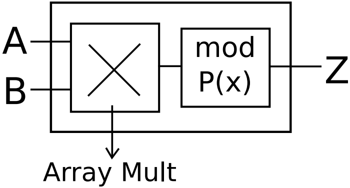


Figure: Mastrovito Multiplier

FLATTENED MASTROVITO MULTIPLIERS. TIME IS GIVEN IN SECONDS.MEMORY IS GIVEN IN MB. $TO = 3$ DAYS $(259,200$ SECONDS.)

Size (k) / # of Gates		163 / 153K	233 / 167K	283 / 399K	409 / 508K	571 / 1.6M
Time (s)	Bug Free	1,443	1,913	11,116	17,848	192,032
	Buggy	1,487	2,106	11,606	20,263	204,194
Max Memory (MB)		213	269	561	845	2,855

Composite Field Arithmetic Circuits: $\mathbb{F}_{2^k} \equiv \mathbb{F}_{(2^m)^n}$

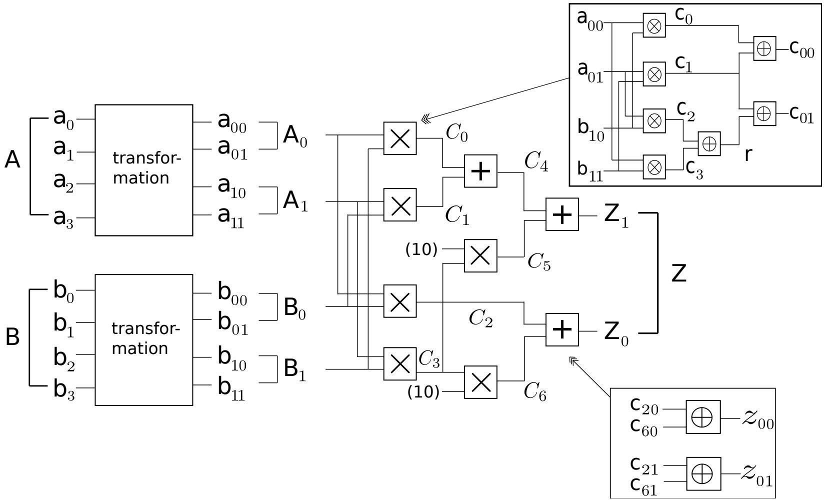


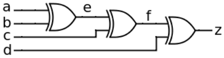
Figure: 4-bit composite multiplier designed over $\mathbb{F}_{(2^2)^2}$

Abstraction of Composite Field Multipliers

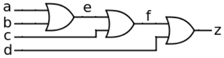
Abstraction of Mastrovito multipliers over $\mathbb{F}_{(2^m)^n}$.
Time is given in seconds. Memory is given in MB.

$k = 1024$				
m	n	Time		Max
		Bug Free	Buggy	Mem
2	512	11,883	12,050	414
4	256	1,520	1,536	106
8	128	209	211	29
16	64	38	37	10
32	32	10	10	5
64	16	4	4	3
128	8	2	2	3
256	4	1	1	3
512	2	1	1	3

See publication [5] for more details



(a) XOR logic



(b) OR logic

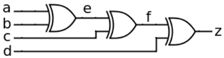
For XOR logic:

$$f_1 : z + f + d \quad f_2 : f + e + c \quad f_3 : e + b + a$$

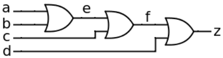
The reduction procedure $z \xrightarrow{f_1, f_2, f_3}_+ r$ will be computed as follows:

- $z \xrightarrow{z+f+d} f + d$
- $(f + d) \xrightarrow{f+e+c} e + d + c$
- $(e + d + c) \xrightarrow{e+b+a} d + c + b + a$

Limitations of the Abstraction Approach



(c) XOR logic



(d) OR logic

For OR logic:

$$f_1 : z + fd + f + d \quad f_2 : f + ec + e + c \quad f_3 : e + ba + b + a$$

The reduction procedure, $z \xrightarrow{f_1, f_2, f_3}_+ r$ is now computed as:

- $z \xrightarrow{z+fd+f+d} fd + f + d$
- $(fd + f + d) \xrightarrow{f+ec+e+c} f + edc + ed + dc + d$;
  $(f + edc + ed + dc + d) \xrightarrow{f+ec+e+c} edc + ed + ec + e + dc + d + c$
- $(edc + ed + ec + e + dc + d + c) \xrightarrow{e+ba+b+a}_+$
  $dcba + dcb + dca + dba + dc + db + da + d + cba + cb + ca + c + ba + b + a$

Use "implicit" representations: ZBDDs

Chain of OR-gates: ZDD size is $2n - 1$ instead of $2^n - 1$

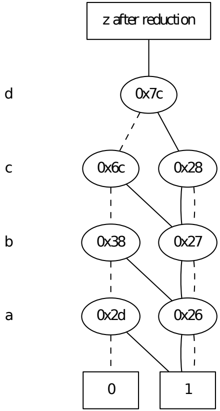


Figure: ZDD for remainder (mod chain of OR gates) for 4 variables

Further Work pursued by my research group

- Implement GB-reduction tool using GPU computing
- Formal verification of sequential Galois field circuits (see [7])
- Designed using optimal normal bases over $\mathbb{F}_{2^k}$
- Extensions of our work to Sequential Circuits
    - Reachability analysis of finite state machines at word-level
- New directions in Boolean Gröbner bases $\mathbb{Z}_2[x_1, \ldots, x_n]$ using implicit representation, such as Zero-suppressed BDDs
- Abstraction from $f : \mathbb{B}^k \to \mathbb{B}^k$ to $f : \mathbb{Z}_{2^k} \to \mathbb{Z}_{2^k}$
- Explore over-approximations of functions of the circuit through elimination ideals

Sequential Galois field circuits

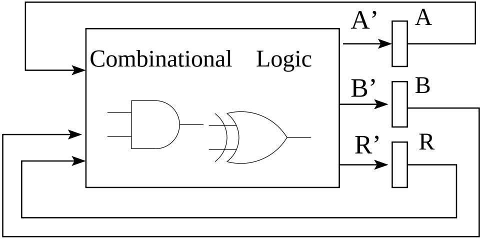


Figure: A typical normal basis GF sequential circuit model.

- $A = (a_0, \ldots, a_{k-1})$ and similarly B, R are k-bit registers;
- k-cycle execution of the FSM: $R = \mathcal{F}(A, B)$
- Project the variety $V(J + J_0)$ on the state-variables
- Word-Level Reachability Analysis of FSM over $\mathbb{F}_{2^k}$
- Efficient solutions for *quantifier elimination* over $\mathbb{F}_{2^k}$ [8]
- See our recent [DATE 2015] paper [7]

Datapath Verification over $\mathbb{Z}_{2^k}$

Consider the signal processing computation:

- $F = \frac{1}{2\sqrt{a^2+b^2}}$
- Let $x = a^2 + b^2 > 0$, then $F = \frac{1}{2\sqrt{x^2}}$

Datapath Verification over $\mathbb{Z}_{2^k}$

Consider the signal processing computation:

- $F = \frac{1}{2\sqrt{a^2+b^2}}$
- Let $x = a^2 + b^2 > 0$, then $F = \frac{1}{2\sqrt{x^2}}$

Approximate using Taylor's series, and implement with $X[15:0]$

Datapath Verification over $\mathbb{Z}_{2^k}$

Consider the signal processing computation:

- $F = \frac{1}{2\sqrt{a^2+b^2}}$
- Let $x = a^2 + b^2 > 0$, then $F = \frac{1}{2\sqrt{x^2}}$

Approximate using Taylor's series, and implement with $X[15:0]$

$$\begin{aligned} F[15:0] \quad = \quad & 156(X[15:0])^6 + 62724(X[15:0])^5 + 17968(X[15:0])^4 \\ & +18661(X[15:0])^3 + 43593(X[15:0])^2 \\ & +40224(X[15:0]) + 13281 \end{aligned}$$

Datapath Verification over $\mathbb{Z}_{2^k}$

Consider the signal processing computation:

- $F = \frac{1}{2\sqrt{a^2+b^2}}$
- Let $x = a^2 + b^2 > 0$, then $F = \frac{1}{2\sqrt{x^2}}$

Approximate using Taylor's series, and implement with $X[15:0]$

$$\begin{aligned} F[15:0] &= 156(X[15:0])^6 + 62724(X[15:0])^5 + 17968(X[15:0])^4 \\ &+18661(X[15:0])^3 + 43593(X[15:0])^2 \\ &+40224(X[15:0]) + 13281 \end{aligned}$$

$$\begin{aligned} G[15:0] &= 156(X[15:0])^6 + 5380(X[15:0])^5 + 1584(X[15:0])^4 \\ &+10469(X[15:0])^3 + 27209(X[15:0])^2 \\ &+7456(X[15:0]) + 13281 \end{aligned}$$

Datapath Verification over $\mathbb{Z}_{2^k}$

Consider the signal processing computation:

- $F = \frac{1}{2\sqrt{a^2+b^2}}$
- Let $x = a^2 + b^2 > 0$, then $F = \frac{1}{2\sqrt{x^2}}$

Approximate using Taylor's series, and implement with $X[15:0]$

$$\begin{aligned} F[15:0] &= 156(X[15:0])^6 + 62724(X[15:0])^5 + 17968(X[15:0])^4 \\ &+18661(X[15:0])^3 + 43593(X[15:0])^2 \\ &+40224(X[15:0]) + 13281 \end{aligned}$$

$$\begin{aligned} G[15:0] &= 156(X[15:0])^6 + 5380(X[15:0])^5 + 1584(X[15:0])^4 \\ &+10469(X[15:0])^3 + 27209(X[15:0])^2 \\ &+7456(X[15:0]) + 13281 \end{aligned}$$

$F \neq G$, but $F[15:0] = G[15:0]$ or $F = G \pmod{2^{16}}$

What's the big deal over $\mathbb{Z}_{2^k}$?

The finite integer ring $\mathbb{Z}_{2^k}$ is a non-unique factorization domain (non-UFD)

$$f = x^2 + 6x \pmod{2^3}$$

$$x(x+6) \qquad\qquad (x+4)(x+2)$$

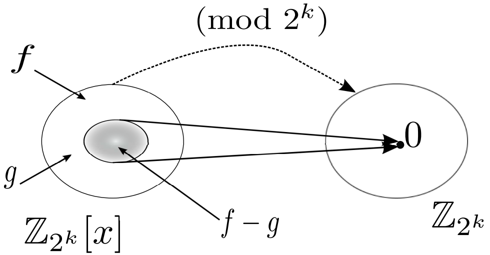
The presence of zero-divisors, lack of inverses, and $\cdots$

The ideal of vanishing polynomials (again!)

$$F = G \pmod{2^k} \iff F - G = 0 \pmod{2^k}$$

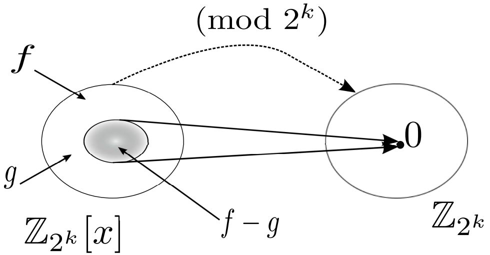
The ideal of vanishing polynomials (again!)

$$F = G \pmod{2^k} \iff F - G = 0 \pmod{2^k}$$



The ideal of vanishing polynomials (again!)

$$F = G \pmod{2^k} \iff F - G = 0 \pmod{2^k}$$



- Ideal of vanishing polynomials (J_0) in $\mathbb{Z}_{2^k}[x]$
- If the generators of J_0 are known in $\mathbb{Z}_{2^k}$, compute Gröbner basis
- How do we generate this ideal?

Number theory: Divisibility properties

- $f \pmod{2^k} = 0$ means that $2^k \mid f$

Number theory: Divisibility properties

- $f \pmod{2^k} = 0$ means that $2^k \mid f$
- $n!$ divides the product of any n consecutive integers
  - E.g. $4!$ divides $99 \times 100 \times 101 \times 102$

Number theory: Divisibility properties

- $f \pmod{2^k} = 0$ means that $2^k \mid f$
- $n!$ divides the product of any n consecutive integers
  - E.g. $4!$ divides $99 \times 100 \times 101 \times 102$
- Find the least integer λ s.t. $2^k \mid \lambda!$

Number theory: Divisibility properties

- $f \pmod{2^k} = 0$ means that $2^k \mid f$
- $n!$ divides the product of any n consecutive integers
  - E.g. $4!$ divides $99 \times 100 \times 101 \times 102$
- Find the least integer λ s.t. $2^k \mid \lambda!$
- Therefore, 2^k divides the product of any λ consecutive integers

Number theory: Divisibility properties

- $f \pmod{2^k} = 0$ means that $2^k \mid f$
- $n!$ divides the product of any n consecutive integers
  - E.g. $4!$ divides $99 \times 100 \times 101 \times 102$
- Find the least integer λ s.t. $2^k \mid \lambda!$
- Therefore, 2^k divides the product of any λ consecutive integers
- Example: In $\mathbb{Z}_{2^3}$, $\lambda = 4$, as $8 \mid 4!$

Number theory: Divisibility properties

- $f \pmod{2^k} = 0$ means that $2^k \mid f$
- $n!$ divides the product of any n consecutive integers
  - E.g. $4!$ divides $99 \times 100 \times 101 \times 102$
- Find the least integer λ s.t. $2^k \mid \lambda!$
- Therefore, 2^k divides the product of any λ consecutive integers
- Example: In $\mathbb{Z}_{2^3}$, $\lambda = 4$, as $8 \mid 4!$
  - Product of 4 consecutive integers vanishes in $\mathbb{Z}_8$

Number theory: Divisibility properties

- $f \pmod{2^k} = 0$ means that $2^k \mid f$
- $n!$ divides the product of any n consecutive integers
  - E.g. $4!$ divides $99 \times 100 \times 101 \times 102$
- Find the least integer λ s.t. $2^k \mid \lambda!$
- Therefore, 2^k divides the product of any λ consecutive integers
- Example: In $\mathbb{Z}_{2^3}$, $\lambda = 4$, as $8 \mid 4!$
  - Product of 4 consecutive integers vanishes in $\mathbb{Z}_8$
  - Factorize f as a product of 4 consecutive integers

Number theory: Divisibility properties

- $f \pmod{2^k} = 0$ means that $2^k \mid f$
- $n!$ divides the product of any n consecutive integers
    - E.g. $4!$ divides $99 \times 100 \times 101 \times 102$
- Find the least integer λ s.t. $2^k \mid \lambda!$
- Therefore, 2^k divides the product of any λ consecutive integers
- Example: In $\mathbb{Z}_{2^3}$, $\lambda = 4$, as $8 \mid 4!$
    - Product of 4 consecutive integers vanishes in $\mathbb{Z}_8$
    - Factorize f as a product of 4 consecutive integers
    - $(x+1)(x+2)(x+3)(x+4) = 0 \pmod 8$

Basis for factorization

In $\mathbb{Z}_{2^k}$, find least λ s.t. $2^k \mid \lambda!$

- $S_0(x) = 1$

Basis for factorization

In $\mathbb{Z}_{2^k}$, find least λ s.t. $2^k \mid \lambda!$

- $S_0(x) = 1$
- $S_1(x) = (x+1)$

Basis for factorization

In $\mathbb{Z}_{2^k}$, find least λ s.t. $2^k \mid \lambda!$

- $S_0(x) = 1$
- $S_1(x) = (x+1)$
- $S_2(x) = (x+1)(x+2)$: Product of 2 consecutive integers
- $\ldots$

Basis for factorization

In $\mathbb{Z}_{2^k}$, find least λ s.t. $2^k \mid \lambda!$

- $S_0(x) = 1$
- $S_1(x) = (x+1)$
- $S_2(x) = (x+1)(x+2)$: Product of 2 consecutive integers
- $\ldots$
- $S_\lambda(x) = (x+\lambda)S_{\lambda-1}(x)$: Product of λ consecutive integers

Basis for factorization

In $\mathbb{Z}_{2^k}$, find least λ s.t. $2^k \mid \lambda!$

- $S_0(x) = 1$
- $S_1(x) = (x+1)$
- $S_2(x) = (x+1)(x+2)$: Product of 2 consecutive integers
- $\ldots$
- $S_\lambda(x) = (x+\lambda)S_{\lambda-1}(x)$: Product of λ consecutive integers
- If $f = F_\lambda \cdot S_\lambda(x), \quad F_\lambda \in \mathbb{Z}_{2^k}[x], \quad$ then $f = 0 \pmod{2^k}$

Basis for factorization

- What is f cannot be factorized as $f = F_\lambda \cdot S_\lambda$?

Basis for factorization

- What is f cannot be factorized as $f = F_\lambda \cdot S_\lambda$?
- Example: In $\mathbb{Z}_{2^3}[x], \lambda = 4$

Basis for factorization

- What is f cannot be factorized as $f = F_\lambda \cdot S_\lambda$?
- Example: In $\mathbb{Z}_{2^3}[x], \lambda = 4$
- $f = 4x^2 + 4x = 4(x+1)(x+2) = 0 \pmod{2^3}$

Basis for factorization

- What is f cannot be factorized as $f = F_\lambda \cdot S_\lambda$?
- Example: In $\mathbb{Z}_{2^3}[x], \lambda = 4$
- $f = 4x^2 + 4x = 4(x+1)(x+2) = 0 \pmod{2^3}$
- The missing factors are compensated by the coefficient

Basis for factorization

- What is f cannot be factorized as $f = F_\lambda \cdot S_\lambda$?
- Example: In $\mathbb{Z}_{2^3}[x], \lambda = 4$
- $f = 4x^2 + 4x = 4(x+1)(x+2) = 0 \pmod{2^3}$
- The missing factors are compensated by the coefficient
- Deciding vanishing polynomials: $V(x) = 0 \pmod{2^k}$ iff
  - $\mathbf{V(x) = F_\lambda \cdot S_\lambda + \sum_{n=0}^{\lambda-1} a_n \cdot S_n(x)}$
  - $a_n =$ integer multiple of $\frac{2^k}{gcd(2^k, n!)}$

Basis for factorization

- What is f cannot be factorized as $f = F_\lambda \cdot S_\lambda$?
- Example: In $\mathbb{Z}_{2^3}[x], \lambda = 4$
- $f = 4x^2 + 4x = 4(x+1)(x+2) = 0 \pmod{2^3}$
- The missing factors are compensated by the coefficient
- Deciding vanishing polynomials: $V(x) = 0 \pmod{2^k}$ iff
  - $\mathbf{V(x) = F_\lambda \cdot S_\lambda + \sum_{n=0}^{\lambda-1} a_n \cdot S_n(x)}$
  - $a_n =$ integer multiple of $\frac{2^k}{gcd(2^k, n!)}$
- $V(x) =$ canonical representation of the vanishing ideal in $\mathbb{Z}_{2^k}[x]$

Basis for factorization

- What is f cannot be factorized as $f = F_\lambda \cdot S_\lambda$?
- Example: In $\mathbb{Z}_{2^3}[x], \lambda = 4$
- $f = 4x^2 + 4x = 4(x+1)(x+2) = 0 \pmod{2^3}$
- The missing factors are compensated by the coefficient
- Deciding vanishing polynomials: $V(x) = 0 \pmod{2^k}$ iff
  - $\mathbf{V(x) = F_\lambda \cdot S_\lambda + \sum_{n=0}^{\lambda-1} a_n \cdot S_n(x)}$
  - a_n = integer multiple of $\frac{2^k}{gcd(2^k, n!)}$
- $V(x)$ = canonical representation of the vanishing ideal in $\mathbb{Z}_{2^k}[x]$
- $V(x)$ constitutes a Gröbner basis

Basis for factorization

- What is f cannot be factorized as $f = F_\lambda \cdot S_\lambda$?
- Example: In $\mathbb{Z}_{2^3}[x], \lambda = 4$
- $f = 4x^2 + 4x = 4(x+1)(x+2) = 0 \pmod{2^3}$
- The missing factors are compensated by the coefficient
- Deciding vanishing polynomials: $V(x) = 0 \pmod{2^k}$ iff
  - $\mathbf{V(x) = F_\lambda \cdot S_\lambda + \sum_{n=0}^{\lambda-1} a_n \cdot S_n(x)}$
  - $a_n =$ integer multiple of $\frac{2^k}{gcd(2^k, n!)}$
- $V(x) =$ canonical representation of the vanishing ideal in $\mathbb{Z}_{2^k}[x]$
- $V(x)$ constitutes a Gröbner basis
- To prove $f = g \pmod{2^k}$, compute $(f - g) \pmod{\mathbf{V(x)}} = r$, is $r = 0$?

For the SMT community....

- Application to simulation and BV-constraint solving
- $\mathbf{V(x) = F_\lambda \cdot S_\lambda + \sum_{n=0}^{\lambda-1} a_n \cdot S_n(x) = 0} \pmod{\mathbf{2^k}}$
- Exhaustive simulation is not always necessary for polyfunction equivalence $\pmod{2^k}$
- $V(x)$ vanishes on any λ consecutive integers
- In $\mathbb{Z}_{2^k}$, λ is very small
  - For example, in $\mathbb{Z}_{2^{16}}$, $\lambda = 18$
  - Instead of a 16-bit solver, can you not design a 5-bit solver?
- Doesn't invalidate NP-hardness results of polynomial identity testing
- In $\mathbb{Z}_p[x], \lambda = p$, so exhaustive simulation is needed
- Related Publications: [9] [10]

Conclusions

- Formal Verification of large Galois Field circuits
- Computer algebra approach:
    - Nullstellensatz+Gröbner Bases methods
    - Engineering $\rightarrow$ a term order to obviate Gröbner basis computation
    - Can verify up to 571-bit multiplier circuits
    - NIST specified 571-bit field.... practical verification!
    - For Composite Field circuits, verification scales to 1024-bit fields
- Our approach relies on Gröbner basis theory, circuit analysis and efficient symbolic computation
- Also described polynomial RTL equivalence checking over finite integer rings
- Nature loves Gröbner basis!

Acknowledgments

- Former PhD students
  - Namrata Shekhar: Synopsys, Formality Equivalence Checker
  - Sivaram Gopalakrishnan: Synopsys, Formality Equivalence Checker
  - Jinpeng Lv: Cadence, Conformal Equivalence Checker
  - Tim Pruss: Apple, Formal Verification Engineer
- Current PhD students
  - Xiaojun Sun: Word-level implicit state enumeration for model checking sequential circuits
  - Utkarsh Gupta: Boolean Gröbner Bases
- Collaborator: Prof. Florian Enescu
  - Mathematics & Statistics, Georgia State Univ.
  - Commutative Algebra & Algebraic Geometry
- Research Funded by the U.S. National Science Foundation

Questions?

Thanks for listening!

Questions?

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

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

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.

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.

——, "Efficient Symbolic Computation for Word-Level Abstraction from Combinational Circuits for Verification over Galois Fields," *submitted, in review, IEEE Trans. on CAD*, 2015.

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.

- 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.
- S. Gao, A. Platzer, and E. Clarke, "Quantifier Elimination over Finite Fields with Gröbner Bases," in *Intl. Conf. Algebraic Informatics*, 2011.
- N. Shekhar, P. Kalla, M. B. Meredith, and F. Enescu, "Simulation Bounds for Equivalence Verification of Polynomial Datapaths using Finite Ring Algebra," *IEEE TransactionsVLSI*, vol. 16, no. 4, pp. 376–387, 2008.
- 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.