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 $(\text{mod } 2^k) = \mathbb{Z}_{2^k}$
    - $k$-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 (mod $2^k$) = $\mathbb{Z}_{2^k}$
    - $k$-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
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
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
- 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

- 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 $\mathbb{F}_{2^k} + \text{Gröbner Basis methods}$
  - 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, \( q = p^k, p = \text{prime} \)
- 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 \( \mathbb{F}_{2^k} \)
- \( \mathbb{F}_{2^k} \equiv \mathbb{F}_2[x] \mod P(x) \)
- \( P(x) \in \mathbb{F}_2[x] \), irreducible polynomial of degree \( k \)
- Operations performed \( \mod P(x) \) and coefficients reduced \( \mod 2 \)
Example Field Construction: $\mathbb{F}_8$

Construct: $\mathbb{F}_{2^3} = \mathbb{F}_2[x] \mod P(x) = x^3 + x + 1$

Consider any polynomial $A(x) \in \mathbb{F}_2[x]$

$A(x) \mod x^3 + x + 1 = a_2x^2 + a_1x + a_0$. Let $P(\alpha) = 0$:

- $\langle a_2, a_1, a_0 \rangle = \langle 0, 0, 0 \rangle = 0$
- $\langle a_2, a_1, a_0 \rangle = \langle 0, 0, 1 \rangle = 1$
- $\langle a_2, a_1, a_0 \rangle = \langle 0, 1, 0 \rangle = \alpha$
- $\langle a_2, a_1, a_0 \rangle = \langle 0, 1, 1 \rangle = \alpha + 1$
- $\langle a_2, a_1, a_0 \rangle = \langle 1, 0, 0 \rangle = \alpha^2$
- $\langle a_2, a_1, a_0 \rangle = \langle 1, 0, 1 \rangle = \alpha^2 + 1$
- $\langle a_2, a_1, a_0 \rangle = \langle 1, 1, 0 \rangle = \alpha^2 + \alpha$
- $\langle a_2, a_1, a_0 \rangle = \langle 1, 1, 1 \rangle = \alpha^2 + \alpha + 1$
Every function is a polynomial function over $\mathbb{F}_q$

Consider 1-bit right-shift operation $Z[2:0] = A[2:0] \gg 1$

<table>
<thead>
<tr>
<th>${a_2a_1a_0}$</th>
<th>$A$</th>
<th>$\rightarrow$</th>
<th>${z_2z_1z_0}$</th>
<th>$Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0</td>
<td>$\rightarrow$</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>$\rightarrow$</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>$\alpha$</td>
<td>$\rightarrow$</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>011</td>
<td>$\alpha + 1$</td>
<td>$\rightarrow$</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>$\alpha^2$</td>
<td>$\rightarrow$</td>
<td>010</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>101</td>
<td>$\alpha^2 + 1$</td>
<td>$\rightarrow$</td>
<td>010</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>110</td>
<td>$\alpha^2 + \alpha$</td>
<td>$\rightarrow$</td>
<td>011</td>
<td>$\alpha + 1$</td>
</tr>
<tr>
<td>111</td>
<td>$\alpha^2 + \alpha + 1$</td>
<td>$\rightarrow$</td>
<td>011</td>
<td>$\alpha + 1$</td>
</tr>
</tbody>
</table>
Polynomial Functions $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$

- Every function is a polynomial function over $\mathbb{F}_q$

\[
\begin{array}{cccc}
\{a_2a_1a_0\} & A & \rightarrow & \{z_2z_1z_0\} & Z \\
000 & 0 & \rightarrow & 000 & 0 \\
001 & 1 & \rightarrow & 000 & 0 \\
010 & \alpha & \rightarrow & 001 & 1 \\
011 & \alpha + 1 & \rightarrow & 001 & 1 \\
100 & \alpha^2 & \rightarrow & 010 & \alpha \\
101 & \alpha^2 + 1 & \rightarrow & 010 & \alpha \\
110 & \alpha^2 + \alpha & \rightarrow & 011 & \alpha + 1 \\
111 & \alpha^2 + \alpha + 1 & \rightarrow & 011 & \alpha + 1 \\
\end{array}
\]

$Z = (\alpha^2 + 1)A^4 + (\alpha^2 + 1)A^2$ 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: $\frac{y_2 - y_1}{x_2 - x_1}$

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{align*}
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) \\
Z_3 &= C^2 \\
E &= A \cdot C \\
X_3 &= A^2 + D + E \\
F &= X_3 + X_2 \cdot Z_3 \\
G &= X_3 + Y_2 \cdot Z_3 \\
Y_3 &= E \cdot F + Z_3 \cdot G
\end{align*}
Point addition using Projective Co-ordinates

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

\[
A = Y_2 \cdot Z_1^2 + Y_1 \\
B = X_2 \cdot Z_1 + X_1 \\
C = Z_1 \cdot B \\
D = B^2 \cdot (C + aZ_1^2) \\
Z_3 = C^2
\]

\[
E = A \cdot C \\
X_3 = A^2 + D + E \\
F = X_3 + X_2 \cdot Z_3 \\
G = X_3 + Y_2 \cdot Z_3 \\
Y_3 = E \cdot F + Z_3 \cdot G
\]

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

- $\mathbb{F}_q[x_1, \ldots, x_n]$: ring of all polynomials with coefficients in $\mathbb{F}_q$
- Polynomial $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$
  - Coefficients $c_i$, monomial $X = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, $\alpha_i \in \mathbb{Z}_{\geq 0}$
  - A monomial ordering is imposed on the ring, so $f : X_1 > X_2 > \cdots > X_t$
  - Leading term $lt(f) = c_1X_1$, $tail(f) = c_2X_2 + \cdots + c_tX_t$
  - 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

- \( \mathbb{F}_q[x_1, \ldots, x_n] \): ring of all polynomials with coefficients in \( \mathbb{F}_q \)
- Polynomial \( f = c_1X_1 + c_2X_2 + \cdots + c_tX_t \)
  - Coefficients \( c_i \), monomial \( X = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n} \), \( \alpha_i \in \mathbb{Z}_{\geq 0} \)
  - A monomial ordering is imposed on the ring, so \( f : X_1 > X_2 > \cdots > X_t \)
  - Leading term \( \text{lt}(f) = c_1X_1 \), \( \text{tail}(f) = c_2X_2 + \cdots + c_tX_t \)
  - Leading coefficient \( \text{lt}(f) = c_1 \) and leading monomial \( \text{lm}(f) = X_1 \)
- Example: \( f = 2x^2yz + 3xy^3 - 2x^3 \)
  - LEX with \( x > y > z \) : \( f = -2x^3 + 2x^2yz + 3xy^3 \)
  - DEGLEX with \( x > y > z \) : \( f = 2x^2yz + 3xy^3 - 2x^3 \)
  - DEGREVLEX with \( x > y > z \) : \( f = 3xy^3 + 2x^2yz - 2x^3 \)
Let $\mathbb{F}_q = GF(2^k)$, and $\overline{\mathbb{F}}_q$ be its closure

- $\mathbb{F}_q[x_1, \ldots, x_n]$: ring of all polynomials with coefficients in $\mathbb{F}_q$
- Polynomial $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$
  - Coefficients $c_i$, monomial $X = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdots x_n^{\alpha_n}$, $\alpha_i \in \mathbb{Z}_{\geq 0}$
  - A monomial ordering is imposed on the ring, so $f : X_1 > X_2 > \cdots > X_t$
  - Leading term $lt(f) = c_1X_1$, $tail(f) = c_2X_2 + \cdots + c_tX_t$
  - Leading coefficient $lt(f) = c_1$ and leading monomial $lm(f) = X_1$

**Example:** $f = 2x^2yz + 3xy^3 - 2x^3$

- LEX with $x > y > z$ : $f = -2x^3 + 2x^2yz + 3xy^3$
- DEGLEX with $x > y > z$ : $f = 2x^2yz + 3xy^3 - 2x^3$
- DEGREVLEX with $x > y > z$ : $f = 3xy^3 + 2x^2yz - 2x^3$

Leading terms $lt(f)$ play an important role
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{align*}
2x^2 + 3x + 1 \) & \quad x^3 - 2x^2 + 2x + 8 \\
& \quad - (x^3 - \frac{3}{2}x^2 - \frac{1}{2}x) \\
& \quad - \frac{7}{2}x^2 + \frac{3}{2}x + 8 \\
& \quad - \frac{7}{2}x^2 + \frac{21}{4}x + \frac{7}{4} \\
& \quad \frac{27}{4}x + \frac{39}{4}
\end{align*}
\]
Polynomial Division as Cancellation of Terms

Divide \( f = x^3 - 2x^2 + 2x + 8 \) by \( g = 2x^2 + 3x + 1 \)

2x^2 + 3x + 1) x^3 - 2x^2 + 2x + 8
\[ \underline{- x^3 - \frac{3}{2}x^2 - \frac{1}{2}x} \]
\[ - \frac{7}{2}x^2 + \frac{3}{2}x + 8 \]
\[ \underline{- \frac{7}{2}x^2 + \frac{21}{4}x + \frac{7}{4}} \]
\[ \frac{27}{4}x + \frac{39}{4} \]

- 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 \( F = \{f_1, \ldots, f_s\} \)
- In verification, we need solutions to the system of equations:

\[
\begin{align*}
  f_1 &= 0 \\
  f_2 &= 0 \\
  &\vdots \\
  f_s &= 0
\end{align*}
\]

- **Variety**: Set of ALL solutions to a given system of polynomial equations: \( V(f_1, \ldots, f_s) \)
- Variety depends on the ideal generated by the polynomials
- Reason about the Variety by analyzing the Ideals
Ideals in Rings

**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 $f_1(a) = f_2(a) = \cdots = f_s(a) = 0$; if $f \in \langle f_1, \ldots, f_s \rangle$ then $f(a) = 0$.
Different generators can generate the same ideal
\[ \langle f_1, \cdots, f_s \rangle = \cdots = \langle h_1, \ldots, h_r \rangle = \cdots = \langle g_1, \cdots, g_t \rangle, \text{ such that} \]
\[ V(f_1, \ldots, f_s) = V(h_1, \ldots, h_r) = V(g_1, \ldots, g_t) \]

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
  \[ \langle f_1, \ldots, f_s \rangle = \cdots = \langle h_1, \ldots, h_r \rangle = \cdots = \langle g_1, \ldots, g_t \rangle, \]  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)**

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

**Definition (Gröbner Basis for Ideal Membership Test)**

\[
G = GB(J) \iff \forall f \in J, f \xrightarrow{g_1, g_2, \ldots, g_t} 0
\]

Implies a “decision procedure” for ideal membership
Buchberger’s Algorithm Computes a Gröbner Basis

**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 $f$
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 \( I(V(J)) \)

\[
I(V(J))
\]

\[
\cdot f = x + y
\]

\[
J = \langle x^2, y^2 \rangle \\
V(J) = (0, 0)
\]

- 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, \( f \in I(V(J)) \).
**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:

\[
I(V(J)) = \{ f \in R : f(a) = 0 \ \forall a \in V(J) \}
\]

- 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:** \( I(V_{\mathbb{F}_q}(J)) = I(V_{\mathbb{F}_q}(J + J_0)) = \sqrt{J + J_0} = J + J_0 \)
Verification Formulation: The Mathematical Problem

- 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 \( f \xrightarrow{\text{GB}(J+J_0)} + 0 \)?
Example Verification Formulation

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

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

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

Verification problem: Check if $f \xrightarrow{\text{GB}(J+J_0)} + 0$?
In general, Complexity of Gröbner basis: doubly-exponential
  
  Degree of polynomials in $G$ is bounded by $2(\frac{1}{2}d^2 + d)^{2n-1}$ [2]
  
  However, for zero dimensional ideals: single-exponential complexity
  
  For $J \subset \mathbb{F}_q[x_1, \ldots, x_n]$, 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 $x > y > z$: $f = -2x^3 + 2x^2yz + 3xy^3$
- DEGLEX $x > y > z$: $f = 2x^2yz + 3xy^3 - 2x^3$
- DEGREVLEX $x > y > z$: $f = 3xy^3 + 2x^2yz - 2x^3$

Recall, S-polynomial depends on term ordering:

$$S(f, g) = \frac{L}{lt(f)} \cdot f - \frac{L}{lt(g)} \cdot g; \quad L = LCM(lm(f), lm(g))$$
Buchberger’s Algorithm Computes a Gröbner Basis

**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  
  $S(f, g) \rightarrow^G r$  
  IF $r \neq 0$ THEN $G := G \cup \{r\}$  
UNTIL $G = G'$

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

\[
S(f, g) = \frac{L}{lt(f)} \cdot f - \frac{L}{lt(g)} \cdot g
\]
Effect of Term Orderings on Buchberger’s Algorithm

Product Criteria

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

$l\text{m}(f) \cdot \text{lm}(g) = \text{LCM}(\text{lm}(f), \text{lm}(g))$, implies $\text{lm}(f), \text{lm}(g)$ are relatively prime

Our investigations...

Find a term order that makes ALL $\{\text{lm}(f), \text{lm}(g)\}$ 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

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

- Reverse Topological Traversal of the Circuit
- LEX with \( Z > \{ A > B \} > \{ z_0 > z_1 \} > \{ r_0 > s_0 > s_3 \} > \{ s_1 > s_2 \} > \{ a_0 > a_1 > b_0 > b_1 \} \)
Our Discovery: Gröbner Basis of $J + J_0$

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

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

$$x_i^{q-1}P + x_i \xrightarrow{x_i + P} x_i^{q-2}P^2 + x_i \xrightarrow{x_i + P} \ldots \xrightarrow{x_i + P} P^q - P \xrightarrow{J_0} + 0$$
Our Overall Approach

- Given the circuit, perform reverse topological traversal
- Derive the term order to represent the polynomials for every gate
- The set: \( \{ F, F_0 \} = \{ f_1, \ldots, f_s, \ x_1^q - x_1, \ldots, x_n^q - x_n \} \) 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
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 $f \overset{f_1,\ldots,f_s}{\longrightarrow} + r$

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

- Buchberger’s algorithm $S(f, g) \overset{G}{\longrightarrow} + r$
- 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: $f \overset{F,F_0}{\longrightarrow} + r$
F₄-style reduction

- **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

\[
\begin{pmatrix}
Z & AB & Ba_0 & Ba_1 & z_0 & z_1 & r_0 & a_0b_0 & a_0b_1 & a_1b_0 & a_1b_1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
f_3 & 1 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 \\
Bf_1 & 0 & 1 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 \\
a_0f_2 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 \\
a_1f_2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \alpha \\
f_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
f_6 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
f_4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0
\end{pmatrix}
\]
$F_4$-style reduction

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

\[
\begin{array}{cccccccccccc}
  & Z & AB & Ba_0 & Ba_1 & z_0 & z_1 & r_0 & a_0b_0 & a_0b_1 & a_1b_0 & a_1b_1 \\
  f & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  f_3 & 1 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 \\
  Bf_1 & 0 & 1 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  a_0f_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \alpha & 0 & 0 \\
  a_1f_2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & \alpha \\
  f_5 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
  f_6 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
  f_4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
\end{array}
\]
To cancel the term $AB$

- $f_1: A = a_0 + a_1\alpha$
- $Bf_1: AB = Ba_0 + Ba_1\alpha$

<table>
<thead>
<tr>
<th></th>
<th>$Z$</th>
<th>$AB$</th>
<th>$Ba_0$</th>
<th>$Ba_1$</th>
<th>$z_0$</th>
<th>$z_1$</th>
<th>$r_0$</th>
<th>$a_0b_0$</th>
<th>$a_0b_1$</th>
<th>$a_1b_0$</th>
<th>$a_1b_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$f_3$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\alpha$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$Bf_1$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$\alpha$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a_0f_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\alpha$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a_1f_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\alpha$</td>
<td>0</td>
</tr>
<tr>
<td>$f_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$f_6$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$f_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
\[ F_4\text{-style reduction} \]

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

\[
\begin{pmatrix}
Z & AB & Ba_0 & Ba_1 & z_0 & z_1 & r_0 & a_0 b_0 & a_0 b_1 & a_1 b_0 & a_1 b_1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & \alpha & 1 & \alpha & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha & 1 & \alpha & 0 & 1 & \alpha & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \alpha & 0 & 1 & \alpha & \alpha & \alpha \\
0 & 0 & 0 & 0 & 0 & \alpha & 0 & 0 & \alpha & \alpha & \alpha^2 \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & 0 & \alpha & \alpha^2 + 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & \alpha & \alpha^2 + \alpha + 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha & \alpha^2 + \alpha + 1 \\
\end{pmatrix}
\]

See publication [Lv et al, TCAD 2013] [3] for more details
Problem 2: Polynomial Interpolation from Circuits

- Circuit: \( f : \mathbb{B}^k \to \mathbb{B}^k \)
- Model it as a polynomial function \( f : \mathbb{F}_{2^k} \to \mathbb{F}_{2^k} \)
- Interpolate a word-level polynomial from the circuit: \( Z = \mathcal{F}(A) \)
- \( A = a_0 + a_1 \alpha + \ldots a_{k-1} \alpha^{k-1}, \quad Z = z_0 + z_1 \alpha + \ldots z_{k-1} \alpha^{k-1} \)
- 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
\[ \begin{align*}
    f_1 &: z_0 + z_1\alpha + Z; \\
    f_2 &: b_0 + b_1\alpha + B; \\
    f_3 &: a_0 + a_1\alpha + A; \\
    f_4 &: s_0 + a_0 \cdot b_0; \\
    f_5 &: s_1 + a_0 \cdot b_1; \\
    f_6 &: s_2 + a_1 \cdot b_0; \\
    f_7 &: s_3 + a_1 \cdot b_1; \\
    f_8 &: r_0 + s_1 + s_2; \\
    f_9 &: z_0 + s_0 + s_3; \\
    f_{10} &: z_1 + r_0 + s_3. \\
\end{align*} \]

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 \( G : \)

\[ \begin{align*}
    g_1 &: z_0 + z_1\alpha + Z; \\
    g_2 &: b_0 + b_1\alpha + B; \\
    g_3 &: a_0 + a_1\alpha + A; \\
    g_4 &: s_3 + r_0 + z_1; \\
    g_5 &: s_1 + s_2 + r_0; \\
    g_6 &: s_0 + s_3 + z_0; \\
    g_7 &: Z + AB; \\
    g_8 &: a_1 b_1 + a_1 B + b_1 A + z_1; \\
    g_9 &: r_0 + a_1 b_1 + z_1; \\
    g_{10} &: s_2 + a_1 b_0
\end{align*} \]
A Proof Outline for this result

- Let unknown specification polynomial $f : Z + \mathcal{F}(A)$ ($Z = \mathcal{F}(A)$)
- 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 $lm(g_i) | Z$
- 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, $\pi_I(V(J)) \subseteq V(J_I)$. However, over Galois fields $\mathbb{F}_q$, $\pi_I(V(J + J_0)) = V((J + J_0)_I)$. 
Initial experiments with \textsc{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
Flattened Mastrovito multipliers. Time is given in seconds. Memory is given in MB. \( TO = 3 \text{ days} \ (259,200 \text{ seconds}) \)

<table>
<thead>
<tr>
<th>Size (k)</th>
<th>163</th>
<th>233</th>
<th>283</th>
<th>409</th>
<th>571</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Gates</td>
<td>153K</td>
<td>167K</td>
<td>399K</td>
<td>508K</td>
<td>1.6M</td>
</tr>
<tr>
<td>Time (s)</td>
<td>Bug Free</td>
<td>1,443</td>
<td>1,913</td>
<td>11,116</td>
<td>17,848</td>
</tr>
<tr>
<td></td>
<td>Buggy</td>
<td>1,487</td>
<td>2,106</td>
<td>11,606</td>
<td>20,263</td>
</tr>
<tr>
<td>Max Memory (MB)</td>
<td>213</td>
<td>269</td>
<td>561</td>
<td>845</td>
<td>2,855</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$n$</th>
<th>Time</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Bug Free</td>
<td>Buggy Mem</td>
</tr>
<tr>
<td>2</td>
<td>512</td>
<td>11,883</td>
<td>12,050 414</td>
</tr>
<tr>
<td>4</td>
<td>256</td>
<td>1,520</td>
<td>1,536 106</td>
</tr>
<tr>
<td>8</td>
<td>128</td>
<td>209</td>
<td>211    29</td>
</tr>
<tr>
<td>16</td>
<td>64</td>
<td>38</td>
<td>37     10</td>
</tr>
<tr>
<td>32</td>
<td>32</td>
<td>10</td>
<td>10     5</td>
</tr>
<tr>
<td>64</td>
<td>16</td>
<td>4</td>
<td>4      3</td>
</tr>
<tr>
<td>128</td>
<td>8</td>
<td>2</td>
<td>2      3</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td>1</td>
<td>1      3</td>
</tr>
<tr>
<td>512</td>
<td>2</td>
<td>1</td>
<td>1      3</td>
</tr>
</tbody>
</table>

Limitations of the Abstraction Approach

(a) XOR 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:

1. \( z \xrightarrow{z+f+d} f + d \)
2. \( (f + d) \xrightarrow{f+e+c} e + d + c \)
3. \( (e + d + c) \xrightarrow{e+b+a} d + c + b + a \)
Limitations of the Abstraction Approach

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 $f : \mathbb{B}^k \rightarrow \mathbb{B}^k$ to $f : \mathbb{Z}_{2^k} \rightarrow \mathbb{Z}_{2^k}$
- Explore over-approximations of functions of the circuit through elimination ideals
Sequential Galois field circuits

![Diagram of a typical normal basis GF sequential circuit model.](image)

**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: $R = F(A, B)$
- 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]
Consider the signal processing computation:

- \( F = \frac{1}{2\sqrt{a^2+b^2}} \)
- Let \( x = a^2 + b^2 > 0 \), then \( F = \frac{1}{2\sqrt{x^2}} \)
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 \( X[15:0] \)
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 \( X[15 : 0] \)

\[
F[15 : 0] = 156(X[15 : 0])^6 + 62724(X[15 : 0])^5 + 17968(X[15 : 0])^4 \\
+ 18661(X[15 : 0])^3 + 43593(X[15 : 0])^2 \\
+ 40224(X[15 : 0]) + 13281
\]
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 \( X[15 : 0] \)

\[
F[15 : 0] = 156(X[15 : 0])^6 + 62724(X[15 : 0])^5 + 17968(X[15 : 0])^4 \\
+ 18661(X[15 : 0])^3 + 43593(X[15 : 0])^2 \\
+ 40224(X[15 : 0]) + 13281
\]

\[
G[15 : 0] = 156(X[15 : 0])^6 + 5380(X[15 : 0])^5 + 1584(X[15 : 0])^4 \\
+ 10469(X[15 : 0])^3 + 27209(X[15 : 0])^2 \\
+ 7456(X[15 : 0]) + 13281
\]
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 $F = \frac{1}{2\sqrt{x}}$

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

\[
F[15 : 0] = 156(X[15 : 0])^6 + 62724(X[15 : 0])^5 + 17968(X[15 : 0])^4 \\
+ 18661(X[15 : 0])^3 + 43593(X[15 : 0])^2 \\
+ 40224(X[15 : 0]) + 13281
\]

\[
G[15 : 0] = 156(X[15 : 0])^6 + 5380(X[15 : 0])^5 + 1584(X[15 : 0])^4 \\
+ 10469(X[15 : 0])^3 + 27209(X[15 : 0])^2 \\
+ 7456(X[15 : 0]) + 13281
\]

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

\[
f = x^2 + 6x \quad (\text{mod } 2^3)
\]

\[
x(x + 6) \quad (x + 4)(x + 2)
\]

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 \( 2^k | f \)
- \( n! \) 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 \( \lambda \) 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 \( \lambda \) s.t. \( 2^k \mid \lambda! \)
- Therefore, \( 2^k \) divides the product of any \( \lambda \) 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 $\lambda$ s.t. $2^k \mid \lambda!$
- Therefore, $2^k$ divides the product of any $\lambda$ consecutive integers
- Example: In $\mathbb{Z}_{23}$, $\lambda = 4$, 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 $\lambda$ s.t. $2^k \mid \lambda!$
- Therefore, $2^k$ divides the product of any $\lambda$ consecutive integers
- Example: In $\mathbb{Z}_{23}$, $\lambda = 4$, 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 \( \lambda \) s.t. \( 2^k \mid \lambda! \)
- Therefore, \( 2^k \) divides the product of any \( \lambda \) consecutive integers
- Example: In \( \mathbb{Z}_{23} \), \( \lambda = 4 \), 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 $\lambda$ s.t. $2^k \mid \lambda!$
- Therefore, $2^k$ divides the product of any $\lambda$ consecutive integers
- Example: In $\mathbb{Z}_{23}$, $\lambda = 4$, 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 $\lambda$ s.t. $2^k | \lambda!$

- $S_0(x) = 1$
In $\mathbb{Z}_{2^k}$, find least $\lambda$ s.t. $2^k \mid \lambda!$

- $S_0(x) = 1$
- $S_1(x) = (x + 1)$
In $\mathbb{Z}_{2^k}$, find least $\lambda$ s.t. $2^k | \lambda!$

- $S_0(x) = 1$
- $S_1(x) = (x + 1)$
- $S_2(x) = (x + 1)(x + 2)$: Product of 2 consecutive integers
- $\ldots$
In $\mathbb{Z}_{2^k}$, find least $\lambda$ 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 $\lambda$ consecutive integers
In $\mathbb{Z}_{2^k}$, find least $\lambda$ 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 $\lambda$ consecutive integers
- If $f = F_{\lambda} \cdot S_{\lambda}(x)$, $F_{\lambda} \in \mathbb{Z}_{2^k}[x]$, then $f = 0 \pmod{2^k}$
What is $f$ cannot be factorized as $f = F_\lambda \cdot S_\lambda$?
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
  - $V(x) = F_\lambda \cdot S_\lambda + \sum_{n=0}^{\lambda-1} a_n \cdot S_n(x)$
  - $a_n = \text{integer multiple of } \frac{2^k}{\gcd(2^k, n!)}$
Basis for factorization

- What is \( f \) cannot be factorized as \( f = F_\lambda \cdot S_\lambda \)?
- Example: In \( \mathbb{Z}_2^3[x] \), \( \lambda = 4 \)
  \[ f = 4x^2 + 4x = 4(x + 1)(x + 2) = 0 \mod 2^3 \]
- The missing factors are compensated by the coefficient
- Deciding vanishing polynomials: \( V(x) = 0 \mod 2^k \) iff
  \[ V(x) = F_\lambda \cdot S_\lambda + \sum_{n=0}^{\lambda-1} a_n \cdot S_n(x) \]
  \[ a_n = \text{integer multiple of } \frac{2^k}{\gcd(2^k,n!)} \]
  \[ V(x) = \text{canonical representation of the vanishing ideal in } \mathbb{Z}_{2^k}[x] \]
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

  $V(x) = F_\lambda \cdot S_\lambda + \sum_{n=0}^{\lambda-1} a_n \cdot S_n(x)$

  $a_n = \text{integer multiple of } \frac{2^k}{\text{gcd}(2^k, n!)}$

  $V(x) = \text{canonical representation of the vanishing ideal in } \mathbb{Z}_{2^k}[x]$

  $V(x)$ 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

  \[
  V(x) = F_\lambda \cdot S_\lambda + \sum_{n=0}^{\lambda-1} a_n \cdot S_n(x)
  \]

  \( a_n = \text{integer multiple of} \ \frac{2^k}{\gcd(2^k, n!)} \)

- \( V(x) = \text{canonical representation of the vanishing ideal in} \ \mathbb{Z}_{2^k}[x] \)
- \( V(x) = \text{constitutes a Gröbner basis} \)
- To prove \( f = g \pmod{2^k} \), compute \( (f - g) \pmod{V(x)} = r \), is \( r = 0 \)?
Application to simulation and BV-constraint solving

\[ V(x) = F_\lambda \cdot S_\lambda + \sum_{n=0}^{\lambda-1} a_n \cdot S_n(x) = 0 \pmod{2^k} \]

Exhaustive simulation is not always necessary for polyfunction equivalence \((\pmod{2^k})\)

\(V(x)\) vanishes on any \(\lambda\) consecutive integers

In \(\mathbb{Z}_{2k}\), \(\lambda\) 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 → 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?


