Formal Verification of Galois Field Arithmetic Circuits using Computer Algebra Techniques

Priyank Kalla

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

Some Background...

- My interests – Design Automation and Verification
  - Formal Verification of RTL-descriptions
  - Word-level abstractions from designs, symbolic techniques
  - Verification of finite-precision arithmetic
- Equivalence check: specification (*Spec*) vs implementation (*Impl*)
  - RTL-1 & RTL-2: same function?
  - Word-level spec (polynomial, RTL) vs gate-level circuit: same function?
  - RTL: functions over *k*-bit-vectors
    - 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 $\mathbb{Z}_{2^k}$ or $\mathbb{F}_{2^k}$
  - Devise decision procedures for polynomial function equivalence
  - Commutative algebra, algebraic geometry + contemporary verification
- This talk, mostly about verification over Galois fields

My Collaborators

- Former PhD students
  - Namrata Shekhar: Synopsys, Formality Equivalence Checker
  - Sivaram Gopalakrishnan: Synopsys, Formality Equivalence Checker
  - Jinpeng Lv: Cadence, Conformal Equivalence Checker
- Collaborator: Prof. Florian Enescu
  - Mathematics & Statistics, Georgia State Univ.
  - Commutative Algebra & Algebraic Geometry
  - NOT a computer-algebra specialist, but thats good!
    - Think about problems “theoretically”, algorithms can come later...

Agenda: Verification of Galois field circuits

- Motivation
  - Galois fields, hardware applications & their verification
- Target problems
  - Given Galois field $\mathbb{F}_{2^k}$, polynomial f, and circuit C
  - Verify: circuit C implements f; or find the bug
  - Given circuits C_1, C_2, is $C_1 \equiv C_2$ over $\mathbb{F}_{2^k}$?
  - 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
- Approach: **Computer Algebra Techniques**
  - Nullstellensatz + Gröbner Basis methods
  - Challenge: Complexity of Gröbner Basis algorithm
  - Contribution: A **term order** to **obviate** the Gröbner Basis algorithm for verification + custom F_4-style reduction
- Results & Conclusions

Motivation

- Wide applications of Galois field circuits
  - **Cryptography**: RSA, Elliptic Curve Cryptography (ECC)
  - Error Correcting Codes, Digital Signal Processing, etc.
- Bugs in hardware can leak secret keys [*Biham et al.*, "Bug Attacks", Crypto 2008]
- Data-path size in ECC crypto-systems can be very large
  - In $\mathbb{F}_{2^k}$, $k = 163, 233, \ldots$ (NIST standard)
  - ECC-point addition for encryption, decryption, authentication
  - Custom arithmetic architectures – hard to verify
  - Synthesized circuits are "easier" to verify
- Why use computer algebra?
  - Algebraic nature (finite field) of the computation (polynomial)
  - Verification infeasible with contemporary verification tools

Galois Field Overview

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

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

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

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

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

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

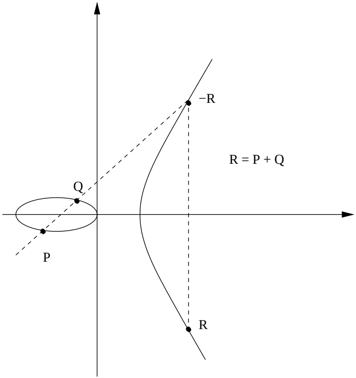
Field Elements: *e.g.* $\mathbb{F}_8$

Consider: $\mathbb{F}_{2^3} = \mathbb{F}_2[x] \pmod{x^3 + x + 1}$
$A \in \mathbb{F}_2[x]$
$A \pmod{x^3 + x + 1} = a_2x^2 + a_1x + a_0$. Let $P(\alpha) = 0$:

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

Elliptic Curves – Point addition

$$y^2 + xy = x^3 + ax^2 + b \text{ over } \mathrm{GF}(2^k)$$



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 $(X_3,\ Y_3,\ Z_3) = (X_1,\ Y_1,\ Z_1) + (X_2,\ Y_2,\ 1)$

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

No inverses, just addition and multiplication

Multiplication in GF(2^4)

Input:
$A = (a_3 a_2 a_1 a_0)$
$B = (b_3 b_2 b_1 b_0)$
$A = a_0 + a_1 \cdot \alpha + a_2 \cdot \alpha^2 + a_3 \cdot \alpha^3$
$B = b_0 + b_1 \cdot \alpha + b_2 \cdot \alpha^2 + b_3 \cdot \alpha^3$

Irreducible Polynomial:
$P = (11001)$
$P(x) = x^4 + x^3 + 1, \quad P(\alpha) = 0$

Result:
$A \times B \pmod{P(x)}$

Multiplication over GF(2^4)

$\times$			a_3 b_3	a_2 b_2	a_1 b_1	a_0 b_0
			$a_3 \cdot b_0$	$a_2 \cdot b_0$	$a_1 \cdot b_0$	$a_0 \cdot b_0$
		$a_3 \cdot b_1$	$a_2 \cdot b_1$	$a_1 \cdot b_1$	$a_0 \cdot b_1$	
	$a_3 \cdot b_2$	$a_2 \cdot b_2$	$a_1 \cdot b_2$	$a_0 \cdot b_2$		
$a_3 \cdot b_3$	$a_2 \cdot b_3$	$a_1 \cdot b_3$	$a_0 \cdot b_3$			
s_6	s_5	s_4	s_3	s_2	s_1	s_0

In polynomial expression:
$S = s_0 + s_1 \cdot \alpha + s_2 \cdot \alpha^2 + s_3 \cdot \alpha^3 + s_4 \cdot \alpha^4 + s_5 \cdot \alpha^5 + s_6 \cdot \alpha^6$

S should be further reduced $\pmod{P(x)}$

Multiplication over GF(2^4)

$$
\begin{array}{ccc|cccc|cl}
s_6 & s_5 & s_4 & s_3 & s_2 & s_1 & s_0 & & \\
\hline
 & & & s_4 & 0 & 0 & s_4 & \Leftarrow & s_4 \cdot \alpha^4 \pmod{P(\alpha)} \\
 & & & s_5 & 0 & s_5 & s_5 & \Leftarrow & s_5 \cdot \alpha^5 \pmod{P(\alpha)} \\
 & & + & s_6 & s_6 & s_6 & s_6 & \Leftarrow & s_6 \cdot \alpha^6 \pmod{P(\alpha)} \\
\hline
 & & & g_3 & g_2 & g_1 & g_0 & &
\end{array}
$$

$$s_4 \cdot \alpha^4 \pmod{\alpha^3 + \alpha + 1} = s_4 \cdot \alpha^3 + s_4$$
$$s_5 \cdot \alpha^5 \pmod{\alpha^3 + \alpha + 1} = s_5 \cdot \alpha^3 + s_5 \cdot \alpha + s_5$$
$$s_6 \cdot \alpha^6 \pmod{\alpha^3 + \alpha + 1} = s_6 \cdot \alpha^3 + s_6 \cdot \alpha^2 + s_6 \cdot \alpha + s_6$$

$$G = g_0 + g_1 \cdot \alpha + g_2 \cdot \alpha^2 + g_3 \cdot \alpha^3$$

Montgomery Architecture

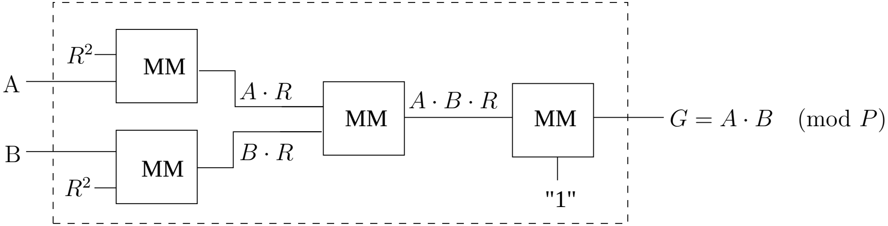


Figure: *Montgomery* multiplier over GF(2^k)

Montgomery Multiply: $F = A \cdot B \cdot R^{-1}$, $R = \alpha^k$

- Barrett architectures do not require precomputed R^{-1}
- We can verify 163-bit circuits, and also catch bugs!
- Conventional techniques fail beyond 16-bit circuits

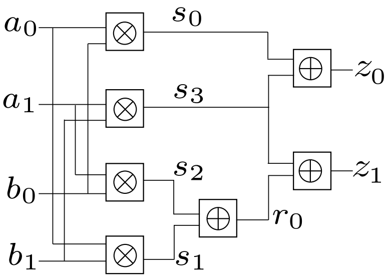
Verification: The Mathematical Problem

- 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: $A = \{a_0, \dots, a_{k-1}\}, B = \{b_0, \dots, b_{k-1}\}$
  - Primary Output $Z = \{z_0, \dots, z_{k-1}\}$
  - $A = a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_{k-1}\alpha^{k-1}$
  - $B = b_0 + b_1\alpha + \cdots + b_{k-1}\alpha^{k-1}, \; Z = z_0 + z_1\alpha + \cdots + z_{k-1}$
- Does the circuit C correctly compute specification f?

Mathematically:

- Model the circuit (gates) as polynomials $\{f_1, \dots, f_s\} \in \mathbb{F}_{2^k}[x_1, \dots, x_d]$
- Do solutions to $f = 0$ (spec) agree with solutions to $f_1 = f_2 = \cdots = f_s = 0$ (implement)?
- Does f vanish on the Variety $V(f_1, \dots, f_s)$?

Example Formulation



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

$$
\begin{aligned}
z_0 = s_0 + s_3 \quad &\Longrightarrow \quad f_1 : z_0 + s_0 + s_3 \\
s_0 = a_0 \cdot b_0 \quad &\Longrightarrow \quad f_2 : s_0 + a_0 \cdot b_0 \\
&\vdots
\end{aligned}
$$

$$A + a_0 + a_1\alpha, \quad B + b_0 + b_1\alpha, \quad Z + z_0 + z_1\alpha$$

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

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

Definition

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

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

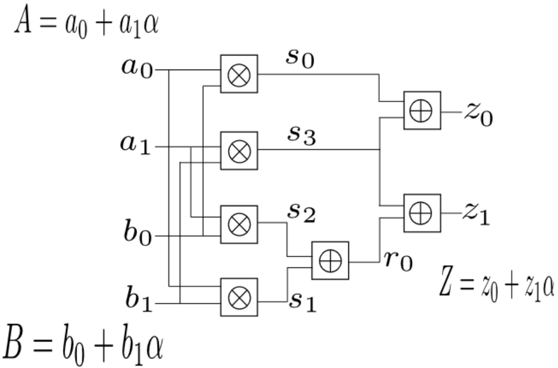
$J = \langle f_1, f_2 \ldots, f_s \rangle$ is an ideal generated by $f_1, \ldots, f_s$ and the polynomials are called the generators.

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 $f \in J$.

If f is a member of the ideal $J = \langle f_1, f_2 \ldots, f_s \rangle$, then f agrees with solutions of $f_1 = \cdots = f_s = 0$.

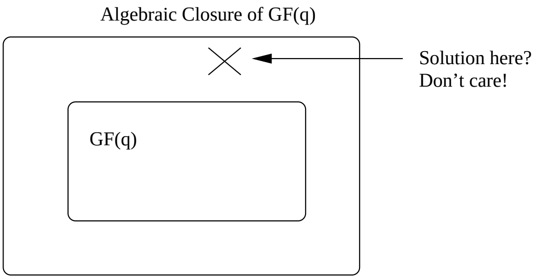
Example Formulation



- Polynomials for all the gates: $f_1, \ldots, f_s$; ideal $J = \langle f_1, \ldots, f_s \rangle$
- Polynomial specification: $f : Z = A \times B$
- Spec f "agrees with" all solutions to $f_1 = \cdots = f_s = 0$
- f vanishes on variety $V_{\mathbb{F}_{2^k}}(J)$?

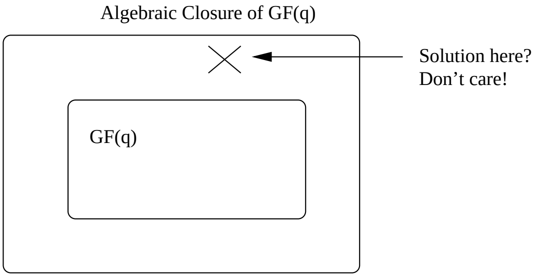
Variety over $\mathbb{F}_q$ or over $\overline{\mathbb{F}_q}$?

Where are the solutions of $f_1 = f_2 = \cdots = f_s = 0$?



- Algebraically closed field $\mathbb{F}$: Every $f(x) \in \mathbb{F}[x]$ has a root in $\mathbb{F}$
- Galois fields are not algebraically closed!
- $\overline{\mathbb{F}_q}$ = algebraic closure of $\mathbb{F}_q$: $\overline{\mathbb{F}_q} \supset \mathbb{F}_q$
    - Similar to $\mathbb{C} \supset \mathbb{R}$

Restricting the Variety to $\mathbb{F}_q$?



- Property of Finite fields: $\forall x \in \mathbb{F}_q, x^q - x = 0$
- **Vanishing Polynomials:** $x^q - x$ are vanishing polynomials of $\mathbb{F}_q$
- Therefore $V(x^q - x) = \mathbb{F}_q$
- Restrict the solutions to $\mathbb{F}_q$:
  $V_{\mathbb{F}_q}(f_1, \dots, f_s) = V_{\overline{\mathbb{F}_q}}(f_1, \dots, f_s, \quad x_1^q - x_1, \dots, x_n^q - x_n)$

Our Problem Formulation (Finally!)

Given f (spec) and $f_1, \ldots, f_s$ (circuit) over $\mathbb{F}_q[x_1, \ldots, x_n]$

- $J = \langle f_1, f_2 \ldots, f_s \rangle$, Polynomials from the design
- $J_0 = \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$, Vanishing polynomials generated
- $J + J_0 = \langle f_1, f_2 \ldots, f_s, \;\; x_1^q - x_1, \ldots, x_n^q - x_n \rangle$; Variety $V(J + J_0) =$ circuit configuration
- Is $f \in J + J_0$? If so, the circuit is correct. Otherwise there is a bug.
- This result is derived from Strong Nullstellensatz over $\mathbb{F}_q$

If f vanishes on $V_{\mathbb{F}_q}(J)$, then:

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

Our problem: Test of $f \in (J + J_0)$

Requires the computation of a Gröbner basis of $J + J_0$

Ideal Membership Test Requires a Gröbner Basis

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

Given $F = \{f_1, f_2, \cdots, f_s\}$, Compute a Gröbner Basis $G = \{g_1, g_2, \cdots, g_t\}$, such that $I = \langle F \rangle = \langle G \rangle$

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

Grobner Basis G decides ideal membership:

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

Buchberger's Algorithm

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

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

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

- For $J \subset \mathbb{F}_q[x_1, \ldots, x_n]$, Complexity $GB(J + J_0) : q^{O(n)}$
- GB complexity very sensitive to **term ordering**
- A term order has to be imposed for systematic polynomial computation

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

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

Recall, S-polynomial depends on term ordering:

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

Effect of Term Orderings

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

LEX: $x_0 > x_1 > x_2 > x_3$

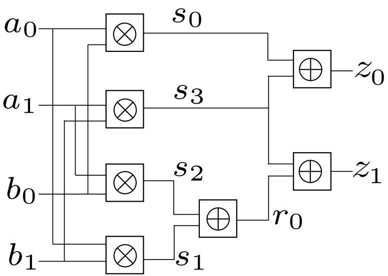
- $f = x_0x_1 + x_2$, $g = x_1x_2 + x_3$
- $lm(f) = x_0x_1$; $\quad lm(g) = x_1x_2$
- $S(f,g) \xrightarrow{G'}_+ x_0x_3 + x_2^2$

LEX: $x_3 > x_2 > x_1 > x_0$

- $f = x_2 + x_0x_1$, $g = x_3 + x_1x_2$
- $lm(f) = x_2$; $\quad lm(g) = x_3$, $S(f,g) \xrightarrow{G'}_+ 0$

Problem: Find a "term order" that makes ALL $\{lm(f),\ lm(g)\}$ relatively prime.

For Circuits, such an order can be derived



$$f_1 : s_0 + a_0 \cdot b_0,\ lm = s_0; \quad f_2 : s_1 + a_0 \cdot b_1,\ lm = s_1$$
$$f_3 : s_2 + a_1 \cdot b_0,\ lm = s_2; \quad f_4 : s_3 + a_1 \cdot b_1,\ lm = s_3$$
$$f_5 : r_0 + s_1 + s_2,\ lm = r_0; \quad f_6 : z_0 + s_0 + s_3,\ lm = z_0$$
$$f_7 : z_1 + r_0 + s_3,\ lm = z_1$$

- Reverse Topological Traversal of the Circuit
- $\{z_0 > z_1\} > \{r_0 > s_0 > s_3\} > \{s_1 > s_2\} > \{a_0 > a_1 > b_0 > b_1\}$
- Make every gate output a leading term

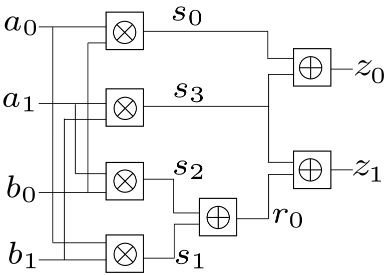
Using Our Topological Term Order:

- $F = \{f_1, \dots, f_s\}$ 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: $f_i = x_i + P$
- Only pairs to consider: $S(f_i, \quad x_i^q - x_i)$ in Buchberger's Algorithm:

$$S(f_i, \quad x_i^q - x_i) \xrightarrow{J}_+ P^q - P \xrightarrow{J_0}_+ 0$$

Conclusion: Our term order makes $\{f_1, \dots, f_s, x_1^q - x_1, \dots, x_n^q - x_n\}$ a Gröbner Basis

Our Term Order: Already a Gröbner basis



$$f_1 : s_0 + a_0 \cdot b_0, \; lm = s_0; \quad s_0^q - s_0$$

$$f_2 : s_2 + a_1 \cdot b_0, \; lm = s_2; \quad s_2^2 - s_2$$

- Every gate: $f_i : x_i + P \in J$
- Every vanishing polynomial: $x_i^q - x_i \in J_0$

$S(f_i, \quad x_i^q - x_i) \xrightarrow{J}_+ P^q - P \xrightarrow{J_0}_+ 0$
$\{f_1, \ldots, f_s, \quad x_1^q - x_1, \ldots, x_n^q - x_n\}$ is a Gröbner basis

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

Experimental Results: Correctness Proof

Table: Verification Results of SAT, SMT, BDD, ABC.

	Word size of the operands k-bits		
Solver	8	12	16
MiniSAT	22.55	*TO*	*TO*
CryptoMiniSAT	7.17	16082.40	*TO*
PrecoSAT	7.94	*TO*	*TO*
PicoSAT	14.85	*TO*	*TO*
Yices	10.48	*TO*	*TO*
Beaver	6.31	*TO*	*TO*
CVC	*TO*	*TO*	*TO*
Z3	85.46	*TO*	*TO*
Boolector	5.03	*TO*	*TO*
Sonolar	46.73	*TO*	*TO*
SimplifyingSTP	14.66	*TO*	*TO*
ABC	242.78	*TO*	*TO*
BDD	0.10	14.14	1899.69

Table: Verify bug-free and buggy Mastrovito multipliers. SINGULAR computer algebra tool used for division.

Size k-bits	32	64	96	128	160	163
#variables	1155	4355	9603	16899	26243	27224
#polynomials	1091	4227	9411	16643	25923	26989
#terms	7169	28673	64513	114689	179201	185984
Compute-GB:	93.80	*MO*	*MO*	*MO*	*MO*	*MO*
Ours: Bug-free	1.41	112.13	758.82	3054	9361	16170
Ours: Bugs	1.43	114.86	788.65	3061	9384	16368

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 $f \xrightarrow{f_1,\dots,f_s}_{+} r$
- Find a polynomial f_i that divides f, or "cancels" $LT(f)$
- $Z = z_0 + z_1\alpha, \quad A = a_0 + a_1\alpha, \quad B = b_0 + b_1\alpha$
- Construct a matrix: rows = polynomials, columns = monomials, entries = coefficient of monomial present in the polynomial

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

F_4-style reduction

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

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

F_4-style reduction

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

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

F_4-style reduction

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

$$
\begin{array}{ccccccccccc}
Z & AB & Ba_0 & Ba_1 & z_0 & z_1 & r_0 & a_0b_0 & a_0b_1 & a_1b_0 & a_1b_1
\end{array}
$$

$$
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & \alpha & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & \alpha & 1 & \alpha & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \alpha & 1 & \alpha & 0 & 1 & \alpha & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & \alpha & 0 & 1 & \alpha & \alpha & \alpha^2 \\
0 & 0 & 0 & 0 & 0 & \alpha & 0 & 0 & \alpha & \alpha & \alpha^2+1 \\
0 & 0 & 0 & 0 & 0 & 0 & \alpha & 0 & \alpha & \alpha & \alpha^2+\alpha+1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha^2+\alpha+1
\end{pmatrix}
$$

Results

Table: Runtime for verifying bug-free and buggy Montgomery multipliers. TO = timeout of 10hrs. Time is given in seconds. $*$ denotes SINGULAR's capacity exceeded.

Operand size k	32	48	64	96	128	163
#variables	1194	2280	4395	6562	14122	91246
#polynomials	1130	2184	4267	6370	13866	89917
#terms	10741	18199	40021	55512	134887	484738
Bug-free (Singular)	1.50	11.03	27.70	1802.75	10919	$*$
Bug-free (F_4)	0.86	4.47	10.11	700.59	4539	18374
Bugs (Singular)	1.52	11.10	28.18	1812.15	11047	$*$
Bugs (F_4)	0.88	4.49	10.12	709.03	4564	17803

F_4-style reduction 2.5X faster than use of Singular

Prior Work

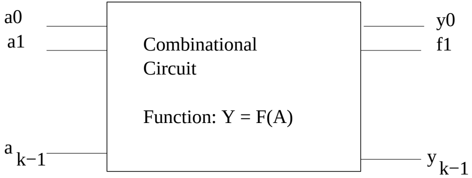
Wienand *et al* CAV'2008: Similar approach for verification of integer multipliers

- Works over rings $\mathbb{Z}_{2^k}$
- They derive the same term order: $f \xrightarrow{F}_+ g$
- Then the circuit is correct if g is a *vanishing polynomial*; $g \in F_0$ over $\mathbb{Z}_{2^k}$
- But they do not investigate if F, F_0 is a Gröbner basis....

Mukopadhyaya, TCAD 2007 ($<$ 16-bit circuits), our own approach VLSI Design 2012, other theorem proving papers....

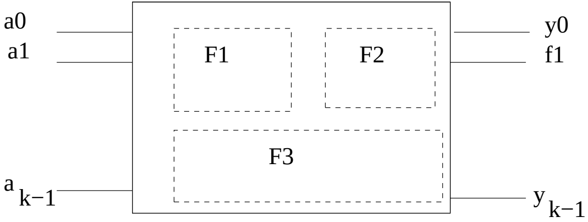
BLUVERI from IBM, *A. Lvov, et al.*, FMCAD 2012.

Polynomial Interpolation from Circuits



- Circuit: $f : \mathbb{B}^k \rightarrow \mathbb{B}^k$
- $f : \mathbb{Z}_{2^k} \rightarrow \mathbb{Z}_{2^k}$ or $f : \mathbb{Z}_{2^k} \rightarrow \mathbb{Z}_{2^k}$
- Interpolate a polynomial from the circuit: $Y = F(A)$
- $A = a_0 + a_1\alpha + \dots a_{k-1}\alpha^{k-1}, \quad Y = y_0 + y_1\alpha + \dots y_{k-1}\alpha^{k-1}$
- Compute Gröbner basis of circuit polynomials with Elimination order: circuit-variables $> Y > A$
- Obtain $Y = F(A)$ as a unique, canonical, polynomial representation from the circuit

Polynomial Interpolation from Circuits



Hierarchical Interpolation

- Partition the circuit into sub-circuits
- Interpolate Polynomials $F_1, F_2, \ldots$ from Partitions
- Re-compute Gröbner basis of $\{F_1, F_2, \ldots\}$
- Eliminate internal variables to obtain $Y = F(A)$

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 upto 163-bit circuits
  - NIST specified 163-bit field.... practical verification!
- Our approach relies only on polynomial division
- Complexity of polynomial division: Polynomial in the size of $f_1, \ldots, f_s$
- Almost the same time to catch bugs
- Conventional approaches fail miserably.....
- Future Work: Verify sequential GF-arithmetic Circuits
  - State-space traversal: Quantifier Elimination over Gröbner Basis